Fakultät Maschinenbau

RUHR
UNIVERSITÄT
BOCHUM

RUB

Multifidelity-Optimierungsverfahren für Turbomaschinen

Dissertation
zur
Erlangung des Grades
Doktor-Ingenieur

an der
Fakultät für Maschinenbau
der Ruhr-Universität Bochum

von

Andreas Schmitz
geb. in Waldbröl

Bochum 2020

Dissertation eingereicht am: 25.09.2020

Tag der mündlichen Prüfung: 18.11.2020

Erstgutachter: Prof. Dr.-Ing. R. Mönig (Ruhr-Universität Bochum)

Zweitgutachter: Prof. Dr. rer. nat. H. Gottschalk (Bergische Universität Wuppertal)

Kurzfassung

Das Ziel dieser Arbeit ist die Entwicklung eines industriell einsetzbaren Multifidelity-Optimierungsverfahrens. Dieses ist speziell auf die Problemstellungen in der Turbomaschinenentwicklung angepasst. Die Herausforderungen bei der Optimierung von Turbomaschinen sind die Existenz mehrerer Zielfunktionen und zahlreicher Nebenbedingungen, sowie hochdimensionale Suchräume und sehr laufzeitintensive Prozessketten. Das hier vorgestellte Multifidelity-Optimierungsverfahren sorgt dafür, dass hochdimensionale Suchräume effizienter abgetastet werden und damit die Laufzeit der Optimierungen deutlich reduziert werden. Hierfür werden schnellere Prozessketten niedrigerer Güte herangezogen und mit Hilfe der dort enthaltenen Informationen die Anzahl an teuren Prozesskettenauswertungen reduziert.

Um dies zu erreichen wird zum einen die Entwicklung einer hocheffizienten, flexiblen und modularisierten Ersatzmodellsoftware vorgestellt und zum anderen die Entwicklung einer vollständig automatisierten Multifidelity-Optimierungsstrategie. Die entwickelte Software unterstützt zahlreiche moderne Hardwarearchitekturen und ist für den Betrieb innerhalb einer Großrechnerumgebung ausgelegt.

Das entwickelte Verfahren wurde zahlreichen Tests in Form von Optimierungen analytischer Testfunktionen unterzogen, welche die Vorteile dieses Verfahrens belegen. Neben diesen Tests wurde das hier entwickelte Verfahren innerhalb von unterschiedlichen Turbomaschinen-Anwendungen aus Industrie und Forschung angewendet. Die dort gewonnenen Ergebnisse und Erkenntnisse werden in dieser Arbeit vorgestellt.

Abstract

The aim of this thesis is the development of an industrially applicable multifidelity optimization process. This process is specially adapted to the problems of turbomachinery development. The challenges in the optimization of turbomachinery are the existence of several objective functions and numerous constraints, as well as high-dimensional search spaces and very runtime-intensive process chains. The multifidelity optimization method presented here ensures that high-dimensional search spaces are sampled more efficiently and thus the runtime of the optimizations is significantly reduced. For this purpose, faster process chains of lower quality are used and the number of expensive process chain evaluations is reduced with the help of the information contained therein.

In order to achieve this, the development of a highly efficient, flexible and modularized surrogate model software is presented on the one hand, and the development of a fully automated multifidelity optimization strategy on the other hand. The developed software supports numerous modern hardware architectures and is designed for operation within a supercomputer environment.

The developed procedure is subjected to numerous tests in the form of optimizations of analytical test functions, which prove the advantages of this procedure. In addition to these tests, the procedure developed here has been applied in several turbomachinery applications in industry and research. The results and findings obtained there are presented in this paper.

Vorwort

Ich möchte das Vorwort nutzen, um ein paar persönliche Gedanken auszudrücken und mich bei den Menschen zu bedanken, die mich die letzten Jahre in besonderem Maße unterstützt und begleitet haben. An erster Stelle möchte ich Christian Voß und Marcel Aulich danken. Wir haben immer sehr eng und auch erfolgreich zusammengearbeitet und ich kann für meinen Teil nur sagen, dass ich unglaublich viel von euch gelernt habe und sehr von eurem Wissen und eurer Geduld profitiert habe. Dafür ein wirklich großes Dankeschön und ich hoffe, dass ich euch auch etwas zurückgeben konnte.

Weiterhin möchte ich auch Eberhard Nicke für seine Unterstützung und für seinen Einsatz bedanken. Meine persönliche Meinung ist, dass er ein außergewöhnlicher Abteilungsleiter ist, welcher sich mit sehr viel Herzblut für jeden seiner Mitarbeiter einsetzt. Auch fachlich habe ich, besonders in den ersten Jahren beim DLR, sehr stark von seinem enormen Wissen profitiert und gelernt. Danke dafür.

Während der Promotion durfte ich auch die Studenten Fabian Küppers, Erich Reimer und Pascal Eifinger betreuen. Ich glaube von allen Dingen die ich während dieser Zeit gemacht habe, hat mir eure Betreuung am meisten Spaß gemacht und mich auch sehr weitergebracht. Auch euch vielen Dank für die Hilfe und die tolle Zeit.

Meiner Frau möchte ich dafür danken, dass sie die Fehler, die durch meine bescheidenen Sprach- und Rechtschreibfähigkeiten entstanden sind, verbessert und korrigiert hat. Diesen Satz würde sie wohl auch sofort korrigieren.

Zu guter Letzt, bleibt dann nur noch zu sagen: Endlich geschafft und großen Respekt an alle Kollegen, die dies auch bewältigt haben!

Nomenklatur

Abkürzungen

AVX	Advanced Vector Extensions, eine Befehlssatzerweiterung für Prozessoren. Nachfolger der SSE Architektur.
CFD	Computational Fluid Dynamics
CSM	Computational Structure Mechanics
EGO	Efficient Global Optimization. Ein Optimierungsverfahren, welches die Unsicherheiten in einer Ersatzmodellvorhersage miteinbezieht
EVG	Erwarteter Volumenzugewinn, auch Expected Volume Gain oder Expected Hypervolume Improvement
GPGPU	General Purpose Computation on Graphics Processing Unit. Beschreibt die Verwendung von GPUs über den ursprünglichen Einsatz hinaus. Meist für die Verwendung von massiv parallelen und rechenintensiven Algorithmen.
GPU	Abkürzung für Graphics Processing Unit. Grafikbeschleuniger, in der Regel als Zusatzkarte in einem Rechner
HPC	High-Performance-Computing
Member	Ein Satz freier Variablen mit dazugehörigen Funktionswerten. Beispielsweise ein Satz geometrischer Parameter mit berechneten mechanischen und aerodynamischen Größen.
MO	Multi-Objective
ROI	Region of Interest. Ein für die Zielfunktionen festgelegtes Gültigkeitsintervall.
RPROP	Resilient Backpropagation. Ein Optimierungsalgorithmus erster Ordnung
SIMD	Single Instruction Multiple Data, eine Architektur welche es erlaubt dieselbe Operation parallel auf einen sich verändernden Datenstrom anzuwenden.
SSE	Streaming SIMD Extensions, eine Befehlssatzerweiterung für Prozessoren.

Formelzeichen

δ	Kronecker Delta

κ	Konditionszahl
λ	Diagonalaufschlag
$\mathbf{Cov}$	Kovarianzmatrix
$\mathbf{R}$	Korrelationsmatrix
ω	Druckverlustbeiwert
σ^2	Varianz
$\mathrm{cov}(\vec{x}, \vec{y})$	Ortsabhängige Kovarianzfunktion zwischen den Ortsvektoren x und y
$\mathrm{cov}(X, Y)$	Kovarianzfunktion zwischen den Zufallsvariablen X und Y
$\mathrm{var}(X)$	Varianz der Zufallsvariable X
$\vec{\beta}$	Beta Vektor, beinhaltet alle Erwartungswerte des Modells.
$\vec{\theta}$	Hyperparameter einer Korrelationsfunktion, ohne Varianzen oder Regularisierungsterme
$\vec{c}$	Kovarianzvektor
$\vec{r}$	Korrelationsvektor
$\vec{w}$	Gewichte eines Kriging Modells
$\vec{x}$	Ortsvektor
$\vec{y}_s$	Vektor, welcher alle bekannten Stützstellen enthält
Ξ	Eigenwert einer Matrix
$c(X, Y)$	Korrelationsfunktion zwischen den Zufallsvariablen X und Y
$E[X]$	Erwartungswert der Zufallsvariable X
$F(\vec{x})$	Fehlerfunktion an der Stelle $\vec{x}$
$f_{dec}(...)$	Entscheidungsfunktion
P_t	Totaldruck
T_t	Totaltemperatur
$y(\vec{x})$	Bekannter Funktionswert oder Stützstelle an der Stelle $\vec{x}$
$y^*(\vec{x})$	Geschätzter Funktionswert an der Stelle $\vec{x}$
a	Skalierungsfaktor für das CO-Kriging Kovarianzmodel
c	Anzahl der Nebenbedingungen
h	Allgemeiner Hyperparameter

I	Informationsgehalt
k	Anzahl der freien Variablen eines Members
L	Likelihood Funktion
m	Anzahl der gegebenen partiellen Ableitungen
N	Multivariate Normalverteilung
n	Anzahl der beprobten Stützstellen, auch Member oder Samples genannt
o	Anzahl der Hyperparameter
P	Wahrscheinlichkeit
q	Anzahl der verwendeten Ersatzmodelle innerhalb einer Optimierung
s	Anzahl der verschiedenen Gütestufen bei einem Multifidelty Modell
t	Zeit
w	Anzahl der Flowparameter
z	Anzahl der Zielfunktionen einer Optimierung

Inhaltsverzeichnis

Kurzfassung I

Abstract II

Vorwort III

1 Einleitung 1

1.1 Motivation 1

1.2 Stand der Forschung 2

1.3 Anforderungen und Zielsetzung 7

2 Optimierungsstrategie für Turbomaschinen 10

2.1 Grundlagen Turbomaschinenoptimierung 10

2.1.1 Ingenieurwissenschaftliche Disziplinen in der Turbomaschinenoptimierung 10

2.1.2 Begriffserläuterungen zu Optimierungen 11

2.2 Besondere Herausforderungen bei Turbomaschinenoptimierungen . . . 14

2.3 Automatisierte Optimierung im DLR 16

2.3.1 Grundlegender Optimierungsprozess 16

2.3.2 Ersatzmodellbeschleunigung 18

3 Multifidelity Optimierungsstrategie Turbomaschine 21

3.1 Gütestufen in der Turbomaschinenauslegung 21

3.1.1 Beispiel für verschiedene Gütestufen innerhalb eines Simulationsverfahrens 22

3.1.2 Beispiel für verschiedene Simulationsverfahren 24

3.2 Multifidelity-Optimierungsstrategie in AutoOpti 25

3.2.1 Multifidelity-Optimierungsprozess 25

3.2.2 Entscheidungsfunktion 26

3.2.2.1 Benutzergesteuerte Entscheidungsfunktionen 27

3.2.2.2 Ersatzmodellgesteuerte Entscheidungsfunktionen . . . 28

3.2.3 Theroretische Beschleunigung einer Multifidelity Optimierung . . 34

4 Kriging-Verfahren: Analytische Herleitung 36
4.1 Grundlagen Kriging 36
4.2 Kriging: Analytische Herleitung 37
4.2.1 Allgemeine Vorbemerkungen und Definitionen 37
4.2.2 Herleitung Krige-Schätzer 39
4.2.3 Erwartungswert 41
4.2.4 Varianz des Schätzfehlers 41
4.2.5 Kovarianz zwischen zwei Vorhersagen 42
4.3 Umsetzung der Kriging-Unterverfahren 42
4.3.1 Co-Kriging 42
4.3.2 Ordinary-Kriging 43
4.3.3 Gradient-Enhanced-Kriging 43
4.4 Vergleich des hier vorgestellten Co-Kriging-Verfahrens mit anderen Verfahren aus der Literatur 45

5 Implementierung der Verfahren 47
5.1 Modellierung der Kovarianzfunktion 47
5.1.1 Ortsabhängigkeit Kovarianzfunktion 48
5.1.2 Parametrisches Kovarianzmodell 49
5.1.2.1 Kovarianzfunktion Co-Kriging 51
5.1.2.2 Aufteilung der Hyperparameter in Klassen 51
5.2 Regularisierung und Behandlung verrauschter Funktionen 52
5.2.1 Regularisierung 52
5.2.2 Behandlung verrauschter Funktionen 55
5.3 Training 58
5.3.1 Maximum-Likelihood für alle Kriging-Verfahren 58
5.3.2 Likelihood-Schätzer für Erwartungswerte und Varianzen 59
5.3.3 Initialisierung der Hyperparameter für alle Krigingmodelle 61
5.3.4 Minimierungsverfahren 64
5.4 Umsetzung der Entscheidungsfunktion 68
5.5 Softwaretechnische Umsetzung 71
5.5.1 Softwarearchitektur und Laufzeitumgebung 71
5.5.2 Verteiltes Rechnen 76
5.5.3 Effiziente Berechnung der Kovarianzfunktion 80
5.5.4 Reduktion von Stützstellen 83
5.5.5 Likelihood: Inverse durch Gleichungssysteme ersetzen 84

5.5.6 Ableitung Likelihood: Verzicht auf Vor- und Rückwärtssubstitution 85
5.5.7 Vergleich zwischen der Invertierung und der Rückwärtsdifferenzierung . . . 88
5.5.8 Ableitung Likelihood: Verzicht auf Inverse - Approximation . . . 90
5.5.9 Verwendung von GPUs . . . 90
5.6 Laufzeit-Analysesoftware für Krigingmodelle . . . 96

6 Anwendungen 99
6.1 Analytische Testfälle . . . 99
6.1.1 Co-Kriging Testfall . . . 99
6.1.2 Analytische Optimierungsbeispiele . . . 100
6.2 Optimierung einer Fan-Stufe . . . 106
6.2.1 Optimierungssetup . . . 107
6.2.2 Ergebnisse . . . 110
6.3 Übersicht der bisherigen Anwendungen in Industrie und Forschung . . . 117
6.3.1 „Design of a counter rotating fan using a multidisciplinary and multifidelity optimisation under high level of restrictions“ . . . 117
6.3.2 „High-dimensional multi-fidelity optimisation of a 2.5 stage low pressure compressor“ . . . 119
6.3.3 „Entwicklung und Anwendung von modernen optimierungsfähigen Auslegungsverfahren unter Berücksichtigung des Realgaseinflusses“ . . . 120
6.3.4 „Multi-fidelity Method for Aerodynamic Shape Optimisation of Axial Compressor Blades“ . . . 122
6.3.5 „Optimizing Surge Margin and Efficiency of a Transonic Compressor“ . . . 124

7 Zusammenfassung 128

Literaturverzeichnis 132

A Anhang 144
A.1 Kapitel 3 . . . 144
A.1.1 Aerodynamische Größen . . . 144
A.1.2 Beispiel Entscheidungsfunktion globales/lokales Varianzkriterium 144
A.2 Kapitel 4 . . . 149
A.2.1 Gauss Korrelationsfunktion mit Ableitungen . . . 149
A.2.2 Exponential Korrelationsfunktion mit Ableitungen . . . 150
A.2.3 Kubische Spline Korrelationsfunktion mit Ableitungen . . . 151
A.2.4 Herleitung der Kriging-Gewichte . . . 157

A.2.5 Beispiel Dichtefunktion für ein CO-Kriging Model 158
A.2.6 Varianz der Fehlerfunktion 159
A.2.7 Kovarianz zwischen Vorhersagen: Allgemeine Formulierung . . . 162
A.2.8 Kovarianz zwischen Vorhersagen: Gewichte eingesetzt 162
A.2.9 Varianz des Schätzfehlers 164
A.2.10 Co-Kriging Kovarianzfunktion Herleitung 167
A.3 Kapitel 5 169
A.3.1 Ableitungen des Kovarianzmodells nach den Rauschtermen . . . 169
A.3.2 Likelihood Schätzer Erwartungswerte und Varianz 169
A.3.3 Beweis: Variogramm Abhängigkeit von Distanz 170
A.3.4 Differentiation der Likelihood-Funktion 172
A.3.5 Likelihood Schätzer Varianzen im CO-Kriging 172
A.3.6 Vereinfachung der Vorhersagegleichung 173
A.3.7 Iterative Varianzbestimmung innerhalb des Co-Kriging Trainingsverfahrens 174
A.3.8 Restandardisierung der Hyperparameter 175
A.3.9 UML Diagramm: ZMQClient Klasse 177
A.3.10 UML Diagramm: Point 178
A.3.11 UML Diagramm: CorrelationFunction 178
A.3.12 UML Diagramm: DensityFunction 179
A.3.13 UML Diagramm: MinimizerInterface 180
A.3.14 UML Diagramm: Matrix 181
A.3.15 UML Diagramm: ZMQConnection, ZMQServer, MatrixServerFunctions 182
A.3.16 Cholesky Zerlegung 183
A.3.17 Likelihood: Inverse durch Gleichungssysteme ersetzen 183
A.3.18 Wirtschaftliche Betrachtung CPU/GPU 184
A.3.19 GPU Resourcenverteilung bei parallelen Trainings 185
A.3.20 Benchmark Rückwärtsdifferenzierung Likelihood 187
A.4 Kapitel 6 187
A.4.1 Testfunktion ZDT3 187
A.4.2 Zusätzliche Abbildungen der Fanstufenoptimierung 188

1 Einleitung

1.1 Motivation

Der Luftverkehr im Fracht- und Passagierbereich ist in den letzten Jahren sehr stark angestiegen. Gleichzeitig steigt auch der weltweite Energieverbrauch [p.l.c., 2016]. In beiden Bereichen stellen Turbomaschinen eine sehr wichtige technische Komponente dar. Das Thema Klimaschutz ist zunehmend in den Vordergrund gerückt. Die Gesellschaft und die Politik stellen hier sehr hohe Anforderungen, welche bei der Auslegung von Turbomaschinen mit berücksichtigt werden müssen. Außerdem herrscht ein sehr hoher Konkurrenz- und Preisdruck unter den Triebwerks- und den Energiemaschinenherstellern. Diese konkurrierenden Anforderungen machen die Auslegung von Turbomaschinen zu einer sehr großen Herausforderung für Forschung und Industrie und erfordern bereits während des Auslegungsprozesses ein besonderes Augenmerk auf Emissionen, Effizienz, Betriebs- und Wartungskosten.

Um diesen Herausforderungen zu begegnen, sind moderne CFD-, CSM- und Optimierungsverfahren ein sehr wichtiges Werkzeug. Sie erlauben es, die komplexen Strömungsvorgänge in Turbomaschinen am Computer zu simulieren, strukturmechanische Kennwerte zu ermitteln und somit die Turbomaschine unter Berücksichtigung multipler Ziele und Nebenbedingungen zu optimieren. Diese noch relativ neuen Werkzeuge führen zu einer Verlagerung von sehr kostspieligen experimentellen Untersuchungen hin zu Simulationen am Computer durch CSM- und CFD-Verfahren.
Durch die neuen numerischen Verfahren wurden die Auslegungszyklen im Turbomaschinenbereich in den letzten Jahren nicht nur beschleunigt, sondern auch robuster. Allerdings sind die genutzten Verfahren mit einem so hohen numerischen Aufwand verbunden, dass oftmals vereinfachte Modelle verwendet werden. Durch diese Vereinfachungen können diese Modelle allerdings nicht alle physikalischen Phänomene auflösen.

Zur Verwendung von genaueren und damit auch komplexeren Modellen werden zunehmend sehr große HPC-Cluster eingesetzt, welche die benötigte Rechenleistung zur Verfügung zu stellen. Solch ein Cluster ist allerdings kostspielig und die Optimierungs- und CFD-Verfahren müssen auf die spezielle Hardware angepasst werden. Diese Problemstellung ist so komplex, dass sich daraus ein eigener Forschungszweig entwickelt hat. Dennoch ist man trotz der enormen Rechenleistung, die moderne HPC-Cluster bieten, immer noch auf stark vereinfachte CFD-Modelle und hocheffiziente Optimierungsverfahren angewiesen. Beispielsweise ist man von dem Ziel, ein vollständiges Flugzeug inklusive Antrieb am Rechner zeitlich aufgelöst zu simulieren und zu optimieren noch sehr weit entfernt. Selbst die Optimierung eines

vollständigen Triebwerks ist mit den momentan zur Verfügung stehenden Ressourcen nicht durchführbar.

Die wissenschaftliche Herausforderung liegt also in der Methodenentwicklung und Anwendung von Optimierungsstrategien, welche die industriellen Anforderungen berücksichtigen. So soll es möglich werden, komplette Komponenten bzw. Subsysteme von Gasturbinen auszulegen und zu optimieren. Zu diesem Zweck wird ein effizientes Optimierungsverfahren entwickelt und getestet, wobei ein besonderes Augenmerk auf die HPC-Hardwareumgebung und eine hohe numerische Effizienz gelegt wird.

1.2 Stand der Forschung

Industriell genutzte Optimierungsverfahren für den Turbomaschinenbereich sind schon seit längerer Zeit ein umfangreiches Forschungsgebiet und Thema zahlreicher wissenschaftlicher Arbeiten. Eine vollständige Übersicht der vorhandenen Literatur ist daher nahezu unmöglich. Dennoch soll dem Leser ein kurzer Überblick über einige aktuelle Forschungsarbeiten im Bereich der Turbomaschinenoptimierung gegeben werden.

Bei Optimierungen im Turbomaschinenbereich handelt es sich meist um nichtlineare, mehrdimensionale, multidisziplinäre Mehrziel-Optimierungsprobleme mit einer hohen Zahl an Nebenbedingungen. Es werden oftmals Geometrien für Turbomaschinenkomponenten gesucht, welche einen besonders hohen Wirkungsgrad aufweisen. In der Regel unter Einhaltung aerodynamischer, mechanischer, akustischer oder anderer Restriktionen. Aufgrund der Komplexität der hier beschriebenen Optimierungsprobleme sind in den letzten Jahren zahlreiche Optimierungsverfahren entwickelt und erfolgreich für die Turbomaschinenauslegung eingesetzt worden. Innerhalb dieses Abschnitts werden die in der Turbomaschinenforschung aktuell verwendeten Verfahren kurz beschrieben und zu jedem Verfahren allgemeine Übersichtsliteratur vorgestellt sowie anschließend einige Veröffentlichungen zu industrienahen Anwendungen dieser Verfahren aus den letzten sieben Jahren gezeigt.

Optimierungen auf Basis der Evolutionsstrategie: Bei evolutionären Optimierungsalgorithmen handelt es sich um stochastische Optimierungsverfahren, welche der Funktionsweise der natürlichen Evolution von Lebewesen nachempfunden sind. Im Wesentlichen werden bei einem evolutionären Algorithmus aus einem vorhandenen Pool bekannter Daten die besten selektiert, anschließend rekombiniert und/oder mutiert und wieder evaluiert. Dieser Prozess wird solange wiederholt, bis ein zufriedenstellendes Ergebnis erreicht ist. Die Vorteile dieses Verfahrens liegen in der Suche nach einem globalen Optimum ohne dass viele Anforderungen an Zielfunktionen und Nebenbedingungen gestellt werden. Die Zielfunktion muss weder differenzierbar noch an jeder Stelle auswertbar sein. Dies macht das Verfahren sehr robust. Der größte Nachteil bei dieser Art von Optimierung liegt in der Konvergenzgeschwindigkeit: Es sind meist sehr viele Tastschritte nötig, um ein zufriedenstellendes Optimum zu finden.

Grundlegende Literatur über evolutionäre Algorithmen bieten zum einen die beiden Arbeiten von Holland [Holland, 1975] und Rechenberg [Rechenberg, 1973], welche den

Grundstein der heutigen modernen Algorithmen darstellen. Zum anderen bieten die Bücher von [Gen and Cheng, 2000], [Weicker, 2015] und [Gill et al., 1981, Gill, 2007] einen umfassenden und aktuellen Überblick über die gesamte Thematik.

Die Arbeit von Sozio et al. [Sozio et al., 2013] stellt ein aktuelles und industrienahes Beispiel dar. Innerhalb dieser Arbeit wird ein Auslegungsprozess für gegenläufige Axialturbinen auf Basis eines evolutionären Algorithmus vorgestellt. Dieser Prozess erzeugt ausgehend von eindimensionalen Kenngrößen eine dreidimensionale, aerodynamisch optimierte Geometrie. Hierfür nutzt dieser mehrere seriell ausgeführte evolutionäre Optimierungen. Sozio et. al. konnten mit ihrem Auslegungsprozess eine dreidimensionale, aerodynamisch sinnvolle Geometrie einer mehrstufigen Axialturbine erzeugen.

Beschleunigung von Optimierungen mit Ersatzmodellen: Bei Ersatzmodellen handelt es sich um Interpolations- oder Approximationsverfahren, welche während einer Optimierung Vorhersagen über unbekannte Parametersätze machen. So können besonders vielversprechende Parametersätze schneller gefunden werden als bei der Verwendung einer reinen Evolutionsstrategie.

Die Arbeit von Queipo et al. [Queipo et al., 2005] befasst sich unter anderem mit der Initialisierung von Optimierungen, der Auswahl geeigneter Ersatzmodelle für verschiedene Anwendungen und deren Verwendung innerhalb von multidisziplinären Optimierungen und bietet damit einen sehr umfassenden Überblick über dieses Themengebiet. In dem Beitrag von Simpson et al. [Simpson et al., 2008] wird die historische Entwicklung von ersatzmodellgestützten multidisziplinären Optimierungen innerhalb der letzten 20 Jahre beschrieben. Simpson et al. kommen zu dem Schluss, dass Herausforderungen wie der „Fluch der Dimensionen“ oder die Komplexität der numerischen Verfahren immer noch dieselben sind wie vor 20 Jahren. Sie heben allerdings hervor, dass die Größe dieser Herausforderungen deutlich gestiegen ist (vgl. [Simpson et al., 2008], Seite 14).

Der Beitrag von Forrester et al. [Forrester and Keane, 2009] bietet einen technisch fundierten Einblick in diesen Bereich und beschreibt mehrere Ersatzmodellverfahren und deren Anwendung im Luft- und Raumfahrtbereich. Es wird ebenfalls ein kurzer Einblick in die Nutzung von Gradienteninformation und mehreren Gütestufen gegeben.

Verstraete et al. [Verstraete et al., 2014] zeigen die erfolgreiche Anwendung von neuronalen Netzwerken bei der Optimierung eines Diffusors einer Niederdruckdampfturbine. Die durchgeführte Optimierung hat zum einen den Wirkungsgrad erhöht und zum anderen den Betriebsbereich der Maschine erweitert. Weiterhin wurde die Frage behandelt, inwiefern die Verwendung von mehreren Betriebspunkten eine solche Optimierung beeinflusst. Das verwendete Optimierungsverfahren wird im Von-Karman-Institut in Belgien entwickelt und in den Arbeiten von Pierret [Pierret and Van den Braembussche, 1998, Pierret, 1999] und Verstraete et al. [Verstraete, 2008] beschrieben. Verstraete et al. kommen zu dem Schluss, dass die Berücksichtigung mehrerer Betriebspunkte eine verbesserte Charakteristik des Betriebsverhaltens (vgl. [Verstraete et al., 2014], Seite 9) erzeugt.

Die Arbeit von Aulich et al. [Aulich and Siller, 2011] beschreibt die Optimierung einer hochbelasteten Fan-Stufe mit über 230 Design-Parametern und zahlreichen Neben-

bedingungen. Das dort angewendete Verfahren ist ein durch Ordinary-Kriging beschleunigter evolutionärer Algorithmus. Beim Ordinary-Kriging handelt es sich um ein robustes statistisches Interpolationsverfahren das nicht nur den reinen Funktionswert voraussagt, sondern auch eine statistische Verteilung dessen. Praktisch bedeutet dies, dass eine Abschätzung über den Fehler der Vorhersage gemacht werden kann. Historisch sind die ersten Ansätze statistischer Ersatzmodelle innerhalb der mathematischen Geologie [Cressie, 1990, Cressie, 1993, Matheron, 1963] und der Statistik [Sacks and Ylvisaker, 1970, Sacks et al., 1989] zu finden. Die ersten Ansätze einer globalen Optimierungsstrategie auf Basis von statistischen Modellen ist in dem Seminarbeitrag von [Kushner, 1964] zu finden und wurde dann von vielen Wissenschaftlern aufgegriffen und weiterentwickelt [Cox and John, 1992, Mockus, 1994, Perttunen, 1991, Jones et al., 1998, Gibbs and MacKay, 1997].

Das EGO-Verfahren (für Efficient Global Optimization) [Jones et al., 1998, Jones, 2001] hat in modernen Optimierungsalgorithmen besondere Bedeutung erlangt. Dieses Verfahren ermöglicht eine gute Balance zwischen der Exploration und Exploitation innerhalb globaler Optimierungen. Die Arbeit von Voß et al. [Voß et al., 2014] nutzt einen solchen EGO-Algorithmus innerhalb einer aeromechanischen Optimierung eines existierenden transsonischen Radialverdichterlaufrades. Die durchgeführte Optimierung übertrifft den Wirkungsgrad der ursprünglichen Geometrie um ca. 2% und auch der Sperrmassenstrom wurde um ca. 7% gesteigert. Die darauf aufbauende Arbeit von Elfert et al. [Elfert et al., 2016] bestätigt diese Ergebnisse experimentell.

Eine weitere Nutzung des EGO-Algorithmus wird in der Arbeit von Lepot et al. [Lepot et al., 2011] dargestellt. Es wird die aeromechanische Optimierung eines gegenläufigen offenen Rotors unter Berücksichtigung von akustischen Kriterien beschrieben. Weitere aktuelle Beiträge zu Anwendungen dieses Verfahrens sind z.B. in [Keane, 2006, Sacks et al., 2007, J. Forrester et al., 2006, Forrester and Keane, 2009] zu finden.

In der Arbeit von Baert et al. [Baert et al., 2015] wird ein Verfahren zur Optimierung von kontinuierlichen und nicht kontinuierlichen Designparametern (z.B. ganzzahlige Parameter wie Schaufelzahlen) auf Basis von Ersatzmodellen vorgestellt und an einem Bypass-Kanal getestet. Baert et al. kommen zu dem Schluss, dass diese Art von Ansatz wichtige Design-Entscheidungen in die Optimierung integriert (vgl. [Baert et al., 2015], Seite 11-12) und belegen dies mit der erfolgreichen Optimierung eines Bypass-Kanals.

Eine andere Sicht auf ersatzmodellgestützte Optimierungen bietet der Beitrag von Chahine et al. [Chahine et al., 2012]. Die Autoren stellen den Nutzen von neuronalen-Netzwerken zur Beschleunigung von evolutionsbasierten Optimierungen in Frage und belegen diese Aussage mit der Vergleichsoptimierung eines Radialverdichters. Dieses Ergebnis ist allerdings verwunderlich, da die Verwendung von Ersatzmodellen zur Beschleunigung von evolutionären Optimierungsverfahren als besonders effizientes Verfahren gilt. Es sei aber zu beachten, dass die Effizienz neuronaler Netzwerke äußerst abhängig von deren Aufbau und Struktur ist. Dies ist Gegenstand zahlreicher aktueller Forschungsarbeiten im Bereich des maschinellen Lernens, siehe z.B. die Arbeit von Silver et al. [Silver et al., 2016]. Verfahren wie das Kriging sind in dieser Hinsicht robuster. Dennoch zeigt das Ergebnis sehr deutlich, dass die Effizienz von Optimierungen

mit Ersatzmodellen sehr unterschiedlich ausfallen kann und stark von den verwendeten Optimierungsverfahren, Ersatzmodellen und auch deren Anwendung abhängt. Die Komplexität des Verfahrens wird durch den Einsatz von Ersatzmodellen deutlich erhöht. Dies erfordert in der Regel einen erfahrenen Anwender zur Überwachung solcher Optimierungen. Innerhalb dieser Arbeit werden Möglichkeiten der Automatisierung und der Analyse gezeigt, die diese Verfahren auch einem breiteren Anwenderbereich zugänglich machen sollen.

Gradienteninformation in Optimierungen: Viele Forschungsbeiträge beschäftigen sich mit der Verwendung von Gradienteninformation in Optimierungen. Moderne Strömungslöser gestatten immer häufiger die effiziente Berechnung partieller Ableitungen von Funktionalen, wie z.B. Wirkungsgrad nach geometrischer Deformation. Die numerischen Kosten für die Bestimmung der partiellen Ableitungen befinden sich hierbei in einer ähnlichen Größenordnung wie die Strömungslösung selbst. In einer Optimierung mit wenig Funktionalen und vielen Parametern, können diese Gradienteninformationen gewinnbringend verwendet werden. Dies wird dadurch erreicht, dass von einem beliebigen Startpunkt in Gradientenrichtung abgestiegen wird und somit schnell ein lokales Minimum gefunden werden kann. Eine andere Möglichkeit ist die Verwendung von Ersatzmodellen. Einige Ersatzmodelle können die vorhandene Gradienteninformation zur Verbesserung der Vorhersagen heranziehen und bieten den zusätzlichen Vorteil, dass keine vollständige Gradienteninformation vorhanden sein muss. Dies ist wichtig, da in den meisten Turbomaschinenoptimierungen für wesentliche Funktionale keine Gradienteninformation verfügbar ist.

Ein aktuelles Beispiel in diesem Bereich bietet die Arbeit von Willeke et al. [Willeke and Verstraete, 2015]. Es wird eine Optimierung beschrieben, welche den Druckverlust eines Kühlkanals mithilfe eines Gradientenabstiegverfahrens optimiert und mit einem rein evolutionären Algorithmus vergleicht. Bei dem angestellten Vergleich konnte sich das Gradientenverfahren als das schnellere behaupten. Weiterhin schreiben Willeke et al., dass die Sorge in ein lokales Minimum zu geraten, in dem analysierten Testfall nicht relevant ist, da der verwendete Ansatz in sehr kurzer Zeit eine ausreichende Verbesserung erzielen kann (vgl. [Willeke and Verstraete, 2015], Seite 11-12).

In dem Beitrag von Tang [Tang et al., 2016] wird eine ersatzmodellgestützte Optimierung mit einem Gradientenabstiegsverfahren verglichen. Das verwendete Ersatzmodell basiert auf radialen Basisfunktionen, welche die vorhandene Gradienteninformation verarbeiten können. Tang kommt in seiner Arbeit zu dem Schluss, dass das reine Abstiegsverfahren sehr schnell ein lokales Minimum findet und sich nicht sehr weit von der initialen Geometrie wegbewegt. Das ersatzmodellgestützte Verfahren konnte im selben Zeitraum die Region des globalen Optimums auffinden (vgl. [Tang et al., 2016], Seite 12).

Erwähnenswert sind innerhalb der Verfahren der gradientenbasierten Optimierung auch die sogenannten One-Shot-Verfahren. Im Gegensatz zu den klassischen Optimierungsverfahren wird dabei innerhalb des Iterationsverfahrens einer Strömungssimulation eine optimale Geometrie gesucht. Hierfür sind innerhalb der Lösung ebenfalls Gradienteninformationen notwendig und das verwendete Rechennetz muss deformiert werden. Diese Verfahren sind in der Regel um ein Vielfaches schneller als klassische

Optimierungsansätze. Allerdings sind multidisziplinäre Problemstellungen, instationäre Strömungslösungen und auch Berechnungen mit verschiedenen gekoppelten Simulationsverfahren eine sehr große Herausforderung. Daher bieten diese Verfahren oftmals nicht die benötigte Flexibilität einer „konventionellen“ Optimierung. Für eine aktuelle und umfassende Übersicht in dem Bereich der One-Shot-Verfahren sei hier auf die Disseration von Özkaya [Özkaya, 2014] verwiesen.

Multifidelity-Verfahren: Innerhalb dieser Arbeit wird ein ersatzmodellgestütztes Multifidelity-Verfahren entwickelt, welches in der Lage ist, während einer Optimierung verschiedene Gütestufen von Daten mit Hilfe von Ersatzmodellen zu verarbeiten. Mögliche Gütestufen basieren z.B. auf der örtlichen Diskretisierung eines zu simulierenden Strömungskanals in der Form eines Rechengitters oder auch die Verwendung von stationärem und zeitgenauen CFD-Simulationen. Mit steigender Anzahl der Zellen eines Rechengitters werden Strömungsphänomene besser aufgelöst, was allerdings mit einer erheblich gesteigerten Rechenzeit einhergeht. Der Anwender muss vor einer Optimierung immer einen Kompromiss zwischen Genauigkeit und Geschwindigkeit treffen.

In der Regel sind die Gütestufen korreliert, oftmals aber verzerrt, verschoben oder skaliert. Es ist also ein Ersatzmodell notwendig, welches die Ergebnisse der verschiedenen Gütestufen verarbeiten kann und die Transferfunktion zwischen diesen automatisch auffindet. Die einfachste Möglichkeit eines solchen Ersatzmodells stellen sogenannte Brückenfunktionen dar. Eine solche Brückenfunktion wird aus zwei Ersatzmodellen gebildet. Das eine stellt die niedrigere Stufe dar und das andere die Differenz zwischen hoher und niedriger Stufe. Addiert man die Vorhersagen beider Modelle, so bekommt man eine Vorhersage für die höchste Gütestufe. Je nach Komplexität der Differenzfunktion können bereits mit sehr wenigen Daten der höherwertigen Funktion gute Vorhersagen aus den Daten niedriger Güte gewonnen werden. Beispielsweise verwendet die bekannte Open-Source-Optimierungssoftware Dakota solche additiven Brückenfunktionen in der aktuellen Version 6.5 (vgl. [Adams et al., 2016]).

Ein statistisches Verfahren, welches auch komplexere Zusammenhänge beschreiben kann, ist das CO-Kriging-Ersatzmodell. Hier sei besonders auf die grundlegenden Arbeiten von [Kennedy and O’Hagan, 2000] verwiesen. In der Arbeit von Han et al. [Z.-H. Han, R. Zimmermann, 2010] wird ein Vergleich zwischen einem CO-Kriging-Modell und einer additiven Brückenfunktion angestellt, wobei sich das CO-Kriging-Modell als das leistungsfähigere herausgestellt hat. Ebenso wie das Ordinary-Kriging, sagt auch das CO-Kriging-Modell eine Verteilung voraus und lässt sich damit in dem „Efficient-Global-Optimization“-Verfahren verwenden.

Es gibt bereits Umsetzungen des CO-Kriging Verfahrens, wobei die meisten auf der Arbeit von Kennedy und O’Hagan [Kennedy and O’Hagan, 2000] beruhen. Für dieses CO-Kriging-Modell muss an jeden bekannten Ort der höchsten Gütestufe auch eine Stützstelle der niedrigeren Gütestufen vorhanden sein. Innerhalb der vorliegenden Arbeit wird eine CO-Kriging-Variation vorgestellt, die dies nicht mehr benötigt.

Eine effiziente Umsetzung der Methode von [Kennedy and O’Hagan, 2000] wird in der Arbeit von Le Gratiet [Le Gratiet, 2013] dargestellt. Weiterhin existieren noch die Arbeiten von Han et al. [Han et al., 2012, Z.-H. Han, R. Zimmermann, 2010] welche allerdings nur mit starken Einschränkungen anwendbar sind (siehe Kapitel 4.4).

Neben der technischen Umsetzung des CO-Kriging-Verfahrens ist auch die Verwendung eines solchen Modells innerhalb einer Multifidelity-Optimierung ein sehr wichtiger Punkt. Dabei sind die Entscheidung, wann welche Gütestufe berechnet wird sowie die Daten- und Prozesskettenverwaltung die wesentlichen Herausforderungen. Leider existieren zu dieser Thematik nur wenige Forschungsbeiträge. Eine akademische Anwendung bietet hier die Arbeit von Brooks [Brooks et al., 2011]. Diese stellt einen Vergleich zwischen einer Multifidelity-Optimierung und einer konventionellen Ordinary-Kriging-Optimierung an. Brooks et al. optimierten für diesen Zweck den bekannten NASA-Rotor 37 [Reid and Moore, 1978] mit dem Ziel den Wirkungsgrad zu verbessern, wobei das originale Druckverhältnis und der Massenstrom beibehalten werden sollten. Um eine Vergleichbarkeit gewährleisten zu können, stellten Brooks et al. für beide Optimierungen dasselbe Rechenbudget zur Verfügung. Weiterhin wurde innerhalb der Optimierung ein industriell genutztes 3D-CFD-Verfahren verwendet [Lapworth and Shahpar, 2004]. Bei gleicher Rechenzeit konnte die Multifidelity-Optimierung eine Steigerung des Wirkungsgrads um 2.34% erreichen, wobei die Referenzoptimierung eine Steigerung von 1.79% erzielte. Dieses Ergebnis lässt darauf hoffen, dass bei komplexeren industriellen Optimierungen ein deutlicher Geschwindigkeitszuwachs zu erwarten ist.

1.3 Anforderungen und Zielsetzung

Im Rahmen dieser Arbeit wird ein industriell einsetzbares und erweiterbares Multifidelity-Optimierungsverfahren entwickelt und getestet. Das Verfahren wird in die bereits bestehende Optimierungssoftware AutoOpti integriert und die speziellen Anforderungen im Bereich der Turbomaschinenentwicklung berücksichtigt. In diesem Abschnitt werden zuerst die Anforderungen an das entwickelte Verfahren herausgearbeitet und daraus folgend eine detaillierte Zielsetzung abgeleitet.

Die Anforderungen lassen sich hierbei in zwei Kategorien unterteilen. Die eine umfasst die Anforderungen durch die spezielle Problematik in der Turbomaschinenauslegung. Besonders kritische Punkte sind hierbei die sehr aufwendigen Prozessketten, die hochdimensionalen Suchräume und die Vielzahl an auftretenden physikalischen Disziplinen. Die andere Kategorie betrifft die computertechnischen Anforderungen, wie die Unterstützung einer HPC-Umgebung oder eine sinnvolle Integration in die bereits vorhandene Software AutoOpti.

Anforderungen Turbomaschinenoptimierung: Die speziellen Anforderungen von Turbomaschinenoptimierungen können wie folgt zusammengefasst werden:

- Mehrere Zielfunktionen und physikalische Disziplinen
- Hohe Anzahl an Restriktionen
- Partielle Gradienteninformationen
- Hochdimensionale Suchräume
- Lange Prozesskettenzeiten
- Prozessketten liefern nicht immer Ergebnisse

Durch das Multifidelity-Verfahren sollen insbesondere die hochdimensionalen Such-

räume effizienter abgetastet werden können und dadurch der Aufwand durch die numerisch aufwendigen Prozessketten akzeptabel werden und so ein verlässlicheres Ergebnis bei der Produktentwicklung liefern.

Software- und hardwaretechnische Anforderungen: Die Entwicklung eines solchen Verfahrens bringt auch zahlreiche softwaretechnische Anforderungen mit sich. Im Folgenden sind die wichtigsten Punkte aufgelistet:

- Integration in die bestehende Optimierungssoftware AutoOpti
- Unterstützung einer HPC-Rechnerarchitektur mit GPUs (Graphics Processing Unit)
- Offene objektorientierte Softwarestruktur
- Hohe Effizienz des Optimierungsverfahrens, insbesondere des Ersatzmodells

Die Software AutoOpti ist vollständig in der Programmiersprache C99 geschrieben. Zusammen mit der Forderung nach einer objektorientierten Softwarelösung wurde für die Entwicklung des neuen Verfahrens die Programmiersprache C++ inklusive der Verwendung der umfangreichen Boost Bibliothek [Karlsson, 2006] gewählt. Der Vorteil liegt darin, denselben Compiler verwenden zu können und den AutoOpti-Entwicklern eine verwandte Sprachumgebung anzubieten.

Weiterhin werden Optimierungen im Turbomaschinendesign aufgrund der komplexen numerischen Prozesskette meist auf HPC-Großrechnern ausgeführt. Die innerhalb dieser Arbeit entwickelte Software muss in einer solchen Umgebung lauffähig sein. In den letzten Jahren ist besonders im HPC-Bereich die Verwendung von GPUs (Graphics Processing Unit, siehe Kapitel 5.5.9) für komplexe Rechenaufgaben ein sehr wichtiges Thema geworden. Diese Zusatzkarten bieten eine im Vergleich zu herkömmlichen CPUs größere Rechenleistung bei vergleichsweise niedrigem Energieverbrauch. Allerdings benötigen diese GPUs stark parallelisierte SIMD-Algorithmen (siehe Kapitel 5.5.3), was den Entwicklungsaufwand erhöht. Da das in dieser Arbeit entwickelte Optimierungsverfahren selbst sehr aufwendig ist und eventuelle Rechenzeiten den Optimierungsverlauf empfindlich stören könnten, soll die hier entwickelte Software durch aktuelle GPUs optional beschleunigt werden und auf Eignung überprüft werden.

Zielsetzung: Die aus den Anforderungen resultierende Zielsetzung wird folgend zusammengefasst und im Text genauer beschrieben.

1. Entwicklung eines CO-Kriging-Ersatzmodells
   (a) Objektorientierte Sprache
   (b) Ersatzmodellsoftware muss erweiterbar aufgebaut sein
   (c) Prozessinterne synchrone Parallelisierung
   (d) Prozessweite asynchrone Parallelisierung: Client/Server System
   (e) GPU-Unterstützung
2. Erweiterung des bisherigen Optimierungsverfahrens
   (a) Offene Schnittstellen zwischen Optimierer und Ersatzmodellen
   (b) Automatisierte Entscheidung zur Laufzeit über die verwendete Gütestufe
   (c) Erweiterung der Optimierer-Datenbank auf verschiedene Gütestufen
   (d) Erweiterung des Optimierers auf verschiedene Prozessketten für die jeweiligen Gütestufen

3. Testen des entwickelten Verfahrens
4. Nachweis der praktischen Anwendbarkeit an einer industriellen Turbomaschinenoptimierung

Der erste Punkt umfasst die Entwicklung und Integration eines CO-Kriging-Ersatzmodells für die AutoOpti-Umgebung. In der hier beschriebenen Umsetzung werden alle Kriging Verfahren (Ordinary-, Gradient-Enhanced-,CO-Kriging) und auch Klassifikatoren wie z.B. Supporting-Vector-Machines in einer Softwarestruktur untergebracht. Hierfür wird eine moderne Interface-basierte Programmierung verwendet. Die in der Literatur beschriebenen Programme beschränken sich in der Regel auf einzelne Ersatzmodelle welche oftmals in Frameworks wie Matlab (siehe [Toal, 2016, Z.-H. Han, R. Zimmermann, 2010, Han et al., 2012, Forrester et al., 2008] umgesetzt sind und bieten daher nicht die Benutzerfreundlichkeit, den Programmumfang und die Geschwindigkeit die für eine industrienahe Umgebung benötigt wird.

Weitergehend wird die Software über eine prozessinterne synchrone Parallelisierung durch Threads verfügen und moderne Befehlssatzerweiterungen wie SSE unterstützen. Eine weitere Möglichkeit der Beschleunigung ist die prozessweite Parallelisierung. Damit sollen die notwendigen Berechnungen auch über Rechnergrenzen hinaus (bspw. auf einem Rechencluster) parallelisierbar sein. Hierfür wurde im Rahmen dieser Arbeit ein asynchrones Client-/Server-System entwickelt, welches die Nutzung verschiedener Hardware-Architekturen über Rechnergrenzen hinweg erlaubt. Weiterhin soll die Verwendung von GPUs möglich sein, womit die notwendigen Berechnungen für das Ersatzmodell beschleunigt werden sollen. Vergleicht man diese Punkte mit den Softwarepaketen, welche in der Literatur beschrieben werden, so kann diese Art der Umsetzung nicht wiedergefunden werden. Das gilt insbesondere für ein asynchrones Client/Server System zur Parallelisierung und ebenfalls für den Mischbetrieb unterschiedlicher Architekturen.

Der zweite Punkt umfasst die Erweiterung der bestehenden Optimierungssoftware AutoOpti. Grundsätzlich müssen allgemeine Schnittstellen zwischen Optimierer und Ersatzmodellen definiert und umgesetzt werden. Eine automatisierte Strategie ist zu entwickeln, um während einer laufenden Optimierung zu bestimmen, welche Güte in der nächsten Evaluierung verwendet werden soll. Hierfür muss der aktuelle Stand der Optimierung und Ersatzmodelle automatisiert beobachtet und analysiert werden, um die entsprechende Entscheidung treffen zu können. Die Optimierungssoftware muss außerdem die Daten verschiedener Güte speichern, auswerten können und zudem die Verwaltung unterschiedlicher Prozessketten für die jeweiligen Gütestufen übernehmen. Die hier vorgestellten Ansätze einer Entscheidungsfunktion sind ebenfalls neuartig und unterscheiden sich von anderen Multifidelity-Ansätzen.

Der dritte Punkt umfasst erste analytische Tests, in denen einzelne Komponenten des neuen Verfahrens auf ihre Effizienz und Fehlerfreiheit getestet werden sollen.

Der vierte und letzte Punkt ist die Beschreibung einiger bereits durchgeführter Anwendungen des gesamten Verfahrens in umfangreichen industriehnahen Turbomaschinenoptimierungen. Es sollen insbesondere die Eignung von Multifidelity-Optimierungen für verschiedene Anwendungen ausgewertet werden.

2 Optimierungsstrategie für Turbomaschinen

In diesem Kapitel erfolgt eine Einführung in die Turbomaschinenoptimierung. Hierfür werden im ersten Teil allgemeine Grundlagen und Begrifflichkeiten erklärt, wobei ein Grundwissen über mathematische Optimierungen vorausgesetzt wird. Im zweiten Teil werden die Besonderheiten und die damit verbundenen Schwierigkeiten der Optimierung von Turbomaschinen erläutert. Darauf aufbauend wird im dritten Teil dieses Kapitels die bisher im DLR umgesetzte Optimierungsstrategie beschrieben.

2.1 Grundlagen Turbomaschinenoptimierung

Die Auslegung und Optimierung von Turbomaschinen ist eine multidisziplinäre Aufgabe, für die verschiedenen Anforderungen aus einer Vielzahl von physikalischen Disziplinen gegeneinander abgewogen und bewertet werden müssen. Diese Disziplinen sind für einen sicheren, effizienten und umweltverträglichen Betrieb notwendig. Zum besseren Verständnis der Komplexität dieser Aufgabenstellung werden nachfolgend die wesentlichen Fachdisziplinen aufgelistet und anhand von Beispielen erläutert.

2.1.1 Ingenieurwissenschaftliche Disziplinen in der Turbomaschinenoptimierung

Aerodynamik: Die Aerodynamik von Turbomaschinen kann sich auf verschiedene Baugruppen einer Turbomaschine beziehen, bspw. den Fan, Verdichter oder die Turbine. Ein häufiges Ziel ist z.B. das Auffinden einer Geometrie für eine oder mehrere Schaufelreihen, welche den Wirkungsgrad (Verhältnis von genutzter Energie zu zugeführter Energie) und gleichzeitig auch das Totaldruckverhältnis (Verhältnis vom Totaldruck am Austritt der Komponente zum Totaldruck am Eintritt der Komponente) maximiert. In der Regel werden zusätzlich Nebenbedingungen definiert, bspw. ein bestimmter Abströmwinkel und/oder ein einzuhaltender Betriebsbereich. Diese Ziele und Nebenbedingungen müssen für mehrere Betriebspunkte der Komponente definiert und eingehalten werden, wodurch sehr viele Nebenbedingungen und zu minimierende Funktionale (Zielfunktionen) entstehen können. Um diese Kennzahlen zu erhalten, wird aus einem vorher definierten Parametersatz eine Geometrie erzeugt und diese dann mittels eines CFD-Verfahrens simuliert. Dieser Prozess ist numerisch extrem aufwendig und benötigt in der Regel mehrere Stunden.

Bei komplexen Simulationen sind auch Rechenzeiten von mehreren Tagen möglich. Einen guten Überblick über diese Problematik bieten die folgenden Literaturstellen [Anderson and Wendt, 1995, Bräunling, 2004, Schlichting and Gersten, 2006]

Strukturmechanik: Die Strukturmechanik von Turbomaschinen beschäftigt sich mit der Bestimmung der mechanischen Belastung der verwendeten Bauteile. Hierzu gehören die Berechnung der auftretenden Spannungen in den Schaufeln und weiteren Bauteilen (z.B. der Scheibe) durch die Fliehkraft- und Druckbelastung. Zudem müssen dynamische Problemstellungen beachtet werden. Beispielsweise müssen im gesamten Betriebsbereich der Turbomaschine resonante Schaufelschwingungen vermieden werden. Um dies zu realisieren werden für verschiedene Drehzahlen die Eigenfrequenzen und Eigenformen der Schaufelreihen mit modernen CSM-Verfahren berechnet. Im Falle einer Turbinenauslegung sind durch die hohen Temperaturen thermomechanische Simulationen notwendig, welche den Aufwand nochmals erhöhen. Weitere Informationen zu diesem Thema können in [Bräunling, 2004, Zienkiewicz and Taylor, 2005, Gere and Weaver, 1965, Przemieniecki, 1985] gefunden werden.

Aeroelastik: Die Aeroelastik kann als eine Kopplung zwischen der Aerodynamik und Strukturmechanik angesehen werden. Ein sehr wichtiges Phänomen ist das Schaufelflattern. Hierbei handelt es sich um eine aeroelastische Instabilität, die durch die Wechselwirkung einer Schaufelschwingung mit den dadurch hervorgerufenen, instationären Druckverteilungen auf den Schaufeln verursacht wird. Diese instationären aerodynamischen Druckverteilungen können die Schwingung entweder dämpfen oder anfachen. Im letzten Fall treten selbsterregte Schwingungen auf (Eigenschwingungen) und die Schaufel wird instabil. Die hierfür notwendigen Berechnungen sind allerdings mit hohem numerischem Aufwand verbunden, sodass diese erst nach einer Optimierung für ausgewählte Geometrien durchgeführt werden. Einen umfassenden Überblick dazu bieten: [Hodges and Pierce, 2011, Bisplinghoff et al., 1957, Sechler, 1952]

Aeroakustik: Die Aeroakustik beschäftigt sich mit der Entstehung und Ausbreitung aerodynamisch erzeugter Geräusche. Da es sich bei der Aeroakustik um ein rein instationäres Phänomen handelt, ist die Berechnung akustischer Kriterien für eine Optimierung meist zu aufwendig. Daher wird oftmals auf vereinfachte Modelle zurückgegriffen oder direkt Einfluss auf stationäre Strömungsphänomene genommen, von denen man eine starke Lärmentstehung vermutet.

Die hier aufgelisteten Fachdisziplinen stellen nur die wesentlichen Disziplinen dar, die in einer Turbomaschinenoptimierung auftreten können. Dennoch wird ersichtlich, dass durch diese Vielzahl sehr hohe Anforderungen an das Optimierungsverfahren sowie den Anwender gestellt werden. [Lighthill, 1952, Lighthill, 1954]

2.1.2 Begriffserläuterungen zu Optimierungen

Im Folgenden werden wichtige Begrifflichkeiten zu Optimierungsverfahren eingeführt und anhand von Beispielen aus dem Bereich der Turbomaschinenauslegung erläu-

tert. Für weitere Informationen sei der Leser auf folgende Literaturstellen verwiesen: [Miettinen, 2012, Ehrgott, 2005, Deb, 2014, Marler and Arora, 2004]

Zielfunktion oder Fitness: Grundsätzlich ist es das Ziel einer Optimierung ein oder mehrere Zielfunktionen zu minimieren oder zu maximieren, wobei im Folgenden grundsätzlich von einer Minimierung ausgegangen wird. Typische Zielfunktionen für die Turbomaschine sind die Erhöhung des Wirkungsgrads, die Erweiterung des Betriebsbereichs, die Verminderung von Lärm etc.

Nebenbedingung: Nebenbedingungen werden oftmals als Gültigkeitsintervalle festgelegt, in denen die berechneten Größen liegen sollen. Beispiele für Nebenbedingungen sind die maximale mechanische Spannung einer durch Fliehkraft belasteten Schaufel oder die Einhaltung eines gewissen Massenstrombereichs im aerodynamischen Designpunkt.

Region Of Interest (ROI): Ein für eine Zielfunktion oder Nebenbedingung festgelegtes Gültigkeitsintervall.

Freie Variablen: Um die Zielfunktion(en) zu optimieren, kann der Optimierungsalgorithmus eine bestimmte Anzahl von freien Variablen, innerhalb eines festgelegten Intervalls verändern. Bei einem Verdichter sind das typischerweise Parameter, welche die Schaufelgeometrien beschreiben. Beispielsweise die Länge oder Dicke einer Schaufel für eine bestimmte radiale Höhe. Nach jetzigem Stand sind Optimierungen in der Größenordnung von bis zu 50-100 freien Variablen üblich. In sehr komplexen Optimierungen werden bis zu 250 freie Variablen verwendet, siehe [Aulich and Siller, 2011].

Flowparameter: Der Begriff „Flowparameter“ gehört grundsätzlich nicht zu den allgemeinen Optimierungsbegriffen, sondern wird speziell im DLR-Institut für Antriebstechnik verwendet. Da der Begriff dennoch eine gewisse Allgemeingültigkeit bekommen hat, wird er in diese Auflistung aufgenommen.

Wird mit einem Satz freier Parameter beispielsweise eine Verdichtergeometrie erzeugt und diese im Anschluss aeromechanisch bewertet, so entstehen durch das nachfolgende Post-Processing eine Vielzahl von Ergebnisgrößen. Oftmals werden hunderte bis tausende an Ergebnisgrößen pro Simulation erzeugt. Als Flowparameter bezeichnet man alle Ergebnisgrößen, welche von den genutzten numerischen Verfahren erzeugt und im Optimierungsprozess abgespeichert werden. Aus diesen Flowparametern werden dann die Zielfunktionen und Nebenbedingungen berechnet. Erfahrungsgemäß werden Optimierungsziele und auch Nebenbedingungen während einer laufenden Turbomaschinenoptimierung mehrfach geändert und angepasst. Aus diesem Grund ist es wichtig, möglichst alle wesentlichen Ergebnisgrößen zu speichern. Dies stellt auch die Möglichkeit eines Neustarts oder Wiederaufnehmens einer Optimierung sicher.

Individuum (auch „Member“): Ein Tupel freier Parameter $(x_1, ..., x_k)$ mit zugeordneten Flowparametern $(f_1, ..., f_z)$ und Zielfunktionswerten $(y_1, ..., y_z)$ und den Nebenbedingungen $(w_1, ..., w_c)$ eines Optimierungsproblems mit k-dimensionalen Parameterraum, z-dimensionalen Zielfunktionsraum und c-dimensionalen Nebenbedingungsraum bezeichnet man als Member oder Individuum.

Optimierungsproblem: Ein Optimierungsproblem lässt sich mit den vorhergehenden Bezeichnungen wie folgt definieren:

$$\min_{x_i \in [a_i, b_i] \forall i \in \{1,...,k\}} \left\{ \left(y_1 \left(\vec{x} \right), ..., y_z \left(\vec{x} \right) \right) \middle| \, w_j \left(\vec{x} \right) \in [c_j; d_j] \, \forall j \in \{1, ..., c\} \right\} \tag{2.1}$$

Aus der Formulierung wird ersichtlich, dass dieser Ansatz die Möglichkeit bietet, mehrere Ziele gleichzeitig zu verfolgen. Da die Ziele oftmals nicht gemeinsam optimiert werden können, weil diese z.B. widersprüchlich sind, fasst man sie oftmals über Gewichtungsfaktoren als gemeinsame Zielfunktion zusammen. Diese Faktoren optimal zu bestimmen, ist im Allgemeinen sehr schwierig. Um dieses Problem zu umgehen wird in der Regel ein Paretorang-Gütekriterium verwendet. Damit können mehrere Ziele gleichzeitig optimiert werden, ohne diese gegeneinander zu gewichten und so eine Menge an optimalen Lösungen zu finden.

Paretorang und Paretofront: Die Begriffe Paretorang und Paretofront sind nach dem Ökonomen und Soziologen Vilfredo Pareto (1848–1923) benannt und werden im Folgenden kurz definiert (vgl. [Pareto, 1964, Ehrgott, 2005, Miettinen, 2012, Voß and Nicke, 2008]). Man sagt, dass ein bewertetes Individuum M_1 ein Individuum M_2 dominiert $M_1 \prec M_2$, falls bzgl. aller Zielfunktionen gilt:

$$M_1 \prec M_2 \Longleftrightarrow y_i \left(M_1 \right) \leq y_i \left(M_2 \right) \forall i \in \{1, ..., z\} \tag{2.2}$$

$$\wedge \, \exists j \in \{1, ..., z\} : y_j \left(M_1 \right) < y_j \left(M_2 \right) \tag{2.3}$$

Mit dem Begriff der Dominanz lässt sich der Paretorang P eines Individuums definieren:

$$P \left(M \right) \coloneqq \# \left\{ \hat{M} \in M_{all} \middle| \, \hat{M} \prec M \right\} + 1 \tag{2.4}$$

Der Operator $\#$ gibt die Anzahl einer endlichen Menge an und M_{all} die Menge aller bereits bewerteten Individuen. Die Paretofront PF bezeichnet die Teilmenge aus der Menge aller Individuen M_{all} mit Paretorang $P = 1$.

$$PF \coloneqq \left\{ \hat{M} \in M_{all} \middle| \, P \left(\hat{M} \right) = 1 \right\}$$

Behandlung von Nebenbedingungen

In AutoOpti können Gültigkeitsintervalle für alle gespeicherten physikalischen Größen und Zielfunktionen angegeben werden. Liegt ein berechnetes Individuum nicht innerhalb dieses Gültigkeitsbereich, so wird der Abstand $d \left(M \right)$ des Individuums M zum

Gültigkeitsbereich als zusätzliches Gütekriterium herangezogen. Die Definition der Dominanz (siehe Gleichung 2.2) muss daher erweitert werden zu:

$$M_1 \prec M_2 \Longleftrightarrow d\left(M_1\right) < d\left(M_2\right) \vee \left[d\left(M_1\right) \leq d\left(M_2\right)\right. \tag{2.5}$$

$$\wedge \left(y_i\left(M_1\right) \leq y_i\left(M_2\right) \forall i \in \{1, ..., z\}\right.$$

$$\left.\wedge \exists j \in \{1, ..., z\} : y_j\left(M_1\right) < y_j\left(M_2\right)\right)\Big] \tag{2.6}$$

Ein Individuum M_1 dominiert ein Individuum M_2, wenn der Abstand zum Gültigkeitsbereich verbessert wird oder der Abstand zum Gültigkeitsbereich gleich bleibt und gleichzeitig die ursprüngliche Definition der Dominanz (siehe Gleichung 2.2) erfüllt wird. Durch Definition 2.5 hat bei der Bestimmung des Paretorangs die Erfüllung der Nebenbedingungen Vorrang vor den Zielfunktionswerten.

2.2 Besondere Herausforderungen bei Turbomaschinenoptimierungen

Die Auslegung und damit auch Optimierung von Turbomaschinen stellt eine besondere Herausforderung dar, welche speziell angepasste Optimierungsverfahren erfordert. In diesem Abschnitt sollen diese Besonderheiten kurz aufgelistet und erläutert werden.

Mehrere Disziplinen: Die Anzahl und Komplexität der benötigten physikalischen Disziplinen ist sehr hoch. Jede Disziplin benötigt in der Regel eine eigene Software, welche aus unterschiedlichen Quellen stammt. Um diese in einer Prozesskette verbinden zu können, müssen oft Anpassungen an der Software vorgenommen und Schnittstellen definiert werden. Dies erschwert die notwendige Fehlerbehandlung einer Prozesskette erheblich und verlangt ein großes Maß an interdiziplinärem Fachwissen.

Mehrere Zielfunktionen und Nebenbedingungen: Meist gibt es mehrere Zielfunktionen welche wiederum aus mehreren physikalischen Größen zusammengesetzt werden. Ein typisches Beispiel ist der Wirkungsgrad $\eta_{is} = f\left(P_{t,ein}, P_{t,aus}, T_{t,ein}, T_{t,aus}\right)$, insbesondere wenn dieser über viele Betriebspunkte gemittelt wird. Hinzu kommt, dass die unterschiedlichen Zielfunktionen meist negativ korreliert sind und eine große Anzahl relevanter Nebenbedingungen existiert. Beim Start einer Optimierung werden diese Nebenbedingungen häufig von keinem Individuum vollständig erfüllt und oftmals ist es auch nicht sicher, ob die Nebenbedingungen überhaupt alle erfüllbar sind.

Diese Punkte führen dazu, dass eine Optimierung anfangs nur sehr grob definiert werden kann und die Ziele und Nebenbedingungen während der laufenden Optimierung umformuliert oder angepasst werden müssen. Das Optimierungsverfahren muss also auch Änderungen von Zielen und Nebenbedingungen während der laufenden Optimierung zulassen.

Gradienteninformation: Sind die partiellen Ableitungen der Zielfunktion $\frac{\partial y_i}{\partial x_j}; i \in \{1, ..., z\}$ oder der Nebenbedingungen $\frac{\partial w_l}{\partial x_j}; l \in \{1, ..., c\}$ nach dem Parameter $x_j; j \in$

$\{1, ..., k\}$ mit überschaubaren Rechenaufwand verfügbar, können diese die Optimierung stark beschleunigen. Allerdings sind oft nicht alle Ableitungen verfügbar, da das Funktional nicht differenzierbar ist oder die Programme diese Information nicht zur Verfügung stellen. *Das alles führt dazu, dass das verwendete Optimierungsverfahren auch unvollständige Gradienteninformationen verarbeiten können muss.*

Konvergenz der verwendeten Programme: Während einer Optimierung werden zahlreiche Geometrien erzeugt. Diese Geometrien werden anschließend in der Prozesskette (siehe Abbildung 2.1) bewertet. Die Prozessketten enthalten eine Vielzahl von Rechenschritten, wie z.B. die Netzerzeugung und die darauf basierenden aeromechanischen Simulationen. Dabei können unsinnige Geometrien oder schlechte Rechennetze entstehen. Teils treten auch numerische Probleme oder Programmabbrüche auf, die der Anwender nicht direkt analysieren und nachvollziehen kann. In vielen Fällen benötigen diese Rechnungen sehr viel Zeit und liefern letztlich keine oder nur partielle Informationen. Dieses Verhalten darf nicht zum Abbruch der Optimierung führen.

Lange Prozesskettenzeiten: Dieser Punkt ist im Bereich der Turbomaschinenoptimierung sehr entscheidend, denn besonders in der Aerodynamik sind die Simulationsverfahren sehr aufwendig.

Trotz der steigenden Rechnerkapazität ist nicht zu erwarten, dass die Prozesskettenlaufzeiten abnehmen. Dies liegt daran, dass für die Berechnung von Strömungen vereinfachte Modelle verwendet werden und die steigenden Rechenkapazitäten eher für aufwendigere Verfahren eingesetzt werden, um komplexere Strömungsvorgänge simulieren zu können.

Ein Optimierungsverfahren sollte also mit so wenig Berechnungen wie möglich zu einem gewünschten Ergebnis führen.

Hochdimensionaler Suchraum: Die Problematik von hochdimensionalen Suchräumen wird auch als „Curse of dimensionality“ bezeichnet.

Dieser bekannte Begriff entspringt einer Arbeit von Richard Bellman [Bellman, 1972]. Er beschreibt eine Problematik, welche in vielen Bereichen zu finden ist, wie z.B. bei der Verteilung von Stichproben (Sampling) oder der Optimierung. Geht man bspw. von einem Raum aus, welcher dicht mit vorhandenen Stichproben besetzt ist, so kann dieser nach dem Hinzufügen von weiteren Dimensionen dünn besetzt sein und nur noch unzureichend beschrieben werden. [Friedman et al., 2001, Trunk, 1979]

Die Dimension des Suchraums ist im Bereich der Turbomaschinenoptimierung sehr hoch. Typische Parameterzahlen findet man bspw. in der Arbeit von Voss mit 50 Parametern [Voß et al., 2014] oder bis zu 230 Parametern in der Arbeit von Siller [Siller et al., 2009]. Berechnet man die Anzahl der Ecken eines solchen Hyperwürfels, so bekommt man für eine Dimension von 50 eine Anzahl von ca. 10^{15} Ecken, bei 230 Parametern liegt die Anzahl bei ca. 10^{69}. Einen Suchraum dieser Größe ausreichend abzutasten, ist bei den vorhandenen Prozesskettenzeiten unmöglich. Der Optimierungsprozess muss speziell für eine so große Parameteranzahl ausgelegt sein, um

ausgehend von einer Initialisierung möglichst schnell Verbesserungen zu finden.

Für weitere Informationen bieten folgende Literaturstellen einen umfassenden Vergleich verschiedenster Optimierungsverfahren im Bereich der Turbomaschinen: [Müller-Töws, 2000, Shahpar, 2000, Bellman, 1972, Ehrgott, 2005, Giannakoglou and Karakasis, 2006, Miettinen, 2012, Spiegel, 2000].

2.3 Automatisierte Optimierung im DLR

Auf die Optimierungssoftware AutoOpti wurde bereits Bezug genommen. Daher wird in diesem Abschnitt der bisherige Stand von AutoOpti dokumentiert und erläutert. Dies ist notwendig, da die hier entwickelten Verfahren innerhalb dieser Optimierungsumgebung implementiert wurden.

Der erste Teil erläutert den grundlegenden Ablauf einer Optimierung mit AutoOpti. Im darauffolgenden zweiten Teil wird dann die Verwendung von Ersatzmodellen innerhalb von AutoOpti beschrieben.

2.3.1 Grundlegender Optimierungsprozess

Eine automatisierte Optimierung, wie sie im DLR ausgeführt wird, basiert auf der Evolutionsstrategie [Rechenberg, 1973]. Diese ist ein stochastisches Optimierungsverfahren, welches kaum Voraussetzungen (z.B. Differenzierbarkeit oder Stetigkeit) an die Zielfunktion stellt. Die Abstiegsrichtung wird durch zufällige Tastschritte in der näheren Umgebung bekannter Individuen bestimmt.

Abbildung 2.1 zeigt den zugrundeliegenden Optimierungsablauf, bei dem es sich im Wesentlichen um eine Endlosschleife handelt. Innerhalb dieses Kreislaufes gibt es verschiedene Arten von Prozessen, welche über eine asynchrone Kommunikation verfügen.

Dies bedeutet, dass die Berechnungen in der Prozesskette und die Datenbankverwaltung sowie die Erzeugung neuer Individuen unabhängig voneinander agieren. Um solch ein asynchrones Verfahren umzusetzen, bedient sich AutoOpti eines Master-/Slave-Ansatzes. Hierfür sind grundlegend zwei verschiedene Arten von Prozessen vorgesehen:

- Der Master-Prozess ist für die Verwaltung und Steuerung der Optimierung zuständig.
- Die Slave-Prozesse dienen zur Berechnung der Prozesskette und werden vom Master-Prozess gestartet und gesteuert. Meist wird innerhalb einer Prozesskette eine Geometrie erzeugt und diese dann mit den numerischen Simulationsverfahren bewertet. Damit übernehmen die Slave-Prozesse den numerisch aufwendigen Teil der Optimierung.

Der in Abbildung 2.1 dargestellte Ablauf einer AutoOpti-Optimierung soll im Folgenden genauer erläutert werden.

1. Initialisierungsphase: Der hier beschriebe initiale Schritt ist in Abbildung 2.1 nicht dargestellt, da diese nur den zyklischen Teil des Optimierungsprozesses beschreibt. Während dieser Initialisierungsphase spielt die Minimierung der Zielfunktionen noch keine Rolle. Ziel ist es vielmehr, ein möglichst breites Spektrum an konvergenten Individuen zu erzeugen und damit einen möglichst großen Parameterbereich abzudecken. Hierfür bietet AutoOpti diverse Verfahren an:

- Zufällige Variation der freien Variablen
- Latin-Hypercube-Sampling (siehe McKay et al. [McKay et al., 1979])
- Mutation eines initialen Individuums

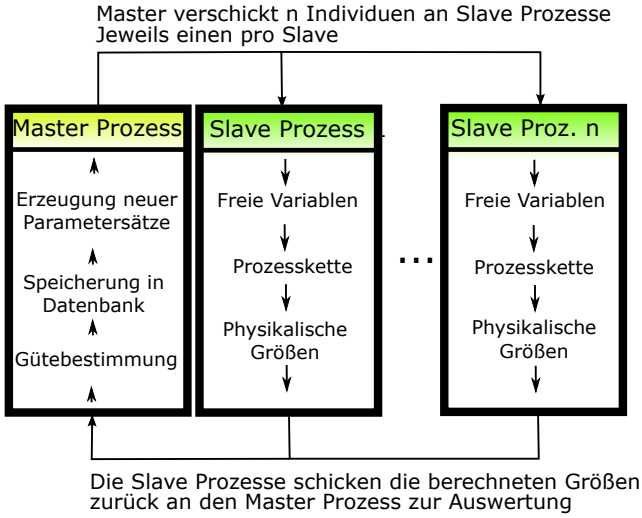


Abbildung 2.1: Prozesskette des genetischen Algorithmus in AutoOpti

Das erste Verfahren erzeugt zufällige, gleichverteilte Parametersätze, wobei die Parameter jeweils einzeln und vollkommen unabhängig voneinander bestimmt werden. Bei einfachen Optimierungen ist diese Art der initialien Erzeugung oftmals ausreichend.

Das Latin-Hypercube-Verfahren versucht den gegebenen Parameterraum möglichst raumfüllend abzutasten und ist auch in hochdimensionalen Räumen anwendbar.

Beide Verfahren erscheinen nur dann sinnvoll, wenn zumindest ein gewisser Prozentsatz das erzeugte Individuum die Prozesskette erfolgreich durchläuft. Bei den hier behandelten Turbomaschinenoptimierungen ist dies oft aber nicht der Fall.

Eine mögliche Lösung bietet das dritte Verfahren. Dafür muss ein initiales Individuum bekannt sein, welcher die Prozesskette erfolgreich durchläuft. Dieser initiale Parametersatz wird dann immer wieder mutiert und evaluiert. Da sich diese Art der Suche nur lokal um den initialen Parametersatz bewegt, kann meist eine ausreichende Quote von erfolgreichen Prozesskettendurchläufen erreicht werden.

2. Erzeugung neuer Parametersätze: Der eigentliche Optimierungsprozess startet nach der Initialisierungsphase mit der bereits erzeugten initialen Datenbank. Der Master-Prozess erzeugt einen oder mehrere vielversprechende Parametersätze auf Basis der bereits vorhandenen Individuen. Für die Erzeugung dieser neuen Parametersätze gibt es zahlreiche Verfahren, welche die Geschwindigkeit und den Konvergenzverlauf der Optimierung maßgeblich beeinflussen. AutoOpti nutzt hauptsächlich eine Ersatzmodel-beschleunigte Evolutionsstrategie zur Erzeugung neuer Individuen. Diese Strategie zur Erzeugung vielversprechender Parametersätze hat sich in der Vergangenheit als sehr effizient und flexibel herausgestellt und wird in Kapitel 2.3.2 genauer erläutert.

3. Verschicken neuer Parametersätze an die Slave-Prozesse: Nachdem der Master-Prozess einen vielversprechenden Parametersatz generiert hat, muss dieser

zur Berechnung an einen freien Slave-Prozess verschickt werden. Die Kommunikation findet bei AutoOpti über ein gemeinsames Dateisystem statt.

4. Slave-Prozesse durchlaufen die Prozesskette: Nachdem der Master-Prozess die neuen Parametersätze an die Slave-Prozesse verschickt hat, können diese die Prozesskette durchlaufen. Bei erfolgreicher Ausführung der Prozesskette müssen alle relevanten Ergebnisse in einer Datei bereit gestellt werden. Die gesammelten Ergebnisse werden vom Slave-Prozess eingelesen und als Flowparameter abgespeichert. Bei nicht erfolgreichen Prozessen werden diese in der Datenbank markiert und die bis dahin erzeugten Ergebnisse abgespeichert.

5. Slave-Prozesse senden die Ergebnisse an den Master-Prozess: Die Slave-Prozesse schicken die berechneten Flowparameter zurück an den Master-Prozess. Die Kommunikation findet auch hier über ein gemeinsames Dateisystem statt.

6. Auswertung der Ergebnisse durch den Master-Prozess: Aus den vom Slave bereitgestellten Flowparametern berechnet der Master-Prozess nun die Zielfunktionen, Nebenbedingungen und daraus auch das Gütekriterium. Ein möglicher Gütewert wäre hier der Paretorang (siehe Kapitel 2.1.2). Nachdem diese Werte bestimmt worden sind, trägt der Master-Prozess das Individuum gemäß seiner Güte in die Datenbank ein. Danach springt der Master-Prozess wieder zu Punkt 2.

2.3.2 Ersatzmodellbeschleunigung

Im vorhergehenden Abschnitt wurde der grundlegende Ablauf einer Optimierung mit AutoOpti beschrieben. Der zentrale Punkt der Erzeugung neuer Parametersätze wurde allerdings nur kurz erwähnt und soll im Folgenden genauer erklärt werden. Eine weit verbreitete Möglichkeit der Erzeugung neuer Individuen ist die Verwendung einer Evolutionsstrategie. Wie bereits beschrieben, verwendet die Evolutionsstrategie zufallsbasierte Tastschritte in der unmittelbaren Nähe bekannter Individuen. Typische Operatoren um die zufallsbasierte Änderung von Parametern vorzunehmen, sind z.B. Mutation oder Kreuzung von Individuen. Dies führt jedoch dazu, dass die erreichten Verbesserungen eher klein sind. Um größere Fortschritte zu erreichen, sind meist sehr viele Tastschritte und damit Funktionsauswertungen notwendig. In der Kombination mit den großen Parameterräumen und langen Prozesskettenzeiten wäre eine solche Optimierung nicht mehr rentabel. Um den Gesamtprozess zu beschleunigen, versucht man die Individuenerzeugung im Master-Prozess (siehe Abbildung 2.1) zu verbessern. Es sollen zielgerichtet neue Individuen erzeugt werden, die eine möglichst große Verbesserung der Zielfunktionen versprechen und zusätzlich alle Nebenbedingungen erfüllen.

Ein weit verbreitetes Mittel hierfür sind Ersatzmodelle oder auch Metamodelle. Dabei handelt es sich um relativ schnell auszuwertende mathematische Modelle, welche vorhandene Daten approximieren oder interpolieren. Die Ersatzmodelle werden innerhalb einer Optimierung dafür verwendet, Vorhersagen der Zielfunktionen und Nebenbedingungen zu treffen und damit besonders aussichtsreiche Parametersätze zu finden. Die

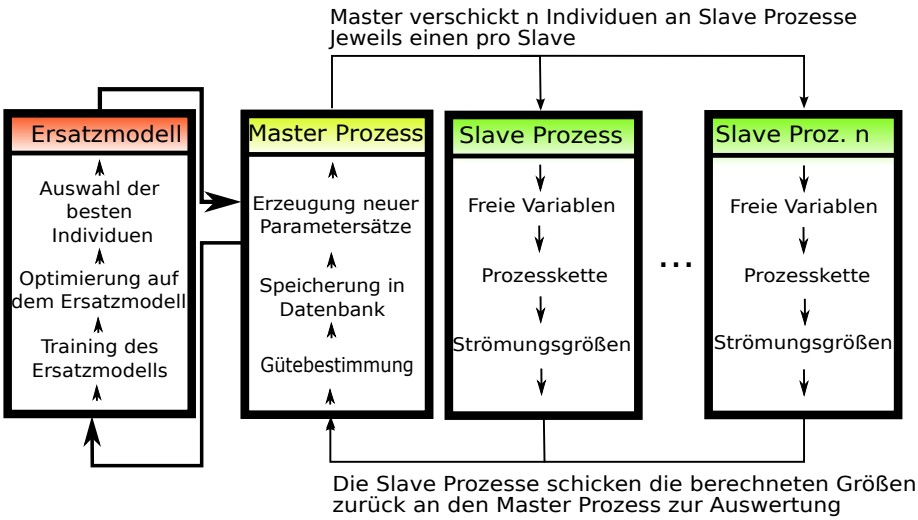


Abbildung 2.2: Nutzung von Ersatzmodellen im Optimierungsprozess

Erzeugung kann dann z.B. durch eine zusätzliche Optimierung auf den Ersatzmodellen erfolgen. Das aussichtsreichste Individuum wird anschließend mit der echten, zeitaufwendigen Prozesskette nachgerechnet und danach in die Optimierungsdatenbank eingetragen und den Ersatzmodellen als zusätzliche Stützstelle hinzugefügt. Dieses Verfahren bietet die Möglichkeit, den Optimierungsfortschritt stark zu beschleunigen. Wie stark, hängt von der Güte und Leistungsfähigkeit des verwendeten Modells ab. Mithilfe von Abbildung 2.2 soll der in AutoOpti verwendete Prozess zur Nutzung von Ersatzmodellen genauer beschrieben werden.

1. Training der Ersatzmodelle: Die in AutoOpti verwendeten Ersatzmodelle sind bayesisch trainierte neuronale Netzwerke [Mackkay, 1991] und verschiedene Kriging-Verfahren (siehe Kapitel 4). Die Kriging-Verfahren haben sich als Standard in AutoOpti etabliert, da diese robust zuverlässige Vorhersagen liefern.

