

Bachelorarbeit

Tobias Wulf

Winkelmessung durch magnetische Sensor-Arrays und
Toleranzkompensation mittels Gauß-Prozess

Tobias Wulf

Winkelmessung durch magnetische Sensor-Arrays und Toleranzkompensation mittels Gauß-Prozess

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer Prüfer: Prof. Dr. Karl-Ragmar Riemschneider
Zweitgutachter: Prof. Dr. Klaus Jünemann

Eingereicht am: TT. Monat Jahr

Tobias Wulf

Thema der Arbeit

Winkelmessung durch magnetische Sensor-Arrays und Toleranzkompensation mittels Gauß-Prozess

Stichworte

Sensor-Array Simulation, Dipol, Magnetfeld, Kugelmagnetapproximation, TMR, TDK TAS2141, AMR, NXP KMZ60, Toleranzkompensation, Gauß-Prozess, Kovarianzmatrix, Regression, Winkelvorhersage

Kurzzusammenfassung

...

Tobias Wulf

Title of Thesis

Angular Measurement by Magnetic Sensor Arrays and Tolerance Compensation by Gaussian Process

Keywords

Sensor Array Simulation, Dipole, Magnetic Field, Spherical Magnet Approximation, TMR, TDK TAS2141, AMR, NXP KMZ60, Tolerance Compensation, Gaussian Process, Covariance Matrix, Regression, Angular Prediction

Abstract

...

Inhaltsverzeichnis

1 Motivation 0.0.1 17.02.2021	1
1.1 Stand der Vorarbeiten	2
1.2 Zielstellung	7
2 Grundlagen 0.0.4 25.04.2021	8
2.1 Kreisdarstellung des klassischen Anwendungsfalls	8
2.2 Euklidischer Abstand in Normschreibweise	11
2.3 Magnetische Sensoren und Drehwinkelerfassung	13
2.4 Kennfeldmethode zur Charakterisierung von Sensoren	17
2.5 Prinzip des Sensor-Arrays	21
2.6 Sensor-Array-Simulation über die Dipol-Feldgleichung	25
2.7 Gauß-Prozesse für Regressionsverfahren	29
3 Software-Entwicklung für Optimierungsexperimente 0.0.4 25.04.2021	32
3.1 Aufbau und Funktion	32
3.2 Simulationsprozesse und Ausführung	35
3.2.1 Sensor-Array-Simulation	35
3.2.2 Gauß-Prozess-Regression	37
4 Erprobungs- und Optimierungsexperimente 0.0.2 26.04.2021	44
4.1 Vergleich der Kovarianzfunktionen	44
4.2 Anpassung der Referenzwinkelanzahl	44
4.3 Anpassung des Rauschniveaus	44
4.4 Anpassung der Parametergrenzen	44
5 Auswertung 0.0.1 13.01.2021	45
5.1 Gegenüberstellung der GPR-Modelle	45
6 Zusammenfassung und Bewertung 0.0.1 13.01.2021	46

Algorithmenverzeichnis	50
Glossar	51
Abkürzungen	53
Literatur	54
Anhang	56
A TDK TAS2141-AAAB Kennfelddatensatz 0.0.1 29.03.2021	56
B Sensor-Array-Simulation Implementierung 0.0.1 07.04.2021	59
C Gauß-Prozess-Regression Implementierung 0.0.2 26.04.2021	63
C.1 Modellinitialisierung	64
C.2 Modelloptimierung	74
C.3 Modellvorhersagen	76
C.4 Modellgeneralisierung	79
D Parametrierung der Erprobungs- und Optimierungsexperimente 0.0.1 26.04.2021	81
D.1 Vergleich der Kovarianzfunktionen	82
D.2 Anpassung der Referenzwinkelanzahl	82
D.3 Anpassung des Rauschniveaus	82
D.4 Anpassung der Parametergrenzen	83
E Ergebnisse der Erprobungs- und Optimierungsexperimente 0.0.1 26.04.2021	84
E.1 Vergleich der Kovarianzfunktionen	84
E.2 Anpassung der Referenzwinkelanzahl	87
E.3 Anpassung des Rauschniveaus	88
E.4 Anpassung der Parametergrenzen	89
F Genutzte Software 0.0.3 08.01.2021	92
G Software-Dokumentation 0.0.7 18.04.2021	93
G.1 GaussianProcessDipoleSimulation	94
G.2 Workflows	96
G.2.1 Project Preparation	97

G.2.2	Project Structure	102
G.2.3	Git Feature Branch Workflow	105
G.2.4	Documentation Workflow	107
G.2.5	Simulation Workflow	110
G.3	Executable Scripts	111
G.3.1	publishProjectFilesToHTML	112
G.3.2	generateConfigMat	116
G.3.3	generateSimulationDatasets	126
G.3.4	deleteSimulationDatasets	128
G.3.5	deleteSimulationPlots	129
G.3.6	exportPublishedToPdf	130
G.3.7	demoGPRModule	135
G.3.8	investigateKernelParameters	140
G.3.9	compareGPRKernels	147
G.3.10	compareCpuTimeVsError	154
G.3.11	compareNoiseOptAbility	158
G.4	Source Code	161
G.4.1	sensorArraySimulation	162
G.4.1.1	rotate3DVector	164
G.4.1.2	generateDipoleRotationMoments	166
G.4.1.3	generateSensorArraySquareGrid	171
G.4.1.4	computeDipoleH0Norm	175
G.4.1.5	computeDipoleHField	177
G.4.1.6	simulateDipoleSquareSensorArray	182
G.4.2	gaussianProcessRegression	189
G.4.2.1	initGPR	192
G.4.2.2	initGPROptions	194
G.4.2.3	initTrainDS	197
G.4.2.4	initKernel	199
G.4.2.5	initKernelParameters	201
G.4.2.6	tuneKernel	203
G.4.2.7	computeTuneCriteria	205
G.4.2.8	predFrame	206
G.4.2.9	predDS	209
G.4.2.10	lossDS	211
G.4.2.11	optimGPR	213

G.4.2.12	computeOptimCriteria	215
G.4.2.13	kernelQFCAPX	217
G.4.2.13.1	QFCAPX	218
G.4.2.13.2	meanPolyQFCAPX	220
G.4.2.13.3	initQFCAPX	222
G.4.2.14	kernelQFC	224
G.4.2.14.1	QFC	225
G.4.2.14.2	meanPolyQFC	227
G.4.2.14.3	initQFC	229
G.4.2.15	basicMathFunctions	231
G.4.2.15.1	sinoids2angles	233
G.4.2.15.2	angles2sinoids	237
G.4.2.15.3	decomposeChol	239
G.4.2.15.4	frobeniusNorm	241
G.4.2.15.5	computeInverseMatrixProduct	243
G.4.2.15.6	computeTransposeInverseProduct	246
G.4.2.15.7	addNoise2Covariance	249
G.4.2.15.8	computeAlphaWeights	251
G.4.2.15.9	computeStdLogLoss	252
G.4.2.15.10	computeLogLikelihood	254
G.4.2.15.11	estimateBeta	256
G.4.3	util	258
G.4.3.1	removeFilesFromDir	259
G.4.3.2	publishFilesFromDir	261
G.4.3.3	plotFunctions	264
G.4.3.3.1	plotTDKCharDataset	266
G.4.3.3.2	plotTDKCharField	273
G.4.3.3.3	plotTDKTransferCurves	278
G.4.3.3.4	plotKMZ60CharDataset	282
G.4.3.3.5	plotKMZ60CharField	288
G.4.3.3.6	plotKMZ60TransferCurves	293
G.4.3.3.7	plotDipoleMagnet	297
G.4.3.3.8	plotSimulationDataset	302
G.4.3.3.9	plotSingleSimulationAngle	312
G.4.3.3.10	plotSimulationSubset	321
G.4.3.3.11	plotSimulationCosSinStats	329

G.4.3.3.12	plotSimulationDatasetCircle	337
G.5	Datasets	346
G.5.1	TDK TAS2141 Characterization	347
G.5.2	NXP KMZ60 Characterization	353
G.5.3	Config Mat	359
G.5.4	Training and Test Datasets	360
G.6	Unit Tests	364
G.6.1	runTests	366
G.6.2	removeFilesFromDirTest	367
G.6.3	rotate3DVectorTest	368
G.6.4	generateDipoleRotationMomentsTest	370
G.6.5	generateSensorArraySquareGridTest	371
G.6.6	computeDipoleH0NormTest	372
G.6.7	computeDipoleHFieldTest	373
G.6.8	tiltRotationTest	377
	Selbstständigkeitserklärung	379

1 Motivation 0.0.1 17.02.2021

Magnetische Sensoren erlauben die berührungslose Erfassung von Drehzahlen und Winkelinformationen. In modernen Automobilen werden sie unter anderem in der Motorelektronik und im Bremsystem eingesetzt. Neuentwicklungen in der Halbleitertechnik, auf Basis des TMR-Effekts, ermöglichen den Aufbau komplexerer Sensorstrukturen [15]. Die Arbeitsgruppe Sensorik an der HAW Hamburg erforscht moderne Ansätze der Signalverarbeitung für neu gewonnene Sensorstrukturen, verwirklicht als magnetische Sensor-Arrays. Durch den Aufbau von Sensoren als Arrays, bieten sich Möglichkeiten zur Nutzung von Algorithmen und Regressionsverfahren an, die eine Kompensation und Detektion von mechanischen Toleranzen zulassen [19].

Das Verarbeiten einer Vielzahl an Messwerten, bedingt durch Sensor-Array-Strukturen, ist hierbei eine der Herausforderungen die es zu bewältigen gilt. Mit Hilfe moderner Algorithmen, die Ansätze des maschinellen Lernens beinhalten, ergeben sich weitere Problemstellungen in Bezug auf Modellabbildung- und Optimierung. Das übergeordnete Ziel bei der Lösung und Bewältigung der einzelnen Etappen ist die Verbesserung der Messgenauigkeit, indem individuelle Abweichungen des Sensors einem geeigneten Modell antrainiert und Modellparameter optimiert werden.

Moderne Regressionsverfahren liefern dabei statistische Ansätze um geeignete Qualitätskriterien zu bilden und somit trainierte Modelle und ihre Messwertgenauigkeit bewerten zu können, sodass eine Erprobung und Bewertung der erstellten Modelle, mit Toleranz-Abweichungen in den Eingangsdaten, während einer Arbeitsphase untersucht werden können. Diese Arbeit konzentriert dabei auf die simulative Abbildung eines Tunnel-Magnetoresistance (TMR)-Sensormodells für die Drehwinkelerfassung.

1.1 Stand der Vorarbeiten

Zur Erörterung der Ziele und Inhalte dieser Arbeit, findet einleitend eine kurze Zusammenfassung der Vorarbeiten statt. Für den Inhalt relevante Aspekte der Vorarbeiten werden im Kapitel 2 näher beleuchtet und erklärt.

Aktuell steht kein magnetisches TMR-Sensor-Array als integrierte Lösung zur Verfügung. Im Zuge des Forschungsprojekts Signalverarbeitung für Integrated-Sensor-Array (ISAR) sind in der Arbeitsgruppe Sensorik Machbarkeitsstudien erbracht worden [14][17]. Zielstellung war dabei die Untersuchung der generellen Funktionalität und technischen Umsetzung eines magnetischen Sensor-Arrays im Maßstabsmodell.

Platinen-Sensor-Array

Für den Aufbau des Platinen-Sensor-Arrays sind einzelne Winkelsensoren in Sensorbändern angeordnet, Abbildung 1.1. Die Messwerterfassung erfolgt über ein Multiplexing-Verfahren. Eine Steuerung des Multiplexings und die weitere Messwertverarbeitung erfolgt mit Hilfe eines Mikrocontrollers.

Diese Herangehensweise lässt eine Untersuchung der technischen Machbarkeit auf der Basis von aktuell verfügbaren Technologien und Winkelsensoren zu. So ist das Platinen-Sensor-Array in verschiedenen Versionen, mit Anisotrope-Magnetoresistance (AMR)-Sensoren der Firma NXP Semiconductors (KMZ60) [8] und TMR-Sensoren der Firma TDK (TAS2141-AAAB) [11] verwirklicht worden. Das Maßstabsmodell des magnetischen Sensor-Arrays kann zu Vergleichs- und weiteren Erprobungsarbeiten genutzt werden. Diese können beispielsweise in Simulationen und Hardware-Optimierungsarbeiten einfließen.

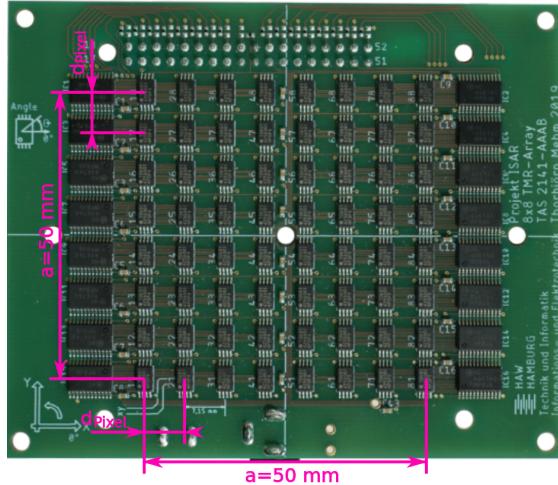


Abbildung 1.1: Platinen-Sensor-Array im Maßstab aufgebaut als 8×8 Sensor-Array, dass als Aufsteckmodul für eine Mikrocontroller getriebene Signalverarbeitung bereitsteht. Die einzelnen Sensoren sind in Sensorbänken angeordnet. Die Anordnung erfolgt in eine linke und rechte Sensorbank pro Reihe auf der Platine. Eine Sensorbank besteht jeweils aus einem Multiplexer-IC und vier daneben liegenden Sensor-ICs. Abbildung entnommen und bearbeitet aus [14].

Simulationsmodell des Sensor-Arrays

Einen weiteren Ansatz, der durch die Arbeitsgruppe Sensorik verfolgt wird, ist die Entwicklung eines Simulationsmodells auf Grundlage von Charakterisierungsdatensätzen. Hierfür wird ein einzelnes Sensor-IC, z.B. der TMR-Sensor TAS2141-AAAB der Firma TDK, nach einer bestimmten Kennfeldmethode [15] charakterisiert. Der so gewonnene Datensatz kann dann, durch geeignete Interpolationsverfahren, in einer Simulation zur Generierung eines magnetischen Sensor-Arrays genutzt werden. In Abbildung 1.2 ist das Kernprinzip des Simulationsansatzes vereinfacht dargestellt. Es wird ein Simulationsmodell aufgebaut, dass Charakterisierungsdatensätze verarbeiten kann und entsprechende Charakteristiken eines einzelnen Sensor-ICs zu einem Sensor-Array interpoliert. Abhängig von weiteren gewählten Eigenschaften des Sensor-Arrays, wie geometrische Anordnung und Größe, produziert das interpolierte Modell Simulationsdatensätze, die das Verhalten des einzelner Sensor-ICs ortsabhängig im Sensor-Array abbilden.

Der Simulationsansatz besitzt ebenfalls den Vorteil Modelle aufzubauen, die sich auf heute zur Verfügung stehenden Technologien beziehen. Weitere Vorteile sind die Mani-

pulationsfähigkeit der Sensor-Array-Geometrie und -Größe. So bieten sich Möglichkeiten magnetische Sensor-Arrays in verschiedenen Maßstäben und geometrischen Formen zu simulieren. Des weiteren können verschiedene Anwendungsszenarien simuliert werden. Eine Problemstellung die sich dabei ergibt, ist die physikalisch sinnvolle Stimulanz des Simulationsmodell. Für das Platinen-Sensor-Array ist im trivialen Anwendungsfall die Stimulanz ein simpler Permanentmagnet. In der Simulation muss eine entsprechende Stimulierung des Sensor-Arrays über magnetische Feldgleichungen gelöst werden [12][15], wobei weitere Problemstellungen zur richtigen Dimensionierung oder Approximation des zu simulierenden Magnetfeldes auftreten.



Abbildung 1.2: Ansatzdarstellung zur Generierung eines Simulationsmodells des magnetischen Sensor-Arrays. Sensor spezifische Charakteristiken (Kennfelder) werden in einem Charakterisierungsdatensatz gespeichert und im Anschluss das Verhalten des Einzelexemplars zu einem Sensor-Array interpoliert. Die Simulation des interpolierten Sensor-Arrays erzeugt eine höhere Abstraktionsebene, deren Ergebnisse wiederum in Simulationsdatensätze gespeichert sind und zur weiteren Analyse und Evaluierung genutzt werden können. Die Abstraktion der Kennfelder soll hier das Prinzip des Simulationsansatzes veranschaulichen. Im Simulationsmodell werden keine Arrays von Kennfeldern aufgebaut, sondern Charakteristiken des einzelnen Kennfeldes entnommen und interpoliert. Die grau unterlegten Abschnitte kennzeichnen Verfahrensschritte, in denen Datensätze zur Verfügung stehen oder erzeugt werden.

Das Sensor-Array-Modell, ob als Platinen-Modell oder Simulationsmodell, repräsentiert im Kontext nur die erste Hälfte eines modernen, vollwertigen Sensor-ICs. Seine Aufgabe besteht darin eine physikalische Anregung (Magnetfeld) in elektrische, analoge Signale umzuwandeln. Dieser Teil eines Sensor-ICs wird zumeist als Sensorkopf bezeichnet, da eine sinnbildliche darunter liegende Einheit die weitere Signalverarbeitung und -Auswertung übernimmt. Es handelt sich dabei um eine anwendungsspezifische integrierte Schaltung, engl. Application-Specific-Integrated-Circuit (ASIC). Beide Teile zusammen, der Sensorkopf und das Signalverarbeitungs-ASIC, bilden ein vollständiges Sensor-IC mit der Fähigkeit zur modernen Signalverarbeitung. Unterstützend zeigt Abbildung 1.3 die allgemeine Aufbaubeschreibung eines Sensor-IC und Unterteilung in Sensorkopf und ASIC, respektive Signalerzeugung und Signalverarbeitung.

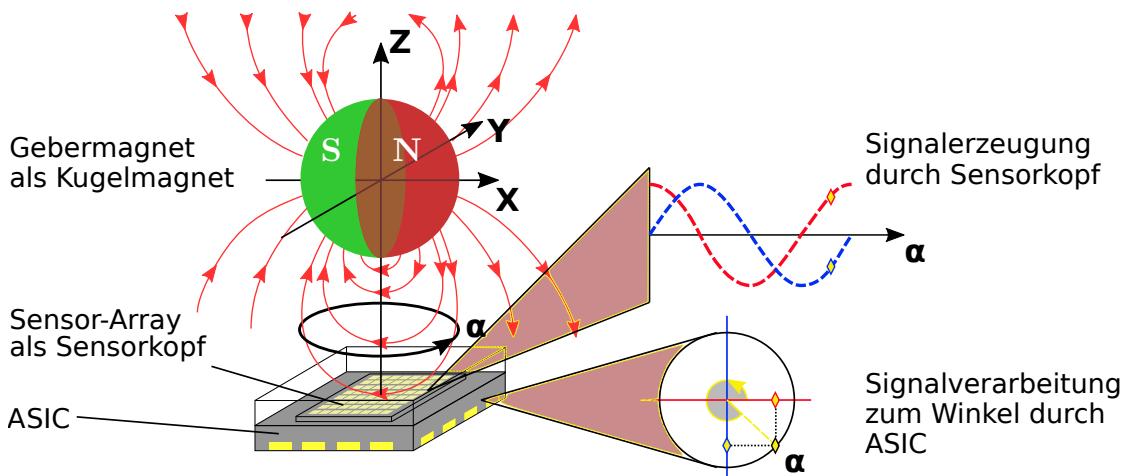


Abbildung 1.3: Veranschaulichung eines vollständigen Sensor-ICs für die Drehwinkel erfassung. Stark vereinfachte Darstellung eines Sensor-IC bestehend aus einem Sensorkopf und ASIC. Zu sehen sind die übergeordneten Aufgaben von Sensorkopf und ASIC. Der Sensorkopf erfasst die physikalische Stimulation (hier Kugelmagnetfeld) und setzt diese in analoge Signale um. Eine anschließende Signalverarbeitung findet im ASIC statt, der die elektrischen Signale zur entsprechenden Winkelausgabe abstrahiert. Dargestellt ist die Signalerzeugung eines einzelnen Punktes auf dem magnetischen Sensor-Arrays. Grafik entnommen und bearbeitet aus [19].

ASIC - Konzeptionierung der Kernfunktionalität

Derzeitig befinden sich die Forschungsprojektarbeiten für einen tauglichen ASIC in der Konzeptionsphase. Die Kernfunktionalität eines ASIC-Designs wird durch ein mathematisches Modell oder Verfahren abgebildet, dass in der Lage ist vom Sensorkopf erzeugte Messwerte adäquat und ausreichend schnell zu verarbeiten. Dabei muss ein solches Modell oder Verfahren grundlegende Eigenschaften des physikalischen Gesamtsystems in sich vereinigen und diese repräsentativ in den Gesamtkontext der Applikation setzen können. Im Kontext dieser Arbeit ist die Sensorapplikation, durch die Drehwinkelerfassung einer kreisförmigen Sensoranregung dargestellt, wie es in Abbildung 1.3 angedeutet ist.

Erfolgte Vorarbeiten der Arbeitsgruppe Sensorik für ein ASIC-Design, umfassen die Entwicklung eines mathematischen Modells und erste theoretische Simulationen [15][18][19]. Die Simulation bindet dabei Datensätze ein, die durch das Sensor-Array-Simulationsmodell erzeugt werden. Das mathematische Modell der ASIC-Kernfunktionalität ist auf Grundlage von Gauß-Prozessen für Regressionsverfahren entwickelt [3] worden. Die bisherigen Simulationsarbeiten beschränken sich auf mathematische Simulationen, die auf eine Gültigkeitsprüfung des mathematischen ASIC-Modells abzielen und Ansätze zur Modellqualifizierung und Qualitätskriterien für die Signalverarbeitung mit beinhalten.

1.2 Zielstellung

- Bezug zu Vorarbeiten
- Verfeinerung des Simulationsmodell des magnetischen Sensor-Arrays
- Skalierung des approximierten Kugelmagnetanregungsfeldes
- Optimierung des mathematischen Model für die ASIC-Kernfunktionalität
- Aufschlüsselung der Modellparameter
- Überführung von Skript basierten Entwürfen in Funktionsmodule
- Modularer Modellaufbau, der Modularerweiterungen zulässt

2 Grundlagen 0.0.4 25.04.2021

Das Fundament für die Drehwinkel erfassung mittels magnetischem Sensor-Array und lernender Signalverarbeitung [15][19][20] bildet das Regressionsverfahren für Gauß-Prozesse [3] und die damit verbundene Abstandsmessung von Winkelpositionen auf einer Kreisbahn. Für eine anschauliche Erklärung der Grundlagen, sollen die Zusammenhänge anhand einfacher Kreisdarstellung des Messprinzips eines einzelnen Winkelsensors gezeigt werden, sodass dieses später in der Verwendung eines Sensor-Arrays adaptierbar ist und mittels geeigneter Rechen- und Normierungsverfahren auf die Problemstellung eines höherdimensionalen Systems projiziert werden kann.

2.1 Kreisdarstellung des klassischen Anwendungsfalls

Im klassischen Anwendungsfall, zu sehen in Abbildung 2.1, ist ein Gebermagnet räumlich zentriert über einem magnetischen Sensor platziert. Bei Drehung des Gebermagneten rotiert sein Magnetfeld entsprechend mit. Die Rotation findet um die Z -Achse des Gebermagneten statt. Die Nord-Süd-Ausrichtung des Magneten liegt in der X - bzw. Y -Achse des Koordinatensystems [8][11].

Der Winkelsensor misst die zueinander und zur Rotationsachse orthogonal stehenden X - und Y -Feldstärkenkomponenten des Gebermagneten H_x und H_y . Diese setzt der Winkelsensor in elektrische Spannungssignale um. Die Winkelstellung α des Magnet wird somit nicht direkt gemessen. Sie kann aber, mittels der gemessenen H_x und H_y Feldstärkenkomponenten, durch einfache Vektorrechnung berechnet werden.

Bei idealer und gleichbleibender Position des Gebermagneten in Relation zum Winkelsensor, liefern die aufgenommen H_x -/ H_y -Messwerte eine Cosinus-Funktion $V_{cos}(H_x, H_y)$ sowie eine um 90° zur Cosinus-Funktion phasenverschobene Sinus-Funktion $V_{sin}(H_x, H_y)$. Genaue physikalische Größenzusammenhänge und technische Umsetzung sind dabei vorerst in den weiteren Darstellungen vernachlässigt.

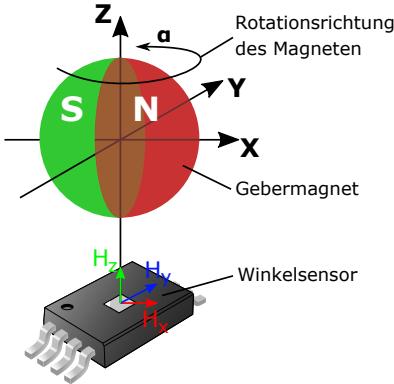


Abbildung 2.1: Klassischer Anwendungsfall für die Drehwinkelerfassung. Zeigt einen, um seine Z -Achse rotierenden, Gebermagneten und einen Winkelsensor in zentrierter und orthogonaler Ausrichtung zur Z -Achse des Magneten. Idealerweise befinden sich Magnet und Sensor, ohne Verkipplungen in X - oder Y -Richtung, parallel zueinander. Grafik entnommen und bearbeitet aus [20].

Durch die Phasenverschiebung der Sinus-Funktion stehen die Messwerte $V_{cos}(H_x, H_y)$ und $V_{sin}(H_x, H_y)$ vektoriell orthogonal zueinander. Bedingt durch die Orthogonalität der Messwerte $V_{cos}(H_x, H_y) \perp V_{sin}(H_x, H_y)$ und gleichförmige Kreisbewegung des Magneten um seine Z -Achse, beschreibt die Winkelmessung in polarer Darstellung eine konstante Kreisbahn. Diese besitzt einen konstanten Bahnradius r und die Winkelstellung α des Gebermagneten [15].

Für eine beliebige Winkelmessung \mathbf{A} , die eine entsprechende Winkelstellung α des Gebermagneten abbildet $\mathbf{A} \mapsto \alpha$, ergibt sich somit folgender vektorieller Zusammenhang in Gleichung 2.1 [19].

$$\underbrace{\begin{pmatrix} H_x(\alpha) \\ H_y(\alpha) \end{pmatrix}}_{\text{Gebermagnetfeld}} \Rightarrow \underbrace{\begin{pmatrix} V_{cos}(H_x, H_y) \\ V_{sin}(H_x, H_y) \end{pmatrix}}_{\text{Winkelsensormesswerte}} = \underbrace{\begin{pmatrix} r \cdot \cos(\alpha) \\ r \cdot \sin(\alpha) \end{pmatrix}}_{\text{Kreisdarstellung}} = \underbrace{\begin{pmatrix} a_x \\ a_y \end{pmatrix}}_{\text{Winkelmessung}} = \mathbf{A}(\alpha) \quad (2.1)$$

Die so erhobene Winkelmessung \mathbf{A} nach Gleichung 2.1, bildet ein eindimensionales Vektorfeld mit $\{a_x, b_x\} \in \mathbb{R}$ ab. Wobei sich der Bahnradius r für die Kreisdarstellung, aus dem Betrag der Messung $|\mathbf{A}|$, nach Gleichung 2.2 gewinnen lässt.

$$r = |\mathbf{A}| = \sqrt{(V_{cos}(H_x, H_y))^2 + (V_{sin}(H_x, H_y))^2} = \sqrt{a_x^2 + a_y^2} \quad (2.2)$$

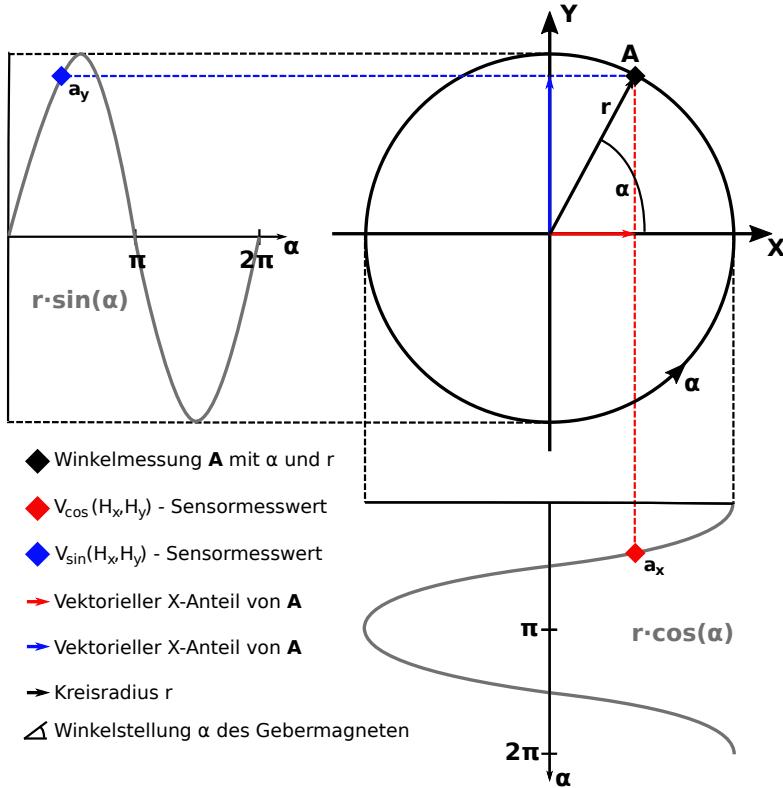


Abbildung 2.2: Kreisdarstellung der Winkelmessung. Als Abbildung der Winkelmessung $\mathbf{A} \mapsto \alpha$ aus Gleichung 2.1. Die Zusammensetzung der Messung erfolgt durch die vom Winkelsensor gemessenen vektoriellen Anteile für die polare Darstellung der Gebermagnetwinkelstellung.

