

Entwicklung eines Systems zur Entfernungsabschätzung für Phasen basiertes UHF RFID Tracking durch Verwendung evolutionärer Berechnungsverfahren

Masterthesis eingereicht
zur Erfüllung der Anforderungen zum Erwerb des
akademischen Grades
Master of Science der Medizintechnik

Erstellt von
Christoph Gnip

Fachbereich Elektrotechnik und angewandte Naturwissenschaften
Westfälische Hochschule

September 2013

Master's Thesis

Titel: Entwicklung eines Systems zur Entfernungsabschätzung für Phasen basiertes UHF RFID Tracking durch Verwendung evolutionärer Berechnungsverfahren

Title: Development of a Distance Estimation System for Phase-Based UHF RFID Tracking by Utilizing Methods of Evolutionary Computation

University: Westphalian University of Applied Sciences
Department Electrical Engineering and Applied Sciences
Neidenburger Str. 43
45897 Gelsenkirchen
Germany

In Cooperation with: amedo Smart Tracking Solutions GmbH
Universitätstraße 142
Bochum

Author: Christoph Gnip
Luggendelle 28
48954 Gelsenkirchen
Germany

Matrikelnumber: 200720362

Supervisor: Prof. Dr. Frank Bärmann
Co-supervisor: Dipl.-Ing. Volker Trösken

Inhaltsverzeichnis

1. Einleitung	1
1.1. Allgemein	1
1.2. Motivation	3
1.3. Anforderungen	6
1.4. Ziel und Herangehensweise	6
2. Grundlagen	9
2.1. Mathematische Voraussetzungen	9
2.1.1. Kondition	9
2.1.2. SVD	10
2.2. Optimierung	11
2.2.1. Objektfunktion	11
2.2.2. Arten von Optima	12
2.3. Auffinden von Minima	13
2.3.1. Optimierungsräume	14
2.4. Evolutionäre Strategien	15
2.4.1. Evolutionsstrategien - Grundlagen	15
2.4.2. Strategien mit mehreren Populationen	19
2.5. Covariance Matrix Adaption - Anpassung der Kovarianzmatrix	20
2.6. Technische Voraussetzungen	22
2.6.1. Positionsgenauigkeit auf Funk basierender Verfahren	22
2.6.2. RFID	23
2.6.3. PRPS-Messystem	25
2.6.4. Phase und Wellenzahl	26
3. Hauptteil	29
3.1. Vorüberlegung zur Komplexität	29
3.2. Entwicklung des Modells	33
3.3. Antennen Permutationen	40
3.3.1. Anwendung der Permutationen	41
3.4. Erweiterte Betrachtung der Kondition	42
3.4.1. Weitere Anwendung der Konditionszahl	43
3.5. Einsatz des Modells	44
3.6. Betrachtung der Komplexität	44
3.7. Realisierung der Kalibrierung	48
3.7.1. Implementation	48

3.8. Software	52
3.8.1. Shark	52
3.8.2. Implementation	52
3.8.3. Paralleler Ablauf	58
3.8.4. Schnittstellen für Dateneingabe	59
3.8.5. Ablaufdiagramme	59
4. Ergebnisse	63
4.1. Ergebnisse der Kalibrierung	63
4.1.1. Visualisierung der Ergebnisse	64
4.2. Ergebnisse des Trilaterationsmodells	70
4.2.1. Experimente	70
4.2.2. Ergebnisse ideale Messwerte	70
4.2.3. Ergebnisse ideale Messwerte	72
4.2.4. Ergebnisse reale Messwerte	72
4.2.5. Evolutionsverlauf - real vs ideal	72
4.2.6. Performance	73
5. Diskussion	81
6. Schluss	83
6.1. Ausblick	83
6.2. Zusammenfassung	84
A. Abbildungen	87
A.1. Messaufbauten	87
B. Quellcodeauszüge	89
B.1. ObjectiveFunktion - Evolutionary Calibration Header	89
B.2. ObjectiveFunktion - Evolutionary Calibration Source	92
C. Gnuplot Skripte	93
C.1. Boxplot	93
C.2. Lineplot	97
C.3. Scatterplot	100
D. Fitness Plots	105
D.1. Antennen 1,2 und 3	106
D.2. Antennen 4,5 und 6	107
D.3. Antennen 7 und 8	108
E. Projektverwaltung	109
E.1. Projektlaufplan	109

Abbildungsverzeichnis

1.1.	Anforderungsspinne	6
2.1.	Übersicht numerische Verfahren	12
2.2.	Übersicht nichtlinearer Optimierungsstrategien	16
2.3.	Ablauf Evolutionsstrategie	17
2.4.	Konzept direkter Optimierung mittels CMA-ES	21
2.5.	Beispiele für Transponder und Lesegeräte	24
2.7.	PRPS der amedo STS	25
2.8.	PRPS der amedo GmbH	26
2.9.	Zusammenhang Wellenlänge - Wellenzahl	27
3.1.	Profil einer Phasenmessung	29
3.2.	Reale Messwerte visualisiert	31
3.3.	Normierte Messwerte von Kalibriermessung	32
3.4.	Antennen-Szene mit einem Tag	34
3.5.	Ablauf libPermuatae	41
3.6.	Ergebnisse der Konditionsanalyse alle Permutationen	43
3.9.	Fitness Ebenen Heatmap	45
3.10.	Fitness Ebenen Heatmap, vergrößert	46
3.11.	Übrige Ebenen für Antenne 1	47
3.13.	Kalibrierwerkzeuge	49
3.15.	Ablauf der Kalibrierung	50
3.16.	Ablauf der libCalibration	51
3.17.	Ablauf Programmstart	60
3.18.	Finde Modelllösung	61
3.19.	Modelllösung bestimmen	62
4.1.	Box-Plot der Endergebnisse der Kalibrierung	65
4.2.	Linien-Plot der Endergebnisse der Kalibrierung	66
4.3.	Kalibrierung Scatter-Plot	67
4.4.	Statistisch verteilte Ergebnisse der Kalibrierung mittels ES	68
4.5.	Visualisierung des Kalibrierendergebnis	69
4.6.	Ergebnis-Heatmap - Ideale Messwerte	74
4.7.	Ergebnis-Heatmap - Reale Messwerte	75
4.8.	Limitierte Ergebnisse - Ideale Messwerte	76
4.9.	Limitierte Ergebnisse - Reale Messwerte	77

4.10. Evolutionsverlauf der Ergebnisse	78
4.12. Performance Ergebnisse - Ideale Messwerte	79
4.13. Performance Ergebnisse – Reale Messwerte	80
5.1. Anforderungsspinne	81
A.1. PRPS-Kalibiersystem	87
A.2. Übersicht Kalibrieraufbau	88

Tabellenverzeichnis

1.1.	Anforderungen Trackingsysteme	2
1.2.	Übersicht Navigationsverfahren	3
3.1.	Parameter der Fitness Ebenen	47
4.1.	Finale Antennen Koordinaten	63
4.2.	Experimente - Ideale Messdaten	71
4.3.	Experimente - Reale Messdaten	71

Listings

3.1.	Quellcodeschnipsel für die Deklaration einer Objektfunktion	53
3.2.	Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark	55
3.3.	Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark	56
3.4.	Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark	57
3.5.	Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark	57
3.6.	Erstellen von mehreren Threads bei gleichzeitiger Übergabe der Parameter	58
3.7.	Abfragen der Asynchronen Ergebnisse mit Hilfe des auto-Datentyp. Ausgabe des Ergebnis in einer Datei.	58
C.1.	Gnuplot Boxplot-Skript	93
C.2.	Gnuplot Lineplot-Skript	97
C.3.	Gnuplot Scatterplot-Skript	100

Verwendete Abkürzungen

ES	Evolutionäre Strategie (<i>Evolutionary Strategy</i>)
CMA-ES	Covariance Matrix Adaption - Evolutionary Strategy
C++11	Programmiersprache C++ in der Version 11
MRT	Magnetresonanztomografie
RFID	Radio-Frequency Identification
LOS	Line of Sight
CSV	Comma seperated Values
Rö	Röntgen
CT	Computertomografie
UHF	Ultra High Frequency; Frequenzbereich zwischen 0,3 und 3 GHz
GHz	Gigahertz
EM	Elektromagnetismus
TAG	Transponder/ Receiver für Funk Kommunikation
Tracking	Positionsbestimmung
PRPS	Passiv RFID Positioning System - Produkt der amedo STS
TOF	Time Of Flight
PD	Phasendifferenz
RSSI	Indikator für die empfangene Signalstärke (Received Signal Strength Indication)
CMake	Cross-Platform make; Werkzeug für Erstellung von Software
ETSI	European Telecommunications Standards Institute
amedo STS	amedo Smart Tracking Solutions

Verwendete Symbole

A	Matrizen werden mit fetten Großbuchstaben notiert
b	Vektoren werden mit fetten Kleinbuchstaben notiert
0	Nullvektor
<i>k</i>	ist der Index der Antennen im Aufbau verwendeten Antennen
r_k	Abstand vom Tag zur indizierten Antenne
r_0	Abstand vom Tag zur Referenzantenne
d_{k0}	Abstand zur Landmarke (Index 0) zur indizierten Antenne
μ	Eigenwert; Bezeichnet einen Eigenwert; Im Rahmen der SVD-Verfahrens.
μ	Elternteil; Im Rahmen der evolutionären Berechnung; Anzahl der Eltern pro Generation.
λ	Wellenlänge; Im Rahmen der Positionsberechnung.
λ	Nachkomme; Im Rahmen der evolutionären Berechnung; Anzahl der Nachkommen pro Generation
(μ, λ)	"Komma"- Evolutionsstrategie
$(\mu + \lambda)$	"Plus"- Evolutionsstrategie
ϱ	Phase

1. Einleitung

1.1. Allgemein

Mit der Entwicklung der minimal-invasiven Chirurgie, einer Operationsmethode bei der durch sehr kleine Einschnitte in den Körper mit besonders filigranen Operationsinstrumenten operiert wird, verändert sich die Art Operationen durchzuführen grundlegend. Eingriffe können schneller, schonender und effizienter durchgeführt werden. Möglich wird diese Entwicklung durch eine Vielzahl neuartiger technischer Systeme. Die Vorteile gegenüber herkömmlichen Operationstechniken begründen die weite Verbreitung und den häufigen Einsatz der minimal-invasiven Techniken.

Mit fortschreitender Miniaturisierung der Instrumente und der Verlagerung des Operationsgebietes in den Patienten geht die optische Kontrolle über das Operationsgebiet sowie Instrumentarium verloren. Diese ist unabdingbar für einen Erfolg der Operation und die Information (z.B. Lage der Instrumente) müssen dem Operierenden jeder Zeit zur Verfügung stehen. Eine Übersicht über das Operationsgebiet im Inneren des Körpers kann durch Kameratechniken (Endoskope) gewonnen werden, allerdings wird dafür viel Platz benötigt. Die Kenntnis über den Aufenthaltsort und die Lage des Instrumentariums ist, gerade bei schwierigen Operationen, unabdingbar. Um an diese Informationen zu gelangen ist es Stand der Technik, bildgebende Verfahren intraoperativ, d.h. während der Operation, Bildvolumina zu generieren und auszuwerten.

Beispielsweise werden bei kardiologischen Interventionen (z.B. Platzierung eines Stents durch die Arteria iliaca interna¹ in den Coronargefäßen des Herzens) permanente Lagekontrolle der Katheter mittels Röntgentechnik durchgeführt. Dabei wird der Patient und das Operationsteam permanent ionisierender Bestrahlung ausgesetzt. Es können auch Bilder durch Magnetresonanztomografie oder durch andere bildgebende Verfahren erzeugt werden. Die Gewinnung dieser Bilddaten ist schwierig (MRT) oder nicht gut geeignet (Ultraschall) ist. Ein weiteres Erschwernis ist, dass der Patient dafür häufig mitsamt den Instrumenten umgelagert werden muss. Das Umlagern bringt weitere Risiken mit sich und ist mit weiterem Aufwand verbunden. Besonders könne die so gewonnenen Informationen nicht als genau angenommen weden.

¹innere Beckenarterie- Standardzugang für diese Art von Operationen

Eine Lösung für diese Probleme bringen sog. Trackingsysteme. Diese sind in der Lage die Position, z.B. eines Instrumentes, zu ermitteln und stellen die benötigten Informationen für den Arzt zur Verfügung. Dabei kommen bei den verfügbaren Systemen unterschiedliche physikalische Prinzipien zu Einsatz, die verschiedene Vor- und Nachteile bieten.

Die Anwendung solcher Systeme erlaubt außerdem eine softwaregestützte Planung und assistierte Durchführung der Operation. Die Kombination dieser Techniken wird Navigation genannt. Die Möglichkeit der Planung und Kontrolle macht diese Systeme im Zuge der stets steigenden Ansprüche an das Qualitätsmanagement interessant. Die Anforderungen die vom Anwender im klinischen Alltag an die Systeme gestellt werden sind:

- Gute Genauigkeit
- Hohe Verfügbarkeit
- Leichte Bedienbarkeit
- Einfache Einbindung Workflow
- Geringe Kosten
- Sicherheit

Tabelle 1.1.: Anforderungen an ein medizintechnisches Messsystem. Nicht vollständig.

Die Anforderungen an ein solches System sind somit sehr hoch. Sie müssen über eine entsprechende technische Leistungsfähigkeit verfügen (Genauigkeit, Verfügbarkeit etc.) und gleichzeitig muss der Umgang mit ihnen leicht sein und dürfen keine hohen Kosten für Anschaffung und Betrieb generieren.

Stand der Technik

Es befinden sich Trackingsysteme unterschiedlicher Hersteller am Markt. Sie beruhen auf unterschiedlichsten Messprinzipien und unterliegen den daraus resultierenden Limitierungen. Die wichtigsten technischen Unterschiede sind im Folgenden tabellarisch zusammengefasst:

Arbeitsweise	Optisch	Magnetisch	Ultraschall	Funk (UHF)
Genauigkeit	gut	ausreichend	gut	sehr gut
Frequenz	mittel	hoch	gering	hoch
Volumen	mittel	klein	mittel	groß
LOS	Ja	Ja	Nein	Nein
In Vivo	Nein	Nein	Nein	Ja

Tabelle 1.2.: Grobe Übersicht und Einteilung verschiedener Navigationsverfahren anhand ihres physikalischen Messprinzips.

Die Tabelle 1.2 teilt die unterschiedlichen Systeme anhand ihres physikalischen Messprinzips ein. Herausgestellt werden vor allem die wesentlichen Messparameter der Verfahren. Zusätzlich sind die Punkte LOS und In Vivo aufgeführt. In Vivo ist lateinisch und bedeutet: im Lebenden. Damit ist gemeint, dass Messsignale aus dem Patienten empfangen können. Aus der Auflistung lassen sich Vor- und Nachteile ableiten. Das größte Problem ist das eine direkten Sicht auf die Objekte vorausgesetzt wird, Line of Sight (LOS) genannt. Dem sog. LOS-Problem unterliegen fast alle Verfahren, die ein großes Messvolumen abdecken. Auf Funk basierende Methoden erfahren dieses Problem nicht, ihre Wechselwirkungsmechanismen erlauben die Durchdringung von Materie. Andere Wechselwirkungen beeinflussen die in Tabelle 1.1 beschriebenen Anforderungen negativ und limitieren das jeweilige Verfahren zusätzlich. Der größte Vorteil des auf Funk basierenden RFID-Verfahrens ist es verschiedene Objekte von einander zu unterscheiden, bzw. sogar zu identifizieren. Die Genauigkeit (im technischen Sinne: hohe Präzision plus hohe Richtigkeit) der Messung ist bei allen Verfahren mindestens ausreichend.

Diese Anforderungen allein sind eine große Herausforderung an die Technik. Hinzukommen weitere Anforderungen, die sich z.B. aus dem Ablauf einer Intervention ergeben. Ein System muss eine einfache Integrationsmöglichkeit in diesen Arbeitsablauf bieten. Zusätzlich dürfen durch die Anschaffung und das Material keine großen Kosten entstehen.

1.2. Motivation

Bestimmung der Position (im Folgenden "Tracking" genannt) mittels RFID ist eine vielversprechende Technik und konkurrierenden Verfahren in viele Punkten überlegen, vgl. Tabelle 1.2. Dabei stehen drei Unterscheidungsmerkmale hervor:

1. Es wird keine LOS benötigt

2. Separation und Identifikation mehrerer Objekte
3. Messsignal kann aus de Patienten empfangen werden

Die Vorteile lassen sich aus dem zugrunde liegende physikalischen Messprinzip ableiten. Es werden elektromagnetische Signale ausgewertet, die Wechselwirkungen vom EM-Signalen erlauben es Materie zu durchdringen. Insbesondere im Vergleich mit optischen Verfahren ist die auf Funk basierende RFID damit überlegen. Die Eigenschaft erlaubt es Objekte im Patienten zu lokalisieren. Entsprechende Untersuchungen über die erreichbare Positionsgenauigkeit dieser Verfahren im Körper sind vielversprechend. [17]

Es können mehrere Objekte von einander unterschieden und identifiziert werden. Man kann zusätzliche Informationen auf den Objekten ablegen und abfragen. Durch das einfache Anbringen von RFID-Tags unterschiedlicher Bauarten kann (siehe Abbildung 2.6a) kann nahezu jeder Gegenstand oder Person einem Tracking unterzogen werden. Dadurch wächst das Anwendungsspektrum weiter. Besonders das Auslesen von zusätzlichen Informationen ist mit keiner der anderen Technologien möglich.

Auf Funk basierende Verfahren bieten zudem ein sehr gute Ortsauflösung. Diese ist essentiell für eine sichere Positions- und Lagebestimmung von Objekten. Die Auflösung ist jedoch abhängig von dem Messprinzip und wird in Abschnitt 2.6.1 genauer beschrieben. Das von dem Messsystem der amedo STS verwendete Verfahren basiert auf der Messung der Phasendifferenz der Antwort eines Objekts. Aus den dort aufgeführten Gründen verwendet das PRPS der amedo STS eine Phasendifferenzmessung. Die Phasenlage ist direkt proportional zu einer Entfernung. Die Messung erreicht in der Theorie eine sehr gute Auflösung ($\leq 1 \text{ mm}$), sie ist jedoch nicht eindeutig (siehe Abschnitt 2.6.2). Das Problem kann umgangen werden, indem man ein ganzzahliges Vielfaches der Wellenlänge auf das Messergebnis aufaddiert. Man erhält die sog. *Wellenzahl*². Ist diese für alle Objekte und allen Antennen bekannt kann die Postion sicher bestimmt werden.

Die Annahme, dass die Wellenzahl für alle Zeiten t bekannt ist, ist naiv und höchstens unter Laborbedingungen richtig. In der Praxis müssen hier starke Einschränkungen, aufgrund der komplexen Wechselwirkungen auf EM-basierende Verfahren, gemacht werden. Die Betrachtung der Komplexität wird in Abschnitt 3.1 und 3.6 behandelt. Kann ein Objekt für kurze Zeit nicht abgefragt werden ist sofort ersichtlich, dass die Postion nicht mehr bestimmt werden kann. Die Wellenzahl(en) muss neu ermittelt werden.

²Diese ist nicht identisch mit der in der Physik gebräuchlichen Wellenzahl zur Beschreibung der Eigenschaften einer Welle.

Bisherige Ansätze die Wellenzahl nach Verlust des Objektes zu ermitteln basieren häufig auf Methoden der Statistik. Diese scheitern an der Komplexität des Problems oder benötigen sehr aufwändige Messreihen mit großer Anzahl an Messpunkten [3]. Eine weitere Methode die einen Schätzwert auf Basis des RSSI-Wertes berechnet wurde in vielen Ansätzen implementiert, z.B. [25]. Die Praxistauglichkeit dieser Methoden ist limitiert, die Abschätzung anhand des RSSI-Wertes funktioniert im Labor ausreichend gut, auch hier kommt es zu komplexen Wechselwirkungen. Ein weiteres Problem ist ihr große Anzahl an Wellenzahlen die bei Messaufbau vorkommen, die ein großes Volumen abdecken. Der Zusammenhang wird mit der Herleitung der Wellenzahl später beschrieben.

Zusammenfassend lassen sich über das Problem folgende Aussagen treffen. Das Problem ist:

1. Sehr komplex
2. Hochdimensional
3. Nicht linear³
4. Ohne analytische Lösung

Traditionell werden Probleme mit diesen Eigenschaften durch Methoden der Stochastik, Optimierung oder des maschinellen Lernens behandelt. Eine vollständige Übersicht und Abgrenzung der verschiedenen Gebiete ist im Rahmen dieser Arbeit nicht möglich bzw. sinnvoll. Eine Übersicht findet ist in Abbildung 2.1 zu finden.

Ein Teilgebiet der Optimierung stellen evolutionäre Berechnungsverfahren dar. Dies sind eine Klasse von Algorithmen, die sich für komplexe Problemfälle eignen. Sie stellen kaum Forderungen an die mathematische Formulierung des Problems, wie z.B. Stetigkeit, vollst. Differenzierbarkeit etc. Unter dem Begriff der Evolutionären Berechnungsverfahren fallen viele verschiedene Techniken, eine Übersicht über die Teilgebiete gibt die Abbildung ???. In dieser Arbeit wird das Teilgebiet der Evolutionären Optimierung verwendet um die Problemstellung zu lösen. Später wird diese Klasse von Verfahren ausführlich dargestellt.

Es sollte in jedem Fall möglich sein eine Lösung über diesen Ansatz zu finden. Vorüberlegungen und Machbarkeitsstudien wurden vom Institut für Neuroinformatik (INI) an der Ruhr-Universität Bochum angestellt. [28, 19]. An dem Institut wird seit mehreren Jahren an dieser Art von Algorithmen geforscht und unterschiedlichste Problemstellungen sind mit diesen Algorithmen gelöst worden, bspw. [23, 29]

³Positionsbestimmung aus superposition von EM-Wellen

In dieser Arbeit soll mittels evolutionärer Optimierung das beschriebene Problem gelöst werden. Im Endergebnis soll dabei eine Abschätzung der Entfernung zu einem Referenzpunkt möglich sein. Darüber lässt sich im Anschluss die Wellenzahl ermitteln.

1.3. Anforderungen an die Lösung

Aus den bisher vorgestellten Überlegungen können nun folgende Anforderungen abgeleitet werden:

1. Gute Performance
2. Unabhängigkeit von Stütz- und Kalibrierpunkten
3. Eindeutigkeit der Lösung
4. Eignung für ein großes Messvolumen
5. Nahtlose Integration in das bestehende Software Ökosystem
6. Aktualität; Stand der Softwaretechnik entsprechend

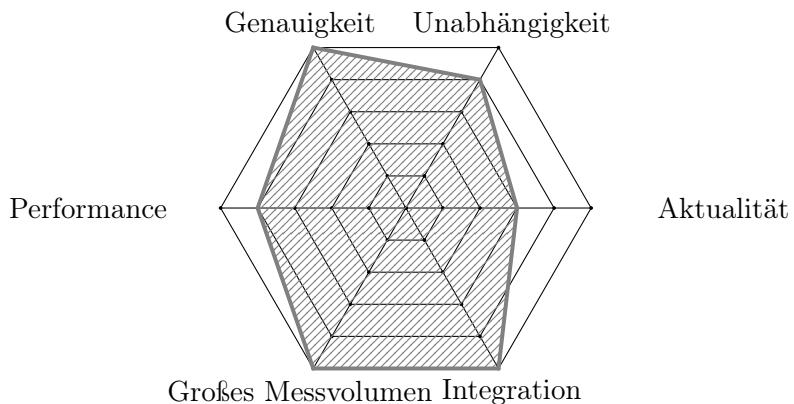


Abbildung 1.1.: Grafische Übersicht der Anforderungen an das System

1.4. Ziel und Herangehensweise

Das Ziel der Arbeit ist die Entwicklung eines Systems zur Abschätzung der Position eines Tags. Das Auffinden der Lösung soll die oben abgeleiteten Anforderungen erfüllt. Die Ermittlung einer korrekten Lösung ist jedoch das Wichtigste. Das System wird im Kern die Lösung über ein numerisches Optimierungsverfahren finden, im speziellen kommt das sog. '*Covarianz Matrix Adaption - Evolutionary Strategy*' (CMA-ES) zum Einsatz. Bei diesem Verfahren handelt es sich um ein stochastische, ableitungsfreies Verfahren, dass

für nicht lineare, nicht konvexe, kontinuierliche Probleme geeignet ist. Dazu wird zuerst ein Modell entworfen, dass sich für einen Einsatz in diesem Verfahren eignet. Das eingesetzte Lösungsverfahren stellt praktisch keine Anforderungen an ein solches Modell. Daher soll es mit möglichst wenig Annahmen bzw. Einschränkungen auskommen und dennoch ein relativ sicheres, reproduzierbares Ergebnis liefern. Weiterhin soll eine Integration der Lösung in das Software-Ökosystem der amedo STS erfolgen. Softwarekomponenten sollen in weiteren Projekten zum Einsatz kommen. Daher wird bei der Umsetzung auf eine größtmögliche Wiederverwendbarkeit geachtet. Es werden verschiedene Implementierungen des CMA-ES-Algorithmus recherchiert, verglichen und die geeignete gewählt. Das System soll unmittelbar in den Produkten der amedo STS zum Einsatz kommen können, daher wird eine entsprechende Schnittstelle für andere Software implementiert. Im Rahmen dieser Arbeit wird eine Methode entwickelt, um die Position von frei im Raum angeordnete Antennen zu ermitteln und dem Messaufbau zu kalibrieren.

2. Grundlagen

Im Rahmen dieses Kapitels werden die Grundlagen für diese Arbeit beschrieben. Zielgruppe dieser Arbeit sind fachkundige Personen die ein gewisses Grundwissen/-verständnis mitbringen oder solche die es werden wollen. Die Ausführungen werden in der, für das Verständnis dieser Arbeit angebrachten Tiefe, beschrieben. Da das Spektrum der Themen zu groß ist, kann nicht auf alles im Detail eingegangen werden. Allgemeine Zusammenhänge und Techniken, denen einen großen Stellenwert in dieser Arbeit zukommt, werden zusammengefasst präsentiert. Für detaillierte Beschreibungen wird stets auf entsprechende Fachliteratur verwiesen.

2.1. Mathematische Voraussetzungen

Dieser Abschnitt behandelt die mathematischen Voraussetzungen für diese Arbeit.

2.1.1. Kondition

Gegeben ist ein lineares Gleichungssystem der Form:

$$\mathbf{Ax} - \mathbf{b} = \mathbf{0}$$

Eine numerische Lösung führt in der Regel zu einer von $\mathbf{0}$ verschiedenen Lösung (insbesondere bei überbestimmten Systemen), so das wir:

$$\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} = \mathbf{r}$$

schreiben. Man nennt \mathbf{r} den Residuumvektor. Es ist offensichtlich, dass ein kleines Residuum nicht hinreichend ist um von einem kleinen relativen Fehler auszugehen.

Weiter folgt aus $\mathbf{Ax} - \mathbf{b} = \mathbf{0}$ und $\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} = \mathbf{r}$, dass

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{r}$$

und damit: $\|\mathbf{b}\| = \|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$, $\|\Delta\mathbf{x}\| = \|-\mathbf{A}^{-1}\mathbf{r}\| \leq \|\mathbf{A}^{-1}\|\|\mathbf{r}\|$. Wir können nun für den relativen Fehler schreiben:

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}^{-1}\|\|\mathbf{r}\|}{\|\mathbf{b}\|/\|\mathbf{A}\|} = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

Der Term $\|\mathbf{A}\| \|\mathbf{A}^{-1}\| := \text{cond}(\mathbf{A})$ heißt Konditionszahl. Auch der Begriff Konditionsmaß ist gebräuchlich und bezieht sich auf die gewählte Matrixnorm. Es kann gezeigt werden, dass $\text{cond}(\mathbf{A}) \gg 1$ für eine schlechte Konditionierung der Matrix steht. Wird im Folgenden von einer speziellen Matrixnorm gesprochen schreiben wir $\text{cond}(\mathbf{A})$ zu

$$\text{cond}_k(\mathbf{A}) = \|\mathbf{A}\|_k \|\mathbf{A}^{-1}\|_k$$

Der Index k wird entsprechend für die verwendete Norm ersetzt. Beispielsweise ergibt sich für die Konditionszahl der Spektralnorm¹:

$$\text{cond}_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 = \sqrt{\frac{\mu_{\max}}{\mu_{\min}}}$$

Die Symbole μ_{\max} und μ_{\min} stehen für die Eigenwerte des Systems.

Die Konditionszahl ermöglicht eine Analyse der Güte einer Lösung, die mittels Numerischer Verfahren ermittelt wurde. Nach [13] kann man folgende Aussage über die Konditionszahl treffen:

"Wird ein lineares Gleichungssystem $Ax = b$ mit t -stelliger dezimaler Gleitpunktarithmetik gelöst und beträgt die Konditionszahl $\text{cond}(A) \approx 10^\alpha$, so sind auf Grund der im allgemeinen unvermeidbaren Fehler in den Eingabedaten A und b nur $t - \alpha - 1$ Dezimalstellen der berechneten Lösung \tilde{x} (bezogen auf die betragsgrößte Komponente) sicher."

2.1.2. SVD

Bei dem Verfahren der Singular Value Decomposition (oder auch Singulärwertzerlegung), kurz SVD, handelt es sich um eine Faktorisierung einer Matrix. Die Matrix wird dabei als Produkt von drei Matrizen dargestellt. Diese Matrizen enthalten die sog. Singulärwerte und können aus einer der Matrizen abgelesen werden. Die Eigenschaften des Systems sind, ähnlich den Eigenwerten, aus den Singulärwerten bestimmbar. Besonders an der SVD ist, die Existenz für jede Form von Matrix - einschließlich nicht quadratischer Matrizen.

Die SVD basiert auf folgender Theorie der linearen Algebra: Jede $M \times N$ Matrix \mathbf{A} kann als Produkt einer $M \times N$ Spalten-orthogonalen Matrix \mathbf{U} , einer $N \times N$ Diagonalmatrix Σ mit Werten ≥ 0 und einer dritten adjungierten $N \times N$ -Matrix \mathbf{V}^* , so ergibt sich:

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^* = \mathbf{U} \Sigma \mathbf{V}^T \quad (2.1)$$

¹<http://de.wikipedia.org/w/index.php?title=Spektralnorm&oldid=118988565>

Ist \mathbf{A} eine reelwertige Matrix gilt: $\mathbf{V}^* = \mathbf{V}^T$. Die Matrix Σ ist im Rahmen dieser Arbeit von besonderem Interesse, denn sie enthält die Singulärwerte σ_r . Ihre Gestalt ist wie folgt:

$$\Sigma = \left(\begin{array}{ccc|cc|c} \sigma_1 & & & & & \vdots \\ & \ddots & & & & \vdots \\ & & \sigma_r & & & \vdots \\ \hline & & & & & \vdots \\ \vdots & & & & & \vdots \\ \dots & 0 & \dots & \dots & 0 & \dots \\ \vdots & & & & & \vdots \end{array} \right)$$

, wobei $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$

Da die σ_r der Matrix mit den Eigenwerten in Verbindung stehen, kann aus dieser Matrix die Konditionszahl bestimmt werden. Sie ist durch folgendes Verhältnis gegeben:

$$cond(\mathbf{A}) = \frac{\max(\sigma_r)}{\min(\sigma_r)} = \frac{\max(\sigma_1)}{\min(\sigma_r)} \quad (2.2)$$

Es gibt bereits viele Implementationen des Verfahrens, z.B. [20]. Diese Implementation wird durch den Erwerb der entsprechenden Lizenz im Rahmen dieser Arbeit verwendet. Weitere Informationen zum Verfahren sind z.B. in [4, Kapitel 4.6.3] zu finden.

2.2. Optimierung

Dieser Abschnitt führt in die Optimierung ein und gibt einen Überblick über die Grundbegriffe, die im Rahmen dieser Arbeit verwendet werden. Tiefe Einsichten und die theoretischen Grundlagen sind z.B. in [1, 26] zu finden.

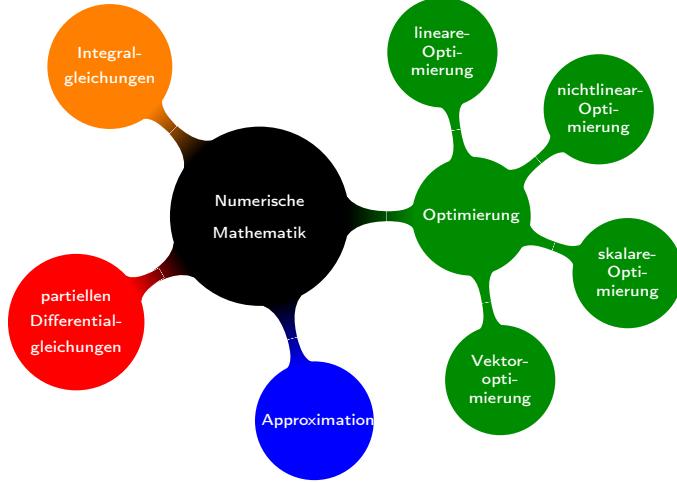
2.2.1. Objektfunktion

Für eine Optimierung wird eine Größe benötigt, die optimal werden soll. Dazu bedarf es einer Formulierung der Funktion die das Optimierungsziel enthält. Diese Funktion wird Objektfunktion², Fitnessfunktion oder Zielfunktion genannt. Auch Güte- oder Qualitätsfunktion sind gebräuchlich. Es sind eine Reihe von Optimierungskriterien denkbar, z.B. können Gewicht, Größe, Fläche etc. optimiert werden. Es kann auch mehr als ein Ziel Bedingung sein, in diesem Fall spricht man von Mehrzieloptimierung³.

²Wird in dieser Arbeit verwendet

³Keine weitere Erläuterung im Rahmen dieser Arbeit

Abbildung 2.1.: Übersicht über numerische Verfahren (unvollständig). Interessant für diese Arbeit ist der teil *nichtlineare Optimierung*. Dieser wird an andere Stelle ausführlicher gezeigt.



Damit man eine Optimierung durchführen kann, muss die Objektfunktion freie Parameter (Variablen) enthalten.

$$y = f(x), \quad x \in \mathbb{R}$$

Diese Formulierung ist eine eindimensionale Objektfunktion. Dabei ist x der freie Parameter den man variieren kann, um ein Optimum der Funktion $f(x)$ zu finden. In der Regel ist der Wert des erreichten Optimums nicht sehr interessant. Vielmehr ist man an dem auffinden der Optimalen Einstellung der freien Parameter interessiert.

2.2.2. Arten von Optima

Man kann leicht verstehen, dass es unterschiedliche Optima gibt. Es kann das z.B. das geringste Gewicht oder der größte Gewinn Ziel der Optimierung sein. Dazu notieren wir:

$$y = \min\{f(\mathbf{x})\} = -\max\{-f(\mathbf{x})\}, \quad \mathbf{x} \in \mathbb{R}^n$$

Die Gleichung drückt aus, dass ich jedes Maximierungsproblem in ein äquivalentes Minimierungsproblem überführen lässt. Daraus kann eine allg. Formulierung des Optimierungsproblems angegeben werden.