Jedes dieser Verfahren benötigt ein Training, welches oftmals als Lernverfahren bezeichnet wird. Es dient zur Bestimmung von ersatzmodellspezifischen Parametern, wie z.B. Korrelationslängen, Gewichte, Skalierungsfaktoren usw.. Die Ersatzmodellparameter werden mithilfe der vorhandenen Daten eingestellt, um möglichst gute Vorhersagen zu erlangen. Meistens sind diese Trainingsverfahren sehr aufwendig und benötigen je nach Typ des Ersatzmodells und der Anzahl an Stützstellen viel Zeit. Nach dem Training können die Ersatzmodelle für Vorhersagen der trainierten Funktion verwendet werden, wobei die benötigten Vorhersagezeiten gegenüber der Trainings- oder Prozesskettenzeiten zu vernachlässigen sind (siehe Kapitel 4).

In AutoOpti werden für jede Zielfunktion und Nebenbedingung separate Ersatzmodelle trainiert. Bei aus mehreren „Flowparametern“ zusammengesetzten Zielfunktionen wird für jeden in der Zielfunktion verwendeten Flowparameter ebenfalls ein Ersatzmodell trainiert und aus den einzelnen Vorhersagen der Zielfunktionswert berechnet.

Das Training kann entweder synchron durch den Master-Prozess geschehen oder asynchron durch einen eigenen Prozess. Bei der synchronen Variante trainiert der Master-Prozess die Ersatzmodelle und ist in dieser Zeit blockiert. Wird in dieser Zeit

ein Slave-Prozess frei, so muss dieser warten bis das Training beendet ist.

Bei der asynchronen Variante wird das Training in einem externen Prozess gestartet und der Master-Prozess verwendet für die Vorhersagen dann die jeweils vorliegenden Modelle.

2. Optimierung auf den Ersatzmodellen: Ist das Training abgeschlossen, wird eine eigene Optimierung auf den Ersatzmodellen gestartet, wobei die Ersatzmodelle die Prozesskette ersetzen. Da die Funktionsauswertungen auf den Ersatzmodellen um Größenordnungen schneller sind als die Prozesskette selbst, nimmt dieser Schritt wenig Zeit in Anspruch. Ziel der Optimierung auf den Ersatzmodellen ist es, einen vielversprechenden Parametersatz zu finden, welcher die Zielfunktionen möglichst stark verbessert und zudem eine hohe Wahrscheinlichkeit besitzt, alle Nebenbedingungen einzuhalten.

3. Auswahl der vielversprechendsten Individuen: Meistens werden in Turbomaschinenoptimierungen mehrere Zielfunktionen und Nebenbedingungen verwendet. Dies führt dazu, dass bei der Optimierung auf den Ersatzmodellen mehrere vielversprechende Parametersätze generiert werden. Bei der Verwendung eines Gütekriteriums wie dem Paretorang würden viele Parametersätze mit dem Paretorang Eins erzeugt werden und könnten untereinander nicht weiter gewichtet werden. Für die Optimierung auf dem Ersatzmodell ist also ein Gütekriterium zu bevorzugen, welches mehrere Zielfunktionen und Nebenbedingungen vereint und auch den zu erwartenden Optimierungsfortschritt zu einer bestehenden Paretofront genauer quantifizieren kann. In AutoOpti findet dafür das „Expected Volume Gain“ als Gütekriterium Anwendung. Dieses Kriterium berechnet den zu erwartenden Volumenzugewinn von einem vorhergesagten Individuum zu einer bereits vorhandenen Paretofront unter Berücksichtung von bereits rechnenden Individuen.

Ein großer Vorteil bei diesem Verfahren ist die Berücksichtigung der Unsicherheiten der Vorhersage. Bei statistischen Ersatzmodellen werden Erwartungswerte und auch Varianzen für die Zielgrößen vorhergesagt. Bleiben die Varianzen hierbei unberücksichtigt, kann dies zu einer ungeeigneten Individuen-Auswahl führen (vgl. [Forrester et al., 2008]). Zum Beispiel neigen die in Kapitel 4 vorgestellten Kriging-Verfahren dazu, in Bereichen des Raums mit niedriger Stützstellendichte einen Erwartungswert mit einer großen Varianz vorherzusagen. Da die globalen Erwartungswerte in der Regel nicht das Optimum darstellen, würde ohne Berücksichtigung der Varianz kein Verbesserungspotential erkannt werden. Da die Entwicklung dieses EVG-Verfahrens nicht Teil dieser Arbeit ist und eine adäquate Beschreibung zu viel Platz in Anspruch nehmen würde, sei der interessierte Leser hierfür auf folgende Literaturstellen verwiesen: [Aulich and Siller, 2011, Jones et al., 1998, Jones, 2001, Keane, 2006, Emmerich, 2005, Emmerich and Klinkenberg, 2008]

3 Multifidelity Optimierungsstrategie Turbomaschine

Das folgende Kapitel beschreibt die Erweiterung der in Kapitel 2 beschriebenen Strategie basierend auf der Nutzung von Prozessketten verschiedener Gütestufen. Diese Art der Optimierungsstrategie wird Multifidelity-Optimierung genannt.

Im ersten Teil dieses Kapitels werden allgemeine Grundlagen sowie die Vorteile und Grenzen einer solchen Strategie ausgearbeitet. Darauf aufbauend wird im zweiten Teil dieses Kapitels die entwickelte Umsetzung und Neuentwicklungen dieses Verfahrens beschrieben. Die Entwicklung der hier beschriebenen Entscheidungsfunktion stellt neben den in Kapitel 5 gezeigten Arbeiten ein zentrales Thema dieser Arbeit dar.

3.1 Gütestufen in der Turbomaschinenauslegung

Um die Idee hinter einem Multifidelity-Verfahren zu verstehen ist es sinnvoll, einen exemplarischen Ablauf einer manuellen, aerodynamischen Turbomaschinenauslegung zu zeigen (vgl. [Domercq, 2006]). Denn erfahrungsgemäß werden innerhalb von Turbomaschinenauslegungen grundsätzlich mehrere Gütestufen verwendet. Diese gewinnbringend in einem Optimierungsverfahren zu nutzen, erscheint daher sinnvoll.

Die Vorgehensweise bei einer solchen Auslegung ist es, die Dimensionalität und die Komplexität der Problemstellung stetig zu erhöhen. Zu Anfang sind viele Randbedingungen durch die geplante Anwendung der Maschine festgelegt. Bei einem Verdichter können diese Anforderungen bestimmte Massenströme und geforderte Druckverhältnisse sein. Damit können dann die ungefähren geometrischen Maße und auch die benötigte Stufenanzahl geschätzt werden. Aufbauend auf diesen Schätzungen wird ein rotationssymmetrischer Strömungskanal bestimmt. Hierfür finden 2D-Throughflow-Verfahren Anwendung, welche ein Meridianströmungsfeld auf der S2-Ebene (siehe [Wu, 1952]) berechnen. Ein sehr großer Vorteil dieser Verfahren ist es, dass noch keine genaue Kenntnis über die Geometrie der Verdichterschaufeln benötigt wird (vgl. [Willburger, 2011]). Diese werden nur über aerodynamische Kenngrößen beschrieben. Typisch sind hier die Umlenkung und Totaldruckverluste einer Schaufel. Es existieren zahlreiche Umsetzungen solcher Throughflow-Verfahren, bspw. [Mönig et al., 2000, Schmitz et al., 2011, Schnoes and Nicke, 2015], welche im Kern auf der Arbeit von Howard und Gallimore [Howard and Gallimore, 1992] basieren.

Nachdem der Strömungskanal und auch die aerodynamischen Kenngrößen für die Schaufelreihen bestimmt worden sind, versucht man Schaufelgeometrien zu finden,

welche die vorher bestimmten Kenngrößen erfüllen. Hierfür werden entweder S1-Schnittverfahren wie z.B. MISES [Drela and Youngren, 2008] angewendet oder man greift auf bereits bestehende Profildatenbanken zurück (siehe [Käters et al., 2000, Köller, 1999]). Ist dieser Schritt abgeschlossen, kann eine 3D-Blattgeometrie mittels der voroptimierten Profilschnitte aufgebaut und durch ein 3D-Navier-Stokes-Simulationsverfahren aerodynamisch bewertet werden. Damit können erstmals komplexere, dreidimensionale Strömungsphänomene aufgelöst werden.

Aufgrund dieses stufenweisen Aufbaus einer solchen Auslegung, existieren meist eine Prozesskette niedriger und eine Prozesskette hoher Güte. Die Prozesskette niedriger Güte ist zwar sehr schnell berechnet, jedoch ist diese normalerweise auch mit einer größeren Ungenauigkeit behaftet. Die Prozesskette hoher Güte hingegen ist deutlich aufwendiger, dafür aber genauer. In der folgenden Aufzählung sollen einige Beispiele für verschiedene Gütestufen im Bereich der Turbomaschinenauslegung widergegeben werden:

- Gütestufen innerhalb eines Simulationsverfahrens
  - Auflösung des Rechennetzes (siehe [Forrester et al., 2007])
  - Anzahl an Iterationen innerhalb einer Strömungssimulation (siehe [Forrester et al., 2006])
  - Verschiedene Turbulenzmodelle
- Gütestufen über verschiedene Simulationsverfahren
  - Zeitlich gemittelt oder zeitlich aufgelöste Simulationen
  - Dimensionalität: 1D-, 2D- oder 3D-Strömungslöser

3.1.1 Beispiel für verschiedene Gütestufen innerhalb eines Simulationsverfahrens

Um einen Eindruck von den unterschiedlichen Gütestufen innerhalb aerodynamischer Simulationsverfahren zu bekommen dient folgend die Berechnung eines DCA-ähnlichen Verdichterprofils (Double Circular Arc, siehe [Lieblein and Seymour, 1955]) mit MISES (siehe [Drela and Youngren, 2008]) als einfaches aber aussagekräftiges Beispiel. Zuerst werden geometrische Variationen vom Verdichterprofil erzeugt und anschließend mit verschiedenen Gütestufen aerodynamisch simuliert. Dafür werden zwei Parameter variiert, zum einen der Staffelungswinkel β_{st}, der in diesem Fall eine Festkörperdrehung des Profils darstellt und zum anderen der Vorderkantenradius r_{LE}. In Abbildung 3.1 werden zwei dieser möglichen Variationen gezeigt. Innerhalb dieses Beispiels werden zwei verschiedene Netzauflösungen miteinander verglichen. Ein grobes Netz mit ca. 920 Zellen und ein feines Netz 16680 Zellen. Die Berechnung mit dem groben Netz ist ca. 10x schneller als die Berechnung mit dem feinen Netz. Die beiden Rechennetze werden in Abbildung 3.2 für eine mögliche Geometrie dargestellt. Die betrachtete Strömungsgröße ist der massenstromgemittelte Totaldruckverlustbeiwert ω (siehe Anhang A.1.1) und in

Name	Min.	Max.
Anströmmachzahl	0.9	0.9
Anströmwinkel	141°	141°
Kanalkontraktion	1.1	1.1
Staffelungswinkel	27°	32°
VK-radius	0.006	0.011

Tabelle 3.1: Randbedingungen für den Testfall

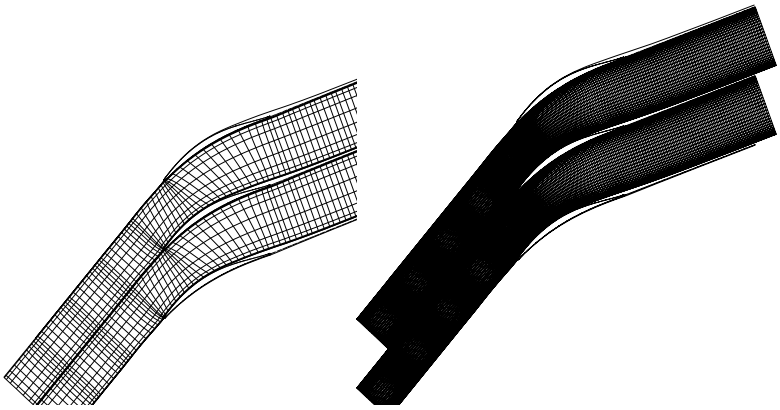


Abbildung 3.2: Darstellung der untersuchten Rechengitterauflösungen

Tabelle 3.1 werden die wichtigsten Randbedingungen für den Testfall aufgelistet.

Abbildung 3.3 zeigt die Totaldruckverluste der verschiedenen Gütestufen über dem variierten Staffelungswinkel für jeweils zwei verschiedene Vorderkantenradien. Alle vier Kurvenverläufe besitzen eine starke Ähnlichkeit. Bei r=0.011 hat sich das Minimum der Verlustpolaren um ca. 1 ° verschoben. Außerdem besitzt die Verlustpolare bei höheren Staffelungswinkeln einen steileren Anstieg. Insbesondere die Verschiebung des Minimums hat in einer Optimierung einen wesentlichen Einfluss auf das Ergebnis. Dennoch bietet die niedrigere Gütestufe einen sehr hohen Informationsgehalt bei deutlich kürzerer Laufzeit. Zusammenfassend lässt sich sagen, dass das gezeigte Beispiel die starken Ähnlichkeiten der Gütestufen zeigt, welche sich innerhalb eines Multifidelity-Verfahrens effizient nutzen lassen.

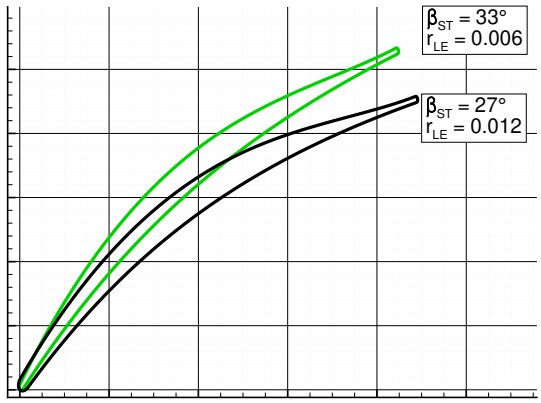


Abbildung 3.1: Beispielhafte Geometrische Variationen des DCA-ähnlichen Profils

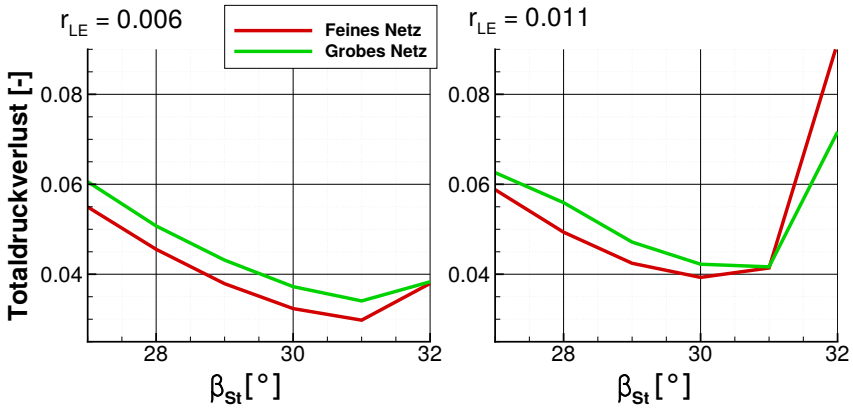


Abbildung 3.3: Verlustpolaren für zwei verschiedene Vorderkantenradien.

3.1.2 Beispiel für verschiedene Simulationsverfahren

Neben der Verwendung verschiedener Gütestufen innerhalb eines Simulationsverfahrens, lassen sich auch unterschiedliche Simulationsverfahren koppeln. Vorraussetzung dafür ist, dass beide Verfahren den gleichen Parametrisierungsvektor haben.

Schnös et al. [Schnös and Nicke, 2017] vergleichen in ihrer Arbeit die Ergebnisse zweier Simulationsverfahren für dieselbe Profilgeometrie. Die verwendeten Strömungslöser sind MISES (siehe [Drela and Youngren, 2008]) und der 3D-Navier-Stokes-Strömungslöser TRACE (siehe [Kügeler, 2005, Nürnberger, 2004]). Aerodynamisch werden die Verlustpolaren des Rotor 1-Mittelschnittprofils vom RIG250-Verdichter (siehe [Schönweitz et al., 2013]) verglichen. Abbildung 3.4 zeigt die berechneten Verlustpolaren und den Abströmwinkel als Funktion des Staffelungswinkels. Während die Verlustpolaren eine sehr ähnliche Charakteristik aufweisen, sind beim Verlauf des Abströmwinkels deutliche Unterschiede im berechneten Verhalten zu erkennen.

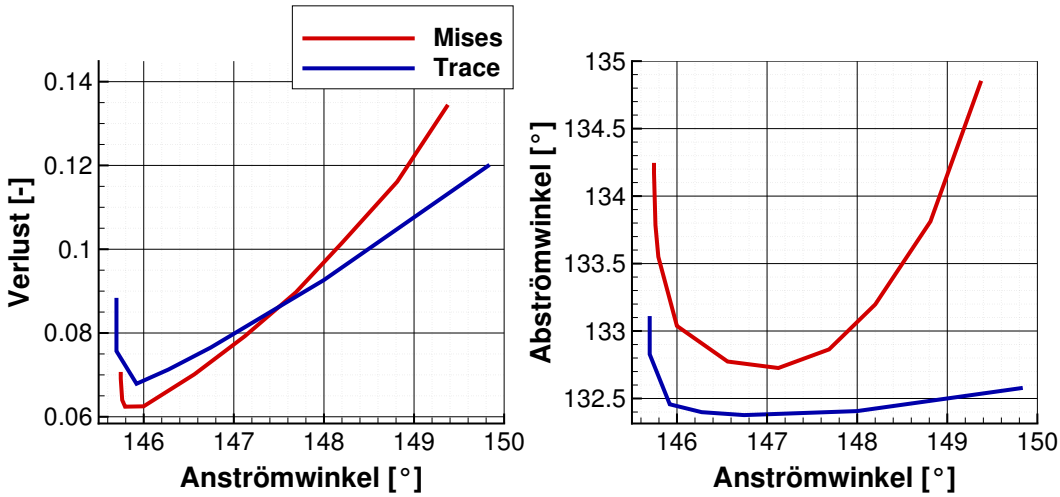


Abbildung 3.4: Vergleich zwischen dem 2D-CFD-Verfahren MISES und dem 3D-CFD-Verfahren TRACE anhand eines transsonischen Verdichterprofils

Die vorangestellten Beispiele lassen allerdings keine allgemeine Aussage über die Nutzbarkeit unterschiedlicher Simulationsverfahren innerhalb von Multifidelity-Optimierungen zu, da die funktionalen Zusammenhänge zwischen den Gütestufen sehr unterschiedlich ausfallen können und vom jeweiligen Anwendungsfall sowie den verwendeten Simulationsverfahren abhängen. Zudem können die Funktionale auch große Unterschiede aufweisen, sodass die Daten nur sehr schwach korreliert sind und der Einsatz eines Multifidelity-Verfahrens dann nicht mehr lohnenswert ist.

Letztlich muss für jeden Einzelfall untersucht werden, inwiefern die verwendeten Gütestufen für einen Einsatz innerhalb einer Multifidelity-Optimierung geeignet sind oder auch nicht. In Kapitel 6 werden verschiedene Anwendungen gezeigt, welche im Rahmen dieser Arbeit betreut oder durchgeführt wurden. Mit Hilfe dieser Ergebnisse ist eine erste Einschätzung einiger Anwendungen möglich.

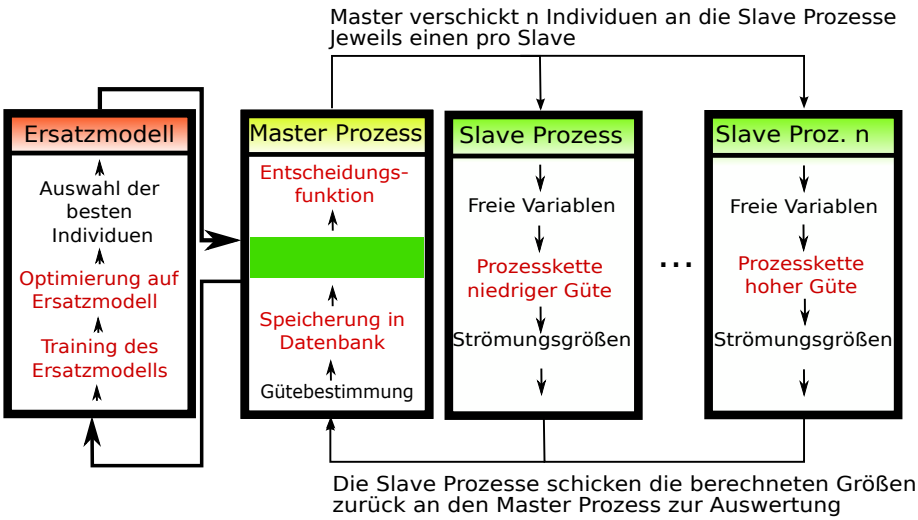


Abbildung 3.5: Darstellung des Multifidelity-Optimierungsprozesses

3.2 Multifidelity-Optimierungsstrategie in AutoOpti

Innerhalb dieses Abschnitts soll die hier entwickelte Multifidelity-Optimierungsstrategie beschrieben werden. Das Verfahren wurde in die Optimierungssoftware AutoOpti eingebunden, welches in Kapitel 2.3 beschrieben wird. Die notwendigen Änderungen an dem grundlegenden Optimierungs-Prozess und der Datenhaltung werden im folgenden Unterkapitel 3.2.1 erläutert. Darauffolgend wird in Unterkapitel 3.2.2 ein besonderes Augenmerk auf die im Rahmen dieser Arbeit entwickelte Entscheidungsfunktion gelegt. Diese Funktion soll während einer laufenden Optimierung automatisiert entscheiden können, mit welcher Gütestufe das nächste Individuum berechnet werden soll. Da die Entscheidungsfunktion einen sehr großen Einfluss auf die Effizienz einer Multifidelity-Optimierung hat, wird innerhalb dieses Kapitels ein effizientes automatisiertes Verfahren vorgestellt.

3.2.1 Multifidelity-Optimierungsprozess

Der hier vorgestellte Prozess stellt eine Erweiterung des in Kapitel 2.3.1 vorgestellten Optimierungsprozesses dar und ist schematisch für den Spezialfall von zwei Gütestufen in Abbildung 3.5 dargestellt. Die Veränderungen zu dem in Kapitel 2.3.1 beschriebenen Prozess sind in rot markiert und werden im Folgenden beschrieben.

Änderungen am Training der Ersatzmodelle: Die in AutoOpti eingesetzten Ersatzmodelle müssen vor der Verwendung trainiert werden. Bei der hier vorgestellten Multifidelity-Strategie liegt die hauptsächliche Neuerung in der Verwendung eines anderen Ersatzmodells, welches die verschiedenen Gütestufen verwerten kann. Die Entwicklung der Software des neuen Ersatzmodells und dessen Qualifizierung für den industrienahen Einsatz ist einer der Kernpunkte dieser Arbeit. Zur Anwendung kommt ein Co-Kriging-Verfahren (siehe [Kennedy and O'Hagan, 2000]), dessen theoretische Grundlage in Kapitel 4 beschrieben werden.

Bewertung der Individuen: Für eine Multifidelity-Optimierung stellt sich die Frage, wie man die Gütebestimmung der Individuen durchführt, bspw. die Bestimmung des Paretorangs oder des Volumenzugewinns. Grundsätzlich erscheint es sinnvoll, dafür nur die Individuen hoher Güte zu verwenden, da der Vergleich zwischen verschiedenen Gütestufen sehr schwer fällt. Außerdem interessiert den Anwender in der Regel nur das hochwertigste Ergebnis.

Aus diesen Gründen wird die Gütebestimmung nur an der höchsten Gütestufe durchgeführt. Das wiederum bedeutet, dass nur Individuen der höchsten Stufe einen direkten Optimierungsfortschritt in Form eines Volumenzugewinns bringen können. Individuen niedriger Güte können also nur indirekten Einfluss auf den Optimierungsfortschritt nehmen, indem sie das Ersatzmodell verbessern und den weiteren Optimierungsverlauf so günstig beeinflussen.

Änderungen an der Datenbank: Die bewerteten Individuen der verschiedenen Gütestufen werden in getrennten Datenbanken gespeichert.

Änderungen an der Prozesskette: Die Prozesskette wird in AutoOpti als Liste von nacheinander auszuführenden Prozessen hinterlegt und bei Bedarf mit einem Parametersatz gestartet. Alle dafür notwendigen Programme und Dateien werden vom Benutzer in einem Template-Ordner hinterlegt. Im Falle einer Multifidelity-Optimierung, muss für jede Gütestufe eine eigene Prozesskette und ein eigener Template Ordner angelegt werden. An der eigentlichen Prozesskettensteuerung muss also nichts verändert werden.

Optimierung auf den Ersatzmodellen: Wie auch bei einer Optimierung mit nur einer Gütestufe, wird nach erfolgreichem Training eine eigene Optimierung auf den Ersatzmodellen gestartet. An diesem Punkt ändert sich nichts, da die Optimierung auf dem Ersatzmodell nur auf der höchsten Gütestufe stattfindet.

Entscheidungsfunktion: Bevor ein neues Individuum mittels Prozesskette bewertet wird, muss entschieden werden, mit welcher Gütestufe dieser berechnet wird. Dies wird durch die Entscheidungsfunktion festgelegt, welche eine fundamentale Bedeutung für die Optimierungsbeschleunigung und die effektive Nutzung der neuen Ersatzmodelle hat.

3.2.2 Entscheidungsfunktion

Folgend wird eine Übersicht über mögliche Entscheidungsfunktionen gegeben und darauf aufbauend die hier entwickelte Strategie beschrieben.

Eine Gütestufe wird mit dem Index $g \in J = \{1, ..., s\}$ gekennzeichnet, wobei eine niedrigere Gütestufe einen höheren Index besitzt. Da alle Gütestufen aus einem Simulationsprozess stammen, geben sie die Realität nicht fehlerfrei wieder. Jeder dieser Prozesse hat somit einen mittleren Fehler F_g und eine Ausführungszeit t_g. Es wird

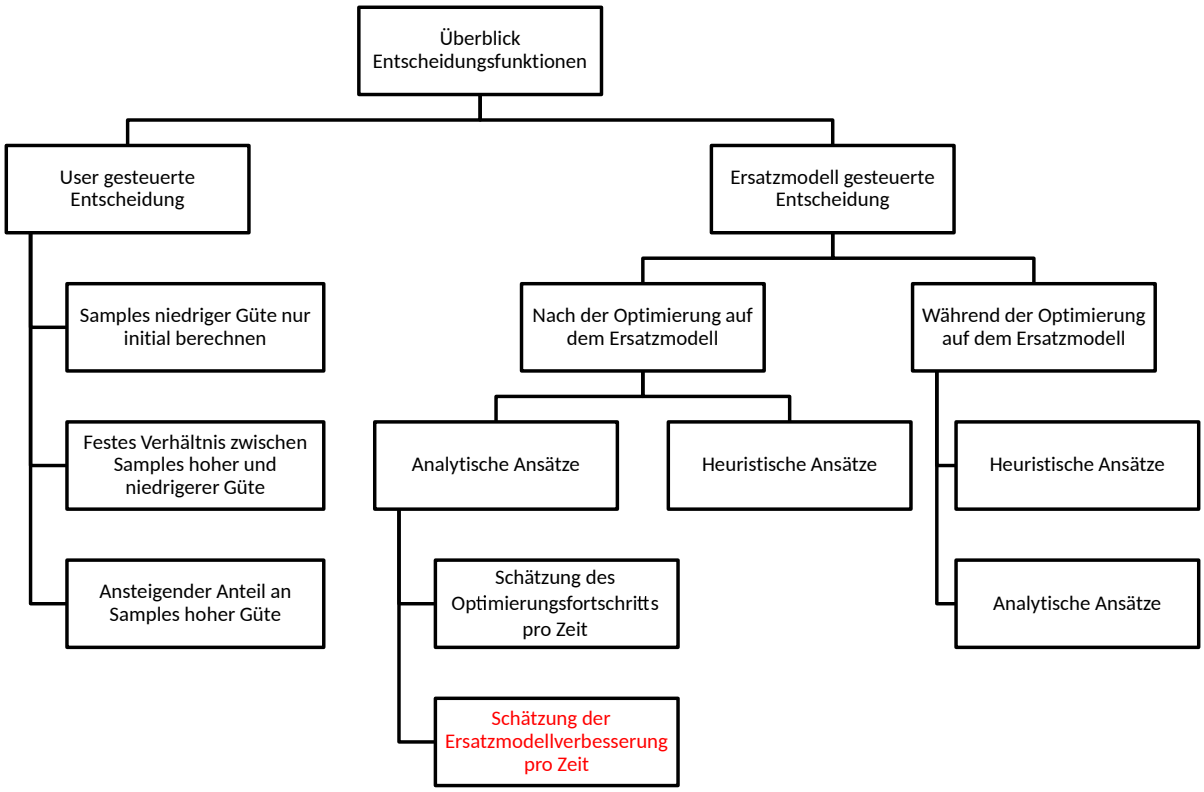


Abbildung 3.6: Möglichkeiten der Individuenerzeugung mit mehreren Gütestufen

davon ausgegangen, dass ein Individuum höherer Güte einen niedrigeren Fehler hat, aber länger für die Ausführung benötigt. Es gilt also $F_g \leq F_{g+1}$und $t_g \geq t_{g+1}$. Da für die Bewertung einer Gütestufe viele Kriterien denkbar sind (beispielsweise ein erwarteter Fehler oder ein erwarteter Optimierungsfortschritt), werden diese durch den Begriff des Informationszugewinns I_g zusammengefasst. Eine nähere Definition des Informationszugewinns folgt im Abschnitt „Nach der Optimierung auf dem Ersatzmodell:“.

Auf Basis dieser Werte (I_g, F_g, t_g) wird während einer laufenden Multifidelity-Optimierung permanent eine Entscheidung getroffen, welche Prozesskette als nächstes durchlaufen werden soll. Benötigt wird also eine Entscheidungsfunktion f, welche diese Entscheidung während einer laufenden Optimierung vornimmt und den Optimierungsverlauf möglichst günstig beeinflusst. In Abbildung 3.6 sind verschiedene Möglichkeiten der Entscheidungsfindung zusammengefasst und kategorisiert, welche im Folgenden beschrieben werden.

3.2.2.1 Benutzergesteuerte Entscheidungsfunktionen

Eine häufig verwendete Methode ist es, vor dem Beginn der eigentlichen Optimierung einen festen Datensatz an Individuen niedriger Güte zu erzeugen und diese dem Ersatzmodelltraining hinzuzufügen. Während der Optimierung wird dann nur noch die Prozesskette hoher Güte verwendet. Durch die anfangs hinzugefügten Individuen niedriger Güte kann das Ersatzmodell verbessert werden, was dann zu einer Beschleunigung der Optimierung führt. Diese Methode ist sehr einfach umzusetzen, wird aber nur am Anfang der Optimierung einen signifikanten Zugewinn bringen. Zudem muss die

Beschleunigung der Optimierung größer sein als die anfangs investierte Arbeit für die Erzeugung der Individuen niedriger Güte.

Die einfachste Variante einer nicht trivialen Entscheidungsfunktion ist die Vorgabe einer Gesetzmäßigkeit durch den Anwender. Hier wird der Verlauf der Optimierung bzw. die vorhandenen Informationen über die Güte des Ersatzmodells nicht berücksichtigt. Sinnvoll erscheint es am Anfang einer Optimierung relativ viele Individuen niedriger Güte zu erzeugen um den groben Funktionsverlauf darzustellen und dann gegen Ende nur noch Individuen hoher Güte zu erzeugen. Hierbei ist zu berücksichtigen, dass auch am Anfang einer Optimierung eine minimale Anzahl von Individuen aller Gütestufen erforderlich ist, um ein initiales Ersatzmodell zu trainieren. Die praktische Umsetzung erfolgt dann über die Vorgabe der gewünschten Wahrscheinlichkeit ein Individuum niedriger Güte zu erzeugen in Abhängigkeit des Optimierungsfortschritts. Der Erfolg dieser Art von Entscheidung hängt stark von den gewählten Parametern (z.B. die Wahrscheinlichkeit für die Erzeugung einer Gütestufe zu einem bestimmten Zeitpunkt der Optimierung) und dem vorliegenden Optimierungsproblem ab, sodass die Wahl dieser selbst für einen erfahrenen Anwender schwierig ist. Zudem bieten benutzergesteuerte Entscheidungsfunktionen einen geringen bis keinen Automatisierungsgrad und sind daher für industrienahe Anwendungen ungeeignet. Aus diesen Gründen soll diese Art der Entscheidung nicht weiter betrachtet werden.

3.2.2.2 Ersatzmodellgesteuerte Entscheidungsfunktionen

In diesem Abschnitt werden Entscheidungsfunktionen vorgestellt, die die Möglichkeit bieten, die vorhandenen Informationen der Ersatzmodelle einfließen zu lassen. Zum einen sind dies Entscheidungsfunktionen vom Typ „Nach der Optimierung auf dem Ersatzmodell“ und zum anderen „Während der Optimierung auf dem Ersatzmodell“. In dieser Arbeit wird das Verfahren „Nach der Optimierung auf dem Ersatzmodell“ favorisiert. Die Begründung dafür wird in den Abschnitten selbst erläutert.

Während der Optimierung auf dem Ersatzmodell: Die Optimierung auf dem Ersatzmodell schlägt in der Regel einen neuen Vektor freier Parameter $\vec{x_0} \in \mathbb{R}^k$ für einen vielversprechendes Individuum vor. Eine sinnvolle Strategie ist es, während dieser Optimierung auf dem Ersatzmodell die Wahl der Gütestufe einfließen zu lassen. Ein solches Verfahren ist allerdings schwer umsetzbar:

Am Ende einer Optimierung sind nur die Paretorang 1 Individuen der höchsten Gütestufe von Interesse, da das Vertrauensniveau für diese am höchsten ist. Aus diesem Grund bringt ein neues Individuum niedriger Gütestufe keinen direkten Optimierungsfortschritt, sondern nur eine Reduktion der Varianz der Ersatzmodelle. Durch die verbesserten Ersatzmodelle kann im übernächsten Optimierungsschritt ein Fortschritt erzielt werden. Um diese Verbesserung berechnen zu können, muss geschätzt werden, inwiefern ein Individuum hoher oder niedriger Gütestufe (die ebenfalls geschätzt werden müssen) den Optimierungsfortschritt im übernächsten Optimierungsschritt beeinflusst.

Eine typische Optimierung auf dem Ersatzmodell in AutoOpti wird in der Regel mit 5000 Optimierungsschritten durchgeführt. Eine Möglichkeit diese Schätzung zu realisieren

ist die Durchführung einer weiteren Optimierung in jedem der oben genannten 5000 Optimierungsschritte. Der Aufwand ist damit zu hoch.

In der einschlägigen Literatur ist bisher nur sehr wenig zu dieser Thematik vorhanden. Eine heuristische Lösung für dieses Problem kann in der Arbeit von Huang et. al. (siehe [Huang et al., 2006]) gefunden werden. Dieser Ansatz stellt allerdings keine analytisch nachvollziehbare Lösung des Problems dar und inwiefern ein heuristischer Ansatz immer zu einer Beschleunigung führt, kann ebenfalls nur schwer abgeschätzt werden.

Vielversprechender scheint es zu sein, ein Individuum mit hohem erwarteten Optimierungsfortschritt zu suchen und erst dann zu entscheiden mit welcher Gütestufe dieser berechnet werden soll. Diese Art der Strategie wird im nächsten Abschnitt „*Nach der Optimierung auf dem Ersatzmodell*“ vorgestellt.

Nach der Optimierung auf dem Ersatzmodell: Die hier beschrieben Strategie setzt nach der Optimierung auf dem Ersatzmodell an. Innerhalb dieser wird ein Individuum mit hohem erwarteten Optimierungsfortschritt ausgewählt. Daraufhin muss entschieden werden, mit welcher Gütestufe $g \in J = \{1, ..., s\}$ dieses Individuum M berechnet werden soll. Das gewählte Individuum M besitzt einen festen Ort $\vec{x_0} \in \mathbb{R}^k$ mit den Ersatzmodellvorhersagen $\vec{y}^* \in \mathbb{R}^q$ für q Ersatzmodelle und den zugehörigen Varianzen $\vec{\sigma^2}^* \in \mathbb{R}^q$. Weiterhin ist die Zeit für die Erzeugung eines Individuums t_g der Gütestufe $g \in J$ bekannt. Hierzu zählen die Prozesskettenzeiten, die Trainingszeit und die Zeit für die Optimierung auf dem Ersatzmodell. Mithilfe der Zeiten und der Ersatzmodellvorhersagen soll eine Entscheidung getroffen werden.

Im Folgenden wird erst allgemeingültig die Funktionsweise einer geeigneten Entscheidungsfunktion erläutert und darauf aufbauend eine konkrete Umsetzung gezeigt. Hierfür ist ein Maß für den Zugewinn an Information durch ein neues Individuum notwendig. Informationszugewinn wird in diesem Zusammenhang als Überbegriff für alle möglichen Bewertungskriterien (wie z.B. Reduktion der Ersatzmodellunsicherheit, Zuwachs im erwarteten Volumenzugewinn, etc.) verwendet. Daher wird eine Größe $I_g \in \{I_0, ..., I_s\}$ wobei $0 \leq I_g < \infty$ definiert, welche den Informationszugewinn durch ein Individuum beschreibt. Ein Informationszugewinn von $I_g = 0$ bedeutet, dass keine Information aus dieser Gütestufe gewonnen werden kann. Der Informationszugewinn einer niedrigeren Gütestufe ist kleiner oder gleich dem Informationszugewinn einer höheren Stufe $I_{g+1} \leq I_g$. Außerdem gibt es für jede Gütestufe jeweils nur einen Informationszugewinn, jedoch viele Zielfunktionale und Nebenbedingungen, also eine Abbildung in der Form $\mathbb{R}^{c+z} \to \mathbb{R}$. Neben dem Informationszugewinn spielen die Laufzeiten t_g der Prozessketten der jeweiligen Gütestufen eine erhebliche Rolle. Je größer der Faktor zwischen der Laufzeit hoher Güte zu niedriger Güte $\frac{t_g}{t_{g+1}}$ ist, desto eher lohnt sich ein Durchlauf niedriger Güte (vorausgesetzt $I_{g+1} > 0$). Beachtet man zusätzlich noch den Informationszugewinn, so ist es sinnvoll, diesen ins Verhältnis zu den Prozesskettenzeiten zu setzten $\frac{I_g}{t_g}$, also dem Informationszugewinn pro Zeit für jede Gütestufe. Ein größerer Wert entspricht mehr Informationszugewinn pro benötigter Erzeugungszeit. Eine Entscheidungsfunktion $f_{dec}(\vec{x_0})$ für beliebig viele Gütestufen s wählt

dann die Gütestufe $g_{best} \in J = \{1, ..., s\}$ mit dem höchsten Verhältnis von $\frac{I_g}{t_g}$:

$$f_{dec}(\vec{x_0}) = \arg\max_{j \in J} \left(\frac{I_j(\vec{x_0})}{t_j} \right) \tag{3.1}$$

Die gewählte Gütestufe wird wie folgt definiert: $g_{best} := f_{dec}(\vec{x_0})$. Um eine solche Entscheidungsfunktion umsetzen zu können, müssen $I_g(\vec{x_0})$ und t_g geschätzt werden. Die Schätzung der Erzeugungszeit t_g kann aus älteren Daten erfolgen. Die in dieser Arbeit entwickelte Umsetzung dieser Schätzung wird in Kapitel 5.4 gezeigt. Für die Modellierung des Informationszugewinns eignen sich statistische Ersatzmodelle, welche neben einem Erwartungswert für einen unbekannten Ort $\vec{x_0} \in \mathbb{R}^k$ auch eine Unsicherheit in Form einer Varianz $\sigma^{2*} \in \mathbb{R}$ oder Standardabweichung $\sigma^* \in \mathbb{R}$ voraussagen. Fügt man einem solchen Ersatzmodell an einem unbekannten Ort eine neue Stützstelle hinzu, so wird die vorhergesagte Varianz dort reduziert. Im Falle eines Co-Krigings wird die Varianz je nach Gütestufe mehr oder weniger stark reduziert. Dieses Verhalten lässt sich für eine Entscheidung heranziehen indem man den Informationszugewinn als Varianzreduktion definiert. Die Größe der Reduktion ist eine sehr komplexe Funktion, die von zahlreichen Faktoren abhängt: dem verwendeten Ersatzmodell, der betrachteten Gütestufe, ersatzmodellspezifischen Parametern, dem Abstand von $\vec{x_0} \in \mathbb{R}^k$ zu allen anderen bekannten Stützstellen, usw.

Um einen Eindruck der Varianzreduktion zu bekommen, wird in Abbildung 3.7 ein einfaches Beispiel gezeigt. Die blaue gestrichelte Kurve stellt die reale Funktion dar, in diesem Fall eine Sinus Schwingung. Die Stützstellen hoher Güte werden durch rote Quadrate dargestellt und die Stützstellen niedriger Güte durch blaue Dreiecke. Den Stützstellen niedriger Güte wurde ein normalverteilter Fehler hinzugefügt. Bei dem genutzten Ersatzmodell handelt es sich um ein CO-Kriging wie es im nächsten Kapitel vorgestellt wird. Die rote Kurve stellt die Vorhersage dieser Funktion anhand von den Stützstellen hoher und niedriger Güte dar. An der markierten Stelle $\vec{x_0}$ erfolgte eine Vorhersage niedriger Güte. Die schwarz gestrichelten Linien zeigen im oberen Diagramm die Vorhersage ohne diese Stützstelle und die dazugehörige Standardabweichung im unteren Diagramm.

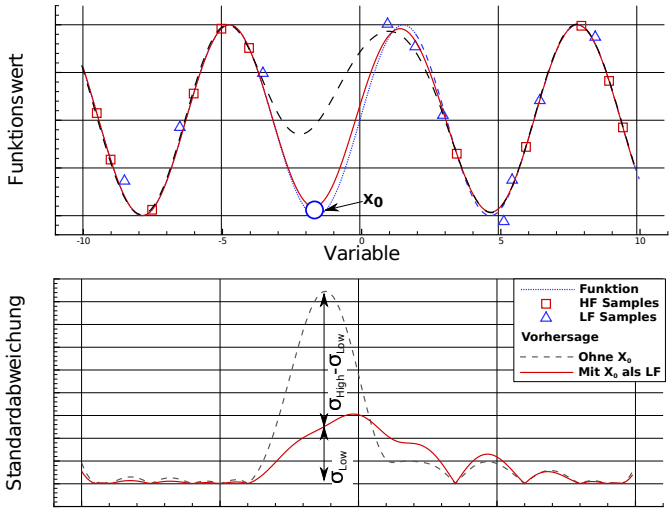


Abbildung 3.7: Beispielhafte Entwicklung der vorhergesagten Standardabweichung beim Hinzufügen von Stützstellen verschiedener Gütestufen (LF bedeutet niedrige- und HF hohe Güte).

Wichtig ist hierbei, dass die vorhergesagte Standardabweichung beim Hinzufügen einer Stützstelle niedriger Güte reduziert wird. Es bleibt in der Regel also eine Restun-

sicherheit bestehen. Beim Hinzufügen eines Stützstellen der höchsten Güte wird die Standardabweichung im Normalfall mit Null vorhergesagt. Die Reduzierung der Standardabweichung ist also je nach Gütestufe unterschiedlich groß, wobei die niedrigeren Gütestufen eine größere Restunsicherheit aufweisen als die höheren.

Folgend soll ein Entscheidungskriterium für ein einzelnes Ersatzmodell auf Basis dieser Varianzreduktion vorgestellt werden. Im weiteren Verlauf des Kapitels wird diese dann auf viele Ersatzmodelle erweitert.

Die Vorhersage der Varianz einer Gütestufe g für ein Ersatzmodell an einer Stelle $\vec{x_0} \in \mathbb{R}^k$ wird im Folgenden mit $\sigma_g^2 \in \mathbb{R}$ benannt. Bei Kenntnis einer Stützstelle $y_j(\vec{x}_0)$ der Güte j wird die bedingte Varianz mit $\sigma_{g|j}^2 = \sigma_g^2(\vec{x}_0) \mid y_j(\vec{x}_0) \in \mathbb{R}$ bezeichnet. Eine effiziente Möglichkeit die bedingte Varianz $\sigma_{g|j}^2$ zu berechnen, wird in Kapitel 5.4 beschrieben. Der Informationszugewinn einer Gütestufe g unter der Bedingung, dass eine Stützstelle der Gütestufe j für eine Stelle $\vec{x_0} \in \mathbb{R}^k$ bekannt ist, kann damit als Reduktion der vorhergesagten Varianz modelliert werden:

$$I_{g|j}(\vec{x_0}) = \sigma_g^2 - \sigma_{g|j}^2 \tag{3.2}$$

Zur Ausführung von Gleichung 3.1 werden die jeweiligen Zeiten für die Erzeugung eines neuen Individuums $t_j \in \mathbb{R}$ der Gütestufe j benötigt. Diese Zeiten setzen sich im Wesentlichen aus drei Teilen zusammen. Der erste Teil umfasst das Training der Ersatzmodelle und benötigt die Zeit $t_{train} \in \mathbb{R}$. Der zweite Teil besteht aus der Optimierung auf dem Ersatzmodell und benötigt die Zeit $t_{opti} \in \mathbb{R}$. Der dritte und damit letzte Teil beinhaltet die Berechnung der Prozesskette $t_{pr,j} \in \mathbb{R}$. Die Zeiten $t_{train} \in \mathbb{R}$ und $t_{opti} \in \mathbb{R}$ sind unabhängig von den Gütestufen, $t_{pr,j} \in \mathbb{R}$ hingegen ist abhängig von der Gütestufe j. Damit können die jeweiligen Erzeugungszeiten definiert werden, wobei die genaue Berechnung dieser Zeiten in Kapitel 5.4 beschrieben wird:

$$t_j = t_{train} + t_{opti} + t_{pr,j} \tag{3.3}$$

Mit dem Informationszugewinn und den Zeiten kann man die in Gleichung 3.1 definierten Verhältnisse von Informationszugewinn pro Zeit festlegen:

$$\frac{I_{g|j}}{t_j} = \frac{\sigma_g^2 - \sigma_{g|j}^2}{t_{train} + t_{opti} + t_{pr,j}} \tag{3.4}$$

Eingesetzt in Gleichung 3.1, ergibt sich eine Entscheidungsfunktion für ein Ersatzmodell:

$$f_{dec}(\vec{x_0}) = \arg\max_{j \in J} \left(\frac{\sigma_g^2 - \sigma_{g|j}^2}{t_{train} + t_{opti} + t_{pr,j}} \right) \tag{3.5}$$

Gewichtung der Ersatzmodelle In einer typischen Optimierung werden mehrere Ersatzmodelle trainiert (vgl. Kapitel 2.3.2). Daher soll Gleichung 3.5 entsprechend angepasst werden. Für jede der $z \in \mathbb{N}$ „Flowparameter“ der Zielfunktionen (Definition Kapitel 2.1.2) und $c \in \mathbb{N}$ Nebenbedingungen existiert ein Ersatzmodell. Um alle Vorhersagen in der Entscheidungsfunktion zu berücksichtigen, bietet sich eine gewichtete

Summe an, womit sich Gleichung 3.4 erweitert:

$$\frac{I_{g|j}}{t_j} = \frac{1}{\sum_i^{c+z} w_i} \sum_i^{c+z} w_i \frac{\sigma_{g,i}^2 - \sigma_{g,i|j}^2}{t_{train} + t_{opti} + t_{pr,j}} \tag{3.6}$$

Für die Bestimmung der Gewichte $\vec{w} \in \mathbb{R}^{c+z}$ ist es sinnvoll, die Ersatzmodelle in zwei unterschiedliche Gruppen einzuteilen:

1. Ersatzmodelle für Zielfunktionen
2. Ersatzmodelle für Nebenbedingungen

Typischerweise gibt es in Turbomaschinenoptimierungen zahlreiche Nebenbedingungen, welche am Anfang der Optimierung nicht erfüllt werden. Wie in Kapitel 2.1.2 beschrieben, wird die Erfüllung der Nebenbedingungen vorrangig behandelt. Die eigentlichen Zielfunktionen werden zweitrangig behandelt, bis die Nebenbedingungen erfüllt sind.

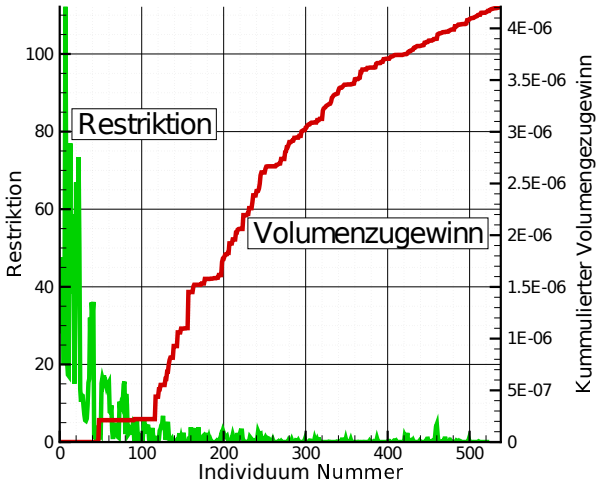


Abbildung 3.8: Typischer Verlauf einer Optimierung im Bezug auf die Restriktionen und Zielfunktionen

Abbildung 3.8 zeigt den typischen Verlauf einer solchen Optimierung (siehe Benchmark-Optimierung Kapitel 6.2). Die grüne Kurve zeigt den summierten Restriktionswert, wobei ein Wert von Null die Erfüllung aller Restriktionen bedeutet und ein Wert größer Null die Nichterfüllung mindestens einer Nebenbedingung. Die rote Kurve beschreibt den kummulierten Volumenzugewinn und kann somit als Optimierungsfortschritt bezogen auf die Zielfunktionen angesehen werden.

Die Abbildung zeigt deutlich, wie im Verlauf der Optimierung ein Wechsel von den Nebenbedingungen hin zu den Zielfunktionenen stattfindet. Die Ersatzmodelle für die Nebenbedingungen werden selbstverständlich weiterhin gepflegt und sind immer Bestandteil der ersatzmodellgestützten Optimierung. Sobald der Optimierungsalgorithmus die Nebenbedingungen einmal erfüllt hat, ist die Wahrscheinlichkeit, diese weiterhin zu erfüllen, höher.

Da die verschiedenen Nebenbedingungen unterschiedlich schwer zu erfüllen sind, sollten diese unterschiedlich stark gewichtet werden. Als geeignete Gewichtung für die Nebenbedingungen kann die Wahrscheinlichkeit $P_i \in \mathbb{R}$ für die Erfüllung der jeweiligen Nebenbedingungen verwendet werden. Nimmt man ferner an, dass die Nebenbedingungen unabhängig sind, so kann die Wahrscheinlichkeit $P_i \in \mathbb{R}$ über die Ersatzmodellvorhersage und die bekannten Intervallgrenzen der Nebenbedingung geschätzt werden. Zudem gilt dann, dass die Gesamtwahrscheinlichkeit zur Erfüllung aller Nebenbedingungen über das Produkt der Einzelwahrscheinlichkeiten $\prod_{i=1}^{c} P_i$ bestimmt werden kann. Über diese Wahrscheinlichkeiten können schließlich die Gewichtungen vorgenommen werden:

1. Gewichtung der einzelnen Nebenbedingungen über die Einzelwahrscheinlichkeiten $w_i = (1 - P_i)\ \forall i \in \{1, ..., c\}$. Je höher die Wahrscheinlichkeit ist eine Nebenbedingung zu erfüllen, desto niedriger ist das Gewicht der Nebenbedingung. Wenn z.B. die Wahrscheinlichkeit bei 100% liegt, so ist diese Nebenbedingung für die Entscheidung unerheblich, da diese unabhängig von der Entscheidung erfüllt wird.
2. Gewichtung aller Zielfunktionen zu allen Nebenbedingungen über die Gesamtwahrscheinlichkeit zur Erfüllung aller Nebenbedingungen.
   (a) Zielfunktionen $\prod_{i=1}^{z} P_i$
   (b) Nebenbedingungen $(1 - \prod_{i=1}^{c} P_i)$

Die Zielfunktionen untereinander werden gleich gewichtet. Daraus ergibt sich:

$$\begin{aligned}
\frac{I_{g|j}}{t_j} &= \frac{1}{\sum_i^{c+z} w_i} \sum_i^{c+z} w_i \frac{\sigma_{g,i}^2 - \sigma_{g,i|j}^2}{t_{train} + t_{opti} + t_{pr,j}} \\
&= \frac{1}{t_{train} + t_{opti} + t_{pr,j}} \frac{1}{\sum_i^{c+z} w_i} \sum_i^{c+z} w_i (\sigma_{g,i}^2 - \sigma_{g,i|j}^2) \\
&= \frac{1}{t_j} \left[\left(1 - \prod_{i=1}^{c} P_i \right) \frac{1}{\sum_i^{c} w_i} \sum_i^{c} w_i (\sigma_{g,i}^2 - \sigma_{g,i|j}^2) \right. \\
&\left. + \left(\prod_{i=1}^{c} P_i \right) \frac{1}{\sum_i^{z} w_i} \sum_i^{z} w_i (\sigma_{g,i}^2 - \sigma_{g,i|j}^2) \right]
\end{aligned} \tag{3.7}$$

$$w_i = (1 - P_i)\ \forall i \in \{1, ..., c\} \tag{3.8}$$

$$w_i = 1\, \forall i \in \{c, ..., z\} \tag{3.9}$$

$$t_j = t_{train} + t_{opti} + t_{pr,j} \tag{3.10}$$

Mit dieser Formulierung kann die Entscheidungsfunktion (siehe Gleichung 3.1) dann verwendet werden. Die algorithmische und softwaretechnische Umsetzung des Verfahrens wird in Kapitel 5.4 detailliert beschrieben.

Globale Varianzreduktion Das Kriterium, welches durch Gleichung 3.7 und 3.6 gegeben ist, bezieht sich auf einen lokalen Ort $\vec{x_0} \in \mathbb{R}^k$. Grundlegend ist es auch möglich, den Einfluss der Entscheidung auf den gesamten Parameterraum zu schätzen oder zumindest auf die unmittelbare Umgebung von $\vec{x_0} \in \mathbb{R}^k$. Um dies zu erreichen, muss die Formulierung aus Gleichung 3.7 zu einem räumlichen Integral erweitert werden. Dieses Vorgehen ist in der Theorie der optimalen Versuchsplanung (auch bekannt unter „Optimal Design of Experiments“) als I-optimales Design zu verstehen. Dort wird die vorhergesagte Varianz über dem gesamten Designraum minimiert, siehe hierzu [Kiefer, 1974, Kiefer, 1959, Bandemer, 1980, Pukelsheim, 1980].

$$I_{g|j}(\vec{x_0}) = \int \left(\sigma_g(\vec{x}) - [\sigma_g(\vec{x}) \mid y_j(\vec{x}_0)] \right) d\vec{x} \tag{3.11}$$

Die Formulierung $\sigma_g(\vec{x}) \mid y_j(\vec{x}_0)$ soll die Vorhersage der Unsicherheit $\sigma_g(\vec{x}) \in \mathbb{R}$ der Gütestufe $g \in J = \{1, ..., s\}$ an einer beliebigen Stelle $\vec{x} \in \mathbb{R}^k$ darstellen und zwar

unter der Voraussetzung, dass an der Stelle $\vec{x_0} \in \mathbb{R}^k$ eine Stützstelle $y_j(\vec{x_0}) \in \mathbb{R}$ der Gütestufe $j \in J = \{1, ..., s\}$ bekannt ist. Damit ändert sich Gleichung 3.7, die Gewichte und Zeiten bleiben wie in den Gleichungen 3.7 - 3.10:

$$\frac{I_{g|j}}{t_j} = \frac{1}{t_j}\left[\left(1 - \prod_{i=1}^{c} P_i\right)\frac{1}{\sum_i^c w_i}\sum_i^c w_i I_{g|j}(\vec{x_0}) + \left(\prod_{i=1}^{c} P_i\right)\frac{1}{\sum_i^z w_i}\sum_i^z w_i I_{g|j}(\vec{x_0})\right] \quad (3.12)$$

Um zu testen, inwiefern sich eine globale Entscheidungsfunktion gemäß Gleichungen 3.12 und 3.1 von einer lokalen gemäß Gleichungen 3.7,3.1 unterscheidet, wurden einige eindimensionale analytische Beispiele untersucht. Von 10 unterschiedlichen Beispielen konnte jedoch nur eins generiert werden, in dem das Kriterium 3.12 zu einer anderen Entscheidung führt als das Lokale. In allen anderen Fällen liefern die globale und lokale Entscheidungsfunktion immer ein identisches Ergebnis. In Anhang A.1.2 sind die Ergebnisse dieses Beispiels dargestellt.

Die größte Schwierigkeit zur Nutzung einer globalen Entscheidungsfunktion gemäß Gleichung 3.12 liegt in der numerischen Berechnung der Integrale. Da die Dimension der Parameterräume und auch die Anzahl der Zielfunktionen und Nebenbedingungen bei typischen Turbomaschinenoptimierungen sehr hoch ist, ist die Berechnung eines solchen Integrals mit einem enormen numerischen Aufwand verbunden.

3.2.3 Theroretische Beschleunigung einer Multifidelity Optimierung

In diesem Abschnitt wird beschrieben, wie die Beschleunigung einer Multifidelity-Optimierung gegenüber einer Single-Fidelity-Optimierung zu berechnen und zu bewerten ist. Insbesondere wird erklärt, wie die maximal mögliche Beschleunigung unter bestimmten Voraussetzungen realistisch eingeschätzt werden kann und von welchen Faktoren diese abhängt.

Gleichung 3.13 beschreibt den zeitlichen Faktor t_{fact} zwischen einer Single-Fidelity-Referenzoptimierung und einer Multifidelity-Optimierung. $t_{fact} > 1$ bedeutet hierbei eine Beschleunigung gegenüber der Referenz und $t_{fact} < 1$ eine Verlangsamung.

$$t_{fact} = \frac{t_{HF,Referenz}}{t_{MF}} = \frac{t_h\left(i_{best} + i_{initH}\right)}{t_h\left(i_{initH} + \frac{t_l}{t_h}i_{initL} + i_{bestH} + \frac{t_l}{t_h}i_{bestL}\right)} \quad (3.13)$$

i_{bestH}, i_{bestL} beschreiben die Anzahl der Individuen ab der Initialisierung bis zur Konvergenz, der Index H, L steht jeweils für hohe und niedrige Güte. i_{initH}, i_{initL} geben die Anzahl der Individuen der Initialisierung der jeweiligen Optimierung an. Das theoretische Beschleunigungsmaximum $max(t_{fact})$ (siehe Gleichung 3.14) einer Multifidelity-Optimierung ist dann erreicht, wenn die niedrige Gütestufe gleich der hohen Gütestufe ist und das Optimum mit einem Individuum gefunden wurde. In diesem Fall müsste dieselbe Anzahl an Individuen ausreichen $i_{best} = i_{bestL}$.

$$max(t_{fact}) = \frac{t_h\left(i_{best} + i_{initH}\right)}{t_h\left(i_{initH} + \frac{t_l}{t_h}i_{initL} + \frac{t_l}{t_h}i_{best} + 1\right)} \quad (3.14)$$

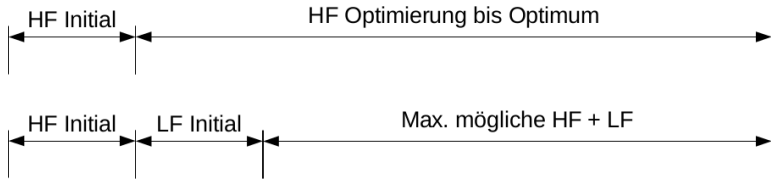


Abbildung 3.9: Die Zeitstrahlen von zwei fiktiven Optimierungen. Der obere entspricht einer Single-Fidelity-Optimierung und der untere einer Multifidelity-Optimierung.

Der Grenzwert dieser Funktion ist das Verhältnis von $\lim_{i_{best} \to \infty} max(t_{fact}) = \frac{t_h}{t_l}$. Dieser Wert kann aber aus folgenden Gründen praktisch nicht erreicht werden:

- Die Stützstellen aus der Intitialisierungsphase liegen oft nicht in der Nähe des Optimums. Daher müssen meistens noch einige Stützstellen erzeugt werden, bis die Umgebung des Optimums entsprechend aufgelöst wird.
- Bei einer Mehrzieloptimierung gibt es kein eindeutiges Optimum. Es muss eine Paretofront erzeugt werden, die sich mit einer Stützstelle nicht auflösen lässt.
- Zudem hängt die Beschleunigung vom Informationsgehalt der Funktion niedriger Güte ab. Bietet diese nur einen geringen Informationsgehalt, so kann die theoretische Beschleunigung ebenfalls nicht erreicht werden.

Was aus Gleichung 3.13 ersichtlich ist, ist die Wichtigkeit der Auswahl der initialen Anzahl an Individuen niedriger Güte. Diese sollte mit dem Faktor $\frac{t_l}{t_h}$ korrelieren, da der initiale Zeitverlust sonst zu hoch ist und durch die Multifidelity-Optimierung nicht mehr aufgeholt werden kann. Abbildung 3.9 verdeutlicht diese Problematik. Der obere Zeitstrahl zeigt eine High-Fidelity-Optimierung und der untere eine Multifidelity-Optimierung. Diese besitzt einen zusätzlichen Initialsierungsteil niedriger Güte. Soll mit der Multifidelity-Optimierung eine Beschleunigung gegenüber der High-Fidelity-Optimierung erreicht werden, so ergibt sich eine maximal mögliche Optimierungzeit.

Um eine optimale Initialisierung zu erreichen, sind folgende Punkte von besonderer Bedeutung:

1. Die Prozesskettengeschwindigkeit der niedrigen Güte: Je schneller diese ist, desto mehr initiale Individuen sollten berechnet werden.
2. Die Korrelation zwischen den Gütestufen: Nur bei ausreichendem Zusammenhang können die Stützstellen niedriger Güte gewinnbringend verwendet werden.
3. Die Komplexität der Zielfunktion und der Nebenbedingungen. Je komplexer die Optimierung ist, desto länger würde eine reine High-Fidelity-Referenzoptimierung benötigen und desto mehr Potential ist für eine Beschleunigung vorhanden.

Zumindest die ersten beiden genannten Punkte lassen sich bereits vor dem Start einer Optimierung relativ gut abschätzen:

1. Die Prozesskettengeschwindigkeit ist in der Regel bekannt oder kann während einer Optimierung einfach gemessen werden.
2. Die Korrelation der Gütestufen kann mit einer geringen Anzahl an Stützstellen abgeschätzt werden oder ist unter Umständen bereits bekannt. Oftmals reicht hier eine grobe Schätzung aus.

4 Kriging-Verfahren: Analytische Herleitung

Da die in dieser Arbeit entwickelte Ersatzmodell-Software einen hohen Anwendungsbezug hat und auch industriell eingesetzt werden soll, wurde bei der mathematischen Herleitung ein besonderes Augenmerk auf eine möglichst gute softwaretechnische Umsetzbarkeit gelegt. Dieser Punkt wird in der einschlägigen Literatur oftmals außer Acht gelassen, was die Effizienz der Software sowie die Generalisierbarkeit negativ beeinflusst.

Im ersten Abschnitt wird ein kurzer Überblick über die Historie und die speziellen Eigenschaften der Kriging-Verfahren gegeben. Darauf folgt die Herleitung der allgemeinen Kriging-Vorhersage-Gleichungen, welche für alle Kriging-Varianten (wie z.B. das Co-Kriging) Gültigkeit besitzen und auch alle innerhalb der entwickelten Software umgesetzt wurden. Anschließen werden die Unterschiede in den Kriging-Unterverfahren und deren Umsetzung beschrieben. Im letzten Abschnitt wird die hier entwickelte Umsetzung des Kriging-Verfahrens mit anderen Verfahren aus der Literatur verglichen.

4.1 Grundlagen Kriging

Unter Kriging versteht man spezielle statistische Verfahren zur Interpolation oder Approximation von Werten an unbeprobten Orten. Der Name des Verfahrens stammt von dem südafrikanischen Bergbauingenieur Daniel Krige (1951). Dieser versuchte eine optimale Interpolationsmethode für den Bergbau zu entwickeln, die auf der räumlichen Abhängigkeit von Messpunkten basiert [Krige, 1953]. Das Verfahren wurde später nach ihm benannt und vom französischen Mathematiker Georges Matheron (1963) zu der heute bekannten Kriging-Theorie [Matheron, 1963] weiterentwickelt. Das Kriging Verfahren hat heute in den Geowissenschaften sowie vielen anderen Forschungsbereichen Verwendung gefunden.

Die Vorteile von Kriging gegenüber anderen Interpolations- oder Approximationsmethoden, wie z.B. Inverser Distanzwichtung [Shepard, 1968], sind:

- Statistische Verfahren wie das Kriging können zusätzlich zum Funktionswert auch eine Unsicherheit vorhersagen. Dies kann innerhalb einer Optimierung ausgenutzt werden (siehe Kapitel 2 und 3).
- Die Initialisierung ist sehr einfach, d.h. das Verfahren liefert so gut wie immer plausible Ergebnisse, unabhängig von irgendwelchen Startparametern. Bei neuronalen Netzwerken bspw. muss vor dem Training die Netztopologie, also die

Anzahl der Gewichte und die Struktur des Netzes angegeben werden. Werden diese Parameter ungünstig gewählt, liefert das neuronale Netz schlechte Ergebnisse.
- Das Extrapolationsverhalten ist bei einer Optimierung robust, da das Kriging-Verfahren in diesem Fall einen globalen Erwartungswert mit hoher Varianz liefert.
- Kriging ist ein BLUE (Best Linear Unbiased Estimator) Schätzer [Plackett, 1950]. Also ein linearer, erwartungstreuer Schätzer minimaler Varianz. Im nächsten Abschnitt wird dies genauer erläutert.
- Viele nichtstatistische Verfahren, wie z.B. die Inverse Distanzwichtung [Shepard, 1968], beachten eine lokale Häufung von Stützstellen nicht. Ein statistisches Verfahren wie Kriging hingegen berücksichtigt die gesamte räumliche Verteilung der Stützstellen [Krüger, 2012] und gewichtet lokal gehäufte Stützstellen geringer.
- Die Kriging-Verfahren sind sehr vielfältig. Beispielsweise ist die Nutzung von Gradienteninformationen (Gradient Enhanced Kriging) oder verschiedener Güteklassen (CO-Kriging) möglich.

4.2 Kriging: Analytische Herleitung

Dieses Kapitel bietet eine kompakte Herleitung der hier verwendeten Kriging-Grundgleichungen und setzt einige grundlegende Kenntnisse im Bereich der multivariaten Statistik voraus. Einen guten Einstieg in diese Thematik bieten die Arbeiten von [Lophaven et al., 2002] und [MacKay, 1998].

4.2.1 Allgemeine Vorbemerkungen und Definitionen

Angenommen es existiert ein Programm (bspw. ein Strömungslöser), welches in s verschiedenen Gütestufen unterteilt werden kann und zu einem Parametersatz $\vec{x} \in \mathbb{R}^k$ jeweils einen Output für jede Gütestufe $y_1(\vec{x}), ..., y_s(\vec{x})$ berechnet.

Da alle Krigingmodelle statistische Interpolationsverfahren darstellen, werden die Stützstellen der einzelnen Gütestufen $y_i(\vec{x}_j), i \in \{1, ..., s\}, j \in \{1, ..., n_i\}$ als Realisierung eines stationären Zufallsprozesses $Z_i(\vec{x}), i \in \{1, ..., s\}$ angesehen. Der Krige-Schätzer für die Gütestufe $g \in \{1, ..., s\}$ wird mit $Z_g^*(\vec{x})$ bezeichnet. Desweiteren sollen die Zufallsprozesse der einzelnen Gütestufen einen konstanten Erwartungswert $E[Z_i] = const$ besitzen.

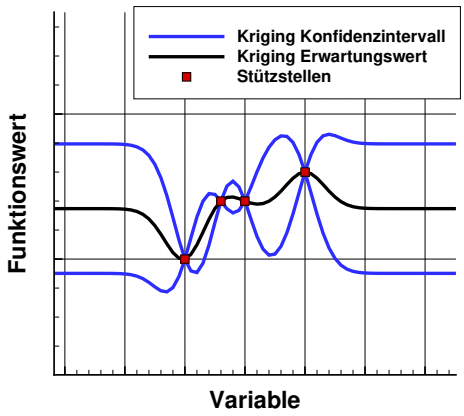


Abbildung 4.1: Kriging Erwartungswert- und Varianz-Vorhersage für eine eindimensionale Funktion mit einer Gütestufe. Die Varianz ist als 99% (3σ) Konfidenzintervall eingezeichnet.

Verwendung der Kriging-Verfahren in einer Optimierung

Für die Nutzung des Kriging-Verfahrens innerhalb einer Optimierung (siehe Kapitel 2.3.2 und Kapitel 3.2) muss das Kriging-Verfahren drei verschiedene Schätzungen für eine unbekannte Stelle $\vec{x} \in \mathbb{R}^{\mathrm{k}}$ liefern, diese werden folgend vorgestellt und dienen als Leitfaden für die darauffolgenden Herleitungen:

1. Den geschätzten bedingten Erwartungswert $Z_g^*(\vec{x})$ der Gütestufe g, welcher durch Gleichung 4.14 berechnet wird. In Abschnitt 4.2.2 ist die analytische Herleitung dieser Gleichungen zu finden.
2. Die Unsicherheit der Vorhersage in Form einer Varianz: $var\left[Z_g(\vec{x}) - Z_g^*(\vec{x})\right]$, welche durch Gleichung 4.15 berechnet wird. In Abschnitt 4.2.2 ist die analytische Herleitung dieser Gleichungen zu finden.
3. Die Kovarianzen $cov\left(Z_{g_1}^*(\vec{x_1}), Z_{g_2}^*(\vec{x_2})\right) \big| \vec{y_i}, \vec{w_i}$, folgend $cov\left(Z_{g_1}^*(\vec{x_1}), Z_{g_2}^*(\vec{x_2})\right)$ zwischen den Vorhersagen der unbeprobten Orte $\vec{x_1}, \vec{x_2} \in \mathbb{R}^{\mathrm{k}}$ unter der Bedingung aller bekannten Daten $\vec{y}$ und Gewichte $\vec{w}$ der Gütestufen $g_1, g_2 \in \{1, ..., s\}$, welche durch Gleichung 4.17 berechnet wird. In Abschnitt 4.2.2 ist die analytische Herleitung dieser Gleichungen zu finden.

Abbildung 4.1 zeigt die exemplarische Vorhersage eines trainierten Kriging-Modells mit einer Gütestufe ($g = 1$) und vier Stützstellen. In schwarz eingezeichnet ist der Erwartungswert $Z_g^*(\vec{x})$ und in blau die vorhergesagte Varianz $var\left[Z_g(\vec{x}) - Z_g^*(\vec{x})\right]$ dargestellt als 3σ Konfidenzintervall, wobei $\sigma = \sqrt{var\left[Z_g(\vec{x}) - Z_g^*(\vec{x})\right]}$). Das Beispiel dient dazu, das grundlegende Verhalten des Kriging-Verfahren zu erläutern:

- Das Verfahren besitzt den globalen Erwartungswert β_g, wobei für jede Gütestufe g ein eigener Erwartungswert vorhanden ist. Nur in der Nähe von bekannten Stützstellen ändert sich dieser Erwartungswert. Die Bestimmung dieses globalen Erwartungswerts wird in Kapitel 5.3.2 beschrieben.
- Das Verfahren besitzt eine globale Varianz $\mathrm{var}[Z_g]$ für jede Gütestufe g, welche sich in der Nähe von bekannten Stützstellen verringert. Diese wird über das Trainingsverfahren bestimmt, siehe Kapitel 5.3.2.
- Das Abstandsmaß oder die „Wirkweite“ von Stützstellen wird während des Trainings eingestellt, siehe Kapitel 5.1. Das Auffinden dieser „Wirkweiten“ ist die Hauptaufgabe des Trainingsverfahrens und wird in Kapitel 5.3 beschrieben.
- Innerhalb einer Trainingsiteration muss jeweils eine Matrix der Größe $n_{all} \times n_{all}$ (n_{all} steht für die gesamte Anzahl an Stützstellen oder partiellen Ableitungen, siehe Kapitel 4.2.2) invertiert werden. Diese Operation besitzt eine Komplexität von $\mathcal{O}(n^3)$, was das Verfahren numerisch sehr aufwändig macht. Aus diesem Grund wurden im Rahmen dieser Arbeit zahlreiche Methoden entwickelt, um diesen Prozess zu beschleunigen, siehe Kapitel 5.3 und 5.5.
- Im Normalfall werden die Stützstellen interpoliert und die vorhergesagte Unsicherheit ist an diesen Stellen nahe Null. Das Kriging-Verfahren bietet zudem aber auch die Möglichkeit die Stützstellen zu approximieren (siehe Kapitel 5.2) und ist somit sehr flexibel einsetzbar.

Im Folgenden sollen diese drei Schätzer hergeleitet werden. Die Herleitung soll dabei so allgemein gehalten werden, dass alle drei Schätzer für das Co-, Ordinary- und Gradient-Enhanced-Kriging ohne Änderungen anwendbar sind.

4.2.2 Herleitung Krige-Schätzer

Der in diesem Kapitel vorgestellte Kriging Ansatz ist angelehnt an die Arbeiten von [Z.-H. Han, R. Zimmermann, 2010], [Han et al., 2012] ,[Kennedy and O'Hagan, 2000] und [Krüger, 2012]. Die Unterschiede des hier vorgestellten Ansatzes zu den genannten Arbeiten werden in Kapitel 4.4 erläutert.

Das Kriging-Verfahren soll an einer unbekannten Stelle $\vec{x} \in \mathbb{R}^{\mathrm{k}}$ eine Schätzung $y_i^*(\vec{x})$ über den Funktionswert $y_i(\vec{x})$ einer bestimmten Gütestufe $i \in \{1, ..., s\}$ vorhersagen.

Die grundlegende Annahme dabei ist, dass der zu schätzende Wert $y_i^*(\vec{x})$ durch eine gewichtete Summe aus den bekannten Stützstellen $y_i(\vec{x}_j)$ und den noch zu bestimmenden Gewichten $w_{i,j}(g,\vec{x}), i \in \{1, ..., s\}, j \in \{1, ..., n_i\}$ gebildet werden kann, wobei die Gewichte von der vorherzusagenden Gütestufe g abhängen. n_i beschreibt die Anzahl der Stützstellen für die jeweilige Gütestufe. Der Einfachheit halber werden die Stützstellen wie auch die Gewichte zukünftig durch die folgenden Vektoren beschrieben

$$\vec{y}_i = (y_i(\vec{x}_1), ..., y_i(\vec{x}_{n_i}))^T, i \in \{1, ..., s\} \tag{4.1}$$

und

$$\vec{w}_i(g,\vec{x}) = (w_{i,1}(g,\vec{x}), ..., w_{i,n_i}(g,\vec{x}))^T, i \in \{1, ..., s\} \tag{4.2}$$

Da die Stützstellen $y_i(\vec{x}_j)$ einer Gütestufe i als Realisierung des jeweiligen stationären stochastischen Prozesses $Z_i(\vec{x}), i \in \{1, ..., s\}$ angesehen werden, stammen diese aus dem Vektor $\vec{Z}_i = (Z_i(\vec{x}_1), ..., Z_i(\vec{x}_{n_i}))^T, i \in \{1, ..., s\}$. Der Krige-Schätzer $Z_g^*(\vec{x})$ für die Gütestufe g wird damit zu:

$$Z_g^*(\vec{x}) = \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i(g,\vec{x}) \tag{4.3}$$

In diesem Abschnitt sollen die gesuchten Gewichte bestimmt werden. F sei der Schätzfehler, $Z_g(\vec{x})$ sei der reale stochastische Prozess der vorherzusagenden Gütestufe g.

$$F_g(\vec{x}) = Z_g(\vec{x}) - Z_g^*(\vec{x}) = Z_g(\vec{x}) - \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i(g,\vec{x}) \tag{4.4}$$

Kriging ist ein linearer erwartungstreuer Schätzer, was bedeutet, dass der Erwartungswert des Schätzfehlers 0 ist.

$$E[F_g(\vec{x})] = 0 \tag{4.5}$$

$$\begin{aligned}
&\Leftrightarrow E[Z_g(\vec{x})] - E\left[Z_g^*(\vec{x})\right] = 0 \\
&\Leftrightarrow E[Z_g(\vec{x})] - E\left[\sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i(g,\vec{x})\right] = 0 \\
&\Leftrightarrow E[Z_g(\vec{x})] - E\left[\sum_{i=1}^{s} \sum_{j=1}^{n_i} Z_i(\vec{x}_j) w_{i,j}(g,\vec{x})\right] = 0
\end{aligned}$$

$$\Leftrightarrow E\left[Z_g\left(\vec{x}\right)\right] - \sum_{i=1}^{s}\sum_{j=1}^{n_i} E\left[Z_i(\vec{x}_j)\right] w_{i,j}(g,\vec{x}) = 0 \tag{4.6}$$

Aufgrund der Stationarität soll gelten $E\left[Z_i(\vec{x}_j)\right] = E\left[Z_i(\vec{x}_l)\right] = \beta_i, \forall j,l \in \{1,...,n_i\}$ und $E\left[Z_g\left(\vec{x}\right)\right] = \beta_g$, somit folgt aus Gleichung 4.6:

$$\sum_{i=1}^{s}\beta_i\sum_{j=1}^{n_i} w_{i,j}(g,\vec{x}) = \beta_g \tag{4.7}$$

Eine weitere Bedingung an das Kriging-Verfahren ist die minimale Varianz des Schätzfehlers. Folgende Gleichung ergibt sich aus der Definition der Varianz:

$$\mathrm{var}\left[F_g\left(\vec{x}\right)\right] = \mathrm{var}\left[Z_g(\vec{x}) - Z_g^*(\vec{x})\right] = E\left[\left(\left(Z_g\left(\vec{x}\right) - Z_g^*\left(\vec{x}\right)\right) - E\left[Z_g\left(\vec{x}\right) - Z_g^*\left(\vec{x}\right)\right]\right)^2\right] \tag{4.8}$$

Diese Gleichung lässt sich unter Berücksichtigung der Annahme 4.7 umformen zu (Herleitung siehe Anhang A.2.6):

$$\mathrm{var}\left[F_g\left(\vec{x}\right)\right] = \mathrm{var}\left[Z_g\left(\vec{x}\right)\right] - 2\overrightarrow{c}(g,\vec{x})^T\vec{w}(g,\vec{x}) + \vec{w}(g,\vec{x})^T\mathbf{Cov}\vec{w}(g,\vec{x}) \tag{4.9}$$

Wobei $\vec{w} \in \mathbb{R}^{\mathrm{n_{all}}}$ den Gewichtsvektor, $\mathbf{Cov} \in \mathbb{R}^{\mathrm{n_{all}}\times\mathrm{n_{all}}}$ die Kovarianzmatrix (Bildungsvorschrift in Kapitel 5.1 erläutert) und $\overrightarrow{c}(g,\vec{x}) \in \mathbb{R}^{\mathrm{n_{all}}}$; $\overrightarrow{c}(g,\vec{x}) = \left(cov\left(Z_i(\vec{x}_j), Z_g\left(\vec{x}\right)\right), ..., cov\left(Z_s(\vec{x}_{n_s}), Z_g\left(\vec{x}\right)\right)\right)^T, i \in \{1,...,s\}, j \in \{1,...,n_i\}$ den Kovarianzvektor zwischen $Z_i(\vec{x}_j), i \in \{1,...,s\}, j \in \{1,...,n_i\}$ und $Z\left(\vec{x}\right)$ darstellt. Die Variable n_{all} beschreibt die Anzahl aller Stützstellen der verschiedenen Gütestufen.

Damit lässt sich folgende Minimierungsaufgabe mit Nebenbedingung stellen:

$$\begin{cases} \min_{w_1,...,w_n}\left(\mathrm{var}[Z_g\left(\vec{x}\right)] - 2\overrightarrow{c}(g,\vec{x})^T\vec{w}(g,\vec{x}) + \vec{w}(g,\vec{x})^T\mathbf{Cov}\vec{w}(g,\vec{x})\right) \\ \sum_{i=1}^{s}\beta_i\sum_{j=1}^{n_i} w_{i,j}(g,\vec{x}) - \beta_k = 0 \end{cases} \tag{4.10}$$

Diese Minimierungsaufgabe aus Gleichung 4.10 kann gemäß der Multiplikatorenmethode von Lagrange gelöst werden [Bronstejn and Semendjajew, 2008]. Im Minimum müssen die partiellen Ableitungen der Lagrange Funktion $\Lambda\left(w,\lambda\right)$ mit dem Lagrange Multiplikator λ Null werden:

$$\Lambda\left(w,\lambda\right) := \mathrm{var}\left[F_g\left(\vec{x}\right)\right] + \lambda\left(\sum_{i=1}^{s}\beta_i\sum_{j=1}^{n_i} w_{i,j}(g,\vec{x}) - \beta_g\right)$$

$$\nabla_{\lambda,w}\left(\mathrm{var}\left[F_g\left(\vec{x}\right)\right] + \lambda\left(\sum_{i=1}^{s}\beta_i\sum_{j=1}^{n_i} w_{i,j}(g,\vec{x}) - \beta_g\right)\right) = \vec{0}$$

Aus diesem Gleichungssystem lässt sich nun die Lösung für die gesuchten Gewichte bestimmen. Die vollständige Herleitung kann in Anhang A.2.4 gefunden werden.

$$\vec{w}(g,\vec{x}) = \mathbf{Cov}^{-1}\vec{c}(g,\vec{x}) - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c}(g,\vec{x}) + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_g \tag{4.11}$$

Der Vektor $\vec{F} \in \mathbb{R}^{\mathrm{n_{all}}}$ ist wie folgt definiert:

$$\vec{F} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_1 \\ \vdots \\ \beta_s \\ \vdots \\ \beta_s \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} n_1\, Einträge \\ \vdots \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n_s\, Einträge \end{matrix} \tag{4.12}$$

Die Schätzung der Einträge $\beta_1 ... \beta_s$ wird in Kapitel 5.3.2 beschrieben.