Der entsprechende Winkel des Gebermagneten lässt sich, mittels Überführung in Polarkoordinaten, nach Gleichung 2.3 zurückrechnen. Die Abbildung 2.2 veranschaulicht den Zusammenhang zwischen Messwerten und Abbildung der Gebermagnetwinkelstellung. Die sinoiden Messergebnisse sind der Funktion arctan2 zuzuführen. Diese bildet einen Winkel von null bis π ab und besitzt eine Sprungstelle bei π . Der Y -Anteil kann dabei als Entscheider genutzt werden, um eine Abbildung des Winkels auf eine volle Kreisumdrehung (2π) umzusetzen.

$$\alpha = \begin{cases} \text{arctan2}(a_y, a_x) & \text{f. } a_y > 0 \\ \pi & \text{f. } a_y = 0 \\ \text{arctan2}(a_y, a_x) + 2\pi & \text{f. } a_y < 0 \end{cases} \quad (2.3)$$

2.2 Euklidischer Abstand in Normschreibweise

Um adäquate Bezüge bzw. Abstände zwischen einzelnen Messwerten herzustellen, ist ein Wechsel der Betrachtungsweise notwendig. Es erleichtert die Handhabung der Problemstellung Vektorbeträge als normierte Längen und Distanzen zu sehen. Betrachtet man die vektoriellen Zusammenhänge der klassischen Anwendung aus Abschnitt 2.1 in Normschreibweise, ergibt sich der Radius r für eine Winkelstellung $\mathbf{A} \mapsto \alpha_1$ nach Gleichung 2.4. Die einzelnen Vektorelemente sind entsprechend der Vektor-2-Norm [9] zum Radius r normiert.

$$r = |\mathbf{A}| = \sqrt{a_x^2 + a_y^2} = \sqrt{\sum_{i=1}^n |A_i|^2} = \|\mathbf{A}\|_2 \quad (2.4)$$

Es ist weithin von einer idealen Ausrichtung von Sensor und Gebermagnet wie in Abbildung 2.1 auszugehen. Somit bleibt der Kreisbahn Radius r für eine zweite Winkelstellung mit $\mathbf{B} \mapsto \alpha_2$ konstant.

$$r = \|\mathbf{A}\|_2 = \|\mathbf{B}\|_2 = \text{konst.} \quad (2.5)$$

Der direkte Abstand zwischen den beiden Winkelstellungen $\mathbf{A} \mapsto \alpha_1$ und $\mathbf{B} \mapsto \alpha_2$ lässt sich geometrisch über den Satz des Pythagoras ermitteln. Dafür werden Abstandquadrate aus den Einzeldifferenzen der vektoriellen X -/ Y -Anteile gebildet. Das resultierende Abstandsquadrat bildet mit seiner Kantenlänge dann den Winkelabstand zwischen beiden Winkelstellungen. Abbildung 2.3 veranschaulicht das Vorgehen.

$$\begin{aligned} d_E(\mathbf{A}, \mathbf{B}) &= \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \\ &= \sqrt{\sum_{i=1}^n (A_i - B_i)^2} = \|\mathbf{A} - \mathbf{B}\|_2 \end{aligned} \quad (2.6)$$

Die Überführung in die Normschreibweise des Abstandes ergibt nach Gleichung 2.6 eine Vektor-2-Differenznorm und ist allgemein als euklidischer Abstand bekannt. Analog dazu bildet sich das Quadrat nach Gleichung 2.7.

$$d_E^2 \langle \mathbf{A}, \mathbf{B} \rangle = (a_x - b_x)^2 + (a_y - b_y)^2 = \|\mathbf{A} - \mathbf{B}\|_2^2 \quad (2.7)$$

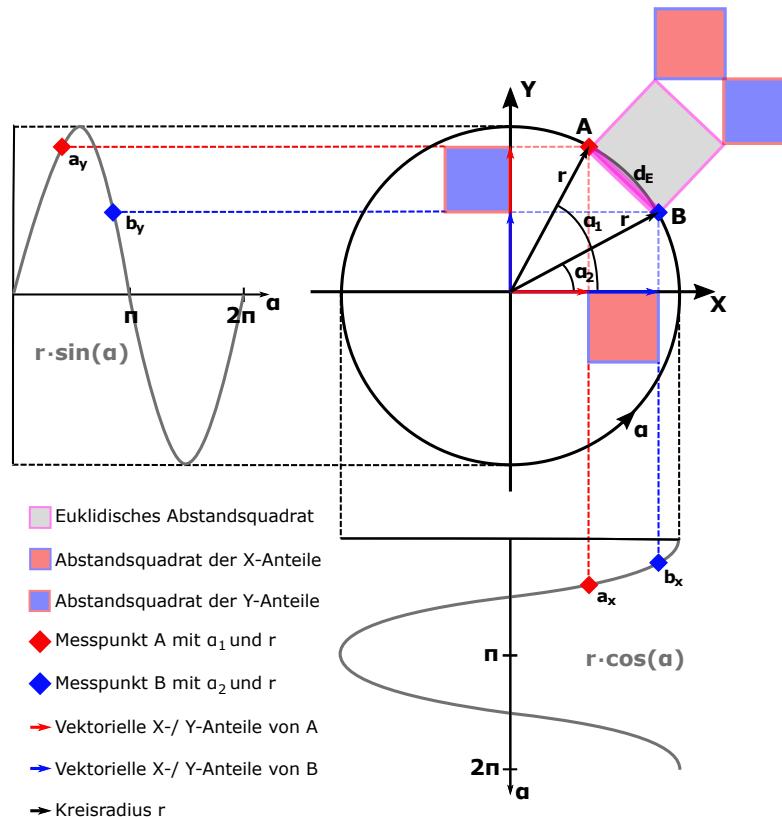


Abbildung 2.3: Allgemeine Kreisdarstellung des euklidischen Winkelabstands. Die Kreisdarstellung zeigt den euklidischen Winkelabstand zweier Winkelmesspunkte \mathbf{A} und \mathbf{B} mit gleichem Kreisradius r . Der euklidische Abstand, bzw. das Abstandsquadrat, zwischen den Winkelposition \mathbf{A} und \mathbf{B} ist zerlegt in Abstandsquadrateanteile. Die Abstandsquadrateanteile ergeben sich aus der vektoriellen Zusammensetzung in X-/Y-Anteile für die einzelnen Messpunkte \mathbf{A} und \mathbf{B} .

Für Vektor-2-Normen muss die Dreiecksungleichung aus Gleichung 2.2 [4][9] gelten. Über die Ungleichung lassen sich Einzelnormen approximiert im Vergleich zu Differenznormen zwischen zwei Punkten **A** und **B** darstellen. Dieser Ansatz kann genutzt werden, wenn der Bahnradius r nicht mehr konstant ist und somit $\|\mathbf{A}\|_2 \neq \|\mathbf{B}\|_2$ ist. Der Ansatz begünstigt die Projektion von orthogonalen Systemen in höherem Normraum [4] und stellt somit die Grundlage für eine Adaptierung auf ein Sensor-Array dar.

$$\begin{aligned} |\|\mathbf{A}\|_2 - \|\mathbf{B}\|_2| &\leq \|\mathbf{A} \pm \mathbf{B}\|_2 \leq |\|\mathbf{A}\|_2 + \|\mathbf{B}\|_2| \\ &\Updownarrow \\ (\|\mathbf{A}\|_2 - \|\mathbf{B}\|_2)^2 &\leq \|\mathbf{A} \pm \mathbf{B}\|_2^2 \leq (\|\mathbf{A}\|_2 + \|\mathbf{B}\|_2)^2 \end{aligned} \quad (2.8)$$

2.3 Magnetische Sensoren und Drehwinkelerfassung

Magnetische Sensoren besitzen eine lange Tradition in der Automobilindustrie. Sie eignen sich besonders durch die berührungslose Erfassung von mechanischen Bewegungen und die kontaktlose Strommessung für den Einsatz in der Fahrzeugtechnik. Es existieren verschiedene Sensoren, die durch unterschiedliche magnetoresistive Effekte realisiert sind. Dabei bildet sich das Grundprinzip durch Anlegen eines äußeren Magnetfeldes und eine resultierende Änderung des elektrischen Widerstandes eines Materials [21].



Abbildung 2.4: Schichtmodelle dreier magnetoresistiver Effekte. a) AMR-Effekt, schwache Widerstandsänderung. b) GMR-Effekt stärkere Widerstandsänderung. c) TMR-Effekt stärkste Widerstandsänderung. Grafik entnommen aus [10].

AMR-Effekt

In der Mitte des 19. Jahrhunderts entdeckte der britische Physiker William Thomson den anisotropen magnetoresistiven Effekt (AMR). Der AMR-Effekt basiert auf einer von Strom- und Magnetisierungsrichtung abhängigen Streuung von Elektronen in einer einzelnen aktiven Schicht, Teil a) der Abbildung 2.4. Diese Schicht besteht in der Praxis oftmals aus einer Nickel-Eisen-Legierung. Die typische Variation der relativen Widerstandsänderung $\Delta R/R$ liegt im Bereich von 2% bis 3% [21]. Für eine eindeutig Winkelmessung werden zwei Wheatstone'sche Brücken aus dem Schichtmaterial aufgebaut. Die Stromdurchflussrichtung ist horizontal. Bedingt durch den AMR-Effekt ist eine Periodizität von 180° abgedeckt [10][21]. Ein mittels AMR-Effekt entwickelter Sensor für die Drehwinkel erfassung, besitzt daher zwei um 45° verdrehte Wheatstone-Brücken. Durch die schwache Widerstandsänderung des Materials ist eine nachgeschaltete Verstärkerschaltung notwendig [8].

GMR-Effekt

Der riesige magnetoresistive Effekt, engl. Giant-Magnetoresistance (GMR), ist 1988 von Grünberg und Fert entdeckt worden. Beide erhielten dafür 2007 den Nobelpreis für Physik, da unter Ausnutzung des GMR-Effekts sich die Speicherkapazität von Computerfestplatten stark erhöhen ließ [10]. Das Minimalprinzip für einen solchen Sensor bildet sich aus zwei magnetischen Dünnschichten, die durch eine nicht magnetische Schicht (z.B. Kupfer) voneinander getrennt sind. Dabei folgt die Magnetisierung der aktiven Schicht (z.B. Nickel-Eisen) einem von außen angelegten Magnetfeld, während die Magnetisierung der zweiten Schicht (z.B. Kobalt-Eisen) durch eine darunter liegende antiferromagnetische Schicht (z.B. Platin-Mangan) fixiert ist. Die Stromdurchflussrichtung bleibt wie beim AMR horizontal [10][21]. Die relativen Widerstandsänderung $\Delta R/R$ ist abhängig von der relativen Ausrichtung der Magnetisierungen in beiden magnetischen Schichten und liegt für einfache Schichtsystem, wie in Teil b) der Abbildung 2.4 gezeigt, bei etwa 10%. Die Herstellung eines GMR-Sensors ist deutlich aufwendiger, als es beim AMR-Effekt der Fall ist. So können aber in Multilagen mit vielfacher Wiederholung der magnetischen Schichten bis zu 80% $\Delta R/R$ erreicht werden [21]. Mit der GMR Technologie aufgebaute Drehwinkelsensoren haben eine Periodizität von 360° und besitzen zwei um 90° verdrehte Wheatstone-Brücken und ebenfalls nachgeschaltete Verstärkerseinheiten [13].

TMR-Effekt

Im Jahr 1975 ist der tunnel-magnetoresistive Effekt (TMR) durch M. Jullière entdeckt worden. Im einfachsten Fall, wie Teil c) Abbildung 2.4 zeigt, tritt der Effekt bei Schichtsystemen auf, die durch eine isolierende Schicht (z.B. Magnesiumoxid) getrennt sind [10]. Die Stromdurchflussrichtung ist im Gegensatz zum AMR und GMR vertikal zu den Schichten. Die relativen Widerstandsänderung $\Delta R/R$ erfolgt in Abhängigkeit zur relativen Ausrichtung der Magnetisierungen beider magnetischen Schichten, die an der Isolationsschicht angrenzen.

Wie beim GMR folgt die aktive Schicht einem äußeren Magnetfeld, ebenso ist die zweite Schicht durch eine antiferromagnetische Schicht fixiert [21]. Der zugrunde liegende Effekt ist aber physikalisch ein gänzlich anderer. Hier “tunnelt” der Stromfluss durch die Isolationsschicht. Das ist ein quantenmechanischer Effekt und kann mit Ansätzen der “normalen” Physik nicht mehr erklärt werden. Zurückzuführen ist der Effekt auf die Spin-Polarisation der einzelnen Elektroden eines magnetischen Tunnel-Kontaktes [21].

In praktischen Ausführungen bei Raumtemperatur liegen heute relative Widerstandsänderungen $\Delta R/R$ im Bereich von 30 % bis zu 200 % und sind somit deutlich höher als beim GMR [21]. Unter Laborbedingungen konnten bei sehr tiefen Temperaturen mittlerweile Widerstandsänderungen bis zu 1000 % erreicht werden [10].

Praxistaugliche Ausführungen von TMR-Sensoren stehen erst seit einigen Jahren zur Verfügung. Die Herstellung eines Sensors erfordert einen enormen apparativen Aufwand und Produktionsanlagen mit entsprechenden Fertigkeiten mussten erst entwickelt werden. Der Aufwand ist mit Vorteilen gegenüber dem AMR- und GMR-Sensor entlohnt worden. Somit besitzt ein TMR-Sensorelement einen viel höheren Widerstand bei gleicher Abmessung. AMR/ GMR Technologien müssen im Vergleich flächenhungrige Strukturen realisieren. Aufgrund der vergleichsweise kleinen Flächen und einer äußerst geringeren Stromaufnahme ist ein engmaschiger Aufbau von Array-Strukturen möglich [10]. TMR-Flächen sind typischer Weise $< 2 \mu\text{m}$ im Radius. Die hohe Widerstandsänderung generiert entsprechend hohe Signalamplituden in der Magnetfelderfassung, daher kann eine nachgeschaltete Verstärkung entfallen. Es ist eine Periodizität von 360° abgedeckt. Ein TMR-Sensor für die Drehwinkelerfassung besteht aus zwei um 90° verdrehte Wheatstone-Brücken [11].

Wheatstone-Brücke

Ein einzelnes TMR-Element bzw. Widerstand kann bereits als eigenständiger magnetischer Sensor betrachtet werden. Allerdings können Temperatureinflüsse den Widerstand mitunter variieren lassen. Um dem Temperatureinfluss entgegenzuwirken, werden daher Wheatstone'sche Brückenschaltungen genutzt. Die einzelnen Widerstände der Brücke sind dabei so angeordnet, dass sie einen gemeinsamen Temperaturkoeffizienten besitzen und entsprechend miteinander gleichmäßig schwanken [21]. Über die Differenzmessung an den Brückenmittelabgriffen wird somit der Temperatureinfluss weitestgehend unterdrückt [11][21]. Abbildung 2.5 zeigt den schematischen Brückenaufbau für einen TMR-Sensor. Bei gleichförmiger Rotation eines Anregungsmagnetfeld wird durch die 90° -Verdrehung beider Brücken zueinander erreicht, dass die benötigte Cosinus- und Sinus-Funktion, mit entsprechender Phasenverschiebung um 90° , für den Anwendungsfall in Abschnitt 2.1 ausgeben werden [11].

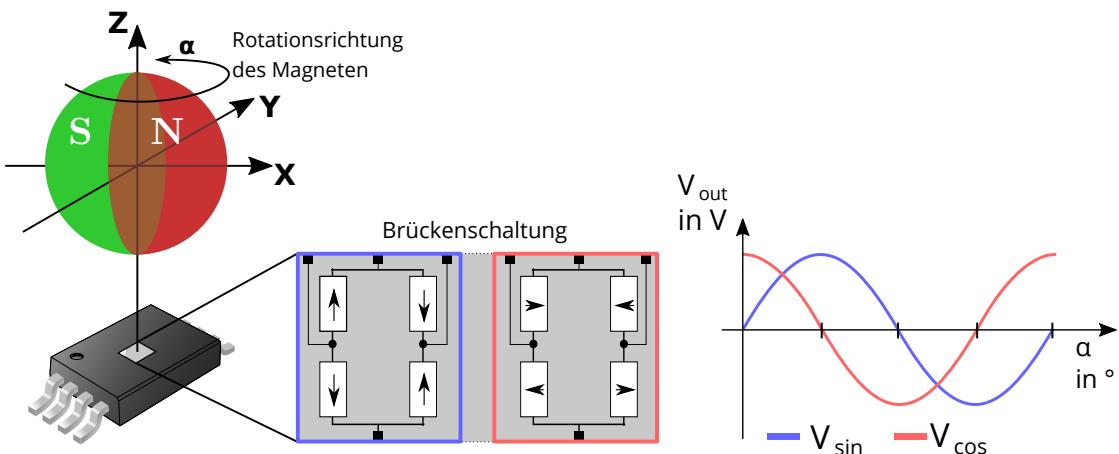


Abbildung 2.5: TMR Drehwinkelapplikation. Schematisch gezeigt für eine volle Rotation des Gebermagneten um 360° . Zu sehen sind die um 90° verdrehten Wheatstone-Brücken des Sensors. Die Brücken bilden, bei rotierendem Gebermagnetfeld, eine Sinus- und Cosinus-Funktion nach. Die Pfeile in den einzelnen Widerständen weisen auf ihre magnetische Ausrichtung hin. Grafik entnommen und bearbeitet aus [20].

2.4 Kennfeldmethode zur Charakterisierung von Sensoren

Die physikalisch-mathematische Beschreibung von magnetischen Sensormodellen für eine simulative Nutzung ist nicht trivial. Jede in Abschnitt 2.3 zusammengefasste Sensortechnologie birgt bestimmte verhaltensbezogene Eigenschaften in sich. So müssen technologiebasierte Abhängigkeiten in Linearität und Sättigungsverhalten in komplexen mathematischen Gleichungen beschrieben sein, wobei bestimmte Parameter der Modelle experimentell bestimmt werden müssen [15]. Das stellt einen erheblichen Arbeitsaufwand dar. Der Arbeitsgruppe Sensorik ist es gelungen, diesen Aufwand durch Messmethodik zu umgehen. So ist die Kennfeldmethode zur Charakterisierung von magnetoresistiven Sensoren entwickelt worden. Das Ziel der Messmethode ist es, repräsentative Datensätze zu generieren, die physikalische Charakteristiken eines Sensors in sich vereinigen und sein Verhalten zur weiteren Nutzung zugänglich machen.

Im Labor der Arbeitsgruppe Sensorik sind automatisierte Messstände für Sensoren verschiedenster Technologien eingerichtet. Je nach Sensortechnologie und Applikation können diese mit unterschiedlichen Messmitteln bestückt werden. So ist es möglich, die richtige Stimulanz, für die jeweilige Sensorapplikation zu erzeugen. Für das Ausmessen von Winkelsensoren ist ein Kreuzspulen-Messstand zur Anwendung gekommen [15][19]. Der Messstand eignet sich besonders gut dazu, rotierende Magnetfelder zu erzeugen, was einer idealen Stimulanz für Winkelsensoren entspricht. Das Magnetfeld wird dabei durch ein speziell abgestimmtes Kreuzspulen-System erzeugt. Die dafür eingespeisten Spulenströme sind dabei direkt proportional zum erzeugten Magnetfeld. Das Anregungsfeld kann somit über Spulenfaktoren zurückgerechnet werden [19].

Abbildung 2.6 zeigt das verwendete Anregungsfeld zur Charakterisierung. Stimuliert wird in X - und Y -Richtung der räumlichen Sensorebene. So verwendet die H_x -Feldgenerierung ein dreieckmodulierter Cosinus-Strom. Die H_y -Feldgenerierung nutzt einen mit gleicher Frequenz dreieckmodulierten Sinus-Strom. Es entstehen dabei steigende und fallende Modulationsflanken bzw. Messverläufe. Es ergeben sich in polarer Darstellung Trajektorien mit wachsender und abnehmenden Amplituden, die durch Rotation einen nach außen (steigend) bzw. innen (fallend) gerichteten Verlauf besitzen [15]. Es werden die Spulenströme und Spannungsausgaben des Sensors aufgezeichnet.

In einem weiteren Evaluierungsschritt sind Stimuli und Sensorsignale programmatisch zu indizieren, um eine gegenseitig Referenzierung zu ermöglichen, sodass resultierend zweidimensionale Kennfeldpaare, bestehend aus je einem Kennfeld für eine entsprechende Winkelsensor-Wheatstone-Brücke, als Charakterisierungsergebnis zur Verfügung stehen [15].

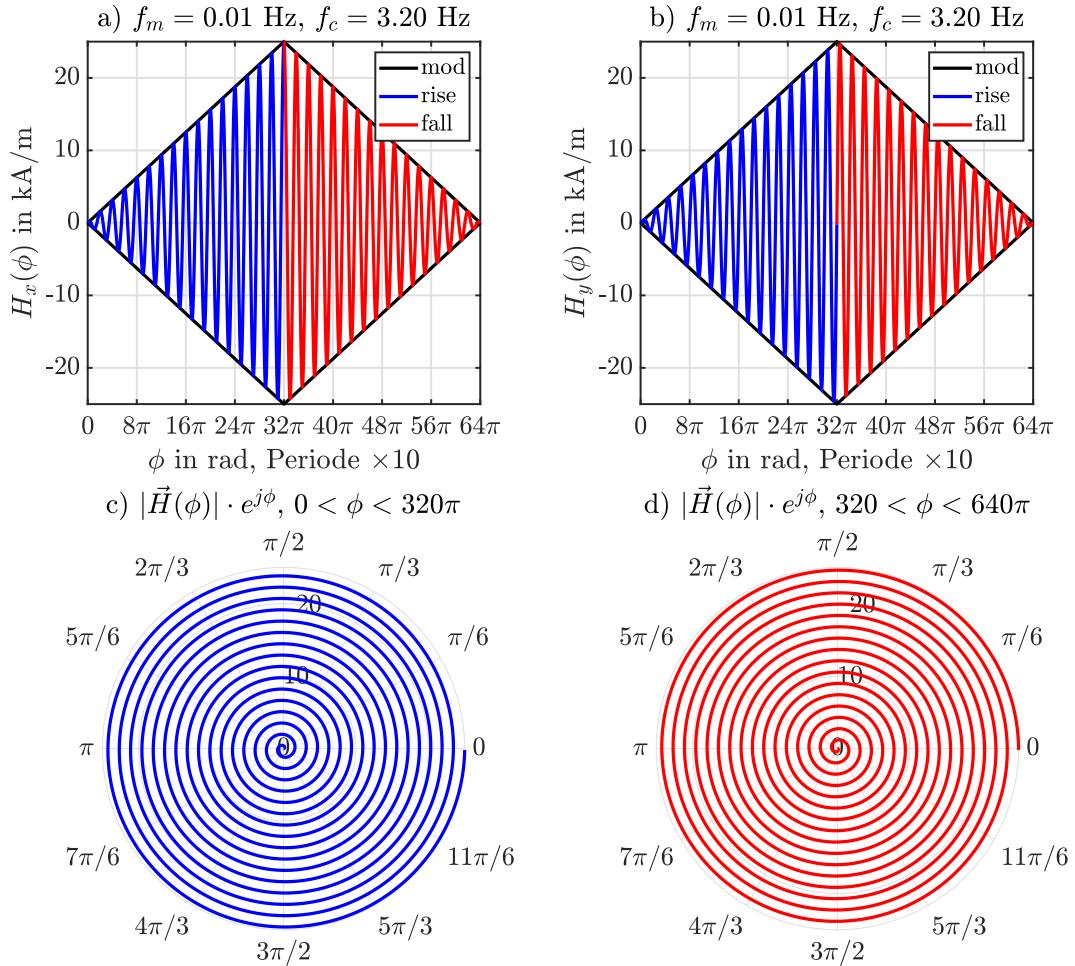


Abbildung 2.6: Magnetfeldstimuli zur Erzeugung von Sensorkennfeldern. Es sind die Bestandteile des magnetischen Sensorstimuli dargestellt, die zum Ausmessen des Sensorkennfeldes in H_x - und H_y -Richtung verwendet worden sind. Es ist das Prinzip des Verfahrens dargestellt. In a) und b) ist die Dreiecksmodulation des magnetischen Anregungsfeldes abgebildet. Für a) die H_x -Feldanregung mit Cosinus-Trägerwelle und für b) die H_y -Feldanregung mit Sinus-Trägerwelle. Es sind für beide Anregungsrichtungen niedrige Frequenzen gewählt um ein quasi-statisches Anregungsmagnetfeld zu erzeugen. Es ergeben sich für die Betragsamplitude des Stimulus, in polarer Darstellung c) und d), konzentrische Trajektorien. Diese verlaufen von innen nach außen für die steigende Flanke der Amplitudenmodulation c) und von außen nach innen für die fallende Flanke d). Die Dreieckmodulationsfrequenz liegt bei $f_m = 0,1$ Hz und einer Trägerwellenfrequenz $f_c = 3,2$ Hz. Grafik nachempfunden aus [15].

Im Anhang A ist der Kennfelddatensatz eines TMR-Sensors [11] gezeigt. Der Datensatz dient, als Arbeitsgrundlage für die Sensor-Array-Simulation und ist von der Arbeitsgruppe Sensorik zur Verfügung gestellt worden. Zur Simulation sind die Kennfelder aus Abbildung 2.7 zu verwenden, a) für die Erzeugung der Cosinus-Funktion und b) für die Sinus-Funktion. Beide Kennfelder zeichnen sich besonders durch ihren linearen Arbeitsbereich zwischen $\pm 8,5 \text{ kA m}^{-1}$ aus. Der Arbeitsbereich ist für beide Kennfelder nahezu identisch, zu sehen in c) und als Kreis in a) und b) markiert.

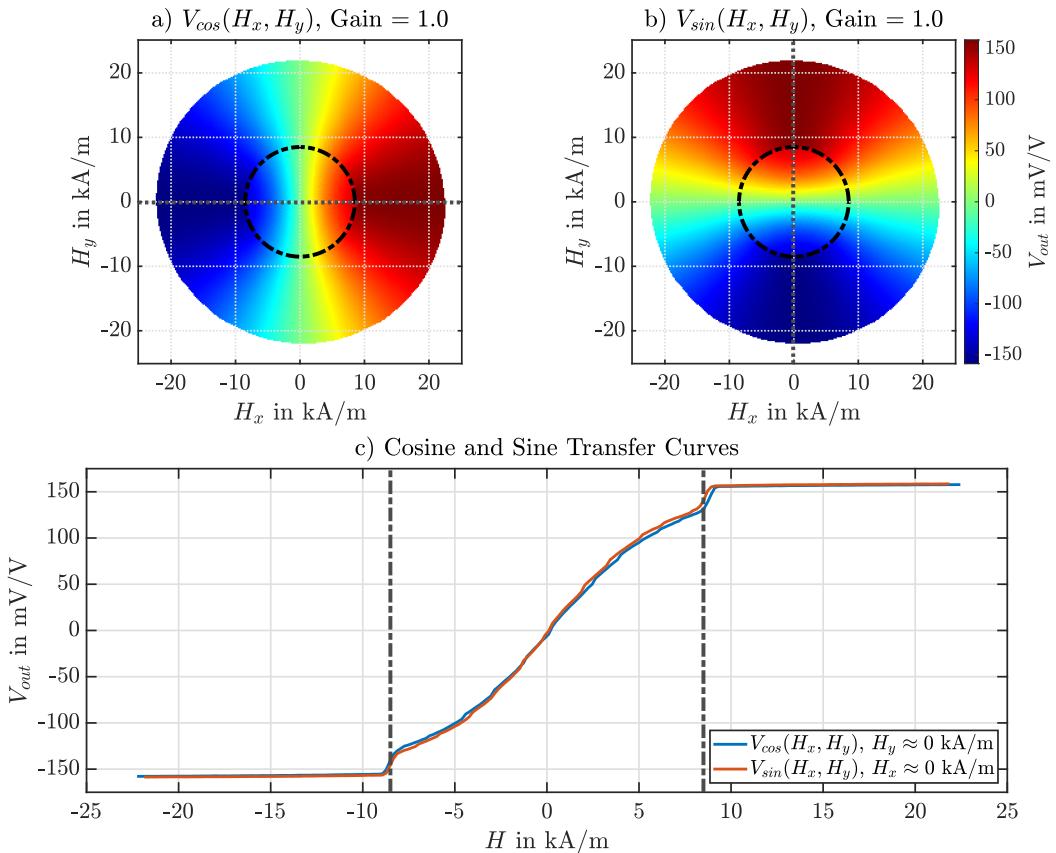


Abbildung 2.7: TDK TAS2141-AAAB Übertragungskennlinie. Es sind wieder die Kennfelder aus der steigenden Amplitudenmodulation in a) und b). In c) sind die Übertragungskennlinien für den Sensor gezeigt mit Kennzeichnung für den Betrieb auf dem linearen Plateau des Kennfeldes bei $8,5 \text{ kA m}^{-1}$. Ebenfalls zu sehen in a) und b) durch die sich ergebende Kreisbahn mit einem Radius des aufgelegten Intervalls aus c). Grafik nachempfunden aus [15].

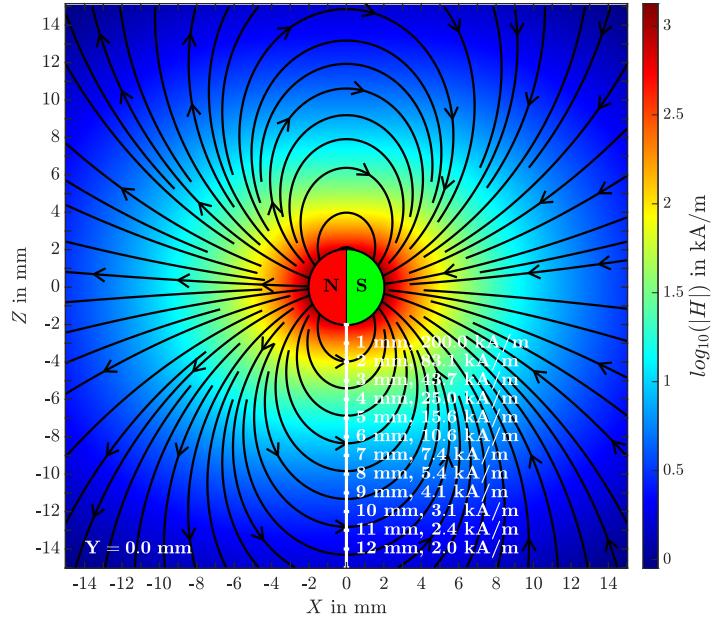


Abbildung 2.8: Approximierter Kugelmagnet. Die Approximation des Kugelmagneten erfolgt über die Dipol-Feldgleichung in Näherung des Kugelmagnetfernfeldes. Das Magnetfeld ist auf 200 kA m^{-1} bei einem Abstand von 1 mm zur Kugelmagnetoberfläche normiert. Der Radius des Kugelmagneten beträgt 2 mm.

