Fachhochschule Westküste
Hochschule für Wirtschaft & Technik

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Marco Gimm

Gegenüberstellung von zeit- und unterraumbasierten Algorithmen zur Echtzeit-Sprecherlokalisierung mit Mikrofonarrays

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Marco Gimm

Gegenüberstellung von zeit- und unterraumbasierten Algorithmen zur Echtzeit-Sprecherlokalisierung mit Mikrofonarrays

Masterarbeit eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Fachbereich Technik
der Fachhochschule Westküste
und
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Ulrich Sauvagerd
Zweitgutachter : Prof. Dr.-Ing. Hans-Dieter Schütte

Abgegeben am 28. August 2015

Marco Gimm

Thema der Masterarbeit

Gegenüberstellung von zeit- und unterraumbasierten Algorithmen zur Echtzeit-Sprecherlokalisierung mit Mikrofonarrays

Stichworte

Echtzeit-Sprecherlokalisierung, DOA, TDOA, MCCC, MUSIC, DSP, Mikrofonarray

Kurzzusammenfassung

In dieser Arbeit werden zeit- und unterraumbasierte Algorithmen zur Echtzeit-Sprecherlokalisierung miteinander verglichen. Dazu werden zunächst einige Vertreter beider Ansätze theoretisch betrachtet, bevor diese simuliert werden. Es folgt die Implementierung eines Algorithmus von jedem Ansatz auf einem Echtzeitsystem basierend auf einem TMS320C6657 DSP und einem linearen Mikrofonarray mit acht Mikrofonen. Hierauf werden die Algorithmen anhand verschiedener Gesichtspunkte untersucht. Zuletzt werden die Ergebnisse sowie weitere Erkenntnisse aus theoretischen Untersuchungen genutzt, um beide Verfahren gegenüberzustellen.

Marco Gimm

Title of the paper

Comparison of time- and subspace based algorithms for real time speaker localization with microphone arrays

Keywords

Real Time Speaker Localization, DOA, TDOA, MCCC, MUSIC, DSP, Microphone Array

Abstract

In this master thesis time- and subspace based algorithms for real time speaker localization are compared to each other. Therefore, the principles of some methods of both types are considered and simulated. After that one algorithm of each type is implemented on a realtimesystem based on a TMS320C6657 DSP and a linear microphone array with eight microphones. On this the algorithms are evaluated based on different aspects. At last the results and the results of the theoretically evaluation are used to compare both methods.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AKF	Autokorrelationsfunktion
AMDF	Average Magnitude Difference Function
BSLP	Backward Spatial Linear Prediction
bzw.	beziehungsweise
CIC	Chip Interrupt Controller
cm	Zentimeter
dB	Dezibel
DOA	Direction of Arrival
DSP	Digital Signal Processor
DTFT	discrete-time Fourier transform
etc.	et cetera
ESPRIT	Estimation of Signal Parameters via Rotational Invariance Techniques
evtl.	eventuell
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FSLP	Forward Spatial Linear Prediction
ggf.	gegebenenfalls
GPIO	General-purpose input/output
Hz	Hertz
IFFT	Inverse Fast Fourier Transform
ISR	Interrupt Service Routine
KKF	Kreuzkorrelationsfunktion
LDO	Low Drop Out Regulator
m	Meter
MCCC	Multichannel Cross-Correlation Coefficient
McBSP	Multichannel Buffered Serial Port
MUSIC	Multiple Signal Classification
o.g.	oben genannt
PCA	Principal Component Analysis
RAM	Random-Access Memory
RMSE	Root-Mean-Square Error
s	Sekunde
SDRAM	Synchronous Dynamic Random Access Memory
SLP	Spatial Linear Prediction

SNR	Signal-to-noise ratio
SRP	Steered Response Power
SVD	Singular Value Decomposition
TDOA	Time Difference of Arrival
ULA	Uniform Linear Array
VGA	Video Graphics Array
z.B.	zum Beispiel

Symbolverzeichnis

α ..... Frequenzauflösung
$\boldsymbol{A}$ ..... Ursprünglicher Datensatz
$\boldsymbol{A}_s(f)$ ..... Autokorrelationsmatrix der Quellsignale
$\boldsymbol{a}_{2:N}(p)$ ..... Vektor mit Koeffizienten für die Forward Linear Prediction
a_n ..... Dämpfungsfaktor an einem Mikrofon
$\boldsymbol{b}(k)$ ..... Eingangssignale
c ..... Schallgeschwindigkeit bei 20°C ; c=343,2$\frac{m}{s}$
$c_{con,x,y}$ ..... Faltungsfunktion
c_{xy} ..... Kovarianzfunktion
$\boldsymbol{D}$ ..... Verschiebungsmatrix
d ..... Abstand zwischen zwei Mikrofonen
$det(\cdot)$ ..... Determinante einer Matrix
$E[\cdot]$ ..... Erwartungswertoperator
η ..... Eigenwert mit Rauschanteil
$e_1(\cdot)$ ..... Prädiktionsfehler
$F(\omega)$ ..... Funktion im Frequenzbereich
$\boldsymbol{F}$ ..... Hilfsmatrix
$\mathcal{F}\{\dot{\}}$ ..... Fouriertransformierte
$\mathcal{F}^{-1}\{\dot{\}}$ ..... inverse Fouriertransformierte
$\mathcal{F}_n(\tau)$ ..... Funktion der Laufzeitunterschiede bezogen auf Referenzmikrofon
f ..... Frequenz
$f(t)$ ..... Funktion im Zeitbereich
$f_X(x)$ ..... Verteilungsdichte
f_a ..... Abtastfrequenz
$f_{XY}(x,y)$ .... Verbunds-Verteilungsdichte
$G_{xy}(f)$ ..... Kreuzleistungsspektrum
$\gamma_{xy}(f)$ ..... Kohärenz
$g(x)$ ..... Funktion
$g(x_i,y_i)$ ..... Funktion von zwei Variablen

$\boldsymbol{I}$ ..... Einheitsmatrix
$J_1(\cdot)$ ..... Funktion des mittleren quadratischen Prediktionsfehlers
K ..... Anzahl von Samples
κ ..... Ganzzahlige Verzögerung von Samples
L_s ..... Schalldruckpegel
$\boldsymbol{L}$ ..... Untere Dreiecksmatrix
$\boldsymbol{\Lambda}$ ..... Diagonalmatrix der Eigenwerte
λ ..... Eigenwert
M ..... Anzahl von Zufallsexperimenten
μ ..... Mittelwert
N ..... Anzahl von Mikrofonen im Array
$\boldsymbol{O}$ ..... Diagonalmatrix der Standardabweichungen
$P(\cdot)$ ..... Wahrscheinlichkeit
P_{ROOT} ..... Polynom mit konjungiert komplexen Nullstellen
$\boldsymbol{P}$ ..... Matrix (Produkt des Rauschunterraums mit adjungierten Rauschunterraum)
ϕ ..... Einfallswinkel einer Schallwelle
p ..... Variable für angenommene TDOA
$p_{a,y_n,y_m}(p)$ ... Normierte Korrelationsfunktion zwischen Signal n und Signal m
$\boldsymbol{Q}$ ..... Matrix aus Eigenvektoren
$\boldsymbol{q}$ ..... Eigenvektor
R_{cor} ..... Korrelationsfunktion im Frequenzbereich
$\boldsymbol{R}$ ..... Kovarianzmatrix
$\boldsymbol{R}_a(f,\phi)$ ..... Kovarianzmatrix im Frequenzbereich
$\boldsymbol{R}_a(p)$ ..... Parametrisierte Korrelationsmatrix
$\boldsymbol{R}_{vv}$ ..... Rauschkovarianz-Matrix
$\boldsymbol{R}_{a,s}(f,\phi)$ .... Kovarianzmatrix im Frequenzbereich ohne Rauschanteil
$\boldsymbol{r}_{a,2:N}(p)$ ..... Vektor mit Korrelationskoeffizienten des ersten Mikrofons in Abhängigkeit von p
$\widetilde{\boldsymbol{R}}_a(p)$ ..... Normierte parametrisierte Korrelationsmatrix
r ..... Radius
$r_{a,y_n,y_m}(p)$ ... Korrelationsfunktion zwischen Signal n und Signal m
r_{xx} ..... Autokorrelationsfunktion
$rank(\cdot)$ ..... Rang einer Matrix
$S(f)$ ..... Quellsignal im Frequenzbereich
$\boldsymbol{S}(f)$ ..... Quellsignale

$\boldsymbol{S}_{DSB}(p)$ ..... Funktion der Leistung des Delay and Sum Beamformers
$\boldsymbol{\Sigma}(\tau(\phi))$ ..... Matrix aus Steering-Vektoren
$\boldsymbol{\varsigma}(\tau(\phi))$ ..... Steering-Vektor
χ ..... Eigenwerte der Matrix $\boldsymbol{\Psi}$
σ_n ..... Standardabweichung des Signals n
σ_n^2 ..... Varianz des Signals n
$s(\cdot)$ ..... Signal einer Quelle im Zeitbereich
$span(\cdot)$ ..... Lineare Hülle
$\boldsymbol{T}$ ..... Hilfsmatrix
τ ..... Zeitverzögerung
$\boldsymbol{U}(f_i)$ ..... Transformationsmatrix
$\Upsilon_n(f)$ ..... Rauschen an einem Mikrofon im Frequenzbereich
$\boldsymbol{U}$ ..... Obere Dreiecksmatrix
$\upsilon_n(\cdot)$ ..... Rauschen an einem Mikrofon
$w(f_0, , \tau(\phi))$. Zur Transformationsmatrix korrespondierender Steering-Vektor
$\boldsymbol{V}$ ..... Matrix mit der Quadratwurzel der Rauschkovarianz
v ..... Geschwindigkeit
var_{x_n} ..... Varianz von x_n
$X_n(f)$ ..... Eingangssignal an einem Mikrofon im Frequenzbereich
x_i ..... Zufallsvariable
$x_n(\cdot)$ ..... Eingangssignal eines Mikrofons im Zeitbereich
$Y_n(f, \phi)$ ..... Ausgangssignal an einem Mikrofon im Frequenzbereich
$\boldsymbol{Y}$ ..... Rotierter Datensatz
$\boldsymbol{Y}(f, \phi)$ ..... Zusammengefasste Eingangssignale im Frequenzbereich
$\boldsymbol{\Psi}$ ..... Hilfsmatrix
$\boldsymbol{y}_a(k)$ ..... Zusammengefasste Eingangssignale im Zeitbereich
$y_n(\cdot)$ ..... Ausgangssignal eines Mikrofons im Zeitbereich
$z_\tau(k)$ ..... Ausganssignal des Delay and Sum Beamformers
z ..... Frequenzvariable in zeitdiskreten Systemen

1. Einleitung

In dieser Arbeit werden verschiedene Verfahren zur Echtzeit-Sprecherlokalisierung mit Mikrofonarrays gegenübergestellt.
Mikrofonarrays haben ein breites Anwendungsgebiet. So ist es neben der Lokalisierung und Verfolgung eines Sprechers möglich, Störgeräusche auszublenden bzw. mittels Beamforming eine räumliche Filterung vorzunehmen. Anwendungen hierfür sind Freisprecheinrichtungen in Fahrzeugen oder moderne Telefon- bzw. Videokonferenzeinrichtungen. Eine denkbare Anwendung für die Lokalisierung ist die Ortung von Störgeräuschen in Maschinen. Diese können mithilfe von Mikrofonarrays eingegrenzt werden, wodurch gerade bei der Entwicklung ein enormer Zeitgewinn entsteht. Für diese und andere Anwendungen wurden in den letzten Jahrzehnten diverse Algorithmen mit individuellen Vor- und Nachteilen entwickelt. Diese Algorithmen lassen sich dabei grob in zwei übergeordnete Methodiken unterteilen. Zum einem sind das zeitbasierte Algorithmen, die die Laufzeitunterschiede von Schallwellen an dem Mikrofonarray ausnutzen und zum anderen sind das unterraumbasierte Algorithmen, die eine Trennung von Signal- und Rauschunterraum vornehmen und aus diesen Richtungsinformationen gewinnen.

1.1. Motivation und Hintergrund

Die Idee für die Arbeit entstand aufgrund diverser vorangegangener Arbeiten an der Hochschule für angewandte Wissenschaften Hamburg. In diesen wurde zumeist ein konkretes Verfahren implementiert und untersucht, welches jeweils auf einem der beiden übergeordneten Ansätzen basiert. Es stellte sich die Frage, ob sich einer der Ansätze möglicherweise besser für die Echtzeit-Sprecherlokalisierung eignet und falls das so ist, aus welchen Gründen. Dabei sollen nicht einfach die theoretischen Fähigkeiten betrachtet werden, sondern

auch die tatsächliche Implementierung auf einem DSP-basierten Echtzeitsystem. Diese Systeme haben diverse Restriktionen bezüglich ihrer Rechenleistung und ihres Speicherplatzes, sodass diese Gesichtspunkte bei der tatsächlichen Anwendung der Algorithmen eine sehr bedeutsame Rolle spielen. Auch wenn ein Algorithmus in der Theorie sehr gut funktioniert, ist dieser nur anwendbar, wenn er auf der Zielplattform implementiert werden kann.
Auch die Genauigkeit der Ergebnisse ist ein wichtiger Gesichtspunkt. Auch hier kann es einen Unterschied zwischen Theorie und den tatsächlich erzielten Ergebnissen geben. Denn in einer tatsächlichen Anwendung kommen oft diverse Einflüsse zum Tragen, die in der Theorie nicht berücksichtigt wurden. Darunter fallen beispielsweise Einflüsse durch Rauschen, durch Quantisierung etc. Ziel der vorliegenden Arbeit ist es, beide Methoden hinsichtlich der genannten Gesichtspunkte zu untersuchen und einen Vergleich anzustellen.

1.2. Aufbau der Arbeit

Dieses Kapitel dient der Einleitung des Themas sowie Erläuterungen über Hintergründe und Motivation.
Das folgende Kapitel behandelt die theoretischen Grundlagen zu den behandelten Algorithmen sowie Grundsätzliches zur Mikrofonarray-Signalverarbeitung.
Anschließend geht es im dritten Kapitel um die Simulation der Algorithmen in MATLAB und die Beurteilung der Ergebnisse.
Daraufhin wird im vierten Kapitel auf die DSP-basierte Testumgebung mit den zugehörigen Komponenten eingegangen.
Im fünften Kapitel werden die Verfahren in der Programmiersprache C auf dem DSP implementiert.
Das sechste Kapitel behandelt das Profiling der Algorithmen bezüglich ihres Bedarfs an Rechenleistung und Speicherplatz.
In Kapitel sieben werden weitere Untersuchungen der Algorithmen durchgeführt. Unter anderen geht es um den Einfluss von Nachhall und um Messungen im Schallmessraum.
Nachdem die Verfahren theoretisch beschrieben, simuliert, implementiert und getestet wurden, soll in Kapitel acht eine Zusammenfassung der Ergebnisse und eine Gegenüberstellung stattfinden.
Kapitel neun beinhaltet eine Zusammenfassung und einen Ausblick auf mögliche weitere Arbeiten.

2. Theoretische Grundlagen

2.1. Signalverarbeitung mit Mikrofonarrays

Mikrofonarrays sind Anordnungen von mehreren Mikrofonen, welche verschiedenartig aufgebaut sein können. Eine Möglichkeit ist zum Beispiel die lineare Anordnung in einem sogenannten Uniform Linear Array, in dem mehrere Mikrofone im gleichen Abstand in einer Reihe angeordnet sind. Eine weitere Möglichkeit ist das sogenannte Uniform Circular Array, in welchem die Mikrofone gleichmäßig auf einem Kreisumfang angeordnet sind. Es gibt diverse weitere Varianten, z.B. kugelförmige und viele mehr, welche jeweils verschiedene Eigenschaften aufweisen und verschiedene Verfahren zulassen.
Der Grundgedanke ist dabei immer der gleiche. Durch den räumlichen Abstand der Mikrofone entstehen Laufzeitunterschiede von Schallwellen, die verschieden lange Wege zu den einzelnen Mikrofonen zurücklegen. In diesen Laufzeitunterschieden stecken je nach Anordnung der Mikrofone verschiedene Informationen über die Position der Schallquelle. Wichtig ist dabei die Unterscheidung zwischen Nah- und Fernfeld. Direkt vor der Schallquelle befindet sich das Nahfeld, welches nicht gleichmäßig verteilte Interferenz-Orte aufweist. In einiger Entfernung von der Quelle befindet sich das Fernfeld. Dieses ist durch planare Wellenfronten gekennzeichnet, die durch Kombination von komplexen Exponentialfunktionen beschrieben werden können.[3]
Das Fernfeld-Modell soll in der gesamten Arbeit Anwendung finden. Die Abbildung 2.1 zeigt eine lineare äquidistante Anordnung von Mikrofonen, in welcher sich der Laufzeitunterschied leicht aus den Winkelfunktionen und der Schallgeschwindigkeit c wie folgt berechnen lässt.

$$\tau = \frac{d}{c} \cdot sin(\phi) \tag{2.1}$$

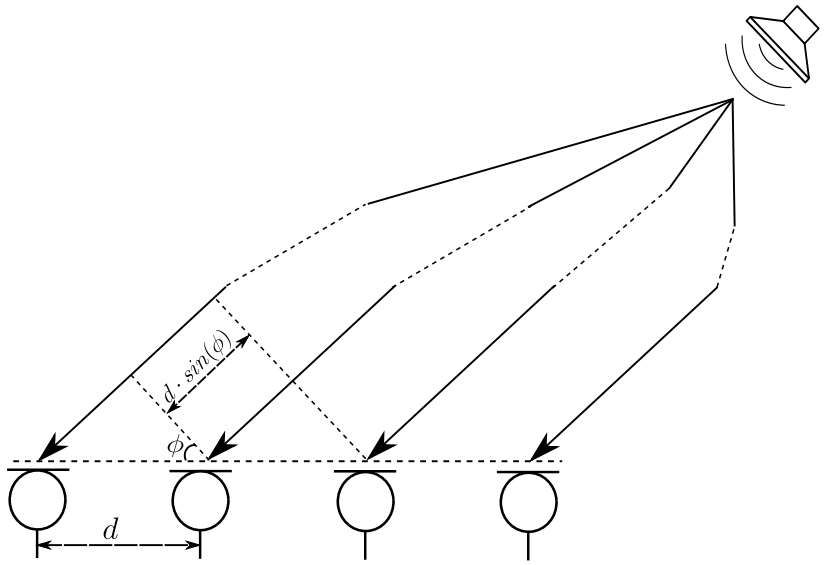


Abbildung 2.1.: Äquidistante lineare Anordnung von Mikrofonen

2.2. Räumliches Aliasing

Wenn es um die Dimensionierung des Mikrofonarrays geht, ist einer der bedeutsamsten Parameter der Abstand zwischen den einzelnen Mikrofonen. Je größer dieser Abstand ist, desto größer ist auch die zeitliche Differenz zwischen den Eingangssignalen, sofern die Quelle nicht direkt vor dem Array platziert ist. Der naheliegendste Gedanke ist, dass dies positiv zu bewerten ist und somit eine bessere Auflösung möglich wäre. Dieser Gedanke ist zunächst nicht einmal falsch, allerdings lässt sich der Abstand nicht beliebig groß wählen.
Der Grund dafür kann recht gut bildlich erklärt werden. Angenommen das Quellsignal besteht nur aus einem Sinus mit einer festen Frequenz. Aufgrund der Periodizität des Sinus entsteht bei einer Verzögerung von einer oder mehr als einer Wellenlänge eine Mehrdeutigkeit. Da diese Verzögerung aus Sicht des Referenzmikrofons auch negativ sein kann, darf die Phasenverschiebung lediglich $\phi < \pi$ betragen, da $2\pi - \phi$ und umgekehrt sonst nicht zu unterscheiden wären. Diese Mehrdeutigkeit gilt es für eine ordnungsgemäße Funktion der Algorithmen zur Bestimmung der so genannten Time Difference of Arrival (TDOA) bzw. Direction of Arrival (DOA) zu vermeiden. Dies geschieht zum einen durch eine sinnvolle Festlegung des Abstandes zwischen zwei Mikrofonen und zum anderen durch die Bandbegrenzung der Eingangssignale.

Diese beiden Parameter sollen im Folgenden diskutiert werden.
Zunächst wird dafür die höchste im Signal vorkommende Frequenz auf 3400 Hz festgelegt. Bis zu dieser Frequenz sind die signifikanten Anteile der menschlichen Sprache enthalten [11, S. 4]. Nun ist der zuvor erläuterte Ansatz zu berücksichtigen:

$$2\pi f\tau < \pi \tag{2.2}$$

Die Zeitdifferenz τ hängt von verschiedenen Faktoren ab und lässt sich wie folgt berechnen:

$$\tau = \frac{d}{v} \cdot sin(\phi) \tag{2.3}$$

Dabei ist:

- d: Abstand zwischen zwei Mikrofonen
- v: Geschwindigkeit des Signals
- ϕ : Einfallswinkel der Quelle

Die größte Phasenverschiebung entsteht, wenn die Quelle bei $\pm$ 90° lokalisiert ist. An diesen Stellen hat der Sinus einen Funktionswert von $\pm$1, sodass sich die Gleichung 2.3 unter dieser Worst-Case-Annahme zu dem folgenden Ausdruck vereinfacht:

$$\tau = \pm\frac{d}{v} \tag{2.4}$$

Eingesetzt in Gleichung 2.2 ergibt sich:

$$\pm 2\pi f\frac{d}{v} < \pi \tag{2.5}$$

Mit der Formel zur Berechnung der Wellenlänge λ:

$$\lambda = \frac{v}{f} \tag{2.6}$$

folgt:

$$d < \pm\frac{\lambda}{2} \tag{2.7}$$

$$d < \frac{\lambda}{2} \tag{2.8}$$

Das Vorzeichen kann in diesem Fall ignoriert werden, da es lediglich einen Bezug auf die Lage des Referenzmikrofons herstellt. Entscheidend ist, dass der Abstand zwischen zwei Mikrofonen geringer gewählt wird als die halbe Wellenlänge der höchsten im Signal vorkommenden Frequenz.
Die folgende Abbildung verdeutlicht den Zusammenhang zwischen der Korrelationsfunktion zweier Sinusfunktionen, der Signalfrequenz und der Zeitverschiebung bei einem Signaleinfallswinkel von 0°.

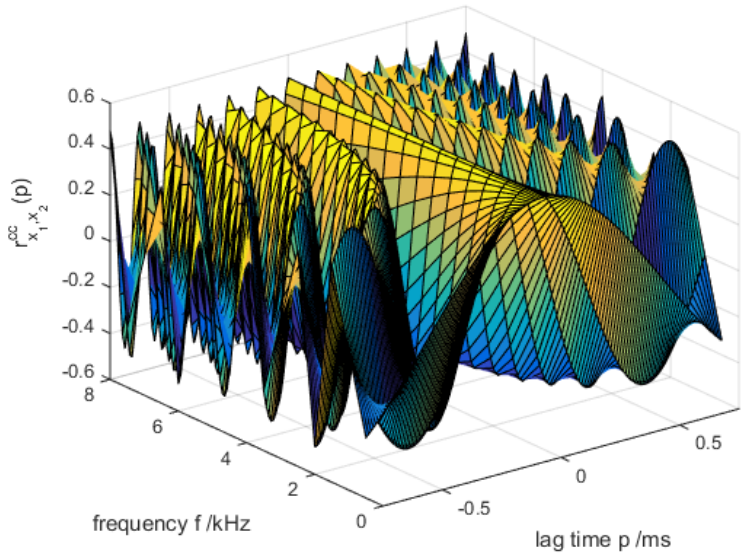


Abbildung 2.2.: Räumliches Alias in Abhängigkeit der Signalfrequenz (Eigene Darstellung in Anlehnung an [5, S. 189])

Es ist deutlich zu erkennen, dass mit steigender Frequenz schon bei geringeren Zeitverschiebungen sich wiederholende Maxima der Korrelationsfunktion ausbilden. Dies ist auf die zuvor beschriebene Mehrdeutigkeit zurückzuführen. Ebenso ist zu erkennen, dass die Frequenz nichtlinear in die Funktion eingeht.

2.3. Zeitbasierte Verfahren

In diesem Teil der Arbeit soll auf verschiedene Verfahren eingegangen werden, die auf der Korrelation zwischen den Mikrofonsignalen bzw. auf deren Laufzeitunterschied beruhen.

2.3.1. Signalmodell

Um im weiteren Verlauf näher auf die verschiedenen Verfahren einzugehen, ist es sinnvoll, zunächst ein geeignetes Signalmodell zu definieren. Dafür soll hier von lediglich einer Nutzsignalquelle ausgegangen werden. Weiterhin wird festgelegt, dass das betrachtete Mikrofonarray aus N Mikrofonen besteht. Die Umgebung wird als reflexionslos betrachtet und das Mikrofonarray befindet sich im Fernfeld.
Ein Signal, welches von einem Mikrofon erfasst wird, lässt sich beschreiben als [5, S. 185]:

$$\begin{aligned} y_n(k) &= a_n \cdot s(k - t - \tau_{n1}) + \upsilon_n(k) \\ &= a_n \cdot s(k - t - \mathcal{F}_n(\tau)) + \upsilon_n(k) \\ &= x_n(k) + \upsilon_n(k), \quad (n = 1, 2, \ldots, N) \end{aligned} \tag{2.9}$$

In Matrixschreibweise lässt sich das Signalmodell für alle Mikrofone wie folgt beschreiben [13]:

$$\begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ \vdots \\ y_N(k) \end{pmatrix} = \begin{pmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ 0 & 0 & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & a_N \end{pmatrix} \begin{pmatrix} s(k-t) \\ s(k - t - \mathcal{F}_2(\tau)) \\ s(k - t - \mathcal{F}_3(\tau)) \\ \vdots \\ s(k - t - \mathcal{F}_N(\tau)) \end{pmatrix} + \begin{pmatrix} \upsilon_1(k) \\ \upsilon_2(k) \\ \upsilon_3(k) \\ \vdots \\ \upsilon_N(k) \end{pmatrix} \tag{2.10}$$

Das Signal am Mikrofon $n = 1, 2, \ldots, N$ setzt sich somit aus einem Dämpfungsfaktor a_n, welcher in einem Bereich zwischen null und eins liegt, einem individuellen Rauschanteil $\upsilon_n(k)$ und dem zeitverzögerten Ursprungssignal $s(k - t - \mathcal{F}_n(\tau))$ zusammen. Die Zeit t ist die Zeit, welche die Schallwellen zwischen der Quelle und dem Referenzmikrofon benötigt.

τ ist die TDOA zwischen zwei Mikrofonen und $\tau_{n1} = \mathcal{F}_n(\tau)$ ist die TDOA zwischen dem ersten und dem Mikrofon an der Stelle n. Diese Funktion hängt von der Arraygeometrie ab. Für das hier betrachtete ULA lautet die Funktion:

$$\mathcal{F}_n(\tau) = (n-1) \cdot \tau, \quad n = 2, 3, \ldots, N \tag{2.11}$$

An dieser Stelle soll angemerkt werden, dass das hier vorgestellte Signalmodell nicht vollständig den tatsächlichen Gegebenheiten in realen Umgebungen entspricht. Dort gibt es neben den bisher beschriebenen Signalanteilen weitere durch Reflexionen entstehende Signalanteile. Diese stammen von derselben Quelle, gelangen jedoch nicht auf dem direkten Weg zu den Mikrofonen. Die folgende Abbildung illustriert die Entstehung dieser Signalanteile.

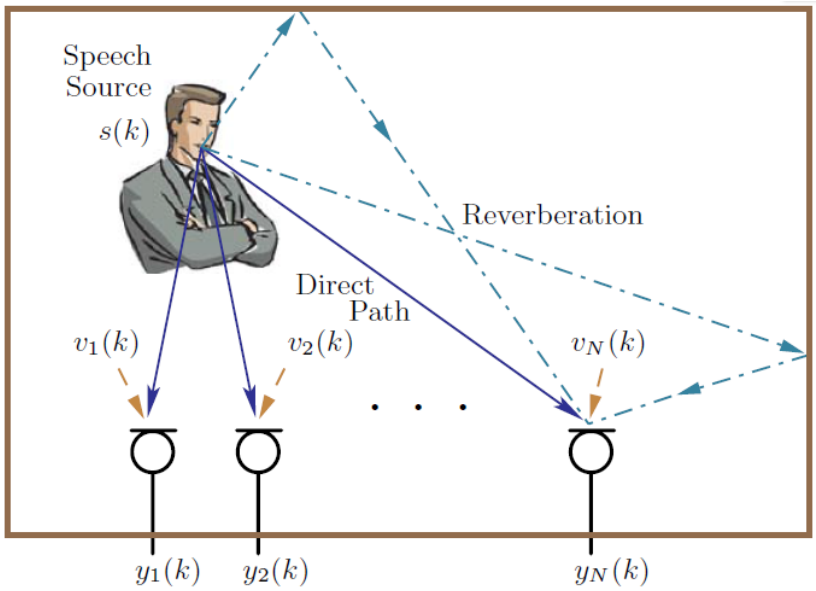


Abbildung 2.3.: Signalanteile durch Nachhall [5, S. 186]

Diese Anteile sind unerwünscht und haben in den vorgestellten Anwendungen keinen Nutzen. Daher genügt zur Beschreibung der Funktionsweise der Verfahren zunächst das in Gleichung 2.10 vorgestellte Signalmodell. Allerdings können diese zusätzlichen Signalanteile durchaus Auswirkungen auf die im Folgenden beschriebenen Algorithmen haben, sodass diese noch zu diskutieren sind.

2.3.2. Spatial Linear Prediction

Das Verfahren der Spatial Linear Prediction (SLP) beruht auf der parametrisierten Korrelationsmatrix und eignet sich für N > 2 Mikrofone. Beschrieben wird es z. B. in [5], woran die folgenden Ausführungen angelehnt sind. Durch die größere Anzahl von Mikrofonen steigt die Redundanz des Systems an. Das Ergebnis wird genauer. Dieser Zusammenhang ist dadurch gegeben, dass bei bekannter Arraygeometrie auch die Laufzeitdifferenzen eines Quellsignals bei gegebenen Winkel bekannt sind. Diese betragen zwischen dem ersten und dem zweiten Mikrofon τ_{12}, zwischen dem ersten und dem dritten Mikrofon $\tau_{13} = \tau_{12} + \tau_{23}$ usw. Auch hier soll wieder vom Fernfeld-Modell ausgegangen werden. Jedoch wird zunächst anders als in Gleichung 2.9 ein rauschfreies Signal modelliert:

$$y_n(k + \mathcal{F}_n(\tau)) = a_n \cdot s(k - t) \tag{2.12}$$

Das Referenzsignal, welches am ersten Mikrofon gemessen wurde, ist in Phase mit dem Signal, welches am Mikrofon an der Stelle n gemessen wurde, sofern der korrekte Wert für τ gefunden wurde. Dafür ist es sinnvoll, eine Variable p einzuführen, die sämtliche Werte für τ zwischen $[\tau_{min} \dots \tau_{max}]$ annehmen kann, und die Definition eines Fehlersignals erlaubt.

$$e_1(k, p) = y_1(k) - \boldsymbol{y}^T_{a,2:N}(k, p)\boldsymbol{a}_{2:N}(p) \tag{2.13}$$

Darin ist $y_1(k)$ das Signal am Referenzmikrofon und $\boldsymbol{y}^T_{a,2:N}(k, p)$ ein Vektor aus den Signalen der anderen Mikrofone, welche jeweils um $\mathcal{F}_n(p)$ verschoben sind.

$$\boldsymbol{y}_{a,2:N}(k, p) = [y_2(k + \mathcal{F}_2(p)) \ \cdots \ y_N(k + \mathcal{F}_N(p))]^T \tag{2.14}$$

Der Vektor

$$\boldsymbol{a}_{2:N}(p) = [a_2(p) \ a_3(p) \ \dots \ a_n(p)]^T \tag{2.15}$$

beschreibt die so genannten "Forward Linear Prediction Coefficients", mit welchen die verschobenen Signale gewichtet werden. Gleichung 2.13 ist so gestaltet, dass der Fehler für $p = \tau$ zu null wird. Dies funktioniert in der Praxis jedoch nicht, da immer ein Rauschen vorliegt und durch die räumliche Auflösung die diskrete Variable p sehr wahrscheinlich niemals den genauen Wert τ annimmt. Daher wird der Wert für p gesucht, bei dem der Fehler

e_1 minimal ist. Dazu wird aus Gleichung 2.13 der mittlere quadratische Prädiktionsfehler gebildet,

$$J_1(p) = E[e_1^2(k,p)] \tag{2.16}$$

dessen Minimierung zu dem linearen System

$$\boldsymbol{R}_{a,2:N}(p)\boldsymbol{a}_{2:N}(p) = \boldsymbol{r}_{a,2:N}(p) \tag{2.17}$$

führt. Darin ist $\boldsymbol{R}_{a,2:N}(p)$ die räumlich parametrierbare Korrelationsmatrix, die wie folgt gebildet wird:

$$\begin{aligned}\boldsymbol{R}_{a,2:N}(p) &= E[\boldsymbol{y}_{a,2:N}(k,p)\ \boldsymbol{y}_{a,2:N}^T(k,p)] \\ &= \begin{pmatrix} \sigma_{y_2}^2 & r_{a,y_2,y_3}(p) & \cdots & r_{a,y_2,y_N}(p) \\ r_{a,y_3,y_2}(p) & \sigma_{y_3}^2 & \cdots & r_{a,y_3,y_N}(p) \\ \vdots & \vdots & \ddots & \vdots \\ r_{a,y_N,y_2}(p) & r_{a,y_N,y_3}(p) & \cdots & \sigma_{y_N}^2 \end{pmatrix}\end{aligned} \tag{2.18}$$

Diese Matrix beinhaltet auf der Diagonalen die Varianzen der einzelnen Signale ab dem zweiten Mikrofon.

$$\sigma_n^2 = E[y_n^2(k)] \tag{2.19}$$

Die anderen Elemente sind die Kovarianzen zwischen den Signalen der verschiedenen Mikrofone in Abhängigkeit von p.

$$r_{a,y_n,y_m}(p) = E[y_n(k+\mathcal{F}_n(p))y_m(k+\mathcal{F}_m(p))] \tag{2.20}$$

Der Vektor $\boldsymbol{r}_{a,2:N}(p)$ beinhaltet alle Korrelationskoeffizienten zwischen dem Signal am Referenzmikrofon und den anderen Mikrofonen in Abhängigkeit der angenommenen TDOA p.

$$\boldsymbol{r}_{a,2:N}(p) = [r_{a,y_1,y_2}(p)\ r_{a,y_1,y_3}(p) \ldots\ r_{a,y_1,y_N}(p)] \tag{2.21}$$

Durch Umstellen von Gleichung 2.17 kann nach $\boldsymbol{a}_{2:N}(p)$ aufgelöst werden.

$$\boldsymbol{a}_{2:N}(p) = \boldsymbol{R}^{-1}_{a,2:N}(p)\boldsymbol{r}_{a,2:N}(p) \tag{2.22}$$

Dadurch ergibt sich für das Fehlersignal in Gleichung 2.13:

$$e_1(k,p) = y_1(k) - \boldsymbol{y}^T_{a,2:N}(k,p)\boldsymbol{R}^{-1}_{a,2:N}(p)\boldsymbol{r}_{a,2:N}(p) \tag{2.23}$$

Und aus dem minimalen quadratischen Fehler wird:

$$J_{1,min}(p) = E[e_1^2(k,p)] = \sigma^2_{y_1} - \boldsymbol{r}^T_{a,2:N}(p)\boldsymbol{R}^{-1}_{a,2:N}(p)\boldsymbol{r}_{a,2:N}(p) \tag{2.24}$$

Wird nun der Fehler über alle möglichen p berechnet, so ist die gesuchte TDOA an der Stelle, an der $J(p)$ minimal ist:

$$\hat{\tau}^{FSLP} = \underset{p}{\operatorname{argmin}}\ J_{1,min}(p) \tag{2.25}$$

FSLP steht für Foward Spatial Linear Prediction und bedeutet, dass das erste im Array befindliche Mikrofon als Referenz benutzt wird. Daneben existiert noch die Backward Spatial Linear Prediction (BSLP), bei der das letzte im Array befindliche Mikrofon als Referenz herangezogen wird. [4]
Bei Betrachtung von Gleichung 2.23 fällt auf, dass die hier gemachten Annahmen, was das Rauschen betrifft, sogar eine Notwendigkeit für diese Methode darstellen. Denn in einer absolut rauschfreien Umgebung würde für $p = \tau$ die Matrix $\boldsymbol{R}_{a,2:N}(p)$ den Rang 1 annehmen, da bei vollständiger Korrelation alle Vektoren in der Matrix linear abhängig wären. Die Matrix wäre nicht mehr invertierbar.

