IGTE Institut für Grundlagen und Theorie der Elektrotechnik

TU Graz Graz University of Technology

virtual vehicle

Diplomarbeit

Robuste Optimierung mit stochastischen Verfahren

durchgeführt von

Claudia Skrabl

Institut für Grundlagen und Theorie der Elektrotechnik
Technische Universität Graz

Vorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Oszkár Bíró

Begutachterin: Dipl.-Ing. Dr.techn. Alice Reinbacher-Köstinger

Graz, Februar 2013

Danksagung

Diese Diplomarbeit ist am Institut für Grundlagen und Theorie der Elektrotechnik an der Technischen Universität Graz in Kooperation mit dem Kompetenzzentrum Das Virtuelle Fahrzeug Forschungsgesellschaft mbH entstanden.

Ich möchte mich in diesem Zusammenhang vor allem bei Frau Alice Reinbacher-Köstinger bedanken, die sich für meine Anliegen immer Zeit genommen hat und mir bei aufkommenden Fragen immer hilfreich unter die Arme gegriffen hat. Christian Magele vom Instituf für Grundlagen und Theorie der Elektrotechnik sowie Herrn Michael Alb vom Kompetenzzentrum Das Virtuelle Fahrzeug möchte ich für die Möglichkeit danken, dieses Thema bearbeiten zu dürfen.

Meiner gesamten Familie gebührt der größte Dank. Ohne sie wäre das Studium niemals möglich gewesen. Danke für die liebevolle Unterstützung und Begleitung.

Graz, Februar 2013 Claudia Skrabl

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..

(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..

date (signature)

Kurzfassung

Viele der heutigen Problemstellungen verlangen nach einer Möglichkeit, diverse Aspekte innerhalb eines Optimierungsprozesses auf einmal zu berücksichtigen. So werden z.B. unterschiedliche Anforderungen an ein Problem gestellt und die Aufgabe des Optimierers ist es nun, eine optimale Lösung zu finden, die allen Anforderungen gerecht wird. Es sollen hier einerseits unterschiedliche Methoden vorgestellt werden, um ein solches Optimierungsproblem behandeln zu können, und andererseits soll auch die Möglichkeit der Variabilität gewisser Parameter und deren Auswirkung auf die optimale Lösung gezeigt werden.

Eine optimale Lösung ist somit nicht immer gleich eine Lösung, welche realisiert werden kann. Parameter unterliegen in der Realität unterschiedlichen Einflüssen. Dabei ist es wichtig, zu zeigen, dass diese Parameterveränderungen die Lösung stark beeinflussen können und damit von einem theoretischen, optimalen Ergebnis zum Teil weit entfernt sind. Die Suche nach einem so genannten robusten Optimum ist zudem auch sehr rechenintensiv, womit auch ein Kompromiss zwischen Aufwand und Nutzen getroffen werden muss.

Abstract

Many of today´s problem statements request for a probability to optimize a process with respect to different aspects. The solution of this problem depends on many different requirements of this problem. So it isn´t very simple to meet with all these requirements. This work should reveal which different possibilities there are to solve such optimization problems. Furthermore, it should be shown what happens to the optimal solution, if the parameters are changing.

An optimal solution isn´t really always a solution, which can be realized. Actually the parameters are liable to different influences. And it is very important to show that changes in parameters have a direct effect on the solution so the robust optimum could be different to the theoretical optimal solution. The searching of an optimum is very computationally intensive, so there one needs to aim at a compromise between benefit and cost.

Inhaltsverzeichnis:

Kapitel 1: Einleitung

„When something is optimal, it is essentially perfect, impossible to improve, non-plus-ultra. …
Simulation, in fact, enables to obtain designs that are not optimal, but fit. Robust designs versus fragile designs. Optimization is actually just the opposite of robustness.“ (Jacek Marczyk, 2000) [1]

Ziel einer Optimierung ist das Finden eines Maximums bzw. eines Minimums einer sogenannten Zielfunktion. Um ein solches Maximum bzw. Minimum zu finden, werden verschiedene Verfahren angewendet. (Anmerkung: im weiteren Verlauf der Arbeit wird das Suchen nach einem Minimum bzw. Maximum nur noch als das Suchen nach einem Optimum bezeichnet werden.) Dabei kommt es immer auf die Problemstellung an sich an, welches Verfahren zum Einsatz kommt. Optimierungsverfahren werden in deterministische, sprich auf Gradienten bezogene Verfahren, und in stochastische, auch heuristische Verfahren genannt, eingeteilt.

Stochastische Verfahren wie z.B. die Evolutionsstrategie, sind für viele Anwendungen in der Technik am besten geeignet. Im Vergleich zu den deterministischen Verfahren benötigen sie meist einen wesentlich höheren Rechenaufwand und waren daher eine sehr lange Zeit nicht reell anwendbar. Durch den exponentiellen Anstieg der Prozessorleistung heutiger Geräte, kommen sie immer mehr zum Einsatz. Nichts desto trotz sollte auf eine effiziente Programmierung geachtet werden. Parallele Datenverarbeitung ist in diesem Zusammenhang ebenfalls von Vorteil, um die Rechenzeit herabzusetzen.

Es sei weiter zu beachten, dass die Zielfunktion in einem Optimierungsprozess immer nur ein Modell, also eine Näherung eines realen Problems, darstellt. Das bedeutet, dass das mit Hilfe des Modells gefundene Optimum in der Realität vom tatsächlichen Optimum mehr oder weniger stark abweichen kann. Einen weiteren Aspekt der Optimierung stellt die Realisierung oder Herstellung eines Prozesses bzw. eines Bauteils, Geräts, einer Anlage etc. dar. Jeder Prozess in einer Entwicklung unterliegt diversen Herstelltoleranzen. Würde man jeden Herstellungsschritt präzise durchführen, wäre die Produktion zu kostspielig. Es sollten also auch hier gewisse Kompromisse getroffen und in Betracht gezogen werden. Optimierungsmodelle verhalten sich statisch, was wiederum einem realen Verhalten widerspricht. So sind Zielfunktion und Nebenbedingungen dynamische Prozesse, die z.B. von Temperaturschwankungen, Änderungen des Druckes oder anderen Störquellen abhängig sind und nur für gewisse Zeitspannen, in denen sich die Parameter nicht ändern, ihre Gültigkeit bezüglich des Optimierungsmodelles behalten.

In Anbetracht dieser Abweichungen des Modells von der wahren Funktion, wurde die Idee entwickelt, die Optimierung robust bezüglich gewisser Schwankungen zu gestalten. Gen´ichi Taguchi (1924-2012), ein japanischer Ingenieur, Statiker und Experte für Qualitätsverbesserungen gilt hierbei als Erfinder der Robusten Optimierung. Er entwickelte das Drei-Stufen-Modell, um diese Schwankungen und Prozessparameter in die Projektierung eines robusten Modells mit einfließen zu lassen. Dabei bezieht sich die erste Stufe auf die Entwicklung des Systems. Hier werden die grundlegende Struktur und die vom System verwendeten Parameter festgelegt. Die zweite Stufe bezieht sich auf die Entwicklung der Parameter, die das System beeinflussen werden. Hier sollen die Parameter den Qualitätsanforderungen genügen. Die dritte Stufe bezieht sich auf die Gestaltung der Toleranzbereiche und stellt nur eine Feinabstimmung der Parameter dar, [3]. In weiterer Folge werden Schwankungen der Parameter betrachtet, da sich die Gestaltung der Toleranzbereiche wiederum auf die Parameter bezieht, mit dem Unterschied, dass hier die Nebenbedingungen andere sein können (in [3] werden hier die Design-Zeit vs. Betriebszeit als Beispiel genannt).

Eine robuste Ausführung einer Optimierungsmethode zieht immer einen erheblichen Mehraufwand an Rechenoperationen nach sich. Das kommt daher, dass für jeden vom Optimierungsprozess vorgeschlagenen Punkt zusätzlich die Umgebung durch Variation der Parameter betrachtet werden muss. Ein robustes Verfahren ist erst dann erreicht, wenn die Zielfunktion gegenüber Veränderungen der Eingangsparameter unempfindlich wird. Dabei kann es durchaus so weit gehen, dass eine robuste Lösung gefunden wird, die allen Anforderungen entspricht, deren Performance aber so schlecht wird, dass sie nicht mehr als optimal betrachtet werden kann.

Kapitel 2 dieser Arbeit behandelt bekannte und erprobte stochastische Optimierungsverfahren. Dabei wird das Hauptaugenmerk auf die Differentielle Evolutionsstrategie gelegt, die in dieser Arbeit das Grundmodell für die Robuste Optimierung bildet.

In Kapitel 3 sollen robuste Methoden betrachtet werden. Dabei werden die Taguchi-Methode, die Worst-Case Methode, die Monte-Carlo Methode und die Latin-Hypercube Methode genannt. Es soll gezeigt werden, wodurch sich ein Optimum, welches sich durch einen allgemeinen Optimierungsprozess herauskristallisiert, von einem robusten Optimum unterscheidet und welche Möglichkeiten bestehen, die Parameterschwankungen, die zu einem robusten Optimum führen, zu generieren.

Kapitel 4 befasst sich mit der Implementierung der in Kapitel 3 vorgestellten robusten Methoden in den Optimierungsalgorithmus. Hier sollen zwei

Strategien (single loop und double loop - Strategie) vorgestellt werden. Wobei in dieser Arbeit nur die single-loop Strategie Verwendung finden wird.

Zum Schluss werden die behandelten Methoden und Verfahren in einem Beispiel vorgestellt. Dazu wurde vorerst eine eigens zu diesem Zweck konstruierte Testfunktion gebildet, welche ein globales Optimum und zwei unterschiedliche, lokale Optima aufweist. Die Differentielle Evolutionsstrategie soll als Optimierungsalgorithmus dienen und in weiterer Folge mit zwei in Kapitel 3 vorgestellten Methoden so erweitert werden, dass der entstehende Algorithmus ein robustes Optimum findet.

Kapitel 2: Stochastische Optimierungsverfahren

Optimierung bedeutet, dass Parameter eines Systems gefunden werden müssen, um eine Zielfunktion zu minimieren oder zu maximieren. Im mathematischen Sinn muss $f(\mathbf{x}_0) \leq f(\mathbf{x})$ für alle $\mathbf{x} \in \Re^n$ gelten, [2]. Somit wäre $\mathbf{x}_0$ das globale Minimum. Für das globale Maximum gilt analog das Gleiche nur ist $f(\mathbf{x}_0) \geq f(\mathbf{x})$. Ein lokales Minimum würde bedeuten, dass $f(\mathbf{x}_0) \leq f(\mathbf{x})$ für alle $\mathbf{x}$ aus einer Umgebung um $\mathbf{x}_0$ ist.

Um eine eindimensionale Funktion auf ihre Extremwerte hin zu überprüfen, müssen die erste und die zweite Ableitung gebildet werden. Dabei gilt, dass ein Extremwert gefunden wurde, wenn die erste Ableitung Null ist. Um das Krümmungsverhalten in diesem Punkt bestimmen zu können, wird die zweite Ableitung gebildet. Ist diese größer als Null, ist ein lokales Minimum erreicht. Ist die zweite Ableitung kleiner als Null, ist ein lokales Maximum erreicht. Ein lokales Minimum, von dem es mehrere im zulässigen Bereich geben kann, unterscheidet sich vom globalen Minimum durch einen schlechteren Funktionswert. Abbildung 1 soll dies verdeutlichen. Die hier angewendete Funktion lautet: $f(\mathbf{x}) = 2\sin(10\mathbf{x}e^{-0,08\mathbf{x}})e^{-0,25\mathbf{x}}$

Es ist zu erkennen, dass die in Abbildung 1 dargestellte eindimensionale Funktion, neben dem globalen Optimum, sieben lokale Optima aufweist.

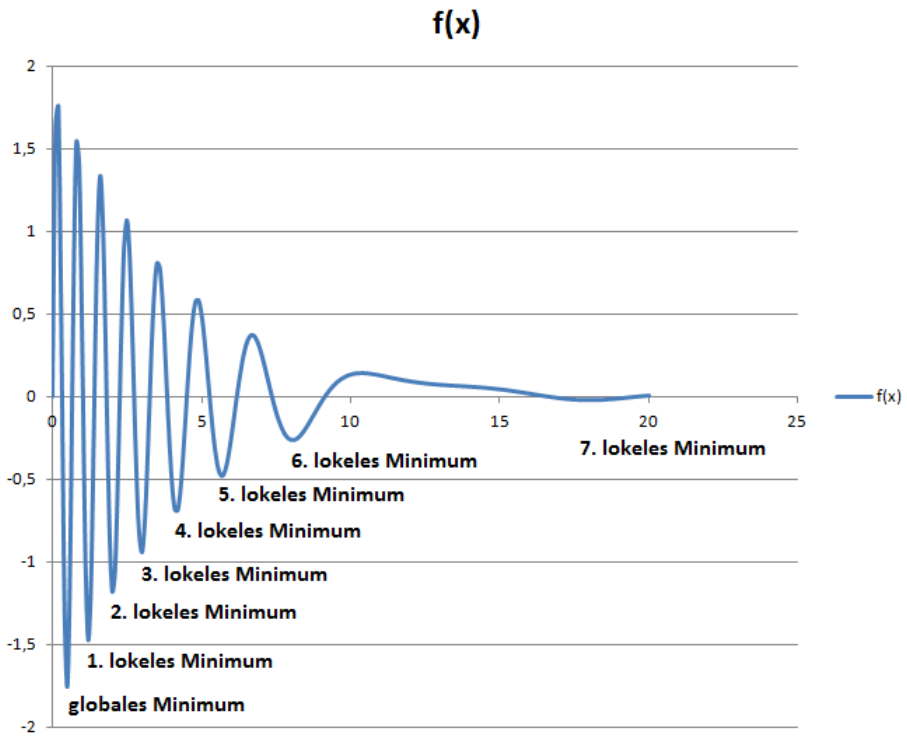

Abbildung 1 Globales Minimum und Lokale Minima einer eindimensionalen Funktion.

Im mehrdimensionalen Bereich $\Re^n$ müssen zur Untersuchung der Funktion der Gradient und die Hessematrix gebildet werden, [2].

$$\mathrm{g}(\mathbf{x}) \equiv \nabla \mathrm{f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial \mathrm{f}}{\partial \mathrm{x}_1} \\ \vdots \\ \frac{\partial \mathrm{f}}{\partial \mathrm{x}_\mathrm{n}} \end{pmatrix} \qquad (2.1)$$

$$G(\mathbf{x}) \equiv \nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \vdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \vdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \qquad (2.2)$$

Dabei ist der Gradient $g(\mathbf{x})$ aus (2.1) der Richtungsvektor, der immer in die Richtung des steilsten Anstieges zeigt. Wird er Null, gibt es keinen Anstieg und damit ist ein mögliches Minimum/Maximum gefunden (Optimalitätsbedingung 1. Ordnung: $\nabla \mathrm{f}(\mathbf{x}) = 0$, wenn an der Stelle $\mathbf{x}$ ein lokales Minimum existiert und dieses differenzierbar ist). Um die Funktion auf ein Minimum bzw. Maximum hin zu überprüfen, muss die Hessematrix $G(\mathbf{x})$ aus (2.2) gebildet werden. Ist die Hessematrix positiv definit, würde das bedeuten, dass die Steigung zunimmt und damit würde gelten, dass der Punkt $\mathbf{x}$ ein Minimum ist und damit existiert eine Lösung welche eindeutig lösbar ist. (Optimalitätsbedingung 2. Ordnung: $\nabla \mathrm{f}(\mathbf{x}) = 0$ und die Determinante der Hessematrix $G(\mathbf{x})$ (2.2) ist größer als null, $det\left(\nabla^2 f(x)\right) > 0$).

Optimierungsverfahren werden unter anderem durch folgende Eigenschaften klassifiziert:

- differenzierbar / nicht differenzierbar
- kontinuierlich / diskret
- linear / nicht linear
- restringiert (mit Nebenbedingungen) / nicht restringiert (ohne Nebenbedingungen)
- multimodal (nicht konvex) / unimodal (konvex)

Allgemein lässt sich ein Optimierungsproblem folgendermaßen anschreiben:

$$\min_{x \in \Re^n} f(\mathbf{x}) \qquad (2.3)$$

abhängig von: $g_i(\mathbf{x}) \leq 0 \quad i = 1 \ldots I$

$$h_j(\mathbf{x}) = 0 \quad j = 1 \ldots J$$

Diese Gleichungen stehen für ein nicht-lineares Optimierungsproblem [3]. Es ist das Minimum der nichtlinearen Funktion $f(\mathbf{x})$ für den n-dimensionalen Parametervektor $\mathbf{x}$ gesucht (2.3). $g(x)$ sind die

Ungleichheitsnebenbedingungen und $h(x)$ sind die Gleichheitsnebenbedingungen, die zusätzlich erfüllt sein müssen. Jeder Parametervektor $\mathbf{x}$ ist gültig, wenn er alle Nebenbedingungen erfüllt. Als einen zulässigen Raum bezeichnet man alle zulässigen Punkte $\mathbf{x}$, [4].

Angewendete Verfahren zur Optimierung der Funktion $f(\mathbf{x})$ teilen sich in deterministische Verfahren und stochastische (heuristische) Verfahren auf.