Auf den TMR-Sensor-Kennfeldern basierende Simulationen, sollten daher so parametriert sein, dass sie innerhalb des markierten Bereiches stattfinden. Zur Generierung von Spannungsausgaben in einer Sensorsimulation, sind für korrespondierende Anregungsfeldstärken entsprechende Referenzwerte aus den Kennfeldern zu entnehmen. Die Referenzwerte sind normiert und nach Gleichung A.1 in Spannungsausgaben der Wheatstone-Brückenschaltungen umzurechnen. Ein geeignetes Verfahren für die Entnahme von Referenzwerten bietet hier die in Matlab implementierte 2D-Interpolation. Diese kann so parametriert werden, dass sie nach dem Nearest-Neighbor-Verfahren für beliebige Feldstärkeneingaben den nächstgelegenen Referenzwert ausgibt. Wichtig sind hierbei eine gute magnetische Stimulanz, die den Arbeitsbereich des Kennfeldes trifft.

Als Beispiel für eine passende Anregung soll hier, der in Abbildung 2.8 approximierte, Kugelmagnet dienen. Das Magnetfeld ist so normiert, dass eine Betragsfeldstärke von 200 kA m^{-1} bei 1 mm Abstand zur Magnetenoberfläche anliegt. Die angelegte Skala zeigt, dass ein Sensor mit seinen Kennfeldern aus Abbildung 2.7, einen Abstand in Z -Richtung größer als 6 mm einhalten sollte, um den Arbeitsbereich des Kennfeldes zu treffen. Weitere Beschreibungen und Erläuterungen zur Simulation und Dimensionierung der magnetischen Feldanregung finden sich in beiden folgenden Unterkapiteln Abschnitt 2.5 und Abschnitt 2.6.

2.5 Prinzip des Sensor-Arrays

Ein Sensor-Array stellt in seiner Funktionsweise ein Array aus einzelnen Winkelsensoren dar. Jeder einzelne Winkelsensor des Arrays bildet somit einen Sensor-Pixel. Die einzelnen Sensor-Pixel besitzen im Simulationsbetrieb dasselbe Verhalten, basierend auf den TMR-Senor [11], aus Kennfeldern (Anhang A) entnommen wird. Resultierende Messwerte der einzelnen Sensor-Pixel sind positionsabhängig von ihrer Lage im Sensor-Array[15]. Die Sensor-Pixel-Anordnung erfolgt quadratisch mit gleichen Abständen zueinander. Das Gesamtsystem ist somit eine Adaption des klassischen Anwendungsfall aus Abbildung 2.1 für multiple Winkelsensoren [14][15]. So ergibt sich der Anwendungsfall für das Sensor-Array nach Abbildung 2.9.

Das Gesamtsystem aus Gebermagnet und Sensor-Array behält seinen Koordinatenursprung in der Gebermagnetmitte. Das Sensor-Array ist zentriert und lotrecht zur Z -Achse des Magneten auszurichten [15], sodass ein konstantes Flächenniveau in Z eingehalten wird. Bedingt durch die aufgespannte Array-Fläche, sind die einzelnen Sensor-Pixel nicht mehr ideal unter dem Magneten ausgerichtet [18]. Es ist somit davon auszugehen, dass die Kreisbahnen der einzelnen Pixel verzerrt sind, wobei sich die Bahnbeschreibungen, der Pixel mit dem geringsten X -/ Y -Versatz, einem idealen Kreisverlauf annähern müssen. Physikalische Kleinstabstände zwischen den Sensorbrücken eines Sensor-Pixels sind vernachlässigt. Im Simulationsbetrieb ist die Annahme getroffen, dass die Brückenabstände innerhalb eines Sensor-Pixel vernachlässigbar klein sind.

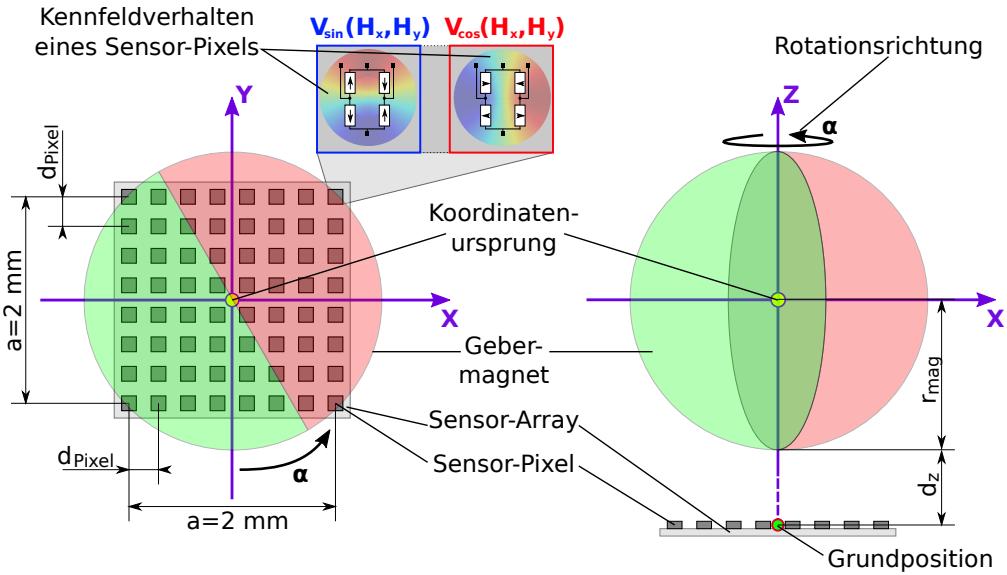


Abbildung 2.9: Geometrischer Aufbau und Ausrichtung des Sensor-Arrays. Quadratische Anordnung von Sensor-Pixeln zu einem 8×8 Sensor-Array. Alle Pixel sind gleich verteilt auf der Array-Fläche. Sensor-Pixel-Verhalten ist aus Kennfeldern entnommen und ortsabhängig von Pixel-Position im Koordinatensystem. Array-Kantenlängen sind mittig von Eck-Pixel zu Eck-Pixel bestimmt. Ebenfalls der Z -Abstand zur Magnetoberfläche. Abstände innerhalb der Pixel sind vernachlässigt. Das Array ist zentriert in der Z -Achse ausgerichtet. Ideal lotrecht zur Nord-Süd-Ausrichtung des Magneten. Koordinatenursprung des Gesamtsystems liegt in der Gebermagnetmitte. Gebermagnet ist ein Kugelmagnet, der um seine Z -Achse rotiert. Grafik nachempfunden und bearbeitet aus [18].

Gemäß der geometrischen Vorgaben, konstantem Flächenniveau in Z und der Wahl des Koordinatenursprungs, fächer sich ein Koordinaten-Meshgrid für $N_{Pixel} \times N_{Pixel}$ für $i, j \in \{1 \dots N_{Pixel}\}$ Sensor-Pixel auf. Dabei ist von einer, im Mittelpunkt des Sensor-Array festgelegten, Grundposition \vec{p} des Sensor-Arrays nach Gleichung 2.9 vorzugehen.

$$\begin{aligned}
 \vec{p} &= \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} & A_{Array} &= a_{Array}^2 & x_{i,j} &= p_x - \frac{a_{Array}}{2} + j \cdot d_{Pixel} \\
 d_{Pixel} &= \frac{a_{Array}}{N_{Pixel} - 1} & y_{i,j} &= p_y + \frac{a_{Array}}{2} - i \cdot d_{Pixel} & & (2.9) \\
 z_{i,j} &= p_z - r_{mag} = konst.
 \end{aligned}$$

Die Z -Koordinate entspricht dem Z -Abstand zum Magneten mit $d_z = p_z$. Das Sensor-Array ist über diesen Vektor im Koordinatensystem zu verschieben. Der Koordinatenursprung bleibt im Magneten. Das Meshgrid fächert sich für die $i - te$ Reihe und die $j - te$ Spalte des Sensor-Arrays auf. Somit gibt jedes Sensor-Pixel, entsprechend seiner Zuordnung im Array, Spannungsausgaben wie ein einzelnes Sensor-IC aus. Es stehen dadurch nicht mehr Skalare bzw. ein Vektor als Winkelmesswert zur Verfügung, sondern Array-Daten für korrespondierende Cosinus- und Sinus-Vektorfelder [14][19]. Abbildung 2.10 zeigt den dimensionalen Zuwachs.



Abbildung 2.10: Resultierende Sensor-Array-Daten. Zwei Matrizen, je eine für alle Cosinus-Brückenausgaben und Sinus-Brückenausgaben. Die $i - ten$ und $j - ten$ Matrixelemente bilden einen Vektor entsprechend des klassischen Anwendungsbeispiels. Jeweils ein Matrix-Paar für die $n - te$ Winkelstellung α .

Bedingt durch den Zuwachs an Daten und ihre Anordnung, benötigt es eine modifizierte Abstandsfunktion [18][19] im Vergleich zur Gleichung 2.7. Als erstes braucht es skalare Repräsentanten der Matrizen. Diese sind durch eine zutreffende Matrix-Norm zu bilden. Matrizen können als lange Vektoren betrachtet werden, daher bietet sich die Rechenvorschrift für $j - te$ Spalten nach Gleichung 2.10 an. Diese Norm ist als Frobenius-Norm bezeichnet.

$$\|\mathbf{A}_x\|_F = \sqrt{\sum_{j=1}^n \|A_{xj}\|_2^2} = \sqrt{\mathbf{A}_x \mathbf{A}_x^T} \quad (2.10)$$

Angewandt auf beide Vektormatrizen für Cosinus- und Sinus-Anteile, ergibt sich eine normierte Kreisbahn nach Gleichung 2.11. Der Radius ist durch Versatz der einzelnen Sensor-Pixel nicht konstant und muss je nach Position des Sensor-Arrays eine weniger oder stärkere Ellipsenform beschreiben [15]. Das ergibt sich durch Überlagerung der Sinoide nach Frobenius-Norm. Der Versatz jedes einzelnen Sensor-Pixel wirkt dabei wie eine Dämpfung auf die Ausgangsspannungen der Sensor-Pixel. So zeigt sich ein Versatz in X-Richtung in einer Dämpfung der Cosinus-Funktion und entsprechender Versatz in Y-Richtung mit einer Dämpfung der Sinus-Funktion [15].

$$\|r\|_F = \|\mathbf{A}\|_F = \sqrt{\|\mathbf{A}_x\|_F^2 + \|\mathbf{A}_y\|_F^2} \quad (2.11)$$

Durch simples einsetzen der normierten Messwertmatrizen in die euklidische Abstandsquadratfunktion Gleichung 2.7, folgt ein approximiertes Abstandsergebnis nach Gleichung 2.12. Über die Dreiecksungleichung erhält man genau die Lösung und Projektion in den höheren Normraum [4][9] und somit die modifizierte Abstandsfunktion nach der Frobenius-Norm [19][18] in Gleichung 2.13. Es sind dargestellte Beschreibungen aus Abschnitt 2.2, für zwei Winkelstellungen \mathbf{A} und \mathbf{B} , entsprechend der Array-Datenformate angepasst.

$$d_E^2 \langle \mathbf{A}, \mathbf{B} \rangle = (\|\mathbf{A}_x\|_F - \|\mathbf{B}_x\|_F)^2 + (\|\mathbf{A}_y\|_F - \|\mathbf{B}_y\|_F)^2 \quad (2.12)$$

$$\leq$$

$$d_F^2 \langle \mathbf{A}, \mathbf{B} \rangle = \|\mathbf{A}_x - \mathbf{B}_x\|_F^2 + \|\mathbf{A}_y - \mathbf{B}_y\|_F^2 = \|\mathbf{A} - \mathbf{B}\|_F^2 \quad (2.13)$$

2.6 Sensor-Array-Simulation über die Dipol-Feldgleichung

Die Sensor-Array-Simulation nutzt einen Kugelmagneten als Stimulanz [15]. Es ist die Anwendungsbeschreibung aus Abbildung 2.9 gewählt. Der Vorteil darin liegt, dass ein Kugelmagnetfeld mittels der Feldgleichung für einen magnetischen Dipol approximiert werden kann [12]. Das innere Magnetfeld des Kugelmagneten ist dabei zu vernachlässigen. Der Radius des Kugelmagneten r_{mag} ist als Offset für den räumlichen Abstand zum Magneten zu verwenden. Weitere physikalische Effekte wie magnetische Remanenz werden vernachlässigt.

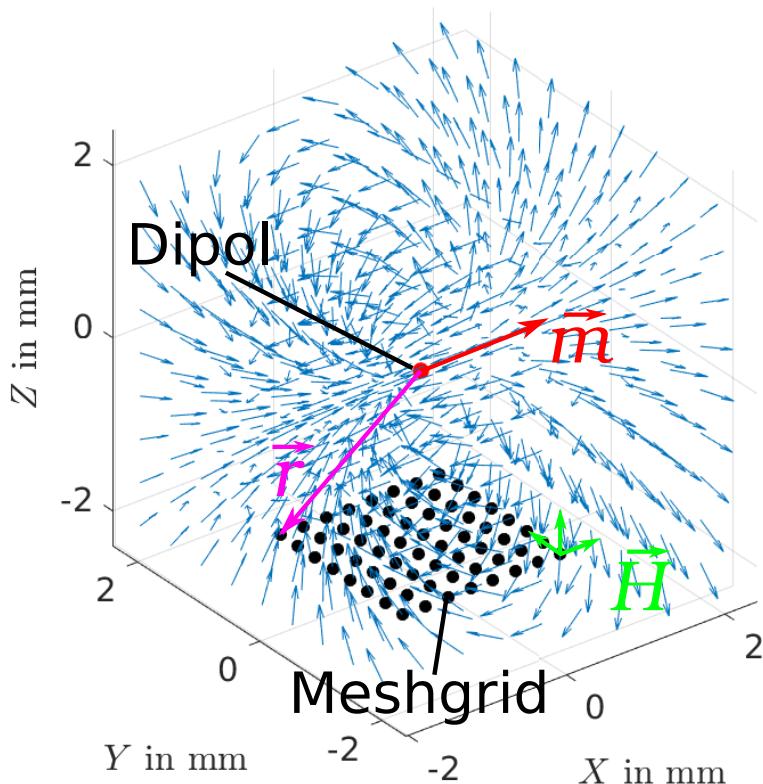


Abbildung 2.11: Simulation der Dipol-Feldgleichung. Als veranschaulichendes Beispiel ist ein Magnetfeld über die Dipol-Feldgleichung simuliert. Der Dipol bildet den Koordinatenursprung bei $\vec{r} = (0, 0, 0)^T$. In Relation zum Dipol ist ein Meshgrid der einzelnen Sensor-Pixel-Positionen \vec{r} unterhalb des Dipoles gelegt. Abhängig vom magnetischen Moment \vec{m} des Dipoles, wird an jeder Meshgrid-Position \vec{r} die Dipol-Feldgleichung gelöst und punktuell die magnetische Feldstärke \vec{H} berechnet. Das magnetische Moment \vec{m} bestimmt die Nord-Süd-Ausrichtung und Winkelstellungen des Dipoles.

Es wird ein Meshgrid für die einzelnen Sensor-Pixel nach Gleichung 2.9 in ein dreidimensionales Koordinatensystem gelegt. Wie in Abbildung 2.11 zu sehen, liegt der Dipol im Koordinatenursprung bei $\vec{r} = (0, 0, 0)^T$. Die einzelne Pixel-Position \vec{r} und das magnetische Dipol-Moment \vec{m} , sind durch Gleichung 2.14 beschrieben. Das Dipol-Moment \vec{m} bestimmt die räumliche Ausrichtung des Magnetfeldes.

$$\vec{r} = \hat{r} \cdot |\vec{r}| = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \vec{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} \quad (2.14)$$

Die Feldstärken \vec{H} an den jeweiligen Pixel-Positionen \vec{r} sind für das aktuelle Moment \vec{m} nach Gleichung 2.15 zu berechnen. Gleichung 2.15 ist durch Einsetzen von Gleichung 2.14 zu Gleichung 2.16 vereinfacht. Somit ist die Feldstärke \vec{H} mit dazugehörigem Moment \vec{m} nur von der Richtung des Einheitsvektors \hat{r} abhängig und durch $\frac{1}{4\pi|\vec{r}|^3}$ skaliert. Die entsprechende Betragfeldstärke $|\vec{H}|$ setzt sich aus den resultierenden Feldstärkenkomponenten in Gleichung 2.17 zusammen.

$$\vec{H}(\vec{r}, \vec{m}) = \frac{1}{4\pi} \cdot \left(\frac{3\vec{r} \cdot (\vec{m}^T \cdot \vec{r})}{|\vec{r}|^5} - \frac{\vec{m}}{|\vec{r}|^3} \right) \quad (2.15)$$

$$= \frac{1}{4\pi|\vec{r}|^3} \cdot \left(3\hat{r} \cdot (\vec{m}^T \cdot \hat{r}) - \vec{m} \right) = \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix} \quad (2.16)$$

$$|\vec{H}| = \sqrt{H_x^2 + H_y^2 + H_z^2} \quad (2.17)$$

Um eine Rotation und etwaige Verkippungen des Dipol-Magnetfeldes im Raum zu erwirken, müssen nach Gleichung 2.18 entsprechende axiale Rotationen in X , Y und Z , durch Aufschalten von Drehmatrizen hergestellt werden. Durch drehen bzw. verkippen des Dipol-Momentes \vec{m} nach Gleichung 2.18, ergibt sich das neue Dipol-Moment \vec{m}' und somit weiter resultierende Feldstärken \vec{H}' bei gleichbleibenden Pixel-Positionen \vec{r} [15].

$$\underbrace{\begin{pmatrix} m'_x \\ m'_y \\ m'_z \end{pmatrix}}_{\vec{m}'} = \underbrace{\begin{pmatrix} \cos \alpha_z & -\sin \alpha_z & 0 \\ \sin \alpha_z & \cos \alpha_z & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R_z(\alpha_z)} \underbrace{\begin{pmatrix} \cos \alpha_y & 0 & \sin \alpha_y \\ 0 & 1 & 0 \\ -\sin \alpha_y & 0 & \cos \alpha_y \end{pmatrix}}_{R_y(\alpha_y)} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha_x & -\sin \alpha_x \\ 0 & \sin \alpha_x & \cos \alpha_x \end{pmatrix}}_{R_x(\alpha_x)} \underbrace{\begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}}_{\vec{m}} \quad (2.18)$$

Für den Standardanwendungsfall aus Abschnitt 2.5, also Rotation in Magnet- Z -Achse ohne Verkippung, sind die Drehmatrizen R_x und R_y auszuschalten. Das kann durch Nullsetzen der Verkippungswinkel α_x und α_y erreicht werden. Die Drehmatrizen sind dadurch zur Einheitsmatrix I gleichgeschaltet. Es ergeben sich somit $i - te$ Dipol-Rotationsmomente \vec{m}_i , für $i - te$ Winkelstellungen des Magneten α_i mit $\alpha_i \in \{0^\circ, \dots, 360^\circ\}$, nach Gleichung 2.19. Dabei ist $\vec{m}_0 = -(m_0, 0, 0)^T$ das Startmoment und legt die Nord-Süd-Ausrichtung des Gebermagneten zu Beginn in seine X -Achse. Für Rotationen mit konstanten Verkippungen sind die Verkippungswinkel $\alpha_x \neq 0$ bzw. $\alpha_y \neq 0$ zu setzen und allgemein nach Gleichung 2.18 zu berechnen.

$$\vec{m}_i(\alpha_i) = R_z(\alpha_i) \cdot I \cdot I \cdot \vec{m}_0 \quad \text{f. } \vec{m}_0 = - \begin{pmatrix} m_0 \\ 0 \\ 0 \end{pmatrix} \quad (2.19)$$

Als Anfangswert für das Startmoment empfiehlt sich $m_0 > 1000 \text{ Am}^2$ zu wählen, dass unterdrückt numerische Fehler beim Berechnen der Feldstärke \vec{H} . In einem weiteren Normierungsschritt zum Aufprägen einer Betragsfeldstärke H_{mag} , bei definierten Abstand $r_{mag} + d_z$ zur Magnetenoberfläche, löscht sich der hohe Anfangswert für m_0 rechnerisch aus, sodass über das Dipol-Moment nur die Ausrichtung des Gebermagneten gesteuert ist und kein nominaler Einfluss auf errechnete Feldstärken \vec{H} besteht.

Damit in der Sensor-Array-Simulation magnetische Anregungen erzeugt werden können, die den empfohlenen Kennfeldarbeitsbereich aus Anhang A treffen, ist es notwendig, das approximierte Kugelmagnetfeld im Weiteren zu manipulieren. Die Manipulation des Magnetfeldes erfolgt durch das Aufprägen einer Betragsfeldstärke H_{mag} für die Ruhelage des Magneten mit dazugehöriger Feldstärke \vec{H}_0 . Dabei ist ein definierter Abstand zur Kugelmagnetoberfläche festzulegen, bei dem sich die aufzuprägende Betragsfeldstärke H_{mag} einstellt.

$$\vec{r}_0(\alpha_1, \alpha_y, \alpha_x) = R_z(\alpha_1) \cdot R_y(\alpha_y) \cdot R_x(\alpha_x) \cdot (0, 0, -(r_{mag} + d_z))^T \quad (2.20)$$

$$\vec{m}_0(\alpha_1, \alpha_y, \alpha_x) = R_z(\alpha_1) \cdot R_y(\alpha_y) \cdot R_x(\alpha_x) \cdot (-m_0, 0, 0)^T \quad (2.21)$$

Die Ruhelage bezieht sich auf den Startwinkel α_1 der Simulation und ist gemäß gewünschter Verkippungen axial getreu einzustellen. Die Normierungsposition ist mit Gleichung 2.20 vorgegeben und definiert den Abstand entlang der Magnet-Z-Achse und zur Magnetoberfläche. Der Kugelmagnet ist mit entsprechendem Ruhemoment, der Normierungsposition folgend, nach Gleichung 2.21 auszurichten. Anschließend ist die Betragsfeldstärke $|\vec{H}_0(\vec{r}_0, \vec{m}_0)|$ auszurechnen. Über den Quotient aus gewünschter Prägung H_{mag} und Betrag $|\vec{H}_0(\vec{r}_0, \vec{m}_0)|$ in Ruhelage mündet die Berechnung für ein normiertes Kugelmagnetfeld $\vec{H}_{Norm}(\vec{r}, \vec{m}_i)$, für beliebige Positionen \vec{r} im Koordinatenraum und $i - te$ Rotationsmomente \vec{m}_i in Gleichung 2.22.

$$\vec{H}_{Norm}(\vec{r}, \vec{m}_i) = \vec{H}(\vec{r}, \vec{m}_i) \cdot \frac{H_{mag}}{|\vec{H}_0(\vec{r}_0, \vec{m}_0)|} \quad (2.22)$$

Die Anwendungskonfigurierung für eine optimale Simulation und Treffen der Arbeitsbereiche ist Anhang B zu entnehmen. Simulierte Feldstärken sind gemäß der Meshgrid-Anordnung in Matrizen zu speichern, sodass sich die in Abschnitt 2.5 beschriebenen Array-Datenformate ergeben. Diese können im Simulationsverlauf fortführend, direkt im Array-Format auf die Kennfelder, zur Entnahme von korrespondierenden Spannungsangaben angewandt und gespeichert werden. In der Sensor-Array-Simulation ist die Verkippung in der X-Achse deaktiviert mit $\alpha_x = 0$. Im weiteren Kontext bezieht sich der Begriff Verkippung (engl. “tilt”), ausschließlich auf Verkippungen in der Y-Achse des Gebermagneten.

2.7 Gauß-Prozesse für Regressionsverfahren

Das in der Arbeit zur Anwendung kommende Regressionsverfahren für Gauß'sche Prozesse orientiert sich maßgebend an der 2006 von Rasmussen und Williams veröffentlichter Leitliteratur [3]. Vorarbeiten der Arbeitsgruppe Sensorik [18][19] basieren dabei auf Winkelvorhersagen, die über dem mittelwertfreien Ansatz gewonnen werden [3]. Als funktionaler Entwurf des Regressionsmodells [19] decken die Vorarbeiten Modellinitialisierung Abschnitt C.1 und Vorhersage mit Winkelkonfidenzintervall Abschnitt C.3 ab. Dabei bezieht sich die Modellinitialisierung auf die Kernel-Implementierung mittels Kovarianzfunktion nach Gleichung C.4 und für die Abstandsfunktion d_F^2 nach Gleichung 2.13. Das aufgestellte Regressionsmodell ist in der Lage Simulationsergebnisse aus Anhang B zu verarbeiten [18][19].

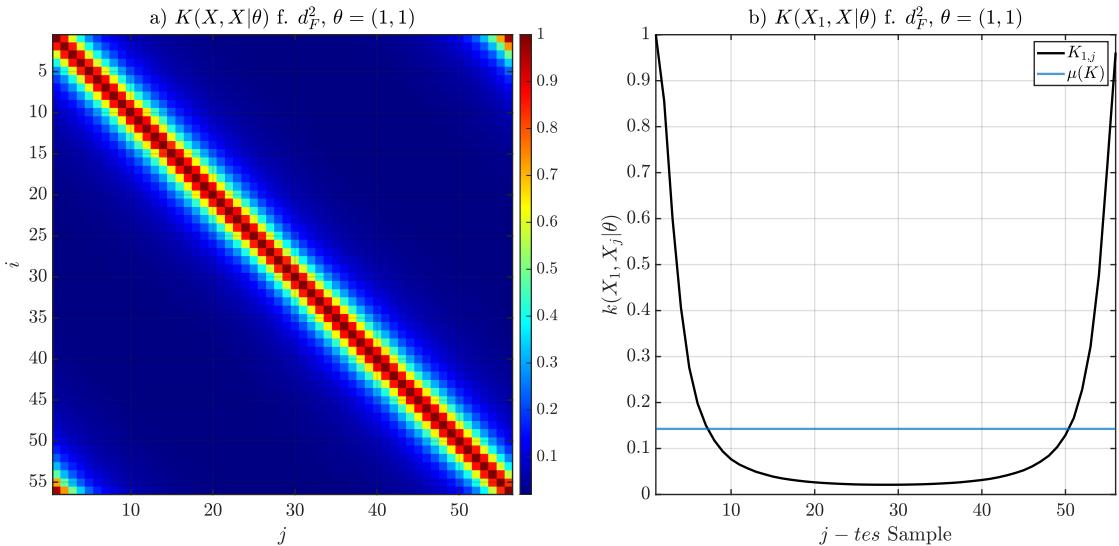


Abbildung 2.12: Kernel-Implementierung der Vorarbeiten mittels Kovarianzfunktion $k(X_i, X_j)$ nach Gleichung C.4 f. d_F^2 nach Gleichung 2.13 und einem Trainingsdatensatz X mit $N_{Ref} = 56$ gleichverteilten Simulationswinkeln $X_i \mapsto \alpha_i$. Die Skalierung durch Kernel-Parameter ist mit $\theta = (1, 1)$ ausgeschaltet. In a) ist Algorithmus 3 f. $K(X, X|\theta)$ genutzt und alle Trainingsdaten X somit gegenseitig referenziert. Abbildung b) zeigt Kovarianzen $K_{1,j}$ des Teildatensatz X_1 gegenüber sich selbst und jedem weiteren Satz X_j . Der Matrix-Mittelwert $\mu(K)$ in b), ist als Indikator über die gegenseitige Beeinflussung von Datensätzen im Regressionsverfahren zu interpretieren. Der genutzte Trainingsdatensatz X basiert auf erfolgter TMR-Sensor-Charakterisierung in Anhang A [11][15]. Grafik nachempfunden aus [7]

Ab hier folgt ein Wechsel der Notation, um Bezüge zur Fachliteratur [3] zeigen zu können. Die veränderte Schreibweise ist im Abschnitt C.1 unter Trainings- bzw. Testdatensätze zusammengefasst und beschrieben.

Die Parametrierung des Regressionsverfahren ist bisher empirisch ermittelt worden und stellt einen Angelpunkt in dieser Arbeit dar. Für eine selbstständige Optimierung, von Modellparametern und verbundener Modellgeneralisierung müssen entsprechende Kriterien gebildet werden. Diese sind in verwandter Literatur als Minimierungsprobleme beschrieben [3][5][7].

Im Gegensatz zur Leitliteratur [3], die Lösungsansätze für Regressionen eindimensionaler Funktionen beschreibt, ist für die Winkelvorhersage eine kombinierte Regression einer zweidimensionalen Funktion herzustellen. Ableitungen erfolgen für Winkelstellungen über orthogonal zueinander stehenden Cosinus- und Sinus-Funktionen. Dabei ist die Adaption der Array-Daten-Formate aus Abschnitt 2.5, bereits in den Vorarbeiten gelöst worden [19] und über die Kovarianzfunktion Gleichung C.4 für d_F^2 Gleichung 2.13 implementiert. Abbildung 2.12 zeigt die Implementierung aus den Vorarbeiten anhand der Kovarianzfunktion und resultierender Kovarianzmatrix für ein Beispiel eines $N_{Ref} = 56$ Observierungen großen Trainingsdatensatz. Für diese Implementierungsform wäre das ein viel zu großer Datensatz in der realen Anwendung. In Bezug auf das Sensor-Array Abschnitt 2.5 müssten $2 \times 56 = 112$ Matrizen abgelegt werden, sodass diese dem Regressionsprozess als Referenzen zur Verfügung stehen. Was hier zwar ein drastisches Beispiel ist, das allerdings sehr schön das Verhalten der Kovarianz in Verbindung mit TMR basierten Daten zeigt.

Die Kovarianzfunktion mit resultierender Kovarianzmatrix muss in der Lage sein systemische Eigenschaften wiedergeben zu können. Auf den TMR-Sensor [11] bezogen, muss die Matrix einfach periodisch sein. Das ist durch Kurvenverlauf in Abbildung 2.12 b) und durch ansteigenden Ecken links unten und rechts oben in a) zu erkennen. Es gibt nur eine vollwertige Diagonale. Bei Systemen höherer Periodizität müsste die Kovarianzfunktion, entsprechend mehrere dieser Diagonalen, durch Superposition oder Trigonometrie-Funktionen erzeugen [3].

Die homogenen Bereiche der Matrix zeigen, dass die Trainingsdaten X mit gleichbleibender magnetischer Stimulanz erzeugt wurden. Gäbe es Fehllagen in der Erzeugung, sprunghaften Versatz des Sensor-Arrays oder Verkippung des Magneten wären die Flächen unterbrochen. Ebenfalls indiziert die durchgehende Diagonale, dass die Rotation konstant mit gleichbleibenden Abständen vollzogen worden ist. Würden Sprünge in der Rotation auftauchen, müssten diese durch Schnitte in der Diagonalen und eventuell durch Absenkungen der Ecken ersichtlich werden. Durch Bruch in der Periodizität würden Ecken der Matrix ganz verschwinden.

