Weiterentwicklung des Analysewerkzeugs SUSA

Adaptive Monte-Carlo-Simulation basierend auf maschinellen Lernalgorithmen und Entwicklungen zur Plattformunabhängigkeit

GRS - 634

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Weiterentwicklung des Analysewerkzeugs SUSA

Adaptive Monte-Carlo-Simulation basierend auf maschinellen Lernalgorithmen und Entwicklungen zur Plattformunabhängigkeit

Martina Kloos
Nadine Berner
Jörg Peschke
Josef Scheuer

Januar 2021

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für Wirtschaft und Energie (BMWi) unter dem Förderkennzeichen RS1559 durchgeführt.

GRS - 634
ISBN 978-3-949088-23-0

Deskriptoren

Adaptive Monte-Carlo-Simulation, Decision Tree, Gauß Prozess, Genetischer Algorithmus, Maschinelles Lernen, Random Forest, Stützvektorregression, Subset-Simulation

Inhaltsverzeichnis

1 Einleitung

Die GRS entwickelt und verwendet seit vielen Jahren das Analysewerkzeug SUSA (*S*oftware for *U*ncertainty and *S*ensitivity *A*nalyses), um damit die mit einem Simulationsergebnis verbundene Unsicherheit zu quantifizieren und die Hauptursachen dieser Unsicherheit zu ermitteln. Auch außerhalb der GRS wird SUSA von vielen in- und ausländischen Institutionen genutzt.

Die in SUSA bisher bereitgestellten Methoden basieren auf Wahrscheinlichkeitsrechnung, klassischer Monte-Carlo-Simulation und statistischen Verfahren. Auf der Basis von Wahrscheinlichkeitsverteilungen für die nicht eindeutig festlegbaren (unsicheren) Eingangsparameter eines Simulationsprogramms können mit SUSA mögliche Wertekombinationen für diese Parameter ausgespielt und damit entsprechende Simulationsläufe gestartet werden. Die erzielten Simulationsergebnisse können dann bzgl. sicherheitsrelevanter Fragestellungen statistisch ausgewertet werden. So kann mit SUSA u. a. ein Toleranzintervall berechnet werden, das einen hohen Anteil (i. Allg. ≥ 95 %) der möglichen Werte eines sicherheitsrelevanten Simulationsergebnisses mit hoher statistischer Sicherheit (i. Allg. ≥ 95 %) abdeckt. Damit leistet SUSA eine wichtige und effiziente Unterstützung beim Nachweis der Sicherheitskriterien im Rahmen von sogenannten Best Estimate Plus Uncertainty (BEPU) Analysen. Außerdem lassen sich mit den in SUSA implementierten Methoden der Sensitivitätsanalyse die unsicheren Eingangsparameter mit den größten Einflüssen auf die mit einem Simulationsergebnis verbundenen Unsicherheit bestimmen.

Die der BEPU-Analyse zugrunde liegenden Sicherheitskriterien implizieren, dass diese Analyse in erster Linie darauf abzielt, ein Intervall abzuschätzen, das einen hohen Anteil (≥ 95 %) der möglichen Werte eines Simulationsergebnisses abdeckt. Es ist weder das Ziel, die Wahrscheinlichkeit zu schätzen, mit der das Simulationsergebnis einen Grenzwert überschreitet, noch den Bereich der Eingabeparameterwerte (kritischer Parameterbereich) zu lokalisieren, der zu dem unerwünschten Ergebnis führt. Die Rechenkosten dafür wären viel zu hoch, da die Überschreitung eines Grenzwerts normalerweise ein seltenes Ereignis ist und deshalb eine große Zahl von Simulationsläufen im Rahmen der BEPU Analyse mit klassischer Monte-Carlo-Simulation erforderlich ist.

Da genauere Kenntnisse über den kritischen Parameterbereich dazu beitragen können, die Anlagensicherheit noch besser zu bewerten und weitere Indikatoren zum Nachweis des sicheren Betriebs zu ermitteln, wurden zwei verschiedene adaptive Sampling-

Ansätze entwickelt, um diesen Bereich zu ermitteln und seine Wahrscheinlichkeit abzuschätzen. In Abschnitt 2 werden diese auf maschinellen Lernalgorithmen basierenden Ansätze vorgestellt.

Ein anderer Bereich, der insbesondere für Analysen von Unfällen von großem Interesse ist, ist der Parameterbereich, der zu Cliff-Edge-Effekten (CEE) führt. In diesem Zusammenhang beziehen sich CEE auf nichtlineares Systemverhalten, bei dem kleine Abweichungen in den Eingabeparameterwerten das System von unkritischen zu kritischen Zuständen und umgekehrt ändern können. Ein Ansatz zur Lokalisierung der CEE-Region kann dazu beitragen, wirksame Gegenmaßnahmen zu entwickeln, um Unfallfolgen zu verhindern oder abzuschwächen. In Abschnitt 3 wird ein solcher Ansatz beschrieben. Er verwendet genau wie die Ansätze in Bezug auf den kritischen Parameterbereich maschinelle Lernalgorithmen.

Abschnitt 4 geht auf die Anwendung von zwei Ansätzen zusammen mit dem Simulationscode ATHLET ein. Ziel des Anwendungsbeispiels war die Ermittlung der Wahrscheinlichkeit des Parameterbereichs, der bei einem Kühlmittelverluststörfall zu maximalen Hüllrohrtemperaturen über 1200 °C führt. Ein weiteres Ziel war die genauere Charakterisierung dieses kritischen Parameterbereichs.

SUSA setzt sich aus einem Hauptmodul und mehreren Kernmodulen zusammen. Das Hauptmodul übernimmt mit seiner grafischen Benutzeroberfläche die Aufgaben eines übergeordneten Anwendungsprogramms. Es ruft unter anderem die Kernmodule auf, die für die Berechnungen zuständig sind. Während die Kernmodule (in Fortran oder Python) plattformunabhängig eingesetzt werden können, stand das Hauptmodul (Visual Basic (VB) .NET) und damit die volle Funktionalität von SUSA lange Zeit nur unter Windows-basierten Betriebssystemen zur Verfügung. Ein Überblick über diese Windows-basierte SUSA-Version ist in Abschnitt 5 zu finden.

Durch die Entwicklung einer Python-basierten Anwendungsplattform (Python Notebooks) stehen aktuell alle Komponenten für Validierungs-Tests und konkrete Anwendungen sowohl unter Windows als auch unter Unix/Linux-basierten Betriebssystemen zur Verfügung. Auf dieser Basis können die entwickelten Methoden schnell überprüft und nach Bedarf ergänzt werden, um in die plattformunabhängige SUSA-Version nachhaltig integriert zu werden. In Abschnitt 6 wird der aktuelle Stand der plattformunabhängigen SUSA-Version vorgestellt.

2 Adaptive Monte-Carlo-Simulation zur Lokalisierung kritischer Parameterbereiche

In diesem Abschnitt werden zwei praktikable Ansätze für eine adaptive Monte-Carlo-Simulation (MCS) vorgestellt. Ziel dieser Ansätze ist die Bestimmung des mit einem kritischen Simulationsergebnis verbundenen Parameterbereichs und seine Wahrscheinlichkeit. Ein kritisches Simulationsergebnis tritt dann ein, wenn eine sicherheitsrelevante Größe des simulierten physikalischen Prozesses einen Grenzwert verletzt und damit z. B. der Ausfall einer Komponente oder eines Systems verbunden ist.

Der adaptive Ansatz, der in Abschnitt 2.1 beschrieben wird, basiert auf einem Subset-Simulationsansatz in Verbindung mit einem aus der Stützvektorregression hergeleiteten Ersatzmodell (Metamodell) für den eigentlichen Simulationscode. Der Ansatz in Abschnitt 2.2 verwendet einen genetischen Algorithmus und ein Ensemble von verschiedenen maschinellen Lernalgorithmen (hauptsächlich Klassifikationsverfahren) kombiniert mit einem Bayes'schen Ansatz.

2.1 Subset-Simulation mit einem auf Stützvektorregression basierenden Metamodell (SuSSVR-Verfahren)

Der hier beschriebene Ansatz verwendet ein auf Stützvektorregression (SVR) basierendes Metamodell (/VAP 95/, /SMO 04/), um schnell das Ergebnis des eigentlichen Simulationscodes vorherzusagen. Das SVR-Metamodell wird innerhalb einer Subset-Simulation (SuS) angewendet (/AUS 01/, /PAP 15/), um Kandidaten für solche Kombinationen von Eingabeparametern zu gewinnen, die potenziell zu einem kritischen Simulationsergebnis (Über- bzw. Unterschreiten von Grenzwerten) führen können. Der eigentliche Simulationscode wird basierend auf einer kleinen Auswahl dieser Kandidaten ausgeführt. Die Ergebnisse der Simulationsläufe und die zugehörigen Vektoren von Eingabeparameterwerten werden dem Trainingsdatenpool hinzugefügt, der zur weiteren Anpassung des SVR-Metamodells verwendet wird.

Der anfängliche Trainingsdatenpool besteht aus einer Zufallsstichprobe von Eingabeparametervektoren und den jeweiligen Ergebniswerten des Simulationscodes. Sobald neue Parametervektoren mittels des SuS-Verfahrens und dem aktuell vorliegenden SVR-Modell ausgewählt und die dazugehörigen Simulationsergebnisse des eigentlichen Codes berechnet wurden, wird der Pool erweitert und das SVR-Modell neu angepasst.

Am Ende des iterativen Prozesses steht ein relativ robustes SVR-Modell zur Verfügung. Dieses wird innerhalb eines umfangreichen SuS-Verfahrens angewendet, um eine große Auswahl von Parametervektoren aus dem kritischen Bereich zu generieren sowie die jeweilige Wahrscheinlichkeit abzuschätzen.

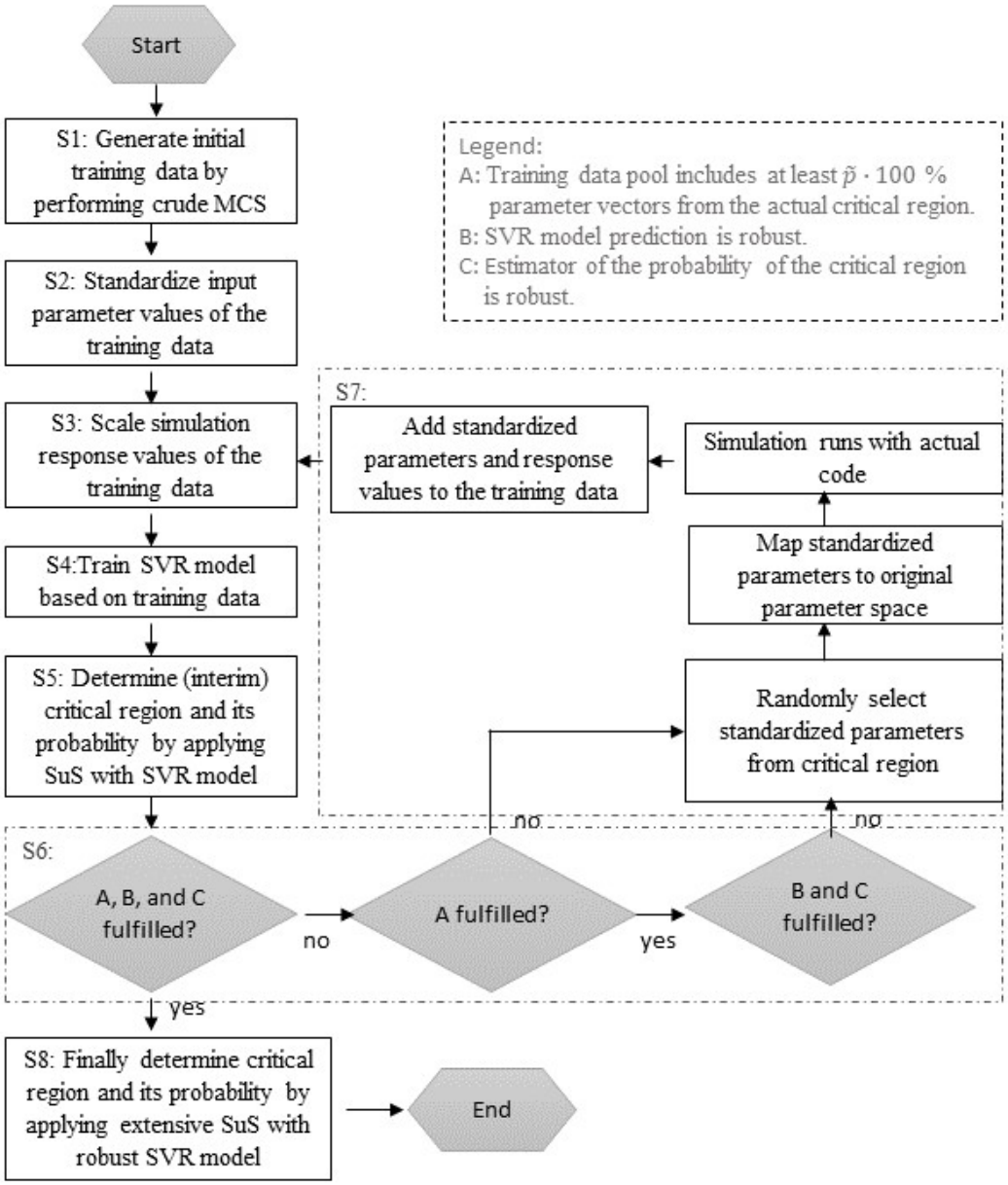

Abb. 2.1 Flussdiagramm der adaptiven MCS mit dem SuSSVR-Verfahren

Das Flussdiagramm des SuSSVR-Verfahrens ist in Abb. 2.1 dargestellt. Eine detaillierte Beschreibung dieses iterativen Verfahrens ist in Abschnitt 2.1.1 zu finden. Die Kernideen für die Konstruktion des SVR-Metamodells und des SuS-Verfahrens werden in den Abschnitten 2.1.2 und 2.1.3 erläutert. Ein Anwendungsbeispiel ist in Abschnitt 2.1.4 beschrieben.

2.1.1 Iteratives Verfahren

Die Schritte des iterativen SuSSVR-Verfahrens können wie folgt beschrieben werden:

(S1) Generieren der anfänglichen Trainingsdaten durch klassische MCS

Der anfängliche Datenpool besteht aus einer Zufallsauswahl von Parametervektoren $x \in R^m$ und der zugehörigen Auswahl von Ergebnissen $r_0(x)$ des eigentlichen Simulationscodes.

(S2) Standardisieren der Eingabeparameterwerte des Trainingsdatenpools

Standardisierte Eingabeparameter sind erforderlich, um zu vermeiden, dass Eingabeparameter, die sich um Größenordnungen von anderen unterscheiden, die SVR-Regression in Schritt (S4) dominieren und deshalb einen überschätzten Einfluss haben, wenn das SVR-Metamodell angewendet wird. Darüber hinaus basiert das SuS-Verfahren in Schritt (S5) auf standardnormalverteilten Parametern.

Jeder Vektor x aus dem Parameterraum X wird in einen Vektor z von unabhängigen standardnormalverteilten Parametern aus dem Raum Z transformiert. Wenn Assoziationen zwischen den Parametern bestehen, wird die Nataf-Transformation /NAT 62/, /LEB 09/ angewendet, um z zu bekommen. Es sollte jedoch berücksichtigt werden, dass es durch die Nataf-Transformation zu einer Verzerrung kommen kann, wenn sich die Assoziationsstruktur von einer Gaußschen Assoziationsstruktur unterscheidet.

(S3) Skalieren der Simulationsergebnisse des Trainingsdatenpools

Jedes Simulationsergebnis $r_0(x)$ wird so transformiert, dass man den Wert der Grenzzustandsfunktion (limit state function, LSF) $g_0(x)$ erhält:

$$g_0(x) = r_{th} - r_0(x) \tag{2.1}$$

wobei r_{th} der für $r_0(x)$ angegebene Grenzwert ist (z. B. r_{th} = 1200 C für PCT).

Unter Verwendung der LSF ist die kritische Parameterregion D definiert als:

$$D = \{x \in R^m : g_0(x) \leq 0\} \tag{2.2}$$

Die LSF im Parameterraum Z unabhängiger standardnormalverteilter Parameter (siehe Schritt (S2)) ist gegeben durch

$$g(z) = r_{th} - r(z) \tag{2.3}$$

wobei $r(z) = r_0\left(F^{-1}(\Phi(z))\right)$, $F^{-1} =$ Inverse der multivariaten Verteilung des Parametervektors x und $\Phi =$ Verteilungsfunktion der Standardnormalverteilung.

Jeder LSF-Wert $g(z)$ wird zusätzlich auf $y = {(g(z) - \overline{y_0})}/{s_{y_0}}$ skaliert, wobei $\overline{y_0}$ der Mittelwert und s_{y_0} die Standardabweichung der $g(z)$-Werte im anfänglichen Trainingsdatenpool ist. Skalierte Simulationsergebnisse werden für die in Schritt (S4) angewendete SVR-Modellkonstruktion und den in Schritt (S5) angewendeten SuS-Algorithmus benötigt.

(S4) Erstellen (trainieren) des SVR-Metamodells basierend auf standardisierten/skalierten Trainingsdaten

Eine genaue Beschreibung zur Konstruktion eines SVR-Metamodells wird in Abschnitt 2.1.2 gegeben

(S5) Anwendung des SuS-Verfahrens mit dem aktuellen SVR-Metamodell zur Ermittlung des (vorläufigen) kritischen Parameterbereichs und seiner Wahrscheinlichkeit

Das SuS-Verfahren liefert eine Stichprobe von Parametervektoren, die entsprechend der Vorhersage des aktuell vorliegenden SVR-Metamodells im (vorläufigen) kritischen Parameterbereich im Raum Z liegt. Darüber hinaus liefert es eine Schätzung der Wahrscheinlichkeit des (vorläufigen) kritischen Bereichs.

Der vorläufige kritische Bereich wird als derjenige Parameterbereich definiert, dessen LSF-Ergebniswerte unter einen bestimmten Grenzwert $b_{\tilde{p}}$> 0 fallen. Solange der aktuelle Trainingsdatenpool nicht mindestens $\tilde{p} \cdot 100$ % (e.g. $\tilde{p}$ = 0.1) Parametervektoren aus dem tatsächlichen kritischen Bereich ($g(z) \leq 0$) enthält, wird der vorläufige kritische Parameterbereich so definiert, dass er zu LSF-Ergebniswerten führt, die das $\tilde{p} \cdot 100$ % - Quantil ($= b_{\tilde{p}}$) des Trainingsdatenpools nicht überschreiten. Wenn der aktuelle Trainingsdatenpool mindestens $\tilde{p} \cdot 100$ % Parametervektoren aus dem tatsächlichen kritischen Bereich enthält, wird der kritische Bereich auf den tatsächlichen kritischen Parameterbereich gesetzt.

Der Wert von $\tilde{p}$ zur Definition des vorläufigen kritischen Bereichs sollte in Abhängigkeit von der Beziehung zwischen den Ergebniswerten und den Eingabeparametern im Trainingsdatenpool ausgewählt werden. Für eine komplexere Beziehung sollte ein höherer Wert von $\tilde{p}$ gewählt werden, beispielsweise aus [0,2, 0,3]. Für Beziehungen, die annähernd monoton sind, wird ein niedrigerer Wert von $\tilde{p}$, beispielsweise aus [0,1; 0,2] empfohlen, um die Rechenkosten gering zu halten.

Eine Übersicht über das SuS-Verfahren und den angewendeten SuS-Algorithmus ist in Abschnitt 2.1.3 zu finden.

(S6) Entscheidung über den nächsten auszuführenden Schritt

Wenn der Trainingsdatenpool mindestens $\tilde{p} \cdot 100$ % Parametervektoren aus dem tatsächlichen kritischen Bereich enthält (Bedingung A in Abb. 2.1 ist erfüllt) und die SVR-Modellvorhersage sowie der Schätzer für die Wahrscheinlichkeit des kritischen Bereichs robust sind (Bedingungen B und C in Abb. 2.1 sind erfüllt; Einzelheiten unter Schritt (S7)), ist Schritt (S8) der nächste auszuführende Schritt des iterativen Verfahrens. Andernfalls ist Schritt (S7) der nächste Schritt.

(S7) Generieren von zusätzlichen Trainingsdaten und Fortfahren mit Schritt (S3)

Wenn der Trainingsdatenpool nicht mindestens $\tilde{p} \cdot 100$ % Parametervektoren aus dem tatsächlichen kritischen Bereich enthält (Bedingung A in Abb. 2.1 ist nicht erfüllt), werden n_{pc} (z. B. n_{pc}= 5) neue Parameterkandidaten zufällig aus derjenigen Stichprobe ausgewählt, die entsprechend der Vorhersage des SuS-Verfahrens mit dem SVR-Metamodell aus dem vorläufigen kritischen Bereich stammt (Ergebnis aus Schritt (S5)). Da der

angewendete SuS-Algorithmus Parametervektoren aus dem Standardnormalraum Z liefert, müssen die ausgewählten Parameterkandidaten in den tatsächlichen Parameterraum X abgebildet werden, bevor sie vom eigentlichen Simulationscode verwendet werden können. Sobald alle Ergebnisse aus den Läufen des Simulationscodes vorliegen, wird mit Schritt (S3) und den folgenden Schritten weitergemacht.

Wenn im Trainingsdatenpool zwar mindestens $\tilde{p} \cdot 100$ % Parametervektoren aus dem tatsächlichen kritischen Bereich sind (Bedingung A in Abb. 2.1 ist erfüllt), aber die SVR-Vorhersage oder die geschätzte Wahrscheinlichkeit des kritischen Bereichs noch nicht robust genug sind (Bedingung B oder C in Abb. 2.1 ist nicht erfüllt), werden n_{pc} neue Parameterkandidaten zufällig aus der Stichprobe S ausgewählt, die potenziell aus dem eigentlichen kritischen Bereich stammt (Ergebnis aus Schritt (S5)).

Die SVR-Vorhersage wird als schwach beurteilt, wenn der mittlere Anteil der Wechselelemente (switching elements) mindestens $p_{swi} \cdot 100$ % des Stichprobenumfangs von S beträgt (z. B. p_{swi} = 0.05). Wechselelemente sind solche Parametervektoren aus der Stichprobe S, die nicht dem kritischen Bereich zugeordnet werden, wenn die n_{svr} (z. B. n_{svr} = 5) vorher konstruierten SVR-Modelle angewendet werden. Der Schätzer für die Wahrscheinlichkeit wird als schwach angenommen, wenn die mittlere Änderungsrate zwischen den Schätzern der n_{svr} letzten SVR-Modelle mehr als einem vorgegebenen λ_P-Wert beträgt (z. B. λ_P = 0.1).

(S8) Umfangreiche Anwendung des SuS-Verfahrens mit dem zuletzt konstruierten robusten SVR-Modell zur endgültigen Abschätzung des kritischen Bereichs und seiner Wahrscheinlichkeit

In diesem Schritt werden mehr als 10000 Parameterkombinationen aus dem kritischen Bereich generiert. Das umfangreiche SuS-Verfahren wird in einer Schleife über verschiedene Anfangswerte (seeds) des Zufallszahlengenerators angewendet. Dadurch erhält man eine Bandbreite von Schätzwerten für die Eintrittswahrscheinlichkeit des kritischen Bereichs und unterschiedliche Sets von SuS-Stichproben. Für die Charakterisierung des kritischen Bereichs wird dasjenige Set ausgewählt, für das der zugehörige Wahrscheinlichkeitsschätzwert den kleinsten Variationskoeffizienten (Verhältnis der Standardabweichung zum Mittelwert, siehe Abschnitt 2.1.3.3 aufweist.

(S9) Beenden des Iterationsprozesses

Das eben beschriebene SuSSVR-Verfahren wurde bereits in /KLO 20/ publiziert. Es basiert auf dem Ansatz, der in /BOU 16/ vorgeschlagen wurde. In /PED 17/ wird ein ähnlicher Ansatz vorgestellt, bei dem im Rahmen des SuS-Verfahrens ein künstliches neuronales Netz anstelle eines SVR-Modells angewendet wird. Eine Kombination aus SuS-Verfahren und Gauß-Prozess /WIL 06/ wird in /HUA 16/ beschrieben.

2.1.2 Konstruktion des Metamodells

Zur Konstruktion des Metamodells wird die Stützvektorregression (SVR) angewendet, die zu den maschinellen Lernalgorithmen zählt. Dabei wird die Komplexität des Metamodells unter der Bedingung, dass seine Vorhersage nicht mehr als einen vordefinierten ε-Wert von dem tatsächlichen Simulationsergebnis abweicht, so gering wie möglich gehalten.

Auf der Grundlage eines gegebenen Trainingsdatenpools $\{(x_i, y_i)\}_{i=1}^{n}$ mit n Eingabeparametervektoren $x_i \in R^m$ und den dazugehörigen Simulationsergebnissen $y_i \in R$ zielt der SVR-Ansatz darauf ab, eine möglichst glatte Funktion f zu finden, die die Vektoren x_i vom Parameterraum auf den Ergebnisraum abbildet, während gleichzeitig die Vorhersagen $f(x_i)$ innerhalb einer ε-Abweichung um die tatsächlichen y_i-Werte liegen. Es ist sowohl eine lineare als auch eine nichtlineare Abbildung möglich.

Im Falle einer linearen Abbildung kann die SVR-Funktion f wie folgt formuliert werden

$$f(x) = \langle w, x \rangle + b \tag{2.4}$$

wobei $w \in R^m$ den Vektor der Regressionskoeffizienten, $b \in R$ einen Bias-Term und $\langle \cdot, \cdot \rangle$ ein Skalarprodukt bezeichnet.

Eine möglichst glatte Funktion f erhält man durch Minimieren der Norm $\|w\|^2 = \langle w, w \rangle$.

Das Problem der SVR-Modellkonstruktion kann also als konvexes Optimierungsproblem formuliert werden /SMO 04/:

$$\begin{aligned} &\text{Minimiere } \frac{1}{2}\|w\|^2 \\ &\text{u. d. N. } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{aligned} \tag{2.5}$$

Da es nicht immer möglich ist, jedes Simulationsergebnis y_i innerhalb der ε-Abweichung vorherzusagen, werden die beiden Größen (slack variables) ξ_i und ξ_i^* eingeführt, um Vorhersagen außerhalb des ε-Bereichs zu tolerieren. Das heißt, das Optimierungsproblem wird neu definiert als:

$$\begin{aligned} &\text{Minimiere } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &\text{u. d. N. } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{2.6}$$

wobei C ein Regularisierungsparameter ist, der zwischen der Glätte der Regressionsfunktion f und dem Vorhersagefehler abwägen soll. Der Vorhersagefehler ist dabei die Summe der Abweichungen zwischen vorhergesagten und tatsächlichen Ergebniswerten, die größer als ε sind /SMO 04/.

Das Optimierungsproblem in Gl. (2.6) wird durch Transformation in ein Lagrange-duales Optimierungsproblem gelöst /SMO 04/. Dabei wird der Vektor w der Regressionskoeffizienten wie folgt berechnet:

$$w = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\, x_i \tag{2.7}$$

wobei α_i und α_i^* Lagrange Multiplikatoren sind. Die Parametervektoren x_i mit Koeffizienten ungleich Null sind die sogenannten Stützvektoren.

Aus Gl. (2.7) folgt für die SVR-Funktion f:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x, x_i \rangle + b \tag{2.8}$$

Eine nichtlineare anstelle einer linearen SVR-Funktion erhält man, indem die Parametervektoren x_i vom Raum X auf einen Raum V höherer Dimension abgebildet werden und dann die lineare SVR-Funktion angewendet wird. Da das Lagrange-duale Problem die Berechnung von Skalarprodukten im Raum V erfordert, kann dieser Ansatz sehr rechenintensiv werden. Dieses Problem wird jedoch gelöst, indem eine Kernelfunktion $k(x_i, x_j)$ eingeführt wird, die das Skalarprodukt in V definiert, d. h. $k(x_i, x_j) := \langle \upsilon(x_i), \upsilon(x_j) \rangle$. Die Abbildungsfunktion υ muss nicht bekannt sein, da die Abbildung von X nach V implizit erfolgt. Somit ist die nicht-lineare SVR-Funktion gegeben durch

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \, k(x, x_i) + b \tag{2.9}$$

wobei $k(x, x_i)$ eine geeignete Kernelfunktion darstellt.

Neben dem linearen Kernel $k_l(x_i, x_j) = \langle x_i, x_j \rangle$ können die folgenden nicht-linearen Kernel-Funktionen angewendet werden:

- Polynomischer Kernel:

$$k_p(x_i, x_j) = \left(\frac{\langle x_i, x_j \rangle}{2\sigma^2} + r \right)^d \tag{2.10}$$

Dabei ist σ die Bandbreite des Kernels, d ist der Grad des Polynoms, und r ist ein unabhängiger additiver Term.

- Sigmoider Kernel:

$$k_s(x_i, x_j) = \tanh\left(\frac{\langle x_i, x_j \rangle}{2\sigma^2} + r\right) \tag{2.11}$$

- Gaußsche radiale Basisfunktion:

$$k_r(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{2.12}$$

Äquivalente Definitionen dieser Kernel verwenden den Parameter γ statt σ:

$$\gamma = \frac{1}{2\sigma^2} \tag{2.13}$$

Für die Konstruktion des SVR-Metamodells wird die Implementierung SVR der frei verfügbaren Python-Bibliothek scikit-learn verwendet.

2.1.2.1 Hyperparameter Tuning

Die Leistung des SVR-Metamodells hängt wesentlich von seinen Hyperparametern ab. Das Tuning dieser Hyperparameter ist daher ein wichtiger Schritt bei der Konstruktion eines SVR-Metamodells. Die für das Tuning in Frage kommenden Hyperparameter sind der SVR-Parameter ε, der den Vorhersagefehler bestimmt, und der Regularisierungsparameter C, der ein Abwägen zwischen der Glätte der Regressionsfunktion f und dem Vorhersagefehler ermöglicht (siehe Gl. (2.6)).

Wenn ε groß ist, ist die Komplexität des Metamodells gering, was bedeutet, dass nur wenige Stützvektoren ausgewählt werden, um das Modell zu beschreiben. Umgekehrt führt ein kleiner Wert von ε zu einem komplexeren Modell mit einer höheren Anzahl von Stützvektoren. Ein niedrigerer Wert für Parameter C macht die Funktion glatter, während ein höherer C-Wert zu einer komplexeren Funktion führt.

Die für die SVR-Hyperparameter angenommenen Größenordnungen beziehen sich auf standardisierte / skalierte Trainingsdaten (Schritte (S2) und (S3) des Iterationsverfahrens in Abschnitt 2.1.1).

Der erste Tuningprozess erfolgt auf der Basis des anfänglichen Pools von standardisierten/skalierten Trainingsdaten. Danach wird der Tuningprozess bei jedem n_{it}-ten Iterationsschritt durchgeführt (z. B. n_{it} = 10). Im ersten Schritt des Tuningprozesses werden 100 mögliche Wertekombinationen der Hyperparameter C und ε zufällig gezogen. Jede ausgewählte Kombination wird durch k-fache Kreuzvalidierung (z. B. k = 5) (/PIC 84/, /STO 74/) unter Verwendung des mittleren quadratischen Fehlers (MSE) als Score bewertet. Die am besten bewertete Kombination wird dann im zweiten Schritt des Tuningprozesses mit der Wertekombination verglichen, die aus dem vorangegangenen Tuningprozess resultierte. Schließlich wird die Kombination verwendet, die bei diesem Vergleich am besten bewertet wird. Wenn am Anfang des Iterationsprozesses, noch keine Hyperparameter aus einem vorangegangenen Tuningprozess vorliegen, wird die am besten bewertete Kombination aus dem ersten Schritt des Tuningprozesses mit der vorgegebenen Wertekombination C= 100 und ε = 0.1 verglichen.

Für die Kreuzvalidierung im ersten Schritt des Tuningprozesses wird die Implementierung RandomizedSearchCV der frei verfügbaren Python-Bibliothek scikit-learn verwendet. Der Vergleich im zweiten Schritt des Tuningprozesses erfolgt mit der Implementierung GridSearchCV von scikit-learn.

Im Folgenden wird ein kurzer Überblick über die Kreuzvalidierung gegeben.

2.1.2.2 Kreuzvalidierung

Für die k-fache Kreuzvalidierung (k -fold crossvalidation) wird der Trainingsdatenpool in k kleinere Teilmengen (folds) aufgeteilt. Dann wird ein SVR-Modell unter Verwendung des Hyperparametersatzes $h, h = 1, \dots, n_h,$ basierend auf den Daten aus k–1 Teilmengen (Trainingsmenge) konstruiert. Das konstruierte SVR-Modell wird an derjenigen Teilmenge (Testmenge) validiert (getestet), die übrig bleibt, nachdem die k–1 Teilmengen für die Modellkonstruktion entfernt wurden. Das Validierungsergebnis ist der mittlere quadratische Fehler MSE_{hf}, der sich aus dem Vergleich der tatsächlichen Ergebniswerte (bereitgestellt durch den Simulationscode) in der Testmenge f mit den jeweiligen SVR-Ergebnissen basierend auf dem ausgewählten Hyperparametersatz h ergibt.

Der Vorgang wird für jede der k möglichen Testmengen und der entsprechenden Trainingsmengen wiederholt. Der Durchschnitt über die k mittleren quadratischen Fehler $MSE_{h1}, \dots, MSE_{hk}$ wird als Bewertung für die Güte des SVR-Modells mit der Hyperparameterkombination $h, h = 1, \dots, n_h,$ verwendet. Die beste Hyperparameterkombination ist die mit dem kleinsten durchschnittlichen mittleren quadratischen Fehler. Ein Sonderfall der k-fachen Kreuzvalidierung ist die Leave-One-Out Kreuzvalidierung, wobei k der Anzahl n der Trainingsdaten entspricht.

2.1.3 Subset-Simulation

Die Subset-Simulation (SuS) ist eine fortschrittliche MCS-Methode zur effizienten Abschätzung einer kleinen Ausfallwahrscheinlichkeit in einem hochdimensionalen Parameterraum /AUS 01/. Die Schlüsselidee besteht darin, die kleine Wahrscheinlichkeit als Produkt signifikant größerer bedingter Wahrscheinlichkeiten abzuschätzen. Diese bedingten Wahrscheinlichkeiten beziehen sich auf verschachtelte Parameterbereiche, wobei der größte dieser Bereiche dem Parameterraum Z entspricht und der kleinste Bereich alle Parametervektoren umfasst, die zum entsprechenden Ausfallereignis führen.

Sei $D = \{z \in R^m : g(z) \leq 0\}$ der kritische Parameterbereich (Parameterregion) im Raum Z, der zu einem Fehler oder einem sonstigen kritischen Ereignis mit geringer Wahrscheinlichkeit führt. Gemäß der Mengenlehre kann D als Schnittmenge geeigneter Teilmengen (subsets) $\{D_i\}_{i=0}^{n_D}$ mit $D_0 \supset D_1 \supset D_2 \supset \cdots \supset D_{n_D}$, $D_0 = Z$ und $D_{n_D} = D$ ausgedrückt werden:

$$D = D_{n_D} = \bigcap_{i=0}^{n_D} D_i \tag{2.14}$$

Die Wahrscheinlichkeit von D kann als Produkt der bedingten Wahrscheinlichkeiten $\{P(D_i|D_{i-1})\}_{i=1}^{n_D}$ berechnet werden:

$$P(D) = P\left(D_{n_D}\right) = P\left(\bigcap_{i=0}^{n_D} D_i\right) = \prod_{i=1}^{n_D} P(D_i|D_{i-1}) \tag{2.15}$$

Vorausgesetzt, dass die bedingten Wahrscheinlichkeiten $\{P(D_i|D_{i-1})\}_{i=1}^{n_D}$ signifikant größer sind als die Wahrscheinlichkeit $P(D)$, können diese aus den Ergebnissen einer relativ geringen Anzahl von MCS-Läufen effizient geschätzt werden. Ein Schätzer für $P(D_1|D_0) = P(D_1|Z) = P(D_1)$ kann aus den Ergebnissen einer klassischen MCS berechnet werden, wobei die Parameterwerte aus unabhängigen Standardnormalverteilungen gezogen werden (siehe Schritt (S2) des iterativen Verfahrens in Abschnitt 2.1.1).

Die Schätzung der Wahrscheinlichkeiten $\{P(D_i|D_{i-1})\}_{i=2}^{n_D}$ erfordert Parametervektoren, die aus den bedingten Verteilungen $\{\phi(z|D_{i-1})\}_{i=2}^{n_D}$ ausgespielt werden mit

$$\phi(z|D_{i-1}) = \frac{\phi(z) I_{D_{i-1}}(z)}{P(D_{i-1})}, i = 2, \dots, n_D \tag{2.16}$$

wobei $\phi(\cdot)$ die multivariate (bedingte) Dichte von m unabhängigen Standardnormalverteilungen ist und $I_{D_{i-1}}(z)$ die Indikatorfunktion von $D_{i-1}, i = 2, \dots, n_D$. Das Ausspielen aus der bedingten Verteilung $\phi(z|D_i)$, $i = 1, \dots, n_D - 1$, erfolgt durch eine Markov-Chain-Monte-Carlo-Simulation (siehe Abschnitt 2.1.3.1).

Das SuS-Verfahren kann wie folgt skizziert werden: Sei $S_{i-1} = \{z_{i-1,j}\}_{j=1}^{nsus}$ eine Stichprobe von $nsus$ Parametervektoren aus der Teilmenge D_{i-1}, $i > 1$. Unter der Annahme, dass der Schätzer für die bedingte Wahrscheinlichkeit $P(D_i|D_{i-1})$ gleich p_0 ist, ist die Anzahl der Parametervektoren aus der Stichprobe S_{i-1}, die in die Teilmenge D_i fallen, $n_{p_0} = nsus \cdot p_0$. Dabei ist $D_i = \{z \in R^m : g(z) \leq b_i\}$, wobei b_i das p_0-Quantil der Stichprobe der LSF-Werte $\{g(z_{i-1,j})\}_{j=1}^{nsus}$ ist, die auf der Parameterstichprobe S_{i-1} basiert. Die n_{p_0} Parametervektoren aus der Stichprobe S_{i-1}, die in die Teilmenge D_i fallen, werden als Startwerte (seeds) für die Auswahl von $nsus = n_{p_0} \cdot n_s$ neuen Elementen aus der Teilmenge D_i gemäß der Verteilung $\phi(z|D_i)$ verwendet, wobei n_s die Anzahl der pro Startwert erzeugten Stichprobenelemente ist. Wenn die Stichprobe S_i aus der Teilmenge D_i verfügbar ist, fallen n_{p_0} Elemente dieser Stichprobe in die Teilmenge $D_{i+1} = \{z \in R^m : g(z) \leq b_{i+1}\}$, wobei b_{i+1} das p_0-Quantil der entsprechenden Stichprobe von LSF-Werten ist. Die Prozedur wird fortgesetzt, bis das p_0-Quantil negativ ist, was impliziert, dass der kritische Parameterbereich $D = D_{n_D} = \{z \in R^m : g(z) \leq b_{n_D}\}$ mit $b_{n_D} = 0$ erreicht wurde.

Die Wahrscheinlichkeit $P(D)$ wird wie folgt geschätzt:

$$\hat{P}(D) = p_0^{n_D-1} \cdot \hat{P}\left(D_{n_D} \middle| D_{n_D-1}\right) \tag{2.17}$$

wobei $\hat{P}\left(D_{n_D} \middle| D_{n_D-1}\right) = \frac{\#\{z \in S_{n_D-1} : g(z) \leq 0\}}{nsus}$. # bezeichnet die Kardinalität einer Menge.

2.1.3.1 Markov-Chain-Monte-Carlo-Simulation

Die Markov-Chain-Monte-Carlo-Simulation (MCMC-Simulation) liefert Zustände einer Markov-Kette, die die gewünschte Verteilung, aus der ausgespielt werden soll, als stationäre Verteilung hat. Die MCMC-Methode des SuS-Verfahrens basiert auf dem Metropolis-Hastings-Algorithmus (MH-Algorithmus) /AUS 01/. Sie erzeugt einen neuen Vektorzustand z_1 basierend auf einem aktuellen Zustand z_0 aus der Teilmenge D_i gemäß den folgenden Schritten:

(i) Erzeugen von Kandidat v

- durch Ausspielen eines vorläufigen Kandidaten $\tilde{v}$ aus der Verteilung $\mathrm{q}(\cdot\,|z_0)$, die bedingt ist durch den aktuellen Zustand z_0,
- durch zufällige Auswahl des Zustands v

$$v = \begin{cases} \tilde{v} \text{ mit Wahrsch. } a_0(z_0, \tilde{v}) \\ z_0 \text{ mit Wahrsch. } 1 - a_0(z_0, \tilde{v}) \end{cases}$$

wobei $a_0(z_0, \tilde{v}) = \min\left(1, \frac{\phi(\tilde{v}) \cdot q(z_0|\tilde{v})}{\phi(z_0) \cdot q(\tilde{v}|z_0)}\right)$

(ii) Festlegen des neuen Vektors z_1

$$z_1 = \begin{cases} v \text{ , wenn } v \in D_i \\ z_0, \text{wenn } v \notin D_i \end{cases}$$

Wenn für die Verteilung $q(\cdot\,|z_0)$ aus Schritt (i) gilt, dass $q(z_0|\,\tilde{v}) = q(\tilde{v}|z_0)$ ist die Wahrscheinlichkeit, den vorläufigen Kandidaten $\tilde{v}$ zu akzeptieren, gegeben durch $a_0(z_0, \tilde{v}) = \min\left(1, \phi(\tilde{v}) \middle/ \phi(z_0)\right)$.

Da die Akzeptanzwahrscheinlichkeit $a_0(z_0, \tilde{v})$ mit zunehmender Anzahl von Parametern schnell kleiner wird, ist der MH-Algorithmus in einem hochdimensionalen Parameterraum ineffizient. Eine niedrige Akzeptanzwahrscheinlichkeit bedeutet, dass der MH-Algorithmus viele identische Vektorzustände erzeugt, sodass die jeweiligen

Korrelationen zwischen den Elementen der MCMC-Stichprobe und mit ihnen die jeweiligen Varianzen der Schätzer für die bedingten Wahrscheinlichkeiten größer werden. Folglich ist der Schätzer für die Wahrscheinlichkeit $\hat{P}(D)$ in einem hochdimensionalen Parameterraum mit einem großen Bias behaftet. Aus diesem Grund wird in /AUS 01/ vorgeschlagen, den sogenannten komponentenweisen (parameterbezogenen) MH-Algorithmus anzuwenden.

Anstatt $v = (v_1, v_2, \dots, v_m)$ aus einer multivariaten Verteilung der Dimension m auszuspielen, werden die jeweiligen Kandidaten v_j für die Parameter $j,\ j = 1, \dots, m$ aus entsprechenden univariaten Verteilungen ausgespielt. Jede dieser univariaten Verteilungen hängt vom aktuellen Wert des jeweiligen Parameters ab. /AUS 01/ haben gezeigt, dass wenn der aktuelle Vektor z_0 gemäß der multivariaten Verteilung $\phi(\cdot\,|D_i)$ verteilt ist, auch der nächste durch den komponentenweisen MH-Algorithmus erzeugte Vektor z_1 aus dieser Verteilung ist. Daher ist $\phi(\cdot\,|D_i)$ die stationäre multivariate Verteilung der Markov-Kette.

In /AUS 01/ wird empfohlen, die in z_{0j} zentrierte Gleichverteilung mit einer Breite von 2 als univariate Verteilung $\mathrm{q}_j\,(\cdot\,|\,\cdot)$ zu verwenden für $j = 1, \dots, m$. Nach /AUS 10/ sollte zusätzlich der relative Einfluss jedes Parameters berücksichtigt werden, indem für die Varianz der jeweiligen Verteilung $\mathrm{q}_j\,(\cdot\,|\,\cdot)$ die Varianz derjenigen Parameter aus Stichprobe S_{i-1} genommen wird, die in D_i liegen für $i = 1, \dots, n_D$.

In /PAP 15/ werden verschiedene MCMC-Algorithmen diskutiert. Zusätzlich wird ein neuer adaptiver MCMC-Algorithmus vorgeschlagen, der immer einen Kandidaten v liefert, der sich vom aktuellen Zustand z_0 aus der Teilmenge D_i unterscheidet. v wird als neuer Wert akzeptiert, wenn er zu D_i gehört. Die Effizienz des Algorithmus wird verbessert, indem die Varianz der Verteilung in Abhängigkeit von der Akzeptanzwahrscheinlichkeit kontinuierlich angepasst wird.

2.1.3.2 Metropolis-Hastings-Algorithmus

Für das SuS-Verfahren im Rahmen der adaptiven MCS wird ein komponentenweiser (parameterbezogener) MH-Algorithmus angewendet. Als Verteilung wird eine univariate Gleichverteilung verwendet, die um den jeweiligen Startparameterwert z_{0j} zentriert ist. Die Varianz der Gleichverteilung wird basierend auf dem von /PAP 15/ vorgeschlagenen Ansatz in Abhängigkeit von der Akzeptanzwahrscheinlichkeit kontinuierlich angepasst.

Begonnen wird mit der Varianz der Startwerte (seeds) in der Teilmenge D_1 aus der Stichprobe S_0. Diese Varianz ist gleichzeitig die maximale Varianz, die angenommen werden kann.

Da die durch die MCMC-Simulation erzeugten Stichprobenelemente nicht unabhängig sind, sind die Schätzer für die bedingten Wahrscheinlichkeiten und damit auch der endgültige Schätzer $\hat{P}(D)$ mit einem Bias behaftet. /AUS 01/ haben jedoch gezeigt, dass der Schätzer asymptotisch unverzerrt ist. Darüber hinaus zeigten sie, dass $\hat{P}(D)$ ein konsistenter Schätzer ist, d. h. der Variationskoeffizient konvergiert mit zunehmender Stichprobengröße $nsus$ gegen Null.

2.1.3.3 Variationskoeffizient

Wenn eine klassische MCS vom Umfang n durchgeführt wird, ist der Variationskoeffizient δ des Schätzers $\hat{P}$ einer Wahrscheinlichkeit gegeben durch

$$\delta = \sqrt{\frac{1-\hat{P}}{n\hat{P}}} \tag{2.18}$$

Wenn das SuS-Verfahren angewendet wird, lässt sich der Variationskoeffizient wie folgt herleiten /AUS 01/:

Die Variationskoeffizienten δ_i der Schätzer $\hat{P}_i$ für die bedingten Verteilungen $P(D_i|D_{i-1})$, $i = 1, \dots, n_D$, lassen sich abschätzen durch:

$$\widehat{\delta_i} = \sqrt{\frac{1-\hat{P}_i}{(n\hat{P}_i)}(1-\gamma_i)} \tag{2.19}$$

mit

$$\gamma_i = 2\sum_{k=1}^{n_s-1}\left(1-\frac{k}{n_s}\right)\rho_i(k) \tag{2.20}$$

Dabei ist $\rho_i(k)$ der sogenannte Autokorrelationskoeffizient bzgl. des Abstands (lag) k, $k = 1, \ldots, n_s - 1$, zwischen den Indikatorfunktionen $\{I_{D_i}(z_{jl}), j = 1, \ldots, n_{p_0},\ l = 1, \ldots, n_s\}$, d.h. $\rho_i(k)$ ist der Korrelationskoeffizient zwischen den Indikatorfunktionen $I_{D_i}(z_{jl})$ und $I_{D_i}(z_{jl+k})$ für alle $j = 1, \ldots, n_{p_0}$ und alle $l = 1, \ldots, n_s$.

$\widehat{\delta_i}$ kann aus den SuS-Stichproben berechnet werden. Gl. (2.19) setzt voraus, dass die unterschiedlichen Markov-Ketten unkorreliert sind durch die Indikatorfunktionen.