Die deterministischen Verfahren seien hier nur am Rande erwähnt, [4]. Sie werden in Methoden nullter, erster und zweiter Ordnung eingeteilt. Bei den Methoden nullter Ordnung wird keine Gradientenbildung im mathematischen Sinn benötigt, dazu zählen die Simplex Methode und Pattern Search (Methode von Hooke and Jeeves). Zu den Methoden erster Ordnung zählen die Quasi-Newton und die Gauss-Newton - Methode. Hier wird die benötigte Krümmungsinformation für den Iterationsprozess durch das Betrachten der Objektfunktion $f(\mathbf{x})$ und des Gradienten $g(\mathbf{x})$ erzeugt, ohne die Hessematrix $G(\mathbf{x})$ explizit bilden zu müssen. Zu den Methoden zweiter Ordnung gehören die Newton und die Levenberg-Marquardt - Methode. Hierzu werden der Gradient $g(\mathbf{x})$, (2.1), und die Hessematrix $G(\mathbf{x})$, (2.2) gebildet.

Ist die zu optimierende Funktion $f(\mathbf{x})$ stetig und differenzierbar, eignen sich deterministische Verfahren am besten. Bei unzureichender Kenntnis über das Verhalten der Objektfunktion bieten sich stochastische Optimierungsverfahren an, [4]. Sie eignen sich am besten, wenn die Zielfunktion mehrere lokale Optima und viele Unstetigkeiten aufweist.

Stochastische Verfahren lehnen sich an Beispielen der Natur an. Z.B. ahmt das Particle Swarm - Verfahren das Verhalten eines Schwarms untersuchter Lebewesen, wie Vögel oder Fische, auf der Suche nach Futter nach. Es ist eine Zufallssuche, aber nach zielgerichteten Methoden, und ist damit für komplexe Probleme anwendbar, über deren Eigenschaften man nichts oder nur wenig weiß. Ein weiterer Vorteil dieser Verfahren ist das Nichtbenötigen der Bildung der 1. und 2. Ableitung. Diese Berechnungen sind meist sehr zeitaufwendig und auch nur für spezielle Probleme in weiterer Folge anwendbar.

Das stochastische Verhalten dieser Methoden hat den Vorteil, dass es lokale Optima und Unstetigkeiten „überspringen“ kann und somit das globale Optimum findet. Dies geschieht dadurch, dass stochastische Verfahren auch Verschlechterungen der Funktionswerte zulassen können und damit nicht nur zielstrebig in eine Richtung wandern sondern auch die Umgebung nach einem möglichen, besseren Ergebnis absuchen. Sie verlassen also mit einer gewissen Wahrscheinlichkeit das lokale Optimum und begeben sich auf die Suche nach einem weiteren Ergebnis. Dabei kann es sein, dass sie das globale Optimum finden. Diesem Vorteil steht aber ein sehr hoher Rechenaufwand gegenüber, weshalb diese Verfahren erst durch die

Entwicklung leistungsstarker Rechnersysteme an Bedeutung gewonnen haben bzw. handhabbar wurden.

Stochastische Methoden sind Methoden nullter Ordnung und haben damit den Vorteil ohne Gradientenbildung auszukommen. Sie arbeiten nur mit dem Funktionswert der Objektfunktion $f(\mathbf{x})$ und können somit während des Optimierungsprozesses die Kontinuität (und damit den Gradienten) der Objektfunktion außer Acht lassen [4].

Abbildung 2 zeigt die grundlegenden Bestandteile stochastischer Algorithmen. Dabei kann sich die Reihenfolge, in der die Bestandteile abgearbeitet werden auch von Algorithmus zu Algorithmus unterschiedlich sein. So ist z.B. bei der Evolutionsstrategie nach der Initialisierung die Rekombination, dann die Mutation und zuletzt die Selektion während bei der differentiellen Evolutionsstrategie nach der Initialisierung die Mutation folgt und erst anschließend die Rekombination sattfindet.

Abbildung 2 Hauptbestandteile stochastischer Verfahren

Initialisierung: Hier wird festgelegt, wie groß die anfängliche Population ist, sprich aus wie vielen Individuen sie besteht und wie diese im Suchbereich verteilt werden. Die Verteilung im Suchbereich sollte zufällig geschehen, um die Individuen auf der gesamten Fläche gleichmäßig zu verteilen und somit den gesamten Suchbereich abdecken zu können. Des weiteren werden noch die Parameter konfiguriert, die Anzahl der Iterationsschritte und die anfänglichen Schrittweiten festgelegt.

Rekombination: Durch Kombination der Eigenschaften von Individuen (Eltern) entstehen neue Individuen (Kinder), wodurch Diversität im Parameterraum realisiert wird. Als Eltern werden meist jene ausgewählt, die bessere Funktionswerte aufweisen, also fitter sind als andere Individuen ihrer Generation.

Mutation: Als Mutation wird eine zufällige Veränderung der Individuen verstanden. Jeder einzelne Parameter wird also mit einer bestimmten Wahrscheinlichkeitsverteilung

(z.B. normalverteilt) modifiziert. Wie groß die Veränderung sein kann, wird mit Hilfe der sogenannten Schrittweite festgelegt.

Selektion: Hier wird die Entscheidung getroffen, ob das neu entwickelte Individuum bzw. die neu entwickelten Individuen, in die nächste Generation aufgenommen werden. Jene Individuen mit den besseren Funktionswerten bilden die nächste Generation. Dieser Schritt ist der entscheidende für die Effektivität des Algorithmus. Er spiegelt die Bedingungen aus Vorbildern der Natur wieder und zwar, dass nur jene Individuen überleben können, die sich an die Umgebung besser und schneller anpassen können.

Zu Beginn eines stochastischen Verfahrens steht immer die Initialisierung einer Population. Bei der Mutation wird festgelegt, wie sich die Individuen der Population verändern. Hier wird die Wahl getroffen, welcher Algorithmus angewendet wird (z.B. Simulated Annealing, Particle Swarm, Evolutions- und Genetische Algorithmen). Unter der Rekombination versteht man die neue Verteilung des Informationsmaterials zweier Eltern, um ein Individuum der nächsten Generation zu zeugen. Dabei werden u. a. z.B. Schrittweiten an die nächste Generation „vererbt“. Bei der Selektion wird entschieden, welches Individuum der ersten Generation zur Bildung der nächsten Generation herangezogen wird. Dabei gibt es je nach Algorithmus unterschiedliche Verfahren. Man möchte, dass sich die Qualität von Generation zu Generation zwar verbessert, sollte aber bis zu einem gewissen Grad auch eine Verschlechterung zulassen, um die Möglichkeit der Untersuchung der Umgebung zu haben und somit auf das globale Optimum zu stoßen. Z.B. besteht bei der $(\mu/\rho,\lambda)$ Evolutions-Strategie mit Roulette Wheel-Verfahren eine gewisse Chance für ein schlechteres Individuum an der Bildung der nächsten Generation teil zu nehmen. Abbruchkriterien sind bei diesen Verfahren sehr wichtig. Würde man keine Bedingungen schaffen, bei deren Auftreten der Rechenprozess abgebrochen wird, würde der Algorithmus ewig rechnen. Abbruchkriterien sind für jeden Algorithmus unterschiedlich festzulegen. Im großen und ganzen kann aber gesagt werden, dass die Berechnung gestoppt werden soll, wenn sich der Funktionswert nicht mehr verbessert oder eine maximale Anzahl an Generationen erreicht wurde, sich die Population zu stark in einem Punkt sammelt (Particle Swarm) oder sich der Funktionswert über einige Generationen hinweg nicht mehr sehr stark ändert (das muss aber nicht automatisch bedeuten, dass das globale Optimum gefunden wurde, sondern wie bei Particle Swarm sammeln sich die Punkte und sind „festgefahren“.)

Jede stochastische Methode baut auf den Faktor Zufall auf (die Startpunkte werden zufällig festgelegt). Dadurch erscheint es nicht sinnvoll einen solchen

Algorithmus nur einmal zu durchlaufen. Es sollten immer mehrere Runs durchgeführt werden, um die Wahrscheinlichkeit, das globale Optimum auch wirklich zu finden, zu erhöhen.

2.1 Simulated Annealing

Simulated Annealing ist die Nachbildung eines Abkühlungsprozesses. Wenn sich geschmolzenes Metall langsam abkühlen kann, haben die Moleküle Zeit, den Grundzustand (energiearmer Zustand) zu erreichen [4]. Das Gegenbeispiel stellt eine sehr rasche Abkühlung dar. Dabei haben die Moleküle keine Zeit einen energiearmen Zustand einzunehmen und es verbleiben Unregelmäßigkeiten in der Gitterstruktur. Dieser Prozess baut auf der Boltzmann-Wahrscheinlichkeitsverteilung auf, die in (2.4) zu sehen ist. E ist dabei die Energie eines Teilchens, welche durch $E = mv^2/2$ bestimmt ist, k_B ist die Boltzmann-Konstante und T die Temperatur.

$$p_B = e^{-\frac{E}{k_B T}} \qquad (2.4)$$

Im Simulated Annealing – Verfahren ist die Wahrscheinlichkeitsverteilung abhängig von der Zielfunktionsverbesserung $f_{new} - f_{old}$, der Temperatur T und einem Normalisierungsfaktor c_{norm}, wie in (2.5) dargestellt.

$$p = e^{-\frac{f_{new} - f_{old}}{c_{norm}\ T}} \qquad (2.5)$$

Die Temperatur T verringert sich innerhalb einiger Iterationen von $1\ gegen\ 0$ wobei die Schrittweite problemabhängig und frei wählbar ist. Je näher die Temperatur gegen Null geht, desto unwahrscheinlicher ist die Akzeptanz des neuen Zielfunktionswertes $f(\mathbf{x})_{new}$ bei gleicher Zielfunktionswert-Differenz. Das bedeutet, dass bei steigender Iterationszahl, die Möglichkeit der Verschlechterung verringert wird und das Verfahren im nächstgelegenen (lokalen) Optimum konvergiert.

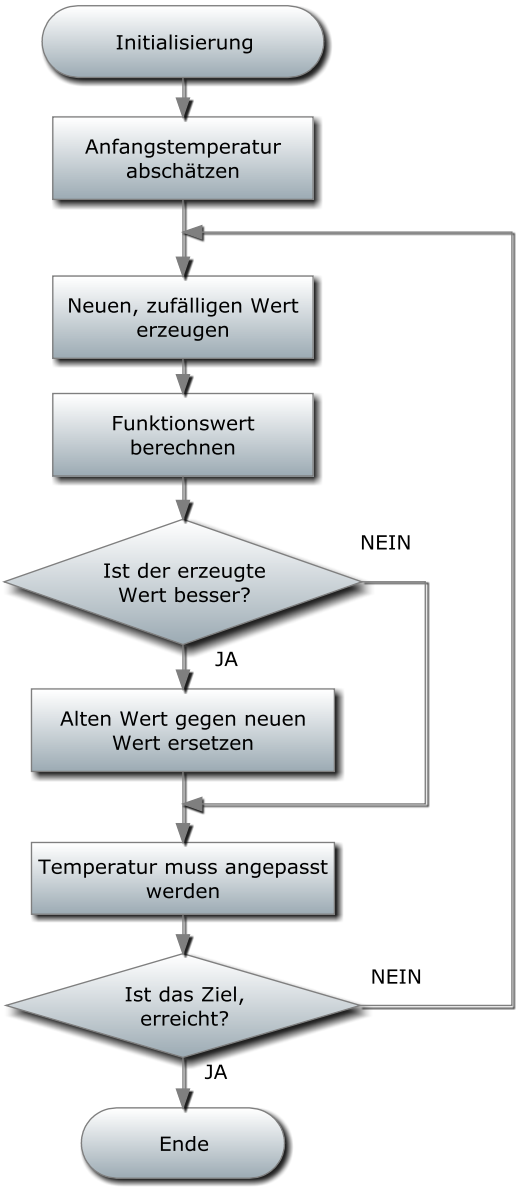

Abbildung 3 Flussdiagramm Simulated Annealing [18]

Abbildung 3 zeigt das generelle Flussbild des Simulated Annealing Verfahrens. Die Temperatur wird immer weiter nach unten gesetzt. Durch die zufällige Wahl einer Temperatur in Anlehnung an die aktuelle entsteht die Möglichkeit, dass sich das Ergebnis auch verschlechtern kann. Damit ist wiederum die Möglichkeit gegeben, den Umgebungsbereich auch anzusehen.

2.2 Particle Swarm

Bei diesem Verfahren wird das natürliche Schwarmverhalten von Vögeln oder Fischen nachgeahmt [4]. Die Individuen suchen den Parameterraum nach dem Optimum ab und kommunizieren währenddessen miteinander. Dabei orientieren sie sich am Individuum mit der besten „Fitness“. Die Individuen werden durch ihre Position x und ihre Geschwindigkeit v beschrieben. Das Schwarmverhalten wird durch die Differentialgleichungen (2.6) und (2.7) beschrieben.

$$v_{t+1} = \left(1 - \frac{\sqrt{\kappa\Phi}}{2}\right) v_t + \kappa\Phi\left(x_t^{best} - x_t\right) \qquad (2.6)$$

$$x_{t+1} - x_t = \left(1 - \frac{\sqrt{\kappa\Phi}}{2}\right) v_t + (1 - \kappa\Phi)\left(x_t^{best} - x_t\right) \qquad (2.7)$$

In (2.6) und (2.7) ist x_t^{best} der aktuelle „Anführer“ des Schwarms, also das „fitteste“ Individuum im Schwarm, $\boldsymbol{\Phi}$ ist ein zufälliger Faktor zwischen Null und Eins, und $\boldsymbol{\kappa}$ kontrolliert die Beweglichkeit des Schwarms.

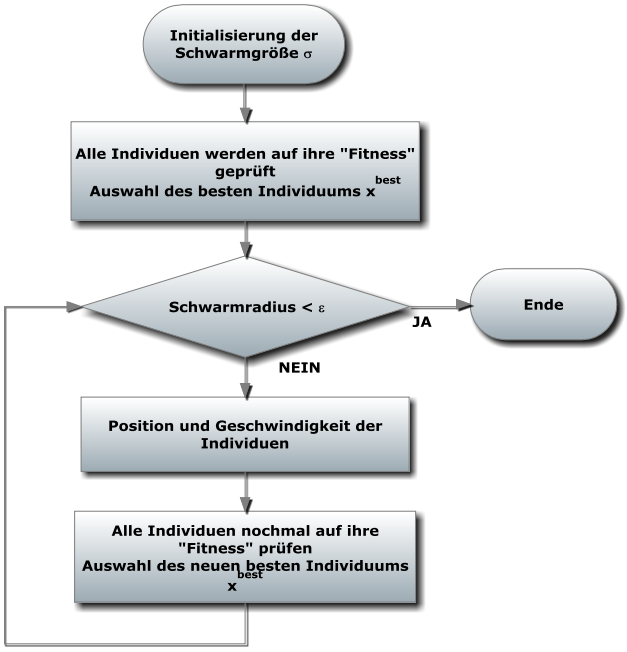

Abbildung 4 Flussdiagramm Particle Swarm, [4]

Abbildung 4 zeigt das Flussdiagramm einer Particle Swarm Methode. Um ein gutes Ergebnis zu erhalten, sollte die Populationsgröße nicht zu klein gewählt werden (z.B. ca. 30 -100 Individuen). Durch das Einführen von Gewichtungsfaktoren in (2.8) kann der Suchverlauf beeinflusst werden, [5]. Dabei wurde festgestellt, dass für ein gutes Ergebnis c_2 etwas größer als c_1 sein sollte, sprich der Teil der Formel, der sich auf das Globale Optimum bezieht, sollte etwas stärker in das

Gesamtergebnis einfließen. u_d und U_d entsprechen hier zufällig gewählten Werten zwischen Null und Eins.

$$v_{t+1} = wv_t + c_1 u_d \left[x_t^{best} - x_t\right] + c_2 u_d \left[x_t^{globalbest} - x_t\right] \qquad (2.8)$$

Der Algorithmus wird beendet, wenn der Radius des Schwarms kleiner als eine gewählte Konstante ε wird, oder die maximale Anzahl an Funktionsaufrufen erreicht wurde.

2.3 Genetischer Algorithmus

Jede Population besteht aus mehreren Individuen. Diese werden in Zeichenketten codiert und zu einem Strang zusammengefügt, der in weiterer Folge als Chromosom bezeichnet wird. Zu Beginn des Algorithmus wird eine zufällig gewählte Zeichenkette erzeugt. Der Faktor Zufall spielt hier eine entscheidende Rolle, da man festgestellt hat, dass, wenn die Zeichenketten zufällig gewählt wurden, die Wahrscheinlichkeit das globale Optimum zu finden, höher ist. Verwendet man beeinflusste Anfangswerte, kann es sein, dass der Algorithmus zu einem lokalen Optimum tendiert, und damit eine zu frühe Konvergenz eintritt. Ist die zufällige Anfangszeichenkette gebildet, wird die Überlebenswahrscheinlichkeit durch Berechnung des Funktionswertes gebildet. Die Funktionswerte werden für jedes Chromosom gebildet und miteinander verglichen. Je besser der Funktionswert ist, desto größer ist der Wahrscheinlichkeitswert, und damit ist die Wahrscheinlichkeit, bei der Selektion zu überleben, am größten. Es folgen Mutation und Crossover-Operationen. Diese verändern die Chromosomen derart, dass neue, bessere Chromosomen entstehen. Dies geschieht so lange, bis ein Abbruchkriterium erreicht wurde und damit die Berechnung beendet wird. Als Abbruchkriterien gelten in diesem Fall: z.B. die max. Anzahl an Generationen ist erreicht, keine Verbesserung des Funktionswertes oder aber es wurde ein gewisser Wert, der die mittlere Verbesserung der vorangegangenen Generation wiederspiegelt, unterschritten. [6]

2.4 Evolutions-Strategien

Der Grundgedanke dieser Strategie ist die Evolutionstheorie nach Charles Darwin die besagt, dass sich die Natur den Gegebenheiten anzupassen versucht. Durch Selektion werden schwächere Individuen ausgeschieden und schaffen es nicht in die nächste Generation.

