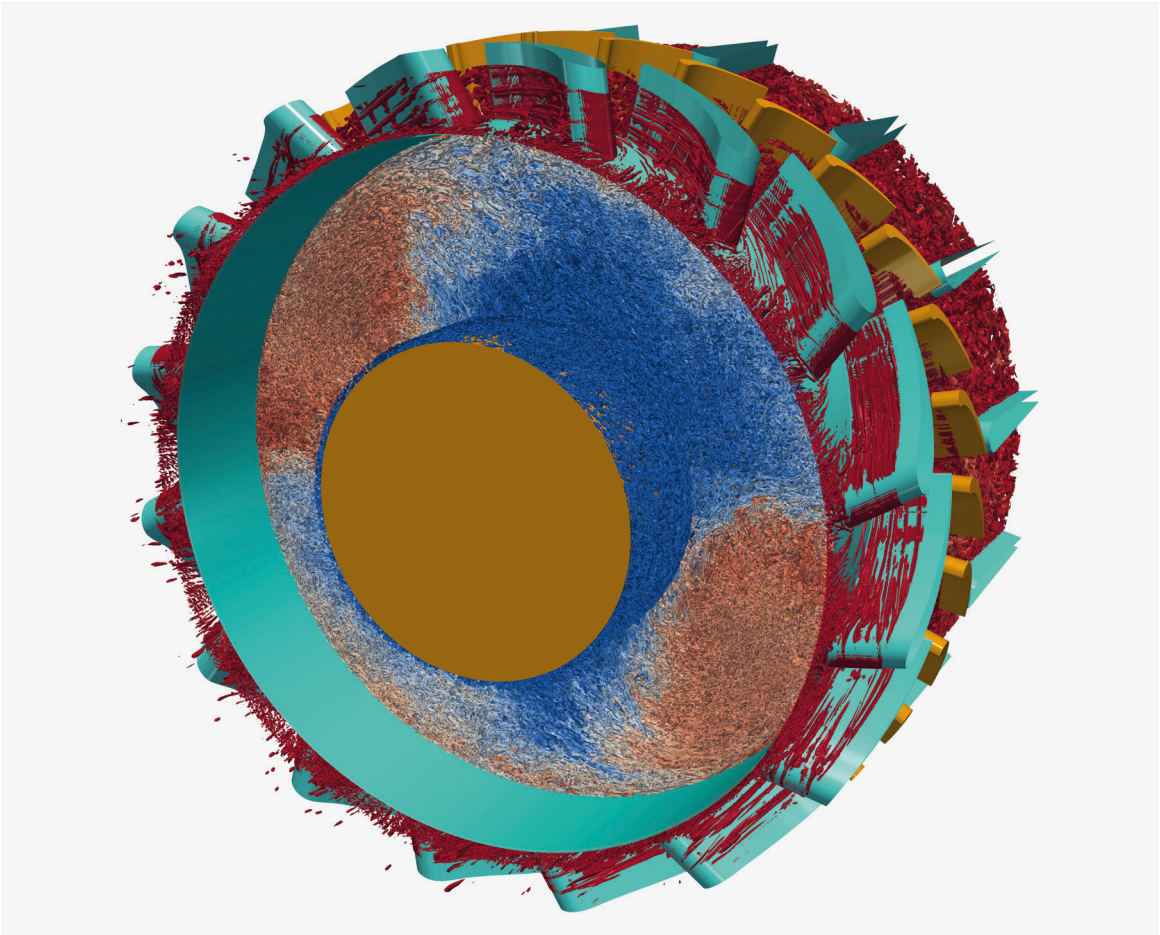


Proceedings of the 35th Parallel Computational Fluid Dynamics International Conference 2024

Andreas Lintermann, Sohel Sebastian Herff and Jens Henrik Göbbert

IAS Series
Band / Volume 69
ISBN 978-3-95806-819-3

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Schriften des Forschungszentrums Jülich
IAS Series Band / Volume 69

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Proceedings of the 35th Parallel Computational Fluid Dynamics International Conference 2024

Andreas Lintermann,
Sohel Sebastian Herff
and Jens Henrik Göbbert

Schriften des Forschungszentrums Jülich
IAS Series Band / Volume 69

ISSN 1868-8489 ISBN 978-3-95806-819-3

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2025

Schriften des Forschungszentrums Jülich
IAS Series, Band / Volume 69

ISSN 1868-8489
ISBN 978-3-95806-819-3

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess.

Proceedings of the 35th Parallel Computational Fluid Dynamics International Conference 2024

September 02.-04., 2024, University Club Bonn, Germany

Preface

The world of fluids, which can be gases or liquids, fascinates mankind already for centuries. Studies reach back to the antiquity and prominent figures of sciences, such as Archimedes, Leonardo da Vinci, Benedetto Castelli, Evangelista Torricelli, or Blaise Pascal were early pioneers in researching the behavior of fluids under various conditions. Later, in the 18th century, Isaac Newton, Daniel Bernoulli, and Leonhard Euler contributed to a more sophisticated understanding of the involved physics. They preceded Claude-Louis Navier and George Gabriel Stokes, who progressively developed the Navier–Stokes equations, partial differential equations describing the motion of viscous fluids, over several decades in the 19th century. The Navier-Stokes equations, given by the conservation of mass, momentum, and energy, in all their beauty, build the foundation of nowadays experimental and numerical methods to investigate, for example, laminar, transitional, or turbulent, gaseous or liquid flows, aeroacoustics, species or particle transport, fluid-structure interaction, or combustion processes.

The Navier-Stokes equations are defined on continuous space. With the lack of understanding in developing numerical methods to solve such partial differential equations it took until the 1920s until Lewis Fry Richardson made fundamental developments in approximating corresponding solutions with numerics. As a meteorologist, he introduced the finite difference (FD) method in 1928 to calculate the fluid dynamics for weather prediction, and with this provided the tool set to go from continuous to discrete space. In this, his work was groundbreaking for the field of computational fluid dynamics (CFD). Subsequently, in the 1940s, John von Neumann and Robert D. Richtmyer contributed to the stability analysis of numerical schemes, addressing key challenges in the numerical solution of fluid dynamics problems. Particularly considering irregular geometries, breakthroughs were made by Richard Courant, who developed the finite element (FE) method in the 1950s. With the 1960s, the first CFD software became available, making the field of numerical fluid mechanics more practical. This gave rise to new excitement and more and more people became fascinated by CFD. From this momentum rose the finite volume (FV) method, which was developed by McDonald, Mac-Cormack, and Paullay in 1971 and 1972. While the FD method uses nodal representations, the FV method discretizes the Navier-Stokes equations in their integral form, and is a generalization of the FD method. The discontinuous Galerkin (DG) method was developed in the 1970s and combined features of both the FE and FV methods. Also the smoothed-particle hydrodynamics (SPH) method, which is a meshfree Lagrangian method, was developed in this century by Gingold, Monaghan, and Lucy. The 1980s were governed by the development of turbulence models, e.g., Launder and Spalding developed the k-ϵ model. Furthermore, the lattice-Boltzmann (LB) method was born, which takes another approach to predicting fluid dynamics than the aforementioned numerical meth-

ods. In the LB method, the Boltzmann equation, an integro-differential equation describing the statistical behavior of particles, is solved in its discretized form. Also the spectral element (SE) method, which couples the tensor product efficiency of global spectral methods with the geometric flexibility of finite elements, was invented by Patera and his peers in the 1980s. The nowadays existing methods are all derivatives of the advancements made in the 20th century, with a strong trend towards tackling coupled multi-physics multi-scale problems.

The early work of Richardson was at that time not adoptable to practical applications. The computational power to transition from coarse toy problems to more complex real-life scenarios was simply not available. This changed in the 1950s and 1960s, when digital computers with sufficient power became available, and John von Neumann and Stanislaw Ulam started studying fluid behavior with them. Over decades the computational power continuously increased and the first Cray machines with scalar and vector processors appeared in the 1970s with further advancements in scalability and speed into the 1980s. Increased computational power was achieved through higher clocking, larger vector units, and shared memory multiprocessing. While an increase in the clocking frequency in general decreased the time to solution, more powerful vector units were able to perform single floating point operations on larger and larger datasets, and shared memory multiprocessing enabled multiple processors to access a joint memory space. Also in the 1980s, the jump to distributed memory computing was made with multiple processors not sharing anymore the same memory space, but requiring communication via messages - the concept of cluster computing was born. Despite vector machines enjoyed popularity and are still being used today for specific applications, the trend in supercomputing transitioned to distributed computing with a strong boost across the the 1980s and 1990s towards the 2000s. The Intel iPSC Hypercube machine from 1985 was succeeded by the CM-1, which was a massively parallel computer with 65,536 single instruction multiple data (SIMD) processing elements and later by the the CM-5, a multiple instructions multiple data (MIMD) machine, and the Beowulf cluster. The latter was developed by Donald Becker and Tom Stirling at NASA, and connected normal personal computers (PCs) via a network. Soon it became clear, than an increase of the clocking speed is limited by physical constraints, especially by the energy consumption and the heat production. More computational power was hence achieved by increasing the number of processor cores on a single chip as well as in the whole cluster by increasing the number of nodes. This trend, as can be seen from the Top500 list[1], has not stopped and high-performance computing (HPC) centers around the globe compete in the the challenge to host the most powerful and energy efficient supercomputers.

With the appearance of different computing architectures, e.g., graphics processing units (GPUs), that were initially intended for graphics work, or field programmable gate array (FPGA) devices, the idea of HPC systems uniting different architectures under one roof – the modular supercomputing architecture (MSA) – was born. Especially GPUs with their potential to scale up frequently used operations in many simulation codes, such as vector/vector, vector/matrix, and matrix/matrix operations, became prominent. This trend is these days massively fueled by the need for accelerated training of artificial intelligence (AI) technologies, showing a strong shift towards a higher share of GPUs over central processing unit (CPU)-based systems in nowadays MSA systems. As of today, we are in the Exascale era, where a single HPC system

[1] Top500 list https://top500.org

is capable of performing more than 10^{18} floating point operations per second. Through all the generations, the transistor density has increased, coming to a foreseen physical limit. This is why physicists and computer architects thought about novel disruptive technologies for computing. This led to the invention of quantum devices, e.g., to gate-based quantum computers and analogue quantum annealers, requiring a complete rethinking of algorithm design. These devices are still being advanced and researched, offering potential to efficiently tackle NP-hard problems.

Machine learning (ML) is around for many decades with fundamental work already made in the 1940s and 1950s. However, the deep learning (DL) era, where DL as an ML technique started to be applied to a wider set of problems and also started to require increased compute power did not start before 2010. It was not before 2012 that GPUs were used to train ML models with a limited use of this architecture (up to 8 GPUs) until 2014. A large-scale era, where up to 100 GPUs were used followed until 2016. Since then, also with the advent of large language models (LLMs), massive investments and developments in AI training and inference at scale have been made. The general trend across all disciplines goes towards training multi-modal foundation models with a subsequent refinement to specific tasks. These developments have not stopped at the field of CFD. In the last decade, major advancements have been made in predicting fluid mechanics with neural networks. Surrogate modeling, integrating complex physics predicted by DL models into large-scale simulations have been developed. AI-based closure and subgrid-scale (SGS) models trained on highly-resolved simulation data, and optimization algorithms based on reinforcement learning (RL) techniques, are available. Training and inferencing at scale for CFD problems using MSAs has become possible and moves – similar to other disciplines – towards advanced AI with a sophisticated and deepened knowledge about the underlying physics.

The *Parallel CFD International Conference* (*ParCFD*), which was inaugurated in 1989 in Manhattan Beach, California, has always picked up the latest trends in CFD and computing. Since

Figure 1: Locations of the *ParCFD International Conference* since 1989.

its first event, the *ParCFD* took place every year (except 2020 due to the COVID-19 pandemic) and traveled around the world, i.e., it was held in the USA, Germany, France, Japan, Italy, UK, Taiwan, Norway, The Netherlands, Russia, Spain, South Korea, Turkey, China, and Ecuador, see Fig. 1. The latest *35th Parallel CFD International Conference 2024* took place in Bonn, Germany, from Sep. 02-04., 2024. It brought together experts from different disciplines in CFD, e.g., from aerospace, combustion, or biomedical engineering, HPC and quantum specialists, algorithm, workflow, and tool developers, industry, and AI experts. In total, the Conference had 108 participants from 21 countries, and 84 presentations in nine mini-symposia and other topics sessions in addition to five keynote presentations were given.

This proceedings document holds 67 scientific publications from the nine mini-symposia and from four sessions on other topics. All papers have undergone an iterative single-blind peer-review process to guarantee novelty and a high-quality.

With this, we wish the audience of this proceedings document an interesting and insightful reading and look forward to the next *ParCFD International Conference* in 2025.

The Editors and Organizers of *ParCFD 2024*,

Dr.-Ing. Andreas Lintermann Dr.-Ing. Sohel Sebastian Herff Jens Henrik Göbbert

Sponsors

We thank our sponsors for supporting us in organizing the *35th Parallel CFD International Conference 2024* in Bonn, Germany, from Sep. 02-04., 2024. Only with their contribution, we were able to prepare and host such a successful and memorable event.

Acknowledgements

Our special thanks go to our keynote speakers Lennart Schneiders (SIEMENS Digital Industries Software), Christian Hasse (Technical University Darmstadt, Simulation of reactive Thermo-Fluid Systems), Niclas Jansson (KTH Royal Institute of Technology, PDC Center for High Performance Computing), Ricardo Vinuesa Motilva (KTH Royal Institute of Technology, School of Engineering Sciences, Teknisk Mekanik, Fluid Mechanics), and Linda Gesenhues (EuroHPC Joint Undertaking). In addition, our thanks go to all mini-symposium organizers, who were able to gather many contributors for their sessions, helped in reviewing the submissions, and with this contributed to the success of the conference with high-quality submissions and presentations.

For the local organization, we received fantastic support from Beate Schmitz and Sylvie Lemke from the Corporate Communication Department (UK-M) of Forschungszentrum Jülich, which we also gratefully acknowledge. In Bonn, we were supported by Marcel Aach, Dirk Baker, Jonathan Windgassen, Luis Cifuentes, and Astrid Schwaiger from the Jülich Supercomputing Centre of Forschungszentrum Jülich, who did an amazing job in taking care of technical matters in all sessions and the registration desk. In addition, we also thank Oleksandr Krochak, also from the Jülich Supercomputing Centre, for his support in assembling the Proceedings document.

Furthermore, we acknowledge the advise of the previous *ParCFD* organizers, specifically Stefano Rolfo from the Science and Technology Facilities Council, UK, as well as from several members of the *ParCFD* International Scientific Committee, i.e., Hasan Akay from Atilim University, Turkey, and Guillaume Houzeaux, Barcelona Supercomputing Center, Spain.

Last but not least, we gratefully acknowledge the Gauss Centre for Supercomputing e.V. (GCS)[2] and the European Center of Excellence in Exascale Computing "Research in AI- and Simulation-Based Engineering" (CoE RAISE)[3] for their support. The latter received funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

Funded by the European Union

RAISE Center of Excellence

GCS Gauss Centre for Supercomputing

JÜLICH Forschungszentrum

[2]GCS https://www.gauss-centre.eu
[3]CoE RAISE https://www.coe-raise.eu

Contents

Invited Speakers and Keynotes

Lennart Schneiders
SIEMENS Digital Industries Software

Lennart Schneiders is a Software Engineer and researcher at Siemens Digital Industries Software. He received his PhD from RWTH Aachen University in 2017. Until 2022, he was a postdoctoral researcher at RWTH Aachen, Jülich Aachen Research Alliance, and California Institute of Technology. Lennart is currently a developer of the multiphysics CFD software Simcenter STAR-CCM+. His research interests lie in numerical method development, turbulent multiphase flow, and high-performance computing.

Title: The intricacies of adaptive unstructured mesh refinement for industrial flows

Abstract. The simulation of industrial flows is associated with significant uncertainties arising from the quality of the computational mesh. In many industries, meshing therefore is considered an expert job. With the advances in parallel computing and GPU hardware reducing solver times, meshing can become a bottleneck in certain industrial workflows. While adaptive mesh refinement (AMR) is an intriguing approach to alleviate some of those problems, developing such a technique in a robust form is a challenge in its own.

In this talk, the intricacies of developing a general purpose AMR scheme are discussed. This includes the definition of generic solution-based refinement strategies. And it will be demonstrated that refining an unstructured mesh does not automatically guarantee lower truncation errors, but can even lead to the opposite.

Christian Hasse

Technical University Darmstadt, Simulation of reactive Thermo-Fluid Systems

Christian Hasse is full professor in the Department of Mechanical Engineering at Darmstadt University of Technology. He holds the chair of Simulation of reactive Thermo-Fluid Systems. He has supervised successfully more than 30 PhD students and currently 25 PhD students and post-docs are working in his group. He has more than 270 peer-reviewed publications and has served in multiple editorial boards and as associate/guest editor. He is organizer of scientific workshops and conferences focusing on sustainable combustion to achieve net zero emissions.

Prof. Hasse is elected Fellow of the Combustion Institute for his contributions on turbulent combustion, solid fuel combustion, multi-phase flow and soot formation. He has been elected to the Board of Directors of the International Combustion Institute in 2024. His main research interests are modeling and high-fidelity simulation of reactive and non-reactive flows, especially for CO2-free and CO2-neutral fuels such as hydrogen, ammonia, biomass, E-fuels and metals. In addition to fundamental studies on flame structures and dynamics, he also actively works on transferring these results to real-world applications including (aero-)engines, boilers and processes chemical engineering. For these topics, his group has developed a number of high-fidelity software applications that are deployed national Tier-2 and European Tier-0/1 supercomputers. In 2024 he has received an ERC Advanced Grant on aluminum steam combustion in which he aims to unravel the fundamental properties of pressurized Al-steam flames for the entire scientific chain, from single particles to turbulent flames with millions of particles, through a well-orchestrated combination of high-fidelity simulations, advanced modeling, and tailored experiments.

Title: How high-fidelity simulations of hydrogen combustion on supercomputers accelerate the energy transition

Abstract. Reactive Computational Fluid Dynamics (rCFD) has become an indispensable tool in fundamental and applied research. In addition, rCFD is used in industrial design, such as aero engine combustors or gas turbines. This success is based on the combination of decades of scientific model development, efficient numerics and ever-increasing computing power. This situation is about to change as both the energy system (1) and the HPC architecture (2) are undergoing disruptive changes.

First, the urgent need to shift from fossil fuels to renewable fuels such as hydrogen requires the redesign of energy conversion systems due to the completely different combustion charac-

teristics of renewable fuels. Combustion models for high-fidelity simulations are lacking and must be developed based on in-depth physical understanding. Second, the next generation of supercomputers will be mostly based on GPUs rather than CPUs. Efficient use of these systems will require a new class of CFD software with specialized numerics.

In this talk I will first introduce the role of rCFD in combustion system design before I highlight the role of hydrogen in the energy transition and how it differs from conventional fuels. After discussing the impact of GPU-based supercomputers on rCFD software development, I will present how GPU-based direct numerical simulations can unravel the complexities of hydrogen combustion.

In summary, in a rapidly changing environment, simulations on upcoming Exascale systems will provide physical insights that were deemed impossible just a few years ago. This will increase the impact of rCFD on the entire spectrum from fundamental science to industrial design of innovative systems.

The recording of this talk is available on YouTube.

Niclas Jansson

KTH Royal Institute of Technology, PDC Center for High Performance Computing

Niclas Jansson is a researcher at PDC Center for High Performance Computing at the KTH Royal Institute of Technology, Stockholm. He received his M.S. in computer science in 2008 and a PhD in numerical analysis in 2013 from KTH. Between 2013 and 2016, Niclas was a postdoctoral researcher at RIKEN Advanced Institute for Computational Science, where he was part of the application development team of the Japanese exascale program, Flagship 2020, focusing on developing extreme-scale multiphysics solvers for the K computer, and held a visiting scientist position at RIKEN between 2018 and 2021. He has extensive experience in extreme-scale computing as a developer of RIKEN's multiphysics framework CUBE, the HPC branch of FEniCS and the next-generation spectral element flow solver Neko, and is currently the coordinator of the EuroHPC Center of Excellence for Exascale CFD.

Title: Towards Exascale Simulations of Turbulent Flow

Abstract. Recent trends and advancements in including more diverse and heterogeneous hardware in High-Performance Computing (HPC) are challenging scientific software developers in their pursuit of efficient numerical methods with sustained performance across a diverse set of platforms. As a result, researchers are today forced to refactor their codes to leverage these powerful new heterogeneous systems. We present our work on addressing the extreme-scale computing challenges in computational fluid dynamics, ensuring exascale readiness of turbulence simulations. Focusing on Neko, a high-fidelity spectral element code, we outline the optimisation and algorithmic work necessary to ensure scalability and performance portability across a wide range of platforms. Finally, we present performance measurements on a wide range of accelerated computing platforms, including the EuroHPC pre-exascale system LUMI and Leonardo, where Neko achieves excellent parallel efficiency for an extreme-scale direct numerical simulation (DNS) of turbulent thermal convection using up to 80% of the entire LUMI supercomputer.

Ricardo Vinuesa Motilva

KTH Royal Institute of Technology, School of Engineering Sciences, Teknisk Mekanik, Fluid Mechanics

Dr. Ricardo Vinuesa is an Associate Professor at the Department of Engineering Mechanics, KTH Royal Institute of Technology in Stockholm. He is also Vice Director of the KTH Digitalization Platform and Lead Faculty at the KTH Climate Action Centre. He studied Mechanical Engineering at the Polytechnic University of Valencia (Spain), and he received his PhD in Mechanical and Aerospace Engineering from the Illinois Institute of Technology in Chicago. His research combines numerical simulations and data-driven methods to understand, control and predict complex wall-bounded turbulent flows, such as the boundary layers developing around wings and urban environments. Dr. Vinuesa has received, among others, an ERC Consolidator Grant, the TSFP Kasagi Award, the Goran Gustafsson Award for Young Researchers, the IIT Outstanding Young Alumnus Award, the SARES Young Researcher Award and he leads several large Horizon Europe projects. He is also a member of the Young Academy of Science of Spain.

Title: Explaining and controlling turbulent flows through deep learning

Abstract. In this presentation we first use a framework for deep-learning explainability to identify the most important Reynolds-stress (Q) events in a turbulent channel (simulated via DNS) and a turbulent boundary layer (obtained experimentally). This objective way to assess importance reveals that the most important Q events are not the ones with the highest Reynolds shear stress. This framework is also used to identify completely new coherent structures, and we find that the most important coherent regions in the flow only have an overlap of 70% with the classical Q events. In the second part of the presentation we use deep reinforcement learning (DRL) to discover completely new strategies of active flow control. We show that DRL applied to a blowing-and-suction scheme significantly outperforms the classical opposition control in a turbulent channel: while the former yields 30% drag reduction, the latter only 20%. We conclude that DRL has tremendous potential for drag reduction in a wide range of complex turbulent-flow configurations.

The recording of this talk is available on YouTube.

Linda Gesenhues

EuroHPC Joint Undertaking

Dr. Linda Gesenhues is a Programme Manager at the European High Performance Computing Joint Undertaking (EuroHPC JU). Linda has a longstanding interest in HPC and in particular has worked on research on High Performance Computing applications for computational mechanics and fluid dynamics.

Linda graduated from RWTH Aachen University with a Bachelor and Master in Mechanical Engineering. She then went on to complete her bi-national doctoral studies at the High Performance Computing Center at the Federal University of Rio de Janeiro in Brazil and the Chair for Computational Analysis of Technical Systems at RWTH Aachen University in Germany, researching on finite element simulations of geophysical flows. She continued her career as a research group leader at the Chair for Computational Analysis of Technical Systems focusing on massively parallel simulations of phase boundaries during melting processes. In 2022, Linda joined the Research&Innovation sector of the European High Performance Computing Joint Undertaking as a Programme Mananger for HPC applications, training and skills.

Title: The EuroHPC Joint Undertaking: Leading the Way in European Supercomputing

Abstract. In this presentation the European High Performance Computing Joint Undertaking (EuroHPC JU) will be introduced. The EuroHPC JU joins together the resources of the European Union, 31 European countries and 3 private partners to develop a World Class Supercomputing Ecosystem in Europe. Linda will present the operational EuroHPC supercomputers located across Europe and give details on how to access this infrastructure.

Linda will then present some of the JU's missions, such as the acquisition of new supercomputers including exascale systems and quantum computers, the implementation of an ambitious research and innovation programme and how to further strengthen Europe's leading position in HPC applications.

The recording of this talk is available on YouTube.

Mini-Symposium 1:

Quantum Computing for CFD Applications

Organizers: Matthias Möller, Julia Kowalski, Norbert Hosters, Rakesh Sarma, and Jaka Vodeb

Computational Fluid Dynamics (CFD) has been among the first disciplines to explore emerging compute technologies like vector processors in the 1990s and early 2000s and multi-core CPUs and GPUs since the mid-2000s to push the capabilities of simulation- based fluid flow analysis to their limits. Every new compute paradigm required a modernization of CFD codes and has brought a plethora of advanced methodologies to date ranging from classical grid-based approaches to particle and hybrid methods.

An emerging compute technology that promises to become a game-changer in the quest for ultimate compute power is quantum computing. In contrast to vector processors and multi-core CPUs and GPUs, quantum computing requires more than the adaption of established codes. The potential power of quantum computers will come from the strict exploitation of quantum mechanical effects such as superposition, entanglement, and quantum parallelism. This requires us to rethink the usefulness of established approaches such as grid-based Navier-Stokes solvers as a methodological base for future quantum-CFD applications.

The aim of this mini-symposium is to bring together pioneers and interested stakeholders in quantum-CFD to discuss recent advances in this young discipline. In particular, we welcome contributions in the fields of quantum Navier-Stokes methods, quantum Lattice Boltzmann methods, hybrid quantum-classical approaches, and quantum Machine Learning such as Quantum Gaussian Process emulation and hybrid workflows for Hyperparameter Optimization. Due to the early stage of the field, contributions can range from theoretical complexity analysis results to practical implementations of algorithms on quantum computers or their simulators. We also encourage newcomers in the field to pitch their ideas to stimulate open discussions and thereby contribute towards nurturing a stable quantum-CFD community.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Quantum Annealing Computations to Obtain Converged Flow Solutions

Yuichi Kuya[a,*], Takahito Asaga[a]

[a]Tohoku University, Department of Aerospace Engineering, 6-6-01 Aramaki Aza Aoba, Aobaku, Sendai, 980-8579 Miyagi, Japan

ARTICLE INFO[†]

Keywords:
Quantum Annealing;
Lattice Gas Automata;
Finite Difference

ABSTRACT

This study proposes numerical methods for flow computations to obtain a converged solution using quantum annealing. In general numerical simulations, a time integration or iterative method is required to obtain a converged solution. The proposed methods extract a converged solution by quantum annealing, making use of the quantum superposition state, which is one of the unique characteristics of quantum mechanics used in quantum annealing machines. The proposed algorithms are built for lattice gas automata (LGA) and finite difference methods. This paper mainly outlines the algorithm for LGA.

1. Introduction

Continuous progress in computers has expanded the range of computational fluid dynamics (CFD) applications over the years. However, due to the limitations in the performance of semiconductors used in current computers, such as processing technology and power consumption, the performance gains of current computers are expected to reach a plateau eventually.

Quantum computers, designed based on a different operating principle from existing computers, are expected to be one of the next-generation computers. Current quantum computers can be broadly divided into gate-based and annealing-based quantum computers. Gate-based quantum computers perform computations using quantum circuits incorporating gates corresponding to matrix calculations. Therefore, gate-based quantum computers are considered to be adaptable to a wide variety of applications by changing the combinations of gates (or matrices).

Quite a few flow computation algorithms for gate-based quantum computers were proposed by previous studies (e.g., [1, 2, 3]). One of the disadvantages of current quantum computers is that the number of available "qubits," which correspond to conventional computer bits, is still limited, and they are prone to induce computation errors. On the other hand, quantum annealing machines are designed to find the lowest energy states and are thus widely used for solving combinatorial optimization problems. The number of qubits available for quantum annealing machines is greater than for gate-based quantum computers; for example, the D-Wave Advantage has more than 5,000 qubits [4]. However, since quantum annealing machines can basically only find the lowest energy states of a cost function, few algorithms have been proposed for flow computations [5, 6].

This study proposes quantum annealing methods that obtain a converged flow solution by annealing computation, utilizing quantum superposition states. The proposed algorithms are built for lattice gas automata (LGA) and finite difference methods. This paper outlines the algorithm built for LGA.

2. Quantum annealing

Quantum annealing machines obtain the optimum combinations of a binary parameter $q_i \in \{0, 1\}$ or $\sigma_i \in \{-1, 1\}$ that minimizes a cost function formulated in the following QUBO (quadratic unconstrained binary optimization) model or Ising model:

QUBO model

$$E(\{q_i\}) = \sum_i \sum_j Q_{ij} q_i q_j, \tag{1}$$

Ising model

$$E(\{\sigma_i\}) = \sum_i h_i \sigma_i + \sum_{i<j} J_{ij} \sigma_i \sigma_j, \tag{2}$$

where $E(\{q_i\})$ and $E(\{\sigma_i\})$ are the cost functions, Q_{ij} is the QUBO matrix, h_i is the local bias acting on σ_i, and J_{ij} is the coupling constants between σ_i and σ_j, respectively. The QUBO and Ising models become equivalent by the following relation:

$$q_i = \frac{\sigma_i + 1}{2}. \tag{3}$$

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02435 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
kuya@tohoku.ac.jp (Y. Kuya)
ORCID(s): 0000-0002-3658-2992 (Y. Kuya); - (T. Asaga)

In this study, the proposed algorithm for LGA is expressed in the form of a QUBO model.

3. Proposed quantum annealing methods for LGA

In this section, the overview of LGA is given first. Then, the quantum annealing lattice gas automata (qaLGA) proposed by a recent study is briefly introduced. Finally, the proposed quantum annealing methods to obtain a converged flow solution is described.

3.1. Overview of LGA

LGA considers particles, which correspond to a set of molecules or atoms, and statistically reproduces flow physics by the collision and streaming processes. The following equation describes the time evolution of LGA:

$$q_i(t+1,\ \mathbf{r}+\mathbf{c}_i) = q_i(t,\ \mathbf{r}) + \Delta_i[q(t,\ \mathbf{r})], \tag{4}$$

where q_i describes the state of a lattice node at time t and position $\mathbf{r} = (x, y)$ and equals 1 when a particle having velocity in the i-direction exists and 0 otherwise. Δ_i is the collision operator. The well-known two-dimensional FHP models consist of particles of the same unit mass on a hexagonal lattice. Zeros and ones represent the presence of particles moving along the lattice lines. The FHP-I model considers six particle states, while the FHP-Ⅱ and FHP-Ⅲ models consider stationary particles (i.e., zero velocity) in addition to the six moving particles.

3.2. Quantum annealing lattice gas automata: qaLGA

Recently, Kuya et al. [6] proposed the following cost function, i.e., QUBO model, which reproduces the collision process of the FHP-Ⅲ model:

$$\begin{aligned} &E(\{q_i(t',\ \mathbf{r})\}) = \\ &\quad\left(-2C^{\rho}\sum_i q_i'q_i' + \sum_i\sum_j q_i'q_j'\right) \\ &+\left(-2C^{\rho u}\sum_i u_i\cdot q_i'q_i' + \sum_i\sum_j u_iu_j\cdot q_i'q_j'\right) \\ &+\left(-2C^{\rho v}\sum_i v_i\cdot q_i'q_i' + \sum_i\sum_j v_iv_j\cdot q_i'q_j'\right) \\ &+\lambda\sum_i q_iq_i'q_i', \end{aligned} \tag{5}$$

where q_i' denotes the particle state after the collision and before the propagation processes, and u_i and v_i are the velocity components in the x- and y-directions. Also, C^{ρ}, $C^{\rho u}$, and $C^{\rho v}$ are the sum of the conservative variables at each lattice node:

$$\begin{aligned} &\text{Mass:}\\ &\textstyle\sum_i x_i(t,\ \mathbf{r}) = C^{\rho}(t,\ \mathbf{r}),\\ &\text{Momentum } (x\text{-direction}):\\ &\textstyle\sum_i u_ix_i(t,\ \mathbf{r}) = C^{\rho u}(t,\ \mathbf{r}),\\ &\text{Momentum } (y\text{-direction}):\\ &\textstyle\sum_i v_ix_i(t,\ \mathbf{r}) = C^{\rho v}(t,\ \mathbf{r}). \end{aligned} \tag{6}$$

The first to third terms on the right-hand side of this equation correspond to the mass, x-momentum, and y-momentum conservation equations, which must be satisfied through the collision process, respectively. The fourth term leads to the post-collision state different from the pre-collision state, and the coefficient λ adjusts the strength of this constraint term. If only the first to third terms in Eq. 5 are solved by quantum annealing, the same pre-collision state may also be obtained as the post-collision state since both the pre-collision state also satisfies the conservation equations. The fourth term increases the cost function when the post-collision state becomes the same as the pre-collision state. A detailed description of this QUBO model can be found in Ref. [6].

3.3. Obtaining a converged flow solution using quantum annealing by LGA

By extending the idea of qaLGA described above, this study proposes a QUBO model to obtain a converged flow solution by quantum annealing, utilizing the quantum superposition state. The proposed QUBO model for LGA consists of the sub-cost functions as follows:

$$\begin{aligned} E(\{q\}) = E_T(\{q,q'\}) + E_\Omega(\{q,q'\}) \\ +E_w(\{q,q'\}) + E_o(\{q\}), \end{aligned} \tag{7}$$

where E_T, E_Ω, E_w, and E_o are the sub-cost functions given to satisfy the convergence condition, collision rules, boundary conditions, and flow field conditions. For example, the sub-cost function of the collision is given by

$$\begin{aligned} E_\Omega = \sum_{\mathbf{r}\neq\mathbf{r}_{wall}}\Big[&\left(\sum_i\sum_j q_i'q_j' - \sum_i\sum_j q_i'q_j + \sum_i\sum_j q_iq_j\right) \\ &+\left(\sum_i\sum_j u_iu_jq_i'q_j' - \sum_i\sum_j u_iu_jq_i'q_j + \sum_i\sum_j u_iu_jq_iq_j\right) \\ &+\left(\sum_i\sum_j v_iv_jq_i'q_j' - \sum_i\sum_j v_iv_jq_i'q_j + \sum_i\sum_j v_iv_jq_iq_j\right) \\ &+\lambda_\omega\textstyle\sum_i q_i'q_i\Big], \end{aligned} \tag{8}$$

where Eq. (5) is extended to a whole computational domain except for the wall boundaries.

Figure 1 compares the ρu distribution of a channel flow obtained by the proposed quantum annealing algorithm with

a conventional LGA solution. In conventional LGA computation, the converged solution is obtained using a standard time-advancing method from an initial condition with constant velocity. In contrast, the proposed algorithm obtains the converged solution by quantum annealing with several sampling processes.

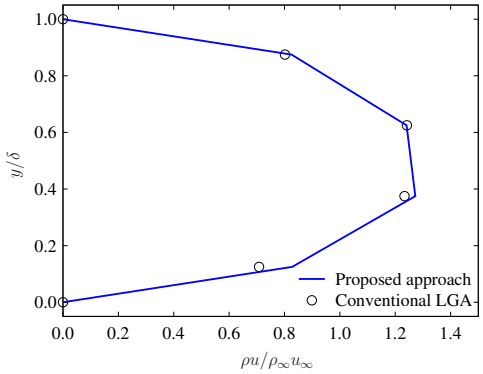


Figure 1: ρu distributions of a channel flow obtained by proposed algorithm and conventional LGA.

4. Conclusions

This paper has discussed novel numerical methods to obtain a converged flow solution using quantum annealing. The algorithms are built for lattice gas automata (LGA) and finite difference methods, but this paper only outlines the one for LGA. The proposed QUBO model built for LGA consists of sub-cost functions, which are the constraints imposed to satisfy the convergence condition, collision rules, boundary conditions, and flow field conditions. In a numerical test, the ρu distribution of a channel flow is obtained by solving the proposed QUBO model using quantum annealing, and the obtained solution is in good agreement with that obtained by the conventional LGA.

References

[1] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, Physical Review E 63 (2001) 046702. doi:10.1103/PhysRevE.63.046702.

[2] F. Gaitan, Finding flows of a navier–stokes fluid through quantum computing, npj Quantum Information 6 (1) (2020). doi:10.1038/s41534-020-00291-0.

[3] F. Gaitan, Finding solutions of the navier-stokes equations through quantum computing—recent progress, a generalization, and next steps forward, Advanced Quantum Technologies 4 (10) (Sep. 2021). doi:10.1002/qute.202100055.

[4] D-Wave Systems, Advantage (https://www.dwavesys.com/solutions-and-products/systems, accessed 2024-01-29).

[5] N. Ray, T. Banerjee, B. Nadiga, S. Karra, On the viability of quantum annealers to solve fluid flows, Frontiers in Mechanical Engineering 8 (2022). doi:10.3389/fmech.2022.906696.

[6] Y. Kuya, K. Komatsu, K. Yonaga, H. Kobayashi, Quantum annealing-based algorithm for lattice gas automata, Computers and Fluids 274 (2024) 106238. doi:10.1016/j.compfluid.2024.106238.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Strategies for the Application of Quantum Computers in Computational Fluid Dynamics

Stefan Görtz[a,*], Stefan Langer[a]

[a]German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Lilienthalplatz 7, 38108 Braunschweig, Germany

ARTICLE INFO[†]

Keywords:
Machine Learning for Computational Fluid Dynamics;
Quantum Computing;
Physics-Informed Neural Networks;
Physics-Informed Quantum Circuits

ABSTRACT

A three-year DLR project entitled "Machine Learning and Quantum Computing – Digitalization of Aircraft Development 2.0", was established in spring 2021 with the goal to investigate whether and how high-fidelity aerodynamic simulations can be carried out using innovative methods from the field of machine learning and in which way these methods can also be implemented on quantum computers.

1. Introduction

The aviation industry faces the challenge of having to make significant contributions to achieving the ambitious global climate and environmental targets. To achieve this, future aircraft must consume significantly less fuel than today's or use more environmentally friendly engines, and also need to be quieter, especially during take-off and landing. In order to be able to assess these properties as early as possible in the development of a new aircraft, a very large number of computer simulations are required, which are barely feasible even on today's high-performance computers.

Accelerating these simulations, thanks to the use of quantum computers, promises to close this gap in the future. Unexpected characteristics of an aircraft that would only become evident later in flight testing can be uncovered in advance through a large number of scale-resolving simulations and remedied during the design process. As a vision, a complete flight from take-off to landing could be simulated on a quantum computer long before the real first flight, thus creating an accurate image of the aircraft and its environmental effects in advance. At DLR, these topics are addressed by the virtual product which has a central and important meaning in the context of the digitalization of aviation. The virtual product aims to comprehensively simulate and design aircraft in the computer. Its realization requires further development of modern methods and algorithms, which also have the potential to be implemented efficiently on hardware of the future, for example on quantum computers.

[†]This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02443 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
stefan.goertz@dlr.de (S. Görtz); stefan.langer@dlr.de (S. Langer)
ORCID(s): 0009-0007-5379-785X (S. Görtz); - (S. Langer)

2. Content of the talk

Within this context, a three-year DLR project entitled "Machine Learning and Quantum Computing – Digitalization of Aircraft Development 2.0", was established in spring 2021 with the goal to investigate whether and how high-fidelity aerodynamic simulations can be carried out using innovative methods from the field of machine learning and in which way these methods can also be implemented on quantum computers.

2.1. PINNs and PIQCs

Among other things, novel simulation algorithms based on neural networks were being developed and applied to compressible fluid mechanics equations, such as the Euler equations [1, 2]. These so-called physics-informed neural networks (PINNs) can also be transferred to quantum computers, with certain adjustments, using physics-informed quantum circuits [3]. In contrast to established, classic methods, in which generally only a given, specific simulation can be carried out, this novel approach even opens up the possibility of integrating various parameters such as object shape or flow conditions as variables into the simulation. If a parameter-dependent solution is calculated, many different simulations can be carried out very efficiently by varying the parameters, demonstrated in Fig. 1. Physics-informed quantum circuits have so far only been applied to comparably simple differential equations, due to the computational effort required to simulate quantum computers. Nevertheless, these initial results look promising.

2.2. Correction of coarse grid results

Another research area of the project group dealt with the question of how machine learning can be used to improve the resolution of relatively imprecise simulation results that are based on relatively coarse discretizations, that is, a limited number of degrees of freedom corresponding to relatively

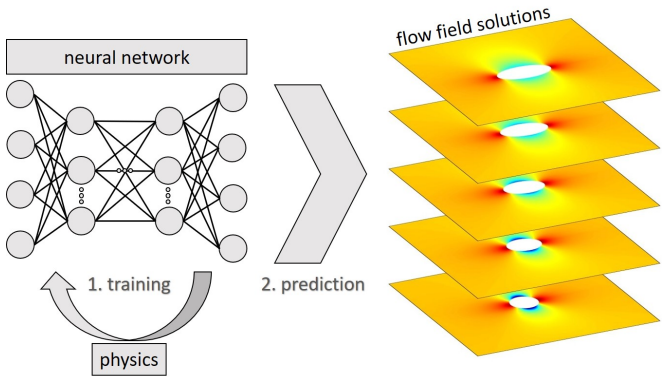


Figure 1: Physics-informed neural network for parametric prediction of compressible flow around a parameterized ellipse.

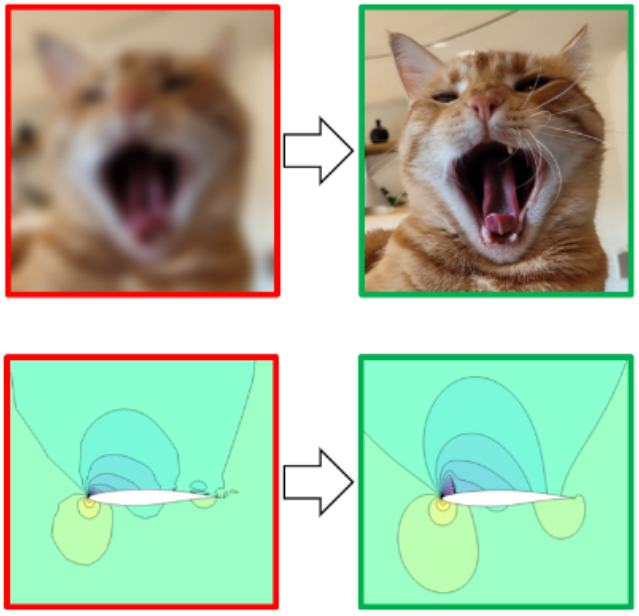


Figure 2: So-called "super-resolution problem" known from image processing (top), which inspires the correction of coarsely discretized flow simulations (bottom).

coarse grids. Well-known methods from image processing were used and adapted to enable corrections to flow simulations, as shown in Fig. 2. These methods open up the possibility of accelerating the still computationally very complex method for solving the Reynolds averaged Navier-Stokes equations. A reduction of this complex step, namely solving with a high number of degrees of freedom, is realized by carrying out a relatively inexpensive correction to a solution obtained by a reduced number of degrees of freedom, as a post-processing step [4, 5].

3. Conclusion

The work done in the project demonstrated that machine learning methods and in particular the use of neural networks can be applied to aerodynamically motivated problems. It was also possible to demonstrate ideas on how these methods can be used to apply novel technologies, such as quantum computers, for the simulation of compressible flows at high REYNOLDS numbers. Further investments and future work in this direction seem worthwhile to leverage the synergy effects of quantum computers and AI methods to make them accessible for industrial requirements.

The talk will give an overview of the key result of the project group, with a focus on topics that will be further investigated in the recently launched DLR Quantum-Computing Initiative project "Towards Quantum Fluid Dynamics" (ToQuaFlics), in particular quantum-inspired topics.

References

[1] S. Wassing, S. Langer, P. Bekemeyer, Physics-informed neural networks for parametric compressible Euler equations, Computers & Fluids 270 (2024) 106164. doi:10.1016/j.compfluid.2023.106164.

[2] S. Wassing, S. Langer, P. Bekemeyer, Parametric Compressible Flow Predictions using Physics-Informed Neural Networks, in: 8th European Congress on Computational Methods in Applied Sciences and Engineering, CIMNE, 2022. doi:10.23967/eccomas.2022.217.

[3] P. Siegl, S. Wassing, D. M. Mieth, S. Langer, P. Bekemeyer, Solving transport equations on quantum computers—potential and limitations of physics-informed quantum circuits, CEAS Aeronautical Journal (2024). doi:10.1007/s13272-024-00774-2.

[4] A. Kiener, S. Langer, P. Bekemeyer, Data-driven correction of coarse grid CFD simulations, Computers & Fluids 264 (2023) 105971. doi:10.1016/j.compfluid.2023.105971.

[5] A. Kiener, S. Langer, P. Bekemeyer, Correcting the discretization error of coarse grid CFD simulations with machine learning, in: 8th European Congress on Computational Methods in Applied Sciences and Engineering, CIMNE, 2022. doi:10.23967/eccomas.2022.072.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Towards Large-Scale Computational Fluid Dynamics Solvers with Quantum Iterative Algorithms

Chelsea A. Williams[a,b,*], Antonio A. Gentile[b], Vincent E. Elfving[b], Daniel Berger[c] and Oleksandr Kyriienko[a]

[a] University of Exeter, Department of Physics and Astronomy, Stocker Road, EX4 4QL Exeter, UK
[b] PASQAL SAS, 7 Rue Léonard de Vinci, 91300 Massy, France
[c] Siemens AG, Gleiwitzer Str. 555, 90475 Nürnberg, Germany

ARTICLE INFO[†]

Keywords:
Quantum Differential Equations Algorithm;
Iterative Method;
CFD Pipeline

ABSTRACT

We introduce a quantum algorithm to solve differential equations iteratively, employing the Jacobi scheme on a quantum register with trajectory information stored using a linear combination of unitaries. We benchmark the approach on paradigmatic fluid dynamics problems where our results stress that instead of inverting large matrices, one can program quantum computers to perform multigrid-type computations and leverage corresponding advances in scientific computing.

1. Introduction

Differential equations are crucial for understanding various natural and engineered systems. The field of computational fluid dynamics (CFD) faces significant challenges due to the complexity of solving Navier-Stokes equations, especially for turbulence modeling and aircraft aerodynamics. Overcoming these challenges requires developing scalable and parallelizable differential equation solvers to accurately capture multi-scale phenomena in high-performance simulations.

Quantum computing offers a distinct approach to computational problem-solving through exponentially growing state spaces and entangled configurations. While quantum algorithms show potential for enhancing differential equation algorithms, current approaches relying on full matrix inversion differ from the typical methods used in CFD solvers. These solvers opt for memory-efficient iterative methods, avoiding large matrix inversions and instead employing relaxation-based techniques to approximate solutions. Thus, we note the need for extending the quantum algorithmic toolbox to include iterative and multigrid methods.

Our work focuses on implementing a quantum algorithm for iterative differential equation solvers, emphasizing the construction of gate-based quantum circuits. Using the Jacobi method as an illustrative example, we demonstrate key algorithmic elements for developing quantum-based iterative solvers and evaluate its performance in CFD applications through examples using the Burgers equation and coupled Euler equations.

2. Algorithm

The steps involved in implementing the quantum Jacobi method algorithm are outlined in Fig. 1. The aim is to iteratively solve a differential equation by off-loading the intensive computational calculations involved in large-scale Jacobi iterations to a quantum processor.

The implementation begins by applying finite differencing to the differential equation. The problem then reduces to solving a system of N equations specified by $A\boldsymbol{x} = \boldsymbol{b}$ with unknown solution $\boldsymbol{x}$, known vector $\boldsymbol{b}$ and known diagonally dominant matrix $A = D+R$ divided into a diagonal matrix D and off-diagonal matrix R. Starting from a guess for $\boldsymbol{x}_0$, the Jacobi solver iterates over K steps to obtain an approximate solution $\boldsymbol{x}_K$ using the scheme

$$\boldsymbol{x}_k = D^{-1}(\boldsymbol{b} - R\boldsymbol{x}_{k-1}) \; \forall \; k \in [1, K]. \tag{1}$$

Firstly, an appropriate block-encoding scheme is used to embed the problem data into the unitary operators $U_l \; \forall \; l \in [D^{-1}, R, \boldsymbol{b}, \boldsymbol{x}_0]$ [1]. Secondly, the Jacobi iteration method is recast into a quantum state calculation

$$|x_k\rangle = U_{D^{-1}}\left[U_b\,|\emptyset\rangle - U_R\,|x_{k-1}\rangle\right] \tag{2}$$

where $|\emptyset\rangle$ is the computational zero state. The difficulty with constructing a circuit to implement this is finding a way to

[†] This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02444 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
cw962@exeter.ac.uk (C.A. Williams); andrea.gentile@pasqal.com (A.A. Gentile); vincent.elfving@pasqal.com (V.E. Elfving); berger.daniel@siemens.com (D. Berger); o.kyriienko@exeter.ac.uk (O. Kyriienko)
ORCID(s): 0009-0007-5018-1160 (C.A. Williams); 0000-0002-1763-9746 (A.A. Gentile); 0000-0002-5105-5664 (V.E. Elfving); - (D. Berger); 0000-0002-6259-6570 (O. Kyriienko)

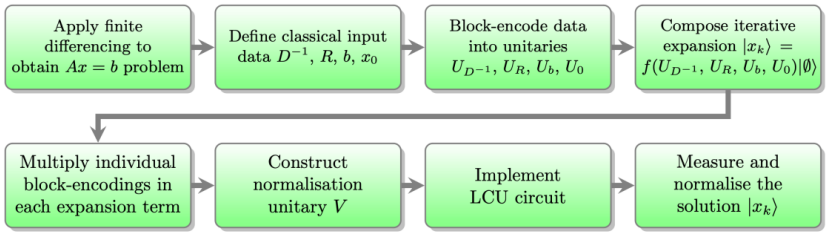


Figure 1: Algorithmic pipeline of the quantum Jacobi method. The input is the differential equation and the output is the k^{th} iterate solution.

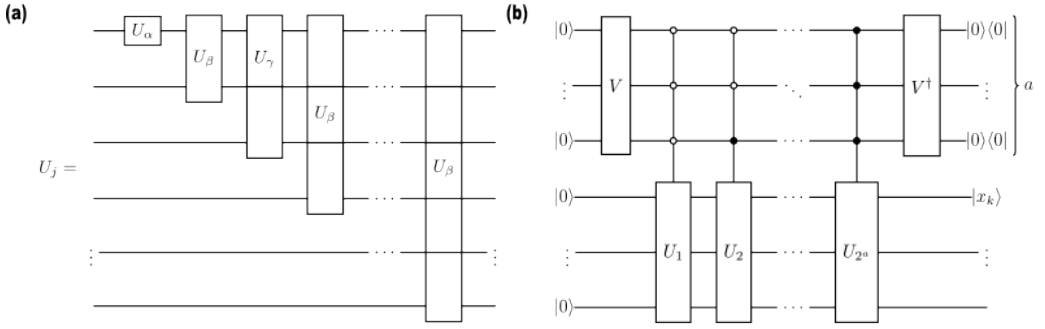


Figure 2: Quantum Jacobi method circuit. **(a)** Multiplication circuit to obtain the block-encoding U_j of the j^{th} expansion term. The block-encoding U_α represents the vector encodings of U_b or U_{x_0}. The block-encoding U_β and U_γ represents the matrix encodings $U_{D^{-1}}$ and U_R. **(b)** LCU circuit used to solve for the k^{th} iterate $|x_k\rangle$ using $a = \lceil \log_2(k+1) \rceil$ ancillary qubits. The U_j represent the block-encodings of the $j^{\text{th}} \in [1, 2^a]$ expansion terms and V encodes their respective normalization factors c_j.

subtract the amplitudes of an unknown iterate state from a known state. The no-cloning theorem prohibits the creation of a deterministic circuit for state vector subtraction [2]. Instead, we employ probabilistic methods like the linear combination of unitaries (LCU) technique [3].

The k^{th} iterate solution can be written as an expansion of the recursion scheme in terms of a linear combination of known operators acting on the zero state

$$|x_k\rangle = \left[\sum_{j=1}^{k} (-U_{D^{-1}} U_R)^{j-1} U_{D^{-1}} U_b + (-U_{D^{-1}} U_R)^k U_{x_0} \right] |\emptyset\rangle . \tag{3}$$

Each of the $k+1$ terms in this expansion is obtained by multiplying the individual block-encodings U_l using the circuit in Fig. 2(a). Due to the nature of the embedding scheme of the block-encoded data, ancillary qubits are required to eliminate cross-terms that arise when the individual unitaries are multiplied [4].

After obtaining the multiplied expansion terms U_j, the solution can be written as a summation

$$|x_k\rangle = \sum_{j=1}^{k+1} \pm c_j U_j |\emptyset\rangle . \tag{4}$$

This is implemented on a quantum device with the LCU circuit in Fig. 2(b). The coefficients c_j refer to the multiplication of the normalization constants associated with the multiplied encodings and are embedded into the unitary V [3]. The superposed solution $|x_k\rangle$ is then obtained by projecting the auxiliary register onto the zero state.

Information is extracted from the quantum solution which naturally embeds all the prior iterate solutions $|x_{k<K}\rangle$ in its history. A prediction of an observable represented by operator M is obtained by taking an expectation value $\langle x_k| M |x_k\rangle$. This could be used to measure features of the solution such as principal components or statistical moments.

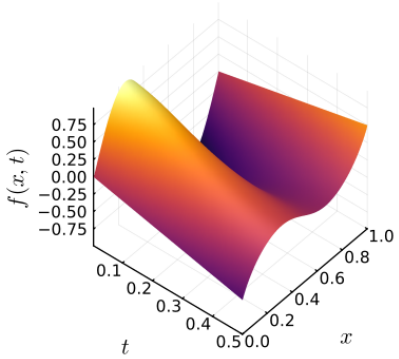


Figure 3: Solving the Burgers equation with $K = 80$ iterations of the quantum Jacobi method. The scalar field surface $f(x,t)$ represents a traveling sinusoidal wave in a dissipative system.

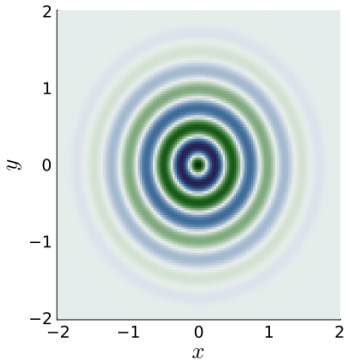


Figure 4: Solving the Euler equations with $K = 12$ iterations of the quantum Jacobi method. The pressure field solution $p(x,y)$ represents the transmission of a Gaussian-modulated noise, emitted from a static point source, through quiescent air.

3. Results

We apply the quantum Jacobi solver to CFD problems. We first focus on the modeling of convection-diffusion systems. The physics of these systems is governed by the viscous Burgers equation, which in one-dimension is given by

$$\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial x} = \mu\frac{\partial^2 f}{\partial x^2} \text{ for } 0 \le x \le 1 \text{ and } 0 \le t \le 0.5. \quad (5)$$

This differential equation models the scalar field $f(x,t)$ within a fluid of viscosity $\mu = 0.08$.

The problem is recast to a system of equations with a suitable finite differencing scheme

$$f(x,t_m) = A^{-1} f(x,t_{m-1}) \; \forall \; m \in [0,150] \quad (6)$$

and solved iteratively with the quantum Jacobi method. The amplitudes of the quantum state solution over time are shown in Fig. 3 after simulating the quantum Jacobi algorithm with dynamic boundary conditions $f(0,t) = -t$, $f(1,t) = t$ and initial condition $f(x,0) = \sin(2\pi x)$. We observe that high-quality solutions can be obtained from the full state vector using limited resources. The surface plot visualizes the scalar field solution, which represents a wave traveling through a medium with non-zero viscosity.

We apply the quantum Jacobi solver to a second problem in CFD regarding aeroacoustics. We look to model the propagation of sound waves in an quiescent fluid. The fluid dynamics is governed by the linearized Euler equations, which in two-dimensions is given by

$$\frac{\partial p}{\partial t} + \bar{\rho}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0, \quad (7)$$

$$\frac{\partial u}{\partial t} + \frac{1}{\bar{\rho}}\frac{\partial p}{\partial x} = 0, \quad (8)$$

$$\frac{\partial v}{\partial t} + \frac{1}{\bar{\rho}}\frac{\partial p}{\partial y} = 0. \quad (9)$$

Here, $u(x,t)$ is the velocity component in the x-direction, $v(y,t)$ is the velocity component in the y-direction and $\bar{\rho}$ is the mean density of the fluid. These equations model the propagation of acoustic waves via the pressure field solution $p(x,y,t)$ with zero source terms and negligible base flow.

As before, the problem is recast to a system of equations with a suitable finite differencing scheme and solved iteratively with the quantum Jacobi method. The discretization is applied in the x-direction using 128 spatial nodes with $x \in [-2,2]$ and in the y-direction using 128 spatial nodes with $y \in [-2,2]$. The temporal domain is divided into 60 temporal nodes. The amplitudes of the quantum state solution in the xy-plane is shown in Fig. 4 after simulating the quantum Jacobi algorithm with non-reflective boundary conditions and a Gaussian-modulated sinusoidal initial condition $\cos(2\pi\omega r)e^{-r^2}$ with frequency $\omega = 2$ and radial component $r^2 = x^2 + y^2$. The contour plot visualizes the pressure field solution of the noise transmitted from a static point source, demonstrating how the Gaussian-modulated sinusoidal waves travel through a quiescent fluid.

4. Discussion

The main advantage of using a quantum iterative solver over a classical analog stems from the memory savings associated with manipulating the problem data and calculating

the iterate solutions on quantum processors. An exponential speed-up in memory comes from the fact that optimized classical iterative approaches achieve a scaling of $\mathcal{O}(N)$, while the quantum Jacobi solver has an improved scaling of $\mathcal{O}(\log_2(N))$. This characteristic is indeed relevant for extensive industrial CFD simulations where the memory required to store and manipulate the data exceeds the amount available in classical computing. This can be further justified by employing quantum iterative solvers as part of preconditioning subroutines within larger multigrid algorithms. These CFD pipelines are a necessity for future surrogate modeling and multiphysics simulations in leveraging the computational upside of quantum physics.

In terms of quantum resources estimates, the overall circuit requires $\log_2(N) + 2k + \lceil \log_2(k+1) \rceil$ qubits and $\mathcal{O}(K^2 C)$ gates, where C is the cost associated with the chosen block-encoding protocol [1]. We also highlight that the number of iterations scales linearly with the condition number of the system.

5. Conclusion

We proposed an algorithm for implementing a quantum iterative solver of differential equations based on a linear combination of unitaries represented by block-encodings. Specifically, we demonstrated the building blocks for the quantum Jacobi iterative solver and developed a pipeline for preparing the approximate solution as a quantum state. The tools and techniques developed can be readily applied to other iterative schemes.

We tested the approach by preparing solutions of convection-diffusion systems described by the viscous Burgers equation and for sound wave propagation described by the Euler equations. We observe that the iterative approach can recover high-quality solutions in very few iterations, even for cases where the solution is discontinuous. This is especially notable in turbulent CFD simulations where shock waves appear.

Further work regarding quantum iterative solvers could benefit from incorporating machine learning approaches to increase the accuracy and performance of the algorithm. For example, improvements in the convergence rate could come from using a physics-informed approximation to $\boldsymbol{x}_0$ based on a variational ansatz circuit. Another example could involve reducing the number of iterations by employing quantum physics-informed neural networks within the quantum iterative algorithm. Utilizing quantum solvers in this way will enable quantum computing to become a reality sooner than anticipated.

References

[1] D. Camps, L. Lin, R. V. Beeumen, C. Yang, Explicit quantum circuits for block encodings of certain sparse matrices (2023). doi:10.48550/arXiv.2203.10236.

[2] U. Alvarez-Rodriguez, M. Sanz, L. Lamata, E. Solano, The forbidden quantum adder, Scientific Reports 5 (1) (2015). doi:10.1038/srep11983.

[3] A. M. Childs, N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, Quantum Info. Comput. 12 (11–12) (2012) 901–924. doi:10.48550/arXiv.1202.5822.

[4] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, F. G. S. L. Brandão, Quantum algorithms: A survey of applications and end-to-end complexities (2023). doi:10.48550/arXiv.2310.03011.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Solving Fluid Dynamics Equations with Differentiable Quantum Circuits

Smit Chaudhary[a,*], Giorgio Tosti Balducci[a], Oleksandr Kyriienko[b], Panagiotis Kl. Barkoutsos[a], Lorenzo Cardarelli[a] and Antonio A. Gentile[a]

[a] *Pasqal SAS, 7 Rue L. de Vinci, Massy, 91300, France*
[b] *University of Sheffield, School of Mathematical and Physical Sciences, Western Bank, Sheffield, S10 2TN, UK*

ARTICLE INFO[†]

Keywords:
Fluid Dynamics;
Quantum Computing;
Physics-Informed Neural Networks

ABSTRACT

Differentiable quantum circuits (DQCs) are the hybrid quantum-classical alternative to Physics-Informed Neural Networks (PINNs). The latter ones have been introduced from the machine learning community to avoid the curse of dimensionality in mesh-based computational fluid dynamics (CFD) solvers, and allow for seamless inclusion of information from available data. The adoption of quantum circuits is motivated by enabling access to highly expressive feature maps, which might be key in capturing intricate solutions to selected fluid dynamics problems. In this work, we discuss the potential of DQCs and its recent extensions to address paradigmatic CFD use cases.

1. Introduction

Recent advances in the domain of CFD allowed for better solutions of fluid dynamics problems and their respective partial differential equations (PDEs). Many of CFD's most successful techniques use finite differences/elements/volume methods or spectral methods to solve some form of the Navier-Stokes equations [1]. Despite the high fidelity that these methods can reach, their reliance on meshes or modes exposes them to the curse of dimensionality, i.e., high computational costs in multi-scale PDEs involving multiple equations in 2D and 3D geometries.

Recently, PINNs have been proposed as an alternative paradigm to solve PDEs [2]. In essence, PINNs approximate the PDE solution with the output of a neural network (NN), trained to minimize loss terms directly derived from the equations. PINNs offer an edge with respect to standard supervised learning (SL), as in principle, they do not require any sample of the solution, be that analytical, numerical or experimental. Thanks to their efficiency and flexibility, PINNs found applications also in CFD problems [3].

Using NNs as a solution approximator is an obvious choice due to the popularity of neural architecture, however any universal, differentiable, trainable model can be used in their stead. One proposal replaces NNs with Differentiable Quantum Circuits (DQCs) [4], as represented in Fig. 1. The approach is variational (which makes it more viable on near-term quantum hardware [4]), and offers exact differentiation [5], removing the need of numerical differentiation entirely. Finally, DQC offers efficient ways to encode problem features (e.g., coordinates), leveraging the exponentially large Hilbert space (w.r.t. the number of qubits) accessed by the quantum circuits [4, 6]. Therefore, their execution can be more energy-efficient, when compared to GPU training of NNs.

2. Methods

Consider the following generic differential problem

$$\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0, \quad t \in [0, T], \, x \in \Omega, \tag{1}$$

$$u(t = 0, x) = g(x), \quad x \in \Omega, \tag{2}$$

$$\mathcal{B}[u] = 0, \quad t \in [0, T], \, x \in \partial\Omega, \tag{3}$$

where $\mathcal{N}$ is a generic nonlinear operator of u and its x-derivatives, $g(x)$ is an initial condition and $\mathcal{B}[u]$ is a boundary operator.

In the DQC methodology, we approximate the solution as $u(x) \approx u_\theta(x)$, via the expectation value of an observable $\hat{C}$:

$$u_\theta(x) = \langle u_\theta(x) | \hat{C} | u_\theta(x) \rangle, \tag{4}$$

[†] This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02445 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
smit.chaudhary@pasqal.com (S. Chaudhary); giorgio.tosti-balducci@pasqal.com (G. Tosti Balducci); o.kyriienko@sheffield.ac.uk (O. Kyriienko); pbarkoutsos@gmail.com (P.Kl. Barkoutsos); lorenzo.cardarelli@pasqal.com (L. Cardarelli); andrea.gentile@pasqal.com (A.A. Gentile)
ORCID(s): 0000-0003-0243-6513 (S. Chaudhary); 0000-0001-9242-676X (G. Tosti Balducci); 0000-0002-6259-6570 (O. Kyriienko); 0000-0001-9428-913X (P.Kl. Barkoutsos); 0000-0001-5010-5167 (L. Cardarelli); 0000-0002-1763-9746 (A.A. Gentile)

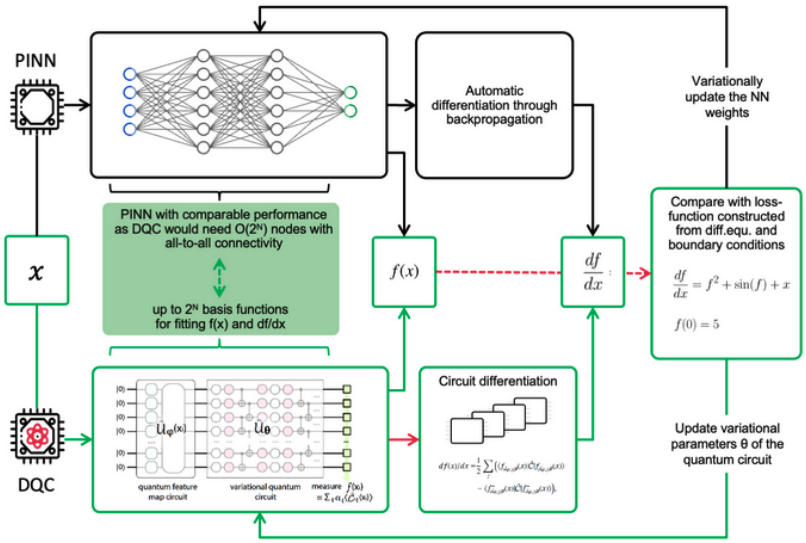


Figure 1: Diagram of the DQC algorithm.

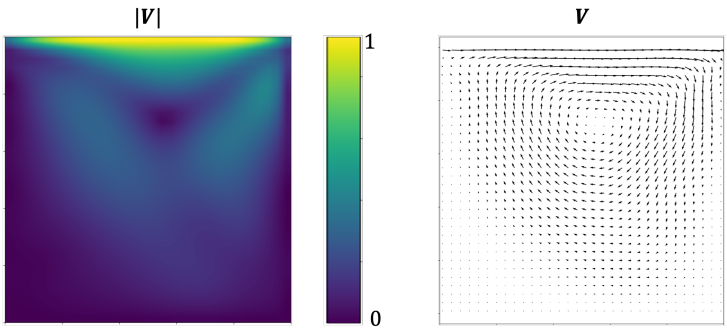


Figure 2: Lid-driven flow in a square cavity solved with DQC. On the left is the total velocity magnitude and on the right the velocity profile.

measured on the output state $|u_\theta(x)\rangle$ of a Quantum Neural Network (QNN), a circuit described by (trainable) θ parameters and encoding the features x as [7]

$$|u_\theta(x)\rangle = U_\theta U_\varphi(x)\,|0\rangle\,. \tag{5}$$

θ's are trained to minimize a loss function evaluating the residuals of Eqs. 1-3, when $u_\theta(x) \to u_(x)$, with $U_\varphi(x)$ and U_θ unitary operations.

3. Results

Figure 2 shows the solution of the lid-driven cavity flow problem obtained with DQC. We present the solution of the incompressible Navier-Stokes equation in its non-dimensional, steady state form

$$(\mathbf{V}\cdot\boldsymbol{\nabla})\mathbf{V} = -\boldsymbol{\nabla}p + \frac{1}{Re}\nabla^2\mathbf{V}, \tag{6}$$

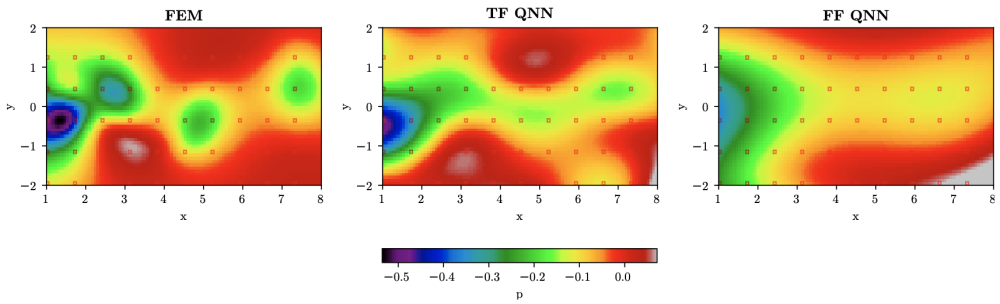


Figure 3: Pressure field downstream from an obstacle at $x = 0$. On left is the reference FEM result. In middle is the prediction of a Trainable Frequency (TF) QNN. On the right is the prediction of a Fixed Frequency (FF) QNN.

where $\mathbf{V} = (v_x, v_y)^T$ is the 2D velocity vector, p is the pressure and Re is the REYNOLDS number. The QNN surrogate for $\mathbf{V}$ and p employed 4 qubits for both x and y coordinates, and used 8 layers of the so-called *hardware efficient ansatz* [8] as U_θ. The solution for Re=150 matches closely the benchmark obtained via a finite element method (FEM) solver.

The spectrum of frequencies accessible to the feature map in the quantum circuit can be augmented by including in U_φ additional trainable variational parameters [9]. Figure 3 shows the pressure field downstream from a cylinder for 2D time-dependent Navier-Stokes equations.

A QNN approximating the stream function $\widetilde{\psi}$ is used to derive the two components of the velocity $\mathbf{V}$ and to satisfy the continuity relation automatically. Another QNN is used to approximate the pressure field p. Each QNN employs 8 layers of the ansatz. The results for Re=100 show a noticeable improvement compared to a fixed frequency version of the QNNs.

4. Conclusions

In this work, we applied the DQC algorithm to fluid dynamics PDEs. We presented promising results obtained both with various feature map architectures. Ongoing development aims to also include inductive biases [10], targeting, e.g., irrotational flows.

References

[1] T. J. Chung, Computational Fluid Dynamics, Cambridge University Press, 2002. doi:10.1017/CBO9780511606205.

[2] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.

[3] S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mechanica Sinica 37 (12) (2021) 1727–1738. doi:10.1007/s10409-021-01148-1.

[4] O. Kyriienko, A. E. Paine, V. E. Elfving, Solving nonlinear differential equations with differentiable quantum circuits, Physical Review A 103 (5) (2021) 052416. doi:10.1103/PhysRevA.103.052416.

[5] D. Wierichs, J. Izaac, C. Wang, C. Y.-Y. Lin, General parameter-shift rules for quantum gradients, Quantum 6 (2022) 677. doi:10.22331/q-2022-03-30-677.

[6] A. E. Paine, V. E. Elfving, O. Kyriienko, Physics-informed quantum machine learning: Solving nonlinear differential equations in latent spaces without costly grid evaluations (2023). arXiv:2308.01827.

[7] D. Seitz, N. Heim, J. P. Moutinho, R. Guichard, V. Abramavicius, A. Wennersteen, G.-J. Both, A. Quelle, C. de Groot, G. V. Velikova, V. E. Elfving, M. Dagrada, Qadence: a differentiable interface for digital and analog programs, IEEE Software (2025) 1–14. doi:10.1109/MS.2025.3536607.

[8] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature 549 (7671) (2017) 242–246. doi:10.1038/nature23879.

[9] B. Jaderberg, A. A. Gentile, Y. A. Berrada, E. Shishenina, V. E. Elfving, Let quantum neural networks choose their own frequencies, Physical Review A 109 (4) (2024) 042421. doi:10.1103/PhysRevA.109.042421.

[10] A. Ghosh, A. A. Gentile, M. Dagrada, C. Lee, S.-H. Kim, H. Cha, Y. Choi, B. Kim, J.-I. Kye, V. E. Elfving, Harmonic (Quantum) Neural Networks, arXiv:2212.07462 (2022). arXiv:2212.07462.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

A Practical Implementation of Quantum Lattice Boltzmann Algorithms

Călin A. Georgescu[a,*], Matthias Möller[a]

[a]Delft University of Technology, Department of Applied Mathematics, Mekelweg 4, Delft, The Netherlands

ARTICLE INFO[†]

Keywords:
Quantum Computing;
Quantum Algorithm;
Lattice Boltzmann;
Python

ABSTRACT

We introduce QLBM, a software package that aims to unify the fragmented landscape of quantum Lattice Boltzmann Methods (QBMs) and facilitate future work. We highlight several nuances of competing approaches and introduce the user- and researcher-facing sides of QLBM. We demonstrate how QLBM provides an end-to-end environment for developing and analyzing QBMs for both simulators and quantum hardware. QLBM is publicly available at https://github.com/QCFD-Lab/qlbm under an MPL-2.0 license.

1. Introduction

The field of Quantum Computing (QC) [1] has recently received a substantial amount of attention from both academia and industry. QC carries the potential to augment the current hybridized computational landscape with a drastically different yet complementary archetype. The exponential memory efficiency and quantum parallelism traits of the quantum computing paradigm makes it especially attractive for the increasingly demanding large-scale CFD applications of today.

The current state of QC hardware has been undergoing rapid development and is currently in the so-called *Noisy Intermediate-Scale Quantum* phase [2]. While quantum computers available today showcase some of the core advantages that theoretical physics promises, they are limited in both the number of qubits available and the time span that qubits can retain coherent states. These constraints impose great limitations on the applications that quantum computers can presently carry out.

To facilitate the research of quantum algorithms in an era without *Fault-Tolerant* QCs, scientists have turned to simulation methods instead. Recently, increasingly many software frameworks have emerged to bridge the gap between theoretical advances in algorithmics and hardware availability. These range from general purpose simulation tools [3, 4, 5] to specialized packages aimed at machine learning [6, 7] and material simulation [8]. The current state of *quantum software* intersects QC theory and available hardware such that researchers can leverage classical hardware to verify large-scale algorithmic prototypes while quantum counterparts edge closer to fault-tolerance. At the same time, available software retains compatibility with the circuit model of universal quantum computers.

In this work, we introduce QLBM, a software package for developing quantum Lattice Boltzmann Methods. We first briefly introduce the Lattice Boltzmann Method (LBM) and its quantum counterparts that motivate the importance of our work in Sec. 1.1 and Sec. 1.2, respectively. Section 2 describes the QLBM software package and its features.

1.1. Lattice Boltzmann methods

The Boltzmann Equation (BE) describes the kinetic behavior of fluid at the mesoscopic scale, nestled between microscopic Newtonian dynamics and macroscopic Navier-Stokes continua. The BE models the state of populations of fluid particles as a statistical distribution function over physical space, velocity, and time. The discretization of the BE along phase space and time yields the Lattice Boltzmann Equation, which can in turn be solved numerically by the Lattice Boltzmann Method (LBM). Each LBM time step can be conceptually broken down into three subroutines: streaming through physical space, reflection at the boundaries of the fluid domain, and (non-linear) particle collision.

In recent years, the LBM has become a more and more popular option to solving fluid flow problems for several reasons. From a theoretical standpoint, it allows for the computation of macroscopic quantities such as mass and momentum density [9]. From a practical standpoint, the LBM lends itself well to massively parallel computing paradigms [10, 9]. Over the years, several parallel software implementations of the LBM have emerged, including OPENLB [11], WALBERLA [12], LBMPY [13], and PYLBM [14], which are able to carry out distributed simulations on hundreds of heterogeneous compute nodes. Simultaneously, researchers have been investigating formulations that allow the parallelism of the LBM to be exploited through paradigm of quantum computing.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02446 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
c.a.georgescu@tudelft.nl (C.A. Georgescu); m.moller@tudelft.nl (M. Möller)
ORCID(s): 0000-0002-8102-6389 (C.A. Georgescu); 0000-0003-0802-945X (M. Möller)

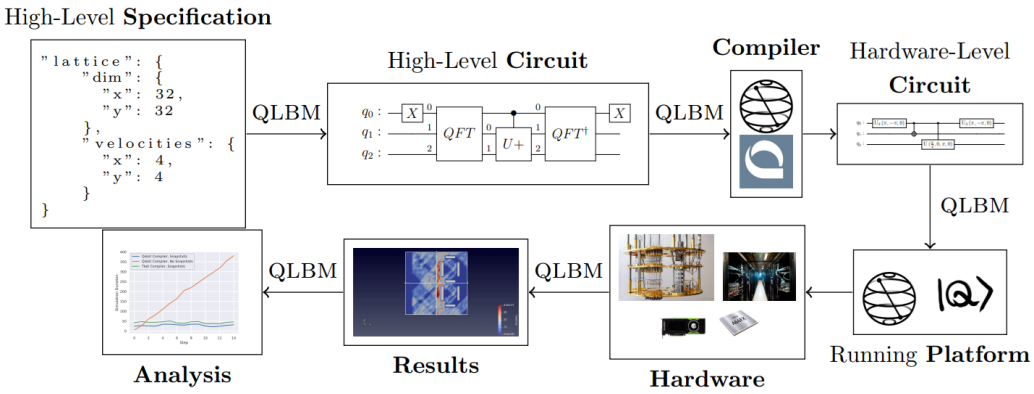


Figure 1: Overview of the QLBM workflow.

1.2. Quantum algorithms for Lattice Boltzmann methods

The initial wave of research into Quantum Computational Fluid Dynamics (QCFD) occurred between 2001 and 2003 and largely focused on extending the lattice-gas model to distributed quantum devices [15, 16, 17, 18]. This work tailors quantum lattice-gas solvers to a decentralized system of quantum computers with limited number of qubits per device, linked together through classical communication channels. Though this approach enables the balancing of the computational workload through horizontal scaling, it requires a number of qubits that grows linearly with the number of grid points of the lattice.

Recently, Quantum Lattice Boltzmann Methods (which we abbreviate as QBMs, to distinguish from the software) have emerged as promising candidates for the future direction of QCFD. Research surrounding QBMs has largely focused on the development of quantum primitives that implement (parts of) the LBM time-marching loop. These initiatives have given rise to several techniques that accommodate specific subroutines of the LBM, imposing trade-offs between scalability and versatility. One way to categorize existing QBMs is by how they address the inherent nonlinearity of collision.

Todorova and Steijl [19] and Schalkers and Möller [20] propose *collisionless* methods that include primitives for particle streaming and boundary conditions, but omit the collision operator entirely. Steijl and co-workers [21, 22] alternatively propose a method in which quantum primitives that implement floating point arithmetic can compute nonlinear terms, but require a reversible conversion between the encoding of the quantum state used to perform streaming and the encoding that enables the computation of the nonlinear velocity terms at each time step. Succi and collaborators [23, 24] adopt an approach based on truncated Carleman linearization, that approximates the non-linear LBE by a finite-dimensional linear system of equations that can be expressed in terms of (unitary) quantum operators. Budinski [25, 26] further developed an approach that enables both streaming and collision but that incurs a probability of measuring an orthogonal (irrelevant) quantum state after each time-step. Finally, Schalkers and Möller [27] extended a previously developed encoding equipped it with a collision operator at the cost of requiring a number of qubits that scales with the number of simulated time steps.

The current state of QBMs is fragmented between several approaches that each present different strengths and weaknesses. This poses several challenges for researchers seeking to advance the field. First, the many nuances of present QBM techniques make the comparison of the performance and scalability difficult. From various quantum state encodings to the decomposition of exponentially sized matrices into quantum gates, QBMs build on top of extensive knowledge and technology stacks that make implementation a significant challenge. Second, the fractured nature of the field poses challenges for techniques that augment existing work. Third, the scarce availability of QBM implementations detracts from the reproducibility of the field.

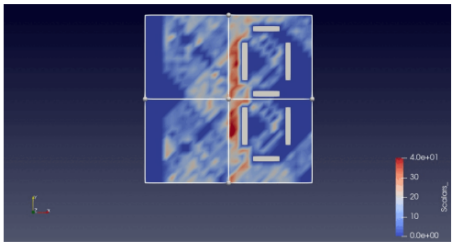
2. The QLBM package

In this talk we present QLBM, an open-source software package that aims to accelerate QBM research. With this

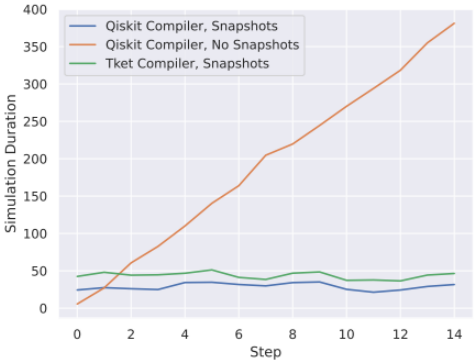
initiative we seek to lay the foundation of a unified end-to-end development environment for quantum Lattice Boltzmann algorithms. QLBM aims to facilitate both the seamless development and comparison of existing methods and their integration into the broader quantum software technology stack, reducing redundant effort for researchers and practitioners alike.

Figure 1 provides a high-level schematic of the end-to-end workflow that QLBM enables. In the first part of the talk, we focus on the application of QBMs already implemented in the software. This includes a demonstration of how QLBM converts human-readable JSON specifications of 2D and 3D systems into quantum circuits, and how to convert the generated circuits to specific platforms. We also showcase the integration of QLBM with PARAVIEW [28] for flow field and geometry visualization. Figure 2a displays an example of the supported visualization techniques. Finally, we show several concrete simulation use cases that users can implement with just a few lines of Python code.

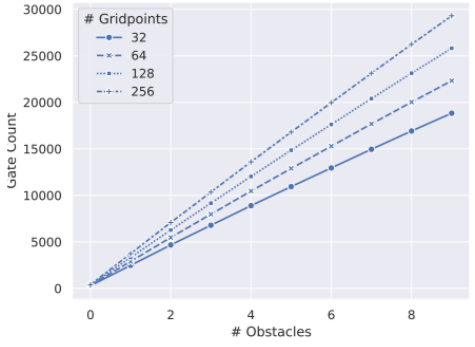
In the second part of the talk, we concentrate on QLBM as a tool for developing novel QBMs. To this end, we highlight the internal structure of the framework, the segregation of quantum circuit primitives into a complexity hierarchy, and the extensibility of already implemented methods. We also spotlight QLBM's out-of-the-box integration with state-of-the-art quantum circuit compilers and simulators that support a broad range of CPU and GPU architectures. To complement the research-oriented side of QLBM, we showcase mechanisms for exploiting potential speedups in quantum simulators and an accompanying set of analysis tools that allow for fair and replicable comparisons between competing methods. Figure 2b shows one such mechanism, which reduces the runtime of simulating time steps of the collisionless QBM (CQBM) developed by Schalkers and Möller [20] from linear to constant. QLBM additionally contains tools meant to ease the transition from simulators to (possibly fault-tolerant) quantum hardware. We demonstrate QLBM's capacity to accurately estimate critical properties of transpiled quantum circuits for a variety of different quantum hardware vendors. Such properties include circuit depth, and the number exact of hardware-native gates. QLBM relies on QISKIT to enable flexible analyses with regard to qubit connectivity and gate sets, which come at no additional cost to the software's internal representation. Figure 2c depicts how the number of gates of circuits implementing CQBM [20] scales with the number of obstacles for 4 different grid refinement levels. We conclude the talk by discussing prospects of QCFD software and QLBM's place within the broader domain.



(a) 2D flow field visualization around boundary-conditioned geometry.



(b) Run time reduction for simulating one time step using QLBM exploits.



(c) Number of high-level gates for 2D CQBM algorithm for different grids.

Figure 2: Example of visualization and analysis options built in QLBM.

References

[1] M. A. Nielsen, I. L. Chuang, Quantum computation and quantum information, Cambridge University Press, 2010. doi:10.1017/CBO9780511976667.

[2] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum 2 (2018) 79. doi:10.22331/q-2018-08-06-79.

[3] Qiskit contributors, Qiskit: An open-source framework for quantum computing (2023). doi:10.5281/zenodo.2573505.

[4] Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya, T. Yamamoto, T. Yan, T. Kawakubo, Y. O. Nakagawa, Y. Ibe, Y. Zhang, H. Yamashita, H. Yoshimura, A. Hayashi, K. Fujii, Qulacs: a fast and versatile quantum circuit simulator for research purpose, Quantum 5 (2021) 559. doi:10.22331/q-2021-10-06-559.

[5] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, R. Duncan, tl ket: a retargetable compiler for nisq devices, Quantum Science and Technology 6 (1) (2020) 014003. doi:10.1088/2058-9565/ab8e92.

[6] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, N. Killoran, PennyLane: Automatic differentiation of hybrid quantum-classical computations (2018). arXiv:1811.04968.

[7] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. Von Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, M. Mohseni, TensorFlow Quantum: A Software Framework for Quantum Machine Learning (2020). arXiv:2003.02989.

[8] L. Bassman Oftelie, C. Powers, W. A. De Jong, Arqtic: A full-stack software package for simulating materials on quantum computers, ACM Transactions on Quantum Computing 3 (3) (2022) 1–17. doi:10.1145/3511715.

[9] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. M. Viggen, The lattice boltzmann method, Springer International Publishing 10 (978-3) (2017) 4–15. doi:10.1103/PhysRev.94.511.

[10] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, 2001. doi:10.1093/oso/9780198503989.001.0001.

[11] M. J. Krause, A. Kummerländer, S. J. Avis, H. Kusumaatmaja, D. Dapelo, F. Klemens, M. Gaedtke, N. Hafen, A. Mink, R. Trunk, J. E. Marquardt, M.-L. Maier, M. Haussmann, S. Simonis, OpenLB—Open source lattice Boltzmann code, Computers & Mathematics with Applications 81 (2021) 258–288. doi:10.1016/j.camwa.2020.04.033.

[12] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum, C. Schwarzmeier, D. Thönnes, H. Köstler, U. Rüde, waLBerla: A block-structured high-performance framework for multiphysics simulations, Computers & Mathematics with Applications 81 (2021) 478–501. doi:10.1016/j.camwa.2020.01.007.

[13] M. Bauer, H. Köstler, U. Rüde, lbmpy: Automatic code generation for efficient parallel lattice Boltzmann methods, Journal of Computational Science 49 (2021) 101269. doi:10.1016/j.jocs.2020.101269.

[14] Pylbm contributors, Pylbm: an all-in-one package for numerical simulations using lattice boltzmann solvers. (2023). URL https://github.com/pylbm/pylbm

[15] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, Physical Review E 63 (4) (2001) 046702. doi:10.1103/PhysRevE.63.046702.

[16] J. Yepez, B. Boghosian, An efficient and accurate quantum lattice-gas model for the many-body Schrödinger wave equation, Computer Physics Communications 146 (3) (2002) 280–294. doi:10.1016/S0010-4655(02)00419-8.

[17] J. Yepez, Quantum lattice-gas model for the Burgers equation, Journal of Statistical Physics 107 (2002) 203–224. doi:10.1023/A:1014514805610.

[18] M. A. Pravia, Z. Chen, J. Yepez, D. G. Cory, Experimental demonstration of quantum lattice gas computation, Quantum Information Processing 2 (2003) 97–116. doi:10.1023/A:1025835216975.

[19] B. N. Todorova, R. Steijl, Quantum algorithm for the collisionless Boltzmann equation, Journal of Computational Physics 409 (2020) 109347. doi:10.1016/j.jcp.2020.109347.

[20] M. A. Schalkers, M. Möller, Efficient and fail-safe quantum algorithm for the transport equation, Journal of Computational Physics 502 (2024) 112816. doi:10.1016/j.jcp.2024.112816.

[21] R. Steijl, Quantum Algorithms for Nonlinear Equations in Fluid Mechanics, in: Quantum Computing and Communications, IntechOpen, 2022. doi:10.5772/intechopen.95023.

[22] Y. Moawad, W. Vanderbauwhede, R. Steijl, Investigating hardware acceleration for simulation of CFD quantum circuits, Frontiers in Mechanical Engineering 8 (2022) 925637. doi:10.3389/fmech.2022.925637.

[23] W. Itani, S. Succi, Analysis of Carleman linearization of lattice Boltzmann, Fluids 7 (1) (2022) 24. doi:10.3390/fluids7010024.

[24] C. Sanavio, S. Succi, Lattice Boltzmann–Carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number, AVS Quantum Science 6 (2) (2024). doi:10.1116/5.0195549.

[25] L. Budinski, Quantum algorithm for the advection–diffusion equation simulated with the lattice Boltzmann method, Quantum Information Processing 20 (2) (2021) 57. doi:10.1007/s11128-021-02996-3.

[26] L. Budinski, Quantum algorithm for the Navier–Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method, International Journal of Quantum Information 20 (02) (2022) 2150039. doi:10.1142/S0219749921500398.

[27] M. A. Schalkers, M. Möller, On the importance of data encoding in quantum Boltzmann methods, Quantum Information Processing 23 (1) (2024) 20. doi:10.1007/s11128-023-04216-6.

[28] J. Ahrens, B. Geveci, C. Law, 36 - ParaView: An End-User Tool for Large-Data Visualization, in: Visualization Handbook, Elsevier, 2005, pp. 717–731. doi:10.1016/B978-012387582-2/50038-1.

Mini-Symposium 2:

Advances in Parallel Simulation of Reacting Flow

Organizers: Jian Fang, Zhi X. Chen, Umair Ahmed, and Daniel Mira

The simulation of reacting flow on modern high-performance computing systems is still a very active and challenging topic. The objective of achieving carbon neutrality requires further investigation into the mechanisms of reacting flow related carbon neutral fuels, such as, hydrogen and ammonia. However, the complexity of turbulence/chemistry interaction and the variety of flow scales in reacting flows raise a lot of challenges to algorithm, numerical scheme, turbulence and combustion models. The rapid development in computing hardware and data technologies has largely boosted the large-scale high-fidelity simulations of turbulent reacting flows, although there are still many on-going works needs to be further discussed within the community. This mini-symposium is to communicate the progresses in simulation of reacting flow.

Topics of interest of this mini-symposium include, but are not limited to:

1. High-performance computing of combustion applications, including direct-numerical/large-eddy simulations.
2. Data-driven techniques related to combustion.
3. Numerical scheme and algorithm that can be used in simulations of reacting flows.
4. Tackling multi-physics phenomenon in combustion (e.g. droplets, detonation, explosions, ignition, flame-wall interaction and quenching).
5. Combustion dynamics and instabilities.
6. Combustion modelling.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

An Efficient Parallel Implementation of a Hybrid Method for the Advection of High Schmidt Scalars in Flows

Simon Santoso[a,*], Guillaume Houzeaux[a] and Damien Dosimont[a]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain

ARTICLE INFO[†]

Keywords:
Particle Method;
Finite-Element Method;
High-Performance Computing;
Passive Scalar

ABSTRACT

We present here a parallel implementation of a hybrid method combining a high order semi-Lagrangian method for the advection-diffusion of scalars and a finite element method for the Navier-Stokes equations. Then, we introduce an overloading strategy to enhance the computational performance of this method. Finally, we present an application test case of a high SCHMIDT scalar in a turbulent jet flow.

1. Introduction

The phenomenon of advection-diffusion equations in turbulent flows manifests in diverse scenarios, including heat transfer [1] or the dispersion of pollutants [2]. Scalars passively advected mainly vary in their diffusivity properties, which are characterized by their SCHMIDT number, defined as the ratio of viscosity to diffusivity. Handling scalars with high SCHMIDT numbers remains a challenge for direct numerical simulation. Specifically, the Kolmogorov scale μ_K, representing the smallest velocity variation scale, is associated with the Batchelor scale μ_B, which denotes the smallest mixing variation scale, through the following relation:

$$\mu_B = \frac{\mu_K}{\sqrt{S_c}} \tag{1}$$

As a consequence, when dealing with SCHMIDT numbers higher than one, scalar dynamics operate at scales smaller than the Kolmogorov scale. Achieving accurate treatment of scalar advection-diffusion at high SCHMIDT numbers requires a finer mesh for the scalar than for the momentum. Consequently, monolithic numerical investigations are limited to moderate SCHMIDT numbers. To address these challenges, we have developed and implemented a method to overcome these limitations.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02447 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
simon.santoso@bsc.es (S. Santoso); guillaume.houzeaux@bsc.es (G. Houzeaux); damien.dosimont@bsc.es (D. Dosimont)
ORCID(s): 0000-0001-7478-4379 (S. Santoso); 0000-0002-2592-1426 (G. Houzeaux); 0000-0002-6620-9873 (D. Dosimont)

2. Equations and numerical method

For incompressible flows, the dynamics of the flow are governed by the Navier-Stokes equations

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \mathbf{0} \tag{2}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{3}$$

with $\mathbf{u}$ the velocity field, p the pressure field, and ν and ρ the viscosity and the density, respectively. Those equations are solved on a unstructured grid thanks to a finite-element method with the EMAC formulation [3].

The dynamics of a passive scalar θ advected in the flow are governed by a advection-diffusion equation:

$$\frac{\partial\theta}{\partial t} + \mathbf{u}\cdot\nabla\theta = \kappa\Delta\theta \tag{4}$$

where κ is the diffusion coefficient of θ. The SCHMIDT number is then defined as $S_c = \nu/\kappa$. The scalar advection-diffusion equation is solved on a fine Cartesian mesh. A semi-Lagrangian particle method is used for the scalar transport. It consists in concentrating the transported quantity θ on a set of particles. Particles are then advected thanks to the advection velocity field and remeshed after each time-step on the grid using an interpolation kernel. This methodology has been validated and applied in [4, 5, 6]. In particular, Cottet showed the high order of the method in [7]. Nevertheless, the authors of [6] highlight the presence of load balancing issues when very unstructured velocity mesh are used. The presented parallel implementation aims to solve those problems.

3. Parallel implementation

In the present work, Navier-Stokes and advection-diffusion equations are solved on two different groups of processors. The first group aims to solve Navier-Stokes equations.

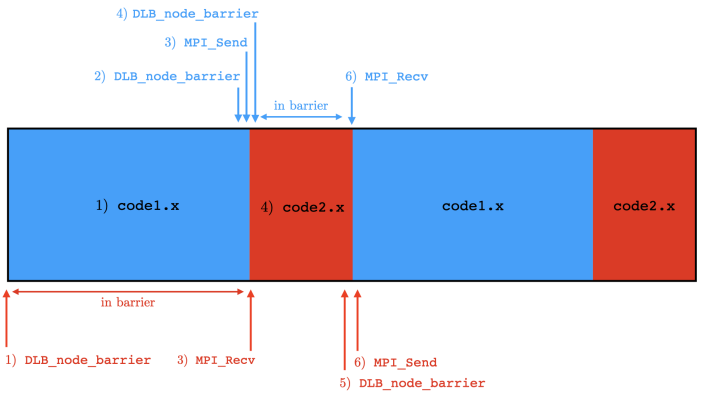


Figure 1: Workflow with or without the use of DLB Barrier.

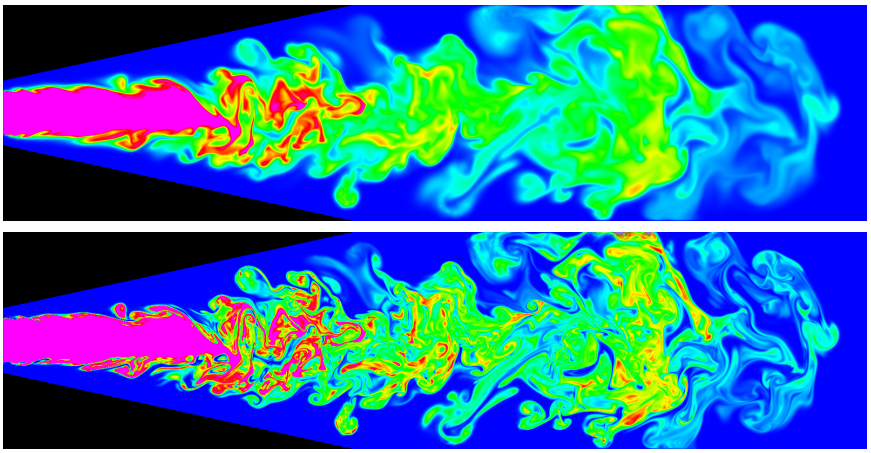


Figure 2: Instantaneous contours of a scalar injected in a turbulent flow. Top picture $Sc = 1$. Bottom picture $Sc = 10$.

The second group deals with the remeshed particles method. At each time step, the velocity field is communicated from the first group to the second group in order to advect the particles with a fourth order Runge-Kutta time scheme. Using this partitioning, we avoid the load balancing issues highlighted in [6] where the partitioning of the particle mesh follows the velocity mesh. However, with this parallel implementation, both groups of processors works in an alternate fashion as the second group needs the velocity field to advect particles. A way to enhance the computational performance of this implementation is to use an overloading strategy thanks to DLB, a library that improves the performance of hybrid applications by addressing the load imbalance at runtime [8]. With this strategy, all the resources are used for both solvers. Figure 1 shows the workflow when overloading resources. Code1.x represents the tasks for solving Navier-Stokes equations and Code2.x represents the tasks for the remeshed particles method. When Code1.x is running, Code2.x is in a *DLB.node.barrier*, which means that its processes do not do a busy wait and consequently,

do not consume cycles. At the end of Code1.x, the communication of the velocity field is done. Code1.x enters in the *DLB.node.barrier* and Code2.x does its tasks.

4. High Schmidt scalar advection in a turbulent flow

A round jet in transition to turbulence is considered. The flow configuration is defined by its inlet velocity profile. At the inlet, the mean velocity field is non zero only for the streamwise component, which is given by a hyperbolic tangent profile:

$$\mathbf{u}_i = \frac{U_1 + U_2}{2} - \frac{U_1 - U_2}{2} \tanh\left(\frac{R}{4\Gamma_0}\left(\frac{r}{R} - \frac{R}{r}\right)\right)$$

where U_1 is the centerline velocity, U_1 is a small co-flow, Γ_0 is the momentum thickness of the initial shear layer, r the radial coordinates, and R the initial jet radius. The REYNOLDS number is fixed at $Re = U_1 \cdot (R/\nu) = 2{,}000$. A forcing term is added to accelerate the transition. This forcing is first composed by a random part only added in the shear layer of the jet to the three velocity components. It follows a Passot-Pouquet spectrum with an amplitude set to 10% of U_1. This forcing is then complemented by a deterministic part, which consists in a varicose axissymmetric excitation.
Figure 2 shows instantaneous contours of scalars for values of SCHMIDT number equal to 1 and 10, with 960 CPUs used. The velocity mesh consists of 62M elements and the particles mesh of 115M particles.

References

[1] A. Pushkarev, G. Balarac, W. J. T. Bos, Reynolds and Prandtl number scaling of viscous heating in isotropic turbulence, Physical Review Fluids 2 (8) (2017) 084606. doi:10.1103/PhysRevFluids.2.084606.

[2] I. Calmet, J. Magnaudet, High-Schmidt number mass transfer through turbulent gas–liquid interfaces, International Journal of Heat and Fluid Flow 19 (5) (1998) 522–532. doi:10.1016/S0142-727X(98)10017-6.

[3] O. Lehmkuhl, G. Houzeaux, H. Owen, G. Chrysokentis, I. Rodriguez, A low-dissipation finite element scheme for scale resolving simulations of turbulent flows, Journal of Computational Physics 390 (2019) 51–65. doi:10.1016/j.jcp.2019.04.004.

[4] J.-B. Lagaert, G. Balarac, G.-H. Cottet, Hybrid spectral-particle method for the turbulent transport of a passive scalar, Journal of Computational Physics 260 (2014) 127–142. doi:10.1016/j.jcp.2013.12.026.

[5] C. Mimeau, S. Marié, I. Mortazavi, A comparison of semi-Lagrangian vortex method and lattice Boltzmann method for incompressible flows, Computers & Fluids 224 (2021) 104946. doi:10.1016/j.compfluid.2021.104946.

[6] S. Santoso, J.-B. Lagaert, G. Balarac, G.-H. Cottet, Hybrid particle-grid methods for the study of differential diffusion in turbulent flows, Computers & Fluids 227 (2021) 105018. doi:10.1016/j.compfluid.2021.105018.

[7] G.-H. Cottet, J.-M. Etancelin, F. Perignon, C. Picard, High order semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues, ESAIM: Mathematical Modelling and Numerical Analysis 48 (4) (2014) 1029–1060. doi:10.1051/m2an/2014009.

[8] M. Garcia, J. Labarta, J. Corbalan, Hints to improve automatic load balancing with LeWI for hybrid applications, Journal of Parallel and Distributed Computing 74 (9) (2014) 2781–2794. doi:10.1016/j.jpdc.2014.05.004.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Enhancing Insight Into Turbulent Lifted Hydrogen Jet Flames Using a Reynolds Stress, Stretched Flamelet Model

Chengpei Wu[a,*], Junfeng Yang[a] and Xiaojun Gu[b]

[a] University of Leeds, School of Mechanical Engineering, Leeds, LS2 9JT UK
[b] STFC Daresbury Laboratory, Scientific Computing Department, Warrington, WA4 4AD UK

ARTICLE INFO†

Keywords:
Turbulence;
Jet Flame;
Hydrogen;
Liftoff Height

ABSTRACT

This study provides a numerical study of turbulent lifted hydrogen jet flames by employing a Reynolds stress, stretched flamelet model. Computations were conducted on turbulent lifted hydrogen jet flames using several pipe diameters with a range of jet exit velocities and different pipe exit shapes. The computed isotherms and the volumetric heat release rate contour lines are used to determine the thermal and reaction zone liftoff heights. By comparing computational results with experimental data, the limitations inherent in previous experiments concerning liftoff height are elucidated. The scrutinizing of the lifted flame base enhances the understanding of the dynamics governing hydrogen jet flames.

1. Introduction

Turbulent hydrogen jet flames exhibit intricate interactions between fluid dynamics, and chemical kinetics. Understanding the liftoff height of turbulent hydrogen jet flames is crucial for optimizing combustion efficiency, controlling emissions, and enhancing the performance of various combustion devices [1]. However, definitions of liftoff height vary across the literature, reflecting discrepancies among different research groups, particularly for hydrogen flames, which have relatively low visibility compared to flames produced by hydrocarbon fuels such as methane or propane.

In the present study, a second-order Reynolds stress turbulence model [2], in conjunction with the mixedness-reaction strained flamelet combustion model initially developed for methane [3], is adopted. This approach incorporates adjustments to accommodate the unique combustion characteristics of hydrogen. A parametric study of different pipe shapes and sizes, along with a range of jet exit velocities, is conducted to establish a precise and universally applicable standard for measuring liftoff height, considering the specific temperature and heat release rate conditions of prior experimental work [4].

† This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02448 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
ml18c25w@leeds.ac.uk (C. Wu); J.Yang@leeds.ac.uk (J. Yang); xiaojun.gu@stfc.ac.uk (X. Gu)
ORCID(s): 0009-0003-2344-0009 (C. Wu); 0000-0003-4401-8061 (J. Yang); 0000-0001-9310-1465 (X. Gu)

2. Methodology and simulation setup

2.1. Turbulence model

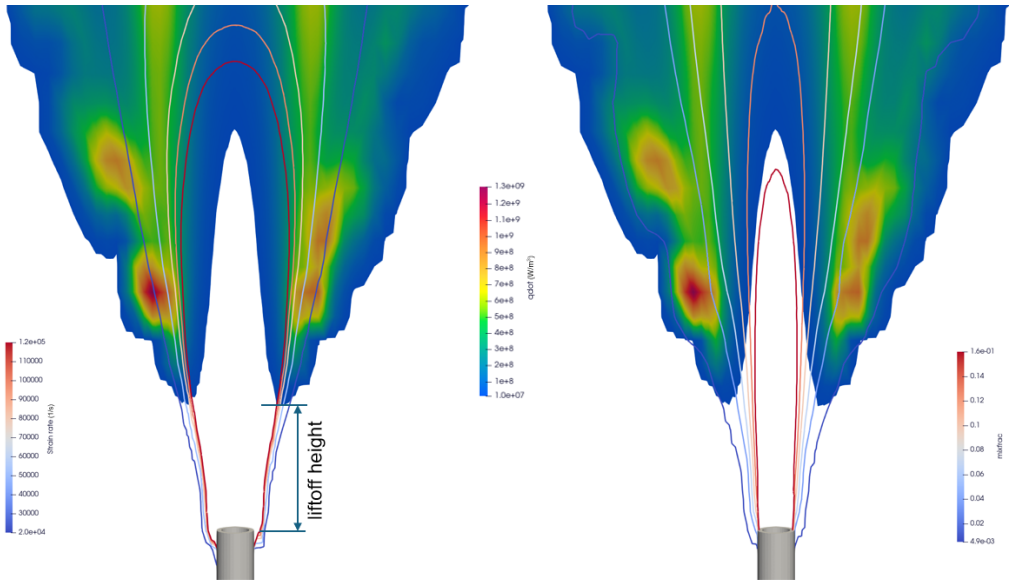
The Favre-averaged Navier-Stokes equations, coupled with the energy equation in term of temperature, are utilized to compute both velocity and temperature fields. These equations are complemented by the second-order turbulence model of the Launder-Reece-Rodi (LRR) turbulence model [2] to address Reynolds stresses. Turbulent heat fluxes in temperature equation are determined through second-order closure [5]. Additionally, the temperature variance equation is solved to build the reactedness distribution function. To model the mixing process, a set of equations for mixture, employing second-order closures [5], are integrated. The counter-gradient effects are embedded in the turbulent heat flux equations [6].

For problems involving obvious free jets, such as circular jets, the LRR model can better capture the anisotropy of turbulence and thus provide in-depth insights into complex flow structures, especially in engineering in applied and basic research.

2.2. Combustion model

The mixedness-reactedness flamelet model, proposed by Bradley et al. [3], serves as the foundation for our turbulent combustion analysis in the present study. It facilitates the determination of the mean volumetric heat release rate within the temperature equation.

As fuel is introduced into the airflow through a pipe, the initial high aerodynamic strain rate dampens both diffusion and premixed flamelets. As the fuel disperses downstream and mixes with air, the strain rate gradually decreases. This reduction allows for premixed combustion to take place, as



(a) 2D contour lines of the mean strain rate.

(b) 2D contour lines of the mixture fraction.

Figure 1: The base of the turbulent lifted hydrogen jet from a circular pipe of a diameter of 2.9 mm with a jet exit velocity of 765 m/s. In the background, the mean volumetric heat release contour field is shown.

the strain rates required to quench premixed flamelets are considerably higher than those for diffusion flamelets.

In the mixedness-reactedness flamelet model of lifted turbulent hydrogen jet flames, stretched premixed hydrogen flamelets with varying equivalence ratios are used to compute the mean heat release rate. These flamelets are generated using the opposed flame reactor in CHEMKIN [7], employing detailed chemistry comprising 9 species and 20 reaction steps. The extinction strain rates are also determined using the counterflow configuration.

To integrate laminar flamelets with detailed chemistry into turbulent flames, the assumed beta-probability distribution functions for mixedness and reactedness are employed based on their corresponding mean and variance values. To incorporate the effects of turbulence stretch on laminar flamelets, a Gaussian distribution of strain rates, defined by mean and variance values on the Taylor microscale of turbulence [8], as observed in direct numerical simulations by Yeung et al. [9] on both random and material surfaces in constant density, homogeneous, isotropic turbulence.

2.3. Simulation setup

The open-source software package Code_Saturne [10] were used a platform to conduct the computational tasks. Version 8.0 of the code incorporates various turbulence models, including Reynolds stress equations and turbulent heat flux equations. The combustion model was integrated into the software through user-defined subroutines, acting as supplementary source terms for both temperature and its variance equations.

Our simulations eschewed symmetry and steady-state assumptions in favor of comprehensive three-dimensional, time-dependent computations. The size of the computational domain was contingent upon factors such as pipe dimensions and jet exit velocity. The most extensive domain measured 3m x 3m x 7m and comprised approximately 4 million grid cells. Except for the pipe wall, all other boundaries are free inlet or outlet. During computation, a time step of 10^{-5} to 10^{-4} second was employed. The simulations extended beyond one second to ensure the full development of the flame.

The average velocity profile at the simulated inlet was determined based on experimental measurements from previous studies on similar flow configurations [4]. Turbulent kinetic energy and dissipation rates are suggested using standard empirical correlations based on average flow velocity and pipe diameter.

Specific inlet conditions are as follows

$$TKE_{inlet} = 1.5 \cdot \left(U_{mean} \cdot 0.16 \cdot \mathrm{Re}^{-\frac{1}{8}}\right)^2, \quad (1)$$

$$Epsilon_{inlet} = \frac{\left(C_{\mu}^{0.25} \cdot \sqrt{TKE_{inlet}}\right)^3}{0.07 \cdot diameter}. \quad (2)$$

3. Results and discussion

The computations have been carried out for a range of jet exit velocities from a pipe of different sizes and shapes. Shown in Fig. 1 is the base of the turbulent lifted hydrogen jet from a circular pipe of a diameter of 2.90 mm with a jet exit velocity of 765 m/s. The flame is stable after 1 s after ignition. The contour lines of the mixture fraction in the plane through the pipe centerline superimposed on the mean volumetric heat release rate contour field are presented in Fig. 1b to illustrate the flame structure at the base. The flame is lifted with a liftoff height indicated in the figure, which depends on the value of the mean heat release. A lower mean heat release rate corresponds to a short liftoff height. Clearly, in the experimental measurements, the value of the liftoff height will be affected by the instruments used and environmental conditions, particularly as the visibility of a hydrogen flame is low. The liftoff heights will be measured with different values of the mean heat release rate from the computational results across a wide range of conditions to compare with the available experimental data. Additionally, the possible causes for discrepancies in the experimental measurements will be explored.

The tip of the lifted jet flame anchors on the mixture fraction with a value of 0.0436, which corresponds equivalence rate of 1.6 for the premixed hydrogen flame. At this equivalence ratio, the laminar hydrogen flame has the maximum laminar burning velocity. The contour goes through the maximum mean heat release region as shown in Fig. 1b. The contour lines of the mean strain rate around the flame base are shown in Fig. 1a. The mean strain is so high that no sufficient laminar flamelets can survive. The flame stabilizes at the contour with a value of 15,000 (1/s), which is slightly lower than the maximum laminar extinction strain rate of the mixture, which is about 17,540 (1/s). The effect of the strain rate on the liftoff height will be further examined with different pipe shapes and sizes.

Shown in the Fig. 2 is the velocity streamline of the turbulent lifted hydrogen jet from a circular pipe of a diameter of 2.90 mm with a jet exit velocity of 521 m/s. The streamlines indicate that the velocity predominantly directs towards the flame's root, where it is denser, thereby accentuating the entrainment effect. Near the flame's initiation zone, this entrainment draws unburnt air into the flame, enhancing combustion efficiency and flame stability.

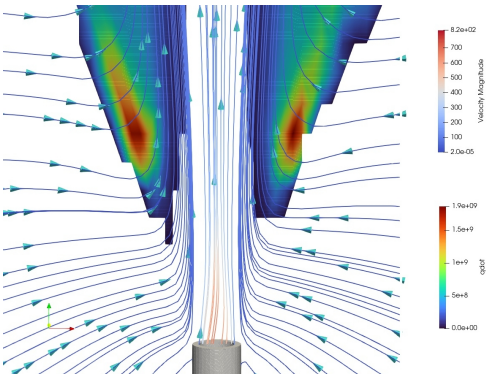


Figure 2: Velocity streamlines of a lifted hydrogen jet flame with a 2.90 mm diameter and an inlet velocity of 521 m/s.

4. Conclusion

The study employs a Reynolds stress, stretched flamelet model to investigate turbulent lifted hydrogen jet flames, focusing on liftoff height and stabilization mechanisms. Detailed analysis of the flame base structure is conducted across various conditions. Factors influencing liftoff height will be thoroughly analyzed.

Acknowledgements

We extend our appreciation to the University of Leeds for providing access to the ARC4 computer facility, and to the STFC Scientific Computing Department for granting access to the SCARF computer facility.

References

[1] D. Bradley, P. H. Gaskell, X. Gu, A. Palacios, Jet flame heights, lift-off distances, and mean flame surface density for extensive ranges of fuels and flow rates, Combustion and Flame 164 (2016) 400–409. doi:10.1016/j.combustflame.2015.09.009.

[2] B. E. Launder, G. J. Reece, W. Rodi, Progress in the development of a Reynolds-stress turbulence closure, Journal of Fluid Mechanics 68 (3) (1975) 537–566. doi:10.1017/S0022112075001814.

[3] D. Bradley, P. Gaskell, A. Lau, A mixedness-reactedness flamelet model for turbulent diffusion flames, in: Symposium (International) on Combustion, Vol. 23, 1991, pp. 685–692. doi:10.1016/S0082-0784(06)80317-6.

[4] T. G. Kalghatgi, Lift-off Heights and Visible Lengths of Vertical Turbulent Jet Diffusion Flames in Still Air, Combustion Science and Technology 41 (1-2) (1984) 17–29. doi:10.1080/00102208408923819.

[5] M. M. Gibson, B. E. Launder, Ground effects on pressure fluctuations in the atmospheric boundary layer, Journal of Fluid Mechanics 86 (3) (1978) 491–511. doi:10.1017/S0022112078001251.

[6] D. Bradley, P. Gaskell, X. Gu, Application of a reynolds stress, stretched flamelet, mathematical model to computations of turbulent burning velocities and comparison with experiments, Combustion and Flame 96 (3) (1994) 221–248. doi:10.1016/0010-2180(94)90011-6.

[7] ANSYS CHEMKIN 17.0 (15151), ANSYS Reaction Design: San Diego, 2016. [link].
URL https://personal.ems.psu.edu/~radovic/ChemKin_Theory_PaSR.pdf

[8] D. Bradley, A. K. C. Lau, M. Lawes, F. T. Smith, Flame stretch rate as a determinant of turbulent burning velocity, Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences 338 (1650) (1992) 359–387. doi:10.1098/rsta.1992.0012.

[9] P. Yeung, S. Girimaji, S. Pope, Straining and scalar dissipation on material surfaces in turbulence: Implications for flamelets, Combustion and Flame 79 (3-4) (1990) 340–365. doi:10.1016/0010-2180(90)90145-H.

[10] A. Frederic, N. Mehitoua, S. Marc, Code Saturne: A Finite Volume Code for the computation of turbulent incompressible flows - Industrial Applications, International Journal On Finite Volumes 1 (1) (2004).
URL https://hal.archives-ouvertes.fr/hal-01115371

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Parallel Adaptive High-Resolution Simulation of Rotating Detonation Engines in 3D

Han Peng[a], Ralf Deiterding[b,*]

[a]Xiamen University, School of Aerospace Engineering, 422 South Siming Road, 361005 Xiamen, China
[b]University of Southampton, School of Engineering, University Road, Southampton, SO17 1BJ, UK

ARTICLE INFO[†]

Keywords:
Combustion Simulation;
Detailed Chemical Kinetics;
Parallel Adaptive Mesh Refinement;
Shock-Capturing;
Curvilinear Meshes

ABSTRACT

Simulations of rotating detonation engines are still dominated by solvers on uniform or statically refined meshes. Here, we demonstrate the application of 3D parallel block-structured adaptive mesh refinement to this problem class. The computations employ a generic shock-capturing curvilinear high-speed combustion solver within the parallel adaptive mesh refinement framework AMROC. The ability to not only capture the rotating waves effectively, but to resolve sub-scale phenomena down to the cellular structures, intrinsic to detonation propagation, demonstrates the potential of the approach.

1. Introduction

Rotating detonation engines (RDEs) employ one or multiple detonation waves to burn continuously injected propellants at the base of an annular chamber. Each detonation spins circumferentially, creating a self-sustained reaction front followed by high-pressure gas. As the burned gases are exhausted in the axial direction, they generate thrust. RDEs are of considerable interest nowadays as they have the potential to operate thermodynamically more efficiently than conventional pressure-constant combustion engines. RDEs can also operate over a wide range of speeds and pressures, which makes them attractive for use in a variety of applications, e.g., in aerospace propulsion and land-based power generation systems.

Investigating the internal flow field of RDEs in experiments is very challenging as it is difficult to capture the details of waves propagating at velocities of about 2000 m/s. Numerical simulations provide an alternative, however they also face serious obstacles. To be sufficiently sensitive for reliable detonation prediction, it is necessary to employ detailed chemical reaction models. Since practical RDEs generally mix propellants only in the combustion chamber, diffusion cannot be neglected. While using the Navier-Stokes equations with full chemistry models seems paramount, the resolution requirements to capture the instationary detonation waves accurately make in particular reliable 3D simulations very expensive.

An effective approach in mitigating the computational cost is dynamic adaptive mesh refinement (AMR). Because of the need to handle an annular cylindrical chamber, the number of reported 3D simulations of RDEs, that have applied on-the-fly mesh adaptation techniques, is comparatively small. Simply assuming all propellants to be premixed, usually the inviscid Euler equations have been solved, in most cases with a simplified one-step reaction model. Only recently, Pal et al. [1] adopted the commercial CFD code CONVERGE with a full chemistry model for unsteady Reynolds-averaged Navier-Stokes equations, albeit with a cut-cell Cartesian AMR method.

2. Methods

Here, we conduct 3D numerical simulations of an RDE using an extended version of our parallel bock-structured finite volume AMR framework AMROC [2]. AMROC has recently been extended for body-fitted curvilinear meshes defined by a mapping function [3]. For RDE simulation, the multi-component Navier-Stokes equations with a detailed chemical model are solved as governing equations. The HLLC (Harten-Lax-van Leer contact) scheme is applied to approximate the inviscid fluxes, where facet-dependent rotation matrices are used to rotate the velocities to align with the 3D curvilinear mesh. A second-order-accurate MUSCL-Hancock method with mapping-dependent spatial reconstruction is applied for the temporal update. The viscous fluxes are calculated at each face in physical space through the coordinate transformation and are incorporated with a conventional second-order-accurate finite volume approach. As it is common for detonation problems, the chemically reactive source terms are integrated cell-wise in an operator

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02449 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
han.peng@xmu.edu.cn (H. Peng); r.deiterding@soton.ac.uk (R. Deiterding)
ORCID(s): 0000-0003-4503-360X (H. Peng); 0000-0003-4776-8183 (R. Deiterding)

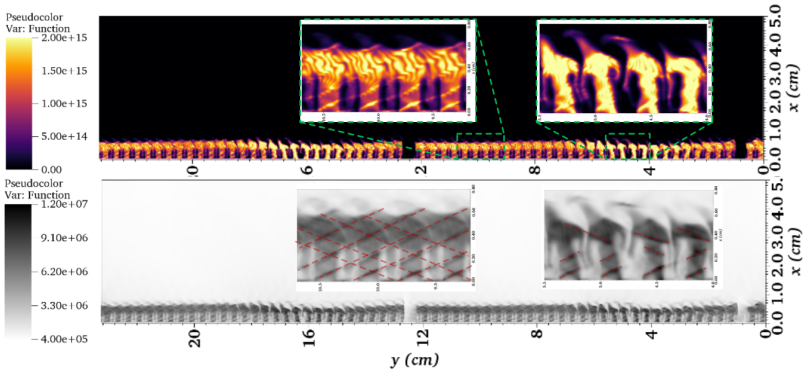


Figure 1: Unwrapped slices of pressure (bottom) and heat release rate (top) during half a rotation cycle.

splitting scheme. A semi-implicit generalized Runge–Kutta method of fourth order (GRK4A) is adopted to integrate the chemical kinetics [4].

AMROC employs a patch-wise refinement strategy, where the entire domain comprises a collection of smaller discrete elements known as blocks. Within these blocks, cells are dynamically flagged using specified refinement criteria. These flagged cells are subsequently grouped into a region of various-sized rectangular blocks. Once the refined cells are created from their parent, namely coarser cells, a hierarchy of embedded grid patches with multiple levels is established. Data between different refinement levels is transferred by averaging and prolongation operations that consider the geometric mapping. To ensure a fully conservative scheme, a flux correction approach is applied and equally extended to handle facets of variable area [3].

For distributed memory parallelization with MPI, the hierarchical mesh is partitioned on-the-fly to processors using a space-filling curve in computational space [2]. A rigorous domain decomposition approach is applied, in which higher refinement levels follow the distribution of the base level cells, however increased workload from spatial and temporal refinement is considered whenever the hierarchical mesh is recreated. The regridding and redistribution process occurs at a specified frequency, ensuring that the region of interest is dynamically captured by the highest-level mesh.

3. Results

A 3D annular model of an RDE running on ethylene and oxygen is simulated. The outer diameter of the chamber is 75 mm, the channel width is 1 mm and the axial height is 50 mm. This chambers corresponds to an actual experiment from our laboratory [5], although the detailed plenum geometry and also plug and aerospike nozzles at the exit are not considered. Initially, the chamber is filled with air at atmospheric conditions. A layer of stoichiometric ethylene-oxygen is initialized with a height of 10 mm. A patch with an analytic 1D detonation wave solution is then used to artificially generate a single detonation wave in the first cycle. A stoichiometric ethylene-oxygen mixture with 20% nitrogen dilution is injected from slots at the head plane. A reduced ethylene/oxygen reaction mechanism [6] with 10 species and 10 reactions is utilized to model the chemistry. Nitrogen is treated as an inert gas. A velocity inlet is used for the injection and the top side is set as an extrapolation outflow boundary condition. The diluted mixture is injected at 300 K and 2 bar, with an inflow velocity of 200 m/s. The average combined inlet mass flow rate is measured at 42.1 g/s. All other boundaries are treated as adiabatic slip walls, thereby neglecting any viscous boundary layers. Because detonation waves in gases propagate at a velocity of around 2000 m/s, this is a justifiable and common simplification in RDE simulations.

Figure 1 depicts the unwrapped plane in the middle of the channel, showcasing two distinct structures observed during the detonation propagation process throughout half of a rotation cycle. The discrete injection leads to flow disturbances ahead of the detonation, and the mixing between the mixture jets and their interval determines the detonation strength locally. The reflected waves are enhanced in the region where the mixture is present. The computational resolution is sufficient to capture the “cells” formed by pressure waves propagating at regular intervals perpendicular to the detonation front. Their size is estimated to be 1.5 mm. These numerical "soot foils" offer a visual representation of the internal wave structure. Weaker reflected waves are observed

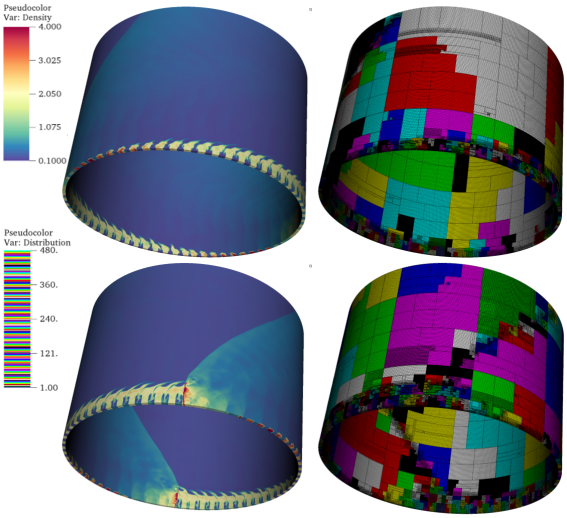


Figure 2: Two snapshots of the evolving density (left) and processors distribution (right) in the 3D RDE.

in regions where mixing is incomplete, which leads to the disappearance of the cellular structure.

The computation was run with two additional refinement levels and it uses approximately 11.6 M to 12.3 M instead of 94.4 M cells in the uniform case. The calculation is performed on 480 cores (Intel Xeon E5-2670 2.0GHz). Typical run times for a simulated time of 1 ms were approximately 12 days wall clock time. Two snapshots of density and processor distribution are displayed in Fig. 2. The images visualize how, after a transient early state without clear detonation (upper row), two stable detonation waves have developed again (lower row). The adaptive mesh clearly follows the fronts at the highest level (right) and most processors are utilized in regions where the workload is high. Note that the number of self-sustaining periodic waves dependent critically but sensitively on the flow conditions and it is one of the most important predictions of RDE simulations.

References

[1] P. Pal, S. Demir, S. Som, Numerical analysis of combustion dynamics in a full-scale rotating detonation rocket engine using large eddy simulations, Journal of Energy Resources Technology 145 (2) (2023) 021702. doi:10.1115/1.4055206.

[2] R. Deiterding, Block-structured adaptive mesh refinement-theory, implementation and application, in: Esaim: Proceedings, Vol. 34, EDP Sciences, 2011, pp. 97–150. doi:10.1051/proc/201134002.

[3] H. Peng, C. W. Atkins, R. Deiterding, A solver for simulating shock-induced combustion on curvilinear adaptive meshes, Computers & Fluids 232 (2022) 105188. doi:10.1016/j.compfluid.2021.105188.

[4] R. Deiterding, A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains, Computers & Structures 87 (11-12) (2009) 769–783. doi:10.1016/j.compstruc.2008.11.007.

[5] H. Law, T. Baxter, C. N. Ryan, R. Deiterding, Design and testing of a small-scale laboratory rotating detonation engine running on ethylene-oxygen, in: AIAA Propulsion and Energy 2021 Forum, 2021, p. 3658. doi:10.2514/6.2021-3658.

[6] D. J. Singh, C. J. Jachimowski, Quasiglobal reaction model for ethylene combustion, AIAA Journal 32 (1) (1994) 213–216. doi:10.2514/3.11972.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

DNS of a Hydrogen Flame Interacting With Homogeneous Isotropic Turbulence Maintained by a Deterministic Force

Yifan Xu[a], Jian Fang[b,*], Zhen Lu[a], Xiaojun Gu[b] and Zhi X. Chen[a]

[a]State Key Laboratory of Turbulence and Complex Systems, College of Engineering Peking University, 100871, Beijing, China
[b]Scientific Computing Department, Science and Technology Facility Council (STFC) Daresbury Laboratory, Warrington, WA4 4AD, UK

ARTICLE INFO†

Keywords:
Premixed Hydrogen Flame;
Homogeneous Isotropic Turbulence;
Deterministic Force

ABSTRACT

Studying the interplay between a hydrogen flame and turbulence is crucial for the advancement of next-generation carbon-neutral combustion systems. In our present work, we conduct a series of direct numerical simulations (DNS) to investigate the dynamics of a premixed hydrogen flame interacting with the compressible homogeneous isotropic turbulence (HIT) maintained by a deterministic force under different pressure and turbulence intensity. Under this particular forcing method applied to turbulence at large scales, the relationship between the forcing intensity and the resulting fluctuating velocity aligns well with the experimental results. In our study, we compared the normalized turbulent burning velocity of hydrogen flames under different conditions, verified the common occurrence of bending effects at elevated pressures and validated existed turbulent burning velocity models. To further explore the dynamics of the HIT-flame interaction and fully leverage the advantages of high-precision direct numerical simulations, we analyzed several flame behaviors such as stretch and instability. The probability density functions (PDF) for the tangential strain rate and curvature are displayed and the results indicate a strong correlation between the flame surface structure and the turbulence generated by the large-scale forcing.

1. Introduction

Hydrogen energy has always been crucial for low-carbon combustion. To better utilize hydrogen energy, it's essential to explore the combustion mechanisms of hydrogen, especially its combustion characteristics in turbulent flow. Direct Numerical Simulation (DNS) can provide detailed insights into combustion and flow dynamics. Song et al. analyzed statistics of flame speed and diffusion effects of turbulent hydrogen flame [1, 2]. Lu and Yang carried out a series of DNS of lean H_2-air turbulent flames and proposed a turbulent burning velocity model [3, 4]. Building upon these previous works, we developed a turbulent forcing method adapted to our inhouse compressible solver. We used this method to compute turbulent premixed flames under different pressures and turbulence intensities, investigating physical phenomena and validating turbulent burning velocity models.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02450 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
yifanxu@pku.edu.cn (Y. Xu); jian.fang@stfc.ac.uk (J. Fang); zhen.lu@pku.edu.cn (Z. Lu); xiaojun.gu@stfc.ac.uk (X. Gu); chenzhi@pku.edu.cn (Z.X. Chen)
ORCID(s): 0009-0005-3498-3265 (Y. Xu); 0000-0003-2474-1762 (J. Fang); 0000-0002-6729-3771 (Z. Lu); 0000-0001-9310-1465 (X. Gu); 0000-0002-1149-1998 (Z.X. Chen)

2. Numerical methods

Our simulations are conducted using an in-house finite-difference code, Advanced flow Simulator for Turbulence Research (ASTR), which has been used for several previous studies [5, 6, 7].

2.1. Forcing method

To maintain isotropic turbulence in our computations, we need to apply forcing to sustain turbulent fluctuations. Here, we have developed a large-scale forcing method adapted to our solver. This forcing method applies random forces to our cells in the form of sine and cosine functions within the computational domain, which is a cubic box L^3, while fixing $F_c = 0$ to ensure divergence-free conditions. The magnitude of forcing is controlled by an initially given parameter F_r for our code. The quantities ϖ_{ij} and ω_{ij} are two random variables ranging from $[-1, 1]$. The equations to solve are

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = \mu\nabla^2\boldsymbol{u} + \frac{1}{3}\mu\nabla(\nabla\cdot\boldsymbol{u}) + \rho\boldsymbol{f}, \quad (1)$$

$$f_i = a_{ij}\sin\left(2\pi\frac{x_j}{L}\right) + b_{ij}\cos\left(2\pi\frac{x_j}{L}\right), \quad (2)$$

$$a_{ij} = \begin{cases} F_r\varpi_{ij} & i- \neq -j \\ F_c\varpi_{ij} & i = j \end{cases}, b_{ij} = \begin{cases} F_r\omega_{ij} & i- \neq -j \\ F_c\omega_{ij} & i = j \end{cases}. \quad (3)$$

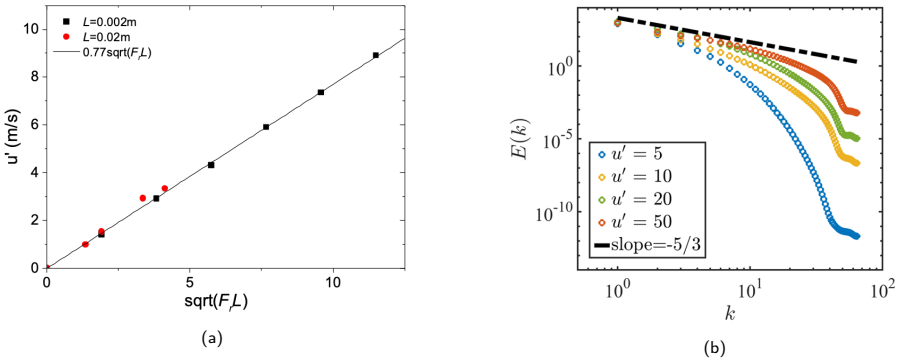


Figure 1: (a) Relationship between u' and $\sqrt{F_r L}$ in different computing domain. (b) Turbulent spectra with different u'.

Figure 1a shows the linear relationship between $\sqrt{F_r L}$ and rms velocity fluctuation u' obtained with our forcing method. Figure 1b shows the turbulence spectra with different u', observing that the turbulence spectra align closely with the -5/3 power law.

2.2. Case configurations

Figure 2 presents our case setup. $L_y = L_z = 5.3\delta_L$ (flame thickness), $L_x = 8L_y$. This implies that when we choose different flame thicknesses (e.g., different pressures) for our cases, the computational domain size will change accordingly with the flame thickness variation. The domain is discretized on uniform grid points $N_x \times N_y \times N_z = 12N \times N \times N$. The numerical resolution in all the cases is ensured to resolve the smallest turbulent and flame length scales by the criterion $k_{max}\eta \geq 1.5$ and a minimum of 24 grid points within a flame thickness δ_L, respectively, where $k_{max} = \pi N/L$ is the maximum wave number magnitude in DNS of HIT and η is the Kolmogorov length scale. To meet the criteria, we set $N = 128$ for all our cases. For a specific case, we first compute a laminar one-dimensional premixed flame and scale the state variables profile of the flame surface into the three-dimensional space mentioned above, positioning the flame surface near the outlet. Then, we set up eight turbulent boxes in the x direction with the desired fluctuating velocity already evolved. This initialization aims to minimize the initial time required for reaching the statistically steady state. Eight cases are conducted with $p = 2, 5\ atm$ and $u'/S_L = 2, 5, 10, 20$.

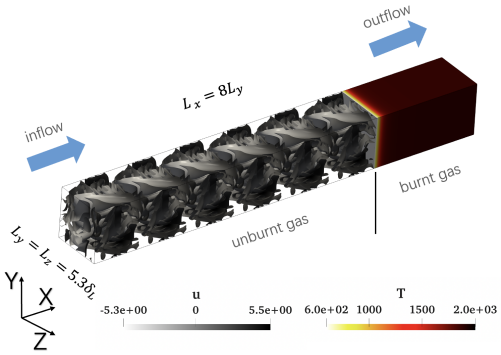


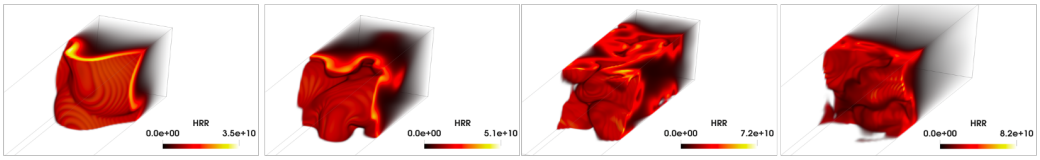
Figure 2: Initial configuration for cases. Iso-surface of vorticity is shown within the grey region and visualized by u magnitude.

3. Results and discussion

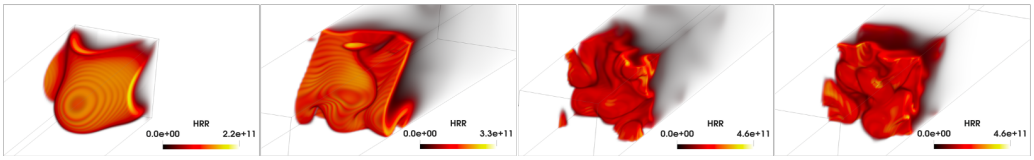
Figure 3 shows the comparison of the flame surface structures after reaching statistical steady state through displaying the isovolume of heat release rate among the cases. The upper line shows the 2 *atm* cases and the bottom line shows the 5 *atm* cases. $u'/S_L = 2, 5, 10, 20$ from left to right column. It can be observed that, in cases with the same pressure, as the turbulence intensity increases, the wrinkling of the flame surface also increases. The heat release rate in high-pressure cases is higher than that in low-pressure cases which matches our expectations.