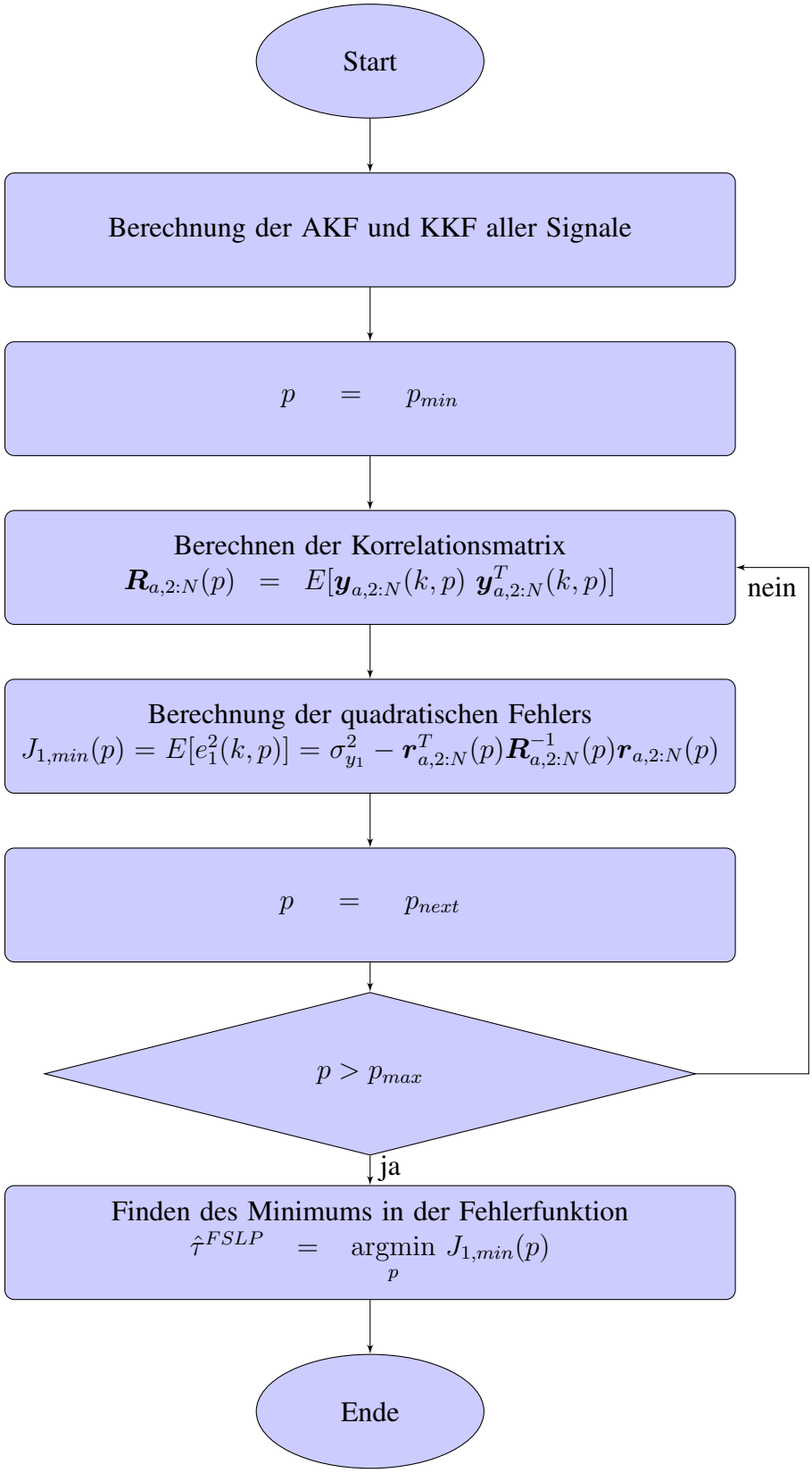


Abbildung 2.4.: Ablauf der SLP

2.3.3. Multichannel Cross-Correlation Coefficient

Die Multichannel Cross-Correlation Coefficient-Methode ist eine weitere Möglichkeit, um die TDOA aus der parametrierbaren Korrelationsmatrix zu bestimmen. Sie wird unter anderen in [5] beschrieben, woran die folgenden Ausführungen angelehnt sind. Auch hierbei geht es um die Quellsuche mittels parametrierbarer Korrelationsmatrix und dem Parameter p, welcher jeweils eine mögliche TDOA darstellt. Der Unterschied ist allerdings, dass nicht ein Referenzmikrofon herangezogen wird, mit dem alle anderen verglichen werden, sondern ein Vergleich aller Signale untereinander stattfindet. Dazu wird der Signalvektor wie folgt definiert:

$$\boldsymbol{y}_a(k,p) = [y_1(k,p)\ y_2(k+\mathcal{F}_2(p))\ \cdots\ y_N(k+\mathcal{F}_N(p))]^T \tag{2.26}$$

Die Korrelationsmatrix lässt sich schreiben als:

$$\boldsymbol{R}_a(p) = E[\boldsymbol{y}_a(k,p)\ \boldsymbol{y}_a^T(k,p)] \tag{2.27}$$

$$= \begin{pmatrix} \sigma_{y_1}^2 & r_{a,y_1,y_2}(p) & \cdots & r_{a,y_1,y_N}(p) \\ r_{a,y_2,y_1}(p) & \sigma_{y_2}^2 & \cdots & r_{a,y_2,y_N}(p) \\ \vdots & \vdots & \ddots & \vdots \\ r_{a,y_N,y_1}(p) & r_{a,y_N,y_2}(p) & \cdots & \sigma_{y_N}^2 \end{pmatrix} \tag{2.28}$$

Diese beinhaltet im Fall des MCCC-Algorithmus auch alle Kovarianzen des Signals am ersten Mikrofon. Die Korrelationsmatrix lässt sich wie folgt faktorisieren:

$$R_a(p) = \boldsymbol{O}\widetilde{\boldsymbol{R}}_a(p)\boldsymbol{O} \tag{2.29}$$

Darin ist $\boldsymbol{O}$ die Diagonalmatrix der Standardabweichungen der einzelnen Signale.

$$\boldsymbol{O} = \begin{pmatrix} \sigma_{y_1} & 0 & \dots & 0 \\ 0 & \sigma_{y_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{y_N} \end{pmatrix} \tag{2.30}$$

Somit wurde die räumliche Korrelationsmatrix normalisiert und in die folgende Form gebracht:

$$\widetilde{\boldsymbol{R}}_a(p) = \begin{pmatrix} 1 & p_{a,y_1,y_2}(p) & \cdots & p_{a,y_1,y_N}(p) \\ p_{a,y_2,y_1}(p) & 1 & \cdots & p_{a,y_2,y_N}(p) \\ \vdots & \vdots & \ddots & \vdots \\ p_{a,y_N,y_1}(p) & p_{a,y_N,y_2}(p) & \cdots & 1 \end{pmatrix} \tag{2.31}$$

Diese Matrix ist symmetrisch und positiv semi-definit. Die Elemente

$$p_{a,y_n,y_m}(p) = \frac{r_{a,y_n,y_m}(p)}{\sigma^2_{y_m}\sigma^2_{y_m}} \tag{2.32}$$

sind die parameterabhängigen Korrelationskoeffizienten zwischen zwei Mikrofonsignalen. Durch die Eigenschaften der Matrix $\widetilde{R}_a(p)$ lässt sich zeigen, dass gilt:

$$0 \leq det\left[\widetilde{\boldsymbol{R}}_a(p)\right] \leq 1 \tag{2.33}$$

Für den Fall, dass das betrachtete Array nur zwei Mikrofone besitzt, ergibt sich für die Determinante:

$$det\left[\widetilde{\boldsymbol{R}}_a(p)\right] = 1 - p^2_{a,y_1,y_2}(p) \tag{2.34}$$

Durch Umstellen der Formel lässt sich anhand des Beispiels für zwei Mikrofone zeigen, wie der Zusammenhang der Korrelationskoeffizienten und der normalisierten Matrix $\widetilde{R}_a(p)$ lautet:

$$p^2_{a,y_1,y_2}(p) = 1 - det\left[\widetilde{\boldsymbol{R}}_a(p)\right] \tag{2.35}$$

Dies lässt sich für den allgemeinen Fall wie folgt beschreiben:

$$\begin{aligned} p^2_{a,y_1:y_N}(p) &= 1 - det\left[\widetilde{\boldsymbol{R}}_a(p)\right] \\ &= \frac{1 - det\left[\boldsymbol{R}_a(p)\right]}{\prod_{n=1}^{N}\sigma^2_{y_n}} \end{aligned} \tag{2.36}$$

Der MCCC hat die folgenden Eigenschaften:

- $0 \leq p^2_{a,y_1:y_N}(p) \leq 1$
- Sobald zwei oder mehr Signale perfekt korreliert sind, ist $p^2_{a,y_1:y_N}(p)$=1
- Sollten alle Signale unkorreliert miteinander sein, ist $p^2_{a,y_1:y_N}(p)$=0
- Ist ein Signal unkorreliert mit den N-1 anderen, so gibt der MCCC die Korrelation zwischen den N-1 verbliebenen Signalen an.

Auf der Grundlage dieser Eigenschaften lässt sich eine Schätzung der TDOA anhand des MCCC wie folgt anstellen:

$$\hat{\tau}^{MCCC} = \underset{p}{\operatorname{argmax}}\, p^2_{a,y_1:y_N}(p) \tag{2.37}$$

Wird die Gleichung 2.35 eingesetzt, ergibt sich:

$$\begin{aligned}
\hat{\tau}^{MCCC} &= \underset{p}{\operatorname{argmax}}\, p^2_{y_1:y_N}(p) \\
&= \underset{p}{\operatorname{argmax}}\, \left\{1 - det\left[\widetilde{\boldsymbol{R}}_a(p)\right]\right\} \\
&= \underset{p}{\operatorname{argmax}}\, \left\{\frac{1 - det\left[\boldsymbol{R}_a(p)\right]}{\prod_{n=1}^{N}\sigma^2_{y_n}}\right\} \\
&= \underset{p}{\operatorname{argmin}}\, det\left[\widetilde{\boldsymbol{R}}_a(p)\right] \\
&= \underset{p}{\operatorname{argmin}}\, det\left[\boldsymbol{R}_a(p)\right]
\end{aligned} \tag{2.38}$$

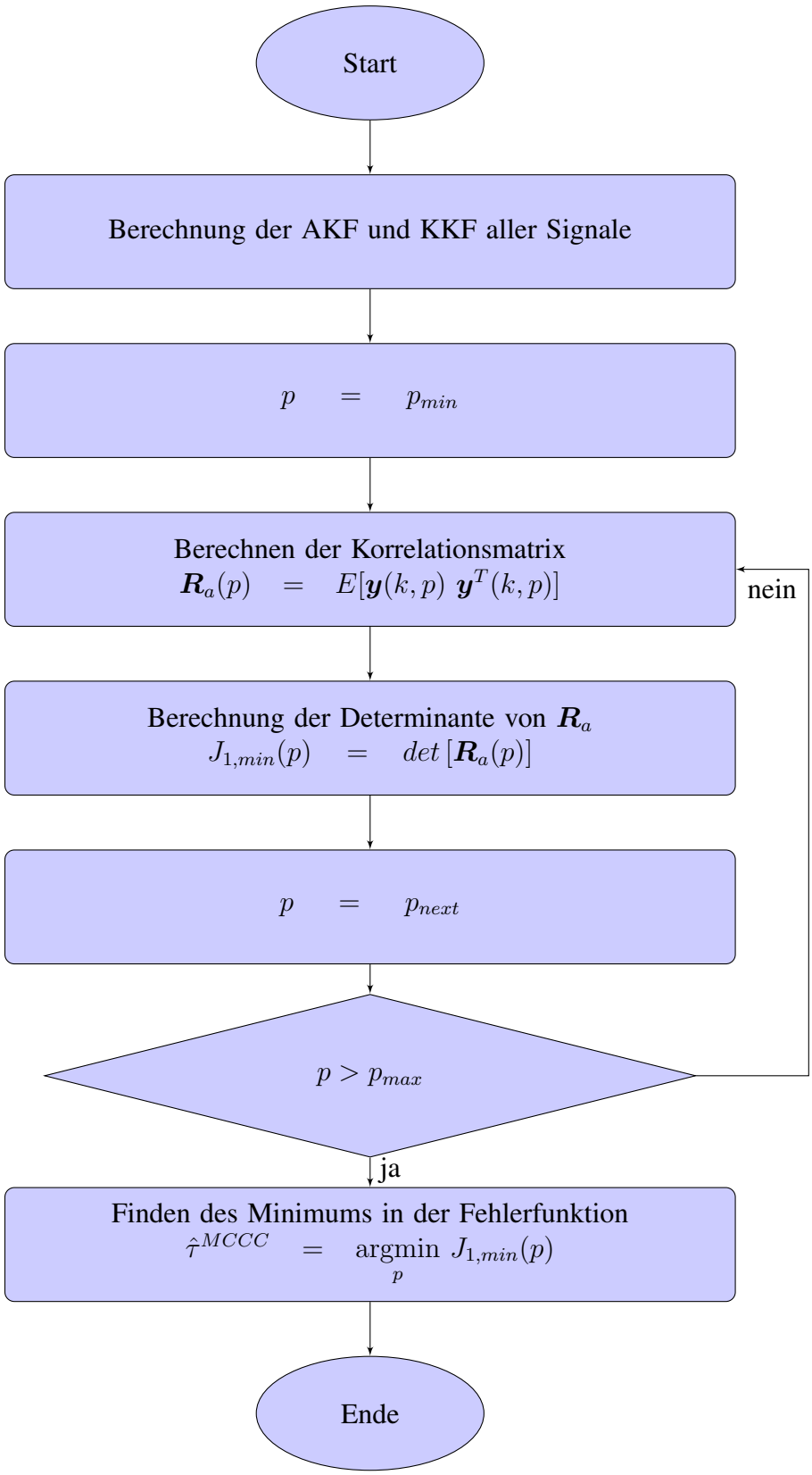


Abbildung 2.5.: Ablauf des MCCC

2.3.4. Steered Response Power

Die Idee hinter der Steered Response Power-Methode ist es, mit einem einfachen Delay and Sum Beamformer die gesamte Umgebung abzutasten und für jeden abgetasteten Winkel die Signalleistung am Ausgang des Beamformers zu bestimmen. Diese ist genau dann maximal, wenn der Beamformer auf die Signalquelle ausgerichtet ist.[6][13] Somit lässt sich eine Funktion der Leistung über den angenommenen Winkel wie folgt darstellen:

$$z_\tau(k) = \sum_{n=1}^{N} w_{\tau,k} \cdot y_n(k - \mathcal{F}_n(\tau)) \tag{2.39}$$

Dabei steuern die Verzögerungen $\mathcal{F}_n(\tau)$ den Beamformer und $w_{\tau,n}$ gewichtet die Signale abhängig vom Winkel.
Es lässt sich eine Funktion von der Leistung des Beamformers über die angenommene TDOA wie folgt bilden:

$$\begin{aligned} \boldsymbol{S}_{DSB}(p) &= E[z_\tau^2[k]] \\ &= \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} w_{\tau,k,n1} w_{\tau,k,n2} \cdot E[y_{n1}(k - \mathcal{F}_{n1}(\tau)) y_{n2}(k - \mathcal{F}_{n2}(\tau))] \end{aligned} \tag{2.40}$$

Dieser Ausdruck lässt sich folgendermaßen in Matrixschreibweise darstellen:

$$\boldsymbol{S}_{DSB}(p) = \boldsymbol{w}_\tau^T \boldsymbol{R}_a(p) \boldsymbol{w}_\tau \tag{2.41}$$

Klassischerweise ist die SRP so definiert, dass die Gewichte für alle Winkel zu 1 gewählt werden:

$$\tau^{SRP} = \underset{p}{\operatorname{argmin}}\ \mathbf{1}^T \boldsymbol{R}_a(p) \mathbf{1} \tag{2.42}$$

Dies ist gleichbedeutend mit der Summe aller Elemente von $\boldsymbol{R}_a(p)$

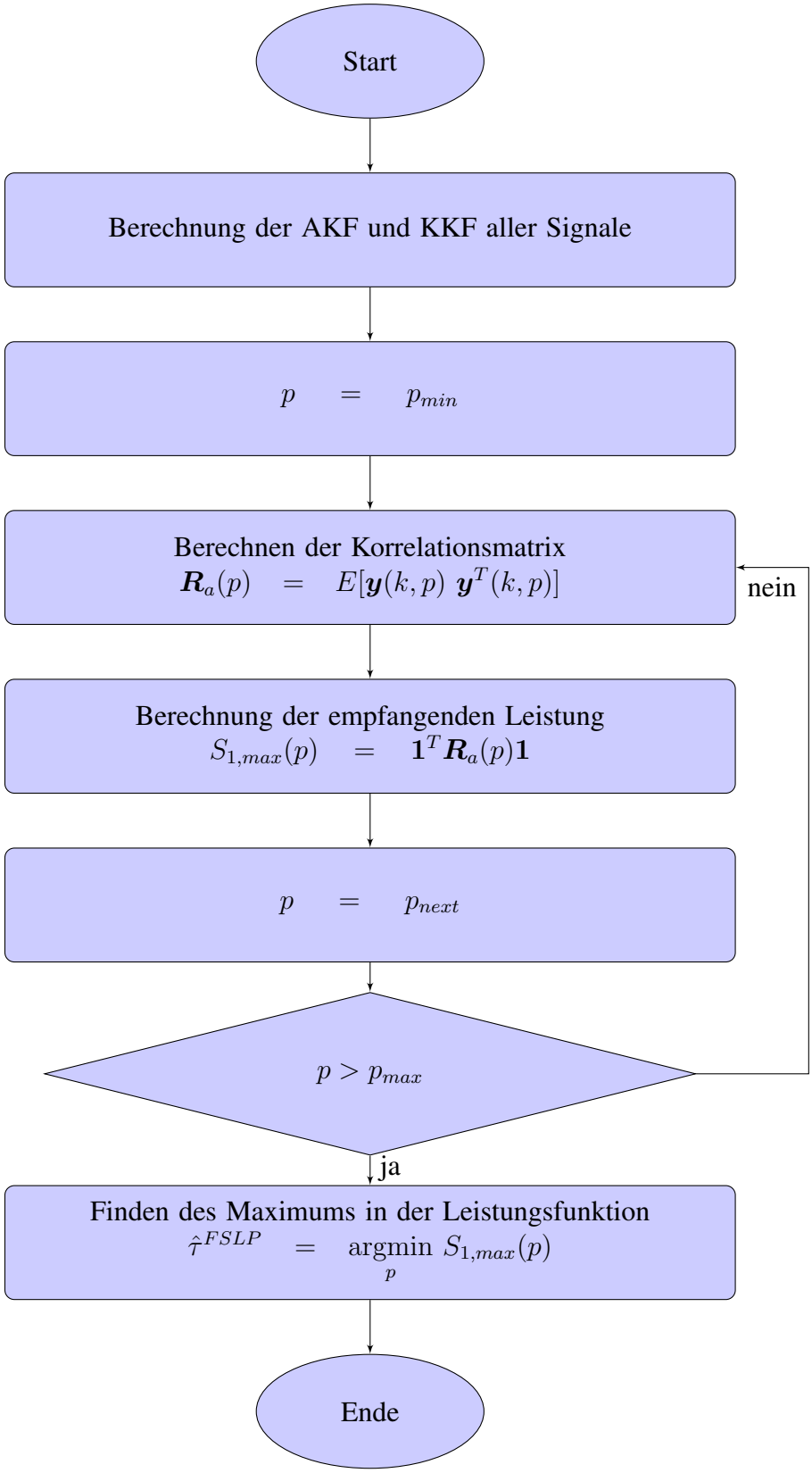


Abbildung 2.6.: Ablauf der SRP

2.4. Unterraumbasierte Verfahren

In diesem Abschnitt geht es um Verfahren zur Bestimmung der DOA, welche auf den in Abschnitt A.2 vorgestellten Methoden zur Zerlegung der Kovarianzmatrix in Signal- und Rauschunterraum, basieren.

2.4.1. Komplexes Signalmodel im Frequenzbereich

Da die unterraumbasierten Verfahren im Frequenzbereich arbeiten, ist es sinnvoll, ein entsprechendes Signalmodell zu definieren, welches auf dem in Abschnitt 2.3.1 vorgestellten basiert. Es soll zusätzlich mehrere Quellen gleichzeitig beinhalten, da dies für die Herleitung der Algorithmen notwendig ist.
Eine Einschränkung für die Modellierung des Signalmodells ist die Bandbreite des Quellsignals. Diese muss sehr klein sein.
Zunächst wird das Signalmodell aus Gleichung 2.9 herangezogen, welches mittels DTFT in den Frequenzbereich transformiert wird.

$$y_n(k) = x_n(k) + \upsilon_n(k) \circ\!\!-\!\!\bullet\, Y_n(f) = X_n(f) + \Upsilon_n(f),\ n = 1, 2, \ldots, N \tag{2.43}$$

Nun geht es darum, einen Weg zu finden, um die Verschiebung im Zeitbereich im Frequenzbereich zu modellieren. Dabei hilft der Verschiebungssatz der Fouriertransformation [31, S. 574].

$$f(t - \tau) \circ\!\!-\!\!\bullet\, F(\omega) \cdot \mathrm{e}^{-j\omega\tau} \tag{2.44}$$

Das Signal am Mikrofon n lässt sich demnach schreiben als [5, S. 201]:

$$\begin{aligned} Y_n(f, \phi) &= a_n S(f) \cdot \mathrm{e}^{-j\omega(t + \mathcal{F}_n(\tau(\phi)))} + \Upsilon_n(f) \\ &= a_n S(f) \cdot \mathrm{e}^{-j\omega t} \cdot \mathrm{e}^{-j\omega\mathcal{F}_n(\tau(\phi))} + \Upsilon_n(f) \end{aligned} \tag{2.45}$$

In Matrixschreibweise für sämtliche Mikrofone lässt sich das Signal wie folgt beschreiben:

$$\begin{pmatrix} Y_1(f,\phi) \\ Y_2(f,\phi) \\ Y_3(f,\phi) \\ \vdots \\ Y_N(f,\phi) \end{pmatrix} = \begin{pmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ 0 & 0 & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & a_N \end{pmatrix} \begin{pmatrix} 1 \\ \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi))} \\ \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi))} \\ \vdots \\ \mathrm{e}^{-j\omega\mathcal{F}_N(\tau(\phi))} \end{pmatrix} \mathrm{e}^{-j\omega t} \cdot S(f) + \begin{pmatrix} \Upsilon_1(f) \\ \Upsilon_2(f) \\ \Upsilon_3(f) \\ \vdots \\ \Upsilon_N(f) \end{pmatrix} \tag{2.46}$$

Der Einfachheit halber soll ab sofort die Annahme getroffen werden, dass $a_1 = a_2 \cdots = a_N = 1$ gilt. Das bedeutet, dass die Dämpfung jedes Eingangssignals identisch ist. Außerdem wird der Term $\mathrm{e}^{-j\omega t}$ vernachlässigt, da dieser lediglich die Laufzeit des Signals von der Quelle zu dem ersten Mikrofon beschreibt und somit keinerlei Nutzen zur Bestimmung der DOA hat. Damit vereinfacht sich Gleichung 2.46 zu:

$$\boldsymbol{Y}(f,\phi) = \boldsymbol{\varsigma}(\tau(\phi)) \cdot S(f) + \boldsymbol{\Upsilon}(f) \tag{2.47}$$

Wird nun eine Erweiterung auf D Quellen vorgenommen, lässt sich $\boldsymbol{Y}(f)$ als Summe darstellen.

$$\boldsymbol{Y}(f,\phi) = \sum_{i=1}^{D} \boldsymbol{\varsigma}(\tau(\phi_i)) S_i(f) + \boldsymbol{\Upsilon}(f) \tag{2.48}$$

Darin ist $\tau(\phi_i)$ die aus dem Winkel zwischen der jeweiligen Quelle und dem Mikrofonarray resultierende Laufzeitdifferenz zwischen zwei Mikrofonen, welche von dem Winkel der Quelle und dem Abstand der Mikrofone abhängt.
Das Eingangssignal bei mehreren Quellen lässt sich in Matrixschreibweise wie folgt ausdrücken:

$$\boldsymbol{Y}(f,\phi) = \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f) + \boldsymbol{\Upsilon}(f) \tag{2.49}$$

Darin ist $\boldsymbol{S}(f) = [S_1(f)\ S_2(f)\ \ldots\ S_D(f)]^T$ ein Vektor der Dimension D, bestehend aus den Quellsignalen.
$\boldsymbol{\Upsilon}(f) = [\Upsilon_1(f)\ \Upsilon_2(f)\ \ldots\ \Upsilon_N(f)]^T$ ist ein Vektor der Dimension N, welcher die individuellen Rauschanteile für jedes Mikrofon beinhaltet.

Die Matrix $\boldsymbol{\Sigma}(\tau)$ beinhaltet die durch die jeweiligen Richtungen der Quellen resultierenden Zeitverschiebungen an den einzelnen Mikrofonen.

$$\boldsymbol{\Sigma}(\tau(\phi)) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi_1))} & \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi_2))} & \cdots & \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi_D))} \\ \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi_1))} & \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi_2))} & \cdots & \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi_D))} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{e}^{-j\omega\mathcal{F}_N(\tau(\phi_1))} & \mathrm{e}^{-j\omega\mathcal{F}_N(\tau(\phi_2))} & \cdots & \mathrm{e}^{-j\omega\mathcal{F}_N(\tau(\phi_D))} \end{pmatrix} \tag{2.50}$$

2.4.2. MUSIC

MUSIC steht für **MU**ltiple **SI**gnal **C**lassification und ist wohl eines der bekanntesten Verfahren zur Quelllokalisierung. Dieses wurde erstmals im Jahr 1979 von Ralph O. Schmidt vorgestellt und wird zum Beispiel in [1] erläutert, woran die nachfolgenden Ausführungen angelehnt sind.
Aus dem in Abschnitt 2.4.1 vorgestellten Signalmodell lässt sich die Korrelations- bzw. Kovarianzmatrix[1] folgendermaßen bilden:

$$\boldsymbol{R}_a(f,\phi) = E\left[\boldsymbol{Y}_n(f,\phi)\boldsymbol{Y}_n^H(f,\phi)\right] \tag{2.51}$$

Nun kann für $\boldsymbol{Y}_n(f,\phi)$ der Ausdruck $\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f) + \boldsymbol{\Upsilon}(f)$ eingesetzt werden. Somit wird aus Gleichung 2.51:

$$\boldsymbol{R}_a(f,\phi) = E\left[(\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f) + \boldsymbol{\Upsilon}(f))(\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f) + \boldsymbol{\Upsilon}(f))^H\right] \tag{2.52}$$

Weiteres Auflösen des Ausdrucks führt zu:

$$\begin{aligned} \boldsymbol{R}_a(f,\phi) =& E\left[(\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f) + \boldsymbol{\Upsilon}(f))(\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{S}^H(f) + \boldsymbol{\Upsilon}^H(f))\right] \\ =& E\left[\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{S}^H(f)\right] + \\ & E\left[\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f)\boldsymbol{\Upsilon}^H(f)\right] + \\ & E\left[\boldsymbol{\Upsilon}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{S}^H(f)\right] + \\ & E\left[\boldsymbol{\Upsilon}(f)\boldsymbol{\Upsilon}^H(f)\right] \end{aligned} \tag{2.53}$$

[1] Für mittelwertfreie Signale, von welchen hier ausgegangen wird, sind Kovarianz- und Korrelationsmatrix identisch.

In Gleichung 2.53 sind einige Terme, in denen Signal- und Rauschanteile vorkommen. Da zu erwarten ist, dass das Rauschen und die Signale gänzlich unkorreliert (orthogonal zueinander) sind, gilt:

$$\begin{aligned} & E\left[\boldsymbol{\Upsilon}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{S}^H(f)\right] \\ = & E\left[\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f)\boldsymbol{\Upsilon}^H(f)\right] \\ = & \ 0 \end{aligned} \tag{2.54}$$

Gleichung 2.53 vereinfacht sich somit zu:

$$\boldsymbol{R}_a(f,\phi) = E\left[\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{S}^H(f)\right] + E\left[\boldsymbol{\Upsilon}(f)\boldsymbol{\Upsilon}^H(f)\right] \tag{2.55}$$

$$= \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A}_s(f)\boldsymbol{\Sigma}^H(\tau(\phi)) + \sigma^2\boldsymbol{I} \tag{2.56}$$

Darin ist $\boldsymbol{I}$ die N x N-dimensionale Einheitsmatrix. σ^2 ist die Varianz des Rauschens und $\boldsymbol{A}_s$ ist die Autokorrelationsmatrix der D Quellsignale mit der Dimension D x D.

$$\boldsymbol{A}_s(f) = \begin{pmatrix} E\left[|S_1(f)|^2\right] & 0 & \dots & 0 \\ 0 & E\left[|S_2(f)|^2\right] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E\left[|S_D(f)|^2\right] \end{pmatrix} \tag{2.57}$$

Der Ausdruck $\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A}_s(f)\boldsymbol{\Sigma}^H(\tau(\phi))$ wird zusammengefasst als:

$$\boldsymbol{R}_{a,s}(f,\phi) = \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A}_s(f)\boldsymbol{\Sigma}^H(\tau(\phi)) \tag{2.58}$$

Die Kovarianzmatrix $\boldsymbol{R}_a(f,\phi)$ ist eine Matrix der Dimension N x N, welche, wenn $\boldsymbol{A_s}(f)$ den Rang D hat, ebenfalls den Rang D hat. Das bedeutet, dass $\boldsymbol{R}_a(f,\phi)$ genau N-D Eigenvektoren, welche mit dem Nulleigenwert korrespondieren, aufweist.
Angenommen $\boldsymbol{q_m}$ ist ein zum Nulleigenwert korrespondierender Eigenvektor, denn gilt:

$$\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A_s}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{q_m} = 0 \tag{2.59}$$

$$\implies \boldsymbol{q_m}^H\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{S}(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{q_m} = 0 \tag{2.60}$$

$$\implies \boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{q_m} = 0 \tag{2.61}$$

Dies gilt, da $\boldsymbol{A}_s(f)$ positiv definit ist. Gleichung 2.61 zeigt, dass alle N-D Eigenvektoren $\boldsymbol{q_m}$, welche zum Nulleigenwert korrespondieren, orthogonal zu den D Steering-Vektoren aus $\boldsymbol{\Sigma}(\tau(\phi))$ sind.
Wenn nun $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_N$ die Eigenwerte der Kovarianzmatrix $\boldsymbol{R}_a(f,\phi)$ und $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ die Eigenwerte von $\boldsymbol{R}_{a,s}(f,\phi)$ sind, lässt sich zeigen, dass sich die Eigenwerte der Kovarianzmatrix wie folgt zusammensetzen:

$$\eta_m = \lambda_m + \sigma^2 \tag{2.62}$$

Dafür wird $\boldsymbol{Q}$ als Matrix aus den Eigenvektoren von $\boldsymbol{R_{a,s}}(f,\phi)$ definiert. Es gilt für jeden Eigenvektor $\boldsymbol{q_m} \in \boldsymbol{Q}$:

$$\boldsymbol{R}_{a,s}(f,\phi)\boldsymbol{q_m} = \lambda \boldsymbol{q_m} \tag{2.63}$$

$$\begin{aligned} \Longrightarrow \boldsymbol{R}_a(f,\phi)\boldsymbol{q_m} &= \boldsymbol{R}_{a,s}(f,\phi)\boldsymbol{q_m} + \sigma^2 \boldsymbol{I}\boldsymbol{q_m} \\ &= (\lambda_m + \sigma^2)\boldsymbol{q_m} \end{aligned} \tag{2.64}$$

Das bedeutet, dass jeder Eigenvektor von $\boldsymbol{R}_{a,s}(f,\phi)$ auch Eigenvektor von $\boldsymbol{R}_a(f,\phi)$ mit dem zugehörigem Eigenwert $\lambda_m + \sigma^2$ ist. Wird demnach $\boldsymbol{R}_{a,s}(f,\phi)$ in die Form

$$\boldsymbol{R}_{a,s}(f,\phi) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^H \tag{2.65}$$

gebracht, so gilt für $\boldsymbol{R}_a(f,\phi)$:

$$\boldsymbol{R}_a(f,\phi) = \boldsymbol{Q}\left[\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I}\right]\boldsymbol{Q}^H \tag{2.66}$$

$$= \boldsymbol{Q}\begin{pmatrix} \lambda_1+\sigma^2 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \lambda_2+\sigma^2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \lambda_D+\sigma^2 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & \sigma^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & \sigma^2 \end{pmatrix}\boldsymbol{Q}^H \tag{2.67}$$

Das Rauschen sorgt folglich nur für eine Anhebung der Eigenwerte, welche zu den Quellen korrespondieren und dafür, dass die Matrix $\boldsymbol{R}_a(f,\phi)$ vollen Rang hat.

Diese hat somit keine Eigenvektoren, die mit dem Nulleigenwert korrespondieren. Die N-D Eigenvektoren, die zum Rauschunterraum gehören, sind orthogonal zum Signalunterraum. Somit gilt für diese analog zu Gleichung 2.61:

$$\boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A}_s(f)\boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{q}_{m,n} = 0 \tag{2.68}$$

$$\implies \boldsymbol{\Sigma}^H(\tau(\phi))\boldsymbol{q}_{m,n} = 0 \tag{2.69}$$

Das bedeutet, dass der vom Rauschen aufgespannte Unterraum orthogonal zu dem Unterraum aufgespannt durch die Steering-Vektoren aus $\boldsymbol{\Sigma}(\tau(\phi))$ ist.

$$span\left[\boldsymbol{q}_{D+1}\dots\boldsymbol{q}_N\right] \perp span\left[\boldsymbol{\varsigma}(\tau(\phi_1))\dots\boldsymbol{\varsigma}(\tau(\phi_D))\right] \tag{2.70}$$

Da bekanntermaßen die Eigenvektoren der Kovarianzmatrix $\boldsymbol{R}_a(f,\phi)$ orthogonal sind, gilt dies auch für die jeweils dadurch aufgespannten Unterräume.
Die jeweiligen Eigenvektoren werden zu Matrizen zusammengefasst, sodass die Eigenwertematrix $\boldsymbol{Q}$ in einen Rausch- und einen Signalanteil aufgeteilt wird.

$$\boldsymbol{Q}_s = [\boldsymbol{q}_1\dots\boldsymbol{q}_D] \tag{2.71}$$

$$\boldsymbol{Q}_n = \left[\boldsymbol{q}_{D+1}\dots\boldsymbol{q}_N\right] \tag{2.72}$$

Es gilt:

$$\boldsymbol{Q}_s \perp \boldsymbol{Q}_n \tag{2.73}$$

Wenn der durch die Eigenvektoren, welche zum Rauschen korrespondieren, aufgespannte Unterraum orthogonal zu dem von den Steering-Vektoren aufgespannten Unterraum ist, so muss gelten:

$$span\left[\boldsymbol{q}_1\dots\boldsymbol{q}_D\right] = span\left[\boldsymbol{\varsigma}(\tau(\phi_1))\dots\boldsymbol{\varsigma}(\tau(\phi_D))\right] \tag{2.74}$$

MUSIC nutzt die soeben aufgezeigten Zusammenhänge aus, indem zunächst die Kovarianzmatrix der N Eingangssignale geschätzt wird, um anschließend deren Eigenwerte zu bestimmen. Diese können nun in N-D zum Rauschen korrespondierende Eigenwerte und D zu den Signalen korrespondierende Eigenwerte aufgeteilt werden. Dafür bietet es sich an,

diese zunächst der Größe nach zu sortieren und anhand einer Schwelle dem Signal- bzw. dem Rauschunterraum zuzuordnen. Nach Ermittlung der zugehörigen Eigenvektoren werden diese ebenfalls aufgeteilt und zu $\boldsymbol{Q}_s$ und $\boldsymbol{Q}_n$ zusammengefasst. Nun müssen Steering-Vektoren gefunden werden, die orthogonal zum Rauschunterraum sind.
In der Praxis wird es durch diverse Fehlereinflüsse nicht dazu kommen, so einen Vektor zu finden. Es geht vielmehr darum einen Vektor zu finden, auf den dies bestmöglich zutrifft. Dafür wird ein Fehlervektor definiert, welcher Aufschluss über die Nähe zur Orthogonalität gibt. Die euklidische Norm des Fehlervektors lautet wie folgt:

$$\begin{aligned} \|\varepsilon(\varsigma(\tau(\phi)))\|^2 &= \sum_{m=D+q}^{N} |\boldsymbol{q}_m^H \boldsymbol{\varsigma}(\tau(\phi))|^2 \\ &= \boldsymbol{\varsigma}(\tau(\phi))^H \cdot \boldsymbol{P} \cdot \boldsymbol{\varsigma}(\tau(\phi)) \end{aligned} \tag{2.75}$$

Darin ist:

$$\boldsymbol{P} = \boldsymbol{Q}_n \cdot \boldsymbol{Q}_n^H \tag{2.76}$$

Mithilfe dieses Zusammenhangs lässt sich nun der Steering-Vektor finden, der den Fehler minimal werden lässt. Werden einige Steering-Vektoren geschätzt und getestet, so lassen sich die zugehörigen Werte des euklidischen Fehlers über dem Winkel ϕ, welcher Parameter der Steering-Vektoren ist, auftragen. Dazu wurde ein Pseudospektrum, das sogenannte MUSIC-Spektrum definiert, welches das Reziprok des euklidischen Fehlers über den möglichen DOAs darstellt.

$$S_{MUSIC}(\phi) = \frac{1}{\boldsymbol{\varsigma}(\tau(\phi))^H \cdot \boldsymbol{P} \cdot \boldsymbol{\varsigma}(\tau(\phi))} = \frac{1}{\boldsymbol{\varsigma}(\tau(\phi))^H \cdot \boldsymbol{Q}_n \cdot \boldsymbol{Q}_n^H \cdot \boldsymbol{\varsigma}(\tau(\phi))} \tag{2.77}$$

Steering-Vektoren, die mit der Richtung einer Quelle korrespondieren, sorgen dafür, dass sich deutliche Maxima im Pseudospektrum ausbilden.