Es wurde nun versucht, diesen Grundgedanken für die Optimierung zu nutzen. Dabei gehen hier keine genetischen Strukturen in die Berechnung ein, sondern die Erbinformation wird als ein Vektor von Zahlen, nämlich den Parameterwerten, betrachtet. Der grundlegende Verlauf sieht folgendermaßen aus: Zuerst wird eine Population gebildet, die aus einer bestimmten Anzahl von Individuen besteht. Aus diesen Individuen werden durch Rekombination Nachkommen gezeugt. Anschließend mutieren diese Nachkommen, um Diversität zu behalten. Die mutierten Individuen werden auf ihre Fitness hin überprüft und entsprechend ihrer Zielfunktionswerte sortiert. Entsprechend der Größe der initialen Populationsgröße werden zum Schluss die besten Individuen selektiert und in die nächste Generation übernommen. Dieser Algorithmus arbeitet so lange, bis ein Abbruchkriterium erfüllt ist. Abbruchkriterien sind z.B. das Erreichen des Optimums oder eines entsprechenden Schwellwertes, die Konvergenz der Zielfunktion, das Unterschreiten eines Schwellwertes der Populationsdiversität oder letztendlich, wenn eine maximale Anzahl an Generationen erreicht ist, [7].

Es entstanden daraus die (1+1)-Evolutionsstrategie oder z.B. die (μ/ρ, λ)-Evolutionsstrategie, [4].

Die (1+1)-Evolutionsstrategie stellt hier noch die einfachste Form dar. Das Duplikat eines einzigen Individuums mutiert durch Addition eines elementweise normalverteilten Parametervektors, und der daraus resultierende Funktionswert wird mit dem Funktionswert des Original-Individuums (Elter) verglichen. Derjenige mit der besseren Qualität wird anschließend zum neuen Elter. Dabei kann es sein, dass ein Elter über alle Generationen hinweg den besten Funktionswert besitzt und damit die Richtung, in die der Algorithmus konvergiert, vorgibt, [7].

Bei der (μ/ρ,λ)-Evolutionsstrategie werden aus einer Gruppe potentieller Eltern (ρ) mit Hilfe der Roulette-Wheel - Methode zwei Eltern gewählt, die an der Bildung der Kinder (λ) der nächsten Generation beteiligt sein werden. Dabei wird die Schrittweite für die nächste Generation auch verändert. Der Vorteil der Roulette-Wheel - Methode ist, dass auch ein qualitativ schlechteres Individuum an der Bildung der Kinder teilnehmen kann. Die Wahrscheinlichkeit ist zwar eher gering, wenn man sich vorstellt, dass die Fläche des Roulette-Wheels für ein schlechteres Individuum kleiner ist als für eines, dessen Qualitätswert größer ist, aber sie ist gegeben und damit auch die Möglichkeit, ein lokales Optimum zu verlassen und nach einem anderen Optimum zu suchen. Der vorige Schritt wird als Rekombination bezeichnet. Bei der anschließenden Mutation werden die gebildeten Kinder mit einem

normalverteilten Vektor addiert. Es folgt die Selektion, bei der die besten Kinder (μ) in der nächsten Generation zu neuen Eltern werden, [4].

2.5 Differentielle Evolutionsstrategie

Die Differentielle Evolutionsstrategie wurde von Storn und Price im Jahre 1995 entwickelt, [8], [9], und hat den Vorteil gegenüber anderen Evolutionsstrategien, in relativ kurzer Zeit ein brauchbares Ergebnis zu liefern. Dabei wird darauf geachtet, dass sich die Population immer in die bessere Richtung hin entwickelt, sprich, es werden keine Verschlechterungen zugelassen. Der Beste gewinnt. Die Schritte der Differentiellen Evolutionsstrategie sind Initialisierung, Mutation, Rekombination und Selektion siehe Abbildung 5. Hier wird im Vergleich zu der Evolutionsstrategie die Mutation vor der Rekombination durchgeführt.

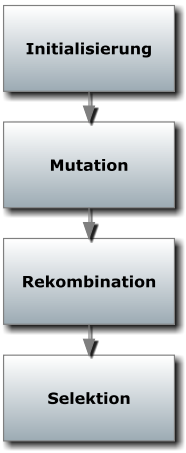

Abbildung 5 Grober Aufbau der Differentiellen Evolutionsstrategie

Die Differentielle Evolutionsstrategie ist auch für Funktionen, die nichtlinear, nicht differenzierbar, multimodal sind und rauschen anwendbar und ist in vielen Fällen wesentlich effektiver als andere bekannte Verfahren. Der Algorithmus verwendet drei Steuerparameter, den Skalierungsfaktor F, den Crossover-Faktor c_r und die Populationsgröße, welche alle drei einen starken Einfluss auf die Effizienz und Leistungsfähigkeit haben, da sie auf die Konvergenz und auf die Geschwindigkeit einwirken. Dabei ist der Skalierungsfaktor am ausschlaggebendsten. In [14] werden die unterschiedlichen Auswirkungen der Veränderungen dieser Steuerparameter näher behandelt und an zehn bekannten Testfunktionen probiert.

2.5.1 Mutation

Die Individuen werden durch Rekombination von Differenzvektoren verändert. Dabei wird die allgemeine Formel:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F_1(\mathbf{x}_{r2} - \mathbf{x}_{r3}) \qquad (2.8)$$

angewendet. $\mathbf{v_i}$ (2.8) stellt einen neuen Vektor dar, der sich aus einem zufällig gewählten Individuum der Population ($\mathbf{x}_{r1}$) und einer mit dem Skalierungsfaktor F multiplizierten Differenz zweier weiterer, zufällig gewählter Individuen ($\mathbf{x}_{r2} - \mathbf{x}_{r3}$) bildet. Dieser Schritt wird als Mutation bezeichnet. Um die Mutation effektiver durchzuführen, gibt es verschiedene Ansätze, z.B. den, dass man das Individuum mit der besten Performance in die Berechnung mit einbezieht. Bei diesem stochastischen Verfahren gilt ebenfalls, dass die Population nicht zu klein gewählt werden darf. Man sieht in (2.8), dass man für die Mutation mindestens drei Individuen benötigt. Ist die Population zu klein, wird der Faktor „Zufall“ bei der Auswahl der Individuen, die an der Mutation teilnehmen, zu klein, und damit geht eine wichtige Eigenschaft dieses Verfahrens verloren.

$$\mathbf{v}_i = \mathbf{x}_{best} + F_1(\mathbf{x}_{r2} - \mathbf{x}_{r3}) \qquad (2.9)$$

$$\mathbf{v}_i = \mathbf{x}_{r1} + F_1(\mathbf{x}_{r2} - \mathbf{x}_{r3}) + F_2(\mathbf{x}_{best} - \mathbf{x}_{r1}) \qquad (2.10)$$

In (2.19) wird das Individuum mit der besten Performance zu der „gewichteten“ Differenz zweier zufällig gewählter Individuen addiert, während in (2.9) ein zufällig gewähltes Individuum mit zwei unterschiedlich „gewichteten“ Differenzen addiert wird. Dabei handelt es sich immer um Individuen der gleichen Generation.

$\mathbf{x}_{ri}$ ist immer ein Vektor eines zufällig gewählten Individuums. Es sei ebenfalls erwähnt, dass die hier vorgestellten Gleichungen nicht die einzigen Möglichkeiten darstellen, um eine Differentielle Evolutionsstrategie durchführen zu können. In [15] wurden verschiedene Differentielle Evolutionsverfahren mit unterschiedlichen Testfunktionen ausprobiert. Darunter waren:

$$\mathbf{v}_i = \mathbf{x}_{r3} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \quad (2.8)$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \quad (2.9)$$

$$\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) + F(\mathbf{x}_{r3} - \mathbf{x}_{r4}) \quad (2.11)$$

$$\mathbf{v}_i = \mathbf{x}_i + F(\mathbf{x}_{best} - \mathbf{x}_i) + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) \quad (2.12)$$

$$\mathbf{v}_i = \mathbf{x}_{r5} + F(\mathbf{x}_{r1} - \mathbf{x}_{r2}) + F(\mathbf{x}_{r3} - \mathbf{x}_{r4}) \quad (2.13)$$

Das Ergebnis dieser Versuchsreihe war, dass die Differentiellen Evolutionsverfahren (2.9), (2.11) und (2.12) eine höhere Konvergenzgeschwindigkeit aufwiesen aber öfter auch in einem lokalen Optimum konvergierten. Während die Verfahren aus (2.8) und (2.13) zwar eine höhere Konvergenz zum globalen Optimum aufwiesen, aber die Konvergenzgeschwindigkeit nicht so hoch war.

Nichtsdestoweniger hängt die Wahl der verwendeten Variante immer noch von der jeweiligen Aufgabenstellung ab.

2.5.2 Rekombination

Bei der Rekombination werden funktionierende Lösungen der vorigen Generation mit neuen Individuen gemischt und daraus ein neuer Vektor $\mathbf{u}$ gebildet. Siehe dazu (2.14).

$$\mathbf{u}_{i,j} = \begin{cases} \mathbf{v}_{i,j} & \text{für } rand_{i,j} \leq c_r \quad oder\ j = irand \\ \mathbf{x}_{i,j} & sonst \end{cases} \quad (2.14)$$

Der neu generierte Vektor $\mathbf{u}_{i,j}$ besteht entweder aus Individuen, welche aus der Mutation hervorgehen oder aus einem nicht mutierten Individuum. Dabei wird eine wiederum zufällig gewählte Zahl zwischen 1 und 0 ($rand_{1,j}$) mit dem Crossover-Faktor verglichen. Ist die zufällig gewählte Zahl kleiner oder gleich wie der Crossover-Faktor, wird das neu gebildete Individuum in den Vektor eingetragen. Es werden hier die einzelnen Parametereigenschaften mit den Zufallszahlen und dem Crossover-Faktor verglichen. Entweder wird der neue Vektor $\mathbf{u}_{i,j}$ mit der mutierten Parametereigenschaft des Individuums aufgefüllt oder mit der ursprünglichen Parametereigenschaft des Individuums besetzt. Dadurch entsteht ein neuer Vektor aus zufällig gewählten Eigenschaften, die je nach Crossover-Faktor den Schwerpunkt entweder beim mutierten oder beim ursprünglichen Individuum haben.

2.5.3 Selektion

Bei der anschließenden Selektion werden die Funktionswerte miteinander verglichen. Siehe (2.15).

$$x_i^{k+1} = \begin{cases} u_i^k & \text{für } f\left(u_i^k\right) < f(x_i^k) \\ x_i^k & sonst \end{cases} \qquad (2.15)$$

Ist der Funktionswert des rekombinierten Individuums u_i^k kleiner, sprich besser (wenn das Minimum der Funktion $f(\mathbf{x})$ gefunden werden soll) als der Funktionswert des jeweiligen ursprünglichen Individuums x_i^k, so wird das rekombinierte Individuum in die nächste Generation aufgenommen. Andernfalls bleibt das ursprüngliche Individuum für die nächste Generation bestehen. Man erkennt anhand von (2.15), dass keine Verschlechterung des Funktionswertes zugelassen wird und damit ist eine zielgerichtete Konvergenz möglich.

Kapitel 3: Robuste Methoden

Grundsätzlich kann gesagt werden, dass die Berechnung der Zielfunktion innerhalb des Optimierungsprozesses nur eine Nachbildung eines realen Problems darstellt. Solange man also die Einzelheiten des realen Problems nicht kennt, wie z.B. das Verhalten der Nebenbedingungen, die sich mit der Zeit durch z.B. Temperaturschwankungen oder Druckveränderungen sowie Abnutzungserscheinungen verändern können, kann man nicht mit Gewissheit sagen, dass das gefundene Optimum des Modells auch das Optimum des realen Problems darstellt. Es sollte ebenfalls berücksichtigt werden, dass bei der Herstellung von Produkten Erstelltoleranzen vorhanden sein müssen. Würde man jeden Prozessschritt vollkommen präzise und für alle Produkte gleichbleibend durchführen, könnte man die Produktion dieses Produkts nicht mehr wirtschaftlich gestalten. Die Kosten für den Prozess würden ins Unermessliche steigen.

Zur Veranschaulichung sei hier ein Praxisbeispiel gegeben: Eine Anlage produziert in einer gewissen Zeit, eine gewisse Menge an Produkten. Diese Produkte können nicht alle komplett gleich produziert werden, da sich die Rahmenbedingungen von Stück zu Stück etwas verändern können. Es kommen am Ausgang also Produkte heraus, die einer gewissen Streuung unterliegen. Maschinell gesehen muss hier eine Streutoleranz eingeführt werden um die Produktion aufrecht erhalten zu können. Diese Toleranz befindet sich meist in einem prozentualen Bereich. Wenn also am Ende die Produkte geprüft werden, und sie befinden sich in diesem Toleranzbereich, werden sie weiterverarbeitet oder verkauft. Befinden sie sich außerhalb des Toleranzbereiches, ist dies die Ausschussware.

Diese Idee sollte der Leitgedanke der Entwicklung des robusten Algorithmus sein. Dabei gilt, man toleriert den prozentuellen Bereich um einen Parameter und ermittelt daraus die Lösung zur Verwirklichung des ursprünglichen Problems.

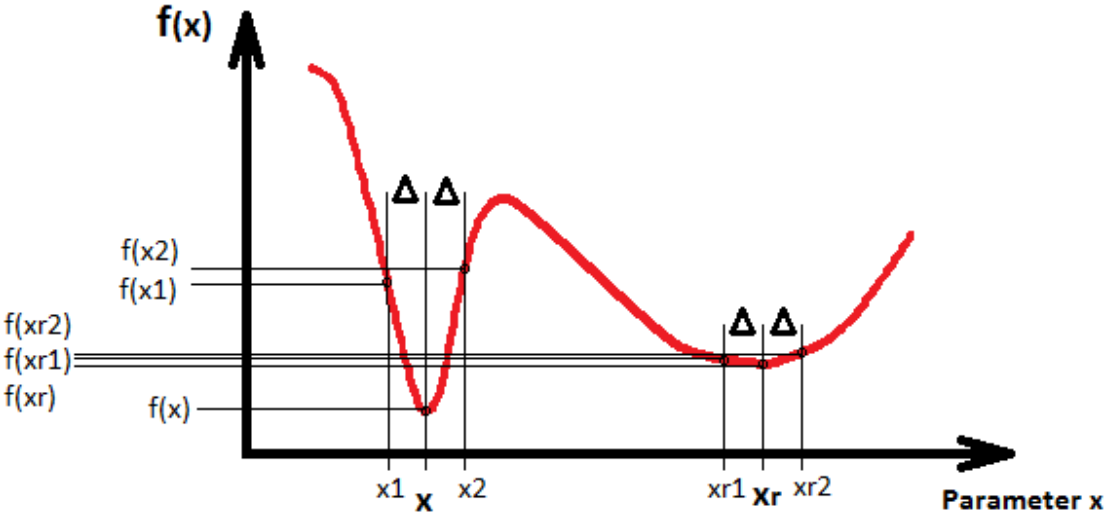

Abbildung 6 Skizze des Unterschiedes zwischen einem globalen und einem robusten Optimum

In Abbildung 6, [10], wird der Gedanke, der hinter einer robusten Optimierung steht, skizziert. Auf der x-Achse wird der Parameter *x* aufgetragen. Die y-Achse stellt den zugehörigen Funktionswert dar. Wenn man den Funktionsverlauf betrachtet, erkennt man, dass sich das globale Optimum an der Stelle **x** befindet. Hier ist der Funktionswert $f(\mathbf{x})$ am kleinsten und weist links und rechts relativ steile Flanken auf. Betrachtet man den Funktionsverlauf weiter, kann man ein lokales Minimum an der Stelle $\mathbf{x_r}$ erkennen, dessen Funktionswert $f(\mathbf{x}_r)$ größer als der von $f(\mathbf{x})$ und somit schlechter ist. Würde man das globale Optimum ohne Berücksichtigung der Robustheit suchen, wäre das Ergebnis $f(\mathbf{x})$ mit dem optimalen Parameter **x**. Wie eingangs erwähnt, können die Parameter aber um einen gewissen Bereich variieren. Dieser Bereich wird hier als Δ bezeichnet. Betrachtet man nun die Funktionswerte der Parameter $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_{r1}$ und $\mathbf{x}_{r2}$, stellt man fest, dass die Funktionswerte $f(\mathbf{x}_{r1})$ und $f(\mathbf{x}_{r2})$ kleiner, also optimaler sind als die Funktionswerte $f(\mathbf{x}_1)$ und $f(\mathbf{x}_2)$. Damit wird aus dem lokalen Optimum ein robustes Optimum. Da wir wissen, dass der stochastische Optimierungsalgorithmus die Bestrebung hat, das globale Optimum zu finden, werden die Individuen nur das globale Optimum bei $\mathbf{x}$ finden und das lokale Minimum überspringen. Somit muss der Suchalgorithmus so angepasst werden, dass er das globale Minimum übergeht und ein robustes Optimum findet.

3.1 Taguchi-Methode

In diesem Zusammenhang taucht immer wieder der Name Taguchi auf. Er gilt als Begründer der Philosophie, Optimierung so zu verändern, dass die Toleranz der Parameter mit einbezogen wird, [3].