We calculated the statistical burning velocity of our cases and compared them with the S_T model and observed the expected bending effect phenomenon as shown in Fig. 4. The turbulent burning velocity

$$s_T = \frac{1}{\rho_u Y_{\mathrm{F},u} A_L \left(t_2 - t_1\right)} \int_{t_1}^{t_2} \int_{\Omega} -\dot{\omega}_{\mathrm{F}} dV dt \qquad (4)$$

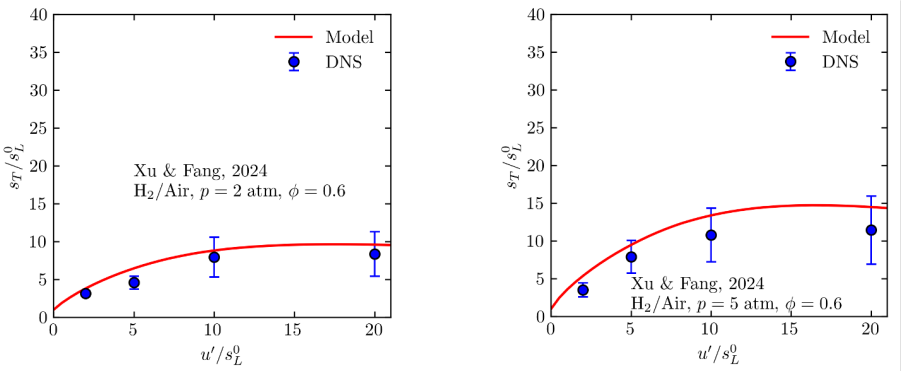


(a) $p = 2\ atm$; $u'/S_L = 2, 5, 10, 20$



(b) $p = 5\ atm$; $u'/S_L = 2, 5, 10, 20$

Figure 3: Flame surface structures displayed by the isovolume of heat release rate for different pressures p.



(a) (b)

Figure 4: Comparison of S_T calculated from DNS and the model.

is defined by the consumption speed. We found that our results generally align well with the Lu and Yang model [3]. This overall validation confirms the rationality of our forcing method and the accuracy of our results.

4. Conclusions

We developed a deterministic forcing method to maintain statistically stationary isotropic turbulence for studying flame/turbulence interaction. By using the deterministic forcing method and the high-order solver, we studied the hydrogen flame/turbulence interaction at different turbulence intensity and pressure. The bending curves of turbulence burning velocity are observed in our cases and the results agree well with the model proposed.

Acknowledgements

JF and XG gratefully acknowledges the support from the Computational Science Centre for Research Communities (CoSeC) and Royal Society (IEC/NSFC/223421). All the simulations were conducted on ARCHER2 and the computing time was award through the UK Turbulence Consortium (EP/R029326/1 and EP/X035484/1).

References

[1] W. Song, F. E. Hernández Pérez, E.-A. Tingas, H. G. Im, Statistics of local and global flame speed and structure for highly turbulent H_2/air premixed flames, Combustion and Flame 232 (2021) 111523. doi:10.1016/j.combustflame.2021.111523.

[2] W. Song, F. E. Hernández-Pérez, H. G. Im, Diffusive effects of hydrogen on pressurized lean turbulent hydrogen-air premixed flames, Combustion and Flame 246 (2022) 112423. doi:10.1016/j.combustflame.2022.112423.

[3] Z. Lu, Y. Yang, Modeling pressure effects on the turbulent burning velocity for lean hydrogen/air premixed combustion, Proceedings of the Combustion Institute 38 (2) (2021) 2901–2908. doi:10.1016/j.proci.2020.06.162.

[4] J. You, Y. Yang, Modelling of the turbulent burning velocity based on Lagrangian statistics of propagating surfaces, Journal of Fluid Mechanics 887 (2020) A11. doi:10.1017/jfm.2019.1081.

[5] J. Fang, F. Gao, C. Moulinec, D. Emerson, An improved parallel compact scheme for domain-decoupled simulation of turbulence, International Journal for Numerical Methods in Fluids 90 (10) (2019) 479–500. doi:10.1002/fld.4731.

[6] J. Fang, Z. Li, L. Lu, An Optimized Low-Dissipation Monotonicity-Preserving Scheme for Numerical Simulations of High-Speed Turbulent Flows, Journal of Scientific Computing 56 (1) (2013) 67–95. doi:10.1007/s10915-012-9663-y.

[7] J. Fang, A. A. Zheltovodov, Y. Yao, C. Moulinec, D. R. Emerson, On the turbulence amplification in shock-wave/turbulent boundary layer interaction, Journal of Fluid Mechanics 897 (2020) A32. doi:10.1017/jfm.2020.350.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Computation of Transport and Chemistry for Combustion Applications in the Code Alya Using Accelerated Architectures

Álvaro Moure[a,*], Alejandro Lamas Daviña[a], Anurag Surapaneni[a] and Daniel Mira[a]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain

ARTICLE INFO[†]

Keywords:
Finite-Element Method;
Combustion;
OpenACC;
Graphics Processing Unit;
Matrix Assembly;
Finite-Rate Chemistry

ABSTRACT

Combustion is a phenomenon present in most of the fundamental industries and sectors that 21st century society depends on, such as energy generation, transportation and heating systems. However, to further understand the complexities of the upcoming challenges; pollution reduction, hydrogen combustion systems, Sustainable Aviation Fuels (SAF), etc, numerical simulations are crucial. Latest supercomputers that come with heterogeneous architectures open new frontiers in the computational capabilities for simulating reacting flows. In the present work, a methodology of porting to the GPU the computation of the species transport using detailed chemistry is explained.

1. Introduction

Combustion is a multi-physics and multi-scale phenomenon which use and understanding is essential in many energy and transportation sectors. Numerical simulations of real scale and real combustion regimes of the devices used by those industries are extremely computationally demanding [1]. Today's cutting edge supercomputers are enabling researchers to reach new horizons such as making high-fidelity models, reducing simplifications and storing and processing huge amounts of data. Particularly in the field of engineering simulations, the CPU-GPU heterogeneous architectures have the potential to reduce the time-to-results simulations by offloading parts of the code that are arithmetically intensive to the GPU [2, 3, 4, 5].

The parallel Finite-Element Method (FEM) based code Alya [6] is a multiphysics modular code that can solve a wide range of engineering problems, from pure structural mechanics to fully coupled turbulent multi-phase reacting flows problems. In the latter problems, the chemical module (`chemic`) is activated to compute the transport and the chemical reactions of the species that are involved in the case (reactants, combustion products, pollutants, etc.). The present work explains the strategy adopted for adapting to the GPU both, the matrix assembly algorithm, involved in FEM [7], and the stiff chemical integration found in the *finite-rate* chemistry approach [8].

2. Methodology

Computational Fluid Dynamics (CFD) focuses on solving numerically the Navier Stokes Equations (NSE) to determine the thermofluid dynamic state of a given flow. Furthermore, when dealing with reacting flows, apart from the NSE, one transport equation for each species participating in the chemical reactions must be added to the system

$$\frac{\partial \left(\rho Y_k\right)}{\partial t} + \nabla \cdot \left(\rho \mathbf{u} Y_k\right) = -\nabla \cdot \left(\rho \mathbf{V}_k Y_k\right) + \dot{\omega_k}, \qquad (1)$$

where the temporal evolution of the species mass fraction "k" is given by the fluid transport, which is decomposed in the convective (second term of the left-hand-side) and diffusion (first term of the right-hand-side) terms, and by the production-consumption of the species "k". The convection term represents the transport of the species "k" due to the bulk motion of the fluid. The diffusion term represents the transport of the species "k" due to molecular diffusion. Lastly, the most right hand side term of the Eq. (1), known as the chemical source term, is where the non-linear chemical reactions are calculated to dictate if the given species are being generated or consumed. The difference between the time-scales of the chemistry and the flow dynamics makes this equation difficult to be numerically solved.

2.1. Operator splitting in reacting flows with stiff chemistry

Operator splitting schemes allow to decompose the Partial Differential Equation (PDE) into decoupled sub-equations that deal only with a portion of the physics [9]. In the `chemic` module of Alya, the first order operator splitting

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02451 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
alvaro.moure@bsc.es (Á. Moure); alejandro.lamas@bsc.es (A.L. Daviña); anurag.surapaneni@bsc.es (A. Surapaneni); daniel.mira@bsc.es (D. Mira)
ORCID(s): 0009-0000-8195-046X (Á. Moure); 0009-0006-9130-2101 (A.L. Daviña); 0000-0001-7026-558X (A. Surapaneni); 0000-0001-9901-7942 (D. Mira)

proposed by Lie-Trotter [10] is implemented, dividing the simulation time-step into a separated evaluation of a transport sub-step followed by a finite-rate chemistry sub-step.

2.2. Element operations of the transport terms

In the transport sub-step, a loop iterates over the elements of the mesh and performs the following operations to assembly the matrix:

1. **Gathering**. Change of coordinates of the needed properties from global to elemental, bringing mass fractions, velocity, and temperature values to the nodes of the element.
2. **Element preparation**. Interpolation of the element nodes values to the Gauss points, using the elemental shape functions.
3. **Element assembly**. Computation of the convective and diffusive terms of the Eq. (1), performing the space integration over the element domain, and projecting it back to the right-hand-side of the element nodes.
4. **Matrix assembly**. Assembly of the global system matrix accumulating the influence of each element in each global node.

Due to the large portion of code needed to perform the 4 steps, many variables have to be declared as private. The registers on the GPU limit the occupancy of the kernel and the performance obtained is not optimal. The strategy followed to get performance on GPUs is partitioning the elemental loop in 2 kernels, one for the computation of the convective term, and other for the diffusive term, performing both of them a partial assembly of the matrix.

2.3. Finite-rate chemistry

The chemical integration sub-step drives to an Initial Value Problem of an autonomous Ordinary Differential Equation (ODE) system of the form:

$$\frac{dY_k}{dt} = \frac{\dot{w}_k}{\rho} \rightarrow \frac{dy}{dt} = f(y) \tag{2}$$

Normally the time-scales involved in the reactions of the combustion mechanism can differ orders of magnitude, making the ODE system very stiff.

ODE stiff methods

In the present work two ODE stiff numerical methods algorithms are adapted to accelerate the calculation using GPUs. Given the modularity of Alya, the implementation has been done in a standalone C++ library, creating a common interface that Alya can use regardless the integration method selected.

- **Backward Differentiation Formula (BDF) methods**. BDF methods are robust and accurate for stiff problems, with the drawback of being computationally expensive. In this case the solver has not been implemented itself, but adapted to run on the GPU, using the well-known package CVODE [11]. Porting CVODE to the GPU involves using the Magma library to solve the linear systems, that appear when solving non-linear problems, in a batched LU factorization [12], and pyJac to compute the Jacobian matrix analytically on the GPU [13].
- **Rosenbrock's methods**. These semi-implicit methods are proven to work well with relatively high stiff problems with the advantage of being faster than the BDF methods. The solvers ROS4 and RODAS from Hairer, et al. [14] has been implemented and parallelized with GPUs. In this case, each GPU thread solves one ODE system, which corresponds to the thermochemical state at each degree of freedom of the domain.

3. Results

3.1. Matrix assembly

Figure 1 illustrates a strong scaling study of the matrix assembly performed on LEONARDO Booster, where each process has exclusive access to one GPU device simultaneously. Compared to the ideal lineal scaling, the results suggest that the followed approach is highly parallel and can take the most of the accelerated partition that the upcoming supercomputers bring with them.

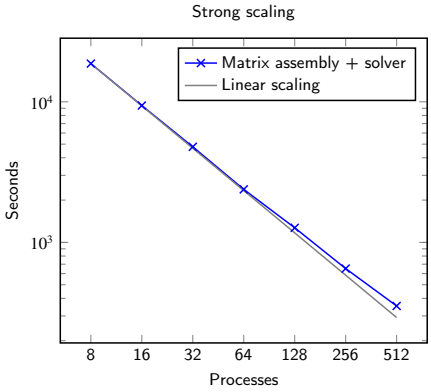
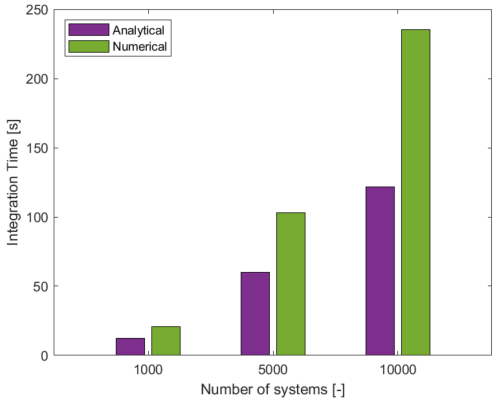
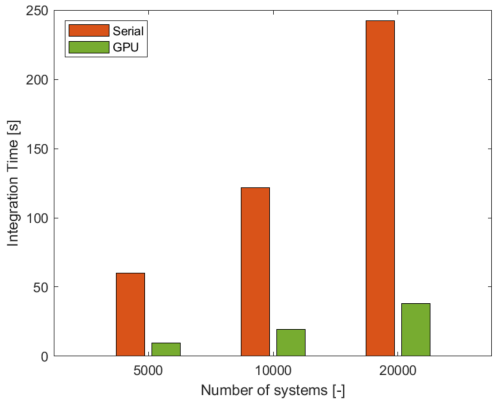


Figure 1: Turbulent premixed flame mixing layer. Alya time to solve 200 timesteps (details in Tab. 1).



(a) CVODE's integration times with analytical and numerical approaches for the Jacobian matrix in the BDF method.



(b) Integration times of serial and GPU implementations of CVODE adaptation.

Figure 2: Integration times of perfectly stirred reactors.

	Seconds	
Proc.	Mat. assembly + solver	Linear
8	18702.88	18702.88
16	9407.03	9351.44
32	4786.37	4675.72
64	2384.23	2337.86
128	1270.60	1168.93
256	651.45	584.47
512	353.27	292.23

Table 1
Timings for the graph in Fig. 1.

3.2. Chemical integration

Figure 2 shows the preliminary results of the GPU adaptation of the CVODE solver in the standalone library, given a chunk of perfectly stirred reactors in a mixture of equivalence ratio of $\phi = 0.8$ for a 53 species and 325 reactions GRI30 combustion mechanism [15]. It is shown that an analytical Jacobian evaluation is faster than the conventional finite differences approach, and that the GPU accelerated solver runs around x6 times faster that a single CPU processor.

4. Conclusions and outlook

The strategy followed to speedup the calculation of the transport terms, focused on the matrix assembly, shows a potential methodology that can be adopted for the rest of the modules of Alya, however, further studies need to be performed. Regarding the GPU acceleration of the chemical integration sub-step in an standalone library, it shows promising results and it will be easily integrated in future versions of Alya given the modularity and the interface design of both the C++ library and Alya.

References

[1] D. Mira, E. J. Pérez-Sánchez, R. Borrell, G. Houzeaux, HPC-enabling technologies for high-fidelity combustion simulations, Proceedings of the Combustion Institute 39 (4) (2023) 5091–5125. doi:10.1016/j.proci.2022.07.222.

[2] K. Spafford, J. Meredith, J. Vetter, J. Chen, R. Grout, R. Sankaran, Accelerating S3D: A GPGPU Case Study, in: Euro-Par 2009 – Parallel Processing Workshops, Lecture Notes in Computer Science (LNTCS, Vol. 6043), Springer Berlin Heidelberg, 2010, pp. 122–131. doi:10.1007/978-3-642-14122-5_16.

[3] F. Piscaglia, F. Ghioldi, GPU Acceleration of CFD Simulations in OpenFOAM, Aerospace 10 (9) (2023) 792. doi:10.3390/aerospace10090792.

[4] S. Rao, B. Chen, X. Xu, Heterogeneous CPU-GPU parallelization for modeling supersonic reacting flows with detailed chemical kinetics, Computer Physics Communications 300 (2024) 109188. doi:10.1016/j.cpc.2024.109188.

[5] S. Desai, Y. J. Kim, W. Song, M. B. Luong, F. E. Hernández Pérez, R. Sankaran, H. G. Im, Direct numerical simulations of turbulent reacting flows with shock waves and stiff chemistry using many-core/GPU acceleration, Computers & Fluids 215 (2021) 104787. doi:10.1016/j.compfluid.2020.104787.

[6] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen,

A. Taha, E. D. Burness, J. M. Cela, M. Valero, Alya: Multiphysics engineering simulation toward exascale, Journal of Computational Science 14 (2016) 15–27. doi:10.1016/j.jocs.2015.12.007.

[7] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, J. Zhu, The Finite Element Method, 3rd Edition, McGraw-Hill Book Company (UK), London, UK, 1977.

[8] C. K. Law, Combustion Physics, Cambridge University Press, 2006. doi:10.1017/CBO9780511754517.

[9] G. Strang, On the Construction and Comparison of Difference Schemes, SIAM Journal on Numerical Analysis 5 (3) (1968) 506–517. doi:10.1137/0705041.

[10] H. F. Trotter, On the product of semi-groups of operators, Proceedings of the American Mathematical Society 10 (4) (1959) 545–551. doi:10.1090/S0002-9939-1959-0108732-6.

[11] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software 31 (3) (2005) 363–396. doi:10.1145/1089014.1089020.

[12] A. Abdelfattah, S. Tomov, J. Dongarra, Progressive Optimization of Batched LU Factorization on GPUs, in: 2019 IEEE High Performance Extreme Computing Conference (HPEC), IEEE, 2019, pp. 1–6. doi:10.1109/HPEC.2019.8916270.

[13] K. E. Niemeyer, N. J. Curtis, C.-J. Sung, pyJac: Analytical Jacobian generator for chemical kinetics, Computer Physics Communications 215 (2017) 188–203. doi:10.1016/j.cpc.2017.02.004.

[14] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Vol. 14 of Springer Series in Computational Mathematics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. doi:10.1007/978-3-642-05221-7.

[15] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, W. C. Gardiner, Jr., V. V. Lissianski, Z. Q, GRI-Mech 3.0.
URL http://www.me.berkeley.edu/gri_mech/

Mini-Symposium 3:

Convergence of Artificial Intelligence and High-Performance Computing for Computational Fluid Dynamics (AI + HPC4CFD PT. 3)

Organizers: Guillaume Houzeaux, Corentin Lapeyre, and Mario Rüttgers

Artificial Intelligence (AI) technologies are penetrating into all sectors of research and industry. They automate and accelerate processes, and uncover new unseen relations in huge datasets. The successful AI+HPC4CFD ParCFD 2022 and 2023 mini-symposia already impressively showed that the Computational Fluid Dynamics (CFD) community drastically benefits from these technologies. AI methods and notably deep learning techniques are used to develop new models for CFD, e.g., reduced-order models, surrogates, and closure models aiming at efficiently modeling complex physics that are otherwise expensive to compute. Furthermore, reinforcement learning algorithms can be used for flow control applications, while receiving feedback from CFD solvers after an action. The quality of these methods is often a function of both the quantity and the accuracy of the underlying data used for training as well as the physical constraints imposed on the training. The generation and processing of high fidelity simulation data necessitates the application of High-Performance Computing (HPC) systems, with an increasing number CFD solvers running on both CPU and GPU partitions. Modular and heterogeneous systems with accelerator and/or specialized AI-components as blueprints for upcoming Exascale systems have the potential to deal with the demands of complex and intertwined simulations and AI-data processing workflows. This minisymposium aims at continuing the successful 2022 and 2023 AI+HPC4CFD minisymposia. It will gather experts in the fields of development and application of parallel CFD methods incorporating novel AI methods, and pure AI method developers contributing to the fields of CFD and HPC alike. It will again offer a platform for discussion and exchange in the context of the convergence of AI and HPC with respect to parallel CFD methods that could benefit from the power of next-generation Exascale computing systems.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Hard Constraint Projection in a Physics Informed Neural Network

Miranda J. S. Horne[a,*], Peter K. Jimack[a], Amirul Khan[b] and He Wang[c]

[a] University of Leeds, School of Computing, Woodhouse Lane, Leeds, LS2 9JT, UK
[b] University of Leeds, School of Civil Engineering, Woodhouse Lane, Leeds, LS2 9JT, UK
[c] University College London, Department of Computer Science, Gower Street, London, WC1E 6BT, UK

ARTICLE INFO[†]

Keywords:
Physics-Informed Machine Learning;
Hard Constraints;
Navier-Stokes Equations;
Non-Linear Partial
Differential Equations

ABSTRACT

In this work, we embed hard constraints in a physics informed neural network (PINN) which predicts solutions to the 2D incompressible Navier-Stokes equations. We extend the hard constraint method introduced by Chen et al. [1] from a linear PDE to a strongly non-linear PDE. The PINN is used to estimate the stream function and pressure of the fluid, and by differentiating the stream function we can recover an incompressible velocity field. An unlearnable hard constraint projection (HCP) layer projects the predicted velocity and pressure to a hyperplane that admits only exact solutions to a discretized form of the governing equations.

1. Introduction

Machine learning provides a promising framework to simulate fluid dynamics at a fraction of the computational cost of traditional numerical methods [2]. Furthermore, the incorporation of domain knowledge into a neural network can improve the prediction accuracy, increase the model's explainability, and result in a neural network that is less reliant on training data. Typically, the incorporation of the physical constraints into a neural network is only weakly enforced, for example, a PINN [3] weakly enforces the governing equation by incorporating a penalty term (often the equation's residuals) into the loss function. In the cases where a physical constraint is strongly imposed, the enforced governing equation is often either linear [1], or an additional conservation law (such as the incompressibility constraint [4]). In this paper we propose a method to strictly enforce the discretized form of a nonlinear partial differential equation, through projection, inspired by the linear analogue used by Chen et al. in 2021 [1].

2. Physics informed neural network (PINN)

We will consider the 2D incompressible Navier-Stokes Eqs. (1), (2), and (3) [5]. The solution of this system is given by the 2D instantaneous velocity ($u(x, y, t)$, $v(x, y, t)$) and the pressure ($p(x, y, t)$) of the fluid, where x, y are the spatial coordinates, and t is the temporal coordinate. The kinematic viscosity of the fluid is given by ν, and the constant density by ρ. That is, we solve for

$$0 = \frac{\partial u}{\partial t} + \left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) - \nu\frac{\partial^2 u}{\partial x^2} - \nu\frac{\partial^2 u}{\partial y^2} + \frac{1}{\rho}\frac{\partial P}{\partial x}, \quad (1)$$

$$0 = \frac{\partial v}{\partial t} + \left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) - \nu\frac{\partial^2 v}{\partial x^2} - \nu\frac{\partial^2 v}{\partial y^2} + \frac{1}{\rho}\frac{\partial P}{\partial y}, \quad (2)$$

$$0 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}. \quad (3)$$

The network proposed is a feed forward neural network (FFNN) that takes as input the coordinates of the system (x, y, t) and outputs a prediction of the stream function (ψ) and the pressure (p) at those coordinates. The velocity components are defined as $u = \partial\psi/\partial y$, $v = -\partial\psi/\partial x$, using the automatic differentiation [6] (AD) built into the network. This method strictly imposes the incompressibility, see Eq. (3) of the system.

As just coordinates alone would be insufficient to predict a unique solution, we use N ground truth solutions (u_i, v_i) to anchor the model's predictions $(\hat{u}_i, \hat{v}_i)$ to our test case. The data error

$$\mathrm{DE} = \frac{1}{N}\sum_{i=1}^{N}(u_i - \hat{u}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(v_i - \hat{v}_i)^2 \quad (4)$$

measures the distance between the ground truth solutions and the network predictions. We calculate the data error only on the velocity as in practice it would be significantly more difficult to measure the pressure of a fluid through the

[†] This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02453 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
scmho@leeds.ac.uk (M.J.S. Horne); p.k.jimack@leeds.ac.uk (P.K. Jimack); a.khan@leeds.ac.uk (A. Khan); he_wang@ucl.ac.uk (H. Wang)
ORCID(s): 0000-0002-4715-4707 (M.J.S. Horne); 0000-0001-9463-7595 (P.K. Jimack); 0000-0002-7521-5458 (A. Khan); 0000-0002-2281-5679 (H. Wang)

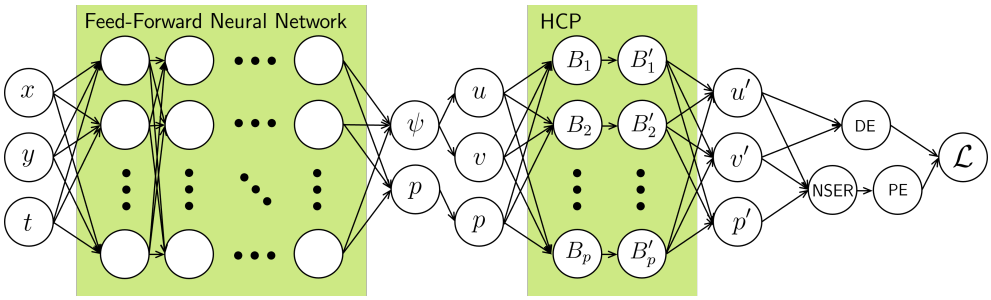


Figure 1: HCP-PINN architecture. The left box is the FFNN with learnable weights and biases, and the right box is the unlearnable HCP.

domain than the velocity when creating the ground truth data set.

Training the FFNN using only the DE would neglect the other knowledge we have of the system, the governing PDE. A PINN [3] appends the loss function to include some measure of the prediction's deviation from the governing equations, the physics error

$$\mathrm{PE} = \frac{1}{M}\sum_{i=1}^{M}(r_i^x)^2 + \frac{1}{M}\sum_{i=1}^{M}(r_i^y)^2. \tag{5}$$

The Navier-Stokes equation residuals (NSER) are found by evaluating the RHS of Eq. (1) and Eq. (2) using AD for M of the network predictions at locations (x_i, y_i, t_i), denoted r_i^x and r_i^y, for $i \in \{1, ..., M\}$.

In some PINN literature (e.g. [3]) the error from the initial condition and boundary conditions are incorporated in the loss function as separately weighted penalty terms. In the models presented here, the values on the spatial and temporal boundaries are incorporated into the data error, such that the loss function is defined only as a weighted sum of the DE and PE.

3. Hard constraint projection (HCP)

The HCP-PINN has the same learnable network and loss function as a PINN, but between the initial prediction of the solution and the loss evaluation there is an unlearnable hard constraint projection layer (see Fig. 1). Following the methodology used by Chen et al. [1], the governing Eqs. (1) and (2) are discretized with a central finite difference scheme for the spatial derivatives, and a backwards finite difference scheme for the temporal derivatives. The discretized forms of the governing equations are then decomposed into two matrices, the constraint matrix A (containing all the constant terms) and the prediction matrix B (containing all of the transient terms) such that multiplying AB recovers the discretized following set of governing equations

$$AB = \begin{bmatrix} 1/\Delta t \\ 1/\Delta t \\ 1/\Delta x \\ 1/\Delta x \\ 1/\Delta x \\ 1/\Delta y \\ 1/\Delta y \\ 1/\Delta y \\ 1/(\Delta x)^2 \\ 1/(\Delta y)^2 \end{bmatrix}^T \begin{bmatrix} u & v \\ -u_{-\Delta t} & -v_{-\Delta t} \\ (1/2)u(u_{+\Delta x} - u_{-\Delta x}) & (1/2)u(v_{+\Delta x} - v_{-\Delta x}) \\ (1/2\rho)P_{+\Delta x} & 0 \\ (-1/2\rho)P_{-\Delta x} & 0 \\ (1/2)v(u_{+\Delta y} - u_{-\Delta y}) & (1/2)v(v_{+\Delta y} - v_{-\Delta y}) \\ 0 & (1/2\rho)P_{+\Delta y} \\ 0 & (-1/2\rho)P_{-\Delta y} \\ \nu(-u_{+\Delta x} + 2u - u_{-\Delta x}) & \nu(-v_{+\Delta x} + 2v - v_{-\Delta x}) \\ \nu(-u_{+\Delta y} + 2u - u_{-\Delta y}) & \nu(-v_{+\Delta y} + 2v - v_{-\Delta y}) \end{bmatrix}. \tag{6}$$

Using tools from linear algebra we can define the projection matrix as $P = I - A^T(AA^T)^{-1}A$. When an arbitrary prediction matrix (B) is multiplied by the projection matrix, $PB = B'$, the outcome (B') is the closest point to B that lies on the hyperplane $AB = 0$. Thus, the predictions after projection satisfy the discretized form of the governing equations exactly. For a proof of this, please see the appendix of the paper by Chen et al. [1]. In practice, each prediction matrix contains only the values from one finite difference stencil, so each of the B_i in Fig. 1 corresponds to each of the input coordinate tuples (x, y, t).

4. Provisional results

To test the HCP-PINN, two models were trained, a PINN with no HCP layer, and a HCP-PINN as depicted in Fig. 1. Both models had 6 hidden layers with 50 neurons each, and were optimized with default settings of the Adam [7] optimizer with a loss function defined as $\mathcal{L} = 0.9\mathrm{DE}{+}0.1\mathrm{PE}$. The hyperbolic tangent was used as the activation function. The discretization used in the hard constraint projection has values $\Delta x = \Delta y = \Delta t = 0.01$.

The models were trained and tested on one dataset of periodic vortex shedding past a bluff body. The data domain is defined by $t \in [0, 10]$, $x \in [1, 8]$, and $y \in [-2, 2]$,

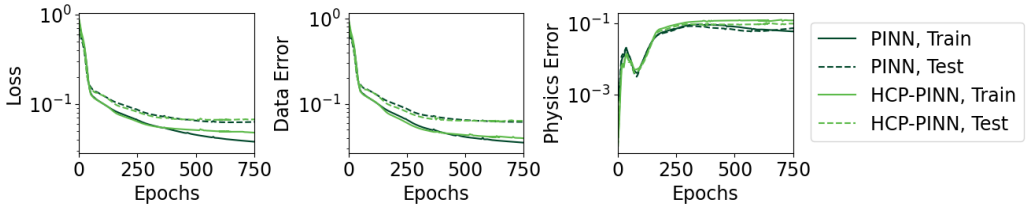


Figure 2: Graphs comparing the value of the loss, data error, and physics error, on the test and training set, during the training of both models.

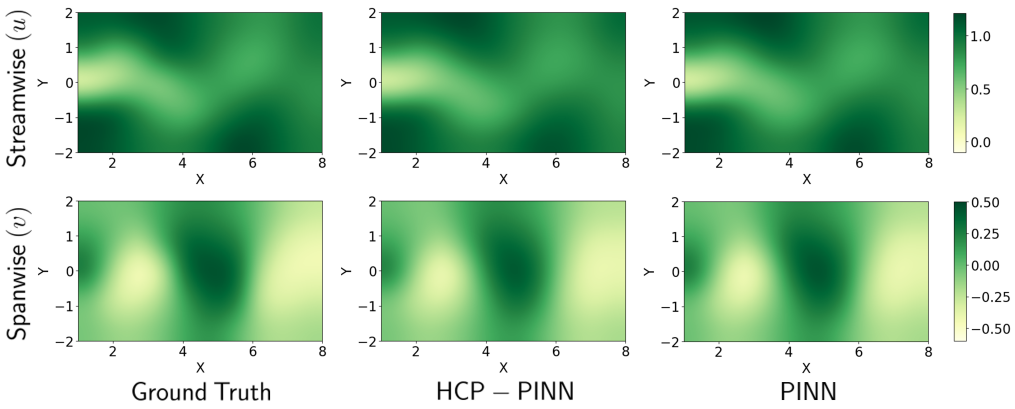


Figure 3: Ground truth velocity field, and corresponding model predictions at epoch 750, of the velocity field at $t = 4.8$.

downstream of the bluff body and with the wake fully realized. The training dataset was randomly, uniformly sampled across the domain with $N = 500$ ground truth data points (230 of which lie on the spatial and temporal boundaries) and $M = 1{,}000$ physics collocation points. The test dataset is selected on a grid with $t = \{1.6, 4.8, 8.0\}$ and $\Delta x = 0.7070, \Delta y = 0.8164$ which both the test DE and the test PE are evaluated on ($N_{test} = M_{test} = 150$).

The training trajectory of the performance of the models can be seen in Fig. 2, and the predicted velocity fields at $t = 4.8$ are displayed alongside the ground truth in Fig. 3. We see that the training trajectories of the two models follow a similar shape, implying that the HCP-PINN optimizes in a similar manner to the vanilla PINN. This is supported by the predictions in Fig. 3, where the two models predict similar flow fields that qualitatively represent the key features of the flow. We would not expect the physics error of the HCP-PINN to be exactly zero, as the governing equation is only exactly obeyed in its discretized form, locally. Unfortunately, we also find that the physics error associated with the HCP-PINN is not consistently lower than for the PINN, which was one of the motivations behind this implementation.

An established issue in the literature on the training of PINNs is the imbalance between loss function terms [8], and it was hoped by the authors that we would find the HCP-PINN less sensitive to the hyperparameter w in the function $\mathcal{L} = (w)\text{DE} + (1 - w)\text{PE}$. We had also anticipated that the HCP-PINN would potentially be less dependent on the quantity and sampling strategy for the physics collocation points, especially since the calculation of the residual at these points is a computational bottle-neck for both models. We report that the HCP-PINN and PINN appear to respond equally sensitively from our studies into this (which have been omitted from the extended abstract for brevity). The authors suspect that the hard constraint projection, despite requiring greater computational resources, has a minimal effect on the predictions made during training, as implied by the similar training trajectories and predictions in Fig. 2 and Fig. 3.

The authors intend to look into modifying the implementation of the HCP-PINN, with the intention of more favorable results. We will look into the execution of the HCP, which uses a low order and potentially unstable discretization, and a non-unique decomposition of the governing equation. We will also look at employing established machine learning methods such as batch training and transfer learning with the goal of fully exploring the potential of this method. We finally note that Chen et al. [1] investigated only one linear PDE when originally proposing this method for hard constraint projection, and it is possible that the HCP method is only appropriate for a subset of PDEs, such as linear or weakly non-linear PDEs.

5. Current work

We aim to refine the HCP-PINN further. The directions stated in the previous section will be our immediate goals, however this model also has the potential to embed the boundary conditions strictly through the use of ghost cells. Additionally, we would like to investigate the model's robustness to noise and outliers, generalisability to other flow regimes, and extrapolation capabilities given only boundary and initial conditions.

References

[1] Y. Chen, D. Huang, D. Zhang, J. Zeng, N. Wang, H. Zhang, J. Yan, Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method, Journal of Computational Physics 445 (2021). doi:10.1016/j.jcp.2021.110624.

[2] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2020). doi:10.1146/annurev-fluid-010719-060214.

[3] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019). doi:10.1016/j.jcp.2018.10.045.

[4] A. T. Mohan, N. Lubbers, M. Chertkov, D. Livescu, Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence, Physical Review Fluids 8 (2023). doi:10.1103/PhysRevFluids.8.014604.

[5] A. R. Paterson, A First Course in Fluid Dynamics, Cambridge University Press, 1983. doi:10.1017/CBO9781139171717.

[6] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of Machine Learning Research 18 (2018). URL http://jmlr.org/papers/v18/17-468.html

[7] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, ArXiv (2017). doi:10.48550/arXiv.1412.6980.

[8] S. Wang, S. Sankaran, H. Wang, P. Perdikaris, An Expert's Guide to Training Physics-informed Neural Networks, ArXiv (2023). doi:10.48550/arXiv.2308.08468.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Drag Correlations for Multiphase Flows Using Artificial Neural Networks

Julian Vorspohl[a,*], Laurent André[a], Mario Rüttgers[a,b] and Wolfgang Schröder[a]

[a]RWTH Aachen University, Chair of Fluid Mechanics and Institute of Aerodynamics, Wüllnerstrasse 5a, Aachen 52062, Germany
[b]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, Jülich 52428, Germany

ARTICLE INFO†

Keywords:
Multiphase;
Lagrangian Particle;
Artificial Neural Network

ABSTRACT

A novel approach for the generation of drag correlations for multiphase flows is presented. Fully resolved computational fluid dynamics simulations for multiphase flows are performed to provide ground truth data. An artificial neural network is trained to learn the accurate particle behavior based on less accurate flow data from the Lagrangian particle simulation. For the case of a settling spherical particle, this approach outperforms the existing empirical model.

1. Introduction

Due to the computational intensity of fully resolved simulations, Lagrangian point particle models are widely spread for the simulation of particle laden flow with technical relevant numbers of particles. These models rely on empirical drag correlations for the determination of particle forces. It is well known that these models are only valid for $d_p \ll \Delta x$ [1]. The goal of this work is to use artificial neural networks (ANN) instead of empirical correlations to remedy this shortcoming.

2. Methods

2.1. Resolved particle simulations

For the fully resolved particle simulation, a Lattice Boltzmann method (LBM) is combined with a rigid body solver (Fig. 1a). A detailed description of the LBM implementation can be found in [2]. The base grid has a grid size of Δx_α with additional adaptive mesh refinement around the moving particle, providing a local resolution of Δx_β. The forces acting on the particle surface $\vec{F}$ are determined by integrating the hydrodynamic forces over the particle surface using the momentum exchange method for LBM. This information is used to solve the motion equation of each particle, which is defined in a Lagrangian frame of reference

$$\frac{d^2\vec{x}_p}{dt^2} = \frac{\vec{F}_p}{m_p} + \vec{g}\left(1 - \frac{\rho_f}{\rho_b}\right) \tag{1}$$

by using a predictor-corrector scheme. At the boundary Γ_b (see Fig. 1a), the no-slip condition is enforced with $\vec{u}_\Gamma = \vec{u}_p$ where $\vec{u}_p$ refers to the particle velocity.

2.2. Lagrangian particle tracking

For the Lagrangian particle tracking simulations, the rigid body solver is replaced by a point particle approach. The particle motion is described by the Maxey-Riley equation [1]. If the added mass and history force are neglected, the motion eq. (1) is solved. Here, the momentum equation for the carrier fluid is solved on a uniformly refined grid with Δx_α. The force $\vec{F}_p$ is provided by empiric correlations C_D, such that

$$\frac{\vec{F}_p}{m_p} = \frac{C_D}{24}\frac{Re_p}{\tau_p}, \quad Re_p = \frac{\rho\|\vec{u} - \vec{u}_p\|d_p}{\mu}, \quad \tau_p = \frac{\rho_p d_p^2}{18\mu},$$

with the particle REYNOLDS number Re_p, the particle relaxation time τ_p the carrier fluid viscosity μ and particle density ρ_p. Here, the classical Schiller-Naumann equation for the drag part [1] is used, which reads as

$$C_D = \frac{24}{Re_p}(1 + \frac{1}{6}Re_p^{2/3}). \tag{2}$$

A two-way coupled approach is chosen to capture the particle motion feedback onto the flow field. The momentum equation for the carrier fluid phase is extended by the momentum source term associated with the particle reaction force, which reads as

$$\vec{F}_{f,p} = -\mathcal{G}(\|\vec{x} - \vec{x}_p\|)\vec{F}_p, \tag{3}$$

†This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02454 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
j.vorspohl@aia.rwth-aachen.de (J. Vorspohl);
l.andre@aia.rwth-aachen.de (L. André); (M. Rüttgers); (W. Schröder)
ORCID(s): 0000-0001-5564-723X (J. Vorspohl); 0000-0001-8430-0155 (L. André); 0000-0003-3917-8407 (M. Rüttgers); 0000-0002-3472-1813 (W. Schröder)

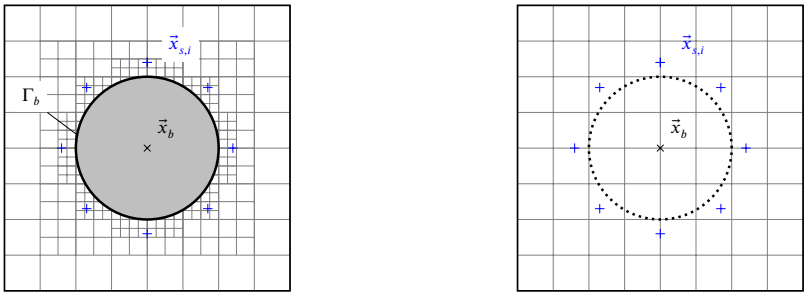


Figure 1: Grid setup for fully resolved simulations (a) and Lagrange particle simulations (b). Body-relative sampling locations $\vec{x}_{s,i}$ are shown as +.

where $\mathcal{G}(\cdot)$ is a Gaussian kernel, which distributes the momentum source to the vicinity of the particle. The standard deviation is chosen as $6\sigma = 1.6d_p$. By doing so, the intrinsic drawback of the Lagrange particle tracking method becomes apparent: The velocity field is locally disturbed by the two-way coupled approach, resulting in an erroneous evaluation of the given empirical correlation defined for the undisturbed carrier fluid velocity.

2.3. ANN based correlations from fully resolved simulations

The first step is to provide an improved drag correlation by using data from fully resolved simulations. For each time step n, the hydrodynamic force $\vec{F}^n$ acting on a given particle is known. Additionally, the velocity field at N_s points $\vec{x}^n_{s,i}$ relative to the body position $\vec{x}^n_b$ is extracted (see Fig. 1a). This information forms a dataset

$$\{\vec{u}^n_{rel,i}, \vec{F}^n\} \quad \text{with} \quad \vec{u}_{rel,i} = \vec{u}_{s,i} - \vec{u}_b, \tag{4}$$

which serves as training data for an artificial neural network (ANN) to find a suitable relation in the form of $\vec{F} \approx f(\vec{u}_{s,i})$.

The ANN is a multilayer perceptron composed of an input layer with N_s neurons, three fully connected layers with 64 neurons each, and a final layer with 3 neurons, one for each force component. The relatively small network architecture is chosen to guarantee an efficient performance for parallel CFD computations on high-performance computing (HPC) systems. A larger network architecture would negatively affect the efficiency of the proposed method, since the ANN must be employed by each rank for the number of particles in the corresponding rank for every time step. The weights and biases are updated by an adaptive moments (ADAM) optimizer [3] and a mean-squared error (MSE) loss function. A leaky-Rectified linear unit (ReLU) activation function is chosen [4], a variation of the ReLU activation function [5].

The obtained ANN is used as a novel force correlation for the Lagrange particle tracking. Different to the known correlations such as eq. (2), the input data accounts for the velocity disturbance caused by the boundary condition in the fully resolved simulation. To enable the use of the obtained ANN inside the LPT simulation, the analogous velocity data at the points $x_{s,i}$ has to be obtained (see Fig. 1b).

While this approach makes use of the most accurate data, systematic deviations of the input data (i.e., the velocity field) have to be expected in Lagrangian particle simulations. This is due to the lower grid resolution and the fact, that the momentum feedback compared to the fully resolved simulation is fundamentally different.

2.4. ANN based correlations from coupled simulations

To address these drawbacks, a novel approach is proposed. Here, both the fully resolved simulation and the LPT simulation are run simultaneously. For each time step, the particle trajectory from both simulation is coupled by setting

$$\vec{u}_{p,LPT} = \vec{u}_{p,resolved}. \tag{5}$$

From this coupled simulation, a dataset similar to eq. (4) is extracted by sampling the carrier fluid velocity from the LPT. The resulting ANN can be used in LPT simulations, as described in Sec. 2.3.

3. Results

The generic case of spherical particle settling in an initially quiescent fluid is considered. The REYNOLDS number with respect to terminal velocity is chosen as $Re = 32$. Both

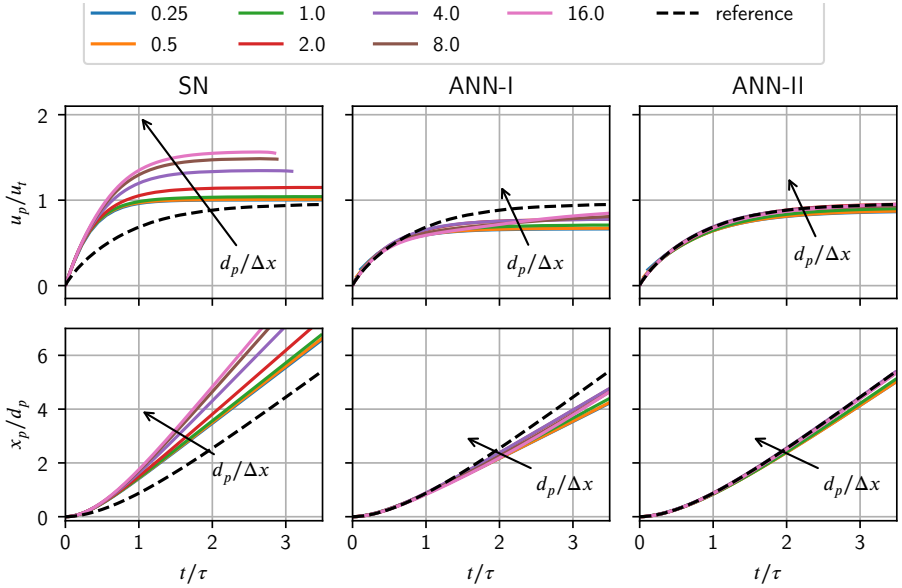


Figure 2: Particle velocity u_p/u_t and position x_p for Schiller-Naumann (SN), ANN from fully resolved simulations (ANN-I) and ANN from coupled simulations (ANN-II), with u_t as theoretical terminal velocity.

new models are only trained with data for $d_p/\Delta x = 4$. The results for all models and resolutions in the range $d_p/\Delta x = [0.25, 16.0]$ are shown in Fig. 2.

As expected, the SN model yields significant deviations from the reference values for large $d_p/\Delta x$. For smaller $d_p/\Delta x$, the solution approaches the one-way coupled case with $u_p/u_t = 1$. The variation of the result over $d_p/\Delta x$ is greatly reduced for both ANN based models. Additionally, the transient behavior is improved. Overall, the ANN-II model shows the most promising results with respect to the fully resolved reference solution.

4. Acknowledgment

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the Research Unit FOR 5595 Archimedes (Project number 510921053) and within the framework of the SFB/Transregio 129 'Oxyflame' (subproject B2). The support is gratefully acknowledged. Computing resources were provided by the High Performance Computing Centre Stuttgart (HLRS) and by the Jülich Supercomputing Centre (JSC) within a Large-Scale Project of the Gauss Centre for Supercomputing (GCS).

References

[1] F. Evrard, F. Denner, B. van Wachem, Euler-Lagrange modelling of dilute particle-laden flows with arbitrary particle-size to mesh-spacing ratio, Journal of Computational Physics: X 8 (2020) 100078. doi:10.1016/j.jcpx.2020.100078.

[2] A. Lintermann, W. Schröder, Lattice–Boltzmann simulations for complex geometries on high-performance computers, CEAS Aeronautical Journal 11 (2020) 745 – 766. doi:10.1007/s13272-020-00450-1.

[3] D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, International Conference on Learning Representations (12 2014). doi:10.48550/arXiv.1412.6980.

[4] A. L. Maas, Rectifier Nonlinearities Improve Neural Network Acoustic Models, International Conference on Machine Learning 30 (1) (2013) 3.
URL https://api.semanticscholar.org/CorpusID:16489696

[5] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Communications of the Association for Computing Machinery 60 (6) (2017) 84–90. doi:10.1145/3065386.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

HydroGym-GPU: From 2D to 3D Benchmark Environments for Reinforcement Learning in Fluid Flows

Christian Lagemann[a,*], Mario Rüttgers[b], Miro Gondrum[c], Matthias Meinke[c], Wolfgang Schröder[c], Andreas Lintermann[b] and Steven L. Brunton[a]

[a] University of Washington, Department of Mechanical Engineering, Stevens Way NE, Box 352600, 98195, Seattle, WA, USA
[b] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany
[c] Institute of Aerodynamics and Chair of Fluid Mechanics, RWTH Aachen University, Wüllnerstr. 5a, 52062 Aachen, Germany

ARTICLE INFO†

Keywords:
Reinforcement Learning;
Flow Control;
Graphics Processing Units;
Lattice Boltzmann Method

ABSTRACT

Fluid flow modeling and control is a significant modern challenge with potential impacts across science, technology, and industry. Improved flow control could enhance drag reduction, mixing, and noise reduction in areas like transportation, energy, and medicine. However, progress in flow control is currently hindered by the lack of systematically standardized benchmarks and the high computational cost of fluid simulations. While two-dimensional problems have been extensively studied, three-dimensional simulations with larger meshes are rarely considered due to the need for highly parallelized and specialized solvers. As a result, the engineering burden of encapsulating these simulations in benchmark environments has proven to be a significant barrier. In this paper, a GPU-based extension of the HydroGym platform coupling the multiphysics solver framework m-AIA with a state-of-the-art reinforcement learning platform is presented for fluid flow control problems. Based on the highly-parallelized lattice Boltzmann solver, which is part of m-AIA, a new set of three-dimensional, non-differentiable fluid flow environments is added that extend existing flow control challenges to a new level of physical and computational complexity.

1. Introduction

The effective control of fluid dynamics is a critical challenge in many scientific, technological, and industrial systems and improved flow control has the potential to dramatically enhance performance in domains as diverse as energy, transportation, security, and medicine. For example, turbulent wall-bounded fluid flows are of significant importance for numerous engineering [1, 2, 3, 4, 5] and biomedical applications [6, 7, 8, 9, 10], e.g., in the context of reducing the CO_2 emissions in the transportation sector or enhancing disease prevention and monitoring in human medicine. Moreover, understanding and controlling the dynamics of mixing processes and multi-phase flows, such as microscopic fibers or gas bubbles in turbulent flows, is crucial for various environmental and industrial applications, including pollution control [11, 12], marine biology [13, 14], and chemical engineering [15, 16, 17]. However, controlling fluid flow is notoriously difficult due to the nonlinear and multiscale nature of fluid dynamics, which leads to high-dimensional and non-convex control problems, but rapid advancements in machine learning have significantly improved our ability to tackle these complex optimization challenges [18]. For example, reinforcement learning has recently achieved notable successes in various modern tasks, including decision-making in planning [19], robotics [20], and protein design [21]. A major factor driving these advances is the development of scalable reinforcement learning frameworks and standardized environments, which facilitate direct comparisons between learned policies.

In contrast, progress in reinforcement learning for flow control has been limited by the scarcity of such platforms. To overcome this fundamental limitation, a new scalable and extensible platform called HydroGym was recently developed that closes the loop between efficient flow solvers, flow control benchmark problems, and state-of-the-art reinforcement learning. However, rooting in a finite-element based PDE solver called Firedrake, the existing HydroGym environments are limited to a variety of two-dimensional benchmark flows since running and validating advanced three-dimensional simulations is not feasible due to scalability issues of the Firedrake solver.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02455 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
clage@uw.edu (C. Lagemann); m.ruettgers@fz-juelich.de (M. Rüttgers); m.gondrum@aia.rwth-aachen.de (M. Gondrum); m.meinke@aia.rwth-aachen.de (M. Meinke); office@aia.rwth-aachen.de (W. Schröder); a.lintermann@fz-juelich.de (A. Lintermann); sbrunton@uw.edu (S.L. Brunton)
ORCID(s): 0000-0003-1150-4987 (C. Lagemann); 0000-0003-3917-8407 (M. Rüttgers); 0000-0002-2796-6644 (M. Gondrum); 0000-0003-4812-8495 (M. Meinke); 0000-0002-3472-1813 (W. Schröder); 0000-0003-3321-6599 (A. Lintermann); 0000-0002-6565-5118 (S.L. Brunton)

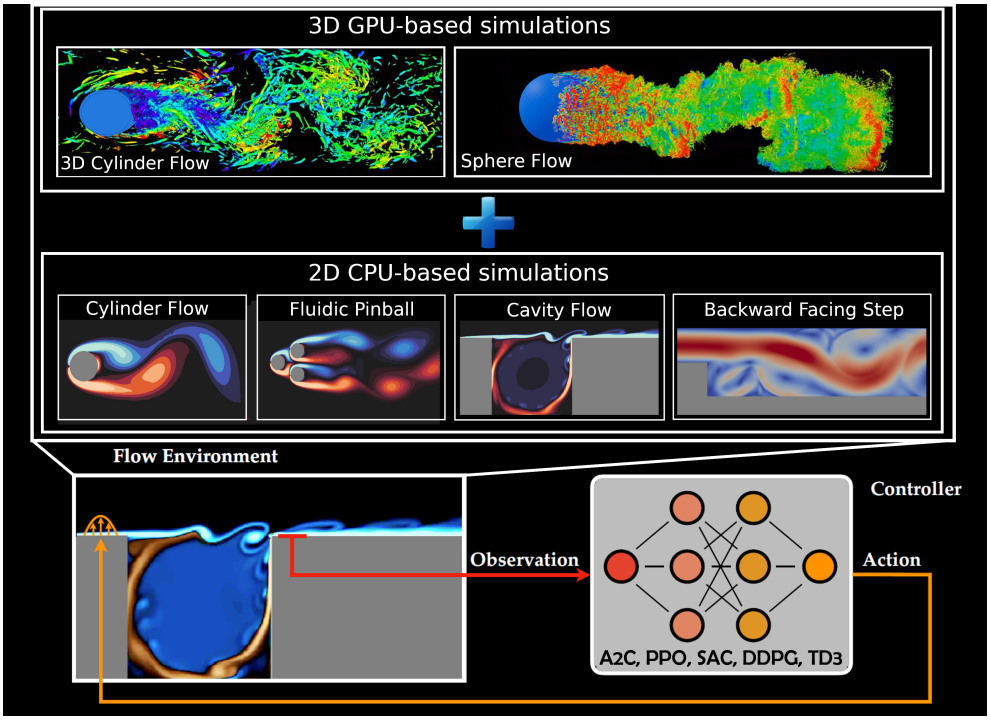


Figure 1: Flow configurations and benchmark algorithms included in the initial and extended HydroGym-GPU platform.

To tackle this limitation, a new GPU-accelerated extension bridging the lattice Boltzmann (LB) solver implemented in the multiphysics solver framework m-AIA[1] (formerly known as Zonal Flow Solver - ZFS [22, 23]) with the extensible HydroGym platform is introduced. Based on previous, successful RL applications [24] using this highly parallelized LB solver, a novel collection of three-dimensional, non-differentiable fluid flow environments has been incorporated. These additions elevate the existing flow control challenges to an unprecedented level of physical and computational complexity.

2. HydroGym-GPU

Recent advances have combined reinforcement learning with flow control, focusing on three main areas: controlling individual environments, developing strategies to navigate through flow environments, and using multi-agent reinforcement learning to learn components of numerical solvers. Despite these impressive results, the studies have been limited to specific flow control environments and lack the diversity found in modern machine learning frameworks. Hence, to effectively apply modern reinforcement learning to a broader class of fluid flow control problems, it appears to be beneficial to train policies across multiple environments. As a result, this approach allows for fine-tuning existing agents for future applications and significantly reduces computational efforts, especially for three-dimensional simulations. Therefore, integrating various flow control environments with different computational complexities requires scaling both the environments and the reinforcement learning agents to optimally use available computational resources.

One solution to tackle these problems is the usage of highly efficient and parallelized CFD solvers combined with modern data exchange protocols suitable for large-scale HPC application to enable efficient communication between RL agents and environments during runtime. Therefore,

[1] *multiphysics - Aerodynamisches Institut Aachen*

the present work leverages the existing LB solver of the m-AIA framework offering compute efficient and largely scalable simulations. It benefits from a hybrid parallelization approach based on *MPI* and a shared memory model either based on *OpenMP* or on the parallel algorithms defined in the C++ standard 17 (PSTL) implemented in the *NVIDIA HPC SDK* allowing for i) a hardware-agnostic implementation on CPU and GPU and ii) favorable strong and weak scaling on modern HPC architectures. The LB solver operates on hierarchical unstructured Cartesian grids, which are generated using a massively parallel grid generator [25] being part of m-AIA. The discretized form of the *Boltzmann* equation is solved with the *Bhatnagar-Gross-Krook* (BGK) approximation of the right-hand side collision process [26], i.e.,

$$f_i(\boldsymbol{x}+\boldsymbol{\xi_i}\delta t, t+\delta t) - f_i(\boldsymbol{x},t) = -\omega(f_i(\boldsymbol{x},t) - f_i^{eq}(\boldsymbol{x},t)), \quad (1)$$

is solved for the particle probability distribution functions f_i (PPDFs) at neighboring fluid cells at locations $\boldsymbol{x}+\boldsymbol{\xi_i}\delta t$. They are functions of the location vector $\boldsymbol{x} = (x_1, x_2, x_3)^T$, the discrete molecular velocity vector $\boldsymbol{\xi_i} = (\xi_{i1}, \xi_{i2}, \xi_{i3})^T$, and the time and time increment t and δt. The collision frequency is expressed by ω.

To enable highly efficient communication between our LB solver and the RL agents, we extended m-AIA's MPI interface leveraging the Multiple Program Multiple Data (MPMD) mode. The MPMD interface has the great advantage of executing different programs across multiple processors and heterogeneous compute hardware, e.g., CPU-CPU, CPU-GPU, GPU-GPU setups while facilitating complex, distributed computations. Moreover, MPMD supports the use of different programming languages and tools within the same application. This interoperability is extremely beneficial for the present HydroGym benchmark platform as it facilitates efficient communication between different m-AIA flow environments and standard, Python-based RL libraries. Moreover, it allows for frequent data exchange and coordination between different computational tasks with barely any computational overhead enabling very efficient training runs. Finally, it also enables multi-agent/multi-environments training protocols in a straightforward fashion without requiring changes in the code. To this end, all communications between environments and agents as wells as inter-environment communication (if necessary) is handled by our MPMD interface, while relevant inter-agent communication, e.g., gradient and weight sharing, is realized using existing deep learning libraries (JAX, PyTorch, TensorFlow, etc.).

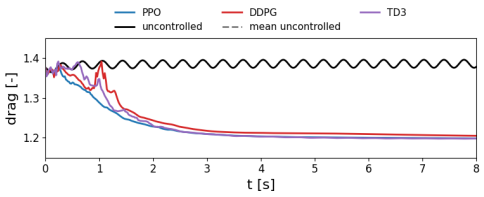
Leveraging this computational efficiency of the GPU-accelerated LB solver and the MPMD communication, three-dimensional fluid flow environments with grid sizes on the order of $100 \cdot 10^6$ cells are added to the HydroGym platform, paving the way for novel transfer learning and control applications in the largely unexplored chapter of three-dimensional flow control. Precisely, two distinct drag reduction scenarios which are typically considered as benchmark problems in CFD code development are implemented:

3D Cylinder Flow: The first test case of our HydroGym-GPU extension targets the flow around a smooth circular cylinder which is characterized by a large range of interesting fluid mechanics phenomena as the REYNOLDS number (Re) is increased from a low to high Re-number regime, e.g., $100 < Re < 10^5$. In more detail, the flow develops from a two-dimensional steady wake to three-dimensional unsteady vortex shedding, followed by wake transition, shear layer instability and boundary layer transition. The present flow simulation setup is validated to be accurate for $Re <$ 4,000. To facilitate interactions between the RL agent and the flow environment, multiple mass sources are equally distributed across the circumference of the cylinder, each being independently controllable. Note that the mass sources extend over the entire spanwise direction. The state space, which the RL agent observes, can include a mixture of an arbitrary number of probes, including velocity, vorticity, pressure, and density sensors, that can be distributed in the entire computational domain. The reward metric is calculated based on the weighted total force value in all three dimensions similar to two-dimensional counterparts.

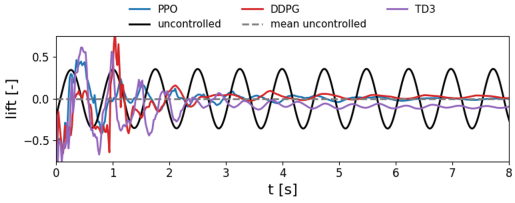
3D Sphere Flow: The second test case consists of an unsteady flow past a sphere. In the subcritical regime, e.g. $800 < Re < 3.7 \cdot 10^5$, the dynamics show a variety of complex flow patterns including a thin laminar boundary layer, flow separation at a location that is not known a priori, transition to turbulence in thin shear layers, and an unsteady recirculation zone followed by a turbulent wake. Without faithful representation of these flow features, it is not possible to accurately predict aerodynamic or hydrodynamic loading on complex-geometry objects. This is important since a drag reduction setup similar to the cylinder flow case is considered here. In the current implementation, the flow simulations are validated for $Re < 10^4$. To enable interactions between the RL agent and the flow environment, up to 32 individually controllable point mass sources are distributed across the sphere surface. Similar to the cylinder test case, the observational state space can comprise an arbitrary number of velocity, vorticity, pressure, and density sensors. Again, the reward is based on the integral force value in all three dimensions.

3. Results

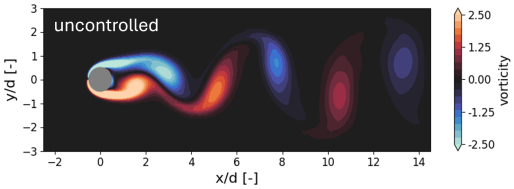
In the following, we briefly present validation results for the introduced 3D flow environments and compare them to



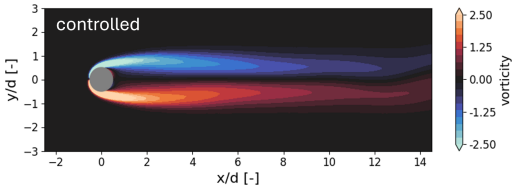
(a) Drag distribution over time for a single test episode.



(b) Lift distribution over time for a single test episode.



(c) Instantaneous vorticity snapshot of an uncontrolled flow environment.



(d) Instantaneous vorticity snapshot of a flow environment controlled by a PPO agent.

Figure 2: Exemplary test results of on- (PPO) and off-policy (DDPG, TD3) agents for the 2D cylinder flow test case at $Re = 100$.

literature. Afterwards, we shortly discuss training results for different agents interacting with a 2D cylinder flow at $Re = 100$ and outline how successfully learned control policies can be leveraged in transfer learning experiments to reduce computational costs in complementing 3D environments.

Validation: A grid refinement study has been conducted for 3D simulations with $Re = 200$ by comparing the drag coefficient $C_d = (2 \cdot F_x)/(\rho \cdot U \cdot A)$ to the results in [27], with the temporally averaged force in the streamwise direction acting on the cylinder F_x, the density ρ, the inlet velocity U, and the cross-sectional area of the cylinder A. The grids are locally refined with 2 cubic refinement patches that capture the wake region and 3 cylindrical refinement patches for the near-wall regions. The results are shown in Tab. 1. Since an additional gain in accuracy of only 0.1% for the simulation with the fine grid compared to the medium grid an increased grid size from $40 \cdot 10^6$ to $80 \cdot 10^6$ cells cannot be justified and the medium grid is used for training the RL agents.

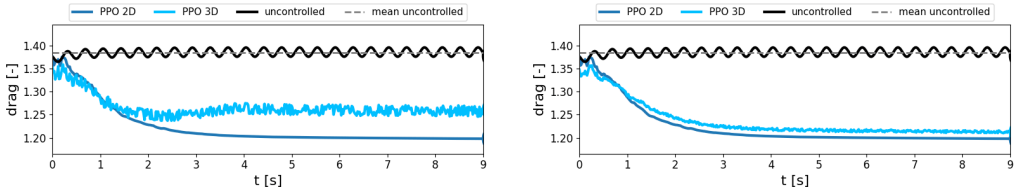
Training results: To investigate the effectiveness of our GPU-enhanced benchmark platform, we performed multiple tests for a variety of RL agents proposed in literature and conducted extensive hyperparameter optimizations and transfer learning experiments across different flow environments. However, for sake of brevity, here we only elaborate on the selected test case of a flow around cylinder at a $Re = 100$. Extensive results for all other test cases will be discussed in follow-up work.

Grid resolution	No. cells	C_d [dev. to $C_d = 1.338$ [27]]
Coarse	$20 \cdot 10^6$	1.180 [−11.8%]
Medium	$40 \cdot 10^6$	1.317 [−1.6%]
Fine	$80 \cdot 10^6$	1.318 [−1.5%]

Table 1
Grid refinement study for 3D simulations at $Re = 200$.

2D flow environment: Figure 2 exhibits exemplary test results for different RL agents including one on-policy (PPO) and two off-policy methods (DDPG and TD3). As illustrated in Fig. 2d, all agents learn a competitive control policy that stabilizes the wake and mostly reduce the lift fluctuations caused by the vortex shedding (see Fig. 2b). Overall, a total drag reduction of approximately 12% can be achieved across all agents (see Fig. 2a) which is in line with previous studies reported in literature. More importantly, by leveraging the highly efficient m-AIA LB solver in combination with the developed MPMD interface, our HydroGym-GPU extension requires only approximately 40 minutes on a single NVIDIA A100 GPU for a full training cycle (400 episodes, each containing 20 vortex shedding periods), marking a new milestone for comparable flow environments in terms of efficiency.

Transfer learning to 3D flow environments: In the following, we highlight another interesting aspect about the



(a) zero-shot transfer learning (no training)

(b) fine-tuned control policy (20 episodes)

Figure 3: Transfer learning experiments from 2D to 3D environments leveraging (a) zero-shot applications without further training in the target environment and (b) fine-tuning experiments with limited adaptation (20 training episodes) in the target environment.

HygroGym-GPU platform enabling transfer learning experiments in a straightforward fashion. Here, we demonstrate this feature in two experiments - first, we directly transfer a control policy learned in the 2D flow case to the corresponding 3D environment, e.g. a zero-shot transfer with no subsequent training/adaptation, and second, we fine-tune the 2D flow control policy for 20 episodes in the 3D environment. Results are shown in Fig. 3 and exhibit promising trends for future compute intensive 3D flow environments. That is, even in the zero-shot transfer, the agent can mostly suppress the vortex shedding in the wake and achieves a drag reduction of approximately 8 % (see Fig. 3a). Considering that the agent has never explored the 3D environment before and consequently could never adapted to it, these results indicate a good performance for future, more general control agents leveraging foundation models across multiple flows simultaneously.

Furthermore, we can improve the performance of the control policy in the 3D environment using a fine-tuning training step in the new test environment. Precisely, we now allow the agent to adapt its network parameters to the new environment in a limited training sequence (max. 20 episodes for the 3D test case). Results are shown in Fig. 3b. The fine-tuned control agent can achieve a similar drag reduction performance compared to the original 2D test case. As a result, this hybrid transfer learning strategy requires only a fraction of the computational costs of training procedures exploring exclusively 3D environments, but still learns competitive control policies. Hence, pre-training in a simpler 2D environment with low or moderate compute requirements followed by a fine-tuning sequence in more challenging 3D environments massively reduces the training time and costs. This finding is particularly promising for future HydroGym-GPU extensions which will investigate various complex and compute intensive 3D flow cases exhibiting turbulence dominated flow features.

4. Conclusion

In this contribution, a novel GPU-accelerated extension of the recently introduced HydroGym platform is presented, connecting the multi-physics solver framework m-AIA with this adaptable RL benchmark. By utilizing a highly parallelized LB solver, a representative set of three-dimensional, non-differentiable fluid flow scenarios has been integrated and validated. Furthermore, we outlined that hybrid transfer learning strategies, which pre-train an agent in a simple flow environment of moderate computational complexity and fine-tune the policy in the target environment, require only a fraction of the computational costs compared to training procedures exploring costly 3D environment, yet learn competitive control policies. In future, we will extend our highly efficient HydroGym-GPU platform and introduce more challenging and real-world oriented test cases such as aviation applications, noise reduction in aeroacoustics settings, and mixing enhancement in multi-phase flows. Additionally, future benchmarks will investigate model-based RL agents that use latent dynamical models to uncover physical mechanisms [28], as part of our quest for general foundation models in fluid flow control. These advancements will significantly elevate the complexity of existing flow control challenges, introducing new levels of physical and computational intricacy.

Acknowledgements

The research leading to these results partially has been conducted in the HANAMI project, which receives funding from the European Union Horizon Europe Programme - Grant Agreement Number 101136269 under the call HORIZON-EUROHPC-JU-2022-INCO-04. The authors gratefully acknowledge the computing time granted by the JARA Vergabegremium and provided on the JARA Partition part of the supercomputer JUWELS [29] at *Forschungszentrum Jülich*. Moreover, this research was partially funded by the German Research Foundation within the Walter Benjamin fellowship LA 5508/1-1 (CL) and CL and SLB

acknowledge support from the National Science Foundation AI Institute in Dynamic Systems (grant number 2112085) and the Boeing Company. Furthermore, the authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. for funding this project by providing computing time on the GCS Supercomputers (CL, MM, MG, WS).

References

[1] I. Marusic, R. Mathis, N. Hutchins, Predictive Model for Wall-Bounded Turbulent Flow, Science 329 (5988) (2010) 193–196. doi:10.1126/science.1188765.

[2] P. Ricco, M. Skote, M. A. Leschziner, A review of turbulent skin-friction drag reduction by near-wall transverse forcing, Progress in Aerospace Sciences 123 (2021) 100713. doi:10.1016/j.paerosci.2021.100713.

[3] E. Lagemann, M. Albers, C. Lagemann, W. Schröder, Impact of Reynolds Number on the Drag Reduction Mechanism of Spanwise Travelling Surface Waves, Flow, Turbulence and Combustion 113 (1) (2024) 27–40. doi:10.1007/s10494-023-00507-1.

[4] E. Mäteling, M. Albers, W. Schröder, How spanwise travelling transversal surface waves change the near-wall flow, Journal of Fluid Mechanics 957 (2023) A30. doi:10.1017/jfm.2023.54.

[5] E. Lagemann, S. L. Brunton, W. Schröder, C. Lagemann, Towards extending the aircraft flight envelope by mitigating transonic airfoil buffet, Nature Communications 15 (1) (2024) 5020. doi:10.1038/s41467-024-49361-3.

[6] J. Bellien, M. Iacob, V. Richard, J. Wils, V. Le Cam-Duchez, R. Joannidès, Evidence for wall shear stress-dependent t-PA release in human conduit arteries: role of endothelial factors and impact of high blood pressure, Hypertension Research 44 (3) (2021) 310–317. doi:10.1038/s41440-020-00554-5.

[7] G. Zhou, Y. Zhu, Y. Yin, M. Su, M. Li, Association of wall shear stress with intracranial aneurysm rupture: systematic review and meta-analysis, Scientific Reports 7 (1) (2017) 5331. doi:10.1038/s41598-017-05886-w.

[8] L. Adamo, O. Naveiras, P. L. Wenzel, S. McKinney-Freeman, P. J. Mack, J. Gracia-Sancho, A. Suchy-Dicey, M. Yoshimoto, M. W. Lensch, M. C. Yoder, G. García-Cardeña, G. Q. Daley, Biomechanical forces promote embryonic haematopoiesis, Nature 459 (7250) (2009) 1131–1135. doi:10.1038/nature08073.

[9] E. Tzima, M. Irani-Tehrani, W. B. Kiosses, E. Dejana, D. A. Schultz, B. Engelhardt, G. Cao, H. DeLisser, M. A. Schwartz, A mechanosensory complex that mediates the endothelial cell response to fluid shear stress, Nature 437 (7057) (2005) 426–431. doi:10.1038/nature03952.

[10] E. Lagemann, S. L. Brunton, C. Lagemann, Uncovering wall-shear stress dynamics from neural-network enhanced fluid flow measurements, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 480 (2292) (2024). doi:10.1098/rspa.2023.0798.

[11] J. González-Martín, N. J. R. Kraakman, C. Pérez, R. Lebrero, R. Muñoz, A state–of–the-art review on indoor air pollution and strategies for indoor air pollution control, Chemosphere 262 (2021) 128376. doi:10.1016/j.chemosphere.2020.128376.

[12] M. He, Z. Xu, D. Hou, B. Gao, X. Cao, Y. S. Ok, J. Rinklebe, N. S. Bolan, D. C. W. Tsang, Waste-derived biochar for water pollution control and sustainable development, Nature Reviews Earth & Environment 3 (7) (2022) 444–460. doi:10.1038/s43017-022-00306-8.

[13] C. D. O'Dowd, M. C. Facchini, F. Cavalli, D. Ceburnis, M. Mircea, S. Decesari, S. Fuzzi, Y. J. Yoon, J.-P. Putaud, Biogenically driven organic contribution to marine aerosol, Nature 431 (7009) (2004) 676–680. doi:10.1038/nature02959.

[14] A. Paytan, K. R. M. Mackey, Y. Chen, I. D. Lima, S. C. Doney, N. Mahowald, R. Labiosa, A. F. Post, Toxicity of atmospheric aerosols on marine phytoplankton, Proceedings of the National Academy of Sciences 106 (12) (2009) 4601–4605. doi:10.1073/pnas.0811486106.

[15] V. Hessel, H. Löwe, F. Schönfeld, Micromixers—a review on passive and active mixing principles, Chemical Engineering Science 60 (8-9) (2005) 2479–2501. doi:10.1016/j.ces.2004.11.033.

[16] V. Giurgiu, L. Beckedorff, G. C. Caridi, C. Lagemann, A. Soldati, Machine learning-enhanced PIV for analyzing microfiber-wall turbulence interactions, International Journal of Multiphase Flow 181 (2024) 105021. doi:10.1016/j.ijmultiphaseflow.2024.105021.

[17] E. Lagemann, J. Roeb, S. L. Brunton, C. Lagemann, A deep learning approach to wall-shear stress quantification: From numerical training to zero-shot experimental application (2024). arXiv:2409.03933.

[18] S. L. Brunton, J. N. Kutz, Data-Driven Science and Engineering, 2nd Edition, Cambridge University Press, 2022. doi:10.1017/9781009089517.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489. doi:10.1038/nature16961.

[20] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn, S. Levine, Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills (2021). arXiv:2104.07749.

[21] I. D. Lutz, S. Wang, C. Norn, A. Courbet, A. J. Borst, Y. T. Zhao, A. Dosey, L. Cao, J. Xu, E. M. Leaf, C. Treichel, P. Litvicov, Z. Li, A. D. Goodson, P. Rivera-Sánchez, A.-M. Bratovianu, M. Baek, N. P. King, H. Ruohola-Baker, D. Baker, Top-down design of protein architectures with reinforcement learning, Science 380 (6642) (2023) 266–273. doi:10.1126/science.adf6591.

[22] A. Lintermann, M. Meinke, W. Schröder, Zonal Flow Solver (ZFS): a highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34 (7-8) (2020) 458–485. doi:10.1080/10618562.2020.1742328.

[23] Insitute of Aerodynamics and Chair of Fluid Mechanics, RWTH Aachen University, multiphysics - Aerodynamisches Institut Aachen (2024). doi:10.5281/zenodo.13350585.

[24] M. Rüttgers, M. Waldmann, K. Vogt, J. Ilgner, W. Schröder, A. Lintermann, Automated surgery planning for an obstructed nose by combining computational fluid dynamics with reinforcement learning, Computers in Biology and Medicine 173 (2024) 108383. doi:10.1016/j.compbiomed.2024.108383.

[25] A. Lintermann, S. Schlimpert, J. Grimmen, C. Günther, M. Meinke, W. Schröder, Massively parallel grid generation on HPC systems, Computer Methods in Applied Mechanics and Engineering 277 (2014) 131–153. doi:10.1016/j.cma.2014.04.009.

[26] X. He, L.-S. Luo, Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, Physical Review E 56 (6) (1997) 6811–6817. doi:10.1103/PhysRevE.56.6811.

[27] B. Rajani, A. Kandasamy, S. Majumdar, Numerical simulation of laminar flow past a circular cylinder, Applied Mathematical Modelling 33 (3) (2009) 1228–1247. doi:10.1016/j.apm.2008.01.017.

[28] K. Lagemann, C. Lagemann, S. Mukherjee, Invariance-based learning of latent dynamics, in: The Twelfth International Conference on Learning Representations, 2024.
URL https://openreview.net/forum?id=EWTFMkTdkT

[29] D. Alvarez, JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre, Journal of large-scale research facilities JLSRF 7 (2021) A183. doi:10.17815/jlsrf-7-183.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Super-Resolution and Parallel-in-Time Integration to Accelerate Simulations With the ICON-O Ocean Model

Philip Freese[a,*], Maximilian Witte[b], Fabricio R. Lapolli[c], Sebastian Götschel[a], Peter Korn[c], Christopher Kadow[b] and Daniel Ruprecht[a]

[a]Hamburg University of Technology, Chair Computational Mathematics, Institute of Mathematics, 21073, Hamburg, Germany
[b]German Climate Computing Centre (DKRZ), Division data analysis, 20146, Hamburg, Germany
[c]Max-Planck Institute for Meteorology, Climate Variability, 20146, Hamburg, Germany

ARTICLE INFO†

Keywords:
Super-Resolution;
Machine Learning;
Parallel-in-Time Integration;
Ocean Modeling

ABSTRACT

In this talk, we present recent results on improving the throughput of ocean simulations of the ICON-O ocean model. Our goal is to achieve more simulated days per computing day without sacrificing accuracy. To achieve this, we propose the use of machine learning (ML) combined with a small-scale parallel-in-time time stepping scheme. We show that the ML correction makes it possible to compute on coarser grids and be at least as accurate as on a grid twice as fine. In addition, the spectral deferred correction (SDC) time integration allows for higher order and larger time steps. Moreover, parallel SDC makes even better use of the available resources. Thus, the combination of both approaches leads to an acceleration of the simulation.

1. Introduction

Ocean models are a key component of any Earth system model. As many models are based on partial differential equations, their complexity requires the use of state-of-the-art high-performance computing systems for simulations. However, global simulations over multiple decades with resolutions below one kilometer capable of representing submesoscale eddies on the numerical grid, are still not feasible due to excessive runtimes. This talk will explore two strategies to speed up simulations of ICON-O: super-resolution to reduce the required spatial resolution and parallel-in-time integration to accelerate numerical time stepping.

2. The ICON-O ocean model

The Icosahedral Nonhydrostatic Weather and Climate Model (ICON) is the Earth system modeling framework of the Max Planck Institute for Meteorology (MPI-M), the German Weather Service (DWD), the German Climate Computing Center (DKRZ), and the Karlsruhe Institute of Technology (KIT). ICON-O is the ocean component of ICON and is based on the hydrostatic Boussinesq equations. It uses a special form of structure-preserving finite elements on an Arakawa C-grid as spatial discretization [1, 2] and a semi-implicit Adams–Bashforth-2 (AB) method for numerical time stepping.

3. Super-resolution

The incorporation of small scale features in simulations of partial differential equations to capture the effective or macroscopic behavior of the solution is a challenging task. We propose to use a machine learning motivated super-resolution approach trained with fine scale data to enhance coarse scale simulations during runtime. A deep neural network Θ is trained using pairs of high-resolution/low-resolution simulation snapshots. The network is then used to periodically correct a low-resolution solution u toward the restriction of a high-resolution simulation

$$\bar{u}(t + \Delta c) = \Theta(u(t + \Delta c)). \tag{1}$$

This correction is subsequently used as the initial value in the next time step. In our approach, the correction step size Δc is chosen much larger than the time step size Δt of the underlying time stepping scheme, i.e., $\Delta c \gg \Delta t$. A schematic overview of the application of the model during runtime is shown in Fig. 1.

We show that this allows us to achieve discretization errors much lower than what would be possible with an

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02456 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
philip.freese@tuhh.de (P. Freese); witte@dkrz.de (M. Witte); fabricio.lapolli@mpimet.mpg.de (F.R. Lapolli); sebastian.goetschel@tuhh.de (S. Götschel); peter.korn@mpimet.mpg.de (P. Korn); kadow@dkrz.de (C. Kadow); ruprecht@tuhh.de (D. Ruprecht)
ORCID(s): 0000-0002-9838-6321 (P. Freese); 0000-0002-2923-8907 (M. Witte); 0000-0003-2444-3861 (F.R. Lapolli); 0000-0003-0287-2120 (S. Götschel); 0000-0002-7525-5732 (P. Korn); 0000-0001-6537-3690 (C. Kadow); 0000-0003-1904-2473 (D. Ruprecht)

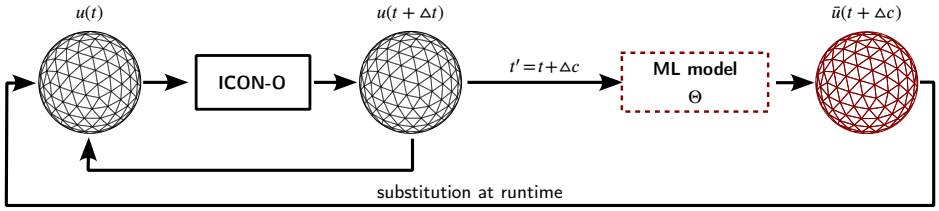


Figure 1: Hybrid approach combining numerical simulation using the ICON-O model with machine-learning-based super-resolution. The data flow during runtime is indicated by the arrows. Δt is the time step used in the numerical simulation (ICON-O), while $\Delta c \gg \Delta t$ is the correction step size.

uncorrected coarse solution. Further details on the training process and network architecture are provided by Witte et al. [3]. The results are based on the shallow water equation (SWE) using the so-called Galewsky [4] test case.

4. Parallel spectral deferred corrections

The ICON code is already highly scalable and exploits spatial parallelism, which provides generally good performance. However, time integration is sequential by nature and is therefore a major bottleneck in transient simulations, requiring special algorithms to accelerate this time integration. Here, we present parallel spectral deferred corrections (SDC) as a way to incorporate time parallelism into the ICON-O model. SDC introduced in 2000 by Dutt et al. [5], iteratively use a low order integrator to compute an approximate solution for an initial value problem

$$u'(t) = f_{\mathrm{E}}(u(t)) + f_{\mathrm{I}}(u(t)),\ u(t_n) = u_n \tag{2}$$

posed on an interval $[t_n, t_{n+1}]$ based on collocation. We assume that the right-hand side is split into a term f_{I} modeling fast, stiff but linear dynamics and a second term f_{E} that models slow and nonlinear processes. Let the M quadrature nodes of the collocation method applied to the interval $[t_n, t_{n+1}]$ be given as $t_n \leq \tau_1 < \cdots < \tau_M \leq t_{n+1}$. Using an implicit-explicit Euler as base method, one iteration (or sweep) of SDC for Eq. (2) computes approximate solutions u_m^k at the nodes $m = 1, \ldots, M$ as

$$\begin{aligned} u_m^{k+1} = u_n &+ \Delta t \sum_{j=1}^{M} q_{m,j} \left[f_{\mathrm{E}}(u_j^k) + f_{\mathrm{I}}(u_j^k) \right] \\ &+ \Delta t \sum_{j=1}^{m-1} \Delta\tau_{j+1} [f_{\mathrm{E}}(u_j^{k+1}) - f_{\mathrm{E}}(u_j^k)] \\ &+ \Delta t \sum_{j=1}^{m} \Delta\tau_j [f_{\mathrm{I}}(u_j^{k+1}) - f_{\mathrm{I}}(u_j^k)]. \end{aligned} \tag{3}$$

Here, $\Delta t = t_{n+1} - t_n$ is the step size, $q_{m,j}$ are the entries of the Butcher table of the collocation method and $\Delta\tau_j = \tau_j - \tau_{j-1}$, $j = 1, \ldots, M$ the distance between the nodes ($\Delta\tau_1 = \tau_1 - t_n$). The initial guess on each node u_m^0 is a copy of the solution u_n. Each iteration thereby increases the order of the scheme by one, up to the order of the underlying collocation method.

As noted in [6], the original definition of the $\Delta\tau_j$ can be replaced by algebraic parameters $q_{m,j}^{\Delta}$ to speed up the convergence. Moreover, it is even possible to choose $q_{m,j}^{\Delta} = 0$ if $m \neq j$, which allows parallelism across the method [7]. The latter choice is equivalent to replacing the classical lower-triangular SDC matrix with a diagonal one. Recently, Čaklović et al. [8] provided a novel set of (diagonal) parameters called MIN-SR-FLEX (minimization of the spectral radius of a given difference of the lower triangular matrix and its diagonal approximation). These enable the parallelism and still provide fast convergence and numerical stability. Freese et al. [9] provide parallel benchmarking of the resulting parallel SDC integrator for the (nonlinear) SWE using the ICON-O model.

5. Conclusions

In Fig. 2, we show results for the well-known Galewsky test case for the SWE up to day eight. We only depict the zonal velocity component, and note that the meridional and height components show similar behavior. We demonstrate that a simulation with a 20 km resolution corrected every $\Delta c = 12$ h achieves errors of an uncorrected simulation on a 10 km grid. As a reference, we use a simulation on a 2.5 km grid. In all cases, the time step Δt is chosen to be 10 s, i.e., $\Delta c \gg \Delta t$.

Figure 3 shows the speedup of parallel SDC on one node of the JUWELS cluster, utilizing up to 96 threads. To be precise, we plot the speedup of two SDC variants ($\Delta t = 960$ s) against the established AB ($\Delta t = 30$ s) with 2 OpenMP threads. All these methods result in a similar relative error of 1.3% with respect to an AB simulation with $\Delta t = 1$ s. For the established AB scheme, we observe speedup due to the intrinsic OpenMP space parallelization of ICON. Using the parallel SDC without actual OpenMP

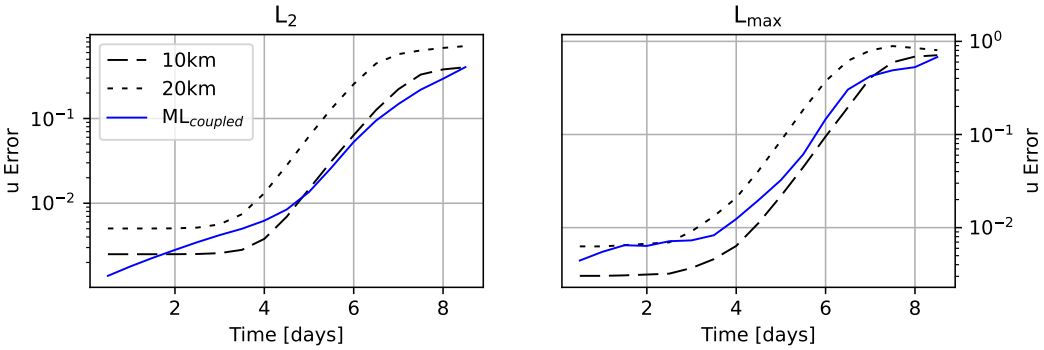


Figure 2: Spatial L_2 and $L_{\max}$ error over time in the zonal velocity component u for a 10 km, 20 km and ML-corrected 20 km resolution for the Galewsky test case. The error is computed with respect to a reference on a 2.5 km grid [3].

parallelism in time, yields a speedup mainly due to the higher order of the scheme, which allows for a 32 times larger time step. Using the additional time parallelism gives even more speedup due to better utilization of available resources.

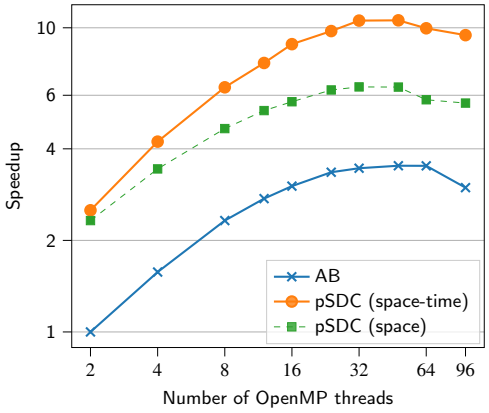


Figure 3: Speedup with respect to AB with 2 OpenMP threads for AB ($\Delta t = 30$ s) and parallel SDC (pSDC). We show a space-parallel only pSDC using ICON's intrinsic OpenMP parallelization and the novel space-time parallel SDC, both using $\Delta t = 960$ s. All methods yield a relative error of 1.3% for the Galewsky test case after 144 h [9].

References

[1] P. Korn, Formulation of an unstructured grid model for global ocean dynamics, Journal of Computational Physics 339 (2017) 525–552. doi:10.1016/j.jcp.2017.03.009.

[2] P. Korn, L. Linardakis, A conservative discretization of the shallow-water equations on triangular grids, Journal of Computational Physics 375 (2018) 871–900. doi:10.1016/j.jcp.2018.09.002.

[3] M. Witte, F. R. Lapolli, P. Freese, S. Götschel, D. Ruprecht, P. Korn, C. Kadow, Dynamic deep learning based super-resolution for the shallow water equations, Machine Learning: Science and Technology (2024). doi:10.1088/2632-2153/ada19f.

[4] J. Galewsky, R. K. Scott, L. M. Polvani, An initial-value problem for testing numerical models of the global shallow-water equations, Tellus A: Dynamic Meteorology and Oceanography 56 (5) (2004) 429–440. doi:10.3402/tellusa.v56i5.14436.

[5] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT. Numerical Mathematics 40 (2) (2000) 241–266. doi:10.1023/A:1022338906936.

[6] M. Weiser, Faster SDC convergence on non-equidistant grids by DIRK sweeps, BIT. Numerical Mathematics 55 (4) (2015) 1219–1241. doi:10.1007/s10543-014-0540-y.

[7] R. Speck, Parallelizing spectral deferred corrections across the method, Computing and Visualization in Science 19 (3-4) (2018) 75–83. doi:10.1007/s00791-018-0298-x.

[8] G. Čaklović, T. Lunet, S. Götschel, D. Ruprecht, Improving efficiency of parallel across the method spectral deferred corrections, SIAM Journal on Scientific Computing 47 (1) (2025). doi:10.1137/24M1649800.

[9] P. Freese, S. Götschel, T. Lunet, D. Ruprecht, M. Schreiber, Parallel performance of shared memory parallel spectral deferred corrections (2024). doi:10.48550/arXiv.2403.20135.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Deep Reinforcement Learning Strategies for Optimizing Flow Control in Wings

Ricard Montalà[a,*], Bernat Font[b], Pol Suárez[c], Jean Rabault[d], Oriol Lehmkuhl[e], Ricardo Vinuesa[c] and Ivette Rodriguez[a]

[a]Universitat Politècnica de Catalunya (UPC), Turbulence and Aerodynamics Research Group (TUAREG), ESEIAAT. Colom 11, 08222 Terrassa (Barcelona), Spain
[b]Delft University of Technology (TU Delft), Faculty of Mechanical Engineering, Mekelweg 2, 2628 CD Delft, Netherlands
[c]KTH Royal Institute of Technology, FLOW, Engineering Mechanics, SE-100 44 Stockholm, Sweden
[d]Independent Researcher, Oslo Norway
[e]Barcelona Supercomputing Center, Computer Applications in Science and Engineering (CASE), Plaza Eusebi Güell, 1-3, 08034 Barcelona, Spain

ARTICLE INFO[†]

Keywords:
Active Flow Control;
Separation Control;
Aerodynamics;
Machine Learning;
Deep Reinforcement Learning

ABSTRACT

A computational fluid dynamics (CFD) framework to combine deep reinforcement learning (DRL) with active flow control (AFC) simulations is presented. This framework is first validated in a canonical geometry, the flow past a cylinder at REYNOLDS number $Re = 100$, achieving a drag reduction of roughly 10%. Then, the methodology is applied to a more complex case in order to exploit the DRL capabilities to control massive separations of turbulent flows. To do this, the flow past the SD7003 wing at $Re = 60,000$ and angle of attack $AoA = 14°$ is analyzed, where the DRL control achieves an increase in the aerodynamic efficiency of approximately 157%.

1. Introduction

The application of active flow control (AFC) has already proven its capabilities in reducing flow separation in wings [1, 2]. However, these traditional AFC methods are inherently limited by their ability to target only a single frequency within the full spectrum of turbulence scales, which constrains the achievable level of drag reduction.

The development of new computing hardware combined with novel data-driven methods has made machine learning emerge as a powerful tool to tackle problems where physical knowledge is limited. In this manner, combining deep reinforcement learning (DRL) and AFC can yield promising results in the field of flow control, discovering more sophisticated actuation strategies that can produce more complex flow interactions and hence push the limits of AFC.

The first study that successfully applied DRL to AFC problems was Rabault et al. [3], who considered a 2D cylinder at a REYNOLDS number of $Re = 100$. Subsequent investigations extended the REYNOLDS numbers up to $Re = 2000$ [4] and the DRL technique was later implemented to a 3D cylinder with a similar methodology but at a maximum REYNOLDS number of $Re = 400$ [5]. Other studies also applied DRL-based controls to reduce the skin friction of wall-bounded flows at $Re_\tau = 180$ [6]; or to reduce the flow separation bubble in a boundary layer at $Re_\tau = 180$ [7]. Nevertheless, all these studies were applied in canonical cases (cylinders, channels, boundary layers…) and at low REYNOLDS numbers, showcasing the early stage of this methodology.

In the present study, an AFC-DRL framework is presented and validated with a three-dimensional cylinder at $Re = 100$. Later, as a step forward to test the capabilities of the methodology in more complex geometries with higher turbulent conditions, the SD7003 wing at $Re = 60,000$ and $AoA = 14°$ is studied. This will represent the application of DRL to more realistic industrial cases, hence pushing the boundaries of AFC for turbulent flows of industrial interest.

2. Methodology

In DRL, two main entities can be identified: (i) The environment and (ii) the agent. In the current framework, the environment is the CFD simulation that predicts how the flow evolves with a given actuation and the agent is a neural network (NN) that predicts the probability distribution of possible actions given the current state of the environment.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02457 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
ricard.montala@upc.edu (R. Montalà); b.font@tudelft.nl (B. Font); p.suarez@kth.se (P. Suárez); j.rabault@researcher.com (J. Rabault); o.lehmkuhl@bsc.es (O. Lehmkuhl); r.vinuesa@kth.se (R. Vinuesa); i.rodriguez@upc.edu (I. Rodriguez)
ORCID(s): 0009-0000-7911-7102 (R. Montalà); 0000-0002-2136-3068 (B. Font); 0000-0002-6031-5536 (P. Suárez); 0000-0002-7244-6592 (J. Rabault); 0000-0002-2670-1871 (O. Lehmkuhl); 0000-0001-6570-5499 (R. Vinuesa); 0000-0002-3749-277X (I. Rodriguez)

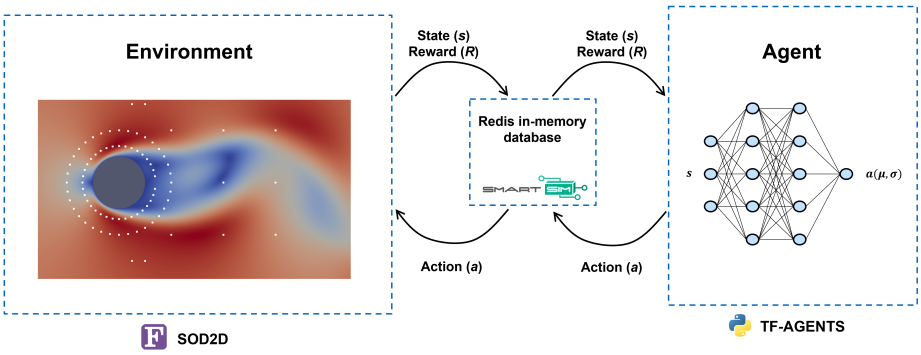


Figure 1: CFD - DRL setup.

For the simulations, the CFD solver SOD2D [8] is used, which is a spectral element method (SEM) in-house code developed at the Barcelona Supercomputing Center (BSC). In this study, the isothermal and incompressible Navier-Stokes equations are solved. The general form of the equations is given by

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} - \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} = 0, \tag{2}$$

where u and p denote the velocity and the pressure fields to be solved, respectively, and ν and ρ represent the kinematic viscosity and density of the fluid. For the flow past a cylinder case studied here, these equations are solved directly without filtering. For the flow past a wing, however, the filtered approach is used, i.e., large eddy simulation (LES). Therefore, the filtered velocity $\overline{u}$ and pressure $\overline{p}$ fields are solved instead, and an additional term is added to the right-hand side (RHS) of the momentum equation (Eq. 2) to account for the effects of unresolved turbulent scales, yielding

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial \overline{u_i u_j}}{\partial x_j} - \nu \frac{\partial^2 \overline{u_i}}{\partial x_j \partial x_j} + \frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} = -\frac{\partial \tau_{ij}}{\partial x_j}, \tag{3}$$

where τ_{ij} represents the subgrid-scale (SGS) tensor. In the present LES, the subgrid-scale (SGS) stress tensor is closed using a subgrid-scale viscosity model through the expression $\tau_{ij} - 1/3\tau_{kk}\delta_{ij} = -2\nu_{sgs}\overline{S}_{ij}$, where $\overline{S}_{ij}$ is the resolved strain rate tensor. The Vreman model [9] is employed to compute the subgrid-scale viscosity ν_{sgs}.

On the other hand, the Python library TF-Agents is used to create the DRL model and train it. The communication between the CFD solver (written in Fortran) and the DRL model (written in Python) is done through a Redis in-memory database, managed through the library SmartSim. This setup is schematized in Fig. 1.

The idea of DRL is that an agent receives a state, e.g., some probes located in the domain, and this returns an action that is applied back into the environment, e.g., the mass flow rate of the jet. However, to correctly determine the best actuation, the DRL agent must first undergo a training process. Therefore, during the training, the agent also receives a reward, which represents the magnitude that aims to be optimized. Then, the environment and the agent engage in a trial-and-error process organized in episodes. An episode, in this context, denotes a simulation period wherein the CFD solver and the DRL agent exchange information, including states, actions, and rewards. Following the completion of an episode, this data is used to refine the agent through a training step. The proximal policy optimization (PPO) algorithm is used in this case [10].

Regarding resource allocation, the presented framework uses GPUs for the CFD simulations while the DRL algorithm runs on CPUs. Note that most of the computational time is devoted to the CFD simulations (practically the 100%). The training of the cylinder presented in the results section involved a total of 2,000 episodes and 36 U_∞/D (convective time units) per episode. This took approximately 26 wall-time hours in 4 GPUs using a mesh of $3.3 \cdot 10^6$ grid points. This translates into a computational cost of 104 GPU-h.

In contrast, the simulation for the wing case incurs a higher computational cost due to the increased complexity. Previous investigations [2] suggest that a mesh of about $30 \cdot 10^6$ grid points is necessary to perform LES. However, to significantly reduce the computational cost, the training is conducted in a coarser mesh of about $4.8 \cdot 10^6$

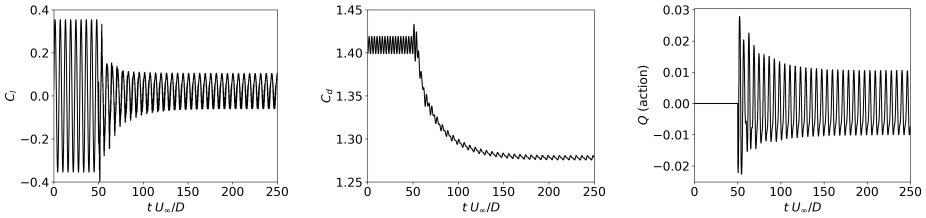


Figure 2: Lift coefficient C_l (left), drag coefficient C_d (middle), and mass flow rate Q (right) for the cylinder case with no control ($tU_\infty/D < 50$) and with DRL control ($tU_\infty/D > 50$).

grid points. This coarser mesh is still capable of capturing the largest scales, which have the most significant impact on the aerodynamic coefficients, thereby making use of an equivalent and computationally cheaper training setup. For the moment, the DRL agent is trained for about 660 episodes, with each episode running for 15 U_∞/c, which is the approximate required non-dimensional time to reach the statistical steady state once the actuation is activated. The training took approximately 144 wall-time hours in 16 GPUs, resulting in a total computational cost of 2,304 GPU-h. All the simulations presented in this study are conducted on the new MareNostrum 5 machine (BSC).

3. Results

3.1. DRL of the flow past a cylinder

First, the aforementioned methodology is tested in a well-known case, the flow past a circular cylinder at $Re = 100$. Here, the DRL control aims at reducing the drag coefficient and, at the same time, the fluctuations of the lift coefficient. For this reason, the reward to train the model is selected to be $r = C_{d,b} - C_d - \alpha|C_l|$; the first part rewarding the reduction of the drag coefficient $C_d = d/(1/2\rho U_\infty^2 S)$ with respect to the baseline scenario $C_{d,b}$, and the second penalizing the increase of the lift $C_l = l/(1/2\rho U_\infty^2 S)$, with α being a weighting factor to adjust the importance of each term. In this case, the reference surface S is defined in terms of the cylinder diameter D and the spanwise length L_z as $S = L_z D$, and the drag d and lift l forces correspond to the resulting components of the aerodynamic force in the streamwise and cross-stream directions relative to the freestream U_∞. To perform the AFC actuations, two jets are located on the top ($\theta = 90°$) and bottom ($\theta = 180°$) surfaces of the cylinder, where their mass flow rate $Q = U_{jet} S_{jet}$ is controlled by the DRL agent; U_{jet} and S_{jet} being the velocity and surface of the jets. The mass flow rates of the jets are set to be opposites, i.e., $Q_{\theta=90°} = -Q_{\theta=180°}$. As observed in Fig. 2, the trained agent effectively reduced the drag of the cylinder by 9.32% and the standard deviation of the lift coefficient signal by 78.4%. Additionally, it displaced the vortex shedding frequency towards smaller values. Concretely, the STROUHAL number ($St = fD/U_\infty$) is reduced from $St = 0.172$ to $St = 0.161$.

3.2. DRL of the separated flow past the SD7003 wing

The SD7003 airfoil at $Re = 60,000$ and $AoA = 14°$ is analyzed by means of large eddy simulations (LES). This case was already investigated by Rodriguez et al. [2], who showed promising results using classical AFC approaches to mitigate the massive flow separation. In the present study, the classical single-frequency approach is initially tested, with jets located at the leading edge of the airfoil ($x/c = 0.007$) and oriented perpendicular to the wall. The actuation is set to a non-dimensional frequency of $F^+ = 1$ and a momentum coefficient of $C_\mu = 0.003$. The non-dimensional frequency is defined as $F^+ = f x_{TE}/U_\infty$, and the momentum coefficient is given by $C_\mu = (h\rho U_{jet,max}^2)/(c\rho U_\infty^2)$, where x_{TE} represents the x-distance from the actuator to the trailing edge, h is the jet streamwise width, $U_{jet,max}$ is the maximum outlet velocity of the jet and c denotes the wing chord. Results are depicted in Fig. 3 (dotted lines) and are compared against the baseline scenario (dashed lines), hence demonstrating that the separation is considerably reduced. The aerodynamic efficiency ($E = C_l/C_d$) is increased by 126%, with a drag reduction of 39% and a lift increase of 38%. Note that here the reference surface S is defined in terms of the wing chord c. The results of the trained DRL agent are also displayed in Fig. 3 (solid lines). Similar parameters as in the classical approach are used, i.e., location and angle of the jets, but in this case, the DRL agent autonomously selects the optimal mass flow rate to maximize aerodynamic efficiency, with the reward being $r = (C_l/C_d - C_{l,b}/C_{d,b})/(C_{l,b}/C_{d,b})$. As observed in Fig. 3, the DRL agent achieved slight improvements over the

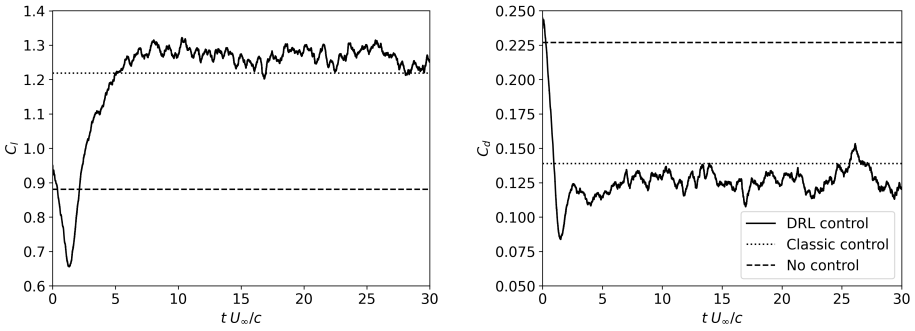


Figure 3: Lift coefficient C_l (left) and drag coefficient C_d (right) for the SD7003 wing case with DRL control.

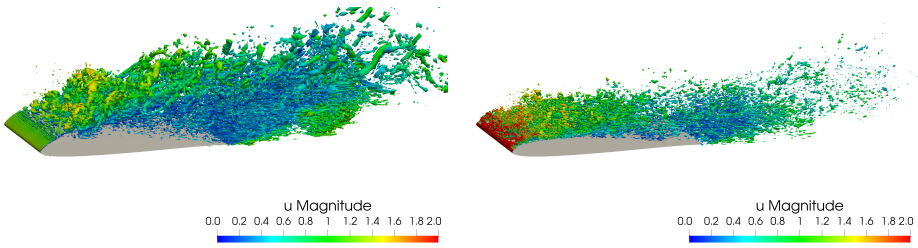


Figure 4: Q-criterion iso-contours illustrating vortical structures in the flow past the SD7003 wing. No control (left) and DRL control (right).

classical AFC, increasing the lift and reducing drag coefficients by 44% each (compared to the baseline scenario), with a 157% increase in aerodynamic efficiency. Thus, the DRL approach further improved efficiency by 31% compared to the classical AFC method. Finally, Fig. 4 presents the Q-criterion iso-contours for both the baseline (no control) and DRL-controlled cases, highlighting the effective reduction of the recirculation bubble achieved by the DRL strategy.

4. Conclusions

This study presents a framework that integrates a CFD solver with a deep reinforcement learning (DRL) model to optimize active flow control (AFC) strategies. The framework is tested on two different flow scenarios: the flow past a circular cylinder at $Re = 100$ and the flow past an SD7003 wing at $Re = 60,000$ and an angle of attack $AoA = 14°$. For the cylinder, the DRL agent shifted the vortex shedding frequency from $St = 0.172$ to $St = 0.161$, contributing to flow stabilization, and yielding a reduction of the drag coefficient and the lift coefficient fluctuations of 9.32% and 78.4%, respectively. On the other hand, for the SD7003 wing, the DRL-controlled setup achieved a 157% increase in the aerodynamic efficiency, improving upon the classical AFC method by an additional 31%.

Overall, the DRL agent demonstrated the ability to further enhance aerodynamic performance beyond traditional AFC methods. The results highlight the potential of DRL to optimize AFC in complex flow environments, dynamically adapting control inputs to achieve improved drag and lift coefficients. However, the results for the wing case are preliminary, and further investigations are needed to explore additional parameters, such as the neural network architecture, episode length, reward formulation, and jet placement and angle, among others. Furthermore, all simulations were conducted using a coarse mesh. Future work aims to transfer the training to a finer mesh after initial training on the coarser one, allowing for results that more accurately reflect the true length scales of the problem.

Acknowledgements

This research has received financial support from the *Ministerio de Ciencia e Innovación* of Spain (PID2020-116937RB-C21 and PID2020-116937RB-C22). Simulations were conducted with the assistance of the *Red Española de Supercomputación*, who granted computational resources to the HPC facility of *MareNostrum 5*, at the Barcelona Supercomputing Center (IM-2024-2-0004). Ricard Montalà work is funded by a FI-SDUR grant (2022 FISDU 00066) from the *Departament de Recerca i Universitats de la Generalitat de Catalunya*. Oriol Lehmkuhl has been partially supported by a *Ramon y Cajal* postdoctoral contract (RYC2018-025949-I). The authors also acknowledge the support of the *Departament de Recerca i Universitats de la Generalitat de Catalunya* through the research group Large-scale Computational Fluid Dynamics (2021 SGR 00902) and the Turbulence and Aerodynamics Research Group (2021 SGR 01051).

References

[1] M. Atzori, R. Vinuesa, A. Stroh, D. Gatti, B. Frohnapfel, P. Schlatter, Uniform blowing and suction applied to nonuniform adverse-pressure-gradient wing boundary layers, Physical Review Fluids 6 (2021) 113904. doi:10.1103/PhysRevFluids.6.113904.

[2] I. Rodriguez, O. Lehmkuhl, R. Borrell, Effects of the Actuation on the Boundary Layer of an Airfoil at Reynolds Number Re = 60000, Flow, Turbululence and Combustion 105 (4) (2020). doi:10.1007/s10494-020-00160-y.

[3] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, Journal of Fluid Mechanics 865 (2019) 281–302. doi:10.1017/jfm.2019.62.

[4] P. Varela, P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, L. García-Cuevas, O. Lehmkuhl, R. Vinuesa, Deep Reinforcement Learning for Flow Control Exploits Different Physics for Increasing Reynolds Number Regimes, Actuators 11 (12) (2022). doi:10.3390/act11120359.

[5] P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, O. Lehmkuhl, R. Vinuesa, Active flow control for three-dimensional cylinders through deep reinforcement learning, in: Ercoftac Symposium on Engineering, Turbulence Modelling and Measurements (ETMM14), Barcelona, 2023. doi:10.48550/arXiv.2309.02462.

[6] L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour, R. Vinuesa, Deep reinforcement learning for turbulent drag reduction in channel flows, European Physical Journal E 46 (27) (2023). doi:10.1140/epje/s10189-023-00285-8.

[7] B. Font, F. Alcántara-Ávila, J. Rabault, R. Vinuesa, O. Lehmkuhl, Active flow control of a turbulent separation bubble through deep reinforcement learning, arXiv:2403.20295 (2024). doi:10.1088/1742-6596/2753/1/012022.

[8] L. Gasparino, F. Spiga, O. Lehmkuhl, SOD2D: A GPU-enabled Spectral Finite Elements Method for compressible scale-resolving simulations, Computer Physics Communication 297 (2024). doi:10.1016/j.cpc.2023.109067.

[9] A. Vreman, An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications, Physics of Fluids 16 (10) (2004) 3670–3681. doi:10.1063/1.1785131.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347 (2017). doi:10.48550/arXiv.1707.06347.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

A Study on the Effect of the Number of Collocation Points in the Training of Physics-Informed Neural Networks for Unsteady Flows

Junya Onishi[a,*], Makoto Tsubokura[a,b]

[a]Center for Computational Science, RIKEN, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan
[b]Graduate School of System Informatics, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan

ARTICLE INFO†

Keywords:
Physics-Informed Neural Networks;
Unsteady Flow;
Collocation Points;
Parallel Training;
Data Parallelism

ABSTRACT

This study investigates the effectiveness of physics-informed neural networks (PINNs) in simulating unsteady flows, with a particular focus on the impact of the number of collocation points used during the training process. A parallel code is implemented based on data parallelism, and its performance and accuracy is assessed by applying it to the advection problem of the volume-of-fluid function. The scalability of the parallel implementation is also assessed, showing that increasing the number of GPUs can enhance computational efficiency. The developed code is further applied to a two-dimensional unsteady flow over a cylinder, with a specific focus on reproducing vortex shedding. The results demonstrate that increasing the number of collocation points improves the accuracy of the PINN's predictions.

1. Introduction

Physics-informed neural networks (PINNs) [1] represent a novel class of deep neural networks (DNNs) that can incorporate both observation data and physical principles, i.e., governing equations and their initial/boundary conditions, into the training process. With this distinctive feature, PINNs are expected to address some of the key drawbacks of conventional DNNs, such as the need for large datasets and the limitation of interpretability. Furthermore, PINNs are regarded as an alternative to traditional numerical methods for solving differential equations, as PINNs can be trained solely with differential equations and their initial and boundary conditions. Such PINNs that are trained without observation data are referred to as data-free PINNs (DF-PINNs).

The validity of DF-PINNs has been already demonstrated for the Burgers equation by Raissi et al. [1] and for various differential equations by other authors. However, there has been limited research on their application to the Navier-Stokes equations, and the validity of DF-PINNs remains uncertain in this context. For instance, Chuang and Barba reported that their PINN failed to reproduce vortex shedding in the simulation of two-dimensional unsteady flow over a cylinder [2]. Similarly, our preliminary tests have shown that DF-PINNs predict steady flow even at sufficiently high REYNOLDS number, as observed in the literature [2]. Moreover, they have also shown that the flow pattern predicted by DF-PINNs is significantly influenced by the number of collocation points per unit time used in the training.

These observations indicate that increasing the number of collocation points per unit time can enhance the prediction accuracy of DF-PINNs. However, there is currently no clear criteria for determining the appropriate number of collocation points in the training of DF-PINNs. This study aims to address this issue by investigating the impact of the number of collocation points on the accuracy and performance of DF-PINNs, particularly in the case of unsteady flow problems. To this end, we implement a parallel code for training DF-PINNs, based on data parallelism, and

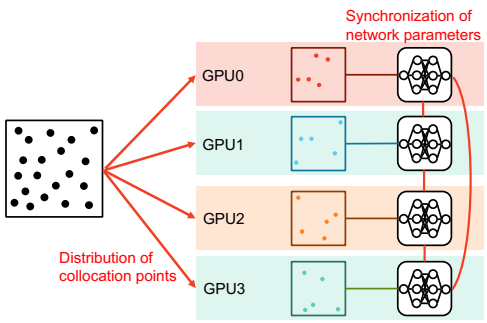
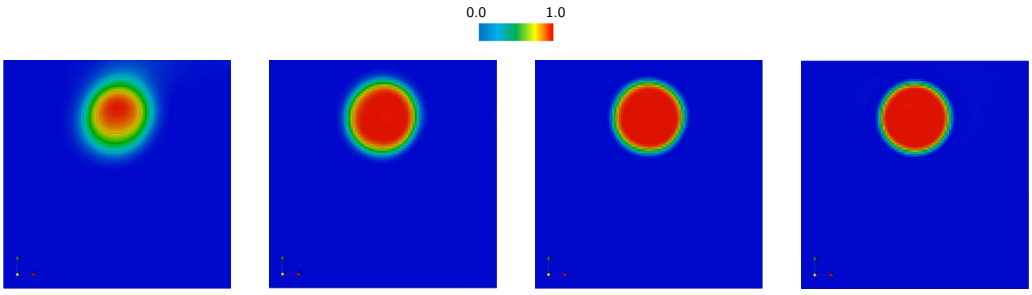


Figure 1: Illustration of the parallelized PINN training process.

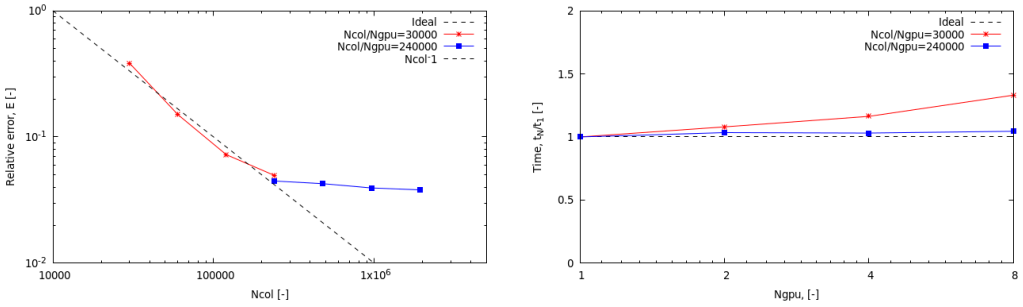
†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02458 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
junya.onishi@riken.jp (J. Onishi); tsubo@tiger.kobe-u.ac.jp (M. Tsubokura)
ORCID(s): 0000-0002-0375-8859 (J. Onishi); 0000-0001-6555-9575 (M. Tsubokura)



(a) $N_{gpu} = 1$. (b) $N_{gpu} = 2$. (c) $N_{gpu} = 4$. (d) $N_{gpu} = 8$.

Figure 2: The distribution of VOF after a single rotation obtained using different number of GPUs N_{gpu}. The number of collocation points assigned to each GPU is fixed to 30,000.



(a) Relative error. (b) Elapsed time.

Figure 3: Results of a the weak-scaling test. The number of collocation points assigned to each GPU is fixed to 30,000 or 240,000.

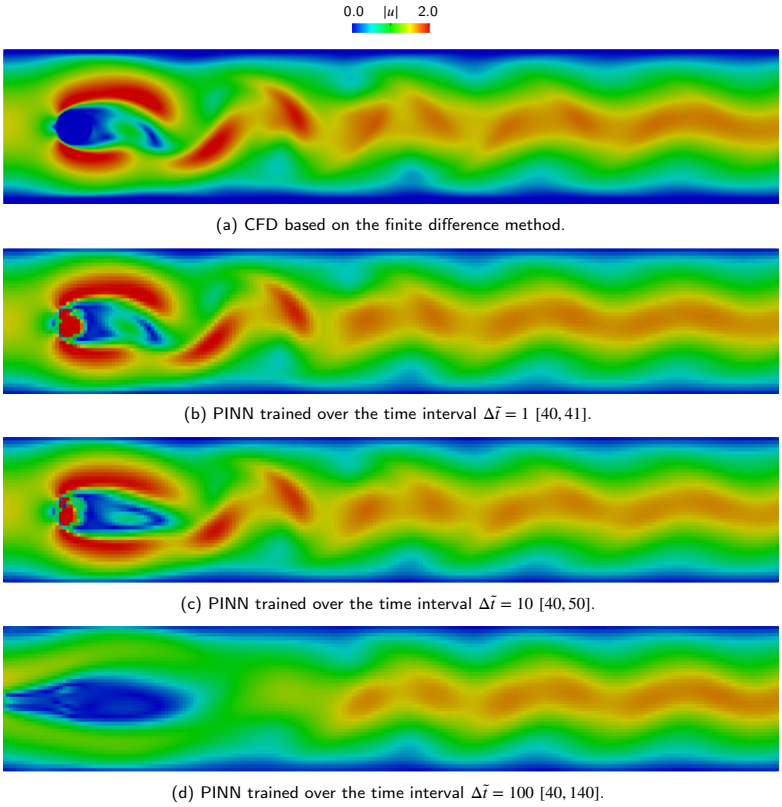
assess its performance and accuracy by applying the code to an advection problem. We further apply the code to two-dimensional unsteady flow over a cylinder and investigate the effect of the number of collocation points to reproduce vortex shedding.

2. Method

The parallelized training process adopted in this study is based on data parallelism, and is illustrated in Fig. 1. At the first step, the collocation points at which the residual of the governing equation is evaluated are distributed to each compute unit, i.e. GPU in this study. Then, the gradients of the loss function with respect to the network parameters (weights and biases) are calculated on each compute unit by utilizing the back-propagation mechanism. Finally, the gradients are averaged by using the `allreduce` operation among all compute nodes and the averaged gradients are used to update the network parameters. This procedure ensures the identity of the neural network on each node.

The architecture of the neural network on each node depends on the problem to be solved. In particular, suitable quantities must be chosen for the input and output of the neural network. For example, in the PINN built for solving two-dimensional unsteady flow, the input layer has three neurons for the spatial position, x and y, and the time, t, and the output layer has three neurons for the velocity components, u and v, and the pressure p. In addition, in the case of the advection problem shown later, the input is the spatial position, x and y, and the time, t, and the output is a scalar function ϕ.

For evaluating the loss of the neural network, the residual of the governing equations, and the disparity between the network's predictions and the ground truth which is specified according to the initial and boundary conditions for the governing equations. Note that these losses are evaluated



(a) CFD based on the finite difference method.

(b) PINN trained over the time interval $\Delta\tilde{t} = 1$ [40, 41].

(c) PINN trained over the time interval $\Delta\tilde{t} = 10$ [40, 50].

(d) PINN trained over the time interval $\Delta\tilde{t} = 100$ [40, 140].

Figure 4: Comparison of velocity distribution around a cylinder at a non-dimensional time $t = 41.0$.

on the collocation points which are selected randomly from spatial and temporal domains.

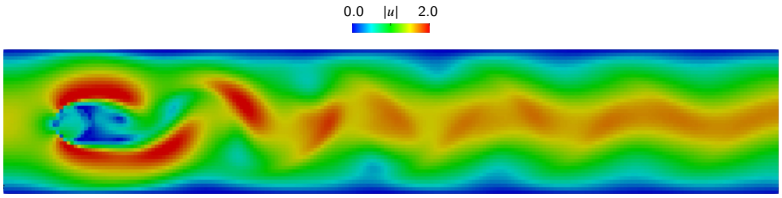
3. Results and discussion

We apply the developed parallel code to advection of the volume of fluid (VOF) function and assess its performance and accuracy.

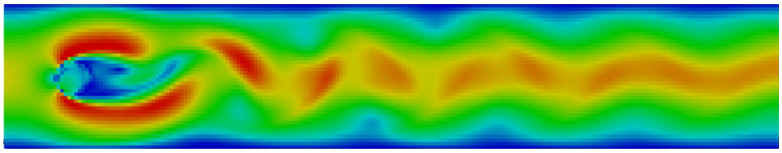
In this assessment, the number of collocation points assigned to each GPU is fixed ($N_{col}/N_{gpu} = 30{,}000, 240{,}000$), and the number of GPUs used in the training is varied. Note that two-thirds of the collocation points are used to evaluate the residual of the governing equation and the rest are used for the initial condition.

First, we clearly observe that increasing the total number of collocation points effectively reduces interface smearing, and thus enhances the accuracy of the PINN prediction, as shown in Fig. 2. We confirm this result quantitatively by evaluating the relative L2 error with respect to the analytical exact solution, as demonstrated in Fig. 3a. Note that the dashed line is drawn according to the scaling $\propto N_{col}^{-1}$. Then, we assess the scalability of the present parallel training by measuring the elapsed time in the case of different number of GPUs. The results of this weak scaling test is shown in Fig. 3b. Note that the training time is normalized by the elapsed time of $N_{gpu} = 1$ in each case. It can be confirmed that the training time does not increase significantly even when a relatively smaller number of collocation points are used ($N_{col}/N_{gpu} = 30{,}000$), and the scalability can be improved when the larger number collocation points are used ($N_{col}/N_{gpu} = 240{,}000$).

Furthermore, we use the code to simulate two-dimensional flow over a cylinder, and investigate the effect of the number of collocation points. In this investigation, we



(a) Snapshot at the initial state at non-dimensional time $t = 40.0$.



(b) Snapshot at non-dimensional time $t = 43.2$.

Figure 5: Snapshots of velocity distribution around a cylinder.

first vary the time interval over which the PINN is trained while keeping the number of collocation points, in order to clarify how many collocation points per time are required to reproduce vortex shedding. The results are summarized in Fig. 4. It can be clearly observed that the PINN solution approaches the CFD solution as the time interval for PINN training becomes shorter, indicating the minimum number of collocation points per time. Then, we increase the time interval for training to cover a full periodic of vortex shedding while keeping the requirements for the number of collocation points per time described above. An example of a periodic vortex shedding simulated using the current PINN is presented in Fig. 5. It can be seen that a full cycle of vortex shedding is well reproduced by the current PINN. Note that in this case the number of collocation points assigned to each GPU is approximately $N_{col}/N_{gpu} = 200{,}000$, including those for the residual of the governing equation, initial condition, and boundary conditions.

4. Conclusions

This study demonstrates the effectiveness of physics-informed neural networks (PINNs) in reproducing cyclic vortex shedding patterns in unsteady flow simulations, even in a data-free training context. It has been demonstrated that the number of collocation points per unit time is critical to the accuracy of predictions. Additionally, it has been shown that the use of data parallelism in PINN training significantly enhances computational efficiency without compromising accuracy. These observations emphasize the potential of PINNs as a reliable and efficient tool for simulating complex fluid dynamics.

Acknowledgements

This research was partly supported by JSPS KAKENHI Grant Number JP22K12058. This research used computational resources of the Fugaku computer provided by the RIKEN Center for Computational Science through the HPCI System Research project (Project ID:hp220180, hp230193).

References

[1] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.

[2] P.-Y. Chuang, L. A. Barba, Predictive Limitations of Physics-Informed Neural Networks in Vortex Shedding (2023). arXiv:2306.00230.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Predicting Turbulent Boundary Layer Flows Using Transformers Coupled to the Multi-Physics Simulation Tool m-AIA

Rakesh Sarma[a,*], Fabian Hübenthal[b], Fabian Orland[c], Christian Terboven[c] and Andreas Lintermann[a]

[a]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52425 Jülich, Germany
[b]RWTH Aachen University, Institute of Aerodynamics and Chair of Fluid Mechanics, Wüllnerstraße 5a, 52062 Aachen, Germany
[c]RWTH Aachen University, Chair for High-Performance Computing, Seffenter Weg 23, 52074 Aachen, Germany

ARTICLE INFO†

Keywords:
Transformers;
Turbulent Boundary Layers;
Computational Fluid Dynamics / Artificial Intelligence Coupling

ABSTRACT

Time-marching of turbulent flow fields is computationally expensive with traditional numerical solvers. In this regard, transformer neural network, which has been largely successful in many other technical and scientific domains, can potentially predict complex flow fields faster compared to physics-based solvers. In this study, a transformer model is trained for a turbulent boundary layer problem, which is then coupled to the multi-physics solver m-AIA to make predictions of velocity fields. The method can potentially contribute to significant reduction in computational effort while maintaining high accuracy.

1. Introduction

Numerical prediction of turbulence remains challenging due to its multiscale nature that requires highly-resolved simulations to accurately capture the temporal and spatial dynamics [1]. In the Computational Fluid Dynamics (CFD) community, numerical solvers capable of simulating complex turbulent dynamics have been developed, albeit demanding substantial computational resources and high resolutions. Alternatively, Machine Learning models, for instance based on Convolutional Neural Networks [2], have emerged as promising alternatives. The use of transformer architecture-based models for time-marching turbulent fields have been limited to few recent successful efforts, often for prediction of compressed representations of the flow field to reduce computational effort [3, 4], which may suppress information of the high-frequency components. Nonetheless, transformers can potentially be an effective neural network to perform complex long-term temporal predictions of turbulent flows, while allowing distributed training given the multi-head attention configuration. In this study, the transformer architecture is applied to the prediction of a Turbulent Boundary Layer (TBL) problem, with a unique reconstruction strategy of the input velocity fields, which allows the prediction of the full velocity field without any intermediate compression step. Furthermore, the trained model is coupled to the highly-parallel multi-physics simulation tool m-AIA formerly known as the Zonal Flow Solver (ZFS) [5], which was further developed towards m-AIA. The coupling framework is needed to couple the physical solver (in this case, m-AIA) to the distributed deep learning inference with the transformer. Replacing m-AIA-based expensive time-marching of the turbulent fields with the transformer model is expected to significantly reduce the computational costs.

2. Turbulent boundary layer problem specification

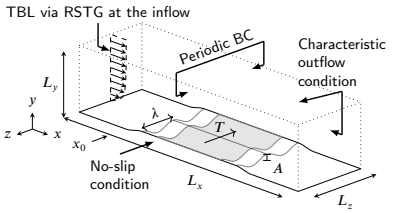


Figure 1: Sketch of the CFD domain with the three-dimensional actuated turbulent boundary layer flow. The gray area indicates the region of interest where the TBL data is extracted.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02459 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
r.sarma@fz-juelich.de (R. Sarma);
f.huebenthal@aia.rwth-aachen.de (F. Hübenthal);
orland@itc.rwth-aachen.de (F. Orland); terboven@itc.rwth-aachen.de (C. Terboven); a.lintermann@fz-juelich.de (A. Lintermann)
ORCID(s): 0000-0002-7069-4082 (R. Sarma); 0009-0000-7159-8220 (F. Hübenthal); 0000-0002-8681-266 (F. Orland); 0000-0003-2284-2957 (C. Terboven); 0000-0003-3321-6599 (A. Lintermann)

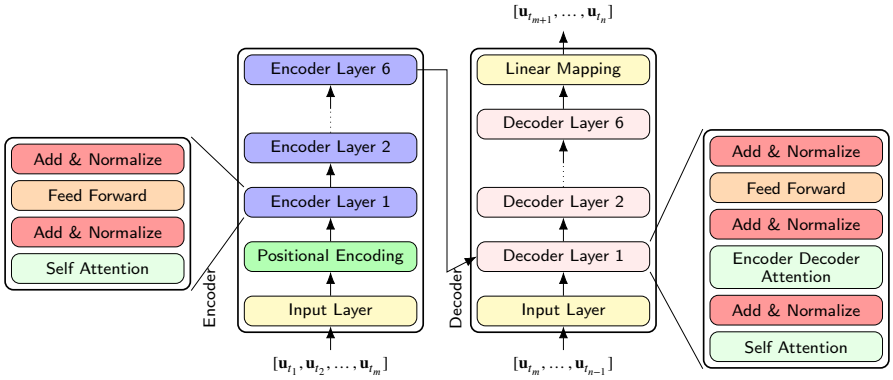


Figure 2: Architecure of the transformer model based on encoder-decoder configuration adapted from [8].

The CFD model is based on a validated Zero-Pressure Gradient flat plate approximation of the active drag reduction technique using spanwise traveling transversal surface waves. Wall-resolved Large-Eddy Simulation (LES) is performed using the in-house CFD solver m-AIA[1] [6, 7]. The physical domain of the flat plate model is given in Fig. 1, where the dimensions in the Cartesian directions are L_x, L_y and L_z. The actuation parameters are the wavelength λ, the time period T and the amplitude A. At the inflow of the domain, the reformulated synthetic turbulence generation (RSTG) method is used to initiate a TBL flow. The onset of the surface actuation, analyzed in [6, 7], is located at x_0, where a fully developed TBL is established. The surface area A_{surf} for the integration of the wall-shear stress τ_{w} is shaded in gray. Periodic Boundary Conditions (BC) are used in the spanwise direction z, characteristic outflow conditions are applied on the downstream and upper boundaries, and the no-slip condition is imposed on the wall [6].

Further details on the numerical method, the computational setup, validation of the LES and BC can be found in Albers et al. [6].

3. Transformer for temporal prediction

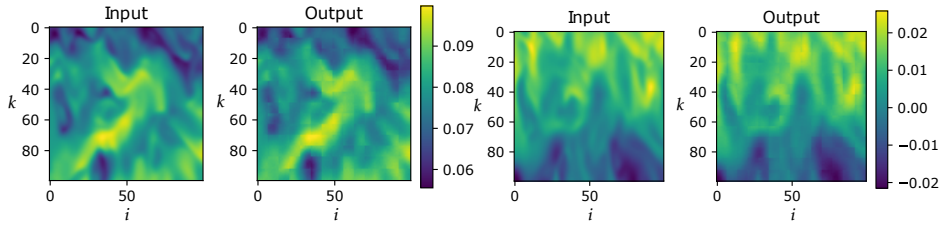
The transformer architecture is adapted from an encoder-decoder configuration, used for temporal predictions [8], which is shown in Fig. 2. The encoder consists of an input layer, positional encoding and a stack of six encoding layers, where each layer consists of a self-attention and a fully connected layer, followed by a normalization layer. The input layer is a fully-connected network and the positional encoding consists of sine and cosine functions. For the decoder layers, there is additional layer to apply self-attention over encoder outputs, where the input is from the encoder output and the decoder output is a linear mapping to the target sequence. Look-ahead masks are applied to ensure that the decoder only sees information from the previous time-steps.

If the training dataset consists of n velocity fields at time-instances $t_1, t_2, \ldots, t_n$, the encoder takes in a sequence of $\mathbf{u}_{t_1}, \mathbf{u}_{t_2}, \ldots, \mathbf{u}_{t_m}$ as input, and the decoder outputs the velocities at time-instances $\mathbf{u}_{t_{m+1}}, \ldots, \mathbf{u}_{t_n}$, where $1 < m < n$. Here, $\mathbf{u}$ is the velocity field vector, m is the encoder sequence length and $n - m$ is the target sequence length. The decoder input in this case would consist of velocities at time-instances $\mathbf{u}_{t_m}, \ldots, \mathbf{u}_{t_{n-1}}$. The input velocity field is reshaped to smaller cubic sub-domains (8^3 for this study), where each sub-domain is treated as a separate batch by the transformer. This allows to limit the number of features that the model needs to predict, thus reducing the complexity of the self-attention mechanism. Furthermore, 16 attention heads are employed and the Adam optimizer is used for training. The trainings are conducted with the DeepSpeed[2] framework in a distributed training setup provided by the AI4HPC[3] library. Exemplary predictions by the transformer model of the streamwise (u) and spanwise (w) components of the velocity field are shown in Fig. 3. It can be seen that the model is able to provide good qualitative predictions of the velocity field. This trained model is then coupled to the m-AIA solver.

[1] m-AIA https://git.rwth-aachen.de/aia/m-AIA/m-AIA

[2] DeepSpeed https://github.com/microsoft/DeepSpeed

[3] AI4HPC https://ai4hpc.readthedocs.io/en/latest/



(a) Streamwise (u) component (b) Spanwise (w) component

Figure 3: Exemplary prediction (Output) from the transformer of TBL velocity field slice in the wall-normal direction on the i-k plane, compared to the original LES field (Input), where i is the streamwise and k is the spanwise direction. In this case, Input and Output refer to the LES-predicted and transformer-predicted velocity field at a future time-step.

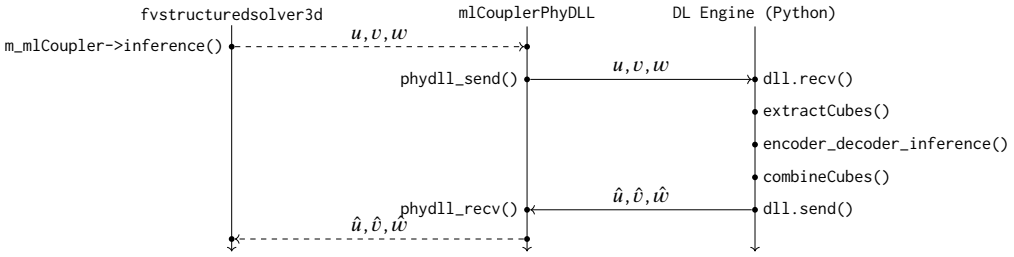


Figure 4: Coupling workflow between m-AIA and PhyDLL.