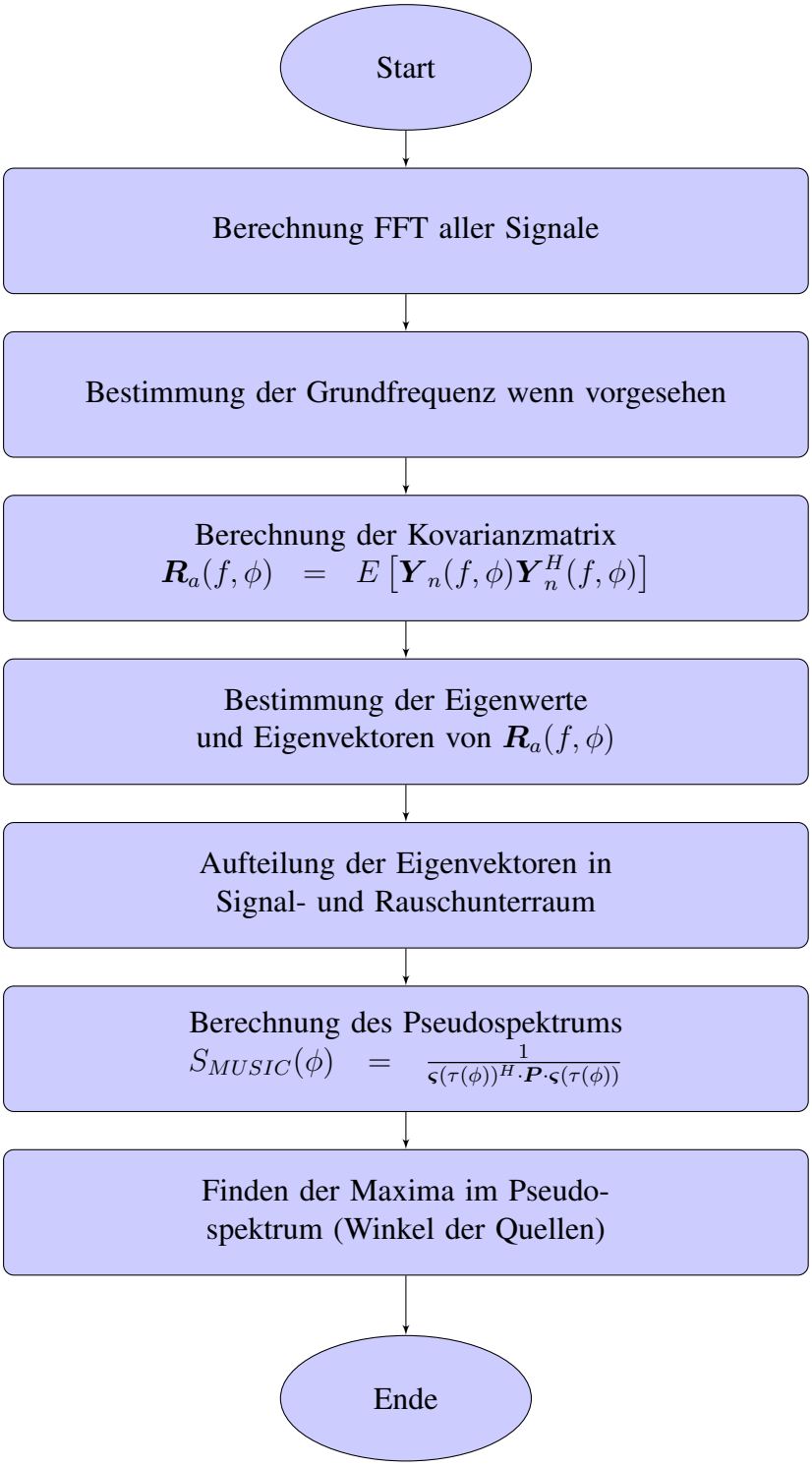


Abbildung 2.7.: Ablauf MUSIC

2.4.3. ROOT-MUSIC

Der ROOT-MUSIC Algorithmus basiert grundsätzlich auf der Herleitung des vorherigen Abschnitts. Der Unterschied ist die Suche nach den Maxima des Pseudospektrums. Beschrieben wird dieses Verfahren z.B. in [42]. Bei Betrachtung von Gleichung 2.77 sind die Stellen, an denen das Pseudospektrum Maxima hat, gleichbedeutend mit den Stellen, an denen der Nenner der Gleichung gegen null geht. ROOT-MUSIC sucht genau diese Stellen, indem die Nullstellen des Nenners gesucht werden, anstatt verschiedene Steering-Vektoren zu testen.
In den Steering-Vektoren stehen die vom Array-Element abhängigen Verzögerungen als exponential Terme, welche nach

$$z = \mathrm{e}^{j\omega\tau} = \mathrm{e}^{j\frac{2\pi d}{\lambda} \cdot sin(\phi)} \tag{2.78}$$

auch in folgender Form dargestellt werden können:

$$\varsigma(\boldsymbol{\tau}(\boldsymbol{\phi})) = \begin{pmatrix} 1 & \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi))} & \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi))} & \cdots & \mathrm{e}^{-j\omega\mathcal{F}_N(\tau(\phi))} \end{pmatrix}^T \tag{2.79}$$

$$= \begin{pmatrix} 1 & z \cdots & z^{N-1} \end{pmatrix} \tag{2.80}$$

Dadurch kann der Nenner-Term aus Gleichung 2.77 wie folgt geschrieben werden:

$$P_{ROOT} = \begin{pmatrix} 1 & z^{-1} \cdots & z^{-(N-1)} \end{pmatrix} \cdot \boldsymbol{P} \cdot \begin{pmatrix} 1 & z \cdots & z^{N-1} \end{pmatrix} \tag{2.81}$$

Ausgerechnet ergibt sich für P_{ROOT} ein Polynom mit N-1 konjugiert komplexen Nullstellen. Die Nullstellen würden idealerweise auf dem Einheitskreis liegen. Durch das Vorhandensein von Rauschen und weiterer Fehlereinflüsse ist dies aber nicht gewährleistet, sodass die D dem Einheitskreis am nächsten liegenden Nullstellen, welche allerdings noch innerhalb liegen, den D Quellen zugeordnet werden können. [2]
Diese Nullstellen sind komplexe Werte in z und müssen entsprechend Gleichung 2.78 zurückgerechnet werden. Dies geschieht durch Auflösung der Gleichung nach ϕ.

$$\phi_k = sin^{-1}\left(\frac{\lambda}{2\pi d} arg(z_k)\right) \tag{2.82}$$

ROOT-MUSIC hat eine entschiedene Einschränkung. Durch die Substitution in z ist es nur möglich ganze Vielfache von Verzögerungen darzustellen. $z^{-1}, z^{-2}, \cdots, z^{-(N-1)}$. Somit ist ROOT-MUSIC nur für lineare Mikrofonarrays geeignet, bei denen der Abstand entweder gleichmäßig ist oder ganzzahlige Vielfache voneinander darstellt.[41, S. 162]

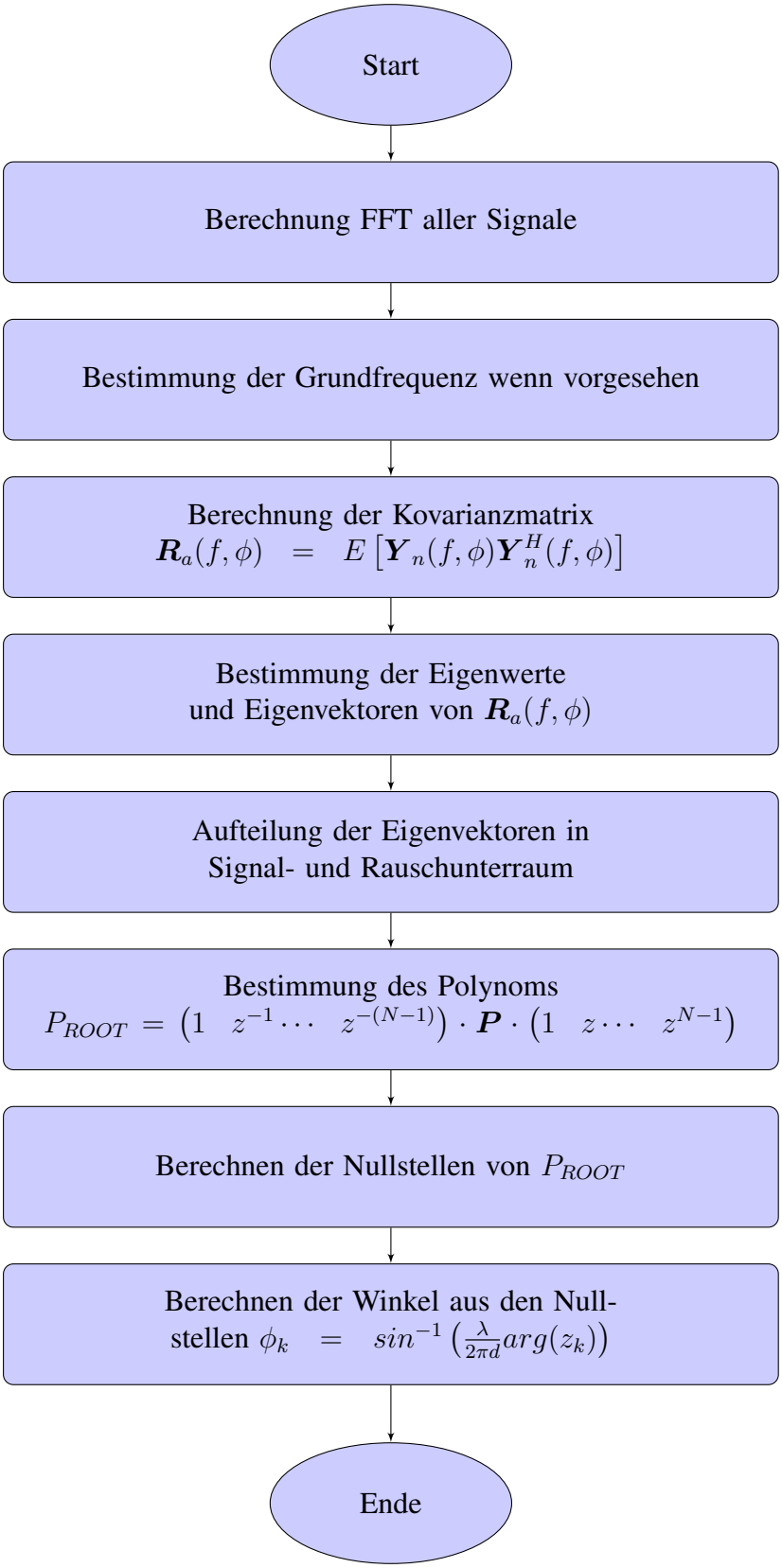


Abbildung 2.8.: Ablauf ROOT-MUSIC

2.4.4. ESPRIT

ESPRIT wurde erstmals im Jahr 1989 von Richard Roy und Thomas Kailath in [35] vorgestellt. Die folgenden Ausführungen orientieren sich daran und an [16]. Bei dem Algorithmus wird im Gegensatz zu MUSIC und ROOT-MUSIC nicht der Rauschunterraum zur Schätzung der DOA, sondern der Signalunterraum verwendet.
Der ESPRIT-Algorithmus beruht darauf, dass das Mikrofonarray in zwei Teile (Unterarrays) gleicher Struktur aufgeteilt wird. Diese müssen zueinander verschoben sein. Genau dieser Versatz wird nun im Signalmodell berücksichtigt.

$$\boldsymbol{y}_{Array1}(k) = \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{s}(k) + \boldsymbol{v}_{Array1}(k) \tag{2.83}$$

$$\boldsymbol{y}_{Array2}(k) = \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{D}\boldsymbol{s}(k) + \boldsymbol{v}_{Array2}(k) \tag{2.84}$$

Die Matrix $\boldsymbol{D}$ beschreibt hier die Verschiebung der an den Unterarrays empfangenden Signale.

$$\boldsymbol{D} = \begin{pmatrix} \mathrm{e}^{j\omega\tau_1} & 0 & \cdots & 0 \\ 0 & \mathrm{e}^{j\omega\tau_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{e}^{j\omega\tau_N} \end{pmatrix} \tag{2.85}$$

Durch Bestimmung von $\boldsymbol{D}$ lässt sich demnach die DOA schätzen. Zusammengefasst zu einem Ausdruck ergibt sich für die Eingangssignale:

$$\begin{aligned} \boldsymbol{b}(k) = \begin{pmatrix} \boldsymbol{y}_{Array1}(k) \\ \boldsymbol{y}_{Array2}(k) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{\Sigma}(\tau(\phi)) \\ \boldsymbol{D}\boldsymbol{\Sigma}(\tau(\phi)) \end{pmatrix} \boldsymbol{s}(k) + \boldsymbol{v}_b(k) \\ &= \tilde{\boldsymbol{\Sigma}}\boldsymbol{s}(k) + \boldsymbol{v}_b(k) \end{aligned} \tag{2.86}$$

Hiervon lässt sich analog zu den zuvor beschriebenen Abschnitten eine Kovarianzmatrix bilden.

$$\boldsymbol{R}_a = E\left[\boldsymbol{b}(k)\boldsymbol{b}(k)\right] \tag{2.87}$$

Diese kann ebenfalls in Rausch- und Signalunterraum zerlegt werden und es gilt weiter-

hin:

$$span(\tilde{\boldsymbol{\Sigma}}(\tau(\phi))) = span(\boldsymbol{Q}_s) \tag{2.88}$$

Folglich muss eine reguläre Matrix $\boldsymbol{T}$ der Dimension D x D existieren, sodass gilt:

$$\tilde{\boldsymbol{\Sigma}}(\tau(\phi))\boldsymbol{T} = \boldsymbol{Q}_s \tag{2.89}$$

Wird für $\tilde{\boldsymbol{\Sigma}}(\tau(\phi))$ wieder die in Gleichung 2.86 eingeführte Beziehung, eingesetzt ergibt sich:

$$\begin{pmatrix} \boldsymbol{\Sigma}(\tau(\phi)) \\ \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{D} \end{pmatrix} \boldsymbol{T} = \begin{pmatrix} \boldsymbol{Q}_{s,Array1} \\ \boldsymbol{Q}_{s,Array2} \end{pmatrix} \tag{2.90}$$

Demnach gilt:

$$span(\tilde{\boldsymbol{\Sigma}}(\tau(\phi))) = span(\boldsymbol{Q}_{s,Array1}) = span(\boldsymbol{Q}_{s,Array2}) \tag{2.91}$$

Es wird eine Matrix mit dem Rang D wie folgt definiert:

$$\boldsymbol{Q}_{Array1,Array2} = \left[\boldsymbol{Q}_{s,Array1} | \boldsymbol{Q}_{s,Array2}\right] \tag{2.92}$$

Das impliziert, dass es eine weitere Matrix $\boldsymbol{F}$ der Dimension 2D x D geben muss, die ebenfalls den Rang D hat und orthogonal zu $\boldsymbol{Q}_{Array1,Array2}$ ist.

$$0 = \left[\boldsymbol{Q}_{s,Array1} | \boldsymbol{Q}_{s,Array2}\right] \boldsymbol{F} = \boldsymbol{Q}_{s,Array1}\boldsymbol{F}_{Array1} + \boldsymbol{Q}_{s,Array2}\boldsymbol{F}_{Array2} \tag{2.93}$$

$$= \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{T}\boldsymbol{F}_{Array1} + \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{D}\boldsymbol{T}\boldsymbol{F}_{Array2} \tag{2.94}$$

Es wird $\boldsymbol{\Psi}$ wie folgt definiert:

$$\boldsymbol{\Psi} = -\boldsymbol{F}_{Array1}\boldsymbol{F}_{Array2}^{-1} \tag{2.95}$$

Somit lässt sich Gleichung 2.94 folgendermaßen umformen:

$$\boldsymbol{T}\boldsymbol{\Psi}\boldsymbol{T}^{-1} = \boldsymbol{D} \tag{2.96}$$

Daraus ist erkennbar, dass $\boldsymbol{\Psi}$ dieselben Eigenwerte wie $\boldsymbol{D}$ hat. Die Signaleigenschaften

sind nicht lineare Funktionen der Eigenwerte der Matrix $\boldsymbol{\Psi}$, welche die Matrix $\boldsymbol{Q}_{s,Array1}$ des ersten Unterarrays auf die Matrix $\boldsymbol{Q}_{s,Array1}$ des zweiten Unterarrays abbildet

$$\boldsymbol{Q}_{s,Array1} = \boldsymbol{Q}_{s,Array2}\boldsymbol{\Psi} \tag{2.97}$$

In der Praxis lässt sich dieses Gleichungssystem aufgrund von Rauschen und anderer Störeinflüsse nicht lösen. Es ergibt sich das Least-Squares-Problem:

$$\boldsymbol{\Psi} = (\boldsymbol{Q}^{H}_{s,Array2} \cdot \boldsymbol{Q}_{s,Array2})^{-1}(\boldsymbol{Q}^{H}_{s,Array2} \cdot \boldsymbol{Q}_{s,Array1}) \tag{2.98}$$

Die gesuchten Winkel sind in den Eigenwerten $\chi_1 \dots \chi_D$ von $\boldsymbol{\Psi}$ enthalten und lassen sich mit dem folgenden Zusammenhang berechnen.

$$\phi_k = sin^{-1}\left(\frac{\lambda}{2\pi d} arg(\chi_k)\right) \tag{2.99}$$

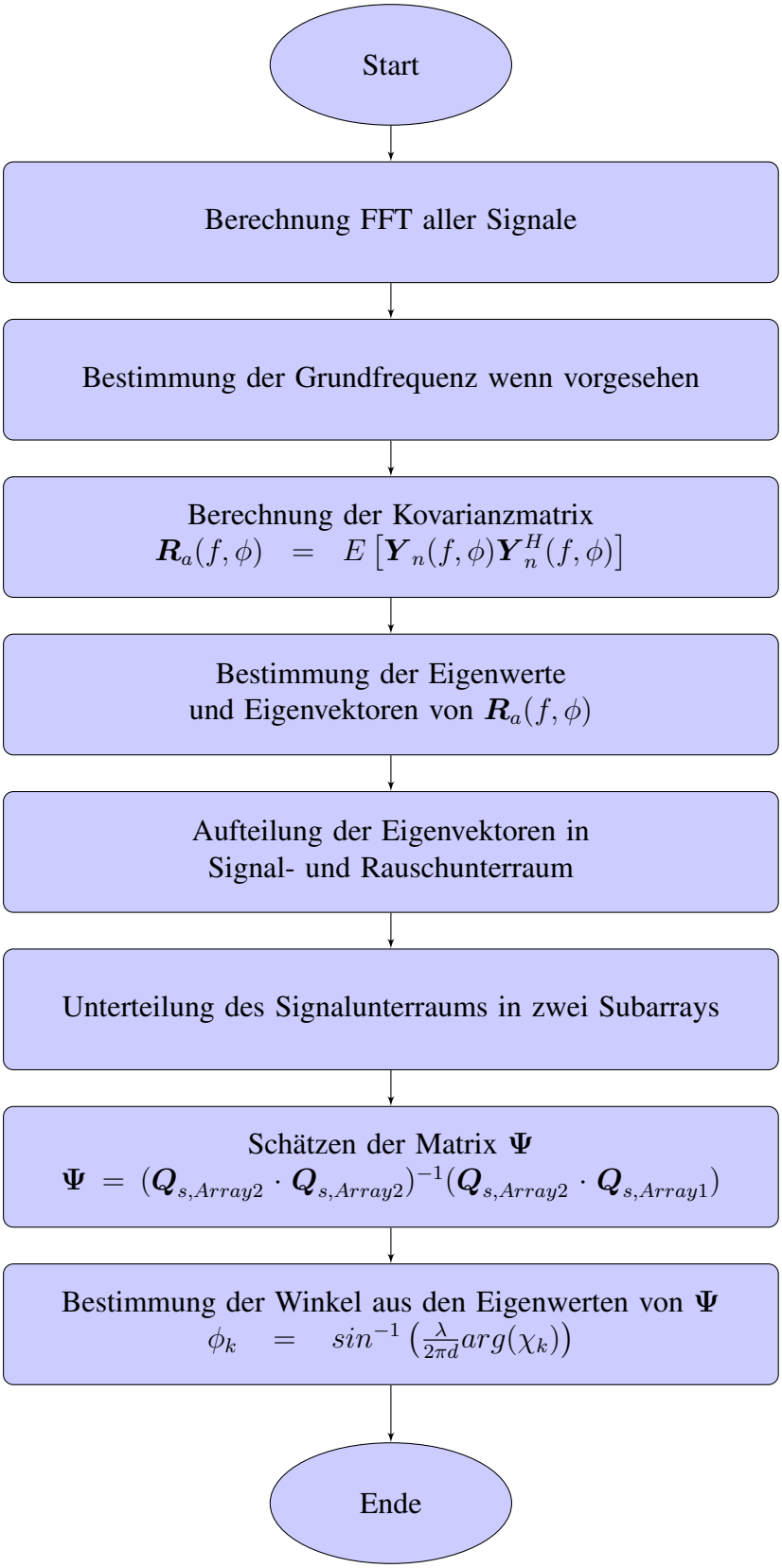


Abbildung 2.9.: Ablauf ESPRIT

2.4.5. Breitbandansatz

Alle bis hier vorgestellten unterraumbasierten Verfahren haben eine Gemeinsamkeit. Sie schätzen den Winkel des Signals für genau eine Wellenlänge. Um diese Methoden auch für breitbandige Signale anzuwenden, gibt es verschiedene Ansätze. Einer davon ist es, sehr schmalbandige Filter einzusetzen, um nur einzelne Frequenzen aus dem Signal zu betrachten. Dafür kann das Spektrum des eingehenden Signals nach signifikanten Stellen, an denen die Leistung am höchsten ist, untersucht werden. Diese Frequenzanteile können anschließend für die Schätzung der DOA genutzt werden. Natürlich werden in diesem Fall viele Informationen aus den Signalen verworfen. Um Informationen aus der gesamten Bandbreite des Signals zu nutzen, gibt es ebenfalls verschiedene Ansätze. Der inkohärente Ansatz beruht darauf, separate DOAs für unterschiedliche Schmalbänder zu berechnen und die Ergebnisse zusammenzuführen. Eine Einschränkung dieses Verfahrens ist die hohe Streuung der DOAs für die verschiedenen Frequenzen und die damit verbundene Notwendigkeit vieler Stichproben für eine gute Schätzung des Erwartungswerts [6, S. 185]. Der Ablauf der DOA-Schätzung mit MUSIC unter Verwendung eines inkohärenten Ansatzes ist, dass das Signal eines Mikrofons im Frequenzbereich auf die L größten Maxima untersucht wird. Anschließend werden L extrem schmalbandige Bandpassfilter genutzt, um die Eingangssignale jeweils auf die L Frequenzanteile mit der höchsten Leistung zu beschränken. Danach wird für jede der L Frequenzen die Kovarianzmatrix gebildet und die zugehörigen Eigenvektoren in Signal- und Rauschunterraum aufgeteilt. Es wird wiederum für alle L Rauschunterräume das MUSIC-Spektrum bestimmt. Zuletzt wird das gesamte MUSIC-Spektrum aus einer Kombination aller L bestimmt. Die gesuchten DOAs sind die Maxima. Es müssen folglich alle Verarbeitungsschritte bis auf die Transformation in den Frequenzbereich L mal durchgeführt werden.
Ein weiterer Ansatz ist die so genannte Coherent Signal Subspace Method (CSSM), welche in [6, S. 185 f.] und [9] beschrieben ist. Diese nutzt eine Transformations-Matrix $\boldsymbol{U}(f_i)$, um jeden Steering-Vektor $\boldsymbol{\varsigma}(f_i, \tau(\phi))$ auf einen dazu korrespondierenden Steering-Vektor $\boldsymbol{w}(f_0, , \tau(\phi))$ eines Referenzarrays, welches bei der Frequenz f_0 arbeitet, zu transformieren.

$$\boldsymbol{U}(f_i)\boldsymbol{\Sigma}(f_i, \tau(\phi)) = [\boldsymbol{w}(f_0, \tau(\phi_1)) \cdots \boldsymbol{w}(f_0, \tau(\phi_D))] = \boldsymbol{W}(f_0, \tau(\phi)) \tag{2.100}$$

Die Kovarianzmatrix wird nun aus den Kovarianzmatrizen für die einzelnen Frequenzen berechnet.

$$\boldsymbol{R}_a = \sum_{i=1}^{J} \alpha_i \boldsymbol{U}(f_i)\boldsymbol{R}_a(f_i)\boldsymbol{T}^H(f_i) \cong \boldsymbol{W}(f_0, \tau(\phi))\boldsymbol{A}_s\boldsymbol{W}^H(f_0, \tau(\phi)) + \sigma_\upsilon \boldsymbol{R}_{\upsilon\upsilon} \quad (2.101)$$

Auch hier gilt das Prinzip der Orthogonalität zwischen Signal- und Rauschunterraum. Der Rauschunterraum kann durch die Eigenwertzerlegung der folgenden Matrix geschätzt werden.

$$\boldsymbol{V}^{-1}\boldsymbol{R}_a(f_i)(\boldsymbol{V}^{-1})^H \quad (2.102)$$

Darin ist $\boldsymbol{V}$ die Matrix der Varianzen der Rauschkovarianzmatrix $\boldsymbol{\Upsilon}$, welche sich folgendermaßen bestimmen lässt:

$$\boldsymbol{R}_{\upsilon\upsilon} = \boldsymbol{V}\boldsymbol{V}^H = \sum_{i=1}^{J} \alpha_i \boldsymbol{U}(f_i)\boldsymbol{U}^H(f_i) \quad (2.103)$$

Es muss somit die Kovarianzmatrix für jede verwendete Frequenz aus dem Signal geschätzt werden, welche jeweils eigene Unterräume besitzt. Die DOA lässt sich unter Verwendung der CSSM für den MUSIC-Algorithmus wie folgt bestimmen:

$$\phi_{MUSIC} = \underset{\phi}{\operatorname{argmin}} \left\{ \frac{|\boldsymbol{Q}_n\boldsymbol{V}^{-1}\boldsymbol{w}(f_0, \tau(\phi_D))|_2}{|\boldsymbol{V}^{-1}\boldsymbol{w}(f_0, \tau(\phi_D))|_2} \right\} \quad (2.104)$$

Der Unterschied dieser und auch anderer kohärenter Methoden ist, dass die eigentliche Winkelberechnung nur einmalig stattfindet. Das Herausrechnen der Frequenzabhängigkeit erfordert jedoch ebenfalls Rechenaufwand, sodass der Komplexitätsunterschied durch diese Tatsache alleine nicht beschrieben werden kann.

3. Simulation der Algorithmen

In diesem Kapitel sollen die theoretisch behandelten Verfahren in MATLAB simuliert werden. Dazu werden die Verfahren zunächst implementiert und anhand verschiedener Testsignale verifiziert. Zudem wird der Einfluss verschiedener Umgebungsparameter auf die Verfahren untersucht. Am Ende des Kapitels soll jeweils eine Methode für unterraumbasierte Verfahren und eine für zeitbasierte Verfahren ausgewählt werden, welche für weitere Vergleiche stellvertretend für die übergeordneten Ansätze herangezogen werden.

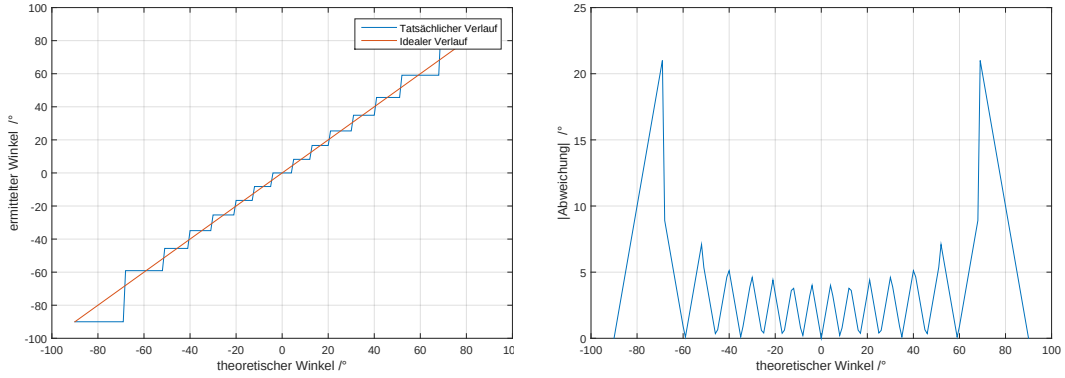
3.1. Simulation zeitbasierter Verfahren

3.1.1. Verifikation zeitbasierter Verfahren

Nach der Implementierung der Verfahren in der Simulationsumgebung muss zunächst die grundsätzliche Funktionalität festgestellt werden. Hierzu wurde ein synthetisch erzeugtes Signal mit einem Frequenzbereich von 100 bis 2000 Hz benutzt. Anschließend wurde dieses für 8 Mikrofone entsprechend der simulierten Winkel verzögert und als Eingangssignale genutzt. Es wurden alle Winkel zwischen -90° und +90° in Schritten von 1° eingestellt und die ermittelten Winkel über den eingestellten Winkeln aufgetragen. Die Parameter der Simulation lauten wie folgt:

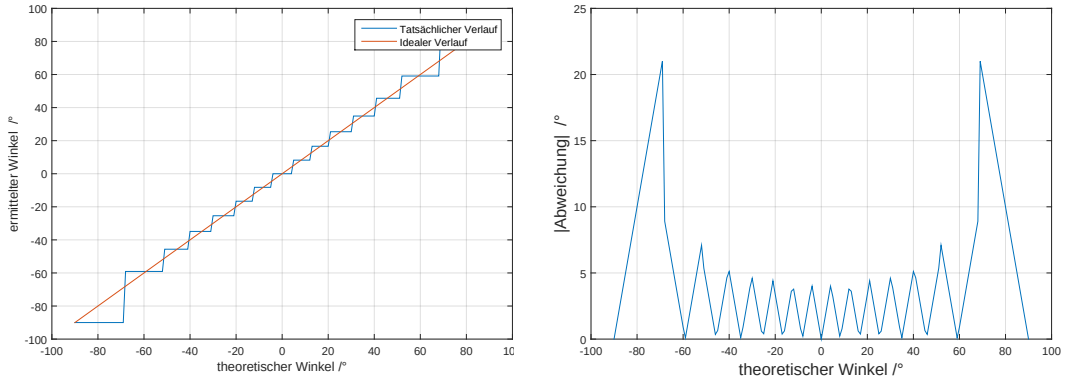
- Abtastfrequenz: 48 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz

- SNR: 20 dB



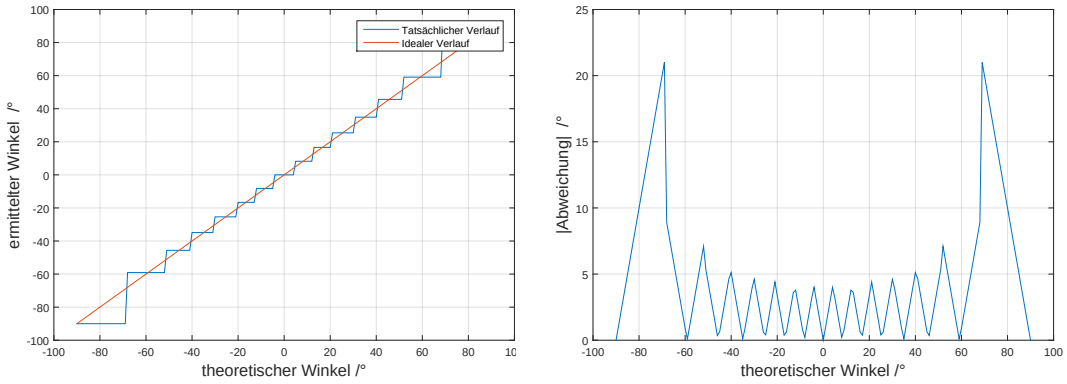
(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.1.: Verifizierung MCCC ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.2.: Verifizierung SLP ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.3.: Verifizierung SRP ermittelter und simulierter Winkel und absolute Abweichung

Es konnte gezeigt werden, dass alle Verfahren prinzipiell funktionieren. Es kann außerdem beobachtet werden, dass alle Verfahren exakt identische Ergebnisse liefern. Das ist allerdings noch keine Angabe über die Qualität der Verfahren. Alle Verfahren haben gemeinsam, dass der ermittelte Winkel in das durch die Abtastrate und den Mikrofonabstand vorgegebene Raster fallen muss. Das bedeutet hier, dass alle Verfahren über den gesamten Bereich korrekt funktioniert haben.

3.1.2. Simulation zeitbasierter Verfahren mit synthetischen Signalen

Im Folgenden sollen die implementierten Verfahren möglichst realitätsnah simuliert werden. Dazu werden - wie auch in einer Echtzeitanwendung - Blöcke der Länge K für jedes der N Mikrofone gebildet, welche als Eingangssignal für die Algorithmen dienen. Ziel ist es, die Qualität der Algorithmen anhand ihrer Standardabweichung zu bestimmen und mögliche Schwachstellen ausfindig zu machen. Zunächst soll für die Simulation ein synthetisch erzeugtes Signal verwendet werden, welches bandbegrenzt wurde. Die Simulationsparameter lauten wie folgt:

- Abtastfrequenz: 48 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8

- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 1000 Hz
- SNR: 20 dB

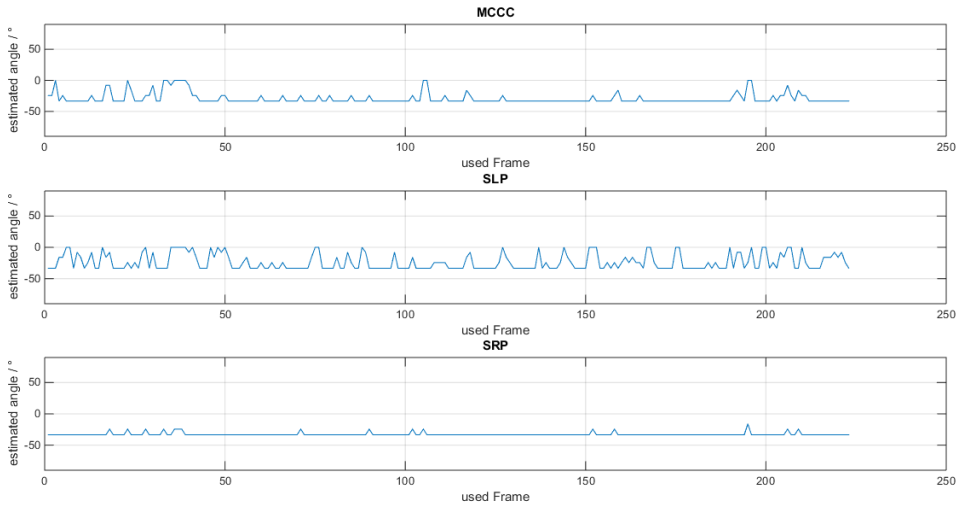


Abbildung 3.4.: Zeitbasierte Verfahren bei Quellsignal zwischen 100 und 1000 Hz

Tabelle 3.1.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal zwischen 100 und 1000 Hz

Verfahren	**Standardabweichung $\sigma/°$**
MCCC	8,7297
SLP	12,7758
SRP	2,1872

Abbildung 3.4 zeigt den ermittelten Winkel über die zur Ermittlung benutzten Rahmen. Die besten Ergebnisse liefert in dieser Simulation die SRP. SLP und MCCC scheinen fehleranfällig und weisen neben der zu erwartenden Streuung rund um den korrekten Winkel auch häufige Fehldetektionen eines Winkels bei $0°$ auf.
In der nächsten Simulation wird das Quellsignal verändert. Es wird ein breitbandigeres Signal mit Frequenzanteilen zwischen 100 und 2000 Hz verwendet. Die übrigen Parameter bleiben bestehen.

- Abtastfrequenz: 48 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz
- SNR: 20 dB

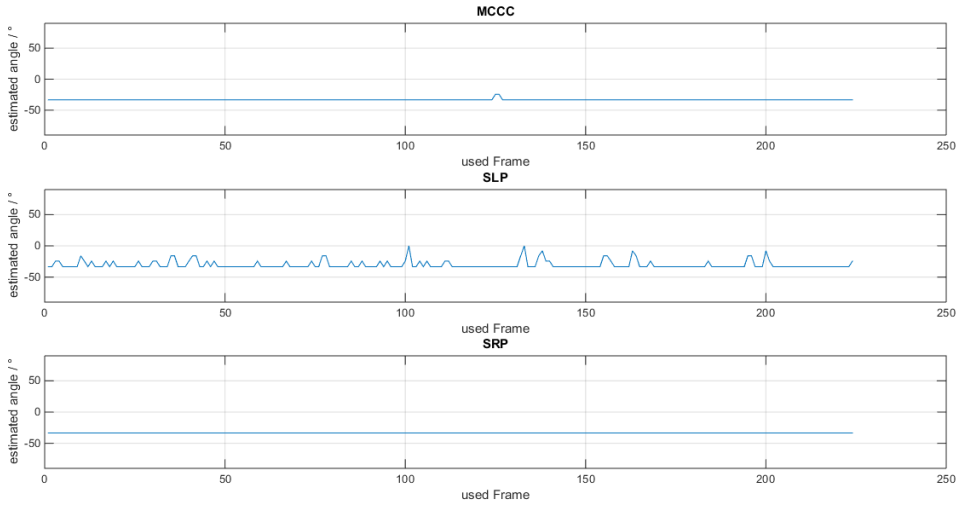


Abbildung 3.5.: Zeitbasierte Verfahren bei Quellsignal zwischen 100 und 2000 Hz

Tabelle 3.2.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal zwischen 100 und 2000 Hz

Verfahren	**Standardabweichung** $\sigma/°$
MCCC	0,8490
SLP	6,3341
SRP	0

Es wird deutlich, dass durch ein breitbandigeres Eingangssignal eine Verbesserung eintritt. Das liegt daran, dass die Korrelationsfunktionen hier eine kürzere Periode haben und deren

Maximum dadurch definierter ist.
Die nachfolgende Simulation verwendet wieder ein Quellsignal zwischen 100 und 1000 Hz. Dieses Mal wird die Blocklänge verdoppelt.

- Abtastfrequenz: 48 kHz
- Blocklänge: 512 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 1000 Hz
- SNR: 20 dB

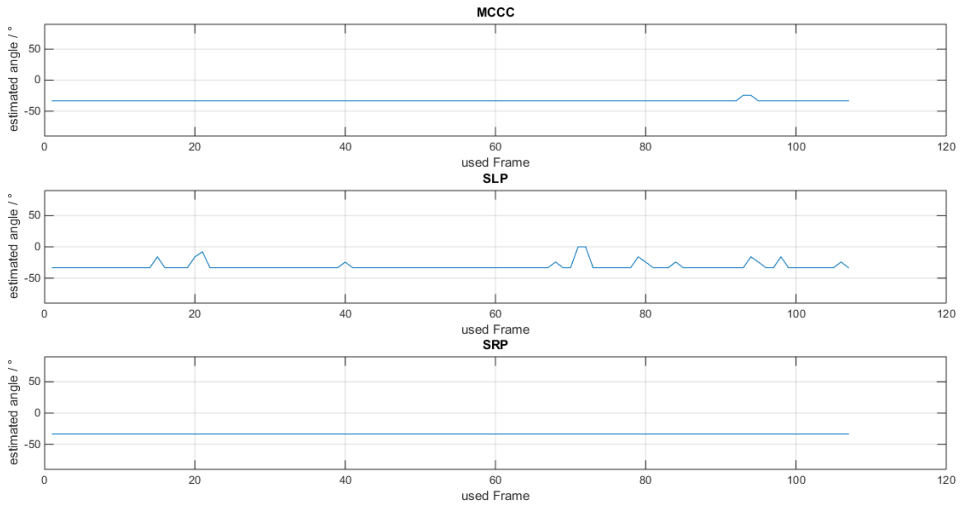


Abbildung 3.6.: Zeitbasierte Verfahren bei Quellsignal zwischen 100 und 1000 Hz mit doppelter Blocklänge

Tabelle 3.3.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal zwischen 100 und 1000 Hz mit doppelter Blocklänge

Verfahren	**Standardabweichung $\sigma/°$**
MCCC	1,2253
SLP	6,4192
SRP	0

Auch hier ist eine Verbesserung aller Verfahren zu sehen. Die Blocklänge hat einen Einfluss auf den Skalierungsfehler, welcher unter Abschnitt 7.3 erläutert wird. Außerdem handelt es sich bei den Zeitsignalausschnitten immer nur um Stichproben, welche das Signal repräsentieren sollen. Durch einen größeren Block steigt folglich die Anzahl dieser Stichproben pro Block, sodass das zugrunde liegende Signal besser repräsentiert wird.

3.1.3. Simulation zeitbasierter Verfahren mit aufgenommenen Sprachsignalen

Im Rahmen einer vorhergegangenen Masterarbeit wurden Sprachsignale mit einem ULA in einem reflexionsarmen Schallmessraum aufgezeichnet. Diese sollen hier ebenfalls verwendet werden, um eine realitätsnahe Simulation durchzuführen. Die Signale wurden mit einer Abtastrate von 8 kHz aufgezeichnet und werden für die Simulation interpoliert. Ansonsten bleiben sie unverfälscht. Das bedeutet, es wird kein zusätzliches Rauschen addiert. Es findet eine Energieberechnung statt, um Sprechpausen zu erkennen. In diesen wird der Algorithmus nicht ausgeführt und der aktuelle Block verworfen.
Die durchgeführte Simulation verwendet folgende Parameter:

- Abtastfrequenz: 48 kHz
- Blocklänge: 512 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals: -50°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Sprachsignal aus schalldichtem Messraum Frauenstimme 130 bis 200 Hz

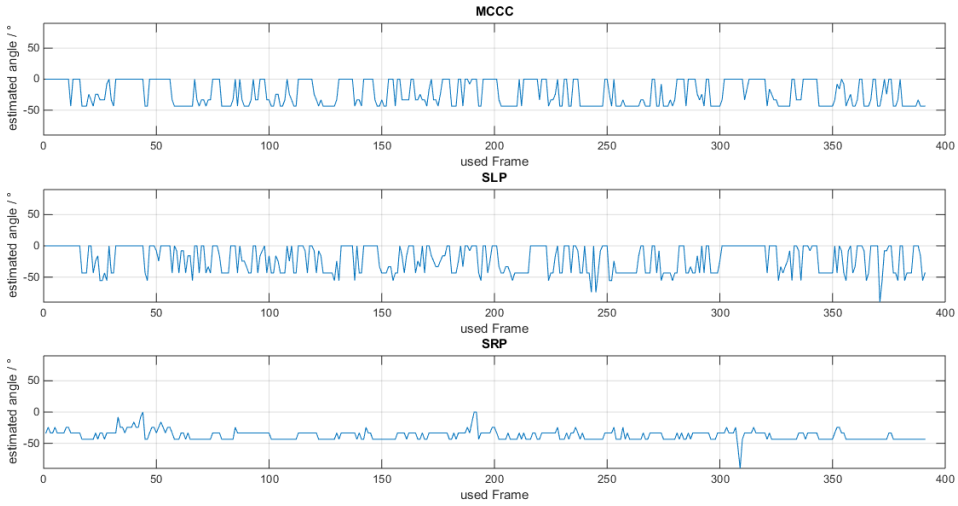


Abbildung 3.7.: Zeitbasierte Verfahren bei Frauenstimme 130 bis 200 Hz

Tabelle 3.4.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal Frauenstimme 130 bis 200 Hz

Verfahren	**Standardabweichung $\sigma/°$**
MCCC	19,8125
SLP	21,7851
SRP	8,6040

Die Simulationsergebnisse aus Abbildung 3.7 zeigen einen sehr hohen Fehleranteil. Es werden mehr Fehldetektionen als korrekte ermittelt. Auch hier ist auffällig, dass bei SLP und MCCC nahezu alle Fehldetektionen bei einem Einfallswinkel von $0°$ liegen. Die wenigsten wären durch normale Streuung zu erklären. Es scheint ein systematischer Fehler vorzuliegen, dessen Ursache es zu untersuchen gilt.
Zunächst sollen allerdings noch weitere Simulationen durchgeführt werden. Nun wird abermals die Blocklänge erhöht und dieses Mal ein Block mit einer Länge von 2048 Samples benutzt.

- Abtastfrequenz: 48 kHz
- Blocklänge: 2048 Samples
- Anzahl der Mikrofone: 8

- Einfallswinkel des Signals: -50°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Sprachsignal aus schalldichtem Messraum Frauenstimme 130 bis 200 Hz

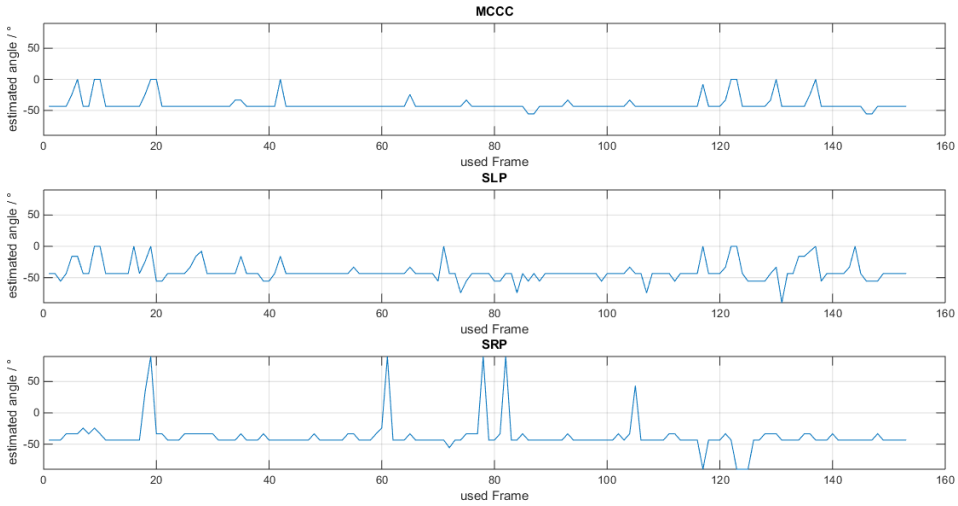


Abbildung 3.8.: Zeitbasierte Verfahren bei Frauenstimme 130 bis 200 Hz mit vierfacher Blocklänge

Tabelle 3.5.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal Frauenstimme 130 bis 200 Hz mit vierfacher Blocklänge

Verfahren	**Standardabweichung $\sigma/°$**
MCCC	11,6903
SLP	15,2822
SRP	27,9811

Bei dem Vergleich zur vorherigen Simulation ist bei dem MCCC und der SLP eine deutliche Verbesserung zu sehen. Das liegt an dem bereits zuvor beschriebenen Einsatz eines längeren Blocks. Bei der SRP lassen sich neben des allgemein deutlich glatteren Verlaufs einige extreme Abweichungen beobachten, wodurch der Wert der Standardabweichung deutlich größer wird.
Nun wird zu den größeren Blöcken ein Medianfilter angewendet, um die beobachteten starken kurzzeitigen Abweichungen zu unterdrücken.