4.2.3 Erwartungswert

Setzt man die Gewichte aus Gleichung 4.11 in Gleichung 4.3 ein, erhält man folgende Formulierung, wobei die Vektoren die Gewichte und Stützstellen aller Gütestufen enthalten:

$$E\left[Z_g^*(\vec{x})\right] = \vec{c}(g,\vec{x})^T \mathbf{Cov}^{-1}\vec{y} - \vec{c}(g,\vec{x})^T \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{y} + \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{y}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_g$$

Durch einsetzen von $\mu := \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{y}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}$, ergibt sich:

$$E\left[Z_g^*(\vec{x})\right] = \vec{c}(g,\vec{x})^T \mathbf{Cov}^{-1}\left(\vec{y} - \mu\vec{F}\right) + \mu\beta_g \tag{4.13}$$

Verwendet man den Likelihood-Schätzer aus Kapitel 5.3.2 so ergibt sich:

$$\mu = 1$$

Der entsprechende Beweis kann in Anhang A.3.6 gefunden werden. Daraus ergibt sich dann die letztendliche Formel für den Krige-Schätzer:

$$E\left[Z_g^*(\vec{x})\right] = \vec{c}(g,\vec{x})^T \mathbf{Cov}^{-1}\left(\vec{y} - \vec{F}\right) + \beta_g \tag{4.14}$$

4.2.4 Varianz des Schätzfehlers

Setzt man die Gewichte aus Gleichung 4.11 in Gleichung 4.9 ein, so erhält man die Formel für die Vorhersage der Varianz des Schätzfehlers:

$$\mathrm{var}\left[F_g(\vec{x})\right] = \mathrm{var}\left[Z_g(\vec{x})\right] - \vec{c}(g,\vec{x})^T\mathbf{Cov}^{-1}\vec{c}(g,\vec{x}) + \frac{\left(\vec{c}^T\mathbf{Cov}^{-1}\vec{F} - \beta_g\right)^2}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \tag{4.15}$$

Die vollständige Herleitung ist in Anhang A.2.9 zu finden.

4.2.5 Kovarianz zwischen zwei Vorhersagen

Oftmals wird auch die Kovarianz zwischen zwei unbekannten Orten $\vec{x_1}, \vec{x_2}$ unter Berücksichtigung der bereits bekannten Stützstellen benötigt. Dies entspricht der Kovarianz zwischen den zwei Vorhersagen $Z^*_{g_1}(\vec{x_1}), Z^*_{g_2}(\vec{x_2})$, wobei $g_1, g_2 \in \{1, ..., s\}$ die jeweiligen vorherzusagenden Gütestufen angeben, wobei die Herleitung in Anhang A.2.7 zu finden ist:

$$\begin{aligned} cov\left(Z^*_{g_1}(\vec{x_1}), Z^*_{g_2}(\vec{x_2})\right) &= cov\left(\sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_{1i}(g_1, \vec{x_1}), \sum_{j=1}^{s} \vec{Z}_j^T \vec{w}_{2j}(g_2, \vec{x_2})\right) \\ &= \sum_{i=1}^{s}\sum_{j=1}^{s}\sum_{l=1}^{n_i}\sum_{m=1}^{n_i} w_{1i,l}(g_1, \vec{x_1}) w_{2j,m}(g_2, \vec{x_2}) cov\left(Z_i(\vec{x_l}), Z_j(\vec{x_m})\right) \\ &= \vec{w}_1^T(g_1, \vec{x_1}) \mathbf{Cov} \vec{w}_2(g_2, \vec{x_2}) \end{aligned} \tag{4.16}$$

Durch das Einsetzen der Gleichung 4.11 in Gleichung 4.16 ergibt sich (Beweis siehe Anhang A.2.8):

$$\begin{aligned} cov\left(Z^*_{g_1}(\vec{x_1}), Z^*_{g_2}(\vec{x_2})\right) =& \vec{c_1}(g_1, \vec{x_1})^T \mathbf{Cov}^{-1} \vec{c_2}(g_2, \vec{x_2}) - \\ & \frac{\vec{c_1}(g_1, \vec{x_1})^T \mathbf{Cov}^{-1} \vec{F} \vec{F}^T \mathbf{Cov}^{-1} \vec{c_2}(g_2, \vec{x_2})}{\vec{F}^T \mathbf{Cov}^{-1} \vec{F}} + \frac{\beta_g^2}{\vec{F}^T \mathbf{Cov}^{-1} \vec{F}} \end{aligned} \tag{4.17}$$

4.3 Umsetzung der Kriging-Unterverfahren

In diesem Abschnitt werden die notwendigen Schritte zum Umsetzen der einzelnen Kriging-Unterverfahren (Co-, Ordinary- und Gradient-Enhanced-Kriging) beschrieben.

4.3.1 Co-Kriging

Der vorgestellte CO-Kriging-Ansatz basiert in den Grundzügen auf den Arbeiten von Kennedy und Forrester [Kennedy and O'Hagan, 2000],[Forrester et al., 2007]. Der erste Schritt besteht darin, die notwendigen Kovarianzfunktionen zu definieren (siehe Abschnitte 4.2.3,4.2.4 und 4.2.5) und dann eine geeignete Modellierung für diese zu finden. Geht man von den stochastischen Prozessen Z_i mit den unterschiedlichen Gütestufen $i \in \{1, ..., s\}$ aus, wobei die höhere Gütestufe mit dem niedrigeren Index gekennzeichnet ist, so sind für die vollständige Beschreibung der Kovarianzmatrix an den Stellen $\vec{x}$ und $\vec{x}'$ folgende Kovarianzfunktionen zu modellieren:

$$\mathrm{cov}\Big(Z_i(\vec{x}), Z_i(\vec{x}')\Big) \tag{4.18}$$

$$\mathrm{cov}\Big(Z_{i+1}(\vec{x}), Z_i(\vec{x}')\Big) \tag{4.19}$$

$$\mathrm{cov}\Big(Z_{i+1}(\vec{x}), Z_{i+1}(\vec{x}')\Big) \tag{4.20}$$

Weiterhin sollen zwei Gütestufen $s = 2$ betrachtet werden. Ein häufig gewählter Ansatz für ein Kovarianzmodell ist der Folgende:

$$Z_1(\vec{x}) = aZ_2(\vec{x}) + Z_{diff}(\vec{x}) \tag{4.21}$$

Der Prozess hoher Güte Z_1 wird also aus einem skalierten Prozess niedrigerer Güte Z_2 und einem Differenzprozess Z_{diff} modelliert. Weiterhin wird angenommen, dass der Differenzprozess $Z_{diff}(\vec{x})$ und der Prozess $Z_2(\vec{x})$ unkorreliert sind, also gilt:

$$cov\left(Z_2(\vec{x}), Z_{diff}(\vec{x}')\right) = cov\left(Z_{diff}(\vec{x}), Z_2(\vec{x}')\right) = 0 \tag{4.22}$$

Setzt man Gleichung 4.21 in die Gleichungen 4.18 ein, so ergibt sich (vollständige Herleitung siehe Anhang A.2.10):

$$cov\left(Z_1(\vec{x}), Z_1(\vec{x}')\right) = a^2 cov\left(Z_2(\vec{x}), Z_2(\vec{x}')\right) + cov\left(Z_{diff}(\vec{x}), Z_{diff}(\vec{x}')\right) \tag{4.23}$$

$$cov\left(Z_1(\vec{x}), Z_2(\vec{x}')\right) = a\,cov\left(Z_2(\vec{x}), Z_2(\vec{x}')\right) \tag{4.24}$$

$$cov\left(Z_2(\vec{x}), Z_2(\vec{x}')\right) = cov\left(Z_2(\vec{x}), Z_2(\vec{x}')\right) \tag{4.25}$$

So sind innerhalb der Software nur noch zwei Kovarianzfunktionen $cov\left(Z_2(\vec{x}), Z_2(\vec{x}')\right), cov\left(Z_{diff}(\vec{x}), Z_{diff}(\vec{x}')\right)$ zu modellieren. Die konkrete Modellierung und Verwendung dieser Kovarianzfunktionen innerhalb der Kriging-Verfahren wird in Kapitel 5.1 behandelt.

4.3.2 Ordinary-Kriging

Das Ordinary-Kriging kann als Untermenge des CO-Krigings angesehen werden. Der größte Unterschied besteht darin, dass nur von einer möglichen Gütestufe $s = 1$ ausgegangen wird. Ansonsten behalten alle Gleichungen aus Kapitel 4.2 ihre Gültigkeit und können vollständig verwendet werden. Das in Kapitel 4.3.1 beschriebene Kovarianzmodell vereinfacht sich jedoch sehr stark. Es wird nur noch eine parametrisierte Kovarianzmodellfunktion $cov\left(Z(\vec{x}), Z(\vec{x}')\right)$ für das gesamte Modell benötigt und der Skalierungsfaktor wird für diesen Fall konstant auf eins gesetzt: $a = 1$.

4.3.3 Gradient-Enhanced-Kriging

Das Gradient-Enhanced-Kriging (GEK) ist eine Erweiterung des Ordinary-Kriging. Beim GEK gehen partielle Ableitungen $\frac{\partial y(x)}{\partial x}$ an beprobten Orten mit in die Bildung des Modells ein, wobei ein großer Vorteil des Verfahrens ist, dass nicht alle partiellen Ableitungen benötigt werden. Der Trainingsvektor $\vec{y} \in \mathbb{R}^{n+m}$ kann in diesem Fall also die folgende Form annehmen:

$\vec{y} = \left(y(\vec{x}_1), ..., y(\vec{x}_n), \frac{\partial y(\vec{x}_{n+1})}{\partial x^j}, ..., \frac{\partial y(\vec{x}_{n+m})}{\partial x^j}\right), j \in \{1, ..., k\}$, wobei der obere Index von x die Variable darstellt, nach der abgeleitet wird und m die Anzahl der gegebenen partiellen Ableitungen. Es müssen also auch Kovarianzen zwischen den partiellen Ableitungen und den Funktionswerten gebildet werden können. Um im Kriging Modell

partielle Ableitungen verarbeiten zu können, ist es also notwendig, dass die Kovarianzfunktionen in der folgenden Form gebildet werden:

$$\operatorname{cov}(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}, \frac{\partial Z\left(\vec{x}_2\right)}{\partial x_2^l}), p, l \in \{1, ..., k\} \tag{4.26}$$

$$\operatorname{cov}(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}, Z\left(\vec{x}_2\right)), p \in \{1, ..., k\} \tag{4.27}$$

$$\operatorname{cov}(Z\left(\vec{x}_1\right), \frac{\partial Z\left(\vec{x}_2\right)}{\partial x_2^l}), l \in \{1, ..., k\} \tag{4.28}$$

Um die in Kapitel 5.1 vorgestellten Kovarianzmodelle weiterhin nutzen zu können, muss die Kovarianzfunktion umformuliert werden. Per Definition der Kovarianz ergibt sich für Gleichung 4.27:

$$\operatorname{cov}(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}, Z\left(\vec{x}_2\right)) = E\left[\left(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p} - E\left[\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}\right]\right)\left(Z\left(\vec{x}_2\right) - E\left[Z\left(\vec{x}_2\right)\right]\right)\right] \tag{4.29}$$

Dasselbe Ergebnis lässt sich auch durch äußeres Differenzieren erreichen:

$$\begin{aligned}\frac{\partial}{\partial x_1^p}\operatorname{cov}(Z\left(\vec{x}_1\right), Z\left(\vec{x}_2\right)) =& \frac{\partial}{\partial x_1^p}\left(E\left[\left(Z\left(\vec{x}_1\right) - E\left[Z\left(\vec{x}_1\right)\right]\right)\left(Z\left(\vec{x}_2\right) - E\left[Z\left(\vec{x}_2\right)\right]\right)\right]\right) \\ =& E\left[\left(Z\left(\vec{x}_2\right) - E\left[Z\left(\vec{x}_2\right)\right]\right)\frac{\partial}{\partial x_1^p}\left(Z\left(\vec{x}_1\right) - E\left[Z\left(\vec{x}_1\right)\right]\right)\right] \\ &+ E\left[\left(Z\left(\vec{x}_1\right) - E\left[Z\left(\vec{x}_1\right)\right]\right)\frac{\partial}{\partial x_1^p}\left(Z\left(\vec{x}_2\right) - E\left[Z\left(\vec{x}_2\right)\right]\right)\right]\end{aligned}$$

Es gilt $\frac{\partial}{\partial x_1^p}\left(Z\left(x_2\right) - E\left[Z\left(x_2\right)\right]\right) = 0$, da der Ausdruck unabhängig von x_1 ist

$$= E\left[\left(Z\left(\vec{x}_2\right) - E\left[Z\left(\vec{x}_2\right)\right]\right)\left(\frac{\partial}{\partial x_1^p} Z\left(\vec{x}_1\right) - E\left[\frac{\partial}{\partial x_1^p} Z\left(\vec{x}_1\right)\right]\right)\right]$$

Dieser Ausdruck entspricht Gleichung 4.29, also gilt:

$$\operatorname{cov}(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}, Z\left(\vec{x}_2\right)) = \frac{\partial}{\partial x_1^p}\operatorname{cov}(Z\left(\vec{x}_1\right), Z\left(\vec{x}_2\right)) \tag{4.30}$$

für die Gleichungen 4.26 und 4.28 gilt entsprechendes:

$$\begin{aligned}\operatorname{cov}(\frac{\partial Z\left(\vec{x}_1\right)}{\partial x_1^p}, \frac{\partial Z\left(\vec{x}_2\right)}{\partial x_2^l}) =& \frac{\partial}{\partial x_1^p \partial x_2^l}\operatorname{cov}(Z\left(\vec{x}_1\right), Z\left(\vec{x}_2\right)) \\ \operatorname{cov}(Z\left(\vec{x}_1\right), \frac{\partial Z\left(\vec{x}_2\right)}{\partial x_2^l}) =& \frac{\partial}{\partial x_2^l}\operatorname{cov}(Z\left(\vec{x}_1\right), Z\left(\vec{x}_2\right))\end{aligned}$$

Mit diesen Gleichungen ist es nun möglich, die Modelle aus Kapitel 5.1 zu verwenden, indem man diese differenziert. Dies stellt eine elegante und praktisch umsetzbare Lösung des Problems dar. Die Gleichungen aus Kapitel 4.3 behalten ansonsten vollständig ihre Gültigkeit, allerdings muss der Erwartungswertvektor $\vec{F} \in \mathbb{R}^{\mathrm{n+m}}$ angepasst werden. Dieser wird genauso aufgebaut wie der Stützstellenvektor $\vec{y}$, wobei an jeder

entprechenden Stelle einer partiellen Ableitung eine Null eingetragen werden muss. Damit ergibt sich der Erwartungswertvektor für das GEK-Verfahren zu:

$$\vec{F} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_1 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} n\,Einträge \\ \vdots \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} m\,Einträge \end{matrix} \tag{4.31}$$

Die effiziente Verwendung dieses Verfahrens innerhalb von Turbmaschinenoptimierungen stellt ein aktuelles Forschungsthema dar. Um diese gewinnbringend einsetzen zu können, ist eine effiziente Erzeugung der partiellen Ableitungen notwendig. Für weitere Informationen sei der Leser auf [Backhaus et al., 2012, Backhaus et al., 2017, Peter and Dwight, 2010] und [Krüger, 2012, Han et al., 2009] verwiesen.

4.4 Vergleich des hier vorgestellten Co-Kriging-Verfahrens mit anderen Verfahren aus der Literatur

Innerhalb dieses Abschnittes werden die Gemeinsamkeiten und Unterschiede zu anderen Verfahren aus der Literatur herausgestellt. Die Kovarianzmodelle (siehe Kapitel 4.3.1), der Aufbau der Kovarianzmatrix $\mathbf{Cov} \in \mathbb{R}^{\mathrm{n_{all}} \times n_{all}}$ und des Erwartungswertvektors $\vec{F}$ (siehe Gleichung 4.31) sowie des Stützstellenvektors $\vec{y}_s$ ähneln im Wesentlichen den Formulierungen aus der Arbeit von [Kennedy and O'Hagan, 2000]. Die Unterschiede in den Verfahren können durch folgende Punkte zusammengefasst werden:

1. In dem Modell von [Kennedy and O'Hagan, 2000] existiert eine Unabhängigkeitsbedingung (vgl. [Kennedy and O'Hagan, 2000] Gleichung 8). Diese besagt, dass die Hyperparameter (siehe Kapitel 5.1.2.1) $\vec{\theta}_2, \sigma_2^2$ des Modells niedriger Güte unabhängig von den Hyperparametern $a, \vec{\theta}_1, \sigma_1^2$ geschätzt werden können. Das hat zufolge, dass das Modell niedrigerer Güte im ersten Schritt alleine trainiert werden kann und darauf aufbauend das Differenzmodell trainiert wird. Dies bietet numerische Vorteile, da hierfür nur die jeweiligen Submatrizen der gesamten Kovarianzmatrix invertiert werden müssen und auch der jeweilige Hyperparamterraum kleiner ist. Dieser Ansatz bedingt allerdings, dass an jedem bekannten Punkt hoher Güte auch eine Stützstelle niedriger Güte bekannt sein muss. Aufbauend auf der Arbeit von [Kennedy and O'Hagan, 2000] schlägt [Forrester et al., 2007] ein weiteres Modell vor. In diesem wird ein Ordinary-Kriging-Modell mit den Daten niedrigerer Güte aufgestellt. Die fehlenden Punkte niedriger Güte an Stellen hoher Güte werden dann aus den Vorhersagen des Ordinary-Kriging Modells niedriger Güte approximiert. Die verwendeten Vorhersagen sind allerdings mit einer Unsicherheit behaftet, stellen also selbst Verteilungen dar. Inwiefern dieser Umstand beachtet werden sollte, wird in der Arbeit von [Forrester et al., 2007] nicht erwähnt.

2. In dem hier entwickelten Modell werden die Hyperparameter-Sätze vollständig auf einmal trainiert und sind damit nicht mehr unabhängig, siehe Kapitel 5.3. Das hat folgende Auswirkungen:
   (a) Die Punktverteilung der Gütestufen ist unabhängig voneinander.
   (b) Der Hyperparameterraum ist deutlich größer und das Modell aus [Kennedy and O'Hagan, 2000] ist somit eine Untermenge von dem in dieser Arbeit beschriebenen.
   (c) Innerhalb dieser Arbeit wurden beide Strategien anhand von analytischen Funktionen getestet. Es wurden jeweils dieselben Trainingsdatensätze verwendet. Dabei konnte beobachtet werden, dass durch den größeren Parameterraum ein besserer Likelihood-Term erzielt werden konnte als mit dem in [Kennedy and O'Hagan, 2000] beschriebenen Modell.
   (d) Die Trainingszeiten steigen an, da die gesamte Kovarianzmatrix Choleskyzerlegt werden muss und der zu trainierende Hyperparameterraum deutlich größer ist. In den hier betrachteten Turbomaschinen-Anwendungsfällen sind die Prozesskettenzeiten allerdings so dominant, dass die Trainingszeit keine wesentliche Rolle spielt (siehe Kapitel 6).

Eine weitere Herangehensweise kann in [Han et al., 2012] gefunden werden. In dieser Arbeit wird ein Co-Kriging-Modell mit drei unabhängigen Kovarianzfunktionen beschrieben. Dies bedeutet, dass drei verschiedene Hyperparametersätze gleichzeitig trainiert werden und entspricht damit den Gleichungen (4.18-4.20). Dieses Vorgehen führt zu erheblichen numerischen Problemen. Der Grund liegt darin, dass die positive Definitheit der Kovarianzmatrix nicht mehr gesichert ist. Um dieses Problem zu umgehen, wurde in [Han et al., 2009] vorgeschlagen, für alle drei Kovarianzfunktionen denselben Hyperparametersatz zu verwenden. Verglichen mit einem Ordinary-Kriging, welches die Stützstellen aller Gütestufen beinhaltet, besteht der Unterschied dann nur in einer eigenen globalen Varianz und einem eigenen globalen Erwartungswert für die verschiedenen Gütestufen. Damit könnte nur noch eine für das gesamte Modell gültige konstante Verschiebung zwischen den Gütestufen dargestellt werden. Dies stellt jedoch eine erhebliche Vereinfachung des Co-Kriging-Modells dar, welche für die meisten Anwendungen nicht ausreichend ist.

5 Implementierung der Verfahren

Innerhalb dieser Arbeit wurde eine Kriging-Software entwickelt und die Implementierung des hier vorgestellten Multifidelity-Verfahrens vorgenommen. Im ersten Teil dieses Kapitels werden die verwendete Modellierung der Kovarianzfunktion, die Behandlung von verrauschten Funktionen und eine mögliche Regularisierung der Kovarianzmatrix beschrieben. Darauf folgend wird das Training der Krigingmodelle mithilfe verschiedener Minimierungsverfahren erläutert, wobei sich das RPROP-Minimierungsverfahren (siehe [Riedmiller and Braun, 1993, Helbig and Scherer, 2011]) als besonders geeignet für die Problemstellung herausgestellt hat. Das RPROP-Verfahren stammt in seiner ursprünglichen Form aus dem Bereich des Neuronalen-Netzwerk Trainings und wurde im Rahmen dieser Arbeit auf die Bedürfnisse eines Co-Kriging-Modells angepasst und weiterentwickelt. Die Ergebnisse werden innerhalb dieses Kapitels vorgestellt.

Da die Verwendung eines Kriging-Verfahrens innerhalb einer größeren Optimierung numerisch sehr aufwendig ist, wurden zahlreiche Methoden entwickelt und umgesetzt, welche die numerische Effizienz steigern. Diese Methoden werden in den Abschnitten 5.3,5.4 und 5.5 vorgestellt.

Eine in dieser Arbeit entwickelte Analysesoftware, mit der es möglich ist, während einer laufenden Optimierungen die Ersatzmodelle auf Plausibilität und Effizienz zu überprüfen und evtl. Fehler aufzudecken wird in Kapitel 5.6 vorgestellt.

5.1 Modellierung der Kovarianzfunktion

In Kapitel 4 wurden die verschiedenen Kriging-Verfahren beschrieben. Innerhalb dieses Kapitels werden die Begriffe der Kovarianzmatrix und des Kovarianzvektors verwendet. Um diese aufstellen zu können, ist eine Kovarianzfunktion in der Form $cov\left(Z^*_{g_1}\left(\vec{x_1}\right), Z^*_{g_2}\left(\vec{x_2}\right)\right)$ nötig, wobei $\vec{x}_1, \vec{x}_2$ die Orte von bekannten Stützstellen beschreiben und $g_1, g_2 \in \{1, ..., s\}$ die jeweiligen Gütestufen. Da in der praktischen Anwendung der reale Zufallsprozess $Z(\vec{x})$ nicht bekannt ist und damit auch nicht die Kovarianzfunktion, bedient man sich parametrisierter Kovarianzmodelle in der Form $cov_{g_1,g_2}\left(\vec{x}_1, \vec{x_2}\right)$.

Diese Funktionen müssen jedem möglichen Punktepaar eine Kovarianz zuordnen. Weiterhin muss die daraus resultierende Matrix positiv definit und symmetrisch sein, was die Wahl einer geeigneten Funktion erschwert. Das folgende Beispiel zeigt den Aufbau einer beispielhaften Kovarianzmatrix $\mathbf{Cov} \in \mathbb{R}^{\mathrm{n_{all}} \times \mathrm{n_{all}}}$ für vier verschiedene

Samples $y_g(\vec{x})$ der jeweiligen Gütestufe g:

$$\mathbf{Cov} = \left[\begin{array}{c|cccc} & y_1(\vec{x}_1) & y_1(\vec{x}_2) & y_2(\vec{x}_3) & y_2(\vec{x}_4) \\ \hline y_1(\vec{x}_1) & cov_{1,1}(\vec{x}_1,\vec{x}_1) & cov_{1,1}(\vec{x}_1,\vec{x}_2) & cov_{1,2}(\vec{x}_1,\vec{x}_3) & cov_{1,2}(\vec{x}_1,\vec{x}_4) \\ y_1(\vec{x}_2) & cov_{1,1}(\vec{x}_2,\vec{x}_1) & cov_{1,1}(\vec{x}_2,\vec{x}_2) & cov_{1,2}(\vec{x}_2,\vec{x}_3) & cov_{1,2}(\vec{x}_2,\vec{x}_4) \\ y_2(\vec{x}_3) & cov_{2,1}(\vec{x}_3,\vec{x}_1) & cov_{2,1}(\vec{x}_3,\vec{x}_2) & cov_{2,2}(\vec{x}_3,\vec{x}_3) & cov_{2,2}(\vec{x}_3,\vec{x}_4) \\ y_2(\vec{x}_4) & cov_{2,1}(\vec{x}_4,\vec{x}_1) & cov_{2,1}(\vec{x}_4,\vec{x}_2) & cov_{2,2}(\vec{x}_4,\vec{x}_3) & cov_{2,2}(\vec{x}_4,\vec{x}_4) \end{array}\right] \tag{5.1}$$

Um ein solches Modell praktisch nutzbar zu machen, müssen drei verschiedene Problemstellungen behandelt werden:

1. Die Definition der Ortsabhängigkeit der Kovarianzfunktion: Beispielsweise kann die Kovarianz nur abhängig vom Radius (radiale Basisfunktionen) sein oder nur durch den Verbindungsvektor $\triangle\vec{x} = \vec{x_1} - \vec{x_2}$ beschrieben werden. Es gibt auch Modelle, welche eine Abhängigkeit von der absoluten Lage im Raum ermöglichen. Diese benötigen allerdings eine erhebliche Anzahl an ersatzmodellspezifischen Parametern (siehe [Schmidt and O'Hagan, 2003]).
2. Es wird ein parametrisiertes Modell für die Kovarianzfunktion benötigt, welches zu einer positiv definiten und symmetrischen Matrix führt.
3. Die darin verwendeten Parameter müssen in einem Trainingsverfahren kalibriert werden können.

Diese Problemstellungen werden in den folgenden zwei Abschnitten behandelt.

5.1.1 Ortsabhängigkeit Kovarianzfunktion

In diesem Abschnitt wird der Begriff des Variogramms und des Kovariogramms erklärt (siehe [Matheron, 1963]). Diese stellen die Basis für die Kovarianzmodellfunktionen dar und sind somit zentraler Bestandteil des Kriging-Verfahrens.

Sei $Z(\vec{x})$ ein räumlicher Zufallsprozess, so ist das Variogramm $2\gamma(\vec{x}_1,\vec{x}_2)$, $\vec{x}_1,\vec{x}_2 \in \mathbb{R}^k$ definiert über:

$$\begin{aligned} 2\gamma(\vec{x}_1,\vec{x}_2) &:= var\left[Z(\vec{x}_1) - Z(\vec{x}_2)\right] \\ &= E\left[\left(\left(Z(\vec{x}_1) - Z(\vec{x}_2)\right) - E\left[Z(\vec{x}_1) - Z(\vec{x}_2)\right]\right)^2\right] \\ &= E\left[\left(\left(Z(\vec{x}_1) - E\left[Z(\vec{x}_1)\right]\right) - \left(Z(\vec{x}_2) - E\left[Z(\vec{x}_2)\right]\right)\right)^2\right] \end{aligned} \tag{5.2}$$

$\gamma(\vec{x}_1,\vec{x}_2)$ wird als Semivariogramm bezeichnet. Weiterhin wird das Kovariogramm wie folgt definiert:

$$cov(\vec{x}_1,\vec{x}_2) := \frac{1}{2}var\left[Z(\vec{x_1})\right] + \frac{1}{2}var\left[Z(\vec{x_2})\right] - \gamma(\vec{x}_1,\vec{x}_2) \tag{5.3}$$

Einfacher formuliert beschreibt das Variogramm und auch das Kovariogramm die räumliche Abhängigkeit eines Punktes zu Nachbarpunkten. Es wird die Annahme getroffen, dass dieser räumliche Zufallsprozess schwach stationär ist. Die Definition eines räumlich schwach stationären Zufallsprozesses ist gegeben durch (vgl.

[Cressie, 1993, Özkaya, 2014, Akin and Siemes, 2013]):

$$E\left[Z\left(\vec{x}\right)\right] = \mu, \forall \vec{x} \in \mathbb{R}^{\mathrm{k}} \tag{5.4}$$

$$cov(\vec{\triangle x}) := cov(\vec{x}, \vec{x} + \triangle\vec{x}) \tag{5.5}$$

Die Kovarianz ist also nur noch abhängig von dem Verschiebevektor $\triangle\vec{x} \in \mathbb{R}^{\mathrm{k}}; \triangle\vec{x} = \vec{x}_1 - \vec{x}_2$. Unter Berücksichtigung von Gleichung 5.4 und Gleichung 5.5 folgt dann für die gesuchte Kovarianzfunktion aus Gleichung 5.3 (siehe Anhang A.3.3):

$$cov(\triangle\vec{x}) = cov(\vec{0}) - 2\gamma(\triangle\vec{x}) = \sigma^2 - 2\gamma(\triangle\vec{x}) \tag{5.6}$$

$cov(\vec{0}) = \sigma^2$ entspricht der stationären Varianz des Prozesses (siehe Kapitel 5.3). Die Modellierung von $\gamma(\triangle\vec{x})$ wird in Abschnitt 5.1.2 behandelt.

5.1.2 Parametrisches Kovarianzmodell

Der folgende Abschnitt erläutert die parametrisierten Kovarianzmodelle, welche für die Nutzung eines Kriging-Verfahrens notwendig sind. Die Basis für ein solches Modell ist durch Gleichung 5.6 gegeben und in Abbildung 5.1 grafisch dargestellt. Der Einfachheit halber beschränkt sich das gezeigte Kovariogramm auf eine räumliche Dimension.

Auf der X-Achse ist die räumliche Distanz und auf der Y-Achse die Kovarianz $cov(\triangle\vec{x})$ aufgetragen, diese nimmt mit steigender Distanz ab.

Empirisch ist oftmals zu beobachten, dass das Kovariogramm bei $\triangle\vec{x} = 0$ springt. Dieser Wert $\lambda \in \mathbb{R}$ wird als „Nugget“ bezeichnet und entspricht einem ortsunabhängigen Rauschen. Um dies zu modellieren, wird eine Sprungfunktion δ definiert:

$$\delta(\triangle\vec{x}) = \begin{cases} 1 & falls\ \triangle\vec{x} = \vec{0} \\ 0 & sonst \end{cases}$$

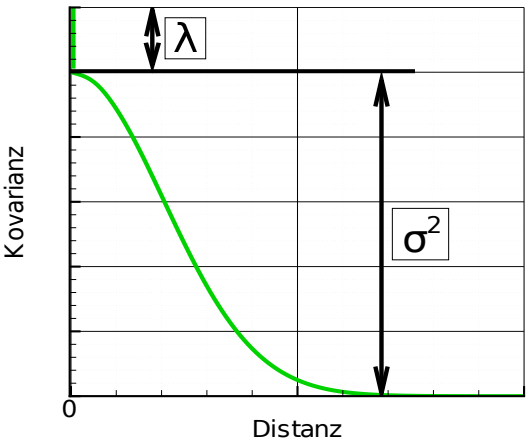


Abbildung 5.1: Beispielhaftes Kovariogramm

Die Summe $\sigma^2 + \lambda$ (auch „Sill“ genannt) ist gleichzusetzen mit der Gesamtvarianz des stochastischen Prozesses selbst und der eingezeichnete Wert σ^2 wird als „partial Sill“ bezeichnet. Damit lässt sich Gleichung 5.6 umformulieren zu:

$$cov(\triangle\vec{x}) = \sigma^2 + \lambda\delta(\triangle\vec{x}) - 2\gamma(\triangle\vec{x}) \tag{5.7}$$

Die Bestimmung der Varianz σ^2 und des Nuggets λ innerhalb des Kriging-Modells werden mithilfe eines Trainingsverfahrens geschätzt das in Kapitel 5.3 beschrieben wird.

Für die Variogrammfunktion $\gamma\left(\triangle\vec{x}\right)$ muss allerdings noch ein geeignetes Modell gefunden werden. Eine Möglichkeit besteht in der Erzeugung eines empirischen Variogramms (siehe [Cressie, 1993]). Dieses Verfahren liefert für die Abstände bekannter

Stützstellen den entsprechenden räumlichen Zusammenhang und ist als nicht parametrisches Verfahren einzuordnen. Es wird aber eine sehr große Anzahl an Stützstellen benötigt, um plausible Variogramme zu erhalten. Auf der anderen Seite werden keine Annahmen über den Verlauf des Variogramms benötigt.

Typischerweise wird auf parametrisierte Modellfunktionen zurückgegriffen, welche an die vorhandenen Stützstellen bestmöglich angepasst werden und mit weniger Stützstellen auskommen als eine empirische Schätzung, dafür sind sie allerdings eingeschränkter was die Form der Funktion angeht. Weiterhin müssen diese Funktionen positiv semidefinit sein (vgl. [Cressie, 1993, Krüger, 2012, Schmid and Feilke, 2012]), was die Auswahl dieser einschränkt. Ein besonders häufig verwendetes Modell ist das Folgende [Lophaven et al., 2002, Sacks et al., 2007]:

$$2\gamma(\triangle\vec{x}) = \sigma^2 \left(1 - e^{-\frac{1}{2}\sum_{l=1}^{l<=k}(\theta_l|\triangle x_l|^p)} \right) \tag{5.8}$$

$\vec{\theta} \in \mathbb{R}^{\mathrm{k}}$ wird als Hyperparametervektor bezeichnet und jeder Eintrag θ_l stellt eine Skalierung für den entsprechenden Parameter l dar. Die Wahl eines sinnvollen Hyperparametervektors ist Aufgabe des Trainingsverfahrens (siehe Kapitel 5.3). In dem hier implementierten Kriging-Verfahren wird eine abgeänderte Modellfunktion verwendet:

$$2\gamma(\triangle\vec{x}) = \sigma^2 \left(1 - e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^p\right)} \right) \tag{5.9}$$

Die Benutzung der Exponentialfunktion e^{θ_l} anstelle von θ_l dient der numerischen Stabilität des Trainingsverfahrens, weil die Terme $e^{\theta_l}\,|\triangle x_l|^p$ in Gleichung 5.9 nicht negativ werden dürfen und somit für Formulierung 5.8 eine Nebenbedingung in der Form $\theta_l > 0 \forall l \in \{1, ..., k\}$ notwendig wäre. Durch die Verwendung der Exponentialfunktion wird dies vermieden.

Für den Fall $p = 2$ spricht man auch von einer Gauss'schen Kovarianzfunktion, welche in der hier beschriebenen Software als Standard gewählt wird. Der Vorteil dieser Korrelationsfunktion liegt in der schnellen Berechnung und damit in der schnellen Erzeugung der benötigten Kovarianzmatrix.

Neben diesen Korrelationsfunktionen wird häufig ein parametrisierter kubischer Spline verwendet (siehe [Lophaven et al., 2002]). Im Rahmen dieser Arbeit sind diese drei Korrelationsfunktionen implementiert und frei wählbar, wobei ein besonderes Augenmerk auf Effizienz gelegt wurde. Die genaue softwaretechnische Umsetzung wird in Kapitel 5.5.3 beschrieben.

Setzt man die Kovarianzfunktion aus Gleichung 5.8 in Gleichung 5.7 ein, dann folgt daraus die resultierende Kovarianzfunktion:

$$cov(\triangle\vec{x})) = \sigma^2 + \lambda\delta(\triangle\vec{x})) - \sigma^2 \left(1 - e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^2\right)} \right) \tag{5.10}$$

$$= \lambda\delta(\triangle\vec{x})) + \sigma^2 e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^2\right)} \tag{5.11}$$

Die für das Trainingsverfahren notwendigen Ableitungen sehen für die gauss'sche Kovarianzfunktion ($p = 2$) wie folgt aus:

$$\frac{\partial cov(\triangle\vec{x}))}{\partial\theta_a} = -\frac{1}{2}\sigma^2 e^{\theta_a} |\triangle x_a|^2 \left(e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^2\right)} \right) \tag{5.12}$$

$$\frac{\partial cov(\triangle\vec{x})}{\partial\sigma^2} = e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^2\right)} \tag{5.13}$$

Die restlichen Gleichungen aller umgesetzten Modellfunktionen sowie deren Ableitungen nach Hyperparametern und Ort sind in Anhang A.2.3, A.2.2 und A.2.1 zu finden.

5.1.2.1 Kovarianzfunktion Co-Kriging

Die Kovarianzfunktion für das Co-Kriging wird aus mehreren Kovarianzfunktionen zusammengesetzt, daher werden keine neuen Modelle benötigt. Es wird allerdings für jede Gütestufe ein zusätzlicher Hyperparametervektor $\vec{\theta}_2, \vec{\theta}_{diff}$ verwendet. Geht man von den Gleichungen 4.25 aus und setzt das entsprechende Kovarianzmodell aus Gleichung 5.10 ein, wobei $c_2(\triangle\vec{x}) = e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_{l,2}}|\triangle x_l|^2\right)}$ und $c_{diff}(\triangle\vec{x}) = e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_{l,diff}}|\triangle x_l|^2\right)}$, so erhält man:

$$\begin{aligned} cov_{1,1}(\vec{x}_1, \vec{x}_2) =& a^2\sigma_2^2 c_2(\triangle\vec{x}) + \lambda_{diff}\delta(\triangle\vec{x}) + \sigma_{diff}^2 c_{diff}(\triangle\vec{x}) \\ cov_{1,2}(\vec{x}_1, \vec{x}_2) =& a\sigma_2^2 c_2(\triangle\vec{x}) \\ cov_{2,2}(\vec{x}_1, \vec{x}_2) =& \lambda_2\delta(\triangle\vec{x}) + \sigma_2^2 c_2(\triangle\vec{x}) \end{aligned} \tag{5.14}$$

5.1.2.2 Aufteilung der Hyperparameter in Klassen

Die in den Kapiteln 4, 5.1 und 5.2 beschriebenen Hyperparameter lassen sich in Kategorien aufteilen, welche in Tabelle 5.1 aufgelistet sind.

Diese Einteilung ist für die Initialisierung sehr wichtig, da jede dieser Gruppen anders initialisiert werden muss. Für die Diagonalaufschläge λ wird das in Kapitel 5.2.1 beschriebene Verfahren verwendet. Die Initialisierung der anderen Gruppen wird in Kapitel 5.3.3 beschrieben.

	Zeichen	Anzahl
Korrelationsl.	$\vec{\theta}$	s
Varianzen	σ^2	s
Diagonalauf.	λ	s
Skalierung	a	$s-1$

Tabelle 5.1: Kategorien von Hyperparametern bei s Gütestufen

Im Falle eines Co-Krigings skaliert die Anzahl der Hyperparameter $o = sk + 3s$ mit der Anzahl der Gütestufen s und der Anzahl an freien Variablen k. Die Anzahl der Hyperparameter o ist also grundsätzlich deutlich höher als bei einem Ordinary-Kriging.

5.2 Regularisierung und Behandlung verrauschter Funktionen

Für die Anwendung der verschiedenen Kriging-Verfahren sind die Berücksichtigungen von schlecht konditionierten Kovarianzmatrizen und der Umgang mit verrauschten Funktionen von großer Bedeutung. Da die praktische Behandlung beider Problematiken sehr ähnlich ist, werden sie innerhalb dieses Kapitels gemeinsam beschrieben.

5.2.1 Regularisierung

Während des Trainings wird ein Satz Hyperparameter gesucht, welcher den Likelihood-Wert (siehe Kapitel 5.3.1) maximiert. Dabei muss für jeden Satz an Hyperparametern die zugehörige Kovarianzmatrix aufgestellt werden. Bei diesem Prozess kann es passieren, dass die resultierende Matrix schlecht konditioniert ist und damit die verwendeten numerischen Verfahren instabil werden. Die Ursache einer schlecht konditionierten Matrix können sehr vielfältig sein und werden in [Davis and Morris, 1997] genauer beschrieben. Meist lassen sich die Ursachen auf schlecht verteilte Stützstellen oder auf eine ungünstige Initialisierung der Hyperparameter zurückführen. Auch die Wahl der Kovarianzfunktion kann enormen Einfluss auf die numerische Stabilität haben.

Eine gebräuchliche Methode zur Verbesserung der Konditionszahl ist die Erhöhung der Hauptdiagonalelemente der Kovarianzmatrix um einen Wert $\lambda \in \mathbb{R}$ (folgend Diagonalaufschlag). Dieses Vorgehen entspricht einer Tikhonov-Regularisierung, wie sie häufig für Least-Squares-Verfahren eingesetzt wird (siehe [J. Forrester et al., 2006, Tikhonov et al., 2013, Hoerl and Kennard, 2000]).

Der Diagonalaufschlag λ wurde bei Herleitung von Gleichung 5.10 bereits berücksichtigt. Bezüglich der Regularisierung der Matrix sollte er jedoch so klein wie möglich gehalten werden, da dessen Größe einen starken Einfluss auf die Vorhersagen hat.

Ordinary-Kriging: Zur Bestimmung eines geeigneten Wertes ist es sinnvoll, die Konditionszahl κ der Kovarianzmatrix zu betrachten. Diese ist definiert als Quotient aus maximalem Ξ_{max} und minimalem Eigenwert Ξ_{min}:

$$\kappa = \left| \frac{\Xi_{max}\left(\mathbf{Cov}\right)}{\Xi_{min}\left(\mathbf{Cov}\right)} \right|$$

Geht man weiterhin von einer Ordinary-Kriging-Kovarianzmatrix $\mathbf{Cov} \in \mathbb{R}^{n \times n} mit\, Cov_{i,j}(\triangle\vec{x}) = \lambda\delta(\triangle\vec{x}) + \sigma^2 c_{i,j}(\triangle\vec{x})$ aus, so kann (im Falle $\lambda = 0$) die Varianz σ^2 aus der Matrix ausgeklammert werden und die Korrelationsmatrix $\mathbf{R} \in \mathbb{R}^{n \times n} mit\, R_{i,j}(\triangle\vec{x}) = c_{i,j}(\triangle\vec{x})$ bleibt:

$$\mathbf{Cov} = \sigma^2 \mathbf{R}$$

Die Werte innerhalb der Korrelationsmatrix $\mathbf{R} \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}}$ können (im Falle $\lambda = 0$) einen maximalen Wert von 1 annehmen (siehe Gleichung 5.11). Daraus lässt sich der ungünstigste Fall $\widetilde{\mathbf{R}} \in \mathbb{R}^{n \times n}$ für die Konditionierung der Matrix ableiten, welcher einer

Einsmatrix entspricht.

$$\mathbf{Cov} = \sigma^2 \begin{bmatrix} 1 & ... & 1 \\ ... & ... & ... \\ 1 & ... & 1 \end{bmatrix} = \sigma^{\mathbf{2}}\widetilde{\mathbf{R}} \tag{5.15}$$

Die minimalen und maximalen Eigenwerte sind für diesen Fall bekannt:

$$\Xi_{min}\left(\mathbf{Cov}\right) = 0 \tag{5.16}$$
$$\Xi_{max}\left(\mathbf{Cov}\right) = \sigma^2 n \tag{5.17}$$

Dies würde einer unendlichen Konditionszahl entsprechen $\kappa = \infty$. Addiert man nun den Diagonalaufschlag, bekommt man folgende Kovarianzmatrix:

$$\mathbf{Cov} = \begin{bmatrix} \sigma^2 + \lambda & ... & \sigma^2 \\ ... & ... & ... \\ \sigma^2 & ... & \sigma^2 + \lambda \end{bmatrix} \tag{5.18}$$

Daraus ergeben sich die folgenden maximalen und minimalen Eigenwerte:

$$\Xi_{min}\left(\mathbf{Cov}\right) = \lambda \tag{5.19}$$
$$\Xi_{max}\left(\mathbf{Cov}\right) = \lambda + \sigma^2 n \tag{5.20}$$

Und die entsprechende Konditionszahl verbessert sich zu:

$$\kappa = \left| \frac{\lambda + \sigma^2 n}{\lambda} \right| \tag{5.21}$$

Da die Matrix positiv definit sein muss und damit nur positive Eigenwerte hat, kann der Betrag vernachlässigt werden. Wählt man nun für die Konditionszahl eine obere Grenze, erhält man eine Untergrenze für den Diagonalaufschlag:

$$\kappa_{max} > \frac{\lambda + \sigma^2 n}{\lambda} \tag{5.22}$$
$$\lambda > \frac{\sigma^2 n}{(\kappa_{max} - 1)} \tag{5.23}$$

Co-Kriging: Im Falle des Co-Krigings ist aufgrund der unterschiedlichen Gütestufen die Kovarianzmatrix partitioniert. Der Ansatz aus dem Ordinary-Kriging ist mit dieser partitionierten Matrix nicht mehr möglich, dennoch kann man den maximalen Eigenwert $\Xi_{max}\left(\mathbf{Cov}\right)$ der Matrix schätzen. Für eine diagonalisierbare Matrix gilt allgemein, dass die Summe der Eigenwerte $\Xi_i, i \in \{1, ..., n_{all}\}$ der Spur der Matrix entspricht:

$$\sum_{i=1}^{n_{all}} \Xi_i = spur\left(\mathbf{Cov}\right) \tag{5.24}$$

Geht man von den Gleichungen 5.14 aus, so ergibt sich die Spur als Summe der

Diagonaleinträge wie folgt:

$$spur\,(\mathbf{Cov}) = \sum_{i=1}^{n_1}\left(a^2\sigma_2^2 c_2(\vec{0}) + \sigma_{diff}^2 c_{diff}(\vec{0}) + \lambda_{diff}\right) + \sum_{i=1}^{n_2}\left(\sigma_2^2 c_2(\vec{0}) + \lambda_2\right) \tag{5.25}$$

Da die Korrelationen der Diagonalen den Wert Eins besitzen,$c_2(\vec{0}) = 1$ und $c_{diff}(\vec{0}) = 1$, gilt für die Spur der Matrix:

$$\begin{aligned}\sum_{i=1}^{n_{all}} \Xi_i = spur\,(\mathbf{Cov}) &= \sum_{i=1}^{n_1}\left(a^2\sigma_2^2 + \sigma_{diff}^2 + \lambda_{diff}\right) + \sum_{i=1}^{n_2}\left(\sigma_2^2 + \lambda_2\right) && (5.26)\\ &= n_1\left(a^2\sigma_2^2 + \sigma_{diff}^2 + \lambda_{diff}\right) + n_2\left(\sigma_2^2 + \lambda_2\right) && (5.27)\\ &\geq n_1\lambda_{diff} + n_2\lambda_2 && (5.28)\\ &\geq (n_1 + n_2)\min(\lambda_{diff}, \lambda_2) =: (n_1 + n_2)\lambda_{min} && (5.29)\end{aligned}$$

Geht man weiterhin von nur einem Regularisierungsterm $\lambda_{diff} = \lambda_2 = \lambda$ für alle Gütestufen aus, gilt für den maximalen und minimalen Eigenwert Ξ_{max}, Ξ_{min}:

$$\begin{aligned}\Xi_{max} &\leq n_1\left(a^2\sigma_2^2 + \sigma_{diff}^2 + \lambda\right) + n_2\left(\sigma_2^2 + \lambda\right) && (5.30)\\ \Xi_{min} &\geq \lambda; falls\,\Xi_i = \Xi_{min} \forall i && (5.31)\end{aligned}$$

Für die Konditionszahl κ lässt sich so eine obere Grenze bestimmen:

$$\kappa \leq \frac{n_1\left(a^2\sigma_2^2 + \sigma_{diff}^2 + \lambda\right) + n_2\left(\sigma_2^2 + \lambda\right)}{\lambda} \tag{5.32}$$

Legt man nun für die Konditionszahl einen maximalen Wert κ_{max} fest, ergibt sich als Schätzung für den Diagonalaufschlag:

$$\lambda_{min} \geq \frac{n_1 a^2\sigma_2^2 + n_1\sigma_{diff}^2 + n_2\sigma_2^2}{(\kappa_{max} - (n_2 + n_1))} \tag{5.33}$$

Die Werte $\sigma_2^2, \sigma_{diff}^2, a$ werden während des Trainings oftmals variiert, dadurch ändert sich auch die untere Grenze des Diagonalaufschlags λ_{min} permanent. Für dieses Problem sind zwei Lösungen möglich:

1. Die untere Grenze für λ_{min} in jedem Trainingsschritt neu bestimmen und bei Unterschreitung auf den Grenzwert setzen.
2. Die untere Grenze für λ_{min} nur beim Start des Trainings bestimmen.

Die erste Lösung kann dazu führen, dass sich der Grenzwert λ_{min} während des Trainings ändert und der aktuelle Wert $\lambda < \lambda_{min}$ die Grenze unterschreitet. Da der Diagonalaufschlag danach bei Unterschreitung der Grenze abhängig von den Variablen $\sigma_2^2, \sigma_{diff}^2, a$ wird, muss dies bei der Ableitung des Likelihoods (siehe Kapitel 5.3.1) Berücksichtigung finden, was einen hohen Aufwand bedeutet.

Aus diesem Grund wird hier die zweite Variante gewählt, also die untere Grenze initial eingestellt und nur dann verändert, wenn numerische Probleme während des Trainings auftreten. Ein geeigneter Wert für die obere Grenze der Konditionszahl liegt erfah-

rungsgemäß bei $\sim 10^9$. Dieser Wert kann im Einzelfall natürlich angepasst werden.

5.2.2 Behandlung verrauschter Funktionen

Bei Vernachlässigung des Diagonalaufschlags λ ist das beschriebene Kriging-Verfahren rein interpolierend. In einigen Fällen ist allerdings ein approximierendes Verhalten gewünscht. Ein solches Verhalten entspricht dem in Kapitel 5.1.1 beschriebenen „Nugget-Effekt“. Die praktische Umsetzung innerhalb des Kriging-Verfahrens ist letztlich dieselbe wie bei dem in Kapitel 5.2.1 beschriebenen Regularisierungsterm. Der hauptsächliche Unterschied besteht in der Größenordnung des Diagonalaufschlags λ und in dessen Bedeutung, da es sich bei dem Diagonalaufschlag in diesem Fall um eine Varianz handelt, die einem zufälligen Rauschen entspricht (siehe Kapitel 5.1.2). Deswegen wird der Diagonalaufschlag als Hyperparameter mit Untergrenze behandelt und muss mit trainiert werden. Die Umsetzung soll folgend gezeigt werden.

Um den Rauschterm innerhalb eines gradientenbasierten Trainingsverfahrens (siehe Kapitel 5.3) zu verwenden, müssen die Ableitungen der Kovarianzfunktionen nach den Rauschtermen gebildet werden. Diese sind in Anhang A.3.1 zu finden. Mit diesen Ableitungen ist das Maximum-Likelihood-Verfahren (siehe Kapitel 5.3.1 und Kapitel 5.3) direkt anwendbar.

Rauschterm Beispiel In Abbildung 5.2 wird ein einfaches Beispiel gezeigt, welches die Wirkung von unterschiedlichen Rauschtermen auf die Vorhersage darstellt. Es handelt sich um eine konstante Null-Funktion mit einem normalverteilten Rauschen, welche den Erwartungswert Null und die Standardabweichung Eins besitzt: $f(x) = \mathcal{N}(0, 1)$. Jeder eingezeichnete Punkt entspricht einer Stützstelle. Mithilfe dieser Stützstellen wird ein Ordinary-Kriging trainiert und dann für den entsprechenden Wertebereich Vorhersagen getroffen. Abbildung 5.4 zeigt zwei mögliche Vorhersagen, beide werden durch den Likelihood-Term gleichermaßen gut bewertet.

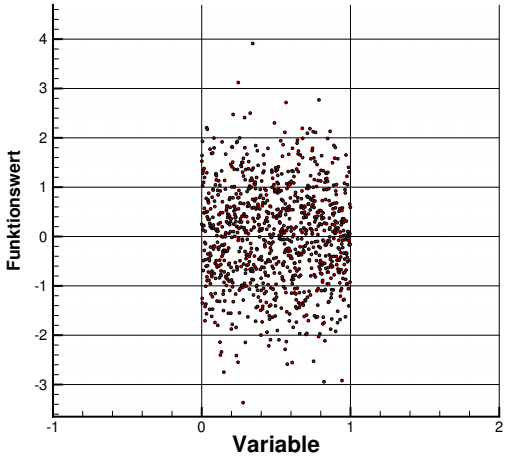


Abbildung 5.2: Konstante Funktion mit normalverteiltem Rauschen

Die schwarze Linie entspricht dem vorhergesagten Erwartungswert und die Fehlerbalken der vorhergesagten Standardabweichung. Auf dem linken Bild zeigt das trainierte Kriging ein approximierendes und auf dem rechten Bild ein interpolierendes Verhalten. Auf dem rechten Bild wird jeder der Stützstellen exakt widergegeben und die vorhersagte Standardabweichung ist an diesen Stellen Null. Der Diagonalaufschlag ist bei dieser Lösung ebenfalls Null.

Auf dem linken Bild hat der Diagonalaufschlag den Wert Eins, was genau der Varianz des Rauschterms entspricht. Im Extrapolationsbereich zeigen die verschiedenen

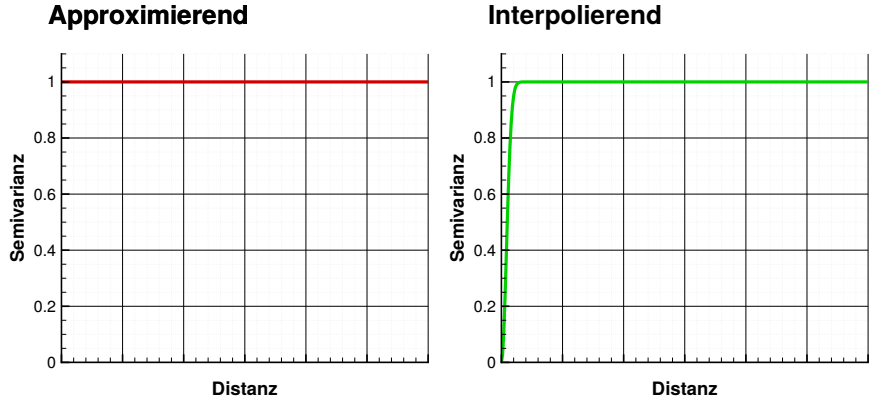


Abbildung 5.3: Qualitatives Variogramm für das interpolierende und approximierende Kriging

Vorhersagen exakt dieselben Ergebnisse. Diese Doppeldeutigkeit kann auf die Einstellungen der Hyperparameter zurückgeführt werden. Die wesentlichen Hyperparameter sind die Prozessvarianz σ^2, der Diagonalaufschlag λ und die Korrelationslänge θ. Tabelle 5.2 zeigt die Einstellungen der beiden Lösungen. Für das Training der beiden Krigingmodelle wurde der Diagonalaufschlag λ vorher festgelegt und die anderen Größen durch das Training automatisch bestimmt.

Die interpolierende Variante besitzt eine Varianz von $\sigma^2 = 1$ und entspricht damit dem in Kapitel 5.1.1 beschriebenen „Sill“. Also der Varianz bei großem Abstand zu einem bekannten Punkt. Der Abstand wird durch die Korrelationslänge θ beeinflusst. Wobei ein großer θ Wert auf einen kleinen räumlichen Einfluss deutet. Diese Lösung würde also einem sehr steilen Variogramm entsprechen, welches bei kleinstem Abstand sofort auf den „Sill“ springt. Das Variogramm wird qualitativ in dem rechten Diagramm der Abbildung 5.3 dargestellt.

	Interpolierend	Approximierend
σ^2	1	0
λ	0	1
θ	e^{15}	e^{-50}

Tabelle 5.2: Einstellungen der verschiedenen Kriging-Lösungen

Die approximierende Variante hat einen sehr kleinen Skalierungsfaktor $\theta = e^{-50}$ (siehe Gleichung 5.10) und eine Varianz von $\sigma^2 = 0$, was mit einer konstanten Funktion ohne Unsicherheit korrespondiert. Durch den Rauschterm $\lambda = 1$ wird dem Modell eine grundlegende Unsicherheit aufaddiert, welche dem „Nugget“ entspricht. Das dazugehörige Variogramm wäre unabhängig von der Distanz und entspricht dem linken Diagramm in Abbildung 5.3.

Die beiden gezeigten Lösungen haben denselben Likelihood-Wert und werden somit vom Trainingsverfahren als gleichwertig angesehen. Welche Lösung vom Training letztlich gewählt wird, hängt hauptsächlich von der gewählten Initialisierung ab.

Abbildung 5.5 zeigt für verschiedene Diagonalaufschläge den resultierenden Likelihood-Wert, wobei ein kleiner Likelihood ein besseres Resultat darstellt. Die Abbildung zeigt, dass der Likelihood-Wert für einem Diagonalaufschlag im Bereich von $\lambda > 0 \wedge \lambda < 1$ konstant und minimal bleibt. Darüber wird der Likelihood-Wert dann deutlich größer. Um das Verhalten zu verstehen, wird in Abbildung 5.5 die eingestellte Varianz σ^2 und der Diagonalaufschlag für den Bereich des optimalen Likelihood-Wertes gezeigt. Das Ergebnis ist auch hier plausibel, denn in der Summe ergeben die

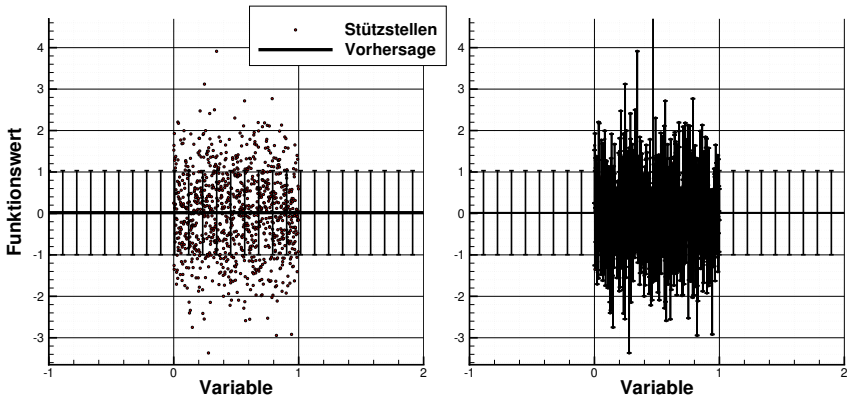


Abbildung 5.4: Normalverteiltes Rauschen und zwei mögliche Kriging Modelle. Die Fehlerbalken stellen die vorhergesagte Standardabweichung dar

beiden Werte immer Eins $\sigma^2 + \lambda = 1$, was dann wiederum der Varianz des Erzeugungsprozesses entspricht. Das hier gezeigte Verhalten entspricht also den Erwartungen. Allerdings kann dies bei einer geringeren Datenlage oder einer komplexeren Funktion dazu führen, dass der Diagonalaufschlag deutlich überschätzt wird. Aus diesem Grund sollte das Training des Diagonalaufschlags als optionale Funktion des Trainings verfügbar sein.

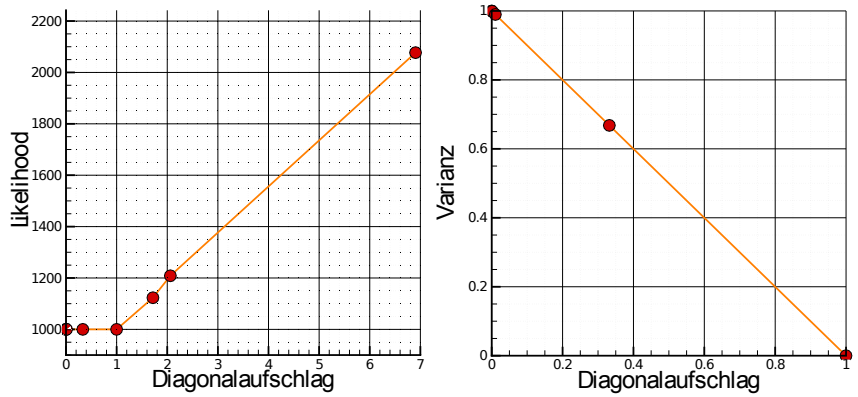


Abbildung 5.5: (a) Diagonalaufschlag, Varianz und resultierender Likelihood Wert (b) Bereich des optimalen Likelihood-Wertes und Entwicklung des Diagonalaufschlag mit Varianz

Fazit: In den Kapiteln 5.2.1 und 5.2.2 wird ein Diagonalaufschlag verwendet, um eine Regularisierung der Matrix zu erreichen und auch um ein approximierendes Verhalten herbeizuführen. Praktische Unterschiede ergeben sich nur in der Größe der Werte und in der Bestimmung während des Trainings. Auf der einen Seite steht der Regularisierungsterm, welcher für die numerische Stabilität sorgt und die Vorhersage möglichst nicht beeinflussen soll. Deswegen muss dieser sehr klein gewählt werden. Zudem ist dieser nicht als Hyperparameter anzusehen und daher keine zu trainierende Größe. Auf der anderen Seite steht der Rauschterm, welcher für ein approximierendes Verhalten sorgen soll. Dieser Parameter wird auch innerhalb des Trainingsverfahrens geschätzt und ist als Hyperparameter einzustufen. Um nicht für den Regularisierungsterm und den Rauschterm jeweils einen eigenen Wert zu verwenden, ist es möglich, den Regularisierungsterm als minimale Grenze anzunehmen. Der Rauschterm kann dann trainiert werden, besitzt in diesem Fall allerdings eine untere Schranke, die sich wie in Kapitel 5.2.1 beschrieben festlegen lässt.

5.3 Training

In diesem Kapitel wird das Trainingsverfahren für alle Krigingmodelle beschrieben. Dafür wird im ersten Teil das Maximum-Likelihood-Verfahren vorgestellt. Dieses liefert ein Maß für die Güte des Modells unter Berücksichtigung der eingestellten Hyperparameter.

Ziel des Trainingsverfahrens ist es, die Hyperparameter der Kovarianz-Modellfunktion (siehe Kapitel 5.1) zu finden, die diesen Term maximieren. Zu diesem Zweck werden diverse numerische Minimierungsverfahren und auch mögliche Initialisierungsverfahren vorgestellt.

5.3.1 Maximum-Likelihood für alle Kriging-Verfahren

Die Maximum-Likelihood-Methode ist ein Schätzverfahren, um Parameter einer angenommenen Verteilungsfunktion bei gegebenen Realisierungen zu schätzen. Es werden Werte für Hyperparameter gewählt, gemäß derer die Realisierung der bereits bekannten Daten am plausibelsten erscheint.

Um diesen Ansatz bei einem Verfahren wie dem Kriging anzuwenden, werden die bekannten Funktionswerte als Realisierung einer multivariaten Normalverteilung $\mathcal{N}\left(\vec{F}, \mathbf{Cov}(\tilde{\mathrm{h}})\right)$ angenommen, wobei $\mathbf{Cov}(\tilde{\mathrm{h}}) \in \mathbb{R}^{\mathrm{n_{all}} \times \mathrm{n_{all}}}$ die jeweilige Kovarianzmatrix in Abhängigkeit der Hyperparameter $\vec{h} \in \mathbb{R}^o$ darstellt und $\vec{Z} = (\vec{Z}_0, ..., \vec{Z}_g, ...\vec{Z}_{mf})^T$ den Vektor mit einem Zufallsprozess $\vec{Z}_g = \left(Z_{g,1}, ..., Z_{g,n_g}\right)^T$ für jede Gütestufe g mit jeweils n_g Realisierungen. Analog dazu beinhaltet der Vektor $\vec{F}$ alle Erwartungswerte der Zufallsprozesse $\vec{Z}$ (siehe Gleichung 4.12). Der Vektor $\vec{y}_s \in \mathbb{R}^{n_{all}}$ beinhaltet alle bekannten Stützstellen. Daraus ergibt sich die folgende Dichtefunktion L der multivariaten Normalverteilung $\mathcal{N}\left(\vec{F}, \mathbf{Cov}(\tilde{\mathrm{h}})\right)$:

$$L(\vec{Z}, \vec{F}(\vec{h}), \mathbf{Cov}(\tilde{\mathrm{h}})) = \frac{1}{(2\pi)^{\frac{n}{2}} \det\left(\mathbf{Cov}(\tilde{\mathrm{h}})\right)^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\vec{Z}-\vec{F}(\vec{h})\right)^T \mathbf{Cov}(\tilde{\mathrm{h}})^{-1}\left(\vec{Z}-\vec{F}(\vec{h})\right)} \tag{5.34}$$

Es gilt: Bei Kenntnis der Stützstellen $\vec{y}_s$ sind diejenigen Hyperparameter $\vec{h}$ am plausibelsten, welche den Dichtefunktionswert (auch Likelihood) L der multivariaten Normalverteilung $\mathcal{N}\left(\vec{F}, \mathbf{Cov}(\vec{h})\right)$ maximieren.

$$\max_{h_1,...,h_o} L(\vec{h}) = \max_{h_1,...,h_o} \left(\frac{1}{(2\pi)^{\frac{n}{2}} \det\left(\mathbf{Cov}(\tilde{\mathrm{h}})\right)^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\vec{y}_s-\vec{F}(\vec{h})\right)^T \mathbf{Cov}(\tilde{\mathrm{h}})^{-1}\left(\vec{y}_s-\vec{F}(\vec{h})\right)} \right) \tag{5.35}$$

Numerisch ist diese Formulierung ungünstig, da die Exponentialfunktion im Zähler sehr kleine Werte annehmen kann und die Determinante im Nenner sehr große. Eine gebräuchliche Lösung ist die Verwendung der logarithmierten Likelihood-Funktion (vgl.

[Viertl, 2013]):

$$\log(L(\vec{h})) = \log(1) - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\det(\mathbf{Cov})\right) - \frac{1}{2}\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\left(\vec{y}_s - \overrightarrow{F}\right) \tag{5.36}$$

Da der Logarithmus eine streng monoton wachsende Funktion ist, besitzen Gleichung 5.34 und Gleichung 5.36 das gleiche Maximum. Die Konstanten können ignoriert werden, da diese für das Maximum keine Rolle spielen:

$$\max_{h_1,\ldots,h_o} L(\vec{h}) = \max_{h_1,\ldots,h_o} \log(L(\vec{h})) = \max_{h_1,\ldots,h_o} \left(-\log\left(\det(\mathbf{Cov})\right) - \left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\left(\vec{y}_s - \overrightarrow{F}\right)\right) \tag{5.37}$$

Im weiteren Verlauf wird der Term kurz mit $\log(L)$ bezeichnet, wobei die Ableitung nach einem beliebigen Hyperparameter h_l wie folgt aussieht (die vollständige Herleitung ist in Anhang A.3.4 zu finden):

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\operatorname{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right) + \left(\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\mathbf{Cov}^{-1}\left(\vec{y}_s - \overrightarrow{F}\right)\right) \tag{5.38}$$

Damit ist es nun möglich, die unbekannten Hyperparameter mit einem geeigneten Optimierungsverfahren zu bestimmen. Einige in dieser Arbeit verwendeten Optimierungsverfahren werden in Kapitel 5.3.4 beschrieben.

Um eine Vorstellung von der Form einer solchen Likelihood-Funktion für das Co-Kriging-Verfahren zu bekommen, wird in Anhang A.2.5 ein einfaches Beispiel in Abhängigkeit verschiedener Hyperparameter gezeigt. Die Form der Funktion spielt auch für das gewählte Trainingsverfahren eine wesentliche Rolle.

5.3.2 Likelihood-Schätzer für Erwartungswerte und Varianzen

Die Maximierung von Gleichung 5.37 mittels der Gradienten aus Gleichung 5.38 erfordert die Kenntnis der Erwartungswerte $\overrightarrow{F} \in \mathbb{R}^{\mathrm{n_{all}}}$. Durch die Bildung der Kovarianzfunktionen (siehe Gleichung 5.14) werden auch die Varianzen und der Skalierungsfaktor benötigt.

Die Maximierung der Likelihood-Funktion in Abhängigkeit der Erwartungswerte $\overrightarrow{F} \in \mathbb{R}^{\mathrm{n_{all}}}$ besitzt eine analytische Lösung, welche folgend beschrieben wird. Die Anzahl der Gütestufen wird mit s gekennzeichnet, die Anzahl der Stützstellen einer Gütestufe $k \in \{1, ..., s\}$ wird mit n_k bezeichnet und n_{all} bezeichnet die Anzahl der Stützstellen aller Gütestufen. Das Maximum der Dichtefunktion L nach dem Maximum-Likelihood-Ansatz ($MLE()$ oder Maximum Likelihood Estimation) bezüglich des Vektors $\overrightarrow{F} \in \mathbb{R}^{\mathrm{n_{all}}}$ sieht wie folgt aus:

$$MLE\left(\vec{F}\right) = \max_{\vec{F}} \left(\frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{Cov}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\vec{y_s}-\vec{F}\right)^T \mathbf{Cov}^{-1}\left(\vec{y_s}-\vec{F}\right)} \right)$$

Sei $\mathbf{G} \in \mathbb{R}^{\mathrm{n_{all}} \times \mathrm{s}}$ wie folgt definiert:

$$\mathbf{G} = \overbrace{\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}}^{s\ Einträge} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix

perparameter freigegeben und dann über das Training bestimmt. Die Ableitung des Likelihood-Terms aus Gleichung 5.38 besitzt auch für die Varianz Gültigkeit.

5.3.3 Initialisierung der Hyperparameter für alle Krigingmodelle

Um das Training starten zu können, ist eine geeignete Initialisierung der Hyperparameter von großer Bedeutung. Diese kann die Konvergenz und auch die Stabilität der Minimierung stark beeinflussen. Innerhalb dieser Arbeit wurden mehrere Ansätze entwickelt um eine geeignete Initialisierung zu finden, welche folgend vorgestellt werden:

Allgemeine Vorbemerkungen und Definitionen

Die Anzahl an Stützstellen mit der ein Training initialisiert wird, kann dieses wesentlich beeinflussen. In [Gibbs and MacKay, 1997] wird vorgeschlagen, mindestens das zehnfache der Parameteranzahl zu verwenden, bevor eine vernünftige Schätzung möglich ist. Die Arbeit von [Jin et al., 2001] hingegen zeigt, dass je nach Funktion auch die dreifache Anzahl der Parameter genügt, um vernünftige Schätzungen zu erwarten. Letztlich ist keine einfache Aussage möglich, da die benötigte Anzahl der Stützstellen zu stark von der abzubildenden Funktion abhängt. Weiterhin sind die Prozessketten oftmals so langwierig oder die Konvergenzrate so niedrig, dass sehr aufwendige Initialisierungen, wie sie [Gibbs and MacKay, 1997] vorschlägt zeitlich nicht möglich sind. Eine sehr gute Möglichkeit, um die Datenlage zu beurteilen, ist die Nutzung der hier entwickelten Kriging-Analysesoftware (siehe Kapitel 5.6). So kann in der Initialisierungsphase regelmäßig ein Training durchgeführt und beurteilt werden. Besonders der Verlauf des Likelihood-Terms über der Optimierungslaufzeit eignet sich, um erkennen zu können, ob eine ausreichende Anzahl von Stützstellen vorhanden ist oder nicht.

Standardisierung der Daten Die innerhalb dieser Arbeit entwickelte Software nimmt intern eine Standardisierung der Daten vor. Dabei werden sowohl die Funktionswerte $\overrightarrow{y}_s \in \mathbb{R}^{\mathrm{n_{all}}}$ als auch die Parameter $\vec{x} \in \mathbb{R}^{\mathrm{k}}$ standardnormalverteilt standardisiert.

$$E[\overrightarrow{y}_s] = 0 \tag{5.40}$$
$$var[\overrightarrow{y}_s] = 1$$

$$\begin{matrix} E[x_i] = 0 \\ var[x_i] = 1 \end{matrix} \quad \forall x_i, i \in \{1, ..., k\} \tag{5.41}$$

Soll ein Training auf Basis eines älteren Trainings mit einer veränderten Datenbasis initialisiert werden, so ändern sich die Konstanten für die Standardisierung. Dadurch müssen die Hyperparameter (re)standardisiert werden. In Anhang A.3.8 wird ein solches Verfahren beschrieben.

Abschätzung konstanter Hyperparameter

Eine einfache und schnelle Methode eine Initialisierung für die Hyperparameter zu wählen, ist in [Schmitz, 2013] beschrieben. Dafür wird für alle Hyperparameter nur ein

Wert verwendet und eine obere und untere Grenze $\theta_{min}, \theta_{max} \in \mathbb{R}$ bestimmt.

$$\theta = \theta_i \forall i \in \{1, ..., s * k\} \tag{5.42}$$
$$\theta_{min} \leq \theta \leq \theta_{max}$$

Danach werden zwischen diesen Grenzen einige Werte ausprobiert und der Hyperparameter mit dem besten Likelihood-Term gewählt. Zur Bestimmung der minimalen und maximalen Grenze wird ein in [Schmitz, 2013] (Kapitel 5.2) beschriebenes Schätzverfahren verwendet.

Zufällige Initialisierung der Hyperparameter

Eine weitere Möglichkeit zur Initialisierung der Hyperparameter liegt in der zufälligen Erzeugung von Parametersätzen. Diese werden dann mit der Likelihood-Funktion bewertet und sortiert. Die zufällige Variation wird mit einer Gleichverteilung realisiert. Letztlich wird dann der Parametersatz gewählt, welcher die beste Likelihood-Funktion aufweist. Die Wahrscheinlichkeit einen sinnvollen Parametersatz zu finden, ist bei diesem Verfahren allerdings sehr klein.

Initialisierung auf Basis bereits vorhandener Krigingmodelle

Innerhalb einer Optimierung werden permanent neue Stützstellen berechnet. Deswegen wird regelmäßig neu trainiert und damit neue Hyperparameter bestimmt. In Annahme, dass sich die Hyperparameter bei ausreichender Datenlage nicht mehr wesentlich ändern, ist es sinnvoll die Hyperparameter aus den bereits trainierten Modellen zur Initialisierung zu verwenden. Diese Art der Wiederverwendung von alten Modellen ist für das Kriging Verfahren ohne Probleme möglich. Allerdings müssen zwei Problemstellungen beachtet werden:

1. Das letzte Modell befindet sich in einem lokalen Minimum der Likelihood Funktion
2. Die Hyperparameter der Kovarianz Funktion(en) sind noch nicht richtig eingestellt

Im ersten Fall besteht die Gefahr, dass das Training durch die ungünstige Initialisierung in einem lokalen Minimum bleibt und so nicht die optimalen Hyperparameter findet. Das wiederum führt zu schlechten Vorhersagen. Im Extremfall kann es sogar passieren, dass das lokale Minimum während der gesamten Optimierung nicht mehr verlassen wird. Der zweite Fall ist insbesondere am Anfang der Optimierung interessant, da dann nur wenig Samples verfügbar sind und somit die Schätzung der Hyperparameter noch stark variiert. Das in dieser Arbeit neu entwickelte Initialisierungsverfahren stellt eine hybride Form der zufälligen Initialisierung und der Verwendung alter Modelle dar. Ein Kriterium entscheidet darüber, ob ein altes Kriging-Modell verwendet wird oder eine zufällige Initialisierung durchgeführt wird. Hierfür werden die Likelihood-Werte einiger älterer Modelle mit den neuen Daten berechnet und das Modell mit dem besten Wert gewählt. Zusätzlich gilt, dass eine Verbesserung zum vorherigen Likelihood-Wert erreicht werden muss. Ist dies nicht der Fall, wird zufällig initialisiert. Zeitlich gesehen benötigt die Berechnung eines Likelihood-Werts im Vergleich zum gesamten Training nur einen Bruchteil der Zeit und fällt deswegen kaum ins Gewicht. Es werden im weiteren Verlauf noch einige Beispiele gezeigt, die den Nutzen dieses Verfahrens belegen.

Der verwendete Algorithmus wird durch folgenden Pseudcode beschrieben:

```
1  krigingFiles = getLastKrigingFiles(20)
2  wenn krigingFiles.size() < 20
3  dann
4      initType = random
5      end()
6
7
8  Schleife von i=0 bis  i < krigingFiles.size() Schrittweite 1
9      oldLikelihood = getLikelihood(krigingFiles[i])
10     newLikelihood = calculateLikelihood(krigingFiles[i])
11     wenn newLikelihood < bestLikelihood
12         und newLikelihood < oldLikelihood
13         und newLikelihood < -numberSamples/4.0
14     dann
15         bestLikelihood =newLikelihood
16         bestKrigingFile = krigingFiles[i]
17
18 wenn bestKrigingFile==leer
19 dann initType = random
```

Der Algorithmus startet mit dem Auslesen der Dateinamen der letzten 20 Krigingmodelle aus der laufenden Optimierung (Zeile 2). Sind noch keine 20 Krigingmodelle erzeugt worden, soll die Initialisierung zufällig erfolgen (Zeile 3-6), wobei dieser Wert vom Anwender eingestellt werden kann. In der darauf folgenden Schleife findet die Bewertung der einzelnen Kriging Modelle statt. Für die Bewertung muss erst der alte Likelihood-Wert ausgelesen werden. Dies geschieht in Zeile 10. Im nächsten Schritt wird der Likelihood-Wert mit der aktuellen Datenbasis neu berechnet. In der darauffolgenden If-Abfrage geht es darum, das beste Modell der 20 eingelesenen Krigingmodelle zu finden. Hierfür wird der beste Likelihood-Wert gesucht (siehe Kapitel 5.3.1).

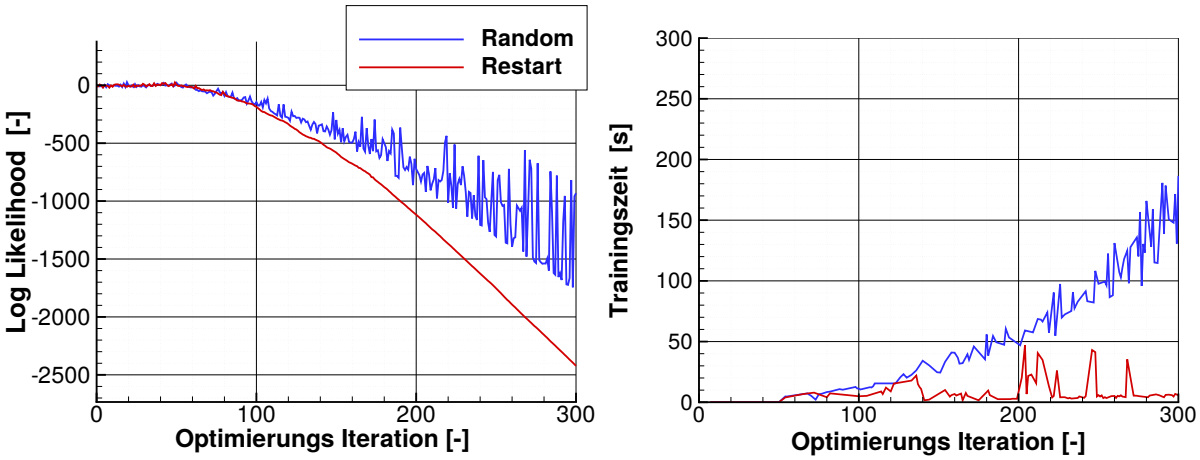


Abbildung 5.6: Vergleich verschiedener Initialisierungsverfahren und deren Auswirkung auf eine Testoptimierung

Zudem gilt die Bedingung, dass der neu berechnete Likelihood kleiner sein muss als der bereits eingelesene aus dem vorhergehenden Modell. Die Überlegung ist hierbei, dass eine neue Stützstelle, welche nicht in die bisher angenommene Verteilung passt, den Likelihood-Wert verschlechtert und die Hyperparameter neu eingestellt werden müssen. Als letzte Bedingung ist eine absolute Grenze für den Likelihood-Wert angegeben. Diese basiert auf Erfahrungswerten und soll sicherstellen, dass zu schlechte Modelle neu initialisiert werden. Dies ist oft am Anfang einer Optimierung der Fall,

wenn noch nicht genügend Daten vorhanden sind. Dieses Verfahren wird in der hier entwickelten Software als Standard verwendet. Abbildung 5.6 zeigt einen Vergleich der unterschiedlichen Initialisierungsverfahren anhand des Beispiels aus Kapitel 6.1.2. Auf der Ordinate ist der erreichte Likelihood-Term (kleiner ist besser) und auf der Abzisse der entsprechende Optimierungsschritt dargestellt. Durch die Initialsierung mit älteren Modellen wird ein erheblicher Zeitvorteil ohne Einbußen im Likelihood-Term beim Training erreicht. Weiterhin wird in Kapitel 5.3.4 ein Vergleich zwischen verschiedenen Trainingsverfahren angestellt, bei dem die hier beschriebene Initialisierung gewinnbringend eingesetzt wurde. In Kapitel 6 werden weitere Anwendungen gezeigt, in denen das Verfahren verwendet wurde.