Allgemeine Formulierung des Optimierungsproblems

Ableitung der allgemeinen Formulierung aus den bisherigen Ausführungen.
Die Funktion:

$$y = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n$$

sei zu optimieren. Dabei ist eine Minimierungsaufgabe gleich der Maximierungsaufgabe. Das bringt die Formulierung:

$$\min\{ y(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X} \} \quad (2.3)$$

In der Formulierung ist eine Einschränkung enthalten. Wir fordern, dass $\mathbf{x} \in \mathbf{X}$ sein soll. Diese Einschränkung entstammt den Nebenbedingung des Problems. Herleitung:

$$g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \dots, n$$

Ergibt den zulässigen Bereich:

$$\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^n \mid g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \dots, n\}$$

Allgemeine von Minima

Eine Objektfunktion kann zwei Arten von Minima enthalten:

- **Lokales Minimum** - Ein zulässiger Punkt, bei dem in der Nachbarschaft keine niedrigeren Funktionswerte zu finden sind
- **Globales Minimum** - Ein zulässiger Punkt, der den geringsten Funktionswert des gesamten zulässigen Bereichs aufweist. Gleichzeitig ein lokales Minimum.

Wir haben ein mathematisches Modell der Optimierungsaufgabe erstellt. Wir können noch keine Aussage über die Komplexität des Problem treffen. Um den Schwierigkeitsgrad bestimmen zu können muss man die Zielfunktion analysieren. Eine Analyse der in dieser Arbeit verwendeten Objektfunktion wird im Hauptteil gezeigt. Ein Optimierungsproblem mit Nebenbedingungen wird registriertes Optimierungsproblem genannt. Ohne spricht man von unregistrierten Optimierungsproblemen.

2.3. Auffinden von Minima

Die Kenntnis der Zielfunktion und des mathematischen Modells erlaubt die Anwendung eines Algorithmus, der das Minima bestimmt. Übergibt man die Zielfunktion und die Fragestellung an einen solchen Algorithmus berechnet dieser auf unterschiedlichen Wegen mögliche Optima. Man kann zwei Hauptklassen von Verfahren unterscheiden:

- Lineare Optimierungsverfahren
- Nichtlineare Optimierungsverfahren

Auf die linearen Verfahren wird im Rahmen dieser Arbeit nicht eingegangen. Sie eignen sich nicht für komplexe Fragestellungen. Eine Übersicht über die nichtlinearen Verfahren ist in der Abbildung 2.2

2.3.1. Optimierungsräume

In der Optimierung kann man verschiedene Optimierungsräume betrachten. Der häufigste und auf den alle Algorithmen Anwendbar sind ist der reellwertige oder kontinuierliche Raum. Daneben kommen Probleme vor, die auf einem ganzzahligen (diskreten) Raum ablaufen sowie gemischte Probleme. Im Folgenden wird die Modellformulierung für alle Räume besprochen.

Kontinuierliche Optimierung

Es wurde bereits das Modell für die kontinuierliche Optimierung besprochen siehe Gleichung 2.3. Wird im Folgenden von kontinuierlichen Problemen gesprochen, wird ein $_k$ an die Bezeichnungen angehängt.

Diskrete Optimierung

Darf ein Variablenvektor nur Werte einer diskreten Wertemenge annehmen, spricht man von einer diskreten Optimierung. Als Spezialfälle lassen sich eine ganzzahlige sowie eine binäre Optimierung ableiten, siehe unten. Die Definition von kontinuierlichen und diskreten Optimierungsproblem sind gleich:

$$\min\{ y(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}_D \} \quad (2.4)$$

Dabei stammt der zulässige Bereich \mathbf{X}_D aus der Menge diskreter Werte:

$$\mathbf{X}_D = \{ \mathbf{x} \in \mathbb{R}^n \mid x_i \in \mathbf{D}_i, \quad i = 1, 2, \dots, m; \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \dots, n \}$$

Für den diskreten Charakter sorgt die Auswahl von m Wertemengen aus \mathbb{R} :

$$\mathbf{D}_i = \{x_{i1}, x_{i2} \dots x_{id_i}\}, \quad \mathbf{D}_i \subset \mathbb{R}, \quad i = 1, 2, \dots, m;$$

Spezialfall - Ganzzahlige Optimierung

Der Spezialfall einer diskreten Optimierung ist die ganzzahlige

$$\min\{ y(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}_Z \} \quad (2.5)$$

$$\mathbf{X}_Z = \{ \mathbf{x} \in \mathbb{Z}^n \mid x_i \in \mathbf{Z}_i, \quad i = 1, 2, \dots, m; \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \dots, n \}$$

Dadurch entstehen die m Wertemengen:

$$\mathbf{Z}_i = \{-d_1^-, \dots, -1, 0, +1, \dots d_i^+\}, \quad \mathbf{Z}_i \subset \mathbb{Z}, \quad i = 1, 2, \dots, m;$$

Ein weiterer Spezialfall ist die binäre Optimierung. Die Erweiterung ist für diesen Fall trivial und wird hier nicht gezeigt.

Gemischte Optimierung

Die auch als diskret-kontinuierliche Optimierung bekannte Methode formuliert das Modell für einen gemischten Variablenvektor. Man kann in einem solchen Fall den Variablenvektor aus zwei Teilen zusammensetzen:

$$\mathbf{x} = [\mathbf{x}_K \ \mathbf{x}_D]^T$$

Die Notation meint mit

- \mathbf{x}_K den Vektor der m_K kont. Variablen und mit
- \mathbf{x}_D den Vektor der m_D disk. Variablen.

Analog dazu lässt sich der Suchraum in zwei Teile Zerlegen:

$$\mathbf{x}_K \in \mathbf{X}_K \quad \text{und} \quad \mathbf{x}_D \in \mathbf{X}_D$$

Nun lautet das disk.-kont. Optimierungsproblem wie folgt:

$$\min\{ y(\mathbf{x}) \mid \mathbf{x}_K \in \mathbf{X}_K, \mathbf{x}_D \in \mathbf{X}_D \} \quad (2.6)$$

Diskret-kontinuierliche Probleme sind nur schlecht lösbar. In dieser Arbeit wird ein rein reellwertiges Problem behandelt.

2.4. Evolutionäre Strategien

Folgende Informationen entstammen im Wesentlichen aus [18, Kapitel 7],[4] sowie [10] und sind auf den folgenden Seiten zusammengefasst und neu arrangiert. Das erleichtert die Einarbeitung in die Thematik und das Verständnis der Arbeit. Evolutionsstrategien nach natürlichem Vorbild gehen auf die Arbeiten von RECHENBERG [21] und SCHWEFEL [24] zurück. Ihr Ansatz war von Anfang an für die Optimierung gedacht, anders als z.B. Evolutionäre Programmierung (EP). Im Gegensatz zu Genetischen Algorithmen spielt die Repräsentationsform der Daten keine Rolle. Der Variablenvektor kann sowohl diskrete als auch kontinuierliche Komponenten haben.

2.4.1. Evolutionsstrategien - Grundlagen

Nach dem Vorbild natürlicher Evolution entworfene stochastische Optimierungsverfahren werden als Evolutionsstrategie bezeichnet. Sie verwenden die Prinzipien der Mutation, Rekombination und Selektion analog zu der natürlichen Evolution. Die Vorgänge werden dabei abstrahiert und vereinfacht implementiert, bilden die Prozesse der Evolution jedoch nach. Durch ihre Allgemeingültigkeit haben diese Verfahren ein großes Anwendungsspektrum

Abbildung 2.2.: Klassifizierung von nicht linearen Optimierungsstrategien.

Die evolutionären Strategien gehören zu den ableitungs-freien Verfahren. Diese grenzen sich zu anderen Verfahren ab, da sie keine stetige Differenzierbarkeit erwarten. Das ist ein Vorteil, da damit praktisch alle Probleme gelöst werden können.



in Wissenschaft und Industrie. Einige Anwendungsmöglichkeiten und Beispiele sind in [18, Kapitel 11] zu finden.

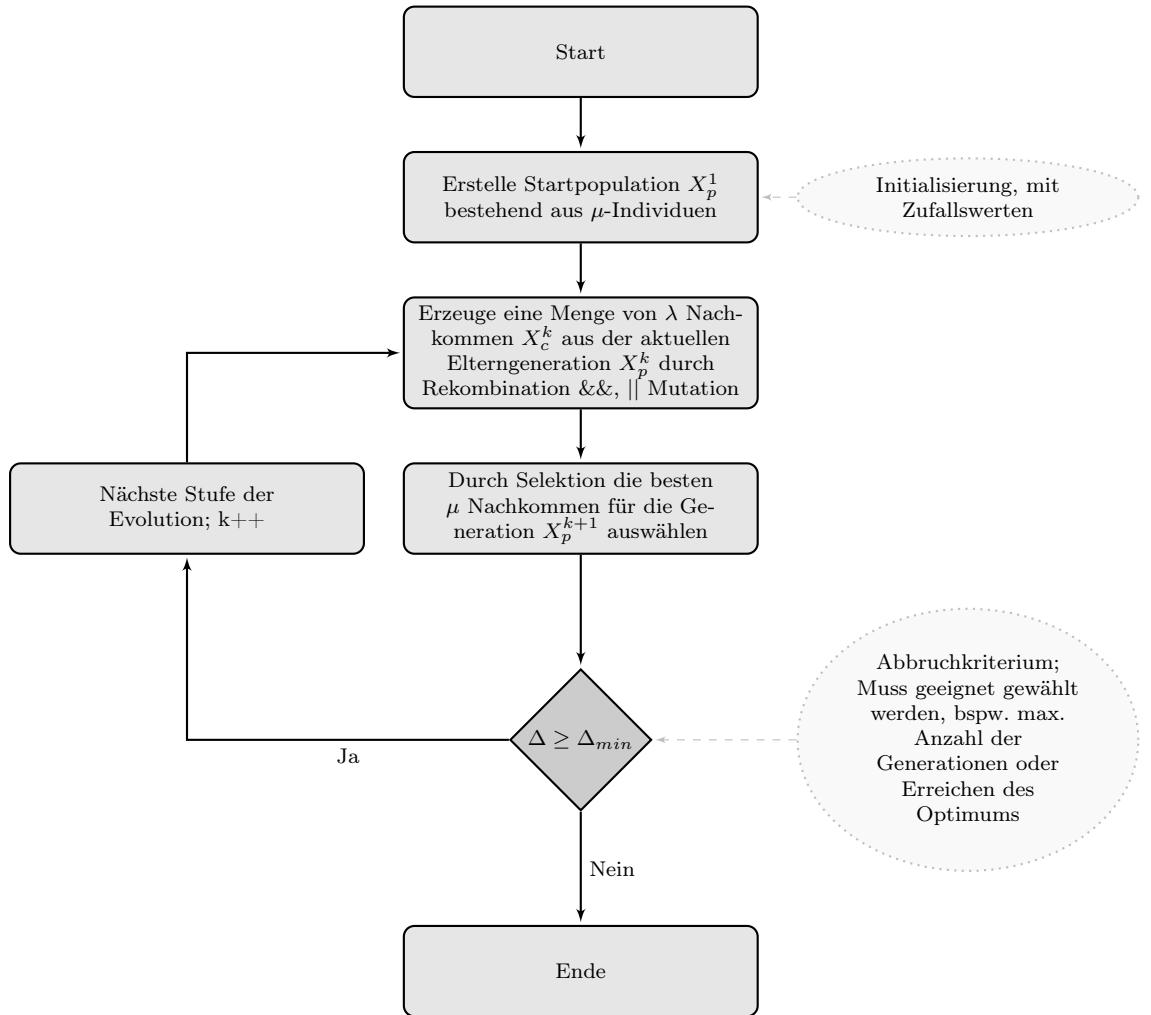
Die Abstrahierung der Prozesse verwendet dabei die selbe Nomenklatur, wie im biologischen Kontext üblich. Dabei bezeichnet im Folgenden:

- μ die Anzahl der Eltern (\Rightarrow Größe der Population)
- λ die Anzahl der Eltern die bei Rekombination neue Kinder erzeugt; Die Anzahl der erzeugten Nachkommen einer neuen Generation⁴
- \mathbf{x}_p Eltern (Parent)
- \mathbf{x}_c Nachkommen einer Generation (Child)
- X_p^1 Die Menge aller Eltern der ersten Generation $X_p = \{\mathbf{x}_{p_1}^1, \dots, \mathbf{x}_{p_\mu}^1\}$
- X_p^k Die Menge aller Eltern der k-ten Generation $X_p = \{\mathbf{x}_{p_1}^k, \dots, \mathbf{x}_{p_\mu}^k\}$

⁴Anmerkung: Die Verwendung des Symbols λ ist in diesem Kontext nicht eindeutig. Im Rahmen dieser Arbeit steht dieses Symbol auch für die Wellenlänge. In diesem Abschnitt wird jedoch weiterhin λ verwendet um die gleiche Nomenklatur wie bei dieser Thematik üblich zu verwenden.

Wir wollen nun in Abbildung 2.3 einen Blick auf den prinzipiellen Ablauf dieses Algorithmus werfen und anschließend auf die Details eingehen.

Abbildung 2.3.: Der Ablauf des (λ, μ) -Evolutionsalgorithmus ist in dieser Abbildung gezeigt. Dies ist eine leicht zu verstehende Variante der Algorithmen. Die wesentlichen Schritte gleichen sich in den Varianten.



Mutation

Ein Nachkomme \mathbf{x}_C wird aus seinem Elternteil \mathbf{x}_P und einer zufälligen Variation \mathbf{d} gebildet.

$$\mathbf{x}_c = \mathbf{x}_P + \mathbf{d} \quad (2.7)$$

Dabei ist \mathbf{d} ein bei jeder Mutation neu zu bestimmender $(0, \sigma^2)$ -normalverteilte Zufallszahl $Z(0, \sigma^2)$:

$$\mathbf{d} = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} Z(0, \sigma_1^2) \\ \vdots \\ Z(0, \sigma_n^2) \end{pmatrix} = \begin{pmatrix} Z(0, 1)\sigma_1 \\ \vdots \\ Z(0, 1)\sigma_n \end{pmatrix} \quad (2.8)$$

Die Normalverteilung der Variation ist nützlich, da kleine Änderungen wahrscheinlicher sind als Große. Die maximale Größe der Variation wird durch die Standardabweichung σ_i bestimmt. Sie steuert somit die Schrittweite von Generation zu Generation.

Rekombination

Durch Rekombination zweier oder mehr Eltern aus der Menge aller μ -Eltern $X_\varrho \subset X_E$. Die Wahl der Eltern sollte zufällig erfolgen um *Inzuchtprobleme*⁵ zu verhindern.

Zwei Arten der Rekombination sind denkbar:

Die *intermediär Rekombination* erstellt einen Nachkommen durch das gewichtete Mittel von ϱ Eltern.

$$\mathbf{x}_c = \sum_{i=1}^{\varrho} \alpha_i \mathbf{x}_{p_i}, \sum_{i=1}^{\varrho} \alpha_i = 1, 2 \leq \varrho \leq \mu \quad (2.9)$$

Bei der *diskreten Rekombination* vom ϱ -Eltern wird die i -te Komponente x_{ic} eines Nachkommen \mathbf{x}_c mit der i -te Komponente eines zufällig gewählten Elternpunktes gleichgesetzt.

$$\mathbf{x}_{ic} = \mathbf{x}_{ip_j}, j \in \{1, \dots, \varrho\}, i = 1, \dots, n \quad (2.10)$$

Selektion

Die durch Rekombination und/oder Mutation erzeugten Nachkommen werden in dem Schritt ausgewählt um einen Evolutionsfortschritt zu erreichen. Dies erfolgt anhand des Vergleichs mit dem Zielfunktionswert $f(\mathbf{x})$. Das beste Individuum, bzw. die besten, werden für die nachfolgende Generation ausgewählt. Dabei gibt es Strategien, bei denen zum einen nur die Nachkommen an der Auswahl beteiligt sind oder Andere bei denen Eltern und Kinder teilnehmen.

Evolutionsalgorithmus

Der eigentliche Evolutionsalgorithmus ist in Abbildung 2.3 dargestellt. Er enthält im Wesentlichen die in den vorherigen Abschnitten beschriebenen

⁵d.h. sich schlechte Ergebnisse weiter durchsetzen

Schritte. Der prinzipielle Ablauf ist für alle Evolutionsalgorithmen gleich. Eine Unterscheidung der Verfahren kann durch verschiedene Parameter beschrieben werden. Wesentlich dabei sind die Populationsgröße μ , die Anzahl an der Rekombination beteiligten Eltern ϱ , die gewählte Selektionsstrategie sowie die Anzahl der Nachkommen λ . Im Folgenden sind zuerst einige Beispiele für die Nomenklatur der Selektionsstrategie aufgeführt, die im Anschluss genauer beschrieben werden.

Für Strategien die nur auf Mutation für die Erzeugung von Nachkommen setzten sind folgende Nomenklaturen gebräuchlich:

- $(\mu + \lambda)$ Elternelemente werden in der Selektion berücksichtigt
- (μ, λ) Ausschließlich Nachkommen nehmen an der Selektion teil

Die Strategien werden Plus- bzw. Komma-Strategie genannt. bei der Plus-Strategie wird zusätzlich noch ein Gewichtungsfaktor benötigt, der das *Alt*ern der Elterngeneration darstellt. Dieser Mechanismus soll verhindern, dass die Eltern für alle Zeiten überlegen. Somit werden sie nach einer gewissen Anzahl an Generationen, nicht mehr berücksichtigt werden. Sie sind zu *alt*. Wird die Rekombination eingesetzt kann auch die Anzahl der beteiligten Elternelemente angegeben werden:

- $(\mu/\varrho + \lambda) \& (\mu/\varrho, \lambda)$ Angabe der Anzahl beteiligter Eltern bei der Rekombination.

Mit Hilfe der hier beschriebenen Klassifikationen werden die Algorithmen im Folgenden stets angegeben. In Abbildung 2.3 wird der Ablauf einer Optimierung mit evolutionären Verfahren dargestellt. Es wird die Komma-Strategie gezeigt, ein Struktogramm der Plus-, oder anderer Strategien ist nicht gezeigt. Die Unterschiede würden sich in dem Punkt Rekombination zeigen.

2.4.2. Strategien mit mehreren Populationen

Es ist möglich die Strategien auf die Ebene von Populationen zu erweitern. Das bedeutet, man lässt ganze Populationen miteinander in Wettstreit treten und nur diejenigen überleben, die die besten Ergebnisse liefern. Das mündet in einem zweistufigen Evolutionsprozess. Man kann die Notation um diesen Umstand erweitern und erhält so:

$$[\mu_2/\varrho_2,^+ \lambda_2(\mu_1/\varrho_1,^+ \lambda_1)]$$

Sprich aus μ_2 -Elternpopulationen werden durch Rekombination mit jeweils ϱ_2 Populationen, λ_2 Nachkommenpopulationen generiert. Innerhalb der Populationen erfolgt die Optimierung anhand einer $(\mu_1/\varrho_1 + \lambda_1)$ oder $(\mu_1/\varrho_1, \lambda_1)$ -Strategie. Nun können nach einer bestimmten Zahl von Generationen die

besten Populationen für die nächste Generation ausgewählt werden. Auch hier stehen verschiedene Auswahlkriterien zur Verfügung. Man kann z.B. die Population anhand des Zielfunktionswert des besten Individuums wählen oder den Mittelwert über alle Individuen wählen.

Nicht behandelte Themen

Das Gebiet der evolutionären Optimierung/- Algorithmen ist groß. Daher wurden in diesem Abschnitt nur die für das Verständnis unbedingt notwendigen Grundlagen erläutert. Eine Reihe von Themen, die in der Evolutionären eine Rolle spielen, wurde ausgelassen. Beispielsweise:

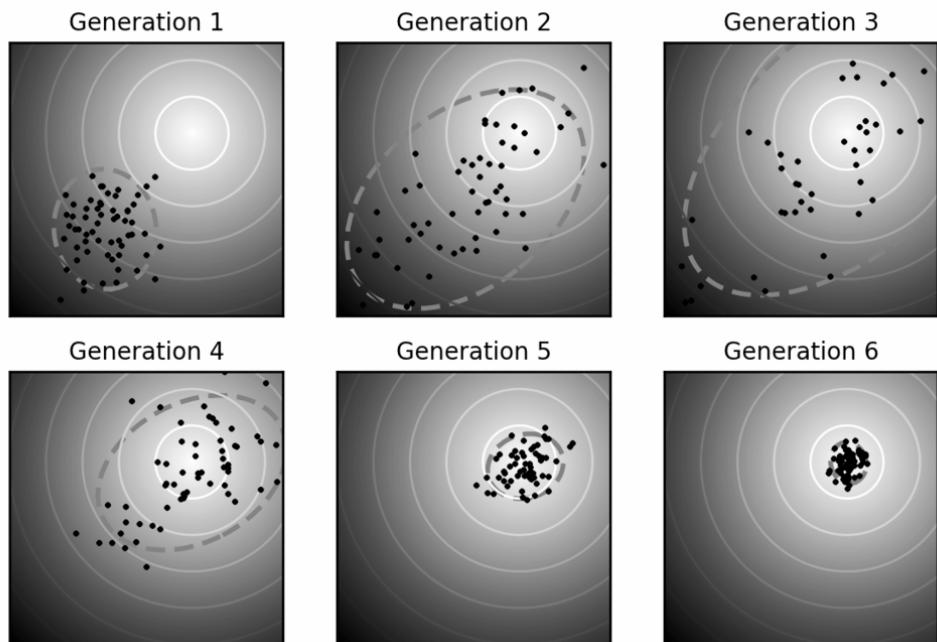
- Schrittweitensteuerung
- Diskrete Optimierungsprobleme/ Variationsmechanismen
- Neustart mit anderer Populationsgröße
- Behandlung von Nebenbedingungen

2.5. Covariance Matrix Adaption - Anpassung der Kovarianzmatrix

Das CMA-ES-Verfahren. Es stellt den "State of the Art"- der Evolutionären Berechnungsverfahren dar. Für spezialisierte Probleme gibt es zwar bessere Lösungen, als allgemeiner Solver für unregistrierte, kontinuierliche Probleme ist dieses Verfahren mehr als tauglich. Es wurde um die Jahrtausendwende entwickelt und veröffentlicht. Typische Einsatzgebiete der CMA-ES sind unregistrierte Optimierungsprobleme mit einem Suchraum zwischen drei und einhundert Dimensionen. Der Algorithmus kann eingesetzt werden wenn auf Ableitung angewiesene Verfahren (z.B. Newtonverfahren) aufgrund zerklüftete Suchräume, Rauschen, viele lokale Optima oder Ausreißern zu Fehlern neigen. Die Methode ist geeignet für nicht separierbare, schlecht konditionierte Probleme. Das Verfahren benötigt keine Gradienten, oder erfordert ihre Existenz. Dadurch kann der Algorithmus auch auf nicht stetige Probleme angewandt werden. Es existieren verschiedene Abwandlungen des Algorithmus, die für spezielle Fragestellungen geeigneter sind, und sogar eine Lösung zur Multiobjekt-Optimierung (MO-CMA-ES) wurde beschrieben [12]. Das Verfahren wird aktuell stets weiterentwickelt [8, 7].

Der Algorithmus steht in verschiedene Implementationen, Programmiersprachen und Umgebungen zur Verfügung [11]. Teilweise sind die Implementierungen proprietär (z.B. Matlab), teilweise quelloffen. Die in dieser Arbeit zur Anwendung kommende Umsetzung ist die Shark-Library. Diese Bibliothek ist eine in *C++* geschriebene, quoelloffene Software, die am Institut für

Abbildung 2.4.: Die Abbildung zeigt sechs Lösungsschritte des CMA-ES-Verfahrens. Eingangs wird eine Population mit einer Gauß-Verteilung generiert. Diese bewegt sich in jeder Generation näher an das globale Optimum heran. Die gestrichelte Linie zeigt dabei eine Isolinie der Wahrscheinlichkeitsdichte. Wir befinden uns auf einem 2D-Problem, somit wird das vorankommen der Population über zwei σ 's gesteuert. Dadurch kommt die Ellipse zu Stande. Die Linie bedeutet nicht, dass nur in diesem Bereich Nachkommen erzeugt werden, siehe z.B. Generation 4. Je nach Beschaffenheit des Problems und nach Nähe zum Optimum werden die steuernden σ 's kleiner. [27]



Neuroinformatik der Ruhr Universität Bochum entwickelt wird. Detailliert wird Shark im Rahmen des Hauptteils in Abschnitt 3.8.1 vorgestellt.

Auf die Darstellung der mathematischen Grundlagen des Algorithmus wird hier verzichtet, da diese nicht zum Verständnis der Arbeit beitragen. Die CMA-ES ist ein Aufsatz auf der $(\mu + \lambda)$ -Strategie, bei der durch die Anpassung der Kovarianz-Matrix eine Abschätzung der Konturlinien der Objektfunktion $f(\mathbf{x})$ gefunden wird. Das Verfahren wurde in vielen Fragestellungen angewendet, eine unvollständige, etwas veraltete Übersicht ist unter [9] zu finden.

2.6. Technisch-Physikalische Voraussetzungen

In diesem Abschnitt werden die technischen und physikalischen Grundlagen für diese Arbeit vorgestellt und das Wichtigste erörtert. Es wird auf die Besonderheiten und Merkmale des auf Funk basierenden RFID-Verfahrens eingegangen. Die andere Trackingverfahren werden, aufgrund der Unterschiedlichkeit der Systeme wird im Rahmen dieser Arbeit nicht weiter behandelt. Es kann nicht im vollem Umfang auf die Details der Technik eingegangen werden ohne den Rahmen dieser Arbeit zu sprengen. Interessierte sei die referenzierte Literatur für eine weite Lektüre empfohlen.

2.6.1. Positionsgenauigkeit auf Funk basierender Verfahren

Die Positionsgenauigkeit eines auf EM basierenden Systems ist von dem Messprinzip abhängig. Dabei bieten sich im Wesentlichen drei Möglichkeiten:

- | | |
|--|--|
| 1. Laufzeitmessung
2. Messung der Signalstärke
3. Phasendifferenzmessung | TOF
RSSI
PD |
|--|--|

Eine Laufzeitmessung des Signals kommt aufgrund der Ausbreitungsgeschwindigkeit der EM-Welle nicht in Frage, da diese typischerweise gleich der Lichtgeschwindigkeit ist und die Distanz zwischen Sender und Empfänger zu gering ist. Das reduziert die Möglichkeiten auf zwei Verfahren. Bei der RSSI wird die Stärke des empfangenen Signals ausgewertet. Dies stellt eine einfache Art der Positionsermittlung dar. Jedoch kann die Signalstärke stark schwanken und erlaubt nur eine geringe Ortsauflösung. Bei der PD wird die Position anhand der zurückgestrahlten Welle ermittelt, genauer der Phase der Welle. Das Auflösungsvermögen dieser Methode kann sehr gute Werte erreichen. Ist die Messelektronik Empfindlich genug, lassen sich leicht Ortsauflösungen von $\leq 1 \text{ mm}$ erreichen. Zusätzlich ist es für die Ortsauflösung von Vorteil eine möglichst hohe Messfrequenz zu verwenden, dazu betrachten wir ein Beispiel. Sei a die Empfindlichkeit des Messelektronik in [$^{\circ}$] und λ die Wellenlänge in [m], die Ortsauflösung \tilde{x} kann man angeben mit:

$$\tilde{x} = \frac{a}{360^{\circ}} \times \lambda$$

Bereits bei $\lambda = 0.35 \text{ m}$ und $a = 1^{\circ}$ beträgt die Ortsauflösung:

$$\tilde{x} = \frac{1^{\circ}}{360^{\circ}} \times 0.35 \text{ m} = 9.722^{-4} \text{ m}$$

Die angenommenen Werte sind in der Realität gut zu erreichen. Setzt man nun $\lambda = \frac{c_0}{f}$, wobei c_0 die Ausbreitungsgeschwindigkeit vom EM-Wellen im Vakuum und f die Frequenz der Welle meint. Durch

$$\tilde{x} = \frac{1^\circ}{360^\circ} \times \frac{c_0}{f} = \frac{c_0}{360^\circ} \times \frac{1}{f} = \text{const.} \times \frac{1}{f}$$

erkennt man, dass sich $\tilde{x} \sim \frac{1}{f}$ verhält.

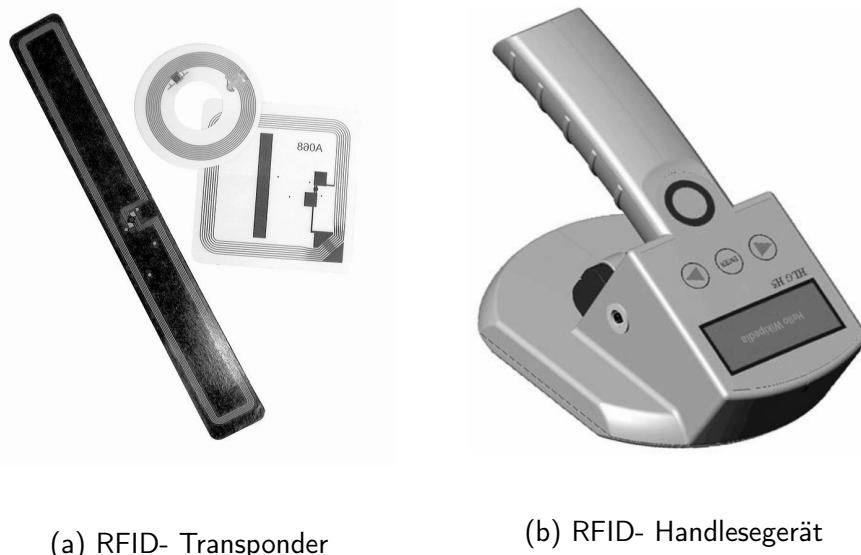
2.6.2. RFID

Bei *Radio-Frequency Identification* (RFID) handelt es sich um einen Funkstandard der die kontaktlose Identifikation bei gleichzeitiger Erfassung zusätzlicher Informationen ermöglicht (Payload). Zur Technik gehört ein Auslesegerät (Reader) und ein oder mehrere Transponder (Tags). Eine sehr grobe Übersicht über typische Bauformen von Tags und Reader ist in 2.5 zu finden. Die dargestellten Tags sind für verschiedene Frequenzbänder. Heute verfügbare Transponder lassen sich auf nahezu jeder beliebigen Oberfläche anbringen. Das ermöglicht ein großes Anwendungsspektrum, praktisch wird die Technik in jeder Umgebung eingesetzt in der es erforderlich oder nützlich ist Dinge kontaktlos zu identifizieren.

Im Rahmen dieser Arbeit wird kein umfassender Überblick über die Technik geboten, da die Bauformen und Spezifikationen sehr stark variieren. Ein umfassendes Werk, das eine gute Einführung und Übersicht zur Technik bietet ist das Standardwerk FINKENZELLER [6]. Dort werden detailliert die physikalischen Grundlagen verschiedener Antennenbauformen und Tags erläutert. Eine gute Übersicht über Branchen und Anwendungsgebiete für RFID bietet [22]. Aufgrund des großen Anwendungsspektrums und der weiten Verbreitung ist die Technik in die Kritik geraten. Unter dem Dach des Vereins *digitalcourage e. V.* existiert die Kampagne *StopRFID*. Die Kampagne hat sich zum Thema gemacht über die Anwendungsmöglichkeiten und Risiken von RFID aufzuklären [14]. Die Webseiten der Kampagne bieten eine sehr weitgehende Auflistung der Anwendungen für RFID.

In der EU sind verschiedene RFID-Frequenzen zulässig, sie reichen von 865,0 MHz bis 868,0 MHz [5]. Man kann damit die Wellenlänge mit: $\lambda \simeq 0,35m$ angeben. Daraus folgt, dass alle 35 cm die gleiche Konfiguration der Phase vorliegt. Man spricht auch von *Isophasen*. Daraus folgt, dass die gewonnene Information aus der Phase nicht eindeutig ist. Das bedeutet es lässt sich durch die Kenntnis der Phase nicht unmittelbar auf die korrekte Position des Tags schließen. Man kann das Problem umgehen in dem man auf die errechnete Position ein ganzzahliges Vielfaches der Wellenlänge, die Wellenzahl, hinzufügt.

Abbildung 2.5.: Hier gezeigt sind Beispiele für RFID Transponder und Lesegeräte. Das linke Bild zeigt drei typische Tags, nahezu jede Gestalt ist mittlerweile erhältlich. Die hier gezeigten Tags eignen sich für eine Anbringung an glatten Oberflächen. Es gibt zig weitere Bauformen, die unterschiedlichste Anwendungsspektren bedienen und sogar eine Implantation ermöglichen (nicht gezeigt). Im rechten Bild ist ein Handlesegerät gezeigt. Zum Mobilen Auslesen über mittlere bis kurze Distanzen. Auch bei den Readern gibt es unterschiedlichste Bauformen, die je nach Anwendungsfall ausgewählt werden.



Das System der amedo STS verwendet eine spezielle Antennenanordnung um die Position zu ermitteln. Dabei wird eine Antennenanzahl >4 eingesetzt. Für jede dieser Antennen muss eine eigene Wellenzahl bestimmt werden. Durch Auslöschung des Signals, Absorption etc. kann es dazu kommen, dass eine Antenne eine unbestimmte Zeit lang kein Signal vom Tag empfängt. Wenn die Antenne nach dieser Zeit erneut ein Signal empfängt ist die ihr zugehörige Wellenzahl unbekannt und muss neu bestimmt werden.

In realen Umgebungen treten zusätzlich noch Reflexionen auf, durch die ein sog. Multipath-Effekt entsteht. Dabei kann sein Signal den Reader auch über einen nicht direkten Weg erreichen und z.B. von Wänden oder Decken zurück auf die Antenne treffen. Das Signal wird nicht auf dem direkten Weg (Antenne->Tag->Antenne) empfangen sondern über einen unbekannten, längeren Weg. Darüber hinaus kann es in einer solchen Umgebung zu Auslöschen und Verstärkungen kommen. Man stelle sich dazu ein Becken

voller Wasser vor, genauer die Wellenausbreitung. Die Polarisation der Antennen begünstigt Auslöschung und Verstärkung zusätzlich. Es kommt es zu einem Fehler in der Phase und zu einer falschen Positions berechnung. Die Fehlerbetrachtung ist nicht Gegenstand dieser Arbeit und wurde in anderen Arbeiten bereits behandelt z.B. [2].

2.6.3. PRPS-Messsystem

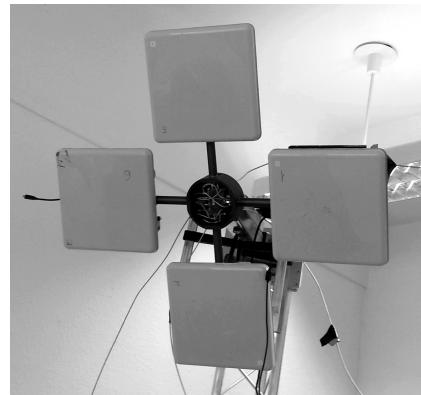
Das in dieser Arbeit entwickelte Verfahren wird für das Messsystem der amedo STS entwickelt. Die Entwicklung wird Teil der Softwarekomponenten des Systems werden. Aktuelle befindet sich das gesamte System in der Revision, in diesem Abschnitt wird der Stand der Entwicklung dargestellt. Details über Hardware und Software werden im Rahmen der Arbeit nicht besprochen, sofern sie diese Arbeit nicht unmittelbar betreffen.

Das auf RFID basierende PRPS (Passiv RFID Positioning System) besteht aus mehreren frei positionierbaren Antennen, einer Mess- und Steuereinheit, sowie einem Rechner zur Kommunikation mit Endkundensoftware. Die Systemkomponenten für die Messwertaufnahme und Steuerung sind in einem Gehäuse untergebracht, dieses zeigt Abbildung 2.7. Eine typische Installation ist in Abbildung 2.8 gezeigt. Dort wurde ein System mit insgesamt acht Antennen installiert. Vier Antennen sind frei aufgestellt, vier weitere in einer festen Anordnung (*Spinne*) installiert. Der Aufbau ist auf ein Messvolumina gerichtet.