- Abtastfrequenz: 48 kHz
- Blocklänge: 2048 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals : -50°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Sprachsignal aus schalldichtem Messraum Frauenstimme 130 bis 200 Hz
- Medianfilter der Länge 6

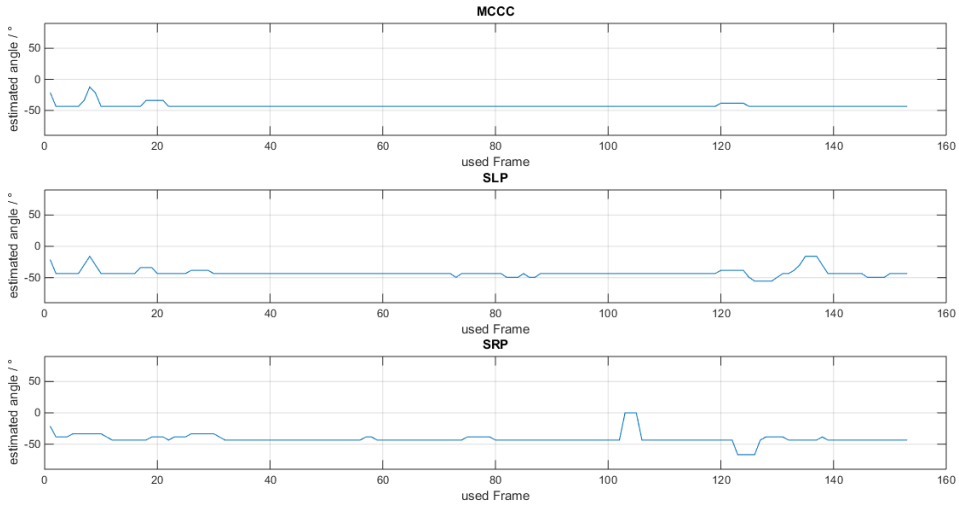


Abbildung 3.9.: Zeitbasierte Verfahren bei Frauenstimme 130 bis 200 Hz mit vierfacher Blocklänge und Medianfilter

Tabelle 3.6.: Standardabweichungen bei zeitbasierten Verfahren mit Quellsignal Frauenstimme 130 bis 200 Hz mit vierfacher Blocklänge und Medianfilter

Verfahren	**Standardabweichung $\sigma/°$**
MCCC	3,9394
SLP	6,1075
SRP	9,3312

Abbildung 3.9 zeigt, dass unter Verwendung einer ausreichend großen Blocklänge und eines Medianfilters, um kurzzeitige Abweichungen zu unterdrücken, alle Algorithmen akzeptable

Ergebnisse bei der Anwendung mit aufgezeichneten Signalen eines ULA liefern. Der MC-CC erweist sich hier als die Methode mit den besten Ergebnissen.
Allerdings ist es ebenfalls notwendig zu erwähnen, dass hierfür eine sehr große Blocklänge notwendig war, was die Komplexität deutlich erhöht.

3.1.4. Einfluss der verschiedenen Parameter auf zeitbasierte Verfahren

Nachdem die grundlegende Funktionalität der Algorithmen unter Beweis gestellt wurde, soll im Folgenden der Einfluss der Variation von unterschiedlichen Parametern auf die diese untersucht werden.

Einfluss des Signal-zu-Rausch-Verhältnises

Das Signal-zu-Rausch-Verhältnis beschreibt das Verhältnis zwischen Signal- und Rauschleistung und somit die Qualität eines Signals. Eine Verschlechterung der Qualität durch Rauschen kann durch verschiedene Einflüsse geschehen. Einige Beispiele dafür sind nach [29, S. 141]:

- Thermisches Rauschen: Thermisches Rauschen entsteht zufällig durch Schwankungen der Ladungsträgerdichte in verlustbehafteten Strukturen.
- Quantisierungsrauschen: Quantisierungsrauschen ist nicht zufällig und entsteht durch das Auf- oder Abrunden einer Größe, welches die Quantisierung bedingt.

Wichtig für die hier vorgestellte Untersuchung sind, neben den Ursachen von Rauschen, die Stelle bzw. die Stellen im System, an denen Rauschen entsteht.
Die vorgestellten Algorithmen basieren darauf, die Zeitverschiebung eines Quellsignals, welches an mehreren Mikrofonen gemessen wird, zu schätzen. Dafür ist die Art des Quellsignals zunächst nicht wichtig, solange es die Einschränkungen bezüglich der Bandbreite erfüllt. Es ist demnach möglich neben Sprachsignalen ein rein zufälliges Signal (Schmalbandrauschen) zu orten. Auch eine Überlagerung von Sprachsignalen und Rauschen zum Beispiel verrauschte Aufnahmen, die mit einem Lautsprecher abgespielt werden, sind denkbar und können als Quelle gewertet werden.

Einen wirklichen Einfluss auf die Verfahren hat das Rauschen, was an den jeweiligen Mikrofonen vorhanden ist. Dieses ist vollständig unkorreliert zu dem Rauschen der jeweils anderen Mikrofone und enthält somit nicht die gewünschte zeitliche Information, welche in dem Nutzanteil der Signale vorhanden ist.
Im Folgenden wird genau dieses individuelle Rauschen simuliert. Die Parameter der Simulation lauten wie folgt:

- Abtastfrequenz: 48 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz

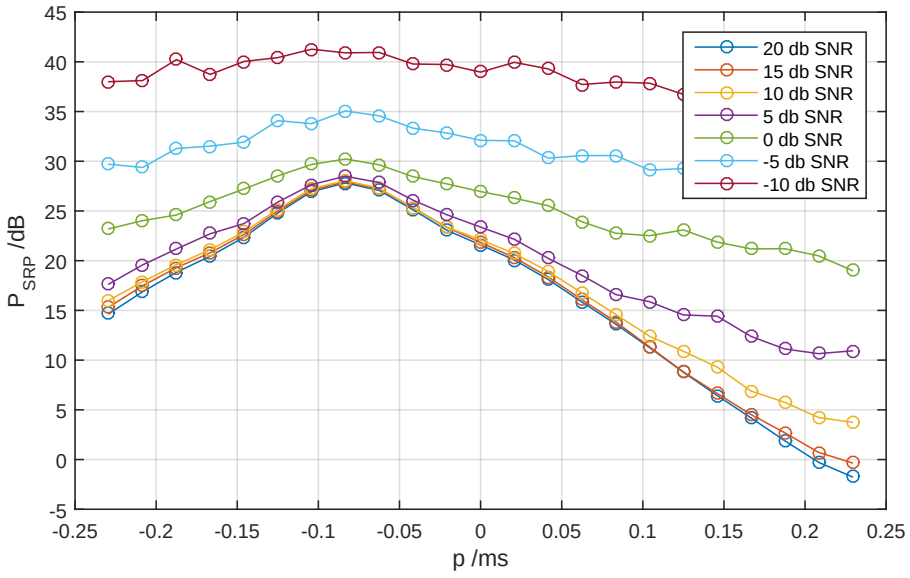


Abbildung 3.10.: Leistungsfunktion der SRP für verschiedene SNR

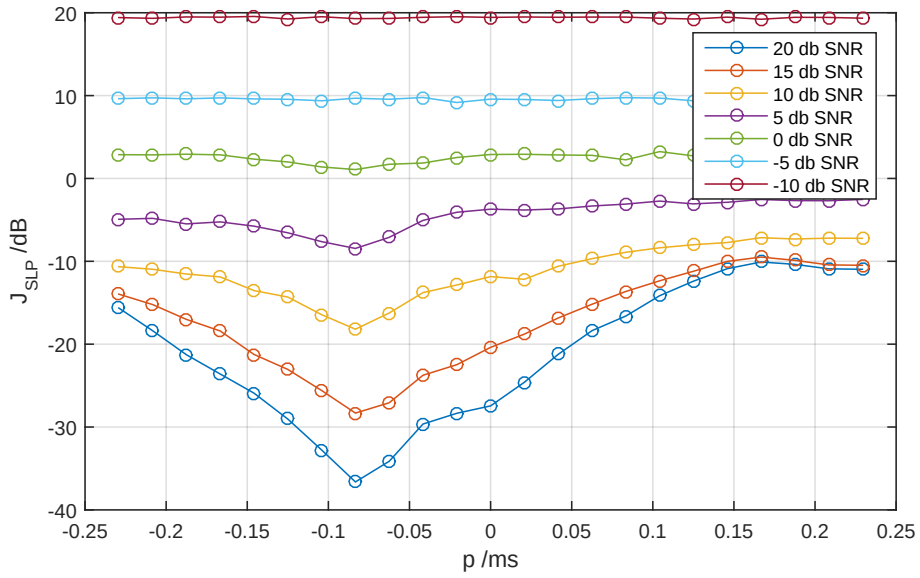


Abbildung 3.11.: Fehlerfunktion der SLP für verschiedene SNR

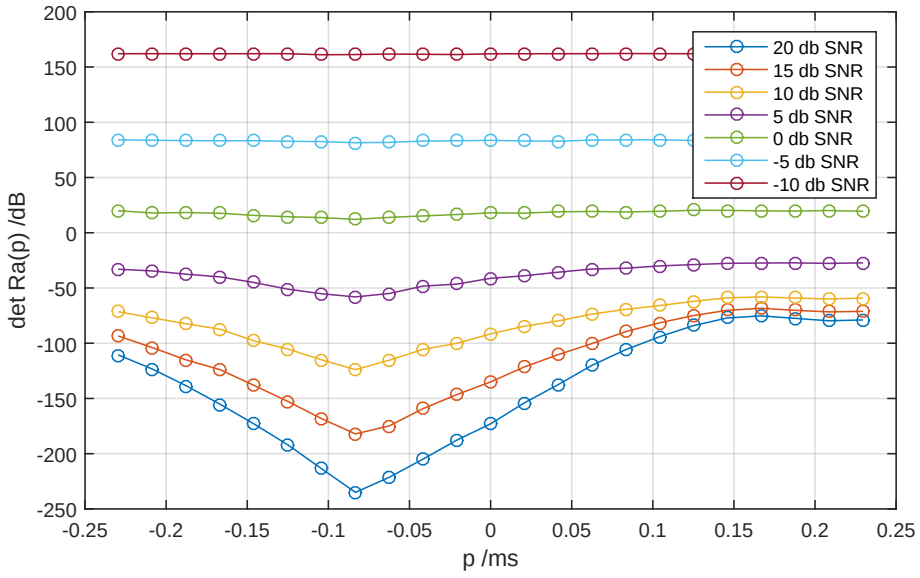


Abbildung 3.12.: Fehlerfunktion des MCCC für verschiedene SNR

Bei allen drei Verfahren ist zu erkennen, dass eine Verringerung des Signal-zu-Rausch-Abstands eine Verschlechterung herbeiführt. Dies ist in der Natur der zeitbasierten Verfahren begründet. In diesen geht es darum eine Zeitverzögerung zu schätzen, welche Parameter der räumlichen Korrelationsmatrix ist. Da das Rauschen unkorreliert ist, hat es keinen Nutzen für diese Verfahren, verändert jedoch die Werte in der räumlichen Korrelationsmatrix, sodass die korrelierten Signalanteile mit größer werdender Rauschamplitude immer mehr an Bedeutung verlieren.
Bei dem MCCC und der SLP ist dieser Einfluss gut erkennbar. Jede Änderung des SNR bewirkt ein schwächer abgezeichnetes Minimum. Bei der SRP ist dies zwischen 0 und 20dB nicht so. Die Leistungsfunktion verläuft annähernd identisch, was die Abzeichnung der Minima angeht. Dies liegt daran, dass bei diesem Verfahren die Leistung der Korrelationsmatrix berechnet wird. Diese steigt deutlich an je größer die Rauschamplitude ist, da dadurch die Varianzen auf der Hauptdiagonalen einen größeren Wert annehmen. Alle anderen Elemente in der Matrix sind maßgeblich durch die korrelierten Signalanteile bestimmt, sodass diese den Verlauf der Funktion bestimmen, wogegen die Elemente auf der Hauptdiagonalen einen Offset bedeuten.

Einfluss der Anzahl der Mikrofone

In diesem Abschnitt wird der Einfluss der Anzahl der verwendeten Mikrofone auf die Verfahren untersucht. Für die Simulation wurden die folgenden Parameter verwendet:

- Abtastfrequenz: 48 kHz
- Blocklänge: 256 Samples
- SNR: 20 dB
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz

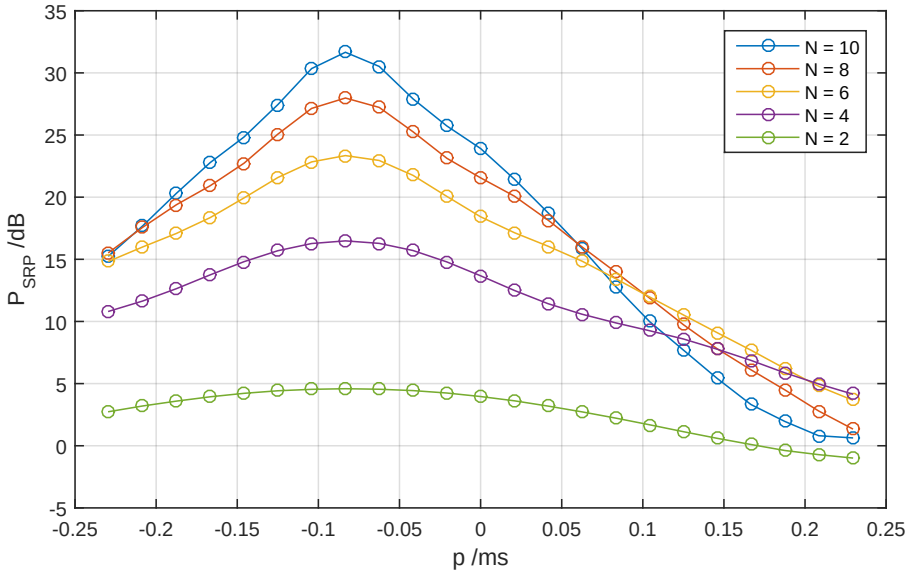


Abbildung 3.13.: Leistungsfunktion der SRP für verschiedene Anzahl von Mikrofonen

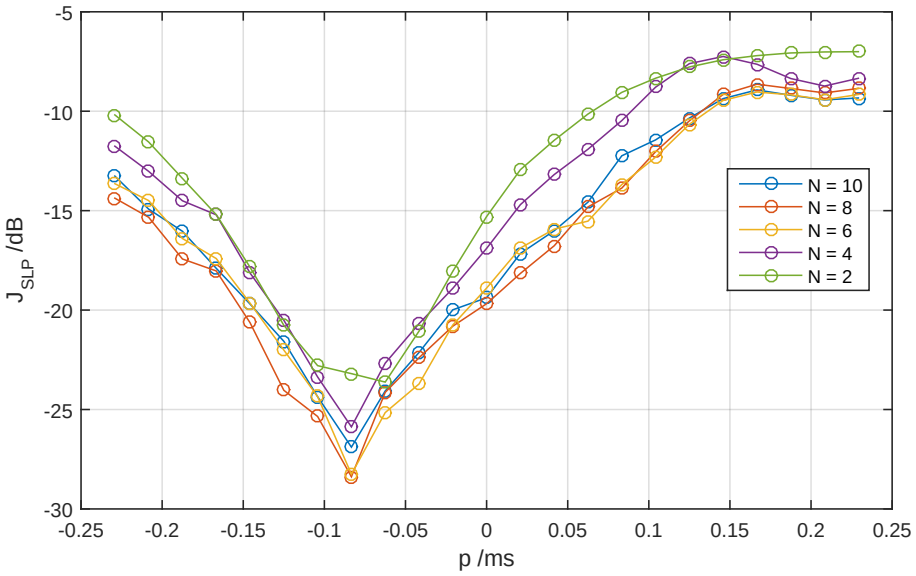


Abbildung 3.14.: Fehlerfunktion der SLP für verschiedene Anzahl von Mikrofonen

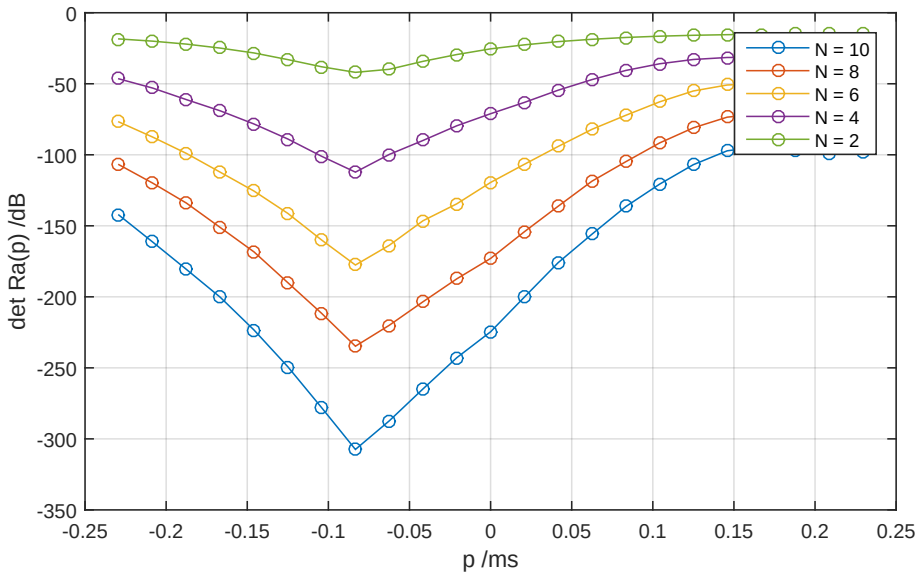


Abbildung 3.15.: Fehlerfunktion des MCCC für verschiedene Anzahl von Mikrofonen

Auch in diesem Fall ist zu erkennen, dass getreu der Erwartung eine Verbesserung in Form von einer deutlicheren Absetzung des Minimums der Fehler- bzw. des Maximums der Leistungsfunktion mit der Anzahl der verwendeten Mikrofone bei der MCCC und der SRP eintritt. Dies ist dadurch zu begründen, dass die Anzahl der Korrelationen (Matrixelemente) quadratisch ansteigt. Bei der SLP verhält sich das anders. Dort wird die Zeitverzögerung zwischen dem ersten und den anderen Mikrofonen herangezogen, jedoch nicht die der anderen Mikrofone untereinander. Somit ist der Einfluss, den die Anzahl der Mikrofone hat, deutlich geringer.

3.2. Simulation unterraumbasierter Algorithmen

3.2.1. Verifikation Unterraumbasierter Verfahren

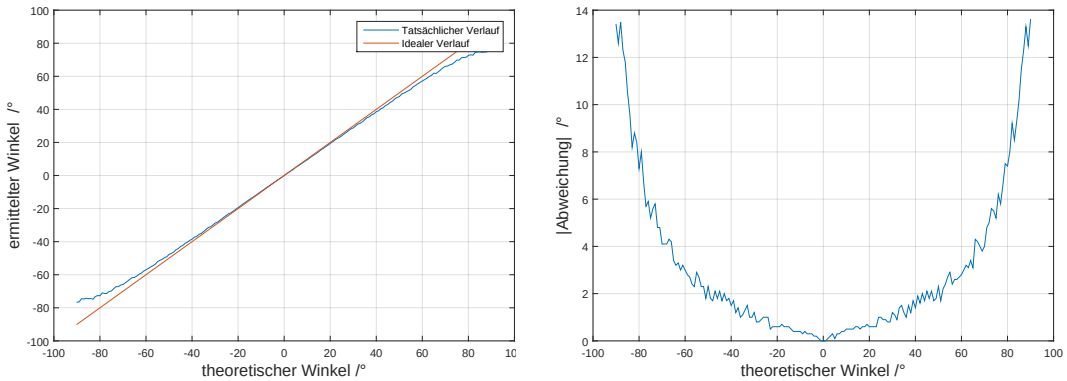
Auch die unterraumbasierten Verfahren sollen zunächst über den gesamten Winkelbereich verifiziert werden. Es werden dieselben synthetischen Signale wie zur Verifizierung der zeit-

basierten Verfahren genutzt. Der einzige Unterschied ist, dass mit einer Abtastfrequenz von 8 kHz gearbeitet wird.

Verifikation der Algorithmen im Schmalband-Fall

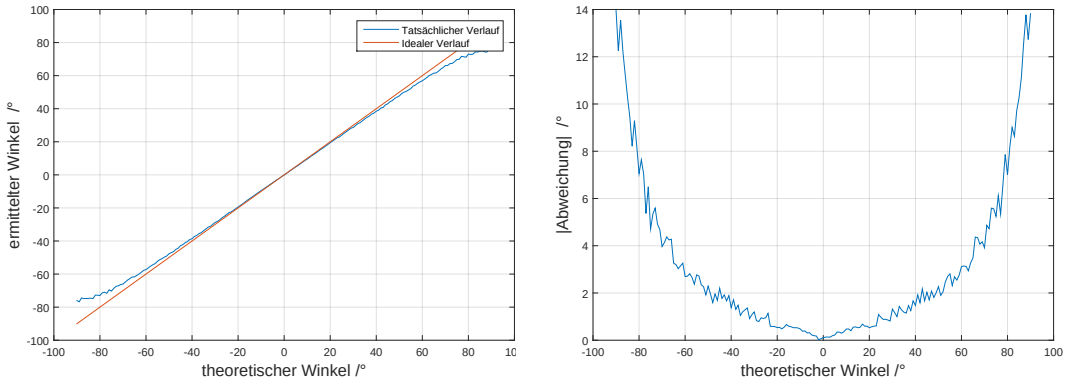
Zunächst sollen die Algorithmen für den Schmalband-Fall, das bedeutet für eine feste Testfrequenz, verifiziert werden. Diese wird fest vorgegeben. Der Ansatz ist nicht optimal, da die Signale vorher unbekannt sind und somit nicht sichergestellt sein kann, dass das Signal rund um die Testfrequenz überhaupt signifikante Anteile aufweist. Für die Verifizierung wurde eine Frequenz gewählt, für die dies jedoch zutrifft.
Die Simulationsparameter lauten wie folgt:

- Abtastfrequenz: 8 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone : 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz
- SNR: 20 dB
- Testfrequenz: 625 Hz



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.16.: Verifizierung Schmalband-MUSIC ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.17.: Verifizierung Schmalband-ROOT-MUSIC ermittelter und simulierter Winkel und absolute Abweichung

ESPRIT basiert bekannterweise darauf, dass das Mikrofonarray in Subarrays unterteilt wird. Dafür gibt es verschiedene Ansätze. Ein Ansatz ist es die Subarrays überlappen zulassen, sodass einige Mikrofone beiden Arrays zugeordnet werden. Diese Methode wird hier und in den weiteren Simulationen als ESPRIT Methode 1 bezeichnet.

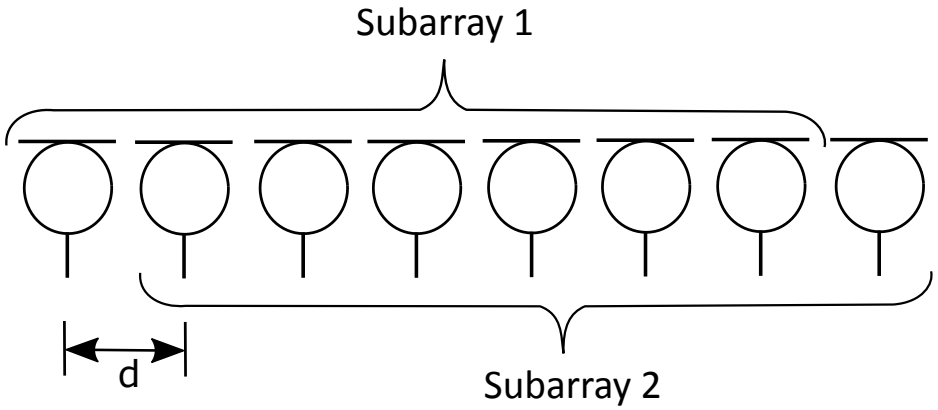


Abbildung 3.18.: Anordnung der Mikrofone für Methode 1 des ESPRIT-Algorithmus

Weiterhin besteht die Möglichkeit, das Array nicht überlappend in zwei gleich große Subarrays zu unterteilen. Diese Methode wird hier und im Folgenden ESPRIT Methode 2 genannt.

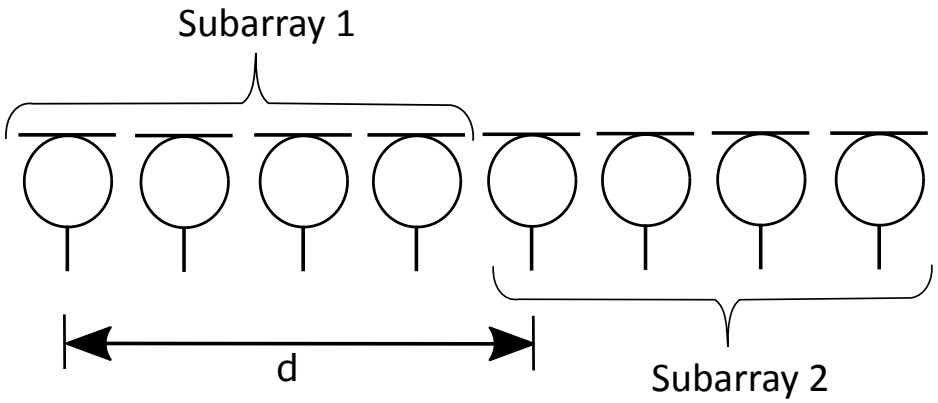
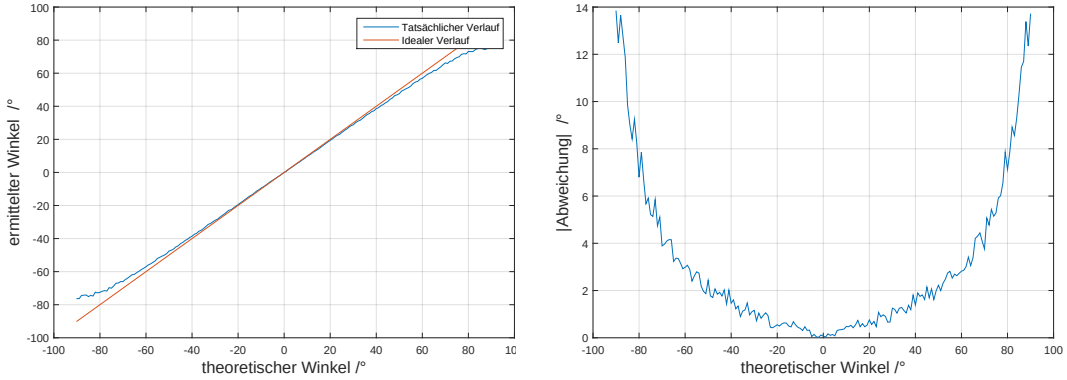


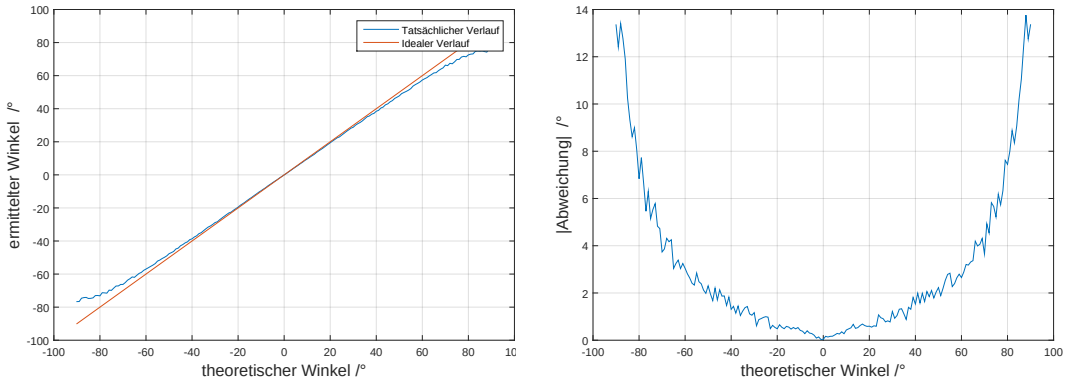
Abbildung 3.19.: Anordnung der Mikrofone für Methode 2 des ESPRIT Algorithmus



(a) Ermittelter und simulierter Winkel

(b) Absolute Abweichung

Abbildung 3.20.: Verifizierung Schmalband-ESPRIT (Methode 1) ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.21.: Verifizierung Schmalband-ESPRIT (Methode 2) ermittelter und simulierter Winkel und absolute Abweichung

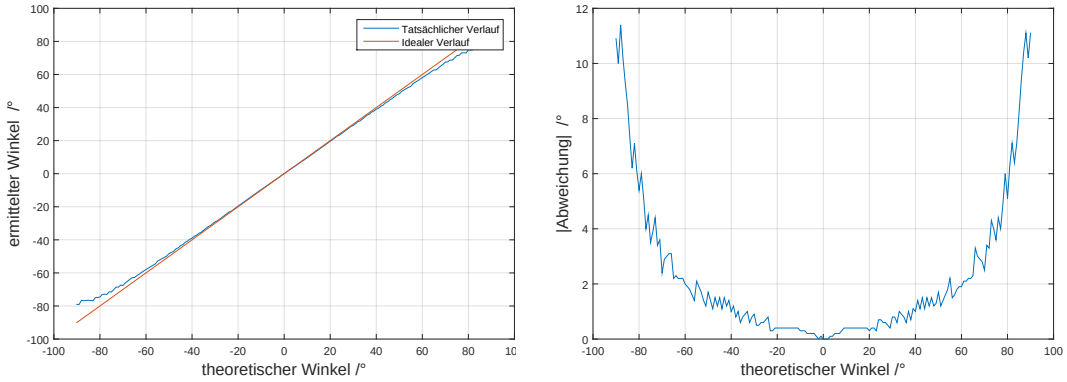
Die Abbildungen 3.16, 3.17, 3.20 und 3.21 und zeigen, dass alle Algorithmen prinzipiell funktionieren. Sie alle liefern im Bereich von -75° bis +75° sehr gute Ergebnisse. Außerhalb dieses Bereichs sind allerdings starke Abweichungen zu erkennen. Dies gilt für alle Verfahren.

Verifikation der Algorithmen im Breitband-Fall

Im vorherigen Abschnitt wurden die Algorithmen jeweils für eine Testfrequenz verifiziert. Implementiert werden soll jedoch ein Breitbandansatz, sodass die Verfahren jeweils für den Breitband-Fall verifiziert werden sollen. Dazu wurde ein inkohärenter Ansatz gewählt, indem jeweils der Schmalband-Ansatz für vier verschiedene Testfrequenzen angewendet wird. Das Ergebnis setzt sich anschließend aus einer Kombination der Teilergebnisse zusammen. Die Testfrequenzen wurden hier ebenfalls fest vorgegeben. Dazu wurde eine Frequenz mit signifikanten Leistungsanteilen ausgesucht und Vielfache von dieser als weitere Testfrequenzen gewählt.

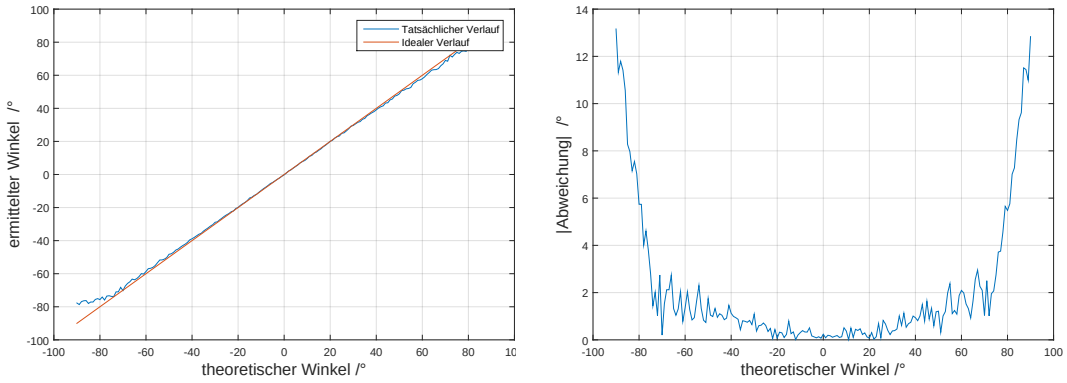
- Abtastfrequenz: 8 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm

- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz
- SNR: 20 dB
- Testfrequenzen: 156,25 Hz; 312.5 Hz; 625 Hz; 1250 Hz



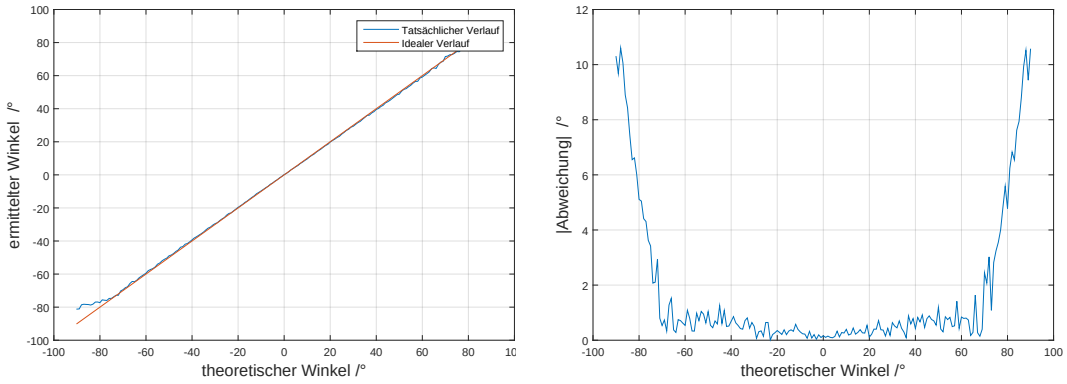
(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.22.: Verifizierung Breitband-MUSIC ermittelter und simulierter Winkel und absolute Abweichung



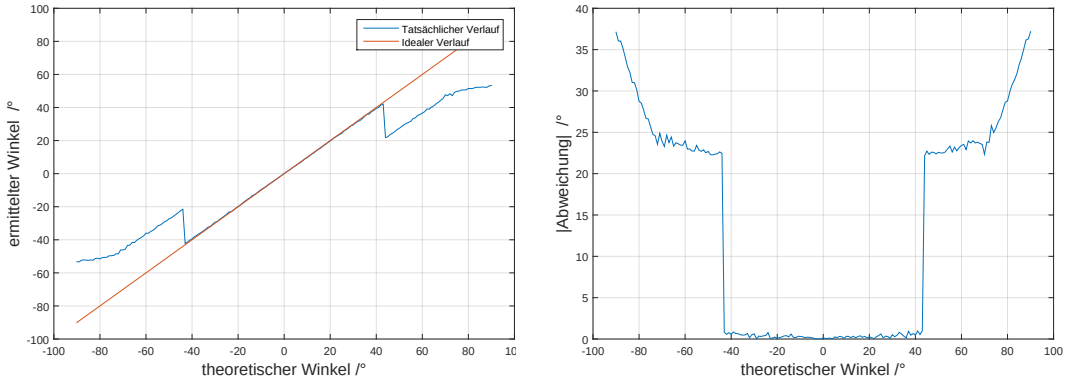
(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.23.: Verifizierung Breitband-ROOT-MUSIC ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.24.: Verifizierung Breitband-ESPRIT (Methode 1) ermittelter und simulierter Winkel und absolute Abweichung



(a) Ermittelter und simulierter Winkel (b) Absolute Abweichung

Abbildung 3.25.: Verifizierung Breitband-ESPRIT (Methode 2) ermittelter und simulierter Winkel und absolute Abweichung

Die Ergebnisse der Verifizierung zeigen, dass alle Algorithmen, bis auf den ESPRIT-Algorithmus, bei dem das Array in zwei gleich große Subarrays unterteilt wird, auch im Breitband-Fall funktionieren. Bei dem ESPRIT-Algorithmus in der Methode 2 sind bei ca. -45° und bei ca. 45° Sprünge zu beobachten. Jenseits dieser Winkel liegt ein konstanter Fehler im Vergleich zum Verlauf bei der Methode 1 vor. Dies liegt an dem räumlichen Aliasing, welches hier auftritt. Denn der Unterschied zwischen den beiden Methoden ist, dass der Abstand zwischen den Subarrays bei der Methode 2 vierfach größer ist als bei der Methode 1. Dadurch tritt bei der Methode 2 bei der höchsten Testfrequenz der Alias-Effekt ein. Da der

inkohärente Ansatz eine Mittlung der ermittelten Winkel bedeutet, wirkt sich dieser Fehler sofort sehr stark auf das Ergebnis aus.
Das ist beim inkohärenten Breitbandansatz für MUSIC nicht der Fall. Dieser lautet wie folgt:

$$S_{MUSIC}(\phi) = \frac{1}{\sum_{i=1}^{L} \boldsymbol{\varsigma}^H(\tau(\phi_i)) \cdot \boldsymbol{Q}_n \cdot \boldsymbol{Q}_n^H \cdot \boldsymbol{\varsigma}(\tau(\phi_i))} \tag{3.1}$$

3.2.2. Simulation unterraumbasierter Verfahren mit synthetischen Signalen

Zunächst wurde für die Simulation ein synthetisches Signal mit einem Frequenzbereich von 100 Hz bis 1000 Hz verwendet. Simuliert wird der Breitband-Fall mit vier Testfrequenzen. Dazu wird für jeden Rahmen eine Average Magnitude Difference Function (AMDF) gebildet, um die vier Frequenzen mit den größten Leistungsanteilen zu bestimmen. Anschließend werden die Steering-Vektoren für diese berechnet.
Die folgende Abbildung zeig die Testergebnisse für folgende Parameter:

- Abtastfrequenz: 8 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 1000 Hz
- SNR: 20 dB

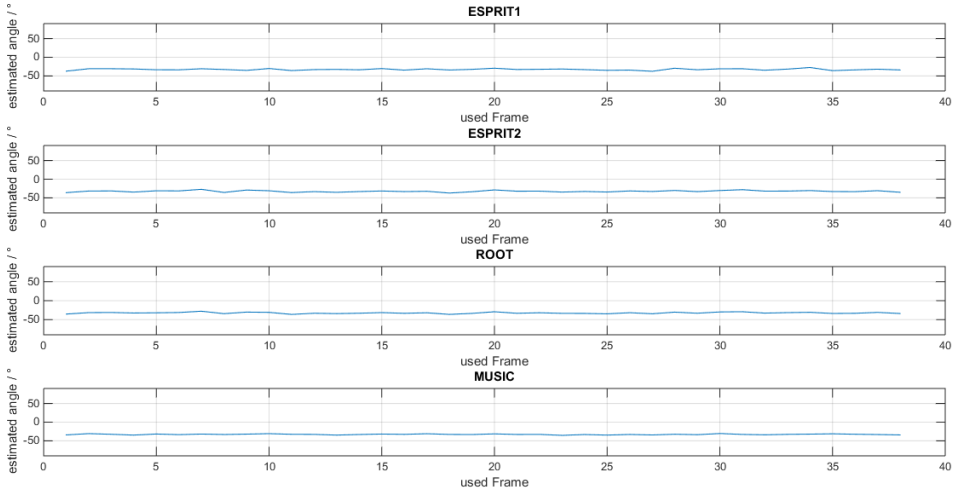


Abbildung 3.26.: Unterraumbasierte Verfahren bei Quellsignal zwischen 100 und 1000 Hz

Tabelle 3.7.: Standardabweichungen bei unterraumbasierten Verfahren mit Quellsignal zwischen 100 und 1000 Hz

Verfahren	**Standardabweichung $\sigma/°$**
MUSIC	1,0445
ROOT-MUSIC	2,2781
ESPRIT 1	3,0445
ESPRIT 2	2,2155

Alle Verfahren funktionieren ähnlich gut, wobei MUSIC die genausten Ergebnisse liefert. Die nächste Simulation verwendet das gleiche Testsignal und auch die anderen Parameter, mit Ausnahme der Blocklänge, bleiben gleich. Die Blocklänge wird von 256 auf 1024 Samples erhöht.