5.3.4 Minimierungsverfahren

Innerhalb der hier entwickelten Software werden zwei verschiedene Minimierungsverfahren eingesetzt, welche im Folgenden beschrieben werden.

Minimierungsverfahren angelehnt an Resilient Backpropagation (RPROP)

Das erste hier verwendete Minimierungsverfahren ist angelehnt an ein Trainingsverfahren für neuronale Netzwerke, genannt RPROP (Resilient Backpropagation) [Riedmiller and Braun, 1993, Helbig and Scherer, 2011] und ist ein Verfahren erster Ordnung. Besonderheit des Verfahrens ist, dass es nur das Vorzeichen der partiellen Ableitungen verwendet und nicht die Werte selbst.

Die Änderung der Hyperparameter h_i für den nächsten Iterationsschritt $t+1$ ergibt sich aus der Schrittweite γ_i. Diese wird für jeden Hyperparameter einzeln bestimmt und in jeder Iteration geändert und hängt nur von dem Vorzeichen der entsprechenden partiellen Ableitung $\frac{\partial f}{\partial h_i}$ zum Zeitpunkt t der zu minimierenden Funktion f ab.

$$h_i^{t+1} = h_i^t - \gamma_i^t \text{sgn}\left(\left(\frac{\partial f}{\partial h_i}\right)^t\right)$$

Die Schrittweite wird in jedem Iterationsschritt für jeden Hyperparameter über zwei Multiplikatoren $\eta^+ \in \mathbb{R}; \eta^+ > 1$ und $\eta^- \in \mathbb{R}; \eta^- < 1$ einzeln angepasst. Ist die entsprechende partielle Ableitung des letzten Schritts multipliziert mit dem aktuellen Schritt größer als Null, wird die Schrittweite γ_i^t mit η^+ multipliziert und dadurch erhöht. Ist dieses Produkt kleiner als Null, dann wird die Schrittweite verkleinert durch Multiplikation mit η^-. Für die Schrittweite wird zudem eine Unter- und Obergrenze ($\gamma_{min}, \gamma_{max}$) festgelegt.

$$\gamma_i^{t+1} = \begin{cases} \min\left(\gamma_i^t \eta^+, \gamma_{max}\right) & wenn \left(\frac{\partial f}{\partial h_i}\right)^t \left(\frac{\partial f}{\partial h_i}\right)^{t-1} > 0 \\ \max\left(\gamma_i^t \eta^-, \gamma_{min}\right) & wenn \left(\frac{\partial f}{\partial h_i}\right)^t \left(\frac{\partial f}{\partial h_i}\right)^{t-1} < 0 \\ \gamma_i^t & sonst \end{cases}$$

In flachen Bereichen der zu minimierenden Funktion bietet dieses Verfahren große Vorteile gegenüber anderen gradienten basierten Verfahren, da in solchen Bereichen die Größe der Gradienten sehr klein ist und das RPROP-Verfahren die Größe nicht berücksichtigt. Das ist bei der Likelihood Funktion von besonderem Vorteil, da diese bereits durch Ihre Definition flache Gebiete aufweist, siehe Beispiel Anhang A.2.5.

Erweiterung des RPROP-Verfahrens an sehr viele Hyperparameter

Ein Nachteil des RPROP-Verfahrens ist, dass es relativ viele Iterationen benötigt bis es konvergiert. In jedem Iterationsschritt muss zum einen der Dichtefunktionswert der Likelihood-Funktion berechnet werden (siehe Kapitel 5.5.5) und zum anderen die partiellen Ableitungen nach den Hyperparametern (siehe Kapitel 5.5.6). Im Falle eines Co-Krigings skaliert die Anzahl der Hyperparameter $o = sk + 3s$ mit der Anzahl der Gütestufen s und der Anzahl an freien Variablen k. Die Anzahl der Hyperparameter o ist also grundsätzlich deutlich höher als bei einem Ordinary-Kriging. Aus diesem Grund kann es passieren, dass der Hauptaufwand dann in der Berechnung der partiellen Ableitungen des Likelihood-Terms liegt. Diesen Umstand kann man sich innerhalb des RPROP-Verfahrens zunutze machen:

	RPROP	RPROP2
Zeit	346.3s	186s
Mittl. Fehler	0.0201	0.0149
Iterationen	781	398

Tabelle 5.3: Ergebnisse beider Verfahren im direkten Vergleich. Die Ergebnisse stellen den Mittelwert der 10 durchgeführten Trainings dar.

Die Lernraten η^+, η^- sind im RPROP-Verfahren konstant (siehe Kapitel 5.3.4), insbesondere bei den anfänglichen Iterationsschritten führt dies zu einem relativ langsamen Anpassen der Deltas γ_i^t. Es wäre daher wünschenswert, die Lernraten ebenfalls zu variieren und dadurch eine schnellere Anpassung der Deltas γ_i^t zu erreichen. Eine gute Möglichkeit ist es, verschiedene Lernraten auszuprobieren. Dadurch müssen zwar mehr Likelihood-Auswertungen gemacht werden, dafür werden aber deutlich weniger Iterationen und damit weniger Berechnungen von partiellen Ableitungen benötigt. Beim Co-Kriging führt diese Vorgehensweise zu einer deutlichen Beschleunigung. Der folgende Pseudo-Programmcode zeigt die Umsetzung des neuen Verfahrens:

```
1  density = RPROPDensity(eta_plus, eta_minus)
2
3  // Teste kleinere und größere Lernraten für eta_plus
4  Schleife von eta_plusFact=0.9 bis eta_plusFact<=1.1 Schrittweite 0.2
5      newEtaPlus = eta_plus*eta_plusFact
6      wenn newEtaPlus<1.2 dann newEtaPlus=1.2
7      wenn newEtaPlus>2.0 dann newEtaPlus=2.0
8      wenn eta_plus==newEtaPlus dann überspringe
9
10     newDensity = RPROPDensity(newEtaPlus, eta_minus)
11     wenn newDensity<density dann eta_plus=newEtaPlus
12
13 // Teste kleinere und größere Lernraten für eta_minus
14 Schleife von eta_minusFact=0.9 bis eta_minusFact<=1.1 Schrittweite 0.2
15     newEtaMinus = eta_minus*eta_minusFact
16
17     wenn newEtaMinus<0.4 dann newEtaMinus=0.4
18     wenn newEtaMinus>0.7 dann newEtaMinus=0.7
19     wenn eta_minus==newEtaMinus dann überspringe
20
21     newDensity = RPROPDensity(eta_plus, newEtaMinus)
```

```
22      wenn newDensity<density dann eta_minus=newEtaMinus
```

Als erstes wird der initiale Likelihood-Wert mit den bisherigen Lernraten η^+, η^- berechnet. Danach wird η^+ leicht erhöht/verringert und jeweils der Likelihood-Wert bestimmt. Sollte sich das neue η^+ nicht geändert haben, so wird die Berechnung des Likelihood-Werts gespart. Am Ende wird die Lernrate beibehalten, welche den besten Likelihood-Wert erzielt hat. Für die Lernrate η^- wird dasselbe Prinzip durchgeführt.

Für diese Art der Lernratenregelung sind maximal 4 neue Likelihood-Auswertungen notwendig, im Gegenzug hat man allerdings eine deutliche Verringerung der Iterationsanzahl und muss somit weniger partielle Ableitungen bestimmen. Diese sollte insbesondere für das Co-Kriging von großen Vorteil sein.

Um das Verfahren zu validieren, wurde eine Datenbasis aus einer aktuellen Optimierung für einen gegenläufigen Rotor verwendet ([Lengyel-Kampmann, 2015]). Es gab ca. 113 freie Parameter und somit für das Co-Kriging 228 Hyperparameter (130*2 und 2x die Prozessvarianzen der Kovarianzfunktionen). Die Datenbasis enthielt zu diesem Zeitpunkt 373 Individuen. Das Co-Kriging wurde mit zufälligen Hyperparametern initialisiert und jeweils 10x mit dem RPROP/RPROP2-Verfahren trainiert und verglichen. Die Vorhersagen der fertig trainierten Ersatzmodelle wurden dann anhand einer Testdatenbasis validiert und ein mittlerer Vorhersagefehler bestimmt. Tabelle 5.3 zeigt die Ergebnisse beider Verfahren.

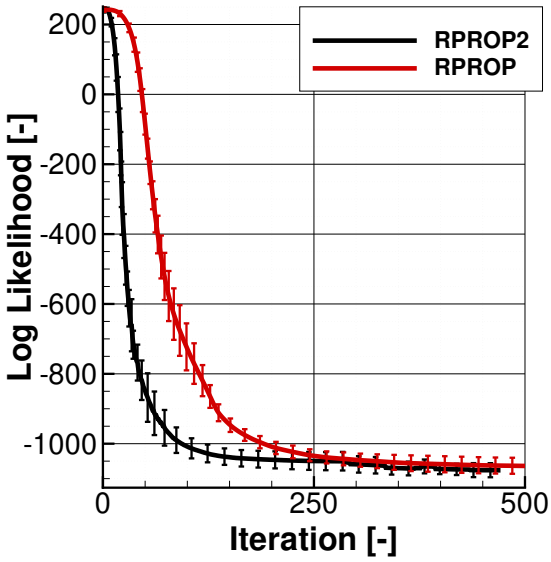


Abbildung 5.7: Gemittelter Konvergenzverlauf der beiden Verfahren anhand eines Beispiels

Das RPROP-Verfahren benötigt in diesem Beispiel die 1.86-fache Zeit, um Konvergenz zu erreichen. Dies wird hauptsächlich durch die geringere Iterationsanzahl erreicht. Die mittleren Fehler sind in etwa vergleichbar. Abbildung 5.7 zeigt den gemittelten Trainingsverlauf beider Verfahren. Die rote und schwarze Kurve stellen den mittleren Dichtefunktionswert über den Iterationsschritten dar. Die Fehlerbalken sind die Standardabweichungen der verschiedenen Trainings. Es lässt sich erkennen, dass das RPROP2-Verfahren insbesondere am Anfang einen deutlich schnelleren Konvergenzverlauf besitzt. Bei einem Verfahren wie dem Co-Kriging, in dem eine hohe Anzahl an Hyperparametern zu erwarten ist, kann dieses Verfahren einen deutlichen Geschwindigkeitszuwachs bringen.

Quasi-Newton

Das zweite implementierte Minimierungsverfahren ist ein Quasi-Newton Verfahren höherer Ordnung. Basis für diese Art der mehrdimensionalen Minimierung ist eine Taylor Approximation zweiten Grades. Sei t der Iterationsschritt und $\mathbf{H}$ die Hesse Matrix, so

gilt:

$$f\left(\vec{\theta}\right) \approx f\left(\vec{\theta}_t\right) + \left(\vec{\theta} - \vec{\theta}_t\right)^T \nabla f\left(\vec{\theta}_k\right) + \frac{1}{2}\left(\vec{\theta} - \vec{\theta}_t\right)^T \mathbf{H}\left(\vec{\theta}_t\right)\left(\vec{\theta} - \vec{\theta}_t\right)$$

Die entsprechende Ableitung dieser Funktion muss im Minimum oder Maximum der Funktion Null ergeben:

$$\nabla f\left(\vec{\theta}\right) \approx \nabla f\left(\vec{\theta}_t\right) + \mathbf{H}\left(\vec{\theta}_t\right)\left(\vec{\theta} - \vec{\theta}_t\right) = 0$$

Besonderheit bei der Quasi-Newton-Methode ist, dass die Hesse Matrix $\mathbf{H}$ nicht direkt berechnet werden muss, sondern sukzessive über die Gradienten angenähert wird. Das Verfahren bietet den Vorteil, dass es deutlich schneller konvergiert als das bereits vorgestellte Verfahren erster Ordnung. Allerdings ist es weniger robust und kann in flachen Gebieten der Funktion langsam bis gar nicht konvergieren. Die exakte Umsetzung des Algorithmus und weitere Details können in [Press et al., 2007, Gill et al., 1981, Gill, 2007] gefunden werden.

Vergleich Quasi-Newton/RPROP/RPROP2 und Initialisierungsverfahren

In diesem Kapitel soll ein direkter Vergleich zwischen den verschiedenen Optimierungsverfahren angestellt werden. Hierfür wurden Daten aus der Fanstufenoptimierung (siehe Kapitel 6.2) zusammengestellt, welche im Folgenden beschrieben werden.

- Trainingsfunktion: $\dot{m}_{OP1}$ (Massenstrom Betriebspunkt 1)
- Stützstellen (niedrige/hohe Güte): 300 / 500
- Testdatenbasis: 400 Stützstellen hoher Güte
- Maximal 1000 Trainingsschritte
- 20 Threads auf zwei Intel(R) Xeon(R) CPU E5-2650 v3
- Pro Testfall 10 aufeinanderfolgende Trainings
- Das gewählte Initialisierungsverfahren ist das „Initialisierung auf Basis vorhandener Kriging Modelle“ aus Kapitel 5.3.3. Das Verfahren nutzt 5 bereits trainierte Krigingmodelle

Abbildung 5.8 zeigt die Ergebnisse der 3 verschiedenen Testfälle. Aufgetragen ist jeweils der Wert des Likelihood-Terms, die Trainingszeit und der Testfehler. Nach 5 Iterationen startet die Initialisierung aus Kapitel 5.3.3, da ab diesem Zeitpunkt genügend alte Krigingmodelle vorhanden sind. Dieser Übergang wird mit einer gestrichelten Linie gekennzeichnet.

RPROP	RPROP2	Quasi Newton
53.5 s	42.83 s	12.7 s

Tabelle 5.4: Mittlere Trainingszeit für alle durchgeführten Trainings in Sekunden.

Der Testfehler ist bei allen Verfahren auf einem ähnlich guten Niveau: Der Mittelwert des Massenstroms liegt bei 514 kg/s und der Testfehler (absolute Differenz zwischen

Vorhersage und Echtwert) liegt damit bei ca. 1%. Das RPROP2-Verfahren zeigt insgesamt den besten Testfehler. Das Quasi-Newton-Verfahren zeigt bei Iteration 4 einen starken kurzfristigen Anstieg des Testfehlers, weil der Skalierungsfaktor a (siehe Kapitel 4.3.1) einmalig falsch eingeschätzt wurde. Der genaue Trainingsverlauf (hier nicht dargestellt) lässt auf ein lokales Minimum in der Likelihood-Funktion schließen. Zu diesem Zeitpunkt war noch die zufällige Initialisierung aktiviert. Nachdem die „Initialisierung auf Basis vorhandener Krigingmodelle“ aktiviert wurde, blieb der Testfehler bei allen Verfahren nahezu konstant. Der Anstieg des Testfehlers beim Quasi-Newton-Verfahren lässt deshalb auf eine einmalig schlechte Initialisierung als Ursache schließen. Die Verläufe des Likelihood-Terms ähneln sich bei den RPROP-Verfahren sehr stark, beide Verfahren finden bereits in der Phase zufälliger Initialisierung direkt den optimalen Likelihood-Term. Das Quasi-Newton-Verfahren ist in dieser Phase noch sehr instabil. Die Trainingszeiten zwischen den Verfahren variieren in der „zufälligen“ Phase noch sehr stark. Tabelle 5.4 zeigt die gemessenen mittleren Trainingszeiten. Das RPROP2-Verfahren bietet eine Zeitersparnis von ca. 20% gegenüber dem RPROP-Verfahren und das Quasi-Newton Verfahren eine Zeitersparnis von ca. 74% gegenüber dem RPROP-Verfahren. Nach der „zufälligen“ Phase sind die Trainingszeiten zwischen den Verfahren allerdings nahezu identisch.

Fazit: Dieses Testbeispiel zeigt zum einen die Robustheit der beiden RPROP-Verfahren gegenüber dem Quasi-Newton Verfahren und zum anderen den Nutzen des in dieser Arbeit entwickelten Intialisierungsverfahrens. Letzteres verbessert zum einen die Trainingszeit und zum anderen sorgt es auch für ein stabileren Likelihood-Verlauf in allen Verfahren.

Aufgrund dieser Ergebnisse werden als Standardverfahren innerhalb der hier entwickelten Co-Kriging Software die „Initialisierung auf Basis vorhandener Kriging Modelle“ und das RPROP2-Verfahren gewählt. Für den Ordinary- und Gradient-Enhanced-Modus der Kriging-Software wird allerdings das RPROP-Verfahren bevorzugt, da dieses bei einer geringeren Anzahl von Hyperparametern zeitlich besser abschneidet.

5.4 Umsetzung der Entscheidungsfunktion

Die Nutzung der in dieser Arbeit entwickelten Entscheidungsfunktion (siehe Kapitel 3.2.2) benötigt die Berechnung der bedingten Varianz $\sigma^2_{g|j}(\vec{x}) = \sigma^2_g(\vec{x}) \mid y_j(\vec{x}) \in \mathbb{R}$ (Definition Seite 31) und der verschiedenen Zeiten $T = \frac{t_{train}+t_{opti}+t_{prh}}{t_{train}+t_{opti}+t_{prl}}$. Wie die genaue Berechnung innerhalb der hier entwickelten Software funktioniert, soll in diesem Abschnitt erläutert werden.

Bedingte Varianzen: Die bedingte Varianz $\sigma^2_{g|j}(\vec{x})$ kann mithilfe einer bedingten Normalverteilung bestimmt werden. Hierfür wird die bedingte Kovarianz $cov\left(Z_g(\vec{x}), Z_j(\vec{x})\right) | \vec{y_s}, \vec{w}$ (siehe Gleichung 4.17) zwischen den verschiedenen Gütestufen g, j an dem Ort $\vec{x} \in \mathbb{R}^k$ unter der Bedingung der Kenntnis aller bekannten Daten $\vec{y_s}$ und Gewichte $\vec{w}$ benötigt. Ist diese bekannt, so gilt für $\sigma^2_{g|j}(\vec{x})$ bei Annahme einer

Multivariaten-Normalverteilung und unter der Kenntnis einer Stützstelle $y_j(\vec{x})$ folgender Zusammenhang:

$$\sigma^2_{g|j}(\vec{x}) = \sigma^2_g(\vec{x}) - \frac{\left(cov\left(Z_g(\vec{x}), Z_j(\vec{x})\right)|\vec{w}, \vec{y}_s\right)^2}{\sigma^2_j(\vec{x})} \tag{5.43}$$

Damit lässt sich die bedingte Kovarianz einfach und effizient bestimmen und für eine Anwendung wie die Entscheidungsfunktion sinnvoll nutzen.

Berechnung der Zeiten: Grundlegend gibt es für die Entscheidungsfunktion drei verschiedene Zeiten, welche bestimmt werden müssen.

1. Die Trainingszeit t_{train}
2. Die Zeit für die Optimierung auf dem Ersatzmodell t_{opti}
3. Die Prozesskettenzeit t_{pr}, wobei es für jede Gütestufe eine einzelne Zeit gibt.

Die Zeiten innerhalb der Optimierung werden gemessen und gespeichert. Auf Basis dieser Messungen können Schätzungen über den weiteren Verlauf angestellt werden. Hierbei sollte unterschieden werden:

Die Prozesskettenzeit t_{pr} ändert sich während einer laufenden Optimierung in der Regel nur unwesentlich, da sich die Prozesskette während der Optimierung nicht ändert. Bei CFD-Optimierungen ist oftmals eine kleine Reduktion der Prozesskettenzeit zum Ende einer Optimierung zu beobachten, da Individuen nahe dem Optimum oftmals schneller konvergieren. Allerdings ist dies nicht immer der Fall und die zeitliche Differenz nicht ausschlaggebend. Aus diesem Grund wird in der hier entwickelten Software das Mittel aus allen bereits gemessenen Prozesskettenzeiten $t_{pr,i}, i \in \{1, ..., n\}$ verwendet, wobei n die Anzahl aller bisher berechneten Individuen der betrachteten Gütestufe angibt.

$$t_{pr} = \frac{1}{n} \sum_{i=1}^{i \leq n} t_{pr,i} \tag{5.44}$$

Die Trainingszeit t_{train} sowie die Zeit für die Optimierung auf dem Ersatzmodell t_{opti} ändern sich im Verlauf der Optimierung stetig. Typischerweise wachsen diese mit steigender Stützstellenanzahl an. Verwendet man bswp. eine Restart-Methode, wie die in dieser Arbeit entwickelte (siehe Kapitel 5.3.3), dann kann die Trainingszeit aber auch schnell fallen. Aufgrund dieser zeitlichen Schwankungen sollte eine kleinere Anzahl m an bereits gemessenen Zeiten verwendet werden, um mögliche Änderungen schneller abzubilden:

$$t_{train} = \frac{1}{m} \sum_{i=1}^{i \leq m} t_{train,(n-i)} \tag{5.45}$$

$$t_{opti} = \frac{1}{m} \sum_{i=1}^{i \leq m} t_{opti,(n-i)} \tag{5.46}$$

Standardeinstellung in der hier entwickelten Software ist eine Anzahl von $m = 15$ und für die Prozesskettenzeiten $m = 100$.

Minimale Menge an Individuen hoher Güte: Es kann vorkommen, dass die Funktionen der Gütestufen so stark korreliert sind, dass die Entscheidungsfunktion nur noch Individuen niedriger Güte wählt. Prinzipiell ist dieses Verhalten auch sinnvoll und erwünscht, jedoch interessiert den Anwender letztendlich das Ergebnis hoher Güte. Aus diesem Grund wird der Volumenzugewinn im Raum hoher Güte berechnet.

Daher kann es sinnvoll sein, eine Mindestrate von Individuen hoher Güte vorzugeben. In der hier entwickelten Software wird dafür ein prozentualer Anteil hoher Güte vorgegeben. Wird dieser Anteil unterschritten, so soll ein Individuum hoher Güte erzwungen werden. Bei besonders aussichtsreichen Individuen niedriger Güte kann dieser Zwang allerdings auch umgangen werden. Die Entscheidungsfunktion wird also nur teilweise ausgehebelt.

Als besonders aussichtsreich werden Individuen bezeichnet, welche einen höheren Informationszugewinn pro Zeit (siehe Gleichung 3.7) aufweisen, als alle Individuen der letzten 10 Optimierungsschritte:

1. Bestimmung des maximalen Informationszugewinns pro Zeit $f_{max} := \max_{i \in \{1,...,10\}} \left(\left[\max_{g \in J} \left(\frac{I_g}{t_g} \right) \right]_i \right)$ der Gütestufe $g \in J = \{1, ..., s\}$ für den vergangenen Optimierungszeitschritt i. In diesem Fall werden die letzten 10 Schritte betrachtet, wobei $i = 0$ den aktuellen Schritt beschreibt und $i = 1$ den vorhergehenden.
2. Besitzt im aktuellen Zeitschritt $i = 0$ eine Gütestufe (außer der höchsten) $g_l \in L = \{2, ..., s\}$ einen höheren Informationszugewinn pro Zeit als die letzten 10, dann wähle diese Stufe. Ansonsten wird ein Individuum der höchsten Stufe erzwungen.

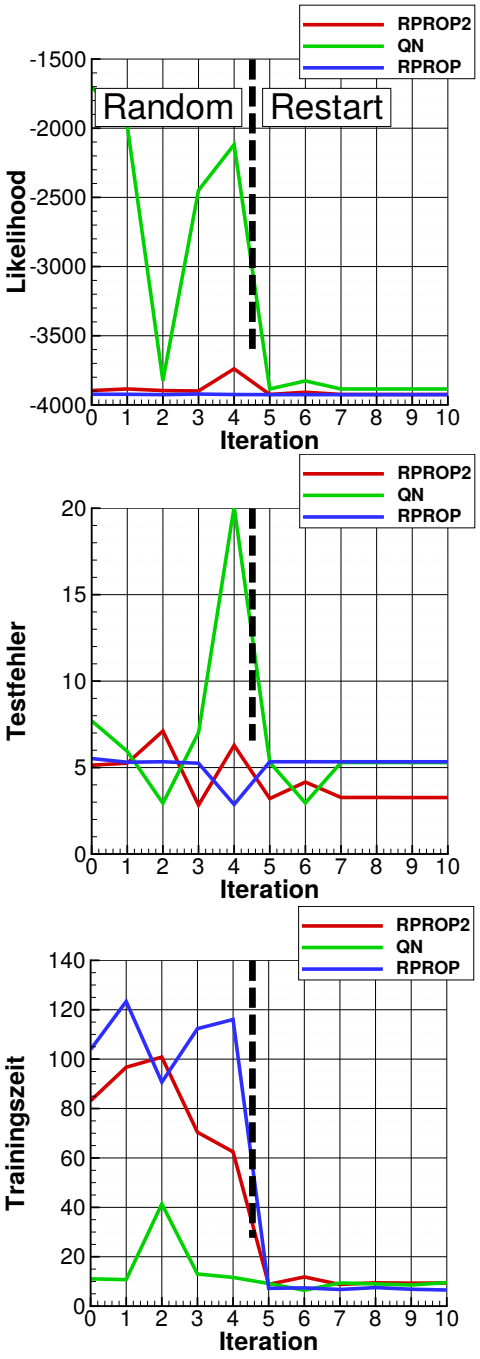


Abbildung 5.8: Ergebnisse des Testfalls. Dargestellt sind die Ergebnisse des Likelihoods, der Trainingszeit und des Testfehlers über den Trainingsiterationen.

Dieser Algorithmus sorgt im Normalfall für ein gleichmäßiges „Erzwingen“ von Individuen der höchsten Gütestufe, allerdings können besonders aussichtsreiche Individuen der niedrigeren Gütestufen so vorgezogen werden.

5.5 Softwaretechnische Umsetzung

Innerhalb dieses Abschnittes werden wichtige softwaretechnische Entwicklungen dieser Arbeit vorgestellt. Insbesondere der modulare Aufbau der Software, der es ermöglicht, verschiedene Verfahren innerhalb einer Software zu implementieren und so Redundanzen zu vermeiden. Darauffolgend werden Verfahren vorgestellt, welche die Effizienz der Software steigern und auf verschiedenen Hardwarearchitekturen einsetzbar sind. Am Ende des Abschnitts wird eine im Rahmen dieser Arbeit entwickelte asynchrone und ausfallsichere Netzwerkbibliothek vorgestellt, welche es ermöglicht, zahlreiche numerische Operationen asynchron auf mehrere Rechner zu verteilen.

5.5.1 Softwarearchitektur und Laufzeitumgebung

Zahlreiche in der Literatur vorgestellte Umsetzungen der Kriging-Verfahren (siehe [Le Gratiet, 2013, Forrester et al., 2007]) sind oftmals in Matlab oder ähnlichen Umgebungen realisiert. Dabei entstehen oftmals Schwächen in der Performance, der Erweiterbarkeit und bei der Benutzerfreundlichkeit. Das hier entwickelte Verfahren wurde von Anfang an auf Erweiterbarkeit und Modularität ausgelegt. Hierfür kommen Objektorientierung und eine moderne interfacebasierte Programmierung (siehe [Steimann et al., 2016]) zum Einsatz. Dies ermöglicht eine sehr große Flexibilität und Modularität in der Software, was an den zahlreichen integrierten Verfahren und nutzbaren Architekturen ersichtlich ist. Die gewählte Programmiersprache ist C++, wobei der GCC-Compiler ab Version 4.7 unterstützt wird. Eine Kompilierung mit dem Intel Compiler ist ebenfalls ab Version 12.1 möglich. Weiterhin werden die Messaging-Middleware-ZeroMQ und die Boost-Bibliothek Version 1.62 benötigt. Um einen Einblick in die Softwarearchitektur zu bekommen, werden im Folgenden die wichtigsten Klassen benannt und deren Funktionsweise kurz beschrieben.

Point: Die Klasse „Point“ (siehe Abbildung A.9) beschreibt eine Stützstelle innerhalb des Kriging-Verfahrens. Ein Objekt dieser Klasse beinhaltet alle Parameter, Funktionswerte und ggf. partielle Ableitungen für das Gradient-Enhanced-Kriging. Die partiellen Ableitungen werden als assoziatives Datenfeld gespeichert, um die Zuordnung zu den Parametern zu gewährleisten und um unvollständige Daten zuzulassen. Der Zugriff auf Variablen, Funktionswerte und partielle Ableitungen erfolgt über entsprechende Zugriffsmethoden („Get“ und „Set“ Methoden). Zudem implementiert die Klasse das Interface „SaveableOnServer“. Objekte dieser Klasse können somit über das Netzwerk verteilt werden. In Kapitel 5.5.2 wird dieser Punkt noch genauer erklärt.

CorrelationFunction: Die Klasse „CorrelationFunction“ ist eine virtuelle Klasse und legt den Aufbau der Kovarianzklassen fest. Die Spezialisierungen entsprechen den Implementierungen der einzelnen Kovarianzfunktionen. Im Rahmen dieser Arbeit wurden die Gauss-, Spline- und Exponential-Kovarianzfunktionen umgesetzt. Die Gleichungen der Funktionen und auch deren Ableitungen nach Hyperparametern und Ort sind in Anhang A.2.3, A.2.2, A.2.1 zu finden. Für ein Co-Kriging gibt es die Spezialisierung „CorrelationFunctionLinearCombination“, die Funktionsweise dieser Kovarianzfunktion

soll hier kurz erläutert werden: Um die Kovarianzmatrix für das hier beschriebene Co-Kriging-Modell aufzustellen, werden gemäß Gleichung 5.14 zwei Kovarianzmodellfunktionen mit eigenen Hyperparametern benötigt.

Konkret bedeutet dies, dass zwei Instanzen einer Kovarianzfunktion erzeugt werden müssen ($cov_2\left(\vec{x}_1, \vec{x}_2\right), cov_{diff}\left(\vec{x}_1, \vec{x}_2\right)$). Die Kovarianzfunktion $cov_{1,1}\left(\vec{x}_1, \vec{x}_2\right)$ wird dann über die „CorrelationFunctionLinearCombination“ abgebildet, indem diese die Referenzen auf die beiden bereits vorhandenen Kovarianzfunktionen erhält. Die Steuerung und Instanziierung der Objekte wird von der Klasse „CorrelationContainer“ übernommen, welche noch erläutert wird.

Die Methoden getAllCov und getAllCovPartialDerTheta berechnen die eigentliche Kovarianzfunktion zwischen zwei Punkten „point1“ und „point2“ und deren partielle Ableitung nach einem Hyperparameter „partialDerNumber“. Die Ergebnisse werden in die Matrix „CovMat“ und „CovMatrixDer“ geschrieben, welche jeweils als Referenz übergeben werden. Durch die unterschiedlichen Verfahren, wie z.B. dem Gradient-Enhanced- oder dem Co-Kriging, können die Kovarianzmatrizen unterschiedliche Strukturen aufweisen. Die Position der unterschiedlichen Einträge der Matrix wird daher festgelegt und in einem global verfügbaren internen assoziativen Container gespeichert. Hierdurch wird der Aufbau der Matrix zentral gespeichert, was mögliche Änderungen vereinfacht. Zudem ist damit eine effiziente asynchrone Parallelisierung möglich. Weiterhin wird in Abschnitt 5.5.3 gezeigt, wie die Berechnung der Kovarianzfunktion mit SIMD-Instruktionen beschleunigt werden kann. In Anhang A.3.11 wird das UML-Diagramm der Klasse mit den entsprechenden Spezialisierungen dargestellt.

CorrelationContainer - Behälter und Kontrolle für die benötigten Korrelationsfunktionen: Die Klasse CorrelationContainer beinhaltet und steuert alle Instanzen der verwendeten Kovarianzfunktionen. Bei diesem Container handelt es sich um ein „Singleton“, es existiert also nur eine Instanz innerhalb des Programms. Aus diesem Grund wird auf ein UML-Diagramm verzichtet, da keine Abhängigkeiten bestehen. Bei der Initialisierung des Kriging-Modells erzeugt diese Klasse alle nötigen Kovarianzfunktionen und setzt im Falle des Multifidelity-Verfahrens die entsprechenden Referenzen (siehe Unterabschnitt „CorrelationFunction“). Alle weiteren Zugriffe auf die Kovarianzfunktionen erfolgen immer über ein Objekt dieser Klasse und niemals direkt, wodurch Redundanzen vermieden werden. Weiterhin wird die Zuordnung der Kovarianzobjekte auf entsprechende Gütestufenpaare von dieser Klasse übernommen. Dies bedeutet konkret, dass das Aufstellen der Kovarianzmatrizen immer mithilfe des CorrelationContainer erfolgt und niemals direkt über die Kovarianzfunktionen selbst. Da die Kovarianzfunktionen an sehr vielen Stellen im Programm aufgerufen werden, kann durch diese Zentralisierung die Fehleranfälligkeit des Programms deutlich reduziert werden.

Neben der Steuerung der Kovarianzfunktionen übernimmt diese Klasse auch die Steuerung der Hyperparameter. Innerhalb des Programms werden die Hyperparameter in mehrere Gruppen aufgeteilt und in unterschiedlichen Objekten gespeichert. Dies kommt durch die unterschiedliche Natur der Hyperparameter zustande. Beispielsweise kann der Skalierungsfaktor a nicht direkt einer Kovarianzmodellfunktion zugeordnet werden und sollte aus diesem Grund in einer übergeordneten Instanz verwaltet werden. Diese übergeordnete Instanz ist der CorrelationContainer. Ebenfalls stellen die Diagonalaufschläge eine Besonderheit dar und werden dort ebenfalls zentral verwal-

tet.

Eine weitere Aufgabe ist die zentrale Speicherung der Positions-Tabelle, welche einem Paar der Klasse „Point“ eine Position in der Kovarianzmatrix und deren Ableitungen zuordnet.

DensityFunction - Implementierung unterschiedlicher Verfahren in einem Softwarepaket: Die abstrakte Klasse „DensityFunction“ und deren Spezialisierungen sollen es ermöglichen, unterschiedliche Interpolations-/Klassifikations-Verfahren innerhalb des Programms umzusetzen. Prinzipiell lassen sich mit dieser Klassenstruktur alle Arten von Verfahren einbinden, die die folgenden Merkmale aufweisen:

- Abstands- oder Kovarianzmatrix mit Ableitungen
- Verwendung einer Abstandsfunktion in Form der Klasse „CorrelationFunction“, siehe Abschnitt 5.5.1
  - Hyperparameter, welche die Kovarianzmatrix beeinflussen
- Bewertungsfunktion des aktuellen Hyperparametersatzes, z.B. Likelihood mit Ableitungen nach den Hyperparametern

Sofern diese Merkmale vorhanden sind, kann eine neue Spezialisierung der Klasse „DensityFunction“ erzeugt werden. Die Bewertungsfunktion soll mit der Methode „calcDensity“ umgesetzt werden und einen Fließkommawert zurückgeben, welcher die Qualität des aktuellen Modells zurückgibt, wobei hier kleinere Werte als besser angesehen werden. Die Methode „calcDensityDerivative“ gibt alle Ableitungen der Bewertungsfunktion nach allen Hyperparametern in Form eines Vektors zurück. Die Methode „createAndInvertCovMat“ erzeugt die Kovarianzmatrix und nimmt die notwendige Invertierung oder Zerlegung vor. Bisher implementierte Spezialisierungen sind:

- ReducedNormalDistribution: Likelihood-Funktion für alle Kriging-Verfahren
- SVMDistribution: Eine Implementierung des Klassifikators „Supporting Vector Machines“. Diese wurden im Rahmen einer studentischen Bachelorarbeit implementiert. Weitere Informationen sind in [Schumacher, 2014] zu finden.
- CorrelationClassificator: Einfacher Klassifikator auf Basis von Korrelationsfunktionen. Zu diesem Verfahren existieren bis zum jetzigen Zeitpunkt noch keine Veröffentlichungen.

Die Verfahren „SVMDistribution“ und „CorrelationClassificator“ sollen hier nicht weiter thematisiert werden, da sie nicht Teil dieser Arbeit sind und innerhalb von studentischen Arbeiten realisiert wurden. Dennoch kann damit gezeigt werden, dass die hier etablierte Klassenstruktur die Implementierung von unterschiedlichen Verfahren einfach realisierbar macht und vorhandene Softwarestrukturen so effizient wiederverwendet werden können. In Anhang A.3.12 ist das entsprechende UML-Diagramm dieser Klasse zu finden.

Minimierungsverfahren: Die abstrakte Klasse „MinimizerInterface“ beschreibt die Schnittstelle für ein Minimierungsverfahren, wobei fünf Spezialisierungen diese Schnittstelle implementieren. Die Struktur wird in dem UML-Diagramm in Anhang A.3.13 beschrieben. Die Spezialisierungen „MinimizerRandom“ und „MinimizerRandom2“ werden in Kapitel 5.3.3 beschrieben. Dabei handelt es sich um zufallsbasierte Initialisierungsverfahren. Weiterhin macht die Software keinen Unterschied zwischen

einem Trainings- und Initialisierungsverfahren, beide Verfahren sind auch als Trainingsverfahren zulässig und umgekehrt. Die Spezialisierungen „MinimizerRPROP“, „MinimizerRPROP2“ und „MinimizerQN“ stellen die Trainingsverfahren aus Kapitel 5.3.4 dar.

Für alle diese Verfahren schreibt die Schnittstelle „MinimizerInterface“ zwei öffentliche Methoden vor:

- callMinimizer: Mit diesem Aufruf wird das Verfahren gestartet und das Minimum gesucht.
- MinimizerInterface (Konstruktor): Diesem Konstruktor muss ein Objekt vom Typ „DensityFunction“ übergeben werden, womit dann das gesamte Minimierungsproblem beschrieben ist (siehe Kapitel 5.5.1 - „DensityFunction“). Innerhalb des Objekts vom Typ „DensityFunction“ werden dann die Methoden „calcDensity“ und „calcDensityDerivative“ verwendet, welche den zu minimierenden Term und dessen Ableitungen liefern.

Die privaten Methoden implementieren das zu minimierende Problem und sprechen dafür die Methoden aus dem übergebenen Objekt vom Typ „DensityFunction“ an. Aufgerufen werden diese innerhalb der Methode „callMinimizer“, welche die Ergebnisse dann weiterverwendet.

Beispielsweise beschreibt die Methode „function“ die Auswertung der zu minimierenden Funktion und wird in der Methode „callMinimizer“ verwendet. Diese wird mit dem aktuellen Hyperparametersatz „variables“ aufgerufen und übergibt diese dann weiter an das Objekt des Typs „DensityFunction“. Wurden die Parameter erfolgreich übergeben, wird die Methode „calcDensity“ aufgerufen und der Funktionswert bestimmt.

Die Methode „saveFunction“ beschreibt das Verhalten des Verfahrens, wenn ein Iterationsschritt nicht erfolgreich war, z.B. wenn die Cholesky-Zerlegung der Matrix nicht funktioniert. In solch einem Fall wird die Schrittweite halbiert, der letzte Schritt noch einmal ausgeführt und der Diagonalaufschlag erhöht. Wenn keine dieser Maßnahmen greift, wird das Minimierungsverfahren abgebrochen.

Die Methode „convergenceCheck“ prüft die Konvergenz des Verfahrens. Das Verfahren gilt als konvergiert, wenn sich die Funktionswerte seit einer bestimmten Anzahl an Iterationen nicht mehr verändert oder verbessert haben.

Die Methoden „constraintFunction“ und „constraintFunctionDerivative“ geben einen Restriktionsterm, bzw. dessen Ableitung zurück und sind nur optional.

Die Methode „functions“ bekommt eine Liste von Variablensätzen und wertet diese parallel aus.

Matrixoperationen:

Für die Vorhersage (siehe Gleichungen 4.2.3, 4.2.4, 4.2.5) wie auch für das Training (siehe Gleichungen 5.38, 5.37) eines Kriging-Modells sind viele komplexe Matrixoperationen notwendig von denen die Laufzeit der Vorhersage und des Trainings hauptsächlich abhängt. Weiterhin werden diese Modelle innerhalb verschiedenster Hardware-Architekturen verwendet, bspw. AMD/Intel CPUs und NVidia GPUs. Für jede dieser Hardwarearchitekturen gibt es verschiedene Hersteller-Bibliotheken und damit Implementierungen der benötigten Matrixoperationen. Diese Implementierungen sind stark

auf die jeweilige Architektur optimiert und bringen enorme Geschwindigkeitsvorteile. Eine einheitliche Softwareschnittstelle existiert hierfür leider nicht. Aus diesem Grund wurde im Rahmen dieser Arbeit ein einheitliches polymorphes Matrix-Interface und verschiedene Implementierungen entwickelt (siehe Abbildung A.13). Die Implementierungen sind untereinander konvertier- und austauschbar. Bislang stehen drei verschiedene Spezialsierungen zur Verfügung, welche im Folgenden erläutert werden:

Matrix: Innerhalb der Superklasse „Matrix“ werden hauptsächlich die Speicherverwaltung und Zugriffsmethoden auf eine beliebige Matrix-Klasse vorgegeben. Durch diese einheitliche Speicherverwaltung ist die Austauschbarkeit der einzelnen Spezialisierungen erst möglich. Diese Klasse ist also keine abstrakte Klasse, sondern vererbt einige Methoden und auch Attribute an die Sub-Klassen. Weiterhin implementiert die Klasse das Interface *„SaveableOnServer“,* wodurch die Matrix-Klasse über das in Kapitel 5.5.2 beschriebene Verfahren asynchron und rechnerweit parallelisierbar wird.

Die vollständige Datenstruktur der Klasse Matrix und deren Spezialisierungen sind in dem Array „elements“ gespeichert. In Tabelle 5.5 ist die Struktur ersichtlich. Neben den eigentlichen Matrixeinträgen sind in den ersten Feldern noch zusätzliche Informationen, wie Zeilen-, Spaltenanzahl und ein Symmetrieflag gespeichert. Diese Art der Speicherung ermöglicht eine einfache Serialisierung der Daten für das Netzwerksystem. Zusätzlich sind noch 320Bit freier Speicher verfügbar, dies hat zwei Gründe:

n	m	isSym	frei	Matrix
64Bit	64Bit	64Bit	320Bit	x Byte
$\overbrace{\qquad\qquad\qquad\qquad}^{512BitHeader}$				
$x = \begin{cases} 8 * m * n & nicht\, symmetrisch \\ 4 * n * (n-1) & symmetrisch \end{cases}$				

Tabelle 5.5: Interne Speicherstruktur der Matrix Klassen

1. Falls noch zusätzliche Informationen über das Netzwerk transportiert werden müssen, ist keine Änderung der Speicherstruktur nötig.
2. Die gesamten 512Bit müssen linear im Speicher ausgerichtet sein, um SSE/AVX/AVX512 verwenden zu können (siehe Kapitel 5.5.3 „Speicherausrichtung“). Da die SIMD-Routinen allerdings nur auf die Daten der Matrix selbst angewendet werden sollen, müssen die „Header-Daten“ von den SIMD-Routinen übersprungen werden. Dies ist nur möglich, wenn die entsprechende lineare Speicherausrichtung in der Größe der SIMD-Routine selbst (SSE:128Bit,AVX:256Bit,AVX512:512Bit) gewährleistet ist. Andernfalls würde das Programm abstürzen.

OpenMPMatrix: Die Klasse *OpenMPMatrix* beeinhaltet eine SSE/AVX-beschleunigte und threadparallelisierte Implementierung der Matrixoperationen. Die Parallelisierung erfolgt hierbei durch OpenMP [Dagum and Menon, 1998]. Alle Algorithmen sind im Rahmen dieser Arbeit umgesetzt worden und bieten für die meisten CPU-Architekturen eine hohe Effizienz.

MKLMatrix: Die Klasse *MKLMatrix* bietet alle Optimierungen der Intel-MKL-Bibliothek [Library, 2011] und ist somit nur für Intel-CPUs brauchbar. Die Bibliothek

gilt allgemein als äußerst effizient, besonders im Hinblick auf sehr große und komplexe Matrizenoperationen.

CudaMatrix: Die Klasse *CudaMatrix* bietet die Möglichkeit der Nutzung von Nvidia-GPUs [Nvidia, 2007] zur Beschleunigung der Matrixoperationen. Als Basis wurde die CuBLAS-Bibliothek von Nvidia verwendet, weil sie eine effiziente Umsetzung aller Level 1-3 BLAS Routinen [Lawson et al., 1979] auf modernen Nvidia-GPUs bietet. Nähere Informationen zu der genauen Implementierung der Klassen sind in [Küppers, 2016a] zu finden. Weiterhin wird die Verwendung von GPUs für die Beschleunigung der hier entwickelten Ersatzmodelle innerhalb von Optimierungen in Kapitel 5.5.9 diskutiert.

Der Aufbau und einige Methoden der „CudaMatrix“ und „MKLMatrix“ sind innerhalb der studentischen Arbeiten [Küppers, 2016b, Küppers and Schmitz, 2016] entwickelt worden. Diese Arbeiten wurden im Rahmen dieser Dissertation betreut und angeleitet. Die Methoden zur Rückwärtsdifferenzierung der Cholesky Zerlegung (siehe Abschnitt 5.5.6) und auch die OpenMP-Klasse sowie der gesamte Aufbau der Schnittstellenklasse „Matrix“ erfolgte innerhalb der vorliegenden Arbeit.

5.5.2 Verteiltes Rechnen

Für diese Arbeit wurde eine umfangreiche Bibliothek entwickelt, welche es ermöglicht, Matrixoperationen asynchron auf mehrere Rechner zu verteilen. Die Rechner können dabei unterschiedliche Architekturen besitzen, bspw. ist eine gleichzeitige Nutzung von Servern mit GPUs und konventionellen CPUs möglich.

Bei der Entwicklung wurden MPI-Bibliotheken vermieden, da zahlreichen inkompatiblen MPI-Umsetzungen existieren und MPI eine Synchronität der Netzwerkkommunikation voraussetzt. Die Synchronität war hierbei der wichtigste Punkt, da Optimierungen oftmals über Wochen laufen und es durchaus vorkommt, dass einzelnen Rechenknoten ausfallen. Verwendet man MPI, so fällt die gesamte Rechnung damit aus. Bei der hier vorgestellten asynchronen Umsetzung übernimmt ein anderer Knoten die Aufgaben des ausgefallenen Rechners.

Nach der anfänglichen Entwicklung wurde im Rahmen dieser Dissertation die Bachelorarbeit [Küppers, 2016b] betreut und angeleitet. In dieser wurde der Client um weitere Matrixoperationen erweitert, der Scheduler auf ein Round-Robin-Verfahren umgestellt, weitere Klassen serialisiert und die Fehlertoleranz erhöht (siehe auch [Küppers and Schmitz, 2016]).

Abbildung 5.9 zeigt den schematischen Aufbau der etablierten Client-/Server-Verbindungen. Der Client stellt das auszuführende Programm dar, welches die zu berechnenden Probleme beinhaltet. In diesem Fall handelt es sich um die hier entwickelte Kriging-Software, welche dafür die Client-Klasse implementieren muss. Die Client-Klasse wird im späterem Verlauf dieses Abschnitts erklärt. Auf der anderen Seite sind mehrere Server, die jeweils einen Dämon-Prozess starten, einen Netzwerk-Port öffnen und auf Aufgaben des Clients warten. Die Server-Programme können auf CPUs sowie GPUs rechnen und verbrauchen im Wartemodus praktisch keine Resourcen.

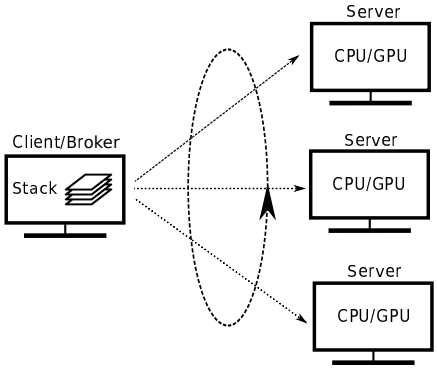


Abbildung 5.9: Schematische Darstellung der Client/Server Verbindungen und des Round-Robin Schedulers

Eine weitere Besonderheit ist, dass jeder Client auf jedem Server virtuelle Variablen/Objekte allokieren kann, z.B. eine Matrix. Hierfür werden die Daten auf Client-Seite serialisiert und übertragen, womit dem Server dann ein vollständiges Objekt mit eigenen Namen und allen zugehörigen Methoden sowie Attributen zur Verfügung steht. Die Methoden und Attribute können Client-seitig über ein eigenes Protokoll angesprochen und auf dem Server ausgeführt werden. Ergebnisse können dann in eine neues, virtuelles Objekt gespeichert werden, welches der Client wieder zurück übertragen kann. Voraussetzung dafür ist allerdings, dass die Klasse des Objekts das „SaveableOnServer“-Interface implementiert hat (siehe Kapitel 5.5.1). Steht auf Client-Seite eine Aufgabe an, wird sie in kleinere Teilaufgaben zerlegt und auf einem „Aufgabenstack“ abgelegt. Ein Round-Robin-Scheduler verteilt diese Aufgaben an die verbundenen Server, sofern diese bereit sind. Dann erledigen sie diese Teilaufgabe, senden das Ergebnis an den Client zurück und stehen danach für neue Aufgaben zur Verfügung. Der Client sammelt alle Teillösungen, setzt diese am Ende zu einer Gesamtlösung zusammen und kann anschließend die Server beenden oder im Wartemodus belassen. Im folgenden sollen einige Begrifflichkeiten und Funktionen der hier vorgestellten Netzwerkbibliothek erläutert werden:

Netzwerkobjekte: Alle Objekte, die potenziell über das Netzwerk übertragen werden und als virtuelle Variable auf Servern zur Verfügung stehen, müssen das Interface „SaveableOnServer“ implementieren. Dieses erzwingt folgende Methoden, die die entsprechende Klasse bereitstellen muss:

- getSerializedData: Stellt einen typlosen Datenstrom welcher die Daten des Objekts beschreibt bereit. Tabelle 5.5 zeigt einen solchen Datenstrom.
- setSerializedData: Ein leeres Objekt kann über diese Methode einen Datenstrom erhalten und deserialisieren. Dieser Datenstrom wird dann wieder zerlegt und in die entsprechenden Attribute kopiert oder verschoben.
- getBaseType: Der Ursprungstyp des Objekts (z.B. Matrix) als String.
- freeMem: Der belegte Speicher des Objekts wird vollständig geleert.

Die Serialisierung und Deserialisierung ist in diesem Fall über das Interface sichergestellt. Die korrekte Implementierung der (De-)Serialisierung aber nicht, da diese Art der Zusicherung (z.B. durch Design by Contract) mit der Programmiersprache C++ nicht möglich ist. Im Falle einer korrekten Implementierung ist es mit diesem System möglich, vollständige Objekte über das Netzwerk zu verschicken. Das besondere dabei ist, dass diese Objekte auf dem Server typunabhängig gespeichert werden und dadurch sehr flexibel miteinander interagieren können.

Client: Der Client ist das ausführende Programm, in diesem Fall das Kriging selbst. Dieses stellt die Aufgaben bereit und muss die entsprechenden Client-Klassen implementieren. Das UML-Diagramm dieser ist in Anhang A.3.9 zu finden. Die Struktur besteht im Wesentlichen aus einer „ZMQClient“-Klasse, welche alle Verbindungen und Kommunikation mit den Servern verwaltet. Weiterhin gibt es eine „ClientFunctions“-Klasse, die die aufrufbaren Methoden wie z.B. die Matrix Multiplikation usw. beinhaltet. Ein Anwender muss für die Verwendung nur die gewünschte Methode aufrufen, die Kommunikation und Verteilung laufen dann im Hintergrund ab.

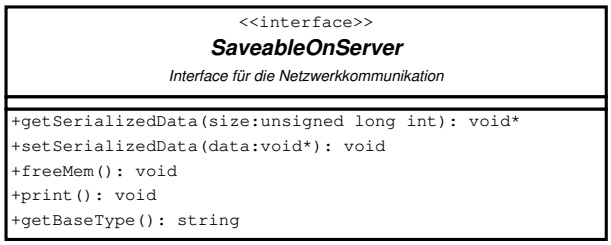


Abbildung 5.10: UML Diagramm des SaveAbleOnServer Interface

Server: Die hier beschriebene Server-Software ist ein eigenständiges Programm, welches auf einem Rechner ausgeführt wird und dann auf Aufgaben wartet. Der Aufbau der zugehörigen Klassen ist in Anhang A.3.15 dargestellt. Kern des Programms ist die „ZMQServer“-Klasse, welche zwei Verbindungen enthält: Eine zur Kommunikation mit dem Client und eine Ping-Verbindung, welche in regelmäßigen Abständen Netzwerk-Pings sendet und dem Client dadurch mitteilt, dass der Server noch läuft.

Weiterhin stellt der „ZMQServer“ die Verwaltung der virtuellen Variablen bereit. Hierfür ist ein assoziativer Container in Form einer „Map“ namens „vars“ vorgesehen. Dieser bildet einen Variablennamen auf ein „SaveableOnServer“-Objekt ab. Dadurch können alle Objekte mit dem Interface SaveableOnServer innerhalb dieses Containers gespeichert werden.

Die Klasse „MatrixServerFunctions“ stellt alle Methoden bereit, welche vom Client benötigt werden und hat Zugriff auf den Variablen-Container. Beispielsweise die Berechnung einer Matrix Addition. Um die Typkonvertierungen zu verdeutlichen, wird folgend eine Addition der Matrizen „mat1“ und „mat2“ auf Serverseite beschrieben:

1. Der Client wandelt die beiden Matrizen in „SaveableOnServer“-Objekte um, was nur dann möglich ist, wenn das Interface „SaveableOnServer“ implementiert wurde. Diese Objekte können dann serialisiert und übertragen werden.
2. Der ZMQServer erhält diese Objekte und speichert sie in dem Container „vars“ mit dem vorgesehenen Namen.
3. Der Client ruft über das Netzwerkprotokoll den Befehl auf, die Netzwerk-Matrizen „mat1“ und „mat2“ zu addieren und in der Netzwerk-Matrix „mat3“ zu speichern.
4. Der Server erhält den Befehl und ruft die „matrixAddition“-Methode auf. Als Parameter werden nur die Strings „mat1“,“mat2“ und „mat3“ übergeben. Zusätzlich können noch die Werte „from“ und „to“ belegt werden, welche den Index der zu addierenden Zeilen beschreibt. Damit könnte auch nur ein Teil der Matrix addiert werden.
5. Die Funktion führt eine Typumwandlung auf die Einträge „mat1“ und „mat2“ des „vars“-Containers durch und wandelt diese in Objekte vom Typ „Matrix“ um. Diese können dann addiert und in einem neuen Matrix-Objekt „mat3“ gespeichert werden. Das Objekt „mat3“ wird wieder in ein „SaveableOnServer“-Objekt um-

gewandelt und kann dann in dem „vars“-Container unter dem Namen „mat3“ gespeichert werden.
6. Der Client kann nun das „SaveableOnServer“-Objekt „mat3“ zurück übertragen, in ein Objekt vom Typ „Matrix“ umwandeln und erhält so das fertige Ergebnis.

Messaging Middleware ZeroMQ: ZeroMQ ist eine asynchrone Messaging-Bibliothek, welche eine hocheffiziente Nachrichtenübermittlung anbietet. ZeroMQ ist für Problemstellungen entwickelt worden, welche eine sehr schnelle Datenübertragung mit kurzen Verzögerungen benötigen. Für weitere Informationen sei auf [Dworak et al., 2011, Hintjens, 2013] verwiesen.

Exemplarischer Ablauf einer Matrix-Addition: Folgend wird ein exemplarischer Ablauf einer Matrix-Addition auf zwei Servern abgebildet, um die Prozessinteraktion zu verdeutlichen:

1. Die Server-Software muss auf den entsprechenden Rechnern gestartet werden.
2. Der Client wird gestartet und belegt lokal die zu addierenden Matrizen.
3. Der Client verbindet sich mit beiden Servern.
4. Der Client startet für jeden Server einen Überwachungsthread, der in regelmäßigen Abständen einen Ping sendet und so überprüft, ob der jeweilige Server noch verfügbar ist. Ist dies nicht der Fall, wird der Server von der Liste verfügbarer Rechner gestrichen und die laufende Aufgabe an einen anderen Server weitergegeben.
5. Der Client startet die Netzwerk-Matrix-Addition mit einem Befehl.
   (a) Der Client zerlegt die Addition in mehrere Teilmatrix-Additionen.
   (b) Der Client allokiert alle nötigen Teilmatrizen auf jedem Server und übersendet die serialisierten Daten.
   (c) Alle Teil-Matrix-Additions-Befehle werden auf den Stack vom Client gelegt.
   (d) Der Client startet den Scheduler, der die Aufgaben des Stacks nimmt und an die Server sendet.
   (e) Jeder Server addiert die jeweiligen Teilmatrizen und speichert die Ergebnisse lokal in eine neue Matrix. Diese wird anschließend in den Typ „SaveableOnServer“ konvertiert.
   (f) Jeder Server sendet danach die Teilergebnisse serialisiert zurück und ist wieder bereit für die nächste Aufgabe.
   (g) Die Schritte b-f werden solange wiederholt, bis der Stack leer ist.
6. Hat der Client alle Teilergebnisse, wandelt er diese in Matrix-Objekte um und setzt die Teilmatrizen zu einer Gesamtmatrix zusammen.

Geschwindigkeit bei mehreren Servern: Bei einem verteilten System leidet die Geschwindigkeit der Operationen oft erheblich, da für die Kommunikation und die Verwaltung der Teilberechnungen ein großer Anteil der Rechenzeit aufgewendet wird. Die Anzahl der Server muss an die Problemgröße angepasst sein. Abbildung 5.11 zeigt die Messung des zeitlichen Faktors eines Kriging-Trainings bei steigender Server-Anzahl und unterschiedlichen Problemgrößen. Das Training wurde auf dem DLR-AT-Cluster ausgeführt (siehe Kapitel 6.2.1). Als Testfall dienten die zufälligen Stützstellen der in

Anhang A.4.1 beschriebenen ZDT3-Funktion mit 20 freien Variablen. Eine Anzahl an Servern von 0 stellt ein lokales Training dar und eine Anzahl von 1 eine 1:1 Verbindung.

Man kann sehen, dass der Verlust einer 1:1 Verbindung sehr klein ist, womit die Verzögerung durch die Netzwerkkommunikation als gering einzustufen ist. Bei einer Anzahl von 2-3 Servern ist der Verlust ab 2000 Stützstellen ebenfalls relativ gering. Mit zunehmender Anzahl an Servern nimmt der Verlust ebenfalls zu. Dieses Verhalten ist zu erwarten, da die Problemstellung für eine so hohe Anzahl an leistungsstarken Rechnern noch zu klein ist. Dennoch lässt sich bei 7 Servern eine maximale Beschleunigung des Faktors 5 erzielen und somit können 71% der Rechenleistung effektiv genutzt werden. Bedenkt man weiterhin die Flexibilität und Ausfallsicherheit des Client-/Server-Systems, so ist der Einsatz bei entsprechend großen Problemen sehr lohnenswert.

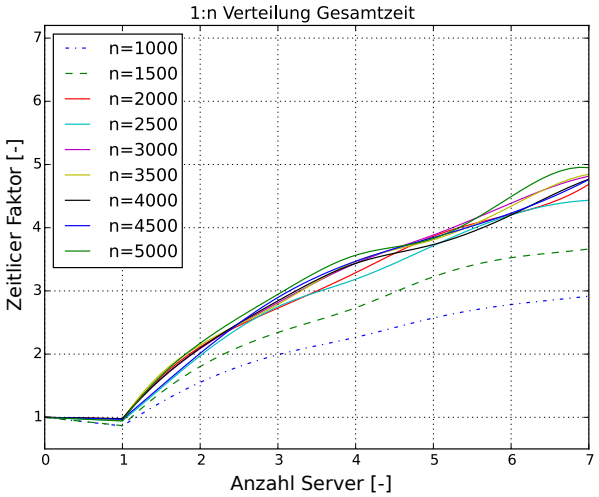


Abbildung 5.11: Beschleunigungsfaktor eines Kriging-Trainings mit n Stützstellen im Vergleich zu einer Einzelrechnung bei Nutzung von mehreren Servern. Quelle: [Küppers and Schmitz, 2016, Küppers, 2016b]

5.5.3 Effiziente Berechnung der Kovarianzfunktion

Die in Kapitel 5.1 beschriebenen Kovarianzfunktionen und deren Ableitungen werden während des Trainings sehr häufig ausgeführt. Für jeden Trainingsschritt mit einem Dichtefunktionsaufruf und den dazugehörigen partiellen Ableitungen ergeben sich folgende Anzahl an Aufrufen der Kovarianzfunktion:

1. Berechnung der Dichtefunktion: $\frac{n(n+1)}{2}$ Aufrufe der Kovarianzfunktion
2. Berechnung von o partiellen Ableitungen der Likelihood-Funktion: $o\frac{n(n+1)}{2}$

Geht man von einem üblichen Beispiel mit 1000 Stützstellen und 80 Hyperparametern aus, so ergeben sich für die Dichtefunktion ca. 500.000 Aufrufe, bzw. 80x500.000 Aufrufe für die partiellen Ableitungen. Insgesamt sind dies ca. 40.5 Millionen Aufrufe für eine Iteration. Das bedeutet, dass jeder gesparte Befehl innerhalb des Kovarianzfunktionsaufrufs sehr stark auf die Laufzeit auswirkt. Folgend sind einige Möglichkeiten zur Beschleunigung der Berechnung aufgelistet:

- Exponentialfunktion der Hyperparameter e^{θ_l} im Voraus berechnen
- Gauss-Kovarianzfunktion: Puffern von Zwischenergebnissen
- Standard Implementierung der Exponentialfunktion durch eine schnellere Version ersetzen
- CPU-Befehlssätze SSE/AVX verwenden

Exponentialfunktion im Voraus berechnen: Da die Hyperparameter für eine Iteration konstant bleiben, wird die Berechnung von e^{θ_l} (siehe Gleichung 5.9) nur einmal zu Anfang der Iteration durchgeführt und dann gepuffert.

Puffern von Zwischenergebnissen: Da für die Aufstellung der Kovarianzmatrix der innere Teil $e^{-\frac{1}{2}\sum_{l=1}^{l<=k}\left(e^{\theta_l}|\triangle x_l|^p\right)}$ (siehe Gleichung 5.10) in den Gleichungen für die partiellen Ableitungen der Kovarianzfunktion nach den Hyperparametern (siehe Gleichungen 5.12, 5.13) derselbe ist, ist es effizient diesen zu puffern und bei Bedarf wiederzuverwenden. Der Vorteil liegt darin, dass ein Abruf aus dem RAM deutlich schneller ist als die erneute Berechnung der Exponentialfunktion.

Standardimplementierung der Exponentialfunktion austauschen: Da die Implementierung der Exponentialfunktion der GCC-Bibliothek (Stand GCC v4.9) relativ langsam ist, diese Funktion aber mehrere Millionen Mal aufgerufen wird, ist es sinnvoll, eine schnellere Implementierung zu verwenden. In [Shigeo, 2011] wird eine AVX-beschleunigte Implementierung der Exponentialfunktion beschrieben, welche fünfmal schneller ist als die GCC-Version. Der Fehler der berechneten Werte ist vergleichbar mit der GCC-Version.

SSE und AVX - SIMD Instruktionen: Die Streaming-SIMD-Extensions (SSE) sind eine von Intel entwickelte Befehlssatzerweiterung der x86-Architektur (siehe [Raman et al., 2000]). Aufgabe der SSE-Befehle ist es, Programme durch Parallelisierung auf Instruktionslevel zu beschleunigen. Diese Art der Parallelisierung wird auch SIMD (Single Instruction Multiple Data) genannt und ist der Vorgehensweise der massiv parallelen Berechnung auf Grafikkarten ähnlich. Die SSE-Befehlssatzerweiterung umfasst ursprünglich 70 Instruktionen und 8 neue Register, genannt XMM0 bis XMM7.

Es gibt zahlreiche Umsetzungen der SSE-Befehle. Diese reichen von SSE-Version 1 bis 5, wobei ab SSE-Version 3 AMD und Intel jeweils eigene Implementationen dieser Architektur vornehmen. Der momentane Nachfolger von SSE heißt AVX (Advanced Vector Extensions) und wird wieder gemeinsam von AMD und Intel verfolgt. Die Register wurden auf 16x 256 Bit (AVX Version 1) erweitert und jeder Prozessorkern besitzt eigene AVX-Einheiten. Weiterhin kann jede AVX-Einheit (je nach CPU-Typ) zusätzlich eine FMA-Operation (fused multiply add) in der Form $a \leftarrow a(b + c)$ in nur einem Takt ausführen, wodurch die theoretische Rechenleistung nochmals ansteigt. Der Parallelisierungsgrad und damit auch die Performance aktueller CPUs wird heutzutage also nicht nur über die Anzahl der Kerne und deren Kernfrequenz bestimmt, sondern maßgeblich durch die Breite, Frequenz und dem Befehlssatz der SIMD-Einheiten. Da diese Operationen allerdings nicht in jeder Anwendung effizient genutzt werden können, macht dies die Auswahl von modernen CPUs zu einer sehr komplexen und anwendungsspezifischen Angelegenheit.

Um diese Funktionen zu nutzen, müssen im C++-Code spezielle SSE Befehle verwendet werden [Fog, 2013, Library, 2011]. Das folgende Listing zeigt die Umsetzung der Gauss-Korrelationsfunktion mit SSE-Befehlen, wobei die Parallelisierung hier über die Hyperparameter gemacht wird. *Auf die Anwendung von Pseudocode wird in diesem*

```
1  __m128d correlSSE=0.0
2  __m128d thetasExpSSE, point1SSE, point2SSE, pointDiffSSE
3
4  array thetasExpArray
5  array point1Array
6  array point2Array
7
8  for(i=0; i<point1.getNumVars()-1 ; i+=2)
9          thetasExpSSE =_mm_load_pd(&(thetasExpArray[i]))
10         point1SSE =_mm_load_pd(&(point1Array[i]))
11         point2SSE =_mm_load_pd(&(point2Array[i]))
12
13         pointDiffSSE = _mm_sub_pd(point1SSE,point2SSE)
14         pointDiffSSE = _mm_mul_pd(pointDiffSSE,pointDiffSSE)
15         pointDiffSSE = _mm_mul_pd( thetasExpSSE, pointDiffSSE  )
16         correlSSE = _mm_add_pd( correlSSE,  pointDiffSSE )
17
18
19 correlSSE = _mm_hadd_pd(correlSSE,correlSSE)
20 _mm_store_sd(&correl,correlSSE)
21 for(; i<point1.getNumVars() ; i++)
22         correl += thetasExp[0][i] * (point1.getVar(i) - point2.getVar(i))^2
23
24 correl=e^(-0.5*correl)
```

Fall, aufgrund der sehr speziellen SSE-Befehle, weitestgehend verzichtet. Diese Methode wird zur Berechnung der Einträge der Kovarianzmatrix verwendet und dementsprechend oft aufgerufen.

In den Zeilen 1-2 werden verschiedene Variablen vom Typ "__m128d" definiert, welcher einen 128-Bit großen SSE-Datentyp darstellt und zwei 64-Bit Werte aufnehmen kann. Zudem wird die Variable correlSSE mithilfe der Funktion_mm_setzero_pd() auf 0 gesetzt. Das Array thetasExpArray in Zeile 4 beinhaltet die berechneten Hyperparameter e^{θ_l}. Da diese Werte für alle Einträge in der Kovarianzmatrix identisch sind, werden sie daher vor dem Belegen der Kovarianzmatrix berechnet. Danach werden die beiden Arrays in Zeile 5-6 initialisiert, welche die Ortsvariablen $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^k$ beinhalten. Die darauffolgende for-Schleife iteriert über die Anzahl an freien Variablen. Der Zähler wird hier immer um den Wert zwei erhöht, da mit den SSE-Routinen zwei 64-Bit-Werte gleichzeitig aus dem Speicher in die Register geladen und berechnet werden. Dies hat zufolge, dass der Speicher eine 128-Bit-Ausrichtung besitzen muss. In den Zeilen 9-11 werden 128-Bit aus den Arrays an der Stelle i in die SSE-Register der CPU übertragen.

Einschub: Speicherausrichtung *In aktuellen C++ Compilern wird eine Speicherausrichtung von 128-Bit gewährleistet. Tabelle* 5.6 *zeigt diese Speicherausrichtung und die damit entstehende Problematik beim Programmieren. Die erste Zeile der Tabelle beschreibt den Index eines normalen Arrays mit 6 Einträgen und jeder Eintrag hat die Größe eines 64-Bit-Werts. Der Compiler garantiert in diesem Fall einen zusammenhängenden Speicher von 128-Bit, dargestellt durch die dritte Zeile.* Die Zeilen 13-16 stellen die eigentliche Berechnung in den SSE-Registern der CPU dar. In Zeile 19 werden die beiden Werte in den Registern zusammenaddiert und dann in Zeile 20 wieder zurück in den RAM kopiert.

0	1	2	3
64Bit	64Bit	64Bit	64Bit
$\underbrace{128Bitaligned}$			

Tabelle 5.6: 128Bit Speicherausrichtung

Ist die Anzahl der Variablen nicht durch zwei teilbar, so bleibt ein Rest bestehen. Dieser wird in den Zeilen 21-23 auf konventionelle Weise berechnet.

Die gemessenen Geschwindigkeitsvorteile durch SSE4 für diese Methode liegen bei ca. 30 %.

5.5.4 Reduktion von Stützstellen

Während einer Optimierung können eine Vielzahl an Stützstellen erzeugt werden und damit große Kovarianzmatrizen entstehen. Um Rechenzeit zu sparen, können Stützstellen aussortiert werden, welche dem jeweiligen Ersatzmodell keinen oder nur einen geringen Informationszugewinn bringen. Als notwendiges Maß bietet sich die Verwendung der Korrelationsfunktion (siehe Kapitel 5.1) an. Stützstellen mit sehr hohen Korrelationen haben einen kleinen Abstand zueinander und sollten dem Ersatzmodell nur einen geringen Informationsanteil liefern. Daher kann eine der beiden im Training außer Acht gelassen werden. Allerdings macht dieses Vorgehen nur Sinn, wenn die Hyperparameter des jeweiligen Kriging-Modells bereits gut geschätzt sind. Um dies zu gewährleisten, wird die Reduktion der Stützstellen nur durchgeführt, wenn die in Kapitel 5.3.3 beschriebene Initialisierungsoption gewählt wurde und sich die Initialisierung bereits im „*Restart*“-Modus (siehe Kapitel 5.3.3) befindet.

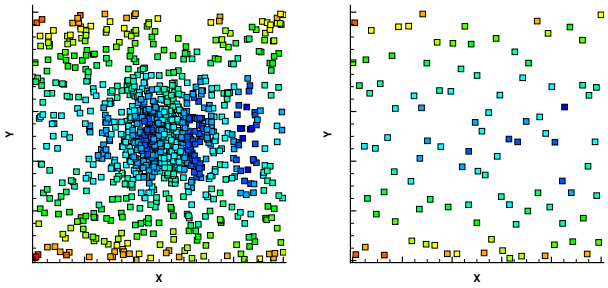


Abbildung 5.12: Anwendung des Filters anhand eines analytischen Beispiels. Die Färbung zeigt den Funktionswert an.

Folgend wird der verwendete Algorithmus vereinfacht dargestellt:

```
1  korrMap = AssoziativerContainer[float][paar(int,int)]
2
3  schleife p1 über alle points
4      schleife p2=p1.next() über alle points
5          korrelation = correlation(p1,p2)
6          korrMap[korrelation] =paar(p1,p2)
7
8
9
10 pointsToDelete =[]
11 schleife korrPair über alle korrMap
12     wenn (korrMap.p1 in pointsToDelete) oder (korrMap.p2 in pointsToDelete)
13         überspringe
14     deleteIndices.append(random(p1,p2))
15 }
16
17 schleife deletePoint über alle deleteIndices
18     points.remove(deletePoint)
```

In den Zeilen 3-8 werden die Korrelationen für alle möglichen Punktepaare berechnet und innerhalb der Map direkt sortiert. Die Schleife in den Zeilen 11-15 startet bei dem größten Korrelationswert und läuft durch die sortierte Map. Innerhalb der Schleife wird geprüft, ob einer der beiden Punkte bereits zum Löschen markiert wurde. Ist dies der Fall, so wird nichts weiter unternommen. Ansonsten wird dieser zum Löschen einge-

tragen. Auf diese Weise kann sichergestellt werden, dass nicht beide Punkte gelöscht werden. In Abbildung 5.12 wird ein Beispiel auf Basis der ZDT3-Funktion mit zwei Variablen gezeigt (siehe Kapitel A.4.1). Auf dem linken Bild wurden 500 gleichverteilte und nochmals 500 normalverteilte Stützstellen erzeugt, wobei die normalverteilten Stützstellen um die Mitte des Bereichs gelegt wurden. Damit wird eine lokale Anhäufung von Stützstellen simuliert.

Daraufhin wurde ein Ordinary-Kriging trainiert und die Stützstellen mithilfe des Filters auf 100 reduziert. Das Ergebnis ist in Abbildung 5.12 rechts zu sehen. Die lokale Anhäufung in der Mitte ist nicht mehr sichtbar und die Verteilung der Punkte ist sehr gleichmäßig. Weiterhin ist der Algorithmus sehr schnell, sodass die Laufzeit des Filters im Vergleich zur restlichen Laufzeit zu vernachlässigen ist.

5.5.5 Likelihood: Inverse durch Gleichungssysteme ersetzen

Mithilfe der Cholesky-Zerlegung (siehe Anhang A.3.16) können symmetrisch, positiv definite lineare Gleichungssysteme effizient gelöst werden. So kann bei der Likelihood Berechnung auf die direkte Bestimmung der Inversen verzichtet werden.

Das Maximum des Likelihoodterms (siehe 5.37) sieht wie folgt aus:

$$\max_{\vec{h}} \left(\log(L)\right) = \max_{\vec{h}} \left(-\log\left(\det(\mathbf{Cov})\right) - \left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1} \left(\vec{y}_s - {}_1\overrightarrow{F}\right)\right)$$

Typischerweise werden folgende Schritte zur Berechnung dieses Terms durchgeführt:

1. Die Berechnung der Cholesky Zerlegung von $\mathbf{Cov}$ mit $\frac{1}{6}n^3 + \mathcal{O}(n^2)$ Gleitkomma-Operationen (Quelle [Press et al., 2007]), wobei $\mathbf{Cov} \in \mathbb{R}^{n \times n}$. Die Determinante kann direkt mithilfe der Cholesky-Zerlegung bestimmt werden.
2. Die Vor- und Rückwärtssubstitution mit n^3 Gleitkomma-Operationen, zur Bestimmung der Inversen Kovarianzmatrix $\mathbf{Cov}^{-1}$ um den quadratischen Term $\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1} \left(\vec{y}_s - \overrightarrow{F}\right)$ zu berechnen.

Der quadratische Term kann allerdings auch ohne die Vor- und Rückwärtssubstitution, mithilfe eines einfachen Gleichungssystems gelöst werden. Die genaue Berechnung wird in Anhang A.3.17 gezeigt. Damit ergibt sich folgender Aufwand:

1. Die Berechnung der Cholesky-Zerlegung mit $\frac{1}{6}n^3 + \mathcal{O}(n^2)$ Gleitkomma-Operationen (Quelle [Press et al., 2007]), wobei $\mathbf{Cov} \in \mathbb{R}^{n \times n}$
2. Die Lösung von zwei Gleichungssystemen mit $2n^2$ Gleitkomma-Operationen

Der zweite Schritt ist in diesem Fall deutlich günstiger. Allerdings wird die Inverse bei der Berechnung der partiellen Ableitungen der Likelihood Funktion benötigt. Eine effiziente Lösung für dieses Problem wird in Kapitel 5.5.6 aufgezeigt.

5.5.6 Ableitung Likelihood: Verzicht auf Vor- und Rückwärtssubstitution

Innerhalb dieses Abschnitts wird eine Methode gezeigt, welche die Berechnung der partiellen Ableitung des Likelihood-Terms $\frac{\partial L(\vec{h})}{\partial h_l}$ (Definition siehe Gleichung 5.38) beschleunigt. Diese Beschleunigung ist besonders im Falle des Co-Krigings von Vorteil, da dort eine große Anzahl an Hyperparametern vorhanden ist. Um das Beschleunigungsverfahren zu erläutern, werden zuerst die notwendigen Schritte zur Bestimmung der partiellen Ableitungen des Likelihood Terms gezeigt:

1. Über alle benötigen Hyperparameter h_l
   (a) Bestimmung der Ableitung der Kovarianzmatrix: $\frac{\partial \mathbf{Cov}}{\partial h_l}$
   (b) Berechnung der Ableitung der quadratischen Form (siehe Gleichung 5.37): $\left(\vec{y}_s - \beta_1 \overrightarrow{F}\right)^T \mathbf{Cov}^{-1} \frac{\partial \mathbf{Cov}}{\partial h_l} \mathbf{Cov}^{-1} \left(\vec{y}_s - \beta_1 \overrightarrow{F}\right)$
   (c) Berechnung der Ableitung der Determinante (siehe Gleichung 5.37): $\mathrm{Spur}\left(\mathbf{Cov}^{-1} \frac{\partial \mathbf{Cov}}{\partial h_l}\right)$

Für Punkt (a) muss jeweils die symmetrische Matrix $\frac{\partial \mathbf{Cov}}{\partial h_l}$ aufgestellt werden, die Komplexität des Algorithmus liegt bei $\mathcal{O}(n^2)$ und kann gut parallelisiert werden.

Punkt (b) ist vom Aufwand her nahezu vernachlässigbar, da der Vektor $\left(\vec{y}_s - \beta_1 \overrightarrow{F}\right)^T \mathbf{Cov}^{-1}$ bereits in der Likelihood-Berechnung bestimmt wird und damit nur noch eine Vektor-Matrixmultiplikation bleibt.

Punkt (c) ist der aufwendigste Teil, da die inverse Kovarianzmatrix direkt bestimmt werden muss und damit die Vorwärts- und Rückwärtssubstitution der Invertierung notwendig wird (siehe Kapitel 5.5.5).

Der genaue Ablauf zur Bestimmung der Inversen folgt dem Schema aus Kapitel A.3.16 und besteht aus zwei Schritten:

1. Cholesky-Zerlegung der Kovarianzmatrix
2. Vorwärts- und Rückwärtssubstitution der Standardeinheitsbasis zur Bestimmung der Inversen (siehe [Press et al., 2007] S. 100)

Der Aufwand beider Schritte liegt bei

1. $\frac{1}{6}n^3$ Multiplikationen/Additionen, $\frac{1}{2}n^2$ Divisionen, n Wurzeloperationen
2. Vorwärts- und Rückwärtseinsetzen insgesamt: n^3 Multiplikationen/Additionen

Der Hauptaufwand liegt bei der Vorwärts- und Rückwärtssubstitution, womit eine alternative Berechnung von Punkt (c) von besonderem Interesse ist. Eine Möglichkeit bietet die algorithmische Differentiation im Rückwärtsmodus (siehe [Mader et al., 2008, Griewank and Walther, 2008]). Ein solcher Ansatz für das Kriging-Verfahren kann in [Toal et al., 2009] gefunden werden und bietet die Möglichkeit auf die Vor- und Rückwärtssubstitution zu verzichten, allerdings werden rückwärtsdifferenzierte Algorithmen des Cholesky-Algorithmus und auch der Vor- und Rückwärtssubstitution benötigt.

Aus diesem Grund wählen die Autoren in [Toal et al., 2011] einen Algorithmus, welcher die Invertierung zwar benötigt, aber keine rückwärtsdifferenzierten Algorithmen. Ein Geschwindigkeitsvorteil wird dadurch erreicht, dass die partiellen Ableitungen der Kovarianzmatrix $\frac{\partial \mathbf{Cov}}{\partial h_l}$ nicht mehr direkt aufgestellt werden müssen. Die Vorwärts- und

Rückwärtssubstitution muss dennoch durchgeführt werden und stellt den Hauptanteil bei der Berechnung des Likelihoods und der Ableitungen dar. Weiterhin muss für jede vorhandene Kovarianzmodellfunktion eine eigene softwaretechnische Implementierung vorgenommen werden. Dadurch wird der Entwicklungsaufwand erhöht und die Fehleranfälligkeit steigt.

Um diese Probleme zu umgehen, wurde im Rahmen dieser Arbeit eine weitere Möglichkeit entwickelt: die alleinige Rückwärtsdifferentiation der Determinante der Kovarianzmatrix. Dieser Ansatz bietet nicht nur einen Geschwindigkeitsvorteil sondern kommt mit nur kleinen Änderung des Quellcodes aus und ist für alle Kriging-Verfahren wie auch Kovarianzfunktionen gültig. Es wird allerdings eine rückwärtsdifferenzierte Version des Cholesky Algorithmus benötigt. Für diese bestehen aber mittlerweile sehr effiziente BLAS-Implementierungen, welche am Ende des Abschnitts vorgestellt werden.