Der annähernd keilförmige Kurvenverlauf in Abbildung 2.12 b) bei ausgeschalteter Skalierung $\theta = (1, 1)$ weist auf ein System mit einfacher Komplexität hin [3]. Das heißt, es benötigt entweder eine erhöhte Anzahl von Trainingssamples, oder eine Parameteroptimierung und Aufbohren der Modellkomplexität, um von den Trainingsdaten abweichende Daten mit geringen Regressionsvarianzen prozessieren zu können [3]. Andersherum gesagt, gibt man alle möglichen in Winkelpositionen über die Trainingsdaten vor, muss der Regressionsfehler automatisch gegen null gehen. Die Modellanpassung auf eingespeiste Trainingsdaten sowie gleichzeitige und sinnvolle Verringerung des Datenumfangs, bilden dabei die zu haltende Balance in Bezug auf akzeptable Winkelfehler [3].

3 Software-Entwicklung für Optimierungsexperimente 0.0.4

25.04.2021

Die Software-Entwicklung erfolgt unter dem Gesichtspunkt zur Durchführung von Versuchsreihen. Ziel ist dabei die Parameterfindung zur Modelloptimierung für die Winkelauswertung. Durchzuführende Simulationen und Erprobungsexperimente basieren teilweise auf Zwischenergebnissen, die strukturiert im Software-Projektverzeichnis zwischen gespeichert sind, siehe Unterabschnitt G.2.2. Für die Auswertung der Simulationen sind unterstützende Grafiken angefertigt worden. Diese sind als Funktionen im Unterabschnitt G.4.3.3 und in den ausführbaren Skripten im Abschnitt G.3 enthalten. Das Kapitel dient zur näheren Erläuterung des modularen Software-Aufbaus und der zur Verfügung stehenden Simulationsprozesse. Die Entwicklungs-Roadmap der Simulations-Software ist mit Aufführung der einzelnen Beiträge in Abschnitt G.1 einzusehen.

3.1 Aufbau und Funktion

Als erster Schritt der hier stattfindenden Entwicklungsarbeiten wird ein Konzept aufgestellt, dass den erheblichen Simulationsaufwand praktisch benutzbar macht. Dafür sind die Simulationsbestandteile nach Aufgabenbereich und Charakter in Abbildung 3.1 zugeteilt. Die Implementierung erfolgt in zwei eigenständigen Simulationen. Für das Sensor-Array mittels Dipol-Feldgleichung in Anhang B und für die ASIC Simulation mit Gauß'scher Prozess-Regression in Anhang C. Zur Klassifizierung der Simulationen dienen hierbei Datendirektion und Datenbeschaffenheit in den einzelnen Simulationsabschnitten. Die Kopplung und Steuerung beider Simulationsabschnitte erfolgt über Datensätze. Die Datensatzbeschreibung und ihre Nutzung ist im Abschnitt G.5 nachzulesen.

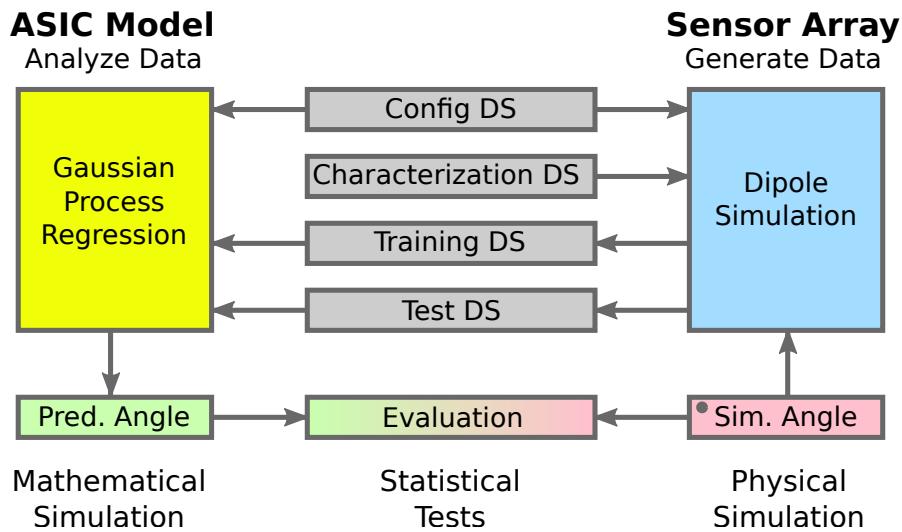


Abbildung 3.1: Simulationsaufbau im Überblick. Konzeptionelle Aufteilung der Simulationen für Sensor-Array und ASIC nach Simulationscharakter. Simulative Kopplung erfolgt, durch prozessierte Datensätze (DS). Eine statistische Auswertung bezieht sich auf angefahrene Simulationswinkel. Der Punkt kennzeichnet die Simulationswinkeleingabe und markiert den gedanklichen Startpunkt des Gesamtkonzepts.

Im ersten Simulationsschritt (Sensor-Array) werden Trainings- und Testdatensätze erzeugt. Die Datengenerierung in der Simulation basiert auf Charakterisierungsdatensätze. Diese stellen Technologieeigenschaften und Verhalten für die Simulation bereit, siehe Anhang A. Zur Datengenerierung sind physikalische Gleichungen aus Abschnitt 2.6 genutzt. Als zweiter Simulationsschritt (ASIC-Modell) folgt die Analyse der generierten Daten aus Schritt eins. Zur Analyse wird ein mathematisches Regressionsverfahren verwendet, siehe Abschnitt 2.7. Das Regressionsverfahren gewichtet Daten und macht entsprechende Vorhersagen gemäß eingestellter Regressionsziele und gewichteten Referenzdaten Anhang C. Das sind rein mathematische Vorhersagen für vorgegebene Funktionen. Ein physikalischer Gesamtbezug ist durch eine verbundene Auswertung beider Simulationsabschnitte herzustellen. Die Simulationssteuerung ist über einen gemeinsamen Konfigurationsdatensatz umgesetzt, siehe Tabelle B.1 und Tabelle C.1. Die jeweiligen Parametergruppen sind entsprechend ihrer Zugehörigkeit partiell in den Simulationsbetrieb eingebunden. Der Konfigurationsdatensatz kann je nach Bedarf erzeugt und manipuliert werden. Zur Konfigurationsgenerierung ist das Skript aus Unterabschnitt G.3.2 zu verwenden.

Die Zweischritt-Lösung bietet den Vorteil, dass zuerst verschiedene Trainings- und Testdatensätze generiert werden können. Nachfolgende und voneinander variierende Simulationen basieren hierbei auf gleichen Datensätzen. Sie sind damit vergleichbar für weitere Auswertungen, Diagnosen oder Optimierungen. Ein empfohlener Arbeitsablauf ist in Unterabschnitt G.2.5 festgehalten.

Zur Gestaltung der Simulations-Software ist ein modularer Ansatz nach Abbildung 3.2 verfolgt worden. Das modulare Konzept erhöht die Wiederverwendbarkeit des Quellcodes. Es ermöglicht einzelne Quellcodebestandteile miteinander zu kombinieren. Die Einbindung und Ausführung der Quellcodemodule aus Abschnitt G.4 erfolgt in Skripten des Abschnitt G.3. Die Software ist als Projekt in der Multi-Paradigmen-Programmiersprache Matlab umgesetzt, siehe Anhang F. Konzeptrichtlinien sind im Abschnitt G.2 festgehalten. Zusätzliche Anweisungen zur Arbeitsweise, Projektpflege, Dokumentation und Vorlagen für Skripte und Funktionen sind beigefügt. Alle Entwicklungsschritte sind mittels Git-Versionsverwaltung kommentiert und nachvollziehbar dokumentiert. Jede Quellcode- und Skript-Datei ist nach festgelegten Konventionen geschrieben worden [6]. Es ermöglicht eine automatisierte Dokumentation des gesamten Software-Projekts in Anhang G. Erzeugt ist diese mit Skripten aus Unterabschnitt G.3.1 und Unterabschnitt G.3.6.

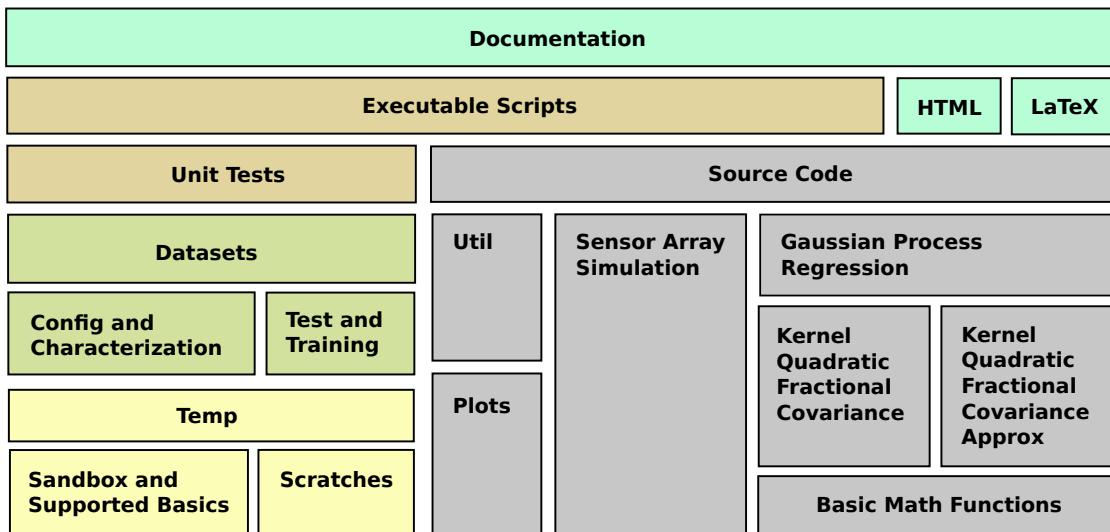


Abbildung 3.2: Blockschema Simulations-Software. Modularer Software-Gestaltung. Kernbestandteil ist funktionaler Quellcode. Quellcodemodule sind mit Datensätzen nach Bedarf in Skripten zu laden und ausführbar. Software-Dokumentation steht in HTML und LaTeX bereit. Als HTML ist diese in Matlab integriert. Entwurfsarbeiten sind nicht Bestandteil der Dokumentation, aber im Projektverzeichnis vorliegend.

3.2 Simulationsprozesse und Ausführung

In diesem Abschnitt der Arbeit werden Ausführungen zum Gesamt simulationsbetriebs und zur Moduleinbindung anhand von Prozessdarstellungen und Blockschemata erläutert. Jeder Simulationsabschnitt wird getrennt betrachtet. Prozesse und Schemata beziehen sich auf die Implementierung in den Anhängen und die dort beschriebenen Algorithmen. Gleichzeitig wird entsprechender Bezug zur Software-Dokumentation hergestellt, sodass ausgeführte Implementierungen zu umgesetztem Quellcode referenzierbar wird.

3.2.1 Sensor-Array-Simulation

Das Sensor-Array-Simulationsmodul Unterabschnitt G.4.1 ist funktional aufgebaut. Geschriebener Quellcode des Moduls mündet in eine Hauptsimulationsfunktion, siehe Unterabschnitt G.4.1.6. Diese ist mittels Skripten nach Abbildung 3.3 für den Simulationsbetrieb inklusive Konfiguration und Kennfelddatensatz zu laden.

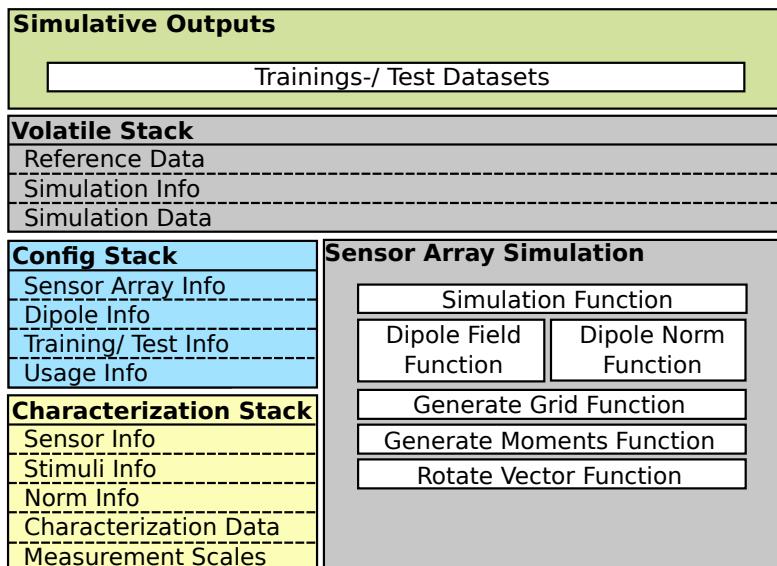


Abbildung 3.3: Blockschema Einbindung der Sensor-Array-Simulation. Das Schema beschreibt die Quellcodeeinbindung des Sensor-Array-Simulationsmodul für eine Ausführung mittels Skript-Dateien. Das Modul ist über die Modulsimulationsfunktion im Skript ausgeführt. Der Simulationsfunktion werden geladene Datensätze als entsprechende Simulations-Stacks zugeführt. Innerhalb die Funktions-Workspace sind alle Simulationsergebnisse prozessiert und zyklisch in Trainings-/ Testdatensätze zu speichern.

Das ausführende Skript der Simulation ist im Unterabschnitt G.3.3 zu finden. Es werden jeweils Trainings- und Testdatensätze gemäß eingestellter Konfiguration prozessiert. Die Sichtung der Datensätze und eine Analyse der Rohdaten ist mit Plot-Funktionen des Unterabschnitt G.4.3.3 durchzuführen. Die Plot-Funktionen können direkt ausgeführt werden. Eine Auswahl der zu sichtenden Daten ist über Nutzereingaben in der Konsole gesteuert. Erstellte Datensätze und Plots können mit Lösch-Skripten aus Unterabschnitt G.3.4 und Unterabschnitt G.3.5 entfernt werden.

Die Simulationsausführung nach Algorithmus 1 mittels Skript aus Unterabschnitt G.3.3 ist nochmals zur Verdeutlichung des Ablaufs als Prozess in Abbildung 3.4 dargestellt. Als erstes sind alle notwendigen Datensätze in den Skript-Workspace zu laden und als Eingabeargumente der Simulationsfunktion zuzuführen.

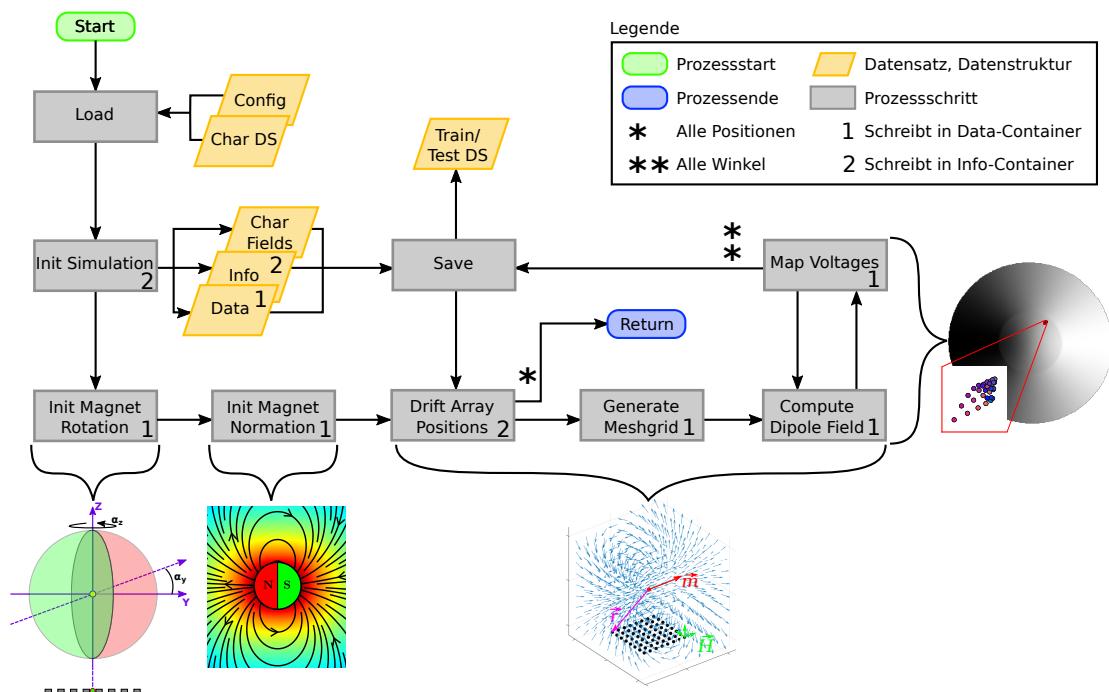


Abbildung 3.4: Sensor-Array-Simulation Prozessansicht. Prozessabfolge entsprechend der Implementierung aus Anhang B und Ausführung mit Skripten aus Abschnitt G.3 in Kombination mit Datensätzen (DS) nach Anhang A bzw. Abschnitt G.5.

Innerhalb der Funktion folgt eine schrittweise Initialisierung der Simulation nach einge stellter Konfiguration. Es werden Daten-Container angelegt, die Metadaten (Info) der Simulation und prozessierte Simulationsdaten (Data) beinhalten. Die Simulation erfolgt für fest eingestellte Sensor-, Magnet-, Rotationsparameter in Training-/ Testdaten und Verkippung. Die räumliche Lage des Sensor-Arrays ist dynamisch über Positionsvektoren in der Konfiguration gesteuert. So wird eine Positionsdrift des Arrays bei gleichbleibende Simulationsparameter umgesetzt. Die Simulation ist in verschachtelten For-Schleifen eingebettet. Die äußeren Schleifen setzen die Positionsdrift um. Die innere Schleife fährt die Magnetrotation durch. Für jede Position wird ein Datensatzpaar aus Trainings-/ Testdaten gespeichert, dass die volle Rotation enthält. Entsprechend des Schleifendurch laufs werden Info- und Data-Container aktualisiert. Anzahl der Rotationswinkel können für Trainings und Testdaten unterschiedlich eingestellt werden. Ebenfalls können unter schiedliche Rotationsauflösungen und Startphasen parametrisiert werden, siehe Unterab schnitt G.3.2.

3.2.2 Gauß-Prozess-Regression

Für die Verarbeitung von simulierten Sensor-Array-Datensätzen aus Unterabschnitt 3.2.1, dient das Quellcodemodule der Gauß-Prozess-Regression in Unterabschnitt G.4.2. Es ist funktional aufgebaut und erzeugt Regressionsmodelle in einer Trainingsphase. Ein erzeugtes Modell bildet zusammen mit einem Sensor-Array-Datensatz die Eingabeargumente für eine Arbeitsphase. In dieser werden Winkelvorhersagen auf Basis des Sensor-Array- Datensatzes berechnet. Das Quellcodemodul setzt sich aus drei weiteren Submodulen zusammen:

1. Mathematische Grundfunktionen, Unterabschnitt G.4.2.15
2. Kovarianzfunktion f. d_F^2 Gleichung C.4 (Matrixdaten), Unterabschnitt G.4.2.14
3. Kovarianzfunktion f. d_E^2 Gleichung C.4 (Vektordaten), Unterabschnitt G.4.2.13

Erläuterungen zur Einbindung von Trainingsphase und Arbeitsphase folgen zusammenfassend wie in Unterabschnitt 3.2.1 und beziehen sich auf das Demonstrationsskript für die Gauß-Prozess-Regression im Unterabschnitt G.3.7. Das Demonstrationsskript bedingt ein Trainings- und Testdatensatzpaar, dass mit gleichen Simulationsparametern in Bezug auf Position und Verkippung erstellt worden ist.

Trainingsphase

Die Bildung des Regressionsmodell in der Trainingsphase ist modular nach Abbildung 3.5 umgesetzt. Einstiegspunkt für die optimierte Modellerstellung nach Algorithmus 7 ist die Funktion in Unterabschnitt G.4.2.11. Diese entspricht der Implementierung in Abschnitt C.2. Der Funktion werden zuvor geladene Konfigurationen sowie Trainings- und Testdaten zugeführt. Als Rückgabe erfolgt das optimierte Regressionsmodell mit der Fähigkeit zur Winkelvorhersage auf Basis von weiteren Testdatensätzen.

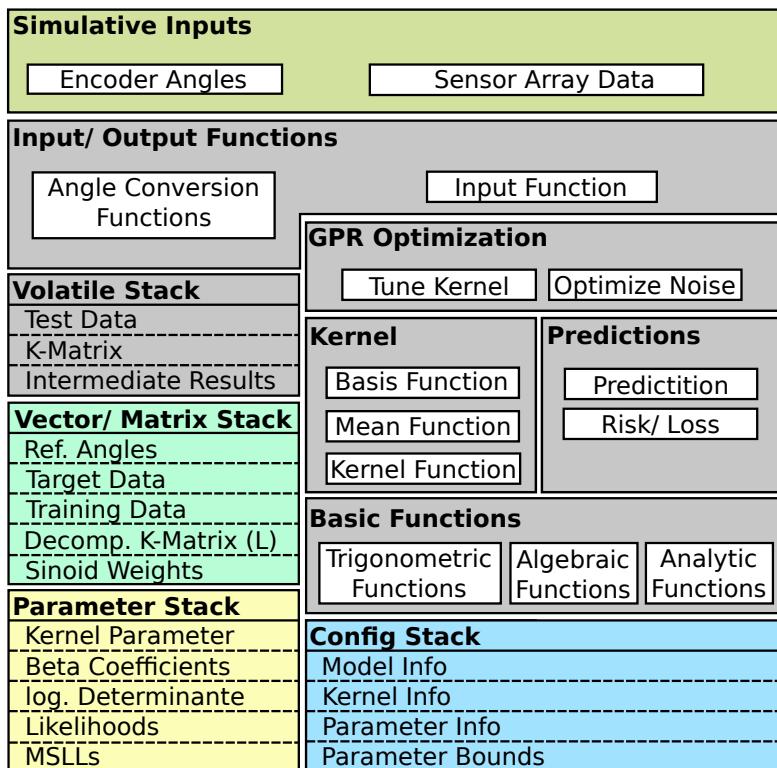


Abbildung 3.5: Blockschema Trainingsphase Regression. Veranschaulichung des Funktionsaufrufs für die Trainingsphase in einem Skript. Basierend auf übergebenen Konfigurations-Stack werden Modelfunktionalität aus Submodulen geladen und im Struct-Modell via Funktions-Handles verlinkt. Eingespeistes Trainings- und Testdatensatzpaar dient zur schrittweisen Modellinitialisierung und -Optimierung. Sich einstellende Parameterkonfigurationen werden mit Trainingsdaten als Stacks abgelegt. Resultierendes Struct-Modell enthält alle für Winkelvorhersagen nötigen Daten sowie Parameter und ihren Grenzen (engl. parameter bounds) aus der Optimierung.

Die Trainingsphase ist sehr aufwendig in ihrer Umsetzung und daher in Teilprozessschritte realisiert. Abbildung 3.7 zeigt den Hauptprozessablauf mit Weiterleitung zu den Teilprozessen in Abbildung 3.8 und Abbildung 3.9. Nach Übergabe der Regressionskonfiguration sowie der Trainings- und Testdaten, wird das Regressionsmodell in einem Struct initialisiert, siehe Funktion im Unterabschnitt G.4.2.1 und Abbildung 3.6.

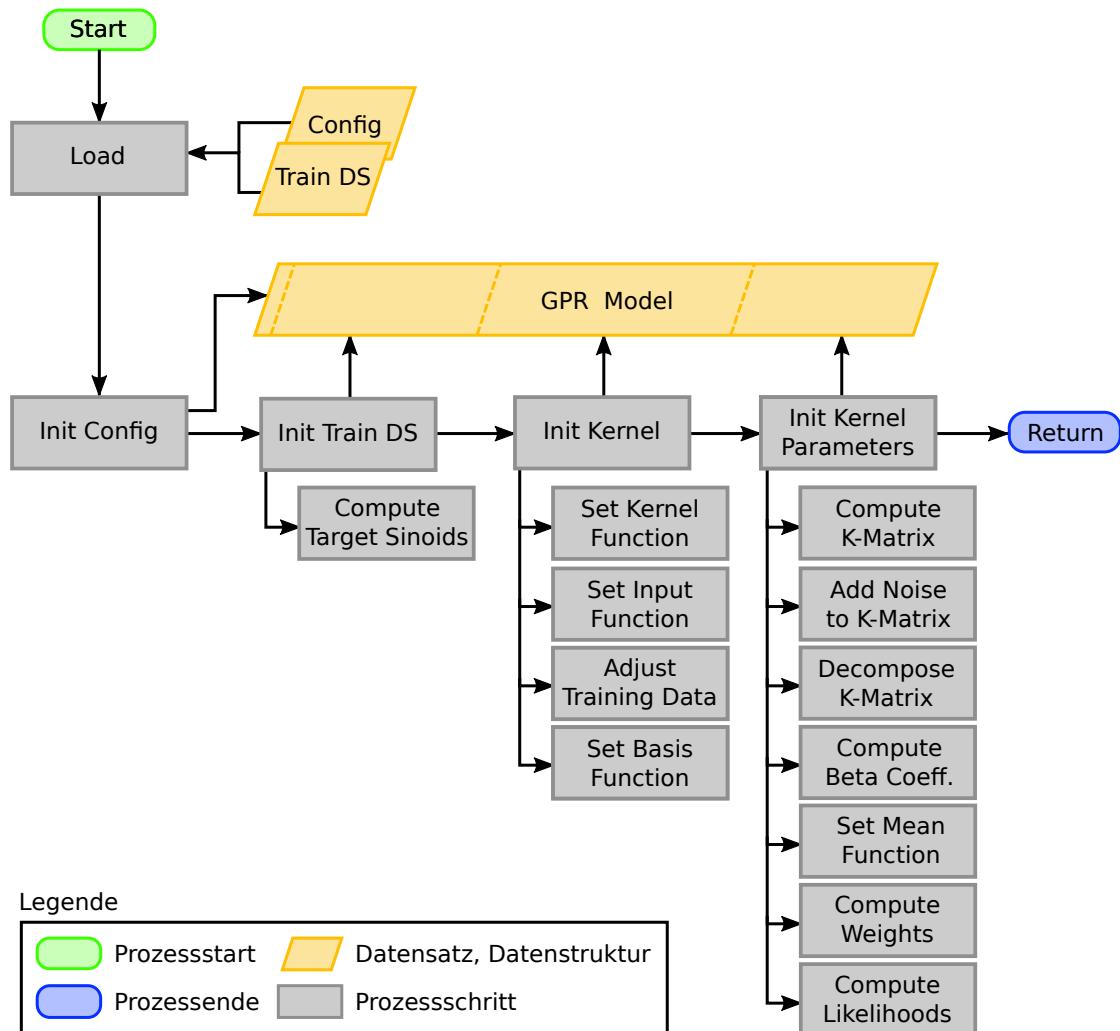


Abbildung 3.6: Regressionsinitialisierung Prozessansicht. Nach Algorithmus 2 in mehreren Teilschritten. Konfiguration, Trainingsdatensatz, Kernel-Module und abschließende Regressionsinitialisierung sind mit Funktionen aus Unterabschnitt G.4.2.2 bis Unterabschnitt G.4.2.5 realisiert.

Anschließend wird nach eingestellter Konfiguration die Rauschniveaumodellierung mittels der Matlab BayesOpt-Funktion initialisiert. Dazu werden entsprechende Berechnungen auf den Testdaten mit der Funktion aus Unterabschnitt G.4.2.12 umgesetzt. Für jedes ausprobierte Rauschniveau folgt eine Optimierung des Modells nach Abbildung 3.9 und eine darauffolgende Berechnung Modellverluste mittels Funktion aus Unterabschnitt G.4.2.10. Das Gesamtverfahren über die BayesOpt-Funktion ist ein Standardverfahren, dass durch Probieren sich einer optimalen Lösung annähert. Es werden ausprobierte Rauschniveaus und evaluierte Verluste nach Gleichung C.22 und Algorithmus 7 über alle Versuchsdurchläufe zwischengespeichert. Nach erreichen der eingestellten Durchlaufzahl für die Rauschniveaumodellierung mittels BayesOpt-Funktion, gibt die BayesOpt-Funktion alle ausprobierten Niveaus und errechneten Modellverluste in einem Result-Objekt zurück. Aus dem Result-Objekt werden für die finale Feinabstimmung das gefundene Rauschniveau am Minimum der aufgezeichneten Modellverluste entnommen.

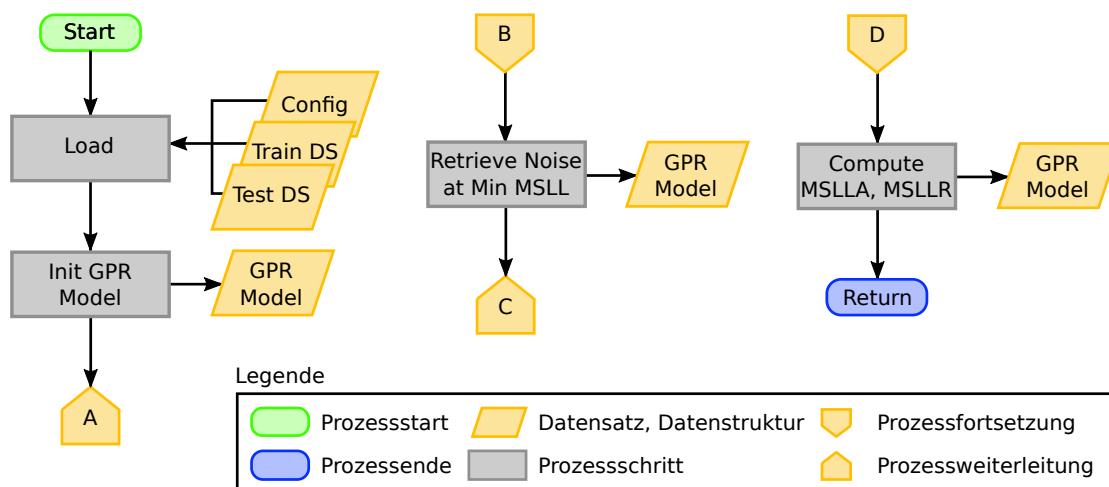


Abbildung 3.7: Regressionsoptimierung/-Generalisierung Prozessansicht. Darstellung Algorithmus 7 Teilprozessschritte. Modellinitialisierung in Abbildung 3.6. Rauschoptimierung (A, B in Abbildung 3.8) und finale Feinabstimmung (C, D in Abbildung 3.9).

Dieser Prozess wird innerhalb der BayesOpt-Funktion wiederholt, bis die eingestellte maximale Durchlaufzahl erreicht ist. Wird zu früh abgebrochen, kann bei falsch gewählten Parametergrenzen (engl. parameter bounds) keine hinreichend genaue Lösung gefunden werden. Daher ist die Durchlaufzahl bei weiten Parameter-Bounds entsprechend hoch zu wählen, sodass der Algorithmus genügend Versuche hat schlechte Wertebereiche auszuschließen.

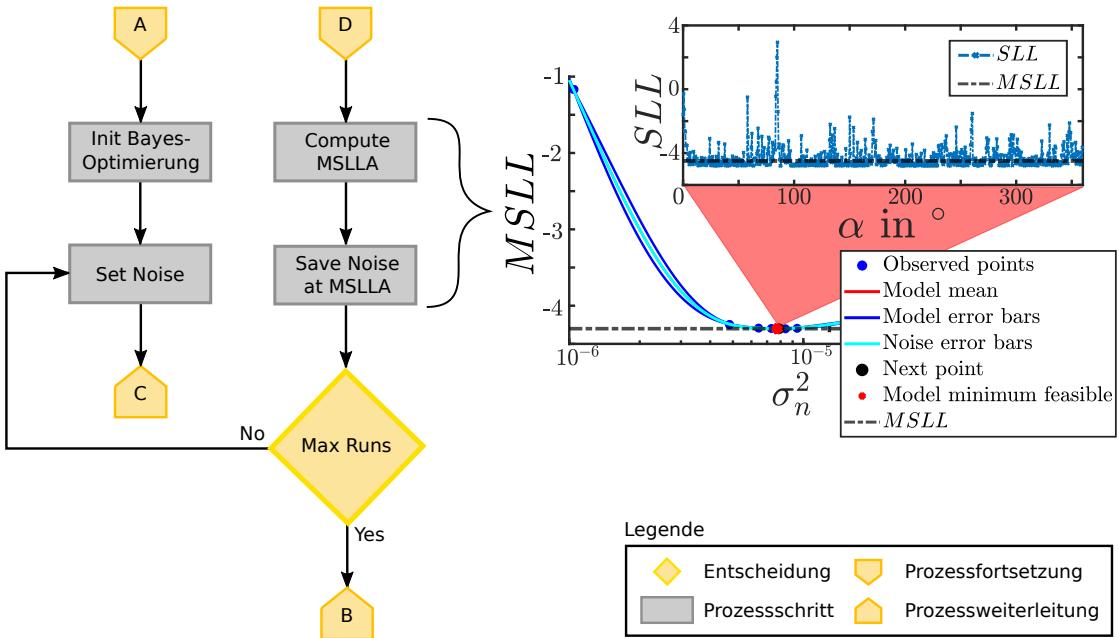


Abbildung 3.8: Rauschniveaumodellierung Prozessansicht. Kernprozess in Algorithmus 7 bzw. Abbildung 3.7. Löst Min-Kriterium nach Gleichung C.22. Die Berechnung ist als Funktions-Handle von Unterabschnitt G.4.2.12 der BayesOpt-Funktion zu übergeben. Die Rauschniveaumodellierung misst durch Probieren das Rauschniveau σ_n^2 gegenüber dem mittleren Modellverlust $MSLL$. Zwischenprozess ist die Parameteroptimierung (C, D) in Abbildung 3.9

Der Zwischenprozessschritt zur inneren Parameteroptimierung nach Abbildung 3.9 ist in der Funktion Unterabschnitt G.4.2.6 implementiert. Diese bedient sich der Matlab Fmincon-Funktion um Algorithmus 5 zu lösen. Das aufzulösende Min-Kriterium aus Gleichung C.14 wird der Fmincon-Funktion als Funktions-Handle bereitgestellt. Das Min-Kriterium wird mittels der Funktion aus Unterabschnitt G.4.2.7 berechnet. Dabei wird das Regressionsmodell z.T. reinitialisiert. Dieser Prozessschritt wird auch zur finalen Feinabstimmung mit optimiertem Rauschniveau genutzt. Nach Abarbeitung aller Optimierungsprozesse steht das fertige Regressionsmodell als Eingabeargument für die Arbeitsphase bereit.

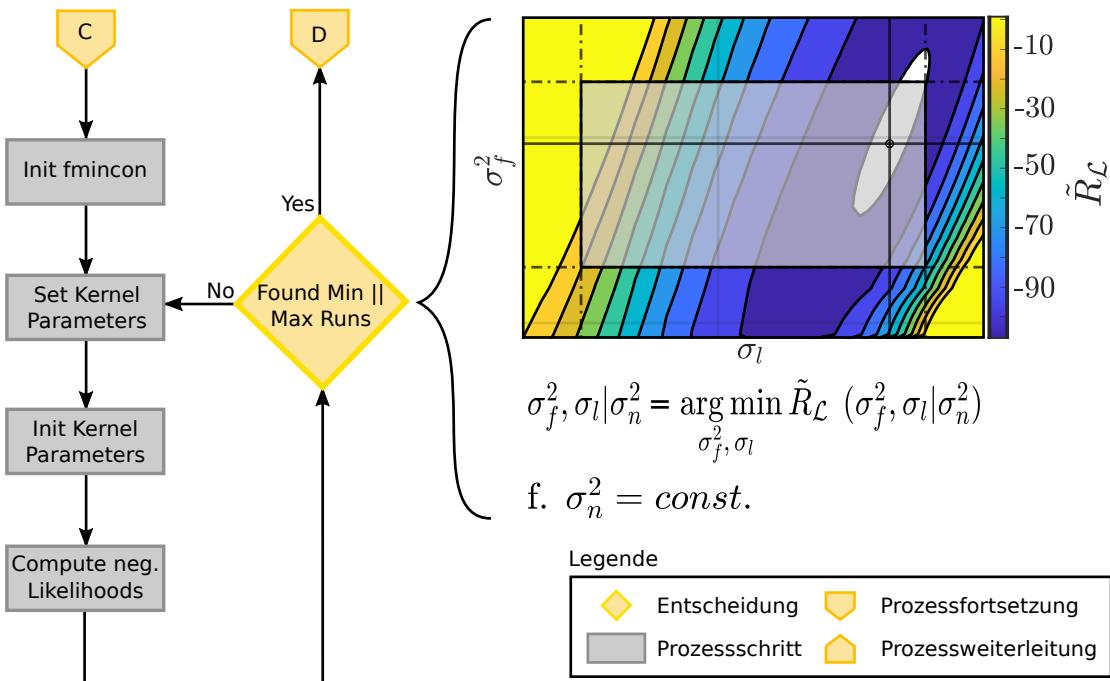


Abbildung 3.9: Regressionsparameteroptimierung Prozessansicht. Lösung des Algorithmus 5 mittels Matlab Fmincon-Funktion. Das Min-Kriterium aus Gleichung C.14 ist als Funktion in Unterabschnitt G.4.2.7 als Funktions-Handle an die Fmincon-Funktion zu übergeben. Die Parameteroptimierung für die Kovarianzfunktion $\theta = (\sigma_f^2, \sigma_l)$ sucht, in einem durch Parameter-Bounds eingeschränkten Areal, nach lokalem Minimum $\tilde{R}_{\mathcal{L}}$ bei logarithmisch gegeneinander aufgetragenen Parameterkombination für σ_f^2 und σ_l .

Arbeitsphase

Die Einbindung der Arbeitsphase ist im Vergleich zur Trainingsphase trivial. Das fertig optimierte Struct-Model wird einfach zusammen mit einem kompletten Testdatensatz der Vorhersage-Funktion aus Unterabschnitt G.4.2.9 und der Verlustfunktion aus Unterabschnitt G.4.2.10 übergeben. Berechnete Vorhersagen und Verluste stehen dann gemäß Abschnitt C.3 als Ergebnisvektoren bereit und können zur weiteren Auswertung genutzt werden.

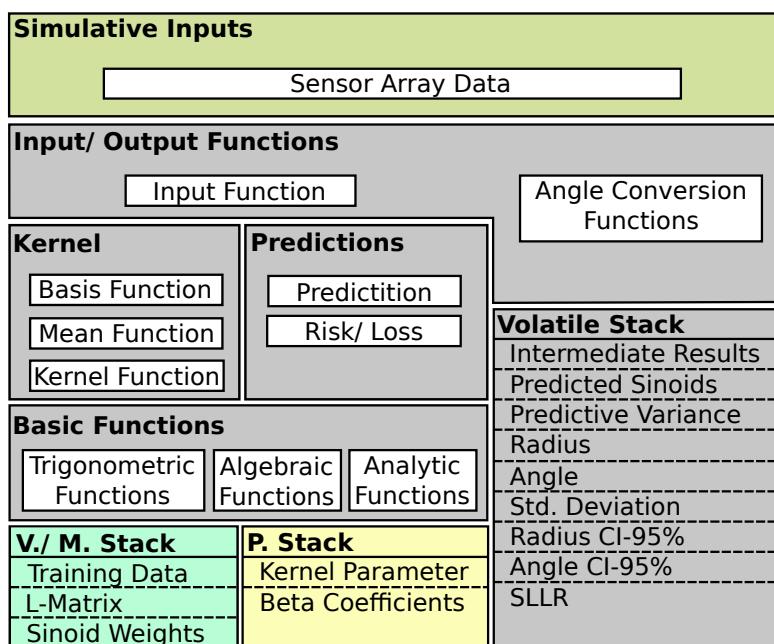


Abbildung 3.10: Blockschema Arbeitsphase Regression. Das fertig optimierte Modell wird mit minimalem Parameterbetrieb in Vorhersage- und Verlustfunktion als Argument zusammen mit den Testdaten übergeben. Einbindung ist funktional, siehe Unterabschnitt G.4.2.9 und Unterabschnitt G.4.2.10.

4 Erprobungs- und Optimierungsexperimente 0.0.2

26.04.2021

4.1 Vergleich der Kovarianzfunktionen

4.2 Anpassung der Referenzwinkelanzahl

4.3 Anpassung des Rauschniveaus

4.4 Anpassung der Parametergrenzen

5 Auswertung 0.0.1 13.01.2021

5.1 Gegenüberstellung der GPR-Modelle

- Aufwand der Trainingsphase
- Nötige Parameter und zu Speichernde Werte
- Arbeitsphase, Genauigkeit, Fehlererkennung, Stabilität

6 Zusammenfassung und Bewertung 0.0.1

13.01.2021

- Kurzdarstellung der Ergebnisse der Arbeit
- Offene Punkte und Probleme
- Ansätze zur Weiterführung für zukünftige Arbeiten
- Bewertung der Ergebnisse in Bezug auf die Anwendung

Abbildungsverzeichnis

1.1	Platinen-Sensor-Array im Maßstab	3
1.2	Ansatzdarstellung zur Generierung eines Simulationsmodell des magnetischen Sensor-Arrays	4
1.3	Veranschaulichung eines vollständigen Sensor-ICs für die Drehwinkelerfassung	5
2.1	Klassischer Anwendungsfall für die Drehwinkelerfassung	9
2.2	Kreisdarstellung der Winkelmessung	10
2.3	Allgemeine Kreisdarstellung des euklidischen Winkelabstands	12
2.4	Schichtmodelle dreier magnetoresistive Effekte	13
2.5	TMR Drehwinkelapplikation	16
2.6	Magnetfeldstimuli zur Erzeugung von Sensorkennfeldern	18
2.7	TDK TAS2141-AAAB Übertragungskennlinie	19
2.8	Approximierter Kugelmagnet	20
2.9	Geometrischer Aufbau und Ausrichtung des Sensor-Arrays	22
2.10	Resultierende Sensor-Array-Daten	23
2.11	Simulation der Dipol-Feldgleichung	25
2.12	Kernel-Implementierung der Vorarbeiten	29
3.1	Simulationsaufbau im Überblick	33
3.2	Blockschema Simulations-Software	34
3.3	Blockschema Einbindung der Sensor-Array-Simulation	35
3.4	Sensor-Array-Simulation Prozessansicht	36
3.5	Blockschema Trainingsphase Regression	38
3.6	Regressionsinitialisierung Prozessansicht	39
3.7	Regressionsoptimierung/-Generalisierung Prozessansicht	40
3.8	Rauschniveaumodellierung Prozessansicht	41
3.9	Regressionsparameteroptimierung Prozessansicht	42
3.10	Blockschema Arbeitsphase Regression	43

A.1	TDK TAS2141-AAAB Brückenkennfelder	57
A.2	TDK TAS2141-AAAB Kennfeldquerschnitte	58
B.1	Kennfeld-Mapping	61
B.2	Sensor-Array-Datensatz Teilansicht	62
E.1	Kovarianzfunktionen im Vergleich	84
E.2	Gegenüberstellung der Kovarianzmatrizen	85
E.3	Vergleich der Modellverluste nach Winkel und Radius für die erste Kovarianzfunktion	85
E.4	Vergleich der Modellverluste nach Winkel und Radius für die zweite Kovarianzfunktion	86
E.5	Timing vs Error	87
E.6	MSLL vs Error für Nref 17	88
E.7	QFCAPX Z N17 Rotation	89
E.8	Angepaster Parameter Bounds	90
E.9	QFCAPX Z N17 Optimierung Runs 10	90
E.10	QFCAPX Z N17 Modell	91

Tabellenverzeichnis

A.1	Eckdaten TDK TAS2141-AAAB Kennfelder	56
B.1	Sensor-Array-Simulationsparameter	59
C.1	Gauß-Prozess-Regression-Simulationsparameter	63
D.1	Simulationsparameter der Erprobungs- und Optimierungsexperimente . .	81
F.1	Genutzte Software	92

Algorithmenverzeichnis

1	Sensor-Array-Simulation	60
2	Modellinitialisierung mit konst. Trainingsdaten und Parametern	64
3	Berechnung der Kovarianzmatrix $K(X, X \theta)$	67
4	Berechnung der β Polynomkoeffizienten aus Gleichung C.10	70
5	Modelloptimierung über Fmincon-Funktion f. $\sigma_n^2 = \text{konst.}$	74
6	Modellvorhersage f. Sinoide eines Testwinkel mit $X_* \mapsto \alpha_*$	76
7	Modellgeneralisierung über BayesOpt-Funktion f. alle $X_* \mapsto \alpha_*$	80

Glossar

AMR-Effekt Anisotroper-Magnetoresistiver-Effekt.

Arbeitsgruppe Sensorik Die Arbeitsgruppe Sensorik steht unter Leitung von Prof. Dr.-Ing. Karl-Ragmar Riemschneider und ist unter dem Department Informations- und Elektrotechnik Teil der Fakultät Technik un Informatik an der HAW Hamburg.

GMR-Effekt Riesiger-Magnetoresistiver-Effekt.

HAW Hamburg Die HAW Hamburg ist die Hochschule für Angewandte Wissenschaften in Hamburg und war die ehemalige Fachhochschule am Berliner Tor.

Kennfeld Zweidimensionales Charakterisierungsabbild einer Wheatstone'schen Sensorbrücke eines magnetoresistiven Winkelsensors. Erstellt durch die Kennfeldmethode zur Charakterisierung von magnetischen Winkelsensoren.

Kennfeldmethode Charakterisierungsverfahren zur Ausmessung magnetoresistiver Winkelsensoren, bestehend aus zwei zueinander verdrehten Wheatstone-Brücken. Die resultierenden Charakterisierungsergebnisse können als Kennfelddatensätze zur Simulation von magnetischen Sensoren genutzt werden.

Kennfeldpaar Charakterisierungsergebnis der Kennfeldmethode für die Charakterisierung magnetoresistiver Winkelsensoren. Bestehend aus zwei Kennfeldern, jeweils als Repräsentanten der Wheatstone-Brücken eines Winkelsensors.

Kreuzspulen-Messstand Automatisierter Messtand zur Charakterisierung von Winkelsensoren. Der Messtand nutzt ein Kreuzspulen-System, in dessen Mitte der Winkelsensor platziert ist. Das Spulensystem erzeugt ein moduliertes, langsam rotierendes Anregungsmagnetfeld. Parallel zeichnet der Messtand, die zur Charaktisierung nötigen, Spannungsausgaben des Sensors und Anregungsströme der Spulen auf. Beides

erfolgt programmatisch. Die aufgezeichneten Messdaten können im Anschluss zu Kennfeldern evaluiert werden.

Kreuzspulen-System Spulensystem in Kreuzanordnung in dessen Mitte ein zu messendes Senor-IC platziert wird. Die Spulen sind Maßanfertigungen mit ganz bestimmten Messeigenschaften. Spulenfaktoren sind speziell ausgerechnet und nachgemessen. Kernelement des Kreuzspulen-Messstandes. Eingespeiste Spulenströme erzeugen entsprechende magnetische Felder in X - und Y -Richtung. Die Spulenströme erzeugen, entsprechend der Spulenfaktoren, H_x - und H_y -Feldstärken. Die Feldstärken sind direkt proportional zu den Einspeiseströmen. Die Ströme werden über niederohmige Shunt-Widerstände mit gemessen. Die Feldstärken können so über die Spulenfaktoren zurückgerechnet und zur Auswertung genutzt werden..

Sensor-Array Array aus geometrisch, gleichmäßig angeordneten Einzelsenoren, die messtechnische Aufgaben im Verbund bewältigen. Jeder einzelne Senor stellt dabei ein Senor-Pixel dar. Erhobene Messdaten sind entsprechend der Sensor-Array-Struktur in Matrixdatenformate zu bewerten und zu behandeln..

Sensorkopf Signal erzeugender Teil eines Sensor-ICs, dem eine Einheit zur weiteren Signalverarbeitung nachgeschaltet ist. Für die Drehwinkelerfassung besteht besteht die Signalerzeugung zumeist aus zwei verdrehten Wheatstone-Brücken, deren einzel Widerstände mittels magnetoresistiven Materialien aufgebaut sind.

Sensor-Pixel Einzelsensor im Sensor-Array. Allein betrachtet als vollwertiger, analoger Sensor zu betrachten. Mehrere Sensor-Pixel zusammen verschaltet ergeben ein Sensor-Array..

TMR-Effekt Tunnel-Magnetoresistiver-Effekt.

Wheatstone'sche Brückenschaltungen Messbrückenschaltung bestehend aus zwei Spannungsteilern, die parallel zu einer gemeinsamen Quelle geschaltet sind. Es wird eine Differenzspannung über die Mittelabgriffe der Spannungsteiler gemessen. Allgemein bekanntes Messprinzip. 1833 von Samuel Hunter Christie erfunden und nach dem britischen Physiker Sir Charles Wheatstone benannt. Abgekürzt auch Wheatstone-Brücken oder in der Sensorik auch Sensorbrücken genannt. Ein Sensorkopf für die Winkelmessung setzt sich in der Regel aus zwei solch gearteter Brückenschaltungen zusammen.

Abkürzungen

AMR Anisotrope-Magnetoresistance.

ASIC Application-Specific-Integrated-Circuit.

CPU Prozessorkern.

GMR Giant-Magnetoresistance.

HDD Festplattenlaufwerk.

ISAR Integrated-Sensor-Array.

OS Betriebssystem.

RAM Arbeitsspeicher.

SW Software.

TMR Tunnel-Magnetoresistance.

Literatur

- [1] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993. ISBN: 9780511564345.
- [2] K. V. Mardia und P. E. Jupp. *Directional Statistics*. Bd. Wiley Series in Probability and Statistics. John Wiley und Sons, Inc., 1999. ISBN: 9780471953333.
- [3] C. E. Rasmussen und C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN: 026218253X. URL: www.gaussianprocess.org/gpml (besucht am 30.10.2020).
- [4] M. Plum. „Orthogonalprojektionen, Orthonormalsysteme und -basen“. In: *Vorlesung Differentialgleichungen und Hilberträume*. Vorlesung (2011). KIT, 2012.
- [5] P. Guerrero und J. Ruiz del Solar. „Circular Regression Based on Gaussian Processes“. In: *2014 22nd International Conference on Pattern Recognition*. 2014. DOI: [10.1109/ICPR.2014.631](https://doi.org/10.1109/ICPR.2014.631).
- [6] R. Johnson. *MATLAB Style Guidelines 2.0*. Version 2. MATLAB Central File Exchange, 2014. URL: <https://de.mathworks.com/matlabcentral/fileexchange/46056-matlab-style-guidelines-2-0> (besucht am 21.09.2020). Online.
- [7] M. Lang, O. Dunkley und S. Hirche. „Gaussian process kernels for rotations and 6D rigid body motions“. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014. DOI: [10.1109/ICRA.2014.6907617](https://doi.org/10.1109/ICRA.2014.6907617).
- [8] NXP Semiconductors. *KMZ60 Angle sensor with integrated amplifier*. Datenblatt, 2014.
- [9] R. A. van de Geijn. *Notes on Vector and Matrix Norms*. The University of Texas Austin, 2014.
- [10] H. Lemme. *Messung durch den Tunnel*. Hrsg. von Elektroniknet. 2016. URL: <https://www.elektroniknet.de/messen-testen/sensorik/messung-durch-den-tunnel.133265.html> (besucht am 25.01.2021). Online.

- [11] TDK. *TMR Angle Sensor TAS2141-AAAB*. Datenblatt, 2016.
- [12] H. Pape. „Simulation und Auswertung von Permanentmagneten für manetoresistive Sensor-Arrays“. Bachelorarbeit HAW Hamburg, 2017.
- [13] infineon. *TLE5x09A16(D) Analog AMR/GMR Angle Sensors*. Datenblatt, 2018.
- [14] T. Mehm. „Schaltungsentwurf und Mikrocontrollersteuerung für ein Tunnel-Magnetoresistives Sensor-Array“. Bachelorarbeit HAW Hamburg, 2019.
- [15] T. Schüthe, A. Albounyan und K. Riemschneider. „Two-Dimensional Characterization and Simplified Simulation Procedure for Tunnel Magnetoresistive Angle Sensors“. In: *Sensors Applications Symposium (SAS)*. (13. März 2019). IEEE, 2019. DOI: [10.1109/SAS.2019.8706125](https://doi.org/10.1109/SAS.2019.8706125). URL: <https://ieeexplore.ieee.org/document/8706125> (besucht am 05.10.2020). Online.
- [16] Bitbucket. *Feature Branch Workflow in Git*. Hrsg. von ATlassian. 2020. URL: <https://www.atlassian.com/de/git/tutorials/comparing-workflows/feature-branch-workflow> (besucht am 10.09.2020). Online.
- [17] J. Ernsting. „Funktionsdemonstrator für magnetische Sensor-Arrays auf Basis des Mikrocomputers Raspberry PI“. Bachelorarbeit HAW Hamburg, 2020.
- [18] T. Schüthe, K. Jünemann und K. Riemschneider. *Tolerance Compensation based on Gaussian Processes for Angle Measurements with Magnetic Sensor Arrays*. HAW Hamburg, 2020.
- [19] T. Schüthe u. a. „Positionserfassung mittels Sensor-Array aus Tunnel-Magnetoresistiven Vortex-Dots und lernender Signalverarbeitung“. In: *Tille T. (eds) Automobil-Sensorik 3*. Springer Vieweg, Berlin, Heidelberg, 2020. ISBN: 978-3-662-61259-0. URL: https://doi.org/10.1007/978-3-662-61260-6_14.
- [20] T. Schüthe u. a. „Positionserfassung mittels Sensor-Array aus Tunnel-Magnetoresistiven Vortex-Dots und lernender Signalverarbeitung“. In: *8. Fachtagung Sensoren im Automobil*. 2020.
- [21] T. Tille. *Automobil-Sensorik-3*. Springer Vieweg, 2020. ISBN: 978-3-662-61259-0.

A TDK TAS2141-AAAB

Kennfelddatensatz 0.0.1 29.03.2021

Der Anhang beinhaltet die Kennfelddarstellung eines TAS2141-AAAB TMR-Winkelsensor der Firma TDK [11]. Die Charakterisierung des Sensor-ICs ist mittels Kennfeldmethode [15] vorgenommen worden. Das Ausmessen des TMR-Sensors hat im Labor der Arbeitsgruppe Sensorik stattgefunden. Als Charakterisierungsergebnis liegt entsprechender Datensatz vor und dient in dieser Arbeit als Simulationsgrundlage für die Sensor-Array-Simulation. Der Datensatz ist von der Arbeitsgruppe Sensorik zur Verfügung gestellt worden. Eine detaillierte Zusammensetzung des Datensatzes ist im Unterabschnitt G.5.1 aufgeführt.

Eigenschaft	Wert	Einheit
H_x -Skala	$-25 \dots 25$	kA m^{-1}
H_y -Skala	$-25 \dots 25$	kA m^{-1}
H_x -Schrittweite	0,1961	kA m^{-1}
H_y -Schrittweite	0,1961	kA m^{-1}
Auflösung	256×256	Pixel
Wertebereich $V(H_x, H_y)$	Normiert	mV V^{-1}
Normfaktor	$1 \cdot 10^3$	mV
Gain	1	-
Brückenverdrehung	90	°
Periodizität	360	°

Tabelle A.1: Eckdaten TDK TAS2141-AAAB Kennfelder

Die Charakterisierung mittels Kennfeldmethode generiert zwei Kennfeldpaare zu sehen in Abbildung A.1. Das erste Kennfeldpaar a) und c) referenziert sich aus dem steigenden Messverlauf, der amplitudenmodulierten H_x -/ H_y -Stimuli. Das zweite b) und d) setzt sich aus dem fallenden Stimuli zusammen. Ein Kennfeld repräsentiert dabei eine Wheatstone-Brücke des Winkelsensors [15]. Bedingt durch die Verdrehung beider Brücken [11], ist ein

entsprechendes Kennfeldpaar zueinander um 90° verdreht. Die Kennfelder besitzen, je ein Minimum und Maximum, dass bei Abfahren eines Kreises auf einem Kennfeld zur 360° Periodizität führt. Die Kennfelder entsprechen somit dem Kernverhalten des Winkelsensors [11]. Tabelle A.1 fasst die Grundeigenschaften der Kennfelder zusammen. Beim zusammensetzen der Kennfelder, sind die gemessenen Ausgangsspannungen normiert und von Offsets bereinigt worden. Das erleichtert den Simulationseinsatz mit variablen Betriebsspannungen. So können für beliebige Feldstärken, Ausgangsspannungen nach Gleichung A.1 aus den Kennfeldern entnommen werden.

$$V_{out}(H_x, H_y) = Gain \cdot \frac{V_{cos,sin}(H_x, H_y)}{Normfaktor} \cdot V_{CC} + V_{Offset} \quad (\text{A.1})$$

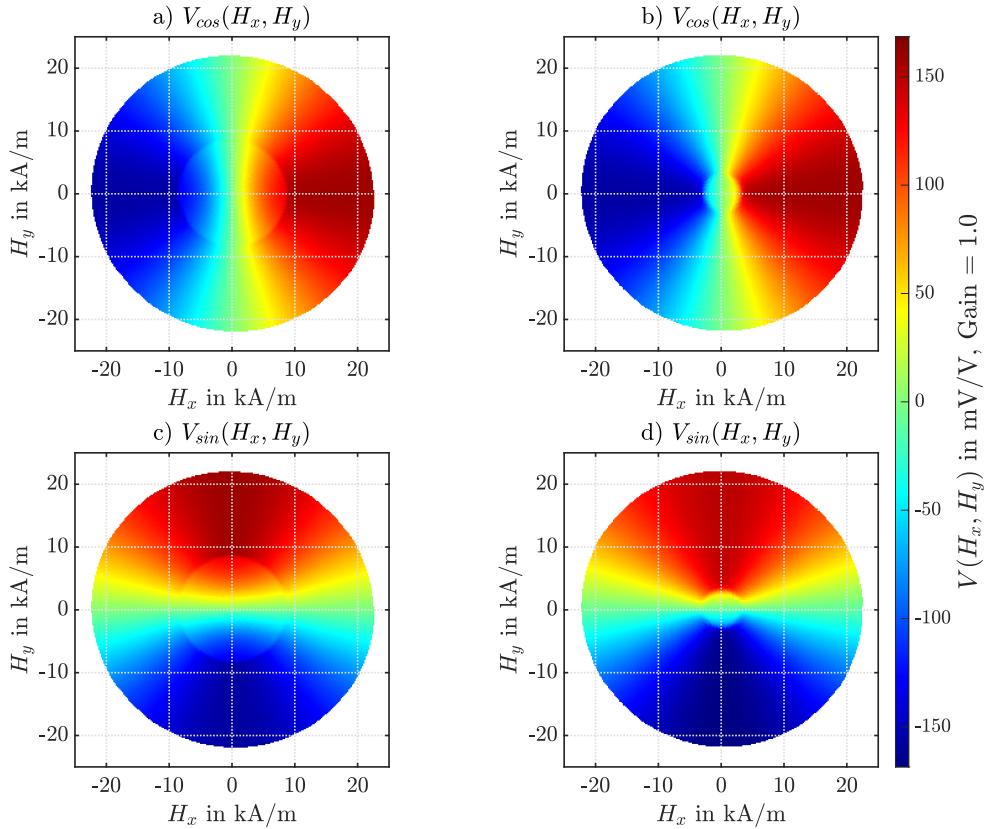


Abbildung A.1: TDK TAS2141-AAAB Brückenkennfelder. Kennfelder der Cosinus-Brücke a) und b). Kennfelder der Sinus-Brücke c) und d). a) und c) gewonnen aus steigenden Amplitudenmodulation. b) und d) gewonnen fallenden Modulation. Die Kennfelder sind normiert in mV V^{-1} . Grafik nachempfunden aus [15].

Im Vergleich der Kennfeldpaare biete sich das erste aus aus Abbildung A.1 a) und c) für eine Simulation an. Die Kennfelder besitzen größere Plateauflächen [15]. In Abbildung A.2 a) und c) ist das Kennfeldpaar nochmals gesondert dargestellt. Für das Kennfeld, der Cosinus-Wheatstone-Brücke in a), sind Querschnitte für variable H_x - und verschiedenen konstante H_y -Feldstärken in b) aufgetragen. Das gleiche vice versa in d) für Sinus-Wheatstone-Brücke aus c). Die Plateau-Grenzen liegen in H_x - und H_y -Richtung ca. bei $\pm 8,5 \text{ kA m}^{-1}$ und sind als Limits in Abbildung A.2 b) und d) gekennzeichnet. Es zeigt sich ein annähernd linearer Bereich für die Übertragungskennlinien bei $H_{x,y} = 0 \text{ kA m}^{-1}$. Dieser Arbeitsbereich ist für die Simulation einzustellen [15].