Leistungsmerkmale

Das PRPS erlaubt eine Identifikation mehrerer Objekte oder genauer: RFID-Tags, die auf den Objekten angebracht sind. Wird ein einzelner Tag ausgewählt, kann seine Position mit einer sehr hohen Frequenz ausgegeben werden. Die momentan Erreichbaren Werte liegen bei 60 Hz. Je nach Umgebung kann dieser Wert jedoch variieren, siehe Kapitel 3.1. Sollen mehrere Tags ausgegeben werden, verringert sich die Frequenz entsprechend. Bei der Verwendung von drei Tags ist eine Leserate von 20 Hz pro Tag realistisch. Die Positionsgenauigkeit liegt aktuell bei ≈ 5 mm. Die Antennen können frei

Abbildung 2.7.: Abgebildet ist der Messaufbau aus vier Antennen. In dem Aufbau verbaut sind die wesentlichen elektronischen Komponenten wie Auswerte- und Steuereinheit. Nicht einzeln gezeigt.



Arrangiert werden. Das erlaubt eine Anpassung an nahezu jeden Raum und jeden Kundenbedarf. Als Tags kommen alle handelsüblichen RFID-Tags im verwendeten Frequenzband im Bereich der ETSI-Frequenzen [5] in Frage. Der Messbereich liegt bei mehreren Kubikmetern und einer erprobten maximalen Entfernung von 6 – 7 Metern. Theoretisch ist das maximale Volumen noch größer. Die leistungstarke Messeinheit besteht aus einer sehr flexiblen Messelektronik. Diese ist in der Lage eine sehr hohe Güte und Frequenz der Messwerte zu realisieren.

Abbildung 2.8.: Abgebildet ist der Messaufbau mit unterschiedlichen Antennen. Der Aufbau ist auf ein Messvolumina von mehreren Kubikmetern ausgerichtet



2.6.4. Phase und Wellenzahl

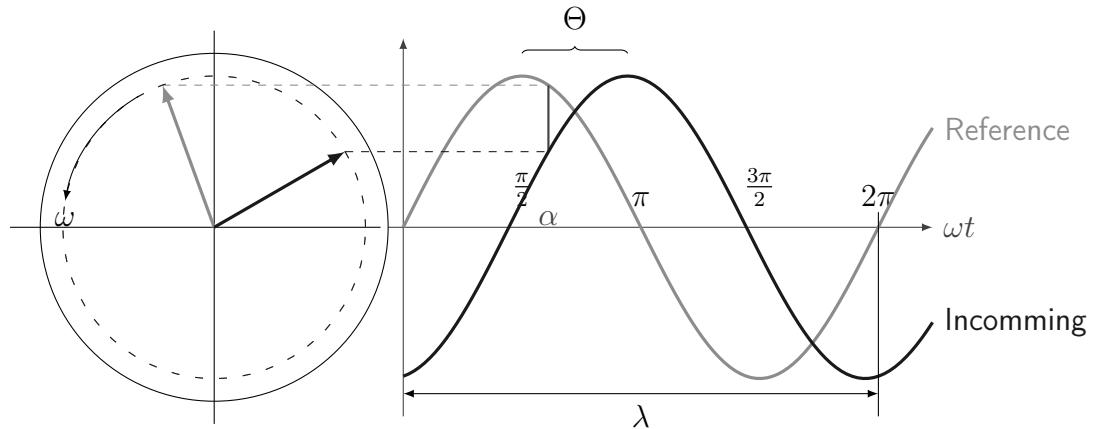
Aus der Abbildung 2.9 lässt sich folgender Zusammenhang ableiten.

$$d(\Theta, n) = \lambda(\Theta/2\pi + n) \quad (2.11)$$

Berechnung idealer Phasenwerte

Die Gleichung 2.11 kann dazu verwendet werden künstliche Messdaten zu generieren. Dazu muss zuerst die Wellenzahl aus der bekannten Entfernung

Abbildung 2.9.: Dargestellt ist der Zusammenhang zwischen der Wellenlänge λ und der Wellenzahl n . Da die Phase alle 2π den gleichen Wert annimmt, wird mit dem Faktor n ein Vielfaches der Wellenlänge aufaddiert. Dadurch erhält man die Entfernung zu dem Tag.



und der Wellenlänge ermittelt werden.

$$n_k = \frac{d_k}{\lambda}$$

Nachdem die Wellenzahl für Antenne k ermittelt wurde kann man das jeweilige Θ bestimmen.

$$\Theta = 2\pi \left(\frac{d_k}{\lambda} - n \right)$$

Die Berechnung idealer Messwerte wird im späteren Kapiteln verwendet um die Ergebnisse der Modelle mit echten Messdaten zu verifizieren.

3. Hauptteil

Im Folgenden werden ausführlich die Methoden und Lösungen zur beschriebenen Problemstellung vorgestellt. Zuerst wird eine Betrachtung der Komplexität des Problems präsentiert. Es werden die Modelle vorgestellt die zum Auffinden der Lösung verwendet wurden. Im Anschluss wird die Implementation der evolutionären Optimierung und die Integration in die PRPS-Software beschrieben.

3.1. Vorüberlegung zur Komplexität

In diesem Abschnitt wird eine Betrachtung der Komplexität des Problems gegeben. Bereits in Kapitel 2.6 wurde auf die Komplexität eingegangen. Hier werden Erkenntnisse präsentiert, die auf das Basis realer Messdaten ein Gefühl für die Komplexität geben sollen. Später, in Abschnitt 3.6, wird das entwickelte Modell analysiert, dazu sind die Einsichten aus diesem Abschnitt eine Orientierungshilfe.

Abbildung 3.1 zeigt die Visualisierung einer typischen Kalibriermessung. Der verwendete Aufbau ist in Abbildung A.1. gezeigt. Er besteht aus vier Antennen, die in einer Ebene angeordnet sind. Es wurde eine reproduzierbare Aufstellung verwendet (Abbildung A.2) und eine Fläche von 1×1 Meter vermessen. Alle 10 cm wurde eine Messung gespeichert. In der Abbildung kann man deutlich das Verhalten der Phasendaten sehen. Um diesen Verlauf deutlicher zu zeigen, wurden die Phasenwerte normiert und als Oberfläche in den Plot gelegt. Am Boden gezeigt ist der Kontur-Plot der Werte. Zwischen den Werten wurde interpoliert um die Nulldurchgänge deutlicher zu zeigen. Die Übersicht aus der Sicht aller Antennen ist in Abbildung 3.2 gezeigt.

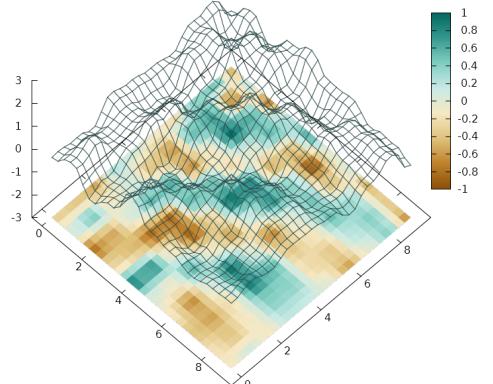


Abbildung 3.1.: Normiertes Höhenprofil einer Phasenmessung aus der Sicht von Antenne 1

In der Abbildung 3.3 werden die Daten ohne Interpolation dargestellt. Es wurden die Höhenlinien eingezeichnet. Die Anordnung der Plots soll ein Gefühl dafür vermitteln, wie die Messwerte eines Tags sich an unterschiedlichen Positionen und aus Sicht verschiedener Antennen verhalten.

Die Darstellung echter Messwerte lässt Rückschlüsse auf die Komplexität des Problems zu. Es ist leicht nachzuvollziehen, dass das Zusammenspiel der Messwerte eine sehr komplexe Szene ergibt. Hier dargestellt ist bereits das Verhalten bei der Verwendung von vier Antennen. Der aktuelle Messaufbau erlaubt sogar acht Antennen. Das ergibt insgesamt eine komplexe Szenerie.

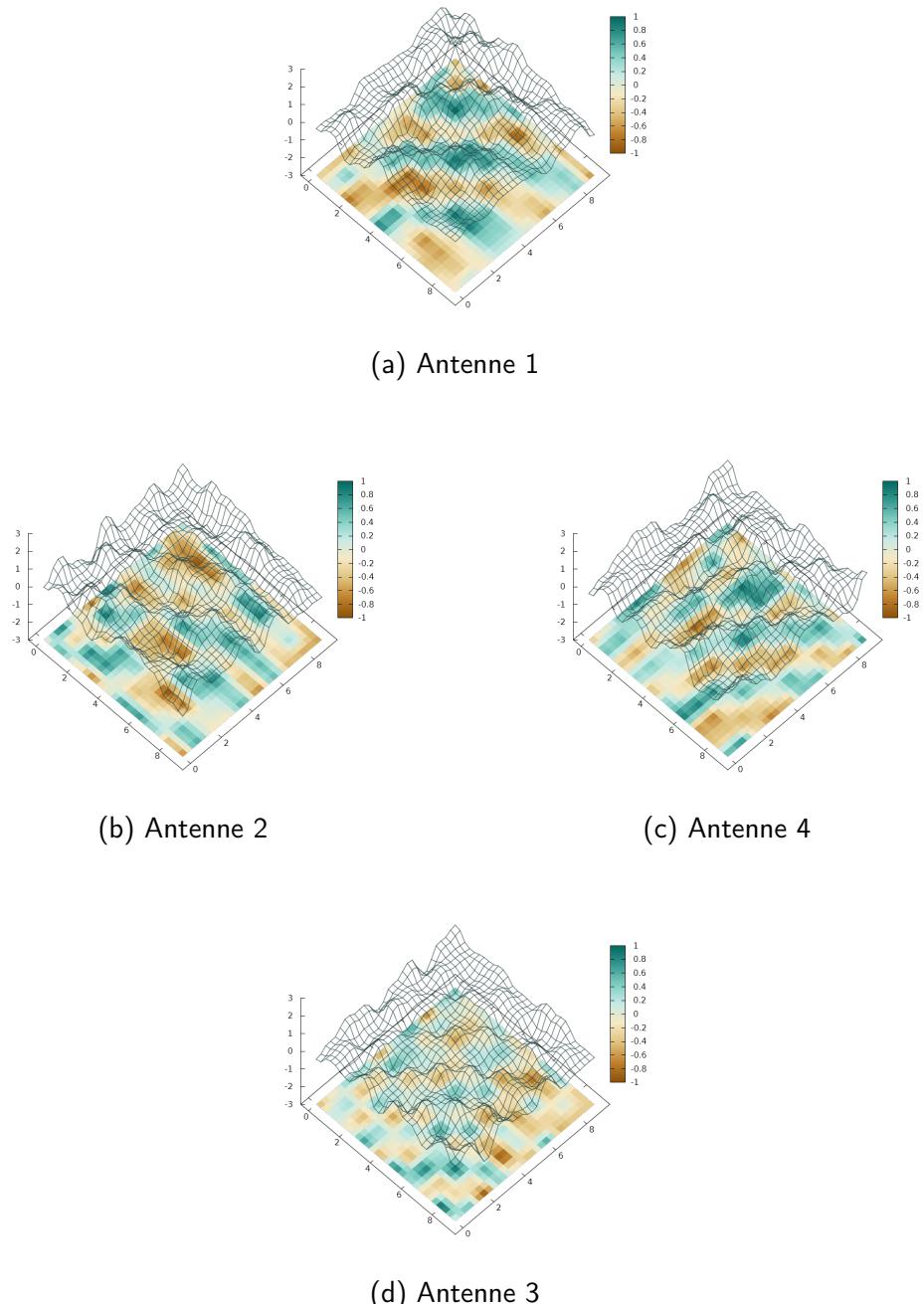


Abbildung 3.2.: Blick auf die Messwerte der Kalibrierplatte aus der "Sicht" der Antennen. Dabei zeigt sich deutlich der Wellencharakter der Messung, dieser ist zu erwarten. Die Messungen wurden mit einer Frequenz von 865,7 MHz unter Laborbedingungen aufgenommen.



Abbildung 3.3.: Diese Grafik zeigt die Visualisierung von realen Phasen-Messwerten. Die Daten wurden durch Vermessung einer 1×1 -Kalibrierplatte mit reproduzierbarer Aufstellung gewonnen. Die Daten wurden normiert. In jeder Dimension wurden 10×10 Werte aufgenommen. Die Darstellung der Phasenwerte erfolgt als Heatmap, es soll qualitativ der Verlauf der Phasenwerte gezeigt werden. Zur Orientierung sind in jedem Plot Höhenlinien eingezeichnet. Pro Plot werden die Daten einer Antenne dargestellt. Die Antenne von der die Daten stammen ist angegeben.

3.2. Entwicklung des Modells

Im folgenden Abschnitt wird das Modell für die Lösung des Zusammenhangs entwickelt. Zur Veranschaulichung des Sachverhalts dient die Abbildung 3.4. Dort skizziert ist der Messaufbau mit einem Tag. Die Szene ist in 2D dargestellt, die Ableitung des Modells erfolgt direkt für drei Raumkoordinaten. Folgende Nomenklatur und Symbole gelten für diesen Abschnitt:

- $r_k :=$ Abstand vom Tag zur Antenne
- $d_{kJ} :=$ Abstand zur Landmarke
- $N_0 :=$ Menge der verfügbaren Antennen $N = \{1, \dots, 8\}$
- $N :=$ Menge der Antennen die verfügbar sind¹ ($N \subseteq N_0$)
- $N' :=$ Menge der Antennen für die Optimierung ($N' \subseteq N$)
- j ist der Index der Referenzantenne, es gilt $j = \{1, 2, \dots, 8\}$
- k ist der Index der Antennen einer Messung, es gilt $k = 1, 2, \dots, |N'| - 1$

Wir starten mit der Überlegung über den geometrischen Zusammenhang zwischen der Antennenposition von Antenne k zu der Position des Tags r_k :

$$r_k^2 = (x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2 \quad (3.1)$$

Diese Gleichung stellt die euklidische Vektornorm dar und entspricht der Strecke Antenne-Tag. Für die Ermittlung einer Postion (mit drei Raumkoordinaten) sind drei Antennen notwendig. Daraus ergibt sich:

- 3 Gleichungen
- 3 Unbekannte
- Quadratisches Gleichungssystem

Das Gleichungssystem sieht wie folgt aus:

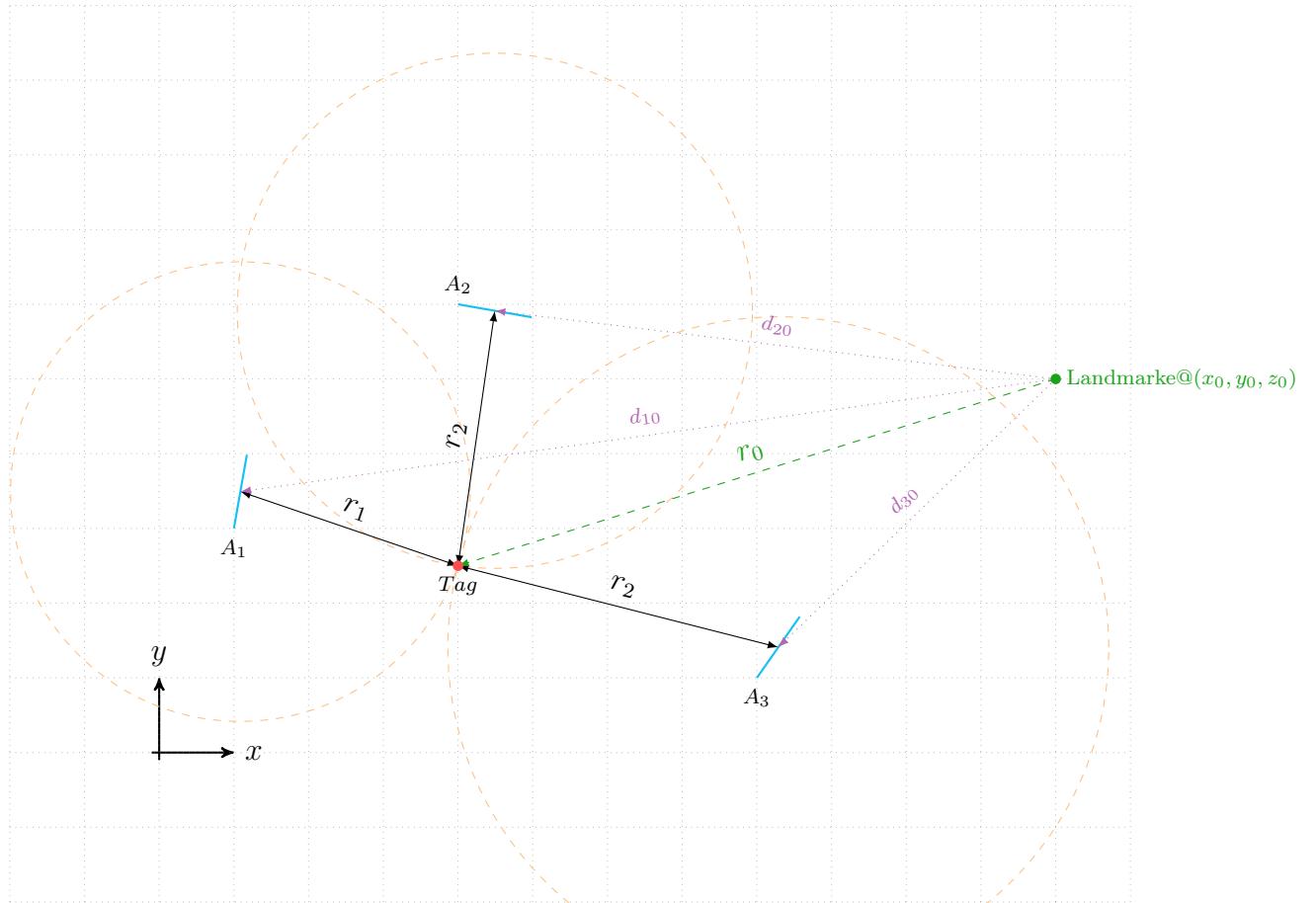
$$\begin{aligned} r_1^2 &= (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 \\ r_2^2 &= (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 \\ r_3^2 &= (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 \end{aligned}$$

Es ist trivial und es wird in verschiedenen Beispielen gezeigt², dass man die Koordinaten aus dem quadratischen Gleichungssystem unmittelbar berechnen kann. Es muss jedoch ein quadratisches Gleichungssystem gelöst werden, was zu den bekannten Problematiken führt, insbesondere der Ausschluss mehrdeutiger Ergebnisse. Der Messaufbau der amedo STS erlaubt die

¹d.h. ein Messergebnis liefern

²z.B. <http://en.wikipedia.org/w/index.php?title=Trilateration&oldid=553215995>

Abbildung 3.4.: 2D-Übersicht auf die Szene mit drei Antennen, einem Tag und einer Landmarke. Die Position von $\{A_1, A_3, A_3\}$, sowie der Landmarke, zum Koordinatenursprung sind bekannt. Die Vektoren r_1, r_2, r_3 sind die gemessene Entfernung zu einer Antenne. Die Landmarke wird im späteren Verlauf eine Antenne sein, die ihrerseits eine gemessene Entfernung r_0 produziert. Der Schnittpunkt aller Kreise ist die Lösung der gemessenen Entfernung und der geom. Anordnung, die sich für die Position des Tags ergibt.



Verwendung von mehr als 3 Messwertgebern. Diese zusätzlichen Informationen lassen sich für eine Linearisierung des Gleichungssystems verwenden. Dieser Ansatz wird für ein Modell im Rahmen dieser Arbeit verwendet und wird im Folgenden beschrieben.

Von den Antennen sind die Raumkoordinaten (x, y, z – Koordinaten) bekannt, bzw. wurden durch Kalibrierung (vgl. Abschnitt 3.7) in einem vorherigen Schritt bestimmt. Wir können zusätzlich zu notieren:

$$d_{kj}^2 = (x_k - x_0)^2 + (y_k - y_0)^2 + (z_k - z_0)^2 \quad (3.2)$$

Linearisierung des Modells. Dazu wird Gleichung 3.1 in mehreren Schritten umgebaut. Zuerst wird eine neutrale Erweiterung durchgeführt und die Terme zusammengefasst. Das führt zu:

$$\begin{aligned} r_k^2 &= (x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2 \\ &= (x - x_k + x_0 - x_0)^2 + (y - y_k + y_0 - y_0)^2 + (z - z_k + z_0 - z_0)^2 \\ &= ((x - x_0) - (x_k - x_0))^2 + ((y - y_0) - (y_k - y_0))^2 + ((z - z_0) - (z_k - z_0))^2 \\ &= (x - x_0)^2 - 2(x - x_0)(x_k - x_0) + (x_k - x_0)^2 \underbrace{+ \dots + \dots}_{\text{y- & z-Terme analog}} \end{aligned} \quad (3.3)$$

Um Platz zu sparen sind die y- und z-Terme nicht explizit notiert. Sie ergeben sich durch einfaches Ersetzen der Indizes und werden im finalen Modell eingefügt. Durch Umstellen von (3.3) erhalten wir:

$$\begin{aligned} (x - x_0)(x_k - x_0) + \dots + \dots &= -\frac{1}{2}[r_k^2 - (x_k - x_0)^2 - (x - x_0)^2 + \dots + \dots] \\ (x - x_0)(x_k - x_0) + \dots + \dots &= -\frac{1}{2}[(x_k - x_0)^2 + (x - x_0)^2 + \dots + \dots - r_k^2] \\ (x - x_0)(x_k - x_0) + (y - y_0)(y_k - y_0) + (z - z_0)(z_k - z_0) &= \\ \frac{1}{2}[(x_k - x_0)^2 + (x - x_0)^2 - (y_k - y_0)^2 + (y - y_0)^2 & \\ - (z_k - z_0)^2 + (z - z_0)^2 - r_k^2] \end{aligned} \quad (3.4)$$

Vergleich von (3.4) mit (3.2) bringt:

$$\begin{aligned} (x - x_0)(x_k - x_0) + (y - y_0)(y_k - y_0) + (z - z_0)(z_k - z_0) &= \\ \frac{1}{2}[\underbrace{(x_k - x_0)^2 + (z_k - z_0)^2 + (y_k - y_0)^2}_{d_{kj}^2} & \\ + \underbrace{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r_k^2}_{r_j^2}] \end{aligned} \quad (3.5)$$

$$(x - x_0)(x_k - x_0) + (y - y_0)(y_k - y_0) + (z - z_0)(z_k - z_0) = \frac{1}{2}[d_{kj}^2 + r_j^2 - r_k^2] \quad (3.6)$$

mit

$$\mathbf{c}_{kj} = \frac{1}{2}[d_{kj}^2 + r_j^2 - r_k^2] \quad (3.7)$$

können wir für das lineare Gleichungssystem abschließend schreiben:

$$\mathbf{0} = \begin{pmatrix} x_1 - x_j & y_1 - y_j & z_1 - z_j \\ x_2 - x_j & y_2 - y_j & z_2 - z_j \\ x_3 - x_j & y_3 - y_j & z_3 - z_j \end{pmatrix} \begin{pmatrix} x - x_j \\ y - y_j \\ z - z_j \end{pmatrix} - \begin{pmatrix} c_{1j} \\ c_{2j} \\ c_{3j} \end{pmatrix} \quad (3.8)$$

Das Gleichungssystem ist linear und hat die allg. Form: $\mathbf{0} = \mathbf{Ax} + \mathbf{b}$. Es lässt sich daher mit bekannten Methoden lösen.

Zusammenhang mit der Wellenzahl

Wie gezeigt wurde ergibt sich für den Fall der Trilateration und der Annahme, dass vier Antennen Messwerte liefern, die Gleichung:

$$\mathbf{0} = \begin{pmatrix} x_k - x_0 & y_k - y_0 & z_k - z_0 \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} - \begin{pmatrix} c_{kj} \end{pmatrix} \quad (3.9)$$

Wir stellen fest, dass dieses Modell rein geometrisch ist. Es erlaubt bereits einen Einsatz im Rahmen der Kalibrierung (siehe 3.7). Es wird im Folgenden eine Erweiterung dieses Modells gezeigt. Ziel ist es, einen Zusammenhang zwischen diesem Modell, der gemessenen Phase und der Wellenzahl zu erzeugen. Folgender Ansatz wird gewählt:

$$r(\varrho, n) = \frac{\lambda}{2} \left(\frac{\varrho}{2\pi} + n \right), \lambda = \frac{c}{f}, n := \text{Wellenzahl} \quad (3.10)$$

In dem Modell steht ϱ_k für die gemessene Phase vom Messsystem und n_k ist die gesuchte Wellenzahl. Der Index k deutet eine Existenz der beiden Parameter für jede Antenne an. Durch einsetzen von (3.10) in (3.7), erhalten wir:

$$c_{kj}(\varrho_0, \varrho_k, n_0, n_k) = \frac{1}{2} \left[d_{kj}^2 + \frac{\lambda^2}{4} \left(\frac{\varrho_j}{2\pi} + n_0 \right)^2 - \frac{\lambda^2}{4} \left(\frac{\varrho_k}{2\pi} + n_k \right)^2 \right] \quad (3.11)$$

Wir stellen Gleichung (3.11) um:

$$\begin{aligned} c_{kj}(\varrho_0, \varrho_k, n_0, n_k) &= \frac{1}{2} \left\{ d_{kj}^2 + \frac{\lambda^2}{4} \left[\left(\frac{\varrho_j}{2\pi} \right)^2 + 2 \frac{\varrho_j}{2\pi} n_0 + n_0^2 \right. \right. \\ &\quad \left. \left. - \left(\frac{\varrho_k}{2\pi} \right)^2 - 2 \frac{\varrho_k}{2\pi} n_k - n_k^2 \right] \right\} \end{aligned} \quad (3.12)$$

$$\begin{aligned} &= \frac{1}{2} \left\{ d_{kj}^2 + \frac{\lambda^2}{4} \left[\left(\frac{\varrho_j}{2\pi} \right)^2 - \left(\frac{\varrho_k}{2\pi} \right)^2 \right. \right. \\ &\quad \left. \left. + 2 \frac{\varrho_j}{2\pi} n_0 - 2 \frac{\varrho_k}{2\pi} n_k + n_0^2 - n_k^2 \right] \right\} \end{aligned} \quad (3.13)$$

$$\begin{aligned} &= \frac{1}{2} d_{kj}^2 + \frac{\lambda^2}{8} \left[\frac{1}{(2\pi)^2} (\varrho_0^2 - \varrho_k^2) \right. \\ &\quad \left. + \frac{1}{\pi} (\varrho_0 n_0 - \varrho_k n_k) + (n_0^2 - n_k^2) \right] \end{aligned} \quad (3.14)$$

Führen wir nun:

$$a_{0k} := \frac{1}{2} d_{kj}^2$$

$$a_1 := \frac{\lambda^2}{8}$$

$$a_2 := a_1 \frac{1}{\pi}$$

$$a_{3kj} := a_1 \frac{1}{(2\pi)^2} (\varrho_j^2 - \varrho_k^2)$$

in Gleichung (3.14) ein, erhalten wir die finale Form der Gleichung:

$$c_{kj}(\varrho_0, \varrho_k, n_0, n_k) = a_{0k} + a_1(n_0^2 - n_k^2) + a_2(\varrho_0 n_0 - \varrho_k n_k) - a_{3kj} \quad (3.15)$$

Die Einführung der Konstanten macht zum einen die Gleichung übersichtlicher und zum anderen können so in der spätere Softwareimplementation Rechenschritte gespart werden. Das wirkt sich günstig auf den Rechenaufwand aus. Im Weiteren erkennt man, dass in Gleichung (3.15), für $\varrho_k = \text{const.}$ & $\varrho_0 = \text{const.}$ gilt. Der Grund dafür liegt darin, dass ϱ zwar die Messwerte beschreibt, diese jedoch nur in dem Modell eingeführt werden. Im Sinne der später durchgeföhrten Optimierung sind diese Parameter keine Variablen. Es ermöglicht uns zu schreiben:

$$c_{kj}(\varrho_0, \varrho_k, n_0, n_k) = c_{kj}(n_0, n_k) \quad (3.16)$$

Im engeren Sinne einer mathematischen Funktion sollten wir die Parameter alle als Argument aufnehmen. Diese Form soll darstellen, welche Größen von

Interesse sind. Im späteren Gebrauch wird diese Gleichung in der Optimierung eingesetzt werden.

Für unser Gleichungssystem aus (3.9) ergibt sich:

$$\mathbf{0} = \begin{pmatrix} x_k - x_0 & y_k - y_0 & z_k - z_0 \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} - \begin{pmatrix} c_{kj}(n_0, n_k) \end{pmatrix} \quad (3.17)$$

Konkretes Beispiel

Für ein konkretes Beispiel Betrachten wir nun (3.17). Dabei wählen wir $|N'| = 4$ (d.h. wir verwenden 4 Antennen) und setzen $j = 0$. Diese exemplarische Konfiguration kann wie folgt beschrieben werden: Antenne 0 ist die Referenz-Antenne und Antennen 1, 2 und 3 sind Messwertgeber für die Phaseninformation. Im praktischen Gebrauch werden die Konfigurationen anders zusammengestellt. Strategien für die Zusammenstellung werden später beschrieben.

Für die gewählte Konfiguration ergibt sich explizit:

$$\mathbf{0} = \underbrace{\begin{pmatrix} x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 \\ x_3 - x_0 & y_3 - y_0 & z_3 - z_0 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}}_{\mathbf{x}} - \underbrace{\begin{pmatrix} c_{10}(n_0, n_1) \\ c_{20}(n_0, n_2) \\ c_{30}(n_0, n_3) \end{pmatrix}}_{\mathbf{b}} \quad (3.18)$$

Wir wollen den Vektor \mathbf{b} nun explizit betrachten:

$$\mathbf{b} = \begin{pmatrix} a_{01} + a_1(n_0^2 - n_1^2) + a_2(\varrho_0 n_0 - \varrho_1 n_1) - a_{310} \\ a_{02} + a_1(n_0^2 - n_2^2) + a_2(\varrho_0 n_0 - \varrho_2 n_2) - a_{320} \\ a_{03} + a_1(n_0^2 - n_3^2) + a_2(\varrho_0 n_0 - \varrho_3 n_3) - a_{330} \end{pmatrix} \quad (3.19)$$

Das Ergebnis ist ein um ϱ und n erweitertes Gleichungssystem. Zusätzlich enthält es mehrere geometrische Konstanten (a_{0k}), mehrere Phasen-Konstanten (a_{3k0}), sowie zwei Systemparameter abhängige Konstanten (a_1 und a_2). Allgemeiner formuliert ergibt sich:

$$0 = \begin{pmatrix} x_k - x_0 & y_k - y_0 & z_k - z_0 \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} - \begin{pmatrix} a_{0k} + a_1(n_0^2 - n_k^2) + a_2(\varrho_0 k_0 - \varrho_k n_k) - a_{3kj} \end{pmatrix} \quad (3.20)$$

Hinzufügen von Antennen - Der allgemeine Fall

Aus dem oben beschriebenen Beispiel, Gleichung (3.20), und der dort getroffene Wahl von $|N'| = 4$ ergibt sich wie viele Veränderliche sich für eine gewählte Konstellation an Antennen ergeben. Leiten wir daraus nun einen

allgemeinen Fall ab. Für k gilt in diesem Fall $k = \{1, \dots, N' - 1\}$, wir wählen die Referenzantenne $j = 0$ und die Menge an verwendeten Antennen gleich der Anzahl der verfügbaren ($N' = N$). Es ist leicht ersichtlich, dass sich die Anzahl der verwendeten Antenne unmittelbar auf die Zahl der Variablen auswirkt. Es ergeben sich für das Modell mit vier Antennen insgesamt 7 Variablen $(\mathbf{x}, n_0, n_1, n_2, n_3)$, wobei sich für ein Modell mit allen 8 Antennen, 11 Variablen $(\mathbf{x}, n_0, \dots, n_7)$ ergeben. Andere Konfigurationen verhalten sich analog dazu.

Relevanz dieses Modells

Dieses Modell hat unmittelbare Relevanz für die Praxis. Es trägt dem Umstand Rechnung, dass zu einem Messzeitpunkt ein Teil der Antennen keine Messwerte liefern könnte. Das Modell erlaubt daher, dass die Anzahl und die Auswahl der Antennen variieren kann. Damit ist das Modell uneingeschränkt tauglich für den Einsatz in dem PRPS-Messsystem.

Abschließend soll das bisher verwendete Modell umgeschrieben werden, damit die Allgemeingültigkeit darin enthalten ist.

$$\mathbf{A} = \begin{pmatrix} x_k - x_0 & y_k - y_0 & z_k - z_0 & \sum_{i=1, j=0}^k (-a_1 \delta_{ij}) & -a_2 \Theta_0 & \sum_{i=1, j=0}^k (a_2 \Theta_k \delta_{ij}) \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \\ n_0^2 - n_k^2 \\ n_0 \\ n_k \end{pmatrix}$$

$$\mathbf{b} = a_{0k} - a_{3kj} = c'_{kj}$$

Dabei steht δ_{ij} für den bekannten Kronecker-Operator und bedeutet:

$$\delta_{ij} = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{für } i \neq j \end{cases}$$

Im Expliziten sehen die Matrix \mathbf{A} und der Vektor \mathbf{b} , für den Fall $N' = 3$ und $k = \{1, 2, 3\}$, wie folgt aus:

$$\mathbf{A} =$$

$$\begin{pmatrix} x_1 - x_0 & y_1 - y_0 & z_1 - z_0 & -a_1 & 0 & 0 & -a_2 \Theta_0 & a_2 \Theta_3 & 0 & 0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 & 0 & -a_1 & 0 & -a_2 \Theta_0 & 0 & a_2 \Theta_3 & 0 \\ x_3 - x_0 & y_3 - y_0 & z_3 - z_0 & 0 & 0 & -a_1 & -a_2 \Theta_0 & 0 & 0 & a_2 \Theta_3 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \\ n_0^2 - n_1^2 \\ (\dots) \\ n_0^2 - n_3^2 \\ n_0 \\ n_1 \\ (\dots) \\ n_3 \end{pmatrix}$$

Bemerkungen - Finales Modell

Das Ergebnis ist eine 3×10 Matrix und ein 1×10 Vektor. Es ist möglich diesem Modell eine beliebige Anzahl an Antennen hinzuzufügen. Fügt man eine Antenne zur Berechnung hinzu, würde sich die Matrix \mathbf{A} um zwei Spalten und eine Zeile erweitern, der Vektor \mathbf{x} analog um 2 Zeilen.

3.3. Antennen Permutationen

Um die Beurteilung der Konditionszahl der Matrizen effizient durchführen zu können, wurde im Rahmen dieser Arbeit ein Programm geschrieben, das diese Aufgabe erledigt. Es erstellt automatisch, auf Basis der durch die Kalibrierung bestimmten Koordinaten, alle möglichen Permutationen von Antennen. Die sich ergebenden Matrizen sind immer auf eine Referenzantenne bezogen (vgl. Gleichung 3.20). Ausgehen von einem Aufbau aus 8 Antennen und der Auswahl von 4 Antennen pro Matrix, wobei eine die Referenzantenne ist, ergeben sich folgende Anzahl an möglichen Matrizen:

$$\frac{7!}{3!(7-3)!} = 35 \quad (3.21)$$

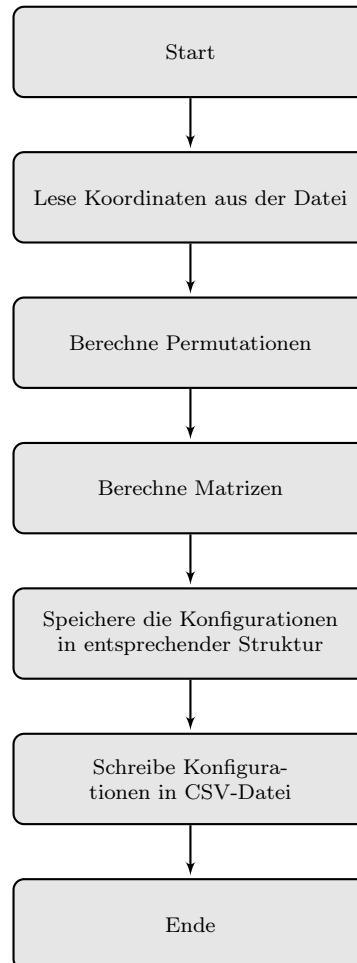
Für einen Aufbau mit acht Antennen ergeben sich so $8 \times 35 = 280$ mögliche Anordnungen. Aus dem Zusammenhang kann man unmittelbar bestimmen wie viele mögliche Matrixkombinationen sich ergeben, sollten einige Antennen keine Messwerte zu einem Zeitpunkt beitragen können.