4. Coupling m-AIA with the transformer

Coupling m-AIA with the trained transformer is achieved using the open-source "Physics Deep Learning coupLer" (PhyDLL)[4], which provides a coupling mechanism for parallel physical solvers via the Message Passing Interface (MPI) using the Multiple Program Multiple Data paradigm. It establishes a mapping between processes running the physical solver and processes via a user-defined DL Engine. In this work, the DL Engine is a Python script, which implements the inference of the transformer. To communicate data between the physical solver and the DL Engine, PhyDLL offers a Fortran, C, and Python interface around MPI's point-to-point operations.

The coupling workflow is illustrated in Figure 4. Inside m-AIA's 3D structured solver (`fvstructuredsolver3d`), **u** is passed to the `mlCouplerPhyDLL` as three one-dimensional fields (u,v,w). This class is derived from an abstract class following the popular strategy design pattern proposed by Gamma et al. [9] to keep the solver independent from the actual coupling algorithm. The `mlCouplerPhyDLL` uses PhyDLL's API to send the flow fields to the DL Engine. First, the 3D flow field is divided into cubic subdomains inside the function `extractCubes()`. The DL Engine manages a ring buffer, storing the data from the m most recent timesteps, to construct sequences of cubic subdomains as input to the transformer. After the input has been constructed, the inference of the transformer is implemented as proposed by Ludvigsen[5]. Multiple forward passes through the transformer are performed to create a suitable target sequence of length $n-m$, which the model can decode to make the final prediction. Afterwards, the function `combineCubes()` combines the cubic subdomains from the final prediction to form the full flow field ($\hat{u}$,$\hat{v}$,$\hat{w}$) again. The final predicted flow field is then sent back to the `mlCouplerPhyDLL` of m-AIA using PhyDLL's API. Finally, the flow field in m-AIA is updated and the simulation continues based on the transformer's prediction.

5. Conclusions

The manuscript explored the use of a transformer model to perform coupled predictions of TBL velocity fields with the CFD solver m-AIA. For this, a transformer model is trained with full velocity field data that is restructured into

[4] PhyDLL `https://phydll.readthedocs.io/en/latest`

[5] `https://github.com/KasperGroesLudvigsen/influenza_transformer`

smaller cubic sub-domains. The trained model, which shows good qualitative agreement, is being coupled to the m-AIA solver using the PhyDLL framework, which is an ongoing work at the time of writing this abstract. Therefore, so far qualitative analyses are performed based on the comparison of outputs from the CFD solver and the transformer model. The proposed methodology promises to accurately predict turbulent fields, while significantly reducing the computational effort for time-marching TBL fields by reducing the number of time-steps computed by the physical solver. For a single time step, computational savings of upto 53 times was possible during transformer inference, while also reducing the memory consumption by 1,100 times. As an outlook for the conference talk, quantitative analyses are performed to assess the prediction accuracy also for the coupled configurations and the speed-up compared to the stand-alone CFD solver. Furthermore, the conservation properties (in terms of mass and momentum) of the time-marched velocity fields obtained through the interaction of the CFD solver and the transformer model will be analyzed, and the imbalances will be quantified in future work.

Acknowledgements

The research leading to these results has been conducted in the CoE RAISE project, which receives funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

References

[1] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence Modeling in the Age of Data, Annual Review of Fluid Mechanics 51 (2019) 357–377. doi:10.1146/annurev-fluid-010518-040547.

[2] Y. Wang, Z. Xie, K. Xu, Y. Dou, Y. Lei, An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning, Neurocomputing 174 (2016) 988–998. doi:10.1016/j.neucom.2015.10.035.

[3] A. Solera-Rico, C. Sanmiguel Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. T. M. Dawson, R. Vinuesa, β-variational autoencoders and transformers for reduced-order modelling of fluid flows, Nature Communications 15 (2024) 1361. doi:10.1038/s41467-024-45578-4.

[4] M. Z. Yousif, M. Zhang, L. Yu, R. Vinuesa, H. Lim, A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers, Journal of Fluid Mechanics 957 (2023) A6. doi:10.1017/jfm.2022.1088.

[5] A. Lintermann, M. Meinke, W. Schröder, Zonal Flow Solver (ZFS): a highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34 (7-8) (2020) 458–485. doi:10.1080/10618562.2020.1742328.

[6] M. Albers, P. S. Meysonnat, D. Fernex, R. Semaan, B. R. Noack, W. Schröder, Drag reduction and energy saving by spanwise traveling transversal surface waves for flat plate flow, Flow, Turbulence and Combustion 105 (1) (2020) 125–157. doi:10.1007/s10494-020-00110-8.

[7] D. Fernex, R. Semaan, M. Albers, P. S. Meysonnat, W. Schröder, B. R. Noack, Actuation response model from sparse data for wall turbulence drag reduction, Physical Review Fluids 5 (7) (2020) 073901. doi:10.1103/PhysRevFluids.5.073901.

[8] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, ArXiv 2001.08317 (2020). doi:10.48550/arXiv.2001.08317.

[9] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1st Edition, Addison-Wesley Professional, 1994.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Creating a Virtual Population of the Human Nasal Cavity for Velocity-Based Predictions of Respiratory Flow Features Using Graph Convolutional Neural Networks

Hadrien Calmet[a,*], Joan Calafell[a], Rakesh Sarma[b], Mario Rüttgers[b], Andreas Lintermann[b] and Guillaume Houzeaux[a]

[a]*Barcelona Supercomputing Center, Department of Computer Applications in Science and Engineering (CASE), Plaça d'Eusebi Güell, 1-3, 08034 Barcelona, Spain*
[b]*Forschungszentrum Jülich (FZ) GmbH, Jülich Supercomputing Centre (JSC), Wilhelm-Johnen-Straße, 52425 Jülich, Germany*

ARTICLE INFO[†]

Keywords:
Virtual Population;
Computational Fluid Dynamics;
Deep Learning;
Graph Convolutional Neural Network;
Respiratory System;
Data Augmentation

ABSTRACT

GNNs can be applied to any shape or volume represented by a graph, e.g., triangulated shapes, or computational grids. Convolutional filters in GNNs operate on nodes and their neighboring nodes. This allows more efficient training compared to convolutional neural networks (CNNs), whose convolutional filters are rectangular and operate in Cartesian directions. The goal is to predict respiratory system flow features such as air resistance, wall shear stress, and energy flux within the human nasal cavity during inspiration. The initial step involves generating a virtual population through random scaling applied simultaneously to length, width, and height. Three distinct geometries are chosen to generate 297 virtual patients, including an average one based on 35 healthy patients, a Caucasian healthy patient, and an Asian healthy patient. The second part of the talk exposes the preliminary results based on 297 patients with physiological observations and discussions on the accuracy result of the GCNN model.

1. Introduction

The pressure loss inside the nasal airway is an important indicator to estimate a patient's breathing condition [1, 2]. Moreover, the airflow features of nasal respiration are closely related to the Wall-Shear Stress (WSS) distribution. When air flows through the nasal passages, it comes into contact with the nasal mucosa, creating frictional forces. These frictional forces result in the generation of WSS, which plays a crucial role in various physiological functions of the nasal cavity, including the air resistance. As a consequence, altering the WSS distribution may impact the breathing efficiency.

The high geometrical complexity of the nasal cavity causes complicated laminar, transitional, and turbulent flow patterns depending on the considered respiratory flow rate. These complex dynamics require high-fidelity CFD-based modeling techniques, for which computational challenges are significant.

Fast Deep Learning (DL) models utilizing graph architectures and inductive methods have demonstrated significant potential in efficiently predicting fluid mechanical properties within complex geometries across meshes of varying resolutions [3]. This capability is exploited in the present study to address the aforementioned limitations of CFD, enabling the development of accurate predictive tools for coarser meshes and thereby streamlining the pre-processing step.

In the present study, a GCNN and volumetric velocity data-based approach for predicting respiratory flow features, such as the pressure loss and the WSS distribution, is investigated. To overcome a lack of measured velocity data, the GCNN is trained with velocity fields from high-fidelity CFD simulations, a valid approach considering the proven accuracy of this methodology [3].

2. Method

In bioengineering, obtaining a significant amount of patient geometries is a work-intensive undertaking, primarily due to confidentiality policy considerations and limited access to hospital databases. The tedious task of image segmentation adds another layer of complexity to this process. Additionally, training of deep learning models demands a

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02460 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
hadrien.calmet@bsc.es (H. Calmet); joan.calafell@bsc.es (J. Calafell); r.sarma@fz-juelich.de (R. Sarma); m.ruettgers@fz-juelich.de (M. Rüttgers); a.lintermann@fz-juelich.de (A. Lintermann); guillaume.houzeaux@bsc.es (G. Houzeaux)
ORCID(s): 0000-0001-5443-761X (H. Calmet); 0000-0002-2333-7314 (J. Calafell); 0000-0002-7069-4082 (R. Sarma); 0000-0003-3917-8407 (M. Rüttgers); 0000-0003-3321-6599 (A. Lintermann); 0000-0002-2592-1426 (G. Houzeaux)

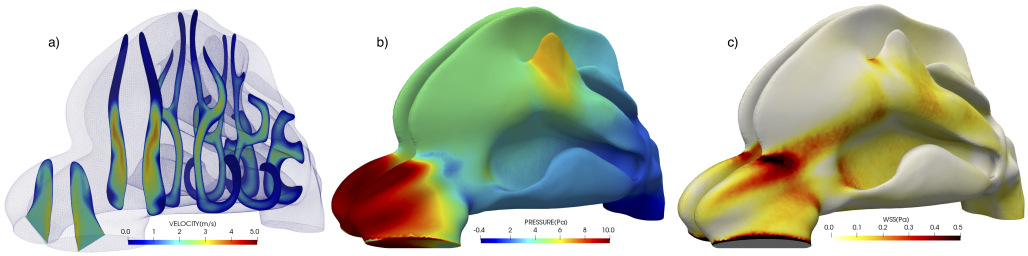


Figure 1: Numerical results of the patient 86: a) Magnitude of velocity vector, b) Map of pressure and c) Map of wall shear stress.

substantial amount of data to nourish the algorithms adequately. To bypass this challenge, the authors constructed a virtual population using three geometries, employing random scaling simultaneously across length, width, and height. This data augmentation resulted in the generation of 297 virtual patients with 99 virtual patients per original geometry.

For each geometry, an unstructured mesh with prism layers is crafted using the software ANSYS-ICEM-CFD (Ansys Inc., USA). Further insights into the meshing process can be found in the work by [4]. The high performance computational mechanics code Alya [5], developed at the Barcelona Supercomputing Centre (BSC), is used to solve the Navier-Stokes equations. The Dakota software (Sandia National Laboratories) [6] is used to set up and execute 297 cases with random scaling across length, width, and height for each model. The toolkit software DARE, which is developed at the BSC, is used as an automation software.

A no-slip boundary condition is imposed on the nasal airway walls. The inlet velocity profile is imposed as a Dirichlet condition on the inlet flat surface of the nostril. A constant flow rate was set to Q=20L/min. This is frequently used in the literature and considered as normal constant inspiration flow rate [7, 8, 9]. A zero-traction outflow condition is imposed as a Neumann condition (the surface is free from external stress) at the outlet of the nasopharynx. Figure 1 displays numerical results for variation 86 of the first patient, as example, showing three different airflow features: the velocity contour fields, pressure map, and the WSS map. This patient

The GCNN architecture used in this study is based on [3], where a method for data-driven prediction from flow fields defined on irregular and unstructured meshes, using a GCNN framework, is presented. The GCNN training is performed in a distributed training setup. This is achieved through data distributed training where each worker (e.g., a GPU) performs training on a subset of the data samples. The open-source library AI4HPC[1] is utilized in this work. The parallel computations are performed on the DEEP system at JSC. Further insights into the parallelism process can be found on the AI4HPC website[2].

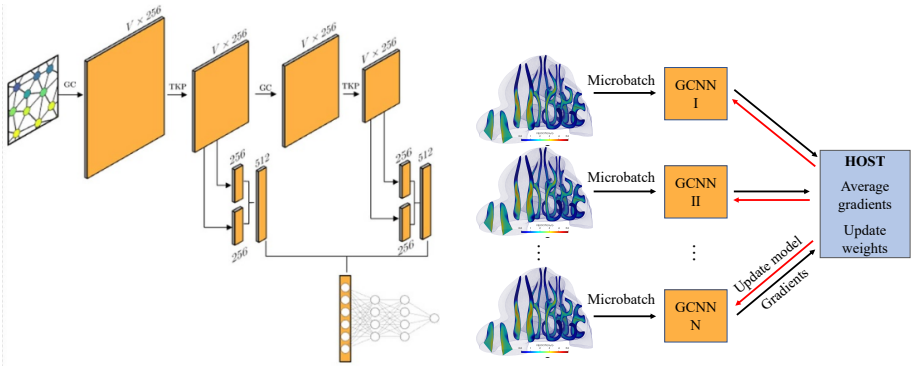
3. Results

The objective is to leverage the dataset created to predict physiological aspects, such as air resistance or the WSS. As an initial step, all available data samples (297 patients) are employed to train a model, aiming to assess the model's capability of performing the required regression task. That is, taking a large spatial graph containing nodal velocity data and determining the pressure drop across the geometry with a minimum mean squared error. Then, following the successful completion of this preliminary test, the generalization capabilities of the model are evaluated by splitting the dataset, allocating 80% for training and 20% for validation. The model's demonstrate excellent performance in predicting the pressure drop, achieving a Mean Squared Error (MSE) of 0.0002, and mean accuracy, which is measured as the relative error between the model output and the target output, of 99.85% for the validation sets.

A comparison of the ground truth values of the pressure drop from CFD result and the predictions from the network qualitatively shows the high accuracy of the model for both datasets, see Fig. 3. As can be seen, the validation points (blue dots) are selected to cover the entire range of the pressure drop, demonstrating high accuracy across all patient ranges. The current model achieves a relative validation error of 0.86% in predicting the pressure drop.

[1] https://ai4hpc.readthedocs.io/en/latest/index.html

[2] https://ai4hpc.readthedocs.io/en/latest/AI4HPC/distributedfws.html



(a) Architecture of the GCNN (b) Distributed training

Figure 2: Schemes of the architecture of the GCNN and distributed training for N workers: a), the architecture of the GCNN. 'GC' refers to the graph convolution operation, 'TKP' refers to Top-K pooling and 'V' to the velocity field, image from the courtesy of the authors [3]; b), the architecture of the distributed training framework AI4HPC.

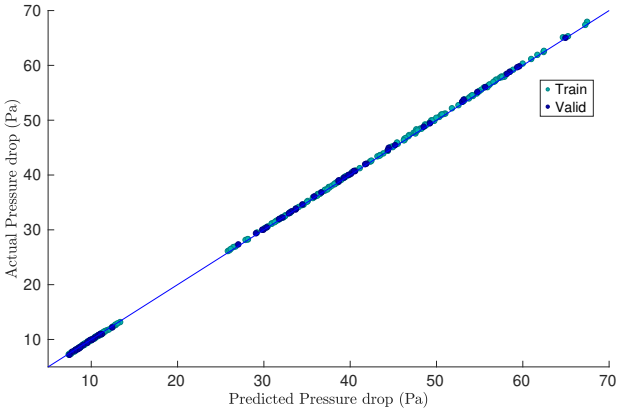


Figure 3: A comparison of GCNN predictions with the ground truth pressure drop. The training and validation datasets contain 236 and 61 samples, respectively.

4. Conclusions

The current study investigates a velocity data-based approach using GCNNs to predict respiratory flow features, including pressure loss. Understanding nasal physiology and respiratory function requires studying nasal airway resistance and wall shear stress in the nasal cavity and their relationship with airflow. Developing machine learning models capable of achieving a relative validation error of 0.86% for the air resistance can be highly beneficial for the development of treatments for nasal disorders.

References

[1] M. Waldmann, A. Grosch, C. Witzler, M. Lehner, O. Benda, W. Koch, K. Vogt, C. Kohn, W. Schröder, J. H. Göbbert, A. Lintermann, An effective simulation- and measurement-based workflow for enhanced diagnostics in rhinology, Medical & Biological Engineering & Computing 60 (2) (2022) 365–391. doi:10.1007/s11517-021-02446-3.

[2] M. Rüttgers, M. Waldmann, W. Schröder, A. Lintermann, A machine-learning-based method for automatizing lattice-Boltzmann simulations of respiratory flows, Applied Intelligence 52 (8) (2022) 9080–9100. doi:10.1007/s10489-021-02808-2.

[3] F. Ogoke, K. Meidani, A. Hashemi, A. B. Farimani, Graph convolutional networks applied to unstructured flow field data, Machine Learning: Science and Technology 2 (4) (2021) 045020. doi:10.1088/2632-2153/ac1fc9.

[4] H. Calmet, D. Oks, A. Santiago, G. Houzeaux, A. Le Corfec, L. Deruyver, C. Rigaut, P. Lambert, B. Haut, J. Goole, Validation and sensitivity analysis for a nasal spray deposition computational model, International Journal of Pharmaceutics 626 (2022) 122118. doi:10.1016/j.ijpharm.2022.122118.

[5] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, et al., Alya: Multiphysics engineering simulation toward exascale, Journal of Computational Science 14 (2016) 15–27. doi:10.1016/j.jocs.2015.12.007.

[6] B. Adams, W. Bohnhoff, K. Dalbey, M. Ebeida, J. Eddy, M. Eldred, R. Hooper, P. Hough, K. Hu, J. Jakeman, et al., Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.13 user's manual., Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States) (2020). doi:10.2172/1829573.

[7] H. Shi, C. Kleinstreuer, Z. Zhang, Modeling of inertial particle transport and deposition in human nasal cavities with wall roughness, Journal of Aerosol Science 38 (4) (2007) 398–419. doi:10.1016/j.jaerosci.2007.02.002.

[8] H. Shi, C. Kleinstreuer, Z. Zhang, Dilute suspension flow with nanoparticle deposition in a representative nasal airway model, Physics of Fluids (1994-present) 20 (1) (2008) 013301. doi:10.1063/1.2833468.

[9] H. Calmet, C. Kleinstreuer, G. Houzeaux, A. Kolanjiyil, O. Lehmkuhl, E. Olivares, M. Vázquez, Subject-variability effects on micron particle deposition in human nasal cavities, Journal of Aerosol Science 115 (2018) 12–28. doi:10.1016/j.jaerosci.2017.10.008.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Evaluating Random Forest Classifiers to Optimize Load Balancing of Parallel Mesh Generation

Ananya Gangopadhyay[a,*], Paul Bartholomew[a] and Michèle Weiland[a]

[a]The University of Edinburgh, EPCC, Bayes Centre, 47 Potterrow, Edinburgh, EH8 9BT, UK

ARTICLE INFO[†]

Keywords:
Computational Fluid Dynamics;
Mesh Generation;
Domain Decomposition;
Load Balancing;
Machine Learning;
Parameter Optimization

ABSTRACT

Mesh generation performance with OpenFOAM's *snappyHexMesh* utility is impacted by the decomposition method used for load balancing on HPC systems. Testing shows that 3D hierarchical decomposition with optimal configuration can significantly improve mesh generation runtimes. However, this optimal configuration depends on several factors, such as geometric complexity and mesh resolution requirements. We evaluate the use of random forest classifiers to select optimal subdomain allocations with minimal interaction from the user. Our current implementation achieves a 76% classification accuracy on a limited dataset, showing potential for improvements in future work.

1. Introduction

Mesh generation is a key pre-processing step in a Computational Fluid Dynamics (CFD) design and analysis cycle that is executed prior to the solver phase [1]. However, with long runtimes that can be comparable to that of the solver itself, meshing can be a significant bottleneck [2]. The *snappyHexMesh* (SHM) utility in the OpenFOAM CFD toolkit[1] aims to minimize runtimes through parallelization (using MPI), and it uses graph decomposition to distribute the meshing task across processes. For many OpenFOAM *cases*[2], the mesh generation step often scales poorly on large high-performance computing (HPC) platforms. As mesh generation has traditionally been conducted on workstations, this has been largely overlooked as an optimization concern [3]. The poor scaling behavior becomes evident when analyzing the complete execution of the SHM utility, for the Onera M6 Wing[3] modeled with OpenFOAM[4], on the ARCHER2 system[5]. As seen in the plots in Fig. 1, while increasing the number of cores provides notable reduction in runtimes initially, the runtimes level off before increasing again (Fig. 1a) and the parallel efficiency declines throughout (Fig. 1b). The analysis also shows that OpenFOAM's *hierarchical* 3D geometric decomposition method has the lowest runtimes and exhibits better parallel efficiency than the third-party libraries: METIS [4, 5], Scotch [6], PT-Scotch [7] and KaHIP [8].

Profiling indicates that at higher core counts, the cost of the MPI calls significantly outweighs the compute work, resulting in poor parallel efficiency. However, these calls predominantly take place when recomputing the decomposition, which is triggered due to load imbalance that occurs as the mesh evolves. While this is true for all the methods, the communication overhead scales more favorably for the hierarchical method. The advantage of the third-party methods is to optimize interprocessor communication, while geometric decomposition aims to keep the workload fully balanced across processors. As communication is less frequent in the mesh generation process relative to the solver, hierarchical decomposition can provide the most balanced parallel execution without the higher communication overhead that occurs in the third-party methods. In fact, a case study by Amazon Web Services (AWS) recommends running SHM with hierarchical decomposition[6].

Users can configure the hierarchical decomposition process by providing n: the number of subdomains in x, y, and z. Further optional configuration can come from altering the sorting order from the default xyz to yxz, zyx etc. These configuration changes can have a significant impact on performance: Table 1 shows the runtimes of the mesh generation step with SHM for the Onera M6 Wing, with an approximately 2× difference between the worst and the best configuration.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02461 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
a.gangop@ed.ac.uk (A. Gangopadhyay); p.bartholomew@epcc.ed.ac.uk (P. Bartholomew); m.weiland@epcc.ed.ac.uk (M. Weiland)
ORCID(s): 0000-0002-4948-9674 (A. Gangopadhyay); 0000-0001-7413-670X (P. Bartholomew); 0000-0003-4713-3073 (M. Weiland)

[1]https://www.openfoam.com/

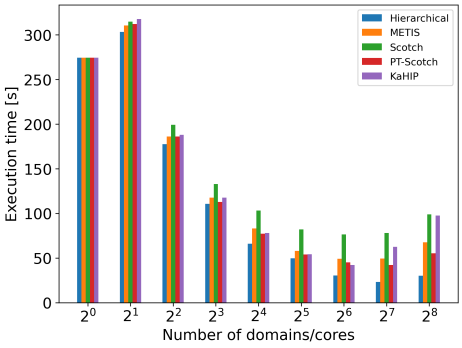
[2]https://www.openfoam.com/documentation/user-guide/2-openfoam-cases

[3]https://www.grc.nasa.gov/www/wind/valid/m6wing/m6wing.html

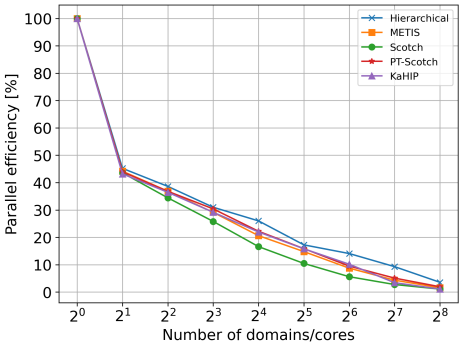
[4]https://wiki.openfoam.com/OneraM6_by_Michael_Alletto

[5]https://www.archer2.ac.uk

[6]https://aws.amazon.com/blogs/hpc/getting-the-best-openfoam-performance-on-aws/



(a) Execution time per core per decomposition method.



(b) Parallel efficiency per core per decomposition method.

Figure 1: Analyzing the performance of the *snappyHexMesh* utility for the Onera M6 Wing[4]. METIS, Scotch, PT-Scotch and KaHIP methods use default parameters. Hierarchical uses "balanced" subdomain allocations (i.e., $2 \times 2 \times 2$ for 8 cores, $4 \times 8 \times 4$ for 128 cores, etc.) with default xyz sorting order.

n_x	n_y	n_z	Execution time [s]
1	1	8	145.80
1	2	4	100.00
1	4	2	75.56
1	8	1	79.92
2	1	4	108.03
2	2	2	76.93
2	4	1	78.11
4	1	2	76.45
4	2	1	80.77
8	1	1	84.35

Table 1
Runtimes for the Onera M6 Wing[4] mesh generation with different subdomain allocations on 8 cores of ARCHER2[5] with default xyz sorting order.

In OpenFOAM, most cases start off as a structured mesh created using the *blockMesh* utility, which is easily decomposed using the *decomposePar* utility. However, while generating the final mesh, SHM creates new cells during its *refinement* and *layer addition* phases. These are managed with a load re-balancing step after almost every iteration in the *refinement* phase and once at the end of the *layering* phase. During this step, the mesh generation is paused and the new decomposition is determined based on the selected method. The mesh is then re-distributed across cores, after which the mesh generation is resumed. The frequency of load re-balancing is governed by the user-provided *maxLoadUnbalance* and *maxLocalCells* settings: load balancing occurs if either of these is violated. As the final cell count and the locations of high mesh resolution are not known at the start, it is non-trivial to determine the decomposition and load balancing settings to minimize the need to re-balance. While this could lead CFD engineers to use more "automated" methods like Scotch[7], this might result in a loss of valuable resources and an inefficient workflow due to poor parallel efficiency.

The CFD Vision 2030 [2] stipulates that mesh generation should ideally be more automatic and require limited interaction from the user. On this basis, we investigate if machine learning (ML) can be used to automatically determine the best hierarchical decomposition subdomain allocation, or *"Optimum n"* for an OpenFOAM mesh generation case. Our aim is to make the hierarchical decomposition as easy to use as one of the third-party libraries by simplifying the configuration step.

2. Methodology

2.1. Method selection

OpenFOAM mesh generation settings can be consolidated in a tabulated form as nearly all configurations affecting the final mesh and the generation process are numerical. While many ML algorithms can operate on structured, tabulated datasets, decision trees are established as the ML method of choice for datasets with around 10,000 samples [9] as they offer a good balance between speed and accuracy. Random forests, i.e., collections of decision trees trained on different subsets of the training data through bootstrapping, improve on decision trees by reducing the risk of overfitting. This comes with the disadvantage of making

[7] https://www.cfdengine.com/newsletter/091/

the logic behind the final decision harder to interpret as the method aggregates over several possibilities and combinations of the data [10, 11]. However, it also provides a more balanced measure of a feature's contribution to the decision making. We have implemented our ML model as a *random forest classifier*[8] with the *scikit-learn* [12] framework, where the inferred class is a string representing the *Optimum n* in the form of "$n_x \times n_y \times n_z$".

Although the mesh generation and decomposition settings can be easily tabulated, representing the geometries with scalar values requires further pre-processing. Using the *Open3d*[9] package in Python, geometries in `.stl` and `.obj` formats can be analyzed and manipulated. The geometry's orientated bounding box provides an approximation of its lengths and orientation, and its voxel grid representation gives an estimate of its volume. The current implementation focuses only on SHM's refinement phase, which employs load re-balancing frequently.

2.2. Dataset generation

Our dataset is created to tie the geometries' complexities and refinement requirements to the decomposition configuration. It covers 5 different geometries: cylinder, NACA6512 airfoil[10] [13], twisted hex vase[11], teapot[12] and OpenFOAM's motorbike[13], see Fig. 2. For each of the geometries, we create 10 cases covering a range of positions, sizes, orientations and settings, giving a total of 50 cases. Each case is run with 9 possible subdomain allocations for 9 different core counts on ARCHER2, namely: 30, 42, 64, 77, 112, 128, 143, 198 and 256 cores. The final dataset contains a total of 4050 samples. Any samples for which SHM reported errors in the mesh (e.g. face errors) are removed prior to training and testing to minimize the effect of those errors on the results. For each of the 50 cases, *Optimum n* is determined per core count and is the subdomain allocation that provided the best execution time. All the runs of a case on a given core count were assigned the same *Optimum n* as their class.

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[9] https://www.open3d.org/

[10] https://github.com/MiroslavKabat/pythonNacaProfileGeneratorSTL

[11] https://www.thingiverse.com/thing:72040

[12] https://www.thingiverse.com/thing:821

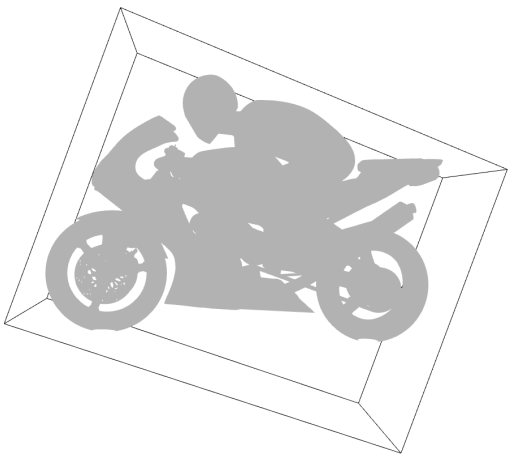


Figure 2: Motorbike geometry[13] with orientated bounding box.

2.3. Model training

To maximize the accuracy of the *scikit-learn RandomForestClassifier* in determining the *Optimum n*, a *RandomizedSearchCV* tool employing the *RepeatedStratifiedKFold* cross-validation method is used to determine the optimal hyperparameters. The random forest is built with 92 decision trees with a maximum depth of 7. The cross-entropy loss criterion is used to determine the splits in the tree and each tree is trained on a maximum of 95 samples. The configuration of the cross-validation and number of features to be selected using the *SelectKBest* method is further determined using hyperparameter sweeps with the *Weights & Biases* tool[14]. The model is trained on 67% of the post-selection data (the remainder is used for testing), with the split percentage also determined using hyperparameter sweeps. The top selected features and their *importances*, i.e., their contribution to the determination of the classification, are provided in Tab. 2. A feature's importance is a measure of its significance in decreasing the impurity criterion (the cross-entropy loss in this model) for the prediction, averaged over all the trees in the random forest[8,15]. The "number of cores" feature has the greatest impact on the classification; this is expected because that feature is implicitly linked to the subdomain allocation. The remainder of the features, which relate to the mesh refinement, load balancing and geometry characteristics, have a lower but still notable influence.

3. Evaluation and future work

The model is evaluated using an accuracy score which determines the percentage of the dataset that the *Optimum n* was classified correctly for. The accuracy achieved for our dataset is 76%, with a precision score of 71% and an F_1 score of 64%. This is a promising outcome, given the limited size of the data we trained our proof-of-concept model on.

[13] https://develop.openfoam.com/Development/openfoam/-/tree/master/tutorials/incompressible/simpleFoam/motorBike

[14] https://wandb.ai/site

[15] https://scikit-learn.org/stable/modules/ensemble.html

Feature	Description	Importance [%]
cores	# of processors	56.30
(xSD, ySD, zSD)	# of subdomains in each direction/ $\sqrt[3]{\text{cores}}$	(1.25, 1.20, 1.34)
(xVox, yVox, zVox)	Max voxels in the geom. in each direction	(1.80, 2.10, 3.09)
(xPos, yPos, zPos)	Geom. centre/boundary mesh center	(3.20, 1.98, 2.56)
maxLoadUnbalance	Max allowed load unbalance	1.87
maxLocalCells	Max # of cells per processor	1.72
(xCe, yCe, zCe)	# of cells in the initial mesh	(1.63, 2.47, 1.28)
(xLen, yLen, zLen)	Geom. (bounding box) lengths/boundary mesh lengths	(0.00, 2.53, 3.19)
(yaw, pitch, roll)	Geom. (bounding box) orientation as Tait-Bryan angles	(0.00, 2.75, 3.81)
surfaceRefLevel	Min & max surface refinement levels	(1.31, 1.22)
featureEdgeRefLevel	Feature edge refinement level	1.40

Table 2
List of features in the dataset, and their rates of importance in the model.

The accuracy is expected to improve with a larger dataset, and with a greater variety of geometries and case setups. Aside from increasing the size of the dataset, additional features can improve the accuracy of the classification. This includes settings for SHM's *snapping* and *layer addition* phases, which also impact the runtime of the mesh generation. Further improvements to overcome current limitations include a scheme for generalizing the subdomain allocation to decouple the classification from the number of cores, expanding the performance data to include multiple HPC systems, and accounting for the impact of the decomposition on the quality of the final mesh.

In this work, we have focused on selecting optimal settings for the hierarchical decomposition method. Future work could extend this approach to optimize the rate of load balancing by tuning the load imbalance parameters, with potential application across all decomposition methods used by SHM. Additionally, although we have not attempted to interpret the random forest classifier beyond reporting the feature importances, further studies to do so may yield insight into the problem to guide novel optimization strategies for SHM.

Acknowledgements

The authors thank the BWT Alpine F1 Team for providing funding for this project. This work used the ARCHER2 UK National Supercomputing Service[5].

References

[1] F. Moukalled, L. Mangani, M. Darwish, The Finite Volume Method in Computational Fluid Dynamics, Vol. 113 of Fluid Mechanics and Its Applications, Springer International Publishing, 2016. doi:10.1007/978-3-319-16874-6.

[2] J. P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. J. Mavriplis, CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences, Tech. rep., NASA, Langley Research Center (2014).
URL https://ntrs.nasa.gov/citations/20140003093

[3] J. R. Chawner, J. Dannenhoffer, N. J. Taylor, Geometry, Mesh Generation, and the CFD 2030 Vision, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2016. doi:10.2514/6.2016-3485.

[4] G. Karypis, V. Kumar, A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392. doi:10.1137/S1064827595287997.

[5] G. Karypis, V. Kumar, METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Tech. rep., Computer Science & Engineering (CS&E) Technical Reports, 97-061 (1997).
URL https://conservancy.umn.edu/items/2f610239-590c-45c0-bcd6-321036aaad56

[6] Francois Pellegrini, Scotch and PT-Scotch Graph Partitioning Software: An Overview, in: Combinatorial Scientific Computing, Chapman and Hall/CRC. doi:10.1201/b11644.

[7] C. Chevalier, F. Pellegrini, PT-Scotch: A tool for efficient parallel graph ordering, Parallel Computing 34 (6-8) (2008) 318–331. doi:10.1016/j.parco.2007.12.001.

[8] P. Sanders, C. Schulz, Think Locally, Act Globally: Highly Balanced Graph Partitioning, Springer Berlin / Heidelberg, 2013, pp. 164–175. doi:10.1007/978-3-642-38527-8_16.

[9] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2024.
URL https://proceedings.neurips.cc/paper_files/paper/2

022/file/0378c7692da36807bdec87ab043cdadc-Paper-Datasets_and_Benchmarks.pdf

[10] M. Haddouchi, A. Berrado, A survey of methods and tools used for interpreting Random Forest, in: 2019 1st International Conference on Smart Systems and Data Science (ICSSD), IEEE, 2019, pp. 1–6. doi:10.1109/ICSSD47982.2019.9002770.

[11] M. Aria, C. Cuccurullo, A. Gnasso, A comparison among interpretative proposals for Random Forests, Machine Learning with Applications 6 (2021) 100094. doi:10.1016/j.mlwa.2021.100094.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (85) (2012) 2825–2830. arXiv:1201.0490.

[13] Frank Walter Anderson, Orders of Magnitude: A History of NACA and NASA, 1915-1980, Vol. 4403, National Aeronautics and Space Administration, Scientific and Technical Information Branch.
URL https://ntrs.nasa.gov/api/citations/19760019059/downloads/19760019059.pdf

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Investigating the Effects of Spanwise Transversal Traveling Waves on a Turbulent Compressible Flat Plate Flow With the Aid of a Deep Autoencoder Network

Xiao Shao[a,*], Hilmi Oguzhan Ayan[a], Fabian Hübenthal[a], Mario Rüttgers[b], Andreas Lintermann[b] and Wolfgang Schröder[a]

[a]RWTH Aachen University, Institute of Aerodynamics, Wüllnerstraße 5a, 52062 Aachen, Germany
[b]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52428 Jülich, Germany

ARTICLE INFO†

Keywords:
Drag Reduction;
Autoencoder;
Machine Learning;
Traveling Waves;
Deep Learning

ABSTRACT

The impact of spanwise traveling transversal surface waves on drag reduction in turbulent compressible flat plate flow is explored. The findings indicate that when the traveling phase speed approaches the freestream velocity at $M = 0.7$, a shock wave is induced in the spanwise direction. This shock wave effectively breaks down streamwise vortices into smaller scales, which significantly enhances drag reduction. The spanwise shock wave is a large-scale quasi-periodic phenomenon. To understand its impact on the multi-scale nature of turbulent flows, a nonlinear mode decomposing deep convolutional autoencoder is employed. The results show that the autoencoder reconstructs the flow field more accurately compared to Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD). Additionally, it effectively separates the large-scale spanwise shock wave and small-scale turbulent structures, which achieves a clearer distinction than POD and DMD.

1. Introduction

In civil aviation, skin friction contributes significantly to the total aerodynamic drag, accounting for up to 50% [1], which leads to higher fuel consumption. Therefore, reducing skin friction is paramount for environmental reasons. Drag reduction techniques can be classified into two main groups: active, e.g., spanwise wall oscillations [2] or spanwise traveling waves [3], and passive methods, e.g., riblets [4]. While the former requires an introduction of energy input, the latter doesn't depend on external energy. The basic idea of spanwise traveling transversal surface waves is to introduce a secondary flow field by introducing a wavy motion to the surface, resulting in a drag reduction.

Combining the fast-emerging field of machine learning (ML) and fluid dynamics shows great potential to enhance the understanding of fluid dynamics, e.g., by predicting the resulting flow under different inlet conditions [5]. One common order reduction method is Proper Orthogonal Decomposition (POD) introduced to the fluid-dynamics community by Lumley in 1967 [6], which breaks the flow into different modes for a deeper analysis. Although POD techniques allow a reasonably good analysis of different modes in flow fields, its ability to handle non-linear relations, which are crucial aspects of turbulent flows, is limited [7]. Another relatively novel method for analyzing nonlinear systems is Dynamic Mode Decomposition (DMD) [8], which can be interpreted as an extension to POD with an additional consideration for the temporal aspects. However, DMD is also a linear approximation, presenting challenges when analyzing heavily nonlinear systems.

Artificial neural networks can overcome this obstacle due to their ability to capture non-linearity. In [9], the so-called Mode Decomposing Conventional Neural Network Autoencoder (MD-CNN-AE) is employed for mode decomposition. Although a conventional autoencoder uses only one mode, increasing the number of decoder branches is proven to improve learning and the capability to observe more complex flow structures. The superiority of MD-CNN-AE over POD is further shown in [10], where flow fields around square cylinders under varying geometric conditions are analyzed.

The current work extends the previous studies to the turbulent compressible flat plate flow, which has a high amount of non-linearity. That is, the effects of the surface

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02462 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
x.shao@aia.rwth-aachen.de (X. Shao);
oguzhan.ayan@rwth-aachen.de (H.O. Ayan);
f.huebenthal@aia.rwth-aachen.de (F. Hübenthal);
m.ruettgers@fz-juelich.de (M. Rüttgers); a.lintermann@fz-juelich.de (A. Lintermann); office@aia.rwth-aachen.de (W. Schröder)
ORCID(s): - (X. Shao); 0009-0007-5452-2786 (H.O. Ayan); 0009-0000-7159-8220 (F. Hübenthal); 0000-0003-3917-8407 (M. Rüttgers); 0000-0003-3321-6599 (A. Lintermann); 0000-0002-3472-1813 (W. Schröder)

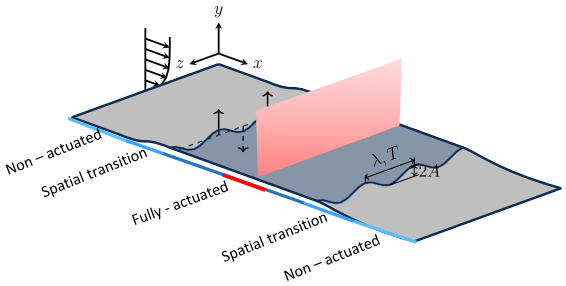


Figure 1: Schematic of the spanwise traveling transversal surface waves setup. The function $y|_{\text{wall}}(x,t) = g(x)A\cos\left(\frac{2\pi}{\lambda}(z-ct)\right)$ defines the wall-normal motion. The quantity A is the wave amplitude, λ is the wavelength, T is the period, c ($= \lambda/T$) is the phase speed, and $g(x)$ are step functions to manage a gradual spatial implementation and decay of the actuation in the x direction.

actuation on the flow field are examined in detail including a comparison between POD, DMD, and the MD-CNN-AE.

2. Methods

Turbulence scale-resolving numerical simulations for actively controlled turbulent boundary layer flow are conducted by the finite volume (FV) module of the in-house multiphysics flow solver m-AIA (multiphysics Aerodynamics Institute Aachen)[1], formerly known as Zonal Flow Solver (ZFS) [11]. An advanced numerical methodology is utilized to precisely simulate the flow field over moving boundaries, incorporating high-order discretization techniques and effective shock-capturing schemes. More details on the numerical method are given in [12].

The flow domain is defined by the Cartesian coordinates x, y, and z representing the streamwise, wall-normal, and spanwise directions, illustrated in Fig. 1. The simulations are done for a momentum thickness based Reynolds number $\mathrm{Re}_\theta = \frac{u_\infty \theta}{\nu} = 1{,}000$, with a momentum thickness of $\theta = 1$ at location $x_1/\theta = 165$ and the freestream velocity u_∞, and $M = 0.7$. The domain size is defined by $L_x \times L_y \times L_z = 361\theta \times 101\theta \times 64.94\theta$. The actuation parameters are specified by wavelength λ^+, time period T^+, and amplitude A^+ in inner scaling based on the friction velocity u_τ and kinematic viscosity ν. The actuated parameter values are $\lambda^+ = 3{,}000$, $T^+ = 88$, and $A^+ = 74$.

As depicted in Fig. 2, the autoencoder follows a similar structure of [9], where the encoder is followed by two mode-decomposing decoder branches. The activation functions are selected to be linear and tanh or ReLU for the convolutional layers. The activation function is used in the intermediate convolutional layers to handle non-linearities, whereas the linear activation function is used at the final convolutional layer. The yz planes at the center of the domain ($x/\theta = 180$), highlighted by the red rectangle in Fig. 1, are used for training. In total, 9230 snapshots are used and the model is trained for 2,000 epochs with the Adaptive Moment Estimation (Adam) [13] optimizer and a Mean Squared Error (MSE) loss function. The snapshots are randomly divided into training and validation sets and PyTorch's [14] distributed data parallel is utilized for distributed training on multiple GPUs. The parameters under investigation are the three-dimensional velocity components (u, v, w) and the pressure (p).

[1] https://doi.org/10.5281/zenodo.13350586

3. Results

The vortex topology based on the λ_2 criterion and the pressure contour in the yz plane is shown in Fig. 3. The isosurfaces of the vortex structures are generated by $\lambda_2 = -0.02$, colored by the relative Mach number defined by $M_{\text{rel}} = \sqrt{u^2 + v^2 + (c-w)^2}/\sqrt{\gamma p/\rho}$, where γ is the ratio of specific heats, and ρ is the density. At $c = 1.028$ relative to the stagnation sound speed a shock wave shown in Fig. 3(b) develops. The shock wave travels as the surface wave in the positive z direction. Within the shock wavefront, the vortices experience a reduction in number and size, influenced by the passing wave.

The reconstruction results for the autoencoder in comparison to the DMD and the POD with the 16 modes/16 decoders configuration for variables u, v, w, p are depicted in Fig. 4. The results indicate the high capability of the autoencoder relative to other methods, specifically in the three areas, namely in reconstructing the shock, the flow in the vicinity of the shock, and the turbulent structures. The inability of POD to regenerate the turbulent structures is an expected outcome as it heavily relies on linear spatial

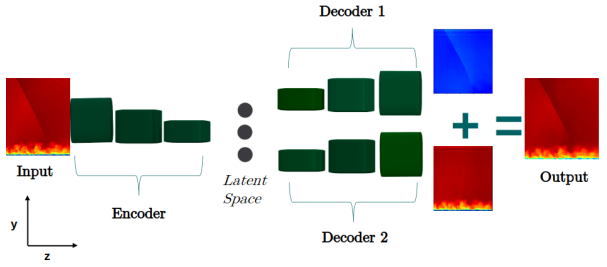


Figure 2: General concept the MD-CNN-AE for predicting two modes.

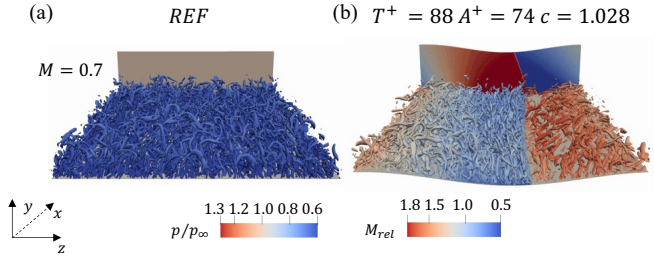


Figure 3: Instantaneous flow field for the non-actuated reference case (a), and the actuated case (b). Each illustration contains an yz plane pressure contour at $x/\theta = 180$ and a top view of the vortex topology in the streamwise direction and $y^+ < 60$.

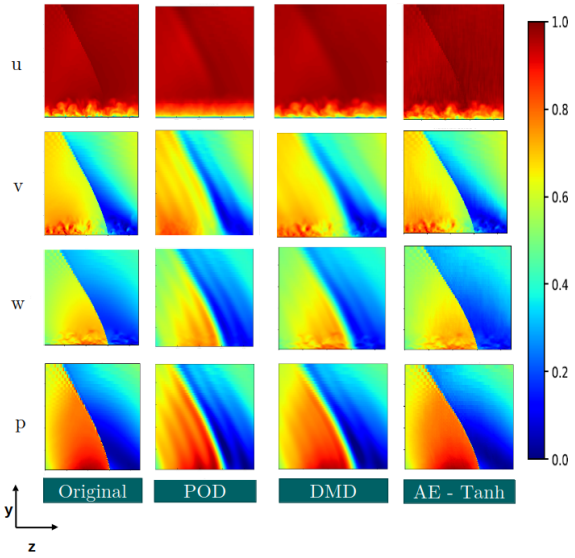


Figure 4: Original and reconstructed variables u, v, w, p for POD, DMD, and Autoencoder. The values are normalized by $(a_i - a_{min})/(a_{max} - a_{min})$, where a represents the variables under investigation.

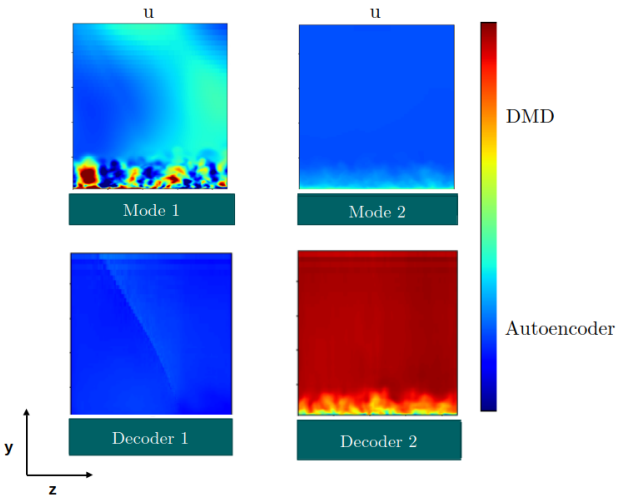


Figure 5: Decoder and the mode outputs of the channel u for the autoencoder and DMD for a 2 mode/decoder configuration. The values are normalized by $(a_i - a_{min})/(a_{max} - a_{min})$, where a represents the channels under investigation.

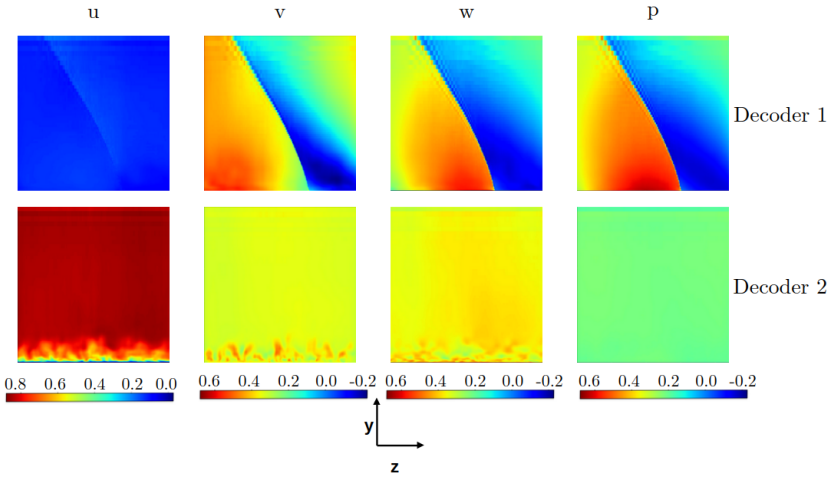


Figure 6: Decoder outputs of an autoencoder configuration with 2 decoders for all variables u, v, w, p. The values are normalized by $(a_i - a_{min})/(a_{max} - a_{min})$, where a represents the variable under investigation.

patterns, making it an unsuitable method for an in-depth analysis of turbulent flows. In contrast, DMD is able to reconstruct small-scale structures, which can be linked to its ability to consider the temporal dynamics of the system. However, DMD also fails to reconstruct the shock accurately as the shock wave is blurry, and the flow around the shock is also not well presented.

The individual outputs of the 2 decoders for the autoencoder and 2 modes for the DMD for variable u are shown in Fig. 5. It can be concluded that the autoencoder separates the shock and the turbulence in the flow, where the shock is

completely reconstructed. On the contrary, the DMD is not able to do this. The shock is vaguely represented and quite blurry in mode 1 in Fig. 5. The POD results here were not included as it was deemed an unfitting method, where the turbulence was not captured in detail. The ability to separate the large-scale shock waves and multi-scale turbulence structures is not unique to variable u. The same separation pattern exhibits in v, w, p.

The decoder results for all variables are shown in Fig. 6. In each velocity channel, decoder 1 captures the large-scale structures associated with the shock, whereas decoder 2 captures the multi-scale turbulence. On the other hand, only the large-scale structures are captured in the pressure channel, whereas decoder 2 does not output anything. This is affiliated with the nature of the pressure characteristics, which do not possess fluctuating turbulence.

In conclusion, the mode decomposing autoencoder can capture the shock and the high-frequency turbulent fluctuations accurately compared to the conventional methods, i.e., Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD). Moreover, the autoencoder can separate large-scale shock waves from multi-scale turbulence structures, which results in a continuous turbulence flow field. Clear decomposition makes it possible to examine the alter of turbulence structures due to the shock wave.

References

[1] P. Ricco, M. Skote, M. A. Leschziner, A review of turbulent skin-friction drag reduction by near-wall transverse forcing, Progress in Aerospace Sciences 123 (2021) 100713. doi:10.1016/j.paerosci.2021.100713.

[2] W. J. Jung, N. Mangiavacchi, R. Akhavan, Suppression of turbulence in wall-bounded flows by high-frequency spanwise oscillations, Physics of Fluids A: Fluid Dynamics 4 (8) (1992) 1605–1607. doi:10.1063/1.858381.

[3] M. Itoh, S. Tamano, K. Yokota, S. Taniguchi, Drag reduction in a turbulent boundary layer on a flexible sheet undergoing a spanwise traveling wave motion, Journal of Turbulence 7 (27) (2006) 27. doi:10.1080/14685240600647064.

[4] M. Walsh, Turbulent boundary layer drag reduction using riblets, in: 20th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, Reston, Virigina, 1982. doi:10.2514/6.1982-169.

[5] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166. doi:10.1017/jfm.2016.615.

[6] J. L. Lumley, The structure of inhomogeneous turbulent flows, Atmospheric Turbulence and Radio Wave Propagation (1967) 166–178.
URL https://cir.nii.ac.jp/crid/1574231874542771712

[7] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2020) 477–508. doi:10.1146/annurev-fluid-010719-060214.

[8] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics 656 (2010) 5–28. doi:10.1017/S0022112010001217.

[9] T. Murata, K. Fukami, K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, Journal of Fluid Mechanics 882 (2020). doi:10.1017/jfm.2019.822.

[10] A. Higashida, K. Ando, M. Rüttgers, A. Lintermann, M. Tsubokura, Robustness evaluation of large-scale machine learning-based reduced order models for reproducing flow fields, Future Generation Computer Systems 159 (2024) 243–254. doi:10.1016/j.future.2024.05.005.

[11] A. Lintermann, M. Meinke, W. Schröder, Zonal Flow Solver (ZFS): a highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34 (7-8) (2020) 458–485. doi:10.1080/10618562.2020.1742328.

[12] M. Albers, P. S. Meysonnat, D. Fernex, R. Semaan, B. R. Noack, W. Schröder, Drag reduction and energy saving by spanwise traveling transversal surface waves for flat plate flow, Flow, Turbulence and Combustion 105 (1) (2020) 125–157. doi:10.1007/s10494-020-00110-8.

[13] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). arXiv:1412.6980.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Predicting NOx Emissions From Porous Media Burners Using Physics-Informed Graph Neural Networks

Rishabh Puri[a,*], Oliver T. Stein[a] and Thorsten Zirwes[b]

[a]Karlsruhe Institute of Technology, Engler-Bunte Institute, Simulation of Reacting Thermo-Fluid Systems, Engler-Bunte Ring 7, 76131 Karlsruhe, Germany
[b]University of Stuttgart, Institute for Reactive Flows, Pfaffenwaldring 31, 70569 Stuttgart, Germany

ARTICLE INFO[†]

Keywords:
Ammonia Combustion;
Direct Numerical Simulation;
Graph Convolution Neural Networks

ABSTRACT

The combustion of ammonia is accompanied by the release of NO_x, which refers to both NO and NO_2, i.e., the nitrogen oxides contributing to air pollution. A potential technology for complete combustion of ammonia and reduced NO_x emissions is combustion in porous media. Multiple test cases are defined using 2D burner configurations that are investigated computationally by means of Direct Numerical Simulations (DNS). The main objective of this work is to study the exhaust gas products for a wide range of burner configurations. However, simulating a large number of burner setups using DNS is time-consuming and costly. To reduce the number of DNS computations, a physics-informed deep-learning model based on Graph Convolution Neural Networks (GCNNs) is employed. In a first step, the effectiveness of data-driven GCNNs is validated for reactive flows in porous media and preliminary data is shown here. Subsequently, GCNNs augmented with constraints from combustion chemistry are implemented to train on sparse data and to predict the combustion characteristics of the porous media burner with high efficiency.

1. Introduction

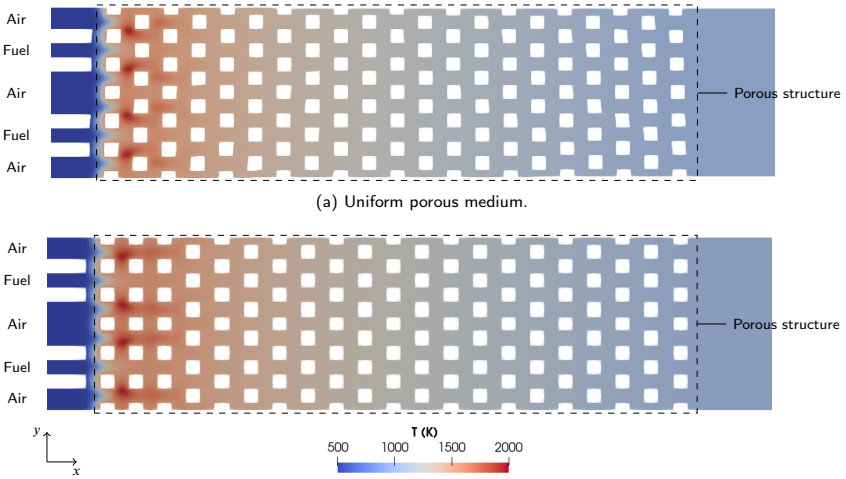
Ammonia offers a promising possibility as a carbon-free energy carrier. Although it can easily be transported using the existing infrastructure and stored for future energy needs, combustion of ammonia poses considerable challenges. The three major challenges are its low burning velocity compared to hydrocarbons resulting in poor flame stability, its high levels of nitrogen oxide formation and its high toxicity even at trace levels. To tackle these challenges, combustion in porous media can be utilized. Porous media combustion (PMC) has long been investigated for conventional fuels [1, 2]. Conduction and radiation heat transfer between the porous medium and the reaction zone can help in stabilizing the flame for fuels with poor combustion characteristics [3, 4]. The porous medium is provided in the form of a ceramic structure, which forms a rigid matrix. Within the porous cavities of this rigid matrix the fuel and oxidizer flow, mix and react. Direct numerical simulations of combustion in porous media can aid in designing and optimizing porous media burners. Direct pore-level simulations (DPLS) require a coupling of such fully-resolved direct flow simulations with thermal simulations of the solid matrix. Such simulations are cost-intensive and performing DPLS for a wide range of burner configurations is computationally intractable. Data-driven graph neural networks have shown potential in predicting flow fields while training on data from numerical simulations [5, 6]. To study the effectiveness of such deep-learning based models in predicting NO_x emissions, the GCNN approach is employed to investigate porous media combustion. The performance of the proposed GCNN is evaluated against DNS results from the EBIdnsFoam solver [7] and results from a simplified 1D model.

2. Method

Two-dimensional burner configurations with porous solid structures are considered as shown in Fig. 1. The properties of the porous structure such as porosity and pore size as well as the initial/boundary conditions of velocity and temperature are varied for different burner configurations. Direct numerical simulations are performed for 10% of the total generated configurations. For the remainder of the configurations, training data for temperature is generated from 1D volume-averaged simulations (1D-VAS) of PMC [8]. Cantera is used to calculate the species reaction rates and material properties appearing in the governing equations. Thus, a combination of DNS and 1D-VAS results is used for the training of the GCNN model, which are considered via two data loss functions L_{data} from DNS and 1D-VAS. It should be noted that all GCNN outputs can be considered

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02463 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
rishabh.puri@kit.edu (R. Puri); oliver.t.stein@kit.edu (O.T. Stein); thorsten.zirwes@irst.uni-stuttgart.de (T. Zirwes)
ORCID(s): 0009-0000-5691-1450 (R. Puri); 0000-0002-7954-0506 (O.T. Stein); 0000-0002-3563-1422 (T. Zirwes)



(a) Uniform porous medium.

(b) Regular porous medium with first three layers arranged in parallel.

Figure 1: Two-dimensional burner with solid structures representing the porous material.

in the L_{data} loss function when DNS data is available, while only temperature predictions are considered in L_{data} when using data from 1D-VAS. Physics-informed models can provide good accuracy in predicting solutions to boundary value problems. In addition to full DNS data, the inclusion of sparse temperature data from 1D-VAS can help to further improve the prediction quality. Due to the laminar flow in the porous media, steady state solutions for all configurations are obtained.

The proposed model architecture is shown in Fig. 2. A physics-informed GCNN is defined in which the governing equations for reacting flows are enforced as physical constraints. The input for the model is defined by the grid coordinates and the boundary conditions for velocity and temperature. The output of the model are the velocity, density, temperature, mixture fraction and NO_x mass fraction fields inside the burner. The NO_x emissions at the outlet of the burner are extracted from the predicted NO_x mass fraction fields. Assuming constant pressure, low velocity, equal species heat capacities, and no viscous heating, the physical constraints for the GCNN are defined using the simplified equations for continuity, momentum, energy, mixture fraction, and species (NO_x) mass conservation for steady laminar reactive flow [9]. The predicted value of the H_2O mass fraction serves as a progress variable and is used with Cantera to compute the source terms and material properties in the governing equations. The viscosity is obtained from Sutherland's law [10]. The physical constraints imposed by the governing equations are implemented as a loss function $L_{physical}$ for the GCNN

$$\begin{aligned} L_{physical} \quad = \quad & L_{continuity}(\rho, \vec{U}) + L_{momentum}(\rho, \vec{U}) + \\ & L_{energy}(\rho, \vec{U}, T) + L_{mix.fraction}(\rho, \vec{U}, f) + \\ & L_{NOx}(\vec{U}, Y_{NOx}), \end{aligned} \tag{1}$$

where ρ is the mixture density, $\vec{U}$ the velocity field, T the temperature, f the mixture fraction and Y_{NOx} is the NO_x mass fraction. Finally, the physical loss $L_{physical}$ is combined with the data loss L_{data} from both DNS and 1D-VAS, such that the total loss function for the model is defined as

$$L_{GCNN} = L_{physical} + L_{data}. \tag{2}$$

The automatic differentiation functionality of the `PyTorch` library is used for implementing the partial and ordinary differential equations (PDE/ODE) in the loss function. Thus, the proposed GCNN model uses the grid data from OpenFOAM as well as the boundary conditions as input and the species concentration of the NO_x species in the exhaust gas can be predicted from the output of the model. As the training progresses, the PDE/ODE residuals in the loss functions decrease, thus improving the predictive capability of the GCNN.

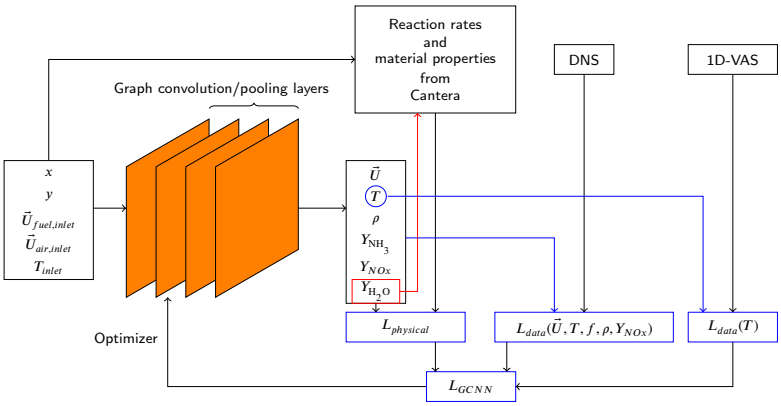


Figure 2: Schematic of the GCNN model.

ϕ	Fuel	Burner	$T_{max,PIM}$ (K)	Y_{NO_x} (DNS)	Y_{NO_x} (GCNN)	error (%)
0.9	NH_3	P1	1767	3.164×10^{-4}	2.307×10^{-4}	27.1
0.95	$0.9NH_3$+$0.1H_2$	P2	1591	2.207×10^{-5}	3.378×10^{-5}	67.8
0.95	NH_3	P1	1591	1.078×10^{-4}	5.393×10^{-5}	49.9

Table 1
Comparison of prediction results of data-driven GCNN with the interpolated DNS data. The burner configuration with uniform porous structures (Fig. 1a) is defined as P1 and the configuration with first three parallel layers of porous structures (Fig. 1b) is defined as P2. The fuel is defined using the mole fraction of fuel species.

3. Results

Here, we present preliminary results from the data-driven GCNN model which is trained with DNS data from 12 cases. The performance of the GCNN model is measured by the error in percent of the predicted values of the NO_x emissions from the exhaust of the burner. The DNS data is interpolated onto uniform grids and the adjacency list is generated for this uniform grid. Each uniform grid is set with an x-spacing of $0.5d$ and a y-spacing of $0.24d$, with d being the characteristic length of each solid strut in the porous medium. All variables except for the velocity are scaled to a range from zero to one. For the data-driven model, a 70:30 % split is used to distribute the data into training and testing datasets. The testing dataset is later also used for validation of the trained model. The hyperparameters for the model are tuned using the random search method and the model is trained on the the GPU partition of the JURECA-DC cluster [11] installed at the Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich. The model is comprised of six `GraphConv` convolution layers and two `EdgePooling` pooling layers coupled with two un-pooling layers. The intermediate dimensions in consecutive graph convolution layers are $(7, 256, 512, 1028, 1028, 512, 256, 7)$ and the hyperbolic tangent activation function is used. The ADAM optimizer is used with a learning rate of 0.0001 and the model is trained for 3,000 epochs.

As observed in Tab. 1, the deviation in prediction from the data-driven model from the DNS results is higher, as the NO_x mass fraction in the exhaust decreases. This deviation is likely attributed to the limited DNS data available for training. Further improvements are expected from the physics-informed model in future work.

4. Conclusion

A physics-informed graph convolution neural network is proposed to predict the NO_x emissions from the combustion of ammonia in porous media burners. Preliminary results from a data-driven GCNN are shown in the present work. The data-driven GCNN model is capable of predicting the order of magnitude of the NO_x emissions, with increasingly higher errors for lower absolute values of NO_x mass fraction. Based on the learning outcomes from the current configuration future work will enhance the data-driven part of the model and further develop the physics-informed model.

References

[1] J. L. Ellzey, E. L. Belmont, C. H. Smith, Heat recirculating reactors: Fundamental research and applications, Progress in Energy and Combustion Science 72 (2019) 32–58. doi:10.1016/j.pecs.2018.12.001.

[2] D. Trimis, F. Durst, Combustion in a Porous Medium-Advances and Applications, Combustion Science and Technology 121 (1-6) (1996) 153–168. doi:10.1080/00102209608935592.

[3] H. Nozari, G. Karaca, O. Tuncer, A. Karabeyoglu, Porous medium based burner for efficient and clean combustion of ammonia–hydrogen–air systems, International Journal of Hydrogen Energy 42 (21) (2017) 14775–14785. doi:10.1016/j.ijhydene.2017.03.234.

[4] G. Vignat, B. Akoush, E. R. Toro, E. Boigné, M. Ihme, Combustion of lean ammonia-hydrogen fuel blends in a porous media burner, Proceedings of the Combustion Institute 39 (4) (2023) 4195–4204. doi:10.1016/j.proci.2022.07.054.

[5] J. Chen, E. Hachem, J. Viquerat, Graph neural networks for laminar flow prediction around random two-dimensional shapes, Physics of Fluids 33 (12) (2021) 123607. doi:10.1063/5.0064108.

[6] J.-Z. Peng, N. Aubry, Y.-B. Li, M. Mei, Z.-H. Chen, W.-T. Wu, Physics-informed graph convolutional neural network for modeling geometry-adaptive steady-state natural convection, International Journal of Heat and Mass Transfer 216 (2023) 124593. doi:10.1016/j.ijheatmasstransfer.2023.124593.

[7] T. Zirwes, M. Sontheimer, F. Zhang, A. Abdelsamie, F. E. H. Pérez, O. T. Stein, H. G. Im, A. Kronenburg, H. Bockhorn, Assessment of Numerical Accuracy and Parallel Performance of OpenFOAM and its Reacting Flow Extension EBIdnsFoam, Flow, Turbulence and Combustion 111 (2) (2023) 567–602. doi:10.1007/s10494-023-00449-8.

[8] T. Zirwes, G. Vignat, E. R. Toro, E. Boigné, K. Younes, D. Trimis, M. Ihme, Improving volume-averaged simulations of matrix-stabilized combustion through direct X-ray μCT characterization: Application to NH_3/H_2-air combustion, Combustion and Flame 257 (2023) 113020. doi:10.1016/j.combustflame.2023.113020.

[9] T. Poinsot, D. Veynante, Theoretical and numerical combustion, 2nd Edition, Edwards, Philadelphia, 2005.

[10] W. Sutherland, The viscosity of gases and molecular force, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 36 (223) (1893) 507–531. doi:10.1080/14786449308620508.

[11] D. Krause, P. Thörnig, JURECA: Modular supercomputer at Jülich Supercomputing Centre, Journal of large-scale research facilities JLSRF 4 (2018) A132. doi:10.17815/jlsrf-4-121-1.

Mini-Symposium 4:

Modernizing CFD: Exploring CI/CD for Improved Software Development Life Cycle

Organizers: Damien Dosimont, Dennis Hoppe, Harald Köstler, and Guillaume Houzeaux

This Mini-symposium is co-organized by the European Centers of Excellence in Exascale Computing CEEC, EXCELLERAT P2, EOCOE3, and HIDALGO 2 supported by the coordination activity CASTIEL 2.

Continuous Integration/Continuous Deployment (CI/CD) has evolved into a pivotal methodology, significantly enhancing software development and deployment processes. In Computational Fluid Dynamics (CFD) applications, CI/CD plays a crucial role in guaranteeing simulation applications' stability, reproducibility, portability, and performance. This mini-symposium aims to delve into innovative approaches across various stages of the CI/CD pipeline, making substantial contributions to the scientific aspects of CFD applications. We welcome contributions in the following areas:

Builds:

- In-depth discussions on efficient build processes tailored for CFD applications.
- Achieving portability across various compilers, architectures, CPU/GPU configurations, programming standards, code generation, and managing multiple code versions through containerization.

Tests:

- Exploration of advanced testing methodologies designed explicitly for CFD simulations.
- Integration of automated testing frameworks seamlessly into the CI/CD pipeline.
- Addressing challenges associated with large-scale parallel testing.
- Discussion on the integral role of CI/CD in ensuring the accuracy and reliability of simulations.
- Collaboration strategies for testing within interdisciplinary teams.

Deploy:

- Investigation into methods ensuring seamless and reliable deployment of CFD applications.
- Formulation of strategies for versioning and deployment in diverse computing environments.
- Guaranteeing application scalability and adaptability.

Monitor:

- Implementation of monitoring infrastructures and code instrumentation for performance analysis.
- Continuous monitoring methodologies to detect anomalies and enhance overall efficiency.

General:

- Development of strategies for designing modular and maintainable CFD software.
- Presentation of case studies illustrating successful implementations of CI/CD in CFD research groups.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Adapting the Development of a CFD Application Following the CI-CD Life Cycle and a DevOps Approach

Damien Dosimont[a,*], Guillaume Houzeaux[a]

[a]*Barcelona Supercomputing Center, Computer Applications in Science and Engineering (CASE), Plaça d'Eusebi Güell, 1-3, Les Corts, 08034 Barcelona, Spain*

ARTICLE INFO[†]

Keywords:
Computational Fluid Dynamics;
Continuous Integration / Continuous Deployment;
High-Performance Computing;
Tools, Build, Test;
Release;
Monitoring;
Slurm

ABSTRACT

Continuous integration and continuous deployment (CI/CD) methodologies have gained traction in the computational fluid dynamics (CFD) community, offering streamlined code evolution and deployment. This article addresses the modernization challenge of Alya, a 19-year-old CFD simulation code, by tailoring a framework to ensure code stability, reproducibility, portability, and reliability. Emphasis lies on robust building, rigorous testing, and vigilant monitoring to uphold optimal performance levels and prevent degradation. Additionally, efforts are directed toward enhancing developer collaboration during the CI cycle. This abstract outlines our methodological approach and contributions since migrating to CI/CD practices.

1. Introduction

In this abstract, we outline our adaptation of CI/CD practices to Alya [1], a CFD simulation code with a 19-year legacy and robust High-Performance Computing (HPC) capabilities. We've tailored a framework to ensure stability, reproducibility, and reliability throughout Alya's development life cycle, emphasizing robust building, testing, and vigilant monitoring. Our approach prioritizes optimal performance and code integrity. Moreover, we are striving to improve developer collaboration throughout the CI cycle. This abstract summarizes our methodological approach and technical contributions post-migration to CI/CD practices.

2. Following the CI-CD life cycle

Alya's development process follows the standard CI/CD life cycle, involving phases like *planning*, *coding*, *building*, *testing*, *releasing*, *deploying*, *operating*, and *monitoring*. Continuous Integration (CI), from planning to testing, integrates new code changes iteratively, promoting collaboration and early issue detection. Conversely, Continuous Deployment (CD), from release to monitoring, deploys code changes sequentially after thorough testing, ensuring system stability and performance. This approach ensures a systematic and controlled roll-out of updates, minimizing disruptions and deployment risks. The workflow is summarized in Figure 1.

Our CI/CD process centers on a GitLab instance serving as the core of our workflow. During the *plan* phase, we utilize GitLab's issue functionality to outline specifications and facilitate collaboration, ensuring thorough documentation. In the *code* phase, developers leverage GitLab's merge request features, incorporating Git's commits and branches for seamless code review and version control, in line with the GitLab flow.

The *build* and *test* phases are crucial parts of the GitLab CI pipeline, triggered automatically by merge request commits. This pipeline, outlined in the repository's `.yaml` file, coordinates stages containing one or multiple jobs executed by GitLab runner services. These phases are merged into unified jobs, improving efficiency through shared *docker* and *staging* environments. Initially, tasks involve building and testing within docker environments, followed by compiling and executing unity and smoke tests to ensure regression absence. Additionally, new builds are promptly incorporated when required by code changes. The *build* phase is controlled by `CMake`, while `CTest` manages test execution.

Subsequently, the pipeline advances to the build and testing phases within staging environments, covering BSC HPC platforms MareNostrum IV, Power-CTE, and AMD-CTE[1]. To streamline this process, we've developed *squidient*[2] as a custom build and test manager, specifically adapted to Alya. This tool handles build configurations, interfaces with HPC machines, and utilizes `slurm` for scheduling tasks across platforms. After builds, the pipeline executes unity

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02464 and of the Proceedings 10.34734/FZJ-2025-02175.

[*]Corresponding author
damien.dosimont@bsc.es (D. Dosimont);
guillaume.houzeaux@bsc.es (G. Houzeaux)
ORCID(s): 0000-0002-6620-9873 (D. Dosimont); 0000-0002-2592-1426 (G. Houzeaux)

[1]https://www.bsc.es/marenostrum/marenostrum/technical-information
[2]https://alya.gitlab.bsc.es/alya/ci-cd/squidient

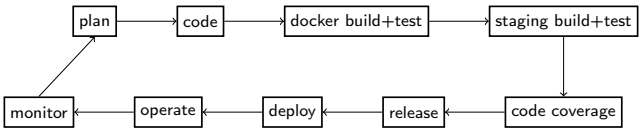


Figure 1: Simplified CI/CD workflow.

and smoke tests, followed by comprehensive regression testing. Post-evaluation, *squidient* compares results with reference files, considering tolerances set by test designers to ensure simulation output integrity and accuracy. Utilizing `Intel PGO` (Profile-Guided Optimization) provides valuable insights into code coverage post-*squidient* execution. This analysis helps evaluate the extent to which unity, smoke, and regression tests cover new contributions.

Following a DevOps approach, developers test their contributions. If failures occur in the *build-test* phases in either the *docker* or *staging* environment, we reset the cycle iteratively until issues are resolved. For code coverage, strict guidelines are enforced to ensure thorough testing and prevent regressions. New tests must cover changes, validating functionality and integrity. Additionally, overall code coverage should remain stable, except during specific circumstances like refactoring or cleaning, ensuring the testing suite evolves with the codebase, enhancing resilience and efficacy over time.

The completion of the CI cycle signifies the acceptance of the merge request, marking it as a *release* of the proposed changes into the main repository branch.

The subsequent deployment process involves provisioning *Alya* on MareNostrum IV, orchestrated by *squidient* and initiated by the GitLab CI pipeline of the main branch. As for during the build and test phases, *squidient* operates on a GitLab runner. This time, it compiles and installs *Alya*, and generates the corresponding `slurm` modules. The generic *json*configuration management of *squidient* facilitates the addition of new deployment platforms, provided they can be accessed via SSH and utilize `slurm` as the job manager.

During the *operate* phase, we gather feedback from developers and simulation users, creating a feedback loop for issue identification and improvement suggestions. We conduct benchmarks using the CI pipeline and *squidient*, covering aspects like I/O operations, kernel features, and solver performance. Utilizing the TALP library [2], we collect metrics such as load balancing and communication efficiency, which product is the parallel efficiency.

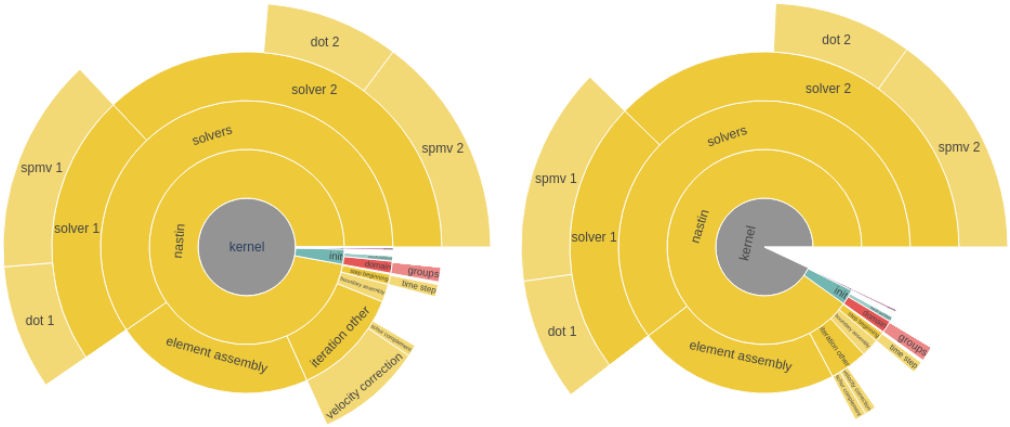
In the *monitoring* phase, our visualization tool, *rooster* [3], aids in presenting metrics and conducting analysis over time. Through *rooster*, we monitor performance indicators, detecting anomalies indicating underlying issues, which may stem from machine-related factors or programming errors. Conversely, *rooster* tracks instances of improvement, offering insights into optimization efforts and code enhancements.

3. Results

In the *plan* phase, since migrating to GitLab in 2020, we've handled a total of 2042 issues, resolving 1909 of them and leaving 133 still active. These issues led to 1796 merge requests, with 547 abandoned and 1221 successfully merged. On average, there are 40 active merge requests, with 9.2 commits daily over the past 2000 commits, contributed by 28 active authors. In the *build* phase, our focus is on application stability and portability. We evaluate 17 combinations in docker environments and execute 34 build combinations across three HPC machines (Intel, IBM, and AMD architectures), covering `Intel`, `GNU`, `NVHPC/PGI`, and `IBM` compilers and debug options, ensuring compliance with standard Fortran 2008 and validating external component integration. We perform 130 unity tests and smoke tests, along with up to 632 tests during regression testing for each build. Code coverage has increased significantly from 16% to 45% over four years. Regarding CI pipeline performance, the Docker *build+test* phase takes one to two hours. The GitLab runners are executed on two personal computers, each equipped with an Intel i7 processor with 16 cores and 32 GB of RAM. Due to the simultaneous builds, *Alya* is compiled using only one process, which likely accounts for the prolonged compilation time. The *staging* phase extends up to 375 minutes. The entire pipeline, consisting of 56 jobs, runs for 540 minutes, allowing approximately 2 merge requests to be merged per day. The performance metrics collection and visualization tool *rooster* effectively monitor Alya's performance. An example of the improvement in a cough simulation [4], one of the benchmarks used to monitor Alya, is visualized with *rooster* following the vectorization of the velocity correction (Figure 2).

4. Conclusion and future works

In summary, our paper demonstrates the successful implementation of CI/CD methodologies for Alya, a CFD simulation code with strong HPC capabilities. Prioritizing stability, reproducibility, and reliability, our framework has



(a) Before vectorization.

(b) After vectorization.

Figure 2: Sunbursts of the cough simulation showing the execution time improvement thanks to the vectorization of the velocity correction.

shown significant progress. Efficient issue resolution and improved collaboration in the planning phase have enhanced developer teamwork. The rise in code coverage and optimized collaboration underscore our approach's effectiveness. Tools like *rooster* aid in performance monitoring, ensuring Alya's ongoing enhancement and reliability in HPC-based CFD simulations.

In the future, we aim to streamline the CI pipeline, focusing on reducing merge request times. We're exploring branch-level workflows to optimize resource allocation and plan to integrate AI algorithms for smarter build and test selection. Additionally, we're working on collecting hardware counters to detect machine-related deviations during monitoring.

References

[1] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, M. Valero, Alya: Multiphysics engineering simulation toward exascale, Journal of Computational Science 14 (2016) 15–27. doi:10.1016/j.jocs.2015.12.007.

[2] V. López, G. Ramirez Miranda, M. Garcia-Gasulla, TALP: A Lightweight Tool to Unveil Parallel Efficiency of Large-scale Executions, in: Proceedings of the 2021 on Performance EngineeRing, Modelling, Analysis, and VisualizatiOn STrategy, 2021, pp. 3–10. doi:10.1145/3452412.3462753.

[3] D. Dosimont, G. Houzeaux, Monitoring the development of CFD applications on unstable HPC platforms, arXiv preprint arXiv:2401.08447 (2024). doi:10.48550/arXiv.2401.08447.

[4] H. Calmet, K. Inthavong, A. Both, A. Surapaneni, D. Mira, B. Egukitza, G. Houzeaux, Large eddy simulation of cough jet dynamics, droplet transport, and inhalability over a ten minute exposure, Physics of Fluids (Woodbury, N.Y.: 1994) 33 (12) (2021) 125122. doi:10.1063/5.0072148.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Verificarlo CI: Continuous Integration for Numerical Optimization and Debugging

Aurélien Delval[a,b,*], François Coppens[a], Eric Petit[c], Roman Iakymchuk[d] and Pablo de Oliveira Castro[a]

[a]Université Paris-Saclay, UVSQ, LI-PaRAD, 9 boulevard d'Alembert, bâtiment François Rabelais, 78280 Guyancourt, France
[b]SiPearl, 44 rue Jean Mermoz, 78600 Maisons-Laffitte, France
[c]Intel Corporation, 2200 Mission College Blvd., M/S RNB4-145, Santa Clara, 95054, CA, USA
[d]Umeå University and Uppsala University, 90187 Umeå, Sweden

ARTICLE INFO†

Keywords:
Continuous Integration / Continuous Deployment;
Verificarlo;
Numerical Accuracy;
High-Performance Computing

ABSTRACT

Floating-point accuracy is an important concern when developing numerical simulations or other compute-intensive codes. Tracking the introduction of numerical regression is often delayed until it provokes unexpected bug for the end-user. In this paper, we introduce Verificarlo CI, a continuous integration workflow for the numerical optimization and debugging of a code over the course of its development. We demonstrate applicability of Verificarlo CI on two test-case applications.

1. Introduction

Despite Floating-point (FP) accuracy being a known issue [1], modern tools for software development do not provide automated numerical accuracy regression tests. To fill this need, we propose Verificarlo CI (Continuous Integration). GitHub and GitLab are popular platforms for developing software, and both have features for CI. CI services are triggered on specific events, such as merging a pull request. Verificarlo CI is designed to be integrated with them, but it can also be used with custom workflows.

To facilitate its adoption, Verificarlo CI has been designed to be easy and fast to deploy, while still being flexible enough to be relevant for most applications. We provide the user with a simple API to insert FP probes in their tests, execute them with Verificarlo, setup CI Actions, and finally access and interpret the results.

Finally, we demonstrate Verificarlo CI on two use-cases: exploring reduced mixed-precision in the Nekbone CFD proxy application; tracking numerical bugs during the development of a the Quantum Monte Carlo Chemistry Kernel library (QMCkl).

2. Verificarlo CI for numerical correctness

Verificarlo [2] is a tool based on the LLVM compiler framework modifying at compilation each floating point operation with custom operators. After compilation, the program can be linked against various backends to explore FP related issues and optimizations. The latest version of Verificarlo fully supports OpenMP and MPI parallel programs.

Verificarlo computes the number of significant bits to evaluate the numerical accuracy of a computation. It captures the number of accurate bits in the FP mantissa against a chosen reference. Unfortunately, an exact reference value is not known beforehand for many complex programs or intermediate computations. To overcome this problem, Verificarlo uses Monte Carlo arithmetic (MCA) [3], a stochastic method that can simulate numerical errors and estimate the number of significant bits directly: the result of each FP operation is replaced by a perturbed computation modeling the losses of accuracy. From a set of MCA samples, it is possible to estimate the significant bits of a computation, $s_2 = -\log_2 |\sigma/\mu|$, where σ and μ are the sample's standard deviation and mean [4].

Verificarlo includes six backends, which are extensively documented in the user manual. The two most important backends are: the `MCA` backend, described previously, and the `VPREC` backend that emulates the effect of using mixed-precision in a program [5].

Verificarlo CI automates numerical accuracy tests by using a separate Git branch to store test results. Whenever users make modifications to the main branch, a remote runner carries out predefined tests and uploads the results

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02465 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
aurelien.delva@uvsq.fr (A. Delval); francois.coppens@uvsq.fr (F. Coppens); eric.petit@intel.com (E. Petit); riakymch@cs.umu.se (R. Iakymchuk); pablo.oliveira@uvsq.fr (P. de Oliveira Castro)
ORCID(s): 0000-0002-2707-6940 (A. Delval); 0000-0003-1695-352X (F. Coppens); - (E. Petit); 0000-0003-2414-700X (R. Iakymchuk); 0000-0001-9007-6145 (P. de Oliveira Castro)

to the CI branch. Users can view their results dynamically using a simple web server.

Verificarlo CI offers a command-line interface that helps configuring the CI pipeline on a given application and hooks it up to a GitLab or GitHub repository : it automatizes the initial setup by creating both the CI pipeline and the dedicated results branch. Users are then free to further customize their pipelines.

Developers use a dedicated C or Fortran API to include *probes* in their code. Each probe is associated with a test and a variable name. During testing, the probe measures the accuracy of a chosen variable. Optionally, an alert can be triggered if the relative or absolute error exceeds a user-set threshold. In the below C example, the probe is identified by the `"test"`/`"var"` couple, and an absolute error threshold is set to 0.01:

```
vfc_probe_check(probes, "test", "var", var, 0.01);
```

In order to be able to run the tests, Verificarlo CI requires a description of the tests and backends to run. It is specified in a JSON configuration file which supports complex test setups. The test results are exported to an HDF5 file, a hierarchical format commonly used in HPC applications. Test results are stored on the dedicated CI branch, allowing robust archival. The HDF5 files can optionally embed the raw test results. In the default CI workflow, this raw data is stored as a job artifact and accessible for a limited time, to enable user defined additional analysis. Finally, Verificarlo CI analyzes the data and generates dynamic reports organized into different views: temporal, cross-test, or cross-variable comparisons and accuracy violations.

3. Mixed-precision for Nekbone proxy application

Nek5000[1] is a high-order solver for Computational Fluid Dynamics (CFD) based on the Spectral Element Method (SEM) that solves the Navier-Stokes equation for incompressible flow. Nekbone is a proxy application for Nek5000 that focuses on important computational and scaling aspects of the entire solver. We used the `VPREC` backend to examine precision appetites in Nekbone using the polynomial degree of 10 and different number of elements. The results of tracking the residual of the Conjugate Gradient (CG) solver, see Fig. 1, suggest a possibility of using as little as 16 bits of mantissa (the beginning of the plateau) and still being able to converge, while the original version relies on FP64 (double precision) with 52 bits of mantissa. We verified this assumption with the `MCA` backend, confirming such a possibility for the precision reduction to single in the CG solver on CPUs.

[1] `https://nek5000.mcs.anl.gov/` and `https://github.com/Nek5000/Nekbone`

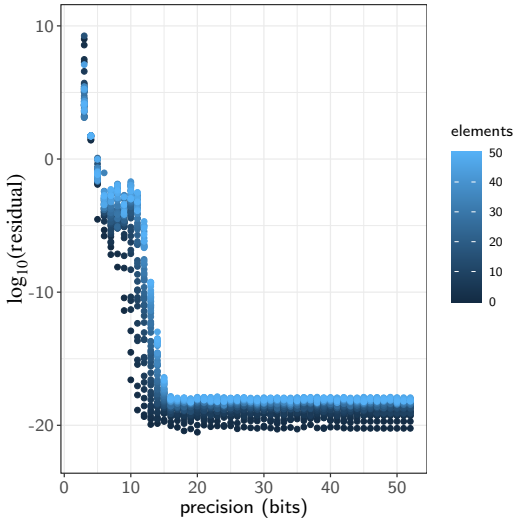


Figure 1: Examining precision needs in Nekbone for various numbers of elements: the residual (L^2 norm) in CG.

Recently, following this precision inspection, we modified Nekbone to support mixed single-double precision and we were able to reduce the time-to-solution by 34 %. Once a suitable precision is found, a Verificarlo CI probe can be inserted in the code, to automatically monitor the residual error of each subsequent code version.

4. Tracking accuracy in the QMCkl library

The Sherman-Morrison-Woodbury (SMWB) kernel was developed as part of the QMCkl library[2], an open-source library of highly-optimized building blocks for implementing Quantum Monte Carlo methods in the TREX European Center of Excellence.

Given a matrix A and its inverse A^{-1}, Sherman-Morrison (SM) is a formula to efficiently compute the inverse after a rank-1 update uv^T on A

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}. \quad (1)$$

This formula can be generalized for rank-k updates using the Woodbury (WB) formulation [6]. In WB, the denominator of SM is replaced by the inverse of a small square $k \times k$ matrix. For $k = 2$ and $k = 3$, WB is expected to be faster than

[2] `https://github.com/TREX-CoE/qmckl`

iterating 2 or 3 times with SM. QMCkl implements different algorithms to apply SM with a set of updates (u_j, v_j), for $j = 1, ..., m$. The naive approach, *SM1*, applies these updates in sequence.

Depending on the updates ordering, the SM denominator can be close to zero, meaning that the matrix A becomes singular. This makes the method numerically unstable. A refined algorithm using Slagel splitting [7] is called *SM2*. Below a minimum threshold for the denominator, the update is split in two, the first half is applied, and the second half enqueued with remaining updates.

To implement the Woodbury formula, blocks of rank-3 and rank-2 updates are built. If the intermediate matrix update is singular, the corresponding updates are split with SM2. This method is called *SMWB*. Since *SMWB* changes the order of operations, one must ensure that the numerical accuracy is preserved compared to *SM2*.

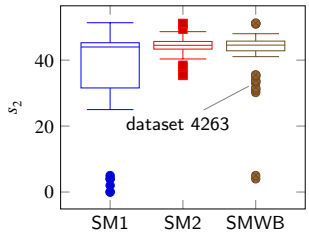


Figure 2: Significant bits of Frobenius norm, for all datasets and algorithm combinations, for commit `6f282`, grouped by algorithms. SMWB fails catastrophically in some cases.

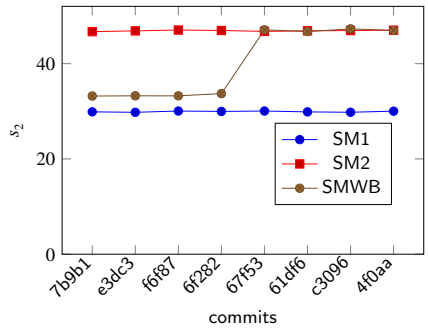


Figure 3: Significant bits of Frobenius norm, for our different algorithms, over commits for dataset 4263. SMWB's accuracy improves after the fix.

To track the accuracy of these algorithms during development with Verificarlo CI, we use a large number of datasets from a QMC=Chem [8] use case on Benzene. All datasets are run with all algorithms in `MCA` mode. The main development branch in the repository was instrumented with probes identified with *dataset number* / *algorithm* couples allowing a factored analysis in the dynamic reports. Finally, we set up a CI branch using the command-line helper to track accuracy on the main development branch, from which Fig. 2 and Fig. 3 were generated.

During the development of SMWB, the *run inspection* report, reproduced in Fig. 2, highlighted some outputs for which SMWB fails with a high error under the MCA backend. After investigation, we discovered that in the initial implementation of SMWB, delayed updates were directly applied after each WB step. This reduces the numerical stability because it increases the probability of producing singular intermediate matrices. It was fixed in commit `67f53` by moving all the updates to the very end of the update queue as shown in Fig. 3, obtained from the *temporal view*.

5. Conclusion

Verificarlo CI automates numerical accuracy tests within a continuous integration workflow: it grants users the ability to define such tests. It provides an easy way to visualize results throughout the development process of a code. Better integration of numerical checks in the CI process saves developers precious time to focus on their area of expertise.

Verificarlo CI has been used in the context of TREX and CEEC EuroHPC JU Centers of Excellence to detect numerical regressions, pin-point faulty commits, and predict the effect of mixed-precision. A tutorial demonstrating its use is available on GitHub[3]. Furthermore, we believe that using such a tool as a part of the regular CI/ CD process would help for early stage identification of numerical bugs and re-ensuring numerical reliability of codes.

Acknowledgements

This work was partially supported from EC by EuroHPC Centers of Excellence TREX (952165) and CEEC (101093393), as well as by the French National Agency for Research via the InterFLOP project (ANR-20-CE46-0009).

References

[1] D. Goldberg, What every computer scientist should know about floating-point arithmetic, ACM Computing Surveys 23 (1) (1991) 5–48. doi:10.1145/103162.103163.

[2] C. Denis, P. De Oliveira Castro, E. Petit, Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic,

[3] https://github.com/verificarlo/vfc_ci_tutorial

in: 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH), IEEE, 2016, pp. 55–62. doi:10.1109/ARITH.2016.31.

[3] D. S. Parker, D. Langley, Monte carlo arithmetic: exploiting randomness in floating-point arithmetic (1997).
URL https://api.semanticscholar.org/CorpusID:2321215

[4] D. Sohier, P. D. O. Castro, F. Févotte, B. Lathuilière, E. Petit, O. Jamond, Confidence Intervals for Stochastic Arithmetic, ACM Transactions on Mathematical Software 47 (2) (2021) 1–33. doi:10.1145/3432184.

[5] Y. Chatelain, E. Petit, P. de Oliveira Castro, G. Lartigue, D. Defour, Automatic Exploration of Reduced Floating-Point Representations in Iterative Methods, Vol. 11725, 2019, pp. 481–494. doi:10.1007/978-3-030-29400-7_34.

[6] M. Woodbury, P. U. D. of Statistics, Inverting Modified Matrices, Memorandum Report / Statistical Research Group, Princeton, Department of Statistics, Princeton University, 1950.
URL https://books.google.de/books?id=_zAnzgEACAAJ

[7] J. T. Slagel, The sherman morrison iteration, Master's thesis, Virginia Tech (2015).

[8] A. Scemama, M. Caffarel, E. Oseret, W. Jalby, QMC=Chem: A Quantum Monte Carlo Program for Large-Scale Simulations in Chemistry at the Petascale Level and beyond, Vol. 7851, 2013, pp. 118–127. doi:10.1007/978-3-642-38718-0_14.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Assessing Computational Fluid Dynamics on GPU Using Portable Languages

Youssef Faqir-Rhazoui[a,*], Carlos García Sánchez[b]

[a]Eviden, R&D Department, C/ Albarracín, 25, 28037 Madrid, Spain
[b]Complutense University of Madrid, Avda. de Séneca, 2 Ciudad Universitaria, 28040 Madrid, Spain

ARTICLE INFO[†]

Keywords:
Computational Fluid Dynamics;
Graphics Processing Unit;
OpenMP;
SYCL;
Compute Unified Device Architecture;
Heterogeneous-Computing Interface for Portability

ABSTRACT

Accelerators are essential for achieving optimal performance and energy efficiency in computing. However, market segmentation often leads to language lock-ins, limiting flexibility across accelerators and increasing development costs. To address this, alternatives like SYCL or OpenMP have emerged, enabling code portability across diverse hardware platforms. The study reveals that while both OpenMP and SYCL demonstrate comparable performance to native languages on NVIDIA and Intel GPUs, SYCL significantly outperforms OpenMP on AMD platforms. These findings underscore the potential of multi-device and open languages in enhancing performance and reducing development overhead in parallel CFD simulations.

1. Introduction

Hardware acceleration is a crucial component in the quest for enhanced performance and energy efficiency. However, due to market segmentation, many accelerators (e.g., GPUs) suffer from language lock-ins (e.g., CUDA, HIP), restricting the use of the same language across multiple vendor accelerators [1]. Moreover, CI/CD pipelines would be compromised by these practices, as maintaining multiple pipelines for each architecture would be error-prone and increase the complexity of the process. CD would require maintaining multiple implementations of the same algorithm in different languages. CI would also be tied to each language, as each language is tied to its own compiler [2]. To address these issues, various alternatives have emerged, such as OpenCL, SYCL, or OpenMP. These languages have the capability to run on multi-vendor accelerators such as CPUs or GPUs while using the same code.

Computational Fluid Dynamics (CFD) is a powerful tool for simulating complex environments such as turbomachinery, aerodynamics, or heat transfer. CFD simulations were traditionally performed on CPUs and distributed across clusters using MPI. Due to the inherently parallel nature of CFD, GPU acceleration is key to leveraging performance and energy efficiency. However, some areas of CFD, such as particle simulation, still remain mostly implemented on the CPU [3].

This paper presents a suite of benchmarks for testing CFD codes on NVIDIA, AMD, and Intel GPUs. We compare the native language of each platform with the portable languages OpenMP and SYCL.

The following paper is structured as follows: In Sec. 2, we present the environmental conditions and methods used in the experiment. Section 3 describes the results obtained and discusses them.

2. Methods

The experiments were conducted using the GPUs listed in Tab. 1. The selected GPUs cover NVIDIA, AMD and Intel architectures. While the table specifies the driver used for each GPU, it is worth mentioning that the V100 employs the CUDA 12.4 toolkit, the RX 6700XT is powered by ROCm 5.4.3, and the Max 1100 is integrated with oneAPI 2024.1.

Transitioning to portable languages (OpenMP & SYCL), it is worth distinguishing each one. While OpenMP is supported in all the previously mentioned toolkits. SYCL is supported by the oneAPI's compiler and is suitable for NVIDIA and AMD GPUs.

Regarding the benchmarks, we employed a set of eleven CFD codes combined with common CFD equations. For the sake of space, we are not providing detailed descriptions and parameters of the benchmarks here. Instead, we strongly recommend that readers refer to the repository where this information is provided[1]. Concerning the benchmark implementation of each language and how we keep them as similar as possible, the baseline implementation was CUDA.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02466 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
youssef.el@eviden.com (Y. Faqir-Rhazoui); garsanca@ucm.es (C. García Sánchez)
ORCID(s): 0000-0001-7833-1086 (Y. Faqir-Rhazoui); 0000-0002-3470-1097 (C. García Sánchez)

[1]The repository used can be found at: https://github.com/A924404/cfd-bench

	NVIDIA Tesla V100	AMD RX 6700 XT	Intel Max 1100
Frequency (GHz)	Up to 1.38	Up to 2.58	Up to 1.55
Cores	80 SM	40 CU	56 Xe cores
Perf. (FP64)	7.06 TFLOPS	0.825 TFLOPS	22.22 TFLOPS
Driver	550.54.14	5.18.3	23.52 (ocl) 1.3 (level0)

Table 1
Specifications of the GPU used in the experimentation.

Translating the code to HIP was achieved using the HIPIFY tool[2] Since OpenMP syntax is quite different from the other employed languages and there is no official tool to port it, the port was done manually to be as similar as possible to the CUDA code. Once the OpenMP code was delivered, it was compiled for each architecture using the vendor compiler for that architecture. Finally, SYCL code was mainly ported using the SYCLomatic tool[3]. In this case, SYCL code was always compiled for all architectures using the oneAPI compiler.

3. Experimental Results and Results Discussion

3.1. Results

Reviewing Fig. 1 for the Tesla V100, CUDA serves as the native benchmark. SYCL fails to run the *miniWeather* benchmark due to an issue related to its interaction with MPI. In the case of OpenMP, the *d3q19-bgk* test was not implemented in the original suite. SYCL averaged 91% of CUDA performance, while OpenMP reached 80%. Depending on the benchmark, these percentages may vary.

Figure 2 shows results from the AMD 6700 XT. The native implementation for AMD GPU is HIP. While both HIP and SYCL ran all benchmarks, OpenMP failed the *lid-driven-cavity* test, the main issue found relies on memory allocation. While HIP and SYCL are able to allocate the amount of memory required by the benchmark, OpenMP do not. The results show poor performance on AMD architecture. OpenMP achieves only 51% of HIP times on average, while SYCL reaches up to 66% of native performance. The standard deviation (SD) for OpenMP is 44% and for SYCL it is 35%. SYCL's performance on AMD is primarily affected by the *adv* and *miniWeather* tests, but removing them increases performance to 81% with an SD of 19%. OpenMP's issue is spread across all benchmarks.

Finally, the Intel Max 1100 results are shown in Fig.3. SYCL runs on two backends: Level0 and OpenCL, both maintained by Intel with no notable differences expected. In

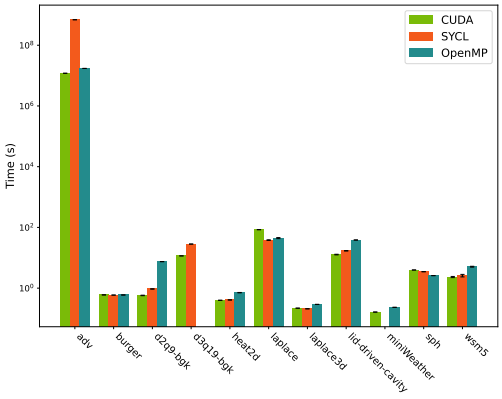


Figure 1: NVIDIA Tesla V100 performance comparison across CUDA, OpenMP, and SYCL.

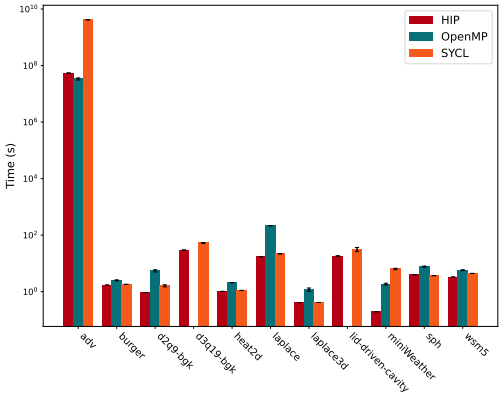


Figure 2: AMD RX 6700 XT performance comparison across HIP, OpenMP, and SYCL.

[2] https://github.com/ROCm/HIP?tab=readme-ov-file.
[3] https://github.com/oneapi-src/SYCLomatic

	CUDA	HIP	SYCL	OpenMP
Cuda ptx	33,455	-	46,259	144,999
Amdgcn	-	23,757	43,321	132,692

Table 2
Total assembly code lines by language and architecture.

this instance, all three implementations failed to execute the *miniWeather* mini-app due to MPI call incompatibilities in the system, unrelated to the languages used, but rather due to the absence of an MPI installation. Additionally, the authors did not have root access to the system employed for Intel GPUs.

Regarding the numbers and SYCL, both backends are virtually equivalent in performance, achieving an average speedup of ×1. Transitioning to OpenMP, it shows a slight speedup over the SYCL equivalents, averaging ×1.05.

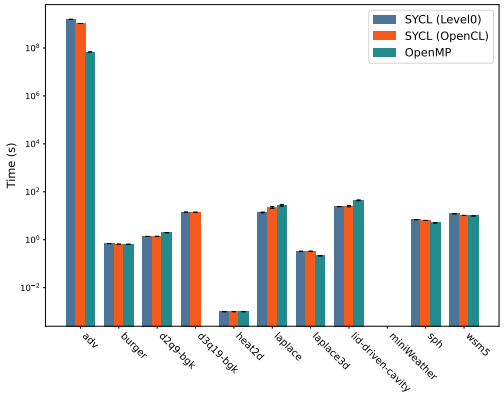


Figure 3: Intel Max 1100 performance comparison across SYCL(Level0), SYCL(OpenCL), and OpenMP.

3.2. Discussion

Open and portable languages such as SYCL and OpenMP are promising in scenarios where changing GPUs necessitates completely altering the underlying source code and CI/CD pipelines due to vendor lock-ins. In performance results, SYCL has shown a performance comparable to NVIDIA's (10% difference). However, when moving to AMD GPUs, the performance is degraded due to the immaturity of the language on this platform. SYCL is also a native language for Intel GPUs, allowing for highly optimized performance. Regarding OpenMP, we found mixed results. On Intel platforms, there is no performance loss, while on NVIDIA, the gap is approximately 20%, and on AMD, it is around 50%. Both SYCL and OpenMP are ultimately translated into native assembly code that the GPU executes. Therefore, the efficiency of translating high-level code to low-level code is crucial for achieving performance. Table 2 shows the number of assembly lines into which the target languages are translated[4]. Generally, more lines imply more time for the GPU to run them. For CUDA PTX, SYCL employs 27% more code, while OpenMP uses 77% more. For Amdgcn, SYCL uses 45% more assembly code, and OpenMP uses up to 82% more.

Acknowledgements

This paper is part of the HIDALGO2 project, co-funded by the European Union under grant agreement number: 101093457.

References

[1] M. Breyer, A. Van Craen, D. Pflüger, A Comparison of SYCL, OpenCL, CUDA, and OpenMP for Massively Parallel Support Vector Machine Classification on Multi-Vendor Hardware, in: International Workshop on OpenCL, ACM, New York, NY, USA, 2022, pp. 1–12. doi:10.1145/3529538.3529980.

[2] P. Rostami Mazrae, T. Mens, M. Golzadeh, A. Decan, On the usage, co-usage and migration of CI/CD tools: A qualitative analysis, Empirical Software Engineering 28 (2) (2023) 52. doi:10.1007/s10664-022-10285-5.

[3] A. Zhu, Q. Chang, J. Xu, W. Ge, A dynamic load balancing algorithm for CFD–DEM simulation with CPU–GPU heterogeneous computing, Powder Technology 428 (2023) 118782. doi:10.1016/j.powtec.2023.118782.

[4] We did not include the same information for the Intel GPU because we could not generate the assembly codes. The required tool needs to be installed with root access, which we do not have.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Comparing Several HPC CFD Software Through Codemetrics: A Case Study

Thibault Marzlin[a,*], Antoine Dauptain[a]

[a]CERFACS, ALGO / COOP, 42 Av. Gaspard Coriolis, 31300 Toulouse, France

ARTICLE INFO[†]

Keywords:
High-Performance Computing;
Software Engineering;
Cyclomatic Complexity;
Software Architecture

ABSTRACT

This study compares various major Computational Fluid Dynamics (CFD) solvers used in High-Performance Computing (HPC) environments, focusing on their structures and communities. The findings reveal common patterns in the strategies employed to address technical debt in HPC. Multiple code quality evaluation (codemetrics) approaches are utilized in this research, tailored to accommodate the specific constraints of the HPC domain, thereby examining both technical and human factors involved.

1. Introduction

In High-Performance Computing, as the hardware evolves rapidly without reaching complete standardization, the cost incurred by code development and aging - technical debt - keeps increasing. Additionally, achieving full maturity in the validity and performance of software is harder. This redirects experts' focus on these aspects at the expense of technical debt. In areas such as physical modeling, numerical methods, applied mathematics and high-performance computing, the expertise needed to develop and master a code requires more time than ever and more specialized skills. This makes any waste of experts time even more unacceptable.

Regarding the HPC domain, Reed et al. [1] suggested that high-performance technologies evolve faster than people can keep up with. For example, GPU-powered (Graphics Processing Unit) supercomputers have experienced notable prominence since 2010, yet many legacy-HPC applications born before this era are still figuring how to adapt to GPU-based supercomputers at an affordable cost. Meanwhile, Codemetrics - a set of measurements that estimates code complexity - are useful to provide accurate information to development teams. Codebase analysis isn't the only focus of Codemetrics [2, 3]. Indeed, Codemetrics also target the team involved [4]: their perception of the code, how they navigate and retrieve information from it.

2. Codemetrics

Codemetrics is a recent field of investigation, well-established in mainstream software development but seldom applied to HPC software. After a short state of the art on the existing solutions, the need of HPC-specific codemetrics will be detailed.

2.1. Existing solutions

While existing solutions address mainstream development processes, particularly focusing on the detection and management of complex segments. Additionally, some of these solutions (CodeScene[1], Doxygen[2]) offer graphical representations of code: a network view that illustrates interdependencies within the codebase.

These tools, such as those provided by SonarSource[3], Understand[4] and CodeScene, enable users to gain qualitative insights such as maintainability, duplication rate, coverage of their codebases. Codee[5], offers solutions tailored to the unique challenges posed by legacy systems. Indeed, different coding standards and versions of coding languages are covered. These type of tools have already been used in academic studies [5, 6] by bridging the gap between the metric computed and the overall perception of a set of developers towards given code snippets.

2.2. Why HPC need tailored codemetrics

Regrettably, mainstream codemetrics yield highly unfavorable results when applied to HPC software. A codebase originating in the 1990s, amalgamating the efforts of numerous PhD students, and continually adjusted to keep pace with

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02467 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
thibault.marzlin@cerfacs.fr (T. Marzlin);
antoine.dauptain@cerfacs.fr (A. Dauptain)
ORCID(s): 0009-0001-2183-2936 (T. Marzlin); 0009-0009-1933-555X (A. Dauptain)

[1]https://codescene.com
[2]https://doxygen.nl
[3]https://www.sonarsource.com
[4]https://scitools.com
[5]https://www.codee.com

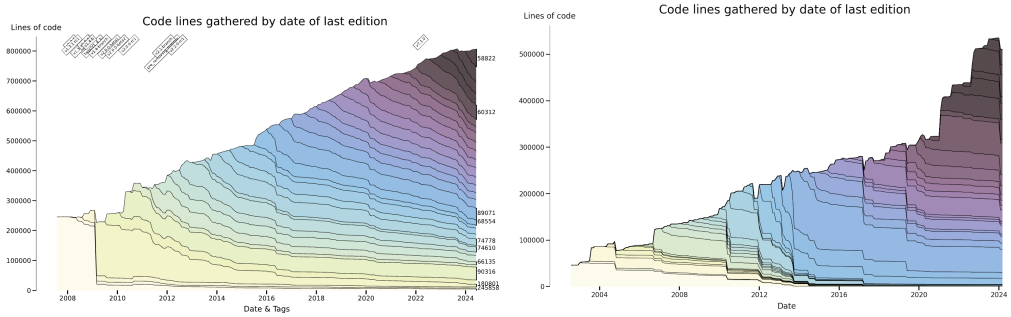


Figure 1: The lines of code of the HPC-CFD solver *SATURNE* (left) and *AVBP* (right) are stacked over time, colored by the date of the edition.

the latest hardware advancements, inevitably exhibits significant complexity, limited contributor engagement, code bloat, and instances of dead code, among other issues.

The current approach concentrates on well-established, and consequently successful, HPC Computational Fluid Dynamics (CFD) software, assessed through codemetrics. To the best of our knowledge, there has been no systematic comparison between communities and projects within the HPC realm.

A community-aware analysis will be employed to elucidate the human aspect of the development process. The technical debt of HPC codes will be evaluated from both a historical standpoint and a structural perspective.

First of all, with the use of *Git* platform (Versioning Control System) the in-house tool: *Anubis*[6] is giving access to the history of the codebase and its evolution. It allows for a deeper understanding of developers' relationships with the codebase, as well as providing a visualization of workforce dynamics.

A loop processes each month's version of the code using `git checkout`, building a database of various quantities of interest : authors, commits by authors, complexity, and branches deltas. For the 20+ years of the code's existence (AVBP), this loop takes approximately ten hours to complete. Afterward, customizable reports on several code metrics can be generated.

Figure 1 illustrates the growth of the codebase for two mature HPC codes, *SATURNE* and *AVBP*. While *AVBP* shows several strong refactoring in its history, *SATURNE* exhibits a continuous and steady evolution.

A second tool: *Maraudersmap*[7] create a geographical representations of the codebase. Through the depiction of a Callgraph, which illustrates a network of code blocks and their interdependencies, we can discern inadequate structural practices and streamline the tracking of complexity, size, and dependencies.

For a 500,000-line Fortran codebase, it takes 3 minutes to clean the CPP directives, analyze the code structure, and build the call graph connectivity. The tool then offers various features to filter the graph, collapse abstractions (such as objects and methods) into single nodes, and color by structure, complexity, or custom string patterns.

Figure 2 shows the general overview of two large CFD solvers *SATURNE* and *AVBP*. The most striking difference is *SATURNE* using a large unified interface "bindings" where AVBP is relying on a procedural "Main" structure. Several filtering were used in both cases to untangle the callgraphs to this point.

2.3. Case studies

The primary objective is to gather data from authentic projects encompassing diverse HPC codes characterized by varying standards, teams, and management approaches. Ultimately, the goal is to uncover commonalities among these communities to pinpoint the most effective and enduring strategies employed by teams in the HPC field. Naturally, a wide array of development standards were observed. Legacy systems frequently keep parts written in older versions of programming languages such as Fortran 77 and 90.

Nine large codes ($> 100k$ lines) have been analyzed: *Alya*, *AVBP*, *Yales2*, *Neko*, *NekRS*, *Nek5000*, *MesoNH*, *Oasis-MCT*, *Saturne*. These comparisons yield valuable insights into how a code evolves and is monitored over time.

A comprehensive discussion of the results is beyond the scope of this abstract. However, this exercise yielded significant feedback from the communities involved. Many groups highlighted the importance of monitoring the impact of code abstractions. Specifically, hardware-agnostic HPC

[6] https://gitlab.com/cerfacs/anubis
[7] https://gitlab.com/cerfacs/maraudersmap

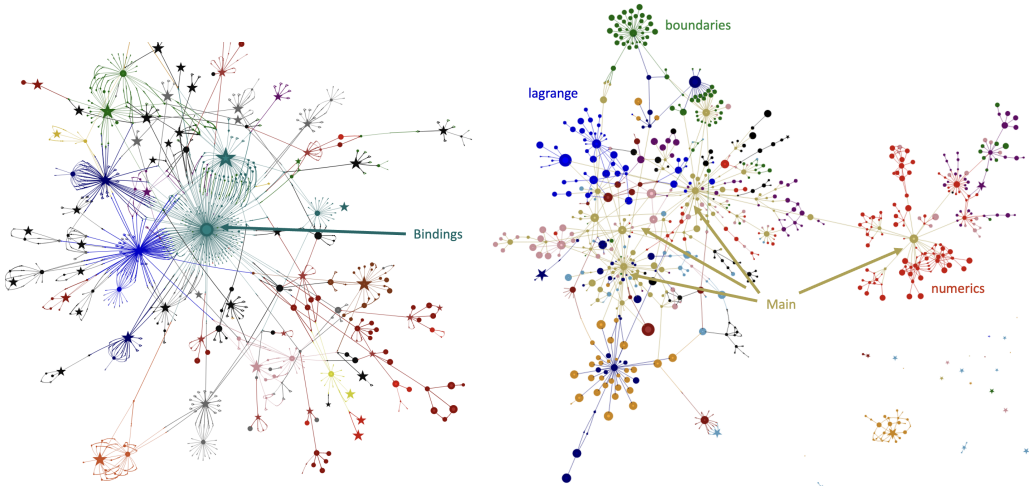


Figure 2: General, filtered, callgraphs of CFD HPC codes *SATURNE* (left) and *AVBP* (right), both featuring a large amount of physical modeling. Arbitrary colors are denoting main folders.

libraries like Kokkos are used through templates, which has consequently increased the learning curve for newcomers.

3. Conclusion

This work demonstrates the need of an initial implementation as well as early results from decision-making tools designed to address technical debt in High-Performance Computing (HPC) software. Augmented call graphs significantly improve training, code reviews, refactoring, and flaw detection. The subjective nature of code metrics is mitigated by observing their long-term evolution, providing historical context that facilitates team discussions and decision-making.

Future publications will include rigorous quantitative comparisons across multiple codes for deeper insights. Our ongoing research will also develop new complexity indicators for HPC, such as a 'structural complexity' index for abstraction layers and a 'preprocessing index' for CPP directives, to better manage template programming and legacy code maintenance.

Acknowledgements

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Italy, Slovenia, Spain, Sweden, and France under grant agreement No. 101092621.

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU) and Germany, Italy, Slovenia, Spain, Sweden, and France. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] D. Reed, D. Gannon, J. Dongarra, Reinventing High Performance Computing: Challenges and Opportunities (2022). arXiv:2203.02544.

[2] T. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering SE-2 (4) (1976) 308–320. doi:10.1109/TSE.1976.233837.

[3] A. Tornhill, Your Code as a Crime Scene, Pragmatic Bookshelf, 2024.
URL https://www.adamtornhill.com/articles/crimescene/codeascrimescene.htm

[4] S. Himayat, D. J. Ahmad, Software Understandability using Software Metrics: An Exhaustive Review, SSRN Electronic Journal (2023). doi:10.2139/ssrn.4447189.

[5] M. Esposito, A. Janes, T. Kilamo, V. Lenarduzzi, Early Career Developers' Perceptions of Code Understandability. A Study of Complexity Metrics (2023). arXiv:2303.07722.

[6] L. Lavazza, S. Morasca, M. Gatto, An empirical study on software understandability and its dependence on code characteristics, Empirical Software Engineering 28 (6) (2023) 155. doi:10.1007/s10664-023-10396-7.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Enabling Lighter and Faster Simulations With Repeated Matrix Blocks

Josep Plana-Riu[a,*], F. Xavier Trias[a], Guillem Colomer[a], Ádel Alsalti-Baldellou[b], Xavier Álvarez-Farré[c] and Assensi Oliva[a]

[a]Technical University of Catalonia, ESEIAAT, Heat and Mass Transfer Technological Centre, Carrer de Colom 11, 08222, Barcelona, Spain
[b]University of Padova, Department of Information Engineering, Via Giovanni Gradenigo, 6b, 35131, Padova PD, Italy
[c]SURF, High-Performance Computing and Visualization Team, Science Park 140, 1098 XG, Amsterdam, The Netherlands

ARTICLE INFO†

Keywords:
Sparse Matrix-Vector Product;
Sparse Matrix-Matrix Product;
Arithmetic Intensity;
High Performance Computing

ABSTRACT

When pushing towards lighter and faster Computational Fluid Dynamics simulations, the contribution and construction of every component should be considered. The sparse matrix-vector product (SpMV) is the most expensive kernel among all operations. In some situations, e.g, with spatial reflection symmetries, the sparse matrices have some repeated blocks that could be exploited for better performance. By transferring the repeated blocks only once, the amount of data to transfer is reduced, and thus, the memory footprint of the simulation will be reduced. Moreover, with this framework, the SpMV is transformed into a sparse matrix-matrix product (SpMM), reducing the memory footprint and speeding-up the simulation. The method is tested in a differentially heated cavity in order to test the performance gains with the use of the SpMM compared to the use of a SpMV.

1. Introduction

Pushing toward bigger and bigger simulations of the incompressible Navier-Stokes equations requires an increase in the computational power available in high-performance computing (HPC) systems. Nonetheless, as these cases require a lot of memory and data transfer, it is not only the time spent computing that is relevant. Yet, the time spent in data transferring becomes the most relevant part of the time budget of the simulation, leading to what is known as a memory-bound process [1].

If the Navier-Stokes equations are solved numerically, it is straightforward to divide the operations into the well-known sparse matrix-vector product (SpMV), the dot product (dot), the linear combination of vectors (axpy), and the element-wise product of vectors (axty). These operations appear naturally in an algebraic approach, yet in stencil-based approaches these operations are implemented based on nested mesh-loops.

SpMV, being the most computationally expensive operation, requires transferring the data of the sparse matrix and the data of the whole full vector. This implies that the amount of data to be transferred for bigger simulations takes a relevant time compared to the time spent executing the operation. In this sense, some conditions might be exploited to transfer a smaller amount of data, either in the matrix, in the vector, or both. This implies that the kernel's arithmetic intensity (AI), defined as the ratio of the computing load and the data transferred, is low. Other techniques as the presented by Greathouse and Daga [2] for GPU compute units aim to improve the performance of SpMV by mapping properly the loads of the sparse matrix, leading to remarkable speed-ups compared to the original CSR-based algorithms.

For instance, Krasnopolsky [3] developed the concept of solving n flow states simultaneously to later on ensemble averaging the results of these flow states. In this paper, the SpMV operations were translated to sparse matrix-matrix products (SpMM) to all the n flow states simultaneously as a single operation. By doing so, the amount of data to be transferred compared to running n times the simulation was reduced, as the sparse matrix was only transferred once. By doing so, the AI would increase notably, as the amount of computations would be preserved while the amount of data transferred was reduced.

Later on, Alsalti-Baldellou et al. exploited the domain's symmetries [4] or repetitions [5] to reduce the matrix's size to transfer by splitting the domain in the inner cells within the symmetric part and the bounds between every symmetric contribution. By doing so, the number of right-hand sides (RHS), i.e., the number of columns in the full matrix, would be 2^d, being d the number of symmetries in the domain.

† This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02468 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
josep.plana.riu@upc.edu (J. Plana-Riu); francesc.xavier.trias@upc.edu (F.X. Trias); guillem.colomer@upc.edu (G. Colomer); adel.alsaltibaldellou@unipd.it (Á. Alsalti-Baldellou); xavier.alvarezfarre@surf.nl (X. Álvarez-Farré); asensio.oliva@upc.edu (A. Oliva)
ORCID(s): 0000-0002-9086-0519 (J. Plana-Riu); 0000-0002-5966-0703 (F.X. Trias); 0000-0003-4592-0217 (G. Colomer); 0000-0002-5331-4236 (Á. Alsalti-Baldellou); 0000-0002-1684-7658 (X. Álvarez-Farré); 0000-0002-2805-4794 (A. Oliva)

While in the former, the methodology was only applied in the solution of the linear system of equations of the projection method, and no results in the speed-up of SpMM were presented, the latter applied this methodology to all the SpMV operations within the simulation, i.e., in the use of the gradient, divergence, and Laplacian operators.

Thus, the presented methodology can be applied in different situations apart from the ones previously mentioned: ensemble averaging of the time averaged results of turbulent flow simulations, application in domains with mirror symmetries or repeated geometrical structures (e.g. wall mounted cubes, wind farm), or parametric studies by changing relevant values in the simulations. Nonetheless, the framework of this study is based on ensemble averaging the solutions, following the works of Krasnopolsky [3].

2. Application to CFD simulations

Following the notation of Verstappen and Veldman [6], the semi-discrete incompressible Navier-Stokes equations read as follows for a staggered case,

$$M\mathbf{u}_s = 0, \tag{1}$$

$$\Omega\frac{d\mathbf{u}_s}{dt} + C(\mathbf{u}_s)\mathbf{u}_s = -\Omega G\mathbf{p}_c + D\mathbf{u}_s + \mathbf{f}_s, \tag{2}$$

where M is the face-to-cell divergence operator, $\Omega = I_3 \otimes \Omega_s$, where Ω_s is a diagonal matrix containing the staggered volumes; C is the convective operator represented by a skew-symmetric matrix given a symmetry-preserving discretization; D is the diffusive operator and G is the cell-to-face gradient operator, and $\mathbf{f}_s$ represents the body forces.

More precisely, the tests were run in one high memory node of MareNostrum5 supercomputer (2x Intel Xeon Platinum 8480, 2x56 CPU cores) loading the node with around 400k cells per CPU core (46.65M cells), so that the largest cases would fit in the nodes.

The time-integration scheme used was a Heun third-order Runge-Kutta scheme (RK3) with a self-adaptive time step size computed based on the eigenbounds of the convective and diffusive operators. The results were tested to be independent of the scheme used.

The discretization scheme for the operators considered only the first neighbor cell, leading to 7 non-zeros per row in a three-dimensional case. Nonetheless, the Laplacian operator for the Poisson equation was tested considering the first neighbor (7p), a cross pattern (13p, 13 non-zero entries per row), and a cube pattern (27p, 27 non-zero entries per row). A two-dimensional representation of these stencils is shown in Fig. 1.

The tests were run by setting a fixed number of iterations in the Poisson solver solution. The speed-up was computed as follows: considering the $n = 1$ simulation as the baseline, with a SpMV time T_{SpMV}, the speed-up for a given n with $T_{\text{SpMM(n)}}$ is determined by

$$P_{n,\text{SpMM}}(n) = \frac{nT_{\text{SpMV}}}{T_{\text{SpMM(n)}}}. \tag{3}$$

In this case, the tests will be applied to a differentially heated cavity (DHC) setup with RAYLEIGH number $\text{Ra} = 10^{10}$ and PRANDTL number $\text{Pr} = 0.71$ with aspect ratio 4 in a setup similar to Krasnopolsky [3] so that multiple flow states, which correspond to 1, 2, 4, and 8 RHS, will be launched simultaneously for a few iterations to compute the speed-ups of the SpMM operations. A higher number of non-zeros in the sparse matrix positively affects the speed-ups obtained, as for 8 RHS, the speed-ups go from maximum values of ≈2.5 to ≈3.5 for 7p and 27p, respectively.

Figure 2 presents the results for 150, 350, and 550 iterations per solution of the Poisson equation for all 7p, 13p, and 27p. The results in all three cases lay between the theoretical upper and lower bounds, according to Alsalti-Baldellou et al. [5]. It would be expected to obtain a slightly better performance for a greater number of iterations as the weight of the Poisson equation would rise in the overall wall clock time. However, the relevance of a greater number of iterations in the speed-up should decrease the bigger the value.

The method has been tested as well for other cases such as a turbulent planar channel flow of $\text{Re}_\tau = 180$ with a similar mesh and load, leading to equivalent results to the presented in Fig. 2.

3. Conclusions

In this work, the speed-up analysis obtained in the use of SpMM in simulations in which there are repeated matrix blocks compared to the use of a single RHS (i.e., a SpMV) is presented and compared against the theoretical upper and lower bounds for three different configurations, presented in Fig. 1: 7p, 13p and 27p.

It can be seen in Fig. 2 that the numerical speed-ups are obtained between the theoretical upper and lower bounds, being approximately equidistant to both bounds in all the cases run for both DHC and CF. Moreover, it can be observed that the denser the sparse matrix, the higher the speed-up, leading to an increased interest in applying the method to simulations in which the discretization is of a higher order, i.e., with more non-zeros per row, compared to using only the first neighbor, with 7 non-zeros per row.

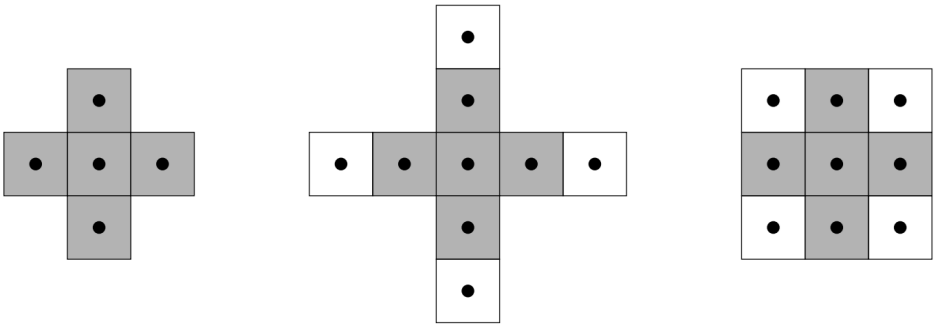


Figure 1: Two-dimensional representations of the stencil for 7p (left), 13p (center) and 27p (right). The shaded cells represent the first neighbors in all representations.

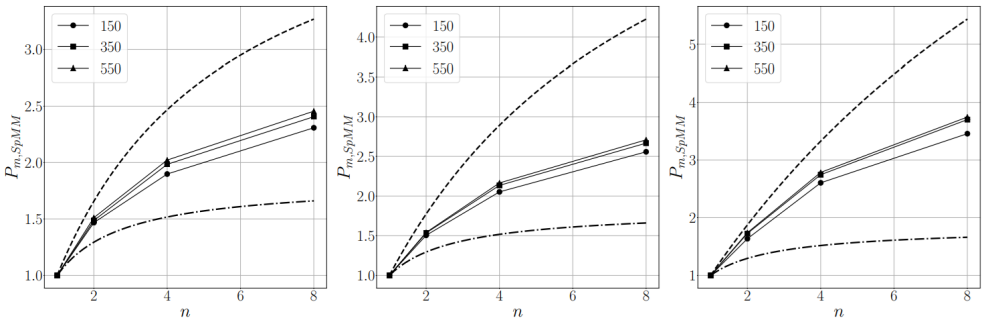


Figure 2: Speed-up for the 7p (left), 13p (center), and 27p (right) discretizations for a given number of Poisson solver iterations in the DHC case. The dashed line provides the upper bound of the speed-up, whereas the dot-dash line defines the lower bound, according to Alsalti-Baldellou et al. [5].

Acknowledgements

The work presented in this abstract is supported by the Ministerio de Economía y Competitividad, SIMEX project (PID2022-142174OB-I00). J. Plana-Riu is also supported by the Catalan Agency for Management of University and Research Grants (AGAUR), 2022 FI_B 08110. Á. Alsalti-Baldellou is also supported by the project "National Centre for HPC, Big Data and Quantum Computing ", CN00000013 (approvato nell'ambito del Bando M42C – Investimento 1.4 – Avvisto "Centri Nazionali" – D.D. n. 3138 del 16.12.2021, ammesso a finanziamento con Decreto del MUR n. 1031 del 17.06.2022). Calculations were performed on the MareNostrum 5 General Purpose partition (MN5 GPP) supercomputer at the Barcelona Supercomputing Centre (BSC). The authors thankfully acknowledge these institutions.

References

[1] S. Williams, A. Waterman, D. Patterson, Roofline: an insightful visual performance model for multicore architectures, Communications of the ACM 52 (4) (2009) 65–76. doi:10.1145/1498765.1498785.

[2] J. L. Greathouse, M. Daga, Efficient sparse matrix-vector multiplication on GPUs using the CSR storage format, in: SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2014, pp. 769–780. doi:10.1109/SC.2014.68.

[3] B. I. Krasnopolsky, An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles, Computer Physics Communications 229 (2018) 8–19. doi:10.1016/j.cpc.2018.03.023.

[4] A. Alsalti-Baldellou, X. Álvarez Farré, F. Trias, A. Oliva, Exploiting spatial symmetries for solving Poisson equation,

Journal of Computational Physics 486 (2023) 112133. doi:10.1016/j.jcp.2023.112133.

[5] A. Alsalti-Baldellou, X. Álvarez Farré, G. Colomer, A. Gorobets, C. D. Pérez-Segarra, A. Oliva, F. X. Trias, Lighter and faster simulations on domains with symmetries, Computers and Fluids 275 (2024) 106247. doi:10.1016/j.compfluid.2024.106247.

[6] R. Verstappen, A. Veldman, Symmetry-preserving discretization of turbulent flow, Journal of Computational Physics 187 (1) (2003) 343–368. doi:10.1016/S0021-9991(03)00126-8.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

A Portable Algebraic Implementation for Reliable Overnight Industrial LES

Marcial Mosqueda-Otero[a,*], Àdel Alsalti-Baldellou[a,b], Xavi Álvarez-Farré[c], Josep Plana-Riu[a], Guillem Colomer[a], Francesc Xavier Trias[a] and Asensio Oliva[a]

[a] Universitat Politècnica de Catalunya, BarcelonaTech, Heat and Mass Transfer Technological Center, Carrer de Colom 11, 08222 Terrassa, Barcelona, Spain
[b] University of Padova, Department of Information Engineering, Via Giovanni Gradenigo, 6b, 35131 Padova, Italy
[c] SURF, High-Performance Computing Team, Science Park 140, 1098 XG Amsterdam, Netherlands

ARTICLE INFO†

Keywords:
Message Passing Interface;
OpenMP;
OpenCL;
Heterogeneous Computing;
Large-Eddy Simulation;
Overnight Industrial Applications

ABSTRACT

This work assesses the feasibility of large-scale simulations for industrial applications using a symmetry-preserving discretization method for unstructured collocated grids in LES of turbulent flows. The method ensures stability without artificial dissipation and maintains portability through minimal algebraic kernels. Key challenges are addressed, such as the low arithmetic intensity of sparse linear algebra and efficient computation. A scalability analysis across MPI-only, MPI+OpenMP, and GPU architectures demonstrates the method's effectiveness in enhancing parallel efficiency and supporting large-scale simulations.

1. Introduction

The continuous development of novel numerical methods, coupled with the rapid evolution of high-performance computing (HPC) systems, has significantly expanded the role of computational fluid dynamics (CFD) in various industrial applications. Despite these advancements, the development of CFD faces persistent challenges. Early implementations were hindered by the compute-bound limitations of processors, which led to the adoption of compute-centric programming models. Over time, processor designs have evolved, addressing these limitations and resulting in a mismatch between computational power and memory bandwidth. This, in turn, forces the creation of complex memory hierarchies, complicating the optimization of traditional programs. At the same time, the widespread use of accelerators in diverse technological fields has driven the rise of hybrid architectures, offering greater computational throughput while improving power efficiency in large-scale applications [1]. However, this shift introduces a new challenge: ensuring the portability of legacy applications. This challenge requires versatile software architectures and the development of specialized APIs, such as CUDA, OpenCL, and HIP [2, 3].

In this context, the conservative discretization method for unstructured grids, as proposed by [4], has been adopted and implemented under TermoFluids Algebraic (TFA)—our in-house code based on an innovative algebra-dominant framework, HPC^2 [5]. This robust framework facilitates seamless integration into open-source codes [6] and hybrid supercomputing environments.

While computational power has improved, the time and resources required for detailed simulations remain a significant bottleneck. Achieving large-scale simulations is crucial for meeting industry demands for rapid decision-making, shorter product development cycles, and expanding CFD's industrial application fields. Thus, our research seeks to ensure the integration of modern CFD methodologies into industry practices, enabling precise and accurate simulations of complex processes while efficiently utilizing available resources and reducing simulation costs within limited timeframes.

2. Portability for CFD

The construction of codes based on a minimal set of algebraic kernels has become essential for ensuring portability, optimization, and ease of maintenance, particularly

† This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02469 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
marcial.francisco.mosqueda@upc.edu (M. Mosqueda-Otero); adel.alsalti@upc.edu (À. Alsalti-Baldellou); xavier.alvarezfarre@surf.nl (X. Álvarez-Farré); josep.plana.riu@upc.edu (J. Plana-Riu); guillem.colomer@upc.edu (G. Colomer); francesc.xavier.trias@upc.edu (F.X. Trias); asensio.oliva@upc.edu (A. Oliva)
ORCID(s): 0009-0003-8282-0300 (M. Mosqueda-Otero); 0000-0002-5331-4236 (À. Alsalti-Baldellou); 0000-0002-1684-7658 (X. Álvarez-Farré); 0000-0002-9086-0519 (J. Plana-Riu); 0000-0003-4592-0217 (G. Colomer); 0000-0002-5966-0703 (F.X. Trias); 0000-0002-2805-4794 (A. Oliva)

in light of the growing diversity of computational architectures and hardware vendors. The hybrid nature of modern high-performance computing (HPC) systems presents additional challenges, as effective utilization of both processors and parallel accelerators often requires heterogeneous computations and complex data exchanges. Traditional CFD codes, however, rely on intricate data structures and specialized computational routines, which complicates portability. To address this, algorithms centered on algebraic kernels, such as the sparse matrix-vector product (`SpMV`), linear combination of vectors (`axpy`), element-wise product of vectors (`axty`), and the `dot` product, emerge as promising solutions [5].