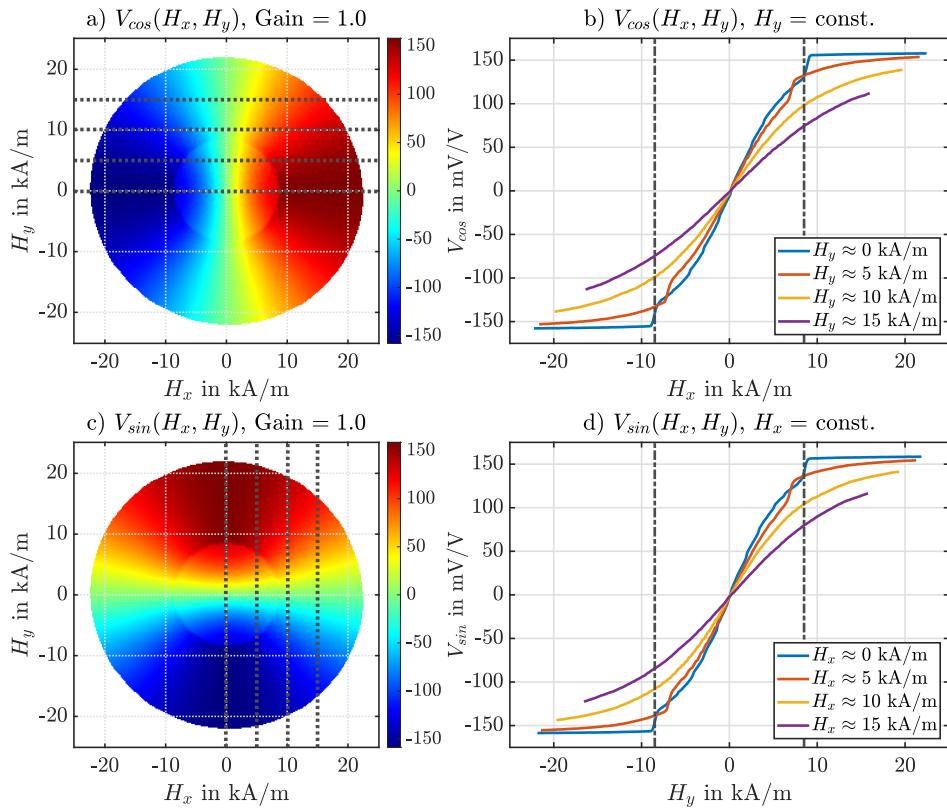


Abbildung A.2: TDK TAS2141-AAAB Kennfeldquerschnitte. Cosinus-Brücken-Kennfeld in a) und Sinus-Brücke in c). In b) und d) sind folgend der Verdrehung Querschnitte aus a) und c) aufgetragen. In b) V_{cos} f. variable H_x - und verschieden konst. H_y -Feldstärken. In d) vice versa für V_{sin} mit verschiedenen konst. H_x - bei variablen H_y -Feldstärken. Breite lineare Plateaus liefern einen annähernd linearer Arbeitsbereich in c) und d) zw. $\pm 8,5 \text{ kA m}^{-1}$ f. Übertragungskennlinien $H_{x,y} = 0 \text{ kA m}^{-1}$. Grafik aus [15].

B Sensor-Array-Simulation

Implementierung 0.0.1 07.04.2021

Der Anhang veranschaulicht die Handhabung und Standardparametrierung für die Sensor-Array-Simulation. Es sind optimale Simulationsparameter in Tabelle B.1 aufgeführt. Diese sind im Konfigurationsskript einzustellen und vorab der Simulation auszuführen. Das Skript erstellt eine MAT-Datei. Diese enthält, die in Gruppen zusammengefasste Simulationsparametrierung. Bei Simulationsausführung sind betreffende Parametergruppen aus der Konfigurationsdatei zu laden.

Parametergruppe	Parameter	Wert	Einheit	Kurzbeschreibung
SensorArrayOptions	geometry	'square'	-	Array-Geometrie-Indikator
	dimension	8	-	Sensor-Array-Pixel $N_{Pixel} \times N_{Pixel}$
	edge	2	mm	Sensor-Array-Kantenlänge
	V_{cc}	5	V	Sensor-Array-Betriebsspannung
	V_{off}	2,5	V	Sensor-Brücken-Offset-Spannung
	V_{norm}	$1 \cdot 10^3$	mV	Kennfeldnormierung
DipoleOptions	sphereRadius	2	mm	Kugelmagnetradius
	H_{0mag}	200	kA m ⁻¹	Betragsfeldstärke Magnetfeldnormierung
	z_0	1	mm	Z-Abstand Magnetfeldnormierung
	m_{0mag}	$1 \cdot 10^6$	A m ²	Magnitude d. mag. Moments
Training-/ TestOptions	useCase	'Training' / 'Test'	'char'	Datensatzindikator f. Anwendungszweck
	xPos	[0,]	mm	Sensor-Array X-Positionsvektor
	yPos	[0,]	mm	Sensor-Array Y-Positionsvektor
	zPos	[7,]	mm	Sensor-Array Z-Positionsvektor
	tilt	0	°	Magnetverkippung in Y-Achse
	angleRes	0,5	°	Winkelauflösung f. Magnetrotation
	phaseIndex	0	-	Phasenverschiebung-Index f. Startwinkel
	nAngles	20 / 720	-	Anzahl gleich verteilter Simulationswinkel
	BaseReference	'TDK'	char	Kennfelddatensatzindikator
	BridgeReference	'Rise'	char	Kennfeldindikator

Tabelle B.1: Sensor-Array-Simulationsparameter. Default-Parameter für die Simulation mit ideal ausgerichteten Gesamtsystem, bestehend aus mag. Dipol und Sensor-Array.

Die Sensor-Array-Simulation folgt der Aufführung in Algorithmus 1. Zur Generierung von Trainings-/ Testdatensätzen dient die Konfigurationsdatei als Eingabe. Entsprechend der Parametrierung in Tabelle B.1, sind notwendige Kennfelddatensätze initialisiert und Funktionsmodule eingebunden. Für die eingestellte Anzahl von Simulationswinkeln und angegebenen Winkelauflösung, wird ein Rotationsvektor aufgestellt, indem alle Simulationswinkel gleich verteilt sind. Die Simulation fährt alle X, Y, Z Sensor-Array-Positionen im Koordinatenraum ab. Für jede angefahrenene Position wird ein Meshgrid und entsprechender Datensatz mit voller Rotation erzeugt. Für verschiedene Magnetverkippung ist die Konfigurationsdatei anzupassen und die Simulation zu wiederholen.

Algorithmus 1 : Sensor-Array-Simulation

Input : Konfigurationsdatensatz

Output : MAT-Dateipfad

Result : Sensor-Array-Datensätze (Training/ Test)

1. Laden der Konfigurierung \leftarrow Tabelle B.1;

2. Laden der Kennfelder \leftarrow Anhang A;

3. Initialisierung Simulationssparameter \leftarrow Tabelle B.1;

4. Initialisierung Rotation \leftarrow Gleichung 2.20, Gleichung 2.21;

5. Initialisierung Kennfelder \leftarrow Gleichung A.1;

6. Initialisierung Dipol-Rotationsmomente \leftarrow Gleichung 2.18;

7. Initialisierung Sensor-Array-Positionen \leftarrow Tabelle B.1;

8. Initialisierung Dipol-Feldnormierung \leftarrow Betragkonstante in Gleichung 2.22;

10. Speicherallokation f. Ergebnisse;

11. Anlegen Metadaten (Info) und Simulationsdaten (Data) Structs;

12. for z in Z -Positionsvektor **do**

for x in X -Positionsvektor **do**

for y in Y -Positionsvektor **do**

 Update Info-Struct (Positionsdaten);

 Initialisierung Sensor-Array-Meshgrid \leftarrow Gleichung 2.9;

for \vec{m}_i in Momentenmatrix **do**

 Normierte Dipol-Feldberechnung auf Meshgrid \leftarrow Gleichung 2.22;

 Kennfeld-Mapping interp2(Nearest-Neighbor) \leftarrow Abbildung B.1;

end

 Update Data-Struct (Ergebnisse);

 Speichern Info- und Data-Struct in Ziel-MAT-Datei;

 Ausgabe MAT-Dateipfad;

end

end

end

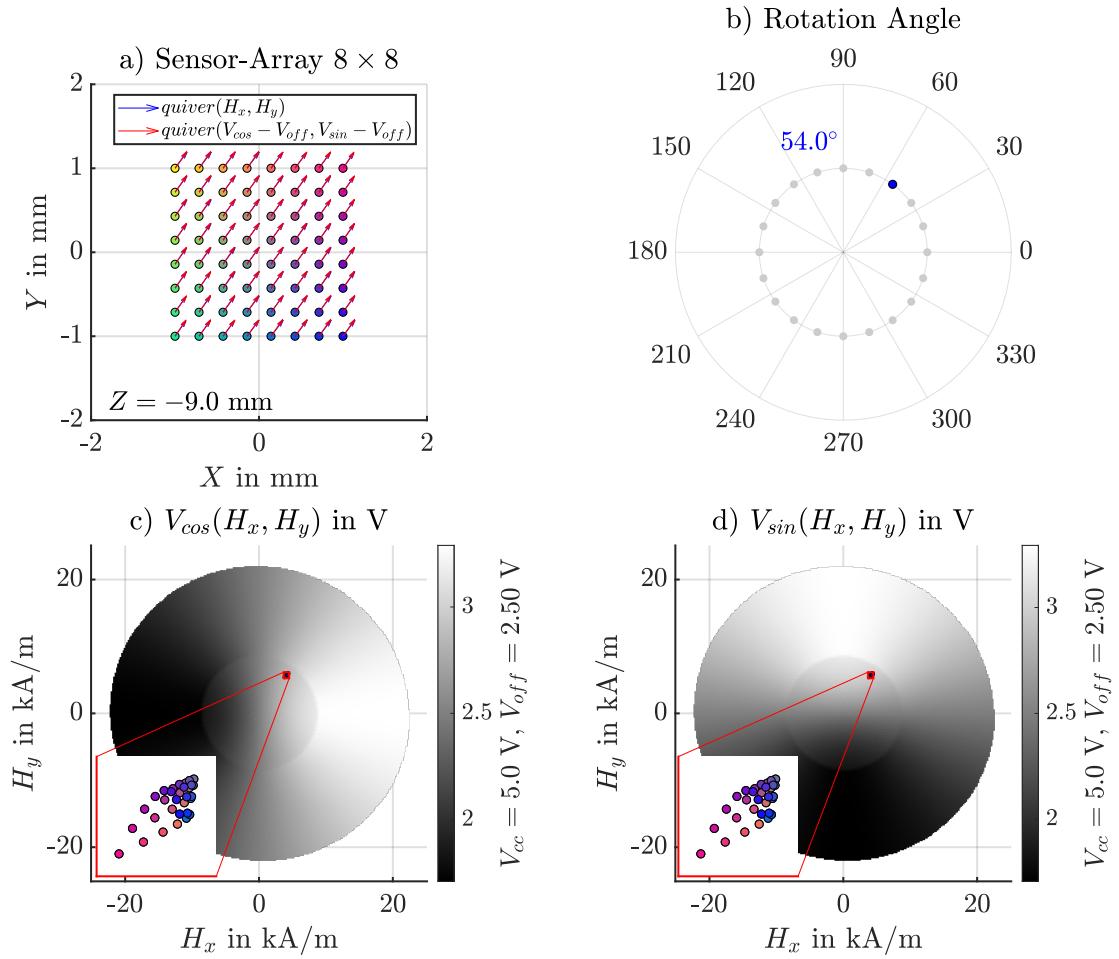


Abbildung B.1: Kennfeld-Mapping. Entnahme von Referenzspannungen aus Sensor-Kennfelder, gezeigt für einen beliebigen Simulationswinkel. In a) Meshgrid des Sensor-Arrays mit Grundposition $(0, 0, -9)^T \text{ mm}$ relativ zum Koordinatenursprung (mag. Dipol). Simulation ohne Dipol-Verkippung. b) Simulation für 20 gleich verteilte Winkel, gezeigt ist der vierte Winkel bei 54° . Simulationsparameter sind wie in Tabelle B.1 eingestellt. Magnet und Array-Position sind ideal konfiguriert. Ersichtlich anhand des eng gegliederten Feldstärken-Mappings auf c) Cosinus-Kennfeld und d) Sinus-Kennfeld. Die simulierten Feldstärken für alle Sensor-Pixel-Koordinaten, liegen innerhalb des linearen Kennfeldarbeitsbereich. Die Kennfelder sind entsprechend Betriebsspannungsparametrierung in V umgerechnet.

Abbildung B.1 zeigt das Mapping, auf TMR-Sensor-Kennfeldern (Anhang A), für prozessierte H_x - und H_y -Feldstärken. Die Simulation ist mit der Standardparametrierung aus Tabelle B.1 ausgeführt worden. Hier am Beispiel für 20 Simulationswinkel. Gezeigt ist der vierte Winkel bei 54° . Die errechneten Feldstärken sind auf die Kennfelder projiziert und Spannungswerte mittels Matlab-2D-Interpolation für Nearest-Neighbor entnommen. Abbildung B.2 zeigt, das nochmals für 720 Winkel und vier Sensor-Pixel. Es sind die Eck-Pixel angezeigt. Da sich der mag. Dipol ohne Verkipfung, zentriert über dem Sensor-Array befindet, überlagern sich die Signalverläufe für diagonal gegenüberliegende Sensor-Pixel. Die leicht ellipsenförmige Verlauf der projizierten Feldstärken, der äußeren Pixel, ergibt sich durch den Versatz zur Magnetfeldmitte. Ein Pixel direkt lotrecht zur Magnet-Z-Achse platziert, erzeugt einen optimale Kreisbahn. Eine detaillierte Beschreibung des Funktionsmoduls ist in Unterabschnitt G.4.1 einzusehen.

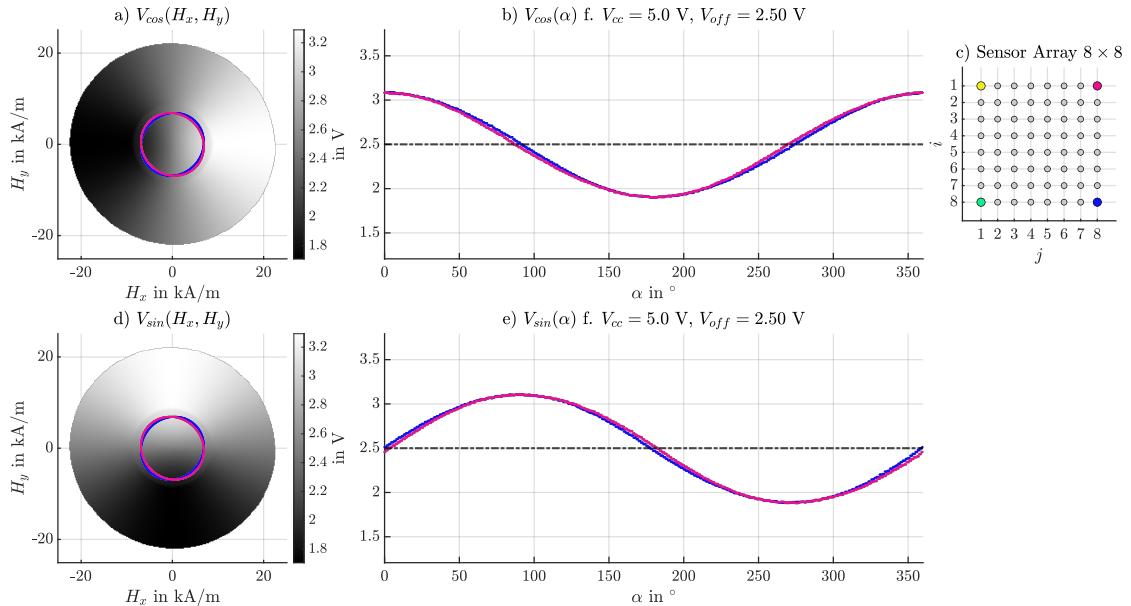


Abbildung B.2: Sensor-Array-Datensatz Teilansicht. Teilansicht der Simulationsergebnisse, entsprechend der Parametrierung nach Tabelle B.1. Es ist der gesamte Simulationsdurchlauf für die Eck-Sensor-Pixel c) gezeigt. a) und d) zeigen das Kennfeld-Mapping, für korrespondierende Feldstärken, bei einer vollständigen Dipol-Drehung. b) und e) stellt die Rotation aufgetragen über alle Simulationswinkel dar. Die sich diagonal gegenüberliegenden Sensor-Pixel, überlagern sich in den Darstellungen a), b), d) und e). Das entspricht der Kreuzsymmetrieeigenschaft Dipol-Magnetfeldes. Das Sensor-Array ist zentriert zum Dipol ausgerichtet. Dipol ist nicht verkippt. Grafik nachempfunden aus [15]

C Gauß-Prozess-Regression

Implementierung 0.0.2 26.04.2021

Im Anhang befindet sich die Beschreibung für die Regression mittels Gauß'scher Prozesse. Die implementierten Mechanismen sind auf die Sensor-Array-Simulation Anhang B angepasst. Es wird sich an den Anforderungen für ein TMR-Sensor-Array Abschnitt 2.5 orientiert. Eine Standardparametrierung für die Simulationsdurchführung ist in Tabelle C.1 einzusehen. Die Notation ist der kompakten Schreibweise für Gauß'sche Prozesse angepasst [3]. Es zwei Kernel implementiert. Als erster der Kernel aus den Vorarbeiten [18][19] und darauf aufbauend ein zweiter mit angepasster Eingangswertverarbeitung. Beide Kernel besitzen die Fähigkeit zur Mittelwert freien und Polynom gestützten Regression bzw. Vorhersage [3]. Das Regressionsmodell kann ganze Sensor-Array-Datensätze verarbeiten. Es sind verschiedene Qualitätskriterien implementiert, die Aussagen zur Modellgenauigkeit, Vorhersagesicherheit und Generalisierung treffen. Die Trainingsphase für das Regressionsmodell wird durch Algorithmus 7 durchgeführt. Eine anschließende Arbeitsphase ist durch Algorithmus 6 umgesetzt. Einen Überblick über die Gesamtsoftware, in der dieser Teil ein Modul einnimmt, ist im Unterabschnitt G.4.2 einzusehen. Der Anhang besitzt einen Glossar ähnlichen Charakter und soll als Schnellnachschlagewerk unterstützen.

Parametergruppe	Parameter	Wert	Einheit	Kurzbeschreibung
GPROptions	kernel	'QFCAPX'	char	Kernel-Funktion-Indikator (C.4), ' $QFC \leftarrow d_F^2$ '
	θ	(1, 1)	-	Kernel-Parametervektor θ (C.5)
	σ_f^2 -Bounds	(0.1, 100)	-	Parameter-Bounds θ_1 f. Algorithmus 5
	σ_l -Bounds	(0.1, 100)	-	Parameter-Bounds θ_2 f. Algorithmus 5
	σ_n^2	$1 \cdot 10^{-6}$	-	Rauschniveau, Rauschaufschaltung (C.6)
	σ_n^2 -Bounds	($1 \cdot 10^{-8}, 1 \cdot 10^{-4}$)	-	Parameter-Bounds σ_n^2 f. Algorithmus 7
	OptimRuns	30	-	Durchlaufanzahl f. Algorithmus 7
	SLL	'SLLA'	char	Verlust-Indikator f. Winkel (A)/ R (Radius) Algorithmus 7
	mean	'zero'	char	Indikator Mittelwertpolynom Ein ('poly')/ Aus ('zero')
	polyDegree	1	-	Grad des Mittelwertpolynoms wenn mean = 'poly'

Tabelle C.1: Gauß-Prozess-Regression-Simulationsparameter. Default-Parameter für die Prozessierung von Simulationsergebnissen aus der Sensor-Array-Simulation Anhang B.

C.1 Modellinitialisierung

Die Modellinitialisierung zur Gauß-Prozess-Regression erfolgt nach Algorithmus 2. Dabei sind Modellparametrierung über einen Konfigurationsdatensatz Tabelle C.1 zu laden. Ebenfalls sind alle gewählten Trainingsdaten, mit dazugehörigen Simulationswinkel $X \mapsto \alpha_{Ref}$, in die Initialisierung einzuspeisen. Das Modell wird Schritt für Schritt in einem Struct aufgebaut. Dabei sind bestimmte Funktionalitäten, entsprechend der gewählten Konfigurierung, in Funktions-Handles zugewiesen. So sind die Schritte 2, 4, 5 und 7 als Funktions-Handles umgesetzt. Das verringert den Speicheraufwand und benötigte Rechenergebnisse können dynamisch bei Bedarf erzeugt werden. Nach der Initialisierung müssen, Modellkonfiguration aus Schritt 1, die Regressionsziele aus Schritt 3, die L -Matrix aus Schritt 8 und die Regressionsgewichte aus Schritt 12 als gespeicherte Werte, für die Vorhersage nach Algorithmus 6 vorliegen. Die Modellkonfiguration aus Schritt 1 und die berechneten Modellplausibilitäten aus Schritt 13 sind für die Modelloptimierung nach Algorithmus 5 entscheidend. Das Modell wird in der Optimierung z.T. reinitialisiert. Die einzelnen Initialisierungsschritte sind nachfolgend zusammengefasst aufgeführt. Es ist ein mathematisch Bezug zur Implementierung in Anhang G und Notation nach Fachliteratur [3] vorgenommen worden.

Algorithmus 2 : Modellinitialisierung mit konst. Trainingsdaten und Parametern

Input : Konfigurationsdatensatz, Trainingsdatensatz $X \mapsto \alpha_{Ref}$

Result : Regressionsmodell mit Fähigkeit zur Datensatzverarbeitung aus Anhang B

1. Initialisierung Modellkonfiguration \leftarrow Tabelle C.1;
 2. Initialisierung X , α und X -Formatierung \leftarrow Gleichung C.1, Gleichung C.2;
 3. Initialisierung Regressionsziele \leftarrow Gleichung C.3;
 4. Initialisierung Kernel-Funktion \leftarrow Gleichung C.4;
 5. Initialisierung Basis-Funktion \leftarrow Gleichung C.8;
 6. Berechnung $K(X, X|\theta) \leftarrow$ Algorithmus 3;
 7. Rauschaufschaltung $K_y \leftarrow$ Gleichung C.6;
 8. Cholesky-Zerlegung von K_y zu L u. Berechnung $\log |K_y| \leftarrow$ Gleichung C.7;
 9. Initialisierung Mittelwertpolynome \leftarrow Gleichung C.9;
 10. Berechnung Polynomkoeffizienten \leftarrow jeweils Algorithmus 4 f. Gleichung C.10;
 11. Initialisierung Mittelwertfunktion \leftarrow Gleichung C.11;
 12. Berechnung Regressionsgewichte \leftarrow Gleichung C.12;
 13. Berechnung Modellplausibilität \leftarrow Gleichung C.13;
-

Der Trainingsdatensatz ist definiert nach der kompakten Notation aus Rasmussen und Williams[3]. Ein Trainingsdatensatz X beinhaltet alle Referenzwinkelstellungen α_i mit $X_i \mapsto \alpha_i$, nach der Beschreibung in Abschnitt 2.5 mit $X_{cos,i} = A_x$ und $X_{sin,i} = A_y$. Für die Implementierung mit Gleichung 2.7 müssen alle Trainingsdatenmatrizen normiert sein, sodass sich Vektoren als Trainingsdaten abbilden, siehe Gleichung C.1.

$$X = [X_1, \dots, X_{N_{Ref}}] \quad \text{f. } i = 1, 2, 3, \dots, N_{Ref} \quad (\text{C.1})$$

$$X_i = \begin{cases} [X_{cos,i}, X_{sin,i}] & \text{f. } d_F^2 \text{ (2.13)} \\ [\|X_{cos,i}\|_F, \|X_{sin,i}\|_F] & \text{f. } d_E^2 \text{ (2.7)} \end{cases}$$

$$X_i \mapsto \alpha_i$$

Der Testdatensatz ist analog zum Trainingsdatensatz definiert [3]. Ein Testdatensatz X_* repräsentiert einen Testwinkel mit $X_* \mapsto \alpha_*$. Auch hier gilt, wie für Trainingsdatensätze $X_i \mapsto \alpha_i$, die Umschreibung nach Abschnitt 2.5 mit $X_{cos*} = A_x$ und $X_{sin*} = A_y$. Jeweils für die gewählte Implementierung ist auch hier eine Eingangsverarbeitung der Datensätze notwendig Gleichung C.2, sodass die Implementierung nach Gleichung 2.7 mit Skalaren statt Matrizen arbeitet.

$$X_* = \begin{cases} [X_{cos*}, X_{sin*}] & \text{f. } d_F^2 \text{ (2.13)} \\ [\|X_{cos*}\|_F, \|X_{sin*}\|_F] & \text{f. } d_E^2 \text{ (2.7)} \end{cases} \quad (\text{C.2})$$

$$X_* \mapsto \alpha_*$$

Regressionsziele sind als Spaltenvektoren nach [3] wie in Gleichung C.3 definiert. Die Besonderheit hier sind zwei Zielvektoren statt einer, wie in der Fachliteratur [3] angegeben. Abstrahiertes Regressionsziel ist der Einheitskreis, daher ergeben sich einfache Sinoide aus den Referenzwinkeln in Gleichung C.3.

$$y_{cos} = (\cos \alpha_i, \dots, \cos \alpha_{N_{Ref}})^T \quad \text{f. } i = 1, 2, 3, \dots, N_{Ref} \quad (\text{C.3})$$

$$y_{sin} = (\sin \alpha_i, \dots, \sin \alpha_{N_{Ref}})^T$$

Kernel-Funktion als Kernelement des Regressionsverfahren für Gauß'sche Prozesse [3]. Es sind zwei Versionen nach gleichen Vorbild der fraktalen Kovarianz [18][19] implementiert. Die Implementierung mittels Gleichung 2.13 resultiert aus den Vorarbeiten der Arbeitsgruppe Sensorik und stellt die genaue Lösung dar und arbeitet direkt mit Matrizen als Trainingsdaten. Die Implementierung nach mit Gleichung 2.7 ist innerhalb dieser Arbeit entstanden und bedingt ein vorab Prozessieren der Trainingsdaten zu Vektoren. Ebenfalls müssen weitere Testdaten eingangs zu skalaren verarbeitet werden. Dazu wird die Frobenius-Norm aus Gleichung 2.10 verwendet. Die Implementierung nach Gleichung 2.7 folgt dem Beispiel, aus der Veröffentlichung von Lang, Dunkley und Hirche[7], für die Anwendung der Gauß-Prozess-Regression auf Themenfeld der Robotik.

$$k(X_i, X_j) = \begin{cases} \frac{a}{b+d_F^2 \langle X_i, X_j \rangle} & \text{f. } d_F^2 \text{ (2.13)} \\ \frac{a}{b+d_E^2 \langle X_i, X_j \rangle} & \text{f. } d_E^2 \text{ (2.7)} \end{cases} \quad (\text{C.4})$$

mit $i, j = 1, 2, 3, \dots, N_{Ref}$

Kernel-Parameter für die Kovarianz- oder Kernel-Funktion bilden sich die Funktionsparameter, wie in Gleichung C.5 beschrieben. Bei der Parameteridentifizierung ist das Auslöschkriterium für gültige Kovarianzfunktionen elementar [3]. Dabei ist der Fall abzudecken, dass die Abstandsfunktion der Kovarianz zu null wird, wenn Datensätze mit sich selbst prozessiert werden. In diesem Fall muss die Kovarianzfunktion σ_f^2 ergeben.

$$a = \sigma_f^2 \cdot 2\sigma_l^2 \quad b = 2\sigma_l^2 \quad \theta = [\sigma_f^2, \sigma_l] \quad (\text{C.5})$$

Kovarianzmatrix als Autokorrelationsergebnis, ist für alle bereitgestellten Trainingsdaten untereinander mit Algorithmus 3 zu berechnen [3]. Das Ergebnis, in quadratischer Matrixform, definiert das Verhalten des Gesamtsystems über gemessene Einzelabstände der Trainingsdaten zueinander. Es bedeutet, dass für ein Sensor-Array auf Basis des TMR-Sensors [11] in Drehwinkelapplikation Abbildung 2.9, die 360° Periodizität ersichtlich sein muss. Was eine Maxima-Diagonale von links nach rechts und annähernde Max-Werte in den beiden übrigen Ecken der Matrix impliziert.

Algorithmus 3 : Berechnung der Kovarianzmatrix $K(X, X|\theta)$

Input : Kernel-Funktion $k(X_i, X_j)$, Trainingsdaten X , Kernel-Parameter θ

Result : K -Matrix $N_{Ref} \times N_{Ref}$

```

1. Initialisierung Parameter  $(a, b) \leftarrow \theta$  Gleichung C.5;
2. for  $i = 1, 2, 3, \dots, N_{Ref}$  do
    for  $j = 1, 2, 3, \dots, N_{Ref}$  do
         $| K_{i,j} = k(X_i, X_j) \leftarrow$  Gleichung C.4;
    end
end

```

Rauschaufschaltung durch konstantes Rauschniveau σ_n^2 und minimaler Anhebung der Kovarianzmatrix-Diagonalen, für verrauschte bzw. fehlerbehaftete Regression [3]. Entsprechend der Fachliteratur ist die Kovarianzmatrix inklusive additives Rauschen als K_y bezeichnet [3].

$$K_y = K(X, X|\theta) + \sigma_n^2 I \quad (\text{C.6})$$

Cholesky-Zerlegung K_y als Ansatz zum Lösen der Regressionsmechanismen. Dem Regressionsverfahren zugrundeliegenden Lösungen der Wahrscheinlichkeitsdichte-Integrale sind über Matrix- und Vektormultiplikation mit Inversen Matrizen in linearen Gleichungssystemen gelöst [3]. Für das Lösen der Gleichungssysteme mit inversen Matrizen, ist die Zerlegung der nicht inversen Matrix in eine untere Dreiecksmatrix möglich. Die Cholesky-Zerlegung schafft entsprechenden Repräsentant L für die Matrix K_y . Die logarithmierte Determinante ist mit Gleichung C.7 zu berechnen. Matrizen müssen symmetrisch und positiv definit sein, um die Cholesky-Zerlegung anwenden zu können [3].

$$LL^T = K_y \quad (C.7)$$

$$\log|K_y| = 2 \sum_{i=1}^{N_{Ref}} \log L_{i,i}$$

Es sollen damit Gleichungssysteme wie $Ax = b \Leftrightarrow x = A^{-1}b$ über zerlegten Repräsentanten gelöst sein $Ly = b \Leftrightarrow L^Tx = y$. Als Notation für die notwendige Lösung der linearen Gleichungssystem ist der Backslash-Operator genutzt $x = L^T \backslash (L \backslash b)$ [3]. Der Backslash-Operator steht ebenfalls in Matlab zur Verfügung.