Der Variationskoeffizient δ_{SuS} des Schätzers $\hat{P}(D)$ lässt sich wie folgt schätzen

$$\hat{\delta}_{SuS} = \sqrt{\sum_{i,j=1}^{n_D} \widehat{\delta_i}\widehat{\delta_j}\rho_{ij}} \tag{2.21}$$

Dabei ist ρ_{ij} der Korrelationskoeffizient zwischen den Schätzern $\hat{P}_i$ und $\hat{P}_j, i, j = 1, \ldots, n_D$. Wenn $\hat{P}_i$ und $\hat{P}_j$ unkorreliert sind, ist $\hat{\delta}_{SuS}$ gegeben durch

$$\hat{\delta}_{SuS} = \sqrt{\sum_{i=1}^{n_D} \hat{\delta}_i^2} \tag{2.22}$$

2.1.4 Anwendungsbeispiel: Biologisches Dosismodell

Das hier beschriebene Anwendungsbeispiel bezieht sich auf eine Verordnung der US-Umweltschutzbehörde zur Freisetzung von Radioaktivität bei normalem Betrieb eines KKW /EPA 77/. Die Verordnung setzt den Grenzwert für die maximale Äquivalentdosis auf 0.25 Millisievert (mSv) pro Person und Jahr fest. Im Beispiel wird angenommen, dass während des normalen Betriebs eines KKWs kleine Konzentrationen von Radionukliden freigesetzt werden und in die Nahrungskette einer Bevölkerungsgruppe gelangen. Um das maximale jährliche Dosisäquivalent eines Individuums der Bevölkerungsgruppe zu berechnen, wird das folgende einfache deterministische Modell angewendet:

$$y = c \cdot (x\_rate_1 \cdot x\_conc_1 + x\_rate_2 \cdot x\_conc_2) \cdot e^{-0.2 \cdot dt} \tag{2.23}$$

Die Erläuterung der einzelnen Parameter des Dosismodells befindet sich in Tab. 2.1. Die Verteilungen dieser Parameter sind in Tab. 2.2 beschrieben.

Tab. 2.1 Eingabeparameter für das in Gl. (2.23) definierte Dosismodell

Parameter	Bedeutung	Einheit	Nominalwert
c	Dosiskonversionsfaktor	Sv/Bq	3E-08
x_rate_1	Aufnahmerate von Nahrungsmittel 1	kg/a	60
x_conc_1	Radioisotopenkonzentration in Nahrungsmittel 1	Bq/kg	25
x_rate_2	Aufnahmerate von Nahrungsmittel 2	kg/a	120
x_conc_2	Radioisotopenkonzentration in Nahrungsmittel 2	Bq/kg	20
dt	Verzögerungszeit	d	8

Tab. 2.2 Verteilungen der Eingabeparameter für das in Gl. (2.23) definierte Dosismodell

Parameter	Verteilung	Verteilungsparameter	Minimum; Maximum
c	normal	3.29E-08, 1.11E-08	1.E-08; 5.E-08
x_rate_1	log. uniform	10,100	10; 100
x_conc_1	uniform	10,35	10; 35
x_rate_2	log. normal	0.1993, 4.7552	0.5; 400
x_conc_2	uniform	10,30	10; 30
dt	triangular	8	0.5; 20

Drei verschiedene Analysen wurden durchgeführt, um eine probabilistische Bewertung für die Strahlenexposition eines Individuums der betroffenen Bevölkerungsgruppe zu erhalten und die Effizienz des adaptiven SuSSVR-Verfahrens zu bewerten.

In der ersten Analyse wurde der BEPU-Ansatz basierend auf einer einfachen MCS unter Verwendung des Rechenmodells in Gl. (2.23) durchgeführt. Auf Basis der Ergebnisse aus n = 1000 Rechenläufen wurde die obere einseitige (95 %; 95 %) Toleranzgrenze /WIL 41/ für die maximale jährliche Strahlungsäquivalentdosis mit 6.69E-05 Sv (= 0,0669 mSv) berechnet. Dies impliziert, dass die Wahrscheinlichkeit, dass y die Grenze von 2.5E-04 Sv (= 0.25 mSv) überschreitet, höchstens 0.05 beträgt. Da keiner der 1000 Läufe einen y-Wert > 2.5E-04 Sv lieferte, beträgt die obere 95 % Konfidenzgrenze nach Pearson und Clopper /CLO 34/ 2.99E-03. Bei n = 1.0E+06 MCS-Läufen lieferte genau ein Lauf einen y-Wert von mehr als 2.5E-04 Sv. Die Überschreitungs-

wahrscheinlichkeit sollte also ~1.0E-06 betragen. Das 95 % Konfidenzintervall nach Pearson und Clopper beträgt in diesem Fall [2.53E-8; 5.57E-06].

In der zweiten Analyse wurde das reine SuS-Verfahren (Abschnitt 2.1.3) in Kombination mit dem Rechenmodell in Gl. (2.23) angewendet. Die Größe einer SuS-Stichprobe S_i, die in jeder Stufe $i,\ i = 1, \dots, n_D$ erzeugt wurde, betrug $nsus$= 3000; und die Wahrscheinlichkeit p_0 betrug 0.1(Gl. (2.17)). Mit diesen Spezifikationen berechnete das SuS-Verfahren eine Wahrscheinlichkeit von 3.27E-06, dass y ≥ 2.5E-04 Sv. Insgesamt wurden $n = n_D \cdot nsus = 6 \cdot 3000 = 18000$ Läufe mit dem Rechenmodell in Gl. (2.23) durchgeführt.

In der dritten Analyse wurde das SuSSVR-Verfahren angewendet. Die Parametereinstellungen für die Anwendung und die Ergebnisse werden in den folgenden beiden Abschnitten beschrieben.

2.1.4.1 Parametereinstellungen

Die Parameter des SuSSVR-Verfahrens werden unter den einzelnen Schritten des Verfahrens aufgeführt (vgl. Abschnitt 2.1.1):

(S1) – (S2)

Der anfängliche Pool von Trainingsdaten bestand aus n = 50 zufällig ausgewählten Parametervektoren $x \in R^m$, m = 6, und den zugehörigen y-Werten. Jeder Parametervektor x wurde in einen Vektor z von unabhängigen standardnormalverteilten Parametern transformiert.

(S3)

Jeder Wert $y = r(z)$ wurde in den LSF-Wert $g(z)$ transformiert, wobei $g(z) = r_{th} - r(z)$ und r_{th}= 2.5E-04 Sv. Zusätzlich wurde auf $(g(z) - \overline{y_0})/s_{y_0}$ skaliert, wobei $\overline{y_0}$ der Mittelwert und s_{y_0} die Standardabweichung der $g(z)$-Werte im anfänglichen Trainingsdatenpool vom Umfang n = 50 sind.

(S4)

Für das SVR-Metamodell wurde eine nicht-lineare Funktion des Parametervektors x angenommen und die Gaußsche radiale Basisfunktion als Kernelfunktion verwendet. Die Hyperparameter C und ε des SVR-Modells wurden im Laufe der Iteration immer wieder

neu eingestellt (getunt). Der erste Tuningprozess erfolgte auf der Basis des anfänglichen Pools von standardisierten/skalierten Trainingsdaten vom Umfang n = 50. Danach wurde der Tuningprozess bei jedem 10. Iterationsschritt durchgeführt.

(S5)

Beim SuS-Verfahren wurden Stichproben von $nsus$ = 5000 Parametervektoren aus den Teilmengen des Parameterraums generiert. Die Wahrscheinlichkeit zur Definition der Teilmengen $\{D_i\}_{i=1}^{n_D-1}$ betrug p_0= 0.25 (Gl. (2.15) und Gl. (2.17)). Als Zufallszahlengenerator wurde der Zufallszahlengenerator PCG64 der frei verfügbaren Python-Programmbibliothek NumPy verwendet.

Solange im aktuellen Trainingsdatenpool nicht mindestens 10 % Parametervektoren aus dem eigentlichen kritischen Bereich enthalten waren, wurden vorläufige kritische Parameterbereiche als angestrebte Zielbereiche für das SuS-Verfahren festgelegt. Diese waren so definiert, dass ihre Parametervektoren zu LSF-Ergebniswerten führen, die das 10 %-Quantil des jeweils aktuellen Trainingsdatenpools nicht überschreiten. Erst als der aktuelle Trainingsdatenpool mindestens 10 % Parametervektoren aus dem eigentlichen kritischen Bereich enthielt, wurde als angestrebter Zielbereich für das SuS-Verfahren auch der eigentliche kritische Parameterbereich verwendet.

(S6)

Sobald mindestens 10 % Parametervektoren aus dem eigentlichen kritischen Bereich im Trainingsdatenpool vorhanden waren und die SVR-Modellvorhersage sowie der Schätzer für die Wahrscheinlichkeit des kritischen Parameterbereichs als robust beurteilt wurden, wurde abschließend Schritt (S8) ausgeführt. Andernfalls wurde Schritt (S7) ausgeführt.

(S7)

Solange weniger als 10 % Parametervektoren aus dem eigentlichen kritischen Bereich im Trainingsdatenpool vorhanden waren, wurden immer fünf neue Parameterkandidaten zufällig aus der Stichprobe aus dem vorläufigen kritischen Bereich (Ergebnis aus Schritt (5)) ausgewählt und als Eingabe für das Rechenmodell in Gl. (2.23) verwendet. Da der angewendete SuS-Algorithmus immer standardnormal-verteilte Parametervektoren liefert, mussten die ausgewählten Parametervektoren in den eigentlichen Parameterraum X abgebildet werden, bevor sie für die Berechnungen verwendet werden konnten. Sobald alle Rechenergebnisse vorlagen, wurden das Verfahren ab Schritt (S3) wiederholt.

Wenn im Trainingsdatenpool zwar mindestens 10 % Parametervektoren aus dem eigentlichen kritischen Bereich waren, aber die SVR-Vorhersage oder die geschätzte Wahrscheinlichkeit des kritischen Bereichs noch nicht robust genug waren, wurden immer 5 neue Parameterkandidaten zufällig aus der Stichprobe S aus dem eigentlichen kritischen Bereich (Ergebnis aus Schritt (5)) ausgewählt.

Die SVR-Vorhersage wurde als schwach beurteilt, wenn der mittlere Anteil der Wechselelemente (switching elements) mehr als 2.5 % (d.h. p_{swi}= 0.025) des Stichprobenumfangs von S betrug. Als Wechselelemente wurden diejenigen Parametervektoren aus der Stichprobe S betrachtet, die nicht dem kritischen Bereich zugeordnet wurden, wenn die 5 vorher konstruierten SVR-Modelle angewendet wurden. Der Schätzer für die Wahrscheinlichkeit des kritischen Bereichs wurde als schwach beurteilt, wenn die mittlere Änderungsrate zwischen den Schätzern der letzten beiden SVR-Modelle mehr als λ_P = 0.10 betrug.

Das Abbruchkriterium des iterativen Verfahrens lautet also: $p_{swi} \leq 0.025 \,\&\, \lambda_P \leq 0.10$

(S8)

Im Rahmen des umfangreichen SuS-Verfahrens wurden Stichproben von $nsus$ = 20000 Parametervektoren aus den Teilmengen des Parameterraums generiert. Die Werte für die Wahrscheinlichkeit p_0 zur Definition der Teilmengen $\{D_i\}_{i=1}^{n_D-1}$ und den Zufallszahlengenerator wurden wie in Schritte (S5) gewählt. Das SuS-Verfahren in Kombination mit dem robusten SVR-Metamodell wurde insgesamt 10 Mal angewendet; jedes Mal mit einem anderen Anfangswert (seed) für den Zufallszahlengenerator.

2.1.4.2 Ergebnisse

Die Anwendung des SuSSVR-Verfahrens zur Ermittlung des kritischen Parameterbereichs mit einer Äquivalentdosis über 2.5E-04 Sv ergab Schätzwerte für die Wahrscheinlichkeit dieses Bereichs zwischen 3.16E-06 und 4.19E-06. Die Variationskoeffizienten lagen bei 0.037. Die Anzahl der mit dem tatsächlichen Code durchgeführten Läufe betrug n = 215 und ist signifikant kleiner als diejenigen, die für die einfache MCS oder das SuS-Verfahren benötigt wurden.

In Schritt (S8) des SuSSVR-Verfahrens wurden 10 verschiedene Sets von SuS-Stichproben generiert. Für die nachfolgenden Betrachtungen wurden dasjenige Set

ausgewählt, für das der zugehörige Schätzwert für die Eintrittswahrscheinlichkeit des kritischen Bereichs den kleinsten Variationskoeffizienten aufwies. Das ausgewählte Set enthielt SuS-Stichproben aus 9 Parameterbereichen (Regionen 0 – 8). Region 0 entspricht dem gesamten Parameterraum, die Regionen 1 – 8 sind ineinandergeschachtelte Teilbereiche des Parameterraums, wobei Region 8 als kleinster Teilbereich dem kritischen Parameterbereich entspricht. Die Stichproben aus den einzelnen Regionen umfassten jeweils 10725 – 20000 unterschiedliche Kombinationen von Parameterwerten. Für den kritischen Parameterbereich (Region 8) wurden 12856 unterschiedliche Parametervektoren generiert.

Abb. 2.2 zeigt für die einzelnen Parameter aus Tab. 2.2 die empirischen Verteilungen in Form der Kerndichteschätzer (kernel densities) in den verschiedenen Regionen. Jede Spalte in der Abbildung ist einem Parameter zugeordnet; die Zeilen beziehen sich auf die Regionen 0 – 8. Große Unterschiede zwischen den Verteilungen eines Parameters in den einzelnen Regionen weisen auf einen deutlichen Einfluss dieses Parameters hin. Die Gegenüberstellung der Verteilungen eines Parameters bzgl. der Regionen 0 (gesamter Parameterraum) und 8 (kritischer Parameterbereich) zeigt die Bedeutung dieses Parameters für die Festlegung des kritischen Bereichs. Aus Abb. 2.2 wird deutlich, dass bis auf Parameter $xrate_2$ alle Parameter für die Festlegung des kritischen Bereichs wichtig sind.

Abb. 2.3 zeigt die Assoziationen zwischen den Parametern in Form von Scatterplots sowie deren Häufigkeitsverteilungen (Histogramme) im kritischen Bereich. Die Assoziationen geben Auskunft darüber, welche Werte eines Parameters zusammen mit welchen Werten eines anderen Parameters im kritischen Bereich vorkommen. Für den Parameter dt zum Beispiel fällt auf, dass hauptsächlich Werte < 3. in der kritischen Region liegen. Wenn höhere Werte (zwischen 3. und 4.) dieses Parameters in der kritischen Region liegen, so kommen diese vor zusammen mit den höheren Werten der Parameter c und x_conc_2 sowie Werten > 200 des Parameters x_rate_2. Bei Parameter c liegen mit großer Wahrscheinlichkeit Werte > 3.5E-08 in der kritischen Region. Werte < 3.5E-08 kommen vor zusammen mit niedrigen Werten des Parameter dt und hohen Werten des Parameters x_conc_2.

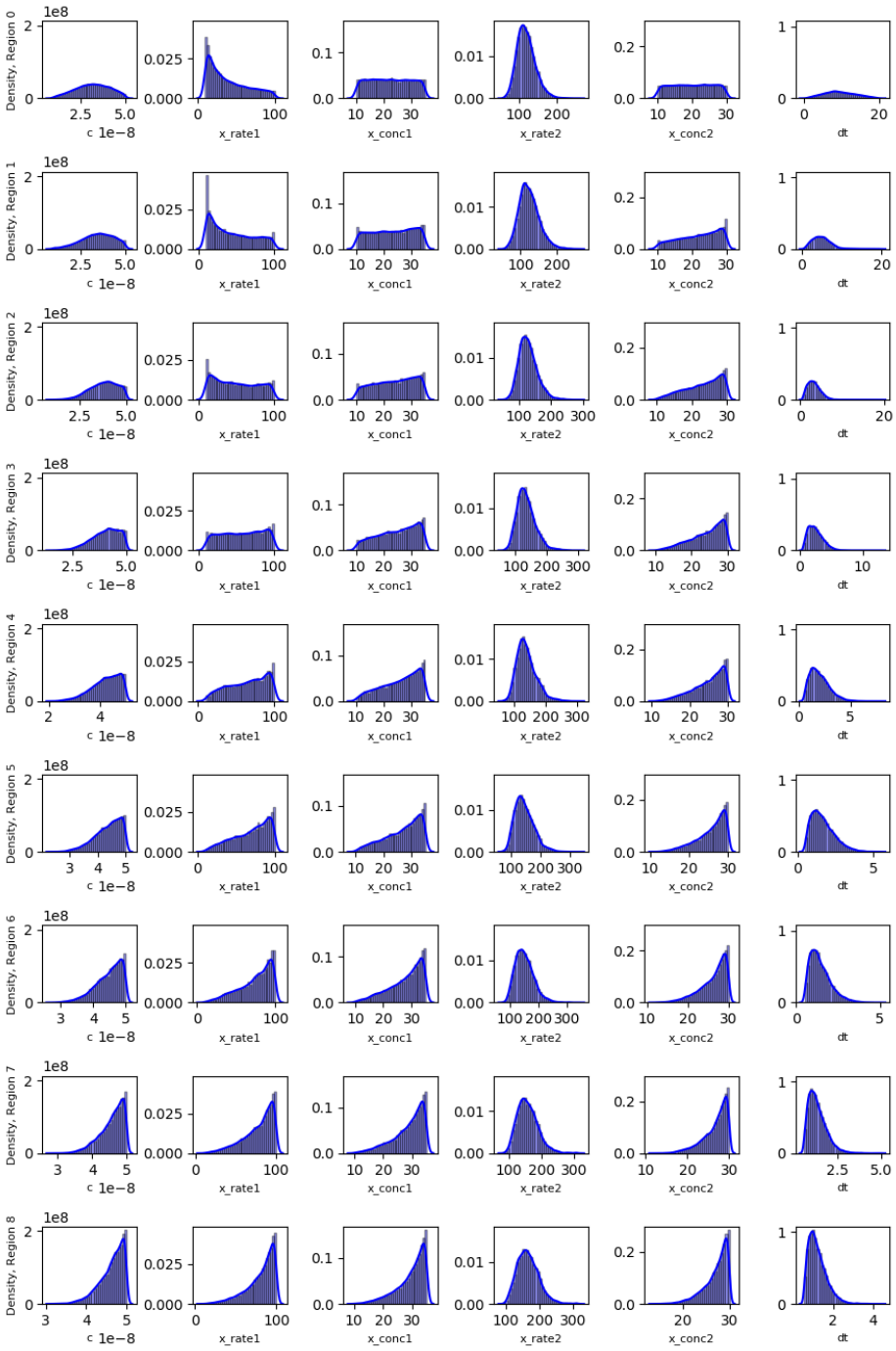

Abb. 2.2 Verteilungen der Parameter aus Tab. 2.2 in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 8 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 8 = kleinste Region = kritischer Parameterbereich

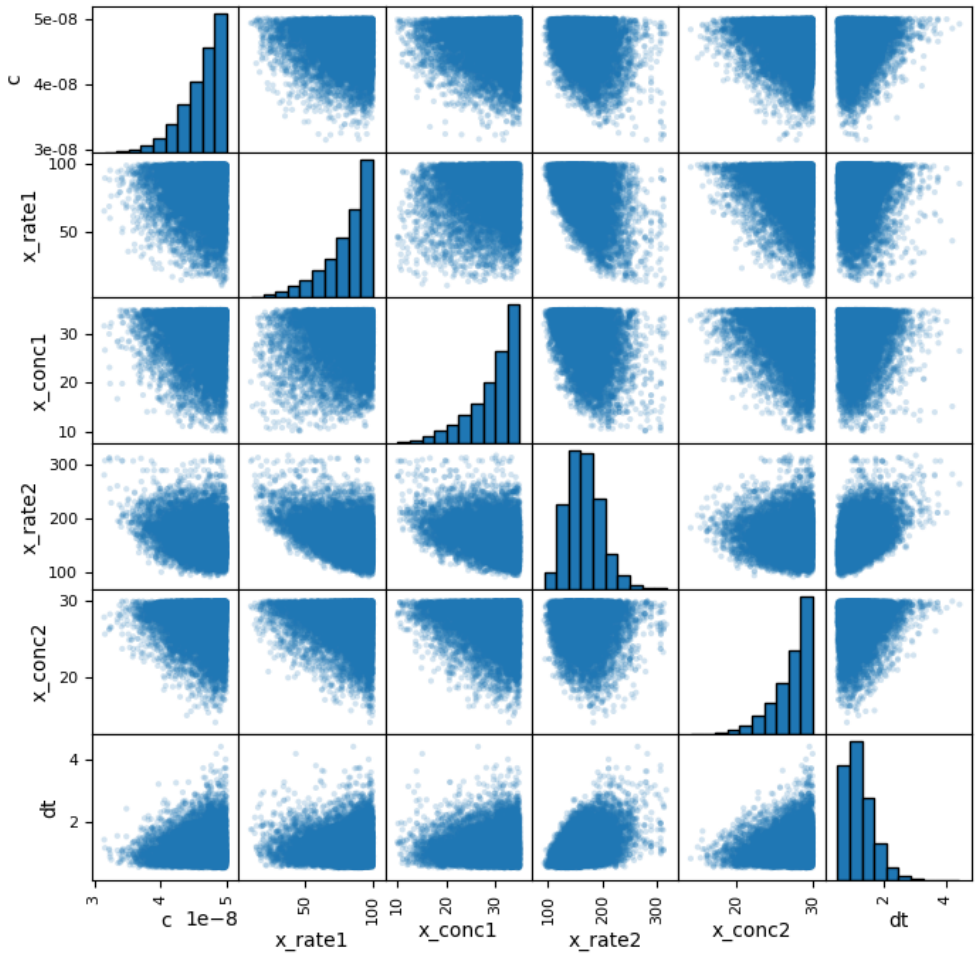

Abb. 2.3 Scatter Plots und Verteilungen in der kritischen Region für alle Parameter in Gl. (2.23)

2.2 Genetischer Algorithmus und Ensemble von Klassifikatormodellen als Metamodell

Bei der Methode, die in diesem Abschnitt beschrieben wird, kommen sowohl ein genetischer Algorithmus als auch verschiedene Verfahren des maschinellen Lernens (ML) zum Einsatz. Die ML-Verfahren werden zur Konstruktion von Ersatzmodellen (Metamodellen) unter Verwendung möglichst weniger Trainingsdaten eingesetzt. Mit einer kleineren Auswahl der Ersatzmodelle wird die Wahrscheinlichkeit für einen definierten kritischen Bereich einer komplexen Funktion geschätzt.

In Abschnitt 2.2.1 wird auf die Zielsetzung und die grundlegende Struktur der entwickelten Methode eingegangen.

2.2.1 Zielsetzung und Struktur der entwickelten Methode

Ziel der in diesem Abschnitt beschriebenen Methode ist es, anhand möglichst weniger Trainingsdaten (Eingabeparametervektoren und entsprechende Rechenergebnisse) ein Ersatzmodell (Metamodell) zu erstellen, mit dem die Ergebnisse einer komplexen Funktion möglichst gut prognostiziert werden können und mit dem relativ wenig Rechenzeit benötigt wird, um eine Wahrscheinlichkeitsabschätzung des kritischen Parameterbereiches der zugrundeliegenden komplexen Funktion durchzuführen.

Formal ausgedrückt kann die Problemstellung wie folgt beschrieben werden: Gegeben seien m Variablen (unsichere Parameter, Merkmale, Einflussfaktoren) $X_1, \ldots, X_m$ einer komplexen Funktion $G(X_1, \ldots, X_m)$. Die Variablen werden durch ihre jeweiligen Verteilungen $F_{X1}, \ldots, F_{Xm}$ beschrieben. Ein zufällig ausgespielter Wert x_i aus F_{Xi} wird als eine Realisation der Variablen X_i, i = 1, …, m, bezeichnet. Zu jeder Realisation des Parametervektors $\mathbf{x} = (x_1, \ldots, x_m)$ wird durch die Funktion G ein zugehöriger Funktionswert $y = G(\mathbf{x})$ ermittelt. Das Ziel der entwickelten Methode ist es, anhand einer Stichprobe von Beobachtungen $y_j = G(\mathbf{x}_j)$, j = 1, …,n, die zugrundeliegende funktionale Beziehung G zwischen $(X_1, \ldots, X_m)$ und dem zugehörigen Ergebnis Y unter Verwendung eines einfacheren Metamodells näherungsweise abzubilden. Das Metamodell M soll die Funktionswerte $G(\mathbf{x})$ möglichst genau prognostizieren, d. h. $M(\mathbf{x}) = G(\mathbf{x}) \pm \varepsilon$, wobei der Prognosefehler ε möglichst klein ist. Angenommen, der kritische Ergebnisbereich D_y der Funktion $G(X_1, \ldots, X_m)$ ist durch Ergebniswerte y definiert, die kleiner oder gleich einem gewissen Schwellenwert y_{crit} sind, d.h. $D_y = \{y: y=G(\mathbf{x}) \leq y_{crit}\}$, dann soll mit dem Metamodell zusätzlich eine gute Wahrscheinlichkeitsabschätzung des kritischen Bereiches D_y möglich sein.

Die Methode soll folgende Kriterien zu erfüllen:

- Sie soll eine möglichst allgemeine Anwendbarkeit auf Funktionen G unterschiedlicher Komplexität gewährleisten. Sie soll insbesondere auch für multivariate und nicht-lineare Funktionen anwendbar sein.

- Sie soll einfach anzuwenden sein. D. h., die angewendeten Metamodelle sollen möglichst keine Hyperparameter enthalten, die in Abhängigkeit von der Funktion G eingestellt werden müssen, z. B. damit das Metamodell möglichst gut an die Daten aus der Anwendung von G angepasst werden kann. Damit unterscheidet sich dieser Ansatz vom Kriging-Verfahren (Gauss-Prozess), bei denen solche Parametereinstel-

lungen normalerweise durchgeführt werden, um eine gute Anpassung an die zugrundeliegende Funktion zu gewährleisten.

- Sie soll effizient sein. D. h., der Aufwand zur Erstellung eines Trainingsdatensatzes, der eine gute Anpassung des Metamodells erlaubt, soll möglichst gering sein.

Die Methode verwendet mehrere unterschiedliche maschinelle Lernalgorithmen, um Metamodelle zu konstruieren. Grund dafür sind die Ergebnisse aus eigenen Untersuchungen, die gezeigt haben, dass es keinen Lernalgorithmus gibt, der in allen Situationen als der beste betrachtet werden kann. D. h. für verschiedene Funktionen G können sich unterschiedliche Lernalgorithmen als vorteilhaft erweisen. Um die Methode für ein möglichst großes Spektrum unterschiedlicher Funktionen G anwenden zu können, wurden die folgenden Lernalgorithmen eingebunden:

- Weighted Nearest Neighbor (wNN) Ansatz
- Entscheidungs-Baum (Decision Tree)
- Naiver Gaus'scher Bayes Klassifikator (Gaussian Naive Bayes)
- Nicht-Parametrischer Naiver-Bayes Klassifikator (Non-Parametric Naive Bayes)
- Random-Forest Klassifikator
- Random-Forest Regressor

Die Algorithmen werden zum Teil zur Auswahl von neuen Kandidaten für den Trainingsdatensatz und zum Teil zur Schätzung der Wahrscheinlichkeit des kritischen Bereichs herangezogen. Zusätzlich wird noch ein Clusteranalyseverfahren eingesetzt, um eine möglichst günstige Auswahl von Kandidaten für den Trainingsdatensatz zu erhalten.

Der nicht-parametrische Naive-Bayes und der Weighted Nearest Neighbor Ansatz wurden im Rahmen des Vorhabens neu entwickelt. Alle anderen der oben aufgeführten Verfahren basieren auf bereits vorhandenen Algorithmen der frei verfügbaren Python Software-Bibliothek scikit-learn. Der nicht-parametrische Naive-Bayes Klassifikator wurde entwickelt, um die Abhängigkeit von der Normalverteilungsannahme der Eingabeparameter zu vermeiden. Der Weighted Nearest Neighbor Ansatz stellt eine gewichtete Modifikation des klassischen K-Nearest Neighbor Ansatzes dar. Die Ansätze sind im Einzelnen im SUSA-Methodenband /KLO 21b/ beschrieben.

Die Methode zur Anpassung eines Metamodells besteht aus zwei Phasen:

- **Phase 1:** Das Ziel von Phase 1 besteht in der möglichst effizienten Erzeugung eines anfänglichen Trainingsdatensatzes, an dem die in Phase 2 eingesetzten Metamodelle trainiert werden können. Dieser anfängliche Trainingsdatensatz muss die Bedingung erfüllen, dass er ausreichend Trainingspunkte ($\mathbf{x}_i, y_i$) sowohl im kritischen als auch im nicht-kritischen Bereich enthält. Als Mindestanforderung sollen mindestens fünf Trainingspunkte in jedem Bereich vorliegen.

  Ein Problem bei der Einhaltung dieser Mindestanforderung ergibt sich insbesondere dann, wenn kritische Ergebnisse der Funktion G nur mit einer sehr kleinen Wahrscheinlichkeit vorkommen und die Berechnung der Ergebnisse relativ viel Rechenzeit in Anspruch nimmt. Bei einer einfachen Monte-Carlo-Simulation mit zufällig gezogenen Werten für die Eingabeparameter wären zu viele Berechnungen durchzuführen, um kritische Ergebnisse zu erhalten.

  Zur Lösung dieses Problems wurde ein Genetischer Algorithmus entwickelt. Durch diesen Algorithmus sollen mit möglichst wenig Rechenläufen Kombinationen von Eingabeparameterwerten gefunden werden, die zu kritischen Ergebnissen führen. Da der Algorithmus in jeder Generation Stichproben von Parametervektoren erzeugt, die sich immer mehr dem kritischen Bereich annähern, wird der Genetische Algorithmus im Folgenden als ‚Genetischer Adaptiver Stichproben Algorithmus' (GASA - **G**enetic **A**daptive **S**ampling **A**lgorithm) bezeichnet. Der GASA-Ansatz wird in Abschnitt 2.2.2 beschrieben.

- **Phase 2**: Wenn der Trainingsdatensatz durch den GASA-Ansatz die Mindestanforderungen erfüllt, wird er verwendet, um verschiedene Metamodelle auf der Basis unterschiedlicher maschineller Lernalgorithmen zu trainieren. Da es keinen einzigen Lernalgorithmus gibt, der die Klassifikation für jede Funktion G gleich gut durchführen kann, wurde ein Ensemble von Lernalgorithmen verwendet. Da dieses Ensemble nicht nur zur Selektion neuer Kandidaten für den Trainingsdatensatz sondern auch zur Schätzung der Wahrscheinlichkeit des kritischen Bereichs verwendet wird, wird die in Phase 2 verwendete Methode im Folgenden als PRECLAS-Ansatz (**Pr**obability **E**stimation with an Ensemble of **Clas**sification Algorithms) bezeichnet. Die Methodik des PRECLAS-Ansatzes wird in Abschnitt 2.2.3 beschrieben.

2.2.2 Genetischer Algorithmus zur Erzeugung eines anfänglichen Trainingsdatensatzes

Um die einzelnen Lernalgorithmen in Phase 2 anwenden zu können, benötigt man einen Trainingsdatensatz **T**, der aus einer sogenannten Lernstichprobe $(\mathbf{x}_i, y_i)_{i=1, \ldots, n}$ von Parametervektoren $\mathbf{x}_i$ und zugehörigen Ergebniswerten y_i einer Funktion G besteht. Die Trainingspunkte $(\mathbf{x}_i, y_i)$ können dabei definierten Klassen zugeordnet werden. Die Methode ist zunächst dafür entwickelt worden, dass die Einteilung in zwei Klassen erfolgt: eine Klasse, die im Folgenden als kritische Klasse (kritischer Bereich) und eine zweite Klasse, die als nicht-kritische Klasse (nicht-kritischer Bereich) bezeichnet wird. Eine kritische Klasse kann z. B. so definiert werden, dass sie einen Bereich von Parametervektoren mit unerwünschten kritischen Funktionswerten umfasst, deren Auftreten möglichst vermieden werden soll. Die Methode kann prinzipiell auch für allgemeinere Fälle weiterentwickelt werden, so dass sie auf mehr als zwei Klassen angewendet werden kann.

Die maschinellen Lernalgorithmen, die in der Phase 2 der Methode eingesetzt werden, können nur mit einem Trainingsdatensatz arbeiten, der sowohl Trainingspunkte $(\mathbf{x}_i, y_i)$ in der kritischen als auch in der nicht-kritischen Klasse enthält. Als Minimalanforderung wurde festgelegt, dass der anfängliche Trainingsdatensatz mindestens fünf Trainingspunkte in jeder Klasse enthalten muss. Diese Minimalanforderung ist willkürlich festgelegt und kann ohne weiteres auch durch eine höhere Anzahl in jeder Klasse ersetzt werden. Für das Nachfolgeprojekt wäre die Untersuchung interessant, ob verschiedene Minimalanforderungen für den anfänglichen Trainingsdatensatz einen Effekt auf die Effizienz des zu erzeugenden Metamodells in Phase 2 haben.

Um mit möglichst wenigen Stichproben einen anfänglichen Trainingsdatensatz zu erzeugen, der die Minimalanforderung erfüllt, wurde der GASA-Ansatz entwickelt, der auf einem Quasi Genetischen Algorithmus beruht. Quasi Genetischer Algorithmus deshalb, weil der entwickelte Algorithmus gegenüber einem echten Genetischen Algorithmus eine vereinfachte Form der Chromosomenbildung verwendet und auf eine binäre Kodierung und Dekodierung der Gene verzichtet. Außerdem verfolgt der hier entwickelte Ansatz eine modifizierte Form der Vererbung, die für die vorliegende Problematik als geeignet erscheint /GOL 89/.

Bevor der GASA-Ansatz vorgestellt wird, sollen zunächst die Voraussetzungen für den GASA-Algorithmus beschrieben werden.

Definition der Klassen

Voraussetzung zur Anwendung des Algorithmus ist die Definition der zu berücksichtigenden Klassen. Dabei muss derjenige Wert der interessierenden Ergebnisgröße angegeben werden, bei dessen Überschreiten (Unterschreiten) ein unerwünschter kritischer Zustand der Größe erreicht wird (z. B. Hüllrohrtemperatur > 1200 °C). Dieser Wert wird im Weiteren kurz als y_{crit} bezeichnet. Die Benutzerspezifikation zur Definition der Klassen ist in Tab. 2.3 angegeben.

Tab. 2.3 Benutzerspezifikation zur Definition kritischer und nicht-kritischer Funktionswerte

Benutzerspezifikation	**Kritischer Bereich**	**Nicht-Kritischer Bereich**
[y_{crit} , '<']	{y: $y \leq y_{crit}$}	{y: $y > y_{crit}$}
[y_{crit} , '>']	{y: $y \geq y_{crit}$}	{y: $y < y_{crit}$}

Mit der Spezifikation [y_{crit} , '<'] wird der kritische Bereich durch die Menge der Funktionswerte definiert, die kleiner oder gleich dem Wert y_{crit} sind. Mit [y_{crit} , '>'] wird der kritische Bereich durch die Menge der Funktionswerte definiert, die größer oder gleich dem Wert y_{crit} sind.

Trainingsdatensatz:

Gegeben sei eine Funktion G mit m variablen Eingabeparametern X_1, …, X_m, die durch ihre jeweiligen Verteilungen F_1, …, F_m definiert sind. Für die Trainingspunkte im Trainingsdatensatz werden bzgl. X_i, i = 1, …, m, zufällig ausgewählte Werte q_i aus einer Gleichverteilung über dem Intervall (0, 1) benötigt. Den eigentlichen Wert von X_i aus der Verteilung F_i erhält man dann unter Ausnutzung der Inversionsmethode, die auf dem sogenannten Simulationslemma basiert. Das Simulationslemma besagt:

Sei F: R -> [0,1] die Verteilungsfunktion der Variablen X und q ein Zufallswert aus der Gleichverteilung zwischen 0 und 1.Dann ist $F^{-1}(q) = x_q$ der zu q gehörige Wert der Verteilung bzw. das q-Quantil der Verteilung F.

Für das q-Quantil einer stetigen Verteilung F gilt:

$$F(x_q) = \int_{-\infty}^{x_q} f(x)dx = q \qquad (2.24)$$

D. h. für q und x_q gilt folgende Beziehung: $P(X \leq x_q) = q$.

Die eigentlichen Werte $\mathbf{x} = (x_1, \ldots, x_m) = (F_1^{-1}(q_1), \ldots, F_m^{-1}(q_m))$ der Eingabeparameter $X_1, \ldots, X_m$ werden nur verwendet, um die entsprechenden Funktionswerte $y = G(\mathbf{x})$ zu ermitteln. Die Stichprobenwerte $\mathbf{q} = (q_1, \ldots, q_m)$ sind zusammen mit den zugehörigen Funktionswerten y die Trainingspunkte für die verwendeten Lernalgorithmen. Die Berücksichtigung von **q** hat den Vorteil, dass die Einflussgrößen bereits auf den Wertebereich zwischen 0 und 1 skaliert sind und nicht nachträglich standardisiert werden müssen.

Der anfängliche Trainingsdatensatz besteht aus einer einfachen Zufallsstichprobe $\mathbf{q_1}, \ldots, \mathbf{q_n}$ und den Funktionswerten $y_1, \ldots, y_n$ mit $= y_j = G(\mathbf{x_j})$ und $\mathbf{x_j} = (x_{1j}, \ldots, x_{mj}) = (F_1^{-1}(q_{1j}), \ldots, F_m^{-1}(q_{mj}))$, $j = 1,\ldots,n$. Anhand der Definition der Klassen werden die Funktionswerte den entsprechenden Klassen zugeordnet.

Der anfängliche Trainingsdatensatz wird vom GASA-Algorithmus verwendet, um neue Kandidaten für den Trainingsdatensatz zu finden, bis die Minimalanforderungen erreicht sind. Wenn die Minimalanforderungen bereits erreicht sind, wird der GASA-Ansatz nicht benötigt. In diesem Fall kann die Wahrscheinlichkeit des kritischen Bereiches als so groß angenommen werden, dass Daten aus dem kritischen Bereich mit akzeptablem Aufwand durch eine einfache Zufallsstichprobe erlangt werden können. Der Default-Wert für die Anzahl der anfänglichen Trainingspunkte beträgt n = 20. Dieser kann aber geändert werden.

Der GASA-Algorithmus beruht auf den wesentlichen Eigenschaften, die ein Genetischer Algorithmus grundsätzlich aufweist:

- Selektion (Survival of the Fittest),
- Vererbung (Cross-Over) und
- Mutation (Random Variation of Genes).

Er kann durch folgende Definitionen und Verfahren beschrieben werden:

(1) **Definition von Individuen und Chromosomen:**

Jeder der n Datenpunkte des Trainingsdatensatzes besteht aus dem Vektor $\mathbf{q_j}$ und dem Funktionswert $y_j = G(\mathbf{x_j})$ mit $\mathbf{x_j} = (x_{1j}, \ldots, x_{mj}) = (F_1^{-1}(q_{1j}), \ldots, F_m^{-1}(q_{mj}))$, j = 1, …, n. Jeder Datenpunkt ($\mathbf{q_j}$, y_j) repräsentiert im Genetischen Algorithmus ein Individuum, das vererbbare Informationen enthält, die in einem Chromosom gespeichert sind. Ein Chromosom besteht aus einer Sequenz von Genen. Im GASA-Ansatz wird ein Chromosom durch den Vektor **q** beschrieben, der sich aus den Genen (q_1, …, q_m) zusammensetzt.

Aus den Individuen des Trainingsdatensatzes werden diejenigen als Eltern ausgewählt, die am besten zur Produktion von gut angepassten Nachkommen geeignet zu sein scheinen. Gut angepasste Individuen sind solche, deren Funktionswerte im oder in der Nähe des kritischen Bereichs liegen. Das Verfahren zur Selektion der Eltern, die zur Produktion von Nachkommen verwendet werden, wird in Abschnitt (3) **'Selektion von Individuen zur Reproduktion'** beschrieben.

(2) **Gene der Individuen:**

Die Gene eines Individuums sind durch die Komponenten q_1, …, q_m des Vektors **q** (Chromosom) gegeben. Die Anzahl der Gene in einem Chromosom entspricht also der Anzahl der Einflussgrößen der Funktion G gegeben. Wie in Abschnitt (4) '**Vererbung**' beschrieben wird, werden den Nachkommen die Gene der Eltern zufällig zugeordnet, wobei die Gene zusätzlich von Mutationen betroffen sind.

(3) **Selektion von Individuen zur Reproduktion:**

Der Prozess der Vererbung wird in aufeinanderfolgenden Generationen ausgeführt. Die erste Generation von Nachkommen basiert auf den Individuen (Datenpunkten) des anfänglichen Trainingsdatensatzes. Von den n Individuen wird lediglich die ‚bessere Hälfte' zur Produktion von Nachkommen zugelassen. Um die 50 % der am besten geeigneten Individuen auszuwählen, werden die Funktionswerte der Datenpunkte in Abhängigkeit von den definierten Klassen in aufsteigender bzw. absteigender Reihenfolge sortiert. Aus den sortierten Werten werden die ersten n/2 Individuen ausgewählt, die als Eltern in Frage kommen. Diese seien mit I_1, I_2, …, $I_{n/2}$ bezeichnet. Bzgl. der Definition der [y_{crit} , '<'] (s. Tab. 3.1) beschreibt I_1 das Individuum mit dem kleinsten Funktionswert, I_2 das Individuum mit dem zweit kleinsten Funktionswert etc. Die Individuen $I_{n/2+1}$, …, I_n werden

für die Produktion von Nachkommen nicht zugelassen. Ist n/2 ungerade, wird zusätzlich das Individuum $I_{n/2+1}$ zur Elterngruppe zugelassen, damit jedem Individuum genau ein Partner zur Erzeugung von Nachkommen zugeordnet werden kann.

Die ausgewählten Individuen werden durch einen Zufallsprozess zu Paaren angeordnet. D. h., einem Individuum wird zufällig ein Partner für die Fortpflanzung zugeordnet. Jedes zufällig zusammengesetzte Elternpaar erzeugt eine feste Anzahl von M Nachkommen (Default: M = 25).

Um die Variabilität der Nachkommen zu steigern, werden die ausgewählten Individuen erneut zu zufälligen Elternpaaren zusammengesetzt. In der Regel unterscheiden sich die neuen Elternpaare von den ersteren. Die neuen Elternpaare produzieren wiederum M Nachkommen. Nach diesem Verfahren werden pro Generation insgesamt 2·n/4·M Nachkommen erzeugt, wobei n/4 die Anzahl der Paare in jedem der beiden Durchgänge der Paarbildung ist.

Alle n/2·M erzeugten Nachkommen einer Generation werden unter Verwendung einer Fitness-Funktion $\mathbb{F}$ danach beurteilt, ob sie für den Trainingsdatensatz geeignet sind oder nicht. Die im GASA-Ansatz angewendete Fitness-Funktion $\mathbb{F}$ wird in Abschnitt (6) ‚**Fitness Funktion**' beschrieben. Die Nachkommen, die mittels $\mathbb{F}$ die besten Beurteilungen erhalten, werden als geeignete Kandidaten für den Trainingsdatensatz ausgewählt. Für die ausgewählten Kandidaten wird unter Verwendung von G(**x**) der zugehörige Funktionswert berechnet. Anschließend werden die Kandidaten mit ihren jeweiligen Funktionswerten als neue Individuen dem Trainingsdatensatz hinzugefügt.

Nachdem der Trainingsdatensatz durch die neuen Individuen ergänzt wurde, wird überprüft, ob der Trainingsdatensatz die Mindestanforderung erfüllt. Falls nicht, wird der genetische Algorithmus zur Erzeugung der nächsten Generation von Nachkommen angewendet.

(4) **Vererbung der Gene:**

Wie in Abschnitt (3) beschrieben, produziert jedes Elternpaar M Nachkommen. Jedes Elternpaar besteht aus zwei Individuen mit unterschiedlichen Chromosomen bzw. Gensequenzen. Die Vererbung der Eltern-Gene auf ihre Nachkommen folgt einem Zufallsprozess. D. h., die Zusammensetzung der Gene eines Nachkommen besteht aus einer zufälligen Anordnung der Eltern-Gene. Zur Veranschaulichung sei angenommen, dass die Chromosomen aller Individuen aus drei Genen zusammengesetzt sind.

Elternteil 1 habe Chromosom (q_1^1, q_2^1, q_3^1) und Elternteil 2 das Chromosom (q_1^2, q_2^2, q_3^2). Das Chromosom eines Nachkommen sei mit (q_1^c, q_2^c, q_3^c) bezeichnet. Für jedes Gen q_i^c des Nachkommen wird zufällig ausgespielt, ob es das Gen q_i^1 vom Elternteil 1 oder q_i^2 vom Elternteil 2 erhält. Wenn der Nachkomme zufällig alle seine Gene vom Elternteil 2 erhält, wäre $(q_1^c, q_2^c, q_3^c) = (q_1^2, q_2^2, q_3^2)$.

Da ein Elternpaar M Nachkommen erzeugt, setzen sich die Chromosomen der Nachkommen aus einer zufälligen Anordnung der Gene der beiden Elternpaare zusammen, z. B.: Nachkomme 1: (q_1^2, q_2^2, q_3^2)

Nachkomme 2: (q_1^1, q_2^2, q_3^1)

$\vdots$

Nachkomme M: (q_1^2, q_2^1, q_3^1).

Zusätzlich sind alle von den Eltern vererbten Gene einer zufälligen Mutation unterworfen. Der Prozess der zufälligen Mutation wird im nachfolgenden Abschnitt (5) beschrieben.

(5) **Mutation der vererbten Gene:**

Alle von den Eltern vererbten Gene sind einer zufälligen Mutation unterworfen. D. h., jedes Gen eines Nachkommen variiert mehr der weniger stark vom vererbten Gen seiner Eltern. Die zufällige Mutation wird durch eine Verteilung bestimmt, die vom vererbten Eltern-Gen abhängt. Die Verteilung ist eine Gleichverteilung U über dem Intervall (q_{low}, q_{up}). Der Wert q_i des vererbten Eltern-Gens wird dabei als Mittelwert der Verteilung $U(q_{low}, q_{up})$ betrachtet. Minimum q_{low} und Maximum q_{up} der Gleichverteilung werden in Abhängigkeit vom vererbten Eltern-Gen q_i bestimmt:

$$q_{low} = q_i - q_i \cdot (1 - q_i)$$

$$q_{up} = q_i + q_i \cdot (1 - q_i)$$

Unter dem Einfluss einer zufälligen Mutation erhält der Nachkomme 1 aus dem Beispiel in Abschnitt (4) nicht exakt die Werte des Elternteils 2, sondern eine mehr oder weniger starke Variation dieser Werte. D. h., anstatt (q_1^2, q_2^2, q_3^2) erhält Nachkomme 1 die Gene $(\widetilde{q_1^2}, \widetilde{q_2^2}, \widetilde{q_3^2})$ wobei $\widetilde{q_i^2}$ zufällig aus der Verteilung $U(q_i^2 - q_i^2 (1 - q_i^2), q_i^2 + q_i^2 \cdot (1 - q_i^2))$ ausgespielt wird mit i=1, 2, 3.

Wie die Gene der Eltern sind auch die Gene **q** der Nachkommen Werte aus dem Intervall (0, 1). Daraus können die Werte **x** der Einflussgrößen unter Anwendung der Inversionsmethode (s. Abschnitt **‚Informationen des Trainingsdatensatzes'**) ermittelt werden.

(6) **Fitness-Funktion:**

Pro Generation wird eine Anzahl von n/2·M Nachkommen erzeugt, wobei n die Anzahl der Individuen (Beobachtungen) im Trainingsdatensatz bzgl. einer Generation und M die Anzahl der Nachkommen pro Elternpaar ist. Aus diesem Pool von Nachkommen werden diejenigen als neue Kandidaten für den Trainingsdatensatz ausgewählt, die potenziell im kritischen Bereich oder in seiner Nähe liegen. Da die Anzahl der Nachkommen zu groß ist, um den tatsächlichen Funktionswert G(**x**) zu berechnen, erfolgt die Beurteilung der Nachkommen durch eine Fitness-Funktion $\mathbb{F}$.

Die Fitness-Funktion $\mathbb{F}$ basiert auf prognostischen Werten, die sich aus der Random-Forest und der Nearest-Neighbor-Methode ergeben. Diese Methoden werden deshalb verwendet, weil mit ihnen sehr schnell Prognosewerte für die entsprechenden Genkombinationen der Nachkommen erzielt werden können. Des Weiteren wird davon ausgegangen, dass die Prognosewerte beider Methoden – auch wenn sie zunächst nicht sehr exakt sein sollten – zumindest der Lage der tatsächlichen Funktionswerte tendenziell entsprechen. D. h. die Prognosewerte korrelieren positiv mit den Funktionswerten. Diese Eigenschaft wird ausgenutzt, um Kandidaten aus den Nachkommen auszuwählen.

Für jeden Nachkommen C_j , j=1, …, n/2·M, werden in der Fitness-Funktion $\mathbb{F}$ vier Größen zur Auswahl der besten Kandidaten verwendet:

i) Der Prognosewert y_{RF}, der sich aus der Random Forest Regressor Methode ergibt.

ii) Der mittlere Prognosewert y_{NN} bzgl. zweier Individuen aus dem Trainingsdatensatz, deren Gene denen des Nachkommens C_j am ähnlichsten sind.