- Abtastfrequenz: 8 kHz
- Blocklänge: 1024 Samples
- Anzahl der Mikrofone : 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 1000 Hz

- SNR: 20 dB

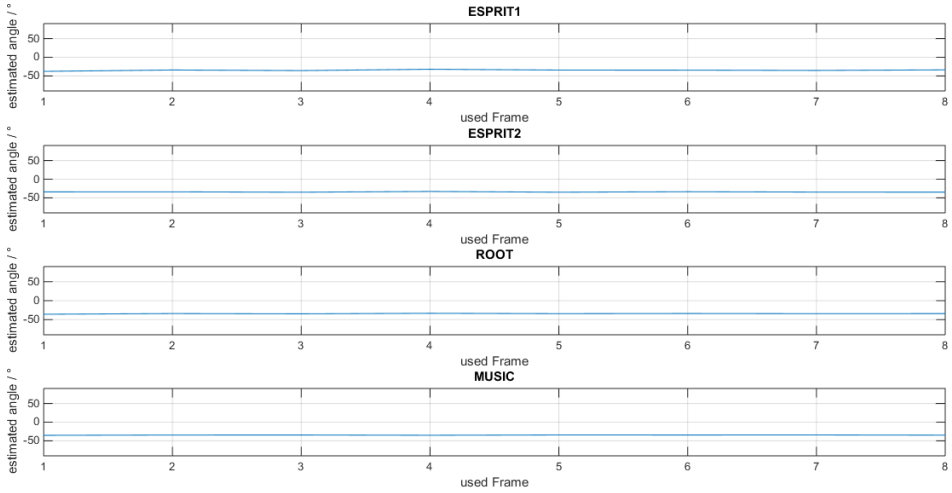


Abbildung 3.27.: Unterraumbasierte Verfahren bei Quellsignal zwischen 100 und 1000 Hz mit vierfacher Blocklänge

Tabelle 3.8.: Standardabweichungen bei unterraumbasierten Verfahren mit Quellsignal zwischen 100 und 1000 Hz mit vierfacher Blocklänge

Verfahren	**Standardabweichung** $\sigma/°$
MUSIC	0,3980
ROOT-MUSIC	0,7607
ESPRIT 1	1,3782
ESPRIT 2	0,8771

Es wird sehr deutlich, dass bei allen Verfahren alleine durch die Vergrößerung der Blocklänge deutliche Verbesserungen in den Ergebnissen eintreten. Das hat vor allem den Grund, dass so eine deutlich feinere Frequenzauflösung zustande kommt, wodurch die Bereiche rund um die Testfrequenzen, welche für die Algorithmen genutzt werden, deutlich schmalbandiger sind.

3.2.3. Simulation unterraumbasierter Verfahren mit aufgenommenen Sprachsignalen

Auch die unterraumbasierten Verfahren sollten mit den schon zuvor verwendeten Originalaufnahmen getestet werden. Im Gegensatz zu den Tests der zeitbasierten Verfahren wird hier auf die Interpolation der Signale verzichtet und die Abtastfrequenz von 8 kHz wird beibehalten.

- Abtastfrequenz: 8 kHz
- Blocklänge: 1024 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals: -50°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Sprachsignal aus schalldichtem Messraum Frauenstimme 130 bis 200 Hz

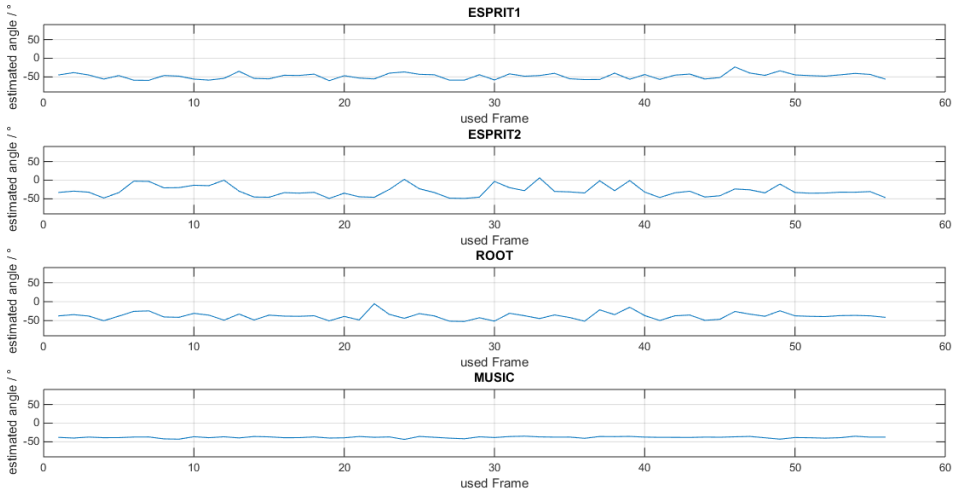


Abbildung 3.28.: Unterraumbasierte Verfahren bei Frauenstimme 130 bis 200 Hz

Tabelle 3.9.: Standardabweichungen bei unterraumbasierten Verfahren bei Frauenstimme 130 bis 200 Hz

Verfahren	**Standardabweichung $\sigma/°$**
MUSIC	2,0120
ROOT-MUSIC	16,7570
ESPRIT 1	15,5148
ESPRIT 2	19,1165

Es zeigt sich, dass mit Ausnahme von MUSIC deutlich schlechtere Ergebnisse erzielt wurden, als es bei den synthetischen Signalen der Fall war. Das liegt an der Art der Implementierung des inkohärenten Ansatzes. Bei ESPRIT und ROOT-MUSIC wurde hier eine Mittelwertbildung der Winkel für jede Testfrequenz vorgenommen, was zu Folge hat, dass Fehler bei einzelnen Frequenzen sehr starke Einflüsse haben. Bei MUSIC wurde ein kombiniertes MUSIC-Spektrum nach Gleichung 3.1 berechnet. Hier haben Abweichungen einen weniger starken Einfluss.

- Abtastfrequenz: 8 kHz
- Blocklänge: 1024 Samples
- Anzahl der Mikrofone: 8
- Einfallswinkel des Signals : -50°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Sprachsignal aus schalldichtem Messraum Frauenstimme 130 bis 200 Hz
- Medianfilter der Länge 6

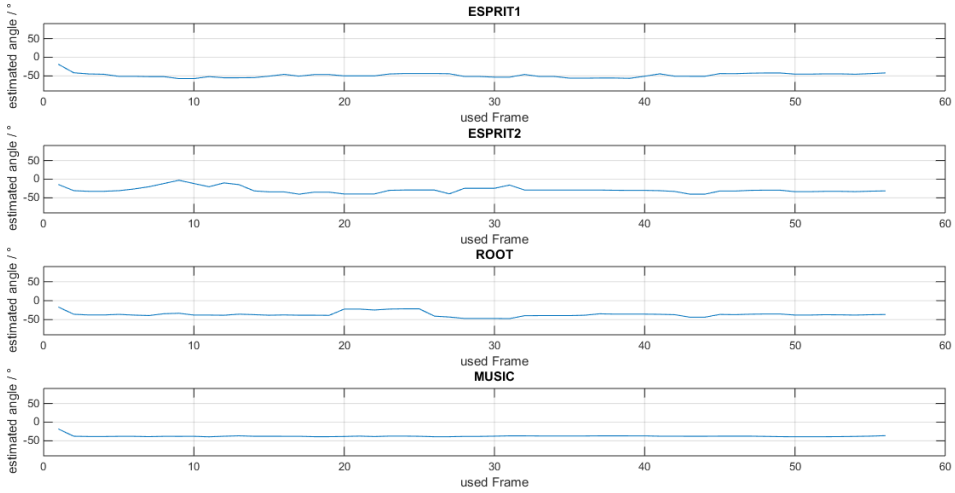


Abbildung 3.29.: Unterraumbasierte Verfahren bei Frauenstimme 130 bis 200 Hz mit Medianfilter

Tabelle 3.10.: Standardabweichungen bei unterraumbasierten Verfahren bei Frauenstimme 130 bis 200 Hz mit Medianfilter

Verfahren	**Standardabweichung $\sigma/°$**
MUSIC	2.6780
ROOT-MUSIC	9.6411
ESPRIT 1	7.6928
ESPRIT 2	9.9939

Durch die Verwendung eines Medianfilters kann das Ergebnis für die Verfahren mit Mittelwertbildung noch verbessert werden. Das Ergebnis für MUSIC wurde jedoch etwas schlechter. Das liegt daran, dass der Medianfilter mit Nullen initialisiert wurde. Seine gewünschte Funktion tritt erst ein, nachdem er vollständig mit den zuletzt ermittelten DOAs gefüllt ist. Bei MUSIC trat in der Simulation auch ohne Medianfilter keine große Streuung auf, sodass dieser Einfluss am Anfang für ein schlechteres Ergebnis sorgt. Bei einer tatsächlichen Implementierung kann dieser Einfluss umgangen werden, indem z. B. erst ein Wert ausgegeben wird, sobald der Medianfilter richtig arbeitet.

3.2.4. Einfluss der verschiedenen Parameter auf unterraumbasierte Verfahren

Genau wie zuvor für die zeitbasierten Verfahren soll an dieser Stelle die Auswirkung verschiedener Parameter auf die Performance der Algorithmen untersucht werden. Die Ergebnisse von ROOT-MUSIC und ESPRIT sind im Gegensatz zu den anderen vorgestellten Verfahren einfache Werte, welche nicht durch Minimum- oder Maximum-Suche aus einem Funktionsverlauf bestimmt werden. Dadurch lassen sich die Einflüsse der Parameter hier nicht anhand dieser Funktionen anschaulich darstellen. Stattdessen wurde hier der Root-Mean-Square Error (RMSE) gewählt, welcher aus jeweils 1000 Monte Carlo Simulationen[2] bestimmt wurde. So kann die Auswirkung der Parameter ebenfalls dargestellt werden.

Einfluss des Signal-zu-Rausch-Verhältnises

Für die Simulation der Einflüsse des SNR wurden folgende Parameter gewählt:

- Abtastfrequenz: 8 kHz
- Blocklänge: 256 Samples
- Anzahl der Mikrofone : 8
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz

[2]Monte Carlo Simulationen beruhen auf der mehrmaligen Wiederholung eines Zufallsexperiments mit immer neuen Zufallsdaten. [37]

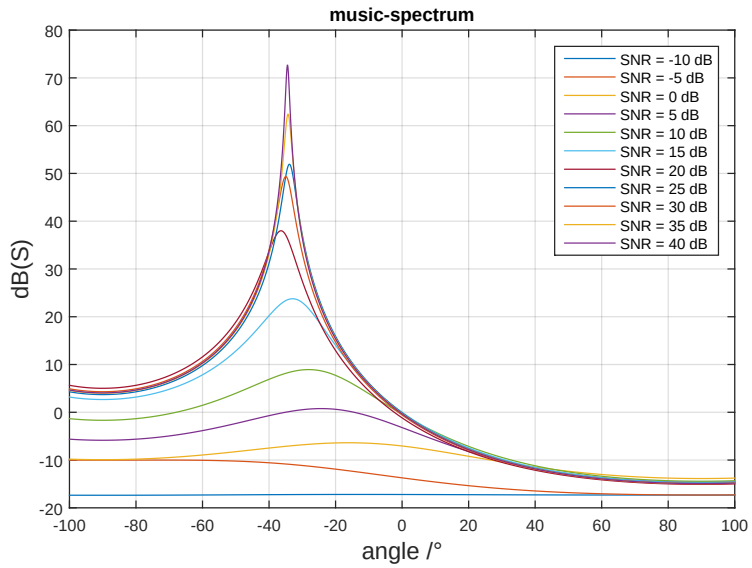


Abbildung 3.30.: MUSIC-Spektrum für verschiedene SNR

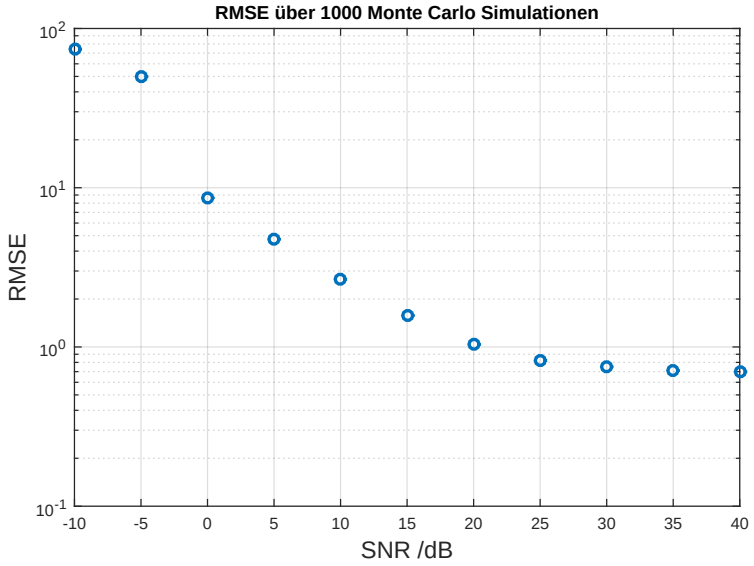


Abbildung 3.31.: RMSE des ROOT-MUSIC Verfahrens für verschiedene SNR

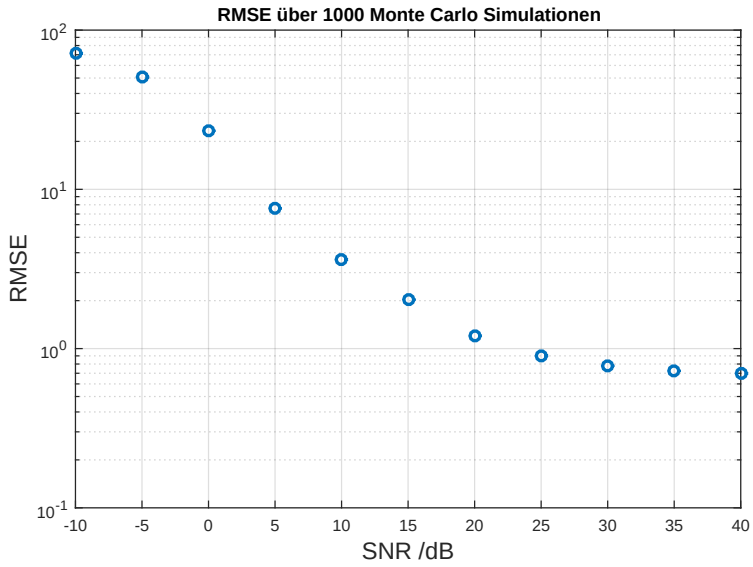


Abbildung 3.32.: RMSE des ESPRIT (Methode 1) Verfahrens für verschiedene SNR

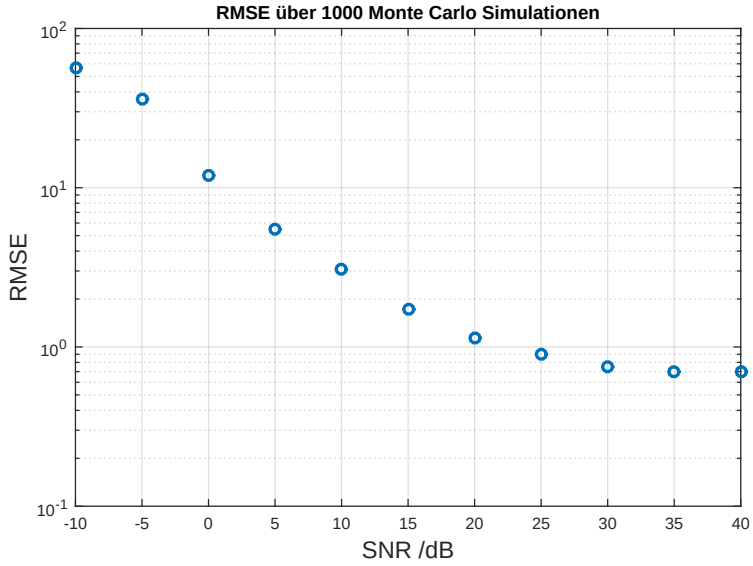


Abbildung 3.33.: RMSE des ESPRIT (Methode 2) Verfahrens für verschiedene SNR

Die Simulationen zeigen, dass alle Verfahren besser funktionieren, sofern das Rauschen geringer ist. Es wird weiterhin deutlich, dass die Verfahren erst brauchbare Ergebnisse liefern, sobald der Signal-zu-Rausch-Abstand größer als 0 dB ist. In diesem Fall ist die Trennung von Signal- und Rauschunterraum leichter möglich.

Einfluss der Anzahl der Mikrofone

In diesem Abschnitt wird der Einfluss der Anzahl der verwendeten Mikrofone auf die Verfahren untersucht. Für die Simulation wurden die folgenden Parameter verwendet:

- Abtastfrequenz: 8 kHz
- Blocklänge: 256 Samples
- SNR: 20 dB
- Einfallswinkel des Signals: -35°
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen 100 bis 2000 Hz

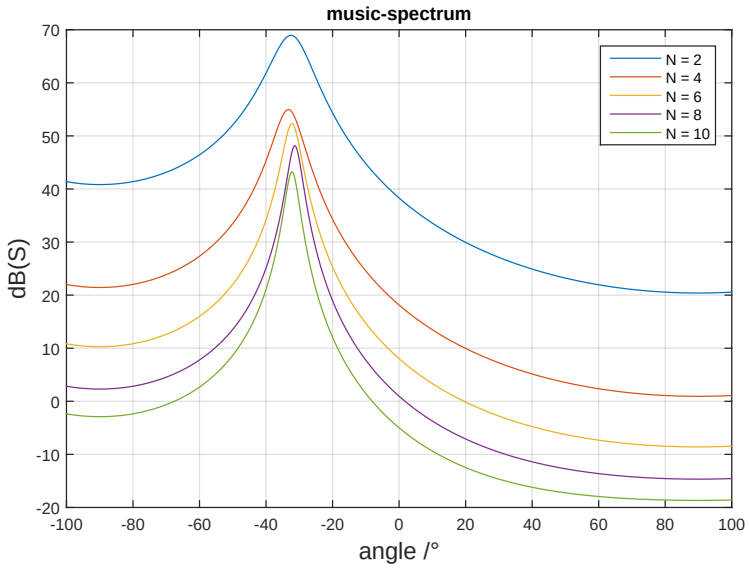


Abbildung 3.34.: MUSIC-Spektrum für verschiedene Anzahl von Mikrofonen

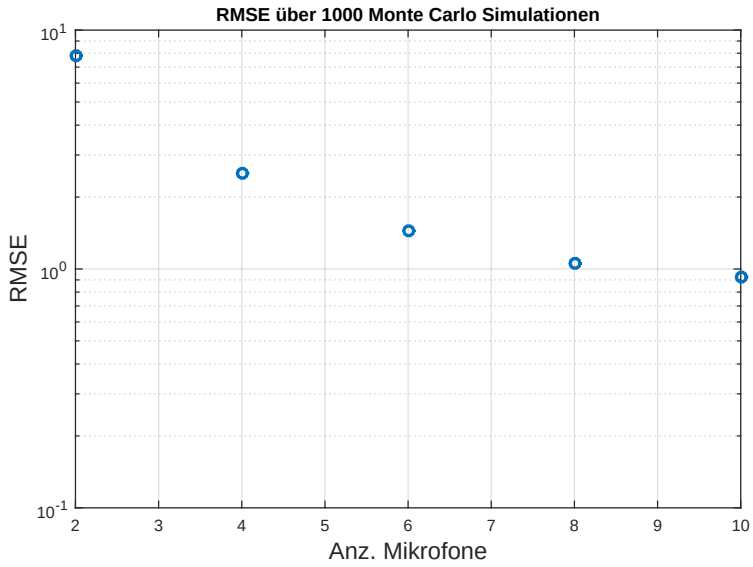


Abbildung 3.35.: RMSE des ROOT-MUSIC Verfahrens für verschiedene Anzahl von Mikrofonen

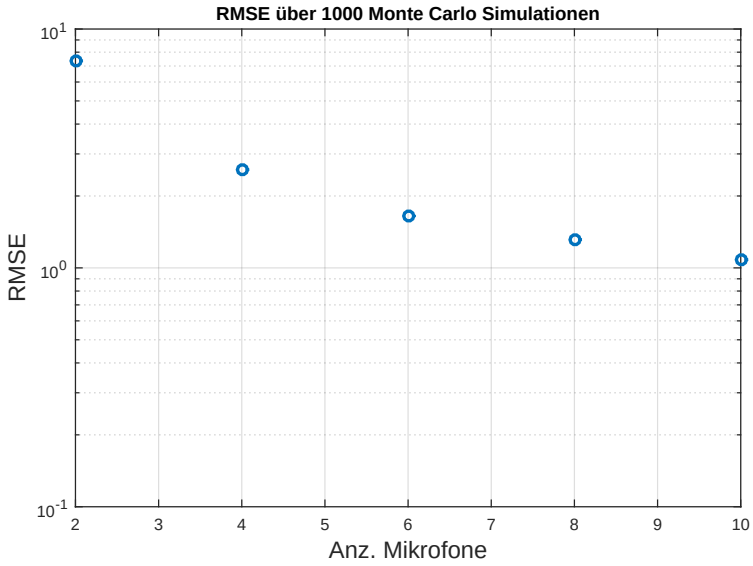


Abbildung 3.36.: RMSE des ESPRIT (Methode 1) Verfahrens für verschiedene Anzahl von Mikrofonen

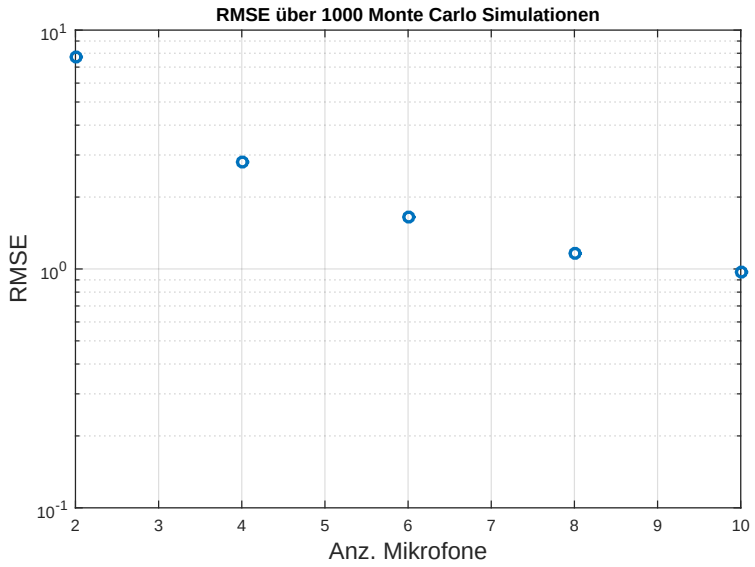


Abbildung 3.37.: RMSE des ESPRIT (Methode 2) Verfahrens für verschiedene Anzahl von Mikrofonen

In Abbildung 3.34 ist zu erkennen, dass das MUSIC-Spektrum deutlich ausgeprägter ist, sobald die Anzahl der Mikrofone größer ist. Auch die anderen Abbildungen, in denen der RMSE dargestellt ist, zeigen, dass alle Verfahren besser mit mehr Mikrofonen funktionieren.

3.3. Wahl der Algorithmen zur weiteren Betrachtung

Da eine Implementierung aller Verfahren den Rahmen dieser Arbeit deutlich übersteigen würde, soll an dieser Stelle jeweils ein zeit- und ein unterraumbasiertes Verfahren zur Implementierung auf einem DSP und zur weiteren Untersuchung ausgewählt werden.
Als Vertreter der zeitbasierten Verfahren wurde der MCCC ausgewählt, da dieser bei der Anwendung auf die aufgenommenen Sprachsignale die besten Ergebnisse bezüglich der Streuung des Winkels liefert.
Als Vertreter der unterraumbasierten Verfahren wurde MUSIC gewählt. Denn auch dieser Algorithmus lieferte im Vergleich zu den andren unterraumbasierten Verfahren die besten Ergebnisse. Dies liegt nicht an der Performance der Verfahren an sich, sondern an der Art der Implementierung des inkohärenten Breitbandansatz, wie in Abschnitt 3.2.3 beschrieben.

4. DSP-basierte Testumgebung

In diesem Kapitel wird zunächst eine geeignete DSP-Plattform für die Implementierung der Algorithmen ausgewählt und beschrieben. Anschließend wird das verwendete Mikrofonarray und dessen Komponenten vorgestellt. Daraufhin geht es um die Implementierung eines grafischen Benutzer-Interface zur Visualisierung und Aufzeichnung von Messergebnissen. Im letzten Teil folgt ein Gesamtüberblick des Systems.

4.1. Wahl einer geeigneten Hardware-Plattform

Ein Ziel der Arbeit war es, die behandelten Algorithmen in realer Umgebung zu testen. Dafür ist es notwendig, mehrere Mikrofon-Signale annähernd zeitgleich zu erfassen. Es muss ein System gefunden werden, das dies leistet. Weiterhin muss das System ausreichend Rechenleistung zur Verfügung stellen, damit die zu implementierenden Algorithmen auch darauf funktionieren können.
Grundsätzlich muss sich dabei zunächst für eine Technologie entschieden werden. Die Wahl der Programmiersprache fiel bereits vorher auf C, sodass FPGAs von vornherein ausschieden. Die engere Auswahl fiel auf eine PC-basierte Plattform mit externem Audio-Interface oder eine DSP-basierte Plattform mit Audiocodecs.
Da bei der PC-basierten Variante zwar sehr viel Rechenleistung und auch Speicher zur Verfügung steht, dafür allerdings mit nicht deterministischen Latenzzeiten bedingt durch das Betriebssystem und ähnlichen Problemen zu rechen ist, fiel die endgültige Wahl auf eine DSP-Plattform.
Auch in vorherigen Arbeiten im Bereich der Mikrofonarray-Signalverarbeitung wurden bereits DSP-Plattformen verwendet. Diese basierten auf dem TMS320C6713 von Texas Instruments, welcher mit einer maximalen Taktfrequenz von 300 MHz bis zu 1800 MFLOPS erreicht [24]. Diesen Prozessor gibt es integriert in eine vollständige Hardwareplattform von

verschiedenen Herstellern zu beziehen. Eine dieser Plattformen ist das D.Module.C6713 von D.SignT, welches folgende Eigenschaften bietet[24]:

- 128 MB SDRAM
- 2 MB Flash
- 256 kB internes RAM

Die nachfolgende Abbildung zeigt das Blockschaltbild der Plattform.

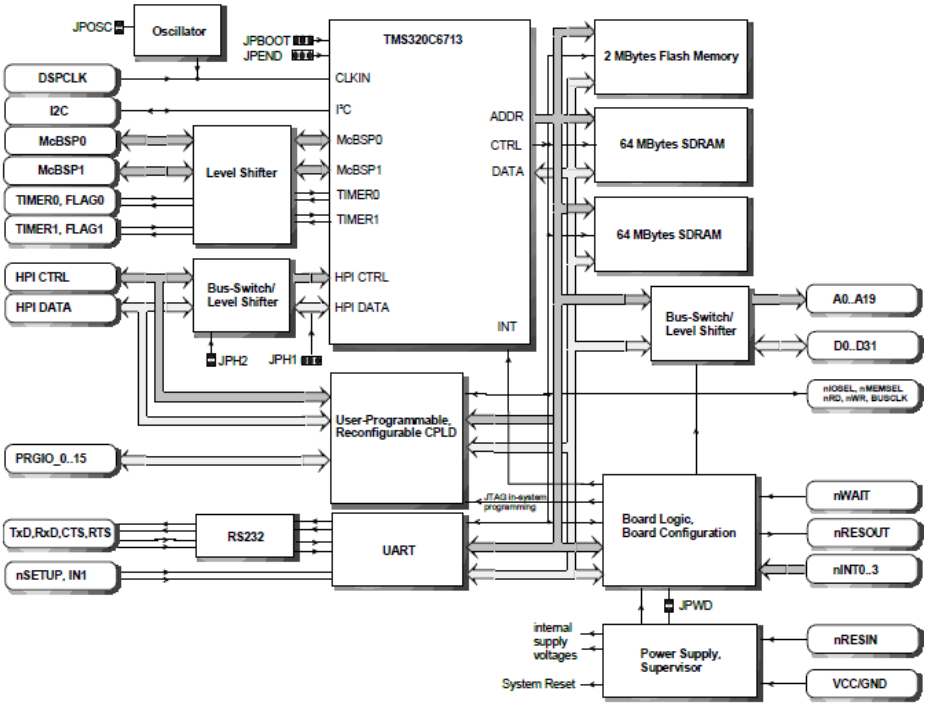


Abbildung 4.1.: Blockdiagramm D.Module.C6713 [24]

Diese Plattform wurde zunächst zur Implementierung der Algorithmen genutzt. Allerdings fiel sehr schnell auf, dass bedingt durch den geringen Speicherplatz des schnellen internen Speichers nur sehr kleine Blocklängen für den MCCC-Algorithmus genutzt werden konnten. Diese Tatsache wurde bereits in [32] festgestellt. Eine Abhilfe ist es, auf den ausreichend vorhandenen externen Speicher zurückzugreifen. Da ein wichtiger Bestandteil dieser Arbeit der Aufwandsvergleich der verschiedenen Algorithmen sein soll, ist diese

Vorgehensweise nicht zielführend. Denn bei der Verwendung von externem Speicher verzögern sich die Zugriffszeiten und eine Vergleichbarkeit wäre nur gegeben, wenn das bei allen Verfahren gleichermaßen der Fall wäre. Durch die Verschiedenheit der Verfahren ist dies nur schwer zu gewährleisteten. Außerdem sollen die Verfahren echtzeitfähig implementiert werden. Große Datenmengen, auf die häufig zugegriffen werden muss, sind dafür sehr hinderlich, wenn dazu lange Zugriffszeiten erforderlich sind,
Schließlich wurde das D.Module.C6657 basierend auf einem TMS320C6657 Signalprozessor, welcher ebenfalls von der Firma Texas Instrument stammt, verwendet. Dabei handelt es sich um eine neuere Generation von DSPs mit mehr Peripherie und mehr Speicher. Die folgende Tabelle soll die für die vorliegende Arbeit relevanten Eigenschaften gegenüberstellen.

Tabelle 4.1.: Vergleich zwischen C6657 und C6713

	C6657	C6713
Prozessor Kerne	2	1
max. Prozessakt/MHz	1250	300
chipinternes RAM	1 MB / Kern + 1 MB shared	256 kB
externes RAM	512 MB DDR3	128 MB SDRAM
Anzahl der Timer	2	2
Anzahl McBSP	2	2

Die folgende Abbildung zeigt das Blockschaltbild des D.Module.C6657.

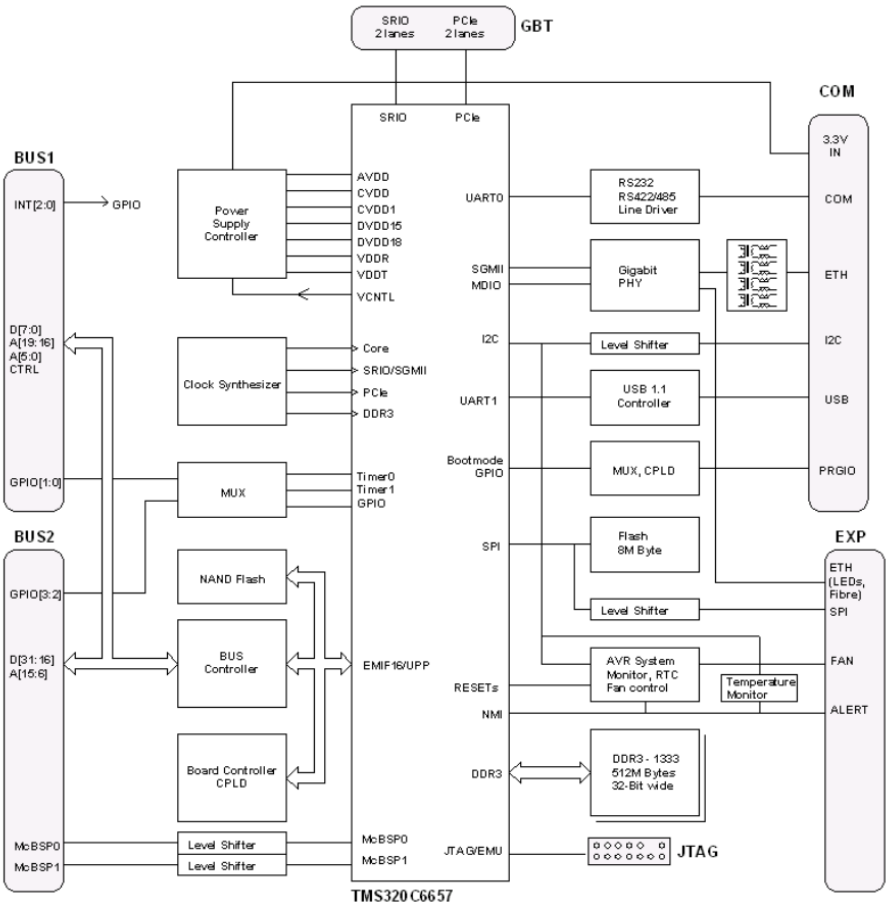


Abbildung 4.2.: Blockdiagramm D.Module.C6657 [25]

Für die Erfassung der Mikrofonsignale dient eine Audiocodec-Platine vom Typ D.Module.PCM3003, welche auf vier PCM3003 16 Bit Audiocodecs von Texas Instruments basiert. Damit ist es möglich bis zu acht Kanäle zur gleichen Zeit auszulesen, was dieses Modul für die Arbeit mit Mikrofonarrays besonders interessant macht. Es lassen sich sechs verschiedene Abtastraten zwischen 8 kHz und 48 kHz einstellen. Die Kommunikation zwischen DSP und D.Module.PCM3003 geschieht seriell über die beiden McBSP-Schnittstellen. Die folgende Abbildung zeigt den Aufbau des D.Module.PCM3003.

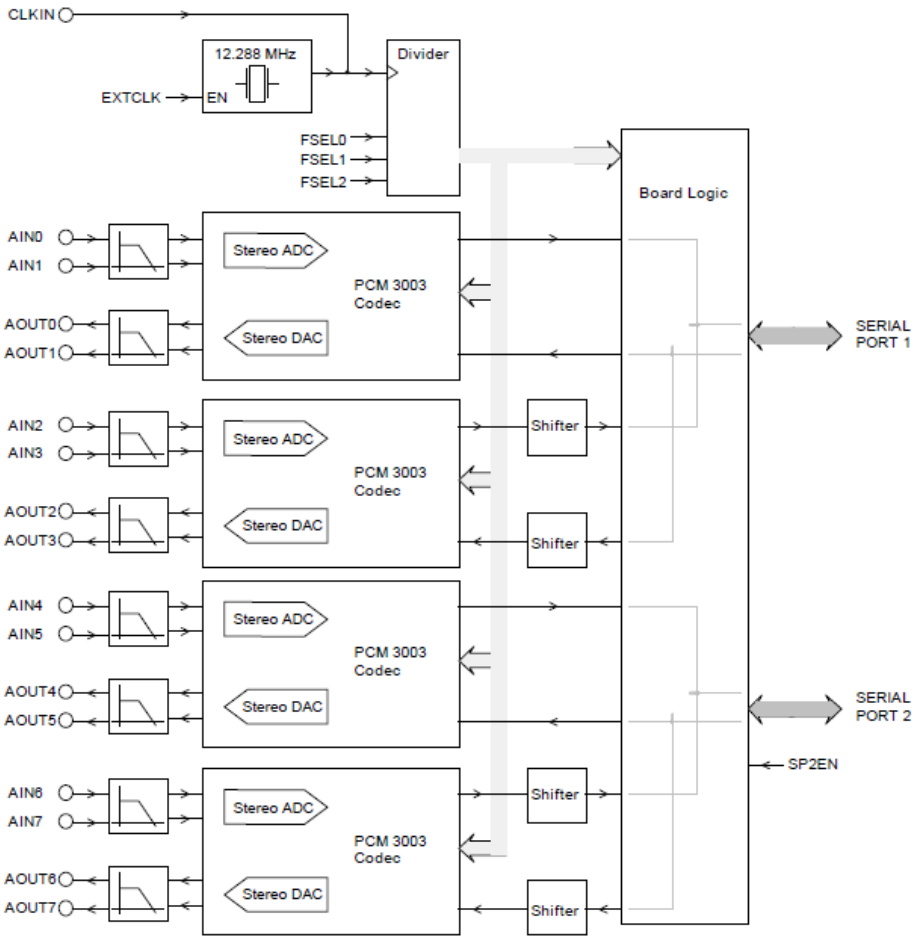


Abbildung 4.3.: Blockdiagramm D.Module.PCM3003 [26][34]

Ein Nachteil von diesem Modul ist, dass die Abtastraten fest über Lötbrücken eingestellt werden müssen. Für viele Anwendungen ist dies durchaus praktikabel. Hier sollen jedoch verschiedene Verfahren mit verschiedenen Abtastraten implementiert werden. Jede Veränderung der Abtastrate würde eine Veränderung der Hardware erfordern. Eine Möglichkeit dies zu umgehen ist es, anstelle der Lötbrücken Schalter einzusetzen. Hier wurde ein CMOS Baustein vom Typ 4066 verwendet, welcher vier bilaterale Schalter für stromrichtungsunabhängige Anwendungen beinhaltet. So ist es möglich die Abtastrate aus der Software, welche auf dem DSP läuft, mithilfe von GPIO-Pins einzustellen. Dafür wurde die folgende Schaltung auf einer Lochrasterplatine aufgebaut und in das D.Module.PCM3003 Board integriert.

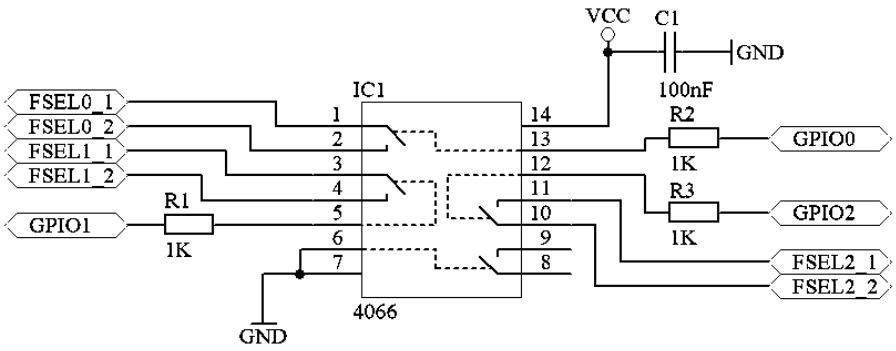


Abbildung 4.4.: Schaltung zur Wahl der Abtastfrequenz auf dem D.Module.PCM3003

4.2. Mikrofonarray

Bei dem in dieser Arbeit verwendeten Mikrofonarray handelt es sich um ein Uniform Linear Array, welches als modulares System konzipiert ist. Es besteht aus einer Aluminiumschiene, welche 15 Mikrofon-Module in einem Abstand von 5 cm aufnehmen kann. Die Module bestehen jeweils aus einem Kondensator-Mikrofon vom Typ WM-52BT, dessen Frequenzbereich zwischen 20 Hz und 16 kHz ist [30]. Außerdem ist eine Verstärkerschaltung direkt in diese Module integriert, welche folgendermaßen aufgebaut ist:

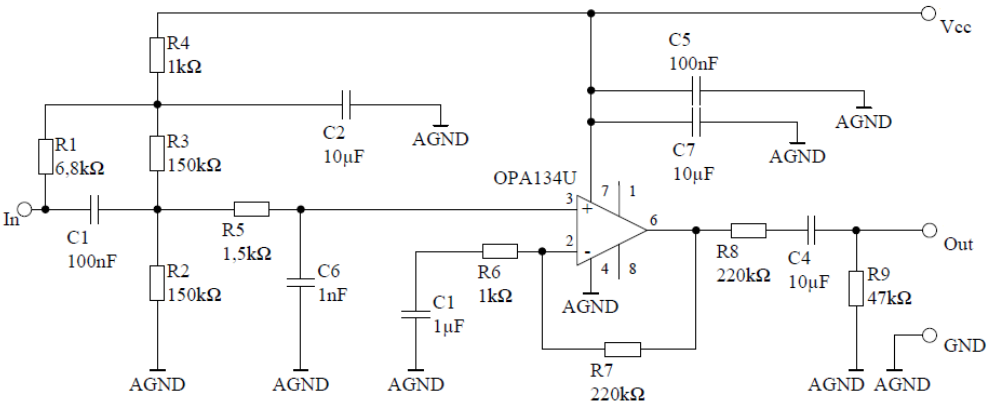


Abbildung 4.5.: Vertärkerschaltung für Mikrofon [34] (angepaast)

Diese Schaltung basiert auf einem nicht invertierenden Verstärker mit einem Operationsverstärker. Das Mikrofonsignal wird mit einem Offset versehen, um negative Spannungen

zu vermeiden. Nachdem es verstärkt wurde, sorgt ein Kondensator am Ausgang dafür, dass dieser Gleichanteil wieder entfernt wird.
Die Schaltung wird über ein dreipoliges 3,5 mm Klinkenkabel mit Spannung versorgt. Außerdem wird das Ausganssignal darüber geleitet. Die Klinkenkabel der Mikrofone werden an einem Hub angeschlossen, der gleichzeitig als Halter für die Aluschiene dient. Von diesem aus werden die Signale sowie die Leitungen für die Spannungsversorgung zusammengefasst in einem VGA-Kabel zu einer extra Platine geführt. Auf dieser befindet sich neben dem VGA-Stecker lediglich ein LDO mit Peripherie zur Spannungsversorgung der Mikrofone, sowie ein weiterer Stecker, über welchen die Mikrofonsignale direkt zu dem D.Module.PCM3003 geführt werden können. Die folgende Abbildung zeigt das verwendete Mikrofonarray, welches mit acht Mikrofon-Modulen bestückt ist.

Abbildung 4.6.: Mikrofonarray

4.3. GUI zur Visualisierung und Aufzeichnung von Messergebnissen

Das Ergebnis, welches die hier behandelten Algorithmen liefern, ist jeweils ein Winkel. In vielen Anwendungsfällen, zum Beispiel bei der adaptiven Auslegung eines Beamformers, wird dieser nur intern verwendet. Hier sollen jedoch lediglich die Verfahren zur Lokalisierung implementiert und getestet werden, sodass die Ergebnisse der Algorithmen aufgezeichnet bzw. sichtbar gemacht werden müssen. Eine Möglichkeit ist es, diese innerhalb des Programmcodes in einen Buffer zu schreiben und das Programm anzuhalten, sobald dieser gefüllt ist. Diese Methode bringt einige Nachteile mit sich. Ein entscheidender Nachteil ist, dass das Programm so nur aus der Entwicklungsumgebung im Debug-Modus ausgeführt

werden kann. Daher fiel die Entscheidung auf eine Schnittstelle zwischen DSP und einem PC, um die Ergebnisse direkt vom DSP zu übertragen und um sie auf dem PC anzeigen und auch speichern zu können.

4.3.1. Schnittstelle zwischen PC und DSP

Für die Kommunikation zwischen PC und DSP wurde die RS232-Schnittstelle gewählt. Diese bietet den Vorteil, dass bei der Übertragung sehr wenig Overhead benötigt wird. Natürlich bedeutet ein Verzicht auf diesen Overhead auch ein Verzicht auf Fehlerkorrekturmaßnahmen und Ähnliches. Eine weitere Einschränkung der seriellen Schnittstelle ist, dass pro Nachricht lediglich ein Byte versendet werden kann. Soll wie im vorliegenden Fall eine Variable vom Typ float versendet werden, muss diese auf mehrere Nachrichten aufgeteilt werden. Hierzu wurde ein Protokoll eingeführt. Sobald aufeinander folgend zunächst das Byte “0xff” und anschließend “0x00” empfangen wird, folgt der Nutzdateninhalt in den folgenden vier Bytes.

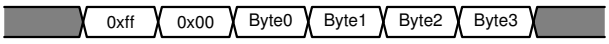


Abbildung 4.7.: Datenpaket bei Übertragung über die serielle Schnittstelle

Mittels der zuvor beschriebenen Pakete lassen sich Inhalte von Variablen des Typs *float* von der DSP-Plattform aus wie folgt versenden:

Listing 4.1: Versenden einer Variable vom Typ *float* über die serielle Schnittstelle

```
1  void send_float(T_Handle handle,float value){
2
3  char buf[6];
4
5  buf[0] = 0xff;
6  buf[1] = 0x00;
7  buf[2] = ((uint8_t*)&value)[0];
8  buf[3] = ((uint8_t*)&value)[1];
9  buf[4] = ((uint8_t*)&value)[2];
10 buf[5] = ((uint8_t*)&value)[3];
11
12 DM2_uartWriteBlock(handle, buf ,6, DM2_UART_NOTIMEOUT);
13 }
```

Auch die Nachrichten vom PC an den DSP beginnen jeweils mit derselben Kennung. Anschließend folgen zwei Bytes zur Einstellung der Abtastrate und des gewünschten Algorithmus. Die letzten beiden Bytes bleiben für künftige Erweiterungen reserviert.

4.3.2. Grafisches Benutzer-Interface

Die GUI, in welcher die gemessenen DOAs zur Anzeige gebracht, Einstellungen vorgenommen sowie Messergebnisse gespeichert werden können, wurde mit Qt erstellt. Dabei handelt es sich um eine C++ Klassenbibliothek für die Programmierung grafischer Oberflächen. So ist es mit dem Qt-Designer leicht möglich, Benutzeroberflächen zu gestalten. In Qt wird mit dem Signal/Slot-Konzept gearbeitet. So werden Verbindungen zwischen Objekten hergestellt. Ein Beispiel hierfür wäre ein Fenster mit einem Button und einem Textfeld. Der Button ist ein Objekt, welches ein Signal aussendet. Das Textfeld ist in diesem Beispiel ein Objekt, welches einen Slot beinhaltet. Ein Slot ist eine Aktion des Empfängers. Hier könnte es die Anzeige eines Textes sein, der in dem Textfeld angezeigt wird. Der Sender wäre somit der Button und der Empfänger das Textfeld. Nachdem der Button geklickt wurde, erscheint ein Text im Textfeld. Ein Programm kann diverse dieser Verbindungen beinhalten. Die Signale, die ein Objekt erzeugen kann, hängen dabei von den jeweiligen Objekten ab. Dieser Ansatz eignet sich besonders gut für GUIs, da hier Benutzereingaben reaktiv behandelt werden. [20]
Der vom Algorithmus ermittelte Winkel soll hier über der Zeit bzw. über den verwendeten Datenblock angezeigt werden. Diese Anzeigemöglichkeit wird präferiert, da eine Anzeige als Zahlenwert unpraktisch ist. Das liegt daran, dass sehr schnell neue Daten ankommen, sodass sich der angezeigte Wert so schnell ändern würde, dass dies kaum noch wahrgenommen werden kann. Wird zum Beispiel mit einer Blocklänge von 512 Samples bei einer Abtastfrequenz von 48 kHz gearbeitet, so werden annähernd 94 Winkel pro Sekunde ermittelt. Sofern davon ausgegangen wird, dass jeder Block auch verarbeitet wird.
Für die Darstellung der Winkel über den verwendeten Blöcken wurde die Qwt-Bibliothek verwendet. Diese beinhaltet diverse Klassen für technische Anwendungen. So auch die Möglichkeit Graphen zu darzustellen.
Insgesamt besteht das Programm im Wesentlichen aus zwei Ansichten. Die Hauptansicht beinhaltet den zuvor beschreibenden Qwt-Plot sowie ein Textfeld zur Eingabe eines Dateipfades und einige Steuerelemente. Hierin findet die Anzeige des ermittelten Winkels statt.

Außerdem besteht die Möglichkeit, die Kurve im csv-Format zu speichern. Mit den Steuerelementen lässt sich zwischen den Ansichten wechseln und eine Aufzeichnung starten oder stoppen. Ein weiteres Steuerelement öffnet ein Dialogfenster zur Konfiguration der seriellen Schnittstelle.

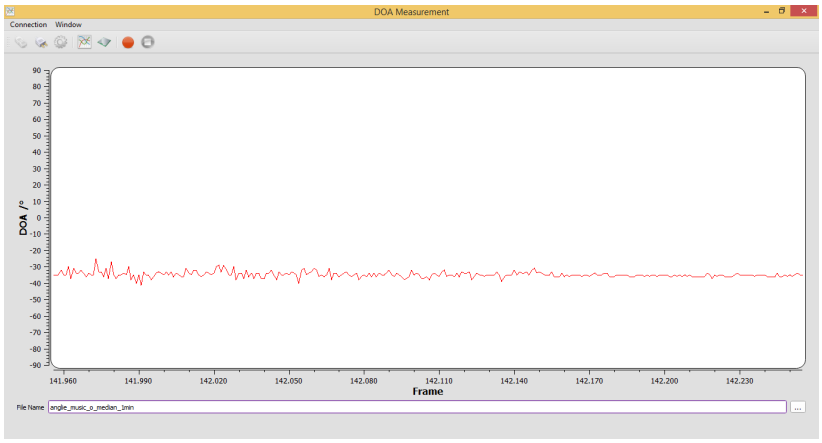


Abbildung 4.8.: Benutzeroberfläche des Mess-Tools

Die Kurve wird mit jedem neuen Eingangsdatum aktualisiert, sodass sich ein fortlaufender Plot ergibt.
Die zweite Ansicht dient dazu, Parameter auszuwählen und sie an den DSP zu senden. Nachdem Parameter ausgewählt wurden, wird mit den Sendebutton ein Signal erzeugt, welches veranlasst, dass die Parameter ausgelesen und über die serielle Schnittstelle versendet werden. Die Ansicht wechselt daraufhin zur Hauptansicht.

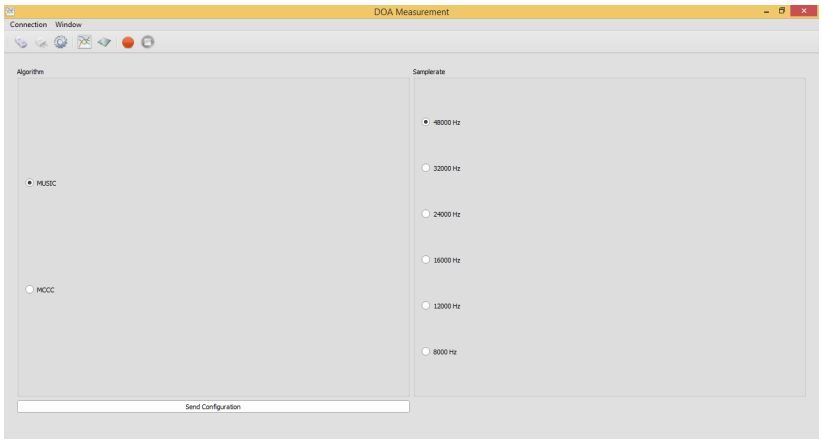


Abbildung 4.9.: Konfigurationsansicht des Mess-Tools

4.4. Aufbau des Gesamtsystems

Das Gesamtsystem besteht aus einer DSP-Plattform vom Typ D.Module.C6657, einer Grundplatine vom Typ D2.Base, welche die Spannungsversorgung der DSP-Plattform übernimmt und außerdem dazu dient, Erweiterungen anzuschließen. Zudem wird ein Codecmodul vom Typ D-Module.PCM3003 verwendet, welches über einen Adapter vom Typ SON.D2-DM an die Hauptplatine angeschlossen wird und um einen IC vom Typ 4066 zur Wahl der Abtastfrequenzen erweitert wurde. Ein weiterer Bestandteil ist ein Mikrofonarray bestehend aus acht Mikrofonmodulen mit integriertem Verstärker, welche in einer Aluminiumschiene eingesetzt sind. Alle Mikrofonmodule sind über einen Hub mit einer Zusatzplatine verbunden, welche die Spannungsversorgung der Module übernimmt und außerdem mit einem Flachbandkabel mit den Eingängen des D.Module.PCM3003 verbunden ist. Über dieses werden die Signale der Mikrofone geleitet. Weiterhin findet eine Kommunikation zwischen einem PC und dem DSP mittels RS232 statt, welche zur Ergebnisanzeige und Parametrierung dient. Die folgende Abbildung zeigt eine Übersicht des gesamten Systems mit den signalrelevanten Verbindungen.

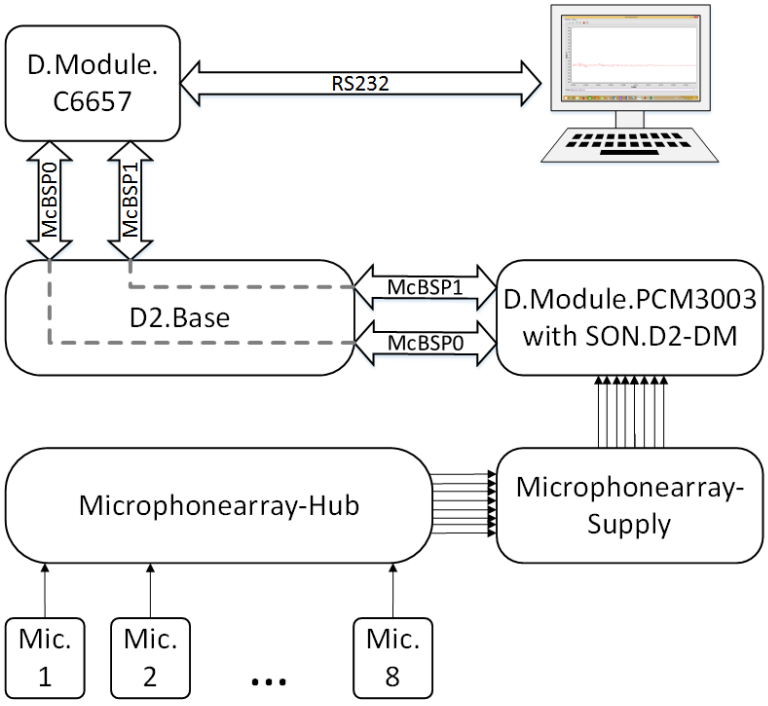


Abbildung 4.10.: Übersicht Gesamtsystem

5. Implementierung der Algorithmen auf dem DSP

Um die Algorithmen unter realen Gegebenheiten testen zu können, war es notwendig, diese auf einem Echtzeitsystem zu implementieren. Dazu wurde das zuvor vorgestellte DSP-basierte Echtzeitsystem genutzt. Um bei den späteren Tests nicht ständig andere Software auf den DSP übertragen zu müssen, sollen beide Algorithmen in demselben Programm implementiert werden. Die Auswahl des zu verwendenden Algorithmus, sowie andere Konfigurationen finden über die serielle Schnittstelle statt.
Dieses Kapitel behandelt die Implementierung der Algorithmen in C. Dazu wird zunächst auf die jeweiligen algorithmusspezifischen Funktionen eingegangen und anschließend das gesamte Programm vorgestellt.

5.1. Implementierung des MCCC-Algorithmus

5.1.1. Korrelation mittels schneller Faltung

Für den MCCC ist es notwendig eine hohe Anzahl an Korrelationen durchzuführen, da jedes Signal von den Mikrofonen mit sich selbst und mit allen anderen korreliert werden muss, um die Korrelationsmatrizen für verschiedene angenommene Zeitverschiebungen p zu bilden. Bei der Betrachtung der Eigenschaften der Korrelationsmatrix fällt sofort die erste Vereinfachung auf. Da es sich um eine symmetrische Matrix handelt, genügt es, die Werte der Hauptdiagonalen sowie die Werte oberhalb dieser zu berechnen. Die übrigen Werte ergeben sich aus den zuvor berechneten. Bei einer Anzahl von N Mikrofonen müssen so lediglich $(N^2+N)/2$ Korrelationen berechnet werden. Weiterhin lässt sich zeigen, dass auf

der Hauptdiagonalen, unabhängig von der Variablen p, immer die Varianzen der Mikrofonsignale stehen, sodass es hier genügt diese anstelle der Autokorrelationsfunktionen (AKF) zu berechnen. Die Anzahl der benötigten Kreuzkorrelationsfunktion (KKF) reduziert sich somit zu $N^2/2$-N. Die KKF zweier Signale ist definiert als:

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y(n+k) \tag{5.1}$$

Es lässt sich leicht die Ähnlichkeit zur Faltung erkennen, welche folgendermaßen definiert ist:

$$c_{conv,xy}(n) = x(n) * y(n) = \sum_{k=-\infty}^{\infty} x(k)y(n-k) \tag{5.2}$$

Die Faltung lässt sich bekanntermaßen als so genannte schnelle Faltung im Frequenzbereich durchführen, wodurch eine nicht unerhebliche Rechenzeitersparnis zustande kommt. Dies liegt an der Vielzahl an Multiplikationen und Additionen, die bei einer Berechnung im Zeitbereich notwendig sind. Eine Möglichkeit, die Kreuzkorrelation auf ähnliche Weise im Frequenzbereich durchzuführen, würde somit einen enormen Vorteil mit sich bringen.So ein Ansatz ist in [32, S. 19ff.] beschrieben.
Die Formel zur Beschreibung der KKF lässt sich folgendermaßen umformen:

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y(k-(-n)) \tag{5.3}$$

Weiterhin kann anstelle von $x(n)$ der Term $x(-(-n))$ eingesetzt werden, wodurch folgender Zusammenhang entsteht:

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(-(-n))y(k-(-n)) \tag{5.4}$$

Ein Vergleich von Gleichung 5.4 und Gleichung 5.2 zeigt, dass sich die Kreuzkorrelation nun als Faltungsoperation mit der Möglichkeit einer Durchführung im Frequenzbereich ausdrücken lässt.

$$r_{xy}(k) = x(-k) * y(k) \circ\!\!-\!\!\bullet R_{corr} = X(-l) * Y(l) \tag{5.5}$$

Somit ergibt sich für die KKF im Frequenzbereich mit $X(-l) = X^*(l)$:

$$r_{xy}(k) = \mathcal{F}^{-1}\left\{X^*(l) * Y(l)\right\} \tag{5.6}$$

In der Software muss dafür zunächst jedes Signal mit Nullen aufgefüllt werden, da das Ergebnis einer diskreten Faltung zweier Blöcke ein Block der doppelten Länge abzüglich eines Samples ist. Bei der Durchführung im Frequenzbereich ist dies nicht möglich. Hier müssen beide Signale gleichermaßen auf die doppelte Länge gebracht werden. Anschließend werden sie mittels schneller Fouriertransformation in den Frequenzbereich transformiert. Anders als bei der Faltung wird nun das erste Spektrum konjungiert, bevor beide miteinander multipliziert werden. Diese beiden Schritte wurden in der tatsächlichen Implementierung zusammengefasst. Das Produkt wird nun mittels IFFT in den Zeitbereich transformiert. Damit die Reihenfolge der Elemente im Ergebnisblock mit der der Elemente der Korrelation im Zeitbereich übereinstimmt, muss ein FFT-Shift durchgeführt werden. Das Ergebnis ist bis auf den Unterschied, dass es um ein Element länger ist, identisch mit dem Ergebnis im Zeitbereich. Korrigieren lässt sich das, indem das erste Element entfernt wird. Die beiden letzten Schritte bestehend aus FFT-Shift und anschließenden Entfernen des ersten Elements wurden in der Implementierung ausgelassen, da diese zusätzliche Rechenzeit kosten und ein entsprechend angepasster Zugriff auf die Elemente dies überflüssig macht.
Die folgende Abbildung zeigt den Ablauf der Kreuzkorrelation im Frequenzbereich.

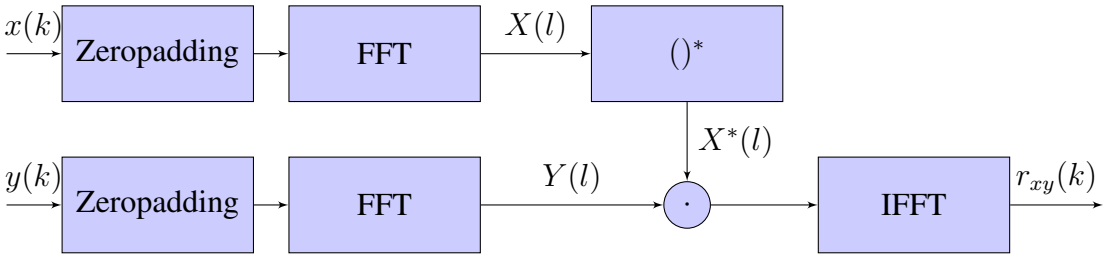


Abbildung 5.1.: Darstellung des Ablaufs einer Kreuzkorrelation im Frequenzbereich

Für die FFT und die IFFT werden hier jeweils Funktionen aus der TMS320C6000 DSP Library von Texas Instrument verwendet. Dies hat den Vorteil, dass es sich hierbei um gut getestete und handoptimierte Funktionen handelt. Ein Nachteil ist, dass diese Bibliothek plattformabhängig ist. Soll der Code auf einen DSP eines anderen Herstellers bzw. einer anderen Prozessorfamilie desselben Herstellers oder auf einer ganz anderen Plattform portiert

werden, können FFT und IFFT ggf. nicht verwendet werden.
Die soeben genannte Bibliothek beinhaltet diverse Varianten der FFT bzw. IFFT. Diese unterscheiden sich sowohl in den Dimensionen (eindimensional oder zweidimensional) als auch den Datentypen der Eingangs- und Ausgangsblöcke. In der vorliegenden Arbeit wurden die Funktionen: *DSPF_sp_fftSPxSP* und *DSPF_sp_ifftSPxSP* verwendet. Diese Funktionen benötigen einen komplexwertigen Eingansblock vom Typ *float*. Wichtig ist, dass diese im Speicher hintereinander angeordnet sind, sodass der Imaginärteil eines Elements immer auf den Realteil folgt und anschließend der Realteil des nächsten Elements usw. Dies lässt sich gewährleisten, indem der Compiler mittels Pragma-Direktive dazu angewiesen wird. So lässt sich Array vom Typ *float*, bei dem sichergestellt sein soll, dass die Elemente korrekt angeordnet sind, wie folgt anlegen:

Listing 5.1: Sicherstellung der korrekten Speicheranordnung eines float-Arrays

```
1 #pragma DATA_ALIGN(x_i,  2*sizeof(float));
2 float x_i [2*K];
```

Um leichter auf den Real- bzw. Imaginärteil zugreifen zu können, wurde mittels Typendefinition eine Struktur definiert, welche zwei Variablen vom Typ *float* beinhaltet. Davon kann ebenfalls ein Array angelegt werden, welches ebenso mittels Pragma-Direktive im Speicher angeordnet werden kann.

Listing 5.2: Sicherstellung der korrekten Speicheranordnung eines Arrays vom Typ COMPLEX

```
1 typedef struct COMPLEX_TAG{
2 float re,im;
3 } COMPLEX;
4 #pragma DATA_ALIGN(x_i,sizeof(COMPLEX))
5 COMPLEX x_i[K];
```

Real- und Imaginärteil nehmen zusammen pro komplexer Variable acht Byte Speicherplatz in Anspruch, sodass sich die Anordnung wie folgt darstellt:

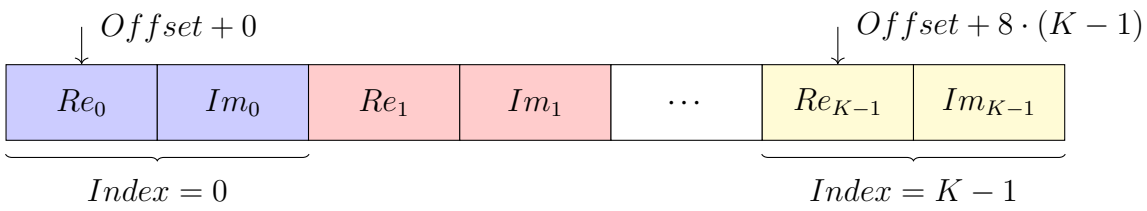


Abbildung 5.2.: Für FFT und IFFT benötigte Anordnung komplexer Variablen im Speicher

Die Übergabe an die FFT als auch an die IFFT erfolgt via Zeiger auf die Anfangsadresse des Arrays. Der Funktionsprototyp lautet am Beispiel der FFT wie folgt:
*void DSPF_sp_fftSPxSP(int N, float *ptr_x, float *ptr_w, float *ptr_y, unsigned char *brev, int n_min, int offset, int n_max).*
Darin sind laut [40]:

N Blocklänge des komplexen Eingangs. Es wird eine Zweierpotenz vorausgesetzt. Die Werte dürfen zwischen 8 und 65536 liegen.

ptr_x Zeiger auf Eingangs-Array. Dieses muss die Größe 2 K haben und wie zuvor beschrieben im Speicher angeordnet sein.

ptr_w Zeiger auf komplexes Array mit Twiddle-Faktoren der Länge 2 K. Diese müssen ebenfalls wie zuvor beschrieben im Speicher angeordnet sein.

ptr_y Zeiger auf Ausgangs-Array. Dieses muss die Größe 2 K haben und wie zuvor beschrieben im Speicher angeordnet sein.

brev Zeiger auf eine spezielle Bitreversal-Tabelle mit 64 Einträgen.

n_min Kleinstes FFT-Butterfly, welches benutzt werden kann. Wenn K eine Viererpotenz ist, wird eine reine Radix 4 FFT durchgeführt. Ansonsten wird eine gemischte FFT durchgeführt, denn muss für n_min der Wert 2 übergeben werden.

offset Index (in komplexen Samples) von welchen an die FFT ausgeführt werden soll (typisch 0).

n_max Länge der FFT (typisch K).

Die KKF werden ausgeführt, indem zunächst für alle N Signalblöcke eine FFT mit vorhergegangenem Zeropadding ausgeführt wird. Die Ergebnisse der FFT werden anschließend miteinander multipliziert, wobei im selben Schritt die Invertierung der ersten komplexen Variable vorgenommen und anschließend die IFFT gebildet wird. Das Resultat sind $N^2/2$-N Blöcke des komplexen Datentyps, wobei der Imaginärteil jeweils null ist. Die Blöcke sind in einem zweidimensionalen Array folgendermaßen angeordnet:

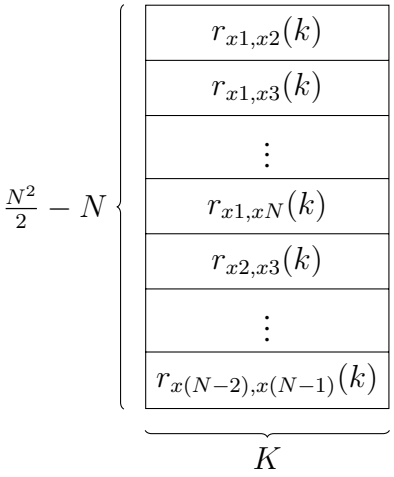


Abbildung 5.3.: Anordnung der KKF in zweidimensionalem Array

5.1.2. Erstellung der parametrierbaren Korrelationsmatrix und Berechnung der Determinante

Nachdem alle KKF berechnet wurden, können diese genutzt werden, um daraus die von der Variablen p abhängige Korrelationsmatrix zu erstellen und deren Determinante zu berechnen. Auf der Hauptdiagonalen stehen - wie bereits beschrieben - die Varianzen der einzelnen Signale, welche nicht von p abhängen. Darum genügt es, diese anstelle der AKF einmalig wie folgt zu berechnen:

$$var_{x_n} = E[x_n^2] = \frac{1}{K} \sum_{i=0}^{K-1} x_n^2(i) \tag{5.7}$$

Die übrigen Elemente der Matrix werden abhängig von der Variable p aus den zuvor berechneten KKF entnommen. Die Funktion *Ra_p_extra_vars* übernimmt genau die zuvor beschriebene Funktionalität, indem ihr als Parameter ein Zeiger auf das erste Element eines zweidimensionalen N x N großen Arrays, in welchem die Matrix $R_a(p)$ hinterlegt ist sowie ein Zeiger auf das erste Element der KKF und ein Zeiger auf das erste Element eines N-Elemente umfassenden Arrays mit den Varianzen erhält.
Zunächst wird das Vorzeichen von p geprüft. Der zuvor beschriebene Wegfall des FFT-Shift erfordert dies. Im Normalfall würde das mittlere Array-Element den Wert für eine Zeitverschiebung von null beinhalten. Ein Inkrementieren des Index würde einer positi-

ven und eine Dekrementierung einer negativen Zeitverschiebung entsprechen. Da auf einen FFT-Shift verzichtet wurde, befindet sich das Element mit der Zeitverschiebung von null am Index eins. Eine Inkrementierung bewirkt nun ebenfalls eine positive Zeitverschiebung. Dies gilt für die erste Hälfte des Arrays. In der zweiten Hälfte befinden sich die negativen Zeitverschiebungen. Hier ist die kleinste Verschiebung im letzten Element des Arrays. Eine Dekrementierung des Index bedeutet eine größer werdende Verschiebung in den negativen Bereich. Somit findet bei der Erstellung der Matrix eine Fallunterscheidung statt. Ist das Vorzeichen negativ, wird von dem letzten Element im Array aus dekrementiert. Die Verschiebung hängt jeweils von den Abstand zwischen den derzeit betrachteten Mikrofonen ab und ist nichts Anderes als die Umsetzung der Funktion:

$$\mathcal{F}_n(p) = (n-1) \cdot p, \quad (n = 2, 3, \ldots, N) \tag{5.8}$$

Nachdem unterschieden wurde, ob es sich bei p um eine Variable mit negativen oder positiven Vorzeichen handelt, wird die Matrix $R_a(p)$ wie folgt zusammengesetzt:

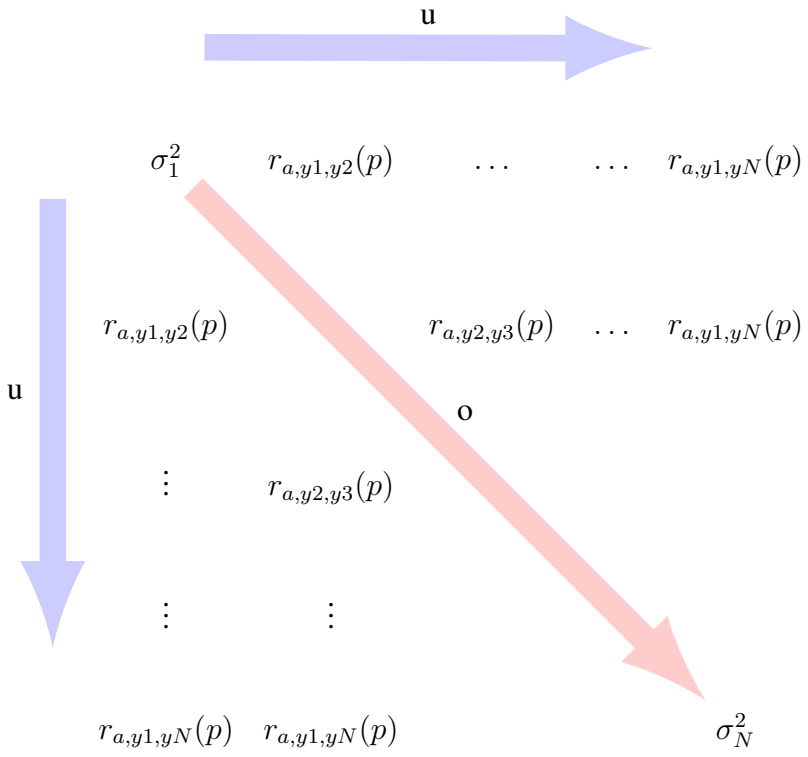


Abbildung 5.4.: Zusammensetzung der Matrix $R_a(p)$

Dies geschieht mittels zwei verschachtelter Schleifen, wobei die Äußere mit der Zählervariablen ”o” entlang der Hauptdiagonalen läuft und die Innere mit der Zählervariablen ”u” jeweils alle Elemente rechts und unter dem Element auf der Hauptdiagonalen einsetzt. Durch die Symmetrie der Matrix sind diese Werte identisch.
Die Berechnung der Determinante erfolgt ebenfalls für jeden Wert von p. Eine Möglichkeit der Berechnung ist die Methode mit Unterdeterminanten. Dabei wird die Matrix so lange in Unterdeterminanten aufgeteilt, bis die zu berechnenden Determinanten eine Ordnung aufweisen, bei der diese mit einfachen Mitteln zu berechnen sind. Diese Methode hat den Nachteil, dass eine Umsetzung in einem Programm, bei dem die Dimension der Matrix variieren darf, sehr aufwendig wäre. Aus diesem Grund wurde auf ein anderes Verfahren zurückgegriffen. Dazu wird zunächst eine elementare Rechenregel bezüglich Determinanten eingeführt. Sei eine Matrix $\boldsymbol{R}$ folgendermaßen darstellbar:

$$\boldsymbol{R} = \boldsymbol{L} \cdot \boldsymbol{U} \tag{5.9}$$

so gilt:

$$det(\boldsymbol{R}) = det(\boldsymbol{L} \cdot \boldsymbol{U}) = det(\boldsymbol{L}) \cdot det(\boldsymbol{U}) \tag{5.10}$$

Diese Regel lässt sich ausnutzen, indem jeweils eine Matrix $\boldsymbol{L}$ und eine Matrix $\boldsymbol{U}$ gefunden werden, mit denen Gleichung 5.9 gilt. Durch Zuhilfenahme der LU-Faktorisierung lassen sich ebensolche Matrizen finden, die zusätzlich folgende Eigenschaften aufweisen:

$\boldsymbol{L}$ Ist ein untere Dreiecksmatrix, welche auf der Hauptdiagonalen nur Einsen aufweist.

$\boldsymbol{U}$ Ist eine obere Dreicksmatrix

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{1,0} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{(N-1),1} & l_{2,(N-1)} & \dots & 1 \end{pmatrix} \tag{5.11}$$

$$\boldsymbol{U} = \begin{pmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,(N-1)} \\ 0 & u_{1,1} & \dots & u-1,(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{(N-1),(N-1)} \end{pmatrix} \tag{5.12}$$

Es lässt sich leicht erkennen, dass durch die besondere Struktur der Matrizen die Determinante auf einfache Weise bestimmt werden kann. Diese ist gegeben durch das Produkt der Elemente auf der Hautdiagonalen der Matrix $\boldsymbol{U}$.

$$det(\boldsymbol{R}) = \prod_{i=1}^{N} u_{ii} \tag{5.13}$$

Die LU-Faktorisierung gestaltet sich wie folgt [21, S. 35]:

Algorithm 1 LU-Faktorisierung

function LU($\boldsymbol{R}$)
  Setze $l_{ii} = 1$ für alle i
  for all $j(Spalten)$ **do**
    for all $i = 0, 1, \ldots, j$ **do**
      $u_{ij} = a_{ij} - \sum_{k=0}^{i-1} l_{ik} u_{kj}$
    end for
    for all $i = j+1, j+2 \ldots, N-1$ **do**
      $l_{ij} = \dfrac{a_{ij} - \sum_{k=0}^{i-1} l_{ik} u_{kj}}{u_{jj}}$
    end for
  end for
return $\boldsymbol{L}, \boldsymbol{U}$
end function

5.1.3. Bestimmung des Winkels

Der gesamte Algorithmus läuft wie in Abbildung 2.5 dargestellt ab. Dazu muss der minimale sowie maximale Wert für p bekannt sein. Dieser ergibt sich aus der maximal möglichen Auflösung sowie des Suchbereichs.

$$\kappa = floor\left(1 + \frac{d}{c} \cdot sin(\phi) \cdot f_a\right) \tag{5.14}$$

Soll beispielsweise zwischen 50° und minus 50° gesucht werden, so ergibt sich für p ein Wertebereich zwischen minus und plus sechs bei der verwendeten Abtastrate und dem Mikrofonabstand. Das bedeutet, dass insgesamt 13 Korrelationsmatrizen erstellt und deren De-

terminanten berechnet werden müssen. Alle Werte werden in einem Array hinterlegt. Nach der Berechnung aller Determinanten muss in dem Array das kleinste Minimum gefunden werden. Aus dem Index der Stelle im Array, an der das Minimum gefunden wurde, ergibt sich der korrespondierende Winkel folgendermaßen:

$$\phi = asin\left(\frac{index - \kappa_{max} \cdot c}{f_a \cdot d}\right); \tag{5.15}$$

Anstatt jedes Mal den Arcussinus zu berechnen, wäre es deutlich effektiver dies für jeden Wert von κ ein Mal vor der Laufzeit zu tun und die Werte in einer Lookup-Tabelle so zu hinterlegen, dass mittels des ermittelten Index leicht darauf zugegriffen werden kann. Dieses Vorgehen wird für eine Implementierung in einer konkreten Anwendung des Algorithmus empfohlen. Hier sollte der Code jedoch flexibel für Veränderungen der Parameter gehalten werden, sodass die Berechnung pro Durchlauf ein Mal ausgeführt wird. Der Mehraufwand an Rechenleistung dadurch fällt im Bezug zum Gesamtaufwand jedoch sehr gering aus.

5.2. Implementierung des MUSIC-Algorithmus

5.2.1. Aufbereitung der Daten

Da der MUSIC-Algorithmus Signale im Frequenzbereich voraussetzt, muss zunächst eine Transformation der Signale in diesen stattfinden. Das geschieht mithilfe der FFT, wie sie unter Abschnitt 5.1.1 beschrieben wurde.
Daraufhin werden alle Elemente des Arrays, in welchen das MUSIC-Spektrum hinterlegt ist, zu null gesetzt. Dies ist notwendig, da hier eine Breitband-Variante des Algorithmus angewendet wird, in der die Werte für jedes Band zum MUSIC-Spektrum hinzuaddiert werden. Würden die Werte nicht zu null gesetzt werden, hätte das eine Addition der Werte zu denen der vorherigen Berechnung zur Folge.

5.2.2. Berechnung des MUSIC-Spektrum

Die Berechnung des MUSIC-Spektrum erfolgt in der vorliegenden Implementierung für vier Frequenzstützpunkte, sodass die meisten der im Folgenden beschriebenen Verfahren

insgesamt vier Mal durchgeführt werden müssen.
Ein Unterschied zu dem bislang in dieser Arbeit beschriebenen Vorgehen ist, dass hier nicht auf die Eigenwertzerlegung der Kovarianzmatrix zurückgegriffen wurde. Stattdessen wurde eine Singulärwertzerlegung verwendet. Diese hat den bedeutsamen Vorteil, dass sie anders als die Eigenwertzerlegung nicht lediglich auf quadratische, sondern beliebige Matrizen anwendbar ist. Die Definition der Singulärwertzerlegung lautet wie folgt [15]:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V} \tag{5.16}$$

Darin ist $\boldsymbol{A}$ eine Matrix der Dimension M x N. Für diese gibt es die orthogonalen Matrizen $\boldsymbol{U}$ und $\boldsymbol{V}$ sowie eine Diagonalmatrix $\boldsymbol{W}$ mit den nicht negativen Singularwerten $w_{11} > w_{22} > \ldots$. Da die Matrix $\boldsymbol{A}$ nicht quadratisch sein muss, ist es möglich, die Ausgangs-Matrix der Mikrofone direkt zu nutzen. In [36] wird gezeigt, wie Eigenwertzerlegung und Singulärwertzerlegung zusammenhängen. Die Matrix $\boldsymbol{U}$ in Gleichung 5.16 wird als linke Eigenvektormatrix bezeichnet und beinhaltet die Eigenvektoren von $\boldsymbol{A}\boldsymbol{A}^T$. Die Matrix $\boldsymbol{V}$ wird als rechte Eigenvektormatrix bezeichnet und enthält die Eigenvektoren von $\boldsymbol{A}^T\boldsymbol{A}$. Mit Gleichung 5.16 gilt:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{V}^T \tag{5.17}$$

Darin handelt es sich bei dem Produkt $\boldsymbol{W}^T\boldsymbol{W}$ um eine Diagonalmatrix mit den quadrierten Singulärwerten, welche bei einem Vergleich mit Gleichung 2.66 offensichtlich der Diagonalmatrix der Eigenwerte von $\boldsymbol{A}^T\boldsymbol{A}$ entspricht. Somit bleibt festzuhalten, dass die Singulärwertzerlegung der Matrix $\boldsymbol{A}$ mit der Matrix $\boldsymbol{V}$ die Eigenvektoren der Matrix $\boldsymbol{A}^T\boldsymbol{A}$ liefert. Die Matrix $\boldsymbol{W}$ entspricht hier der Diagonalmatrix der Eigenwerte von $\boldsymbol{A}^T\boldsymbol{A}$, mit dem Unterschied, dass es sich jeweils um die Quadratwurzel dieser handelt.
Die Matrix $\boldsymbol{U}$ spielt eine Rolle, wenn die Eigenvektoren von $\boldsymbol{A}\boldsymbol{A}^T$ gefunden werden sollen. Denn gilt folgender Zusammenhang:

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{W}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{U}^T \tag{5.18}$$

Diese Betrachtungen lassen es durchaus sinnvoll erscheinen, die Singulärwertzerlegung für die Suche nach den Eigenvektoren zu benutzen.
Für die Implementierung der SVD gibt es verschiedene Ansätze. Der von Golub und Kahan [19] vorgestellte Ansatz beruht darauf, zunächst die Matrix $\boldsymbol{A}$ zu einer Diagonalmatrix zu

reduzieren und davon die Eigenwerte zu berechnen. Es existieren diverse bereits implementierte C-Funktionen, die diesen Ansatz umsetzen. So wurde eine Funktion verwendet, die die Möglichkeit bietet, direkt mit komplexen Matrizen zu arbeiten und unter die *GNU GENERAL PUBLIC LICENSE* fällt. Diese verwendet bzw. erwartet komplexwertige Datentypen, welche nach Einbindung des Headers "complex.h" zur Verfügung stehen. Die Übergabeparameter sind:

float _Complex a M x N Matrix, für welche die SVD durchgeführt werden soll.

int MMAX Anzahl der Zeilen von a und u.

int NMAX Anzahl der Spalten von a und v.

int NU Anzahl der Spalten, die von u berechnet werden sollen.

int NV Anzahl der Spalten, die von v berechnet werden sollen.

int p Vektor in a, ab welchem die SVD berechnet werden soll.

float w Vektor, in dem die Ergebnis-Singulärwerte stehen.

float _complex u Matrix der Dimension MMAX x NU.

float _complex v Matrix der Dimension NMAX x NV.

Diese Funktion liefert als Ergebnis die rechte und die linke Eigenvektormatrix sowie einen Vektor mit den Quadratwurzeln der Eigenwerte. Ein bedeutsamer Vorteil ist, dass diese bereits nach dem Betrag der Singulärwerte sortiert sind. So kann direkt eine Trennung von Signal- und Rauschunterraum vorgenommen werden.
Im vorliegenden Fall wird immer von einer Quelle ausgegangen, sodass für die Berechnung des MUSIC-Spektrum lediglich der erste Eigenvektor von u entfernt werden muss.
Zwar ist diese Methode durch die Verwendung von komplexwertigen Datentypen und durch die Sortierung sehr komfortabel, dennoch ist es schwer abzuschätzen, ob diese auch tatsächlich effektiv arbeitet. Aus diesem Grund wurde eine zweite Variante herangezogen, welche auf einer reellwertigen sogenannten thin SVD basiert. Dabei handelt es sich um eine Singulärwertzerlegung, bei der nur Teile der Matrix $\boldsymbol{U}$ berechnet werden, sodass diese die Dimension M x N aufweist. Dabei gilt M > N [28].

$$\boldsymbol{A} = \boldsymbol{U}_1 \boldsymbol{W}_1 \boldsymbol{V} \tag{5.19}$$

mit

$$\boldsymbol{U}_1 = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n] \tag{5.20}$$

und

$$\boldsymbol{W}_1 = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n] \tag{5.21}$$

Das bedeutet, dass hier lediglich die ersten N Spalten der Matrix $\boldsymbol{U}$ berechnet werden. Diese hat in diesem Fall die Dimension M x N. Außerdem werden lediglich die ersten N Singulärwerte berechnet. Die Matrix $\boldsymbol{V}$ unterscheidet sich nicht zu der aus einer normalen SVD. Die gewünschten Eigenvektoren befinden sich in der Matrix $\boldsymbol{U}$, sodass eine thin SVD einen Informationsverlust bedeuten würde. Wird allerdings der in [8, S. 52f.] beschriebene Zusammenhang ausgenutzt, lässt sich für die weitere Berechnung die Matrix $\boldsymbol{V}$ anstelle der Matrix $\boldsymbol{U}$ verwenden. Sei $\boldsymbol{A}$ eine Matrix der Dimension N x M und $\boldsymbol{B}$ die adjungierte von $\boldsymbol{A}$ mit Dimension M x N, so gilt:

$$\begin{aligned} \boldsymbol{B} = \boldsymbol{A}^H &= \left(\boldsymbol{U}_A \boldsymbol{W}_A \boldsymbol{V}_A\right)^H = \left(\boldsymbol{W}_A \boldsymbol{V}_A^H\right) \boldsymbol{U}_A^H \\ &= \boldsymbol{V}_A \boldsymbol{W}_A^H \boldsymbol{U}_A^H \end{aligned} \tag{5.22}$$

Gleichermaßen gilt:

$$\boldsymbol{B} = \boldsymbol{U}_B \boldsymbol{W}_B \boldsymbol{V}_B^H \tag{5.23}$$

und

$$\boldsymbol{W}_A = \boldsymbol{W}_A^H \tag{5.24}$$

Ein Vergleich von Gleichung 5.22 mit Gleichung 5.23 und Gleichung 5.24 führt zu dem Schluss, dass die Matrix $\boldsymbol{V}_B$ genau der Matrix $\boldsymbol{U}_A$ entspricht. Falls lediglich die Matrix $\boldsymbol{U}_A$ sowie die Singulärwerte von Interesse sind, können diese mit der thin SVD ermittelt werden, indem die Matrix $\boldsymbol{A}$ zunächst adjungiert wird.
Eine weitere Einschränkung, die bei der Verwendung einer reellwertigen thin SVD zum Tragen kommt, ist die Tatsache, dass in der vorliegen Arbeit komplexwertige Daten verwendet werden. Eine Lösung, mit der das Problem in eine Berechnung mit reellwertigen Matri-

zen überführt wird, findet sich in [12]. Demnach lässt sich eine komplexwertige Matrix $\boldsymbol{A}$ folgendermaßen ausdrücken:

$$\boldsymbol{A} = \boldsymbol{A}_{re} + j\boldsymbol{A}_{im} \tag{5.25}$$

Wenn für die Matrix $\boldsymbol{A}$ der Dimension M x N die Gleichung 5.16 gilt, so gilt für eine neue Matix der Dimension 2M x 2N:

$$\begin{pmatrix} \boldsymbol{A}_{re} & -\boldsymbol{A}_{im} \\ \boldsymbol{A}_{im} & \boldsymbol{A}_{re} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}_{re} & -\boldsymbol{U}_{im} \\ \boldsymbol{U}_{im} & \boldsymbol{U}_{re} \end{pmatrix} \begin{pmatrix} \boldsymbol{W} & 0 \\ 0 & \boldsymbol{W} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}_{re} & -\boldsymbol{V}_{im} \\ \boldsymbol{V}_{im} & \boldsymbol{V}_{re} \end{pmatrix}^T \tag{5.26}$$

So lässt sich die in [33, S. 70ff.] vorgestellte Funktion *svdcmp* verwenden. Die Übergabeparameter lauten wie folgt:

float u Matrix der Dimension M+1 x N+1, welche die Eingangsdaten beinhaltet. Diese werden mit den Werten der Matrix $\boldsymbol{U}$ überschrieben. Die Dimensionen müssen jeweils um eins erweitert sein, da die Funktion nicht am Index Null, sondern am Index Eins startet.

float v Matrix der Dimension N+1 x N+1, in welche die Werte der Matrix $\boldsymbol{V}$ hinterlegt werden.

float w Vektor der Dimension N, in welchen die Singulärwerte hinterlegt werden.

int m Anzahl der Zeilen.

int n Anzahl der Spalten.

Zusammenfassend muss somit für die Verwendung der reellwertigen thin SVD zunächst die transponierte der Eingangsmatrix gebildet werden. Diese muss wie in Gleichung 5.26 in eine realwertige Matrix der Dimension 2M x 2N bzw. für die konkrete Implementierung in eine Matrix der Dimension 2M+1 x 2N+1 überführt werden, welche nun mittels thin SVD zerlegt wird. Die hier ermittelte Matrix $\boldsymbol{V}$ ist realwertig. Sie enthält jedoch alle Einträge, die notwendig sind, um daraus die gesuchte komplexwertige Matrix zu gewinnen. Genau wie der Vektor mit den Singulärwerten. Die zusammengesetzte Matrix $\boldsymbol{V}$ mit:

$$\boldsymbol{V} = \boldsymbol{V}_{Re} + \boldsymbol{V}_{Im} \tag{5.27}$$

enthält die gesuchten Eigenvektoren. Die hier verwendete Implementierung hat gegenüber der zuvor beschriebenen Implementierung, welche mit komplexwertigen Datentypen arbeitet den Nachteil, dass die Eigenwerte und Eigenvektoren nicht in sortierter Form ermittelt werden. Daraus entsteht die Notwendigkeit eine nachträgliche Sortierung vorzunehmen. Hierfür kommt der Bubblesort-Algorithmus zum Einsatz. Sortiert wird das Array mit den Singulärwerten. Gleichzeitig wird ein Array erstellt, in welchem die vorherigen Indexe der Elemente genau so umsortiert wurden, wie die Elemente selber. Anhand dieses Arrays werden anschließend die Eigenvektoren in der Eigenvektormatrix umsortiert. Diese Sortierungen werden von den Funktionen *sort* und *get_noise_real_SVD* durchgeführt. Letztere übernimmt gleichzeitig die Rücktransformation von einer reellwertigen zu einer komplexwertigen Matrix.
Ziel ist es das MUSIC-Spektrum für den Breitbandfall zu ermitteln. Dieses lässt sich folgendermaßen beschreiben:

$$S_{MUSIC}(\phi) = \frac{1}{\sum_{i=1}^{L} \boldsymbol{\varsigma}^H(\tau(\phi_i)) \cdot \boldsymbol{P} \cdot \boldsymbol{\varsigma}(\tau(\phi_i))} \tag{5.28}$$

mit:

$$\boldsymbol{P} = \boldsymbol{Q}_n \cdot \boldsymbol{Q}_n^H \tag{5.29}$$

Um die Matrix $\boldsymbol{P}$ zu erhalten, muss nun aus den sortierten Eigenwertmatrizen, welche durch die verschiedenartigen Implementierungen der Singulärwertzerlegung zur Verfügung stehen, das Produkt des Rauschunterraums mit dem adjungierten Rauschunterraum gebildet werden. Die Matrix $\boldsymbol{P}$ wird dann von links und von rechts mit sämtlichen Steering-Vektoren multipliziert. Dies geschieht für alle vier Frequenzen, wobei für jede Frequenz eine neue Signal-Matrix aus dem Spektrum der Mikrofonsignale für die jeweilige Frequenz vorbereitet und anschließend eine Singulärwertzerlegung vorgenommen wird. Nach einer ggf. notwendigen Sortierung folgt die Trennung von Rausch- und Signalunterraum sowie die gleichzeitige Berechnung der Matrix $\boldsymbol{P}$ mithilfe der Funktion *transjun_mult*, welche die adjungierte Multiplikation mit einem Spalten-Offset durchführt. Dieser kann hier frei gewählt werden, sodass es möglich ist, beliebig viele zum Signalunterraum gehörigen Eigenvektoren bei dieser Multiplikation zu vernachlässigen. Da hier konstant von nur einer Signalquelle ausgegangen wird, ist dieser Parameter immer eins. Falls bei späteren Erwei-

terungen mehrere Quellen berücksichtigt werden sollen, kann dies geschehen, indem zuvor anhand des Betrags der Eigen- bzw. Singulärwerte die Anzahl der Quellen ermittelt und dieser Wert als Parameter übergeben wird.

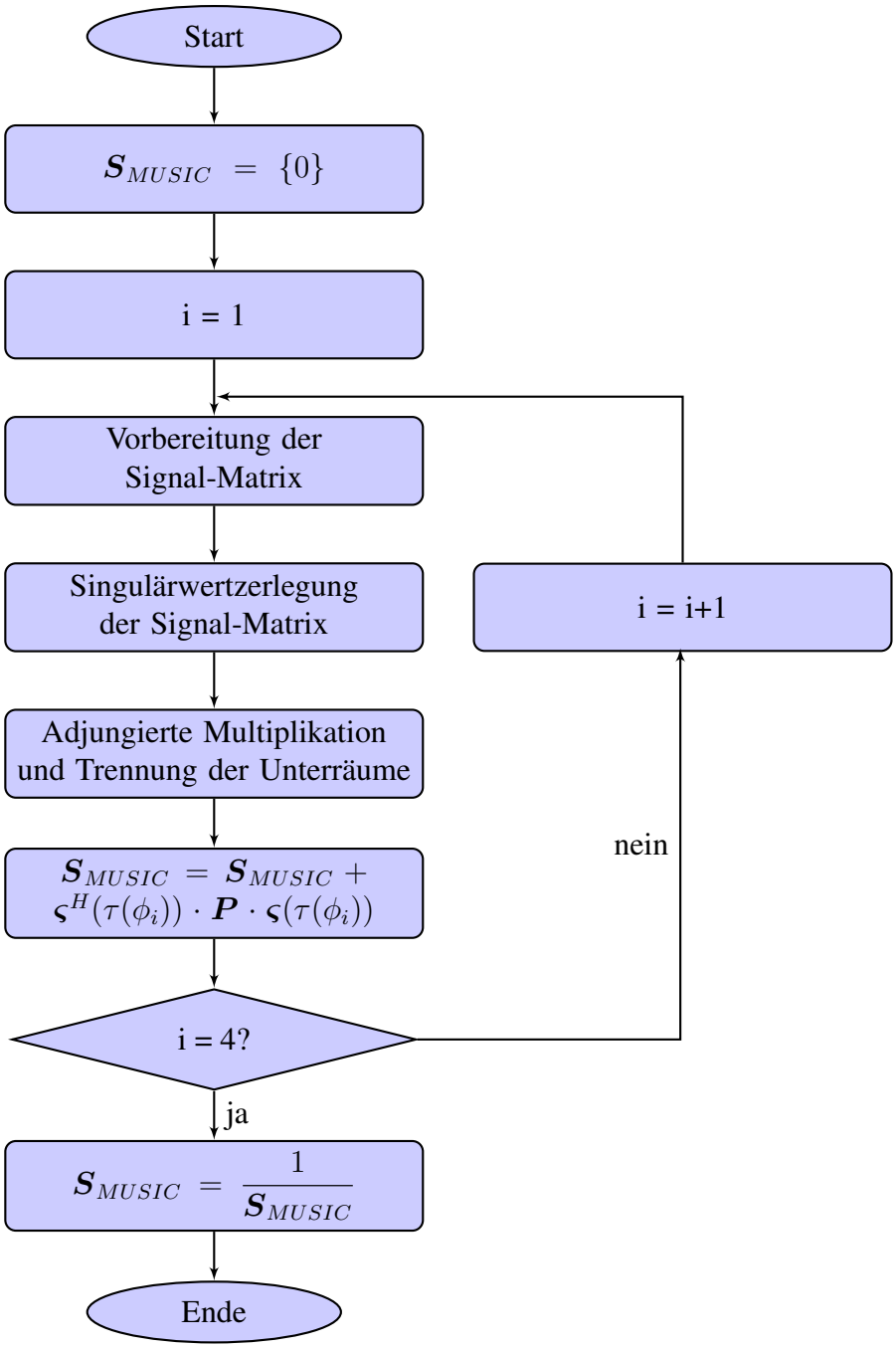


Abbildung 5.5.: Ablauf Breitband-MUSIC

Bei der Berechnung von

$$\boldsymbol{S}_{MUSIC} = \boldsymbol{S}_{MUSIC} + \boldsymbol{\varsigma}^{H}(\tau(\phi_i)) \cdot \boldsymbol{P} \cdot \boldsymbol{\varsigma}(\tau(\phi_i)) \tag{5.30}$$

muss unterschieden werden, ob es sich um feste oder variable Frequenzstützstellen handelt. Sollte es sich um feste handeln, werden die Steering-Vektoren für sämtliche Stützstellen vor der Laufzeit berechnet und je nach Winkelauflösung in jeweils einem eigenen entsprechend großen zweidimensionalen Array hinterlegt. Dafür wurde die Funktion *init_fix_steering* geschrieben, welche als Übergabeparameter einen Zeiger zu dem zuvor beschriebenen Array und den Wert der Frequenz benötigt. Mithilfe des definierten Wertes *RESOLUTION_MUSIC* werden anschließend alle komplexen Werte der Steering-Vektoren berechnet. Zur Laufzeit wird in diesem Fall die Funktion *add_NULL_SPECTRUM_fix_steering* ausgeführt. Diese benötigt als Übergabeparameter einen Zeiger auf das Array, in welchem das MUSIC-Spektrum hinterlegt werden soll, sowie einen Zeiger zu dem zweidimensionalen Array, in dem das Produkt aus Rauschunterraum und adjungierten Rauschunterraum hinterlegt ist, und außerdem einen Zeiger auf das Array, in welchem die Steering-Vektoren hinterlegt sind. In der Funktion wird nun für jeden möglichen Winkel ϕ die Funktion *calc_S_inv_fix_steering* aufgerufen. In dieser wird genau ein Wert des MUSIC-Spektrum für einen Winkel berechnet. Dies geschieht, indem der zu dem Winkel gehörige Steering-Vektor zunächst mit der Matrix $\boldsymbol{P}$ multipliziert wird. Anschließend wird das Ergebnis dieser Multiplikation noch einmal mit dem adjungierten Steering-Vektor multipliziert.

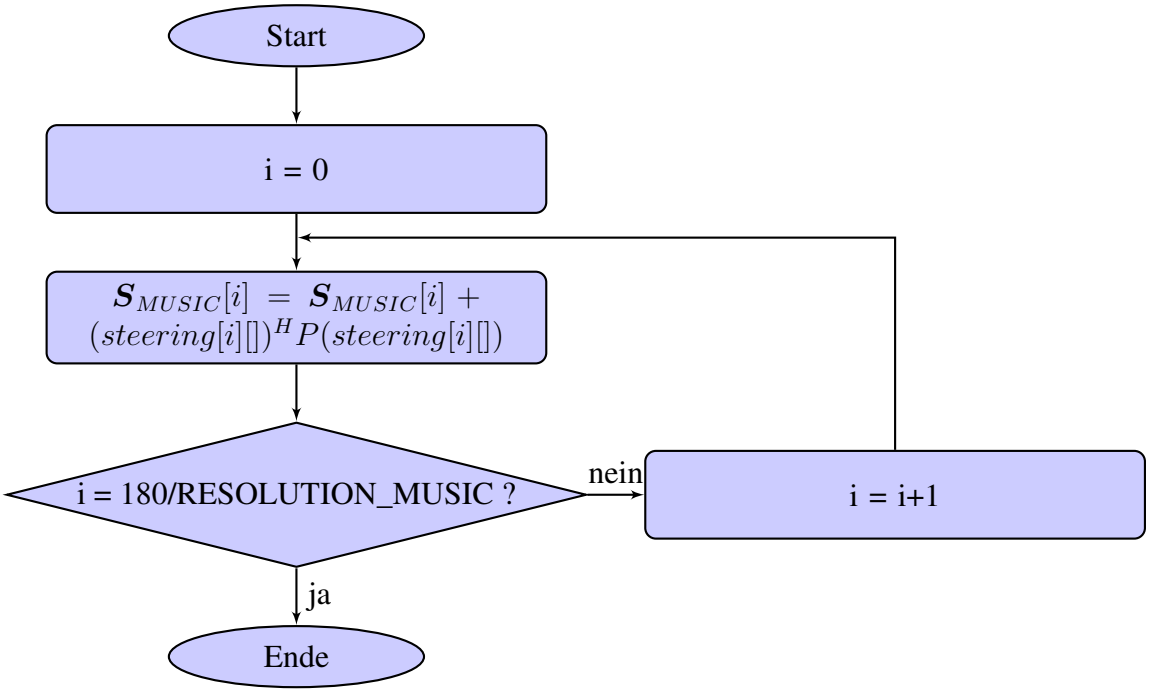


Abbildung 5.6.: Berechnung und Addition des reziproken MUSIC-Spektrums für eine feste Frequenz

Sollen die Frequenzen, welche zur Bestimmung des MUSIC-Spektrum herangezogen werden, zur Laufzeit verändert werden, so ist eine vorherige Berechnung aller Steering-Vektoren nicht notwendig. Es bestünde zwar die Möglichkeit diese jedes Mal nach der Bestimmung der Frequenzstützstellen mit der zuvor vorgestellten Funktion *init_fix_steering* zu berechnen. Der übrige Programmablauf würde sich nicht von der zuvor vorgestellten Variante unterscheiden und es könnten dieselben Funktionen verwendet werden. Dabei ergibt sich gegenüber der tatsächlich implementierten Variante jedoch ein entscheidender Nachteil. Da jedes Mal neue Steering-Vektoren berechnet werden müssen, ist es nicht entscheidend zu welchem Zeitpunkt im Programmablauf dies geschieht. Somit ist es ausreichend immer nur genau den Steering-Vektor zu berechnen, der für die aktuelle Berechnung benötigt wird. Für N = 8 Mikrofone bedeutet ein einzelner Steering-Vektor einen Speicherbedarf von 8 komplexen Variablen vom Typ *float* jeweils mit einem Real- und einem Imaginärteil, welche zusammen einen Speicherplatz von 64 Byte belegen. Bei einer Auflösung von einem Grad und der Verwendung von vier Frequenzstützstellen resultiert eine 720-fache Speicherplatzbelegung, wenn alle Steering-Vektoren vorberechnet werden.

Diese Ersparnis rechtfertigt einen zusätzlichen Programmieraufwand. Zur Umsetzung wurden zwei Funktionen geschrieben, welche starke Ähnlichkeit mit den Varianten für feste Steering-Vektoren haben. Die Funktion *add_NULL_SPECTRUM* unterscheidet sich von der Funktion *add_NULL_SPECTRUM_fix_steering* darin, dass anstelle eines Zeigers zu einem Array mit Steering-Vektoren ein Wert für eine Frequenzstützstelle übergeben wird. Außerdem benutzt diese Funktion anstelle von *calc_S_inv_fix_steering* die Funktion *calc_S_inv*. Auch hier unterscheiden sich die Übergabeparameter. Anstelle des Zeigers zu einem Steering-Verktor wird auch hier der Frequenzwert der Frequenzstützstelle übergeben. Der jeweils benötigte Steering-Vektor wird in dieser Funktion berechnet.

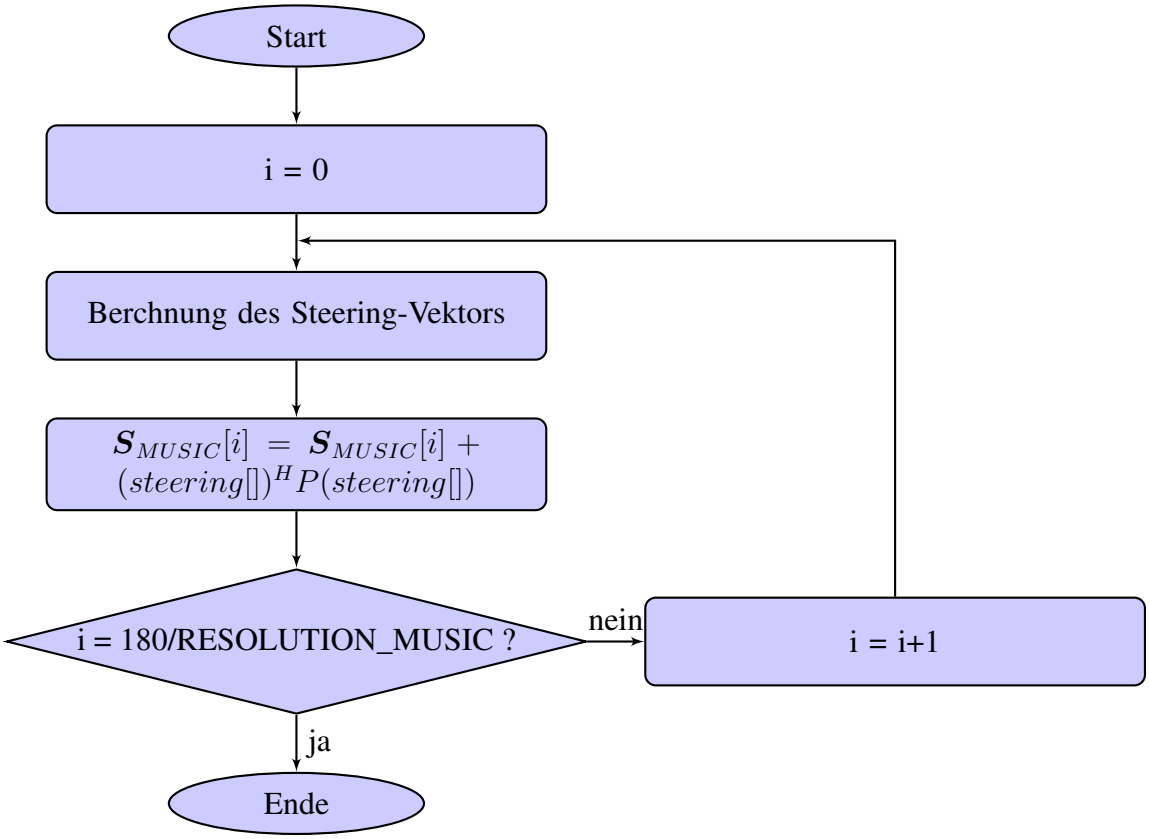


Abbildung 5.7.: Berechnung und Addition des reziproken MUSIC-Spektrums für eine variable Frequenz

Nachdem das reziproke MUSIC-Spektrum als Summe der reziproken MUSIC-Spektren für alle vier Frequenzstützstellen ermittelt wurde, müssen die Werte elementweise invertiert werden, um so das MUSIC-Spektrum zu erhalten.

5.2.3. Maxima-Suche im MUSIC-Spektrum

Die gesuchten Winkel befinden sich bekanntermaßen in den Maxima des MUSIC-Spektrum. Hier wurde - wie bereits zuvor beschrieben - von nur einer Quelle ausgegangen, sodass die Suche nach dem größten Wert im Array, in welchem das MUSIC-Spektrum hinterlegt ist, ausreichend wäre. Da allerdings Wert darauf gelegt wurde, bei künftigen Erweiterungen möglichst wenig ändern zu müssen, wurde auch an dieser Stelle eine Funktion entwickelt. Diese ist in der Lage mehrere Maxima in dem MUSIC-Spektrum zu finden, diese zu sortieren und die korrespondierenden DOAs zu ermitteln. Die Funktion *get_angles* sucht zunächst nach allen Maxima in einem Array, indem Stellen gesucht werden, an denen die Werte links und rechts daneben jeweils kleiner sind. Werden solche Stellen gefunden, wird der Wert und der zugehörige Index jeweils in einem dafür vorgesehenen Array hinterlegt. Wurden mehrere Maxima gefunden, werden diese der Größe nach sortiert. Das Array mit den korrespondierenden Indexen wird anhand der neuen Anordnung der Maxima ebenfalls umsortiert. Nun befinden sich die Maxima in absteigender Reihenfolge sortiert in dem dafür vorgesehenem Array. Anhand des zugehörigen Index lässt sich leicht ein korrespondierender Winkel ermitteln, indem diese lediglich mit dem Betrag der Auflösung multipliziert werden. Beträgt die Auflösung beispielsweise 0,5° und befindet sich das größte Maximum am Index 20 so wäre der zugehörige Winkel:

$$20 \cdot 0,5^\circ - 90^\circ = -80^\circ \tag{5.31}$$

Die Subtraktion von 90° ist notwendig, da hier eine Winkelangabe von -90° bis 90° gewünscht ist. Da ein Array nicht mit negativen Zahlen indiziert werden kann, muss hier mit einem Offset gearbeitet werden. Im letzten Schritt bei der Winkelbestimmung ist dieser wieder zu entfernen.

5.3. Implementierung des gesamten Programms

Als Projekt-Vorlage wurde das Projekt *pcm3003_DM2c6657* verwendet, welches von dem Hersteller der verwendeten Hardware zur Verfügung gestellt wird. Diese verwendet zwei McBSP-Interrupts, um alle acht Kanäle des PCM3003-Moduls zu erfassen. Die folgende Abbildung zeigt die Kommunikation zwischen DSP und PCM3003-Board.

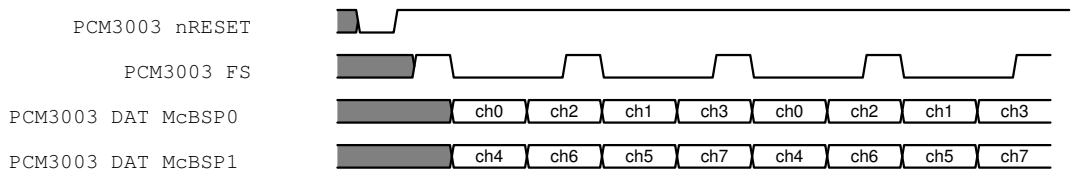


Abbildung 5.8.: Kommunikation zwischen DSP und PCM3003-Board

Das mitgelieferte Programm ist für eine samplebasierte Verarbeitung vorgesehen, sodass es für eine blockweise Verarbeitung angepasst werden musste.
Ziel war es beide Algorithmen in einem Programm zu implementieren, welches so gestaltet werden sollte, dass die verwendeten Algorithmen zur Laufzeit ausgewählt und auch andere Parameter verändert werden können. Dafür ist zunächst eine Schnittstelle notwendig, worüber der DSP kommunizieren kann. Hier wurde - wie in Abschnitt 4 beschrieben - die serielle Schnittstelle gewählt. Da davon auszugehen ist, dass sehr selten Daten empfangen werden, diese allerdings in mehreren direkt aufeinanderfolgenden Bytes empfangen werden, wurde hier auf einen Interrupt zurückgegriffen. Dieser muss zunächst konfiguriert werden. Dafür sind folgende Schritte notwendig:

1. Interrupt am entsprechenden Chip Interrupt Controller (CIC) aktivieren
2. Interrupt auf einen Channel mappen
3. Host-Interrupt für den Channel aktivieren
4. Handle für das Event öffnen und zu einem Interrupt-Vektor zuordnen
5. Interrupt-Vektor zu der entsprechenden Service-Routine zuordnen
6. Event aktivieren

Zusätzlich ist es notwendig Interrupts global zu aktivieren, sofern dies nicht bereits im Zuge der Konfiguration eines anderen Interrupts geschehen ist. Nun wird bei jedem über die serielle Schnittstelle empfangenden Byte die ISR *eventUartHandler* aufgerufen. Die empfangenden Bytes werden in ein Array geschrieben und ein Laufindex erhöht. Dieser wird bei jedem Durchlauf der Programmschleife geprüft. Sobald dieser der Länge einer Konfigurationsnachricht entspricht, werden die entsprechenden Konfigurationen vorgenommen. Dafür wird zunächst geprüft, ob es sich bei den ersten beiden Bytes jeweils um den Wert ”0xff” und den Wert ”0x00” handelt. Ist das nicht der Fall, wird die Nachricht verworfen. Zum

Hinterlegen der Konfigurationen wurde eine Struktur definiert, in welcher die verschiedenen Parameter hinterlegt sind. Die Einstellung der Abtastrate erfolgt - wie zuvor beschrieben - über GPIO-Pins, welche so verschaltet sind, dass der Wert aus der Konfigurationsnachricht direkt für den GPIO-Port übernommen werden kann.
Das Programm prüft in der Endlosschleife, nachdem ggf. Konfigurationen übernommen wurden, ob ein neuer Block vorliegt. Ist dies der Fall, wird im nächsten Schritt die Energie des Blocks vom ersten Mikrofon berechnet. Anhand eines Schwellwertes wird so geprüft, ob es sich um ein Sprachsignal handelt. Sollte dies nicht der Fall sein, wird der Block verworfen und die Algorithmen werden nicht ausgeführt. Andernfalls werden zunächst die Signale aus den jeweiligen Arrays vom Typ *short* in Array vom Typ *float* kopiert und entsprechend umgerechnet. Diese Arrays haben die zweifache Länge und werden mit Nullen aufgefüllt. Weiterhin wird geprüft, ob der MCCC als aktueller Algorithmus gewählt ist. Sollte das der Fall sein, wird die jeweilige Varianz der Signale ebenfalls berechnet.
Im nächsten Schritt wird unabhängig von der Wahl des zu benutzenden Algorithmus eine FFT von allen Signalen durchgeführt. Dies ist möglich, da das Signal-Array zuvor mit Nullen gefüllt wurde, was im konkreten Fall einem zweifachen Zeropadding entspricht. Somit ist es möglich, die Ergebnisse der FFT zur Berechnung der KKFs zu benutzen. Anschließend wird geprüft, ob es sich bei dem zu verwendenden Algorithmus um MUSIC handelt. Ist dies der Fall, wird weiter geprüft, ob die SVD für reelle oder komplexe Matrizen ausgeführt werden soll. Nach der SVD erfolgt die Berechnung und Addition des reziproken MUSIC-Spektrum entweder für feste oder für variable Steering-Vektoren. Diese beiden Schritte erfolgen für jede Frequenzstützstelle. Zuletzt wird - wie bereits beschrieben - der entsprechende Winkel aus dem MUSIC-Spektrum extrahiert. Sollte der MCCC als Algorithmus gewählt sein, werden zunächst alle KKFs berechnet. Anschließend wird für alle Werte für p die Korrelationsmatrix erstellt und deren Determinante berechnet. Zuletzt wird anhand der Werte der Determinanten ein Winkel bestimmt. Um temporäre Ausreißer zu unterdrücken, wird ein Medianfilter bei beiden Verfahren eingesetzt. Dafür wird der Winkel jeweils in ein Array an die Stelle geschrieben, an der sich der älteste Wert befindet. Dieses Array wird zunächst kopiert und die Kopie anschließend mit Bubblesort sortiert. Da das Array eine ungerade Anzahl an Elementen hat, befindet sich der Median genau im mittleren Element des Arrays, welches im konkreten Fall den Index vier hat. Dadurch, dass jeweils die neun aktuellsten Werte herangezogen werden, ergibt sich ein gleitender Median. Zuletzt wird das jeweilige Ergebnis über die serielle Schnittstelle ausgegeben.

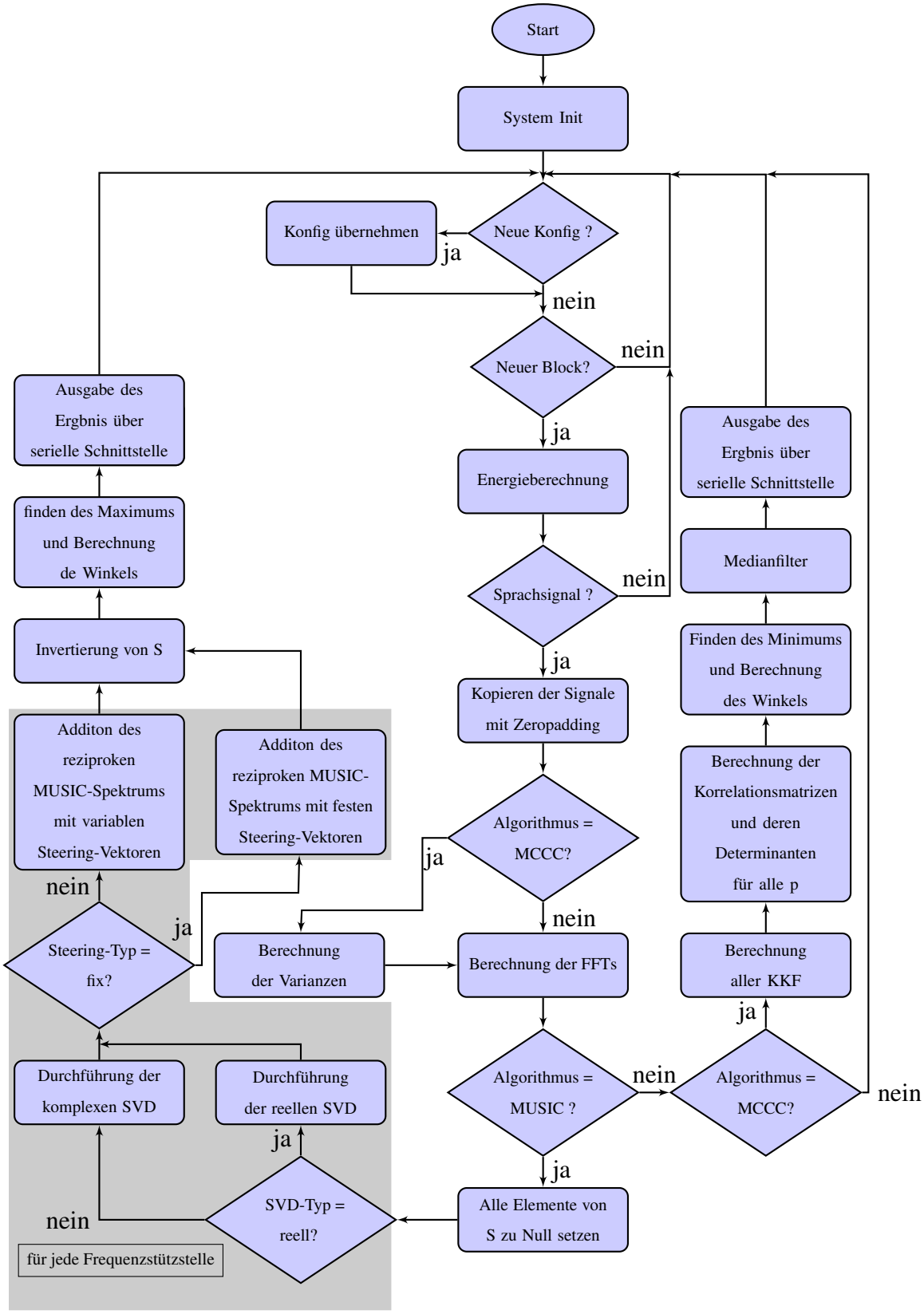


Abbildung 5.9.: Ablauf des Hauptprogramms

6. Profiling der Algorithmen

Eine häufige Anforderung an Algorithmen ist ihre Echtzeitfähigkeit. Für diesen Begriff gibt es viele Definitionen, die nicht immer einheitlich sind. Ernst und Schmidt [18, S. 301] beschreiben Echtzeitfähigkeit als: „Die Fähigkeit, innerhalb definierter zeitlicher Grenzen auf Ereignisse von außen zu reagieren“. Im Bezug auf die Sprecherlokalisierung würde dies bedeuten, dass sobald ein Block neuer Samples vorhanden ist, der Winkel ausreichend schnell als gesuchte Größe berechnet werden muss. Die verlangte Geschwindigkeit hängt dabei stark von der Anwendung ab. Geht es zum Beispiel darum, eine statische Quelle einmalig zu orten, um diese bei der Auslegung eines Beamformers zu berücksichtigen, ist die benötigte Rechenzeit nicht so entscheidend, wie bei der Verfolgung einer sehr dynamischen Quelle. Aufgrund dieser Tatsache lässt sich der Begriff Echtzeitfähigkeit hier nicht konkret beziffern. Jedoch kann festgehalten werden, dass ein Algorithmus potenziell besser für Echtzeitanwendungen geeignet ist, je schneller er ein Ergebnis liefert, sofern die Geschwindigkeit nicht durch Fehler erkauft wird.
Ein weiterer wichtiger Gesichtspunkt, wenn es um Ressourcen geht, ist der benötigte Speicher. In der Signalverarbeitung sowie in diversen anderen Bereichen kommen häufig eingebettete Systeme zum Einsatz, welche mit Prozessoren arbeiten, die nur über wenig internen Speicher verfügen. Einige Systeme verwenden zusätzlichen externen Speicher, welcher in der Regel Nachteile durch längere Zugriffszeiten mit sich bringt.
Die Analyse des Ressourcenbedarfs lässt sich theoretisch zum Beispiel mit der O-Notation für die Rechenzeit zumindest grob anhand von Klassen abschätzen. Hier soll dies jedoch anhand von Messungen an den jeweils auf einem DSP implementierten Algorithmen geschehen.