Taguchi verwendet zum einen Kontrollparameter $\mathbf{x}$, welche zu optimieren sind, und zum anderen Rauschfaktoren ξ, die schwer zu kontrollierende, umweltbedingte oder toleranzbedingte Faktoren darstellen. Dabei findet hier keine Optimierung im herkömmlichen Sinn statt, es werden vielmehr *design of experiments* gebildet, um z.B. unterschiedliche x-Werte beurteilen bzw. abschätzen zu können. Es wird ein orthogonales Gitter aufgespannt. Die Kontrollparameter werden über das sogenannte „control“ Gitter systematisch gewechselt, und für jeden Kontrollparameter werden die Rauschvariablen gemäß eines „äußeren“ Gitters systematisch verändert. Anschließend wird ein zugehöriger Qualitätswert $y_i = y(x, \xi_i)$ generiert. Dieser Qualitätswert wird für jeden Kontrollparameter $\mathbf{x}$ mit seiner Rauschumgebung ξ_i separat gebildet. Aus all diesen Punkten wird die mittlere quadratische Abweichung, siehe (3.1), berechnet.

$$MSD_1 := \frac{1}{\kappa}\sum_{i=1}^{\kappa}(y(x,\xi_i) - \hat{y})^2 \qquad (3.1)$$

Die mittlere quadratische Abweichung aus (3.1) besteht aus der Summe aller Quadrate der Differenzen aller Qualitätswerte und des gewünschten Zielwertes $\hat{y}$. Diese Summe wird nochmals durch die Anzahl der Rauschfaktoren dividiert.

Aus der mittleren quadratischen Abweichung aus (3.1) wird das Signal/Rauschverhältnis gebildet (3.2), welches in Bezug zu den Kontrollparametern zu maximieren ist.

$$SNR := -10\log_{10}(MSD) \qquad (3.2)$$

Durch eine statistische Datenanalyse kann der Parameter $\mathbf{x}$, welcher das beste Ergebnis liefert, herausgefunden werden. Der Nachteil dieser Methode ist das große Datenvolumen. Man benötigt pro Dimension des Kontrollparameters mindestens zwei Punkte und somit 2^{Dim} designs und daraus entstehen über die Anzahl der Rauschfaktoren um jeden Kontrollparameter $\kappa 2^{Dim}$ Experimente. Wie man anhand dieser Datenflut erkennen kann, ist es in diesem Fall besser, die Parameteranzahl so gering wie möglich zu halten. Des Weiteren gibt es auch Aspekte, die die Sinnhaftigkeit der Definition von MSD und SNR hinterfragen (in Taguchi parameter design; a panel discussion). Nichtsdestoweniger ist diese Methode der Leitgedanke zur Entwicklung anderer Verfahren.

3.2 Worst-Case Methode:

Eine weitere Methode ist die Worst-Case Methode, [10]. Hier wird der Bereich der Varianzen abgegrenzt und nur der Maximalwert des Bereichs zur Berechnung verwendet. Dieses Verfahren setzt voraus, dass man den variablen Bereich, also die Parametertoleranzen, kennt und damit abgrenzen kann. Man kann sagen, dass sich um einen Mittelpunkt $\mathbf{x}$ eine rechteckige Fläche im zweidimensionalen Raum aufspannt. Im dreidimensionalen Raum wäre es ein Quader. Hier wird zunächst nur der zweidimensionale Fall betrachtet, (Abb. 7).

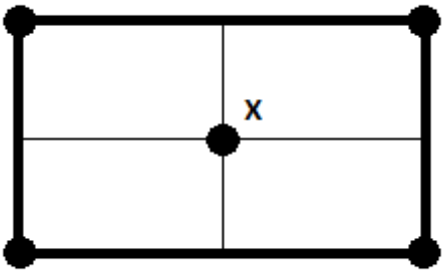

Abbildung 7 Aufspannen des Rechtecks um den Parameter x im 2D-Raum

Für jeden Eckpunkt wird der Funktionswert berechnet und jener mit dem schlechtesten Wert wird als Worst-Case – Punkt bezeichnet. Dem Punkt $\mathbf{x}$ wird also der Funktionswert des Worst-Case – Punktes anstatt des Wertes $f(\mathbf{x})$ zugeordnet. Wie später gezeigt wird, verändern sich damit auch die Zielfunktion zur robusten Zielfunktion $F(\mathbf{x})$, [10].

Der Vorteil dieser Methode ist, dass die Anzahl von Funktionsaufrufen eindeutig definiert und im niederdimensionalen Raum vergleichsweise gering ist. Im zweidimensionalen Fall etwa müssen 2^2, also vier Zielfunktionswerte pro vorgeschlagenen Punkt $\mathbf{x}$ zusätzlich berechnet werden. Bei einer Parameter-Dimension von n = 20 sind es allerdings bereits 2^{20} Punkte, also 1,05 Mio. Funktionsaufrufe.

Die Worst-Case-Methode rechnet immer nur die Eckpunkte um einen Parameter aus und rechnet mit dem schlechtesten Funktionswert dieser Punkte weiter. Dieses Verfahren berücksichtigt – wie der Name schon sagt, den schlechtesten Fall, also die größtmögliche Parameterschwankung. Wenn die Problem-Dimension klein ist, sind verhältnismäßig wenige Punkte zu berechnen. Erhöht sich die Dimension aber, wird eine Berechenbarkeit so vieler Funktionswerte immer schwieriger.

Dieses Beispiel soll verdeutlichen, wie sehr sich durch die Veränderung der Dimension, der Rechenaufwand erhöht. Angenommen man

benötigt zur Berechnung eines Funktionswertes eine halbe Sekunde (Anmerkung: dies ist ein frei angenommener Wert). Die Rechenzeit für den zweidimensionalen Bereich würde somit $2^2 * 0{,}5[s] = 2\ [s]$ dauern. Bei einer Dimension von 30 würde man zur Berechnung der 1,07 Mrd. Funktionswerte bereits 17 Jahre benötigen. Hier kann man deutlich erkennen, dass es durchaus auf Rechnerleistung und Effektivität des verwendeten Algorithmus ankommt. Es sollte auch ein Kompromiss geschlossen werden, was die Dimensionalität des Problems betrifft, sprich welche Bedingungen wirklich optimiert werden sollen, denn eine Berechnung aller Wünsche wäre zwar theoretisch möglich, aber praktisch mit einem so hohen Zeitaufwand verbunden, dass das Ergebnis uninteressant wird.

Können Parameterschwankungen statistisch, also mit einer Wahrscheinlichkeitsverteilung, angesehen werden, sind andere Verfahren besser geeignet, [3]. Um diese Streuung im Modell abzubilden, wurden verschiedene Methoden in Betracht gezogen. [11] Dabei soll hier die Monte-Carlo-Verteilung und die Latin-Hypercube-Verteilung näher betrachtet werden.

3.3 Monte-Carlo Methode

Beim Monte-Carlo Verfahren wird der streuende Bereich mit einer Verteilung von Zufallszahlen simuliert. Um eine möglichst gleichmäßige Verteilung zu erhalten, ist ein größerer Stichprobenumfang nötig. Im Idealfall sind es unendlich viele Stichproben, damit der Fehler der Toleranzsimulation gegen null konvergiert. Ist die Stichprobengröße zu gering, kann eine Normalverteilung der Stichproben nur unzureichend nachgebildet werden. Bei der Monte-Carlo Methode ist die Stichprobengröße von der Dimension des Problems unabhängig. Eine Variante der Monte-Carlo Methode ist die Plane Monte-Carlo Methode, [21]. Dabei wird durch einen Zufallszahlengenerator eine unsystematische Streupunktverteilung erzeugt. Die Anzahl der streuenden Punkte soll hier $(Anzahl\ der\ zuf\ddot{a}lligen\ response - Gr\ddot{o}\ss en)^2$ sein. Dadurch bedarf es in größeren Parameterräumen sehr vieler Stichproben und hat den Nachteil, dass, wenn zu wenig Stichproben genommen werden, die statistischen Unsicherheiten zu groß werden. Jede Stichprobe stellt in weiterer Folge einen Punkt im variablen Bereich dar, für den ein Funktionswert berechnet werden muss. Ist zu jeder Stichprobe nun der Funktionswert gebildet worden, müssen diese Funktionswerte in Bezug gebracht werden. Dazu bieten sich Methoden aus der Statistik, wie die Bildung von Mittelwert,

Erwartungswert, Varianz und Standardabweichung an (siehe dazu Kapitel 4).

3.4 Latin-Hypercube-Verteilung

Die sehr große Anzahl an zu berechnenden Streupunkten aus der Monte-Carlo Methode führte dazu, dass über Möglichkeiten nachgedacht wurde, wie man eine gute Streuung der Parameter mit einer geringeren Anzahl an Stichproben erzeugen kann. Aus dieser Überlegung entstand die Latin-Hypercube- Verteilung. Hier wird die notwendige Anzahl an Stichproben aus $2(Anzahl\ der\ zufälligen\ response - Größen)$ ermittelt. Dabei sollte eine Mindestanzahl von 10 Stichproben eingehalten werden, [21]. Die Stichproben werden systematisch so erzeugt, dass die Eingangsgrößen unabhängig voneinander sind, die Variationsbreiten aber eingehalten werden.

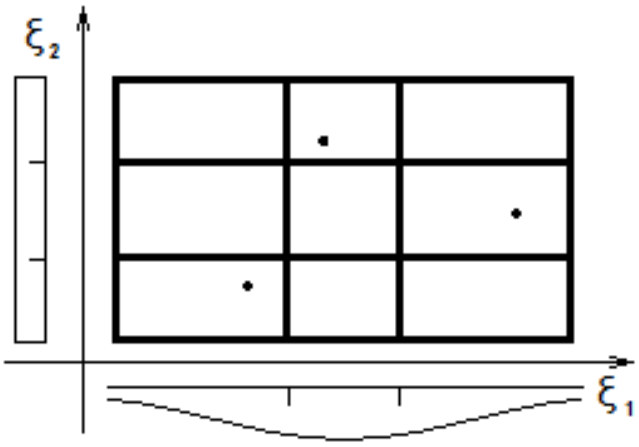

Abbildung 8 Latin Hypercube-Verteilung mit zwei Variablen und N=3, [17]

Laut [17] wird der gesamte Wertbereich des Störgrößenvektors $\xi = [\xi_1, \xi_2, \xi_3, \dots, \xi_n]$ in *N* Teilintervalle eingeteilt. Diese dürfen sich nicht schneiden und sollten alle eine gleiche Wahrscheinlichkeit von *1/N* besitzen. Gemäß der Wahrscheinlichkeitsdichte wird ein zufälliger Wert aus jedem Intervall gewählt. Die so erhaltenen *N* Werte für ξ_1 werden mit den *N* Werten aus ξ_2 kombiniert, siehe dazu Abbildung 8, und diese wiederum mit *N* Werten aus ξ_3 und so weiter, bis ein Set von *N n*-tuples gebildet ist, welches als ein Latin-Hypercube Sample bezeichnet wird. Mit dieser Methode erhält man mit wenigen Stichproben eine gute Verteilungsfunktion, und sie ist von der Dimension unabhängig.

Zum Schluss sei noch gesagt, dass den Vorteilen einer robusten Methode, wie der Unempfindlichkeit gegenüber Störungen oder die Berücksichtigung einer statistischen Abweichung im Prozess, ein immenser, rechnerischer Aufwand gegenüber steht und, wie im Zitat in Kapitel 1 schon vorweggenommen, die robuste Lösung im Großen und Ganzen keine wirkliche Optimierung mehr ist (wenn nicht gar schon das Gegenteil einer Optimierung ist).

Kapitel 4: Robuste Optimierungsverfahren

.

Um robuste Methoden in den Optimierungsalgorithmus implementieren zu können, sollen hier zwei Möglichkeiten vorgestellt werden. Zum einen die double loop- und zum anderen die single loop – Strategie.

Double loop-Strategie

Dabei wird zuerst optimiert, anschließend auf Robustheit geprüft. Wenn diese nicht ausreichend ist, soll erneut optimiert werden.

Single loop-Strategie

Schwankungen der Parameter sollen hier schon im Optimierungsprozess einfließen. Der Optimierungsalgorithmus verwendet hier bei der Suche nach dem Optimum schon die variierenden Parameter und liefert somit nach Beendigung des Algorithmus sofort ein robustes Optimum. Es entsteht dadurch allerdings auch ein höherer Rechenaufwand.

Die double loop Strategie arbeitet also zuerst einen Optimierungsalgorithmus ab und überprüft erst anschließend das gefundene Ergebnis auf Robustheit. Dabei kann es passieren, dass die Optimierungsschleife sehr oft durchlaufen werden muss, bis ein robustes Ergebnis gefunden wird. Ein weiterer Nachteil ist die Ineffizienz dieses Verfahrens, da die Funktionswerte immer berechnet werden müssen, obwohl sie zum Schluss nicht benötigt werden. Es geht somit wertvolle Rechenzeit verloren.

Bei der single loop Strategie wird die Robustheit schon im Algorithmus mitberücksichtigt. Hier entsteht während des Berechnens zwar ein hoher Rechenaufwand, dafür erhält man sofort ein robustes Ergebnis ohne unnotwendige Durchlaufzeiten, wie sie bei der double loop - Strategie vorhanden sind.

Im Folgenden, werden hier nur single loop - Strategien in Betracht gezogen.

Um die in Kapitel 3 vorgestellten Methoden in den Optimierungsalgorithmus einbauen zu können, werden die generierten Individuen des Optimierungsverfahrens in den robusten Algorithmus gebracht. Dort wird der Streubereich generiert. Bei der Worst-Case Methode wird ein klar abgegrenzter Bereich um den Punkt x gelegt, die Eckpunkte ermittelt, daraus die neuen Funktionswerte berechnet und aus diesen der schlechteste Funktionswert gesucht. Dieser Funktionswert wird wieder an den Optimierungsalgorithmus zurückgegeben, und dieser sucht dadurch das robuste Optimum. Wird statt der Worst-Case Methode die Monte-Carlo Methode angewendet, werden auch hier die Individuen an den

robusten Algorithmus übergeben und dort wird ein Streubereich um den Punkt x generiert. Zu jedem dieser Streupunkte wird der Funktionswert berechnet. Aus diesen Funktionswerten muss ein genereller Funktionswert gebildet werden, welcher anschließend wieder an den Optimierungsalgorithmus zurückgegeben werden kann. Im Folgenden sollen die Möglichkeiten besprochen werden, wie die Funktionswerte der Streupunkte in Beziehung gebracht werden können.

Abbildung 9 [13] zeigt hier den generellen Aufbau, um einen Algorithmus nach diesem Verfahren robust machen zu können. Dabei ist der erste Schritt die Initialisierung einer Population, welche aus N-Individuen besteht. Aus dieser Population wird ein Individuum $\mathbf{x}_i$ ausgewählt von dem der Funktionswert $f(\mathbf{x}_i)$ berechnet wird. Im nächsten Schritt wird eine Wahrscheinlichkeit, sprich eine Streuung, um den Funktionswert $f(\mathbf{x}_i)$ gebildet $P_k = P[f_k(\mathbf{x}_i) > 0]$. Die Wahrscheinlichkeits-Verteilung ist eine Gaußverteilung und soll konstant zwischen null und eins sein. Diese Wahrscheinlichkeit wird nun noch mit einem Faktor λ_k multipliziert und die Summe aller generierten Streufaktoren um den Punkt $\mathbf{x}_i$ werden zu einem Strafterm zusammengefasst und daraus über (4.1) der neue Funktionswert des Individuums gebildet. (4.1) ist an Verfahren aus der Statistik (Bildung von Mittelwert und Standardabweichung) angelehnt.

$$f_{ineu} = f(\mathbf{x}_i) + \sum_{k=1}^{m} \lambda_k P_k \qquad (4.1)$$

Nun wird noch überprüft, ob alle Individuen schon berechnet wurden. Wenn ja, startet die Bildung einer neuen Generation. Wenn nein, wird das nächste Individuum ausgewählt und der neue Funktionswert dieses Individuums berechnet. Dabei wird hier eine Schleife über alle Individuen gezogen (erste Schleife) und eine weitere wird über alle Generationen gezogen (zweite Schleife).

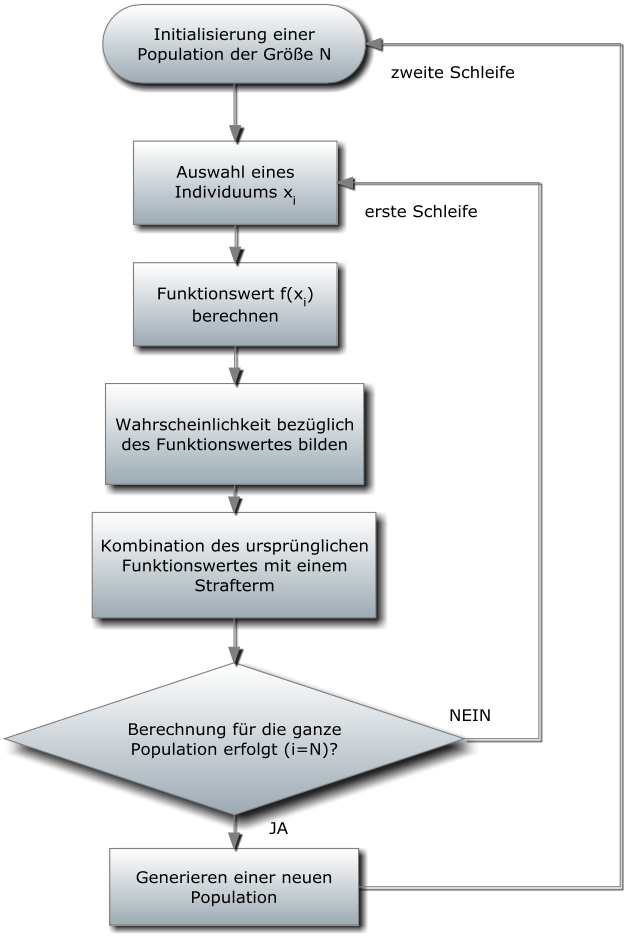

Abbildung 9 Optimierungsschleifen, [13]

Das deterministische, globale Minimum hat einen wesentlich schlechteren Durchschnitt über die gestreuten Störfunktionswerte sowie einen höheren Streubereich der Standardabweichung als das robuste Optimum. Der aus z.B. Durchschnitt und Standardabweichung gebildete Strafterm ist somit für das globale Optimum größer und hat die Auswirkung, dass der Gesamtfunktionswert für diese Optimum schlechter wird, als jener dessen Strafterm aufgrund geringerer Streuung der Funktionswerte kleiner ist, sprich das robuste Optimum, siehe (4.1).