Die Ähnlichkeit der Gene zwischen den Individuen des Trainingsdatensatzes und der erzeugten Nachkommen wird anhand der Nearest-Neighbor-Methode bestimmt. Für die Fitness-Funktion erfolgt die Ermittlung des nächsten Nachbarn (Nearest Neighbor) auf zwei unterschiedliche Arten: Im Berechnungsansatz 1 wird der nächste Nachbar durch das mittlere Manhattan-Distanzmaß berechnet (s. Gleichung 2). Im Berechnungsansatz 2 wird der nächste Nachbar bestimmt, indem das gewichtete Manhattan-Distanzmaß berechnet wird. Die Gewichte der jeweiligen Parameter werden durch das

sogenannte Feature-Importance'-Maß festgelegt, das im Rahmen der Random-Forest Methode ermittelt wird. Eine genauere Beschreibung dazu ist im SUSA-Methodenband /KLO 21b/ zu finden.

Für alle erzeugten Nachkommen wird in beiden Berechnungsansätzen das ähnlichste Individuum aus dem Trainingsdatensatz ermittelt und sein Funktionswert als Prognosewert übernommen. Wenn der Prognosewert des nächsten Nachbarn bzgl. des Berechnungsansatzes 1 y_{NN1} ist und bzgl. des Berechnungsansatzes 2 y_{NN2}, so wird für den endgültigen Prognosewert y_{NN} eines Nachkommen der Mittelwert von y_{NN1} und y_{NN2} verwendet, d. h. $y_{NN} = \frac{y_{NN1} + y_{NN2}}{2}$.

iii) Ein Maß für die Ähnlichkeit des Nachkommens mit seinem ähnlichsten Individuum.

Sei $\mathbf{q}(C_j)$ das Chromosom des Nachkommen C_j und $\mathbf{q}(I_{NN})$ das Chromosom seines ähnlichsten Individuums I_{NN}. Als Maß für die Ähnlichkeit von C_j und I_{NN} wird der Abstand der Chromosomen der beiden Individuen berechnet. Der Abstand wird mit δq bezeichnet und wird durch die mittlere Manhattan-Distanz ermittelt:

$$\delta q = \frac{\sum_{i=1}^{m} | q_i(C_j) - q_i(I_{NN}) |}{m} \tag{2.25}$$

wobei $q_i(C_j)$ und $q_i(I_{NN})$ die entsprechenden Gene des Nachkommens C_j und des Individuums I_{NN} sind.

iv) Der Abstand δy zwischen den Prognosewerten y_{RF} und y_{NN}, d. h. $\delta y = | y_{RF} - y_{NN} |$.

Die obigen vier Größen y_{RF}, y_{NN}, δq und δy werden für alle erzeugten Nachkommen jeder Generation berechnet, um die Kandidaten für den Trainingsdatensatz auszuwählen. Die Auswahlprozedur der Kandidaten wird im folgenden Absatz (7) beschrieben.

(7) **Auswahlprozedur der Kandidaten für den Trainingsdatensatz:**

Da die Auswahl der Kandidaten von der Definition der Klassen abhängt, wird das Auswahlverfahren der Fitness-Funktion für die Klassendefinition [y_{crit}, ‚<'] (s. Tab. 2.3)

beschrieben. Änderungen der Auswahlprozedur, die sich für den Fall [y_{crit}, ‚>'] ergeben, werden in der Beschreibung ausdrücklich erwähnt.

- Für alle in einer Generation erzeugten Nachkommen C_j, j = 1, …, n/2·M wird eine Vorauswahl durchgeführt. In dieser Vorauswahl wird ein Nachkomme C_j verworfen, wenn entweder sein Prognosewert $y_{RF}(C_j) \geq y_{50}$ oder der mittlere Prognosewert der beiden ähnlichsten Individuen $y_{NN}(C_j) \geq y_{50}$ ist. y_{50} bezeichnet dabei den Median der Funktionswerte y_1, …, y_n des Trainingsdatensatzes. Im Fall [y_{crit} , '>'] wird ein Nachkomme verworfen, wenn $y_{RF}(C_j) \leq y_{50}$ oder $y_{NN}(C_j) \leq y_{50}$ ist. Diese Vorauswahl basiert auf der Annahme, dass Nachkommen mit zu schlechten Prognosewerten, keine gute Annäherung des Individuums an die kritische Region aufzeigen werden.

- Alle Nachkommen, die die Vorauswahl bestanden haben, werden bzgl. $y_{RF,}$, y_{NN}, δq und δy sortiert. Im Fall von [y_{crit} , '<'] werden $y_{RF,}$ und y_{NN} in aufsteigender Reihenfolge, im Fall von [y_{crit} , '>'] in absteigender Reihenfolge sortiert. Die Größen δq und δy werden in beiden Fällen in absteigender Reihenfolge sortiert. Die sortierten Werte seien in den Vektoren $\mathbf{y}_{RF}$, $\mathbf{y}_{NN}$, **δ**q und **δ**y angeordnet.

- Von den sortierten Werten in $\mathbf{y}_{RF}$ werden die ersten 5 ausgewählt. Das sind die Nachkommen, die bzgl. y_{RF} die 5 besten Prognosewerte aufweisen. Im Fall von [y_{crit} , '<'] sind das die Nachkommen mit den 5 kleinsten, im Fall von [y_{crit} , '>'] die Nachkommen mit den 5 größten y_{RF} Werten.

- Von den 5 ausgewählten Nachkommen mit den besten y_{RF} Prognosewerten werden zwei Kandidaten für den Trainingsdatensatz ausgewählt. Als erster Kandidat wir der ausgewählt, der die größte Distanz δq zu seinem nächsten Nachbarn (Nearest Neighbor) aufweist. Als zweiter Kandidat wird der ausgewählt, der von den 5 ausgewählten Nachkommen den größten Wert $\delta y = |y_{RF} - y_{NN}|$ aufweist. Handelt es sich um den gleichen Nachkommen, wird derjenige mit dem nächstgrößten Wert δy ausgewählt. Dieses Vorgehen wird als erste Regel der Kandidatenauswahl bezeichnet.

- Die Idee hinter dieser Auswahl besteht darin, Eingabekombinationen zu entdecken, die im oder in der Nähe des kritischen Bereichs liegen und die sich von den Eingabekombinationen der Datenpunkte des Trainingsdatensatzes möglichst unterscheiden. Es soll also die Auswahl von solchen Kandidaten vermieden werden, deren Gene im Trainingsdatensatz in ähnlicher Weise bereits enthalten sind. Dies wird in Abb. 2.4 veranschaulicht.

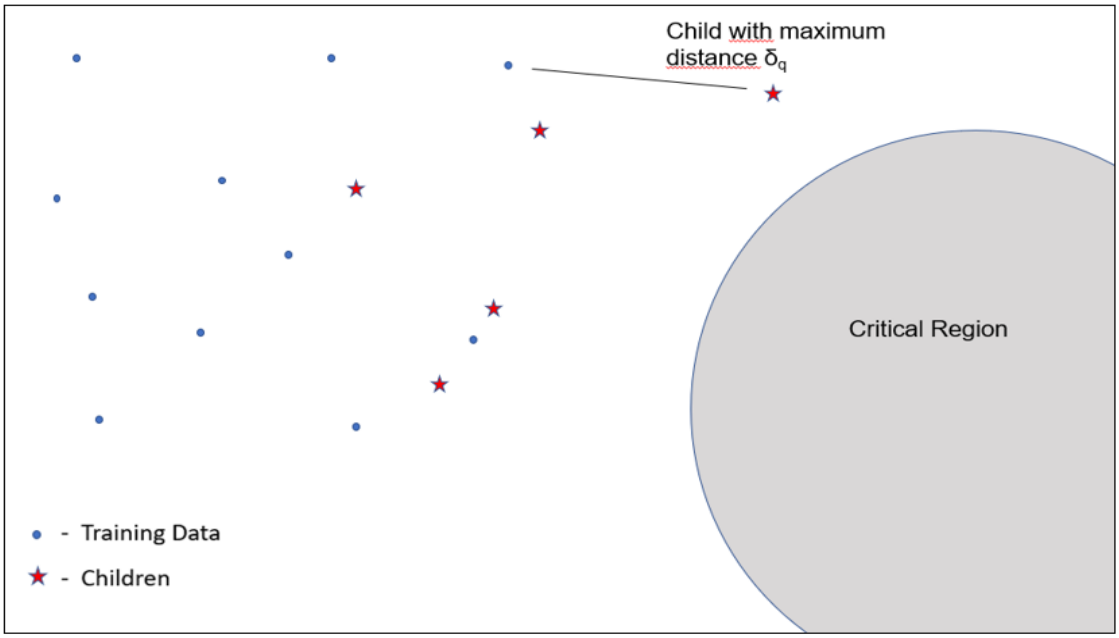

Abb. 2.4 Auswahl des Nachkommens mit maximaler Distanz δq zu seinem nächsten Nachbarn

In Abb. 2.4 sind die fünf ausgewählten Nachkommen mit den besten y_{RF} Prognosewerten dargestellt. Der Kandidat mit der größten Distanz δq zu seinem nächsten Nachbarn ist derjenige im oberen rechten Bereich der Abbildung, der in der Nähe des kritischen Bereichs liegt. Dieser Bereich ist noch nicht von Individuen des Trainingsdatensatzes besetzt. Um Gebiete von Eingabekombinationen zu finden, deren Funktionswerte in der Nähe des kritischen Bereichs liegen und die noch nicht in ähnlicher Weise im Trainingsdatensatz vorliegen, wurde darauf geachtet, dass die Auswahlkriterien der Kandidaten eine gewisse Variabilität beinhalten. Aus diesem Grund hängt der Auswahlprozess nicht nur von einer Größe, sondern von den vier Größen y_{RF}, y_{NN}, δq and δy ab.

Bzgl. der 2. Auswahlregel der Kandidaten erfolgt die Sortierung der Prognosewerte bzgl. der Summe der beiden Größen y_{RF} und y_{NN}, d.h. $y_{RF} + y_{NN}$. Die Ergebnisse der Sortierung sind im Vektor $\mathbf{y}_{NN+RF}$ angeordnet. Aus dem Vektor $\mathbf{y}_{NN+RF}$ werden die ersten 5 ausgewählt. Analog zur 1. Auswahlregel werden von diesen diejenigen als Kandidaten für den Trainingsdatensatz bestimmt, die den maximalen Wert δq bzw. δy aufweisen.

Die 3. Auswahlregel der Kandidaten für den Trainingsdatensatz besteht darin, dass die Nachkommen bzgl. der Größe δq in absteigender Reihenfolge sortiert werden. Aus den sortierten Werten in **δ**q werden diejenigen 10 Nachkommen mit den Größen Distanzen zu ihrem Nächsten-Nachbarn ausgewählt. Von diesen werden diejenigen als Kandidaten

ausgewählt, die die kleinsten Prognosewerte y_{RF} bzw. die kleinsten Werte $y_{RF} + y_{NN}$ im Fall von [y_{crit} , '<'] aufweisen. Im Fall von [y_{crit} , '>'] sind dies die Nachkommen mit den jeweils größten Werten y_{RF} bzw. $y_{RF} + y_{NN}$.

Bei der 4. Auswahlregel werden die Nachkommen bzgl. des Abstands $\delta y = |y_{RF} - y_{NN}|$ in absteigender Reihenfolge sortiert. Von den 10 Nachkommen mit den größten Abständen werden analog zur 3. Auswahlregel diejenigen als Kandidaten für den Trainingsdatensatz gewählt, die die kleinsten Werte y_{RF} bzw. $y_{RF} + y_{NN}$ im Fall von [y_{crit}, '<'] aufweisen. Im Fall von [y_{crit}, '>'] sind dies die Nachkommen mit den jeweils größten Werten y_{RF} bzw. $y_{RF} + y_{NN}$.

Bei der Anwendung der Regeln für den Selektionsprozess wird darauf geachtet, dass ein Nachkomme nur einmal als Kandidat ausgewählt wird. D. h., ein Nachkomme wird nur dann als Kandidat ausgewählt, wenn er nicht schon vorher als Kandidat bestimmt wurde.

Ein Problem, das insbesondere im Fall komplexer nichtlinearer Funktionen auftritt, besteht darin, dass kritische Werte in sehr unterschiedlichen Parameterbereichen auftreten können. Diese Problematik soll am Beispiel der nicht-linearen Ishigami-Funktion veranschaulicht werden (siehe Abschnitt 2.2.4.2). In Abb. 2.5 bis Abb. 2.7 können für jeden Parameter die Bereiche der Werte abgelesen werden, bei denen kritische Funktionswerte auftreten. Die Definition des kritischen Bereichs ist dabei gegeben durch [-10, ,<']. D. h., der kritische Bereich ist durch Funktionswerte $y \leq -10$ definiert.

In Abb. 2.5 wird die Abhängigkeit der Funktionswerte y von den Werten der Gene (q_1) des Parameters 1 dargestellt. Es wird deutlich, dass kritische Funktionswerte ≤ -10 dann auftreten, wenn der Parameter X_1 Werte x annimmt, deren Wahrscheinlichkeiten $q_1 = F_{X1}(x)$ im Bereich zwischen 0.2 und 0.3 liegen. D. h. für Parameter X_1 kann ein Wertebereich von q_1 spezifiziert werden, bei dem die Funktionswerte y in den kritischen Bereich fallen.

In Abb. 2.6 zeigt sich, dass es bzgl. Parameter X_2 drei unterschiedliche Bereiche gibt, bei denen kritische Funktionswerte ≤ -10 auftreten können. Der erste Bereich ist durch Parameterwerte von X_2 gegeben, deren Wahrscheinlichkeiten $q_2 < 0.05$ sind. Der zweite Bereich ist durch Parameterwerte von X_2 gegeben, deren Wahrscheinlichkeiten q_2 ungefähr zwischen 0.45 und 0.55 liegen und der dritte Bereich durch Werte $q_2 > 0.95$.

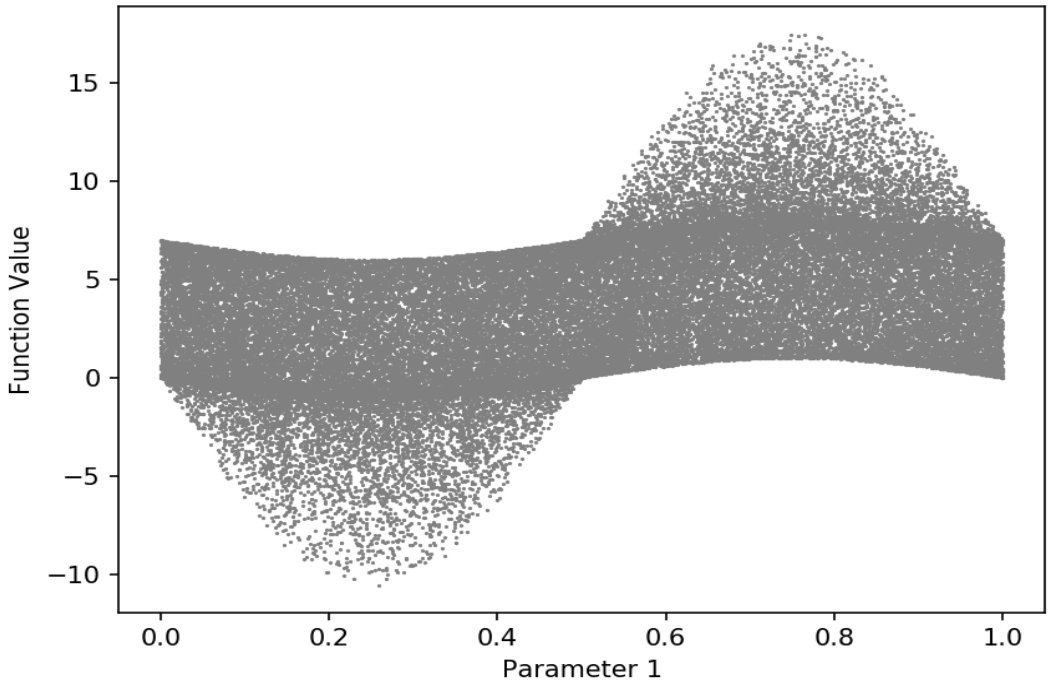

Abb. 2.5 Abhängigkeit der Funktionswerte y von den Werten der Gene (q_1) des Parameters X_1 (Simulation von 50000 Werten)

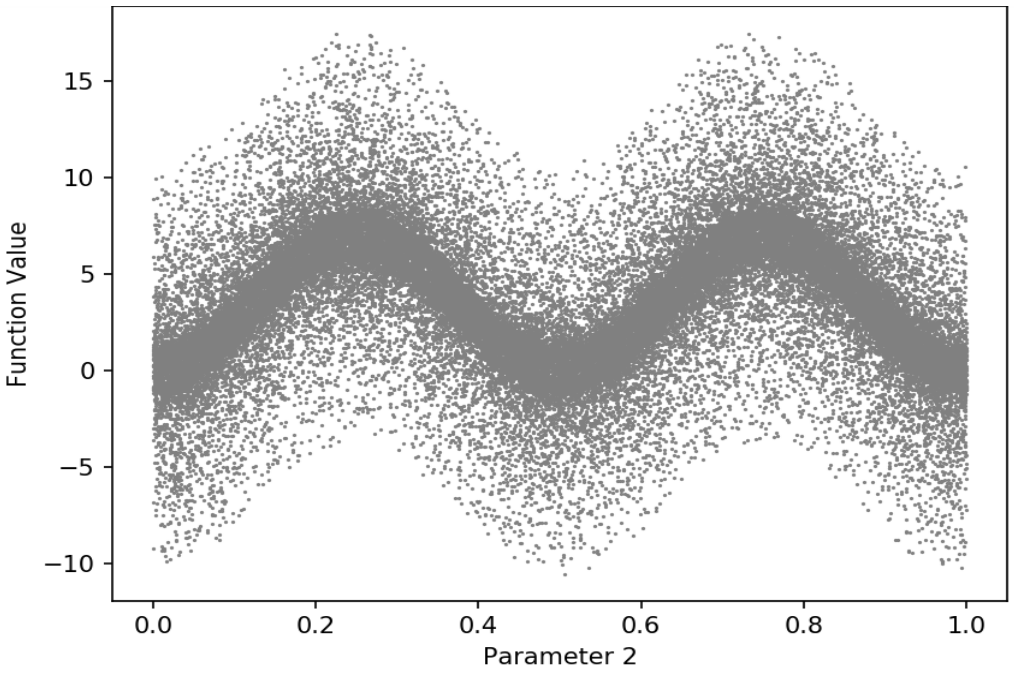

Abb. 2.6 Abhängigkeit der Funktionswerte y von den Werten der Gene (q_2) des Parameters X_2 (Simulation von 50000 Werten)

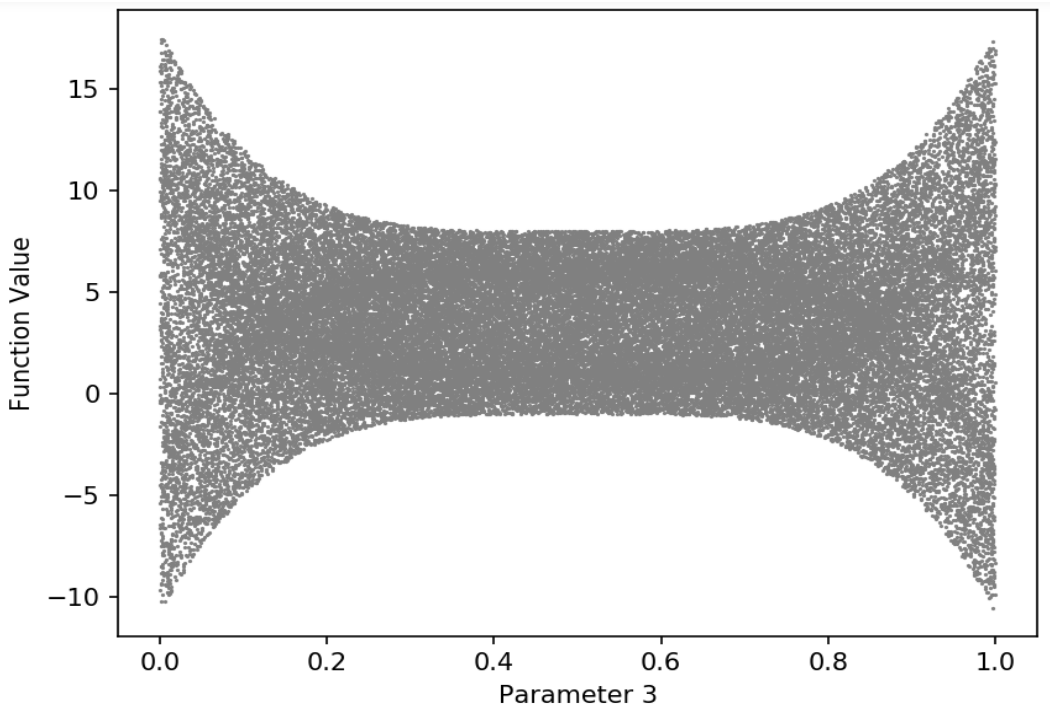

Abb. 2.7 Abhängigkeit der Funktionswerte y von den Werten der Gene (q_3) des Parameters X_3 (Simulation von 50000 Werten)

Parameter X_3 weist zwei Bereiche auf, bei denen kritische Funktionswerte auftreten (s. Abb. 2.7). Dies sind die Bereiche, bei denen Parameter X_3 Werte annimmt, deren Wahrscheinlichkeiten q_3 im Bereich < 0.05 oder im Bereich > 0.98 auftreten.

Eine wünschenswerte Eigenschaft des Selektionsprozesses ist es, dass er in der Lage ist, möglichst viele Parameterregionen zu entdecken, die mit kritischen Funktionswerten verbunden sind. Aus diesem Grund werden in den oben beschriebenen vier Auswahlregeln unterschiedliche Kriterien für den Selektionsprozess angewendet. Damit soll die Variabilität des Selektionsprozesses erhöht werden, um möglichst viele kritische Bereiche der zugrundeliegenden Funktion zu finden.

Die Auswahl der Eltern, ihr Vererbungsprozess sowie der Auswahlprozess neuer Kandidaten, wie sie für die GASA-Methode entwickelt und implementiert wurden, haben die Eigenschaft, dass sich die ausgewählten Kandidaten in aufeinanderfolgenden Generationen immer mehr dem kritischen Bereich der zugrundeliegenden Funktion annähern. Mit dem GASA-Ansatz wurde eine Methode entwickelt, die geeignet ist, einen für Klassifikationsverfahren anwendbaren Trainingsdatensatz effizient zu erzeugen. Der iterative Ablauf des GASA-Ansatzes wird in dem Flussdiagramm in Abb. 2.8 veranschaulicht.

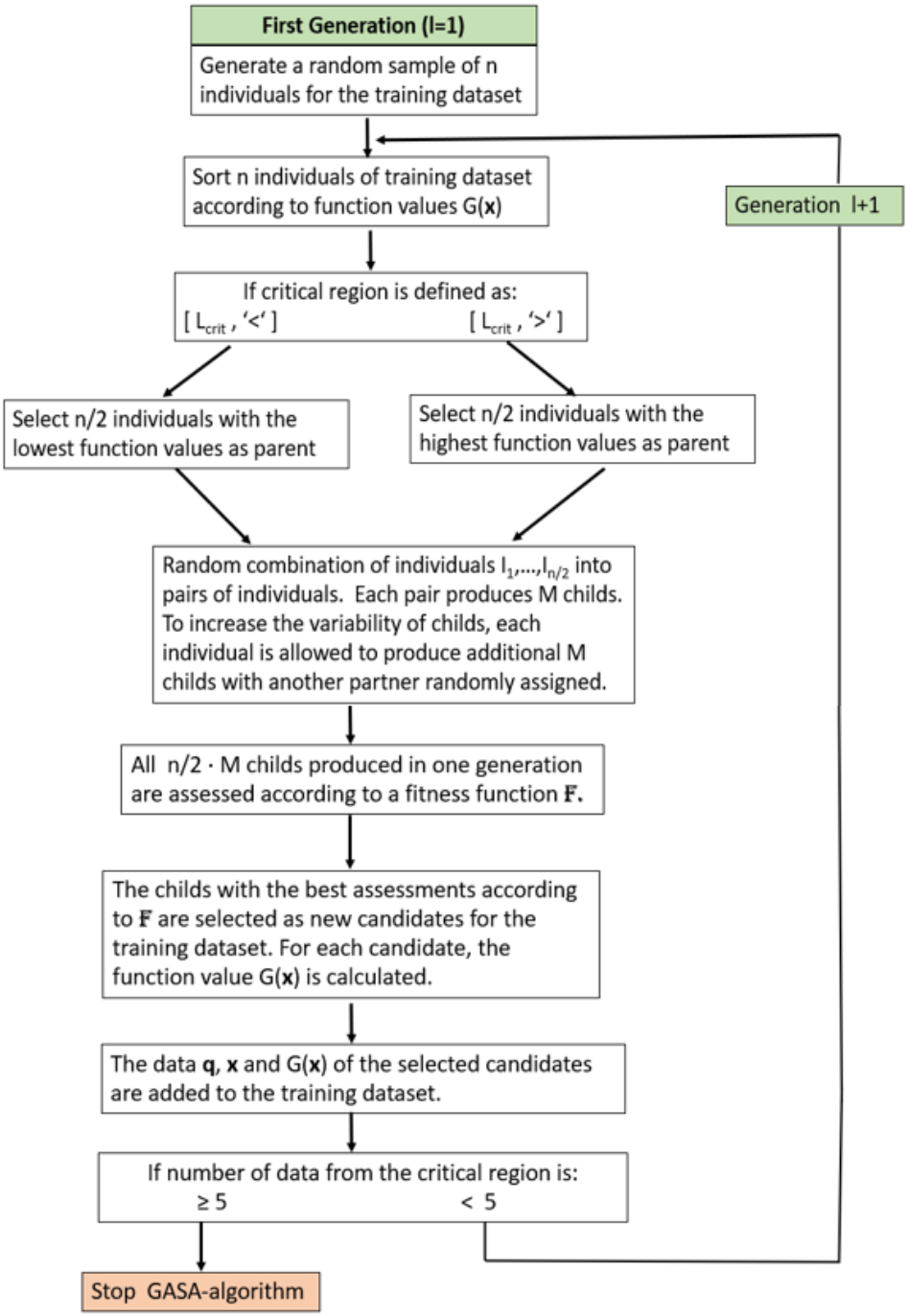

Abb. 2.8 Flussdiagramm für den Ablauf des GASA Algorithmus

Wenn der Trainingsdatensatz durch den GASA-Algorithmus soweit erstellt ist, dass er die Mindestanforderung erfüllt, wird der GASA-Algorithmus beendet, und der erstellte Trainingsdatensatz kann für das Ensemble von Klassifikationsverfahren in Phase 2 verwendet werden. Die dazu entwickelte Methode PRECLAS (Probability Estimation by an Ensemble of Classification Approaches) wird im nachfolgenden Abschnitt 2.2.3 beschrieben.

2.2.3 PRECLAS-Methode zur Wahrscheinlichkeitsschätzung kritischer Bereiche

Wenn der anfängliche Trainingsdatensatz durch den GASA-Ansatz erstellt wurde, kann auf diesen in Phase 2 die PRECLAS Methode für folgende Aufgaben angewendet werden:

(i) Zur Schätzung der Wahrscheinlichkeit, mit der kritische Funktionswerte einer komplexen Funktion auftreten.

(ii) Zur Verbesserung des Metamodells durch Auswahl weiterer geeigneter Kandidaten, die dem Trainingsdatensatz hinzugefügt werden.

Die PRECLAS Methode kann als ein Ensemble von maschinellen Lernalgorithmen betrachtet werden, die eine Klassifikation von Testdaten in den kritischen bzw. nicht-kritischen Bereich der zugrundeliegenden Funktion erlauben. Das Ensemble besteht aus den folgenden Methoden:

- K-Nearest Neighbor (KNN)
- Entscheidungs-Baum (Decision Tree)
- Gauss'scher Naiver-Bayes-Klassifikator (Gaussian Naive Bayes)
- Nicht-Parametrischer Naiver-Bayes-Klassifikator (Non-Parametric Naive Bayes)
- Random-Forest-Klassifikator
- Random-Forest-Regressor

Eine genauere Beschreibung der Algorithmen ist im SUSA Methodenband /KLO 21b/ zu finden. Die Vorgehensweise der entwickelten PRECLAS Methode kann durch folgende Schritte beschrieben werden:

(1) Für jeden Klassifikator in der PRECLAS Methode wird ein Metamodell auf der Basis des durch GASA erzeugten anfänglichen Trainingsdatensatzes erstellt. Die Modelle werden verwendet, um vorherzusagen, ob die Funktionswerte neuer Testdaten, in den kritischen oder nicht kritischen Bereich fallen. Der Grund, warum die Vorhersage auf Metamodellen mehrerer unterschiedlicher Verfahren basiert, besteht darin, dass kein Lernalgorithmus bekannt ist, der in allen Situationen als der Beste betrachtet werden kann. D. h. für verschiedene Funktionen können sich unterschiedliche Lernalgorithmen als vorteilhaft erweisen. Um ein möglichst breites

Spektrum unterschiedlichster Funktionen bearbeiten zu können, wurden deshalb mehrere unterschiedliche Lernalgorithmen verwendet.

(2) Nach der Konstruktion der unterschiedlichen Metamodelle auf Basis des Trainingsdatensatzes wird eine große Stichprobe $\mathbf{x}_{test} = (x_{1test}, \ldots, x_{mtest})$ von n Testdaten zufällig ausgespielt. Daraus werden die entsprechende Wahrscheinlichkeitswerte $\mathbf{q}_{test} = (q_{1,test}, \ldots, q_{m,test})$ durch $q_{i,test} = F_{Xi}(x_{i,test})$ (s. Gl. (2.24)) bestimmt. Die Metamodelle werden eingesetzt, um Prognosen bzgl. der Klassifikation der Testdaten zu liefern.
Die Vorhersage bzgl. des Nearest-Neighbor-Algorithmus ist durch den Funktionswert derjenigen Beobachtung des Trainingsdatensatzes gegeben, dessen Input-Vektor **q** dem des Testinputs $\mathbf{q}_{test}$ am ähnlichsten ist. Als Ähnlichkeitsmaß wird der Mittelwert der absoluten Differenzen (mittlere Manhattan-Distanz)

$\delta(\mathbf{q}, \mathbf{q}_{Test}) = \frac{\sum_{i=1}^{m} | q_i - q_{i,test} |}{m}$ (vgl. Gl. (2.25)) verwendet.

Der Prognosewert des Nearest-Neighbor-Verfahrens wird im Folgenden mit $\hat{y}_{\mathrm{NN}}$ bezeichnet. Der Vorhersagewert wird als kritisch klassifiziert, wenn

$\hat{y}_{\mathrm{NN}} \leq y_{crit}$, falls [y_{crit} , '<'] (s. Tab. 2.3)

$\hat{y}_{\mathrm{NN}} \geq y_{crit}$, falls [y_{crit} , '>']

Die Vorhersage $\hat{y}_{\mathrm{DT}}$ des Decision-Tree Modells gibt die Klasse an, in die die Testdaten gemäß ihrem Input $\mathbf{q}_{test}$ jeweils klassifiziert werden.

Die Vorhersagen $\hat{y}_{\mathrm{GNB}}$ bzw. $\hat{y}_{\mathrm{npNB}}$ des Gaussian bzw. des nicht-parametrischen Naive Bayes Modells geben ebenfalls die Klasse an, in die die Testdaten jeweils klassifiziert werden. Zusätzlich erhält man die Wahrscheinlichkeiten p_{GNB}^{crit} und $p_{GNB}^{non\text{-}crit}$ bzw. p_{npNB}^{crit} und $p_{npNB}^{non\text{-}crit}$, mit denen die Testdaten der kritischen bzw. nicht-kritischen Klasse gemäß ihrem Input $\mathbf{q}_{test}$ jeweils zugeordnet werden. Ein Datenpunkt aus den Testdaten wird der kritischen Klasse zugeordnet, falls $p_{GNB}^{crit} > p_{GNB}^{non\text{-}crit}$ bzw. $p_{npNB}^{crit} > p_{npNB}^{non\text{-}crit}$ gilt.

Mit dem Random-Forest-Algorithmus können zwei unterschiedliche Vorhersagen ermittelt werden:

a) Der Random-Forest-Klassifikator liefert als Vorhersage $\hat{y}_{\mathrm{RFp}}$ die Klasse, in der ein Datenpunkt der Testdaten liegt. Zusätzlich erhält man die Wahrscheinlichkeiten p_{RF}^{crit} und $p_{RF}^{non\text{-}crit}$. Die Testdaten werden der kritischen Klasse zugeordnet, falls $p_{RF}^{crit} > p_{RF}^{non\text{-}crit}$ gilt.

b) Der Random-Forest-Regressor liefert den Wert $\hat{y}_{\mathrm{RF}}$, der als Funktionswert für den Testinput $\mathbf{q}_{test}$ vorhergesagt wird. Die Testdaten werden der kritischen Klasse zugeordnet, wenn $\hat{y}_{\mathrm{RF}} \leq y_{crit}$ und [y_{crit} , '<'] bzw. $\hat{y}_{\mathrm{RF}} \geq y_{crit}$ und [y_{crit} , '>'] (s. Tab. 2.3) gilt.

(3) Die Random-Forest-Modelle benötigen im Fall großer Stichproben mehr Rechenzeit als die einfacheren Nearest-Neighbor-, Decision-Tree- und Naiv-Bayes-Modelle. Um die Effizienz der PRECLAS-Methode zu verbessern, werden bei einer gegebenen Stichprobe von Testdaten für jeden Testinput $\mathbf{q}_{test}$ in einer ersten Abschätzung zunächst die Vorhersagen $\hat{y}_{\mathrm{NN}}$, $\hat{y}_{\mathrm{DT}}$, $\hat{y}_{\mathrm{GNB}}$ und $\hat{y}_{\mathrm{npNB}}$ der schneller laufenden Modelle ermittelt.

Nach der Ermittlung dieser Vorhersagewerte werden verschiedene Regeln zur Entscheidung angewendet, ob die Berechnungen der aufwendigeren Random-Forest-Modelle zusätzlich durchgeführt werden oder nicht. D. h., die entwickelten Regeln sollen eine Bewertung über die Unsicherheit der prognostizierten Werte $\hat{y}_{\mathrm{NN}}$, $\hat{y}_{\mathrm{DT}}$, $\hat{y}_{\mathrm{GNB}}$ und $\hat{y}_{\mathrm{npNB}}$ abgeben. Werden die Prognosewerte als zu unsicher bewertet, werden zusätzlich die Prognosen $\hat{y}_{\mathrm{RFp}}$ und $\hat{y}_{\mathrm{RF}}$ der Random-Forest Modelle angefordert. (Untersuchungen zur Validität der Entscheidungsregeln bleiben dem Nachfolgevorhaben vorbehalten.). Hierzu wird folgende Vorgehensweise angewendet:

- Wenn die Klassifikationen der Vorhersagewerte $\hat{y}_{\mathrm{NN}}$, $\hat{y}_{\mathrm{DT}}$, $\hat{y}_{\mathrm{GNB}}$ und $\hat{y}_{\mathrm{npNB}}$ für einen Testfall $\mathbf{q}_{test}$ nicht übereinstimmen, werden auf jeden Fall die Prognosen der Random Forest Modelle ermittelt.
- Wenn die Vorhersagewerte $\hat{y}_{\mathrm{NN}}$, $\hat{y}_{\mathrm{DT}}$, $\hat{y}_{\mathrm{GNB}}$ und $\hat{y}_{\mathrm{npNB}}$ alle dieselbe Klassenzugehörigkeit bzgl. $\mathbf{q}_{test}$ liefern, werden verschiedene Regeln zur

Entscheidung angewendet, ob die Berechnungen der aufwendigeren Random-Forest Modelle zusätzlich durchgeführt werden oder nicht.

Regel 1: Wie oben erläutert wurde, basiert $\hat{y}_{\mathrm{NN}}$ auf dem Distanzmaß $\delta(\mathbf{q}, \mathbf{q}_{\mathbf{Test}})$ der mittleren Manhattan Abweichung. Falls $\delta(\mathbf{q}, \mathbf{q}_{\mathbf{Test}}) \leq 0.005$, wird die Ähnlichkeit zwischen den Input-Werten als so groß betrachtet, dass der Funktionswert des nächsten Nachbarn $\hat{y}_{\mathrm{NN}}$ als hinreichend sichere Vorhersage betrachtet wird. In diesem Fall wird kein anderer Prognosewert mehr benötigt. Falls $\delta(\mathbf{q}, \mathbf{q}_{\mathbf{Test}}) > 0.005$, kommt die Regel 2 zur Anwendung.

Regel 2: Die Wahrscheinlichkeiten (p_{GNB}^{crit} , $p_{GNB}^{non\text{-}crit}$) des Gauss'schen Naive-Bayes Modells und (p_{npNB}^{crit} , $p_{npNB}^{non\text{-}crit}$) des nicht-parametrischen Naive-Bayes Modells werden verwendet, um die mittleren Wahrscheinlichkeiten der Zuordnung zur kritischen bzw. nicht-kritischen Klasse zu berechnen:

$$p_{Mean1}^{crit} = \frac{1}{2} \cdot \left(p_{GNB}^{crit} + p_{npNB}^{crit} \right)$$

$$p_{Mean1}^{non\text{-}crit} = \frac{1}{2} \cdot \left(p_{GNB}^{non\text{-}crit} + p_{npNB}^{non\text{-}crit} \right)$$

Falls das Maximum der Wahrscheinlichkeiten $\max(p_{Mean}^{crit}, p_{Mean}^{non\text{-}crit}) \leq 0.6$ ist, wird die Zuordnung zu einer Klasse als nicht sicher genug betrachtet, obwohl alle Prognosen von $\mathbf{q}_{test}$ die gleiche Klassifizierung ergeben haben. In diesen Fall lautet die Entscheidung, die Prognosen $\hat{y}_{\mathrm{RFp}}$ und $\hat{y}_{\mathrm{RF}}$ der Random-Forest Modelle zusätzlich zu berechnen.

Falls zusätzlich zur eindeutigen Klassifizierung der Prognosen das Maximum der Wahrscheinlichkeiten $\max(p_{Mean1}^{crit}, p_{Mean1}^{non\text{-}crit}) > 0.6$ gilt, wird die Zuordnung zu einer Klasse als hinreichend eindeutig bewertet, so dass auf die Berechnung von $\hat{y}_{\mathrm{RFp}}$ und $\hat{y}_{\mathrm{RF}}$ verzichtet wird.

(4) Falls die Vorhersagen $\hat{y}_{\mathrm{NN}}$, $\hat{y}_{\mathrm{DT}}$, $\hat{y}_{\mathrm{GNB}}$ und $\hat{y}_{\mathrm{npNB}}$ nicht eindeutig sind oder $\max(p_{Mean}^{crit}, p_{Mean}^{non\text{-}crit}) \leq 0.6$ werden zusätzlich die Vorhersagen $\hat{y}_{\mathrm{RFp}}$ und $\hat{y}_{\mathrm{RF}}$ der Random-Forest Modelle berechnet. Unter Verwendung aller Vorhersagen werden verschiedene Berechnungen zur Erfüllung folgender Aufgaben durchgeführt:

a) Identifikation derjenigen Testdaten, deren Zuordnung zu einer Klasse zu unsicher ist.

b) Schätzung der Wahrscheinlichkeit des kritischen Bereichs.

c) Entscheidung, ob der PRECLAS Algorithmus beendet und die Wahrscheinlichkeitsschätzung als Ergebnis verwendet werden kann, oder ob der Trainingsdatensatz erweitert werden muss, um die Schätzung der Wahrscheinlichkeit des kritischen Bereiches zu verbessern.

d) Auswahl geeigneter Kandidaten für den Trainingsdatensatz aus der Stichprobe der Testdaten.

(5) Zur Identifikation von Testdaten, deren Zuordnung zu einer Klasse als zu unsicher bewertet werden, werden nachfolgende Kriterien angewendet:

i) Die Wahrscheinlichkeiten (p_{GNB}^{crit} , $p_{GNB}^{non\text{-}crit}$) des Gauss'schen Naive-Bayes Modells und (p_{npNB}^{crit} , $p_{npNB}^{non\text{-}crit}$) des nicht-parametrischen Naive-Bayes Modells sowie die Wahrscheinlichkeiten (p_{RF}^{crit}, $p_{RF}^{non\text{-}crit}$) des Random Forest Klassifikators werden verwendet, um die mittleren Wahrscheinlichkeiten der Zuordnung zur kritischen bzw. nicht-kritischen Klasse zu berechnen:

$$p_{Mean2}^{crit} = \frac{1}{3} \cdot (p_{RF}^{crit} + p_{GNB}^{crit} + p_{npNB}^{crit})$$

$$p_{Mean2}^{non\text{-}crit} = \frac{1}{3} \cdot (p_{RF}^{non\text{-}crit} + p_{GNB}^{non\text{-}crit} + p_{npNB}^{non\text{-}crit})$$

Eine Zuordnung wird als zu unsicher betrachtet, falls das Maximum der mittleren Wahrscheinlichkeiten $\max(p_{Mean2}^{crit}, p_{Mean2}^{non\text{-}crit}) \leq 0.6$ ist. Das bedeutet, dass in diesem Fall die Wahrscheinlichkeit der Zuordnung zu einer Klasse als zu gering betrachtet wird.

ii) Weitere Kriterien einer unsicheren Zuordnung sind gegeben, wenn die Vorhersage des Random-Forest Regressors sich nicht mit der des Nearest-Neighbor Verfahrens oder mit der Klassifizierung des Random-Forest Klassifikators deckt. D. h., eine Zuordnung wird als unsicher betrachtet, wenn gilt:

$(\hat{y}_{RF} \leq y_{crit} \wedge \hat{y}_{NN} > y_{crit})$ bzw. $(\hat{y}_{RF} > y_{crit} \wedge \hat{y}_{NN} \leq y_{crit})$

oder

($\hat{y}_{\mathrm{RF}}$ liegt im kritischen Bereich $\wedge$ $p_{RF}^{crit} < p_{RF}^{non\text{-}crit}$) bzw.

($\hat{y}_{\mathrm{RF}}$ liegt im nicht-kritischen Bereich $\wedge$ $p_{RF}^{crit} > p_{RF}^{non\text{-}crit}$)

Wenn mindestens einer von den in i) oder ii) genannten Kriterien erfüllt ist, wird die Vorhersage als unsicher bewertet. Der Wahrscheinlichkeitswert von 0.6 in der Bedingung i) wurde zwar willkürlich festgelegt, hat sich in den durchgeführten Tests jedoch als praktikabel erwiesen. Durch eine Erhöhung der Zuordnungswahrscheinlichkeit würde sich die Anzahl der unsicheren Testdaten erhöhen, was auch zu einer Erhöhung des Rechenaufwandes führen würde.

(6) Aus der Gesamtheit der als unsicher bewerteten Testdaten werden durch einen Selektionsprozess die Kandidaten ausgewählt, deren Funktionswerte mit dem eigentlichen Simulationscode berechnet und die als neue Datenpunkte dem Trainingsdatensatz hinzugefügt werden. Für den PRECLAS-Algorithmus wird für die Auswahl der Kandidaten folgendes Verfahren angewendet:

Für die Gesamtheit der Testdaten, deren Klassifikation nach den unter Punkt (5) beschriebenen Kriterien als zu unsicher betrachtet werden, wird eine Clusteranalyse nach dem k-Means-Algorithmus durchgeführt. Die prinzipielle Eigenschaft der Clusteranalyse ist es, Daten in Gruppen (Cluster) einzuteilen, wobei sich die Daten innerhalb der Cluster möglichst wenig und die Daten zwischen den Clustern möglichst stark unterscheiden. Diese Eigenschaft der Clusteranalyse wird ausgenutzt, um solche Kandidaten auszuwählen, die möglichst unterschiedlich sind und potenziell zu noch nicht entdeckten kritischen Bereichen gehören.

Die k-Means Clusteranalyse teilt die Daten x in k disjunkte Cluster $C_1,\ldots, C_k$ auf, wobei die Summe der Abweichung der Daten in den jeweiligen Clustern von ihren jeweiligen Cluster-Zentroiden minimal ist, d. h.:

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} || x_j - \mu_i ||^2 = min! \qquad (2.26)$$

wobei $||x_j - \mu_i||^2$ die Euklidische Distanz zwischen einem Datenvektor x_j und dem Zentroid des Clusters C_i ist. Das k-Means Verfahren ordnet damit jeden Datenvektor dem Cluster zu, dessen Zentroid die geringste euklidische Distanz zum

Datenvektor aufweist. Die Cluster-Zentroide sind die Mittelwerte der Daten in den jeweiligen Clustern.

Im PRECLAS-Ansatz werden die unsicheren Testdaten in fünf Cluster aufgeteilt. Zur Auswahl der Kandidaten in den jeweiligen Clustern werden die Prognosewerte $\hat{y}_{\mathrm{RF}}$ und $\hat{y}_{\mathrm{NN}}$ verwendet. D. h., in jedem Cluster werden die Daten nach $\hat{y}_{\mathrm{RF}}$ + $\hat{y}_{\mathrm{NN}}$ sortiert. Im Fall von [y_{crit} , ,<'] wird derjenige als Kandidat gewählt, dessen Summe $\hat{y}_{\mathrm{RF}}$ + $\hat{y}_{\mathrm{NN}}$ am kleinsten ist, im Fall von [y_{crit} , ,<'] derjenige, dessen Summe $\hat{y}_{\mathrm{RF}}$ + $\hat{y}_{\mathrm{NN}}$ am größten ist.

Im Fall, dass der ausgewählte Kandidat seinem nächsten Nachbarn im Trainingsdatensatz zu ähnlich ist, wird dieser als Kandidat verworfen. Dieser Kandidat würde für den Trainingsdatensatz eine Redundanz darstellen, die für die Lernalgorithmen keine zusätzliche Information liefert. In diesem Fall werden die nächstkleineren bzw. größeren in der Rangfolge der sortierten Werte als Kandidaten ausgewählt.

(7) Der PRECLAS-Ansatz vollzieht seine Schätzungen in sukzessiven Schleifen (Loops), wobei in jeder Schleife eine Schätzung der Wahrscheinlichkeit des kritischen Bereiches und eine Auswahl der Kandidaten erfolgt, die dem Trainingsdatensatz hinzugefügt werden. Danach werden in der nächsten Schleife die Metamodelle auf Grundlage des erweiterten Trainingsdatensatzes modifiziert. Mit den modifizierten Metamodellen wird erneut eine Schätzung der Wahrscheinlichkeit für den kritischen Bereich und eine Auswahl von Kandidaten durchgeführt, die dem Trainingsdatensatz hinzugefügt werden. Dies wird solange fortgesetzt bis die Wahrscheinlichkeitsschätzung als hinreichend konvergent bewertet wird. Die Kriterien dazu sind unter Punkt (10) beschrieben.

(8) Für die Entscheidung, welche Prognosen als unsicher zu bewerten sind, werden die Prognosen aller Metamodelle betrachtet. Für die Schätzung der Wahrscheinlichkeit des kritischen Bereiches werden nur die ermittelten Wahrscheinlichkeiten des Random-Forest Klassifikators und Regressors verwendet. Diese beiden Verfahren wiesen in Testrechnungen bereits bei relativ geringer Zahl an Trainingsdaten bessere Prognoseergebnisse als die restlichen Metamodelle auf.