However, this approach introduces two primary challenges: (i) computational, particularly the low arithmetic intensity of the `SpMV` operation. This limitation can be mitigated by employing the more computationally intensive sparse matrix-matrix product (`SpMM`), which is advantageous in various scenarios such as matrices $\hat{\mathsf{A}} \in \mathbb{R}^{N\times N}$ that can be decomposed as the Kronecker product of a diagonal matrix, $\mathsf{C} \equiv diag(\boldsymbol{c}) \in \mathbb{R}^{K\times K}$, and a sparse matrix, $\mathsf{A} \in \mathbb{R}^{N/K\times N/K}$, *i.e.*, $\hat{\mathsf{A}} = \mathsf{C} \otimes \mathsf{A}$. The following transformation is achieved:

$$\boldsymbol{y} = \hat{\mathsf{A}}\boldsymbol{x} \implies \left(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\right) = \mathsf{A}\left(c_1\boldsymbol{x}_1, \ldots, c_K\boldsymbol{x}_K\right), \quad (1)$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{N/K}$. Substituting `SpMV` with `SpMM` in such cases significantly reduces memory access demands and the memory footprint by reusing matrix coefficients. (ii) Algorithmic challenges, such as redefining boundary conditions, can be naturally addressed within an algebraic framework using affine transformations, e.g.,

$$\boldsymbol{\varphi}_h \rightarrow \mathsf{A}\boldsymbol{\varphi}_h + \boldsymbol{b}_h, \quad (2)$$

which enables algebraic treatments suitable for both explicit and implicit time integration methods [7].

Beyond portability, algebra-based CFD implementations offer distinct numerical benefits. For example, they facilitate the development of efficient CFL-like conditions, reducing computational cost by up to 4x compared to classical approaches. The algebraic CFL method relies on constructing stable eigenvalue bounds during preprocessing, requiring only minimal vector updates over time [8]. Additionally, algebraic frameworks allow for the streamlined incorporation of advanced techniques such as flux limiters, yielding compact and efficient implementations by utilizing incidence matrices and local operations to control gradient ratios, thus reducing the number of computing kernels required for porting [9].

Building on these principles, the work in [10] proposed an effective approach to accelerate Poisson solvers by exploiting domain symmetries. By carefully ordering the unknowns, `SpMV` operations could be replaced with `SpMM`, leading to a 2.5x increase in the performance of compute-intensive kernels while significantly reducing the solver's memory footprint and setup costs. This enhancement highlights the potential of algebra-based methods for large-scale simulations on diverse HPC architectures.

3. Algorithm scalability analysis

The objective of the numerical experiments is to evaluate the performance and scalability of TFA's base algorithm under different parallel computing paradigms. Specifically, we compare the performance of an MPI-only configuration – where each CPU core is assigned a single task – against a hybrid MPI+OpenMP configuration, which utilizes 2 MPI processes and 56 multi-threaded executions per node. This hybrid approach is intended to reduce communication overhead by leveraging shared memory, which is particularly advantageous in multicore environments. Furthermore, the scalability of the code on hybrid HPC systems is analyzed, taking advantage of TFA's underlying structure, which is based on minimal algebraic kernels. This architecture enables broad portability across diverse GPU hardware using the OpenCL API.

The numerical test case solves a turbulent channel flow using a conjugate gradient solver with a Jacobi preconditioner for Poisson's equation, combined with an explicit time integration scheme and a variable time step. Since the primary objective is to assess the scalability of TFA+HPC^2 kernels, each case is limited to 10 time steps, with 800 solver iterations per step. The test problem is solved over the entire domain without exploiting symmetries, thereby isolating the key algebraic kernels – `SpMV`, `axpy`, `axty`, and `dot` product – for performance measurement and analysis.

Strong scalability tests were conducted using both MPI-only and MPI+OpenMP configurations on the MareNostrum 5 GPP supercomputer at BSC. The experiments were run on nodes equipped with two Intel Xeon Platinum 8480+ processors (56 cores, 2 GHz, 105 MB L3 cache, and 307.2 GB/s memory bandwidth) with 256 GB of RAM, interconnected via ConnectX-7 NDR200 InfiniBand. Additionally, GPU-accelerated tests were conducted on the Snellius supercomputer at SURF, utilizing nodes with two Intel Xeon Platinum 8360Y processors (36 cores, 2.4 GHz, 54 MB L3 cache, and 204.8 GB/s memory bandwidth), 512 GB of RAM and interconnected through dual ConnectX-6 HDR100 cards.

Figure 1 illustrates the strong scalability results for both parallel paradigms using two baseline configurations: (i) a single-node configuration (left plot) with a $305 \times 480 \times 350$

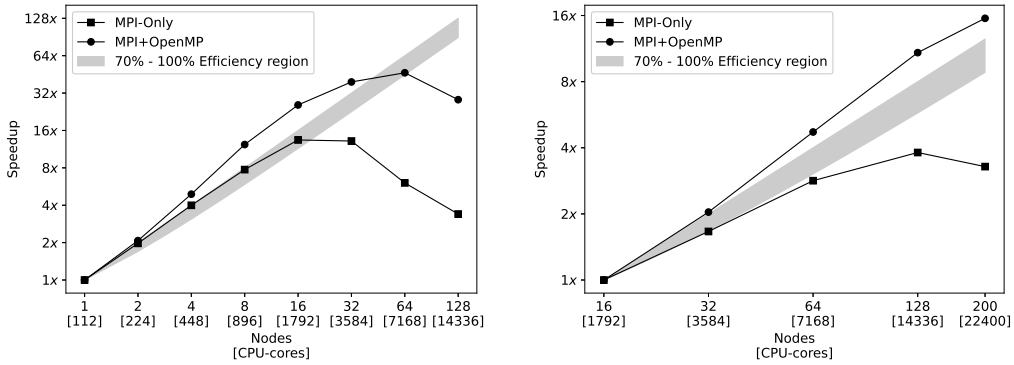


Figure 1: MPI-only vs. MPI+OpenMP strong scalability; with a 350×480×350 - 58.8M CVs - grid (left plot) and a 800×1470×800 - 940.8M CVs - grid (right plot).

grid and (ii) a 16-node configuration (right plot) with a 800× 1470×800 grid. Both configurations maintain a workload of 525k control volumes (CV) per CPU core. The results show a marked super-linear speedup for hybrid MPI+OpenMP processes, which can be attributed to enhanced cache utilization, a pattern consistently observed across both baselines.

Moreover, the MPI-only solution exhibits significant communication overhead, particularly when using 16 or more computational nodes in the single-node baseline, where a drop in performance is observed. In contrast, the 16-node baseline for the MPI-only configuration depicts a higher overhead due to the increased number of halo cells.

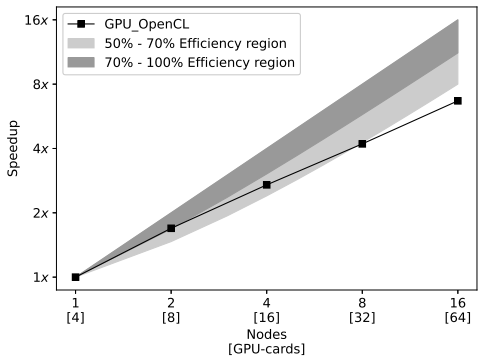


Figure 2: GPU-based strong scalability analysis on Snellius GPU island with a $400 \times 640 \times 400$ - 102.4M CVs - grid.

The strong scalability analysis for TFA+HPC[2] on a hybrid architecture was conducted on a domain of $400 \times 640 \times 400$, using a single-node configuration as the baseline on the Snellius GPU island. As shown in Fig. 2, a steady speedup is observed as the system scales. However, despite the memory-bound nature of CFD applications, the efficiency remains between 50% and 70% up to 8 nodes. For a 16-node implementation, the efficiency drops to approximately 45%, indicating the increasing impact of communication overhead in hybrid architectures as the scale increases.

Figure 3 presents the weak scalability analysis[1] for both MPI+OpenMP and GPU-based tests. The hybrid parallel paradigm (left plot) shows an 11% drop in performance when scaling up to 200 nodes, compared to the 16-node baseline. In contrast, the GPU-accelerated weak scalability (right plot) displays a significant initial drop in efficiency, approximately 10%, when transitioning from intra-node to inter-node execution. This performance decrease is attributed to the increased latency associated with inter-node communication. However, once scaling beyond two nodes, the GPU weak scaling stabilizes, showing only a minor decline in performance. At the 16-node scale, the efficiency drop remains modest at around 5%, indicating relatively consistent performance as the system scales across nodes.

[1]The main objective of this study is to assess the scalability of TFA+HPC[2] kernels. Thus, the number of iterations per time step was fixed to ensure a consistent number of kernel calls as the problem scales.

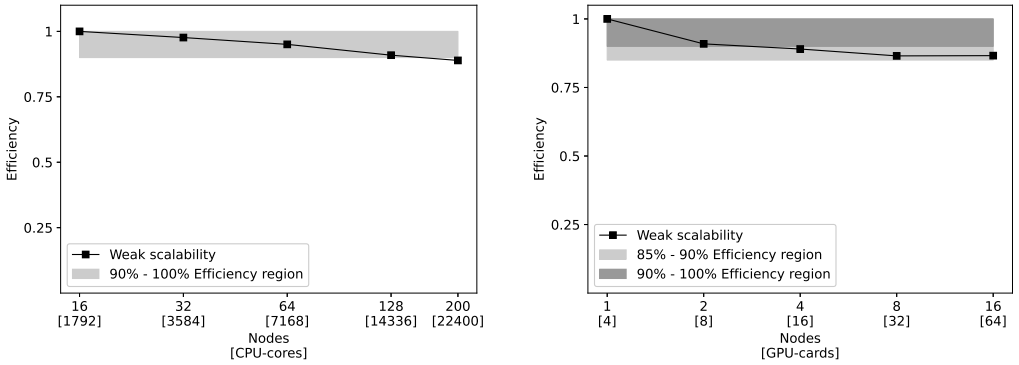


Figure 3: Weak scalability analysis; MPI+OpenMP paradigm with 525k CVs per CPU-core, starting with 16 nodes up to 200 nodes (left plot) and GPU-based implementation with 25.6M CVs per GPU card, starting with 1 node up to 16 nodes (right plot).

4. Performance analysis

In order to measure the implementation performance, an equivalent arithmetic intensity (AI_{eq}) and equivalent Performance (P_{eq}) were defined by applying a weighted average:

$$AI_{eq} = \frac{\sum_{k \in K} \alpha_k FLOPS_k}{\sum_{k \in K} \alpha_k BYTES_k}, \tag{3}$$

and

$$P_{eq} = \frac{\sum_{k \in K} P_k N_k}{\sum_{k \in K} N_k}, \tag{4}$$

where K is the set of kernels ($K = \{\texttt{SpMV}, \texttt{axpy}, \texttt{axty}, \texttt{dot}\}$), N_k and P_k corresponds with the number of operations and the performance of each kernel, respectively, while α_k, $FLOPS_k$, and $BYTES_k$ represents the operations ratio, the number of floating-point operations, and the number of memory transfers for each kernel, respectively.

Further, AI_{SpMV} is computed by the expression proposed in [10]:

$$AI_{SpMV} = \frac{2nnz(\mathsf{A}) + 1}{8nnz(\mathsf{A}) + 4nnz(\mathsf{A}) + 4(n+1) + 8n + 8m + 8}, \tag{5}$$

where $nnz(\mathsf{A})$, n and m correspond with the number of non-zeros, 7 in the current implementation, and the number of rows and columns of matrix A, respectively.

Figure 4 illustrates the roofline model analysis of TFA + HPC^2 performance on two distinct HPC architectures, highlighting their behavior in the memory-bound region. Both solutions demonstrate an equivalent arithmetic intensity (AI_{eq}) of approximately 0.125, with a noticeable gap between the theoretical peak performance (P_{peak}) of the supercomputers and the achieved performance (P_{eq}). Despite this, the results show that TFA+HPC^2 efficiently utilizes the available resources for its given arithmetic intensity, as the performance points for both architectures are located near the memory bandwidth limit. Notably, the GPU-based implementation (right plot) exhibits higher efficiency than the MPI+OpenMP configuration (left plot).

Furthermore, while both architectures are constrained by memory bandwidth, the GPU-accelerated system consistently outperforms the MPI+OpenMP solution at lower arithmetic intensities, benefiting from higher throughput and more optimized parallel execution routines. This highlights the advantage of GPU architectures in achieving better performance despite the inherent memory-bound limitations.

5. Closing remarks

The TFA+HPC^2 framework demonstrates strong portability across various HPC architectures, leveraging its minimal algebraic kernel design to ensure broad compatibility and efficiency. The MPI+OpenMP hybrid paradigm shows superior strong scalability over MPI-only, benefiting from better cache utilization and reduced communication overhead, making it ideal for large-scale CPU-based simulations. While the GPU-accelerated implementation is promising,

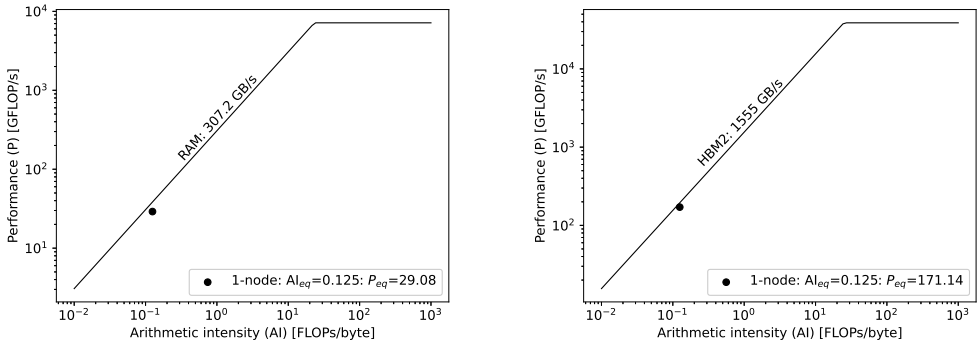


Figure 4: Roofline model; MPI+OpenMP paradigm on 1 node (112 CPU-cores) with a 350×480×350 - 58.8M CVs - grid solution (left plot) and GPU-based implementation on 1 node (4 GPU cards) with a $400 \times 640 \times 400$ - 102.4M CVs - grid solution (right plot).

it faces performance limitations due to inter-node communication overhead, indicating the need for increased computational load per GPU. Nonetheless, the current implementation is highly optimized, with performance primarily constrained by memory bandwidth.

Future work will focus on increasing the arithmetic intensity of TFA+HPC^2 to overcome its memory-bound limitations, e.g., by exploiting domain symmetries in large-scale urban simulations; replacing SpMV operations with SpMM to enhance solver performance. The weak scaling analysis shows excellent efficiency, confirming the framework's robustness and scalability for demanding industrial applications.

Acknowledgements

This work is supported by the Ministerio de Economía y Competitividad, Spain, SIMEX project (PID2022-142174OB-I00). M.MO. is supported by the Catalan Agency for Management of University and Research Grants (2024 FI-1 00684). In addition, J.PR is also supported by the Catalan Agency for Management of University and Research Grants (2022 FI_B 00810). Calculations were performed on the MareNostrum 5 GPP at BSC and Snellius supercomputer at SURF. The authors thankfully acknowledge these institutions.

References

[1] F. D. Witherden, A. M. Farrington, P. E. Vincent, Pyfr: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach, Computer Physics Communication 185 (11) (2014) 3028–3040. doi:10.1016/j.cpc.2014.07.011.

[2] H. Carter Edwards, C. R. Trott, D. Sunderland, Kokkos: Enabling manycore performance portability through polymorphic memory access patterns, Journal of Parallel and Distributed Computing 74 (12) (2014) 3202–3216. doi:10.1016/j.jpdc.2014.07.003.

[3] Y. Zhang, M. Sinclair, A. A. Chien, Improving performance portability in opencl programs, in: J. M. Kunkel, T. Ludwig, H. W. Meuer (Eds.), Supercomputing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 136–150. doi:10.1007/978-3-642-38750-0_11.

[4] F. X. Trias, O. Lehmkuhl, A. Oliva, C.D. Pérez-Segarra, R. W. C. P. Verstappen, Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, Journal of Computational Physics 258 (2014) 246–267. doi:10.1016/j.jcp.2013.10.031.

[5] X. Álvarez Farré, A. Gorobets, F. X. Trias, R. Borrell, G. Oyarzun, HPC^2 – A fully portable algebra-based framework for heterogeneous computing. Application to CFD, Computers & Fluids 173 (2018) 285–292. doi:10.1016/j.compfluid.2018.01.034.

[6] E. Komen, J. A. Hopman, E. M. A. Frederix, F. X. Trias, R. W. C. P. Verstappen, A symmetry-preserving second-order time-accurate PISO-based method, Computers & Fluids 225 (2021) 104979. doi:10.1016/j.compfluid.2021.104979.

[7] A. Alsalti-Baldellou, G. Colomer, J. A. Hopman, X. Álvarez-Farré, A. Gorobets, F. X. Trias, C. D. Pérez-Segarra, A. Oliva, Reliable overnight industrial LES: challenges and limitations. application to CSP technologies, in: 14th International

ERCOFTAC Symposium on Engineering, Turbulence, Modelling and Measurements, European Research Community on Flow, Turbulence, and Combustion (ERCOFTAC), 2023. URL https://www.fxtrias.com/docs/ETMM14_TF+HPC_vs_OF_abstract.pdf

[8] F. X. Trias, X. Álvarez-Farré, À. Alsalti-Baldellou, A. Gorobets, A. Oliva, An Efficient Eigenvalue Bounding Method: Cfl Condition Revisited (2023). doi:10.2139/ssrn.4353590.

[9] N. Valle, X. Álvarez-Farré, A. Gorobets, J. Castro, A. Oliva, F. X. Trias, On the implementation of flux limiters in algebraic frameworks, Computer Physics Communications 271 (2022) 108230. doi:10.1016/j.cpc.2021.108230.

[10] A. Alsalti-Baldellou, X. Álvarez-Farré, F. X. Trias, A. Oliva, Exploiting spatial symmetries for solving poisson's equation, Journal of Computational Physics 486 (2023) 112133. doi:10.1016/j.jcp.2023.112133.

Mini-Symposium 5:

HPC Biomechanics and New Challenges

Organizers: Cristóbal Samaniego, Beatriz Eguzkitza, Silvia Ceccacci, and Hadrien Calmet

Thanks to the utilization of massive computational resources, nearly all aspects of unsteady flow dynamics within the human body can now be accurately resolved. Numerical simulations have advanced to the point where they can effectively capture both spatial and temporal scales of airflow and blood circulation. HPC biomechanics encompasses a broad spectrum, ranging from the respiratory system to the cardiovascular system.

The multi-physics capabilities of supercomputers enable simulations that incorporate fluid-electro-mechanical coupling, such as modeling the interaction between a beating heart and arterial blood flow. These precise computational tools offer fresh perspectives and countless potential applications. It is evident that such methods represent the future of medical diagnosis and treatment processes.

In particular, the application of nasal/oral drug delivery holds significant importance in the respiratory system, presenting new challenges such as modeling the mucosa layer and simulating fluid-structure interaction within the nasal cavity. Additionally, given the previous impact of COVID-19 on public health, accurately understanding and modeling the propagation of human biological aerosols has become crucial.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

An Embedded Boundary Mesh Method for Simulating Rigid Heart Valves

Cristobal Samaniego[a,*], Mariano Vázquez[b] and Guillaume Houzeaux[a]

[a]Barcelona Supercomputing Center, Computer Applications in Science and Engineering (CASE) Department, 1-3 Plaça d'Eusebi Güell, 08034, Barcelona, Spain
[a]ELEM Biotech SL, Pier07 - Via Laietana, 26, 08034, Barcelona, Spain

ARTICLE INFO†

Keywords:
Fluid and Rigid Solid Interaction;
Embedded Mesh Method;
Parallel Algorithms Implementation

ABSTRACT

Unlike bioprosthetic valves, rigid heart valves provide exceptional durability, making them a compelling option to consider for long-term treatment. In this context, our objective is to demonstrate the capability of our numerical method in simulating the behavior of such rigid mechanical heart valves. From the physical point of view, the solution requires of two distinct yet interrelated physics: fluid dynamics and the behavior of solid components. Moreover, from the implementation perspective, a parallel implementation is an absolute necessity in order to reproduce the behavior of such rigid mechanical heart valves. For the numerical coupling strategy, we implement an embedded boundary mesh method. This approach involves discretizing the fluid using a non-body-conforming mesh, where the boundaries of the rigid bodies are embedded within the fluid mesh. The force exerted by the fluid on a body is determined from the residual of the momentum equations, while the body's velocity is imposed as a boundary condition in the fluid.

1. Introduction

Two main types of heart valve replacements are rigid mechanical valves and bioprosthetic valves. Characteristic as its durability and longer useful life make rigid heart valve replacements an important option to take into account. In particular, younger patients have the potential to benefit significantly from these properties, as described in [1].

Simulations of the behavior of real heart valves demands a considerable number of elements for fluid discretization. To tackle this challenge, we performed the implementation within the Alya system, using its computational fluid dynamics massively parallel solver. As a result, we can focus on the integration of two components into the Alya system: a computational rigid solid solver and an algorithm for solving the fluid and rigid body interaction using an embedded boundary mesh method.

2. Fluid

The physics of the fluid is described by the incompressible Navier-Stokes equations. Let μ be the viscosity of the fluid, and ρ its density. Additionally, let $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ be the velocity rate of deformation and the stress tensors, respectively, defined as:

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^t\right) \text{ and}$$
$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}).$$

The problem is stated as follows. Find the velocity $\boldsymbol{u}$ and mechanical pressure p in a domain Ω such that they satisfy in a time interval $(0, T]$:

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho\left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u} - \nabla\cdot\left(2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})\right) + \nabla p = \rho\boldsymbol{f} \quad \text{in } \Omega\times(0,T]$$
$$\text{and } \nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega\times(0,T]$$

together with initial and boundary conditions. The boundary conditions considered in this work are:

$$\boldsymbol{u} = \boldsymbol{u}_D \quad \text{on } \Gamma_D\times(0,T],$$
$$\boldsymbol{u} = \boldsymbol{u}_S \quad \text{on } \Gamma_S\times(0,T], \text{ and}$$
$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{t} \quad \text{on } \Gamma_N\times(0,T],$$

where Γ_D, Γ_S and Γ_N are the boundaries of Ω where Dirichlet, rigid body Dirichlet and Neumann boundary conditions are prescribed respectively, and $\partial\Omega = \overline{\Gamma_D\cup\Gamma_S\cup\Gamma_N}$. Note that the wet boundary of the solid Γ_S, and the associated prescribed solid surface velocity $\boldsymbol{u}_S$ will change in time. They are respectively the boundary and the variable used in the coupling with the rigid body.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02474 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
cristobal.samaniego@bsc.es (C. Samaniego); mariano@elem.bio (M. Vázquez); guillaume.houzeaux@bsc.es (G. Houzeaux)
ORCID(s): 0000-0002-3961-2261 (C. Samaniego); 0000-0002-2526-6708 (M. Vázquez); 0000-0002-2592-1426 (G. Houzeaux)

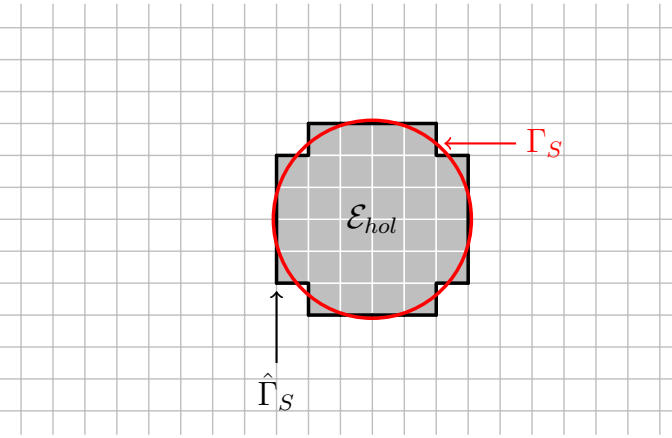


Figure 1: The set of hole elements, the solid boundary Γ_S, and the approximation of the solid boundary on the fluid mesh $\hat{\Gamma}_S$ in the context of a simple structured mesh.

3. Rigid solid

The linear acceleration $\boldsymbol{a}(t)$ and angular acceleration $\boldsymbol{\alpha}(t)$ of the body at a time t are related to the input force $\boldsymbol{f}_F(t)$ and input torque $\boldsymbol{\tau}_F(t)$ by the Newton-Euler equations:

$$\boldsymbol{f}_F(t) = m\boldsymbol{a}(t) \tag{1}$$

and

$$\boldsymbol{\tau}_F(t) = \boldsymbol{I}(t) \cdot \boldsymbol{\alpha}(t) + \boldsymbol{\omega}(t) \times (\boldsymbol{I}(t) \cdot \boldsymbol{\omega}(t)), \tag{2}$$

where m is the total mass of the body, $\boldsymbol{\omega}(t)$ the angular velocity, and $\boldsymbol{I}(t)$ is the inertia tensor. By integrating in time the Equations (1) and (2), the velocity and the position of the rigid body can be determined.

4. Fluid and rigid body interaction for an embedded boundary mesh method

The main steps of the algorithm for solving the interaction consist of four parts, as outlined below.

- **The hole elements identification**. First, the program identifies the elements whose volumes of intersection with the rigid body domain are sufficiently large to consider them as part of the solid. They will be excluded from the finite element assembly process. Let this set of elements be called as the set of hole elements $\mathcal{E}_{hol}$, represented by gray squares as shown in Fig. 1.
- **The fringe nodes identification**. The exclusion of the assembly process of the set $\mathcal{E}_{hol}$ defines a new internal boundary $\hat{\Gamma}_S$ in the fluid mesh, the bold black line in Fig. 1. This internal boundary can be seen as an approximation on the fluid mesh to the actual solid boundary surface Γ_S, the red line in Fig. 1. Let the nodes that that define $\hat{\Gamma}_S$ be called as the set fringe nodes: $\mathcal{N}_{fri}$. This set allow us to define other important sets of nodes: the set of free $\mathcal{N}_{fre}$ and the set of hole nodes $\mathcal{N}_{hol}$. The set of free nodes will be belong to the fluid mesh domain and the set of hole nodes will be considered as part of solid mesh domain.
- **The imposition of the rigid body velocity on the fluid mesh**. The velocity of the rigid solid is imposed on the set $\mathcal{N}_{fri}$ in an interpolating way using the universal kriging methodology.
- **The fluid force exerted on the rigid solid**. The force exerted by the fluid on the solid, which determines its new position inside the fluid mesh, is calculated using the residual of the momentum equations corresponding to the set $\mathcal{N}_{fri}$.

Some of the implementation details of the embedded mesh boundary methods described next in this work was published previously in [2].

5. Numerical results

To demonstrate the capability of our numerical method in simulating the behavior of such rigid mechanical heart valves a real experiment described in [3] is considered. In the experimental setup, a bileaflet mechanical heart valve was placed within a rigid artificial aorta, and an inflow boundary condition was prescribed as a plug flow profile (forward and reverse), facilitating the opening and closing of the two valves. The REYNOLDS number of the flow varies from 0, when the valves are closed, to nearly 6000 when the valves are fully open.

The velocity in the x-direction and vorticity obtained during the simulation at the plane of symmetry can be used to visualize the motion of the leaflets during the opening and closing phases, as shown in Fig. 2 and 3, respectively.

The vorticity field reveals that during the opening phase of the valves (see Fig. 3, left), the flow field remains laminar. During the closing phase (see Fig. 3, right), the vorticity field is characterized by a recirculation zone generated near the sinus root walls.

With respect to both the velocity field and vorticity, the numerical simulation successfully reproduces the key behaviors of the rigid mechanical heart valves described by Dasi et al. [3].

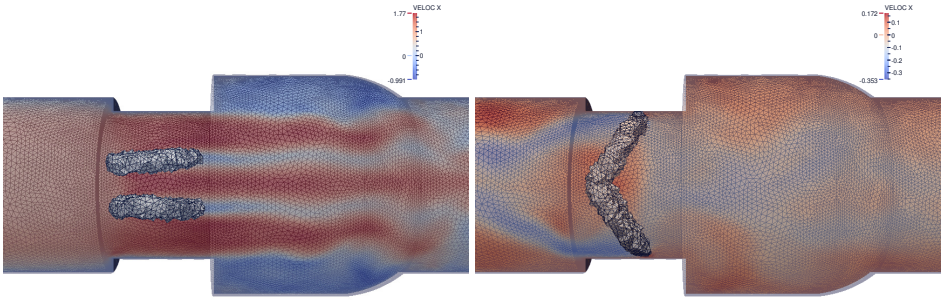


Figure 2: Velocity field around the valves when closing and opening.

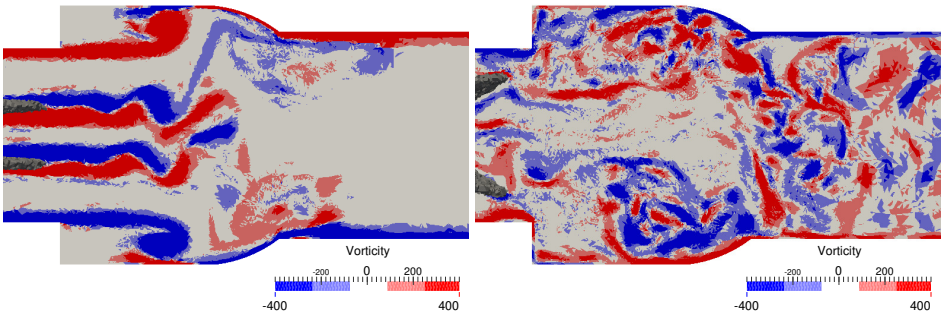


Figure 3: Vorticity around the valves when closing and opening.

6. Conclusions

Simulation results demonstrate that the method presented here is capable of replicating real experimental outcomes. These promising findings suggest a future where simulations could play a crucial role in assisting patient diagnosis.

References

[1] R. P. Whitlock, G. R. McClure, J. W. Eikelboom, Aortic valve replacement in younger patients, European Heart Journal 38 (45) (2017) 3378–3381. doi:10.1093/eurheartj/ehx367.

[2] C. Samaniego, G. Houzeaux, E. Samaniego, M. Vázquez, Parallel embedded boundary methods for fluid and rigid-body interaction, Computer Methods in Applied Mechanics and Engineering 290 (2015) 387–419. doi:10.1016/j.cma.2015.03.008.

[3] L. P. Dasi, L. Ge, , H. A. Simon, F. Sotiropoulos, A. P. Yoganathan, Vorticity dynamics of a bileaflet mechanical heart valve in an axisymmetric aorta, Physics of Fluids 19 (2007) 1–17. doi:10.1063/1.2743261.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Parallel Mesh Developments to Prepare for Biosimulations

Phoebe Cullen[a,b], Charles Moulinec[a,*], James Gebbie-Rayet[a], Jérôme Bonelle[c] and Yvan Fournier[c]

[a] UKRI STFC Daresbury Laboratory, Scientific Computing, Sci-Tech Daresbury, WA4 4AD, UK
[b] University of Leeds, Woodhouse, Leeds, LS2 9JT, UK
[c] EDF R&D, MFEE, 6 quai Watier, 78400, FR

ARTICLE INFO†

Keywords:
Mesh Optimization;
Compatible Discrete Operator

ABSTRACT

A parallel mesh procedure is developed to create a single mesh from 2 or more meshes, in case proteins would merge together. The various steps of the algorithm are presented in details, before some preliminary results are shown using the Compatible Discrete Operator method.

1. Introduction

Biomolecular simulations are critical to understanding the highly dynamic environments within biological cells. Biosimulations work in conjunction with imaging techniques to relate static experimental structures to functionality. For large time and length scales, continuum mechanics has been proposed for use in a coarse-grained approach. Fluctuating Finite Element Analysis (FFEA) [1] is such a method. It uses a mesh-based approach in place of a particle-based one, i.e., computing bulk quantities as opposed to atomic positions. Protein interactions are fundamental to biological mechanisms that underpin life. They can interact in a number of different ways, for instance by sticking to each other, where mesh 'subsurfaces' temporarily glue together, or by merging together. The latter is a permanent interaction, where 2 meshes join together by a 'bridge'. Sticking is the more frequent phenomenon in biology, but merging is more challenging in terms of algorithm, and as such is considered here.

This work presents the parallel algorithm used to merge 2 or more meshes (see Section 2). Some preliminary results are shown for a steady 3-D vector-valued diffusion equation computed using the Compatible Discrete Operator (CDO) method [2] implemented in code_saturne[1] [3] (see Section 3), before some conclusions are drawn (see Section 4).

2. Principle of the Method

The merging algorithm is split into several operations. Figure 1 shows the steps to build a 'bridge' between 2 neighboring meshes, before collapsing into a single body. The principle of this method is to use a plane (in red in Fig. 1), equidistant from 2 meshes' centers of gravity, as an interface for the projection, extrusion and finally merging of each pair of meshes. All the steps following the computation of the plane orientation are local to each mesh, until the 2 projected 'subsurface' footprints are glued together at the plane. The footprint boundary faces are then changed into inner faces, in the final step of the method. The method is

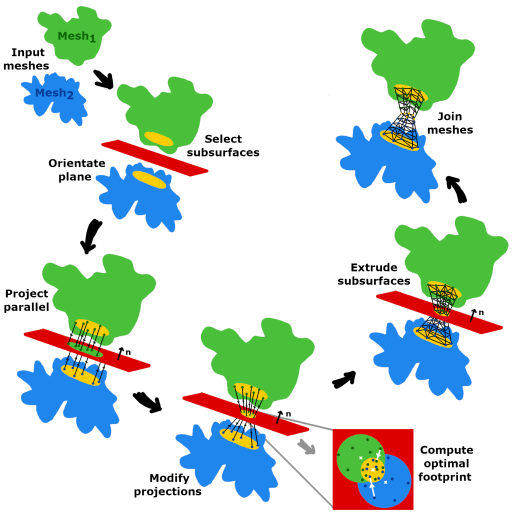


Figure 1: Merging procedure.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02475 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
cm20pic@leeds.ac.uk (P. Cullen); charles.moulinec@stfc.ac.uk (C. Moulinec); james.gebbie@stfc.ac.uk (J. Gebbie-Rayet); jerome.bonelle@edf.fr (J. Bonelle); yvan.fournier@edf.fr (Y. Fournier)
ORCID(s): 0009-0005-1781-5323 (P. Cullen); 0009-0003-7011-7327 (C. Moulinec); 0000-0001-8271-3431 (J. Gebbie-Rayet); 0000-0002-8164-4646 (J. Bonelle); 0000-0001-5271-007X (Y. Fournier)

[1] https://www.code-saturne.org/cms/web

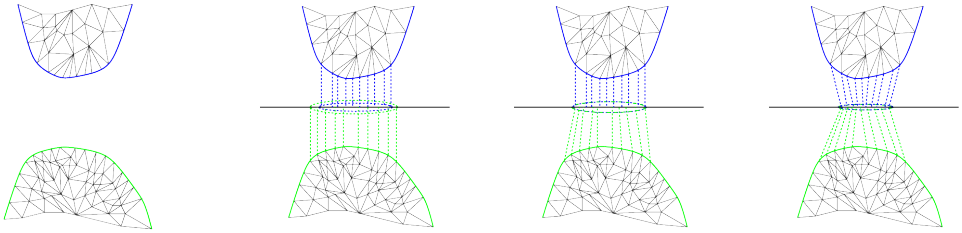


Figure 2: Sketch of the combined various steps of the projection procedure.

easily generalized to many concurrent merging occurrences between 2 individual meshes, and fully parallel.

2.1. Projection procedure

The projection procedure is the pivotal step of this algorithm (see Fig. 2). Its steps are enumerated here:

1. Construct/import the meshes to be merged
2. Select the local boundary 'subsurfaces' to be projected
3. Build the plane from the distance between the meshes
4. Perform initial projections onto the plane, parallel to the plane normal
5. Find the optimal projection footprints, as the 2 footprints might not be conformal
6. Define vectors between 'subsurfaces' and average footprint vertices
7. Re-project the 'subsurface' vertices, along these new vectors

2.2. Extrusion and interface gluing

For each mesh, an extrusion is computed between the 'subsurface' and the average footprint, creating layers of prisms. The average footprint is generally non-conformal, and the code_saturne embedded gluing algorithm is activated to get it conformal, creating new faces on both sides of the plane. This results in a layer of polyhedral cells on both sides of the plane, the average footprint faces being made inner faces.

2.3. Comments on the current implementation and potential improvements

- In this work, the location of 2 or more meshes is set a priori, and the distance between them computed from the gravity centers of the interacting objects. This is not optimal, in general, but an iterative process could be implemented to compute the minimum 'physical' distance, starting from the existing technique, based on the centers of gravity of the individual meshes.
  In the future, information could be obtained from Molecular Dynamics (MD), when the distance between 2 proteins is below a given threshold, depending on the studied case. MD could then inform whether the proteins are attracting or repulsing each other. In the former case, the merging algorithm would be activated, otherwise, the whole simulation would continue to compute the protein evolution.
- The position of the subsurface center depends on the 2 end points of the segment linked to the distance. A cylinder, which axis' center is one of these ends and direction's is based on the segment's, is used to select the subsurface elements, its radius being approximated from the shape of the physical boundaries of the protein.
- In order to get a workable number of layers to be used by each extrusion, information is gathered from the volumic elements the subsurface belongs to, focusing on their edge size.

3. Preliminary Results

The projection procedure has successfully been implemented into code_saturne, including the 'subsurface' extrusions, though some final developments are required to get the full algorithm in place. Figure 3 (left) shows the example of a 'bridge' merging between 2 cubes, and Fig. 3 (right) of 2 'bridge' merging between 3 ellipsoids. The newly created

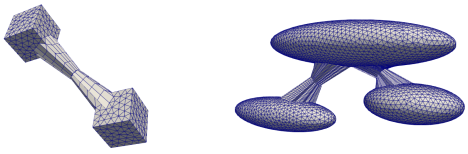


Figure 3: Examples of merging for 2 cubes (right) and 3 ellipsoids (left).

meshes will be used as supports to simulate 3-D vector-valued diffusion equations using the Compatible Discrete Operator approach. To demonstrate the performance of the

method, a simulation is carried out in a cubic box, with zero-Dirichlet boundary conditions in all 3 directions. The mesh is made of a mix of tetrahedral (inner mesh) and prismatic (at the wall) elements to assess the ability of the CDO approach for mixed element meshes. Figure 4 (top)

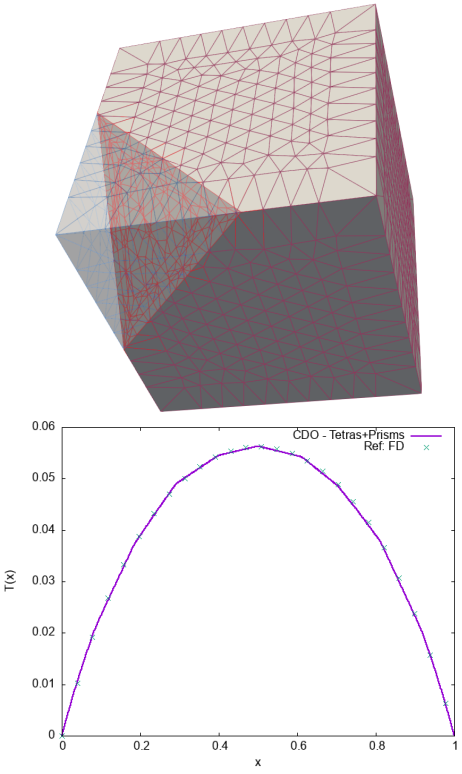


Figure 4: Top: Sketch of the cubic mesh with tetrahedral cells in the inner domain and prism layers at each wall. Bottom: CDO solution against a reference produced by a Finite Difference code.

shows a clip of the mesh, with tetrahedral and prismatic elements, and Fig. 4 (bottom) the profile of the solution in 1 direction (the test is such that all 3 directions show the same solution), obtained using the CDO method, and compared to a reference produced by a Finite Difference (FD) code ran on a refined homogeneous mesh. The CDO solution is very closed to the FD reference, despite the use of a very coarse grid. Figure 5 shows a clip of a 3-D array of 256 ellipsoids. For each of them, a steady vector-valued diffusion equation is computed by CDO in 1 single instance of code_saturne. The ellipsoidal shape is used to make the geometry more

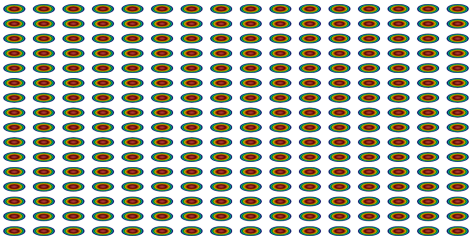


Figure 5: Simulation using 256 separate ellipsoids.

general and concurrent 'bridges' will be computed between neighboring ellipsoids. The same equation will be solved on the resulting single mesh.

4. Conclusions

A merging algorithm between 2 or more meshes has been presented, and its projection and extrusion procedure have been implemented into code_saturne. A 3-D steady vector-valued diffusion equation has been computed as a reference case to show the accuracy of the Compatible Discrete Operator method when using tetrahedral and prismatic elements. Furthermore, it has been shown that one instance of code_saturne's solver is able to compute 256 independent ellipsoidal meshes. More results will be presented for the full merging algorithm applied to 256 ellipsoids. Parallelization will also be explained in detail and timing of the merging procedure will be given, as well as results on the performance of the algorithm.

References

[1] A. Solernou, B. Hanson, R. Richardson, R. Welch, D. Read, O. Harlen, S. Harris, Fluctuating Finite Element Analysis (FFEA): A continuum mechanics software tool for mesoscale simulation of biomolecules, PLoS 14(3) (2018). doi:10.1371/journal.pcbi.1005897.

[2] J. Bonelle, Y. Fournier, C. Moulinec, New polyhedral discretisation methods applied to the Richards equation: CDO schemes in Code_Saturne, Computers & Fluids 173 (2018) 93–102. doi:10.1016/j.compfluid.2018.03.026.

[3] Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. Sunderland, J. Uribe, Optimizing Code_Saturne computations on Petascale systems, Computers & Fluids 45 (2011) 103–108. doi:10.1016/j.compfluid.2011.01.028.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Comparison of Airflow and Particle Deposition in Different Acinus Geometries

Ane Beatriz Eguzkitza[a,*], Carlos Arnedo[b], Filippo Nassetti[c], Jernimo Calderon[a], Fernando Muñoz[d] and Guillaume Houzeaux[a]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain
[b]Universidad Autónoma de Barcelona, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain
[c]Urban Morphogenesis Lab at UCL, The Bartlett School of Architecture, 22 Gordon Street, WC1H 0QB London, UK
[d]Universitat Politècnica de Catalunya, 1-3 Carrer de Jordi Girona, Les Corts, 08034 Barcelona, Spain

ARTICLE INFO†

Keywords:
Computational Fluid Dynamics;
Particle Dynamics;
High-Performance Computing;
Respiratory System

ABSTRACT

This work is one of three components (upper airways, lower conducting airways, and respiratory zones) of a digital twin lung model developed by the Physical and Numerical Modelling research group at the CASE department of the Barcelona Supercomputing Center (BSC). To the best of the authors' knowledge, the challenge of accurately computing the local effective dose at the site of action remains unresolved. Our group's aim is to simulate patient-specific aerosol transport and deposition using physics-based 3D computational fluid particle dynamics (CFPD) models that account for airway or alveolar tissue pathology. In this study, we focus on the particularly challenging respiratory or alveolar zone of the lung. We present a novel in-silico approach with unparalleled spatio-temporal resolution, enabling the tracking of aerosol particles throughout the entire respiratory cycle within both the conducting airways and the alveolar region. Among the analyzed geometries, Mesh 4 has demonstrated superior performance in replicating realistic particle deposition patterns, aligning closely with segmentation images obtained through scanning electron microscopy. This result highlights the potential of Mesh 4 for advancing physiologically accurate modeling of the human pulmonary acinus. The key unprecedented contribution of this work lies in its realistic geometry for the acini and the distal anatomical structure of the respiratory lung zone, including the subacini. The team focused on modeling the alveolar internal space, where the gases are able to flow. The first step was defining a 3-dimensional, branching arrangement of lines representative of the logic by which bronchioles transition into alveolated airways. This diagram was then interpreted as a substrate for the distribution of spherical objects, indicative of single alveolar sacs. These spherical objects were finally merged into a unified, continuous geometry representative of the volumetric alveolar space. The specific context of the current work is framed within the European Project CREXDATA. Its general vision is to develop a generic platform for real-time critical situation management including flexible action planning and agile decision making over streaming data of extreme scale and complexity. One of the use cases of the project is the COVID-19 pandemic crisis, studying at the microscopic level viral evolution for forecasting emerging mutations of clinical relevance. To that end, the first step is to develop a mechanistic multi-scale model to build a toolbox aimed at having a digital twin for the treatment of patients.

1. Introduction

Existing 3D CFPD models struggle to accurately compute aerosol transport and deposition in the entire human lung, particularly with pathologies. Airflow in the pulmonary acinus is primarily laminar, characterized by low REYNOLDS numbers and the absence of turbulence, which aids in predicting particle transport through the airways [1]. Aerosols ranging from 0.001 to 10 μm can reach these regions, with their behavior influenced by sedimentation, convection, and diffusion [2]. While gravity significantly affects larger particles, smaller ones are subject to Brownian motion.

To address these limitations, we have developed a novel model to simulate fluid flow and transported particles for the respiratory zone of the lung. Our approach accounts for both geometric and dynamic similarities, ensuring accurate

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02476 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
beatriz.eguzkitza@bsc.es (A.B. Eguzkitza);
carlosarnedo19@gmail.com (C. Arnedo); filippo@filipponassetti.com (F. Nassetti); jcaldero@bsc.es (J. Calderon);
fernando.munoz@estudiantat.upc.edu (F. Muñoz);
guillaume.houzeaux@bsc.es (G. Houzeaux)
ORCID(s): 0000-0002-3302-6667 (A.B. Eguzkitza);
0009-0008-0448-167X (C. Arnedo); - (F. Nassetti); - (J. Calderon);
0009-0003-6182-1347 (F. Muñoz); 0000-0002-2592-1426 (G. Houzeaux)

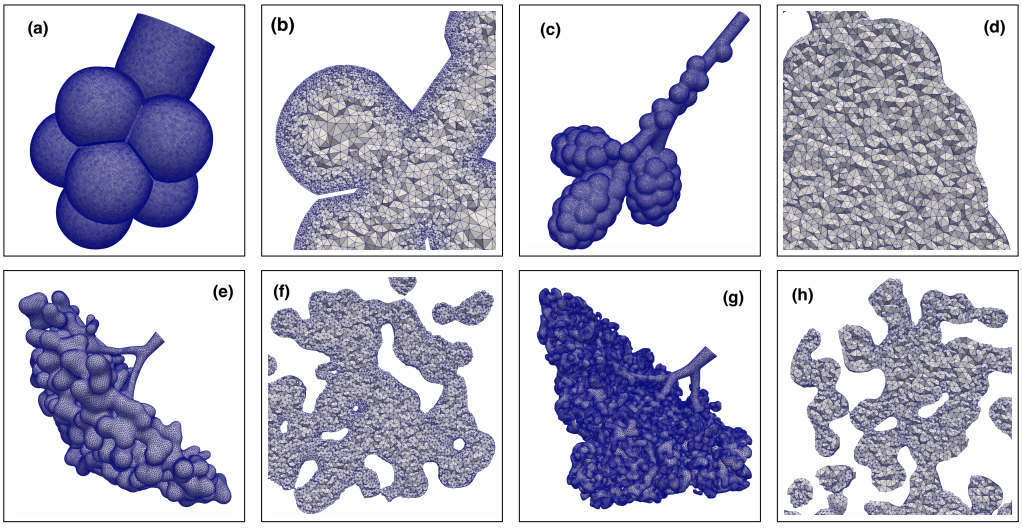


Figure 1: (a, c, e) Three dimensional geometries representing the more distal region of the respiratory zone, the alveolar sacs; (b, d, f) mesh visualization of the interior of each of the geometries, respectively. (g) Three dimensional geometry representing a sub-acinus, one of the sub-units of the acinus; (h) and its interior mesh visualization.

representation of flow regimes critical to particle deposition [3]. We reconstructed alveolated structures across different length scales, allowing an evaluation of the influence of geometry and volume on deposition patterns.

The results obtained lay the foundation for addressing the major challenge of drug delivery to human lungs: evaluating the dose delivered to the local site of action in the respiratory zone, which is crucial for treating diseases like COPD. Additionally, these findings are significant for understanding the progression of infections caused by bio-aerosols, such as viruses or bacteria, within the human lungs.

2. Methodology

To describe the methodology of the present work we divide this section into separate subsections, i.e., Sec. 2.1 describes the geometry and mesh, and Sec. 2.2 the physical and numerical methods.

2.1. Geometry and mesh description

Due to the tiny dimensions of the ducts and alveoli within the respiratory zone of the lung, obtaining geometries of these regions through medical imaging has become a extreme challenge not resolved until now. Consequently, all geometries are synthetically generated. This work presents a range of scales for such complex structures, based on different approaches, ranging from purely geometric to more anatomically accurate representations. The scaling of these geometries was adjusted based on the alveolar diameters from [4]. Geometries used for meshes 3 and 4 have been developed under the ST+ARTS AIR project[1], by making use an space colonization algorithm combined with medical images from the lung's respiratory zone.

High-quality mesh discretization was performed to ensure numerical stability and accurate simulation results. The mesh generation software ANSYS-ICEM-CFD (Ansys Inc., USA) was used. An octree-based method generated a fine surface mesh, followed by volumetric meshing using the Delaunay method. To improve the quality of the meshes, techniques like Laplace smoothing [5] were applied, ensuring well-shaped elements that conform to the complex domain geometry. Given the laminar nature of airflow in the

[1] https://starts.eu/air/

Mesh	No. elem.	Elem. type	Boundaries	Nodes
1	2857071	TET04	329456	560414
2	1195847	TET04	93116	223957
3	1102221	TET04	174998	229220
4	8090162	TET04	1292494	1684917

Table 1
Summary of different mesh features.

acinar airways, prism layers near boundaries were unnecessary, simplifying the meshing process to use exclusively TET04 tetrahedral elements. In Fig. 1, the generated meshes are displayed, while in Tab. 1, the features of the different meshes are shown.

2.2. Physical and numerical method

The work presented has been implemented and achieved using an in-house code, Alya [6]. It is a high performance computational mechanics code to solve complex coupled multi-physics problems. This software has been developed by the CASE Department of the Barcelona Supercomputing Center. It is a modular scientific code written in Fortran and parallelized with MPI that is broken into multiple modules.

2.2.1. Fluid Solver

To simulate the airflow inside the pulmonary acinus, we use the incompressible Navier-Stokes equations combined with the Approximate Lagrangian-Eulerian (ALE) formulation to emulate Fluid-Solid Interaction (FSI). The fluid conservation laws are expressed in a moving Eulerian domain, with the governing momentum and continuity equations, i.e.,

$$\rho_f \frac{\partial \mathbf{u}_f}{\partial t} + \rho_f[(\mathbf{u}_f - \mathbf{u}_m)\cdot\nabla]\mathbf{u}_f - \nabla\cdot[2\mu\boldsymbol{\epsilon}(\mathbf{u}_f)] + \nabla p = \rho_f \mathbf{f}, \quad (1)$$

$$\nabla\cdot\mathbf{u}_f = 0. \quad (2)$$

Here, μ is the fluid's viscosity, ρ_f its density, $\mathbf{u}_f$ its velocity, p the mechanical pressure, $\boldsymbol{\epsilon}(\mathbf{u}_f)$ the strain rate tensor defined as $\boldsymbol{\epsilon} = (1/2)\cdot(\nabla\mathbf{u}_f + \nabla\mathbf{u}_f^T)$, $\mathbf{f}$ the body force term, and $\mathbf{u}_m$ the mesh velocity. In our implementation, $\mathbf{u}_m$ is derived from the mesh displacement at the boundaries, $\mathbf{d}_m$, which is calculated by solving a diffusion equation

$$\nabla\cdot[c_m\nabla\mathbf{d}_m] = 0, \quad (3)$$

where $\mathbf{d}_m$ the node displacement and c_m is a diffusion coefficient, computed element-wise to control element stiffness [7]. In this work, the displacement function is a modified expression from [8]

$$d_m(t) = \frac{d_{\max}}{2}\left[1 - \cos\left(\frac{2\pi}{T_{\text{cycle}}}t\right)\right], \quad (4)$$

where $d_{\max}$ is the maximum displacement of the mesh and T_{cycle} is the breathing period. To discretize the incompressible Navier-Stokes equations, the VMS method is used [9], and for mesh displacements with the ALE formulation, see [10]. The resulting linear system at each time step is given by

$$\begin{bmatrix}\mathbf{A}_{uu} & \mathbf{A}_{up}\\ \mathbf{A}_{pu} & \mathbf{A}_{pp}\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{p}\end{bmatrix} = \begin{bmatrix}\mathbf{b}_u\\ \mathbf{b}_p\end{bmatrix}, \quad (5)$$

where $\mathbf{u}$ and $\mathbf{p}$ are velocity and pressure unknowns.

Numerical treatment. The time discretization is based on a second-order Backward Finite Difference (BFD) scheme. To solve the coupled algebraic system in Eq. (5), we apply a split approach using the pressure Schur complement system [11].

The momentum equation is solved using the GMRES method, with a diagonal preconditioner. The continuity equation is solved using the Deflated Conjugate Gradient (DCG) method [12], together with a diagonal preconditioner.

Boundary conditions. Regarding the boundary conditions for the fluid, a predefined flow rate could not be specified because the meshes contain only a single surface through which the fluid flows in a periodic motion. To enable fluid entry, the mesh boundaries must move, thereby increasing the internal volume. Using Eq. (4), a normal displacement was applied to the mesh nodes to induce flow from the inlet. Consequently, the fluid velocity at the walls matches the velocity induced by the described boundary displacement. To emulate the typical flow rates found in the acinar region, as described in the literature, different maximum normal displacements were applied: $d_{\max} = 10, 15, 20, 25\ \mu m$, occurring over half of the breathing period $T_{\text{cycle}} = 4$ seconds. The inlet was fixed.

2.2.2. Particle Solver

In order to solve particle transport in Alya, a one-way coupling is employed. This means particles are transported using the previously obtained airflow solution. Particle transport is solved in a Lagrangian frame of reference, each particle is tracked along its trajectory individually. The model has a few assumptions:

- Particles are small enough to neglect their effects on airflow, i.e., one-way coupling;
- Particles are spherical and don't interact with each other;
- Particle rotation is negligible;
- Thermophoretic forces are neglected;
- The forces considered are: drag $\boldsymbol{F}_d$, gravitational and buoyancy $\boldsymbol{F}_g$, and the force due to Brownian diffusion $\boldsymbol{F}_B$;

Let $\mathbf{x}_p$, $\mathbf{u}_p$ and $\mathbf{a}_p$ be the position, velocity and acceleration of each particle, respectively. Let m_p be its mass, ρ_p its density, d_p its diameter and V_p its volume. Particle trajectories are obtained by solving Newton's second law, with the forces considered,

$$\mathbf{a}_p = \frac{1}{m_p}(\mathbf{F}_g + \mathbf{F}_d + \mathbf{F}_B). \quad (6)$$

The equation for the drag force assumes the particle has reached its terminal velocity and is given by

$$\mathbf{F}_d = -\frac{\pi}{8}\mu d_p \frac{C_d}{C_c} Re_p(\mathbf{u}_p - \mathbf{u}_f), \tag{7}$$

where Re_p is the particle REYNOLDS number involving its relative velocity with the fluid

$$Re_p = \frac{\rho_f |\mathbf{u}_p - \mathbf{u}_f| d_p}{\mu}, \tag{8}$$

and C_d and C_c are the drag coefficient and the Cunningham slip correction factor, respectively. The Cunningham slip correction factor [13] is an empirical correction applied to account for non-continuum effects that arise due to the small size of the particles relative to the mean free path of the gas molecules. This factor is crucial for accurately calculating the drag force and Brownian force on the particles studied. It is defined as:

$$C_c = 1 + \frac{2\lambda}{d_p}\left[1.257 + 0.4\exp\left(\frac{-1.1 d_p}{2\lambda}\right)\right], \tag{9}$$

here λ is the mean free path of the air molecules, 68 nm in our case. With respect to the drag force we have considered Cheng's model [14], where the drag coefficient is given by

$$\begin{aligned} C_d &= \frac{24}{Re_p}\left(1 + 0.27 Re_p\right)^{0.43} \\ &\quad + 0.47\left(1 - \exp\left(-0.04 Re_p^{0.38}\right)\right). \end{aligned} \tag{10}$$

The gravity and buoyancy forces contribute to the dynamic of the particle whenever exists a density difference, i.e.,

$$\mathbf{F}_g = V_p \mathbf{g}(\rho_p - \rho_f), \tag{11}$$

with $\mathbf{g}$ being the gravity vector. The particle density is mostly considered to be 1 g/cm^3, which is about 1,000 times the density of air [15]. Finally, the stochastic force due to a particle's Brownian motion $\mathbf{F}_B$ is expressed as [13]

$$\mathbf{F}_B = \xi\sqrt{\frac{\pi S_0}{\Delta t}}, \tag{12}$$

where ξ is a random vector following a normal Gaussian distribution with zero mean and unit variance, Δt is the time step size in which the amplitudes of the Brownian force components are evaluated, in our case $\Delta t = 10^{-3}$, and S_0 corresponds to the spectral density of the white noise process

$$S_0 = \frac{216\mu k_B T}{\pi^2 \rho_f d_p^5 \left(\frac{\rho_p}{\rho_f}\right)^2 C_c}, \tag{13}$$

where k_B is the Boltzmann constant and the temperature of the airways is T, in our case we have considered 36 °C [16].

Numerical treatment: particle dynamics and distance to the wall. To obtain particle's dynamics in each time step the Newmark method is used. Consider that the particle path must be computed from time step n to the next one $n+1$, where the time step size is defined as $\Delta t := t^{n+1} - t^n$. This method consists on obtaining particle's position and velocity after obtaining the acceleration on timesteps n and $n+1$, solving the following equations:

$$\begin{cases} \mathbf{a}_p^{n+1} = \mathbf{F}^{n+1}/m_p, \\ \mathbf{u}_p^{n+1} = \mathbf{u}_p^n + \Delta t(\gamma \mathbf{a}_p^{n+1} + (1-\gamma)\mathbf{a}_p^n), \\ \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \mathbf{u}_f^n \Delta t + \frac{1}{2}\Delta t^2(2\beta \mathbf{a}_p^{n+1} + (1-2\beta)\mathbf{a}_p^n). \end{cases} \tag{14}$$

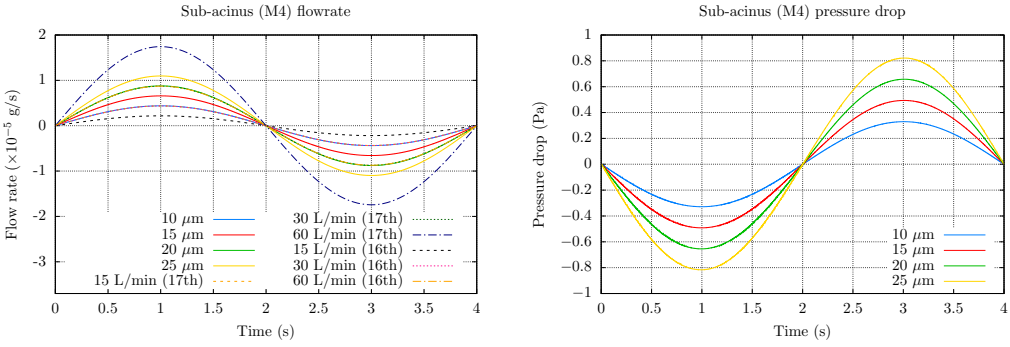
The Newton-Raphson method is used iteratively to solve for the implicit dependence, ensuring that the updated accelerations, velocities, and positions are consistent at each time step. However, due to the low mass of nanoparticles, they experience very high accelerations. Their extremely short relaxation time means they rapidly respond to changes in the surrounding air velocity, needing a significant number of iterations to accurately capture their dynamics. Consequently, the simulation often failed to reach the required number of iterations before hitting the imposed limit, leading to the disregard of most injected particles. To address this, the relaxation factor is updated at each iteration to control the velocity update. If the residual indicates poor convergence, the relaxation factor is reduced by 10% to potentially improve the stability and convergence of the iterative process. Recall that the fluid's velocity is present when updating the position. The values used in our case are $\beta = 0.25$ and $\gamma = 0.5$, values at which the numerical method is unconditionally stable, i.e., always stable.

To compute the distance to the wall for the particles, the iterative solver used is the Deflated-Conjugate Gradient method with a diagonal preconditioner.

Boundary Conditions. Particles of various sizes were injected during the inhalation period over 3 breathing cycles, with sizes ranging from 1 nm to 10 μm, based on [17], at intervals of $5 \cdot 10^{-3}$ s. For meshes M1, M2, and M3, 100 particles of each type were injected every five time steps, while for M4, 250 particles were injected also each five time steps. This resulted in a total of $1.2 \cdot 10^6$ particles being injected into meshes M1, M2, and M3, and $3 \cdot 10^6$ particles into M4, since it represents a larger portion of the pulmonary acinus. Particles were deposited on touch. particles are concentrated in very few message passing interface (MPI) partitions, resulting in a very poor load balance, which will be analyzed in the present work. The partitioning performed in the preprocessing step of the simulation aims to balance the workflow for solving the Navier-Stokes equations using the

	M1				M2			
	10 μm	15 μm	20 μm	25 μm	10 μm	15 μm	20 μm	25 μm
# of CPUs	96	96	96	96	96	96	96	96
Elapsed time (min)	371.4	389.1	362.1	386.9	100.6	154.9	102.1	113.61
	M3				M4			
	10 μm	15 μm	20 μm	25 μm	10 μm	15 μm	20 μm	25 μm
# of CPUs	96	96	96	96	288	288	288	288
Elapsed time (min)	91.5	92.3	91.0	91.8	375.1	371.8	369.9	371.4

Table 2
Computational resources and timing details for the different meshes on MN5, detailing the number of CPUs used and elapsed time for simulations with the different mesh deformations.



(a) Flowrate.

(b) Pressure drop.

Figure 2: Features of M4 for the different mesh deformations.

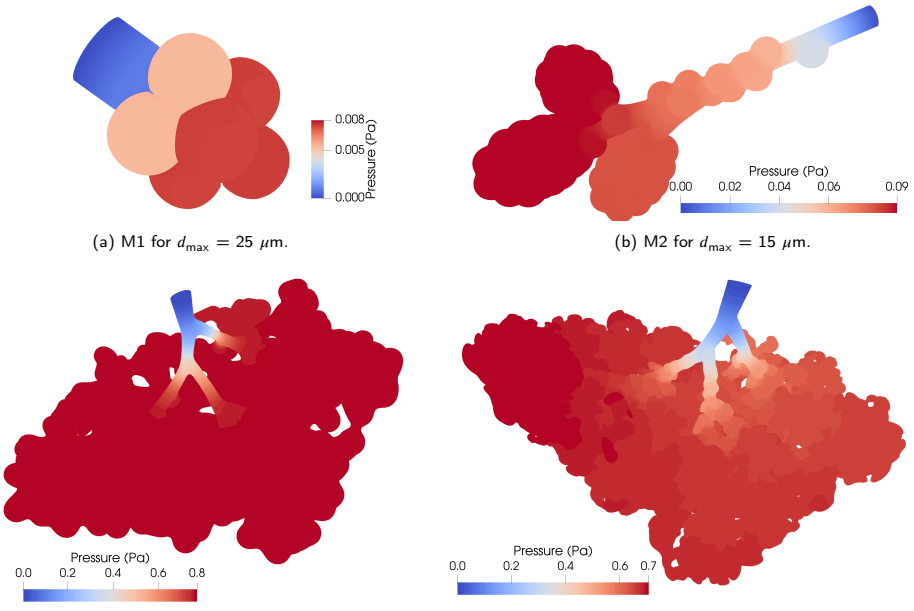
FEM method. A primary challenge arises when balancing particle transport, as particles are typically injected into a small part of the entire domain. Consequently, a load imbalance occurs when different processors or nodes in a parallel computing system handle unequal workloads. This disparity results in some processors completing their tasks earlier, or having no tasks at all, leading to idle time while waiting to be synchronized with others. Such inefficiencies increase the overall computation time, as faster processors must wait for the slower ones. When performing CFPD simulations on supercomputers efficient resource management becomes crucial. To attack these problems, we propose a dynamic solution, which is applied at runtime, the Dynamic Load Balancing Library (DLB) [18, 19]. DLB is applied at runtime meaning that we do not need to analyze specific inputs or modify the application code. The philosophy of the library is to exploit the computational resources, i.e., CPUs or cores, of the MPI processes blocked in an MPI blocking call by other processes running on the same node, by spawning more threads of the second level of parallelism, i.e., in our case, OpenMP.

3. Results

In the following, the simulation results are discussed with respect to the airflow features in Sec. 3.1 and the particle deposition in Sec. 3.2.

3.1. Airflow features

In this section we show the simulation results obtained using Alya, examining the airflow dynamics across different mesh deformations and delving into the analysis of particle transport and deposition. The study focuses on particles ranging from 1 nm to 10 μm, which are critical for understanding health impacts in the pulmonary acinus. Particles smaller than 10 μm penetrate deep into the lungs [17]. This



(a) M1 for $d_{max} = 25$ μm.

(b) M2 for $d_{max} = 15$ μm.

(c) M3 for $d_{max} = 15$ μm.

(d) M4 for $d_{max} = 20$ μm.

Figure 3: Pressure drop map for the four meshes during the peak exhalation, where pressure is maximum (t = 3 s).

size range encompasses various aerosols such as viruses, smoke, and dust and can be used for inhalation therapies.

Computational resources. With respect to the computational resources used and the elapsed time on MareNostrum 5 (MN5) supercomputer for the simulations, these details are presented in Tab. 2. All the timesteps were post-processed by Alya to later simulate particle transport.

Respiratory physiology. Regarding pressure dynamics, the observed pressure drop across the acinar walls aligns with respiratory mechanics. During inhalation, negative pressure relative to the atmosphere facilitates air entry into the respiratory zone, driven by thoracic expansion and diaphragm contraction. This expansion lowers intrathoracic pressure, increasing alveolar volume and drawing air in. Conversely, during exhalation, acinar pressure turns positive, aiding in air expulsion. This occurs as the diaphragm relaxes and the lung tissues' elasticity compresses the alveoli, pushing air outward.

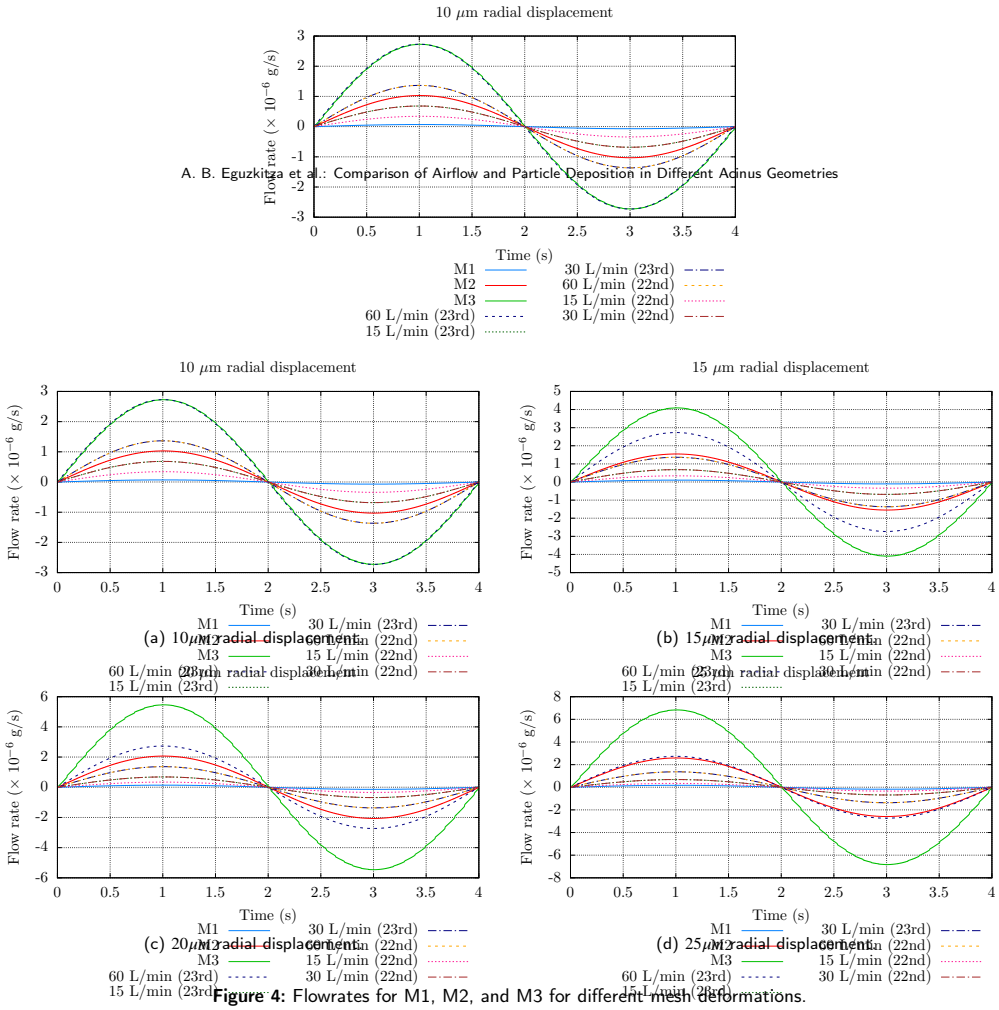
The pressure drop profile exhibits a sinusoidal pattern across all meshes. Figure 2b illustrates the pressure drop profiles for all deformations in M4. Figure 3 depicts the pressure variations within the acinar walls during peak exhalation, the phase of the breathing cycle where pressure reaches its maximum.

Literature offers limited information on acinar volume expansion, typically ranging between 20% and 35%. Our results, detailed in Tab. 3, correspond to the moment of maximum expansion t = 2 s. To align with physiological characteristics of the acinus, some experimental cases need to be discarded due to either excessive or insufficient expansion. Specifically, some cases with $d_{max} = 25$ μm show excessive expansion, while a few with $d_{max} = 10$ μm exhibit insufficient expansion.

Another critical aspect is comparing the measured flowrates with those expected if typical flowrates were to enter the respiratory system through the nose. Considering the lung's dichotomous branching pattern, the flowrate at a given generation g can be calculated by

$$\dot{M}_g = \frac{\dot{M}_0}{2^g}, \tag{15}$$

where $\dot{M}_0$ is the flowrate entering the respiratory system. Typical values for $\dot{M}_0$ range from 15, 30 to 60 L/min, varying according to different respiratory conditions. However, the assumptions in Eq. (15) become less accurate beyond the 16th generation, where the lung's branching pattern



(a) 10 μm radial displacement. (b) 15 μm radial displacement. (c) 20 μm radial displacement. (d) 25 μm radial displacement.

Figure 4: Flowrates for M1, M2, and M3 for different mesh deformations.

Volume Expansion (%)				
Mesh	10 μm	15 μm	20 μm	25 μm
M1	23.2	35.7	49.1	63.8
M2	8.7	13.1	17.8	22.5
M3	15.7	23.9	32.1	40.7
M4	13.8	20.8	27.9	35.2

Table 3
Volume expansion for each mesh for the different deformations for t = 2 s.

does not strictly follow a simple dichotomous model and changes more frequently. Additionally, the equation does not consider losses from friction or bends in the airways. These limitations mean that the formula, while providing a useful initial estimate, likely underestimates true flowrates in the deeper parts of the lung. This highlights the need for more sophisticated models that can more accurately account for these variations, as minor differences can lead to significant errors.

Figure 4 and Fig. 2b show the flowrates measured at the inlets for M1, M2, M3, and for M4, alongside theoretical values calculated using the dichotomous branching model.

For mesh M1, the flowrate entering the mesh is so minimal that it does not align well with predictions from the branching model, suggesting that this synthetic model may not accurately represent alveolar sacs. Conversely, certain configurations of the other meshes exhibit a better fit. Additionally, the results for M4 align more closely with theoretical expectations. However, these alignments are sensitive; even minor changes in size can significantly affect the flowrate entering the mesh. The consistency of the measured flowrates with theoretical predictions highlights the accuracy of our model, confirming its ability to effectively mirror physiological conditions. Nevertheless, it is important to note that the theoretical model has several limitations.

3.2. Particle deposition

Simulations of particle transport and deposition were conducted over three respiratory cycles, assuming periodic

	M1				M2			
	10 μm	15 μm	20 μm	25 μm	10 μm	15 μm	20 μm	25 μm
# of CPUs	62	62	62	62	36	36	36	42
Elapsed time (min)	165.0	169.2	167.1	208.9	114.9	108.0	103.7	96.3
	M3				M4			
	10 μm	15 μm	20 μm	25 μm	10 μm	15 μm	20 μm	25 μm
# of CPUs	36	36	36	36	224	224	224	224
Elapsed time (min)	100.1	85.3	44.3	88.5	166.2	213.9	152.5	198.5

Table 4
Computational resources used in MN5 to solve the particle transport for the different meshes and the elapsed time in minutes obtained for the different mesh deformations.

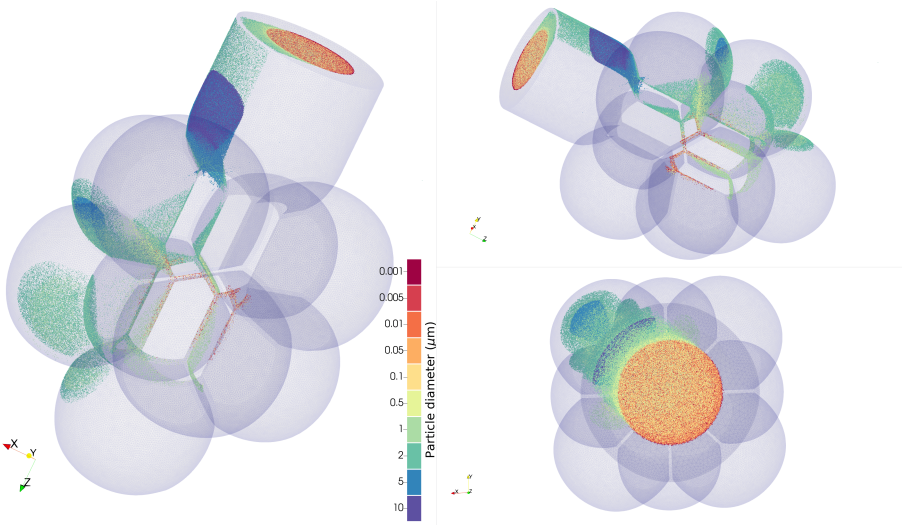
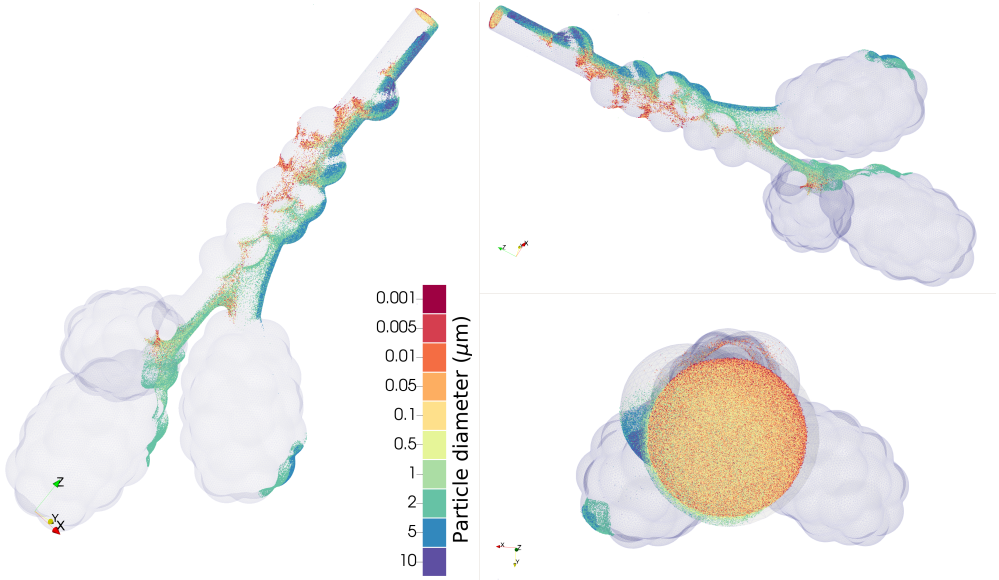


Figure 5: Particle deposition across M1 for $d_{max} = 15$ μm from different views, showing the spatial distribution of particles within the mesh.

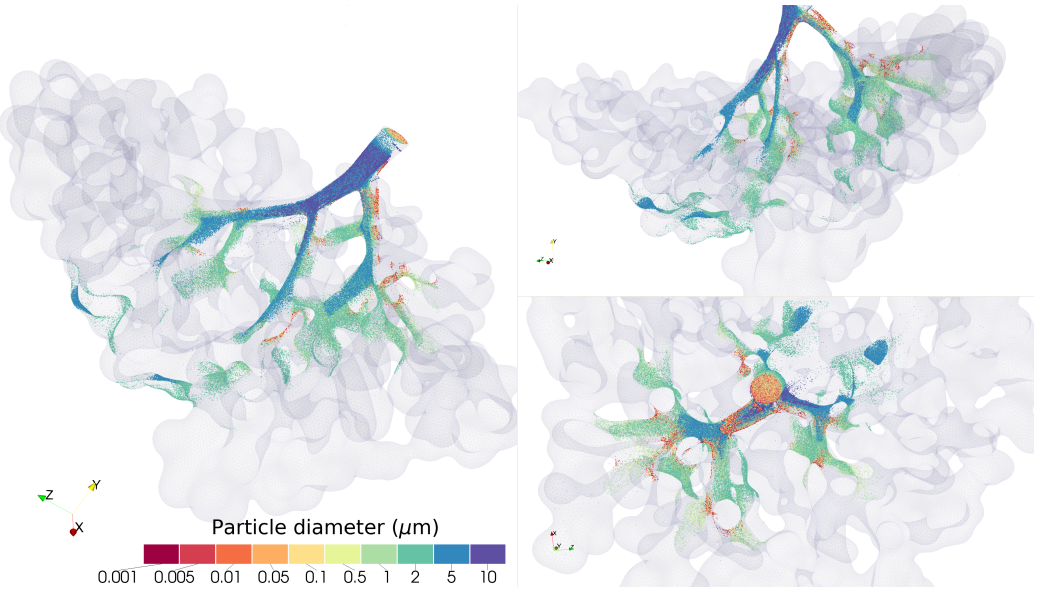
airflow solutions with the same time step as the initial airflow simulations.

Computational Resources. Solving particle transport is less resource-intensive than simulating incompressible fluid dynamics in deformable meshes. To efficiently explore different configurations in particle transport, involving various particle types and drag models, we stored the airflow solutions beforehand. This method significantly reduces computational demands. Consequently, fewer CPUs were utilized for particle simulations compared to fluid dynamics simulations. Detailed information on CPU usage and elapsed times for various mesh configurations is provided in Table 4. The results of the simulations were post-processed each 10 time steps by Alya.

Particle transport and deposition. In our study, particle transport varied by size and mesh complexity. Larger particles (5–10 μm) primarily followed gravity, with M4 showing extended fluid-driven movement. Particles of 2 μm traveled further due to fluid velocity, while submicron particles (< 1 μm) were dominated by convection and diffusion, traveling the greatest distances. During exhalation, nanometric particles were expelled, whereas 0.5–2 μm particles often



(a) Particle deposition across M2 for $d_{\text{max}} = 25$ μm.



(b) Particle deposition across M3 for $d_{\text{max}} = 10$ μm.

Figure 6: Particle deposition from different views, showing the spatial distribution of particles within the mesh.

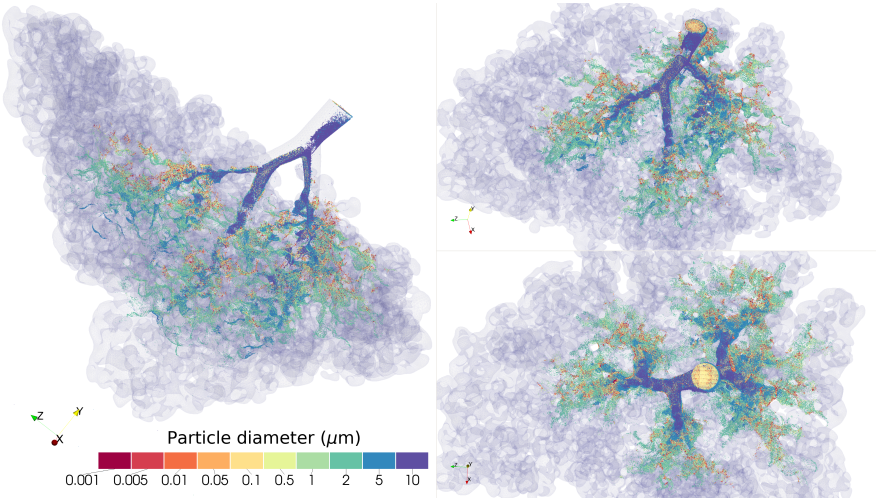


Figure 7: Particle Deposition in Mesh M4 for $d_{\text{max}} = 20$ μm from different views, illustrating the distribution and concentration of particles within the pulmonary acinus region.

lingered in distal regions. Micron-sized particles approached the inlet more easily in M1 and M2, but were trapped in alveoli in M3 and M4, reflecting their complex geometries.

Figure 5, Fig. 6a, Fig. 6b, and Fig. 7 illustrate how the deposited particles are spatially distributed across the meshes. In Figure 8a, Fig. 8b, Fig. 9a, and Fig. 9b we present the deposition efficiencies (DE) for different deformations across all meshes, alongside the percentage of particles exhaled through the inlet and those that remain suspended. The deposition efficiency for each particle type is calculated by

$$\text{DE}_i = \frac{\#\text{ of deposited particles of type } i}{\#\text{ of injected particles of type } i}. \tag{16}$$

This formula allows us to quantitatively assess the effectiveness of each mesh in trapping specific particle types under the given conditions.

As d_{max} increases, particle deposition slightly rises, particularly for larger particles (5–10 μm). Mesh geometry and fluid flow profile play a more significant role in deposition than volume changes.

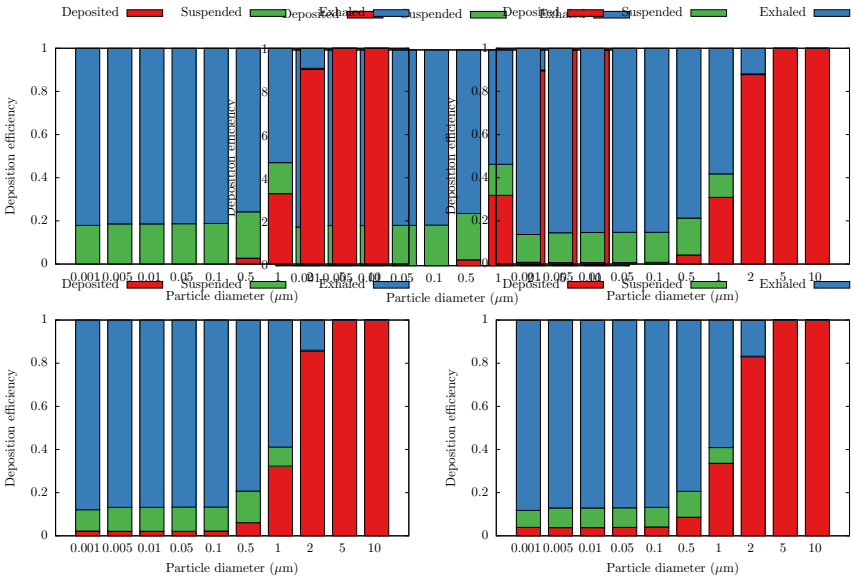
Meshes M1 and M2 show similar deposition efficiencies, with M1 retaining more suspended submicron particles, while M3 and M4 favor submicron particle deposition. Deposition of 1 μm particles is consistent across meshes, but 2 μm particles deposit less in M3 and M4.

For 5–10 μm particles, deposition occurs primarily in ducts, influenced by gravity, with M3 and M4 showing more even distribution due to their longer, non-alveolated ducts. Smaller particles (0.5–2 μm) travel further, depositing throughout alveoli in M1, alveolated ducts in M2, and terminal alveoli in M3 and M4. Submicron particles (<0.5 μm) are airflow-driven, largely exhaled, but deposit along alveolar edges, especially in M3 and M4 due to their complex geometries.

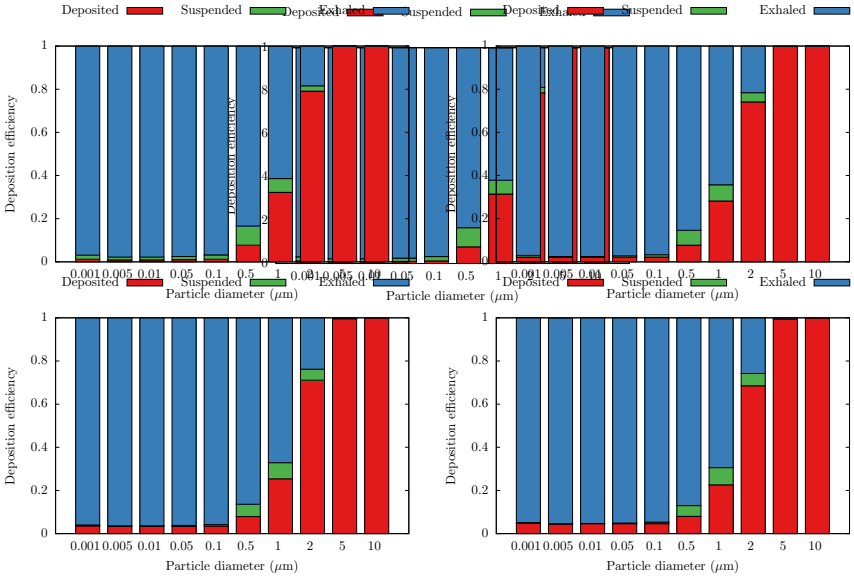
4. Conclusions

We will present the particles transported by fluid flow in the lung's respiratory zone. Due to the limitations of resolution clinical images, this part is not solved in general in the literature of the field. In this multidisciplinary work we have address this challenge offering a multiple version of such complex geometries with the solutions of mesh movement leading a fluid flow and transported particles.

This study marks a significant advancement in the field of personalized medicine. By offering a comprehensive understanding of airflow and particle dynamics, particularly in the distal respiratory zone, it provides critical insights into several areas. These include assessing the damage caused by pollutants to lung epithelial tissue, preventing the progression of respiratory infections caused by viruses and bacteria, and improving the efficacy of inhaled treatments. Ultimately, this work lays the groundwork for the development of personalized medicine strategies, specifically through the creation of digital twin models of the lungs.

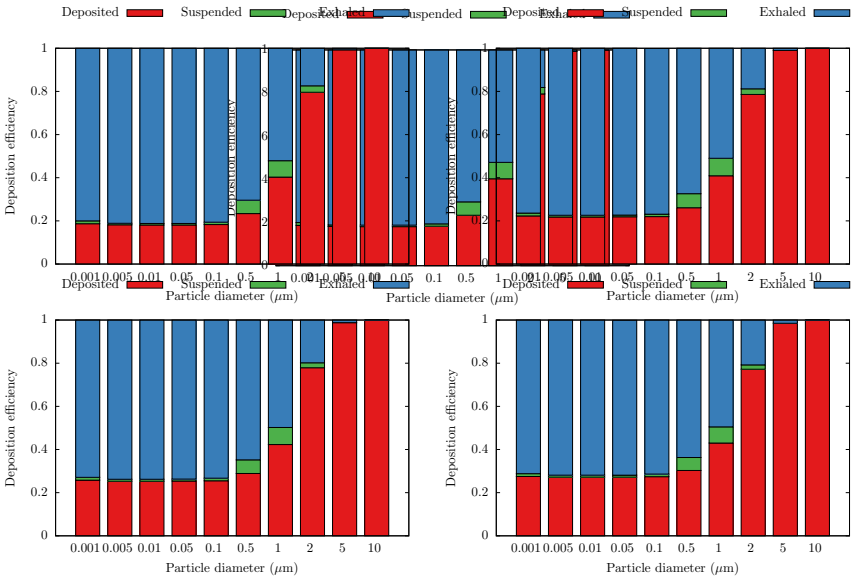


(a) Deposition efficiency of M1.

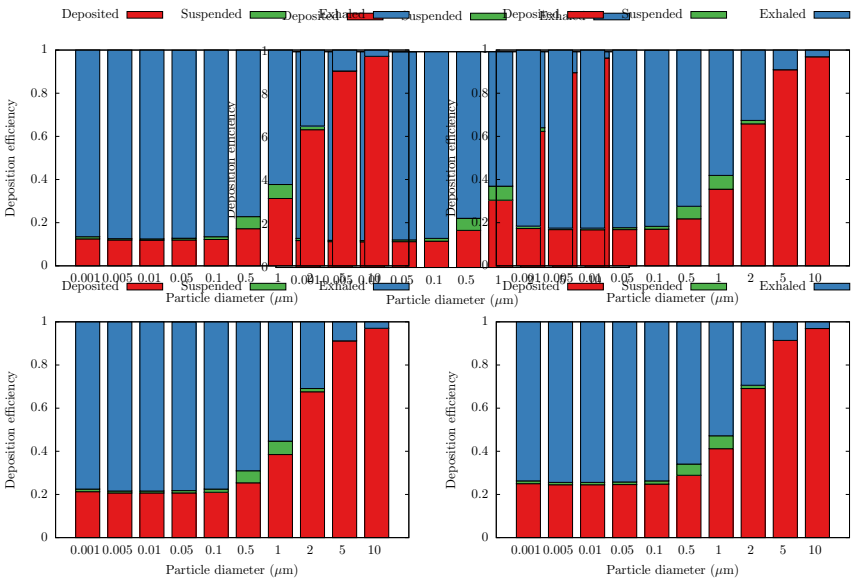


(b) Deposition efficiency of M2.

Figure 8: Deposition efficiency of M1 and M2 for the different mesh deformations of $10\mu m$ (upper left), $15\mu m$ (upper right), $20\mu m$ (lower left), and $25\mu m$ (lower right) radial displacement. The exhaled and suspended particles are also indicated.



(a) Deposition efficiency of M3.



(b) Deposition efficiency of M4.

Figure 9: Deposition efficiency of M3 and M4 for the different mesh deformations of $10\mu m$ (upper left), $15\mu m$ (upper right), $20\mu m$ (lower left), and $25\mu m$ (lower right) radial displacement. The exhaled and suspended particles are also indicated.

References

[1] J. Tu, K. Inthavong, G. Ahmadi, Computational Fluid and Particle Dynamics in the Human Respiratory System, Biological and Medical Physics, Biomedical Engineering, Springer Netherlands, Dordrecht, 2013. doi:10.1007/978-94-007-4488-2.

[2] J. Heyder, Deposition of Inhaled Particles in the Human Respiratory Tract and Consequences for Regional Targeting in Respiratory Drug Delivery, Proceedings of the American Thoracic Society 1 (4) (2004) 315–320. doi:10.1513/pats.200409-046TA.

[3] J. Dong, Y. Yang, Y. Zhu, Recent advances in the understanding of alveolar flow, Biomicrofluidics 16 (2) (2022). doi:10.1063/5.0084415.

[4] J. Dong, Y. Yang, Y. Zhu, New insight into air flow distribution in alveoli based on air- and saline-filled lungs, Microfluidics and Nanofluidics 24 (9) (2020) 71. doi:10.1007/s10404-020-02377-9.

[5] D. A. Field, Laplacian smoothing and Delaunay triangulations, Communications in Applied Numerical Methods 4 (6) (1988) 709–712. doi:10.1002/cnm.1630040603.

[6] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, M. Valero, Alya: Multiphysics engineering simulation toward exascale, Journal of Computational Science 14 (2016) 15–27. doi:10.1016/j.jocs.2015.12.007.

[7] J. C. Cajas, G. Houzeaux, M. Vázquez, M. García, E. Casoni, H. Calmet, A. Artigues, R. Borrell, O. Lehmkuhl, D. Pastrana, D. J. Yáñez, R. Pons, J. Martorell, Fluid-Structure Interaction Based on HPC Multicode Coupling, SIAM Journal on Scientific Computing 40 (6) (2018) C677–C703. doi:10.1137/17M1138868.

[8] J. Sznitman, T. Heimsch, J. H. Wildhaber, A. Tsuda, T. Rösgen, Respiratory Flow Phenomena and Gravitational Deposition in a Three-Dimensional Space-Filling Model of the Pulmonary Acinar Tree, Journal of Biomechanical Engineering 131 (3) (2009). doi:10.1115/1.3049481.

[9] G. Houzeaux, J. Principe, A variational subgrid scale model for transient incompressible flows, International Journal of Computational Fluid Dynamics 22 (3) (2008) 135–152. doi:10.1080/10618560701816387.

[10] R. Codina, G. Houzeaux, H. Coppola-Owen, J. Baiges, The fixed-mesh ALE approach for the numerical approximation of flows in moving domains, Journal of Computational Physics 228 (5) (2009) 1591–1611. doi:10.1016/j.jcp.2008.11.004.

[11] G. Houzeaux, R. Aubry, M. Vázquez, Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement, Computers & Fluids 44 (1) (2011) 297–313. doi:10.1016/j.compfluid.2011.01.017.

[12] R. Löhner, F. Mut, J. R. Cebral, R. Aubry, G. Houzeaux, Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: Extensions and improvements, International Journal for Numerical Methods in Engineering 87 (1-5) (2011) 2–14. doi:10.1002/nme.2932.

[13] A. Li, G. Ahmadi, Dispersion and Deposition of Spherical Particles from Point Sources in a Turbulent Channel Flow, Aerosol Science and Technology 16 (4) (1992) 209–226. doi:10.1080/02786829208959550.

[14] N.-S. Cheng, Comparison of formulas for drag coefficient and settling velocity of spherical particles, Powder Technology 189 (3) (2009) 395–398. doi:10.1016/j.powtec.2008.07.006.

[15] S. Haber, D. Yitzhak, A. Tsuda, Gravitational deposition in a rhythmically expanding and contracting alveolus, Journal of Applied Physiology 95 (2) (2003) 657–671. doi:10.1152/japplphysiol.00770.2002.

[16] E. R. McFadden, D. M. Denison, J. F. Waller, B. Assoufi, A. Peacock, T. Sopwith, Direct recordings of the temperatures in the tracheobronchial tree in normal man., Journal of Clinical Investigation 69 (3) (1982) 700–705. doi:10.1172/JCI110498.

[17] W. C. Hinds, Y. Zhu, Aerosol Technology: Properties, Behavior, and Measurement of Airborne Particles, 3rd Edition, John Wiley & Sons, Inc., 2022.

[18] M. Garcia, J. Corbalan, J. Labarta, LeWI: A Runtime Balancing Algorithm for Nested Parallelism, in: 2009 International Conference on Parallel Processing, IEEE, 2009, pp. 526–533. doi:10.1109/ICPP.2009.56.

[19] M. Garcia-Gasulla, M. Josep-Fabrego, B. Eguzkitza, F. Mantovani, Computational Fluid and Particle Dynamics Simulations for Respiratory System: Runtime Optimization on an Arm Cluster, in: Proceedings of the 47th International Conference on Parallel Processing Companion, ACM, New York, NY, USA, 2018, pp. 1–8. doi:10.1145/3229710.3229736.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Lung Digital Twin COVID-19 Infection: A Multiphysics - Multiscale HPC-Modeling Based on CFPD and Agent-Based Model Coupled Simulations

Alice Novell[a], Fernando Muñoz[b], Thaleia Ntiniakou[a], Arnau Montagud[a], Guillaume Houzeaux[a] and Ane Beatriz Eguzkitza[a,*]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain
[b]Universitat Politècnica de Catalunya, 1-3 Carrer de Jordi Girona, Les Corts, 08034 Barcelona, Spain

ARTICLE INFO†

Keywords:
Computational Fluid and Particle Simulations;
SARS-CoV-2 Transport and Infection;
Agent-Based Models;
Coupled Multi-Physics Problems;
Personalized Medicine;
Multiscale Lung Digital Twin

ABSTRACT

The present work is one of the three pieces (upper airways, lower conductive airways and respiratory zone) of a digital twin lung model developed by the *Physical and Numerical Modelling* research group from the CASE department in Barcelona Supercomputing Center (BSC). In particular, the study presents the solution of fluid flow and SARS-COV-2 particle transport in the lower conductive zone of the lungs, using a geometry based on patient specific images. The specific context of the current work is framed within the European Project: *CREXDATA: Critical Action Planning over Extreme-Scale Data*. Its general vision is to develop a generic platform for real-time critical situation management including flexible action planning and agile decision making over streaming data of extreme scale and complexity. One of the use cases of the project is the COVID-19 pandemic crisis, studying viral evolution in patients. To that end, the first step is to develop a mechanistic multiscale model to build a toolbox aimed at having a digital twin for the treatment of patients.

1. Introduction

In response to recent health crisis like the COVID-19 pandemic, researchers have utilized modeling to address complex challenges in crisis management. However, existing models often lack deep insights into viral evolution for novel therapeutic strategies [1, 2]. Our aim is to obtain a multiscale, multicellular, spatiotemporal model for simulating lung tissue infected by SARS-CoV-2, spanning from organ to cell level. This model aims to facilitate the discovery of patient-specific therapeutic targets and enable full-sized lung organ simulations.

Our approach integrates Alya [3] and PhysiBoSS [4] simulators for optimized, patient-specific interventions. Alya, a HPC multiphysics tool, simulates airflow in the lung airways and viral particle transport, while PhysiBoSS, an agent-based tool, assesses alveolar states and cellular impacts.

This workflow can be summarized in:

1. The transport of viral particles in a mesh geometry of the lower conducting airways, considering up to 17th generation.
2. Coupling with the lung respiratory zone, to assess the viral deposition in the alveolar tissue.
3. Simulation of cell-level infection evolution in the epithelium.

In the current work, our focus lies on the first step of the proposed workflow: simulating airflow and viral aerosol transport in the lower conducting airways to evaluate the fraction of inhaled particles that reach the respiratory zone.

2. Methodology

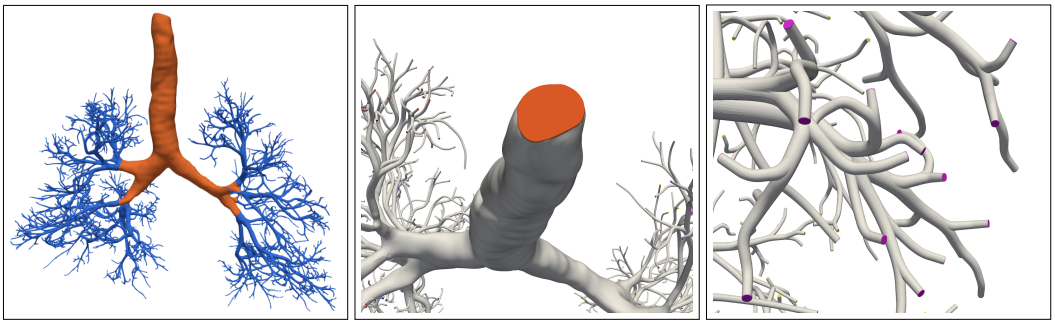
This section describes the geometry and the computational mesh in Sec. 2.1, how the airflow simulations are conducted in Sec. 2.2, and the method for the particle transport in Sec. 2.3.

2.1. Geometry and mesh description

The geometry is patient-specific from generation 0 to 3 and synthetically generated from generation 3 to ∼ 17. The patient-specific part is reconstructed from a clinically

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02477 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
alice.novell@bsc.es (A. Novell);
fernando.munoz@estudiantat.upc.edu (F. Muñoz);
thaleia.ntiniakou@bsc.es (T. Ntiniakou); arnau.montagud@bsc.es (A. Montagud); guillaume.houzeaux@bsc.es (G. Houzeaux);
beatriz.eguzkitza@bsc.es (A.B. Eguzkitza)
ORCID(s): 0009-0008-0201-6328 (A. Novell); 0009-0003-6182-1347 (F. Muñoz); 0000-0001-6470-9885 (T. Ntiniakou); 0000-0002-7696-1241 (A. Montagud); 0000-0002-2592-1426 (G. Houzeaux); 0000-0002-3302-6667 (A.B. Eguzkitza)



(a) (b) (c)

Figure 1: (a) Three dimensional geometry of airways, with patient-specific part shown in orange and synthetic part shown blue. (b) Trachea inlet. (c) Terminal outlets.

	Very shallow	Shallow	Normal	Deep	Very deep
$Q_{\max}$ (L/min)	7.5	15	22.5	30	60
$V_{inhaled}$ (L)	0.159	0.318	0.477	0.637	1.274

Table 1
Characteristics of 5 breathing patterns in a sinusoidal model.

acquired computed tomography (CT) scan, and each terminal is connected to the corresponding synthetically generated extension to obtain the conducting zone of the airways (Fig. 1). The synthetic airways are computed following [5]. An initial study to asses the fidelity of the resulting geometry with anatomical experimental data found in literature is performed.

An unstructured mesh is employed, due to the complex shape of the geometry. The mesh has more than 45M elements and is hybrid, made of tetrahedrons with prism layers at the wall to resolve the wall boundary layer profile.

2.2. Airflow simulations

In order to obtain the airflow simulations, we solve the Navier-Stokes equations. The numerical model to solve these equations is based on a stabilized finite element method. A description of this numerical method can be found in [6].

Regarding boundary conditions, an initial approach involves imposing flow rate at the trachea inlet and setting zero pressure at all outlets, aligning with existing literature for result validation [7, 8]. However, further exploration of alternative boundary conditions is undertaken to obtain a more accurate solution that better captures the underlying physical phenomena.

We simulate both stationary flows and sinusoidal breathing patterns. The parameters are shown in Tab. 1. In the case of stationary flow, a constant flow rate of $Q_{\max}$ is imposed at the trachea, while for dynamic breathing a sinusoidal function is imposed:

$$Q_{\text{trachea}}(t) = \frac{d\Delta V}{dT} = \left(\frac{V_{\text{inh}}\,\pi}{T}\right)\sin\left(\frac{2\pi}{T}t\right) = Q_{\max}\sin\left(\frac{2\pi}{T}t\right), \quad (1)$$

where the breathing cycle period T is 4 seconds in all cases.

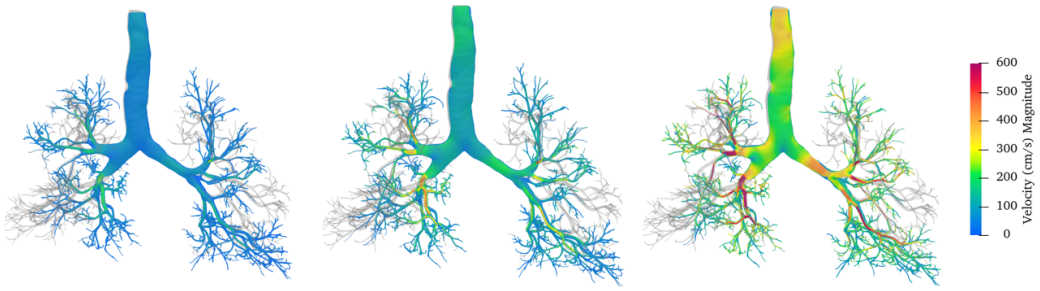
2.3. Particle transport

The transport of particles is simulated in a Lagrangian frame of reference, following each particle individually. From the numerical point of view, the main assumptions to develop the model are: particles are assumed sufficiently small to neglect their effect on the air, therefore, a one way coupling is considered; particles do not interact with each other; and particle rotation is neglected.

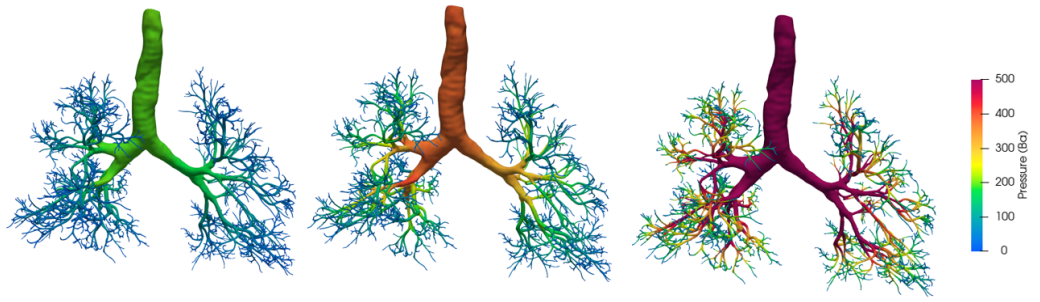
Particles are injected at the trachea inlet following a uniform distribution. For the steady simulations, all particles are released at the initial time step. For the unsteady simulations, a number of particles proportional to the flow rate are injected at each timestep during inspiration. Different particle sizes in a range around 120 nm are considered, representing viral particles [9].

3. Results

This section presents the results of the study. In more detail, Sec. 3.1 presents and analysis of the conductive zone



(a) Velocity streamlines in the bifurcating model for different airflow rates: (left) 7.5 L/min, (middle) 15 L/min, and (right) 30 L/min.



(b) Pressure for three different airflow rates: (left) 7.5 L/min, (middle) 15 L/min, and (right) 30 L/min.

Figure 2: Visualization of the velocity streamlines and the pressure distribution in the lung model.

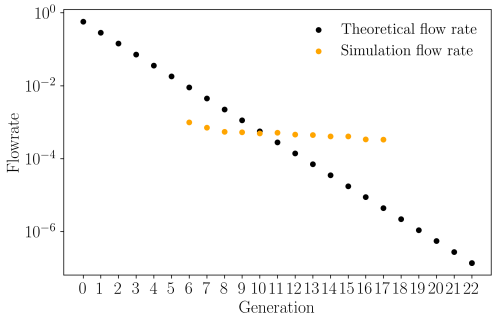


Figure 3: Flow rate per generation for 30 L/min.

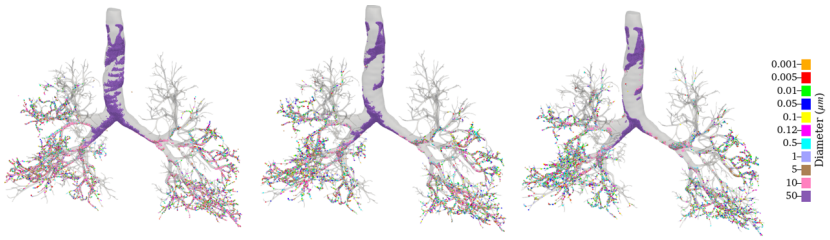
geometry, Sec. 3.2 investigates stationary simulations of the conductive zone, and Sec. 3.3 looks at transient simulations.

3.1. Analysis of the conductive zone geometry

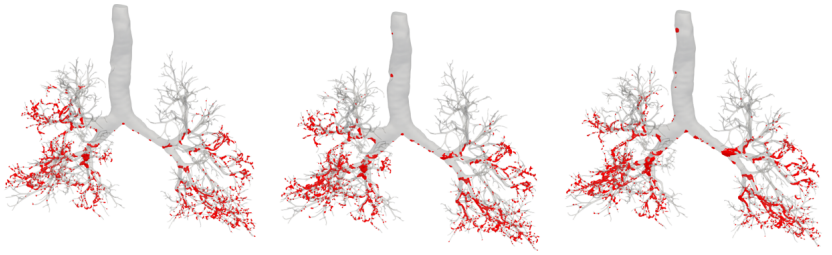
The validation of the conductive zone geometry confirms the model's adherence to anatomical structures, while highlighting some differences. The relationship between airway diameter and generation shows close agreement with experimental data in the patient-specific portion (generations 0–2), whereas the synthetic geometry exhibits a faster diameter reduction.

Characterization of branching indicates a deviation from ideal dichotomous branching, as not all branches divide beyond generation 8, resulting in a distribution of terminal branches centered around generation 11. Moreover, the terminal diameters remain fixed regardless of generation, leading to abrupt changes in diameter at higher generations.

The model's lobe-specific terminal distribution is anatomically realistic, with the right side containing more terminals and the right middle lobe having the fewest. However, the observations that not all branches reach the same generation, combined with inaccuracies in diameters, underscore the need for improving the geometry and setting boundary conditions that accurately capture the resulting pressure differences.



(a) Particle deposition for three different airflow rates: (left) 7.5 L/min, (middle) 15 L/min, and (right) 30 L/min.



(b) SARS-CoV-2 particle deposition for three different airflow rates: (left) 7.5 L/min, (middle) 15 L/min, and (right) 30 L/min.

Figure 4: Visualization of the particle deposition in the lung model.

3.2. Stationary simulations of the conductive zone

The stationary simulations of the conductive zone are validated to ensure solution accuracy and stability. Velocity profiles at various positions in the airways are validated against prior studies, revealing similar velocity trends. Reynolds number calculations confirm laminar or transitional flow, justifying the neglect of turbulence.

The air velocity streamlines shown in Fig. 2a reveal peak velocities after the initial bifurcation areas of all five main lobes, marking the transition from the patient-specific to the synthetic model. This peaks in velocity are likely due to the abrupt changes in diameter at these transition points.

Flow rate comparisons with theoretical values as depicted in Fig. 3 highlight discrepancies due to uniform terminal diameters and boundary conditions, which fail to account for realistic variability across generations.

The pressure distribution results, see Fig. 2b, show a steady decrease from the trachea to lower generations, with maximum pressure observed near the tracheal walls, as expected. Higher flow rates exhibit more significant pressure drops in lower airways.

Particle deposition patterns as found in Fig. 4a reveal that larger particles, e.g., 50 μm, predominantly deposit in the trachea, with deposition spreading into deeper airways as particle size decreases. The SARS-CoV-2 particle deposition, see Fig. 4b, shows higher concentrations in the bronchial regions for higher flow rates.

Flow and particle outflow analyses demonstrate a consistent pattern of higher outflow in the right lung, with the right middle lobe showing the lowest outflow. Particle outflow is inversely related to flow rate, with higher deposition observed at increased flow rates. Larger particles exhibit greater deposition than smaller particles, emphasizing size-dependent behavior in particle transport and deposition.

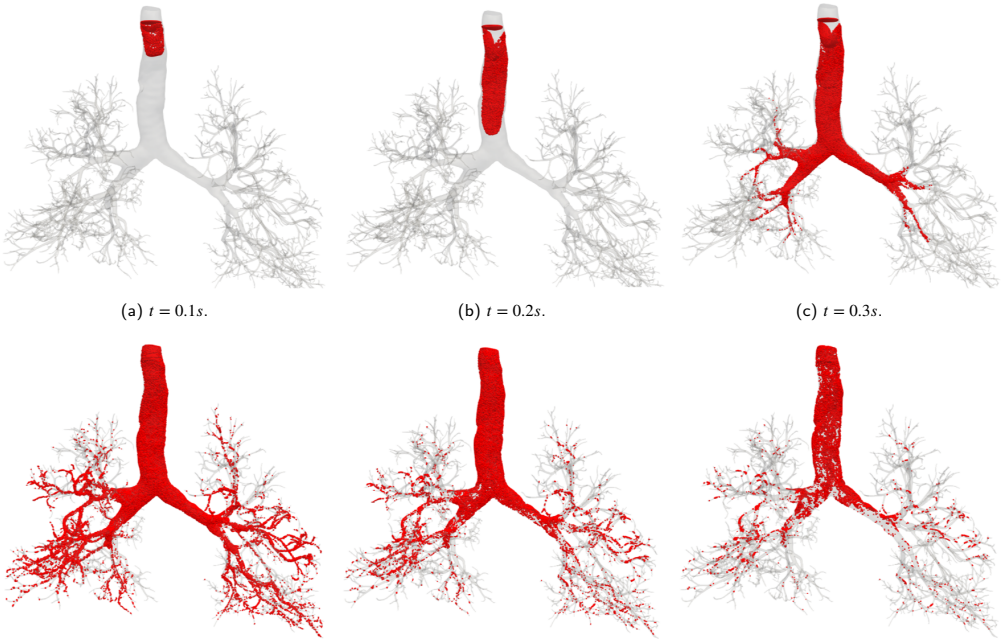
3.3. Transient Simulations: Breathing cycle

SARS-CoV-2 particle transport is visualized at the start of inhalation ($t = 0.1, 0.2, 0.3s$) and exhalation ($t = 2.1, 2.3, 2.5s$) in Fig. 5.

Deposition progresses as particles initially deposit in the deep airways during inhalation and then in the first generations during exhalation, see Fig. 6.

The percentage of particles exiting through the terminal airways in each lobe as depicted in Fig. 7a show that more particles exit through the right lung than the left, with the lowest outflow observed in the right middle lobe.

The percentage distribution of deposited, outgoing, and exhaled particles for each flow rate, see Fig. 7b, shows high deposition percentages, particularly for 60 L/min and 15 L/min. It is noted that deposition may be overestimated due



(a) $t = 0.1s$. (b) $t = 0.2s$. (c) $t = 0.3s$.

(d) $t = 2.1s$. (e) $t = 2.3s$. (f) $t = 2.5s$.

Figure 5: Particle transport at different time stances for 30L/min.

to the assumption that all particles touching the wall remain deposited, while in reality, not all particles stay attached.

4. Conclusions

This study examines SARS-CoV-2 particle behavior within the conductive zone of the respiratory system, providing key insights into particle deposition dynamics. The results demonstrate that airflow rates significantly affect deposition patterns, with elevated airflow (such as during physical exertion) increasing the likelihood of particles reaching deeper regions. This suggests a heightened risk of infection in the lower respiratory tract under certain conditions.

However, the current model has limitations. The synthetic geometry used diverges from anatomical accuracy, with inconsistencies in branch diameters and incomplete branching to terminal generations.

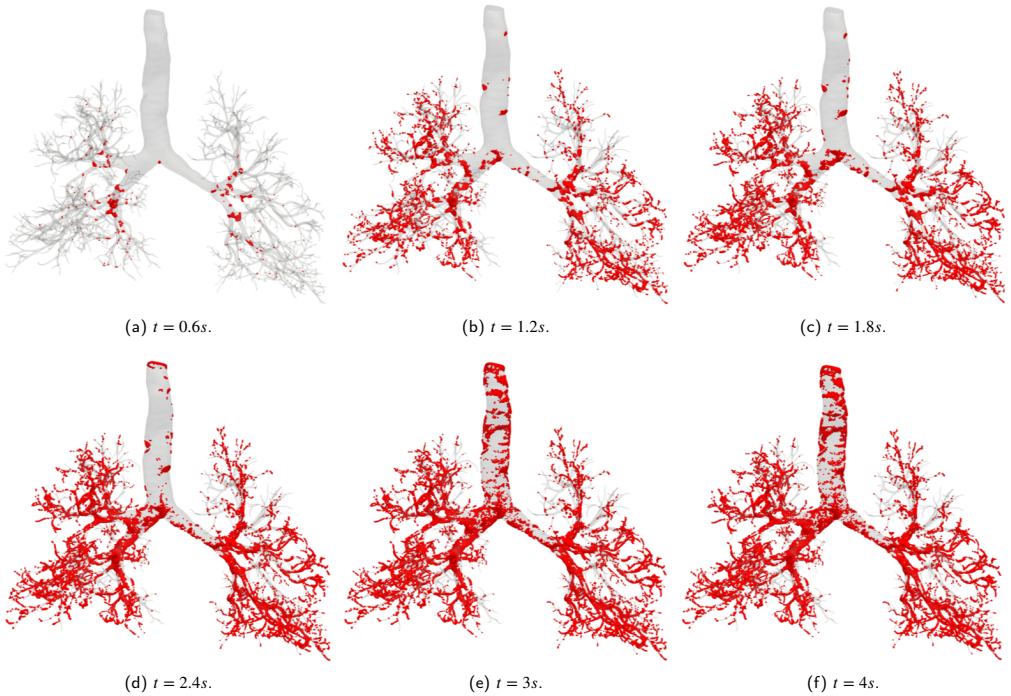
5. Future work

Future efforts will focus on addressing the geometric limitations of the conductive zone model by incorporating anatomically accurate branching structures and implementing more sophisticated boundary conditions, such as assigning unique conditions to individual outlets to better simulate physiological variations.

Moreover, expanding the model to include a comprehensive respiratory zone geometry, currently under development, will enable a more detailed analysis of particle behavior throughout the entire respiratory tract.

Additionally, improvements to the particle deposition model will account for particle bouncing upon wall contact, offering a more accurate representation of deposition dynamics. The model will also be adapted to study the impact of pathological conditions, such as asthma and COPD, by modifying airway geometries to reflect disease-specific changes.

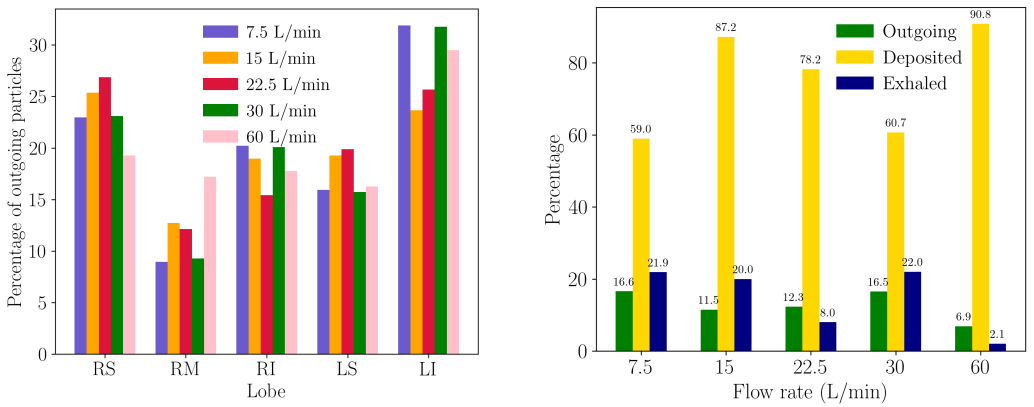
Coupling the model with PhysiBoSS will enable detailed simulations of virus-cell interactions, which will provide deeper insights into viral behavior and infection progression.



(a) $t = 0.6s$. (b) $t = 1.2s$. (c) $t = 1.8s$.

(d) $t = 2.4s$. (e) $t = 3s$. (f) $t = 4s$.

Figure 6: Deposition progression at different time instances for 30L/min.



(a) Percentage of outgoing particles through the terminals in each lobe for the five flow rates.

(b) Percentage distribution of deposited, outgoing, and exhaled particles for the five flow rates.

Figure 7: Distributions of particles.

References

[1] I. Cooper, A. Mondal, C. G. Antonopoulos, A SIR model assumption for the spread of COVID-19 in different communities, Chaos, Solitons & Fractals 139 (2020) 110057. doi:10.1016/j.chaos.2020.110057.

[2] J. Shang, G. Ye, K. Shi, Y. Wan, C. Luo, H. Aihara, Q. Geng, A. Auerbach, F. Li, Structural basis of receptor recognition by SARS-CoV-2, Nature 581 (2020) 221–224. doi:10.1038/s41586-020-2179-y.

[3] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, M. Valero, Alya: Multiphysics engineering simulation toward exascale, Journal of Computational Science 14 (2016) 15–27. doi:10.1016/j.jocs.2015.12.007.

[4] M. Ponce-de Leon, A. Montagud, V. Noël, A. Meert, G. Pradas, E. Barillot, L. Calzone, A. Valencia, PhysiBoSS 2.0: a sustainable integration of stochastic Boolean and agent-based modelling frameworks, npj Systems Biology and Applications 9 (1) (2023) 54. doi:10.1038/s41540-023-00314-4.

[5] M. H. Tawhai, P. Hunter, J. Tschirren, J. Reinhardt, G. McLennan, E. A. Hoffman, CT-based geometry analysis and finite element models of the human and ovine bronchial tree, Journal of Applied Physiology 97 (2004) 2310–2321. doi:10.1152/japplphysiol.00520.2004.

[6] G. Houzeaux, R. Aubry, M. Vázquez, Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement, Computers and Fluids 44 (2011) 297–313. doi:10.1016/j.compfluid.2011.01.017.

[7] M. S. Islam, P. Larpruenrudee, A. R. Paul, G. Paul, T. Gemci, Y. Gu, S. C. Saha, SARS CoV-2 aerosol: How far it can travel to the lower airways?, Physics of Fluids 33 (6 2021). doi:10.1063/5.0053351.

[8] T. Gemci, V. Ponyavin, Y. Chen, H. Chen, R. Collins, Computational model of airflow in upper 17 generations of human respiratory tract, Journal of Biomechanics 41 (2008) 2047–2054. doi:10.1016/j.jbiomech.2007.12.019.

[9] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao, W. Tan, A Novel Coronavirus from Patients with Pneumonia in China, 2019, New England Journal of Medicine 382 (8) (2020) 727–733. doi:10.1056/NEJMoa2001017.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Computational Modeling of Particles Fate in Nasal Drug Treatments

Silvia Ceccacci[a], Jose Angel Vicente Porres[b], Nerea Rio Grela[c], Hadrien Calmet[a], Abel Gargallo Peiro[a], Clement Rigaut[d], Benoit Haut[a], Guillaume Houzeaux[a] and Ane Beatriz Eguzkitza[a,*]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034 Barcelona, Spain
[b]Universitat Politecnica de Catalunya, Facultat d'Informatica de Barcelona, B6 Building Campus Nord, C/Jordi Girona Salgado, 1-3, 08034 Barcelona, Spain
[c]Universitat Autónoma de Barcelona, Department of Mathematics, Building C Science Faculty, 08193 Bellaterra, Barcelona, Spain
[d]Transfers, Interfaces and Processes Laboratory, Université Libre de Bruxelles, Avenue F.D. Roosevelt, 50, 1050 Bruxelles, Belgium

ARTICLE INFO†

Keywords:
Computational Fluid Dynamics;
Numerical Modeling;
Multi-Scale;
Multi-Physics

ABSTRACT

The present work is one of the three pieces (upper airways, lower conductive and respiratory zones) of a digital twin lung model developed by the *Physical and Numerical Modelling* research group of the CASE department at the Barcelona Supercomputing Center (BSC). In this study, we focus on the upper airways, and we present a computational model to analyze the fate of particles in nasal drug treatments. By integrating fluid dynamics with particle transport algorithms, the computational model, implemented in the in-house code Alya, aims at predicting the particles behavior, from their interactions with the nasal cavity walls considering mucus layer, to their deposition and ultimately their fate once deposited. This work is framed within the project *DREAMS: Particle Deposition Computational Model for ChildREn Airways with Mucus Surface.*

1. Introduction

Inflammatory upper airway diseases, encompassing a range of conditions, such as allergic rhinitis, rhinosinusitis and laryngitis, are estimated to affect around 20% of the global population. The symptoms involve obstruction of the upper airways, resulting in poor breathing capacity, thus inevitably causing poor life quality and high economical costs. Inhalation therapy, including nasal spray treatments, is an attractive approach to treat such respiratory diseases. To improve treatment efficacy, it is crucial to understand the movement of drug particles through the nasal cavity, their interactions with the nasal walls, and their eventual deposition in the Airway Surface Liquid (ASL) layer, which is made of two substrates: mucus layer (highly viscous), and periciliary layer (less viscous). Following deposition, it is important to assess the drug uptake from these particles.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02478 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
silvia.ceccacci@bsc.es (S. Ceccacci); jose.angel.vicente@estudiantat.upc.edu (J.A. Vicente Porres); Nerea.Rio@autonoma.cat (N. Rio Grela); hadrien.calmet@bsc.es (H. Calmet); abel.gargallo@bsc.es (A. Gargallo Peiro); clement.rigaut@ulb.be (C. Rigaut); benoit.haut@bsc.es (B. Haut); guillaume.houzeaux@bsc.es (G. Houzeaux); beatriz.eguzkitza@bsc.es (A.B. Eguzkitza)
ORCID(s): 0000-0002-8780-9739 (S. Ceccacci); 0009-0006-4681-4722 (J.A. Vicente Porres); 0009-0008-4329-1668 (N. Rio Grela); 0000-0001-5443-761X (H. Calmet); 0000-0003-3742-2197 (A. Gargallo Peiro); 0000-0003-4999-8601 (C. Rigaut); 0000-0002-3021-0207 (B. Haut); 0000-0002-2592-1426 (G. Houzeaux); 0000-0002-3302-6667 (A.B. Eguzkitza)

2. Particle dynamics modeling

Deposition and uptake processes occur at different time scales. While deposition involves the interaction of drug particles with the nasal surfaces over a short time frame (i.e., seconds), uptake encompasses the gradual absorption of the drug into the ASL, which occurs over a longer period of time (i.e., minutes). By decoupling these processes, we can simulate each phase and provide insights into the effectiveness of nasal drug delivery.

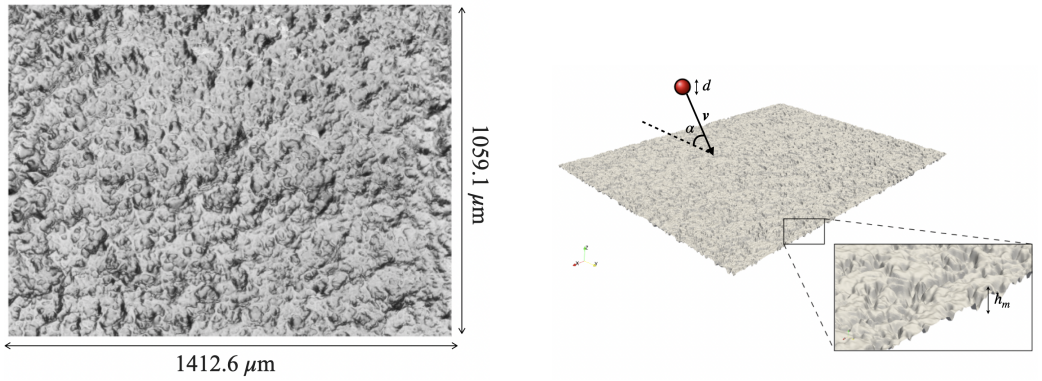
2.1. Particle-wall interaction and deposition model

We propose a computational model for the interaction between solid particles and nasal cavity walls. Unlike conventional approaches that assume a sticking or "deposit-on-touch" condition, our model considers the mucus layer coating the nasal cavity walls. In our approach, a critical collision velocity criterion, based on that proposed by Ohsaki et al. [1], determines whether the particle deposits or rebounds upon colliding with the wall. We define the critical collision velocity as

$$u_{\mathrm{cr}} = \frac{3\pi\mu r_p^2}{2m_p}\left(1+\frac{1}{e}\right)\ln\left(\frac{z}{z_a}\right), \tag{1}$$

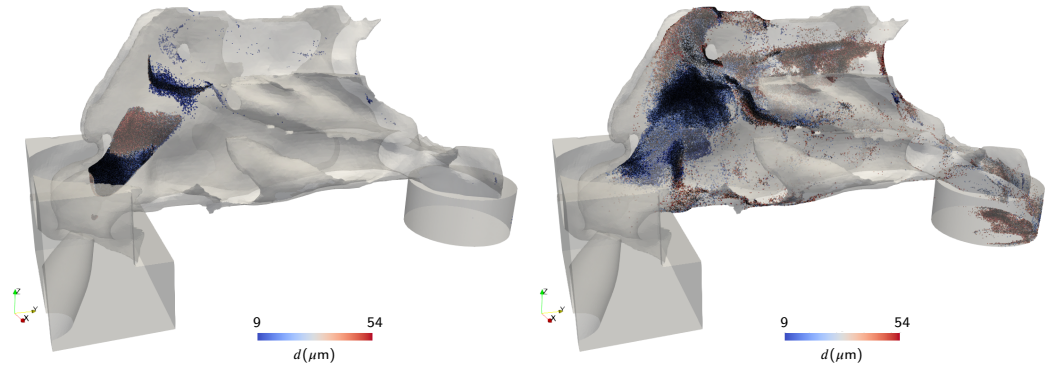
where μ is the mucus viscosity coefficient, r_p is the particle radius, m_p is the particle mass, where ρ_p is the particle density, e is the restitution coefficient, z is the mucus layer thickness, and z_a is the particle surface roughness.

In cases of rebound, the new particle direction is calculated with a statistical model based on the surface roughness



(a) (b)

Figure 1: (a) Microscopic image of the gel coating the 3D nasal cast, simulating mucus layer in the experiments and (b) geometry of the surface roughness obtained from (a), where h_m is the average surface roughness height.



(a) (b)

Figure 2: Deposition maps in a 9-year-old patient nasal geometry, obtained with (a) "deposit-on-touch" condition and (b) particle-wall interaction model, for the flow rate $15L/\text{min}$, where d is the particle diameter (μm).

of the mucus layer [2], illustrated in Fig. 1a. By geometrically characterizing the mucus surface, such model samples the angles of rebound θ and ϕ, in spherical polar coordinates, for a colliding particle of diameter d and angle of attack α (see Fig. 1b). The particle then continues its trajectory within the nasal cavity, gradually losing kinetic energy until it meets the deposition criterion (i.e., $\|\boldsymbol{u}_p\| \leq u_{cr}$, where $\boldsymbol{u}_p$ is the particle velocity), hence it deposits on the ASL layer.

Results. The efficacy of the particle-wall collision rebound model was assessed against experimental results using the same nasal cast geometry (from a 9-year-old patient), for the flow rates of $15L/\text{min}$ and $60L/\text{min}$. A parameter study has been carried out to identify the optimal values for the numerical deposition against the experimental one [3]. In Fig. 2, the deposition maps computed with"deposit-on-touch" and particle-wall interaction models, show a substantial difference in the particles distribution. In addition, the proposed model significantly enhances the accuracy of deposition maps, demonstrating a high level of consistency between the computational predictions and experimental observations.

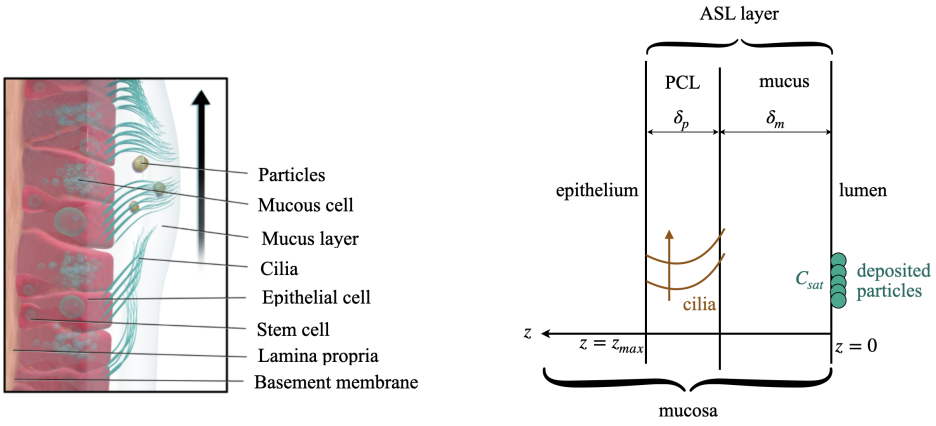


Figure 3: (a) Illustration of the mucosa in the respiratory tract and (b) schematic diagram for mathematical modeling.

2.2. Dissolution and diffusion models for deposited particles

Once the particles are deposited on the ASL layer, they are subject to dissolution, diffusion and advection processes [4]. The dissolution of drug particles in a liquid medium, through which they will progressively be reducing their mass and size, is described by the Noyes–Whitney equation

$$\frac{dm}{dt} = \frac{-AD_m(C_s - C_b)}{h}, \tag{2}$$

where m is the particle mass, A is the particle surface area, D_m is the diffusion coefficient, h is the mucus layer thickness, C_s is the drug solubility and C_b is the concentration of the dissolved drug in the bulk phase.

The dissolved particles diffuse across both layers of the ASL towards the epithelium, are advected towards the pharynx due to the mucociliary clearance, and are subject to enzymatic degradation. A diagram of the mucosa is illustrated in Fig. 3a (image adapted from Intersurgical[1]). The diffusion-advection-reaction equation is given as

$$\frac{\partial C}{\partial t} + \nabla \cdot (\boldsymbol{u}C) = \nabla \cdot (D_m \nabla C) + \rho(C), \tag{3}$$

where C is the drug concentration, $\boldsymbol{u}$ is velocity of the mucus and $\rho(C)$ is the reaction term, accounting for the enzymatic degradation. Eq. (3) is solved subject to boundary conditions, such that $C(z = 0) = C_{sat}$, where C_{sat} is the saturation concentration, assuming local equilibrium at the lumen-mucus interface, and $C(z = z_{max}) = kC$, where $z_{max} = \delta_m + \delta_p$, and k is the absorbance of the epithelium (see Fig. 3b).

[1] https://au.intersurgical.com/info/filtrationandhumidification

3. Conclusions

Once the particle-wall interaction and diffusion models are implemented separately, the outlook of this work is to couple them to achieve a more comprehensive simulation. In this integrated approach, the concentration of deposited particles will be evaluated at the nodes of the computational mesh at each time step. These particles will then be iteratively dissolved into the ASL layer and advected through the nasal cavity.