Allgemein gesagt verwenden Methoden, die auf der Wahrscheinlichkeitstheorie basieren wie z.B. die Monte Carlo Methode (siehe Kap. 3.3) oder das Latin Hypercube Verfahren (siehe Kap. 3.4), zur Implementierung der Streuwerte in das robuste Verfahren die Bildung von Mittelwert, Erwartungswert, Varianz, und oder Standardabweichung.

$$Mittelwert: \quad \overline{f(\mathbf{x})} = \frac{\sum_{i=1}^{n} f(\mathbf{x}_i)}{n} \tag{4.2}$$

$$Erwartungswert: \quad E\big(f(\mathbf{x})\big) = \sum_{i=1}^{k} p_i f(\mathbf{x}_i) \tag{4.3}$$

$Varianz$:

$$V^2\big(f(\mathbf{x})\big) = \frac{\big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_1)\big)^2 + \big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_2)\big)^2 + \ldots + \big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_n)\big)^2}{n} \tag{4.4}$$

$Standardabweichung$:

$$S\big(f(\mathbf{x})\big) = \sqrt{V^2\big(f(\mathbf{x})\big)} = \sqrt{\frac{\big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_1)\big)^2 + \big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_2)\big)^2 + \ldots + \big(\overline{\mathbf{f(x)}} - f(\mathbf{x}_n)\big)^2}{n}} \tag{4.5}$$

Der Mittelwert aus (4.2) ist die Summe aller Funktionswerte $f(\mathbf{x}_i)$ über die Anzahl der Parameter n. Die Varianz aus (4.4) ergibt sich aus dem Quadrat der Differenz zwischen Mittelwert und jedem Parameter. Sie ist ein Maß für die Streuung der Variablen. In (4.5) ist die Standardabweichung dargestellt. Sie ist ein Maß für die Streuung um den Mittelwert eines zufälligen Parameters. Der Erwartungswert aus (4.3) ist die Summe aller Parameter, die mit der Wahrscheinlichkeit des Auftretens des Parameters multipliziert werden. Die Summe aller Wahrscheinlichkeiten ist eins. Der Erwartungswert entspricht dem Mittelwert einer unendlich großen Anzahl an Stichproben.

Um nun eine Bewertungsfunktion zu erhalten, damit die Optimierung weiter berechnet werden kann, verwendet [11] für den neuen Funktionswert die Kombination aus (4.6) mit Erwartungswert und Standardabweichung. Dabei kann der Strafterm zusätzlich noch gewichtet werden.

$$F(\mathbf{x}) = w_1 f(\mathbf{x}) + w_2 \left(E\big(f(\mathbf{x})\big) + \sqrt{Var\big(f(\mathbf{x})\big)} \right) \tag{4.6}$$

Der neue Funktionswert $F(\mathbf{x})$ ist nun eine Addition des ursprünglichen Funktionswertes $f(\mathbf{x})$ und dessen Erwartungswertes sowie der Standardabweichung. Man könnte den Einfluss der Funktion $f(\mathbf{x})$ und des „Strafterms“ $E\big(f(\mathbf{x})\big) + \sqrt{Var\big(f(\mathbf{x})\big)}$ auch noch über die beiden Gewichte ($w_1\ und\ w_2$) verändern.

Statt dieser Methode verwendet z.B. Sundareson (*et. al*. 1993) nur den Mittelwert und Yoon (*et. al.* 1999) den Mittelwert und die Abweichung (aus [10]).

Yaochu Jin und Bernhard Sendhoff verwenden in [12] zwei verschiedene Ansätze. Im ersten wird der Mittelwert über die Funktionswerte der in der Umgebung von $\mathbf{x}$ liegenden Punkte gebildet. Im zweiten Ansatz werden während der Berechnung des Funktionswertes stochastische Störeinflüsse mit einbezogen.

Im ersten Ansatz sollten mindestens zwei Punkte aus der Nachbarschaft ausgesucht werden. Die Wahl der Anzahl der verwendeten Punkte sollte nicht zu groß sein, da sich mit jedem Punkt automatisch der Rechenaufwand erhöht. In (4.7) wird der Funktionswert $f(\mathbf{a},\mathbf{x})$ aus der Summe der gewichteten Funktionswerte $f(\mathbf{a},\mathbf{x}+\Delta\mathbf{x}_i)$ gebildet. Dabei sind $\mathbf{a}$ Parameter der Umgebung, $\mathbf{x}$ sind design-Variablen und $\Delta\mathbf{x}_i$ ist ein Vektor von kleinen Zahlen, die zufällig oder deterministisch gewählt wurden. Die Gewichte w_i können für jede Evaluation unterschiedliche Werte annehmen. Im einfachsten Fall aber werden sie alle gleich eins gesetzt.

$$f(\mathbf{a},\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i f(\mathbf{a},\mathbf{x}+\Delta\mathbf{x_i})}{\sum_{i=1}^{N} w_i} \qquad (4.7)$$

Sind die Gewichte alle gleich eins, kann man die Gleichung auch umschreiben und erhält (4.8). Man kann erkennen, dass $f(\mathbf{a},\mathbf{x})$ der Mittelwert über alle Funktionswerte der Parameter und deren Distanz $\Delta\mathbf{x}_i$ zu den Funktionswerten ist.

$$f(\mathbf{a},\mathbf{x}) = \frac{\sum_{i=1}^{N} f(\mathbf{a},\mathbf{x}+\Delta\mathbf{x_i})}{N} \qquad (4.8)$$

Die zweite Möglichkeit, die hier [12] beschrieben wird um robuste Lösungen zu finden, ist die, Funktionswerte stochastischer Störeinflüsse während der Berechnung mit einzubeziehen. Siehe (4.9).

$$f\big(\mathbf{a},\mathbf{x}(t)\big) = f(\mathbf{a},\mathbf{x}(t)+\Delta\mathbf{x}(t)) \qquad (4.9)$$

Dabei ist hier t als der Generationenindex zu betrachten. Es wird zu jedem Funktionswert der Individuen jeder Generation ein Störeinfluss $\Delta\mathbf{x}(t)$, wiederum für jede Generation addiert. Dieser Störeinfluss wird durch einen Zufallsgenerator erzeugt. Wenn die Populationsgröße

unendlich groß ist, so ist diese Methode äquivalent zu der Methode, welche zur Ermittlung des Funktionswertes über die Störgrößen, statt dem Mittelwert, den Erwartungswert verwendet.

Dieser Artikel [12] beschreibt in weiterer Folge den Kompromiss, der zwischen dem Finden eines robusten Optimums und dem schlechter werdenden Ergebnis des Funktionswertes zu schließen ist. Entweder möchte man eine optimale Lösung, sprich das globale Minimum finden, oder man möchte eine robuste Lösung, wobei hier wiederum gilt: je robuster die Lösung ist, desto weiter ist sie vom Optimum entfernt.

Zusammenfassend kann am Ende dieses Kapitels gesagt werden, dass man zwischen double loop und single loop Strategie unterscheiden kann, wobei hier nur die single loop Strategie, welche die Veränderungsmöglichkeit der Parameter gleich im Algorithmus mitberücksichtigt, behandelt wurde. Bei der Worst-Case Methode wird dem neuen Funktionswert der schlechteste Funktionswert der ihn umgebenden Streuparameter direkt zugewiesen. Dabei werden von allen Eckpunkten die Funktionswerte miteinander verglichen und der schlechteste bestimmt. Im Gegensatz zu den Verfahren, die auf statistischen Schwankungen beruhen, werden hier Funktionswerte aller Stichproben gebildet, die in weiterer Folge in einen Strafterm fließen und so in der Gesamtheit den neuen Funktionswert bilden. Voreilig könnte man nun behaupten, dass damit die Worst-Case Methode eine einfachere und schnellere Form der Implementierung robuster Methoden bietet. Dies würde für Problemstellungen geringerer Dimension durchaus zutreffen. Steigt die Dimension, so verliert diese Methode stark an Ausführbarkeit.

Dazu als Beispiel folgende Rechnung. Die Monte Carlo Methode benötigt z.B. mehr als 1 Mio. Stichproben, um den Toleranzgrößenfehler gering zu halten. Da dieses Verfahren weitgehend unabhängig von der Dimension ist, bleibt die Anzahl der Berechnungen gleich. Die Worst-Case Methode benötigt 2^n Punkte, die die Eckpunkte darstellen. Ist die Dimension nun z.B. Dim.=3 so benötigt die Monte-Carlo Variante 1 Mio. Stichproben und die Worst Case Methode $2^3 = 8$ Eckpunkte. Ab einer Dimension von 20 kehrt sich die Situation um. Hier benötigt die Monte-Carlo Methode wiederum 1 Mio. Stichproben, aber die Worst-Case Methode 1,05 Mio. zu berechnende Eckpunkte. Als Vergleich soll hier noch erwähnt sein, dass z.B. in der Finanzwirtschaft typischerweise Dimensionen von 365 vorliegen, [16].

Kapitel 5: Anwendungsbeispiel und Ergebnisse

5.1 Aufgabenstellung:

Es soll eine Differentielle Evolutionsstrategie (DE, siehe dazu Kap. 2.5) implementiert und anhand einer Testfunktion evaluiert werden. Die DE soll zu einer robusten Optimierungsstrategie adaptiert werden, um Parametertoleranzen entsprechend zu berücksichtigen. Anhand einer Testfunktion, welche eigens dafür konstruiert wurde, soll gezeigt werden, wie sich das Ergebnis durch die Erweiterung mit robusten Methoden verändert.

5.2 Differentielle Evolutionsstrategie:

Die Populationsgröße der Differentiellen Evolutionsstrategie wurde mit 30 gewählt. Eine zu geringe Populationsgröße würde den Parameterraum nicht ausreichend ausfüllen, eine zu große Population kostet in jeder Generation viel Berechnungszeit. Für die Mutation wurde $\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r2} - \mathbf{x}_{r3})$ aus Kapitel 2.5.1 gewählt. Rekombination und Selektion siehe Kapitel 2.5.2 (2.14) und Kapitel 2.5.3 (2.15). Für den Skalierungsfaktor F wurde der Wert 0,8 und für den Crossover-Faktor c_r aus (2.14) der Wert 0,5 angenommen. Letzterer bedeutet, dass bei der Rekombination eine 50/50 Chance besteht, mutierte Vektorelemente $\mathbf{v}_{i,j}$ mit bestehenden Individuen-Elementen $\mathbf{x}_{i,j}$ zu mischen, [5].

Nach dem Rekombinationsprozess werden die neuen Individuen auf ihre Plausibilität hin geprüft. Liegt ein Individuum außerhalb der festgelegten Parametergrenzen, muss es entweder verworfen oder durch zusätzliche Veränderung wieder in den Suchbereich zurückgebracht werden. Dafür gibt es drei Verfahren:

1. Loop-around: Hier wird das Individuum auf der anderen Seite des Suchbereichs wieder eingefügt. Überschreitet es also z.B. die obere Grenze um den Wert a (entspricht $ub + a$), wird dieser Wert nun zur unteren Schranke hinzuaddiert, also das Individuum auf $lb + a$ versetzt.

2. Spiegeln: Dabei wird das Individuum in den Suchbereich zurückgespiegelt, also aus $ub + a$ wird $ub - a$.

3. Zufalls-Verfahren (random): Das Individuum wird zufällig in den Suchbereich zurückgeworfen.

Eine graphische Darstellung dieser drei Möglichkeiten ist in Abbildung 10 gezeigt.

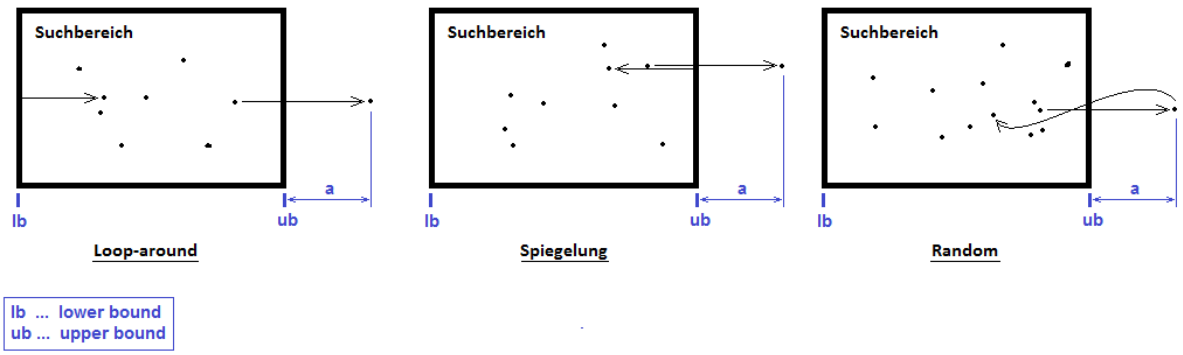

Abbildung 10 Verfahren, um Individuen in den Suchbereich zurückzubringen

Im letzten Schritt einer Generation, nämlich der Selektion, werden die Funktionswerte der neu generierten Individuen $\mathbf{u}$ (2.15) gebildet und mit den Funktionswerten der ursprünglichen Individuen $\mathbf{x}$ verglichen. Jene 30 (≙ der Populationsgröße) Individuen mit den besten Funktionswerten werden in die nächste Generation übernommen. Das beste Individuum $\mathbf{x}_{best}$ wird in der nächsten Generation als Mutationsbasis in (2.9) verwendet.

Bevor auf die Robustheit der DE eingegangen wird, wird vorerst das globale Verhalten der Differentiellen Evolutionsstrategie überprüft. Als Testfunktion wurde eine zweidimensionale Funktion entwickelt, die eine Kombination aus 3 Gaußverteilungen darstellt. Damit lassen sich ein globales und zwei lokale Minima mit unterschiedlichen Varianzen, also unterschiedlichen Steigungen, realisieren. Die Koordinaten der Minima sowie die entsprechenden Funktionswerte sind in Tabelle 1 gegeben. Der Suchraum wurde zwischen den Grenzen [-2, 2] in beiden Dimensionen beschränkt. Die beschriebene Testfunktion „Skrabl“ ist in Abbildung 11 dargestellt.

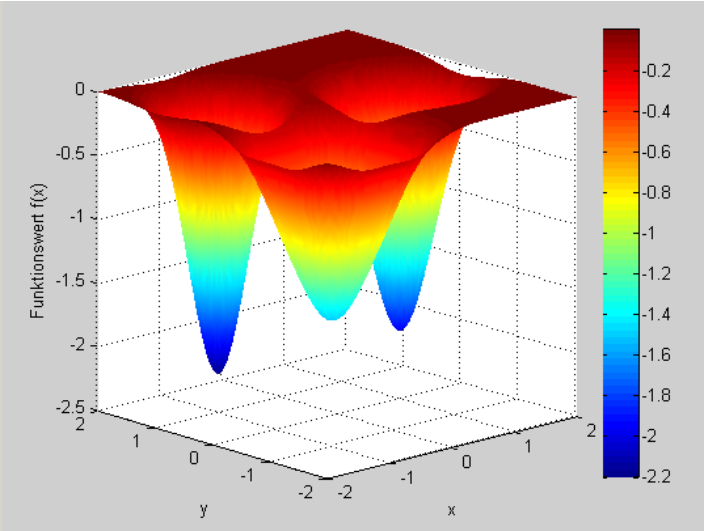

Abbildung 11 Testfunktion „Skrabl“

	Position		Funktionswert
	Parameter 1	Parameter 2	f(x)
Globales Optimum	-1	1	-2,200
1. lokales Optimum	1	0	-1,998
2. lokales Optimum	-1	-1	-1,500

Tabelle 1: Globales Optimum und lokale Optima der Testfunktion „Skrabl“

Um das Verhalten der Differentiellen Evolutionsstrategie zu überprüfen, wurden 500 Runs, also voneinander unabhängige Durchläufe des Optimierungsverfahrens, durchgeführt (Tabelle der Ergebnisse siehe Anhang 1). Es konnte festgestellt werden, dass die Differentielle Evolutionsstrategie zu 70,6% das globale Optimum findet. Zu 27,4% wurde als Lösung das 1. lokale Optimum gefunden und zu 2% das 2. lokale Optimum. Abbildung 12 zeigt exemplarisch die Konvergenz dieses Algorithmus, wie sie aus Tabelle 1 hervorgeht. Im oberen Bereich ist der Funktionswert dargestellt. Man erkennt, dass der Algorithmus zum globalen Optimum bei -2,2 nach 11 Iterationen konvergiert. Der untere Bereich zeigt die Position des gefundenen Optimums an. Dabei stellen die blauen Punkte die Position des ersten Parameters dar und die grünen Punkte die Position des zweiten Parameters.