Der grundlegende Ansatz ist die Bestimmung der folgenden Ableitung:

$$\frac{\partial \left(ln \left(det \left(\mathbf{Cov} \left(h_l \right) \right) \right) \right)}{\partial h_l} \tag{5.47}$$

Wobei die Determinante das Produkt über alle quadrierten Diagonalelemente der choleskyzerlegten Dreiecksmatrix LL^T ist:

$$ln \left(det \left(\mathbf{Cov} \left(h_l \right) \right) \right) = ln \left(\prod_i L_{i,i}^2 \right) = 2 \sum_i ln \left(L_{i,i} \right) \tag{5.48}$$

Daraus ergibt sich die folgende Verkettung, wobei f_{chol} die Cholesky-Zerlegung darstellt:

$$\frac{\partial \left(ln \left(det \left(f_{chol} \left(\mathbf{Cov} \left(h_l \right) \right) \right) \right) \right)}{\partial h_l} \tag{5.49}$$

Die bestehenden Abbildungen sehen wie folgt aus:

$$\begin{aligned} \mathbf{Cov} &: \mathbb{R} \longmapsto \mathbb{R}^{n^2} \\ f_{chol} &: \mathbb{R}^{n^2} \longmapsto \mathbb{R}^{n^2} \\ det &: \mathbb{R}^{n^2} \longmapsto \mathbb{R} \\ ln &: \mathbb{R} \longmapsto \mathbb{R} \end{aligned} \tag{5.50}$$

Der Logarithmus der Determinante wird als f_{lndet} zusammengefasst:

$$\begin{aligned} &\frac{\partial \left(f_{lndet} \left(f_{chol} \left(\mathbf{Cov} \left(h_l \right) \right) \right) \right)}{\partial h_l} \\ &f_{lndet} : \mathbb{R}^{n^2} \longmapsto \mathbb{R} \end{aligned} \tag{5.51}$$

Es gilt also:

$$f_{lndet} \circ f_{chol} \circ \mathbf{Cov} : \mathbb{R} \longmapsto \mathbb{R} \tag{5.52}$$

Daraus resultieren die Jacobi Matrizen D_{cov} der Größe $n^2 \times 1$, D_{chol} der Größe $n^2 \times n^2$

und D_{flndet} der Größe $1 \times n^2$ wobei $\mathbf{C_{i,j}}$ einen Eintrag der Matrix $\mathbf{Cov}$ bedeutet. Damit wird die gesamte Ableitung zu:

$$\frac{\partial \left(f_{lndet} \left(f_{chol} \left(\mathbf{Cov} \left(h_l \right) \right) \right) \right)}{\partial h_l} = D_{flndet} D_{chol} D_{cov} \tag{5.53}$$

$$= \sum_i \sum_{j<=i} \sum_k \sum_{m<=k} \frac{\partial f_{lndet}}{\partial \mathbf{L_{i,j}}} \frac{\partial \mathbf{L_{i,j}}}{\partial \mathbf{C_{k,m}}} \frac{\partial \mathbf{C_{k,m}}}{\partial h_l} \tag{5.54}$$

Wesentlich ist es nun, die gezeigte mehrdimensionale Kettenregel möglichst effizient zu bestimmen. Wird die vollständige Summe berechnet, dann wäre die Komplexität für die Berechnung der Ableitung bei $\mathcal{O}\left(n^4\right)$. Der Aufwand wäre also deutlich größer als über die Bestimmung der Inversen. Benötigt wird demnach ein Algorithmus, welcher die notwendigen Terme der mehrdimensionalen Kettenregel ohne große Matrizen berechnen kann. Hierfür werden folgend die einzelnen Jacobi Vektoren betrachtet. Der Vektor D_{cov} entspricht allen Einträgen der Matrix $\frac{\partial \mathbf{Cov}}{\partial h_l}$ und ist daher bekannt. Als nächstes kann der Vektor D_{flndet} bestimmt werden, da die Funktion f_{lndet} bekannt ist:

$$D_{flndet}^T = \begin{pmatrix} \frac{\partial f_{lndet}}{\partial \mathbf{L_{1,1}}} \\ \frac{\partial f_{lndet}}{\partial \mathbf{L_{2,1}}} \\ \vdots \\ \frac{\partial f_{lndet}}{\partial \mathbf{L_{n-1,n}}} \\ \frac{\partial f_{lndet}}{\partial \mathbf{L_{n,n}}} \end{pmatrix} = \begin{pmatrix} \frac{\partial 2 \sum ln(L_{i,i})}{\partial \mathbf{L_{1,1}}} \\ \frac{\partial 2 \sum ln(L_{i,i})}{\partial \mathbf{L_{2,1}}} \\ \vdots \\ \frac{\partial 2 \sum ln(L_{i,i})}{\partial \mathbf{L_{n-1,n}}} \\ \frac{\partial 2 \sum ln(L_{i,i})}{\partial \mathbf{L_{n,n}}} \end{pmatrix} = \begin{pmatrix} \frac{2}{L_{1,1}} \\ 0 \\ \vdots \\ 0 \\ \frac{2}{L_{n,n}} \end{pmatrix} \tag{5.55}$$

Der Vektor lässt sich auch als Diagonalmatrix interpretieren und kann mit wenig Zeitaufwand aus der Cholesky-zerlegten Matrix bestimmt werden:

$$\bar{D}_{flndet} = \begin{bmatrix} \frac{2}{L_{1,1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{2}{L_{n,n}} \end{bmatrix} \tag{5.56}$$

Der Großteil des Jacobi-Vektors besteht aus Null-Einträgen, wodurch sich Gleichung 5.53 stark vereinfachen lässt, da $\frac{\partial f_{lndet}}{\partial \mathbf{L_{i,j}}} \neq 0 \forall i = j$:

$$\frac{\partial \left(f_{lndet} \left(f_{chol} \left(\mathbf{Cov} \left(h_l \right) \right) \right) \right)}{\partial h_l} = \sum_i \sum_k \sum_{m<=k} \frac{\partial f_{lndet}}{\partial \mathbf{L_{i,i}}} \frac{\partial \mathbf{L_{i,i}}}{\partial \mathbf{C_{k,m}}} \frac{\partial \mathbf{C_{k,m}}}{\partial h_l} \tag{5.57}$$

Die Komplexität liegt somit nur noch bei $\mathcal{O}\left(n^3\right)$. Als nächster Schritt muss der mittlere Teil der Kettenregel ($\frac{\partial \mathbf{L_{i,i}}}{\partial \mathbf{C_{k,m}}}$) bestimmt werden. Hierfür gibt es zwei Möglichkeiten:

1. Man geht von den Werten $\frac{\partial \mathbf{C_{k,m}}}{\partial h_l}$ aus und bestimmt ausgehend von diesen die Einträge von $\frac{\partial \mathbf{L_{i,i}}}{\partial h_l}$ über eine vorwärtsdifferenzierte Cholesky Zerlegung
2. Man geht von den Werten $\frac{\partial f_{lndet}}{\partial \mathbf{L_{i,i}}}$ aus und bestimmt ausgehend von diesen die Einträge von $\frac{\partial f_{lndet}}{\partial \mathbf{C_{k,m}}}$ über eine rückwärtsdifferenzierte Cholesky Zerlegung

Grundsätzlich sollten beide Vorgehensweisen vom numerischen Aufwand gleichwertig

sein. Da allerdings für jeden der Hyperparameter h_l die Matrix $\frac{\partial \mathbf{C_{k,m}}}{\partial h_l}$ bestimmt werden muss und damit wiederum $\frac{\partial \mathbf{L_{i,i}}}{\partial \mathbf{C_{k,m}}}$, muss also die vorwärtsdifferenzierte Cholesky-Zerlegung o-mal aufgerufen werden. In Fall 2 ist dies nicht so, denn die Berechnung ist unabhängig von der Anzahl der Hyperparameter. Aus diesem Grund verspricht der rückwärtsdifferenzierte Fall einen Vorteil bei der Bestimmung der partiellen Ableitungen der Hyperparameter.

Der numerische Aufwand für diesen rückwärtsdifferenzierten Cholesky Algorithmus ist ungefähr doppelt so hoch wie bei einer normalen Cholesky Zerlegung [Smith, 1995]. Das ist immer noch deutlich schneller als eine Vor- und Rückwärtssubstitution. Der in [Smith, 1995] beschriebene Algorithmus ist allerdings nur schwer parallelisierbar und für SIMD Architekturen schlecht geeignet. In der hier entwickelten Matrix-Klasse, wurde für diese Arbeiten eine SIMD-beschleunigte Version des von [Smith, 1995] beschriebenen Algorithmus entwickelt. Dieser wird nicht mehr verwendet und soll daher nicht vorgestellt werden. Stattdessen wird in der Arbeit von [Murray, 2016, Särkkä, 2013] eine moderne Variante präsentiert, welche es ermöglicht Level 2-3 BLAS-Routinen zu verwenden. Diese stellen aktuell die effizientesten Methoden dar und werden in der hier entwickelten Software auch verwendet.

5.5.7 Vergleich zwischen der Invertierung und der Rückwärtsdifferenzierung

In diesem Abschnitt soll ein Vergleich zwischen der Invertierung und Rückwärtsdifferenzierung anhand eines Benchmarks erfolgen. Ein direkter Vergleich der benötigten Operationen innerhalb der Algorithmen ist in diesem Fall so gut wie unmöglich, da stark optimierte Bibliotheken wie die Intel-MKL oder ATLAS-Bibliothek ([Whaley and Dongarra, 1998]) verwendet werden. Diese bedienen sich komplexer Algorithmen wie z.B. den Strassen-Algorithmus, welcher eine Matrixmultiplikation mit einer Komplexität von $\mathcal{O}\left(n^{log_2 7}\right)$ berechnen kann. Häufig werden die Algorithmen je nach Matrix-Größe automatisch umgeschaltet, sodass sich die echte Komplexität kaum noch bestimmen lässt. Die verwendeten Algorithmen werden meist auch nicht dokumentiert. Dennoch lässt sich davon ausgehen, dass diese Bibliotheken das aktuelle Maximum an Geschwindigkeit darstellen, sodass eine gute und auch realistische Vergleichbarkeit gewährleistet ist.

Gleichung 5.47 beschreibt den Teil der Berechnung der Ableitung des Likelihoods in dem sich die beiden Verfahren unterscheiden. Die analytische Ableitung ist wie folgt bestimmt:

$$\frac{\partial \left(ln\left(det\left(\mathbf{Cov}\left(h_l\right)\right)\right)\right)}{\partial h_l} = \text{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right) \tag{5.58}$$

Für beide Verfahren gilt die gleiche Ausgangssituation: Die nach den Hyperparametern abgeleitete Kovarianzmatrix ist bekannt $\frac{\partial \mathbf{Cov}}{\partial h_l}$ sowie die choleskyzerlegte Kovarianzmatrix $\mathbf{L}$. Um eine bessere Vergleichbarkeit zu gewährleisten werden auch die Zwischenschritte der Algorithmen gemessen und verglichen. Daher werden folgend die Abläufe beider Verfahren in 5 Schritte aufgeteilt:

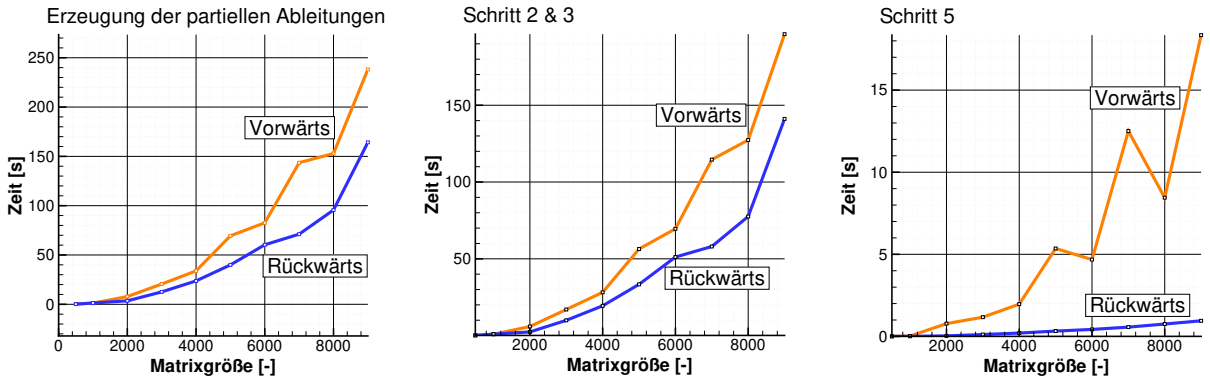


Abbildung 5.13: Vergleich der Zeiten der Berechnung der partiellen Ableitungen des Likelihood-Terms

Ablauf Vorwärtsdifferentiation

1. Bestimmung der differenzierten Kovarianzmatrix $\frac{\partial \mathbf{Cov}}{\partial h_l}$ für jeweils einen Hyperparameter h_l.
2. Bestimmung der Cholesky Zerlegung $\mathbf{L}$.
3. Bestimmung von $\mathbf{Cov}^{-1}$, da die zerlegte Matrix bekannt ist, muss noch eine Vor- und Rückwärtssubstitution mit n^3 Operationen durchgeführt werden.
4. Die Bestimmung der quadratischen Form $\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1} \frac{\partial \mathbf{Cov}}{\partial h_l} \mathbf{Cov}^{-1} \left(\vec{y}_s - \overrightarrow{F}\right)$.
5. Das Matrixprodukt $\mathbf{Cov}^{-1} \frac{\partial \mathbf{Cov}}{\partial h_l}$, wobei hiervon nur die Diagonale berechnet werden muss. Daher liegt diese Berechnung bei ca. n^2 Operationen.

Ablauf Rückwärtsdifferentiation

1. Bestimmung der differenzierten Kovarianzmatrix $\frac{\partial \mathbf{Cov}}{\partial h_l}$ für jeweils einen Hyperparameter h_l.
2. Bestimmung der Cholesky Zerlegung $\mathbf{L}$.
3. Die Bestimmung der rückwärtsdifferenzierten Cholesky Zerlegung $\frac{\partial \mathbf{L_{i,i}}}{\partial \mathbf{C_{k,m}}}$ inklusive Aufstellen der Seed-Matrix $\bar{\mathbf{L}}$ (siehe $\frac{\partial f_{lndet}}{\partial \mathbf{L_{i,i}}}$) mit n Operationen.
4. Die Bestimmung der quadratischen Form $\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1} \frac{\partial \mathbf{Cov}}{\partial h_l} \mathbf{Cov}^{-1} \left(\vec{y}_s - \overrightarrow{F}\right)$.
5. Die Bestimmung des Produkts der aus Punkt 2 berechneten Matrix und den Einträgen der differenzierten $\frac{\partial \mathbf{Cov}}{\partial h_l} \sim \frac{n^2}{2} + \frac{n}{2}$.

Benchmark: Folgend wird ein zeitlicher Vergleich über ein Benchmark angestellt. Die Stützstellen wurden aus der ZDT3 Funktion (siehe Kapitel A.4.1) mit jeweils 8 Parametern zufällig erzeugt. Dazu wurden jeweils 10 Kriging Iterationen durchgeführt und die Zeiten für die Erzeugung der partiellen Ableitungen des Likelihoods gemessen. Berechnet wurde der Testfall auf zwei Xeon E5 2640 v3 mit insgesamt 16 Threads. Für alle Berechnungen wurde die Intel-MKL-Bibliothek verwendet. Abbildung 5.13 (links) zeigt die benötigte Gesamtzeit für die Erzeugung der partiellen Ableitungen. Der Rückwärtsmodus ist in diesem Fall ca. 1.5x schneller als der Vorwärtsmodus. Die folgenden Abbildungen zeigen die Zeiten der verschiedenen Schritte im Vergleich. In Schritt 2-3

und Schritt 5 ist der Rückwärtsmodus schneller. Die Zeiten der Schritte 1 und 4 sind identisch und werden daher nicht gezeigt. Die Ergebnisse können aber im Anhang A.3.20 gefunden werden. Anhand dieses Beispiels kann demonstriert werden, dass der Rückwärtsmodus in allen durchgeführten Testfällen besser abgeschnitten hat.

5.5.8 Ableitung Likelihood: Verzicht auf Inverse - Approximation

Wie bereits in Kapitel 5.5.6 beschrieben, liegt die Schwierigkeit der Bestimmung der partiellen Ableitung des Likelihood-Terms in der Berechnung der Spur. Ein approximatives Verfahren um die Spur einer Matrix R zu schätzen, wird in [Avron and Toledo, 2011, Hutchinson, 1989, Mark Gibbs, 1997] beschrieben. Das Verfahren funktioniert mithilfe eines Zufallszahlen-Vektors $\vec{d}$.

$$Spur\,(R) \approx E\left[\vec{d}^T R \vec{d}\right]$$

$$Spur\,(R) \approx \frac{1}{N} \sum \vec{d}^T R \vec{d}$$

$$\vec{d} = \begin{bmatrix} N\,(0,1) \\ \vdots \\ N\,(0,1) \end{bmatrix}$$

Dabei gilt: Je mehr Zufallszahlen dieser Vektor beinhaltet, desto genauer ist die Schätzung. Die Approximation benötigt daher einen sehr großen Zufallsvektor $\vec{d}$ um eine ausreichende Genauigkeit zu erhalten. In den hier durchgeführten Tests konnte keine Beschleunigung erreicht werden, da bei geringer Dimension von $\vec{d}$ der Verlust an Genauigkeit zu einer erhöhten Trainingsiterationsanzahl führte oder bei höherer Dimension von $\vec{d}$ der Zeitvorteil verloren ging.

5.5.9 Verwendung von GPUs

Die Verwendung von GPUs (Graphical Processing Unit) zur Beschleunigung von intensiven Rechenoperationen findet innerhalb der wissenschaftlichen Gemeinde immer größere Anwendung und Zuspruch (vgl. [Cook, 2012]**)**. Meistens werden solche GPUs als Zusatzkarten für Rechner angeboten und verfügen über einen eigenen schnellen Speicher. Der Unterschied zwischen einer konventionellen CPU und einer herkömmlichen GPU besteht hauptsächlich in der Architektur der Prozessoren. Während CPUs über bis zu ca. 20 (nach heutigem Stand) hochgetaktete Kerne und komplexe (bis zu 3-Schichtige) Cache Strukturen verfügen, so sind GPUs meist mit mehreren tausend niedrig getakteten Kernen und einer sehr breiten Speicheranbindung ausgestattet. Weiterhin ist die direkte Anbindung des Speichers auf der Platine der GPU selbst ein Vorteil. Durch die immer größer werdende SIMD-Einheiten (Single Instruction Multiple Data) in modernen CPUs (bspw. AVX512 mit 512 breiten Registern) verschwindet der Unterschied zwischen den Architekturen allerdings zusehends. Tabelle 5.7 zeigt die Anzahl der Recheneinheiten

	Jahr	SIMD Einheiten	Cores	FMA Einheiten	Gesamt
Intel E7-8870	2011	8	10	1	80
Intel Plat. 8180	2017	16	28	2	896
Nvidia P100	2017	-	-	-	3584

Tabelle 5.7: Anzahl der Rechneneinheiten für verschiedene CPUs/GPUs

heiten von zwei verschiedenen Intel CPUs und einer aktuellen GPU von Nvidia. Auch auf programmiertechnischer Seite verschwindet der Unterschied zwischen CPU und GPU. So ist der Aufwand um eine einfache Matrixoperation (wie z.B. eine Matrixmultiplikation) zu programmieren, (welche die entsprechende Hardware vollständig ausnutzt) zu einer Expertenaufgabe geworden. Aus diesem Grund bieten sowohl Intel als auch NVidia zahlreiche hardwareoptimierte Bibliotheken für alle Arten von Rechenoperationen an. Größere Unterschiede zwischen GPUs und CPUs zeigen sich allerdings bei dem Transfer von Daten. Abbildung 5.14 zeigt den vereinfachten Aufbau eines Bus-Systems wie es auf aktuellen Mainboards zu finden ist.

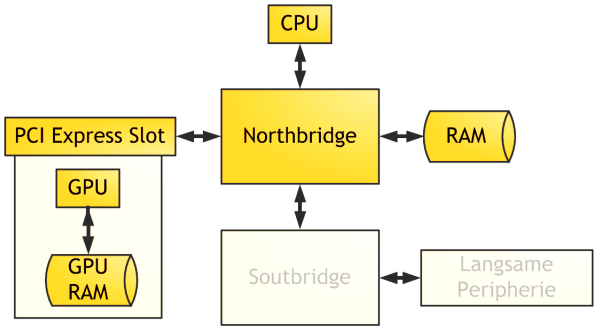


Abbildung 5.14: Typischer Aufbau eines Bus-Systems

Folgend wird der Ablauf einer Matrix-Zerlegung auf einer GPU und auf einer CPU beschrieben, wobei der Fokus auf dem Datentransfer liegt:

CPU:

1. CPU erzeugt und belegt Matrix A im RAM (typische Transferraten von 100GB/-sec)
2. CPU zerlegt Matrix A im RAM

GPU:

1. CPU erzeugt und belegt Matrix A im RAM (typische Transferraten von 100GB/-sec)
2. GPU kopiert Matrix A vom RAM in den GPU-RAM über den PCIe Bus (typische Transferraten von 15GB/sec)
3. GPU zerlegt Matrix A im GPU-RAM (typische Transferraten von 320GB/sec)
4. GPU kopiert Matrix A zurück in den RAM (typische Transferraten von 15GB/sec)

Dabei kann der PCIe-Bus mit 15GB/sec (Stand 2017) zum Flaschenhals der Operation werden. Die Beschleunigung der Rechnung auf der Karte selbst muss den Overhead des Transfers überwiegen. Erst dann lohnt sich der Einsatz einer solchen GPU.

Ob der praktische Einsatz für das hier entwickelte Kriging-Verfahren sinnvoll ist, soll innerhalb dieses Abschnittes betrachtet werden. In [Toal, 2016] wird die Verwendung von GPUs innerhalb eines Kriging-Verfahrens bereits als gewinnbringend beschrieben,

es werden allerdings nur kleine Matrizen betrachtet (n<1000) und die verwendete Bibliothek nicht beschrieben. Weiterhin werden die Overhead-Anteile und auch die Fließkommagenauigkeit nicht angegeben. Dies kann von Bedeutung sein, da eine höhere Genauigkeit (von 64Bit) bei einigen GPUs zu massiven Leistungseinbrüchen führt. Zudem stellt sich die Frage, ob es während einer Optimierung möglich und sinnvoll ist, mehrere Trainings auf einer GPU durchzuführen. Folgend werden diese Fragestellungen betrachtet.

Umgesetzte Algorithmen Es erscheint nicht sinnvoll, alle Algorithmen als GPU-Version umzusetzen, da diese durch den Transfer letztlich langsamer wären. Aus diesem Grund sind in der „*CudaMatrix*“-Klasse (siehe Kapitel 5.5.1), nur die zeitkritischen Algorithmen (Cholesky Zerlegung, Rückwärtsdifferenzierte Cholesky-Zerlegung, Matrix-Multiplikation, Lösen eines Gleichungssystems) implementiert. Für alle anderen wird die „*OpenMPMatrix*“-Klasse verwendet.

Die genaue Umsetzung wird in [Küppers, 2016b, Küppers and Schmitz, 2016, Küppers, 2016a] beschrieben und soll hier nicht weiter thematisiert werden.

Fließkommagenauigkeit Beim Kriging-Verfahren ist die numerische Berechnung mit doppelter Fließkommagenauigkeit (64Bit) aus Stabilitätsgründen unerlässlich. Aus diesem Grund sollte die Leistung der einzelnen GPU-Architekturen berücksichtigt werden. In Tabelle 5.8 wird die relative Rechenleistung bei 64Bit bezogen auf einfache Genauigkeit von verschiedenen Nvidia Architekturen der letzten Jahre verglichen.

Auffällig ist hier die Maxwell-Architektur, die eine deutlich niedrigere Leistung bei doppelter Genauigkeit bringt. Die Unterschiede liegen in der Prozessorarchitektur begründet, so haben die Kepler-GPUs der Tesla-Reihe alle pro Single-Precision-Core einen zusätzlichen Double-Precision-Kern. Daher auch das Verhältnis von 1/3. Die Maxwell Architektur verzichtet hingegen auf jegliche Double-Precision-Einheiten und kann diese nur emulieren.

Architektur	Jahr	DP/SP
Fermi	2010	1/2x
Kepler	2012	1/3x
Maxwell	2014	1/32x
Pascal	2016	1/2x
Volta	2017	1/2x

Tabelle 5.8: Vergleich der relativen Rechenleistung verschiedener NVidia Architekturen bei Double Precision (64Bit) bezogen auf Single Precision (32Bit)

Wirtschaftliche Betrachtung Neben der reinen Rechenleistung müssen auch wirtschaftliche Aspekte berücksichtigt werden. Insbesondere spielen der Anschaffungspreis und auch der Stromverbrauch eine sehr wichtige Rolle. In Anhang A.3.18 ist ein tabellarischer Vergleich zwischen verschiedenen GPUs und CPUs aufgestellt. Besonders wichtig sind dabei die Werte Watt/GFlop und GFlop/Euro. Hier ist schnell ersichtlich, dass in beiden Punkten der Einsatz von GPUs rentabel sein kann. Die oftmals vertretene Meinung, dass viele Algorithmen aufgrund der schlechten Parallelisierbarkeit besser auf CPUs ausführbar sind, stimmt nur noch bedingt bis garnicht, da moderne CPUs auch auf SIMD-Befehle mit vielen Recheneinheiten zurückgreifen (siehe

	Energiebedarf	CPUs/GPUs	Rechenleistung	Standort
Summit	15000kW	IBM Power9+Nvidia V100	200000 TFlop	USA
TaihuLight	15379kW	Sunway SW26010	93014 TFlop	China
Tianhe-2	17808 kW	Intel E5-2692+Phi 31S1P	33862 TFlop	China
Piz Daint	2272kW	Intel E5-2690+Nvidia P100	19590 TFlop	Schweiz
...	...	...	...	...
Hazel Hen	3200kW	Intel E5-2680	7420 TFlop	Deutschland

Tabelle 5.9: Architektur und Energiebedarf der Supercomputer mit der höchsten theoretischen Rechenleistung (Stand Juni 2018)

Tabelle 5.7).

Der Energieverbrauch pro Rechenleistung ist neben der reinen Rechenleistung ein wichtiges Kriterium für die Auswahl einer geeigneten CPU/GPU-Kombination für einen Supercomputer. Der Energiebedarf (siehe Tabelle 5.9) der aktuell leistungsstärksten Supercomputer (Stand Juni 2018) überschreitet bereits mehrere Megawatt und verdeutlicht die Wichtigkeit dieses Kriteriums.

Aus Abbildung 5.15 wird ersichtlich, dass der Anteil an GPUs stark zugenommen hat. Dies kann auf das Verhältnis von benötigter Energieleistung zu Rechenleistung zurückgeführt werden. Vergleicht man die Spitzenmodelle von Nvidia (V100) mit dem Platinum 8180 von Intel, so liegt der Energiebedarf bei 0.04Watt/GFlop (V100) im Vergleich zu 0.134 Watt/GFlop (Platinum 8180). Der TaihuLight-Supercomputer stellt hier eine Ausnahme dar und verwendet die in China entwickelten SunWay-Prozessoren. Die Entscheidung auf SunWay-CPUs zurückzugreifen wurde auch aus politischen Gründen getroffen, weil seitens der US-Regierung die Auslieferung von Intel CPUs behindert wurde (siehe [Government, 2009]).

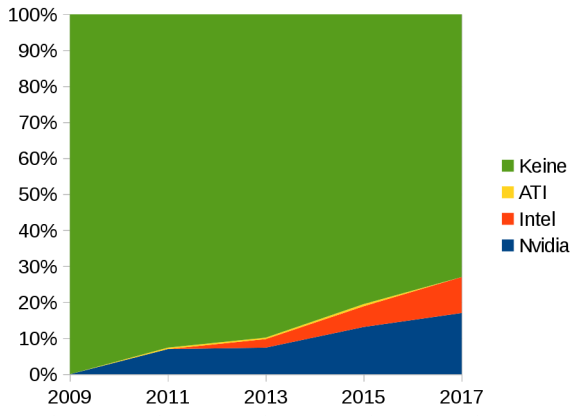


Abbildung 5.15: Entwicklung der Beschleuniger-Anteile an Supercomputern (Quelle:[top, 2018])

Im Hinblick auf Leistung und Verbrauch sind die Supercomputer in Deutschland im globalen Vergleich als eher ineffizient anzusiedeln (siehe Tabelle 5.9). Besonders im Hinblick auf kommende Anforderungen im Bereich von Deep-Learning Verfahren sind die Rechner nicht ausreichend ausgestattet. Dies ist unter anderem auch durch die ablehnende Haltung gegenüber Beschleunigerkarten begründet wie sie in anderen Ländern nicht anzutreffen ist.

Benchmarks Innerhalb dieses Abschnittes werden einige Benchmark-Ergebnisse der entwickelten Kriging-Software gezeigt. Im ersten Teil geht es um die Ausnutzung der theoretischen Rechenleistung und dabei soll die Frage geklärt werden, ob es möglich ist, die volle Rechenkapazität einer GPU für das hier entwickelte Kriging-Verfahren

Name	Theoret. Beschleunigung	Erreichte Beschleunigung
2xIntel E5-2695v2 (Ref.)	1	1
2xIntel E5-2698v3	2.55	2.58
Nvidia K40	3.11	3.1
2xNvidia K40	6.22	6.21
Nvidia K80	6.3	5.32

Tabelle 5.10: Theoretische und erreichte Beschleunigung des Benchmarks

zu nutzen. Anschließend werden die Overhead-Anteile in Abhängigkeit der Matrixgröße für ein Training dargestellt. Weiterhin wird die Machbarkeit und Effizienz der gleichzeitigen Ausführung mehrerer Trainings auf einer GPU getestet.

Vergleich der theoretischen und tatsächliche erreichten Rechenleistung für ein Training Folgende Tabelle zeigt die theoretische und die real gemessene Rechenleistung für ein Training mit 10000-15000 Stützstellen und 20 Parametern für verschiedene Hardware-Konfigurationen. Die Anzahl der Trainingsschritte sowie die Ergebnisse sind jeweils identisch.

Als Referenz wurde ein Rechner mit 2xIntel Xeon E5-2695v2 verwendet. Die GPU-Benchmarks wurden auf dem NVidia-PSG-Cluster ausgeführt[1].

Die Ergebnisse zeigen, dass bei nahezu allen Benchmarks die theoretische Beschleunigung erreicht wird. Die K80 stellt hier einen Sonderfall dar, da sie über zwei Prozessoren auf einer Platine verfügt. Die theoretischen Werte liegen bei dieser Karte erwartungsgemäß niedriger. Grundsätzlich lässt sich aber sagen, dass die Rechenkapazität der GPUs für ein Kriging-Verfahren sehr gut nutzbar ist.

Overhead-Anteile bei der Berechnung Im Vergleich zu einer CPU muss ein zusätzlicher Transfer durchgeführt und eine Initialisierung durchlaufen werden. Beides schmälert die Rechenleistung der GPU. Daher wird folgend ein Benchmark gezeigt, das die Anteile der Initialisierung und die Anteile des Transfers im Vergleich zu der Rechenzeit ins Verhältnis setzt. Es handelt sich hierbei um denselben Testfall wie in Kapitel 5.5.9, wobei der hier gezeigte auf einer Nvidia-Quadro-K6000 durchgeführt wurde.

In Abbildung 5.16 sind die Ergebnisse dargestellt. Bis zu einer Matrixgröße von 2000x2000 steigt der prozentuale Anteil des Transfers auf ca. 15% der gesamten

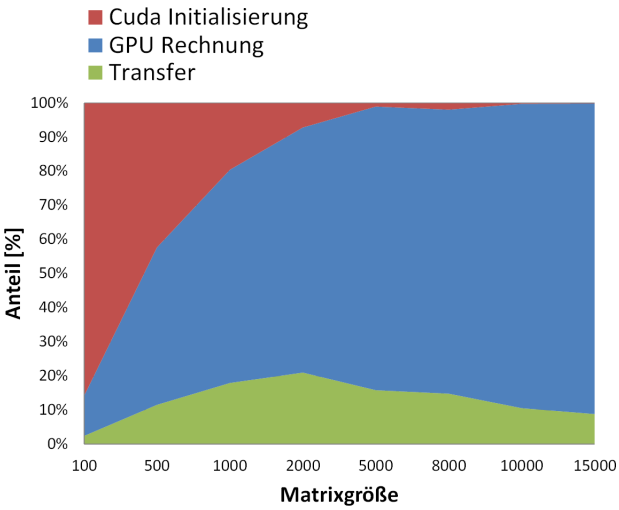


Abbildung 5.16: Overhead-Anteile bei einem Kriging-Training mit einer GPU (Nvidia K6000)

[1] An dieser Stelle möchte ich mich persönlich bei NVidia für die Möglichkeit bedanken, die Benchmarks auf dem PSG-Cluster ausführen zu dürfen.

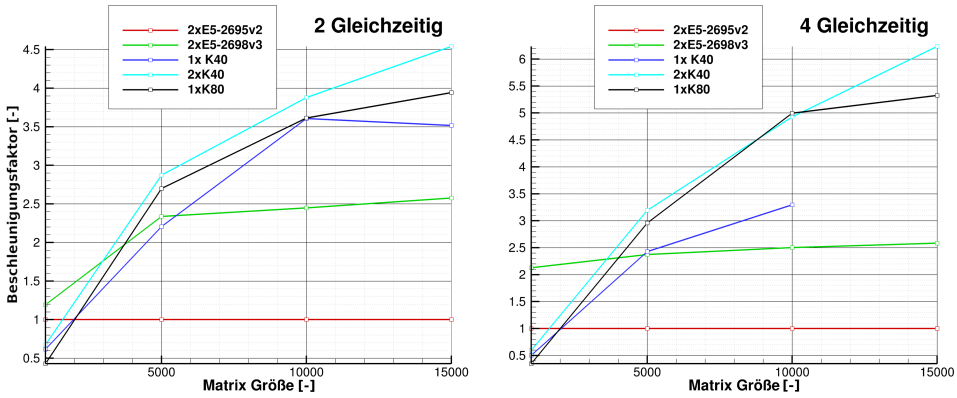


Abbildung 5.17: Vergleich von mehreren Trainings gleichzeitig auf verschiedenen CPUs und GPUs

Zeit, wobei die Initialisierung immer eine
konstante Zeit benötigt und daher der Anteil mit steigender Größe kontinuierlich abnimmt. Erst ab einer Größe von 2000x2000 kann die GPU das volle Potential nutzen. Weitere Entwicklungen der Hardware (PCIe 4.0 und NVLink) werden diesen Wert mit der Zeit allerdings verkleinern, da die Transferzeiten damit gesenkt werden können.

GPU Ressourcenverteilung bei parallelen Trainings Aufgrund der besonderen Beschaffenheit von GPUs stellt sich die Frage, inwiefern GPUs Multiprozessfähig sind. Von besonderem Interesse ist hierbei das Verhalten der Karten, wenn sich mehrere Prozesse die Ressource teilen. Die Schwierigkeit liegt dabei in der Übertragung auf die GPU und in der Ausführung der unterschiedlichen Prozesse auf einer Karte. GPUs sind sehr stark auf einen Prozess ausgelegt, welcher den GPU-Speicher linear durchläuft und so die volle Bandbreite des Speichers nutzen kann.

Greifen mehrere Prozesse auf unterschiedliche Bereiche des GPU-Speichers zu, so kann dies die Performance der Karte schwächen. Das hängt aber von der Lastaufteilung innerhalb der Karte ab, welche von „außen“ schwer beeinflussbar ist. Aus diesem Grund wurde ein Benchmark ausgeführt (siehe Abbildung 5.17), bei dem jeweils 2 und 4 gleichzeitige Trainings bei unterschiedlicher Matrixgröße getestet wurden.

Das Ergebnis zeigt, dass die Geschwindigkeit nahezu linear mit Anzahl der Trainings skaliert und sich damit die Rechenkapazität der Karten noch deutlich besser nutzen lassen. Jedoch kann der GPU-RAM eine Beschränkung darstellen, so wie es bei 4 gleichzeitigen Trainings auf einer K40 bei 15000 Stützstellen zu sehen ist. Die Karten sind somit für diesen Fall sehr gut geeignet, sofern der Speicher ausreichend groß ist.

Weiterhin wurde getestet, wie gut sich zwei unterschiedlich starke GPUs mit mehreren Trainings bedienen lassen, die Ergebnisse sind in Anhang A.3.19 zu finden.

Nvidia K40 / K6000 Bios Nvidia stellte für die Ausführung der Benchmarks das PSG-Cluster mit K40 und K80 GPUs zur Verfügung. Nachdem die Tests abgeschlossen waren, konnten nur noch die institutseigene Nvidia Quadro K6000 verwendet werden. Damit war eine Vergleichbarkeit der weiteren Tests nur schwer möglich. Jedoch besitzen die K40 und die K6000 dieselben Prozessoren (GK110) und auch denselben RAM (GDDR5 384Bit ECC). Dennoch wurde zwischen den GPUs ein Leistungsunterschied von ca. 30% festgestellt. Der Unterschied in den Karten wird durch das GPU-Bios im

sogenannten „PowerTable“ bestimmt. In dieser Tabelle sind die elektrischen Leistungen für die verschiedenen Taktfrequenzbereiche der GPU festgelegt. Die Werte der maximalen Leistung der K6000 liegen deutlich unter der K40, daher ist es der Karte nicht gestattet, die volle Taktfrequenz zu erreichen. Sofern das Mainboard die Leistung zulässt, können die Leistungswerten in dieser Tabelle erhöht werden. Hierfür kann das NVFlash-Tool verwendet werden, welches die Manipulation des GPU-Bios ermöglicht. Dafür muss zuerst ein Download des Bios durchgeführt werden, dann die Binärdatei mit einem Hexeditor editiert und das geänderte Bios wieder hochgeladen werden. Für die K6000 wurde diese Prozedur durchgeführt und die Leistungswerte entsprechend der K40 angepasst. So war es möglich weitere Benchmarks mit der K6000 auf K40-Niveau durchzuführen ohne eine Neuanschaffung zu tätigen.

5.6 Laufzeit-Analysesoftware für Krigingmodelle

Während einer Optimierung werden mehrere Ersatzmodelle trainiert, wobei das Training sehr häufig durchgeführt wird. Aus diesem Grund ist es von großer Bedeutung, die Ersatzmodelle stetig zu überwachen. Besonders wichtig sind hierbei folgende Punkte:

- Die Güte der einzelnen Ersatzmodelle
- Die benötigte Trainingszeit
- Evtl. Fehler in den Daten oder in der Prozesskette

Um eine solche Überwachung übersichtlich zu gestalten, wurde im Rahmen dieser Arbeit eine Software entwickelt, welche den aktuellen Stand der Ersatzmodelle und die Entwicklung grafisch darstellt. Das Programm wurde in der Programmiersprache Python geschrieben und erzeugt nach Ausführung eine PDF-Datei. Folgend sollen einige der wichtigsten grafischen Ausgaben gezeigt und deren Einsatz erläutert werden. Als Beispiel wird dafür das Mukoko-Projekt verwendet, welches in Kapitel 6.3.2 noch ausführlicher beschrieben wird.

Vorhersagefehler

Eine einfache Möglichkeit die Qualität der Ersatzmodelle zu bestimmen, ist die Überprüfung der Differenz zwischen Vorhersage und realen Werten. Da der Optimierungsalgorithmus ein exploratives Vorgehen (siehe Kapitel 2.3.2) wählen kann, ist es möglich, dass dieser bewusst Vorhersagen mit hoher vorhergesagter Standardabweichung auswählt. Entsprechend sollte auch der Vorhersagefehler an solchen Stellen größer sein. Daher ist diese Art der Überprüfung recht ungenau und gibt erst über einen längeren Zeitraum belastbare Ergebnisse. Abbildung 5.18 zeigt den typischen Verlauf eines solchen Vorhersagefehlers. In blau sind die Vorhersagen der Stützstellen dargestellt und in rot die nachgerechneten Werte. Die angezeigten Fehlerbalken entsprechen der vorhergesagten Standardabweichung. Die X-Achse stellt die jeweilige Optimierungsiteration dar. Die grüne Kurve beschreibt die absolute Differenz zwischen Vorhersage und realem Wert.

Zeitlicher Verlauf des Likelihood-Werts

Über den Verlauf des Likelihood-Werts können das Training der Ersatzmodelle überwacht und auch um fehlerhafte Daten schnell detektiert werden. Um dies zu verdeutlichen, werden in Abbildung 5.19 zwei verschiedene Plots unterschiedlicher Ersatzmodelle derselben Optimierung gezeigt. Auf der Ordinate ist der Likelihood-Wert aufgetragen, wobei ein möglichst kleiner Wert zu bevorzugen ist. Auf der Abszisse ist der Iterationsschritt der Optimierung dargestellt. Die roten Punkte markieren die Optimierungsschritte, in denen das Training noch nicht den „*Restart-Modus*“ (siehe Kapitel 5.3.3) verwendet hat. Solange dieser Plot noch rote Punkte aufweist, kann davon ausgegangen werden, dass die Hyperparameter der Ersatzmodelle noch nicht optimal eingestellt sind.

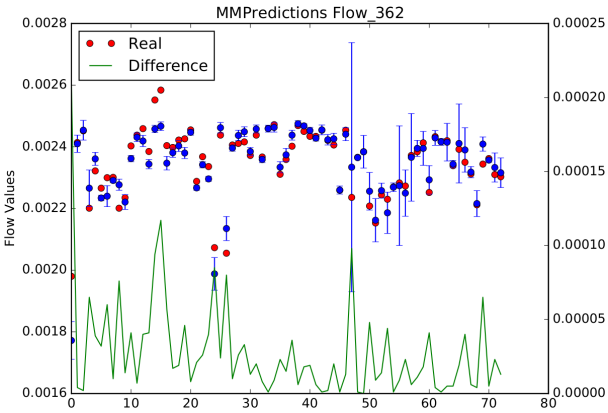


Abbildung 5.18: Exemplarische Ausgabe eines vorhergesagten Fehlers der maximalen mechanischen Dehnung eines Rotors.

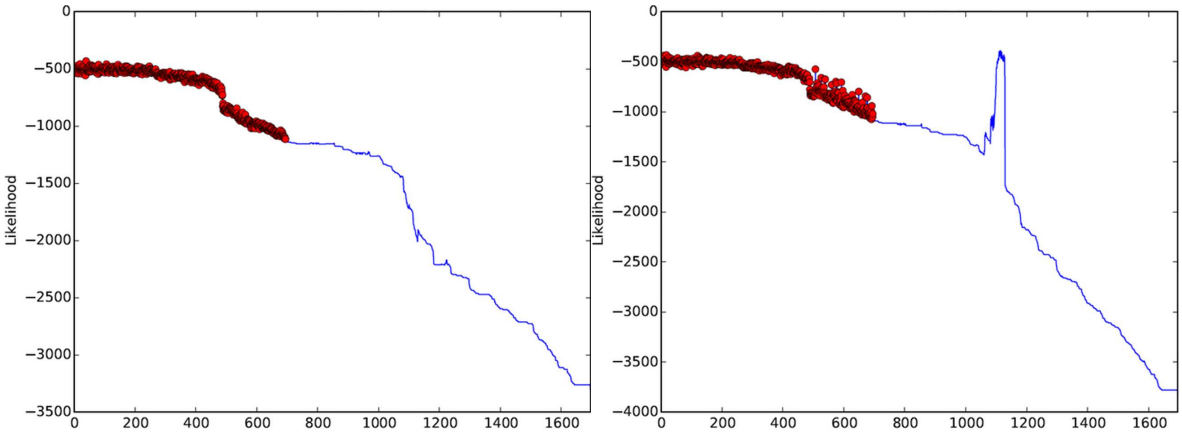


Abbildung 5.19: Verlauf von zwei Ersatzmodellen der Mukoko Optimierung, wobei das rechte Ersatzmodell fehlerhafte Daten bekam

Weiterhin lassen sich auch Aussagen über die Güte der Daten anhand des Kurvenverlaufs treffen. In Abbildung 5.19 (rechts), ist bei Iteration 1000 ein Ausreißer im Verlauf des Likelihoods zu sehen. Dieser Ausreißer wurde während der laufenden Optimierung bemerkt und konnte auf einen systematischen Fehler in der Prozesskette zurückgeführt werden, welcher aus einer Prozesskettenänderung während der Laufzeit resultierte. Der Fehler konnte während der Laufzeit behoben und die Optimierung mit wenig Zeitverlust fortgesetzt werden.

Hyperparameter auf Plausibilität überprüfen

Die Überwachung der Hyperparameter während einer laufenden Optimierung ist aufgrund der hohen Anzahl schwierig. Betrachtet man bspw. die Mukoko Optimierung, so wurden 38 Ersatzmodelle und 158 Parameter vewendet, was zu über 6000 Hyperparametern führt. Dennoch ist es zumindest möglich, eine grafische Plausibilitäts-Überprüfung durchzuführen. Hierfür müssen die Optimierungsparameter und auch die Ersatzmodelle sinnvoll nummeriert werden. Beispielsweise sollten alle Parameter eines Rotors durchlaufend nummeriert werden. Wird dies gemacht und die Hyperparameter nach Größe eingefärbt, so lässt sich eine grafische Matrix erzeugen, wie sie exemplarisch in Abbildung 5.20 gezeigt wird. In dem gewählten Plot stellt grün einen hohen Wert dar und rot einen sehr niedrigen, wobei ein sehr niedriger Wert darauf schließen lässt, dass das Funktional von dem zugehörigen Parameter kaum beeinflusst wird. Innerhalb dieses Plots sind grüne und rote/gelbe Blöcke zu erkennen. Diese zeigen für diesen Fall, dass die Statoren 1 und 2 für die Strukturmechanik keine Rolle spielen, was auch plausibel ist, da keine strukturmechanische Bewertung der Statoren durchgeführt wurde. Weiterhin kann man zwischen Rotor 1 und Rotor 2 alternierende grüne/rote Blöcke im Bereich der strukturmechnischen Ersatzmodelle erkennen. Auch diese Information ist plausibel, da die freien Variablen für Rotor 1 nur einen geringen Einfluss auf die strukturmechanischen Ergebnisse von Rotor 2 haben sollten. Zusammenfassend kann man sagen, dass es mithilfe dieser Informationen möglich ist, die Plausibilität der Ersatzmodelle auf einfache Weise zu überprüfen.

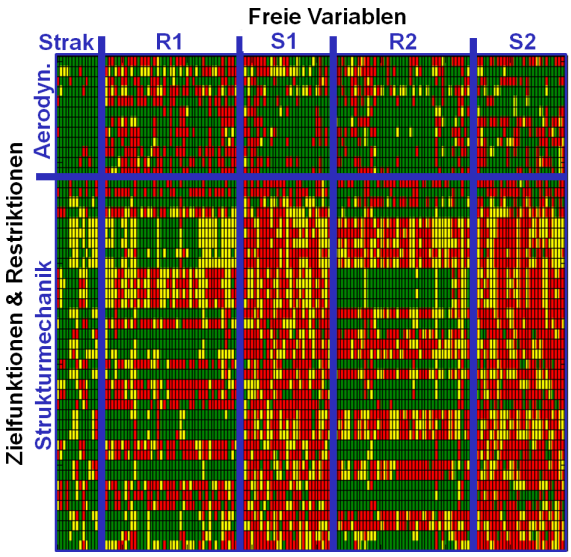


Abbildung 5.20: Hyperparameter-Matrix aus einer laufenden Optimierung

6 Anwendungen

Innerhalb dieses Kapitels werden Anwendungen des hier entwickelten Verfahrens gezeigt. Im ersten Abschnitt werden die Ergebnisse analytischer Tests des Multifidelity-Optimierungsverfahrens mittels Co-Kriging-Ersatzmodellen beschrieben. Darauf folgend wird ein Vergleich zwischen dem Multifidelity- und dem Single-Fidelity-Verfahren an einer aeromechanischen Fanstufen-Optimierung gezeigt. Als letztes werden Anwendungen in Forschung und Industrie beschrieben, in denen das hier entwickelte Verfahren bereits verwendet wurde.

6.1 Analytische Testfälle

In diesem Abschnitt wird ein einfacher Testfall vorgestellt, welcher die grundlegende Funktion des hier entwickelten Co-Kriging-Ersatzmodells verifiziert. Im zweiten Teil werden die Ergebnisse einer Multifidelity-Optimierung basierend auf einer analytischen Testfunktion gezeigt. Dabei werden verschiedene Tests durchgeführt, um die Eigenschaften des neuen Verfahrens zu ermitteln und zu demonstrieren.

6.1.1 Co-Kriging Testfall

Der erste Testfall basiert auf einer aus der Literatur bekannten Testfunktion (siehe [Forrester et al., 2007, Le Gratiet, 2013]) und soll die grundlegende Funktion des hier entwickelten Co-Kriging-Verfahrens zeigen. Wie bereits in Kapitel 4.4 beschrieben, ist es mit dem hier verwendeten Trainingsverfahren möglich, die Stützstellen verschiedener Gütestufen frei zu verteilen. Aus diesem Grund wurden für dieses Beispiel bewusst die Stützstellen niedriger Güte an den Stellen hoher Güte weggelassen und die Abstände der Punkte niedriger Güte variiert (vgl. [Forrester et al., 2007, Le Gratiet, 2013]). Die Testfunktion ist durch die Gleichungen 6.1 für die Funktion hoher Güte $f_{hf}(x)$ und niedriger Güte $f_{lf}(x)$ definiert. Die Funktion niedriger Güte stellt dabei eine verzerrte Form der Funktion hoher Güte dar.

$$\begin{aligned} f_{hf}(x) &= (6x-2)^2 \sin(12x-4), x \in [0,1] \\ f_{lf}(x) &= A f_{hf}(x) + 10(x-0.5) + 5 \end{aligned} \tag{6.1}$$

Die grundlegende Annahme (siehe [Forrester et al., 2007, Le Gratiet, 2013]) des Co-Kriging-Verfahrens (siehe Gleichung 4.25) ist, dass die Daten hoher Güte aus

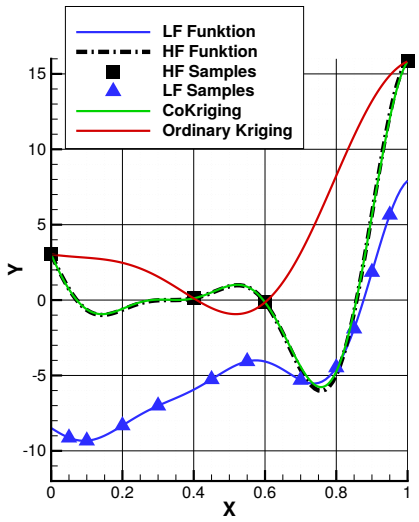


Abbildung 6.1: Eindimensionales analytisches Beispiel mit zwei Gütestufen.

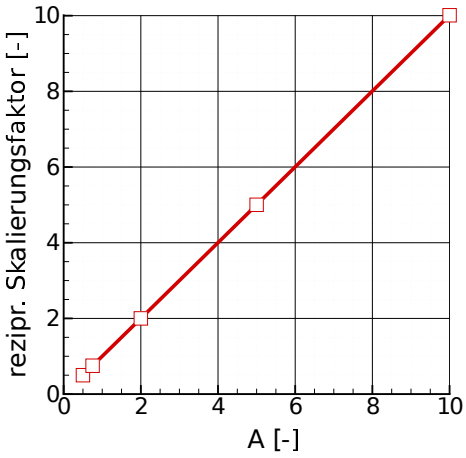


Abbildung 6.2: Vorhergesagter und realer Skalierungsfaktor im Vergleich

den Daten niedriger Güte entspringen. Die Eignung dieses analytischen Testfalls ist nicht optimal, weil dieser genau umgekehrt definiert ist. Ein Vorteil liegt laut [Forrester et al., 2007] aber darin, dass der Skalierungsfaktor A dem reziproken Skalierungsfaktor a des trainierten Modells entsprechen sollte (siehe Gleichung 4.25) und das Ergebnis des Trainings somit direkt verifiziert werden kann.

Abbildung 6.1 zeigt zum einen die Funktionen sowie die gewählten Stützstellen und zum anderen die Erwartungswert-Vorhersagen des Ordinary- und des Co-Kriging-Verfahrens, wobei dem Ordinary-Kriging nur die Stützstellen hoher Güte zur Verfügung standen. Das Co-Kriging kann die Funktion nahezu fehlerfrei darstellen, was auf eine optimale Einstellung der Hyperparameter schließen lässt.

In Abbildung 6.2 wird der mit dem Training bestimmte reziproke Skalierungsfaktors $\frac{1}{a}$ mit den verschiedenen Skalierungsfaktoren A verglichen. Der Einfachheit halber werden direkt die Reziprokwerte dargestellt, da diese mit dem eingestellten Skalierungsfaktor A übereinstimmen sollten. Wie man erkennen kann, stimmen die trainierten Ergebnisse alle sehr gut mit den eingestellten Faktoren der Funktion überein, der Fehler liegt im Bereich des numerischen Rauschens.

Zusammenfassend kann man festhalten, dass die Funktionalität und auch die Leistungsfähigkeit dieses Verfahrens mit diesem Testfall gut demonstriert wurde.

6.1.2 Analytische Optimierungsbeispiele

Innerhalb dieses Abschnitts sollen mehrere Multifidelity-Optimierungstestfälle vorgestellt und deren Ergebnisse diskutiert werden. Hierfür wurden zwei Arten von Testfällen entwickelt:

1. Testfälle, welche das Zusammenspiel von Co-Kriging und Entscheidungsfunktion während einer Optimierung zeigen.
2. Testfälle, welche an die Grenzen des entwickelten Verfahrens gehen und das hier

Bezeichnung	$f_{hf}(\vec{x})$	$f_{lf}(\vec{x})$
Referenz	$f_{ZDT}(\vec{x})$	-
1a	$f_{ZDT}(\vec{x})$	$f_{ZDT}(\vec{x}) + U(0,0)$
1b	$f_{ZDT}(\vec{x})$	$f_{ZDT}(\vec{x}) + U(0,0.1)$
1c	$f_{ZDT}(\vec{x})$	$f_{ZDT}(\vec{x}) + U(0,1)$
2a-g	$0.1 f_{ZDT}(\vec{x}) + 0.5x_0 - 10x_1 + 10$	$f_{ZDT}(\vec{x})$
3	$f_{ZDT}(\vec{x})$	0

Tabelle 6.1: Auflistung der Optimierungstestfälle

entwickelte Multifidelity-Verfahren in besonderer Weise beanspruchen.

In Tabelle 6.1 werden die Testfälle aufgelistet, wobei Testfall 1(a-c) und 2(a-g) zu der ersten Kategorie gehören und Testfall 3 zu der zweiten. Für jeden Testfall ist eine Funktion hoher Güte $f_{hf}(\vec{x})$, $\vec{x} \in \mathbb{R}^{\mathrm{k}}$ und eine Funktion niedriger Güte $f_{lf}(\vec{x})$ mit einem zeitlichen Faktor von $\frac{t_{hf}}{t_{lf}} \approx 70$ definiert. Die Anzahl der freien Variablen wurde auf $k = 20$ festgelegt. Eine einzelne freie Variable x_r wird über den Index $r \in \{0, ..., k-1\}$ beschrieben. Die verwendete analytische Funktion $f_{ZDT}(\vec{x})$ ist die ZDT3-Funktion aus Anhang A.4.1 - Gleichung A.12, auf deren Basis die Funktionen der unterschiedlichen Gütestufen gebildet werden. $U(a,b)$ beschreibt einen Zufallszahlengenerator, welcher eine beliebige Zahl aus einer Gleichverteilung $\Omega[a,b]$ zieht.

Die Aufgabe ist die Minimierung der Funktion hoher Güte $\min_{\vec{x} \in \mathbb{R}^{\mathrm{k}}} f_{hf}(\vec{x})$. Für alle Testfälle wird das entwickelte Multi-Fidelity-Verfahren mit der in Kapitel 3.2.2.2 beschriebenen Entscheidungsfunktion verwendet und zusätzlich eine Referenzoptimierung durchgeführt. Dabei handelt es sich um eine „Single-Fidelity“-Optimierung, welche das Ordinary-Kriging-Verfahren verwendet.

Testfall 1 - Entscheidungsfunktion bei unterschiedlichem Informationsgehalt der niedrigen Güte: Mit diesem Testfall wird das Verhalten der Entscheidungsfunktion bei einer verrauschten Funktion niedriger Güte $f_{lf}(\vec{x})$ bestimmt. Die Funktion hoher Güte ist in diesem Fall die Funktion $f_{ZDT}(\vec{x})$. Die Funktion niedriger Güte ist zusätzlich mit einem gleichverteilten Rauschen belegt $f_{ZDT}(\vec{x}) + U(a,b)$. Man beachte, dass die Funktion durch das Rauschen im Mittel zu höheren Werten verschoben wird, bei z.B. $f_{ZDT}(\vec{x}) + U(0,1)$ würde eine mittlere Verschiebung von $\frac{b-a}{2} = 0.5$ erwartet. Das Konvergenzkriterium wurde bei einem Zielfunktionswert von $f_{hf}(\vec{x}) < 0.228$ festgelegt.

Um einen Eindruck des Rauschens zu bekommen, ist in Abbildung 6.3 die $f_{ZDT}(\vec{x})$ Funktion mit einem Rauschterm $U(a = 0, b = 1)$ dargestellt. Die Anzahl der freien Variablen ist für diese Darstellung auf $k = 2$ reduziert.

Ein stärkeres Rauschen in der Funktion niedriger Güte drückt sich im Co-Kriging durch einen höheren Rauschterm λ_2 (siehe Kapitel 5.1.2.1) aus. Der Erwartungswert niedriger Güte wird approximiert und damit geglättet. Das Kovarianzmodell hoher Güte $cov_{1,1}(\vec{x}_1, \vec{x}_2)$ sollte davon unberührt bleiben. Der Informationsgehalt der Daten niedriger Güte sinkt also mit steigendem Rauschen, weshalb die Entscheidungsfunktion den Optimierungsverlauf bei einem größeren Rauschen schneller in Richtung hoher Güte lenken sollte.

Weiterhin wurde für die Entscheidungsfunktion die Trainingszeit und die Zeit für die Optimierung auf dem Ersatzmodell auf 0 gesetzt, da diese aufgrund der extrem niedrigen Prozesskettenzeiten überwiegen würde. Wenn die Trainingszeit deutlich höher als die Prozesskettenzeit selbst ist, verschiebt sich die Entscheidung immer mehr in Richtung hoher Güte (siehe Gleichung 3.8). Dieses Verhalten ist in einer realen Anwendung erwünscht, bei diesem analytischen Testfall allerdings nicht.

In Tabelle 6.2 ist die Anzahl der benötigten Individuen für die Referenz sowie die Varianten 1a, 1b und 1c angegeben. Der Index H bzw. L steht für hohe und niedrige Güte. i_{bestH}, i_{bestL} sind die benötigte Anzahl an Individuen nach der Initialisierungsphase bis zur Konvergenz. Die letzte Spalte gibt den Beschleunigungsfaktor t_{fact} an, der in Kapitel 3.2.3 definiert wurde.

Alle Optimierungen starten mit denselben 60 Initialisierungsindividuen hoher Güte, wobei die Multifidelity-Optimierungen weitere 200 Individuen niedriger Güte zur Verfügung haben. Alle Parametersätze der initialen Individuen wurden zufällig erzeugt.

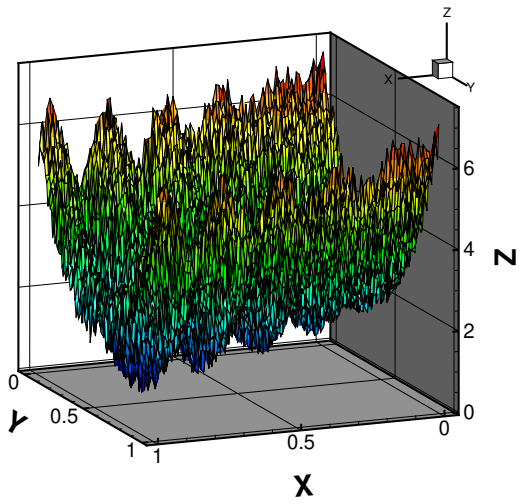


Abbildung 6.3: Exemplarische Funktion niedriger Güte bei $a = 0, b = 1$ für zwei freie Variablen. Die in der Optimierung verwendete Funktion hat allerdings 20 freie Variablen.

Vergleicht man die Beschleunigungsfaktoren t_{fact} miteinander, so hat die Optimierung 1a die stärkste Beschleunigung mit einem Faktor von 2.2 erreicht. Weiterhin ist zu sehen, dass die Beschleunigung mit steigender Breite b abnimmt. Dies entspricht auch dem erwarteten Ergebnis. Betrachtet man nach der Initialisierungsphase die Anzahl der Individuen hoher und niedriger Güte i_{bestH}, i_{bestL}, so ist auch hier die oben beschriebene Tendenz zu erkennen: Je höher das Rauschen in der Funktion niedriger Güte ist, desto weniger Individuen niedriger Güte werden erzeugt.

In Abbildung 6.4 wird der Verlauf des Entscheidungskritieriums über der Anzahl bewerteter Individuen gezeigt. Die schwarze Linie beschreibt die Entscheidungsgrenze: Liegt der Wert des Kriteriums darüber, so wird ein Individuum niedriger Güte berechnet und darunter eines hoher Güte. Die rote Kurve (Testfall 1a) ist ein besonderer Fall, da hier die Funktionen niedriger und hoher Güte identisch sind. Allerdings ist die Funktion niedriger Güte erheblich schneller. Aus diesem Grund sollte die Entscheidungsfunktion nur noch Individuen niedriger Güte vorschlagen. Weiterhin muss der Wert des Kriteriums in etwa mit dem Wert des zeitlichen Quotienten übereinstimmen, da in diesem Testfall die Trainingszeit t_{train} und Optimierungszeit auf dem Ersatzmodell t_{opti} auf 0 gesetzt wurden.

Die gemessenen Werte liegen am Ende der Optimierung bei einer durchschnittlichen Prozesskettenzeit von $\frac{t_{prh}}{t_{prl}} = 64.464$ und das Kriterium der Entscheidungsfunktion liegt bei $f_{dec}(M) = 64.81$. Bis auf kleine Abweichungen trifft das erwartete Ergebnis also zu. Abbildung 6.4 zeigt dieses Verhalten nochmals deutlich. Das Kriterium ist nach ei-

ner anfänglichen Initialisierungsphase nahezu konstant und entscheidet sich erst nach Erreichen einer sehr hohen Genauigkeit für die Prozesskette hoher Güte.

Die beiden anderen Optimierungen 1b und 1c wechseln wie erwartet früher auf die Prozesskette hoher Güte. Zudem sinkt der Anteil an berechneten Individuen niedriger Güte mit steigendem Rauschen der Funktion.

Testfall 2 - Einfluss der Optimierungs-Initialisierung: Dieser Testfall soll den Einfluss der Initialisierung bei einer komplexen Funktion niedriger Güte zeigen. Komplex bezieht sich in diesem Fall darauf, dass alle Kovarianzmodellparameter benötigt werden, um diese Funktion darstellen zu können. Weiterhin sind die Minima der Funktion niedriger und hoher Güte an verschiedenen Orten. Als Startdatenbasis wurde eine sehr kleine Menge von Stützstellen hoher Güte gewählt und die Menge an Stützstellen niedriger Güte variiert. Die verwendete Entscheidungsfunktion war wiederum die in dieser Arbeit entwickelte „Zusätzliche Gewichtung der Ersatzmodelle“ aus Kapitel 3.2.2.

	i_{bestH}	i_{bestL}	i_{initH}	i_{initL}	t_{fact}
Referenz	116	0	60	0	1
1a	12	377	60	200	2.2
1b	24	214	60	200	1.96
1c	85	41	60	200	1.19

Tabelle 6.2: Anzahl der jeweiligen erzeugten Member hoher und niedriger Güte bis zum Erreichen des Optimums.

Ziel ist es zu zeigen, wie das Verfahren bei unterschiedlichen Initialisierungen reagiert. Hierbei ist das Zusammenspiel von Entscheidungsfunktion und Co-Kriging von erheblicher Bedeutung. Die Entscheidungsfunktion kann nur gut arbeiten, wenn das Co-Kriging zuverlässige Vorhersagen trifft. Besonders am Anfang der Optimierung kann dies problematisch sein, da oftmals nur eine unzureichende Menge an Initialisierungsdaten vorliegt und das Co-Kriging damit die Hyperparameter noch nicht optimal trainieren kann. Hier wird ebenfalls der Extremfall betrachtet, dass bereits genügend Initialisierungsdaten niedriger Güte existieren und die Entscheidungsfunktion dann keine zusätzlichen Daten niedriger Güte mehr erzeugen sollte.

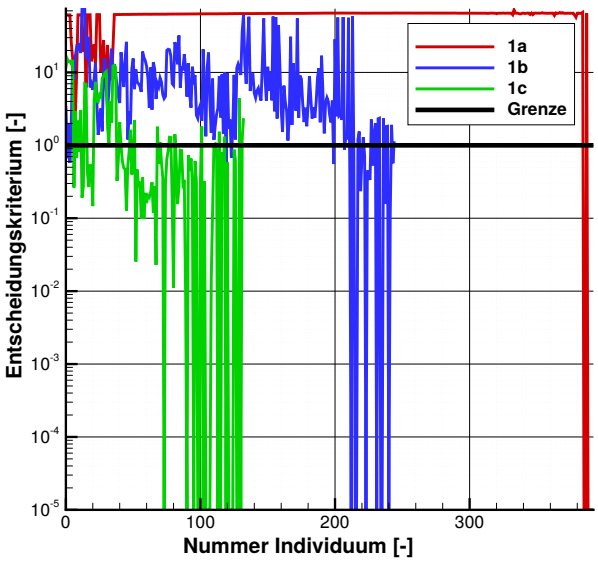


Abbildung 6.4: Verlauf des Kriteriums (logarithmische Achsenskalierung) der Entscheidungsfunktion für die Testfälle 1a - 1c. Die Anzahl der berechneten Individuen beinhaltet beide Gütestufen ohne die Initialisierung. Ein Kriterium größer 1 führt zu einer Entscheidung niedriger Güte und unter Eins zu hoher Güte, die Grenze ist als schwarze Linie eingezeichnet.

Als konvergiert gilt die Optimierungsaufgabe, sobald ein Wert von $f_{hf}(\vec{x}) < 1.248$ unterschritten wurde. Weiterhin wurde für die Entscheidungsfunktion die Trainingszeit und die Zeit für die Optimierung auf dem Ersatzmodell auf 0 gesetzt, da diese aufgrund der extrem niedrigen Prozess-

kettenzeiten überwiegen würde. Tabelle 6.3 zeigt die Anzahl der erzeugten Individuen der jeweiligen Testfälle 2(a-g). Typischerweise durchläuft eine solche Optimierung drei Entscheidungsphasen:

1. Zu Beginn ist die Schätzung der Hyperparameter aufgrund der niedrigen Datenlage oftmals instabil. Es gilt also genügend Samples zu erzeugen, bis die Hyperparameter stabil bleiben. Die Entscheidung fällt in dieser Phase in der Regel nicht optimal aus.
2. Ist eine ausreichend genaue Schätzung der Hyperparameter erreicht, so wird die Entscheidungsfunktion ab dieser Phase viele Individuen niedriger Güte erzeugen, sofern diese genügend Informationsgehalt bieten. Hierdurch wird eine Abtastung des Raums durch „günstige“ Stützstellen niedriger Güte erreicht.
3. Gegen Ende der Optimierung schwenkt die Entscheidungsfunktion auf Individuen hoher Güte um, da die niedrige Güte keine zusätzliche Information mehr über die Lage des Optimums liefern kann.

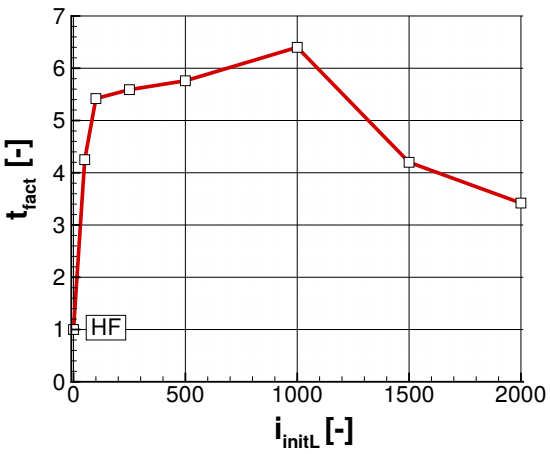


Abbildung 6.5: Beschleunigung im Vergleich zu einer Single-Fidelity-Optimierung über der Anzahl initialer Stützstellen niedriger Güte

Je nach initialer Datenlage variiert die Länge der drei genannten Phasen. Abbildung 6.5 zeigt die Beschleunigung über der Anzahl an initialen Individuen. Hier sind drei Bereiche zu erkennen:

- Testfall 2(a): In diesem Bereich ist die Anzahl an initial erzeugten Individuen noch sehr gering. Da die Hyperparameter mit dieser geringen Anzahl noch nicht gut geschätzt worden sind, fällt Entscheidungsphase 1 verhältnismäßig groß aus. Als Beispiel für dieses Verhalten ist in Abbildung 6.6 der Skalierungsfaktor a des Ersatzmodells über der Trainingsnummer zu erkennen. Es wird jedes mal trainiert, sobald ein neues Individuums berechnet wurde. Man sieht, dass bei Testfall a der Skalierungsfaktor bis ca. 28-30 Trainings noch sehr stark schwankt. Bei Testfall d sind diese Schwankungen von Beginn an kleiner und diese Entscheidungsphase bereits nach 10 Trainings durchlaufen.
- Testfall 2(b-d): In diesem Bereich fällt die Entscheidungsphase 1 sehr kurz aus. Die initiale Datenlage ist bereits sehr gut, sodass Entscheidungsphase 2 schnell erreicht wird. Die Entscheidungsfunktion arbeitet stabil und erzeugt hauptsächlich Individuen niedriger Güte, bis Entscheidungsphase 3 erreicht und das Optimum mit wenigen Individuen hoher Güte gefunden wird.
- Testfall 2(e-g): In diesem Bereich fallen die Entscheidungsphasen 1-2 fast voll-

ständig weg. Die initiale Anzahl an Individuen ist so groß, dass das Ersatzmodell von Anfang an nahezu perfekte Vorhersagen trifft. Die Entscheidungsfunktion geht aus diesem Grund direkt in Entscheidungsphase 3 über und erzeugt nur noch Individuen hoher Güte. Das Optimum wird dann mit einer sehr geringen Anzahl von Individuen hoher Güte schnell gefunden. Allerdings ist bei Testfall f und g die Anzahl an Individuen niedriger Güte bereits höher als benötigt, sodass der Beschleunigungsfaktor t_{fact} wieder sinkt.

Bezeichnung	i_{bestH}	i_{bestL}	i_{initH}	i_{initL}	t_{fact}
Referenz	139	0	10	0	1
a	22	162	10	50	4.25
b	14	142	10	100	5.42
c	12	77	10	250	5.59
d	8	52	10	500	5.76
e	5	1	10	1000	6.4
f	4	0	10	1500	4.2
g	5	0	10	2000	3.42

Tabelle 6.3: Anzahl der jeweiligen erzeugten Individuen hoher und niedriger Güte bis zum Erreichen des Optimums, vor und nach der Initialisierungsphase.

Dieser Testfall zeigt das Verhalten der Entscheidungsfunktion bei sehr unterschiedlichen Startbedingungen der hier gestellten Optimierungsaufgabe, bei der immer ein hoher Beschleunigungsfaktor erzielt werden konnte. Die unterschiedliche Anzahl an Individuen niedriger Güte für die Initialisierung wurde durch die Entscheidungsfunktion also effektiv ausgeglichen. Besonders bei einem unerfahrenen Anwender kann dieses Verhalten von Vorteil sein, da eine sinnvolle Anzahl der initialen Individuen nur schwer abschätzbar ist.

Testfall 3 - Hohe und niedrige Güte vollständig unkorreliert Dieser Testfall stellt einen Grenzfall dar: Das Co-Kriging-Verfahren muss lernen, dass die Funktion niedriger Güte keinen Zusammenhang zur Funktion hoher Güte besitzt. Die Entscheidungsfunktion erzeugt dann nur noch Individuen hoher Güte. Diese Multifidelity-Optimierung kann nur so gut werden wie eine reine Optimierung hoher Güte, da keine Beschleunigung durch die Funktion niedriger Güte zu erreichen ist. Besonders interessant ist hier der zeitliche Verlust durch eine Multifidelity-Optimierung im Vergleich zu einer reinen Single-Fidelity-Optimierung.

Tabelle 6.4 zeigt die Ergebnisse der Optimierung. Die Entscheidungsfunktion hat keine zusätzlichen Individuen niedriger

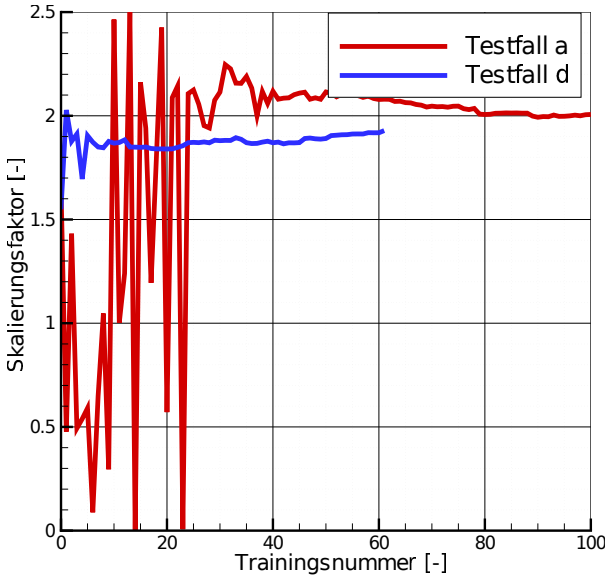


Abbildung 6.6: Konvergenz des Skalierungsfaktors a für Testfall a und d.

Güte erzeugt, was die optimale Lösung
für dieses Problem darstellt. Die Optimierung ist insgesamt ca. 2% langsamer als eine reine Optimierung hoher Güte. Dieser Testfall gibt einen Hinweis auf die Robustheit des vorgeschlagenen Verfahrens, da der zeitliche Verlust gering gehalten wurde und schnell erkannt wurde, dass die Prozesskette niedriger Güte keinen Informationsgehalt liefert.

6.2 Optimierung einer Fan-Stufe

Innerhalb dieses Abschnitts wird ein Vergleich zwischen einer Multifidelity- und einer Single-Fidelity-Fanstufenoptimierung dargestellt. Als Basis für die hier gezeigten Optimierungen dient die Arbeit von [Reimer, 2016], welche im Rahmen dieser Dissertation betreut wurde. Dort wird die aeromechanische Optimierung einer modernen Fan-Stufe beschrieben und eine frühe Version der hier entwickelten Multifidelity-Verfahren getestet. Aufgrund der gemachten Weiterentwicklungen wurden die Optimierungen in [Reimer, 2016] nochmals mit der aktuellen Version wiederholt, wobei sich das Multifidelity-Verfahren in den folgenden Punkten vom aktuellen Stand unterscheidet:

	i_{bestH}	i_{bestL}	i_{initH}	i_{initL}	t_{fact}
HF	139	0	10	0	
MF	142	0	10	50	0.98

Tabelle 6.4: Anzahl der jeweiligen erzeugten Individuen hoher und niedriger Güte bis zum Erreichen des Optimums, vor und nach der Initialisierungsphase.

- Die Entscheidungsfunktion berücksichtigt aktuell die Gewichtung der Nebenbedingungen und Zielfunktionen.
- Das Co-Kringing-Modell bietet im Vergleich zu [Reimer, 2016] die Nutzung von Rauschtermen für jede Gütestufe und einem Skalierungsfaktor (siehe Kapitel 5.1). Weiterhin wurde das Initialisierungsverfahren, welches in Kapitel 5.3.3 „Initialisierung auf Basis bereits vorhandener Kriging-Modelle“ beschrieben wird, verwendet.
- Das Trainingsverfahren wurde erheblich beschleunigt (siehe hierfür Kapitel 5).
- Für die Optimierung auf dem Ersatzmodell wurde bei [Reimer, 2016] eine Restriktion auf die Wahrscheinlichkeit die Zielfunktionen und Nebenbedingungen zu verbessern, gesetzt. Die Wahrscheinlichkeit musste mindestens 80% betragen. Damit konnten Konvergenzprobleme bei den CFD-Simulationen verringert werden. Um diese Restriktion erfüllen zu können, muss der Optimierungsalgorithmus „Erwarteter Volumenzugewinn“ immer zu kleinen vorhergesagten Varianzen tendieren. Dies wiederum bedeutet, dass sich die Suche auf dem Ersatzmodell vom Startpunkt kaum wegbewegt und das explorative Vorgehen dadurch stark eingeschränkt wird. Bei einer Multifidelity-Optimierung sollten aber besonders dieses „explorative“ Verhalten genutzt werden, da mit wenig Aufwand eine Schätzung gegeben werden kann. Aus diesem Grund wurde diese Restriktion in diesem Anwendungsbeispiel nicht mehr verwendet.

6.2.1 Optimierungssetup

Hardwareumgebung: Jede Optimierung erhielt 6+1 Knoten auf dem DLR-AT Cluster, wobei ein Knoten folgende Konfiguration besaß:

- 2x Intel® Xeon E5-2695 v2 mit 2.4GHz
- 128GB ECC RAM

Für die Prozesskettenberechnung waren 6 Knoten vorgesehen. Das Training fand auf jeweils einem eigenen Knoten statt.

Prozesskette der Gütestufen: In Tabelle 6.5 ist die Abfolge der Prozesskettenschritte für beide Gütestufen und die mittleren Ausführungszeiten (t_h für die hohe und t_l für die niedrige Gütestufe) dargestellt. Beide Gütestufen unterscheiden sich durch die Auflösung des CFD-Netzes. Der Laufzeitunterschied zwischen den Prozessketten liegt bei einem Faktor von ungefähr 10.

Prozess	t_h	t_l
Strömungskanal	0.7s	0.7s
Schaufelgeometrie	9s	9s
CFD-Netze	30.5s	14s
CFD-Simulation OP0	1600s	140s
CFD-Simulation OP1	2100s	180s
FEM-Simulation	17.5s	17.5s
Gesamt	3760s	360s

Tabelle 6.5: Prozesskette und die ungefähren Ausführungszeiten der einzelnen Schritte für hohe und niedrige Güte.

Die Prozesskette startet in beiden Gütestufen mit der Geometrieerzeugung. Mit diesen Geometriedaten wird dann das CFD-Rechennetz erzeugt, wobei sich das Rechennetz der verschiedenen Gütestufen durch die Auflösung unterscheidet. Die Strömungssimulation erfolgt in zwei Betriebspunkten (OP0 und OP1). Schwanken der Wirkungsgrad, der Massenstrom und das Totaldruckverhältnis während der letzten 100 Zeitschritte um weniger als 0.01%, so gilt die Rechnung als konvergiert.

Als letzter Schritt der Prozesskette wird eine strukturmechanische Simulation des Rotors durchgeführt. Hierfür wird die maximale Dehnung der Schaufel bestimmt und als Nebenbedingung verwendet. Das verwendete FEM-Netz ist für beide Gütestufen identisch und die Druckverteilung wird auf der Schaufeloberfläche aus dem jeweiligen CFD-Ergebnis ermittelt. Die Auswirkungen der unterschiedlichen Druckverteilungen ist im Vergleich zu den auftretenden Fliehkräften aber sehr gering.

Unterschied hohe und niedrige Gütestufe: Der Unterschied zwischen der hohen und der niedrigen Gütestufe liegt in der Auflösung des CFD-Netzes. In Tabelle 6.6 sind einige Netzparameter aufgelistet. Der Faktor der Zellenanzahl liegt bei etwa 8.1 und die zeitliche Differenz der gesamten Rechnung liegt bei einem Faktor von ca. 10. Die Anzahl der Zellen in axialer Richtung wurde halbiert. Damit kommt es zu einem schnelleren Informationstransport innerhalb des Rechennetzes, womit die Konvergenz beschleunigt wird.

Die Anzahl der Zellen im Spaltblock wird bei der hohen Gütestufe verdreifacht und die radiale Auflösung ungefähr verdoppelt. Dadurch können Sekundärströmungseffekte besser modelliert werden. In Anhang A.4.2 ist eine exemplarische Abbildung der beiden Rechennetze dargestellt.