Angenommen in jeder Schleife werden N zufällig ausgespielte Testdaten durch die jeweiligen Metamodelle klassifiziert, $n_{crit,1}$ ($n_{non\text{-}crit,1}$) ist die Anzahl der mit dem Random Forest Klassifikator als kritisch (nicht-kritisch) klassifizierten Daten und $n_{crit,2}$

($n_{non\text{-}crit,2}$) ist die Anzahl der mit dem Random-Forest-Regressor als kritisch (nicht-kritisch) klassifizierten Daten. Dann erfolgt die Schätzung der Wahrscheinlichkeit für den kritischen Bereich nicht durch eine einfache Punktschätzung $\widehat{P_{crit,j}} = \frac{n_{crit,j}}{N}$ für j = 1,2, sondern über das Bayes'sche Schätzverfahren, das die Unsicherheiten der Schätzung mitberücksichtigt. Die Unsicherheit bzgl. der Wahrscheinlichkeit $P_{crit,j}$ wird dabei durch eine geeignete Wahrscheinlichkeitsverteilung ausgedrückt:

$$f(P_{crit} | n_{crit}) = \frac{f(n_{crit} | P_{crit}) \cdot f_0(P_{crit})}{\int_0^1 f(n_{crit} | P_{crit}) \cdot f_0(P_{crit})\, dP_{crit}} \quad (2.27)$$

Dabei bezeichnet $f_0(P_{crit})$ die Dichtefunktion der a-priori Verteilung von P_{crit}. Die a-priori Verteilung beschreibt die Vorinformation über P_{crit}. Im vorliegenden Fall wird angenommen, dass bzgl. P_{crit} keine Vorinformation vorliegt und deshalb eine nicht-informative a-priori Verteilung verwendet wird. $f(P_{crit} | n_{crit})$ steht für die Dichtefunktion der a-posteriori Verteilung von P_{crit}, die den Kenntnisstand von P_{crit} beschreibt, nachdem n_{crit} kritische Daten beobachtet wurden.

Das Integral im Nenner erstreckt sich über den gesamten Wertebereich von P_{crit} und dient der Normierung, damit $f(P_{crit} | n_{crit})$ zu einer Verteilungsdichte wird. Da es sich bei der zu schätzenden Größe um einen Wahrscheinlichkeitswert handelt, liegt der Wertebereich von P_{crit} im Intervall zwischen den Werten 0 und 1.

$f(n_{crit} | P_{crit})$ ist die Likelihood-Funktion der Beobachtung n_{crit} bei einem gegebenen Wert von P_{crit}. Durch die Likelihood der Beobachtung wird der a-priori Kenntnisstand aktualisiert. Im vorliegenden Fall ist die Likelihood-Funktion durch eine Binomialverteilung gegeben. Diese beschreibt, mit welcher Wahrscheinlichkeit wieviel von N Testdaten in den kritischen Bereich fallen, wenn für jede Beobachtung eine Erfolgswahrscheinlichkeit von P_{crit} gegeben ist:

$$f(ncrit | Pcrit) = \binom{N}{n_{crit}} \cdot {P_{crit}}^{n_{crit}} . (1 - P_{crit})^{N - n_{crit}} \quad (2.28)$$

Als nichtinformative a-priori Verteilung des Parameters P_{crit} einer Binomialverteilung erhält man nach dem Verfahren von Jeffreys /BOX 73/ eine Verteilung $f_0(P_{crit})$, für die gilt

$$f_o(P_{crit}) \propto {P_{crit}}^{-0.5} \cdot (1 - P_{crit})^{-0.5} \quad (2.29)$$

Werden die Gleichungen (2.28) und (2.29) in Gleichung (2.27) eingesetzt, ergibt sich nach algebraischen Umformungen und Lösung des Integrals eine Beta-Verteilung Beta(α,β) mit den Parametern $\alpha = n_{crit}+0.5$ und $\beta = N–n_{crit}+0.5$ als a-posteriori.

(9) Für jedes Metamodell wird aus den Prognoseergebnissen zu den N Testdaten eine Betaverteilung nach dem unter Punkt (8) beschriebenen Bayes'schen Verfahren erzeugt. Sei Beta(α_1, β_1) die Verteilung der Wahrscheinlichkeit des kritischen Bereiches, die aus den Prognosewerten des Random-Forest Klassifikators ermittelt wurde, und Beta(α_2, β_2) die Verteilung, die aus den Prognosewerten des Random-Forest Regressors resultiert. Dann wird die Mischverteilung aus den beiden Betaverteilungen als endgültige Verteilung der Wahrscheinlichkeit des kritischen Bereichs ermittelt, wobei jede Verteilung mit dem gleichen Gewicht von 0.5 in die Berechnung der Mischverteilung eingeht. D. h., jedes der beiden Metamodelle wird als gleich gut für die Schätzung angenommen. Falls sich in späteren Anwendungen zeigen sollte, dass ein Algorithmus bessere Prognosen liefert als der andere, können die Gewichtungen entsprechend modifiziert werden. Als Schätzergebnis der Wahrscheinlichkeit für den kritischen Bereich erhält man also die Mischverteilung $f_{mix}(P_{crit})$ in Gl. (2.30).

$$\begin{aligned} f_{mixed}(P_{crit}) &= \sum_{j=1}^{2} Beta_j(\alpha_j, \beta_j) \\ &= \sum_{j=1}^{2} w_j \cdot \frac{\Gamma(N+1)}{\Gamma(n_{crit,j}+0.5)\cdot\Gamma(N-n_{crit,j}+0.5)} \\ &\cdot {P_{crit}}^{n_{crit,j}-0.5} \cdot (1-P_{crit})^{N-n_{crit,j}-0.5} \end{aligned} \tag{2.30}$$

Dabei bezeichnet:

w_j – das Gewicht, mit dem die Beta-Verteilung des Metamodells j in die Mischverteilung eingeht, j=1,2;

$n_{crit,j}$ – die Anzahl der Testdaten, die vom Metamodell j als kritisch klassifiziert wurden;

N – die Anzahl der Testdaten;

P_{crit} – die zu schätzende Wahrscheinlichkeit des kritischen Bereiches;

Γ(x) – die Gamma-Funktion von x.

Die Mischverteilung $f_{mixed}(P_{crit})$ beschreibt den Kenntnisstand bzgl. der Wahrscheinlichkeit für den kritischen Bereich, der sich aus den beobachteten Prognosen der beiden Metamodelle bei fehlender Vorinformation ergibt.

(10) Die in jeder Schleife berechnete Mischverteilung wird durch den Mittelwert sowie die 5 %, 50 % und 95 %-Quantile der Verteilung ausgedrückt. Das sukzessive Hinzufügen von neuen Kandidaten zum Trainingsdatensatz sowie die darauf basierende erneute Schätzung der Wahrscheinlichkeit des kritischen Bereiches wird abgebrochen, wenn die Mischverteilungen in aufeinanderfolgenden Loops hinreichend konvergent sind und sich nicht mehr wesentlich voneinander unterscheiden. Zur Beurteilung, ob die ermittelten Mischverteilungen hinreichend konvergent sind, wird folgendes Kriterium verwendet:

Im PRECLAS-Ansatz werden mindestens vier Schleifen ausgeführt. Von den letzten vier durchgeführten Loops werden die ermittelten Verteilungen zur Bewertung der Konvergenz herangezogen und die jeweiligen Mittelwerte und Mediane miteinander verglichen. Die Berechnungen werden als hinreichend konvergent betrachtet, wenn folgende beiden Bedingungen erfüllt sind:

i) Der Variationskoeffizient δ der Mediane der berechneten Verteilungen der letzten 4 Loops ist < 0.25.

   Seien $x_1, \ldots, x_4$ die Mediane der letzten vier erzeugten Verteilungen. Aus diesen vier Werten wird die Standardabweichung s und der Mittelwert $\bar{x}$ berechnet. Der Variationskoeffizient δ ist gegeben durch $\delta = s \,/\, \bar{x}$.

ii) Die relative Differenz Δ_{rel} zwischen minimalem und maximalem Mittelwert der letzten vier Loops ist < 0.5.

   Seien $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4$ die Mittelwerte der letzten vier erzeugten Verteilungen. Mit $\bar{x}_{\min}$ und $\bar{x}_{\max}$ seien der Minimal- und Maximalwert der Mittelwerte $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4$ bezeichnet. Die relative Differenz des minimalen und maximalen Wertes Δ_{rel} wird berechnet durch $\Delta_{rel} = (\bar{x}_{\max} - \bar{x}_{\min}) \,/\, \bar{x}_{\min}$.

Die Iteration des PRECLAS Ansatzes wird beendet, wenn sich aus den letzten vier ermittelten Verteilungen das Abbruchkriterium δ < 0.25 und Δ_{rel}< 0.5 ergibt. In diesem Fall werden die aus den Prognosewerten der Metamodelle geschätzten Verteilungen als hinreichend konvergent betrachtet. Die Anzahl der zu berücksichtigenden Schleifen der Iteration wurde auf vier gesetzt, da Testrechnungen gezeigt

haben, dass dieser Wert eine vernünftige Abwägung zwischen der Konvergenz der Schätzung und dem Rechenaufwand darstellt.

2.2.4 Anwendungsbeispiele

Für das Testen des GASA-PRECLAS Ansatzes wurden drei verschiedene nicht-lineare multivariate Testfunktionen unterschiedlicher Komplexität verwendet. Anhand dieser Testfunktionen konnte gut demonstriert werden, dass der entwickelte GASA-PRECLAS Algorithmus die Zielsetzung erfüllt, mit möglichst wenig Rechenaufwand eine gute Schätzung bzgl. der Wahrscheinlichkeit des kritischen Parameterbereichs zu liefern. Der Algorithmus zeigte sich insbesondere in denjenigen Fällen als äußerst effizient, wenn kritische Funktionswerte nur mit einer sehr kleinen Wahrscheinlichkeit auftreten.

In den folgenden Abschnitten werden die drei Testfunktionen und die zugehörigen Analysen beschrieben.

2.2.4.1 Testfunktion F_1

$$F_1: \quad y = \left(\frac{X_1}{X_2}\right)^2 - \frac{(X_3 \cdot X_2)}{X_1} \tag{2.31}$$

Die Parameter X_1, X_2 and X3 sind verteilt als

X_1 ~ Normal (200 , 30)
X_2 ~ Normal (150 , 50)
X_3 ~ Uniform (1 , 5)

Für die Funktion F_1 wird der kritische Bereich durch die Parameterkombinationen definiert, deren Funktionswerte ≤ -10 sind. Um die Qualität des entwickelten Verfahrens beurteilen zu können, wurde zunächst aufgrund einer Monte-Carlo-Simulation (MCS) mit 1000000 Testdaten (Parameterkombinationen und F1-Ergebnisse) die Wahrscheinlichkeit des kritischen Bereichs abgeschätzt. Dabei ergab sich ein Wert von P_{MCS} = 5.6E-05. Dieser Wert dient als Vergleichswert für das Schätzergebnis über den GASA-PRECLAS Algorithmus.

Im ersten Schritt wurde eine Stichprobe von 20 zufällig gezogenen Eingabekombinationen der drei Parameter erstellt und die entsprechenden Funktionswerte durch F_1 ermittelt. In dieser Stichprobe ergab sich ein minimaler Wert von -3.95. D. h., nach der

Zufallsstichprobe vom Umfang 20 enthält die Trainingsstichprobe 0 Beobachtungen im kritischen und 20 Beobachtungen im nicht-kritischen Bereich. Dies Ergebnis wird kurz mit [0, 20] bezeichnet.

Da für die Anwendung des PRECLAS-Algorithmus ein Trainingsdatensatz benötigt wird der mindestens fünf kritische Beobachtungen enthält, wurde der GASA-Algorithmus angewendet, um mit möglichst wenigen Stichproben Parameterkombinationen zu finden, die kritische Funktionswerte ≤ -10 liefern und somit im kritischen Bereich liegen. Die Anwendung des GASA-Algorithmus ergab dabei folgende Ergebnisse:

In der 1. Generation von Nachkommen wurden acht Kandidaten für den Trainingsdatensatz ausgewählt, für die folgende Funktionswerte berechnet wurden:

$[y_1, \ldots, y_8] = [-2.85, -1.09, -2.88, -4.92, -3.32, -4.05, -2.57, -2.79]$

Diese Funktionswerte wurden mit den zugehörigen Parameterkombinationen dem Trainingsdatensatz hinzugefügt. Nach der 1. Generation enthielt der Trainingsdatensatz 28 Beobachtungen, deren Verteilung auf die Klassen [kritisch, nicht-kritisch] durch [0, 28] gegeben ist. D. h., 0 Beobachtungen lagen in der kritischen Klasse, 28 Beobachtungen in der nicht-kritischen Klasse. In Tab. 2.4 sind die Ergebnisse der aufeinanderfolgenden Generationen des GASA-Algorithmus aufgeführt.

Tab. 2.4 Ergebnisse der Testfunktion F_1 und Verteilung der Parameterkandidaten auf die kritische und nicht-kritische Klasse in aufeinanderfolgenden Generationen des GASA-Algorithmus

Generation	Anzahl Beobachtungen		F_1-Ergebnisse bzgl. der Parameterkandidaten
	$F_1 \leq -10$	$F_1 > -10$	
0	0	20	Anfänglicher Trainingsdatensatz nach zufälliger Stichprobe
1	0	28	[-2.8, -1.1, -2.9, -4.9, -3.3, -4.1, -2.6, -2.8]
2	0	36	[-5.1, -2.8, -3.6, -4.5, -6.2, -6.3, -2.7, -4.9]
3	0	44	[-4.4, -5.2, -6.10, -7.9, -3.8, -4.6, -4.5, -4.6]
4	0	52	[-8.0, -8.8, -7.8, -8.6, -5.9, -6.0, -3.6, -7.6]
5	0	60	[-9.1, -8.6, -9.1, -7.9, -4.5, -5.0, -6.2, -6.3]
6	2	66	[-9.4, -9.1, -8.6, -9.2, -2.6, -4.1, -10.3, -10.6]
7	7	69	[-13.6, -9.6, -10.5, -9.4, -12.6, -10.9, -13.1, -6.3]
8	13	71	[-12., -13.4, -13.1, -12.5, -13.9, -12.2, -7.4, -7.2]

Aus Tab. 2.4 wird deutlich, wie sich die Funktionswerte der ausgewählten Kandidaten in den nachfolgenden Generationen zunehmend dem kritischen Bereich ≤ -10 annäherten. In der 6. Generation traten erstmals zwei Kandidaten auf, deren Parameterkombinationen zu kritischen Funktionswerten ≤ -10 geführt haben. In der 7. Generation wurden fünf weitere Kandidaten gefunden, deren Parameterkombinationen zu kritischen Funktionswerten geführt haben. Damit wurde das Kriterium von mindestens fünf kritischen Beobachtungen erreicht. Dass das Kriterium erfüllt ist, wird erst in der nächsten Generation erkannt. Deshalb wurde noch die Generation 8 erzeugt, in denen weitere sechs kritische Beobachtungen dem Trainingsdatensatz hinzugefügt wurden. D. h., durch die Anwendung des GASA-Algorithmus konnten bereits nach 84 Funktionsberechnungen 13 kritische Beobachtungen ermittelt werden.

Um die Relation zu verdeutlichen, welcher Vorteil mit dem GASA-Ansatz erzielt werden kann, soll folgender Vergleich dienen. Wenn mit einer Sicherheit von 90 % mindestens eine Parameterkombination gefunden werden soll, die zu einem kritischen Wert führt, der nur mit einer Eintrittswahrscheinlichkeit von 5.6E-5 vorkommt, würde man bei reiner Zufallsauswahl eine Stichprobe von mindestens 41250 Parameterkombinationen mit zugehörigen Funktionsberechnungen benötigen. Entsprechend mehr würde man benötigen, um mindestens fünf kritische Beobachtungen in der Stichprobe zu erhalten.

Um eine Vorstellung zu erhalten, in welchen Bereichen die standardisierten Parameterwerte (= Verteilungsfunktionswerte der Parameter = q-Werte, siehe Gl. (2.24) liegen, die zu kritischen bzw. nicht-kritischen Funktionswerten geführt haben, sind die entsprechenden Scatter Plots in Abb. 2.9 bis Abb. 2.11 dargestellt.

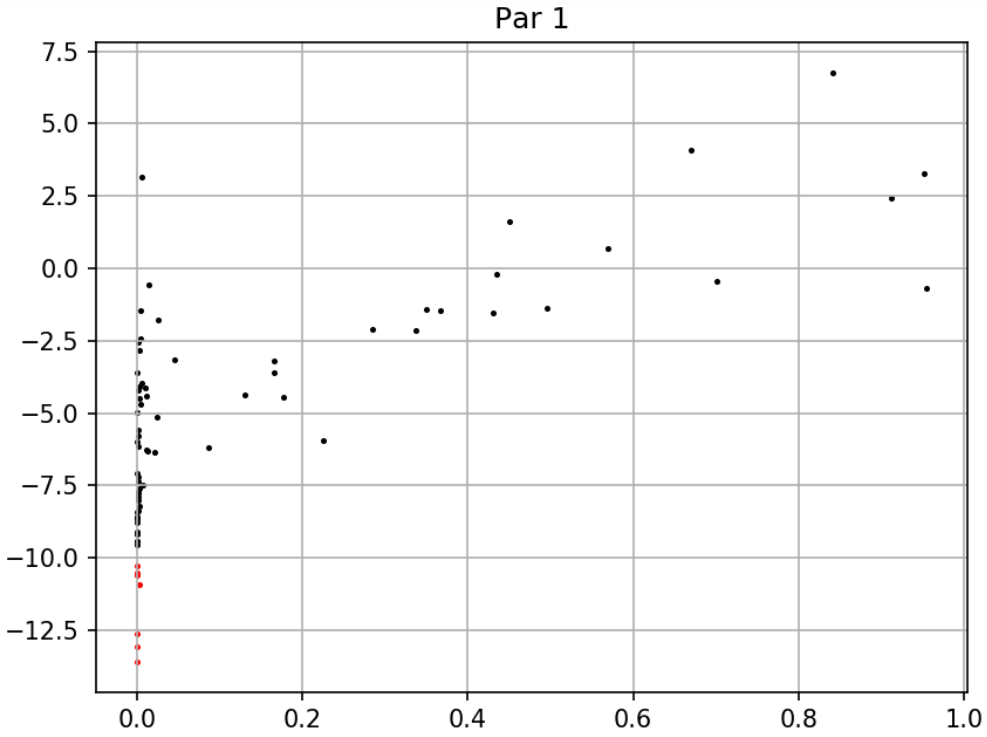

Abb. 2.9 Standardisierte Werte von Parameter 1 in der kritischen (rot) und nicht-kritischen (schwarz) Klasse

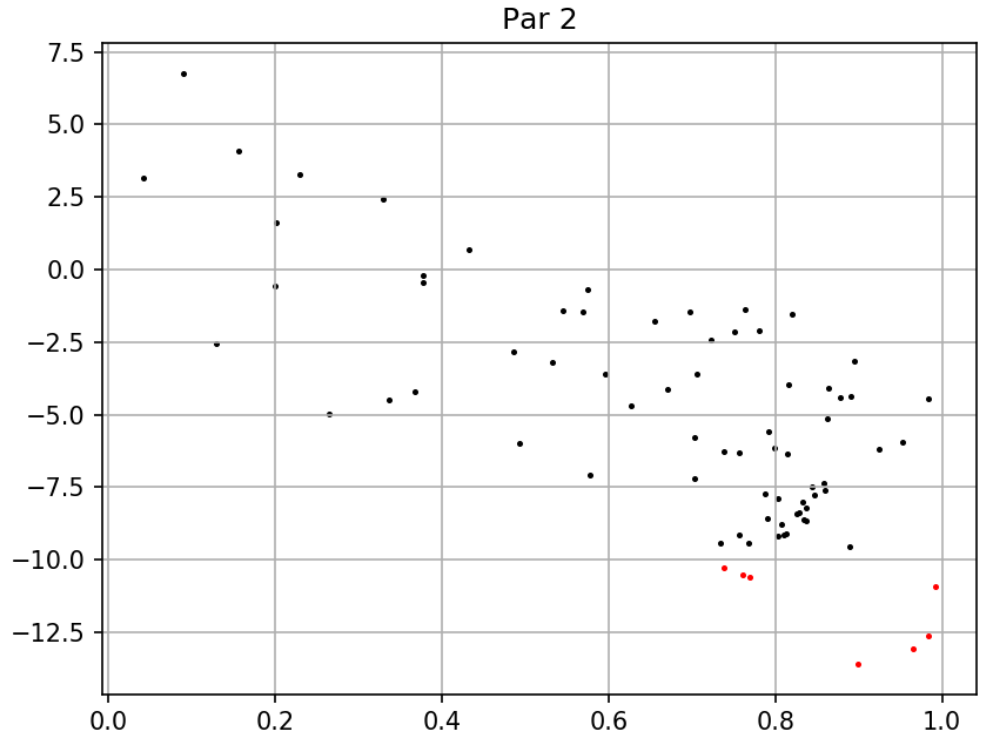

Abb. 2.10 Standardisierte Werte von Parameter 2 in der kritischen (rot) und nicht-kritischen (schwarz) Klasse

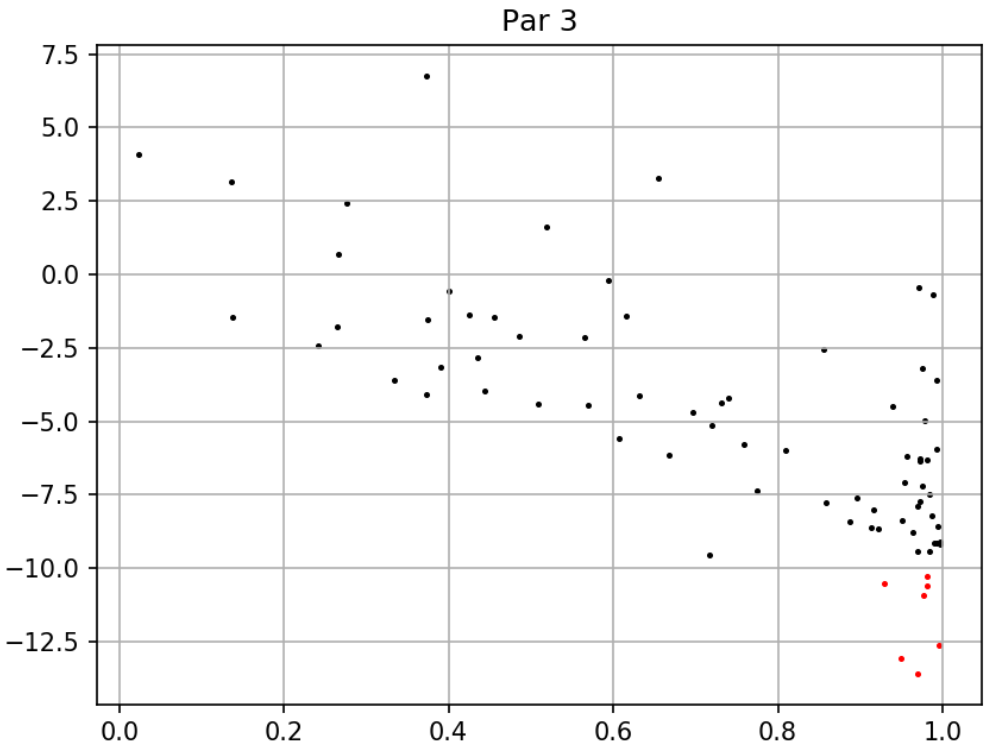

Abb. 2.11 Standardisierte Werte von Parameter 3 in der kritischen (rot) und nicht-kritischen (schwarz) Klasse

Als die Mindestkriterien für den anfänglichen Trainingsdatensatz erfüllt waren, wurde der GASA-Algorithmus beendet, und der PRECLAS-Ansatz für die Wahrscheinlichkeitsschätzung des kritischen Bereiches und die weitere Auswahl von Kandidaten für den Trainingsdatensatz gestartet.

Der PRECLAS-Algorithmus wurde in sukzessiven Schleifen durchgeführt, wobei in jeder Schleife 200000 zufällig gezogene Testdaten durch die Metamodelle klassifiziert wurden und aus den Prognosen die entsprechende Verteilung der Wahrscheinlichkeit geschätzt wurde. Zusätzlich wurden in jeder Schleife die neuen Parameterkandidaten bestimmt, von denen der tatsächliche Funktionswert berechnet wurde und die dem Trainingsdatensatz als neue Beobachtungen hinzugefügt wurden. Falls der Stichprobenumfang des Testdatensatzes nicht ausreichen sollte, wird dies angezeigt und der Stichprobenumfang kann entsprechend (z. B. um den Faktor 2 oder 3) erhöht werden. Alternativ kann auch das Importance Sampling Verfahren, das in Abschnitt 2.2.4.3 beschrieben wird, angewendet werden. Für den kritischen Bereich der Testfunktion F_1 war der gewählte Stichprobenumfang von 200000 pro Loop groß genug.

Die Ergebnisse der Wahrscheinlichkeitsschätzungen der einzelnen Loops sind in Tab. 2.5 angegeben. Die geschätzten Verteilungen aus den einzelnen Berechnungsschleifen werden durch ihre jeweiligen Mittelwerte sowie ihre 5 %-, 50 %- und 95 %-Quantile beschrieben.

Der erste Trainingsdatensatz, an dem der PRECLAS-Algorithmus angewendet wurde, ist derjenige, der über den GASA-Algorithmus erzeugt worden ist. In diesem Trainingsdatensatz befanden sich 13 kritische und 71 nicht-kritische Beobachtungen, was durch die Bezeichnung [13, 71] in Loop 0 angegeben wird. Die Klassifikation der Testdaten durch die Metamodelle, die auf der Basis dieses Trainingsdatensatzes konstruiert wurden, lieferte eine mittlere Wahrscheinlichkeit von 9.25E-4. 5 %- und 95 %-Quantil der geschätzten Verteilung sind durch 4.44E-4 und 1.46E-3 gegeben. In Loop 0 wurden 10 neue Kandidaten ausgewählt, die dem Trainingsdatensatz hinzugefügt wurden.

Auf der Basis dieses erweiterten Trainingsdatensatzes wurden in Loop 1 die Metamodelle angepasst und anhand von 200000 neuen zufällig ausgespielten Testdaten erfolgte eine weitere Wahrscheinlichkeitsschätzung. Von den 10 Kandidaten, die in Loop 0 ausgewählt wurden, wiesen alle nicht-kritische Funktionswerte auf. Dies kann der Angabe [13, 81] entnommen werden. D. h., im Trainingsdatensatz von Loop 1 befanden sich 13 kritische und 81 nicht-kritische Beobachtungen. Die geschätzte Verteilung in Loop 1 hat eine mittlere Wahrscheinlichkeit von 2.81E-4. 5 % und 95 %-Quantile der Verteilung sind durch 7.19E-5 und 5.99E-4 gegeben. Analog sind die Schätzungen in den nachfolgenden Loops zu interpretieren.

Tab. 2.5 Geschätzte Verteilungen der Eintrittswahrscheinlichkeit kritischer Werte der Testfunktion F_1 in aufeinanderfolgenden Loops (basierend auf 200000 zufällig ausgewählten Testdaten pro Loop)

Loop	Beobachtungen		Mittelwert	5 %	50 %	95 %	δ	Δ_{rel}
	$F_1 \leq -10$	$F_1 > -10$						
0	13	71	9.25E-4	4.44E-4	8.57E-4	1.46E-3		
1	13	81	2.81E-4	7.19E-5	2.48E-4	5.99E-4		
2	13	91	1.34E-4	1.29E-5	1.02E-4	3.61E-4		
3	13	101	1.64E-4	2.28E-5	1.37E-4	4.15E-4		
4	13	111	3.13E-4	5.69E-5	2.74E-5	7,05E-4	0.38	1.34
5	13	120	1.5E-4	1.41E-5	1.16E-4	3.99E-4	0.44	1.34
6	13	130	1.54E-4	1.53E-5	1.21E-4	4.08E-4	0.4	1.09
7	15	138	1.31E-4	1.17E-5	9.9E-5	3.58E-4	0.46	1.34
8	15	148	6.46E-5	5.4E-6	4.82E-5	1.8E-4	0.3	1.38
9	15	158	1.23E-4	2.76E-5	1.07E-4	2.73E-4	0.29	1.38

Loop	Beobachtungen		Mittelwert	5 %	50 %	95 %	δ	Δ_{rel}
	$F_1 \leq -10$	$F_1 > -10$						
10	15	167	1.11E-4	1.7E-5	9.36E-5	2.65E-4	0.26	1.03
11	15	177	7.26E-5	8.12E-6	5.66E-5	1.92E-4	0.32	0.9
12	15	185	9.55E-5	1.58E-5	7.98E-5	2.29E-4	0.22	0.69
13	16	193	8.46E-5	1.22E-5	6.83E-5	2.12E-4	0.18	0.53
14	17	201	6.8E-5	6.26E-6	5.14E-5	1.86E-4	0.17	0.46

Wie in Abschnitt 2.2.3 unter Punkt (10) beschrieben wurde, dient δ < 0.25 und Δ_{rel}< 0.5 als Abbruchkriterium, wann der PRECLAS-Algorithmus beendet werden und die Wahrscheinlichkeitsschätzungen als hinreichend konvergent betrachtet werden kann. Dazu werden δ und Δ_{rel} anhand der Median- und Mittelwerte der letzten vier berechneten Verteilungen ermittelt. Ab Loop 4 wird mit der Berechnung von δ und Δ_{rel} begonnen. D. h., dass im PRECLAS-Ansatz mindestens vier Berechnungsschleifen durchgeführt werden müssen, um das Abbruchkriterium erfüllen zu können.

Das Abbruchkriterium wurde in Loop 14 erreicht. D. h., die in Loop 14 geschätzte Verteilung wird als hinreichend konvergent betrachtet und wird als Wahrscheinlichkeitsschätzung für den kritischen Bereich verwendet. Die Schätzung weist dabei eine mittlere Wahrscheinlichkeit von 6.8E-5 auf, dass bei der Testfunktion F1 Werte auftreten die kleiner als -10 sind. 5 %- und 95 %-Quantil der Verteilung liegen bei 6.26E-6 und 1.86E-4. Diese Schätzung wurde mit insgesamt 218 Berechnungen durch die Funktion F1 erlangt.

Der Mittelwert von 6.8E-5 für die Wahrscheinlichkeit des kritischen Bereichs hat die gleiche Größenordnung wie der Schätzer P_{MCS} = 5.6E-05 aus der MCS mit 1000000 Berechnungen. Um einen Schätzer (klassische obere 95 %-Konfidenzgrenze) in der Größenordnung von P_{MCS} zu erhalten, müssten mindestens 54000 Berechnungen im Rahmen einer einfachen MCS durchgeführt werden und bei keiner Rechnung dürfte ein kritisches Ergebnis berechnet werden. Wenn genau ein kritisches Ergebnis berechnet wird, muss der Umfang der MCS erhöht werden. Es müssten dann mindestens 85000 Berechnungen durchgeführt werden. Dies verdeutlicht die Effizienz des GASA-PRECLAS Ansatzes.

2.2.4.2 Testfunktion F_2: Ishigami-Funktion

$$F_2: \quad y = sin(X_1) + c_1 \cdot sin^2(X_2) + c_2 \cdot X_3^4 \cdot sin(X_1) \tag{2.32}$$

Jeder Parameter X_j, ist gleichverteilt mit $X_j \sim$ Uniform ($-\pi$, π) , j = 1, 2, 3. Die Konstanten sind gegeben durch $c_1 = 7$ und $c_2 = 0.1$.

Für die Funktion F_2 wird der kritische Bereich durch die Parameterkombinationen definiert, deren Funktionswerte ≥ 15 sind. Anhand einer MCS von 1000000 zufälligen Parameterkombinationen wird die Wahrscheinlichkeit des kritischen Bereichs mit P_{MCS} = 2.94E-3 abgeschätzt. Dieser Wert dient als Vergleichswert für das Schätzergebnis über den GASA-PRECLAS Algorithmus.

Zu Beginn wird eine Stichprobe von 20 zufällig gezogenen Eingabekombinationen der drei Parameter erstellt und die entsprechenden Funktionswerte durch F_2 ermittelt. In dieser Stichprobe ergibt sich ein maximaler Funktionswert von 12.73. Nach der Zufallsstichprobe vom Umfang 20 enthält der Trainingsdatensatz 0 Beobachtungen im kritischen und 20 Beobachtungen im nicht-kritischen Bereich.

Anschließend wird der GASA-Algorithmus angewendet, um mit möglichst wenigen Funktionsberechnungen Parameterkombinationen zu finden, die Funktionswerte ≥ 15 liefern und somit im kritischen Bereich liegen. Die Ergebnisse des GASA-Algorithmus sind in Tab. 2.6 aufgeführt.

Tab. 2.6 Ergebnisse der Testfunktion F_2 und Verteilung der Parameterkandidaten auf die kritische und nicht-kritische Klasse in aufeinanderfolgenden Generationen des GASA-Algorithmus

Generation	Anzahl Beobachtungen		F_2-Ergebnisse bzgl. der Parameterkandidaten
	$F_2 \geq 15$	$F_2 < 15$	
0	0	20	Anfänglicher Trainingsdatensatz nach zufälliger Stichprobe
1	0	28	[-3.9, 13.0, -2.8, 5.0, 3.9, 6.5, 5.3, 7.7]
2	3	33	[8.6, 17.1, 16.2, 15.9, 14.3, 12.4, 6.2, 13.0]
3	5	39	[12.1, 16.7, 14.0, 16.7, -4.1, -4.4, 13.3, 0.3]
4	10	42	[17.2, 15.7, 16.1, 17.5, 10.2, 3.0, 8.6, 15.2]

Die ersten kritischen Werte, die über den GASA-Ansatz gefunden wurden, sind in Generation 2 aufgetreten. Von den acht Kandidaten ergaben die Funktionsberechnungen drei kritische und fünf nichtkritische Funktionswerte, die dem Trainingsdatensatz

hinzugefügt wurden. Nach der 2. Generation enthielt der Trainingsdatensatz somit drei kritische und 33 nicht-kritische Beobachtungen.

Nach drei Generationen wurde die Bedingung von mindestens fünf kritischen Werten im Trainingsdatensatz erfüllt. Dies wurde in Generation 4 erkannt und der GASA-Algorithmus wurde nach der 4. Generation beendet. In Generation 4 wurden noch zusätzlich fünf kritische und drei nicht-kritische Beobachtungen dem Trainingsdatensatz hinzugefügt.

Nach Beendigung des GASA-Algorithmus waren im Trainingsdatensatz 52 Beobachtungen, von denen 10 kritische und 42 nicht-kritische Werte aufwiesen. Auf der Basis des Trainingsdatensatzes wurden die Berechnungsschleifen des PRECLAS-Algorithmus gestartet. Die Ergebnisse des PRECLAS-Algorithmus sind in Tab. 2.7 aufgeführt. Die Verteilungsschätzungen wurden anhand von 100000 Testdaten in den jeweiligen Loops durchgeführt.

Tab. 2.7 Geschätzte Verteilungen der Eintrittswahrscheinlichkeit kritischer Werte der Testfunktion F_2 in aufeinanderfolgenden Loops (basierend auf 100000 zufällig ausgewählten Testdaten pro Loop)

Loop	Beobachtungen		Mittelwert	5 %	50 %	95 %	δ	Δ_{rel}
	$F_2 \geq 15$	$F_2 < 15$						
0	10	42	2.6E-3	1.24E-3	2.52E-3	4.24E-3		
1	12	50	3.04E-3	1.88E-3	2.98E-3	4.43E-3		
2	14	58	2.32E-3	1.02E-3	2.23E-3	3.94E-3		
3	16	66	4.1E-3	1.47E-3	3.85E-3	7.27E-3		
4	16	76	2.68E-3	1.24E-3	2.57E-3	4.49E-3	0.208	0.77
5	20	82	3.29E-3	1.95E-3	3.21E-3	4.87E-3	0.209	0.77
6	24	88	4.92E-3	3.18E-4	4.84E-3	6.93E-3	0.232	0.84
7	25	97	3.02E-3	1.73E-3	2.94E-3	4.55E-3	0.256	0.84
8	28	104	2.79E-3	1.64E-3	2.72E-3	4.19E-3	0.243	0.76
9	31	111	3.31E-3	1.43E-3	3.17E-3	5.64E-3	0.245	0.76
10	35	117	2.8E-3	1.5E-3	2.72E-3	4.37E-3	0.065	0.19

Der erste Trainingsdatensatz, an dem der PRECLAS-Algorithmus angewendet wurde, ist derjenige, der über den GASA-Algorithmus erzeugt wurde. Die Verteilung der Beobachtungen auf die kritische und nicht-kritische Klasse ist in Loop 0 angegeben. Die Klassifikation der Testdaten durch die Metamodelle, die auf der Basis dieses Trainingsdatensatzes erzeugt wurden, lieferte eine mittlere Wahrscheinlichkeit von 2.6E-3.

Zusätzlich wurden in Loop 0 10 Kandidaten ausgewählt, die dem Trainingsdatensatz hinzugefügt wurden.

Auf der Basis des erweiterten Trainingsdatensatzes wurden in Loop 1 die Metamodelle angepasst. Mit den Prognosen der Metamodelle erfolgte dann anhand von 100000 neuen zufällig ausgespielten Testdaten eine weitere Wahrscheinlichkeitsschätzung. Von den 10 Kandidaten, die in Loop 0 ausgewählt wurden, wiesen alle nicht-kritische Funktionswerte auf. Dies kann der Angabe [12, 50] bzgl. Loop 1 entnommen werden. D. h., im Trainingsdatensatz von Loop 1 befanden sich 12 kritische und 50 nicht-kritische Beobachtungen. Die geschätzte Verteilung in Loop 1 hat eine mittlere Wahrscheinlichkeit von 3.04E-3. 5 % und 95 %-Quantile der Verteilung sind durch 1.88E-3 und 4.43E-3 gegeben. Analog sind die Schätzungen in den nachfolgenden Loops zu interpretieren.

Das Abbruchkriterium δ < 0.25 und Δ_{rel} < 0.5 des PRECLAS-Algorithmus wurde in Loop 10 erreicht. D. h., die in Loop 10 geschätzte Verteilung wurde als hinreichend konvergent betrachtet und wird als Wahrscheinlichkeitsschätzung für den kritischen Bereich verwendet. Die Verteilung weist eine mittlere Wahrscheinlichkeit von 2.8E-3 auf, dass bzgl. der Testfunktion F_2 Werte auftreten, die ≥ 15 sind. 5 %- und 95 %-Quantil der Verteilung liegen bei 1.5E-3 und 4.37E-3. Diese Schätzung wurde anhand von 152 Beobachtungen erlangt.

Die über den GASA-PRECLAS Algorithmus geschätzte mittlere Wahrscheinlichkeit von $P_{PRECLAS}$ = 2.8E-3 stimmt gut mit dem Wert von P_{MCS} = 2.93E-3 überein, der über eine einfache MCS mit 1000000 Berechnungen ermittelt wurde. Um einen Schätzer (klassische obere 95 %-Konfidenzgrenze) in der Größenordnung von P_{MCS} zu erhalten, müssten mindestens 1000 Berechnungen im Rahmen einer einfachen MCS durchgeführt werden und bei keiner Rechnung dürfte ein kritisches Ergebnis berechnet werden.

2.2.4.3 Testfunktion F_3: Biologisches Dosis-Modell

Testfunktion F_3 ist durch Gl. (2.23) gegeben. Die Verteilungen der Eingangsparameter dieses Modells sind in Tab. 2.2 festgehalten.

Für die Funktion F_3 wird der kritische Bereich durch die Menge aller Beobachtungen definiert, deren Parameterkombinationen Funktionswerte ≥ 2.5E-4 ergeben. Anhand einer MCS von 10 Millionen zufälligen Parameterkombinationen wird die mittlere Wahrscheinlichkeit des kritischen Bereichs mit P_{MCS} = 4.E-6 abgeschätzt. 5 %- und 95 %-

Quantil der Schätzung liegen bei 1.7E-6 und 8.5E-6. Diese Schätzungen aus der MCS dienen als Vergleichswerte für das Schätzergebnis, das über den GASA-PRECLAS Algorithmus erzielt wird. Der Vergleich mit den Ergebnissen aus der MCS erlaubt zugleich eine Beurteilung über die Qualität der entwickelten Methode.

Im ersten Schritt wurde eine Stichprobe von 20 zufällig gezogenen Eingabekombinationen der drei Parameter erstellt und die entsprechenden Funktionswerte durch F_3 ermittelt. Danach wurde der GASA-Algorithmus angewendet, der nach 10 Generationen abgebrochen werden konnte, da die erforderliche Anzahl von mindestens fünf kritischen Werten im Trainingsdatensatz erreicht war. Der Trainingsdatensatz, der durch den GASA-Algorithmus erzeugt wurde, bestand aus 100 Beobachtungen, von denen neun kritisch und 91 nicht-kritisch waren.

Die Wahrscheinlichkeitsschätzung über den PRECLAS-Algorithmus in Loop 0 wurde zunächst anhand von 100000 zufällig ausgespielten Testdaten und deren Klassifizierung durch die Metamodelle ermittelt. Dabei hat sich gezeigt, dass der Stichprobenumfang von 100000 zu klein war, um Parameterkombinationen zu erhalten, die von den Metamodellen als kritisch klassifiziert wurden. Ebenso war es mit diesem Stichprobenumfang nicht möglich eine ausreichende Anzahl möglicher Kandidaten für den nachfolgenden Loop zu erhalten. Diese Situation wird durch das Programm angezeigt und weist darauf hin, dass der Stichprobenumfang erhöht werden muss. Aber auch eine Erhöhung des Stichprobenumfanges auf 500000 hat sich für die Anwendung des PRECLAS-Ansatzes als zu klein herausgestellt.

Um den Stichprobenumfang bei der Anwendung des PRECLAS-Algorithmus zur Schätzung sehr kleiner Wahrscheinlichkeiten nicht übermäßig groß werden zu lassen, wurde ein Importance-Sampling-Verfahren angewendet, bei dem die standardisierten Parameterwerte (q-Werte, siehe Gl. 2.24) aus einem eingeschränkten Wertebereich ausgespielt wurden.

In der Abb. 2.12 sind die Verteilungen der kritischen und nicht kritischen Beobachtungen bzgl. der q-Werte der einzelnen Parameter dargestellt, die sich nach der Anwendung des GASA-Algorithmus ergeben haben. Die nicht kritischen Beobachtungen sind durch die schwarzen, die kritischen durch die roten Punkte gekennzeichnet. Abb. 2.12 weist beispielsweise darauf hin, dass sich kritische Funktionswerte ergeben, wenn die q-Werte von Parameter 6 sehr klein sind und die für Parameter 1, 2, 3 und 5 im oberen Wertebereich liegen.

Anhand der Informationen in Abb. 2.12 wurde der Wertebereich der q-Werte der einzelnen Parameter für das Sampling Verfahren so eingeschränkt, dass für die Parameter 1, 2, 3 und 5 die q-Werte aus dem Intervall [0.5, 1], für Parameter 4 aus dem Intervall [0, 1] und für Parameter 6 aus dem Intervall [0., 0.05] zufällig ausgespielt wurden. Dies sind die für die Klassifikation relevanten Bereiche der q-Werte. Für die q-Werte der Parameter, die nicht aus den jeweiligen Bereichen stammen, wurde angenommen, dass sie von den Metamodellen eindeutig als nicht-kritisch klassifiziert werden. Dies konnte anhand verschiedener Testrechnungen bestätigt werden. Bei der Einschränkung der Bereiche sollte darauf geachtet, dass gewisse Unsicherheitsmargen berücksichtigt werden. Z. B. erkennt man bzgl. Parameter 6 relativ gut eine funktionale Beziehung, die es erlaubt, den Wertebereich einigermaßen genau auf das Intervall [0., 0.05] einzuschränken. Bis auf Parameter 4 war bzgl. der anderen Parameter zumindest eine gewisse Tendenz erkennbar. Um mögliche Unsicherheiten zu berücksichtigen, erfolgte dort eine gröbere Einschränkung des Wertebereichs auf das Intervall [0.5, 1]. Bzgl. Parameter 4, bei dem ein solche Tendenz nicht erkennbar war, erfolgte keine Einschränkung des Wertebereichs.

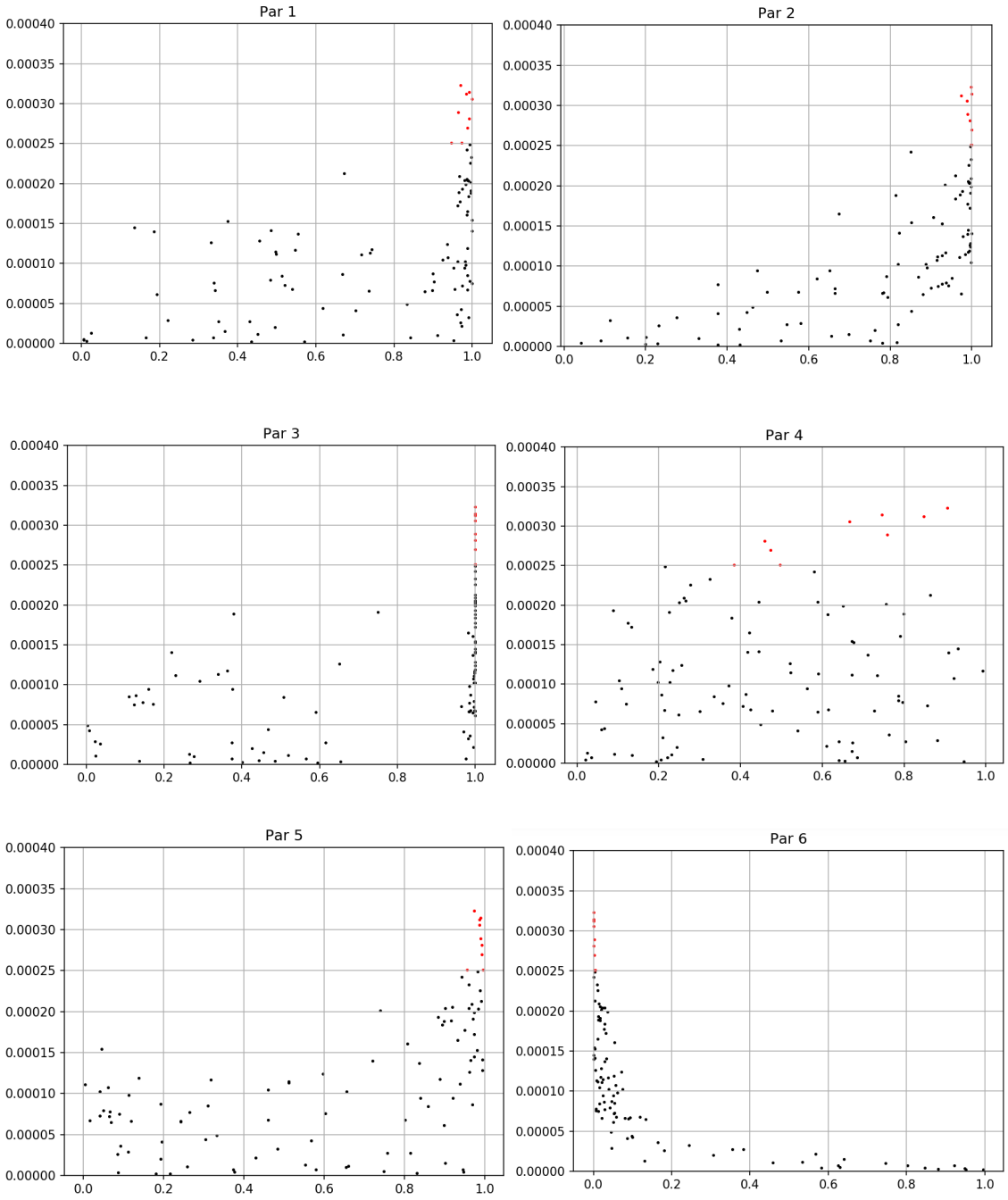

Abb. 2.12 Kritische und nicht-kritische Datenpunkte im Trainingsdatensatz nach Anwendung des GASA-Algorithmus unter Berücksichtigung der standardisierten Werte (q-Werte) der sechs Parameter der Funktion F_3

Durch die Einschränkung der Wertebereiche musste die Wahrscheinlichkeit ermittelt werden, mit der die q-Werte aus diesen Bereichen stammen. In diesem Fall berechnet sich diese Wahrscheinlichkeit zu $p_{lmp} = 0.5^4 \cdot 1 \cdot 0.05 = 3.125E\text{-}3$. p_{lmp} wird als Korrekturfaktor für die Wahrscheinlichkeitsschätzung aus dem PRECLAS-Ansatz verwendet. Der Grund dafür ist, dass sich die Wahrscheinlichkeitsschätzung aufgrund des

Importance-Sampling-Verfahrens auf die bedingte Wahrscheinlichkeit für den kritischen Bereich bezieht, wobei die Bedingung dadurch gegeben ist, dass die Parameter aus ihrem eingeschränkten Parameterbereich stammen.

