TU BERLIN
INSTITUT FÜR LUFT- UND RAUMFAHRT
Fachgebiet für Luftfahrzeugbau und Leichtbau

Masterarbeit

Surrogate Models für den Flugzeugvorentwurf

Verfasser: Juri Bieler
Matr.-Nr: 339117

Betreuerin: MSc. Claire Jacob
Betreuender Professor: Prof. Dr.-Ing. Andreas Bardenhagen

Abgabe: Berlin, 6.11.2018

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig, sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle in dieser Arbeit dargestellten Abbildungen und Tabellen sind eigene Darstellungen des Autors, sofern sie nicht gesondert gekennzeichnet.

Berlin, den 6. November 2018

Juri Bieler

Kurzzusammenfassung

Durch Voranschreiten der technischen Möglichkeiten werden zunehmend die auf Statistiken beruhenden Verfahren des Flugzeugvorentwurfs durch hochauflösende FEM- und CFD-Methoden ersetzt. Dabei ist trotz leistungsstarker Computer die Rechenzeit noch immer ein großes Problem. Oft wird iterativ mit MDAO-Software optimiert, was viele Berechnungen der hochauflösenden Lösung verlangt. In dieser Arbeit wird überprüft, ob Surrogate Modelle dieses Problem lösen und mit nur einer begrenzten Zahl von bekannten Lösungen ein genaues Optimum finden können. Daher wurden verschiedene Methoden, die zum Bilden von Surrogate Modellen verwendet werden können, mathematisch beschrieben und an einem Beispielproblem angewandt. In diesem Beispiel werden die Rippenzahl und die Blechdicke eines Flügelkastens optimiert, sodass das Strukturgewicht minimal wird. Dabei darf die maximal zulässigen Materialbelastungen nicht überschritten werden. Um einen Vergleich zu haben, wird das Optimum mit bereits etablierten MDAO-Verfahren und mit Surrogate-Methoden gesucht. Für die Surrogate Modelle werden der Polynom-Ansatz, Radial Basis Funktionen und Kriging getestet. Es stellte sich heraus, dass Surrogate-Methoden nicht nur einfacher in der Handhabung sind, sondern auch mit großer Präzision und sehr viel weniger Rechenaufwand ein Optimum finden können.

Abstract

With advancing technical solutions, the simple statistical methods of preliminary aircraft design get replaced by high-fidelity calculations like FEM or CFD. These techniques need even with modern computers long calculation time. Often MDAO-software is used to optimize problems based on these high-fidelity calculations, which requires many iterations. In this thesis the potential of surrogate models is investigated, to see if they can find a optimum by using less evaluations than MDAO. Therefore, some of the common surrogate-techniques are mathematical discussed. Moreover, a sample problem is created to test the algorithms and get experience in their use. A simple wing structure gets optimized for its minimum weight by modifying the number of Ribs and the shell thickness while making sure that the maximum allowed stress is obtained using an Abaqus FEM-calculation. To make a comparison, this problem gets solved by MDAO and by surrogate-methods. As surrogate-techniques, a polynomial approach, radial basis functions and Kriging are tested. The conclusion is, that surrogate models are compared to MDAO easy to use and still precise in finding an optimum while requiring less evaluations of the high-fidelity-function.

Technische Universität Berlin

FAKULTÄT V

VERKEHRS- UND MASCHINENSYSTEME

Institut für
Luft- und Raumfahrt

Fachgebiet
Luftfahrzeugbau und Leichtbau

Prof. Dr.-Ing. A. Bardenhagen

Berlin, den 28.05.2018

Masterarbeit

für
Herrn Juri Bieler, Matr.-Nr. 339117

Surrogate Models für den Flugzeugvorentwurf

Erläuterungen zum Thema

Durch steigende Anforderungen an heutige Luftfahrzeugsysteme, bezüglich Lärm-, Emission- und Kostenreduktion, werden sich zukünftige Konfigurationen signifikant von den heute bekannten Flugzeugtypen unterscheiden. Zu den bekannten Flugzeugtypen zählen die sogenannten Drachenflugzeugkonfigurationen oder Tube-Wing-Aircraft. Dazu trägt auch wesentlich der zu erwartende Einsatz elektrischer Systeme einschließlich des elektrischen Antriebssystems bei. Um diese neuartigen Konfigurationen flexibel modellieren, analysieren und optimieren zu können, sind hochwertige numerische Simulationsverfahren bereits im Flugzeugvorentwurf notwendig.

Im Rahmen des am Fachgebiets stattfinden Forschungsprojects *Turbo electRic Aircraft Design Environment* (TRADE) werden Verkehrsflugzeugkonfigurationen mit hybriden Antrieben untersucht. Dort wird ein hochwertiges numerisches Simulationsprogramm entwickelt und für die Untersuchung genutzt. Um in Zukunft zeitnah erste Aussagen über die Güte des Entwurfs treffen zu können, ist es notwendig schnelle Analysen durchführen zu können. Dafür eignen sich die hochwertigen numerischen Simulationsverfahren oft nicht, da sie zu viel Zeit benötigen. Auf Datenbasis dieser ist es jedoch möglich Ersatzmodelle (Surrogate Modells) zu bilden und damit eine schnelle Auswertung zu ermöglichen. Gleichzeitig bieten diese Surrogate Modells die Möglichkeit, andere Konfigurationen, die nicht den klassischen entsprechen, zu untersuchen. Die verschiedenen Plattformen wie OpenMDAO des NASA Glenn Research Centers haben die Möglichkeit implementiert Surrogate Modells zu bilden und gleichzeitig mit diesen Optimierungen hinsichtlich bestimmter Parameter durchzuführen.

> Seite 1/2

Technische Universität Berlin

Ziel der Arbeit

Es soll untersucht werden, ob es möglich ist, Surrogate Modells für den Flugzeugvorentwurf zu bilden. Dabei soll darauf geachtet werden, dass die Datenbasis der Modelle zu einem späteren Zeitpunkt beliebig geändert werden kann.

Im Einzelnen sind folgende Punkte zu bearbeiten:

1. Literaturrecherche zu Surrogate Models und deren Anwendung im Bereich der Ingenieurswissenschaften speziell dem Luftfahrzeugentwurf.
2. Literaturrecherche zu den Prinzipien von MDAO (Multidisciplinary Design Analysis and Optimization).
3. Bilden von einfachen Surrogate Models auf Basis von erweiterten Methoden aus dem Flugzeugentwurf unter Berücksichtigung, dass später hochauflösende Modelle die Datenbasis für die Surrogate Modells bilden sollen.
4. Untersuchung ob die Anzahl der Parameter und deren Variationsbereich mittels Design of Experiments bestimmt werden kann, um den Datenraum für die Rechnungen mit dem genauen Model abzustecken.
5. Begründete und fundierte Auswahl des Optimierungszieles hinsichtlich Anwendbarkeit im Luftfahrzeugentwurf.
6. Analyse verschiedener Luftfahrzeugentwürfe mittels des Surrogate Modells hinsichtlich des Optimierungszieles

Die Arbeit wird am Fachgebiet durchgeführt und von Claire Jacob, MSc. und dem Unterzeichnenden betreut. Nach Abschluss sind die Ergebnisse der Arbeit auf einem Poster oder in einer 20-minütigen Kurzpräsentation im Rahmen des Luftfahrzeugbau Colloquiums am ILR vorzustellen.

Prof. Dr.-Ing. Andreas Bardenhagen

Literaturempfehlungen:

[1] E. Torenbeek, *Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes*. 2013.

[2] J. Gray, K. Moore, and B. Naylor, "OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization," *13th AIAA/ISSMO Multidiscip. Anal. Optim. Conf.*, pp. 1–12, 2010.

[3] A. I. J. Forrester, A. Sbester, and A. J. Keane, *Engineering Design via Surrogate Modelling*. Chichester, UK: John Wiley & Sons, Ltd, 2008.

> Seite 2/2

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Nomenklatur

Lateinische Symbole

Symbol	Bedeutung	Einheit
a	Einstellparameter (bei RBF)	-
b	Basis	-
C	Korrelations-Matrix (Kriging)	-
f	wahre, genaue Funktion	-
$\widehat{f}$	nachgebildete Funktion (Surrogate)	-
i	Zählvariable (auch $i1$, $i2$ bei zwei Dimensionen)	-
j	Zählvariable (zählt unabhängig von i)	-
k	Anzahl der Eingangsgrößen einer Funktion	-
l	Anzahl der Levels in DoE	-
L	Likelihood - Wahrscheinlichkeit (Kriging)	-
n	Anzahl der Stützstellen	-
o	Grad eines Polynom (Polynom Ansatz)	-
$\emptyset$	Durchschnitt	-
p	Exponentenparameter für den Teilfaktorplan (Doe)	-
$\vec{p}$	Vektor von Kriging Einstell-Exponenten	-
r	Radius	-
V	Vandermonde-Matrix	-
$\vec{w}$	Vektor aus Gewichtungen für die verschiedenen Eingangsgrößen bzw. Terme einer Funktion	-
x	Eingangswert bzw. Koordinate in Spannweitenrichtung	-, m
x^*	Eingangswert an bekannter Stützstelle	-
$\vec{x}$	Vektor aus Eingangswerten für einen Punkt	-
X	Matrix aus Vektoren aus Eingangswerten für einen Punkt	-
y	Lösungswert	-
y^*	Lösungswert an bekannter Stützstelle	-
$\vec{y}^*$	Vektor aus Lösungswerten an verschiedenen Punkten	-
z	Koordinate senkrecht auf der Tragfläche, nach unten	m

Griechische Symbole

Symbol	Bedeutung	Einheit
ϕ	Radial-Basis-Funktion	-
Ψ	Matrix aus Lösungen von RBF an Stützstelle ausgewertet	-
μ	Offset Wert für Kriging Approximation	-
$\vec{\theta}$	Vektor von Kriging Einstell-Faktoren	-

Abkürzungen

Abkürzung	Bedeutung
ALPSO	Augmented Lagrangian Particle Swarm Optimizer
API	Application Programming Interface
CCD	Central Composite Design
CFD	Computational Fluid Dynamics
cgx	CalculiX GraphiX
CSV	Comma-Separated Values
DOC	Direct Operating Costs
DoE	Design of Experiment
FEM	Finite Element Method
GPML	Gaussian Process for Machine Learning
MAE	Maximum Absolute Error
MDAO	Multidisciplinary Analysis and Optimization
MDO	Multidisciplinary Design Optimization
MTOW	Maximum TakeOff Weight
PRESS	Prediction Sum of Squares
RAE	Relative Absolute Error
RBF	radiale Basis-Funktion
RMSE	Root Mean Square Error
SLP	Sequential Linear Programming
SLSQP	Sequential Least Squares Programming
TW	Triebwerk

Kapitel 1

Einleitung

Der klassische Flugzeugvorentwurf wurde in der Vergangenheit anhand von auf Statistik beruhenden Methoden vorgenommen. Diese haben den Vorteil, dass sie auf sehr einfachen Rechnungen basieren und somit wenig Rechenleistung benötigen. Nachteilig an diesen Methoden ist jedoch, dass sie lediglich Flugzeugkonfigurationen zulassen, die sehr stark an der Masse der existierenden Flugzeugen angelehnt sind, da auch nur für diese Statistiken in ausreichendem Maß zur Verfügung stehen. Um gänzlich neue Konzepte zu untersuchen, werden deshalb oft schon im Vorentwurf äußerst genaue Berechnungen benötigt, wie sie beispielsweise mit CFD- oder FEM-Programmen erstellt werden können. Diese sind sehr rechenintensiv und können selbst mit moderner Computerhardware mehrere Tage für eine Berechnung in Anspruch nehmen. Um Optimierungen durchzuführen kommt häufig MDAO (Multidisciplinary Analysis and Optimization) zum Einsatz, was jedoch eine Vielzahl an Programmdurchläufen benötigt, um eine Lösung zu finden. An dieser Stelle können Surrogate Modelle Abhilfe schaffen. Sie sollen ermöglichen, mit einer geringen Anzahl an Evaluationen der hochauflösenden Berechnung den Verlauf des gesamtem Modells zu approximieren und somit leichter auf den optimalen Designpunkt zu schließen. Voraussetzung dafür ist, dass die Lösung der Zielfunktion kontinuierlich ist. Auf diese Weise wird es möglich durch Interpolation Ersatzmodelle zu generieren, mit denen in der späteren Auswertung ebenso leicht gerechnet werden kann wie mit den statistischen Gleichungen, die jedoch auf sehr viel mehr spezialisierten Verfahren beruhen.

Im Rahmen dieser Arbeit sollen folgende Fragen beantwortet werden:
Eignen sich Surrogate Modelle für den Flugzeugvorentwurf im Zusammenhang mit hochauflösenden Rechenverfahren?
Kann bei der Optimierung mit Surrogate Modellen, im Vergleich zu MDAO, Rechenzeit gespart werden?
Welche Probleme können bei der Umsetzung von Surrogate-Methoden auftreten und wie kann

mit ihnen umgegangen werden?
Um die Fragen zu klären, werden in dieser Arbeit zunächst die Grundlagen zu MDAO und DoE (Design of Experiment), was zur Versuchsvorbereitung genutzt wird, besprochen. Zudem werden beispielhaft einige ausgewählte Methoden, die zum Bilden von Surrogate Modellen verwendet werden können, mathematisch beschrieben und implementiert. Diese setzen sich zusammen aus Algorithmen zum Generieren von Versuchsplänen, dem Erstellen der eigentlich Surrogate Modelle und deren Training, sowie der Validierung und Optimierung. Dabei wird die Programmiersprache Python genutzt, da diese viele Bibliotheken bereit stellt die Matrizen-Rechnungen ermöglichen, aber auch ganze Optimierungen übernehmen.
Es wird gezeigt, was der aktuelle Stand der Technik zum Thema Surrogate Modelle im Flugzeugentwurf ist, indem einige Beispiele aus der Literatur vorgestellt werden. Um die Verfahren selbst zu testen wird ein beispielhaftes Problem konstruiert, bei dem die Rippenanzahl und die Blechdicke eines Flügelkastens so optimiert werden, dass das Gewicht minimal ist und die maximal zulässigen Spannungen im Material nicht überschritten werden. Das Problem wird in Python beschrieben, während Abaqus die Belastung der Struktur ermittelt. Parallel wird mit einem MDAO- und einem Surrogate-basierten Ansatz das Optimum gesucht. Für die Lösung mit Hilfe der Surrogate Modelle werden die verschiedenen Versuchsplan- und Surrogate-Methoden angewandt. Die dabei gewonnenen Erkenntnisse werden ausgewertet und ein Vergleich der Methoden auf Anwendbarkeit, Zuverlässigkeit und Aufwand vorgenommen.

Kapitel 2

Grundlagen

In diesem Kapitel werden grundlegende Methoden der Optimierung wie sie im Flugzeugvorentwurf Anwendung finden, besprochen. Dabei wird zunächst das bereits etablierte Verfahren MDAO besprochen. Zusätzlich wird auf DoE eingegangen, welches Grundlage für alle Optimierungsverfahren ist. Im Anschluss werden verschiedene Methoden, die für das Bilden von Surrogate Modellen nötig sind, vorgestellt und mathematisch beschrieben. Da die Literatur zu diesen Disziplinen fast ausschließlich in englischer Sprache existiert soll hier, um den Wiedererkennungswert zu erhöhen, auf ein genaues Übersetzten der Verfahrensnamen weitestgehend verzichtet werden.

2.1 Aktueller Stand

In diesem Kapitel wird der aktuelle Stand der Technik des Flugzeugvorentwurfs vorgestellt. In der Vergangenheit wurde mit simplen, statistischen Methoden, iterativ vorgegangen, um ein Flugzeug grob auszulegen. Zunehmend wird dieses Verfahren, wie es von Torenbeek ([46] und [47]) bekannt ist, durch hochauflösende CFD und FEM Analysen ersetzt. Bei diesen Rechenzeit intensiven Methoden ist es wichtig, die Anzahl der Versuche so gering wie möglich zu halten. Hierzu sollte zunächst betrachtet werden, welche Eingangsgrößen der hochauflösenden Funktion tatsächlich eine Rolle spielen und in welchen Wertebereichen diese variieren. Dabei kann DoE helfen, was im ersten Teil dieses Kapitel genauer beschrieben wird. Es gibt zudem spezielle Optimierungsverfahren, die auf schnellem Weg die optimale Lösung durch aufeinander aufbauende Versuche finden. Hierbei müssen in der Regel mehrere Eingangsgrößen variiert und Randbedingungen beachtet werden. Ein Ansatz, der eine Lösung dafür bietet und hier exemplarisch genauer beschrieben wird, ist Multidisciplinary Analysis and Optimization (MDAO).

2.1.1 Design of Experiment

Design of Experiment (DoE) beschreibt im Allgemeinen den Vorgang der Versuchsplanung. Es findet in der Literatur jedoch keine klare Definition. Oft ist, besonders im Zusammenhang mit Surrogate Modellen, der Term DoE als Synonym für Versuchsplan verwendet. Beispiele hierfür sind [15, S. 52], [19] und [16]. In der deutschen Literatur hingegen beschreibt DoE in der Regel den gesamten Prozess vom Versuchsplan bis hin zum Bilden der Approximation (vgl. [43] und [21]). In spezieller DoE Literatur wie die von J. Antony [4] und D. Montgomery [29] sind hingegen die Analyse der Eingänge eines Experiments gemeint (was in dieser Arbeit als Definition übernommen wird). Hierbei ist besonders wichtig herauszufinden, ob es Eingangsgrößen gibt, die keinen oder nur wenig Einfluss auf das Ergebnis der Zielfunktion haben. Diese können dann während des Experiments konstant gehalten werden und das Problem reduziert sich in seiner Komplexität. Dadurch kann sich die Rechenzeit, die für eine Optimierung benötigt wird, drastisch reduzieren, da die Größe des Designraums (und somit auch der Rechenaufwand) mit der Anzahl der Eingänge in der Potenz wächst. Um eine DoE-Analyse durchzuführen müssen zunächst einige Punkte der zu untersuchenden Funktion bekannt sein. Dabei wird ein Versuchsplan auf sogenannten Levels erstellt. Jeder Eingang wird in Levels aufgeteilt. Diese repräsentieren konkrete Werte für den jeweiligen Eingang. Im einfachsten Fall gibt es zwei Levels, ein tiefes und ein hohes. Sinnvoll ist es dabei die vermuteten Grenzen des Wertebereichs des jeweiligen Eingangs als Levels zu nutzen. Die Verwendung von lediglich zwei Levels setzt die Annahme voraus, dass die Ausgangsfunktion etwa linear (oder zumindest streng monoton) im untersuchten Bereich verläuft. Bei der Durchführung der DoE-Analyse muss das Experiment (bzw. die hochauflösende Funktion) nun an den Levels der verschiedenen Eingängen ausgeführt werden. Welche Kombinationen von Eingangswerten auszuführen sind, wird von einem Vollfaktor- oder Teilfaktorplan festgelegt. [43]

Vollfaktorplan

Ein Vollfaktorplan (full factorial design) besteht aus allen möglichen Kombinationen von Level aller Eingänge. Die Summe der benötigten Experimente ergibt sich somit bei k Eingängen mit je l Level zu l^k. Der Vollfaktorplan eignet sich besonders, wenn nicht mehr als vier Eingänge existieren und wenig über deren Wirkung und Interaktion bekannt ist. [4, S. 70]
Um einen Vollfaktorplan zu erstellen, welcher alle möglichen Kombinationen von Level und Eingängen berücksichtigt, hilft ein Zahlensystem zur Basis l. Wird die maximale Anzahl der möglichen Kombinationen in diesem Zahlensystem dargestellt, so hat diese Zahl genau k Stellen. Jede Stelle kann als Level-Index für einen Eingang interpretiert werden. Wird nun von Null bis l^k-1 hoch gezählt, sind alle Kombinationen enthalten. Ein Beispiel für zwei Eingänge (A und B) und drei Levels (gekennzeichnet durch $-$, 0 und $+$) ist in Tabelle 2.1 dargestellt.

Dezimal	Ternärsystem (Basis 3)	A	B
0	00	−	−
1	01	−	0
2	02	−	+
3	10	0	−
4	11	0	0
5	12	0	+
6	20	+	−
7	21	+	0
8	22	+	+

Tabelle 2.1: Vollfaktorplan mit zwei Eingängen (A und B) auf drei Levels

Teilfaktorplan

Steht nicht genug Zeit zur Verfügung, um einen Vollfaktorplan auszuführen oder ist die Zahl der Eingänge sehr hoch, kann ein Teilfaktorplan (fractional factorial design) angewandt werden. Hiermit werden nur die offensichtliche Effekte untersucht und Effekte höherer Ordnung (dritte Ordnung und höher) vernachlässigt. Die Anzahl der benötigten Experimente für k Eingänge auf l Levels ist l^{k-p}. Es macht jedoch Sinn hier $l=2$ zu wählen, da wie zuvor erwähnt, besonders lineare Veränderungen betrachtet werden. p beschreibt die Größe des Anteils (fraction), der von dem Vollfaktorplan untersucht werden soll. Dieser kann prozentual als $\frac{1}{2}^p$ ausgedrückt werden. Soll also beispielsweise 25% des Vollfaktorplans genutzt werden, so muss $p=2$ sein ($\frac{1}{2}^2 = 0.25$). Für jedes p gibt es verschiedene Techniken, um den Versuchsplan zu erstellen, die von J. Antony in [4, S. 77] genauer beschrieben werden. Die im späteren Verlauf dieser Arbeit beschriebenen Versuchsplan-Verfahren für das Bilden von Surrogate Modellen zählen ebenfalls zu den Teilfaktorplänen, eignen sich jedoch durch ihre Verstreuung der Punkte weniger gut für DoE. [43]

Auswertung

Mit den oben beschriebenen Versuchsplänen kann eine hochauflösende Funktion ausgewertet werden. Von den Ergebnissen lassen sich folgende Informationen ableiten:

- Der Einfluss der einzelnen Eingänge auf die Funktion kann abgeschätzt werden.
- Mögliche Interaktion der Eingänge untereinander werden erkannt.
- Die Wertebereiche der Eingänge können überprüft werden.
- Bei mehr-Level Design kann abgeschätzt werden, ob ein Eingang die Funktion linear oder kurvenförmig beeinflusst.

- Es kann grob überprüft werden, ob das Modell (die Funktion) dem erwarteten Verhalten entspricht (Evaluation auf Erfahrung und Logik basierend).

[4]

weiterführende Literatur

In [24] schreibt R. Jin über den Gebrauch von DoE im Zusammenhang mit Surrogate Modellen. Ihm zufolge dient DoE vor allem dazu, herauszufinden, ob es Eingänge gibt, die komplett vernachlässigt werden können.
S. Misra beschreibt in [28], wie der Grad einer Funktion mittels DoE bestimmt werden kann. Er schreibt jedoch, dass die genutzten, sehr komplexen Verfahren nicht immer zu dem richtigen Ergebnis führen.

2.1.2 Multidisciplinary Analysis and Optimization

Multidisciplinary Analysis and Optimization (kurz MDAO oder auch MDO: Multidisciplinary Design Optimization) ist eine zunehmend häufig angewandte Methode der Optimierung im Flugzeugvorentwurf (vgl. [37] und [25]). Nach ihr werden alle zu einem Problem gehörenden Teilprobleme simultan betrachtet und optimiert. Sie eignet sich besonders bei Problemen mit vielen Eingängen, die untereinander Abhängigkeiten und Grenzwerte besitzen. Das gefundene Optimum ist in der Regel genauer, als das, welches sich durch Optimieren jedes Teilproblems einzeln findet, da durch MDAO auch die Interaktionen zwischen den einzelnen Komponenten betrachtet werden können. Für den Flugzeugvorentwurf eignet sich die Anwendung von MDAO besonders gut, da dort viele Einzelkomponenten betrachtet werden, deren Änderungen jedoch direkten Einfluss aufeinander haben, da beispielsweise jede Gewichtsänderung die Anforderungen an den Auftrieb und somit den Schub verändert. Ein Nachteil des Verfahrens ist, dass die Kombination aller Teilprobleme zu einem zunehmend komplexen System führt. Dies vergrößert die Zahl der Eingänge, die während einer Optimierung variiert werden müssen und somit die Rechenzeit.
[6]

Daniel P. Raymer beschreibt in [37] ausführlich wie MDAO im Flugzeugvorentwurf verwendet werden kann und vergleicht verschiedene Techniken der Optimierung.