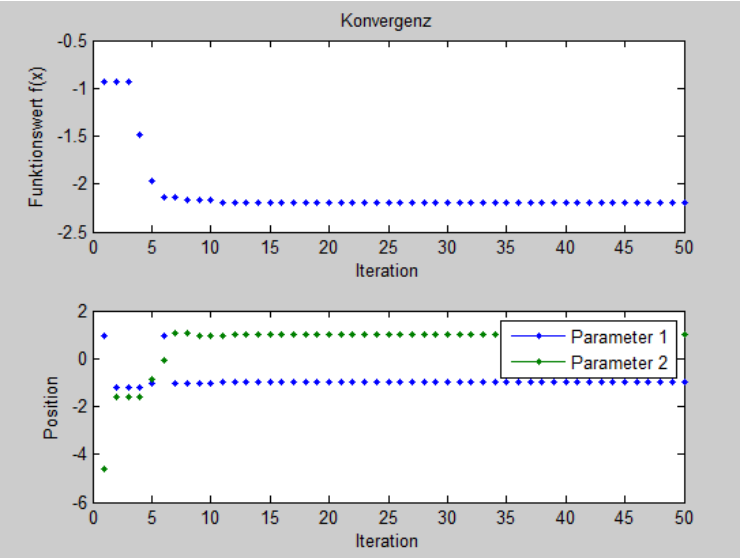

Abbildung 12 Konvergenzverhalten des DE-Algorithmus (ohne robuste Methode)

Das globale und das 1. lokale Optimum sind vom Funktionswert her nicht sehr weit auseinander, dennoch findet der Algorithmus in mehr als 70% der Fälle das globale Optimum. Man kann also durchaus sagen, dass der Algorithmus globales Verhalten aufweist und für diese Testzwecke geeignet ist.

5.3 Robuste Methode:

Es sollen nun robuste Methoden zur Anwendung kommen. Diese wurden nach der single loop-Strategie in den bestehenden Algorithmus der Differentiellen Evolutionsstrategie implementiert. Die in dieser Arbeit behandelten Verfahren berücksichtigen Parameterschwankungen, die von Toleranzen der Eingangsgrößen herrühren. Das robuste Verfahren muss also zwischen der Individuen-Bildung und der entsprechenden Funktionswertberechnung eingreifen. Es wird also nicht mehr nur das Individuum selbst betrachtet, sondern auch die Umgebung im Parameterraum. Das bedeutet, zu jedem Individuum werden mehrere Vektoren mit einer bestimmten Schwankungsbreite (mit positiven oder negativen Werten) hinzugefügt. Für jedes dieser neu entstandenen Umgebungsindividuen wird der entsprechende Funktionswert berechnet. Je nach verwendeter robuster Methode wird nur der schlechteste Funktionswert gewählt (Worst-Case Methode), oder es werden alle Funktionswerte zur Berechnung des neuen, robusten Funktionswertes herangezogen. Der restliche Teil der DE bleibt völlig gleich. Wie später gezeigt wird, ändert sich durch die Erweiterung mit der robusten Methode die Zielfunktion, da sie nun nicht mehr nur die Funktion von $\mathbf{x}$ selbst, sondern, je nach angewendetem Verfahren, z.B. ein neuer Funktionswert (Worst-Case)

oder z.B. eine Kombination aus den Umgebungsfunktionswerten von $\mathbf{x}$ (Monte-Carlo) ist.

Um die Parameterschwankungen in den bestehenden Algorithmus der Differentiellen Evolutionsstrategie implementieren zu können, wurde für die beiden gewählten Verfahren (Worst-Case und Monte-Carlo) folgendes Schema überlegt:

5.3.1 Worst-Case Verfahren

Im Wort-Case Verfahren werden ausgehend vom Punkt $\mathbf{x}$ die Parameterdimensionen so kombiniert, dass man die Extrempunkte erhält. Würde man hierfür den zweidimensionalen Raum betrachten, wären die Extrempunkte die Eckpunkte eines Rechteckes. Wählt man den Toleranzbereich um einen Punkt $\mathbf{x}$ in beiden Richtungen gleich groß, ergäbe sich ein Quadrat. Für den dreidimensionalen Raum ergäbe sich für unterschiedliche Toleranzbereiche ein Quader bzw. für gleich große Toleranzbereiche ein Würfel. Abbildung 13 zeigt die Struktur eines dreidimensionalen Problems mit den maximalen Toleranzen ±Δ. Es wurde nun um jedes Individuum im robusten Abschnitt dieses neue „Koordinatensystem“ angelegt, mit dem Toleranzbereich Δ multipliziert, und damit konnten die Eckpunkte generiert werden.

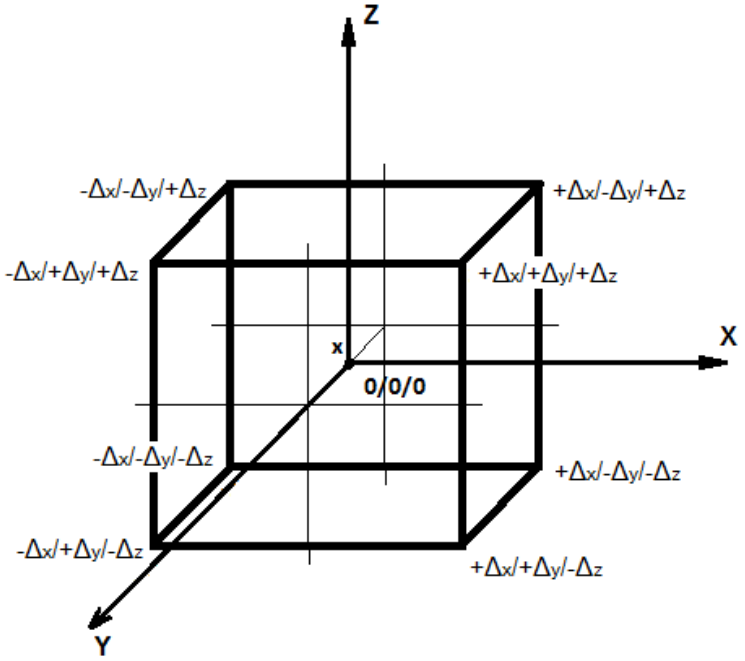

Abbildung 13 Koordinaten der verteilten Eckpunkte um einen Parameter in 3D

Der Toleranzbereich um jeden Parameter wurde in alle Richtungen gleich groß angenommen. Da in diesem Beispiel nur der zweidimensionale Raum betrachtet wird, entsteht eine quadratische Fläche. Es wurden nun, wie oben beschrieben, die Eckpunkte dieser

Fläche bestimmt und daraus der schlechteste Funktionswert ermittelt, mit welchem die Differentielle Evolutionsstrategie weiterarbeitet, um so das robuste Optimum zu finden. Dabei ist der Toleranzbereich als nicht veränderbare Grenze zu betrachten. In Abbildung 14 kann man die Worst-Case Methode auf der Testfunktion „Skrabl“ angewandt sehen. Man erkennt, dass die „Täler“ viel spitzer zulaufen und die Funktionswerte angehoben wurden.

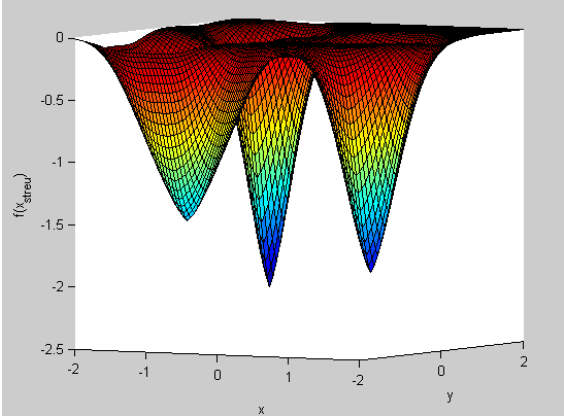

Abbildung 14 Worst-Case Methode

5.3.2 Monte-Carlo Verfahren

Beim Monte-Carlo Verfahren werden wieder die Individuen in die robuste Unterfunktion geführt, und dort wird für jedes eine Streufunktion generiert. Es sei angemerkt, dass es sich um eine robuste Unterfunktion handelt, und keine Änderungen an dem Algorithmus der Differentiellen Evolutionsstrategie vorgenommen werden müssen. Um eine Streuung erzeugen zu können, wurde im Programm eine Funktion aufgerufen, die eine Normalverteilung generiert. Abbildung 15 zeigt eine Normalverteilung, wie sie für die Monte-Carlo Methode verwendet wurde.

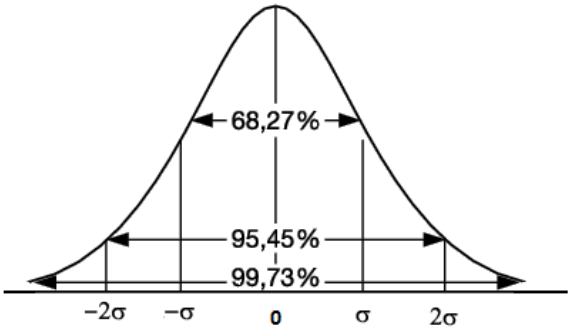

Abbildung 15 Normalverteilung [19]

Zuerst wird zu jedem Parameter eine normalverteilte Stichprobenmenge addiert. Von jedem daraus neu entstandenen Punkt

wird der Funktionswert berechnet. Aus den Funktionswerten wird anschließend der Mittelwert und die Standardabweichung gebildet und nach (5.1) zum ursprünglichen Funktionswert addiert.

$$F(\mathbf{x}) = w_1 f(\mathbf{x}) + w_2 \left(\overline{f(\mathbf{x}_{streu})} + \sqrt{Var\left(f(\mathbf{x}_{streu})\right)} \right) \quad (5.1)$$

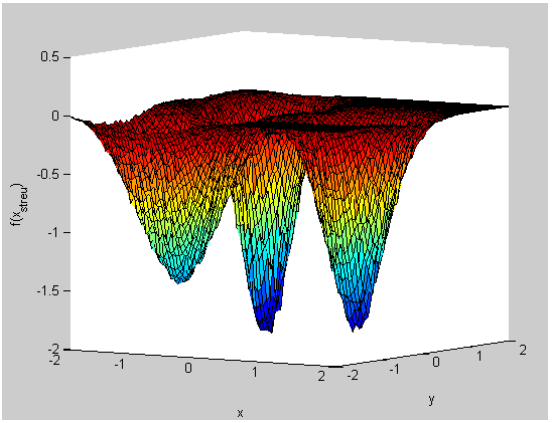

Abbildung 16 Monte-Carlo Methode

In Abbildung 16 kann man die Monte-Carlo Methode auf der Testfunktion „Skrabl“ angewandt sehen. Man erkennt, dass im Gegensatz zur Worst-Case Methode die „Täler“ nicht mehr so spitz zulaufen, und die Oberfläche aufgrund des Einflusses der Streuung nicht mehr glatt ist. Durch den Einfluss der Monte-Carlo Methode ist aber auch zu sehen, dass sich der Kegelradius nicht wesentlich verkleinert (im Gegensatz zur Worst-Case Methode).

Der in der jeweiligen Unterfunktion berechnete robuste Zielfunktionswert $F(\mathbf{x})$ wird an den Algorithmus der DE zurückgegeben, welcher mit diesem neuen Funktionswert, wie in Kapitel 2.5 besprochen, optimiert.

5.3.3 Anwendung der robusten Methoden

Vorerst wurde die Auswirkung der Implementierung der robusten Methode (zuerst der Worst-Case Methode) auf den Algorithmus betrachtet. Dazu wurden der Toleranzbereich auf null gesetzt und 100 Runs mit den gleichen Einstellungen, wie zu Beginn bei der Prüfung des Verhaltens der DE ohne Implementierung der robusten Methode, durchgeführt. Dabei sollte der Algorithmus dasselbe Ergebnis zeigen wie bei der reinen Differentiellen Evolutionsstrategie. Die Auswertung der Ergebnisse, siehe Anhang 2, ergab auch ein ähnliches Bild (zu 75% wurde das globale Optimum gefunden).

Anschließend wurde der Toleranzbereich der Parameter stark erhöht, um zu sehen, wie sich das Testbeispiel verhält. In Abbildung 17 ist das Ergebnis bei einer Parameterschwankung von 30% des gesamten Parameterbereichs

anhand der Testfunktion „Skrabl“ zu sehen. Man erkennt, dass sich die Minima verändern. Die Kegel des globalen Minimums sowie der lokalen Minima beginnen kleiner zu werden. Das globale Minimum schrumpft vom Funktionswert her von -2,2 auf ca. -1,2. Durch den Einfluss der robusten Methode interagieren die einzelnen Minima aber auch miteinander, und es entstehen neue Minima. Um eine eindeutige Trennung der Minima zu bekommen, wurden der Parameterbereich erweitert und der Abstand der Kegel zueinander vergrößert.

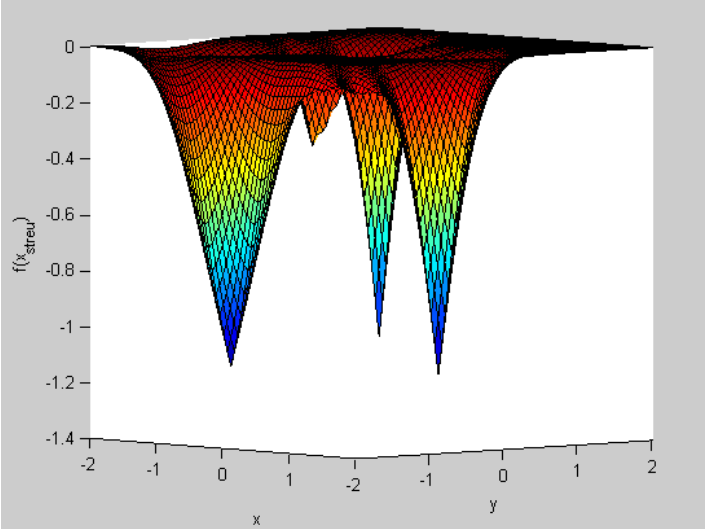

Abbildung 17 Testbeispiel "Skrabl" mit Toleranzbereich von 30% des Parameterbereichs

Abbildung 18 zeigt die Testfunktion „Skrabl2“ im erweiterten Suchbereich und mit einer größeren Distanz zwischen den einzelnen Optima.

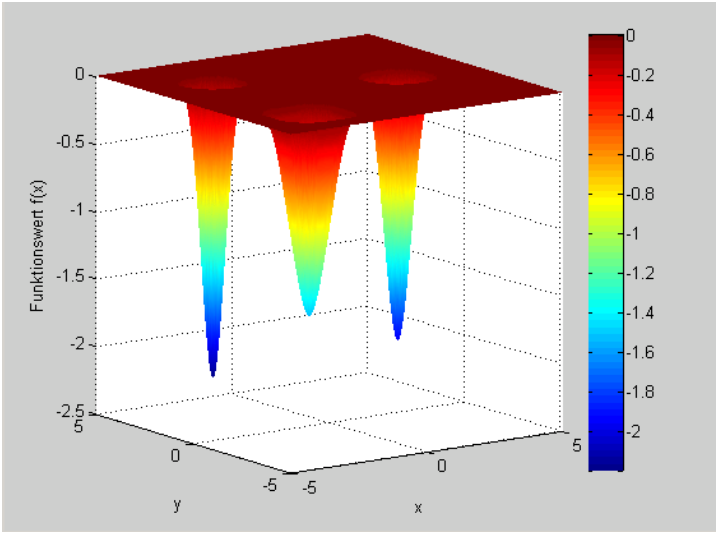

Abbildung 18 Testfunktion "Skrabl2" mit erweiterten Grenzen

Es wurde wiederum der Toleranzbereich der Parameter stark erhöht. Man erkennt in Abbildung 19, dass sich auch hier die Funktionswerte zwar verändert haben, aber keine neuen Minima entstanden sind. Der Abstand ist somit zu groß, sodass die unterschiedlichen Minima bei einer vorhandenen

Variabilität nicht mehr interagieren können. Die Positionen der Minima bleiben gleich, und die Funktionswerte verkleinern sich. Man kann erneut erkennen, dass die Kegel schmäler werden. Da auch hier wieder 30% des Parameterbereichs für den Toleranzbereich angenommen wurde, und es sich in diesem Fall um den erweiterten Parameterbereich handelt, ist der Einfluss der robusten Methode (siehe Abbildung 19) wesentlich stärker zu sehen. Die Kegel werden immer schmäler und die Funktionswerte immer kleiner. Dabei ist hier der längste Kegel auf der linken Seite der Kegel des 2. lokalen Optimums. Der mittlere Kegel, welcher den geringsten Funktionswert besitzt, stellt das globale Optimum dar. Wie gesagt, es wurde ein sehr großer Toleranzbereich gewählt, um zu sehen, wie sich die Testfunktion unter Extrembedingungen verhält.