Beispiel

Wenn zu einem Messzeitpunkt nur 6 von 8 Antennen einen Messwert liefern, verringert sich die Anzahl der für die Berechnung benutzbaren Kombinationen auf

$$\frac{5!}{3!(5-3)!} = 10$$

Abbildung 3.5.: Ablauf in dem Modul libPermutate



Möglichkeiten. Damit erhalten wir eine Gesamtheit von $10 \times 6 = 60$ Matrizen.

Die Implementation der Permutationsberechnung findet sich in dem Modul 'libPermutate'. Das Modul generiert bei der Instanziierung automatisch alle möglichen Kombinationen von Antennen und speichert diese in einer geeigneten Struktur für den späteren Gebrauch. Das Ablaufdiagramm ist in Abbildung 3.5 zu finden.

3.3.1. Anwendung der Permutationen

Durch das Einbringen der Kombinationen von Antennen in ein Modell, ist es möglich der Optimierung zusätzliche Informationen zum Lösen des Problems

zu geben. In dieser Arbeit wird in diesem Zusammenhang von Gruppengröße gesprochen.

3.4. Erweiterte Betrachtung der Kondition

Die vorgestellte erweiterte Form des Modells erleichtert die Implementation und Verifikation, da große Teile vorberechnet und in geeigneten Strukturen abgelegt werden können. Diese statischen Teile des Modells sind in Gleichung 3.22 ersichtlich. Es sind nun auch die gemessenen Phasenwerte Teil des Modells, genauer: der Matrix \mathbf{A} . Im Folgenden werden die Auswirkungen auf die Kondition der Matrix betrachtet, wenn man diese Phasendaten hinzurechnet. Weiterhin wird untersucht inwieweit die Zerlegung in Blockmatrizen und die Untersuchung der Kondition dieser eine Abschätzung der vollständigen Konditionszahl im Allgemeinen darstellt.

$$\mathbf{A} = \begin{pmatrix} \mathbf{Z} & \mathbf{P} & \mathbf{V} \end{pmatrix} \quad (3.22)$$

Dabei ist:

$$\mathbf{Z} \in \mathbb{R}^{3x3} \quad \mathbf{P} \in \mathbb{R}^{3x3} \quad \mathbf{V} \in \mathbb{R}^{4x3} \quad (3.23)$$

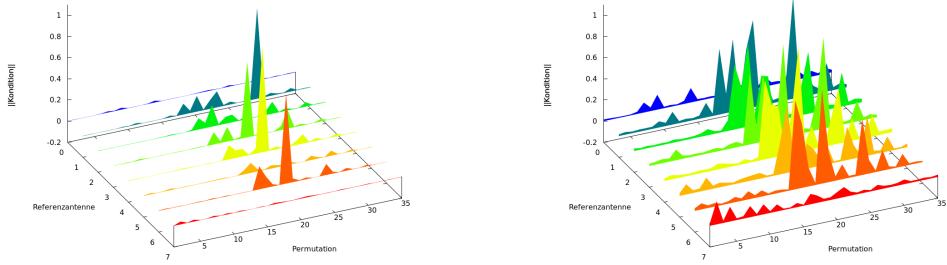
Die Matrizen \mathbf{Z} und \mathbf{P} sind statisch. Hingegen enthält die Matrix \mathbf{V} die gemessenen Phasenwerte Θ_k der Antennen für diese Konfiguration.

Die Abbildung 3.6 zeigt die bereits angestellte Untersuchung zu dieser Überlegung. Abbildung 3.7a stellt die Konditionszahl der rein geometrischen 3×3 -Matrix dar. In der Abbildung 3.7b sehen wir die Kondition der erweiterten Matrix. Neben der geometrischen sind auch die beiden anderen Blockmatrizen in diese Konditionsbetrachtung eingeflossen. Als zusätzliche Angabe sind die Skalierungsfaktoren angegeben. Legt man beide Grafiken übereinander erkennt man:

1. Geometrisch gut konditionierte Konfigurationen (linke Grafik), bleiben im erweiterten Modell (rechte Grafik) weiterhin gut konditioniert.
2. Die Konditionszahl der *schlechten* Konfiguration ist wesentlich kleiner (ca. Faktor 10) als im rein geometrischen Modell

Aus der Grafik lässt sich entnehmen, dass für jede Referenzantenne aus der Geometrie alleine gute Konfigurationen existieren. Aus diesen Erkenntnissen kann in späteren Aufbauten die Position der Antennen optimiert werden. Dieses Verfahren wird in Abschnitt 6.1 weiter beschrieben. Die Grafik lässt erkennen, dass eine Konfiguration, die ohne Phasendaten eine gut Kondition aufwies, eine ähnliche Kondition behält, wenn diese Daten in das Modell einfließen.

Abbildung 3.6.: Analyse der Konditionszahlen aller möglichen Matrizen für den Messaufbau. Die Konditionszahl ist für jede mögliche Permutation an Messantennen für eine Referenzantenne angegeben



(a) Konditionszahl der rein geometrischen 3×3 Matrix normiert auf den größten vorkommenden Wert ($= 2149, 16$). Auf den Achsen finden sich der Index der Referenzantenne sowie die Nummer der Permutation. Die z-Achse enthält die normierte Kondition

(b) Konditionszahl der 10×3 Matrix normiert auf den größten vorkommenden Wert ($= 257, 13$); In dieser Konfiguration sind die Konstanten (a_1 & a_2) sowie die variablen, gemessenen Phasen Θ_k enthalten

3.4.1. Weitere Anwendung der Konditionszahl

Weitere Anwendungen, die sich aus der Konditionszahl der Matrix ableiten, sind denkbar. Die Steuereinheit wird, parallel zu diesem Projekt, um eine intelligente Umschaltung der Antennen erweitert. Das Problem ist dabei die 'wissende' Umschaltung auf andere Antennen. Dazu lässt sich die Kondition der Antennenkombinationen nutzen. Die Kondition der geometrischen Matrix verändert sich nach dem Kalibrieren nicht mehr. Dadurch und durch die oben beschriebenen Überlegungen kann statisch eine Abschätzung für die Konditionszahl, von zwei der drei Blockmatrizen im Vorfeld erstellt werden. Die Konditionszahl dient zum Steuern der Umschaltung. Ordnet man die möglichen Konfiguration anhand ihrer Konditionszahl (niedrigste zuerst) in einer statischen Liste an, so kann in der Steuereinheit eine einfache, schlaue Umschaltung implementiert werden. Diese würde immer dafür sorgen, dass Messdaten von einer Konfiguration bevorzugt werden, die eine niedrige Konditionszahl hat und somit relativ sicher zu einer guten Lösung führen. Diese Überlegungen werden im Rahmen dieser Arbeit nicht näher beschrieben.

Eine weitere Anwendung ergibt sich für die Kalibrierung. Der Aufbau der Antennen kann unter Berücksichtigung der Kondition optimiert werden. Ziel der Optimierung wäre es, durch eine geeignete Positionierung der Antennen,

die Anzahl der Antennenpermutationen mit kleiner Konditionszahl zu maximieren.

3.5. Einsatz des Modells

Im vorherigen Abschnitt wurde das Modell aus den geometrischen Gegebenheiten hergeleitet. Das Modell ist in seinen Eingabeparametern flexibel und erlaubt verschiedene Arten des Einsatzes. Diese werden im Folgenden erläutert.

3.6. Betrachtung der Komplexität

Es soll nun Komplexität des in Abschnitt 3.2 entwickelten Modells betrachtet werden. Diese Betrachtung ist wichtig für die Parametrisierung des Optimierungsverfahrens, sowie für eine Beurteilung der generellen Lösbarkeit mit den verwendeten Verfahren. Es wird eine Visualisierung der Fitness-Landschaft³ vorgestellt.

Um einen Ausschnitt der Fitnesslandschaft zu erstellen, die sog. Fitness-Ebenen, wurde ein eigener Programmteil (der *FitnessPlaneCalculator*) entwickelt. Das Programm nimmt ein in dieser Arbeit entwickeltes Modell, füttert es mit vorgegebenen Daten und schreibt den Rückgabewert in eine Datei. Das Programm lässt sich per Eingabedatei steuern und erlaubt die Definition der Ebenen die dargestellt werden sollen. Es lassen sich immer zwei Variablen der Objektfunktion variieren und der Rest wird dabei auf feste Werte gesetzt. Das erlaubt eine Visualisierung durch eine 2D-Heatmap (siehe folgende Plots). Formal lässt sich das vorgehen wie folgt beschreiben:

$$f(\mathbf{x}) \in \mathbb{R}^N \rightarrow \mathbb{R}^2$$

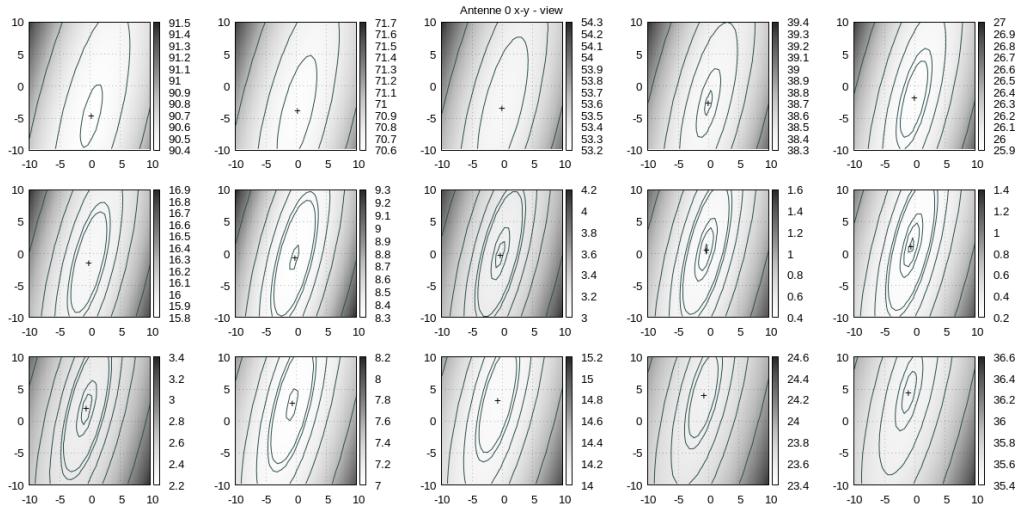
Aus der Visualisierung können Rückschlüsse auf die Gestalt der Fitnessebene gezogen werden.

Die Parameter wurden für diese Plots so gewählt, dass sie über einen Bereich iterieren, indem die richtige Lösung liegen sollte. Eingezeichnet sind verschiedene Höhenlinien, die zur Orientierung dienen sollen. Diese sind für die Werte des diskreten Intervalls: $[0, 1, 5, 10, 50, 100, 200]$ ausgeführt. Sie tragen zur besseren Übersicht bei. Die Farbskala des Heatmap skaliert den Plot auf den Wertebereich $[0, 1]$ und addiert das im Plot vorkommende Minimum dazu. So kann man leicht eine qualitative Beurteilung der Ebene ablesen. Da

³Auch Fitness-Raum genannt

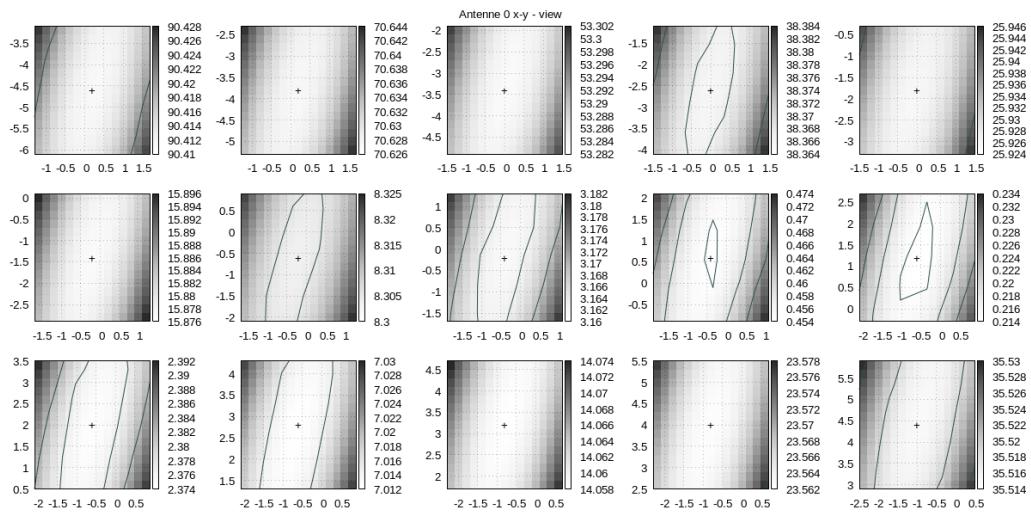
nur drei Parameter variiert werden, erhalten die übrigen vier analytisch bestimmte, wahre Werte. Diese sind in Tabelle 3.1 zusammen mit den wahren Werten aufgeführt.

Abbildung 3.9.: Diese Grafik zeigt die Fitnessebenen \mathbb{R}^2 des Problems für eine Anordnung aus vier Antennen und einem Sweep über die $x - y$ -Ebene. Jeder Plot ist für einen festen Wert z erstellt worden. In der oberen linken Ecke beginnend und nach rechts laufend. Die x, y -Werte wurden über ein Intervall von $[-5, 5]$ m und mit einem Inkrement von 0.2 variiert. Daraus ergibt sich eine für die Abschätzung der Gestalt der Fitnessebene ausreichende Datengrundlage. Die z -Werte der 15-Plots stammen aus dem Intervall $[-7, -7]$ mit einem Inkrement von 1. Bereits in dieser Ansicht ist zu erkennen, dass der Verlauf sehr flach ist. Eine Schlucht bildet sich etwa in Nord-Süd-Richtung aus. Jeweils verzeichnet ist das lokale Minima (+) eins Plots, sowie die Höhenlinien⁴. Die farbliche Kodierung gibt den Fitnesswert an diesem Punkt an. Die Fitnesswerte wurden normiert und um den Wert ihres Minimums verschoben.



In den Abbildungen 3.9 und 3.10 zeigen sich erste Herausforderungen für den Algorithmus. Eine große, sehr flache Fitnessebene für verschiedene Parameter ist nicht leicht zu handhaben. Eine Möglichkeit dieses Problem zu umgehen ist eine große Anzahl an Nachkommen zu generieren. Im Jargon der EA: großes λ . In den Ergebnissen wurde mit unterschiedlichen Populationsgrößen experimentiert, dabei zeigte sich, dass die Anzahl der Nachkommen groß sein sollte, damit eine gute Güte der Lösung gefunden werden kann.

Abbildung 3.10.: Vergrößerung der Fitnessebenen der Antenne 1. Es ist hier deutlich zu erkennen, wie gering die Funktionen ansteigen. Das ist ein Problem für die meisten Algorithmen. Es bleibt zu klären wie sensitiv der Algorithmus auf diesen Umstand reagiert. Es wurde um das lokale Minima zentriert und der Bereich auf



Ein anderer, steuernder Faktor ist die Schrittweite σ . Diese darf nicht zu klein werden, sonst besteht die Gefahr mit der Population auf der flachen Ebene liegen zu bleibt. Was zu mehrdeutigen Ergebnissen führt. Die Fitness-Ebenen der anderen Antennen zeigen das gleiche Verhalten. Diese sind im Anhang ?? zu finden.

Aufgrund der Komplexität des in dieser Arbeit erstellten Modells und der Komplexität des zugrunde liegenden Sachverhalts (siehe Abschnitt 3.3), ist bereits der Fall mit vier Antennen sehr hochdimensional. Er erreicht 7 Dimensionen und er kann im Rahmen dieser Arbeit nicht vollständig untersucht bzw. dargestellt werden. Die Fitness-Plots aller Antennen und für drei Ebenen sind in Anhang D.1 zu finden. Eine vollständige Untersuchung ist indes auch nicht notwendig, da der Algorithmus den Suchraum natürlicherweise untersucht.

Abbildung 3.11.: Auf diesen Abbildungen zeigen sich die Fitness-Ebenen für die übrigen Ansichten der Antenne 1, x-z und y-z. In der oberen Reihe sind Ebenen über den gesamten Bereich. Die Unteren zeigen eine vergrößerte Darstellung um das Minimum. Zu erkennen ist ein zum Verlauf der x-y-Ebene sehr ähnliches Bild. Ein flaches, längliches Tal mit Minimum. Die Dimension der x-Achse ist [m].

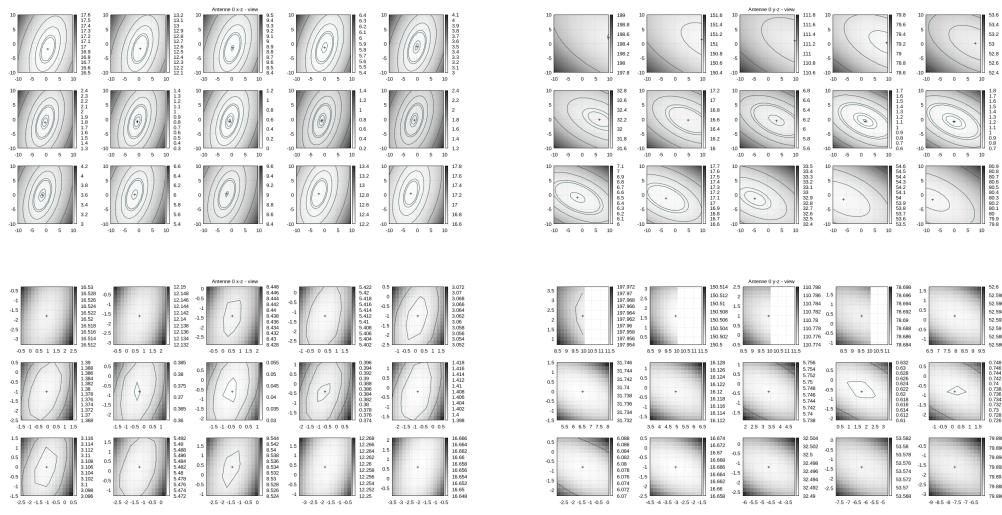


Tabelle 3.1.: Tabellarisch sind hier die Parameter (Intervalle) der Fitness Ebenen aufgelistet. Zusätzlich sind die Werte angegeben, in denen eine optimale Lösung liegen sollte. Notation: [Start:Inkrement:Ende]

Ebene	x	y	z	n_0	n_1	n_2	n_3
x-y	[-10:2:10]	[-10:2:10]	[-7:1:7]	[7:0:7]	[10:0:10]	[13:0:13]	[9:0:9]
x-z	[-10:2:10]	[-7:1:7]	[-10:2:20]	[7:0:7]	[10:0:10]	[13:0:13]	[9:0:9]
y-z	[-7:1:7]	[-10:2:10]	[-10:2:10]	[7:0:7]	[10:0:10]	[13:0:13]	[9:0:9]
Wahre	0.479	-1.012	0.607	7	10	13	9

3.7. Realisierung der Kalibrierung

In diesem Abschnitt wird die Implementierung der Kalibrierung des Messaufbaus präsentiert und die Ergebnisse werden kurz zusammengefasst. Es werden zwei unterschiedliche Berechnungsverfahren vorgestellt. Zuerst die Berechnung über das SVD-Verfahren (numerisch), danach durch das CMA-ES-Verfahren (evolutionär). Es ist sinnvoll zu erwarten, dass beide Ergebnisse die gleichen Koordinaten liefern.

3.7.1. Implementation

Der Ablauf der Kalibrierung ist in Abbildung 3.15 in Form eines Ablaufdiagramms dargestellt. Beschrieben werden die wesentlichen Schritte. Es sind sowohl Interaktion mit der Person enthalten, die die Kalibrierung durchführt, als auch die Schritte, die von den beteiligten Softwarekomponenten ausgeführt werden enthalten. Es wurden im Rahmen der Arbeit zwei unterschiedliche Wege implementiert, um ein Ergebnis für die Kalibrierung zu berechnen. Diese Wege werden im Folgenden vorgestellt und die Ergebnisse miteinander verglichen. Die Präsentation der Resultate wird vor allem dazu verwendet werden, die gewählte Form der Diagramme zu erläutern. Diese werden in den Ergebnissen des komplexeren Modells ebenfalls verwendet.

SVD

Das unter 2.1.2 vorgestellte Verfahren der Singular-Value-Decomposition kann dazu verwendet werden eine Lösung eines linearen Gleichungssystems zu berechnen. Das Modell, das zur Kalibrierung verwendet wird, ist ein Gleichungssystem der Form $\mathbf{Ax} = \mathbf{b}$ und hat drei Gleichungen mit drei Unbekannten. Daher kann sofort eine Lösung mit dem Verfahren hergeleitet werden. Das Ergebnis eines Messaufbaus mit 3 Antennen ist in Tabelle 4.1 und in Abbildung 4.5 gezeigt. Die Implementation des Algorithmus stammt aus [20] und wurde für diese Arbeit angeschafft.

CMA-ES

Da in dieser Arbeit der CMA-ES-Algorithmus eingesetzt wird und damit ohnehin eine Implementation vorgenommen wird, kann die Kalibrierung dazu verwendet werden die Umsetzung des Algorithmus zu verifizieren. Dazu vergleichen wir die Lösung der SVD-Methode mit der des CMA-ES.

Das über den evolutionären Algorithmus gefundene Ergebnis gleicht dem des SVD-Verfahrens (siehe Tabelle 4.1). Der SVD-Algorithmus ist um ein vielfaches effizienter⁵ beim Lösen des Gleichungssystems. Die Gründe, warum an

⁵d.h. weniger Rechenzeit ist erforderlich

dieser Stelle die Berechnung mit dem evolutionären Verfahren durchgeführt und hier dargestellt wird sind folgende:

1. Die Komplexität ist gering, daher kann der Ablauf des evolutionären Verfahrens besser dargestellt und verstanden werden
2. Der Vergleich der beiden Ergebnisse ermöglicht die Verifizierung der Implementation beider Verfahren.

Dem ersten Punkt kommt im Rahmen dieser Arbeit eine besondere Stellung zu. Es ist einfacher anhand dieses übersichtlichen Problems (mit nur drei Unbekannten) den Ablauf des Algorithmus sowie die Visualisierung der Ergebnisse zu veranschaulichen. Die verwendete Darstellung gleicht der, die später bei der Präsentation und Beurteilung der komplexeren Modell verwendet wird. Die Ergebnisse werden in dem Kapitel 4.1 gezeigt.

Abbildung 3.13.: Werkzeuge, die bei der Kalibrierung verwendet werden.



(a) Laser Distanzmesser



(b) Kalibrierstück mit vier Messpositionen

Abbildung 3.15.: Ablauf der Kalibrierung

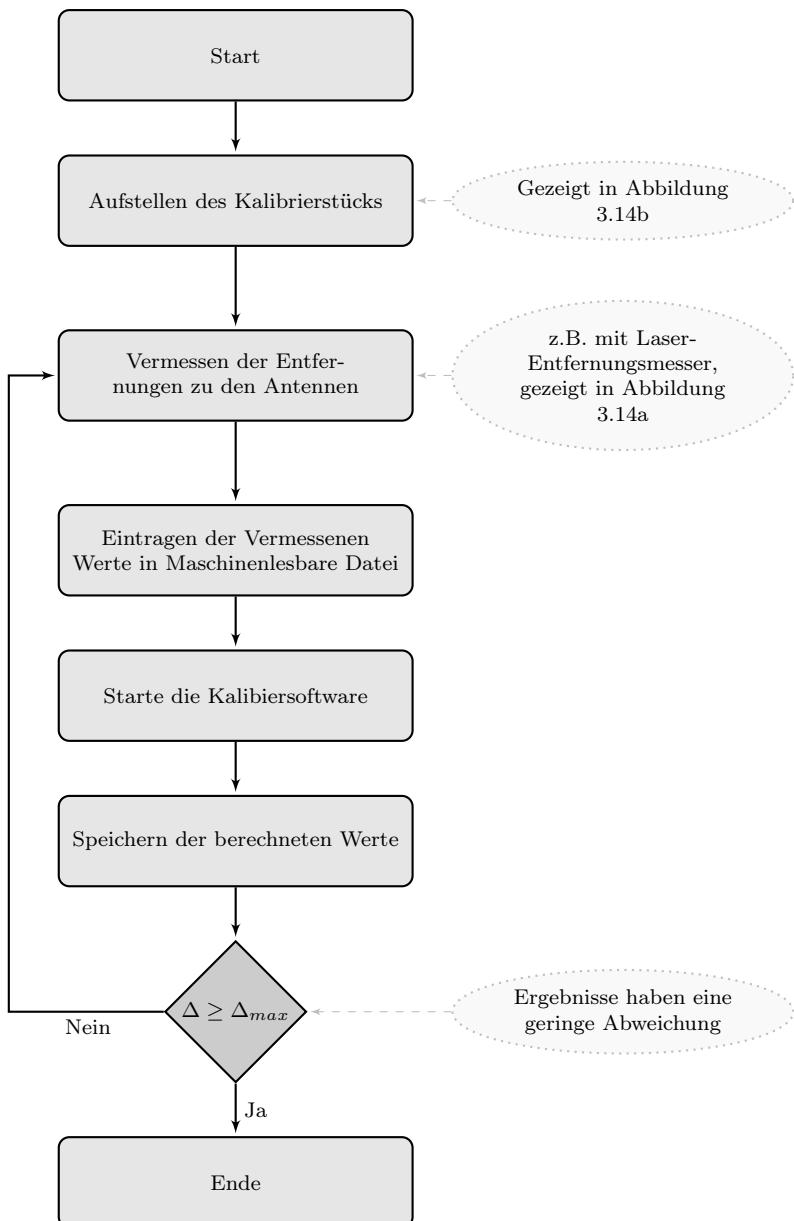
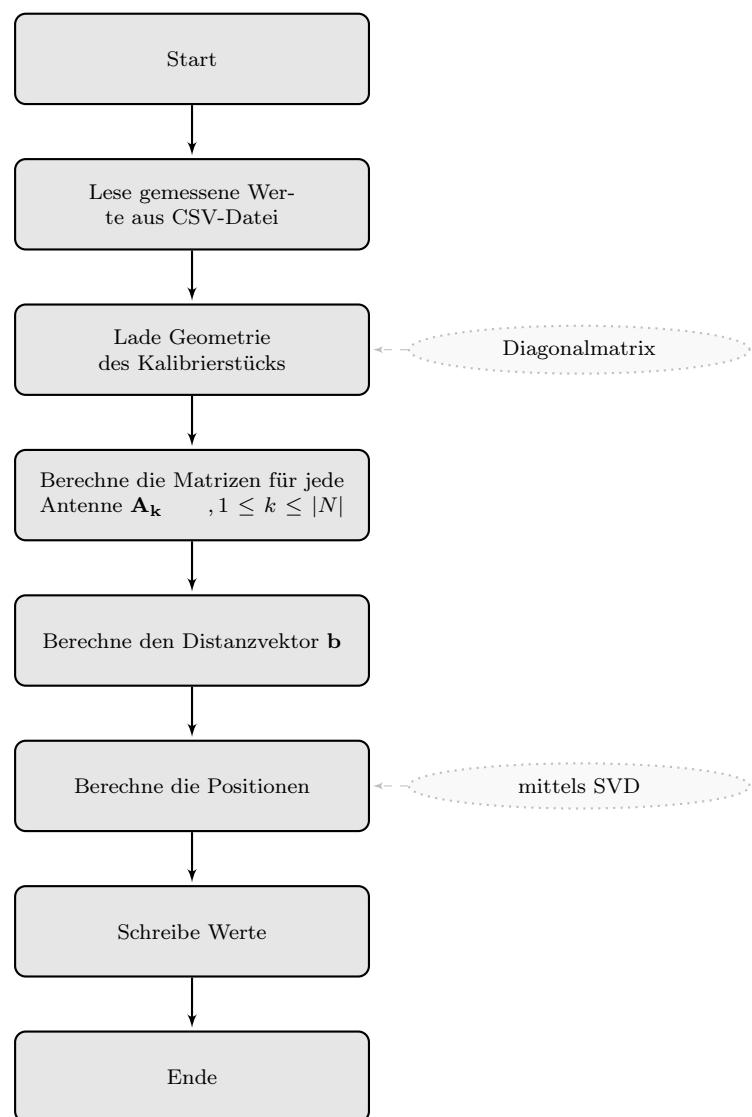


Abbildung 3.16.: Ablauf der libCalibration



3.8. Software

3.8.1. Shark

Shark ist eine Open-Source *C++*-Bibliothek für maschinelles Lernen und Optimierung [15]. Es implementiert Methoden für lineare und nicht lineare Optimierung, Kernel-basierte Lernverfahren, künstliche neuronale Netzwerke, und Weitere. Details der Umsetzung finden sich in [16]. Es wird sowohl in Forschung als auch in industriellem Umfeld eingesetzt und implementiert, nach eigenen Angaben, Algorithmen, die in anderen Bibliotheken nicht verfügbar sind. Shark baut auf den *Boost C++ Libraries* auf und verwendet CMake⁶. Dadurch ist es auf nahezu jeder Plattform verfügbar. Die Integration in ein Projekt ist sehr einfach und daher wird es in dieser Arbeit eingesetzt.

Eine Integration von Shark in dem Software Ökosystem ist aufgrund der vielen Features sehr sinnvoll. Weitere Entwicklungen können vom Funktionsumfang partizipieren.

3.8.2. Implementation

In diesem Abschnitt wird auf interessante Details der Softwareimplementation eingegangen. Es ist nicht möglich die gesamte Implementation der Software zu besprechen, dafür ist die Software zu Umfangreich⁷. Lediglich sollen hier die Implementationen verschiedener Evolutionsalgorithmen unter Verwendung von Shark gezeigt werden. Verschiedene evolutionäre Algorithmen wurden in Kapitel 2.4 besprochen. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Es wird beispielhaft die Implementation einer Objektfunktion beschrieben, wie sie typischerweise in Shark vorgenommen wird.

⁶CMake (cross-platform make) ist ein plattformunabhängiges Programmierwerkzeug für die Entwicklung und Erstellung von Software.

⁷Für diese Arbeit wurden ca. 5000 Zeilen Quellcode erstellt – einzelne Modelle nicht mitgerechnet

Objektfunktion in Shark

Als Beispiel für die Implementation einer Objektfunktion wird im Folgenden das Modell der evolutionären Kalibrierung besprochen. Dieses Modell, bzw. diese Objektfunktion, hat eine überschaubare Komplexität und wurde bereits im Abschnitt 3.7 zur Veranschaulichung verwendet. Daher eignet es sich gut um die Implementation zu zeigen. Im Rahmen dieser Arbeit sind eine Vielzahl von Modellen entstanden, die nicht in aller Ausführlichkeit diskutiert werden können.

Das Listing 3.1 zeigt die Headerdatei für die Implementation einer Objektfunktion. Sofort ist zu erkennen, dass sie von der abstrakten Klasse *SingleObjectiveFunction* abgeleitet ist. Diese ist eine von Shark bereitgestellte Klasse. Sie beschreibt die Funktionen, die eine Objektfunktion implementieren muss, damit sie von Optimizern verwendet werden kann. Ein von dieser Klasse abgeleitetes Modell erlaubt die Verwendung in verschiedenen Algorithmen. Die Funktion:

```
double eval(const SearchPointType &p) const;
```

wird in von den Optimizern aufgerufen und implementiert die eigentliche Funktion des Modells.

```
void proposeStartingPoint(SearchPointType &x) const;
```

Diese Methode wird von einem Solver beim Start aufgerufen und liefert passende Startwerte für das Modell zurück. Der Rückgabewert der Funktion ist ein Vektor der Dimensionalität $m_numberOfVariables$, dessen Werte in einem Intervall von $[-10, 10]$ gleichverteilt sind. Der Wert für $m_numberOfVariables$ wird bei der Instanziierung des Modells, also beim Aufruf des Konstruktors, gesetzt. Dort wird auch das die Eigenschaft des Modells *CAN_PROPOSE_STARTING_POINT* gesetzt. Diese teilt den unterschiedlichen Optimizern mit, welche Features vom Modell unterstützt werden. Hier geben wir bekannt, dass das Modell seine eigenen Startwerte generieren kann.

```
EvolutionaryCalibration() {
    m_numberOfVariables = Solve::ProblemDimensions::Calibration;
    m_features |= CAN_PROPOSE_STARTING_POINT;
}
```

Listing 3.1: Quellcodeschnipsel für die Deklaration einer Objektfunktion

```
struct EvolutionaryCalibration : public SingleObjectiveFunction {

    typedef AbstractOptimizer<shark::VectorSpace<double>,double,
        SingleObjectiveResultSet<typename shark::VectorSpace<double>::PointType> > base_type;

    typedef typename base_type::ObjectiveFunctionType
        ObjectiveFunctionType;

    EvolutionaryCalibration() {
        m_numberOfVariables = Solve::ProblemDimensions::Calibration;
```

```

        m_features |= CAN_PROPOSE_STARTING_POINT;
    }

    /// \brief From INameable: return the class name.
    std::string name() const
    { return "Evolutionary Calibration"; }

    std::size_t numberOfVariables() const{
        return m_numberOfVariables;
    }

    bool hasScalableDimensionality() const{
        return true;
    }

    void setNumberOfVariables( std::size_t numberOfVariables ){
        m_numberOfVariables = numberOfVariables;
    }

    void configure(const PropertyTree &node) {
        m_numberOfVariables = node.get("numberOfVariables", 51);
    }

    /**
     * Generate a starting value
     * @param[out] x The suggested search point
     */
    void proposeStartingPoint(SearchPointType &x) const {
        x.resize(numberOfVariables());
        for (unsigned int i = 0; i < 3; i++) {
            x(i) = Rng::uni(-10, 10);
        }
    }

    /**
     *
     */
    double eval(const SearchPointType &p) const;

    /**
     *
     */
    void setParams( const NRmatrix< Doub > &M,
                    const NRvector< Doub > &v
                ) {
        setMat(M);
        setVec(v);
    }

    /**
     */
    void setMat( const NRmatrix< Doub > &M ) {
        A = M;
        A_isSet = true;
    }

    /**
     */
    void setVec( const NRvector< Doub > &v ) {

```

```

        b = v;
        b_isSet = true;
    }

inline double mkII( const NRmatrix<Doub> &A, const double* x, const
    NRvector<Doub> &b ) const;

private:
    std::size_t m_numberOfVariables;

/* The Matrices we need to solve the Problem */
NRmatrix< Doub > A;
bool A_isSet = false;

/* The b-vector needed to find a Solution */
NRvector< Doub > b;
bool b_isSet = false;

};

```

Die in Listing 3.2 gezeigte cpp-Datei ist die Implementation der oben beschriebenen Modellfunktionen. Aus diesen beiden Dateien ist ersichtlich, dass die Deklaration des Modells sehr überschaubar ist. Praktische jedes Modell hat diese übersichtliche Struktur, was die Wartbarkeit enorm erhöht. Außerdem erlaubt es einen einfachen Austausch in der Implementation sowie die Verwendung in unterschiedlichen Algorithmen.

```

inline double EvolutionaryCalibration :: mkII( const NRmatrix<Doub> &A,
    const double* x, const NRvector<Doub> &b ) const

```

Diese Funktion ist die eigentliche Berechnung des Gleichungssystems der Form $\mathbf{Ax} = \mathbf{b}$. Die Lösung wird an die Aufrufende Funktion übergeben. Als Eingabe wird der Variablenvektor *const double*x* von Shark sowie die geom. Matrix **A** und der Distanzvektor **b** erwartet. Der vollständige Quellcode des Modells ist im Anhang B.1 und B.2 gelistet.