References

[1] S. Ohsaki, R. Mitani, S. Fujiwara, H. Nakamura, S. Watano, Effect of Particle–Wall Interaction and Particle Shape on Particle Deposition Behavior in Human Respiratory System, Chemical and Pharmaceutical Bulletin 67 (12) (2019) 1328–1336. doi:10.1248/cpb.c19-00693.

[2] R. Zhang, Y. Li, Y. Liu, Improvement and application of a two-dimensional fractal particle-wall collision model, Powder Technology 411 (2022) 117910. doi:10.1016/j.powtec.2022.117910.

[3] H. Calmet, D. Dosimont, D. Oks, G. Houzeaux, B. V. Almirall, K. Inthavong, Machine learning and sensitivity analysis for predicting nasal drug delivery for targeted deposition, International Journal of Pharmaceutics 642 (2023) 123098. doi:10.1016/j.ijpharm.2023.123098.

[4] S. Chari, K. Sridhar, R. Walenga, C. Kleinstreuer, Computational analysis of a 3D mucociliary clearance model predicting nasal drug uptake, Journal of Aerosol Science 155 (2021) 105757. doi:10.1016/j.jaerosci.2021.105757.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Wet-Surface Modeling in Lattice-Boltzmann Simulations for Evaluating Surgery Impacts on the Humidity Transfer in Nasal Flows

Shota Ito[a,*], Mario Rüttgers[b], Moritz Waldmann[c] and Andreas Lintermann[b]

[a]Karlsruhe Institute of Technology, Lattice Boltzmann Research Group, Institute for Mechanical Process Engineering and Mechanics, Englerstraße 2, 46131, Karlsruhe, Germany
[b]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, 52458, Jülich, Germany
[b]RWTH Aachen University, Chair of Fluid Mechanics and Institute of Aerodynamics, Aachen, Germany

ARTICLE INFO†

Keywords:
Lattice Boltzmann Method;
Nasal Cavity Flows;
Humidity Transport;
Turbinectomy

ABSTRACT

A numerical study using the lattice-Boltzmann method is conducted to investigate the conditioning ability of the human nose, where a boundary treatment is implemented to model the latent heat effect. The humidity exchange at the wet surface of the nasal mucosa influences the wall temperature, imitating the thermal inertial effects of the mucosa tissue. To capture the curvature of the cavity geometry, interpolated bounce-back schemes are used to set the wall temperature and water concentration computed by the boundary model. The impact of evaporation on the conditioning ability is investigated for pre- and post-surgery cavity geometries of a patient that was diagnosed with enlarged turbinates and underwent turbinectomy. The widening of the nasal passages in the course of the turbinectomy cause a reduced pressure loss between the inlets (nostrils) and the outlet (pharynx), but also dry air streaming towards the back part of the airway-throat interface. This coincides with the patient's perception, who reported less efforts for breathing in, but at the same time a dry and sometimes painful feeling at the back of the throat.

1. Introduction

Clinical investigations of the conditioning ability of the nose demonstrate a challenge due to the difficult access of the nasal cavity with measurement equipment. Computational fluid dynamics (CFD) enables in-depth studies of the nasal conditioning ability, allowing to obtain velocity, pressure, temperature, and humidity concentration distributions inside the cavity. The lattice Boltzmann method (LBM) has proven to be a popular approach to address fluid flow problems in the medical field [1, 2] due to the adaptability towards complex geometries and easy extension to solve transport phenomena.

This article presents numerical studies on the influence of the latent heat effect of evaporation on the inhalation process through the nose. The nasal cavity geometry is obtained via computer tomography scans converted into a surface model with the automated machine learning-based pipeline described in [3]. CFD simulations are conducted for the pre- and post-surgical anatomy of a patient that underwent turbinectomy, a surgical intervention to treat enlarged turbinates. Whereas previous attempts for estimating and planning surgical interventions with an LBM mainly focus on a surgery's influence on the pressure or temperature distributions [4, 5], the current study allows to analyze the main complication of a turbinectomy, i.e., a decreased capability to humidify incoming air. On the one hand, turbinectomy is known to reduce nasal resistance and ease breathing [6]. On the other hand, it alters nasal conditioning, since less heat exchange and humidification are possible, resulting in streams of significantly colder and dryer air in the nasopharynx [7].

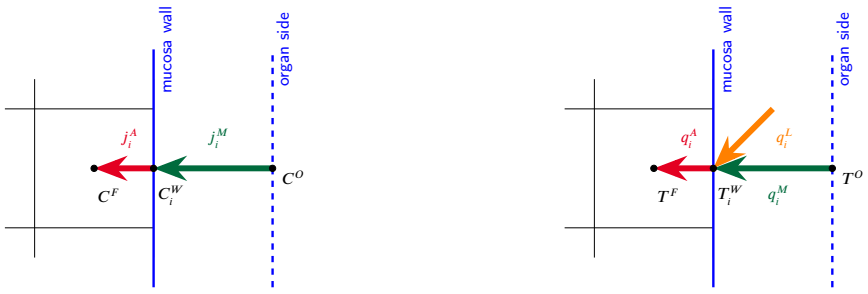
2. Numerical Method

For solving each macroscopic equation, i.e., Navier-Stokes equation (NSE) for the airflow and advection-diffusion equations (ADE) for the heat and humidity transfer, the LBM implemented in the open source multiphysics solver framework m-AIA [1] (multiphysics - Aerodynamisches Institut Aachen, formerly known as Zonal Flow Solver (ZFS) [8]) is employed where the temperature and water concentration distribution are computed via passive scalar transport. The velocity discretization model is the D3Q27 lattice and the second-order equilibrium distribution is used for the BGK-collision operator for all solved equations. The boundary treatment for the NSE system, in- and outflow for the ADE systems are chosen similarly as in [5]. The interpolated

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02479 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
shota.ito@kit.edu (S. Ito); (M. Rüttgers); (M. Waldmann); a.lintermann@fz-juelich.de (A. Lintermann)
ORCID(s): 0000-0002-6946-0203 (S. Ito); 0000-0003-3917-8407 (M. Rüttgers); 0000-0001-7895-761X (M. Waldmann); 0000-0003-3321-6599 (A. Lintermann)

[1] https://git.rwth-aachen.de/aia/MAIA/Solver



(a) Water flux at the mucosa wall

(b) Heat flux at the mucosa wall

Figure 1: The heat and water flux are evaluated at the mucosa wall to obtain the temperature and concentration at the wall. The computed latent heat flux depends on the passed water flux from the mucosa to the fluid.

bounce-back scheme for the ADE [9] is applied for the heat and humidity transport to compute the missing boundary populations to account for the boundary curvature of the nasal cavity wall, i.e.,

$$g_{\bar{i}}(\mathbf{x}, t+\Delta t) = \begin{cases} -2q\tilde{g}_i(\mathbf{x},t) + (2q-1)\tilde{g}_i(\mathbf{x}-\mathbf{c}_i\Delta t, t) + \frac{1}{4}\Phi_i^W & , \mathrm{q} < 0.5 \\ -\frac{1}{2q}\tilde{g}_i(\mathbf{x},t) + \frac{2q-1}{2q}\tilde{g}_{\bar{i}}(\mathbf{x},t) + \frac{1}{8q}\Phi_i^W & , 0.5 \le q \le 1 \end{cases}, \quad (1)$$

where q is the normalized distance between the boundary node and wall intersection point in the discrete velocity direction i, and Φ_i^W the desired value for the wall temperature T^W and water concentration C^W prescribed by the scheme. This scheme uses neighboring populations and the exact boundary distance in contrast to the simple bounce-back scheme which enables to prescribe Φ_i^W at the boundary wall.

In order to compute Φ_i^W, the wet surface model from [10] is used. The wet surface model assumes an additional outer layer with a uniform thickness in the normal direction to the geometry wall, modeling the nasal mucosa, which is referred to as the membrane layer. The temperature at the outside wall of the membrane layer is called the organ-side temperature T^O, which is assumed to be constant due to the continuous heat supply by the blood capillaries. For the same reason, the organ-side water concentration C^O is assumed to be fully saturated at the outer layer. To approximate C^O, the fully saturated water concentration of air as a function of the temperature T is computed via $C = w\rho T$, where ρ is the density of air and w is the water fraction (kg water vapor per kg dry air), given by the empirical function [10, 11]

$$w(T) = \frac{1}{1000} \cdot (2.027 + 0.0006312T^3 - 0.010972T^2 + 0.6036T). \quad (2)$$

To compute the missing wall temperature T^{W} and water concentration C^{W}, the energy and mass balance at the nasal wall are evaluated as depicted in Fig. 1. The heat flux q and water flux j are computed by using linear approximations for the gradients, i.e., the fluxes on the membrane side are

$$q_i^M = k^M \frac{T^O - T_i^W}{\delta_i^M} \quad \text{and} \quad (3)$$

$$j_i^M = D^M \frac{C^O - C_i^W}{\delta_i^M}, \quad (4)$$

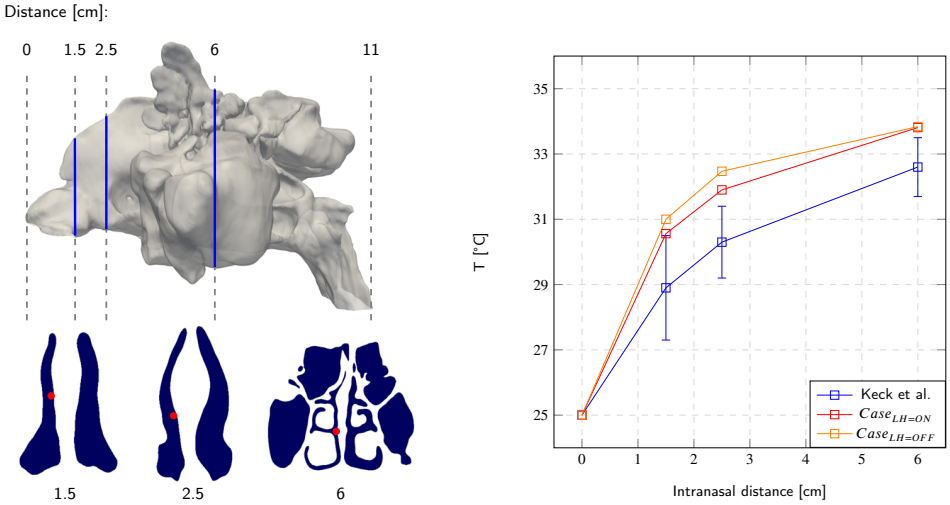
where k^M is the heat conductivity, D^M is the diffusivity constant of water, and δ_i^M is the distance in the membrane layer along the discrete velocity direction. At the opposite side of the nasal cavity wall, the airflow, heat, and concentration transport are simulated. The heat and water flux from a boundary lattice node with the fluid cell temperature T^F and water concentration C^F to the wall are approximated by

$$q_i^A = k^A \frac{T_i^W - T^F}{\delta_i^A} \quad \text{and} \quad (5)$$

$$j_i^A = D^A \frac{C_i^W - C^F}{\delta_i^A}, \quad (6)$$

where δ_i^A is the distance between the fluid cell center and the wall surface in the opposite discrete velocity direction of the missing population. Note, that in Eq. (6) the convective part is neglected due to the assumption of low flow velocities in the wall vicinity. An additional term in the energy balance is introduced, which imitates the heating and cooling of the mucous layer due to condensation and evaporation, respectively. The heat flux of the latent heat is computed by $q_i^L = -j_i^A h_i^L$, where h_i^L is the specific latent heat in kJ/kg, given as a function of the temperature [11]

$$h^L(T) = 2500.8 - 6.1434 \cdot 10^{-6} \cdot T^3$$



(a) Intranasal positions.

(b) Mucosa surface temperature.

Figure 2: Comparison of the intranasal mucosa temperature concentration at the positions 1.5 cm, 2.5 cm, and 6 cm between the simulation and measurements from [12]. The influence of the latent heat effect is shown.

$$+ 1.5893 \cdot 10^{-3} \cdot T^2 - 2.3641 \cdot T. \qquad (7)$$

Evaluating the energy and mass balances as $q_i^A = q_i^M + q_i^L$ and $j_i^A = j_i^M$ along the missing population direction yields the missing wall values Φ_i^W for Eq. (1).

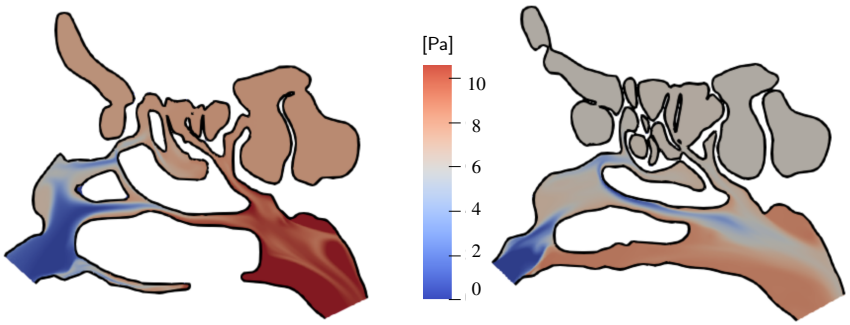
3. Results

The wet surface model was first sanity-checked in a wet pipe flow simulation in [13], similarly as in [10, 11]. Then, the impact of the latent heat on the mucosa surface is compared against the experimental data of [12]. That is, the nasal mucosa temperature is compared at three nasal locations shown in Fig. 2. Therein, it is shown that by considering the latent heat effect, the simulated nasal mucosa temperature approaches the trend of the experimental results. Note that a perfect match is not to be expected, since the nasal cavity model in the current study is different from the model in [12]. More details regarding the conducted tests of the wet surface model are given in the work of [13].

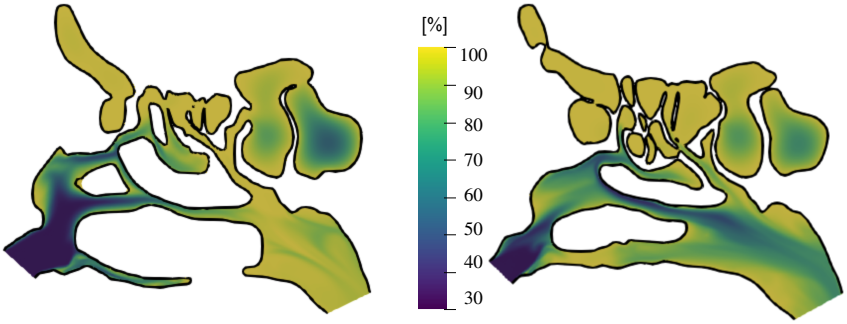
The flow simulations were conducted on 8 nodes of the graphics processing unit (GPU) partition of JURECA-DC [14], i.e., on a total number of 32 NVIDIA A100 GPUs. Traditionally, m-AIA was developed to run CFD simulations in parallel on central processing unit (CPU) partitions of high-performance computing (HPC) systems. However, recently GPU partitions on HPC systems are gaining popularity. To allow computations on both partitions, the lattice-Boltzmann method has been ported to GPUs using the *parallel Standard Library* (pSTL) algorithms of C++17 in combination with NVIDIA's NVHPC compiler.

The simulation domain is resolved by about $220 \cdot 10^6$ cells, using mesh resolutions of $\Delta x = 0.1\ mm$. The grid resolution is finer than the spatial resolution of the CT data to resolve all relevant flow features to analyze the flow. Especially thin wall-bounded shear layers which are relevant to simulate the right heat transfer are resolved accurately this way [1]. The patient has undergone turbinectomy in the left nasal passage (from the patient's view).

The surgical intervention reduces the total pressure loss between the inlets (nostrils) and outlet (pharynx) from 10.02 Pa before the surgery to 7.11 Pa afterwards (-31%). At the same time, the temperature difference between the incoming air at the inlets and outlet is reduced from 304.8 K to 295.8 K (-3%), and the humidity concentration from 96.2% to 90.0% (-7%). Figure 3a illustrates the total pressure loss to the inflow areas for a cross-sectional area through the left nasal passage. It is clearly shown how the removal of parts of the inferior and middle turbinates widen the nasal passage and, therefore, decrease the pressure loss. Figure 3b shows the humidity distribution for the same



(a) Total pressure loss to the inflow areas (nostrils) for a cross-sectional area through the left nasal passage (from the patient's view) of the pre-surgical (left) and the post-surgical (right) cases.



(b) Humidity distribution for a cross-sectional area through the left nasal passage (from the patient's view) of the pre-surgical (left) and the post-surgical (right) cases.

cross-sectional area. The narrowed passages in the pre-surgical case allow closer contact between the incoming air and the airway-nose interface, and, therefore, an increased heat exchange and humidification. In contrast, the widening of the nasal passages in the course of the turbinectomy cause dry air streaming towards the back part of the airway-throat interface. This coincides with the patient's perception, who reported less efforts for breathing in, but at the same time a dry and sometimes painful feeling at the back of the throat.

4. Acknowledgements

The research leading to these results has been conducted in the HANAMI project, which receives funding from the European Union Horizon Europe Programme - Grant Agreement Number 101136269 under the call HORIZON-EUROHPC-JU-2022-INCO-04. The authors gratefully acknowledge the computing time granted by the JARA Vergabegremium and provided on the JARA Partition part of the supercomputer JURECA [14] at *Forschungszentrum Jülich*. The authors would like to thank the patient for providing the computer tomography data.

References

[1] A. Lintermann, M. Meinke, W. Schröder, Fluid mechanics based classification of the respiratory efficiency of several nasal cavities, Computers in Biology and Medicine 43 (11) (2013) 1833–1852. doi:10.1016/j.compbiomed.2013.09.003.

[2] M. Waldmann, A. Lintermann, Y. J. Choi, W. Schröder, Analysis of the Effects of MARME Treatment on Respiratory

Flow Using the Lattice-Boltzmann Method, 2020, pp. 853–863. doi:10.1007/978-3-030-25253-3_80.

[3] M. Rüttgers, M. Waldmann, W. Schröder, A. Lintermann, A machine-learning-based method for automatizing lattice-boltzmann simulations of respiratory flows, Applied Intelligence 52 (2022) 9080–9100. doi:10.1007/s10489-021-02808-2.

[4] M. Waldmann, M. Rüttgers, A. Lintermann, W. Schröder, Virtual surgeries of nasal cavities using a coupled lattice-boltzmann–level-set approach, Journal of Engineering and Science in Medical Diagnostics and Therapy 5 (3) (2022). doi:10.1115/1.4054042.

[5] M. Rüttgers, M. Waldmann, K. Vogt, J. Ilgner, W. Schröder, A. Lintermann, Automated surgery planning for an obstructed nose by combining computational fluid dynamics with reinforcement learning, Computers in Biology and Medicine 173 (2024) 108383. doi:10.1016/j.compbiomed.2024.108383.

[6] X. Chen, S. Leong, H. Lee, V. Chong, D. Wang, Aerodynamic effects of inferior turbinate surgery on nasal airflow - a computational fluid dynamics model, Rhinology 48 (2010) 394–400. doi:10.4193/Rhino09.196.

[7] J. Siu, K. Inthavong, J. Dong, Y. Shang, R. G. Douglas, Nasal air conditioning following total inferior turbinectomy compared to inferior turbinoplasty – a computational fluid dynamics study, Clinical Biomechanics 81 (2021) 105237. doi:10.1016/j.clinbiomech.2020.105237.

[8] A. Lintermann, M. Meinke, W. Schröder, Zonal flow solver (ZFS): a highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34 (7-8) (2020) 458–485. doi:10.1080/10618562.2020.1742328.

[9] L. Li, R. Mei, J. F. Klausner, Boundary conditions for thermal lattice boltzmann equation method, Journal of Computational Physics 237 (2013) 366–395. doi:10.1016/j.jcp.2012.11.027.

[10] S. Hanida, F. Mori, K. Kumahta, M. Watanabe, S. Ishikawa, T. Matsuzawa, Influence of latent heat in the nasal cavity, Journal of Biomechanical Science and Engineering 8 (3) (2013) 209–224. doi:10.1299/jbse.8.209.

[11] K. Inthavong, D. F. Fletcher, M. Khamooshi, S. Vahaji, H. Salati, Wet surface wall model for latent heat exchange during evaporation, International journal for numerical methods in biomedical engineering 38 (4) (2022) e3581. doi:10.1002/cnm.3581.

[12] T. Keck, R. Leiacker, H. Riechelmann, G. Rettinger, Temperature profile in the nasal cavity, The Laryngoscope 110 (4) (2000) 651–654. doi:10.1097/00005537-200004000-00021.

[13] S. Ito, Implementation of a humidity transport equation for the analysis of the nasal airfow using a lattice-boltzmann method, Master's thesis, Institute of Aerodynamics and Chair of Fluid Mechanics, RWTH Aachen University (2022).

[14] D. Krause, P. Thörnig, JURECA: Modular supercomputer at Jülich Supercomputing Centre, Journal of Large-scale Research Facilities 4 (2018) A132. doi:10.17815/jlsrf-4-121-1.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Fundamental Study on Fluid-Structure Interaction Models for Pulse Waveform Analysis Through Blood Vessels

Yuki Kaneko[a], Tomohiro Fukui[a,*]

[a]Kyoto Institute of Technology, Matsugasaki Hashikamicho, Sakyō-ku, Kyōto, 606-0951, Japan

ARTICLE INFO[†]

Keywords:
Atherosclerosis;
Pulse Wave Propagation;
Fluid-Solid Interactions;
Blood Vessels;
Blood Flow;
Arbitrary Lagrangian Eulerian;
Wave Reflection

ABSTRACT

Atherosclerosis is one of the major risk factors for vascular diseases, and its preferable sites suggest that development of atherosclerosis may be related to hemodynamics. The relationship between pulse wave velocity (PWV) and atherosclerosis has long been discussed for non-invasive diagnosis. The evaluation of the measured PWV as a diagnostic indicator for vascular diseases is based on the Moens-Korteweg equation. However, since this theoretical equation is derived based on many ideal assumptions, the evaluation of PWV measured in actual blood vessels using the Moens-Korteweg equation could be uncertain. Therefore, further understanding of the phenomenon of pulse wave propagation is necessary to make PWV a more reliable diagnostic indicator. In this study, to obtain a more reliable diagnostic indicator for atherosclerosis, the evaluation of the measured PWV and changes in pulse waveforms were investigated using a three-dimensional straight cylindrical model of the aorta.

1. Introduction

Atherosclerosis is one of the risk factors for vascular diseases such as heart disease and cerebrovascular disease,which are major causes of death worldwide. Atherosclerosis is a disease in which atherosclerotic plaques develop on the inner surface of arteries, eventually obstructing blood vessels and causing functional decline or arrest of vital organs [1]. One of the non-invasive methods for diagnosing atherosclerosis is measuring pulse wave velocity (PWV). Pulse wave propagation is a phenomenon in which oscillation in the arterial wall (pulse wave) caused by the blood pulsed by the heart's contraction propagate with blood flow to the periphery of the artery. This velocity is called pulse wave velocity, and when measured in vivo, it is obtained by dividing the distance L between two measurement points by the time delay Δt of the waveform.
Therefore, it is

$$PWV_{\mathrm{measure}} = \frac{L}{\Delta t}. \tag{1}$$

Note that $PWV_{measure}$ is calculated as the average velocity between two measurement points. When evaluating the measured $PWV_{measure}$ the Moens-Korteweg equation [2] is commonly used. According to the Moens-Korteweg equation, PWV is expressed as

$$PWV_{\mathrm{M\text{-}K}} = \sqrt{\frac{Eh}{2\rho r}}. \tag{2}$$

Here, E is the Young's modulus of the vessel wall, h is the thickness of the vessel wall, ρ is the blood density, and r is the inner diameter of the vessel. From this equation, it is predicted that changes in properties of the vessel wall due to conditions such as arterial wall thickening, decreased elasticity, or the development of stenosis associated with arteriosclerosis, lead to an increase in PWV. However, the Moens-Korteweg equation is derived based on idealized assumptions and remarkably deviates from the geometric and mechanical environments in actual biological systems. Therefore, diagnosing arterial sclerosis by $PWV_{measure}$ based on $PWV_{M\text{-}K}$ could be uncertain. In addition, $PWV_{measure}$ evaluates only velocity of wave propagation and does not consider many other important biological factors. Therefore, in order to enhance the reliability of $PWV_{measure}$ as a diagnostic indicator, a new evaluation index that considering dynamics of the pulse waveforms are necessary. In this study, $PWV_{measure}$ was firstly evaluated to verify whether the computational models and schemes can be applied to future pulse wave analysis, and then pulse wave analysis was conducted to make $PWV_{measure}$ a more reliable diagnostic indicator for arteriosclerosis.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02480 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
m3623005@edu.kit.ac.jp (Y. Kaneko); fukui@kit.ac.jp (T. Fukui)
ORCID(s): - (Y. Kaneko); - (T. Fukui)

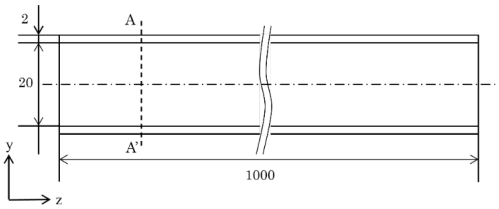


Figure 1: Arterial model.

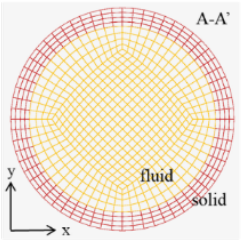


Figure 2: Cross section A-A' of arterial model.

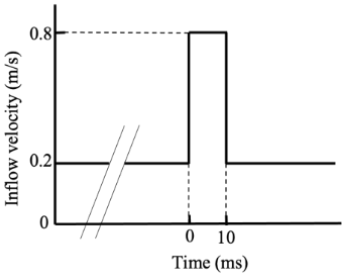


Figure 3: Boundary condition at the inlet of fluid.

2. Methods

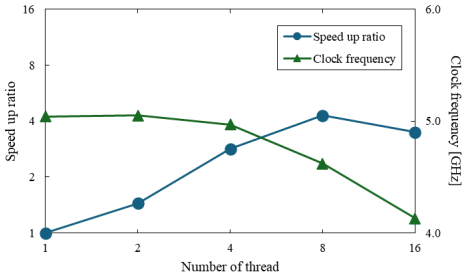
In this study, the nonlinear structural analysis solver, RADIOSS version 2022.3 by Altair, was used to perform fluid-structure interaction analysis using the Arbitrary Lagrangian-Eulerian (ALE) formulation. A three-dimensional straight cylindrical tube model of the aorta was used for the simulation model as shown in Fig. 1+2. The outer part was composed of solid elements simulating the vessel wall, while the inner part composed of fluid elements simulating blood. One cross section contained 256 elements in the solid domain and 512 elements in the fluid domain. The structural part was modeled using isotropic linear elastic material, while the fluid part was modeled using viscous fluid material. The governing equations used were the equilibrium equations for the structural part and the Navier-Stokes equation and the continuity equation for the fluid part. For the boundary conditions, a single rectangular wave was introduced into the fluid part at the inlet, as shown in Fig. 3, to generate a compression wave. Both ends of the structural part were fixed for displacement and rotation, and a no-slip condition was applied at the interface between the structure and fluid parts. The Young's modulus of the blood vessel wall was set to $E = 1.5MPa$, the Poisson's ratio to $\nu = 0.45$, and the initial density to $\rho_i^s = 1000kg/m^3$. The kinematic viscosity of the blood was set to $\nu = 4.0 \cdot 10^{-6}m^2/s$, the sound speed to $C = 80m/s$, and the initial density to $\rho_i^f = 1000kg/m^3$. The specifications of the PC used for the simulations are Intel core i9-13900: clock frequency, 3.0 GHz; number of cores, 24; memory size, 128 GB.

3. Results and discussions

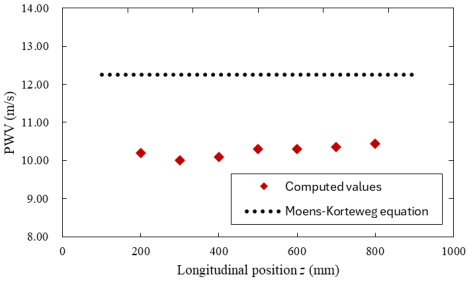
Figure 4a shows the efficiency of the parallel computation. The efficiency of the parallel computation increased the most at 8 parallel processes. As the number of threads increased, the clock frequency gradually decreased to prevent the processor from overheating. However, with 8 parallel processes, the clock frequency remained relatively high compared to other parallel processes. On the other hand, with 16 parallel processes, the clock frequency significantly decreased, and the increased overhead of data communication between computation nodes led to a decrease in efficiency.

Figure 4b shows the computed results of local $PWV_{measure}$ in the blood vessels. $PWV_{measure}$ was calculated using the phase velocity method [3]. This method involves shifting one of the observed waveforms between any two measurement points, aligning the rising edges of the waveforms, and measuring the distance moved as the time delay Δt. $PWV_{measure}$ was then calculated using this time delay Δt and the distance L between the measurement points. At the location z=200 mm, the local $PWV_{measure}$ was determined from the two velocity waveforms at $z = 150$ mm and 250 mm, and this methodology was applied across other locations as well. The results indicate that the maximum error between the calculated $PWV_{measure}$ values and theoretical values was found to be 18.4 %. Hence, further verification is necessary in future studies.

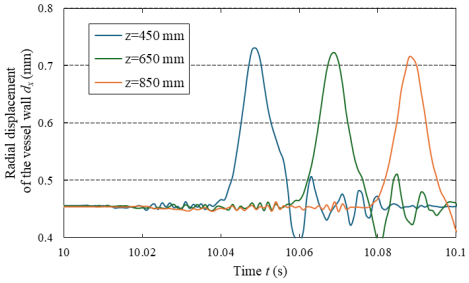
For pulse wave analysis, both volume pulse waves and velocity pulse waves are necessary. Therefore, in this study, the changes in amplitude and shape of both pulse waves toward the peripheral regions were investigated. Figure 4c shows the radial displacement waveforms of the vessel wall at different locations, while Fig. 4d shows the centerline



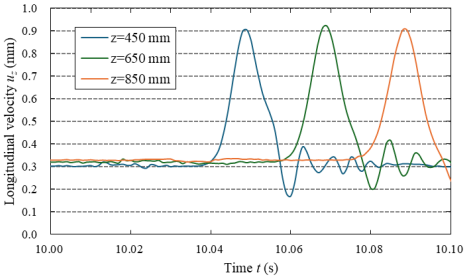
(a) The efficiency of the parallel computation.



(b) PWV values at each point.



(c) Wall displacement waveforms at each point.



(d) Velocity waveforms at each point on the center line.

Figure 4: Figures illustrating computational efficiency, PWV values, wall displacement, and velocity waveforms.

velocity waveforms at different locations. From the results, it can be observed that as the pulse wave propagates distally, the upstroke became less steep and the amplitude decreased. This behavior was attributed to the effect of fluid viscosity dissipation.

4. Conclusions

In order to analyze pulse waves using fluid-structure interaction, the *PWV* of the simulation model was evaluated. The calculated PWV was estimated lower than the theoretical value. In the future work, the simulation model will be validated to accurately analyze pulse waves, and pulse wave analysis will be conducted to deepen the understanding of pulse wave propagation.

References

[1] The Japan Society of Mechanical Engineers, Biomechanics Numerical simulation, Corona Publishing Co., LTD., 1999.

[2] J. C. Bramwell, A. V. Hill, The velocity of the pulse wave in man, Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character 93 (1922) 298–306. doi:10.1098/rspb.1922.0022.

[3] D. A. McDonald, Regional pulse wave velocity in the arterial tree, Journal of Applied Physiology 24 (1) (1968) 73–78. doi:10.1152/jappl.1968.24.1.73.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

GPU Accelerated FEM-Based Lagrangian Particle Tracking Framework for Human Air Pathway

Thivin Anandh[a,*], Sashikumaar Ganesan[a]

[a]*Indian Institute of Science, Department of Computational and Data Sciences, Bangalore, 560012, India*

ARTICLE INFO[†]

Keywords:
Finite Element Method;
Graphics Processing Unit Accelerated Particle Tracking;
Flow Modeling in Human Air Pathway;
Lagrangian Particle Tracking

ABSTRACT

Efficient particle deposition modeling is crucial for understanding regional particle deposition effects on human airways. Lagrangian particle tracking in large, complex domains presents two primary challenges: high computational costs and the requirement for a robust framework with effective interpolation routines and comprehensive mesh information. Finite Element Method (FEM) routines, utilized for fluid modeling, are ideal due to their interpolation capabilities and data structures, which store mesh and geometric information essential for tracking and identifying particle deposition status. However, FEM data structures are complex and usually reside on CPUs, making it challenging to port them to GPUs. Despite this, GPUs offer significant parallelization potential for particle tracking if these data structures can be efficiently managed. In this work, we introduce a GPU-accelerated Lagrangian particle deposition framework utilizing FEM-based routines. Our approach focuses on efficient transfer and simplification of FEM data structures from CPU to GPU, facilitating the implementation of zonal-based particle searching and inertial deposition techniques directly on the GPU. Our model demonstrated lower sequential execution time than ANSYS Fluent's particle tracking module and achieved a 100x speedup over the Sequential CPU implementation and a 4x speedup over the OpenMP implementation for number of particles up to 5 Million. The GPU-accelerated framework reduces execution time from days to hours for complex geometries, enabling deeper exploration of particle deposition in human airways.

1. Introduction

Modeling aerosol deposition in the human airway is crucial for understanding regional particle deposition, aiding the development of targeted drug delivery systems [1, 2, 3, 4]. In-vivo methods are limited by safety and regulatory issues, especially with radioactive aerosols [5, 6, 7]. In-vitro studies show that slight variations in geometry and surface irregularities from fabrication significantly impact aerosol deposition [8, 9, 10]. A wide range of fluid flow solvers, including the finite volume method (FVM) [11], finite element method (FEM) [12], and lattice Boltzmann method (LBM) [13] have been employed to simulate fluid flow in bounded domains. These solvers incorporate turbulence models such as Reynolds–averaged Navier–Stokes (RANS) [14, 12] and large eddy simulation (LES) [14, 12]. In a study by [13], the authors aimed to enhance particle simulations using GPUs, focusing on the lattice Boltzmann method (LBM). However, this approach lacked particle modeling and experimental validation, limiting comprehensive evaluation and practical applicability. In another work by [15], efforts were made to parallelize particle deposition within the FEM framework. However, this parallelization was achieved exclusively using CPU resources.

2. Methodology

2.1. Fluid Modeling

The framework was implemented using our in-house FEM package, ParMooN [16]. We solved the unsteady 3D Navier-Stokes equations to simulate fluid flow in the human airway. Turbulence modeling based on the Variational Multiscale Method (VMS) was applied for the REYNOLDS number 3,725. For further details on VMS, see [17]. The computational mesh used in this study, as shown in Fig. 1, consists of 300,000 tetrahedral cells.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02481 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
thivinanandh@iisc.ac.in (T. Anandh); sashi@iisc.ac.in (S. Ganesan)
ORCID(s): 0000-0003-4969-3242 (T. Anandh); 0000-0003-1858-3972 (S. Ganesan)

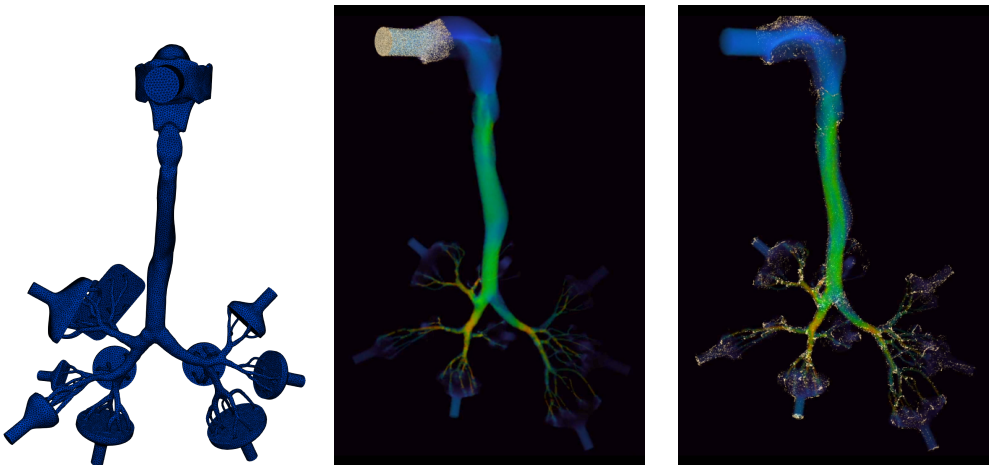


Figure 1: Computational mesh, Particle deposition at 3k and 6k timestep with velocity heat maps.

2.2. Particle Modeling

The equations of motion for the particles [12] are given by

$$m_p \frac{d\mathbf{u}_p}{dt} = \frac{3}{4}\frac{\rho_f}{\rho_p}\frac{m_p}{d_p}\frac{C_D}{C_C}\left|\mathbf{u}_f - \mathbf{u}_p\right|\left(\mathbf{u}_f - \mathbf{u}_p\right) + m_p\mathbf{g}\frac{\rho_p - \rho_f}{\rho_p} + \mathbf{F}_B, \tag{1}$$

where $\mathbf{u}_p$, $\mathbf{u}_f$, ρ_p, ρ_f, represent the particle velocity, fluid velocity, particle density, and fluid density, respectively. The terms in the right-hand side are drag force, gravitational force and Brownian force, respectively.

3. GPU Implementation

The main concept and parallelization scheme for GPU-accelerated particle tracking are illustrated in Fig. 2. Before the simulation begins, all essential data, including simplified FEM data structures and mesh information, are transferred from the CPU to the GPU and stored there. The particle information is initialized on the CPU and then transferred to the GPU. At each timestep, the corresponding fluid solution is read from the stored file and transferred to the GPU for computation. In the GPU, each thread is assigned to track a particle within the domain. This thread is responsible for solving the particle equations of motion, interpolating velocity values at the particle's current position, performing zonal searches to compute the updated particle position within the domain, and calculating the particle deposition parameters. After all threads complete the computation, a CUDA-synchronization call is made on the CPU side. If necessary, particle details can be transferred back to the CPU from GPU for visualization or logging purposes. The fluid solution for the next timestep is then read, and the process is repeated until all particles are deposited or have escaped.

3.1. Adaptation of FEM Data Structures for GPU

FEM utilizes complex nested data structures; for example, a cell class may contain edges, and an edge class may contain vertices. Copying these structures to GPUs is complex and introduces computational overhead, especially in multithreaded GPU environments constrained by memory. The core idea of our implementation was to transfer essential data as primitive arrays from CPU to GPU, enabling the implementation of FEM-based algorithms for deposition, interpolation, and tracking routines.

3.2. Inertial Deposition Algorithm

The deposition logic triggers when a particle moves outside the computational domain. We determine the deposition point by analyzing the boundary face of the preceding cell and constructing a vector between the previous and current particle positions. The intersection of this vector with the current boundary surface identifies the deposition point, as shown in Fig. 3. Using the simplified FEM data structures, we compute the intersection and orientation of the boundary face in 3D using the simplified boundary face and cell connectivity information transferred from CPU to GPU.

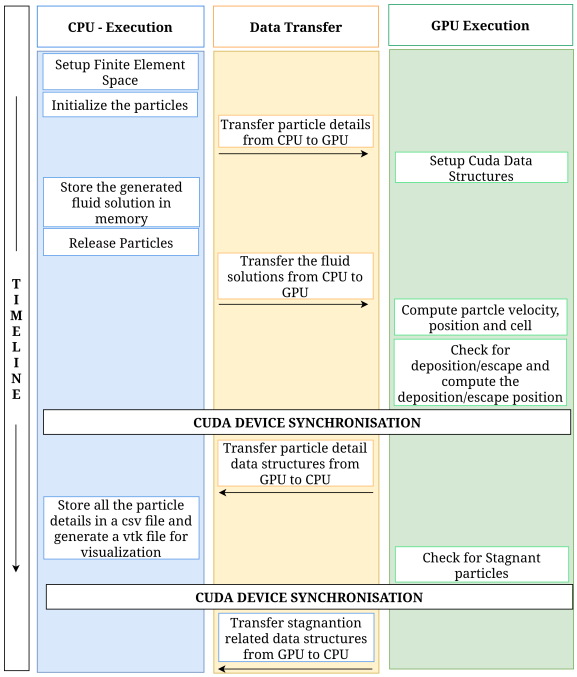


Figure 2: Timeline of GPU accelerated particle deposition.

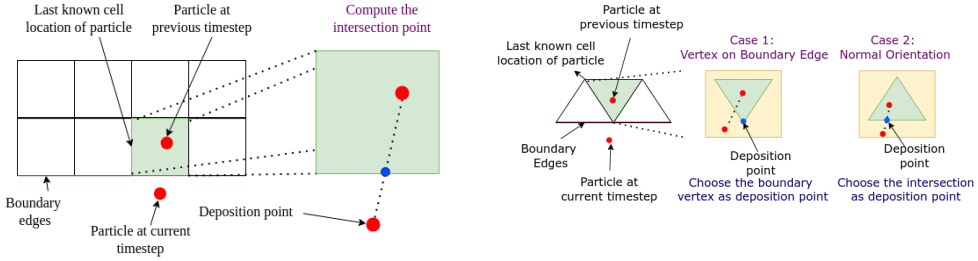


Figure 3: Deposition logic for structured and unstructured cells.

3.3. Zonal Particle Search

In a zonal search approach, we need the data structures of the mesh to compute the neighboring cells of the current cell. This results in searching for the particle within the zonal neighborhood rather than the entire domain. This information is necessary to interpolate the underlying velocity value at the current particle position, which is required for solving the particle equations of motion. The neighboring information can be obtained from the FEM data structures available within the CPU, which hold information such as cells, their faces, vertices, and neighbor information, see Fig. 4 for an example. However, using these data structures to compute the neighboring cells in the GPU presents two main challenges. First, these data structures are complex and will add unnecessary memory load when copied to the GPU. Second, computing these neighbors on-the-fly within the

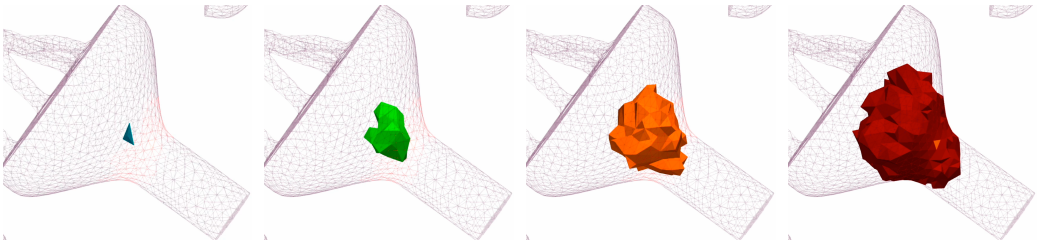


Figure 4: Zonal neighbors visualized at various zonal levels for a sample cell within the domain.

N_Particles	Sequential	OMP	CUDA	OMP Speedup	GPU Speedup
100K	0.219	0.01	0.002	21	88
1M	2.166	0.139	0.022	16	98
2M	4.445	0.18	0.044	25	102
5M	11.101	0.433	0.101	26	**110**

Table 1
Time taken and speedup for various number of particles.

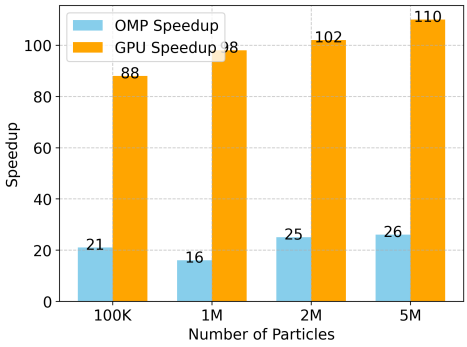


Figure 5: GPU vs. OMP speedup for various number of particles.

GPU using these complex data structures is computationally expensive, as each GPU thread responsible for updating a particle's position must traverse numerous data structures to calculate the neighboring cells.

We compared our sequential implementation with an OpenMP-based multi threaded implementation and our GPU-based implementation. As shown in Tab. 1 and Fig. 5, the GPU implementation achieves a speedup of 100 times compared to the sequential implementation.

Pre-computing this neighboring information externally and transferring it to the GPU is also memory-intensive.

Zonal Levels	Min	Median	Max
1	14	61	114
2	30	210	558
3	31	447	1376

Table 2
Statistics of number of neighboring cells for a given cell in the domain up to that zonal level.

As seen from Tab. 2, each cell may have close to 1,000 level-3 neighbors, resulting in storing thousands of cell IDs as neighboring information for each cell in a domain with millions of cells. This can create an array that uses a large amount of memory, making it challenging to transfer to the GPU. Therefore, we need a simple data structure that can be transferred to the GPU and used to perform minimal calculations in computing the zonal neighbors of the cell.

We observed that the raw mesh file, which provides information about which cells and vertices are connected, can be transformed into an adjacency matrix through proper calculations on the CPU. In this matrix, each row represents a cell, and the column values represent connectivity to other cells in the domain. Furthermore, the resulting adjacency matrix is sparse, allowing it to be stored in a Compressed Sparse Row (CSR) format. This format significantly reduces the memory footprint, making it easier to transfer to the GPU and use to calculate neighboring cells according to the algorithm shown in Alg. 1. Each GPU thread will perform Alg. 1 to compute the zonal neighbors of its respective particle's current cell.

Algorithm 1 Breadth-First Search (BFS) for Neighboring Cells using CSR-based Adjacency Matrix for CUDA implementation.

```
// Create local arrays to store cells and depths for the queue
int queue_cell_no[], queue_depth[], searched_cells[], queue_start_cursor ← 0, queue_end_cursor ← -1, searched_cells_counter ← 0
// Add the current cell to the queue
queue_end_cursor ← queue_end_cursor + 1, queue_cell_no[queue_end_cursor] ← cell_no, queue_depth[queue_end_cursor] ← 0
while queue_start_cursor ≤ queue_end_cursor not inside_domain do
    // Pop the cell from the queue
    current_cell ← queue_cell_no[queue_start_cursor]
    current_depth ← queue_depth[queue_start_cursor]
    queue_start_cursor ← queue_start_cursor + 1
    // Loop through neighbour cells at current level
    start_index ← d_m_row_pointer[current_cell]
    end_index ← d_m_row_pointer[current_cell + 1]
    // For cells in current level
    for index ← start_index to end_index - 1 do
        neighbour_cell ← d_m_col_index[index] // Get the neighbor cell
        is_searched ← false
        // Check if the neighbor cell is already searched
        for k ← 0 to searched_cells_counter - 1
            if neighbour_cell = searched_cells[k]
                is_searched ← true
                break
        if is_searched then
            continue
        // Check if particle is inside current neighbour cell
        bool inside_neighbour_cell ← Is_Point_In_Cell_CUDA()
        searched_cells_counter ← searched_cells_counter + 1
        searched_cells[searched_cells_counter] ← neighbour_cell
        if inside_neighbour_cell
            inside_domain ← true // to get out of while loop
            d_m_previous_cell[tid] ← d_m_current_cell[tid] // update cell information
            d_m_current_cell[tid] ← neighbour_cell
            break
        if current_depth + 1 < search_depth
            // Add the neighbour cell to the queue to be searched
            queue_end_cursor ← queue_end_cursor + 1
            queue_cell_no[queue_end_cursor] ← neighbour_cell
            queue_depth[queue_end_cursor] ← current_depth + 1
    end for
end while
```

Reference	Mesh Size	Number of Particles	Time taken(s) per timestep
ANSYS Fluent	0.2 Million	100K	0.6
Our Implementation	0.3 Million	100K	**0.159**

Table 3
Comparison of sequential timings.

4. Results

To validate the efficiency of our sequential implementation, we compared the time our code takes to track particles (sequential implementation with one OpenMP thread) with ANSYS Fluent on a single CPU. This helps show that our code's baseline version is optimal when compared with the existing implementations. The Tab. 3 shows the time taken for sequential execution of the ANSYS fluent module and our sequential implementation. It's important to note that ANSYS Fluent has its own way of creating meshes (meshing

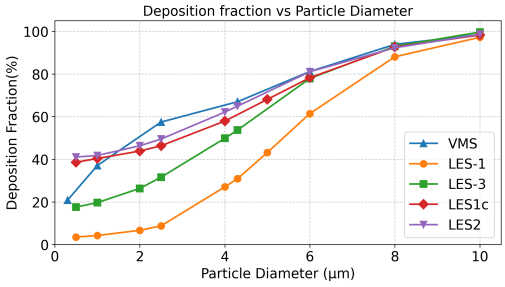


Figure 6: Deposition fraction for various particle sizes. VMS is the deposition fraction from our GPU accelerated model. All other models are from [12].

routines), So a mesh created within ANSYS with 200K elements was used for calculating the time taken for particle tracking. However, our in-house solver cannot work with meshes exported from ANSYS. Because of this, we used our own mesh (with 300,000 elements) generated externally and measured the particle tracking time. Both our code and ANSYS Fluent used the same number of particles. Even though ANSYS Fluent has fewer cells (elements) in its mesh, it still took longer to track particles than our framework. Additionally, we compared our deposition fraction results with those from [12], showing a strong correlation for larger particle sizes, as depicted in Fig. 6. For smaller particle sizes, a more refined mesh is necessary to accurately capture the fluid dynamics.

Acknowledgements

The authors acknowledge Lokesh Mohanty (Indian Institute of Science, Bangalore) and Varun Vaidyanathan (Indian Institute of Technology, Delhi) for their assistance in conducting the experiments for this study.

References

[1] K. Kadota, A. Imanaka, M. Shimazaki, T. Takemiya, K. Kubo, H. Uchiyama, Y. Tozuka, Effects of inhalation procedure on particle behavior and deposition in the airways analyzed by numerical simulation, Journal of the Taiwan Institute of Chemical Engineers 90 (2018) 44–50. doi:10.1016/j.jtice.2017.11.008.

[2] J. Kimbell, R. Segal, B. Asgharian, B. Wong, J. Schroeter, J. Southall, C. Dickens, G. Brace, F. Miller, Characterization of deposition from nasal spray devices using a computational fluid dynamics model of the human nasal passages, Journal of Aerosol Medicine 20 (2007) 59–74. doi:10.1089/jam.2006.0531.

[3] K. Inthavong, Z. Tian, J. Tu, W. Yang, C. Xue, Optimising nasal spray parameters for efficient drug delivery using computational fluid dynamics, Computers in Biology and Medicine 38 (2008) 713–726. doi:10.1016/j.compbiomed.2008.03.008.

[4] A. Gambaruto, E. Olivares, H. Calmet, G. Houzeaux, A. Bates, D. Doorly, Transport and deposition in the upper human airways during a sniff, in: Computational Engineering and Science for Safety and Environmental Problems COMPSAFE2014, Sendai (Japan), 2014.

[5] K. Cheng, Y. Cheng, H. Yeh, R. Guilmette, S. Simpson, Y. Yang, D. Swift, In vivo measurements of nasal airway dimensions and ultrafine aerosol deposition in the human nasal and oral airways, Journal of Aerosol Science 27 (1996) 785–801. doi:10.1016/0021-8502(96)00029-8.

[6] J. Kesavan, R. Bascom, B. Laube, D. Swift, The relationship between particle deposition in the anterior nasal passage and nasal passage characteristics, Journal of Aerosol Medicine 13 (2000) 17–23. doi:10.1089/jam.2000.13.17.

[7] G. Kanapilly, O. Raabe, C. Goh, R. Chimenti, Measurement of in vitro dissolution of aerosol particles for comparison to in vivo dissolution in the lower respiratory tract after inhalation, Health Physics 24 (1973) 497–507. doi:10.1097/00004032-197305000-00004.

[8] Z. Zhang, C. Kleinstreuer, Computational analysis of airflow and nanoparticle deposition in a combined nasal–oral–tracheobronchial airway model, Journal of Aerosol Science 42 (2011) 174–194. doi:10.1016/j.jaerosci.2011.01.001.

[9] G. Garcia, E. Tewksbury, B. Wong, J. Kimbell, Interindividual variability in nasal filtration as a function of nasal cavity geometry, Journal of Aerosol Medicine and Pulmonary Drug Delivery 22 (2009) 139–156. doi:10.1089/jamp.2008.0713.

[10] E. Frederix, A. Kuczaj, M. Nordlund, M. Belka, F. Lizal, J. Jedelsky, J. Elcner, M. Jicha, B. Geurts, Simulation of size-dependent aerosol deposition in a realistic model of the upper human airways, Journal of Aerosol Science 115 (2018) 29–45. doi:10.1016/j.jaerosci.2017.10.007.

[11] T. Gemci, V. Ponyavin, Y. Chen, H. Chen, R. Collins, Computational model of airflow in upper 17 generations of human respiratory tract, Journal of Biomechanics 41 (2008) 2047–2054. doi:10.1016/j.jbiomech.2007.12.019.

[12] P. Koullapis, S. Kassinos, J. Muela, C. Perez-Segarra, J. Rigola, O. Lehmkuhl, Y. Cui, M. Sommerfeld, J. Elcner, M. Jicha, I. Saveljic, N. Filipovic, F. Lizal, L. Nicolaou, Regional aerosol deposition in the human airways: The simhale benchmark case and a critical assessment of in silico methods, European Journal of Pharmaceutical Sciences 113 (2018) 77–94. doi:10.1016/j.ejps.2017.09.003.

[13] T. Miki, X. Wang, T. Aoki, Y. Imai, T. Ishikawa, K. Takase, T. Yamaguchi, Patient-specific modelling of pulmonary airflow using GPU cluster for the application in medical practice, Computer Methods in Biomechanics and Biomedical Engineering 15 (2012) 771–778. doi:10.1080/10255842.2011.560842.

[14] A. Kolanjiyil, C. Kleinstreuer, Computationally efficient analysis of particle transport and deposition in a human whole-lung-airway model. part i: Theory and model validation, Computers in Biology and Medicine 79 (2016) 193–204. doi:10.1016/j.compbiomed.2016.10.020.

[15] E. Olivares Mañas, Parallel lagrangian particle transport: application to respiratory system airways, Ph.D. thesis, Universitat Politècnica de Catalunya (2018).

[16] U. Wilbrandt, C. Bartsch, N. Ahmed, N. Alia, F. Anker, L. Blank, A. Caiazzo, S. Ganesan, S. Giere, G. Matthies, R. Meesala, A. Shamim, J. Venkatesan, V. John, ParMooN—A modernized program package based on mapped finite elements, Computers & Mathematics with Applications 74 (1) (2017) 74–88. doi:10.1016/j.camwa.2016.12.020.

[17] B. Pal, S. Ganesan, A finite element variational multiscale method for computations of turbulent flow over an aerofoil, International Journal of Advances in Engineering Sciences and Applied Mathematics 7 (1–2) (2015) 14–24. doi:10.1007/s12572-015-0126-1.

Mini-Symposium 6:

Mini-Symposium on Tool Support for Developing Highly-Parallel CFD Applications

Organizers: Jana Gericke, Ronny Tschüter, and Immo Huismann

Code complexity of parallel Computational Fluid Dynamics (CFD) solvers has seen tremendous growth in recent years: expanding feature sets and more complex hardware, such as accelerators and hierarchical memories, took their toll. This can be partially mitigated by abstraction, for instance as offered by third-party libraries, but at the expense of larger and more intricate software stacks that need to be managed. The conjunction of abstraction, code complexity, and growing software stacks complicates code analysis and, in turn, leads to performance problems potentially remaining undiscovered and unfixed. Consequently, tools support is indispensable in various aspects throughout the software engineering life-cycle to support both developers and users.

This mini-symposium aims at gathering developers and users of software tools assisting in the development of sophisticated, highly-parallel HPC software for CFD. It provides a platform for experts of different fields empowering discussion and knowledge transfer to achieve the overarching goal: raising reproducibility, automation, and documentation during the whole software-engineering life-cycle of CFD applications on high-performance computers. The tools are ranging from performance analysis, over debugging of HPC codes, to management of the involved software stacks.

A list of potential contributions may include, but it is not limited to:

- Tools for automation in the software development life-cycle
- Workflows for automated testing and deploying of software projects
- Methods and tools for debugging and correctness checking of highly-parallel applications
- Tool support for (node-level) performance analysis
- User success stories of utilizing tools

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Practical Empirical Performance Modeling for CFD Applications Using Extra-P

Lukas Rothenberger[a,*], Gustavo de Morais[a], Alexander Geiß[a] and Felix Wolf[a]

[a]*Technical University of Darmstadt, Department of Computer Science, Laboratory for Parallel Programming, Hochschulstr. 10, 64289 Darmstadt, Germany*

ARTICLE INFO[†]

Keywords:
Empirical Performance Modeling;
Instrumentation Filtering;
OpenFOAM

ABSTRACT

Performance models facilitate the development and optimization of scalable applications for HPC systems. Automatic empirical performance modeling allows the creation of performance models for CFD applications and other large software suites, although challenges regarding profiling time and model accuracy arise from their size and characteristics. To mitigate these challenges, we propose an automatic tool for Score-P filter file creation. We measure exemplary OpenFOAM CFD applications using these filter files and employ Extra-P to generate strong-scaling performance models and identify scalability bottlenecks.

1. Introduction and motivation

Understanding performance at scale and identifying potential bottlenecks are crucial for developing and optimizing efficient HPC applications. While computation- and communication-intensive kernels/functions are typically well understood, implicit performance bottlenecks, such as those arising from caching or synchronization effects, can be easily overlooked. These aspects typically only manifest as significant issues once the scale of the utilized HPC system reaches a certain magnitude. To improve the utilization of computational resources and consequently reduce energy consumption, it is important to identify such bottlenecks as early as possible in the development or optimization process, utilizing a minimal amount of computational resources. This is particularly critical when dealing with large, highly configurable code bases, such as modern CFD solvers. Mathematical performance models are suitable for examining scaling behavior and identifying potential bottlenecks. However, designing these models analytically for an entire large code base is often impractical due to the manual effort required. On the other hand, automatic empirical performance modeling offers a solution to circumvent the challenges of analytical modeling.

For instance, Extra-P [1] is a state-of-the-art performance modeling tool that leverages profiling results from program executions with relatively small yet representative input data. It generates performance models for entire code bases or instrumented sections without requiring significant manual effort aside from code instrumentation. However, particularly in the case of CFD applications, the runtime and memory overhead introduced by profiling and model creation can become significant due to the sizes of modern software suites. Filtering the profiled code regions can mitigate this problem, but the definition of suitable filters might be seen as a challenge on its own. This observation led to the development of a fast, automatic, and easy-to-use tool for creating Score-P [2] filter files using a single profile. With the overarching goal of reducing the resource consumption in HPC, it is the purpose of this work to motivate and introduce the use of empirical performance models for code development and optimization by example of the open source CFD software OpenFOAM v2312 [3], the performance-modeling tool Extra-P, and the Score-P profiling framework.

2. Empirical performance modeling with Extra-P

Extra-P is a tool to automatically generate performance models for HPC applications based on small-scale measurements that offer insights similar to those of analytical performance models. It considers a set of execution parameters, such as the number of processors or input sizes. Using measurement tools like Score-P, the empirical data is gathered for various combinations of the execution parameters. Subsequently, this data is modeled with Extra-P, which returns models based on the Performance Model Normal

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02482 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
lukas.rothenberger@tu-darmstadt.de (L. Rothenberger); gustavo.morais@tu-darmstadt.de (G. de Morais); alexander.geissl@tu-darmstadt.de (A. Geiß); felix.wolf@tu-darmstadt.de (F. Wolf)
ORCID(s): 0009-0005-8988-5427 (L. Rothenberger); 0000-0002-2139-1157 (G. de Morais); 0000-0003-4565-422X (A. Geiß); 0000-0001-6595-3599 (F. Wolf)

Form (PMNF). The PMNF is a general structure of performance models, which describes the impact of the execution parameters on a performance metric (e.g., time) as a human-readable equation using a combination of monomial and logarithmic terms. These models help unveil performance bottlenecks as the application scales.

When using Extra-P for scaling analysis of applications, one has to be aware that it is designed with a focus on weak scaling. That means Extra-P models the runtime for a single average process, given the number of processes and a problem size scaled with the number of processes. Consequently, we cannot directly model the runtime in a strong-scaling scenario, as this would have the same constant problem size for each number of processes. Instead, we can model the overall effort in the form of the sum of all processes' runtimes; this metric behaves similarly to the runtime of a single average process in the weak-scaling scenario so that it can be modeled using Extra-P.

3. Automatic generation of Score-P filter files

Score-P instruments code during compilation to measure the execution time of functions, but these measurements also include the overhead associated with the instrumentation. This overhead can disturb the performance modeling, especially for frequently accessed short-running functions/regions. A solution to reduce the instrumentation overhead is to filter the functions or kernels that necessitate performance measurements (i.e., those responsible for a substantial fraction of the total runtime). This improves the model accuracy while also reducing the time and memory required to generate the performance models. Since manually creating filter files is mostly impractical for large applications, we have developed a fast, user-friendly, and freely available tool to automatically generate Score-P filter files based on one existing Score-P profile.

3.1. Filter generation

Our design for the automatic Score-P filter generator[1] is inspired by prior research on hotspot detection [4]. Our approach examines all call paths in a single measurement to identify functions essential for modeling and those that may disrupt the modeling process. Most relevant for modeling are functions that contribute significantly to the total runtime. Conversely, functions that are frequently visited but have short runtimes potentially disrupt the modeling process. From these observations, we developed the following procedure for filtering call paths: call paths with a runtime per visit in the top 25% are always included. This criterion ensures that call paths contributing significantly to the runtime are retained. To address the potential exclusion of call paths where individual calls have a short runtime but collectively are in the top 25% of runtimes, we include the first parent call-tree node with the number of visits below the median. Once we have identified these two sets of included call paths, we also include all prefixes of the call paths to create a more comprehensive and meaningful call tree.

3.2. Effects of filtering on OpenFOAM

To evaluate the effects of automatically generated filter files in mitigating the impact of overhead on the profiling time, we have conducted the measurements described in the following on the basis of the Cavity3D-1M[2] benchmark contained in the OpenFOAM HPC Benchmark Suite[3]. Four execution configurations were considered: (1) without source code profiling; (2) entailing full instrumentation of the source code; (3) instrumentation filtering during runtime; and (4) filtering at the time of instrumentation. Each experiment was conducted using 520 processors distributed across five nodes of the high-performance computer Lichtenberg 2 and repeated five times[4]. The measured times do not include the time required for preprocessing. Figure 1 illustrates the summed execution times across all used processors. While the full instrumentation led to an overall increase in computation time of roughly 1.34 times in median values, it decreased to 1.30 times and 1.05 times when using the automatically generated filter file during runtime and instrumentation, respectively. Additionally, we noticed significant reductions in memory overhead and execution

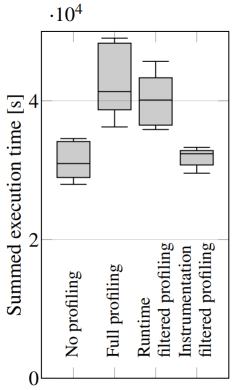


Figure 1: Effects of different filtering schemes on the required resources.

[1] https://github.com/extra-p/extrap/blob/master/tools/scorep_filter_generator.py

[2] 2024-05-14: https://develop.openfoam.com/committees/hpc/-/tree/develop/incompressible/icoFoam/cavity3D

[3] 2024-05-14: https://develop.openfoam.com/committees/hpc/-/tree/develop

[4] 2024-06-03: https://www.hrz.tu-darmstadt.de/hlr/hochleistungsrechnen/index.de.jsp

time required for performance model creation in case a filter file was used. During development, we observed similar reductions in profiling time for other data sets, configurations, and applications.

4. Performance modeling results

We gathered performance measurements for the same 3D lid-driven cavity flow example as above on the Lichtenberg 2 cluster at TU Darmstadt. We performed a strong-scaling analysis using a mesh size of 460^3, a limit of 3000 iterations, and 1, 2, 3, 4, 5, and 6 nodes with 104 processes per node. As mentioned, we cannot directly model the runtime for strong scaling; instead, we model the effort in core–seconds.

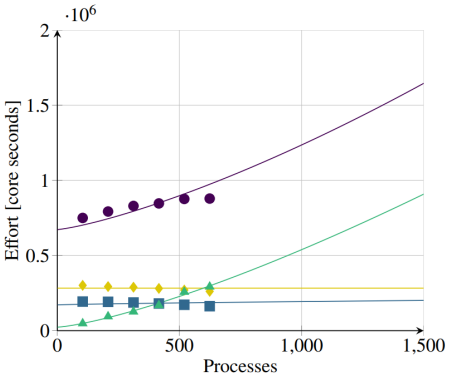


Figure 2: Effort performance models for the lid-driven cavity flow, created using measurements for 1-6 nodes.

Using this metric, perfect scaling would be represented by a constant model, meaning that the work is shared equally among all processes without any overhead. We see this in Fig. 2 for the `precondition` kernel, which dominates the performance of the cavity example up to 600 processes. The matrix multiplication `Amul` also contributes significantly. Beyond 600 processes, `MPI_Allreduce` of the `gSumProd` function overtakes all other kernels and dominates the performance. This also coincides with the results reported by Brogi et al. [5], who found that MPI dominates the performance when the number of cells per process gets smaller. Unlike the other functions, the `main` function's model is created by combining the individual models of all functions within its call path. As a result, the model for the `main` function deviates from the measurement at 624 nodes. Despite this deviation, the model accurately reflects the performance trend, displaying superlinear growth similar to the model and measurements for `MPI_Allreduce`, which exhibit the same behavior. In order to assess the quality of the created models, we extend the set of measurements by 8, 10, 12, and 14 nodes. The corresponding models are shown in Fig. 3. When comparing both figures, it is evident that Extra-P is able to identify the correct scaling behavior within a reasonable projection distance.

5. Conclusion

Performance modeling helps identify potential bottlenecks when scaling up. Unfortunately, crafting such models by hand requires expertise and is laborious for larger applications such as CFD solvers. An empirical modeling tool like Extra-P can simplify this task but requires measuring the application. In many measurements, short-running, often called functions, exhibit comparatively high run-to-run variations, which disturb the modeling process. Our filter generator automatically produces a filter file that prevents the instrumentation of small functions, thus reducing measurement overhead and improving the models. We used this approach for the Cavity3D benchmark of OpenFOAM and found that the `MPI_Allreduce` functions of the PDC solver are the main bottleneck.

Acknowledgements

The authors gratefully acknowledge the German Federal Ministry of Education and Research (BMBF) and the Hessian Ministry of Science and Research, Art and Culture (HMWK) for supporting this work as part of the NHR funding. Furthermore, the authors appreciate the computing time provided to them on the high-performance computer Lichtenberg at the NHR Center NHR4CES@TUDa. This is funded by the BMBF and the HMWK. The computations for this research were performed using computing resources under Project 2306.

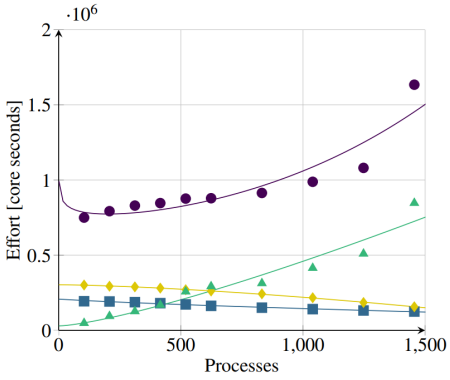


Figure 3: Effort performance models for the lid-driven cavity flow, created using an extended set of measurements.

References

[1] A. Calotoiu, T. Hoefler, M. Poke, F. Wolf, Using automated performance modeling to find scalability bugs in complex codes, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ACM, 2013, pp. 1–12. doi:10.1145/2503210.2503277.

[2] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, F. Wolf, Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir, in: Tools for high performance computing 2011, Springer, 2012, pp. 79–91. doi:10.1007/978-3-642-31476-6_7.

[3] H. Jasak, A. Jemcov, Z. Tukovic, et al., OpenFOAM: A C++ library for complex physics simulations, in: International workshop on coupled methods in numerical dynamics, Vol. 1000, 2007, pp. 1–20.

[4] S. A. Mohammadi, L. Rothenberger, G. de Morais, B. N. Görlich, E. Lille, H. Rüthers, F. Wolf, Filtering and ranking of code regions for parallelization via hotspot detection and OpenMP overhead analysis, in: Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W '23, ACM, New York, NY, USA, 2023, p. 1368–1379. doi:10.1145/3624062.3624206.

[5] F. Brogi, S. Bnà, G. Boga, G. Amati, T. Esposti Ongaro, M. Cerminara, On floating point precision in computational fluid dynamics using OpenFOAM, Future Generation Computer Systems 152 (2024) 1–16. doi:10.1016/j.future.2023.10.006.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

On the Modeling and Improvement of Sub-Optimal Submission Patterns on HPC Workloads

Anthony Larroque[a,*], Baptiste Giraud[a] and Antoine Dauptain[a]

[a]CERFACS, Algo-COOP, 42 Avenue Gaspard Coriolis, 31100 Toulouse, France

ARTICLE INFO[†]

Keywords:
High-Performance Computing;
Wall Clock Time;
Submission Patterns

ABSTRACT

Supercomputers are nowadays widely used in Computational Fluid Dynamics (CFD), Machine Learning (ML) and Artificial Intelligence (AI) applications. Running this kind of programs generally implies to request a certain number of Central Processing Units or Graphical Processing Units and to set a time limit for the job. However, supercomputers users are not always aware about all the implications of their submission patterns and some of them are sub-optimal for the resources management system. In particular, the requested time is crucial for supercomputers efficiency as a poor estimated requested time results in larger slowdown and waiting times for the users. Also, the AI and ML communities have different submission practices than the CFD practitioners making difficult to forecast the supercomputers usage in a close future. Whereas effort is usually done on the scheduler to enhance the user experience, for once, we worked on a practical solution to model and improve the collective behavior of users.

1. Introduction

With the development of computational resources, supercomputers become increasingly used for high-fidelity Computational Fluid Dynamics (CFD), Machine Learning (ML) or Artificial Intelligence (AI) applications. In order to run these simulations on a supercomputer, users generally request a number of Central Processing Units (CPUs) or Graphical Processing Units (GPUs) for a certain amount of time (= requested time or wall clock time).

However, many users do not really pay attention to the requested time and set them to the maximum allowed. This is sub-optimal in various ways for the supercomputer efficiency and the user experience.

Indeed, jobs that run much less time than the requested time could have benefited from the backfill mechanism on the supercomputers where it is allowed. Basically, when some resources are not used but planed for a job (job1) that is waiting for more resources, if another job (job2) can run on the free nodes without altering the start time of job1, then job2 will start to run on the free nodes. In 2002, Chiang *et al.* [1] studied the impact of more accurate requested job times on the production of a supercomputer. Results showed that when the requested times were more accurate, the slowdown (time between the submission and the end of the job divided by the time between the start and the end of the job) was decreased by a factor 2 to 6.

Another sub-optimal practice related is to run a job until the end of the requested time without writing an output at the end. Imagine a numerical simulation that requires 12 chained jobs running on 1,000 CPUs to reach a simulation time. If for each of these chained jobs, the solution to restart the next simulation was written 30 minutes before the end of the job, that means that $12 \cdot 0.5 \cdot 1{,}000 = 6{,}000$ CPU hours were completely wasted.

Finally, the rise of AI runs on clusters is shifting the pressure. AI users' habits are different, with notably node-sharing and interactive jobs. Indeed, contrary to CFD users that can tend to just launch the job on the maximum number of resources and time on the queue to reduce the time to solution, AI/ML users generally process graphically to fit their model or run massively shorter parallel jobs in order to get data for the model. The accuracy in the case of interactive graphical jobs is then directly dictated by whether the AI/ML user is satisfied by his model or not. With the migration of many CFD software on GPUs, hardware mainly used for AI/ML applications, AI/ML and "classical" users will collide in the future. We need solutions to forecast how the communities interact. The first step is an accurate data driven model of users' habits.

In our effort in the assessment of users, supercomputers usage habits are collected in the in-house open-source tool Seed. This collection has two purposes:

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02483 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author

larroque@cerfacs.fr (A. Larroque); bgiraud@cerfacs.fr (B. Giraud); dauptain@cerfacs.fr (A. Dauptain)

ORCID(s): 0000-0003-2355-2740 (A. Larroque); - (B. Giraud); 0009-0009-1933-555X (A. Dauptain)

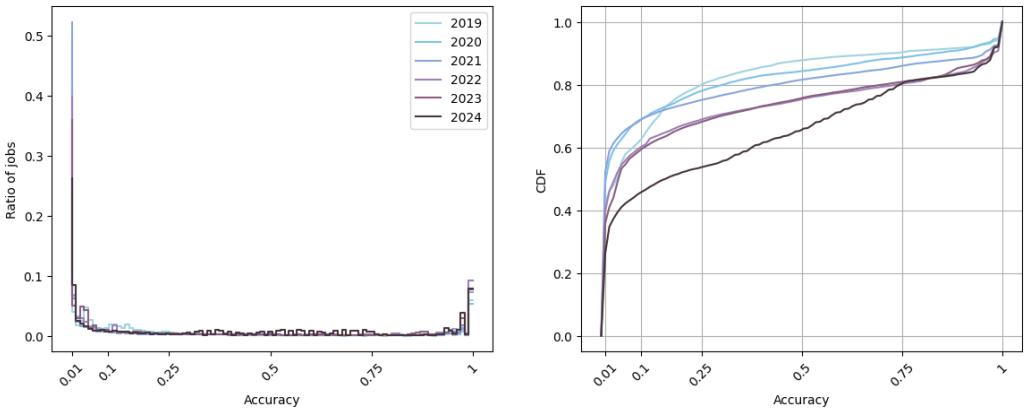


Figure 1: Histograms (left) and Cumulative Distribution Function (right) of the users' accuracy for various years on a supercomputer.

- provide data to model submission habits of various communities of users in order to predict the supercomputer usage and simulate how changes in submission habits can affect the production.
- raise awareness among the users about sub-optimal submission patterns through graphical representations in order to hopefully study in a real case the impact of better submission practices.

2. Seed

Seed is a Python software developed at CERFACS. Seed can analyze basic jobs characteristics submitted by the users such as the distribution of the duration of the jobs, the number of CPUs allocated, the number of CPU hours, the inter arrival time of the jobs, the waiting times, the slowdown and the users' accuracy. The accuracy (the actual time of the job divided by the requested time) is of particular interest, since as mentioned earlier, imprecise accuracy is associated with larger waiting times and slowdown.

Figure 1 displays the histograms and Cumulative Distribution Function (CDF) for the users' accuracy along the years on on the production queues of an in-house supercomputer that is still running. You can observe, especially with the CDF, a progression in terms of accuracy along the years. Indeed, in 2019 approximately 80 % of jobs had an accuracy inferior to 0.25 against roughly 50 % for 2024. Still, in 2024, more than 60% of the jobs have an accuracy inferior to 0.5, and approximately 10 % of the jobs stay running until being evicted by the scheduler. This is clearly sub-optimal for the resources management system.

We can also look at the evolution of users' accuracy month by month as in Fig. 2. We classified the jobs as "nostayers" (the jobs instantly crash/finish despite a large requested time), "understayers" (less than 2/3 of the requested time was used for the jobs), the "normal" ones (between 2/3 of the requested time and strictly inferior to the request time) and the "overstayers" (the ones that stay until the end of the requested time). You can observe the global progression of the accuracy along the years: at the end of January 2019, there were only 1% of "normal" jobs and 91 % of "nostayers" whereas at the end of June 2024, there were around 18 % of "normal" jobs and 13 % of "nostayers". The number of CPU hours attributed to "normal" jobs also increases over time contrary to the number of CPU hours of the "overstayers" that declines.

In order to encourage the users to be more accurate on the requested time, we developed in Seed, an analysis tool to highlight the people with the best ratio of jobs with an accurate requested time (category "normal") on the production jobs. This top 5 is displayed in Fig. 3 where the score is computed as the number of jobs with an accurate requested time divided by the total number of jobs run by the users on production queues. Normally, the users' real names are displayed in this graph but they were anonymized for this paper. With this graph, we aim to both congratulate users who tailored the requested time to their jobs and to follow their progression. Note that we started to show this graph to users at the end of January 2024. If we look back at Fig. 2, we can observe a decrease in the proportion of "nostayers" and an increase in "normal" jobs submitted from this period.

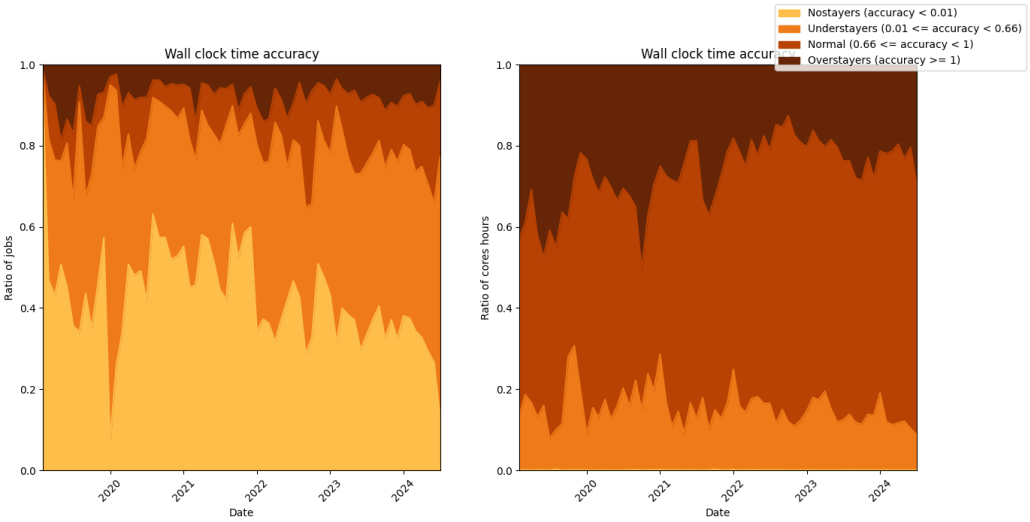


Figure 2: Distribution of various categories of accuracy as a function of the time in terms of number of jobs (left) and CPU hours (right) on production queues. The "nostayers" represent jobs that instantly crash/finish, the "understayers" are the jobs that undervalue the wall clock time, the "normal" stand for the jobs with a good estimation of the requested time, and the "overstayers" are the jobs staying until the end of the wall clock time.

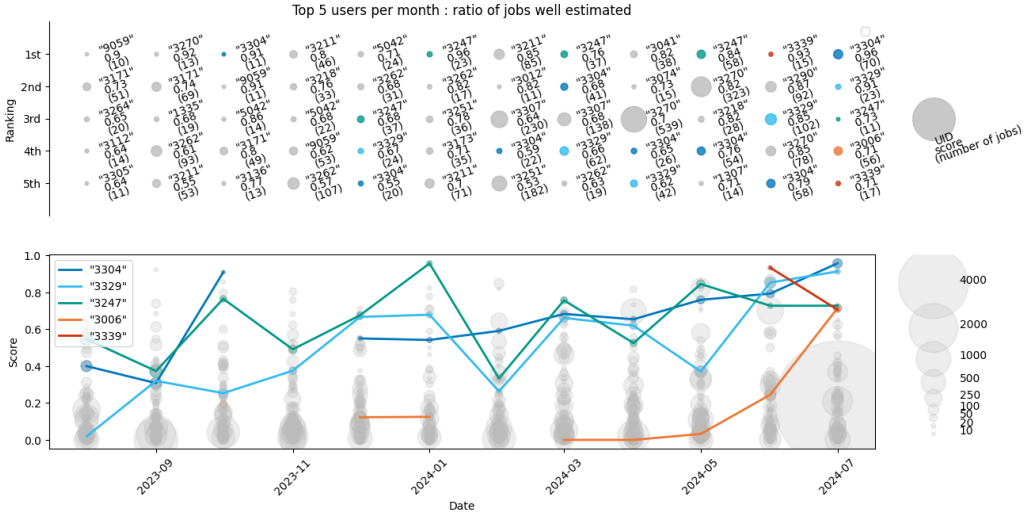


Figure 3: Top 5 of the users with their ratio of jobs with an accurate requested time (top) and evolution of the best 5 users of the last month (bottom). For each user, the radius of the circle is proportional to the number of jobs submitted by the user that run on the supercomputer.

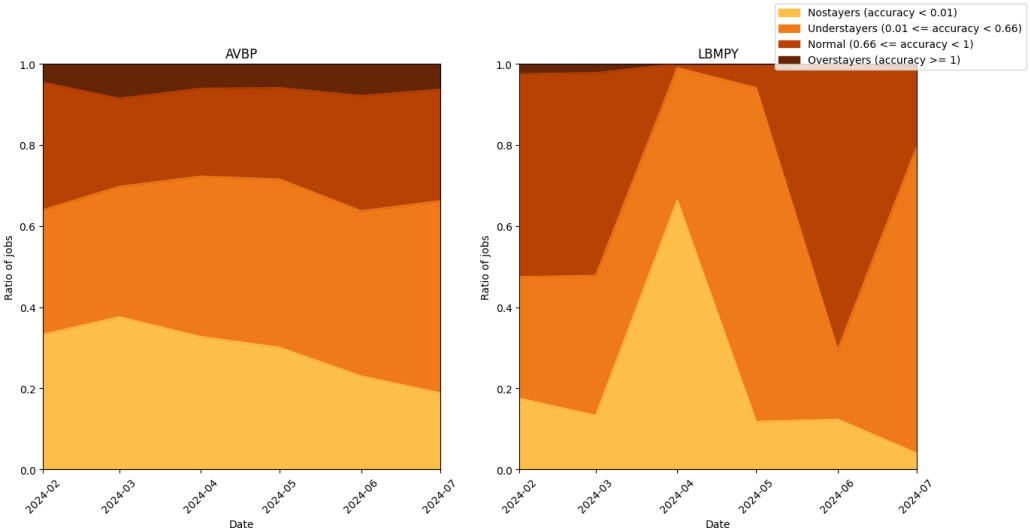


Figure 4: Distribution of various categories of accuracy as a function of the time in terms of number of jobs for a software used mainly for high-fidelity CFD (left) and a software used for AI applications in fluid mechanics (right).

However, it may be too early to attribute this improvement to our awareness campaign.

Finally, in order to gather more information about the users' jobs and to assess which community was the most subject to poor estimated wall clock time, we encouraged the users to tag their jobs with the software they were using. We were then able to evaluate the number of CPU hours dedicated for each software and perform an analysis similar to Fig. 1 and Fig. 2 to check the differences in jobs submissions between various communities of users. Figure 4 displays the evolution of the categories of accuracy for two communities of users: the ones using AVBP, a software used for high-fidelity numerical simulations in turbomachinery and, the ones using LMBPY, a Lattice Boltzmann solver that certain CERFACS users utilize for ML/AI applications. As you can observe on this figure, the AVBP users tend to submit more jobs that will stay until being evicted by the scheduler, whereas the LBMPY users generally do not need all the wall clock time set and we can observe more "normal" submissions.

3. Conclusions and perspectives

In this paper, we introduced a first step in the modeling of users' behavior to forecast the usage of future supercomputers architectures. We especially highlighted the importance of modeling the requested time and introduced Seed to perform analysis of the users' submission patterns and/or improve the behavior of the users on a supercomputer.

In a long run, we would like to develop advanced workload models based on the users' submission patterns. Zakay and Feitelson [2] developed a workload model to preserve the users' experience in terms of time of submission. This kind of models could be improved with an users' accuracy model to assess the impact of sub-optimal and optimal accuracy on various virtual machines.

Acknowledgment

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Italy, Slovenia, Spain, Sweden, and France under grant agreement No 101092621.

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU)

and Germany, Italy, Slovenia, Spain, Sweden, and France. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] S.-H. Chiang, A. Arpaci-Dusseau, M. K. Vernon, The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance, in: Job Scheduling Strategies for Parallel Processing: 8th International Workshop (JSSPP), Lecture Notes in Computer Science (LNCS) Vol. 2537, Springer, Berlin, Heidelberg, Edinburgh, Scotland, UK, 2002, pp. 103–127. doi:10.1007/3-540-36180-4_7.

[2] N. Zakay, D. G. Feitelson, Preserving user behavior characteristics in trace-based simulation of parallel job scheduling, in: Proceedings of the 8th ACM International Systems and Storage Conference, ACM, New York, NY, USA, 2015, pp. 1–1. doi:10.1145/2757667.2778191.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Correctness and Performance Analysis of an Open-Source CFD Application

Fabian Orland[a,*], Joachim Jenke[a] and Radita Liem[a]

[a]RWTH Aachen University, Chair for High-Performance Computing, Seffenter Weg 23, 52074, Aachen, Germany

ARTICLE INFO[†]

Keywords:
Computational Fluid Dynamics;
Finite Volume Method;
OpenMP;
Data Race;
Performance Analysis;
Input/Output

ABSTRACT

This contribution highlights correctness and performance analysis results for an open-source CFD application. During the performance analysis of the code, we observed unexpected non-deterministic behavior in the application. Data race analysis with ThreadSanitizer, available in GNU and LLVM compilers, in combination with our extension for OpenMP-aware data race analysis, identified the root cause of the non-deterministic behavior. We will look into the analysis setup and the reports from the tool. Furthermore, we will discuss insights about the performance and I/O behavior of the code collected with the Scalasca tool-suite & Darshan and highlight performance optimizations.

1. Introduction

The open-source CFD application under investigation in this work is CalculiX[1]. CalculiX was mainly developed to solve problems in structural mechanics using the finite-element method. However, CalculiX also offers the solution to computational fluid dynamics problems using the finite-volume method described by Moukalled et al. [1]. The use case investigated in this work simulates the laminar airflow through a 90° bent pipe with a square cross-section. The code is written in C and Fortran and parallelized using pthreads.

1.1. Pthreads parallelization

To achieve worksharing via multithreading using pthreads, the pattern shown in Fig. 1a is repeatedly found in the CalcluliX code. The code snippets show how the residual of solving the momentum equations is parallelized. The application's main thread creates `num_cpus` new worker threads using `pthread_create` (l.48). The function pointer passed to the creation call, i.e. `calcresvfluid1mt`, is a high-level wrapper function that takes care of the work distribution for each spawned thread (Fig. 1c, l.146) and calls into a Fortran subroutine that implements the actual work (l.148). In this case, it is `calcresvfluid1`, which calculates the residual using a sparse matrix-vector product (Fig. 1d, l.30). After all threads have been spawned, the main thread waits until all spawned worker threads have finished their work by calling `pthread_join` (Fig. 1a, l.51).

This code pattern leads to an excessive total number of threads created during execution because each time the application performs multithreading, *num_cpus* threads are created, but they are not reused. For example, executing the simulation for 5 timesteps with $num\_cpus = 8$ threads creates 3809 threads in total. This pattern prohibits tracing longer simulation runs, because performance analysis tools like Extrae, Score-P and Vampir cannot handle those huge number of threads. For example, Extrae was not able to handle more than 7800 threads.

1.2. OpenMP refactoring

The problematic pattern shown in Fig. 1a can be refactored into equivalent OpenMP code, as shown in Fig. 1b. This enables the application to benefit from the thread pool internally managed by the OpenMP runtime and avoids excessively creating new threads without any reuse. It also highlights the importance of application developers relying on best practices provided by open standards like OpenMP.

2. Correctness analysis

At the beginning of our analysis, the application showed unexpected non-deterministic behavior when executed by more than a single thread. In repeated executions using the same input file and the same number of threads, the internal, parallel GMRES solver required a different number of iterations until convergence. A data race analysis using ThreadSanitizer[2] (TSan) revealed the root cause of this non-determinism. As shown in Fig. 2, TSan detected a data race caused by two concurrent writes by thread T1 and thread T2 to the same memory location at address `0x7b1000000000` in

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02484 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
orland@itc.rwth-aachen.de (F. Orland); jenke@itc.rwth-aachen.de (J. Jenke); liem@itc.rwth-aachen.de (R. Liem)
ORCID(s): 0000-0002-8681-2661 (F. Orland); 0000-0003-0640-8966 (J. Jenke); 0000-0002-2506-1841 (R. Liem)

[1]https://www.calculix.de/

[2]https://github.com/google/sanitizers

```
43  void calcresvfluidmain(ITG *num_cpus, /* ... */){
44      NNEW(ithread,ITG,*num_cpus); pthread_t tid[*num_cpus];
45      NNEW(res1,double,*num_cpus);
46      for(i=0; i<*num_cpus; i++)  {
47        ithread[i]=i;
48        pthread_create(&tid[i], NULL,
49          (void*)calcresvfluid1mt, (void*)&ithread[i]);
50      }
51      for(i=0; i<*num_cpus; i++)  pthread_join(tid[i], NULL);
52      /* ... */
53  }
```

(a) `calcresvfluidmain.c` (pthreads)

```
43  void calcresvfluidmain(ITG *num_cpus, /* ... */){
44      NNEW(ithread,ITG,*num_cpus); pthread_t tid[*num_cpus];
45      NNEW(res1,double,*num_cpus);
46      #pragma omp parallel num_threads(*num_cpus)
47      {
48        int omp_get_thread_num(void);
49        ITG thread_num = omp_get_thread_num();
50        calcresvfluid1mt(&thread_num);
51      }
52      /* ... */
53  }
```

(b) `calcresvfluidmain.c` (OpenMP)

```
143 /* subroutine for multithreading of calcresvfluid1 */
144 void *calcresvfluid1mt(ITG *i){
145     /* number of equations for this thread */
146     ITG n=nestart1[*i+1]-nestart1[*i];
147
148     FORTRAN(calcresvfluid1,(&n,a1,&b1[nestart1[*i]],
149         &au1[nestart1[*i]],ia1,&ja1[nestart1[*i]],
150         &x1[nestart1[*i]],res1));
151     return NULL;
152 }
```

(c) `calcresvfluidmain.c`

```
25  subroutine calcresvfluid1(a,b,x,res)
26  implicit none
27  real*8 a(*),b(*),x(*),res
28
29  res=0.d0
30  ! calculation of res = (A*x)-b
31
32  return
33  end
```

(d) `calcresvfluid1.f`

Figure 1: Multithreading implementation in CalculiX by example of residual calculation. Subcaptions and line numbers refer to CalculiX source code files.

line 29 of `calcresvfluid1.f` (Fig. 1d). This memory location corresponds to the variable `res`. TSan also shows that the main thread allocated this memory location in line 45 of `calcresvfluidmain.c` (Fig. 1a). The `NNEW` macro expands to a `u_calloc` call that allocates `res1` as an array of double-precision values of size `*num_cpus`. In our case `num_cpus` = 8, which matches the size of 64 bytes reported by TSan for the memory block. The developer's intention behind this allocation is that thread i should store its calculated residual at index i of the array `res1`. However, in lines 148-150 of the file `calcresvfluidmain.c`, a pointer to the first element of array `res1` is passed to the subroutine `calcresfluid1` instead of a pointer to the i-th element. We modified the call to the subroutine `calcresvfluid1` and passed `&res1[i]` instead of `res1`, and no data race was detected anymore. After this change, the internal GMRES solver's convergence also behaves deterministically.

```
==================
    WARNING: ThreadSanitizer: data race (pid=115530)
      Write of size 8 at 0x7b1000000000 by thread T2:
        #0 calcresvfluid1_ CalculiX/calcresvfluid1.f:29
        #1 calcresvfluid1mt CalculiX/calcresvfluidmain.c:148

      Previous write of size 8 at 0x7b1000000000 by thread T1:
        #0 calcresvfluid1_ CalculiX/calcresvfluid1.f:29
        #1 calcresvfluid1mt CalculiX/calcresvfluidmain.c:148

    Location is heap block of size 64 at 0x7b1000000000
        allocated by main thread:
        #0 calloc libsanitizer/tsan/tsan_interceptors.cc:964
        #1 u_calloc CalculiX/u_calloc.c:41
        #2 calcresvfluidmain CalculiX/calcresvfluidmain.c:45
        #3 compfluid CalculiX/compfluid.c:856
        #4 nonlingeo CalculiX/nonlingeo.c:1307
        #5 main CalculiX/CalculiX.c:1150

    SUMMARY: ThreadSanitizer: data race CalculiX/calcresvfluid1.f:29
    ==================
```

Figure 2: TSan's analysis output when executing the application with eight threads.

We were able to reproduce and detect the same data race with the refactored OpenMP version of the code (Fig. 1b) using our extensions for OpenMP-aware data race analysis implemented on top of TSan in the Archer tool [2] and distributed as part of LLVM. To enable Archer's analysis capabilities, the application code must be instrumented by adding the flag `-fsanitize=threads` to the compiler. LLVM's OpenMP runtime automatically loads Archer when appropriate. Fortran code can be compiled with `gfortran` if `clang` is finally used to link the code with its OpenMP runtime.

3. Performance analysis

We conducted a performance analysis with the data-race free application and used the Scalasca tool suite, including Score-P, Cube, and Vampir, to collect and investigate

Figure 3: Trace comparison of reference version with nested tasks.

profiles and traces. The profile revealed a function called `dgmres1mt` as the hotspot, accounting for roughly 70% of the application's runtime, depending on the number of threads. This function is the main driver of the internal, parallel GMRES solver, which is used to solve large, sparse, non-linear systems arising from the discretization of the momentum equations. GMRES is parallelized by splitting the whole system matrix into independent subsystems so that each thread can solve one of these subsystems.

To further assess the parallel efficiency of the code, we calculated fundamental model factors proposed in [3] using the Cube Advisor Plugin, which indicated a loss of parallel efficiency caused by a load imbalance inside the GMRES solver. Further investigation of the trace with Vampir revealed a repeating pattern in which the same thread consistently requires more iterations to solve its subsystem compared to the others, as illustrated by the top trace in Fig. 3.

To improve the GMRES solver's load balance, we implemented a nested taskloop inside suitable parallelization subroutines, which are color-coded in the trace. This includes the matrix-vector product (orange), application of the preconditioner (yellow), a `daxpy` operation from LAPACK (blue), and computation of the residual (green) to check the convergence of GMRES. The orthogonalization (pink) of computed Krylov subspace vectors could not be parallelized. The nested taskloop is triggered to split the work between the straggling and idling threads if a threshold of more than 75% idling threads is detected using a global atomic counter variable. This optimization improved the load balance by 15% on average. It resulted in a runtime speedup of around 1.08x, which can also be visually observed using Vampir's trace comparison shown in Fig. 3.

Finally, we also investigated the I/O performance of the application using Darshan [4]. Darshan's analysis revealed that more than 20% of the runtime is spent in write operations. Moreover, the data is written in tiny chunks of 3-6 double-precision values. We enabled the usage of buffered I/O operations by adding the `buffered="yes"` keyword to all `open` calls in the code. This resulted in a speedup of 1.43x for the whole application.

References

[1] F. Moukalled, L. Mangani, M. Darwish, The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab, 1st Edition, Springer Publishing Company, Incorporated, 2015. doi:10.1007/978-3-319-16874-6_5.

[2] J. Protze, J. Hahnfeld, D. H. Ahn, M. Schulz, M. S. Müller, Openmp tools interface: Synchronization information for data race detection, in: IWOMP, Vol. 10468 of Lecture Notes in Computer Science, Springer, 2017, pp. 249–265. doi:10.1007/978-3-319-65578-9_17.

[3] C. Rosas, J. Giménez, J. Labarta, Scalability prediction for fundamental performance factors, Supercomput. Front. Innov. 1 (2) (2014). doi:10.14529/jsfi140201.

[4] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, N. J. Wright, Modular hpc i/o characterization with darshan, in: 2016 5th Workshop on Extreme-Scale Programming Tools (ESPT), 2016, pp. 9–17. doi:10.1109/ESPT.2016.006.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Streamlining Performance Analysis Workflows Using Compiler-Assisted Instrumentation Selection

Sebastian Kreutzer[a,*], Peter Arzt[a] and Christian Bischof[a]

[a]Scientific Computing, TU Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany

ARTICLE INFO[†]

Keywords:
Instrumentation;
Profiling;
Tracing;
Performance Analysis;
Clang/LLVM

ABSTRACT

Efficient performance analysis is crucial for optimizing scientific applications. The traditional instrumentation workflow may require multiple manual refinements of the instrumentation configuration to minimize measurement perturbations, which can be time-consuming. This paper presents two approaches to streamline these workflows, guided by a compiler-generated static whole-program call graph of the target application. The *Compiler-assisted Performance Instrumentation* tool (CaPI) enables the user to build modular selection pipelines, which can be tailored to the application and measurement objective. *Performance Instrumentation Refinement Automation* (PIRA) combines static call graph analysis with dynamic profiling for iterative refinement. Ongoing work aims to integrate these approaches into a unified production tool, emphasizing workflow efficiency.

1. Introduction

Code instrumentation, used by performance tools such as TAU [1] and Score-P [2], is the primary method for collecting fine-grained, non-statistical performance data. By explicitly inserting measurement points directly, this technique ensures that every function invocation is accurately recorded. However, overhead management is key to keep the introduced measurement perturbation to an acceptable level. This is achieved by restricting the instrumentation to a subset of regions, referred to as the *instrumentation configuration* (IC). ICs are commonly created by starting with a full instrumentation, running the application, and subsequently adjusting the instrumentation based on the collected data. Functions deemed irrelevant or too costly to instrument, e.g. because they are short-running or frequently called, are then removed from the instrumentation, either manually or using tools such as *scorep-score*[1]. We refer to this process as the *build-run-analyze* cycle. For complex applications, multiple iterations of this process may be required, taking up a significant amount of time and effort. In this work, we are highlighting our work on alternative approaches relying on static program information in the form of a compiler-generated whole-program call graph of the target application. This information can be used to automate and simplify the build-run-analyze cycle. In prior work, we have focused on exploring tool support the following aspects:

- Tailoring ICs to the application and measurement objective based on user direction
- Automated profile-guided refinement of the IC over multiple iterations

In the following, we give an overview of our work on call graph based instrumentation selection techniques and illustrate their application to highly-parallel scientific codes. Additionally, we discuss current developments.

2. Call-graph-based instrumentation selection

To generate ICs programmatically, we rely on *call graphs* as a representation of the program's structure. Functions are encoded as the nodes of the graph, while potential calls between two functions are represented as edges.

We use *MetaCG* [3] to collect, process and analyze the call graph of the target application. This tool set provides utilities to generate the call graph from the target source code using a Clang-based solution, along with a C++-library for graph management and analysis that we use as the foundation for our performance analysis tools. As part of its design, MetaCG allows the decoration of the call graph with arbitrary, tool-specific meta data to enable advanced analyses.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02485 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
sebastian.kreutzer@tu-darmstadt.de (S. Kreutzer); peter.arzt@tu-darmstadt.de (P. Arzt); christian.bischof@tu-darmstadt.de (C. Bischof)
ORCID(s): 0000-0002-1641-4342 (S. Kreutzer); 0000-0001-6937-1158 (P. Arzt); 0000-0003-2711-3032 (C. Bischof)

[1]Score-P Project, scorep-score manual available at https://perftools.pages.jsc.fz-juelich.de/cicd/scorep/tags/scorep-8.1/html/score.html, version 8.1 (2023).

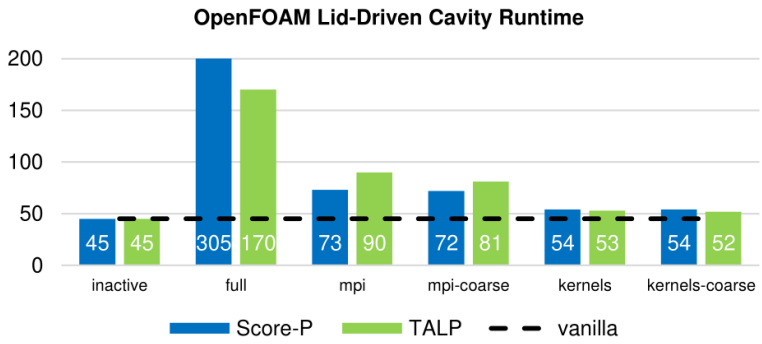


Figure 1: Performance comparison of an OpenFOAM benchmark with various ICs, using XRay instrumentation. The shown CaPI ICs focus on detecting call paths containing compute kernels and MPI calls respectively, including "coarse" versions that further reduce the instrumentation. The `inactive` variant corresponds to running the application with inactive instrumentation, `full` refers to full instrumentation. Measurements are from [8], conducted on a single compute node consisting of two Intel Xeon Platinum 9242 CPUs (96 cores).

2.1. User-directed static selection

The *Compiler-assisted Performance Instrumentation* tool (CaPI) [4] was created within the *exaFOAM*[2] project to aid in the profiling of OpenFOAM and other large-scale scientific applications. It enables the creation of ICs tailored to the application and measurement objective. To this end, CaPI provides a custom domain-specific language (DSL) that enables the user to construct a pipeline of parameterized, pre-defined selector modules. These modules perform selection based on structural or function-local code metrics. Additionally, CaPI provides a runtime library that enables integration with several profiling and tracing tools, currently offering support for Score-P [2], Extrae [5] and TALP [6]. It uses a recently extended version of XRay, the dynamic instrumentation feature of the Clang/LLVM compiler [7], to enable adapting the IC without the need for recompilation or dynamic filtering [8]. In contrast to the instrumentation approach used in Score-P and TAU that inserts profiling calls statically into the binary, XRay uses placeholder instructions (`NO-OP` sleds) that can be selectively patched at runtime. Overhead compared to static instrumentation is mostly limited to a few seconds for initialization and patching at program start, with a slight additional cost incurred by the CaPI runtime for passing call events to the measurement tool. We have used XRay to successfully instrument large code bases with more than 400,000 functions. Figure 1 shows performance results for an OpenFOAM benchmark, profiled with different CaPI-generated ICs, using both Score-P and TALP as measurement back-ends. For this application, CaPI was used to identify relevant call paths containing MPI communication (`mpi` and `mpi-coarse` variants) as well as call paths leading up to computational kernels (`kernels` and `kernels-coarse`). The application of CaPI led to a significant overhead reduction compared to the baseline full instrumentation, while retaining profile information about performance-critical program parts. Notably, running the XRay-instrumented program without any active instrumentation did not produce any overhead compared to the baseline `vanilla` variant. A single build can thus be used for both profiling and production runs.

In our most recent work, we have extended CaPI with selection mechanisms to augment MPI traces using measurements collected via instrumentation of functions within problematic trace regions [9].

[2] `https://exafoam.eu/`

2.2. Iterative instrumentation refinement using profile information

Instead of relying on static call graph information alone, our second approach revolves around improving the generated IC by combining the call graph with dynamic profile information. Inspired by the build-run-analyze workflow that is usually performed manually by performance analysts, it automates the iterative cycle of building an instrumented binary, running performance measurements and then analyzing the results to improve the IC for the next iteration. This approach is implemented in the *PIRA* tool set [10], short for *Performance Instrumentation Refinement Automation*. Starting from an initial IC that is generated using static information alone, PIRA repeatedly refines the IC by incorporating dynamic profile information from the previous

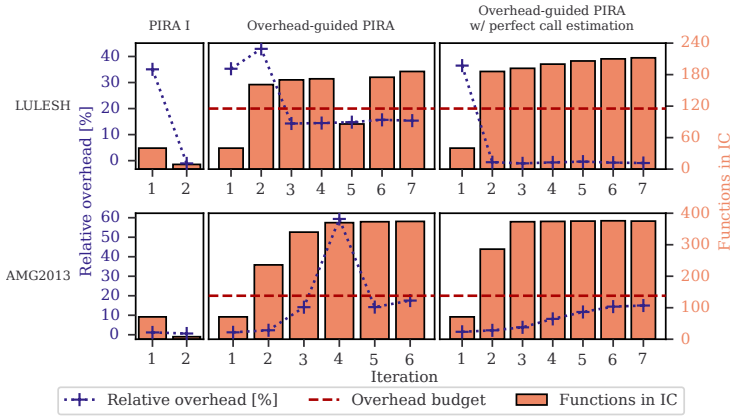


Figure 2: Preliminary overhead measurements for overhead-guided PIRA for the two mini-apps LULESH and AMG2013. The plots show the relative overhead at each refinement iteration, as well as the number of functions in the corresponding ICs. The left plot shows PIRA's initial hotspot detection mode, the middle the new overhead-guided heuristic (estimating call counts of functions not yet instrumented) and the right side an equivalent version that has access to perfect call counts.

iteration. Currently, PIRA offers four principal modes of operation with distinct focuses:

- The initial PIRA version introduced a *hotspot detection* mode that refines the IC towards the parts of the target application that consume the majority of the runtime. The search for hotspots is a common task when analyzing an application's performance.
- PIRA also integrates *empirical performance modeling* [11] for *kernel detection*, i.e. to detect functions with interesting scaling behaviors. To that end, in each iteration, the target application is executed with a user-defined list of input sizes. Then, the empirical performance modeling tool *Extra-P* [12] is invoked for every function included in the profile and produces a mathematical model to predict function runtimes based on input sizes. The models are then evaluated to determine which functions to include in the next IC.
- PIRA LIDe [13] adds load-imbalance detection to PIRA by quantifying the scattering of function runtimes across different MPI ranks and refines the IC based on the imbalance severity.
- Recent developments in PIRA introduced the concept of *overhead-guided instrumentation refinement*. This feature allows the user to specify an *overhead budget*. PIRA then performs hotspot detection, but strives to keep the measurement overhead below the provided budget by mapping the decision which functions to instrument to a binary Knapsack problem. Preliminary results for this mode are shown in Fig. 2.

3. Ongoing and future work

Work on PIRA and CaPI is ongoing, with the central goal of maturing the presented approaches from proof-of-concept implementations to a unified production tool. Current work is focused on:

- Streamlining the generation and validation checking of the global call graph.
- Improving the MetaCG tool set to collect and process call graphs of Fortran applications, thus enabling PIRA and CaPI to operate on such codes.
- Extending and upstreaming new features for LLVM XRay.
- Integrating PIRA features into CaPI, building a unified tool for static and dynamic instrumentation selection.

References

[1] S. S. Shende, A. D. Malony, The Tau Parallel Performance System, The International Journal of High Performance Computing Applications 20 (2) (2006) 287–311. doi:10.1177/1094342006064482.

[2] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, F. Wolf, Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir (2012) 79–91. doi:10.1007/978-3-642-31476-6_7.

[3] J.-P. Lehr, A. Hück, Y. Fischler, C. Bischof, MetaCG: annotated call-graphs to facilitate whole-program analysis, in: Proceedings of the 11th ACM SIGPLAN International Workshop on Tools for Automatic Program Analysis, ACM, New York, NY, USA, 2020, pp. 3–9. doi:10.1145/3427764.3428320.

[4] S. Kreutzer, C. Iwainsky, J.-P. Lehr, C. Bischof, Compiler-Assisted Instrumentation Selection for Large-Scale C++ Codes, in: H. Anzt, A. Bienz, P. Luszczek, M. Baboulin (Eds.), High Performance Computing. ISC High Performance 2022 International Workshops, Springer International Publishing, 2022, pp. 5–19. doi:10.1007/978-3-031-23220-6_1.

[5] H. Servat, G. Llort, K. Huck, J. Giménez, J. Labarta, Framework for a productive performance optimization, Parallel Computing 39 (8) (2013) 336–353. doi:10.1016/j.parco.2013.05.004.

[6] V. Lopez, G. Ramirez Miranda, M. Garcia-Gasulla, TALP: A Lightweight Tool to Unveil Parallel Efficiency of Large-scale Executions, in: Proceedings of the 2021 on Performance EngineeRing, Modelling, Analysis, and VisualizatiOn STrategy, ACM, New York, NY, USA, 2021, pp. 3–10. doi:10.1145/3452412.3462753.

[7] D. M. Berris, A. Veitch, N. Heintze, E. Anderson, N. Wang, Xray: A function call tracing system, Tech. rep., a white paper on XRay, a function call tracing system developed at Google. (2016).
URL https://api.semanticscholar.org/CorpusID:64171003

[8] S. Kreutzer, C. Iwainsky, M. Garcia-Gasulla, V. Lopez, C. Bischof, Runtime-Adaptable Selective Performance Instrumentation, in: 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2023, pp. 423–432. doi:10.1109/IPDPSW59300.2023.00073.

[9] S. Kreutzer, J. P. Serra, C. Iwainsky, M. G. Gasulla, C. Bischof, Augmentation of MPI Traces Using Selective Instrumentation, 2024, pp. 31–44. doi:10.1007/978-3-031-73716-9_3.

[10] J.-P. Lehr, A. Hück, C. Bischof, PIRA: performance instrumentation refinement automation, in: Proceedings of the 5th ACM SIGPLAN International Workshop on Artificial Intelligence and Empirical Methods for Software Engineering and Parallel Computing Systems, ACM, New York, NY, USA, 2018, pp. 1–10. doi:10.1145/3281070.3281071.

[11] J.-P. Lehr, A. Calotoiu, C. Bischof, F. Wolf, Automatic instrumentation refinement for empirical performance modeling, in: 2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools), IEEE, 2019, pp. 40–47. doi:10.1109/protools49597.2019.00011.

[12] A. Calotoiu, T. Hoefler, M. Poke, F. Wolf, Using automated performance modeling to find scalability bugs in complex codes, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ACM, New York, NY, USA, 2013, pp. 1–12. doi:10.1145/2503210.2503277.

[13] P. Arzt, Y. Fischler, J.-P. Lehr, C. Bischof, Automatic low-overhead load-imbalance detection in MPI applications, in: Euro-Par 2021: Parallel Processing: 27th International Conference on Parallel and Distributed Computing, Lisbon, Portugal, September 1–3, 2021, Proceedings, Springer, 2021, pp. 19–34. doi:10.1007/978-3-030-85665-6_2.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Working Towards FAIRness in Performance Data

Maximilian Sander[a,*], William R. Williams[a] and Bert Wesarg[a,b]

[a]TUD Dresden University of Technology, Center for Interdisciplinary Digital Sciences (CIDS), Information Services and High Performance Computing (ZIH), Zellescher Weg 12-14, 01069 Dresden, Germany
[b]GWT-TUD GmbH, Freiberger Str. 33, 01067 Dresden, Germany

ARTICLE INFO[†]

Keywords:
Supercomputing;
Performance Tools;
FAIR Principles;
Data Management;
Ontology;
Continuous Performance Tracking

ABSTRACT

Making sense of the performance data generated by computational fluid dynamics (CFD) experiments is challenging, both due to the high volume of data and the high variability of possible inputs. We present preliminary work on a system to store sufficient metadata alongside the performance data generated by the Score-P measurement tool to make these results more Findable, Accessible, Interoperable, and Reusable (FAIR). This system can be deployed in both continuous integration and production runs of CFD codes to allow developers better insight into potential causes of performance variations.

1. Introduction

Computational Fluid Dynamics (CFD) solvers play a crucial role in the development and design processes across various disciplines, such as turbomachinery, marine propulsion and aerodynamics. The calculations involved are time and resource-intensive, making a fast and well scaling solver infrastructure paramount. Metrics such as runtime and parallel efficiency have been defined to measure and improve on these characteristics. Performance measurement tools such as Score-P [1] together with postprocessing tools like Vampir [2], Cube [3], and Extra-P [4] help programmers to understand runtime behavior and detect scalability issues.

These issues may not always be reproducible or obvious and could even be side effects introduced by new, seemingly unrelated code changes. Therefore, continuously tracking these metrics is essential to address them early and improve on them.

To achieve this, these tools should be run both within and outside of a continuous integration (CI) workflow to ensure comprehensive coverage of configurations at larger scale. Collecting data of every CI execution poses challenges in data management, accessibility and value extraction. This challenge is addressed by the concept of FAIR data, which promotes goals for Findability, Accessibility, Interoperability and Reusability [5].

An essential step towards implementing FAIR principles is defining a data schema and an associated ontology. A data schema outlines which data elements that exist, and the ontology describes the semantic relationships among those elements. RDF (Resource Description Framework)[1] is a framework to describe ontologies and associated data via $subject, predicate, object$ triplets. These triplets connect data elements according to the ontology, forming a graph stored in a triplestore or a graph database which is queried via SPARQL[2].

This work proposes a lightweight but versatile option for continuous performance tracking of Score-P instrumented applications based on the RDFlib [6], PostgreSQL or SQLite databases, and an object store. We extend the existing work by enhancing Score-P with metadata collection capabilities, suggesting a schema definition and simulation case description, and providing proposals for CI implementation and in-field production runs.

2. Related work

Performance monitoring is of growing interest, on both a per-application basis and a site-wide basis. Dai et al. [7] and Evans et al. [8] are representative examples of site-wide monitoring approaches. While the metrics collected by these tools are a useful point of comparison, both aim for site-wide monitoring and are therefore not suitable for our intended purpose, as the recorded metadata is not simulation specific and the backend requires significant infrastructure.

On a per-application basis, Boehme et al. [9] use Caliper for continuous performance data collection, Adiak for user-defined meta data collection, and SPOT for visualization,

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02486 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
maximilian.sander1@tu-dresden.de (M. Sander);
william.williams@tu-dresden.de (W.R. Williams);
bert.wesarg@tu-dresden.de (B. Wesarg)
ORCID(s): 0009-0007-3223-4046 (M. Sander); 0000-0001-5806-3176 (W.R. Williams); 0000-0003-2647-0628 (B. Wesarg)

[1]World Wide Web Consortium (W3C), RDF 1.1 Primer https://www.w3.org/TR/rdf11-primer/

[2]World Wide Web Consortium (W3C), SPARQL 1.1 Overview https://www.w3.org/TR/sparql11-overview/

Environment_Variables	Values	MPI_Ranks	AvgRuntime
OMP_PROC_BIND, TC_NAME, OMP_PLACES, OMP_NUM_THREADS	spread, intel2022a, cores, 13	8	5.274644
TC_NAME, OMP_NUM_THREADS	intel2022a, 13	8	5.336053
TC_NAME, OMP_NUM_THREADS	foss2022a, 13	8	6.037955
OMP_NUM_THREADS, TC_NAME, OMP_PLACES, OMP_PROC_BIND	13, foss2022a, cores, spread	8	6.050352
OMP_NUM_THREADS, OMP_PLACES, OMP_PROC_BIND, TC_NAME	26, cores, spread, intel2022a	4	6.196884
TC_NAME, OMP_PLACES, OMP_PROC_BIND, OMP_NUM_THREADS	foss2022a, cores, spread, 26	4	6.709384
OMP_NUM_THREADS, TC_NAME	26, foss2022a	4	6.853112
TC_NAME, OMP_NUM_THREADS, OMP_PROC_BIND, OMP_PLACES	foss2022a, 52, spread, cores	2	7.979124
TC_NAME, OMP_NUM_THREADS, OMP_PROC_BIND, OMP_PLACES	intel2022a, 52, spread, cores	2	8.090973
TC_NAME, OMP_NUM_THREADS	foss2022a, 52	2	8.556215
TC_NAME, OMP_NUM_THREADS	intel2022a, 26	4	8.807644
OMP_NUM_THREADS, TC_NAME	52, intel2022a	2	11.576881
OMP_PROC_BIND, OMP_NUM_THREADS, TC_NAME, OMP_PLACES	spread, 104, foss2022a, cores	1	13.526483
OMP_NUM_THREADS, TC_NAME	104, foss2022a	1	15.894367
OMP_PROC_BIND, OMP_PLACES, TC_NAME, OMP_NUM_THREADS	spread, cores, intel2022a, 104	1	16.01187
TC_NAME, OMP_NUM_THREADS	intel2022a, 104	1	23.095881

Table 1
Example result of a database query.

storing data in an SQL database or on the file system. Liao et. al. [10] propose a ontology for projects running on HPC with a dedicated view for artificial intelligence datasets and models. Their HPC ontology is defined in RDF making use of well defined ontologies such as OWL and QUDT. Their backend is based on Blazegraph, queried via SPARQL. Our setup builds on their HPC ontology, additionally providing Score-P compilation and performance data, and offering an offline store.

3. Implementation

Score-P is an application-level measurement system that already generates some metadata about measurements it collects. We extend the metadata it collects during instrumentation and runtime to include selected environment variables, compiler flags, time stamps, and the used source or object files. Hashes of the object and binary files are also part of the recorded metadata. Users are encouraged to provide additional source code metadata, such as a commit hash, via environment variable.

We derive different sources of performance-related metadata using Cube, in particular the POP metrics[3], from the Score-P generated profiles and traces.

The proposed schema is based on Liao et. al.'s HPC ontology [10]. We extend it with the compile time and runtime metadata of Score-P directly and those that can be extracted from the traces and profiles via Scalasca and Cube like the POP metrics. Application-specific metadata should be provided by the user to link the runtime metrics to a specific solver configuration.

CI/CD based workflows are usually stateless. As a result the output of the pipeline is discarded. The performance metrics are therefore sent to a PostgreSQL instance and the Score-P experiment directory to the object store.

For production, the metadata collection steps are the same as in the CI case. If the CI infrastructure is available during these runs, it can be used. Otherwise, the backend can be switched from PostgreSQL to a local SQLite database, which stores the information offline. They can later be manually integrated into the global database.

4. Results

Table 1 shows an example result of a database query sorted by average runtime. The database contains metadata from several NPB BT benchmark runs, each with a different configuration of core pinning and core/thread ratio. The `intel2022a` test case without thread pinning is slower than the other runs with the same 4 processes - 26 threads configuration and presents a good candidate for further investigation.

The example above is just one of many possible use cases. Directly linking performance data to solver and cluster configuration allows for advanced analysis on a homogeneous RDF data basis, leveling the path to identifying far-reaching correlations.

5. Summary and outlook

The work presented here is a minimal but versatile approach to continuous performance tracking. We introduce Score-P's metadata schema and propose its integration into the HPC ontology. We give recommendations on what additional metadata the user should collect. We provide tooling for transforming performance-related data into an RDF graph, tools for storing the trace information either locally or in a database + object storage, and predefined

[3] `https://pop-coe.eu/`

queries for retrieval of common questions. These tools only require basic infrastructure to be used successfully.

Future work may include the integration of system-wide monitoring data for an even broader execution context. Models of sensitivity to cross-influence from other jobs on shared HPC systems may be built upon these integration. Native graph analysis algorithms could also be employed for further examination.

6. Acknowledgements

The work presented in this paper was conducted within the framework of the DARWIN research project (20D1911C), funded by Rolls-Royce Deutschland Ltd & Co KG and the Federal Ministry for Economic Affairs and Climate Action. The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR[4].

References

[1] A. Knüpfer, C. Rössel, D. a. Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, et al., Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir, in: Tools for High Performance Computing 2011: Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing, September 2011, ZIH, Dresden, Springer, 2012, pp. 79–91. doi:10.1007/978-3-642-31476-6_7.

[2] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, W. E. Nagel, The vampir performance analysis tool-set, in: Tools for High Performance Computing: Proceedings of the 2nd International Workshop on Parallel Tools for High Performance Computing, July 2008, HLRS, Stuttgart, Springer, 2008, pp. 139–155. doi:10.1007/978-3-540-68564-7_9.

[3] M. Geimer, B. Kuhlmann, F. Pulatova, F. Wolf, B. J. Wylie, Scalable collation and presentation of call-path profile data with cube., in: PARCO, 2007, pp. 645–652, https://hdl.handle.net/2128/4809.

[4] A. Calotoiu, T. Hoefler, M. Poke, F. Wolf, Using automated performance modeling to find scalability bugs in complex codes, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2013, pp. 1–12. doi:10.1145/2503210.2503277.

[5] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (1) (2016) 1–9. doi:10.1038/sdata.2016.18.

[6] D. Krech, G. A. Grimnes, G. Higgins, J. Hees, I. Aucamp, N. Lindström, N. Arndt, A. Sommer, E. Chuc, I. Herman, A. Nelson, J. McCusker, T. Gillespie, T. Kluyver, F. Ludwig, P.-A. Champin, M. Watts, U. Holzer, E. Summers, W. Morriss, D. Winston, D. Perttula, F. Kovacevic, R. Chateauneu, H. Solbrig, B. Cogrel, V. Stuart, RDFLib (2023). doi:10.5281/zenodo.6845245.
URL https://github.com/RDFLib/rdflib

[7] D. Dai, Y. Chen, P. Carns, J. Jenkins, W. Zhang, R. Ross, Managing rich metadata in high-performance computing systems using a graph model, IEEE Transactions on Parallel and Distributed Systems 30 (7) (2018) 1613–1627. doi:10.1109/TPDS.2018.2887380.

[8] R. T. Evans, J. C. Browne, W. L. Barth, Understanding application and system performance through system-wide monitoring, in: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2016, pp. 1702–1710. doi:10.1109/IPDPSW.2016.145.

[9] D. Boehme, P. Aschwanden, O. Pearce, K. Weiss, M. LeGendre, Ubiquitous performance analysis, in: High Performance Computing: 36th International Conference, ISC High Performance 2021, Virtual Event, June 24–July 2, 2021, Proceedings 36, Springer, 2021, pp. 431–449. doi:10.1007/978-3-030-78713-4_23.

[10] C. Liao, P.-H. Lin, G. Verma, T. Vanderbruggen, M. Emani, Z. Nan, X. Shen, Hpc ontology: Towards a unified ontology for managing training datasets and ai models for high-performance computing, in: 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC), IEEE, 2021, pp. 69–80. doi:10.1109/MLHPC54614.2021.00012.

[4] www.nhr-verein.de/unsere-partner

Mini-Symposium 7:

Lattice Boltzmann Method-Based Computational Fluid Dynamics and its Application

Organizers: Amirul Khan and Alessandro de Rosis

The lattice Boltzmann method (LBM) stands as a versatile and powerful computational tool for simulating fluid dynamics and related phenomena. With its unique mesoscopic approach, LBM has gained significant attention for its ability to accurately model complex flows, including multiphase flows, turbulent flows, and flows through porous media. This mini-symposium aims to provide a platform for researchers and practitioners to exchange ideas, discuss recent advancements, and explore emerging applications of the lattice Boltzmann method.

Topics of interest include:

- fundamental developments in LBM;
- novel computational techniques for enhancing simulation efficiency and accuracy, including hardware acceleration and parallelisation strategies;
- applications of LBM in industrial and environmental fluid dynamics;
- challenges associated with coupling LBM with other numerical methods;
- machine learning-based approaches for improving the efficiency, accuracy, and applicability of LBM.

Additionally, the mini-symposium will address issues related to the validation, verification, and benchmarking of LBM simulations, as well as initiatives in the development and dissemination of open-source LBM codes.

Researchers at all career stages are invited to contribute oral presentations. We welcome experts and enthusiasts from academia and industry. This mini-symposium seeks to advance our understanding of LBM, stimulate interdisciplinary discussions, and inspire future innovations in LBM-based computational fluid dynamics.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

On the Influence of the Blending Parameter σ in LBM WMLES With the HRR-BGK Collision Scheme

Jana Gericke[a,*], Kannan Masilamani[a], Harald Klimach[a] and Gregorio Spinelli

[a]German Aerospace Center (DLR), Institute of Software Methods for Product Virtualization, Nöthnitzer Str. 46, 01187 Dresden, Germany
Postal Address: c/o Technische Universität Dresden, Helmholtzstraße 10, 01069 Dresden, Germany

ARTICLE INFO†

Keywords:
Lattice Boltzmann Method;
Wall Modeled Large-Eddy Simulation;
Turbulent Channel Flow;
Parallel Performance;
Regularized Collision Schemes

ABSTRACT

This work investigates the influence of the blending parameter σ on the Hybrid Recursive Regularized BGK collision scheme with and without a correction term on the selection of LES and wall model. The comparison is done using the well-known turbulent channel flow test case at a friction REYNOLDS number Re_τ of 1,000. The wall is either modeled via the implicit Musker profile or explicitly with a combination of the Werner and Wengle and the Schmitt profile. The considered LES models are the Wall-Adapting Local Eddy-viscosity (WALE) and the Vreman model. For the discretization of the velocity space both, the D3Q19 and D3Q27 stencil are considered. Different resolutions of the channel are studied, leading to y^+ values of the first cell at the wall $y_1^+ = 12.5, 25$ and 50. The entire range of σ values in the interval [0.0, 1.0] is analyzed.

1. Introduction

Nowadays, to perform high-fidelity simulations, high performance computing (HPC) and the use of significant computational resources are indispensable. The Lattice Boltzmann Method (LBM) offers an efficient, explicit high-fidelity fluid simulation tool. However, optimal parameters for turbulence and wall modeling in the multitude of possible LBM implementations are not well understood yet. Therefore, we employ the massively-parallel LBM solver *Musubi* [1] in our investigations to extend the previous contributions: Spinelli et al. [2, 3] performed a systematic study of collision schemes, LES and wall models on accuracy and efficiency for turbulent channel flow (TCF) and flow past a circular cylinder using the LBM solver *Musubi*. Their investigations revealed that the suitability of a collision scheme for a certain test case depends on the combination in which it is used (e.g. LES and wall model). This has been observed for the Hybrid Recursive Regularized BGK (HRR-BGK) collision scheme [4] in particular. Besides, HRR-BGK makes use of a blending parameter σ. Thus, last year, Spinelli and Gericke [5] extended these investigations by evaluating the influence of σ on the HRR-BGK as it impacts the numerical dissipation of the scheme. In their investigations, they simulated the TCF by employing a fixed combination of the Vreman LES model and the Musker wall model. They showed that adjusting σ dependent on the stencil as well as the resolution improves the accuracy. This study extends the previous research by analyzing the influence of σ in the HRR-BGK scheme on the selection of LES and wall model by comparing accuracy and performance. The two considered LES models are the Vreman [6] and the WALE [7]. The latter one tends to provide more accurate results, while the Vreman model typically is faster [3]. The wall is modeled by the implicit Musker profile [8] or the explicit combination of the Werner and Wengle [9] and the Schmitt [10] profile, called Power-Law profile (see also [3]). To further enhance the HRR-BGK scheme, the correction term as proposed by Feng et al. [11] is utilized. The velocity space is discretized with both, the D3Q19 and D3Q27 stencil.

2. Method

The method is based on the Lattice Boltzmann equation (LBE). In its post-collision expression, the LBE extended by the correction term ψ_i of Feng et al. [11] is written as

$$f_i^\star(\mathbf{x},\mathrm{t}) = f_i^{\mathrm{eq}}(\mathbf{x},t) + (1-\omega_i) f_i^{\mathrm{neq}}(\mathbf{x},\mathrm{t}) + 0.5\Delta\mathrm{t}\psi_i. \quad (1)$$

f_i is the discrete Probability Density Function (PDF) of particles streaming from one lattice node $\mathbf{x}$ to an adjacent one $\mathbf{x}+\mathbf{c}_i\Delta\mathrm{t}$, $\mathbf{c}_i$ is the lattice velocity and $\Delta\mathrm{t}$ the time increment. The equilibrium and non-equilibrium PDFs are given as

$$f_i^{\mathrm{eq}}(\mathbf{x},t) = w_i \sum_{i=0}^{q=n-1} \frac{1}{n!} \mathbf{a}_0^{(n)} \mathcal{H}_i^{(n)}, \quad (2)$$

$$f_i^{\mathrm{neq}}(\mathbf{x},t) = w_i \sum_{i=2}^{q=n-1} \frac{1}{n!} \mathbf{a}_1^{(n)} \mathcal{H}_i^{(n)}. \quad (3)$$

†This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02489 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
jana.gericke@dlr.de (J. Gericke); kannan.masilamani@dlr.de (K. Masilamani); harald.klimach@dlr.de (H. Klimach)
ORCID(s): 0000-0003-0322-2197 (J. Gericke); 0000-0002-3640-2154 (K. Masilamani); 0000-0002-6054-5681 (H. Klimach); 0000-0001-6578-2494 (G. Spinelli)

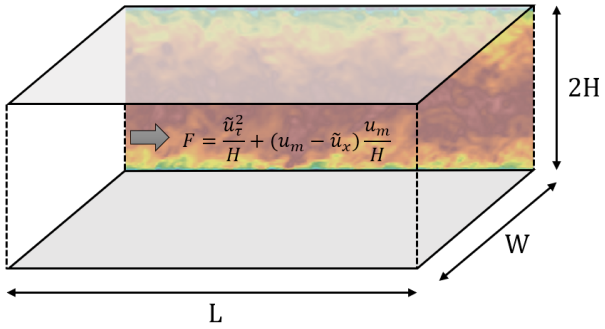


Figure 1: Computational domain for the turbulent channel flow with superimposed flow field and the force driving the flow in horizontal direction. Walls are indicated in grey.

The Hermite polynomials $\mathcal{H}_i^{(n)}$ as well as the Hermite coefficients $\mathbf{a}_0^{(n)}$ and $\mathbf{a}_1^{(n)}$ are chosen as stated in [5]. For this investigation, the HRR-BGK collision scheme introduced by Jacob et al. [4], is used. The second-order Hermite coefficient of f^{neq} with the blending parameter σ is then given given as

$$\mathbf{a}_1^{(2,\text{HRR})} = \sigma \mathbf{a}_1^{(2,\text{RR})} + (1-\sigma)\sigma \mathbf{a}_1^{(2,\text{FD})}, \tag{4}$$

with $0 \leq \sigma \leq 1$. If $\sigma = 0$, $\mathbf{a}_1^{(2,\text{HRR})}$ is reconstructed by the velocity gradients calculated by Finite Difference (FD). This is what Spinelli et al. called Projected Recursive Regularization BGK (PRR-BGK) scheme. It is stable, but highly dissipative [2]. For $\sigma = 1.0$, $\mathbf{a}_1^{(2,\text{HRR})}$ leads to the Recursive Regularized BGK (RR-BGK) scheme proposed by Malaspinas [12].

3. Test case and results

The well-known turbulent channel flow at a Re_τ of 1,000 is used as test case [13]. Its computational domain is shown in Fig. 1.

A test case description, including boundary conditions and domain size (they also investigated the effect of the domain size) is offered in [3]. To account for the entire range of possible σ values, the following parameters are used: $[0.0, 0.5, 0.9, 0.92, 0.94, 0.96, 0.98, 1.0]$. For Vreman, the model coefficient is set to 0.07, for WALE to 0.5. The results are compared to the ones with Vreman and Musker of [5] extended by the runs with $\sigma = 0.0$ and $\sigma = 0.5$ to have a complete dataset. As a reference, DNS data by Lee and Moser [13] is used. Results show that the optimal σ-range lies between 0.9 and 1.0 revealing the PRR-BGK scheme is not suited for this specific test case. Using the correction term makes HRR-BGK slower but it does not increase the accuracy. Regarding the LES model, there is no much difference in terms of accuracy if Vreman or WALE (in conjunction with Musker wall function) is used. As the Vreman model has less floating point operations compared to WALE, it is preferable. Furthermore, the choice of σ depends on the wall model: For Vreman with Musker the best results are obtained for $\sigma = 0.9$, while the optimum for Vreman with Power-Law is $\sigma = 0.94$. Finally, we can take the node-level performance of these collision schemes into account: It was recently investigated by Wendler et al. [14]. They showed that the D3Q19 is ≈ 1.5 times faster than the D3Q27, due to the reduced number of links. Furthermore, they revealed that the HRR-BGK is ≈ 1.5 times faster than the one with the correction term but only improves accuracy by 0.1 % [3]. As the HRR-BGK schemes are implemented with σ as a factor, its choice has no impact on the performance. Contrary to that, the stencil has. Thus, for the TCF test case we can conclude that in terms of balancing accuracy and computational effort, the combination of Vreman-Power-Law-HRR-BGK-D3Q19 with a σ value of 0.94 and without correction term gives the best result independent of the resolution.

4. Conclusions

We investigated the influence of the blending parameter σ on the Hybrid Recursive Regularized BGK (HRR-BGK) collision scheme with and without a correction term on the selection of LES and wall model using the turbulent channel flow test case ($Re_\tau = 1{,}000$). Aiming to determine the influence of σ while identifying the optimal combination of it with LES and WM that balances cost-effectiveness with accuracy we can conclude that this optimum is achieved by the Vreman-Power-Law-HRR-BGK-D3Q19 combination with a σ value of 0.94 and without the correction term. To

enable a more robust validation of the findings, future work will include additional test cases.

References

[1] M. Hasert, K. Masilamani, S. Zimny, H. Klimach, J. Qi, J. Bernsdorf, S. Roller, Complex fluid simulations with the parallel tree-based Lattice Boltzmann solver Musubi, Journal of Computational Science 5 (5) (2014) 784–794. doi:10.1016/j.jocs.2013.11.001.

[2] G. G. Spinelli, T. Horstmann, K. Masilamani, M. M. Soni, H. Klimach, A. Stück, S. Roller, HPC performance study of different collision models using the Lattice Boltzmann solver Musubi, Computers & Fluids 255 (2023) 105833. doi:10.1016/j.compfluid.2023.105833.

[3] G. G. Spinelli, J. Gericke, K. Masilamani, H. G. Klimach, Key ingredients for wall-modeled LES with the Lattice Boltzmann method: Systematic comparison of collision schemes, SGS models, and wall functions on simulation accuracy and efficiency for turbulent channel flow, Discrete and Continuous Dynamical Systems - S 17 (11) (2024) 3224–3251. doi:10.3934/dcdss.2023212.

[4] J. Jacob, O. Malaspinas, P. Sagaut, A new hybrid recursive regularised Bhatnagar–Gross–Krook collision model for Lattice Boltzmann method-based large eddy simulation, Journal of Turbulence 19 (11-12) (2018) 1051–1076. doi:10.1080/14685248.2018.1540879.

[5] G. G. Spinelli, J. Gericke, Influence of the blending parameter σ on the hybrid recursive regularized BGK collision scheme for turbulent LBM simulations, in: 34th International Conference on Parallel Computational Fluid Dynamics - Proceedings, Cuenca, 2023, p. 5.

[6] B. Vreman, B. Geurts, H. Kuerten, On the formulation of the dynamic mixed subgrid-scale model, Physics of Fluids 6 (12) (1994) 4057–4059. doi:10.1063/1.868333.

[7] F. Nicoud, F. Ducros, Subgrid-scale stress modelling based on the square of the velocity gradient tensor, Flow, Turbulence and Combustion 62 (3) (1999) 183–200. doi:10.1023/A:1009995426001.

[8] A. J. Musker, Explicit Expression for the Smooth Wall Velocity Distribution in a Turbulent Boundary Layer, AIAA Journal 17 (6) (1979) 655–657. doi:10.2514/3.61193.