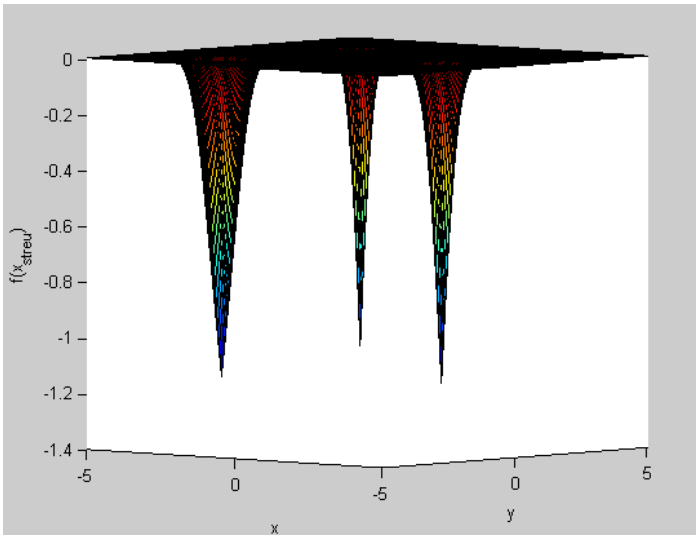

Abbildung 19 Testbeispiel "Skrabl2" mit einem Toleranzbereich von 30% des Parameterbereichs (Worst-Case)

Erneut wurde das Optimum der neuen Testfunktion durch die Differentielle Evolutionsstrategie gesucht. In diesem Fall wurde die Anzahl der Runs auf 250 erhöht. Für die Auswertung wurden die gefundenen Optima in Bezug zu den Runs gesetzt. Da die Runs voneinander unabhängig sind, kann bei einer größeren Anzahl an Runs eine bessere Aussage über das Gesamtverhalten des Algorithmus getroffen werden. Die Populationsgröße von 30 Individuen, 50 Generationen, einem Skalierungsfaktor von 0,8 und einem Crossover-Faktor von 0,5 wurden beibehalten. Die Auswertung der Ergebnisse befindet sich in der Tabelle im Anhang 3. Die Differentielle Evolutionsstrategie mit robuster Worst-Case Methode fand das globale Optimum nun nur mehr zu 51%, das 1. lokale Optimum wurde zu 34% als Optimum erkannt, und das 2. lokale Optimum wurde zu 15% gefunden. Betrachtet man die Funktionswerte des globalen Optimums und des 1. lokalen Optimums, zeigt sich, dass die Differenz der beiden nicht sehr groß ist. Dadurch hat die robuste Methode mit dem gewählten Toleranzbereich

keinen vollen Einfluss auf das Finden des robusten Optimums. Dieser kann durch Veränderung des Toleranzbereichs und/oder der Steilheit der Zielfunktion (z.B. durch Vergrößern der Differenz der beiden Optima) verbessert werden.

Einfluss der DE - Steuerparameter

Es soll der Einfluss der Steuerparameter der Differentiellen Evolutionsstrategie auf das Finden des globalen Optimums untersucht werden. Diese Einflüsse sollen vorerst unabhängig von den robusten Methoden betrachtet werden. Allerdings wurde zuerst der Funktionswert des globalen Optimums vergrößert, um in den folgenden Versuchen, in welchen die robusten Methoden aktiv wurden, den Einfluss der robusten Verfahren zu verbessern. Für den DE-Algorithmus wurden Skalierungsfaktor und Crossover-Faktor unterschiedlich verändert. Die Anzahl der Individuen betrug 30 bei 100 Generationen. Für diese Versuchsreihe wurden jeweils 250 Runs durchgeführt. Siehe dazu die Auswertungen der Tabellen im Anhang 4 bis 9. Man erkennt hier schon (siehe Auswertung im Anhang 4), dass sich der Prozentsatz des Findens des globalen Optimums schon durch die Vergrößerung der Distanz der Funktionswerte zwischen globalem und lokalem Optimum stark verbessert hat.

Der Skalierungsfaktor wurde einmal auf 1,5 und einmal auf 0,4 gesetzt. Dieser Faktor legt die Gewichtung auf die Diversität. Die Einwirkung des Anteils der Differenz zweier zufällig gewählter Individuen wird dadurch erhöht bzw. verringert. Es konnte festgestellt werden, dass bei einem Skalierungsfaktor von 1,5 das globale Optimum zu knapp 79% gefunden werden konnte (Vergleich mit der ursprünglichen Konstellation, bei der das globale Optimum nur zu knapp 74% gefunden wurde). Eine Verkleinerung dieses Faktors auf 0,4 ergab ein Finden des globalen Optimums zu nur mehr ca. 47%. Der Skalierungsfaktor hat somit eine entscheidende Auswirkung auf die Effektivität der Differentiellen Evolutionsstrategie.

Beim Crossover-Faktor verhielt es sich ähnlich. Dieser Faktor bestimmt die Wahrscheinlichkeit, mit der entweder die ursprüngliche Eigenschaft des Individuums oder die Eigenschaft des mutierten Individuums in den neuen Vektor aufgenommen wird. Dabei gilt laut (2.14): je größer der Crossover-Faktor gewählt wird, desto eher wird die mutierte Eigenschaft in den neu generierten Vektor aufgenommen. Da dieser nur zwischen 0 und 1 angenommen werden kann (1 würde in diesem Fall bedeuten, dass zu 100% das mutierte Individuum als neuer Vektor genommen wird, während ein Crossover-Faktor von 0 das mutierte Individuum vollkommen außer Acht lässt), wurde er einmal auf 0,8 und einmal auf 0,3 gesetzt (Die Ergebnisse der Auswertungen befinden sich im Anhang 7 und 8.). Es konnte festgestellt

werden, dass bei einem Crossover-Faktor von 0,8 (zu 80% werden die Eigenschaften des mutierten Individuums übernommen) das globale Optimum beinahe zu 64% gefunden wurde. Der Crossover-Faktor wurde anschließend auf 0,3 herabgesetzt, und es konnte festgestellt werden, dass das Finden des globalen Optimums auf 74% gesteigert werden konnte.

Als Folge aus den vorangegangenen Erkenntnissen wurde noch eine Versuchsreihe durchgeführt. Der Skalierungsfaktor wurde auf 1,5 gesetzt und der Crossover-Faktor auf 0,3. Somit entstand eine Kombination zwischen den beiden besseren Resultaten. Die Population sowie die Anzahl der Iterationsschritte blieben gleich groß. Nach 250 Runs wurden die Ergebnisse ausgewertet (ausgewertete Tabelle der Ergebnisse siehe Anhang 9). Es wurde zu 80% das globale Optimum gefunden, wobei der Funktionswert hier nicht mehr immer das absolute Minimum wiederspiegelt, sondern in einigen Fällen (in 45 Runs von 250 Runs) eine Abweichung von ca. 1% aufweist.

Einfluss des Toleranzbereiches im Worst-Case Verfahren

Nachdem die DE den neuen Erkenntnissen angepasst wurde, wurde für die Worst-Case Methode ein variabler Bereich von anfänglich 10% des Suchbereichs und anschließend 1% des Suchbereichs gewählt. Nach 250 Runs wurde folgendes Ergebnis gewonnen. Die dazugehörende Auswertung der Runs siehe Anhang 10 und Anhang 11. Für den ersten Fall findet der robuste Algorithmus zu knapp 71% das 2. lokale Minimum. Das globale Minimum wurde zu ca. 1% gefunden und das 1. lokale Minimum zu 28%.

Da ein robuster Algorithmus theoretisch das globale Optimum überspringen sollte, um ein robustes Optimum zu finden, kann man sagen, dass der Algorithmus für diesen Fall gut funktioniert. Da der variable Bereich sehr groß gewählt wurde, überspringt der Algorithmus auch das 1. lokale Optimum und findet hauptsächlich das 2. lokale Optimum. Hier wäre die Lösung auch am robustesten, denn dieses Optimum ähnelt einer kleinen Plattform.

Man kann hier schon erkennen, dass die Wahl der Testfunktion nicht unerheblich ist. Wenn die „Täler“ der Testfunktion nämlich alle gleich schmal sind, ist es schwierig bzw. unmöglich, für den Algorithmus ein robustes Optimum zu finden. Es wurde nun der Algorithmus mit einem kleineren variablen Bereich getestet. Dazu wurde 1% des Suchbereichs gewählt. Der Algorithmus fand zu 28% das globale Minimum, zu ca. 29% das 1. lokale Minimum und zu beinahe 43% das 2. lokale Minimum. Aus diesem Ergebnis kann abgeleitet werden, dass der Algorithmus zwar hauptsächlich das robusteste Optimum findet, aber dieses nicht mehr so attraktiv ist. Das globale und das 1. lokale Optimum werden durch die robuste Methode zu zwei ähnlich spitzen Kegeln. Damit findet der Algorithmus fast gleich oft das globale wie auch das 1. lokale Optimum.

Einfluss unterschiedlicher Gewichtungen im Monte-Carlo Verfahren

Für das Monte-Carlo Verfahren wurde der robuste Algorithmus nach dem in 5.3.2 beschriebenen Schema aufgebaut. Der Funktionswert wurde nach (5.1) ermittelt.

Dabei wurden hier vorerst das Gewicht w_1 auf 1 und der Strafterm auf 0,5 gesetzt. Das bedeutet, dass die Streuung, der Strafterm also, nur zu 50% in das Ergebnis einfließt, mit welchem die Differentielle Evolutionsstrategie weiterrechnet.

Anhand der Ergebnisse (siehe Abbildung 21) zeigt sich allerdings, dass man mit dieser Methode sogar bessere Funktionswerte erzielt, welche die ursprüngliche Testfunktion überhaupt nicht besitzt.

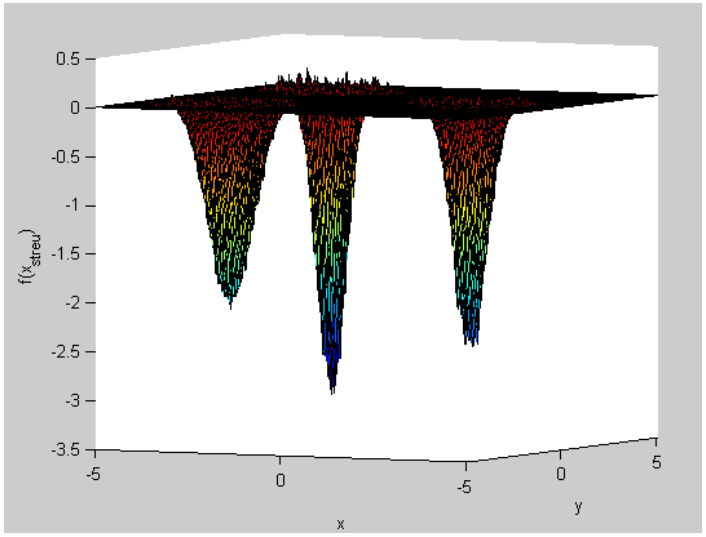

Abbildung 20 Monte-Carlo Methode

In [20] wurde zur Berechnung des Strafterms folgende Gewichtung bzw. Formel eingesetzt:

$$F(\mathbf{x}) = 0{,}5\, f(\mathbf{x}) + 0{,}5\left(E(f(\mathbf{x})) + \sqrt{Var(f(\mathbf{x}))}\right) \qquad (5.2)$$

Es stellte sich die Frage, warum die Gewichtung jeweils auf 1/2 gesetzt wurde. Diese kann damit beantwortet werden, dass, wenn die Summe der Gewichte $w_1\, und\, w_2 \neq 1$ ist, der Funktionswert nicht mehr stimmt. Es wurde nun die Berechnung mit (5.2) erneut durchgeführt. Statt des Erwartungswertes wurde der Mittelwert verwendet. Abbildung 20 zeigt das Ergebnis der neuen Berechnung.

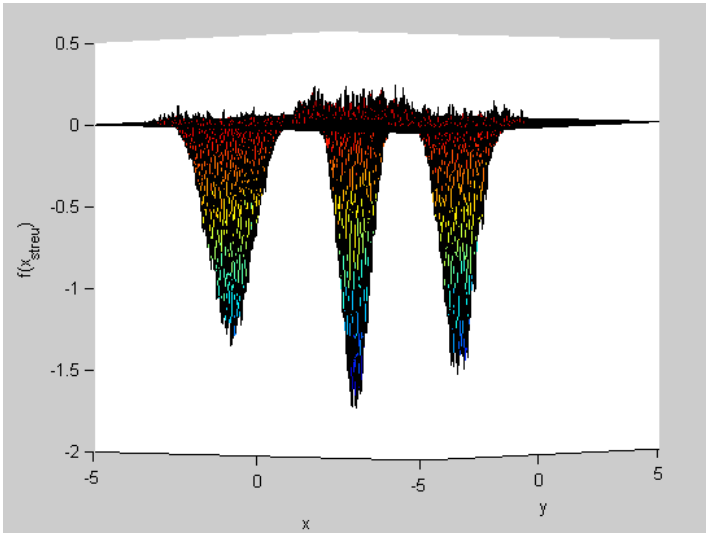

Abbildung 21 Monte-Carlo Methode an Testfunktion "Skrabl3“ mit gleicher Gewichtung aus (5.2)

Man erkennt, dass der Funktionswert im Gegensatz zu der Worst-Case Methode hier durchaus auch positive Werte erreichen kann. Die Standardabweichung ist eine quadratische Funktion, welche somit nur positiv sein kann. Damit muss der Gesamtstreufunktionswert aus (5.1) für die Testfunktion „Skrabl3“ nicht zwangsläufig negativ oder null sein.

Nun wurde die Auswirkung unterschiedlicher Gewichtungen bei einer Stichprobengröße von 100 Stichproben und einem σ-Wert von 0,5 (welcher mit den normalverteilten Stichproben multipliziert wird) betrachtet. Es wurden jeweils 250 Runs durchgeführt (Auswertung der Ergebnisse siehe Anhang 12-14). Es soll hier auch noch angemerkt sein, dass die Berechnung der 100 Stichproben auf 250 Runs mehr als eine halbe Stunde gedauert hat.

Im ersten Fall wurden die Gewichte aus (5.1) auf $w_1 = 0{,}25\ und\ w_2 = 0{,}75$ gesetzt. Daraus ergab sich, dass der robuste Algorithmus mit der Monte-Carlo Methode zu 35% das globale Optimum als robustes Ergebnis lieferte, zu fast 29% das 1. lokale Optimum und zu 36% das 2. lokale Optimum fand. Verändern sich die Gewichte auf $w_1 = 0{,}5\ und\ w_2 = 0{,}5$, wohingegen die anderen Einstellungen gleich geblieben sind, so lieferte der Algorithmus zu 64% das globale Optimum als robustes Ergebnis, zu 28% das 1. lokale Optimum und zu fast 8% das 2. lokale Optimum. Die Gewichte wurden erneut verändert. Bei einer Gewichtung von $w_1 = 0{,}75\ und\ w_2 = 0{,}25$ war der Unterschied zum Ergebnis der vorigen Gewichtung gering. Der Algorithmus fand das globale Optimum zu beinahe 71%, das 1. lokale Optimum zu fast 26%, und das 2. lokale Optimum wurde zu fast 4% gefunden.

Aus diesen Ergebnissen kann die Folgerung getroffen werden, dass die Wahl der Gewichtung eine entscheidende Rolle bei der Ermittlung einer robusten Lösung spielt. Wird die Gewichtung stärker auf den Strafterm gelegt, konvergiert der Algorithmus zwar eher in ein robusteres Optimum, aber nicht

mit einer eindeutigen Mehrheit. Wird die Gewichtung verlagert, ist das Ergebnis eindeutiger, sprich es wird mit einer Mehrheit das globale Optimum gefunden. Nach der Theorie wäre dies aber nicht das robuste Optimum. Das könnte allerdings auch bedeuten, dass der σ-Wert zu klein gewählt wurde.

Der σ-Wert wurde auf 1 gesetzt, und es wurden jeweils 100 Runs bei einer Stichprobengröße von 50 und den unterschiedlichen Gewichtungen durchgeführt. Die ausgewerteten Ergebnisse befinden sich im Anhang 15-17. Auch hier ergibt sich ein ähnliches Bild wie bei einem σ-Wert von 0,5. Bei einer Gewichtung von $w_1 = 0{,}25\ und\ w_2 = 0{,}75$ findet der Algorithmus zu 55% das globale Optimum und zu 35% das 1. lokale Optimum. Diese Verteilung hat sich also verbessert.

Vergleich Worst-Case und Monte-Carlo

Zum Schluss soll noch die Worst-Case Methode mit der Monte-Carlo Methode verglichen werden. Betrachtet man sich die Abbildungen 14 und 16, kann man schon erkennen, dass durch das unterschiedliche Verhalten der beiden Methoden ein 1:1 Vergleich nicht möglich ist. Die Worst-Case Methode verringert den Radius der Kegel, während bei der Monte-Carlo Methode der Radius nicht kleiner wird. Somit verändert sich für den Optimierungsalgorithmus der Funktionswert bei der Worst-Case Methode stärker und er konvergiert dadurch eher zu einem der lokalen Minima. Bei der Monte-Carlo Methode muss der Funktionswert so klein werden (da sich der Kegeldurchmesser nicht so stark ändert wie bei der Worst-Case Methode), dass das robuste Optimum attraktiver wird.

Für die Worst-Case Methode wurde ein variabler Bereich von 0,1% des Suchbereichs angenommen. Es wurden jeweils 100 Runs durchgeführt, da die Berechnung mit der Monte-Carlo Methode bei 250 Runs sehr zeitintensiv ist. Für die Monte-Carlo Methode wurde ein σ-Wert von 0,3 angenommen, die Berechnung des neuen Funktionswertes wurde mit (5.1) und einer Gewichtung $w_1 = 0{,}25\ und\ w_2 = 0{,}75$ durchgeführt, und die Stichprobengröße wurde sukzessive verändert. Die Stichprobengröße betrug 2, 4, 10, 20, 50 und 100. Die Tabellen der ausgewerteten Ergebnisse befinden sich im Anhang 18-24.

Die Worst-Case Methode ergab folgendes Ergebnis: Das globale Optimum wurde zu 45% gefunden, das 1. lokale Optimum zu 24% und das 2. lokale Optimum zu 31%, obwohl der variable Bereich relativ klein angenommen wurde. Die Berechnungen für die Monte-Carlo Methode ergaben, dass ab einer Stichprobengröße von 10 Stichproben das globale Optimum zu 55%, das 1. lokale Optimum zu ca. 30% und das 2. lokale Optimum zu ca. 15% gefunden wurden. Daraus kann der Schluss gezogen werden, dass bei einer zweidimensionalen Problemstellung eine Stichprobengröße von mehr als 10

Stichproben ausreichend ist, da sich das Ergebnis mit einer größeren Stichprobenanzahl nicht mehr wesentlich verändert.