Listing 3.2: Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark

```

double EvolutionaryCalibration :: eval( const SearchPointType &p )
{
    m_evaluationCounter++;

    std::vector<double> res;

    double x[ m_numberOfVariables ];

    x[ 0 ] = p[ 0 ];
    x[ 1 ] = p[ 1 ];
    x[ 2 ] = p[ 2 ];
    return mkII( A, x, b );
}

inline double EvolutionaryCalibration :: mkII( const NRmatrix<Doub>
    &A, const double* x, const NRvector<Doub> &b ) const
{
    double res;
}

```

```

double prod_Ax[3] = {0.,0.,0.};
double x_[m_numberOfVariables];

x_[0]=x[0];
x_[1]=x[1];
x_[2]=x[2];

/* multiply the matrix with the vector */
for( int i = 0; i < A.nrows(); i++ )
    for( int j = 0; j < A.ncols(); j++ )
        prod_Ax[i] += A[i][j]*x_[j];

/* sum up */
res = (prod_Ax[0] - b[0]) * (prod_Ax[0] - b[0]);
res += (prod_Ax[1] - b[1]) * (prod_Ax[1] - b[1]);
res += (prod_Ax[2] - b[2]) * (prod_Ax[2] - b[2]);

return res;
}

```

Process MkII – Finde die Lösung

Die Klasse *ProcessMkII* steht im Zentrum der Lösungsfindung. Sie erlaubt die Ausführung und Parametrierung von verschiedenen, in dieser Arbeit erprobten Testfunktionen. Die Klasse hat verschiedene Konstruktoren um den verschiedenen Modellen gerecht zu werden. Diese erwarten häufig unterschiedliche Parameter. Bei der Instanziierung werden gleichzeitig alle wichtigen Parameter übergeben. Dies zeigt folgendes Schnipsel Quellcode:

```
Solve::Process_MkII process( A, b, names, MU, LAMBDA );
```

Das Listing 3.3 zeigt einen der implementierten Konstruktoren und seine Übergabeparameter. Dieser Konstruktor bereitet das Objekt auf das Modell 'Whole Tomato MkII' vor.

Listing 3.3: Quellschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark

```

Process_MkII(
    std::vector<NRmatrix< Doub >> Mats,
    std::vector<NRvector< Doub >> Vests,
    std::vector<std::string> Names,
    std::vector<std::vector<int>> IDs,
    double Epsilon
) : A( Mats ), b( Vests ), names( Names ), idxs( IDs ), epsilon(
    Epsilon )
{
    init();
}

```

Ein wichtiger Schritt ist noch für alle Solver durchzuführen. Eine Initialisierung des Zufallszahlengenerator muss durchgeführt werden. Das übernimmt die *init*-Funktion, gezeigt in Listing 3.4. Dort wird über den Aufruf 'shark::Rng::seed(seed)' der Generator initialisiert. Die Variable *seed* wird über die Systemzeit

gefüllt. Die Funktionen des in C++11 eingeführte Namespaces `'std::chrono'` erleichtern diese Aufgabe durch den einfachen Zugriff auf Zeitfunktionen.

Listing 3.4: Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark

```
void init()
{
    // Adjust the floating-point format to scientific and increase
    // output precision.
    std::cout.setf( std::ios_base::scientific );
    std::cout.precision( 10 );

    std::chrono::time_point<std::chrono::system_clock> now = std::chrono
        ::system_clock::now();

    auto duration = now.time_since_epoch();
    /* init the seed */
    auto seed = std::chrono::duration_cast<std::chrono::milliseconds>(
        duration).count();

    shark::Rng::seed( seed );
}
```

Nach erfolgreicher Instanziierung kann einfach das gewünschte Modell ausgeführt werden. Beispielsweise das Modell der evolutionären Kalibrierung wird unter Angabe der Problemdimension ausgeführt:

```
process.EvoCal( Solve::ProblemDimensions::Calibration );
```

Intern wird im Anschluss an diesen Aufruf eine Instanz des Solvers erstellt, hier: `shark::CMA`. Nach der Initialisierung des Solvers kann dieser ausgeführt werden. Das zeigt das Listing 3.5. Dort wird ein anderes Modell `'WholeTomatoMkII'` ausgeführt. Anhand der `while`-Schleife kann man die Konvergenzkriterien erkennen. Der Algorithmus läuft so lange, bis das gewünschte `epsilon` unterschritten oder die maximale Anzahl an Evaluationen erreicht ist.

Listing 3.5: Quellcodeschnipsel für die Implementation einer Objektfunktion für verschiedene Optimizer in Shark

```
int WholeTomatoMkII( int dimension ) {
    auto dim = Solve::ProblemDimensions::WholeTomatoMkII;
    dim += dimension;
    PRPSEvolution::Models::WholeTomatoMkII model( dim );

    model.setNumberOfVariables( dim );
    model.setParams( A, b, names );

    /* init the algorithm */
    shark::CMA cma;
    cma.init( model );

    do {
        cma.step( model );
    }
```

```

    } while(cma.solution().value > epsilon
    && model.evaluationCounter() < maxEvaluations);
}

```

3.8.3. Paralleler Ablauf

Die *ProcessMkII*-Klasse ist Multithreading-fähig. Eine Implementierung wurde im Rahmen der Arbeit durchgeführt. Durch das parallele Ablaufen verschiedener Optimierungen kann die Performance wesentlicher verbessert werden und so die Multi-Kern-Architektur aktueller Rechner ausgenutzt werden. Einen großen Beitrag zur schnellen Umsetzung lieferte der neue Standard C++11. Dieser implementiert ein High-Level Konstrukt für Threads und Tasks, sodass es einfacher ist, Abläufe zu Synchronisieren und Ergebnisse abzufragen. Zusätzlich ist die Implementation sehr leicht zu verwenden, wie im folgenden Listing gezeigt:

Listing 3.6: Erstellen von mehreren Threads bei gleichzeitiger Übergabe der Parameter

```

/* result vector; Solve::solveresult_t defines the return from the
model */
std::vector<std::future<Solve::solveresult_t<ChromosomeT<double>,
ChromosomeT<double>, Doub>>> results;

/* create tasks */
for( int Solution = 0; Solution < NO_OF_SOLUTIONS; Solution++ ) {

    t_0 = steady_clock::now();

    results.push_back( std::async( std::launch::async ,
        &Solve::Process::findSolution<Solve::solveresult_t<
            ChromosomeT<double>, Doub>>,
        &process ,
        A,
        b,
        Solve::ESStrategy::OnePlusOne ,
        duration_cast<microseconds>(t_1-t_0).count() ) );

    results.push_back( std::async( std::launch::deferred , &Solve::
        Process::findSolution , &process , A, b ) );

    t_1 = steady_clock::now();
}

```

Das Abfragen der Ergebnisse ist ähnlich einfach:

Listing 3.7: Abfragen der Asynchronen Ergebnisse mit Hilfe des auto-Datentyp. Ausgabe des Ergebnis in einer Datei.

```

for( auto res = results.begin(); res != results.end(); ++res ) {
    while(res->wait_for(chrono::seconds(0)) != future_status::ready);

    auto r = res->get();
    f_fitness << r.iterations << " " << r.fitness << " in: " << r.
        duration << " (us)" << " " << r.converged << std::endl;
}

```

```
for( auto values : r.valCont )
    f << values << "\t";
for( auto values : r.valDis )
    f << values << "\t";

f << std::endl;
```

3.8.4. Schnittstellen für Dateneingabe

Im Folgenden wird die implementierte Schnittstelle besprochen, mit der die Daten unter den Programmteilen ausgetauscht werden können. Die Schnittstelle umfasst im Wesentlichen zwei Teile:

1. Eingabe für die gemessenen Phasenwerte
2. Ausgabe für die ermittelten Wellenzahlen

Hinzukommt eine pseudo-Schnittstelle, über die die Systemparameter eingelesen werden können. Die aktuelle Implementation liest lediglich eine CSV-Eingabedatei mit den Systemparametern in eine entsprechende Struktur ein. Diese Parameter stehen anschließend dem PRPS-Evolutions-System zur Verfügung. Die Kommunikation zwischen dem PRPS-Dienst und dem PRPS-Evolutions-System ist einfach. Der Dienst teilt die gemessenen Phasendaten mit, die vom PRPS-Evolutions-System für die Berechnung der Wellenzahlen verwendet werden. Das bedeutet, dass der Ablauf des Auffindens der Wellenzahl vom PRPS-Dienst als 'Black-Box' angesehen wird.

3.8.5. Ablaufdiagramme

Vieles der Funktionalität der Software ist zu umfangreich für dieses Dokument. Es kann in diesem Rahmen keine vollständige Besprechung des Quellcodes durchgeführt werden. Wichtige Funktionalitäten und Abläufe werden in Ablaufplänen zusammenfassend dargestellt.

Abbildung 3.17.: Prozessschritte nach Start des Programms. Mit dem Punkt "Ende" ist nicht das Ende des Programms gemeint, sondern das Ende des Programmstarts. Nach diesen Schritten stehen alle statisch generierbaren verfügbaren Daten für das Programm zur Verfügung.

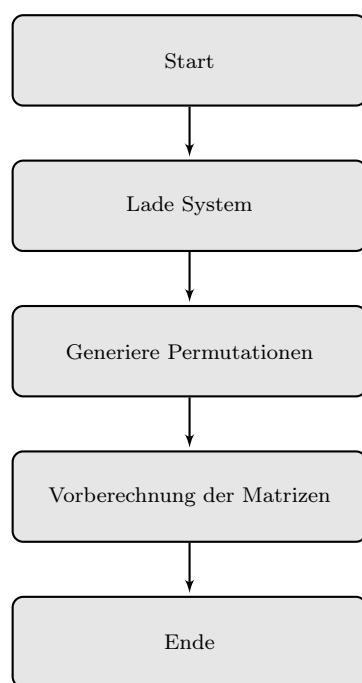


Abbildung 3.18.: Schritte die zur Lösungsfindung abgearbeitet werden.
 Dies ist ein Teil des Hauptprogramms, der die erstellten Informationen vom Programmstart in ein Modell zum Auffinden einer Lösung übergibt. Es werden eine beliebige Anzahl an Lösungen generiert.

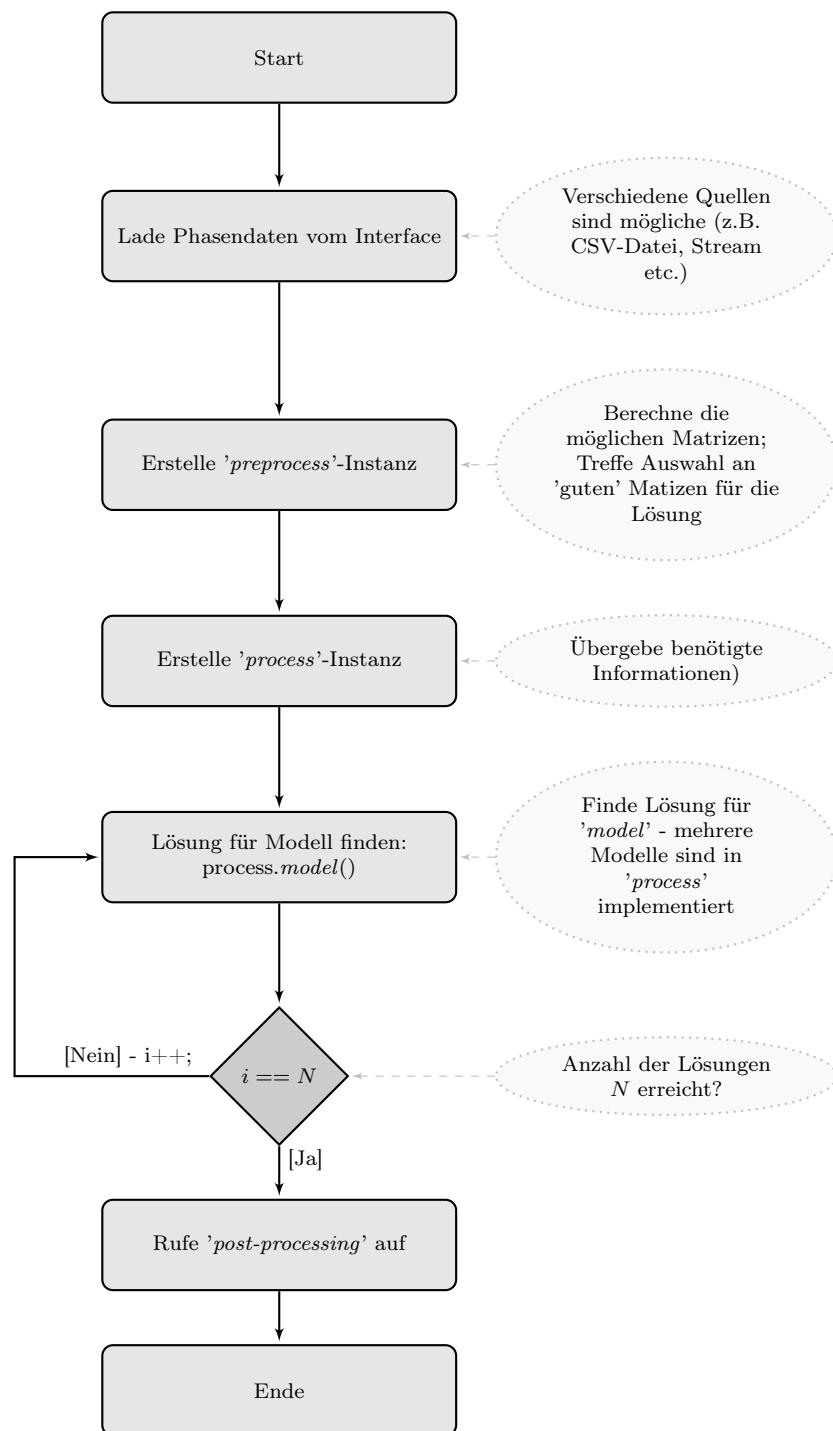
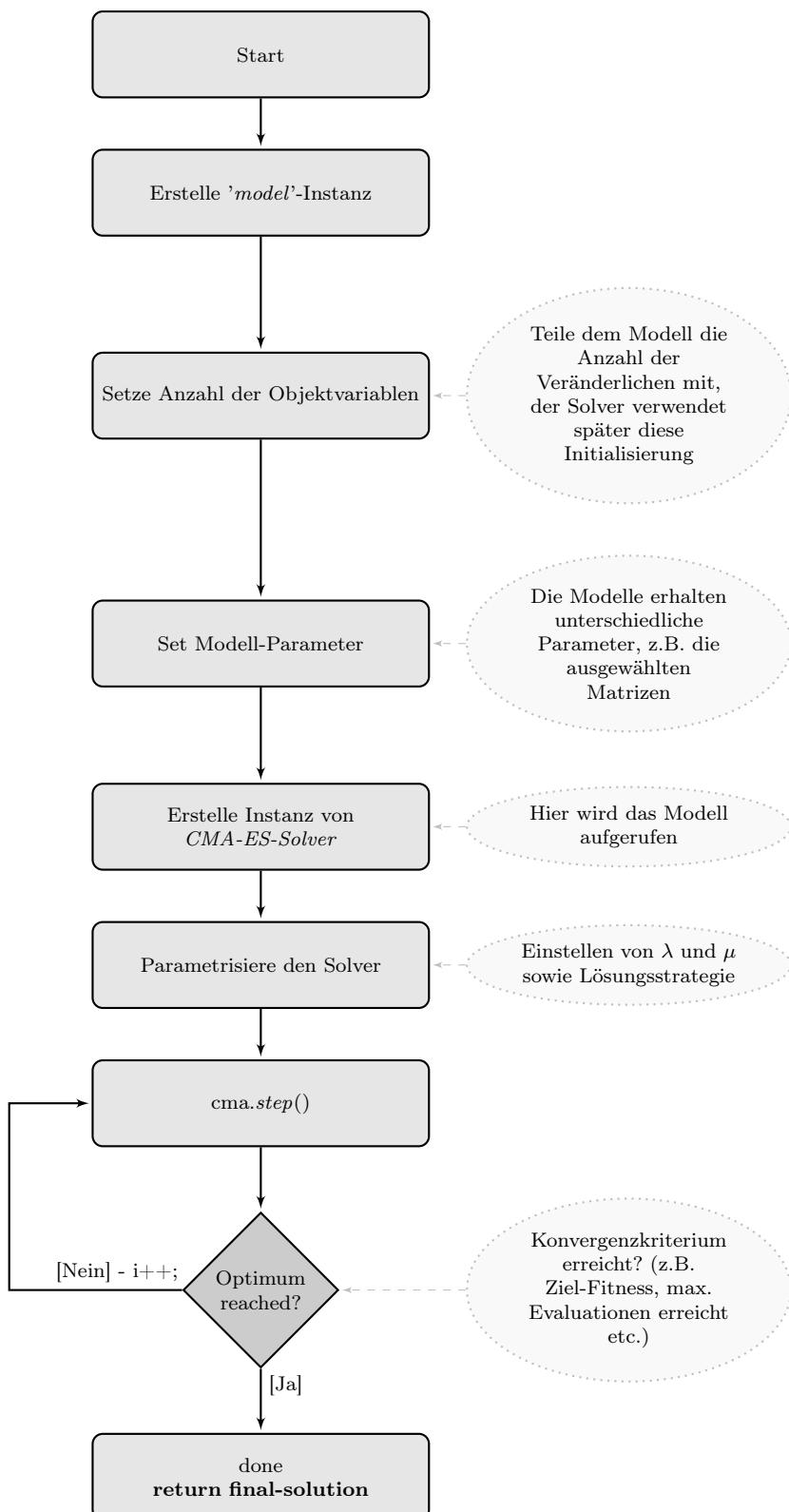


Abbildung 3.19.: Die Grafik zeigt den Ablauf im Modell um eine Lösung zu finden. Es werden so lange Evolutionsschritte durchgeführt bis das Konvergenzkriterium erreicht wird.



4. Ergebnisse

In diesem Kapitel werden die Ergebnisse der Arbeit präsentiert. Zuerst werden künstliche Werte für die gemessene Phase als Eingabe für den Algorithmus verwendet. Darauf folgend werden reale Messdaten als Input verwendet und die Resultate verglichen.

4.1. Ergebnisse der Kalibrierung

Es werden die Ergebnisse der Kalibrierung vorgestellt. Für eine der vermessenen Antennenkonfigurationen sind in der folgenden Tabelle die Koordinaten der Antennen gezeigt. Die Visualisierung der Konfiguration zeigt die Abbildung 4.5. Eine Berechnung mit dem evolutionären Verfahren dauerte ca.

Antenne	x	y	z	d_{meas}	d_{result}	ε_{abs}	ε_{rel}
1	0.48	-1.01	0.60	1,259	1,274	0,015	1,14%
2	-0.77	-1.04	1.34	1,894	1,872	-0,022	1,19%
3	1.52	-1.05	1.37	2,334	2,307	-0,027	1,15%
4	-0.92	-0.19	1.32	1,661	1,628	-0,033	2,01%
5	1.92	0.03	1.39	2,399	2,375	-0,024	1,01%
6	-0.55	1.09	1.43	1,851	1,887	0,036	1,93%
7	1.06	1.07	1.35	2,055	2,031	-0,024	1,19%
8	0.45	1.35	0.67	1,574	1,578	0,004	0,26%

Tabelle 4.1.: Tabelle der finalen Antennenkoordinaten [m], berechnet mit dem in dieser Arbeit entwickelten Modell und dem SVD-Verfahren. Die Ergebnisse wurden auf zwei Nachkommastellen gerundet und sind identisch für beide Methoden. Die Spalte d_{result} enthält die von der Berechnung gefundenen Distanz vom Referenzpunkt zur Antenne. Die Spalte d_{meas} zeigt die gemessenen Werte. Die ϵ -Spalten zeigen die Abweichung.

170 ms mit dem SVD-verfahren wurde eine Lösung in ≤ 1 ms gefunden. Für die in der Praxis eingesetzte Software wird es eine Implementation der Kalibrierung mit dem SVD-Verfahren geben. Das mit dieser Variante berechnete

Ergebnis wird bei Bedarf mit einer Lösung des evolutionären Verfahrens verglichen. Das ermöglicht eine Build-In Verifikation der Kalibrierung.

4.1.1. Visualisierung der Ergebnisse

Für die in den folgenden Abbildungen präsentierten Ergebnisse wurden insgesamt 100 Durchläufe des Algorithmus erstellt. Die Ergebnisse wurden mit einem vom Algorithmus selbst erstellten μ und λ gefunden. In Abbildung 4.1 wird eine statistische Auswertung der Ergebnisse gezeigt. In jedem Plot werden die Endwerte der Lösungen in einem sog. Boxplot gezeigt. Dabei wird die Verteilung mit Hilfe von Boxen dargestellt. Die Fähnchen der Boxen stellen die maximal- bzw. minimal-Werte dar. Die Größe der Boxen enthält das obere und untere Quartil der Daten, der horizontale Strich in der Box zeigt den Mittelwert der Daten. Ausreißer¹ in den Daten werden durch Punkte abseits der Box dargestellt.

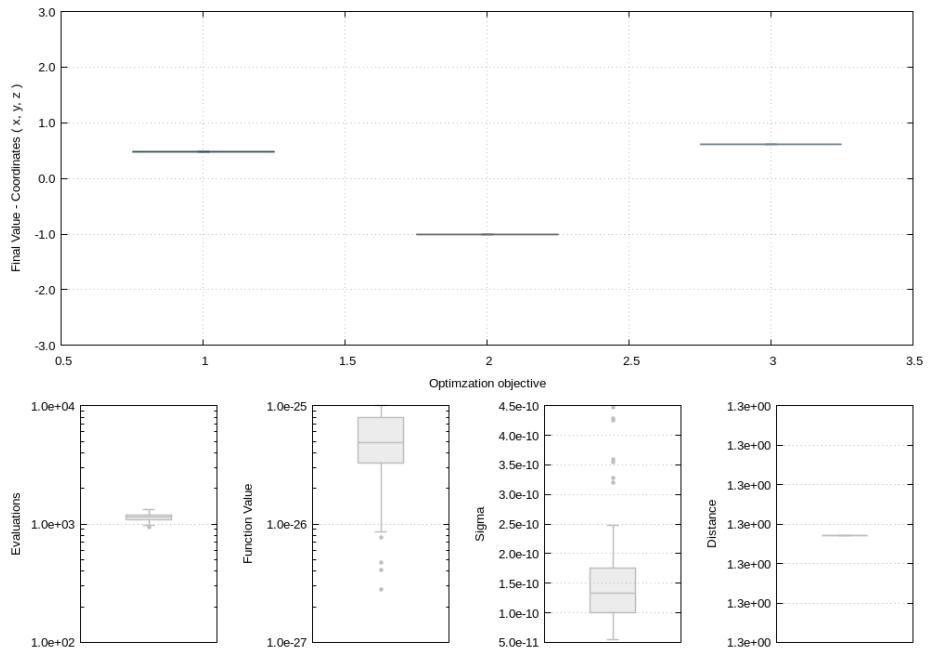
Die Abbildung 4.1 zeigt den Verlauf der drei Objektvariablen (x, y, z -Koordinaten) sowie die Entwicklung der Fitness und des mittleren Sigmas. Als Darstellungsart wird der Liniенplot verwendet und die Verläufe einzelner Lösungen überlagern sich in diesem Plot. Das Abbruchkriterium war eine Fitness von $\leq 10^{-25}$. Für Darstellungszwecke wurde die x -Achse nach 500 Werten beschränkt, daher erreicht der Fitness-Plot diesen Wert in der Abbildung nicht. Der Verlauf ist typisch für den verwendeten Algorithmus. Deutlich zu erkennen ist eine Verbesserung des Ergebnisses mit steigender Zahl der Generationen. Der Verlauf der Variablen ist immer für den erfolgreichsten Nachkommen einer Generation dargestellt.

Abbildung 4.3 ist ein Scatter-Plot. Die Objektvariablen werden hier gegeneinander aufgetragen. Auf der Diagonalen befinden sich stets die Variable gegen sich selbst aufgetragen, daher zeigt sich dort immer eine Linie, bei streuenden Ergebnissen, bzw. ein einzelner Punkt, sollten die Ergebnisse nicht streuen. Der Plot ist praktisch um die Implementation des Algorithmus zu verifizieren. Er lässt Rückschlüsse auf Abhängigkeiten und Einflüsse der Objektvariablen zu. So können die Ergebnisse mit den Erwartungen an die Verläufe verglichen werden.

Das Modell aus 3.2 wurde in verschiedenen Experimenten untersucht. Dabei wurden die Parameter der Optimierung variiert und die Auswirkungen untersucht.

¹Hier Werte die mindestens den 2-Fachen Wert des oberen- bzw. unteren-Quartils aufweisen

Abbildung 4.1.: Boxplot des Kalibierergebnis aus 100 Durchläufen. Im oberen Plot sind die x,y,z-Koordinaten gezeigt, diese landen in allen Durchläufen auf dem selben Ergebnis. Was nicht verwundert, das Problem ist eines der Art „drei Gleichungen und drei Unbekannte“. Die Streuung der Lösung zeigt sich in der Breite der Linien. Die unteren vier Plots zeigen die Anzahl der Evaluationen der Fitness-Funktion, den finalen Funktionswert, das Sigma für die Variablen und die Entfernung zum Referenzpunkt (v.l.n.r.). Das Ergebnis sind die Koordinaten für Antenne 1



Es wird analysiert, wie gut die Lösbarkeit des unregistrierten Problems ist. Dazu werden zuerst für jede Referenzantenne immer eine mögliche Konfiguration gewählt und das Ergebnis aus M -Durchläufen untersucht. Im Anschluss

Zunächst werden die Ergebnisse der Experimente für idealen Messwerte vorgestellt, dann wird dieselbe Darstellung für reale Messdaten vorgenommen. Die Darstellung der Ergebnisse ist eine kondensierte Form der Präsentation. Sie Zeigen

Abbildung 4.2.: Zu erkennen ist, dass nach ca. 300 Evaluationen der Zielfunktion keine großen Änderungen der Variablen zu erkennen sind. Bis zum erreichen des Abbruchkriteriums (Function Value $\leq 10^{-25}$) werden noch ca 400 Evaluationen benötigt, vgl. korrespondierender Boxplot.

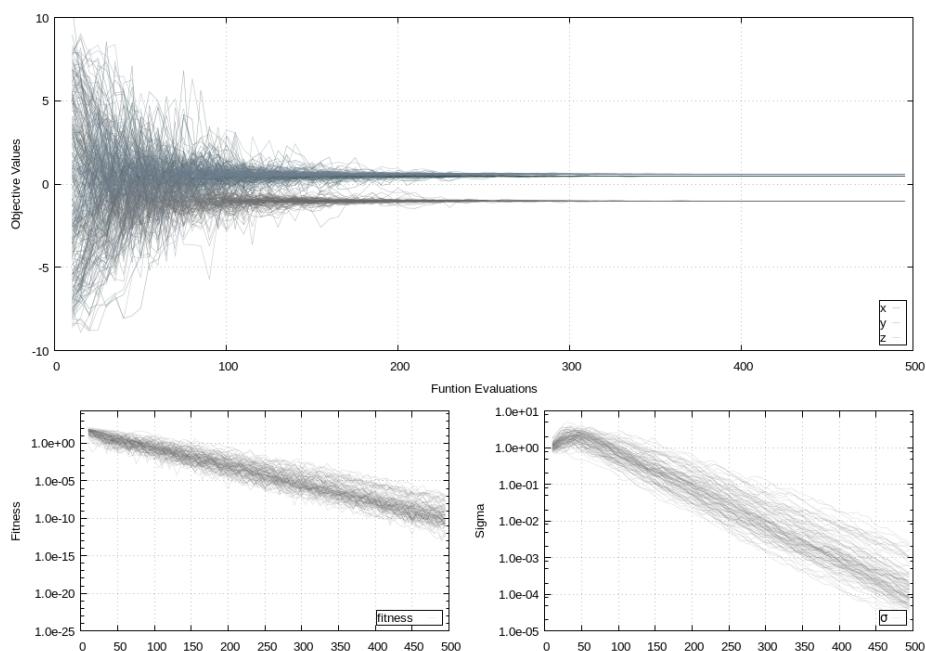
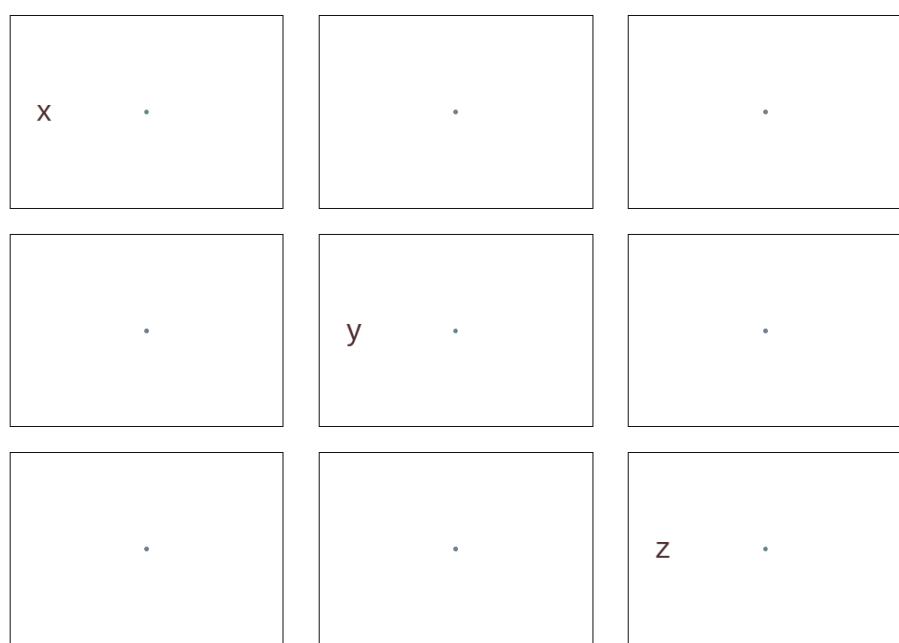
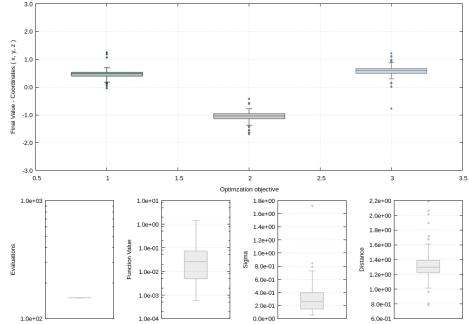
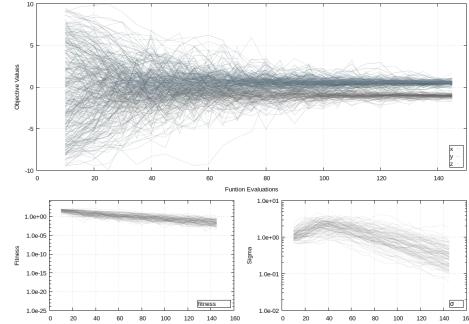


Abbildung 4.3.: Scatter-Plot der Ergebnisse der evolutionären Kalibrierung. Die Endergebnisse streuen in keiner Dimension, das wird aus dieser Darstellung deutlich.

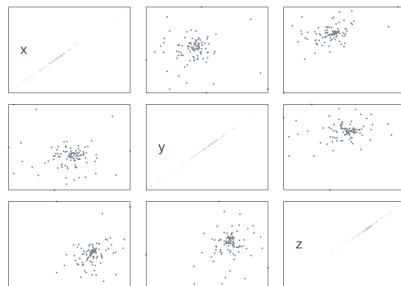




(a) Statistisch verteilte Endwerte für die Koordinaten der Kalibrierung.



(b) Linienplot der bei 140 Evaluierungen abgebrochenen Verläufe. Gut zu sehen ist der Verlauf der Objektvariablen, die sich von Generation zu Generation dem realen Wert nähern.



(c) Statistische Streuung um einen Mittelwert. So in etwa kann man die Lösungen der Komplexen Probleme erwarten.

Abbildung 4.4.: Analog zu den Abbildungen 4.4b, 4.4a und 4.4c zeigen die Plots die gleichen Darstellungen. Hier gezeigt wird, wie sich eine statistische Verteilung in den Plots manifestieren würde. Um das zu demonstrieren wurde das Abbruchkriterium auf lediglich 150 Evaluationen der Zielfunktion eingestellt. Zu diesem Zeitpunkt können die Objektvariablen bereits einen passablen Wert erreicht haben oder noch abweichende Werte aufweisen (vgl. 4.2).

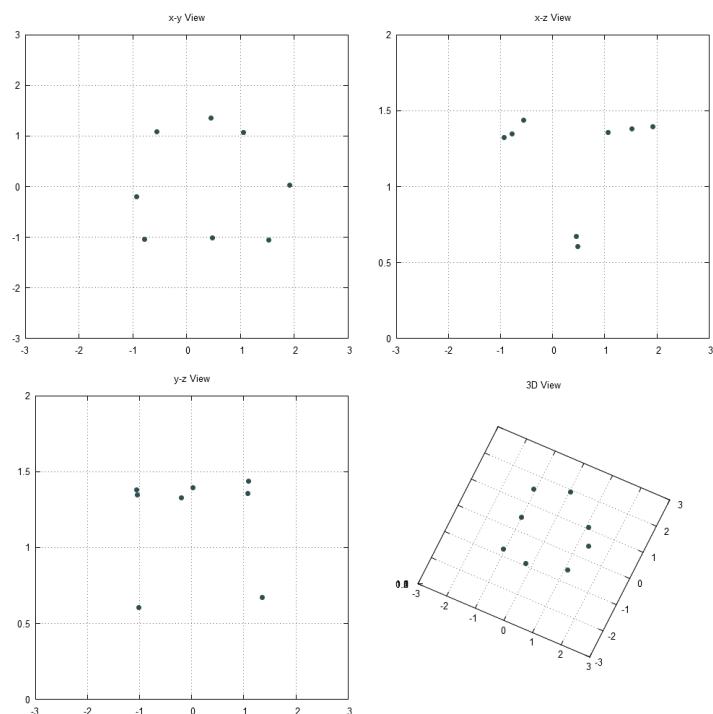


Abbildung 4.5.: Visualisierung des Kalibrierendergebnis. Abgebildet sind die gefundenen Antennenkoordinaten (Punkte) in drei Raumansichten. Die zusätzliche, dreidimensionale Ansicht dient der Übersicht. Das Ergebnis und die reale Anordnung decken sich sehr gut. Siehe Tabelle 4.1

4.2. Ergebnisse des Trilaterationsmodells

Dieser Abschnitt enthält die Ergebnisse der evolutionären Optimierung für das entwickelte Modell der Trilateration. Es werden Experimente definiert und ihre Resultate als Falschfarbenbild (Heatmap) dargestellt. Es werden die Ergebnisse von künstlichen Eingabedaten und realen Messdaten verglichen. Im Anschluss wird der Verlauf der Optimierung auf Besonderheiten hin untersucht. Abschließend wird die Performance der evolutionären Lösung visualisiert.