Basis-Funktion als Aufbaufunktion für Polynome aus Trainings- und Testdaten. Das Regressionsverfahren mittels Gauß'scher Prozesse kann in zwei Ausführungen betrieben werden. Die erste ist eine Regression ohne weitere Mittelwerte als Regressionshilfe. In zweiter Ausführung können Mittelwerte der Daten z.B. über Polynomfindung gebildet sein. Dafür bedingt es eine Basis-Funktion nach Gleichung C.8, die entsprechend der Kernel-Funktion und resultierende Datenformate Polynome bildet [3]. In der Implementierung sind Polynome ersten Grades verwendet. Was einer Offset- und Amplitudenkorrektur der Trainings- und Testdaten entspricht. Die in Gleichung C.8 gezeigten Funktionen müssen, jeweils für beide Cosinus und Sinus Datentypen gebildet werden und beziehen sich hier in der Darstellung für einen einzigen Simulationswinkel $X_* \mapsto \alpha_*$.

$$h_{\cos}(X_{\cos*}) = \begin{cases} 0 & \text{f. } m_{\cos}(X_{\cos*}) = 0 \\ (1, \|X_{\cos*}\|_F, \|X_{\cos*}\|_F^2, \dots)^T & \text{f. } d_F^2 \text{ (2.13), } m_{\cos}(X_{\cos*}) \neq 0 \\ (1, X_{\cos*}, X_{\cos*}^2, \dots)^T & \text{f. } d_E^2 \text{ (2.7), } m_{\cos}(X_{\cos*}) \neq 0 \end{cases} \quad (\text{C.8})$$

$$h_{\sin}(X_{\sin*}) = \begin{cases} 0 & \text{f. } m_{\sin}(X_{\sin*}) = 0 \\ (1, \|X_{\sin*}\|_F, \|X_{\sin*}\|_F^2, \dots)^T & \text{f. } d_F^2 \text{ (2.13), } m_{\sin}(X_{\sin*}) \neq 0 \\ (1, X_{\sin*}, X_{\sin*}^2, \dots)^T & \text{f. } d_E^2 \text{ (2.7), } m_{\sin}(X_{\sin*}) \neq 0 \end{cases}$$

Mittelwertpolynome bauen sich über die Basis-Funktionen aus Gleichung C.8 für Trainingsdaten auf. Es resultieren Matrizen [3], deren erste Reihe gleich eins ist und jede weitere Reihe mit entsprechenden Exponenten für die Polynomgenerierung versehen ist. Die Polynombildung ist jeweils für beide Cosinus- und Sinus-Datensätze durchzuführen, wenn die Mittelwertbildung aktiv ist.

$$H_{\cos}(X_{\cos}) = \begin{cases} 0 & \text{f. } m_{\cos}(X_{\cos}) = 0 \\ [h_{\cos}(X_{\cos,i}), \dots, h_{\cos}(X_{\cos,N_{\text{Ref}}})] & \text{f. } m_{\cos}(X_{\cos}) \neq 0 \end{cases} \quad (\text{C.9})$$

$$H_{\sin}(X_{\sin}) = \begin{cases} 0 & \text{f. } m_{\sin}(X_{\sin}) = 0 \\ [h_{\sin}(X_{\sin,i}), \dots, h_{\sin}(X_{\sin,N_{\text{Ref}}})] & \text{f. } m_{\sin}(X_{\sin}) \neq 0 \end{cases}$$

jeweils für alle $X_{\cos} = [X_{\cos,i}, \dots, X_{\cos,N_{\text{Ref}}}]$ und

alle $X_{\sin} = [X_{\sin,i}, \dots, X_{\sin,N_{\text{Ref}}}]$

mit $i = 1, 2, 3, \dots, N_{\text{Ref}}$

Polynomkoeffizienten zur Mittelwertbildung über gebildet Polynome nach Gleichung C.8 und Gleichung C.9, sind benötigte Polynomkoeffizienten nach Gleichung C.10 zu berechnen [3]. Zur Berechnung der Koeffizienten sind mehrere inverse Matrix-Produkte verschachtelt zu lösen.

$$\beta_{cos} = \begin{cases} 0 & \text{f. } m_{cos}(X_{cos}) = 0 \\ (H_{cos}K_y^{-1}H_{cos}^T)^{-1}H_{cos}K_y^{-1}y_{cos} & \text{f. } m_{cos}(X_{cos}) \neq 0 \end{cases} \quad (\text{C.10})$$

$$\beta_{sin} = \begin{cases} 0 & \text{f. } m_{sin}(X_{sin}) = 0 \\ (H_{sin}K_y^{-1}H_{sin}^T)^{-1}H_{sin}K_y^{-1}y_{sin} & \text{f. } m_{sin}(X_{sin}) \neq 0 \end{cases}$$

jeweils für alle $X_{cos} = [X_{cos,i}, \dots, X_{cos,N_{Ref}}]$ und

alle $X_{sin} = [X_{sin,i}, \dots, X_{sin,N_{Ref}}]$

mit $i = 1, 2, 3, \dots, N_{Ref}$

Zur Bewältigung des Problems ist Algorithmus 4 implementiert worden und jeweils für beide Cosinus- und Sinus-Polynome aus Gleichung C.9 durchzuführen. Die Polynom- und Koeffizientenbestimmung entfällt, wenn die Mittelwertbildung ausgeschaltet ist.

Algorithmus 4 : Berechnung der β Polynomkoeffizienten aus Gleichung C.10

Input : Polynommatrix H , Untere Dreiecksmatrix $L(K_y)$, Regressionsziel y

Result : β -Koeffizienten

1. $a_0 \leftarrow$ Lösen von $K_y^{-1}y$;
 $a_0 = L^T \setminus (L \setminus y)$;
 2. $A_1 \leftarrow$ Lösen von $HK_y^{-1}H^T$;
for j -te Spalte in H^T **do**
| $V_j = L \setminus H_j^T$;
| **end**
| $A_1 = V^T V$;
 3. $L_1 \leftarrow$ cholesky(A_1);
 4. $A_2 \leftarrow$ Lösen von $A_1^{-1}H$;
for j -te Spalte in H **do**
| $V_j = L_1^T \setminus (L_1 \setminus H_j)$;
| **end**
| $A_2 = V$;
 5. $\beta = A_2 \cdot a_0$;
-

Mittelwertfunktionen für die Regression setzen sich aus gebildeten Polynomen und den bestimmten Polynomkoeffizienten nach Gleichung C.11 zusammen [3]. Für alle Trainingsdaten mittels Polynommatrizen und für einzelne Testdaten über die Basis-Funktion. Bei eingeschalteter Mittelwertbildung, bildet sich das Regressionsergebnis über die Summe aus Mittelwertberechnung und Stützwertsumme [3]. Die Mittelwertrechnung ist für beide Cosinus- und Sinus-Datensätze umzusetzen.

$$m_{cos}(X_{cos(*)}) = \begin{cases} 0 & \text{f. mittelwertfreie Regression} \\ H_{cos}(X_{cos}) \cdot \beta_{cos} & \text{f. Trainingsdaten } X_{cos} \\ h_{cos}(X_{cos*}) \cdot \beta_{cos} & \text{f. Testdaten } X_{cos*} \end{cases} \quad (\text{C.11})$$

$$m_{sin}(X_{sin(*)}) = \begin{cases} 0 & \text{f. mittelwertfreie Regression} \\ H_{sin}(X_{sin}) \cdot \beta_{sin} & \text{f. Trainingsdaten } X_{sin} \\ h_{cos}(X_{sin*}) \cdot \beta_{sin} & \text{f. Testdaten } X_{sin*} \end{cases}$$

jeweils für alle $X_{cos} = [X_{cos,i}, \dots, X_{cos,N_{Ref}}]$ und
 alle $X_{sin} = [X_{sin,i}, \dots, X_{sin,N_{Ref}}]$
 mit $i = 1, 2, 3, \dots, N_{Ref}$

Regressionsgewichte oder Stützwerte für die Vorhersage beider Sinoide sind jeweils, in Abhängigkeit der dazugehörigen Regressionsziele und Mittelwerte, über inverse Matrix-Produkt aus K_y^{-1} und das Residual aus Ziel und Mittelwert zu bilden. Gleichung C.12 beschreibt die Lösung des resultierenden Gleichungssystem über die untere Dreiecksmatrix L der Kovarianzmatrix K_y [3]. Die Gewichtsbildung veranschaulicht am besten den Gesamtlauf des Verfahrens. Es sind zwei unterschiedliche Regressionen, jeweils für Cosinus- und Sinus-Funktionen durchzuführen. Dabei stützen sich beide Regressionen auf eine gemeinsame Kovarianzbewertung, der zugrundeliegenden Trainingsdatensätze [19]. Die Kovarianzmatrix stellt somit die vektorielle und orthogonale Kopplung der Daten her und impliziert ihre gegenseitige Abhängigkeit.

$$\begin{aligned}\alpha_{cos} &= K_y^{-1} \cdot (y_{cos} - m_{cos}(X_{cos})) \\ &= L^T \backslash (L \backslash (y_{cos} - m_{cos}(X_{cos})))\end{aligned}\tag{C.12}$$

$$\begin{aligned}\alpha_{sin} &= K_y^{-1} \cdot (y_{sin} - m_{sin}(X_{sin})) \\ &= L^T \backslash (L \backslash (y_{sin} - m_{sin}(X_{cos})))\end{aligned}$$

jeweils für alle $X_{cos} = [X_{cos,i}, \dots, X_{cos,N_{Ref}}]$ und
alle $X_{sin} = [X_{sin,i}, \dots, X_{sin,N_{Ref}}]$
mit $i = 1, 2, 3, \dots, N_{Ref}$

Modellplausibilitäten oder Regressionsevidenzen, sind entsprechend der Regressionsausrichtung, über Residuale und Regressionsgewichte in Gleichung C.13 zu bilden [3]. Jeweils wieder für beide Cosinus- und Sinus-Datensätze. Die so aufgestellten Plausibilitäten bewerten den Regressions-Fit in Bezug auf die Trainingsdaten. In der Fachliteratur [3] sind diese auch als Logarithmic-Marginal-Likelihoods bezeichnet. Sie bieten einen Indikator für den Daten-Fit, der < 0 wird für eine schlechte Anpassung, ≈ 0 ist bei mäßiger Anpassung und > 0 ist für eine gute Modellanpassung. Dabei sind Werte > 30 als sehr gute Anpassung Modellanpassung für jeweilige Sinoide zu interpretieren. Bei Plausibilitäten größer > 60 und zu eng gewählter Parameter-Bounds, kann sich ein zu Starker Fit auf die Trainingsdaten einstellen. Das wird als Overfitting bezeichnet. Ein so über parametrisiertes Modell, verliert dabei seine Fähigkeit zur Generalisierung und liefert nur für die Trainingsdaten selber valide Ergebnisse. Testdaten, die von den Trainingsdaten abweichen, können somit nicht mehr korrekt prozessiert werden. Die Interpretation der Likelihoods ist aus der Fachliteratur [3] entnommen und für empfohlene Werte empirisch bestimmt worden. Die einzelnen Plausibilitäten müssen ungefähr gleich groß sein, andernfalls besteht ein Ungleichgewicht in der Vorhersage. Resultierende Ergebnisse sind dann im Winkel und Radius verfälscht. Hergestellt wird das Gleichgewicht durch die Kovarianzkopplung, mittels gemeinsamer Kovarianzmatrix.

$$\log p(y_{cos}|X_{cos}) = -0,5 \left((y_{cos} - m_{cos}(X_{cos}))^T \alpha_{cos} + \log |K_y| + N_{Ref} \log 2\pi \right) \quad (\text{C.13})$$

$$\log p(y_{sin}|X_{sin}) = -0,5 \left((y_{sin} - m_{sin}(X_{sin}))^T \alpha_{sin} + \log |K_y| + N_{Ref} \log 2\pi \right)$$

jeweils für alle $X_{cos} = [X_{cos,i}, \dots, X_{cos,N_{Ref}}]$ und
 alle $X_{sin} = [X_{sin,i}, \dots, X_{sin,N_{Ref}}]$
 mit $i = 1, 2, 3, \dots, N_{Ref}$

C.2 Modelloptimierung

Die Optimierung bezieht sich auf ein fertig initialisiertes Modell nach Algorithmus 2. Dieses muss dafür einen vollständigen Parametersatz Tabelle C.1 inklusive Bounds beinhalten. Ebenfalls müssen alle Trainingsdaten entsprechend der gewählten Implementierung im Modell enthalten sein. Die Optimierung in Algorithmus 5 ist mittels Fmincon-Funktion (Matlab) implementiert und nutzt einen Sequential-Quadratic-Programming-Algorithmus um das Minimum-Kriterium aus Gleichung C.14 zu steuern. Das Modell wird im Prozess so lange reinitialisiert, bis keine graduelle Änderung des Kriteriums mehr festgestellt werden können. Kritisch im Optimierungsverfahren sind dabei, die zu setzenden Parameter-Bounds für θ . Sind die Grenzen des Suchfeldes zu eng gesetzt, kann das Minimum nicht erreicht werden. Der Algorithmus wird dann die Bounds selbst als Ergebnis liefern. Sind die Bounds zu weit abgesteckt ist es theoretisch möglich, dass das Minimum nicht vor Abbruch gefunden werden kann. In der Regel findet sich das Minimum, mit der getroffenen Implementierung, nach 6 – 24 Durchläufe. Das ist empirisch durch ausgegebene Grafiken beobachtet worden.

Algorithmus 5 : Modelloptimierung über Fmincon-Funktion f. $\sigma_n^2 = \text{konst.}$

Input : Modell inkl. $X, \theta, \sigma_n^2 + \text{Bounds} \leftarrow \text{Algorithmus 2}$

Result : Optimiertes Modell mit neuen Kernel-Parameter $\theta|\sigma_n^2$, f. $\sigma_n^2 = \text{konst.}$

1. Initialisierung Fmincon-Funktion;
 2. Initialisierung Parameter-Bounds \leftarrow Modell-Bounds Tabelle C.1;
 3. Initialisierung Fmincon-Startwert \leftarrow Model-Kernel-Parameter Tabelle C.1;
 4. Initialisierung Min-Kriterium $\tilde{R}_{\mathcal{L}} \leftarrow$ Gleichung C.14;
 5. **while** $\neg(\tilde{R}_{\mathcal{L}} = \text{konst. f. 7 Iterationen}) \wedge (\min \neq \tilde{R}_{\mathcal{L}})$ **do**
 - | Zuweisung innerhalb Parameter-Bounds $\theta \leftarrow$ Fmincon-Funktion;
 - | Modell-Teilreinitialisierung \leftarrow Algorithmus 2, Schritte 6. bis 13.;
 - | Berechnung $\tilde{R}_{\mathcal{L}} \leftarrow$ Gleichung C.14;
 - end**
 6. Speichern $\theta \leftarrow$ Fmincon-Funktion;
 7. Modell-Teilreinitialisierung \leftarrow Algorithmus 2, Schritte 6. bis 13.;
-

Min-Kriterium als zusammengesetztes Kriterium aus den einzelnen Modellplausibilitäten für die Cosinus- und Sinus-Vorhersage. Der Bildungsansatz ist aus einem Regressionsproblem der Computer-Vision [5] adaptiert und nach dem Leitwerk zur Verfahrensentwicklung [3] angepasst worden. Die Findung optimaler Kovarianz- bzw. Kernel-Parameter, kann nach Gleichung Gleichung C.14 vorgenommen werden. Dafür sind Modellplausibilitäten, für die einzelnen Sinoiden, als Funktion von Kernel-Parametern zu betrachten. Das Kriterium ergibt sich, als negative Summe der einzelnen Plausibilitäten Gleichung C.13. Das aufgestellte Minimierungsproblem ist bei einem konstanten Rauschniveau σ_n^2 und verschiedenen Kernel-Parameter θ zu untersuchen.

$$\theta|\sigma_n^2 = \arg \min_{\theta} \tilde{R}_{\mathcal{L}}(\theta|\sigma_n^2) \quad \text{f. } \sigma_n^2 = \text{konst.}$$

(C.14)

$$\tilde{R}_{\mathcal{L}}(\theta|\sigma_n^2) = -(\log p(y_{cos}|X_{cos}, \theta, \sigma_n^2) + \log p(y_{sin}|X_{sin}, \theta, \sigma_n^2))$$

C.3 Modellvorhersagen

Die Implementierung für Winkelvorhersagen, auf Grundlage von Datensätzen der Sensor-Array-Simulation Anhang B, läuft nach Algorithmus 6 ab. Der Algorithmus zeigt, die Vorhersage für eine Winkelstellung und ist für mehrere Winkel zu wiederholen. Die geschriebene Software in ??, kann einen Winkelsatz oder komplette Datensätze, bestehend aus mehreren Winkelsätzen prozessieren. Für letzteres stehen Ergebnisse als Vektoren zur weiteren Analyse oder Optimierung bereit.

Algorithmus 6 : Modellvorhersage f. Sinoide eines Testwinkel mit $X_* \mapsto \alpha_*$

Input : Modell inkl. X, θ, σ_n^2 , Testdatensatz (inkl. Testwinkel α_*) $X_* \mapsto \alpha_*$

Result : Sinoide $\bar{f}_{cos*}, \bar{f}_{sin*}$, Radius \bar{r}_* , Winkel $\bar{\alpha}$, Sinoide Varianz $\mathbb{V}[\bar{f}_*]$, Sinoide Std.-Abweichung s_* , Konfidenzintervalle $CIA_{95\%}, CIR_{95\%}$, std. log. Verluste ($SLLA, SLLR$)

1. Berechnung Kovarianzvektor $\mathbf{k}_* \leftarrow$ Gleichung C.15;
 2. Berechnung Varianzvorhersage $\mathbb{V}[\bar{f}_*] \leftarrow$ Gleichung C.18;
 3. Berechnung Mittelwertvorhersage $\bar{f}_{cos*}, \bar{f}_{sin*} \leftarrow$ Gleichung C.16;
 4. Berechnung Radius $\bar{r}_*,$ Winkel $\bar{\alpha} \leftarrow$ Gleichung C.17;
 4. Berechnung Std.-Abweichung $s_* \leftarrow$ Gleichung C.19;
 6. Berechnung Konfidenzintervalle $CIA_{95\%}, CIR_{95\%} \leftarrow$ Gleichung C.20;
 7. Berechnung std. log. Verluste ($SLLA, SLLR \leftarrow$ Gleichung C.21);
-

Kovarianzvektor als Regressionsmaß für einen einzigen Testdatensatz mit zugehörigen Simulationswinkel $X_* \mapsto \alpha_*$. Ist der Vektor \mathbf{k}_* nach Algorithmus 3 zu bilden. Er stellt den Vergleichsbezug von Testdaten X_* zu allen Trainingsdaten X her und löst die neue Winkelstellung in Relation zu den Trainingsdaten auf [3]. Der Kovarianzvektor \mathbf{k}_* ist mit aktuellen Modellparametern zu berechnen Gleichung C.15.

$$\mathbf{k}_* = K(X, X_* | \theta) \quad \text{mit Algorithmus 3} \quad (\text{C.15})$$

f. Traingsdaten X und einen Testwinkel $X_* \mapsto \alpha_*$

Mittelwertvorhersage als Mittelwertergebnis für eine Standardnormalverteilung, sind über die Summe aus Mittelwertschätzung Gleichung C.11 und Produkt aus Kovarianzvektor Gleichung C.15 mit Regressionsgewichte Gleichung C.12 zu bilden [3]. Jeweils für beide Funktionen Cosinus und Sinus. Die Systematische Kopplung erfolgt über den gemeinsamen Kovarianzvektor.

$$\begin{aligned}\bar{f}_{cos*} &= m_{cos}(X_{cos*}) + \mathbf{k}_*^T \cdot \alpha_{cos} \\ \bar{f}_{sin*} &= m_{sin}(X_{sin*}) + \mathbf{k}_*^T \cdot \alpha_{sin}\end{aligned}\tag{C.16}$$

Regressierter Winkel und Radius aus der Cosinus- und Sinus-Vorhersage, ergeben sich aus der Anwendungsbeschreibung im Abschnitt 2.1 durch Gleichung C.17.

$$\begin{aligned}\bar{r}_* &= \sqrt{\bar{f}_{cos*}^2 + \bar{f}_{sin*}^2} \quad \text{wie Gleichung 2.2} \\ \bar{\alpha}_* &= \text{atan2}(\bar{f}_{sin*}, \bar{f}_{cos*}) \quad \text{wie Gleichung 2.3}\end{aligned}\tag{C.17}$$

Varianzvorhersage als Korrelation eines Testdatensatz X_* mit sich selbst. Dafür ist der Datensatz X_* Algorithmus 3 zuzuführen und zur Varianz der Vorhersage $\mathbb{V}[\bar{f}_*]$ im Vergleich zur Kovarianzmatrix und Kovarianzvektor mit Gleichung C.18 aufzulösen. Die Varianz $\mathbb{V}[\bar{f}_*]$ gilt jeweils für beide Sinoide Regressionsprozesse [3]. Dieser Fall deckt sich mit dem Auslöschungskriterium für gültige Kovarianzfunktion und kann in diesem Fall durch einen numerischen Fehler $\Delta\epsilon$ leicht vom theoretischen Rechenergebnis abweichen.

$$\begin{aligned}\mathbb{V}[\bar{f}_*] &= K(X_*, X_* | \theta) - \mathbf{k}_*^T K_y^{-1} \mathbf{k}_* \\ &= K(X_*, X_* | \theta) - v^T v \\ &= (\sigma_f^2 + \Delta\epsilon) - v^T v\end{aligned}\tag{C.18}$$

mit $v = L \setminus \mathbf{k}_*$ und $\Delta\epsilon$ numerischer Fehler

Standardabweichung als zugehörige Abweichung der Mittelwertvorhersage für eine Standardnormalverteilung, ergibt sich aus der Varianzvorhersage und dem verwendeten Rauschniveau nach Gleichung C.19 [3]. Die Standardabweichung gilt, jeweils als Abweichung für beide Sinoide als eigenständige und statistische Prozesse. Fehler der einzelnen Prozesse, müssen sich in einer kombinierten Auswertung, durch Addition ihrer Einzelvarianzen s_*^2 , für eine gemeinsame Abschätzung fortpflanzen.

$$s_* = \sqrt{\mathbb{V}[\bar{f}_*] + \sigma_n^2} \quad (\text{C.19})$$

Qualitätskriterien können über die ermittelte Standardabweichung s_* der Einzelregressionen gebildet werden. Dafür muss diese gemäß der Fehlerfortpflanzung für die einzelnen statistischen Prozesse mit dem Faktor $\sqrt{2}$ versehen werden, da sich die Varianz für Cosinus und Sinus zusammensetzt und verdoppelt mit $s_*\sqrt{2} = \sqrt{2(\mathbb{V}[\bar{f}_*] + \sigma_n^2)}$. Konfidenzintervalle für ermittelten Winkel $\bar{\alpha}_*$ und Radius \bar{r}_* , können daher direkt mit $95\% \leftarrow z_{CDF}$ -Faktor für normalverteilte Wahrscheinlichkeiten und kumulativer Dichtefunktion berechnet werden. Der Radikant für die Stichprobenanzahl entfällt, da immer nur ein Testwert prozessiert wird. Es ergeben sich das Konfidenzintervall für Winkel $CIA_{95\%}$ und für Radius $CIR_{95\%}$ nach der Gleichung C.20. Für das Winkelintervall ist die statistische Aussage mit $\arcsin(z_{CDF} \cdot s_*\sqrt{2})$ ins Winkelmaß überführt.

$$\begin{aligned} CIA_{95\%} &= \bar{\alpha}_* \pm \arcsin(z_{CDF} \cdot s_*\sqrt{2}) \quad \text{f. } z_{CDF} = 1,96 \leftarrow 95\% \\ CIR_{95\%} &= \bar{r}_* \pm z_{CDF} \cdot s_*\sqrt{2} \end{aligned} \quad (\text{C.20})$$

Zwei weitere Qualitätskriterien, zur Interpretation der Modellgeneralisierung, können über standardisierte logarithmische Verluste (engl. std. log. loss) berechnet werden [3]. Die Berechnung erfolgt, als Vergleich zwischen Soll- und errechneten Istwerten, unter Berücksichtigung der Fehlerfortpflanzung und Winkelmaß nach Gleichung C.21. Es ergibt sich der Verlust $SLLA$ für Winkel und $SLLR$ für Radius.

Verluste $SLLA$ stehen nur zur Verfügung, wenn Simulations- oder Encoder-Winkel in der Vorhersage mit einbezogen sind. Verluste $SLLR$ für Radius stehen immer zur Verfügung, da mit Einheitskreis als Regressionsziel, der Sollradius gleich eins ist. Ein schlecht generalisiertes Modell liefert positive Verlustwerte > 0 . Eine mäßige Generalisierung liefert Verluste ≈ 0 . Eine gute bis sehr gute Generalisierung liefert strikt Verlustwerte < 0 [3].

$$SLLA = 0,5 \cdot \left(\log(2\pi \arcsin^2(s_*\sqrt{2})) + \frac{(\alpha_* - \bar{\alpha}_*)^2}{\arcsin^2(s_*\sqrt{2})} \right) \quad (\text{C.21})$$

$$SLLR = 0,5 \cdot \left(\log(2\pi(s_*\sqrt{2})^2) + \frac{(1 - \bar{r}_*)^2}{(s_*\sqrt{2})^2} \right)$$

C.4 Modellgeneralisierung

Algorithmus 7 zur Modellgeneralisierung vereinigt die Algorithmen 2 und 5 in sich und nutzt diese in Verbindung mit einem Bayes-Optimierungsverfahren zur Ermittlung des passenden Rauschniveau σ_n^2 . Das Bayes-Optimierungsverfahren ist in Matlab über die BayesOpt-Funktion implementiert und wird mit dem Probier-Algorithmus Improve-Per-2^{nd+} betrieben. Entscheidend bei der Ausführung ist die Durchlaufzahl der Bayes-Optimierung, da sich das Verfahren durch Ausprobieren von σ_n^2 und Vergleich des resultierenden Min-Kriterium Gleichung C.22, Schritt für Schritt der optimalen Lösung annähert.

$$\sigma_n^2 | X_* = \arg \min_{\sigma_n^2} MSLL(\sigma_n^2 | X_*) \quad (\text{C.22})$$

$$MSLL = \begin{cases} \frac{SLLA(X_*)}{N_*} & \text{f. Verluste üb. Winkel} \\ \frac{SLLR(X_*)}{N_*} & \text{f. Verluste üb. Radius} \end{cases}$$

f. alle N_* Testdaten X_* und

Testwinkel $X_* \mapsto \alpha_*$ nach Gleichung C.21

Je nach dem wie die Grenzen der einzelnen Modellparameter gewählt sind, kann das unterschiedlich schnell passieren. Wird der Algorithmus zu früh abgebrochen, ist die optimale Lösung wahrscheinlich nicht gefunden. Daher empfiehlt sich für Anfangsuntersuchungen mit weiten Parameter-Bounds eine Durchlaufzahl ≥ 50 zu wählen.

Algorithmus 7 : Modellgeneralisierung über BayesOpt-Funktion f. alle $X_* \mapsto \alpha_*$

Input : Kofigurationsdatensatz, Trainingsdatensatz X, Testdatensatz X*

Result : Generalisiertes Modell mit optimierten Rauschniveau σ_n^2

1. Initialisierung Modell \leftarrow Algorithmus 2;
 2. Initialisierung Rauschniveau-Bounds \leftarrow Tabelle C.1;
 3. Initialisierung Min-Kriterium $MSLL \leftarrow$ Gleichung C.22;
 4. Initialisierung BayesOpt-Funktion mit Durchlaufzahl \leftarrow Tabelle C.1;
 5. **while** Durlaufzahl nicht erreicht **do**
 - Zuweisung innerhalb Rauschniveau-Bounds $\sigma_n^2 \leftarrow$ BayesOpt-Funktion;
 - Modelloptimierung von 1. mit neuen $\sigma_n^2 \leftarrow$ Algorithmus 5;
 - Berechnung f. alle Testwinkel $MSLL \leftarrow$ Gleichung C.22;
 - Speichern und indizieren von σ_n^2 f. jedes Ergebnis $MSLL$;
 - end**
 6. Entnahme von σ_n^2 bei $\min MSLL$;
 7. Speichern von σ_n^2 in 1.;
 8. Finale Modelloptimierung von 1. \leftarrow Algorithmus 5;
 9. Berechnung und Mittelung f. alle Testwinkel $SLLA, SLLR \leftarrow$ Gleichung C.21;
-

Min-Kriterium als Mittelwertbildung aller standardisierten logarithmischen Winkelverluste aus Gleichung C.21. Es sind für jeden verfügbaren Testdatensatz und zugehörigen Simulationswinkel $X_* \mapsto \alpha_*$ die Verluste nach Gleichung C.22 auszurechnen und zu $MSLL$ zu mitteln [3]. Im Anschluss ist gebildeter Mittelwert einem Minimierungsverfahren zur Ermittlung des passenden Rauschniveaus σ_n^2 zuzuführen. Das Minimierungsproblem induziert dabei für jedes ausprobierte σ_n^2 ein Regressionsmodell. Die berechneten Modelle sind über ihre mittleren Verlust $MSLL$ miteinander zu vergleichen [3]. Das Modell für $\min MSLL$, besitzt die stärkste Generalisierung und somit das optimiert Rauschniveau σ_n^2 und sich nach Algorithmus 5 ergebenen optimierten Kernel-Parameter $\theta|\sigma_n^2$. Auch hier gilt wie für Gleichung C.21, dass sich eine gute Generalisierung für $MSLL < 0$ einstellt. Empirisch beobachtet Werte, liegen dabei im Intervall von $-2 < MSLL < -5$.