[9] H. Wengle, H. Werner, Large-eddy Simulation of Turbulent Flow Over Sharp-edged Obstacles in a Plate Channel, in: Physics of Separated Flows - Numerical, Experimental, and Theoretical Aspects, Notes on Numerical Fluid Mechanics (NNFM), Vol. 40, Vieweg+Teubner Verlag, Wiesbaden, 1993, pp. 192–199. doi:10.1007/978-3-663-13986-7_26.

[10] L. Schmitt, Grobstruktursimulation turbulenter Grenzschicht-, Kanal-und Stufenströmungen, Ph.D. thesis (1988).

[11] Y. Feng, P. Boivin, J. Jacob, P. Sagaut, Hybrid recursive regularized thermal lattice Boltzmann model for high subsonic compressible flows, Journal of Computational Physics 394 (2019) 82–99. doi:10.1016/j.jcp.2019.05.031.

[12] O. Malaspinas, Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization, ArXiv e-prints (2015). doi:10.48550/arXiv.1505.06900.

[13] M. Lee, R. D. Moser, Direct numerical simulation of turbulent channel flow up to, Journal of Fluid Mechanics 774 (2015) 395–415. doi:10.1017/jfm.2015.268.

[14] J. Wendler, J. Gericke, M. Cristofaro, N. Ebrahimi Pour, I. Huismann, Accelerating the FlowSimulator: Node-Level Performance Analysis of High-Performance CFD Solvers using LIKWID, in: International Parallel Tools Workshop 2023, (accepted for publication).

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Numerical Simulation of the Effects of Internal and External Viscosity Contrast of a Red Blood Cell in a Non-Newtonian Plasma on Its Motion and Suspension Rheology

Haruki Morimoto[a,*], Tomohiro Fukui[a]

[a]Kyoto institute of technology, Kyoto, Japan

ARTICLE INFO[†]

Keywords:
Rheology;
Suspension;
Power-law index model

ABSTRACT

Blood is composed of plasma, which is the liquid component, and red blood cells (RBCs), white blood cells, and platelets, which are the physical components. Plasma, the solvent portion of blood, is generally treated as a Newtonian fluid. However, plasma is known to exhibit non-Newtonian properties depending on the plasma protein content. RBCs are known to exhibit different behaviors in blood, namely, tank-treading, and tumbling motions. These motions are important for understanding the rheological properties of RBC suspensions because they have a significant effect on the apparent viscosity of blood. As a fundamental study, the effect of the reduced area of RBCs on their motion was investigated. Focusing on the non-Newtonian properties of plasma, the deformation behavior of single red blood cells in two-dimensional shear flow was analyzed. As a result, it was confirmed that the results in non-Newtonian fluid are different compared to the results in Newtonian fluid of others.

1. Introduction

Suspensions are used in industrial and medical applications, and it is important to understand their rheological properties. Blood is a suspension with tangible components such as RBCs suspended in plasma and exhibits non-Newtonian properties. The mode of motion of RBCs changes depending on the properties of the surrounding plasma and RBCs. The main modes of motion are tank-treading motion, in which only the membrane of the RBCs rotates, and tumbling motion, in which the RBCs themselves rotate. This change in motion has a significant effect on the rheological properties of macroscopic RBC suspensions because it changes the local fluid resistance of the blood. Most studies have treated plasma as a Newtonian fluid, and few studies have focused on the effect of the non-Newtonian nature of plasma on the critical point of RBC motion, the critical internal and external viscosity contrast. In this study, we focus on the critical internal and external viscosity contrast to investigate the effect of the non-Newtonian property of plasma on the viscosity of blood. As a fundamental study, we investigated the effects of non-Newtonian properties on the membrane velocity and inclination angle of RBC in shear flow by comparing the results of others.

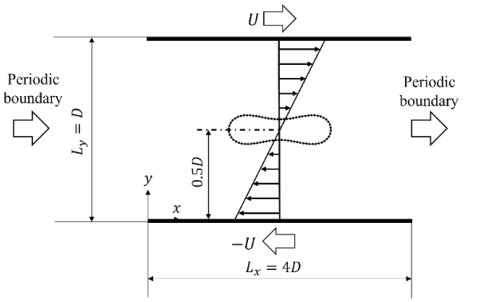


Figure 1: Schematic diagram of the computational model.

2. Methods

2.1. Computational Model

Figure 1 shows a schematic diagram of the computational model used in this study. The red blood cell is at the center of the $4D \times D$ computational domain. The virtual flux method [1], which can accurately determine physical quantities on and near the surface of an object, was used.

2.2. Governing Equation for Fluid

The regularized lattice Boltzmann equation was used as the governing equation for the fluid [2].

The power-law index model [3], which is easy to model as a non-Newtonian fluid, was used. In the equations $n < 1, n = 1, n > 1$ correspond to shear-thinning fluid, Newtonian fluid, and shear-thickening fluid, respectively.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02490 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
m3623033@edu.kit.ac.jp (H. Morimoto); fukui@kit.ac.jp (T. Fukui)
ORCID(s): - (H. Morimoto); - (T. Fukui)

$$\nu(\dot{\gamma}) = m\,|\dot{\gamma}|^{n-1} \tag{1}$$

2.3. Governing Equation for Particle

The spring network model [4] was used as the governing equation for the particle structure. The spring network model performs motion and deformation so that the elastic energy for stretching and bending of the membrane is minimized under constant volume conditions. Since this analysis was two-dimensional, a red blood cell membrane particle was modeled as a red blood cell with N mass points. The membrane particles are held together by springs that resist membrane stretching, deformation, and bending, respectively. The reduced area α is the ratio of the target area of deformation to the area of the true circle under constant perimeter length conditions.

2.4. Parallel computing

All simulations were conducted using parallel computation with OpenMP. The configurations are Intel core i9-12900: clock frequency, 3.2 Hz; number of cores, 16; memory size, 24 GB.

3. Result and Discussion

The efficiency of parallel computation was evaluated. Figure 2 shows the relationship between speed up ratio and the number of threads.Here, the computation time is treated as the time from the start of the computation to the output of the membrane velocity and inclination angle values in the present analysis. Figure 2 shows that the speed up ratio tends to increase as the number of threads increases. However, increasing the number of threads from 8 to 16 did little to change the rate of speedup.This is considered to be because the overhead of parallel computation of the virtual flux method has increased due to the increase in the number of threads.

The effect of reduced area on membrane velocity and inclination angle in non-Newtonian fluids was investigated. In order to compare our results with those of others, the number of lattices to representative lengths were set to $D = 256$, 128 with confinement $C = 0.4$, 0.8 , respectively. $N = 96$ and for the reduced area, $\alpha = 0.60$, 0.70 , 0.80 , 0.90 were used. Figure 3 shows the relationship between membrane velocity and reduced area along with the results of previous studies. Here, the membrane velocity is the average of the non-dimensionalized values of $r_0\dot{\gamma}/2$, which is the theoretical value of the rotational velocity of a circular vesicle flowing in a shear flow at low confinement [5]. Figure 3 shows that the membrane velocity tends to increase as the reduced area increases. Since the strain rate changes in the case of a non-Newtonian fluid, the results of this analysis are considered to

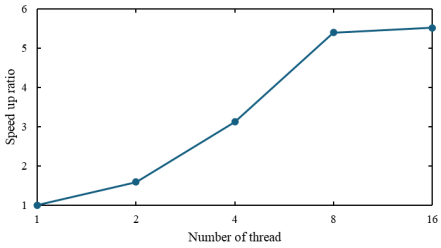


Figure 2: Speed up ratio.

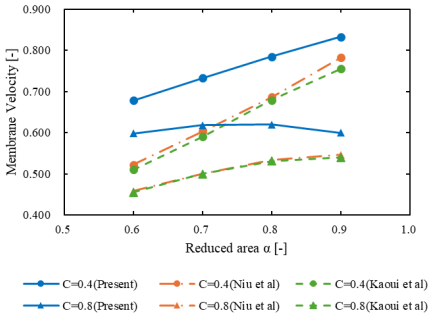


Figure 3: Membrane velocity of a single RBC.

be different from those obtained in the case of a Newtonian fluid. Figure 4 shows the relationship between inclination angle and reduced area along with the results of previous studies. Figure 4 shows that the inclination angle tends to increase as the reduced area increases. These results are in qualitative agreement with those of Niu et al. [6] and Kaoui et al. [7]. Moreover, As in previous experimental studies with a single red blood cell in shear flow [8], the red blood cell exhibited tank-treading motion. The present results for non-Newtonian fluids differ from those of others for Newtonian fluids in the effect of reduced area on membrane velocity. In this study, we conducted a two-dimensional analysis, but when extending to three dimensions, a pattern called the vacillating-breathing motion mode also arises. Therefore, it is necessary to consider this motion mode when extending to three dimensions in the future.

4. Conclusions

In this study, the effects of reduced area on film velocity and inclination angle were investigated. In future studies, the

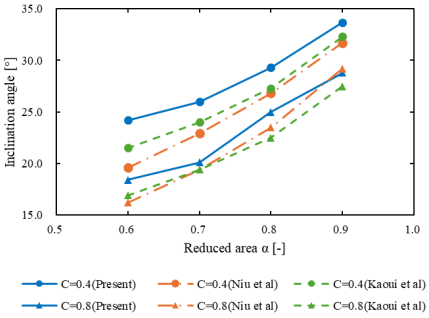


Figure 4: Inclination angle of a single RBC.

effect of the internal and external viscosity contrast on RBC will be investigated. The effects of parameters of RBC such as capillary number and reduced area on the critical internal and external viscosity contrast will be also investigated.

References

[1] I. Tano, K. Morinishi, K. Matsuno, H. Nishida, Validation of virtual flux method for forced convection flow, JSME International Journal Series B 49 (4) (2006) 1141–1148. doi:10.1299/jsmeb.49.1141.

[2] M. Izham, T. Fukui, K. Morinishi, Application of regularized lattice boltzmann method for incompressible flow simulation at high reynolds number and flow with curved boundary, Journal of Fluid Science and Technology 6 (6) (2011) 812–822. doi:10.1299/jfst.6.812.

[3] J. Boyd, J. Buick, S. Green, A second-order accurate lattice boltzmann non-newtonian flow model, Journal of Physics A: Mathematical and General 39 (46) (2006) 14241–14247. doi:10.1088/0305-4470/39/46/001.

[4] K.-i. Tsubota, S. Wada, T. YamaguchiI, Simulation study on effects of hematocrit on blood flow properties using particle method, Journal of Biomechanical Science and Engineering 1 (1) (2006) 159–170. doi:10.1299/jbse.1.159.

[5] B. Kaoui, T. Krüger, J. Harting, How does confinement affect the dynamics of viscous vesicles and red blood cells?, Soft Matter 8 (35) (2012) 9246. doi:10.1039/c2sm26289d.

[6] X. Niu, L. Shi, T.-W. Pan, R. Glowinski, Numerical simulation of the motion of inextensible capsules in shear flow under the effect of the natural state, Communications in Computational Physics 18 (3) (2015) 787–807. doi:10.4208/cicp.260714.260315a.

[7] B. Kaoui, J. Harting, C. Misbah, Two-dimensional vesicle dynamics under shear flow: Effect of confinement, Physical Review E 83 (6) (Jun. 2011). doi:10.1103/physreve.83.066319.

[8] T. M. Fischer, M. Stöhr-Liesen, H. Schmid-Schönbein, The red cell as a fluid droplet: Tank tread-like motion of the human erythrocyte membrane in shear flow, Science 202 (4370) (1978) 894–896. doi:10.1126/science.715448.

Mini-Symposium 8:

Machine Learning-Based Reduced Order Models for Fluid Flow Emulators and Application to Design Optimization

Organizers: Amirul Khan and He Wang

Machine Learning (ML) has emerged as a powerful tool in the development of Reduced Order Models (ROMs) for computational fluid dynamics (CFD) surrogates or emulators, particularly in the context of multidisciplinary design optimisation (MDO). The integration of ML with ROMs offers promising avenues for efficient and accurate predictions, making it well-suited for high-performance computing. In this mini-symposium, we invite contributions that reflect the rapid advancements in ML-based ROMs for the creation of CFD surrogates or emulators and their applications to MDO. We also welcome contributions that explore other ML-based methods and their applications.

Contributions can cover, but are not limited to, the following topics:

- Development and application of non-intrusive ML-based ROMs.
- Uncertainty quantification and robust design using ML-enhanced CFD emulators/surrogates.
- Case studies showcasing the application of ML-based ROMs.
- Challenges and solutions in integrating ML-based ROMs with design optimisation.
- Future directions in the integration of ML and ROMs for design optimization.
- Applications in Multidisciplinary Design Optimization:

    1. Aerospace vehicle design and aerodynamics
    2. Automotive engineering and vehicle performance optimisation
    3. Renewable energy system design and optimization
    4. Biomedical device design and fluid-structure interaction studies

This mini-symposium aims to foster discussions and collaborations among researchers, academicians, and industry professionals interested in the application of ML-based ROMs for flow emulators in design optimization. We look forward to your valuable contributions.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Machine Learning-Based Intelligent CFD Surrogates for Interactive Design Exploration of Built Environments

Usamah Adia[a,*], Amirul Khan[a], Andrew Sleigh[a] and He Wang[b]

[a]University of Leeds, School of Computing, Woodhouse, Leeds, LS2 9JT, UK
[b]University College London, Department of Computer Science, Gower Street, London, WC1E 6BT, UK

ARTICLE INFO†

Keywords:
Computational Fluid Dynamics;
Built Environment;
Machine Learning;
Reduced Order Models

ABSTRACT

This study focuses on the application of reduced order models (ROMs) and various machine learning (ML) techniques to create surrogate computational fluid dynamics (CFD) models for the built environment. By using functional principal component analysis (FPCA), these surrogates can reconstruct flow fields for different geometries significantly faster than traditional CFD methods, providing insights into factors such as thermal comfort and ventilation. The goal is to develop a model that can predict the effects of different geometries or indoor space configurations and produce accurate fluid/air flow results without the need for retraining, enabling rapid, interactive design exploration and optimization.

1. Motivations

Civil engineers must take into account numerous factors when designing the built environment. While the aesthetics of the buildings are important, creating a comfortable and safe environment for work is of greater importance. Understanding the impact of airflow is crucial as it directly influences other design properties, including thermal comfort and energy efficiency [1]. Designing a good system for thermal comfort is essential, research has been done to suggest that temperatures which deviate too far from room temperature can cause a decrease in mental performance, in places such as schools thermal conditions are of great concern [2]. Other issues such as ventilation are of equal concern, when there is insufficient ventilation in enclosed environments there will be a build-up of pollutants such as CO_2 causing what is known as stale air [3]. This is a great enough issue that the term "Sick building syndrome" has been coined, this has led to several symptoms such as skin/eye irritations, headaches, fatigue, nausea, vomiting, and a decrease in concentration [4]. Highlighting the importance of incorporating fluid flow early in the design process which is not feasible with traditional CFD due to the large amount of computational expenses, a faster more efficient method will be required. Creating a machine learning (ML)-based surrogate model offers a potential solution, as CFD generates extensive data that can be leveraged by ML methods to simulate fluid flow or airflow efficiently and quickly. Once trained, the ML surrogate model can generate new flow fields within seconds, enabling fast, interactive modeling of thermal comfort, infection risk, and pollution dispersion with respect to various design parameters. (geometry creation, meshing, solution generation and post-processing).

2. Gap in Literature

This work combines multiple computational and machine-learning methods to produce fast fluid dynamic surrogates. Similar work has been done in the past by several authors for interactive design. Interactive aerodynamic design of 3D structures has been done in the past where a large amount of training data was used with a Gaussian process regressor to produce velocity, drag and pressure values [5]. Other work has been done using internal flows analyzing the internal hemodynamics of flow through arteries [6]. Further work has been done on other engineering design applications such as glass facades where other statistical models such as mixture density networks were used to determine failure rates of glass panels due to high stress/strain on different designs [7]. This work will build upon these methods for the built environment, using reduced order models (ROMs) and testing different ML methods to determine the best method. This includes using Gaussian process regressors (GPRs) and other methods such as transformers.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02492 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
scuaa@leeds.ac.uk (U. Adia); a.khan@leeds.ac.uk (A. Khan); p.a.sleigh@leeds.ac.uk (A. Sleigh); he_wang@ucl.ac.uk (H. Wang)
ORCID(s): 0009-0000-9000-4619 (U. Adia); 0000-0002-7521-5458 (A. Khan); 0000-0001-9218-5660 (A. Sleigh); 0000-0002-2281-5679 (H. Wang)

Number of mesh elements	135160
Turbulence model	K-omega SST
Inlet velocity (m/s)	0.08
Turbulent intensity (%)	5
Turbulent viscosity ratio	10
Total number of simulations	484

Table 1
Table describing the boundary conditions of the training data, mesh information, and total number of simulations.

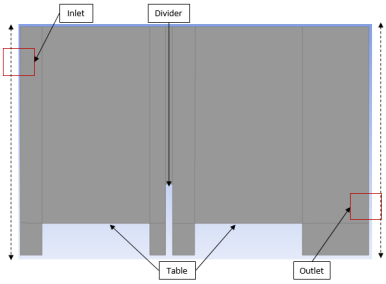


Figure 1: The geometry of the CFD model can be seen which includes the locations of inlet/outlet locations, it is important to note that the dashed lines represent how the inlet/outlet locations move to generate the training data. This geometry is based on a similar 2D model created by [8].

3. Goals

To create these surrogates, several types of machine learning models will be tested such as artificial neural networks, Multilayer Perceptrons, GPRs and transformers. CFD simulations produce a large amount of data which would be fed into these models. ROMs will be used to minimize the required data. Many different ROMs have been considered such as proper orthogonal decomposition (POD), autoencoders and functional principle component analysis (FPCA). FPCA produces eigenfunctions associated with the data, models will be trained to learn how these functions relate to other factors such as geometry. The primary goal is to develop a model capable of adapting to different geometries and producing accurate fluid flow results without the need for retraining. This model would allow engineers and designers to get feedback on the flow field much faster than traditional methods, making design optimization much easier. The project aims to provide a proof of concept demonstrating the viability of combining ROMs and ML techniques and to identify the necessary data for producing accurate models.

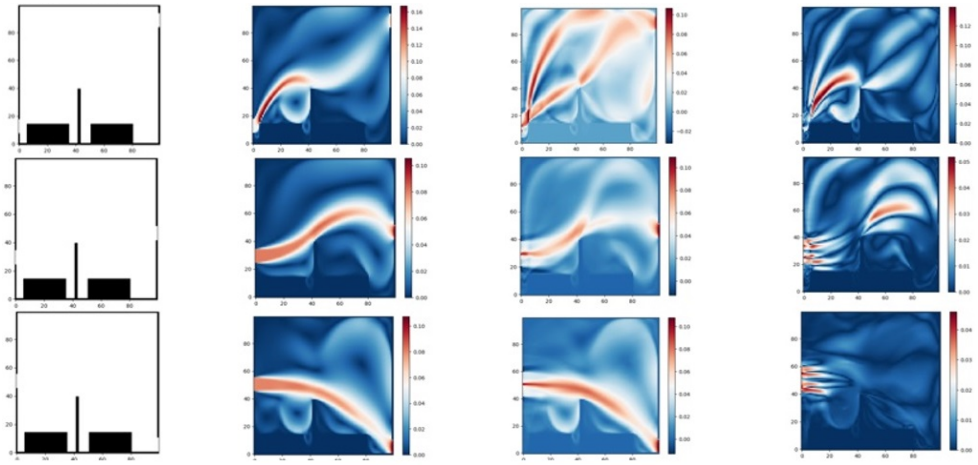
4. Work Done

The work that has been done includes using different ROMs on several configurations for internal flow to better understand specific use cases and how they could be implemented. I have used FPCA to show that it can capture a larger amount of variation using fewer modes when compared to POD, allowing reconstruction to happen with similar accuracy and a lower amount of data. This includes simple cases such as flow past an obstacle and more complex cases such as 2D representations of rooms. To generate the training data I have used Ansys Fluent, the model's geometry can be seen in Fig. 1. This geometry represents a 2D slice of a room which contains two tables and a divider. A fluent script was generated to move the inlet and outlet locations across the dotted lines, providing a range of flow fields for training. The boundary conditions and other information for the CFD simulations can be seen in Tab. 1. The goal here is to apply FPCA to extract the top components of the model, a GPR is then trained to predict several of these components for unknown boundary conditions, which are then used to reconstruct the flow field. Future work will be to train a machine-learning model that can generate modes for different geometries that reconstruct full-flow fields using limited synthetic data sets from CFD. Once the base models have been created there will be a move to deep networks to train a more effective model.

5. Results

Figure 2 shows how the GPR model trained on the ground truth data predicts flow fields based on different inlet/outlet locations. The model accuracy is higher when there is little interaction with obstacles such as walls and dividers. The inlet positions are predicted semi-accurately, where the mode can't find the actual inlet location and attempts to regress based on the two closest inlet locations. This does not occur with the outlet position as these are all known to the GPR model. While these results show that it is possible to use GPR to predict flow fields it takes a significant amount of time for the amount of data used, as for 400 training data simulations it takes over 40 minutes on an average laptop to run. ROMs such as FPCA can be used to increase the training speed.

The variation captured for each of the flow fields in the testing set can be seen in Fig. 3, this is a comparison of FPCA and POD. It shows that FPCA captures a larger range of variation when compared to POD. As for all inlet positions the first component captures at least 60% of the variations compared to POD which is around 25%. This shows that FPCA is better at capturing these variations than traditional POD.



(a) Input geometry. (b) Ground truth for the data. (c) GPR prediction. (d) Absolute error.

Figure 2: Contours plots. The color indicates the velocity magnitude.

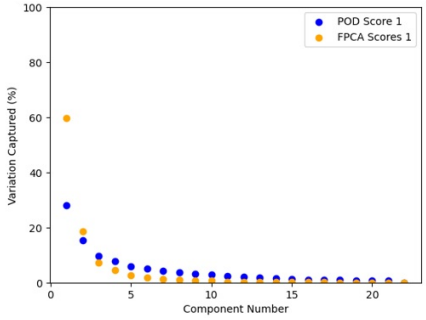
Instead of using a GPR to train the model on a full set of data, it is used to train the principal components, this significantly decreases training time to under a minute. The results can be seen in Fig. 4, the first mode is always predicted accurately and the following modes vary depending on the inlet position. This is seen in Fig. 4c where inlet position 11 has minimal wall interaction, showing that GPR specifically struggles to capture this type of flow field. Something similar can be seen in Fig. 5 where the scores of the FPCA model are being interpolated. This Fig. 5 shows that the inlet position 11 is a near one-for-one perfect prediction while there is a greater error in the other two positions.

By reshaping the matrices obtained from the components and the scores a reconstruction of the flow field can be obtained, the results of these reconstructions can be seen in Fig. 6c. These are then compared to the ground truth data in 6b to get a contour plot of the absolute error (Fig. 6d). The area of greatest error occurs at the inlet and outlet positions, with the error decreasing as the inlet moves higher and away from the walls. Further showing that the model has trouble predicting these interactions, it is, good at predicting the mean at the center of the flow field and can predict the dominant flow feature for most cases.

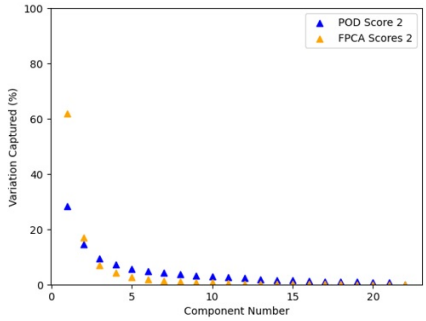
A comparison between FPCA and POD can be seen in Fig. 7 which shows both POD and FPCA can predict inlet positions with a similar accuracy but, FPCA can distinguish the curve associated with the lower outlet position which can be seen in Fig. 7f. Both are unable to capture this flow field with the current model and further work must be done. However, this shows the strengths of using ROMs with machine learning methods as the training time has been dramatically reduced.

6. Conclusions

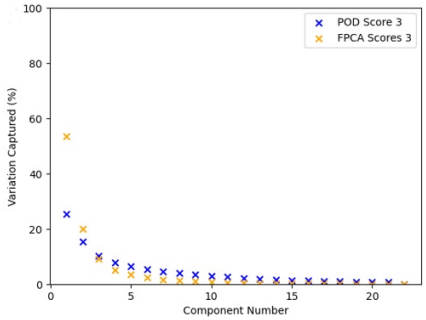
The results show that it is possible to use a GPR to predict the flow fields while varying boundary conditions such as inlet position, this however, requires a large training time. FPCA has been shown to capture a larger range of variations when compared to its predecessor POD, and coupling this ROM with GPR allows for a much faster training time producing good flow fields. However, more work must be done to increase the accuracy of FPCA such as using different ML methods such as Bayesian networks. Recent work suggests the reason GPR struggles is that the model averages the flow fields close to the outlet positions producing the flow that spreads from the top to the bottom, an idea to fix this is to sample grids from the flow field following a similar process where these grids will be projected using POD/FPCA before being predicted with GPR and finally being reconstructed. Future work includes varying other aspects of the geometry such as the divider location and incorporating other factors useful to engineers such as indoor air quality.



(a) Score 1.

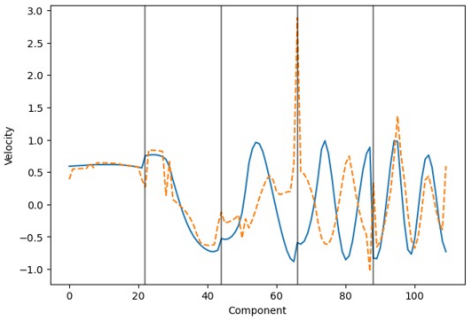


(b) Score 2.

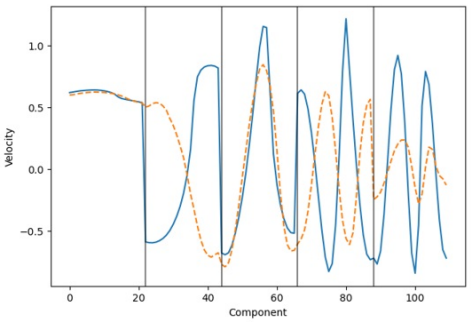


(c) Score 3.

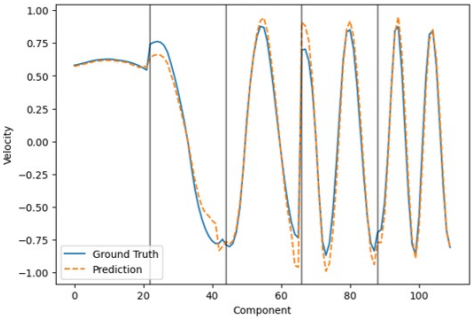
Figure 3: Scatter graphs showing the percentage that each mode captures for three different inlet locations, showing that FPCA captures a larger range of variation with fewer modes.



(a) Inlet position 2.

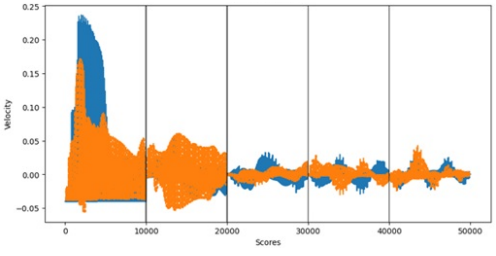


(b) Inlet position 6.

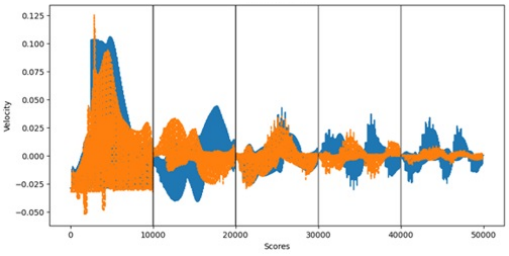


(c) Inlet position 11.

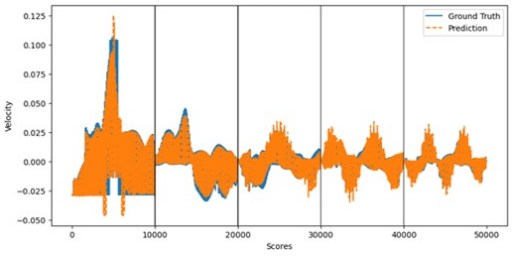
Figure 4: Graphs comparing the predicted components with the ground truth for three separate inlet locations.



(a) Inlet position 2.

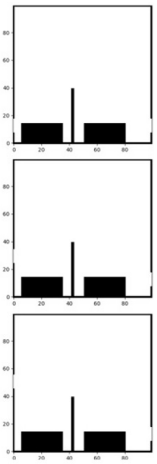


(b) Inlet position 6.

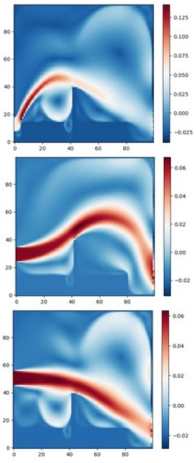


(c) Inlet position 11.

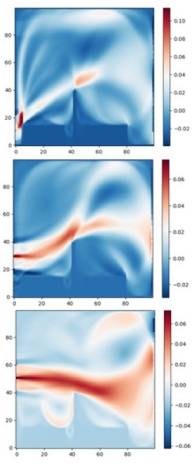
Figure 5: Graphs comparing the predicted scores with the ground truth for three separate inlet locations.



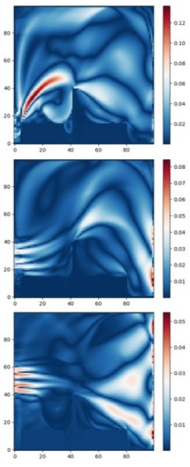
(a) Input geometry.



(b) Ground truth for the reconstructed data.

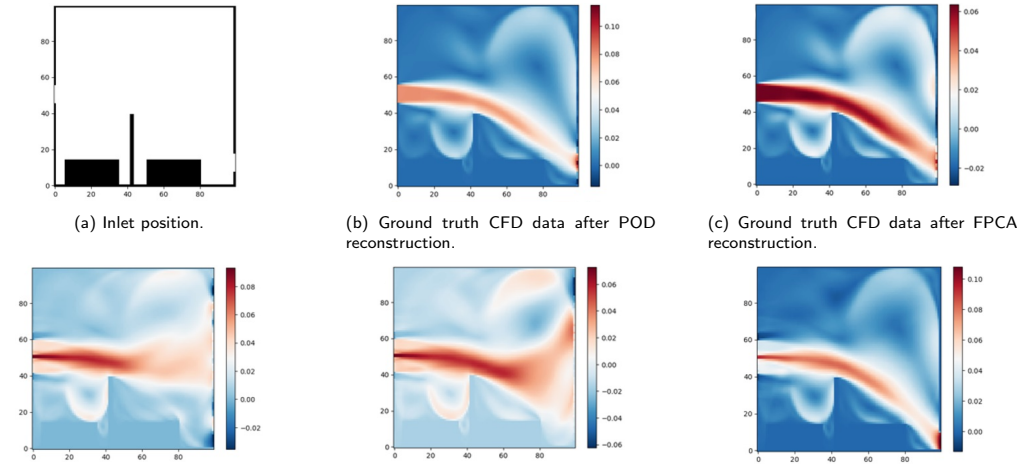


(c) Reconstruction using the eigenfunctions and score predicted by GPR.



(d) Absolute error for three different inlet locations for the FPCA reconstruction using the predicted modes/scores.

Figure 6: Contours plots. The color indicates the velocity magnitude in m/s.



(a) Inlet position.

(b) Ground truth CFD data after POD reconstruction.

(c) Ground truth CFD data after FPCA reconstruction.

(d) GPR prediction using POD data.

(e) GPR prediction using FPCA data.

(f) GPR prediction for the full data set.

Figure 7: Contour plots comparing the flow field for an inlet position. The color indicates the velocity magnitude in m/s.

References

[1] D. H. Chow, Indoor Environmental Quality: Thermal Comfort, in: Encyclopedia of Sustainable Technologies, Elsevier, 2024, pp. 283–295. doi:10.1016/B978-0-323-90386-8.00006-1.

[2] M. J. Mendell, G. A. Heath, Do indoor pollutants and thermal conditions in schools influence student performance? A critical review of the literature, Indoor Air 15 (1) (2005) 27–52. doi:10.1111/j.1600-0668.2004.00320.x.

[3] J. Jansz, Sick building syndrome, in: S. R. Quah (Ed.), International Encyclopedia of Public Health (Second Edition), second edition Edition, Academic Press, Oxford, 2017, pp. 502–505. doi:10.1016/B978-0-12-803678-5.00407-0.

[4] J. Douwes, W. Eduard, P. S. Thorne, Bioaerosols, in: International Encyclopedia of Public Health, Elsevier, 2017, pp. 210–218. doi:10.1016/B978-0-12-803678-5.00032-1.

[5] N. Umetani, B. Bickel, Learning three-dimensional flow for interactive aerodynamic design, ACM Transactions on Graphics 37 (4) (2018) 1–10. doi:10.1145/3197517.3201325.

[6] M. E. Biancolini, K. Capellini, E. Costa, C. Groth, S. Celi, Fast interactive CFD evaluation of hemodynamics assisted by RBF mesh morphing and reduced order models: the case of aTAA modelling, International Journal on Interactive Design and Manufacturing (IJIDeM) 14 (4) (2020) 1227–1238. doi:10.1007/s12008-020-00694-5.

[7] K. Gavriil, R. Guseinov, J. Pérez, D. Pellis, P. Henderson, F. Rist, H. Pottmann, B. Bickel, Computational design of cold bent glass façades, ACM Transactions on Graphics 39 (6) (2020) 1–16. doi:10.1145/3414685.3417843.

[8] S. Hoque, F. B. Omar, Coupling Computational Fluid Dynamics Simulations and Statistical Moments for Designing Healthy Indoor Spaces, International Journal of Environmental Research and Public Health 16 (5) (2019) 800. doi:10.3390/ijerph16050800.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

A Surrogate Model Based Shape Optimization Framework for Compressible Flows

Niyazi Şenol[a,b,*], Hasan U. Akay[a] and Şahin Yiğit[b]

[a]Atılım University, Mechanical Engineering Graduate Program, Kızılcaşar, 1184, Cad No:13, 06830 Ankara, Türkiye
[b]Turkish Aerospace, Computational Fluid Mechanics Department, Fethiye District, Havacılık Avenue No:17, 06980 Ankara, Türkiye

ARTICLE INFO[†]

Keywords:
Aerodynamic Shape Optimization;
Surrogate-Based Methods;
Multiobjective/Multipoint Optimization;
Efficient Global Optimization

ABSTRACT

In this study, a shape optimization framework is designed and developed in order to use surrogate model-based shape optimization methods in the context of compressible flows. While designing the shape optimization framework, various open-source libraries were implemented and made to communicate with each other through Python language. In addition to using Kriging method for surrogate model-based shape optimizations, Gradient Enhanced Kriging (GEK) and Gradient Enhanced Kriging with Partial Least Square (GEKPLS) methods are also investigated. They are investigated from different perspectives and tested on a two-dimensional airfoil as a benchmark case. In addition to optimizations with single-objective functions, as currently modeled using the framework, the capabilities of the framework will be extended to multi-objective and multi-point studies.

1. Introduction

With the advancements in technology, the need for the use of Computational Fluid Dynamics (CFD) in both design and analysis phases in challenging engineering applications in aerospace, defense, energy, and automotive industries increases rapidly. As a result, the need for optimization arises with the aim of improving efficiency of the designed products. The aim of this study is to compare different optimization methods utilized in aerodynamic shape optimizations, which are frequently needed in the aerospace field, by using our recently developed aerodynamic shape optimization framework. For this purpose, a 2D transonic airfoil generic case, which is one of the benchmark cases proposed by the AIAA Aerodynamics Discussion Group (ADODG)[1], is selected, on which three different aerodynamic shape optimization methods are tested. The optimization framework has the flexibility to use different optimization methods based on surrogate models for both single- and multi-objective optimization problems. Optimum design iterations are performed using a method known as the Efficient Global Optimization (EGO) [1] loop until convergence criteria are reached. For this the Expected Improvement (EI) [1, 2] is used during single-objective optimizations, while the Expected Hypervolume Improvement (EHVI) [3] is used during multi-objective optimizations.

2. Methodology

By using the developed framework, it is possible to control both optimization and flow solver stages. Classical optimization steps are performed with the help of EGO. Briefly, these steps comprise the initial design space, selecting the sampling points, building the surrogate models and verifying the optimal design. The Surrogate Modeling Toolbox (SMT) [4], together with smoot[2] and pymoo[3] codes for multi-objective and multi-point optimizations, all of which are open source, are incorporated in the optimization phase. The CFD code SU2 [5], which is also an open source code, is used on the flow solver side. The framework for controlling and manipulating each step is developed in Python programming language. The code SMT has been developed in Python by collaboration of several universities and industrial research groups in USA and Europe for surrogate-based optimization methods with various sampling methods and benchmark problems. It is relatively easy to implement new surrogate models with the basic functions of SMT. Smoot is a code designed for multi-objective optimization cases under the ONERA Lab and is used with Pymoo, which is another code that offers different criteria options. Pymoo provides useful features for both single- and multi-objective optimization-based genetic algorithms.

The purpose of surrogate models is to replace nonlinear objective functions, which are generally expensive and difficult to compute, with models that are cheaper to

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02493 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
niyazi.senol@tai.com.tr (N. Şenol)
ORCID(s): 0000-0003-4528-4150 (N. Şenol); 0000-0003-2574-9942 (H.U. Akay); 0009-0004-1791-6223 (Ş. Yiğit)

[1]https://sites.google.com/view/mcgill-computational-aerogroup/adodg

[2]https://github.com/onera/smoot

[3]https://pymoo.org

compute. Thus, it makes the algorithm more efficient by avoiding the computational burden. In this study, firstly, Kriging method is applied as a surrogate-based optimization method. Theoretically Kriging can be considered as a Radial Basis Function (RBF) model with Gaussian basis. It is a statistical method originally used in the field of geology [6]. Kriging also offers uncertainty estimation as well as mean prediction estimation, which can be easily integrated with EGO. Although the Kriging method provides great advantages in optimization, it has shortcomings in terms of both slowness and accuracy [7]. In this context, steps by Ozkaya et al. [7] have been taken to improve the Kriging method. Secondly, the Gradient Enhanced Kriging (GEK) method has been developed to increase the accuracy of the model by adding gradient information to the training data set. Although different methods provide gradient information during GEK, the adjoint method stands out due to its independence in the number of input variables. The gradient information required by the GEK and GEKPLS methods in their calculations is normally obtained by default in the SMT code with the finite difference method. Considering the advantages of the adjoint method, the gradient information is obtained with the adjoint method with the help of SU2 instead of finite differences in the designed framework. In the GEK model, when the number of input variables and the number of sampling points are high, the size of the GEK correlation matrix becomes large, which leads to a decrease in performance.

For this reason, Bouhlel et al. [8] proposed a new surrogate model called GE-KPLS combining Kriging model with the partial-least squares (PLS) method (KPLS) with GEK [8]. This method incorporates both the advantage of KPLS in reducing the number of Kriging hyperparameters and the advantage of GEK in increasing the accuracy with gradient information, and is particularly successful in performing efficient analyses for high dimensional cases.

Many engineering problems encountered and optimized today inherently involve more than one objective. Therefore, optimization codes are in need of multi-objective algorithms as well. Likewise, especially in the field of aviation, and taking it to the next level, multi-point optimization is used to solve optimization cases that contain more than one operation conditions (such as varying flight conditions) at the same time.

3. Results

The RAE2822 airfoil [9] subjected to transonic flow regime was chosen as the drag minimization case to test the surrogate-based methods in this framework. The flow conditions are

$$Ma = 0.734,$$

C_d	C_l	$S_0(m^2)$
0.0206	0.824	0.0778446

Table 1
RAE2822 aerodynamic coefficients calculated before optimization.

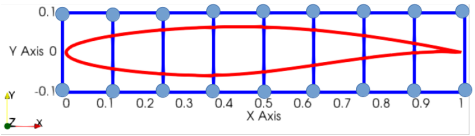


Figure 1: Baseline geometry and FFD box.

$$Re = 6.5 \cdot 10^6.$$

Table 1 presents the calculated aerodynamic coefficients drag and lift for the baseline geometry using SU2 here. These are within the range of values reported by other investigators in the AIAA Aerodynamics Discussion Group [9].

The optimization problem is stated as

$$Min. : C_d, \quad \text{such that:}$$
$$C_l = 0.824, \qquad C_m > 0.092, \qquad S > S_0,$$

where C_d , C_l, and C_m are the drag, lift, and pitching moment coefficients, respectively, and S_0 and S are the initial and optimized airfoil areas, respectively.

The RAE2822 airfoil geometry was parameterized using the Free Form Deformation (FFD) technique and 18 points were selected on the FFD box and the remaining 14 points were defined as design variables so that the corner points are fixed. Figure 1 shows the RAE2822 airfoil and the FFD box around it, respectively.

Table 2 shows the objective function values obtained with each surrogate model, the percentage improvement over the function value obtained with the original airfoil, and elapsed times for each models. As can be seen from this table, the Kriging model gives an improvement of 41.9 %, while the GEK model gives an improvement of 54.1 % and the GEKPLS model gives an improvement of 52.3 %. Here, the GEK and GEKPLS models give very close values, while the GEK model gives a better improvement. As expected elapsed time for Kriging model is less than others since it does not include calculation of gradients. Again, Fig. 2 can be examined to compare the three models. Figure 2 depicts the objective function value obtained in each model for each design iteration. Although the Kriging model produces lower values of drag than the original geometry, it is observed that the oscillations are high, while the GEK model achieves the lowest value compared to both models,

	RAE2822	Kriging	GEK	GEKPLS
C_d	0.02060	0.01197	0.00945	0.00982
$Impr.\ (\%)$	-	41.9	54.1	52.3
$Time\ (min.)$	-	280	350	320

Table 2
Drag coefficient (C_d) values obtained using the three surrogate methods.

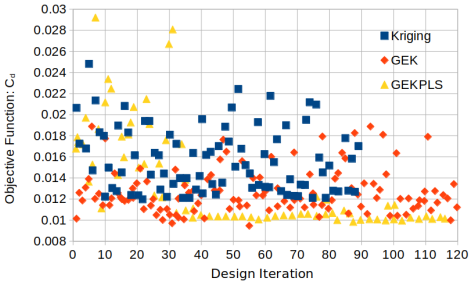


Figure 2: Objective function (C_d) value during design iterations.

but sometimes deviates from this value. In the GEKPLS model, low values are obtained and it is seen that it is close to the lowest value.

4. Conclusions

In this work, an optimization framework designed to include various open-source codes to test different surrogate models was presented. A 2D airfoil case was analyzed using the framework and the results are discussed in detail. It is observed that the Kriging model gives improved results compared to the baseline geometry but is not very stable, while GEK and GEKPLS give much lower drags compared to the baseline airfoil. They are more stable than the Kriging model. Although the work presented here is limited to a single-objective function, multi-objective and multi-point optimization cases on different flow systems will be investigated with the upcoming version of this framework. Our plan for the near future is to work on 3D cases and then solve more advanced cases using the designed framework. Although there are various shape optimization studies reported in the literature with the help of SU2, there is little work with surrogate models, especially with GEKPLS, which is a relatively new method and has been used rarely in the field of aerodynamic shape optimizations. The novelty of this work lies in this aspect of the framework proposed here.

References

[1] Y. Zhang, Z.-H. Han, K.-S. Zhang, Variable-fidelity expected improvement method for efficient global optimization of expensive functions, Structural and Multidisciplinary Optimization 58 (4) (2018) 1431–1451. doi:10.1007/s00158-018-1971-x.

[2] K. Wang, Z. Han, K. Zhang, W. Song, Efficient global aerodynamic shape optimization of a full aircraft configuration considering trimming, Aerospace 10 (8) (2023) 734. doi:10.3390/aerospace10080734.

[3] K. Yang, M. Emmerich, A. Deutz, T. Bäck, Multi-objective bayesian global optimization using expected hypervolume improvement gradient, Swarm and Evolutionary Computation 44 (2019) 945–956. doi:10.1016/j.swevo.2018.10.007.

[4] P. Saves, R. Lafage, N. Bartoli, Y. Diouane, J. Bussemaker, T. Lefebvre, J. T. Hwang, J. Morlier, J. R. Martins, Smt 2.0: A surrogate modeling toolbox with a focus on hierarchical and mixed variables gaussian processes, Advances in Engineering Software 188 (2024) 103571. doi:10.1016/j.advengsoft.2023.103571.

[5] F. Palacios, J. Alonso, K. Duraisamy, M. Colonno, J. Hicken, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, T. Taylor, Stanford university unstructured: An open-source integrated computational environment for multi-physics simulation and design, in: 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, 2013. doi:10.2514/6.2013-287.

[6] D. Krige, A statistical approach to some basic mine valuation problems on the witwatersrand, Journal of the Southern African Institute of Mining and Metallurgy 52 (6) (1951) 119–139. doi:10.10520/AJA0038223X_4792.

[7] E. Özkaya, J. Rottmayer, N. R. Gauger, Gradient enhanced surrogate modeling framework for aerodynamic design optimization, in: AIAA SCITECH 2024 Forum, American Institute of Aeronautics and Astronautics, 2024. doi:10.2514/6.2024-2670.

[8] M. A. Bouhlel, J. R. Martins, Gradient-enhanced kriging for high-dimensional problems, Engineering with Computers 35 (1) (2019) 157–173.

[9] X. He, J. Li, C. A. Mader, A. Yildirim, J. R. Martins, Robust aerodynamic shape optimization—from a circle to an airfoil, Aerospace Science and Technology 87 (2019) 48–61. doi:10.1016/j.ast.2019.01.051.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Controllable Droplet Transport via Inverse Design of Substrate Heterogeneity

Panayiotis-Y. Vrionis[a,*], Andreas Demou[a] and Nikos Savva[b]

[a]Computation-based Science and Technology Research Center, The Cyprus Institute, Aglantzia, Nicosia, 2121, Cyprus
[b]Department of Mathematics and Statistics, University of Cyprus, Aglantzia, Nicosia, 2109, Cyprus

ARTICLE INFO[†]

Keywords:
Wetting Hydrodynamics;
Inverse Design;
Droplet Transport;
Reduced Order Modeling

ABSTRACT

Droplet motion, a phenomenon prevalent in both everyday life and industrial applications, plays a crucial role in processes such as water harvesting, biomedical technologies, and 3D printing. The ability to control droplet transport using heterogeneous substrates can significantly influence the performance of such applications. Herein we explore the usage of inverse design optimization methods to define substrate heterogeneity patterns and achieve targeted droplet transport. By integrating data-driven models with low-fidelity empirical models, a hybrid modeling approach is developed that can predict contact line velocities and facilitate fast and accurate simulations. The results demonstrate the potential of this method to also facilitate the design of substrate patterns that direct droplet movement.

1. Introduction

A phenomenon that occurs in our everyday life and often goes unnoticed is droplet motion. Common everyday examples include droplets sliding on windows due to morning dew or cooling the human body through sweating. In an industrial context, droplet motion is the foundation for applications that deal with water harvesting [1], biomedical technologies [2], and 3D printing methodologies [3].

A key aspect in the study of wetting phenomena and droplet transport, in general, pertains the study of moving contact lines dynamics. The inherent multiscale nature of the underlying dynamics, ranging from the molecular to the continuum scale [4], makes numerically studies of such problems quite challenging. Numerical studies using Direct Numerical Simulations [5, 6] and the Lattice Boltzmann method [7] even though feasible, typically require a daunting amount of computational resources and time. In an effort to reduce the associated computational costs, reduced order models based on the long-wave approximation of the Navier–Stokes equations can offer significant reductions [8, 9]. However, even after these simplifications, the associated costs still require significant efforts [10] rendering them infeasible for practical cases of inverse design and optimization frameworks.

A potential alternative lies in the synergistic use of data-driven models and low-fidelity empirical models [11]. In such approaches, a data-driven model acts as a corrector for the low accuracy reduced order model. The resulting hybrid model can achieve improved accuracy, better generazibility, and requires fewer training samples compared to fully data-driven approaches. In [11] the hybrid model combines the low-fidelity asymptotic model of Lacey [12] with a data-driven model that corrects Lacey's predictions to obtain the velocities along the contact lines of the moving droplets. The contact lines are then evolved using a method of lines approach by integrating the velocities along prescribed points on the initial contact lines. This results in a highly accurate and computationally efficient simulation method.

This work builds upon and extends the work of [11] by developing an inverse design optimization framework. The aforementioned hybrid model is employed to design the substrate heterogeneity pattern and facilitate controllable droplet transport, as dictated by spatial variations of the local contact angle that force the droplet to move from hydrophobic areas to more hydrophilic.

2. AI-assisted low-fidelity model

In the limit of small contact angles, strong surface tension effects and negligible inertia, the long-wave approximation of the Navier–Stokes equations can be invoked to obtain a fourth order parabolic, degenerate PDE that describes the evolution of the droplet thickness $h(\mathbf{x}, t)$ as it moves along a chemically heterogeneous substrate. In non-dimensional form the equation becomes

$$\partial_t h + \boldsymbol{\nabla} \cdot \left[h \left(h^2 + \lambda^2 \right) \boldsymbol{\nabla} \nabla^2 h \right] = 0 \tag{1}$$

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02494 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
p.y.vrionis@cyi.ac.cy (Panayiotis-Y. Vrionis); a.demou@cyi.ac.cy (A. Demou); savva.nikos@ucy.ac.cy (N. Savva)
ORCID(s): 0000-0003-0917-7884 (Panayiotis-Y. Vrionis); 0000-0002-9510-0682 (A. Demou); 0000-0003-1549-3154 (N. Savva)

where λ is the slip length, used to circumvent the stress singularity that would appear at the moving contact line due to the no-slip condition[13]. By invoking Lacey's asymptotic model [12], one obtains

$$\tilde{u}_n = \frac{\phi_\star^3(\mathbf{c}) - \phi^3}{3|\ln\lambda|} \tag{2}$$

where $\tilde{u}_n$ corresponds to the leading contributions of the asymptotic expansion to the contact line normal velocity, $\mathbf{c}$ refers to the contact line position, ϕ refers the apparent contact angle of the droplet and $\phi_\star = \Phi(\mathbf{c})$ with $\Phi(\mathbf{x})$ being the prescribed substrate local contact angle profile.

The apparent contact angle ϕ can be obtained by solving the quasi-equilibrium problem that arises from Eq. (1) using a boundary integral method following $d\mathbf{c}/dt \cdot \mathbf{n} = \tilde{u}_n$. This method yields a highly efficient numerical method that tracks the evolution of the droplet contact line (see [9, 14] for more details).

Equation (2), only contains the main contact line dynamics, as it corresponds to only the first term in the asymptotic expansion, i.e. the most dominant contribution. Incorporating additional terms analytically is highly non-trivial as shown in [15, 9, 16], especially when considering 3D settings. Thus, to improve upon the accuracy of Eq. (2), and incorporate the additional dynamics of higher order corrections, Layce's model is extended with an additional term $\mathcal{H}\left(\mathbf{c}, \tilde{u}_n\right)$, the form of which *should* correspond to Eq. (4.42) in [9], to model these corrections via a data-driven approach. This extension results in

$$u_n = \tilde{u}_n + \frac{\mathcal{H}\left(\mathbf{c}, \tilde{u}_n\right)}{|\ln\lambda|} \tag{3}$$

The higher order corrections are trained using samples computed by solving multiple solutions of Eq. (1) over a diverse set of heterogeneity patterns $\Phi(\mathbf{x})$. The data-driven method employs *Fourier neural operators* [17] that utilize the frequency domain as a learning space and have been found to be highly efficient when considering complex phenomena governed by PDEs. Fig. 1 depicts a comparison between the low-fidelity (green curve), the AI-assisted (orange curve) and the reference solution (blue area), indicating the significant accuracy gained when using the hybrid approach in two significantly different heterogeneity patterns.

3. Inverse design optimization framework

The inverse design optimization framework is formulated by defining a cost function J as the area A encompassed by the trajectory of the desired (target) and the current droplet centroid path, as shown in Fig. 2. The trajectory of the droplet is changed by manipulating the chemical heterogeneity of the substrate, making the optimization problem

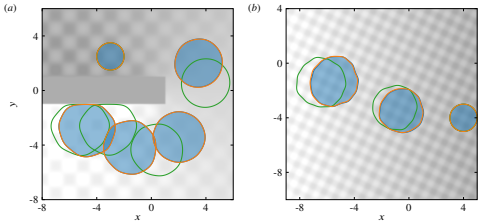


Figure 1: Contact line comparison between the low-fidelity (Lacey's model Eq. (2), green), the AI-assisted (Eq, (3), orange), and reference solutions for two examples of droplet transport. The heterogeneity variations follow shades of gray with white color denoting hydrophilic and darker colors hydrophobic regions, thus directing the droplet towards lighter shades of gray.

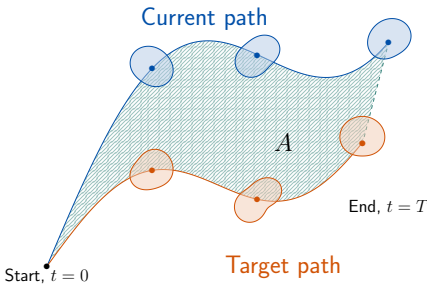


Figure 2: Illustration of the optimization problem cost function A, as defined by its deviation from the target droplet path.

controlled by the design variables $\mathbf{b}$ of size N the can alter the substrate heterogeneity pattern $\Phi(\mathbf{x})$ via the closed-form of function $F(\mathbf{x}, \mathbf{b})$.

To perform the gradient-based optimization, it is first necessary to compute $\partial_{\mathbf{b}} J$. These are computed via automatic differentiation within the JAX [18], and then a gradient descent algorithm is employed to update the design variables. The example shown in Fig. 3 demonstrates a case of inverse design in which an initially homogeneous substrate has its heterogeneity pattern changed to *turn* the droplet and reduce the cost function. On the left, the case used to formulate the target trajectory is shown.

4. Conclusions

This study demonstrates the potential of a hybrid modeling approach to improve the accuracy and reduce the computational costs associated with simulations of droplet dynamics on heterogeneous substrates. By integrating Lacey's

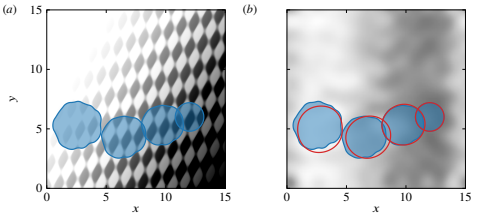


Figure 3: Example of chemical heterogeneity patterning in an effort to minimize the initial deviation of the target path.

asymptotic model with a data-driven model, significant enhancements have been achieved in the prediction of droplet behavior, particularly in terms of contact line velocities. This also facilitates its use as the driver in an inverse design optimization algorithm that is low-cost, and can assist in the design of substrate patterning.

References

[1] A. Lee, M.-W. Moon, H. Lim, W.-D. Kim, H.-Y. Kim, Water harvest via dewing, Langmuir 28 (27) (2012) 10183–10191. doi:10.1021/la3013987.

[2] T. Moragues, D. Arguijo, T. Beneyton, C. Modavi, K. Simutis, A. R. Abate, J.-C. Baret, A. J. deMello, D. Densmore, A. D. Griffiths, Droplet-based microfluidics, Nature Reviews Methods Primers 3 (1) (2023) 32. doi:10.1038/s43586-023-00212-3.

[3] Y. Zhang, Z. Dong, C. Li, H. Du, N. X. Fang, L. Wu, Y. Song, Continuous 3d printing from one single droplet, Nature communications 11 (1) (2020) 4685. doi:10.1038/s41467-020-18518-1.

[4] D. Bonn, J. Eggers, J. Indekeu, J. Meunier, E. Rolley, Wetting and spreading, Reviews of Modern Physics 81 (2009) 739–805. doi:10.1103/RevModPhys.81.739.

[5] I. Seric, S. Afkhami, L. Kondic, Direct numerical simulation of variable surface tension flows using a volume-of-fluid method, Journal of Computational Physics 352 (2018) 615–636. doi:10.1016/j.jcp.2017.10.008.

[6] S. Afkhami, J. Buongiorno, A. Guion, S. Popinet, Y. Saade, R. Scardovelli, S. Zaleski, Transition in a numerical model of contact line dynamics and forced dewetting, Journal of Computational Physics 374 (2018) 1061–1093. doi:10.1016/j.jcp.2018.06.078.

[7] G. G. Wells, É. Ruiz-Gutiérrez, Y. Le Lirzin, A. Nourry, B. V. Orme, M. Pradas, R. Ledesma-Aguilar, Snap evaporation of droplets on smooth topographies, Nature Communications 9 (1) (2018) 1380. doi:10.1038/s41467-018-03840-6.

[8] A. Oron, S. H. Davis, S. G. Bankoff, Long-scale evolution of thin liquid films, Reviews of Modern Physics 69 (1997) 931–980. doi:10.1103/RevModPhys.69.931.

[9] N. Savva, D. Groves, S. Kalliadasis, Droplet dynamics on chemically heterogeneous substrates, Journal of Fluid Mechanics 859 (2019) 321–361. doi:10.1017/jfm.2018.758.

[10] M.-A. Y.-H. Lam, L. J. Cummings, L. Kondic, Computing dynamics of thin films via large scale gpu-based simulations, Journal of Computational Physics: X 2 (2019) 100001. doi:10.1016/j.jcpx.2018.100001.

[11] A. D. Demou, N. Savva, Ai-assisted modeling of capillary-driven droplet dynamics, Data-Centric Engineering 4 (2023) e24. doi:10.1017/dce.2023.19.

[12] A. Lacey, The motion with slip of a thin viscous droplet over a solid surface, Studies in applied mathematics 67 (3) (1982) 217–230. doi:10.1002/sapm1982673217.

[13] C. Huh, L. E. Scriven, Hydrodynamic model of steady movement of a solid/liquid/fluid contact line, Journal of colloid and interface science 35 (1) (1971) 85–101. doi:10.1016/0021-9797(71)90188-3.

[14] N. Savva, S. Kalliadasis, Droplet motion on inclined heterogeneous substrates, Journal of Fluid Mechanics 725 (2013) 462–491. doi:10.1017/jfm.2013.201.

[15] R. Vellingiri, N. Savva, S. Kalliadasis, Droplet spreading on chemically heterogeneous substrates, Phys. Rev. E 84 (2011) 036305. doi:10.1103/PhysRevE.84.036305.

[16] N. Savva, D. Groves, Droplet motion on chemically heterogeneous substrates with mass transfer. ii. three-dimensional dynamics, Physical Review Fluids 6 (12) (2021) 123602. doi:10.1103/PhysRevFluids.6.123602.

[17] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020). doi:10.48550/arXiv.2010.08895.

[18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).
URL https://github.com/jax-ml/jax

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Data-Driven CFD-Based Design Optimization of Flow Pattern in a Gravitational Mixer Settler

Zinedine Khatir[a,*]

[a] University of Leeds, School of Mechanical Engineering, Woodhouse Lane, Leeds LS2 9JT, UK

ARTICLE INFO[†]

Keywords:
Liquid-Liquid Extraction;
Computational Fluid Dynamics;
Optimization;
Data-Driven Applications

ABSTRACT

Gravitational mixer settlers (GMXSs) are widely used for liquid-liquid extraction (LLE). Immiscible fluids are mixed to promote the transfer of compounds, then separated by gravity in a settling chamber. Micromechanical fluid interactions decisive to the separation process are complex, however studies have shown that by optimizing settler flow pattern, separation performance can be significantly improved. In this paper, an optimization framework for GMXSs designs is investigated which uses experimentally validated single phase Computational Fluid Dynamics (CFD) and residence time distribution (RTD) analyses to identify optimal combinations of design features which maximize desirable characteristics such as resident time and pressure drop. The design of the settler is formulated in terms of two design variables: flow rate and position of the inlet baffle. A Radial Basis Function (RBF)-based surrogate modeling approach using a Design of Experiment (DOE) technique and a permutation genetic algorithm was used to establish optimal process parameters. A Pareto front is built which enables designers to explore appropriate compromises between designs with small residence time and those with small pressure drop.

1. Introduction

Gravitational mixer settlers are used in nuclear [1], chemical, pharmaceutical, and hydrometallurgical industries for liquid-liquid extraction [2, 3]. Processes that define their operation, particularly through the settling chamber, are not fully understood and large equipment means pilot studies are time consuming and costly. CFD provides an alternative method of investigating settler's performance, where gravitational settler analysis has focused on inlet geometric and picket fence configuration by means of multiphase flow analysis [4, 5, 6]. Alternatively, success has been found using single phase simulation to improve settler performance through the flow pattern, and without modeling multiphase coalescence mechanisms [7, 8]. Coalescence is important for settler separation and can be encouraged by minimizing viscous shearing; droplets in the dispersed phase remain in contact for a maximum amount of time, encouraging film drainage and coalescence [9, 10]. Furthermore, for a given system, there exists an optimum residence time over which the separation will satisfactorily occur. It is thus desirable that a maximum proportion of the dispersion flow is in the settler for this time. Lane et al. [8] indicate that plug flow is the ideal regime, but this is practically difficult to achieve, and flow patterns significantly deviate from this ideal, ensuing reduced efficiency. It has been shown in [8, 11] that residence time distribution analysis is adequate to assess settler flow pattern quality.

In the present work, a data-driven CFD-based design-optimization framework is developed with combines surrogate modeling as in [12] and residence time analysis to investigate flow patterns in a GMXS.

2. Materials and method

Figure 1a and Fig. 1b show the geometry of the settler model as depicted by Panda et al. [5].

2.1. Single-phase simulations

The incompressible Navier-Stokes equations for single phase are assumed:

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u u) - \mu \nabla^2 u = \nabla p, \tag{1}$$

where u denotes fluid velocity, ρ the density, μ viscosity respectively and p its pressure are solved using the `simpleFoam` solver in OpenFOAM subject to the boundary conditions shown in Fig. 1a. These are solved using second order interpolation, an orthogonal hexahedral mesh and under-relaxation factors for velocity and pressure of 0.3 and 0.7 respectively as these are found to provide the best convergence performance. In total, 0.15M elements, as exemplified in Fig. 1b, are used together with automatic parallelization with 8 subdomains minimizing the number of processor boundaries.

[†] This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02495 and of the Proceedings 10.34734/FZJ-2025-02175.

[*] Corresponding author
Z.Khatir@leeds.ac.uk (Z. Khatir)
ORCID(s): 0000-0002-7559-7644 (Z. Khatir)

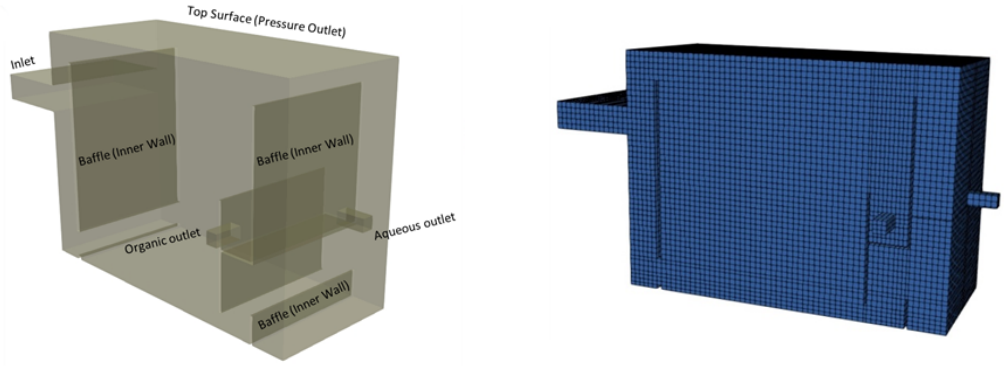


Figure 1: Schematic of the mixer settler together with: (a) Boundary conditions and (b) example of mesh distribution.

2.2. Residence Time Distribution (RTD)

The cumulative RTD, $F(t)$, is the proportion of the flow with a residence time, $E(t)$, less than or equal to time, t. This is calculated by introducing passive scalars at the inlet and calculating the time taken for them to leave the GMXS at the aqueous outlet. The `scalarTransportFoam` solver then solves the convection-diffusion equation for the passive scalar, c, namely

$$\frac{\partial(\rho c)}{\partial t} + \nabla \cdot (uc) - \nabla^2(D_T c) = 0, \tag{2}$$

where the steady, single-phase flow field u is obtained at steady state solution of Eq. (2).

Once $F(t)$ is determined, the residence time distribution $E(t)$, the mean residence time t_m, and the standard deviation σ_t are given via

$$E(t) = \frac{d(F(t)}{dt}, \tag{3}$$

$$t_m = \int_0^\infty tE(t)dt, \tag{4}$$

$$\sigma_t = \sqrt{\int_0^\infty (t - t_m)^2 E(t)dt} \quad . \tag{5}$$

The quantity σ_t is useful since a smaller value indicates that the flow is closer to the ideal flow scenario.

2.3. CFD-based optimization strategy

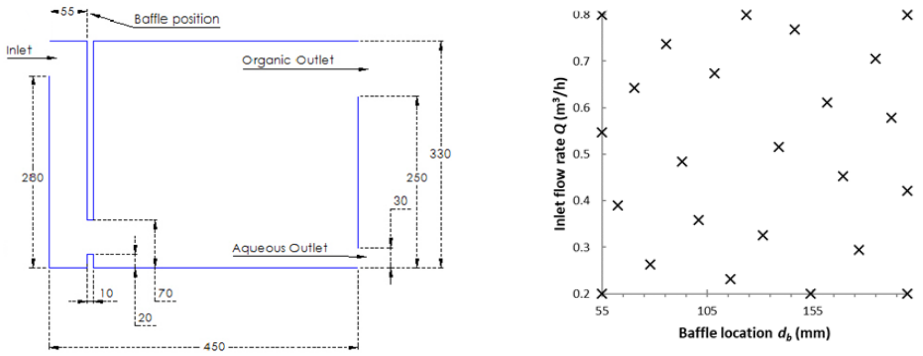
In this section, we consider the optimization of the GMXS system, subject to the conflicting objectives of minimizing both the energy loss, i.e., $E^* = \Delta p \cdot Q \cdot \rho$ with pressure drop Δp, density ρ and flow rate Q, and the RTD σ_t. Two design variables are used, namely: the flow rate Q, and baffle position d_b in the ranges of $0.2\text{m}^3/\text{h} \leq Q \leq 0.8\text{m}^3/\text{h}$ and $55\text{mm} \leq d_b \leq 200\text{mm}$ as indicated in Fig. 2a.

The goal is to generate a Pareto front of non-dominated solutions, from which an appropriate compromise design can be reached. The Pareto front is obtained by building accurate metamodels of both E^* and σ_t, as a function of the two design variables. The metamodels are constructed using values of the E^* and σ_t from numerical simulations carried out at twenty-four Design of Experiments (DOE) points. These points are obtained using Optimal Latin Hypercubes (OLH), by means of a permutation genetic algorithm using the Audze-Eglais potential energy criterion to ensure an efficient distribution of DOE points. The points are laid out as uniformly as possible using criteria of minimizing potential energy of repulsive forces which are inverse square functions of the separation of DOE points [12], i.e.,

$$\min E^{AE} = \min \sum_{i=1}^{N} \frac{1}{L_{ij}^2}, \tag{6}$$

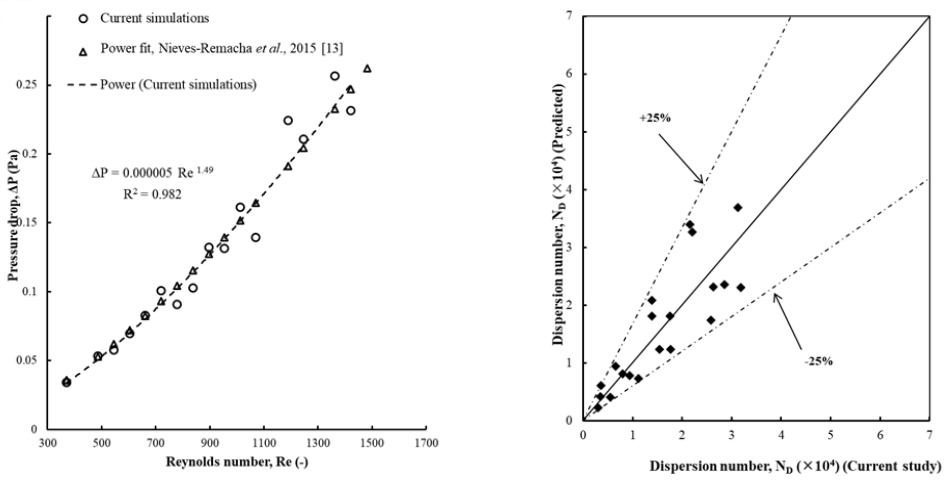
where L_{ij} is the Euclidean distance between points i and j $(i \neq j)$ and, $N = 24$ is the number of DOE points. Figure 2b reveals the uniform distribution of the DOE points within the design space as a combination of the design variables Q and d_b.

A Radial Basis Function (RBF) method is used to build the metamodels for E^* and σ_t, throughout the design space where a cubic radial power function is used to determine the weighting (wg) of points in the regression analysis at each point [12], $wg_i = r_i^3$. The parameter r_i is the normalized distance between the surrogate model prediction location from



(a) (b)

Figure 2: (a) Two-dimensional plane view indicating the two design variables: Inlet flow rate and baffle position. (b) DOE points distribution.



(a) (b)

Figure 3: (a) Pressure drop in the GMXS at different REYNOLDS numbers. The nonlinear behavior is due to the presence of obstacles in the system. (b) Parity plot of the dispersion number N_D.

the i^{th} sampling point. The Pareto front is calculated using a multi-objective genetic algorithm (MOGA) approach as in [12]. Points on the Pareto front are non-dominated in the sense that it is not possible to decrease any of the objective functions (i.e., E^* and σ_t) without increasing the other objective function. Hence, this provides designers the opportunity to select the most convenient compromise point among the optimum designs. In the next section, results of the data-driven CFD-based design optimization are discussed.

3. Results and discussion

3.1. Validation of the method

Simulations are performed with corresponding RTD flow analysis. Results are firstly validated against the work by Nieves-Remacha [13], Fig. 3a, with the pressure drop found to follow a similar power law $\Delta P \approx Re^{1.49}$ where the nonlinear effect is due to the presence of baffles in the settler and Re is the REYNOLDS number.

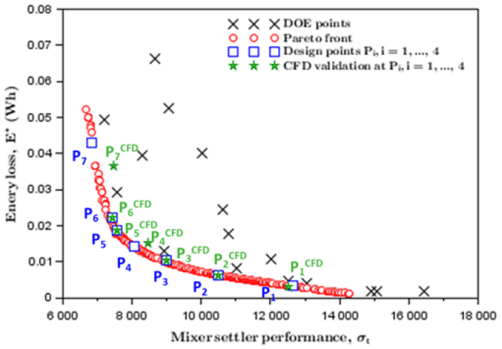


Figure 4: Pareto front showing the compromises that can be obtained in minimizing both σ_t and E^*.

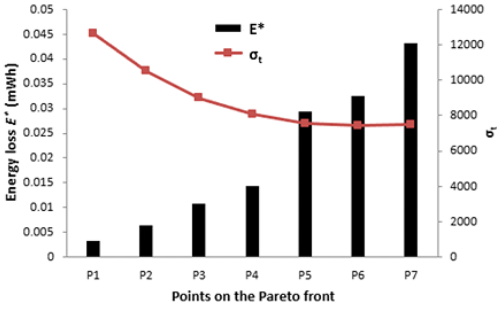
The effectiveness of separation is expressed in terms of following dimensionless number dispersion number (N_D) as noted by Manavalan et al. [14] such as

$$N_D = \frac{1}{t_m}\sqrt{\frac{H}{g}}, \tag{7}$$

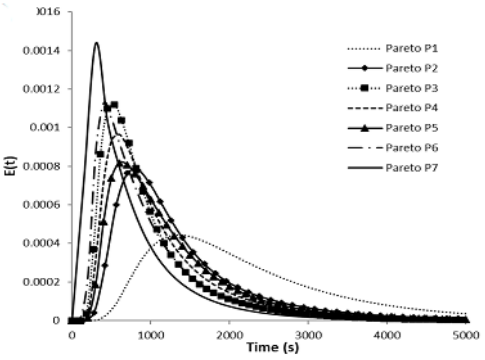
where t_m is the mean residence time as defined in the RTD analysis, g the acceleration due to gravity and H is the dispersion-band thickness using the correlation developed by Panda & Buwa [6]. A parity plot of the dispersion number N_D between the prediction by Jeelani and Hartland [15] and the current work is presented in Fig. 3b.

4. Optimization

The Pareto front curve in Fig. 4 represents the results in terms of both σ_t and E^*. The data reveal that any decrease of E^* or σ_t is followed by an increase of the other objective function. A very good agreement between the metamodel and full numerical calculations occurs demonstrating the accuracy of the metamodeling approach implemented. Results reveal that at point P_4, identified to be the best compromise design, a percentage error for around 5% and 6% for energy E^* and σ_t between the CFD predictions and metamodels suggesting the appropriateness of the data-driven model to predict accurately E^* and σ_t. Further analysis, as depicted in Fig. 5a, indicates point P_4 to provide $E^* = 0.0142$mWh and $\sigma_t = 8062$. The Pareto and E-curves in Fig. 5b might indicate point P3 to be also a good design, however this corresponds to an increase of 11.4% and 25.4% in E^* and σ_t respectively, compared to P_4. Though the E-curves, Fig. 5b, recommend point P_7 to be a good design, an increase of 203% in E^* is observed compared to P_4 then again.



(a)



(b)

Figure 5: (a) Energy loss E^* and σ_t and, and (b) RTD $E(t)$ at seven points on the Pareto front in Fig. 4.

Design P_4 corresponds to a flow rate $Q = 472\text{m}^3/s$ and a baffle position $d_b = 139$mm.

5. Conclusions

A data-driven CFD-based optimization methodology has been successfully developed for the design of efficient GMXS liquid-liquid extraction systems. Current work entails a multi-fidelity approach which combines multiphase flow analysis.

Data statement

All data underlying the results are available as part of the article and no additional source data are required in the Research Data Leeds Repository[1].

[1] https://doi.org/10.5518/1578

References

[1] D. B. Sharma, S. A. Ansari, R. B. Gujar, P. K. Mohapatra, Demonstration of Mixer-Settlers Runs for Separation of Uranium and Neptunium by TBP Vis-a-Vis DHOA Under a Legacy Waste Condition, Solvent Extraction and Ion Exchange 42 (3) (2024) 264–280. doi:10.1080/07366299.2024.2360504.

[2] D. Hadjiev, J. Paulo, Extraction separation in mixer–settlers based on phase inversion, Separation and Purification Technology 43 (3) (2005) 257–262. doi:10.1016/j.seppur.2004.11.006.

[3] J. M. Martins, A. S. Guimarães, A. J. B. Dutra, M. B. Mansur, Hydrometallurgical separation of zinc and copper from waste brass ashes using solvent extraction with D2EHPA, Journal of Materials Research and Technology 9 (2) (2020) 2319–2330. doi:10.1016/j.jmrt.2019.12.063.

[4] S. Ye, Q. Tang, Y. Wang, W. Fei, Structural optimization of a settler via CFD simulation in a mixer-settler, Chinese Journal of Chemical Engineering 28 (4) (2020) 995–1015. doi:10.1016/j.cjche.2020.01.010.

[5] S. K. Panda, K. Singh, K. Shenoy, V. V. Buwa, Numerical simulations of liquid-liquid flow in a continuous gravity settler using OpenFOAM and experimental verification, Chemical Engineering Journal 310 (2017) 120–133. doi:10.1016/j.cej.2016.10.102.

[6] S. K. Panda, V. V. Buwa, Effects of Geometry and Internals of a Continuous Gravity Settler on Liquid–Liquid Separation, Industrial & Engineering Chemistry Research 56 (46) (2017) 13929–13944. doi:10.1021/acs.iecr.7b03756.

[7] D. Stanbridge, J. Sullivan, One example of how offshore oil & gas industry technology can be of benefit to hydrometallurgy., in: Second International Conference on CFD in Minerals and Process Industries CSIRO, Melbourne, Australia, 1999, pp. 317–321.
URL https://www.cfd.com.au/cfd_conf99/Conf99_Papers/067STAN.PDF

[8] G. L. Lane, K. Mohanarangam, W. Yang, D. J. Robinson, K. R. Barnard, Flow pattern assessment and design optimisation for an industrial solvent extraction settler through in situ measurements and CFD modelling, Chemical Engineering Research and Design 109 (2016) 200–214. doi:10.1016/j.cherd.2016.01.021.

[9] J. Kamp, J. Villwock, M. Kraume, Drop coalescence in technical liquid/liquid applications: a review on experimental techniques and modeling approaches, Reviews in Chemical Engineering 33 (1) (2017) 1–47. doi:10.1515/revce-2015-0071.

[10] S.-l. Yan, X.-q. Wang, L.-t. Zhu, X.-b. Zhang, Z.-h. Luo, Mechanisms and modeling of bubble dynamic behaviors and mass transfer under gravity: A review, Chemical Engineering Science 277 (2023) 118854. doi:10.1016/j.ces.2023.118854.

[11] G. Miller, Design of mixer-settlers to maximise performance, in: Design of mixer-settlers to maximise performance, ALTA Metallurgical Services, Melbourne, Australia, 2006.

[12] M. Al-Damook, Z. Khatir, M. Al Qubeissi, D. Dixon-Hardy, P. J. Heggs, Energy efficient double-pass photovoltaic/thermal air systems using a computational fluid dynamics multi-objective optimisation framework, Applied Thermal Engineering 194 (2021) 117010. doi:10.1016/j.applthermaleng.2021.117010.

[13] M. J. Nieves-Remacha, A. A. Kulkarni, K. F. Jensen, OpenFOAM Computational Fluid Dynamic Simulations of Single-Phase Flows in an Advanced-Flow Reactor, Industrial & Engineering Chemistry Research 54 (30) (2015) 7543–7553. doi:10.1021/acs.iecr.5b00232.

[14] B. Manavalan, T. V. Tamhane, J. Patra, A. J. Joshi, J. B. Joshi, N. K. Pandey, S. Kumar, K. U. Mudali, N. Rajamani, V. S. Vitankar, R. N. Patil, Separation Characteristics of Liquid– Liquid Dispersions: Gravity and Centrifugal Settlers, Industrial & Engineering Chemistry Research 56 (27) (2017) 7814–7823. doi:10.1021/acs.iecr.7b00119.

[15] S. A. K. Jeelani, S. Hartland, Prediction of steady state dispersion height from batch settling data, AIChE Journal 31 (5) (1985) 711–720. doi:10.1002/aic.690310503.

Mini-Symposium 9:

Computational Fluid Dynamics with High-Order Spectral Element Methods on GPUs

Organizers: Mathis Bode, Jörg Schumacher, Roshan J. Samuel, Christian Hasse, Hendrik Nicolai, Christos E. Frouzakis, and Ananias Tomboulides

Turbulent fluid flows have been important use cases for high performance computing (HPC) platforms since the first spectral simulations of the Navier-Stokes equations by Orszag and Patterson in the late 1960's. Characterized by exponential convergence that provides high accuracy at lower computational cost, spectral-type numerical schemes are well suited for the efficient simulation of turbulence, where the number of grid points grows faster than quadratic with the Reynolds number when all flow features need to be resolved.

Spectral element methods (SEMs) combine the high accuracy with flexibility in terms of flow geometry. A high-order SEM approximates the solution and data in terms of locally structured Nth-order tensor-product polynomials on a set of globally unstructured elements. Thus, in addition to exponential convergence for smooth solutions with increasing polynomial order, it offers flexibility to handle complex geometries via domain decomposition. For the same accuracy, matrix free SEM solvers also offer low storage and computational cost. In order to exploit the performance potential of existing and upcoming GPU-based exascale supercomputers, SEM solvers for CPU-based HPC systems have to either be ported to GPUs, or to be rewritten from scratch. The potential of SEM solvers for exascale computing has been underlined by the two 2023 Gordon Bell Award finalists with applications using nekRS and neko. Furthermore, the recently developed nekCRF reactive flow plugin showcases how SEMs can be efficiently used for computational fluid dynamics (CFD) including multi-physics effects, like combustion, on exascale supercomputers.

The mini symposium covers spectral element CFD solvers for GPUs. Submissions can include contributions to the development of numerical methods and/or physical models in the context of SEM as well as application examples of CFD using SEM on current GPU HPC systems. CFD can refer to fluid dynamics applications of flows with or without multi-physics effects, such as combustion, multiphase or magnetohydrodynamics.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Towards High-Fidelity Simulations of Urban Flows

Josep M. Duró[a,*], Ming Teng[b], Naïm Munoz[a], Ernest Mestres[a], Jordi Muela[b], Oriol Lehmkuhl[b] and Ivette Rodriguez[a]

[a]Universitat Politècnica de Catalunya, TUAREG, Turbulence and Aerodynamics Research Group, Colom 11, 08222 Barcelona, Spain
[b]Barcelona Supercomputing Center, Large-scale Computational Fluid Dynamics, Eusebi Güell 1-3, 08034 Barcelona, Spain

ARTICLE INFO†

Keywords:
Large-Eddy Simulation;
Urban Flow

ABSTRACT

This work aims at demonstrating the capabilities of using high-fidelity simulations for conducting the study of the turbulent flow in a full-scale urban geometry. This framework takes advantage of today's new generation of supercomputers using accelerated computing architectures, e.g. graphical processing units (GPUs). For the simulations we consider the flow under the influence of an atmospheric boundary layer and specifically we focus on neutral turbulent flows past an urban geometry. For validating the computational framework two different urban geometries are tested and the results compared against wind tunnel data with good agreement. Afterwards, the capabilities of the numerical methodology is showcased by performing simulations on a full-scale urban geometry corresponding with a neighborhood of the city of Barcelona. These results have shown a strong potential to significantly advance our current understanding of urban aerodynamics.

1. Introduction

Cities have long been regarded as centers of economic and demographic growth, resulting in a significant concentration of population in their cores. According to the United Nations, by 2030, two-thirds of the global population will reside in urban areas. Thus, the study and understanding of urban flows to improve forecasting and develop accurate prediction methods is at the focus of urban sustainability.

In this regard, the use of computational fluid dynamics (CFD), particularly high-fidelity simulations, constitutes a reliable tool for performing these studies. Indeed, CFD can provide a complete data field of different variables throughout the whole domain which render the possibility of performing a more thorough analysis of wind flow, pollutant dispersion, street canyon ventilation, among other studies. Urban climate studies can be conducted at different scales, including the meteorological mesoscale, the meteorological microscale (up to about $2km$), the building scale (up to a few hundred meters), and the indoor environment ($\approx 10m$).

Concerning urban microclimate studies, a large number of these studies have been conducted in generic or scaled-down geometries from two-dimensional (2D) street canyons or 3D generic geometries in order to study fundamental fluid dynamics related with urban flows using both Reynolds-Averaged Navier-Stokes (RANS) equations, see, e.g., [1, 2, 3], or using large-eddy simulations (LES) in scaled down geometries, see, e.g., [4, 5, 6].

Only a few studies reported in the literature employ LES in complex urban environments, e.g. [7, 8], highlighting the significance of high-fidelity simulations for understanding the complex turbulent interactions between buildings and the atmospheric boundary layer (ABL). Within the framework of the present work, we aim at characterizing the turbulent flow in an urban environment by performing LES in order to identify the different flow structures and regimes present and their complex interaction, as well as the turbulent statistics of the flow. In order to achieve this ultimate goal, high-fidelity simulations of neutral ABLs over full-scale realistic urban geometries are performed, the results of which are validated against measurements from wind tunnel experiments.

2. Mathematical and Numerical Model

The isothermal and incompressible flow in an urban environment is governed by the conservation of mass and momentum equations. By spatially filtering these equations, the LES formulation can be obtained by

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \quad (1)$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} - \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} + \rho^{-1} \frac{\partial \overline{p}}{\partial x_i} = -\frac{\partial \mathcal{T}_{ij}}{\partial x_j}, \quad (2)$$

where $(\bar{\cdot})$ represents the filtered variables. In the above equations, x_i are the spatial coordinates (or x, y, and z), u_i (or u, v, and w) stands for the velocity components, and p

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02496 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
josep.maria.duro@upc.edu (J.M. Duró); oriol.lehmkuhl@bsc.es (O. Lehmkuhl); ivette.rodriguez@upc.edu (I. Rodriguez)
ORCID(s): 0000-0002-1892-9104 (J.M. Duró); 0009-0008-5429-4207 (M. Teng); - (N. Munoz); - (E. Mestres); 0000-0003-1583-8490 (J. Muela); 0000-0002-2670-1871 (O. Lehmkuhl); 0000-0002-3749-277X (I. Rodriguez)

is the pressure. ν and ρ are the kinematic viscosity and the density of the fluid, respectively. The right-hand-side term in Eq. (2) defines the subgrid stresses modeled, where

$$\mathcal{T}ij - \frac{1}{3}\mathcal{T}kk\delta_{ij} = -2\nu_{sgs}\overline{S}_{ij} \tag{3}$$

To close the above formulation, an expression for the subgrid-scale viscosity, ν_{sgs} has to be introduced. In the present work, the model proposed by Vreman [9] is used. The filtered incompressible Navier-Stokes equations have been numerically solved using SOD2D (Spectral high-Order coDe 2 solve partial Differential Equations) [10], a low-dissipation spectral element method (SEM) code. SOD2D has been developed with the aim of efficiently leveraging the computational power that is being deployed worldwide in which GPU play a central role. For that reason, it has been developed to be able to run on either GPU or CPU architectures. Specifically, the code is written in Fortran, and uses MPI and OpenACC to provide parallelism at both coarse and fine-grained levels. The mesh is partitioned using the GEMPA library, and it uses HDF5 for I/O which are efficient and widely tested libraries for HPC. SOD2D is based on a spectral-element version of Galerkin's finite element-method continuous model with a modified version of Guermond's entropy viscosity stabilization [11]. The principle behind this methodology is that a set of diffusive terms are added to all equations in the Navier-Stokes system, with the related viscosity being solution-dependent. This viscosity depends on the entropy field and is computed per node. With the current LES implementation, there is no interaction between the stabilization and sub-grid models. Additional details can be found in Gasparino et al. [10]. In SOD2D, the aliasing effects of the reduced order integration caused by employing SEM integration for convective terms are countered with the skew-symmetric splitting presented by Kennedy and Gruber [12]. For the temporal discretization, a BDF-EXT3 high-order operator splitting approach is used to solve the velocity-pressure coupling [13].

3. Results

The main objectives of the present work are to implement high-fidelity simulations of neutral ABLs over full-scale realistic urban geometries, and validate the current framework for simulating the turbulent flow in a neutral ABL. For validation purposes, two different cases of increasing complexity will be considered and after that, the flow around an urban zone in Barcelona city will be considered.

The first case studied is the wind tunnel experiment conducted by Brown et al. [14], which also serves as validation of the numerical framework. The experiment considered a neutral ABL approaching a matrix of 7×7 cubes, each with dimensions of $L = W = H = 0.15m$ (L, W, H being length, width and height, respectively). To simulate the ABL in the wind tunnel experiments spires and floor roughness elements were used to obtain an equivalent ABL with reference velocity of $3m/s$ measured at building height H and friction velocity $u^* = 0.24m/s$. In the present simulations, full-scale building arrays of $H = 30m$ are simulated, corresponding to a 1:200 ratio compared to the experiments. For the domain dimensions, in agreement with the best practice guidelines for simulating urban flows [15], 3D blocks array were placed at a distance of 6.7H from the inlet and 10H from the lateral boundaries of the domain. A domain height of 10H is considered. As for the boundary conditions, the top and laterals of the computational domain are modeled as a free-slip wall, which assumes zero normal gradients for all the variables. At the outlet, zero static pressure is imposed. The actual REYNOLDS number (based up in building height, and free stream velocity, U_∞) for the case is $Re_H = 9 \cdot 10^6$.

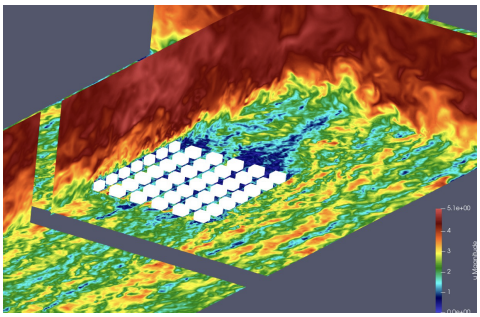


Figure 1: Instantaneous velocity field for the 7x7 full-scale city block case.

To impose a realistic turbulent ABL inlet condition, a precursor simulation is employed, from which the outlet plane is used to "drive" the inlet of the domain of interest. Three different $p = 4$ grid meshes are used in order to assess the influence of the numerical resolution on the accuracy of the results. For these meshes, in the zone of interest, i.e., in the 3D blocks, resolutions of $1.87m$, $1.25m$, and $0.75m$ for a total number of $22.8 \cdot 10^6$, $62.2 \cdot 10^6$, and $219.4 \cdot 10^6$ grid points are considered.

In Fig. 1, an overview of the turbulent flow around the city can be observed, while in Fig. 2 results obtained with the different levels of refinements are compared against the experimental data. In the figure, both streamwise velocity and turbulent kinetic energy (TKE) at different locations are plotted.

A very good agreement at all stations is observed for both streamwise velocity and TKE. To ensure temporal convergence, data is gathered over 18,000 seconds, covering

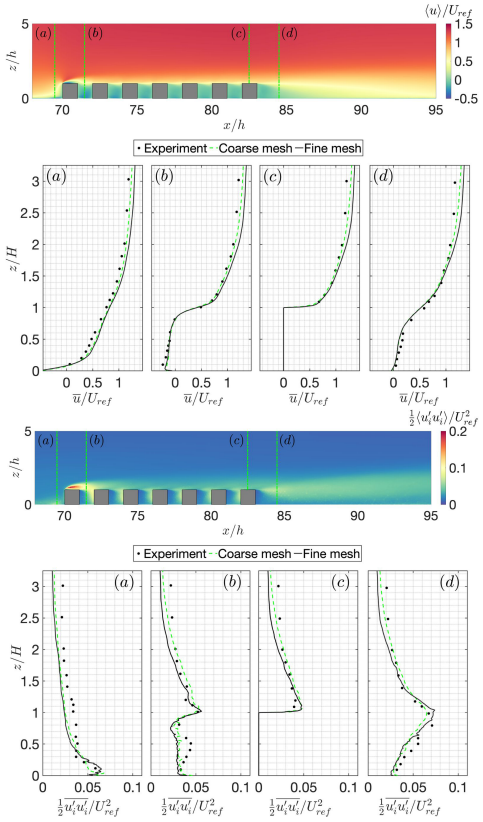


Figure 2: Vertical profiles of the streamwise velocity (top) and turbulent kinetic energy (bottom) at different stations. Comparison between the different meshes and with experimental data from Brown et al. [14].

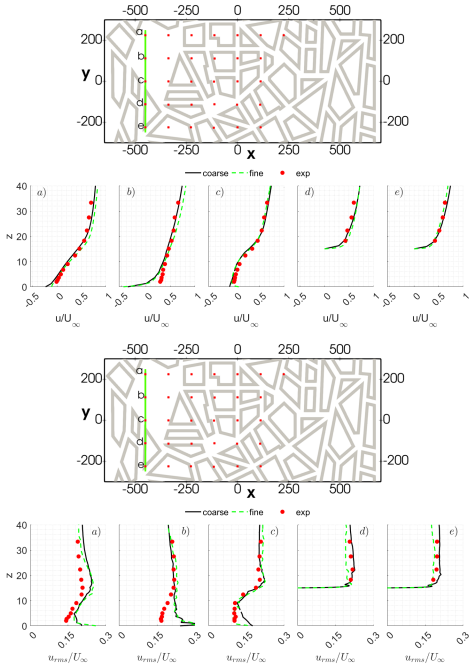


Figure 3: Vertical profiles of the streamwise velocity (top) and root-mean-square (rms) values of the fluctuations (bottom) at different stations. Comparison between the different meshes and with experimental data[2].

40 flow-throughs. The simulation has been run on MareNostrum V, utilizing 2 nodes from the accelerated partition, each equipped with 4 H100 GPUs. This setup achieves a performance of $0.1344 s/ite$, equivalent to $2.154 ns/dof$. For statistical data collection on the finest grid ($219.4 \cdot 10^6$ grid points), the 40 flow-throughs requires a wall time of 7.5 days. This efficient configuration led to a computational cost of 57,600 GPU-hours, highlighting its effectiveness for high-accuracy urban environment simulations.

The second case simulated is a semi-idealized city ("Michel-Stadt"; CEDVAL-LES database, case reference: BL3-3). The modeled city area measures 1,320m in length by 830m in width, with all buildings featuring flat roofs to simplify wind flow interactions. In this experiment, a streamwise reference wind velocity of approximately $6.1 m/s$ was maintained at a height of $100m$. The inflow boundary layer was designed to simulate a very rough flow environment, characterized by a surface roughness length (z_0) of $1.53m$ corresponding to a friction velocity (u^*) of $0.596 m/s$. As in the previous setup, three grid refinement levels - $22.6 \cdot 10^6$, $78.11 \cdot 10^6$, and $217.84 \cdot 10^6$grid points - are evaluated. These refinements correspond to resolutions of $3m$, $1.5m$, and $0.75m$ at pedestrian level. To compute this case, 20 flow-throughs are collected to gather statistical data. The finest grid requires a wall time of 4 days on 32 GPUs, amounting to a total computation time of 61,440 GPU hours. In Fig. 3, results obtained with the different levels of refinement in comparison against results from wind tunnel are presented. Both streamwise velocity and root-mean-square (rms) values of its fluctuations are pretty well predicted with the different meshes.

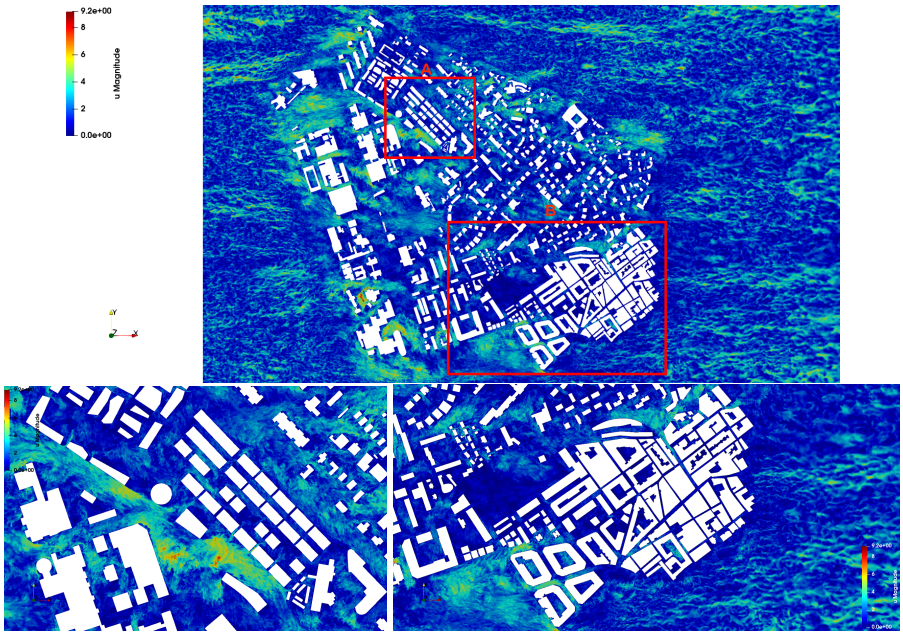


Figure 4: Instantaneous wind velocity map at pedestrian level. Top: Complete domain; Bottom left: Zoom of A; Bottom right: Zoom of B.

After validation of the current framework, the methodology has been applied to a $1700m$ long and $1900m$ wide urban neighborhood of the city of Barcelona. For the simulation, the city is located at a $2L$ distance from the inlet, $3L$ from the lateral boundaries of the domain, whereas the outlet is located at $4L$, L being 10 times the height of the highest building which in this case is $67m$. The inlet conditions have been set similar to that of Michelstadt case. The computational mesh used for solving this case is a 4th-order mesh, yielding a total of about $483 \cdot 10^6$ grid points with a resolution of $1.25m$ at ground level.

Results from the study are shown in Fig. 4 which showcases the capacity of the current methodology for running these simulations. At this stage, simulations with different wind directions are being conducted which will allow to analyze the influence of this in the flow configuration and characteristics in the different street canyons.

[2]Reference: Prof. Leitl and Dr. Frank Harms, University of Hamburg, Environmental Wind Tunnel Laboratory, https://www.mi.uni-hamburg.de/en/arbeitsgruppen/windkanallabor/data-sets.html

4. Conclusions

High-accuracy simulations of neutral atmospheric boundary-layers (ABL) over full-scale urban scenarios are conducted using a high-order numerical framework. In this approach, a precursor simulation is used. The precursor domain allows a fully developed inflow and generates stochastic fluctuations that adequately represent turbulence at inlet boundary conditions of the domain of interest.

To validate the framework's accuracy, two cases with available wind tunnel data are tested. The influence of grid refinement on the results is assessed by comparing simulation outputs directly with experimental data, showing good overall agreement. This supports the framework's suitability for urban Large Eddy Simulation (LES) studies. Results demonstrate accurate mean velocities and turbulence statistics across all grid resolutions, though a 1-meter resolution at pedestrian level yields better predictions for second-order flow statistics.

The methodology is also successfully applied to a real, full-scale urban scenario corresponding to a neighborhood of the city of Barcelona, yielding promising results. Future work will explore the effects of varying wind incidence angles to assess how wind direction impacts flow statistics

based on different street canyon configurations. Overall, this framework shows strong potential for advancing our understanding of urban aerodynamics, especially in characterizing and predicting unsteady flow behaviors.

Acknowledgements

This work has been partially financially supported by 'Agència de Gestió d'Ajuts Universitaris i de Recerca' under the call CLIMA 2023 (ref. 2023 CLIMA 00097) O. Lehmkuhl work is financed by a Ramón y Cajal postdoctoral contract by the Ministerio de Economía y Competitividad, Secretaría de Estado de Investigación, Desarrollo e Innovación, Spain (RYC2018-025949-I). The authors acknowledge the support of the Departament de Recerca i Universitats de la Generalitat de Catalunya through the research group Large-scale Computational Fluid Dynamics (ref.: 2021 SGR 00902) and the Turbulence and Aerodynamics Research Group (ref.: 2021 SGR 01051). The authors also acknowledge Red Española de Supercomputacion the resources provided in MareNostrum V Supercomputer (IM-2024-2-0006).

References

[1] W. Cheng, C.-H. Liu, D. Y. Leung, Computational formulation for the evaluation of street canyon ventilation and pollutant removal performance, Atmospheric Environment 42 (40) (2008) 9041–9051. doi:10.1016/j.atmosenv.2008.09.045.

[2] Z. Ai, C. Mak, Cfd simulation of flow in a long street canyon under a perpendicular wind direction: Evaluation of three computational settings, Building and Environment 114 (2017) 293–306. doi:10.1016/j.buildenv.2016.12.032.

[3] C.-H. Liu, C. C. Wong, On the pollutant removal, dispersion, and entrainment over two-dimensional idealized street canyons, Atmospheric Research 135–136 (2014) 128–142. doi:10.1016/j.atmosres.2013.08.006.

[4] Z.-T. Xie, I. P. Castro, Large-eddy simulation for flow and dispersion in urban streets, Atmospheric Environment 43 (13) (2009) 2174–2185. doi:10.1016/j.atmosenv.2009.01.016.

[5] P. Gousseau, B. Blocken, G. van Heijst, Large-eddy simulation of pollutant dispersion around a cubical building: Analysis of the turbulent mass transport mechanism by unsteady concentration and velocity statistics, Environmental Pollution 167 (2012) 47–57. doi:10.1016/j.envpol.2012.03.021.

[6] S. Hassan, U. H. Akter, P. Nag, M. M. Molla, A. Khan, M. F. Hasan, Large-eddy simulation of airflow and pollutant dispersion in a model street canyon intersection of dhaka city, Atmosphere 13 (7) (2022) 1028. doi:10.3390/atmos13071028.

[7] G.-J. Lee, D. Muñoz-Esparza, C. Yi, H. J. Choe, Application of the cell perturbation method to large-eddy simulations of a real urban area, Journal of Applied Meteorology and Climatology 58 (5) (2019) 1125–1139. doi:10.1175/jamc-d-18-0185.1.

[8] Z. Yan, R. Chen, X.-C. Cai, Large eddy simulation of the wind flow in a realistic full-scale urban community with a scalable parallel algorithm, Computer Physics Communications 270 (2022) 108170. doi:10.1016/j.cpc.2021.108170.

[9] A. W. Vreman, An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications, Physics of Fluids 16 (10) (2004) 3670–3681. doi:10.1063/1.1785131.

[10] L. Gasparino, F. Spiga, O. Lehmkuhl, Sod2d: A gpu-enabled spectral finite elements method for compressible scale-resolving simulations, Computer Physics Communications 297 (2024) 109067. doi:10.1016/j.cpc.2023.109067.

[11] J.-L. Guermond, R. Pasquetti, B. Popov, Entropy viscosity method for nonlinear conservation laws, Journal of Computational Physics 230 (11) (2011) 4248–4267. doi:10.1016/j.jcp.2010.11.043.

[12] C. A. Kennedy, A. Gruber, Reduced aliasing formulations of the convective terms within the navier–stokes equations for a compressible fluid, Journal of Computational Physics 227 (3) (2008) 1676–1700. doi:10.1016/j.jcp.2007.09.020.

[13] G. E. Karniadakis, M. Israeli, S. A. Orszag, High-order splitting methods for the incompressible navier-stokes equations, Journal of Computational Physics 97 (2) (1991) 414–443. doi:10.1016/0021-9991(91)90007-8.

[14] M. J. Brown, R. E. Lawson, Comparison of centerline velocity measurements obtained around 2d and 3d building arrays in a wind tunnel, Tech. rep., Los Alamos National Lab. (LANL), Los Alamos, NM (United States) (2001).

[15] J. Franke, A. Baklanov, Best practice guideline for the cfd simulation of flows in the urban environment: Cost action 732 quality assurance and improvement of microscale meteorological models, Tech. rep. (2007).

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Parallel Performance and Communication Pattern Analysis on SOD2D: A CFD High-Order Spectral Element Code

Jordi Muela[a,*], Lucas Gasparino[a] and Oriol Lehmkuhl[a]

[a]Barcelona Supercomputing Center, 1-3 Plaça d'Eusebi Güell, 08034, Barcelona, Spain

ARTICLE INFO[†]

Keywords:
Spectral Finite Element;
Parallel Methodology;
Parallel Applications;
Exascale;
Galerkin

ABSTRACT

This work presents the parallel performance of the CFD code SOD2D in different brand-new HPC platforms. SOD2D is a Continuous Galerkin High-Order Spectral Element Code designed to solve simulations of both turbulent compressible and incompressible flows. Furthermore, an exhaustive and comprehensive comparison of different communication patterns for the MPI Communications in the code is presented, showing the relevance of the CUDA-aware MPI library when running on GPUs.

1. Introduction

Nowadays, the trend in new High-Performance Computing (HPC) platforms is, in general, to achieve higher computing performances by installing larger and larger accelerated partitions based on General-Purpose Graphical Processing Units (GPGPUs) [1]. For example, in the HPC platforms deployed by the Oak Ridge Leadership Computing Facility (OLCF[1]), the percentage of peak performance provided by GPUs has increased up to above the 98% in their most new HPC cluster [2], Frontier, which is the actual first-ranked supercomputer in the TOP500 list[2].

Therefore, the new scientific computing codes must use GPUs efficiently to exploit all the potential *FLOPS* that the latest HPC platforms are installing. The codes must be able to leverage the fine-grained parallelism and large data throughput that GPUs offer, along with the coarse-grained parallelism based on the decomposition of the whole problem in different partitions, each one of them is solved by a different processing unit. This decomposition and distribution of the calculations in different processing units involves and requires data communication between them to solve the global problem. In general, this data communication in distributed processes is done using the Message Passing Interface (MPI), which is a standardized and portable message-passing standard designed to function on parallel computing architectures[3].

Therefore, the performance of a scientific parallel code running in GPUs is highly influenced by its capability of efficiently exploiting the GPU fine-grained parallelism within computational loops, as well as its communication performance, basically determined by the time spent between the processes exchanging data. In the present work, these aspects are analyzed and assessed in the CFD code SOD2D[4] [3], designed to run efficiently in heterogeneous HPC platforms and aiming to exploit all their computational performance.

2. Physical model

2.1. Mathematical formulation

SOD2D is a Computational Fluid Dynamics (CFD) code designed to solve simulations of both turbulent compressible and incompressible flows. Hence, the governing equations aimed to solve are the Navier-Stokes equations in an n-dimensional domain, which in the compressible form [4] reads as:

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) &= 0, \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p - \nabla \cdot \boldsymbol{\tau} &= \mathbf{f}, \\ \partial_t E + \nabla \cdot ((E + p)\,\mathbf{u}) - \nabla \cdot (\boldsymbol{\tau} \mathbf{u}) - \nabla \cdot (\kappa \nabla T) &= S, \end{aligned}$$

defined on $\Omega \times (t_0, t_f)$. In the above, the tensor $\boldsymbol{\tau} = \tau_{ij}$ introduces the viscous stresses into the model. For incompressible flows, the required equations can be derived straightforwardly assuming constant density $\partial_t \rho = 0$ and uniform viscosity, resulting in $\nabla \cdot \boldsymbol{\tau} = \mu \nabla^2 \mathbf{u}$.

[†] This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02497 and of the Proceedings 10.34734/FZJ-2025-02175.

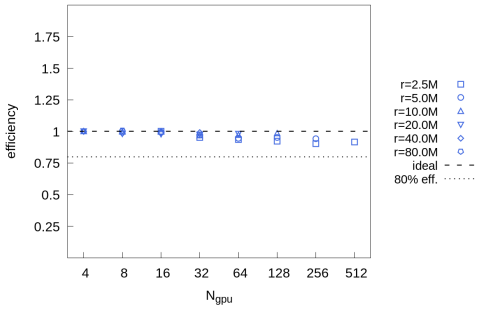
*Corresponding author
jordi.muela@bsc.es (J. Muela); lucas.gasparino@bsc.e (L. Gasparino); oriol.lehmkuhl@bsc.es (O. Lehmkuhl)
ORCID(s): 0000-0003-1583-8490 (J. Muela); 0000-0003-2024-3731 (L. Gasparino); 0000-0002-2670-1871 (O. Lehmkuhl)

[1] https://www.olcf.ornl.gov

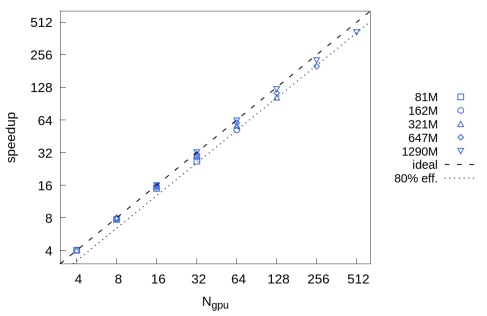
[2] https://www.top500.org/lists/top500/2024/06/

[3] https://hpc.nmsu.edu/discovery/mpi/introduction/

[4] https://gitlab.com/bsc_sod2d/sod2d_gitlab/

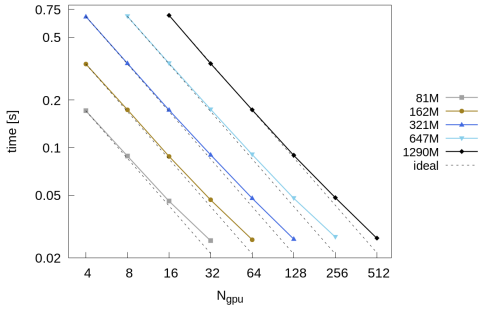


(a) Weak Speedup.

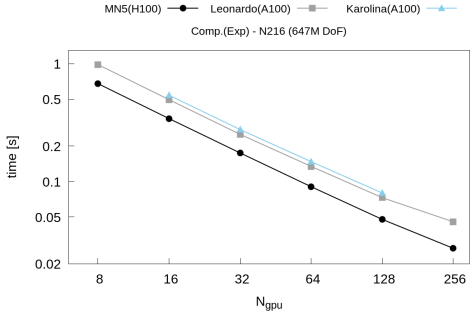


(b) Strong Speedup.

Figure 1: Speedup GPU (MN5).



(a) Mesh comparison (MN5).



(b) Platform comparison.

Figure 2: Iteration wall-clock time GPU.

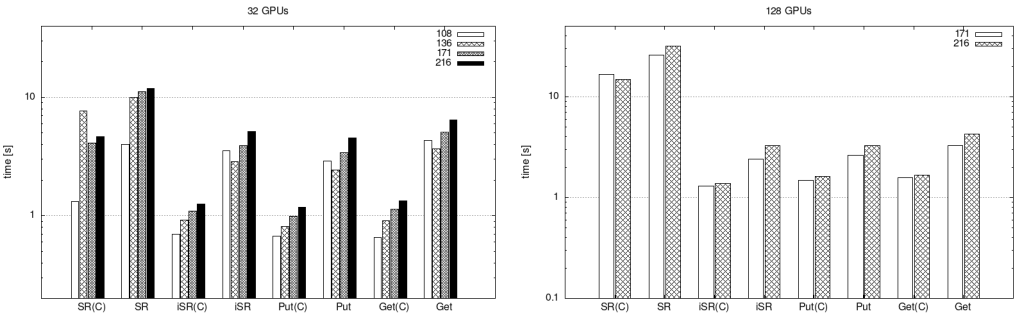
3. Numerical model

The set of Partial Differential Equations (PDEs) describing the physics of turbulent flows presented above can be converted into a solvable set of algebraic equations employing different numerical methods. In the present work the employed method is the spectral formulation of the Continuous Galerkin Finite Elements model. This model is selected since it does not require the reconstruction of fluxes between the elements, which could introduce numerical dissipation. Moreover, using spectral elements is advantageous when using hexahedra for discretizing the spatial domain Ω. The Lobatto-Gauss-Legendre (LGL) quadrature is used in the developed algorithm. The proposed scheme is stabilized by employing a modified version of the Entropy Viscosity model proposed by Guermond et al. [5]. Different integration schemes are implemented: a fully $4th$ order Runge-Kutta explicit scheme and an Implicit-Explicit Runge-Kutta (IMEX-RK) for the compressible solver, and the velocity-correction integration scheme BFD/EXT-3 proposed by Karniadakis et al. [6, 7] for the incompressible solver.

4. Scalability analysis

The parallel performance of the algorithm detailed in the previous sections has been thoroughly analyzed and assessed in different HPC platforms. Specifically, in the present work results for the following HPC platforms are presented: i) the MareNostrum 5 supercomputer, placed at the Barcelona Supercomputing Center (BSC) and equipped with GPUs NVIDIA H100 (Hopper)[5]; ii) the Leonardo HPC system, hosted by CINECA, based on NVIDIA A100 GPUs with all the nodes interconnected through an NVIDIA Mellanox network, with Dragon Fl+[6], iii) and the Karolina supercomputer, located at the IT4Innovations National Supercomputing Center at VSB, the Technical University of Ostrava[7], based as well on NVIDIA A100 GPUs.

[5] https://www.bsc.es/supportkc/docs/MareNostrum5/overview
[6] https://leonardo-supercomputer.cineca.eu/hpc-system/
[7] https://www.it4i.cz/en/infrastructure/karolina/



(a) 32 GPUs (b) 128 GPUs

Figure 3: Communication patterns comparison.

The study is done running simulations of a Taylor-Green Vortex (TGV) at Ma = 0.1 and Re = 1600, so it can be solved using both compressible and incompressible solvers. The case is solved in fully 3D-periodic meshes of N^3 elements, with N being the number of elements per side. In the compressible solver, communications at each time-integration step are required for the solved variables, i.e., mass, energy, the three velocity components and the entropy used for stabilization. On the other hand, in the incompressible solver, the variables communicated at each time-step are the three velocity components and the entropy used for stabilization, besides the communications required in the solvers.

The results for both strong and weak speedup in the MareNostrum 5 supercomputer for the explicit compressible solver are depicted in Fig. 1 for meshes of order $p = 4$ ranging from 81M up to 1290M degrees of freedom (DoF). As can be seen, the scalability is very good for all the load ratios. This excellent scalability is also reflected in Fig. 2 where the iteration wall-clock is shown: on the left, results comparing different mesh sizes in the same cluster (MareNostrum 5); and on the right, the results for a specific mesh (the one with $647M$ DoF) in the three analyzed HPC platforms.

5. Communication patterns

Message Passing Interface (MPI) is a standard specification of message-passing interface for parallel computation in distributed-memory systems[8]. The MPI library allows multiple communication patterns between all the processes involved in the parallel execution, essentially: Point-to-point, Collective and One-sided communications.

In classical MPI implementation, only chunks of memory allocated in host memory can be sent and received using MPI messages. Nonetheless, when using GPUs the computing data is stored in GPU memory. Therefore, the sender GPU first have to be copied into host memory, then perform the MPI communication between the hosts, and then the host receiver copies the data in its GPU memory. On the other hand, if using the CUDA-aware MPI library along with the GPUDirect feature then it is not required to stage the GPU buffers through host memory, and the GPU buffers can be directly passed to MPI.

It is of interest to analyze the most efficient communication pattern for CFD codes like SOD2D. The main data sent and received through MPI messages is the data exchanged between ranks sharing an MPI boundary since the nodes replicated in different MPI ranks need to exchange data with its/their 'clones' in the neighbor MPI ranks. Therefore, an exhaustive analysis comparing different point-to-point and one-sided communication patterns has been carried out. The influence of CUDA-aware MPI library in communications is also analyzed. The study is carried out using a tool that mimics the communications required during a standard SOD2D run, i.e., the data exchange between ranks sharing MPI boundary nodes. Results for different meshes running in 32 and 128 are shown in Fig. 3. The cases using CUDA-aware MPI library are noted by (C). It is clear that using CUDA-aware MPI communications notably reduces communication times. Moreover, asynchronous two-sided Send-Receive (iSR) is the communication pattern with the smallest communication time, and therefore, the chosen as default in SOD2D. Nonetheless, one-sided Put and Get communications give very similar times and perform efficiently.

[8] https://hpc.nmsu.edu/discovery/mpi/introduction/

6. Conclusions

The present work has assessed and analyzed the parallel performance of the CFD code SOD2D, a Continuous Galerkin High-Order Spectral Element Code aimed to solve simulations of both turbulent compressible and incompressible flows. The obtained results in different HPC platforms demonstrate that the developed code presents very good scalability in GPUs, for both strong and weak speedups. Moreover, the study on communication patterns shows that the most efficient communication scheme is the asynchronous two-sided with CUDA-aware MPI. The one-sided communication schemes also showed good efficiency. Nonetheless, the most relevant conclusion is the impact of CUDA-aware MPI, which dramatically improves communication times when running on GPUs.

References

[1] A. Tekin, A. Tuncer Durak, C. Piechurski, D. Kaliszan, F. Aylin Sungur, F. Robertsén, P. Gschwandtner, State-of-the-art and trends for computing and interconnect network solutions for HPC and AI, Partnership for Advanced Computing in Europe (2021).

[2] J. Apostolakis, M. Bandieramonte, S. Banerjee, N. Bartosik, G. Corti, G. Cosmo, V. D. Elvira, T. Evans, A. Gheata, S. Pagan Griso, et al., Detector simulation challenges for future accelerator experiments, Frontiers in Physics 10 (2022) 913510. doi:10.3389/fphy.2022.913510.

[3] L. Gasparino, F. Spiga, O. Lehmkuhl, SOD2D: A GPU-enabled spectral finite elements method for compressible scale-resolving simulations, Computer Physics Communications 297 (2024) 109067. doi:10.1016/j.cpc.2023.109067.

[4] T. B. Gatski, J.-P. Bonnet (Eds.), Compressibility, Turbulence and High Speed Flow, second edition Edition, Academic Press, Oxford, 2013. doi:10.1016/B978-0-12-397027-5.00012-5.

[5] J.-L. Guermond, R. Pasquetti, B. Popov, Entropy viscosity method for nonlinear conservation laws, Journal of Computational Physics 230 (11) (2011) 4248–4267. doi:10.1016/j.jcp.2010.11.043.

[6] G. E. Karniadakis, M. Israeli, S. A. Orszag, High-order splitting methods for the incompressible Navier-Stokes equations, Journal of computational physics 97 (2) (1991) 414–443. doi:10.1016/0021-9991(91)90007-8.

[7] S. Sherwin, G. E. Karniadakis, A triangular spectral element method; applications to the incompressible Navier-Stokes equations, Computer methods in applied mechanics and engineering 123 (1-4) (1995) 189–229. doi:10.1016/0045-7825(94)00745-9.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

DNS of Intrinsically Unstable 3D Flames Using Deficient Reactant Thermochemistry: Validation and Scaling in NekRS

Hamid Kavari[a,*], Pasquale Eduardo Lapenna[a], Mathis Bode[b], Daniel Mira[c] and Francesco Creta[a]

[a]Sapienza University of Rome, Department of Mechanical and Aerospace Engineering, Via Eudossiana 18, 00184 Rome, Italy
[b]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52428 Jülich, Germany
[c]Barcelona Supercomputing Center (BSC-CNS), Plaça d'Eusebi Güell, 1-3, Les Corts, 08034 Barcelona, Spain

ARTICLE INFO†

Keywords:
Intrinsic Flame Instability;
Premixed Combustion;
Direct Numerical Simulation;
Flame Strain and Stretch;
Thermo-Diffusive Instability

ABSTRACT

Understanding the intrinsic instabilities of hydrogen flames is crucial for achieving net zero emissions. Direct Numerical Simulation (DNS) serves as a pivotal tool for this purpose, despite its high computational cost. With advancements in High Performance Computing (HPC) shifting towards GPUs, the deficient reactant model has been integrated into the NekRS framework to improve efficiency. This study validates the deficient reactant thermochemical model within the low-MACH number governing equations in NekRS. In addition, we present the strong scaling performance of this implementation.

1. Introduction

The use of hydrogen as an alternative to carbon-based fuels presents unique challenges due to intrinsic premixed flame instabilities that can profoundly impact flame characteristics [1]. To explore these instabilities Computational Fluid Dynamics (CFD) can be a pivotal tool, where using high fidelity Direct Numerical Simulation (DNS) plays a crucial role in analyzing flame morphology. However, achieving a practical implementation of DNS in complex combustion simulations requires substantial computational resources. Hence, all of the physical scales of flow and flame have to be resolved, and using High Performance Computing (HPC) capabilities becomes crucial [2]. NekRS [3] is a state-of-the-art GPU-accelerated CFD solver that relies on the high-order Spectral Element Method (SEM). In the present study, validation and scaling of the deficient reactant thermochemical model [1, 4] within the low-MACH number governing equations [5] implemented and coupled to NekRS, based on a well-established 2D DNS data set [1, 4, 6, 7] of intrinsic instabilities of premixed flames is presented.

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02498 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
hamid.kavari@uniroma1.it (H. Kavari);
pasquale.lapenna@uniroma1.it (P.E. Lapenna); m.bode@fz-juelich.de (M. Bode); daniel.mira@bsc.es (D. Mira); francesco.creta@uniroma1.it (F. Creta)
ORCID(s): 0009-0000-1122-5342 (H. Kavari); 0000-0002-1966-8743 (P.E. Lapenna); 0000-0001-9922-9742 (M. Bode); 0000-0001-9901-7942 (D. Mira); 0000-0003-1923-0810 (F. Creta)

2. Numerical method

The deficient reactant model employs a single-step, irreversible reaction with constant transport properties, integrated into the low-MACH equations, coupled with the equation of state to solve through high-order SEM. The dimensionless transport equations are expressed as follows [6]

$$\rho\frac{D\boldsymbol{u}}{Dt} = -\nabla p_2 + \frac{1}{Re}\nabla \cdot \left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T - \frac{2}{3}(\nabla\cdot\boldsymbol{u})\boldsymbol{I}\right), \quad (1)$$

$$\rho\frac{D\theta}{Dt} = \nabla\cdot(\delta_c\nabla\theta) + \frac{\Omega}{\delta_c}, \quad (2)$$

$$\rho\frac{DY}{Dt} = \nabla\cdot(\frac{\delta_c}{Le}\nabla Y) - \frac{\Omega}{\delta_c}. \quad (3)$$

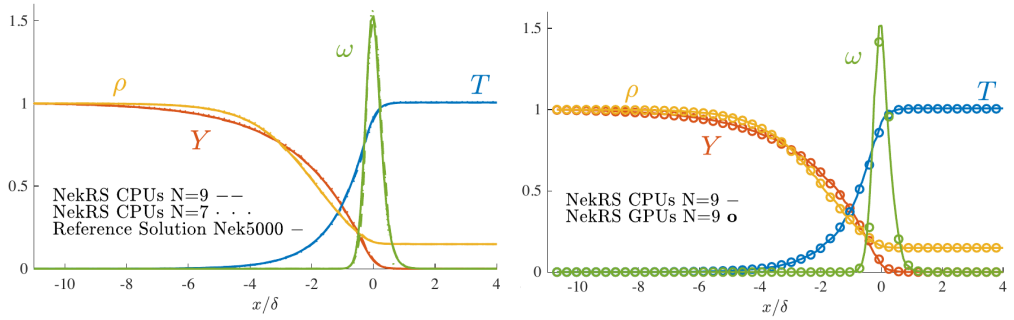
The continuity equation is represented by the non-zero velocity divergence resulting from both the heat release rate and thermal diffusion, as a restriction on velocity divergence, becomes

$$\nabla\cdot\mathbf{u} = (\sigma-1)\left[\nabla\cdot(\delta_c\nabla\theta) + \frac{\Omega}{\delta_c}\right], \quad (4)$$

$$\rho = (\theta(\sigma-1)+1)^{-1}, \quad (5)$$

$$\Omega = \frac{Ze^2}{2Le}Y\exp\left(\frac{Ze(\theta-1)}{1+(1-\sigma^{-1})(\theta-1)}\right), \quad (6)$$

where Ω represents the source or sink term based on a single-step Arrhenius reaction and $\mathbf{I}$ is the identity tensor. Also variables including the velocity $\mathbf{u}$, the hydrodynamic pressure p_2, temperature $\theta = (T-T_u)/(T_{ad}-T_u)$, the mass fraction of the deficient reactant Y are non-dimensionalized. In addition, the LEWIS number Le, ZELDOVICH number Ze, REYNOLDS number Re, the unburnt-to-burned density ratio



(a) CPUs. (b) CPUs and GPUs.

Figure 1: Validation of the EoS independent formulation in NekRS for CPUs and GPUs.

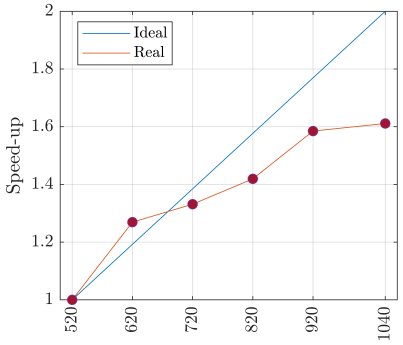


Figure 2: Strong scaling on JUWELS BOOSTER.

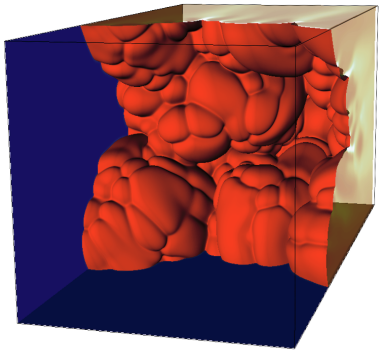


Figure 3: Isosurface of the progress variable of the flame simulation.

$\sigma = \rho_u/\rho_b$ are non-dimensional parameters (see [4] for further details). The only distinction with [6] lies in scaling the non-dimensional flame thickness δ_c, with respect to the cut-off wavelength λ_c rather than the reference hydrodynamic length L, as further detailed in [8].

3. Results

The deficient reactant model has been implemented and extensively utilized in various references [1, 4, 6] within Nek5000, employing a reference solution of a 1D laminar lean hydrogen-air premixed flame to validate the NekRS code. In the Fig. 1a the results of 1D solution, for SERIAL back-end which uses CPUs taking advantage of the Mare-Nostrum 4 cluster at Barcelona Supercomputing Center (BSC), for two different polynomial order of $N = 7$ and $N = 9$, were compared with the reference solution of Nek5000. The comparison shows a good agreement for T, Y, ω, ρ for all three cases. Also, Fig. 1b compares the results of CUDA back-end using CTE-POWER cluster at BSC, with the validated SERIAL back-end with consistent validation results.

In order to perform a strong scaling of the code, a test case of thermo-diffusive unstable flame with around 18.5 M spectral elements of polynomial order of 7, translating into 6.3 B gird point has been tested on JUWELS Booster cluster in Forschungszentrum Jülich. Each node of this cluster is equipped with four NVIDIA A100, 40GB GPUs and the least amount of nodes needed for the test case to fit the device memory is 130 which has 520 MPI rank. To compare the speed-up across different MPI ranks, we selected a time

instance where the flame is corrugated, as a restart point. The calculation time for a single time-step, averaged over one thousand time-steps, is selected as the speed-up metric. As it can be seen in Fig. 2 different MPI ranks speed-up reported and the baseline is the 130 nodes, where the speed-up is compared to it. As can be seen in Fig. 2 the MPI rank 920 is the highest rank which is close to the ideal speed-up line, while increasing the rank will lead to efficiency reduction. In addition, to demonstrate the domain and flame, Fig. 3 represents an isosurface of progress variable of 0.7, showing a corrugated flame.

4. Conclusions

This study emphasizes the importance of understanding hydrogen flame instabilities for achieving net zero emissions. DNS is crucial despite its computational demands. Implementing the deficient reactant model in NekRS enhances efficiency and offers insights into flame behavior. The validation of a low-MACH formulation alongside strong scaling analysis showcases promising performance gains across different computational architectures, advancing our understanding of complex combustion dynamics. In addition, to advance this study and consider species effects in hydrogen flames, NekCRF [9], a detailed kinetic chemistry solver, will be employed for comparative analyses. This approach is expected to yield deeper insights into flame behavior and enhance our understanding of these complex combustion processes.

Acknowledgment

The authors acknowledge computing time grants for the projects TurbulenceSL and rfcd by the JARA-HPC Vergabegremium provided on the JARA-HPC Partition part of the supercomputer JURECA at Jülich Supercomputing Centre, Forschungszentrum Jülich and the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

Part of this work has been performed under the Project HPC-EUROPA3 (INFRAIA-2016-1-730897), with the support of the EC Research Innovation Action under the H2020 Programm; in particular, P.E.L. gratefully acknowledges the support of the Computer Applications in Science & Engineering (CASE) Department and the Propulsion Technologies Group at the Barcelona Supercomputing Center, and Dr. Guillermo Oyarzun for the support in the implementation of the deficient reactant combustion model in nekRS.

References

[1] F. Creta, P. E. Lapenna, R. Lamioni, N. Fogla, M. Matalon, Propagation of premixed flames in the presence of darrieus–landau and thermal diffusive instabilities, Combustion and Flame 216 (2020) 256–270. doi:10.1016/j.combustflame.2020.02.030.

[2] D. Mira, E. J. Pérez-Sánchez, R. Borrell, G. Houzeaux, Hpc-enabling technologies for high-fidelity combustion simulations, Proceedings of the Combustion Institute 39 (4) (2023) 5091–5125. doi:10.1016/j.proci.2022.07.222.

[3] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, et al., Nekrs, a gpu-accelerated spectral element navier–stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[4] F. Creta, R. Lamioni, P. E. Lapenna, G. Troiani, Interplay of darrieus-landau instability and weak turbulence in premixed flame propagation, Physical Review E 94 (5) (2016) 053102. doi:10.1103/PhysRevE.94.053102.

[5] A. Tomboulides, J. Lee, S. Orszag, Numerical simulation of low mach number reactive flows, Journal of Scientific Computing 12 (1997) 139–167. doi:10.1023/A:1025669715376.

[6] R. Lamioni, P. E. Lapenna, G. Troiani, F. Creta, Flame induced flow features in the presence of darrieus-landau instability, Flow, Turbulence and Combustion 101 (4) (2018) 1137–1155. doi:10.1007/s10494-018-9936-0.

[7] H. Kavari, P. E. Lapenna, D. Mira, F. Creta, Flow features induced by thermo-diffusive instability: comparison between two-dimensional and three-dimensional flames, in: Proceedings of the 11th European Combustion Meeting 2023, 2023.

[8] P. E. Lapenna, R. Lamioni, G. Troiani, F. Creta, Large scale effects in weakly turbulent premixed flames, Proceedings of the Combustion Institute 37 (2) (2019) 1945–1952. doi:10.1016/j.proci.2018.06.154.

[9] S. Kerkemeier, C. E. Frouzakis, A. G. Tomboulides, P. Fischer, M. Bode, nekCRF: A next generation high-order reactive low Mach flow solver for direct numerical simulations (2024). arXiv:2409.06404.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Computational Investigation of the Atmospheric Boundary Layer in the GABLS Benchmark Problem Using the Spectral Element Code NekRS

Damaskinos Konioris[a], Dimitrios Papageorgiou[a,*], Ioannis Kavroulakis[a], Mathis Bode[b], Misun Min[c], Paul Fischer[c,d] and Ananias Tomboulides[a]

[a]Aristotle University of Thessaloniki, Department of Mechanical Engineering, Egnatia Street, 54124 Thessaloniki, Greece
[b]Forschungszentrum Jülich (FZJ) GmbH, Jülich Supercomputing Centre (JSC), Wilhelm-Johnen-Straße, 52428 Jülich, Germany
[c]Argonne National Laboratory, Mathematics and Computer Science Division, 9700 S. Cass Avenue, Lemont, IL 60439, USA
[d]University of Illinois at Urbana-Champaign, Department of Computer Science, 901 West Illinois Street, Urbana, IL 61801, USA

ARTICLE INFO†

Keywords:
Atmospheric Boundary Layer;
Turbulence;
Large-Eddy Simulation;
NekRS;
High-Performance Computing;
Graphics Processing Unit

ABSTRACT

This paper investigates the stable atmospheric boundary layer (ABL) using the spectral element code NekRS. Simulations were performed with various grid resolutions to study grid independence and performance. The results show that grid independence is achieved at higher grid resolutions, with minor differences observed between the finest grids. Performance tests on two high-performance computing (HPC) systems, Summit and JUWELS-Booster, reveal that newer generation GPUs offer significant performance improvements. The study underscores the critical role of advanced HPC systems in enabling detailed simulations of complex ABL phenomena and highlights the potential of exascale computing and hybrid CPU-GPU architectures for future research.

1. Introduction

The Atmospheric Boundary Layer (ABL) is pivotal in regulating atmospheric dynamics, facilitating the exchange of heat, moisture, and pollutants between the Earth's surface and the free atmosphere. Its accurate representation is essential for various practical applications, including renewable energy generation and pollution dispersion. Numerical weather prediction models, climate simulations, and environmental impact assessments heavily rely on precise ABL modeling [1]. ABL flows exhibit high turbulence and density stratification due to factors like surface heating, Coriolis effects, and complex weather patterns. Large eddy simulation (LES) is commonly employed for ABL studies, utilizing subgrid-scale (SGS) models to capture smaller flow scales. However, LES demands dense grids for high-fidelity results, as accuracy correlates strongly with grid resolution [2]. HPC systems enable computationally intensive simulations of ABL phenomena, a capability further enhanced by the upcoming transition to exascale computing and the evolution of hybrid CPU-GPU architectures [3].

To quantify the effects of numerical modeling and discretization choices, the ABL community has set up a sequence of benchmark problems, GABLS1–3, which is an acronym for the GEWEX (Global Energy and Water Cycle Experiment) Atmospheric Boundary Layer Study (GABLS). In this work, the GABLS1 case, was simulated using the massively parallel high-order spectral element GPU-based code NekRS [4] in order to investigate the horizontal flow through a stable atmospheric boundary layer utilizing an LES approach. Various grid resolutions were utilized in order to study the convergence of the tested SGS model as a function of mesh refinement. Strong and weak scaling tests were also performed to better understand the code's attributes and possible limitations.