OpenMDAO

OpenMDAO ist eine quelloffene, in Python implementierte MDAO-Software. Sie wird vom NASA Glenn Research Center aktiv entwickelt. [32]

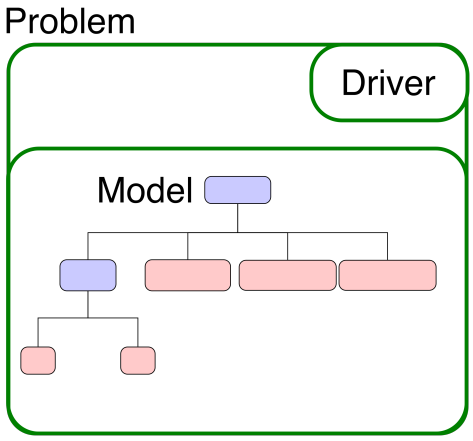

Abb. 2.1: Aufbau eines OpenMDAO Problems [31]

In OpenMDAO werden die Teilprobleme (die roten Kästchen in Abb. 2.1) zu einem Modell zusammengefasst, wobei Schnittstellen zu jedem Teilproblem geschaffen werden. Zudem ist eine Verschachtlung möglich, sollten direkte Abhängigkeiten zwischen bestimmten Teilproblemen bestehen. In dem Modell wird definiert, welche Variablen optimiert werden sollen (design variables), ihre Startwerte und in welchen Grenzen sie sich bewegen können. Es können auch Beschränkungen für Ausgabewerte definiert werden (constraints). Ein sogenannter Driver kümmert sich dann um die Analyse und Optimierung des Modells. Hier können verschiedene Optimierungsalgorithmen zum Einsatz kommen, die aus separaten Python-Paketen wie scipy eingebunden werden. [32]

2.2 Surrogate Modelle

Die in Kapitel 2.1.2 beschriebene Herangehensweise mit MDAO liefert gute Ergebnisse, ist jedoch zeitintensiv und gerade für die Anwendung im Flugzeugvorentwurf evtl. genauer als es nötig wäre. In diesem Kapitel werden Surrogate Modelle, auch Metamodell genannt, beschrieben, die sich genau in diesen Punkten von MDAO unterscheiden.
Das Konzept ist es eine Interpolation der hochauflösenden Daten zu erstellen, auf der dann optimiert werden kann. Dabei wird bereits bei der Planung der Stützstellen angesetzt. Das Kernziel von Surrogate Modellen ist es ein Optimum zu finden und dabei so wenige Auswertungen der hochauflösenden Funktion wie möglich zu benötigen.

Im Allgemeinen ist der Prozess, um ein Surrogate Modell zu erstellen, wie folgt aufgebaut:

- Eingangsgrößen und deren Wertebereiche bestimmen (evtl. mit Hilfe von DoE)
- Stützstellen wählen (Versuchsplan-Methode)
- Surrogate Modell erstellen
- Surrogate Modell trainieren
- Surrogate Modell validieren
- Optimum in dem Surrogate Modell finden

Zu jedem dieser Punkte gibt es verschieden Verfahren, die je nach Anwendung kombiniert werden können. Tabelle 2.2 zeigt bereits, dass es eine Vielzahl von Methoden gibt, deren sinnvolle Anwendung und richtige Kombination Erfahrung verlangt. [14]

Versuchsplan-Methode	**Surrogate Modell**	**Surrogate Modell trainieren**
(Fractional) Factorial	Polynomial	Least Square Regression
Central Composite	Splines	Weighted Least Squares Regression
D-Optimal	Kernel Smoothing	Best Linear Predictor
G-Optimal	Radial Basis Functions	Log-Likelihood
Orthogonal Array	Network of Neurons	Backpropagation
Plackett-Burman	Rulebase or Decision Tree	Entropy
Hexagon Hybrid	Kriging	
Latin Hypercube	Co-Kriging	
Halton		
Select By Hand		
Random Selection		

Tabelle 2.2: Verschiedene Surrogate-Techniken [52, S. 6] [44, S. 3]

Im folgenden Teil dieser Arbeit werden einige dieser Methoden, die sich besonders für Computer-Experimente eignen, genauer beschrieben. Dabei werden jeweils charakteristische Eigenschaften genannt, die in positiv(+), neutral($\sim$) und negativ($-$) unterteilt sind. Alle der folgenden Methoden wurden in Python implementiert, sodass sie erprobt und Beispielrechnungen durchgeführt werden können.

2.2.1 Versuchsplan

Der Versuchsplan (Sampling-Plan) sollte so gestaltet sein, dass mit wenigen Auswertungen der Funktion, möglichst viele Informationen über deren Verlauf gefunden werden. Die einfachste Form ist der bereits in Kapitel 2.1.1 vorgestellte Vollfaktorplan (siehe Abb. 2.2). Eine weitere Möglichkeit ist das central composite Design (CCD). Hier werden die Punkte ähnlich wie bei dem Vollfaktorplan auf den Ecken des Designraums, zusätzlich jedoch auch auf den Achsen und im Mittelpunkt gesetzt (siehe Abb. 2.3). Verglichen mit dem Vollfaktorplan können so einige Punkte eingespart werden [18]. In der Surrogate Modell Literatur wird oft der Begriff D-Optimales Design verwendet. Das Ziel hierbei ist es, die Designpunkte gleichmäßig auf der Zielfunktion zu verteilen, indem die Abstände zwischen den Punkten, gemessen auf der Zielfunktion, möglichst gleich gehalten werden. Da hierfür der Funktionswert an den Stützstellen, sowie der Funktionsverlauf zwischen diesen bekannt sein muss, kann dies jedoch nur im Nachhinein geprüft werden [18]. D-Optimal bezeichnet also keine Versuchsplan-Methode sondern die Validierung dieser. Zudem kann das Verfahren nur bei Polynom-Modellen angewendet werden, weshalb es in dieser Arbeit nicht weiter ausgeführt wird. [48]

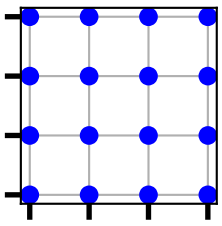

Abb. 2.2: Vollfaktorplan mit 16 Punkten

Bei deterministischen Computerexperimenten werden Versuchsplan-Methoden, wie die Monte Carlo Methode, Latin Hypercube oder Halton angewandt. Diese versuchen die Stützstellen gut verteilt und unregelmäßig zu platzieren. Im Gegensatz zu physikalischen Experimenten, wo vor allem die Versuchsraumränder untersucht werden, eignen sich für Computerexperimente besonders raumfüllende Verfahren [52]. Hierzu kommen zufällige (Monte Carlo) oder quasi-zufällige Methoden zum Einsatz. Für Reproduzierbarkeit werden in dieser Arbeit nur die quasi-zufälligen Algorithmen verwendet. Ziel ist es den Informationsgewinn zu maximieren, indem ausgeschlossen wird, dass ein Eingang an mehreren Stützstellen den selben Wert annimmt. Dies funktioniert besonders deshalb, weil ein Computerexperiment in der Regel keinen zufälligen Fehler (Rauschen) aufweist und somit auch mit wenig Punkten zuverlässig approximiert werden kann (im Gegensatz zu physikalischen Experimenten).

[19]

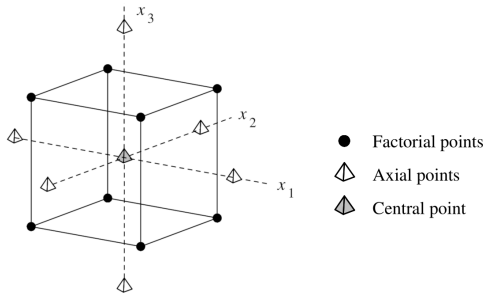

Abb. 2.3: Die verschiedenen Arten von Punkten in einem CCD [18]

Eine noch effizientere Art von Versuchsplan-Methoden ist das sequentielle oder adaptive Sampling. Es werden zunächst mit einem der üblichen Verfahren nur wenige Stützstellen generiert. Die Funktionswerte an diesen Stützstellen werden bestimmt und auf Basis der daraus gewonnenen Erkenntnisse neue Stützstellen hinzugefügt. Hierfür gibt es wiederum zahlreiche Algorithmen, die von R. Ratcliff in [36] genauer beschrieben werden. Eine einfache Anwendung des sequentiellen Samplings wäre es beispielsweise, an dem auf dem Surrogate Modell gefundene Optimum eine neue Stützstelle zu legen und somit genau in diesem Bereich die Präzision zu verbessern. [49] [9]

Vollfaktorplan

Der im Kapitel 2.1.1 beschriebene Vollfaktorplan kann neben der Grundlage einer DoE-Analyse auch als Versuchsplan dienen. Damit er seine Punkte gleichmäßig über den gesamten Designraum verteilt, sollte die Anzahl der Levels der Anzahl der Eingänge entsprechen (siehe Abb. 2.2). Bei Surrogate Modellen findet er keine Anwendung, da er in der Regel sehr ineffizient ist und viele redundante Informationen enthält. Hier soll dieser strukturierte Versuchsplan lediglich als Vergleich heran gezogen werden, um zu prüfen, ob die folgenden Versuchsplan-Strategien einem simplen Raster tatsächlich überlegen sind.

Latin Hypercube

Latin Hypercube ist ein Versuchsplan-Verfahren, welches von einem Raster ausgeht. Jeder Eingang hat die selbe Anzahl von Variationen. Abb. 2.4 verdeutlicht das Prinzip am Beispiel von zwei Eingängen. Zunächst wird ein quadratisches Raster erstellt, welches als Kantenlängen 2^n mit $n \in \mathbb{N}$ hat. Nun werden in diesem Raster Stützstellen verteilt, wobei in jeder Reihe und Spalte genau ein Punkt zu finden ist. Um die Streuung zu erhöhen, wird dies stufenartig vorgenommen. Hierzu wird jede Kante des Rasters in $\sqrt{n}$ Bereiche unterteilt. Wird vom $\mathbb{R}^2$ ausgegangen, ergeben sich also $\sqrt{n} \times \sqrt{n}$ Unterfelder. Diese werden der Reihe nach gefüllt, wobei in den Reihen mit jedem

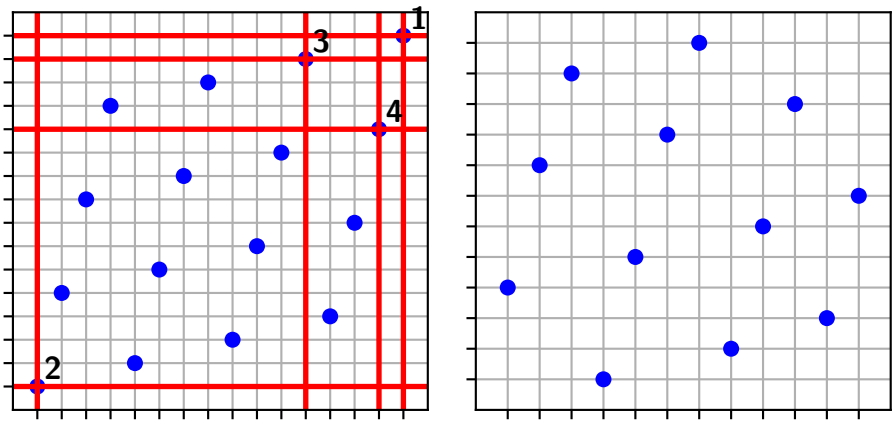

Abb. 2.4: Latin Hypercube Versuchsplan, links: der Plan auf Basis eines 16×16 Raster, rechts: das fertige 12×12 Raster

Feld der Wert des erste Eingangs und für jede neue Reihe der des Zweiten um eins inkrementiert wird (gemessen innerhalb jeden Feldes relativ, siehe Abb. 2.4). Um nun die Größe des Plans auf die gewünschte Anzahl an Punkten zu verkleinern, werden beliebig viele Punkte gelöscht, wobei immer der am weitesten von dem ursprünglichen Zentrum des Plans entfernte Punkt, inklusive dessen Reihe und Spalte entfernt werden. Das Ergebnis ist ein Plan, der die Stützstellen in einem Raster, welches mit natürlichen Zahlen indiziert wird, verteilt. Abb. 2.4 zeigt diesen Prozess für einen Versuchsplan mit zunächst 16 Stützstellen, der auf 12 reduziert wird. [51]

Eigenschaften

\+ Es ist ein simpler Algorithmus.
$\sim$ Das Verfahren ist an ein festes Raster (Schachbrett) gebunden.
$\sim$ Es gibt keine Einstellmöglichkeiten.
$-$ Bei einer Stützstellenanzahl von $n = 2^x$ mit $x \in \mathbb{N}$ ist der Versuchsplan strukturiert (die in Abb. 2.4, links zu erkennende Rautenform).

Halton

Halton Sequenzen weisen eine geringe Diskrepanz auf, sind quasi zufällig und dennoch deterministisch (vgl. Abb. 2.5, rechts). Für jeden Eingangsparameter wird separat eine Reihe erstellt. Diese Reihen werden je auf Basis einer Zahl gebildet. Die Basen müssen untereinander teilerfremd sein. Dies gewährleistet die geringe Diskrepanz. Im Folgenden wird der Algorithmus zum Erstellen der Reihen beschrieben.

- Pro Eingang wird eine teilerfremde Zahl b (Basis) bestimmt.

- Um die i-te Zahl einer Folge der Basis b zu ermitteln, wird zunächst i durch ein Zahlensystem zur Basis b dargestellt.
  - Das einfachste Beispiel ist die Basis $b=2$. Hier wird das in der Informationstechnik weit verbreitete Binärsystem verwendet. Wird beispielsweise die $i=11$-te Zahl gesucht, ergibt sich somit $i=11_{10}=\mathbf{1}\cdot 2^3+\mathbf{0}\cdot 2^2+\mathbf{1}\cdot 2^1+\mathbf{1}\cdot 2^0=1011_2$.
- Nun wird die Inverse von i in Binärschreibweise gebildet.
  - $inv(i_2)=inv(1011_2)=1101_2$
- Diese Zahl wird nun zurück in das Dezimalzahlensystem überführt, wobei sie als Fraktion interpretiert wird.
  - 1101_2 wird interpretiert als 0.1101_2. Umgerechnet wird also mit $0.1101_2=\mathbf{1}\cdot 2^{-1}+\mathbf{1}\cdot 2^{-2}+\mathbf{0}\cdot 2^{-3}+\mathbf{1}\cdot 2^{-4}=0{,}8125$.

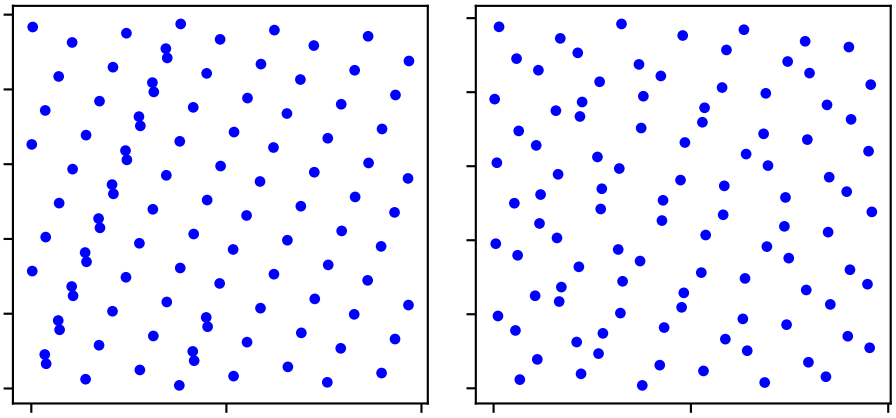

Abb. 2.5: Halton Versuchsplan mit je 100 Punkten, links: mit großer Diskrepanz ($b_1=11$, $b_2=29$), rechts: mit geringe Diskrepanz ($b_1=2$, $b_2=19$)

Dieser Algorithmus kann mit beliebigen Basen angewandt werden. Es empfiehlt sich Primzahlen zu verwenden, da hier garantiert ist, dass diese teilerfremd sind. Die Sequenzen liegen alle in dem Bereich [0..1] und können sehr einfach auf die Eingangsgrößen-Wertebereiche skaliert werden. [34] [53]

Eigenschaften

\+ Halton bietet eine gute Punkteverteilung und geringe Diskrepanz.

∼ Es gibt kein Raster, an das die Punkte gebunden sind.

∼ Das Verfahren ist einstellbar mit großer Variation, durch die unendliche Anzahl an möglichen Kombinationen von teilerfremden Zahlen als Basen.

– Es gibt Kombinationen von Basen (selbst bei der Nutzung von Primzahlen), die zu strukturierten Punkteverteilungen führen (dies kann leicht optisch überprüft werden, siehe Abb. 2.5, links).

2.2.2 Surrogate Modell

Surrogate Modelle haben die Aufgabe aus einer begrenzten Anzahl an diskreten und bekannten Punkten in einem Designraum auf eine Funktion zu schließen, die einen möglichst realistischen Zusammenhang zwischen Eingangswerten und Ausgangswerten darstellt. Um diese Funktion zu generieren, gibt es verschiedene Ansätze, die im Folgenden vorgestellt werden.

Zeichen	Bedeutung
x	Skalar
$\vec{x}$	Vektor
X	Matrix
x^*,$\vec{x}^*$,X^*	Eingangswerte an den bekannten Stützstellen
y^*,$\vec{y}^*$	Ergebnisse an den bekannten Stützstellen
f	die wahre Funktion
$\hat{f}$	die Surrogate-Funktion
y	exakte Lösung
$\hat{y}$	durch Surrogate approximierte Lösung

Tabelle 2.3: Formelzeichen Schreibweise

Tabelle 2.3 erklärt die, in den folgenden Herleitungen verwendeten Symbole. Es wird bei jedem Ansatz davon ausgegangen, dass es eine rechenintensive Funktion $f(\vec{x})$ gibt, wobei $\vec{x}$ eine Liste von $k \in \mathbb{N}$ Eingängen ist. Der skalare Ausgang der Funktion wird y genannt. Durch die vorgestellten Methoden wird eine Approximation $\hat{f}(\vec{x})$ auf Basis von $n \in \mathbb{N}$ bekannten Stützstellen (X^* und zugehörige $\vec{y}^*$) gesucht.

$$f(x_1,x_2,...,x_k) \;=\; f(\vec{x}) \;=\; y \;\approx\; \hat{f}(\vec{x}) \tag{2.1}$$

Es ist zu beachten, dass X^* eine Liste von Stützstellen-Vektoren ist, die die Eingangsgrößen repräsentieren (also eine $n \times k$-Matrix) und $\vec{y}^*$ ein Vektor (mit der Länge n), der die zugehörigen Lösungen enthält. Die folgenden Gleichungen sollen dieses Verhältnis verdeutlichen:

$$\begin{aligned} f(X^*_{(1,1)},X^*_{(1,2)},...,X^*_{(1,k)}) \;=\; & f(X^*_1) \;=\; \vec{y}^*_1 \\ & f(X^*_2) \;=\; \vec{y}^*_2 \\ & f(...) \;=\; ... \\ & f(X^*_n) \;=\; \vec{y}^*_n \end{aligned} \tag{2.2}$$

Zum besseren Verständnis werden die, in diesem Kapitel beschriebenen Verfahren, anhand einer Beispielfunktion dargestellt. Für eine einfache Darstellung hat diese lediglich eine Eingangsgröße ($k=1$):

$$f(\vec{x})=\sin(x_1)+0.95+0.075x_1^2-0.001x^4 \tag{2.3}$$

Polynom Ansatz

Der einfachste Weg eine Funktion zu approximieren ist ein Polynom zu finden, welches die Funktion beschreibt (auch Response Surface genannt). Dies kann mit dem folgenden Ansatz geschehen: [13]

$$\hat{f}(x)=\sum_{i=1}^{o}\vec{w}_i x^i=\vec{w}_1+\vec{w}_2 x+\vec{w}_3 x^2+...+\vec{w}_o x^o \tag{2.4}$$

Mit o, dem Grad des Polynoms und dem Vektor $\vec{w}$, der die Gewichte der einzelnen Anteile enthält. $\vec{w}$ kann bestimmt werden mit:

$$\vec{w}=V\vec{y} \tag{2.5}$$

V ist die Vandermonde-Matrix.

$$V=\begin{bmatrix} 1 & x_1^* & x_1^{*2} & ... & x_1^{*o} \\ 1 & x_2^* & x_2^{*2} & ... & x_2^{*o} \\ ... & ... & ... & ... & ... \\ 1 & x_n^* & x_n^{*2} & ... & x_n^{*o} \end{bmatrix} \tag{2.6}$$

Es ist zu beachten, dass dieser Ansatz lediglich für eine Funktion mit skalarer Eingangsgröße gültig ist.
In [3] wird eine Anpassung des Algorithmus für Funktionen, die zwei Eingangsgrößen verwenden beschrieben. Deren Prinzip ist erweiterbar, sodass auch Funktionen mit beliebig vielen Eingangsgrößen approximiert werden können. Hierzu wird Gleichung 2.4 angepasst, sodass alle möglichen Kombinationen der Eingänge bis zu dem gewählten maximalen Polynomgrad o gebildet werden. Da eine Generalisierung dieser Methode sehr umfangreich ist, wird beispielhaft im Folgenden die

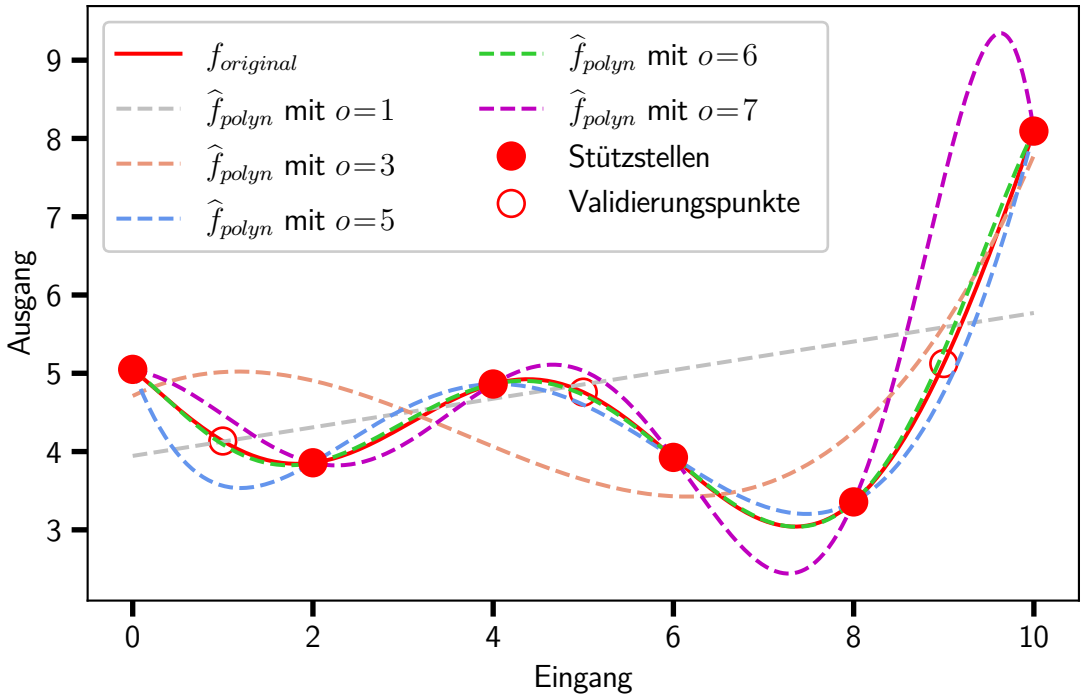

Abb. 2.6: Polynom Modelle verschiedenen Grads im $\mathbb{R}^2$

Gleichung für drei Eingänge und dem maximalen Grad $o=3$ aufgestellt:

$$\begin{aligned}\hat{f}(\vec{x}) = &+\vec{w}_1 \\ &+\vec{w}_2 x_1 + \vec{w}_3 x_2 + \vec{w}_4 x_3 \\ &+\vec{w}_5 x_1^2 + \vec{w}_6 x_2^2 + \vec{w}_7 x_3^2 \\ &+\vec{w}_8 x_1 x_2 + \vec{w}_9 x_2 x_3 + \vec{w}_{10} x_1 x_3 \\ &+\vec{w}_{11} x_1^3 + \vec{w}_{12} x_2^3 + \vec{w}_{13} x_3^3 \\ &+\vec{w}_{14} x_1^2 x_2 + \vec{w}_{15} x_1^2 x_3 + \vec{w}_{16} x_2^2 x_1 + \vec{w}_{17} x_2^2 x_3 + \vec{w}_{18} x_3^2 x_1 + \vec{w}_{19} x_3^2 x_2 \\ &\text{mit} \quad \vec{x} \in \mathbb{R}^n\end{aligned} \tag{2.7}$$

Die entsprechende Vandermonde-Matrix wird analog dazu gebildet:

$$V = \begin{bmatrix} 1 & X_{1,1} & X_{1,2} & X_{1,3} & X_{1,1}^2 & \dots & X_{1,3}^2 X_{1,2} \\ 1 & X_{2,1} & X_{2,2} & X_{2,3} & X_{2,1}^2 & \dots & X_{2,3}^2 X_{2,2} \\ 1 & X_{3,1} & X_{3,2} & X_{3,3} & X_{3,1}^2 & \dots & X_{3,3}^2 X_{3,2} \end{bmatrix} \tag{2.8}$$

Abb. 2.6 zeigt die Beispielfunktion aus Gl. 2.3, die mit Polynomen verschiedenen Grads angenähert wird. Zu erkennen ist, dass das Polynom vom Grad $o=6$ die Funktion sehr genau darstellt. Alle anderen Annäherungen haben jedoch sehr große Abweichungen. Dies verdeutlicht das Potential des Polynom-Ansatzes, zeigt jedoch auch die Schwierigkeit des Trainings, wenn der Grad des Polynoms nicht bekannt ist.

Eigenschaften

+ Der Polynom-Ansatz ist ein sehr einfacher Algorithmus.
+ Das Polynom in bekannter Schreibweise kann leicht in andere Programme überführt werden.
+ Es werden ausschließlich sehr einfachen Rechenoperationen für die Auswertung eines Punktes im Surrogate Modell genutzt.
+ Das Verfahren kann auch zum Glätten von Daten genutzt werden.
∼ Die Rechenzeit für das Erstellen eines Polynom-Surrogate Modells steigt proportional zu der Anzahl der Eingänge an.
∼ Es gibt nur wenige Möglichkeiten der Einstellung.
− Ist der Grad des Polynoms unbekannt, muss er zunächst geraten werden.
− Es besteht die Gefahr des 'overfitting', bei zu hohem Polynomgrad wird der Verlauf der Approximation komplizierter als er eigentlich ist (vgl. Abb. 2.6 $\hat{f}_{polyn}$ mit $o=7$), bzw. 'underfitting' die Funktion wird einfacher nachgebildet als sie eigentlich ist (vlg. Abb. 2.6 $\hat{f}_{polyn}$ mit $o<=5$).