4.2.1. Experimente

Die Tabellen 4.2 und 4.3 geben an welche Parameter variiert wurden. Diese Experimente wurden für die Präsentation der Ergebnisse in dieser Arbeit entworfen. Alle wurden mit der gleichen Objektfunktion durchgeführt. Diese trägt den Namen '*Whole Tomato MkII*'². Der eingesetzte Algorithmus ist das CMA-ES. Andere evolutionäre Strategien werden nicht in separaten Experimenten untersucht und präsentiert. Die spätere Darstellung kondensiert die Parameter und Erkenntnisse die in dieser Arbeit gewonnen wurden. Ziel ist es insgesamt die Lösung zu quantifizieren. Wie im Rahmen der Komplexitätsuntersuchung beschrieben ist die Fragestellung die hier bearbeitet wird recht komplex. Es werden daher die Parameter der Evolutionsstrategie variiert. Dabei wird untersucht ob und wie sich die Güte der Lösung durch diese Parameter verbessern lässt. In den Abbildungen 4.6 und 4.7 sind die Ergebnisse der Experimente als Heatmap dargestellt.

4.2.2. Ergebnisse ideale Messwerte

Um die idealen Messwerte zu generieren wurde ein eigenes Modul entwickelt. Es verwendet die in Kapitel 2.6.4 beschrieben Formeln. Es wurden eine Reihe von Punkten definiert, von denen die idealen Phasenwerte ermittelt wurden. Um im Rahmen dieser Arbeit in der Präsentation der Ergebnisse konsistent zu bleiben, wird als Punkt der Ursprung der Kalibrierung genutzt. Dieser muss von dem Algorithmus wiedergefunden werden. Seine Koordinaten wurden bereits in Kapitel 3.7 besprochen. Die in den Abbildungen gezeigte Darstellung stellt die Abweichung von dem korrekten Wert dar.

$$\varepsilon = | d_{true} - d_{found} |$$

Die Dimension von ε ist dementsprechend [m]. Für die Visualisierung wurden auf den bestimmten M -Lösungen der Median-Wert (mittlere) genommen.

²Implementation des in dieser Arbeit entwickelten Trilaterationsmodells

Tabelle 4.2.: Aufstellung der Experimente die in diesem Abschnitt vorgestellt werden. Die Gruppengröße wird bei jedem Experiment Variiert. Jedes Unterexperiment erhält seinen eigenen Namen. Daher steigt der Name der Experimente bei jedem Eintrag entsprechend der Anzahl der untersuchten Gruppen an. In dieser Untersuchung wurden auch Parameter des Evolutionären Algorithmus variiert. Es werde μ und λ variiert. Der Einfluss der Populations- und Gruppengröße wird damit gleichzeitig untersucht. Es ergeben sich 100 einzelne Experimente.

Name	Trials M	Gruppengröße L	$\mu + \lambda$
E2000	50	1-10	(30+100)
E2010	50	1-10	(40+150)
E2020	50	1-10	(50+200)
E2030	50	1-10	(60+250)
E2040	50	1-10	(70+300)
E2050	50	1-10	(80+350)
E2060	50	1-10	(90+400)
E2070	50	1-10	(100+450)
E2080	50	1-10	(110+500)
E2090	50	1-10	(120+550)

Tabelle 4.3.: Für bestmögliche Vergleichbarkeit wurden diese Experimente mit den gleichen Parametern durchgeführt wie die für ideale Daten. In diesem Experiment werden die realen Messwerte des PRPS in der Positions berechnung verwendet.

Name	Trials M	Gruppengröße L	$\mu + \lambda$
E3000	50	1-10	(30+100)
E3010	50	1-10	(40+150)
E3020	50	1-10	(50+200)
E3030	50	1-10	(60+250)
E3040	50	1-10	(70+300)
E3050	50	1-10	(80+350)
E3060	50	1-10	(90+400)
E3070	50	1-10	(100+450)
E3080	50	1-10	(110+500)
E3090	50	1-10	(120+550)

4.2.3. Ergebnisse ideale Messwerte

Die idealen Messewerte zeigen insgesamt ein gutes Ergebnis (Abbildung 4.6). Im Mittel werden die besten Ergebnisse, bei mittlerer Gruppengröße und großer Populationsgröße gefunden. Es zeigt sich, dass eine steigende Populationsgröße das Ergebnis in der Regel verbessert. Bei der Gruppengröße kann ab einer bestimmten Größe keine weitere Verbesserung festgestellt werden. Eine Besonderheit zeigt sich bei den Antennen 3 und 5. Diese zeigen ein umgekehrtes Verhalten, bei niedriger Gruppengröße und kleiner Populationsgröße zeigen sie die geringste Abweichungen. Woran das liegt konnte nicht weiter untersucht werden. Die meisten Lösungen zeigen eine Abweichung von unter einer Wellenlänge, die wir mit $\lambda \simeq 35$ cm angeben konnten. Das erlaubt prinzipiell eine korrekte Berechnung der Wellenzahl.

In Abbildung 4.8 wurden die Ergebnisse anders eingefärbt, es werden Abweichungen über 35 cm konsequent rot eingefärbt. Damit zeigt sich deutlicher, dass eine recht große Population und eine geeignete Gruppengröße gewählt werden muss um vernünftige Resultate zu erhalten.

4.2.4. Ergebnisse reale Messwerte

Die Ergebnisse (Abbildung 4.7) für reale Messwerte zeigen ein ähnliches Bild wie die idealen Messwerte. Eine steigende Gruppen- und Populationsgröße verbessert das Ergebnis in der Regel. Es fehlen die Plots der Antenne 2 und 6. Das liegt daran, dass diese beiden Antennen keine Messdaten lieferten. Es wurde der selbe Punkt ausgewählt, wie bei Vorstellung der ideal Messergebnisse. Damit sind die Ergebnisse unmittelbar vergleichbar. Genau wie bei den idealen Messdaten wurde der Ursprung der Kalibrierung als Messpunkt genommen. Das Ergebnis der realen Messwerte mutet sogar sicherer an, als das der idealen Phasenwerte. Der Grund dafür kann nicht exakt angegeben werden. Man kann sich dazu überlegen, da jeder Messwert ein Messrauschen enthält. Dieses Rauschen wird Teil des Modell und variiert dadurch unmittelbar die Fitnesslandschaft. Das Aussehen der Fitnesslandschaft wurde in Kapitel 3.6 untersucht. Weitere Untersuchungen und größere Messreihen müssen dieses Verhalten bestätigen.

Auch die neu kolorierte Darstellung der Ergebnisse in Abbildung 4.9 zeigt, wie die der idealen Messwerte, eine Verbesserung der Positionsbestimmung für steigende Populations- und Gruppengrößen.

4.2.5. Evolutionsverlauf - real vs ideal

Abschließend ist in Abbildung 4.10 der Verlauf der letzten beiden Experimente 2099 und 3099 gezeigt. Dort werden in 3 Plots die Lösungen charakte-

risiert und die statistischen Eigenschaften sowie der Verlauf der Optimierung quantifiziert. Zu erkennen ist, dass sie die beiden Ergebnisse grundsätzlich ähneln. Unterschiede zeigen sich nur in Einzelheiten. So zeigt sich z.B., dass die Streuung der Parameter bei idealen Messwerten geringer ist, jedoch mehr Evaluationen notwendig waren. Allgemein zeigt sich der erwartete Verlauf einer evolutionären Optimierung.

4.2.6. Performance

Abschließend soll die Performance der Experimente gezeigt werden. Dabei wird die gleiche farbkodierte Darstellung verwendet, die auch bei den anderen Ergebnissen verwendet wird. Die Ausführungszeiten der Experimente sind in der Abbildung 4.12 und Abbildung 4.13 als Falschfarbenbild dargestellt. Die maximale Ausführungszeit betrug ~ 1200 ms. Durch eine vernünftige Wahl von Populations- und Gruppengröße würde ein Ergebnis zwischen 400 – 600 ms in Anspruch nehmen. Die Performance lässt sich durch Multi-threading weiter verbessern.

Abbildung 4.6.: Ergebnisse verschiedener Experimente mit idealen Messwerten. Die Abbildung zeigt farbkodiert den Betrag der Abweichung vom wahren Wert der Entfernung (Ausgemessen). Zu beachten gilt, dass die Farbskala eine unterschiedliche Skalierung hat und somit jede Antenne separat betrachtet werden muss. Die Parameter die in diesen Experimenten variiert wurden sind die Populationsgröße (μ und λ) und die Gruppengröße L . Mit der x -Achse steigt die Populationsgröße an. Die y -Achse trägt die steigende Gruppengrößen auf.

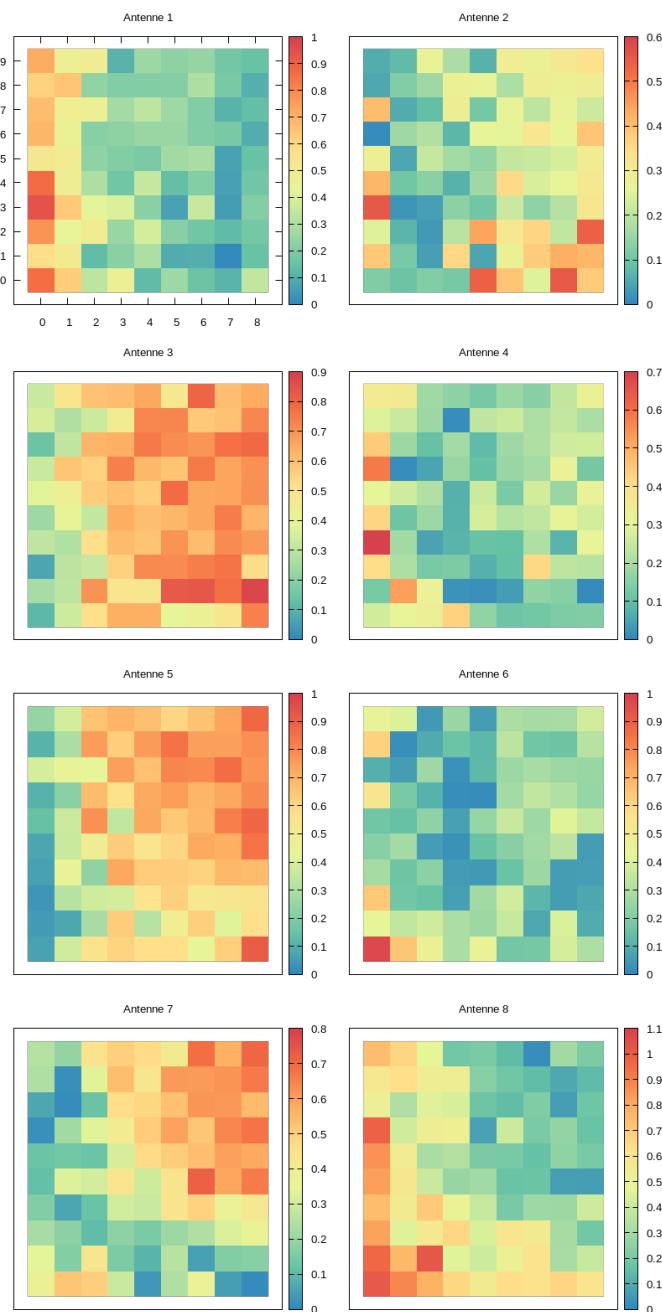


Abbildung 4.7.: Ergebnisse mit realen Messwerten. Die Antennen 2 und 6 lieferten bei dieser Messung keine Messdaten. Daher fehlen diese Plots in der Grafik. Das Antennen keine Daten liefern ist der Regelfall den man in der Praxis begegnet. Das Ergebnis ist in etwa identisch mit dem der idealen Messwerte. Auch hier finden sich gute Ergebnisse bei mittlerer Gruppen- und großer Populationsgröße.

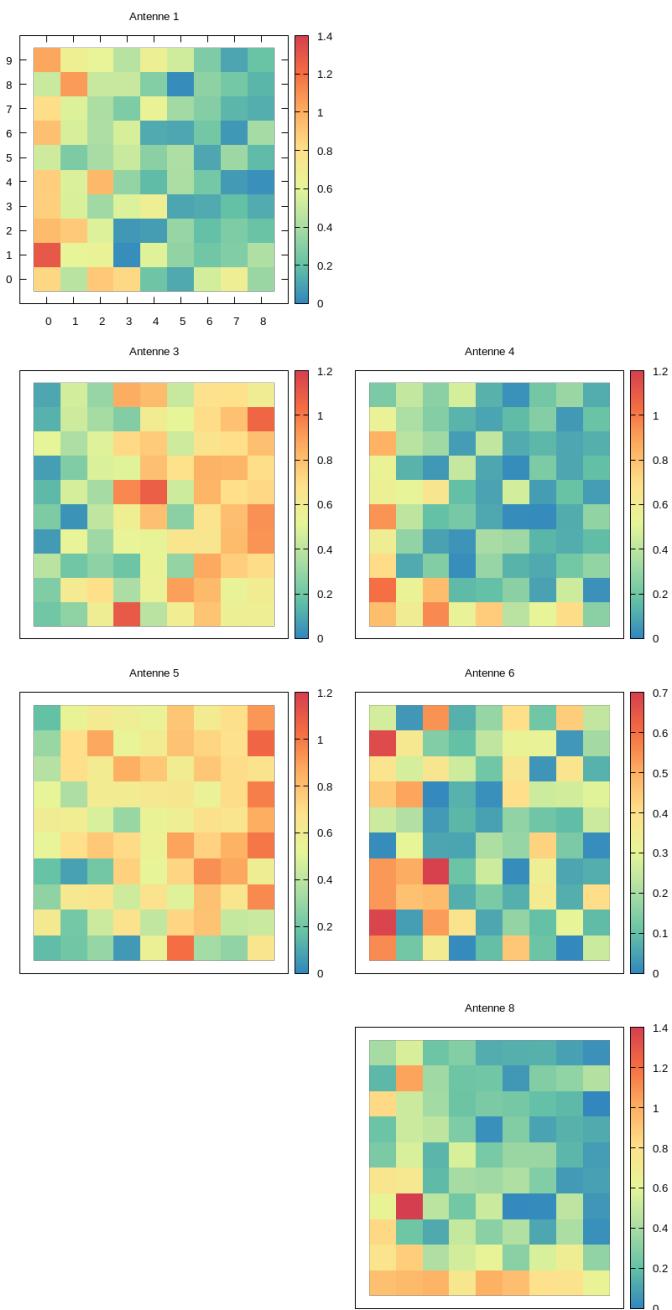


Abbildung 4.8.: Alternative Visualisierung der zuvor gezeigten Ergebnisse, *idealer* Messdaten. Zur besseren Übersicht wurde die Farbskala auf eine Wellenlänge reduziert. Abweichungen ≥ 35 cm werden rot eingefärbt. Gut 50% der Werte führen nicht zu einem zufriedenstellenden Ergebnis.

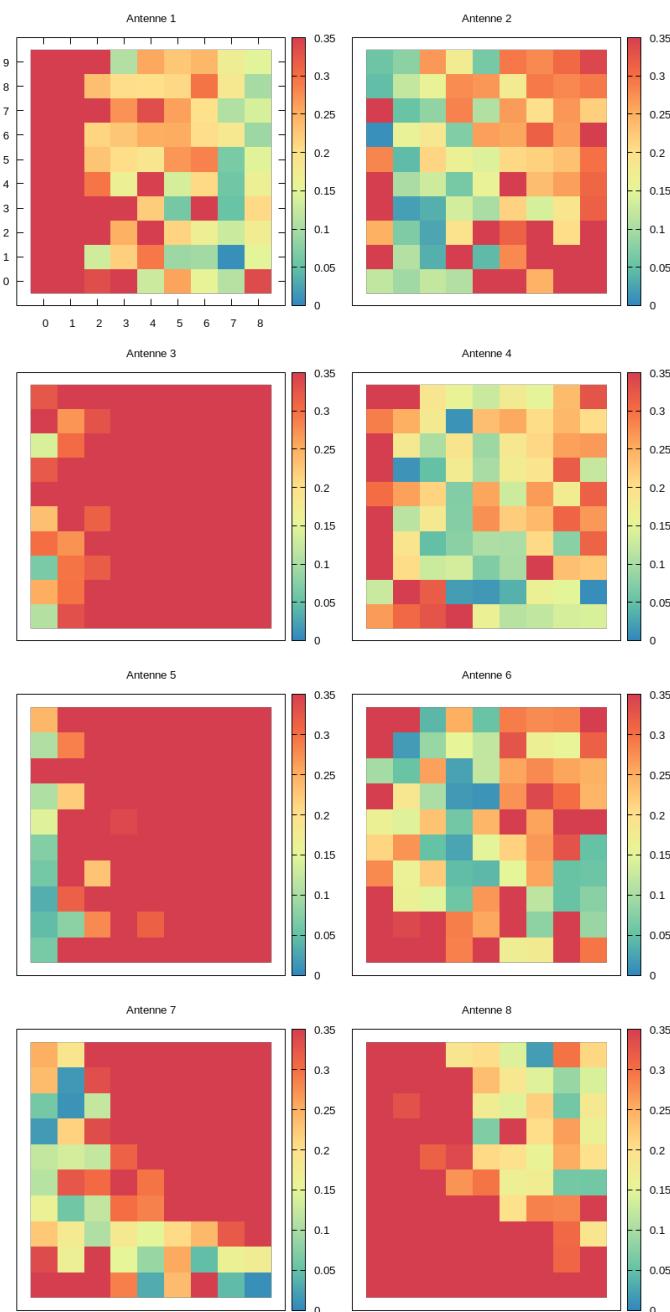


Abbildung 4.9.: Alternative Visualisierung der zuvor gezeigten Ergebnisse, *realer* Messdaten. Abweichungen ≥ 35 cm werden rot eingefärbt. Gut 50% der Werte führen nicht zu einem zufriedenstellenden Ergebnis. Das Ergebnis gleicht dem der idealen Messdaten.

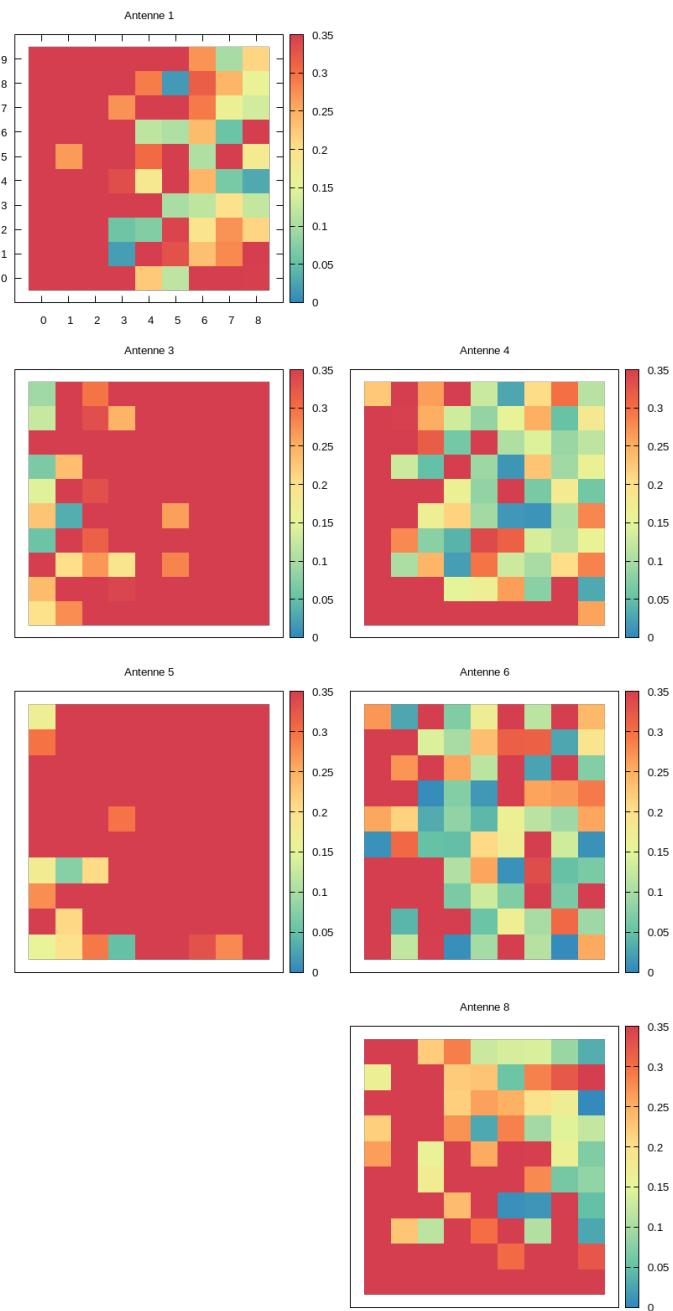
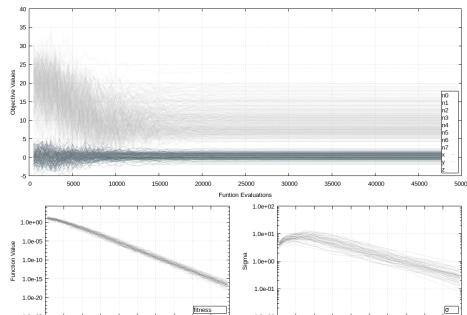
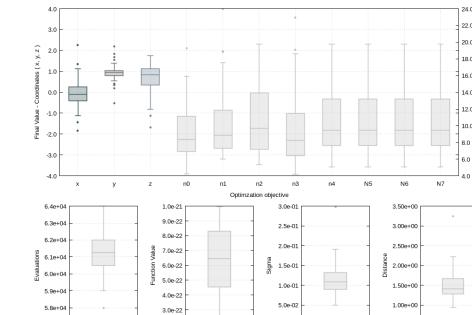


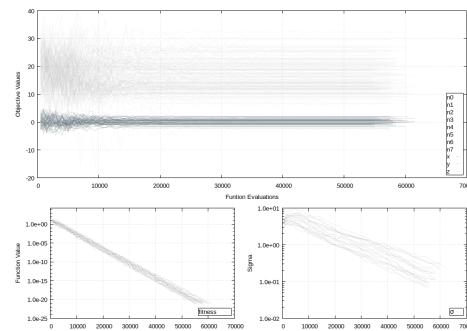
Abbildung 4.10.: Diese Grafik zeigt den Verlauf der Evolution. Es werden die Beiden letzten Experimente 2099 (oben, ideale Messdaten) und 3099 (unten, reale Messdaten) gezeigt. Diese Plots dienen nur der Einschätzung über den generellen Verlauf der Evolution. Es ist nicht sinnvoll sie für alle Experimente hier darzustellen. Anhand des Boxplots (Mitte) erkennend man, dass die Resultate für ideale Messwerte nicht so stark streuen, die Lösung der realen Messdaten ist den der idealen mind. Ebenbürtig. Es zeigt sich sogar, dass weniger Evaluationen der Zielfunktion bei den realen Werten nötig waren.



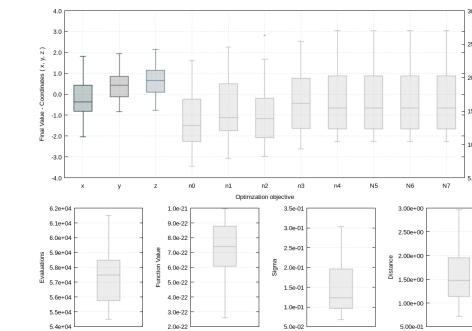
(a) Liniensplot zur Analyse der Variablen und Optimierungsverlaufs.



(b) Boxplot zur Veranschaulichung der Lösungsstatistik



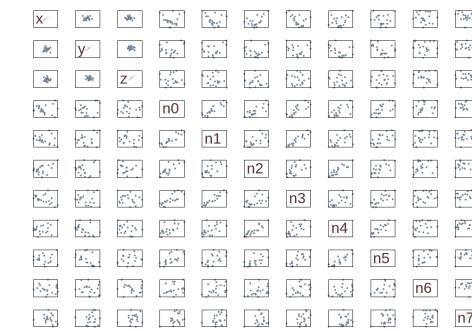
(d) Liniensplot zur Analyse der Variablen und Optimierungsverlaufs.



(e) Boxplot zur Veranschaulichung der Lösungsstatistik



(c) Streuplot zur Untersuchung von Abhängigkeiten der Parameter



(f) Streuplot zur Untersuchung von Abhängigkeiten der Parameter

Abbildung 4.12.: Visualisierung der Performance. Die Resultate wurde in einer virtuellen Umgebung generiert. Aufgrund limitierter Ressourcen (nur ein Prozessor verfügbar). Konnte die Multithreading-Fähigkeit nicht genutzt werden. Die Ausführungszeiten für vernünftige Konfigurationen liegen bei ~ 800 ms ohne Threading. Gut zu erkennen ist, dass mit steigendem Aufwand die Ausführungszeit, wie erwartet steigt.

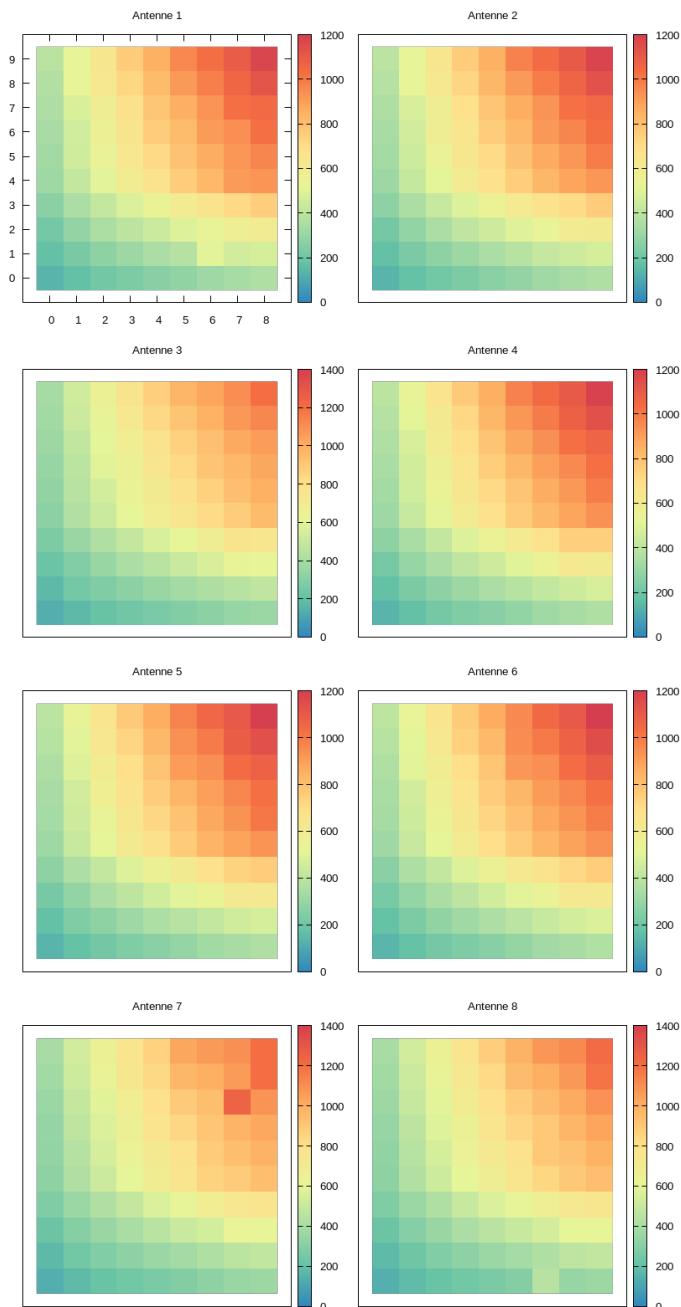
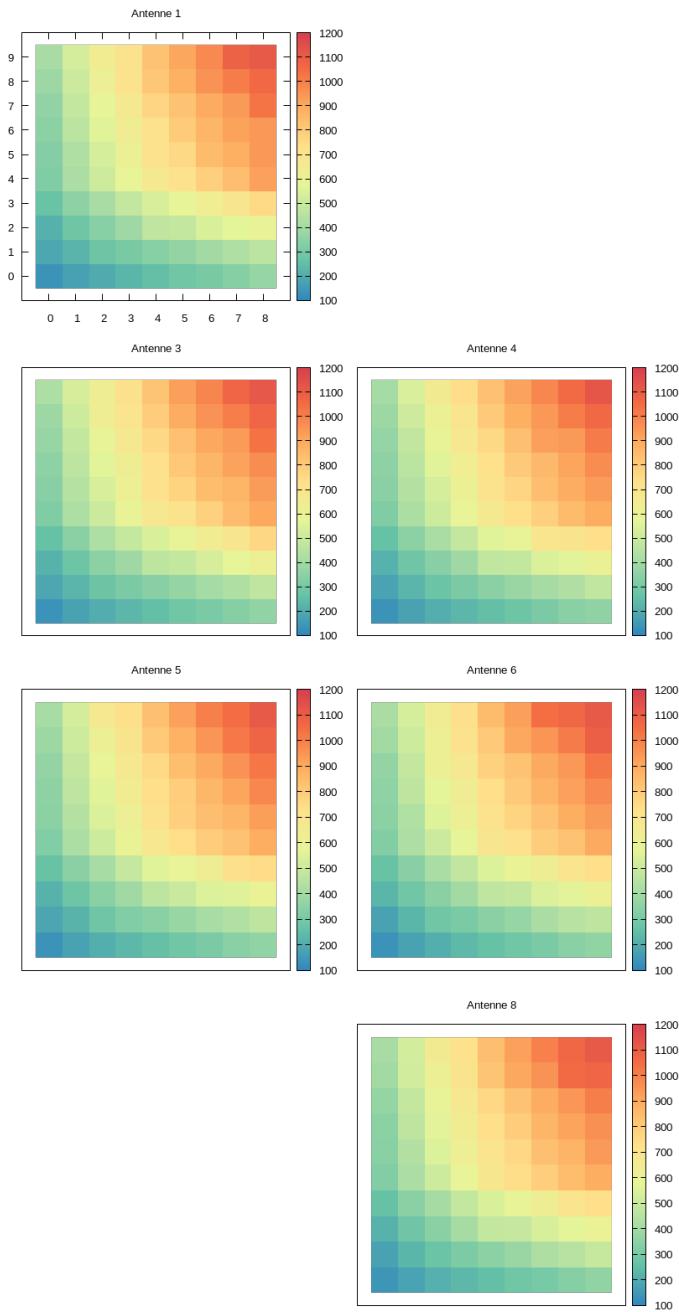


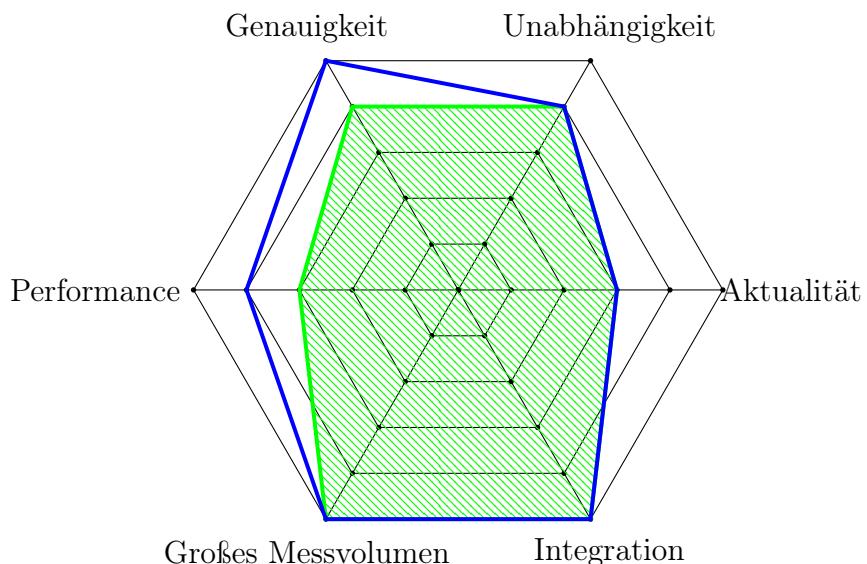
Abbildung 4.13.: Keine Überraschungen hier. Mit zunehmender Komplexität zeigt sich eine Zunahme der Ausführungszeit. Gleiches Ergebnis wie bei den idealen Messdaten.



5. Diskussion

Die in dieser Arbeit bearbeitete Aufgabestellung ist sehr komplex. Die erreichten Ergebnisse erfüllen die Anforderungen nicht ganz. Der Ansatz das Problem über evolutionäre Verfahren zu lösen funktioniert zuverlässig. Auch auf limitierten Ressourcen (wenig Rechenleistung) konnten brauchbare Ergebnisse in einer akzeptablen Zeit gefunden werden. Die Performance kann mit den hier beschriebenen Methoden (Threading) weiter gesteigert werden. Das erhöht die Sicherheit des Messergebnisses. In Abbildung 5.1 wird die Anforderungsspinne (Abbildung 1.1) mit den erreichten Resultaten in Deckung gebracht.

Abbildung 5.1.: Grafische Übersicht der Ergebnisse gegen die gestellten Anforderungen (blau) an diese Arbeit. Der grüne Bereich stellt die erreichten Ziele dar. Wie bereits Diskutiert, werden die Anforderungen an die Genauigkeit nicht gut erfüllt. Auch die Performance ist nicht zufriedenstellend erreicht, die Gründe wurden bereits angegeben. In den anderen Bereichen zeigt sich eine gute Deckung mit den Anforderungen.



Wie in Kapitel 4.2 beschrieben liefern sowohl reale Messdaten als auch künstliche, ideale Eingabewerte korrekte Ergebnisse, wenn man die Evolutionsparameter passend wählt. Brauchbare Konfigurationen wurden präsentiert. Es ist nicht nachgewiesen, dass man daraus eine Allgemeingültigkeit für die Zukunft ableiten kann. Es wurde in dieser Arbeit nur ein Messaufbau vermessen. Die Lösung konnte ohne wesentliche Einschränkungen und Registrierung durchgeführt werden. Der in dem verwendeten Aufbau abgedeckte Messbereich umfasste in etwa $30m^3$. Damit ist erwiesen, dass sich die Methode für ein großes Messvolumen eignet

Die dabei verwendete Kalibrierroutine wurde ausführlich vorgestellt (Kapitel ??). Das Softwaremodul für die Kalibrierung kommt bereits in anderer Software der amedo STS zum Einsatz. Die Integration in die bestehende Softwarestrukturen war einfach.

Die Software wurde mit *C++11* entwickelt und ist somit auf dem Stand der Technik. Sie lässt sich einfach auf eine andere Plattform portieren, die das Buildtool *CMake* unterstützt.

Über die Ziele dieser Arbeit hinaus konnten Vorschläge gemacht werden, die Problemstellung einer automatischen Antennenumschaltung zu lösen.

6. Schluss

6.1. Ausblick

Wie in den Ergebnissen gezeigt, ist die Lösung nicht immer zufriedenstellend gefunden worden. Das wird aus den in den Ergebnissen präsentieren Grafiken deutlich. Wie in der Analyse der Komplexität, Kapitel 3.6, geschildert, ist der Suchraum sehr komplex. Es ist ggf. möglich durch vorhandene Informationen die Problemdimension zu reduzieren. Ein darauf basierendes Modell wurde in dieser Arbeit nicht untersucht. Die Umsetzung eines solchen Modells ist mit dem in dieser Arbeit geschaffenen Rahmen jedoch ohne großen Aufwand möglich. Dazu muss lediglich eine neue Objektfunktion erstellt werden, wie in Kapitel 3.8.2 beschrieben.