2. Simulation setup

In this work, the GABLS1 benchmark problem [5] is considered, emulating a stable ABL in which the basic temperature (at $z = 0$) is cooler than the air temperature and where the ground temperature continues to cool during the simulation. The simulation domain is $\Omega = Lx \times Ly \times Lz = 400\ \text{m} \times 400\ \text{m} \times 400\ \text{m}$, where x, y,z is the streamwise, spanwise direction and vertical direction respectively. The simulations are initialized (at $t = 0$) with constant velocity in the downstream direction equal to the geostrophic wind speed $U = 8$ m/s. The initial potential temperature is 265 K at $0 \leq z \leq 100$ m and increases linearly at a rate of 0.01

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02499 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
dkonioris@meng.auth.gr (D. Konioris); dpapageor@meng.auth.gr (D. Papageorgiou); ikkavroul@meng.auth.gr (I. Kavroulakis); m.bode@fz-juelich.de (M. Bode); mmin@mcs.anl.gov (M. Min); fischerp@illinois.edu (P. Fischer); ananiast@meng.auth.gr (A. Tomboulides)
ORCID(s): 0009-0008-0589-4679 (D. Konioris); 0009-0009-0854-8971 (D. Papageorgiou); 0009-0004-9841-9615 (I. Kavroulakis); 0000-0001-9922-9742 (M. Bode); 0000-0002-5646-5689 (M. Min); 0000-0002-6506-4502 (P. Fischer); 0000-0002-6654-2940 (A. Tomboulides)

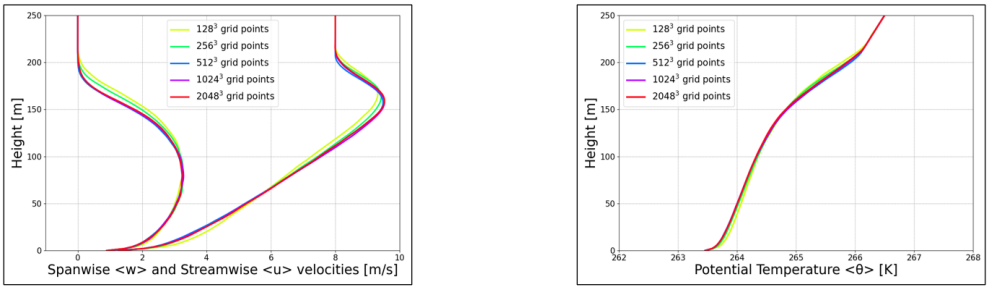
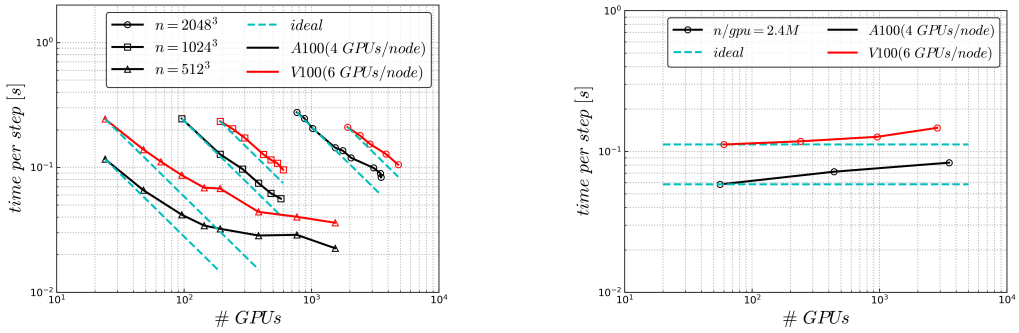


Figure 1: NekRS grid convergence plot for the GABLS1 case of the spanwise and streamwise velocity profiles (left) and potential temperature (right) at t = 9h for grid resolutions of 128^3, 264^3, 512^3, $1{,}024^3$ and $2{,}048^3$ total grid points.



(a) Strong scaling.

(b) Weak scaling.

Figure 2: Strong scaling (a) and weak scaling (b) of GABLS case with $2{,}048^3$ grid points on Summit (NVIDIA V100) [3] and JUWELS-BOOSTER (NVIDIA A100).

K/m at 100 m ≤ z ≤ 400 m. The reference temperature is 263.5 K. The REYNOLDS number is Re = UL_b/ν, where L_b = 100 m is the thickness of the initial thermal boundary layer and ν is the molecular viscosity, and is approximately 50M. An initial perturbation is added to the temperature with an amplitude of 0.1 K at the potential temperature for 0 ≤ z ≤ 50 m. Periodic boundary conditions are used in the streamwise and spanwise directions. At the top boundary, (z = 400 m), a stress-free rigid cap is applied to the momentum and a heat flux condition to the energy equation according to the temperature gradient of 0.01 K/m initially set at the upper region of the flow. In the lower boundary, traction BCs are applied for the velocity where the shear stress is obtained from the Monin-Obukhov similarity theory [6], and a heat flux BC is applied for the energy equation. The surface temperature is obtained from the GABLS specification following the rule θ_b(t) = 265 – 0.25t, where t is in hours. For the SGS stresses, a Smagorinsky (SMG) model [7] for the isotropic part of SGS stresses in conjunction with the mean-field eddy viscosity (MFEV) approach for their anisotropic part [8] are utilized. The convergence capabilities of this SGS approach was tested for various grid resolutions, with 128^3, 256^3, 512^3, $1{,}024^3$ and $2{,}048^3$ total grid points. The simulation was performed until t = 9 hours.

3. Results

A grid independence analysis was conducted with NekRS, utilizing grid resolutions of $128^3, 264^3, 512^3$, $1{,}024^3$ and $2{,}048^3$ total grid points. The horizontally averaged spanwise and streamwise velocity profiles at t = 9 hours are illustrated in Fig. 1 (left) whereas the horizontally averaged potential temperature profiles are shown in Fig. 1 (right). The results indicate that grid independence is practically achieved at a

grid resolution equal or higher than 512^3 as the differences in the velocity and potential temperature profiles between the 512^3 and $1{,}024^3$ grids are minor. These results are compared with the results from the $2{,}048^3$ mesh to confirm this.

Next, we evaluate GPU performance for the larger cases (512^3, $1{,}024^3$ and $2{,}048^3$) on two HPC systems: Summit, which employs NVIDIA V100 GPUs, and JUWELS-Booster, which utilizes NVIDIA A100 GPUs. The results, depicted in Fig. 2, show on the left side performance for strong scaling and on the right side performance for weak-scaling as a function of the number of GPUs. Ideal speed-up curves are also presented for comparison. Our findings indicate that, for the same number of grid points, the performance of the code on A100 GPUs on the Jewels Booster is superior by a factor of 1.5 compared to V100 GPUs on Summit. Despite this difference in raw performance, the parallel efficiency remains consistent for both GPU types when normalized by the number of grid points per GPU. However, a significant decline in parallel efficiency is observed when the grid points per GPU drop below 2.5 million, with efficiency falling below 80%.

4. Acknowledgements

This work has received funding from the European High-Performance Computing Joint Undertaking and Sweden, Germany, Spain, Greece and Denmark under Grant agreement No. 101093393. Also, the authors gratefully acknowledge the Gauss Centre for Supercomputing for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[1] J. Sanz Rodrigo, M. Churchfield, B. Kosovic, A methodology for the design and testing of atmospheric boundary layer models for wind energy applications, Wind Energy Science 2 (1) (2017) 35–54. doi:10.5194/wes-2-35-2017.

[2] M. A. Sprague, S. Boldyrev, P. Fischer, R. Grout, W. I. Gustafson, Jr., R. Moser, Turbulent Flow Simulation at the Exascale: Opportunities and Challenges Workshop: August 4-5, 2015, Washington, D.C., Tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO (United States) (2017). doi:10.2172/1338668.

[3] M. Min, M. J. Brazell, A. Tomboulides, M. J. Churchfield, P. F. Fischer, M. A. Sprague, Towards exascale for wind energy simulations (2022). arXiv:2210.00904.

[4] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[5] R. J. Beare, M. K. Macvean, A. A. M. Holtslag, J. Cuxart, I. Esau, J.-C. Golaz, M. A. Jimenez, M. Khairoutdinov, B. Kosovic, D. Lewellen, T. S. Lund, J. K. Lundquist, A. McCabe, A. F. Moene, Y. Noh, S. Raasch, P. Sullivan, An intercomparison of large-eddy simulations of the stable boundary layer, Boundary-Layer Meteorology 118 (2) (2006) 247–272. doi:10.1007/s10546-004-2820-6.

[6] A. Monin, A. M. Obukhov, Basic laws of turbulent mixing in the ground layer of atmosphere, 1954.
URL https://apps.dtic.mil/sti/citations/AD0672723

[7] J. Smagorinsky, General circulation experiments with the primitive equations, Monthly Weather Review 91 (3) (1963) 99–164. doi:10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[8] P. P. Sullivan, J. C. McWilliams, C.-H. Moeng, A subgrid-scale model for large-eddy simulation of planetary boundary-layer flows, Boundary-Layer Meteorology 71 (3) (1994) 247–276. doi:10.1007/BF00713741.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

In-Situ Visualization With Ascent and NekRS for Large-Scale CFD Problems on GPUs

Jens Göbbert[a,*], Damian Alvarez[a], Mathis Bode[a], Paul Fischer[b,c], Christos Emmanouil Frouzakis[d], Joseph A. Insley[c], Yu-Hsiang Lan[b,c], Victor A. Mateevitsi[c], Misun Min[c], Michael E. Papka[c], Silvio Rizzi[c], Roshan J. Samuel[e] and Jörg Schumacher[e]

[a]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany
[b]University of Illinois at Urbana-Champaign, 601 E. John Street, IL 61820 Urbana-Champaign, IL, USA
[c]Argonne National Laboratory, 9700 S Cass Ave, IL 60439 Lemont, IL, USA
[d]ETH Zurich, Sonneggstr. 3, CH-8092 Zurich, Switzerland
[e]TU Ilmenau, Ehrenbergstraße 29, 98693 Ilmenau, Germany

ARTICLE INFO†

Keywords:
Computational Fluid Dynamics Workflows;
NekRS;
Graphics Processing Unit;
Visualization

ABSTRACT

This paper discusses in-situ visualization in the context of high-order GPU-based CFD solvers. For this purpose, ASCENT was coupled with NekRS and successfully used on JUWELS Booster and Frontier to visualize CFD applications, a Rayleigh-Bénard convection case and a Pebble Bed reactor, at scale. The setup allowed time-resolved visualization at low additional computational cost.

1. Introduction

Exascale supercomputers make it possible to address larger scientific problems using numerical methods and/or to solve them in a shorter time. This enables, for example, the simulation of flow problems with ever increasing scale separation, on the one hand as a result of turbulence or on the other hand due to multi-physics effects such as combustion, or the training of ever larger networks with ever more data. One energy-efficient way to realize an exascale supercomputer is by exploiting GPUs. Systems such as Frontier, the first US exascale supercomputer, or JUPITER, probably the first European exascale supercomputer, obtain the majority of their FLOP/s (floating point operations per second) from GPUs. The resulting heterogeneous systems with CPUs and GPUs increase the level of complexity by a further level, which makes data handling in particular more difficult. In the negative case, separate CPU and GPU memory forces a large number of expensive copy operations. Conversely, applications are generally only computationally efficient if they avoid copy operations to the HDD and limit copy operations between CPU and GPU memory.

Irrespective of the efficient supercomputer utilization described above, the size of the data generated by exascale simulations can be a further obstacle to successfully solving the scientific problem on which a simulation is based. Checkpoint files from exascale simulations can be several 100 TB in size, which is often too large for conventional tools for analysis and visualization. This means that writing the results is not only very expensive, which limits the number of checkpoint files that can be written, but the handling of checkpoint files in post-processing is also difficult.

In-situ analysis and visualization, i.e. the immediate processing of data while it is still in the computing memory, can both reduce copying operations and save the time-consuming handling of checkpoint files. Furthermore, they enable real-time information on the simulation, which can be used for simulation monitoring. Hence, in-situ analysis and visualization give researchers immediate access to visualizations even in the exascale era, enabling faster decision-making, and a deeper understanding of the phenomena under investigation.

Fluid mechanics often features transient or fluctuating processes and is therefore an ideal field of application for in-situ analysis and visualization. Especially in the context of flow solvers, which mainly run on GPUs, enormous

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02500 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
j.goebbert@fz-juelich.de (J. Göbbert); d.alvarez@fz-juelich.de (D. Alvarez); m.bode@fz-juelich.de (M. Bode); fischerp@illinois.edu (P. Fischer); cfrouzakis@ethz.ch (C.E. Frouzakis); insley@anl.gov (J.A. Insley); lan14@illinois.edu (Y. Lan); vmateevitsi@anl.gov (V.A. Mateevitsi); mmin@anl.gov (M. Min); papka@anl.gov (M.E. Papka); srizzi@anl.gov (S. Rizzi); roshan-rohn.samuel@tu-ilmenau.de (R.J. Samuel); joerg.schumacher@tu-ilmenau.de (J. Schumacher)
ORCID(s): 0000-0002-3807-6137 (J. Göbbert); 0000-0002-8455-3598 (D. Alvarez); 0000-0001-9922-9742 (M. Bode); 0000-0002-6506-4502 (P. Fischer); 0009-0006-0963-4650 (C.E. Frouzakis); 0000-0002-6955-869X (J.A. Insley); 0000-0002-1680-675X (Y. Lan); 0000-0002-6677-7520 (V.A. Mateevitsi); 0000-0002-5646-5689 (M. Min); 0000-0002-6418-5767 (M.E. Papka); 0000-0002-3804-2471 (S. Rizzi); 0000-0002-1280-9881 (R.J. Samuel); 0000-0002-1359-4536 (J. Schumacher)

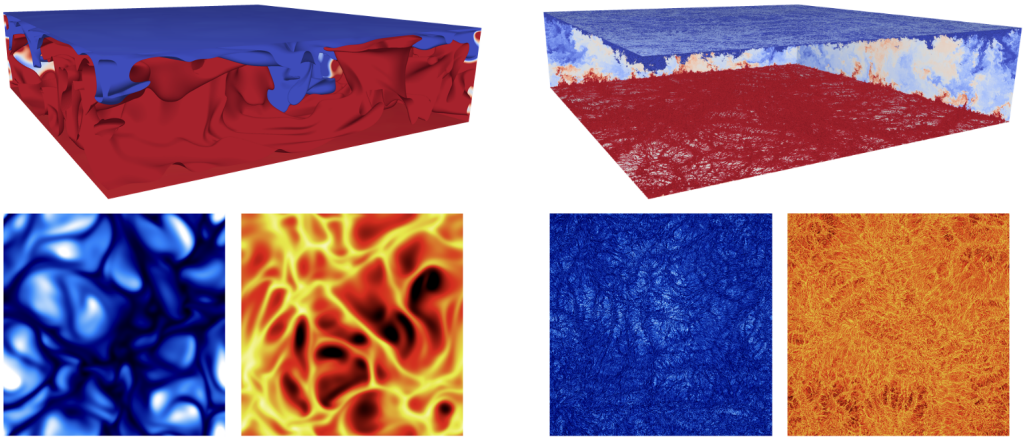


Figure 1: RBC overview for $Ra = 10^6$ (left) and $Ra = 10^{12}$ (right) in the top line. Bottom line shows cut planes through the boundary layers at the top and bottom.

advantages can arise. This is demonstrated in this paper using the example of NekRS [1] for two applications.

In the following, it is first explained how in-situ visualization can be integrated into NekRS using ASCENT. Subsequently, measurements for a Rayleigh-Bénard convection (RBC) case on JUWELS Booster and a pebble bed reactor (PBR) on Summit and Frontier are presented. This work finishes with conclusions.

2. Methodology

This work couples the state-of-the-art GPU-based CFD solver NekRS with the in-situ visualization library ASCENT to reduce the overall time-to-solution (incl. visualization) of large-scale CFD simulations. In selecting a suitable in-situ analysis and visualization library for this work, ASCENT emerged as the preferred choice due to several distinct advantages. First and foremost, its lightweight design and ease of integration into existing simulation code bases significantly reduce the implementation overhead. In addition, ASCENT supports zero-copy GPU-to-GPU data passing, which allows the direct transfer of device pointers between the simulation code and ASCENT, ensuring that data remains exclusively on the GPU. Such an approach effectively eliminates the need to pass data back to the CPU, eliminating a common bottleneck in high-performance computing applications.

The implications of this GPU-centric data handling strategy are profound, particularly in terms of minimizing the memory footprint of the in-situ visualization process. Traditional data movement to the CPU introduces significant memory overhead and duplication [2], and by bypassing this transfer, ASCENT facilitates more efficient use of computational resources.

3. Applications

The coupled in-situ workflow was successfully used for two applications. The focus in each case was to avoid data copying and to visualize directly on the GPUs. The visualization frequency was chosen to capture the smallest temporal scales and the computational overhead was measured. The first use case was Rayleigh-Bénard convection (RBC) on JUWELS Booster. JUWELS Booster has 936 GPU-accelerated nodes, is fully directly liquid cooled, and is based on Atos' Sequana XH2000 architecture. Each node relies on 2x AMD EPYC 7042 processors that orchestrate 4x NVIDIA A100 GPUs. These GPUs have the SXM4 form factor, and are integrated in a so-called Redstone board, in which the GPUs are interconnected in direct all-to-all NVLink3 fabric. The RBC simulation [3] used up to 900 nodes for RAYLEIGH numbers (Ra) up to 10^{12} that ran on 46.7 billion grid points. The resulting visualizations of two different RAYLEIGH numbers, 10^6 and 10^{12}, are presented in Fig. 1.

The second example is the full core of pebble bed reactor (PBR), illustrated in Fig. 2, which has 352,625 spherical pebbles and the fluid mesh comprising 98 million spectral elements of order 7, or 33.8 billions grid points. The PBR

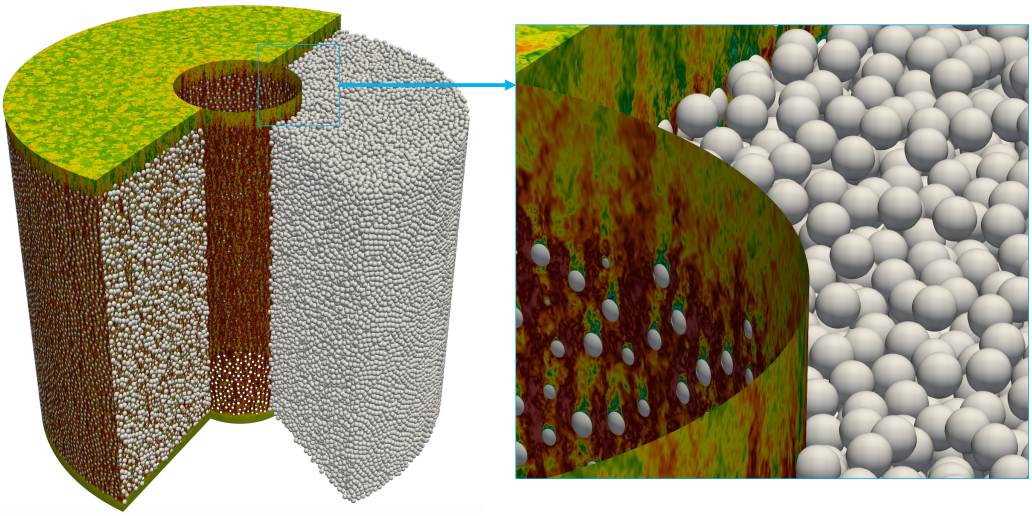


Figure 2: Full core LES of PBR with 352k pebbles and (right) close-up showing velocity distribution.

was the first wall-resolved large Eddy simulation (LES) for the full reactor core [4]. It has been demonstrated NekRS can simulate a single flow-through time in less than 6 hours using the entire Summit, a post-petascale machine in OLCF. The in-situ workflow makes it possible to visualize high fidelity transient solutions at a very high resolution which provides more insights in dynamic turbulent structure. Using 1,400 nodes on Frontier, the PBR case renders 8K images (8,192 x 8,192 pixels) every 20 timesteps and producing 710 images in just 4 hours.

4. Discussion and conclusions

The implemented in-situ workflow allows to reduce the overhead for a time-resolved visualization below 10 % for the RBC application and enables the realization of high-resolution images for the PBR case. In this work, the focus was on avoiding copying processes between GPU and CPU. In the future, it could be interesting to do the visualization in parallel on the otherwise idling CPUs, although this requires a periodic copy operation. For more complex visualizations and if the CPU is otherwise not used at all, this could be the most efficient solution overall.

Acknowledgements

This work was supported by and used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility supported under Contract DE-AC02-06CH11357. This work was supported by Northern Illinois University. This work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract DE-AC02-06CH11357, through the grant “Scalable Analysis Methods and In Situ Infrastructure for Extreme Scale Knowledge Discovery”, program manager Dr. Margaret Lenz. The authors from JSC acknowledge computing time grants for the project TurbulenceSL by the JARA-HPC Vergabegremium provided on the JARA-HPC Partition part of the supercomputer JURECA at Jülich Supercomputing Centre, Forschungszentrum Jülich, the Gauss Centre for Supercomputing e.V.[1] for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC), and funding from the European Union’s Horizon 2020 research and innovation program under the Center of Excellence in Combustion (CoEC) project, grant agreement no. 952181. Support by the Joint Laboratory for Extreme Scale Computing (JLESC)[2] for traveling is acknowledged.

[1] https://www.gauss-centre.eu
[2] https://jlesc.github.io/

References

[1] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[2] V. A. Mateevitsi, M. Bode, N. Ferrier, P. Fischer, J. H. Göbbert, J. A. Insley, Y.-H. Lan, M. Min, M. E. Papka, S. Patel, S. Rizzi, J. Windgassen, Scaling Computational Fluid Dynamics: In Situ Visualization of NekRS using SENSEI, in: Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, ACM, New York, NY, USA, 2023, pp. 862–867. doi:10.1145/3624062.3624159.

[3] R. J. Samuel, M. Bode, J. D. Scheel, K. R. Sreenivasan, J. Schumacher, No sustained mean velocity in the boundary region of plane thermal convection, Journal of Fluid Mechanics 996 (2024) A49. doi:10.1017/jfm.2024.853.

[4] M. Min, Y.-H. Lan, P. Fischer, E. Merzari, S. Kerkemeier, M. Phillips, T. Rathnayake, A. Novak, D. Gaston, N. Chalmers, T. Warburton, Optimization of Full-Core Reactor Simulations on Summit, in: SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, New York, NY, USA, 2022, pp. 1–11. doi:10.1109/SC41404.2022.00079.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Unraveling Turbulent NH_3/H_2 Flames Using High Performance GPU Computing: A Series of Spectral Element Method-Based High-Fidelity DNS

Driss Kaddar[a], Hendrik Nicolai[a,*], Vinzenz Schuh[a], Antonia Bähr[a], Christos Emmanouil Frouzakis[b], Mathis Bode[c] and Christian Hasse[a]

[a]Technical University of Darmstadt (TUDa), Department of Mechanical Engineering, Simulation of reactive Thermo-Fluid Systems, Otto-Berndt-Str. 2, 64287 Darmstadt, Germany
[b]ETH Zurich, Sonneggstr. 3, CH-8092 Zurich, Switzerland
[c]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany

ARTICLE INFO†

Keywords:
Ammonia-Hydrogen Combustion;
Spectral-Element Method;
Direct Numerical Simulation;
Graphics Processing Unit
Parallel Computing;
Reactive Flows;
Flamelet Modeling

ABSTRACT

Ammonia-hydrogen blends are crucial for future carbon-free combustion systems, with staged-combustion technologies like rich-quench-lean being proposed to minimize emissions. However, the combustion behavior of turbulent rich ammonia-hydrogen mixtures is not well understood, particularly regarding phenomena like partial cracking, hydrogen slip, and post-flame stratification. Recent HPC advancements, particularly in GPU-based systems, enable combustion DNS beyond academic configurations. Utilizing nekCRF, a new GPU-based spectral element solver based on nekRS, we perform finite-rate chemistry DNS of a rich, turbulent premixed jet flame. The analysis focuses on NH_3/H_2 interaction, revealing residual H_2, minimized NH_3 slip, and enhanced heat release through turbulent mixing. By leveraging GPU acceleration and employing a novel spectral element solver, this research not only advances our understanding of ammonia combustion but also showcases a paradigm shift in computational efficiency, offering a promising avenue for developing sustainable energy solutions.

1. Introduction

To address the pressing issue of climate change, society confronts the formidable task of transitioning away from fossil fuels in our energy system. Gas turbines for power generation and aviation are at the heart of our energy infrastructure. Renowned for their high thermal efficiencies and unmatched power densities, these machines have traditionally used hydrocarbon-containing fuels. Efforts are underway to develop gas turbines that efficiently use zero-carbon fuels, leading to a sustainable energy future. Although gas turbines are theoretically fuel-flexible, they can face significant operational challenges related to flame stability and emissions due to the vastly different combustion properties of H_2 and NH_3 fuels compared to conventional hydrocarbon fossil fuels [1].

For example, hydrogen's high diffusivity and reactivity drive its strong turbulent burning rate, influenced by turbulent fluctuations and intense thermo-diffusive instabilities from differential diffusion [2, 3]. Conversely, ammonia, a promising hydrogen carrier, requires precise NOx emission control during partial decomposition. Recent experimental and numerical evidence indicates that an RQL (Rich-Quench-Lean) operational strategy ensures both flame stabilization and low emissions in ammonia-fired combustors [4, 5, 6]. This study aims to fill the knowledge gap on pressure effects on these fuels, focusing on H_2/NH_3 flames with controlled flame propagation. By leveraging advanced high-performance computing tools, we seek to unlock the potential of fuel-flexible, high-power density combustion systems for zero-carbon gas turbines.

To fully exploit the potential of current and future HPC centers, a highly scalable reactive flow simulation code (nekCRF) [7] based on nekRS as part of the Center of Excellence in Combustion (CoEC) project during the last three years was developed, enabling high-fidelity direct numerical simulations (DNS) of configurations at technically relevant operating conditions. Here, nekCRF will be leveraged to study ammonia-hydrogen jet flames. The novel benchmark data will enable the assessment and further development of

† This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02501 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
kaddar@stfs.tu-darmstadt.de (D. Kaddar); nicolai@stfs.tu-darmstadt.de (H. Nicolai); schuh@stfs.tu-darmstadt.de (V. Schuh); antonia.baehr@t-online.de (A. Bähr); cfrouzakis@ethz.ch (C.E. Frouzakis); m.bode@fz-juelich.de (M. Bode); hasse@stfs.tu-darmstadt.de (C. Hasse)
ORCID(s): 0000-0002-7812-4738 (D. Kaddar); 0000-0002-0355-2252 (H. Nicolai); 0009-0003-8706-1008 (V. Schuh); 0009-0003-2602-4107 (A. Bähr); 0009-0006-0963-4650 (C.E. Frouzakis); 0000-0001-9922-9742 (M. Bode); 0000-0001-9333-0911 (C. Hasse)

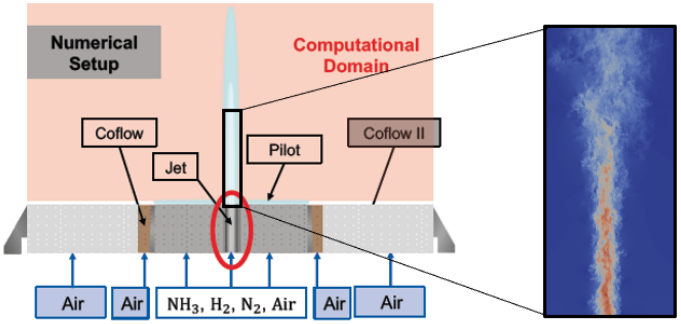


Figure 1: Schematic depiction of the McKenna burner configuration. Zoom shows the jet-flow field.

combustion models urgently needed for designing ammonia-hydrogen-fueled gas turbines.

2. Computational methodology

As simulation framework, nekRS [8] is used in combination with the chemistry plugin nekCRF developed as part of CoEC. It employs high-order spectral elements in which the solution, data, and test functions are represented as locally structured Nth-order tensor product polynomials on a set of E globally unstructured conforming hexahedral brick elements. Time integration in nekRS employs a semi-implicit splitting scheme, utilizing kth-order (k up to three) backward differences (BDFk) to approximate the time derivative, resulting in an implicit treatment of the viscous and pressure terms, and kth-order extrapolation (EXTk) for the advection and forcing terms. The discretization leads to a sequence of symmetric positive definite linear systems for pressure, velocity and temperature. Its scalability to millions of MPI ranks is based on the gsLib communication library, which handles all near-neighbor and other stencil type communications [8].

The thermochemistry (energy and species equations) is treated with the highly optimized chemistry plugin nekCRF that generates optimized kernels for the source term, thermodynamic and transport properties evaluation for GPUs. It also provides consistent advection and diffusion transport operators acting efficiently on multiple scalars. The resulting large system is integrated without further splitting of the convection, diffusion and reaction term using CVODE [9].

3. Configuration

The setup involves a turbulent premixed $NH_3/H_2/N_2$ jet flame operating under slightly rich conditions with an equivalence ratio of $\Phi = 1.2$. This configuration, illustrated in Fig. 1, is adapted from a McKenna burner studied experimentally at TU Darmstadt. This offers the advantage of benchmarking the base operating condition with 45 vol-% H_2 against available experimental data, enabling the exploration of uncertainties in the chemical mechanism. For the experimental conditions, the central jet velocity is set to 50 m/s, resulting in a REYNOLDS number of 15,000. The coflow, which stabilizes the flame, operates with a fuel-lean hydrogen mixture at an equivalence ratio of $\Phi = 0.7$. The flame is stabilized directly above the burner exit, and the exhaust gas velocity of the flame is used as an inlet condition for the DNS domain. The overall objective is to unravel the fundamental combustion physics of propagating hydrogen and ammonia flames under diverse operating conditions, spanning from atmospheric to high pressure (unlike the experiments). A critical free parameter in NH_3/H_2 combustion is the precracking ratio of ammonia. Determining

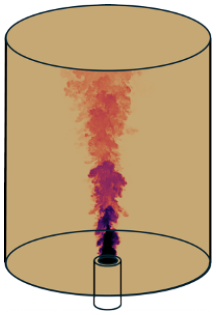


Figure 2: Instantaneous temperature of field of the McKenna burner.

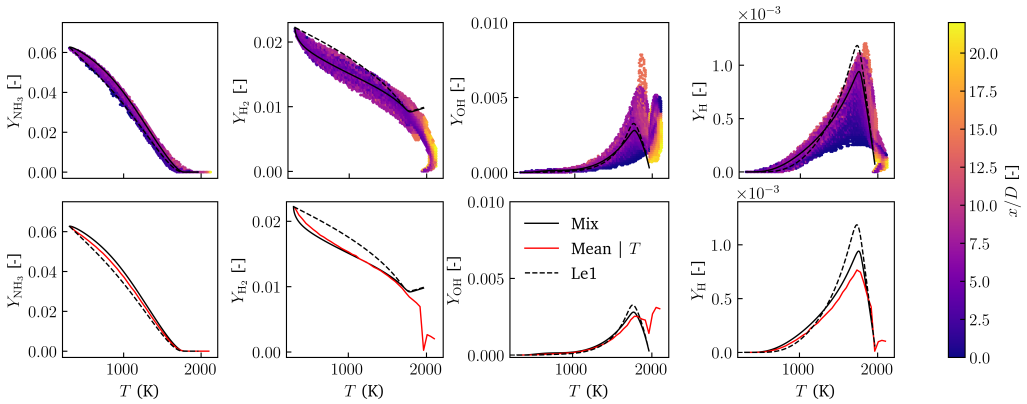


Figure 3: Top row: Instantaneous scatter plots of the thermochemical state together with unity LEWIS numbers and mixture-averaged flamelets. Bottom row: Conditional averaged of DNS data.

an optimum balance between flame stability (with high H_2 content), low emissions, and energy efficiency (with high NH_3 content) is crucial. Given the fully premixed state, this configuration is predestined for the analysis of turbulence-chemistry interactions. Of particular interest are processes of thermal ammonia decomposition, the impact of equivalence ratio variations on NOx formation.

4. Results

The flame depicted in Fig. 2 exhibits significant influence from turbulent flow, leading to intricate interactions between chemical reactions and mixing processes. In the instantaneous temperature field, sharp gradients and fluctuating regions of high and low temperatures are observed. These variations highlight the intense mixing and reaction zones within the flame, where hot combustion products and cooler reactants interact continuously, creating a highly dynamic and ever-changing thermal structure. Capturing these dynamics remains a challenge for state-of-the-art combustion models.

Figure 3 showcases the high-fidelity data from these DNS studies, illustrating flame structures as a function of temperature for the main fuel species (NH_3, H_2), and the OH and H radicals. Additionally, one-dimensional freely propagating flames with unity LEWIS number and mixture fractions are displayed, which are used to build manifold-based combustion models [3]. The scatter in the DNS data demonstrates reasonable agreement with both models. To establish a better comparison, temperature conditional averaging of the DNS data is performed, shown in the bottom row of Fig. 3. This comparison shows a favorable match of the conditional average with the mixture-averaged flamelets for both fuel species, with high-temperature differences attributed to unmodeled mixing with the surrounding co-flow of burnt gases, and low-temperature deviations due to turbulent mixing with ammonia pre-cracking. These effects are not included in current combustion models, which require extensions. Notably, the radicals OH and H also align well with the mixture-averaged flamelet. As these species are markers for curvature and strain impacts on turbulent flame structures, it can be concluded that NH_3/H_2 flames are less sensitive to curvature and strain than hydrogen flames, as reported in previous studies [3]. The scatter data used for conditional averaging are derived from the entire domain. Given that the co-flow is fuel-lean (providing sufficient oxygen for the flame) and the jet is fuel-rich, a mixing process occurs. This process involves the mixing of flue gases from the jet flame with the surrounding co-flow, resulting in the tails observed in the conditionally averaged DNS data. Additionally, the excess hydrogen burns in a weak secondary flame, which is suggested by the temperature increase after the main flame.

5. Conclusions and outlook

In this study, a series of SEM DNS simulations of NH_3/H_2 flames is conducted to explore the complexity of this novel fuel blend. Utilizing the advanced GPU flow solver NekRS with the chemistry plugin NekCRF, we vary parameters such as pressure and pre-cracking ratio. These high-fidelity results enable a thorough evaluation of state-of-the-art combustion models, including the manifold-based

models demonstrated here. At the conference, this assessment will be expanded to cover the entire parameter space and include a performance evaluation of the reactive flow solver.

6. Acknowledgements

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 101118139. The JU receives support from the European Union's Horizon Europe Program. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V.[1] for funding this project by providing computing time on the GCS Supercomputer JUWELS at JSC.

References

[1] D. Noble, D. Wu, B. Emerson, S. Sheppard, T. Lieuwen, L. Angello, Assessment of Current Capabilities and Near-Term Availability of Hydrogen-Fired Gas Turbines Considering a Low-Carbon Future, Journal of Engineering for Gas Turbines and Power 143 (4) (2021). doi:10.1115/1.4049346.

[2] L. Berger, A. Attili, H. Pitsch, Synergistic interactions of thermodiffusive instabilities and turbulence in lean hydrogen flames, Combustion and Flame 244 (2022) 112254. doi:10.1016/j.combustflame.2022.112254.

[3] H. Böttler, D. Kaddar, T. J. P. Karpowski, F. Ferraro, A. Scholtissek, H. Nicolai, C. Hasse, Can flamelet manifolds capture the interactions of thermo-diffusive instabilities and turbulence in lean hydrogen flames?—An a-priori analysis, International Journal of Hydrogen Energy 56 (2024) 1397–1407. doi:10.1016/j.ijhydene.2023.12.193.

[4] E. C. Okafor, K. K. A. Somarathne, A. Hayakawa, T. Kudo, O. Kurata, N. Iki, H. Kobayashi, Towards the development of an efficient low-NOx ammonia combustor for a micro gas turbine, Proceedings of the Combustion Institute 37 (4) (2019) 4597–4606. doi:10.1016/j.proci.2018.07.083.

[5] T. Indlekofer, S. Wiseman, K.-J. Nogenmyr, J. Larfeldt, A. Gruber, Numerical Investigation of Rich-Lean Staging in a SGT-750 Scaled Dry Low Emission Burner With Partially Decomposed Ammonia, Journal of Engineering for Gas Turbines and Power 145 (4) (2023). doi:10.1115/1.4055725.

[6] T. Heggset, O. H. H. Meyer, L. Tay-Wo-Chong, A. Ciani, A. Gruber, Numerical Assessment of a Rich-Quench-Lean Staging Strategy for Clean and Efficient Combustion of Partially Decomposed Ammonia in the Constant Pressure Sequential Combustion System, Journal of Engineering for Gas Turbines and Power 146 (8) (2024). doi:10.1115/1.4063958.

[7] S. Kerkemeier, C. E. Frouzakis, A. G. Tomboulides, P. Fischer, M. Bode, nekCRF: A next generation high-order reactive low Mach flow solver for direct numerical simulations (2024). arXiv:2409.06404.

[8] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[9] S. D. Cohen, A. C. Hindmarsh, P. F. Dubois, CVODE, A Stiff/Nonstiff ODE Solver in C, Computers in Physics 10 (2) (1996) 138–143. doi:10.1063/1.4822377.

[1] https://www.gauss-centre.eu

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Unraveling the Boundary Layers of High Rayleigh Number Convection Through Direct Numerical Simulations

Roshan J. Samuel[a,*], Prafulla P. Shevkar[a], Mathis Bode[b], Janet D. Scheel[c], Katepalli R. Sreenivasan[d] and Jörg Schumacher[a,d]

[a]Technische Universität Ilmenau, Institute of Thermodynamics and Fluid Mechanics, 98684, Ilmenau, Germany
[b]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, 52428, Jülich, Germany
[c]Occidental College, Department of Physics, CA 90041, Los Angeles, USA
[d]New York University, Tandon School of Engineering, NY 11201, New York, USA

ARTICLE INFO†

Keywords:
Rayleigh Bénard Convection;
Spectral Element Method;
Boundary Layers

ABSTRACT

We study the behavior of boundary layers in thermal convection at RAYLEIGH numbers ranging from 10^5 to 10^{12} in a box of aspect ratio $\Gamma = 4$, and periodically extended sides. We use the GPU accelerated spectral element solver, NekRS, for our simulations. At the highest RAYLEIGH number, the simulations utilize nearly 3400 A100 GPUs of the JUWELS Booster facility, to cover as many as 40 free-fall time units, enabling us to obtain reliable statistics in steady-state. This demonstrates the excellent scalability of the solver. Moreover, since the focus of the study is on the boundary layers at the top and bottom walls, we ensure that the near-wall regions are well-resolved. We observe that the boundary layers have both shear dominated and plume dominated regions, with a nearly constant area-fraction between these regions. We also study the structure of wall shear-stress fields and their connection to plume structures emanating from the walls.

1. Introduction

Thermal convection has remained a fertile benchmark for computational studies of fluid flows starting from the earliest works of [1]. The ubiquity of the phenomenon as well as the simplicity of the Rayleigh-Bénard convection (RBC) setup typically used to study natural convection has greatly impelled the progress in this field for more than half a century.

In RBC, a thin layer of fluid is confined between a pair of parallel, horizontal plates, with gravity acting along the vertical direction. The plates are isothermal, with the top plate maintained at a lower temperature than the bottom plate [2]. Ideally, the plates extend infinitely in the horizontal directions. One can numerically approximate this by using a moderately large aspect ratio for the computational domain, and by using periodic boundary conditions along the horizontal directions. This is a key design aspect of the simulations reported here.

Experiments of RBC are typically performed in closed domains with no-slip side-walls, which impact the organization of the flow. Our simulations alleviate this effect, thus eliminating the large-scale circulation (LSC) flow. This imparts a markedly different structure to the boundary layer flow, whose analysis is the focus of this work.

2. Problem Setup and Numerical Method

We solve the non-dimensionalized Navier-Stokes equations with Boussinesq approximation along with the energy equation [3],

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + T\hat{z} + \sqrt{\frac{Pr}{Ra}}\nabla^2\mathbf{u}, \tag{1}$$

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = \frac{1}{\sqrt{RaPr}}\nabla^2 T. \tag{2}$$

Here, $\mathbf{u}$, p and T denote the velocity, pressure and temperature fields respectively. Moreover, mass conservation is imposed in the form of the continuity equation, $\nabla\cdot\mathbf{u} = 0$. The dimensionless parameters Ra (RAYLEIGH number) and Pr (PRANDTL number) dictate the nature of the flow and its properties. While Ra represents the strength of convective forcing, Pr is the ratio between the diffusion of momentum and temperature. We keep Pr fixed at 0.7 (corresponding to

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02502 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
roshan-john.samuel@tu-ilmenau.de (R.J. Samuel); prafulla-prakash.shevkar@tu-ilmenau.de (P.P. Shevkar); m.bode@fz-juelich.de (M. Bode); jscheel@oxy.edu (J.D. Scheel); krs3@nyu.edu (K.R. Sreenivasan); joerg.schumacher@tu-ilmenau.de (J. Schumacher)
ORCID(s): 0000-0002-1280-9881 (R.J. Samuel); 0000-0001-8259-4903 (P.P. Shevkar); 0000-0001-9922-9742 (M. Bode); 0000-0002-1669-4188 (J.D. Scheel); 0000-0002-3943-6827 (K.R. Sreenivasan); 0000-0002-1359-4536 (J. Schumacher)

Ra	N_e	p	grid-points	N_{BL}	dt	Δt	N_{GPUs}
10^5	$100\times100\times64$	5	80 million	71	1.33×10^{-2}	1000	24
10^6	$100\times100\times64$	7	220 million	57	5.73×10^{-3}	1000	48
10^7	$100\times100\times64$	9	467 million	42	2.45×10^{-3}	1000	100
10^8	$150\times150\times96$	7	740 million	24	1.82×10^{-3}	1000	160
10^9	$150\times150\times96$	9	1.6 billion	15	9.49×10^{-4}	400	240
10^{10}	$400\times400\times200$	7	11 billion	16	4.59×10^{-4}	100	1000
10^{11}	$500\times500\times256$	7	22 billion	16	2.73×10^{-4}	100	1440
10^{12}	$500\times500\times256$	9	46.7 billion	10	2.02×10^{-4}	40	3360

Table 1
Details of the simulations.

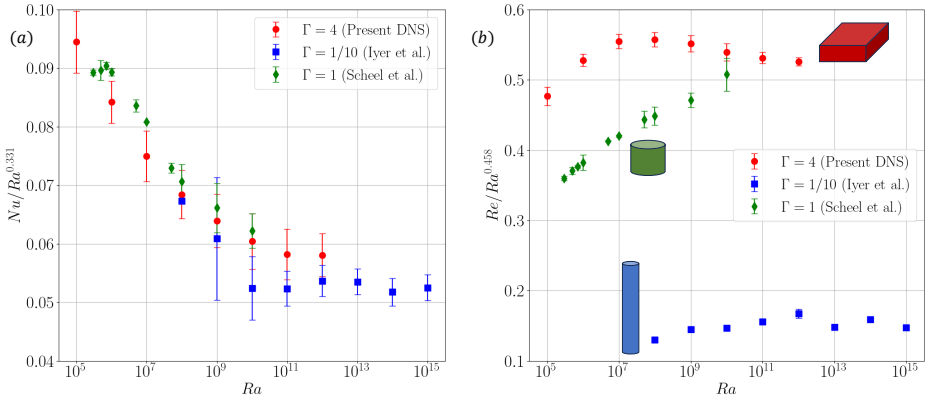


Figure 1: Scaling of Nu and Re with Ra.

air) for all our simulations, but vary Ra from 10^5 to 10^{12} (see Table 1).

The simulations are performed in a Cartesian box of length (L), width (W) and height (H) defined as $L = W = \Gamma H$, where Γ is the aspect ratio of the box. We use a constant $\Gamma = 4$ in all our simulations [4]. The top and bottom walls are isothermal, no-slip plates maintained at $T = 0$ and 1 respectively, whereas periodic boundary conditions are imposed on the sides.

Furthermore, the boundary layers (BLs) become thinner as Ra increases, which demands larger grids to resolve the BLs properly. Consequently the grid sizes also increase with Ra, going up to nearly 47 billion points at $Ra = 10^{12}$. Even at this Ra, we maintain 10 collocation points in the BL (denoted by N_{BL}). Table 1 also lists the number of spectral elements, N_e, polynomial order, p, average time-step, dt, total free-fall times in steady state, Δt, and the number of GPUs required, N_{GPUs}.

We use the spectral element method (SEM) [5, 6] to solve the governing equations. Specifically, we use NekRS [7], which is a GPU accelerated implementation of SEM, derived from the renowned Nek5000 solver. We performed our simulations on the JUWELS Booster at Jülich, utilizing almost the full-cluster (840 nodes) for the $Ra = 10^{12}$ case. This case was run for nearly 30 hours over 4 reservations, using more than 1.1 million GPU core hours.

3. Results and Discussion

Two quantities of key interest in simulations of RBC are NUSSELT number (Nu) and REYNOLDS number (Re). The rate of convective heat flux across the top and bottom plates is quantified by Nu, whereas Re is a measure of momentum transport by the convective flow. The scaling laws for both Nu and Re with respect to Ra are subjects of continuous investigation [8, 9]. Figure 1 shows the scaling of Nu (a)

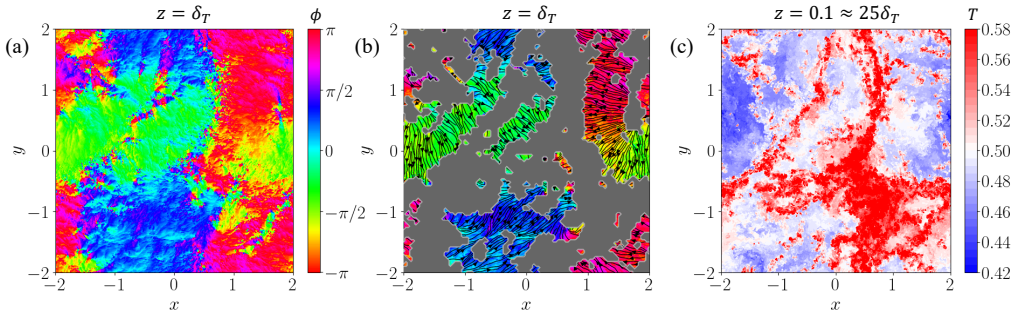


Figure 2: Structure of boundary layer flow and plumes at $Ra = 10^{10}$.

and Re (b) obtained from our simulations, compared against previous results from literature [10, 11].

The Nu scaling seems to collapse well with the classical $1/3$ scaling for $Ra \geq 10^{11}$, whereas the Re scaling shows a significant geometry dependence. The previous works of [10] and [11] used cylindrical domains of aspect ratios $1/10$ and 1 respectively, and their representative shapes are drawn in the plot for clarity.

In Fig. 2, the BL flow and structure of plumes are shown near the bottom plate for $Ra = 10^{10}$. The thermal BL thickness for this case is $\delta_T = 3.72 \times 10^{-3}$. Panel (a) shows the angle of the flow, $\phi \in [-\pi, \pi]$, in the plane parallel to the plate. Regions with smooth color variations denote patches of coherent shear flow, whereas the areas with grainy fluctuations are decoherent regions of plume upwelling. In (b), the plume dominated regions are isolated by thresholding the planar velocity, $u_h = (u_x^2 + u_y^2)^{1/2}$. The grey regions indicate areas with $u_h < U_{\text{rms}}^h$, where U_{rms}^h is the root-mean-square value of u_h averaged over the whole plane. Finally the temperature field at $z = 25\delta_T$ plotted in (c) highlights the alignment between plumes and the decoherent patches on the plate.

Furthermore, we investigated the mean velocity profiles and compared them with Blasius profile to probe for signatures of flat-plate BL flow. We also compared the different measures of momentum and thermal BL thicknesses. Significantly, we do not observe a coherent large-scale circulation typically seen in RBC experiments in closed cylindrical cells of aspect ratio 1 and smaller. This has potential implications on our understanding of the boundary layers dynamics in RBC. Further details and analysis can also be found in [4].

Acknowledgements

The work of RJS is funded by the European Union (ERC, MesoComp, 101052786). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. The work of JDS was supported by a Mercator Fellowship of the Deutsche Forschungsgemeinschaft within the Priority Programme DFG-SPP 1881 on Turbulent Superstructures. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (https://www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[1] E. N. Lorenz, Deterministic nonperiodic flow, Journal of Atmospheric Sciences 20 (2) (1963) 130 – 141. doi:10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[2] F. Chillà, J. Schumacher, New perspectives in turbulent Rayleigh-Bénard convection, The European Physical Journal E 35 (7) (Jul. 2012). doi:10.1140/epje/i2012-12058-1.

[3] M. K. Verma, Physics of Buoyant Flows, WORLD SCIENTIFIC, 2018. doi:10.1142/10928.

[4] R. J. Samuel, M. Bode, J. D. Scheel, K. R. Sreenivasan, J. Schumacher, No sustained mean velocity in the boundary region of plane thermal convection, Journal of Fluid Mechanics 996 (2024) A49. doi:10.1017/jfm.2024.853.

[5] A. T. Patera, A spectral element method for fluid dynamics: Laminar flow in a channel expansion, Journal of Computational Physics 54 (3) (1984) 468–488. doi:10.1016/0021-9991(84)90128-1.

[6] M. O. Deville, P. F. Fischer, E. H. Mund, High-order methods for incompressible fluid flow, Vol. 9, Cambridge university press, 2002. doi:10.1017/CBO9780511546792.

[7] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus,

N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[8] W. V. R. Malkus, The heat transport and spectrum of thermal turbulence, Proceedings of the Royal Society. Series A. Mathematical, Physical and Engineering Sciences 225 (1161) (1954) 196–212. doi:10.1098/rspa.1954.0197.

[9] R. H. Kraichnan, Turbulent Thermal Convection at Arbitrary Prandtl Number, The Physics of Fluids 5 (11) (1962) 1374–1389. doi:10.1063/1.1706533.

[10] K. P. Iyer, J. D. Scheel, J. Schumacher, K. R. Sreenivasan, Classical 1/3 scaling of convection holds up to Ra = 10 15, Proceedings of the National Academy of Sciences 117 (14) (2020) 7594–7598. doi:10.1073/pnas.1922794117.

[11] J. D. Scheel, J. Schumacher, Predicting transition ranges to fully turbulent viscous boundary layers in low Prandtl number convection flows, Physical Review Fluids 2 (12) (2017). doi:10.1103/physrevfluids.2.123501.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Portable Linear Solvers for High-Order Spectral Element Methods on GPUs

Yu-Hsiang M. Tsai[a], Gregor Olenik[a,b], Andreas Herten[c], Mathis Bode[c] and Hartwig Anzt[a,*]

[a]Technical University of Munich (TUM), School of Computation, Information and Technology, Bildungscampus 2, 74076 Heilbronn, Germany
[b]Karlsruhe Institute of Technology (KIT), Scientific Computing Center (SCC), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
[c]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre (JSC), Wilhelm-Johnen-Straße, 52428 Jülich, Germany

ARTICLE INFO[†]

Keywords:
Spectral Element Methods;
Graphics Processing Unit Offloading;
Linear Solver;
Platform Portability

ABSTRACT

The diversification in hardware architectures has become a challenge for computational science: software stacks implemented for a specific hardware architecture often fail to port to other systems. To counter this problem, simulation software stacks increasingly rely on portability layers or software stacks that feature backends for different hardware architectures. We present Ginkgo, a math library that takes platform portability as a central design principle and can be used for numerical calculations in the nekRS CFD code. A runtime and scalability analysis for CFD applications demonstrates that Ginkgo enables platform portability at high performance and scalability.

1. Introduction

In the latest TOP500 list[1] ranking the fastest supercomputers, only one of the fastest ten systems does not feature GPUs. But while the trend incorporating GPU accelerators into the HPC systems is universal, the community still struggles to agree on a universal programming language to implement software for the GPUs of the different vendors. Instead, each vendor supports their own programming ecosystem, often hindering or even blocking portability of software stacks. At the same time, it is impossible for scientific software stacks to develop and maintain multiple versions targeting different hardware architectures. Simulation software stacks currently implement two strategies to tackle the hardware diversification: the use of a portability layer and the use of portable software components. The use of a portability layer requires the rewrite of the simulation software in a portable programming language that can then be compiled for different hardware architectures. Examples for portability layers are the SYCL abstraction[2], Kokkos[3], RAJA[4], and OCCA[5], cf. [1]. Using a portable language enables execution on the architectures that support the portability layer. A disadvantage is that a portability layer has to trade specialization against generalization, and thus can never exploit the latest features of a specific hardware that are not available on other hardware. Hence, a portability layer typically pays some performance penalty to enable the execution on a wide range of hardware. The second approach is to utilize software components that are more limited in functionality, i.e., can not be used for the complete simulation code, but provide high performance on a range of architectures. A viable strategy is to identify the computationally most expensive building blocks in the simulation code, e.g., the numerical computations, and rely on specialized platform-portable libraries for these building blocks. Obviously, the two strategies can also be used in combination: a portability layer enables the execution on a wide range of hardware, a dedicated library ensures high performance for the most expensive building blocks of the simulation code. We here present how the Ginkgo math library can be used to accelerate computational fluid simulations on GPU-accelerated systems. In particular, we demonstrate that the nekRS CFD solver can benefit from using Ginkgo for solving the underlying linear equations by new solver options and platform portability.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02503 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
yu-hsiang.tsai@tum.de (Y.M. Tsai); gregor.olenik@tum.de (G. Olenik); a.herten@fz-juelich.de (A. Herten); m.bode@fz-juelich.de (M. Bode); hartwig.anzt@tum.de (H. Anzt)
ORCID(s): 0000-0001-5229-3739 (Y.M. Tsai); 0000-0002-0128-3933 (G. Olenik); 0000-0002-7150-2505 (A. Herten); 0000-0001-9922-9742 (M. Bode); 0000-0003-2177-952X (H. Anzt)

[1]Top500 as of 06/2024 https://www.top500.org/lists/top500/2024/06/

[2]SYCL https://www.khronos.org/sycl/
[3]Kokkos https://kokkos.github.io
[4]RAJA https://raja.readthedocs.io
[5]OCCA https://github.com/libocca/occa

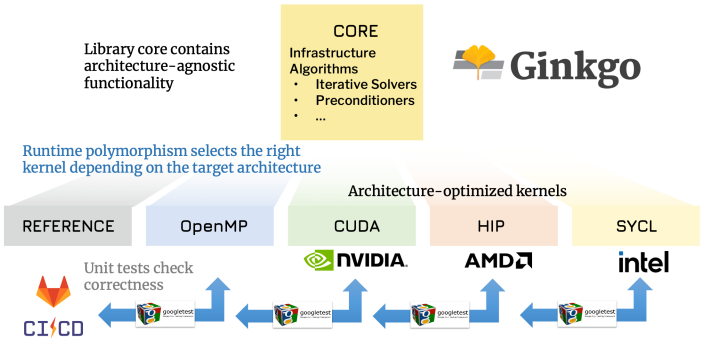


Figure 1: Overview of the Ginkgo library design using the backend model for platform portability [2].

Listing 1: Using Ginkgo as the coarse solver and set the ginkgo solver from the config.json file.

```
[PRESSURE]
preconditioner = multigrid
coarseSolver = ginkgo
[GINKGO]
configFile = config.json
```

Listing 2: Configuration file for selecting the numerical methods and configuring the parameters.

```
{
  "type": "solver::Cg",
  "criteria": [
    {"type": "Iteration", "max_iters": 4},
    {"type": "ImplicitResidualNorm",
     "reduction_factor": 1e-4,
     "baseline": "initial_resnorm"}
    ]
}
```

2. The nekRS and Ginkgo software packages

The nekRS framework[6] [3] is a CFD solver based on libParanumal[7] [4] and Nek5000[8]. It is implemented in C and C++ including Fortran bindings for Nek5000 and uses an MPI + OCCA approach for parallelization, where OCCA acts like a language translator without overhead associated to it during runtime. Numerically, it relies on the spectral element method (SEM), which makes it well suited for the efficient simulation of turbulence, where the number of grid points grows faster than quadratically with the REYNOLDS number when all flow features must be resolved.

Ginkgo is a software library developed under the US Exascale Computing Project (ECP) that focuses on the efficient handling of sparse linear systems on GPUs. Ginkgo is implemented in modern C++ to accommodate a large number of scientific application codes. The software features multiple backends in hardware-native languages: CUDA for NVIDIA GPUs, HIP for AMD GPUs, and SYCL for Intel GPUs. Ginkgo contains a set of iterative solvers, including Krylov solvers, algebraic multigrid (AMG), and parallel preconditioners that serve as a valuable toolbox for application codes. Ginkgo aims to provide not only functional portability but also performance portability [2]. To achieve this, Ginkgo uses a backend model that lifts portability to the software design level. The idea is to rely on a set of kernels implemented using vendor-native programming models, for each hardware target [2]. These kernels are then used to implement the high-level algorithms. This is reflected in Fig. 1 visualizing the backend model used in the Ginkgo library design[9].

Ginkgo provides a wide range of numerical methods for the solution of sparse linear systems, see [2] for an overview of functionality supported by Ginkgo on different hardware. While there are some solvers and preconditioners known to provide good performance for a wide range of nekRS simulations, it is of interest to test a large variety of methods and configuration parameters to identify the best method for a specific simulation. To enable the easy and fast analysis of a wide range of numerical methods, we enabled the use of configuration files that use JSON to encode solver and parameter configurations, The example configuration in Lst. 1

[6] nekRS https://github.com/Nek5000/nekRS
[7] libParanumal https://github.com/paranumal/libparanumal
[8] Nek5000 https://github.com/Nek5000/Nek5000

[9] Ginkgo uses SYCL as one of its backends, which is a portability layer in itself.

Solver	CG(4 iter)	BiCGstab(4 iter)	BiCGstab(20 iter)	GMRES(20 iter)
#coarseSolver	44	41	37	38
runtime[s]	2.03469e-01	3.02990e-01	3.80737e-01	6.32582e-01

Table 1
Performance of different solver settings of GABLS test case with 32 elements per each axis.

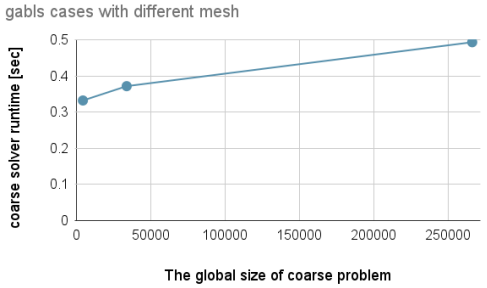
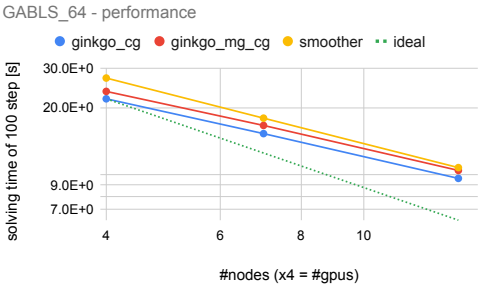
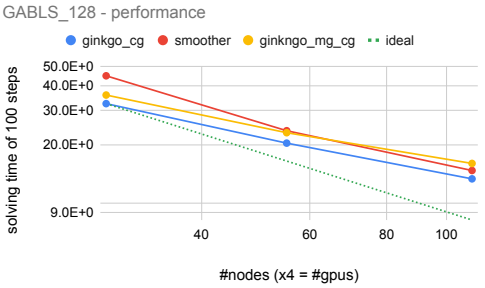


Figure 2: Runtime of the coarse solver on the GABLS test case using different discretizations (16, 32, and 64 elements in each dimension) on 2 machine nodes (8 A100 GPUs/MPI nodes).



(a) The performance of GABLS with 64 elements in each dimension on 4, 7, and 14 physical nodes.



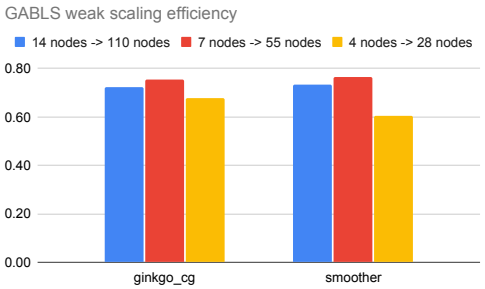
(b) The performance of GABLS with 128 elements in each dimension on 28, 55, and 110 physical nodes.

Figure 3: Performance of GABL: solving time.

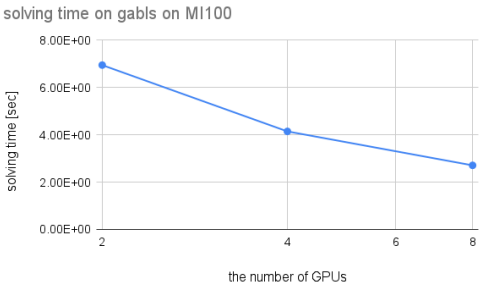
enables ginkgo in nekRS with the configuration file. This concept avoids any compilation of the code, but exclusively works with already compiled libraries. An example for a configuration file is given in Lst. 2. We can try different kinds of solver like CG, BiCGstab, and GMRES or use more iterations in Tab. 1.

3. Preliminary experiments on integration

For testing the correctness, performance and scalability of the integration of the Ginkgo backend, we focus on the GABLS test case that was established by the atmospheric boundary layer community and is an acronym for the GEWEX (Global Energy and Water Cycle Experiment) Atmospheric Boundary Layer Study (GABLS). It is used to quantify the effects of numerical modeling and discretization choices. We initially fix the computational resources to 8 NVIDIA A100 GPUs in two nodes of the Jureca supercomputer and vary the discretization of the unit cube from 16^3 to 64^3. For the different discretizations, we visualize in Fig. 2 the Ginkgo coarse grid solver runtime used in a four level multigrid. We observe an only mild increase in the coarse grid solver runtime for the increasing coarse grid problem sizes. We next investigate the strong and weak scalability of the nekRS simulation using Ginkgo for the numerical computations. We analyze the solving time from 3,000-th time step to 3,100-th time step of the nekRS simulation using the 64^3 discretization on 4, 7, and 14 physical nodes (16, 28, and 56 GPUs respectively) in Fig. 3a and 128^3 discretization on 28, 55, and 110 physical nodes (112, 220, and 440 GPUs respectively) in Fig. 3b with Ginkgo's CG, algebraic Multigrid with CG as coarse solver, and OCCA smoother. In these two cases, CG from ginkgo can get 1.1x speedup against smoother from nekRS. We choose the number of nodes in Fig. 3a and Fig. 3b such that the averaged local sizes are the same between two cases and we can have weak scalability plot in Fig. 4a. We focus the weak scaling efficiency between Ginkgo CG and nekRS smoother in Fig. 4a. Ginkgo CG only performs slightly better weak scalability from GABLS (64^3) on 4 nodes to GABLS (128^3) on 28 nodes than nekRS



(a) The weak scaling efficiency of Ginkgo CG and nekRS smoother between GABLS with 64 and 128 elements in each dimension. We choose the number of nodes between the two cases to make the averaged local size the same.



(b) Solving time of GABLS (16 elements in each dimension) on 2, 4, and 8 AMD MI100 in one machine. It is our self-host machine, so the performance and scaling behavior may be changed in the supercomputer.

Figure 4: Performance of GABLS: weak scaling and solving time on AMD MI100.

smoother, but other cases are quite similar. This preliminary experiments shows that the new ginkgo integration keeps the scalability of nekRS and brings the new solver options to nekRS. Moreover, we run the GABLS with 16 elements in each dimensions on 2, 4, and 8 AMD MI100 GPUs to show the portability of Ginkgo with nekRS in Fig. 4b.

Acknowledgements

This research is supported by the Inno4Scale project under Inno4scale-202301-099. Inno4Scale has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101118139. The JU receives support from the European Union's Horizon Europe Programme. The authors acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputers at Jülich Supercomputing Centre (JSC).

References

[1] W. F. Godoy, P. Valero-Lara, T. E. Dettling, C. Trefftz, I. Jorquera, T. Sheehy, R. G. Miller, M. Gonzalez-Tallada, J. S. Vetter, V. Churavy, Evaluating performance and portability of high-level programming models: Julia, Python/Numba, and Kokkos on exascale nodes (2023). doi:10.48550/arXiv.2303.06195.

[2] T. Cojean, Y.-H. M. Tsai, H. Anzt, Ginkgo—A math library designed for platform portability, Parallel Computing 111 (2022) 102902. doi:10.1016/j.parco.2022.102902.

[3] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[4] N. Chalmers, A. Karakus, A. P. Austin, K. Swirydowicz, T. Warburton, libParanumal: a performance portable high-order finite element library, release 0.5.0 (2022). doi:10.5281/zenodo.4004744.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

NekCRF: A Novel GPU-Accelerated Finite-Rate-Chemistry Solver and Application to Hydrogen

Mathis Bode[a,*], Christos E. Frouzakis[b] and Ananias Tomboulides[c]

[a]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre (JSC), Wilhelm-Johnen-Straße, 52425 Jülich, Germany
[b]ETH Zürich, Sonneggstrasse 3, 8092 Zürich, Switzerland
[c]Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

ARTICLE INFO†

Keywords:
NekRS;
Graphics Processing Unit;
Combustion;
Finite-Rate Chemistry;
Hydrogen

ABSTRACT

This paper presents nekCRF, a GPU-accelerated finite-rate chemistry solver for complex combustion problems. An application to hydrogen is shown and especially the performance compared to CPUs is discussed.

1. Introduction

The nekCRF code follows a low MACH (Ma) approach [1, 2] to solve the resulting system of equations for reactive flows. The conservation equation for the gaseous species mass fractions $Y_k, k = 1, \cdots, N$, temperature T, and velocity $\mathbf{v}$ fields result in

$$\frac{1}{\rho}\frac{\partial Y_k}{\partial t} = \mathbf{v}\cdot\nabla Y_k - \nabla\cdot\rho Y_k\mathbf{V}_k + \dot{\omega}_k \quad k = 1, \cdots, N, \tag{1a}$$

$$\frac{1}{\rho c_p}\frac{\partial T}{\partial t} = -\mathbf{v}\cdot\nabla T + \nabla\cdot\lambda\nabla T + \sum_{k=1}^{N} h_k^0\dot{\omega}_k - \nabla\cdot\rho T\sum_{k=1}^{N} c_{p,k}Y_k\mathbf{V}_k + \frac{dp_0}{dt}, \tag{1b}$$

$$\frac{1}{\rho}\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v}\cdot\nabla\mathbf{v} - \nabla p_1 + \nabla\cdot\mu\left(\underline{\underline{\nabla\mathbf{v}}} + (\underline{\underline{\nabla\mathbf{v}}})^T - \frac{2}{3}(\nabla\cdot\mathbf{v})\underline{\underline{\mathbf{I}}}\right), \tag{1c}$$

$$\nabla\cdot\mathbf{v} = Q_T, \tag{1d}$$

$$p_0 V = nRT, \tag{1e}$$

where t is time, $\mathbf{V}_k, h_k^0, c_{p,k}$ the diffusion velocity, enthalpy of formation and heat capacity of species k, respectively, N the number of the gaseous chemical species, ρ, c_p, λ, μ the mixture density, heat capacity at constant pressure, thermal conductivity and dynamic viscosity, R the ideal gas constant, V volume, and n the total number of moles. The pressure field $p(\mathbf{x}, t) = p_0(t) + \gamma Ma^2 p_1(\mathbf{x}, t)$ is decomposed in the thermodynamic pressure $p_0(t)$, which can only vary in time, and the hydrodynamic pressure $p_1(\mathbf{x}, t)$. Here, $\gamma = c_p/c_v$ is the heat capacities ratio.

Ignoring the Soret and Dufour effects, the species diffusion velocities $\mathbf{V}_k$ are computed by a mixture averaged transport model as

$$\mathbf{V}_k = -\frac{D_k}{X_k}\nabla X_k + \mathbf{V}_c, \tag{2}$$

with X_k being the mole fraction and D_k the mixture-average diffusivity of species k. The correction velocity $\mathbf{V}_c = -\sum_{k=1}^{N} Y_k\mathbf{V}_k$ needs to be introduced [3] to ensure $\sum_{k=1}^{N} Y_k\mathbf{V}_k = 0$, and thus global mass conservation.

The thermal divergence, the non-zero divergence of the velocity field

$$Q_T = \frac{-1}{\rho}\frac{D\rho}{Dt} = \frac{1}{T}\frac{DT}{Dt} + \overline{W}\sum_{k=1}^{N}\frac{1}{W_k}\frac{DY_k}{Dt} + \frac{dp_0}{dt}, \tag{3}$$

with $\overline{W}$ being the mean molecular weight, acts as a constraint and its imposition requires the solution of a variable pressure Poisson equation for the hydrodynamic pressure p_1.

Highly optimized kernels are used to evaluate the costly species production rates as well as the thermodynamic and transport properties. For this purpose, a JIT (just-in-time) kernel generator has been developed, which translates a combustion model (reaction rates, thermodynamic and transport properties) expressed in Cantera format [4] into platform-specific source code.

†This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02504 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
m.bode@fz-juelich.de (M. Bode); cfrouzakis@ethz.ch (C.E. Frouzakis); ananiast@meng.auth.gr (A. Tomboulides)
ORCID(s): 0000-0001-9922-9742 (M. Bode); 0009-0006-0963-4650 (C.E. Frouzakis); 0000-0002-6654-2940 (A. Tomboulides)

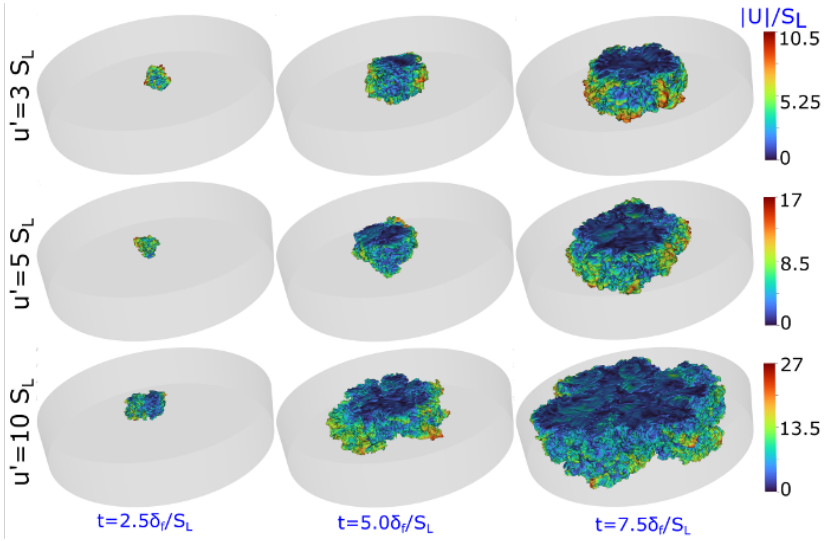


Figure 1: Overview of confined flame kernel simulations at different turbulent intensities. The flame surfaced is colored by the local speed relative to the laminar burning velocity.

2. Application

Flame kernels are important benchmark configurations to better understand complex combustion effects such as thermo-diffusive instabilities. They are often calculated with periodic boundary conditions, but nekCRF also allows to handle confined geometries effectively.

Lean hydrogen flame kernels in confined geometry were calculated under three different turbulent conditions for this work. The turbulent intensity was fixed relative to the laminar burning velocity s_L and varied between $u' = 3, 5, 10 s_L$. The mixture corresponded to $\phi = 0.4$, the ambient temperature was $T_u = 700\,\mathrm{K}$, the wall temperature $T_w = 450\,\mathrm{K}$ and the pressure $p_0 = 2\,\mathrm{atm}$. The mechanism considered 9 species. Figure 1 shows the development of the flame kernels at different times. The effect of confinement is made clear by the selected coloring. In addition, the coloring illustrates the faster expansion speed at higher turbulence. A similar effect can also be observed for the internal heat release rate, which increases with increasing turbulent intensity and is significantly higher than the heat release rate of a laminar flame in all cases.

3. Discussion and conclusions

The application of the lean hydrogen flame kernels demonstrates that nekCRF is capable of performing large-scale simulations of reactive flows. The accuracy of nekCRF [5] was also compared with other codes as well as Cantera (cf. Fig. 2) and showed that the accuracy of the results is in line with the models used. Finally, the performance of nekCRF as a GPU-accelerated code compared to an implementation on CPUs will be discussed. For this purpose, a test case with nekCRF and nek5000 with LAVp (this corresponds to the same models once on GPU and once on CPU) was calculated and the theoretical and practical (due to cache effects and resulting limited throughput) performance compared. Theoretically, nekCRF is up to 22 times faster. Practically still 14 times faster. This means that a simulation for which nek5000 needs two weeks with LAVp can be calculated by nekCRF in one day. This is a significant game

T	H2	O2	H2O	H
1.64e-6	9.48e-6	9.05e-06	9.00E-06	7.35E-04

O	OH	HO2	H2O2	N2
2.46E-03	2.82e-04	1.65e-03	9.33e-06	9.02e-6

Figure 2: Resulting maximum differences between nekCRF and Cantera for a premixed test case.

changer both for the realization of beyond the state-of-the-art simulations and for model development due to much faster iteration times.

Acknowledgment

The authors acknowledge computing time grants for the project TurbulenceSL by the JARA-HPC Vergabegremium provided on the JARA-HPC Partition part of the supercomputer JURECA at Jülich Supercomputing Centre, Forschungszentrum Jülich, the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC), and funding from the European Union's Horizon 2020 research and innovation program under the Center of Excellence in Combustion (CoEC) project, grant agreement no. 952181. Support by the EuroHPC Inno4Scale program acknowledged.

References

[1] B.-T. Chu, L. S. G. Kovásznay, Non-linear interactions in a viscous heat-conducting compressible gas, Journal of Fluid Mechanics 3 (05) (1958) 494. doi:10.1017/S0022112058000148.

[2] R. Rehm, H. Baum, The equations of motion for thermally driven, buoyant flows, Journal of Research of the National Bureau of Standards 83 (3) (1978) 297. doi:10.6028/jres.083.019.

[3] T. Coffee, J. Heimerl, Transport algorithms for premixed, laminar steady-state flames, Combustion and Flame 43 (1981) 273–289. doi:10.1016/0010-2180(81)90027-4.

[4] D. G. Goodwin, H. K. Moffat, I. Schoegl, R. L. Speth, B. W. Weber, Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes, https://www.cantera.org, version 3.0.0 (2023).

[5] S. Kerkemeier, C. E. Frouzakis, A. Tomboulides, P. Fischer, M. Bode, nekCRF: A GPU accelerated high-order reactive flow solver for direct numerical simulations (2024). arXiv:2409.06404.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Large-Scale Engine Direct Numerical Simulations With NekRS: A Multi-Cycle Database

Bogdan A. Danciu[a], Christos Emmanouil Frouzakis[a,*] and Mathis Bode[b]

[a]ETH Zurich, Sonneggstr. 3, CH-8092 Zurich, Switzerland
[b]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany

ARTICLE INFO†

Keywords:
NekRS;
Graphics Processing Unit;
Engine;
Direct Numerical Simulation;
Database

ABSTRACT

This paper presents a collection of large-scale engine direct numerical simulations performed with NekRS on JUWELS Booster. Twelve cycles each for two different engine speeds, 1,500 rpm and 2,500 rpm, of the TU Darmstadt engine were calculated, processed and analyzed.

1. Introduction

Internal combustion engines (ICEs) are still the most widespread mobility drive. Due to their enormous spread, even small improvements can lead to significant global savings. Furthermore, the fight against climate change also requires the use of climate-neutral fuels, such as ammonia for the heavy-duty sector. This requires the fastest possible optimizations, which are not possible without the use of simulations.

Due to the extreme conditions and complex/moving geometries, direct numerical simulations (DNSs) of engines are very challenging. This work has calculated twice twelve cycles of the TU Darmstadt ICE at different engine speeds using NekRS on JUWELS Booster.

2. Case setup and numerics

The direct injection engine at TU Darmstadt is a single, optically accessible cylinder with a pent-roof, four-valve head and an inlet port designed to promote tumble flow. The setup is designed to provide well-defined boundary conditions and reproducible operation. The cylinder of the square engine has a bore of $B = 86$ mm, which is typical for a passenger car engine. Detailed information about the engine and the associated test facility can be found in [1], while the engine operating conditions considered here are listed in Tab. 1.

The multi-cycle DNS was performed using the spectral element solver NekRS/NekCRF [2, 3]. The construction of computational meshes that accurately represent complex ICE geometries is a significant challenge for NekRS due to its requirement for conformal hexahedral meshes. The mesh generation using Coreform Cubit (version 2022.11) involved first filling the cylinder head volume with tetrahedral elements (TETs), which were then divided into four hexahedrons (HEXs) each. The mesh of the lower horizontal plane of the cylinder head was then extruded to the piston to create tensor product element layers capable of accommodating the vertical mesh deformation caused by piston motion while minimizing distortion. The Arbitrary Lagrangian/Eulerian (ALE) formulation [4] was used to account for the mesh deformation resulting from piston motion, with the mesh velocity scaling linearly with the instantaneous piston velocity at the piston and decaying to zero at top dead center (TDC). To compensate for the distortion of the elements as the mesh was compressed, four meshes were constructed with different numbers of spectral elements ranging from $E = 4.8$ to 9.3M. This strategy aimed to maintain good mesh quality throughout the cycle by removing layers as needed. Specifically, the grids were adjusted at different stages: from 600 – 660 crank angle degrees (CAD), $E = 9.3M$ elements were used; from 660 – 690 CAD, the number was reduced

Engine speed [rpm]	Intake p [bar]	Re	No. cycles	OP
1,500	0.95	18,368	12	C
2,500	0.95	30,615	12	E

Table 1
Engine operating conditions (p=pressure; Re =Reynolds number).

†This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02505 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
danciub@eth.ch (B.A. Danciu); cfrouzakis@ethz.ch (C.E. Frouzakis); m.bode@fz-juelich.de (M. Bode)
ORCID(s): 0009-0008-2144-1741 (B.A. Danciu); 0009-0006-0963-4650 (C.E. Frouzakis); 0000-0001-9922-9742 (M. Bode)

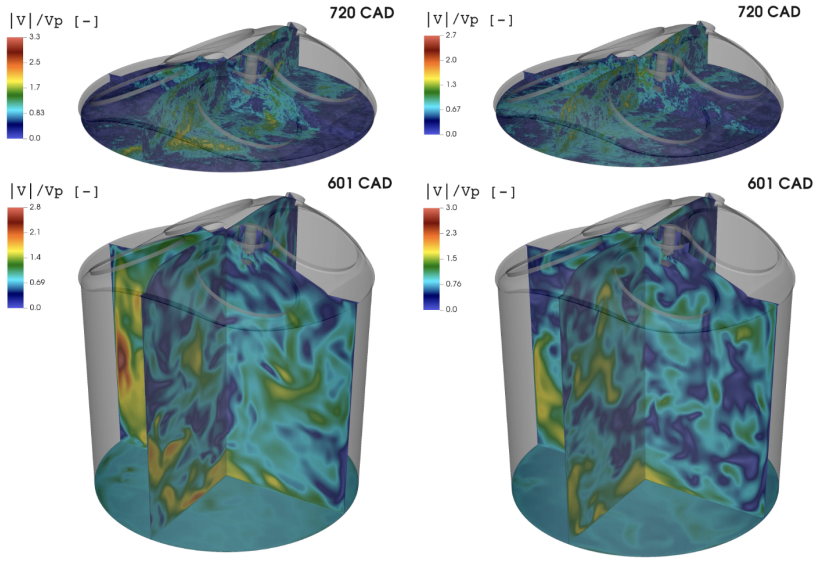


Figure 1: Visualizations of 1,500 rpm (left) and 2,500 rpm (right) for two different CA.

to $E = 6.7M$; from 690 – 710 CAD, it was further reduced to $E = 4.9M$; and from 710–750 CAD, $E = 4.4M$ elements were maintained. A scalable high-order spectral interpolation was used to transition the solution from one grid to the next without compromising the accuracy of the high-order method. Polynomial orders of $N = 7$ and $N = 9$ were chosen for the 1,500 and 2,500 rpm cases, respectively, yielding meshes with 1.5 to 6.8 billion unique grid points. The mesh achieved an average resolution of 30 µm and 23 µm in the bulk, with the first grid point located 3.75 µm and 3 µm away from the wall for 1,500 and 2,500 rpm, respectively.

3. Results and conclusions

Multi-cycle (12 for each engine speed) DNS of a laboratory-scale engine were performed at the practically relevant engine speeds of 1,500 and 2,500 rpm under full-load engine operation (visualizations see Fig. 1. The results confirm previous experimental and numerical findings that the boundary layer (BL) in ICE differs from idealized steady-state turbulent boundary layers, conditions commonly assumed in deriving scaling laws for wall model closures. The large-scale tumble flow generated by the intake process leads to a dynamically changing behavior of the BL, both temporally and locally. Above the piston, the flow undergoes a deceleration-stagnation process, where the tumble vortex hits the piston and then accelerates as the flow diverges from the impact area.

An analysis of the 3D motion of the tumble vortex revealed that the flow also rolls off the cylinder wall, resulting in horizontal vortex structures that are more toward the inlet side at lower heights above the piston and progressively toward the outlet side at higher heights within the cylinder. These structures appear at distances as low as 1 mm from the piston surface during later stages of compression and affect the BL especially at higher engine speeds. As a result, these fluctuations lead to alternating pressure gradients in both the streamwise and spanwise directions, ultimately invalidating the assumption of flow equilibrium. Furthermore, both the BL thickness and the viscous sublayer thickness were found to scale inversely with engine speed and increasing REYNOLDS number Re in the bulk, reaching values as low as 0.41 mm and 13 µm, respectively, at the highest engine speed investigated, posing a significant challenge to both numerical and experimental studies in terms of resolution requirements to properly resolve such BL.

The thermal BL was also found to deviate significantly from ideal scaling laws, even at high engine speeds. Similar to the velocity BL, the thickness of the thermal BL was found to scale inversely with engine speed, but to increase with increasing bulk temperature in the cylinder. In contrast, the thermal displacement thickness, which is sometimes used as

an approximation of the thermal BL thickness, was found to decrease with increasing bulk temperature. Examination of the heat flux distribution confirmed the similarity between the flow and heat flux patterns and revealed areas of increased heat flux, particularly at locations characterized by strong wall-directed flow caused by the tumble vortex impinging on the piston and cylinder head or the horizontal swirl vortices impinging on the cylinder liner. In addition, significant cyclic variations in the surface-averaged wall heat flux were observed for both operating conditions. An analysis of the cyclic tumble ratio revealed that the cycles in which the tumble ratio reaches lower values near TDC, indicating earlier tumble breakdown, also exhibit higher surface-averaged wall heat fluxes.

These first-of-a-kind simulations [5], resulting in one of the largest databases of ICE flows, represent an important step toward the next generation of ICE simulations using GPU-accelerated HPC platforms. This advancement is critical to improve our understanding and optimization of engine performance under various operating conditions with climate-friendly fuels, and to ensure the practical applicability of the developed technologies in real-world engine designs.

Acknowledgements

The authors acknowledge computing time grants for the project TurbulenceSL by the JARA-HPC Vergabegremium provided on the JARA-HPC Partition part of the supercomputer JURECA at Jülich Supercomputing Centre, Forschungszentrum Jülich, the Gauss Centre for Supercomputing e.V.[1] for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC), and funding from the European Union's Horizon 2020 research and innovation program under the Center of Excellence in Combustion (CoEC) project, grant agreement no. 952181.

References

[1] E. Baum, B. Peterson, B. Böhm, A. Dreizler, On The Validation of LES Applied to Internal Combustion Engine Flows: Part 1: Comprehensive Experimental Database, Flow, Turbulence and Combustion 92 (1–2) (2014) 269–297. doi:10.1007/s10494-013-9468-6.

[2] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton, NekRS, a GPU-accelerated spectral element Navier–Stokes solver, Parallel Computing 114 (2022) 102982. doi:10.1016/j.parco.2022.102982.

[3] S. Kerkemeier, C. E. Frouzakis, A. G. Tomboulides, P. Fischer, M. Bode, nekCRF: A next generation high-order reactive low Mach flow solver for direct numerical simulations (2024). arXiv:2409.06404.

[4] L. W. Ho, A legendre spectral element method for simulation of incompressible unsteady viscous free-surface flows, Ph.D. thesis, Massachusetts Institute of Technology (1989). URL http://hdl.handle.net/1721.1/14293

[5] B. A. Danciu, G. K. Giannakopoulos, M. Bode, C. E. Frouzakis, Multi-cycle Direct Numerical Simulations of a Laboratory Scale Engine: Evolution of Boundary Layers and Wall Heat Flux, Flow, Turbulence and Combustion (2024). doi:10.1007/s10494-024-00576-w.

[1] https://www.gauss-centre.eu

Other Topic 1:
Academic Flows

Academic flows have intensively been studied with some cases featuring analytic solutions or experimental evidence. They are hence popular in validating simulation codes, for performing proof-of-concepts, or for understanding the fundamental physics. This session brings together contributions to the *ParCFD International Conference 2024* discussing academic flow cases like the hypersonic flow past an open cavity and particle-bed interaction dynamics in viscous fluids.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Three-Dimensional Parallel Simulations of the Scour Around Multiple Cylinders

Miguel Uh Zapata[a,*], Reymundo Itzá Balam[a] and Damien Pham Van Bang[b]

[a]SECIHTI – Centro de Investigación en Matemáticas A. C., CIMAT Unidad Mérida, 97302, Yucatan, Mexico
[b]Department of Civil and Environmental Engineering, ETS, Université du Québec, 1100 rue Notre Dame Ouest, H3C 1K3, Montreal, Canada

ARTICLE INFO†

Keywords:
Three-dimensional;
Parallel Message Passing Interface;
Large-Eddy Simulation;
Live-Bed;
Scour;
Three Cylinders;
Equilateral-Triangular Configuration

ABSTRACT

The study of flow around single or multiple piles has been conducted extensively due to its significance in both engineering applications and fluid mechanics. In this work, we aim to investigate cases involving multiple piles at different positions and examine how they differ from more extensively studied scenarios, such as a single circular cylinder in a wide channel. Our approach involves solving three-dimensional Navier-Stokes-Exner equations on arbitrary geometries. The numerical formulation is based on a projection method, and the discretization is obtained by combining a second-order unstructured finite-volume method and a sigma transformation. However, three dimensional simulations are computationally expensive, especially when investigating long-term behaviors such as scour around structures. To address this challenge, we introduce a MPI parallelization technique to enhance the Multi-Color Successive Over-Relaxation method using a block domain decomposition. We validate the code by solving the three-dimensional problem and comparing the results against well-established benchmarks featuring complex flows. Specifically in this paper, we focus on an equilateral-triangular configuration involving three circular cylinders.

1. Introduction

Recent research has primarily focused on flow around one or two cylinders, with little attention to more complex arrangements. This paper investigates an equilateral-triangular configuration involving three cylinders, a fundamental configuration in tube arrays and offshore structures [1]. Flow-induced vibrations in cylinders are closely linked to fluctuating forces caused by vortex shedding and flow patterns. Therefore, studying the forces and flow characteristics in this arrangement can enhance understanding of the relationship between fluctuating forces and vortex shedding behavior [2].

While this configuration may appear simple, the investigation and analysis of fluid flow depends on different flow fields upstream and downstream of the cylinders. For instance, the flow patterns and velocity characteristics in an equilateral-triangular arrangement become more complicated compared to a configuration with three cylinders in tandem. This complexity of the problem and the computational effort required for numerical simulations make it rare to find research on this problem. Additionally, the majority of numerical studies were mainly conducted using two-dimensional (2D) models [2, 3]. In recent years, there has been a concerted effort to develop three-dimensional (3D) numerical techniques to study the flow among three cylinders in an equilateral-triangular configuration [4, 5]. Nevertheless, these models are constrained by periodic or slip boundary conditions in the vertical direction, emulating an infinite cylinder configuration. To enhance the comprehension of this phenomenon, the present study introduces 3D parallel simulations using three circular cylinders arranged in an equilateral-triangular configuration under scenarios including both fixed and erodible bed bottoms.

2. Governing equations

This work employs Large-Eddy Simulation (LES) for the hydrodynamic model due to its effectiveness in handling flows around structures [6]. Unlike Direct Numerical Simulation (DNS), which resolves all turbulent scales, LES uses a spatial filter to separate larger turbulent motions, which are explicitly resolved, from smaller, unresolved scales that are modeled separately. The resulting filtered Navier-Stokes equations are:

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_i}{\partial x_i}\right)\right] - \frac{\partial \tau_{ij}}{\partial x_j} + f_i, \quad (1)$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \quad (2)$$

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02506 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
angeluh@cimat.mx (M. Uh Zapata);
damien.pham-van-bang@etsmtl.ca (D. Pham Van Bang)
ORCID(s): 0000-0001-8693-0310 (M. Uh Zapata); 0000-0002-7908-7262 (R. Itzá Balam); 0000-0001-8669-0724 (D. Pham Van Bang)

where x_i ($i = 1, 2, 3$) represents the ith Cartesian coordinate, $\overline{u}_i$ denotes the ith velocity component, $\overline{p}$ is the pressure, ρ stands for the density, ν is the kinematic viscosity, f_i is an external force, and τ_{ij} is the sub-grid scale (SGS) stress tensor that accounts for unresolved length scales. In this paper, the SGS tensor is modeled by $\tau_{ij} = 2\nu_t\overline{S}_{ij} + \frac{1}{3}\tau_{jj}\delta_{ij}$, where δ_{ij} is the Kronecker delta, ν_t is the sub-grid scale turbulent viscosity, and $\overline{S}_{ij}$ is the rate-of-strain tensor for the resolved scale. Using the Smagorinsky model, the eddy viscosity is modeled as $\nu_t = L_s^2|\overline{S}|$, where $|\bar{S}| = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}}$ and L_s is the sub-grid length scale.

The morphological evolution of the sediment bed is described by the Exner equation:

$$(1-\eta)\frac{\partial z_b}{\partial t} + \frac{\partial q_{b_i}}{\partial x_i} = 0, \tag{3}$$

where $\eta = 0.4$ is the porosity of the bed, z_b is the bed elevation, and q_{b_i} ($i = 1, 2$) is the bedload transport rate given by the formula proposed by Engelund and Fredøe: $q_{b_i} = \frac{\pi d}{6} p_{EF} u_{bi}$. Here, d is the particle diameter, p_{EF} is the percentage of particles in motion modeled, and u_{b_i} are the mean velocity components of a sediment particle in movement. For more details on the model, please refer to [6].

3. Numerical method

In the present work, the *Three-dimensional Navier-Stokes Multiphase Flow* (NSMP3D) is utilized as the computational model to solve the hydrodynamics and morphodynamics of the problem. This is a research in-house code developed in FORTRAN. The temporal discretization employs a Crank-Nicholson formulation, and the velocity field and pressure are decoupled using the projection method. NSMP3D is a second-order finite-volume numerical model in both time and space, specifically designed to solve equations (1)-(3) in σ-coordinates across unstructured mesh grids. The transformation from Cartesian to σ-coordinates allows for accurate modeling of vertical domain changes, such as those caused by bed alterations due to erosion. Figure 1 illustrates an example of a physical domain along with its corresponding transformed computational domain. This figure also depicts a three-dimensional prismatic control volume used in our model. For a comprehensive understanding of this numerical model, detailed descriptions can be found in [6, 7].

As a consequence of approximating the volume and surface integrals over prism elements, information about the unknown variable is required at both the center and vertices, see Fig. 1. We address this issue by incorporating an interpolation technique that calculates the value of a vertex point using its surrounding elements. Thus, the finite volume method leads to a linear system of equations

$$A\boldsymbol{\phi}_c + B\boldsymbol{\phi}_v = \mathbf{f}, \tag{4}$$

where A and B are matrices containing the coefficients from the discretization, the vectors $\boldsymbol{\phi}_c$ and $\boldsymbol{\phi}_v$ consist of the unknown variable (pressure or velocity component) at the center and vertex points, respectively, and the vector $\mathbf{f}$ corresponds to the known right-hand side values. The resulting matrix A is a sparse matrix due to both the σ-transformation, unstructured mesh, and the interpolation technique. Furthermore, it is difficult to express the matrix B explicitly and it would require more computer memory. Therefore, instead of $B\boldsymbol{\phi}_v$, we consider a vector $\mathbf{b}_v$ that contains all the values related to the solution at the auxiliary vertex unknowns obtained from the interpolation.

The resulting linear systems are solved using the Multi-Color Successive Over-relaxation Method. The program operates in parallel using domain decomposition and through the Message Passing Interface (MPI). This improvement accelerates the resolution of the resulting sparse linear system governing pressure and velocity components [7].

4. Equilateral-Triangular Configuration

This work employs non-dimensional equations using the diameter of the pile, D, and the mean flow velocity, U, as reference scales for the length and velocity, respectively. This selection of reference scales, allows the governing equations to be expressed in terms of the REYNOLDS number $\mathrm{Re}_D = UD/\nu$, which is based on the cylinder diameter, the mean flow velocity, and kinematic viscosity.

Figure 2 shows the computational domain used in numerical simulations using three circular cylinders in an equilateral triangular configuration. The arrangement of the cylinders follows the distribution presented in [4]. The spacing ratio is defined as $\ell/D = 2$, where ℓ denotes the center-to-center distance between the cylinders, see Fig. 2. The computational domain consist of $10D$ upstream, $20D$ downstream, and $15D$ laterally of the center of this triangular configuration. The x, y, and z axes are aligned with the streamwise, spanwise, and vertical directions, respectively. The water depth is set to be $h = 3D$. The domain is discretized in 25, 332 triangular elements and 13, 021 vertex points in the horizontal direction, with a high-resolution mesh employed in the vicinity of the cylinder. For validation and comparison, we will also conduct numerical simulations using a single cylinder. In this scenario, the domain dimensions are as previously described, and the horizontal mesh contains 25,895 triangular elements and 13,146 vertex points.

The boundary conditions for the flow field, described in Fig. 2, are defined as follows: At the inlet, transverse and

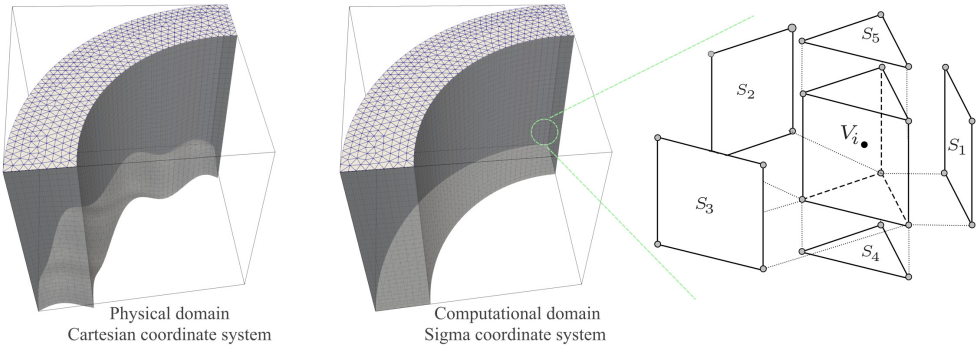


Figure 1: Example of a physical and its corresponding computational domain used in NSMP3D. It also depicts the components of a three-dimensional prismatic control volume employed for the discretization.

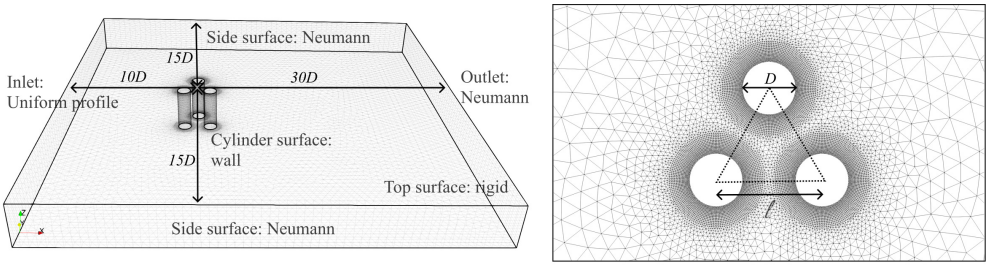


Figure 2: Geometry of the computational domain with specification of the boundary conditions (left), and equilateral-triangular configuration of the three cylinders (right).

vertical velocities are set to zero, and the inflow velocity follows a uniform profile with a unit mean non-dimensional value. At the outlet, zero-gradient conditions are applied to all variables. Neumann conditions are used for the boundary in the spanwise direction. No-slip wall boundary conditions are applied to the cylinder surfaces. While a rigid surface is imposed at the top, the type of boundary condition at the bottom varies depending on the specific test case being investigated: free slip (Neumann), fixed (no-slip wall) or mobile (live-bed scour).

5. Numerical results

In this paper, we initially focus on a free-slip bed simulation with a low REYNOLDS number to validate our results against experimental data and other numerical models, as well as to study the performance of our method. For a single cylinder, it is well-known [8] that at lower REYNOLDS number ($\mathrm{Re}_D < 47$), the flow will form a stable re-circulation bubble behind the cylinder; if Re_D increases up to 200, then a stable 2D vortex shedding is formed. $\mathrm{Re}_D = 200$ is the critical REYNOLDS number for the flow past a single cylinder transitioning from 2D to 3D. For $\mathrm{Re}_D > 1{,}000$, flows become fully 3D.

All the simulations are initially performed using 20 vertical layers for 100 time units (D/U) until statistical stability is achieved, then restarted for another 100 time units to gather the time-averaged data. Under the same conditions, Fig. 3 shows an stage of the instantaneous velocity field around a single and the three-cylinder equilateral arrangement using $\mathrm{Re}_D = 100$. As expected, even at a low REYNOLDS number, the vortices exhibit a complex behavior behind the cylinders. However, it is found that the velocities magnitudes are similar and the flow behind the structures is still 2D.

The time average velocity profile using $\mathrm{Re}_D = 100$ is shown in Fig. 4 for both scenarios. Note that the recirculation length (L/D) differs for each of the three cylinders, and none matches the length observed in the single-cylinder results. To validate the numerical model, simulations of flow past a

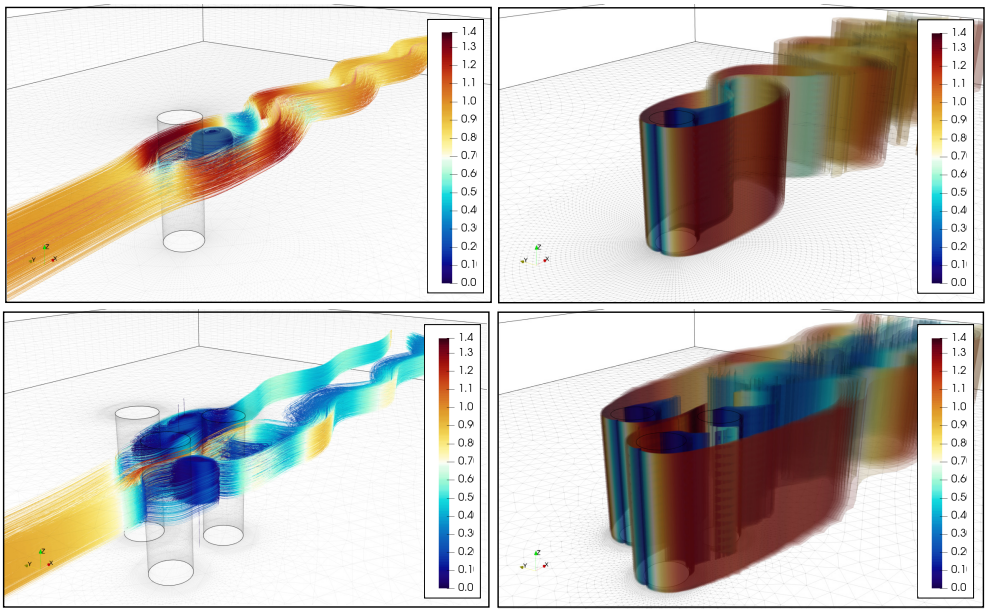


Figure 3: Instantaneous velocity field around the single and the three cylinders arrangement using $\mathrm{Re}_D = 100$: streamlines (left), and vorticity contours (right) colored by the velocity magnitude.

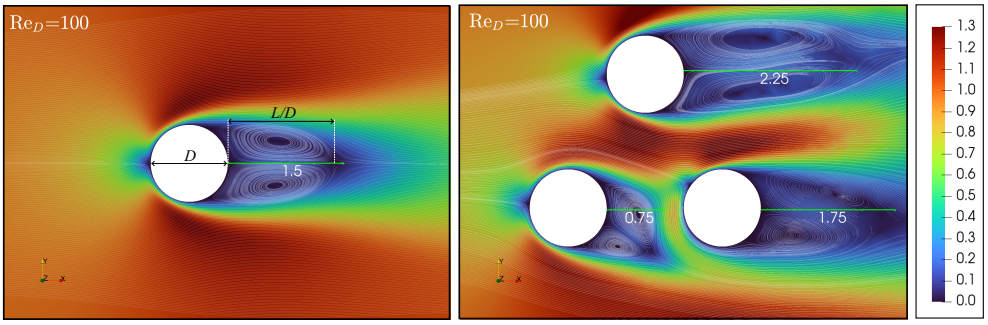


Figure 4: 2D view of the time average velocity using $\mathrm{Re}_D = 100$ for both cylinder configurations.

single cylinder at REYNOLDS numbers from 50 to 200 are performed. The evolution of the recirculation length as a function of the REYNOLDS numbers is presented in Fig. 5. Note that the numerical results for the single cylinder case are close to the experimental measurements of Nishioka and Sato (1974) and the computational results using a high resolution provided by Park et al. (1998) [6].

It is important to emphasize that accurately representing behavior near structures requires a fine mesh resolution, which translates to more mesh elements and higher computational costs. In terms of computational performance, using a mesh resolution of approximately 500,000 3D control volumes, the sequential setup requires about 120 minutes of CPU time to obtain one time unit. However, with a parallel

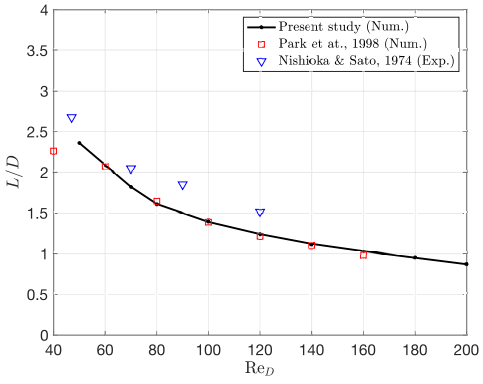


Figure 5: Evolution of the recirculation length as function of the REYNOLDS number for a single cylinder.

formulation using 16 processors, the same calculation is reduced to 10 minutes of CPU time.

While the performance at low REYNOLDS number of our simulations is adequate for short-term studies, erosion test cases require thousands of non-dimensional time units to analyze sediment bed evolution. Additionally, due to the turbulent behavior of the flow, both fixed- and live-bed cylinder test configurations necessitate finer mesh resolution in the horizontal and vertical directions to achieve successful simulations. Initial simulations, although not presented here, have shown that this setup can achieve 1,000 seconds of real-time simulation within one week of CPU time using 64 processors and a mesh resolution of approximately 5 million elements. These results indicate a relatively slow simulation capacity, highlighting the potential for enhancement, particularly speedup improvements employing more processors.

6. Conclusions

This work presents an improved unstructured finite-volume method capable of simulating the flow around multiple piles in parallel. Taking advantage of the applicability of the proposed numerical techniques, we validate our findings with a single-cylinder configuration at low REYNOLDS numbers and extend the model's capabilities by examining the behavior of three cylinders in an equilateral arrangement. Outgoing work includes conducting rigid bed calculations at moderate REYNOLDS numbers to investigate flow characteristics such as horseshoe vortex behavior. Additionally, a live-bed scour simulation at high REYNOLDS numbers will be performed to test the morphological model.

References

[1] Experimental Study on Flow-Induced Motion of an Array of Three Cylinders With Circular, Square, and Diamond Sections, The 28th International Ocean and Polar Engineering Conference.
URL https://onepetro.org/ISOPEIOPEC/proceedings-abstract/ISOPE18/All-ISOPE18/20325

[2] S. Zheng, W. Zhang, X. Lv, Numerical simulation of cross-flow around three equal diameter cylinders in an equilateral-triangular configuration at low reynolds numbers, Computers and Fluids 130 (2016) 94–108. doi:10.1016/j.compfluid.2016.02.013.

[3] Y. Bao, D. Zhou, C. Huang, Numerical simulation of flow over three circular cylinders in equilateral arrangements at low reynolds number by a second-order characteristic-based split finite element method, Computers and Fluids 39 (5) (2010) 882–899. doi:10.1016/j.compfluid.2010.01.002.

[4] Three-dimensional numerical simulation on flow past three circular cylinders in an equilateral-triangular arrangement, Ocean Engineering 189 (2019) 106375. doi:10.1016/j.oceaneng.2019.106375.

[5] M. Zhang, B. Yin, D. Guo, Z. Ji, G. Yang, Numerical study on the flow past three cylinders in equilateral-triangular arrangement at re = 3 × 106, Applied Sciences 12 (22) (2022). doi:10.3390/app122211835.

[6] W. Zhang, M. Uh Zapata, X. Bai, D. Pham-Van-Bang, K. D. Nguyen, Three-dimensional simulation of horseshoe vortex and local scour around a vertical cylinder using an unstructured finite-volume technique, International Journal of Sediment Research 35 (3) (2020) 295–306. doi:10.1016/j.ijsrc.2019.09.001.

[7] M. Uh Zapata, W. Zhang, D. Pham Van Bang, K. D. Nguyen, A parallel second-order unstructured finite volume method for 3d free-surface flows using a σ coordinate, Computers and Fluids 190 (2019) 15–29. doi:10.1016/j.compfluid.2019.06.001.

[8] D. Kim, H. Choi, A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids, Journal of Computational Physics 162 (2) (2000) 411–428. doi:10.1006/jcph.2000.6546.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Hypersonic Flow Past an Open Cavity Using HPC and Open-Source Software

David R. Emerson[a,*], Jian Fang[a] and Benzi John[a]

[a]STFC Daresbury Laboratory, Scientific Computing Department, WA4 4AD, Warrington, UK

ARTICLE INFO[†]

Keywords:
Hypersonic Flow;
Cavity Flow;
ASTR;
SPARTA

ABSTRACT

Simulating hypersonic flow presents many computational challenges and requires use of software that can handle a wide range of external conditions. In general, hypersonic flight occurs at high altitude with flow regimes from the continuum, where the conventional no-slip boundary condition is appropriate, through to the free molecular regime, where the molecules no longer collide. This requires quite distinct software to handle the disparate flow regimes. In this research, we are using an in-house solver, ASTR, which is a high-order finite difference code suitable for continuum problems (altitudes below 70km), and a direct simulation Monte Carlo code, SPARTA, which is suitable for rarefied flows. Both codes are open-source and are very well suited to HPC. We are applying these codes to simulate hypersonic flow (Mach 6.9) past an open cavity (length-to-depth ratio of 2) at altitudes up to 120km.

1. Introduction

Cavities offer a fascinating fluid dynamics problem to study and can involve flow separation and reattachment, quasi-unsteady phenomena, and significant changes to drag and heat transfer. For such a simple geometry, the flow physics associated with cavities is extremely rich. Consequently, cavities have been the subject of many studies, including theoretical [1, 2], experimental [3, 4, 5], and computational [6, 7]. Due to the flow complexity and unsteady motion of the free shear layer, high-performance computing (HPC) is essential to resolve the fluid motion.

In order to separate the different flow characteristics associated with cavities, it is helpful to classify the various configurations using the length, L, depth, D, as illustrated in Fig. 1, and the length-to-depth ratio, L/D. There are essentially two main cavity configurations corresponding to "open" and "closed" cavity flows. In an open cavity, the flow separates and reattaches to the downstream face. Open cavities often suffer from resonance effects due to the shear layer oscillating. Conversely, a closed cavity occurs when the shear layer separates and reattaches to the cavity floor before impinging the downstream face. This creates two distinct recirculating regions with a separation wake forming behind the upstream face, and a recompression wake formed before the downstream face. No single value of L/D provides a demarcation between the two types and it has been found experimentally that a hysteresis region exists between the two states. It is also important to distinguish between "deep" and "shallow" cavities because the flow features can be quite different. Cavities with $L/D < 1$ are referred to as deep cavities. In contrast, when $L/D > 1$ they are referred to as shallow.

In this study, we are interested in hypersonic flow past both deep and shallow cavities and at all altitudes. In particular, to simplify the analysis, we will focus on an open cavity configuration with $L/D = 2$. In the next sections, we will briefly outline the software used and present some recent results. All simulations have been performed on ARCHER2, the UK's national supercomputer[1] which is a HPE Cray system.

2. Numerical methods

2.1. Continuum flow

The approach we have adopted for this work is to use Direct Numerical Simulation (DNS) to solve the Navier-Stokes equations. This is based on an in-house computational fluid dynamics (CFD) code, ASTR, employing a high-order finite difference methodology. A seventh-order low-dissipative monotonicity-preserving (MP7-LD) scheme proposed by Fang et al. [8] is used to capture shock-waves in high-speed flows, and a sixth-order central scheme is adopted to solve the diffusion terms in the Navier-Stokes equations. After the spatial terms are solved, a three-step third-order total variation diminishing Runge–Kutta method is used for the temporal integration. The code has been

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02507 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
david.emerson@stfc.ac.uk (D.R. Emerson); jian.fang@stfc.ac.uk (J. Fang); benzi.john@stfc.ac.uk (B. John)
ORCID(s): 0000-0002-6085-5049 (D.R. Emerson); 0000-0003-2474-1762 (J. Fang); 0000-0003-4130-5508 (B. John)

[1]https://www.archer2.ac.uk/

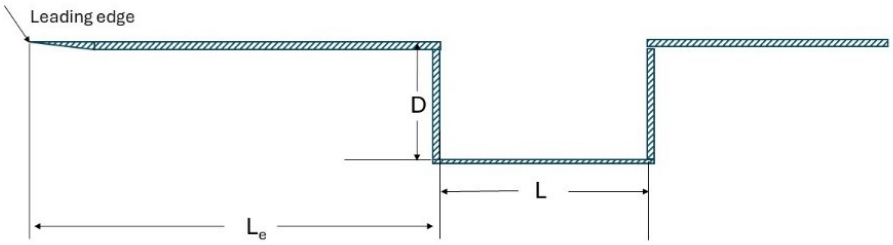


Figure 1: Schematic of generic cavity (not to scale).

applied and validated for a series of high-speed flows [9, 10].

2.2. Rarefied flow

DSMC is a particle method based on kinetic theory for the simulation of dilute gases. The discrete molecular nature of a gas is taken into account by tracking the evolution and collisions of particles that represent a collection of real gas molecules. Being a molecular-level method, DSMC is valid for the entire range from continuum to the free-molecular regime. A detailed explanation of the steps involved in the DSMC method can be found in [8]. Previous studies by John et al. [11, 12, 13] have demonstrated the method's capabilities on a range of problems, including thermal aspects in a lid-driven cavity. All DSMC simulations in this work have been carried out using SPARTA, which is a highly scalable parallel open-source DSMC code by Plimpton et al. [14].

2.3. Initial conditions

The open cavity configuration is presented in Fig. 1 where the upstream (*i.e.*, left-hand) boundary is defined as the inflow, at which freestream flow conditions are imposed. The geometric parameters and freestream MACH number follow the experiment of Wieting [3]. Specifically, here we consider a length to depth ratio of 2 and a freestream MACH number of 6.9. The total (stagnation) temperature considered is 1650K, whereas the wall temperature is fixed at 294K. For continuum flows, the classic no-slip boundary condition is applied at the wall. In the case of high-altitude flight, any variation in KNUDSEN number (Kn) is achieved by changing the flow density conditions, i.e. through the reference pressure, P_∞, at which the simulation is carried out. The mean free path, λ, is defined as in Eq. (1),

$$\lambda = \frac{\mu}{P_\infty}\sqrt{\frac{\pi R T_\infty}{2}}, \qquad (1)$$

where μ is the viscosity of the gas based on the reference temperature, T_∞ , and R is the specific gas constant. In addition to continuum flow ($Kn = 0$), four KNUDSEN numbers have been considered, *i.e.*, $Kn = 0.01$, $Kn = 0.1$, $Kn = 1$, and $Kn = 100$, which roughly corresponds to atmospheric altitudes of H=50km, H=65km, H=80km and H=115km, for the cavity geometry considered in the present work. All simulations are laminar.

3. Parallel performance

For this particular problem, we make use of the UK's national HPC facility, ARCHER2 (`https://www.archer2.ac.uk/`). This is a HPE Cray EX machine using the AMD EPYC 7742 2.25 GHz processor with each node having a dual-CPU arrangement and 128 cores per node. The full system configuration has 750,080 cores and uses the HPE Cray Slingshot network. Both codes make extensive use of this facility to model a variety of problems. A direct comparison of each code's performance is not considered due to the very different nature of solution strategy required by each code.

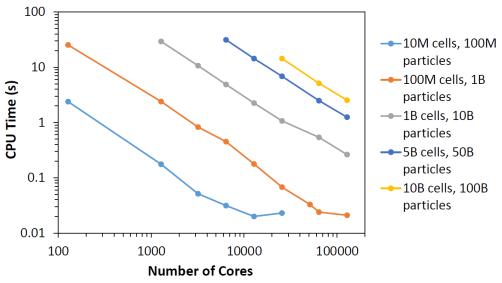


Figure 2: Weak scalability plot for SPARTA up to 128,000 cores (1,000 nodes).

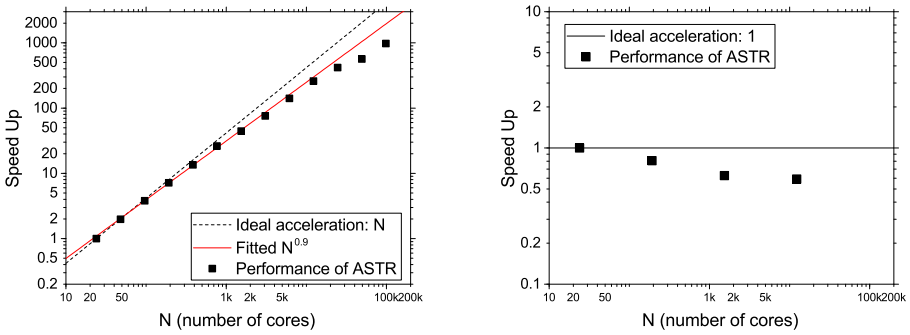


Figure 3: Strong (left) and weak (right) scalability of ASTR code on ARCHER2.

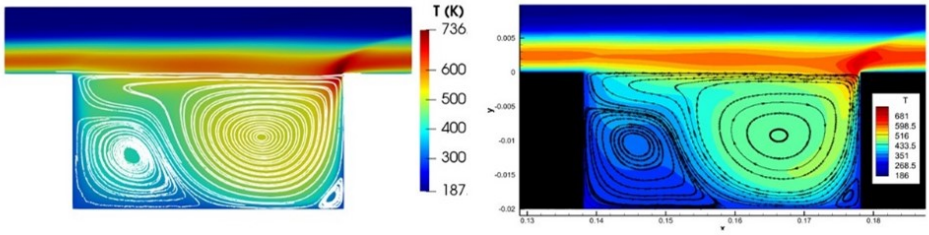


Figure 4: Comparison between continuum (right) and DSMC (left) at $Kn \sim 0.001$.

3.1. Parallel performance of SPARTA

The DSMC code, SPARTA, is a well-established code for simulating rarefied gas dynamics and has been featured in the Exascale Computing Project and used on a wide range of architectures. Here, we provide some results for a typical scalability study using ARCHER2 in Fig. 2. The scalability study considered flow in a 3D closed box with reflective wall boundaries and range from very large simulations with 10 billion (B) cells and 100 billion particles to relatively smaller simulations with 10 million (M) cells and 100 million particles. The code performs excellently with both increase in load and increase in number of cores for all the cases considered.

3.2. Parallel performance of ASTR

The DNS code, ASTR, is under development in the group and has been used to study a wide range of high-speed flow problems. A strong scalability is test for a 1,0243 Taylor- Green vortex case on ARCHER2. The speed-up is plotted in Fig. 3 (left), and a linear acceleration can be roughly maintained up to 24,576 cores. The weak scalability is also tested for the Taylor-Green vortex case. The test is conducted by fixing the number of grid nodes on each core, and the data is reported in Fig. 3 (right). However, for the present 2D cases, only a small number of computing cores (8×128) are required.

4. Results and brief conclusions

Two open-source codes, ASTR and SPARTA, have been used to simulate hypersonic flow past an open cavity with $L/D = 2$. Due to the computationally intense nature of unsteady cavity flow, parallel computing was essential to capture the complex flow features.

Figure 4 illustrates a validation test that compares both ASTR and SPARTA at very low KNUDSEN number, corresponding to continuum flow. The results show very good agreement between the two approaches. For very low KNUDSEN numbers (i.e. near-continuum regime), DSMC simulations are computationally very expensive mainly due to the requirement of cell sizes to be a fraction of the mean free path, which results in an excessively large number of cells and particles. The DSMC simulations for $Kn = 0.001$

utilized 250 nodes (32,000 cores) on ARCHER2 with a compute time of about 8 hours. The corresponding CFD solution at the same KNUDSEN number utilized only 8 nodes (1,024 cores) with a compute time of about 12 hours. More detailed results will be presented at the conference that investigates the unsteady flow features and the impact of rarefaction on the flow physics.

5. Acknowledgements

The authors would like to thank the UK Turbulence Consortium (UKTC) for providing access to the ARCHER2 supercomputer under EPSRC grant EP/X035484/1. We gratefully acknowledge support from the Computational Science Centre for Research Communities (CoSeC) through CCP Turbulence under EPSRC grant EP/T026170/1.

References

[1] D. R. Chapman, A theoretical analysis of heat transfer in regions of separated flow, Tech. rep. (1956).

[2] O. R. Burggraf, A model of steady separated flow in rectangular cavities at high reynolds number, in: Proc. Heat Transfer and Fluid Mech. Inst, 1965, pp. 190–229.

[3] A. R. Wieting, Experimental investigation of heat-transfer distributions in deep cavities in hypersonic separated flow, National Aeronautics and Space Administration, 1970.

[4] R. L. Stallings Jr, F. J. Wilcox Jr, Experimental cavity pressure distributions at supersonic speeds, Tech. rep. (1987).

[5] K. M. Casper, J. Wagner, S. J. Beresh, J. F. Henfling, R. W. Spillers, B. O. M. Pruett, Complex geometry effects on supersonic cavity flows., Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States) (2014).

[6] C. Borland, Numerical prediction of the unsteady flow field in an open cavity, in: 10th Fluid and Plasmadynamics Conference, 1977, p. 673.

[7] D. P. Rizzetta, Numerical simulation of supersonic flow over a three-dimensional cavity, AIAA journal 26 (7) (1988) 799–807.

[8] J. Fang, Z. Li, L. Lu, An optimized low-dissipation monotonicity-preserving scheme for numerical simulations of high-speed turbulent flows, Journal of Scientific Computing 56 (1) (2012) 67–95. doi:10.1007/s10915-012-9663-y.

[9] J. Fang, F. Gao, C. Moulinec, D. Emerson, An improved parallel compact scheme for domain-decoupled simulation of turbulence, International Journal for Numerical Methods in Fluids 90 (10) (2019) 479–500. doi:10.1002/fld.4731.

[10] J. Fang, A. A. Zheltovodov, Y. Yao, C. Moulinec, D. R. Emerson, On the turbulence amplification in shock-wave/turbulent boundary layer interaction, Journal of Fluid Mechanics 897 (Jun. 2020). doi:10.1017/jfm.2020.350.

[11] B. John, X.-J. Gu, D. Emerson, Investigation of heat and mass transfer in a lid-driven cavity under nonequilibrium flow conditions, Numerical Heat Transfer, Part B: Fundamentals 58 (5) (2010) 287–303. doi:10.1080/10407790.2010.528737.

[12] B. John, X.-J. Gu, D. R. Emerson, Nonequilibrium gaseous heat transfer in pressure-driven plane poiseuille flow, Physical Review E 88 (1) (Jul. 2013). doi:10.1103/physreve.88.013018.

[13] B. John, X.-J. Gu, R. W. Barber, D. R. Emerson, High-speed rarefied flow past a rotating cylinder: The inverse magnus effect, AIAA Journal 54 (5) (2016) 1670–1681. doi:10.2514/1.j054782.

[14] S. J. Plimpton, S. G. Moore, A. Borner, A. K. Stagg, T. P. Koehler, J. R. Torczynski, M. A. Gallis, Direct simulation monte carlo on petaflop supercomputers and beyond, Physics of Fluids 31 (8) (Aug. 2019). doi:10.1063/1.5108534.

Other Topic 2:

Aerospace

This session highlights recent advancements in aerospace computational fluid dynamics, with a focus on high-fidelity simulations and turbulence modeling. Presentations cover RANS computations of 3D flows around JAXA and NASA models using various turbulence models, and the study of flow around a rigid oscillating airfoil. Furthermore, the HEMLAB algorithm's applications to delta wing geometries and the 7th AIAA CFD Drag Prediction Workshop are discussed.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Application of the HEMLAB Algorithm to a Delta Wing Geometry and a 5th Generation Fighter Model

Huseyin Akgun[a,*], Mehmet Sahin[a]

[a]*Istanbul Technical University, Aeronautical and Astronautical Engineering, Maslak, 34469 Istanbul, Turkey*

ARTICLE INFO[†]

Keywords:
Parallel Computational
Fluid Dynamics;
Delta Wings;
Anisotropic Mesh Adaptation;
Finite-Volume Method

ABSTRACT

This work presents delta wing simulations using two different geometries: Delta wing geometry provided in the International Workshop on High-Order CFD Methods and the Sydney Standard Aerodynamic Models (SSAM) for a generic fifth-generation high-performance aircraft. This study aims to accurately compute the lift and drag coefficients of each model and to generate low speed, high angle of attack flows for delta wings in order to obtain a further insight into vortex dominated flow physics. The calculations are carried out utilizing a vertex based finite-volume algorithm HEMLAB which is combined with an anisotropic mesh adaption library pyAMG from INRIA in order to further increase its computational efficiency. The main purpose of this study is to validate the HEMLAB solver and use the anisotropic mesh adaptation method with different adaptation functions to get more accurate solutions with optimized mesh resolution in necessary regions.

1. Introduction

The flow study of delta wings have been a challenge due to different flow phenomena. The sharp edge at the front of the body, high sweep angles and high angle of attacks make the study even more challenging. Delta wings flow behavior includes several vortex regimes, separations bubbles, sharp edge/corner problems which are the parts of the flow that have to be examined. Therefore, the numerical calculations has been carried out utilizing the HEMLAB algorithm [1].

The HEMLAB solver is an in house compressible Reynolds averaged Navier-Stokes (RANS) solver with an efficient edge-based (quad- and half-edge) data structure designed for a vertex based finite-volume method algorithm on hybrid unstructured meshes in two- and three-dimensions. The algorithm can use several inviscid flux schemes such as Roe, AUSM+up and HLLC. The unweighted least squares reconstruction is used for the inviscid fluxes, meanwhile the Green-Gauss approach is used for the viscous fluxes. The classical negative Spalart-Allmaras (SA-noft2) turbulence model is used for turbulence modeling. The algorithm uses a fully coupled nonlinear Newton method based on matrix-free approach within the PETSc-3.21.0 library (PETSc-SNES). The algorithm is integrated with the anisotropic mesh adaptation library pyAMG from INRIA[1].

The main goal if this study is to validate the recent version of the HEMLAB solver applying on a simple and complex delta wing case and optimizing the mesh resolution with the anisotropic mesh adaptation for better solution time and accuracy. By using the mesh adaptation method instead of generating the mesh field according to the guess of the flow behavior, the method uses the past solution to upgrade and optimize the new mesh resolution in necessary regions.

2. Delta wing cases

2.1. International Workshop of High Order CFD Delta Wing

The first case is a low REYNOLDS number high angle of attack case for the initial simulations. The case was defined and detailed in the reference [2]. The case execution details were explained in detail at the reference [3]. The case used different mesh adaptation norm (L2 and L4), different inviscid flux interpolations (second and third), and different blending parameters between upwind methods and central schemes. The simulation parameters are shown in Tab. 1.

Simulation	Mesh - AN	Beta (β)	Upwind LS
1	Mesh 1-2	1	2nd
2	Mesh 2-4	1	2nd
3	Mesh 2-4	1/3	2nd
4	Mesh 2-4	1	3rd

Table 1
Simulation parameters including the adaptation norm (AN), the β value, and the upwind Least Squares (LS) order.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02508 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
akgunh17@itu.edu.tr (H. Akgun); msahin.ae00@gtalumni.org (M. Sahin)
ORCID(s): - (H. Akgun); 0000-0001-6502-2209 (M. Sahin)

[1]pyAMG https://pyamg.saclay.inria.fr/pyamg.html

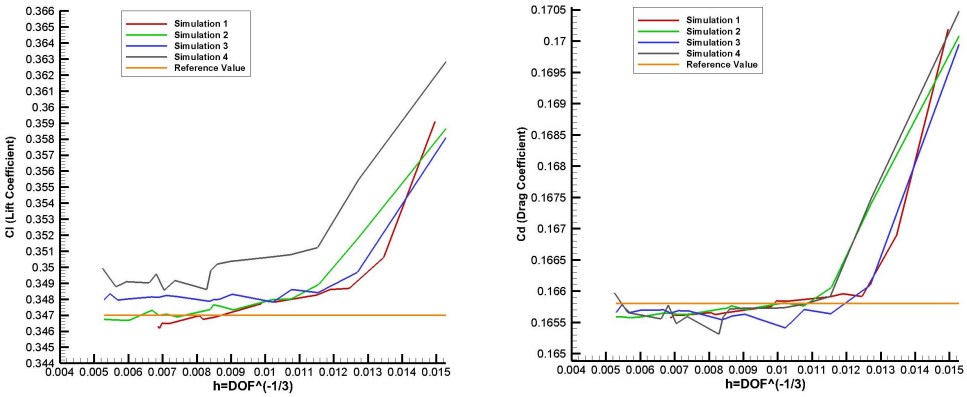


Figure 1: Lift (left) and drag (right) coefficient values for each simulation with vs mesh size h values compared with the reference values.

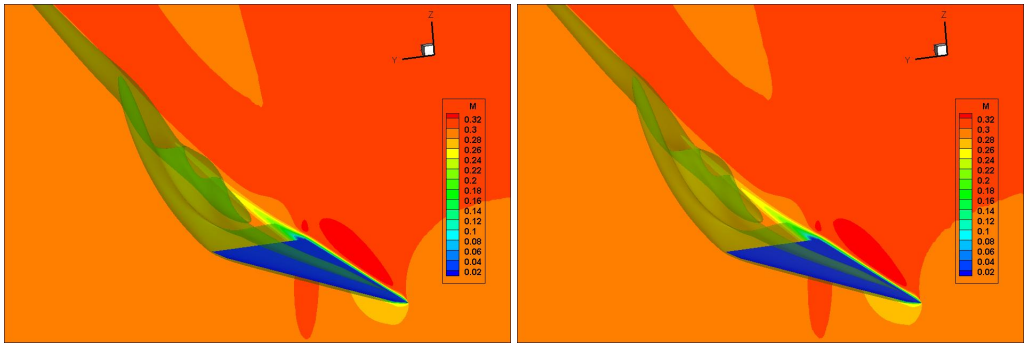


Figure 2: Simulation 1 (left) and simulation 3 (right) laminar delta wing with slice (y = 0) contour of MACH number and an isosurface of 0.2 MACH.

Figure 1 shows the lift and drag coefficient values with respect to approximate mesh size h value with each mesh adaptation iteration. As it may be seen, as the mesh resolution increases the values converge closer to the reference values.

Figure 2 shows the comparisons of simulation 1 and simulation 3 with the slice contour of the MACH number and an isosurface of 0.2 MACH number. While the different mesh adaptation processes have different solution, the contour shows similar behavior.

2.2. Sydney Standard Aerodynamic Models 5th Generation Fighter Model

The second case is the SSAM 5th Generation Fighter model [4]. This case has a much complicated geometry compared to that of the previous delta wing. This case is also studied with the anisotropic mesh adaptation. Computational mesh edge information were also included in the adaptation process in order to retain the sharp geometrical entities. This case uses several different anisotropic mesh adaptation sensor functions which include distance, MACH, entropy, and their combinations in order to better capture the flow details. The geometry and benchmark provided mesh distribution is shown on the left in Fig. 3 below. The right side of Fig. 3

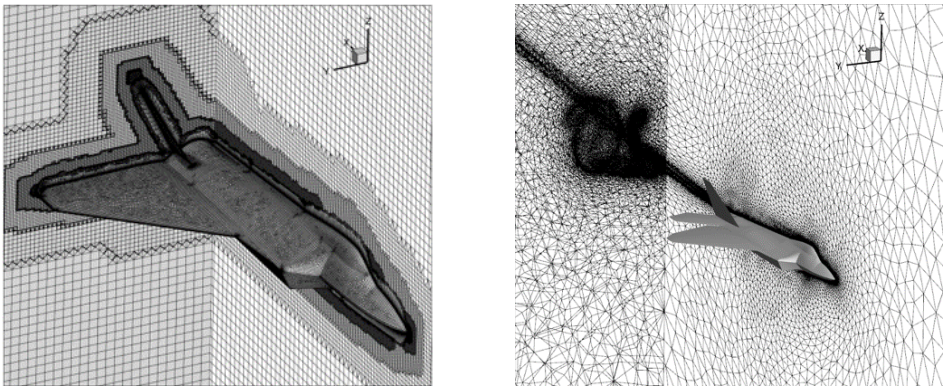


Figure 3: SSAM 5th Generation Fighter Model geometry and mesh [4] (left) and mesh distribution after mesh refinement (right).

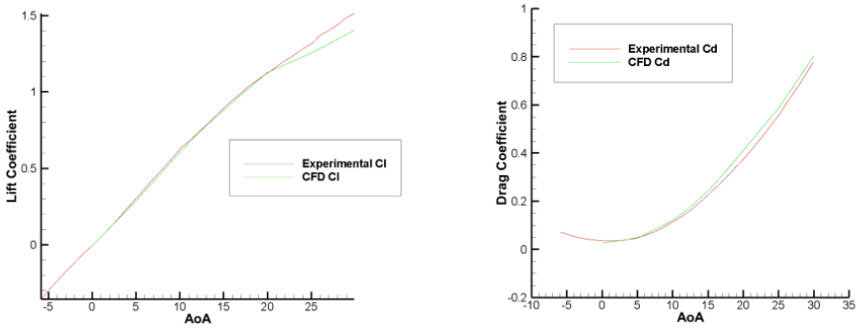


Figure 4: Lift (left) and drag (right) coefficient vs alpha plot compared with the experimental solutions.

shows the present anisotropic adaptive mesh distribution at zero degree. Detailed flow and geometry parameters are mentioned in the reference [4].

The several flux schemes were tested using the mesh distribution shown in Fig. 3 (left), then HLLC flux scheme were executed for angles from 0° to 30°. The computed aerodynamic loads were compared with the experimental data in Fig. 4 and the results are relatively in a good agreement. However, there is slight underprediction of computed forces, which may be due to poor fixed resolution issues.

For the last simulations the anisotropic mesh adaptation was used and the mesh resolution shown in Fig. 3 (right) was generated. As it may be seen the mesh resolution is significantly increased in the vortical and boundary layer regions. For the adaptation sensor, MACH and distance functions were used in collaboration. Figure 5 shows the mesh adaptation at 12 degrees with the pressure coefficient contour on the surface and the contours are compared with the reference fixed mesh calculations. The adaptive calculation captures the leading edge vortex better.

3. Conclusions

In conclusion, the SSAM 5th generation fighter model and delta wing geometry provided in the International Workshop on High-Order CFD Methods were studied. The HEMLAB algorithm was used as the flow solver with the pyAMG anisotropic mesh adaptation library. Different adaptation parameters were studied for the HiOCFD delta wing case.

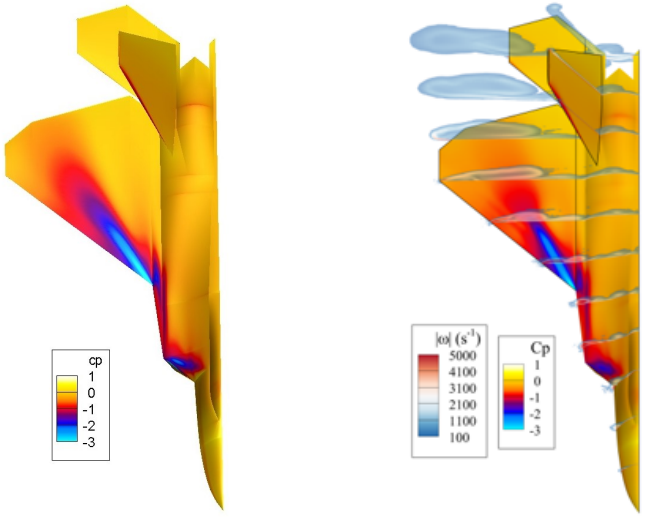


Figure 5: Pressure coefficient contours in the surface for the anisotropic mesh adaptation simulation (left) and the reference study [4] (right).

By using those parameters the SSAM model was studied. Different flux schemes and mesh adaptation sensor functions were tested. While volume mesh resolution was enhanced with the mesh adaptation, better surface mesh could be achieved with different sensor functions.

References

[1] S. Akkurt, M. Sahin, An efficient edge based data structure for the compressible Reynolds-averaged Navier–Stokes equations on hybrid unstructured meshes, International Journal for Numerical Methods in Fluids 94 (1) (2022) 13–31. doi:10.1002/fld.5045.

[2] T. Leicht, R. Hartmann, Error estimation and anisotropic mesh refinement for 3d laminar aerodynamic flow simulations, Journal of Computational Physics 229 (19) (2010) 7344–7360. doi:10.1016/j.jcp.2010.06.019.

[3] H. Akgün, I. Asar, M. Sahin, HEMLAB Algorithm Applied to HIOCFD Delta Wing and OTC1 Missile, in: AIAA AVIATION 2023 Forum, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2023. doi:10.2514/6.2023-4403.

[4] N. F. Giannelis, T. Bykerk, G. A. Vio, A Generic Model for Benchmark Aerodynamic Analysis of Fifth-Generation High-Performance Aircraft, Aerospace 10 (9) (2023) 746. doi:10.3390/aerospace10090746.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Application of the HEMLAB Algorithm to a Case from the 7th AIAA CFD Drag Prediction Workshop

Ibrahim Asar[a,*], Mehmet Sahin[a]

[a]Istanbul Technical University, Aeronautical and Astronautical Engineering, Maslak, 34469 Istanbul, Turkey

ARTICLE INFO[†]

Keywords:
Anisotropic Mesh Adaptations;
NASA Common Research Model;
Parallel Computational
Fluid Dynamics;
Transonic Flow;
Finite-Volume Method

ABSTRACT

This article presents the numerical analysis of the wing-body configuration of the NASA Common Research Model (CRM) within the context of the 7th AIAA CFD Drag Prediction Workshop (DPW-7), using the Reynolds-averaged Navier-Stokes flow solver HEMLAB. Estimating the effect of shock-induced separation on the aerodynamic force, moment coefficients, and pressure distribution in the wing-body configuration that deforms at each given angle of attack in transonic flight is the main goal of the workshop. To achieve this goal, anisotropic mesh adaptation calculations for the NASA-CRM wing-body configuration were carried out as part of test case 4 (alpha sweep). The HEMLAB flow solver algorithm uses an efficient edge-based data structure designed for a vertex based unstructured finite-volume method and is integrated with a python-based anisotropic mesh generation library pyAMG from INRIA to enhance its computational efficiency. The classical negative Spalart-Allmaras turbulence model is employed for turbulence modeling. The computations for the required DPW-7 cases were conducted using the fully coupled nonlinear Newton method available in the PETSc library.