Radiale Basis-Funktion

Name	$\phi(r)$
lineare RBF	r
kubische RBF	r^3
gaußsche RBF	$e^{-(ar)^2}$ mit $a=konst.>0$
inverse-multiquadratische RBF	$(1+r^2)^{a/2}$ mit $a=konst.<0$
multiquadratische RBF	$\sqrt{1+(ar)^2}$ mit $a=konst.$

Tabelle 2.4: Beispiele für radiale Basis-Funktionen [8, S. 3] [45]

Radiale Basis-Funktionen (RBF) sind Funktionen, die nur von ihrem Radius zum Ursprung r abhängig sind. Oft weisen sie zusätzlich eine Konstante a auf, die zum Kalibrieren dient, um eine gute Approximation einstellen zu können. Konkrete Beispiele für RBFs sind in Tabelle 2.4 zusammengefasst. Um die RBF für eine Approximation zu nutzen, kann diese für verschiedenen Stützstellen linear-kombiniert werden.

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n} \vec{w}_i \phi(\| \vec{x} - X_i^* \|) \tag{2.9}$$

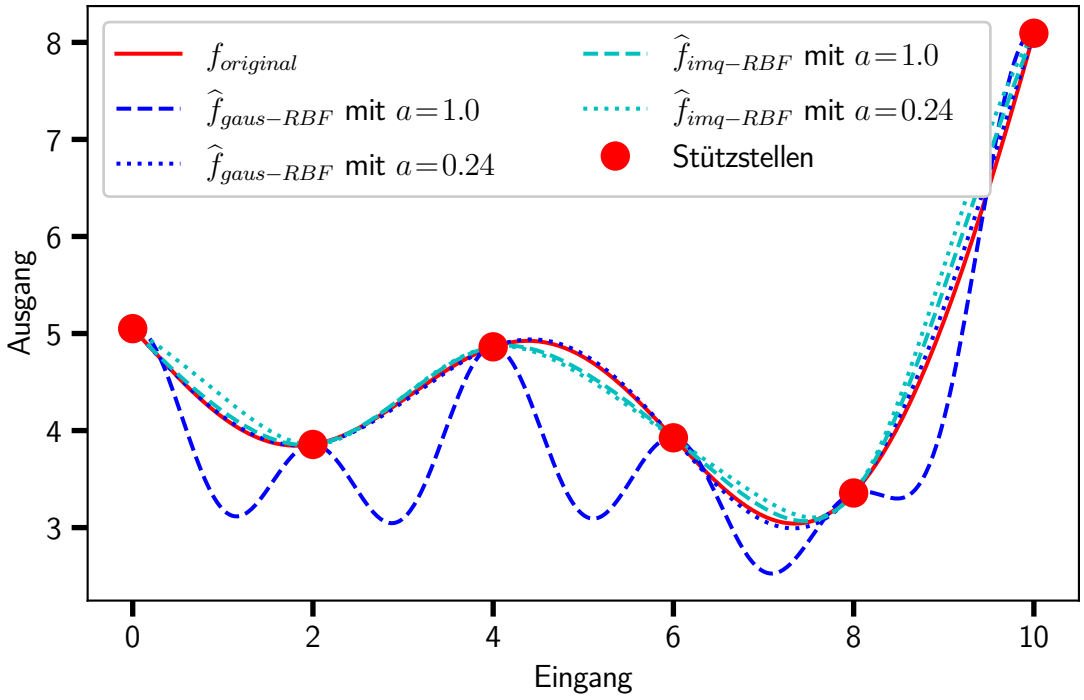

Abb. 2.7: Radial Basis Funktion im $\mathbb{R}^2$

Hier wird die Summe von n RBF (ϕ) mit verschiedenen Zentren (Stützstellen) X_i^* und den Koeffizienten $\vec{w}$ (diese können auch als Gewichtungen der Eingänge interpretiert werden) gebildet. [8, S. 2] Die Koeffizienten lassen sich bestimmen, indem das folgende, lineare Gleichungssystem gelöst wird:

$$\vec{y}^* = \Psi \vec{w} \tag{2.10}$$

Auf der linken Seite steht ein Vektor mit den bekannten Lösungen der Funktion $f(\vec{x})$. Ψ ist eine Matrix, die aus den Lösungen der RBF, an allen Stützstellen untereinander ausgewertet, aufbaut ist:

$$\Psi_{(i1,i2)} = (\phi(\| X_{i1}^* - X_{i2}^* \|)) \qquad \text{mit} \quad 1 \leq i1, i2 \leq n \tag{2.11}$$

Die Indizes $i1$ und $i2$ beschreibt hier die $i1$-te bzw. $i2$-te Stützstelle. [45, S. 11],[8, S. 3]

In Abb. 2.7 ist die Beispielfunktion (Gl. 2.3) mit verschiedenen RBF-Approximationen dargestellt. Blau aufgetragen sind zwei, auf der gaußschen RBF basierende, Approximationen. Es ist zu erkennen, dass mit dem richtigen Wert für die Konstante a (hier $a=0.24$) eine sehr gute Annäherung an die originale Funktion erreicht wird. Mit anderen Werten (bspw. $a=1$) wird die Funktion jedoch instabil. Türkis dargestellt ist eine Approximation mithilfe der inversen-multiquadratischen-RBF.

Es zeigt sich, dass die Wahl der RBF und des Paramters a großen Einfluss auf die Güte der Approximation haben.

Eigenschaften

+ Das Verfahren nimmt nicht nur die bekannten Punkte in das Modell auf, sondern versucht zudem das Gesamtverhalten der Funktion vorherzusagen.
+ Die Koeffizienten werden mit einem einfachen, linearen Gleichungssystem bestimmt, selbst wenn die darzustellende Funktion stark nicht-linear ist.
+ Es ist möglich mit einer einfachen Anpassung Rauschen zu kompensieren (genauer beschrieben in [10, S. 9]).
∼ Die Rechenzeit für das Erstellen eines RBF-Surrogate Modells steigt proportional zu der Anzahl der Eingänge an.
− Es ist viel Erfahrung für die Einstellung der Konstanten a und die Wahl der RBF nötig.
− Mit den falschen Einstellparametern wird das Surrogate schnell sehr instabil (stark oszillierend).

Kriging

Die Interpolation nach Kriging wurde auf einer Theorie des Bergbauingenieurs Daniel G. Krige entwickelt. Sie diente ursprünglich dazu, mit nur wenigen Probebohrungen Aussagen über den Verlauf von Gesteinsschichten im Untergrund zu treffen und somit die Effizienz des Erzabbaus zu steigern. [11] Das Verfahren ist ähnlich aufgebaut wie das mit RBF. Hier wird jedoch zusätzlich eine statistische Untersuchung der Stützstellen durchgeführt, die weitere Informationen über den wahrscheinlichsten Verlauf der Funktion gibt. Die grundlegende Funktion zur Vorhersage eines unbekannten Punktes lautet [14, S. 60]:

$$\hat{f}(\vec{x}) = \mu + \psi^T C^{-1}(\vec{y} - 1\mu) \tag{2.12}$$

μ ist eine Trendfunktion und kann als Offset interpretiert werden. Ist μ eine bekannte Konstante, so spricht man vom Simple Kriging. Ist μ konstant, aber unbekannt, so muss es abgeschätzt werden, was Ordinary Kriging genannt wird. Eine weitere Möglichkeit ist, dass $\mu = \mu(\vec{x})$ eine Funktion des Eingangswerts ist, der Universal Kriging. In diesem Fall wird $\mu(\vec{x})$ in der Regel durch eine polynomiale Regression bestimmt. [5] [26]
In dieser Arbeit wird der im Zusammenhang mit Computerexperimenten meistgenutzte Ordinary Kriging verwendet und mit einer Likelihood-Optimierung trainiert, was im Folgenden genauer beschrieben wird [5].

Somit ist μ eine Konstante und kann abgeschätzt werden mit [14, S. 44]:

$$\mu = \frac{\vec{1}^T C^{-1} \vec{y}}{\vec{1}^T C^{-1} \vec{1}} \qquad \text{mit} \quad \vec{1} = \underbrace{\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}}_{n \times 1} \tag{2.13}$$

ψ ist ein Vektor mit n Einträgen. Die Einträge lassen sich ähnlich des Ansatzes für RBF (siehe Gl. 2.9) mit gaußscher RBF (siehe Tabelle 2.4) bestimmen [14, S. 49]:

$$\psi_{(i)} = exp(-\sum_{j=1}^{k} \vec{\theta}_j \left| X^*_{(i,j)} - \vec{x}_j \right|^{\vec{p}_j}) \qquad \text{mit} \quad 1 \leq i \leq n \tag{2.14}$$

Die Einträge der $n \times n$-Matrix C geben die Korrelation zwischen den Eingangswerten aller bekannten Lösungen an. Sie wird deshalb auch Korrelationsmatrix (eng. Correlation Matrix) genannt. Für jeden einzelnen Eintrag kann diese mit folgender Gleichung bestimmt werden [14, S. 51]:

$$C_{(i1,i2)} = exp(-\sum_{j=1}^{k} \vec{\theta}_j \left| X^*_{i1,j} - X^*_{i2,j} \right|^{\vec{p}_j}) \qquad \text{mit} \quad 1 \leq i1,i2 \leq n \tag{2.15}$$

Hier dient der Exponenten-Vektor $\vec{p}$ zum Einstellen und nimmt Werte im Bereich [1 .. 2] an. $\vec{\theta}$ ist ein Vektor mit Koeffizienten für jeden Eingang, der ebenfalls zur Einstellung dient und einen großen Wertebereich (etwa $[1e^{-5}$ .. $1e^{5}]$) besitzt. Um $\vec{p}$ und $\vec{\theta}$ zu bestimmen, kommt ein statistisches Verfahren zum Einsatz, welches den großen Unterschied zu RBF darstellt. Es wird hierbei davon ausgegangen, dass alle Abweichungen der Approximation von der tatsächlichen Funktion nicht auf Messungenauigkeiten (Rauschen) beruhen, sondern auf Ungenauigkeiten der Interpolation. Dies ist bei deterministischen Verfahren, wie es CFD- und FEM-Rechnungen sein sollten, zutreffend (numerisches Rauschen ist gering und wird vernachlässigt). Daraus wird gefolgert, dass mit ausreichenden Stützstellen die Funktion exakt nachgebildet werden kann. Zu Hilfe kommt hier eine sogenannte Likelihood-Funktion (L, auch Plausibilitäts-Funktion genannt). Eine spezielle Form dieser Funktion wird für die Bestimmung von $\vec{p}$ und $\vec{\theta}$ angewandt. Sie heißt konzentrierte ln-Likelihood-Funktion und ist wie folgt definiert: [14, S. 55]

$$\ln(L) = -\frac{n}{2} \ln\left(\frac{(\vec{y}^* - \vec{1}\mu)^T C^{-1} (\vec{y}^* - \vec{1}\mu)}{n} \right) - 0.5 \ln(|C|) \tag{2.16}$$

Diese Gleichung kann mit verschiedenen Einträgen für $\vec{p}$ und $\vec{\theta}$ ausgeführt werden, bis die logarithmische Wahrscheinlichkeit (Likelihood) $\ln(L)$ minimal ist. Der Verlauf einer solchen

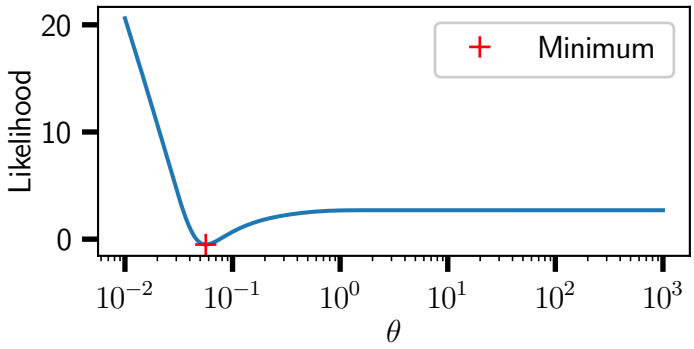

Abb. 2.8: Kriging im $\mathbb{R}^2$, logarithmische Wahrscheinlichkeit von θ

Optimierung ist in Abb. 2.8 für $\vec{\theta}$, unter Verwendung der Beispielfunktion (Gl. 2.3), dargestellt. Zu beachten ist jedoch, dass alle Werte aus $\vec{p}$ und $\vec{\theta}$ zugleich variiert werden müssen, um das Minimum zu finden. Abb. 2.9 zeigt den somit eingepasste Verlauf des Kriging Surrogate Modells. Es ist zu erkennen, dass der Verlauf der Approximation sehr exakt der eigentlichen Funktion folgt.

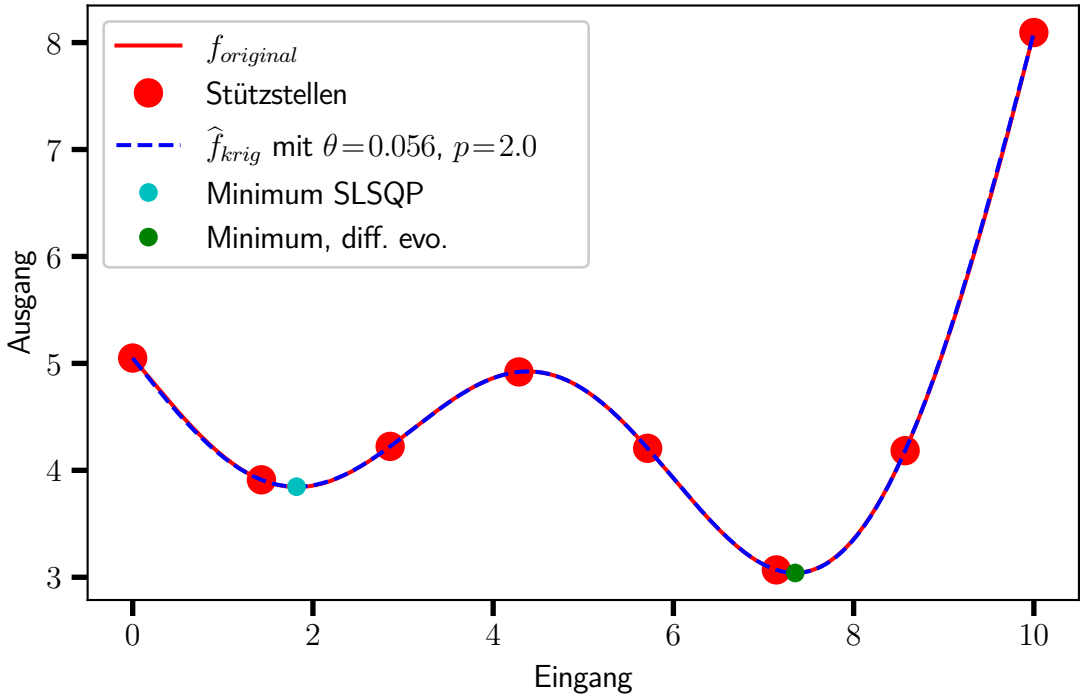

Abb. 2.9: Kriging im $\mathbb{R}^2$

Eigenschaften

+ Kriging approximiert sehr gut.
+ Es werden Informationen über die Position der einzelnen Punkte mit Informationen aus der Verteilung aller Punkte kombiniert (sehr gute Ausbeute der bekannten Informationen).
+ Es wird wenig Erfahrung in der Anwendung erfordert, da die Kalibrationsparameter durch eine Optimierung automatisch bestimmt werden.
∼ Die Wahl der Stützstellen hat großen Einfluss auf den Erfolg des Trainings. Dies gilt für jedes Surrogate Modell, ist jedoch bei Kriging besonders kritisch (vgl. [24, S. 10]).
− Die Implementation ist aufwändige.
− Das Training ist sehr Rechenintensiv und benötigt eine Optimierung (Likelihood-Minimierung), deren Komplexität mit zunehmender Anzahl von Eingängen exponentiell steigt.

[44, S. 31]

2.2.3 Modell-Validierung

Um sicher zu stellen, dass ein Surrogate Modell gut angepasst ist, muss eine Validierung durchgeführt werden. Dies kann mit zusätzlichen Validierungspunkten oder Cross-Validation bewerkstelligt werden, was im Folgenden genauer beschrieben wird. [5]

Zusätzliche Validierungspunkte

Zur Validierung kann die Funktion $f(\vec{x})$ an weiteren, zufälligen (aber möglichst weit von den vorhandenen Stützstellen entfernten) Punkten ausgewertet werden. Anhand dieser Validierungspunkte kann stichprobenartig überprüft werden, wie passend das Surrogate Modell abseits der Stützstellen ist.
Der globaler Fehler kann mittels des root mean square error (RMSE) gemessen werden: [9]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.17}$$

y_i ist der wahre Wert am i-ten Validierungspunkt und $\hat{y}_i$ der vom Surrogate Modell approximierte. Die maximale Abweichung ist gegeben durch den maximum absolute error (MAE): [9]

$$MAE = \max_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.18}$$

Je Validierungspunkt kann der Fehler übersichtlicher als relative absolute error (RAE) dargestellt werden: [9]

$$RAE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2.19}$$

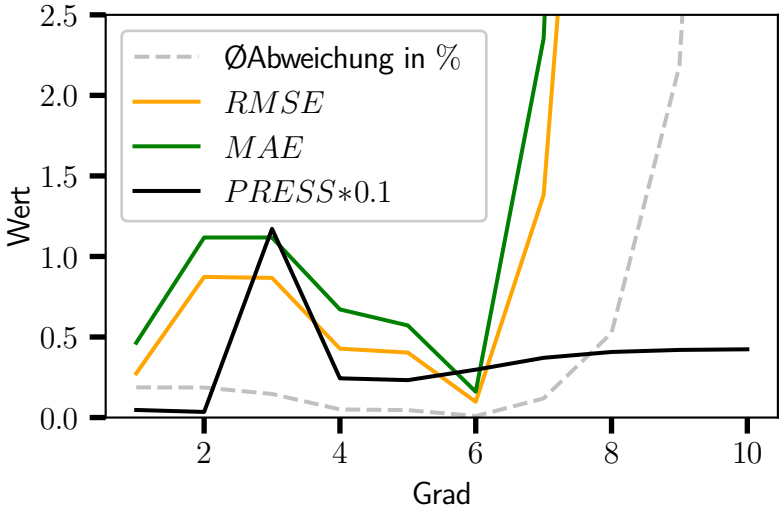

Abb. 2.10: Validierung des Polynom-Ansatzes mit verschiedenen Polynom-Graden

Abb. 2.10 zeigt die Validierung, ausgewertet in den drei, in Abb. 2.6 markierten Validierungspunkten mit verschiedenen Polynom-Graden. Es ist zu erkennen, dass die Fehler RMSE und MAE zwei lokale Minima bei eins und bei sechs haben. Obwohl der Fehler mit einem Polynom ersten Grades relativ gering ist, was hier vor allem an der Wahl der Validierungspunkte liegt, befindet sich das globale Minimum bei sechs, was auch der am besten passenden Approximation entspricht (vergleiche Abb. 2.6). Da die verwendete Beispielfunktion nicht rechenintensiv ist, kann zudem die durchschnittliche Abweichung des Surrogate Modells von der Originalfunktion ermittelt werden. Hierzu werden beide Funktionen an vielen Stellen ausgewertet, die Differenzen zwischen den Ergebnissen errechnet und der Mittelwert von diesen gebildet. In der tatsächlichen Anwendung von Surrogate Modellen ist es nicht möglich die zeitintensive Funktion so oft auszuführen. In dieser Arbeit wird diese Technik jedoch zur genauen Überprüfung der Surrogate-Methoden verwendet.

Die genannten Verfahren haben den Vorteil, dass sie neben der Validierung auch zum Trainieren eines Surrogate Modells genutzt werden können. Im Beispiel des Polynom-Ansatzes kann der Polynom-Grad gefunden werden. Jedoch müssen weitere Punkte der hochauflösenden Funktion berechnet werden, die nicht direkt in das Surrogate Modell einfließen, aber zusätzliche Rechenzeit kosten.

Cross-Validation

Bei der sogenannten Corss-Validation wird das Surrogate Modell n mal jeweils unter Auslassung genau einer Stützstelle trainiert und anschließend der Vorhersagewert an der ausgelassenen Stelle mit dem Originalwert verglichen (leave-one-out Cross-Validation). Zum Vergleich können die oben vorgestellten Fehler genutzt werden, hier wird beispielhaft mit der Summe der Quadrate gerechnet. Dieses spezielle Verfahren wird prediction sum of square (PRESS) genannt (siehe Gl. 2.20). Es gibt darüber Auskunft, wie viel Einfluss die einzelnen Stützstellen auf die Güte des gesamten Surrogate Modells nehmen. [9] [50]

$$PRESS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{f}(X_i^*) - \hat{f}^{(-X_i^*)}(X_i^*))^2} \tag{2.20}$$

Dabei ist $\hat{f}^{(-X_i^*)}$ ein Surrogate Modell auf der selben Datenbasis wie $\hat{f}$, jedoch mit Auslassung der i-ten Stützstellen.

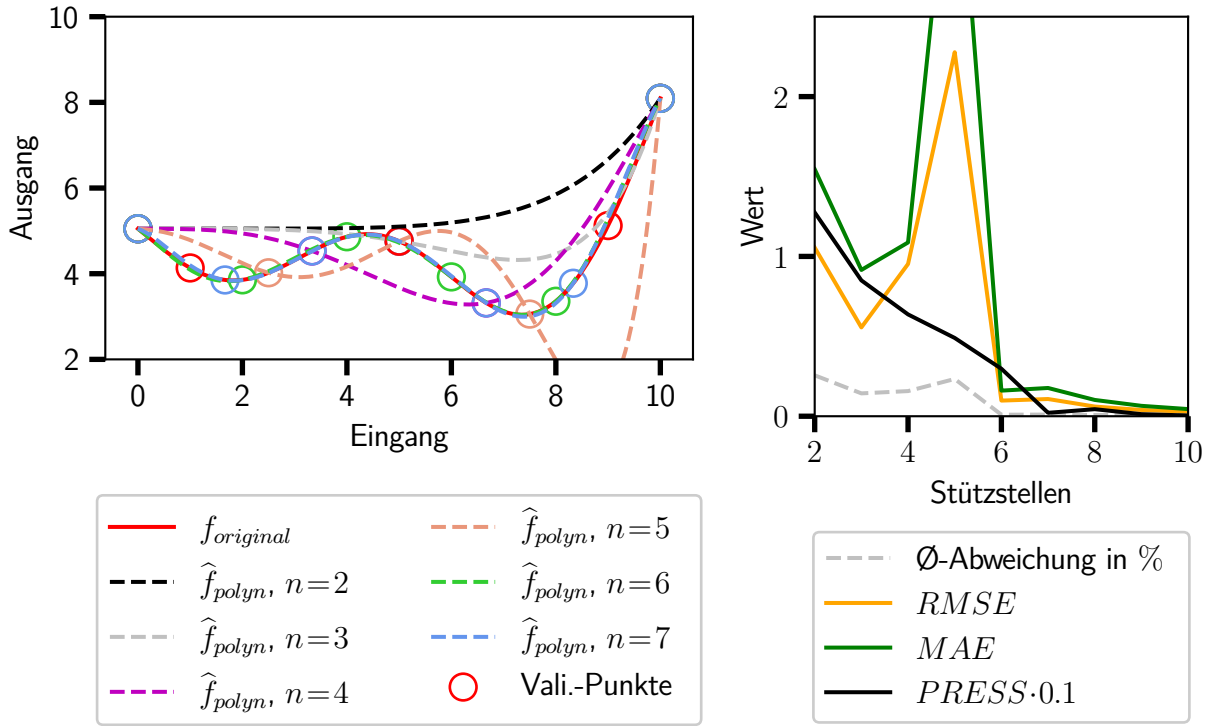

Abb. 2.11: Polynom-Ansatz mit verschiedenen Anzahlen von Stützstellen (links) und zugehörige Fehlerauswertung (rechts)

In Abb. 2.11 wurde der Polynom-Ansatz mit Grad sechs für verschiedene Anzahlen von Stützstellen n (mit strukturiertem Versuchsplan) ausgeführt. Links sind die Polynomverläu-

fe und die dafür genutzten Stützstellen (gleichfarbige Ringe) dargestellt. Auf der rechten Seite sind die jeweiligen Fehler angegeben. Es ist zu erkennen, dass ab einer Anzahl von sechs Punkten der RMSE und der MAE sehr gering sind. Dies wird durch den PRESS-Verlauf bestätigt. Dieser konvergiert deutlich bei sieben gegen Null, was bedeutet, dass bei sieben Stützstellen, selbst bei Auslassung eines beliebigen Punktes, die Güte der Approximation noch etwa die selbe ist. Bei sechs Stützstellen liegt also das Optimum. Dort ist die PRESS höher, was bedeutet, dass jede Stützstelle wichtig ist. Mehr als sechs Stützstellen sind ineffizient, da sie kaum zusätzlichen Gewinn an Informationen bringen (die PRESS ist nahe null).