Bereich	Hohe Güte	Niedrige Güte
Gesamt	$1.1 \cdot 10^6$	$1.36 \cdot 10^5$
Axiale Auflösung - Rotor	123	65
Axiale Auflösung - Stator	137	65
Radiale Auflösung	55	25
Auflösung - Spaltblock	4896	1560

Tabelle 6.6: Unterschiede in den CFD-Netzen zwischen den Gütestufen

Nebenbedingung			Min.	Max.
$\dot{m}_{OP0}$	[kg/s]	Massenstrom OP0	557.0	559.5
$\dot{m}_{OP1}$	[kg/s]	Massenstrom OP1	518.5	520.0
$\Pi_{t,OP1}$	[-]	Totaldruckverh.OP0	1.397	1.399
$\lvert\theta_{out,OP0}\rvert$	[°]	Abströmwinkel OP0	0	3
ε	[-]	max. Rotordehnung	0	0.0025

Tabelle 6.7: Übersicht aller Nebenbedingungen der Optimierung

Zielfunktionen und Nebenbedingungen: Für die Optimierung werden zwei Zielfunktionen festgesetzt. Zum einen die Maximierung des isentropen Wirkungsgrads $\eta_{is} = \frac{\Pi_{t,OP0}^{\frac{\kappa-1}{\kappa}}-1}{\frac{T_{t,out}}{T_{t,in}}-1}$ im Betriebspunkt OP0 und zum anderen die Maximierung der Kennliniensteigung $\frac{\triangle\Pi_t}{\triangle\dot{m}} = \frac{\Pi_{t,OP1}-\Pi_{t,OP0}}{\dot{m}_{OP1}-\dot{m}_{OP0}}$. Weiterhin werden 5 Nebenbedingungen festgelegt, welche in Tabelle 6.7 zusammengefasst sind.

Die Kennliniensteigung $\frac{\triangle\Pi_t}{\triangle\dot{m}}$ ist in Abbildung 6.7 grafisch dargestellt. Da die Maximierung der Steigung nur durch Erhöhung des Druckverhältnisses $\Pi_{t,OP1}$ erreicht wird, werden die Massenströme $\dot{m}_{OP0}$, $\dot{m}_{OP1}$ und das Totaldruckverhältnis $\Pi_{t,OP0}$ in sehr engen Intervallen festgelegt. Diese Bereiche sind in Abbildung 6.7 in grau dargestellt.

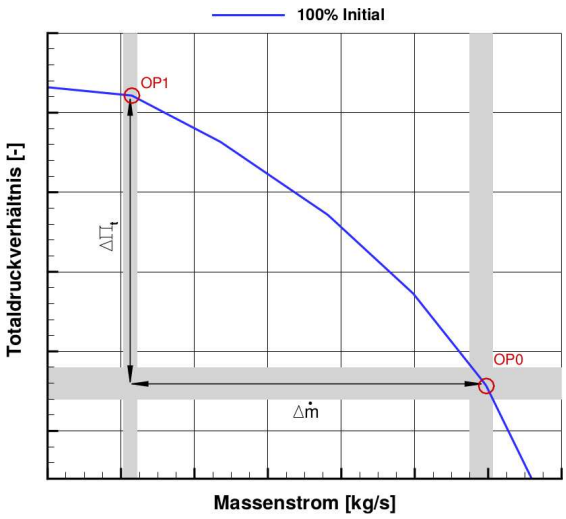


Abbildung 6.7: Darstellung der Betriebspunkte OP0 und OP1 auf der 100%-Drehzahllinie. Der graue Bereich beschreibt die Nebenbedingungen auf den Massenstrom und das Totaldruckverhältnis. Quelle: [Reimer, 2016]

Eine weitere Nebenbedingung ist der absolute Abströmwinkel der Fanstufe $|\theta_{out,OP0}|$. Dieser wird auf maximal 3° festgelegt, um eine axiale Abströmung zu erhalten. Alle aerodynamischen Größen sind massenstromgemittelt. Strukturmechanisch wird die maximale Dehnung ε des Rotors auf 0.25% beschränkt.

Damit ergeben sich insgesamt 7 Ersatzmodelle, welche für die Flowparameter $\Pi_{t,OP1}, \Pi_{t,OP0}, \dot{m}_{OP1}, \dot{m}_{OP0}, T_{t,out}, \theta_{out,OP0}, \varepsilon$ trainiert werden müssen.

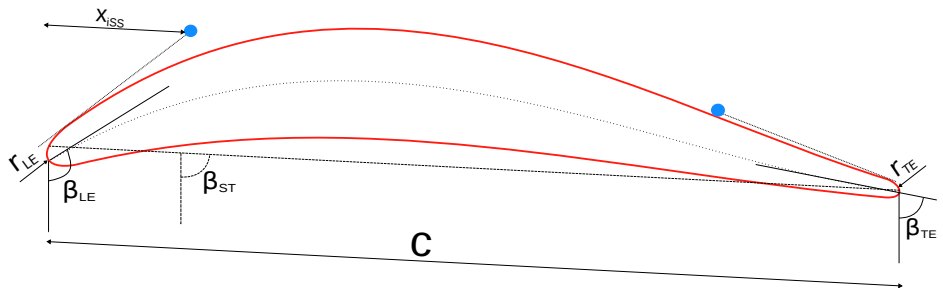


Abbildung 6.8: Exemplarische Darstellung der freigegebenen Profilparameter

Variable		Anzahl
Strak	Nabe	4
	Gehäuse	4
Rotor	3x Profile	3x9
	Rad. Pos. Profil 2	1
Stator	3x Profile	3x9
	Rad. Pos. Profil 2	1
Gesamt		64

Abbildung 6.9: Überblick über alle freien Variablen der Optimierung

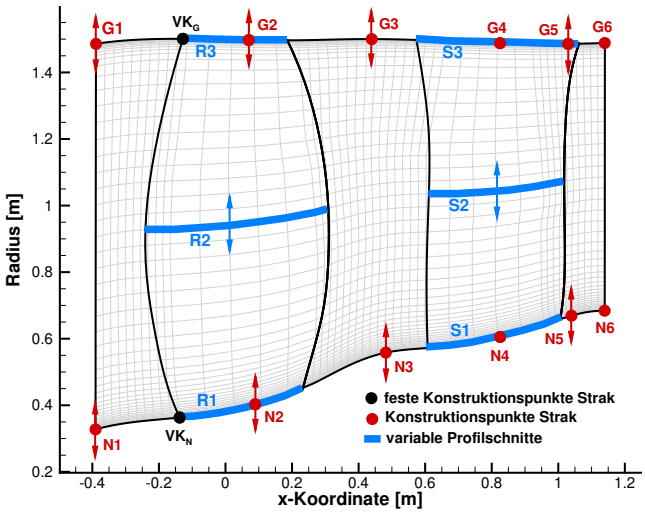


Abbildung 6.10: Freigegebene Konstruktionspunkte am Gehäuse G1,G2,G3,G5 und an der Nabe N1,N2,N3,N5 des Straks in rot dargestellt und freigegebene Profilschnitte in blau

Parametrisierung: Für die geometrische Beschreibung einer Fanstufe ist eine Vielzahl von Parametern notwendig. Daraus wurde eine Auswahl getroffen, welche in der Optimierung in einem festgelegten Intervall frei gewählt werden können.

Abb. 6.10 zeigt den Strakverlauf der Fanstufe, also die Nabenkontur, den Gehäuseverlauf und die Position der Rotor- und Statorschaufel. Die Naben- und Gehäusekontur wird durch Konstruktionspunkte bestimmt. Bei jeweils vier Punkten wurde die y-Koordinate zum freien Parameter deklariert. Mit dieser Wahl ist eine ausreichende Gestaltungsfreiheit der Gehäusekontur gegeben und gleichzeitig die notwendige aerodynamische Vergleichbarkeit der verschiedenen Individuen gesichert.

Die Schaufelform wird hier durch die Beschreibung des Naben-, des Gehäuse- und des Mittelschnittprofiles definiert (blaue Linien in Abb.6.10). Wie diese Schaufelprofile definiert werden, wird mit Hilfe der Abb. 6.8 beschrieben.

Als Kompromiss zwischen dem notwendigen Gestaltungsfreiraum für die Profilform bei einer handhabbaren Gesamtzahl von freien Optimierungsvariablen wurden die folgenden Profilparameter gewählt:

Vorder- und Hinterkantenradien r_{LE}, r_{TE}, Vorder- und Hinterkantenwinkel β_{LE}, β_{TE}, Staffelungswinkel β_{ST}, Sehnenlänge c und zwei Konstruktionspunkte, welche die Form der Saugseite beschreiben. Weiterhin kann die radiale Position des mittleren Profils in engen Grenzen geändert werden.

Tabelle 6.9 fasst die freien Optimierungsparameter und deren Anzahl zusammen. In Summe wird also eine optimale Fanstufenform in einem 64-dimensionalen Raum gesucht. Weitere Informationen über die Parametrisierung sind in [Reimer, 2016] zu finden.

Einstellungen Optimierung und Laufzeitumgebung:

- Es wurden 25 Individuen hoher Güte und 75 Individuen niedriger Güte bestimmt. Diese Startdatenbasis wurde einmal erzeugt und diente dann als Initialisierung für alle durchgeführten Optimierungsläufe. Die Bestimmung der Parameter erfolgte zufällig gleichverteilt.
- Keiner der initialen Individuen erfüllte alle Nebenbedingungen.
- Die Optimierung wurde einmal gestartet und dann ohne Eingriffe durchgeführt.
- Die Optimierung auf dem Ersatzmodell wurde mit jeweils 5000 Iterationen durchgeführt und als Optimierungsalgorithmus der „Erwartete Volumenzugewinn“ (siehe Kapitel 2.3.2) verwendet. Jedes Individuum wurde mit diesem Algorithmus erzeugt.
- Jedes Optimierungssetup wurde 4x wiederholt, um Zufälligkeiten abbilden zu können.
- Jede Optimierung bekam eine feste Laufzeit von 550 Stunden zugewiesen.
- Es wurde ein minimaler prozentualer Anteil von 10% an Individuen hoher Güte festgelegt. Hierfür wurde das in Kapitel 5.4 gezeigte Verfahren verwendet. Zusätzlich wurde eine weitere Optimierung mit einem Anteil von 5% durchgeführt, um das Verhalten der Entscheidungsfunktion zu zeigen.

Es werden zwei Optimierungssetups verglichen:

1. Singlefidelity-Optimierung als Referenz. Ersatzmodell: Ordinary-Kriging.
2. Multifidelity-Optimierung mit der „Zusätzliche Gewichtung der Ersatzmodelle“-Entscheidungsfunktion (siehe Kapitel 3.2.2.2).
   (a) Minimaler prozentualer Anteil an Individuen hoher Güte 10%.
   (b) Minimaler prozentualer Anteil an Individuen hoher Güte 5% (keine Wiederholung).

6.2.2 Ergebnisse

Innerhalb dieses Abschnittes werden die Ergebnisse aller Optimierungen miteinander verglichen und einige davon detailliert beschrieben. Im letzten Teil werden die aerodynamischen Ergebnisse ausgewählter Individuen betrachtet.

Ergebnisse der Testoptimierungen im Vergleich In Abbildung 6.11 ist der zeitliche Verlauf des Volumenzugewinns dargestellt. Bei der Zeit handelt es sich um die

kummulierte Prozesskettenzeit aller Gütestufen. Die Trainingszeit ist dabei unberücksichtigt und wird in Abbildung 6.12 gesondert dargestellt. Weiterhin sind die Zeitpunkte nach der Initialisierung der Multi- und High-Fidelity-Optimierungen eingezeichnet. Die zeitliche Verzögerung der Multifidelity-Optimierungen ergibt sich aus dem zusätzlichen Aufwand, die Startindividuen niedriger Güte zu erzeugen. Die blauen Linien beschreiben die Verläufe der Optimierungen 2a (siehe Kapitel 6.2.1). Die grüne Linie entspricht der Optimierung 2b mit einem minimalen prozentualen Anteil an Individuen hoher Güte von 5%. Die schwarzen Linien entsprechen den High-Fidelity-Referenz-Optimierungen.

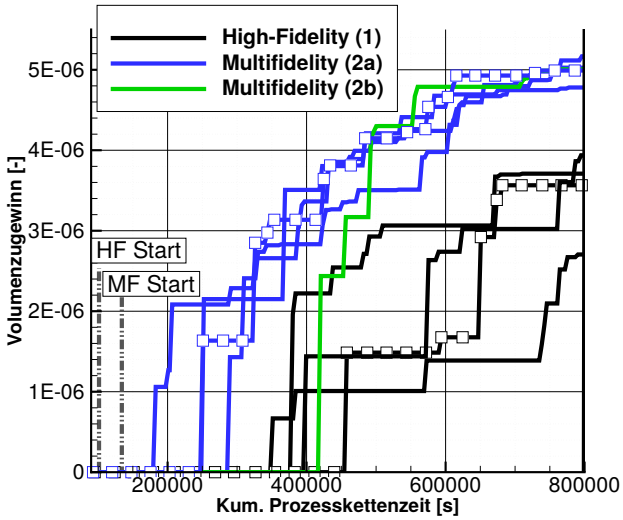


Abbildung 6.11: Zeitlicher Verlauf des Volumenzugewinns der verschiedenen Testoptimierungen bis zur Abbruchzeit. Die mit Symbolen markierten Linien werden im späteren Verlauf noch genauer ausgewertet.

Die mit Symbolen gekennzeichneten Optimierungen werden im nächsten Abschnitt noch genauer untersucht.

Die Optimierungstestläufe 2a erzeugen zuerst einen Volumenzugewinn. Die Ursache dafür liegt in der raschen Erfüllung der Nebenbedingungen. Im Vergleich dazu ist der Anstieg von Optimierung 2b stark verzögert, da die Entscheidungsfunktion zu Anfang der Optimierung mehr Individuen niedriger Güte erzeugt.

Trotzdem kann dasselbe Niveau erreicht werden wie bei den anderen Multifidelity-Optimierungen. Dies zeigt die volle Funktionalität der Entscheidungsfunktion, welche die unterschiedlichen Startbedingungen sinnvoll ausgleichen kann. Betrachtet man den mittleren Volumenzugewinn der Optimierungen hoher Güte bei 800.000s, so erreichten die Multifidelity-Optimierungen denselben Volumenzugewinn bereits nach ca. 400.000s und sind damit fast doppelt so schnell. Weiterhin ist zu erkennen, dass alle Multifidelity-Optimierungen zur Abbruchzeit einen deutlich höheren Volumenzugewinn erzielt haben. Zudem ist die Streuung des Volumenzugewinns bei allen Multifidelity-Optimierungen erheblich geringer. Die Highfidelity-Optimierungen zeigen eine vergleichsweise starke Streuung. Dieses Verhalten kann für industrielle Anwen-

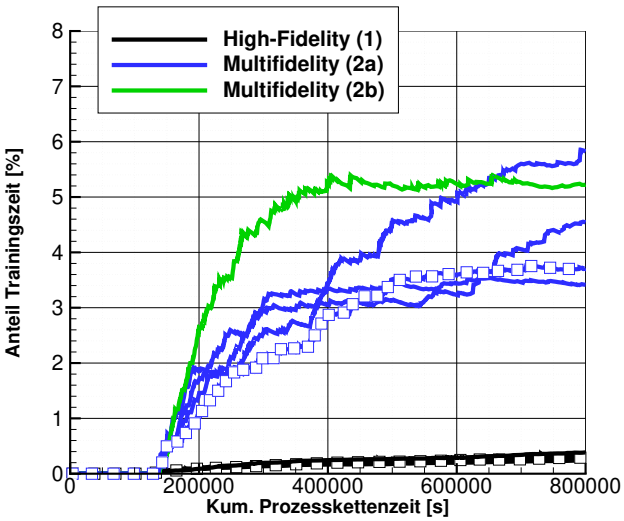


Abbildung 6.12: Prozentualer Anteil der Trainingszeit an der Gesamtlaufzeit der verschiedenen Testoptimierungen. Die mit Symbolen markierten Linien werden im späteren Verlauf noch genauer ausgewertet.

dungen von großer Bedeutung sein, ein Beispiel hierfür ist in Kapitel 6.3.4 zu finden.

Abbildung 6.12 stellt die Trainingszeit der einzelnen Testoptimierungen als prozentualen Anteil an der Gesamtzeit dar. Die schwarzen Kurven markieren den Anteil der High-Fidelity-Optimierungen, welcher unter einem Prozent liegt und damit vernachlässigbar ist. Die Multifidelity-Optimierungen beanspruchen mit 4% einen deutlich höheren Zeitanteil. Die Multifidelity-Optimierung 2b liegt aufgrund der höheren Stützstellenanzahl bei ca. 5%. Die höheren Trainingszeiten entstehen hauptsächlich durch die größere Anzahl an Trainingsmustern. Außerdem ist auch das Trainingsverfahren durch eine höhere Anzahl an Hyperparametern etwas aufwendiger. Weiterhin ist bei den Multifidelity-Optimierungen eine Abflachung der Zeitkurven zu erkennen, welche durch die „Restart-Option auf Basis von vorhandenen Modellen“ (siehe Kapitel 5.3.3) verursacht wird. Sobald genügend trainierte Modelle vorhanden sind und diese die Bedingungen für den „Restart“ erfüllen, können sie zur Initialisierung der Ersatzmodelle verwendet werden und beschleunigen das Training so erheblich.

Die Multifidelity-Optimierungen erreichen den Volumenzugewinn der Referenz bei etwa der halben Zeit bzw. zum gewählten Abbruchzeitpunkt einen deutlich höheren Wert. Trotz einer Vervielfachung der notwendigen Trainigszeit ist diese Art der Multifidelity-Optimierung in diesem Beispiel sehr lohnenswert.

Diese Aussage wird für eine Vielzahl von Turbomaschinenanwendungen gelten, wenn die Fidelitystufen durch die Variation der Netzauflösung erzeugt werden. Diese Prozessketten sind für den Anwender schnell generiert und weisen eine geringe Fehleranfälligkeit auf. Dies wird durch eine unüberschaubare Anzahl von Veröffentlichungen zum Netzeinfluss auf die Strömungsergebnisse belegt.

Detail-Ergebnisse ausgewählter Optimierungen In diesem Abschnitt werden die in Abbildung 6.11 gekennzeichneten Optimierungen detaillierter betrachtet. Als erstes wird die Entwicklung der Nebenbedingungen verglichen. Diese zu erfüllen, stellt einen wesentlichen Anteil der Optimierungslaufzeit dar. Abbildung 6.13 zeigt den Verlauf des summierten Restriktionsterms über der Optimierungslaufzeit. Aufsummiert werden die Restriktionen aus Tabelle 6.7, welche nicht erfüllt wurden. Sind alle Restriktionen erfüllt, erreicht dieser Term den Wert Null. Die Restriktionsterme wurden nicht normiert. Die schwarze Kurve beschreibt die High-Fidelity-Optimierung. Trotz des späteren Starts erfüllen beide Multifidelity-Optimierungen die Restriktionen deutlich schneller und können von den zusätzlichen Individuen niedriger Güte profitieren. Optimierung 2b erzeugt zu Anfang die meisten Individuen niedriger Güte und erfüllt die Restriktionen am schnellsten.

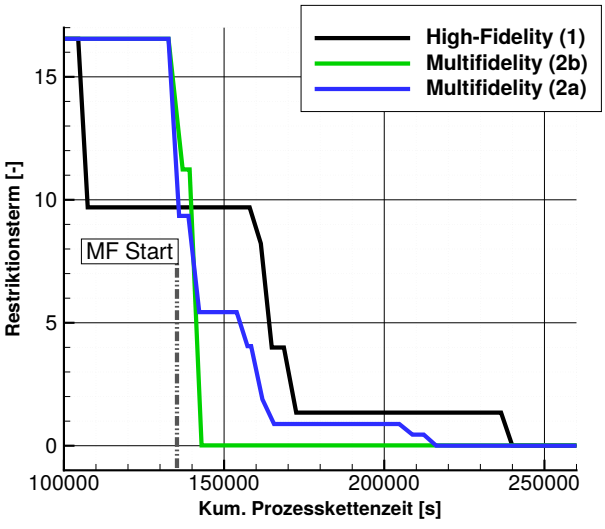


Abbildung 6.13: Summierter Restriktionsterm über der Optimierungslaufzeit.

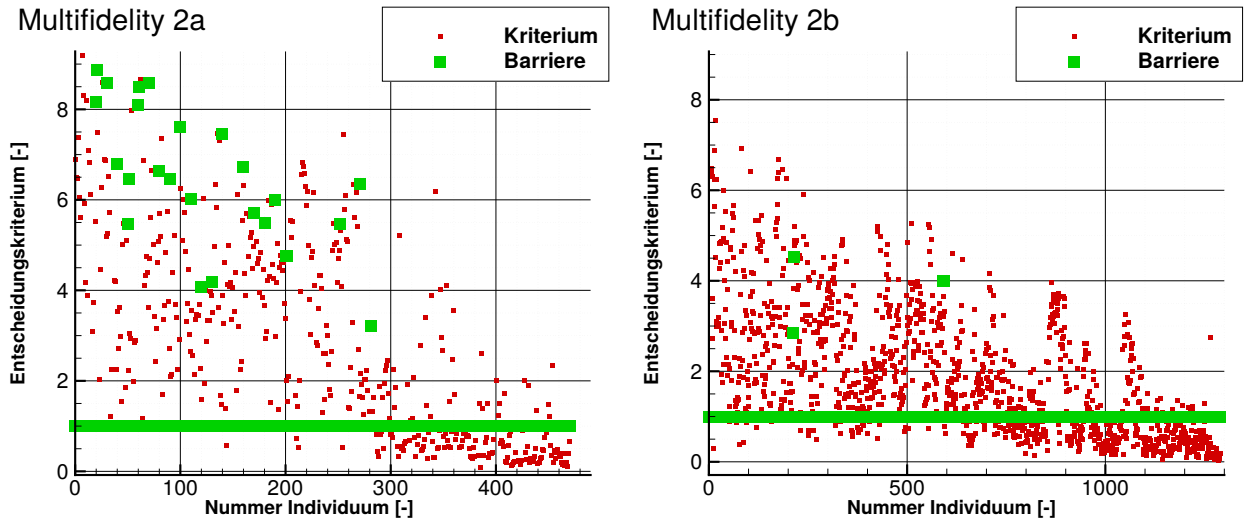


Abbildung 6.14: Verlauf der Entscheidungsfunktion der Optimierungen 2a und 2b. Die roten Symbole stellen den jeweiligen Wert der Entscheidungsfunktion dar und die grünen Punkte markieren die Barriere, wobei ein Überschreiten der Barriere zu einem Member niedriger Güte führt und ein Unterschreiten zu einem hoher Güte. Die unterchiedlichen Bereiche der X-Achse sind zu beachten.

Entsprechend der Ausführungen am Ende des letzten Abschnitts gilt in diesem Beispiel eine klare Abhängigkeit zwischen Erfüllung der Nebenbedingungen und der Anzahl an berechneten Individuen niedriger Güte.

Abbildung 6.14 zeigt den Verlauf der Entscheidungsfunktion von Optimierung 2a und 2b. Die Individuen-Nummer bezieht sich in diesem Fall auf alle Gütestufen. Die roten Punkte beschreiben die jeweiligen Werte des Entscheidungskriteriums und die grünen Punkte stellen den jeweiligen Wert der Barriere dar (siehe Kapitel 5.4 „Minimale Menge an Individuen hoher Güte“). Liegt das Kriterium über der Barriere, so wird zugunsten eines Individuums niedriger Güte entschieden. Liegt der Wert unterhalb der Barriere, dann wird ein Individuum hoher Güte erzeugt. Da bei Optimierung 2a der minimale prozentuale Anteil an Individuen hoher Güte bei 10% liegt und bei Optimierung 2b bei 5%, wird die Barriere bei Optimierung 2a öfter verschoben, um den minimalen Prozentsatz zu erfüllen. Die globale Tendenz des Kriteriums ist bei beiden Optimierungen erwartungsgemäß fallend. Da mit steigender Anzahl an Individuen die Ersatzmodellunsicherheiten immer kleiner werden, verschiebt sich das Kriterium weiter zu niedrigeren Werten und damit zugunsten der höheren Güte. Bei Optimierung 2b fällt der Wert des Entscheidungskriteriums erst nach höherer Individuenanzahl. Da die Individuen-Nummer allerdings auch Individuen niedriger Güte beinhaltet, darf diese nicht mit der Zeit gleichgesetzt werden.

In dem Verlauf von Optimierung 2a ist bei Nummer 300 ein leichter Sprung zu erkennen. Das Entscheidungskriterium zeigt ab dieser Stelle häufiger Werte unter Eins. Dies lässt sich dadurch erklären, dass mit Individuen-Nummer 299 das erste Individuum gefunden wurde, welches alle Restriktionen erfüllt. Die Wahrscheinlichkeit die Nebenbedingungen zu erfüllen, ist ab diesem Zeitpunkt erhöht und übt damit einen großen Einfluss auf die Entscheidungsfunktion aus (siehe Kapitel 3.2.2.2).

Abbildung 6.15 zeigt die Entwicklung der Zielfunktionen von Optimierung 1 und 2a zu

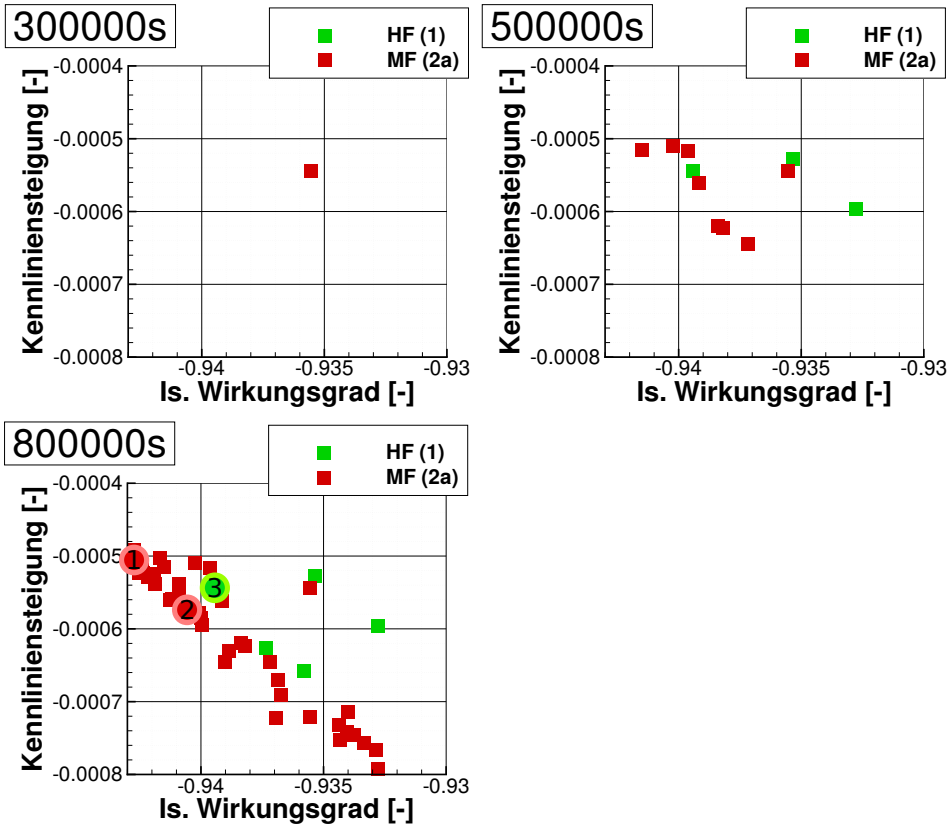


Abbildung 6.15: Zeitliche Entwicklung der High-Fidelity-Optimierung 1 und der Multifidelity-Optimierung 2a.

jeweils drei unterschiedlichen Zeitpunkten. Eingezeichnet sind Individuen hoher Güte, welche die Restriktionen bereits erfüllt haben. Der erste Zeitpunkt wurde bei 300.000 Sekunden gewählt (vgl. Abbildung 6.11), der zweite bei 500.000 Sekunden und der letzte bei 800.000 Sekunden. Bei 300.000 Sekunden wurde nur ein einziges Individuum der Multifidelity-Optimierung gefunden, welches die Restriktionen erfüllt. Bereits ab diesem frühen Zeitpunkt ist die MF-Optimierung der HF-Optimierung überlegen. Bei 500.000 Sekunden konnte die Multifidelity-Optimierung eine kleine Paretofront aufbauen und die ersten Individuen mit sehr hohem Wirkungsgrad auffinden. Die High-Fidelity-Optimierung konnte zu diesem Zeitpunkt erst drei Individuen finden, welche die Restriktionen erfüllen. Bei 800.000 Sekunden konnte die Multifidelity-Optimierung eine sehr breite Paretofront aufstellen. Die High-Fidelity-Optimierung hingegen besaß zu diesem Zeitpunkt eine weniger breite Paretofront und auch schlechtere Zielfunktionswerte. Beispielsweise war der maximale Wirkungsgrad der Multifidelity-Optimierung um ca. 0.5%-Punkte besser.

Einfluss des Anteils an Individuen hoher Güte auf den Optimierungsverlauf: Im Folgenden werden die Ergebnisse aller durchgeführten Optimierungen dafür verwendet, um den Einfluss des Anteils an Stützstellen hoher Güte zu quantifizieren. Abbildung 6.16 stellt die Ergebnisse grafisch dar, wobei jeder Punkt das Ergebnis einer weiteren Optimierung nach 800.000 Sekunden beschreibt. Auf der Abszisse wird der am Optimierungsende erreichte Anteil zwischen Stützstellen hoher n_h und niedriger Güte

n_l dargestellt $\frac{n_h}{n_h+n_l}$. Auf der Ordinate wird der erreichte Volumenzugewinn gezeigt. Es ist erkennbar, dass ein geringerer Anteil an Samples hoher Güte bei einem definierten Zeitumfang einen höheren Volumenzugewinn bewirkt. Die Prozesskette niedriger Güte muss bereits einen hohen Informationsgehalt besitzen. Die Streuung der einzelnen Optimierungen kommt zum einen durch einige Zufälligkeiten innerhalb des Optimierungsprozesses (Initialisierung der Ersatzmodelle und Optimierung auf den Ersatzmodellen) zustande und zum anderen durch die jeweiligen Entscheidungsverläufe der Optimierungen. Weiterhin ist zu beachten, dass der Volumenzugewinn Null ist, wenn kein Individuum hoher Güte berechnet wird.

Zusammenfassend ist erkennbar, dass der Anteil von Individuen niedriger Güte in diesem Optimierungssetup eine wesentliche Rolle spielt und dass ein Optimum an Individuen niedriger Güte existiert.

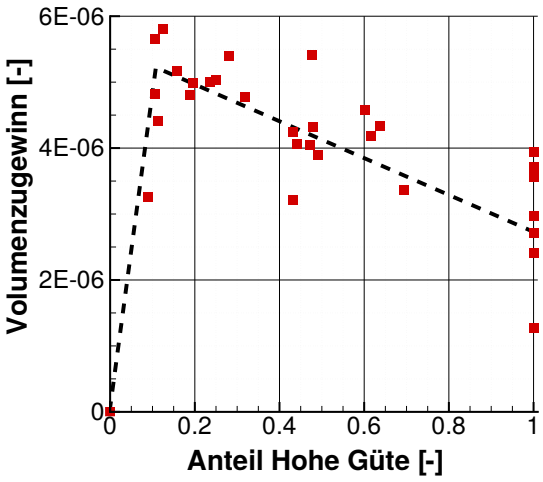


Abbildung 6.16: Vergleich verschiedener Optimierungen mit unterschiedlichen Entscheidungsverläufen.

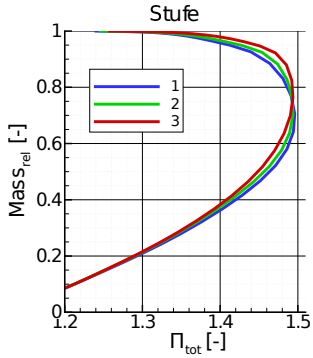
Aerodynamische Auswertung: In diesem Abschnitt werden die in Abbildung 6.15 mit 1,2,3 markierten Individuen aerodynamisch bewertet. Da es sich um einen Benchmark der Optimierungsverfahren handelt, liegt der Fokus bei dieser Auswertung auf den grundlegenden Unterschieden zwischen den Individuen, um den unterschiedlichen Optimierungsstand zu belegen.

Die Individuen wurden nach 800.000 Sekunden Prozesskettenlaufzeit ausgewertet, wobei die Optimierungen zu diesem Zeitpunkt noch nicht vollständig konvergiert waren. Die erreichten Zielfunktionswerte und Nebenbedingungen sind in Tabelle 6.8 aufgelistet. Die Individuen 1 und 3 haben jeweils den höchsten Wirkungsgrad innerhalb ihrer jeweiligen Optimierung.

Nummer	η_{is}	$\frac{\triangle \Pi_t}{\triangle \dot{m}}$	$\dot{m}_{OP0}$	$\dot{m}_{OP1}$	$\Pi_{t,OP1}$	$\lvert\theta_{out,OP0}\rvert$	ε
Unteres Limit			557	518.5	1.397	0	0
1	94,3%	0.0005067	558.24	519.86	1.3973	2.9	0.0024
2	94,1%	0.0005816	557.36	519.34	1.379	2.94	0.0024
3	93,9%	0.0005446	558.0	519.32	1.398	2.72	0.0016
Oberes Limit			559.5	520	1.399	3	0.0025

Tabelle 6.8: Werte der Zielfunktionen und Nebenbedingungen der einzelnen Individuen.

Individuum 2 wurde auf Basis von Individuum 3 ausgewählt und entspricht einer orthogonalen Verschiebung von Individuum 2 ausgehend von der High-Fidelity-Paretofront. In Abbildung 6.17 ist die relative Lage der einzelnen Nebenbedingungen für jedes Individuum zu erkennen. Die obere und untere Linie beschreiben die jeweiligen Intervallgrenzen. Die Multifidelity-Individuen 1 und 2 liegen häufiger an Grenzen von Neben-



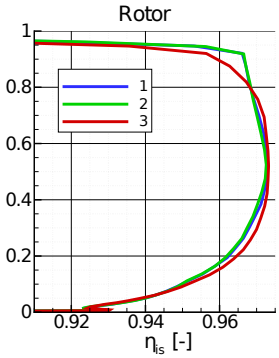
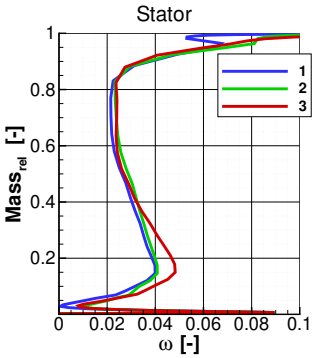


Abbildung 6.18: Radiale Verteilung des isentropen Wirkungsgrads der Stufe und des Rotors sowie des Totaldruckverhälnisses der gesamten Stufe.



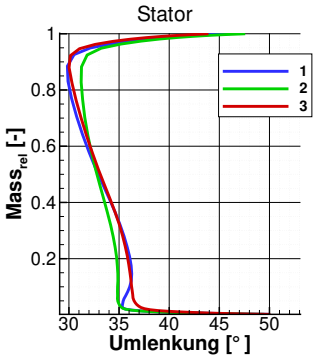


Abbildung 6.19: Radiale Verteilung des Totaldruckverlustbeiwerts und der Umlenkung des Stators.

bedingungen als das High-Fidelity-Individuum 3. Der Grund dafür liegt am größeren Optimierungsfortschritt. Der Optimierer geht z.B. mit der mechanischen Dehnung bis an die obere Grenze und nutzt somit das zulässige Maximum an Belastung aus. Bei der High-Fidelity-Optimierung wurden zu diesem Zeitpunkt erst 5 Individuen gefunden, welche die Nebenbedingungen überhaupt erfüllen.

Abbildung 6.18 zeigt die radialen Verteilungen der Stufe und des Rotors. Beim Totaldruckverhältnis ist bei beiden Multifidelity-Individuen eine Lastumverteilung zu erkennen: Im Bereich von ca. 90% relativen Massenstroms wird der Rotor entlastet und im mittleren Bereich dafür stärker belastet. Beide Rotoren erreichen dadurch einen höheren Wirkungsgrad im Bereich über 80% relativen Massenstroms. Im Bereich zwischen 15%-40% ist der Wirkungsgrad im Vergleich zu Individuum 3 etwas niedriger.

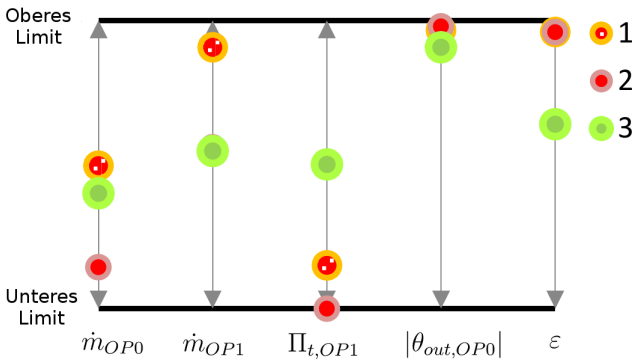


Abbildung 6.17: Relative Lage der Nebenbedingungen einzelner Individuen. Die obere Linie gibt das Maximum und die untere das Minimum der Nebenbedingung an.

Betrachtet man die radialen Verteilungen der Statoren in Abbildung 6.19, so fällt auf, dass die Individuen 1 und 3 eine relativ ähnliche Umlenkung haben. Individuum 1 besitzt allerdings die niedrigeren Totaldruckverluste und kann dadurch den Gesamtwirkungsgrad nochmals erhöhen. Individuum 2 lenkt im gehäusenahen Bereich etwas stärker um und reagiert damit auf die geänderten Zuströmbedingungen durch den Rotor. Dies ermöglicht Individuum 2, die geforderte Abströmung von max. 3° einzuhalten. Die Verluste sind auf einem leicht höheren Niveau als bei Individuum 1, aber niedriger als bei Individuum 3.

Abbildung 6.20 zeigt den Strakverlauf und die Vorder- und Hinterkanten der Beschau-

felung im S2-Schnitt. Bei den Rotoren der Individuen 1 und 2 ist deutlich erkennbar, dass diese eine geringere Sehnenlänge (ca. 25% kürzer) im gehäusenahen Bereich aufweisen als Individuum 3. Dagegen sind im mittleren Bereich die Sehnen um ca. 10% länger. Vergleicht man dies mit den Werten aus Abbildung 6.18, so kann durch die kürzeren Schaufeln im gehäusenahen Bereich ohne einen nennenswerten Einbruch im Totaldruckverhältnis der Wirkungsgrad erhöht werden. Beim Stator hingegen ist die Sehnenlänge bei allen Individuen im mittleren Bereich deutlich kürzer. Allerdings wurde für die Statoren keine mechanische Nebenbedingung berücksichtigt, da bei einer so schmalen Bauweise die Struktur an ihre Belastungsgrenzen kommen können. Bei den Statoren der Individuen 1 und 2 ist die Sehnenlänge im unteren Drittel deutlich kürzer als bei Individum 3. So konnte durch die Reduktion der umströmten Fläche die Verluste reduziert werden. Insgesamt weisen die Multifidelity-Individuen bei der hier gezeigten Problemstellung einen höheren Optimierungsfortschritt auf, sowohl was Nebenbedingungen als auch Zielfunktionen angeht. Allerdings stellen die hier gezeigten Ergebnisse nur einen ersten Schritt in einer aeromechanischen Auslegung einer solchen Fanstufe dar. In einer realen Anwendung würden noch zahlreiche Auslegungsiterationen folgen. Da der Fokus hier aber auf dem Optimierungsverfahren liegt, wird dies nicht weiter verfolgt.

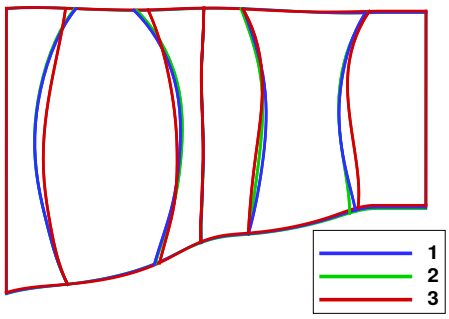


Abbildung 6.20: Naben- und Gehäusekontur der einzelnen Individuen im Vergleich

6.3 Übersicht der bisherigen Anwendungen in Industrie und Forschung

Innerhalb dieses Abschnitts wird ein kurzer Überblick gegeben, in welchen Forschungs- und Industrieprojekten das hier entwickelte Multifidelity-Verfahren bereits Anwendung gefunden hat. Dabei entspricht jede Überschrift der Unterabschnitte den Titeln der jeweiligen Projekte oder Forschungsarbeiten.

6.3.1 „Design of a counter rotating fan using a multidisciplinary and multifidelity optimisation under high level of restrictions“

Im Rahmen des EU-Projekts COBRA wurde eine gegenläufige aerodynamisch und akustisch optimierte Fanstufe ausgelegt. Daran beteiligten sich die Partner DLR, ONERA, COMOTI und Safran Aircraft Engines. Innerhalb des Projekts COBRA wurde eine Multifidelity-Optimierung mit dem hier entwickelten Verfahren erfolgreich durchgeführt und im Rahmen dieser Arbeit betreut. Die Ergebnisse dieser Optimierung sind in [Meillard et al., 2017] veröffentlicht und wurden experimentell überprüft.

Zielfunktionen und Nebenbedingungen: Ausgehend von einem Designvorschlag von Safran Aircraft Engines sollte ohne Einbußen im Wirkungsgrad eine Reduktion der Schallemissionen um -3dB erreicht werden. Daraus ergeben sich die zwei verwendeten Zielfunktionen:

- Maximierung des isentropen Wirkungsgrads in einem Betriebspunkt
- Minimierung des Schalldrucks in Austrittsrichtung

Die zu entwickelnde aeroakustisch optimierte Fanstufe wird von einem Planetengetriebe angetrieben, dessen Konzept der Industriepartner SAFRAN entwickelt hat und für technisch realisierbar hält. Daraus ergebt sich eine Reihe komplexer Nebenbedingungen. Erst während der Auslegungsarbeiten/Optimierungen stellte sich heraus, dass zur Erfüllung aller Nebenbedingungen nur ein begrenzter Lösungsraum existiert. Diese 15 Nebenbedingungen umfassen geometrische, aerodynamische, strukturmechanische und akustische Kriterien. Zu den wichtigsten Nebenbedingungen gehört die Einhaltung eines sehr eng gesetzten Drehmomentenverhältnisses der Rotoren, eine axiale Abtrömung am Austritt der Fanstufe, strukturmechanische Restriktion der maximalen mechanischen Spannungen und die Vermeidung mechanischer Resonanzen. Die initiale Geometrie erfüllt keine dieser Nebenbedingungen.

Parametrisierung: Die parameterbasierte Beschreibung der 3D-Schaufelformen bzw. der gesamten Fanstufe erfolgt ähnlich zum Testfall in Kapitel 6.2. Beide Rotoren werden jeweils durch 5 Profilschnitte beschrieben. Neben der Profilparametrisierung wird auch die Drehzahl beider Rotoren in engen Grenzen freigegeben. Damit ergibt sich eine Gesamtzahl von 112 freien Parametern.

Gütestufen: Es werden zwei Gütestufen verwendet, die sich durch die Anzahl der Zellen der CFD-Netze (2.000.000/300.000 Zellen) und durch die Anzahl der Elemente der FEM-Berechnung (19.000/2376 Elemente) unterscheiden. Die Prozesskette hoher Güte benötigt ca. 7.5h und die Prozesskette niedriger Güte ca. 1h.

Die Entwicklung der Entscheidungsfunktion war zu diesem Zeitpunkt noch nicht abgeschlossen. Daher wurde eine vereinfachte Version der in Kapitel 3.2.2.2 beschriebenen Entscheidungsfunktion verwendet. Diese beinhaltete keine Gewichtung der Zielfunktionen und Nebenbedingungen. Die Ergebnisse und Erfahrungen dieser Optimierung trugen wesentlich zur Weiterentwicklung der in dieser Arbeit beschriebenen „Gewichteten Entscheidungsfunktion“ und auch des Co-Krigings bei.

Ergebnisse: Diese Optimierung hatte eine Laufzeit von ca. einem Monat und wurde auf dem Cluster des Instituts für Antriebstechnik am DLR durchgeführt. Die Anzahl der verwendeten Rechenknoten lag im Mittel bei 10 und wurde je nach Auslastung des Clusters variiert. Die Ausstattung der Knoten entsprach der aus Kapitel 6.2.

Die Durchführung einer solch komplexen Optimierung benötigte eine intensive Betreuung und Begutachtung des Verlaufs der Zielfunktionen, der Nebenbedingungen, der Konvergenz des CFD-Lösers und der Ersatzmodelle.

Der Aufwand konnte aber durch die Ergebnisse gerechtfertigt werden: Die finale Geometrie erreichte einen vergleichbaren Wirkungsgrad wie das Ausgangsdesign, konn-

te die Schallemissionen aber um 3dB senken. Es wurden alle getriebebedingten Nebenbedingungen sowie alle festigkeitsmechanischen Forderungen für den aeroakustischen Test erfüllt.

6.3.2 „High-dimensional multi-fidelity optimisation of a 2.5 stage low pressure compressor“

Im Rahmen des aktuellen wehrtechnischen Projekts Mukoko wird ein 2.5 stufiger hochbelasteter Niederdruckverdichter ausgelegt. Dieses noch laufende Projekt findet in einer Kooperation zwischen dem DLR und der MTU statt. Im Rahmen von Mukoko wurde eine sehr aufwendige Multifidelity-Optimierung mit dem in dieser Arbeit entwickelten Verfahren durchgeführt. Die Ergebnisse dieser Optimierung sind bereits abgeschlossen und in [Hemmert-Pottmann and Voß, 2019] veröffentlicht. Die hier gezeigte Zusammenfassung der vorläufigen Ergebnisse erfolgt in freundlicher Zustimmung der Autoren.

Zielfunktionen und Nebenbedingungen: Es werden die insgesamt 4 verschiedenen aerodynamischen Betriebspunkte OP1-OP4 betrachtet. OP1 und OP3 beschreiben pumpgrenznahe Punkte und OP2 und OP4 befinden sich nahe des Sperrmassenstroms. OP1 und OP2 werden jeweils bei Auslegungsdrehzahl definiert und OP3 und OP4 bei Teillast. Die zwei Zielfunktionen sind die Maximierung des isentropen Wirkungsgrads in OP1 und OP3.

Weiterhin werden 30 Nebenbedingungen definiert, von denen ein Großteil auf mechanische Restriktionen fällt. Die mechanischen Nebenbedingungen umfassen die Vermeidung von Kreuzungen zwischen Eigenfrequenzen der Schaufeln und Anregefrequenzen. Zudem werden zahlreiche Nebenbedingungen auf die maximale Spannung an verschiedenen Teilen der Schaufeln definiert. Aufgrund von Vogelschlagsgefahr unterliegt bspw. die Schaufelvorderkante stärkeren Restriktionen.

Neben der mechanischen werden auch die aerodynamischen Restriktionen definiert. Diese umfassen eine axiale Abströmung, Pumpgrenzkriterien in zwei Betriebspunkten und mehrere Massenstromintervalle. Die initiale Geometrie erfüllt keine dieser Nebenbedingungen.

Parametrisierung: Die Parametrisierung umfasst die Strömungskanalgeometrie, die Anzahl der Schaufeln, die Parametrisierung der Rotoren mit jeweils 5 Schnitten und die Parametrisierung der Statoren mit jeweils 3 Schnitten. Insgesamt wird damit eine Anzahl von 264 freien Parametern erreicht und stellt damit eine enorme Herausforderung für das Optimierungsverfahren dar.

Gütestufen: Es werden zwei Gütestufen verwendet, die sich hauptsächlich durch die Zellen der CFD-Netze (7.600.000 gegen 400.000 Zellen) und durch die Anzahl der Elemente der FEM-Berechnung (43.200 gegen 10.000 Elemente) unterscheiden. Das Konvergenzkriterium wird für die niedrige Gütestufe geringer eingestellt. Dadurch

wird ein zeitlicher Faktor von 7-11 erreicht, wobei die Prozesskette hoher Güte 21-34h (je nach CFD Konvergenzgeschwindigkeit) und die Prozesskette niedriger Güte 3h benötigt. Die Prozesskette niedriger Güte verwendet 24 CPUs, während die mit hoher Güte 48 CPUs nutzt.

Die verwendete Entscheidungsfunktion ist die in dieser Arbeit entwickelte „gewichtete Entscheidungsfunktion“ aus Kapitel 3.2.2.2.

Ergebnisse: Insgesamt wurden mehrere Optimierungsphasen durchlaufen, wobei die Hauptphase die hier beschriebene Multifidelity-Optimierung darstellt. Diese Optimierung dauerte ca. 4.5 Monate und wurde auf dem Cluster des Instituts für Antriebstechnik am DLR durchgeführt. Die Anzahl der verwendeten Rechenknoten wurde je nach Auslastung des Clusters variiert, lag im Mittel aber bei 30 Knoten. Die Hardwareausstattung der Knoten entspricht der aus Kapitel 6.2.

Hauptziel der Optimierung war es, bei möglichst geringen Wirkungsgradeinbußen die Nebenbedingungen zu erfüllen. Die finale Geometrie konnte im Vergleich zur initialen Geometrie alle Nebenbedingungen erfüllen. Der Wirkungsgrad konnte bei Auslegungsdrehzahl um 0.7 Prozentpunkte und bei Teillast um 2.5 Prozentpunkte erhöht werden. Das Pumpgrenzkriterium zeigte allerdings einen Verlust von 2.5 und 2.9 Prozentpunkten in den entsprechenden Betriebspunkten. Beides liegt aber noch im angegebenen Gültigkeitsintervall.

Eine Sorge zu Anfang der Optimierung war die benötigte Trainingszeit, da 35 Ersatzmodelle mit jeweils 533 Hyperparametern trainiert werden mussten. Im Vergleich zu den Prozesskettenzeiten war die Zeit für das Training aber vernachlässigbar. Am meisten profitierte das Training an dem in Kapitel 5.3.3 beschriebenen Initialisierungsverfahren „Initialisierung auf Basis bereits vorhandener Kriging Modelle“. Dies lag daran, dass die Anzahl an Iterationen für das Training durch die Initialisierung von 500-600 Iterationen auf ca. 10 reduziert wurde.

Ferner kamen hier die zahlreichen softwaretechnischen Beschleunigungsverfahren aus Kapitel 5.5 und auch das speziell an das Co-Kriging angepasste Trainingsverfahren aus Kapitel 5.3.4 zum Einsatz. Insbesondere die Bestimmung der partiellen Ableitungen des Likelihood-Terms waren bei 533 Parametern ein erheblicher Zeitaufwand. Ohne diese Techniken hätte das Training der Ersatzmodelle zu einem Flaschenhals der Optimierung werden können.

6.3.3 „Entwicklung und Anwendung von modernen optimierungsfähigen Auslegungsverfahren unter Berücksichtigung des Realgaseinflusses“

Im Rahmen des AG-Turbo-Projekts COOREFLEX-turbo 1.2.4d (siehe [Goinis, 2018]) wurden die 8 hinteren Stufen einer 13-stufigen Verdichterkonfiguration mit dem in dieser Arbeit vorgestellten Multifidelity-Verfahren optimiert. Hauptziel dieses Vorhabens war es, eine für hohen Wirkungsgrad und hohe Stabilität optimierte Konfiguration eines Gasturbinenverdichters zu untersuchen und für dessen hintere Stufen Designerkenntnisse, -regeln und -richtlinien zu erarbeiten. Weiterhin sollte der Einfluss einer

Gehäusekonturierung auf den Wirkungsgrad und das Betriebsverhalten evaluiert werden.

In diesem Abschnitt sollen die Ergebnisse der Multifidelity-Optimierung kurz zusammengefasst werden.

Zielfunktionen und Nebenbedingungen: Ausgehend von einem Basisdesign der Firma Siemens sollte ohne Einschränkungen im Arbeitsbereich eine weitere Erhöhung des polytropen Wirkungsgrads erreicht werden. Daraus ergeben sich die zwei verwendeten Zielfunktionen:

- Maximierung des polytropen Wirkungsgrads im Arbeitspunkt
- Maximierung des Totaldruckverhältnisses in einem angedrosselten Betriebspunkt

Als Nebenbedingung wurde der Massenstrom im Arbeitspunkt in einem engen Intervall festgelegt. Zusätzlich wurde eine strukturmechanische Nebenbedingung festgelegt. Die maximale Vergleichsspannung der Rotoren dürfte nicht schlechter werden als die des Ausgangsdesigns.

Parametrisierung: In dieser Optimierung wurden insgesamt 350 Parameter freigegeben. Ein Großteil davon entfiel auf die Schaufelparametrisierung und die restlichen Parameter wurden für eine Gehäusekonturierung verwendet.

Gütestufen: Es wurden zwei Gütestufen verwendet, die sich durch die Anzahl der Zellen der CFD-Netze (14.700.000 gegen 1.800.000 Zellen) unterschieden. Die FEM-Rechnungen verwendeten für beide Gütestufen dieselben Rechennetze und hatten damit keinen Laufzeitunterschied.

Die Prozessketten hatten insgesamt einen zeitlichen Faktor von ca. 4, wobei die Prozesskette hoher Güte mit 48 CPUs berechnet wurde und die Prozesskette niedriger Güte mit nur 24 CPUs.

Da sich die in Kapitel 3.2.2 beschriebenen Entscheidungsfunktionen zu diesem Zeitpunkt noch in der Entwicklung befanden, wurde diese Optimierung mit einer initialen Datenbasis niedriger Güte gestartet und danach nur noch Individuen hoher Güte erzeugt.

Ergebnisse: Die gesamte Optimierung hatte eine Laufzeit von einem Monat und wurde auf dem Cluster des Instituts für Antriebstechnik am DLR durchgeführt. Die Anzahl der verwendeten Rechenknoten wurde je nach Auslastung des Clusters variiert, lag im Mittel aber bei 10 Knoten. Die Ausstattung der Knoten entspricht der aus Kapitel 6.2.

Die Optimierung wurde in drei Phasen durchgeführt. In der ersten Phase wurden aus der Referenzgeometrie Mutationen in beiden Gütestufen erzeugt, um eine initiale Datenbasis für die Ersatzmodelle zu generieren. Nach dieser initialen Phase wurden hauptsächlich Individuen niedriger Güte berechnet. In der dritten und letzten Phase wurde dann auf Individuen hoher Güte umgestellt. Die finale Paretofront im Raum hoher Güte konnte innerhalb sehr kurzer Zeit gefunden werden. Insgesamt wurde eine Wirkungsgradverbesserung von 0.45% Punkten erreicht.

6.3.4 „Multi-fidelity Method for Aerodynamic Shape Optimisation of Axial Compressor Blades“

Im Rahmen einer Masterarbeit [Verheij, 2017] wurde das in dieser Arbeit entwickelte Multifidelity-Verfahren bei Siemens getestet. Ziel dieses Tests war, eine Gesetzmäßigkeit zu finden, in welchen Fällen sich der Einsatz eines Multifidelity-Verfahrens lohnt. Zu diesem Zweck wurden verschiedene Variationen der Prozesskette niedriger Güte durchgeführt und die Ergebnisse mit einer Referenzoptimierung verglichen. Als Testfall wurde der zweite Rotor einer zweistufigen Verdichterkonfiguration von Siemens verwendet.

Zielfunktionen und Nebenbedingungen: Zwei Zielfunktionen und zwei Nebenbedingungen wurden definiert:

1. Isentroper Wirkungsgad bei 100% Drehzahl
2. Isentroper Wirkungsgrad bei Teildrehzahl und höherer Umgebungstemperatur

Die Nebenbedingungen waren wie folgt definiert:

1. Intervall auf das Totaldruckverhältnis eines pumpgrenznahen Betriebspunktes bei Teildrehzahl
2. Untere Grenze auf den Massenstrom beim Auslegungspunkt

Parametrisierung: Für den zweiten Rotor wurden 80 Parameter auf 5 Profilschnitten freigegeben; die genaue Parametrisierung ist in [Verheij, 2017] zu finden.

Gütestufen: Es wurden insgesamt 6 verschiedene Optimierungen miteinander verglichen:

- Referenz High-Fidelity-Optimierung:
  - Zwei Wiederholungen mit unterschiedlicher Anzahl an Rechenknoten
  - Als Referenz wurde die beste der beiden Optimierungen gewählt
  - 80 initiale Individuen, die zufällig erzeugt wurden
  - 903.000 Zellen für das gesamte Rechengitter
- Multifidelity-Optimierung:
  - Keine Wiederholung
  - 80 Individuen hoher Güte und 400 Individuen niedriger Güte für die Initialisierung
  - 5 verschiedene Low-Fidelity-Prozessketten
    - In 3 Optimierungen wurde die Anzahl an Zellen auf bis zu 223.000 reduziert
    - In einer Optimierung wurde der Spaltblock weggelassen, damit konnten die Zellen auf 132.000 reduziert werden
    - In einer weiteren Optimierung wurde ebenfalls der Spaltblock weggelassen, die Auflösung noch weiter reduziert und mit Euler-Wänden gerechnet. Damit wurden dann 12.800 Zellen erreicht.

Ergebnisse und Fazit: Zusammenfassend lässt sich festhalten, dass alle Optimierungen zu einer signifikanten Verbesserung der Zielfunktionen führten und die Nebenbedingungen alle eingehalten wurden. Ebenfalls waren die aerodynamischen Ergebnisse der Multi- und Single-Fidelity-Optimierungen miteinander vergleichbar. Beide Verfahren lieferten also in allen Testfällen vergleichbare Ergebnisse.

Die Beschleunigung der Multfidelity-Verfahren variierte bei diesen Testoptimierungen allerdings. In drei Fällen konnte eine Verlangsamung und in zwei Fällen eine Beschleunigung der Optimierung beobachtet werden. Als Begründung wird in [Verheij, 2017] eine mangelnde Korrelation und ein zu geringer Laufzeitunterschied zwischen den Gütestufen angegeben. Es existieren noch weitere Unsicherheiten, die in der Art der Ausführung und in den gewählten Optimierungseinstellungen begründet liegen:

- Die Erzeugung der Individuen erfolgte zu ca. 50% mithilfe der Ersatzmodelle und die anderen 50% wurden mit rein genetischen Algorithmen erzeugt. Dadurch geht der generelle Einfluss und auch die mögliche Beschleunigung durch die Ersatzmodelle verloren oder relativiert sich zumindest. Ein robusteres Optimierungsergebnis rechtfertigt allerdings diese Strategie in einer Singlefidelity-Optimierung. Innerhalb einer Multifdidelity-Optimierung kann dies allerdings durch die „günstige“ Abtastung des Raums mit dem Modell niedriger Güte erreicht werden. Dadurch wurde einer der wesentlichen Vorteile dieser Strategie nicht genutzt.
- Die initialen Individuen wurden für jeden Testfall mithilfe von zufällig gewählten Variablen neu erzeugt. Dadurch war die initiale Paretofront und auch der Volumenzugewinn der unterschiedlichen Optimierungen ebenfalls zufällig. Der Startpunkt der Optimierungen ist damit nicht mehr vergleichbar.
- Die Optimierungen wurden jeweils nur einmal durchgeführt. Betrachtet man die in dieser Arbeit durchgeführte Fanstufenoptimierung (siehe Kapitel 6.2), so sind die Schwankungen der Optimierungswiederholungen erheblich, obwohl alle Optimierungen mit derselben Startdatenbasis durchgeführt wurden. Daher sind mehrere Wiederholungen unerlässlich, um eine fundierte Aussage treffen zu können, insbesondere bei einer zufälligen Initialisierung.
- Das Ersatzmodelltraining wurde nur nach jedem fünften konvergierten Individuum durchgeführt. Diese Einstellung führt insbesondere am Anfang einer Optimierung zu einer verlangsamten Anpassung der Hyperparameter und damit zu einer Verlangsamung der Optimierung selbst.
- Die Anzahl an gleichzeitig rechnenden Knoten wurde für jeden Optimierungsfall variiert, was ebenfalls das Ergebnis beeinflusst und einen Vergleich erschwert.
- Die ersten beiden Testoptimierungen besaßen einen zeitlichen Faktor von ungefähr $\frac{t_l}{t_h} = 0.5$. Mit diesem Faktor und der gewählten Anzahl an intitialen Startindividuen ist es für die Multifidelity-Optimierung sehr schwierig, eine Beschleunigung zu erreichen. Die Optimierung hoher Güte startete mit 80 Individuen, die Multifidelity-Optimierung hingegen mit 80 Startindividuen und 400 Individuen niedriger Güte. Zeitlich gesehen entsprechen die Individuen niedriger Güte ungefähr 200 Individuen hoher Güte. Die Single-Fidelity-Optimierung startete also mit einem Zeitverzug von 80 Individuen und die Multifidelity-Optimierung mit einem Zeitverzug von 280 Individuen. Dies entspricht einer 3.5-fach längeren Initialisierungsphase. Zudem ist die Länge der Initialisierungsphase mit 280 Individuen hoher Güte sehr lang. Die Verlangsamung der Optimierung ist für diese beiden Testfälle also erklärbar.

Fazit: Erneute Tests sind geplant, in denen die genetische Komponente vollständig ausgeschaltet ist und ein günstigeres Verhältnis an initialen Individuen gewählt werden soll.

6.3.5 „Optimizing Surge Margin and Efficiency of a Transonic Compressor“

Im Rahmen der ASME-Veröffentlichung [Goinis and Nicke, 2016] wurde untersucht, inwiefern die Anwendung von Casing-Treatments (CTs) an einem transsonischen Verdichter zu einer Verbesserung des Wirkungsgrades und der Pumpgrenze führen können. Bei dem Testverdichter „Rig250“ handelt es sich um einen 4.5-stufigen Axialverdichter, welcher bereits experimentell und numerisch untersucht wurde. Die numerische Studie wurde mithilfe des in dieser Arbeit entwickelten Multifidelity-Optimierungsverfahrens durchgeführt und soll in diesem Abschnitt kurz zusammengefasst werden. Der Fokus liegt auf der Anwendung des Multifidelty-Verfahrens.

Zielfunktionen und Nebenbedingungen: Es wurden die insgesamt 5 verschiedenen, aerodynamischen Betriebspunkte OP1-OP5 betrachtet. OP1 (Auslegungspunkt) und OP2 lagen auf der 100%-Drehzahllinie. OP3, OP4 und OP5 lagen bei jeweils verschiedenen Massenströmen auf der 90%-Drehzahllinie. Der Betriebspunkt OP5 wurde für jede Simulation an die numerische Pumpgrenze iteriert und der genaue Ablauf dieses Verfahrens wird in [Goinis and Nicke, 2016] erläutert.

Es gab insgesamt 2 Zielfunktionen (siehe [Goinis and Nicke, 2016] für die genaue Definition):

1. Isentroper Wikrungsgrad des Verdichters in OP2
2. Pumpgrenzkriterium

Als Nebenbedingungen wurde der Massenstrom bei OP1 und OP2 in engen Grenzen festgelegt und zusätzlich das Totaldruckverhältnis von OP2. Weiterhin wurden die maximalen Vergleichsspannungen des Rotors berechnet und ein Maximum festgelegt.

Parametrisierung: Die Parametrisierung umfasste zum einen die Rotor-Geometrie, für welche Profilparameter auf 6 unterschiedlichen radialen Höhen freigegeben wurden. Weiterhin wurde über dem Rotor eine Gehäusekonturierung ermöglicht, welche mit 6 Spline-Punkten aufgelöst wurde. Neben der Gehäusekonturierung und den Profilparametern wurde eine Umfangsnut definiert und mit 6 Parametern beschrieben. Insgesamt wurde damit eine Anzahl von 70 Parametern erreicht.

Gütestufen: Die in dieser Optimierung verwendeten Gütestufen unterscheiden sich nicht durch die Güte der CFD-Netze, sondern durch die Anzahl der Betriebspunkte und einer einfacheren Pumpgrenzabschätzung.

Während für die hohe Gütestufe 5 Betriebspunkte berechnet wurden und einer davon durch eine numerische Pumpgrenzbestimmung besonders aufwendig war, wurden für

die niedrige Gütestufe nur 3 Betriebspunkte festgelegt. Weiterhin wurde in der niedrigen Gütestufen auf die Pumpgrenzbestimmung verzichtet und stattdessen ein fester pumpgrenznaher Massenstrom eingestellt.

Ergebnisse: Es wurden insgesamt drei verschiedene Optimierungen mit derselben Prozesskette durchgeführt:

1. Optimierung der Umfangsnuten
2. Optimierung der Gehäusekontur
3. Optimierung des Rotors

Besonders interessant ist das Verhalten der Co-Kriging-Ersatzmodelle für die Pumpgrenzabschätzung. In Anhang Abbildung **??** sind einige Ergebnisse der Co-Kriging-Analysesoftware dargestellt. Bspw. wird in Anhang Abbildung

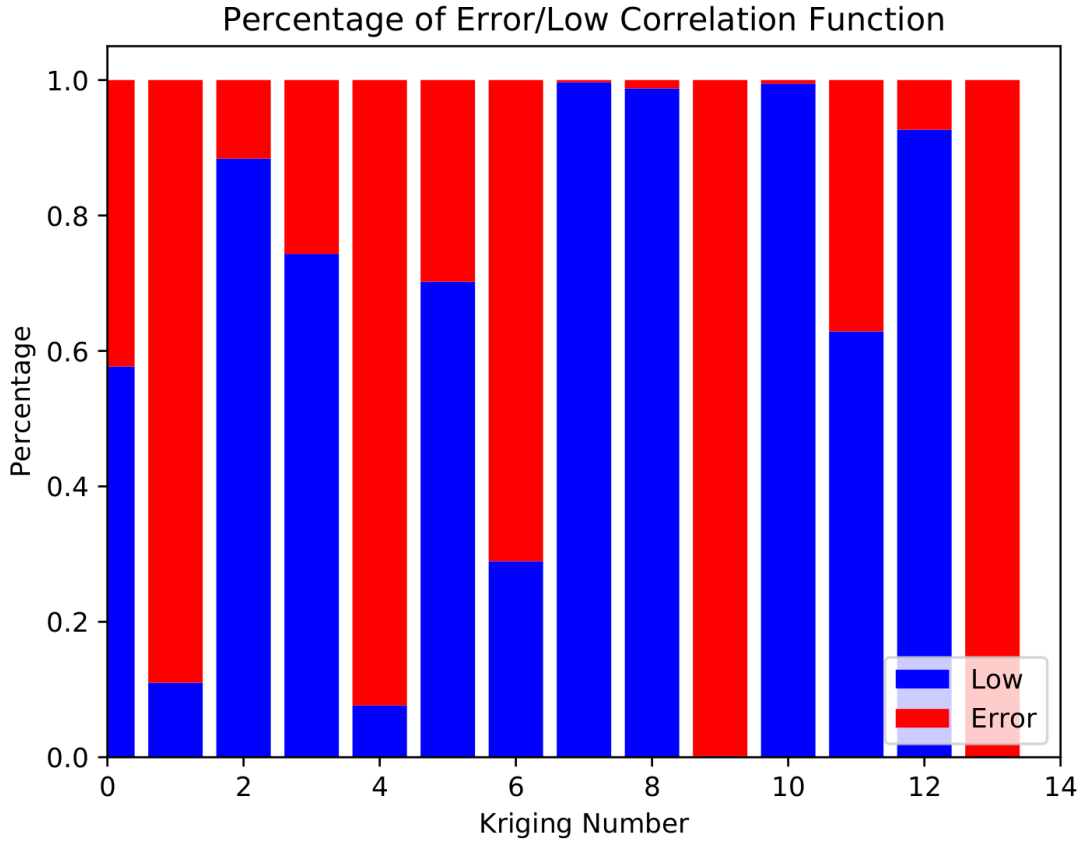


Abbildung 6.21: Anteile der Kovarianzfunktionen verschiedenen Co-Kriging Modellen der Verdichter-Optimierung aus Kapitel 6.3.5. Auszug der Kriging-Analysesoftware

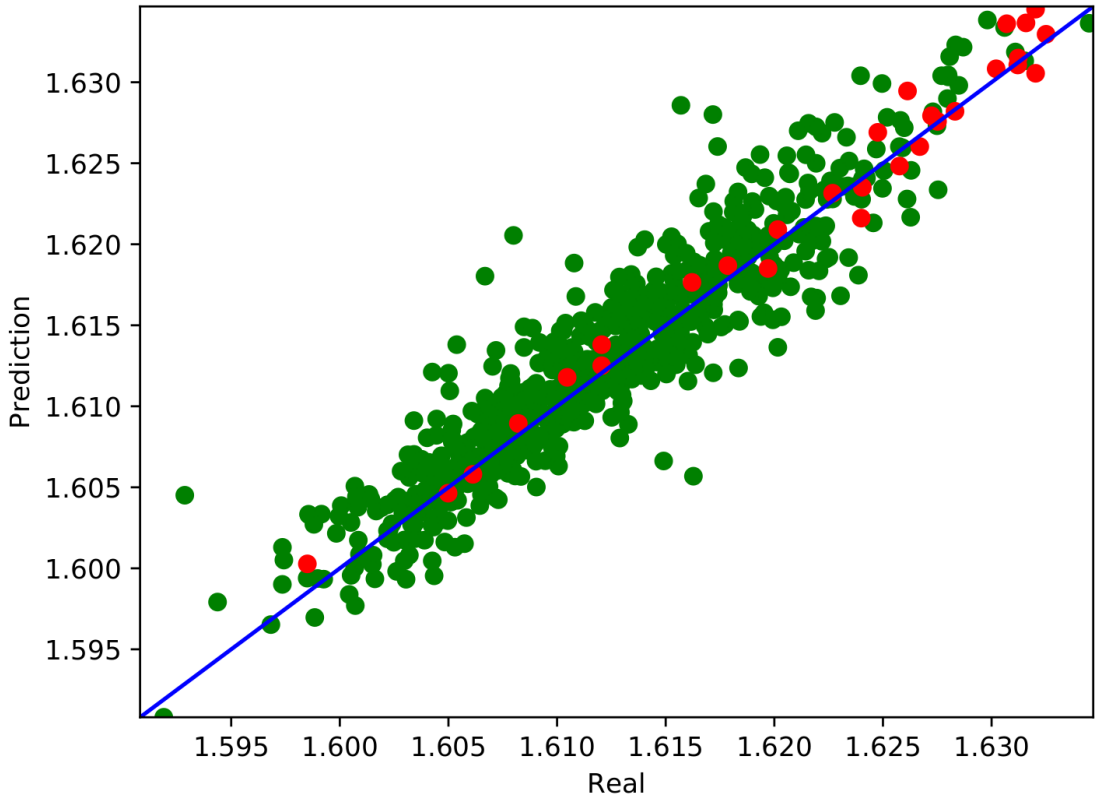


Abbildung 6.22: Vergleich von Vorhersage und Echtwerten des Totaldruckverhältnisses der numerischen Pumpgrenze der Verdichteroptimierung aus Kapitel 6.3.5. Auszug der Kriging-Analysesoftware.

6.21 - Ersatzmodell Nummer 9 das Massenstrom-Ersatzmodell für den erwähnten pumpgrenznahen Betriebspunkt gezeigt. Es ist erkennbar, dass die niedrige Güte vom Ersatzmodell ausgeschaltet wurde. Die Co-Kriging-Ersatzmodelle finden also keinen Zusammenhang zwischen der numerischen Pumpgrenze und einem festen pumpgrenznahen Punkt.

Weiterhin ist es von Interesse, inwiefern das Totaldruckverhältnis der niedrigen Güte zur Verbesserung des Ersatzmodells beiträgt. In Abbildung 6.21 ist das Totaldruckverhältnis durch Ersatzmodell Nummer 10 beschrieben. Dieses weist eine sehr hohe Gewichtung der niedrigen Gütestufe auf, was darauf schließen lässt, dass die Daten niedriger Güte in großem Maße gewichtet wurden. Anhang Abbildung 6.22 zeigt alle Vorhersagen und die dazugehörigen Realwerte des Totaldrucks an. Die blaue Linie beschreibt die Ideallinie, in rot sind die letzten 25 und in grün alle übrigen Vorhersagen dargestellt. Damit lässt sich feststellen, ob die letzten Vorhersagen besser geworden sind.

Die Vorhersagen liegen mit einer kleinen Streuung um die Echtwerte gleichmäßig verteilt. Dies lässt darauf schließen, dass das Ersatzmodell keine Verzerrungen, wie z.B. eine Erwartungswertverschiebung, aufweist. Ferner ist die Fehlerrate sehr gering und zwar bereits am Anfang der Optimierung, was wiederum auf die gute Qualität der Daten niedriger Güte schließen lässt.

Nach Abschluss der Optimierungen wurde versucht, die Einzelergebnisse der Optimierungen miteinander zu kombinieren, bspw. wurden ausgewählte Gehäusekonturierun-

gen und Umfangsnuten an optimierten Rotorgeometrien getestet.

Die Ergebnisse der einzelnen Optimierungen lassen sich in drei Punkten zusammenfassen:

1. Die Optimierung der Umfangsnuten resultierte in einer leichten Erhöhung der Pumpgrenze bei gleichzeitigem Verlust des Wirkungsgrads.
2. Die Optimierung der Gehäusekontur resultierte in einer leichten Erhöhung der Pumpgrenze bei gleichzeitiger leichter Erhöhung des Wirkungsgrads. Die Erhöhung des Wirkungsgrads wurde auf eine Umverteilung des Massenstroms zurückgeführt, welche zu einer besseren Rotoranströmung führt und damit auf ein suboptimales Rotordesign zurückgeführt werden kann.
3. Die Optimierung der Gehäusekontur resultierte in einer Erhöhung der Pumpgrenze bei gleichzeitiger Erhöhung des Wirkungsgrads.

Diese Optimierung dauerte 4 Monate und wurde auf dem Cluster des Instituts für Antriebstechnik am DLR durchgeführt. Die Anzahl der verwendeten Rechenknoten wurde je nach Auslastung des Clusters variiert und lag im Mittel bei 10 Knoten. Die Hardwareausstattung der Knoten entspricht der aus Kapitel 6.2.

7 Zusammenfassung

Ziel dieser Arbeit war es, ein industriell einsetzbares und erweiterbares Multifidelity-Optimierungsverfahren unter Berücksichtigung der Anforderungen im Bereich der Turbomaschinenauslegung zu entwickeln und zu testen. Die Herausforderungen bei der Optimierung von Turbomaschinen sind die Existenz mehrerer Zielfunktionen und zahlreicher Nebenbedingungen sowie hochdimensionale Suchräume und sehr laufzeitintensive Prozessketten, welche nicht immer Ergebnisse liefern. Das hier vorgestellte Multifidelity-Optimierungsverfahren sorgt dafür, dass hochdimensionale Suchräume effizienter abgetastet werden können und damit die Laufzeit der Optimierungen deutlich reduziert wird bzw. deutlich robustere und zuverlässigere Aussagen zu deren Ergebnissen erlauben. Hierfür werden schnelle Prozessketten niedrigerer Güte herangezogen und mit Hilfe der dort enthaltenen Informationen die notwendige Anzahl an teuren Prozesskettenauswertungen minimiert. Die Anforderungen an das Multifidelity-Optimierungsverfahren umfassen die folgenden Punkte (vgl. Kapitel 1.3):

1. Entwicklung eines CO-Kriging-Ersatzmodells
2. Erweiterung des bisherigen Optimierungsverfahrens
3. Testen des entwickelten Verfahrens
4. Nachweis der praktischen Anwendbarkeit an einer industriellen Turbomaschinenoptimierung

Entwicklung eines CO-Kriging-Ersatzmodells: Ein Kernpunkt dieser Arbeit war die Entwicklung eines Co-Kriging-Ersatzmodells. Erst auf dessen Basis ist eine Multifidelity-Strategie effizient umsetzbar. Die theoretischen Hintergründe werden in Kapitel 4 hergeleitet. Eine Besonderheit des hier entwickelten Verfahrens ist die Möglichkeit die Punkte der verschiedenen Gütestufen frei im Suchraum verteilen zu können (siehe Kapitel 4.4) - wiederum eine wichtige Voraussetzung für die anvisierten Anwendungen des verbesserten Optimierungsverfahrens.

Die praktische Anwendung in Forschung und Industrie erfordert eine klare und vor allem erweiterbare Softwarestruktur. Dafür wurde eine moderne objektorientierte interfacebasierte Programmierung in der Programmiersprache C++ gewählt. Diese Art der Softwarearchitektur ermöglicht die effiziente Implementierung unterschiedlicher Verfahren, wie z.B. Ordinary-, Gradient-Enhanced-, Co-Kriging und Klassifikatoren, wie z.B. Supporting-Vector-Machines innerhalb eines Programms. Viele Softwaremodule können geteilt und so Redundanzen vermieden werden, die Übersichtlichkeit wird erhöht und die Fehleranfälligkeit reduziert. Beispielsweise wird dadurch ein Netzwerkaustausch von vollständigen Objekten ermöglicht, es werden mehrere Minimierungsverfahren allgemein nutzbar gemacht und eine Matrix-Klasse bereitgestellt, wel-

che unterschiedlichste Architekturen (GPUs, Intel(c) MKL, OpenMP) unterstützt.

Innerhalb einer Optimierung müssen die Ersatzmodelle einen höheren Zeitgewinn bringen als die Nutzung selbst erfordert. Die hier entwickelten Ersatzmodelle sind in der Lage, viele moderne Hardwarebeschleunigungen zu nutzen. Besonders erwähnt seien hier Advanced-Vector-Extensions (siehe Kapitel 5.5.3), GPU-Beschleunigung (siehe 5.5.9) und OpenMP-Parallelisierung auf Thread-Ebene (siehe Kapitel 5.5.1).

Weiterhin wurden Maßnahmen innerhalb des Verfahrens zur Beschleunigung des Trainings umgesetzt. Hierzu zählen:

- Unterschiedliche Initialisierungsmethoden des Trainingsverfahrens (siehe Kapitel 5.3.3)
- Ein in dieser Arbeit entwickeltes RPROP-Trainingsverfahren, angepasst an das Co-Kriging (siehe Kapitel 5.3.4)
- Die in dieser Arbeit entwickelte Matrix-Klasse mit zahlreichen Beschleunigungsverfahren und der Möglichkeit, verschiedene Architekturen zu nutzen (siehe Kapitel 5.5.1)
- Eine effiziente Umsetzung der Kovarianzfunktionen (siehe Kapitel 5.5.3)
- Die automatisierte Reduktion von Stützstellen (siehe Kapitel 5.5.4) auf Basis eines Kriging-Modells
- Verschiedene Methoden zur effizienten Berechnung des Likelihood-Terms und dessen Ableitungen (siehe Kapitel 5.5.6 und 5.5.8)

Darüberhinaus wurde eine unabhängige Softwarebibliothek entwickelt, welche die asynchrone Verteilung verschiedener Operationen auf mehrere Rechner erlaubt (siehe Kapitel 5.5.2). Mit dieser Bibliothek konnten die wesentlichen Matrix-Operationen parallelisiert werden. Das System ist vollständig asynchron und ermöglicht es auch, die Aufgaben auf unterschiedliche Architekturen zu verteilen, bspw. Server mit GPUs und CPUs oder auch verschieden starke Server im Mischbetrieb zu verwenden. Die dafür nötige Lastverteilung übernimmt ein Scheduler. Zudem kann das System Ausfälle einzelner Server kompensieren und die Aufgaben zeitnah an andere Server verteilen. Dies ist ein erheblicher Vorteil gegenüber den meisten MPI-basierten Systemen.

Erweiterung des bisherigen Optimierungsverfahrens: Parallel zur Entwicklung des Ersatzmodells erfolgte die Erarbeitung entsprechender Optimierungstrategien und deren Integration in eine bestehende Optimierungsumgebung. Hier handelt es sich um die DLR-Optimierungssuite AutoOpti. Deren Aufbau sowie die hier beschriebenen Entwicklungen und Anwendungen sichern die vollständige Verallgemeinerungsfähigkeit der im Rahmen dieser Arbeit erzielten Ergebnisse. Eine neue Schnittstelle in AutoOpti gestattet die Anbindung und Nutzung unterschiedlicher Ersatzmodelle (siehe Kapitel 5.5.1). Selbstverständlich musste auch die Datenbankhaltung und Prozesskettensteuerung in AutoOpti auf mehrere Gütestufen erweitert werden.

Eine automatisierte Entscheidungsfunktion (siehe Kapitel 3.2.2) trifft während der Laufzeit einer Optimierung die Entscheidung, welche Gütestufe als nächstes berechnet werden soll. Die Entscheidung wird dabei so getroffen, dass diese den Optimierungsverlauf möglichst günstig beeinflusst. Hierfür werden Prozesskettenlaufzeiten, Trainings- und Optimierungszeiten, der lokale Informationsgehalt der Gütestufen und die Gewichtung der unterschiedlichen Nebenbedingungen mit einbezogen. Die durch-

geführten Validierungen und die ersten Anwendungen zeigen eine Leistungsfähigkeit und Robustheit, wie sie bisher in der Literatur nicht gefunden wurde.

Testen des entwickelten Verfahrens: Das Hauptziel dieser Arbeit war es, ein in Industrie- und Forschungsumgebung nutzbares Multifidelity-Verfahren zu entwickeln. Begonnen wurde mit einfachen akademischen Testfällen, bei denen neben der prinzipiellen Arbeitsfähigkeit der Umgang mit „Grenzfällen“ getestet wurde. Erste praktische Anwendungen im Turbomaschinenbereich dienten der Überprüfung der laufenden Forschungs- und Entwicklungsarbeiten und der Weiterentwicklung des Verfahrens. Die in Kapitel 6.2 beschriebene aeromechanische Optimierungsreihe einer modernen Fanstufe zeigt den Geschwindigkeitsvorteil des hier entwickelten Multifidelity-Verfahrens gegenüber einer konventionellen Singlefidelity-Optimierung. Hierfür wurden die Optimierungen mehrfach wiederholt, um Zufälligkeiten abzubilden und die Robustheit des Verfahrens messen zu können. Neben der Beschleunigung konnte beobachtet werden, dass die Multifidelity-Optimierungen alle zu demselben guten Ergebnis führten, während die Singlefidelity-Optimierungen stärkere Schwankungen zeigten.