Ungünstige Umstände (z.B: schlechte Konditionierung, geringe Anzahl an Antennen etc.) können zu den unzureichenden Ergebnissen beitragen. Zur Zeit wird in einem solchen Fall ein statistischer Ansatz durchgeführt. Dazu werden mehrere Lösungsversuche durchgeführt und anschließend der Median-Wert der Variablen für die Abschätzung genutzt, siehe Kapitel 4.2. Das Verbessert das Ergebnis zu Lasten der Ausführungszeit. Die Zeit bis ein Resultat vorliegt kann sich zum Teil auf mehrere Sekunden vergrößern. Das liegt außerhalb der Anforderungen. Hier muss durch weitere empirische Analysen ein gutes Mittelmaß gefunden werden.

Es ist im Weiteren auch nicht sinnvoll die in dieser Arbeit gefundene Implementation als fehlerfrei zu betrachten. Obwohl auf die Verifikation einzelner Komponenten genau geachtet wurde (siehe z.B. Kapitel 3.7) konnte die Implementation des Modells mit keiner Referenz verglichen werden. Alle Eingaben in das Modell, wie die statischen Matrizen und der Ergebnisvektor, wurden von Hand berechnet und konnten so geprüft werden. Allerdings gibt es schlicht keine andere Umsetzung des Modells. Es ist ratsam, die Implementation einem Review zu unterziehen.

Im Rahmen dieser Arbeit konnte eine automatische, auf Messungen basierende Kalibrierung nicht mehr erprobt werden. Das hier vorgestellte Konzept der Entfernungsfindung eignet sich dazu, ein solches System zu implementieren. Dazu könnte ein Phantom mit einer bekannten geometrischen Proportion mit Tags ausgestattet werden. Die Antennen würden die Phasenwerte

ausgeben und die Positionen der Antennen würde optimiert werden können. Das bedeutet die Berechnung des umgekehrten Falls zur Entfernungsfindung eines Tags.

Weitere Anwendungsmöglichkeiten

Das Modell, das in dieser Arbeit entwickelt wurde, ermöglicht eine Anwendung abseits der Entfernungsfindung. Der hier vorgestellte Ansatz kann zur Lösung eines anderen Problems beitragen, das durch die freie Anordnung der Antennen entsteht. Wenn mehrere Tags im Raum identifiziert werden, muss die Steuerung der Antennenumschaltung¹ zur Zeit alle Antennen in einem Round-Robin-Verfahren² umschalten. Nach einer Umschaltung kann es dazu kommen, dass keine der Antennen einen der zuvor identifizierten Tags *sieht*. Da die Umschaltung zur Zeit zufällig zur nächsten Antenne übergeht, werden auch Antennen genommen, die keine günstige Positions berechnung erlauben. Die mögliche Lösung dafür ist, eine intelligente Steuerung der Umschaltung zu implementieren, indem die Kondition für jede Konfiguration aus vier Antennen verwendet wird. Ordnete man die sich auf dem Antennenaufbau ergebenen Antennenkombinationen ihrer Konditionszahl nach Aufsteigend³ an, können diese Antennenkombinationen von der Antennenumschaltung bevorzugt angesteuert werden. Dadurch werden stets die am besten konditionierte Kombination gewählt und ein möglichst sicheres Berechnungsergebnis gefunden.

6.2. Zusammenfassung

Evolutionäre Strategien und besonders die CMA-ES sind dazu geeignet die komplexe Aufgabe der Entfernungsabschätzung zu lösen. Die eingesetzten Verfahren entsprechen dem Stand der Technik in der evolutionären Optimierung. Mit Hilfe dieser Verfahren ist wurde eine funktionierende Entfernungsabschätzung für den Messaufbau des PRPS entwickelt. Es ist gelungen Parameter zu identifizieren mit denen sich die evolutionäre Optimierung steuern lässt. Der Einfluss der Steuerparameter auf die Performance wurde untersucht und quantifiziert.

Es wurde ein System entwickelt das die Konditionszahl der Modell-Matrizen berechnet. Anwendungsmöglichkeiten für die Kondition wurden besprochen.

¹Diese bestimmt von welcher Antenne gerade versucht wird die Tags zu lesen - es können nicht gleichzeitig alle Antennen gelesen werden

²Umschalten nach einer festen Zeit und in einem vorgegebenen Muster

³zur Erinnerung, gute Kondition = kleine Konditionszahl

Auf der Basis dieses Modells kann eine selbst-optimierende Kalibrierung erstellt werden. Zur Untersuchung der Fitnesslandschaft der Objektfunktion ist ein eigenes Stück Software entwickelt worden. Es erlaubt die Untersuchung von Fitnessebenen in der Fitnesslandschaft der Modelfunktionen. Die Ergebnisse des Werkzeugs wurden vorgestellt-

Das entwickelte Modell basiert auf der Trilateration und ermöglicht eine Anwendung in der Kalibrierung von Messaufbauten. Eine entsprechende Lösung befindet sich bereits im Einsatz. Die in dieser Arbeit eingesetzte Software-Bibliothek '*Shark*' ist eine leistungsfähig und flexibel. Sie wurde erfolgreich in das Software-Ökosystem der amedo STS integriert und steht für weitere Projekte zur Verfügung. Weiterhin wurde das Softwareerstellungswerkzeug *CMake* in dieser Arbeit erfolgreich erprobt.

Eine Weiterentwicklung, basierend auf den Ergebnissen und Erkenntnissen dieser Masterarbeit, ist geplant. Ansatzpunkte bieten dabei die Reduzierung der Problemdimension, die Verbesserung der Performance und einer Analyse der beschriebenen, weiteren Anwendungsmöglichkeiten. Die Verbesserung der Performance ist direkt gekoppelt an die Reduzierung der Dimension des Problems. Auch die Entwicklung der selbst-optimierende Kalibrierung wird weiterverfolgt werden. Wie beschrieben ist eine Validierung der Objektfunktion sinnvoll und wird in kommenden Schritten durchgeführt werden.

A. Abbildungen

A.1. Messaufbauten

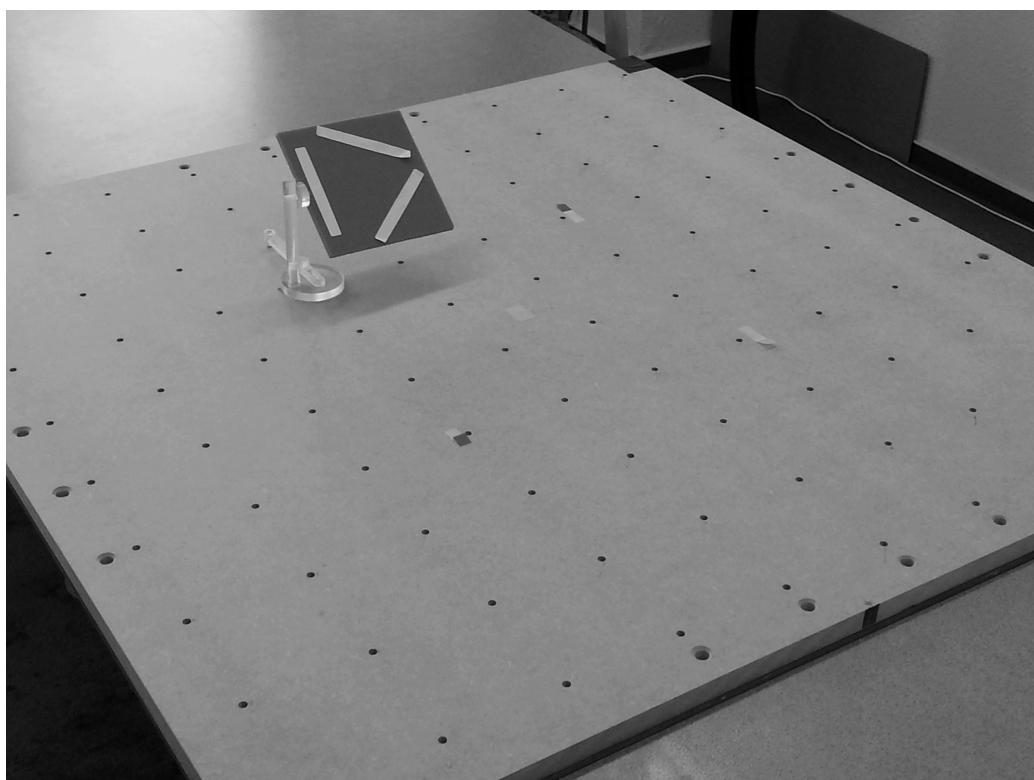


Abbildung A.1.: Reproduzierbare Aufstellung für die Messungen. Im Bild zu erkennen ist ein Taghalter mit drei Tags und eine Grundplatte, die eine Fläche von 1×1 Meter abdeckt. Dabei beträgt die Distanz zwischen den einzelnen Positionen 10 cm, sodass sich insgesamt 100 Messpunkte anfahren lassen. Die Positionen sind aufsteigend nach Richtung der Ebenen benannt. Der *Ursprung* befindet sich in der linken unteren Ecke (vom Bild nicht erfasst). Der Aufsteller mit den Tags befindet sich demnach an Position (3, 6).

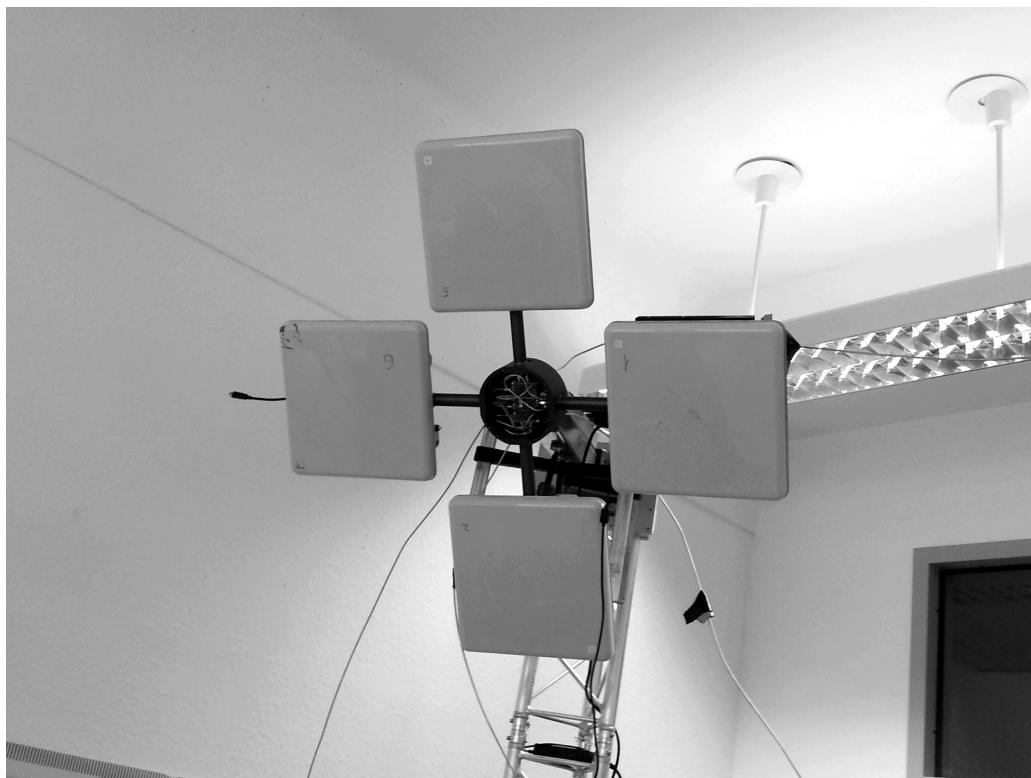


Abbildung A.2.: Messaufbau mit *Blick* auf die reproduzierbare Aufstellung. Antenne 1 ist die oben gelegene Antenne. Im Uhrzeigersinn ordnen sich die restlichen Antennen an. Zur Orientierung: Die reproduzierbare Aufstellung befindet sich in Richtung des Betrachters, von dem Aufbau aus gesehen.

B. Quellcodeauszüge

B.1. ObjectiveFunktion - Evolutionary Calibration Header

Vollständiger Quellcode der EvolutionaryCalibration.h

```
1 #ifndef _EVOLUTIONARY_CALIBRATION_H_
2 #define _EVOLUTIONARY_CALIBRATION_H_
3
4 #include <shark/ObjectiveFunctions/AbstractObjectiveFunction.h>
5 #include <shark/Algorithms/DirectSearch/CMA.h>
6 #include <shark/Algorithms/AbstractOptimizer.h>
7 #include <shark/Rng/GlobalRng.h>
8 #include <nr3/nr3.h>
9 #include "../solve.h"
10
11 namespace PRPSEvolution {
12     namespace Models {
13         using namespace shark;
14
15         /**
16          *
17          */
18     struct EvolutionaryCalibration : public SingleObjectiveFunction {
19
20         typedef AbstractOptimizer<shark::VectorSpace< double >,double,
21             SingleObjectiveResultSet<typename shark::VectorSpace< double >:::
22             PointType> > base_type;
23
24         typedef typename base_type::ObjectiveFunctionType
25             ObjectiveFunctionType;
26
27         EvolutionaryCalibration( ) {
28             m_numberOfVariables = Solve::ProblemDimensions::Calibration ;
29             m_features |= CAN_PROPOSE_STARTING_POINT;
30         }
31
32         /// \brief From INameable: return the class name.
33         std::string name() const
34         { return "Evolutionary Calibration"; }
35
36         std::size_t numberOfVariables() const{
37             return m_numberOfVariables;
38         }
39
40         bool hasScalableDimensionality() const{
41             return true;
42         }
43
44         void setNumberOfVariables( std::size_t numberOfVariables ){
45             m_numberOfVariables = numberOfVariables;
46         }
47 }
```

```

45 void configure(const PropertyTree &node) {
46     m_numberOfVariables = node.get("numberOfVariables", 51);
47 }
48 /**
49 * Generate a starting value
50 * @param[out] x The suggested search point
51 *
52 */
53 void proposeStartingPoint(SearchPointType &x) const {
54     x.resize(numberOfVariables());
55
56     for (unsigned int i = 0; i < 3; i++) {
57         x(i) = Rng::uni(-10, 10);
58     }
59 }
60 /**
61 *
62 */
63 /**
64 */
65 double eval(const SearchPointType &p) const;
66
67 /**
68 *
69 */
70 void setParams( const NRmatrix< Doub > &M,
71                 const NRvector< Doub > &v
72                 ) {
73     setMat(M);
74     setVec(v);
75 }
76
77 /**
78 *
79 */
80 /**
81 */
82 void setMat( const NRmatrix< Doub > &M ) {
83     A = M;
84     A_isSet = true;
85 }
86
87 /**
88 *
89 */
90 void setVec( const NRvector< Doub > &v ) {
91     b = v;
92     b_isSet = true;
93 }
94
95 inline double mkII( const NRmatrix<Doub> &A, const double* x, const
96                     NRvector<Doub> &b ) const;
97
98 private:
99     std::size_t m_numberOfVariables;
100
101    /** The Matrices we need to solve the Problem */
102    NRmatrix< Doub > A;
103    bool A_isSet = false;
104
105    /** The b-vector needed to find a Solution */
106    NRvector< Doub > b;
107    bool b_isSet = false;
108 };

```

```
109 |     ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION( EvolutionaryCalibration ,  soos::
110 |         RealValuedObjectiveFunctionFactory );
111 |
112 |     }
113 | } /* end namespace */
114 |
115 #endif /* _EVOLUTIONARY_CALIBRATION_H_ */
```

B.2. ObjectiveFunktion - Evolutionary Calibration Source

Vollständiger Quellcode der EvolutionaryCalibration.cpp

```

1 #include "EvolutionaryCalibration.h"
2
3 namespace PRPSEvolution {
4     namespace Models {
5         using namespace shark;
6
7         double EvolutionaryCalibration::eval( const SearchPointType &p )
8             const
9         {
10             m_evaluationCounter++;
11             std::vector<double> res;
12
13             double x[ m_numberOfVariables ];
14
15             x[ 0 ] = p[ 0 ];
16             x[ 1 ] = p[ 1 ];
17             x[ 2 ] = p[ 2 ];
18             return mkII( A, x, b );
19
20         }
21
22         inline double EvolutionaryCalibration::mkII( const NRmatrix<Double>
23             &A, const double* x, const NRvector<Double> &b ) const
24         {
25             double res;
26             double prod_Ax[3] = {0.,0.,0.};
27             double x_[m_numberOfVariables];
28
29             x_[0]=x[0];
30             x_[1]=x[1];
31             x_[2]=x[2];
32
33             /* multiply the matrix with the vector */
34             for( int i = 0; i < A.nrows(); i++ )
35                 for( int j = 0; j < A.ncols(); j++ )
36                     prod_Ax[i] += A[i][j]*x_[j];
37
38             /* sum up */
39             res = (prod_Ax[0] - b[0]) * (prod_Ax[0] - b[0]);
40             res += (prod_Ax[1] - b[1]) * (prod_Ax[1] - b[1]);
41             res += (prod_Ax[2] - b[2]) * (prod_Ax[2] - b[2]);
42
43             return res;
44
45         }
46     } /* !namespace */
47 } /* !namespace */

```

C. Gnuplot Skripte

C.1. Boxplot

Listing C.1: Gnuplot Boxplot-Skript

```
1 set style line 1 linetype 1 linecolor rgbcolor "#2f4f4f" linewidth  
2 1.5 pointtype 7 pointsize .5 pointinterval 1  
3 set style line 2 linetype 1 linecolor rgbcolor "#696969" linewidth  
4 1.5 pointtype 7 pointsize .5 pointinterval 1  
5 set style line 3 linetype 1 linecolor rgbcolor "#708090" linewidth  
6 1.5 pointtype 7 pointsize .5 pointinterval 1  
7 set style line 4 linetype 1 linecolor rgbcolor "#bebebe" linewidth  
8 1.5 pointtype 7 pointsize .5 pointinterval 1  
9  
10 if( i == 0 ) set terminal pngcairo truecolor transparent background "#  
11 ffffff" enhanced font "arial ,10" size w, h  
12  
13 set style fill transparent solid 0.3 noborder  
14 #set style boxplot outliers pointtype 19  
15 #set style data boxplot  
16 #set key right bottom vertical Left noreverse enhanced box samplen .2  
17 #set key opaque  
18 set grid  
19  
20 #in this column we find the Data for sigma  
21 lastDataCol = 3+a+2  
22 #int this column we expect the vector  
23 vectorCol = 3+a+3  
24 inputfile = "data/single_ ".i.".dat"  
25 outMultiplot = "img/boxen/kondensiert/".i.".png"  
26  
27 set output outMultiplot  
28 set multiplot layout 1,3  
29 unset logscale  
30 set autoscale  
31 unset label  
32  
33 #  
34 #setup the 1. plot  
35 set style data boxplot  
36 set xlabel "Optimzation objective"  
37 set ylabel "Final Value – Coordinates ( x, y, z )"  
38  
39 set size 1, .6  
40 set origin .0 ,.4  
41 set autoscale  
42 set xrange [.5:a+.5]  
43 set xtics  
44 set ytics format "%1.1f"  
45 set yrange [-4:4]  
46 if(a>3) set y2tics format "%1.1f"  
47 if(a>3) set y2label "Final Value – Wavenumbers ( n )"
```

```

45 print "a=",a
46 if(a==7) set xtics ("x" 1, "y" 2, "z" 3 ) scale 0.0
47 if(a==3) plot inputfile u (1):5 ls 1 axes x1y1 notitle , \
48     '' u (2):6 ls 2 axes x1y1 notitle , \
49     '' u (3):7 ls 3 axes x1y1 notitle
50 if(a==3) unset y2tics
51 if(a==3) unset y2label
52
53 if(a==6) plot inputfile u (1):5 ls 1 notitle , \
54     '' u (2):6 ls 2 notitle , \
55     '' u (3):7 ls 3 notitle , \
56     '' u (4):8 ls 4 notitle , \
57     '' u (5):9 ls 4 notitle , \
58     '' u (6):10 ls 4 notitle
59
60 if(a==7) set xtics ("x" 1, "y" 2, "z" 3, "n0" 4,"n1" 5,"n2" 6,"n3" 7)
61     scale 0.0
62 if(a==7) plot inputfile u ($2 < limit ? (1): 1/0):5 ls 1 axes x1y1
63     notitle , \
64     '' u ($2 < limit ? (2): 1/0):6 ls 2 axes x1y1 notitle , \
65     '' u ($2 < limit ? (3): 1/0):7 ls 3 axes x1y1 notitle , \
66     '' u ($2 < limit ? (4): 1/0):8 ls 4 axes x1y2 notitle , \
67     '' u ($2 < limit ? (5): 1/0):9 ls 4 axes x1y2 notitle , \
68     '' u ($2 < limit ? (6): 1/0):10 ls 4 axes x1y2 notitle , \
69     '' u ($2 < limit ? (7): 1/0):11 ls 4 axes x1y2 notitle
70
71 if(a==8) set xtics ("x" 1, "y" 2, "z" 3, "n0" 4,"n1" 5,"n2" 6,"n3" 7,"
72     n4" 8) scale 0.0
73 if(a==8) plot inputfile u ($2 < limit ? (1): 1/0):5 ls 1 axes x1y1
74     notitle , \
75     '' u ($2 < limit ? (2): 1/0):6 ls 2 axes x1y1 notitle , \
76     '' u ($2 < limit ? (3): 1/0):7 ls 3 axes x1y1 notitle , \
77     '' u ($2 < limit ? (4): 1/0):8 ls 4 axes x1y2 notitle , \
78     '' u ($2 < limit ? (5): 1/0):9 ls 4 axes x1y2 notitle , \
79     '' u ($2 < limit ? (6): 1/0):10 ls 4 axes x1y2 notitle , \
80     '' u ($2 < limit ? (7): 1/0):11 ls 4 axes x1y2 notitle , \
81     '' u ($2 < limit ? (8): 1/0):12 ls 4 axes x1y2 notitle
82 if(a==9) set xtics ("x" 1, "y" 2, "z" 3, "n0" 4,"n1" 5,"n2" 6,"n3" 7,
83     n4" 8, "N5" 9) scale 0.0
84 if(a==9) plot inputfile u ($2 < limit ? (1): 1/0):5 ls 1 axes x1y1
85     notitle , \
86     '' u ($2 < limit ? (2): 1/0):6 ls 2 axes x1y1 notitle , \
87     '' u ($2 < limit ? (3): 1/0):7 ls 3 axes x1y1 notitle , \
88     '' u ($2 < limit ? (4): 1/0):8 ls 4 axes x1y2 notitle , \
89     '' u ($2 < limit ? (5): 1/0):9 ls 4 axes x1y2 notitle , \
90     '' u ($2 < limit ? (6): 1/0):10 ls 4 axes x1y2 notitle , \
91     '' u ($2 < limit ? (7): 1/0):11 ls 4 axes x1y2 notitle , \
92     '' u ($2 < limit ? (8): 1/0):12 ls 4 axes x1y2 notitle , \
93     '' u ($2 < limit ? (9): 1/0):12 ls 4 axes x1y2 notitle
94 if(a==10) set xtics ("x" 1, "y" 2, "z" 3, "n0" 4,"n1" 5,"n2" 6,"n3" 7,
95     "n4" 8, "N5" 9, "N6" 10) scale 0.0
96 if(a==10) plot inputfile u ($2 < limit ? (1): 1/0):5 ls 1 axes x1y1
97     notitle , \
98     '' u ($2 < limit ? (2): 1/0):6 ls 2 axes x1y1 notitle , \
99     '' u ($2 < limit ? (3): 1/0):7 ls 3 axes x1y1 notitle , \
100    '' u ($2 < limit ? (4): 1/0):8 ls 4 axes x1y2 notitle , \
101    '' u ($2 < limit ? (5): 1/0):9 ls 4 axes x1y2 notitle , \

```

```

102      '' u ($2 < limit ? (7): 1/0):11 ls 4 axes x1y2 notitle , \
103      '' u ($2 < limit ? (8): 1/0):12 ls 4 axes x1y2 notitle , \
104      '' u ($2 < limit ? (9): 1/0):12 ls 4 axes x1y2 notitle , \
105      '' u ($2 < limit ? (10): 1/0):12 ls 4 axes x1y2 notitle
106
107 if(a==11) set xtics ("x" 1, "y" 2, "z" 3, "n0" 4,"n1" 5,"n2" 6,"n3" 7,
108           "n4" 8, "N5" 9, "N6" 10, "N7" 11) scale 0.0
109
110 if(a==11) plot inputfile u ($2 < limit ? (1): 1/0):5 ls 1 axes x1y1
111   notitle , \
112   '' u ($2 < limit ? (2): 1/0):6 ls 2 axes x1y1 notitle , \
113   '' u ($2 < limit ? (3): 1/0):7 ls 3 axes x1y1 notitle , \
114   '' u ($2 < limit ? (4): 1/0):8 ls 4 axes x1y2 notitle , \
115   '' u ($2 < limit ? (5): 1/0):9 ls 4 axes x1y2 notitle , \
116   '' u ($2 < limit ? (6): 1/0):10 ls 4 axes x1y2 notitle , \
117   '' u ($2 < limit ? (7): 1/0):11 ls 4 axes x1y2 notitle , \
118   '' u ($2 < limit ? (8): 1/0):12 ls 4 axes x1y2 notitle , \
119   '' u ($2 < limit ? (9): 1/0):12 ls 4 axes x1y2 notitle , \
120   '' u ($2 < limit ? (10): 1/0):12 ls 4 axes x1y2 notitle , \
121   '' u ($2 < limit ? (11): 1/0):12 ls 4 axes x1y2 notitle
122
123 unset label
124 unset y2label
125 unset y2range
126 unset y2tics
127
128 #_____________________________________
129 set boxwidth 0.05 relative
130
131 set autoscale
132 set xlabel ""
133 #set logscale y
134 set ylabel "Evaluations"
135 set size .25, .4
136 set origin .0,.0
137 unset xtics
138 #set xrange [-.2:.4]
139 set ytics format "%.1e"
140
141 plot inputfile u ($2 < limit ? (1): 1/0):2 ls 4 notitle
142
143 #_____________________________________
144 #set logscale y
145 set ytics
146
147 set xlabel ""
148 set ylabel "Function Value"
149 set size .25, .4
150 set origin .25,.0
151 unset xtics
152 set ytics format "%.1e"
153
154 plot inputfile u ($2 < limit ? (1): 1/0):3 ls 4 notitle
155
156 #_____________________________________
157 #setup the 4. plot
158 set xlabel ""
159 set ylabel "Sigma"
160 set ytics format "%.1e"
161
162 set size .25, .4
163 set origin .50,.0
164

```

```
165 unset logscale
166 set autoscale
167 unset xtics
168 set ytics
169
170 plot inputFile u ($2 < limit ? (1): 1/0):lastDataCol ls 4 notitle
171
172 #
173 #setup the 5. plot
174 set xlabel ""
175 set ylabel "Distance"
176 set ytics format "%.2e"
177
178 set size .25, .4
179 set origin .75,.0
180
181 set autoscale
182 unset xtics
183 set ytics
184
185 plot inputFile u ($2 < limit ? (1): 1/0):vectorCol ls 4 notitle
186
187 #
188 i=i+1
189
190 unset multiplot
191 unset xtics
192
193 if ( i < m) reread
194 i=0
```

C.2. Lineplot

Listing C.2: Gnuplot Lineplot-Skript

```

1  #prerequisites set i, n and the number of antennas to proper values
2  at(file , row, col) = system( sprintf("awk -v row=%d -v col=%d 'NR ==
   row {print $col}' %s", row, col, file) )
3
4  set style line 1 linetype 1 linecolor rgbcolor "#882f4f4f" linewidth
   .5
5  set style line 2 linetype 1 linecolor rgbcolor "#88696969" linewidth
   .5
6  set style line 3 linetype 1 linecolor rgbcolor "#88708090" linewidth
   .5
7  set style line 4 linetype 1 linecolor rgbcolor "#ccbebebe" linewidth
   .5
8
9  set style line 5 linetype 1 linecolor rgbcolor "#99696969" linewidth
   .3
10
11 if( i == 0 ) set terminal pngcairo truecolor transparent background "#ffff
   ffffff" enhanced font "arial,10" size w, h
12
13 set key right bottom vertical Left noreverse enhanced box samplen .2
14 set key opaque
15 set grid
16
17 lastDataCol = 3+a+2
18 inputfile = "data/.i./.dat"
19 outMultiplot = "img/linien/kondensiert/.i./.png"
20
21 file=inputfile ; row=2 ; col=2
22
23 set output outMultiplot
24 set multiplot layout 1,3
25 unset logscale
26 set autoscale
27
28 #
29 stats inputfile u 1 name "Stat" nooutput
30
31 #print "test ".at(file ,Stat_records ,1)
32
33 locallimit=0.5*limit
34 if(a<=3)print "local limit is: ",locallimit
35
36 #setup the first plot
37 set xrange [0:locallimit]
38
39 set ytics format "%0.0f"
40 set yrange [-10:10]
41
42 if(a>3) set autoscale
43
44 set clip one
45 set xlabel "Funtion Evaluations"
46 set ylabel "Objective Values"
47 set xtics
48 set ytics
49 set size 1., .6
50 set origin .0 ,.4
51
52
53 #print "local locallimit ",locallimit
54

```

```

55 if( a==3 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):4 w lines
      title "x" ls 1, \
      "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
      "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3
56
57 if( a==7 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):4 w lines
      title "x" ls 1, \
      "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
      "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3, \
58
59      "" u ($1 < locallimit ? $1 : 1/0):7 w lines title "n0" ls 4, \
60      "" u ($1 < locallimit ? $1 : 1/0):8 w lines title "n1" ls 4, \
61      "" u ($1 < locallimit ? $1 : 1/0):9 w lines title "n2" ls 4, \
62      "" u ($1 < locallimit ? $1 : 1/0):10 w lines title "n3" ls 4, \
63      "" u ($1 < locallimit ? $1 : 1/0):11 w lines title "n4" ls 4, \
64      "" u ($1 < locallimit ? $1 : 1/0):12 w lines title "n5" ls 4, \
65      "" u ($1 < locallimit ? $1 : 1/0):13 w lines title "n6" ls 4, \
66
67 if( a==8 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):7 w lines
      title "n0" ls 4, \
      "" u ($1 < locallimit ? $1 : 1/0):8 w lines title "n1" ls 4, \
68      "" u ($1 < locallimit ? $1 : 1/0):9 w lines title "n2" ls 4, \
69      "" u ($1 < locallimit ? $1 : 1/0):10 w lines title "n3" ls 4, \
70      "" u ($1 < locallimit ? $1 : 1/0):11 w lines title "n4" ls 4, \
71      "" u ($1 < locallimit ? $1 : 1/0):12 w lines title "n5" ls 4, \
72      "" u ($1 < locallimit ? $1 : 1/0):13 w lines title "n6" ls 4, \
73      "" u ($1 < locallimit ? $1 : 1/0):14 w lines title "x" ls 1, \
74      "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
75      "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3
76
77 if( a==9 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):7 w lines
      title "n0" ls 4, \
      "" u ($1 < locallimit ? $1 : 1/0):8 w lines title "n1" ls 4, \
78      "" u ($1 < locallimit ? $1 : 1/0):9 w lines title "n2" ls 4, \
79      "" u ($1 < locallimit ? $1 : 1/0):10 w lines title "n3" ls 4, \
80      "" u ($1 < locallimit ? $1 : 1/0):11 w lines title "n4" ls 4, \
81      "" u ($1 < locallimit ? $1 : 1/0):12 w lines title "n5" ls 4, \
82      "" u ($1 < locallimit ? $1 : 1/0):13 w lines title "n6" ls 4, \
83      "" u ($1 < locallimit ? $1 : 1/0):14 w lines title "x" ls 1, \
84      "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
85      "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3
86
87 if( a==10 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):7 w lines
      title "n0" ls 4, \
      "" u ($1 < locallimit ? $1 : 1/0):8 w lines title "n1" ls 4, \
88      "" u ($1 < locallimit ? $1 : 1/0):9 w lines title "n2" ls 4, \
89      "" u ($1 < locallimit ? $1 : 1/0):10 w lines title "n3" ls 4, \
90      "" u ($1 < locallimit ? $1 : 1/0):11 w lines title "n4" ls 4, \
91      "" u ($1 < locallimit ? $1 : 1/0):12 w lines title "n5" ls 4, \
92      "" u ($1 < locallimit ? $1 : 1/0):13 w lines title "n6" ls 4, \
93      "" u ($1 < locallimit ? $1 : 1/0):14 w lines title "x" ls 1, \
94      "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
95      "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3
96
97 if( a==11 ) plot inputfile u ($1 < locallimit ? $1 : 1/0):7 w lines
      title "n0" ls 4, \
98      "" u ($1 < locallimit ? $1 : 1/0):8 w lines title "n1" ls 4, \
99      "" u ($1 < locallimit ? $1 : 1/0):9 w lines title "n2" ls 4, \
100     "" u ($1 < locallimit ? $1 : 1/0):10 w lines title "n3" ls 4, \
101     "" u ($1 < locallimit ? $1 : 1/0):11 w lines title "n4" ls 4, \
102     "" u ($1 < locallimit ? $1 : 1/0):12 w lines title "n5" ls 4, \
103     "" u ($1 < locallimit ? $1 : 1/0):13 w lines title "n6" ls 4, \
104     "" u ($1 < locallimit ? $1 : 1/0):14 w lines title "n7" ls 4, \
105     "" u ($1 < locallimit ? $1 : 1/0):15 w lines title "x" ls 1, \
106     "" u ($1 < locallimit ? $1 : 1/0):5 w lines title "y" ls 2, \
107     "" u ($1 < locallimit ? $1 : 1/0):6 w lines title "z" ls 3
108
109 set autoscale
110
111 set ytics format "%1e"
112 set logscale y
113 set xlabel ""

```

```
114 | set ylabel "Function Value"
115 | set size .5, .4
116 | set yrangle [1e-25:20000]
117 | set origin .0 ,.0
118 | plot inputfile u ($1 < locallimit ? $1 : 1/0):2 w lines ls 5 title "
119 |     fitness"
120 | set size .5, .4
121 | set origin .5 ,.0
122 | set autoscale
123 | #set yrangle [1e-10:2]
124 | #set xrange [0:3500]
125 | #set clip one
126 | set ylabel "Sigma"
127 |
128 | plot inputfile u ($1 < locallimit ? $1 : 1/0):lastDataCol w lines ls
129 |     5 title "{/Symbol s}"
130 | i=i+1
131 |
132 | unset multiplot
133 |
134 | if ( i < m) reread
135 | i=0
```