Der Vorteil dieses Verfahrens ist, dass keine zusätzlichen Punkte für die Validierung benötigt werden, zudem gehen alle berechneten Punkte in das Training des Surrogate Modells ein. Für ein validiertes Surrogate Modell mit ausreichend großem Versuchsplan wird genau eine Stützstelle mehr gebraucht als eigentlich nötig wäre (vgl. Abb. 2.11).

2.2.4 Optimierung

Ist ein Surrogate Modell erstellt und validiert, kann mit beliebigen Eingangswerten (innerhalb der vordefinierten Grenzen) der Funktionswert vorhergesagt werden. Dies kostet sehr wenig Rechenleistung und kann somit ohne Probleme sehr oft geschehen. In der Regel soll nun ein Optimum mit dem Surrogate Modell gefunden werden. Dies sollte so formuliert werden, dass ein Minimum oder eine Nullstelle gesucht wird. Im Folgenden werden einige Standardverfahren hierfür vorgestellt, die alle vom scipy-Python-Paket implementiert werden. Da sie in dieser Arbeit nur als Werkzeug dienen, werden sie nicht hergeleitet, sondern nur ihre Eigenschaften beschrieben.

SLSQP

Sequential Least Squares Programming (SLSQP) ist ein Gradient-basiertes Verfahren. Es muss ein Startwert vorgegeben werden, von dem aus der Algorithmus versucht, das nächstgelegene Minimum zu finden. Es können Grenzen gesetzt werden, in denen der Optimierer die Eingangswerte variieren darf. In Abb. 2.7 ist das Minimum eingezeichnet, dass mit SLSQP gefunden wird, wenn die Stützstelle bei $x = 0$ als Startwert vorgegeben wird. Es wird ein lokales Minimum gefunden. [39]

Differential Evolution

Hierbei handelt es sich um ein stochastisches Verfahren, welches auf mehr Versuche angewiesen ist als SLSQP. Es rät das Globale Minimum in den gesetzten Grenzen und benötigt keinen Startwert.

Nachteilhaft ist, dass es nicht-deterministisch ist und sich somit für automatisierte Verfahren die reproduzierbar sein sollen nicht eignet. [41]

Basin-hopping

Dieser Algorithmus vereint die zwei zuvor vorgestellten. Es wird zufällig der Startwert verändert und von dort mit SLSQP Gradient-basiert nach einem Minima gesucht. Es ist optional möglich, die Schritte, in denen die Startwerte verändert werden, manuell zu setzen. [40]

MDAO

Es ist möglich, die im Kapitel 2.1.2 vorgestellte Technik MDAO zu verwenden, um auf dem Surrogate Modell ein Optimum zu finden. Diese Methode eignet sich besonders, wenn das Finden eines Minimums an Bedingungen geknüpft ist. MDAO lässt es zu, komplizierte Zusammenhänge zu modellieren und verschiedenen Module zu kombinieren. Eines dieser Module kann dann beispielsweise auf Daten eines Surrogate Models zurückgreifen. [32]

Sekantenverfahren

Das Sekantenverfahren ist eine sehr alte, aber bewährte Methode um die Nullstellen einer Funktion zu finden. Es arbeitet sehr effizient und benötigt nur wenige Versuche um die Nullstelle zu lokalisieren. Soll also nach einem bestimmten Ausgabewert des Surrogates gesucht werden, so kann das Sekantenverfahren angewandt werden, indem der gesucht Wert von der Ausgabe des Surrogates subtrahiert und somit ein Nullstellenproblem erzeugt wird. [42]

Kapitel 3

Anwendung von Surrogate Modellen im Flugzeugvorentwurf

In diesem Kapitel werden Beispiele aus der Literatur vorgestellt, bei denen in der Vergangenheit Surrogate Modelle im Flugzeugvorentwurf verwendet wurden. Zudem sollen die im Kapitel 2 beschriebenen Techniken an einem einfachen, praktischen Beispiel angewandt werden. Hierzu wird ein simples Problem mit exemplarischen Daten einer Dash 8 Q400 (siehe Abb. 3.1) konstruiert.

Abb. 3.1: Dash 8 Q400 [2]

3.1 Beispiele aus der Literatur

K. van Velden beschreibt in [49] die Optimierung von Flügelprofilen mit Hilfe von Surrogate Modellen. Auftriebs-, Widerstands- und Momentenbeiwert werden mittels eines CFD-Lösers

(MSES) bestimmt. Es werden verschiedene Versuchsplan- und Surrogate-Methoden vorgestellt und verglichen, wobei van Velden zu dem Ergebnis kommt, dass Kriging die beste Approximation liefert. GPML (Gaussian Process for Machine Learning) ist nur dann besser, wenn sehr viele Stützstellen vorhanden sind.

In [35] nutzt J. Rang ein Surrogate Modell um die DOC (Direct Operating Costs) eines Flugzeugs mit Hochauftriebssystem zu minimieren. Hierzu werden sieben Parameter variiert, die Flügelgeometrie, Triebwerke und die Flugmission beschreiben. Die Berechnung einer Konfiguration beinhaltet eine rechenintensive Optimierung, in der das Flugzeug so abgestimmt wird, dass es auf einer limitierten Startbahnlänge abheben kann. Es werden Polynome mit der least square Methode mit linearem und quadratischem Ansatz verwendet, um ein Surrogate Modell zu erstellen.

D. Neufeld zeigt in [30] wie der Flügelkasten eines Verkehrsflugzeugs im Flugzeugvorentwurf mithilfe von FEM-Berechnungen (ANSIS) durch ein Surrogate Modell optimiert werden kann. Es wird dabei sehr detailliert auf die Auslegung des Flügelkastens eingegangen. Ein dynamisch erweiterbarer Versuchsplan (Sequential Sampling) wird zusammen mit Kriging verwendet. Werden Ergebnisse aus einem Teil des Designraums benötigt, der nicht gut erschlossen ist, werden weitere Stützstellen an diesem Ort hinzu gefügt. Dieses Prinzip kommt in Verbindung mit der MDAO-Software RyeMDO zum Einsatz, die auf dem Surrogate Modell ein Optimum sucht.

In [54] sucht W. Yongliang nach gewichts-optimierten, versteiften Röhren, wie sie beispielsweise für den Körper von Weltraumraketen benötigt werden. Hierzu werden mit einer Abaqus-FEM-Analyse die Spannungen und das Beulen (buckling) der Struktur untersucht. Um Rechenzeit zu sparen, wird mit 100 zufälligen Punkten, in denen fünf Eingänge variiert werden, ein Kriging-Surrogate Modell gebildet. In diesem wird mit dem Multi-Island generic Algorithmus das globale Optimum angenähert. Anschließend wird von dort mit einem Gradient-basierter Optimierungsalgorithmus (SLP, sequential linear programming algorithm) das exakte Optimum gefunden.

[38] behandelt die für den Flugzeugentwurf relevante Optimierung von versteiften Schalen in Faserverbundbauweise, wie sie beispielsweise bei modernen Tragflügeln von Linienflugzeugen Verwendung finden. Es wird eine Kombination von realen Experimenten und FEM-Berechnungen genutzt, um ein Surrogate Modell mit dem Polynom-Ansatz zu generieren. Dabei wird durch Variation der Hautdicke, sowie Stringerdicke und Form das Gewicht minimiert.

A. Giunta schreibt in seiner Dissertation [17] über die Optimierung von sub- und supersonic Aerodynamik eines zivilen Transportflugzeugs. In der bereits 1997 veröffentlichen Arbeit geht es ihm auch darum das Rauschen aus der numerischen Lösung herauszufiltern. Bei den meisten Computerexperimenten spielt gerade bei modernen, hoch-präzisen Rechnern Rauschen jedoch keine Rolle mehr (lediglich wenn auf Daten zurückgegriffen wird, die in diskreten Schritten angegeben sind) [14].

Diese Beispiele zeigen, dass Surrogate Modelle durchaus Potential in dem Flugzeugvorentwurf aufweisen. Jedoch ist auch zu erkennen, dass selbst in erst kürzlich veröffentlichten Forschungsergebnissen Surrogate Modelle nur vereinzelt eine Rolle spielen und ihnen oft viel Aufmerksamkeit gewidmet wird, da sie noch nicht zu den Standardwerkzeugen der Luft- und Raumfahrt Ingenieure gehören.

3.2 Beispielhaftes Entwurfsproblem

Dank leistungsfähigen Computern kommen so genannte Highfidelity-Methoden (hochauflösende Methoden) wie FEM und CFD zunehmend im Flugzeugvorentwurf zum Einsatz. In diesem Kapitel wird ein Beispielproblem kreiert und anschließend unter Verwendung der in Kapitel 2 beschriebenen Verfahren untersucht. Zu den Vorentwurfsmethoden gibt es zahlreiche Literatur, an der sich orientiert wird, wie F. Hürlimann [20], der die strukturelle, sowie aerodynamische Flügelauslegung eines Transportflugzeuges beschreibt oder O. Dababneh [12], der auf den iterativen Optimierungsprozess im Flügelentwurf eingeht.

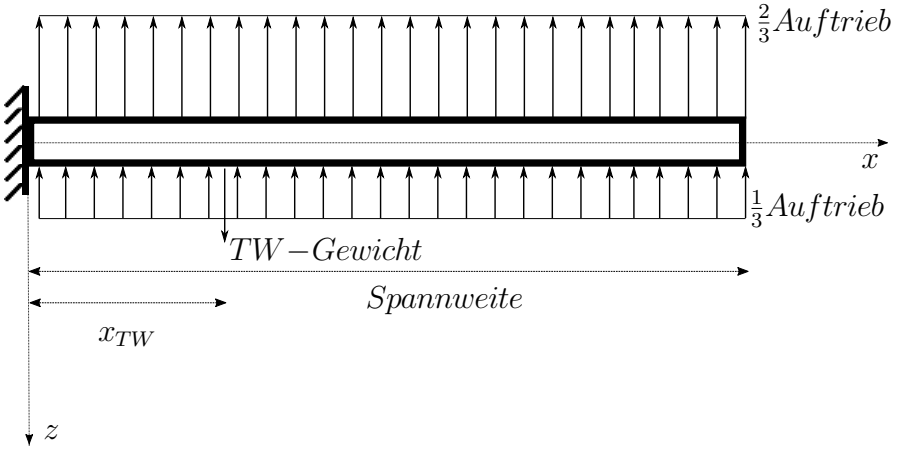

Abb. 3.2: Lasten und Randbedingungen der Tragflächenstruktur [23]

Die Flügelstruktur eines Dash 8 Q400 (siehe Abb. 3.1) ähnlichen Flugzeuges soll auf ihr Gewicht optimiert werden. Dieses Flugzeug hat eine so kleine Flügel-Pfeilung und -Zuspitzung, dass diese Parameter vernachlässigt werden, was die Konstruktion der Struktur stark vereinfacht. Es wird

davon ausgegangen, dass die äußere Geometrie des Flügelkastens bekannt und vorgegeben ist (Daten, siehe Tab. 3.1).

	Parameter	Wert	Einheit
Flugzeuggeometrie	Halbspannweite	12,87	m
	Flügeltiefe	3,00	m
	davon Holmbereich	40	%
	Flügeldicke	0,55	m
	Triebwerksabstand zum Rumpf in Spannweitenrichtung	3,00	m
Flugzeugmassen, -Kräfte	MTOW	27987	kg
	Treibstoffmasse im Flügel	2659	kg
	Triebwerksmasse	1300	kg
Materialeigenschaften (Aluminium 7075)	Dichte	2810	kg/m^3
	Poissonzahl	0,33	-
	Elastizitätsmodul	$71{,}7e^9$	Pa
	Zugfestigkeit	$572e^6$	Pa
Sonstiges	Sicherheitsfaktor	1,5	-
	Lastvielfaches	1,0	-

Tabelle 3.1: Flugzeugparameter [22] und Materialeigenschaften [27]

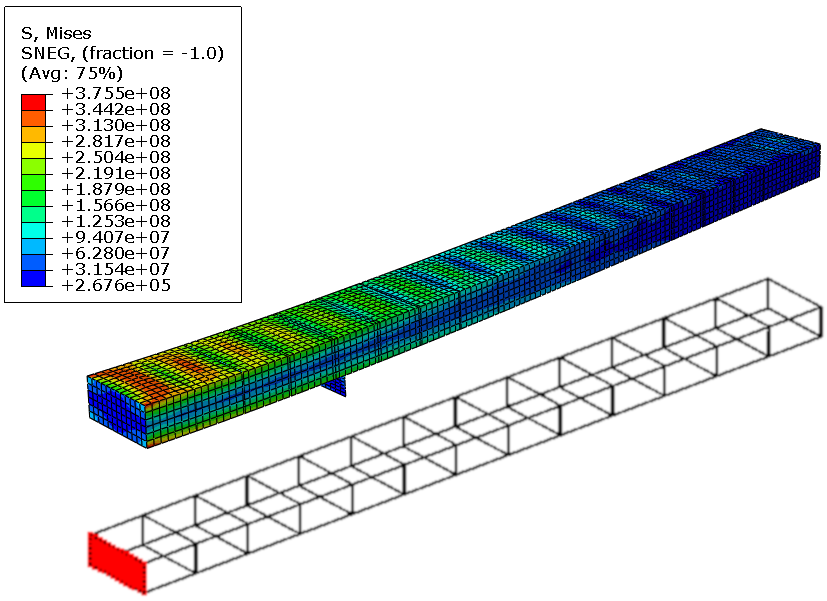

Abb. 3.3: FEM-Modell der vereinfachten Flügelstruktur

Zur Gewichtsoptimierung kann die Anzahl der Rippen, sowie die Materialstärke der gesamten Struktur variiert werden. Dabei muss die Belastung des Materials im zulässigen Bereich bleiben. Tabelle 3.1 listet die als gegeben betrachteten Parameter auf. Die technischen Flugzeugdaten sind dem Airport Planning Manual der Dash 8 Q400 entnommen [22]. Abb. 3.2 zeigt den Flügel und die aus dem Auftrieb resultierende Last von vorne. Da das Flugzeug symmetrisch ist, wird nur ein Flügel und von diesem lediglich die tragende Struktur berechnet. Abb. 3.3 zeigt die Struktur des Flügelkastens und die Ergebnisse einer Abaqus FEM-Berechnung. An der Flügelwurzel ist eine feste Einspannung. Die Kräfte sind so vereinfacht, dass sie lediglich in z-Richtung wirken (vgl. Abb. 3.2). Diese und alle weiteren vereinfachten Annahmen, die getroffen wurden, sind im Folgenden zusammengefasst.

Annahmen

- Es wird lediglich die primär tragende Struktur modelliert (Holme, Rippen, obere und untere Gurtplatte, siehe Abb. 3.3).
- Der Flügel ist ungepfeilt.
- Der Flügel ist nicht zugespitzt.
- Die Lasten aus dem Auftrieb werden zu zwei Dritteln gleichmäßig auf der Flügeloberseite und zu einem Drittel auf der Flügelunterseite verteilt (sie Abb. 3.2).
- Es treten nur Kräfte auf, die senkrecht auf den Flügel wirken (kein Widerstand).
- Es wird nur der Fall mit einfacher, statischer Belastung (Lastvielfaches = 1) betrachtet.

Umsetzung

Kategorie	**Name**	**Version**	**Verwendung**
Betriebssystem	Windows	10 x64	
FEM-Software	Abaqus	6.14	FEM-Solver, Pre- und Postprocessor
	Calculix	6.2	FEM-Solver, Pre- und Postprocessor
Programmiersprache	Python	3.6	Hauptprogramm
Python Pakete	PyKriging	0.1.0	Kriging Surrogate Modell
	pyDOE2	1.1.1	Versuchsplan-Methoden
	OpenMDAO	2.4.0	MDAO
	numpy	1.15.1	Vektor- und Matrix-Operationen
	scipy	1.1.0	Optimierung (für OpenMDAO und Likelihood)
	pyoptsparse	1.0.0	Optimierung (für OpenMDAO)

Tabelle 3.2: Verwendete Software

Die Berechnung der Lasten erfolgt mit dem FEM-Solver Abaqus. Alle übrigen Operationen werden in der Programmiersprache Python durchgeführt. Diese wurde gewählt, da sie frei verfügbar (+ open-source) ist und viele Bibliotheken für mathematische Operationen, sowie Surrogate Modellierung bereit stellt. Eine Übersicht der in dieser Arbeit verwendeten Pakete ist in Tabelle 3.2 zu finden. Dort ist zusätzlich alle andere verwendete Software mit Versionsnummern aufgeführt. Im Anhang A.1 sind die einzelnen Python-Quellcode-Dateien genauer beschrieben. Für Modularität und eine bessere Übersicht ist der Quellcode nach dem objektorientierten Softwarekonzept verfasst.

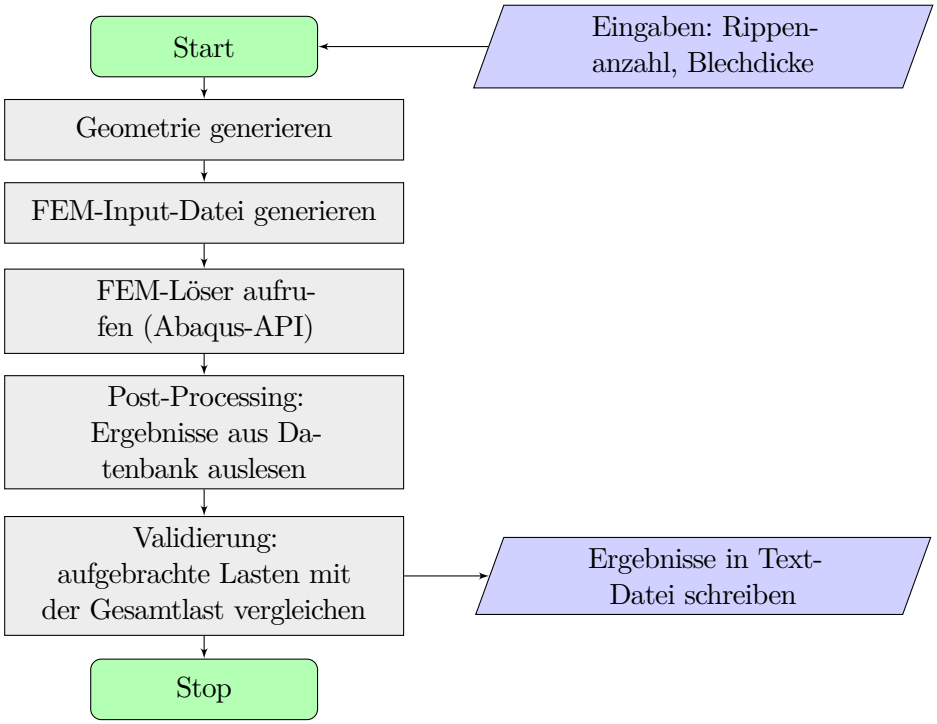

Abb. 3.4: Programmablaufplan: FEM-Berechnung

Abb. 3.4 beschreibt (gem. DIN 66001) den Ablauf des Teilprogramms, welches die FEM-Berechnung von einer Flügelkonfiguration durchführt. Die Variablen, die später optimiert werden sollen, sind die Rippenzahl und die Blechdicke der Struktur und stellen deshalb die Eingangsgrößen dar. Mit diesen Angaben wird in Python ein fbl-File generiert, welches mit Hilfe von Calculix-Befehlen die gewünschte Geometrie beschreibt. Mit dem Calculix Pre-Processor (cgx) wird dann aus der Beschreibung in dem fbl-File ein Netz, sowie die Randbedingungen und Lasten abgeleitet. Calculix wird hier genutzt, da es Kommandozeilen basiert läuft und so automatisierten Zugriff von Python zulässt. Ein inp-File, welches Calculix und Abaqus gleichermaßen unterstützen, wird erstellt, das alle zuvor generierten Informationen zusammenfasst und die Materialeigenschaften, sowie die FEM-Löser-Optionen definiert. Mit Hilfe der nicht ganz so umfangreichen Kommandozeilen API von Abaqus kann das

inp-File an den Löser übergeben und die Datenbank mit den Ergebnissen gespeichert werden. Diese kann mit dem Abaqus Kommandozeilen-Befehl 'odbreport' von dem gegebenen Binärformat in eine CSV umgewandelt werden, welche sich leicht mit Python einlesen und auswerten lässt. Hierbei ist für diese Anwendung lediglich die maximale von Mises Spannung von Interesse, welche für die spätere Verwendung in einer Textdatei gespeichert wird. Das Gewicht wird aus der Summe der Volumina, der an der Struktur beteiligten Bleche, multipliziert mit der Dichte von Aluminium 7075 (aus Tabelle 3.1) berechnet. Da im frühen Stadium der Python-Software oft Inkonsistenzen in der Auftriebskraft-Aufbringung auftraten, wird diese validiert, indem die aufgebrachten Einzellasten aus der FEM-Eingabedatei summiert und mit der vorgegebenen Gesamtlast verglichen werden.

Design of Experiment

Exemplarisch wird eine einfache DoE-Betrachtung durchgeführt, um die Wertebereiche der Eingänge und ihre Einflüsse zu überprüfen. Da es lediglich zwei Eingänge gibt, kommt der Vollfaktorplan auf zwei Levels zum Einsatz. Wo die Levels liegen muss abgeschätzt werden. Die untersuchten Levels werden so gewählt, dass das gesuchte Optimum mit großer Wahrscheinlichkeit zwischen ihnen liegt. Auf Erfahrung basierend fällt die Entscheidung auf die folgenden Werte:

- Rippenanzahl: 5 bis 25
- Blechdicke: 2,0 bis 3,3 mm

Die Blechdicke ist nicht mit tatsächlichen gebräuchlichen Materialstärken vergleichbar, da die Wölbung der Tragfläche und Strukturelemente wie Stringer vernachlässigt werden. Untersucht wird lediglich die von Mises Spannung, da die Einhaltung dieser die dominante Rolle der Optimierung spielt.

3.2.1 Lösung durch MDAO

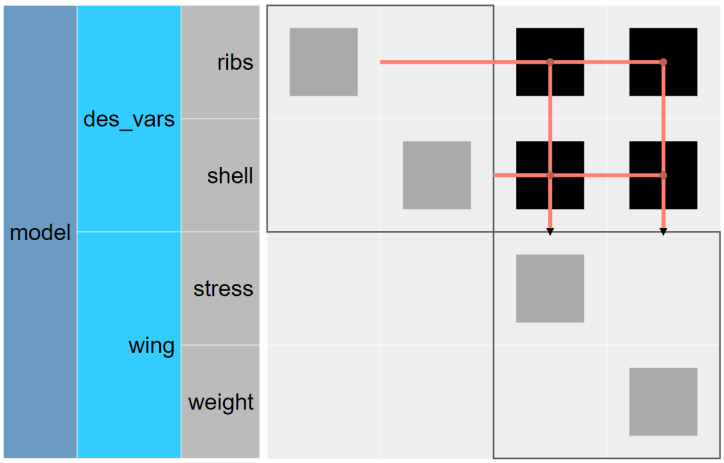

Abb. 3.5: OpenMDAO Modell

Um die im Folgenden beschriebenen Surrogate-Methoden mit einem klassischen Optimierungsverfahren vergleichen zu können, wird hier das Beispielproblem mit Hilfe von MDAO gelöst. Zum Einsatz kommt OpenMDAO (siehe Kapitel 2.1.2). Das zu optimierende Modell ist in Abb. 3.5 dargestellt. Es hat zwei Design-Variablen, die vom Optimierer variiert werden können und gleichermaßen auf die Ausgänge maximale Spannung und Gewicht Einfluss nehmen. Während das Gewicht minimiert wird, wird die maximale Spannung beschränkt (constraint), sodass sie im zulässigen Bereich bleibt. OpenMDAO bietet die Möglichkeit verschiedene Optimierungs-Bibliotheken zu wählen. Da sich der Ausgang der untersuchten Funktion stetig verhalten sollte, führt ein Gradient-basiertes Optimierungsverfahren am schnellsten zum Ziel. Zudem ist wichtig, dass Randbedingungen gesetzt werden, die den Optimierer innerhalb sinnvoller und für die FEM-Rechnung gültiger Werte hält. Der einzige Optimierer aus der scipy-Bibliothek, der diese Voraussetzungen erfüllt, ist SLSQP. [39]
Es wurde festgestellt, dass die Optimierung teilweise abbricht, wenn die Ausgänge nicht auf den Bereich [0..1] skaliert sind. Auch die Eingänge sollten skaliert sein, sodass ihre Variation durch den Optimierer im signifikanten Bereich erfolgt. Standardmäßig erfolgt diese mit SLSQP zunächst in der sechsten Nachkommastelle und orientiert sich nicht an den gesetzten Grenzen, was für beispielsweise die Rippenanzahl unnötige, zusätzliche Versuche kostet.