D Parametrierung der Erprobungs- und Optimierungsexperimente 0.0.1

26.04.2021

Parametergruppe	Parameter	Wert	Einheit	Kurzbeschreibung
SensorArrayOptions	geometry	'square'	-	Array-Geometrie-Indikator
	dimension	8	-	Sensor-Array-Pixel $N_{Pixel} \times N_{Pixel}$
	edge	2	mm	Sensor-Array-Kantenlänge
	V_{cc}	5	V	Sensor-Array-Betriebsspannung
	V_{off}	2,5	V	Sensor-Brücken-Offset-Spannung
DipoleOptions	V_{norm}	$1 \cdot 10^3$	mV	Kennfeldnormierung
	sphereRadius	2	mm	Kugelmagnetradius
	H_{0mag}	200	kAm $^{-1}$	Betragsfeldstärke Magnetfeldnormierung
	z_0	1	mm	Z-Abstand Magnetfeldnormierung
	m_{0mag}	$1 \cdot 10^6$	Am 2	Magnitude d. mag. Moments
Training-/ TestOptions	useCase	'Training' / 'Test'	'char'	Datensatzindikator f. Anwendungszweck
	xPos	[0,]	mm	Sensor-Array X-Positionsvektor
	yPos	[0,]	mm	Sensor-Array Y-Positionsvektor
	zPos	[7,]	mm	Sensor-Array Z-Positionsvektor
	tilt	0	°	Magnetverkipzung in Y-Achse
	angleRes	0,5	°	Winkelauflösung f. Magnetrotation
	phaseIndex	0	-	Phasenverschiebung-Index f. Startwinkel
	nAngles	20 / 720	-	Anzahl gleich verteilter Simulationswinkel
	BaseReference	'TDK'	char	Kennfelddatensatzindikator
	BridgeReference	'Rise'	char	Kennfeldindikator
GPROptions	kernel	'QFC'	char	Kernel-Funktion-Indikator (C.4), 'QFC' $\leftarrow d_F^2$
	θ	(1, 1)	-	Kernel-Parametervektor θ (C.5)
	σ_f^2 -Bounds	(0,1, 100)	-	Parameter-Bounds θ_1 f. Algorithmus 5
	σ_l -Bounds	(0,1, 100)	-	Parameter-Bounds θ_2 f. Algorithmus 5
	σ_n^2	10^{-6}	-	Rauschniveau, Rauschaufschaltung (C.6)
	σ_n^2 -Bounds	($10^{-8}, 10^{-4}$)	-	Parameter-Bounds σ_n^2 f. Algorithmus 7
	OptimRuns	30	-	Durchlaufanzahl f. Algorithmus 7
	SLL	'SLLA'	char	Verlust-Indikator f. Winkel (A) / R (Radius) Algorithmus 7
	mean	'zero'	char	Indikator Mittelwertpolynom Ein ('poly') / Aus ('zero')
	polyDegree	1	-	Grad des Mittelwertpolynoms wenn mean = 'poly'

Tabelle D.1: Simulationsparameter der Erprobungs- und Optimierungsexperimente. Von der Grundparametrierung abweichende Werte werden direkt in den Experimenten definiert und gesetzt. Parametergruppen entsprechend Unterabschnitt G.3.2

D.1 Vergleich der Kovarianzfunktionen

Abweichende Parameter:

- TrainingOptions: nAngles: 56
- GPROptions: kernel: variiert zwischen 'QFC' und 'QFCAPX'
- GPROptions: θ : zuerst $\theta_1 = 1 = \text{konst.}$, θ_2 variiert, danach vice versa

D.2 Anpassung der Referenzwinkelanzahl

Abweichende Parameter:

- TrainingsOptions: nAngles: variieren und werden schrittweise erhöht
- GRPOptions: kernel : 'QFCAPX'
- GPROptions: mean: variiert zwischen 'zero' und 'poly'
- GPROptions: $\sigma_n^2 = 10^{-6} = \text{konst.}$

D.3 Anpassung des Rauschniveaus

Abweichende Parameter:

- TrainingsOptions: nAngles: variieren und werden schrittweise erhöht
- GRPOptions: kernel : 'QFCAPX'
- GPROptions: mean: variiert zwischen 'zero' und 'poly'

D.4 Anpassung der Parametergrenzen

Abweichende Parameter:

- TrainingsOptions: nAngles: 17
- GRPOptions: kernel : 'QFCAPX'
- σ -Parameter-Bounds im Experiment angepasst
- OptimRuns verringert

E Ergebnisse der Erprobungs- und Optimierungsexperimente 0.0.1

26.04.2021

E.1 Vergleich der Kovarianzfunktionen

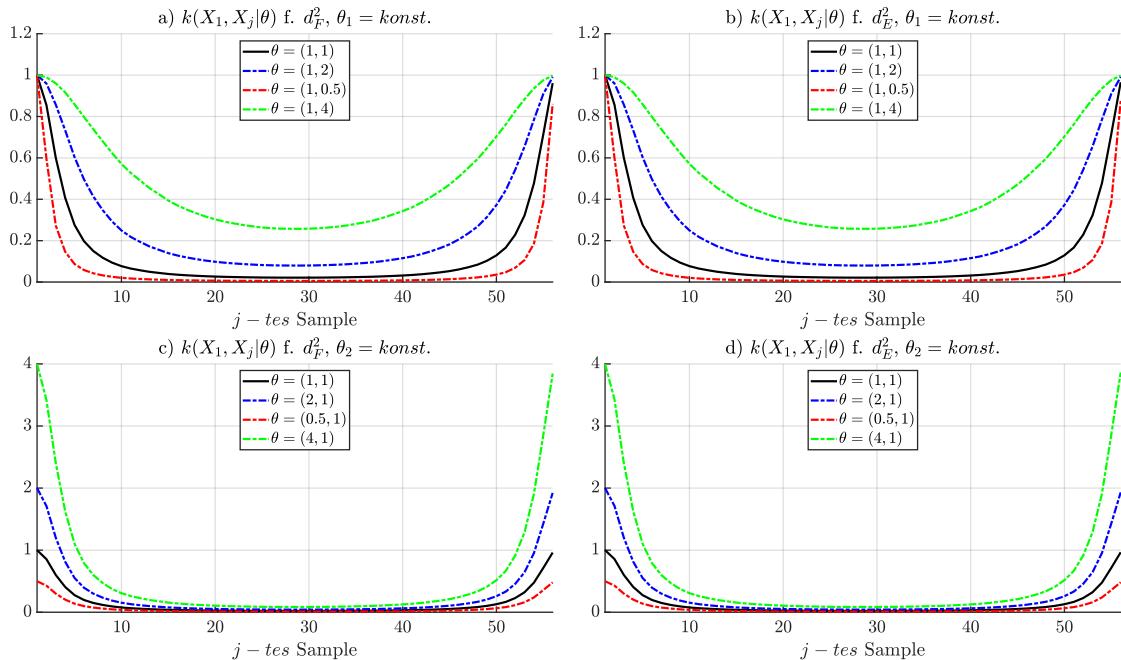


Abbildung E.1: Kovarianzfunktionen im Vergleich für variierende Kernel-Parameter $\theta = (\sigma_f^2, \sigma_l)$ und $N = 56$ Observerierungen.

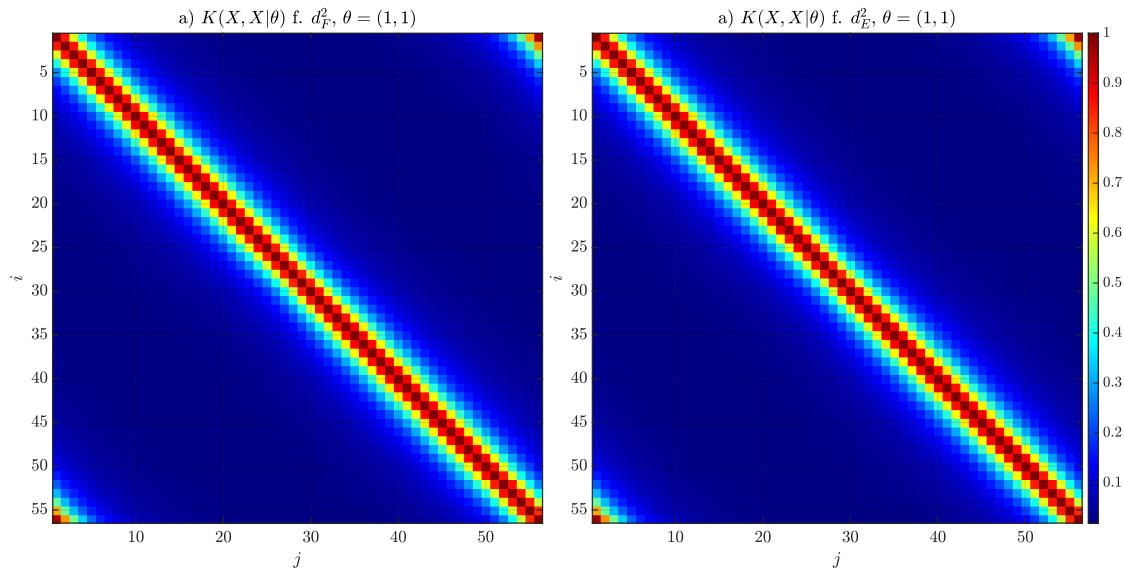


Abbildung E.2: Gegenüberstellung der Kovarianzmatrizen bei ausgeschalter Längen- und Breitenskalierung mit $\theta = (1, 1)$ und $N = 56$ Observerierungen.

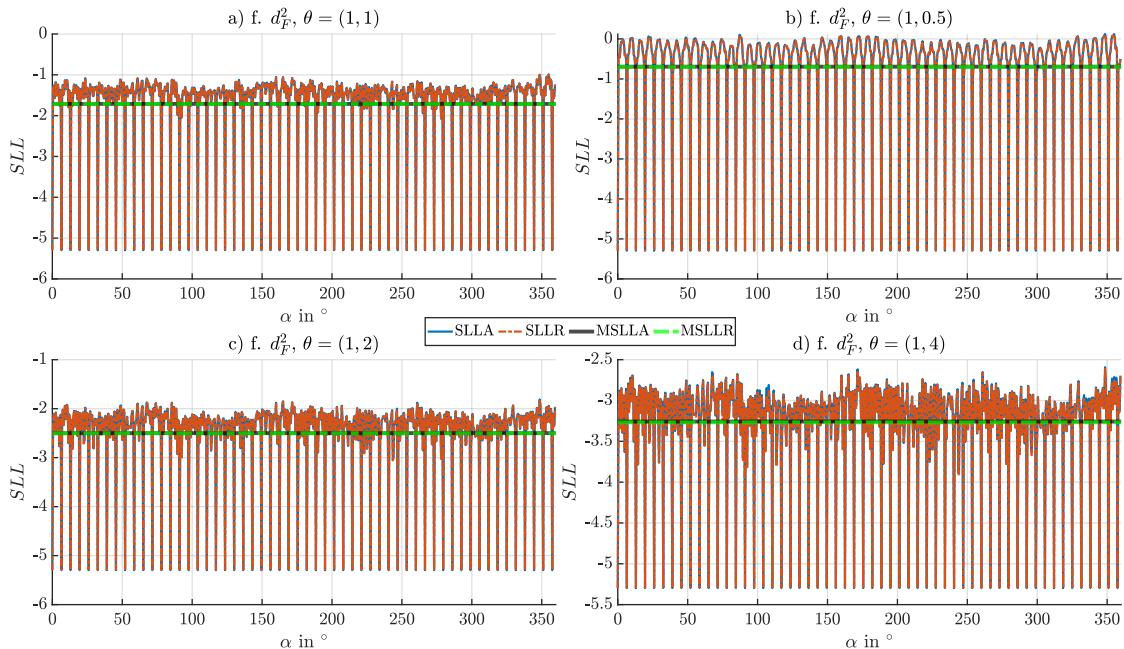


Abbildung E.3: Vergleich der Modellverluste nach Winkel und Radius für die erste Kovarianzfunktion für variierende Breitenskalierung.

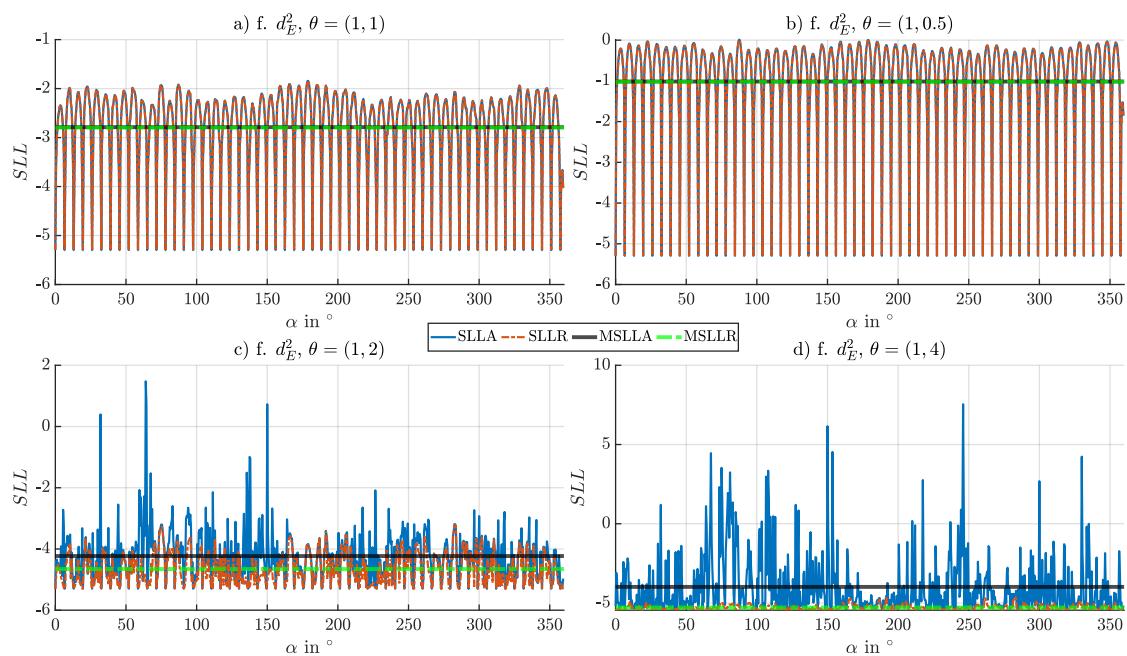


Abbildung E.4: Vergleich der Modellverluste nach Winkel und Radius für die zweite Kovarianzfunktion für variierende Breitenskalierung.

E.2 Anpassung der Referenzwinkelanzahl

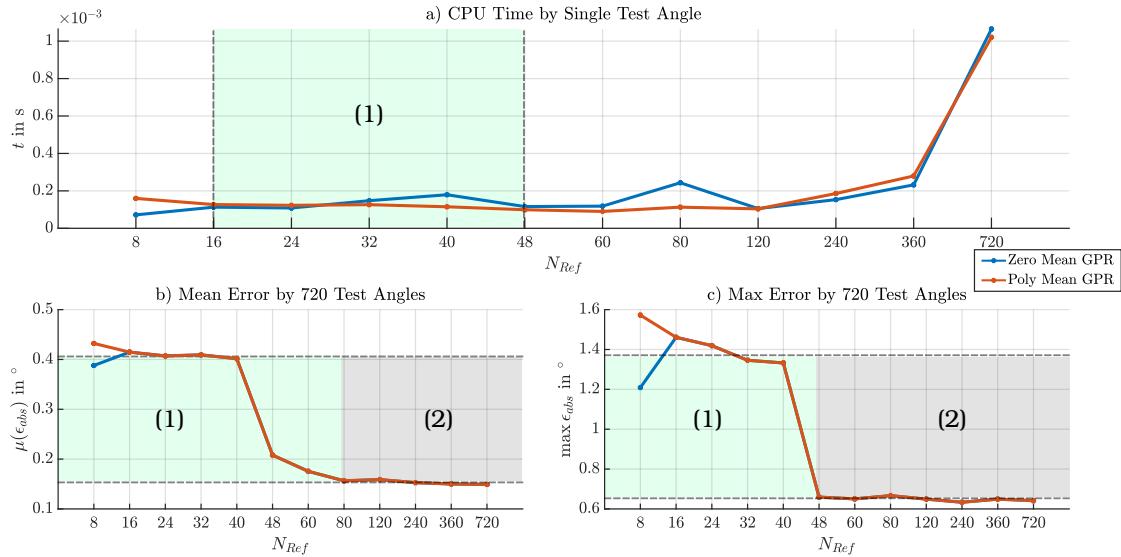


Abbildung E.5: QFCAPX Zero vs Poly 1 Nref Opt 32 (APX) = 8 (QFC) Timing vs Error, Noise Level Gap äußere Optimierung gleicht aus. (1) zu wählender Bereich mit 32 Samples, erfordert Rauschanpassung. (2) Rauschanpassung nicht zwingend notwendig, aber ressourcenintensiv.

E.3 Anpassung des Rauschniveaus

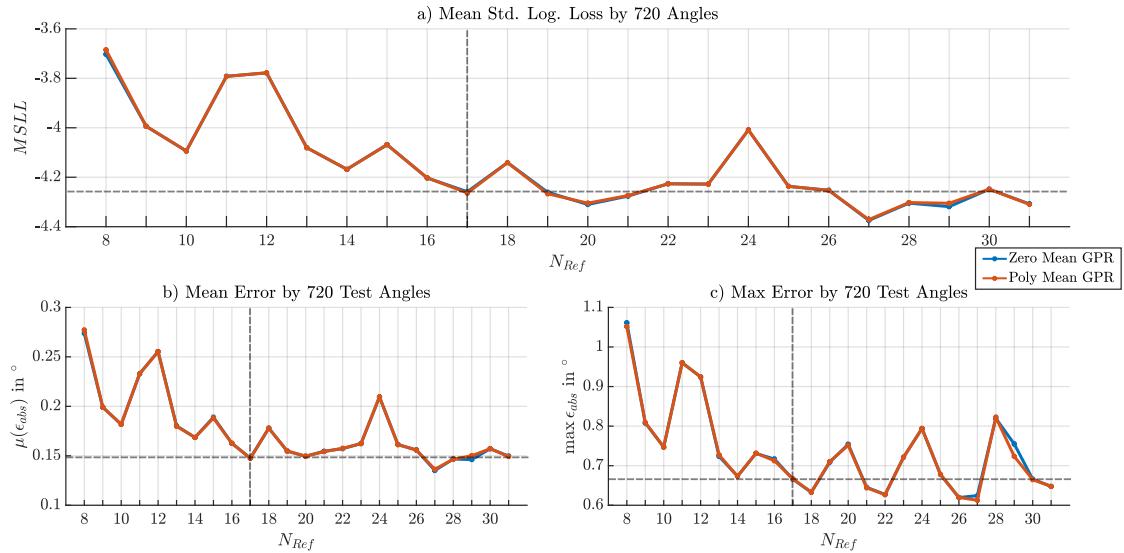


Abbildung E.6: MSLL vs Error für Nref 17 Kompromiss

E.4 Anpassung der Parametergrenzen

Kernel QFCAPX Ergebnis als Zero Mean Kernel für $N_{Ref} = 17$ mit $\sigma_f^2 = 4,83$, $\sigma_l = 23,48$, $\sigma_n^2 = 4,26 \cdot 10^{-1}$

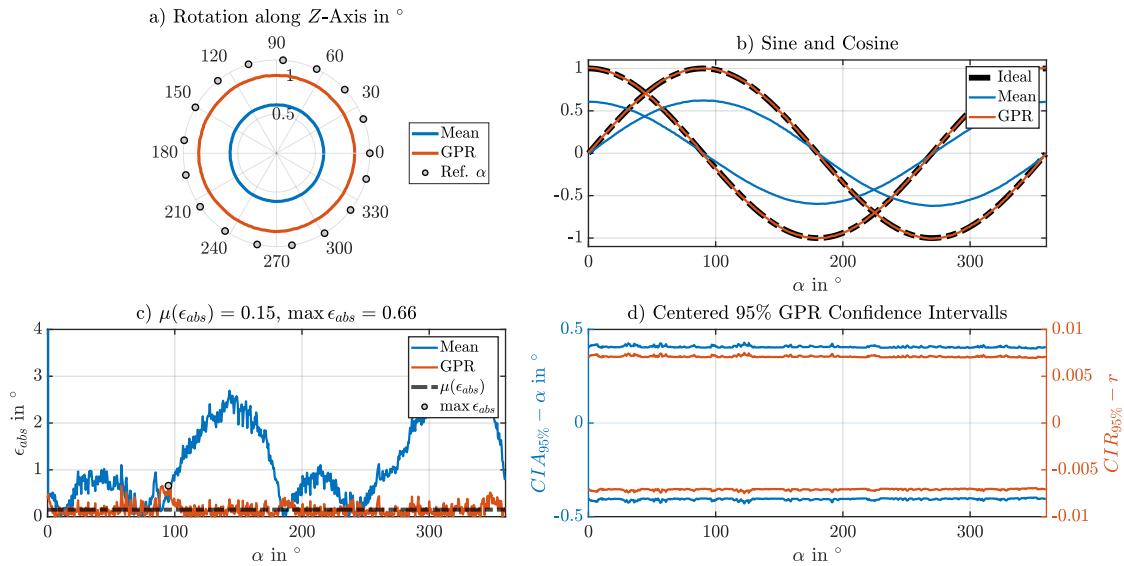


Abbildung E.7: QFCAPX Z N17 Rotation

Anpassung:

σ_f^2 -Bounds: (1, 10) / σ_l -Bounds: (10, 30) / σ_n^2 -Bounds: ($10^{-6}, 10^{-4}$)

Anpassung: Durchlaufzahl 10

Fertiges Modell

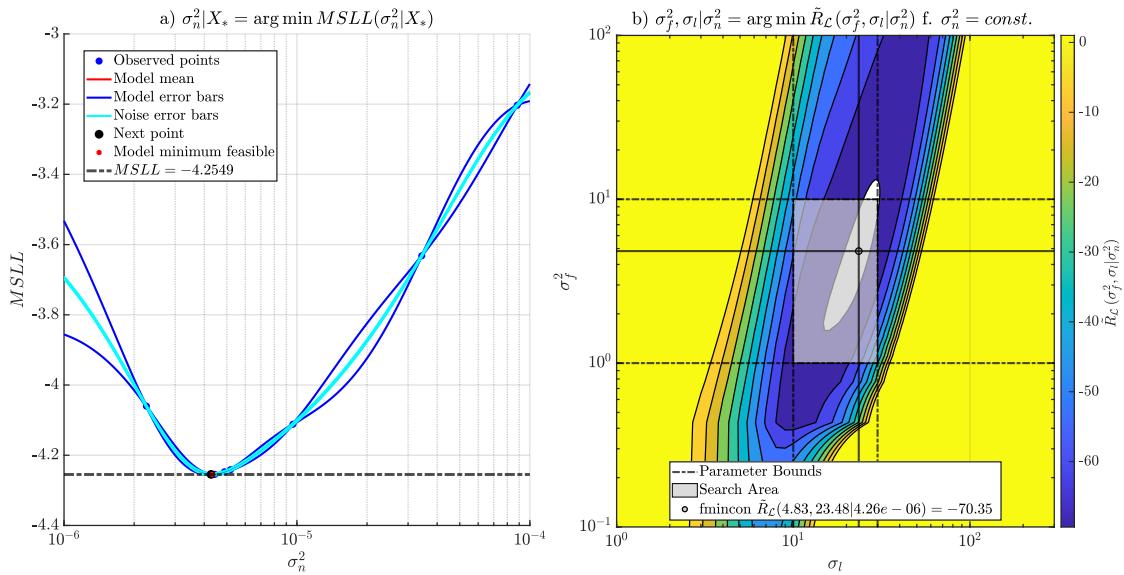


Abbildung E.8: Angepaster Parameter Bounds

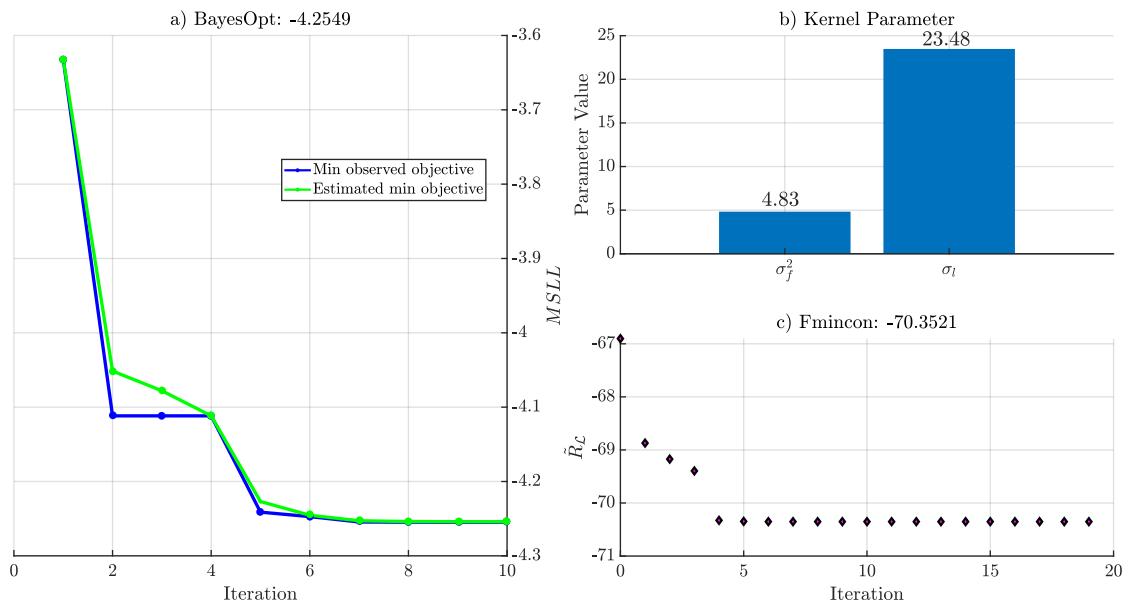


Abbildung E.9: QFCAPX Z N17 Optimierung Runs 10

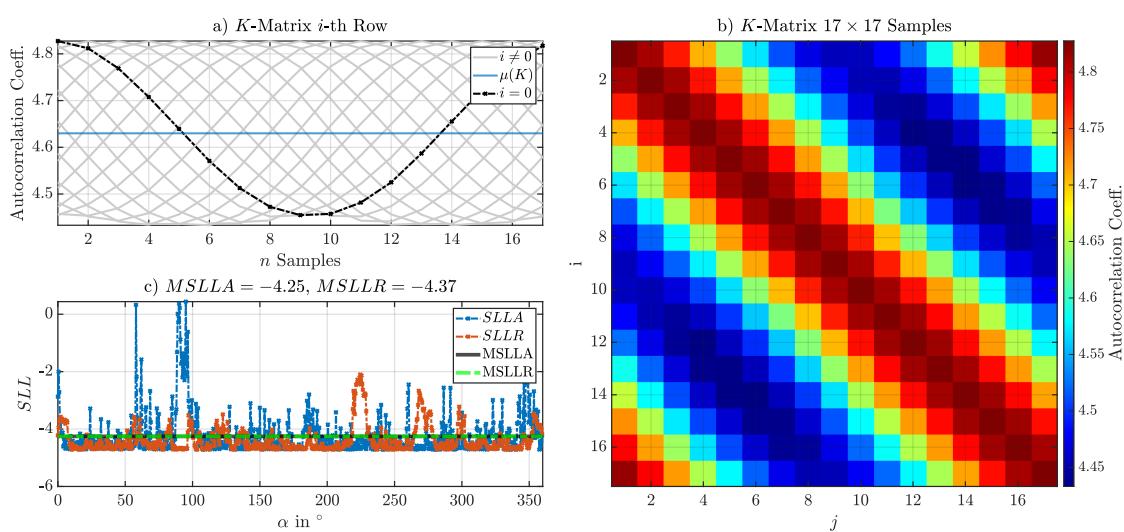


Abbildung E.10: QFCAPX Z N17 Modell

F Genutzte Software 0.0.3 08.01.2021

Für die Nachvollziehbarkeit der getätigten Entwicklungsarbeiten und die Erstellung der Bachelor-Thesis, ist das dafür jeweilige Betriebssystem (OS) und die verwendete Software (SW) tabellarisch aufgeführt. Es finden sich genutzte Versionen der SW und Angaben zur Minimalanforderung für deren Nutzung. Die Anforderungen sind für Prozessorkern (CPU), Arbeitsspeicher (RAM), Festplattenlaufwerk (HDD) näher aufgeschlüsselt. Die Programmierarbeiten mit MATLAB sind jeweils mit Windows und Linux geschrieben bzw. getestet worden.

Software	Verwendungszweck (Typ)	Min.-Anforderung	Version	Erscheinungstag
Ubunut Budgie	Linux-Betriebssystem (Laptop OS)	2 GHz Dual-Core-CPU 4 GB RAM 25 GB freier HDD-Speicher	18.04 LTS	26.04.2018
Windows 10 Enterprise	Windows-Betriebssystem (Laptop OS)	1 GHz Core-CPU 1 GB RAM 32 GB freier HDD-Speicher	1909	12.11.2020
MATLAB	Simulationssoftware (Multi-Paradigmen Programmiersprache, IDE)	Intel/ AMD x86-64 CPU 4 GB RAM 3.5 GB freier HDD-Speicher	2020b	17.09.2020
Git	Versionierung (Kommandozeilenprogramm)	- - -	2.29	29.10.2020
Inkscape	Vektorgrafikzeichenprogramm (Grafikaufbereitung)	1 GHz CPU 256 MB RAM 302 MB freier HDD-Speicher	0.92.3	11.03.2018
Texstudio	Textbearbeitung f. LaTeX Dokumente (Editor)	- - 24.7 MB freier HDD Speicher	2.12.6	25.07.2020
JabRef	Literaturverwaltungsprogramm f.BibLaTeX (Editor)	- - -	5.1	30.08.2020

Tabelle F.1: Genutzte Software zur Erstellung der Thesis und Dokumentation der Ergebnisse, Entwicklungsumgebung für die geschriebene Simulationssoftware zur Generierung und Auswertung der Sensor-Array-Simulation.

G Software-Dokumentation 0.0.7

18.04.2021

Die Software-Dokumentation ist automatisiert mit MATLAB-Skripten erstellt worden. Es ist dafür ein zweistufiger Prozess implementiert, der im ersten Schritt eine in MATLAB integrierte HTML-Dokumentation erstellt. Im Anschluss ist diese in Tex-Dateien. Als letzter Schritt sind diese zu einem LaTeX-Manualzusammengefasst im Anhang eingebunden. Mit diesem Verfahren ist es möglich, eine Dokumentation direkt aus geschriebenen M-Dateien zu generieren. Allerdings ist es dafür nötig, eine spezielle Formatierung und einen gewissen Programmierstil einzuhalten [6]. Die Dokumentation enthält neben dem erstellten Quellcode eine Reihe von Arbeitsanweisungen, wie mit der Software umzugehen ist. Zusätzlich sind Beschreibungen für die Erstellung und Pflege des Software-Projektes mit beigeftigt. Die geschriebene Software ist mithilfe des Software-Versionierungsprogramms Git erstellt worden, was eine genaue Nachvollziehbarkeit in Bezug auf die einzelnen Arbeitsschritte ermöglicht. Zur Versionierung ist der Git-Feature-Branch-Workflow [16] angewandt worden. Aus stilistischen Gründen ist die gesamte Software-Dokumentation in Englisch verfasst. Die Software-Dokumentation ist automatisiert durch eigens dafür geschriebene Skripte erstellt worden. Diese sind in der Dokumentation enthalten.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort

Datum

Unterschrift im Original