1. Introduction

The Drag Prediction Workshop (DPW) series, initiated in 2000 by the Applied Aerodynamics Technical Committee of the American Institute of Aeronautics and Astronautics, has been instrumental in assessing the accuracy of computational fluid dynamic methods for predicting aerodynamic forces and moments in industry-relevant scenarios. The 7th Drag Prediction Workshop, focused on predicting the impact of shock-induced separation on lift and pitching moment variations under transonic conditions. Participants utilized computational methods to analyze geometry and grid configurations, aiming to enhance predictive capabilities in transonic aerodynamics. Therefore, the HEMLAB algorithm [1] is utilized here to assess its capabilities. The HEMLAB solver is a compressible Navier-Stokes solver designed for unstructured hybrid meshes in both two- and three-dimensions. It employs a vertex-based finite-volume algorithm and utilizes a quad/half-edge data structure for cache efficiency. The inviscid state vectors are computed with second- and third-order upwind least squares interpolation, and inviscid fluxes are determined using the Roe [2] and HLLC [3] scheme. The exact Jacobian matrices for the inviscid fluxes are determined using the source code transformation from TAPENADE [4] library. The gradient evaluations for viscous fluxes at edge mid-points are performed using the Green-Gauss theorem. The PETSc-3.21.0 [5] Scalable Nonlinear Equations Solvers (SNES) with a line search technique is employed for the solution of resulting algebraic equations. The METIS library is used for a balanced domain decomposition.

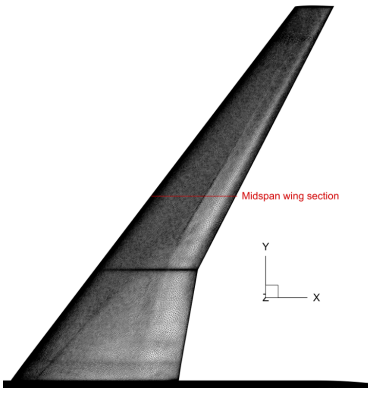
2. Numerical results

The CRM was created in collaboration with the DPW Organizing Committee and the NASA Subsonic Fixed Wing Aerodynamics Technical Working Group [6]. This model represents a contemporary transonic commercial transport aircraft, incorporating key components such as a low wing, fuselage, horizontal tail and engine nacelles mounted under the wing. The details of the geometry can be found on the website of the 7th AIAA CFD Drag Prediction Workshop[1]. The analysis results presented in this section were computed under the flow conditions of a freestream MACH number $M = 0.85$, a REYNOLDS number $Re = 20 \cdot 10^6$, and a reference temperature $T = -250°$F. The calculations are carried with anisotropic mesh adaptation. This method allows the dynamic adjustment of the computational domain resolution, aligning with the flow characteristics. Consequently, it enables a detailed focus on critical flow aspects, while simultaneously increasing the mesh resolution in areas of high gradients. As shown in Fig. 1, the resolution is primarily focused on the shock region over the main wing, allowing for a sharp capture of the flow physics. The boundary layer

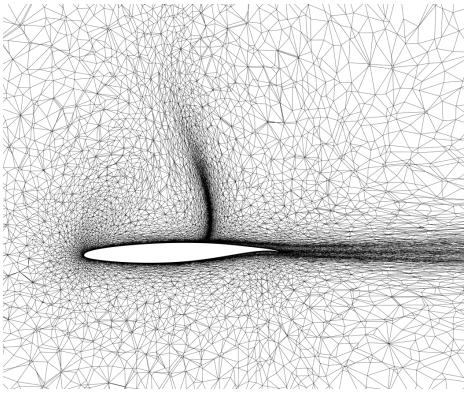
[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02509 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
asar17@itu.edu.tr (I. Asar); msahin.ae00@gtalumni.org (M. Sahin)
ORCID(s): 0000-0002-0951-2227 (I. Asar); 0000-0001-6502-2209 (M. Sahin)

[1]http://aiaa-dpw.larc.nasa.gov



(a) Adapted mesh.



(b) Wing midspan volume mesh.

Figure 1: CRM wing surface.

Level	Vertex number	Element number	Surface element number	C_D	C_L	C_M
Iteration 1-100	2,988,399	17,199,949	455,201	0.03404	0.58581	-0.13483
Iteration 101-200	4,599,847	26,393,588	793,573	0.03121	0.61015	-0.12777
Iteration 201-300	5,193,755	29,813,758	863,203	0.03105	0.61490	-0.13034
Iteration 301-500	5,561,202	31,955,097	896,778	0.03130	0.61367	-0.12978
Iteration 501-700	5,684,338	32,673,574	906,224	0.03127	0.61376	-0.13005

Table 1
Refinement data.

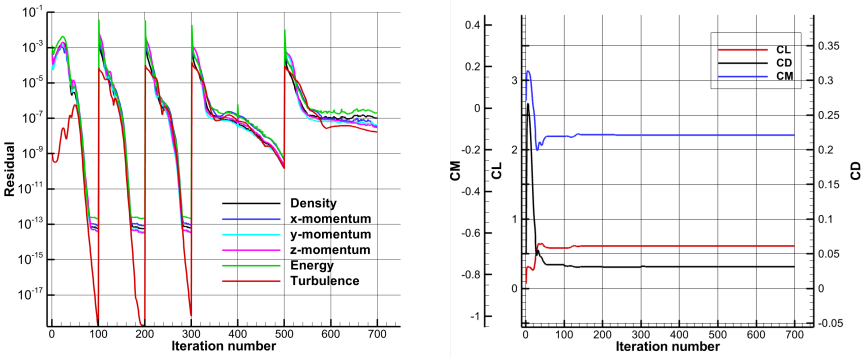
and the wake are also resolved with relatively fine detail. In this analysis, Dell PowerEdge R630 computer with a single node containing a 14-core Intel(R) Xeon(R) CPU E5-2680 v4 at 2.40GHz and 264GB memory was used, and parallelization was achieved using Open MPI. Table 1 provides detailed information on mesh refinement and its impact on aerodynamic coefficients over multiple iterations. The table shows an increase in the number of vertices and elements, indicating finer mesh resolution with each iteration.

Figure 2a shows the residual convergence of density, $x-$, $y-$, $z-$ momentum, energy, turbulence equations. Figure 2b depicts C_L, C_D, and C_M aerodynamic loads with Newton iteration number. At a 3° angle of attack, the coefficient of lift C_L converges to 0.61376, while the coefficient of drag C_D converges to 0.03127. The strong fully coupled nonlinear Newton method (PETSc-SNES) can lead to convergence at machine precision. The present initial calculations were executed without a limiter function and as the mesh becomes finer, it becomes more difficult to achieve machine precision. In the future, we will try to freeze the limiter after a certain residual threshold.

Figure 3a shows the pressure coefficient distribution contours and streamlines on the main-wing top surface, and Fig. 3b the wing midspan MACH contours. As expected, the shock waves were captured sharply on the wing mid-span section. In addition the shock induced separation bubble can be clearly observed over the main wing.

3. Conclusions

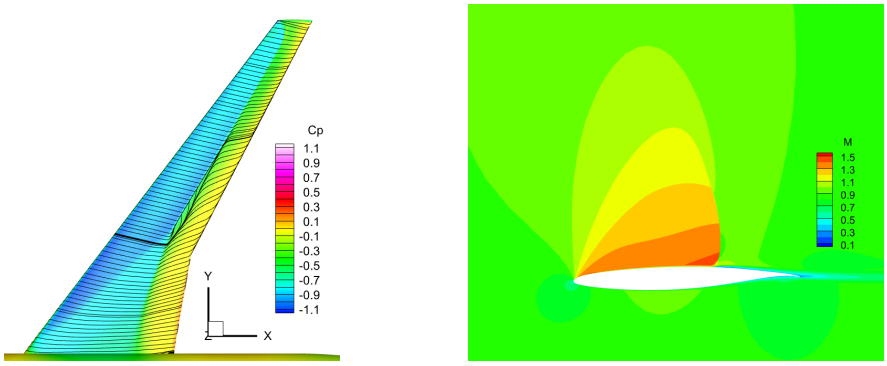
In conclusion, this study was to examine the NASA-CRM under the transonic conditions with the HEMLAB solver. The mesh resolution enhanced by using the pyAMG library from INRIA. Adaptive meshing successfully captured regions with high gradients such as shocks, wakes, tip vortices, etc. The effect of shock-induced separation on the force, moment coefficients, and pressure distribution in a wing-body configuration at a 3° angle of attack has been investigated in detail and highly accurate numerical results are obtained.



(a) Residual.

(b) C_L, C_D, and C_M.

Figure 2: Convergence of the simulation.



(a) Wing surface pressure coefficient.

(b) Wing midspan MACH contours.

Figure 3: Simulation results.

References

[1] S. Akkurt, M. Sahin, An efficient edge based data structure for the compressible Reynolds-averaged Navier–Stokes equations on hybrid unstructured meshes, International Journal for Numerical Methods in Fluids 94 (1) (2022) 13–31. doi:10.1002/fld.5045.

[2] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357–372. doi:10.1016/0021-9991(81)90128-5.

[3] E. F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1) (1994) 25–34. doi:10.1007/BF01414629.

[4] L. Hascoet, V. Pascual, The Tapenade automatic differentiation tool, ACM Transactions on Mathematical Software 39 (3) (2013) 1–43. doi:10.1145/2450153.2450158.

[5] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L. C. McInnes, R. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual (2021). URL https://publications.anl.gov/anlpubs/2021/08/170061.pdf

[6] J. Vassberg, M. Dehaan, M. Rivers, R. Wahls, Development of a Common Research Model for Applied CFD Validation Studies, in: 26th AIAA Applied Aerodynamics Conference, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2008, pp. AIAA 2008–6919. doi:10.2514/6.2008-6919.

Other Topic 3:
Numerical Methods

In this session, advanced numerical Methods are discussed. Topics include parallel-in-time spectral deferred correction for the incompressible Navier-Stokes equations and a comparison of eddy-viscosity models in reactor vessel auxiliary cooling systems. Additionally, a GPU-implementation of a multigrid accelerated projection method and a parallel unstructured conservative level-set (UCLS) method for liquid-vapor phase change phenomena are shown. Attendees gain insights into these sophisticated techniques and their implications for computational fluid dynamics and multiphase flow modeling.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Comparison of Eddy-Viscosity Models Applied to a Reactor Vessel Auxiliary Cooling System

Wei Wang[a,*], Bo Liu[a], Greg Cartland-Glover[a], Jundi He[b,c], Charles Moulinec[a], Stefano Rolfo[a] and Shuisheng He[b]

[a] UKRI STFC Daresbury Laboratory, Scientific Computing, Sci-Tech Daresbury, Warrington WA4 4AD, UK
[b] The University of Sheffield, Sheffield S1 3JD, UK
[c] Department of Mechanical and Aerospace Engineering, The University of Manchester, Manchester M13 9PL, UK

ARTICLE INFO†

Keywords:
Reactor Vessel Auxiliary Cooling Systems;
Reynolds-Averaged Navier-Stokes;
Conjugate Heat Transfer;
Natural Convection;
Forced Convection

ABSTRACT

This study assesses the performance of both high and low REYNOLDS number eddy-viscosity turbulence models in simulating a simplified Reactor Vessel Auxiliary Cooling System with conjugate heat transfer. For forced convection, the standard $k-\epsilon$ model and the low REYNOLDS number Launder-Sharma model show good agreements in streamwise velocity and fluid temperature compared to Direct Numerical Simulation data. For natural convection, the Launder-Sharma model is the most effective at capturing flow and thermal fields.

1. Introduction

Reactor Vessel Auxiliary Cooling Systems (RVACS) are key for nuclear reactor passive cooling, operating solely on convective heat transfer principles without reliance on electrical systems. They efficiently dissipate residual heat through convective exchange between ambient air and structural components. A typical RVACS configuration (see Fig. 1) features the entrance of cold air into a duct, whereupon its downward acceleration induces separation before impinging upon the lower structure. The ensuing upward flow cycle facilitates heat extraction from the reactor vessel, gradually re-attaching to a baffle. This process leads to complicated phenomena of fluid dynamics and heat transfer, accentuated by the interaction between structural elements and flow dynamics through conjugate heat transfer (CHT) mechanisms.

While Direct Numerical Simulation (DNS) offers good accuracy by fully resolving the turbulence and the heat transfer, its application to industrial contexts is hampered by computational resource constraints and time-intensive computations. Consequently, the pragmatic approach often lies in employing Reynolds-Averaged Navier–Stokes (RANS) modeling. An evaluation of several turbulence models within the context of RVACS not only provides an insight into model selection for simulation scenarios but also serves as a validation benchmark for RANS models applied to nuclear reactor design.

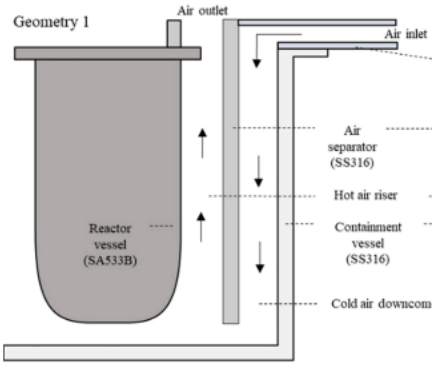


Figure 1: Schematic of a RVACS [1].

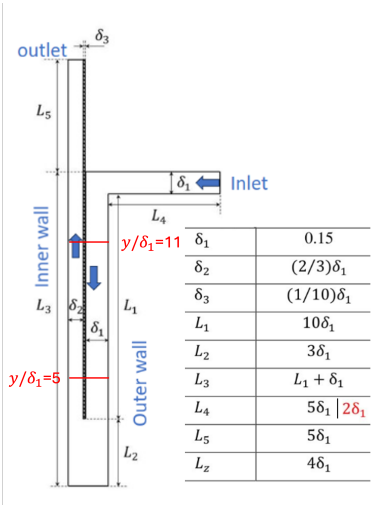
2. Numerical methods and configuration

2.1. Numerical settings

A simplified geometry (see Fig. 2) is utilized to represent the essential components of a complex RVACS setup. The simulation is conducted using a pseudo-3D configuration with only one layer of mesh in the spanwise (z) direction for

† This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02510 and of the Proceedings 10.34734/FZJ-2025-02175.

* Corresponding author
wei.wang@stfc.ac.uk (W. Wang); bo.liu@stfc.ac.uk (B. Liu); greg.glover@stfc.ac.uk (G. Cartland-Glover); jundi.he@sheffield.ac.uk (J. He); charles.moulinec@stfc.ac.uk (C. Moulinec); stefano.rolfo@stfc.ac.uk (S. Rolfo); s.he@sheffield.ac.uk (S. He)
ORCID(s): 0000-0002-7829-7479 (W. Wang); 0000-0002-6840-041X (B. Liu); 0000-0002-2647-2757 (G. Cartland-Glover); 0000-0002-2340-0694 (J. He); 0009-0003-7011-7327 (C. Moulinec); 0000-0001-6325-7629 (S. Rolfo); 0000-0003-0326-2447 (S. He)



δ_1	0.15
δ_2	$(2/3)\delta_1$
δ_3	$(1/10)\delta_1$
L_1	$10\delta_1$
L_2	$3\delta_1$
L_3	$L_1+\delta_1$
L_4	$5\delta_1$ | $2\delta_1$
L_5	$5\delta_1$
L_z	$4\delta_1$

Figure 2: Simplified geometry.

all the simulations conducted in this work. The height of the inlet channel is designated as $\delta_1 = 0.15$ m, with the baffle material specified as stainless steel (SS310), which has a thickness of $\delta_3 = 0.1\delta_1$. For forced convection scenarios, an entrance length L_4 of $5\delta_1$ is employed, whereas, for natural convection, it is reduced to $2\delta_1$ with sealed inlet and outlet. At the inlet boundary, a fully developed turbulent velocity profile with a bulk velocity of 0.6 m/s and a constant temperature of 300 K are prescribed. The inlet REYNOLDS number is 5,729. CHT phenomena are accounted for on the baffle surface. The inner wall is 700 K with a heat transfer coefficient of 200 $\mathrm{W/m^2K}$, while the outer wall is initialized at a temperature of 400 K for the forced convection case and 300 K for the natural convection case, both with a heat transfer coefficient of 200 $\mathrm{W/m^2K}$. The working medium throughout the system is air maintained at atmospheric pressure, with its temperature-dependent thermal properties referenced from the NIST Standard Reference Database [2]. Similarly, the baffle exhibits constant thermophysical properties pertinent to the conjugate heat transfer analysis [2].

2.2. Numerical method

The software code_saturne[1] is used in this study. Several high and low REYNOLDS number eddy-viscosity models are used to assess their behavior in both forced and natural convection in a simplified RVACS. The high REYNOLDS number models use wall-functions to account for the wall effects, while the low REYNOLDS number models do not rely on functions but require high mesh resolution near the wall to capture the near-wall flow features. The high REYNOLDS number RANS turbulence models studied here include the $k-\varepsilon$ model (KE-HiRe) and the SST $k-\omega$ model (KOMG-HiRe) with wall functions. The low REYNOLDS number RANS turbulence models include the Launder-Sharma model (KE-LS-LoRe), the standard SST $k-\omega$ model (KOMG-LoRe) and the V2F model (V2F-LoRe). Details of the implementation of these models can be found in [3].

After a grid sensitivity study using three sets of mesh for each study, the final mesh setup is: a mesh made of 7,775 hexahedral cells is used for the high REYNOLDS number RANS modeling, where the first mesh layer has a y^+ not smaller than 11. A mesh of 927,835 hexahedra is used for the low REYNOLDS number RANS modeling, where the first mesh layer has a y^+ not greater than 1.

3. Simulation results

Defining the line $y/\delta_1 = 0$ at the bottom wall as the current coordinate system, data along two lines at $y/\delta_1 = 5$ and 11 are selected to show the simulation profiles. The right side ($x/\delta_1 > 0.77$) of these lines is for the downward flow for forced and mixed convection, whilst the left side ($x/\delta_1 < 0.67$) is for the upward flow. The DNS data are used as a reference.

3.1. Forced convection

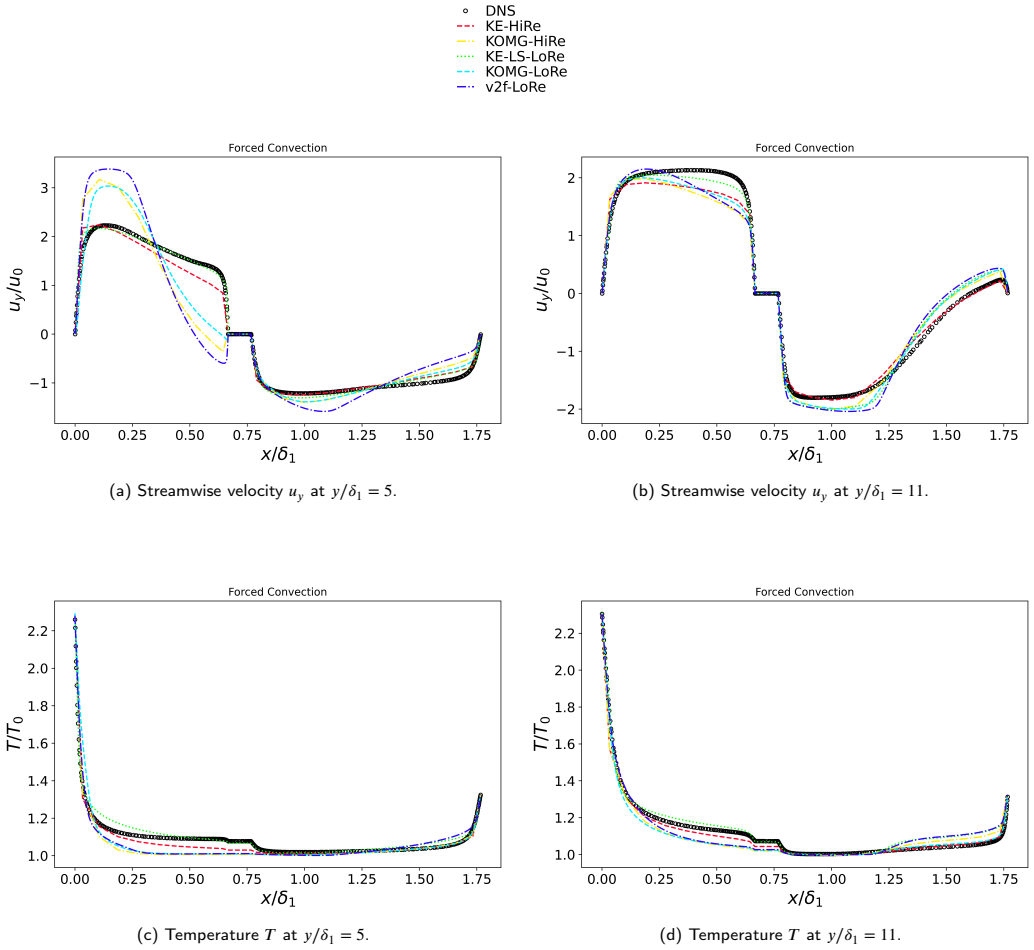
Figures 3a and 3b show the streamwise velocity u_y along the two specified lines. Without going into details in this abstract, we could say the KE-HiRe model captures well the streamwise velocity in the downward flow, while the KE-LS-LoRe model gives the best agreement to the DNS data in the upward flow.

Figures 3c and 3d show the temperature along the two specified lines. In general, the KE-HiRe model provides a good prediction of the thermal field, and the KE-LS-LoRe model is the best to capture both the fluid and the baffle temperature.

3.2. Natural convection

Figures 4a and 4b compare the performance of several RANS models in capturing the streamwise velocity. For the region with $x/\delta_1 > 0.77$, all models overpredict the streamwise velocity near the cold wall and underpredict it near the hot wall, and low REYNOLDS number models show better results than those high REYNOLDS number models. For the region with $x/\delta_1 < 0.67$, the KE-LS-LoRe model provides the closest results to the DNS reference. Figures 4c and 4d show the temperature along the two specified lines. Similar as for forced convection, the KE-LS-LoRe model provides the best results of the flow and the baffle temperature.

[1] https://www.code-saturne.org/cms/web/



(a) Streamwise velocity u_y at $y/\delta_1 = 5$.

(b) Streamwise velocity u_y at $y/\delta_1 = 11$.

(c) Temperature T at $y/\delta_1 = 5$.

(d) Temperature T at $y/\delta_1 = 11$.

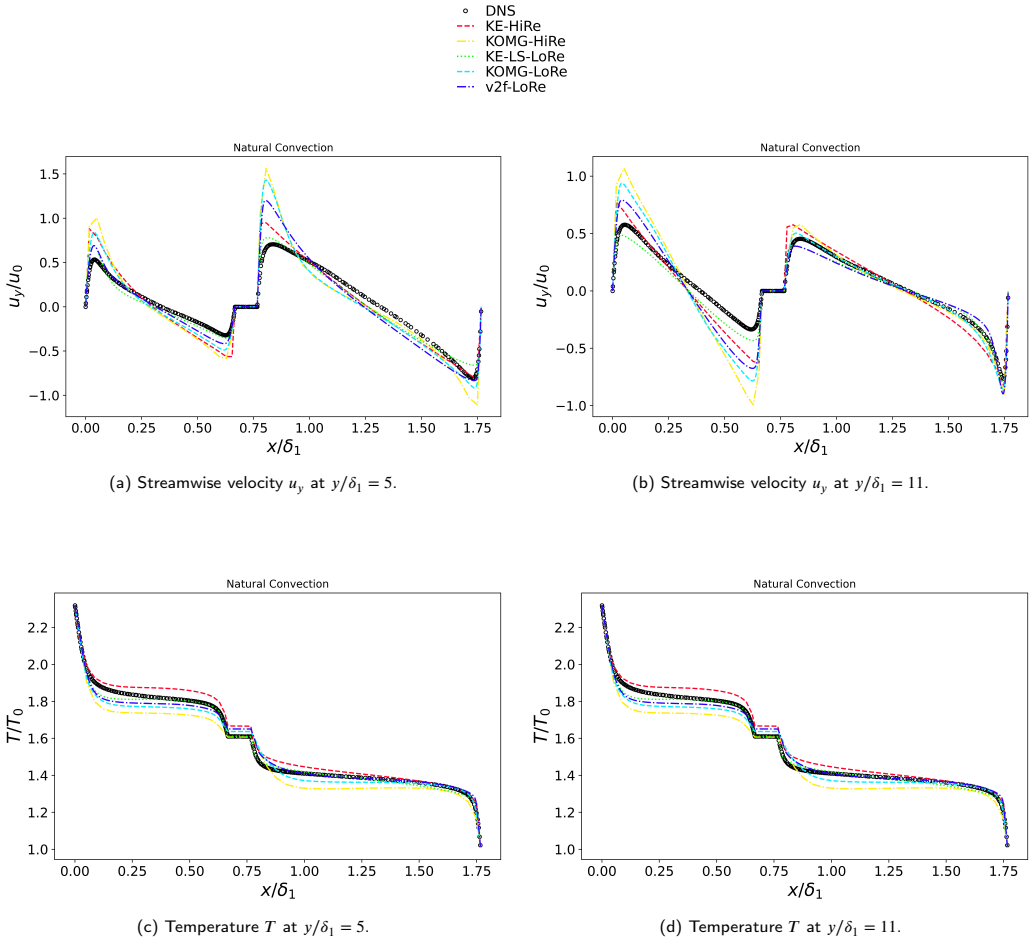
Figure 3: Forced convection: Streamwise velocity u_y and temperature T.

4. Conclusions

The behaviors of several high and low REYNOLDS number eddy-viscosity turbulence models are assessed for modeling a simplified Reactor Vessel Auxiliary Cooling System with Direct Numerical Simulation data as a reference. For the forced convection, the standard $k-\varepsilon$ model and the low REYNOLDS number Launder-Sharma model provide the best comparison regarding the streamwise velocity and temperature distributions. For natural convection, the low REYNOLDS number Launder-Sharma model is the best at capturing flow and thermal fields.

Acknowledgements

This work made use of computational support by CoSeC, through CCP-NTH (EP/T026685/1). The authors thank ARCHER2 for computational resources via Archer2 Pioneer Project (e814).



(a) Streamwise velocity u_y at $y/\delta_1 = 5$.

(b) Streamwise velocity u_y at $y/\delta_1 = 11$.

(c) Temperature T at $y/\delta_1 = 5$.

(d) Temperature T at $y/\delta_1 = 11$.

Figure 4: Natural convection: Streamwise velocity u_y and temperature T.

References

[1] S. Lee, Y. J. Choi, J. I. Lee, Y. H. Jeong, Investigation of various reactor vessel auxiliary cooling system geometries for a hybrid micro modular reactor, Nuclear Engineering and Design 379 (2021) 111239. doi:10.1016/j.nucengdes.2021.111239.

[2] E. W. Lemmon, I. H. Bell, M. L. Huber, M. O. McLinden, NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 10.0, National Institute of Standards and Technology (2018). doi:10.18434/T4/1502528.

[3] Fluid Dynamics, Power Generation and Environment Department and Single Phase Thermal-Hydraulics Group, EDF R&D, code saturne documentation: code saturne 8.0 Theory Guide (2023).
URL https://code-saturne.org/documentation/8.0/theory.pdf

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Multigrid Accelerated Projection Method on GPU

Tzu-Hsuan Chiu[a], Chao-An Lin[a,*]

[a]National Tsing Hua University, Department of Power Mechanical Engineering, 30013, Hsinchu, Taiwan

ARTICLE INFO[†]

Keywords:
Projection Method;
Multigrid;
Coarse Grid Aggregation;
Graphics Processing Unit (GPU);
Incompressible Flow

ABSTRACT

This study introduces a GPU-accelerated numerical framework based on the projection method (PM) for simulating incompressible flows. The core of this approach is an efficient solution of the pressure Poisson equation using a V-cycle geometric multigrid scheme, further enhanced by Coarse Grid Aggregation (CGA) to optimize multigrid levels across multiple GPUs, resulting in substantial performance improvements. Benchmark results demonstrate that, for a 256^3 grid, the proposed implementation not only matches the efficiency of the Lattice Boltzmann Method (LBM) but also surpasses the Dual Time-Stepping Artificial Compressibility (DTAC) method and the explicit weakly compressible General Pressure Equation (GPE) scheme by factors of eleven and three, respectively.

1. Introduction

The graphics processing unit (GPU) has emerged as a powerful computing platform, valued for its exceptional thread parallelism and high memory bandwidth, making it especially effective for large-scale scientific simulations. These advantages have positioned GPUs as an ideal choice for computational fluid dynamics (CFD) applications, where high computational demands are common. Consequently, numerous studies have harnessed GPUs in CFD, achieving substantial performance gains, often surpassing an order of magnitude compared to traditional CPU-based approaches [1, 2].

The multigrid scheme is renowned for its effectiveness in mitigating low-frequency errors, a critical advantage for achieving efficient and accurate numerical simulations [3]. Leveraging these benefits, this study introduces a GPU-enabled projection method augmented with a Coarse Grid Aggregation (CGA)-enhanced multigrid framework for simulating incompressible flows on GPU clusters. By incorporating CGA, we increase the maximum multigrid level attainable on each GPU, leading to significant improvements in scalability and computational efficiency. The focus of this work is to evaluate the relative performance gains of our method against the Lattice Boltzmann Method (LBM) [4], the General Pressure Equation (GPE) solver [5], and the Dual Time-Stepping Artificial Compressibility (DTAC) method [2].

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02511 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
calin@pme.nthu.edu.tw (C. Lin)
ORCID(s): - (T. Chiu); - (C. Lin)

2. Governing equation and discretization

The four-step projection algorithm [6, 7] with a third-order, TVD Runge-Kutta (RK) scheme [8] is carried out in the following equations

$$\begin{aligned}
\boldsymbol{u^{(1)}} &= \boldsymbol{u^n} + \delta t \underbrace{(\nu \Delta \boldsymbol{u^n} - \nabla\cdot(\boldsymbol{u^n u^n}) - \frac{1}{\rho}\nabla p^n)}_{L_u(\boldsymbol{u^n})}, \\
\boldsymbol{u^{(2)}} &= \frac{3}{4}\boldsymbol{u^n} + \frac{1}{4}[\boldsymbol{u^{(1)}} + \delta t L_u(\boldsymbol{u^{(1)}})], \\
\boldsymbol{u^*} &= \frac{1}{3}\boldsymbol{u^n} + \frac{2}{3}[\boldsymbol{u^{(2)}} + \delta t L_u(\boldsymbol{u^{(2)}})], \qquad (1)
\end{aligned}$$

$$\frac{\boldsymbol{u^{**}} - \boldsymbol{u^*}}{\delta t} = \frac{1}{\rho}\nabla p^n, \qquad (2)$$

$$\frac{1}{\delta t}\nabla\cdot\boldsymbol{u^{**}} = \frac{1}{\rho}\Delta p^{n+1}, \qquad (3)$$

$$\frac{\boldsymbol{u^{n+1}} - \boldsymbol{u^{**}}}{\delta t} = -\frac{1}{\rho}\nabla p^{n+1}. \qquad (4)$$

Equations (1) to (4) are primarily explicit, with the exception of Eq. (3), which represents the pressure Poisson equation and is the most computationally demanding component of the scheme. To enable an explicit formulation on the GPU, we apply the Jacobi point scheme by introducing a pseudo-time derivative in the third step of the projection method, as shown in Eq. (3). This approach parallels the dual time-stepping scheme utilized by Shi et al. [2] and was also adopted by Chiu and Lin [9]. As a result, the pressure Poisson equation in Eq. (3) is replaced by the following equation

$$\underbrace{\frac{1}{\beta}\frac{p^{k+1}-p^{k}}{\delta\tau}+\underbrace{\frac{1}{\delta t}\nabla\cdot\boldsymbol{u}^{**}-\frac{1}{\rho}(\Delta_{d}p^{k+1}+\Delta_{od}p^{k})}_{L_{p0}(p)}=0,}_{L_{p}(p)} \quad (5)$$

where k represents the time level in the pseudo-time iteration, while $\delta\tau$ corresponds to the time step. The operators $L_p(p)$ and $L_{p0}(p)$ refer to the versions of the operators that include and exclude the pseudo-time term, respectively. Additionally, Δ represents the sum of the Laplacian operators, Δ_d and Δ_{od}, where are the diagonal and off-diagonal parts, respectively. Equation (5) is iterated in pseudo-time until a steady state is reached, resulting in an approximation where $p^{k+1} \approx p^{k} = p^{n+1}$, effectively recovering Eq. 3.

The present V-cycle multigrid procedure follows the approach presented in [2]. Before delving into the multigrid scheme for solving the pressure Poisson equation, it is necessary to introduce some notations. The pressure variable at different grid levels is denoted as p^h, where h represents the level of the multigrid layers, with $h = 0$ corresponding to the finest level. The restriction operator transfers data from the h-th level to the $(h + 1)$-th level, and vice versa for the prolongation operator. It should be noted that both the time step Δt and the pseudo-time step $\Delta\tau$ need to be adjusted at different grid levels based on the CFL condition. The restriction and prolongation operators are defined as follows

$$\tilde{p}^{h+1} = I_{h}^{h+1}p^{h}, \qquad \hat{p}^{h} = I_{h+1}^{h}p^{h+1}, \quad (6)$$

where $\tilde{p}$ and $\hat{p}$ denote the restricted and prolonged pressure, respectively. The operators $L_p^h(\cdot)$ and $L_{p0}^h(\cdot)$ are employed to solve the discrete equations at grid level h, as expressed in Eq. (5). The residual of the pressure evolution equation at grid level h is computed as follows

$$R_{p}^{h} = (\frac{1}{\delta t}\nabla\cdot\boldsymbol{u}^{**}-\frac{1}{\rho}\Delta p^{k+1})^{h}. \quad (7)$$

It is important to note that the residual is not zero when the pseudo time steady state has not been reached in Eq. (5), i.e., $p^{k+1} \neq p^{k}$. The non-zero residual indicates that further iterations are required to converge to the steady-state solution in the pseudo time.

The coarse-grid version of the pressure Poisson equation can be formulated by considering the restricted approximate solutions and residuals as a source term. This is expressed by the following equation

$$L_{p}^{h+1}(p^{h+1}) = \underbrace{L_{p0}^{h+1}(\tilde{p}^{h+1}) - I_{h}^{h+1}(R_{p}^{h} - \overbrace{RHS_{p}^{h}}^{zero\ at\ h=0})}_{RHS_{p}^{h+1}}, \quad (8)$$

where the RHS is not computed on the finest grid level ($h = 0$). Therefore, we further describe the right-hand side of Eq. (8) as in the following:

$$\begin{aligned} RHS_{p}^{h+1} &= (\frac{1}{\delta t}\nabla\cdot\tilde{\boldsymbol{u}}^{**}-\frac{1}{\rho}\Delta\tilde{p}^{k+1})^{h+1} \\ &\quad -I_{h}^{h+1}(R_{p}^{h}-RHS_{p}^{h}). \end{aligned} \quad (9)$$

Noted that the pseudo-time derivative terms are not included while computing the RHS value. The procedure involves several steps to iteratively solve the pressure Poisson equation at different grid levels.

3. Results

Figure 1 presents the solution times for the pressure Poisson equation, solved with and without Coarse Grid Acceleration (CGA), across various grid resolutions on an 8-GPU cluster. The CGA implementation achieves a consistent speedup of approximately 2.3 to 2.6 times, highlighting the performance benefits of additional multigrid levels. The optimized computational times for grid sizes of 128^3, 256^3, 512^3 and 1024^3, are 0.16, 0.32, 0.89, and 4.2 seconds, respectively.

We further compare the performance of our proposed method with several established approaches: the Dual Time-Stepping Artificial Compressibility (DTAC) method [2], a general pressure equation (GPE) solver [5], and the Lattice Boltzmann Method (LBM) [4]. Additionally, an explicit CPU-based projection method employing the conjugate gradient algorithm is included for reference. The test case employed in these comparisons is the unsteady lid-driven cavity flow at Re=3200, starting from a stationary state, with simulations conducted until a physical time of t=1 on either two NVIDIA Tesla V100 GPUs or an Intel i7-6900K CPU.

Both the GPE solver [5] and the LBM method [4] in these comparisons are weakly compressible, non-iterative, and fully explicit schemes. For DTAC, multigrid acceleration is applied to the inter-time subiterations in the artificial compressibility framework. GPU implementations for DTAC, GPE, and LBM were developed by Shi et al. [2], Shi and Lin [5], and Lee et al. [4], respectively, with a constant CFL number of 0.7. To mitigate compressibility effects on incompressible flows, the MACH number for the weakly compressible GPE and LBM methods is set to Ma=0.05.

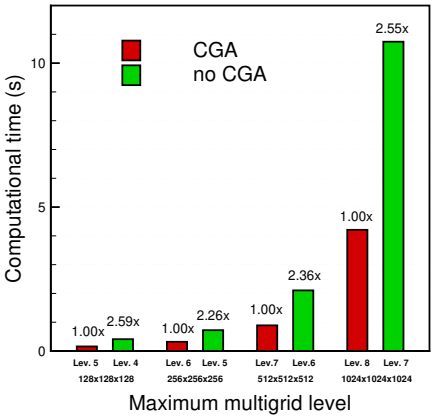


Figure 1: Solution time for pressure Poisson equation with/without optimized CGA level using 8 GPUs- unsteady lid-driven cavity at 10th physical time step (Re=1000).

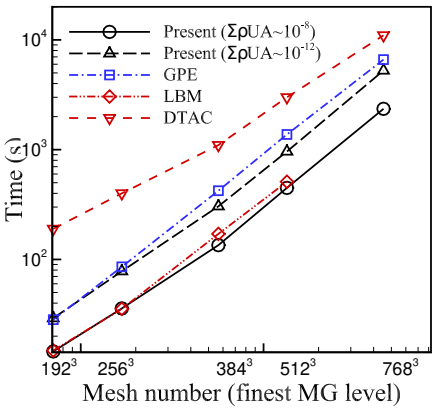


Figure 2: Computational time for the present method, GPE, and LBM on two NVIDIA V100 GPUs and CPU-based CG method-unsteady lid-driven cavity at t=1 (s) (Re=3200).

The stopping criterion for the Poisson subiteration of the present projection method is set to $|\sum \rho U A| < 10^{-8}$ or $|\sum \rho U A| < 10^{-10}$ at each physical time step. However, the divergence accuracy at $|\sum \rho U A| < 10^{-8}$ is acceptable for most simulations.

Figure 2 presents the computational times needed to achieve one second of physical simulation across various grid resolutions. The proposed GPU-based projection method achieves substantial speedups over both the DTAC and GPE methods, offering a threefold performance improvement over the GPE method while remaining compatible with LBM for a convergence criterion of $|\sum \rho U A| < 10^{-8}$. Although the LBM approach is highly efficient, it has limitations with non-uniform grid implementation—essential for simulations of wall-bounded turbulent flows—and requires significantly more memory compared to our method.

As anticipated, the performance decreases with a stricter criterion of $|\sum \rho U A| < 10^{-10}$, but remains comparable to that of the GPE method. Benchmarking results for a 256^3 grid show that our implementation not only matches the efficiency of the Lattice Boltzmann Method (LBM) but also surpasses the Dual Time-Stepping Artificial Compressibility (DTAC) method and the explicit weakly compressible General Pressure Equation (GPE) scheme by factors of eleven and three, respectively.

References

[1] X. Wang, T. Aoki, Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster, Parallel Computing 37 (9) (2011) 521–535. doi:10.1016/j.parco.2011.02.007.

[2] X. Shi, T. Agrawal, C.-A. Lin, F.-N. Hwang, T.-H. Chiu, A parallel nonlinear multigrid solver for unsteady incompressible flow simulation on multi-GPU cluster, Journal of Computational Physics (2020) 109447. doi:10.1016/j.jcp.2020.109447.

[3] A. Brandt, Multilevel adaptive computations in fluid dynamics, AIAA Journal 18 (1980) 1165–1172. doi:10.2514/3.50867.

[4] Y.-H. Lee, L.-M. Huang, Y.-S. Zou, S.-C. Huang, C.-A. Lin, Simulations of turbulent duct flow with lattice Boltzmann method on GPU cluster, Computers & Fluids 168 (2018) 14–20. doi:10.1016/j.compfluid.2018.03.064.

[5] X. Shi, C.-A. Lin, Simulations of Wall Bounded Turbulent Flows Using General Pressure Equation, Flow, Turbulence and Combustion (2020) 1–16. doi:10.1007/s10494-020-00119-z.

[6] H. Choi, P. Moin, Effects of the computational time step on numerical solutions of turbulent flow, Journal of Computational Physics 113 (1) (1994) 1–4. doi:10.1006/jcph.1994.1112.

[7] C.-C. Liao, C.-A. Lin, Influence of Prandtl number on the instability of natural convection flows within a square enclosure containing an embedded heated cylinder at moderate Rayleigh number, Physics of Fluids 27 (1) (2015) 013603. doi:10.1063/1.4906181.

[8] S. Gottlieb, C.-W. Shu, Total variation diminishing Runge-Kutta schemes, Mathematics of Computation 67 (221) (1998) 73–85. doi:10.1090/S0025-5718-98-00913-2.

[9] T.-H. Chiu, C.-A. Lin, Multigrid accelerated projection method on GPU cluster for the simulation of turbulent flows, Journal of Mechanics 39 (2023) 199–212. doi:10.1093/jom/ufad015.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Parallel Unstructured Conservative Level-Set (UCLS) Method for Liquid-Vapor Phase Change Phenomena

Nestor Balcazar-Arciniega[a,*], Joaquim Rigola[a] and Assensi Oliva[a]

[a] Universitat Politècnica de Catalunya-BarcelonaTech., Heat and Mass Transfer Technological Centre, Colom 11, 08222 Barcelona, Spain

ARTICLE INFO[†]

Keywords:
Unstructured Conservative Level-Set Method;
Unstructured Flux-Limiters;
Finite-Volume Method;
Liquid-Vapor Phase Change;
Multiphase Flow and Chemical Engineering Applications;
Parallel Computational Fluid Dynamics

ABSTRACT

This work introduces a parallel unstructured conservative level-set method for liquid-vapor phase change phenomena. The finite-volume method discretizes transport equations on three-dimensional collocated unstructured meshes. Mass transfer induced by the liquid-vapor phase change is calculated using temperature gradients in the liquid and vapor phases around the interface. The fractional-step projection method solves the pressure-velocity coupling. Unstructured flux-limiter schemes solve the convective term of transport equations to avoid numerical oscillations around the interface and minimize numerical diffusion. Verification and validation of numerical method are presented, which prove the robustness and accuracy of the UCLS solver for liquid vapor phase change on unstructured meshes.

1. Introduction

Liquid-vapor phase change phenomena are common in natural and industrial processes [1]. Many engineering devices, such as condensers and cooling towers in power plants, refrigeration systems, and unit operations in chemical engineering, involve the generation of bubbles or droplets through liquid-vapor phase change processes, including boiling, evaporation, and condensation. While empirical correlations are available to predict heat transfer in boiling flows, uncertainties in experimental measurements and significant physical simplifications in analytical results limit their range of applicability. Consequently, developing predictive computational methods based on first physical principles, such as Direct Numerical Simulation (DNS) of liquid-vapor phase change, is justified as a complement to experimental measurements and theoretical methods.

In the context of DNS of two-phase flows, multiple techniques have been reported in the literature [2], including front-tracking [3], Volume-of-Fluid (VoF) [4], level-set (LS) [5], coupled VoF-LS [6, 7], and conservative level-set [8, 9, 10, 11]. Further extensions of interface capturing to film boiling heat transfer have also been proposed [12, 13, 2]. Nevertheless, to the best of the authors' knowledge, liquid-vapor phase change phenomena have not been researched using the Unstructured Conservative Level-Set (UCLS) method [9, 10, 11].

Building upon our previous research in liquid-vapor phase change [14, 15], this work represents a further step in the development of the parallel UCLS method for film boiling heat transfer in 3D unstructured meshes. The UCLS method [9, 10, 11] offers several advantages: it avoids the accumulation of mass conservation errors common in standard level-set methods, unstructured meshes can be easily adapted to complex domains [14], and it enables accurate computation of surface tension forces [16, 11]. Additionally, the UCLS method ensures numerical stability at high physical property ratios [11], and the appropriate selection of unstructured flux-limiter convective schemes proposed by Balcazar-Arciniega et al. [9, 10, 11] avoids numerical oscillations around the interface and minimizes numerical diffusion.

2. Mathematical Formulation and Numerical Methods

The Navier-Stokes equations for two-phase flow with phase change are written in the framework of the so-called one fluid formulation [2], adapted to the UCLS method [14, 15]:

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) = -\nabla p + \nabla\cdot(\mu(\nabla\mathbf{v})) + \nabla\cdot\left(\mu(\nabla\mathbf{v})^T\right) + \rho\mathbf{g} + \mathbf{f}_\sigma, \tag{1}$$

$$\nabla\cdot\mathbf{v} = \left(\rho_v^{-1} - \rho_l^{-1}\right)\dot{m}_{lv}\delta_\Gamma, \tag{2}$$

Here, p is the pressure, $\mathbf{v}$ is the velocity, $\mathbf{g}$ is the gravity, $\mathbf{f}_\sigma$ is the surface tension force concentrated at the interface.

[†] This paper is part of the ParCFD 2024 Proceedings. The DOI of this document is 10.34734/FZJ-2025-02512 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
nestor.balcazar@upc.edu (N. Balcazar-Arciniega);
nestorbalcazar@yahoo.es (A. Oliva)
ORCID(s): 0000-0003-0776-2086 (N. Balcazar-Arciniega);
0000-0002-6685-3677 (J. Rigola); 0000-0002-2805-4794 (A. Oliva)

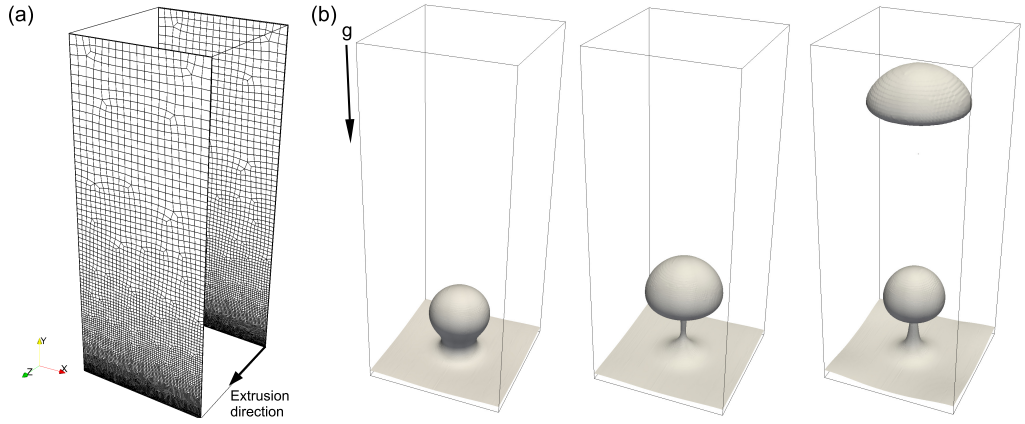


Figure 1: (a) Example of extruded mesh configuration. Ω is discretized by $3.89 \cdot 10^6$ hexahedral control volumes, distributed on 192 CPU cores. (b) Film boiling, Pr = 10, Ja = 1, Gr = 225.1, $\rho_l/\rho_v = \mu_l/\mu_v = 10$, $\lambda_l/\lambda_v = c_{p,l}/c_{p,v} = 1$. Snapshots of the interface at $t^* = t\, t_s^{-1} = \{11.8, 14.2, 28.4\}$.

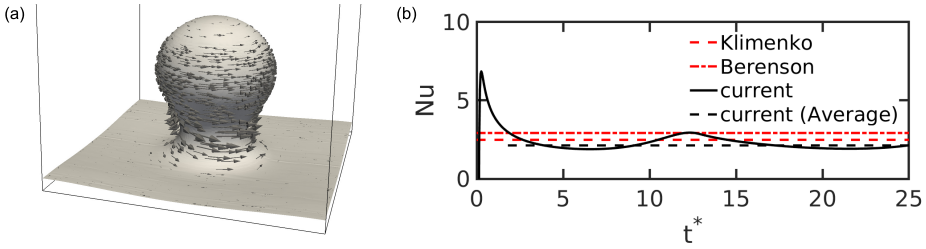


Figure 2: (a) Vorticity vectors ($\nabla \times \mathbf{v}$) on the interface. (b) The Nusselt number calculated by the parallel UCLS method is compared against Klimenko's correlation [17, 18] and Berenson's correlation [19].

The density ρ and dynamic viscosity μ are given by $\rho = \rho_l H_l + \rho_v H_v$ and $\mu = \mu_l H_l + \mu_v H_v$, respectively, where subscripts l and v refers to the liquid and vapor phases. H_v is the Heaviside step function, 1 in Ω_v (vapor phase) and 0 elsewhere, $H_l = 1 - H_v$. Furthermore, δ_Γ is the Dirac delta function concentrated at the interface, and $\dot{m}_{lv}$ is the mass transfer rate promoted by the phase change.

Interface capturing is performed by the Unstructured Conservative Level-Set (UCLS) method proposed by Balcázar et al. [10, 9, 11]. The interface advection equation with phase change [14, 15] and the unstructured re-initialization equation [9, 10] are solved as follows:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{v}) = -\frac{1}{\rho_l} \dot{m}_{lv} \delta_\Gamma^s, \quad \frac{\partial \phi}{\partial \tau} + \nabla \cdot \left(\phi(1-\phi)\mathbf{n}^0 \right) = \nabla \cdot (\varepsilon \nabla \phi), \tag{3}$$

where ϕ is the level-set function [9, 10], $\delta_\Gamma^s = ||\nabla \phi||$ is the smoothed Dirac delta function [10, 9, 14], $\varepsilon = 0.5\, h^{0.9}$ is the interface thickness parameter [10, 11, 9], h is the local grid size. The re-initialization equation is advanced in pseudo-time τ [9, 20, 10, 11], $\mathbf{n}^0$ is the interface normal unit vector evaluated at $\tau = 0$ [9, 10, 11]. At the discretized level, two re-initialization steps ($\Delta\tau$) are enough to keep a constant level-set profile [10, 11, 9]. The Heaviside step function is regularized as $H_l^s = 1 - H_v^s = \phi$ [14, 15]. The Continuous Surface Force (CSF) model [21] calculates the surface tension force in the framework of the UCLS method [10, 16, 22, 23, 9]. Consequently, $\mathbf{f}_\sigma = \sigma \kappa \nabla \phi$, where σ is the surface tension coefficient. Interface unit normal and

curvature are given by $\mathbf{n} = \nabla\phi \,/\!/\, \nabla\phi \,/\!/^{-1}$ and $\kappa = -\nabla \cdot \mathbf{n}$, respectively. The thermal energy equation is solved on Ω_v:

$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{v}T) = \frac{1}{\rho c_p} \nabla \cdot (\lambda \nabla T), \tag{4}$$

where $c_p = c_{p,v}$ is the specific heat capacity at constant pressure, $\lambda = \lambda_v$ is the thermal conductivity. Ω_l and the vapor-liquid interface are at the saturation temperature T_{sat}. The mass transfer rate ($\dot{m}_{lv}$) induced by the phase change is given by $\dot{m}_{lv} = h_{lv}^{-1} \left(\lambda_v (\nabla T \cdot \mathbf{n})_v - \lambda_l (\nabla T \cdot \mathbf{n})_l \right)$ [14]. Here, h_{lv} refers to the heat of vaporization, whereas gradients are extended on the interface cells following an unstructured extrapolation method proposed by [14]. Concerning the finite-volume approach, convective term of transport equations is discretized by unstructured flux-limiters schemes [9, 10, 11, 24] (SUPERBEE limiter). The fractional-step projection method [25, 2] solves the pressure-velocity coupling [14, 15]. Further technical details on the finite-volume discretization of transport equations on 3D collocated unstructured meshes can be found in [10, 11].

3. Numerical Experiments and Discussion

Previous validations and verifications of the UCLS method [9, 10, 11] include thermocapillarity motion of droplets [16, 23], gravity-driven bubbles [9, 26, 27], bubbly flows [20, 27, 10, 28, 11], binary droplet collision [20], collision of a droplet against a fluid-fluid interface [20], falling droplets [22], mass transfer in bubble swarms [10, 11], and liquid-vapor phase change [14, 15]. This work represents a further step toward developing computational methods for film boiling heat transfer on 3D unstructured meshes within the UCLS framework proposed by Balcazar-Arciniega et al. [10, 23, 16, 26, 20, 9, 29, 11].

Film boiling heat transfer is characterized by the PRANDTL number $\mathrm{Pr} = \mu_v \, c_{p,v} \, \lambda_v^{-1}$, GRASHOF number $\mathrm{Gr} = \rho_v \, (\rho_l - \rho_v) \, g \, l_s^3 \, \mu_v^{-1}$, JAKOB number $\mathrm{Ja} = c_{p,v} \, (T_w - T_{sat}) \, h_{lv}^{-1}$, physical properties ratios $\beta_v \, \beta_l^{-1}$ where $\beta = \{\rho, \mu, c_p, \lambda\}$. Here, $g = ||\mathbf{g}||$, $l_s = \left(\sigma \, g^{-1} \, |\rho_l - \rho_v|^{-1} \right)^{1/2}$ is the capillary length scale, $v_s = (g \, l_s)^{1/2}$ is the characteristic velocity, and $t_s = l_s / v_s$ is the characteristic time scale. On the other hand, the NUSSELT number is given by $\mathrm{Nu} = T^{-1} \int_{t_0}^{t_0+T} \mathrm{Nu}^*(t) \, dt$, where $\mathrm{Nu}^*(t) = A_w^{-1} \int_{A_w} \mathrm{Nu}^{**}(\mathbf{x}_w, t) dA$ and $\mathrm{Nu}^{**}(\mathbf{x}_w, t) = l_s \, (T_w - T_{sat})^{-1} \, (\nabla T \cdot \mathbf{e}_w)(\mathbf{x}_w, t)$, $\mathbf{e}_w$ is a unit vector perpendicular to the bottom wall ($y = 0$) pointing toward the fluids, A_w is the surface of the bottom wall ($y = 0$), and T is the period of averaging.

The computational set-up is depicted in Fig. 1a, Ω is a rectangular domain of size $(L_x, L_y, L_z) = (10.9 \, l_s, 27.2 \, l_s, 10.9 \, l_s)$, discretized by $3.89 \cdot 10^6$ hexahedral control volumes, distributed on 192 CPU cores. The mesh is generated by a constant step extrusion h_{min} of a bi-dimensional mesh along the $z-$axis. The mesh is refined close to the bottom wall ($y = 0$), with a grid size $h_{min} = L_x/200$, consistently with the grid convergence study reported in [15]. At the top boundary ($y = L_y$) the mesh is coarser, with a grid size $h_{max} = 4 \, h_{min}$. At $t = 0$, the interface position is set by $y_\Gamma = (4/64) L_x + (1/64) \, L_x \, \left(\cos(2 \, \pi \, x \, L_x^{-1}) + \cos(2 \, \pi \, (z - 0.5 \, L_z) \, L_z^{-1}) \right)$, with the vapor film in the region $y < y_\Gamma$. Furthermore, both fluids are quiescent at the temperature $T(\mathbf{x}, 0) = T_{sat}$. No-slip boundary condition is set for the velocity at $y = 0$, and $T(\mathbf{x}_w, t) = T_w > T_{sat}$. Symmetry boundary condition is applied at the lateral boundaries, and Neumann conditions are applied at the top boundary.

Figure 1b illustrates the interface time evolution of film boiling at $t^* = t \, t_s^{-1} = \{11.8, 14.2, 28.4\}$, with dimensionless numbers $\mathrm{Pr} = 10$, $\mathrm{Ja} = 1$, $\mathrm{Gr} = 225.1$, $\rho_l/\rho_v = \mu_l/\mu_v = 10$, $\lambda_l/\lambda_v = c_{p,l}/c_{p,v} = 1$. Figure 2a depicts the vorticity vectors ($\nabla \times \mathbf{v}$) on the interface. Figure 2b shows the time evolution of the NUSSELT number ($\mathrm{Nu}^*(t)$) and the time-averaged NUSSELT number (Nu), demonstrating a close agreement against empirical correlations reported by Klimenko [17, 18] and Berenson [19]. Consequently, these numerical results validate and prove the robustness of the parallel UCLS method for film boiling heat transfer.

Acknowledgements

The principal author, N. Balcázar-Arciniega, a Serra Húnter Lecturer (UPC-LE8027), gratefully acknowledges financial support from the Catalan Government through this program. The computing time granted by the RES (IM-2023-2-0009) on the supercomputer MareNostrum IV at the BSC is acknowledged. The authors thankfully acknowledge the financial support of the MCIN/AEI/10.13039/501100011033 under the project PID2020-115837RB-100, Spain.

References

[1] V. P. Carey, Liquid-Vapor Phase-Change Phenomena, CRC Press, 2020. doi:10.1201/9780429082221.

[2] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, 2001. doi:10.1017/CBO9780511975264.

[3] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, Journal of Computational Physics 169 (2001) 708–759. doi:10.1006/jcph.2001.6726.

[4] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–225. doi:10.1016/0021-9991(81)90145-5.

[5] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159. doi:10.1006/jcph.1994.1155.

[6] M. Sussman, E. G. Puckett, A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows, Journal of Computational Physics 162 (2000) 301–337. doi:10.1006/jcph.2000.6537.

[7] N. Balcazar, O. Lehmkuhl, L. Jofre, J. Rigola, A. Oliva, A coupled volume-of-fluid/level-set method for simulation of two-phase flows on unstructured meshes, Computers & Fluids 124 (2016) 12–29. doi:10.1016/j.compfluid.2015.10.005.

[8] E. Olsson, G. Kreiss, A conservative level set method for two phase flow, Journal of Computational Physics 210 (2005) 225–246. doi:10.1016/j.jcp.2005.04.007.

[9] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, J. Rigola, A finite-volume/level-set method for simulating two-phase flows on unstructured grids, International Journal of Multiphase Flow 64 (2014) 55–72. doi:10.1016/j.ijmultiphaseflow.2014.04.008.

[10] N. Balcazar-Arciniega, O. Antepara, J. Rigola, A. Oliva, A level-set model for mass transfer in bubbly flows, International Journal of Heat and Mass Transfer 138 (2019) 335–356. doi:10.1016/j.ijheatmasstransfer.2019.04.008.

[11] N. Balcázar-Arciniega, J. Rigola, C. D. Pérez-Segarra, A. Oliva, Numerical study of the drag force, interfacial area and mass transfer in bubbles in a vertical pipe, Chemical Engineering Journal 495 (2024) 153124. doi:10.1016/j.cej.2024.153124.

[12] D. Juric, G. Tryggvason, Computations of boiling flows, International Journal of Multiphase Flow 24 (1998) 387–410. doi:10.1016/S0301-9322(97)00050-5.

[13] S. W. Welch, J. Wilson, A volume of fluid based method for fluid flows with phase change, Journal of Computational Physics 160 (2000) 662–682. doi:10.1006/jcph.2000.6481.

[14] N. Balcazar, J. Rigola, A. Oliva, Unstructured level-set method for saturated liquid-vapor phase change, In: WCCM-ECCOMAS 2020. Volume 600 - Fluid Dynamics and Transport Phenomena. (2021) 1–12. doi:10.23967/wccm-eccomas.2020.352.

[15] N. Balcazar-Arciniega, J. Rigola, A. Oliva, Unstructured Conservative Level-Set (UCLS) Simulations of Film Boiling Heat Transfer, In: Computational Science – ICCS 2023. ICCS 2023. Lecture Notes in Computer Science 10477 (2023) 318–331. doi:10.1007/978-3-031-36030-5.

[16] N. Balcazar, J. Rigola, J. Castro, A. Oliva, A level-set model for thermocapillary motion of deformable fluid particles, International Journal of Heat and Fluid Flow 62 (2016) 324–343. doi:10.1016/j.ijheatfluidflow.2016.09.015.

[17] V. Klimenko, Film boiling on a horizontal plate — new correlation, International Journal of Heat and Mass Transfer 24 (1981) 69–79. doi:10.1016/0017-9310(81)90094-6.

[18] V. Klimenko, A. Shelepen, Film boiling on a horizontal plate — a supplementary communication, International Journal of Heat and Mass Transfer 25 (1982) 1611–1613. doi:10.1016/0017-9310(82)90042-4.

[19] P. J. Berenson, Film-boiling heat transfer from a horizontal surface, Journal of Heat Transfer 83 (1961) 351–356. doi:10.1115/1.3682280.

[20] N. Balcazar, O. Lehmkuhl, J. Rigola, A. Oliva, A multiple marker level-set method for simulation of deformable fluid particles, International Journal of Multiphase Flow 74 (2015) 125–142. doi:10.1016/j.ijmultiphaseflow.2015.04.009.

[21] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (1992) 335–354. doi:10.1016/0021-9991(92)90240-Y.

[22] N. Balcazar, J. Castro, J. Chiva, A. Oliva, DNS of falling droplets in a vertical channel, International Journal of Computational Methods and Experimental Measurements 6 (2017) 398–410. doi:10.2495/CMEM-V6-N2-398-410.

[23] N. Balcazar-Arciniega, J. Rigola, A. Oliva, DNS of mass transfer in bi-dispersed bubble swarms, In: Computational Science – ICCS 2022. ICCS 2022. Lecture Notes in Computer Science, vol 13353. Springer, Cham 13353 (2022) 284–296. doi:10.1007/978-3-031-08760-8_24.

[24] N. Balcázar-Arciniega, J. Rigola, A. Oliva, Unstructured Flux-Limiter Convective Schemes for Simulation of Transport Phenomena in Two-Phase Flows, 2024, pp. 20–32. doi:10.1007/978-3-031-63783-4_3.

[25] A. J. Chorin, Numerical Solution of the Navier-Stokes Equations, Mathematics of Computation 22 (1968) 745. doi:10.2307/2004575.

[26] N. Balcazar, O. Lehmkuhl, L. Jofre, A. Oliva, Level-set simulations of buoyancy-driven motion of single and multiple bubbles, International Journal of Heat and Fluid Flow 56 (2015). doi:10.1016/j.ijheatfluidflow.2015.07.004.

[27] N. Balcazar, J. Castro, J. Rigola, A. Oliva, DNS of the wall effect on the motion of bubble swarms, Procedia Computer Science 108 (2017) 2008–2017. doi:10.1016/j.procs.2017.05.076.

[28] N. Balcazar-Arciniega, J. Rigola, A. Oliva, DNS of Mass Transfer from Bubbles Rising in a Vertical Channel, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11539 LNCS (2019) 596–610. doi:10.1007/978-3-030-22747-0_45.

[29] N. Balcazar-Arciniega, J. Rigola, A. Oliva, A level-set model for two-phase flow with variable surface tension: Thermocapillary and surfactants, 8th European Congress on Computational Methods in Applied Sciences and Engineering (2022). doi:10.23967/eccomas.2022.011.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

A Parallel-In-Time Spectral Deferred Corrections Method for the Incompressible Navier-Stokes Equations

Abdelouahed Ouardghi[a,*], Robert Speck[a]

[a] Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52428 Jülich, Germany

ARTICLE INFO[†]

Keywords:
Parallel-in-Time;
Spectral Deferred Corrections;
Incompressible Navier-Stokes Equations

ABSTRACT

In this work, spectral deferred corrections (SDC) methods are considered as parallel-in-time integrators for the solution of the unsteady incompressible Navier-Stokes equations. These temporal methods are coupled with a high-order finite element spatial approximation. The goal of this work is to illustrate and analyze the properties of the parallel-in-time method through numerical experiments, including flow past a cylinder (using the standard DFG 2D-3 benchmark) which is selected as an example of unsteady flow.

1. Introduction

The numerical simulation of the Navier-Stokes equations (NSE) in the primitive formulation for incompressible flow is an active research topic. Yet, the challenge of accurately resolving these equations in both space and time necessitates the deployment of advanced high-performance computing systems. While spatial parallelization can reduce the runtime per time step, temporal integration of time-sensitive applications often requires a large number of time steps. Therefore, for further speedup, parallel-in-time integrators are required for more parallelism in the temporal domain. One of the time integrators that can be used to enable efficient parallel-in-time integration is the spectral deferred corrections methods introduced in 2000 by Dutt et al. [1], as a more stable variant of the classical deferred corrections approach for solving ordinary differential equations (ODEs). SDC are an iterative approach for the numerical solution of ordinary differential equations. It works by refining the numerical solution for an initial value problem by performing a series of correction sweeps using a low-order time-stepping method, and can be interpreted as a preconditioned Picard iteration to solve a fully implicit collocation problem. SDC has been widely used for various initial value problems, using explicit, implicit or implicit-explicit Euler and other low-order methods as preconditioner [2, 3, 4, 5]. Moreover, two strategies to enable parallelization across the method for spectral deferred corrections are presented by R. Speck [6]. In this work the SDC methods implemented in pySDC [7] are combined with the finite element method (FEM), as facilitated by the frameworks FEniCS providing a powerful tool for computational fluid dynamics applications. FEM excels at accurately representing complex geometries and boundary conditions, making it an ideal choice for spatial discretization. This coupling results in a robust and accurate numerical solution to the Navier-Stokes equations, benefiting from the parallel computing capabilities of both methods.

2. SDC for Navier-Stokes

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with the boundary $\partial\Omega$ and $[0, T]$ is a time interval. The governing equations consist of the incompressible Navier-Stokes equations

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} &= -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g}, \\ \nabla \cdot \mathbf{u} &= 0, \end{cases} \tag{1}$$

where $\mathbf{u}$ is the velocity vector, p is the pressure, ν is the kinematic viscosity, and $\mathbf{g}$ represents external forces. To solve these equations, we begin by subdividing the time interval into sub-intervals $[t_n, t_{n+1}]$. Then, using Chorin's projection scheme, we first compute an intermediate velocity field $\mathbf{u}^*$ with the momentum equation

$$\begin{cases} \dfrac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} &= -\mathbf{u}^n \cdot \nabla \mathbf{u}^n + \nu \nabla^2 \mathbf{u}^* + \mathbf{g}^{n+1}, \\ u^* &= 0 \qquad \text{on } \partial\Omega. \end{cases} \tag{2}$$

In the next step, the intermediate velocity is projected to the space of divergence free vector fields to get the next update of velocity and pressure

$$\mathbf{u}^{n+1} - \mathbf{u}^* = -\Delta t \nabla p^{n+1}. \tag{3}$$

[†] This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02513 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
a.ouardghi@fz-juelich.de (A. Ouardghi); r.speck@fz-juelich.de (R. Speck)
ORCID(s): 0000-0003-3471-7424 (A. Ouardghi); 0000-0002-3879-1210 (R. Speck)

Taking the divergence and requiring the incompressibility condition $\nabla \cdot \mathbf{u}^{n+1} = 0$, we obtain the equation

$$\Delta t \nabla^2 p^{n+1} = \nabla \cdot \mathbf{u}^*, \tag{4}$$

which is a Poisson problem for the pressure p^{n+1}. Finally, the velocity $\mathbf{u}^{n+1}$ can be corrected using Eq. (3).

2.1. Spatial discretization

To tackle the space discretization, we utilize the mixed finite element approach. We will focus on Eq. (2) as it is the only one requiring to be solved using the SDC methods. The matrix form of a weak formulation for Eq. (2) can be written as

$$[\mathtt{M}]\frac{d\mathbf{u}}{dt} = \mathbf{f}(t,\mathbf{u}) = \mathbf{f}_I(t,\mathbf{u}) + \mathbf{f}_E(t,\mathbf{u}), \tag{5}$$

where $[\mathtt{M}]$ is the finite element mass matrix, $\mathbf{f}_I(t,\mathbf{u}) = -[\mathtt{K}]\mathbf{u}+[\mathtt{M}]\mathbf{g}(t)$ is the implicit part of $\mathbf{f}$ with $[\mathtt{K}]$ is the stiffness matrix, and $\mathbf{f}_E(t,\mathbf{u}) = -[\mathtt{C}(\mathbf{u})]\mathbf{u}$ is the non-linear advection part that should be treated explicitly.

2.2. Parallel-in-time SDC

We write the Picard form of the problem in Eq. (5) as follows

$$\begin{aligned}[\mathtt{M}]\mathbf{u}(t) &= [\mathtt{M}]\mathbf{u}_0 + \int_{t_n}^{t} \mathbf{f}(s,\mathbf{u}(s))ds, \\ & t_n \le t \le t_{n+1},\end{aligned} \tag{6}$$

where $\mathbf{u}_0 = \mathbf{u}(t_n)$. The integral in Eq. (6) can be approximated using a spectral quadrature rule. For this reason we discretize the time interval $[t_n, t_{n+1}]$ using M quadrature nodes such that $t_n < \tau_1 < \cdots < \tau_M = t_{n+1}$. Thus the Eq. (6) can be written as

$$\begin{aligned}[\mathtt{M}]\mathbf{u}_m &= [\mathtt{M}]\mathbf{u}_0 + \Delta t \sum_{j=1}^{M} q_{m,j}\mathbf{f}(\tau_j,\mathbf{u}_j), \\ m &= 1,\dots,M,\end{aligned} \tag{7}$$

where $\mathbf{u}_m \approx \mathbf{u}(\tau_m)$ and $q_{m,j} = \int_{t_0}^{\tau_m} L_j(s)ds, \quad j = 1,\cdots,M$ are the set of quadrature weights with L_j is the Lagrange polynomial defined on the quadrature node $\{\tau_j\}$. Next, we combine these M equations into one system of linear or non-linear equations which is called the collocation problem

$$(\mathcal{M} - \Delta t \mathbf{Q}\mathbf{F})(\vec{\mathbf{u}}) = \mathcal{M}\vec{\mathbf{u}}_0, \tag{8}$$

where $\mathcal{M} = [\mathtt{M}] \otimes \mathbf{I}_M$, $\vec{\mathbf{u}} = (\mathbf{u}_1,\cdots,\mathbf{u}_M)^T$, $\vec{\mathbf{u}}_0 = (\mathbf{u}_0,\cdots,\mathbf{u}_0)^T$, $\mathbf{Q} = (q_{ij})_{i,j}$ is the matrix gathering the quadrature weights and the vector function $\mathbf{F}$ is given by $\mathbf{F}(\vec{\mathbf{u}}) = \big(\mathbf{f}(\mathbf{u}_1),\cdots,\mathbf{f}(\mathbf{u}_M)\big)$. Moreover, if $d \ge 2$ then $\mathbf{Q}$ and $\mathcal{M}$ must be replaced by $\mathbf{Q}\otimes\mathbf{I}_d$ and $\mathcal{M}\otimes\mathbf{I}_d$, respectively. It should be noted that this system of equation is equivalent to the fully implicit Runge-Kutta method with $\mathbf{Q}$ being the Butcher tableau. Generally, this system is dense and requires an iterative solution. SDC can be presented as precondtionated Picard iteration for the collocation problem in Eq. (8) as

$$\begin{aligned}\big(\mathcal{M} - \Delta t\mathbf{Q}_\Delta\mathbf{F}\big)(\vec{\mathbf{u}}^{k+1}) &= \mathcal{M}\vec{\mathbf{u}}_0 \\ &\quad +\Delta t\big(\mathbf{Q}-\mathbf{Q}_\Delta\big)\mathbf{F}(\vec{\mathbf{u}}^k).\end{aligned} \tag{9}$$

The ODE (5) contains both stiff and non-stiff components. Therefore, semi-implicit SDC (SISDC) is considered for the solution of Eq. (5). Consequently, the iteration in Eq. (9) becomes

$$\begin{aligned}&\big(\mathcal{M} - \Delta t\mathbf{Q}_{\Delta,I}\mathbf{F}_I - \Delta t\mathbf{Q}_{\Delta,E}\mathbf{F}_E\big)(\vec{\mathbf{u}}^{k+1}) \\ =\ & \mathcal{M}\vec{\mathbf{u}}_0 + \Delta t\big(\mathbf{Q}-\mathbf{Q}_{\Delta,I}\big)\mathbf{F}_I(\vec{\mathbf{u}}^k) \\ &\quad +\Delta t\big(\mathbf{Q}-\mathbf{Q}_{\Delta,E}\big)\mathbf{F}_E(\vec{\mathbf{u}}^k),\end{aligned} \tag{10}$$

where $\mathbf{Q}_{\Delta,I}$ and $\mathbf{Q}_{\Delta,E}$ are lower and strictly lower triangular matrices It has been shown in [6] that SDC can be parallelize over the quadrature nodes using diagonal preconditioners (i.e., diagonal matrix $\mathbf{Q}_\Delta$). In this work the following diagonal preconditioners are considered: $\mathbf{Q}_{\Delta,I} = \mathbf{Q}_\Delta^{IEpar}$, $\mathbf{Q}_\Delta^{MIN}$, $\mathbf{Q}_\Delta^{MIN\_SR\_S}$ and $\mathbf{Q}_\Delta^{MIN\_SR\_NS}$, (see [6, 8] for more details). Given that the matrix $\mathbf{Q}_{\Delta,E}$ must be lower triangular, the null matrix $\mathbf{Q}_{\Delta,E} = \mathbf{0}$ is considered for the explicit part.

3. Numerical results

In this section, we will consider the benchmark of flow past a cylinder. The geometry and parameters are taken from the DFG 2D-3 benchmark in FeatFlow, see Fig. 1. The inflow velocity profile is

$$\mathbf{u}_{in} = \left(\frac{4Uy(0.41-y)}{0.41^2}, 0\right) \tag{11}$$

$$U = U(t) = 1.5\sin\left(\frac{\pi t}{8}\right) \tag{12}$$

and the kinematic viscosity is given by $\nu = 0.001$.

In Fig. 2 we show the lift coefficients in the unsteady Navier-Stokes equations subject to the time-dependent inflow profile $\mathbf{u}_{in}$. These results are computed using 2, 3, and 5 Gauß–Radau nodes with two different time steps $\Delta t = 1/1{,}600$ and $\Delta t = 1/1{,}000$. The data provided by FEATFLOW is also included in the figure as a reference solution. Based on Fig. 2, it is clear that the different SDC methods solves accurately the Navier-Stokes equations. It should be mentioned that there are slight differences between the results obtained using $\Delta t = 1/1{,}600$ and $\Delta t = 1/1{,}000$ because of the CFL constraint required by the semi-implicit scheme. Moreover, the average number of iterations for five different choices of $\mathbf{Q}_{\Delta,I}$ after 50 time steps at four different time points throughout the simulation are displayed in Fig. 3. These choices are the LU trick ($\mathbf{Q}_\Delta^{LU}$) as reference as well

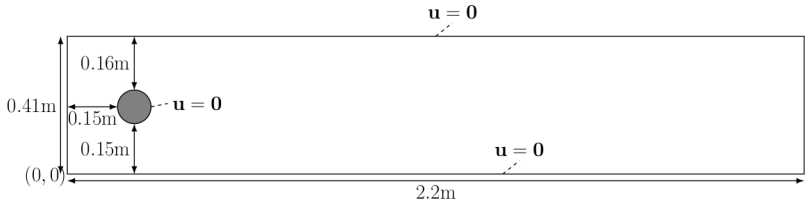


Figure 1: Computational geometry for the DFG 2D-3 benchmark.

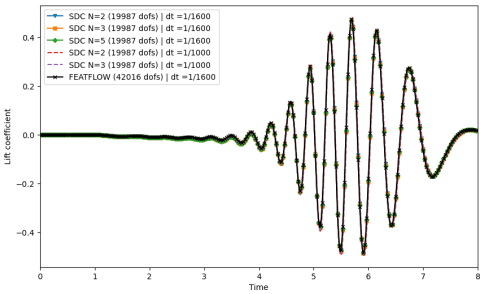


Figure 2: Lift coefficient using different SDC methods and different time steps.

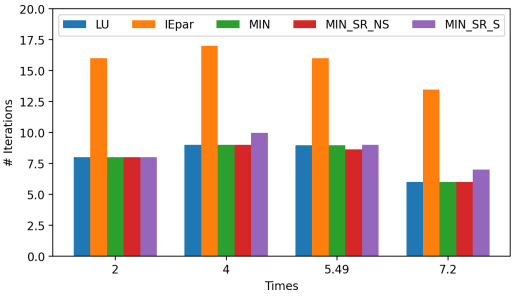


Figure 3: Average number of iterations needed by different choices of preconditioner.

as the four diagonal matrices listed in the previous section. From this figure, it can be clearly seen that the standard option is the best choice for all tests. It can be also seen that the $\mathbf{Q}_\Delta^{IEpar}$ requires at least double the number of iterations to converge, while the minimization-based preconditioners $\mathbf{Q}_\Delta^{MIN}$ and $\mathbf{Q}_\Delta^{MIN\_SR\_NS}$ seem to be reliable and converge about as fast as the standard choice $\mathbf{Q}_\Delta^{LU}$ for the considered problem. However, the choice $\mathbf{Q}_\Delta^{MIN\_SR\_S}$ requires slightly more iterations to converge.

4. Conclusion

Combining SDC with FEM accurately solves the Navier-Stokes equations. Diagonal preconditioners enable parallel SDC, and with minimization-based preconditioners, diagonal SDC converges as fast as standard SDC. The future goal is to use a fully implicit monolithic scheme for the NSE to avoid the CFL limitations of semi-implicit schemes.

References

[1] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT Numerical Mathematics 40 (2000) 241–266. doi:10.1023/A:1022338906936.

[2] D. Ruprecht, R. Speck, Spectral deferred corrections with fast-wave slow-wave splitting, SIAM Journal on Scientific Computing 38 (4) (2016) A2535–A2557. doi:10.1137/16M1060078.

[3] A. C. Hansen, J. Strain, On the order of deferred correction, Applied Numerical Mathematics 61 (8) (2011) 961–973. doi:10.1016/j.apnum.2011.04.001.

[4] I. Akramov, S. Götschel, M. Minion, D. Ruprecht, R. Speck, Spectral deferred correction methods for second-order problems, SIAM Journal on Scientific Computing 46 (3) (2024) A1690–A1713. doi:10.1137/23M1592596.

[5] L. Micalizzi, D. Torlo, A new efficient explicit deferred correction framework: Analysis and applications to hyperbolic pdes and adaptivity, Communications on Applied Mathematics and Computation (2023). doi:10.1007/s42967-023-00294-6.

[6] R. Speck, Parallelizing spectral deferred corrections across the method, Computing and Visualization in Science 19 (2018) 75–83. doi:10.1007/s00791-018-0298-x.

[7] R. Speck, Algorithm 997: pySDC - prototyping spectral deferred corrections, ACM Transactions on Mathematical Software (TOMS) 45 (3) (2019). doi:10.1145/3310410.

[8] G. Čaklović, T. Lunet, S. Götschel, D. Ruprecht, Improving efficiency of parallel across the method spectral deferred corrections (2024). arXiv:2403.18641.

Other Topic 4:
Scalable Solvers

Achieving scalability of methods for numerical simulations is crucial for ever-growing problem sizes and for effectively utilizing high-performance computing (HPC) systems for this purpose. With new appearing hardware technologies, novel algorithms need to be designed. This session brings together contributions to the *ParCFD International Conference 2024* dealing with scalable solvers running on HPC systems, e.g., using parallel computing for space-time simulations and domain decomposition methods in aeroacousics.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

An In-House Overset Supersonic Solver With Grid Refinement Capability on Parallel Environment

Mohamad El Hajj Ali Barada[a,*], Bayram Çelik[b]

[a]Istanbul Technical University, Aeronautical and Astronautical Engineering Program, Istanbul, Turkey
[b]Istanbul Technical University, Department of Astronautical Engineering, Istanbul, Turkey

ARTICLE INFO[†]

Keywords:
Finite Volume;
Domain Decomposition;
Grid Adaptation;
Overset;
p4est

ABSTRACT

A three-dimensional finite volume method solver that runs on an adaptive overset mesh is developed for supersonic flows. The solver utilizes the open source libraries of p4est and OpenFOAM utilities to generate, maintain, and readapt the computational grids in a distributed memory architecture during parallel runs supported by MPI. We achieve to assemble the overset grid using our newly developed in-house strategy and code. We validated the solver by considering experimental data available in the literature for a supersonic flow over a cylinder with a hemisphere tip.

1. Introduction

The major challenges in hypersonic flow regimes are due to the determination of heat loads and its prevention. Dealing with hypersonic flow problems also requires modeling complex flow physics such as shock-shock and shock-boundary layer interactions. Considering complex geometries, the computational cost may considerably scale to unfeasible levels. One can overcome the cost challenge by combining an efficient adaptive mesh refinement (AMR) solver and composite grid strategies also known as overset grids [1] such that the grid is strategically distributed to resolve off body flow regions of interest without compromising the near body grid quality. This approach has been previously attempted and developed upon in the literature. Kenway et al. [2] has proposed a method to efficiently assemble overset grid system in parallel. Peron et al. [3] has proposed an octree data structure to manage the background grid and its adaptation. While some studies has attempted developing efficient parallel overset assembly and management methods that may be extended to structured background grids such as in [4, 5], to our knowledge the integration of p4est, a highly scalable, multi-octree parallel management library [6] to a block based dynamic AMR capable overset finite volume method Navier Stokes supersonic solver in a distributed parallel environment has not been attempted.

In this paper, we outline the process of developing and integrating a parallel solver with overset based AMR capabilities through the combined use of open source libraries p4est, MPI, and meshing/partitioning tools of OpenFOAM [7].

2. Numerical Strategy

2.1. Governing equations, numerical method, and base solver

In the present study, viscous turbulent compressible flows are considered. Favre averaged Navier Stokes equations and turbulence model k-omega SST [8] are preferred to model the flow.

The base solver used in the study is our in-house finite volume Navier-Stokes solver [9]. An upwind flux splitting scheme and the 2nd order central scheme are used to compute convection and diffusion fluxes. The face reconstruction is 1st order. Time integration is performed using Euler explicit scheme. 1st order direct interpolation, a non-conservative interpolation scheme, is used to combine separate grids types. Grids of different refinement level are treated as hanging nodes at their interfaces which is conservative.

2.2. Generation, assembly, partitioning and adaptation of the grid system

An overset grid system that consists of two separate structured grids, a conformal and a Cartesian grids is used in the present study. The former conforms to the geometry and resolves near body flow physics. Extending to the far field, the latter engulfs the former and fills the interior of the geometry. The latter consists of a collection of grid blocks with own refinement level. The conformal grid is generated and partitioned using OpenFOAM grid tools; blockMesh and decomposePar. The Cartesian grid is generated and partitioned using p4est, a C based library that manipulates a collection of connected adaptive octrees, called forest.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02514 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
mbarada@itu.edu.tr (M. El Hajj Ali Barada); celikbay@itu.edu.tr (B. Çelik)
ORCID(s): - (M. El Hajj Ali Barada); 0000-0002-7025-6330 (B. Çelik)

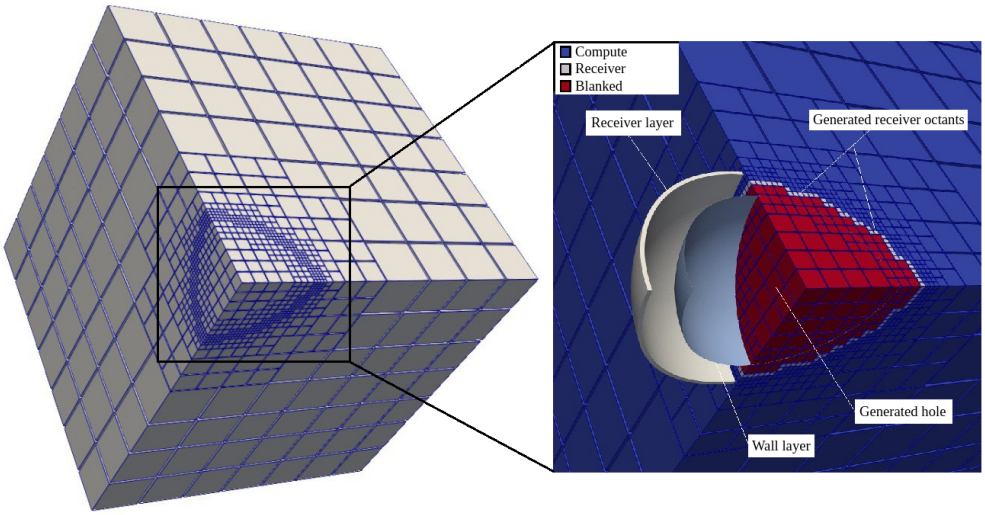


Figure 1: p4est forest (left) and it's overset assembly (right) for a conformal grid over a sphere.

The overset grid assembly, OGA, uses an in-house method, "halo". Briefly, the method identifies a seed octant in the p4est forest. Starting from the seed, a flooding process is initiated. The flood excludes octants within the geometry and up to the outmost cell layer of the conformal grid, the receiver layer, see Fig. 1. Then, the grid blocks are generated.

p4est allows for block based mesh adaptation through a collection of subroutines. It also allows for balancing where neighboring blocks are of maximum 1 refinement level difference. In the developed solver, balancing is called after coarsening and refinement. A balancing call after coarsening may lead to re-refinement of a recently coarsened grid blocks and thereby loss of required resolution. The solver avoids this by temporarily storing severed children data. The solver identifies block grids for adaptation using a sensor function; the undivided difference [10]. The function output is compared against a threshold range where cells with values below or over that range are marked for refinement or coarsening. The threshold range is set using an iterative bi-sectioning method [3].

2.3. Solver parallelization and communication process

Solver parallelization is achieved by implementing three communication modes. Those are inner Conformal (C1), Inner Cartesian (C2), and Conformal donor or receiver front (C3) grid communication, see Fig. 2. The mode C1 is done using a communication table which associates neighboring cells and their target/source cores. Mode C2 is achieved via p4est where it's subroutines generate ghost octants and exchanges them between neighboring octants cores. Finally, C3 is done using a gather & broadcast method; the donor or receiver cells data are exchanged to a master core which then submits the data to all other cores. Methods in C1 and C3 are explicitly implemented using MPI subroutines.

Through using p4est's ghost octants generation and exchange subroutines as well as by enforcing a sufficient number of recursion for the flooding process, the solver successfully implements OGA and AMR process on parallel environment.

3. Solver Validation and Conclusions

An experimental study in the literature [11], where a supersonic turbulent flow over a hemisphere tip cylinder is considered for validation. The flow is at MACH 2 and REYNOLDS number that is equal to $1.31 \cdot 10^6$. An overset grid is generated with a starting total cell count that is equal to 348,461. A case with a classical grid approach, a single body conforming grid is also considered. The cases are run on a cluster with Intel(r) Xeon(r) gold 6130 CPU @ 2.10GHz. 12 cores. The obtained pressure coefficient distribution shows that the solver achieves satisfactory results against the reference data with an error of 4.5% at the stagnation point. It also successfully tracks and captures the shock properly, see Fig. 3.

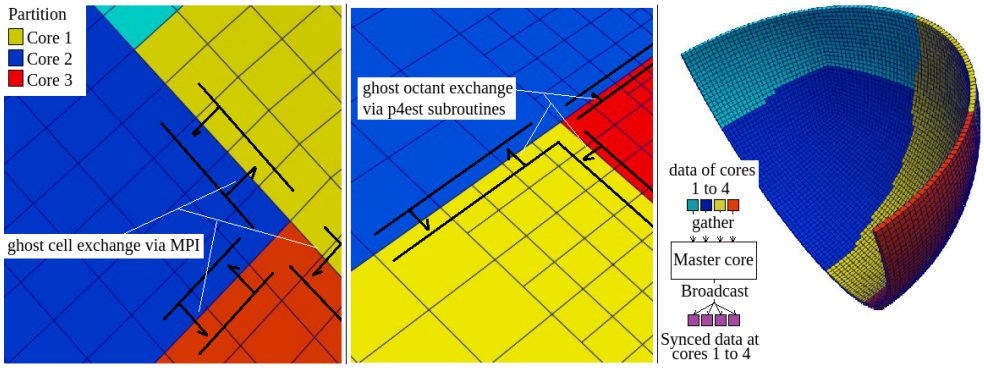


Figure 2: Communication modes C1 (left) C2 (middle) and C3 (right).

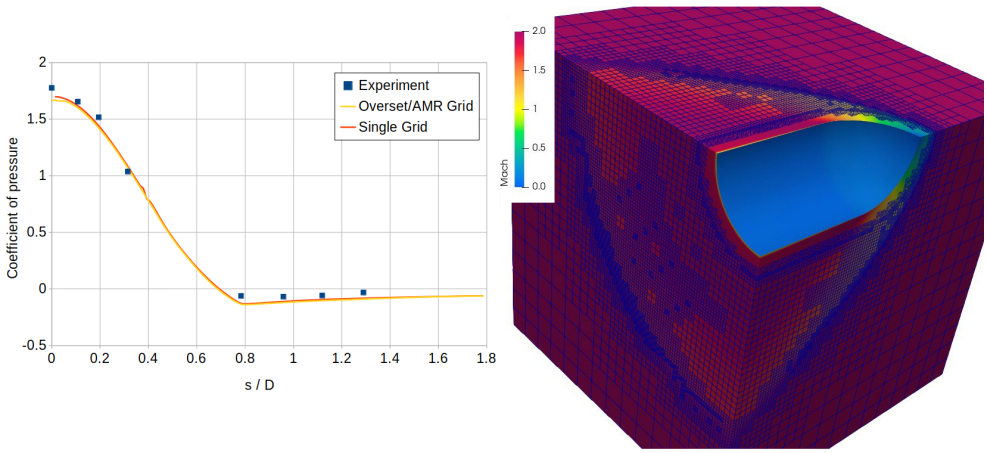


Figure 3: Coefficient of pressure plot against [11] along the surface of the hemisphere tipped cylinder (left) and MACH contour plot in overset grid system view (right).

A preliminary strong scaling parallel performance test is considered. The case is run on a computing node with 32 cores. The performance of sonicFoam that is one of OpenFOAM supersonic solvers is used for comparison with a single grid of equivalent total cell count. The comparison of the solver performance shows similar trend in parallel efficiency and speedup factor. Our in-house solver exhibits a slightly lower efficiency and speedup factor, see Fig. 4. This is likely due to the added interpolation stage in overset. On performing a high AMR cycle setting test, it can be seen that the cell count is reasonably equally distributed between all cores. Hence, load balance is maintained by p4est, see Fig. 5. The small difference in cell per core is attributed to partitioning bias rule where related children are held on the same core.

In summary, we developed an overset based AMR capable parallel solver which uses p4est and an in-house overset grid assembling utilities. The solver is validated against experimental data in the literature, hemisphere tipped cylinder. The obtained preliminary results prove that the developed solver and its capabilities are promising in terms of mesh adaptation, parallel scalability and load balancing.

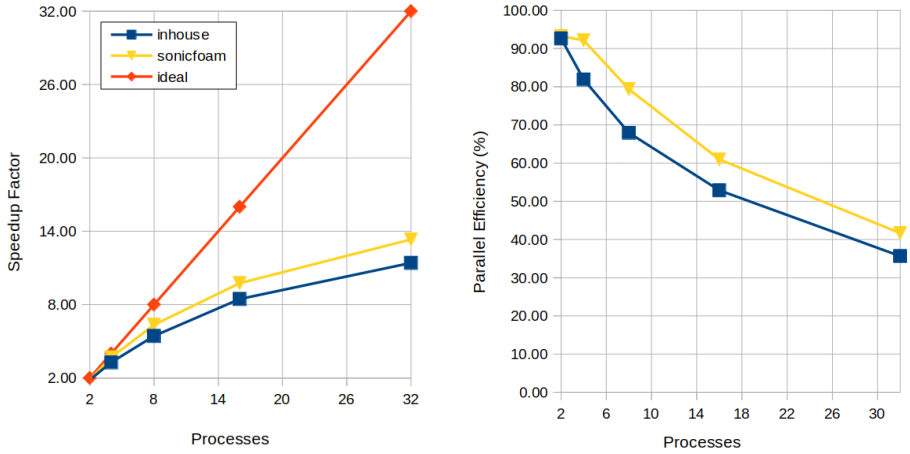


Figure 4: Speedup factor (left) and parallel efficiency plots (right) of the inhouse and sonicFoam solvers.

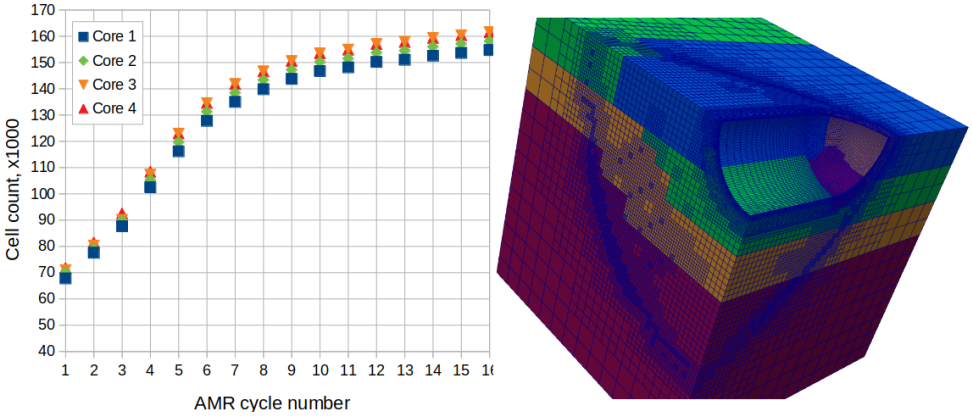


Figure 5: Cell count per core during AMR cycles (left) and domain decomposition plot after AMR cycle 16 (right).

References

[1] J. Benek, J. Steger, F. Dougherty, A flexible grid embedding technique with application to the Euler equations. doi:10.2514/6.1983-1944.

[2] G. K. Kenway, A. Mishra, N. R. Secco, K. Duraisamy, J. R. R. A. Martins, An Efficient Parallel Overset Method for Aerodynamic Shape Optimization. doi:10.2514/6.2017-0357.

[3] S. Péron, C. Benoit, Automatic off-body overset adaptive cartesian mesh method based on an octree approach, Journal of Computational Physics 232 (1) (2013) 153–173. doi:10.1016/j.jcp.2012.07.029.

[4] M. Hedayat, A. M. Akbarzadeh, I. Borazjani, A parallel dynamic overset grid framework for immersed boundary methods, Computers and Fluids 239 (2022) 105378. doi:10.1016/j.compfluid.2022.105378.

[5] Enhancement on parallel unstructured overset grid method for complex aerospace engineering applications, Chinese Journal of Aeronautics 36 (1) (2023) 115–138. doi:10.1016/j.cja.2022.07.015.

[6] C. Burstedde, L. C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133. doi:10.1137/100791634.

[7] H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Computer in Physics 12 (6) (1998) 620–631. doi:10.1063/1.168744.

[8] F. R. Menter, M. Kuntz, R. Langtry, K. Hanjalic, Y. Nagano, M. J. Tummers, Ten years of industrial experience with the SST turbulence model, Turbulence, heat and mass transfer 4 (1) (2003) 625–632.

[9] M. El Hajj Ali Barada, B. Celik, An overset grid based adaptive mesh refinement algorithm, in: 10th Ankara International Aerospace Conference, 2019, pp. AIAC–2019–104.
URL http://aiac.ae.metu.edu.tr/paper.php/AIAC-2019-104

[10] P. G. Buning, T. H. Pulliam, Near-body grid adaption for overset grids, in: 46th AIAA Fluid Dynamics Conference, pp. AIAA 2016–3326. doi:10.2514/6.2016-3326.

[11] J. White, Application of navier-stokes flowfield analysis to the aerothermodynamic design of an aerospike-configured missile, in: Aerospace Design Conference 1993, pp. AIAA–93–0968. doi:10.2514/6.1993-968.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Roadmap for Extreme-Scale Simulations: On the Evolution of Poisson Solvers

F. Xavier Trias[a,*], Àdel Alsalti-Baldellou[a,b] and Assensi Oliva[a]

[a]Technical University of Catalonia, Heat and Mass Transfer Technological Center, C/Colom 11, 08222 Terrassa (Barcelona), Spain
[b]Department of Information Engineering, University of Padova, Via Giovanni Gradenigo, 6b, 35131 Padova PD, Italy

ARTICLE INFO†

Keywords:
Poisson's Equation;
Turbulence;
Large-Scale Simulations;
Direct Numerical Simulation;
Linear Solvers

ABSTRACT

We aim to answer the following question: *is the complexity of numerically solving Poisson's equation increasing or decreasing for very large DNS and LES simulations of incompressible flows?* Physical and numerical arguments are combined to derive power-law scalings at very high REYNOLDS numbers. Theoretical convergence analysis for both Jacobi and multigrid solvers defines a two-dimensional phase space divided into two regions depending whether the number of solver iterations tend to decrease or increase with the REYNOLDS number. Numerical results seem to confirm that we are in the latter region, i.e., in the foreseeable future the numerical complexity of solving Poisson's equation will increase and, therefore, better and better preconditioning techniques will be needed.

1. Introduction: Two competing effects

The never-ending increasing capacity of modern HPC systems enables DNS simulations at higher and higher REYNOLDS numbers, $Re = Ul/\nu$. The number of grid points, N_x, and time-steps, N_t, can be estimated with the classical Kolmogorov theory (K41)

$$\begin{aligned} N_x^{K41} &= \frac{L_x}{\Delta x} \sim \frac{l}{\eta} \sim Re^{3/4}, \\ N_t^{K41} &= \frac{t_{\text{sim}}}{\Delta t} \sim \frac{t_l}{t_\eta} \sim \frac{l}{\eta}\frac{u}{U} \sim Re^{3/4} Re^{-1/4} = Re^{1/2}, \end{aligned} \tag{1}$$

where L_x and t_{sim} are the domain size and the time integration period, which are assumed to be similar to the size of the largest scales, l, and its corresponding characteristic time, $t_l \sim l/U$, *i.e.* $L_x \sim l$ and $t_{\text{sim}} \sim t_l$. For a DNS, we assume that $\Delta x \sim \eta$ and $\Delta t \sim t_\eta \sim \eta/u$, where $t_\eta \sim \eta/u$ and u are the characteristic time and velocity of the Kolmogorov scales, η. Plugging this into the CFL condition, *i.e.* $\Delta t^{\text{conv}} \sim \Delta x/U$ and $\Delta t^{\text{diff}} \sim \Delta x^2/\nu$ leads to

$$\begin{aligned} N_t^{\text{conv}} &\sim \frac{t_l}{\Delta t^{\text{conv}}} \sim \frac{l}{U}\frac{U}{l Re^{-3/4}} = Re^{3/4}, \\ N_t^{\text{diff}} &\sim \frac{t_l}{\Delta t^{\text{diff}}} \sim \frac{l}{U}\frac{\nu}{l^2 (Re^{-3/4})^2} = Re^{1/2}. \end{aligned} \tag{2}$$

Therefore, we can conclude that

$$\Delta t/t_l \sim 1/N_t \sim Re^{\alpha}, \tag{3}$$

where $\alpha = -1/2$ for the K41 theory, see Eq. (1), or diffusion dominated, see Eq. (2) (right), and $\alpha = -3/4$ for convection dominated, see Eq. (2) (left). Therefore, higher Re lead to (i) larger meshes and (ii) smaller time-steps, Δt. These are two competing effects on the convergence of Poisson's equation: namely, the former increases the condition number of the discrete Poisson equation whereas the latter leads to better initial guess. *Who is eventually the winner at very high Re? Read on...*

2. Analysis of the residual of Poisson's equation

Although FFT-based direct solvers are very well-suited for canonical flows with periodic directions [1], the forthcoming analysis assumes that multigrid (MG) methods will eventually be the preferred option for extreme-scale simulations. Then, the next step is to analyze the residual of the Poisson's equation as a function of the REYNOLDS number. Relevant aspects are twofold: the magnitude and the spectral distribution. To study them, we consider a fractional step method where $\boldsymbol{u}^p$ is the predictor velocity. Imposing that $\nabla \cdot \boldsymbol{u}^{n+1} = 0$, leads to a Poisson equation for pressure, p^{n+1},

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^p - \Delta t \nabla p^{n+1} \quad \overset{\nabla\cdot}{\Longrightarrow} \quad \nabla^2 p^{n+1} = 1/\Delta t \nabla\cdot\boldsymbol{u}^p. \tag{4}$$

†This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02515 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
francesc.xavier.trias@upc.edu (F.X. Trias); adel.alsalti@upc.edu (À. Alsalti-Baldellou); assensi.oliva@upc.edu (A. Oliva)
ORCID(s): 0000-0002-5966-0703 (F.X. Trias); 0000-0002-5331-4236 (À. Alsalti-Baldellou); 0000-0002-2805-4794 (A. Oliva)

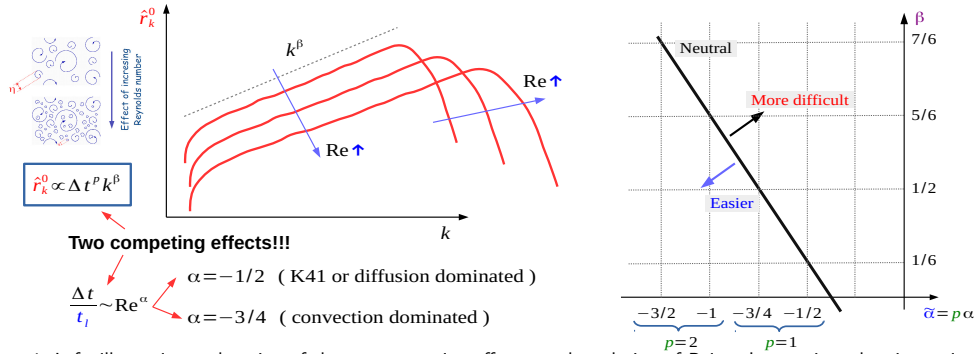


Figure 1: Left: illustrative explanation of the two competing effects on the solution of Poisson's equation when increasing *Re* number: time-step, Δt, decreases whereas the range of scales increases. Right: $\{\tilde{\alpha}, \beta\}$ phase space. Solid black line corresponds to $\propto Re^0$ in Eqs. (15) and (16), *i.e.* neutral effect of *Re*-number in the total number of iterations.

Assuming $\nabla \cdot \boldsymbol{u}^n = 0$ and taking p^n as initial guess, we obtain the following initial residual

$$\begin{aligned} r^0 &= \nabla^2 p^n - \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^{p,n+1} \\ &\overset{(4)}{=} \frac{1}{\Delta t} \left(\nabla \cdot \boldsymbol{u}^{p,n} - \nabla \cdot \boldsymbol{u}^{p,n+1} \right) \\ &\approx \partial_t \nabla \cdot \boldsymbol{u}^p. \end{aligned} \tag{5}$$

Alternatively, we can also consider $\tilde{r}^0 = \Delta t r^0$. In this case, the residual reads

$$\begin{aligned} \tilde{r}^0 &= \nabla^2 \tilde{p}^n - \nabla \cdot \boldsymbol{u}^{p,n+1} \\ &\overset{(4)}{=} \left(\nabla \cdot \boldsymbol{u}^{p,n} - \nabla \cdot \boldsymbol{u}^{p,n+1} \right) \\ &\approx \Delta t \partial_t \nabla \cdot \boldsymbol{u}^p, \end{aligned} \tag{6}$$

where $\tilde{p} = p\Delta t$ is a pseudo-pressure. Notice that the second residual, $\tilde{r}^0$, is more meaningful from a physical point-of-view, since it directly translates how accurately we impose the incompressibility constraint. Then, recalling that $\nabla \cdot \boldsymbol{u}^p$ can be expressed as follows [2]

$$\nabla \cdot \boldsymbol{u}^p \approx \Delta t \nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) = 2\Delta t Q_{\mathsf{G}}, \tag{7}$$

leads to

$$r^0 \approx 2\Delta t \ \partial_t Q_{\mathsf{G}} \qquad \text{and} \qquad \tilde{r}^0 \approx 2\Delta t^2 \partial_t Q_{\mathsf{G}}, \tag{8}$$

where $Q_G = -1/2 tr(\mathsf{G}^2)$ is the second invariant of the velocity gradient, $\mathsf{G} \equiv \nabla \boldsymbol{u}$. Therefore, smaller Δt decrease the magnitude of r^0 (also $\tilde{r}^0$) leading to a better convergence.

On the other hand, increasing *Re* also leads to finer meshes, see Eq. (1), and, therefore, to more ill-conditioned systems with a wider and wider range of scales to be resolved. In the forthcoming analysis, the spectral distribution of the initial residual, $\hat{r}_k^0$, plays a crucial role. In general, we can assume a power-law scaling within the inertial range

$$\partial_t Q_{\mathsf{G}} \propto k^\beta \qquad \Longrightarrow \qquad \hat{r}_k^0 \propto \Delta t^p k^\beta, \tag{9}$$

where k is the wave number and $p \in \{1, 2\}$ depends on the definition of the residual: $p = 1$ for Eq. (5) and $p = 2$ for Eq. (6). Then, a power-law scaling for Q_{G} can be derived from Eqs. (4) and (7), and the $k^{-7/3}$ scaling of the shell-summed squared pressure spectrum [3],

$$(\hat{Q}_{\mathsf{G}})_k \propto k^2 (k^{-7/3})^{1/2} = k^{5/6}. \tag{10}$$

Then, the value of β in Eq. (9) can be estimated from the dynamics of the invariants obtained from the so-called restricted Euler equations [4],

$$\partial_t Q_{\mathsf{G}} = -(\boldsymbol{u} \cdot \nabla) Q_{\mathsf{G}} - 3 R_{\mathsf{G}}, \tag{11}$$

where $R_{\mathsf{G}} = det(\mathsf{G}) = 1/3 tr(\mathsf{G}^3)$ is the third invariant of G. The two terms in the right-hand-side of this equation are expected to have different power-law scalings. Namely,

$$\begin{aligned} (\widehat{(\boldsymbol{u} \cdot \nabla) Q_{\mathsf{G}}})_k \propto (\widehat{\nabla Q_{\mathsf{G}}})_k \propto k (k^{5/6}) &= k^{11/6}, \\ (\hat{R}_{\mathsf{G}})_k \propto (k^{5/6})^{3/2} &= k^{5/4}. \end{aligned} \tag{12}$$

where the Taylor's frozen-turbulence hypothesis is applied to approximate $(\boldsymbol{u} \cdot \nabla) Q_{\mathsf{G}}$, which is eventually the dominant term in the right-hand-side of Eq. (11). Combining this with the results obtained in Eqs. (9) and (12) leads to

$$\boxed{\begin{aligned} &\hat{r}_k^0 \propto \Delta t^p k^\beta \\ &\quad \text{with} \quad \beta = 11/6 \\ &\quad \text{and} \quad p = \left\{ \begin{array}{ll} 1 & \text{if } \hat{r} \text{ defined as Eq. (5)} \\ 2 & \text{if } \hat{r} \text{ defined as Eq. (6)} \end{array} \right\} \end{aligned}} \tag{13}$$

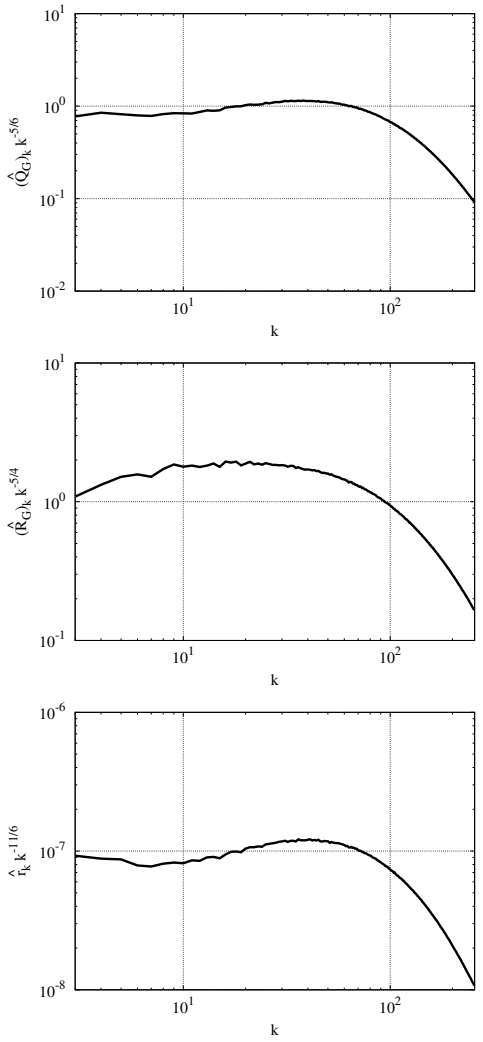


Figure 2: From top to bottom: compensated spectra for pressure, Q_G, R_G, and the initial solver residual, r_o. Results correspond to forced homogeneous isotropic turbulence at $Re_\lambda = 325$. Compensation factors corresponds to those derived in Eqs. (10), (12) and (13), respectively.

In summary, there are two competing effects (see Fig. 1, left) when increasing Re number: time-step, Δt, decreases whereas the range of scales increases. The next step is to analyze how the solver convergence is affected.

3. Analysis of the solver convergence and conclusions

We want to study whether the number of iterations inside the Poisson's solver increases or decreases with Re. To do so, we can relate the L2-norm of the residual with the integral of $\hat{r}_k$ for all the wave numbers using the Parseval's theorem, *i.e.* $||r||^2 = \int_\Omega r^2 dV = \int_1^{k_{\max}} \hat{r}_k^2 dk$, where $k_{\max} \approx 1/\eta \sim Re^{3/4}$. Then, the residual after n iterations can be computed as

$$||r^n||^2 = \int_1^{k_{\max}} \left(\hat{\omega}_k^n \hat{r}_k^0\right)^2 dk \overset{(3)(13)}{\approx} \int_1^{Re^{3/4}} \hat{\omega}_k^{2n} Re^{2\tilde{\alpha}} k^{2\beta} dk, \tag{14}$$

where $\hat{\omega}_k = \hat{r}_k^{n+1}/\hat{r}_k^n$ is the convergence ratio of the solver and $\tilde{\alpha} = p\alpha$. For instance, for a Jacobi solver, $\hat{\omega}_k = \cos(\frac{\pi}{2}\rho)$ where $\rho \equiv k/k_{\max}$. In this case, using a quadratic approximation of $\cos(x) \approx 1 - 4x^2/\pi^2$ leads to

$$||r^n||^2 \approx \frac{Re^{2(\tilde{\alpha}+3/4(\beta+1/2))}}{2(2n+1)}. \tag{15}$$

We can extend this analysis for a MG solver with the Jacobi smoother

$$||r^n||^2 \approx \frac{Re^{2(\tilde{\alpha}+3/4(\beta+1/2))}}{2(2n+1)} \left\{ \left(\sum_{l=0}^{l_{\max}} \frac{(3/4)^{2n+1}}{2^{2l}} \right) + \frac{1}{2^{2l_{\max}+1}} \right\}. \tag{16}$$

where $l_{\max} \sim \log_2 N_x \sim (3/4)\log_2 Re$ is the number of levels. Compared to Eq. (15), MG is strongly accelerated by the term in brackets, which tends to $(3/4)^{2n}$. Nevertheless, the Re-scaling is the same; therefore, the regions defined in the $\{\tilde{\alpha}, \beta\}$ phase space remain unchanged (see Fig. 1, right). Numerical results displayed in Fig. 2 seem to confirm our theory: namely, the slopes of the invariants Q_G and R_G correspond well with the values predicted in Eqs. (10) and (12), respectively. More importantly, the slope of the solver residual, $\hat{r}_k$, fits with the predicted value of $\beta = 11/6$ (see Eq. (13). Altogether leads to the preliminary conclusion that the number of iterations tends to increase with Re.

Acknowledgements

This work was financially supported by two competitive R+D projects: RETOtwin (PDC2021-120970-I00) and SIMEX (PID2022-142174OB-I00), both given by *Ministerio de Ciencia e Innovación* MCIN/AEI/ 10.13039/ 501100011033 and European Union Next GenerationEU. Àdel Alsalti-Baldellou is also supported by the project "National Centre for HPC, Big Data and Quantum Computing",

CN00000013 (approvato nell'ambito del Bando M42C – Investimento 1.4 – Avvisto "Centri Nazionali" – D.D. n. 3138 del 16.12.2021, ammesso a finanziamento con Decreto del MUR n. 1031 del 17.06.2022). Calculations were carried out on the MareNostrum 4 supercomputer at the Barcelona Supercomputing Center. We thankfully acknowledge these institutions.

References

[1] F. X. Trias, M. Soria, C. D. Pérez-Segarra, A. Oliva, A Direct Schur-Fourier Decomposition for the Efficient Solution of High-Order Poisson Equations on Loosely Coupled Parallel Computers, Numerical Linear Algebra with Applications 13 (2006) 303–326. doi:10.1002/nla.466.

[2] S. B. Pope, Turbulent Flows, Vol. 1, Cambridge University Press, 2000. doi:10.1088/1468-5248/1/1/702.

[3] D. I. Pullin, Pressure spectra for vortex models of fine-scale homogeneous turbulence, Physics of Fluids 7 (1995) 849. doi:10.1063/1.868608.

[4] B. J. Cantwell, Exact solution of a restricted Euler equation for the velocity gradient tensor, Physics of Fluids A 4 (1992) 782–793. doi:10.1063/1.858295.

ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Domain Decomposition Method for Equivalent Sources Method in Aeroacoustic

Naima Débit[a], Roland Denis[a], Benoit Fabrege[a] and Damien Tromeur-Dervout[a,*]

[a] Université Claude Bernard Lyon 1, CNRS, Institut Camille Jordan, UMR5208, 69100 Villeurbanne, France

ARTICLE INFO†

Keywords:
Parallel Methodology;
Domain Decomposition Method;
Equivalent Source Method;
Model Order Reduction

ABSTRACT

This contribution focuses on a Schwarz-type domain decomposition method for solving an ill-conditioned dense linear problem overdetermined with complex coefficients arising from the inverse problem of determining, by the method of equivalent sources, the aeroacoustic noise generated by a fluid on the surface of a body.

1. Introduction

The Equivalent Source Method (ESM) aiming at the simulation of realistic Frequency Response Functions (FRF) substitutes the acoustic behavior of a radiating object by a set of N_s acoustic monopoles q calibrated with respect to the boundary condition v on its skin. The method is a meshless approach so that it is easy and simple to implement, and it does not have a numerical singularity problem that occurs in the boundary element method [1]. Such a method allows to perform 3D Conventional Beamforming (CBF) with FRF considering the acoustic environment and the influence structure [2, 3].

The set ω of N boundary points at position $(r_j)_{j\leq N}$ on the object skin and the set Ω of N_s equivalent sources at the position $(r_l)_{l\leq N_s}$, allow to build the matrix coefficients of the transfer function for the wave number k between the equivalent sources and the normal velocity, with θ_{jl} the angle of $r_j - r_l$ and the normal to the object skin at r_j, as

$$A_{jl} = \frac{e^{ik||r_j-r_l||_2}}{4\pi||r_j-r_l||_2^2}(1-ik||r_j-r_l||_2)\cos(\theta_{jl}).$$

This contribution presents a Schwarz-type domain decomposition method for solving this overdetermined, dense and ill-conditioned linear system $Aq = v, A \in \mathbb{C}^{N\times N_s}$. The ESM suffers from the numerical instability that is associated with the ill-conditioned matrix A due to the random distribution of equivalent sources [1, 4].

† This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02516 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
naima.debit@univ-lyon1.fr (N. Débit);
roland.denis@math.univ-lyon1.fr (R. Denis);
benoit.fabrege@math.univ-lyon1.fr (B. Fabrege);
damien.tromeur-dervout@univ-lyon1.fr (D. Tromeur-Dervout)
ORCID(s): 0000-0001-6761-5470 (N. Débit); 0000-0002-9276-4512 (R. Denis); - (B. Fabrege); 0000-0002-0118-8100 (D. Tromeur-Dervout)

2. Schwarz-type domain decomposition

The Restricted Additive Schwarz domain decomposition method (DDM), see, e.g., [5] and references therein, with a well-posed linear system has convergence/divergence properties allowing its acceleration of the convergence to the correct solution. Here, difficulties arise because the matrix A is a complex non-square matrix, and it is also a dense matrix, where each source contributes to each control point. Consequently, there is no natural interface to control the splitting of the domain(s).

2.1. Schwarz DDM based on the geometrical splitting

To define the DDM partitioning in m pairs $\{(\omega_l, \Omega_l), l = 1, \dots, m\}$ of subdomains overlapping or not, ω and Ω are split as follows: from a mesh of the surface (where the vertices are the control points), and the equivalent sources within the object, we divide the surface in m partitions using the METIS graph partitioning library, based on the vertices adjacency. Then, we assign to each equivalent source the rank of the nearest control point using a K-D tree.

With the given partitioning, we iteratively solve Eq. (1), where A_{ij} represents the block of the original matrix with respect to the partition renumbering and A_{ii}^+ is the pseudo inverse of the block A_{ii}, in order to update the iterated solution $x_i^{(k+1)}$ with respect to the iterated solution $x^{(k)}$, i.e.,

$$x_i^{(k+1)} = A_{ii}^+\left(b_i - \sum_{1=j\neq i}^{m} A_{ij}x_j^{(k)}\right). \quad (1)$$

Nevertheless, the resulting method is a divergent method (even with the addition of a relaxation parameter where it converges before diverging). The pure linear convergence/divergence is lost due to ill-conditioning and solving of Eq. (1) in the least squares sense. Moreover, the singular value decomposition (SVD) of the A_{ij} blocks shows that some of them contribute strongly to the b_i RHS, see Fig. 2a.

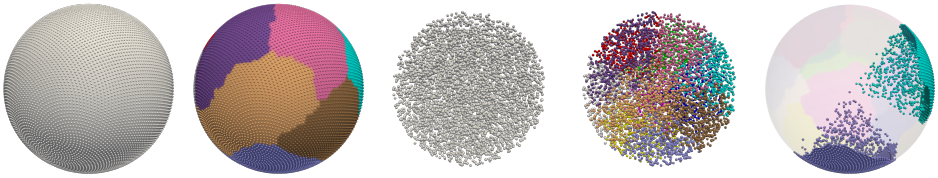


Figure 1: From the left to the right: surface ω of control points, non-overlapping partition of ω, volume Ω of randomly distributed equivalent sources, partitioning of the volume with respect to the ω partitioning, example of two resulting subdomain pairs (ω_l, Ω_l).

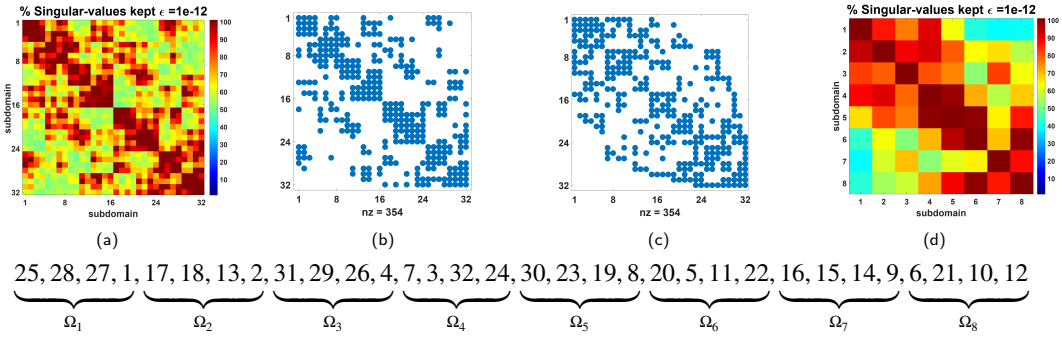


Figure 2: Numerical criterion splitting process starting from a geometrical partitioning of 32 partitions to obtain a partitioning with 8 partitions.

2.2. Numerical criterion splitting

Figure 2 shows the partitioning based on a numerical criterion. The idea is to take into account the factor $1/r^2$ in the matrix coefficients. Starting from the geometrical partitioning with a greater number of partitions than the expected final number, we compute the SVD of each block, see Fig. 2a, and define the threshold criterion τ as % of the number of singular values greater than a given ϵ (here $\epsilon = 10^{-12}$). It produces an adjacency matrix (b) of blocks satisfying this criterion, see Fig. 2b. We reduce the profile of this adjacency matrix with the reverse Cuthill MacKnee's algorithm, see Fig. 2c. We then gather the adjacent blocks to define the new partitioning, see Fig. 2d.

2.3. Projected Schwarz algorithm to handle over-determined systems

We use the SVD $U_{ji}S_{ji}V_{ji}^* = A_{ji}$ of each block A_{ji} of the new partitioning to define a projected Schwarz algorithm where the right hand side of the local problem is projected on the left singular vectors associated with singular values greater than ϵ of each block acting on the components of $x_i^{(k+1)}$. This leads to solving a compressed system with a ξ percentage of the N rows of the original one. The solution of the i^{th} partition reads

$$\bar{A}_i x_i^{(k+1)} = \mathbb{U}_i^* \left(b - \tilde{A}\right), \tag{2}$$

with

$$\bar{A}_i = \begin{pmatrix} S_{1i}V_{1i}^* \\ \vdots \\ S_{mi}V_{mi}^* \end{pmatrix}, \quad \mathbb{U}_i^* = \begin{pmatrix} U_{1i}^* & & \\ & \ddots & \\ & & U_{mi}^* \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

and

$$\begin{aligned} \tilde{A} &= A\left(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, 0_i, x_{i+1}^{(k)}, \ldots, x_m^{(k)}\right)^t \\ &= \begin{pmatrix} A_{11} \cdots A_{1i-1} & A_{1i+1} \cdots A_{1m} \\ \vdots & \vdots \\ A_{m1} \cdots A_{mi-1} & A_{mi+1} \cdots A_{mm} \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ \vdots \\ x_{i-1}^{(k)} \\ x_{i+1}^{(k)} \\ \vdots \\ x_m^{(k)} \end{pmatrix}. \end{aligned}$$

This compressed local system Eq. (2) is solved by SVD with a better conditioning than the full system with about 10 orders

$N_s = 3{,}200$	1	2	3	4	5	6	7	8
1	**390**	352	313	360	252	166	151	159
2	369	**409**	365	364	319	257	264	224
3	323	325	**408**	316	302	211	327	249
4	347	373	321	**394**	393	290	223	275
5	258	335	314	393	**401**	402	254	344
6	171	254	212	276	381	**404**	273	398
7	151	244	285	206	243	259	**395**	351
8	166	224	240	284	360	400	340	**399**
$N = 6{,}400$	**2,175**	**2,516**	**2,458**	**2,593**	**2,651**	**2,389**	**2,227**	**2,399**
$\xi = 100\%$	34%	39%	38%	41%	41%	37%	35%	37%

Table 1
Size of system $\bar{A}_i$ for the sphere problem, $N_s = 3{,}200$, $N = 6{,}400$, $k = 400\,\text{Hz}$, $\epsilon = 10^{-12}$, $\tau = 0.8$.

$log_{10}(cond(A))$	$log_{10}(cond(\bar{A}_i))$							
	1	2	3	4	5	6	7	8
18.87	9.37	7.98	9.12	7.65	7.86	8.71	9.02	7.88

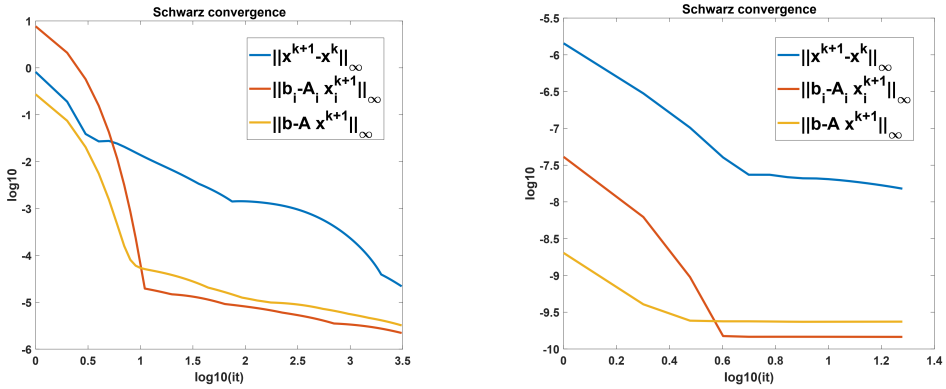
Table 2
The quantity $log_{10}(cond(\bar{A}_i))$ for the sphere problem, $N_s = 3{,}200$, $N = 6{,}400$, $k = 400\,\text{Hz}$, $\epsilon = 10^{-12}$, $\tau = 0.8$.

of magnitude, see Tab. 2. Recalling the SVD complexity of $O(N\,N_s\,\min(N, N_s))$ for $A \in \mathbb{C}^{N\times N_s}$, we have for the SVD computing of $\bar{A}_i$ a theoretical numerical speed-up of m/ξ and m^2/ξ if it is done in parallel over m processes.

3. First results and conclusion

We consider a monopole scattered by a rigid sphere problem as described in [2].

Table 1 gives the size of the $\bar{A}_i$ system after compression, for the sphere problem with: $N_s = 3{,}200$, $N = 6{,}400$, $k = 400\,\text{Hz}$, $\epsilon = 10^{-12}$, $\tau = 0.8$ and the last row represents the ξ percentage of compression of $\bar{A}_i$ with respect to the original size of $N = 6{,}400$. The value of ξ is between 34% and 41%. Table 1 also gives details of the size of the compressed system resulting from the A_{ji} block. For example, A_{81} has 399 rows and the compressed block $S_{81}V_{81}^*$ has 166 rows.



(a) $x^{(0)} = 0$.

(b) SVD solution of $Ax^{(0)} = b$.

Figure 3: Convergence of the Schwarz DDM on 8 partitions with respect to the iterations for two initial guesses for the monopole scattered by a rigid sphere with $N_s = 3{,}200$, $N = 6{,}400$, $k = 400\,\text{Hz}$, $\epsilon = 10^{-12}$, $\tau = 0.8$.

Table 2 exhibits the conditioning number for $\bar{A}_i$ for the sphere problem. They are up to ten orders lower than for the original full system.

Figure 3 shows the convergence behavior of the projected Schwarz algorithm with respect to the iterations. It exhibits a fast convergence starting from the arbitrary initial guess $x^{(0)} = 0$ followed by a slow convergence, see Fig. 3a, and an improvement on the solution starting from the initial guess solution of $Ax^{(0)} = b$ computed by SVD , see Fig. 3b. Let us notice that the algorithm is still stable after the convergence.

The talk will also focus on the numerical and parallel performances of the proposed projected Schwarz algorithm on larger size problems using the PETSc and SLEPc libraries for the implementation.

References

[1] S. Lee, Review: The Use of Equivalent Source Method in Computational Acoustics, Journal of Computational Acoustics 25 (1) (Mar 2017). doi:10.1142/S0218396X16300012.

[2] J. Chambon, T. Le Magueresse, O. Minck, J. Antoni, Three-dimensional beamforming for wind tunnel applications using ESM based transfer functions, in: Proceedings on CD of the 8th Berlin Beamforming Conference, 2-3 March, 2020, 2020.
URL https://www.bebec.eu/fileadmin/bebec/downloads/bebec-2020/papers/BeBeC-2020-S09.pdf

[3] J. Chambon, J. Antoni, S. Bouley, Galerkin equivalent sources method for sound field reconstruction around diffracting bodies, J. Acoust. Soc. Am. 152 (4) (2022) 2042–2053. doi:10.1121/10.0014422.

[4] Y. Bobrovnitskii, T. Tomilina, General-Properties and Fundamental Errors of the Method of Equivalent Sources, Acoustical Physics 41 (5) (1995) 649–660.

[5] L. Berenguer, D. Tromeur-Dervout, Sparse Aitken–Schwarz domain decomposition with application to Darcy flow, Computers & Fluids 249 (2022) 105687. doi:10.1016/j.compfluid.2022.105687.

Epilogue

This proceedings document is a collection of all the accepted mauscripts submitted to the *35th Parallel CFD International Conference 2024*. The presentations on the conference and the content of the papers clearly show that the field of CFD is transitionining to an era, where new computing and AI technologies, and advanced tool sets play a key role in reaching unprecendent accuracy, speed, and confidence. These technologies become more and more interwined and open up opportunities for simulating more realistic scenarios and for further research. It is up to the CFD research community to advertise these new technologies and support bringing them into pratical application in industry. This document and all the containing manuscripts are licensed under CC-BY 4.0.

The next *36th Parallel CFD International Conference 2025* will take place in Merida, Yucatan, Mexico, from Nov. 24-26, 2025. The *ParCFD 2025* Chair Dr. Juan Carlos Cajas García, ENES-Mérida, UNAM, the local organization committee, comprised of Dr. Erick Salcedo Álvarez, ENES-Mérida, UNAM, Dr. Juan Manuel Rivero, ENES-Mérida, UNAM, M.C. Edwin Enrique Pérez R., ENES-Mérida, UNAM, and Dr. Carlos Francisco Brito L., Universidad Autónoma de Yucatán, the *ParCFD* Committee, and the mini-symposia organizers are looking forward to meet you at the next event. More information can be found on the website of *ParCFD 2025*[4].

[4]ParCFD 2025 https://www.parcfd2025.org

Advertisement

EVIDEN

Eviden is a global leader in Quantum, AI, and high-performance computing (HPC), offering deep expertise across industries and scientific domains.

The Center for Excellence in Performance Programming (CEPP) is a team of expert engineers dedicated to helping organizations optimize their HPC applications. By combining cutting-edge computing technologies with industrial and scientific expertise, CEPP helps clients achieve their business and research objectives.

The main goal of the CEPP is to optimize HPC applications across a wide range of fields. It brings together interdisciplinary teams to analyze, profile, and fine-tune codes for both current and next-generation architectures. While the CEPP has expertise in many areas, CFD is a major focus, providing performance engineering services for fluid dynamics simulations — from weather and climate modeling to aerospace and energy.

CEPP contributes to key European initiatives such as EPI-SGA2, EUPEX, ESiWACE3, and ChEESE-2P, working with partners like ECMWF on scalable climate and geophysics codes. Through partnerships with Intel, Nvidia, AMD, Arm, Mellanox, and SiPearl, CEPP ensures early access to emerging technologies.

Whether you're optimizing CFD codes or other HPC applications, CEPP is here to support your performance goals.

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 55
Gradient-Free Optimization of Artificial and Biological Networks using Learning to Learn
A. Yeğenoğlu (2023), II, 136 pp
ISBN: 978-3-95806-719-6

Band / Volume 56
Real-time simulations of transmon systems with time-dependent Hamiltonian models
H. A. Lagemann (2023), iii, 166, XXX pp
ISBN: 978-3-95806-720-2

Band / Volume 57
Plasma Breakdown and Runaway Modelling in ITER-scale Tokamaks
J. Chew (2023), xv, 172 pp
ISBN: 978-3-95806-730-1

Band / Volume 58
Space Usage and Waiting Pedestrians at Train Station Platforms
M. Küpper (2023), ix, 95 pp
ISBN: 978-3-95806-733-2

Band / Volume 59
Quantum annealing and its variants: Application to quadratic unconstrained binary optimization
V. Mehta (2024), iii, 152 pp
ISBN: 978-3-95806-755-4

Band / Volume 60
Elements for modeling pedestrian movement
from theory to application and back
M. Chraibi (2024), vi, 279 pp
ISBN: 978-3-95806-757-8

Band / Volume 61
Artificial Intelligence Framework for Video Analytics:
Detecting Pushing in Crowds
A. Alia (2024), xviii, 151 pp
ISBN: 978-3-95806-763-9

Band / Volume 62
The Relationship between Pedestrian Density, Walking Speed and Psychological Stress:
Examining Physiological Arousal in Crowded Situations
M. Beermann (2024), xi, 117 pp
ISBN: 978-3-95806-764-6

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 63
Eventify Meets Heterogeneity:
Enabling Fine-Grained Task-Parallelism on GPUs
L. Morgenstern (2024), xv, 110 pp
ISBN: 978-3-95806-765-3

Band / Volume 64
Dynamic Motivation in Crowds: Insights from Experiments and Pedestrian Models for Goal-Directed Motion
E. Üsten (2024), ix, 121 pp
ISBN: 978-3-95806-773-8

Band / Volume 65
Propagation of Stimuli in Crowds:
Empirical insights into mutual influence in human crowds
H. Lügering (2024), xi, 123 pp
ISBN: 978-3-95806-775-2

Band / Volume 66
Classification of Pedestrian Streams: From Empirics to Modelling
J. Cordes (2024), vii, 176 pp
ISBN: 978-3-95806-780-6

Band / Volume 67
Optimizing Automated Shading Systems in Office Buildings by Exploring Occupant Behaviour
G. Derbas (2024), 9, x, 168, ccxxiii
ISBN: 978-3-95806-787-5

Band / Volume 68
Speed-Density Analysis in Pedestrian Single-File Experiments
S. Paetzke (2025), XIII, 107 pp
ISBN: 978-3-95806-818-6

Band / Volume 69
Proceedings of the 35th Parallel Computational Fluid Dynamics International Conference 2024
A. Lintermann, S. S. Herff, J. H. Göbbert (2025), xv, 321 pp
ISBN: 978-3-95806-819-3

Weitere ***Schriften des Verlags im Forschungszentrum Jülich*** unter
http://wwwzb1.fz-juelich.de/verlagextern1/index.asp

IAS Series
Band / Volume 69
ISBN 978-3-95806-819-3

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum