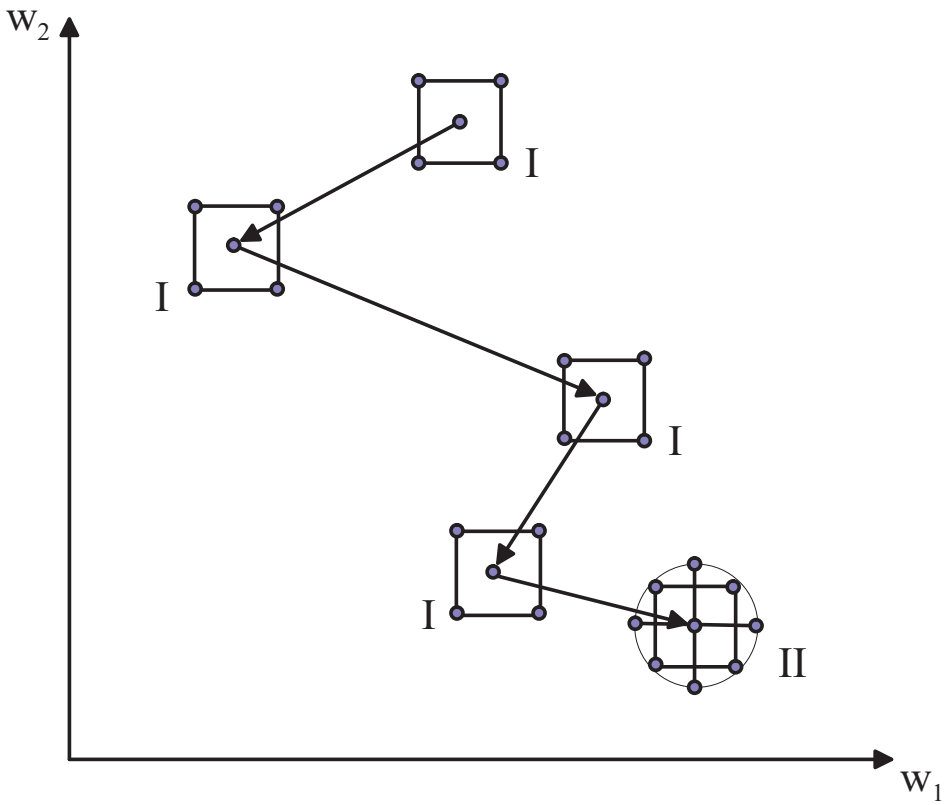
SFB 559
LS Informatik IV
Universität Dortmund
04. März 2005
Version 1.0

Sonderforschungsbereich 559

Modellierung großer Netze in der Logistik

Technical Report 05003
ISSN 1612-1376

Einsatz der Response Surface Methode zur Optimierung komplexer Simulationsmodelle



SFB 559 – Teilprojekt M2

Dennis Müller, Mathias Stöber, Axel Thümmler

Kurzfassung

In diesem Artikel werden die grundlegenden Aspekte der Response Surface Methode vorgestellt. Die umfasst vor allem den Ablauf dieser Methode, die Grundlagen zur Erstellung von Experimentdesigns, die Anpassung linearer Regressionsmodelle und die Überprüfung der Güter dieser Regressionsmodelle. Darüber hinaus wird die aktuelle Implementierung vorgestellt und deren Praktikabilität anhand unterschiedlicher Funktionen untersucht. Dabei wird vor allem das Verhalten bei hochdimensionalen Funktionen betrachtet. Die eingesetzten Funktionen weisen dabei unterschiedliche Eigenschaften wie z.B. Unimodalität oder Multimodalität auf, so dass unterschiedliche, in Simulationsmodellen auftretende Verläufe nachgebildet werden. Darüber hinaus wurde der Einfluss von Fractional Factorial Designs auf das Ergebnis von RSM betrachtet. Weitergehend wird die Anbindung externer Programme exemplarisch durch die Nutzung von ProC/B in der Verbindung mit HIT und die Nutzung der APNN-Toolbox demonstriert. Des weiteren wird die grafische Benutzeroberfläche vorgestellt, die zur besseren Verwendbarkeit der Implementierung von RSM entwickelt wurde und es ermöglicht ohne Detailkenntnisse über RSM dieses für die Optimierung von Modellen zu nutzen.

Inhaltsverzeichnis

Einleitung

Der Entwurfsprozess heutiger Computer- und Kommunikationssysteme muss durch quantitative Modellbildung begleitet werden. Nur so kann unter einem vorgegebenen Kostenrahmen eine formal korrekte, leistungseffiziente und optimale Systemrealisierung gewährleistet werden. In der Modellbildung werden Computer- und Kommunikationssysteme als diskrete ereignisorientierte Systeme dargestellt und mittels Simulation oder numerischer quantitativer Analyse bewertet [LK00]. Bei der Optimierung von Simulationsmodellen wird eine Belegung der Eingabeparameter des Simulationsmodells gesucht, welche eine oder mehrere Ausgabegrößen des Modells minimiert bzw. maximiert. Ein wesentliches Merkmal von Optimierungsalgorithmen für Simulationsmodelle ist, dass diese mit den mehr oder weniger starken stochastischen Schwankungen der Simulationsausgabe zurecht kommen müssen. Ferner ist oft schon die Auswertung einer Parameterbelegung durch die Simulation sehr zeitaufwendig, so dass Optimierungsverfahren, die mit möglichst wenigen Auswertungen auskommen, von Interesse sind.

Auch wenn für die Optimierung von Simulationsmodellen zahlreiche Techniken existieren, ein kompakter Überblick ist in [Fu02] zu finden, treten bei der praktischen Anwendung doch zahlreiche Probleme auf, da viele der vorgeschlagenen Optimierungsansätze nicht robust genug sind, nicht für große Systeme oder zahlreiche Eingabeparameter skalieren, spezielle Anforderungen an Simulationsmodelle stellen, die in der Regel nicht erfüllbar sind, oder aber manuelle Eingriffe des Modellierers erfordern. Die automatische Kopplung von ereignisdiskreten Simulatoren mit klassischen Optimierern führt aber gerade bei komplexen Simulationsmodellen meist zu unbefriedigenden Ergebnissen, wenn sie sich auf ein einfaches Zusammenführen von Simulationsmodell und Optimierungsverfahren beschränkt. Als robuste Methode, die das Potential einer solchen Kopplung mitbringt, hat sich die *Response Surface Methode (RSM)* herausgestellt [MM02]. In erst kürzlich erschienenen eigenen Arbeiten konnte die Praktikabilität von RSM zur Optimierung von Simulationsmodellen als auch numerisch analysierbarer Modellbeschreibungen gezeigt werden [BMT05], [KMT05].

Die Optimierung über Response Surfaces ist eng verbunden mit der Metamodellbildung [KS00], da iterativ zwischen Simulationsexperimenten und der Anpassung und Optimierung von Metamodellen gewechselt wird. Dabei wird mit Hilfe des Metamodells der Raum des Optimums eingegrenzt und auf Basis von Simulationsexperimenten im eingegrenzten Parameterraum ein neues Metamodell erzeugt, welches zur weiteren Eingrenzung des Parameterraums verwendet werden kann. Neuere Arbeiten zur Response Surface Methode beschäftigen sich primär mit der Verbesserung der Methode und der Automatisierung des Ansatzes [NOP+00], was voraussetzt, dass neben der automatischen Anpassung von Metamodellen auch getestet wird, ob diese adäquat für das beobachtete Verhalten sind.

In diesem Artikel wird die die aktuelle Implementierung vorgestellt und deren Praktikabilität anhand unterschiedlicher Funktionen untersucht. Dabei wird vor allem das Verhalten bei hochdimensionalen Funktionen betrachtet. Die eingesetzten Funktionen weisen dabei unterschiedliche Eigenschaften wie z.B. Unimodalität oder Multimodalität auf, so dass unterschiedliche, in Simulationsmodellen auftretende Verläufe nachgebildet werden. Diese Funktionen wurden bereits in [NP98] und [Sal98] als Testfunktionen für andere Optimierungsverfahren genutzt. Darüber hinaus wurde der Einfluss von Fractional Factorial Designs [Kle98] auf das Ergebnis von RSM betrachtet, um abschätzen zu können, wann ein Factorial Design nötig und wann ein Fractional Factorial Design ausreichend ist. Es hat sich gezeigt, dass für Funktionen, die bestimmte Eigenschaften gemeinsam haben, der RSM-Algorithmus ähnlich gute Ergebnisse liefert, wobei die Unterschiede in der Güte der Ergebnisse leicht durch die Unterschiede in den Funktionen erklärbar sind. Daher sollte es in vielen Fällen, in denen der Verlauf der Response Surface abgeschätzt werden kann, möglich sein, die Ergebnisse der Testfunktionen zu übertragen und eine entsprechende, günstige Konfiguration für die RSM-Implementierung zu wählen. Insbesondere hat es sicht gezeigt, dass Stufen innerhalb einer Funktion bis zu einer bestimmten Größe fast keinen Einfluss auf das Ergebnis haben. Demgegenüber hat es sich bestätigt, dass RSM lokale Optima findet. Die Wahrscheinlichkeit, dass RSM ein lokales Optimum und nicht das globale Optimum findet, steigt mit der Ausprägung der lokalen Optima, wobei hierdurch auch der ermittelte maximale Response steigt und somit das Problem relativiert. Zur besseren Verwendbarkeit der Implementierung von RSM wurde eine grafische Benutzeroberfläche entwickelt. Sie ermöglicht es ohne Detailkenntnisse über RSM dieses für die Optimierung von Modellen zu nutzen. Die aktuelle Distribution der RSM-Implementierung kann auf Anfrage von den Autoren bezogen werden.

Die nachfolgenden Kapitel sind wie folgt organisiert. In Kapitel 1 wird die zentralen Teile der Response Surface Methode vorgestellt. Kaptitel 2 befasst sich mit der aktuellen Implementierung und beschreibt insbesondere die Standardkonfiguration und Designentscheidungen. Kapitel 3 werden die eingesetzten Testfunktionen, die durchgeführten Tests und deren Ergebnisse beschrieben und untersucht. Kapitel 4 demonstriert die Integration externer Programme zur Auswertung von Modellen in die aktuelle RSM-Implementierung. In Kapitel 5 wird die grafische Benutzeroberfläche vorgestellt. Abschließend bietet Kapitel 6 eine Zusammenfassung und einen Ausblick.

1 Einführung in die Optimierung mittels der Response Surface Methode

RSM ist eine Sammlung statistischer und mathematischer Methoden, die zur Optimierung von Systemen genutzt werden können. Diese Methode wurde ausführlich im Allgemeinen in [MM02] und in Hinblick auf Simulation in [NOP+00] diskutiert. Die hier aufgeführte Beschreibung orientiert sich im Wesentlichen an diesen Ausführungen.

Systeme mit n sowohl kontinuierlichen als auch diskreten Eingabeparametern werdet betrachtet, wobei der Response y optimiert werden soll, der durch eine Funktion $y = f(x_1, \ldots, x_n)$ festgelegt ist. Des weiteren muss eine Reihe von Nebenbedingungen erfüllt sein. Im Gegensatz zu den Nebenbedingungen ist in den meisten Fällen die Funktion f nicht bekannt. Es ist aber möglich den Response konkreter Faktorwerte zu ermitteln. Dazu stehen ggf. mehrere Möglichkeiten zur Verfügung:

1. Berechnung des Responses auf der Basis bekannter Informationen (z.B. physikalische Zusammenhänge)
2. Messung der Werte am realen System
3. Modellierung und anschließende Simulation oder numerische Auswertung

Die ermittelten Werte enthalten in der Regel einen Fehler unbekannter Größe. Daher kann die Funktion f nur durch eine Funktion g mit $y = g(x_1, \ldots, x_n) + \varepsilon$ approximiert werden, wobei ε für den geschätzten Fehler steht.

Vor Beginn der Anwendung von RSM müssen einige Vorbereitungen durchgeführt werden. Dabei muss zuerst entschieden werden, welche der oben aufgeführten Methoden zur Bestimmung des Responses genutzt werden soll. Anschließend müssen die Faktoren bestimmt werden, die bei der Optimierung betrachtet werden sollen. Dabei sollten Faktoren, deren Einfluss auf den Response gering ist oder die keinen Einfluss auf den Response ausüben, nicht in die Betrachtung einbezogen werden. So kann die Anzahl der während der Durchführung von RSM benötigten Auswertungen reduziert. Darüber hinaus muss für jeden Faktor ein Wertebereich festgelegt werden. Diese Maßnahme dient dem Ausschluss nicht möglicher bzw. nicht sinnvoller Wertebelegungen (z.B. Begrenzung durch minimale und maximale Kapazität eines Lagers, Faktorwerte mit bekanntem und niedrigem Response) oder der Eingrenzung auf einen Bereich mit vermutlich hohem Response. Zuletzt muss ein Startpunkt gewählt werden, der als erster Centerpoint genutzt wird.

1.1 Ablauf der Optimierung und Einsatz der dabei genutzten Methoden

Die Optimierung mittels RSM wird schrittweise durchgeführt. Dabei wird bei jedem Schritt lediglich ein Teil des gültigen Wertebereichs betrachtet. Mittelpunkt dieses Bereichs ist der aktuelle Centerpoint. Die halbe Breite des Bereichs w ist eine relative Größe und bezieht sich

auf die Größe des Wertebereichs des jeweiligen Faktors. Zur besseren Handhabbarkeit werden die Faktorwerte aus ihren natürlichen Wertebereichen in kodierte, einheitenlose Wertebereiche umgerechnet. Dabei gilt:

1. Der Centerpoint ist Nullpunkt
2. Alle Koordinaten der Eckpunkte des untersuchten Bereichs sind *1* oder *-1*

Für einen Werte w_i aus dem natürlichen Wertebereichs des Faktors i mit einer unteren Grenze l_i und einer oberen Grenze u_i ergibt sich der Wert x_i im kodierten Wertebereich wie folgt:

$$\begin{aligned} m_i &= \frac{u_i + l_i}{2} \\ b_i &= \frac{u_i - l_i}{2_i} \\ x_i &= \frac{w_i - m_i}{b_i} \end{aligned} \qquad \textbf{(1)}$$

Im weiteren Verlauf dieses Texts betrachten wir ausschließlich kodierte Variablen.

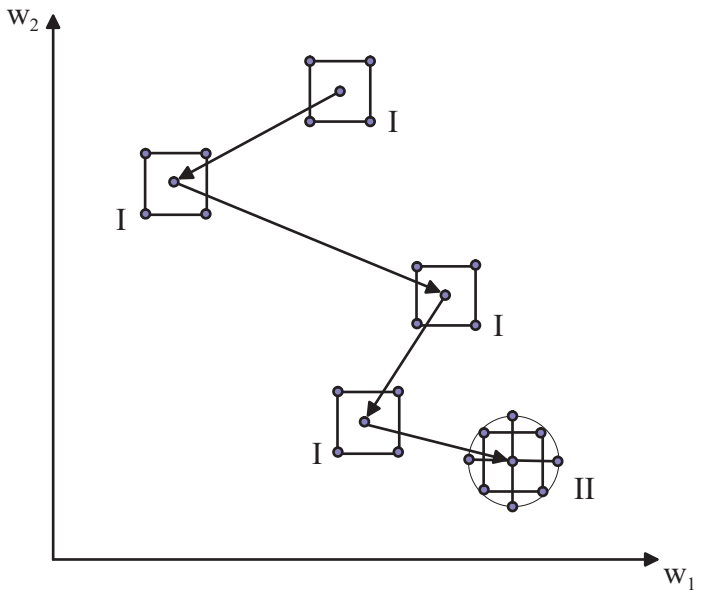


Abbildung 1: Ablauf der Response Surface Methode

Vor dem eigentlichen Optimierungsschritt muss entschieden werden, wie viele Punkte ausgewertet und nach welchen Kriterien diese ausgewählt werden sollen. Die Auswahl der Anzahl der Punkte wird nach unten von der Art des verwendeten Modells und nach oben durch die benötigte Zeit begrenzt. Zur Auswahl der Punkte haben sich verschiedene Designs etabliert, die Kriterien zur Berechnung der Koordinaten der einzelnen Punkte festlegen. Nachdem die Entscheidungen über die Anzahl der Punkte und der Art des Designs getroffen wurden, müssen die Punkte ausgewertet werden. RSM sieht es dabei vor ggf. Punkte repliziert auszuwerten, um genauere Ergebnisse zu erhalten. Dabei wird üblicherweise die Anzahl der

Replikationen des Centerpoints und der Designpoints einzeln festgelegt, wobei es sich als sinnvoll erwiesen hat, den Centerpoint häufiger zu replizieren als die Designpoints. Die empfohlene Standardkonfiguration sieht drei Replikationen des Centerpoints und eine einfache Auswertung der Designpoints vor, wodurch die Anzahl der Auswertungen relativ gering bleibt und trotzdem in vielen Fällen eine erhöhte Genauigkeit erreicht wird. Anschließend kann mit dem eigentlichen Optimierungsschritt auf der Basis eines Modells erster oder zweiter Ordnung begonnen werden. Das Ergebnis eines Optimierungsschritts ist entweder der alte oder ein neuer Centerpoint, der sich durch einen besseren Response auszeichnet. Auf diese Weise nähert man sich einem lokalen Optimum (siehe Abbildung 1).

Nach jedem Optimierungsschritt muss der Optimierer entscheiden, wie weiter vorzugehen ist. Seine wesentlichen Entscheidungsmöglichkeiten sind:

- Fortfahren
    - Verkleinerung des Suchbereichs?
    - Erlauben eines Modells zweiter Ordnung?
- Abbruch

Es gibt eine Reihe von Kriterien anhand derer der Optimierer seine Entscheidung fällen kann. Die zwei Hauptkriterien „Größe des Suchbereichs“ und „Anzahl der Auswertungen bzw. Optimierungsschritte“ sollten dabei auf jeden Fall berücksichtigt werden. Ist der Suchbereich ausreichend klein, ist erstens keine wesentliche Verbesserung des Responses zu erwarten und zweitens haben auch als kontinuierlich angenommene Faktoren häufig eine beschränkte Genauigkeit. Über die Anzahl der Auswertungen bzw. Optimierungsschritte kann der Zeitaufwand für die Optimierung abgeschätzt werden. Diese beiden Bedingungen eignen sich einerseits als Abbruchkriterium andererseits auch als Entscheidungsmerkmal für den Einsatz eines Modells zweiter Ordnung. Modelle zweiter Ordnung sollten aufgrund der hohen Anzahl benötigter Auswertungen nicht von Beginn an genutzt werden. Gegen Ende der Optimierung befindet man sich jedoch meist in der Nähe eines Optimums. Ein solcher Bereich lässt sich durch ein Modell zweiter Ordnung deutlich besser approximieren als durch ein Modell erster Ordnung. Daher ist hier der erhöhte Aufwand gerechtfertigt.

Eine weitere wichtige Entscheidung ist der Zeitpunkt, zu dem der Suchbereich verkleinert werden soll. Es empfiehlt sich den Suchbereich nicht zu schnell zu verkleinern, da gerade bei stark gestörten Responses ein zweiter Optimierungsschritt mit gleicher Suchbereichsgröße und korrigierter Suchrichtung viel versprechend ist. Eine gute Heuristik ist die Verkleinerung des Suchbereichs nach einem erfolglosen Optimierungsschritt, da man in vielen Fällen davon ausgehen kann, dass ein Optimum innerhalb des aktuellen Suchbereichs liegt und damit die verwendete Schrittweite zu groß ist.

Darüber hinaus existiert eine Reihe weiterer Kriterien anhand derer, der Ablauf von RSM verändert werden kann:

- Distanz zwischen aktuellem und letzten Centerpoint
- Differenz der Responses von aktuellem und letzten Centerpoint
- ...

1.2 Experimentdesigns

In den meisten Fällen werden zur Auswahl der Punkte (Fractional) Factorial Designs oder Central Composite Designs genutzt, die hier aufgeführte Beschreibung orientiert sich im Wesentlichen an [Mon01]. Ein Factorial Design aus insgesamt 2^k Punkten, wobei k die Anzahl der Faktoren ist. Die Punkte eines Factorial Designs werden um einen Centerpoint mit einer Weite w erzeugt, wobei für jeden Faktor x_i des Centerpoints die Werte $x_{i1} = x_i - w$ und $x_{i2} = x_i + w$ berechnet und alle möglichen Kombinationen erzeugt werden.

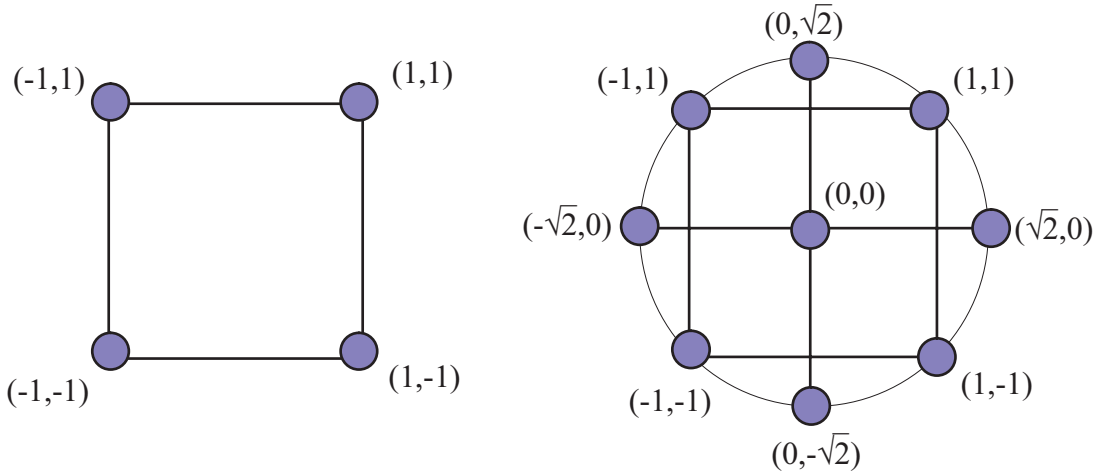


Abbildung 2: Factorial & Central Composite Design

Zur Reduktion der Anzahl dieser Punkte werden Fractional Factorial Designs (siehe Abbildung 2 links) genutzt, die aus 2^{k-p} Punkten bestehen. Dabei werden nur die ersten $k-p$ Punkte entsprechend eines Factorial Designs berechnet. Die Werte der restlichen Faktoren jedes Punkts ergeben sich dann anhand so genannter erzeugender Muster aus den bereits gewählten Werten. Zur Bestimmung der Qualität eines Fractional Factorial Designs wird die so genannte Auflösung bestimmt. Sie entspricht dem Minimum der Anzahl der Faktoren, von denen einer der letzten $k-p$ Werte abhängt. Beispielsweise kann bei einem *5-2* Fractional Factorial Design der vorletzte Faktor von den Faktoren *1* und *2* abhängen und der letzte Faktor von den Faktoren *2* und *3*. Daraus ergäbe sich eine Auflösung von *2*. Wäre der letzte Faktor nur vom Faktor *3* abhängig wäre die Auflösung *1*. Gleiches gilt, wenn der letzte Faktor, von den Faktoren *1* und *2* abhängt, damit entspricht er dem vorletzten Faktor. Die

Auflösung eines Fractional Factorial Designs sollte soweit möglich maximiert werden; ein Fractional Factorial Design mit einer Auflösung von 1 sollte nicht genutzt werden.

Das Central Composite Design (siehe Abbildung 2 rechts) entspricht einem Factorial Design, das um einige Punkte erweitert wurde. Für jeden Faktor werden zwei zusätzliche Punkte erzeugt, die jeweils in Richtung dieses Faktors um $\pm\sqrt{2}\cdot w$ verschoben werden. Die Gesamtzahl der Punkte eines Central Composite Designs ist also 2^k+2k.

1.3 Lineare Regressionsmodelle

Zur Berechnung neuer Centerpoints wird durch die beobachteten Werte ein Metamodell erstellt. Hierzu werden in der Regel lineare Regressionsmodelle erster und zweiter Ordnung genutzt. Regressionsmodelle höherer Ordnung werden im Normalfall nicht eingesetzt, da die Anzahl der benötigten Designpunkte und damit die Anzahl der Auswertungen sehr hoch ist. Die Anpassung eines lineares Regressionsmodells erster bzw. zweiter Ordnung wird in den beiden folgenden Unterkapiteln beschrieben.

1.3.1 Optimierung mittels lineare Regressionsmodelle erster Ordnung

Bei der Optimierung anhand eines Modells erster Ordnung wird auf der Basis ermittelter Responses eine Hyperebene approximiert. Diese Hyperebene dient je nach Optimierungsziel zur Bestimmung der Richtung des steilsten An- oder Abstiegs, die wiederum zur Bestimmung neuer Centerpoints genutzt wird.

$$y = \beta_0 + \sum_{i=1}^{k} (\beta_i \cdot x_i) + \varepsilon \qquad \textbf{(2)}$$

Zur Bestimmung des Modells erster Ordnung (siehe Formel 2) muss zuerst ein (Fractional) Factorial Design mit der vorgegebenen Breite des Suchbereichs w und dem aktuellen Centerpoint erstellt werden. Dabei werden $n \geq k + 1$ Designpunkte mit dem zugehörigen Response benötigt, um die Regressionskoeffizienten β_0 bis β_k zu bestimmen. Mit den Designpunkten $x_i = (x_{i1}, \ldots, x_{ik})$ und den zugehörigen Responses y_i mit $i = 1, \ldots, k$ lassen sich die in Formel 3 beschriebenen Matrizen bilden, wobei der Vektor ε und der Vektor β unbekannt sind. Mit der Methode der kleinsten Fehlerquadrate kann nun Formel 3 berechnet und damit die Regressionskoeffizienten β mit minimalen ε^2 bestimmt werden.

$$\mathbf{y} = \mathbf{X} \cdot \boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

mit

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{pmatrix}, \ \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \ \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \qquad \textbf{(3)}$$

Die Richtung des steilsten Anstiegs kann aus dem Vektor β bestimmt werden und besteht aus den Werten β_1 bis β_k. Zur Bestimmung der Schrittweite können verschiedene Heuristiken angewendet werden. Die empfohlene Methode wählt die Schrittweite konstant, so dass der erste Schritt auf eine der Seiten des Hypercubes führt, der durch die Punkte eines Factorial Designs aufgespannt wird (siehe Formel 3). Potentielle neue Centerpoints ergeben sich nur als Summe des aktuellen Centerpoints und des Richtungsvektors $\bar{\beta}$. Ist der Response des potentiellen neuen Centerpoints besser als der des Alten, wird der neue Punkt als Centerpoint akzeptiert und die schrittweise Suche fortgeführt; ist der Response schlechter, wird die Suche beendet.

$$\bar{\beta} = \frac{\begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}}{\max(\beta_1, \ldots, \beta_k)} \qquad \textbf{(4)}$$

Bei dieser Vorgehensweise ist zu beachten, dass die Nebenbedingungen der zu optimierenden Funktion eingehalten werden. Dazu muss ggf. der Richtungsvektor korrigiert (siehe Abbildung 3) oder der Optimierungsschritt beendet werden.

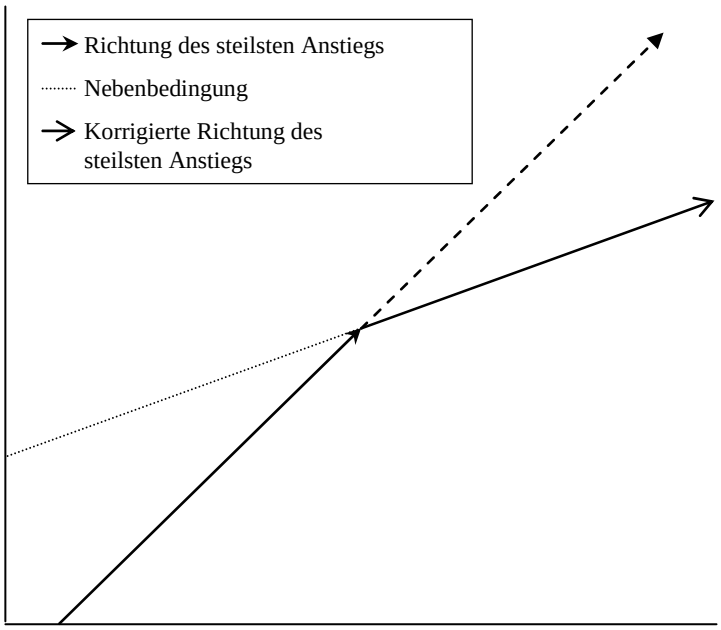


Abbildung 3: Korrektur der Richtung des steilsten Anstiegs

1.3.2 Optimierung mittels linearer Regressionsmodelle zweiter Ordnung

Ein lineares Regressionsmodell zweiter Ordnung ist wie in Formel 5 dargestellt wird für Polynom zweiten Grades mit *1 + (k(k+3))/2* Regressionskoeffizienten, wodurch auch die Mindestanzahl auszuwertender Punkte festgelegt ist, eingesetzt. Die Regressionskoeffizienten β_0 bis β_k entsprechen denen des Regressionsmodells erster Ordnung. Hinzu kommen die Regressionskoeffizient $\beta_{i,i}$ mit $i = 1, \ldots, k$ für die Quadrate aller Faktoren und die

Regressionskoeffizienten $\beta_{i,j}$ mit $i = 1,\ldots, k$ und für jedes i: $j = i+1, \ldots, k$ für alle paarweisen Produkte der Faktoren. Da es sich um ein lineares Regressionsmodell handelt, werden die paarweisen Kombinationen der Variablen als eigenständige Variablen interpretiert. Die dafür notwendige Ersetzung wird wie folgt durchgeführt:

- $\beta_{k+1} := \beta_{1,1}, \ldots, \beta_{2k} := \beta_{k,k}, \beta_{2k+1} := \beta_{1,2}, \ldots$
- $x_{k+1} := x_1 \cdot x_1, \ldots, x_{2k} = x_k \cdot x_k, x_{2k+1} = x_1 \cdot x_2, \ldots$

Anschließend kann die Bestimmung der Regressionskoeffizienten mittels der Methode der kleinsten Fehlerquadrate durchgeführt werden.

$$y = \beta_0 + \sum_{i=1}^{k} (\beta_i \cdot x_i) + \sum_{i=1}^{k} \left(\beta_{i,i} \cdot x_i^2\right) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} \left(\beta_{i,j} \cdot x_i \cdot x_j\right) + \varepsilon \qquad \textbf{(5)}$$

$$\hat{y} = \beta_0 + x'\beta + x'\hat{B}x$$

mit

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \ \hat{B} = \begin{pmatrix} \beta_{11} & \beta_{12}/2 & \cdots & \beta_{1k}/2 \\ & \beta_{22} & \cdots & \beta_{2k}/2 \\ & & \ddots & \vdots \\ \textit{sym.} & & & \beta_{kk} \end{pmatrix}, x = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}, \ x' = (x_1, \ldots, x_k) \qquad \textbf{(6)}$$

Mit den ermittelten Regressionskoeffizienten kann nach Rückersetzung die in Formel 6 dargestellte polynomielle Funktion zweiten Grades erstellt werden, die als Approximation des Response Surface dient. Aus den Matrizen $\hat{B}$ und β lässt sich der stationäre Punkt x_s wie in Formel 7 beschrieben bestimmen. Die Art des stationären Punkts kann anhand der Eigenwerte der Matrix $\hat{B}$ bestimmt werden. Sind alle Eigenwerte negativ, so handelt es sich um ein Maximum, sind alle Eigenwerte positiv, handelt es sich um ein Minimum und in allen anderen Fällen ist x_s ein Sattelpunkt. Je nach Optimierungsziel kann x_s verworfen werden, wenn es sich nicht um ein Maximum bzw. Minimum handelt. Ansonsten muss x_s ausgewertet werden und der Response mit dem Response des aktuellen Centerpoints verglichen werden.

$$x_s = -\frac{1}{2} \hat{B}^{-1} \beta \qquad \textbf{(7)}$$

1.4 Möglichkeiten der Überprüfung der Güte von Regressionsmodellen

In der bisherigen Beschreibung sind wir davon ausgegangen, dass die Modelle erster bzw. zweiter Ordnung eine gute Approximation der (gemessenen) Wirklichkeit darstellen. Vor allem bei starker Störung der Responses ist dies jedoch nicht immer der Fall. Das Ergebnis eines Optimierungsschritts auf der Basis einer schlechten Approximation ist in den meisten

Fällen ein Fehlschlag. Die Bewertung der Qualität einer Approximation ermöglicht es entsprechend darauf zu reagieren, wobei bis zu vier Möglichkeiten zur Verfügung stehen:

- Abbruch des Optimierungsschritts
- Erneute ggf. genauere Auswertung der Responses der aktuellen Designpoints
- Auswertung zusätzlicher Designpoints
- Approximation eines Modells höherer Ordnung

Im folgenden werden die Entscheidungen anhand des „Lack Of Fit Tests“ und des „Coefficient of Determination“ beschrieben. Bei beiden Verfahren ist jedoch zu beachten, dass die Auswertung eines Designs vor allem bei mehr als zwei Faktoren aufwändig ist, und die Auswertung eines neuen Centerpoints einen relativ geringem Aufwand entspricht. Die Tests sollten also so durchgeführt werden, dass nur besonders schlechte Modelle verworfen werden. Zu beachten ist auch, dass diese Verfahren dazu führen können, ein schlechteres Endergebnis zu erhalten, wenn ausreichend gute Modelle verworfen werden.

1.4.1 Coefficient Of Determination

Der „Coefficient of Determination“ (CoD) soll bestimmen, wie gut ein Modell die Responses bekannter Punkte erklärt. Dazu wird die Summe der Fehlerquadrate SS_E und die gesamte Summe der Fehlerquadrate SS_T (siehe Tabelle 1) berechnet. SS_E gibt die Summe der Fehlerquadrate an, die nicht durch das Modell erklärt werden. Also beschreibt SS_E/SS_T den Anteil an der Summe der Fehlerquadrate, der nicht erklärt ist. Der Coefficient of Determination R^2 beschreibt den erklärten Anteil der Summe der Fehlerquadrate und ergibt sich entsprechend Formel 8. R^2 liegt im Intervall $[0, 1]$. Dabei entspricht der Wert 1 einer sehr guten Anpassung des Regressionsmodells an die gemessenen Responses und 0 einer sehr schlechten Anpassung.

$$R^2 = 1 - \frac{SS_E}{SS_T} \qquad \textbf{(8)}$$

Ein Problem dieser Bewertung liegt darin, dass R^2 bei unterschiedlicher Anzahl der Faktoren schwanken kann. In vielen Fällen wird R^2 mit der Anzahl der Faktoren wachsen. Um diesem Effekt entgegen zu wirken, kann ein angepasster CoD R^2_{adj} (siehe Formel 9) eingesetzt werden. Die eingefügten Faktoren entsprechen dabei den Freiheitsgraden von SS_T ($n-1$) und SS_E ($n-k-1$).

$$R^2_{adj} = 1 - \frac{n-1}{n-k-1}\left(1-R^2\right) = 1 - \frac{n-1}{n-k-1}\frac{SS_E}{SS_T} \qquad \textbf{(9)}$$

Symbol	Bedeutung
y_{ij}	j-ter Response von Punkt i
n_i	Anzahl der Responses für Punkt i
m	Anzahl der Punkte
$n = \sum_{i=1}^{m} n_i$	Summe der Auswertungen
$\bar{y} = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n_i} y_{ij}$	Durchschnittlicher Response
$SS_E = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (y_{ij} - \hat{y}_i)^2$	Summe der Fehlerquadrate
$SS_T = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (y_{ij} - \bar{y})^2$	Gesamte Summe der Fehlerquadrate

Tabelle 1: Referenz benötigter Variablen

1.4.2 „Lack of Fit“-Test

Der „Lack of Fit“-Test (LoF-Test) ist ein statistischer Test des Modells. Im Gegensatz zum CoD werden hier für mindestens einen Punkt replizierte Auswertungen benötigt. Der LoF-Test stützt sich auf die Summe der Fehlerquadrate SS_E und teilt diese in SS_{LOF} und SS_{PE} (siehe Formel 10), wobei SS_{LOF} einen Freiheitsgrad von *m-k-1* und SS_{PE} einen Freiheitsgrad von *n-m* besitzt.

$$SS_{PE} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2$$

$$SS_{LOF} = \sum_{i=1}^{m} n_i (\bar{y}_i - \hat{y}_i)^2 \qquad \textbf{(10)}$$

mit

$$\bar{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$$

$$F_0 = \frac{SS_{LOF}}{SS_{PE}} \frac{n-m}{m-k-1} \qquad \textbf{(11)}$$

Die Teststatistik F_0 kann entsprechend Formel 11 berechnet werden. Wenn die Funktion linear ist, können wir annehmen, dass sich F_0 entsprechend der $F_{m-k-1,n-m}$ Distribution verhält.

Dementsprechend können wir aus $F_0 > F_{\alpha,m-k-1,n-m}$ schließen, dass die Funktion nicht linear ist. Hierbei kommt der Wahl des Werts für α besondere Bedeutung zu, da dessen Größe die Wahrscheinlichkeit bestimmt, mit der ein Modell akzeptiert wird.

1.5 Zusammenfassung

RSM ist eine Suchheuristik, die mit einer relativ geringen Anzahl an Auswertungen Faktorbelegungen in der Nähe eines lokalen Optimums bestimmt, wobei das Ergebnis vor allem von der Genauigkeit der gemessenen Responses abhängig ist. Dabei werden während des Verfahrens vor allem Modelle erster Ordnung eingesetzt. Diese werden gegen Ende des Verfahrens durch die aufwändigeren Modelle zweiter Ordnung ergänzt. Zur Vermeidung von Auswertungen, bei denen es unwahrscheinlich ist, dass sie zu einer Verbesserung des Ergebnisses führen, werden verschiedene statistische Tests eingesetzt.

2 Implementierungsaspekte

Die aktuelle Implementierung ist eine automatisierte Umsetzung von RSM. Diese ermöglicht es ein Modell ohne genauere Kenntnis des RSM-Algorithmus zu optimieren. Durch weitere Einstellungen kann das Verhalten des Algorithmus angepasst werden, diese sind jedoch mit Werten vorbelegt, die in der Regel gute Ergebnisse liefern. Zur einfacheren Benutzung der Implementierung existiert eine grafische Benutzeroberfläche. In den folgenden Abschnitten werden die drei Bereiche Verhalten der Implementierung bei unterschiedlichen Konfigurationen, Einbindung von Modellen und die grafische Benutzeroberfläche genauer betrachtet. Die Konfigurationsdatei, die eine detaillierte Beschreibung aller Einstellungen enthält, ist im Anhang enthalten. Die

2.1 Beschreibung des implementieren RSM Algorithmus

In Abbildung 4 ist das Verhalten der RSM-Implementierung in der Standardkonfiguration beschrieben. Dabei werden Centerpoints jeweils dreimal und Designpoints einmal ausgewertet. Zu Beginn wird der Centerpoint mit den Faktorwerten (*0, …, 0*) belegt, sofern diese im gültigen Definitionsbereich der Faktoren liegen. Die halbe Breite des Suchbereichs w beträgt zu Beginn *0,4* bzgl. der Wertebereichs jedes Faktors. Die einzige Abbruchbedingung ist das Erreichen der Mindestgröße des Suchbereichs $w_{stop} = 0{,}01$. Der aktuelle Centerpoint wird für spätere Vergleiche gespeichert. Anschließend werden die Faktorwerte entsprechend der Beschreibung aus Kapitel 1 in kodierte Werte umgewandelt. Daraufhin wird ein Factorial Design erstellt, ausgewertet und auf dessen Basis ein Modell erster Ordnung approximiert. Dieses Modell wird sofern möglich mittels LoF-Test sonst mittels CoD auf Adäquatheit überprüft. Wenn dies der Fall ist, wird die Richtung des steilsten Anstiegs ermittelt und dieser Richtung nach neuen Centerpoints gesucht. Wurde bei der Suche kein besserer Centerpoint gefunden, wird die Größe des Suchbereichs halbiert.

Wenn das Modell erster Ordnung nicht adäquat ist, oder die Optimierung nach diesem Schritt beendet wird, wird ein Central Composite Design erstellt. Wenn der aktuelle Centerpoint auch Centerpoint des Designs für das Modell erster Ordnung war, wird das bereits vorhandene Design auf ein Central Composite Design erweitert, wodurch doppelte Auswertungen von Designpoints vermieden werden. Auf der Basis dieses Designs wird ein Modell zweiter Ordnung approximiert. Entspricht der Typ des stationären Punkts dem Optimierungsziel, wird dieser berechnet und ausgewertet. Wenn der Response des stationären Punkts besser als der des aktuellen Centerpoints ist, wird der stationäre Punkt als neuer Centerpoint akzeptiert. Anschließend wird die Größe des Suchbereichs halbiert.

```
1   Auswahl und Auswertung des ersten Centerpoints c_new = (0,…,0) und der halben Breite des
    Suchbereichs ω = 0.4
2   WHILE ω > ω_stop DO
3       c_old = c_new
4       Umwandlung des Suchbereichs mit Centerpoint c_old in einen [−1,1]^k Hypercube
5       Erstellung eines Factorial Designs und Approximation der Response Surface Funktion innerhalb
        des Suchbereichs über ein Modell erster Ordnung
6       Überprüfung des Modells auf Adäquatheit durch den LoF-Test/CoD
7       IF Modell ist adäquat THEN DO
8           Bestimmen der Richtung des steilsten Anstiegs/Abstiegs und festlegen der Schrittweite
9           REPEAT
10              Gehe einen Schritt in Richtung des steilsten Anstiegs/Abstiegs und ermittle den Response
                an dieser Stelle
11          UNTIL Neuer Response nicht besser als der zuvor ermittelte
12          c_new = Faktorwerte des letzten Punkts mit Verbesserung des Responses
13          IF c_new = c_old THEN Setze die halbe Breite des Suchbereichs ω = ω / 2.0
14      IF ((Modell ist nicht adäquat) AND (ω < 4·ω_stop)) OR (letzter Optimierungsschritt) THEN DO
15          Erstellung eines Central Composite Designs und Approximation der Response Surface
            Funktion innerhalb des Suchbereichs durch ein Modell zweiter Ordnung
16          Ermittle den Typ des stationären Punkts entsprechend der Eigenwerte von Matrix B
17          IF Typ entspricht dem Optimierungsziel THEN DO
18              Ermittle stationären Punkt c_s des Modells und werte diesen aus
19              IF (Response(c_new)<Response(c_s) THEN DO
19                  c_new = Stationärer Punkt
19              OD
20          OD
21          Setze die halbe Breite des Suchbereichs ω = ω / 2.0
22      OD
23  OD
24  RETURN c_new
```

Abbildung 4: Pseudocode einer RSM-Implementierung

Gegenüber der Standardkonfiguration können an verschiedenen Stellen Veränderungen vorgenommen werden. Diese Betreffen z.B. die Anzahl der Auswertungen, die Abbruchkriterien und die Auswahl des Designs.

Die Konfiguration der Anzahl der Auswertungen je Centerpoint und Designpoint mit *3* und *1* ist im allgemeinen sehr erfolgreich. Wenn die Messergebnisse jedoch sehr ungenau sind, können diese Werte erhöht werden, wobei ein Vielfaches der Standardkonfiguration zu empfehlen ist. Auf jeden Fall sollte die Anzahl der Auswertungen je Centerpoint nicht unter der der Designpoints liegen.

Die aktuelle Implementierung bietet eine Reihe weiterer Abbruchkriterien. Zu diesen gehören die minimale Distanz zwischen aktuellem und neuen Centerpoint, die minimale Differenz der Responses von aktuellem und neuen Centerpoint, die maximale Anzahl an Optimierungsschritten oder Auswertungen. Darüber hinaus kann ein Abbruch des Verfahrens für die ersten Optimierungsschritte unterbunden werden.

Bei der Art des Designs kann zwischen einem Factorial und einem Fractional Factorial Design gewählt werden, wobei das Fractional Factorial Design so klein wie möglich gewählt wird. Die erzeugenden Muster sind in der Konfigurationsdatei vorgegeben und müssen so nicht wiederholt berechnet werden. Sie sind so gewählt, dass ein Fractional Factorial Design für ein Modell erster Ordnung eine möglichst hohe Auflösung besitzt. Wenn zwei erzeugende Muster zur gleichen Auflösung führen, wird das erzeugende Muster gewählt, dass wenn man es für ein Modell zweiter Ordnung erweitert, eine höhere Auflösung ergibt. Sollte durch die Größe des Fractional Factorial Designs kein Modell erstellbar sein, kann dieses erweitert werden. Dabei wird das *k-p* Design zu einem *k-(p-1)* Design erweitert, was dazu führt, dass sich die Anzahl der Punkte verdoppelt. Dieses Verhalten kann aktiviert und auf eine maximale Anzahl von Erweiterungen beschränkt werden.

Neben diesen Einstellungen kann festgelegt werden, ob der LoF-Test bzw. CoD genutzt werden soll und welche Werte als ausreichend eingestuft werden. Die Verkleinerung des Suchbereichs kann wie in der Standardkonfiguration konstant bei *2* liegen. Es ist aber auch möglich für jeden Optimierungsschritt einen anderen Divisor festzulegen. Werden dabei mehr Optimierungsschritte durchgeführt, als Divisoren angegeben, wird der letzte eingetragene Divisor verwendet.

2.2 Anbindung externer Programme

Unterschiedliche Modelle können über den Parameter *Model* in der Konfigurationsdatei gewählt werden. Zum Test der RSM-Implementierung sind einige Funktionen implementiert, welche im Kapitel 3 beschrieben werden. Dieser Abschnitt befasst sich ausschließlich mit der Anbindung externer Modelle. Zur Zeit existieren Steuerungsskripte für HIT, DSPNexpressNG und NSolve. Die eingesetzte Skiptsprache ist Perl.

2.2.1 HIT und ProC/B

Der Standardpfad für die Skripte zur Steuerung von HIT [BMW89, BMW94] ist „HIT-Model“. Benötigt wird eine präparierte B1-Datei, bei der die Faktoren als `VARY_PARAMETER1`, `VARY_PARAMETER2`, … gekennzeichnet sind. Diese Zeichenketten werden durch das Skript gegen die korrekten Werten ausgetauscht und die Simulation durchgeführt. Um einen fortlaufenden Seed für die Generierung von Zufallszahlen zu generieren, müssen die in Abbildung 6 kursiv gedruckten Zeilen in die B1-Datei eingefügt werden. Der Name der B1-Datei muss in dem Skript `solve.pl` in der Variable

$modelname angegeben werden. Darüber hinaus müssen die zu betrachtenden Größen und die Berechnung des Responses angepasst werden (siehe Abbildung 5). Die B1-Dateien sind über das ProC/B Toolset [BBM02, BBS03] leicht zu erstellen, wodurch sich diese Anbindung besonders zur Optimierung von Prozesskettenmodellen eignet. Diese Anbindung wurde in [BMT05] zur Optimierung des Modells einer Stückgutumschlagshalle genutzt.

```
...
open(IN, "< t.$modelname.dum") or die "Unable to open t.$modelname.dum!\n";
while ($line = <IN>) {
    chomp $line;
    if ($line =~ /e777forklift THROUGHPUT .* h1truckload1 .* MEAN (.*)/) {
      $result1 = $1;
    }
    if ($line =~ /e902forklift THROUGHPUT .* h2truckload2 .* MEAN (.*)/) {
        $result2 = $1;
    }
    if ($line =~ /e1220forklif THROUGHPUT .* h3truckload3 .* MEAN (.*)/) {
        $result3 = $1;
    }
    if ($line =~ /e1226forklif THROUGHPUT .* h4truckload4 .* MEAN (.*)/) {
        $result4 = $1;
    }
    if ($line =~ /e1232forklif THROUGHPUT .* h5truckload5 .* MEAN (.*)/) {
        $result5 = $1;
    }
}
close(IN);
if ($logging>1) {
    print "done.\n";
}

$revenue = $result1 + $result2 + $result3 + $result4 + $result5;
...
```

Abbildung 5: Ausschnitt aus solve.pl für HIT

```
%COMMON
%PARM = WARN
%ANALYZER
%BIND "myseed" TO seed.txt
%END
%COPY "counter"
...
  OPEN ofile, "myseed" LENGTH 80;
  WRITE FILE ofile, LAST_SEED;
  CLOSE ofile;

END EXPERIMENT versuch;
```

Abbildung 6: (Optionale) Änderungen an der B1-Datei

2.2.2 DSPNexpressNG

Die Anbindung von Modellen für DSPNexpressNG [Lin03] erfolgt analog zur Anbindung von HIT-Modellen. Das zu verwendende Modell muss in der Variablen $modelname eingetragen werden. Die Variablen innerhalb des Modells müssen durch eindeutige Zeichenketten ersetzt werden. Diese Namen müssen in der entsprechenden Variable eingetragen werden und werden während der Auswertung durch die konkreten Werte ersetzt. Für den ersten Faktor enthält z.B. die Variable $factor1type die zu ersetzende

Zeichenkette. Zum Abschluss muss die Berechnung des Responses analog zum HIT-Model angepasst werden.

```
my $factor1name = "ServiceRate1";
my $factor1type = "DELAYPAR";
my $factor1value = $ARGV[1];

my $factor2name = "ServiceRate2";
my $factor2type = "DELAYPAR";
my $factor2value = $ARGV[2];
```

Abbildung 7: Parameternamen für DSPNexpressNG

2.2.3 APNN-Toolbox

Die Anbindung der APNN-Toolbox [BMF+05, BBK98] findet über das Tool NSolve statt. Dabei werden alle Konfigurationsmöglichkeiten unterstützt, die in der Anleitung zu NSolve beschrieben sind. Dabei liegt die Ausrichtung der Skripte auf der Auswertung von APNN-Modellen. Analog zu den vorhergehenden Programmen muss das APNN-Modell soweit modifiziert werden, dass die variablen Werte durch Zeichenketten der Form `VARY_PARAMTER1`, `VARY_PARAMETER2`, … und `COMP_PARAMETER1`, `COMP_PARAMETER2`, … ersetzt werden, wobei `VARY_PARAMETER?` den korrektren Faktorwert und `COMP_PARAMTER?` dessen Komplement zu `1` enthält. Der Name des Modells muss sowohl in der Datei `solve.pl` als auch in der Datei `solve2.pl` eingetragen werden. Die Aufteilung der Auswertung in zwei Skripte ist notwendig, um die Auswertung mit geringer Genauigkeit und einer Mindestzahl von Iterationen zu ermöglichen. Dazu ist die Abbruchgenauigkeit von NSolve im ersten Skript sehr gering gewählt. Die Abbruchgenauigkeit für die eigentliche Auswertung muss dementsprechend in der Datei `solve2.pl` eingestellt werden. Die nötigen weiteren Änderungen an den Skripten entsprechen denen für HIT und DSPNexpressNG. Die Anbindung der APNN-Toolbox wurde in [KMT05] genutzt und erweitert.

2.2.4 Anbindung weiterer Programme

Neben den drei bisher beschriebenen Programmen besteht die Möglichkeit weitere Programme anzubinden. Dafür ist in der Konfigurationsdatei der Wert „`EXTERNAL`“ für das gewählte Modell einzutragen. Daraufhin werden die Skripte im Unterverzeichnis „EXTERNAL“ zur Auswertung verwendet, die dem eingesetzten Programm entsprechend anzupassen sind. Dabei werden an das Skript `solve.pl` die nötigen Informationen in Form von Kommandozeilen-Parametern übergeben. Der erste Parameter ist ein Vorschlag eines Seeds für einen eingesetzten Zufallsgenerator. Die darauf folgenden Parameter sind die Faktorwerte in unkodierter Form. Das Ergebnis der Auswertung wird in der Datei „EXTERNAL.response“ in der Form `Revenue = <WERT>` erwartet.

3 Bewertung der RSM-Implementierung auf der Basis von Testfunktionen

Durch die zum jetzigen Zeitpunkt implementierte Funktionalität ergeben sich eine Reihe von Konfigurationsmöglichkeiten wie z.B. die Anzahl der berechneten Modelle zweiter Ordnung. Da die meisten Konfigurationsmöglichkeiten sich direkt auf die Anzahl der benötigten Auswertungen auswirken, ist es wichtig diesen Effekt bei der Untersuchung zu berücksichtigen. Die folgenden Abschnitte sollen Aufschluss darüber geben, wie gut die aktuelle RSM-Implementierung mit unterschiedlichen Situationen umgehen kann. Die Grundlage dieser Analyse sollen die folgende Abschnitte legen. Hier wird beschrieben unter welchen Bedingungen die Experimente durchgeführt, welche Funktionen zugrunde gelegt, und welche Konfigurationen getestet werden. Im Anschluss werden die erhaltenen Ergebnisse vermittelt und interpretiert.

3.1 Eingesetzte Funktionen und eingestellte Rahmenbedingungen

Die aktuelle Implementierung enthält eine Reihe von Funktionen. Zu diesen gehören acht unimodale und drei polymodale Funktionen, welche in [NP98] und [Sal98] beschrieben und in Tabelle 2 aufgeführt werden. Zur besseren Vergleichbar der Funktionen wurden einige Rahmenbedingungen festgelegt:

- Die Funktionen werden in einem Intervall von *[-1.5, 1.5*] je Faktor betrachtet.
- Innerhalb dieses Intervalls liegen die Funktionswerte weitestgehend im Intervall [*0, 1*]. Geringfügige Überschreitungen dieses Intervalls werden an den Grenzen toleriert.
- Das absolute Maximum hat einen Wert in der Nähe von *1*. Geringfügige Überschreitungen dieses Werts werden toleriert.

Diese Rahmenbedingungen erfordern eine Skalierung der Funktionen, wobei diese so gewählt ist, dass bei fester Anzahl von Faktoren die Skalierungsfaktoren konstant sind. Dies garantiert, dass die Funktionen qualitativ nicht beeinflusst werden. Im Weiteren werden die Funktionen in ihrer skalierten Form dargestellt, die in Tabelle 2 aufgeführten Formeln entsprechen jedoch den unskalierten Funktionen. Im Anhang 6.1 sind die in Tabelle 2 dargestellten Funktionen als dreidimensionaler Plot abgebildet. Dabei existiert für jede Funktion eine Abbildung des ungestörten Responses und eine Abbildung mit einer normalverteilten Störung $N(0, \sigma)$ mit $\sigma = 0{,}1$. Dies entspricht einer durchschnittlichen Störung von $\sigma \cdot \sqrt{2/\pi} \approx 0{,}8$ und damit ungefähr 8% der Größe des Intervalls der Responses.

Name	Formel
Ackley	$F(x) = -c_1 e^{-c_2\sqrt{\frac{1}{k}\sum_{i=1}^{k} x_i^2}} - e^{\frac{1}{k}\sum_{i=1}^{k}\cos(c_3 x_i)} + c_1; c_1 = 20; c_2 = 0,2; c_3 = 2\pi$
Ebene	$F(x) = \sum_{i=1}^{k} x_i$
Ellipsoid	$F(x) = \sum_{i=1}^{k} i x_i^2$
Griewank	$F(x) = 1 + \sum_{i=1}^{k} \frac{x_i^2}{4000} - \prod_{i=1}^{k} \cos\left(\frac{x_i}{\sqrt{i}}\right)$
Polynomfunktion	$y = \sum_{i=1}^{k} x_i + \sum_{i=1}^{k} x_i^2 + \sum_{i=1}^{k}\sum_{j=i+1}^{k} (x_i \cdot x_j)$
Rastrigin	$F(x) = \sum_{i=1}^{k} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$
Rosenbrock	$F(x) = \sum_{i=1}^{k-1} \left(100(x_i^2 + x_{i+1}) + (1 - x_i)^2\right)$
Schaffer[1]	$F(x) = \sqrt[4]{\sum_{i=1}^{k} x_i^2}\left(\sin^2\left(50 \cdot \sqrt[10]{\sum_{i=1}^{k} x_i^2}\right) + 1\right)$
Schwefel	$F(x) = \sum_{i=1}^{k}\left(\sum_{j=1}^{i} x_j\right)^2$
Sinus (Kosinus)	$F(x) = \prod_{i=1}^{k} \cos(x_i)$
Sphere	$F(x) = \sum_{i=1}^{k} x_i^2$
Step	$F(x) = \sum_{i=1}^{k} \lfloor x_i + 0,5 \rfloor^2$
Step 2	$F(x) = \sum_{i=1}^{k} \lfloor \lvert 2x_i \rvert + 0,5 \rfloor^2$

Tabelle 2: Testfunktionen

3.2 Durchgeführte Versuche und Bewertungsgrundlagen

Durch die hier durchgeführten Experimente sollen verschiedene Ziele erreicht werden. Primär soll erkannt werden, wie gut RSM im Idealfall, d.h. auf der Basis ungestörter Responses, arbeitet. Zweitens soll der Einfluss der Störung auf die Güte des Ergebnisses beobachtet werden und drittens soll beobachtet werden, welchen Einfluss die Anzahl der Auswertungen

[1] Die zu grunde gelegte Schaffer-Funktion ist ursprünglich nur für zwei Faktoren definiert. Die aufgeführte Funktion entspricht im zweidimensionalen Fall der ursprünglichen Definition, ist aber nicht auf zwei Faktoren beschränkt.

auf das Ergebnis von RSM hat. Dazu wurden für verschiedene Anzahlen von Faktoren (*2, 4* und *8* Faktoren) Experimentserien für jeweils drei unterschiedliche Konfigurationen durchgeführt. Jede Experimentserie bestand aus *21* Experimenten, wobei die stochastische Störung schrittweise um *0,005* von *0* auf *0,1* erhöht wurde. Die Konfigurationen sind wie folgt gewählt:

1. Fractional Factorial Design mit drei Auswertungen je Centerpoint und einer Auswertung je Designpoint
2. Fractional Factorial Design mit einer Auswertung je Center- und Designpoint
3. Factorial Design mit drei Auswertungen je Centerpoint und einer Auswertung je Designpoint

Betrachtet werden dabei die folgenden Messwerte:

- Anzahl der Auswertungen
- Funktionswerte des besten gefundenen Punkts (Response)
- Distanz des besten gefundenen Punkts zum optimalen Punkt

Um die Distanz zum optimalen Punkt bei unterschiedlicher Anzahl von Faktoren besser vergleichen zu können, wurde die absolute berechnete Distanz durch die halbe Länge der Diagonalen durch den Suchraum geteilt. Dadurch ergibt sich für die relative Distanz ein von der Anzahl der Faktoren unabhängiger Wert im Intervall [*0; 2*].

3.3 Ergebnisse

Die Betrachtung der Ergebnisse gliedert sich in drei Teile. Im ersten Teil wird die Anzahl der benötigten Auswertungen untersucht. Im zweiten Teil werden die Ergebnisse bzgl. Response und Distanz je Funktionengruppe untersucht. Allgemeine Erkenntnisse, die für alle Funktionen gelten, werden im dritten Teil erläutert. Die abgebildeten Ergebnisse beziehen sich jeweils auf die Distanz zum Rand des Optimalen Bereichs bzw. auf die Distanz zum globalen Optimum sowie auf den korrekten Response der ermittelten Faktorenbelegung. Die dargestellten Ergebnisse sind dabei der Durchschnitt über *100* RSM-Durchläufe mit zufällig gewähltem Startpunkt.

3.3.1 Betrachtung der Anzahl benötigter Auswertungen

Weitestgehend unabhängig von der Art der gewählten Funktion ergeben sich eine Reihe von Kurven, wie sie in Abbildung 8 dargestellt sind, wobei mit der Größe der Störung die Anzahl der benötigten Auswertungen leicht fällt. Dies erklärt sich durch eine geringere Anzahl erfolgreicher Optimierungsschritte. Durch die Störung sind vor allem die letzten Optimierungsschritte mit höherer Wahrscheinlichkeit erfolglos. Dies hat die folgenden Ursachen:

1. Richtung des steilsten Anstiegs wird nicht korrekt bestimmt

   ➔ Potentielle neue Centerpoints ergeben keinen besseren Response

2. Der gemessene Response des aktuellen Centerpoints ist auf Grund der Störung erhöht

   ➔ Trotz höherem ungestörten Response werden neue Centerpoints verworfen

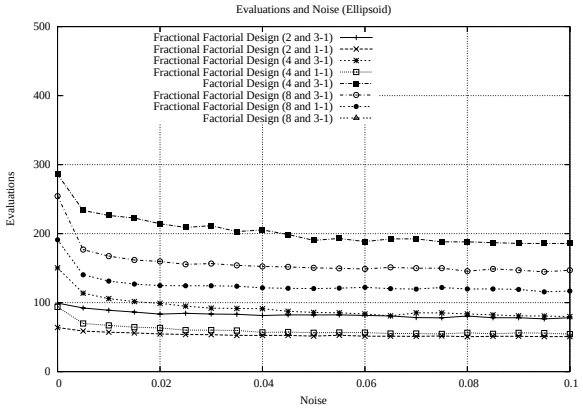


Abbildung 8: Anzahl benötigter Auswertungen (Ellipsoid-Funktion)[2]

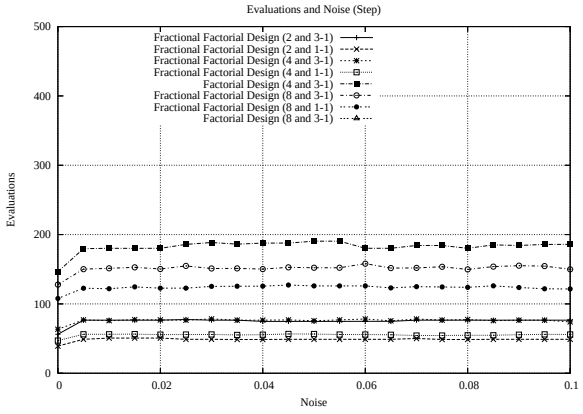


Abbildung 9: Anzahl benötigter Auswertungen (Step-Funktion)[2]

Diese Ergebnisse werden nur von den Stufenfunktionen nicht bestätigt. Hier ist eine Verbesserung des durchschnittlich gefundenen Responses durch eine Störung zu beobachten (siehe Abbildung 9). Dieses Verhalten lässt sich jedoch dadurch erklären, dass Fall eines ungestörten Responses RSM keine besseren Centerpoints finden kann, sobald die Distanz zur nächsten Kante eines Platos größer ist als die halbe Breite des Suchbereichs. Durch die Störung der Responses können Optimierungsschritte innerhalb eines Platos erfolgreich sein, wenn die Summe der Störungen eines neuen Centerpoints größer ist als die des aktuellen Centerpoints. Da jeder erfolgreiche Optimierungsschritt die Anzahl der Optimierungsschritte je RSM-Durchlauf erhöht, erhöht sich so auch die Anzahl der Auswertungen.

3.3.2 Betrachtung von Response und Distanz

In den folgenden Abschnitten wird das Ergebnis der RSM-Durchläufe bei steigender Störung betrachtet. Dabei wird der Schwerpunkt auf die beiden ersten Ziele dieses Benchmarks gelegt. Zur besseren Beurteilung der Beobachtungen werden die Funktionen gruppenweise betrachtet. In der ersten Gruppe sind die Funktionen Sphere, Step 2 und Step enthalten. Die Funktion Sphere dient dabei als Referenzfunktion, da sie als einfache polynomielle Funktion zweiten Grades für Optimierung mittels RSM besonders geeignet sein sollte. Über die Funktionen Step und Step 2 soll dabei beurteilt werden, wie robust RSM gegenüber Platos ist.

[2] Die Anzahl der Auswertungen der Funktionen mit acht Faktoren und Factorial Design wird in der Abbildung nicht dargestellt, da diese unabhängig von der gewählten Funktion deutlich über 2000 liegt.

Die zweite Gruppe wird aus den Funktionen Polynom, Schwefel, Griewank und Sinus gebildet. Hier dient die Polynomfunktion als Referenzfunktion und es soll beurteilt werden, welchen Einfluss die Größe eines flachen Bereichs in der Nähe des Optimums auf die Resultate hat. Die dritte Gruppe enthält drei multimodale Funktionen, wobei Ackley über schwach ausgeprägt lokale Optima und die beiden Funktionen Rastrigin und Schaffer über stark ausgeprägte lokale Optima verfügt.

3.3.2.1 Sphere, Step 2 und Step

Die in Abbildung 10 dargestellten Ergebnisse bzgl. der Sphere-Funktion entsprechen den Erwartungen. Der Response fällt mit steigender Störung geringfügig, so dass beim Einsatz von Factorial Designs der Response im Durchschnitt nicht unter *0,95* liegt. Durch die Einsparung an Auswertungen durch den Einsatz von Fractional Factorial Designs erhöht sich der Einfluss der Störung auf das Ergebnis. Der Response fällt jedoch nicht unter *0,8.* Die Ergebnisse der Step-Funktion 2 (siehe Abbildung 11) entsprechen im Wesentlichen denen der Sphere-Funktion. Dies zeigt, dass RSM robust gegenüber kleinen Platos ist. Dahingegen zeigen die Ergebnisse der Step-Funktion (siehe Abbildung 12) deutlich, dass RSM durch große Platos deutlich schlechtere Resultate liefert.

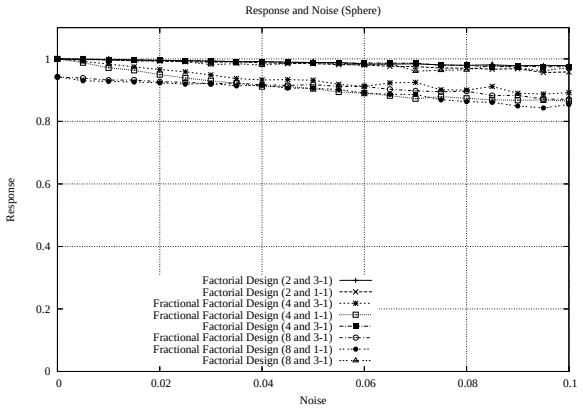


Abbildung 10: Störung zu Response (Sphere-Funktion)

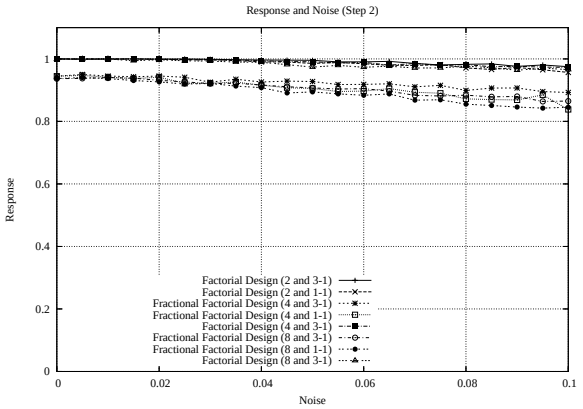


Abbildung 11: Störung zu Response (Step-Funktion 2)

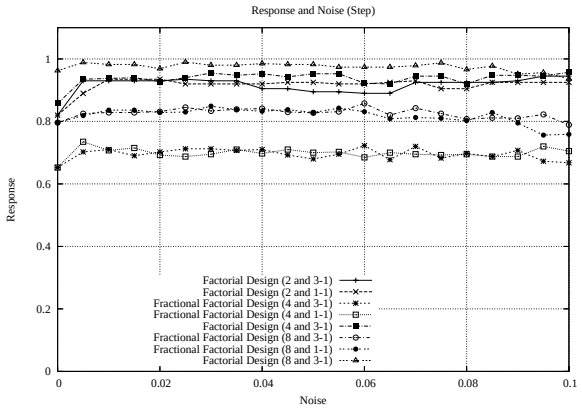


Abbildung 12: Störung zu Response (Step-Funktion)

3.3.2.2 Polynom, Schwefel, Griewank und Sinus

Für die Funktionen Polynom und Schwefel ergeben sich überdurchschnittlich gute Ergebnisse (siehe Abbildung 13 und Abbildung 14). Mit steigender Störung fällt der Response nur sehr geringfügig und liegt im Durchschnitt deutlich über *0,9*. Für diese Funktionen liefert RSM sehr gute Ergebnisse. Im Falle eines ungestörten Responses wird in fast allen RSM-Druchläufen das Optimum gefunden. Dies gilt bemerkenswerter Weise auch für die RSM-Durchläufe mit Fractional Factorial Designs. Die Ergebnisse von RSM bzgl. der Griewank-Funktion (siehe Abbildung 15) sind deutlich stärker von der Störung des Responses abhängig. Diese Abhängigkeit ist beim Einsatz von Factorial Designs deutlich geringer als beim Einsatz von Fractional Factorial Designs. Darüber hinaus ist festzustellen, dass im Falle von Funktionen mit acht Faktoren und dem Einsatz von Fractional Factorial Designs bereits bei ungestörten Responses eine deutliche Abweichung vom Optimum besteht. Noch deutlicher ist der Einfluss der Störung bei der Sinus-Funktion (siehe Abbildung 16). Während die Factorial Designs nur geringfügig durch die Störung beeinflusst werden, fallen die Ergebnisse für Fractional Factorial Designs stark ab.

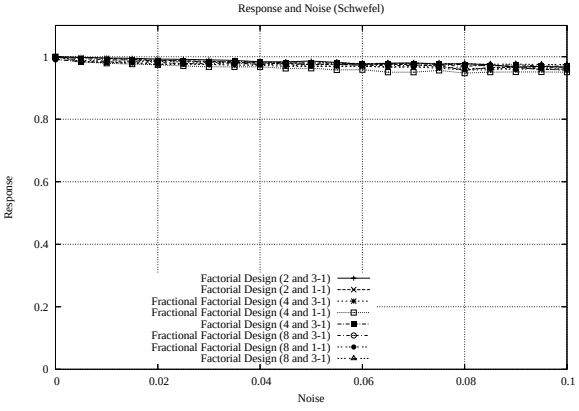


Abbildung 13: Störung zu Response (Schwefel)

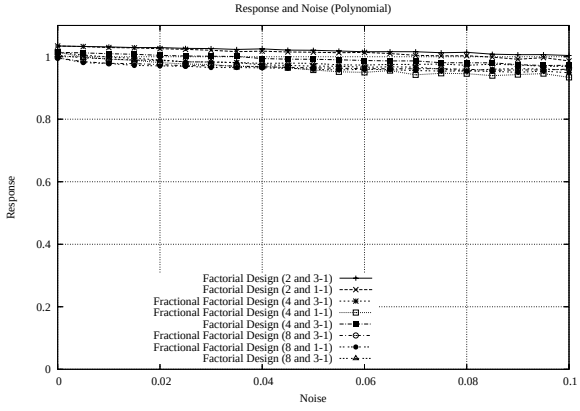


Abbildung 14: Störung zu Response (Polynom)

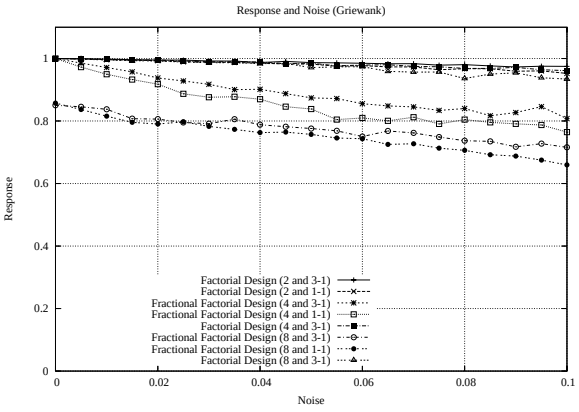


Abbildung 15: Störung zu Response (Griewank)

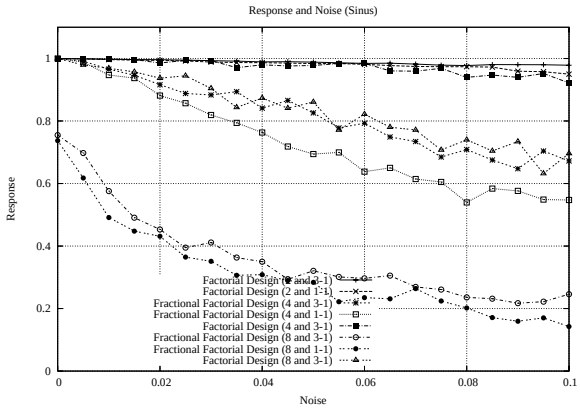


Abbildung 16: Störung zu Response (Sinus)

Diese Ergebnisse zeigen, dass Factorial Designs besser geeignet sind, um Optima von Funktionen vor allem bei steigender Störung zu erreichen. Dabei sind die Ergebnisse

abhängig von der Art der Funktion. Wie in Abbildung 17 und Abbildung 18 zu sehen, steigt die Distanz zum Optimum für die Polynom-Funktion deutlich stärker als die zum Optimum für die Griewank-Funktion. Dies zeigt, dass Funktionen, die wie die Polynom-Funktion einen breiten und flachen Bereich in der Nähe des Optimums aufweisen, bzgl. RSM gutmütiger sind als Funktionen mit einen schmalen Bereich in der Nähe des Optimums. Daraus ergibt sich, dass der Einsatz von Fractional Factorial Designs auch bei starker erwarteter Störung durchaus sinnvoll sein kann, wenn ein breiter Bereich in der Nähe des Optimums hohe Responses liefert, da in diesem Fall geringfügig schlechtere Ergebnisse zu erwarten sind als beim Einsatz von Factorial Designs, jedoch die Anzahl benötigter Auswertungen deutlich gesenkt werden kann.

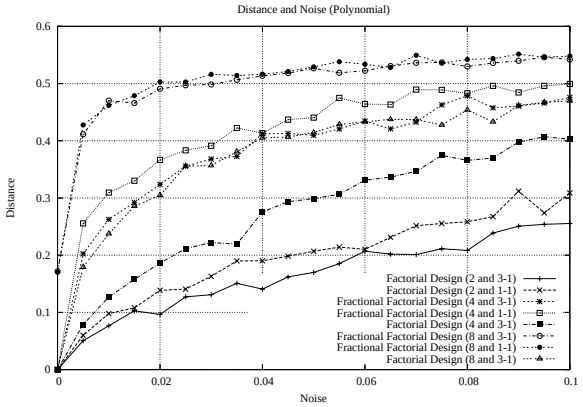


Abbildung 17: Störung zu Distanz vom Optimum (Polynom)

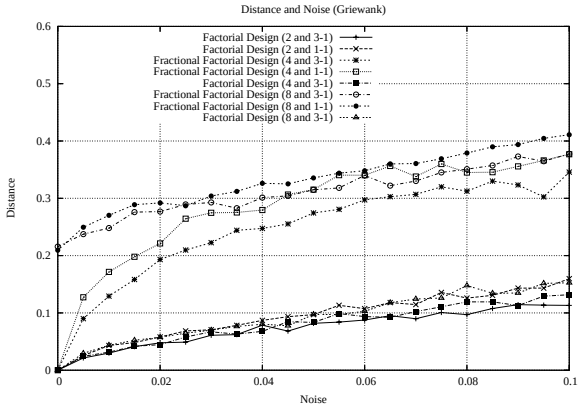


Abbildung 18: Störung zu Distanz vom Optimum (Griewank)

3.3.2.3 Ackley, Schaffer und Rastrigin

Ackley, Schaffer und Rastrigin sind multimodale Funktionen. Ziel der Optimierung mittels RSM ist es, den Response zu maximieren bzw. minimieren. Dabei ist es unerheblich, ob die gefundenen Faktorwerte in der Nähe des globalen Optimums liegen, oder ob ein gutes lokales Optimum gefunden wurde. Die Ergebnisse der Ackley-Funktion (siehe Abbildung 46) zeigen deutlich, dass RSM mit steigender Störung schlechtere Ergebnisse liefert. Dabei zeichnen sich die RSM-Durchläufe mit Factorial Design durch deutlich bessere Ergebnisse aus. Die Ergebnisse der Schaffer-Funktion sind weitestgehend unbeeinflusst von der Höhe der Störung (siehe Abbildung 47) und fallen nur geringfügig ab. Dies lässt sich durch die steile Wellenform erklären, da nur in der Nähe des Optimums die Störung relativ zur Steigung groß ist. Darüber hinaus führen auch durch eine Störung umgeleitete Suchen entlang des (gemessenen) steilsten Anstiegs in Richtung eines Wellenkamms. Schlechte Ergebnisse ergeben sich beim Einsatz von Fractional Factorial Designs. Dies ist vor allem auf die schmalen Wellenkämme der Funktion zurückzuführen, die durch den Verzicht auf zusätzliche Designpunkte nicht ausreichend korrekt im Model erfasst werden. Das Verhalten von RSM bzgl. der Rastrigin-Funktion (siehe Abbildung 48) entspricht im Wesentlichen dem

vorhergehenden. Aufgrund der breiteren Bereiche in der Nähe der lokalen Optima sind die Ergebnisse jedoch leicht besser.

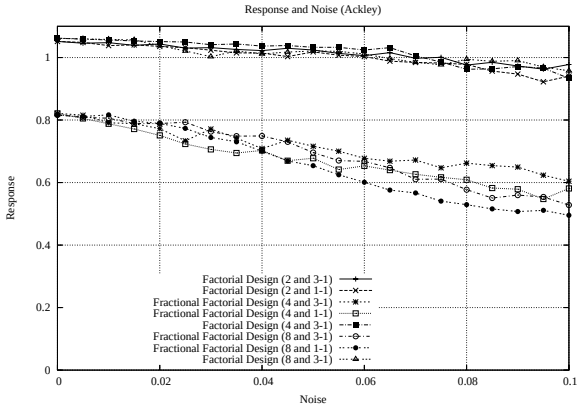


Abbildung 19: Störung zu Response (Ackley)

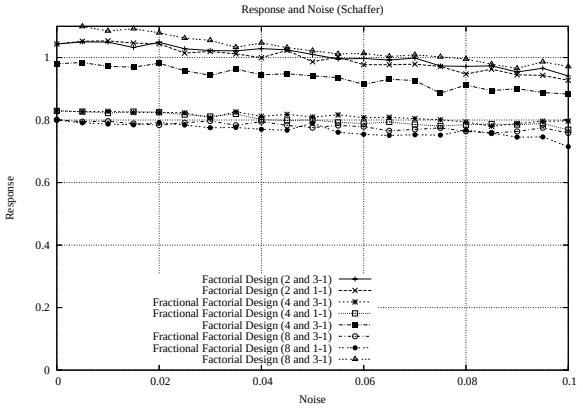


Abbildung 20: Störung zu Response (Schaffer)

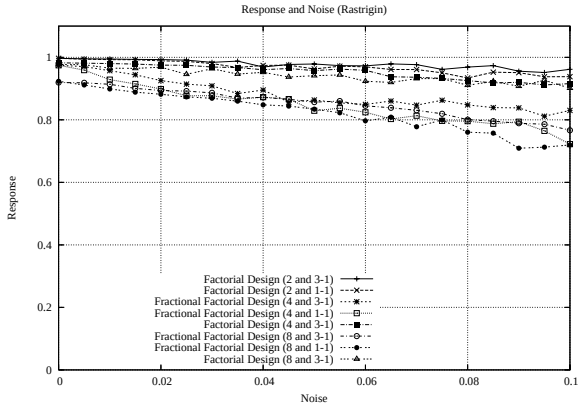


Abbildung 21: Störung zu Response (Rastrigin)

3.4 Zusammenfassung

Es hat sich gezeigt, dass Fractional Factorial Designs gegenüber Factorial Designs ein deutliches Einsparungspotential bzgl. der Anzahl der Auswertungen bieten. Dabei muss jedoch abhängig von der untersuchten Funktion ein Verlust in der Güte des Ergebnisses akzeptiert werden. Wenn bereits Vorkenntnisse über den Verlauf der Funktion bestehen, die nahe legen, dass die Funktion bzgl. des Einsatzes von Fractional Factorial Designs gutmütig ist, ist deren Einsatz zu empfehlen. Grundsätzlich gilt jedoch, dass Factorial Designs bessere Ergebnisse liefern.

Des weiteren hat es sich gezeigt, dass die Anzahl der Auswertungen nur geringfügig von der gewählten Funktion abhängt. Anhand der Zahlen aus Abbildung 8 kann die voraussichtlich benötigte Anzahl von Auswertungen abgeschätzt werden. Ist darüber hinaus die durchschnittliche Dauer einer Auswertung bekannt, so kann der gesamte Zeitaufwand für eine Optimierung mittels RSM abgeschätzt werden.

4 Anwendungsbeispiel

Anhand eines ausgewählten Anwendungsbeispiels soll die Integration von externen Programmen demonstriert werden, wie sie in Unterkapitel 2.2.4 beschrieben wird. Bei diesem Beispiel handelt es sich um zwei in Reihe geschaltete *M/M/1/K*-Warteschlangen. Die Kapazität der Warteschlangen beträgt $K = 10$, wobei überzählige Anfragen verworfen werden. Die durchschnittliche Ankunftsrate beträgt $\lambda = 0.5$. Die Serviceraten w_1 und w_2 der beiden Warteschlangen sind variabel. Sie sollen so gewählt werden, dass $R(w_1, w_2)$ maximal wird:

$$R(w_1, w_2) = 100.000€ \cdot X(w_1, w_2) - 10.000€ \cdot w_1 - 30.000€ \cdot w_2 \quad \textbf{(16)}$$

Dabei entspricht $X(w_1, w_2)$ dem Durchsatz der zweiten Warteschlange, also der Anzahl der produzierten Güter. Die Einnahmen je Stück liegen bei *100.000 €*. Die Kosten für den Betrieb der ersten Warteschlange liegen sind mit einem Faktor von *10.000 €* proportional zur Betriebsgeschwindigkeit. Die Kosten für die zweite Warteschlange verhalten sich analog mit einem Faktor von *30.000 €*. In Abbildung 22 ist die korrekte Response-Surface für das Tandemqueuebeispiel dargestellt.

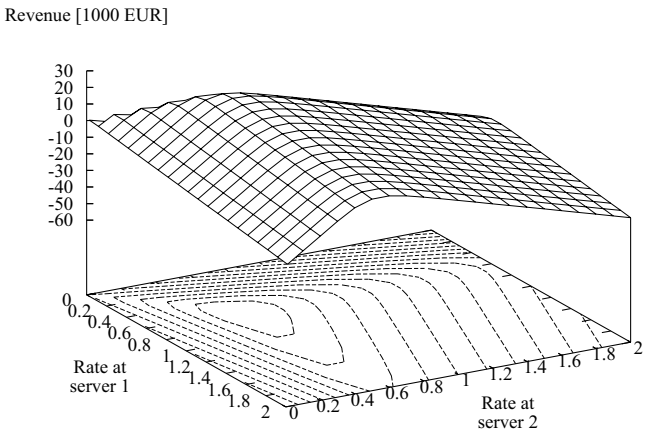


Abbildung 22: Response Surface des Tandemqueue-Modells

In den folgenden Unterkapiteln wird die Modellierung der Tandemqueue als ProC/B- und APNN-Modell, sowie die Ergebnisse der aktuellen RSM-Implementierung beim Einsatz dieser Modelle beschrieben.

4.1 Modellierung mit ProC/B

Die Modellierung der Tandemqueue als ProC/B-Modell wurde für [BMT05] durchgeführt, die folgenden Ausführungen lehnen sich an diesen Beitrag an. In Abbildung 23 ist die Tandemqueue als ProC/B-Modell dargestellt, wobei die gelb hinterlegten Elemente die erste

Warteschlange und die grün hinterlegten Elemente die zweite Warteschlange darstellen. Die blau hinterlegten Elemente dienen der Messung des Durchsatzes des Systems.

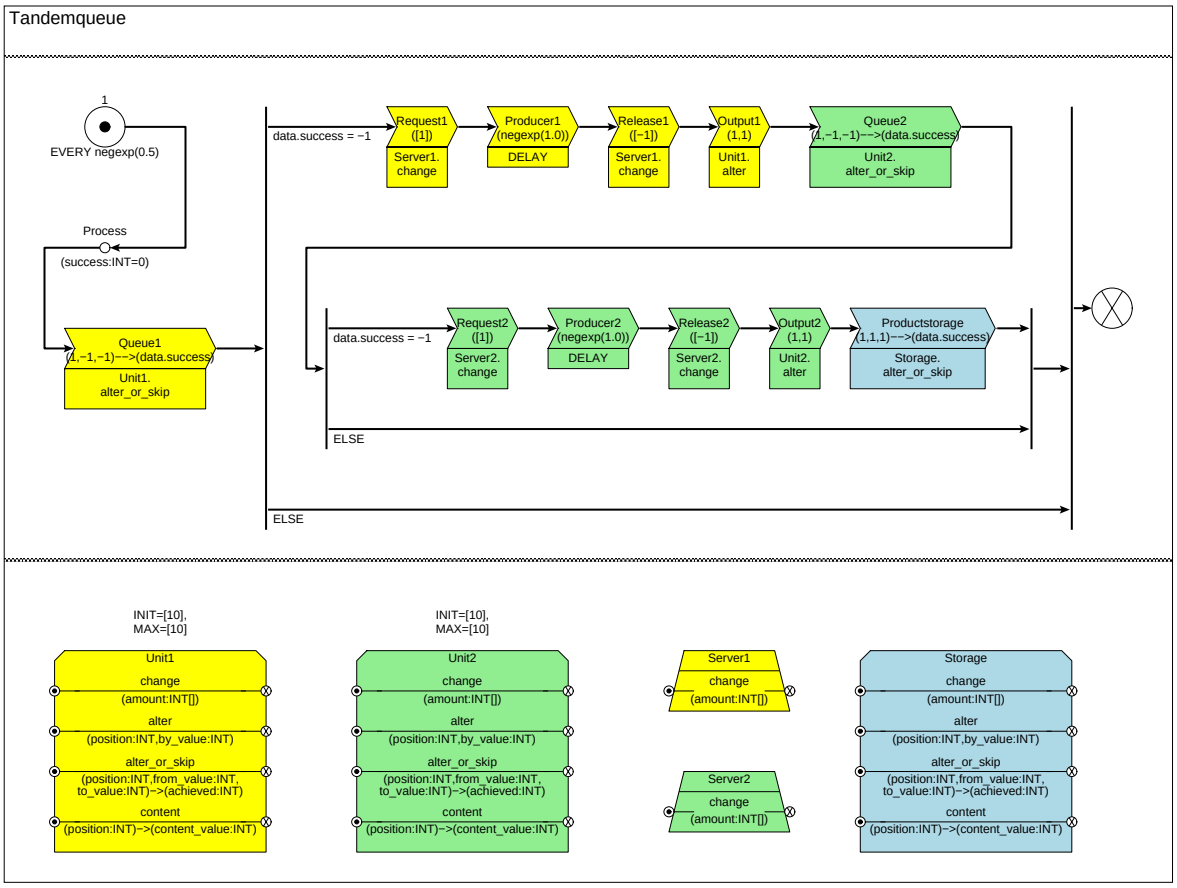


Abbildung 23: ProC/B-Modell einer Tandemqueue

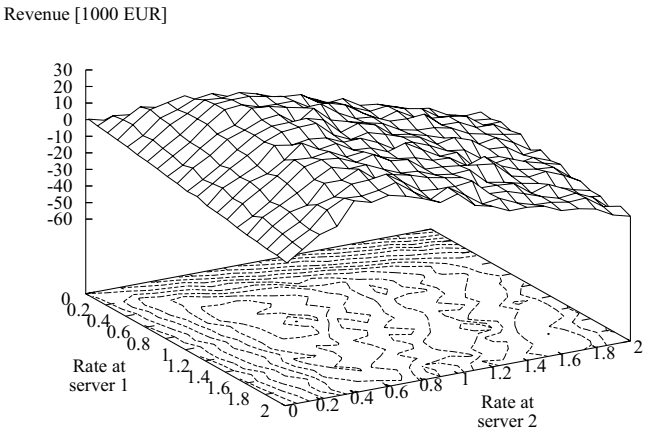


Abbildung 24: Beobachtete Response Surface des Tandemqueue-Modells

In Abbildung 24 ist die Response-Surface dargestellt, die auf der Basis des Modells aus Abbildung 23 und einer beschränken Genauigkeit des Simulators und damit mit einer vertretbaren Rechenzeit ermittelt wurde. Die erkennbaren Schwankungen sind dabei das

zentrale Problem des Optimierers, welches in Abbildung 25 zu erkennen ist. In diesem Diagramm wurde für vier verschiedene RSM-Konfigurationen, wobei sich diese nur in der Anzahl der Centerpoint- und Designpoint-Auswertungen unterscheiden, die Distanz zwischen korrektem Optimum und dem berechneten Ergebnis ermittelt. Dabei wurde zur Verbesserung der Ergebnisse die Anzahl der Replikationen je Auswertung bis auf *20* erhöht. Dies bedeutet, dass je Auswertung eines Punkts für RSM dieser Punkt *20* mal simuliert wurde. Der Durchschnitt dieses Ergebnisses wurde dann als Ergebnis der Auswertung interpretiert. Auf diese Weise konnten die Ergebnisse trotz der schlechten einzelnen Simulationsergebnisse verbessert werden. Dies führte letztendlich zu einer Verbesserung des Gesamtergebnisses.

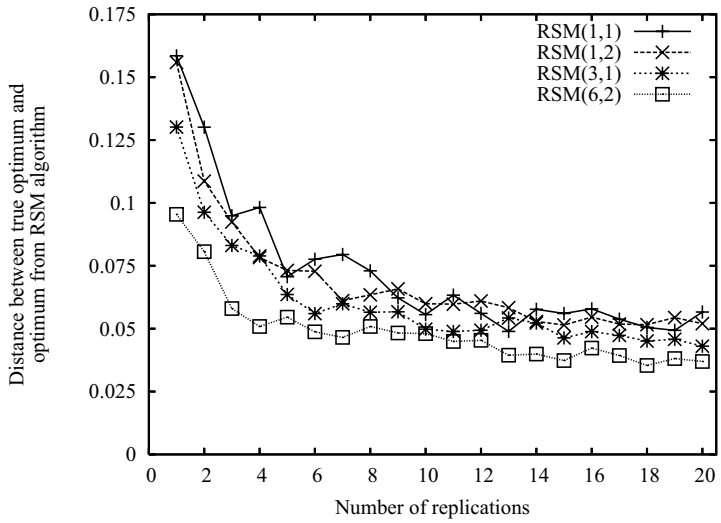


Abbildung 25: Distanz des Optimums von der besten ermittelten Konfiguration

4.2 Modellierung mit der APNN-Toolbox

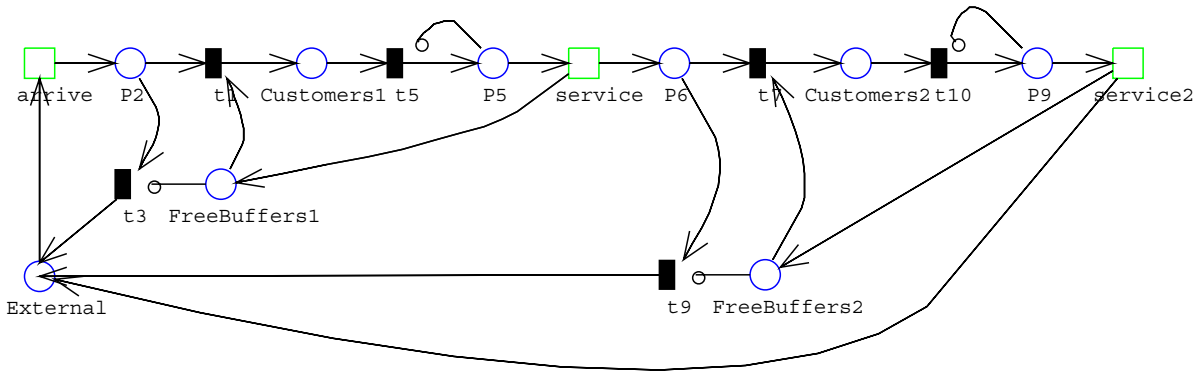


Abbildung 26: APNN-Modell einer Tandemqueue

In Abbildung 26 ist die Tandemqueue als APNN-Modell dargestellt. Da zur Auswertung von APNN-Modellen NSolve also ein numerischer Lösungsansatz verwendet wird, ergeben sich einige Unterschiede zur Simulation mittels ProC/B und HIT. Eine Replikation der Auswertungen ist nicht sinnvoll, da jeweils als Ergebnis der gleiche Wert berechnet wird. Daher ist die einzige Möglichkeit, die Ergebnisse zu Verbessern eine Erhöhung der

Abbruchgenaugigkeit des numerischen Lösers. Darüber hinaus lässt sich Güte der berechneten Ergebnisse nicht anhand der Schwankung innerhalb der Response-Surface erkennen, da aufgrund der eingesetzten Methode ähnliche Konfigurationen zu ähnlichen Ergebnissen führen. Allerdings kann man in einigen Fällen eine Veränderung der Response-Surface feststellen, die im Vergleich zur korrekten Response-Surface insbesondere an der Ausprägung und der Position lokaler Optima erkennbar ist. In Abbildung 27 ist die Response-Surface abgebildet, die bei einer Genauigkeit von 10^{-2} als Abbruchbedingung für den numerischen Löser entsteht. Hier ist zu erkennen, dass das Optimum in Richtung des Punkts *[2, 2]* verschoben ist. Für den Einsatz von RSM sollte also eine höhere Abbruchgenauigkeit gewählt werden.

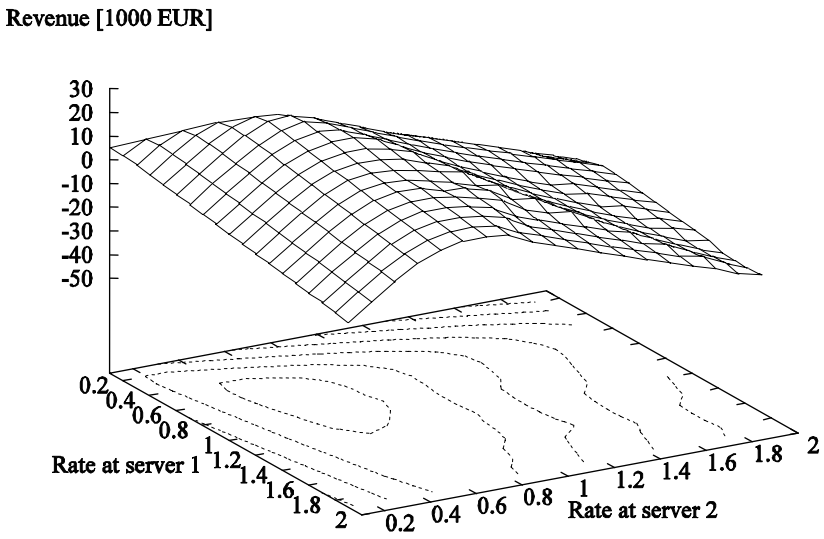


Abbildung 27: Beobachtete Response Surface des Tandemqueue-Modells

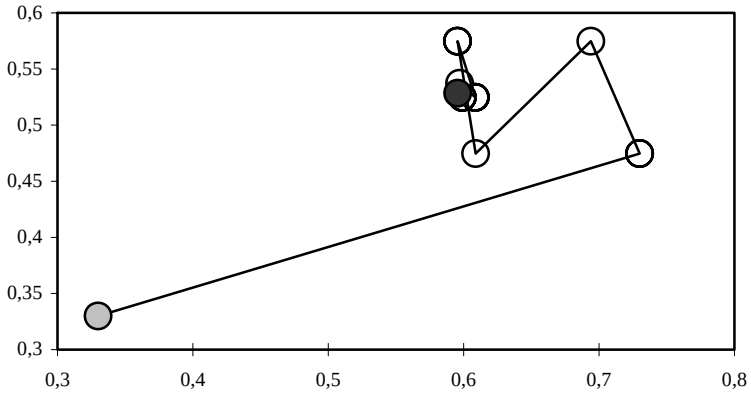


Abbildung 28: Schritte eines RSM-Durchlaufs

Der in Abbildung 28 dargestellt RSM-Durchlauf wurde mit der gleichen Grundkonfiguration durchgeführt, die auch im letzten Unterkapitel eingesetzt wurde. Dabei ist der Startpunkt hellgrau und der beste gefundene Punkt dunkelgrau gekennzeichnet. Aufgrund der Eigenschaften des eingesetzten Auswertungsverfahrens wurde die

Konfiguration RSM(1, 1) mit einer Centerpoint- und einer Designpoint-Auswertung genutzt. Die Ergebnisse jeder Auswertung wurden mit NSolve ermittelt, wobei die Genauigkeit zum Abbruch mindestens 10^{-6} betrug. Abbildung 28 zeigt weiteres typisches Verhalten von RSM. Da der Bereich mit Faktorwerten über *0,5* flacher ist als die restlichen Bereiche, erreicht RSM in den ersten Schritten diesen Bereich und nähert sich von dort dem Optimum. Eine weitere Eigenschaft von RSM ist es, das Optimum zu überspringen. Dieses Verhalten tritt vor allem dann auf, wenn die zur Zeit gewählte Schrittweite nicht mehr als doppelt so groß wie die Distanz zum Optimum ist. Insgesamt ist gut zu erkennen, dass sich RSM mit jedem Schritt dem Optimum nähert.

5 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche (GUI) wurde entwickelt um die Optimierung mit Hilfe von RSM möglichst einfach zu gestalten. Sie soll dem Optimierer helfen, sinnvolle Parameter für RSM zu finden. Des weiteren hilft die GUI auch bei der Auswertung der Optimierungsläufe. Hierzu wurden die einzelnen Optionen des RSM auf verschiedene Ansichten aufgeteilt. Diese Ansichten trennen die Einstellungen in vorausgesetzte- und optionale Optionen auf. Zu den nicht optionalen Einstellungen gehört, z.B. der Wertebereich des Modells, daher ist dieser auf der ersten Ansicht aufgeführt.

Dieses Kapitel gliedert sich in verschieden Teilaspekte. Zuerst wird behandelt, wie sich die Oberfläche starten lässt, dann folgt ein Abschnitt über die einzelnen Teile der GUI. Darauf folgen die Beschreibungen der einzelnen Ansichten.

5.1 Start des Programms

Die GUI befindet sich im Wurzelverzeichnis, welches bei der Installation erstellt wurde (siehe hierzu Kapitel 6.4).Um die GUI zu starten gibt es verschiedene Methoden. Mit einer korrekt eingerichteten Java Umgebung reicht ein Doppelklick auf die Datei *„rsmOberflaeche.jar“*. Sollte dies nicht zum Erfolg führen so existieren für Windows und Linux separate Startskripte. Diese können entweder durch doppelklicken oder durch den Aufruf, der Datei, auf einer Konsole gestartet werden. Für Linux ist dies der Befehl *„./start.sh“*. Für Windows ist der korrespondierende Befehl *„start.bat“*. Sollten diese Methoden nicht funktionieren so ist das Starten der GUI auch direkt über den Befehl *„java –jar rsmOberflaeche.jar“* möglich.

Die GUI gliedert sich in drei wesentliche Teile. Am oberen Rand befindet sich das Menü, über welches Funktionen wie „laden“, „speichern“ und „schließen“ zugänglich sind. Auf der linken Seite unter dem Menu befindet sich die Navigationsleiste, über welche die verschiedenen Ansichten erreichbar sind. Der Navigationsleiste gegenüber liegt auf der rechten Seite der Bereich auf dem die Optionen angezeigt und bearbeitet werden können, die zur aktuellen Ansicht gehören.

5.2 Das Menü

Im Menueintrag "File" (siehe Abbildung 29) befinden sich verschiedene Optionen, die das Programm steuern. Unter Menupunkt "Load Config" verbirgt sich die Möglichkeit eine andere Konfigurationsdatei für das RSM einzulesen. Durch die beiden Einträge "Save Config" und "Save Config as" sind verschiedene Speichermöglichkeiten realisiert. Im ersten Fall werden die aktuellen Einstellungen in die "rsm.config" Datei im Verzeichnis „rsm“ gespeichert. Im zweiten Fall öffnet sich ein Speicherndialog in dem ein Dateiname angeben werden kann, an dem die Datei gespeichert werde soll. Diese kann dann später mittels "Load Config" wieder eingelesen werden.

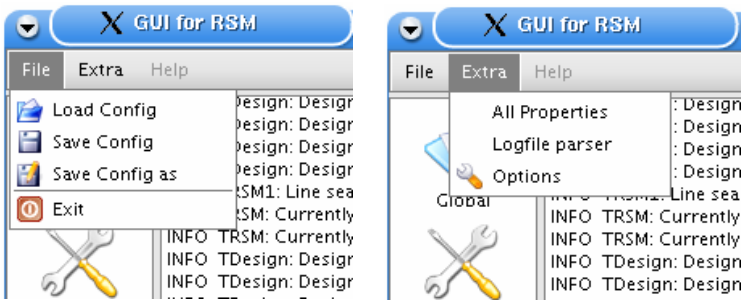


Abbildung 29: Menü

Mit Exit wird das Programm beendet, wobei ein Dialog angezeigt wird in welchem abgefragt wird, ob das Programm wirklich zu schließen ist und ob vorher alle Einstellungen gespeichert werde sollen (siehe hierzu Unterkapitel 5.4).

Der Menueintrag "Extra" beinhaltet Sonderfunktionen wie die Anzeige aller Optionen welche sich in der Konfigurationsdatei befinden, dies geschieht mittels "All Properties". Ist diese Funktion auswählt erscheint in einem weiteren Fenster eine Tabelle in der alphabetisch sortiert alle Optionen, die in der Konfigurationsdatei vorkommen, angezeigt werden. Hier können auch Optionen verändert werde die sonst nicht im Programm zu sehen sind, da in der GUI nur die wichtigsten Optionen einstellbar sind. Durch klicken des Buttons "Save" werden die Einstellungen gespeichert und mittels "Close" der Dialog geschlossen. Der zweite Menueintrag "Logfile Parser" öffnet einen Dialog in welchem eine Logdatei ausgewählt werden kann. Der Inhalt dieser Datei wird auf dem erscheinenden RSM Output Fenster grafisch dargestellt (siehe Abbildung 36).

5.3 Die Panels

In den nun folgenden Abschnitten werden die einzelnen Panels beschrieben. Diese dienen zur Anzeige der verschiedenen Ansichten und können über die Navigationsleiste erreicht werden. Durch klicken eines Buttons auf der Navigationsleiste im linken Bereich der GUI, wird das zu dieser Ansicht gehörende Panel im rechten Bereich der GUI angezeigt.

5.3.1 Global Panel

Nach dem Start präsentiert die GUI zuerst die Globaleinstellungen, welche auch über den Button "Global" der Navigationsleiste erreicht werden können (siehe Abbildung 30). Auf dieser Seite können nun die grundlegenden Einstellungen getroffen werden. Über die Zahlenfelder "Number of cont. Factors" und "Number of disc. Factors" wird die Summe der Faktoren, mit denen RSM arbeiten soll, eingestellt. Sie werden in diskrete und kontinuierliche Werte unterschieden. Diskrete Werte werden allerdings noch nicht von RSM unterstützt.

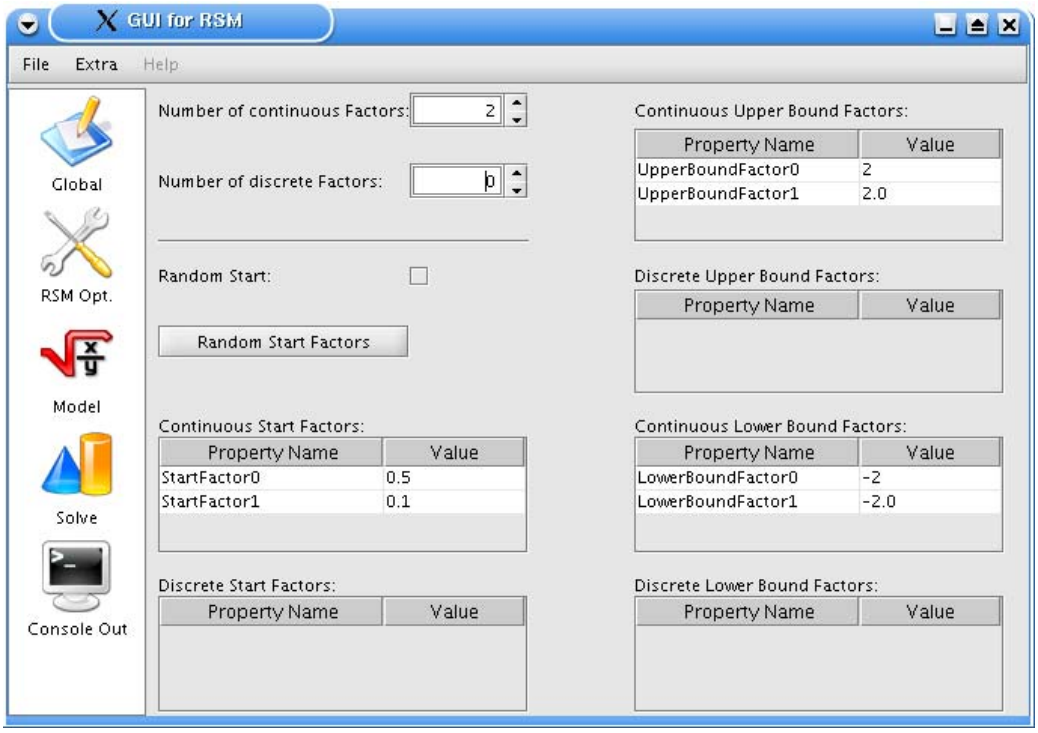


Abbildung 30: Global Panel

Den gewählten Faktoren werden über die Tabellen "Upper Bound" und "Lower Bound" Wertebereiche zugewiesen. Die Werte in den "Upper Bound" Tabellen stehen für den höchsten und der „Lower Bound“ Wert für den niedrigsten Wert, den ein Faktor einnehmen kann. In den Tabellen "Start Factors" werden die Startwerte der Faktoren eingestellt. Durch aktivieren der Checkbox "Random Start" wird das RSM angewiesen bei jedem Durchlauf neue Zufallswerte für die Startwerte zu erzeugen. Wenn diese Option aktiv ist, ist es nicht mehr möglich die Startwerte in den Tabellen "Start Factors" zu editieren. Wenn nicht bei jedem Durchlauf neue Werte erzeugt werde sollen, aber auch keine Startwerte vorliegen, so können durch drücken des Buttons "Random Start Factors" Zufallswerte für diese erzeugt werden. Die so erzeugten Werte erscheinen in den jeweiligen Tabellen und können editiert werden.

5.3.2 RSM Opt. Panel

Durch klicken auf den "RSM Opt." Button wird die zweiten Ansicht angezeigt (siehe Abbildung 31), welche nun Optionen präsentiert, die das RSM direkt beeinflussen. Bei den meisten Werten bedeutet ein Wert von „-1“ das die jeweiligen Funktionen deaktiviert sind. Die Einstellungen "Center Evaluations" und "Design Evaluations". Diese geben an wie oft die Centerpoints und die sie umgebenen Designpoints ausgewertet werden sollen. Höhere Werte sind sinnvoll wenn ein hoher Fehler zu erwarten ist, gehen aber auf Kosten der Berechnungszeit.

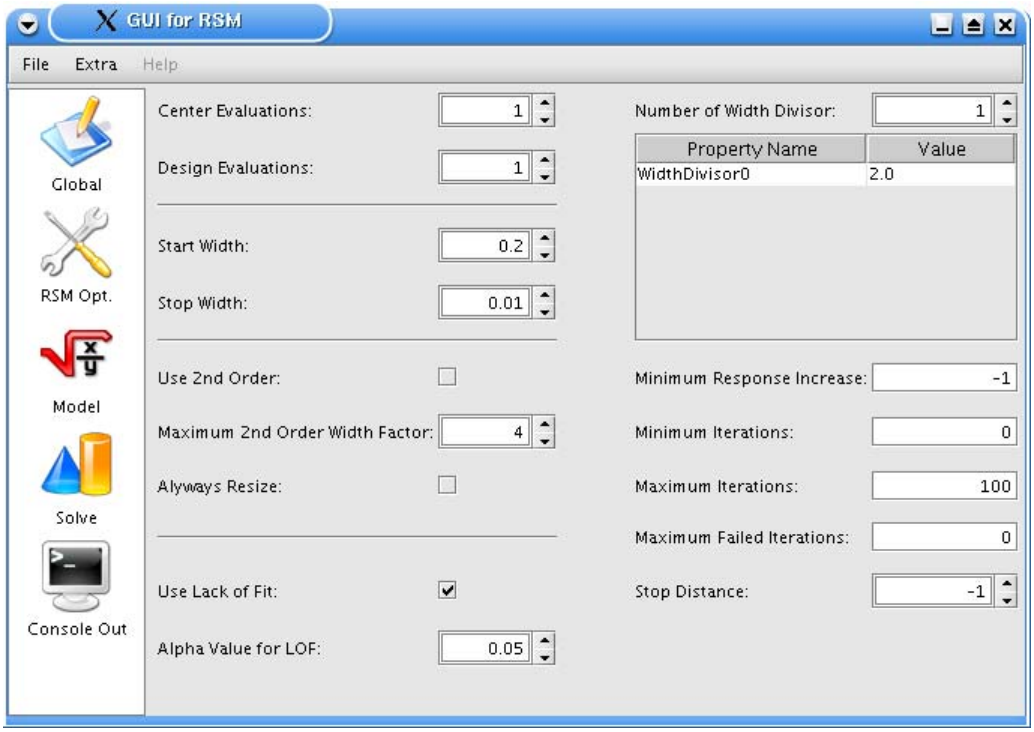


Abbildung 31: RSM Options Panel

Die nächsten beiden Optionen stehen für die halbe Breite des Suchbereichs und ab welcher Größe das RSM aufhören kann mit der weiteren Berechnung. "Start Width" steht für die Startbreite, welche immer weiter verringert wird, wenn in diesem Bereich kein neuer Mittelpunkt gefunden wird. Die Breite wird soweit minimiert, bis der Wert "Stop Width" erreicht ist, dann bricht das RSM ab. Mit dem Schalter "Use 2nd Order" wird bestimmt ob das Modell zweiter Ordnung verwendet werde soll. Ist dies der Fall, wird es ab einem Faktor „Maximum 2nd Order Width Factor“ bzgl. der „Stop Distance“ genutzt. Die Checkbox "Always Resize" bestimmt ob der Suchbereich bei jedem Schritt verkleinert werden soll oder nur wenn das RSM keinen besseren Centerpoint mehr finden kann. Der Schalter "Use Lack of Fit" aktiviert den Lack of Fit Test. Ist diese Option aktiviert wird der berechnete Wert mit dem Erwartungswert verglichen, ist die statistische Differenz höher als unter "Alpha Value for LOF" festgelegt, dann entspricht das Model nicht den Erwartungen und wird neu berechnet. Die Tabelle "Width Divisor" gibt an, durch welchen Wert die Breite des Design

Models geteilt werden soll, wenn sie verkleinert wird. Hier können mehrere Werte eingestellt werden, die der Reihe nach abgearbeitet werden. Ist kein weiterer Wert verfügbar wird der letzte Wert der Tabelle benutzt. Über die beiden Buttons am Kopf der Tabelle kann die Anzahl der Teiler erhöht oder verringert werden. Die Option „Minimum Response Increase“ ist ein Abbruchkriterium. Wenn ein neuer Punkt gefunden wurde, und die Differenz des neuen Response und des vorherigen nicht größer ist als der Wert des „Minimum Response Increase, bricht RSM die Suche ab. Der Wert "Minimum Iterations" bezeichnet die mindest Anzahl an Optimierungsschritten bevor das RSM abbricht. Dem gegenüber wird mit "Maximum Iterations" die maximale Anzahl an Optimierungsschritten eingestellt, damit RSM nicht unnötig viel Optimierungsschritte durchführt, wenn es sich in einer Schleife befindet. Des weiteren steht der Wert "Max. Failed Iterations" für die Anzahl an fehlgeschlagenen Iterationen die der Algorithmus machen kann bevor er die Berechnung abbricht. Mit der Option "Stop Distance" wird der Wert eingestellt den ein neuer Centerpunkt von einem alten entfernt sein muss, damit mit diesem weiter gearbeitet wird. Ist der Punkt nicht weit genug entfernt, bricht das RSM ab.

5.3.3 Model Panel

Auf dem dritten Panel, welches über den Button "Model" der Navigationsleiste zu erreichen ist, werden modellspezifische Optionen angezeigt (siehe Abbildung 32). Die erste Option ist "Model". In diesem Auswahlfeld werden alle verfügbaren Modelle angezeigt und ausgewählt.

Neben einigen Standardmodellen, wie Sinus- oder Sphere-Modell, existieren weitere externe Modelle, welche zur Simulation von eigenen Modellen verwendet werden. Hierzu zählen Modelle wie DSPN oder HIT, diese müssen allerdings überweitere Konfigurationsdateien bearbeitet werden. Unter der Auswahl des Modells befindet sich ein Zahlenfeld mit der Option "Model Replications". Diese Option wird verwendet, um die Anzahl der Auswertungen in einem Punkt einzustellen. Aus den einzelnen Ergebnissen wird dann ein Durchschnittswert berechnet und dieser als Ergebnis zurückgegeben.

Die weiteren Optionen betreffen den Fehler, welcher bei jeder Berechnung mit einfließen kann. Mittels des Schalters "Sigma enable" wird der Fehler ein- bzw. ausgeschaltet. Ist diese Option aktiviert, ist über das Feld "Sigma" die Höhe des Fehlers einzustellen. Des weiteren kann über die Option "Seed" der Initialisierungspunkt des Zufallsgenerators bearbeitet werden. Je nach Modell werden unterhalb der gerade beschriebenen Optionen noch weitere angezeigt. Standardmäßig gibt es jedoch immer folgende Optionen.

Über die Einträge in der Tabelle "Optimal Value for Factors" werden die optimalen Wert für die einzelnen Faktoren eingestellt, diese dienen zur Überprüfung der Arbeitsweise von RSM auf Funktionen mit bekannten Optima. Des weiteren wird mit "Correct Model" ein Modell angeben, welches als Verifikation der berechneten Werte dienen soll. Über die Option "Max. Response" wird der maximale Response des "Correct Model" eingestellt.

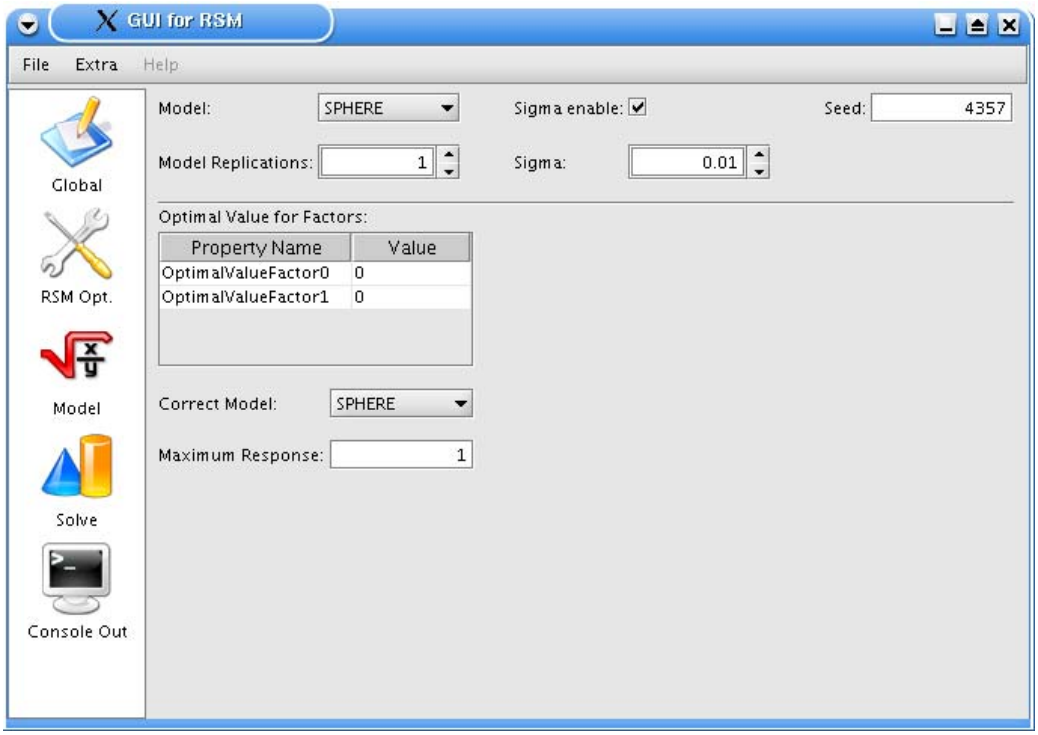


Abbildung 32: Model Panel

Neben den Standardoptionen, besitzt das Tandem Model die Option „Stop Time“ über diese wird die Simulationszeit eingestellt. Neben dem Tandem Model besitzen noch das APNN, DSPN, RSM und das External Model eine Sondereinstellung. Durch diese Option wird der Wert festgelegt, der verwendet werden soll, wenn die externen Modelle keinen Wert berechnen konnten.

5.3.4 Solve Panel

Hinter dem Panel "Solve" verbergen sich die verschiedenen Lösungsmöglichkeiten des RSM (siehe Abbildung 33). Während der verschiedenen Programmläufe wird ständig die aktuelle Ausgabe der Auswertungsalgorithmen auf dem fünften Panel, welches sich hinter dem Button "Console Out" verbirgt, angezeigt. Die Programmläufe können über "Cancel" Buttons gestoppt werden.

Durch klicken auf "RSM Run" wird der normale Lauf des RSM gestartet. Wenn dieser beendet ist, werden auf dem RSM Output Fenster die Ergebnisse des Optimierungslaufs grafisch aufbereitet angezeigt (siehe Abbildung 36). Die genauen Werte und Ergebnisse können auf der fünften Ansicht, dem Console Out Panel betrachtet werden (siehe Abbildung 34). Die graphische Aufarbeitung wird nur für zwei Dimensionen unterstützt. Bei mehr oder weniger Dimensionen wird ein Hinweistext angezeigt. Über die Tabelle "Eval. Single Point" ist es möglich einen einzelnen Punkt einzustellen, welcher mit dem Button "Single Eval." ausgewertet wird. Hierbei wird die Dimension über die Anzahl der Faktoren auf dem Global Panel eingestellt (siehe Abbildung 30). Das Resultat dieser Berechnung wird auf dem Console Out Panel, angezeigt. Über den Button "RSM Multiple" werden mehrere RSM Durchläufe

gestartet, die Anzahl von Läufen ist gleich dem Wert "Num. of Repetions". Durch der mehrfachen Lauf des RSM wird eine höhere Genauigkeit erreicht. Der letzte Knopf dieser Ansicht startet die Erstellung der Response Surface. Die Surface wird auf der Basis des aktivierten Modells berechnet. Die Größe und Auflösung lassen sich über die Textfelder im oberen Bereich einstellen. Wurde das Response Surface erstellt, wir diese auf dem Response Surface Fenster angezeigt.

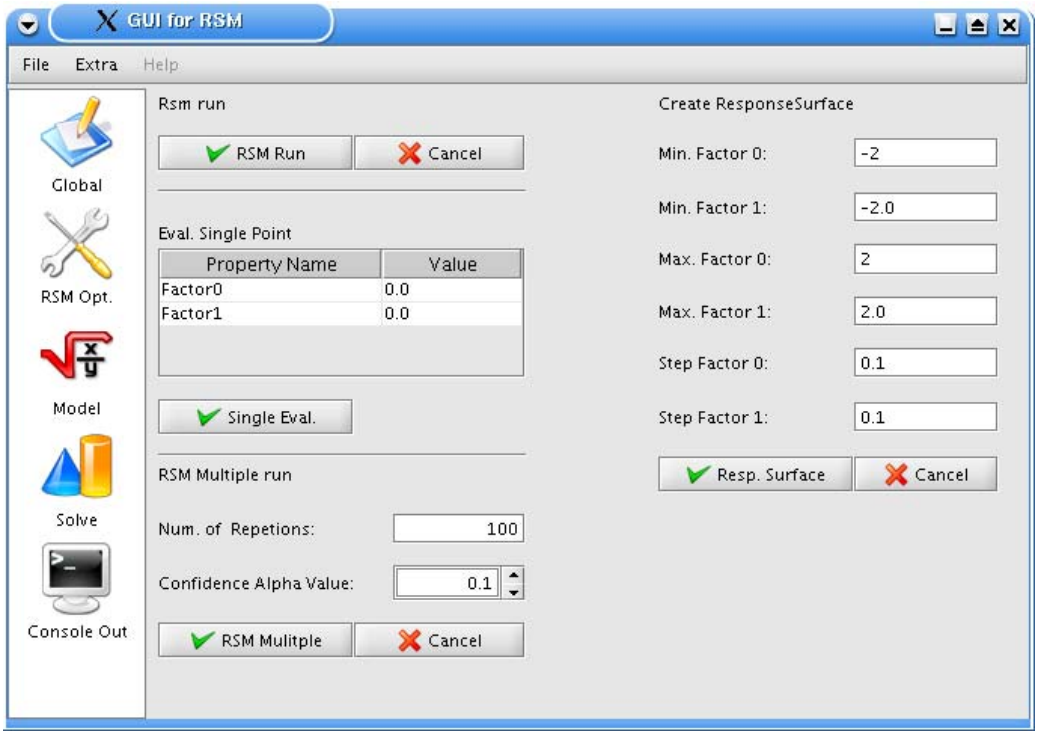


Abbildung 33: Solve Panel

5.3.5 Das Console Out Panel

Auf diesem Fenster wird die Ausgabe des RSM angezeigt. Über den Knopf "Save Output" ist es möglich den Inhalt des Fenster in eine Textdatei zu speichern. So ist es, z.B. im Falle eines RSM Laufs diesen sich später noch einmal anzusehen (siehe Unterkapitel 5.2). Auf der hier dargestellten Abbildung ist die Ausgabe eines RSM Laufs zu sehen. So ist auf dieser Ausgabe zu sehen, dass RSM als letztes das Modell zweiter Ordnung benutzt hat. Dies ist an den Zeilen mit der Bezeichnung „TDesign: DesignPoint“ zuerkennen, da bei diesen die Nummerierung von eins bis acht durchläuft. Im Gegensatz zum vorletzten Lauf, bei diesem wurde das Modell erster Ordnung verwendet. In diesen Zeilen sind auch die genauen Koordinaten der Designpunkte zu erkennen, diese werden in Klammern hinter der Bezeichnung der Punkte angegeben. Die Ausgaben „Currently best Point“ bzw. „Currently best Response“ geben die Momentanen besten Ergebnisse aus. Am Ende werden alle wichtigen Ausgaben noch einmal zusammen gefasst.

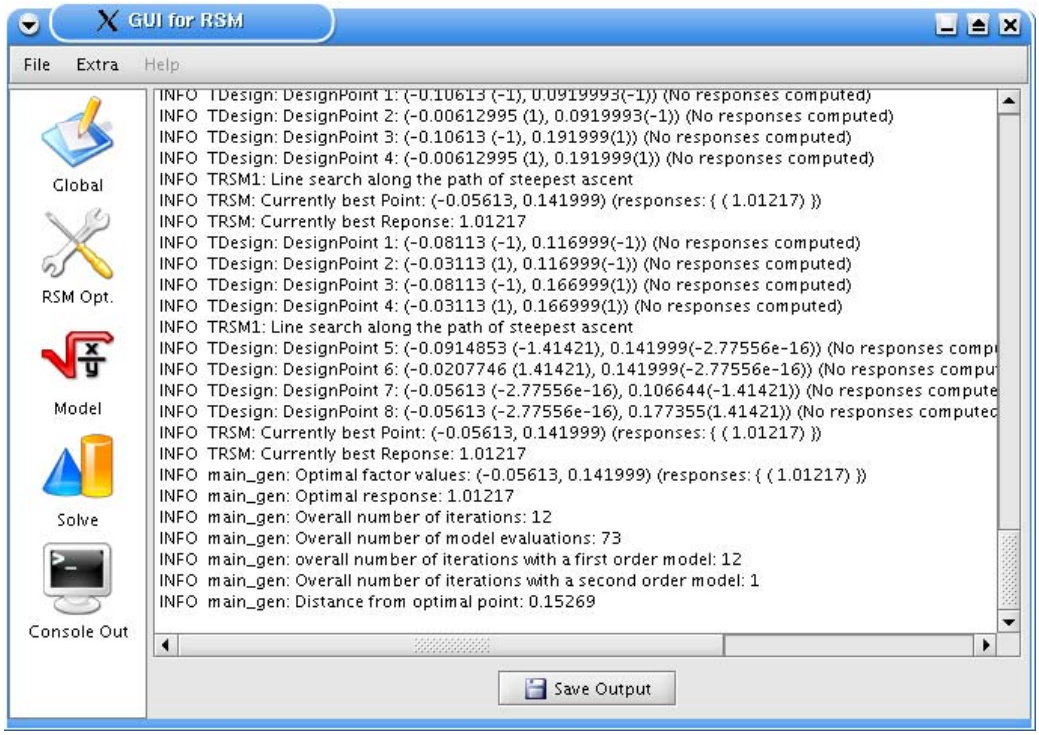


Abbildung 34: Console Out Panel

5.3.6 Das Response Surface Fenster

In diesem Fenster wird das vom RSM erstellte Response Surface angezeigt, es ähnelt Abbildung 35. Am unteren Rand befindet sich der "Close" Button über den sich das Fenster schließen lässt. In der hier gezeigten Abbildung ist das Response Surface der Sphere Funktion abgebildet.

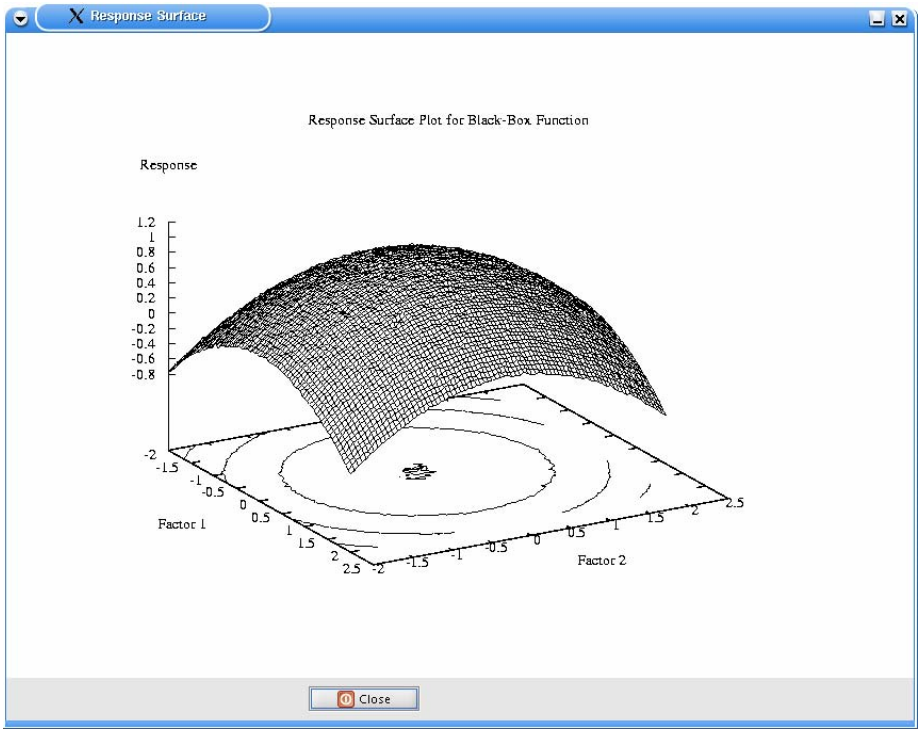


Abbildung 35: Response Surface Fenster

5.3.7 Das RSM Output Fenster

Dieses Fenster wird angezeigt, wenn ein RSM Lauf beendet worden ist (siehe Abbildung 36). Wenn es sich bei diesem Lauf um eine Berechnung mit mehr oder weniger als zwei Dimensionen handelt, wird auf diesem Fenster lediglich der Hinweis angezeigt, dass dieser Durchlauf mehr als zwei Dimensionen hatte, da die grafische Aufarbeitung mit mehr als zwei Dimensionen nicht unterstützt wird.

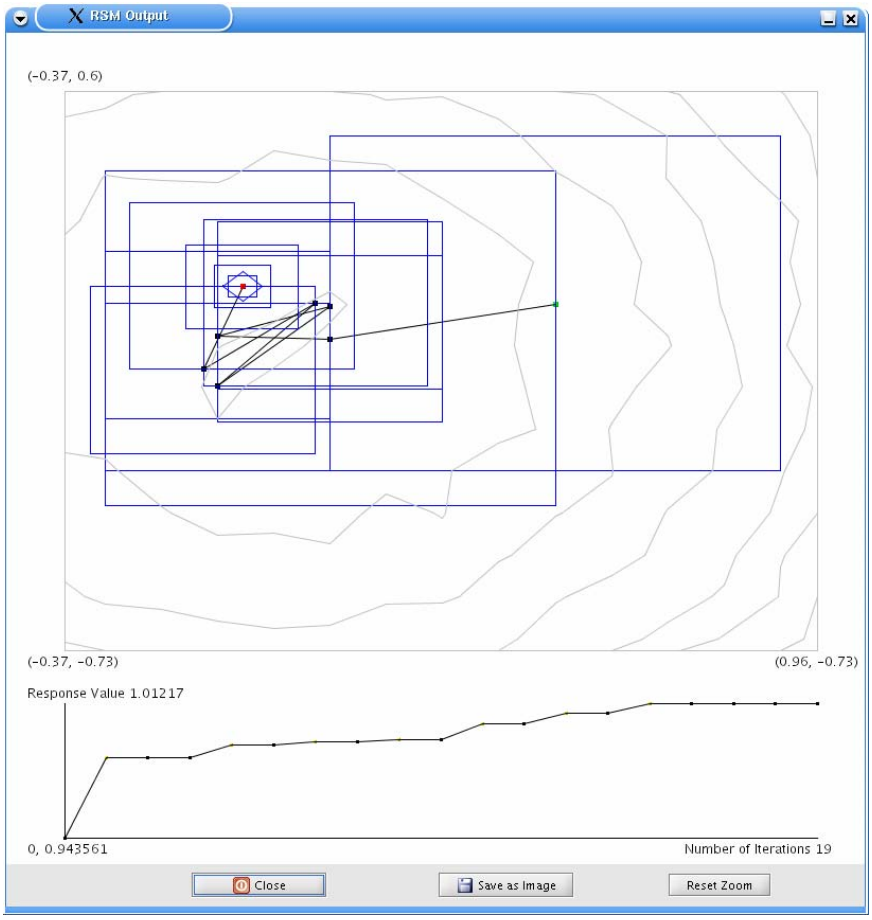


Abbildung 36: RSM Ouput Fenster

Handelt es sich um einen Lauf mit zwei Dimensionen, wird im oberen Bereich des Fensters ein Koordinatensystem mit dem Wertebereich des RSM Laufs angezeigt. Der grüne Punkt bezeichnet den Startpunkt, gelbe Punkte sind Punkte die auf dem „steepest Ascent Path“ liegen, schwarz werden neue Centerpoints abgebildet und der rote Punkt markiert die Stelle an der RSM den besten Punkt gefunden hat. Je nach Einstellung im Options Dialog werden die Centerpoints mit Linien verbunden (siehe Abbildung 37). Die blauen Punkte, bzw. je nach Einstellung Linien, sind die Eckpunkte, bzw. die Kanten des Designmodells. Wenn die entsprechende Option aktiviert ist werden auch die Konturlinie des Response Surfaces angezeigt. Durch markieren eines Bereichs mit der Maus ist es möglich in das Bild hineinzuzoomen. Der Zoom kann wieder auf den Ausgangswert gestellt werden durch drücken des Buttons "Reset Zoom". Im unteren Teil dieses Fensters ist die Entwicklung der

Responses zu sehen. An dieser Kurve ist abzulesen wie schnell RSM sich dem berechneten Optimum nähert. Über den "Close" Button wird das Fenster geschlossen. Mit dem "Save as Image" Knopf wird die grafische Ausgabe als Bild gespeichert. Das Bild wird mit der im Optionen Dialog spezifizierten Auflösung gespeichert.

In dem hier abgebildeten Lauf sind die Sprünge des RSM zuerkennen, die es zurücklegt, bei der Suche nach einer Optimalen Lösung, auch ist um den roten Endpunkt das Modell zweiter Ordnung zuerkennen. Im unteren Bereich der Abbildung ist am Verlauf des Anstiegs der Responsewerte zuerkennen, dass die letzten Punkte sich kaum noch von einander unterscheiden.

5.3.8 Der Options Dialog

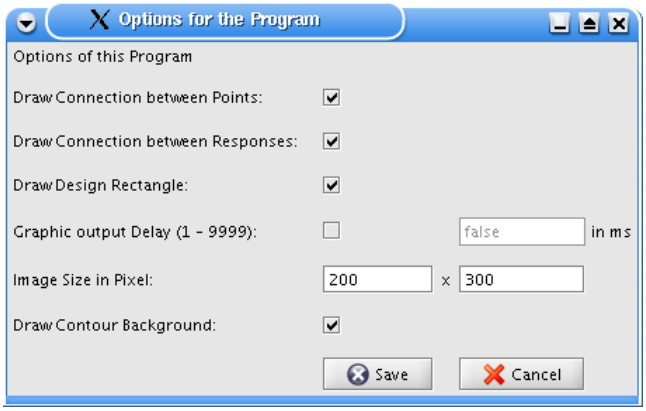


Abbildung 37: Options Dialog

In diesem Dialog werden die grundlegenden Einstellungen der Oberfläche angezeigt. Mit der ersten Option "Draw Connection between Points" kann eingestellt werden, ob eine Verbindungslinie zwischen den Centerpoints auf dem RSM Output Fenster gezeichnet werden soll (siehe Abbildung 36). Die nächste Option steht für die Verbindungslinien zwischen den Responsepoints. Über die Option "Draw Design Rectangle" wird die Art der Darstellung des Designmodells festgelegt. Ist diese Option aktiviert wird es als Rechteck gezeichnet, ansonsten werden nur die einzelnen Eckpunkte gezeichnet. "Graphic output Delay" steuert die Verzögerung beim Zeichnen der Punkte auf dem RSM Output Fenster, so ist es möglich zu sehen wie der Algorithmus die Punkte berechnet. Mit "Image Size in Pixel" wird festgelegt mit welcher Auflösung die Ausgabe des RSM Output Fensters bei der Benutzung des "Save as Image" Button gespeichert werden soll. Des weitern ist es mit der Option "Draw Contour Background" möglich das Zeichnen der Konturlinien auf dem RSM-Output Fenster zu aktivieren.

5.4 Schließen des Programms

Wenn das Programm geschlossen wird erscheint eine Sicherheitsabfrage, die den User fragt ob das Programm wirklich geschlossen werden soll. Es stehen drei Optionen zur Auswahl.

Mit der Standardauswahl wird das Programm geschlossen, wobei vorher alle Einstellungen gespeichert werden, hierzu dient der Button „Save and Exit“. Des weiteren ist es möglich das Programm zu schließen ohne die gewählten Einstellungen zu speichern, dies ist über den Button „Exit“ möglich. Als letzte Möglichkeit gibt es den „Cancel“ Button dieser veranlasst das Programm sich nicht zu schließen und zur normalen Ansicht zurück zukehren.

6 Zusammenfassung

RSM ist ein Optimierungsverfahren, das auf der Basis von Modellen erster und zweiter Ordnung Faktorbelegungen findet, die in der Nähe lokaler Optima liegen. Dabei wird eine Reihe mathematischer und statistischer Verfahren genutzt, um bessere Faktorbelegungen zu ermitteln. Darüber hinaus können Modelle bewertet werden, um deren Übereinstimmung mit dem realen Verlauf zu beurteilen.

Die automatisierte Implementierung bietet den Einsatz von Modellen erster und zweiter Ordnung, die Abschätzung der Güte eines Modells mittels CoD und LoF sowie eine Reihe von Kriterien, die über den Verlauf der Optimierung entscheiden. Es können Modelle mittels der Programme NSolve, DSPNexpressNG und HIT an die Implementierung angebunden werden und für die Konfiguration und Ausführung existiert eine übersichtliche GUI, die darüber hinaus die Ergebnisse grafisch aufbereitet.

Die durchgeführten Experimente zeigen, dass die aktuelle Implementierung in der Lage ist lokale Optima zu finden und auch bei intensiver Störung der Responses gute Ergebnisse liefert, wobei die Güte des Ergebnisses mit der Anzahl der Auswertungen zunimmt. Darüber hinaus ist die Anzahl benötigter Auswertungen für eine Konfiguration weitestgehend konstant, so dass der Aufwand für eine Optimierung im Vorhinein abgeschätzt werden kann.

Da RSM eine Reihe bisher nicht implementierter Methoden und Eigenschaften enthält, können weitere Verbesserungen an der aktuellen Implementierung vorgenommen werden. Diese umfassen vor allem die Unterstützung diskreter Parameter und ein verbesserter Einsatz der unterschiedlichen Experimentdesigns. Darüber hinaus ist es auch sinnvoll, die Eigenschaft von RSM, lokale Optima zu finden, zu optimieren. Hierzu bieten sich Verfahren an, bei denen RSM mit Multistarts genutzt wird. Neben einer einfachen zufallsbasierten Auswahl der Startpunkte scheint hier vor allem eine Kopplung von RSM mit evolutionären Algorithmen viel versprechend zu sein.

Anhang

6.1 3D Plots der Testfunktionen

Die folgenden Abbildungen enthalten je implementierter Testfunktion ein Paar 3D Plots. Der linke Plot entspricht dabei der jeweils angegebenen Funktion *f(x)* (siehe Tabelle 2). Der rechte Plot entspricht der Funktion $g(x) = f(x) + N(0, \sigma)$ mit einer normalverteilten Störung $N(0, \sigma)$ mit Mittel *0* und $\sigma = 0{,}1$ und einer mittleren Abweichung von $\sigma \cdot \sqrt{2/\pi}$. Diese Werte entsprechen den Grenzwerten, der im Kapitel 3 eingesetzten Störungen. Für jeden Plot wurden 961 Auswertungen benötigt, wobei anzumerken ist, dass ein RSM-Durchlauf bei zwei Faktoren deutlich unter 100 Auswertungen benötigt (siehe Abbildung 8 und Abbildung 9).