Nachweis der praktischen Anwendbarkeit an industriellen Turbomaschinenoptimierungen: Die in Kapitel 6.3 beschriebenen Anwendungen zeigen die angestrebte praktische Anwendungsbereitschaft der hier erarbeiteten Ergebnisse. Sie zeigen aber auch, dass eine erfolgreiche Anwendung kein „push-Button“-Prozess ist. Das Aufsetzen, die Beobachtung und Steuerung einer solchen Optimierung erfordert eine hohe Aufmerksamkeit des Anwenders. Bei der in Kapitel 6.3.1 beschriebenen Anwendung konnte aber auch beobachtet werden, dass mit einer reinen Single-Fidelity-Strategie kein Ergebnis erzielt werden konnte, welches alle Nebenbedingungen und Auslegungsziele erfüllt.

Aktuelle Entwicklungen: Auch nach Fertigstellung der vorliegenden Arbeit werden die hier entwickelten Methoden weiter verbessert und deren Anwendungsbereiche erweitert. Bereits zum jetzigen Zeitpunkt sind einige dieser Entwicklungen besonders aussichtsreich. Insbesondere die Nutzung des Co-Kriging-Modells außerhalb der reinen Multifidelity-Optimierungen ist ein neues Feld und ermöglicht vielversprechende Ansätze, wie z.B. die Wiederverwendung älterer Optimierungen. Eine typische Problemstellung in der Entwicklung von Turbomaschinen sind bspw. geänderte aerodynamische Randbedingungen oder eine geänderte Parametrisierung während eines Auslegungszyklus. Dies kann unter anderem dazu führen, dass komplette Optimierungen aufgrund kleinerer Veränderungen neu gestartet werden müssen, ohne dass die alten Daten verwendet werden können. Mithilfe des Co-Krigings können die Daten von älteren Optimierungen als niedrige Güte in einer neuen Optimierung weiterverwendet werden. Je nach Ähnlichkeit der Optimierungen kann der Zugewinn dabei sehr groß sein. Die Wahrscheinlichkeit einer Beschleunigung ist hoch, da die Daten bereits vorhanden sind.

Eine weitere potentielle Nutzung des Co-Kriging-Modells ist die Kopplung von komplexen, nicht statistischen Ersatzmodellen zur Verbesserung der Vorhersagegenauigkeit. Hier sind insbesondere Deep-Learning-Verfahren zu nennen. Dabei handelt

es sich um neuronale Netzwerke mit einem komplexen Aufbau und einer hohen Anzahl an Zwischenschichten. Mithilfe dieser neuronalen Netzwerke sollen komplette 3D-Strömungslösungen gelernt, verallgemeinert und damit Vorhersagen getroffen werden. Ein großer Vorteil liegt dabei in der Unabhängigkeit der Vorhersagen von den freien Optimierungsparametern. Dies ist möglich, da die neuronalen Netzwerke die gesamte 3D-Geometrie als Input verwenden. Erste Arbeiten zu diesem Thema sind in [Aulich et al., 2019] zu finden. Diese tiefen neuronalen Netzwerke bieten bisher noch keine Unsicherheitsvorhersage an, sodass ein „Expected-Volume-Gain"-Algorithmus (siehe Kapitel 2.3.2) nicht anwendbar ist. Eine mögliche Lösung hierfür bietet die Nutzung eines Co-Krigings. Die Vorhersagen des neuronalen Netzwerks könnten als niedrige Gütestufe verwendet werden und die Kriging-Vorhersagen damit verbessern. Diese Entwicklung befindet sich allerdings noch in einem sehr frühen Entwicklungsstadium.

Zukünftig wird die in Kapitel 5.5.2 beschriebene Netzwerkbibliothek weiterentwickelt. Hier liegt das Ziel darin, die Bibliothek so zu erweitern, dass freie Rechnerkapazitäten für größere Anwendungen automatisch genutzt werden können; zuerst innerhalb eines internen Netzwerkes und langfristig auf das Internet ausgedehnt, in Anlehnung an das SETI@home Programm (siehe [Korpela et al., 2001]).

Eine anwendungsbereite Multifidelity-Optimierungsstrategie wurde erarbeitet und deren praktische Anwendbarkeit gezeigt. Die dafür notwendigen Entwicklungen, insbesondere ein Co-Kriging-Ersatzmodell, eine automatisierte Entscheidungsfunktion sowie die für die robuste und zuverlässige Anwendung notwendigen softwaretechnischen Entwicklungen wurden realisiert. Mit diesen Ergebnissen ist eine effiziente Optimierung bei typischen Turbomaschinenauslegungen möglich.

Literaturverzeichnis

[top, 2018] (2018). Top500 supercomputer site - https://www.top500.org/statistics/list/ - abgerufen 09/2018.

[Adams et al., 2016] Adams, B. M., Mohamed S., E., Eldred, M. S., Geraci, G., Jakeman, J. D., Maupin, K. A., Monschke, J. A., Swiler, L. P., Stephens, J. A., Vigil, D. M., and Wildey, T. M. (2016). Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.5 Theory Manual.

[Akin and Siemes, 2013] Akin, H. and Siemes, H. (2013). *Praktische Geostatistik: eine Einführung für den Bergbau und die Geowissenschaften*. Springer-Verlag.

[Anderson and Wendt, 1995] Anderson, J. D. and Wendt, J. (1995). *Computational fluid dynamics*, volume 206. Springer.

[Aulich et al., 2019] Aulich, M., Kueppers, F., Schmitz, A., and Voß, C. (2019). Surrogate estimations of complete flow fields of fan stage designs via deep neural networks. *Proceedings of ASME Turbo Expo 2019: Turbomachinery Technical Conference and Exposition*.

[Aulich and Siller, 2011] Aulich, M. and Siller, U. (2011). High-Dimensional Constrained Multiobjective Optimization of a Fan Stage. In *ASME GT2011-45618*.

[Avron and Toledo, 2011] Avron, H. and Toledo, S. (2011). Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2):1–34.

[Backhaus et al., 2012] Backhaus, J., Aulich, M., Frey, C., Lengyel, T., and Voß, C. (2012). Gradient Enhanced Surrogate Models Based on Adjoint CFD Methods for the Design of a Counter Rotating Turbofan. *ASME Turbo Expo*.

[Backhaus et al., 2017] Backhaus, J., Schmitz, A., Frey, C., Mann, S., Nagel, M., Sagebaum, M., and Gauger, N. R. (2017). Application of an Algorithmically Differentiated Turbomachinery Flow Solver to the Optimization of a Fan Stage. pages 1–20.

[Baert et al., 2015] Baert, L., Beauthier, C., Leborgne, M., and Lepot, I. (2015). Surrogate-Based Optimisation for a Mixed-Variable Design Space: Proof of Concept and Opportunities for Turbomachinery Applications. In *Volume 2C: Turbomachinery*, page V02CT45A015. ASME.

[Bandemer, 1980] Bandemer, H. (1980). *Theorie und Anwendung der optimalen Versuchsplanung*. Number Bd. 2 in Mathematische Lehrbücher und Monographien: Mathematische Monographien. Akademie-Verlag.

[Bellman, 1972] Bellman, R. (1972). *Dynamic programming.* Princeton University Press.

[Bisplinghoff et al., 1957] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L. (1957). *Aeroelasticity*. Addisin-Wesley.

[Bräunling, 2004] Bräunling, W. J. (2004). *Flugzeugtriebwerke: Grundlagen, Aero-Thermodynamik, Kreisprozesse, Thermische Turbomaschinen, Komponenten- und Emissionen*. VDI-Buch Series. Springer.

[Bronstejn and Semendjajew, 2008] Bronstejn, I. N. and Semendjajew, K. A. (2008). *Taschenbuch der Mathematik*. Harri Deutsch.

[Brooks et al., 2011] Brooks, C. J., Forrester, A. I. J., Keane, A. J., and Shahpar, S. (2011). MULTI-FIDELITY DESIGN OPTIMISATION OF A TRANSONIC COMPRESSOR ROTOR. *9th European Conf. Turbomachinery Fluid Dynamics and Thermodynamics, Istanbul, Turkey*.

[Chahine et al., 2012] Chahine, C., Seume, J. R., and Verstraete, T. (2012). The Influence of Metamodeling Techniques on the Multidisciplinary Design Optimization of a Radial Compressor Impeller. In *Volume 8: Turbomachinery, Parts A, B, and C*, page 1951. ASME.

[Cook, 2012] Cook, S. (2012). *CUDA Programming*. Morgan Kaufmann.

[Cox and John, 1992] Cox, D. D. and John, S. (1992). A statistical method for global optimization. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE.

[Cressie, 1990] Cressie, N. (1990). The origins of kriging. *Mathematical geology*, 22(3):239–252.

[Cressie, 1993] Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons.

[Dagum and Menon, 1998] Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55.

[Davis and Morris, 1997] Davis, G. and Morris, M. (1997). Six factors which affect the condition number of matrices associated with kriging. *Mathematical Geology*, 29(5):669–683.

[Deb, 2014] Deb, K. (2014). Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer.

[Domercq, 2006] Domercq, O. (2006). Advances in Axial Compressor Aerodynamics. *Lecture Series 2006-06*, page 38.

[Drela and Youngren, 2008] Drela, M. and Youngren, H. (2008). A User's Guide to MISES 2.63.

[Dworak et al., 2011] Dworak, A., Ehm, F., Sliwinski, W., and Sobczak, M. (2011). MIDDLEWARE TRENDS AND MARKET LEADERS 2011. *CERN*.

[Ehrgott, 2005] Ehrgott, M. (2005). *Multicriteria optimization*, volume 491. Springer Science & Business Media.

[Elfert et al., 2016] Elfert, M., Weber, A., Wittrock, D., Peters, A., Voss, C., and Nicke, E. (2016). Experimental and Numerical Verification of An Optimization of a Fast Rotating High Performance Radial Compressor Impeller. *ASME Turbo Expo*, pages 1–12.

[Emmerich, 2005] Emmerich, M. (2005). *Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels*. PhD thesis.

[Emmerich and Klinkenberg, 2008] Emmerich, M. and Klinkenberg, J.-w. (2008). The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Rapport technique, Leiden University*, 34.

[Fog, 2013] Fog, A. (2013). Software optimization resources.

[Forrester et al., 2006] Forrester, A. I., Bressloff, N. W., and Keane, A. J. (2006). Optimization using surrogate models and partially converged computational fluid dynamics simulations. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 462, pages 2177–2204. The Royal Society.

[Forrester and Keane, 2009] Forrester, A. I. J. and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79.

[Forrester et al., 2007] Forrester, A. I. J., Sóbester, A., and Keane, A. J. (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 463(2088):3251–3269.

[Forrester et al., 2008] Forrester, A. I. J., Sobester, A., and Keane, A. J. (2008). *Engineering design via surrogate modelling : a practical guide*. J. Wiley.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:.

[Gen and Cheng, 2000] Gen, M. and Cheng, R. (2000). *Genetic algorithms and engineering optimization*. Wiley.

[Gere and Weaver, 1965] Gere, J. and Weaver, W. (1965). *Analysis of framed structures*. University series in civil engineering and applied mechanics. Van Nostrand.

[Giannakoglou and Karakasis, 2006] Giannakoglou, K. C. and Karakasis, M. K. (2006). Hierarchical and distributed metamodel-assisted evolutionary algorithms. *VKI Lecture Series*.

[Gibbs and MacKay, 1997] Gibbs, M. and MacKay, D. J. C. (1997). Efficient implementation of Gaussian processes.

[Gill, 2007] Gill, P. (2007). Numerical linear algebra and optimization.

[Gill et al., 1981] Gill, P. E., Murray, W., and Wright, M. H. (1981). Practical optimization.

[Goinis, 2018] Goinis, G. (2018). Entwicklung und Anwendung von modernen optimierungsfähigen Auslegungsverfahren unter Berücksichtigung des Realgaseinflusses. *Forschungsvorhaben 03ET7040R Verbundvorhaben COOREFLEX-turbo Teilvorhaben 1.2.4d*.

[Goinis and Nicke, 2016] Goinis, G. and Nicke, E. (2016). Optimizing surge margin and efficiency of a transonic compressor. In *ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition*, pages V02AT37A048–V02AT37A048. American Society of Mechanical Engineers.

[Government, 2009] Government, U. (2009). Code of federal regulations - 15 cfr ch. vii (1–1–09 edition) §744.

[Griewank and Walther, 2008] Griewank, A. and Walther, A. (2008). *Evaluating derivatives : principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics.

[Han et al., 2012] Han, Z., Zimmerman, R., and Görtz, S. (2012). Alternative Cokriging Method for Variable-Fidelity Surrogate Modeling. *AIAA journal*, 50(5):1205–1210.

[Han et al., 2009] Han, Z.-H., Görtz, S., and Zimmermann, R. (2009). On improving efficiency and accuracy of variable-fidelity surrogate modeling in aero-data for loads context. In *Proceeings of European Air and Space Conference*.

[Helbig and Scherer, 2011] Helbig, H. and Scherer, A. (2011). Neuronale Netze. *Vorlesungsskript*.

[Hemmert-Pottmann and Voß, 2019] Hemmert-Pottmann, S. and Voß, C. (2019). High-dimensional multi-fidelity optimisation of a 2.5 stage low pressure compressor. In *ISABE-2019 Conference*.

[Hintjens, 2013] Hintjens, P. (2013). *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.".

[Hodges and Pierce, 2011] Hodges, D. H. and Pierce, G. A. (2011). *Introduction to structural dynamics and aeroelasticity*. Cambridge University Press.

[Hoerl and Kennard, 2000] Hoerl, A. E. and Kennard, R. W. (2000). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86.

[Holland, 1975] Holland, J. H. (1975). Adaptation in natural and artificial systems. an introductory analysis with applications to biology, control and artificial intelligence. *Ann Arbor: University of Michigan Press, 1975*.

[Howard and Gallimore, 1992] Howard, M. A. and Gallimore, S. J. (1992). Viscous Throughflow Modelling for Multi-Stage Compressor Design. In *Volume 1: Turbomachinery*, page V001T01A109. ASME.

[Huang et al., 2006] Huang, D., Allen, T. T., Notz, W. I., and Miller, R. A. (2006). Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382.

[Hutchinson, 1989] Hutchinson, M. F. (1989). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076.

[J. Forrester et al., 2006] J. Forrester, A. I., Keane, A. J., and Bressloff, N. W. (2006). Design and Analysis of "Noisy" Computer Experiments. *AIAA Journal*, 44(10):2331–2339.

[Jin et al., 2001] Jin, R., Chen, W., and Simpson, T. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13.

[Jones, 2001] Jones, D. R. (2001). A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383.

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492.

[Karlsson, 2006] Karlsson, B. (2006). *Beyond the C++ Standard Library: An Introduction to Boost*. Addison-Wesley.

[Keane, 2006] Keane, A. J. (2006). Statistical improvement criteria for use in multiobjective design optimization. *AIAA journal*, 44(4):879–891.

[Kennedy and O'Hagan, 2000] Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87:1–13.

[Kiefer, 1959] Kiefer, J. (1959). Optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 272–319.

[Kiefer, 1974] Kiefer, J. (1974). General equivalence theory for optimum designs (approximate theory). *The annals of Statistics*, pages 849–879.

[Köller, 1999] Köller, U. (1999). *Entwicklung einer fortschrittlichen Profilsystematik für stationäre Gasturbinenverdichter: 5 Tabellen*. Deutsches Zentrum für Luft- und Raumfahrt e.V. DLR, Bibliotheks- und Informationswesen.

[Korpela et al., 2001] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Lebofsky, M. (2001). Seti@ home—massively distributed computing for seti. *Computing in science & engineering*, 3(1):78–83.

[Krige, 1953] Krige, D. G. (1953). A statistical approach to some basic mine valuation problems on the witwatersrand.

[Krüger, 2012] Krüger, F. (2012). *Entwicklung von parallelisierbaren Gradienten-basierten Verfahren zur automatisierten, Ersatzmodell-gest{ü}tzten Optimierung unter Nebenbedinungen f{ü}r CFD-FEM-Verdichterdesign*. PhD thesis.

[Kügeler, 2005] Kügeler, E. (2005). Numerisches Verfahren zur genauen Analyse der Kühleffektivität filmgekühlter Turbinenschaufeln.

[Küppers, 2016a] Küppers, F. (2016a). Auslagerung von matrizenoperationen auf die gpu. Praxisbericht, DHBW Mannheim.

[Küppers, 2016b] Küppers, F. (2016b). Entwicklung und aufbau eines verteilten systems zur reduktion der rechenzeit für gradienten basierte ersatzmodell algorithmen.

[Küppers and Schmitz, 2016] Küppers, F. and Schmitz, A. (2016). Ersatzmodelltraining für turbomaschinenoptimierungen: Entwicklung eines verteilten systems zur auslagerung rechenintensiver algorithmen auf gpus. In *Deutscher Luft- und Raumfahrtkongress 2016*.

[Kushner, 1964] Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106.

[Käters et al., 2000] Käters, B., Schreiber, H.-A., Köler, U., and Mönig, R. (2000). 1999 turbomachinery committee best paper award: Development of advanced compressor airfoils for heavy-duty gas turbines—part ii: Experimental and theoretical analysis. *Journal of Turbomachinery*, 122(3):406–414.

[Lapworth and Shahpar, 2004] Lapworth, L. and Shahpar, S. (2004). DESIGN OF GAS TURBINE ENGINES USING CFD. *European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004*.

[Lawson et al., 1979] Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T. (1979). Basic linear algebra subprograms for fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, 5(3):308–323.

[Le Gratiet, 2013] Le Gratiet, L. (2013). Bayesian Analysis of Hierarchical Multifidelity Codes *. 1:244–269.

[Lengyel-Kampmann, 2015] Lengyel-Kampmann, T. (2015). *Vergleichende aerodynamische Untersuchungen von gegenläufigen und konventionellen Fanstufen für Flugtriebwerke*. PhD thesis.

[Lepot et al., 2011] Lepot, I., Leborgne, M., Schnell, R., Yin, J., Delattre, G., Falissard, F., and Talbotec, J. (2011). Aero-mechanical optimization of a contra-rotating open rotor and assessment of its aerodynamic and acoustic characteristics. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 225(7):850–863.

[Library, 2011] Library, I. M. K. (2011). Processors - Define SSE2,SSE3 and SSE4.

[Lieblein and Seymour, 1955] Lieblein and Seymour (1955). Aerodynamic Design of Axial-flow Compressors. VI - Experimental Flow in Two-Dimensional Cascades.

[Lighthill, 1952] Lighthill, M. J. (1952). On sound generated aerodynamically i. general theory. *Proc. R. Soc. Lond. A*, 211(1107):564–587.

[Lighthill, 1954] Lighthill, M. J. (1954). On sound generated aerodynamically ii. turbulence as a source of sound. *Proc. R. Soc. Lond. A*, 222(1148):1–32.

[Lophaven et al., 2002] Lophaven, S. N., Nielsen, H. B., and Søndergaard, J. (2002). *Aspects of the Matlab Toolbox DACE*. Citeseer.

[MacKay, 1998] MacKay, D. J. (1998). Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166.

[Mackkay, 1991] Mackkay, D. J. C. (1991). *Bayesian Methods for Adaptive Models*. PhD thesis, Citeseer.

[Mader et al., 2008] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and Van Der Weide, E. (2008). ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers.

[Mark Gibbs, 1997] Mark Gibbs, D. J. M. (1997). Efficient Implementation of Gaussian Processes.

[Marler and Arora, 2004] Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

[Matheron, 1963] Matheron, G. (1963). Principles of geostatistics. *Economic geology*, 58(8):1246–1266.

[McKay et al., 1979] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239.

[Meillard et al., 2017] Meillard, L., Stanica, C. M., Nasr, N. B., and Riéra, W. (2017). Design of a counter rotating fan using a multidisciplinary and multifidelity optimisation under high level of restrictions. In *ISABE-2017- 21387*.

[Miettinen, 2012] Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

[Mockus, 1994] Mockus, J. (1994). Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365.

[Mönig et al., 2000] Mönig, R., Mildner, F., and Röper, R. (2000). Viscous-Flow 2D-Analysis Including Secondary Flow Effects. In *Volume 1: Aircraft Engine; Marine; Turbomachinery; Microturbines and Small Turbomachinery*, page V001T03A104. ASME.

[Müller-Töws, 2000] Müller-Töws, J. (2000). *Aerothermodynamische Auslegung der Meridianströmung mehrstufiger Axialverdichter mit Hilfe von Optimierungsstrategien*. PhD thesis.

[Murray, 2016] Murray, I. (2016). Differentiation of the Cholesky decomposition.

[Nürnberger, 2004] Nürnberger, D. (2004). *Implizite Zeitintegration für die Simulation von Turbomaschinenströmungen*. PhD thesis, Ruhr-Universität Bochum.

[Nvidia, 2007] Nvidia, C. (2007). Compute unified device architecture programming guide.

[Özkaya, 2014] Özkaya, E. (2014). *One-shot methods for aerodynamic shape optimization*. PhD thesis.

[Pareto, 1964] Pareto, V. (1964). *Cours d'économie politique*, volume 1. Librairie Droz.

[Perttunen, 1991] Perttunen, C. D. (1991). A computational geometric approach to feasible region division in constrained global optimization. In *Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on*, pages 585–590. IEEE.

[Peter and Dwight, 2010] Peter, J. E. and Dwight, R. P. (2010). Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391.

[Pierret, 1999] Pierret, S. (1999). *Designing turbomachinery blades by means of the function approximation concept based on artificial neural network, genetic algorithm, and the Navier-Stokes equations*. PhD thesis, Faculté Polytechnique de Mons - Von Karman Institute for Fluid Dynamics.

[Pierret and Van den Braembussche, 1998] Pierret, S. and Van den Braembussche, R. A. (1998). Turbomachinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Network. In *Volume 1: Turbomachinery*, page V001T01A002. ASME.

[Plackett, 1950] Plackett, R. L. (1950). Some theorems in least squares. *Biometrika*, 37(1/2):149–157.

[p.l.c., 2016] p.l.c., B. (2016). BP Statistical Review of World Energy June 2016. Technical report.

[Press et al., 2007] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

[Przemieniecki, 1985] Przemieniecki, J. S. (1985). *Theory of matrix structural analysis*. Courier Corporation.

[Pukelsheim, 1980] Pukelsheim, F. (1980). On linear regression designs which maximize information. *Journal of Statistical Planning and Inference*, 4(4):339–364.

[Queipo et al., 2005] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K. (2005). Surrogate-based Analysis and Optimization.

[Raman et al., 2000] Raman, S. K., Pentkovski, V., and Keshava, J. (2000). Implementing streaming simd extensions on the pentium iii processor. *IEEE micro*, 20(4):47–57.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog.

[Reid and Moore, 1978] Reid, L. and Moore, R. D. (1978). Design and overall performance of four highly loaded, high speed inlet stages for an advanced high-pressure-ratio core compressor.

[Reimer, 2016] Reimer, E. (2016). *Vergleichende Optimierung eines Fans mit High- und Multifidelity Verfahren*. Bachelor thesis, Fachhochschule Aachen.

[Riedmiller and Braun, 1993] Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

[Sacks et al., 1989] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, pages 409–423.

[Sacks et al., 2007] Sacks, J., Welch, W. J., Mitchell, T. J., Wynn, H. P., Science, S., and Nov, N. (2007). Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–423.

[Sacks and Ylvisaker, 1970] Sacks, J. and Ylvisaker, D. (1970). Statistical design and integral approximation. *Proc. 12th Bienn. Semin. Can. Math. Congr.(R. Pyke, ed.)*, pages 115–136.

[Särkkä, 2013] Särkkä, S. (2013). *BAYESIAN FILTERING AND SMOOTHING*. Cambridge University Press.

[Schlichting and Gersten, 2006] Schlichting, H. and Gersten, K. (2006). *Grenzschicht-theorie*. Springer-Verlag.

[Schmid and Feilke, 2012] Schmid, V. and Feilke, M. (2012). Räumliche statistik, vorlesungsskript.

[Schmidt and O'Hagan, 2003] Schmidt, A. M. and O'Hagan, A. (2003). Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758.

[Schmitz, 2013] Schmitz, A. (2013). Entwicklung eines objektorientierten und parallelisierten gradient enhanced kriging ersatzmodells. Technical report, Fernuniversität Hagen.

[Schmitz et al., 2011] Schmitz, A., Aulich, M., and Nicke, E. (2011). Novel approach for loss and flow-turning prediction using optimized surrogate models in two-dimensional compressor design. In *ASME Turbo Expo 2011: Power for Land, Sea and Air*.

[Schnoes and Nicke, 2015] Schnoes, M. and Nicke, E. (2015). Automated calibration of compressor loss and deviation correlations. In *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*.

[Schnös and Nicke, 2017] Schnös, M. and Nicke, E. (2017). Exploring a Database of Optimal Airfoils for Axial Compressor Design. *AIAA/CEAS Aeroacoustics Conference*.

[Schönweitz et al., 2013] Schönweitz, D., Voges, M., Goinis, G., Enders, G., and Johann, E. (2013). Experimental and Numerical Examinations of a Transonic Compressor-Stage With Casing Treatment. In *Volume 6A: Turbomachinery*, page V06AT35A035. ASME.

[Schumacher, 2014] Schumacher, T. (2014). Implementierung von support vector machines in der umgebungeines evolutionären optimierers. Bachelor thesis, Duale Hochschule Baden Würtemberg.

[Sechler, 1952] Sechler, E. E. (1952). *Elasticity in engineering*. Dover Publications.

[Shahpar, 2000] Shahpar, S. (2000). A Comparative Study of Optimisation Methods for Aerodynamic Design of Turbomachinery Blades. In *Volume 1: Aircraft Engine; Marine; Turbomachinery; Microturbines and Small Turbomachinery*, page V001T03A087. ASME.

[Shepard, 1968] Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM.

[Shigeo, 2011] Shigeo, M. (2011). Fast approximate float function fmath.

[Siller et al., 2009] Siller, U., Voß, C., and Nicke, E. (2009). Automated Multidisciplinary Optimization of a Transonic Axial Compressor. *AIAA Aerospace Sciences Meeting*.

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

[Simpson et al., 2008] Simpson, T., Toropov, V., Balabanov, V., and Viana, F. (2008). Design and Analysis of Computer Experiments in Multidisciplinary Design Optimization: A Review of How Far We Have Come - Or Not. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Reston, Virigina. American Institute of Aeronautics and Astronautics.

[Smith, 1995] Smith, S. P. (1995). Differentiation of the Cholesky Algorithm. *Journal of Computational and Graphical Statistics*.

[Sozio et al., 2013] Sozio, E., Verstraete, T., and Paniagua, G. (2013). Design-Optimization Approach to Multistage Axial Contra-Rotating Turbines. In *Volume 6B: Turbomachinery*, page V06BT37A016. ASME.

[Spiegel, 2000] Spiegel, M. (2000). Einsatz deterministischer optimierungsverfahren bei der vorauslegung hochbelasteter turbomaschinen. *Herbert Utz Verlag*.

[Steimann et al., 2016] Steimann, F., Keller, D., and Aziz Safi, B. (2016). Moderne programmiertechniken und-methoden. *Skript zur Vorlesung der FernUniversität Hagen*.

[Tang et al., 2016] Tang, X., Luo, J., and Liu, F. (2016). Adjoint-Response Surface Method in Aerodynamic Shape Optimization of Turbomachinery Blades. In *Volume 2C: Turbomachinery*, page V02CT39A004. ASME.

[Thornburg, 2006] Thornburg, H. (2006). Autoregressive Modeling: Elementary Least-Squares Methods. *Center for Computer Research in Music and Acoustics (CCRMA) Department of Music, Stanford University Stanford, California 94305*.

[Tikhonov et al., 2013] Tikhonov, A. N., Goncharsky, A., Stepanov, V., and Yagola, A. G. (2013). *Numerical methods for the solution of ill-posed problems*, volume 328. Springer Science & Business Media.

[Toal et al., 2009] Toal, D., Forrester, A., Bressloff, N., Keane, A., and Holden, C. (2009). An adjoint for likelihood maximization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2111):3267–3287.

[Toal et al., 2011] Toal, D. J., Bressloff, N., Keane, A., and Holden, C. (2011). The development of a hybridized particle swarm for kriginghyperparameter tuning.

[Toal, 2016] Toal, D. J. J. (2016). A study into the potential of GPUs for the efficient construction and evaluation of Kriging models. *Engineering with Computers*, 32:377–404.

[Trunk, 1979] Trunk, G. V. (1979). A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):306–307.

[Verheij, 2017] Verheij, L. (2017). Multi-fidelity method for aerodynamic shape optimisation of axial compressor blades. Master's thesis, Faculty of AE) Dr. ir. R. Pecnik, TU Delft.

[Verstraete, 2008] Verstraete, T. (2008). *MULTIDISCIPLINARY TURBOMACHINERY COMPONENT OPTIMIZATION CONSIDERING PERFORMANCE, STRESS, AND INTERNAL HEAT TRANSFER*. PhD thesis, University of Ghent - Von Karman Institute.

[Verstraete et al., 2014] Verstraete, T., Prinsier, J., and Cosi, L. (2014). Design and Off-Design Optimization of a Low Pressure Steam Turbine Radial Diffuser Using an Evolutionary Algorithm and 3D CFD. In *Volume 1B: Marine; Microturbines, Turbochargers and Small Turbomachines; Steam Turbines*, page V01BT27A046. ASME.

[Viertl, 2013] Viertl, R. K. (2013). *Einführung in die Stochastik: mit Elementen der Bayes-Statistik und Ansätzen für die Analyse unscharfer Daten*. Springer-Verlag.

[Voß et al., 2014] Voß, C., Aulich, M., and Raitor, T. (2014). Metamodel Assisted Aeromechanical Optimization of a Transonic Centrifugal Compressor.

[Voß and Nicke, 2008] Voß, C. and Nicke, E. (2008). Automatische Optimierung von Verdichterstufen. *AG Turbo COOREFF 1.1.1*, (September):1–71.

[Weicker, 2015] Weicker, K. (2015). *Evolutionäre Algorithmen*. Springer Vieweg.

[Whaley and Dongarra, 1998] Whaley, R. C. and Dongarra, J. J. (1998). Automatically tuned linear algebra software. In *SC'98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, pages 38–38. IEEE.

[Willburger, 2011] Willburger, A. (2011). *Beitrag zur Berechnung der Meridianströmung in Axialverdichtern auf der Basis der umfangsgemittelten Navier-Stokes-Gleichungen unter Berücksichtigung dreidimensionaler Einflüsse*. Kassel University Press.

[Willeke and Verstraete, 2015] Willeke, S. and Verstraete, T. (2015). Adjoint Optimization of an Internal Cooling Channel U-Bend. In *Volume 5A: Heat Transfer*, page V05AT11A029. ASME.

[Wu, 1952] Wu, C.-H. (1952). A general theory of three-dimensional flow in subsonic and supersonic turbomachines of axial-, radial-, and mixed-flow types.

[Z.-H. Han, R. Zimmermann, 2010] Z.-H. Han, R. Zimmermann, S. G. (2010). A New Cokriging Method for Variable-Fidelity Surrogate Modeling of Aerodynamic Data. *48th AIAA Aerospace Sciences Meeting*, (January):1–22.

[Zienkiewicz and Taylor, 2005] Zienkiewicz, O. C. and Taylor, R. L. (2005). *The finite element method for solid and structural mechanics*. Elsevier.

[Zitzler et al., 2000] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195.

A Anhang

A.1 Kapitel 3

A.1.1 Aerodynamische Größen

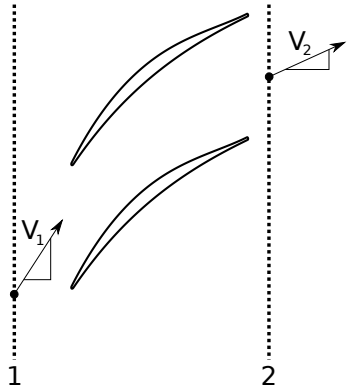


Abbildung A.1: Auswerteebenen eines exemplarischen Verdichtergitters

Definition des Totaldruckverlustbeiwerts in massenstromgemittelter Form:

$$\omega = \frac{\frac{1}{\dot{m}_2}\int p_{t2}^{is} - p_{t2}\mathrm{d}\dot{m}}{\frac{1}{\dot{m}_1}\int p_{t1} - p_1\mathrm{d}\dot{m}}$$

Wobei die Indizes sich auf die in Abbildung A.1 gezeigten Auswerteebenen vor und hinter dem Verdichtergitter beziehen. p_{t2} beschreibt den Totaldruck an der Austrittsebene und p_{t2}^{is} den Totaldruck bei isentroper Zustandsänderung. Der Massenstrom wird beschrieben durch die Variable $\dot{m}$.

A.1.2 Beispiel Entscheidungsfunktion globales/lokales Varianzkriterium

Die eindimensionale Testfunktion ist wie folgt definiert und wird in Abbildung A.2 dargestellt:

$$f_{high} = 2x + 0.1 \sin(20x)$$
$$f_{low} = 0.1 \sin(20x) + 0.2$$

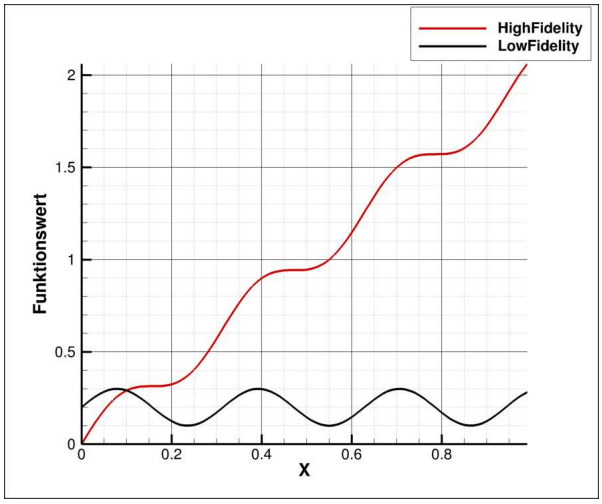


Abbildung A.2: Darstellung der Testfunktion hoher und niedriger Güte

Abbildung A.3 zeigt die Co-Kriging Vorhersage der Funktion bei 11 Stützstellen hoher und 7 niedriger Güte.

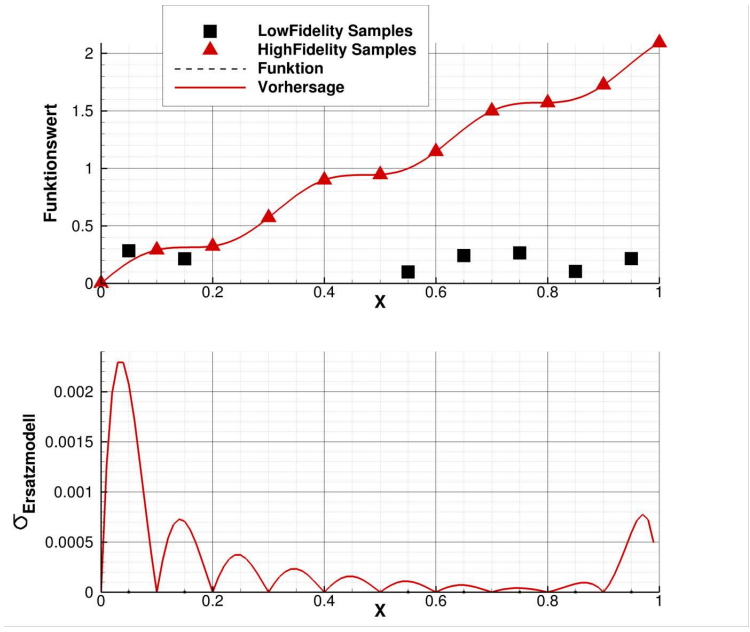


Abbildung A.3: Co-Kriging Vorhersage des Erwartungswerts und der Standardabweichung bei 11 Stützstellen hoher und 7 niedriger Güte

Abbildung A.4 zeigt die vorhergesagte Funktion und die Standardabweichung nach dem Hinzufügen einer Stützstelle hoher Güte bei $x = 0.35$. Die vorhergesagte Stan-

dardabweichung wird lokal auf Null reduziert. Allerdings wird auch die umliegende $x \neq 0.35$ Standardabweichung reduziert.

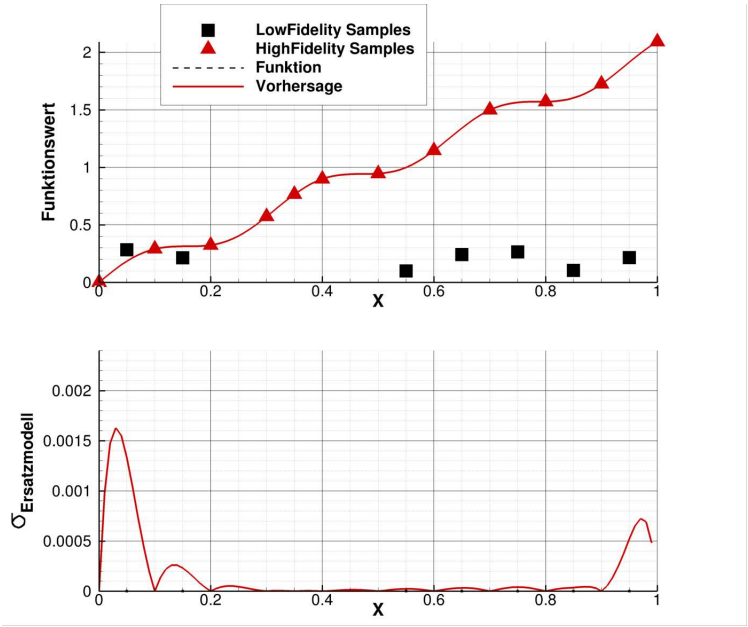


Abbildung A.4: Co-Kriging Vorhersage nach dem Hinzufügen einer Stützstelle hoher Güte bei $x = 0.35$

Abbildung A.5 zeigt die vorhergesagte Funktion und die Standardabweichung nach dem Hinzufügen einer Stützstelle niedriger Güte bei $x = 0.35$. Die vorhergesagte Standardabweichung wird lokal und global ($x \neq 0.35$) reduziert.

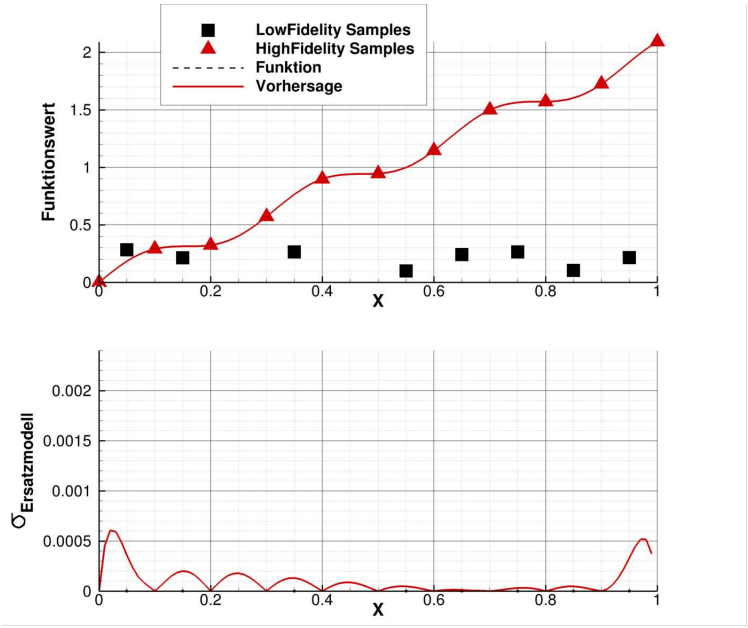


Abbildung A.5: Co-Kriging Vorhersage nach dem hinzufügen einer Stützstelle nierdriger Güte bei $x = 0.35$

In Abbildung A.6 wird die Reduktion der Standardabweichung für alle Fälle verglichen.

Weiterhin ist auch der gemessene absolute Fehler dargestellt.

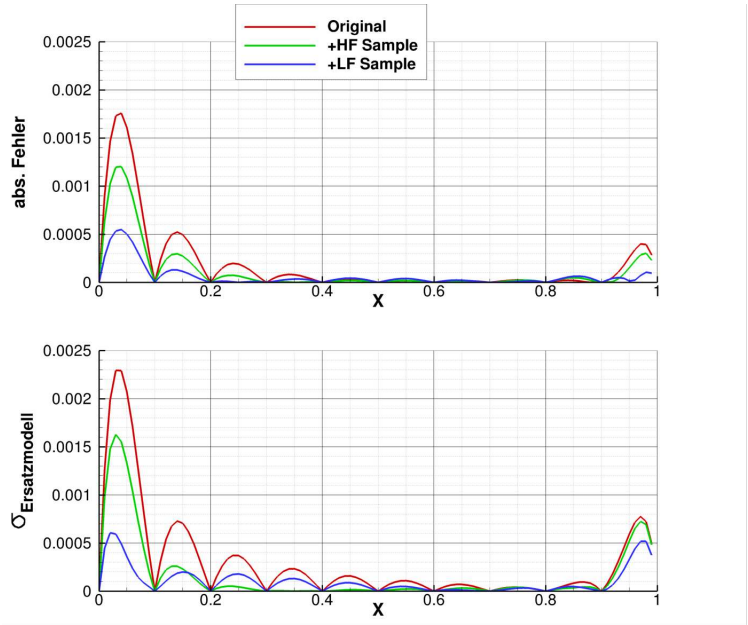


Abbildung A.6: Vorhergesagte Standardabweichung und absoluter Fehler nach dem hinzufügen einer Stützstelle hoher oder niedriger Güte bei $x = 0.35$

Die Standardabweichungen der Testfälle werden wie folgt bezeichnet $\sigma_h(x)$,$\sigma_{h|h}(x)$, $\sigma_{h|l}(x)$. Wobei der Index h für die Referenzfunktion ohne Stützstelle am Ort $x = 0.35$ bezeichnet. Die Indizes $h|h$ und $h|l$ beschreiben die Standardabweichungen unter Kenntnis einer Stützstelle hoher oder niedriger Güte am Ort $x = 0.35$.

Wird das Integral der Standardabweichungen für die verschiedenen Testfälle über dem Raum (von 0-1 in diesem Fall) gebildet:

$$C_h = \int_0^1 \sigma_h(x)dx$$

$$C_{h|h} = \int_0^1 \sigma_{h|h}(x)dx$$

$$C_{h|l} = \int_0^1 \sigma_{h|l}(x)dx$$

Und dieses mit der lokalen Reduktion der Standardabweichungen verglichen:

$$L_{h|h} = \sigma_h(0.35) - \sigma_{h|h}(0.35)$$
$$L_{h|l} = \sigma_h(0.35) - \sigma_{h|l}(0.35)$$

Als Vergleich dient das gemessene Fehlerintegral zwischen Vorhersage f^* und echter Funktion f:

$$F_h = \int_0^1 |f_h^*(x) - f_h(x)|\, dx$$
$$F_{h|h} = \int_0^1 \left|f_{h|h}^*(x) - f_h(x)\right| dx$$
$$F_{h|l} = \int_0^1 \left|f_{h|l}^*(x) - f_h(x)\right| dx$$

In Tabelle A.1 werden die Ergebnisse aller gemessenen Werte dargestellt. Basierend auf diesen Werten könnte eine Entscheidungsfunktion wie sie in Kapitel 3.2.2 vorgestellt wird verwendet werden. Zusätzlich müssten allerdings noch die Zeiten für die Ausführung der jeweiligen Gütestufen berechnet werden, diese sollen für dieses Beispiel allerdings unberücksichtigt bleiben. Eine Entscheidungsfunktion würde anhand der hier gezeigten Werte dann denjenigen aussuchen, welche die meiste Reduktion verspricht. Im Falle des lokalen Kriterium, berechnet sich die Reduktion $\Delta\sigma$ für die jeweilige Gütestufe wie folgt:

$$\Delta\sigma_{h|h} = \sigma_h(0.35) - L_{h|h} = 0.000234$$
$$\Delta\sigma_{h|l} = \sigma_h(0.35) - L_{h|l} = 0.000103$$

Für das globale Kriterium berechnet sich die Reduktion analog:

$$\Delta\sigma_{h|h} = C_h - C_{h|h} = 0.000137$$
$$\Delta\sigma_{h|l} = C_h - C_{h|l} = 0.000195$$

Global		Lokal		Fehlerintegral	
C_h	0.000296	$\sigma_h(0.35)$	0.000234	F_h	0.000184
$C_{h\|h}$	0.000159	$L_{h\|h}$	0.000000	$F_{h\|h}$	0.000115
$C_{h\|l}$	0.000101	$L_{h\|l}$	0.000131	$F_{h\|l}$	0.000060

Tabelle A.1: Berechnete Ergebnisse der unterschiedlichen Fehlerbewertungskriterien.

Die Entscheidungsfunktion würde sich hier immer für die größere Reduktion entscheiden. Im Falle des lokalen Kriteriums würde die hohe Güte gewählt und im Falle des globalen Kriteriums die niedrige Güte. Eine Entscheidung auf Basis des Fehlerintegrals ginge ebenfalls zugunsten der niedrigen Gütestufe. Das untermauert nochmals die Ergebnisse der globalen Entscheidung. Die Ersatzmodelle wurden nach Hinzufügen

einer neuen Stützstelle allerdings nicht neu trainiert, dies könnte zu einer Änderung der hier gezeigten Situation führen.

A.2 Kapitel 4

A.2.1 Gauss Korrelationsfunktion mit Ableitungen

Die hier gezeigten Ableitungen sind für die Entwicklung der unterschiedlichen Kriging-Verfahren nötig und können in anderen Arbeiten dieser Richtung hilfreich sein. Aus diesem Grund werden die Ergebnisse der Ableitungen hier aufgelistet.

$$f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) = e^{-\frac{1}{2}\sum^{k}\left(e^{\theta_k}|x_{1k}-x_{2k}|^2\right)} + \delta e^{\lambda}$$

Ableitungen der Hyperparameter

$$\frac{\partial f}{\partial \theta_d} = f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(-\frac{1}{2}e^{\theta_d}\left|x_{1d}-x_{2d}\right|^2\right)$$

$$\frac{\partial f}{\partial \lambda} = e^{\lambda}\delta \Rightarrow \frac{\partial R}{\partial \lambda} = e^{\lambda}E$$

Erste Ableitungen nach dem Ort

$$\frac{\partial f}{\partial x_1^k} = \left(f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) - \delta e^{\lambda}\right)\left(-e^{\theta_k}\left(x_{1k}-x_{2k}\right)\right)$$

$$\frac{\partial f}{\partial x_2^k} = \left(f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) - \delta e^{\lambda}\right)\left(e^{\theta_k}\left(x_{1k}-x_{2k}\right)\right)$$

Zweite Ableitungen nach dem Ort

$$\frac{\partial f}{\partial x_1^k \partial x_2^l} = \begin{cases} -e^{\theta_l}\left(x_{1l}-x_{2l}\right)\frac{\partial f}{\partial x_2^k} & k \neq l \\ \left(e^{\theta_k} - \left[\left(x_{1k}-x_{2k}\right)e^{\theta_k}\right]^2\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) & k = l \end{cases}$$

$$\frac{\partial f\left(\overrightarrow{x}_1, \overrightarrow{x}_1\right)}{\partial x_1^k \partial x_1^l} = \begin{cases} 0 & k \neq l \\ e^{\theta_k} & k = l \end{cases}$$

Ableitungen nach Ort und Hyperparameter

$$\frac{\partial f}{\partial x_1^k \theta_k} = -e^{\theta_k}\left(x_{1k}-x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(1 - \frac{1}{2}e^{\theta_k}\left(x_{1k}-x_{2k}\right)^2\right)$$

$$\frac{\partial f}{\partial x_1^k \theta_p} = \frac{1}{2}\left(x_{1k} - x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(e^{\theta_p+\theta_k}\left(x_{1p} - x_{2p}\right)^2\right)$$

$$\frac{\partial f}{\partial x_2^k \theta_k} = f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(x_{1k} - x_{2k}\right) e^{\theta_k}\left(1 - \frac{1}{2}\left(x_{1k} - x_{2k}\right)^2 e^{\theta_k}\right)$$

$$\frac{\partial f}{\partial x_2^k \theta_p} = -\frac{1}{2} e^{\theta_k+\theta_p}\left(x_{1k} - x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(x_{1p} - x_{2p}\right)^2$$

Zweite Ableitungen nach Ort und Hyperparameter

$$\frac{\partial f}{\partial x_1^p \partial x_2^p \theta_p} = e^{\theta_p} f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left[\left(1 - \left(x_{1p} - x_{2p}\right)^2 2e^{\theta_p}\right) - \frac{1}{2}e^{\theta_p}\left(1 - \left(x_{1p} - x_{2p}\right)^2 e^{\theta_p}\right)\left(x_{1d} - x_{2d}\right)^2\right]$$

$$\frac{\partial f}{\partial x_1^k \partial x_2^k \theta_p} = -\frac{1}{2} f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) e^{\theta_k} e^{\theta_p}\left(1 - \left(x_{1k} - x_{2k}\right)^2 e^{\theta_k}\right)\left(x_{1p} - x_{2p}\right)^2$$

$$\frac{\partial f}{\partial x_1^k \partial x_2^l \theta_k} = -e^{\theta_l+\theta_k}\left(x_{1l} - x_{2l}\right)\left(x_{1k} - x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(1 - \frac{1}{2}e^{\theta_k}\left(x_{1k} - x_{2k}\right)^2\right)$$

$$\frac{\partial f}{\partial x_1^k \partial x_2^l \theta_l} = -e^{\theta_k}\left(x_{1l} - x_{2l}\right)\left(x_{1k} - x_{2k}\right)\frac{\partial}{\partial \theta_l}\left(e^{\theta_l} f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\right)$$

$$\frac{\partial f}{\partial x_1^k \partial x_2^l \theta_l} = -e^{\theta_k+\theta_l}\left(x_{1l} - x_{2l}\right)\left(x_{1k} - x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(1 - \frac{1}{2}e^{\theta_l}\left(x_{1l} - x_{2l}\right)^2\right)$$

$$\frac{\partial f}{\partial x_1^k \partial x_2^l \theta_p} = \frac{1}{2} e^{\theta_l+\theta_k+\theta_p}\left(x_{1l} - x_{2l}\right)\left(x_{1k} - x_{2k}\right) f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(x_{1p} - x_{2p}\right)^2$$

A.2.2 Exponential Korrelationsfunktion mit Ableitungen

Die hier gezeigten Ableitungen sind für die Entwicklung der unterschiedlichen Kriging-Verfahren nötig und können in anderen Arbeiten dieser Richtung hilfreich sein. Aus diesem Grund werden die Ergebnisse der Ableitungen hier aufgelistet.

$$f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) = e^{-\frac{1}{2}\sum^{k}\left(e^{\theta_k}\left|x_{1k} - x_{2k}\right|^p\right)} + \delta e^{\lambda}$$

$$\frac{\partial f}{\partial \theta_d} = f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right)\left(-\frac{1}{2}e^{\theta_k}\left|x_{1d} - x_{2d}\right|^p\right)$$

$$\frac{\partial f}{\partial p} = \left(f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) - \delta e^{\lambda}\right)\left(-\frac{1}{2}\frac{\partial}{\partial p}\left(\sum^{k}\left(e^{\theta_k}\left|x_{1k} - x_{2k}\right|^{p}\right)\right)\right)$$

$$\frac{\partial f}{\partial p} = \left(f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) - \delta e^{\lambda}\right)\left(-\frac{1}{2}\sum^{k} e^{\theta_k}\frac{\partial}{\partial p}\left(\left|x_{1k} - x_{2k}\right|^{p}\right)\right)$$

$$\frac{\partial f}{\partial p} = \left(f\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) - \delta e^{\lambda}\right)\left(-\frac{1}{2}\sum^{k} e^{\theta_k}\left(\left|x_{1k} - x_{2k}\right|^{p}\right) log\left(\left|x_{1k} - x_{2k}\right|\right)\right)$$

$$\frac{\partial f}{\partial \lambda} = e^{\lambda}\delta \Rightarrow \frac{\partial R}{\partial \lambda} = e^{\lambda}E$$

A.2.3 Kubische Spline Korrelationsfunktion mit Ableitungen

Die hier gezeigten Ableitungen sind für die Entwicklung der unterschiedlichen Kriging-Verfahren nötig und können in anderen Arbeiten dieser Richtung hilfreich sein. Aus diesem Grund werden die Ergebnisse der Ableitungen hier aufgelistet.

$$f\left(x_1, x_2\right) = \prod_{j=1}^{N} p_j + \delta e^{\lambda}$$

$$\xi_j = e^{\theta_j}\left|x_{1j} - x_{2j}\right|$$

$$p_j\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) = \begin{cases} 1 - \frac{3}{a}\xi_j^2 + \frac{1+a}{a^2}\xi_j^3 & , 0 \leq \xi_j \leq a \\ \frac{1}{1-a}\left(1 - \xi_j\right)^3 & , a < \xi_j < 1 \\ 0 & , \xi_j > 1 \end{cases}$$

$$\frac{\partial f}{\partial \theta_j} = \frac{\partial p_j}{\partial \theta_j}\prod_{i \neq j}^{N} p_i$$

$$\frac{\partial p_j}{\partial \theta_j} = \begin{cases} 3\xi_j^2\left(\frac{1+a}{a^2}\xi_j - \frac{2}{a}\right) & , 0 \leq \xi_j \leq a \\ -\frac{3}{1-a}\xi_j\left(1 - \xi_j\right)^2 & , a < \xi_j < 1 \\ 0 & , \xi_j > 1 \end{cases}$$

$$\frac{\partial f}{\partial a} = \sum_{j=1}^{N} \left(\frac{\partial p_j}{\partial a} \prod_{i \neq j}^{N} p_i \right)$$

$$\frac{\partial p_j}{\partial a} = \begin{cases} \frac{3}{a^2}\xi_j^2 - \frac{a+2}{a^3}\xi_j^3 & , 0 \leq \xi_j \leq a \\ \frac{1}{(1-a)^2}\left(1-\xi_j\right)^3, & a < \xi_j < 1 \\ 0 & , \xi_j > 1 \end{cases}$$

$$\frac{\partial f}{\partial \lambda} = e^{\lambda}\delta \Rightarrow \frac{\partial R}{\partial \lambda} = e^{\lambda}E$$

Erste Ableitung nach Ort

$$\frac{\partial f\left(x_1, x_2\right)}{\partial x_1^k} = \frac{\partial p_k}{\partial x_1^k} \prod_{j=1; j \neq k}^{N} p_j$$

$$p_j\left(\overrightarrow{x}_1, \overrightarrow{x}_2\right) = \begin{cases} 1 - \frac{3}{a}\left(e^{\theta_j}\left|x_{1j} - x_{2j}\right|\right)^2 + \frac{1+a}{a^2}\left(e^{\theta_j}\left|x_{1j} - x_{2j}\right|\right)^3 & , 0 \leq \xi_j \leq a \\ \frac{1}{1-a}\left(1 - e^{\theta_j}\left|x_{1j} - x_{2j}\right|\right)^3 & , a < \xi_j < 1 \\ 0 & , \xi_j > 1 \end{cases}$$

$$\frac{\partial p_k}{\partial x_1^k} = \begin{cases} sgn\left(x_{1k} - x_{2k}\right)\frac{3}{a}e^{\theta_k}\xi_k\left(\frac{1+a}{a}\xi - 2\right) & , 0 \leq \xi_k \leq a \\ -sgn\left(x_{1k} - x_{2k}\right)\frac{3}{1-a}e^{\theta_k}\left(1-\xi_k\right)^2 & , a < \xi_k < 1 \\ 0 & , \xi_k > 1 \end{cases}$$

$$\frac{\partial p_k}{\partial x_2^k} = \begin{cases} -sgn\left(x_{1k} - x_{2k}\right)\frac{3}{a}e^{\theta_k}\xi_k\left(\frac{1+a}{a}\xi - 2\right) & , 0 \leq \xi_k \leq a \\ sgn\left(x_{1k} - x_{2k}\right)\frac{3}{1-a}e^{\theta_k}\left(1-\xi_k\right)^2 & , a < \xi_k < 1 \\ 0 & , \xi_k > 1 \end{cases}$$

Zweite Ableitung nach Ort

$$\frac{\partial^2 f\left(x_1, x_2\right)}{\partial x_1^k x_2^k} = \frac{\partial p_k^2}{\partial x_1^k x_2^k} \prod_{j=1; j \neq k}^{N} p_j$$

$$\frac{\partial p_k^2}{\partial x_1^k x_2^k} = \begin{cases} -\frac{6}{a} e^{2\theta_k} \left[\frac{1+a}{a}\xi_k - 1\right] & , 0 \leq \xi_k \leq a \\ -\frac{6}{1-a} e^{2\theta_k} \left[(1-\xi_k)\right] & , a < \xi_k < 1 \\ 0 & , \xi_k > 1 \end{cases}$$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1^k x_2^l} = \frac{\partial p_k \partial p_l}{\partial x_1^k \partial x_2^l} \prod_{j=1; j\neq k; j\neq l}^{N} p_j$$

$$\frac{\partial p_l}{\partial x_2^l} = \begin{cases} -sgn(x_{1l} - x_{2l}) \frac{3}{a} e^{\theta_l} \xi \left(\frac{1+a}{a}\xi_l - 2\right) & , 0 \leq \xi_l \leq a \\ sgn(x_{1l} - x_{2l}) \frac{3}{1-a} e^{\theta_l} (1-\xi_l)^2 & , a < \xi_l < 1 \\ 0 & , \xi_l > 1 \end{cases}$$

Erste Ableitung nach Hyperparametern:

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1^k \partial \theta_k} = \frac{\partial}{\partial \theta_k} \left(\frac{\partial p_k}{\partial x_1^k} \prod_{j=1; j\neq k}^{N} p_j \right)$$

$$= \frac{\partial}{\partial \theta_k} \left(\frac{\partial p_k}{\partial x_1^k} \right) \prod_{j=1; j\neq k}^{N} p_j$$

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1^k \partial \theta_p} = \frac{\partial}{\partial \theta_p} \left(\frac{\partial p_k}{\partial x_1^k} \prod_{j=1; j\neq k}^{N} p_j \right)$$

$$= \frac{\partial p_p}{\partial \theta_p} \frac{\partial p_k}{\partial x_1^k} \prod_{j=1; j\neq k; j\neq p}^{N} p_j$$

Zweite Ableitung nach Hyperparametern:

p=k=l

$$\frac{\partial}{\partial \theta_k} \left(\frac{\partial^2 f(x_1, x_2)}{\partial x_1^k \partial x_2^k} \right) = \frac{\partial}{\partial \theta_k} \left(\frac{\partial^2 p_k}{\partial x_1^k \partial x_2^k} \prod_{j=1; j\neq k}^{N} p_j \right)$$

$$= \frac{\partial}{\partial \theta_k} \left(\frac{\partial^2 p_k}{\partial x_1^k \partial x_2^k} \right) \prod_{j=1;j\neq k}^{N} p_j$$

p!=k=l

$$\frac{\partial}{\partial \theta_p} \left(\frac{\partial^2 f\left(x_1, x_2\right)}{\partial x_1^k \partial x_2^k} \right) = \frac{\partial}{\partial \theta_p} \left(\frac{\partial^2 p_k}{\partial x_1^k \partial x_2^k} \prod_{j=1;j\neq k}^{N} p_j \right)$$

$$= \frac{\partial^2 p_k}{\partial x_2^k \partial x_1^k} \frac{\partial p_p}{\partial \theta_p} \prod_{j=1;j\neq k;j\neq p}^{N} p_j$$

p=k!=l

$$\frac{\partial}{\partial \theta_k} \left(\frac{\partial^2 f\left(x_1, x_2\right)}{\partial x_1^k \partial x_2^l} \right) = \frac{\partial}{\partial \theta_k} \left(\frac{\partial p_k \partial p_l}{\partial x_1^k \partial x_2^l} \prod_{j=1;j\neq k;j\neq l}^{N} p_j \right)$$

$$= \frac{\partial}{\partial \theta_k} \left(\frac{\partial p_k}{\partial x_1^k} \right) \left(\frac{\partial p_l}{\partial x_2^l} \prod_{j=1;j\neq k;j\neq l}^{N} p_j \right)$$

p=l ; k!=l

$$\frac{\partial}{\partial \theta_l} \left(\frac{\partial^2 f\left(x_1, x_2\right)}{\partial x_1^k x_2^l} \right) = \frac{\partial}{\partial \theta_l} \left(\frac{\partial p_k \partial p_l}{\partial x_1^k \partial x_2^l} \prod_{j=1;j\neq k;j\neq l}^{N} p_j \right)$$

$$= \frac{\partial}{\partial \theta_l} \left(\frac{\partial p_l}{\partial x_2^l} \right) \frac{\partial p_k}{\partial x_1^k} \prod_{j=1;j\neq k;j\neq l}^{N} p_j$$

p!=k!=l

$$\frac{\partial}{\partial \theta_p} \left(\frac{\partial^2 f\left(x_1, x_2\right)}{\partial x_1^k x_2^l} \right) = \frac{\partial}{\partial \theta_p} \left(\frac{\partial p_k \partial p_l}{\partial x_1^k \partial x_2^l} \prod_{j=1;j\neq k;j\neq l}^{N} p_j \right)$$

$$= \frac{\partial p_k \partial p_l}{\partial x_1^k \partial x_2^l} \frac{\partial p_p}{\partial \theta_p} \prod_{j=1;j\neq k;j\neq l;j\neq p}^{N} p_j$$

Ableitung nach Ort und a:

$$\frac{\partial^2 f\left(x_1,x_2\right)}{\partial x_1^k \partial a} = \frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\prod_{j=1;j\neq k}^{N} p_j\right)$$

$$= \frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right)\prod_{j=1;j\neq k}^{N} p_j + \frac{\partial p_k}{\partial x_1^k}\frac{\partial}{\partial a}\left(\prod_{j=1;j\neq k}^{N} p_j\right)$$

$$= \frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right)\prod_{j=1;j\neq k}^{N} p_j + \frac{\partial p_k}{\partial x_1^k}\left(\prod_{j=1;j\neq k}^{N} p_j\right)\left(\sum_{j=1;j\neq k}^{N}\frac{\frac{\partial}{\partial a}p_j}{p_j}\right)$$

$$= \left(\prod_{j=1;j\neq k}^{N} p_j\right)\left(\frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right) + \frac{\partial p_k}{\partial x_1^k}\left(\sum_{j=1;j\neq k}^{N}\frac{\frac{\partial p_j}{\partial a}}{p_j}\right)\right)$$

Zweite Ableitung nach Ort nach a abgeleitet:

$$\frac{\partial}{\partial a}\left(\frac{\partial^2 f\left(x_1,x_2\right)}{\partial x_1^k x_2^k}\right) = \frac{\partial}{\partial a}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\prod_{j=1;j\neq k}^{N} p_j\right)$$

$$= \frac{\partial}{\partial a}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\right)\prod_{j=1;j\neq k}^{N} p_j + \frac{\partial p_k^2}{\partial x_1^k x_2^k}\frac{\partial}{\partial a}\left(\prod_{j=1;j\neq k}^{N} p_j\right)$$

$$= \frac{\partial}{\partial a}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\right)\prod_{j=1;j\neq k}^{N} p_j + \frac{\partial p_k^2}{\partial x_1^k x_2^k}\left(\prod_{j=1;j\neq k}^{N} p_j\right)\left(\sum_{j=1;j\neq k}^{N}\frac{\frac{\partial}{\partial a}p_j}{p_j}\right)$$

$$= \prod_{j=1;j\neq k}^{N} p_j\left[\frac{\partial}{\partial a}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\right) + \frac{\partial p_k^2}{\partial x_1^k x_2^k}\left(\sum_{j=1;j\neq k}^{N}\frac{\frac{\partial}{\partial a}p_j}{p_j}\right)\right]$$

$$\frac{\partial}{\partial a}\left(\frac{\partial^2 f\left(x_1,x_2\right)}{\partial x_1^k x_2^l}\right) = \frac{\partial}{\partial a}\left(\frac{\partial p_l}{\partial x_2^l}\frac{\partial p_k}{\partial x_1^k}\prod_{j=1;j\neq k;j\neq l}^{N} p_j\right)$$

$$= \frac{\partial}{\partial a}\left(\frac{\partial p_l}{\partial x_2^l}\right)\frac{\partial p_k}{\partial x_1^k}\prod_{j=1;j\neq k;j\neq l}^{N} p_j + \frac{\partial p_l}{\partial x_2^l}\frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right)\prod_{j=1;j\neq k;j\neq l}^{N} p_j + \frac{\partial p_l}{\partial x_2^l}\frac{\partial p_k}{\partial x_1^k}\frac{\partial}{\partial a}\left(\prod_{j=1;j\neq k;j\neq l}^{N} p_j\right)$$

$$= \prod_{j=1;j\neq k;j\neq l}^{N} p_j \left[\frac{\partial}{\partial a}\left(\frac{\partial p_l}{\partial x_2^l}\right)\frac{\partial p_k}{\partial x_1^k} + \frac{\partial p_l}{\partial x_2^l}\frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right) + \frac{\partial p_l}{\partial x_2^l}\frac{\partial p_k}{\partial x_1^k}\left(\sum_{j=1;j\neq k;j\neq l}^{N}\frac{\frac{\partial}{\partial a}p_j}{p_j}\right)\right]$$

Produktableitungen nach Theta

$$\xi_k = e^{\theta_k}\left|x_{1k} - x_{2k}\right|$$

$$\frac{\partial}{\partial \theta_k}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\right) = \begin{cases} -\frac{6}{a}e^{2\theta_k}\left(\frac{3(1+a)}{a}\xi_k - 2\right) & ,0 \leq \xi_k \leq a \\ -\frac{6}{1-a}e^{2\theta_k}\left(2 - 3\xi_k\right) & ,a < \xi_k < 1 \\ 0 & ,\xi_k > 1 \end{cases}$$

$$\frac{\partial}{\partial \theta_k}\left(\frac{\partial p_k}{\partial x_1^k}\right) == \begin{cases} sgn\left(x_{1k} - x_{2k}\right)\frac{3}{a}e^{\theta_k}\xi_k\left(\frac{3(1+a)}{a}\xi_k - 4\right) & ,0 \leq \xi_k \leq a \\ -sgn\left(x_{1k} - x_{2k}\right)\frac{3}{1-a}e^{\theta_k}\left(1-\xi_k\right)\left(1-3\xi_k\right) & ,a < \xi_k < 1 \\ 0 & ,\xi_k > 1 \end{cases}$$

$$\frac{\partial}{\partial \theta_p}\left(\frac{\partial p_k}{\partial x_1^k}\right) = \begin{cases} 0 & ,0 \leq \xi_k \leq a \\ 0 & ,a < \xi_k < 1 \\ 0 & ,\xi_k > 1 \end{cases}$$

Produktableitungen nach a

$$\frac{\partial}{\partial a}\left(\frac{\partial p_k^2}{\partial x_1^k x_2^k}\right) = \begin{cases} -6e^{2\theta_k}\frac{1}{a^2}\left(1 - \frac{2+a}{a}\xi_k\right) & ,0 \leq \xi_k \leq a \\ -e^{2\theta_k}\frac{6(1-\xi_k)}{(a-1)^2} & ,a < \xi_k < 1 \\ 0 & ,\xi_k > 1 \end{cases}$$

$$\frac{\partial}{\partial a}\left(\frac{\partial p_k}{\partial x_1^k}\right) = \begin{cases} sgn\left(x_{1k} - x_{2k}\right)e^{\theta_k}\xi_k\left(-3\frac{(2+a)}{a^3}\xi_k + \frac{6}{a^2}\right) & ,0 \leq \xi_k \leq a \\ -\left(sgn\left(x_{1k} - x_{2k}\right)\frac{3}{(a-1)^2}e^{\theta_k}\left(1-\xi_k\right)^2\right) & ,a < \xi_k < 1 \\ 0 & ,\xi_k > 1 \end{cases}$$

A.2.4 Herleitung der Kriging-Gewichte

Damit ergibt sich:

$$\nabla_w \left(\mathrm{var}\left[Z\left(\vec{x}\right)\right] - 2\overrightarrow{\mathrm{c}}^T\vec{w} + \vec{w}^T * \mathbf{Cov} * \vec{w}\right) + \nabla_w \left(\lambda \left(\sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \beta_k\right)\right) = \vec{0}$$
$$\wedge\ \nabla_\lambda \left(\mathrm{var}\left[Z\left(\vec{x}\right)\right] - 2\overrightarrow{\mathrm{c}}^T\vec{w} + \vec{w}^T * \mathbf{Cov} * \vec{w}\right) + \nabla_\lambda \left(\lambda \left(\sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \beta_k\right)\right) = 0$$

Aufgelöst:

$$-2\nabla_w \left(\overrightarrow{\mathrm{c}}^T\vec{w}\right) + \nabla_w \left(\vec{w}^T * \mathbf{Cov} * \vec{w}\right) + \nabla_w\lambda \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \nabla_w\lambda\beta_k = \vec{0}$$
$$\wedge\ -2\nabla_\lambda \left(\overrightarrow{\mathrm{c}}^T\vec{w}\right) + \nabla_\lambda \left(\vec{w}^T * \mathbf{Cov} * \vec{w}\right) + \nabla_\lambda\lambda \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \nabla_\lambda\lambda\beta_k = 0$$

Zwischenschritt:

$$-2\overrightarrow{\mathrm{c}} + 2\mathbf{Cov} * \vec{w} + \nabla_w\lambda \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \nabla_w\lambda\beta_k = \vec{0}$$
$$\wedge\ \nabla_\lambda\lambda \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \nabla_\lambda\lambda\beta_k = 0$$

Zwischenschritt:

$$-2\overrightarrow{\mathrm{c}} + 2\mathbf{Cov} * \vec{w} + \lambda\nabla_w \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} = \vec{0}$$
$$\wedge\ \sum_{i=1}^{s} \beta_i \sum_{j=1}^{n_i} w_{i,j} - \beta_k = 0$$

Zwischenschritt:

$$-2\overrightarrow{\mathrm{c}} + 2\mathbf{Cov} * \vec{w} + \lambda\vec{F} = \vec{0}$$
$$\wedge\ \vec{F}^T\vec{w} = \beta_k$$

Umgeformt:

$$\mathbf{Cov} * \vec{w} + \frac{\lambda\vec{F}}{2} = \vec{c}$$
$$\wedge\ \vec{F}^T\vec{w} = \beta_k$$

In Matrix Schreibweise:

$$\begin{pmatrix} \mathbf{Cov} & \vec{F} \\ \vec{F}^T & 0 \end{pmatrix} \begin{pmatrix} \vec{w} \\ \frac{\lambda}{2} \end{pmatrix} = \begin{pmatrix} \vec{c} \\ \beta_k \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} \vec{w} \\ \frac{\lambda}{2} \end{pmatrix} = \begin{pmatrix} \mathbf{Cov} & \vec{F} \\ \vec{F}^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \vec{c} \\ \beta_k \end{pmatrix} \tag{A.1}$$

Die Inverse der Blockmatrix ergibt sich nach [Thornburg, 2006] zu:

$$\begin{pmatrix} \mathbf{Cov} & \vec{F} \\ \vec{F}^T & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{Cov}^{-1} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} & \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{R}^{-1}\vec{F}} \\ \frac{\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}F} & -\frac{1}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \end{pmatrix}$$

Eingesetzt in Gleichung A.1:

$$\begin{pmatrix} \vec{w} \\ \frac{\lambda}{2} \end{pmatrix} = \begin{pmatrix} \mathbf{Cov}^{-1} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} & \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \\ \frac{\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}F} & -\frac{1}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \end{pmatrix} \begin{pmatrix} \vec{c} \\ \beta_k \end{pmatrix}$$

A.2.5 Beispiel Dichtefunktion für ein CO-Kriging Model

Folgend wird die Likelihood-Funktion eines CO-Kriging Modells einer Parabel dargestellt. Die Low-Fidelity Funktion wurde für jede Stützstelle mit einem zufälligen Wert zwischen -0.1 und 0.1 gestört und zusätzlich verschoben.

$$f_{high}(x) = x^2$$
$$f_{low}(x) = x^2 + Random(-0.1, 0.1) - 10$$

Auf der X- und Y-Achse sind die Hyperparameter $\theta_{low}, \theta_{err}$ zu erkennen und auf der Z-Achse der entsprechende Likelihood-Wert. In den verschiedenen Diagrammen wurden die Varianzen $\sigma^2_{err}, \sigma^2_{low}$ der Kovarianzfunkionen cov_{err}, cov_{low} verändert. Grundlegend ist für diese Likelihood-Funktion eine Abflachung zu den Rändern zu erkennen. Dieses Verhalten ist sehr typisch und ebenfalls wichtig, für Gradienten-basierte Trainingsverfahren, da in diesen Bereichen die Gradienten sehr klein werden können. Weiterhin ist zu erkennen, dass die Form der Funktion sehr stark auf die Änderung der Prozessvarianzen reagiert und das globale Minimum ebenfalls sehr flach ist. Das Auffinden des Minimums ist bei dieser Art von Funktion also eine sehr schwierige Aufgabe und erfordert ein robustes Trainingsverfahren.