Die Ergebnisse des PRECLAS-Algorithmus bzgl. der Funktion F_3 sind in Tab. 2.8 aufgeführt. Die Verteilungsschätzungen wurden anhand von 10000 Testdaten durchgeführt, die mittels eines Importance-Sampling Verfahrens in den jeweiligen Loops ausgewählt wurden. Da die zuerst berechneten Wahrscheinlichkeitswerte für den kritischen Bereich aufgrund des Importance-Sampling-Verfahrens bedingte Wahrscheinlichkeitswerte waren (Bedingung war, dass die Parameter aus eingeschränkten Wertebereichen stammen), wurden sie nachträglich mit dem Korrekturfaktor p_{Imp} = 3.125E-3 multipliziert. In Tab. 2.8 sind die unbedingten Wahrscheinlichkeitswerte enthalten.

Tab. 2.8 Geschätzte unbedingte Verteilungen der Eintrittswahrscheinlichkeit kritischer Werte der Testfunktion F_3 in aufeinanderfolgenden Loops (basierend auf jeweils 10000 Testdaten aus einem Importance-Sampling-Verfahren)

Loop	Beobachtungen		Mittel-wert	5 %	50 %	95 %	δ	Δ_{rel}
	$F_3 \geq$ 2.5E-4	$F_3 <$ 2.5E-4						
0	9	91	3.7E-6	2.62E-7	2.98E-6	8.04E-6		
1	9	101	2.25E-6	1.0E-9	1.74E-6	5.42E-6		
2	9	111	2.75E-6	3.53E-8	2.01E-6	6.38E-6		
3	9	121	4.18E-6	1.23E-6	3.8E-6	7.77E-6		
4	9	131	2.82E-6	1.0E-7	2.1E-6	6.39E-6	0.337	0.86
5	10	140	2.56E-6	1.72E-7	2.14E-6	5.7E-6	0.296	0.63
6	11	149	1.03E-5	8.46E-6	1.03E-5	1.23E-5	0.735	3.02
7	12	158	1.75E-5	8.53E-6	1.68E-5	2.73E-5	0.786	5.84
8	12	168	8.95E-6	3.4E-6	8.39E-6	1.53E-5	0.556	5.84
9	12	178	8.0E-6	2.87E-6	7.5E-6	1.39E-5	0.338	1.03
10	12	188	3.95E-6	6.34E-7	3.27E-6	8.04E-6	0.546	3.43
11	13	197	5.87E-6	1.34E-6	5.58E-6	1.12E-5	0.318	1.27
12	14	206	4.62E-6	1.33E-6	4.27E-6	8.52E-6	0.307	1.03
13	16	214	5.19E-6	3.52E-6	5.12E-6	7.05E-6	0.193	0.486

Das Abbruchkriterium δ < 0.25 und Δ_{rel} < 0.5 des PRECLAS-Algorithmus wurde in Loop 13 erreicht. Damit wird die in Loop 13 geschätzte Verteilung als hinreichend konvergent betrachtet und als Wahrscheinlichkeitsschätzung für den kritischen Bereich verwendet. Die Schätzung weist eine mittlere Wahrscheinlichkeit von 5.19E-6 auf, dass bzgl. der

Testfunktion F_3 Werte auftreten, die ≥ 2.5E-4 sind. 5 %- und 95 %-Quantil der Verteilung liegen bei 3.52E-6 und 7.05E-6. Diese Schätzung wurde mit insgesamt 230 Berechnungen erreicht.

Die über den PRECLAS-Algorithmus geschätzte mittlere Wahrscheinlichkeit von $P_{PRECLAS}$ = 5.19E-6 stimmt relativ gut mit dem Wert von P_{MCS}= 4.0E-6 überein, der über eine einfache MCS mit 10 Millionen Samples ermittelt wurde. Die Schätzung $P_{PRECLAS}$ = 5.19E-6 liegt im 90 % Bereich (1.7E-6, 8,5E-6) der Verteilung, die über MCS ermittelt wurde.

3 Adaptive Monte-Carlo-Simulation zur Lokalisierung von Parameterbereichen mit Cliff-Edge-Effekten

Die Sicherheit komplexer technischer Systeme kann durch die Untersuchung der Systemdynamik in Bezug auf Sicherheitskriterien, die Übergänge zu kritischen Systemzuständen anzeigen, bewertet werden. Die Sicherheitsanalyse für ein Kernkraftwerk (KKW) zielt darauf ab, nachzuweisen, dass die Akzeptanzkriterien erfüllt werden, die die Integrität der Barrieren gegen die Freisetzung von Radioaktivität gewährleisten. Die Einhaltung der Akzeptanzkriterien wird in entsprechenden Störfallsimulationen untersucht, die die sicherheitsrelevanten Größen der interessierenden physikalischen Prozesse liefern.

Um wertvolle Einblicke in die Systemdynamik zu gewinnen und wirksame Gegenmaßnahmen zur Vermeidung oder Minderung von Unfallfolgen zu definieren, wird ein Ansatz benötigt, der in der Lage ist, denjenigen Bereich der Eingangsparameterwerte zu bestimmen, der zu Cliff-Edge-Effekten (CEE) führt. Dabei bezieht sich ein CEE auf ein nichtlineares Systemverhalten, bei dem kleine Abweichungen der Eingangsparameterwerte die Veränderung von unkritischen zu kritischen Systemzuständen und umgekehrt bewirken. Aufgrund des hohen Rechenaufwands ist eine einfache MCS für solche Analysen nicht durchführbar.

Im Folgenden wird ein adaptiver Gauß-Prozess-Sampling-Ansatz zur Lokalisierung des für CEE anfälligen Eingangsparameterbereichs (CEE-Region) vorgestellt. Ein Gauß-Prozess (GP, /WIL 06/) wird iterativ als Metamodell auf einen wachsenden Datensatz von Eingangsparametervektoren und zugehörigen Systemantworten trainiert. Durch die Anwendung des aktualisierten GP auf eine große Population von Eingangsparametervektoren und die Bewertung seiner Vorhersage anhand geeigneter Lernkriterien, die darauf abzielen, den Abstand zwischen der vorhergesagten und der angestrebten Systemantwort zu verringern, werden die GP-Trainingsdaten effizient um geeignete Parametervektoren erweitert. Die hinzugefügten Parametervektoren werden verwendet, um die jeweilige Systemantwort durch Ausführen des Simulationscodes zu berechnen. Mit diesen Systemantworten werden die neuen parameterbezogenen Einträge in den GP-Trainingsdaten vervollständigt. Der Fokus des resultierenden Sampling-Prozesses wird adaptiv auf die CEE-Region gerichtet.

Um ein genaues Verständnis für den Begriff CEE zu schaffen, wird die Standarddefinition im kerntechnischen Kontext gegeben. Dann wird der aktive adaptive Lernansatz

beschrieben, der ein GP-Regressionsmodell mit MCS und geeigneten Lernkriterien kombiniert. Basierend auf der Idee, dass die Lokalisierung einer CEE-Region als die Lokalisierung einer beliebigen Region innerhalb des Eingabeparameterraums interpretiert werden kann, kann die Leistung der Methode für eine relativ gut spezifizierte Modellvorgabe untersucht werden. Das verwendete biologische Dosismodell, das im Rahmen der Methodenentwicklung verwendet wurde, wird erläutert. Der adaptive GP-Sampling-Ansatz wird auf das Modell angewandt und seine Leistungsfähigkeit wird diskutiert. Die hier beschriebene und in /BER 20/ publizierte Methodenentwicklung schließt mit nächsten möglichen Schritten zur Verbesserung des vorgestellten adaptiven MCS-Verfahrens im Kontext von Störfallanalysen.

3.1 Cliff-Edge-Effekte

Eine allgemein akzeptierte Definition eines Cliff-Edge-Effekts (CEE) lautet wie folgt (/IAE 16/): "*A cliff edge effect, in a nuclear power plant, is an instance of severely abnormal plant behaviour caused by an abrupt transition from one plant status to another following a small deviation in a plant parameter, and thus a sudden large variation in plant conditions in response to a small variation in an input*." Die Fähigkeit der Lokalisierung von interessierenden Regionen des Eingangsparameterraums einer Simulation stellt somit eine wichtige Voraussetzung dar, um die im Antwortraum eines komplexen dynamischen Systems auftretenden CEEs umfassend zu untersuchen. Aufgrund der hohen Dimensionalität des Eingaberaums und der sehr kleinen Eintrittswahrscheinlichkeiten spezieller Regionen ist es häufig jedoch nicht möglich, diese Lokalisierungsaufgabe durch einen groben MCS-Ansatz zu bewältigen.

Um einen geeigneten Sampling-Ansatz zu entwickeln, der in der Lage ist, adaptiv gegen die Eingangsraumregion einer Simulation zu konvergieren, die zu CEEs des jeweiligen Simulationsergebnisses führt, können maschinelle Lernansätze verwendet werden, um einen automatisierten Lernprozess zu realisieren. Das Hauptziel eines solchen Lernprozesses ist es, nur so viele Informationen zu sammeln, wie nötig sind, um zuverlässig Kombinationen von Eingabeparametern vorherzusagen, die zu Antworten (Simulationsergebnissen) führen, die innerhalb der angestrebten Antwortregion liegen, d. h. die potenzielle CEE-Region anzeigen. Der hier vorgestellte Ansatz des aktiven adaptiven Lernens kann auf robuste Weise Parametervektoren voraussagen, die zu einer anvisierten Antwortregion führen. Dadurch wird eine umfassende Analyse des Eingabeparameterbereichs für bestimmte Systemzustände innerhalb komplexer Simulationen ermöglicht.

3.2 Aktiver Adaptiver Lernansatz

Basierend auf der in /ECH 11/ und /LEL 18/ beschriebenen aktiv lernenden Methode, die Kriging und Monte-Carlo-Simulation (AK-MCS) kombiniert, wird ein Ansatz zur Lokalisierung von Regionen im Eingaberaum einer Simulation vorgestellt, die zu kritischen Zustandsübergängen im Antwortraum des Systems führen. Ziel ist es dabei, einen Ansatz bereitzustellen, der für die Untersuchung von CEEs bei langlaufenden Simulationen mit hochdimensionalem Eingaberaum und komplexem Antwortmuster anwendbar ist. Aus diesem Grund vermeidet der adaptive GP-Sampling-Ansatz komplexe Entscheidungsstrategien, die auf mehreren kombinierten Entscheidungskriterien basieren, um ein transparentes Verständnis des Lernprozesses über die Lernzyklen zu ermöglichen.

Hier wird ein Ansatz des maschinellen Lernens mit einem GP-Regressionsmodell als Metamodell verwendet, um Informationen über den Eingabeparameterraum auf der Grundlage gegebener berechneter Beobachtungen, d. h. Eingangsmuster mit entsprechenden Antwortwerten, zu sammeln. Das trainierte Metamodell wird angewendet, um iterativ den nächsten geeigneten Satz von Beobachtungspunkten, d. h. Eingangsparametervektoren, zu identifizieren, die für die Berechnung weiterer Systemantworten berücksichtigt werden sollen. Somit ermöglicht der GP eine adaptive Abtastung des Eingabeparameterraums in Bezug auf die angestrebte Systemantwort. Der Ansatz wählt aktiv geeignete Eingangsparametervektoren aus, um Systemantworten nahe oder innerhalb eines angestrebten Bereichs zu erhalten.

3.2.1 Lernstrategie

Im Folgenden wird die Lernstrategie des adaptiven GP-Sampling-Ansatzes detailliert erläutert und anhand eines Arbeitsablaufs in Abb. 3.1 dargestellt. Das Ziel der Lernstrategie ist es, Informationen über einen interessierenden Antwortbereich eines Systems abzuleiten, indem Antworten $y_i \in R$ an ausgewählten Punkten im Eingaberaum $\boldsymbol{x}_i \in R^p$ berechnet werden. Der Ansatz basiert auf einer Sammlung von Eingangsvektoren aus dem Eingangsraum R^p, die als Eingangspool $\mathcal{S}$ bezeichnet wird und insgesamt s Eingangsvektoren $\boldsymbol{x}_i$umfasst:

$$\mathcal{S} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_s\}. \tag{3.1}$$

Basierend auf einer zufälligen Auswahl von Eingangsvektoren aus dem Pool $\mathcal{S}$ kann ein anfänglicher Satz von Beobachtungen erzeugt werden, indem zusätzlich die entsprechende Simulation des Systems durchgeführt wird. Die resultierenden n Beobachtungen oder Datenpunkte können formal organisiert werden durch

$$\mathcal{D} = \{(y_i, \boldsymbol{x}_i) | i = 1, \dots, n\} \coloneqq (\boldsymbol{y}, \mathbf{X}) \tag{3.2}$$

$\mathcal{D}$ stellt eine Menge von Antwort- und Eingangsvektorpaaren dar (Trainingsdaten). Der Einfachheit halber werden die Antworten gemeinsam als $\boldsymbol{y} = [y_1, \dots, y_n]^{\boldsymbol{T}}$ und die Eingangsvektoren als $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^{\boldsymbol{T}}$ zusammengefasst.

Angesichts dieser anfänglichen Beobachtungen kann ein Metamodell trainiert werden, das zur Identifizierung geeigneter Kandidaten von Eingangsvektoren verwendet wird. Diese Kandidaten sollen die Informationen über den Parameterbereich erhöhen, der zur angestrebten Systemantwort führt. Ein geeignetes Metamodell liefert das in Abschnitt 3.3.1 besprochene (GP-)Regressionsmodell, da es sicherstellt, dass bereits berechnete Beobachtungen bei der Regressionsinferenz nicht mit Unsicherheit belegt werden. Basierend auf der Vorhersage des trainierten GP können die noch nicht ausgewerteten Eingangsvektoren des Pools $\mathcal{S}$ nach Kriterien angeordnet werden, die die Nähe der vorhergesagten zur angestrebten Antwort bewerten, wie in Abschnitt 3.3.2 besprochen.

Im sogenannten Lernschritt wird eine vordefinierte Anzahl von Kandidaten, d. h. geeigneter - noch nicht ausgewerteter - Eingangsvektoren $\boldsymbol{X}$ aus dem Pool $\mathcal{S}$ zur Berechnung der Antworten $\boldsymbol{y}$ durch Ausführen der eigentlichen Simulation verwendet. Dies vergrößert die Menge der Beobachtungen $\mathcal{D}$ sowie den Informationsgehalt, auf den der GP im letzten Teil des Lernschritts trainiert wird. Durch die iterative Durchführung des Lernschritts wird die resultierende Abtastung des Pools $\mathcal{S}$ adaptiv auf den Bereich gelenkt, der zur angestrebten Systemantwort führt.

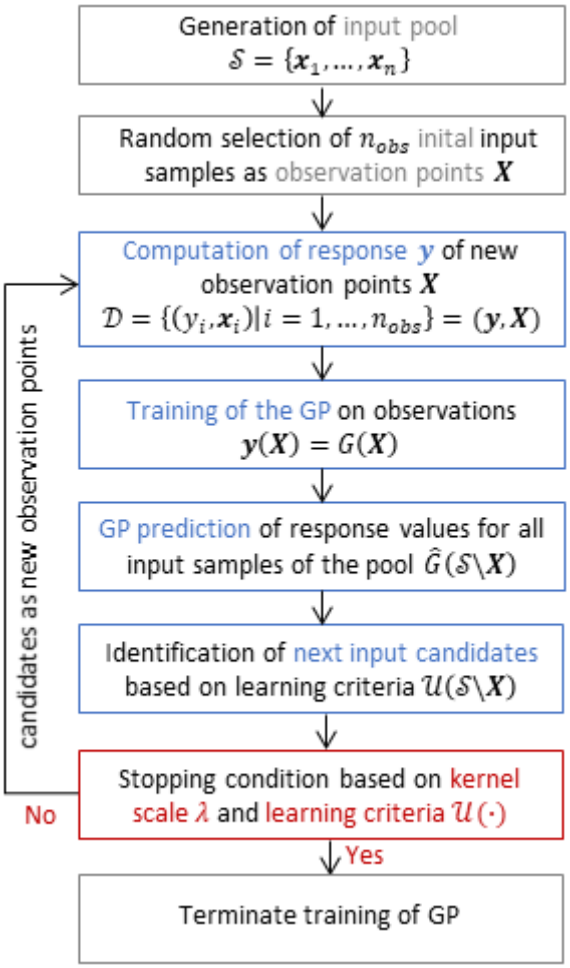

Abb. 3.1 Ablauf des aktiven adaptiven Lernansatzes unter Verwendung eines Gauß-Prozesses (GP) als Metamodell

3.3 Komponenten des Gauß-Prozess-Sampling-Ansatzes

3.3.1 Gauß-Prozesse als Metamodelle

Die GP-Regressionsmodelle stellen eine Klasse von probabilistischen statistischen Modellen dar, die einen GP verwenden, um die Unsicherheit über eine latente Funktion zu charakterisieren /HSI 09/. Die a-priori GP-Verteilung einer latenten Funktion kann verwendet werden, um ein flexibles probabilistisches Klassifikationsmodell zu definieren, das eine Annäherung der Bayes'schen Inferenz für komplexe hochdimensionale Muster ermöglicht.

Grundsätzlich ist ein GP eine Sammlung von Zufallsvariablen, die einer gemeinsamen Gaußschen Verteilung folgen:

$$G(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \sim \mathcal{N}\big((\mu(\boldsymbol{x}_1), \dots, \mu(\boldsymbol{x}_n)\,), \mathcal{K}\big) \tag{3.3}$$

mit Mittelwert $\mu(\boldsymbol{x}_i)$ bei jedem Eingabevektor $\boldsymbol{x}_i \in R^p$ und Kovarianzmatrix $\mathcal{K} \in R^{(nxn)}$, wie im Folgenden erläutert. Der GP kann linear in einen deterministischen Term f und einen Fehlerterm ε zerlegt werden:

$$G(\boldsymbol{X}) = f(\boldsymbol{X}, \boldsymbol{w}) + \varepsilon(\boldsymbol{X}) \tag{3.4}$$

Der deterministische Term stellt eine Annäherung an den Mittelwert der Systemantwort (Simulationsergebnis) bei jedem Eingabevektor $\boldsymbol{x}_i$ dar:

$$y(\boldsymbol{x}_i) \approx \mu(\boldsymbol{x}_i) = f(\boldsymbol{x}_i, \boldsymbol{w}) = \boldsymbol{\phi}^T(\boldsymbol{x}_i) \cdot \boldsymbol{w} \tag{3.5}$$

Er ist modelliert durch einen Satz von angenommenen Basisfunktionen $\boldsymbol{\phi} \in \boldsymbol{R^m}$ und einen Gewichtungsvektor als Vektor der Regressionskoeffizienten $\boldsymbol{w} \in \boldsymbol{R^m}$. Bei dem hier verwendeten Ansatz wird ein gewöhnlicher GP angewendet, der das Problem auf die Bestimmung eines Skalars für den deterministischen Term reduziert.

Der Fehlerterm ist als stationärer GP definiert:

$$\varepsilon(\boldsymbol{X}) \sim \mathcal{N}(0, \mathcal{K}) \tag{3.6}$$

Die Kovarianzmatrix $\mathcal{K}$ charakterisiert die Korrelationen zwischen den Parametervektoren des Eingaberaums über eine Kernel-Funktion $K(\cdot)$:

$$\mathcal{K}_{ij} = K\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) \tag{3.7}$$

Neben der Modellierungsannahme bzgl. der Basisfunktionen des deterministischen Teils stellt die angenommene Kernel-Funktion eine weitere Modellierungsannahme dar, die

für die Spezifikation eines GP notwendig ist. Als häufig verwendete Funktion zur Modellierung von graduellen Mustern des Kernels wird die isotrope Gauß-Kernel-Funktion, auch bekannt als Radialbasisfunktion (RBF), verwendet. Sie ist definiert als

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_0^2 \cdot exp\left(-\frac{1}{2}\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{\lambda^2}\right) \tag{3.8}$$

und verwendet den euklidischen Abstand zwischen zwei Eingabevektoren $\boldsymbol{x}_i, \boldsymbol{x}_j$ und einen Längenskalenparameter λ > 0, der die Zerlegung der Kovarianz zwischen den Eingangsvektoren über den Abstand angibt. Die Kernel-Funktion wird durch eine globale Varianz σ_0^2 skaliert.

Basierend auf dem Bayes'schen Theorem und den GP-Modellspezifikationen kann die Posteriorverteilung der latenten Funktionssammlung $\boldsymbol{f} = [f_1, \ldots, f_n]^T$ ausgedrückt werden als

$$p\left(\boldsymbol{f}\middle|\mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_K\right) = \frac{p\left(\boldsymbol{y}\middle|\boldsymbol{f}, \boldsymbol{\theta}_f\right) \cdot p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\theta}_K)}{p\left(\mathcal{D}\middle|\boldsymbol{\theta}_f, \boldsymbol{\theta}_K\right)} \tag{3.9}$$

Die Posteriorverteilung hängt von den Beobachtungen $\mathcal{D}$ ab, die für das Training des GP herangezogen werden, und von den Hyperparametern $\boldsymbol{\theta}_{(\cdot)}$, die zur Definition des deterministischen Terms $\boldsymbol{\theta}_f$ und der Kernel-Funktion $\boldsymbol{\theta}_K$ verwendet werden. Die Likelihood-Funktion $p\left(\boldsymbol{y}\middle|\boldsymbol{f}, \boldsymbol{\theta}_f\right)$ wird als bedingte Verteilung des Ergebnisses $\boldsymbol{y}$ bei gegebener latenter Funktionssammlung $\boldsymbol{f}$ und entsprechenden Hyperparametern $\boldsymbol{\theta}_f$ berechnet. Die Prior-Verteilung $p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{\theta}_K)$ quantifiziert den Grad der anfänglichen Unsicherheit bzgl. der latenten Funktion angesichts des Eingabevektors X der Trainingsdaten und der angenommenen Kernel-Funktionen, die durch $\boldsymbol{\theta}_K$ parametrisiert sind. Der Nenner $p\left(\mathcal{D}\middle|\boldsymbol{\theta}_f, \boldsymbol{\theta}_K\right)$ stellt im Grunde einen Normalisierungsterm dar. Weiterführende Details des Bayes'schen Inferenzverfahrens hinsichtlich der analytischen Inferenz sowie der numerischen Optimierungsaufgaben werden in /ECH 11/ und /LEL 18/ beschrieben und diskutiert.

Mit der ermittelten Posteriorverteilung wird der GP $G(\boldsymbol{x})$ neu geschätzt. Durch die Anwendung der Posteriorverteilungfunktion auf beliebige Beobachtungspunkte (Eingabe-

vektoren) $\boldsymbol{x}'$, die nicht zum Trainingsdatensatz gehören, erhält man die Vorhersage der entsprechenden Antwort als $y' = \hat{G}(\boldsymbol{x}')$.

Eine wichtige Eigenschaft des geschätzten GP stellt die GP-Varianz $\sigma^2_{\hat{G}}(\boldsymbol{x}')$ dar. Diese ist definiert als das Minimum des mittleren quadratischen Fehlers des trainierten GP-Ergebnisses $G(\boldsymbol{x}')$ und des geschätzten (d.h. vorhergesagten) GP-Ergebnisses $\hat{G}(\boldsymbol{x}')$. Dadurch wird die GP-Varianz $\sigma^2_{\hat{G}}(\boldsymbol{x}')$ für alle bereits für das GP-Training verwendeten Eingabevektoren $\boldsymbol{x}' \in \mathbf{X}$ zu Null. Dies veranschaulicht den Vorteil von GP-Regressionsmodellen, der sich aus der Tatsache ergibt, dass durch die Annahme einer Gauß-Verteilung die Bayes'sche Inferenz, d. h. die Ermittlung der Posteriorverteilung, analytisch nachvollziehbar und exakt ist /HSI 09/.

3.3.2 Lernkriterium

Das Ziel der Lernstrategie ist es, die Informationen in Bezug auf eine Zielregion, d. h. eine angestrebte Region innerhalb des Antwortraums, zu erhöhen. Ein mögliches Ziel, insbesondere im Kontext von Sicherheitsanalysen, könnte ein kritischer Schwellenwert y_{target} einer Antwortvariablen sein.

Hier wird folgendes Lernkriterium angewendet:

$$|\mathcal{U}(\boldsymbol{x})| = \frac{\left|y_{target} - \hat{G}(\boldsymbol{x})\right|}{\sigma_{\hat{G}}(\boldsymbol{x})} \tag{3.10}$$

Es stellt den Abstand zwischen dem Schwellenwert (Zielergebnis) y_{target} und der GP-Vorhersage $\hat{G}(\boldsymbol{x})$ skaliert mit der entsprechenden GP-Standard Abweichung $\sigma_{\hat{G}}(\boldsymbol{x})$ dar. Dabei zeigt ein kleiner Wert des Lernkriteriums für einen gegebenen Beobachtungspunkt $\boldsymbol{x}$ einen relativ kleinen geschätzten Abstand zum Zielergebnis und/oder eine relativ große Unsicherheit des geschätzten Ergebnisses an. Durch Anwendung dieses Lernkriteriums begünstigt der Abtastprozess Eingabevektoren (Beobachtungspunkte), für die der GP Resultate nahe des Schwellenwerts vorhersagt, während die Unsicherheit der Vorhersage des GP-Regressionsmodells verringert wird.

Ein unterer Schwellenwert für den minimalen Wert des Lernkriteriums $|\mathcal{U}(\boldsymbol{x_i})|$ der aktuellen Kandidaten $\boldsymbol{x}_i \in \boldsymbol{X}_c$ in einem Lernschritt kann als Abbruchkriterium für den

Lernprozess verwendet werden /ECH 11/. Da die Entscheidung darüber, welcher Wert des Lernkriteriums eine angemessene Güte der GP-Regression *a priori* zum Lernprozess anzeigt, bei komplexen Simulationen weder sinnvoll noch machbar ist, wird diese Stoppbedingung nicht angewendet. Allerdings wird in der Anwendung des Ansatzes eine Entscheidungsstrategie diskutiert, die für hochdimensionale Probleme, wie sie bei komplexen Simulationscodes zu erwarten sind, wesentlich besser geeignet ist.

3.4 Anwendung auf ein Biologisches Dosismodel

Der vorgestellte adaptive GP-Sampling-Ansatz wird an einem Dosis-Simulationsmodell demonstriert, das eine skalare Antwort $y_i \in R$ basierend auf einem 6-dimensionalen Eingangsvektor $\boldsymbol{x}_i \in R^6$ liefert. Das Modell dient als nützliches Demonstrationsbeispiel, um die Performanz des Ansatzes für einen kontrollierbaren mehrdimensionalen Eingaberaum mit p > 2 und heterogenen Skalen der Eingabevariationen im Detail zu untersuchen. Ein solides Verständnis des Ansatzes in einer solchen realistischeren Umgebung stellt eine wichtige Voraussetzung für die Anwendung der Methode auf eine komplexere und weniger kontrollierbare Umgebung dar, wie z. B. die Simulation eines Störfallszenarios in einem KKW.

Das Dosismodell berechnet die maximale jährliche biologische Dosis y (Äquivalentdosis) eines (durchschnittlichen) Individuums einer Bevölkerungsgruppe, die der Radioaktivität in Lebensmitteln ausgesetzt ist. y ist durch die Formel in Gl. (2.23) gegeben. Als kritischer Wert der biologischen Dosis wird hier der Wert $y_{crit} = 10^{-5} Sv/a$ betrachtet. Die für den GP-Sampling-Ansatz interessante Antwort wird durch die Differenz $y_{crit} - y$ angegeben, die zum Schwellenwert von $y_{target} = 0$ führt.

3.4.1 Standardisierung der Eingabevektoren

Wie durch die Nominalwerte in Tab. 2.1 angedeutet, bewegen sich die Eingabeparameter in unterschiedlichen Größenordnungen, d. h. zwischen $O(10^{-8})$ und $O(10^{2})$. Jeder Eingabeparameter, der für das Training eines Metamodells verwendet wird, wird im Folgenden als Feature bezeichnet. Um die Verwendung eines maschinellen Lernalgorithmus zu erleichtern, wird jedes Feature standardisiert, indem es über den Mittelwert einer vorliegenden Feature-Menge zentralisiert und über die Standardabweichung der Feature-Menge neu skaliert wird. Im Kontext eines GP-Metamodells mit der RBF als Kernelfunktion ist die Standardisierung besonders wichtig, da diese Kernel-Funktion auf der

Annahme beruht, dass jedes Feature um Null zentriert ist und in der Standardabweichung in vergleichbarer Größenordnung variiert. Jedes Feature, dessen Varianz um Größenordnungen größer ist als die der anderen, dominiert die jeweilige Kernelfunktion und verhindert, dass der Schätzer aus den übrigen Features korrekt lernen kann.

Um die numerische Stabilität der iterativen GP-Regression während des Lernprozesses zu verbessern, werden auch die Ergebniswerte über eine nichtlineare Quantilstransformation in standardnormalverteilte Werte umgewandelt.

Tab. 3.1 Veränderte Verteilung des Parameters $\boldsymbol{c}$ aus Tab. 2.2

Parameter	Verteilung	Verteilungsparameter	Minimum; Maximum
c	normal	3.25E-08, 9.75E-09	1.E-08; 5E-08

Basierend auf den in Tab. 2.2 und Tab. 3.1 zusammengefassten Eingangsspezifikationen werden die Eingangsvektoren des Eingangspools $\mathcal{S}$ generiert und auf Grundlage der schnell ablaufenden Simulation die entsprechenden Ergebniswerte des Dosismodells berechnet. Die resultierenden normierten Werte der ausgewählten Feature c (Dosiskonversionsfaktor) und dt (Verzögerungszeit) sind zusammen mit den jeweiligen umgewandelten Ergebniswerten in Abb. 3.2 dargestellt. Die für die Feature- und Ergebnisstandardisierung notwendigen Transformationen werden jeweils neu berechnet und angewendet, sobald sich der Pool der Eingabevariationen $\mathcal{S}$ oder die Menge der verfügbaren Simulationswerte (d. h. die Trainingsdaten) ändert, um die Schritte eines Lernprozesses konsistent zu halten.

3.4.2 Zielbereich der Simulationsergebnisse

Als Zielregion wird der Variationsbereich um den entsprechend transformierten Simulationswert y_{target} betrachtet. Dieser ist definiert als $\Delta y = y_{target} \pm 10^{-7}$. In Abb. 3.2 wird der Parameterbereich mit Ergebniswerten $y \in \Delta y$ (cyanfarbene Punkte) angedeutet durch die konvexe Einhüllende (gestrichelte Linien). Dabei bezieht sich die Farbkodierung auf die normierten Simulationsergebnisse y. Die adaptive Inferenz auf den Eingabebereich, der zu einem angestrebten Ergebnisbereich führt, stellt eine adaptive Näherung bzgl. des eigentlichen Ziels der Methodenentwicklung dar, nämlich der Lokalisierung der CEE-Region. Diese CEE-Region setzt sich aus potenziell getrennten Bereichen innerhalb des Eingaberaums zusammen, die zu kritischen vs. unkritischen Simulationsergebnissen führen.

Basierend auf dieser Idee wurde die Performanz des vorgestellten adaptiven GP-Sampling-Ansatzes anhand seiner Fähigkeit untersucht, einen Eingabebereich mit Ergebnissen $y \in \Delta y$ zu finden, der durch eine Stichprobe (im Folgenden als Population bezeichnet) von $\boldsymbol{n = 10^5}$ Beobachtungen $(\boldsymbol{y, X})$ bekannt ist.

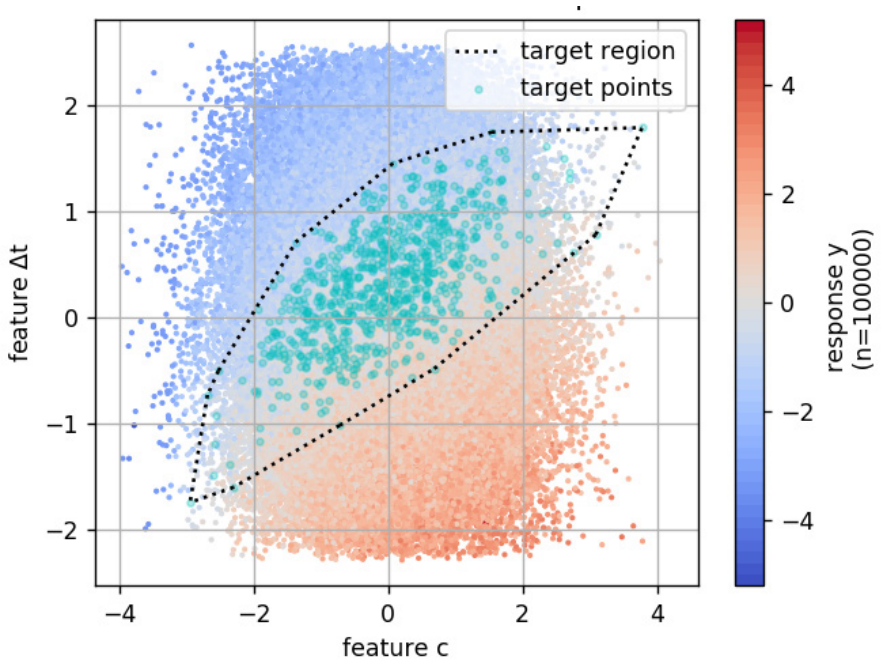

Abb. 3.2 Standardisierte Werte der ausgewählten Parameter (Feature) c (Dosiskonversionsfaktor) und Δt (Verzögerungszeit) des Dosismodells für $n = 10^5$. Der Farbcode stellt den normierten Ergebniswert y dar und die gestrichelten Linien zeigen den angestrebten Parameterbereich an

3.4.3 Zielsetzung des adaptiven Samplings

Der adaptive GP-Sampling-Ansatz wird auf das im vorherigen Abschnitt definierte Dosismodell angewendet. Um die Vorhersageleistung des Ansatzes zu vergleichen, wurde das GP-Metamodell auf einer Untermenge mit den beiden Eingabeparametern c (Dosiskonversionsfaktor) und Δt (Verzögerungszeit) trainiert. Die Auswahl dieser beiden Eingabeparameter basierte auf einer explorativen Analyse der Ergebnisse in der Population von $\boldsymbol{n = 10^5}$ Beobachtungen mit dem Ziel, Merkmalskombinationen zu identifizieren, die die Lokalisierung einer interessierenden Parameterregion und insbesondere die Nachbildung ihrer Ränder ermöglichen (vgl. Abb. 3.2). Ein Ziel der Anwendung war unter anderem, die Robustheit der Vorhersage für den Fall zu untersuchen, dass nicht alle potenziellen Features während der GP-Regressionsadaption berücksichtigt werden.

Um den Ansatz richtig zu verstehen und zu validieren, wird die Leistung der Kandidatenauswahl und die Konvergenz über die Lernzyklen untersucht. Neben der Untersuchung der Vorhersagefähigkeit des adaptiven GP-Sampling-Ansatzes stellt ein weiteres Ziel der Anwendung die Lokalisierung der Eingaberegion dar, die zu einem Ergebnis in der Zielregion führt. Anders als bei üblichen adaptiven GP-Sampling-Anwendungen mit homogenen Parameterregionen, die eine klare Trennung von Parametern mit kritischen und unkritischen Ergebnissen zeigen (vgl. /ECH 11/ oder /LEL 18/), ist die kritische Parameterregion des Dosismodells relativ unregelmäßig und komplex. Aus diesem Grund ist die Fähigkeit der GP-Regression von Interesse, die Ränder der Zielregion im einfachen Feature-Raum zu lokalisieren.

Der adaptive Sampling-Ansatz wurde auf der Basis eines zufällig ausgewählten Trainingsdatensatzes der Größe 20 aus dem Eingabepool $\mathcal{S}$ der Größe $\boldsymbol{n = 10^5}$ initialisiert. In jedem Lernzyklus wurden fünf weitere Kandidaten identifiziert. Insgesamt ergaben sich 37 Lernzyklen inklusive des initialen Lernschritts.

3.4.4 Durchführung der Kandidatenauswahl

Um die Leistungsfähigkeit des aktiven adaptiven Lernansatzes zu verdeutlichen, zeigt Abb. 3.3 die empirische Verteilung der Simulationsergebnisse in originaler (oben) und standardisierter (unten) Darstellung, berechnet aus 200 adaptiv gesampelten Eingabevektoren. Aufgrund des definierten Lernziels $y_{target} = 0$ wird das Sampling adaptiv auf den Parameterbereich fokussiert, der zu Simulationsergebnissen führt, die nahe an dem angestrebten Zielwert liegen.

Da die adaptiv gesampelten Simulationswerte um den Zielwert konzentriert sind, kann die Gesamtleistung des aktiven adaptiven Lernansatzes trotz der komplexen, unregelmäßigen Eigenschaften der kritischen Parameterregion bestätigt werden. Der Zusammenhang zwischen einer komplexen Parameterregion und der Stabilität des Simulationsergebnisses kann durch eine genauere Untersuchung des Ablaufs des Lernprozesses besser verstanden werden.

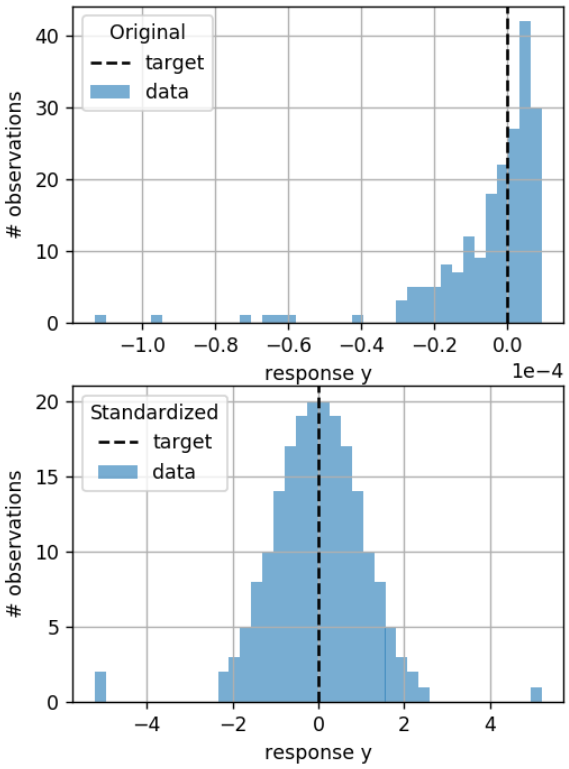

Abb. 3.3 Empirische Verteilung von 200 adaptiv gesampelten Simulationsergebnissen in originaler (oben) und normierter (unten) Darstellung um die jeweiligen Zielwerte. Die Abbildung bezieht sich auf den in Abb. 3.4 zusammengefassten Lernprozess

3.4.5 Fortschritt des Lernprozesses

Der Fortschritt des aktiven adaptiven Lernprozesses kann durch den minimalen Wert des in Gl. (3.10) definierten Lernkriteriums $|\mathcal{U}(\boldsymbol{x})|$ und durch die Kernel-Längenskala λ des GPs (Gl. (3.8)) angezeigt werden, wie in Abb. 3.4 dargestellt. Das Verhalten des minimalen Werts des Lernkriteriums (rot in Abb. 3.4), also $\min\left(|\mathcal{U}(\boldsymbol{x})|\right)$ für alle $\boldsymbol{x}_i \in \mathcal{S}\backslash\mathcal{D}$ pro Lernschritt, zeigt grundsätzlich an, dass vor der Auswahl von Eingabeparametern in der Nähe der Zielregion die Unsicherheit des GP-Regressionsmodells zu reduzieren ist. Die allgemeine Abnahme der Kernel-Längenskala λ (blau in Abb. 3.4) im Verlauf des Lernprozesses bestätigt den aktiven Lerneffekt.

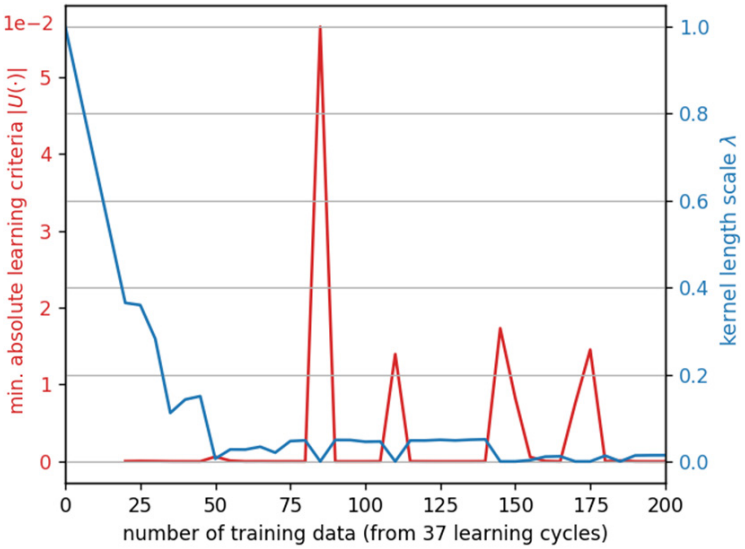

Abb. 3.4 Minimum des Lernkriteriums $|\mathcal{U}(x)|$ und Kernel-Längenskala λ des trainierten GP-Modells als Indikatoren für den Fortschritt des adaptiven Sampling-Lernprozesses bzgl. des Dosismodells

In einigen Lernzyklen (z. B. wenn Anzahl Trainingsdaten = 85 in Abb. 3.4) tritt bei der GP-Regression eine typisch unerwünschte Situation auf: ein punktuelles Overfitting, numerisch angezeigt durch eine extrem kleine Kernel-Längenskala, begleitet von einem starken Ansteigen des Lernkriteriums und visuell angezeigt durch eine unscharfe Posteriorverteilung im Feature-Raum. Im Allgemeinen verschwindet das Overfitting im folgenden Lernzyklus. In Anbetracht komplexer Simulationseinstellungen ist zu beachten, dass dieses Verhalten ein transparenteres Abbruchkriterium erfordert als einen a priori definierten minimalen Schwellenwert des Absolutwerts des Lernkriteriums, wie in /ECH 11/ vorgeschlagen.

3.4.6 Lokalisierung der Zielregion

Für jeden Lernzyklus des adaptiven Lernprozesses wurden die Vorhersagen des GP-Prozesses (Posteriorverteilung und ihr Mittelwert) für die Ränder des Parameterbereichs mit Ergebniswerten $y \in \Delta y$ (siehe Abb. 3.2) untersucht. Die Lernzyklen, die zu einer extremen Überanpassung führen, sind nicht in der Lage, diesen Parameterbereich zu lokalisieren. Trotz dieser punktuellen Overfitting-Ereignisse liefern die verbleibenden Lernzyklen wertvolle Schätzungen, die prinzipiell in der Lage sind, auf die angestrebte Region im Parameterraum zu schließen. In Abb. 3.5 sind die GP-Vorhersagen (Mittelwert der Posteriorverteilung) der Ergebniswerte y bzgl. der interessierenden (standardisierten) Parameterregion beim 13. Lernschritt dargestellt. Wie zu erkennen ist, kann die konvexe

Einhüllende (gestrichelt) der Zielregion Δy (vgl. Abb. 3.2) durch den trainierten GP nicht vollständig nachgebildet werden, jedoch gut genug, um gültige Kandidaten (Dreiecke) für den nächsten Schritt im Lernprozess zu identifizieren.

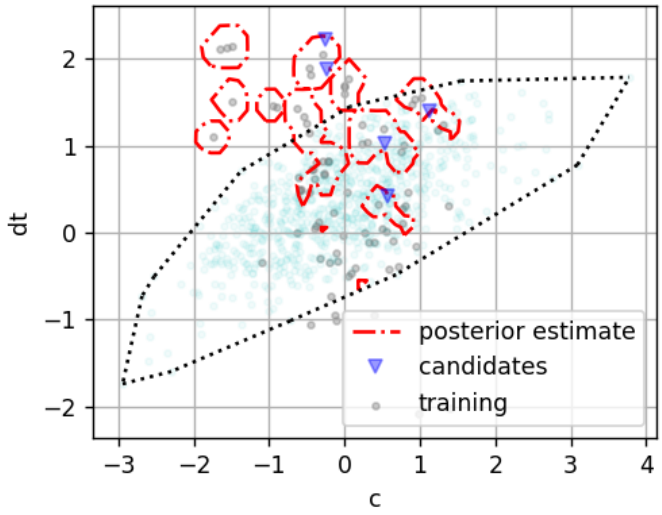

Abb. 3.5 GP-Vorhersage (Mittelwert Posteriorverteilung) der angestrebten Zielregion über dem Parameterraum (Δt, c) beim Zwischenlernschritt 13 und identifizierte Eingabekandidaten für den nachfolgenden Schritt des in Abb. 3.4 dargestellten Lernprozesses

Die Zunahme der Trainingsdaten während des laufenden Lernprozesses führt zu einer kontinuierlichen Überanpassung des GP, was darauf hindeutet, dass die komplexen Eigenschaften des Parameterbereichs mit Ergebnissen in der Zielregion die zuverlässige Nachbildung der Ränder des Parameterbereichs erschweren. Um die Lokalisierungsfähigkeit des adaptiven GP-Sampling-Ansatzes zu verbessern, sollten verschiedene GP-Kernel-Spezifikationen untersucht werden, die weniger empfindlich auf irreguläre, komplexe Variationen der Parameterregion reagieren. Außerdem sollten mehr als zwei Parameter für das GP-Training verwendet werden.

4 Anwendung auf einen Kühlmittelverluststörfall mit ATHLET als Simulationscode

Die Referenzanlage, auf die sich die im Folgenden beschriebenen Analysen beziehen, ist ein Druckwasserreaktor (DWR) vom Typ Vorkonvoi (1425 MW elektrische Leistung, 3950 MW thermische Reaktorleistung). Als Störfall wird ein 2F-Bruch im kalten Strang der Hauptkühlmittelleitung angenommen. Die Simulation des Störfalls wurde mit dem Simulationscode ATHLET in der Version 3.2 durchgeführt. Im zugrundeliegenden Datensatz wurde die Referenzanlage mit einem 4-Loop-Modell nachgebildet. Der Reaktordruckbehälter wurde mit 7 thermohydraulischen Kernkanälen (5 normal belastete Kernkanäle, ein hochbelasteter Kernkanal sowie ein Heißkanal mit dem Heißstab) nachgebildet. Für die Neutronenkinetik wurde das Punktkinetikmodell gewählt. Die herangezogenen Daten entsprechen einem Kern zu Beginn des Lastzyklus (Beginn Of Cycle). Die anfängliche Borkonzentration beträgt 1150 ppm. Die Öffnung des Lecks im kalten Strang (Loop 10) erfolgt nach 600 s.

Die unsicheren Eingabeparameter des ATHLET Anwendungsfalls und ihre Wahrscheinlichkeitsverteilungen sind in Tab. 4.1 zusammengefasst und basieren auf Angaben aus bereits durchgeführten Unsicherheitsanalysen (/AUS 13/, /PAL 14/, /POI 18/). Es ist zu beachten, dass das Leistungsprofil einer konservativ-deterministischen Analyse berücksichtigt wurde. Die höchste Stablängenleistung betrug circa 460 W/cm (16x16 Brennelement).

Ziel der exemplarischen Analysen zur adaptiven MCS ist die Lokalisierung desjenigen kritischen Bereichs der Eingabeparameter, der zu einer maximalen Hüllrohrtemperatur (PCT für *P*eak *C*ladding *T*emperature) von über 1200 °C führt und die Ermittlung seiner Wahrscheinlichkeit.

Zur Feststellung, in welchem Bereich der PCT-Wert in Abhängigkeit von zufällig ausgespielten Werten für die Eingabeparameter variiert, wurde zunächst eine klassische Unsicherheitsanalyse mit SUSA durchgeführt. Mit einer daran anschließenden Sensitivitätsanalyse wurden die Parameter mit dem größten Einfluss auf die PCT-Variation ermittelt.

Die Ergebnisse der Unsicherheits- und Sensitivitätsanalyse werden in Abschnitt 4.1 beschrieben. In den darauffolgenden Abschnitten erfolgt eine Beschreibung der Ergebnisse zur adaptiven MCS.

4.1 Unsicherheits- und Sensitivitätsanalyse

Für die Unsicherheitsanalyse wurden insgesamt 60 verschiedene Wertekombinationen für die 35 unsicheren Eingabeparameter in Tab. 4.1 zufällig ausgespielt und damit entsprechende Rechenläufe mit ATHLET durchgeführt. Das interessierende Rechenergebnis aus diesen Läufen ist der zeitliche PCT-Verlauf und der entsprechende Maximalwert aus diesem Verlauf.