Vergleicht man die Suchergebnisse der Worst-Case Methode mit der Monte-Carlo Methode, stellt man fest, dass sich der Schwerpunkt der Ergebnisse bei der Worst-Case Methode auf das globale Optimum und das 2. lokale Optimum legt, wohingegen sich der Schwerpunkt der Suchergebnisse bei der Monte-Carlo Methode auf das globale und das 1. lokale Optimum legt. Wie schon eingangs erwähnt, verändern sich die Kegelradien und damit die Funktionswerte bei der Anwendung der beiden robusten Methoden unterschiedlich, was sich somit in diesem Ergebnis deutlich wiederspiegelt.

Kapitel 6: Zusammenfassung

Zu Beginn einer robusten Optimierung sollte immer eine generelle Notwendigkeit einer solchen überdacht werden. Wie eingangs erwähnt, entspricht eine robuste Lösung meist nicht der optimalen Lösung.

Bei der Worst-Case Methode hängt die Anzahl der zu berechnenden Funktionswerte mit 2^n von der Dimension n des Beispiels ab. Ist hier ein niederdimensionales Problem gegeben, bietet sich diese Methode an. Der Streubereich wird klar abgegrenzt, und es werden nur die umgebenden Eckpunkte berechnet, aus denen der schlechteste zum Finden der robusten Lösung gewählt wird. Steigt die Dimensionalität aber an, wird die Berechenbarkeit schwieriger, dauert länger und ist bei einer sehr hohen Dimension nicht mehr wirtschaftlich vertretbar.

Hier bietet die Monte-Carlo Methode basierend auf Streufunktionen Vorteile. Die Anzahl der Streuparameter ist grundsätzlich unabhängig von der Dimension. Dies bedeutet allerdings nicht, dass für jede Dimension gleich viele Stichproben benötigt werden. In Kapitel 3.3 wird gesagt, dass die Stichprobenanzahl, um den Fehler der Toleranzsimulation gering zu halten, unendlich groß sein sollte. Wie man aber anhand des Beispiels erkennen kann, ist bei einer Dimension von 2 ab einer Stichprobengröße von 10 keine größere Veränderung des Ergebnisses mehr erkennbar. Wird die Dimension größer, muss die Stichprobenanzahl auch vergrößert werden, aber nicht in demselben Ausmaß wie bei der Worst-Case Methode. Wenn es um den Berechnungsaufwand geht, gewinnt die Monte-Carlo Methode bei höheren Dimensionen. Sind die Dimensionen aber gering, bietet sich die Worst-Case Methode eher an. Ein weiterer Vorteil der Mont-Carlo Methode ist, dass Parameterrauschen auf diese Art sehr gut nachgebildet werden kann. Daraus lässt sich schließen, dass die Worst-Case Methode für Bereiche, die eine prozentuale Abweichung vom Standardwert mit einbeziehen sollen, geeigneter ist, sich aber die Methoden der Streufunktionen für verrauschte Signale besser eignen.

Zufallszahlen bzw. eine richtige Normalverteilung zu erzeugen, ist an sich keine einfache Aufgabe. Rechenmaschinen sind nur bis zu einem gewissen Grad in der Lage, wirklich zufällige Zahlen zu erzeugen. Ab einem gewissen Punkt wiederholen sich die Zahlensequenzen wieder. Einige Mathematiker beschäftigen sich nur mit der Entwicklung von Zufallszahlengeneratoren. Es soll hier nicht näher auf die Erzeugung einer Zufallszahl eingegangen werden, dennoch ist dieser Aspekt nicht unerheblich und sollte somit kurz erwähnt sein.

Anhand des eigens für diese Untersuchungen entwickelten Testbeispiels wurde aufgezeigt, welche Einflüsse die Parameter der Differentiellen

Evolutionsstrategie (Skalierungsfaktor, Crossover-Faktor) sowie der robusten Methoden auf das Finden der Lösung haben. Hier wurde der robuste Teil noch völlig außer Acht gelassen. Nachdem ein Testbeispiel festgelegt wurde, wurde die Veränderung der Parameter der Differentiellen Evolutionsstrategie beobachtet. Die Entwicklung eines robusten Algorithmus erfolgte anschließend. Dabei kam es auf die Wahl des Schwankungsbereiches der Parameter, auf die Stichprobengröße und die Gewichtung des Strafterms bei der Monte-Carlo Methode an. Die Bandbreite der möglichen Veränderungen der Algorithmus-Parameter ist damit nicht erschöpft. Es wurden hier die grundlegenden Effekte der Parameter gezeigt.

In weiterer Folge kann dieser Algorithmus für den mehrdimensionalen Raum betrachtet werden. Dabei sollte sich zeigen, dass die Mont-Carlo Methode hier an Reiz gewinnen kann, da die Anzahl der Stichproben, die für den mehrdimensionalen Bereich verwendet werden muss, kleiner als die Anzahl der Funktionswertberechnungen für die Worst-Case Methode ist und damit eine schnellere Berechnung des robusten Ergebnisses liefert.

Die hier eingesetzten Methoden betrachten des Weiteren einen gleichmäßigen, variierenden Raum um einen Parameter. Hier könnte durch unterschiedliche Grenzgrößen auch die Variabilität der Parameter den Problembedingungen noch angepasst werden. Durch die Gewichtungen bei der Monte-Carlo Methode kann weiters der Einfluss des Strafterms verändert werden. Dadurch entsteht die Möglichkeit, die Wichtigkeit der Variabilität der Parameter zu kontrollieren. Ist der Streubereich für das Ergebnis nicht sehr wichtig, sollte aber trotzdem beachtet werden, so wird man den Einfluss des Strafterms kleiner machen. Ist der Parameter aber ein z.B. sicherheitsrelevanter Parameter, so sollte man den Einfluss des Strafterms verstärken.

Zuletzt sei noch gesagt, dass das Gebiet der Robusten Optimierung beinahe unerschöpflich ist, da sich mit jedem veränderten Parameter wieder neue Gebiete eröffnen. Dadurch ist dieses Verfahren zwar rechenintensiv, aber dafür auch so flexibel, dass man unterschiedliche Probleme mit einer angepassten Methode bearbeiten kann. Durch diese Flexibilität ist der Spielraum aber auch sehr groß, sodass ein großes Potential für die Weiterentwicklung dieser Verfahren vorhanden ist.

Literaturverzeichnis:

[1] J. Marczyk, *Stochastic multidiciplinary improvement: Beyond optimization*, American Institute of Aeronautics and Astronautics AIAA-2000-4929

[2] Cenker, Uschida, *Optimierung mehrdimensionaler Funktionen* $f: \Re^n \rightarrow \Re$, 2004

[3] H.G. Beyer, B. Sendhoff, *Robust Optimization – A Comprehensive Survey*, Elsevier Science, March 2007

[4] U.Baumgartner, T. Ebner, Ch. Magele, *Numerical Optimization*, Institute for Fundamentals and Theory in Electrical Engineering, Technical University of Graz

[5] P. Alotto, *Differential Evolution* Vorlesungsunterlagen 2010, Dept.Electrical Eng., Univ. Of Padova, Italy

[6] Steffen Harbich, *Einführung genetischer Algorithmen mit Anwendungsbeispiel*, Universität Magdeburg 2007

[7] Jana Schäfer, *Evolutionsstrategien*, Aufzeichnungen zum Seminar „Evolutionäre Algorithmen“, Univerität Paderborn 2004

[8] R.Storn and K.Price, *DifferentialEvolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, Tech. Report, International Computer Sience Institute (Berlkeley), 1995

[9] R.Storn and K.Price, *DifferentialEvolution - A Simple and Efficient Heuristk for global Optimization over Continuous Spaces,* Journal of Global Optimization, vol. 11, dec. 1997, pp. 341-359

[10] P. Alotto, Ch. Magele, W. Renhart, G. Steiner, A. Weber, *Robust Target functions in electromagnetic design*, Emerald Compel Vol.22 No. 3, 2003 pp. 549-560

[11] Markus Könning, *Kombination von Optimierung und Robustheit in der Simulation mechanischer Systeme*, Robert Bosch GmbH, CR/ARD4, 71701 Schwieberdingen

[12] Y. Jin, B.Sendhoff, *Trade-Off between Performance and Robustness: An Evolutionary Multiobjective Approach*, Evolutionary Multi-Criterion Optimization, LNCS 2632, pp.237-252, 2003

[13] Christian Bucher, *Structural Optimization*, TU-Wien, Nov 2012

[14] M.G.Epitropakis, V.P. Plagianakos, M.N. Vrahatis, *Evolutionary Adaptation of the Differential Evolution Control Parameters*, IEEE978-1-4244-2959-2 2009

[15] Y. Ao, H. Chi, *Experimental Study on Differential Evolution Strategies*, IEEE 978-0-7695-3571-5, 2009

[16] Ch. Theis, W. Kernbichler, *Grundlagen der Monte Carlo Methoden*, TU-Graz, 2002

[17] M. Keramat, R. Kielbasa, *Efficient Average Quality Index Estimation of Integrated Circuits by Modified Latin Hypercube Sampling Monte Carlo (MLHSMC)*, IEEE 0-7803-3583-X 1997

[18] F. Busetti, *Simulated annealing overview*

[19] Bild Normalverteilung aus (Zugriff im Februar 2013) http://wirtschaftslexikon.gabler.de/Definition/normalverteilung

[20] M. Könning, *Kombination von Optimierung und Robustheit in der Simulation mechanischer Systeme*, Robert Bosch GmbH, präsentiert 2005 zu den 2.Weimarer Optimierungs- und Stochastiktage

[21] J. Will, D. Roos, j. Riedel, *Robustheitsbewertung in der stochastischen Strukturmechanik*, DYNARDO GmbH Weimar, 2003

Abbildungsverzeichnis:

Anhang:

500 Runs „Skrabl“	Position		Funktionswert	Ergebnisse	Prozentsatz
	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-1	1	-2,200	353	70,6
1. lokales Optimum	1	0	-1,998	137	27,4
2. lokales Optimum	-1	-1	-1,500	10	2,0

Anhang 1: Reine DE über 500 Runs mit Testfunktion „Skrabl“

Es wurden 500 Runs über die Testfunktion „Skrabl“ mit der reinen Differentiellen Evolutionsstrategie gebildet und gezählt, wie oft der Algorithmus welches Optimum findet. Anschließend wurde der Prozentsatz gebildet. Man erkennt anhand dieser Tabelle (Anhang 1), dass der Algorithmus zu mehr als 70% das globale Optimum findet.

100 Runs „Skrabl“	Position		Funktionswert	Ergebnisse	Prozentsatz
	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-1	1	-2,200	75	75
1. lokales Optimum	1	0	-1,998	24	24
2. lokales Optimum	-1	-1	-1,500	1	1

Anhang 2: Robuste DE über 100 Runs ohne Parametervariationen

Ergebnisse zur Überprüfung des robusten Algorithmus. Hier sollten die gleichen Ergebnisse vorliegen, wie bei der reinen DE.

250 Runs „Skrabl2“	Position		Funktionswert	Ergebnisse	Prozentsatz
	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	51	51
1. lokales Optimum	2,5	0	-1,998	34	34
2. lokales Optimum	-2,6	-2,6	-1,500	15	15

Anhang 3: Reine DE über 100 Runs über Testfunktion „Skrabl2“

Die reine DE über die Testfunktion „Skrabl2“ mit erweiterten Grenzen. Die Werte für das robuste Optimum und die lokalen Optima bleiben die gleichen. Es verändert sich nur die Position.

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=0,8 ; cr=0,5	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	184	73,6
1. lokales Optimum	2,5	0	-1,998	53	21,2
2. lokales Optimum	-2,6	-2,6	-1,5	13	5,2

Anhang 4: Reine DE über 250 Runs mit Testfunktion „Skrabl3“

Hier wurde nur die Differenz der Funktionswerte zwischen globalem und dem 1. lokalen Optimum vergrößert.

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=1,5 ; cr=0,5	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	197	78,8
1. lokales Optimum	2,5	0	-1,998	47	18,8
2. lokales Optimum	-2,6	-2,6	-1,5	6	2,4

Anhang 5: Reine DE über 250 Runs, Testfunktion „Skrabl3“, F=1,5 und cr=0,5

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=0,4 ; cr=0,5	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	117	46,8
1. lokales Optimum	2,5	0	-1,998	80	32
2. lokales Optimum	-2,6	-2,6	-1,5	53	21,2

Anhang 6: Reine DE über 250 Runs, Testfunktion „Skrabl3“, F=0,4 und cr=0,5

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=0,8 ; cr=0,8	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	159	63,6
1. lokales Optimum	2,5	0	-1,998	63	25,2
2. lokales Optimum	-2,6	-2,6	-1,5	28	11,2

Anhang 7: Reine DE über 250 Runs, Testfunktion „Skrabl3“, F=0,8 und cr=0,8

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=0,8 ; cr=0,3	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	185	74
1. lokales Optimum	2,5	0	-1,998	45	18
2. lokales Optimum	-2,6	-2,6	-1,5	20	8

Anhang 8: Reine DE über 250 Runs, Testfunktion „Skrabl3“, F=0,8 und cr=0,3

250 Runs "Skrabl3"	Position		Funktionswert	Ergebnisse	Prozentsatz
F=1,5 ; cr=0,3	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,5	200	80
1. lokales Optimum	2,5	0	-1,998	38	15,2
2. lokales Optimum	-2,6	-2,6	-1,5	12	4,8

Anhang 9: Reine DE über 250 Runs, Testfunktion „Skrabl3“, F=1,5 und cr=0,3

250 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
WC 10%	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	3	1,2
1. lokales Optimum	2,5	0	-1,998	70	28,0
2. lokales Optimum	-2,6	-2,6	-1,500	177	70,8

Anhang 10: 250 Runs mit der robusten Methode: Worst-Case bei einem variablen Bereich von 10% des Suchbereichs.

250 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
WC 1%	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	70	28,0
1. lokales Optimum	2,5	0	-1,998	73	29,2
2. lokales Optimum	-2,6	-2,6	-1,500	107	42,8

Anhang 11: 250 Runs mit der robusten Methode: Worst-Case bei einem variablen Bereich von 1% des Suchbereichs

250 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	88	35,2
1. lokales Optimum	2,5	0	-1,998	72	28,8
2. lokales Optimum	-2,6	-2,6	-1,500	90	36,0

Anhang 12: 250 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 0,5 und 100 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

250 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,5 w2=0,5	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	161	64,4
1. lokales Optimum	2,5	0	-1,998	70	28,0
2. lokales Optimum	-2,6	-2,6	-1,500	19	7,6

Anhang 13: 250 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 0,5 und 100 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}5\ und\ w_2 = 0{,}5$

250 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,75 w2=0,25	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	177	70,8
1. lokales Optimum	2,5	0	-1,998	64	25,6
2. lokales Optimum	-2,6	-2,6	-1,500	9	3,6

Anhang 14: 250 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 0,5 und 100 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}75\ und\ w_2 = 0{,}25$

100 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	55	55,0
1. lokales Optimum	2,5	0	-1,998	35	35,0
2. lokales Optimum	-2,6	-2,6	-1,500	10	10,0

Anhang15: 100 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 1 und 50 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

100 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,5 w2=0,5	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	73	73,0
1. lokales Optimum	2,5	0	-1,998	22	22,0
2. lokales Optimum	-2,6	-2,6	-1,500	5	5,0

Anhang 16: 100 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 1 und 50 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}5\ und\ w_2 = 0{,}5$

100 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,75 w2=0,25	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	66	66,0
1. lokales Optimum	2,5	0	-1,998	30	30,0
2. lokales Optimum	-2,6	-2,6	-1,500	4	4,0

Anhang 17: 100 Runs mit der robusten Methode: Monte Carlo mit einem σ-Wert von 1 und 50 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}75\ und\ w_2 = 0{,}25$

100 Runs	Position		Funktionswert	Ergebnisse	Prozentsatz
WC 0,1%	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	45	45,0
1. lokales Optimum	2,5	0	-1,998	24	24,0
2. lokales Optimum	-2,6	-2,6	-1,500	31	31,0

Anhang 18: 100 Runs mit der robusten Methode: Worst-Case bei einem variablen Bereich von 0,1% des Suchbereichs

Stichproben: 2	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	69	69,0
1. lokales Optimum	2,5	0	-1,998	15	15,0
2. lokales Optimum	-2,6	-2,6	-1,500	16	16,0

Anhang 19: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 2 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

Stichproben: 4	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	63	63,0
1. lokales Optimum	2,5	0	-1,998	22	22,0
2. lokales Optimum	-2,6	-2,6	-1,500	15	15,0

Anhang 20: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 4 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

Stichproben: 10	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	55	55,0
1. lokales Optimum	2,5	0	-1,998	29	29,0
2. lokales Optimum	-2,6	-2,6	-1,500	16	16,0

Anhang 21: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 10 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

Stichproben: 20	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	53	53,0
1. lokales Optimum	2,5	0	-1,998	32	32,0
2. lokales Optimum	-2,6	-2,6	-1,500	15	15,0

Anhang 22: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 20 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

Stichproben: 50	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	55	55,0
1. lokales Optimum	2,5	0	-1,998	21	21,0
2. lokales Optimum	-2,6	-2,6	-1,500	24	24,0

Anhang 23: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 50 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$

Stichproben: 100	Position		Funktionswert	Ergebnisse	Prozentsatz
MC w1=0,25 w2=0,75	Parameter 1	Parameter 2	f(x)	[# der Treffer]	[%]
Globales Optimum	-2,5	2,5	-2,200	55	55,0
1. lokales Optimum	2,5	0	-1,998	28	28,0
2. lokales Optimum	-2,6	-2,6	-1,500	17	17,0

Anhang 24: 100 Runs, robuste Methode: Monte Carlo mit einem σ-Wert von 0,03 und 100 Stichproben. Strafterm-Ermittlung über (5.1) mit den Gewichten $w_1 = 0{,}25\ und\ w_2 = 0{,}75$