Neben scipy kann auch das sogenannte pyoptsparse-Paket für die Steuerung der Optimierung genutzt werden. Es bietet ebenso wie scipy verschiedene Algorithmen zur Optimierung. Problematisch ist jedoch die Installation unter Windows. Es gibt keine vorkompillierte Istallationsdatei für Python unter Windows, weshalb jeder der im Paket enthaltenen Optimierer manuell kompiliert werden muss. Hierfür werden C und Fortran Compiler benötigt, sowie Visual Studio. An letzterem scheiterten die Versuche. Unter Ubuntu konnte pyoptsparese erfolgreich installiert werden, da aber der FEM-Solver unter Windows eingerichtet ist, wurde dies nicht weiter verfolgt. Lediglich ein Optimierungsalgorithmus namens ALPSO ist in dem pyoptsparse-Paket ohne Aufwand nutzbar, da er komplett in Python geschrieben ist. Dies ist ein Partikel-Schwarm Algorithmus, der ebenso wie SLSQP Randbedingungen unterstützt. Zudem kümmert sich der Algorithmus auf Wunsch selbständig um die Skalierung der Ein- und Ausgänge. [33]

Unabhängig von der Wahl des Optimierungsalgorithmus tritt in dem konkreten Beispiel der Rippenanzahl-Optimierung das Problem auf, dass diese durch eine natürliche Zahl representiert wird. Alle verfügbaren Optimierer unterstützen jedoch nur reelle Eingänge. Die Rippenanzahl lediglich zu runden führt zumindest mit SLSQP nicht zum gewünschten Ziel, da in diesem Fall die Optimierung bei Rippenvariation im Nachkommabereich keinen Gradient feststellen kann. Deshalb wird bei Rippenanzahlen mit fraktalem Anteil die FEM-Berechnung einmal mit abgerundeter und einmal mit

aufgerundeter Zahl ausgeführt und anschließend linear interpoliert. Dies erhöht den Rechenaufwand um den Faktor zwei, führt jedoch zur Konvergenz des Optimierers. Um OpenMDAO dazu zu zwingen bei einer ganzzahligen Rippenanzahl zu konvergieren, wurde versucht eine Bestraffunktion einzuführen, die das Strukturgewicht proportional zum fraktalen Anteil der Rippenzahl künstlich verschlechtert. Dies erfüllt zwar den gewünschten Effekt, hat jedoch den Nebeneffekt, dass der Gradient-basierten Algorithmus SLSQP aufgrund der sich einstellenden hügeligen Funktion schlechter konvergiert. Mit ALPSO hingegen kann dieses Verfahren problemlos angewandt werden.

3.2.2 Lösung durch Surrogate Modell

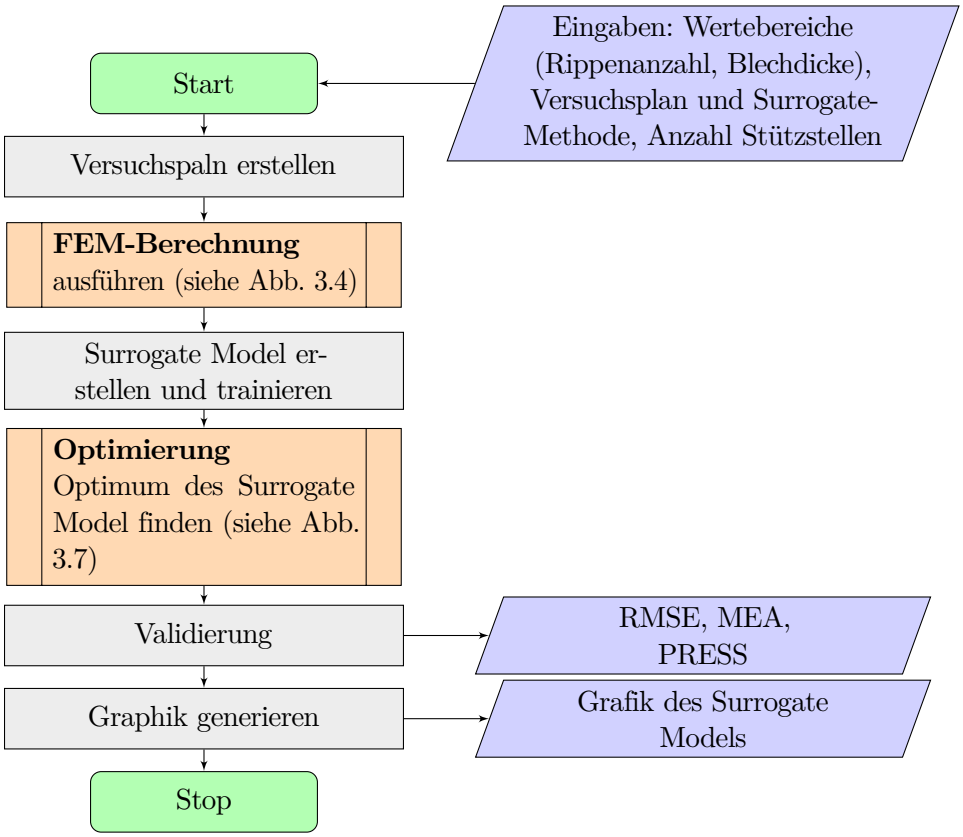

Abb. 3.6: Programmablaufplan: Surrogate Modell erstellen

Der allgemeine Ablauf der Surrogate Modell-Generierung ist in Abb. 3.6 dargestellt. Eingaben sind die Wertebereiche für Rippenanzahl und Blechdicke, die untersucht werden sollen, sowie die zu verwendende Versuchsplan- und Surrogate-Methode. Zusätzlich kann die Anzahl von Stützstellen, die verwendet werden soll, vorgegeben werden. Zunächst werden die konkreten Stützstellen entsprechend des gewählten Versuchsplan-Algorithmus generiert, die dann jeweils mit dem in Abb. 3.4 beschriebenen Teilprogramm analysiert werden. Mit den Ergebnissen wird das gewählte Surrogate Modell erstellt und trainiert. Mit diesem Modell kann das Optimum

gesucht werden, was an späterer Stelle genauer beschrieben wird. Das Surrogate Modell wird dann entsprechend der in Kapitel 2.2.3 vorgestellten Algorithmen validiert und eine grafische Darstellung abgespeichert.

Alle in Kapitel 2.2 beschriebenen Techniken wurden in Python (für den $\mathbb{R}^k$ mit $k \in \mathbb{N}$) so implementiert, dass sie einheitliche Schnittstellen haben und beliebig kombiniert werden können. Im Folgenden werden die dabei aufgetretenen Probleme und gewonnenen Erkenntnisse geschildert.

Versuchsplan

Die Implementation des Vollfaktorplans wurde bereits im Kapitel 2.1.1 beschrieben. Ähnlich funktioniert der Versuchsplan nach Halton. Beide Verfahren lassen sich leicht auf den $\mathbb{R}^k$ implementieren. Beim Vollfaktorplan sind die Dimensionen repräsentiert durch die Stellen der Zahl im Zahlensystem zur Basis n. Bei Halton wird für jeden Eingang eine eigene Reihe aufgestellt, die unabhängig voneinander funktionieren. Etwas komplizierter wird es bei der Implementation von Latin-Hyper-Cubes. Hier muss der gesamte Design Raum zunächst in jeder Dimension in $\sqrt[k]{n}$ Unterräume aufgeteilt werden. Ergibt diese keine natürliche Zahl, muss der nächst größere Raum gebildet werden. Im $\mathbb{R}^2$ handelt es sich dabei um viele Quadrate, im $\mathbb{R}^3$ um Würfel. Anders als bei den anderen vorgestellten Versuchsplan-Methoden wird hier eine k-dimensionale Matrix mit der Kantenlänge n erstellt, die als Raster für die Stützstellen dient. Diese Matrix ist mit Nullen gefüllt, soll eine Stützstelle gesetzt werden, wird an dieser Stelle eine Eins in die Matrix geschrieben. Dazu müssen alle Teilräume nacheinander durchlaufen werden. Das Problem dabei ist, dass dies nicht in normalen Schleifen geschehen kann, da hierzu k verschachtelte Schleifen nötig wären und diese nicht zur Laufzeit erzeugt werden können. Deshalb wurde eine Adressierung der Teilräume eingeführt. In einem Vektor werden die Indizes in alle k Richtungen gespeichert. Nun kann dieser Adress-Vektor (wie bei dem Vollfaktorplan) mit einem Zahlensystem zur Basis k hochgezählt werden. Die Adressierung innerhalb jedes Teilraums erfolgt nach dem selben Prinzip, es gibt einen weiteren Adress-Vektor, der die lokale Adresse im Teilraum hochzählt, um die gewollte stufenartige Verschiebung der Punkte zu erhalten. Bei der Reduzierung des $\sqrt[k]{n}$-Raums auf die tatsächlich gewünschte Raumgröße kann mit dem auf k-Dimensionen erweiterten Satz des Pythagoras (siehe Gl. 3.1) der Abstand der Punkte zum Mittelpunkt verglichen werden. Die Variable *center* entspricht der halben Kantenlänge des unreduzierten Designraums.

$$r = \sqrt{\sum_{i=1}^{k} (\vec{x}_i - center)^2} \tag{3.1}$$

Surrogate Modell

Die Implementierung der Surrogate-Methoden ist Kernbestandteil dieser Arbeit, weshalb hier auf jedes der genutzten Verfahren detailliert eingegangen wird.

Polynom-Ansatz

Die Schwierigkeit hier ist, die k-Dimensionen mit dem maximalen Polynom-Grad o zu verknüpfen. Es wird versucht möglichst alle Eingänge miteinander zu kombinieren und dabei o nicht zu überschreiten. Um dies zu bewerkstelligen, kommen vier verschachtelte Schleifen zum Einsatz. Der folgende Pseudo-Code beschreibt den Aufbau der Vorhersage-Funktion des Polynom-Ansatzs. Diese soll exemplarisch den Aufbau der Schleifen zeigen. Die Funktion zum Berechnen der Vandermonde-Matrix nutzt den selben Aufbau. Es wird ein großes Polynom zusammengebaut, welches um einzelne Terme erweitert wird. Jeder Term wird mit einem eigenen Gewicht $\vec{w}_i$ multipliziert, was mit 'Gewicht[..]' angedeutet wird (vgl. Eq. 2.4). Diese wurden zuvor in einer separaten Funktion berechnet.

```
1  Polynom addieren mit: Gewicht[..]
2  für jeden Polynomgrad o1 in [1 .. o]
3    für jeden Eingang k1 in [1 .. k]
4      Polynom addieren mit: Gewicht * k1^o1
5      für jeden Eingang k2 in [1 .. k]
6        wenn 2*o1 < o
7          Polynom addieren mit: Gewicht[..] * k1^o1 * k2^o1
8        wenn k1 ungleich k2
9          für jeden Polynomgrad o2 in [1 .. min(o1, o - o1)]
10           Polynom addieren mit: Gewicht[..] * k1^o1 * k2^o2
```

Der Algorithmus fügt zunächst den Offset mit dem ersten Gewicht in Zeile 1 ein. In Zeile 4 rechnet er jeden Eingang mit jedem erlaubten Grad hinzu. In Zeile 7 wird je ein Eingang mit jedem anderen Eingang mit der selben Potenz multipliziert. In Zeile 10 werden alle restlichen möglichen Kombinationen von je zwei Eingängen multipliziert. Die Zeilen 8 und 9 fangen dabei Dopplungen und zu hohe Polynom-Grade ab.

RBF

Eine Besonderheit der Approximation mit RBF ist, dass verschiedene RBF genutzt werden können (siehe Tabelle 2.4). In Python kann dies sehr einfach bewerkstelligt werden, indem die zu benutzende Funktion als sogenannter Pointer zu der Funktion übergeben wird. Dabei ist nur zu beachten, dass die Python-RBF-Funktionen alle die gleichen Argumente haben. Ein Beispiel, wie diese Funktionen aufgebaut sind, ist in dem folgenden Python-Code-Auszug dargestellt:

```
1 def lin_rbf(a, r):
2   return r
3 def gaus_rbf(a, r):
4   return math.e**(-((a*r)**2))
```

Hier werden die lineare und die Gauß'sche-RBF implementiert. Es ist zu erkennen, das der Einstellparameter als Argument existieren muss, selbst wenn dieser nicht benötigt wird (Zeile 1).

Kriging

Bei der in Kapitel 2.2.2 beschriebenen Approximation nach Kriging spielt die Minimierung der logarithmischen Wahrscheinlichkeit (Likelihood) eine zentrale Rolle. Diese gestaltet sich in der Implementation sehr kompliziert, da über $2k$ Parameter optimiert werden muss. Zudem haben die Parameter p (mit [1..2]) und θ (mit etwa $[1e{-}5..1e5]$) sehr unterschiedliche Wertebereiche. Dieses Problem kann umgangen werden, indem statt θ eine Variable θ' eingeführt wird, mit $\theta = 10^{\theta'}$. Nun kann θ', mit einem Wertebereich von etwa $[-5..5]$, als Eingang der Optimierung genutzt werden, was zur Stabilität des Prozesses beiträgt.
Die Python Bibliothek scipy bietet verschiedene Algorithmen zur Optimierung von Funktionen mit skalarem Ausgang und einem Vektor von Eingangsgrößen. Der Gradient-basierte Minimierungsalgorithmus SLSQP konvergiert schnell, benötigt jedoch Startwerte (eine erste Vermutung des Minima) und endet in der Regel in einem lokalen Minima nahe dieser. Das rein stochastische Verfahren differential Evolution hat dieses Problem zwar nicht, hat jedoch Schwierigkeiten mit der Größe des Designraums und ist nicht deterministisch, was das Testen erschwert. Eine Kombination dieser Verfahren bietet Basin-hopping. Hier wird an mehreren zufällig gewählten Startwerten mit SLSQP detailliert gesucht. Der Algorithmus eignet sich zwar gut für die Optimierung der logarithmischen Wahrscheinlichkeit, hat jedoch weiterhin die Schwäche, dass der stochastische Teil des Verfahrens nicht reproduzierbare Ergebnisse liefert.[39] [41] [40]
Aus diesem Grund wurde ein weiteres Verfahren entwickelt, bei dem zunächst der gesamte Raum, indem sich die θs und ps befinden können, in ein Raster unterteilt und in jedem Feld dieses Rasters die logarithmische Wahrscheinlichkeit berechnet wird. Auf Basis der besten Werte aus diesem Raster wird nun mit SLSQP das genaue Minima bestimmt. Der Aufwand dieses Verfahrens wächst mit der k-ten Potenz ($k =$ Anzahl der Eingangsgrößen), da das Raster k-dimensional ist.

Optimierung

Abb. 3.7 beschreibt die in Abb. 3.6 verwendete Gewichtsoptimierung. Sie wird auf dem Surrogate Modell für jede Rippenanzahl einzeln vorgenommen, da diese durch eine natürliche Zahl dargestellt wird, was die gängigen Optimierungsalgorithmen nicht unterstützen. In diesem Fall ist nur ein

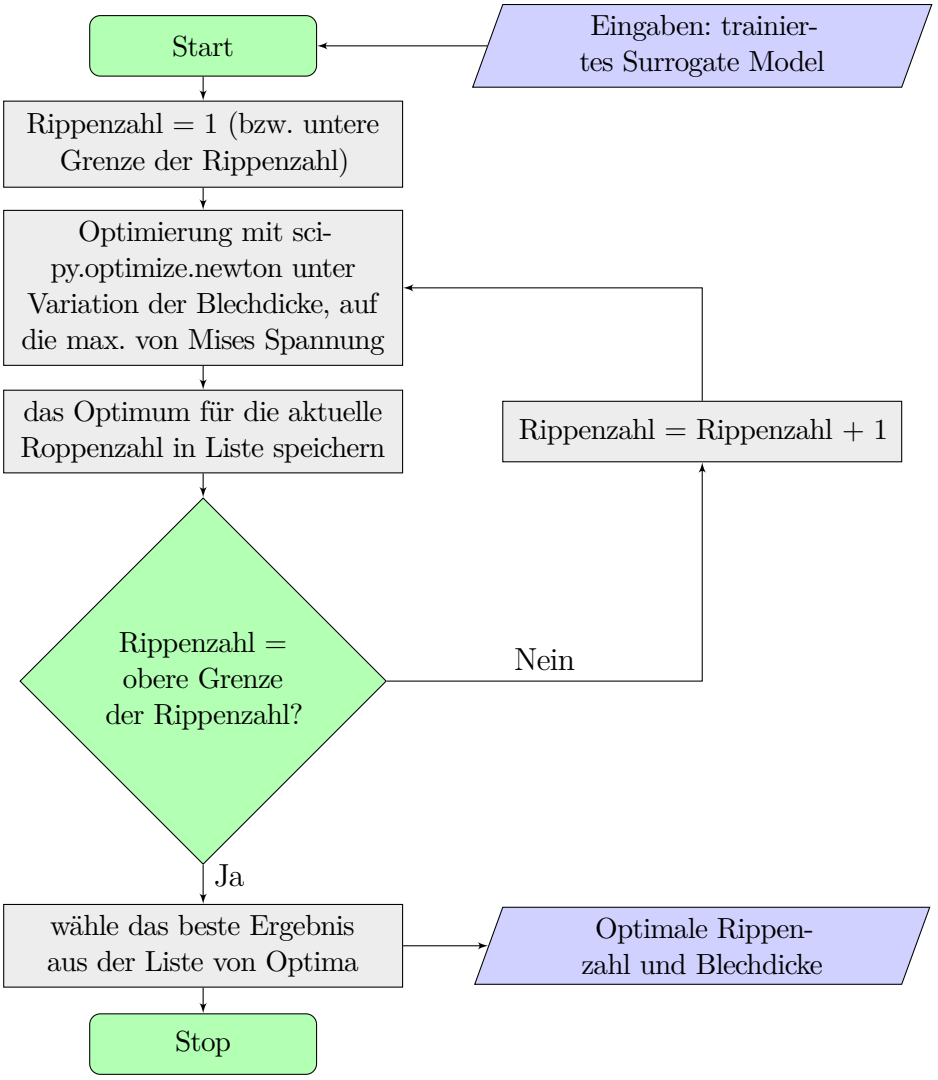

Abb. 3.7: Programmablaufplan: Optimierung auf dem Surrogate Modell

Eingang (die Blechdicke) zu variieren und die Auswertung eines Punktes im Surrogate Modell geht sehr schnell. Deshalb liegt die Rechenzeit hierfür im Sekundenbereich. Obwohl das Gewicht die zu minimierende Größe ist, muss dieses in der Optimierung zunächst nicht betrachtet werden. Es wird der Punkt gesucht, bei dem die maximal zulässige von Mises Spannung knapp unterschritten wird. Unter der Annahme, dass die Spannungsfunktion über der Blechdicke streng monoton steigend ist, muss an dieser Stelle das geringste Strukturgewicht vorliegen, da leichtere Strukturen zum Bruch führen und bei schwereren unnötig viel Material verwendet wird. Dieser Punkt wird gefunden, indem die maximal zulässige Spannung von den vom Surrogate vorhergesagten Spannungen subtrahiert wird. Ein Nullstellen-Problem wird konstruiert, welches mit dem Sekantenverfahren (scipy.optimize.newton [42]) gelöst wird. Die optimale Blechdicke wird mit diesem Verfahren für jede Rippenzahl einzeln bestimmt. Das leichteste dieser Ergebnisse wiederum ist das globale Optimum.

Validierung

Die in Kapitel 2.2.3 vorgestellten Gleichungen zur Validierung können ohne großen Aufwand in Python umgesetzt werden. Es muss jedoch auch bestimmt werden, wo die Validierungspunkte liegen sollen. Sie sollten sich möglichst weit weg von den bekannten Stützstellen befinden, da dort bei den meisten Surrogate-Methoden die Funktion sehr viel besser approximiert ist als abseits von ihnen. Für eine gute Vergleichbarkeit zwischen den verschiedenen Versuchsplan- und Surrogate-Methoden werden Punkte gewählt, die bei jedem Versuch gleich bleiben. Um die Wahrscheinlichkeit zu minimieren, dass die Validierungspunkte mit den Stützstellen übereinstimmen, werden vier Punkte um einen in der Mitte des Designraums liegenden fünften Punkt schneckenförmig angeordnet. Abb. 3.8 stellt diese Anordnung überlagert über einen Halton Versuchsplan dar. Die Linien zwischen den Validierungspunkten dient lediglich der besseren Erkennung.

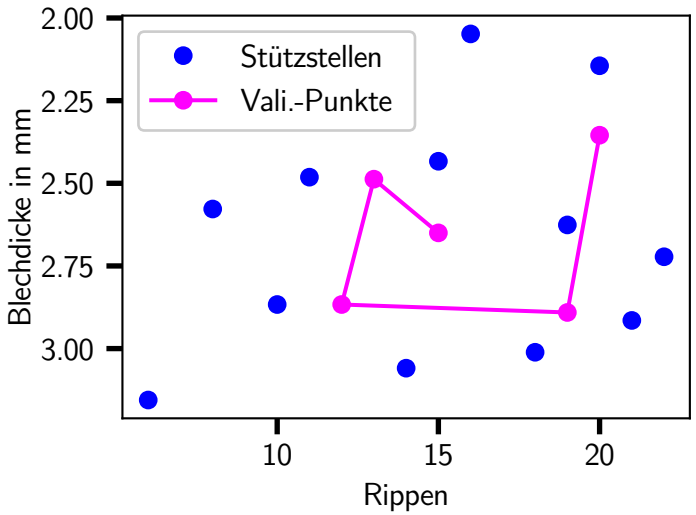

Abb. 3.8: Anordnung der Validierungspunkte über einem Halton Versuchsplan

3.2.3 Ermittlung der exakten Lösung

Dieses Kapitel beschreibt, wie eine exakte Lösung des Flügelkastenproblems ermittelt wird, die als Referenzlösung dient. Dies ist nötig, wie im späteren Verlauf dieser Arbeit erläutert wird, da sich mit Hilfe von OpenMDAO keine vertrauliche Lösung ermitteln ließ. Hierzu kommt das Sekantenverfahren direkt im Zusammenhang mit der FEM-Rechnung zum Einsatz. Wie bereits in Abb. 3.7 beschrieben, wird für jede Rippenzahl separat die Blechdicke gesucht, die die maximal zulässige von Mises Spannung knapp einhält. Auch hier kommt das Python Paket scipy.optimize.newton zum Einsatz [42]. Das Problem wird in Teilprobleme zerlegt, die jeweils

nur einen Eingang (die Blechdicke) haben. Somit vergrößert sich zwar der Rechenaufwand, da jedes Teilproblem zunächst für sich optimiert werden muss, jedoch werden die Schwierigkeiten, die im Zusammenhang mit dem diskreten Rippenanzahlen auftraten umgangen.

Kapitel 4

Auswertung

In diesem Kapitel werden die Ergebnisse und Erfahrungen geschildert, die mit den in Kapitel 3 vorgestellten Programm gewonnen werden konnten. Zunächst wird die unabhängig vom Optimierungs-Algorithmus vorgenommene DoE-Analyse ausgewertet. Dann werden alle zur Optimierung verwendeten Verfahren einzeln vorgestellt und im Anschluss ein Vergleich zwischen ihnen vorgenommen. Bei einigen Versuchen wird die Laufzeit angegeben. Um diese einordnen zu können, sind in Tabelle 4.1 die Daten des verwendeten Rechners aufgelistet.

Prozessor	Intel i7-4800MQ @ 2,70 Ghz
RAM	16 GB
Betriebssystem	Win 10 x64

Tabelle 4.1: Eigenschaften des verwendeten Testrechners

4.1 Design of Experiment

Rippen	Blechdicke	Ergebnis abs.	Ergebnis in %
$-$: 5	$-$: 2mm	6,50e+8	170,4
$-$: 5	+: 3,3mm	3,06e+8	80,1
+: 25	$-$: 2mm	4,82e+8	126,5
+: 25	+: 3,3mm	2,90e+8	76,5

Tabelle 4.2: Die Ergebnisse der DoE Untersuchung

Tabelle 4.2 zeigt die Ergebnisse der einfachen DoE Analyse. In den ersten zwei Spalten ist notiert, welches Level des jeweiligen Eingangs untersucht wird ($-$ für tief, + für hoch, dahinter der Zahlenwert). Darauf folgt die sich einstellende maximalen von Mises Spannung als Zahlenwert und

in Prozent von der maximal zulässigen von Mises Spannung. Das Optimum wird an einem Punkt liegen an dem 100% der maximal zulässigen von Mises Spannung ausgereizt sind. In der letzten Spalte der Tabelle 4.2 ist zu erkennen, dass an zwei Ecken des Designraums dieser Wert unterschritten und an den anderen zwei überschritten wird. Dies bedeutet, dass im Designraum Punkte liegen, die der Forderung, genau die maximal zulässige von Mises Spannung einzuhalten, entsprechen (wird von einem stetigen Funktionsverlauf ausgegangen). Mit einer grafischen Darstellung der Tabellenwerte können weitere Zusammenhänge erklärt werden. Abb. 4.1, oben links zeigt die Einflüsse von Rippenanzahl und Blechdicke. Es ist jeweils die Veränderung von Level − bis + aufgetragen, wobei der jeweils andere Eingang auf − gehalten wird. Beide Eingänge zeigen deutlichen Einfluss, was an der vorhandenen Steigung der Geraden zu erkennen ist. Der Einfluss der Blechdicke übertrifft den der Rippenanzahl, da die Blechdicken-Gerade eine größere Steigung aufweise. Im unteren Teil von Abb. 4.1 sind die Interaktionen der beiden Eingänge dargestellt. Links ist der Einfluss einer Blechdicken-Veränderung dargestellt, einmal mit der Rippenanzahl konstant auf Level − und einmal auf +. Da die beiden Geraden nicht parallel verlaufen und sich nicht schneiden, handelt es sich um synergistische Interaktion. Das bedeutet, dass Erhöhung der Eingänge jeweils die Funktion in die selbe Richtung beeinflusst. In Abb. 4.1 unten rechts ist der selbe Effekt für die Veränderung der Rippenanzahl bei gleichbleibender Blechdicke zu erkennen. Hier ist zudem zu sehen, dass bei einer Blechdicke auf Level + (also 3,3mm) die Anzahl der Rippen nahezu keinen Einfluss mehr auf die maximale von Mises Spannung hat (zu erkennen an dem fast konstanten Verlauf der gestrichelten Linie). [4, S. 22]

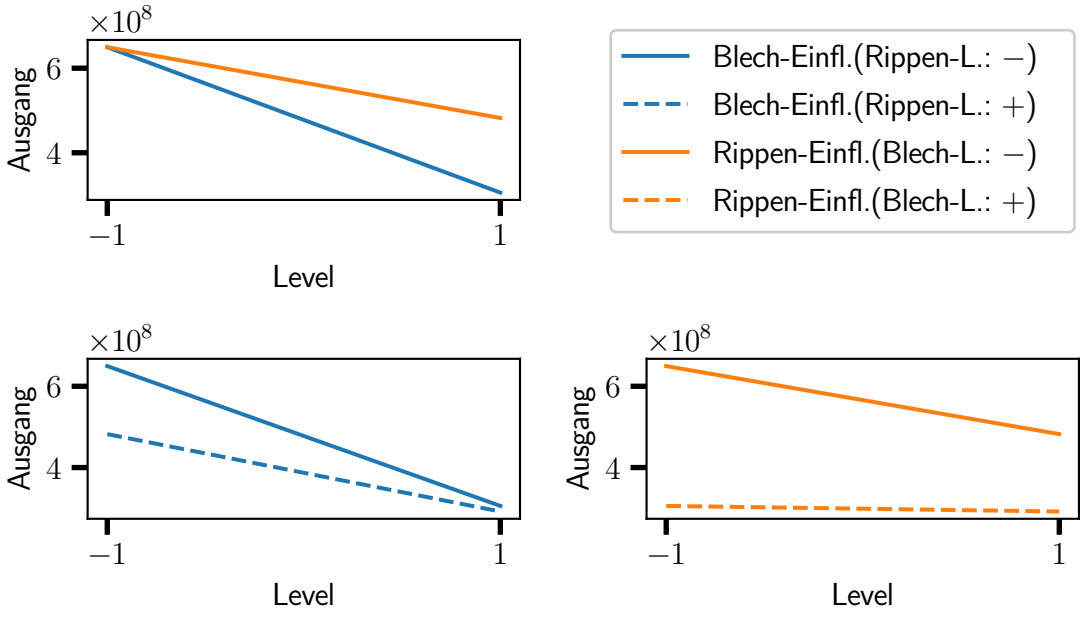

Abb. 4.1: DoE, Einfluss (oben) und Interaktion (unten links und rechts) der Eingänge

4.2 Multidisciplinary Analysis and Optimization

Während der Implementation des OpenMDAO-Lösers traten zahlreiche Probleme auf, die in der folgenden Liste zusammengefasst sind:

- Unabhängig von dem gewählten Optimierer ist das Konvergenzverhalten sehr instabil.
  - Das Ändern der Startwerte hat großen Einfluss auf die Konvergenz. Teilweise wird trotz gültiger Startwerte keine Konvergenz erreicht.
  - Bei unskalierten Ausgängen konvergiert SLSQP bereits nach zwei Iterationen mit falschem Ergebnis.
  - Je nach Skalierung des Ausgangs bricht SLSQP teilweise mit der Fehlermeldung 'Optimization FAILED. Positive directional derivative for linesearch' ab.
- Ob Konvergenz erreicht wird ist oft schwer vorhersagbar, da bei bestimmten Einstellungen die Eingänge periodisch anfangen zu schwingen.
- ALPSO liefert gute Ergebnisse und ist sehr viel stabiler als SLSQP, konvergiert jedoch langsamer und baut auf stochastische Verfahren auf, die schlecht reproduziert werden können.

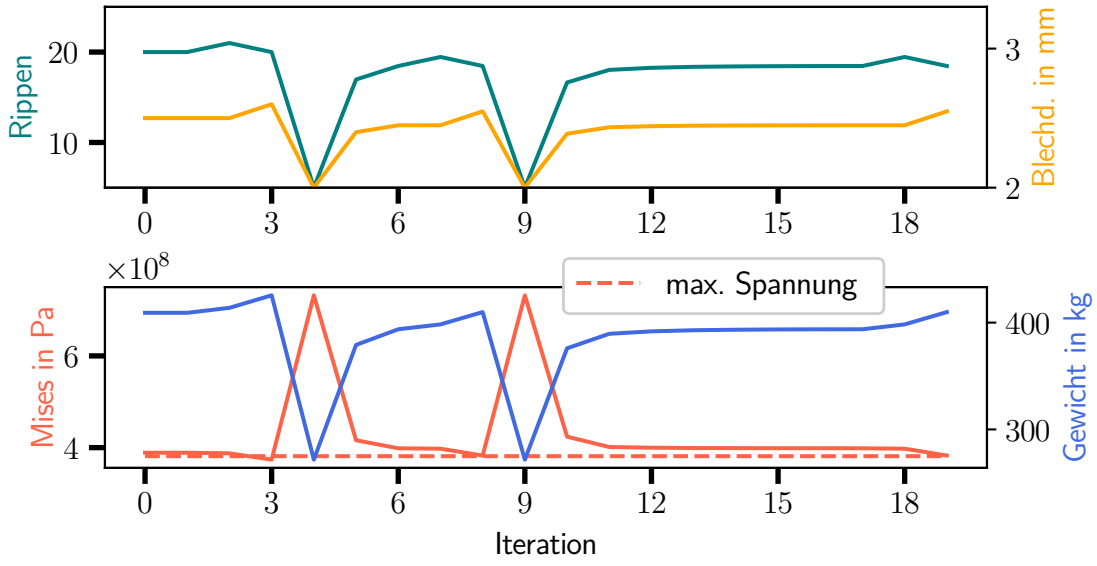

Abb. 4.2: Eingänge und Ausgänge während der Optimierung mit SLSQP

Abb. 4.2 zeigt die Variation der Eingänge und die entsprechenden Ausgangsgrößen über die gesamte Optimierung mit SLSQP. Es ist zu erkennen, dass die Eingänge periodisch verändert werden. Sie sind strukturierter als bei der Optimierung mit ALPSO, die in Abb. 4.3 dargestellt sind. ALPSO variiert besonders in den ersten Iterationen die Eingänge in sehr großen Schritten, während

SLSQP nur wenige große Sprünge macht. Daraus kann folgen, dass SLSQP in einem lokalen Minimum endet und das globale übersieht, während ALPSO mit größerer Wahrscheinlichkeit das globale findet. In den Plots der Ausgänge ist rot-gestrichelt der Grenzwert für die maximal zulässige von Mises Spannung eingezeichnet. Es ist bei beiden Optimierungen zu erkennen wie die von Mises Spannung um diesen Wert schwingt.

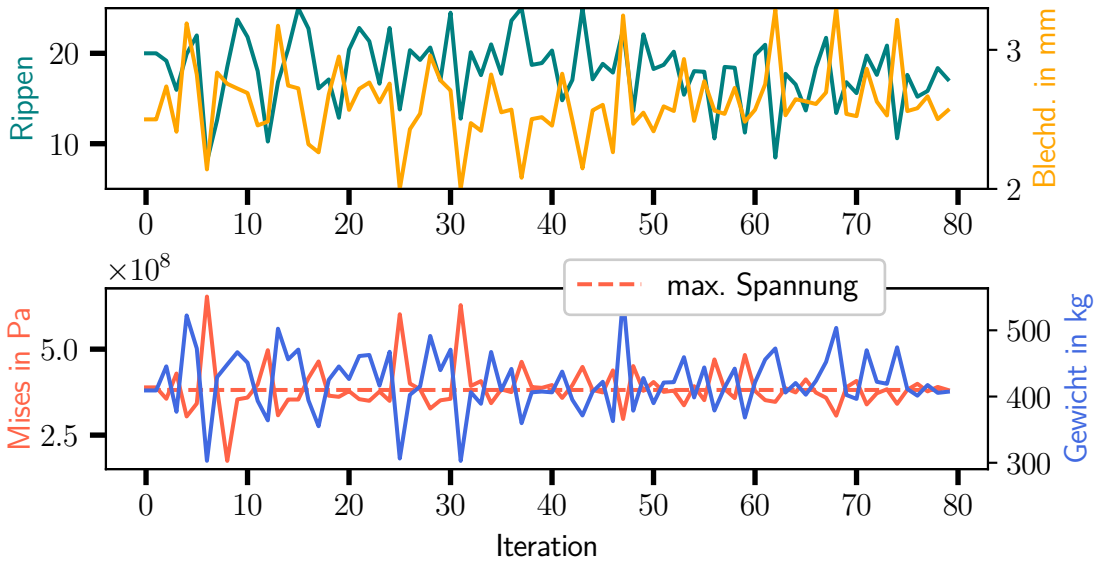

Abb. 4.3: Eingänge und Ausgänge während der Optimierung mit ALPSO

Die Ergebnisse der OpenMDAO-Versuche sind in Tabelle 4.3 aufgeführt. Es zeigt sich, dass mit dem SLSQP-Algorithmus die Bedingung der maximal zulässigen von Mises Spannung nicht eingehalten werden konnte. Damit ist das Gewicht ungültig. Zudem beträgt die Rippenzahl 18,5, was ebenfalls kein gültiger Wert ist. Mit ALPSO hingegen wird der Grenzwert exakt getroffen und die Rippenzahl hat einen geringen fraktalen Anteil.

	Iterationen	Rippenanzahl	Blechdicke	max. von Mises	Gewicht
Grenzen	200	5 - 25	2,0 - 3,3 mm	< 3,81e+8 Pa	-
SLSQP	20	18,5	2,45 mm	3,98e+8 Pa	393,8 kg
ALPSO	80	17,1	2,56 mm	3,81e+8 Pa	407,3 kg

Tabelle 4.3: Die Ergebnisse der OpenMDAO-Versuche

4.3 Surrogate Modell

Im Folgenden werden einige Versuche beschrieben und ausgewertet, die die Qualität und Anwendbarkeit der Versuchsplan- und Surrogate-Methoden untersuchen. Da die verwendete FEM-Rechnung in wenigen Sekunden ausgeführt werden kann, wurden zunächst in einem feinen Raster innerhalb der Designraum-Grenzen Messungen durchgeführt, die später zur optischen Validierung der Surrogate Modelle Verwendung finden. Dies dient nur der Untersuchung der Verfahren und ist in der eigentlichen Surrogate-Anwendung nicht üblich.

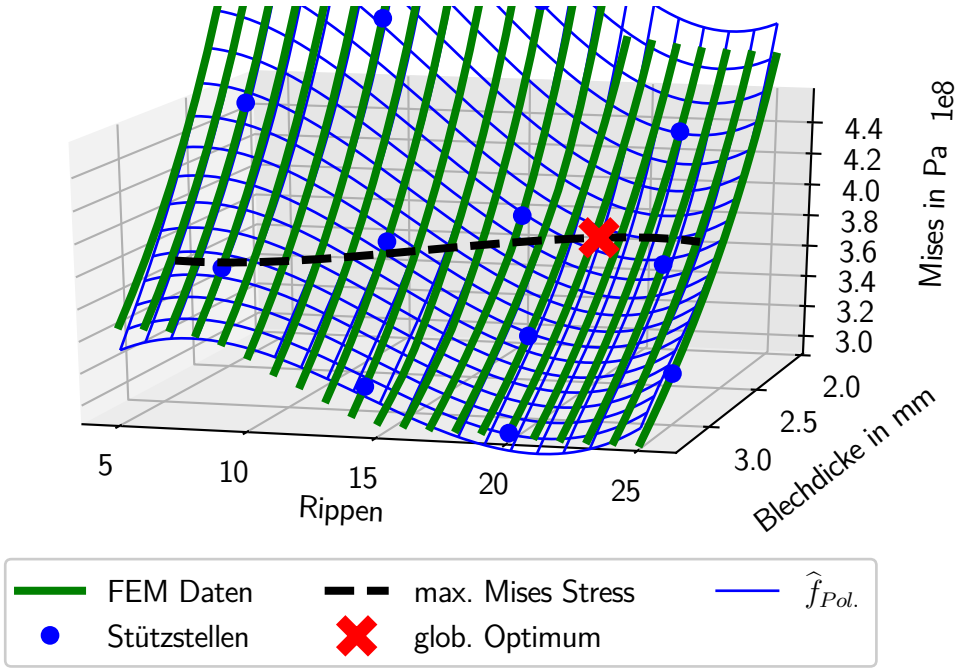

Abb. 4.4: Vergleich FEM-Daten und Surrogate Modell, hier: Polynom-Ansatz und Latin Hyper Cube mit 14 Stützstellen

Von jedem Surrogate Model kann eine Grafik, wie sie in Abb. 4.4 zu sehen ist, abgeleitet werden. Es ist in grünen Linien zunächst das tatsächliche Ergebnis der FEM-Raster-Messungen zu sehen. Darüber ist als blaues Netz ein Surrogate Model gelegt und mit blauen Punkten die Stützstellen markiert. Durch die in Abb. 3.7 beschriebene Optimierung ergibt sich die schwarz-gestrichelte Optimum-Linie, mit dem globalen Optimum als rotes Kreuz dargestellt. Analog dazu sind die Optimum Linien mit globaler Optimum-Markierung für die verschiedenen Surrogate-Methoden in Abb. 4.5 gezeigt. Zudem ist dort die mit dem Sekanten Verfahren gefundene Referenzlösung markiert. Es ist zu erkennen, dass alle Surrogate-Methoden auf eine Rippenanzahl von 18 kommen,

während der Referenzwert 17 beträgt. Die Ergebnisse für das optimale Gewicht sind dennoch nahe beim Referenzgewicht. Die Zahlenwerte zu den Optima sind in Tabelle 4.4 aufgelistet.

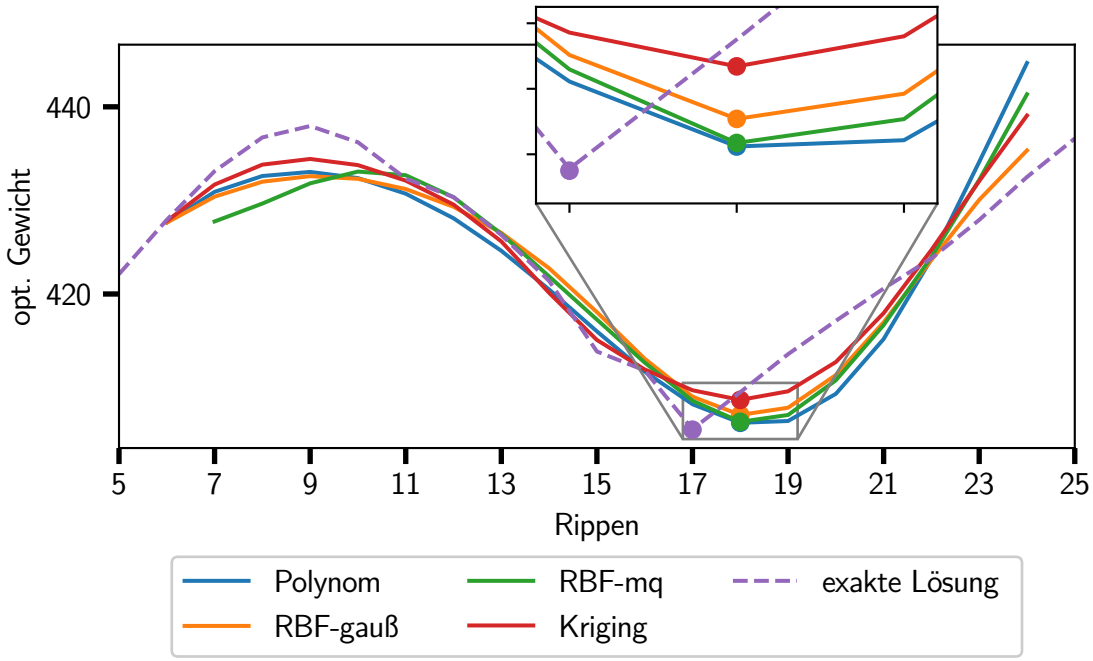

Abb. 4.5: Optimierungs-Linien innerhalb der Surrogate Modelle

Versuchsplan

Bei den Versuchsplänen ist von Interesse wie viele Stützstellen mit der jeweiligen Methode benötigt werden, um eine möglichst präzises Surrogate Model darauf zu bilden. Daher werden mit allen in Kapitel 2.2.1 beschriebenen Versuchsplan-Methoden Surrogate Modelle unter Verwendung verschiedener Anzahlen von Stützstellen gebildet und untersucht. Von den trainierten Modellen werden die PRESS und die RMSE berechnet und in Prozent zu der maximal zulässigen von Mises Spannung ausgedrückt. Die Ergebnisse sind in Abb. 4.6 zu sehen. Als Anhaltspunkt wurde das Kriterium gesetzt, dass der prozentuale RMSE unter einem Prozent liegen soll (grau-gestrichelte Linie). Es ist zu erkennen, dass der strukturierte Versuchsplan besonders bei niedriger Stützstellenanzahlen tatsächlich schlechter abschneidet als die anderen Versuchsplanverfahren und das unabhängig von der verwendeten Surrogate-Methode. Der RMSE sinkt dort erst mit 17 Stützstellen unter ein Prozent. Dies liegt vor allem daran, dass die Zahl der verwendeten Punkte nicht ohne Probleme kontinuierlich verändert werden kann. Ist der strukturierte Versuchsplan nicht quadratisch, also hat nicht $n = x^2$ mit $x \in \mathbb{N}$ Punkte, so reichen diese nicht aus, um den gesamten Designraum zu füllen (es bleibt eine Ecke leer). Bei dem Polynom-Ansatz kommt dies besonders zum Tragen, da dort ohnehin die Modellränder dazu neigen sich abzulösen, was

dann die gesamte Approximation verzerren kann (overfitting bzw. underfitting, siehe Kapitel 2.2.2).

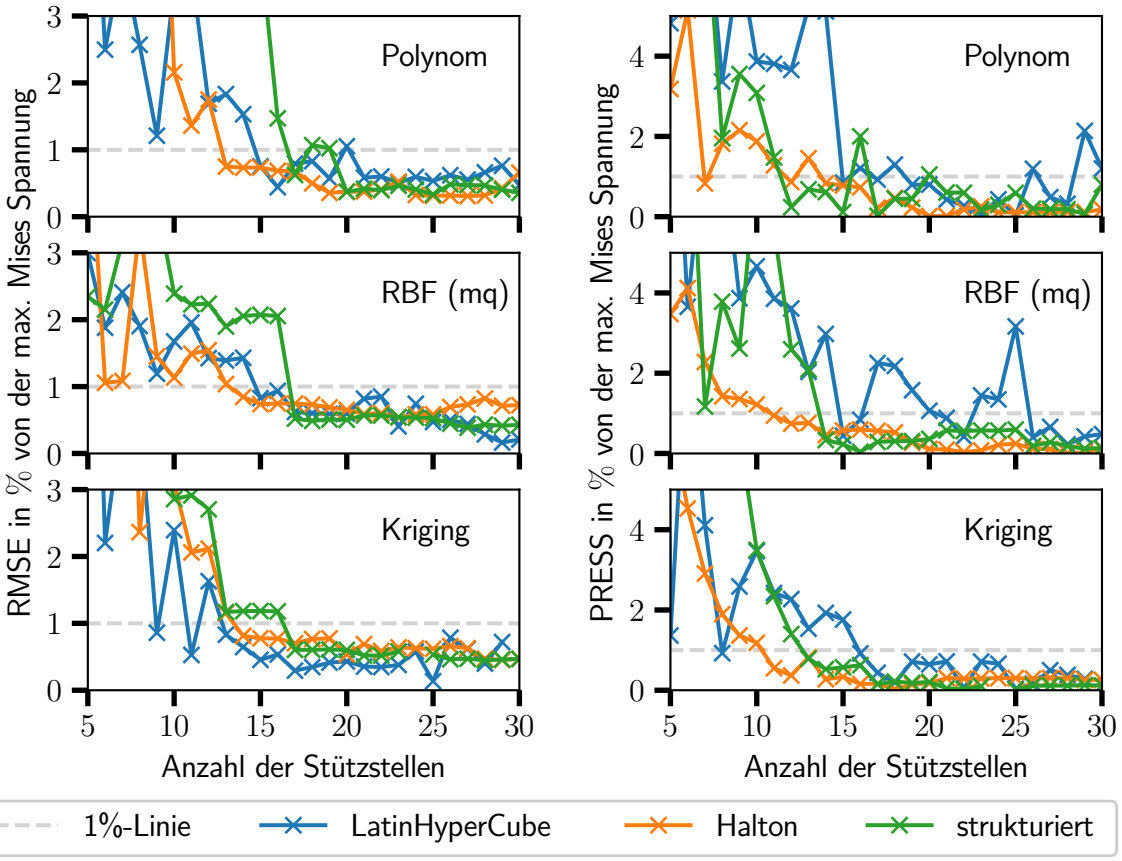

Abb. 4.6: Vergleich der Surrogate Modelle (PRESS und RMSE) mit verschiedenen Versuchsplänen

Beim Vergleich der RMSE zwischen Latin Hyper Cube und Halton fällt auf, dass Halton ab etwa 14 Stützpunkten keine Sprünge mehr macht, während Latin Hyper Cube mehr Variation auch mit vielen Stützstellen zeigt. Noch mehr kommt dies bei dem Verlauf der PRESS zum tragen. Es zeigt sich, dass die Latin Hyper Cube Versuchspläne instabil sind. Dies liegt daran, dass Latin Hyper Cube an ein Raster gebunden ist, welches je nach Anzahl der Stützstellen deutliche Struktur in den Versuchsplan bringt. Halton hingegen ist in seiner Variation der Stützstellenanzahl sehr kontinuierlich und auch die Verteilung erfolgt flächendeckend, sodass das Surrogate Modell auf einzelne Punkte verzichten kann (niedriger Wert für PRESS). Dies gilt nur für eine gute Wahl der Basen (hier 2 und 19), welche nach optischen Kriterien gewählt wurden (vgl. Kapitel 2.2.1 und Abb. 2.5). Aufgrund seiner Stabilität wird für den späteren, detaillierten Vergleich der Surrogate-Methoden Halton als Versuchsplan-Verfahren zum Einsatz kommen.

Polynom

Der Polynom-Ansatz lieferte auf Anhieb gute Ergebnisse. Auffällig ist, dass er das Modell nicht nur gut approximiert sondern auch glättet. In Abb. 4.5 ist der stetig, differenzierbare Verlauf der Optimum-Linie zu erkennen (die leichten Knicke resultieren aus der niedrigen Auflösung des Plots, pro Rippenanzahl ein Punkt), welche gleichzeitig ein exemplarischer Schnitt durch die Surrogate-Funktion ist.

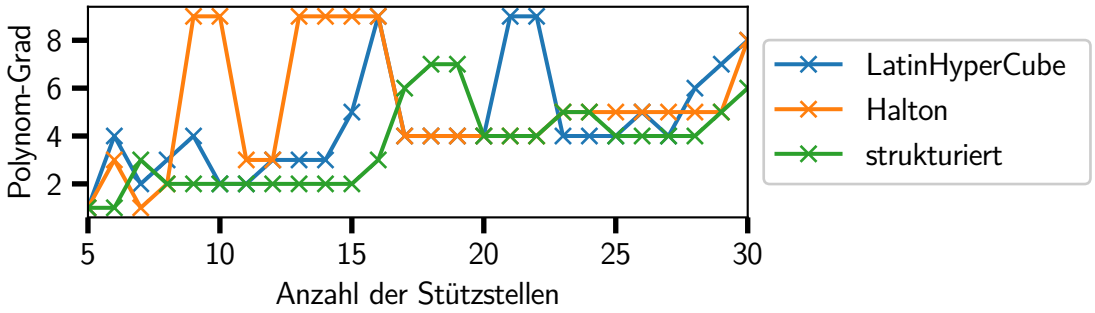

Abb. 4.7: Polynom-Grad in Abhängigkeit der Stützstellenanzahl

Der Grad der Polynome wurde mithilfe des im Kapitel 2.2.3 beschriebenen Verfahrens bestimmt. Zu beachten ist, dass der Grad nicht nur von der Form der Funktion abhängig ist. Er wird ebenfalls stark von der Anzahl der Stützstellen beeinflusst. Tendenziell steigt der Grad mit zunehmender Stützstellenanzahl. Abb. 4.7 zeigt diesen Verlauf für die verschiedenen Versuchsplan-Methoden. Die Sprünge kommen durch die quasi stochastischen Versuchsplan-Prinzipien zustande. Der weniger streuende, strukturierte Versuchsplan hat nur wenige Aussetzer. Die hohen Polynom-Grade, die bei Halton im Bereich unter 17 Stützstellen auftreten, sprechen für ein overfitting. Die Lage der Validierungspunkte ist für die Optimierung des Polynom-Grads sehr wichtig. Es sollten auch Punkte im Randbereich gewählt werden, da gerade dort die Polynome dazu neigen sich von der Zielfunktion abzulösen. Es wurde das in Abb. 3.8 dargestellte Validierungspunkte-Muster für das Training des Polynom-Grads verwendet. Verglichen wurde der MAE, da dieser bessere Ergebnisse lieferte als die Verwendung des RMSE.
An den Designraumrändern lösen die Approximationen mit dem Polynom-Ansatz sofort von der berechneten Lösung ab. Dies stellt dann ein Problem dar, wenn beispielsweise eine Ecke des Designraums von keiner Stützstelle begrenzt wird. Mit hoher Wahrscheinlichkeit wird sich diese Ecke dann nach oben oder unten wölben. Es sollte also bei der Verwendung von Polynomen darauf geachtet werden, den Designraum großzügig zu dimensionieren, um mit der Optimierung nicht in die Randbereiche zu kommen.

RBF

Die ersten Versuche das Flügelkastenproblem mit auf RBF-basierenden Surrogate Modellen nachzustellen, zeigten lediglich eckige Verläufe, die dem eigentlichen Model nicht folgten. Zu erkennen war, dass ein klarer Unterschied der Anpassung zwischen Rippen und Blechdicke bestand. In Richtung der Blechdicke war der Funktionswert nahezu konstant, während sich in Rippenrichtung große Wellen bildeten. Dies liegt in der Natur der RBF, die mit den Radien bzw. den Abständen zwischen den Stützstellen arbeiten. Wird die Entfernung zwischen zwei Stützpunkten gemessen, so ist die Rippenanzahl immer die dominante Größe, da diese um etwa den Faktor 1000 größer ist (das Model wird in SI-Einheiten berechnet, also ist die Blechdicke in Metern angegeben). Deshalb wurde eine Skalierung der Eingänge eingeführt. Rippenanzahl und Blechdicke werden vor der Berechnung des RBF-Surrogate Models auf den Wertebereich [0..1] skaliert. Dabei wird je ein Offset und ein Faktor verwendet. Durch diese Anpassung gelingt die Approximation. Die meisten RBF bieten die Möglichkeit mit einem Parameter die Anpassung zu verbessern (siehe Tabelle 2.4). Dieser muss herausgefunden werden, was durch eine kleine Optimierung geschieht. Mit dem Gradient-basierten Optimierer SLSQP wird der Parameter a variiert, bis der RMSE minimal ist. Zur Ermittlung des RMSE kommen, wie auch schon beim Polynom-Ansatz, die in Abb. 3.8 dargestellten Punkte zum Einsatz. Diese Optimierung geschieht sehr schnell, da sie im Durchschnitt lediglich 10 Iterationen benötigt, um zu konvergieren und RBF-Surrogate-Modelle einen einfachen Trainingsalgorithmus haben (vgl. Kapitel 2.2.2). Bei Variation der Stützstellenanzahl muss auch a angepasst werden. Hier kann jedoch kein Zusammenhang festgestellt werden, wie es bei den Polynomen und ihrem Grad der Fall ist. Die in Tabelle 2.4 aufgelisteten RBF wurden ausprobiert und liefern sehr verschiedene Ergebnisse. Dies erschwert die Anwendung. In Abb. 4.5 sind exemplarisch für die gaußsche und die multiquadratische RBF die Gewichtsverläufe über der Rippenzahl bei exakter Einhaltung der maximalen von Mises Spannung dargestellt. Es ist zu erkennen, dass die multiquadratische RBF das Gewicht sehr präzise vorhersagt. Auch mit der gaußschen RBF werden gute Ergebnisse erzielt. Stabile Ergebnisse werden bei diesem Beispielproblem mit a im Bereich [0.05..10] erreicht. Werden die einfachen RBF ohne Parameter verwendet, wird die Approximation deutlich gröber. Gute Ergebnisse stellen sich dabei erst bei der Verwendung von etwa 25 Stützstellen ein.
Das Python-Paket scipy enthält eine Implementation des RBF-Surrogate Modells. Es liefert die selben Ergebnisse, was die eigene Implementation validiert.

Kriging

Kriging liefert gute Ergebnisse, vorausgesetzt der Optimierer findet die minimale negative logarithmische Wahrscheinlichkeit. Dies erfordert wie bereits in Kapitel 3.2.2 beschrieben etwas

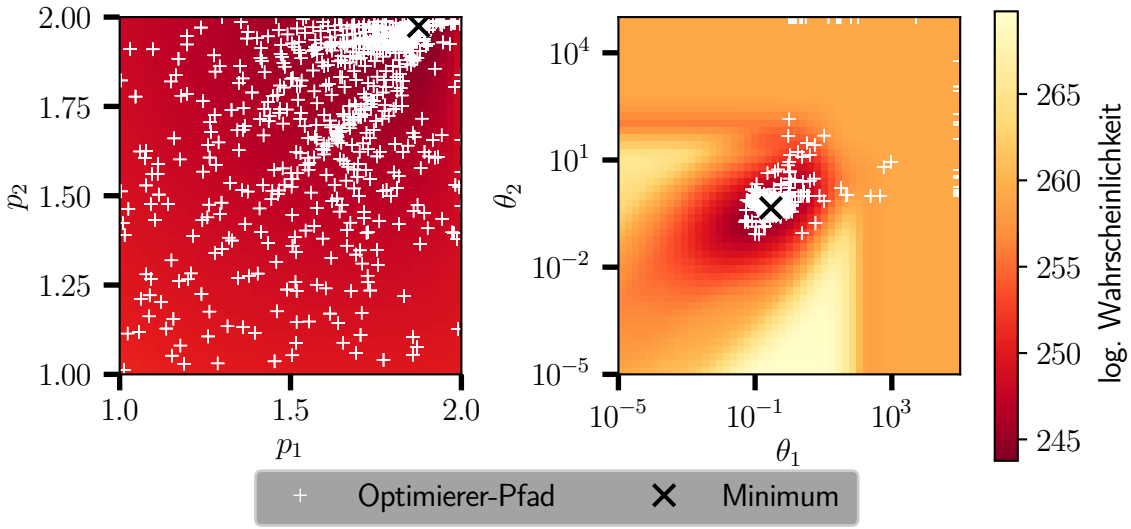

Abb. 4.8: Optimierung der log. Wahrscheinlichkeit mit Basin-hopping, 6468 Versuche

Aufwand. Abb. 4.8 zeigt die Optimierung mit dem Basin-hoppin-Algorithmus. Das vierdimensionale Problem wurde der Übersicht halber auf zwei zweidimensionale Plots aufgeteilt. Die Optimierung jedoch muss vierdimensional erfolgen, da alle beteiligten Eingangsgrößen große Wechselwirkungen zeigen. Farbig ist die logarithmische Wahrscheinlichkeit markiert. Der Verlauf von p_1 und p_2 ist exemplarisch für optimale θ_1 und θ_2 dargestellt. Der Verlauf für p ändert sich bedeutend mit anderen θs. Der θ-Verlauf wiederum gilt nur für die optimalen ps. Überlagert sind die Grafiken mit weißen Kreuzen, die immer dort aufgetragen sind, wo der Optimierer eine Berechnung durchgeführt hat. Mit schwarzem Kreuz markiert ist jeweils das globale Optimum. Es ist zu erkennen, dass p_1 und p_2 weit streuen, während θ_1 und θ_2 längst nicht im gesamten Raum geprüft werden. Dies kann dazu führen, dass das globale Optimum übersehen wird. In den zahlreichen Versuchen, die während der Erstellung dieser Arbeit durchgeführt wurden, kam es sogar vor, dass Kriging sehr schlecht trainiert war, weil eben dies passierte. Durch die stochastische Komponente in Basin-hopping stellt dies gerade für automatisierte Berechnungen ein großes Problem dar.

Darum wurde später nur noch der ebenfalls in Kapitel 3.2.2 vorgestellte Raster-basierte Ansatz verwendet. Abb. 4.9 zeigt den Weg des Optimierers für diesen Ansatz analog zu Abb. 4.8. Es ist zu erkennen, dass die Punkte für p weniger streuen, jedoch den gesamten Bereich erfassen. Für θ wird ebenfalls der gesamte Raum abgedeckt. Es gibt keine unnötigen Versuche fernab eines Minimums. Der Optimierer geht von dem besten Kandidaten aus dem Raster zielstrebig zum globalen Minimum. Das Raster für p ist gröber als das für θ, da p bei allen Versuchen lediglich ein lokales Minimum hat, welches somit dem globalen entspricht. Bei diesem Beispiel besteht das Raster allein bereits aus 1600 Evaluationen. Jedoch werden für die eigentliche Optimierung

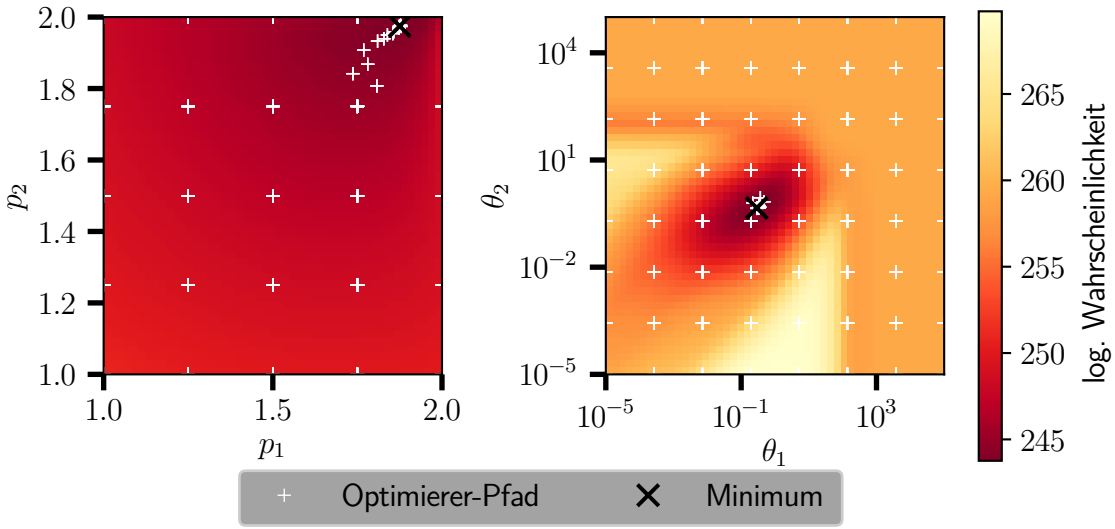

Abb. 4.9: Optimierung der log. Wahrscheinlichkeit mit dem Raster-Ansatz, 1718 Versuche

im Anschluss nur noch 118 Versuche benötigt.
Der stochastisch arbeitende Basin-hopping-Algorithmus findet manchmal das Optimum nach nur wenigen Versuchen, im Schnitt werden jedoch, wie in Abb. 4.8 dargestellt, über 6000 Evaluationen benötigt. Der Raster-basierte-Algorithmus wird für dieses Surrogate Model immer genau 1718 Evaluationen benötigen und jedes mal das globale Minimum finden (vgl. Abb. 4.9).
Die Kriging-Optimum-Linie in Abb. 4.5 liegt etwas weiter von der Referenzlösung entfernt als die zuvor beschriebenen Verfahren. Jedoch ist auch zu bemerken, dass die Approximation im Gesamten gut passt. Besonders bei dem lokalen Maximum auf der linken Seite des Plots ist die Abweichung von den Referenzwerten am geringsten.

Da es ein Python-Paket namens PyKriging gibt, welches Kriging implementiert, wurde dieses ebenfalls angewandt. Die Ergebnisse entsprechen genau denen des in dieser Arbeit implementierten Algorithmus. Die Laufzeit von PyKriging ist jedoch sehr viel schlechter. Das in Abb. 4.9 gezeigte Beispiel benötigt mit dem, in dieser Arbeit erstellten Algorithmus 0,85 Sekunden, während die Optimierung von PyKriging im Durchschnitt über 5 Sekunden benötigt und dies obwohl PyKriging lediglich den θ-Bereich $[10^{-5}..10^{2}]$ untersucht. Es wird dort auf einen stochastischen evolutionären Algorithmus für die Optimierung der logarithmische Wahrscheinlichkeit gesetzt (wahlweise inspyred.ec.GA oder inspyred.swarm.PSO). [7] [1]

4.4 Exakte Lösung

Abb. 4.10 wurde mit dem in Kapitel 3.2.3 beschriebenen Algorithmus generiert. Sie zeigt für jede Rippenzahl die benötigte Blechdicke und das Gewicht, wenn die maximale von Mises Spannung jeweils exakt dem vorgegebenem Maximum entspricht. Es ist zu erkennen, dass das minimale Gewicht bei 17 Rippen erreicht wird. Das Sekantenverfahren stellte sich, zumindest verglichen mit MDAO, als sehr effizient heraus, da es pro Rippe nur etwa 7 Evaluationen der FEM-Berechnung benötigt, um das Optimum zu bestimmen (mit einer Toleranz von $1.48e-08$).

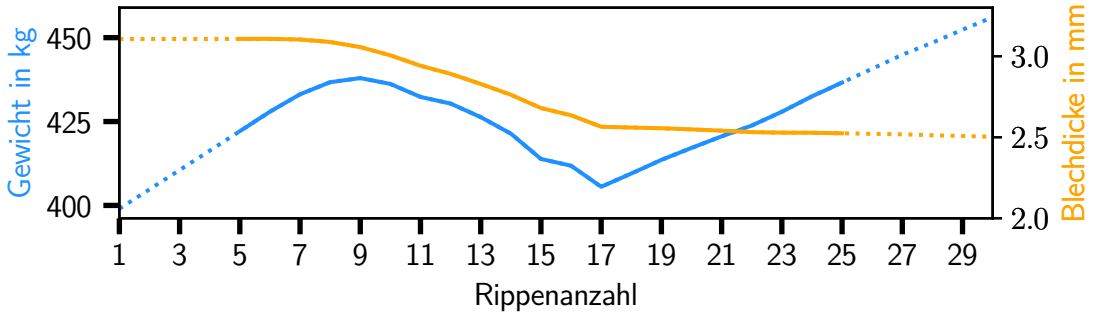

Abb. 4.10: Optimierung der FEM-Rechnung mit dem Sekantenverfahren

Um das Abfallen des Gewichts mit sehr niedrig werdender Rippenzahl zu untersuchen, ist der Graph in Abb. 4.10 bis hin zu einer einzelnen Rippe erweitert. Das Gewicht nimmt hier stark ab, da bei weniger als neun Rippen die benötigte Blechdicke fast konstant bleibt. Dies liegt an den großen Vereinfachungen, die in Kapitel 3.2 getroffen wurden. Es wird kein Beulen betrachtet und die Kraft wirkt rein senkrecht auf dem Balken. Da dies nicht realistisch ist und eine gewisse Anzahl von Rippen allein für Anbauten wie Triebwerke und Klappen Voraussetzung ist, wird die minimale Anzahl von Rippen auf fünf gesetzt. Auf der rechten Seite von Abb. 4.10 ist zu erkennen, dass dort ab etwa einer Rippenzahl von 17 die Blechdicke offenbar ihr Minimum erreicht und von dort an konstant bleibt. Zusätzliche Rippen erhöhen also das Gewicht etwa linear. Aus diesen Erkenntnissen wurden die Designraum-Grenzen abgeleitet, die für alle weiteren Versuche genutzt werden. Für die Rippenzahl sind diese [5 .. 25] und für die Blechdicke [2.0 .. 3.3]mm.

4.5 Vergleich der Verfahren

In Tabelle 4.4 sind die Ergebnisse der vorgestellten Optimierungsverfahren zusammengefasst. Die Anzahl der benötigten Stützstellen der Surrogate Modelle entstammt der Abb. 4.6 und ist jeweils so gesetzt, dass die prozentuale PRESS unter 0,5% ist.

	Iter. gesamt	Rippenanzahl	Blechdicke	max. Mises	Gewicht
exakte Lösung	141	17	2,565 mm	3,81e+8 Pa	405,5 kg
MDAO SLSQP	20 (×2)	18,5	2,45 mm	3,98e+8 Pa	393,8 kg
MDAO ALPSO	80 (×2)	17,1	2,56 mm	3,81e+8 Pa	407,3 kg
Polynom	17 (+5)	18	2,54 mm	3,81e+8 Pa	406,2 kg
RBF (gauß)	14 (+5)	18	2,54 mm	3,81e+8 Pa	407,1 kg
RBF (mq)	20 (+5)	18	2,54 mm	3,81e+8 Pa	406,3 kg
Kriging	16	18	2,55 mm	3,81e+8 Pa	408,7 kg

Tabelle 4.4: Vergleich von Ergebnissen der verschiedenen Verfahren

An der exakten Lösung gemessen, schneidet der Polynom-Ansatz, trotz seines einfachen Aufbaus, am besten ab. An zweiter Stelle steht der multiquadratische RBF-Algorithmus. Dieser benötigt jedoch eine hohe Anzahl an Stützstellen. Der gaußsche Ansatz ist nicht mehr ganz so genau, benötigt jedoch dafür die geringste Anzahl von Stützstellen. Das Kriging Surrogate Modell lässt sich nur schwer mit den Anderen vergleichen. Kriging benötigt, abgesehen von der Validierung, keine zusätzlichen Punkte für das Training. Hier werden sowohl von dem Polynom- als auch dem RBF-Ansatz zusätzliche Evaluationen der hochauflösenden Berechnung benötigt, die jedoch nicht direkt in das eigentliche Surrogate Modell einspielen und später auch nicht als Validierungspunkte wiederverwendet werden können. Diese Trainingspunkte (in Tabelle 4.4 mit (+5) gekennzeichnet da hier fünf Punkte dafür verwendet wurden) erhöhen die Rechenzeit dieser Verfahren, mit weiteren Auswertungen der hochauflösenden Funktion, was Kriging zur schnellsten Surrogate-Methode macht. Das Optimierungsziel von Kriging zeigt zwar die größte Abweichung, der Verlauf der gesamten Approximation ist jedoch sehr präzise (vgl. Abb. 4.5). Dies wird in Tabelle 4.5 bestätigt. Dort sind für die in Tabelle 4.4 dargestellten Ergebnisse die Fehler angegeben. Ausgewertet wurden diese an den in Abb. 3.8 gezeigten Punkten, punktsymmetrisch gespiegelt. Dies ist notwendig, da der Polynom- und RBF-Ansatz anhand der ungespiegelten Punkte trainiert ist und somit nicht mehr unabhängig von diesen. Aussagekräftig sind besonders der RMSE und der MAE, welche beide für das Kriging Modell am geringsten sind. Auch die prozentuale Abweichung, gemessen an dem Raster aus FEM-Lösungen, ist am geringsten, was vor allem daran liegt, dass Kriging im Gegensatz zu den anderen Verfahren auch im Randbereich des Designraums noch gut an der tatsächlichen Lösung anliegt. In Tabelle 4.5 sind neben den Fehlern auch die Programmlaufzeiten notiert, die zum erstellen und

trainieren der jeweiligen Surrogate Modelle nötig sind. Mit dem simplen Beispielproblem benötigt Kriging mit der aufwändigen Optimierung der logaritmischen Wahrscheinlichkeit etwa eine Sekunde. Mit Abstand am schnellsten ist der Polynom-Ansatz. Wie in Kapitel 2.2.2 erwähnt, können diese Zahlen jedoch nicht auf umfangreiche Probleme übertragen werden, da dabei der Rechenaufwand der verschiedenen Surrogate-Methoden in unterschiedlichem Maß wächst. Kriging wird daher schnell sehr viel schlechter abschneiden, während die restlichen Methoden lediglich linear mit der Anzahl der Eingänge langsamer werden. Verglichen mit den Rechenzeiten der hochauflösenden Funktion, sollten die Trainingszeiten der Surrogate Modelle jedoch nicht ins Gewicht fallen.

	ØAbweichung	RMSE	MAE	PRESS	Trainingszeit
Polynom	1,88	0,68	0,96	0,22	0,01 sec.
RBF (gauß)	3,46	1,09	1,87	0,14	0,05 sec.
RBF (mq)	1,26	0,79	1,32	0,11	0,10 sec.
Kriging	0,98	0,57	0,86	0,23	1,06 sec.

Tabelle 4.5: Prozentuale Abweichung, Fehler und Trainingszeit der Surrogate Modelle

Mit dem OpenMDAO Paket und ALPSO konnte das Gewicht ebenso gut wie mit den Surrogate-Methoden bestimmt werden. Bei der Rippenanzahl wird sogar die Referenzlösung erreicht. Allerdings wurden dazu 80 Evaluationen benötigt. Da der in Kapitel 3.2.1 vorgestellte Algorithmus jedoch pro Evaluation eine lineare Interpolation zwischen zwei Rippenanzahlen vornehmen muss, um die Nachkommastelle zu beachten, ergeben sich $2 \times 80 = 160$ Evaluationen. Verglichen mit der Ermittlung der exakten Lösung (Sekantenverfahren) ist dies eine deutliche Verschlechterung. Es gelang nicht für den SLSQP Optimierer Einstellung zu finden, die diesen fehlerfrei konvergieren ließen und so ist das damit errechnete Ergebnis ungültig, da es nicht die maximal zulässige von Mises Spannung einhält. Zudem ist nicht klar, wie eine Rippenanzahl von 18,5 zu interpretieren ist. Dennoch ist bereits abschätzbar, dass deutlich mehr Auswertungen der FEM-Rechnung durchgeführt werden müssen, um ein gutes Ergebnis mit Hilfe von MDAO zu erzeugen.

Da die Rippenanzahl in diskreten Schritten variiert werden muss, lässt sich sagen, dass MDAO nicht die perfekte Werkzeug darstellt, weil dies zumindest von OpenMDAO nicht unterstützt wird. Auch für Surrogate Modelle ist der tatsächliche Funktionsverlauf, wie er in Abb. 4.10 dargestellt ist, nicht einfach nachzustellen. Ihre Stärke liegt in der Approximation von kontinuierlichen Funktionen. Der starke Knick bei 17 Rippen kann von keinem der untersuchten Verfahren nachgebildet werden, wie in Abb. 4.5 zu erkennen ist. Diese Unstetigkeit ist der Grund dafür, dass alle Surrogate-Methoden das Gewichtsminimum bei 18 Rippen finden. Sie können dem Knick nicht folgen und glätten die Zielfunktion an dieser Stelle (vgl. Abb. 4.5).

Kapitel 5

Fazit und Ausblick

Ziel dieser Arbeit war es darzustellen, ob Surrogate Modelle im Flugzeugvorentwurf verwendet werden können. Um diese Frage zu beantworten, mussten zunächst die Grundlagen besprochen werden. Es wurden verschiedene Methoden mathematisch beschrieben, die dabei helfen Surrogate Modelle zu erstellen. Diese wurden in der Programmiersprache Python implementiert und anhand einer Beispiel-FEM-Berechnung eines Dash 8 Flügelkastens eingesetzt. Dabei sollten die Anzahl der Rippen und die Blechdicke des Flügelkastens gefunden werden, die zwar den maximalen Belastungen standhalten, jedoch das geringste Gewicht aufweisen. Es wurde versucht, die verschiedenen Surrogate-Methoden zu kombinieren, sodass mit wenigen Auswertungen der FEM-Berechnung eine möglichst präzise Lösung ermittelt werden konnte.
Es sollte zudem untersucht werden, ob mit Surrogate Modellen im Vergleich zu den bereits etablierten MDAO-Verfahren Rechenzeit gespart werden kann. Um diese Methoden zu vergleichen, wurde die Lösung des Flügelkastenproblems mit einem MDAO-Programm bestimmt. Dabei traten mit dem eigentlich vielversprechenden Gradient-basierten Verfahren Probleme in der Einstellung des Optimierers auf, sodass ein auf Stochastik basierender Optimierer verwendet wurde. Dieser kommt zum richtigen Ergebnis, benötigt jedoch sehr viele Versuche. Hier fehlt Erfahrung mit MDAO-Verfahren, was den objektiven Vergleich erschwert.
Es wurde untersucht, ob DoE dabei helfen kann, die Erstellung eines Surrogate Modells vorzubereiten. DoE eignet sich dafür, die Eingänge mit wenig oder keinem Einfluss auszuschließen und die Abhängigkeiten zwischen den Eingängen zu untersuchen. Zudem kann überprüft werden, ob ein Minimum (bzw. das Optimierungsziel) innerhalb der gewählten Designraumgrenzen liegt. Die Anzahl der für das Surrogate Modell benötigten Stützstellen kann mit DoE jedoch nicht bestimmt werden, da diese nicht nur von der Form der Zielfunktion, sondern auch vom verwendeten Versuchsplan-Verfahren und von der Surrogate-Methode abhängen. Daher spielt die Validierung des fertigen Surrogate Modells eine wichtige Rolle. Am aussagekräftigsten sind der RMSE (root mean square error) um die Genauigkeit der Approximation zu bestimmen und

PRESS (prediction sum of square), die Aussage darüber gibt, ob genügend Stützstellen genutzt wurden.
Um die Stützstellen zu generieren, mit denen das Surrogate Modell gebildet werden soll, wurden ein Vollfaktorplan mit dem Latin Hyper Cube- und dem Halton-Verfahren verglichen. Dabei stellte sich heraus, dass gerade bei wenigen Stützstellen der Vollfaktorplan deutlich mehr Abweichungen in der Approximation hervorruft. Latin Hyper Cube hingegen ermöglicht zwar teilweise auch mit sehr wenigen Stützstellen ein gut trainiertes Surrogate Modell, ist jedoch unzuverlässig. Dies äußert sich daran, dass bei steigender Anzahl von Stützstellen Ausreißer mit schlechter Verteilung und somit ungenauer Approximation auftreten. Mit dem Halton-Verfahren konnten Versuchspläne generiert werden, die ebenfalls bei wenigen Stützstellen gute Ergebnisse liefern und stabil bleiben, wenn die Stützstellenanzahl weiter erhöht wird. Deshalb wurde zum Vergleich der Surrogate-Methoden auf Halton zurückgegriffen. Gemessen an der Referenzlösung des Flügelkastenproblems, die für jede Rippenanzahl einzeln mit dem Sekantenverfahren direkt von der FEM-Berechnung ermittelt wurde, schneidet das Polynom-Surrogate Modell am besten ab. Dies ist beachtlich, da dieses sehr simpel aufgebaut ist. Auch der RBF- und Kriging-Ansatz liefern sehr gute Ergebnisse.
Eine weitere Frage, die in dieser Arbeit beantwortet werden sollte ist die, welche Probleme bei der Umsetzung von Surrogate Modellen auftreten können und wie mit diesen umzugehen ist. Dies sollte mit dem in dieser Arbeit implementierten Flügelkastenproblem untersucht werden. Das konstruierte Problem ist sehr stark vereinfacht und hat dadurch lediglich zwei Eingänge. Die Auswertung der Surrogate Modelle ist damit eher subjektiv. Besonders die Tatsache, dass das Kriging-Surrogate Modell das Optimum am ungenauesten vorhersagt, muss in anderen Anwendungen nicht zutreffen. Es ging in dieser Arbeit vor allem um die Erfahrungen, die während der Umsetzung der Verfahren gewonnen wurden und auf jede Anwendung zutreffen.
Der Polynom-Ansatz ist in seinem simplen Aufbau den anderen Verfahren überlegen. Zudem ist es möglich, das gefundene Polynom in Textform als einfache Gleichung auszugeben, welche dann gespeichert oder in andere Programme überführt werden kann. Zu beachten ist, dass der Polynom-Grad eingestellt werden muss, was über eine Fehlerbetrachtung automatisiert erfolgen kann. Der maximale Polynom-Grad ist nicht nur abhängig von der Form der approximierten Funktion, sondern auch von der Anzahl der verwendeten Stützstellen. Eine weitere Anwendung des Polynom-Ansatzes, neben der Interpolation, ist die Glättung von Daten, da er sehr glatte Funktionen generiert.
Bei dem auf RBF-basierenden Surrogate Modell muss eine einheitliche Skalierung der Eingänge unbedingt beachtet werden. Ohne diese funktioniert das Verfahren prinzipbedingt nicht. RBF-Surrogate Modelle bieten Einstellmöglichkeiten über die Auswahl der RBF und oft auch

über einen frei wählbaren Parameter. Dies kann nützlich sein, wenn das Modell manuell eingestellt wird. In der Praxis zeigte sich, dass die Wahl des Parameters von der Form der Zielfunktion, sowie von dem Versuchsplan abhängig ist. Daher wird in der automatisierten Anwendung eine Optimierung des Einstellparameters anhand einer Fehlerbetrachtung benötigt.
Kriging ist zwar kompliziert in der Implementation und hat mit Abstand den größten Rechenaufwand beim Trainieren, liefert jedoch sehr gute Ergebnisse, ohne dass Parameter manuell angepasst werden müssen oder zusätzliche Trainingspunkte benötigt werden. Dies ist besonders nützlich für unerfahrene Benutzer oder wenn keine Kenntnisse über den ungefähr erwarteten Verlauf der Funktion vorhanden sind. Der Fehler der Kriging-Surrogate Modelle war in allen Versuchen geringer als der der anderen Surrogate-Methoden.
Das konstruierte Beispielproblem warf besonders bei der Lösung mittels MDAO Probleme auf. Die Rippen müssen in diskreten Schritten variiert werden, was zumindest von der verwendeten MDAO-Software (OpenMDAO) nicht unterstützt wird. Für die Surrogate Modelle stellte dies kein Problem dar, was deren Potential für den Flugzeugvorentwurf zeigt, da viele Parameter wie Anzahl der Triebwerke oder Sitze in einer Reihe diskret sind.