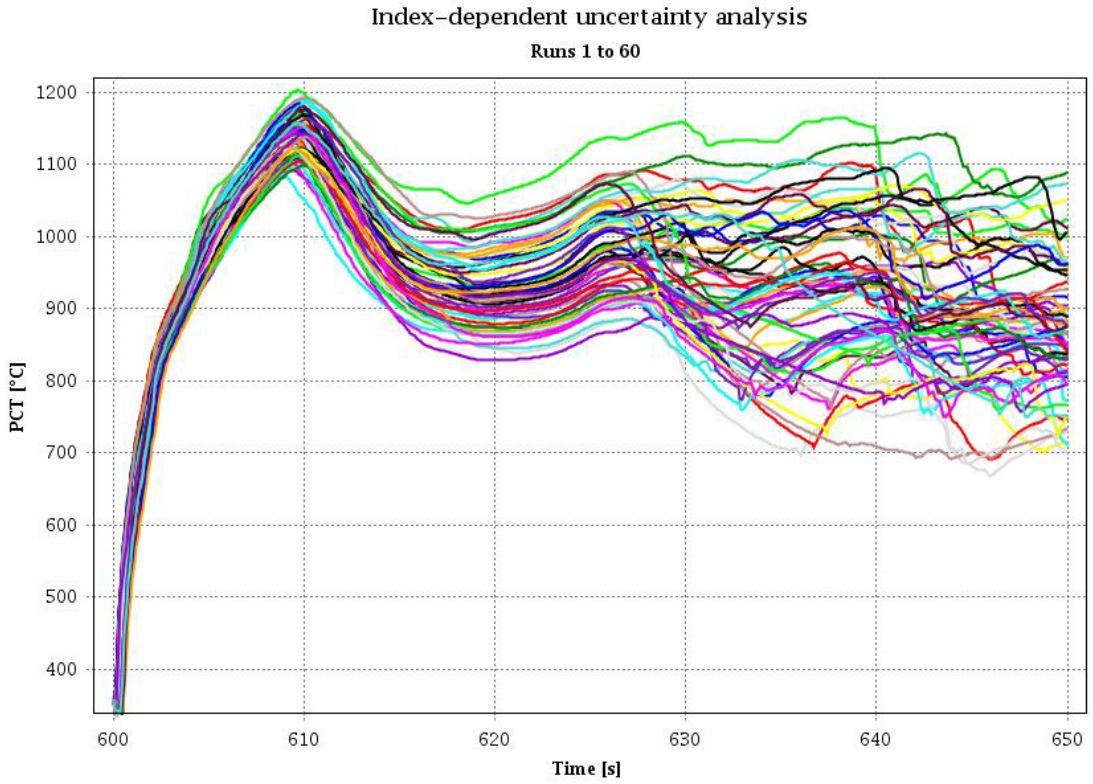

Abb. 4.1 Variation der zeitlichen Verläufe der maximalen Hüllrohrtemperatur (PCT) in 60 ATHLET-Läufen

Die Variation des zeitlichen PCT-Verlaufs zwischen 600 s und 650 s ist in Abb. 4.1 dargestellt. Für den PCT-Maximalwert über die Zeit erhält man die Verteilung in Abb. 4.2 und eine obere (95 %, 95 %) Toleranzgrenze (nach Wilks) von 1203 °C. Diese Toleranzgrenze gibt an, dass mindestens 95 % der PCT-Maximalwerte unterhalb von 1203 °C liegt bei einer statistischen Sicherheit von mindestens 95 %. Hauptgrund für die relativ hohen PCT-Werte ist das konservative Leistungsprofil, das für den Anwendungsfall berücksichtigt wurde.

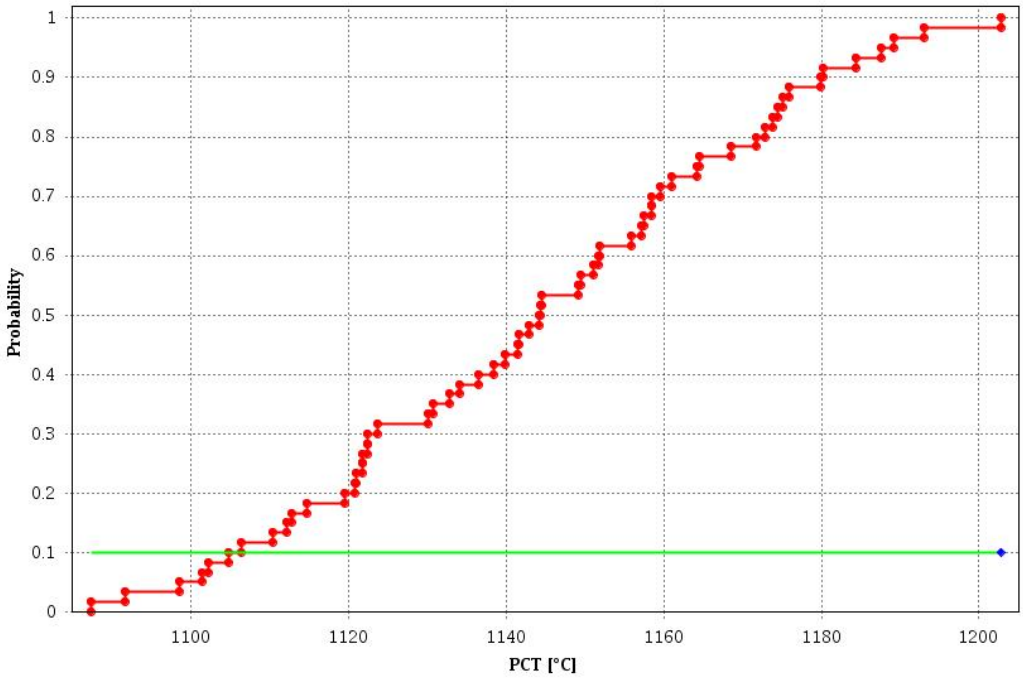

Abb. 4.2 Empirische Wahrscheinlichkeitsverteilung für die maximale Hüllrohrtemperatur (PCT) und obere (95 %, 95 %) Toleranzgrenze nach Wilks (blaue Raute) ermittelt aus 60 ATHLET-Läufen

Der berechnete Wert von 1203 °C für die obere (95 %, 95 %) Toleranzgrenze legt nahe, dass der Anteil der PCT-Werte unterhalb des Grenzwerts von 1200 °C kleiner als 95 % ist. Deshalb wurden obere Toleranzgrenzen mit kleinerer Überdeckungswahrscheinlichkeit β berechnet. Aus der Berechnung der oberen (β, 95 %) Toleranzgrenze mit β = 92 % resultierte ein Wert von 1193 °C. Demnach kann man davon ausgehen, dass mindestens 92 % der möglichen PCT-Werte unterhalb von 1200 °C und damit höchstens 8 % oberhalb des Grenzwertes liegen. Aufgrund des Ergebnisses, dass einer von insgesamt 60 PCT-Maximalwerten oberhalb von 1200 °C liegt, erhält man für das 95 %-Konfidenzintervall nach Pearson und Clopper 4.22E-04 als untere und 8.94E-02 als obere Intervallgrenze für die Wahrscheinlichkeit, dass die 1200 °C-Grenze überschritten wird. In welchem genaueren Bereich sich diese Wahrscheinlichkeit bewegt, kann mit den Methoden der adaptiven MCS mit praktikablem Rechenaufwand ermittelt werden. Diese Methoden können außerdem detaillierte Angaben zum Parameterbereich liefern, der zu PCT-Werten größer als 1200 °C führt.

Eine an die klassische Unsicherheitsanalyse anschließende Sensitivitätsanalyse zeigte, dass die Parameter mit den Nummern 26 (WLFM, Korrekturfaktor der Wärmeleitfähigkeit

des Brennstoffs), 3 (ODBUN, Korrekturfaktor für die relative Geschwindigkeit, Bündelgeometrie, Heizstabbündel) und 7 (IHTC1, Modell für Dampf-Tropfenkühlung) am meisten zur PCT-Variation beitragen. Einen kleineren Einfluss haben die Parameter mit den Nummern 1 (ODVPI, Korrekturfaktor für relative Geschwindigkeit, Vertikale Leitungen), 12 (OTRNB, Kritische Heizflächenbelastung, Minimalwert aus drei Korrelationen – Korrekturfaktor, Kern), 17 (ZT, Zahl der Tropfen pro Einheitsvolumen, Primär- und Sekundärkreislauf), 24 (RPODC, Korrekturfaktor für Nachzerfallswärme), 25 (QROD0KBS, Korrekturfaktor Stableistung, heißer Brennstab) und 32 (TURB, Turbulenzfaktor für Verdampfung bei kritischer Strömung, Bruch). Als Sensitivitätsindex wurde dabei der standardisierte Regressionskoeffizient basierend auf dem Korrelationskoeffizienten nach Pearson ausgewählt. Der standardisierte Regressionskoeffizient wurde deshalb ausgewählt, weil die Stichprobe, aus der die Sensitivitätsindizes ermittelt wurde, mit n = 60 sehr gering ist und deshalb relativ hohe unbeabsichtigte Korrelationen zwischen den Eingangsparametern vorkommen können. Wegen dieser unbeabsichtigten Korrelationen können die normalen Korrelationskoeffizienten zwischen Parametern und Ergebnis (PCT) ein verzerrtes Bild zu den individuellen Einflüssen der Parameter liefern.

Die Abb. 4.3 bis Abb. 4.6 zeigen die Sensitivitätsindizes (standardisierte Regressionskoeffizienten (Pearson)) für die PCT und die einzelnen Parameter im zeitlichen Verlauf. Der dazugehörige zeitliche Verlauf des Bestimmtheitsmaßes R^2, das die Aussagekraft der Sensitivitätsindizes angibt, ist in Abb. 4.7 zu sehen. Werte über 0.8 bedeuten hier eine hohe Aussagekraft. In Abb. 4.8 sind die Sensitivitätsindizes für den im Zeitverlauf erreichten PCT-Maximalwert und die einzelnen Parameter dargestellt. Der dazugehörige R^2-Wert beträgt 0.98 und weist auf eine sehr hohe Aussagekraft dieser Sensitivitätsindizes hin.

Tab. 4.1 Unsichere Parameter des LOCA-Anwendungsfalles und ihre Verteilungen

Par. No.	Parameter ID	Parameter Name	Refe-rence Value	Distribution Type	Distribution Parameters		Mini-mum	Maxi-mum	x y w if given			
1	ODVPI	Korrekturfaktor für relative Geschwin-digkeit, Vertikale Leitungen	1	Polygonal Line			0.5	1.5	0.5 0	0.7 1	1.2 1	1.5 0
2	ODHPI	Korrekturfaktor für relative Geschwin-digkeit, Horizontale Leitungen	1	Polygonal Line			0.75	2.25	0.75 0	1 1	2 1	2.25 0
3	ODBUN	Korrekturfaktor für relative Geschwin-digkeit, Bündelgeometrie, Heizstabbün-del	1	Normal	0.84	0.28	0.3	1.5				
4	OHWFC	Einphasige Konvektion in Wasser (Dit-tus-Boelter, MC Eligot) - Korrekturfak-tor; alle Flächen mit Wärmeübertragung	1	Uniform	0.85	1.15	0.85	1.15				
5	OHWNC	Einphasige Naturkonvektion in Wasser - Korrekturfaktor; alle Flächen mit Wär-meübertragung	1	Uniform	0.85	1.15	0.85	1.15				
6	OHWNB	Blasensieden (modifizierte Chen Korre-lation) - Korrekturfaktor; alle Flächen mit Wärmeübertragung	1	Uniform	0.8	1.2	0.8	1.2				
7	IHTC1	Modell für Dampf-Tropfenkühlung: 1=modifizierte Dougall-Rohsenow / 2=Condie-Bengston IV Korrelation	1	Discrete			1	2	1 0.5	2 0.5		
8	OHWFB	Dampf-Tropfenkühlung: modifizierte Dougall-Rohsenow / Condie-Bengston IV Korrelation - Korrekturfaktor, Kern	1	Uniform	0.65	1.3	0.65	1.3				
9	OTMFB	Minimale Filmsiedetemperatur (Groene-veld-Stewart Korrelation) - Korrekturfak-tor, Kern	1	Uniform	0.99	1.3	0.99	1.3				
10	IHTC3	Modell für einphasige Zwangskonvek-tion in Dampf: 1=Dittus-Boelter II / 2=Mc Eligot	1	Discrete			1	2	1 0.5	2 0.5		

Par. No.	Parameter ID	Parameter Name	Refe-rence Value	Distribution Type	Distribution Parameters		Mini-mum	Maxi-mum	x y w if given			
11	OHVFC	Einphasige Konvektion in Dampf Dittus-Boelter II / Mc Eligot - Korrekturfaktor; alle Flächen, mit Wärmeübertragung	1	Uniform	0.8	1.25	0.8	1.25				
12	OTRNB	Kritische Heizflächenbelastung, Mini-malwert aus 3 Korrelationen - Korrek-turfaktor, Kern	1	Uniform	0.85	1.15	0.85	1.15				
13	OHWPB	Pool Filmsieden bei Naturkonvektion (Bromley Korrelation) – Korrekturfaktor, Kern	1	Uniform	0.75	1.25	0.75	1.25				
14	HTCLO	Wärmeverluste an die Umgebung, Au-ßenflächen des Primärkreislaufes und der Dampferzeuger	1	Uniform	1	7	1	7				
15	OMCON	Korrekturfaktor für Direktkondensation	1	Histogram			0.5	2	0.5 0.5	1 0.5	2	
16	ZB	Zahl der Blasen pro Einheitsvolumen, Primär- und Sekundärkreislauf	5E+09	Log. Trian-gular	5.E09		1.E08	1.E10				
17	ZT	Zahl der Tropfen pro Einheitsvolumen, Primär- und Sekundärkreislauf	5E+09	Log. Trian-gular	5.E09		1.E08	1.E10				
18	OADDI	Maximales spezifisches Volumen für Begrenzung der Verdampfungskorrela-tion, Primär- und Sekundärkreislauf	1	Uniform	0.2	1.2	.2	1.2				
19	SFFJ0	Korrekturfaktor für den Formwiderstand der Abstandshalter	1	Histogram			0.5	2	0.5 0.5	1 0.5	2	
20	SFFJ0O	Korrekturfaktor für den Formwiderstand der oberen Kernplatte	1	Histogram			0.75	1.5	0.75 0.5	1 0.5	1.5	
21	AKITAR	Korrekturfaktor für Reaktivitätstabelle als Dichtefunktion	1	Uniform	0.85	1.15	0.85	1.15				
22	AKITAF	Korrekturfaktor für Reaktivitätstabelle als Funktion der Brennstofftemperatur	1	Uniform	0.94	1.06	0.94	1.06				
23	AKITAZ	Korrekturfaktor für externe Reaktivität	1	Uniform	0.93	1.07	0.93	1.07				

Par. No.	Parameter ID	Parameter Name	Refe-rence Value	Distribution Type	Distribution Parameters		Mini-mum	Maxi-mum	x y w if given			
24	RPODC	Korrekturfaktor für Nachzerfallswärme	1	Polygonal Line			0.85	1	0.85 0	0.9 1	0.95 1	1 0
25	QROD0KBS	Korrekturfaktor Stableistung, heißer Brennstab	1	Uniform	0.98	1.02	.98	1.02				
26	WLFM	Korrektur der Wärmeleitfähigkeit des Brennstoffs	1	Uniform	0.88	1.12	.88	1.12				
27	DHLEV	Abweichung Druckhalterfüllstand vom Sollwert	0	Uniform	-0.1	0.1	-.1	.1				
28	EPS	Konvergenzkriterium	0.0004	Log. Trian-gular	0.001		.0001	0.01				
29	DPJMA	Additiver Term für den Druck im Sicher-heitsbehälter (berechnet mit CONDRU)	0	Uniform	-3000	1000	-3000	1000				
30	FD_AVV	Druckverlust (Beiwert) in der Düse (Wandreibung und Formverluste), FD-Abblaseregelventil	0.02	Polygonal Line			0.02	0.08	0.02 0	0.04 1	0.06 1	0.08 0
31	TODSP	Wassertemperatur in den Druckspei-chern	30	Uniform	20	40	20	40				
32	TURB	Turbulenzfaktor für Verdampfung bei kritischer Strömung, Bruch	20	Log. Normal	2.29	0.65	0	50				
33	CGHTWB	Maximal möglicher Wärmeübergangs-koeffizient für die untere Quenchfront, Brennelement	300000	Log. Uniform	1.E05	1.E06	1.E05	1.E06				
34	CGHTWT	Maximal möglicher Wärmeübergangs-koeffizient für die obere Quenchfront, Brennelement	200000	Log. Uniform	2.E04	1.E06	2.E04	1.E06				
35	FAKGJNA	Variation der JNA Pumpenkennlinie	1	Normal	1	0.1	0.9	1.1				

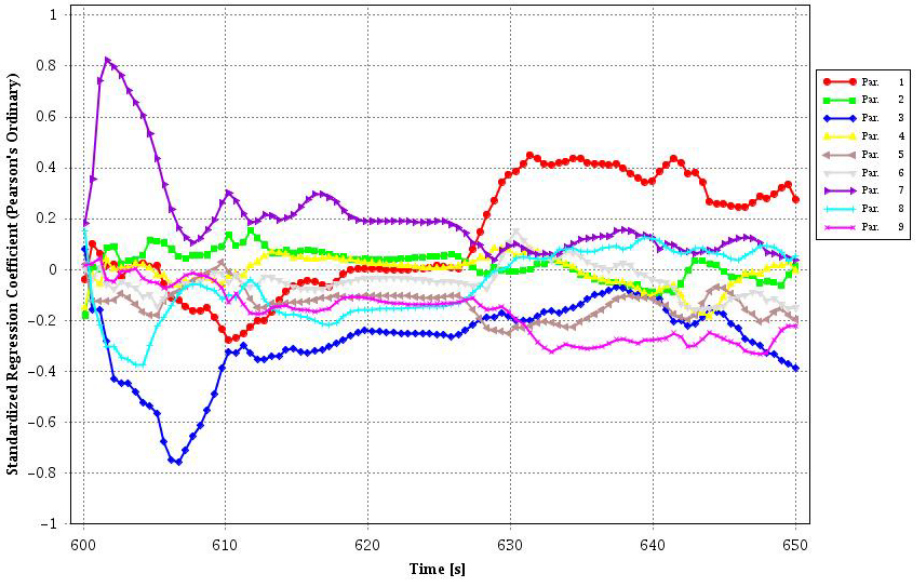

Abb. 4.3 Zeitabhängige individuelle Einflüsse der unsicheren Par. 1 – 9 aus Tab. 4.1 auf die Variation der maximalen Hüllrohrtemperatur (PCT); Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); Datenbasis: 60 Parameterkombinationen und von ATHLET berechnete PCT-Werte

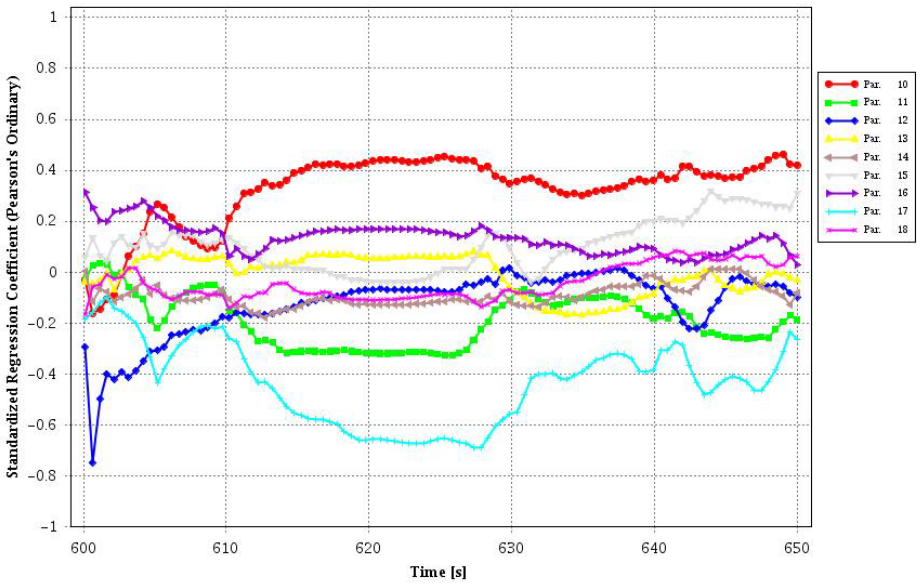

Abb. 4.4 Zeitabhängige individuelle Einflüsse der unsicheren Parameter 10 – 18 aus Tab. 4.1 auf die Variation der maximalen Hüllrohrtemperatur (PCT); Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); Datenbasis: 60 Parameterkombinationen und von ATHLET berechnete PCT-Werte

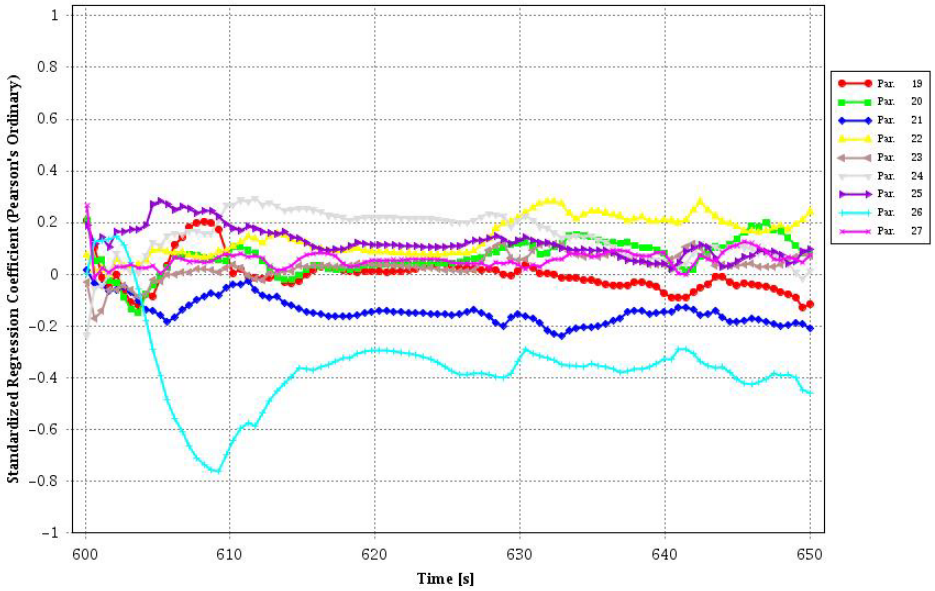

Abb. 4.5 Zeitabhängige individuelle Einflüsse der unsicheren Parameter 19 – 27 aus Tab. 4.1 auf die Variation der maximalen Hüllrohrtemperatur (PCT); Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); Datenbasis: 60 Parameterkombinationen und von ATHLET berechnete PCT-Werte

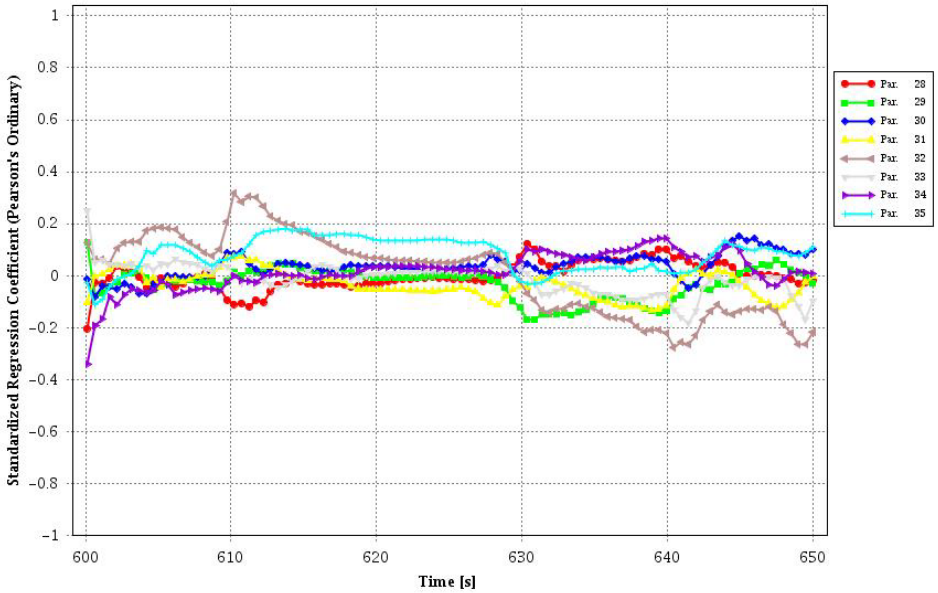

Abb. 4.6 Zeitabhängige individuelle Einflüsse der unsicheren Parameter 28 – 35 aus Tab. 4.1 auf die Variation der maximalen Hüllrohrtemperatur (PCT); Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); Datenbasis: 60 Parameterkombinationen und von ATHLET berechnete PCT-Werte

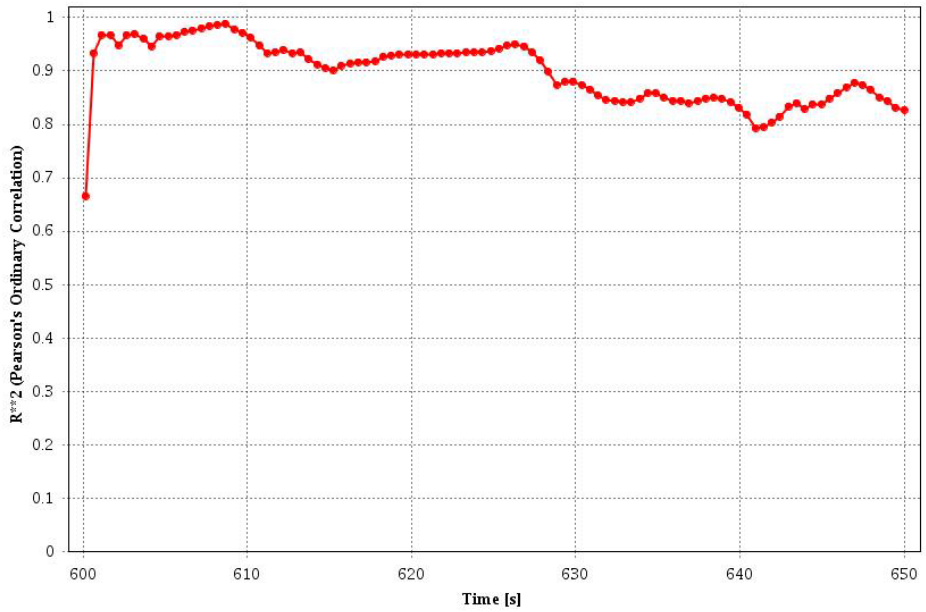

Abb. 4.7 Zeitlicher Verlauf des Bestimmtheitsmaßes als Ausdruck der Aussagekraft der Sensitivitätsindizes in den Abb. 4.1 bis Abb. 4.6

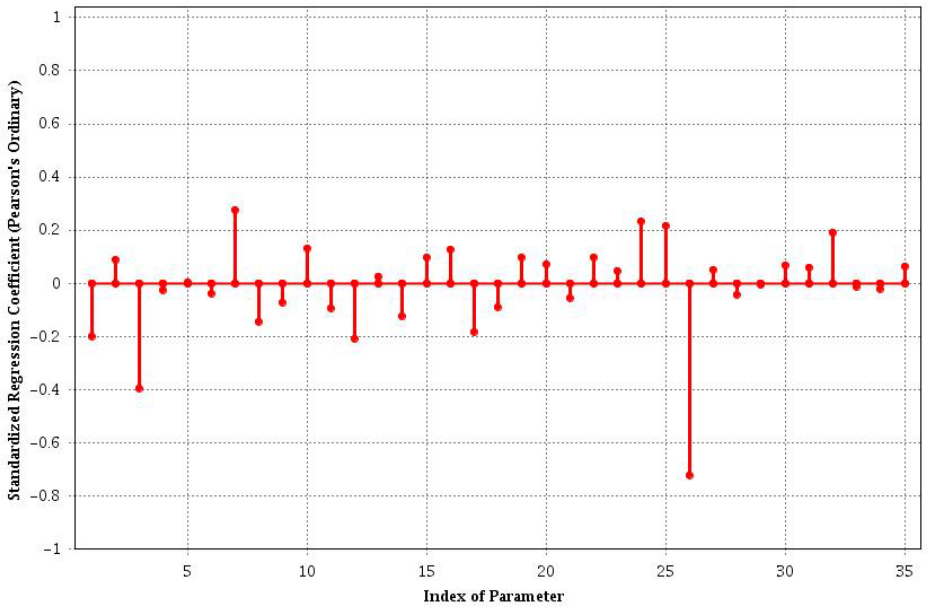

Abb. 4.8 Individuelle Einflüsse aller 35 Parameter auf die Variation der maximalen Hüllrohrtemperatur; Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); $R^2 = 0.98$; Datenbasis: 60 Parameterkombinationen und von ATHLET berechnete PCT-Werte

4.2 Anwendung des iterativen SuSSVR-Verfahrens

Im Folgenden werden die Parameter beschrieben, die in den Schritten des iterativen SuSSVR-Verfahrens (siehe Abschnitt 2.1.1) für den Anwendungsfall verwendet wurden.

4.2.1 Parametereinstellungen

(S1) – (S2)

Der anfängliche Pool von Trainingsdaten bestand aus den ersten n = 50 zufällig ausgewählten Parametervektoren $x \in R^m$, m = 35, und den zugehörigen PCT-Maximalwerten $r_0(x)$ aus der Unsicherheitsanalyse mit ATHLET (siehe Abschnitt 4.1). Jeder Parametervektor x im Trainingsdatenpool wurde in einen Vektor z von unabhängigen standardnormalverteilten Parametern transformiert.

(S3)

Jeder PCT-Wert $r(z)\ (= r_0(x))$ wurde in den LSF-Wert $g(z)$ transformiert, wobei $g(z) = r_{th} - r(z)$ und r_{th}= 1200 °C. Zusätzlich wurde auf $y = {(g(z) - \overline{y_0})}/{s_{y_0}}$ skaliert, wobei $\overline{y_0}$ der Mittelwert und s_{y_0} die Standardabweichung der $g(z)$-Werte im anfänglichen Trainingsdatenpool vom Umfang n = 50 sind.

(S4) – (S7)

Siehe Schritte (S4) – (S7) in Abschnitt 2.1.4.1

(S8)

Im Rahmen des umfangreichen SuS-Verfahrens wurden Stichproben von $nsus$ = 20000 Parametervektoren aus den Teilmengen des Parameterraums generiert. Die Wahrscheinlichkeit zur Definition der Teilmengen $\{D_i\}_{i=1}^{n_D-1}$ betrug p_0= 0.25 (Gl. (2.15) und Gl. (2.17)). Als Zufallszahlengenerator wurde der Zufallszahlengenerator PCG64 der frei verfügbaren Python-Programmbibliothek NumPy verwendet. Das SuS-Verfahren in Kombination mit dem robusten SVR-Metamodell wurde insgesamt 30 Mal angewendet; jedes Mal mit einem anderen Anfangswert (seed) für den Zufallszahlengenerator.

4.2.2 Ergebnisse

Das gesamte SuSSVR-Verfahren wurde automatisch mittels des hierfür entwickelten Algorithmus durchgeführt. Insgesamt waren n = 414 ATHLET-Läufe mit zufällig bzw. adaptiv ausgewählten Parameterkombinationen erforderlich bis das Abbruchkriterium des Verfahrens $(p_{swi} \leq 0.025 \text{ und } \lambda_P \leq 0.10)$ erreicht war, d. h. bis das trainierte SVR-Metamodell sowie der Schätzer für die Wahrscheinlichkeit des kritischen Bereichs als robust genug bewertet wurden (Schritt (S7) in Abschnitt 4.2.1). In dieser Gesamtzahl an ATHLET-Läufen sind auch die ersten 50 ATHLET-Läufe aus der Unsicherheits- und Sensitivitätsanalyse enthalten, deren Eingabeparametervektoren und PCT-Werte im anfänglichen Trainingsdatenpool bereitgestellt wurden. Von den 50 ATHLET-Läufen lieferte ein Lauf einen PCT-Wert von 1203 °C (siehe Abschnitt 4.1).

Das robuste SVR-Metamodell, das auf der Basis der Ergebnisse aus den 414 Läufen trainiert wurde, wurde eingesetzt, um die Wahrscheinlichkeit für den kritischen Bereich zu ermitteln. Die Ergebnisse hierzu sind in Abschnitt 4.2.2.1 beschrieben. Neben dieser Wahrscheinlichkeit liefert das SuSSVR-Verfahren zusätzlich eine umfangreiche Stichprobe von Parameterkombinationen aus dem kritischen Bereich. Diese kann dazu verwendet werden, den kritischen Bereich genauer zu spezifizieren. Die Ergebnisse dieser Auswertung werden in Abschnitt 4.2.2.2 vorgestellt. In Abschnitt 4.2.2.3 werden die Ergebnisse der Auswertung bzgl. der 414 Eingabevariationen und der zugehörigen von ATHLET berechneten PCT-Werte präsentiert.

Da die Anzahl der erforderlichen Rechenläufe im SuSSVR-Verfahren auch von der Anzahl der zu berücksichtigenden unsicheren Parameter abhängt, wurde das SuSSVR-Verfahren mit 12 als besonders wichtig beurteilten Parametern ein weiteres Mal durchgeführt. In Abschnitt 4.2.2.4 werden die Ergebnisse aus der Anwendung des SuSSVR-Verfahrens nach der Dimensionsreduktion im Parameterraum vorgestellt.

4.2.2.1 Eintrittswahrscheinlichkeit des kritischen Bereichs

Die Eintrittswahrscheinlichkeit $P(D^{35})$ für den kritischen Bereich $D^{35} = \{x \in R^{35} \mid r_0(x) > 1200\}$ (siehe Schritt (S3) in Abschnitt 4.2.1), die mit dem SuS-Verfahren und dem robusten SVR-Metamodell ermittelt wurde (Schritt (S8) in Abschnitt 4.2.1), liegt zwischen 8.163E-03 und 9.363E-03. Der zugehörige Variationskoeffizient liegt bei 0.022.

Eine Wahrscheinlichkeitsschätzung in dieser Größenordnung (d. h. von ~0.01) erhält man z. B., wenn 100 Läufe im Rahmen einer einfachen MC-Simulation durchgeführt werden und davon ein Lauf einen PCT-Wert über 1200 °C aufweist. In diesem Fall ist der Schätzer aber mit einem großen Fehler behaftet, d. h. z. B., dass der zugehörige Variationskoeffizient 0.995 beträgt. Um den sehr niedrigen Variationskoeffizienten von 0.022 zu erhalten, müssten insgesamt 204546 Läufe durchgeführt werden (siehe Abschnitt 2.1.3.3). Ein weniger strenges Kriterium für den Wahrscheinlichkeitsschätzer ist z. B. die klassische obere 95 %-Konfidenzgrenze nach Pearson und Clopper /CLO 34/. Sie gibt den Wert an, der mit einer Konfidenzwahrscheinlichkeit von 95 % nicht überschritten wird. Damit diese Grenze ungefähr dem Wert 0.01 entspricht, müssen ~472 Simulationsläufe im Rahmen einer einfachen MC-Simulation durchgeführt werden und einer dieser Läufe darf einen PCT-Wert über 1200 °C liefern.

Nichtsdestotrotz ist die Anzahl der Läufe mit n = 414 relativ hoch für ein adaptives Samplingverfahren bzgl. eines kritischen Bereichs, der mit einer Wahrscheinlichkeit von ~0.01 vorkommt. Bei der Anwendung des SuSSVR-Verfahrens auf das biologische Dosismodell mit insgesamt sechs Parametern (Abschnitt 2.1.4) wurden insgesamt nur n = 215 Rechenläufe benötigt, um einen sehr seltenen kritischen Parameterbereich mit einer Eintrittswahrscheinlichkeit von ~3.5E-06 zu ermitteln.

Grund für die relativ hohe Zahl von n = 414 Rechenläufen ist die hohe Zahl von m = 35 unsicheren Parametern, die für die Anpassung des SVR-Modells berücksichtigt wurden. Je mehr Parameter (Einflussfaktoren) einzubeziehen sind, desto mehr Beobachtungen (Parametervektoren und zugehörige PCT-Werte) sind erforderlich bis ein robustes Regressionsmodell erstellt ist. Dies legt den Schritt nahe, vor der Anwendung des SuSSVR-Verfahrens, eine Sensitivitätsanalyse durchzuführen und nur die dabei ermittelten wichtigsten Parameter im Verfahren zu berücksichtigen. In Abschnitt 4.2.2.4 werden die Ergebnisse aus der Anwendung des SuSSVR-Verfahrens nach einer Dimensionsreduktion im Parameterraum vorgestellt und diskutiert.

Im Vergleich zur einfachen MCS bietet das SuSSVR-Verfahren zusätzlich die Möglichkeit, den kritischen Parameterbereich genauer zu analysieren und wesentliche Charakteristika zu spezifizieren. Mehr dazu ist im nächsten Abschnitt zu finden.

4.2.2.2 Kritischer Parameterbereich

In Schritt (S8) des SuSSVR-Verfahrens wurden SuS-Stichproben aus vier Parameterbereichen (Regionen 0 – 3) generiert. Region 0 entspricht dem gesamten Parameterraum, die Regionen 1 – 3 sind ineinandergeschachtelte Teilbereiche des Parameterraums, wobei Region 3 als kleinster Teilbereich dem kritischen Parameterbereich entspricht. Die Stichproben aus den einzelnen Regionen umfassen jeweils 11479 – 20000 Kombinationen von Parameterwerten. Für den kritischen Parameterbereich (Region 3) wurden 11479 unterschiedliche Parametervektoren generiert.

Die Abb. 4.9 bis Abb. 4.13 zeigen für die einzelnen Parameter aus Tab. 4.1 die empirischen Verteilungen in Form der Kerndichten in den verschiedenen Regionen. Jede Zeile in den Abbildungen ist einem Parameter zugeordnet; die Spalten beziehen sich auf die Regionen 0 – 3. Große Unterschiede zwischen den Verteilungen eines Parameters in den einzelnen Regionen weisen auf einen deutlichen Einfluss dieses Parameters auf den von ATHLET berechneten PCT-Wert hin. Die Gegenüberstellung der Verteilungen eines Parameters bzgl. der Regionen 0 (gesamter Parameterraum) und 3 (kritischer Parameterbereich) zeigt die Bedeutung dieses Parameters für die Festlegung des kritischen Bereichs. Kommt es in Region 3 zu einer deutlichen Verlagerung der Verteilung eines Parameters im Vergleich zur Region 0, so ist ein Parameter als besonders wichtig einzustufen. Wenn sich dagegen die Verteilungen eines Parameters kaum unterscheiden, spielen die Werte dieses Parameters für die Festlegung des kritischen Bereichs keine Rolle. Demgemäß sind folgende neun Parameter als wichtig einzuordnen: ODVPI (Nr. 1 aus Tab. 4.1), ODBUN (Nr. 3), IHTC1 (Nr. 7), OHWFB (Nr. 8), OHVFC (Nr. 11), OTRNB (Nr. 12), ZT (Nr. 17), QROD0KBS (Nr. 25) und WLFM (Nr. 26). Besonders wichtig ist der Parameter WLFM.

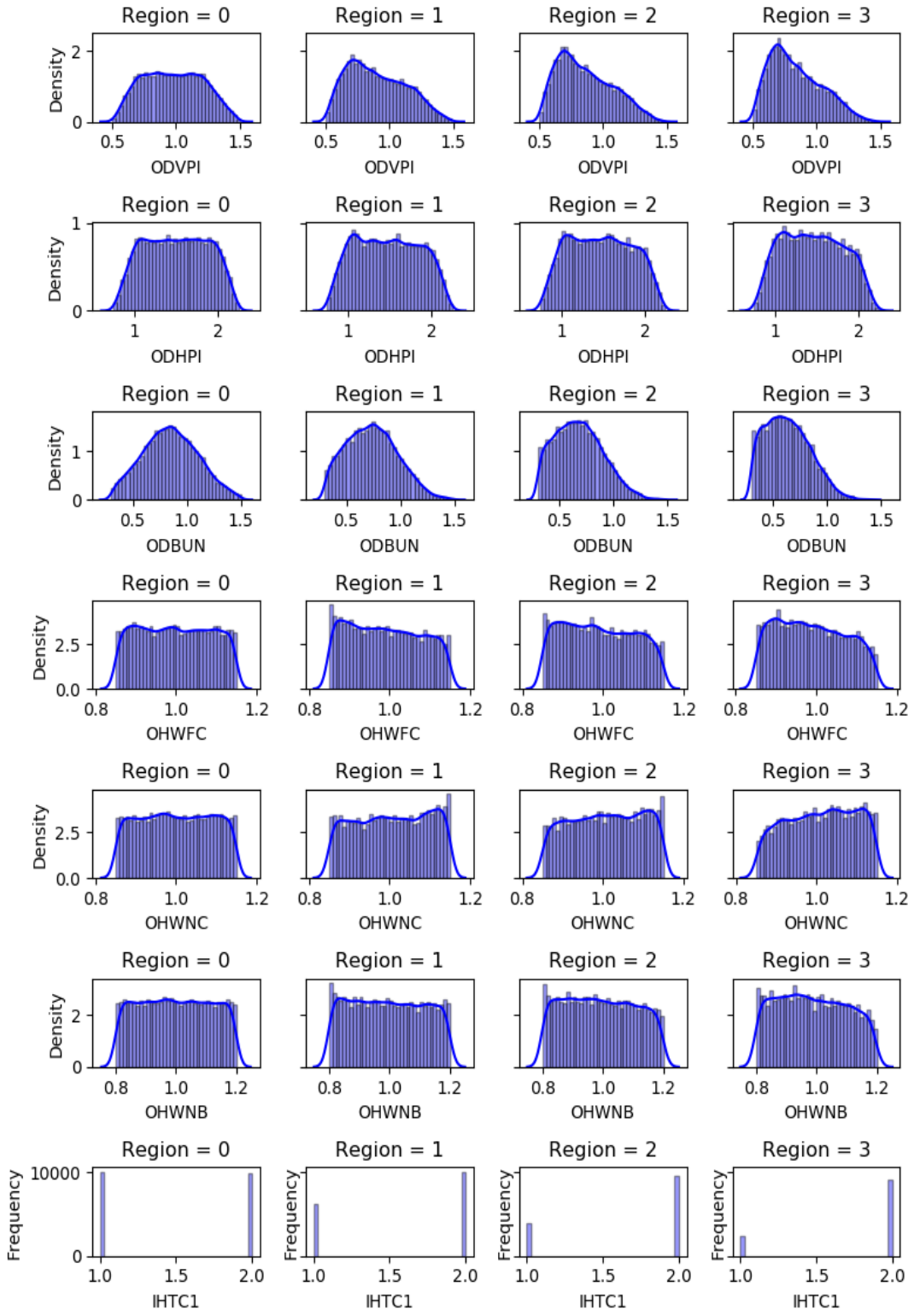

Abb. 4.9 Verteilungen der Parameter 1 – 7 (Tab. 4.1) in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 3 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 3 = kleinste Region = kritischer Parameterbereich

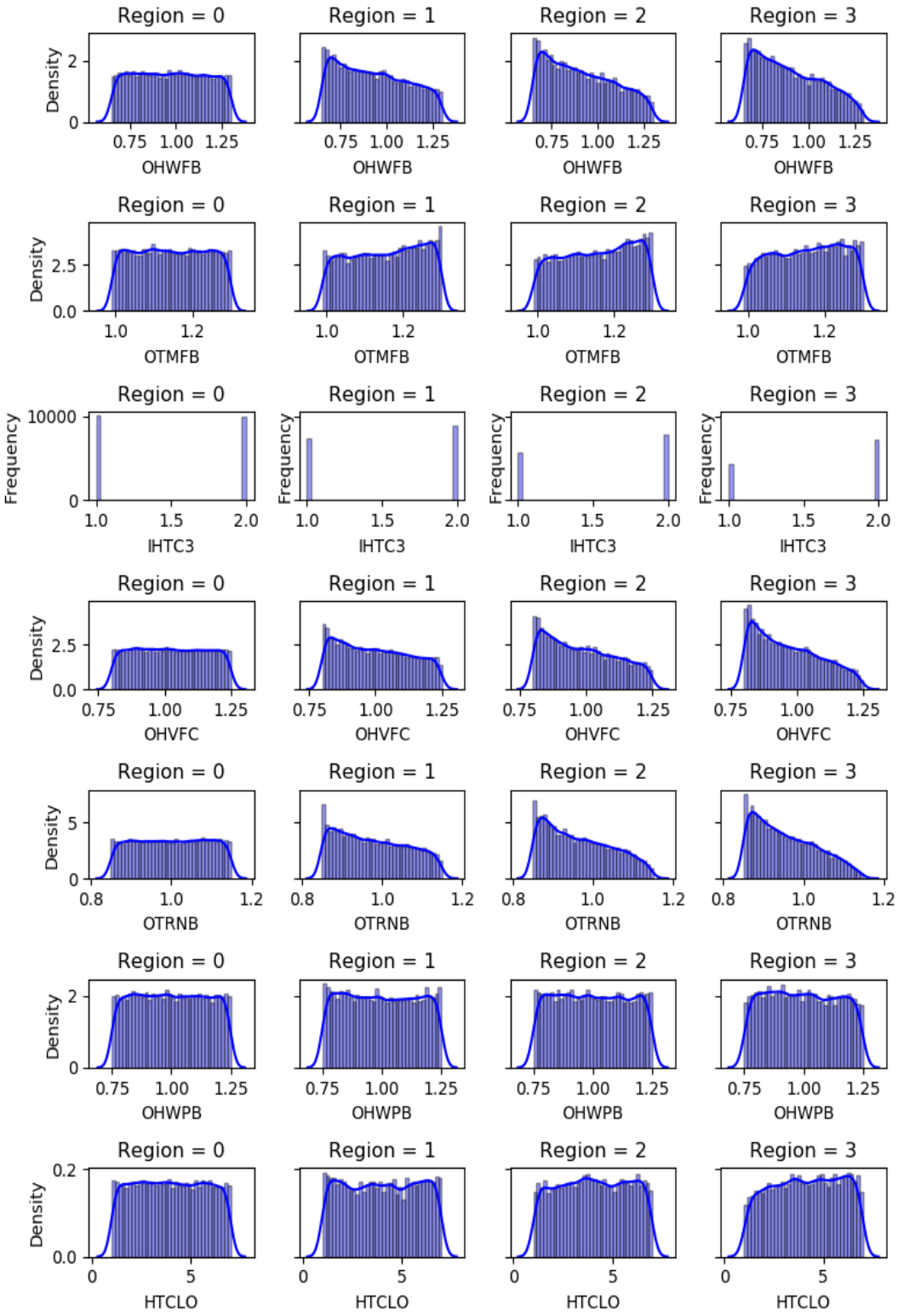

Abb. 4.10 Verteilungen der Parameter 8 – 14 (Tab. 4.1) in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 3 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 3 = kleinste Region = kritischer Parameterbereich