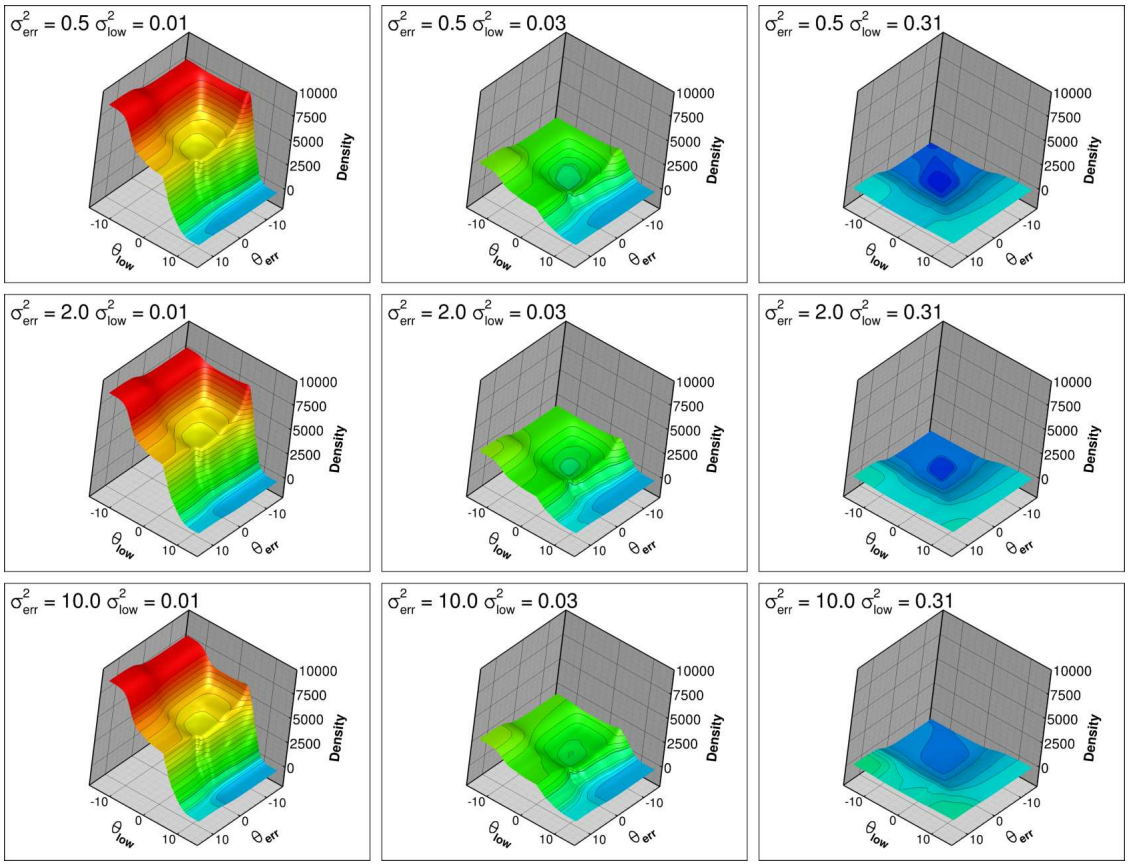


Abbildung A.7:

A.2.6 Varianz der Fehlerfunktion

$$\operatorname{var}\left[F\left(\vec{x}\right)\right] = \operatorname{var}\left[Z_k\left(\vec{x}\right) - \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i\right]$$

$$= E\left[\left(\left(Z_k\left(\vec{x}\right) - \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i\right) - E\left[Z_k\left(\vec{x}\right) - \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i\right]\right)^2\right]$$

$$= E\left[\left(Z_k\left(\vec{x}\right) - \sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i - E\left[Z_k\left(\vec{x}\right)\right] + E\left[\sum_{i=1}^{s} \vec{Z}_i^T \vec{w}_i\right]\right)^2\right]$$

Im folgenden Schritt wird das Skalarprodukt als Summe formuliert:

$$= E\left[\left(\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right] - \left[\sum_{i=1}^{s}\sum_{j=1}^{n_i} Z_i\left(\vec{x}_j\right) w_{i,j} - \sum_{i=1}^{s}\sum_{j=1}^{n_i} E\left[Z_i\left(\vec{x}_j\right)\right] w_{i,j}\right]\right)^2\right]$$

$$= E\left[\left(\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right] - \left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)\right]w_{i,j}\right)\right]\right)^2\right]$$

$$= E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]^2 - 2\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)\right]w_{i,j}\right)\right]\right.$$
$$\left. + \left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)\right]w_{i,j}\right)\right]^2\right]$$

$$= E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]^2\right] - 2E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\right]\right]$$
$$+ E\left[\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\right]^2\right]$$

$$= \mathrm{var}\left[Z_k\left(\vec{x}\right)\right] - 2E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\right]\right]$$
$$+ E\left[\left[\sum_{i=1}^{s}\left(\vec{Z}_i^T\vec{w}_i - E\left[Z_i\right]1^{\vec{T}}\vec{w}_i\right)\right]^2\right]$$

$$= \mathrm{var}\left[Z_k\left(\vec{x}\right)\right] - 2E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\right]\right]$$
$$+ E\left[\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i}\sum_{k=1}^{s}\sum_{l=1}^{n_i}\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\left(Z_k\left(\vec{x}_l\right)w_{k,l} - E\left[Z_k\left(\vec{x}_l\right)w_{k,l}\right]\right)\right]\right]$$

$$= \mathrm{var}\left[Z_k\left(\vec{x}\right)\right] - 2\sum_{i=1}^{s}\sum_{j=1}^{n_i}E\left[\left[Z_k\left(\vec{x}\right) - E\left[Z_k\left(\vec{x}\right)\right]\right]\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\right]$$
$$+ \sum_{i=1}^{s}\sum_{j=1}^{n_i}\sum_{k=1}^{s}\sum_{l=1}^{n_i}E\left[\left(Z_i\left(\vec{x}_j\right)w_{i,j} - E\left[Z_i\left(\vec{x}_j\right)w_{i,j}\right]\right)\left(Z_k\left(\vec{x}_l\right)w_{k,l} - E\left[Z_k\left(\vec{x}_l\right)w_{k,l}\right]\right)\right]$$

$$= \text{var}\left[Z_k\left(\vec{x}\right)\right] - 2\sum_{i=1}^{s}\sum_{j=1}^{n_i}\text{cov}\left(Z_k\left(\vec{x}\right), Z_i\left(\vec{x}_j\right) w_{i,j}\right) + \sum_{i=1}^{s}\sum_{j=1}^{n_i}\sum_{k=1}^{s}\sum_{l=1}^{n_i}\text{cov}\left(Z_i\left(\vec{x}_j\right) w_{i,j}, Z_k\left(\vec{x}_l\right) w_{k,l}\right)$$

$$= \text{var}\left[Z_k\left(\vec{x}\right)\right] - 2\sum_{i=1}^{s}\sum_{j=1}^{n_i} w_{i,j}\text{cov}\left(Z_k\left(\vec{x}\right), Z_i\left(\vec{x}_j\right)\right) + \sum_{i=1}^{s}\sum_{j=1}^{n_i}\sum_{k=1}^{s}\sum_{l=1}^{n_i} w_{i,j} w_{k,l}\text{cov}\left(Z_i\left(\vec{x}_j\right), Z_k\left(\vec{x}_l\right)\right)$$

In Matrix-Schreibweise ergibt sich die folgende Formulierung, wobei $\vec{w} \in \mathbb{R}^{\text{n}_{\text{all}}}$ den Gewichtsvektor, $\mathbf{Cov} \in \mathbb{R}^{\text{n}_{\text{all}} \times \text{n}_{\text{all}}}$ die Kovarianzmatrix und $\overrightarrow{\text{cov}} \in \mathbb{R}^{\text{n}_{\text{all}}}$ den Kovarianzvektor darstellt:

$$\text{var}\left[F\left(\vec{x}_0\right)\right] = \text{var}\left[Z\left(\vec{x}_0\right)\right] - 2\overrightarrow{\text{cov}}\left(Z\left(\vec{x}_0\right), Z\left(\vec{x}\right)\right)^T * \vec{w} + \vec{w}^T * \mathbf{Cov} * \vec{w} \tag{A.2}$$

A.2.7 Kovarianz zwischen Vorhersagen: Allgemeine Formulierung

$$
\begin{aligned}
cov\left(Z^*_{g_1}(\vec{x_1}), Z^*_{g_2}(\vec{x_2})\right) &= cov\left(\sum_{i=1}^{s}\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}), \sum_{j=1}^{s}\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right)\\
&= E\left[\left(\sum_{i=1}^{s}\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}) - E\left[\sum_{i=1}^{s}\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1})\right]\right)\right.\\
&\left.\left(\sum_{j=1}^{s}\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2}) - E\left[\sum_{j=1}^{s}\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right]\right)\right[\\
&= E\left[\sum_{i=1}^{s}\left(\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}) - E\left[\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1})\right]\right)\right.\\
&\left.\sum_{j=1}^{s}\left(\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2}) - E\left[\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right]\right)\right]\\
&= E\left[\sum_{i=1}^{s}\sum_{j=1}^{s}\left(\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}) - E\left[\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1})\right]\right)\right.\\
&\left.\left(\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2}) - E\left[\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right]\right)\right]\\
&= \sum_{i=1}^{s}\sum_{j=1}^{s}E\left[\left(\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}) - E\left[\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1})\right]\right)\right.\\
&\left.\left(\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2}) - E\left[\vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right]\right)\right]\\
&\Leftrightarrow \sum_{i=1}^{s}\sum_{j=1}^{s}cov\left(\vec{Z}_i^T\vec{w}_{1i}(g_1,\vec{x_1}), \vec{Z}_j^T\vec{w}_{2j}(g_2,\vec{x_2})\right)\\
&= \sum_{i=1}^{s}\sum_{j=1}^{s}cov\left(\sum_{l=1}^{n_i}Z_i(\vec{x_l})\,w_{1i,l}(g_1,\vec{x_1}), \sum_{m=1}^{n_i}Z_j(\vec{x_m})\,w_{2j,m}(g_2,\vec{x_2})\right)\\
&= \sum_{i=1}^{s}\sum_{j=1}^{s}\sum_{l=1}^{n_i}\sum_{m=1}^{n_i}cov\left(Z_i(\vec{x_l})\,w_{1i,l}(g_1,\vec{x_1}), Z_j(\vec{x_m})\,w_{2j,m}(g_2,\vec{x_2})\right)\\
&= \sum_{i=1}^{s}\sum_{j=1}^{s}\sum_{l=1}^{n_i}\sum_{m=1}^{n_i}w_{1i,l}(g_1,\vec{x_1})w_{2j,m}(g_2,\vec{x_2})cov\left(Z_i(\vec{x_l}), Z_j(\vec{x_m})\right)
\end{aligned}
$$

A.2.8 Kovarianz zwischen Vorhersagen: Gewichte eingesetzt

$$cov\left(Z^*_{g_1}(\vec{x_1}), Z^*_{g_2}(\vec{x_2})\right) = \vec{w_1}^T(g_1,\vec{x_1})\mathbf{Cov}\vec{w}_2(g_2,\vec{x_2}) \tag{A.3}$$

Die Gewichtsvektoren sehen folgendermaßen aus:

$$\vec{w_1}(g_1,\vec{x_1}) = \mathbf{Cov}^{-1}\vec{c_1}(g_1,\vec{x_2}) - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_1}(g_1,\vec{x_2}) + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1}$$

$$\vec{w_2}(g_2,\vec{x_2}) = \mathbf{Cov}^{-1}\vec{c_2}(g_2,\vec{x_2}) - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_2}(g_2,\vec{x_2}) + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2}$$

Weiterhin werden folgende Abkürzungen einführt: $\vec{c_2}(g_2,\vec{x_2}) = \vec{c_2}$ und $\vec{c_1}(g_1,\vec{x_1}) = \vec{c_1}$
Eingesetzt in die ursprüngliche Formel A.3:

$$cov\left(Z^*_{g_1}(\vec{x_1}), Z^*_k(\vec{x_2})\right) = \vec{w_1}^T(g_1,\vec{x_1})\mathbf{Cov}\vec{w}_2(g_2,\vec{x_2})$$

$$\Leftrightarrow \left(\mathbf{Cov}^{-1}\vec{c_1} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_1} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1}\right)^T * \mathbf{Cov}\left(\mathbf{Cov}^{-1}\vec{c_2} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_2} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2}\right)$$

$$\Leftrightarrow \left(\mathbf{Cov}^{-1}\vec{c_1} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_1} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1}\right)^T \left(\vec{c_2} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_2} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2}\right)$$

$$\Leftrightarrow \left(\vec{c_1}^T\mathbf{Cov}^{-1} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} + \frac{\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1}\right) \left(\vec{c_2} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_2} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2}\right)$$

$$\begin{aligned}\Leftrightarrow \vec{c_1}^T\mathbf{Cov}^{-1}\vec{c_2} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} + \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \\ + \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1} \\ + \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2} - \frac{\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c_2}\beta_{g_2} + \frac{\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1}\beta_{g_2}\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \vec{c_1}^T\mathbf{Cov}^{-1}\vec{c_2} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} + \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} \\ + \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_1} \\ + \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2} - \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_{g_2} + \frac{\beta_{g_1}\beta_{g_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\end{aligned}$$

$$cov\left(Z^*_{g_1}\left(\vec{x_1}\right), Z^*_{g_2}\left(\vec{x_2}\right)\right) = \vec{c_1}^T\mathbf{Cov}^{-1}\vec{c_2} - \frac{\vec{c_1}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}} + \frac{\beta_{g_1}\beta_{g_2}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}$$

A.2.9 Varianz des Schätzfehlers

$$\vec{w} = \mathbf{Cov}^{-1}\vec{c} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k$$

$$\mathrm{var}\left[F\left(\vec{x}\right)\right] = \mathrm{var}\left[Z\left(\vec{x}\right)\right] - 2\overrightarrow{\mathbf{c}}^T\vec{w} + \vec{w}^T * \mathbf{Cov} * \vec{w}$$

$$\begin{aligned}\mathrm{var}\left[F\left(\vec{x}_0\right)\right] = \mathrm{var}\left[Z\left(\vec{x}_0\right)\right] - 2\vec{c}^T\left(\mathbf{Cov}^{-1}\vec{c} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right) \\ + \left(\mathbf{Cov}^{-1}\vec{c} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)^T \\ \mathbf{Cov}\left(\mathbf{Cov}^{-1}\vec{c} - \frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \mathrm{var}\left[Z\left(\vec{x}_0\right)\right] - 2\vec{c}^T\mathbf{Cov}^{-1}\vec{c} + 2\vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} - 2\vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k \\ + \left(\vec{c}^T\mathbf{Cov}^{-1} - \vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\beta_k\right) \\ \left(\vec{c} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] &- 2\vec{c}^T\mathbf{Cov}^{-1}\vec{c} + 2\vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} - 2\vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k \\ &+\vec{c}^T\mathbf{Cov}^{-1}\vec{c} - \frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k \\ &+\left(-\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\beta_k\right) \\ &\left(\vec{c} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] &- \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} - \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k \\ &+\left(-\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\beta_k\right) \\ &\left(\vec{c} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] &- \vec{c}^T\mathbf{Cov}^{-1}\vec{c} - \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} \\ &-\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\beta_k \\ &\left(\vec{c} - \frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} + \frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k\right)\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] &- \vec{c}^T\mathbf{Cov}^{-1}\vec{c} - \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} \\ &-\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\vec{c}\beta_k \\ &-\left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k^2\end{aligned}$$

$$\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} - \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\vec{c}\beta_k$$
$$+\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} - \vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k$$
$$-\left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k^2$$

Zwischenrechnung:

$$\vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k = \frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k = \frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k$$

Da der Nenner und der Zähler ein Skalar ergeben, gilt:

$$\frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k = \left(\frac{\vec{c}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\beta_k = \frac{\vec{c}\mathbf{Cov}^{-1}\vec{F}^T}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k$$

Eingesetzt in die ursprüngliche Gleichung:

$$\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \vec{c}^T\left(\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c}$$
$$-2\vec{c}\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \left(\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\right)^T\frac{\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k^2$$

$$\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c}$$
$$-2\vec{c}\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \frac{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k^2$$

$$\Leftrightarrow \operatorname{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \vec{c}^T\frac{\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\vec{c} - 2\vec{c}\frac{\mathbf{Cov}^{-1}\vec{F}}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\beta_k + \frac{\beta_k^2}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}$$

$$\Leftrightarrow \text{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \frac{1}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\left(\vec{c}^T\mathbf{Cov}^{-1}\vec{F}\vec{F}^T\mathbf{Cov}^{-1}\vec{c} - 2\vec{c}\mathbf{Cov}^{-1}\vec{F}\beta_k + \beta_k^2\right)$$

Bei dem Term in der Klammer handelt es sich nur um Skalare und damit entspricht dies einer quadratischen Gleichung, also gilt:

$$\text{var}\left[F\left(\vec{x}_0\right)\right] = \text{var}\left[Z\left(\vec{x}_0\right)\right] - \vec{c}^T\mathbf{Cov}^{-1}\vec{c} + \frac{1}{\vec{F}^T\mathbf{Cov}^{-1}\vec{F}}\left(\vec{c}^T\mathbf{Cov}^{-1}\vec{F} - \beta_k\right)^2$$

A.2.10 Co-Kriging Kovarianzfunktion Herleitung

$$cov\left(Z_1\left(\vec{x}\right), Z_1(\vec{x})\right) = cov\left(aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right), aZ_2(\vec{x}^{'}) + Z_{diff}(\vec{x}^{'})\right)$$

$$\begin{aligned}
&= E\Big[\Big((aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)) - E[aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)]\Big) \\
&\Big((aZ_2(\vec{x}) + Z_{diff}(\vec{x}^{'})) - E[aZ_2(\vec{x}^{'}) + Z_{diff}(\vec{x}^{'})]\Big)\Big]
\end{aligned}$$

$$\begin{aligned}
=&E\Big[\Big((aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)) - E\left[aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)\right]\Big) \\
&\Big(aZ_2(\vec{x}^{'}) - aE[Z_1(\vec{x}^{'})] + Z_{diff}(\vec{x}^{'}) - E[Z_{diff}(\vec{x}^{'})]\Big)\Big]
\end{aligned}$$

$$\begin{aligned}
=&E\Big[\Big((aZ_2\left(\vec{x}\right) - aE[Z_1\left(\vec{x}\right)]) + (Z_{diff}\left(\vec{x}\right) - E\left[Z_{diff}\left(\vec{x}\right)]\right)\Big) \\
&\Big((aZ_2(\vec{x}^{'}) - aE[Z_1(\vec{x}^{'})]) + (Z_{diff}(\vec{x}^{'}) - E[Z_{diff}(\vec{x}^{'})])\Big)\Big]
\end{aligned}$$

$$\begin{aligned}
=&E\Big[a^2\left(Z_2\left(\vec{x}\right) - E\left[Z_2\left(\vec{x}\right)\right]\right)\left(Z_2(\vec{x}^{'}) - E[Z_2(\vec{x}^{'})]\right)\Big] \\
&+ E\Big[a\left(Z_2\left(\vec{x}\right) - E\left[Z_2\left(\vec{x}\right)\right]\right)\left(Z_{diff}(\vec{x}^{'}) - E[Z_{diff}(\vec{x}^{'})]\right)\Big] \\
&+ E\Big[a\left(Z_{diff}\left(\vec{x}\right) - E\left[Z_{diff}\left(\vec{x}\right)\right]\right)\left(Z_2(\vec{x}^{'}) - E[Z_2(\vec{x}^{'})]\right)\Big] \\
&+ E\Big(\left(Z_{diff}\left(\vec{x}\right) - E\left[Z_{diff}\left(\vec{x}\right)\right]\right)\left(Z_{diff}(\vec{x}^{'}) - E[Z_{diff}(\vec{x}^{'})]\right)\Big)
\end{aligned}$$

$$
\begin{aligned}
=&a^2 cov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) + acov\left(Z_2\left(\vec{x}\right), Z_{diff}(\vec{x}^{'})\right) \\
&+ acov\left(Z_{diff}\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) + cov\left(Z_{diff}\left(\vec{x}\right), Z_{diff}(\vec{x}^{'})\right)
\end{aligned}
$$

Nimmt man ferner an, dass der Differenzprozess $Z_{diff}\left(\vec{x}\right)$ und der Prozess $Z_2\left(\vec{x}\right)$ unkorreliert sind, so gilt:

$$
\begin{aligned}
cov\left(Z_2\left(\vec{x}\right), Z_{diff}(\vec{x}^{'})\right) &= 0 \\
cov\left(Z_{diff}\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) &= 0
\end{aligned}
$$

daraus folgt:

$$
cov\left(Z_1\left(\vec{x}\right), Z_1(\vec{x}^{'})\right) = a^2 cov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) + cov\left(Z_{diff}\left(\vec{x}\right), Z_{diff}(\vec{x}^{'})\right)
$$

Modelliert man nun $cov\left(Z_1\left(\vec{x}\right), Z_2\left(\vec{x}^{'}\right)\right)$ nach demselben Schema, ergibt sich:

$$
cov\left(Z_1\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) = cov\left(aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right), Z_2(\vec{x}^{'})\right)
$$

$$
=E\left[\left(\left(aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)\right) - E\left[aZ_2\left(\vec{x}\right) + Z_{diff}\left(\vec{x}\right)\right]\right)\left(Z_2(\vec{x}^{'}) - E\left[Z_2(\vec{x}^{'})\right]\right)\right]
$$

$$
=E\left[\left(\left(aZ_2\left(\vec{x}\right) - E\left[aZ_2\left(\vec{x}\right)\right]\right) + \left(Z_{diff}\left(\vec{x}\right) - E\left[Z_{diff}\left(\vec{x}\right)\right]\right)\right)\left(Z_2(\vec{x}^{'}) - E\left[Z_2(\vec{x}^{'})\right]\right)\right]
$$

$$
\begin{aligned}
=&E\left[\left(aZ_2\left(\vec{x}\right) - E\left[aZ_2\left(\vec{x}\right)\right]\right)\left(Z_2(\vec{x}^{'}) - E\left[Z_2(\vec{x}^{'})\right]\right)\right] \\
&+ E\left[\left(Z_{diff}\left(\vec{x}\right) - E\left[Z_{diff}\left(\vec{x}\right)\right]\right)\left(Z_2\left(\vec{x}^{'}\right) - E\left[Z_2(\vec{x}^{'})\right]\right)\right]
\end{aligned}
$$

$$
=acov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) + cov\left(Z_{diff}\left(\vec{x}\right), Z_2(\vec{x}^{'})\right)
$$

Es gilt wieder:

$$
cov\left(Z_{diff}\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) = 0
$$

Daraus ergibt sich:

$$
cov\left(Z_1\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) = acov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right)
$$

Analog gilt für die letzte Kovarianzfunktion (siehe Gleichung 4.18):

$$cov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right) = cov\left(Z_2\left(\vec{x}\right), Z_2(\vec{x}^{'})\right)$$

A.3 Kapitel 5

A.3.1 Ableitungen des Kovarianzmodells nach den Rauschtermen

Im Fall des Co-Krigings mit den zwei Rauschtermen $\lambda_{diff}, \lambda_2 \in \mathbb{R}$ sehen die partiellen Ableitungen wie folgt aus:

$$\frac{cov\left(Z_1\left(\vec{x}_1\right), Z_2\left(\vec{x}_2\right)\right)}{\partial \lambda_{diff}} = \begin{array}{l} \frac{cov\left(Z_1\left(\vec{x}_1\right), Z_1\left(\vec{x}_2\right)\right)}{\partial \lambda_{diff}} = 1 \\ \frac{cov\left(Z_1\left(\vec{x}_1\right), Z_1\left(\vec{x}_2\right)\right)}{\partial \lambda_2} = 0 \\ \frac{cov\left(Z_1\left(\vec{x}_1\right), Z_2\left(\vec{x}_2\right)\right)}{\partial \lambda_2} = 0 \\ \frac{cov\left(Z_2\left(\vec{x}_1\right), Z_2\left(\vec{x}_2\right)\right)}{\partial \lambda_{diff}} = 0 \\ \frac{cov\left(Z_2\left(\vec{x}_1\right), Z_2\left(\vec{x}_2\right)\right)}{\partial \lambda_2} = 1 \end{array} \tag{A.4}$$

A.3.2 Likelihood Schätzer Erwartungswerte und Varianz

A.3.3

$$MLE\left(\vec{\vec{F}}\right) = max\left(\frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{Cov}\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)}\right)$$

$$\vec{0} = \frac{\partial}{\partial \vec{\vec{F}}}\left(\frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{Cov}\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)}\right)$$

$$\vec{0} = \frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{Cov}\right|^{\frac{1}{2}}} \frac{\partial}{\partial \vec{\vec{F}}}\left(-\frac{1}{2}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)\right) e^{-\frac{1}{2}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)}$$

Die Ableitung der quadratischen Form mit symmetrischer Matrix A lautet $\frac{\partial}{\partial x} g\left(x\right)^T \mathbf{A} g\left(x\right) = g\left(x\right)^T \mathbf{A} \frac{\partial g(x)}{\partial x} + \frac{\partial g(x)}{\partial x}^T \mathbf{A} g\left(x\right) = 2g\left(x\right)^T \mathbf{A} \frac{\partial g(x)}{\partial x}$

$$\vec{0} = \frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{Cov}\right|^{\frac{1}{2}}}\left(\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\mathbf{G}\right) e^{-\frac{1}{2}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)^T \mathbf{Cov}^{-1}\left(\overrightarrow{y}-\mathbf{G}\vec{\vec{F}}\right)}$$

Da nur $\left(\overrightarrow{y} - \mathbf{G}\vec{\tilde{F}}\right)^T \mathbf{Cov}^{-1}\mathbf{G}$ zu 0 werden kann:

$$\left(\overrightarrow{y} - \mathbf{G}\vec{\tilde{F}}\right)^T \mathbf{Cov}^{-1}\mathbf{G} = \vec{0}$$
$$\left(\overrightarrow{y}^T \mathbf{Cov}^{-1}\mathbf{G}\right)\left(\mathbf{G}^T \mathbf{Cov}^{-1}\mathbf{G}\right)^{-1} = \vec{\tilde{F}}^T$$

A.3.3 Beweis: Variogramm Abhängigkeit von Distanz

Behauptung: Sei $Z(X)$ ein schwach stationärer räumlicher Zufallsprozess, so gilt:

$$cov(\triangle\vec{x}) = cov\left(\vec{0}\right) - 2\gamma(\triangle\vec{x}) \tag{A.5}$$

Weiterhin wird benötigt:

$$E\left[Z(\vec{0})^2\right] = cov\left(Z(\vec{0}), Z(\vec{0})\right) - \mu^2 \tag{A.6}$$

Beweis:

$$\begin{aligned}
cov\left(Z(\vec{0}), Z(\vec{0})\right) =& E\left[\left(Z(\vec{0}) - E\left[Z(\vec{0})\right]\right)\left(Z(\vec{0}) - E\left[Z(\vec{0})\right]\right)\right] \\
=& E\left[Z(\vec{0})^2 - 2E\left[Z(\vec{0})\right]Z(\vec{0}) + E\left[Z(\vec{0})\right]^2\right] \\
=& E\left[Z(\vec{0})^2 - 2\mu Z(\vec{0}) + \mu^2\right] \\
=& E\left[Z(\vec{0})^2\right] - 2\mu^2 + \mu^2 \\
cov\left(Z(\vec{0}), Z(\vec{0})\right) - \mu^2 =& E\left[Z(\vec{0})^2\right]
\end{aligned}$$

Daraus ergibt sich die notwendige Umformung:

$$cov\left(Z(\vec{0}+\triangle\vec{x}), Z(\vec{0}+\triangle\vec{x})\right) - \mu^2 = E\left[Z(\vec{0}+\triangle\vec{x})^2\right] \tag{A.7}$$

Außerdem wird folgende Umformung nötig:

$$E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right] = cov\left(Z\left(\vec{0}+\triangle\vec{x}\right), Z(\vec{0})\right) + \mu^2 \tag{A.8}$$

Beweis:

$$
\begin{aligned}
cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0})\right) =& E\left[\left(Z(\vec{0}+\triangle\vec{x})-E\left[Z(\vec{0}+\triangle\vec{x})\right]\right)\left(Z(\vec{0})-E\left[Z(\vec{0})\right]\right)\right]\\
=& E\left[\left(Z(\vec{0}+\triangle\vec{x})-\mu\right)\left(Z(\vec{0})-\mu\right)\right]\\
=& E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})-Z(\vec{0}+\triangle\vec{x})\mu-\mu Z(\vec{0})+\mu^2\right]\\
=& E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right]-\mu E\left[Z(\vec{0}+\triangle\vec{x})\right]-\mu E\left[Z(\vec{0})\right]+\mu^2\\
=& E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right]-\mu^2\\
cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0})\right)+\mu^2 =& E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right]
\end{aligned}
$$

Beweis Gl. A.5:

$$
\begin{aligned}
2\gamma\left(\vec{0},\vec{0}+\triangle\vec{x}\right) &= \frac{1}{2}var\left[Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right]\\
&= \frac{1}{2}E\left[\left(\left(Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right)-E\left[Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right]\right)^2\right]\\
&= E\left[\left(Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right)^2-2\left(Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right)(\mu-\mu)+(\mu-\mu)^2\right]\\
&= E\left[\left(Z(\vec{0}+\triangle\vec{x})-Z(\vec{0})\right)^2\right]\\
&= E\left[Z(\vec{0}+\triangle\vec{x})^2-2Z(\vec{0}+\triangle\vec{x})Z(\vec{0})+Z(\vec{0})^2\right]\\
&= E\left[Z(\vec{0}+\triangle\vec{x})^2\right]-2E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right]+E\left[Z(\vec{0})^2\right]\\
&= cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0}+\triangle\vec{x})\right)-\mu^2-2E\left[Z(\vec{0}+\triangle\vec{x})Z(\vec{0})\right]\\
&+ cov\left(Z(\vec{0}),Z(\vec{0})\right)-\mu^2\\
&= cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0}+\triangle\vec{x})\right)-2cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0})\right)\\
&+ 2\mu^2+cov\left(Z(\vec{0}),Z(\vec{0})\right)-2\mu^2\\
&= cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0}+\triangle\vec{x})\right)-2cov\left(Z(\vec{0}+\triangle\vec{x}),Z(\vec{0})\right)\\
&+ cov\left(Z(\vec{0}),Z(\vec{0})\right)\\
&= cov\left(\triangle\vec{x}\right)-2cov\left(\triangle\vec{x}\right)+cov\left(Z(\vec{0}),Z(\vec{0})\right)\\
&= cov\left(Z(\vec{0}),Z(\vec{0})\right)-cov\left(\triangle\vec{x}\right)
\end{aligned}
$$

A.3.4 Differentiation der Likelihood-Funktion

Behauptung:

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\operatorname{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right) + \left(\left(\vec{y}_s - \overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\mathbf{Cov}^{-1}\left(\vec{y}_s - \overrightarrow{F}\right)\right) \tag{A.9}$$

Beweis: Da im Co-Kriging die Prozessvarianz σ^2 nicht mehr durch einen Maximum Likelihood Schätzer bestimmt werden kann, muss dies in der Ableitung der Likelihoodfunktion berücksichtigt werden. Also geht man davon aus, dass die Kovarianzmatrix von einem beliebigen Hyperparameter h abhängt, so kann man die Ableitung wie folgt bilden:

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\frac{\partial}{\partial h_l}\left(\log\left(\det(\mathbf{Cov})\right)\right) - \frac{\partial}{\partial h_l}\left(\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)\right)$$

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\frac{1}{\det(\mathbf{Cov})}\frac{\partial}{\partial h_l}\left(\det(\mathbf{Cov})\right) - \frac{\partial}{\partial h_l}\left(\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)\right)$$

Es gilt Allgemein: $\frac{\partial \det(\mathbf{Cov})}{\partial h_l} = \det(\mathbf{Cov})\operatorname{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right)$ und $\frac{\partial \mathbf{Cov}^{-1}}{\partial h_l} = -\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\mathbf{Cov}^{-1}$
Daraus folgt:

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\frac{\det(\mathbf{Cov})}{\det(\mathbf{Cov})}\operatorname{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right) + \left(\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\mathbf{Cov}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)\right)$$

$$\frac{\partial L\left(\vec{h}\right)}{\partial h_l} = -\operatorname{Spur}\left(\mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\right) + \left(\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \mathbf{Cov}^{-1}\frac{\partial \mathbf{Cov}}{\partial h_l}\mathbf{Cov}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)\right)$$

A.3.5 Likelihood Schätzer Varianzen im CO-Kriging

Im Folgenden wird versucht die Kriging Varianzen eines Co-Kriging Modells mit 2 Gütestufen analytisch zu schätzen. Der Ansatz erfolgt über folgende Block-Matrix

$$\mathbf{Cov} = \begin{bmatrix} a^2\sigma_{low}^2 corr_{low} + \sigma_{err}^2 corr_{err} & a\sigma_{low}^2 corr_{low} \\ a\sigma_{low}^2 corr_{low} & \sigma_{low}^2 corr_{low} \end{bmatrix}$$

Teilt man diese in 4 Submatrizen auf, so ergibt sich:

$$\mathbf{Cov} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$\mathbf{Cov} = \begin{bmatrix} a^2\sigma_{low}^2 C_{11A} + a^2\sigma_{err}^2 C_{11B} & a\sigma_{low}^2 C_{12} \\ a\sigma_{low}^2 C_{21} & \sigma_{low}^2 C_{22} \end{bmatrix}$$

$$\mathbf{Cov^{-1}} = \begin{bmatrix} \left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} & -\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1} \\ -C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} & C_{22}^{-1} + C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1} \end{bmatrix}$$

Betrachtet man nur den oberen linken Block der Matrix:

$$\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} = \left(\left(a^2\sigma_{low}^2 C_{11A} + a^2\sigma_{err}^2 C_{11B}\right) - a\sigma_{low}^2 C_{12}\left(\sigma_{low}^2 C_{22}\right)^{-1} a\sigma_{low}^2 C_{12}\right)^{-1}$$

$$= \left(a^2\left(\sigma_{low}^2 C_{11A} + \sigma_{err}^2 C_{11B}\right) - a^2 C_{12}\left(C_{22}\right)^{-1}\sigma_{low}^2 C_{12}\right)^{-1}$$

$$= \left(a^2\left(\sigma_{low}^2 C_{11A} + \sigma_{err}^2 C_{11B} - \sigma_{low}^2 C_{12}\left(C_{22}\right)^{-1} C_{12}\right)\right)^{-1}$$

$\mathbf{Cov^{-1}}\vec{G} =$

$$\begin{bmatrix} \left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} & -\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1} \\ -C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} & C_{22}^{-1} + C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1} \end{bmatrix} \begin{bmatrix} \vec{1} & \vec{0} \\ \vec{0} & \vec{1} \end{bmatrix}$$

$\mathbf{Cov^{-1}}\vec{G} =$

$$\begin{bmatrix} \vec{1}^T\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1}\vec{1} & -\vec{1}^T\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1}\vec{1} \\ -\vec{1}^T C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1}\vec{1} & \vec{1}^T\left(C_{22}^{-1} + C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1}\right)\vec{1} \end{bmatrix}$$

$\vec{G}^T\mathbf{Cov^{-1}}\vec{G} =$

$$\begin{bmatrix} \vec{1}^T\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1}\vec{1} & -\vec{1}^T\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1}\vec{1} \\ -\vec{1}^T C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1}\vec{1} & \vec{1}^T\left(C_{22}^{-1} + C_{22}^{-1}C_{12}\left(C_{11} - C_{12}C_{22}^{-1}C_{12}\right)^{-1} C_{12}C_{22}^{-1}\right)\vec{1} \end{bmatrix}$$

A.3.6 Vereinfachung der Vorhersagegleichung

An dieser Stelle wird die Frage geklärt, inwiefern μ und der Likelihood Schätzer für $\vec{\beta} \in \mathbb{R}^{1\times s}$ zusammenhängen. Der Likelihood Schätzer für $\vec{\beta} \in \mathbb{R}^{1\times s}$ sieht wie folgt aus:

$$\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right) = \vec{\tilde{\beta}}$$

Der Vektor $\vec{F} \in \mathbb{R}^{\mathrm{n}\times 1}$ lässt sich folgendermaßen umformulieren, wobei die Definition der Matrix $G \in \mathbb{R}^{\mathrm{n}\times \mathrm{s}}$ in Gleichung 5.39 zu finden ist.

$$\vec{F} = \begin{bmatrix} \vec{\beta}_1 \\ \vec{\beta}_2 \end{bmatrix} = G\vec{\tilde{\beta}} = G\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)$$

$$\mu = \frac{\left(G\vec{\tilde{\beta}}\right)^T \mathbf{Cov}^{-1} \vec{y}}{\left(G\vec{\tilde{\beta}}\right)^T \mathbf{Cov}^{-1} G\vec{\tilde{\beta}}}$$

$$\mu = \frac{\left(G\vec{\tilde{\beta}}\right)^T \mathbf{Cov}^{-1} \vec{y}}{\left(G\vec{\tilde{\beta}}\right)^T \mathbf{Cov}^{-1} G\vec{\tilde{\beta}}}$$

$$\mu = \frac{\left(G\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)\right)^T \mathbf{Cov}^{-1} \vec{y}}{\left(G\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)\right)^T \mathbf{Cov}^{-1} G\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)}$$

$$\mu = \frac{\left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)^T \left(G^T \mathbf{Cov}^{-1} G\right)^{-1} G^T \mathbf{Cov}^{-1} \vec{y}}{\left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)^T \left(G^T \mathbf{Cov}^{-1} G\right)^{-1} G^T \mathbf{Cov}^{-1} G\left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)}$$

$$\mu = \frac{\left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)^T \left(G^T \mathbf{Cov}^{-1} G\right)^{-1} G^T \mathbf{Cov}^{-1} \vec{y}}{\left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)^T \left(G^T \mathbf{Cov}^{-1} G\right)^{-1} \left(G^T \mathbf{Cov}^{-1} \overrightarrow{y}\right)}$$

Daraus folgt letztlich:

$$\mu = 1$$

A.3.7 Iterative Varianzbestimmung innerhalb des Co-Kriging Trainingsverfahrens

Durch die Umformulierung der Korrelationsmatrix zur Kovarianzmatrix, muss der Maximum Likelihood Schätzer der globalen Varianz σ^2 für eine multivariate Normalverteilung umformuliert werden:

$$\sigma_{k+1}^2 = \frac{1}{n}\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \sigma_k^2 \mathbf{Cov}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)$$

$$\sigma_{k+1}^2 = \frac{1}{n}\left(\vec{y}_s - \beta\overrightarrow{F}\right)^T \sigma_k^2 \frac{1}{\sigma_k^2}\mathbf{Corr}^{-1}\left(\vec{y}_s - \beta\overrightarrow{F}\right)$$

Wobei k den aktuellen Iterationsschritt angibt.

Hierbei gibt es allerdings einen Nachteil. Es ist notwendig bei der ersten Iteration einen Startwert für σ_k^2 zu bestimmen. Mit diesem Startwert wird nun $\mathbf{Cov}^{-1}$ bestimmt. Nachdem $\mathbf{Cov}^{-1}$ bestimmt wurde, wird der Likelihood Schätzer für σ_{k+1}^2 berechnet und damit ist das korrekte σ_{k+1}^2 bekannt. Die Kovarianzmatrix wurde allerdings noch mit dem σ_k^2 aufgestellt und ist somit ungültig. Es wäre an dieser Stelle also nötig die Kovarianzmatrix neu aufzustellen und alle Schritte ein zweites Mal zu durchlaufen.

Bei einem Minimierungsverfahren ist das meist nicht notwendig, da die Schätzungen sich nicht sehr stark verändern. Bei einer zufälligen Initialisierung kann dies allerdings zu Problemen führen, die Update Schritte sollten in diesem Fall öfter wiederholt werden.

A.3.8 Restandardisierung der Hyperparameter

Möchte man mit bestehenden Hyperparametern, aber einer neuen oder erweiterten Datenbasis Vorhersagen treffen, so ändern sich die Erwartungswerte der zu trainierenden Funktion und deren Parametern. Da das alte Training und damit auch die Daten des Trainings mit den alten Erwartungswerten und Standartabweichungen normalisiert worden sind, müssen die Hyperparameter ebenfalls renormalisiert werden. Ansonsten würde man für die Kovarianz zwischen 2 Membern unterschiedliche Werte bekommen.

Daraus ergibt sich folgende notwendige Bedingung:

$$cov\left(\vec{x}_{1alt}, \vec{x}_{2alt}\right) = cov\left(\vec{x}_{1neu}, \vec{x}_{2neu}\right) \tag{A.10}$$

Wobei x_{real} den nicht standardisierten Parameter darstellt, μ_{alt} und σ_{alt} stellen den alten Erwartungswert sowie die Standardabweichung der Parameter dar.

$$\vec{x}_{1alt} = \begin{bmatrix} \frac{x_{1,1real} - \mu_{1alt}}{\sigma_{1alt}} \\ \vdots \\ \frac{x_{1,nreal} - \mu_{nalt}}{\sigma_{nalt}} \end{bmatrix}$$

Für $\vec{x}_{2alt}$, sowies $\vec{x}_{1neu}$ gilt analoges.

Die Bedingung A.10 soll anhand eines Beispiels erläutert werden, es wird hierfür eine Gauss Korrelationsfunktion mit einem Hyperparameter verwendet:

$$\sigma^2_{KriAlt} e^{-\frac{1}{2}e^{\theta_{alt}}\left|\frac{x_{1real}-\mu_{alt}}{\sigma_{alt}}-\frac{x_{2real}-\mu_{alt}}{\sigma_{alt}}\right|^2} = \sigma^2_{KriNeu} e^{-\frac{1}{2}e^{\theta_{neu}}\left|\frac{x_{1real}-\mu_{neu}}{\sigma_{neu}}-\frac{x_{2real}-\mu_{neu}}{\sigma_{neu}}\right|^2}$$

$$\sigma^2_{KriAlt} e^{-\frac{1}{2}e^{\theta_{alt}}\frac{1}{\sigma^2_{alt}}|x_{1real}-x_{2real}|^2} = \sigma^2_{KriNeu} e^{-\frac{1}{2}e^{\theta_{neu}}\frac{1}{\sigma^2_{neu}}|x_{1real}-x_{2real}|^2}$$

Geht man nun davon aus, dass die Kriging Varianz sich nicht ändert:

$$\sigma^2_{KriAlt} = \sigma^2_{KriNeu}$$

Wobei diese Bedingung im Code unbedingt erfüllt sein muss. Im Code ist es so umgesetzt, dass zuerst die Kovarianzmatrix mit der alten Krigingvarianz erzeugt wird und danach der Likelihood Schätzer für die Krigingvarianz aufgerufen wird.

Dieser Schätzer sollte im Normalfall allerdings auf eine neue Krigingvarianz kommen, wodurch es im weiteren Verlauf zu großen Problemen kommen kann. Insbesondere bei der Vorhersage, wo für den Kovarianzvektor $\vec{c}$ dann die neue Krigingvarianz verwendet werden würde. Kovarianzmatrix und Vektor würden dann nicht mehr zusammen passen.

Die Formel lässt sich damit weiterhin vereinfachen:

$$e^{-\frac{1}{2}e^{\theta_{alt}}\frac{1}{\sigma^2_{alt}}|x_{1real}-x_{2real}|^2} = e^{-\frac{1}{2}e^{\theta_{neu}}\frac{1}{\sigma^2_{neu}}|x_{1real}-x_{2real}|^2}$$

$$e^{\theta_{alt}}\frac{1}{\sigma^2_{alt}}\left|x_{1real}-x_{2real}\right|^2 = e^{\theta_{neu}}\frac{1}{\sigma^2_{neu}}\left|x_{1real}-x_{2real}\right|^2$$

$$e^{\theta_{alt}}\frac{1}{\sigma^2_{alt}} = e^{\theta_{neu}}\frac{1}{\sigma^2_{neu}}$$

$$e^{\theta_{alt}}\frac{\sigma^2_{neu}}{\sigma^2_{alt}} = e^{\theta_{neu}}$$

$$\log\left(e^{\theta_{alt}}\frac{\sigma^2_{neu}}{\sigma^2_{alt}}\right) = \theta_{neu}$$

$$\theta_{alt} + \log\left(\frac{\sigma^2_{neu}}{\sigma^2_{alt}}\right) = \theta_{neu}$$

A.3.9 UML Diagramm: ZMQClient Klasse

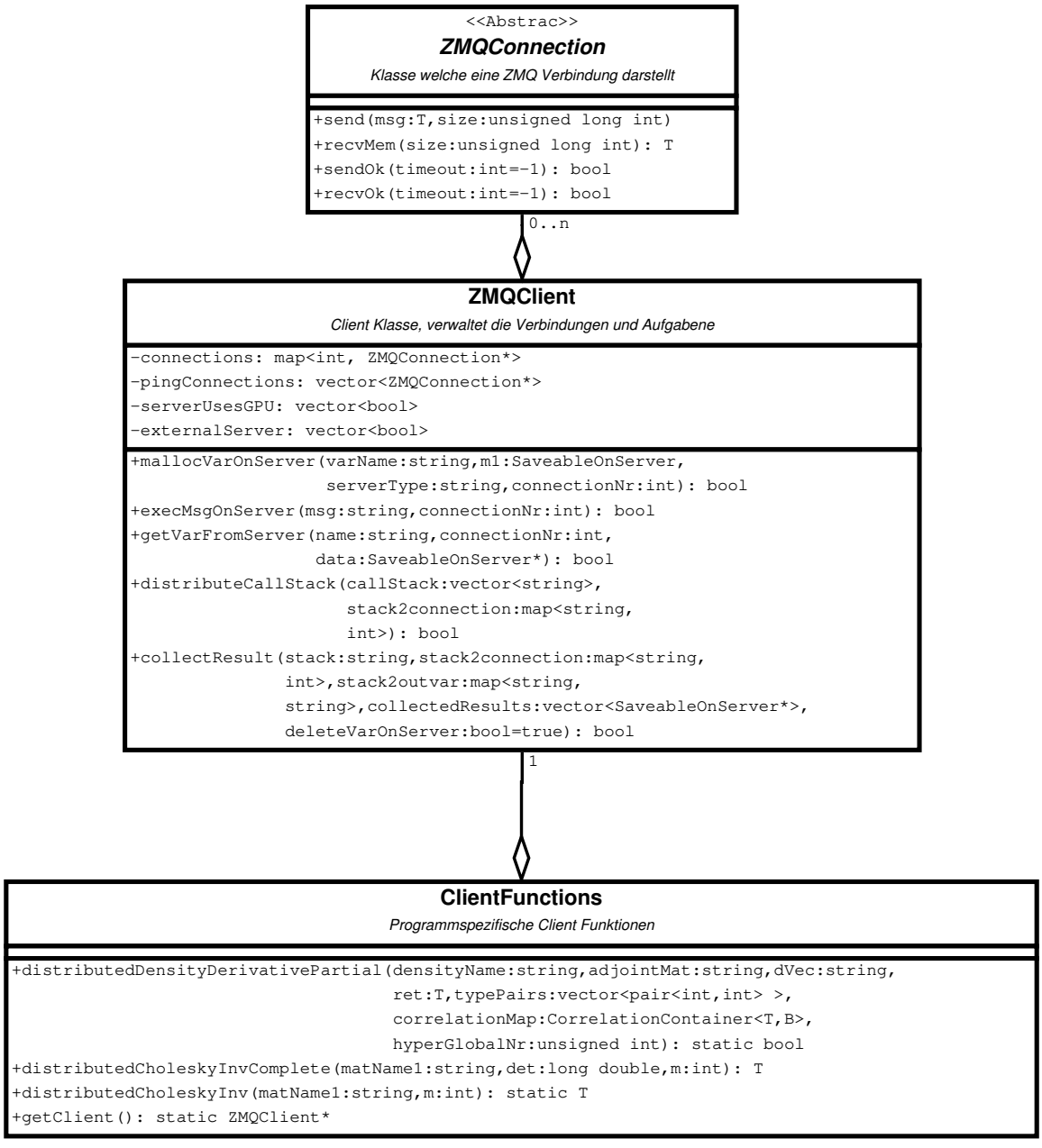


Abbildung A.8: UML Diagramm der Klassenstrukt einer Client Verbindung

A.3.10 UML Diagramm: Point

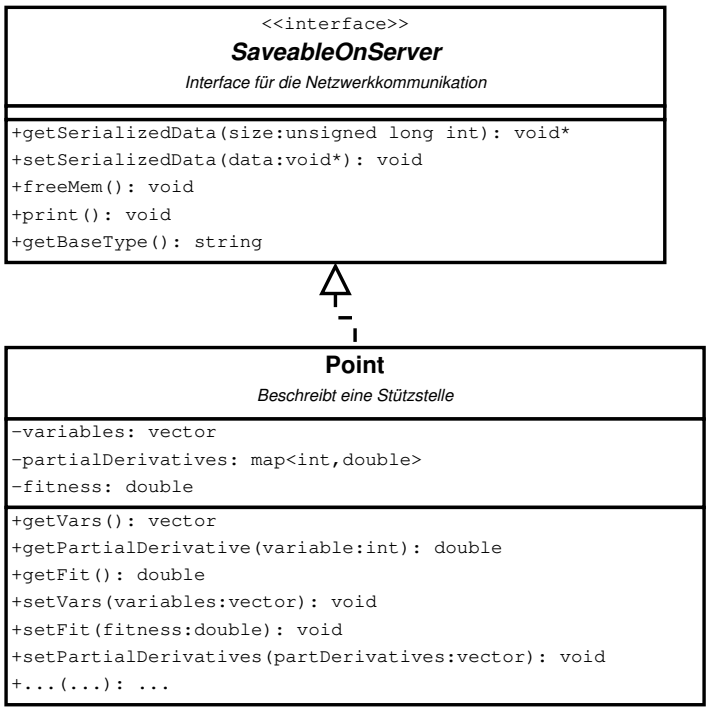


Abbildung A.9: Klassenstruktur der Klasse Point, welche eine Stützstelle beschreibt

A.3.11 UML Diagramm: CorrelationFunction

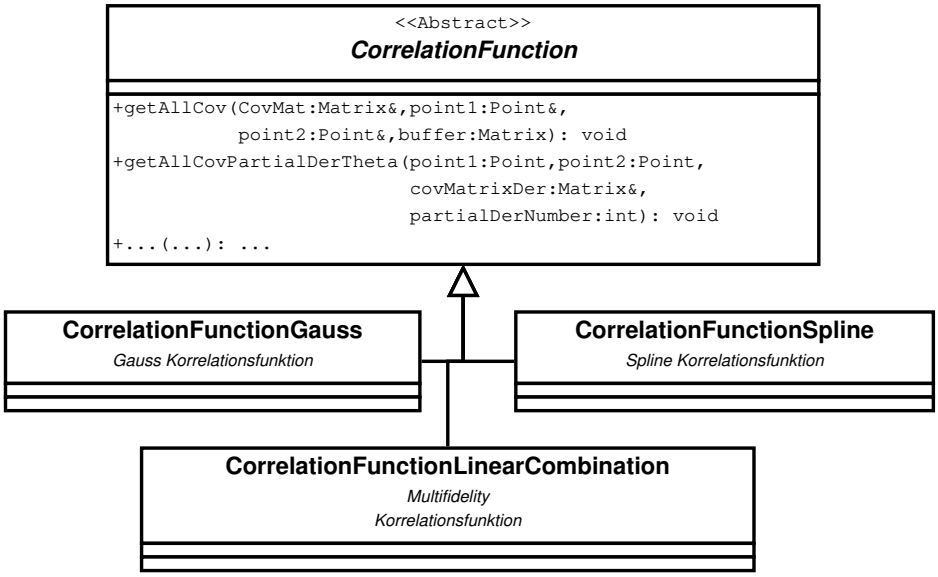


Abbildung A.10: Klassenstruktur der Kovarianzfunktionsklassen

A.3.12 UML Diagramm: DensityFunction

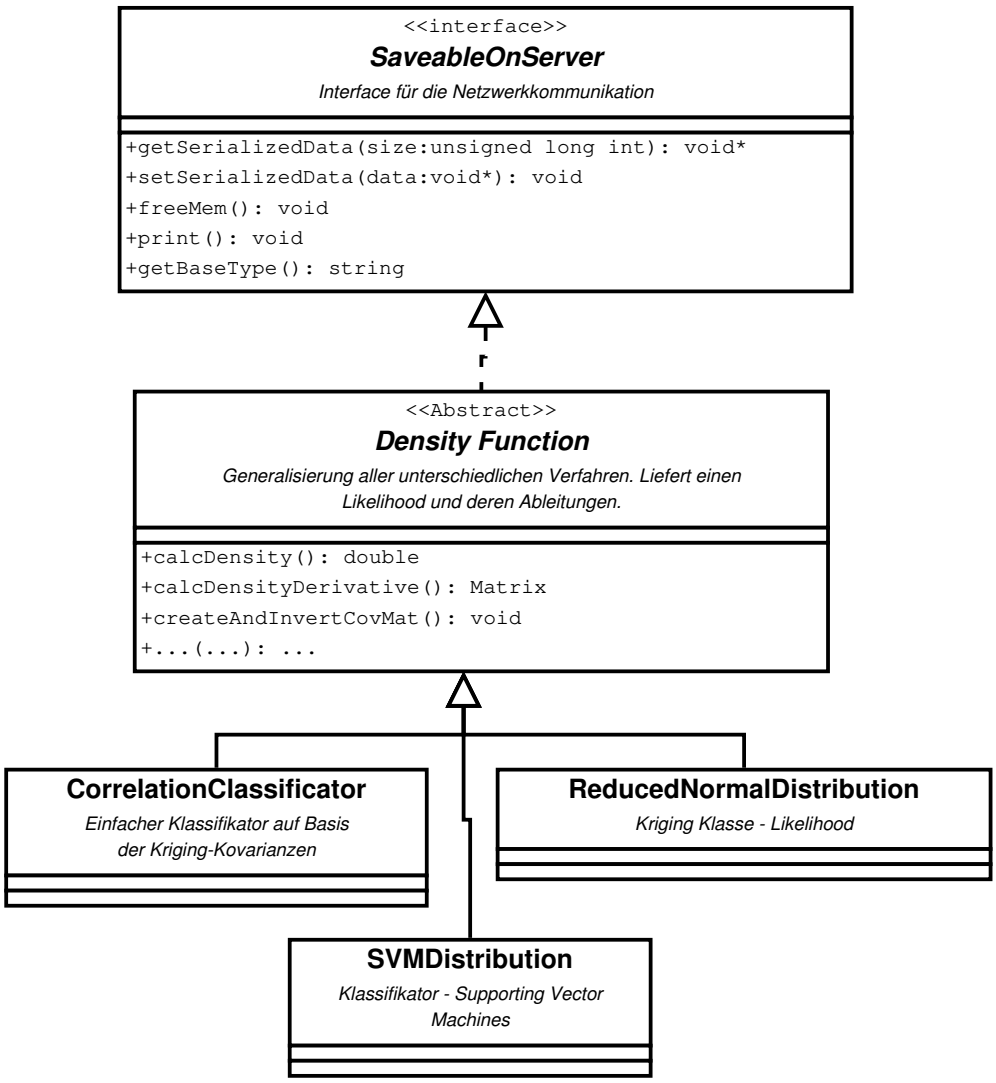


Abbildung A.11: Vereinfachtes UML Diagramm der verschiedenen Verfahren innerhalb der Kriging-Software und deren Anbindung an das Server Interface

A.3.13 UML Diagramm: MinimizerInterface

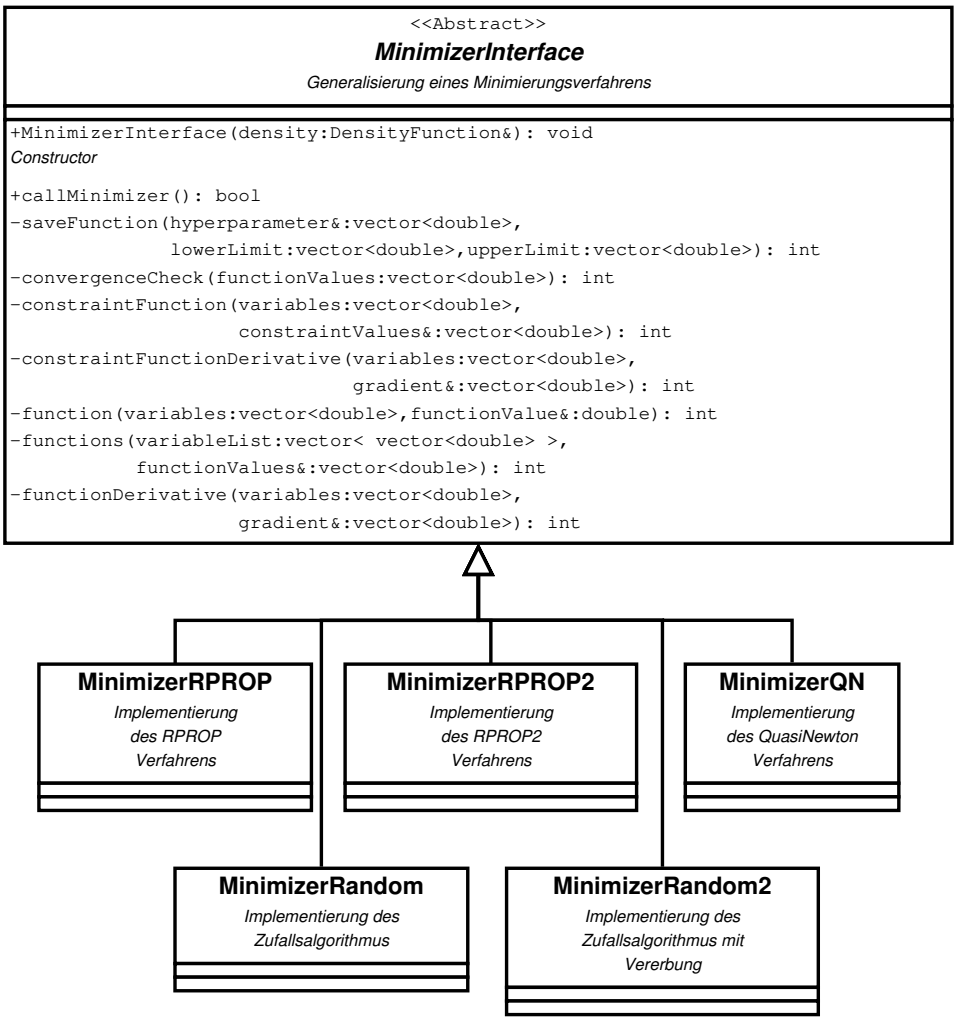


Abbildung A.12: UML-Diagramm der Klasse MinimizerInterface und deren Spezialisierungen

A.3.14 UML Diagramm: Matrix

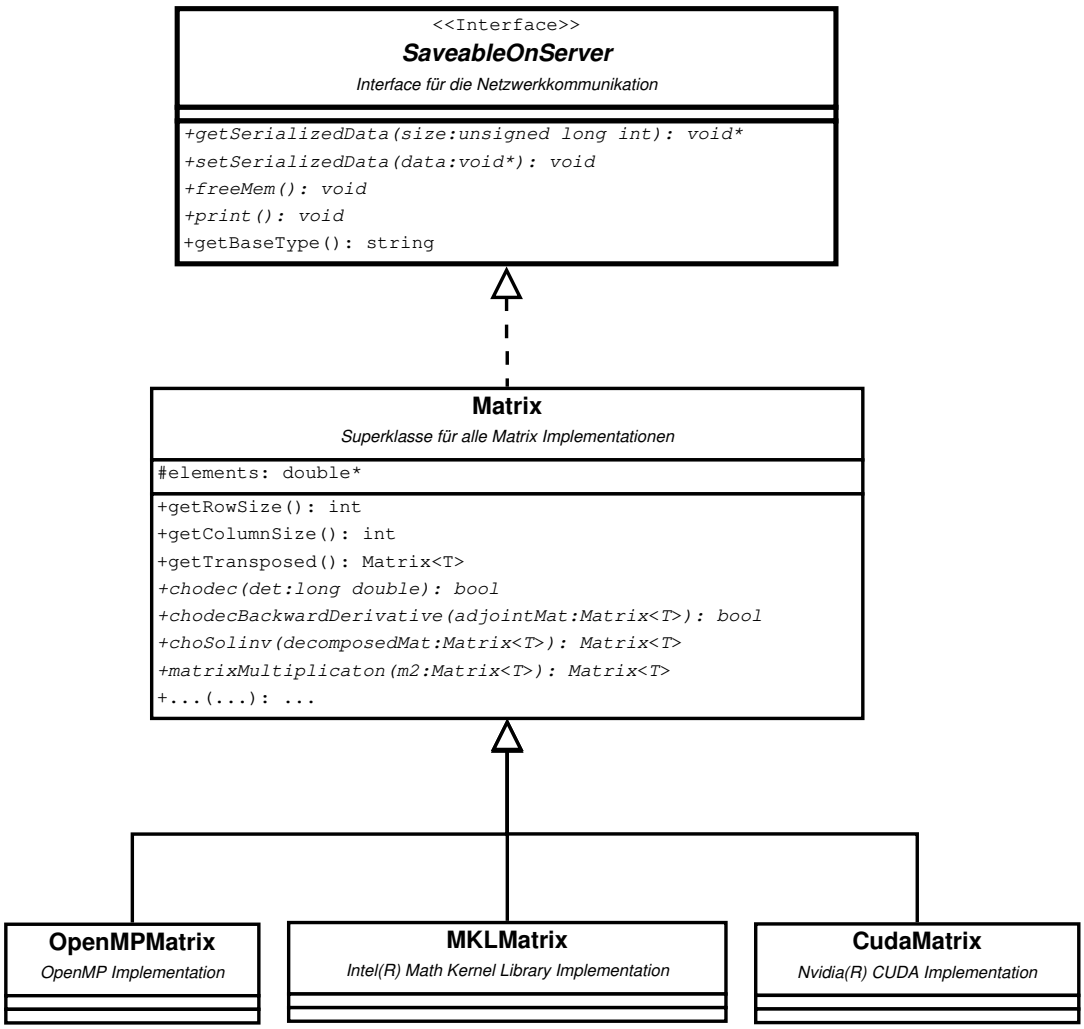


Abbildung A.13: UML Diagramm der Matrix Klassen und deren Anbindung an das Saveable-OnServer Interface

A.3.15 UML Diagramm: ZMQConnection, ZMQServer, MatrixServerFunctions

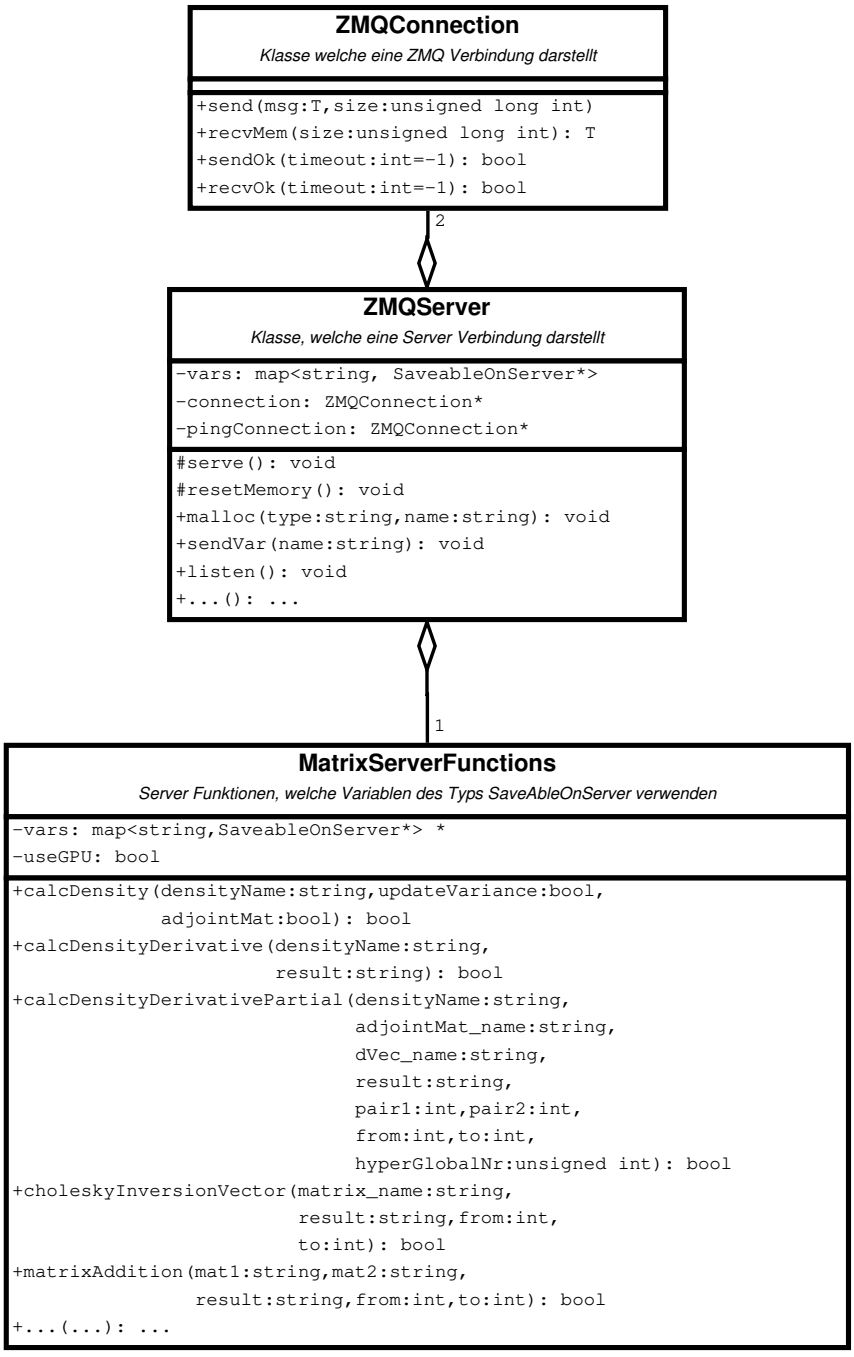


Abbildung A.14: UML Diagramm der Server-Klassen

A.3.16 Cholesky Zerlegung

Bei der Cholesky Zerlegung wird die symmetrische und positiv definite Matrix $\mathbf{R}$ in ein Produkt aus einer unteren Dreicksmatrix und deren Transponierten zerlegt:

$$\mathbf{L}\mathbf{L}^T = \mathbf{R}$$

Die Zerlegung kann nun verwendet werden, um durch Vor- und Rückwärtseinsetzen lineare Gleichungssysteme in der Form $\mathbf{R}\overrightarrow{x} = b$ zu lösen. Hierfür wird zuerst vorwärts eingesetzt:

$$\mathbf{L}\overrightarrow{y} = \overrightarrow{b}$$

und dann durch Rückwärtseinsetzen kann der gesuchte Vektor $\overrightarrow{x}$ erhalten werden:

$$\mathbf{L}^T\overrightarrow{x} = \overrightarrow{y}$$

$$\Longrightarrow \mathbf{R}\overrightarrow{x} = \mathbf{L}\mathbf{L}^T\overrightarrow{x} = \mathbf{L}\overrightarrow{y} = \overrightarrow{b}$$

ersetzt man nun $\overrightarrow{x}$ durch $\mathbf{R}^{-1}$ und $\overrightarrow{b}$ durch die Einheitsmatrix $\mathbf{E}$, kann man mit der Zerlegung die Inverse der Matrix $\mathbf{R}$ berechnen.

Nach diesem Schritt können die Vor- und Rückwärtssubstitutionen spaltenweise durchgeführt werden. Diese lassen sich sehr gut parallelisieren, da jede CPU einfach einen Vektor der Inversen berechnet.

Ein weiterer Vorteil der Zerlegung ist, dass die Determinante der Matrix $\mathbf{R}$ mit der Zerlegung einfach durch Multiplikation der Diagonalelemente der zerlegten Matrix gewonnen werden kann:

$$det\left(\mathbf{R}\right) = \prod_{i=1}^{n} L_{i,i}^2$$

Der verwendete Algorithmus ist in [Press et al., 2007, Gill, 2007] nochmals detaillierter beschrieben.

A.3.17 Likelihood: Inverse durch Gleichungssysteme ersetzen

Es werden die Hilfsvektoren $\vec{e}$ und $\vec{d}$ eingeführt:

$$\left(\vec{y}_s - \overrightarrow{F}\right) = \vec{e}$$
$$\mathbf{Cov}^{-1}\vec{e} = \vec{d}$$

Bei der Cholesky Zerlegung wird die Matrix $\mathbf{Cov}$ in ein Produkt aus einer unteren Dreiecksmatrix und deren Transponierten zerlegt. Die Dreiecksmatrix $\mathbf{L}$ gilt an dieser

Stelle als bekannt:

$$\mathbf{L}\mathbf{L}^T = \mathbf{Cov}$$

Daraus folgt:

$$\left(\mathbf{L}\mathbf{L}^\mathbf{T}\right)^{-1}\vec{e} = \vec{d}$$
$$\vec{e} = \vec{d}\mathbf{L}\mathbf{L}^\mathbf{T}$$

Führt man nun folgende Substitution ein:

$$\vec{d_{tmp}} = \vec{d}\mathbf{L}$$
$$\vec{e} = \vec{d_{tmp}}\mathbf{L}^\mathbf{T}$$

kann dieses Gleichungssystem durch eine einfache Rückwärtssubstitution $\vec{d}_{tmp}$ gelöst werden und schließlich kann der Vektor $\vec{d}$ über Vorwärtseinsetzen bestimmt werden:

$$\vec{d_{tmp}} = \vec{d}\mathbf{L}$$

A.3.18 Wirtschaftliche Betrachtung CPU/GPU

Intel CPUs

Name	GFlops	Preis	el. Leistung	Watt/GFlop	GFlops/€
Gold 6154	1209	5800€	200W	0.165	0.208
Platinum 8180	1523	9700€	205W	0.134	0.157
E7-8890v4	844.8	7600€	165W	0.195	0.111
E5-4669v4	774.4	7500€	135W	0.174	0.103
E5-2699v3	662.4	3830€	145W	0.219	0.173
E5-2698v3	588.8	3000€	135W	0.229	0.196
E5-2650v4	422.4	1200€	105W	0.249	0.352
E5-2650v3	368	1200€	105W	0.285	0.307
E5-2695v2	230.4	2100€	115W	0.5	0.11

Tabelle A.2: Relative Leistung aktueller Prozessoren (Stand 2018). Die theoretische Rechenleistung (mit GFlops bezeichnet) bezieht sich auf 64Bit Gleitkommawerte mit „fused multiply add (FMA)“.

Intel GPUs

Intel bietet mit dem „Xeon Phi“ eine Zusatzkarte für Supercomputer an. Im Vergleich zu Nvidia, besitzen diese einen x86 Befehlssatz, wodurch bestehende Software einfacher portiert werden kann. Intel gab allerdings bereits bekannt, das diese Linie 2019 eingestellt wird.

Name	GFlops	Preis	RAM	GB/s	el. Leistung	Watt/GFlop	GFlops/€
7290*	3456	6800€	16GB	490	245W	0.071	0.508
7250	3046	5400€	16GB	490	215W	0.0706	0.56
7120	1210	4000€	16GB	352	300W	0.248	0.30

Tabelle A.3: Relative Leistung Intel GPUs (Stand 2018). Die theoretische Rechenleistung (mit GFlops bezeichnet) bezieht sich auf 64Bit Gleitkommawerte mit „fused multiply add (FMA)“

Nvidia GPUs

Name	GFlops	Preis	RAM	GB/s	el. Leistung	Watt/GFlop	GFlops/€
V100 NVLink	7500	9500€	16GB	720	300W	0.04	0.79
V100 PCIe	5300	7300€	16GB	720	300W	0.057	0.72
P100 PCIe	4700	6800€	16GB	720	250W	0.053	0.69
K80	2912	5800€	24GB	2x288	300W	0.103	0.5
K40	1430	4400€	12GB	288	235W	0.164	0.325
K6000	1732	5000€	12GB	288	225W	0.17	0.286
GTX1080	277	800€	12GB	320	180W	0.65	0.347

Tabelle A.4: Relative Leistung Nvidia GPUs (Stand 2018). Die theoretische Rechenleistung (mit GFlops bezeichnet) bezieht sich auf 64Bit Gleitkommawerte mit „fused multiply add (FMA)“

A.3.19 GPU Resourcenverteilung bei parallelen Trainings

Trainings	Anteil GTX	Anteil K6000	Max. Time GTX	Max. Time K6000	Zeit gesamt
16	0	16	0s	156s	156s
	1	15	130s	143s	143s
	2	14	133s	132s	133s
	3	13	191s	128s	191s
12	0	12	0s	133s	133s
	1	11	113s	121s	121s
	2	10	122s	119s	122s
	3	9	187s	113s	187s
8	0	8	0s	96s	96s
	1	7	101s	85s	101s
	2	6	142s	83s	142s
	3	5	178s	82s	178s
4	0	4	0s	72s	72s
	1	3	98s	71s	98s
	2	2	123s	66s	123s

Tabelle A.5: Benchmark der besten Verteilung mehrerer Trainings mit 5000 Samples auf 2 verschiedene GPUs

Trainings	Anteil GTX	Anteil K6000	Max. Time GTX	Max. Time K6000	Zeit gesamt
16	0	16	0s	18s	18s
	1	15	11s	16s	16s
	2	14	11s	16s	16s
	3	13	12s	16s	16s
	6	10	13s	15.5s	15.5s
	10	6	14.8s	14.8s	14.8s
8	0	8	0s	10s	10s
	1	7	6s	8s	8s
	2	6	6s	7s	7s
	3	5	7s	7s	7s
4	0	4	0s	5s	5s
	1	3	4s	5s	5s
	2	2	5s	4s	5s

Tabelle A.6: Benchmark der besten Verteilung mehrerer Trainings mit 1000 Samples auf 2 verschiedene GPUs

Trainings	Anteil GTX	Anteil K6000	Max. Time GTX	Max. Time K6000	Zeit gesamt
16	0	16	0s	17.7s	17.7s
	1	15	10.4s	16.2s	16.2s
	2	14	10.2s	16s	16s
	3	13	10.2s	15.1s	15.1s
	9	7	10.4s	12.31s	12.31s
	12	5	10.2s	11.3s	11.3s
	14	2	10.2s	10.8s	10.8s
	15	1	10.2s	10.5s	10.5s
8	0	8	0s	7.7s	7.7s
	1	7	4.5s	6.5s	6.5s
	2	6	4.4s	5.9s	5.9s
	3	5	4.4s	5.7s	5.7s
	4	4	4.1s	5.2s	5.2s
	7	1	4.4s	4.4s	4.4s
4	0	4	0s	3.7s	3.7s
	1	3	2.4s	3.1s	3.1s
	2	2	2.2s	2.7s	2.7s
	1	3	2.4s	2.4s	2.4s

Tabelle A.7: Benchmark der besten Verteilung mehrerer Trainings mit 1000 Samples auf 2 verschiedenen GPUs

A.3.20 Benchmark Rückwärtsdifferenzierung Likelihood

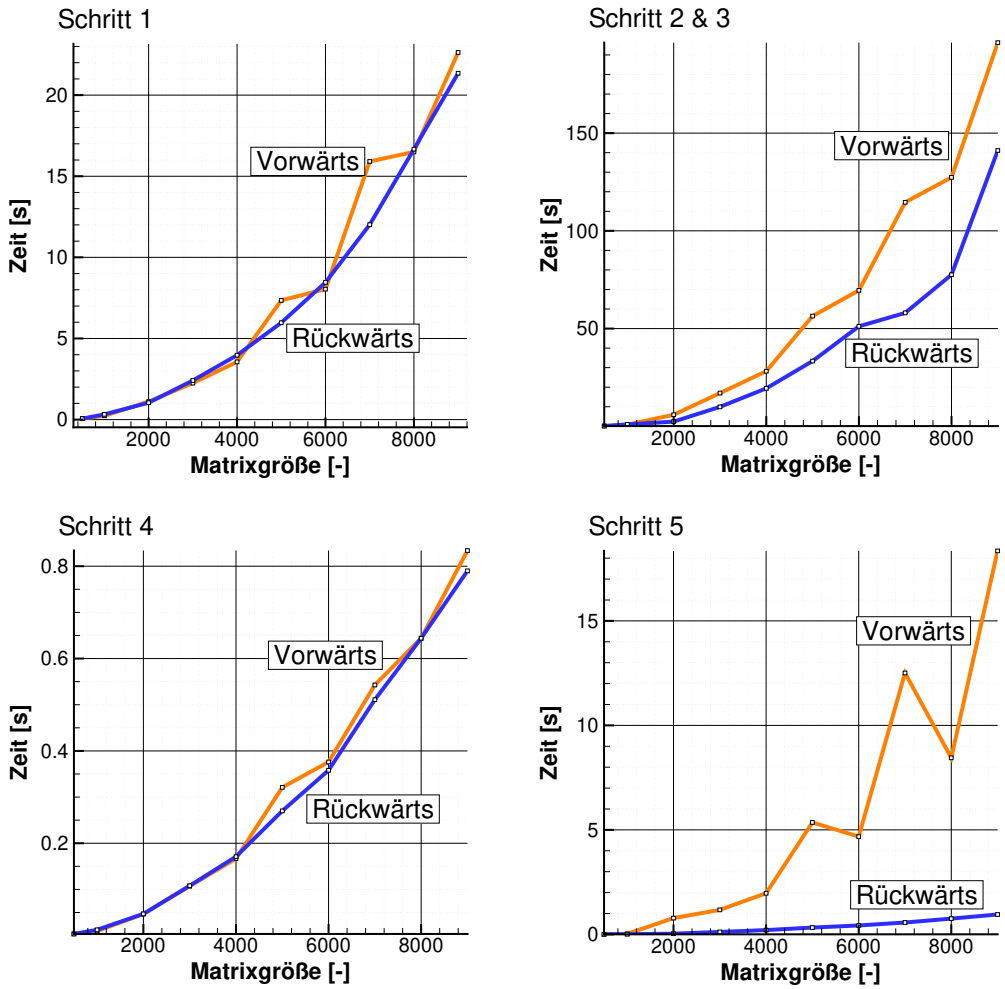


Abbildung A.15: Vergleich der Zeiten der Berechnung der partiellen Ableitungen des Likelihood-Terms

A.4 Kapitel 6

A.4.1 Testfunktion ZDT3

Hier dargestellt ist eine oft verwendete Testfunktion namens ZDT3 (siehe [Zitzler et al., 2000]). Diese Funktion besitzt zwei Zielfunktionen und zeichnet sich durch eine nicht zusammenhängende Paretofront aus. Die Testfunktionen $f_1(\vec{x})$ und $f_2(\vec{x})$ mit der Variable $\vec{x} \in \mathbb{R}^k$ wird wie folgt berechnet:

$$f_1(\vec{x}) = x_0 \tag{A.11}$$

$$f_2(\vec{x}) = h(\vec{x}) * g(\vec{x}) \tag{A.12}$$

$$g\left(\vec{x}\right) = \left(1 + \frac{9}{k-1}\sum_{i=2}^{k}\left(x_i - 0.5\right)^2\right)$$

$$h\left(\vec{x}\right) = \left(2 - \sqrt{\frac{x_1}{g}}\right) - \frac{x_1}{g}\sin\left(10\pi x_1\right)$$

In Abbildung A.16 ist die nicht triviale zweite Zielfunktion im Fall $\vec{x} \in \mathbb{R}^k, k = 2$ dargestellt. Bei der Funktion handelt es sich um einen Paraboloid mit überlagerter Sinusschwingung. Diese Funktion hat einige lokale Minima. Weiterhin kann diese auf beliebig viele Dimensionen angewandt werden und eignet sich daher für analytische Optimierungs-Tests.

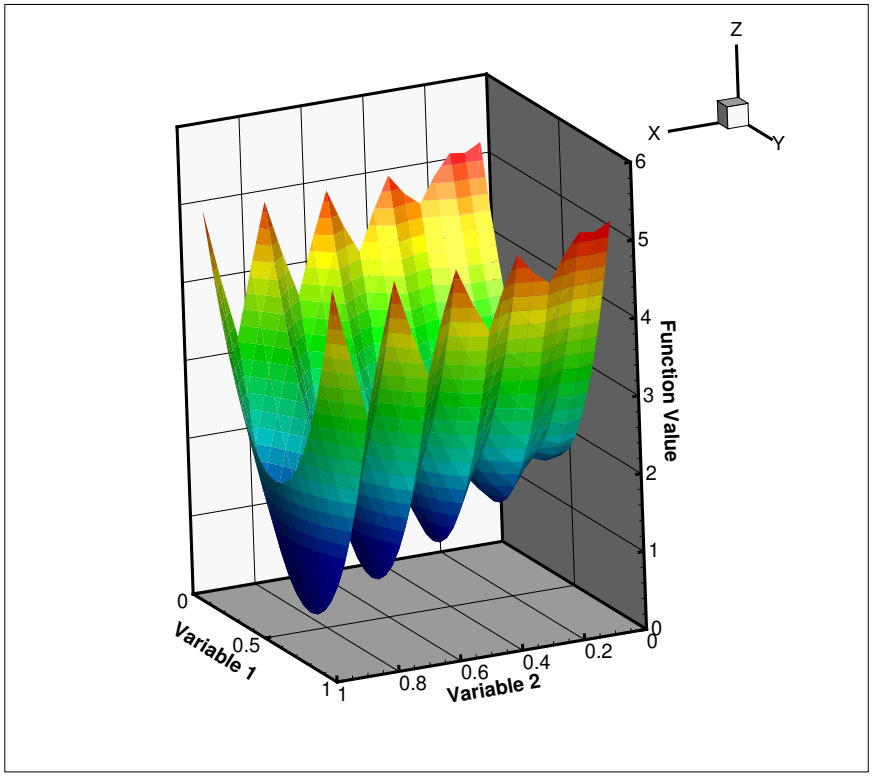


Abbildung A.16: Plot der Testfunktion mit zwei Variablen

A.4.2 Zusätzliche Abbildungen der Fanstufenoptimierung

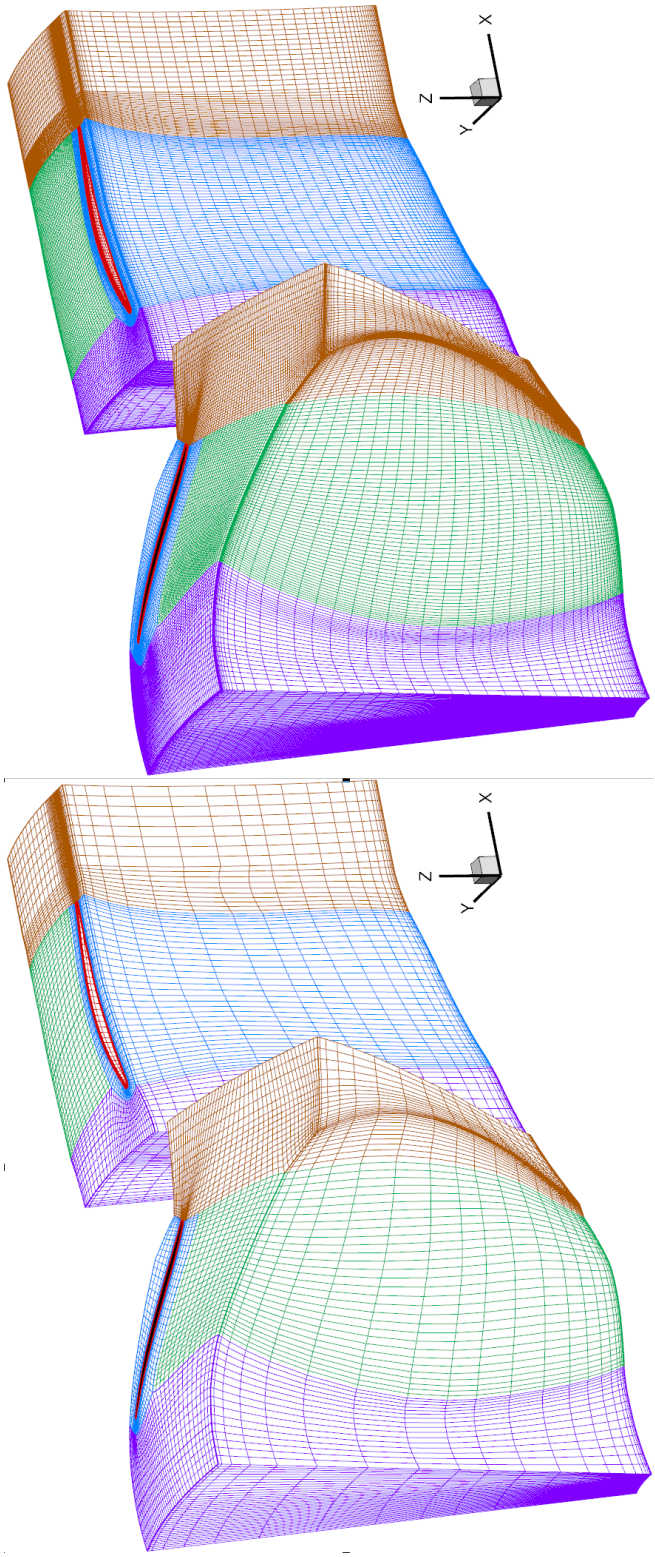


Abbildung A.17: Exemplarisches CFD-Netz der hohen Güte (linkes Bild) und der niedrigen Güte (rechtes Bild) Quelle: [Reimer, 2016]

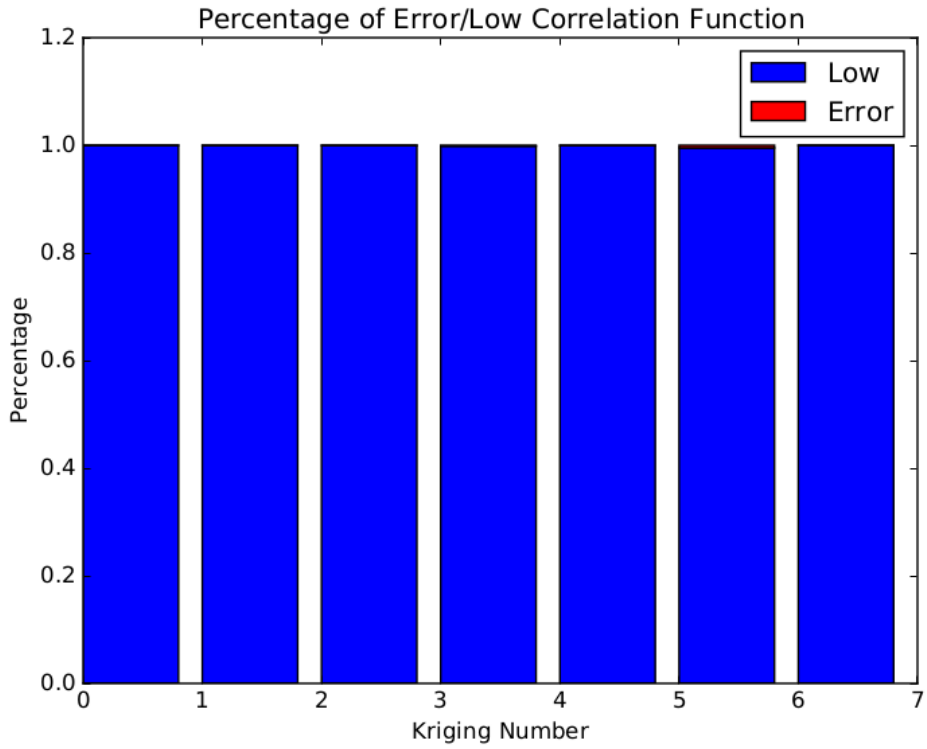


Abbildung A.18: Anteile der Kovarianzfunktionen verschiedenen Co-Kriging Modellen der Fanstufen-Optimierung (siehe Kapitel 6.2). Auszug der Kriging-Analysesoftware

Lebenslauf

Persönliche Daten

Name: Andreas Schmitz

Geburtsdatum: 31.10.1982

Geburtsort: Waldbröl

Berufserfahrung

Seit 2009	**Wissenschaftlicher Mitarbeiter** Deutsches Zentrum für Luft- und Raumfahrt, Institut für Antriebstechnik, Abteilung Fan und Verdichter, Köln

Ausbildung

2009-2014	Fernuniversität Hagen Studienrichtung: Computer Science Abschluss: Master of Science
2004-2009	FH-Köln Fakultät für Informatik und Ingenieurwissenschaften Studienrichtung: Allgemeiner Maschinenbau Abschluss: Diplom Ingenieur (FH)
2004-2007	FH-Köln Fakultät für Informatik und Ingenieurwissenschaften Abschluss: B. Sc. in Mechanical Engineering with Computer Science
2000-2003	Berufskolleg Oberberg Abschluss: Staatlich geprüfter informationstechnischer Assistent
2000-2003	Berufskolleg Oberberg Abschluss: Fachhochschulreife