Mit dem Beispielproblem konnten grundlegende Eigenschaften von Surrogate Modellen untersucht werden. Jedoch ist dies nicht exemplarisch für große Probleme mit einer Vielzahl von Eingängen. Hierzu sind weitere Versuche nötig und die Rangfolge des besten Verfahrens kann sich deutlich ändern. Besonders das komplizierte Optimierungsverfahren, welches in Kriging benötige wird, kann die Laufzeit bei vielen Eingängen unzumutbar machen. Auch die Nutzung von DoE wird bei großen Problemen eine wichtigere Rolle spielen, um die Zahl der Eingänge zu reduzieren und Wertebereiche zu bestimmen. Die Optimierung auf dem trainierten Surrogate Modell wird ebenfalls komplexer, was die Anwendung von MDAO-Software auf einem Surrogate Modell interessant machen kann. Ebenso kann untersucht werden, inwiefern sich ein Surrogate Modell innerhalb einer MDAO-Komponente eignet. Besonders mit dem Polynom-Ansatz wäre es ohne großen Aufwand möglich, das generiert Polynom im Quelltext eines größeren Problems abzuspeichern und als eigenständige Komponente einzubeziehen.
Ein Verfahren, welches Potential haben kann, jedoch für diese Arbeit zu weit geführt hätte, ist Co-Kriging. Dabei wird der Kriging-Algorithmus um Daten aus einem simplen (meist statistischen) Modell erweitert. Damit ist es möglich, als Datenbasis ein ungenaues aber leicht zu berechnendes Modell zu verwenden und dieses mit den Ergebnissen einer hochauflösenden Funktion zu präzisieren. Dabei werden deutlich weniger Stützstellen der zeitaufwendigen Berechnung benötigt.
Die Methodik des sequentiellen Versuchsplans hat ebenso Potential und sollte weiter untersucht werden. Damit kann mit wenigen Stützstellen der Bereich, in dem das Optimum liegen kann, weiter eingegrenzt und iterativ mit mehr Stützstellen präzisiert werden.

Literaturverzeichnis

[1] Aaron Garrett: *Library Reference — inspyred 1.0 documentation.* https://pythonhosted.org/inspyred/reference.html. Version: 2018. – (Zugriff: 24.10.2018)

[2] Agbagni, Dylan: *G-PRPD, Bombardier Dash 8 Q400.* https://commons.wikimedia.org/wiki/File:G-PRPD_Bombardier_Dash_8_Q400_FlyBe_(42248836532).jpg. Version: 2018. – (Zugriff: 05.09.2018)

[3] Ambrosius, Frank: Interpolation of 3D Surfaces for Contact Modeling. (2005), Nr. 1594

[4] Antony, Jiju: *Design of Experiments for Engineers and Scientists.* 2003. – 156 S. – ISBN 0750647094

[5] Bhosekar, Atharv ; Ierapetritou, Marianthi: Advances in surrogate based modeling, feasibility analysis, and optimization: A review. In: *Computers and Chemical Engineering* 108 (2018), 250–267. http://dx.doi.org/10.1016/j.compchemeng.2017.09.017. – DOI 10.1016/j.compchemeng.2017.09.017. – ISSN 00981354

[6] Breitkopf, Piotr ; Coelho, Rajan F.: *Multidisciplinary design optimization in computational mechanics.* 2010. – 549 S. http://dx.doi.org/10.1002/9781118600153. http://dx.doi.org/10.1002/9781118600153. – ISBN ISBN–13: 978–1–84821–138–4

[7] Capaulson-Community: *pyKriging.* http://pykriging.com/. Version: 2018. – (Zugriff: 24.10.2018)

[8] Chen, CS ; Hon, Y.C. ; Schaback, R.A.: Scientific computing with radial basis functions. (2005), 266. http://num.math.uni-goettingen.de/schaback/SCwRBF.pdf

[9] Chowdhury, Souma ; Mehmani, Ali ; Zhang, Jie ; Messac, Achille: Quantifying Regional Error in Surrogates by Modeling its Relationship with Sample Density. In: *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (2013). http://dx.doi.org/10.2514/6.2013-1751. – DOI 10.2514/6.2013–1751. – ISBN 978–1–62410–223–3

[10] Cominetti, Alberto: Surrogate Modeling with PythonTM. (2017), 1–17. http://www.dicat.unige.it/jpralits/AF2016/Relazione_Cominetti.pdf

[11] Cressie, Noel: Spatial prediction and ordinary kriging. In: *Mathematical Geology* 20 (1988), Nr. 4, S. 405–421. http://dx.doi.org/10.1007/BF00892986. – DOI 10.1007/BF00892986. – ISBN 0882–8121

[12] Dababneh, Odeh ; Kipouros, Timoleon: A review of aircraft wing mass estima-

tion methods. In: *Aerospace Science and Technology* 72 (2018), 256–266. http://dx.doi.org/10.1016/j.ast.2017.11.006. – DOI 10.1016/j.ast.2017.11.006. – ISSN 12709638

[13] Edwards, Jeffrey R. ; Shipp, Abbie J.: Polynomial Regression and Response Surface Methodology. In: *Perspectives on Organizational Fit* (2012), S. 209–258. http://dx.doi.org/10.4324/9780203810026. – DOI 10.4324/9780203810026. – ISBN 9780203810026

[14] Forrester, Alexander I. J. ; Sbester, Andrs ; Keane, Andy J.: *Engineering Design via Surrogate Modelling.* 2008. http://dx.doi.org/10.1002/9780470770801. http://dx.doi.org/10.1002/9780470770801. – ISBN 9780470770801

[15] Forrester, Alexander I. ; Keane, Andy J.: Recent advances in surrogate-based optimization. In: *Progress in Aerospace Sciences* 45 (2009), Nr. 1-3, S. 50–79. http://dx.doi.org/10.1016/j.paerosci.2008.11.001. – DOI 10.1016/j.paerosci.2008.11.001. – ISBN 0376–0421

[16] Garud, Sushant S. ; Karimi, Iftekhar A. ; Kraft, Markus ; Street, Pembroke: Design of Computer Experiments : A Review New Museums Site. In: *Computers and Chemical Engineering* 106 (2017), Nr. 182. http://dx.doi.org/https://doi.org/10.1016/j.compchemeng.2017.05.010. – DOI https://doi.org/10.1016/j.compchemeng.2017.05.010. – ISSN 00981354

[17] Giunta, Anthony A. ; Balabanov, V. ; Haim, D. ; Grossman, B. ; Mason, W. H. ; Watson, L. T. ; Haftka, R. T.: AIRCRAFT MULTIDISCIPLINARY DESIGN OPTIMIZATION USING DESIGN OF EXPERIMENTS THEORY AND RESPONSE SURFACE MODELING METHODS. In: *Aeronautical Journal* 101 (1997), Nr. May, 185. http://dx.doi.org/10.1.1.134.6732. – DOI 10.1.1.134.6732. – ISBN 0001–9240

[18] Gunst, Richard F.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments. In: *Technometrics* 38 (1996), Nr. 3, 284–286. http://dx.doi.org/10.1080/00401706.1996.10484509. – DOI 10.1080/00401706.1996.10484509. – ISBN 817808130X

[19] Han, Zhong-Hua ; Zhang, Ke-Shi: Surrogate-Based Optimization. In: *Real-World Applications of Genetic Algorithms* (2012). http://dx.doi.org/10.5772/36125. – DOI 10.5772/36125. – ISBN 9535101463

[20] Hürlimann, Florian: *Mass Estimation of Transport Aircraft Wingbox Structures with a CAD/CAE-Based Multidisciplinary Process*, Diss., 2010

[21] IAV: *Die Bibliothek der Technik Band 272: DoE - Design of Experiments.* 1. München : Verlag Moderne Industrie, 2004. – 72 S. – ISBN 978–3–937889–10–8

[22] Inc., Bombardier: Airport Planning Manual Q400. Version: 2014. http://dx.doi.org/10.1016/0016-3287(69)90006-8. 2014. – Forschungsbericht. – 114 S.. – ISSN 00163287

[23] Jacob, C ; Bieler, J ; Bardenhagen, A: INTRODUCING SURROGATE MODELS TO THE STRUCTURAL PRELIMINARY AIRCRAFT DESIGN PHASE. (2018)

[24] Jin, R. ; Chen, W. ; Simpson, T. W.: Comparative studies of metamodelling techniques under multiple modelling criteria. In: *Structural and Multidisciplinary Optimization* 23 (2001), Nr. 1,

S. 1–13. http://dx.doi.org/10.1007/s00158-001-0160-4. – DOI 10.1007/s00158–001–0160–4. – ISBN 1615–147X

[25] Kesseler, Ernst ; Guenov, Marin D.: *Advances in collaborative civil aeronautical multidisciplinary design optimization*. 2010 (233). – xvii, 436 p., 16 p. of plates S. http://app.knovel.com/web/toc.v/cid:kpACCAMDOK/viewerType:toc/root_slug:advances-in-collaborative. – ISBN 9781600867255\r1600867251

[26] Kis, Ivana M.: Comparison of Ordinary and Universal Kriging interpolation techniques on a depth variable (a case of linear spatial trend) , case study of the Šandrovac Field. In: *The Mining-Geology-Petroleum Engineering Bulletin* (2016), S. 41–58. http://dx.doi.org/10.17794/rgn.2016.2.4. – DOI 10.17794/rgn.2016.2.4

[27] Metals, ASM Aerospace S.: *ASM Material Data Sheet*. http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=ma7075t6. Version: 2018. – (Zugriff: 05.09.2018)

[28] Misra, Shobhit ; Darby, Mark ; Panjwani, Shyam ; Nikolaou, Michael: Design of Experiments for Control-Relevant Multivariable Model Identification: An Overview of Some Basic Recent Developments. In: *Processes* 5 (2017), Nr. 3, 42. http://dx.doi.org/10.3390/pr5030042. – DOI 10.3390/pr5030042. – ISSN 2227–9717

[29] Montgomery, Douglas C.: *Design and Analysis of Experiments*. 2017. – ISBN 9781119113478

[30] Neufeld, Daniel: MULTIDISCIPLINARY AIRCRAFT CONCEPTUAL DESIGN OPTIMIZATION CONSIDERING FIDELITY UNCERTAINTIES. (2010)

[31] OpenMDAO-Community: *Optimization of Paraboloid — OpenMDAO 2.4.0 Beta documentation*. http://openmdao.org/twodocs/versions/latest/basic_guide/first_optimization.html. Version: 2018. – (Zugriff: 16.10.2018)

[32] OpenMDAO-Cummunity: *OpenMDAO*. http://openmdao.org. Version: 2018. – (Zugriff: 16.10.2018)

[33] Perez, Ruben E.: *pyOpt*. http://www.pyopt.org/reference/optimizers.alpso.html. Version: 2011. – (Zugriff: 10.10.2018)

[34] Rafajłowicz, Ewaryst ; Schwabe, Rainer: Halton and Hammersley sequences in multivariate nonparametric regression. In: *Statistics and Probability Letters* 76 (2006), Nr. 8, S. 803–812. http://dx.doi.org/10.1016/j.spl.2005.10.014. – DOI 10.1016/j.spl.2005.10.014. – ISSN 01677152

[35] Rang, Joachim ; Heinze, Wolfgang: An Optimal Configuration of an Aircraft with High Lift Configuration Using Surrogate Models and Optimisation Under Uncertainties. In: *Advances in Structural and Multidisciplinary Optimization* (2018), Nr. June, 375–389. http://link.springer.com/10.1007/978-3-319-67988-4_29

[36] Ratcliff, Roger ; Smith, Philip L.: A Comparison of Sequential Sampling Models for Two-Choice Reaction Time. In: *Psychological Review* 111 (2004), Nr. 2, S. 333–367. http://dx.doi.org/10.1037/0033-295X.111.2.333. – DOI 10.1037/0033–295X.111.2.333. –

ISBN 0033–295X (Print)\n0033–295X (Linking)

[37] Raymer, Daniel P.: *Enhancing Aircraft Conceptual Design Using Multidisciplinary Optimization*. 2002. http://dx.doi.org/10.1.1.476.4103. http://dx.doi.org/10.1.1.476.4103. – ISBN 91–7283–259–2

[38] Rikards, R. ; Abramovich, H. ; Auzins, J. ; Korjakins, A. ; Ozolinsh, O. ; Kalnins, K. ; Green, T.: Surrogate models for optimum design of stiffened composite shells. In: *Composite Structures* 63 (2004), Nr. 2, S. 243–251. http://dx.doi.org/10.1016/S0263-8223(03)00171-5. – DOI 10.1016/S0263–8223(03)00171–5. – ISSN 02638223

[39] SciPy-Community: *minimize(method='SLSQP')*. https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html. Version: 2018. – (Zugriff: 10.10.2018)

[40] SciPy-Community: *scipy.optimize.basinhopping*. https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.basinhopping.html. Version: 2018. – (Zugriff: 17.10.2018)

[41] SciPy-Community: *scipy.optimize.differential_evolution*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html. Version: 2018. – (Zugriff: 17.10.2018)

[42] SciPy-Community: *scipy.optimize.newton*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html. Version: 2018. – (Zugriff: 17.10.2018)

[43] Sieberitz, Karl ; Van Bebber, David ; Hochkirchen, Thomas: *Statische Versuchsplanung DOE*. Bd. 1542. 2015. – 33–36 S. http://dx.doi.org/10.1017/CBO9781107415324.004. http://dx.doi.org/10.1017/CBO9781107415324.004. – ISBN 978–3–642–05492–1

[44] Simpson, W ; Peplinski, Jesse ; Koch, Patrick N.: METAMODELS FOR COMPUTER-BASED ENGINEERING SURVEY AND RECOMMENDATIONS. (2018)

[45] Tobergte, David R. ; Curtis, Shirley: *Radial Basic Function*. Cambridge University Press, 2013. – 1689–1699 S. http://dx.doi.org/10.1017/CBO9781107415324.004. http://dx.doi.org/10.1017/CBO9781107415324.004. – ISBN 9788578110796

[46] Torenbeek, Egbert: *Synthesis of subsonic airplane design*. SPRINGER-SCIENCE+BUSINESS MEDIA, B.V., 1979. http://dx.doi.org/10.1007/978-94-009-9580-2. http://dx.doi.org/10.1007/978-94-009-9580-2. – ISBN 978–94–009–9582–6

[47] Torenbeek, Egbert: *Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes*. 2013. – 1–425 S. http://dx.doi.org/10.1002/9781118568101. http://dx.doi.org/10.1002/9781118568101. – ISBN 9781118568101

[48] Triefenbach, Fabian: Design of experiments: the D-optimal approach and its implementation as a computer algorithm. In: *Bachelor's Thesis in Information and Communication ...* (2008), Nr. January. http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Design+of+Experiments+:+The+D-Optimal+Approach+and+Its+Implementation+As+a+

Computer+Algorithm{#}0. ISBN 9781118146927

[49] Van Velden, K: Surrogate Modelling for Airfoil Shape optimization. (2017)

[50] Viana, Felipe A. ; Haftka, Raphael T. ; Steffen, Valder: Multiple surrogates: How cross-validation errors can help us to obtain the best predictor. In: *Structural and Multidisciplinary Optimization* 39 (2009), Nr. 4, S. 439–457. http://dx.doi.org/10.1007/s00158-008-0338-0. – DOI 10.1007/s00158–008–0338–0. – ISBN 1615–147X

[51] Viana, Felipe A. ; Venter, Gerhard ; Balabanov, Vladimir: An algorithm for fast optimal latin hypercube design of experiments. In: *International Journal for Numerical Methods in Engineering* 82 (2010), Nr. 2, S. 135–156. http://dx.doi.org/10.1002/nme.2750. – DOI 10.1002/nme.2750. – ISBN 978–1–4577–0079–8

[52] Wang, G. G. ; Shan, S.: Review of Metamodeling Techniques in Support of Engineering Design Optimization. In: *Journal of Mechanical Design* 129 (2007), Nr. 4, 370. http://dx.doi.org/10.1115/1.2429697. – DOI 10.1115/1.2429697. – ISBN 0–7918–4255–X

[53] Wong, Tien-Tsin ; Luk, Wai-Shing ; Heng, Pheng-Ann: Sampling with Hammersley and Halton Points. In: *Journal of Graphics Tools* 2 (1997), Nr. 2, 9–24. http://dx.doi.org/10.1080/10867651.1997.10487471. – DOI 10.1080/10867651.1997.10487471. – ISSN 1086–7651

[54] Yongliang, Wu ; Jianping, Chen ; Wei, Wei ; Bin, Wang: Optimization research of stiffened shells based on kriging model and explicit FEM. In: *2017 8th International Conference on Mechanical and Aerospace Engineering, ICMAE 2017* (2017), S. 305–308. http://dx.doi.org/10.1109/ICMAE.2017.8038661. – DOI 10.1109/ICMAE.2017.8038661. ISBN 9781538633052

Anhang A

Anhang

A.1 Programmbeschreibung

Tabelle A.1 beschreibt die Python-Quellcode-Dateien, die für diese Arbeit genutzt wurden. Die Versuchsplan- und Surrogate-Klassen im Verzeichnis *mylibs* sind generisch und können auf jedes Problem angewandt werden. Im Verzeichnis *wingconstruction* befinden sich das Beispielprojekt, welches mit konkreten Werten das im Kapitel 3.2 beschriebene Flügelkastenproblem berechnet. Der einfachste Weg die Dateien auszuführen, ist das Quelltextverzeichnis als PyCharm-Projekt zu laden.

Pfad/Dateiname	Funktion
data_in	Das Verzeichnis beinhaltet Einstellungen und FEM-Skriptvorlagen. Diese Dateien werden nicht zur Laufzeit verändert.
- setup.ini	Diese Datei speichert die Grundeinstellungen wie den Pfad zu Abaqus, die Anzalh der zu verwendeten CPU-Kerne und Materialeigenschaften.
data_out	In dieses Verzeichnis werden alle zur Laufzeit generierten Dateien geschrieben. Dabei werden teilweise Unterordner für einzelne Projekte angelegt.
examples	Ein Verzeichnis, welches ausführbare Beispielskripte enthält. Hierbei handelt es sich um simple 2D und 3D Tests der selbst implementierten Surrogate Methoden (aus dem Verzeichnis *mylibs*). Die Plots aus Kapitel 2.2 wurden mit diesen Skripten erstellt.
lib	In diesem Verzeichnis sind externe Python-Pakete (openMdao, pyoptsparse, pykriging, pydoe2 und inspyred). Diese können auch in der genutzten Python Distribution installiert werden, jedoch wird mit den hier gespeicherten Paketen Versionskompatibilität garantiert.
mylibs	In diesem Verzeichnis sind die selbst erstellten Implementationen der Versuchsplan- und Surrogate-Algorithmen. Sie haben einheitliche Schnittstellen und unterstützen beliebig viele Eingänge.
myutils	Das Verzeichnis enthält generelle Hilfsmethoden.
- plot_helper.py	Diese Klasse bietet eine Abstraktion von matlibplot.pyplot für einheitliches 2D bzw. 3D Plotten und Speichern der Grafiken für LaTex.
- samples.py	Diese Datei enthält einfache Polynome als Beispielfunktionen, die in den Beispielen (in *examples*) genutzt werden.

- time_track.py	Dies ist eine Klasse, die zum Messen der Laufzeit genutzt werden kann.
wingconstruction	In diesem Verzeichnis sind alle Dateien, die für die Berechnungen des Beispielproblems (beschrieben in Kapitel 3.2) genutzt werden.
- **fem**	Dieses Verzeichnis enthält Klassen und Methoden, die die Flügelkastengeometrie generieren und den Abaqus-Löser aufrufen. Zudem steht eine Calculix-Schnittstelle zur Verfügung, die neben Abaqus verwendet werden kann.
- **wingutils**	Das Verzeichnis enthält Hilfsfunktionen und Definitionen speziell für das Beispielproblem.
- - constants.py	Diese Klasse ist ein Singleton und kann vom gesamten Projekt genutzt werden. Sie lädt die Einstellungen aus der *data_in/setup.ini*.
- - defines.py	Diese Datei definiert Variablen mit den Werten aus Tabelle 3.1, die von der Surrogate und OpenMDAO Berechnung verwendet werden.
- analysis.py	Die Datei führt *surrogate_run.py* als Batch-Run mit verschiedenen Parametern aus. So können die verschiedenen Methoden verglichen werden (Bsp., siehe Abb. 4.6).
- doe_analysis.py	Dieses Skript führt eine einfache DoE-Analyse des Flügelkastenproblems aus.
- multy_run.py	Mit dieser Klasse können mehrere FEM-Berechnungen parallel auf verschiedenen CPU-Kernen ausgeführt werden. Dabei müssen lediglich Rippenanzahlen und Blechdicken als Parameter übergeben werden.
- newton_opti.py	Dieses Skript startet eine Optimierung unter direkter Nutzung der FEM-Rechnung mit dem Sekantenverfahren.
- open_mdao.py	Dieses Skript startet eine OpenMDAO Optimierung, direkt mit der FEM-Berechnung. Es kann eingestellt werden, welcher Optimierungs-Algorithmus verwendet werden soll.
- open_mdao_surro.py	Das Skript startet eine OpenMDAO Optimierung wie in *open_mdao.py* allerdings auf einem Surrogate Model. Dadurch geht die Berechnung sehr schnell. Dies hilft bei der Erprobung von OpenMDAO Einstellparametern.

- project.py	Die Datei bildet die Schnittstelle zwischen den FEM-Solver, Prä- und Post-Prozessor zum Rest des Programms. Ein Projekt ist eine FEM-Berechnung, die sich üblicherweise über die Rippenanzahl und die Blechdicke identifiziert. Jedes Projekt bekommt ein eigenes Unterverzeichnis in *data_out*.
- single_run.py	Das Skript führt eine einzelne FEM-Berechnung aus. Es dient hauptsächlich zum Testen und Debugging.
- surrogate_run.py	Diese Klasse erstellt einen Versuchsplan, startet die FEM-Berechnungen, generiert ein Surrogate Modell, trainiert dieses, validiert es und findet ein Optimum mit ihm. Welches Versuchsverfahren, wie viele Stützstellen und welche Surrogate Methode verwendet werden sollen, muss als Parameter übergeben werden.

Tabelle A.1: Beschreibung der Python-Pakete und Dateien