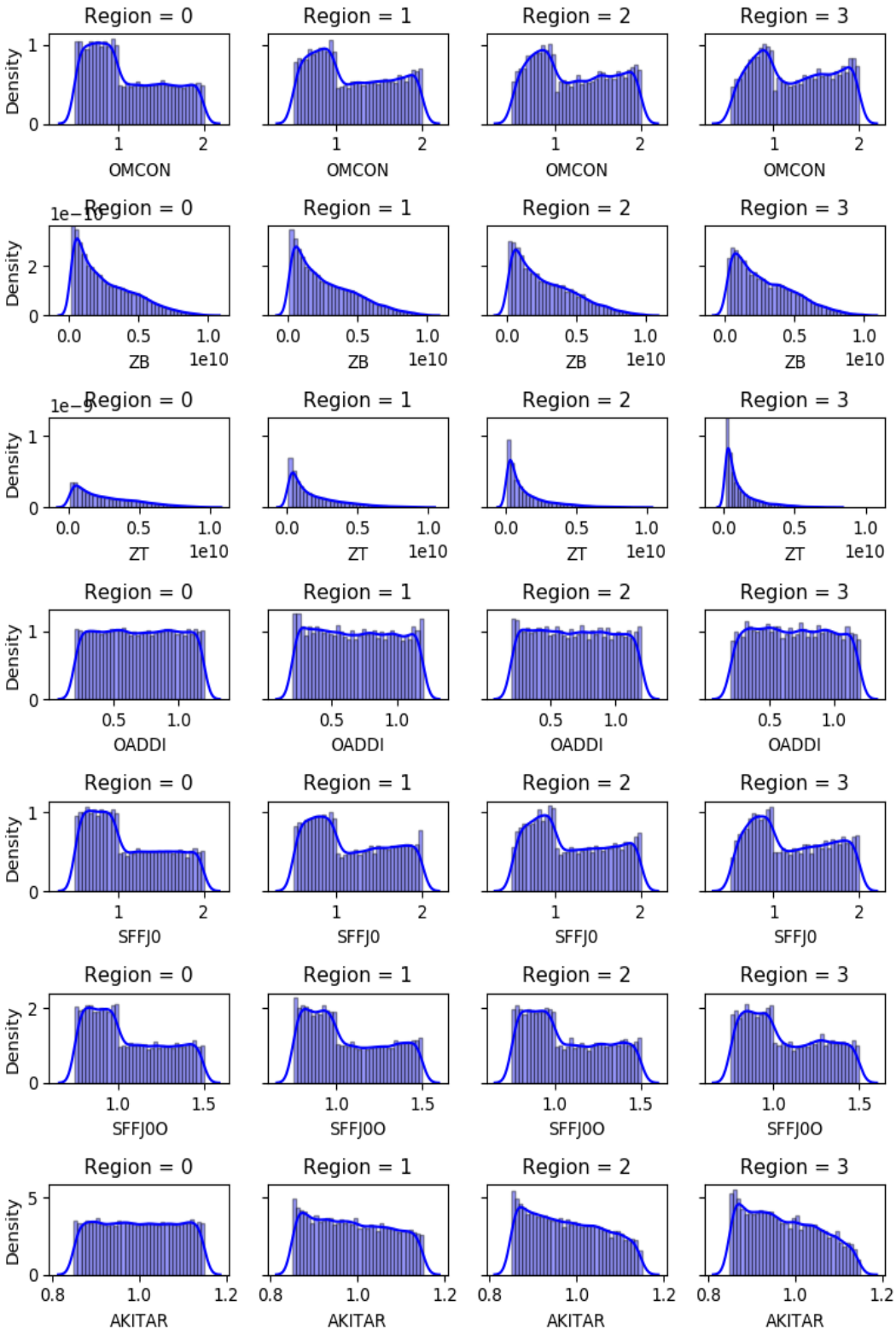

Abb. 4.11 Verteilungen der Parameter 15 – 21 (Tab. 4.1) in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 3 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 3 = kleinste Region = kritischer Parameterbereich

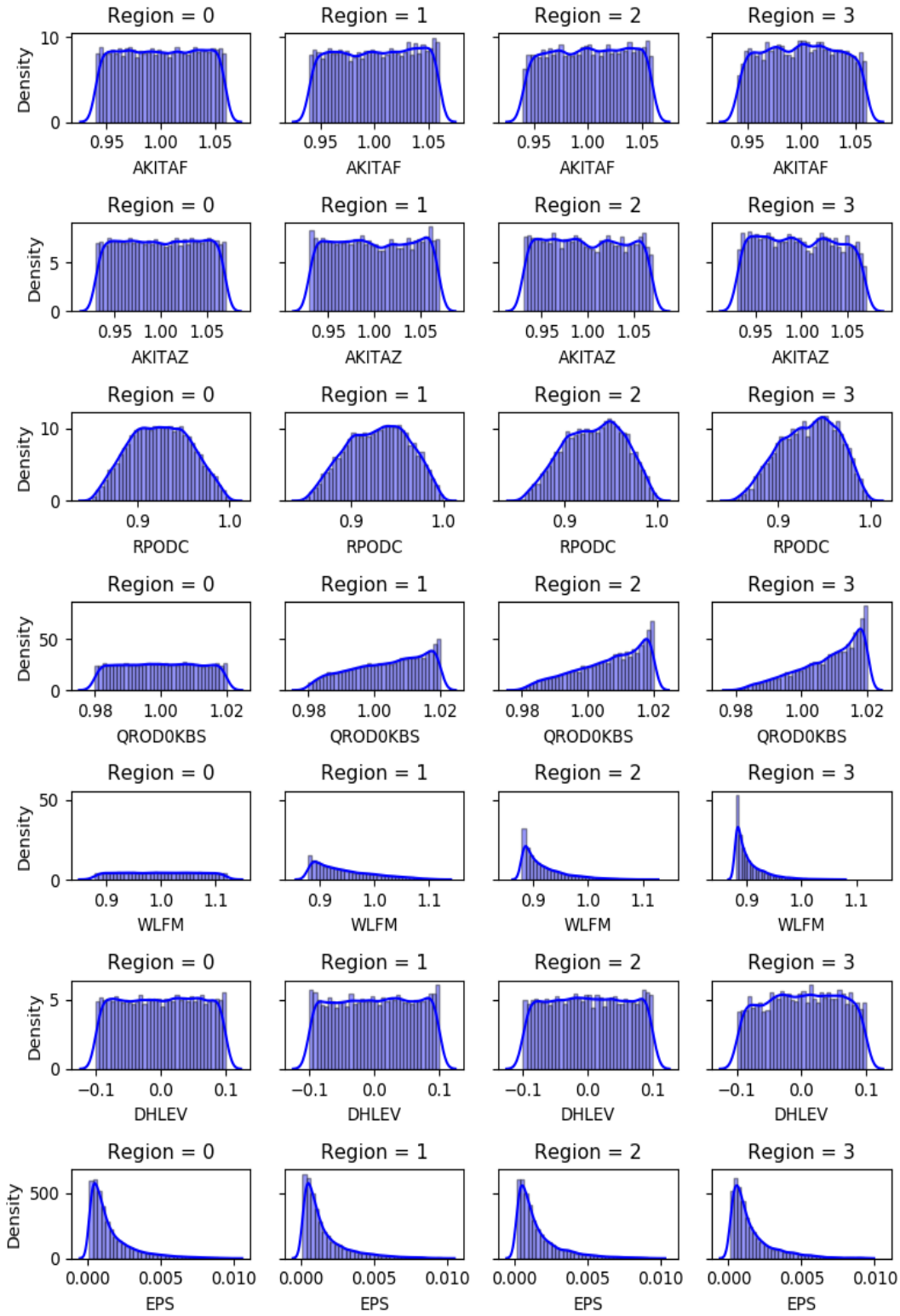

Abb. 4.12 Verteilungen der Parameter 22 – 28 (Tab. 4.1) in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 3 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 3 = kleinste Region = kritischer Parameterbereich

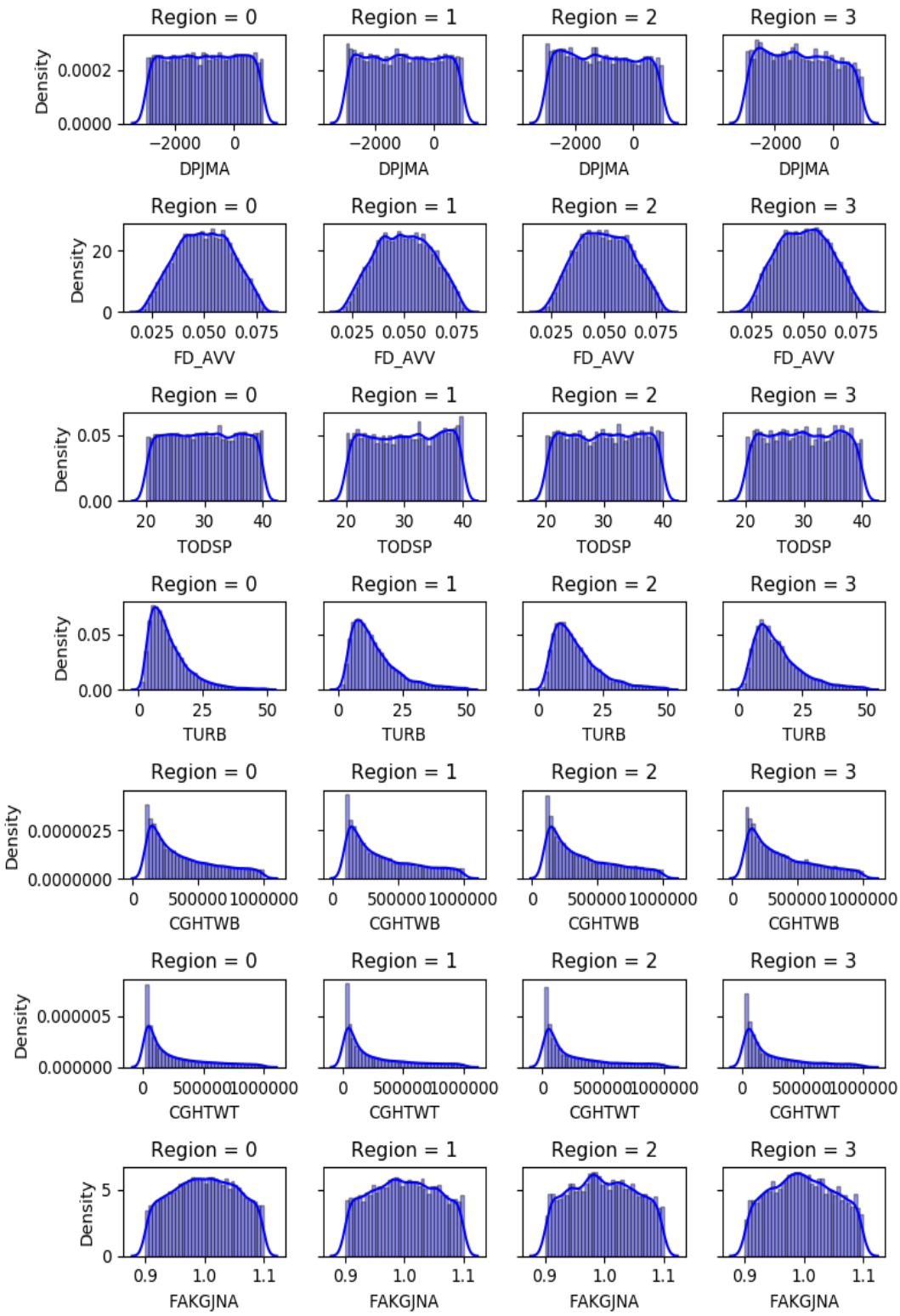

Abb. 4.13 Verteilungen der Parameter 29 – 35 (Tab. 4.1) in verschiedenen Regionen. Region 0 = Parameterraum, Regionen 1 – 3 = ineinander geschachtelte Teilbereiche des Parameterraums mit Region 3 = kleinste Region = kritischer Parameterbereich

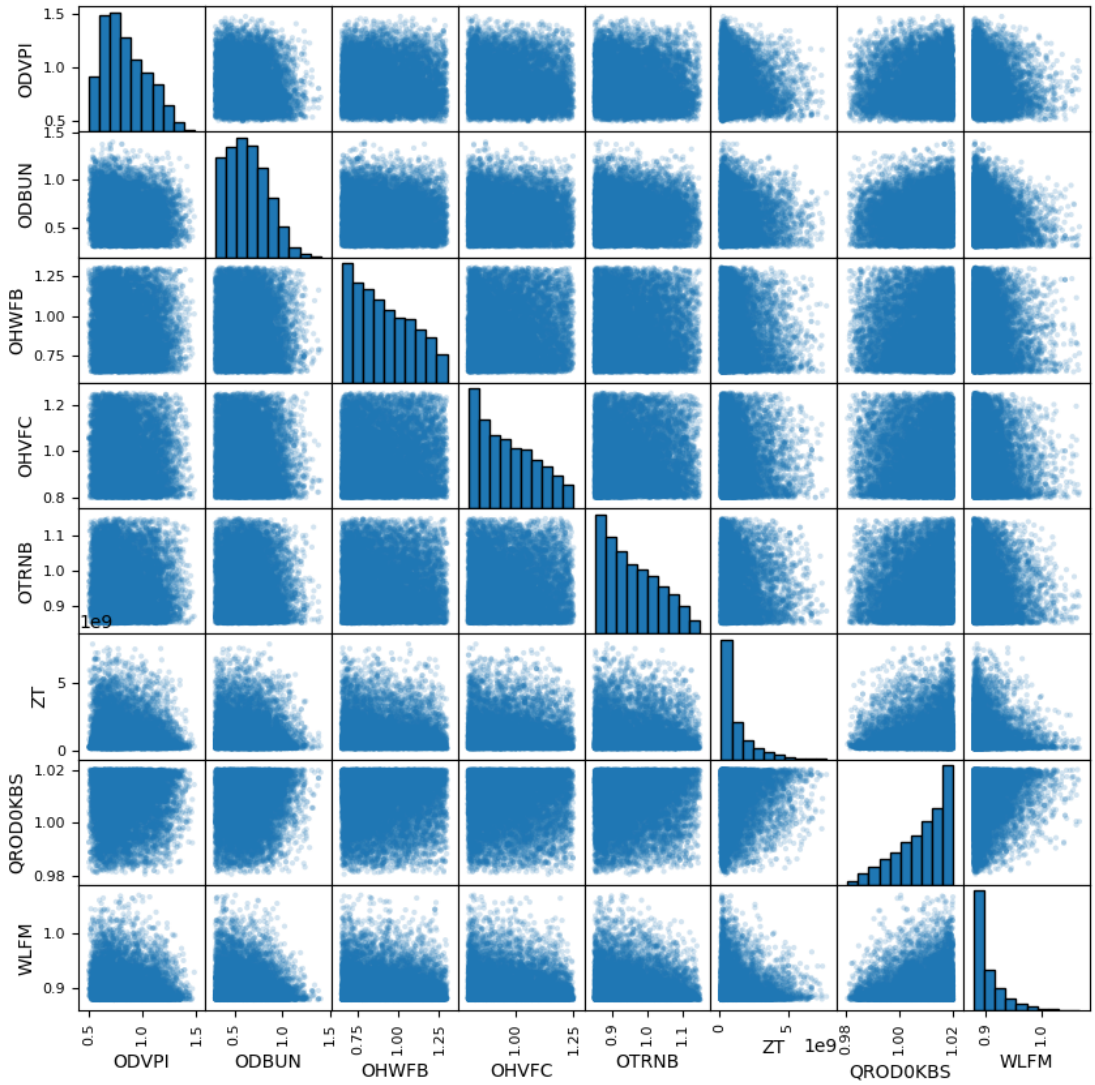

Abb. 4.14 Scatter Plots und Verteilungen in der kritischen Region für die Parameter mit den größten Unterschieden in den regionsbezogenen Verteilungen der Abb. 4.9 bis Abb. 4.13

Abb. 4.14 zeigt die Assoziationen zwischen den wichtigen Parametern in Form von Scatterplots sowie deren Häufigkeitsverteilungen (Histogramme) im kritischen Bereich. Nicht berücksichtigt wurden der Parameter IHTC1, dessen Assoziationen mit den anderen Parametern vernachlässigbar sind. Die Assoziationen geben Auskunft darüber, welche Werte eines Parameters zusammen mit welchen Werten eines anderen Parameters im kritischen Bereich vorkommen. Für den wichtigsten Parameter WLFM fällt auf, dass hauptsächlich Werte < 1. in der kritischen Region liegen. Wenn hohe Werte (> 1.) dieses Parameters in der kritischen Region liegen, so kommen diese vor zusammen mit hohen Werten des Parameters QROD0KBS (> ~1.) sowie niedrigeren Werten der Parameter ZT (< ~1.E09), OTRNB (< ~1.1), ODBUN (< ~1.) und ODVPI (< ~1.2). Bei Parameter

QROD0KBS liegen mit großer Wahrscheinlichkeit die hohen Werte in der kritischen Region. Kleinere Werte kommen tendenziell vor zusammen mit niedrigen Werten der Parameter WLFM und ZT.

4.2.2.3 Auswertung der Variationsläufe mit ATHLET

In diesem Abschnitt werden die Ergebnisse aus der Auswertung der Daten aus den n = 414 Variationsläufen mit ATHLET dargestellt.

Die berechneten PCT-Werte aller 414 ATHLET-Läufe lagen zwischen 1087.21 °C und 1243.95 °C. Bei 318 Läufen wurden PCT-Werte über 1200 °C berechnet. Die Verteilung der PCT-Werte bzgl. der 414 Läufe ist in Abb. 4.15 zu sehen.

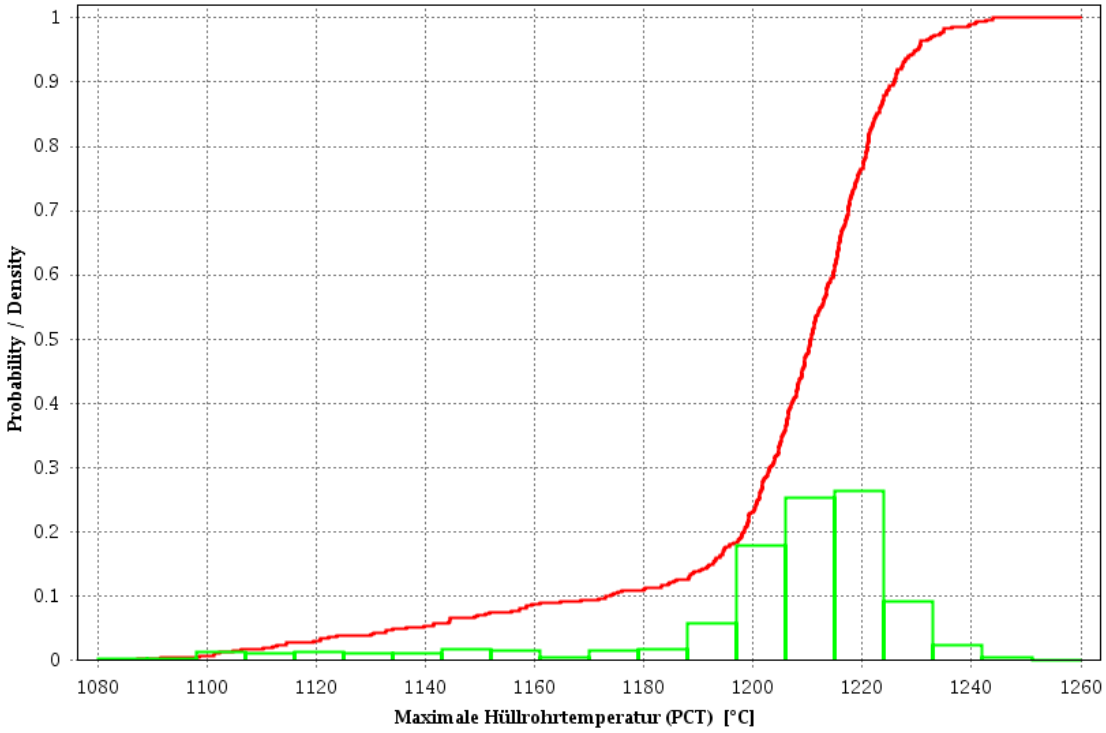

Abb. 4.15 Empirische Wahrscheinlichkeitsverteilung für die maximale Hüllrohrtemperatur (PCT) aus 414 ATHLET-Läufen des adaptiven SuSSVR-Verfahrens

In Abb. 4.16 sind die Sensitivitätsindizes (standardisierte Regressionskoeffizienten (Pearson)) für den im Zeitverlauf erreichten PCT-Maximalwert und die einzelnen Parameter dargestellt. Der dazugehörige R^2-Wert von 0.95 weist auf eine hohe Aussagekraft der Sensitivitätsindizes hin. Wie der Abbildung zu entnehmen ist, ist Parameter Nr. 26 (WLFM, Korrekturfaktor der Wärmeleitfähigkeit des Brennstoffs) bei Weitem der wichtigste Parameter. Weitere wichtige Parameter sind die Parameter mit den Nr. 1 (ODVPI,

Korrekturfaktor für relative Geschwindigkeit, Vertikale Leitungen), 3 (ODBUN, Korrekturfaktor für die relative Geschwindigkeit, Bündelgeometrie, Heizstabbündel), 7 (IHTC1, Modell für Dampf-Tropfenkühlung), 17 (ZT, Zahl der Tropfen pro Einheitsvolumen, Primär- und Sekundärkreislauf) und 25 (QROD0KBS, Korrekturfaktor Stableistung, heißer Brennstab). Auffällig sind außerdem die Parameter mit den Nummern 8 (OHWFB, Dampf-Tropfenkühlung), 10 (IHTC3, Modell für einphasige Zwangskonvektion in Dampf), 11 (OHVFC, Einphasige Konvektion in Dampf), 12 (OTRNB, Kritische Heizflächenbelastung) und 32 (TURB, Turbulenzfaktor für Verdampfung bei kritischer Strömung, Bruch).

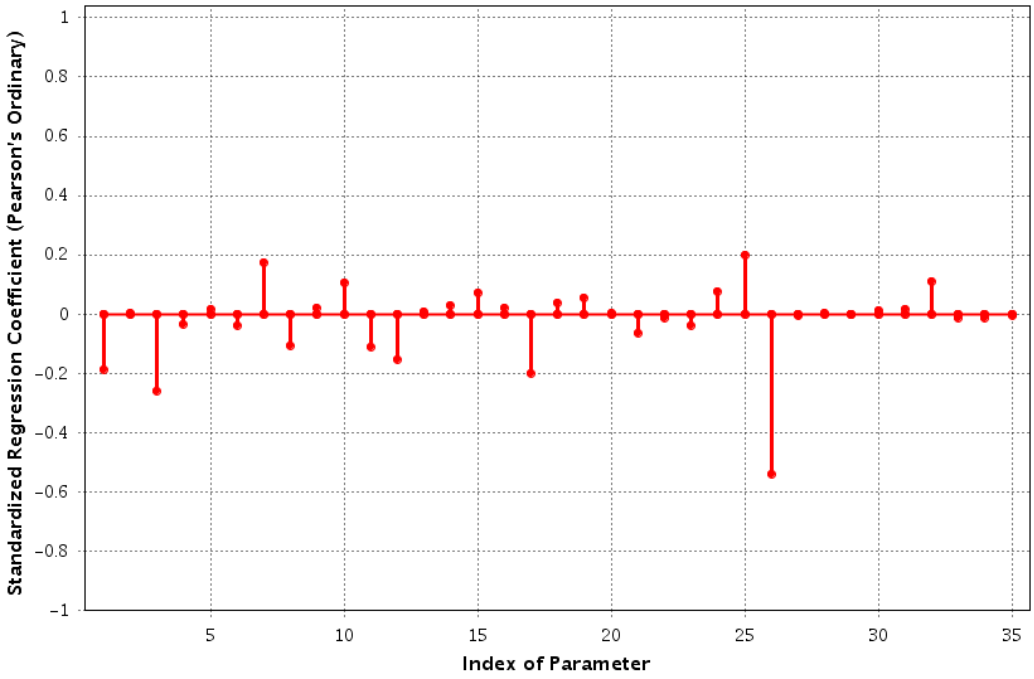

Abb. 4.16 Individuelle Einflüsse aller 35 Parameter auf die Variation der maximalen Hüllrohrtemperatur; Sensitivitätsindex: standardisierter Regressionskoeffizient (Pearson); $R^2 = 0.95$; Datenbasis: 414 Parameterkombinationen und von ATHLET berechneten PCT-Werte aus dem SuSSVR-Verfahren

Abb. 4.17 zeigt die Assoziationen für die maximale Hüllrohrtemperatur und ausgewählte Parameter in Form von Scatterplots sowie die zugehörigen Häufigkeitsverteilungen (Histogramme) im kritischen Bereich (PCT > 1200 °C). Für einen Vergleich der Ergebnisse aus der Anwendung des SVR-Metamodells und aus der Anwendung von ATHLET wurden dieselben Parameter wie in Abb. 4.15 ausgewählt. Datenbasis sind 318 von 414 Parameterkombinationen aus dem kritischen Bereich und die von ATHLET berechneten Werte für die maximale Hüllrohrtemperatur. Es fällt auf, dass die Form der Verteilungen bis auf eine Ausnahme (Parameter OHVFC) gut übereinstimmen. Die Assoziationen

stimmen häufig tendenziell überein, sind aber bei der kleineren Datenbasis von n = 318 noch nicht so ausgeprägt.

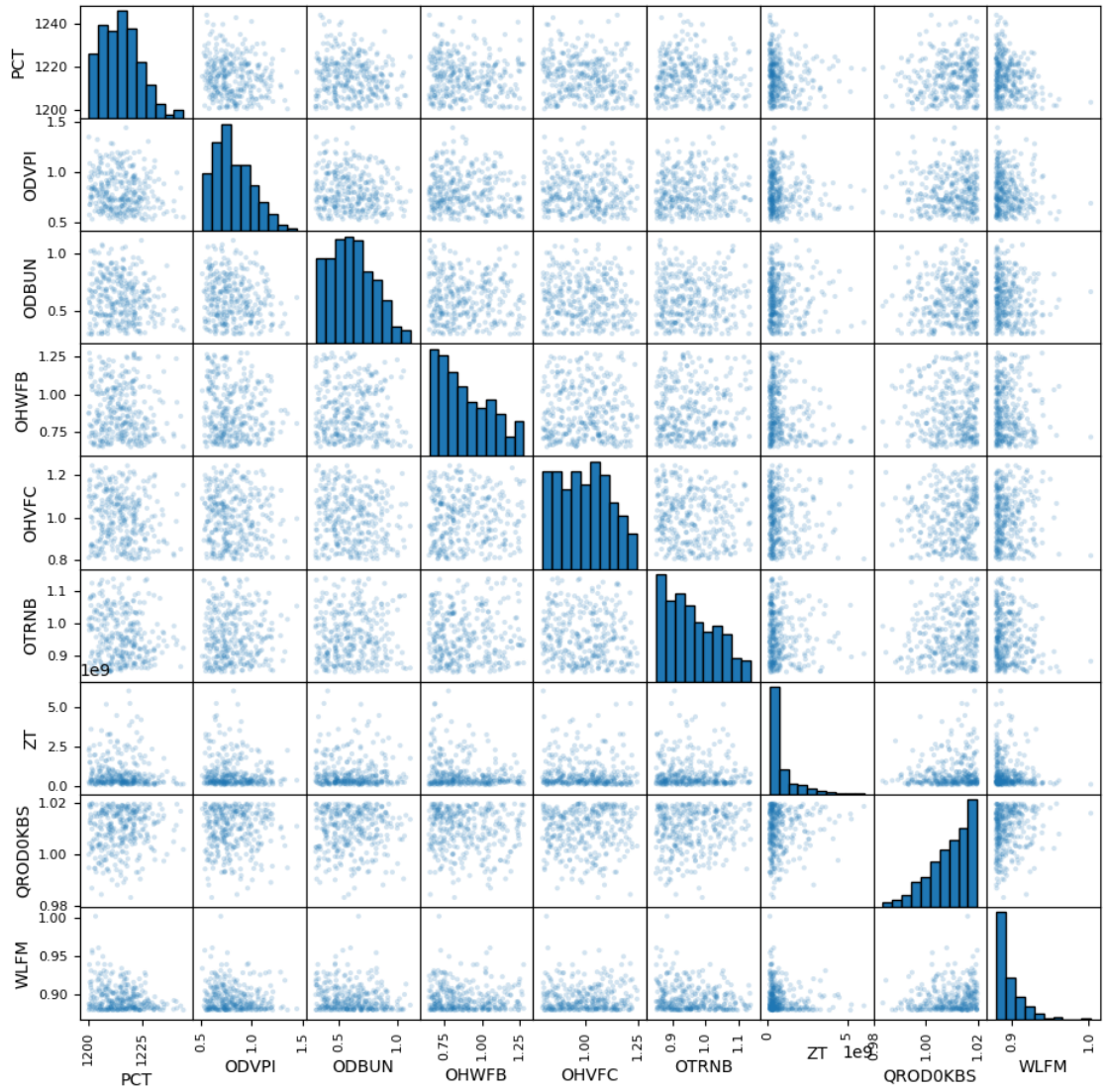

Abb. 4.17 Scatter Plots und Verteilungen in der kritischen Region für die maximale Hüllrohrtemperatur und ausgewählte Parameter (vgl. Abb. 4.14). Datenbasis: 318 (von insgesamt 414) Parameterkombinationen aus der kritischen Region und die von ATHLET berechneten Werte für die maximale Hüllrohrtemperatur aus dem SuSSVR-Verfahren

4.2.2.4 Eintrittswahrscheinlichkeit des kritischen Bereichs nach einer Dimensionsreduktion im Parameterraum

Nach einer Gegenüberstellung der Ergebnisse, die in den Abschnitten 4.1, 4.2.2.2 und 4.2.2.3 vorgestellt wurden, wurden Variationen der folgenden 12 Parameter für eine weitere Anwendung des SuSSVR-Verfahrens ausgewählt: ODVPI (Nr. 1 aus Tab. 4.1)', ODBUN (Nr. 3), IHTC1 (Nr. 7), OHWFB (Nr. 8), IHTC3 (Nr. 10), OHVFC (Nr. 11), OTRNB (Nr. 12), ZT (Nr. 17), RPODC (Nr. 24), QROD0KBS (Nr. 25), WLFM (Nr. 26) und TURB (Nr. 32). Die übrigen Parameter aus Tab. 4.1 wurden auf ihre jeweiligen Referenzwerte gesetzt (siehe Spalte ‚Reference Value' in Tab. 4.1).

Das SVR-Metamodell wurde auf der Basis von zufällig bzw. adaptiv ausgewählten Wertekombinationen der oben erwähnten 12 Parameter und den zugehörigen PCT-Maximalwerten trainiert. Der anfängliche Pool von Trainingsdaten bestand auf der Parameterseite aus n = 50 zufällig ausgewählten Parametervektoren $x \in R^m$ mit m = 12 (statt m = 35 wie in den vorherigen Abschnitten).

Insgesamt waren n = 195 ATHLET-Läufe mit unterschiedlichen Parameterkombinationen erforderlich bis das Abbruchkriterium des Verfahrens ($p_{swi} \leq 0.025$ und $\lambda_P \leq 0.10$) erreicht war, d. h. bis das trainierte SVR-Metamodell mit 12 Parametern sowie der Schätzer für die Wahrscheinlichkeit des kritischen Bereichs als robust genug bewertet wurden (Schritt (S7) in Abschnitt 2.1.4.1). In dieser Gesamtzahl n sind auch die 50 Parameterkombinationen und PCT-Werte des anfänglichen Trainingsdatenpools enthalten.

Die Eintrittswahrscheinlichkeit $P(D^{12})$ für den kritischen Bereich $D^{12} = \{x \in R^{12} \,|r_0(x) > 1200\,°C\}$, die mit dem SuS-Verfahren und dem robusten SVR-Metamodell ermittelt wurde, liegt zwischen 2.519E-02 und 2.898E-02. Der zugehörige Variationskoeffizient liegt bei 0.019.

Die Eintrittswahrscheinlichkeit ist ungefähr um den Faktor 3.1 höher als die Eintrittswahrscheinlichkeiten, die mit dem vollen Satz von 35 Parametern ermittelt wurden (siehe Abschnitt 4.2.2.1). Dafür kann die im folgenden Abschnitt beschriebene Begründung gegeben werden.

4.2.2.5 Probabilistische Begründung zu den Effekten der Dimensionsreduktion

Die Wahrscheinlichkeit $P(D^{12})$ bezieht sich auf den zwölf-dimensionalen Parameterraum R^{12} als Grundgesamtheit, wobei die 23 weniger wichtigen Parameter Par^{23} mit ihren Referenzwerten vertreten sind, d. h. $P(D^{12}) = P(D^{12} \cap R^{12}|(Par^{23} = ref^{23}))$. Die in Abschnitt 4.2.2.1 beschriebene Wahrscheinlichkeit $P(D^{35})$ für den kritischen Bereich bezieht sich dagegen auf den 35-dimensionalen Parameterraum R^{35} als Grundgesamtheit mit $R^{35} = R^{12} \times R^{23}$, d.h. $P(D^{35}) = P(D^{35} \cap R^{35}) = P(D^{35} \cap (R^{12} \times R^{23}))$.

Die Wahrscheinlichkeit $P(D^{12} \cap R^{35})$ für den Bereich D^{12} ist im Parameterraum R^{35} kleiner oder gleich als die Wahrscheinlichkeit $P(D^{12})$ im Parameterraum R^{12}:

$$
\begin{aligned}
P(D^{12} \cap R^{35}) = \; & P(D^{12} \cap R^{12} \times (Par^{23} = ref^{23}) \cap R^{23}) \\
= \; & \frac{P(D^{12} \times (Par^{23} = ref^{23}))}{P(Par^{23} = ref^{23})} \cdot P(Par^{23} = ref^{23}) \\
= \; & P(D^{12}|Par^{23} = ref^{23}) \cdot P(Par^{23} = ref^{23}) \\
= \; & P(D^{12}) \cdot P(Par^{23} = ref^{23})
\end{aligned}
\tag{4.1}
$$

Es besteht Gleichheit zwischen den beiden Wahrscheinlichkeiten $P(D^{12} \cap R^{35})$ und $P(D^{12})$, wenn $P(Par^{23} = ref^{23}) = 1$, d. h. wenn die Referenzwerte der 23 weniger wichtigen Parameter alle möglichen Werte dieser Parameter repräsentieren. Das ist im Anwendungsbeispiel aber nicht der Fall, weil dafür die beiden Wahrscheinlichkeiten $P(D^{12})$ und $P(D^{35})$ für den kritischen Bereich übereinstimmen müssten (siehe Gl. (4.2)).

Im Parameterraum R^{35} ist der kritische Bereich D^{12} ein Teilbereich des kritischen Bereichs D^{35}, d.h. $(D^{12} \cap R^{35}) \subseteq (D^{35} \cap R^{35})$, und daraus folgt:

$$
P(D^{12} \cap R^{35}) = \; P(D^{12}) \cdot P(Par^{23} = ref^{23}) \leq P(D^{35}) = \; P(D^{35} \cap R^{35}) \tag{4.2}
$$

Im Anwendungsbeispiel erhält man mit $P(D^{12}) = 0.02898$ und $P(D^{35}) = 0.009363$ $P(Par^{23} = ref^{23}) \leq 0.323$.

Bei statistischer Unabhängigkeit zwischen den 23 weniger wichtigen Parametern gilt:

$$P(Par^{23} = ref^{23}) = \prod_{i=1}^{23} P(Par_i = ref_i) \tag{4.3}$$

Damit im Parameterraum R^{35} der Bereich D^{12} mit dem Bereich D^{35} übereinstimmt, d.h. $(D^{12} \cap R^{35}) = (D^{35} \cap R^{35})$ (Gl. (4.2), darf bzgl. des kritischen Bereichs der Referenzwert eines weniger wichtigen Parameters aus dem Parameterraum R^{23} im Mittel nur ~95 % $(= \exp\left(\frac{\ln(0.323)}{23}\right) \cdot 100\ \%)$ seiner Parameterwerte repräsentieren (siehe Gl. (4.3)), statistische Unabhängigkeit zwischen den 23 Parametern vorausgesetzt. Das bedeutet, dass im Mittel ~5 % der möglichen Werte eines weniger wichtigen Parameters nicht im kritischen Bereich vorkommt.

4.2.2.6 Folgerung aus den Ergebnissen

Eine Folgerung aus den Ergebnissen in den vorhergehenden Abschnitten ist, dass vor der Anwendung des SuSSVR-Verfahrens auf einen komplexen Fall mit einem langlaufenden Simulationscode und vielen unsicheren Parametern zunächst eine Analyse zur Dimensionsreduktion im Parameterraum (z. B. eine Sensitivitätsanalyse) durchgeführt wird, um die Anzahl der Parameter zu reduzieren. Dadurch wird sich bei dem anschließenden SuSSVR-Verfahren mit dem niedriger dimensionierten Parametervektor die Zahl an erforderlichen Läufen mit dem Simulationscode verringern. Die Wahrscheinlichkeit, die dabei für den kritischen Bereich ermittelt wird, sollte anschließend mit einem Korrekturfaktor multipliziert werden, um einen Schätzer für die Eintrittswahrscheinlichkeit des gesamten kritischen Parameterbereichs zu erhalten. Dieser Korrekturfaktor könnte z. B. 0.95^{npar} betragen, wobei $npar$ die Zahl der aussortierten und als unwichtig beurteilten Parameter ist (siehe Gl. (4.2) und Gl. (4.3)). Die vorgeschlagene Korrektur sollte allerdings an weiteren Anwendungsfällen überprüft werden.

4.3 Anwendung des GASA-PRECLAS Algorithmus

Analog zum SuSSVR-Verfahren in Abschnitt 4.2 bestand der anfängliche Trainingsdatensatz aus den ersten 50 zufällig ausgespielten Parameterkombinationen der 35 Parameter und den zugehörigen maximalen Hüllrohrtemperaturen, die sich im zeitlichen Verlauf ergeben haben. Wie bereits in den vorherigen Abschnitten beschrieben, ist der kriti-

sche Bereich durch die Menge von Parameterkombinationen gegeben, deren Funktionswerte PCT > 1200 °C ergeben.

In der Stichprobe der 50 zufälligen Parameterkombinationen haben sich PCT-Werte zwischen 1087 °C und 1203 °C ergeben. 49 PCT-Werte lagen dabei unterhalb und ein PCT-Wert oberhalb des kritischen Wertes von 1200 °C. Damit erfüllte der Trainingsdatensatz noch nicht die Anforderung des PRECLAS-Ansatzes, der mindestens fünf Werte verlangt, die im kritischen Bereich liegen. Aus diesem Grund wurde der genetische Algorithmus des GASA-Ansatzes angewendet, um mit möglichst wenigen ATHLET-Läufen einen Trainingsdatensatz zu erzeugen, der mindesten fünf PCT-Werte im kritischen Bereich enthält.

Die PCT-Werte der in den jeweiligen Genrationen ausgewählten Kandidaten sind in Tab. 4.2 angegeben. In der Generation 0 ist die Situation im anfänglichen Trainingsdatensatz dargestellt. Demnach befand sich eine Beobachtung im kritischen und 49 Beobachtungen im nicht-kritischen Bereich.

In der ersten Generation wurden durch den GASA-Ansatz nach dem in Abschnitt 2.2.2 beschriebenen Selektionsprozess acht Kandidaten (bzw. Parameterkombinationen) ausgewählt, für welche die entsprechenden ATHLET-Rechnungen durchgeführt wurden. Die ermittelten maximalen Hüllrohrtemperaturen von diesen acht Kandidaten lauten [1151.2, 1184.4, 1206.6, 1203.7, 1191.2, 1215.8, 1170.8, 1188.5]. Von den acht ausgewählten Kandidaten der 1. Generation wurden die Ergebnisse zusammen mit ihren Parameterkombinationen dem Trainingsdatensatz hinzugefügt. Damit waren vier kritische und 54 nicht-kritische Werte im Trainingsdatensatz.

Da die Mindestanforderung für den Trainingsdatensatz noch nicht erreicht war, wurde eine 2. Generation von Nachkommen erzeugt. Aus den erzeugten Nachkommen der 2. Generation wurden wiederum acht geeignete Kandidaten ausgewählt und entsprechende ATHLET-Läufe parallel gestartet. Von den acht gestarteten Läufen brach ein ATHLET-Lauf vorzeitig ab und lieferte keine Ergebnisse. Die berechneten maximalen Hüllrohrtemperaturen der restlichen sieben Läufe sind in Tab. 4.2 unter Generation 2 angegeben.

Tab. 4.2 Ergebnisse des GASA-Algorithmus: PCT-Werte ausgewählter Parameterkandidaten und Anzahl der Beobachtungen in den Klassen {PCT > 1200} und {PCT ≤ 1200}

Generation	Anzahl Beobachtungen		PCT-Werte der Parameterkandidaten
	PCT > 1200	PCT ≤ 1200	
0	1	49	Anfänglicher Trainingsdatensatz nach zufälliger Stichprobe
1	4	54	[1151.2, 1184.4, 1206.6, 1203.7, 1191.2, 1215.8, 1170.8, 1188.5]
2	5	60	[1199.6, 1203.2, 1194.9, 1187.8, 1177.7, 1195.0, 1178.3]

Nach Ausführung der 2. Generation enthielt der Trainingsdatensatz 65 Beobachtungen, von denen fünf im kritischen und 60 im nicht-kritischen Bereich lagen. Damit erfüllte der Trainingsdatensatz die Mindestanforderung, die zur Beendigung des GASA-Ansatzes führte.

Nach der Beendigung des GASA-Ansatzes kam im Weiteren der PRECLAS-Algorithmus zur Anwendung, um die Wahrscheinlichkeit zu schätzen, dass in dem gegebenen LOCA-Szenario Hüllrohrtemperaturen > 1200 °C auftreten. Der PRECLAS-Algorithmus wurde in sukzessiven Berechnungsschleifen (Loops) durchgeführt, wobei in jeder Schleife 10000 zufällig erzeugte Testdaten klassifiziert wurden und aus den zugehörigen Prognosen die entsprechende Verteilung der gesuchten Wahrscheinlichkeit geschätzt wurde. Außerdem wurden in jeder Schleife 8 – 10 neue Kandidaten für die ATHLET-Läufe ausgewählt und dem Trainingsdatensatz als neue Beobachtungen hinzugefügt. Die Ergebnisse der Wahrscheinlichkeitsschätzungen der einzelnen Loops sind in Tab. 4.3 angegeben. Die Schätzung, die auf der Basis des durch den GASA-Algorithmus erzeugten Trainingsdatensatzes ermittelt wurde, ist unter Loop 0 der Tab. 4.3 aufgeführt. Die geschätzten Verteilungen der Eintrittswahrscheinlichkeit kritischer Werte aus den einzelnen Berechnungsschleifen werden durch ihre jeweiligen Mittelwerte sowie ihre 5 %-, 50 %- und 95 %-Quantile beschrieben.

Tab. 4.3 Geschätzte Verteilungen der Eintrittswahrscheinlichkeit kritischer PCT-Werte. Die Schätzung erfolgt auf der Basis von 10000 zufällig ausgewählten Testdaten pro Loop

Loop	Beobachtungen mit PCT		Mittelwert	5 %	50 %	95 %	ν	δ
	> 1200	≤ 1200						
0	5	60	4.35E-3	2.5E-4	3.65E-3	9.11E-3		
1	6	69	4.26E-3	4.3E-4	3.51E-3	8.91E-3		
2	6	79	6.2E-3	5.0E-3	6.17E-3	7.53E-3		
3	8	87	5.95E-3	2.38E-3	5.53E-3	1.01E-2		
4	10	95	9.34E-3	3.79E-3	8.89E-3	1.57E-2	0.317	1.19
5	13	101	1.12E-2	4.43E-3	1.07E-2	1.87E-2	0.266	0.88
6	16	108	1.24E-2	9.2E-3	1.23E-2	1.59E-2	0.27	1.08
7	18	116	1.51E-2	1.22E-2	1.5E-2	1.82E-2	0.193	0.62
8	20	124	1.83E-2	1.48E-2	1.82E-2	2.2E-2	0.203	0.63
9	22	131	1.54E-2	1.06E-2	1.54E-2	2.07E-2	0.137	0.48

Der erste Trainingsdatensatz, an dem der PRECLAS-Algorithmus angewendet wurde, ist derjenige, der über den GASA-Algorithmus erzeugt worden ist. In diesem Trainingsdatensatz befanden sich 5 kritische und 60 nicht-kritische Beobachtungen (Loop 0). Die Klassifikation der 10000 zufällig ausgewählten Testdaten pro Loop, lieferten für diesen Trainingsdatensatz eine mittlere Wahrscheinlichkeit von 4.35E-3. 5 %- und 95 %-Quantil der geschätzten Verteilung sind durch 2.5E-4 und 9.11E-3 gegeben. In Loop 0 wurden aus den 10000 Testdaten 10 Kandidaten (Parameterkombinationen) nach dem in Abschnitt 2.2.3 beschriebenem Verfahren ausgewählt: Für die ausgewählten Kandidaten wurden mittels ATHLET-Rechenläufen die jeweiligen maximalen Hüllrohrtemperaturen berechnet. Diese wurden zusammen mit den zugehörigen Parameterkombinationen dem Trainingsdatensatz hinzugefügt.

Auf der Basis des ergänzten Trainingsdatensatzes wurden in Loop 1 für die Klassifikationsverfahren die Metamodelle angepasst. Mit den angepassten Metamodellen wurden in Loop 1 10000 neue zufällig ausgespielte Testdaten klassifiziert und aus den Klassifizierungen eine erneute Wahrscheinlichkeitsschätzung für kritische PCT-Werte ermittelt. Von den 10 Kandidaten, die in Loop 0 ausgewählt wurden, wies ein Kandidat einen kritischen und neun Kandidaten nicht-kritische Funktionswerte auf. Das heißt, im Trainingsdatensatz von Loop 1 befinden sich nun sechs kritische und 69 nicht-kritische Beobachtungen. Die geschätzte Verteilung in Loop 1 hat eine mittlere Wahrscheinlichkeit von 4.26E-3. 5 %- und 95 %-Quantile der Verteilung sind durch 4.3E-4 und 8.91E-3

gegeben. Analog ist das Verfahren zur Erweiterung des Trainingsdatensatzes, der daraus erzeugten Metamodelle sowie die aus den Klassifikationen von Testdaten ermittelten Wahrscheinlichkeitsschätzungen in den nachfolgenden Loops zu interpretieren.

Wie in Abschnitt 2.2.3 unter Punkt (10) beschrieben wurde, dienen die Größen ν und δ als Kriterium, wann die Wahrscheinlichkeitsschätzung als hinreichend konvergent betrachtet werden kann und deshalb der PRECLAS-Algorithmus beendet werden kann. Wie aus Tab. 4.3 abzulesen ist, wird das Abbruchkriterium $\nu < 0.25$ und $\delta < 0.5$ des PRECLAS-Algorithmus in Loop 9 erreicht. D. h., die in Loop 9 geschätzte Verteilung wird als hinreichend konvergent betrachtet und wird als Wahrscheinlichkeitsschätzung für den kritischen Bereich verwendet. Die Schätzung weist dabei eine mittlere Wahrscheinlichkeit von $\widehat{\mathrm{P}}_{\mathrm{PRECLAS}}$ = 1.54E-2 auf, dass maximale Hüllrohrtemperaturen > 1200 °C bzgl. des definierten Szenarios auftreten. 5 %- und 95 %-Quantil der Verteilung liegen bei 1.06E-2 und 2.07E-2. Für diese Schätzung wurden zusätzlich zu den 50 ATHLET-Rechnungen der Zufallsstichprobe, 103 weitere ATHLET-Rechnungen benötigt.

Wie weiter oben bereits ausgeführt, trat in dem anfänglichen Trainingsdatensatz ein Fall auf, bei dem der kritische PCT-Wert von 1200 °C überschritten wurde. Unter Anwendung eines Bayes'schen Schätzverfahrens mit nicht-informativer a-priori Verteilung erhält man aus dieser Zufallsstichprobe eine mittlere Wahrscheinlichkeit von $\widehat{\mathrm{P}}_{\mathrm{MCS}}$ = 2.35E-2 für das Auftreten kritischer Hüllrohrtemperaturen. 5 %- und 95 %-Quantil der Schätzung liegen bei 3.53E-3 und 7.55E-2.

Um die Qualität der Schätzung durch den GASA-PRECLAS Algorithmus beurteilen zu können, ist zunächst festzuhalten, dass nicht nur die mittlere Wahrscheinlichkeit sondern auch das 5 %- und 95 %-Quantil der PRECLAS-Schätzung im 90 % Bereich der Schätzung liegt, die über das Bayes'sche Schätzverfahren auf Basis der Zufallsstichprobe ermittelt wurde. Dies ist ein erster Hinweis auf die Validität der Schätzung durch den GASA-PRECLAS Algorithmus.

Wie die Testbeispiele in Abschnitt 2.2.4 gezeigt haben, erweist sich die entwickelte Methode insbesondere dann als äußerst effizient, wenn die Wahrscheinlichkeit kritischer Bereiche sehr klein ist.

5 MS Windows-basierte SUSA-Version

SUSA setzt sich aus einem Hauptmodul und mehreren Kernmodulen zusammen (Abb. 5.1). Die Kernmodule wurden nativ in Fortan implementiert und beinhalten die Berechnungsroutinen. Sie stehen dem Hauptmodul in Form einer Bibliothek zur Verfügung. Das Hauptmodul mit seiner grafischen Benutzeroberfläche (Graphical User Interface, GUI) übernimmt die Aufgaben eines übergeordneten Anwendungsprogramms. Es gibt Hilfestellungen bei der Dateneingabe, überprüft auf Eingabefehler, übernimmt den Datentransfer und ruft automatisch die Kernmodule für die Berechnungen auf. Darüber hinaus erstellt es Tabellen und Abbildungen für die Dokumentation von Eingabedaten und Analyseergebnissen. Zusätzlich wird durch seine selbsterklärende Benutzeroberfläche mit ihren klaren Vorgaben der Analyseschritte der Know-how-Transfer für Unsicherheits- und Sensitivitätsanalysen erleichtert.