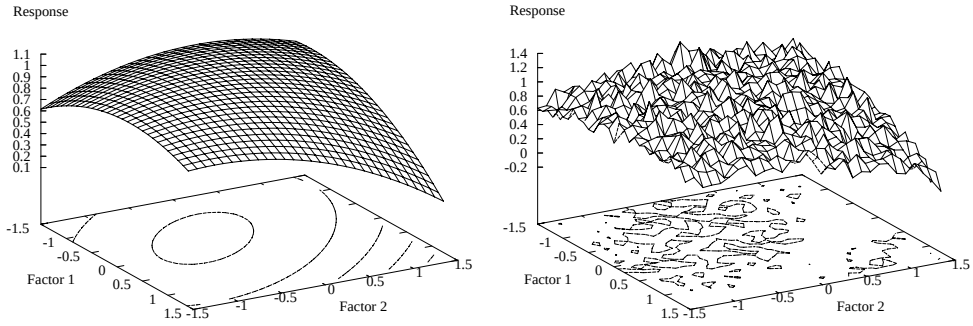


Abbildung 38: Polynomfunktion

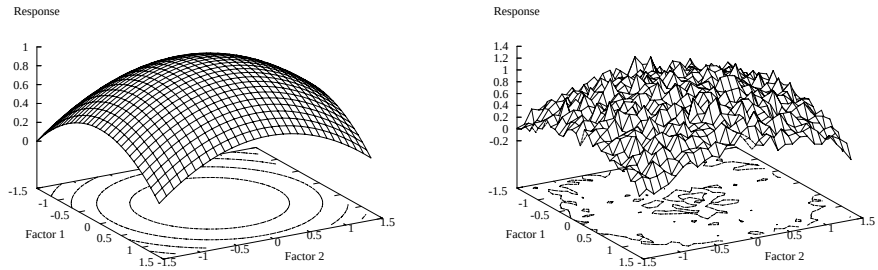


Abbildung 39: Sphere

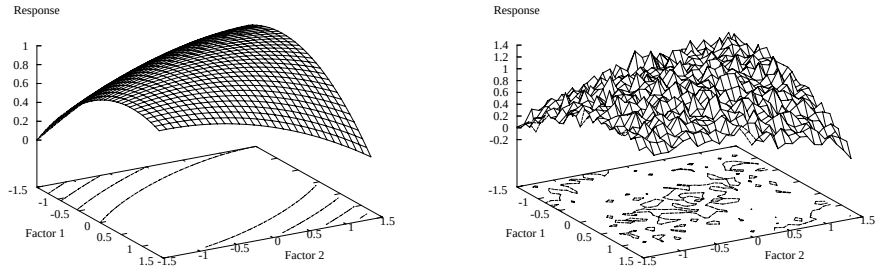


Abbildung 40: Schwefel

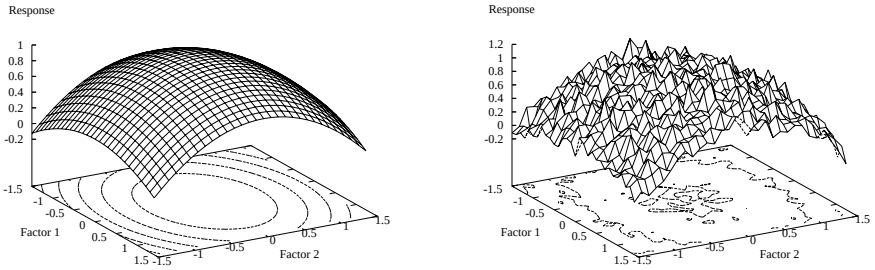


Abbildung 41: Ellipsoid

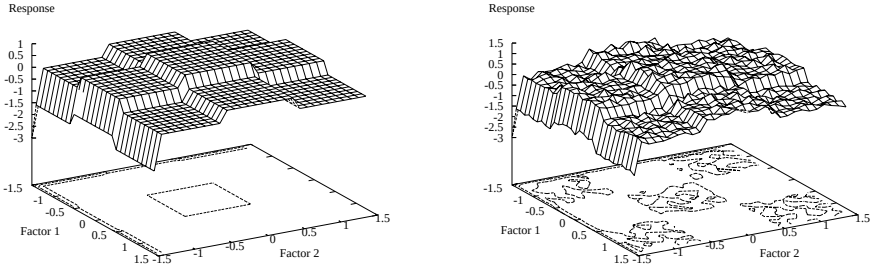


Abbildung 42: Step

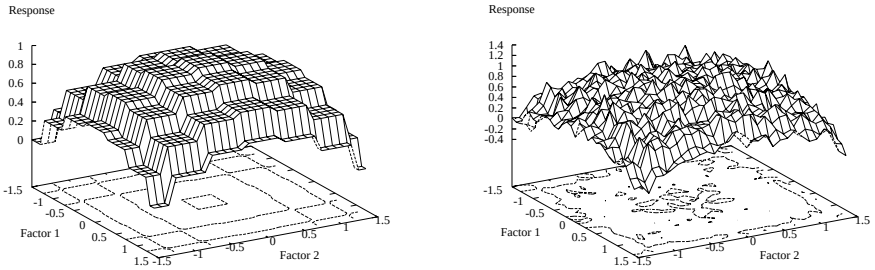


Abbildung 43: Step 2

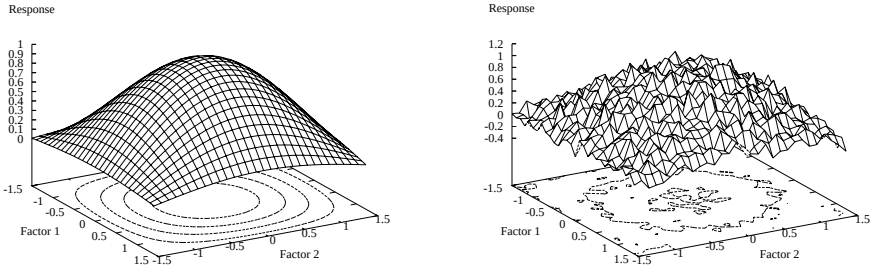


Abbildung 44: Sinus

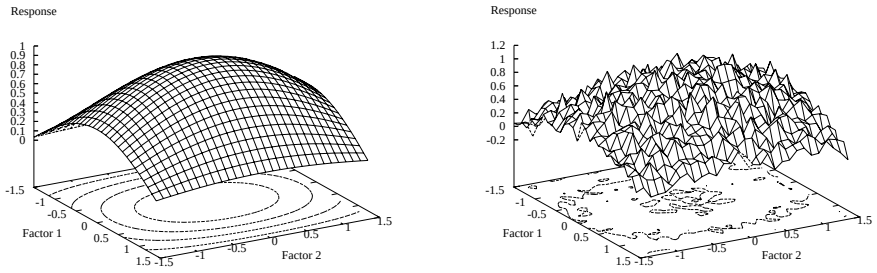


Abbildung 45: Griewank

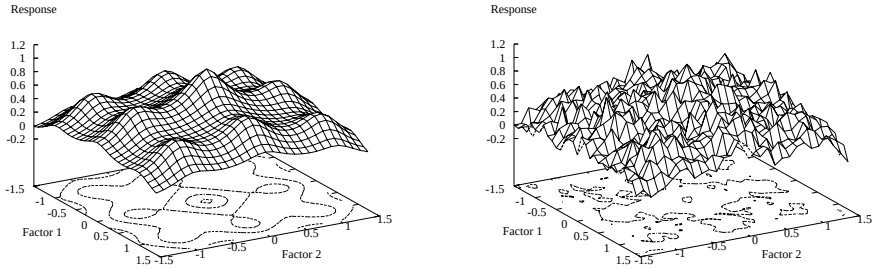


Abbildung 46: Ackley

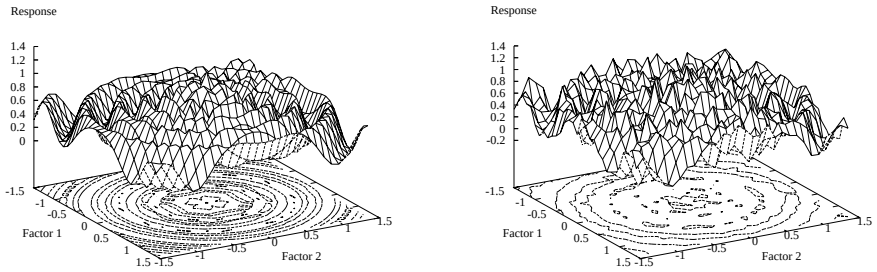


Abbildung 47: Schaffer

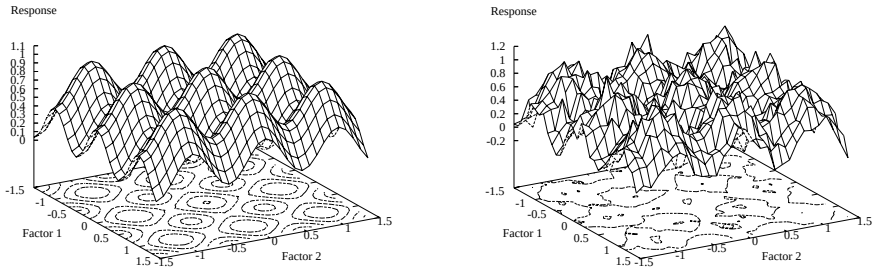


Abbildung 48: Rastrigin

6.2 Konfigurationsdatei

Die Konfigurationsdatei entspricht einem häufig eingesetzten Format, bei dem Kommentare durch „#“ gekennzeichnet werden und Parameter zeilenweise in der Form *name = wert* eingetragen werden, wobei führende und folgende Leerzeichen ignoriert werden. Es werden Zahlen, boolesche Werte und Zeichenketten unterstützt. Der Typ eines Parameters wird dabei nicht explizit angegeben. Werte, die nicht dem erwarteten Typ entsprechen, werden bei Zahlen *0* und bei booleschen Werten als *falsch* interpretiert. Für Zahlen werden verschiedene Schreibweisen wie 1.0 oder 1e-10 akzeptiert, bei booleschen Werten gelten *true* und *yes* als *wahr*.

```
###############################################################################
# SimOpt - configuration file
###############################################################################

###############################################################################
# 1. General settings, specifying the optimization problem
###############################################################################

# Definition of number of continuous and discrete input parameters (factors)

NumberContFactors = 2
NumberDiscFactors = 0

# The bounds must be definined for every input parameter, where the first
# bounds correspond to the continuous parameters followed by the bounds for
# the discrete parameters

LowerBoundFactor0 = -1.5
UpperBoundFactor0 = 1.5

LowerBoundFactor1 = -1.5
UpperBoundFactor1 = 1.5

# Definition of starting point.
# If RandomStart = yes, the starting point is a random point inbetween the
# predefined bounds. If RandomStart = no, the starting point is taken from
# the definition StartFactor0, StartFactor1, ...

RandomStart = no

StartFactor0 = 0.0
StartFactor1 = 0.0

# Definition of fallback value if the black-box function cannot be evaluated.

ResponseFallback = -1


###############################################################################
# 2.1 RSM specific settings
###############################################################################

# Definition of the number of replicated evaluations of points in the
# experimental design. Points on the path of steepest ascent are also evaluated
# as often as specified in CenterEvaluations.

CenterEvaluations = 3
DesignEvaluations = 1

# Definition of start- and stopwidth. Startwidth is the half-width of the local
# region with center point defined by StartFactor0, StartFactor1, etc. in coded
```

```
# variables, i.e., after transformation of the natural search space into a
# [-1,1]^k hypercube. StopWidth <= 0 disables this stop criterium.

StartWidth = 0.4
StopWidth = 0.01

# This criterium defines the minimum amount a new center point has to be
# differ from the current center point to continue.
# StopDistance <= 0 means disabled

StopDistance = -1

# This criterium defines the minimum amount the response of a new center point
# has to be differ from the response of the current center point to continue.
# StopDistance <= 0 means disabled

MinimumResponseIncrease = -1

# Toggles the use of the lack of fit test.
# If it is enabled but the difference between the response values is to low, it
# will be automatically disabled.
# If no replications are made coefficient of determination is used instead.

UseLackOfFit = yes

# Alpha-Value for the lack of fit test

Alpha = 0.05

# If lack of is enabled but no replications are made the test is substituted by
# COD<MinimumCoefficientOfDetermination

MinimumCoefficientOfDetermination = 0.8

# Theses values define the divisors in the main rsm-routine:
# step = 0
# while not stop {
#   rsm1;
#   if (!successfull) {
#     if not adequate {
#       rsm2;
#     }
#     width/WidthDivisor[step];
#   }
#   step++;
# }
#
# WidthDivisor[0] is the value FirstOrderModelDivisor0
# if FirstOrderModelDivisor(step) or SecondOrderModelDivisor(step) does not
# exist, the highest existing step-value is used. At least the last value has
# to be higher than 1.

WidthDivisor0 = 2.0

# Resize even if first order model was successfull

AlwaysResize = no

# If the design of the second order model does not provide enough information
# to calculate the coefficients how often may the design be extended to
# solve this problem.
# ATTENTION: Every extension double the number of points in the design.

MaximumFractionalFactorialDesignExtensions = 0

# This value defines when the 2nd-order-model is first used. The width has to be
# lower than StartWidth/Maximum2ndOrderWidthFactor.

UseSecondOrderModel = 1
```

```
Maximum2ndOrderWidthFactor = 4


# This criterium defines the minimum/maximum number of iterations during one
# RSM call. 0 means no limit.

MinimumIterations = 0
MaximumIterations = 100
MaximumFailedIterations = 0

# Two values (a and b) are considered to be equal if |a - b| < Epsilon

Epsilon = 1e-20

# Definition of the designtype used for the first order model. Note that the
# design with 1 or 2 dimensions will always be a full factorial design 2^k. The
# one half fractional factorial design is a 2^k-1 fractional factorial design.
#
# Available Designs: FactorialDesign, FractionalFactorialDesign

DesignType = FactorialDesign


################################################################################
# 2.2 Settings for RSM statistical evaluations
################################################################################

# Number of repetitions of the complete RSM algorithm

Repetitions = 100

# Confidence Interval

ConfidenceAlpha = 0.1

# Correct Model

CorrectModel = SPHERE
MaximumResponse = 1


################################################################################
# 2.3 Settings for complete response surface generation
################################################################################

ResponseSurfaceMin0 =  0.5
ResponseSurfaceMin1 =  0.05
ResponseSurfaceMax0 =   0.9
ResponseSurfaceMax1 =   0.95
ResponseSurfaceStepsize0 = 0.05
ResponseSurfaceStepsize1 = 0.05

ResponseFilename = curves/ResponseSurface.dat

################################################################################
# 3. Specific settings for included test models
################################################################################


# The model which is used
# Available are: SINUS, POLY, EXP_POLY, DSPN, HIT, PLANE, RSM, SPHERE, STEP,
# ACKLEY, ROSENBROCK, ELLIPSOID, SCHWEFEL, RASTRIGIN, GRIEWANK, SCHAFFER,
# TANDEM, EXTERNAL, APNN

Model = SPHERE

# Number of replicated evaluation of the model

ModelReplications = 1
```

```
################################################################################
# 3.1 Settings for stochastic functions
################################################################################

# Options for normally distributed error with mean 0 and variation Sigma.
# Sigma = 0 means that no random error is added to the response value.

Sigma = 1e-2
Seed = 4357


################################################################################
# 3.2 Settings for the tandem queue model
################################################################################

# Simulation until time StopTime

StopTime = 1000


################################################################################
# 3.3 Settings for functions where the optimal point may change
################################################################################

# Currently these variables are used by TRSMModel, TDSPNModel and THITModel.
# These values define the optimal values for each factor

OptimalValueFactor0 = 0
OptimalValueFactor1 = 0
OptimalValueFactor2 = 0
OptimalValueFactor3 = 0
OptimalValueFactor4 = 0
OptimalValueFactor5 = 0
OptimalValueFactor6 = 0
OptimalValueFactor7 = 0
OptimalValueFactor8 = 0
OptimalValueFactor9 = 0
OptimalValueFactor10 = 0


################################################################################
# 3.4 Settings defining parameters for fractional factorial designs
################################################################################

# Generators for fractional factorial designs
# Explanation:
# FractionalFactorial_<k>-<p>_Generator_<factor> = <generator>
# This configurationstring holds the generator for the factor with number
# <factor> (counting starts with 0) for a k-p fractional factorial design
# The argument <generator> has the following meaning:
# If bit x is set the current factor depends on factor x.
#
# Example:
#
# 3-1 fractional factorial design:
# FractionalFactorialDesign_3-1_Generator_2 = 3
#
# Number Factor 1 2 3
#   0           0 0 0
#   1           1 0 1
#   2           0 1 1
#   3           1 1 0
#
# 0s are interpreted as - and 1s as +.

FractionalFactorialDesign_4-1_Generator_3 = 3
FractionalFactorialDesign_5-1_Generator_4 = 7
```

```
FractionalFactorialDesign_6-2_Generator_4 = 5
FractionalFactorialDesign_6-2_Generator_5 = 6
FractionalFactorialDesign_7-2_Generator_5 = 6
FractionalFactorialDesign_7-2_Generator_6 = 7
FractionalFactorialDesign_8-3_Generator_5 = 6
FractionalFactorialDesign_8-3_Generator_6 = 7
FractionalFactorialDesign_8-3_Generator_7 = 3
FractionalFactorialDesign_9-3_Generator_6 = 7
FractionalFactorialDesign_9-3_Generator_7 = 11
FractionalFactorialDesign_9-3_Generator_8 = 13
FractionalFactorialDesign_10-4_Generator_6 = 7
FractionalFactorialDesign_10-4_Generator_7 = 11
FractionalFactorialDesign_10-4_Generator_8 = 13
FractionalFactorialDesign_10-4_Generator_9 = 14
FractionalFactorialDesign_11-5_Generator_6 = 6
FractionalFactorialDesign_11-5_Generator_7 = 7
FractionalFactorialDesign_11-5_Generator_8 = 9
FractionalFactorialDesign_11-5_Generator_9 = 10
FractionalFactorialDesign_11-5_Generator_10 = 11

FractionalFactorialDesign_2-1_Generator_1 = 0
FractionalFactorialDesign_3-1_Generator_2 = 3
FractionalFactorialDesign_4-2_Generator_2 = 4
FractionalFactorialDesign_4-2_Generator_3 = 3
FractionalFactorialDesign_5-2_Generator_3 = 3
FractionalFactorialDesign_5-2_Generator_4 = 7
FractionalFactorialDesign_6-3_Generator_3 = 3
FractionalFactorialDesign_6-3_Generator_4 = 5
FractionalFactorialDesign_6-3_Generator_5 = 6
FractionalFactorialDesign_7-4_Generator_3 = 3
FractionalFactorialDesign_7-4_Generator_4 = 5
FractionalFactorialDesign_7-4_Generator_5 = 6
FractionalFactorialDesign_7-4_Generator_6 = 7
FractionalFactorialDesign_8-5_Generator_3 = 4
FractionalFactorialDesign_8-5_Generator_4 = 5
FractionalFactorialDesign_8-5_Generator_5 = 6
FractionalFactorialDesign_8-5_Generator_6 = 7
FractionalFactorialDesign_8-5_Generator_7 = 3
FractionalFactorialDesign_9-5_Generator_4 = 3
FractionalFactorialDesign_9-5_Generator_5 = 5
FractionalFactorialDesign_9-5_Generator_6 = 7
FractionalFactorialDesign_9-5_Generator_7 = 11
FractionalFactorialDesign_9-5_Generator_8 = 13
FractionalFactorialDesign_10-6_Generator_4 = 3
FractionalFactorialDesign_10-6_Generator_5 = 5
FractionalFactorialDesign_10-6_Generator_6 = 7
FractionalFactorialDesign_10-6_Generator_7 = 11
FractionalFactorialDesign_10-6_Generator_8 = 13
FractionalFactorialDesign_10-6_Generator_9 = 14
FractionalFactorialDesign_11-7_Generator_4 = 3
FractionalFactorialDesign_11-7_Generator_5 = 5
FractionalFactorialDesign_11-7_Generator_6 = 6
FractionalFactorialDesign_11-7_Generator_7 = 7
FractionalFactorialDesign_11-7_Generator_8 = 9
FractionalFactorialDesign_11-7_Generator_9 = 10
FractionalFactorialDesign_11-7_Generator_10 = 11
```

6.3 Klassen

Dieses Kapitel stellt eine Übersicht über allgemeine Klassen dar, die bei der Entwicklung entstanden sind, jedoch auch für andere Projekte sinnvoll sind. Zu diesen gehört das Paar `TBlackBoxModel` und `TModelFactory`, die es ermöglichen unterschiedliche Funktionen und Programme als Response Generatoren zu nutzen. Die Implementierung ist so gehalten, dass die Konfiguration der Modelle leicht in eine Konfigurationsdatei ausgelagert

werden kann. Des weiteren sind die Klassen `TDesign` und `TDesignPoint` zu nennen, die zur Erstellung von und Arbeit mit verschiedenen Designs dienen (siehe Tabelle 3). Weitere Klassen, die einfache Funktionen zur Arbeit mit C++ zur Verfügung stellen, sind in Tabelle 4 aufgeführt.

Name	Beschreibung
`TBlackBoxModel`	Oberklasse für Funktionen und Modelle
`TModelFactory`	Klasse zur korrekten Erzeugung von TBlackBoxModel-Instanzen
`TDesignPoint`	Klasse zur Speicherung von Responses und Koordinaten inklusive Konvertierung von natürlichen in kodierte Wertebereiche
`TDesign`	Klasse zur Erzeugung und Arbeit mit Experminentdesigns

Tabelle 3: Zentrale Klassen

Name	Beschreibung
`MathUtils`	Einfache Schnittstelle zu Funktionen aus der GSL
`Solve`	Gaus-Jordan Eliminationsverfahren
`StringUtils`	Funktionen zum Umgang mit Zeichenketten
`TConfiguration`	Auslesen von Einstellungen aus Konfigurationsdateien
`TPropertyFile`	Auslesen und Schreiben von Einstellungen in Konfigurationsdateien
`Execute`	Ausführen von externen Programmen unter verschiedenen Betriebssystemen

Tabelle 4: Hilfsklassen

6.4 Installation

6.4.1 Vorraussetzungen

Um RSM nutzen zu können sind mehrere Programm und Bibliotheken nötig. Da die RSM-Distribution im Quellcode vorliegt, wird der C-Compiler „gcc“ verwendet um das RSM auf dem Zielcomputer zu kompilieren [BBE05]. Dieser ist mindestens in der Version 3.3 zu verwenden. Des weiteren wird die „GNU Scientific Libary“, kurz „gsl“, benötigt, diese ist eine freie Implementierung von numerischen Funktionen und sollte in der Version 1.5 vorliegen [GDT05]. Diese wird genutzt da die in ihr enthalten Funktionen mehrfach getestet

und dokumentiert sind. Neben diesen Vorraussetzungen wird für die GUI eine JAVA Umgebung mit der Versionsnummer 1.4 [SUN05], sowie ghostcript-6.53 [Gil05] und gnuplot-4.0.0 [WK05] benötigt. Soll RSM unter Microsoft Windows betrieben werden, ist für dies „Cygwin“ zu installieren und einzurichten. „Cygwin“ ist eine Linux ähnliche Umgebung für Microsoft Windows [FVD05].

6.4.2 Installationvorgang

Zur Installation von RSM und der zugehörigen GUI muss die Datei RSM-X.X.tar.gz entpackt werden, wobei „X.X“ für einen Versionsbezeichnung steht. Hierzu wird unter Linux ein Entpacker wie Ark benutzt, der bei den meisten Distributionen installiert sein sollte. Unter Windows erledigen dies Programme wie WinAce oder Winrar. Sollten diese Programme nicht verfügbar sein oder unter Windows mit der Cygwin Umgebung gearbeitet werde, so kann die Datei auch mit dem Befehl *„tar –xvzf RSM-X.X.tar.gz“* entpackt werden. Das hier entpackte Verzeichnis ist das Wurzelverzeichnis des RSM. Nach dem Entpacken muss das Makefile des jeweiligen Betriebsystems auf die Gegebenheiten des verwendeten Systems angepasst werden. So muss, z.B. unter Linux die Datei „Makefile.Linux“ mit einem Texteditor angepasst werden. Zusetzen sind hier die Pfade zum gcc und zu den verschiedenen Bibliotheken die von RSM benötigt werden. Als dritter Schritt muss das Programm kompiliert werden. Dazu wird der Befehl *„ ./make“* verwendet. Wenn alle Programmabhängigkeiten richtig aufgelöst wurden, sollte der Compiler jetzt das Programm vollständig kompilieren. Nach diesem Schritt ist RSM Einsatzbereit und kann verwendet werden.

Literaturverzeichnis

[BBE05] P. Bothner, J. Buck, D. Edelsohn. gcc homepage, http://gcc.gnu.org, 15.02.2005

[BBK98] F. Bause, P. Buchholz, and P. Kemper, A Toolbox for Functional and Quantitative Analysis of DEDS, *Proc. 10th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation, Palma de Mallorca, Spain, LNCS* **1469**, 356-359, Springer, 1998.

[BBM02] F. Bause, H. Beilner, M. Fischer, P. Kemper, and M. Völker, The ProC/B Toolset for the Modelling and Analysis of Process Chains, *Proc. 12th Int. Conf. Modelling Tools and Techniques for Computer and Communication System Performance Evaluation, London, UK, LNCS* **2324**, 51-70, Springer, 2002.

[BBS03] F. Bause, H. Beilner, and M. Schwenke, *Semantik des ProC/B-Paradigmas*, Technical Report 03001, SFB 559 – Teilprojekt M1, 2003.

[BMF+05] P. Buchholz, M. Fischer, P. Kemper, C. Tepper. APNN-Toolbox, http://www4.cs.uni-dortmund.de/APNN-TOOLBOX, 18.02.2005

[BMT05] P. Buchholz, D. Müller, and A. Thümmler. Optimization of Process Chain Models with Response Surface Methodology and the ProC/B Toolset, in: H.O. Günther, D.C. Mattfeld, L. Suhl (Eds.), Entscheidungsunterstützende Systeme in Supply Chain Management und Logistik, 551-573, Physica-Verlag, 2005

[BMW89] H. Beilner, J. Mäter, and N. Weißenberg, Towards a performance modeling environment: News on HIT, *Proc. Int. Conf. Modelling Tools and Techniques for Computer Performance Evaluation*, 1989.

[BMW94] H. Beilner, J. Mäter, C.Wysocki. The Hierarchical Evaluation Tool HIT in : *Short Papers and Tool Descriptions of the 7th Internaltional Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1994

[Fu02] M.C. Fu. Optimization for Simulation: Theory vs. Practice, *INFORMS Journal on Computing* **14**, 192-215, 2002.

[FVD05] C. Faylor, C. Vinschen, E. Duda. cygwin homepage, http://cygwin.com, 15.02.2005

[GDT05] M. Galassi, J. Davies, J. Theiler. gsl homepage, http://sourceware.redhat.com/gsl, 15.02.2005

[Gil05] R. Giles, ghostscript homepage, http://ghostscript.com, 15.02.2005

[Kle98] J.P.C. Kleijnen. Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models, in: J. Banks (Ed.), *Handbook of Simulation*, 173-223, John Wiley & Sons, 1998.

[KMT05] P. Kemper, D. Müller, A. Thümmler. Combining Response Surface Methodology with Numerical Models for Optimization of Class Based Queueing Systems, *Proc. Int. Conf. on Dependable Systems and Networks (DSN), Yokohama, Japan*, 2005. (to appear)

[KS00] J.P.C. Kleijnen and R.G. Sargent, A methodology for fitting and validating metamodels in simulation, *European Journal of Operational Research* **120**, 14-29, 2000.

[Lin03] C. Lindemann. DSPNexpress: Open Source Toolkit for Performace Evaluation of UML System Specifications, http://www.dspnexpress.de, 18.02.2005, 2003

[LK00] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, 3rd Edition, McGraw-Hill, 2000.

[MM02] D.C. Montgomery and R.H. Myers, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd Edition, John Wiley & Sons, 2002.

[Mon01] D.C. Montgomery, *Design and Analysis of Experiments*, 5th Edition, John Wiley & Sons, 2001.

[NOP+00] H.G. Neddermeijer, G.J. van Oortmarssen, N. Piersma, and R. Dekker, A Framework for Response Surface Methodology for Simulation Optimization, *Proc. Winter Simulation Conference, Orlando, FL, USA*, 129-136, 2000.

[NP98] V. Nissen und J. Propach. On the Robustness of Population-Based Versus Point-Based Optimization in the Presence of Noise, IEEE Transactions on Evolutionary Computation 2, 107-119, 1998

[Sal98] R. Salomon. Evolutionary Algorithms and Gradient Search: Similarities and Differences, IEEE Transactions on Evolutionary Computation 2, 45-55, 1998

[SUN05] SUN Microsystems. JAVA homepage, http://java.sun.com, 15.02.2005

[WK05] T. Williams, C. Kelley. gnuplot homepage, http://www.gnuplot.info, 15.02.2005

Sonderforschungsbereich 559

Bisher erschienene Technical Reports

03019 Anne Schulze im Hove, Frank Stüllenberg, Marcus Völker: Erweiterung des ProC/B-Paradigmas zur Abbildung entscheidungsrelevanter Kosten und zur Kostenverrechnung

03020 Michael Kaczmarek, Marcus Völker: Entwicklung von Simulationsmodellen für die Analyse von Supply Chain-Strategien und -Strukturen im ProC/B-Paradigma

03021 Michael Kaczmarek: Beschreibung ausgewählter Strategien zur Steuerung der Austauschprozesse in der Supply Chain

03022 Michael Kaczmarek: Organisation der Planung und Steuerung in Supply Chains

03024 Anne Schulze im Hove, Frank Stüllenberg, Stefan Weidt: Inhaltliche Ausgestaltung der Netzwerk-Balanced-Scorecard für Beschaffungsketten

03029 Hilmar Heinrichmeyer, Andreas Reinholz: Entwicklung eines Bewertungsmodells für die Depotstandortoptimierung bei Servicenetzen

03032 Marco Motta, Iwo Riha, Stefan Weidt: Simulation eines Regionallagerkonzeptes

03034 Frank Laakmann, Iwo Riha, Niklas Stracke, Stefan Weidt: Workbenchgestützte Konstruktion von Beschaffungsketten

03035 Iwo Riha, Stefan Weidt: Entwicklung einer Bewertungssystematik für Beschaffungsketten

04001 André Alberti, Bernd Hellingrath, Stefan Weidt, Markus Witthaut: Ergebnisse und Schlussforderungen der Simulationsexperimente im Szenario Automobilindustrie

04002 Kay Hömberg, Dirk Jodin, Maren Leppin: Methoden der Informations- und Datenerhebung

04003 Carsten Tepper: Prozessablauf-Visualisierung von ProC/B-Modellen

05001 Jochen Bernhard, Miroslav Dragan, Sigrid Wenzel: Evaluation und Erweiterung der Kriterien zur Klassifizierung von Visualisierungsverfahren für GNL

05002 Entwicklung eines Anaylserahmens für die Untersuchung organisatorischer Aspekte in der Supply Chain

05003 Einsatz der Response Surface Methode zur Optimierung komplexer Simulationsmodelle

Alle Technical Reports können im Internet unter
http://www.sfb559.uni-dortmund.de/
abgerufen werden. Für eine Druckversion wenden Sie
sich bitte an die SFB-Geschäftsstelle
e-mail: grosseca@iml.fhg.de