C.3. Scatterplot

Listing C.3: Gnuplot Scatterplot-Skript

```

1 # This script is to generate a plot from the final values of the
2 # solutions for one antenna
3 #
4 #prerequisites set i , n and the number of antennas to proper values
5 #
6 set macros
7 SETRANGE="set xrange [-5:5];\
8     set yrange [-5:5]"
9 UNSETRANGE=" set autoscale"
10
11
12 at(file , row , col) = system( sprintf("awk -v row=%d -v col=%d 'NR == \
13     row {print $col}' %s", row , col , file ) )
14 to(file , min , first , mean , third , max , row) = system( sprintf("echo %d \
15     %e %e %e %e >> %s", row , min , first , mean , third , max ,file ) )
16 header(file) = system( sprintf("echo \"#Idx min first mean third max\" \
17     >> %s", file ) )
18 toScientific(file , min , first , mean , third , max , row) = system( \
19     sprintf("echo %d %e %e %e %e >> %s", row , min , first , mean ,
20     third , max ,file ) )
21 remove(file) = system( sprintf( "rm %s", file ) )
22 to2(file , value) = system( sprintf("echo 1 2 3 %s %s", value , file ) )
23 echoStats( min , first , mean , third , max) = system( sprintf("echo %e %e \
24     %e %e %e ", min , first , mean , third , max) )
25
26 unset style
27 set style line 1 linetype 1 linecolor rgb "#708090" linewidth 1
28     pointtype 7 pointsize .5
29 set style line 2 linetype -1 linecolor rgb "#2f4f4f" linewidth 1.2
30
31 set style line 3 linetype 1 linecolor rgb "#ee708090" linewidth 1.000
32     pointtype 7 pointsize .5 pointinterval 1
33 #set style line 3 linetype 1 linecolor rgb "red" linewidth 1.000
34     pointtype 7 pointsize 1 pointinterval 5
35 #set style line 4 linetype 1 linecolor rgb "gray" linewidth 1
36     pointtype 2 pointsize default pointinterval 0
37
38 set style arrow 1 heads size screen 0.008,90 ls 2
39
40 if( i == 0 ) set terminal pngcairo truecolor transparent background "# \
41     ffffff" enhanced font "arial,10" size w, h
42
43 set style fill transparent solid 0.3 noborder
44 set key right bottom vertical Left noreverse enhanced box samplen .2
45 set key opaque
46 set grid
47
48 lastDataCol = 3+a+2
49 inputfile = "data/single_".i.".dat"
50 input_all = "data/single_".i.".dat"
51 input_one = "data/".i.".dat"
52
53 outMultiplot = "img/linien/kondensiert/scatter".i.".png"
54
55 #print "Processing: Start"
56
57 set output outMultiplot
58
59 set multiplot layout a,a
60
```

```

50 #collect information about the file
51 unset logscale
52 set autoscale
53 unset label
54 unset xlabel
55 unset ylabel
56
57 #
58 #setup the 1. plot
59 unset ytics
60 unset xtics
61
62 LABELX = sprintf("x" )
63 LABELY = sprintf("y" )
64 LABELZ = sprintf("z" )
65 LABELN0 = sprintf("n0" )
66 LABELN1 = sprintf("n1" )
67 LABELN2 = sprintf("n2" )
68 LABELN3 = sprintf("n3" )
69 LABELN4 = sprintf("n4" )
70 LABELN5 = sprintf("n5" )
71 LABELN6 = sprintf("n6" )
72 LABELN7 = sprintf("n7" )
73
74 unset key
75
76 labelxpos = .1
77 labelypos = .5
78
79 @SETRANGE
80 # generate first row
81 set label at graph labelxpos,labelypos center LABELX front left font "
     Arial,24" textcolor rgb "#4f2f2f"
82 plot inputfile u ($2 < limit ? $5: 1/0):5 ls 3 notitle
83 unset label
84
85 plot inputfile u ($2 < limit ? $5: 1/0):6 ls 1 notitle
86 plot inputfile u ($2 < limit ? $5: 1/0):7 ls 1 notitle
87 @UNSETRANGE
88 if(a>=4)plot inputfile u ($2 < limit ? $5: 1/0):8 ls 1 notitle
89 if(a>=5)plot inputfile u ($2 < limit ? $5: 1/0):9 ls 1 notitle
90 if(a>=6)plot inputfile u ($2 < limit ? $5: 1/0):10 ls 1 notitle
91 if(a>=7)plot inputfile u ($2 < limit ? $5: 1/0):11 ls 1 notitle
92 if(a>=8)plot inputfile u ($2 < limit ? $5: 1/0):12 ls 1 notitle
93 if(a>=9)plot inputfile u ($2 < limit ? $5: 1/0):13 ls 1 notitle
94 if(a>=10)plot inputfile u ($2 < limit ? $5: 1/0):14 ls 1 notitle
95 if(a>=11)plot inputfile u ($2 < limit ? $5: 1/0):15 ls 1 notitle
96
97 # 2.
98 @SETRANGE
99 plot inputfile u ($2 < limit ? $6: 1/0):5 ls 1 notitle
100
101 set label at graph labelxpos,labelypos center LABELY front left font "
     Arial,24" textcolor rgb "#4f2f2f"
102 plot inputfile u ($2 < limit ? $6: 1/0):6 ls 3 notitle
103 unset label
104
105 plot inputfile u ($2 < limit ? $6: 1/0):7 ls 1 notitle
106 @UNSETRANGE
107 if(a>=4)plot inputfile u ($2 < limit ? $6: 1/0):8 ls 1 notitle
108 if(a>=5)plot inputfile u ($2 < limit ? $6: 1/0):9 ls 1 notitle
109 if(a>=6)plot inputfile u ($2 < limit ? $6: 1/0):10 ls 1 notitle
110 if(a>=7)plot inputfile u ($2 < limit ? $6: 1/0):11 ls 1 notitle
111 if(a>=8)plot inputfile u ($2 < limit ? $6: 1/0):12 ls 1 notitle
112 if(a>=9)plot inputfile u ($2 < limit ? $6: 1/0):13 ls 1 notitle

```

```

113 if(a>=10)plot inputfile u ($2 < limit ? $6: 1/0):14 ls 1 notitle
114 if(a>=11)plot inputfile u ($2 < limit ? $6: 1/0):15 ls 1 notitle
115
116 # 3.
117 @SEITRANGE
118 plot inputfile u ($2 < limit ? $7: 1/0):5 ls 1 notitle
119 plot inputfile u ($2 < limit ? $7: 1/0):6 ls 1 notitle
120
121 set label at graph labelxpos ,labelypos center LABELZ front left font "
122   Arial ,24" textColor rgb "#4f2f2f"
123 plot inputfile u ($2 < limit ? $7: 1/0):7 ls 3 notitle
124 unset label
125 @UNSETRANGE
126
126 if(a>=4)plot inputfile u ($2 < limit ? $7: 1/0):8 ls 1 notitle
127 if(a>=5)plot inputfile u ($2 < limit ? $7: 1/0):9 ls 1 notitle
128 if(a>=6)plot inputfile u ($2 < limit ? $7: 1/0):10 ls 1 notitle
129 if(a>=7)plot inputfile u ($2 < limit ? $7: 1/0):11 ls 1 notitle
130 if(a>=8)plot inputfile u ($2 < limit ? $7: 1/0):12 ls 1 notitle
131 if(a>=9)plot inputfile u ($2 < limit ? $7: 1/0):13 ls 1 notitle
132 if(a>=10)plot inputfile u ($2 < limit ? $7: 1/0):14 ls 1 notitle
133 if(a>=11)plot inputfile u ($2 < limit ? $7: 1/0):15 ls 1 notitle
134
135 # 4.
136 if(a>3)plot inputfile u ($2 < limit ? $8: 1/0):5 ls 1 notitle
137 if(a>3)plot inputfile u ($2 < limit ? $8: 1/0):6 ls 1 notitle
138 if(a>3)plot inputfile u ($2 < limit ? $8: 1/0):7 ls 1 notitle
139
140 set label at graph labelxpos ,labelypos center LABELNO front left font
141   "Arial ,24" textColor rgb "#4f2f2f"
142 if(a>=4)plot inputfile u ($2 < limit ? $8: 1/0):8 ls 3 notitle
143 unset label
144
144 if(a>=5)plot inputfile u ($2 < limit ? $8: 1/0):9 ls 1 notitle
145 if(a>=6)plot inputfile u ($2 < limit ? $8: 1/0):10 ls 1 notitle
146 if(a>=7)plot inputfile u ($2 < limit ? $8: 1/0):11 ls 1 notitle
147 if(a>=8)plot inputfile u ($2 < limit ? $8: 1/0):12 ls 1 notitle
148 if(a>=9)plot inputfile u ($2 < limit ? $8: 1/0):13 ls 1 notitle
149 if(a>=10)plot inputfile u ($2 < limit ? $8: 1/0):14 ls 1 notitle
150 if(a>=11)plot inputfile u ($2 < limit ? $8: 1/0):15 ls 1 notitle
151
152 # 5.
153 if(a>3)plot inputfile u ($2 < limit ? $9: 1/0):5 ls 1 notitle
154 if(a>3)plot inputfile u ($2 < limit ? $9: 1/0):6 ls 1 notitle
155 if(a>3)plot inputfile u ($2 < limit ? $9: 1/0):7 ls 1 notitle
156 if(a>=4)plot inputfile u ($2 < limit ? $9: 1/0):8 ls 1 notitle
157
158 set label at graph labelxpos ,labelypos center LABELN1 front left font
159   "Arial ,24" textColor rgb "#4f2f2f"
160 if(a>=5)plot inputfile u ($2 < limit ? $9: 1/0):9 ls 3 notitle
161 unset label
162
162 if(a>=6)plot inputfile u ($2 < limit ? $9: 1/0):10 ls 1 notitle
163 if(a>=7)plot inputfile u ($2 < limit ? $9: 1/0):11 ls 1 notitle
164 if(a>=8)plot inputfile u ($2 < limit ? $9: 1/0):12 ls 1 notitle
165 if(a>=9)plot inputfile u ($2 < limit ? $9: 1/0):13 ls 1 notitle
166 if(a>=10)plot inputfile u ($2 < limit ? $9: 1/0):14 ls 1 notitle
167 if(a>=11)plot inputfile u ($2 < limit ? $9: 1/0):15 ls 1 notitle
168
169 # 6.
170 if(a>3)plot inputfile u ($2 < limit ? $10: 1/0):5 ls 1 notitle
171 if(a>3)plot inputfile u ($2 < limit ? $10: 1/0):6 ls 1 notitle
172 if(a>3)plot inputfile u ($2 < limit ? $10: 1/0):7 ls 1 notitle
173 if(a>=4)plot inputfile u ($2 < limit ? $10: 1/0):8 ls 1 notitle
174 if(a>=5)plot inputfile u ($2 < limit ? $10: 1/0):9 ls 1 notitle

```

```

175
176 set label at graph labelxpos ,labelypos center LABELN2 front left font
177   "Arial,24" textcolor rgb "#4f2f2f"
178 if(a>=6)plot inputfile u ($2 < limit ? $10: 1/0):10 ls 3 notitle
179 unset label
180
181 if(a>=7)plot inputfile u ($2 < limit ? $10: 1/0):11 ls 1 notitle
182 if(a>=8)plot inputfile u ($2 < limit ? $10: 1/0):12 ls 1 notitle
183 if(a>=9)plot inputfile u ($2 < limit ? $10: 1/0):13 ls 1 notitle
184 if(a>=10)plot inputfile u ($2 < limit ? $10: 1/0):14 ls 1 notitle
185 if(a>=11)plot inputfile u ($2 < limit ? $10: 1/0):15 ls 1 notitle
186
187 # 7.
188 if(a>3)plot inputfile u ($2 < limit ? $11: 1/0):5 ls 1 notitle
189 if(a>3)plot inputfile u ($2 < limit ? $11: 1/0):6 ls 1 notitle
190 if(a>3)plot inputfile u ($2 < limit ? $11: 1/0):7 ls 1 notitle
191 if(a>=4)plot inputfile u ($2 < limit ? $11: 1/0):8 ls 1 notitle
192 if(a>=5)plot inputfile u ($2 < limit ? $11: 1/0):9 ls 1 notitle
193 if(a>=6)plot inputfile u ($2 < limit ? $11: 1/0):10 ls 1 notitle
194 set label at graph labelxpos ,labelypos center LABELN3 front left font
195   "Arial,24" textcolor rgb "#4f2f2f"
196 if(a>=7)plot inputfile u ($2 < limit ? $11: 1/0):11 ls 3 notitle
197 unset label
198
199 if(a>=8)plot inputfile u ($2 < limit ? $11: 1/0):12 ls 1 notitle
200 if(a>=9)plot inputfile u ($2 < limit ? $11: 1/0):13 ls 1 notitle
201 if(a>=10)plot inputfile u ($2 < limit ? $11: 1/0):14 ls 1 notitle
202 if(a>=11)plot inputfile u ($2 < limit ? $11: 1/0):15 ls 1 notitle
203
204 # 8.
205 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):5 ls 1 notitle
206 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):6 ls 1 notitle
207 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):7 ls 1 notitle
208 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):8 ls 1 notitle
209 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):9 ls 1 notitle
210 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):10 ls 1 notitle
211 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):11 ls 1 notitle
212
213 if(a>=8) set label at graph labelxpos ,labelypos center LABELN4
214   front left font "Arial,24" textcolor rgb "#4f2f2f"
215 if(a>=8) plot inputfile u ($2 < limit ? $12: 1/0):12 ls 3 notitle
216 if(a>=8) unset label
217
218
219 # 9.
220 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):5 ls 1 notitle
221 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):6 ls 1 notitle
222 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):7 ls 1 notitle
223 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):8 ls 1 notitle
224 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):9 ls 1 notitle
225 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):10 ls 1 notitle
226 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):11 ls 1 notitle
227 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):12 ls 1 notitle
228
229 if(a>=9) set label at graph labelxpos ,labelypos center LABELN5
230   front left font "Arial,24" textcolor rgb "#4f2f2f"
231 if(a>=9) plot inputfile u ($2 < limit ? $13: 1/0):13 ls 3 notitle
232 if(a>=9) unset label
233
234 if(a>=10) plot inputfile u ($2 < limit ? $13: 1/0):14 ls 1 notitle
235 if(a>=11) plot inputfile u ($2 < limit ? $13: 1/0):15 ls 1 notitle

```

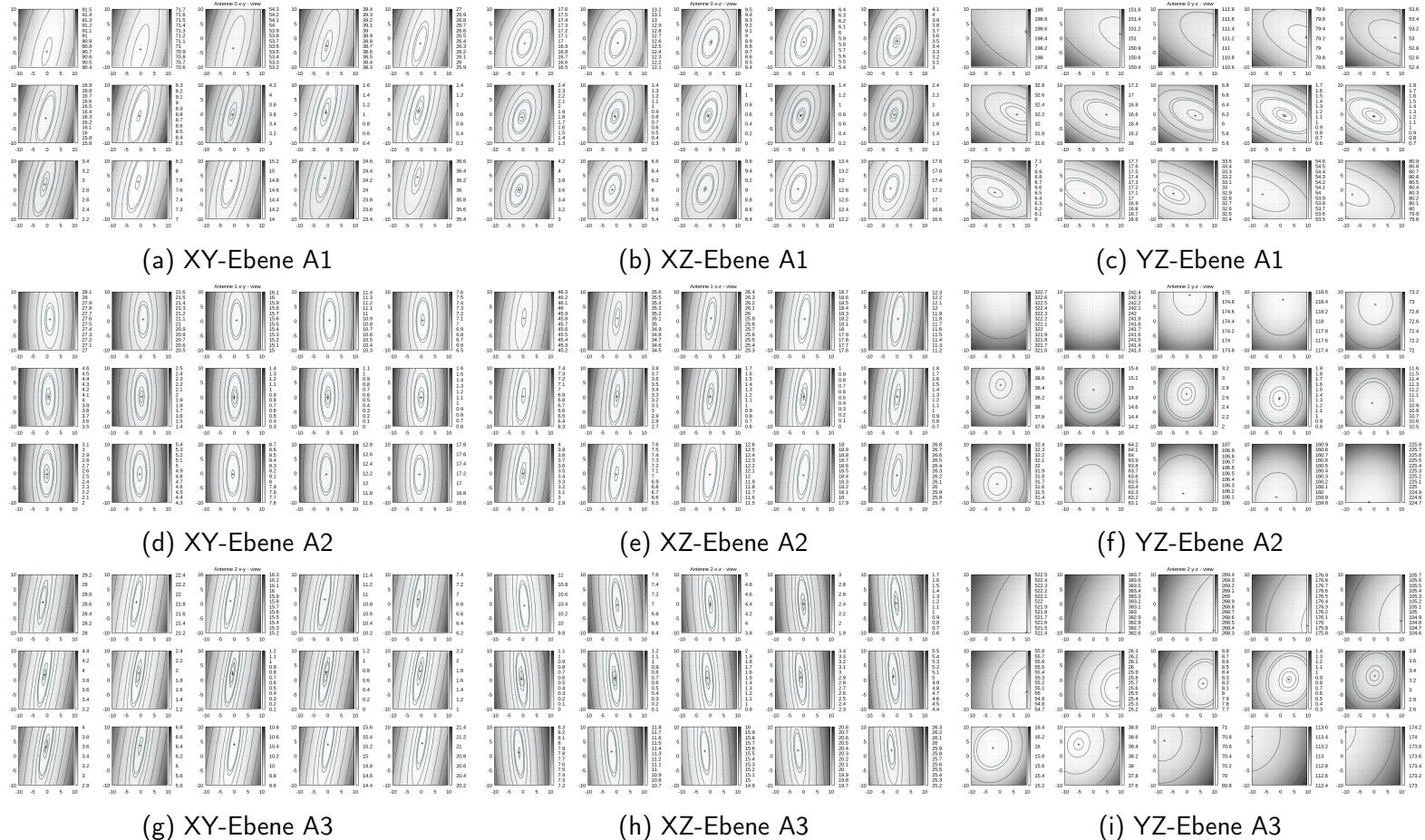
```

236 # 10.
237 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):5 ls 1 notitle
238 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):6 ls 1 notitle
239 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):7 ls 1 notitle
240 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):8 ls 1 notitle
241 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):9 ls 1 notitle
242 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):10 ls 1 notitle
243 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):11 ls 1 notitle
244 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):12 ls 1 notitle
245 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):13 ls 1 notitle
246
247 if(a>=10) set label at graph labelxpos ,labelypos center LABELN6
    front left font "Arial,24" textColor rgb "#4f2f2f"
248 if(a>=10) plot inputfile u ($2 < limit ? $14: 1/0):14 ls 3 notitle
249 if(a>=10) unset label
250
251 if(a>=11) plot inputfile u ($2 < limit ? $14: 1/0):15 ls 1 notitle
252
253 # Generate last row
254 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):5 ls 1 notitle
255 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):6 ls 1 notitle
256 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):7 ls 1 notitle
257 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):8 ls 1 notitle
258 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):9 ls 1 notitle
259 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):10 ls 1 notitle
260 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):11 ls 1 notitle
261 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):12 ls 1 notitle
262 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):13 ls 1 notitle
263 if(a>=10) plot inputfile u ($2 < limit ? $15: 1/0):14 ls 1 notitle
264
265 if(a>=11) set label at graph labelxpos ,labelypos center LABELN7
    front left font "Arial,24" textColor rgb "#4f2f2f"
266 if(a>=11) plot inputfile u ($2 < limit ? $15: 1/0):15 ls 3 notitle
267 if(a>=11) unset label
268
269
270 i=i+1
271
272 unset multiplot
273
274 if ( i < m) reread
275 i=0
276 #print "rm ".ObjectiveOut.remove( ObjectiveOut )
277 #print "rm ".SigmaOut.remove( SigmaOut )
278 #print "rm ".FitnessOut.remove( FitnessOut )
279 #print "rm ".EvalOut.remove( EvalOut )

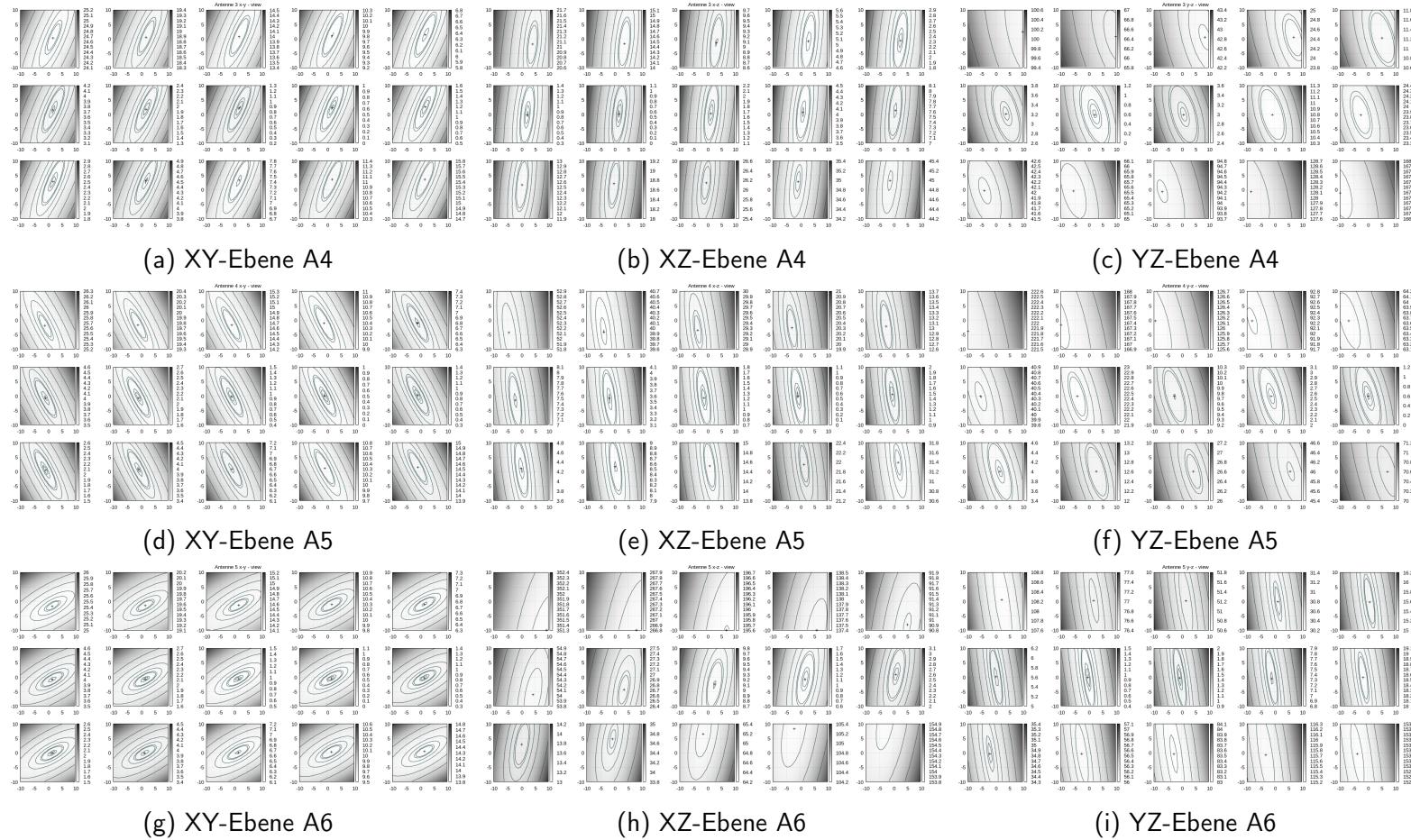
```

D. Fitness Plots

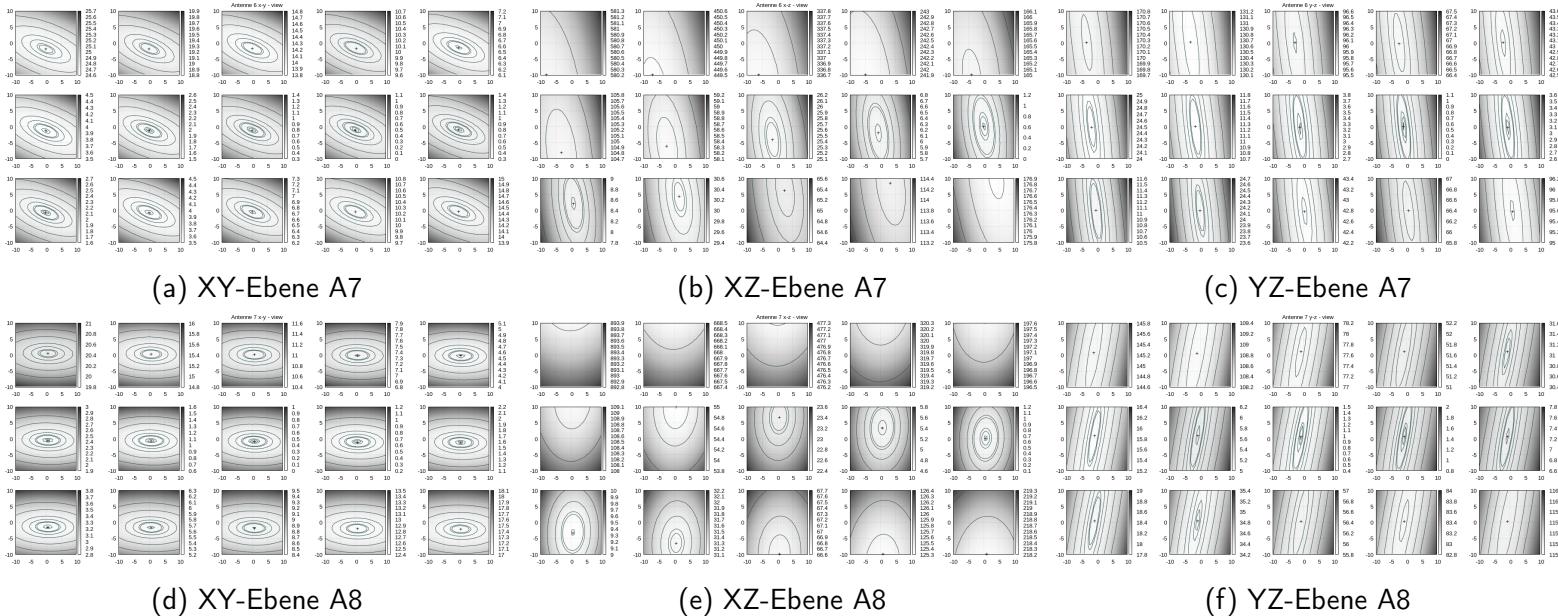
D.1. Antennen 1,2 und 3



D.2. Antennen 4,5 und 6

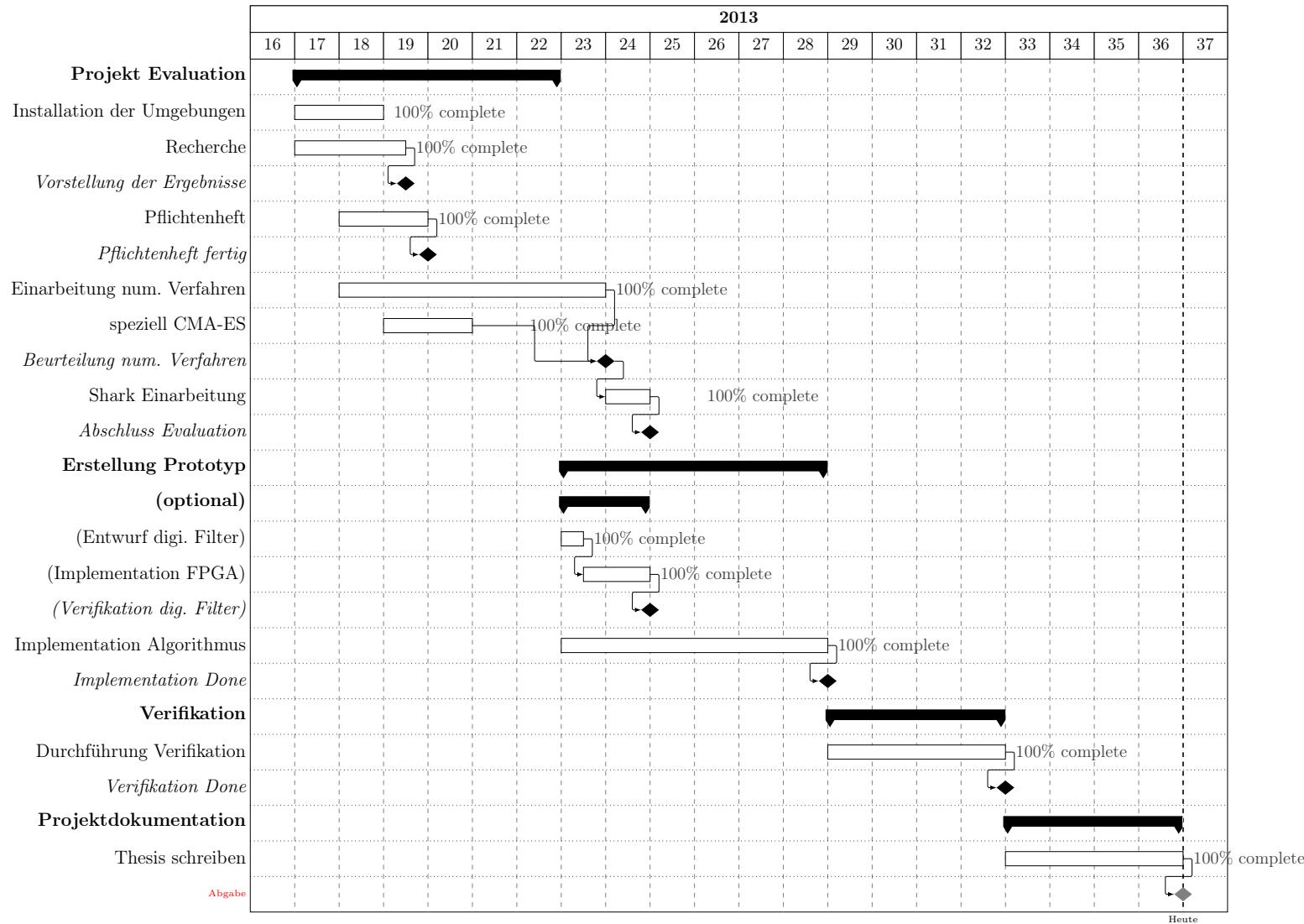


D.3. Antennen 7 und 8



E. Projektverwaltung

E.1. Projektlaufplan



Literaturverzeichnis

- [1] BOMZE, I. ; GROSSMANN, W.: *Optimierung - Theorie und Algorihtmen.* BI Wissensch.Vlg, 1993
- [2] BORGWERTH, Bernd: *Entwicklung von Lösungsmethoden zur Realisierung einer 3D Lokalisierung von RFID-Transpondern in der Medizin, FH-Gelsenkirchen.* 2008
- [3] BORGWERTH, Bernd ; GNIP, Christoph: Abschätzung der Wellenzahl durch Korrelation mit Kaliberpunkten. (2012)
- [4] BRONŠTEJN, I.N. ; SEMENDJAJEV, K.A. ; MUSIOL, G. ; MÜHLIG, H.: *Taschenbuch der Mathematik.* Deutsch Harri GmbH, 2012 <http://books.google.de/books?id=uPKPMAEACAAJ>. – ISBN 9783817120185
- [5] *Electromagnetic compatibility and Radio spectrum Matters (ERM); Radio Frequency Identification Equipment operating in the band 865 MHz to 868 MHz with power levels up to 2 W.* 2010
- [6] FINKENZELLER, K.: *RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC.* Hanser, 2008 <http://books.google.de/books?id=49HTBDrfqFUC>. – ISBN 9783446412002
- [7] HANSEN, N.: Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In: *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, ACM, July 2009, S. 2389–2395
- [8] HANSEN, N. ; KERN, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: YAO, X. (Hrsg.) u. a.: *Parallel Problem Solving from Nature PPSN VIII* Bd. 3242, Springer, 2004 (LNCS), S. 282–291
- [9] HANSEN, Nikolaus: *CMA-ES Anwendungen.* <https://www.lri.fr/~hansen/cmaapplications.pdf>, 2009. – [Online; zuletzt geprüft am 7-Sep-2013]
- [10] In: HANSEN, Nikolaus: *The CMA Evolution Strategy.* 2011

- [11] HANSEN, Nikolaus: *CMA-ES Source Code*. https://www.lri.fr/~hansen/cmaes_inmatlab.html, 2013. – [Online; zuletzt geprüft am 6-Sep-2013]
- [12] HANSEN, Nikolaus ; ROTH, Stefan:
- [13] HERMANN, M.: *Numerische Mathematik*. Oldenbourg Wissensch.Vlg, 2001 <http://books.google.de/books?id=145jSrRdL7AC>. – ISBN 9783486579352
- [14] HTTP://WWW.FOEBUD.ORG/RFIDM: *Digitalcourage*. <http://www.foebud.org/rfid>, 2013. – [Online, zuletzt geprüft am 22.8.2013]
- [15] IGEL, Christian ; HEIDRICH-MEISNER, Verena ; GLASMACHERS, Tobias: *Shark*. In: *Journal of Machine Learning Research* 9 (2008), 993–996. http://image.diku.dk/shark/sphinx_pages/build/html/index.html
- [16] IGEL, Christian ; HEIDRICH-MEISNER, Verena ; GLASMACHERS, Tobias: *Shark*. In: *Journal of Machine Learning Research* 9 (2008), S. 993–996
- [17] KNIPSCHEER, Marius: *Planung eines Entwicklungstools für ein drahtloses 3D-Positionsmesssystem zur Lokalisierung von minimal invasiven chirurgischen Instrumenten*. FH-Gelsenkirchen. 2008
- [18] KOST, Bernd: *Optimierung mit Evolutionsstrategien*. Deutsch Harri GmbH, 2003 <http://books.google.de/books?id=FcgNJIg4lcAC>. – ISBN 9783817116993
- [19] MUZALEWSKI, Mathäus: *Einsatz von Lernverfahren zur Interpolation von Positionsdaten eines RFID-basierten Navigationssystems*. 2011
- [20] PRESS, W.H.: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007 <http://books.google.de/books?id=1aA0dzK3FegC>. – ISBN 9780521880688
- [21] RECHENBERG, I. Werkstatt Bionik und E.: *Evolutionsstrategie '94*. Frommann-Holzboog, Stuttgart, 1994
- [22] RFIDJOURNAL.COM: *RFID-Journal*. <http://www.rfidjournal.com>, 2013. – [Online, zuletzt geprüft am 22.8.2013]
- [23] RITSCHEL, Kai ; DEKOMIEN, Claudia ; WINTER, Susanne: Modellfunktion zur Approximation von Ultraschallkontrastmittelkonzentration zur semi-quantitativen Gewebeperfusionsbestimmung. In: *Bildverarbeitung für die Medizin*, 2012, S. 159–164

- [24] SCHWEFEL, H.-P.: *Evolution and Optimum seeiking*. John Wiley and Son, New York, 1995
- [25] SIMO SÄRKKÄ ; JAAKKOLA, Kaarle ; HUUSKO, Ville V. Viikari M.: Phase-Based UHF RFID Tracking With Nonlinear Kalman Filtering and Smoothing. (2012), February
- [26] SPELLUCCI, M.: *Numerische Verfahren der nichtlinearen Optimierung*. Birkhäuser .Vlg, 1993
- [27] WIKIPEDIA: *CMA-ES Concept of direct optimization — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=File:Concept_of_directional_optimization_in_CMA-ES_algorithm.png&oldid=532567533, 2013. – [Online; zuletzt geprüft am 6-Sep-2013]
- [28] WILLE, Andreas ; WINTER, Susanne: Medical Navigation Based on RFID Tag Signals: Model and Simulation. 55 (2010). <http://dx.doi.org/10.1515>. – DOI 10.1515
- [29] WINTER, Susanne ; RITSCHEL, Kai ; BROLL, Magdalena ; DEKOMIEN, Claudia: Kalibrierung eines 3D-Ultraschallsystems mit evolutionärer Optimierung. In: DESERNO, Thomas M. (Hrsg.) ; HANDELS, Heinz (Hrsg.) ; MEINZER, Hans-Peter (Hrsg.) ; TOLXDORFF, Thomas (Hrsg.): *Bildverarbeitung für die Medizin* Bd. 574, CEUR-WS.org, 2010 (CEUR Workshop Proceedings), S. 187–190