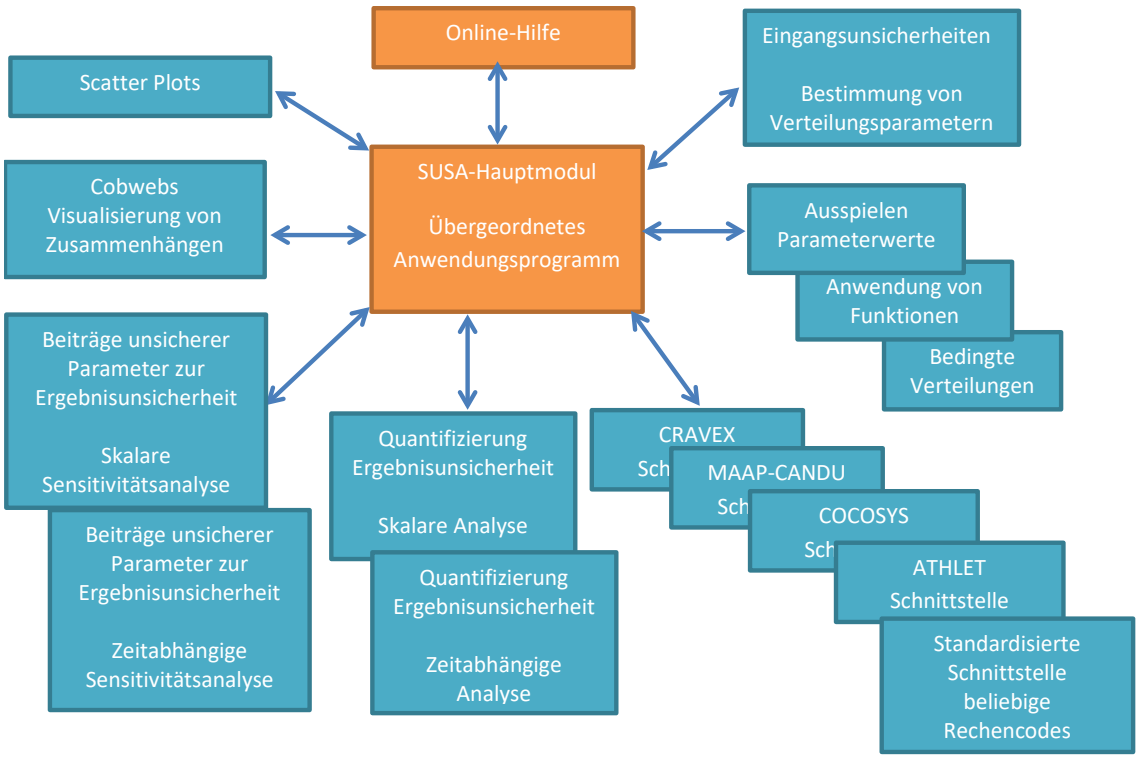

Abb. 5.1 Struktur von SUSA mit Hauptmodul und Kernmodulen

In der Windows-basierten Version von SUSA (SUSA 4.x), die aktuell herausgegeben wird, ist das Anwendungsprogramm in .NET unter der Benutzung der Programmiersprache Visual Basic implementiert. Vor allem die Abhängigkeit von den .NET GUI-Bibliotheken ist Grund dafür, dass die aktuell herausgegebene Version von SUSA ausschließlich auf MS Windows kompatiblen Systemen betrieben werden kann. Die Durchführung

von Rechenläufen auf high-performance Rechenclustern, die überwiegend Unix/Linux-basiert sind, wird durch diese System-Anforderung unnötig erschwert. Darüber hinaus ist die in der Vergangenheit gewählte Programmiersprache Visual Basic für die Kompatibilität als sehr kritisch einzustufen. Da diese auf diversen Zielplattformen, insbesondere in Cluster-Umgebungen, wenn überhaupt nur als frei portierte Variante zur Verfügung steht, ist für die zukünftige Verwendung dieser SUSA-Version mit Inkompatibilitäten und zusätzlichem Portierungsaufwand zu rechnen. Aus diesem Grund wurde im Sinne einer nachhaltigen Entwicklung eine Umstellung auf die plattformübergreifend verfügbare Programmiersprache Python vorgenommen.

Abb. 5.2 Einblendung bei der Aktivierung der aktuellen MS Windows-basierten SUSA-Version

6 Plattformunabhängige SUSA-Version

Die nachhaltige Entwicklung der Softwarearchitektur von SUSA verfolgt das Ziel, das umfassende Analysewerkzeug möglichst unter allen Umgebungen verfügbar und lauffähig zu machen. Außerdem soll die Software noch modularer gestaltet werden, um eine einfachere, kontinuierliche und flexible Erweiterung des Methodenspektrums, z. B. durch die Verwendung wissenschaftlicher Python-Bibliotheken wie scikit-learn oder SciPy, zu erlauben. Durch gezielten API-Aufbau (API: Application Programming Interface, Programmierschnittstelle) und erforderliche Angleichungen soll darüber hinaus eine hinreichende Kompatibilität zu externen Entwicklungen erreicht werden, welche Vergleichsrechnungen, Datenaustausch und Zusammenarbeit von Softwarepaketen oft überhaupt erst möglich macht. Weitere wichtige Ziele sind eine universelle und leichte Handhabbarkeit sowie eine angemessene Unterstützung des Anwenders.

Im Folgenden wird ein Überblick über den aktuellen Entwicklungsstand der plattformunabhängigen SUSA-Version gegeben, dessen wichtigste Entwicklungsergebnisse anhand des Schichtendiagrammes in Abb. 6.1 dargestellt werden können.

6.1 Kernmodule

Die bisher in den Kernmodulen (Fortran) enthaltenen Abhängigkeiten von nicht frei verfügbaren externen Bibliotheken wurden aufgelöst. Darüber hinaus wurden die Schnittstellen der Kernmodule auf eine plattform- und Compiler unabhängige Technik umgestellt. Dazu wurden alle für Einsprung und Datenaustausch zuständigen Funktionen überprüft und deren Signatur korrigiert. Vereinzelt wurden dafür interne Datenkonvertierungen vorgenommen.

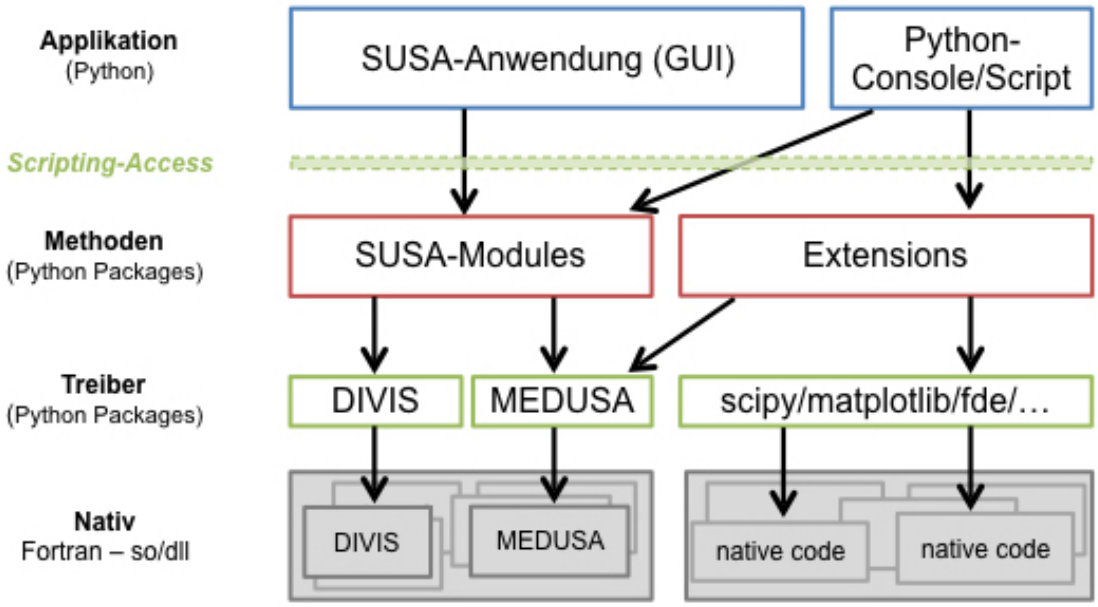

Abb. 6.1 SUSA-Softwarearchitektur mit einer Aufteilung der Software in Schichten zur Steigerung der Wartbarkeit, Kompatibilität und Nutzbarkeit von Programmteilen sowie zur Reimplementierung höherer Schichten in Python (z. B. Treiber, Methoden, GUI)

Für die Bereitstellung der implementierten Routinen als in Python aufrufbare Funktionen wurden die bislang hart codierten Kanalnummern, die für die dateibasierte Ein- und Ausgabe von Daten verwendet wurden, durch dynamisch zugewiesene Kanalnummern ersetzt. Diese Ersetzung ist deshalb so wichtig, weil in den verschiedenen Kernmodulen von SUSA die Kanalnummern oft mehrfach oder für verschiedene Zwecke eingesetzt wurden und deshalb die für die interaktive Verwendung erforderliche ablaufinvariante Arbeitsweise nicht möglich wäre. Darüber hinaus wurden die Kernmodule so geändert, dass bei ihrem Aufruf die Namen der verwendeten Kommunikationsdateien vorgegeben werden können. Dies verhindert, dass bei gleichzeitiger Verwendung zweier Module eine Vermischung der Daten stattfindet.

6.2 Treiber

Auf der Ebene außerhalb des Fortran-Kerns wurde ein Treiber implementiert, der die Schnittstellen für die darüberliegenden Anwendungsschichten bereitstellt. Die hier enthaltenen Funktionen sind dafür zuständig, die in Python formulierten Berechnungsaufrufe in eine Form umzuwandeln, welche von den Kernroutinen verarbeitet werden kann. Vor der Weiterleitung des Aufrufs an den Kern werden die als Argumente übergebenen

Daten und Parameter in eine Kommunikationsdatei geschrieben. Die Fortran-Routinen lesen die so abgelegten Daten aus, verarbeiten diese und legen die berechneten Ergebnisse wiederum in Ausgabedateien ab. Nach erfolgtem Aufruf kann der Treiber die Ergebnisse aus den Dateien lesen und in Python-Datenformate (z. B. numpy-arrays) umwandeln, bevor diese an die höheren Anwendungsschichten zurückgegeben werden.

Durch die Einbettung des Fortran-Kerns in einen Python-Methoden-Layer können die Kernmodule jetzt auch aus Python-Anwendungen und in Kombination mit einschlägigen Python-basierten open-source Programmpaketen verwendet werden. Von der dadurch erreichten Flexibilität wird die Implementierung der darauf aufsetzenden Benutzeroberfläche erheblich profitieren.

6.3 Testframework

Für eine automatische Durchführung von Testdurchläufen wurde ein Python-Testframework entwickelt. Dabei kann für einen Testdurchlauf frei festgelegt werden, welcher SUSA-Kern für die Berechnungen eingeladen wird und welche Auswahl an Tests damit durchgeführt werden sollen. Die durchzuführenden Tests können gezielt vorgegeben werden. Die Tests werden nacheinander vorbereitet, d. h. die notwendigen Eingabe-Dateien werden vorbereitet, danach werden die Routinen des Kerns aufgerufen und nach erfolgter Berechnung werden deren Ergebnisse mit den Referenzdaten des jeweiligen Tests verglichen. Sollten sich bei diesem Vergleich Unterschiede ergeben, werden diese in einer Log-Datei gesammelt, welche dann einen guten Überblick über den Zustand der getesteten SUSA-Version liefert. Dieser Überblick ist vor allem für Codeänderungen und die Weiterentwicklung von Berechnungsroutinen unverzichtbar. Bei entsprechender Testabdeckung können so Fehler schnell erkannt und abweichende Rechenergebnisse auf Plausibilität geprüft werden. Fehlgeschlagene bzw. auffällige Testdurchläufe können unterbrochen und gedebuggt werden. Das automatisierte Testframework in Kombination mit den zugehörigen Referenzdaten ist ein sehr wichtiger Beitrag zur Qualitätssicherung und muss fortlaufend gewartet und um neue Testfälle erweitert werden, um auch die neuen Entwicklungen abdecken zu können.

6.4 Automatisierte Variationsrechnungen

Die auf .NET-basierten Anwendungsteile der Windows-basierten SUSA-Version umfassen nicht nur Code für Aufbau und Ansteuerung der Benutzeroberfläche, sondern enthalten auch Produktivcode, welcher für die plattformunabhängige SUSA-Version neu implementiert werden musste.

Für die neue Python-basierte Anwendung des SUSA-Methodenspektrums wurde eine einheitliche Infrastruktur zur Durchführung von Rechenläufen (Variationsrechnungen) auf der Basis von ausgewählten Kombinationen von Eingabeparametern (Eingabevariationen) benötigt. Für diese Infrastruktur wurde ein Konzept erarbeitet und in Form eines einfach zu verwendenden Python-Pakets implementiert. Dieses zielt darauf ab, Variationsrechnungen sowohl mathematischer Funktionen als auch komplexer Simulationsmodelle möglichst einheitlich durchführen zu können. Hierzu wurde eine Klassenstruktur entwickelt, welche die Schnittstellen festlegt, um die verschiedenen Arten der variationsbezogenen Funktionen aufzurufen. Diese Klassenstruktur erlaubt des Weiteren die Festlegung von Vorgabe- und Variationsargumenten und implementiert die Status- und Fehlerbehandlung sowie den kontrollierten Zugriff auf das Ergebnis einer Variationsrechnung. Durch dieses Design können die übergeordneten methodischen Algorithmen unabhängig von der Art der zu variierenden Funktion implementiert werden und müssen keine spezifischen Behandlungen vorsehen. Abb. 6.2 zeigt den schematischen Aufbau des Packages und das grundsätzliche Zusammenspiel der implementierten Klassen.

Ziel der variationsbezogenen Funktionen ist es, auch Variationsrechnungen (Simulationsläufe auf der Basis von Eingabevariationen) komplexer, durch Eingabedateien gesteuerte Simulationscodes zu unterstützen und diese in gleicher Weise wie mathematische Funktionen handhaben zu können. Hierfür wurde ein weiterer, als Python-Module (FileVarigator) und als Commandline-Interface (varigator) zugänglicher Programmteil entwickelt, welcher die Variation von Eingabedateien und deren Generierung bzw. Anpassung als Stapelverarbeitung erledigt. Der FileVarigator wurde dafür ausgelegt, als Prototyp vorliegende Text-Eingabedateien zu verarbeiten und die hierin als Zuweisung definierten Parameter zu erkennen. Die so erkannten Parameter werden in Form eines Python Dictionarys (bestehend aus Paaren, die sich jeweils aus einem Schlüsselnamen und einem Wert zusammensetzen) als benannte Vorgabewerte bereitgestellt und, falls nicht abweichend definiert, für Variationsläufe herangezogen. Alle weiteren Definitionen der Prototyp-Eingabedatei werden als Template für andere Verarbeitungsschritte bereitgestellt. Zur Erzeugung einer variierten Eingabedatei wird dieses Template mit den

gegebenen Variationswerten bzw. Vorgabewerten belegt. Die so erstellten Eingabedateien können flexibel, z. B. indiziert benannt und/oder in eigenen Verzeichnissen erzeugt werden, welche dann direkt als Arbeitsverzeichnisse für die durchzuführenden Variationsläufe verwendet werden können.

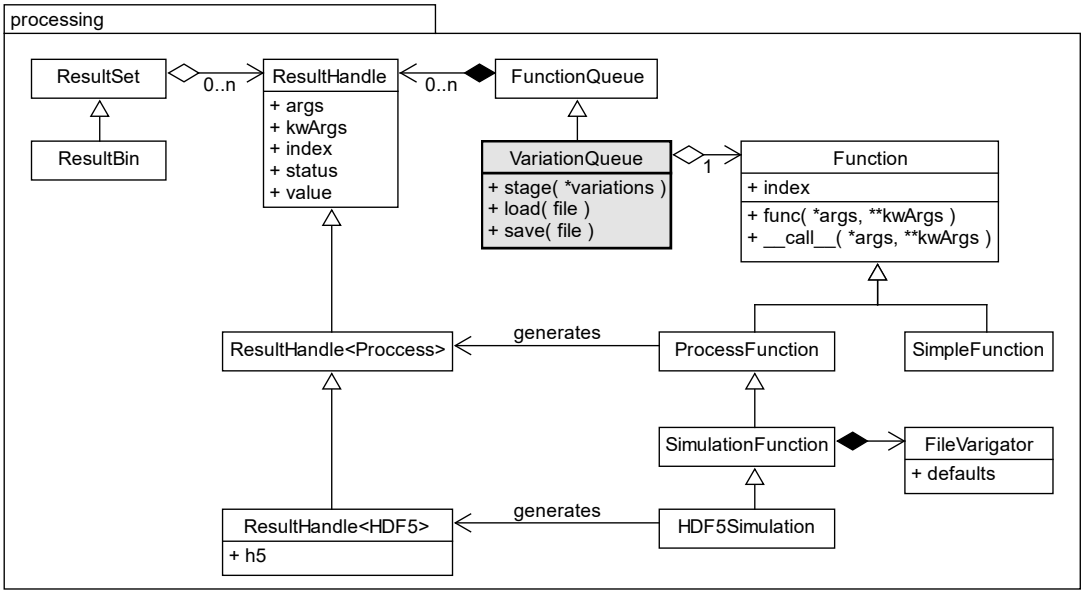

Abb. 6.2 Schematischer Aufbau des entwickelten Python-Pakets zur automatisierten Durchführung von Variationsrechnungen. Die Hauptschnittstelle zur Verwendung in übergeordneten methodischen Algorithmen wird durch die sog. VariationQueue bereitgestellt

Zur Durchführung und Überwachung von Variationsrechnungen wurde ein weiteres Python-Module (FunctionQueue/VariationQueue) implementiert, welches für vorgegebene Eingabevariationen die zu verwendende Funktion aufruft. Die Aufrufe finden nebenläufig statt, sodass der Ablauf von übergeordneten methodischen Algorithmen nicht blockiert wird. Ein gegebenes Set an Eingabevariationen wird dabei in Form eines Bins (ResultBin) zusammengefasst, welcher dem übergeordneten Algorithmus wie z. B. einem zentralen Organisator (Broker) eines Workflows (siehe Abschnitt 6.5) erlaubt, dessen Abarbeitung zu überwachen. Um für komplexe Simulationsläufe verfügbare Rechenkapazitäten auch parallel nutzen zu können, wurde ein Funktionstyp (ProcessFunction) implementiert, welcher die Ausführung in einem eigenen Rechenprozess auslagert. Besonders Simulationscodes wie ATHLET/-CD können dadurch sehr einfach im Hintergrund kontrolliert ablaufen.

Durch die Verschiedenartigkeit der unterstützen Typen von variationsbezogenen Funktionen musste auch der Zugriff auf die Rechenergebnisse von Eingabevariationen abstrahiert werden. Dieser kann für verschiedene Funktionen sehr unterschiedlich sein und durch die große Bandbreite an Ergebnisdaten viele Unterscheidungen erfordern. Zum Beispiel können die Ergebnisse mathematischer Funktionen mit skalaren Rückgabewerten direkt weiterverarbeitet werden, wohingegen die Ergebnisse aus Simulationsläufen, die als Zeitreihe in HDF5-Dateien abgelegt werden, eine ganz andere Handhabung erfordern. Diese Spezialbehandlungen sollten in übergeordneten Methoden-Algorithmen soweit wie möglich minimiert werden. Hierfür wurden die generische Python-Klasse ResultHandle und deren Spezialisierungen entwickelt, welche sowohl die Statusabfrage und den Ergebniszugriff als auch eine kontrollierte Fehlerbehandlung (deferred exceptions) ermöglichen.

6.5 Neukonzeption des methodischen Workflows

Da in der Windows-Version von SUSA auch große Teile der Methodik-Steuerung und des Anwendungsablaufs sehr nahe an der Benutzeroberfläche und auf .NET-basiert implementiert wurden, mussten auch diese Teile neu entworfen und als Python-Packages aufgebaut werden. Um sowohl bestehende als auch neuentwickelte Verfahren von Variationsrechnungen mittels eines möglichst einheitlichen Workflows umzusetzen wurde ein generisches Konzept erarbeitet, dessen Basismodule in Abb. 6.3 skizziert sind.

Zur Implementierung von Verfahren für eine klassische oder adaptive MCS wurde eine einfache Broker-Klasse auf die oben beschriebene Infrastruktur von Eingabevariationen und zugehörigen Variationsrechnungen und -ergebnissen aufgebaut. Die Generierung von Eingabevariationen je nach Bedarf und Methodik wird durch eine Pool-Klasse (GenericPool) bereitgestellt. Je nach Spezifikation des Pools (z. B. SubsetSamplingPool) selbst oder eines übergebenen Pool-Produzenten (z. B. SimpleRandomSamplingProducer) können geeignete Eingabevariationen sukzessive oder blockweise erzeugt werden.

Die optionale Bewertung der Eingabe- und/oder Ausgabevariationen und/oder deren geeignete Transformation (Standardisierung, Normalisierung, etc.) wurde in voneinander unabhängigen Klassen implementiert (Grader und Transformer Klasse), die modular vom Broker als zentralem Organisator in dem jeweiligen Workflow einer Methode kombiniert und angewendet werden können. Im Rahmen einer klassischen oder adaptiven

MCS kann nun der Broker eine bestimmte Anzahl von Eingabevariationen bewerten und auswählen sowie deren Simulationsergebnisse berechnen lassen. Er kann außerdem die Schritte einer MCS durch eine verschachtelte Liste an Referenzen zu den entsprechenden ResultHandles (JobBins) eineindeutig protokollieren. Der Broker dient also als funktionale Zwischenschicht zwischen der Datenbasis (Eingabevariationen), Datengenerierung (Simulationsrechnungen) und Datenverarbeitung (Bewertung bestehender und/oder Generierung neuer Eingabevariationen). Das Verfahren zur Durchführung einer klassischen oder adaptiven MCS wird in einer eigenständigen Klasse implementiert, so dass dessen Komplexität, die Spezifikation des Verfahrens und die hierfür notwendigen Dateninfrastrukturen als separate Einheit gehalten werden können. Dadurch können verschiedene Samplingverfahren implementiert und je nach Anwendungsfall im Broker eingesetzt werden.

Die entwickelten Python-Pakete bieten die Grundlage, um Anwendungen klassischer als auch adaptiver MCS (Variationsrechnungen) durch möglichst einfache, einheitliche und durch Vererbung aufeinander aufbauende Workflows umzusetzen.

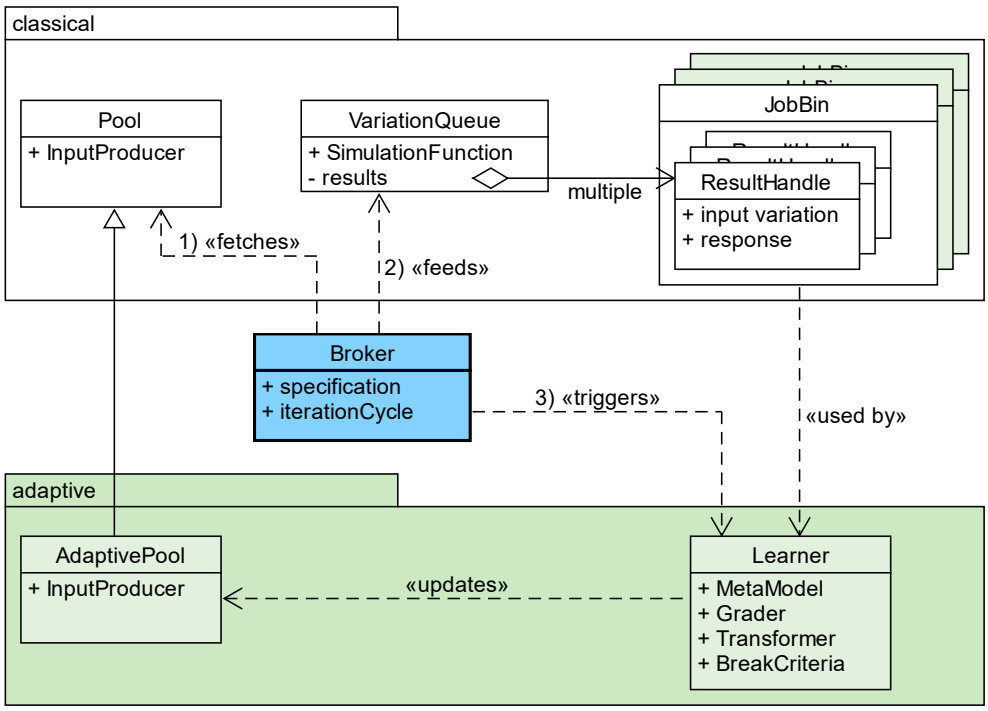

Abb. 6.3 Schematischer Aufbau des generischen Workflows zur Durchführung klassischer (classical) und adaptiver (adaptive in grün) MCS. Der Broker (blau) ist dabei der zentrale Organisator, der das jeweilige MCS-Verfahren durch passende Kombination der Komponenten umsetzt

6.6 Benutzeroberfläche

Es wurde ein auf Jupyter Notebooks basierender Prototyp für eine plattformunabhängige Anwendung von SUSA entwickelt. Dieser ist optimal für Tests und erste Anwendungen einsetzbar, weist allerdings noch nicht die vom Endanwender erwarteten Bedienstandards auf. Die eigentlichen Vorteile eines Jupyter Notebooks sollen im Folgenden anhand des exemplarischen Notebooks in Abb. 6.4 erläutert werden.

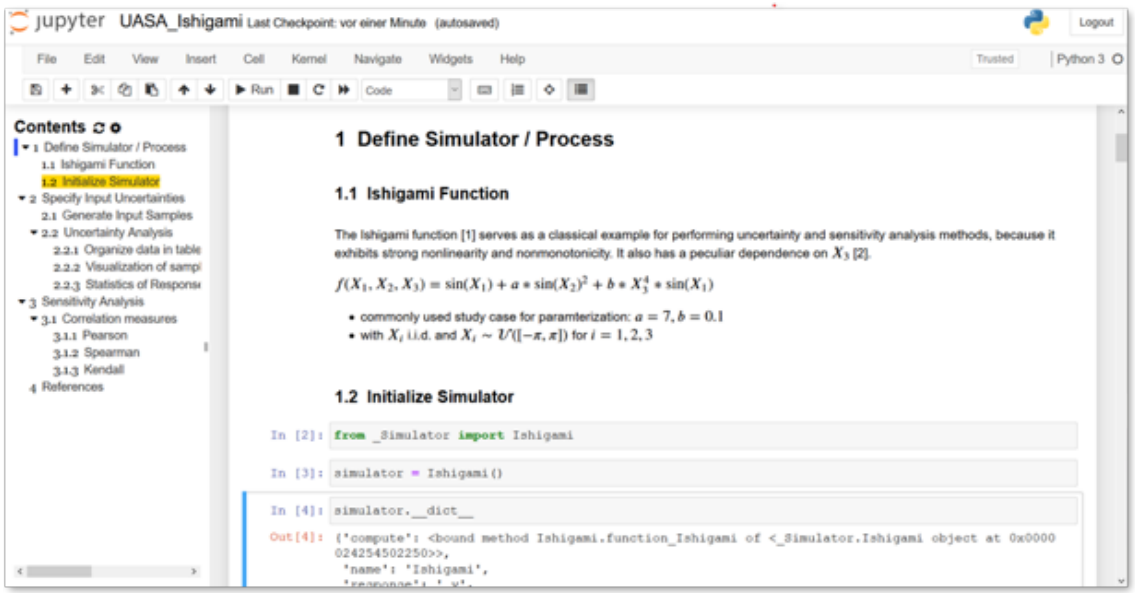

Abb. 6.4 Exemplarisches Jupyter Notebook geöffnet in einem Internetbrowser (hier Firefox) zur statistischen Analyse der Ishigami Funktion

Das Jupyter Notebook ist eine Open-Source-Webanwendung, mit der Dokumente erstellt und geteilt werden können, die sowohl in Zellen unterteilten Code, mathematische Gleichungen, graphische und tabellarische Darstellung (auch Animationen) und dokumentierten Text enthalten können. Als Webanwendung sind Jupyter Notebook per se plattformunabhängig und können prinzipiell in jedem Standardbrowser verwendet werden (offiziell wird der Firefox-Browser als bestmögliche Option empfohlen). Der enthaltene Code kann komplett oder zellenweise ausgeführt werden und zur Laufzeit des Notebooks modifiziert und erweitert werden. Die Dokumentation insbesondere methodischer Entwicklungen kann durch Zellen im Markdown-Format integriert werden (in Abb. 6.4 z. B. Abschnitt 1.1) und erlaubt so durch einfaches Importieren eine interaktive, selbstständige Einarbeitung in die bereitgestellten Module (z. B. in Abb. 6.4 das neu implementierte SUSA-Modul _Simulator). Durch sinnvolle Erweiterung der Jupyter Notebook Umgebung kann sowohl das Arbeiten als Entwickler (u. a. durch einfache Umstellung

auf verschiedene Python Kernel) als auch als Anwender (u. a. durch die Navigation via Inhaltsverzeichnis (englisch: Contents)) effizient und übersichtlich gestaltet werden.

Ein weiterer wichtiger Vorteil von Jupyter Notebooks besteht in dem einfachen Austausch von Analyseergebnissen. Zum Beispiel kann ein ausgeführtes Notebook (d. h. mit allen Rechenergebnissen, tabellarischen und graphischen Darstellungen) abgespeichert und versandt werden, so dass die Ergebnisse ohne erneute Ausführung eingesehen werden können. Durch das zugrunde liegende JSON Format eines Jupyter Notebooks ist auch die Versionierung (z. B. unter git) einfach möglich und erlaubt eine nachhaltige Entwicklung der Notebooks.

Es war ursprünglich geplant, die .NET-basierte Benutzeroberfläche der MS Windows-Version von SUSA (Abschnitt 5) durch eine plattformunabhängige Reimplementierung in Qt zu ersetzen. Allerdings wurden bei den Entwicklungen zu den Methoden in den Abschnitten 2 – 3 und der neuen Software-Architektur viele Neuerungen erarbeitet, welche über das Bedienkonzept der bisherigen Benutzeroberfläche hinausgehen (z. B. bzgl. der Auswahl von Eingabevariationen und der Durchführung von Variationsrechnungen). Hierdurch zeigte sich schnell die Notwendigkeit, die Benutzeroberfläche und deren Dialoge erstmal auf die neuen Anwendungsmöglichkeiten anzupassen und an vielen Stellen komplett neu zu entwickeln. Eine direkte Umsetzung der .NET-Elemente nach Qt war dadurch wenig sinnvoll und wurde durch den Ansatz einer Zwischenlösung mit den oben beschriebenen Jupyter-Notebooks ersetzt. Die Notebooks können zwar keine komfortable Benutzeroberfläche ersetzen, durch ihre Flexibilität erlauben sie aber dennoch, die neuen Methoden zu testen und bereits interaktiv geführt anzuwenden.

7 Methoden- und Benutzerdokumentation

Gemäß den GRS-Vorgaben für Softwareentwicklung /GRS 20/ werden jedem SUSA-Anwender eine Methoden (Programm)- und eine Benutzerdokumentation sowie eine Installationsdokumentation zur Verfügung gestellt. Die Methodendokumentation ist in zwei Handbüchern zusammengefasst. Im ersten Handbuch sind die Methoden für eine klassische Unsicherheits- und Sensitivitätsanalyse beschrieben /KLO 21a/. Das zweite Handbuch beinhaltet eine Beschreibung von Methoden zum adaptiven Sampling und maschinellen Lernalgorithmen /KLO 21b/, die für SUSA entwickelt wurden. Die Methodendokumentation enthält zusätzlich interaktive Jupyter Notebooks, die eine leichte Einarbeitung in die Methoden von SUSA erlauben.

Zur Benutzerdokumentation (Bedienungsanleitung) gehören ein elektronisch verfügbares Benutzerhandbuch sowie eine kontext-sensitive Online-Hilfe. Außerdem wird eine Installationsdokumentation zur Verfügung gestellt. Sie besteht aus einer einfachen Textdatei mit den Installationsanweisungen und einer pdf-Datei, die zusätzlich zu den Anweisungen Abbildungen mit beispielhaften Dialogfenstern enthält. Alle Dokumentationsarten sind in englischer Sprache verfasst.

8 Zusammenfassung und Ausblick

Es wurden drei unterschiedliche Ansätze für eine adaptive Monte Carlo Simulation (MCS) entwickelt. Alle Ansätze beruhen auf Algorithmen des maschinellen Lernens (ML). Zwei der entwickelten Ansätze zielen darauf ab, mit praktikablem Rechenaufwand die niedrigen Wahrscheinlichkeiten für seltene kritische Bereiche im Eingabeparameterraum eines Simulationscodes (z. B. Parameterbereich, der zu einem Systemausfall führt) zu ermitteln und den kritischen Parameterbereich durch Wahrscheinlichkeitsverteilungen sowie funktionelle Abhängigkeiten zwischen den Parametern zu charakterisieren. Ziel des dritten Ansatzes ist die Lokalisierung des Parameterbereichs, in dem Cliff-Edge-Effekte auftreten.

Das iterative SuSSVR-Verfahren, das auf den kritischen Parameterbereich und seine Wahrscheinlichkeit abzielt, verwendet ein auf Stützvektorregression (SVR) basierendes Metamodell im Rahmen einer Subset-Simulation (SuS), um schnell potentielle Kandidaten aus dem kritischen Parameterbereich zu erhalten. Mit einer kleinen Auswahl dieser Parameterkandidaten werden dann die Läufe des eigentlichen Simulationscodes durchgeführt. Die Parameter und Ergebniswerte dieser Läufe werden dem Trainingsdatenpool hinzugefügt, der zur weiteren Anpassung des SVR-Metamodells verwendet wird. Wenn die Vorhersagen des Metamodells und der mit dem SuS-Verfahren ermittelte Wahrscheinlichkeitsschätzer für den kritischen Parameterbereich robust genug sind, wird das iterative Verfahren beendet. Abschließend wird ein umfangreiches SuS-Verfahren mit dem robusten SVR-Modell angewendet, um eine große Stichprobe von Parametern aus dem kritischen Bereich und einen konsistenten Schätzer für seine Wahrscheinlichkeit zu erhalten. Die große Stichprobe wird dazu verwendet, den kritischen Parameterbereich genauer zu charakterisieren.

Die Kombination aus GASA- und PRECLAS-Ansatz, die ebenfalls auf den kritischen Parameterbereich und seine Wahrscheinlichkeit abzielt, verwendet ein Ensemble von Metamodellen, die auf verschiedenen ML-Algorithmen (hauptsächlich Klassifikationsverfahren) beruhen. Die Metamodelle werden angewendet, um eine große Stichprobe zufällig ausgewählter Parametervektoren entsprechend der Zugehörigkeit zum kritischen bzw. nicht-kritischen Bereich zu klassifizieren. Aus der Gesamtheit der Parametervektoren, deren Klassifikation durch die Modelle nicht eindeutig ist, werden dann die Parameterkandidaten für die Läufe des eigentlichen Simulationscodes ausgewählt. Die Parameter und Ergebniswerte dieser Läufe werden dem Trainingsdatenpool hinzugefügt, der zur weiteren Anpassung der Metamodelle verwendet wird. Das iterative Verfahren des

Ansatzes wird beendet, wenn sich die Verteilungen für die Wahrscheinlichkeit des kritischen Bereichs, die bei jedem Iterationsschritt aus den klassifizierten Daten der großen Stichprobe mittels eines Bayes‘schen Ansatzes geschätzt werden, nicht mehr wesentlich voneinander unterscheiden. Der GASA-Ansatz wird am Anfang des Verfahrens eingesetzt, um sicherzustellen, dass ausreichend Parametervektoren aus dem kritischen Bereich im Trainingsdatensatz sind, damit die Klassifikationsmodelle auch entsprechend trainiert werden können.

Der dritte Ansatz – der adaptive Gauß-Prozess-Sampling-Ansatz – ist in der Lage, den kritischen Bereich der Eingangsparametervektoren zu lokalisieren, der zu Cliff-Edge-Effekten (CEE) führt. Dabei wird ein Gauß-Prozess (GP) iterativ als Metamodell auf einen wachsenden Datensatz von Parametervektoren und den zugehörigen Ergebnissen (Systemantworten) eines (komplexen) Simulationscodes trainiert. Durch die Anwendung des aktualisierten GPs auf eine große Stichprobe zufällig ausgewählter Parametervektoren und die Bewertung der Vektoren anhand eines Lernkriteriums, das darauf abzielt, die Parametervektoren in der anvisierten CEE-Parameterregion zu finden, werden die GP-Trainingsdaten effizient um geeignete Parametervektoren erweitert. Die hinzugefügten Parametervektoren werden verwendet, um das jeweilige Ergebnis durch Ausführen des Simulationscodes zu erhalten und die neuen Einträge in den GP-Trainingsdaten zu vervollständigen.

Alle drei entwickelten Ansätze für eine MCS wurden anhand von Anwendungsbeispielen untersucht und erwiesen sich als wesentlich effizienter im Vergleich zur klassischen MCS oder zum reinen SuS-Verfahren. Der SuSSVR- und GASA-PRECLAS Ansatz wurden außerdem erfolgreich auf ein umfangreiches Anwendungsbeispiel mit dem Simulationscode ATHLET angewendet.

Bei der Anwendung der Methoden wurde auch Verbesserungs- und Weiterentwicklungsbedarf deutlich. Dieser bezieht sich auf das Auswahlverfahren von Parameterkandidaten für die Berechnung mit einem (komplexen) Rechencode. Hierzu sollten weitere fortschrittliche Samplingverfahren und verbesserte Scoring-Funktionen zur Bewertung von Parameterkombinationen entsprechend ihrer Zugehörigkeit zum interessierenden Parameterbereich entwickelt und umgesetzt werden. Außerdem sollten weitere Konvergenzkriterien für den iterativen Prozess der adaptiven MCS-Methoden anhand von Anwendungsbeispielen untersucht werden. Durch die Bereitstellung vielfältiger Konvergenzkriterien sowie Scoring-Funktionen kann je nach Anwendungsfall die Performanz der Verfahren weiter optimiert werden. Da auch die Wahl der ML-Methode entscheidend für

eine gute Performanz von adaptiven MCS-Verfahren ist, sollen weitere ML-Methoden wie z. B die Stützvektor-Methode oder neuronale Netze berücksichtigt werden.

Eine wichtige Zielsetzung ist die Plattformunabhängigkeit von SUSA. Dadurch soll das Analysewerkzeug unter möglichst vielen Systemumgebungen verfügbar gemacht werden und so nicht nur seine Anwendung ermöglichen, sondern auch eine kontinuierliche und flexible Erweiterung des Methodenspektrums erlauben. Hierzu konnten entscheidende Fortschritte gemacht werden. Durch die Aufteilung der SUSA-Software in Schichten konnten die Schnittstellen vereinheitlicht und somit die Kompatibilität zu extern entwickelten Bibliotheken hergestellt werden, welche dann für die SUSA-Methoden genutzt werden können. Um die im Fortran-Kern von SUSA implementierten Berechnungsroutinen und Methoden auch unter den Bedingungen der neuen Softwarearchitektur verwenden zu können, waren sowohl Anpassungen im Fortran-Code als auch die Erstellung eines Python-Treibers notwendig. Der Treiber ermöglicht die Ansteuerung der Fortran-Routinen aus Python und konnte somit auch für die Realisierung des Testframeworks genutzt werden, welches in automatisierten Testläufen die Lauffähigkeit und Funktionalität der Kernroutinen überprüft. Durch den Ausbau der Tests wird hiermit ein wichtiger Beitrag zur Qualitätssicherung von SUSA geleistet.

Für den Aufbau der methodischen Schicht der plattformunabhängigen SUSA-Architektur wurde die automatisierte Berechnung von Eingabevariationen sowohl für mathematische Funktionen als auch für komplexe Simulationscodes neu implementiert. Die erarbeitete Infrastruktur wurde in Form von Python-Packages realisiert, welche die einfache und möglichst einheitliche Verarbeitung der Eingabevariationen und zugehörigen Rechenergebnisse für einfache Funktionen und komplexe Simulationscodes erlauben. Auch die Methodik-Steuerung wurde überarbeitet und in einem neu entworfenen Workflow implementiert. Sein generisches Konzept ermöglicht die Durchführung klassischer und adaptiver Variationsrechnungen und schafft die Grundlage für die Entwicklung neuer Methoden bzgl. fortschrittlicher Samplingverfahren, die unter anderem auch maschinelle Lernalgorithmen einsetzen. Für den plattformunabhängigen Einsatz wurde die .NET-basierte Benutzeroberfläche der MS Windows-basierten SUSA-Version durch Jupyter-Notebooks als Zwischenlösung ersetzt. Diese bieten zwar bei weitem nicht den Komfort einer integrierten Benutzeroberfläche, erlauben aber dennoch bereits den interaktiven Einsatz und eine geführte Anwendung der neuen Methoden. Es ist geplant, auf der Basis dieser Notebooks eine Benutzeroberfläche unter Qt zu entwickeln.

Literaturverzeichnis

/AUS 01/ Au, S. K., J. L. Beck: Estimation of small failure probabilities in high dimensions by subset simulation, Probab. Eng. Mech. 16 (4), 263-277, 2001.

/AUS 10/ Au, S.-K., Z.J. Cao, and Y. Wang: Implementing advanced Monte Carlo simulation under spreadsheet environment. *Structural Safety 32*, pp. 281–292, 2010.

/AUS 13/ Austregesilo, H., H. Glaeser, P. Schöffel, T. Skorek: Teilnahme am Internationalen Standardproblem ISP-50 mit ATHLET, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, GRS-A-3685, 2013.

/BER 20/ Berner, N., M. Kloos, J. Peschke: Localizing cliff-edge effects in accident analyses via an adaptive Gauss process sampling approach. in: Baraldi, P., F. Di Maio, and E. Zio (Eds.) Proceedings of 30th European Safety and Reliability Conference (ESREL 2020) and the 15th Probabilistic Safety Assessment and Management Conference (PSAM 15), Research Publishing, Singapore, ISBN: 981-973-4949-00-0, in publication 2020, https://doi.org/10.3850/981-973-4949-00-0 esrel2020psam15-paper.

/BOU 16/ Bourinet, J.-M.: Rare-event probability estimation with adaptive support vector regression surrogates, Reliability Engineering and System Safety 150, 210-221, 2016.

/BOX 73/ Box G. E. P., G. C. Tiao: Bayesian Inference in Statistical Analysis, Addison-Wesley Publishing Company, 1973.

/CLO 34/ Clopper, C., E. S. Pearson: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika. 26 (4): 404–413, 1934.

/ECH 11/ Echard, B., Gayton N., M. Lemaire: AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation, Structural Safety 33, pp. 145–154, 2011.

/EPA 77/ U.S. Environmental Protection Agency (EPA), Office of Radiation Programs: 40 CFR 190, Environmental radiation protection requirements for normal operations of activities in the uranium fuel cycle, 1977.

/GOL 89/ Goldberg, D.: Genetic algorithms in search, optimization, and machine learning; Addison-Wesley-Verlag, 1989.

/GRS 20/ GRS: Managementhandbuch Kap. 2.2.3.5 Softwareentwicklung (TKP 03-05), Stand 26.05.2020.

/HSI 09/ Hsieh, W. W.: Machine learning methods in the environmental sciences: Neural networks and kernels, Cambridge University Press, 2009.

/HUA 16/ Huang, X., J. Chen, H. Zhu: Assessing small failure probabilities by AK–SS: An active learning method combining Kriging and Subset Simulation. *Structural Safety 59,* pp. 86–95, 2016.

/IAE 16/ International Atomic Energy Agency (IAEA): Safety of Nuclear Power Plants: Design, Specific Safety Requirements, IAEA Safety Standards Series No. SSR-2/1, Rev. 1 p. 15, footnote 9, 2016.

/KLO 20/ Kloos M., N. Berner, J. Peschke: Adaptive Monte Carlo simulation for detecting critical regions in accident analyses, in: Baraldi, P., F. Di Maio, and E. Zio (Eds.) Proceedings of 30th European Safety and Reliability Conference (ESREL 2020) and the 15th Probabilistic Safety Assessment and Management Conference (PSAM 15), Research Publishing, Singapore, ISBN: 981-973-4949-00-0, 2020.

/KLO 21a/ Kloos M., N. Berner: SUSA, Software for Uncertainty and Sensitivity Analyses, Classical Methods, GRS-631, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, ISBN: 978-3-949088-20-9, 2021.

/KLO 21b/ Kloos M., J. Peschke: SUSA, Software for Uncertainty and Sensitivity Analyses, Machine Learning Methods for Adaptive Monte Carlo Simulation, GRS-P-5, Vol. 2, Part 2, Rev. 1, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, 2021.

/LEB 09/ Lebrun, R., A. Dutfoy: Do Rosenblatt and Nataf isoprobabilistic transformations really differ? Prob. Eng. Mech. 24, pp. 577-584, 2009.

/LEL 18/ Lelièvre, N., P. Beaurepaire, C. Mattrand, N. Gayton: AK-MCSi: A Kriging-based method to deal with small failure probabilities and time-consuming models. Structural Safety, 73, pp. 1-11, 2018.

/NAT 62/ Nataf, A. Détermination des distributions dont les marges sont données. *C. R. Acad. Sci. Paris 225,* pp. 42–43, 1962.

/PED 17/ Pedroni N., E. Zio: An Adaptive Metamodel-Based Subset Importance Sampling approach for the assessment of the functional failure probability of a thermal-hydraulic passive system, Applied Mathematical Modelling 48 269-288, 2017.

/PIC 84/ Picard, R., D. Cook: Cross-Validation of Regression Models. *Journal of the American Statistical Association. 79 (387*), pp. 575–583, 1984.

/PAL 14/ Palazzo S., G. Pallàs Moner, A. Kerner: Wissensbasis zur Durchführung von „Best Estimate“ Störfallanalysen, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, GRS - A – 3753, 2014.

/PAP 15/ Papaioannou, I., W. Betz, K. Zwirglmaier, D. Straub: MCMC algorithms for Subset Simulation. *Prob. Eng. Mech. 41,* pp. 89–103, 2015.

/POI 18/ Pointer, W., N. Berner, M. Kloos, S. Wenzel: Statistische LOCA-Analysen, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, GRS-519, ISBN 978-3-947685-04-2, 2018.

/SMO 04/ Smola, A.J., B. Schölkopf: A tutorial on support vector regression. Statistics and Computing 14, pp. 199-222. Kluwer Academic Publishers, 2004.

/STO 74/ Stone, M. Cross-validatory choice and assessment of statistical predictions. J. Royal Stat. Soc. 36(2), pp.111–147, 1974.

/VAP 95/ Vapnik, V.: The Nature of Statistical Learning Theory, Springer Verlag, New York, NY, USA, 1995.

/WIL 06/ Williams, C. K., C. E. Rasmussen: Gaussian processes for machine learning. Vol. 2, No. 3. Cambridge, MA: MIT Press, 2006.

/WIL 41/ Wilks, S. S.: Determination of sample sizes for setting tolerance limits, Annals of Mathematical Statistics 1 (1), 91-96, 1941.

/WIL 42/ Wilks, S. S.: Statistical prediction with special reference to the problem of tolerance limits, Annals of Mathematical Statistics 13 (4), 400-409, 1942.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface, Programmierschnittstelle
BEPU	Best Estimate Plus Uncertainty
CEE	Cliff-Edge-Effekte
DWR	Druckwasserreaktor
GASA	Genetischer Adaptiver Stichproben Algorithmus (Genetic Adaptive Sampling Algorithm)
GP	Gauß-Prozess
GUI	Graphical User Interface
KNN	K-Nearest Neighbor
LSF	Limit state function
MCMC	Markov-Chain-Monte-Carlo
MCS	Monte-Carlo-Simulation
MH	Metropolis-Hastings
ML	Maschinelles Lernen
MSE	Mean squared error (mittlerer quadratischer Fehler)
PCT	Peak Cladding Temperature
PRECLAS	Probability Estimation with an Ensemble of Classification Algorithms
RBF	Radialbasisfunktion
SHB	Sicherheitsbehälter
SuS	Subset-Simulation
SuSSVR	Subset-Simulation mit einem SVR-Metamodel
SUSA	Software for Uncertainty and Sensitivity Analyses
SVR	Stützvektorregression
VB	Visual Basic
wNN	weighted Nearest Neighbor

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Forschungszentrum
Boltzmannstraße 14
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

ISBN 978-3-949088-23-0