6.1. Beschreibung der Testumgebung

Um nicht an Restriktionen durch die Notwendigkeit von externen Speicher zu stoßen, was evtl. Einfluss auf die Vergleichbarkeit haben könnte, wurde ein DSP-Plattform der Firma D.SignT mit einem sehr leistungsfähigem DSP vom Typ TMS320C6657 von Texas Instruments verwendet. Der Speicherplatzbedarf lässt sich grob anhand der angelegten Variablen abschätzen.
Es gibt sehr viele Parameter, die für diese Messungen variiert werden können. Eine davon ist die Anzahl der verwendeten Mikrofone. Da das System die Möglichkeit bietet acht Mikrofone zu verwenden, soll auch immer auf alle acht zurückgegriffen werden.

6.2. Messung der Rechenzeiten für verschiedene in den Algorithmen verwendete Funktionen

Es gibt verschiedene Möglichkeiten zum Messen der Laufzeit von Funktionen, welche im Folgenden kurz dargestellt werden sollen.

Messung mit Oszilloskop und GPIO-Pin Der hier verwendete DSP verfügt über einige GPIO-Pins, welche für Messung benutzt werden können. Dafür wird ein freier GPIO-Pin vor dem Funktionsaufruf gesetzt und nach diesem wieder zurückgesetzt. Mit dem Oszilloskop lässt sich der Pegel des Pins über der Zeit darstellen. Die Zeit, in der der Pin auf High war, entspricht der Zeit, die für den Aufruf und die Ausführung der Funktion benötigt wurde. Allerdings beinhaltet sie auch die Zeiten, welche zum setzen und zurücksetzen des GPIO-Pins benötigt wurden.

Messung mit internem Timer Weitere Peripherie-Bausteine des DSPs sind die integrierten Timer. Diese laufen eigenständig und lassen sich über Register auslesen bzw. konfigurieren. Somit ist es möglich die Laufzeit einer Funktion zu messen, indem vor und nach deren Aufruf der Wert eines Timers ausgelesen und die jeweiligen Werte subtrahiert werden. Der Fehler durch die Zugriffszeit lässt sich ebenfalls leicht bestimmen, indem der Timerwert zwei Mal direkt hintereinander ausgelesen wird. Die Differenz ist der systematische Fehler durch die Zugriffszeit.

Profiling mit Code Composer Studio In der Entwicklungsumgebung von Texas Instruments gibt es die Möglichkeit die Taktzyklen zwischen zwei Haltepunkten während des Debuggings zu bestimmen.

Zunächst wurden alle Methoden exemplarisch getestet, um eine Methode für das weitere Profiling zu wählen. Die Messung mit internem Timer hat sich als die komfortabelste herausgestellt. Bei der Messung mit GPIO-Pin und Oszilloskop ist es schwer, ein genaues Ergebnis zu erhalten, da neben dem systematischen Fehler durch das Setzen und Zurücksetzen der Pins auch noch Ablesefehler durch die händische Einstellung der Cursor hinzukommen. Die Nutzung der Profiler-Funktion von Code Composer Studio war ebenfalls nicht zufriedenstellend, da hierfür davon ausgegangen werden muss, dass Haltepunkte im C-Code auch tatsächlich an der gewünschten Stelle greifen. Gerade bei optimiertem Code ist dies nicht immer der Fall.
Ein zusätzlicher Aspekt, der unbedingt beachtet werden muss, ist das Auftreten von Interrupts. Tritt während der Messung ein Interrupt ein, so unterbricht dieser die normale Programmausführung. Die Zeit, die die Interrupt-Service-Routine für ihre Ausführung benötigt, sowie alle Schritte, die zu deren Aufruf und zum Rücksprung in das Programm benötigt werden, werden mit gemessen. Daher ist es unbedingt notwendig alle Interrupts global zu deaktivieren.
Des Weiteren muss sichergestellt sein, dass der Timer während der Messung nicht überläuft. Dies würde ebenfalls das Ergebnis drastisch verfälschen. Für die Messungen wurde ein 32 Bit Timer verwendet, welcher mit einem Sechstel des Systemtakts von 1,25 GHz getaktet wird. Es ergibt sich für die Zeit bis zu einem Überlauf:

$$t_{Overflow} = \frac{2^{Bit}}{f_{Clk}/6} = \frac{2^{32}}{1,25 \cdot 10^9 1/s \cdot 1/6} = 2,06s \tag{6.1}$$

Ein Überlauf lässt sich in diesem Fall leicht daran erkennen, dass eine negative Anzahl von Taktzyklen ermittelt wurde. Das liegt daran, dass der Timer nach einem Überlauf wieder von vorne beginnt zu zählen. Der zweite ausgelesene Wert wäre in dem Fall kleiner als der Erste und es ergibt sich eine negative Differenz. Dies stimmt solange die Laufzeit der Funktionen unter 2,06 Sekunden ist, da diese Zeit benötigt wird, bis der Timer wieder den zuerst ausgelesenen Wert annimmt. Da davon ausgegangen werden kann, dass keine der implementierten Funktion Laufzeiten in dieser Größenordnung aufweist, genügt es Werte mit

negativen Vorzeichen als ungültig zu deklarieren. Zusätzlich wird durch mehrere Messungen, die identische Ergebnisse liefern müssen, ein solcher Fehler ausgeschlossen.

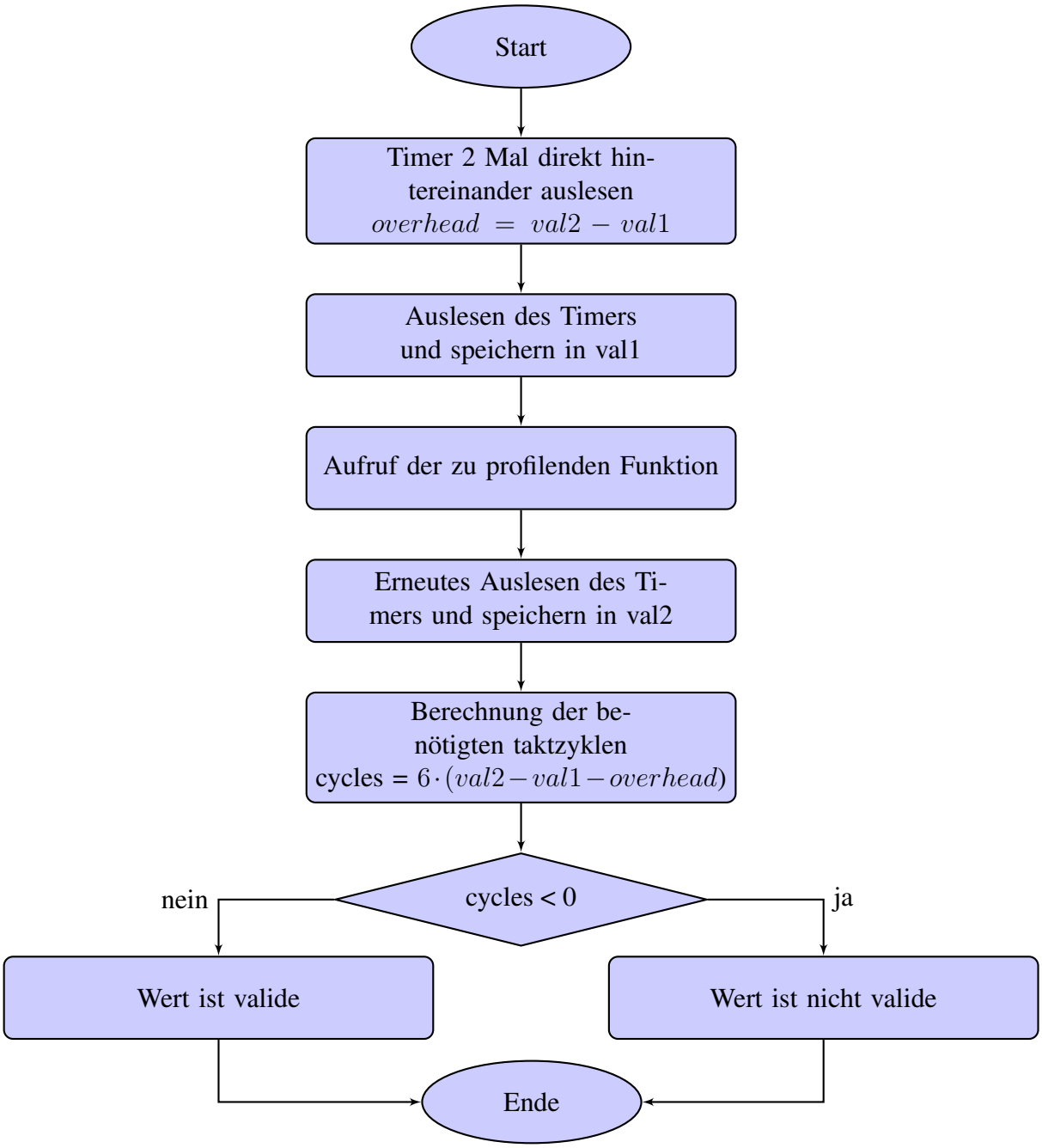


Abbildung 6.1.: Ablauf der Zeitmessung für einen Funktionsaufruf

Im Folgenden sollen die Ergebnisse der Messungen aufgezeigt und Funktionen zur Berechnung der notwendigen Taktzyklen für die jeweiligen Algorithmen in Abhängigkeit der verschiedenen Parameter aufgestellt werden. Dafür wurden die Messungen für verschiedene Werte der jeweiligen Parameter wiederholt, um aus den Datenpunkten mithilfe von der MATLAB Curve Fittig Toolbox eine möglichst gut passende zugrunde liegende Funktion zu ermitteln. Der gesamte Aufwand eines Algorithmus ergibt sich aus der Summe aller Funktionen, die während eines Durchlaufs ausgeführt werden. Falls eine Funktion während eines Durchlaufs des jeweiligen Algorithmus mehrfach aufgerufen wird, muss dies ebenfalls berücksichtigt werden.
Zunächst wird der Aufwand der Funktion *DSPF_sp_fftSPxSP* bestimmt. Diese wird für beide Algorithmen benötigt und je Durchlauf acht Mal ausgeführt. Der Aufwand hängt von der Länge der FFT ab. Diese muss bei dem MCCC-Algorithmus der doppelten Blocklänge entsprechen.

Tabelle 6.1.: Profiling der Funktion *DSPF_sp_fftSPxSP*

FFT-Länge	Optimierung	Aufrufe/Durchlauf	Taktzyklen
128	o3-3	8	936
256	o3-3	8	1818
512	o3-3	8	4356
1024	o3-3	8	8490
1024	/	8	8592

Mithilfe der Curve Fittig Toolbox wurde als mögliche zugrunde liegende Funktion in Abhängigkeit der Blocklänge mit

$$f(K) = 1,188 \cdot K \cdot log(K) + 263,1 \tag{6.2}$$

bestimmt.

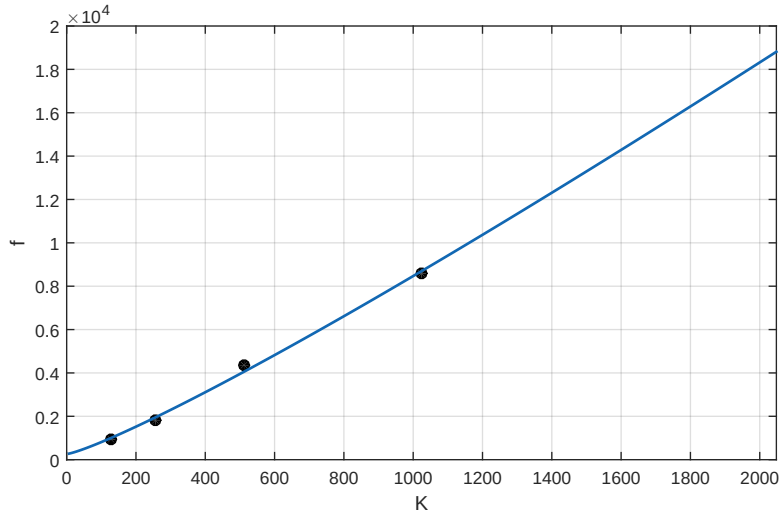


Abbildung 6.2.: Messpunkte des Profilings für die Funktion *DSPF_sp_fftSPxSP* mit Verlauf der möglichen zugrunde liegenden Funktion

Die Funktion *cacl_var* wird für den MCCC-Algorithmus benötigt und wird ebenfalls für jedes Mikrofon ein Mal pro Durchlauf aufgerufen. Da hier im Gegensatz zu der FFT im Vorfeld keine mögliche zugrunde liegende Funktion bekannt ist, muss diese anhand des Programmablaufs geschätzt und dies mit dem Curve Fitting Tool überprüft werden. Wie vermutet, passt hier eine lineare Abhängigkeit sehr gut.

Tabelle 6.2.: Profiling der Funktion *calc_var*

FFT-Länge	Optimierung	Aufrufe/Durchlauf	Taktzyklen
128	o3-3	8	294
256	o3-3	8	540
512	o3-3	8	1038
1024	o3-3	8	2028
1024	/	8	13350

$$f(K) = 1,936 \cdot K + 45,69 \tag{6.3}$$

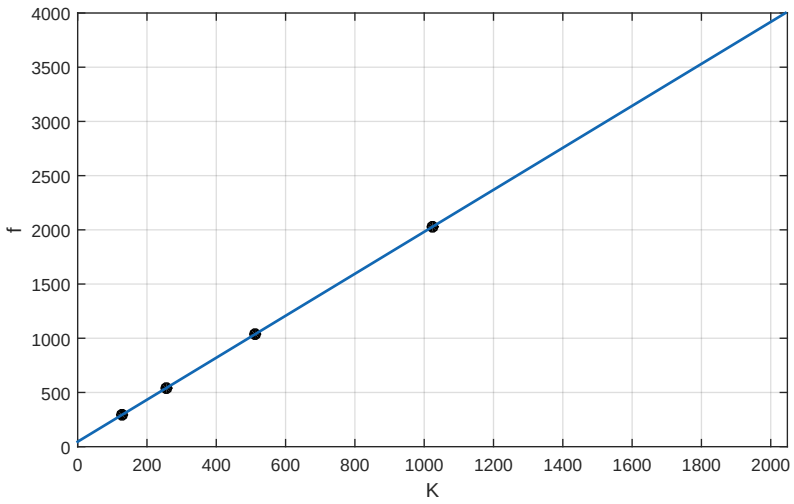


Abbildung 6.3.: Messpunkte des Profilings für die Funktion *calc_var* mit Verlauf der möglichen zugrunde liegenden Funktion

Die Funktion Correlation_except_var enthält N IFFTs sowie $\frac{K \cdot (N^2 - N)}{2}$ konjungiert komplexe Multiplikationen. Da die Anzahl der Mikrofone N als konstant gewählt wurde, wurde der Prototyp für die Funktion als Kombination des FFT-üblichen logarithmischen Anteils und einem linearen Anteil gewählt.

Tabelle 6.3.: Profiling der Funktion *Correlation_except_var*

FFT-Länge	Optimierung	Aufrufe/Durchlauf	Taktzyklen
128	o3-3	1	181782
256	o3-3	1	357396
512	o3-3	1	735696
1024	o3-3	1	1465182
1024	/	1	2421480

$$f(K) = 1,938 \cdot K \cdot log(K) + 1422 \cdot K + 2392 \tag{6.4}$$

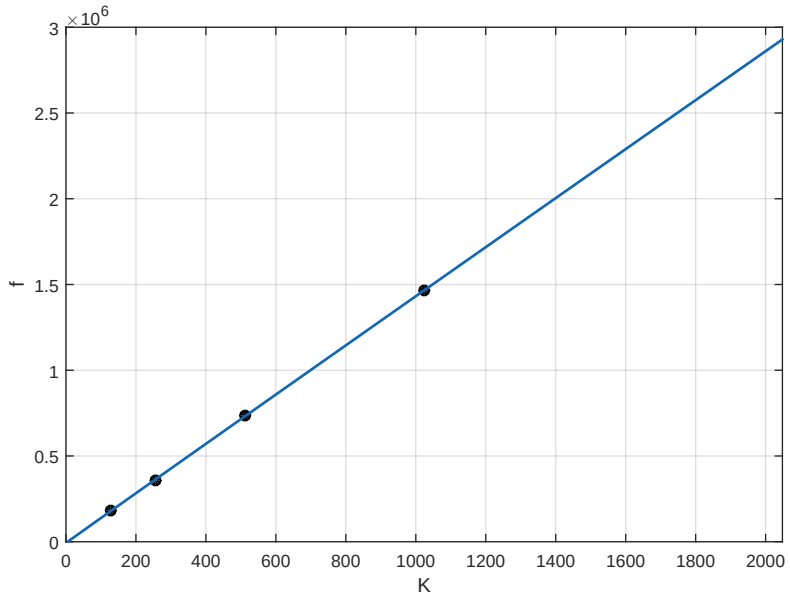


Abbildung 6.4.: Messpunkte des Profilings für die Funktion *Correlation_except_var* mit Verlauf der möglichen zugrunde liegenden Funktion

Alle weiteren für den MCCC-Algorithmus benötigten Funktionen hängen nicht von der Blocklänge, sondern von der Abtastfrequenz und dem Mikrofonabstand, wodurch die Auflösung bestimmt wird, ab. Beide Parameter sollen hier mit einer Abtastrate von 48 kHz und einem Mikrofonabstand von 5 cm fest gewählt werden, womit der Aufwand für diese konstant ist.

Tabelle 6.4.: Profiling von Funktionen für die MCCC, welche nicht von der Blocklänge abhängen

Funktion	Optimierung	Aufrufe/Durchlauf	Taktzyklen
det_Ra_p	/	23	21648
	o3-3		15486
Ra_p_extra_vars	/	23	8544
	o3-3		1458
min_idx	/	1	690
	o3-3		438
get_angle	/	1	1152
	o3-3		1152

Der Gesamtaufwand in Abhängigkeit der FFT-Länge ergibt sich somit zu:

$$
\begin{aligned}
f(k) &= 23 \cdot (15486 + 690) + 1152 + 438 + 1,938 \cdot k \cdot log(k) + 1422 \cdot k + 2392 \\
&\quad + 8 \cdot (1,936 \cdot k + 45,69) + 8 \cdot (1,188 \cdot k \cdot log(k) + 263,1) \\
&= 11,442 \cdot k \cdot log(k) + 1437,488 \cdot k + 396164,32
\end{aligned} \tag{6.5}
$$

In Tabelle 6.5 sind die benötigten Taktzyklen für einige gängige FFT-Längen aufgeführt.

Tabelle 6.5.: Rechenaufwand des MCCC-Algorithmus für verschiedene FFT-Längen

FFT-Länge	Taktzyklen
128	583248
256	771215
512	1148029
1024	1903422
2048	3417734

Auch der Rechenaufwand für MUSIC hängt von diversen Parametern ab, wovon ebenfalls nur die wichtigsten berücksichtigt werden. Auch bei MUSIC ist es notwendig, eine FFT für jedes Signal durchzuführen. Der Aufwand hängt von der FFT-Länge ab und wurde bereits zuvor ermittelt. Ursprünglich handelt es sich bei MUSIC um ein Schmalband-Verfahren, was durch die kombinierte Anwendung des Verfahrens für mehrere Frequenzen auch breitbandig eingesetzt werden kann. Je nach Anzahl dieser Frequezstützstellen erhöht sich der Rechenaufwand linear.
Für die Singulärwertzerlegung wurden zwei unterschiedliche Varianten implementiert, welche jeweils pro Frequenz ein Mal ausgeführt werden müssen. Dabei arbeitet eine Variante mit reellwertigen 14x16 Matrizen und das andere Verfahren mit komplexwertigen 7x8 Matrizen.

Tabelle 6.6.: Profiling von Funktionen für Eigenwertzerlegung mit reellwertigen Matrizen

Funktion	Optimierung	Aufrufe/Durchlauf	Taktzyklen
hilbert_Mat_real_SVD	/	1/Frequenz	6426
	o3-3		2142
svdcmp	/	1/Frequenz	870558
	o3-3		354048
sort	/	1/Frequenz	5880
	o3-3		2526
get_noise_real_svd	/	1/Frequenz	3582
	o3-3		1008

Tabelle 6.7.: Profiling von Funktionen für Eigenwertzerlegung mit komplexwertigen Matrizen

Funktion	Optimierung	Aufrufe/Durchlauf	Taktzyklen
hilbert_Mat	/	1/Frequenz	3276
	o3-3		480
CSVD	/	1/Frequenz	554658
	o3-3		380472

Das reellwertige Verfahren ist bei einem Rechenaufwand von 359724 Taktzyklen etwas schneller als das komplexwertige mit 380952 Taktzyklen pro Frequenz. Darum soll dieses für die Bestimmung des Gesamtaufwands verwendet werden.

Die Funktion *add_NULL_SPECTRUM_fix_steering* addiert das reziproke MUSIC-Spektrum für vorberechnete Steering-Vektoren. Diese Funktion muss ebenfalls für jede Frequenz ein Mal ausgeführt werden und hängt außerdem von der Frequenzauflösung ab. Die Funktion berechnet abhängig von der Auflösung verschieden viele Werte für das Null-Spektrum im Bereich zwischen -90° und +90°. Daher liegt es nahe, dass der Aufwand linear von der Anzahl der berechneten Werte abhängt. Das bedeutet, dass dieser mit dem Verhältnis $1/\alpha$ mit gröber werdender Frequenzauflösung abnimmt. So verdoppelt sich der Aufwand mit der doppelten Frequenzauflösung.

Tabelle 6.8.: Profiling der Funktion *add_NULL_SPECTRUM_fix_steering*

Auflösung / °	Optimierung	Aufrufe/Durchlauf	Taktzyklen
2	o3-3	1/Frequenz	166278
1	o3-3	1/Frequenz	329754
1	/	1/Frequenz	1698138
0,5	o3-3	1/Frequenz	662466

$$f(\alpha, L) = L \cdot \frac{3.31 \cdot 10^5}{\alpha} \tag{6.6}$$

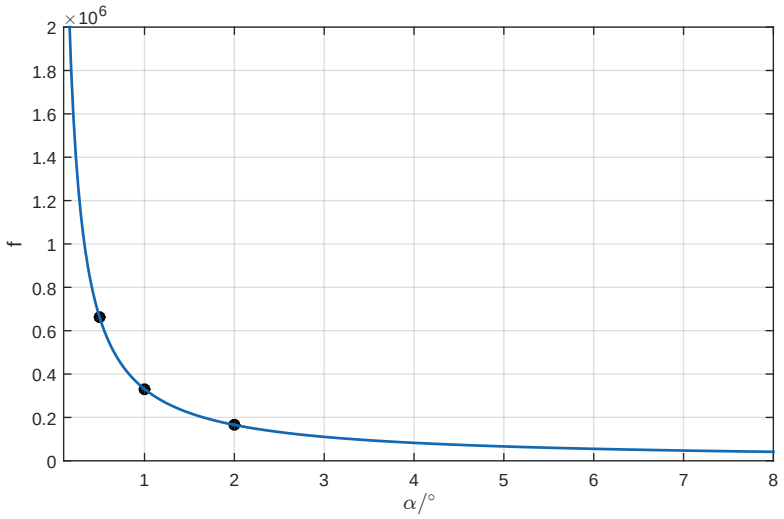


Abbildung 6.5.: Messpunkte des Profilings für die Funktion *add_NULL_SPECTRUM_fix_steering* mit Verlauf der möglichen zugrunde liegenden Funktion

Die Funktion *add_NULL_SPECTRUM* berechnet zusätzlich für jeden Testwinkel einen Steering-Vektor, da die Testfrequenzen erst zur Laufzeit bekannt sind. Der Aufwand folgt somit der gleichen Funktion mit einem unterschiedlichen Faktor.

Tabelle 6.9.: Profiling der Funktion add_NULL_SPECTRUM

Auflösung / °	Optimierung	Aufrufe/Durchlauf	Taktzyklen
2	o3-3	1/Frequenz	298158
1	o3-3	1/Frequenz	592284
1	/	1/Frequenz	2012514
0,5	o3-3	1/Frequenz	1186212

$$f(\alpha, L) = L \cdot \frac{5,931 \cdot 10^5}{\alpha} \tag{6.7}$$

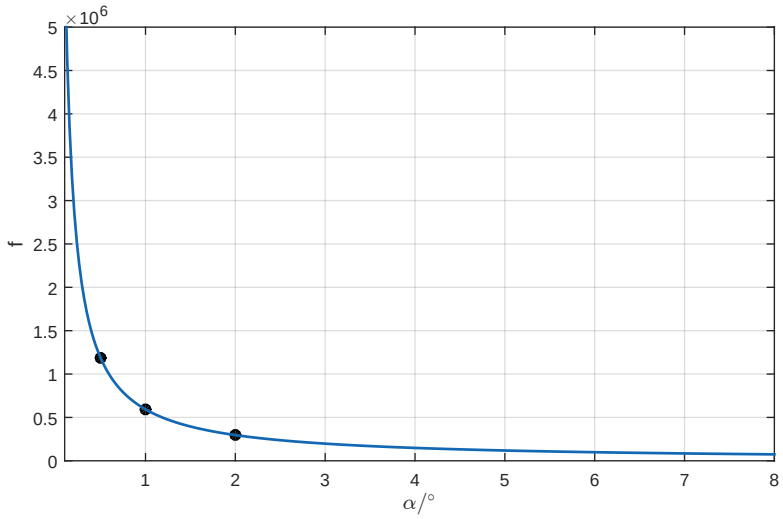


Abbildung 6.6.: Messpunkte des Profilings für die Funktion *add_NULL_SPECTRUM* mit Verlauf der möglichen zugrunde liegenden Funktion

Um aus der Summe aller Nullspektren das MUSIC-Spektrum zu erhalten, muss jeder Wert daraus invertiert werden. Der Aufwand hängt lediglich von der Winkelauflösung ab, da er unabhängig von der Anzahl der Testfrequenzen nur ein Mal auftritt.

Tabelle 6.10.: Profiling der Funktion *invertS*

Auflösung / °	Optimierung	Aufrufe/Durchlauf	Taktzyklen
2	o3-3	1	9468
1	o3-3	1	18828
0,5	/	1	40850
0,5	o3-3	1	36792

$$f(\alpha) = \frac{1.818 \cdot 10^4}{\alpha} + 486 \tag{6.8}$$

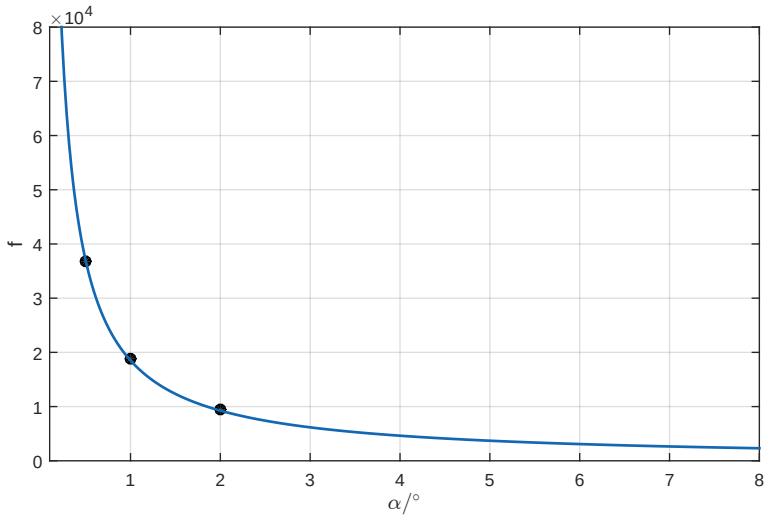


Abbildung 6.7.: Messpunkte des Profilings für die Invertierung des Null-Spektrum mit Verlauf der möglichen zugrunde liegenden Funktion

Wird die Variante des Algorithmus verwendet, welche mit variablen Testfrequenzen arbeitet, muss die Grundfrequenz des Eingangssignals bestimmt werden. Dies geschieht mithilfe der AMDF, welche einen Aufwand hat, der quadratisch von der FFT-Länge abhängt.

Tabelle 6.11.: Profiling der Funktion *AMDF*

FFT-Länge	Optimierung	Aufrufe/Durchlauf	Taktzyklen
256	o3-3	1	300006
512	o3-3	1	604488
1024	o3-3	1	1324194
1024	/	1	3057912

$$f(k) = 0,2816 \cdot k^2 + 973,1 \cdot k + 32440 \tag{6.9}$$

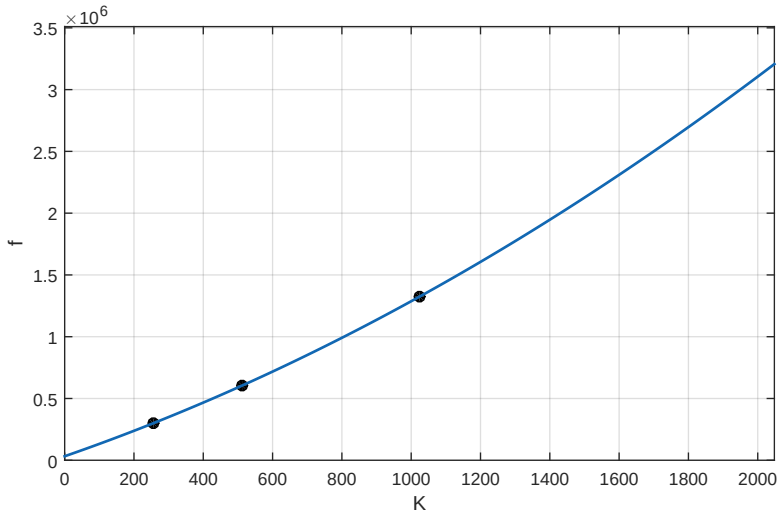


Abbildung 6.8.: Messpunkte des Profilings für die AMFD mit Verlauf der möglichen zugrunde liegenden Funktion

Die Funktion *transjun_mult* muss ein Mal pro Testfrequenz ausgeführt werden. Die Funktion *get_angles* wird pro Durchlauf lediglich ein Mal ausgeführt.

Tabelle 6.12.: Profiling von Funktionen für MUSIC, welche nicht von betrachteten Parametern abhängen

Funktion	Optimierung	Aufrufe/Durchlauf	Taktzyklen
transjun_mult	/	1/Frequenz	63156
	o3-3		18804
get_angles	/	1	9792
	o3-3		6552

Es ergibt sich als Gesamtaufwand für den MUSIC-Algorithmus in der vorliegenden Implementierung mit festen Steering-Vektoren:

$$\begin{aligned} f(K,\alpha,L) &= L \cdot \left(359724 + 18804 + \frac{3,31 \cdot 10^5}{\alpha}\right) + \frac{1,818 \cdot 10^4}{\alpha} + 486 \\ &\quad + 8 \cdot 1,188 \cdot K \cdot log(K) + 2088,8 + 6552 \\ &= L \cdot \left(378528 + \frac{3,31 \cdot 10^5}{\alpha}\right) + \frac{1,818 \cdot 10^4}{\alpha} \\ &\quad + 9,504 \cdot K \cdot log(K) + 9126,8 \end{aligned} \tag{6.10}$$

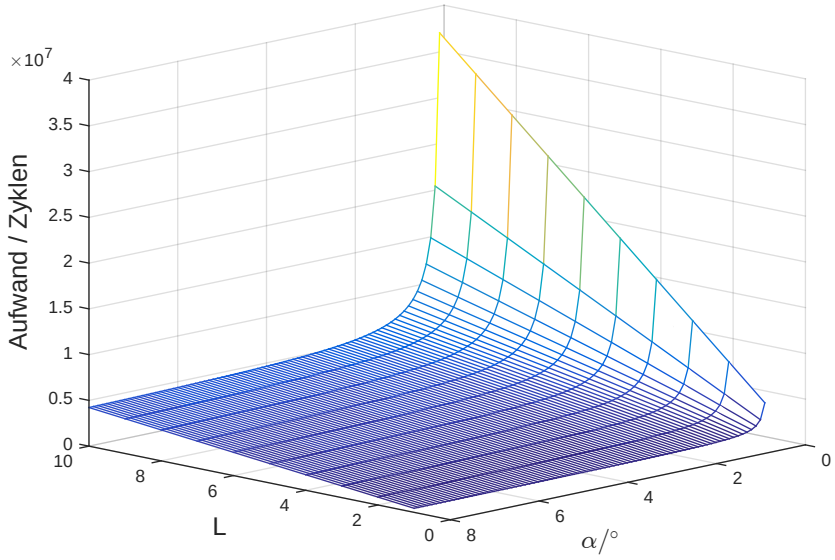


Abbildung 6.9.: Gesamtaufwand des MUSIC-Algorithmus in Abhängigkeit der Testfrequenzen und der Auflösung bei einer FFT-Länge von 1024 bei festen Steering-Vektoren

Die folgende Tabelle zeigt den Aufwand des gesamten Algorithmus bei festen Steering-Vektoren für verschiedene Kombinationen von Parametern.

Tabelle 6.13.: Rechenaufwand des MUSIC-Algorithmus für verschiedene Parameter mit festen Steering-Vektoren

FFT-Länge	Testfrequenzen	Auflösung / °	Taktzyklen
512	4	0,5	4220782
512	8	0,5	8382894
512	8	1	5716714
1024	8	1	5732827
1024	4	8	1720308

Bei der Verwendung von variablen Steering-Vektoren ergibt sich ein Gesamtaufwand von:

$$\begin{aligned} f(K,\alpha,L) &= L \cdot \left(359724 + 18804 + \frac{5,931 \cdot 10^5}{\alpha}\right) + \frac{1,818 \cdot 10^4}{\alpha} + 486 \\ &\quad + 8 \cdot 1,188 \cdot K \cdot log(K) + 2088,8 + 6552 + 0,2816 \cdot k^2 + 973,1 \cdot k + 32440 \\ &= L \cdot \left(378528 + \frac{5,931 \cdot 10^5}{\alpha}\right) + \frac{1,818 \cdot 10^4}{\alpha} \\ &\quad + 9,504 \cdot K \cdot log(K) + 0,2816 \cdot K^2 + 973,1 \cdot K + 41566,8 \end{aligned} \tag{6.11}$$

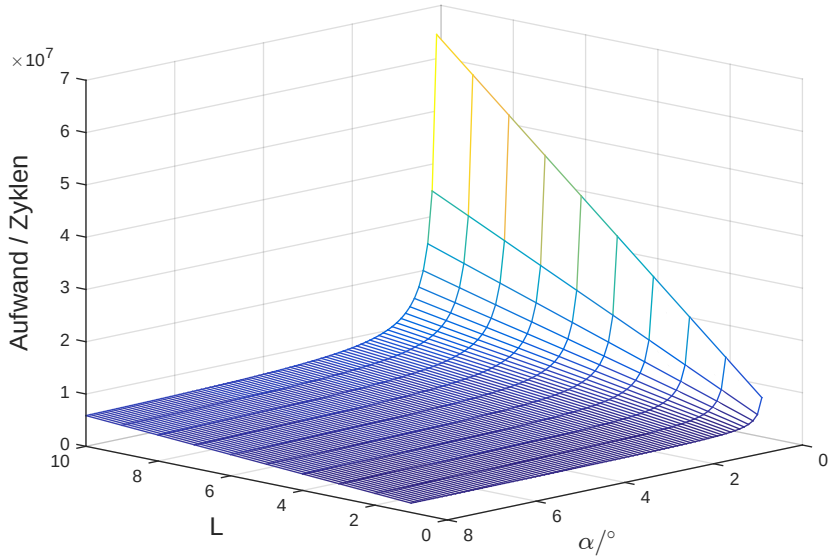


Abbildung 6.10.: Gesamtaufwand des MUSIC-Algorithmus in Abhängigkeit der Testfrequenzen und der Auflösung bei einer FFT-Länge von 1024 bei variablen Steering-Vektoren

Tabelle 6.14.: Rechenaufwand des MUSIC-Algorithmus für verschiedene Parameter mit variablen Steering-Vektoren

FFT-Länge	Testfrequenzen	Auflösung / °	Taktzyklen
512	4	0,5	6922069
512	8	0,5	13180981
512	8	1	8418001
1024	8	1	9153800
1024	4	8	3175531

6.3. Betrachtung des Speicherplatzbedarfs

Im Folgenden soll eine Betrachtung des benötigten Speicherbedarfs stattfinden. Dabei werden die wichtigsten Variablen betrachtet, welche für die jeweiligen Algorithmen benötigt werden. Zählervariablen und andere Variablen, die nur sehr wenig Speicherplatz in Anspruch nehmen, werden hier vernachlässigt.

Tabelle 6.15.: Speicherplatzbedarf für Variablen des MCCC-Algorithmus

Variable	Speicherplatzbedarf in Byte
SignalArray	$64 \cdot K$
OutArray	$64 \cdot K$
CorrInArray	$224 \cdot K$
CorrOutArray	$224 \cdot K$
vars	64
Ra	256
det_Ra_p	92

Der Speicherbedarf für die vom MCCC-Algorithmus benötigten Variablen beträgt somit in etwa:

$$f(K) = 576 \cdot K + 412 \tag{6.12}$$

Für eine FFT-Länge von 1024 werden somit mehr als 590236 Byte benötigt.
Beim MUSIC-Algorithmus muss wiederum unterschieden werden, ob variable oder feste Steering-Vektoren verwendet werden. Bei der Verwendung von festen, die vor der Laufzeit berechnet werden, ist von einem höheren Speicherbedarf auszugehen, als bei der Berechnung zur Laufzeit.

Tabelle 6.16.: Speicherplatzbedarf für Variablen des MUSIC-Algorithmus mit festen Steering-Vektoren

Variable	Speicherplatzbedarf in Byte
SignalArray	$64 \cdot K$
OutArray	$64 \cdot K$
S1	$180/\alpha$
A1	2040
w1	136
v1	2040
R	512
steer	$11520 \cdot L/\alpha$

$$f(K, \alpha, L) = 128 \cdot K + \frac{180 + 11520 \cdot L}{\alpha} + 4728 \tag{6.13}$$

Bei einer FFT-Länge von 1024, einer Auflösung von einem Grad und der Verwendung von vier Testfrequenzen ergibt sich ein ungefährer Speicherplatzbedarf von 182060 Byte.
Bei der Verwendung von variablen Frequenzen kann jeder Steering-Vektor zur Laufzeit berechnet werden, wodurch sich der Speicherplatzbedarf auf 64 Byte verringert. Alle anderen Werte bleiben unverändert. Der gesamte Speicherplatz lässt sich nun wie folgt bestimmen:

$$f(K, \alpha, L) = 128 \cdot K + \frac{180}{\alpha} + 4792 \tag{6.14}$$

Für eine FFT-Länge von 1024 und einer Auflösung von einem Grad ergibt sich unabhängig von der Anzahl der Testfrequenzen ein Speicherplatzbedarf von etwa 136044 Bytes.

7. Weitere Untersuchung der Algorithmen

7.1. Theoretische Genauigkeit der Verfahren

In der vorliegenden Arbeit geht es um eine Gegenüberstellung von verschiedenen Verfahren zur Sprecherlokalisierung. Aus den Abschnitten 2.3 und 2.4 geht hervor, dass diese Verfahren auf grundverschiedenen Ansätzen basieren. In diesem Kapitel sollen die Eigenschaften der Verfahren in Bezug auf ihre Genauigkeit betrachtet werden. Eine überaus wichtige Größe zur Beurteilung der Güte eines Verfahrens ist die Winkelauflösung. Diese gibt Auskunft über die höchstmögliche Genauigkeit.

7.1.1. Zeitbasierte Verfahren

Echtzeitsysteme auf DSP-Basis arbeiten meist mit einer oder mehreren festen Abtastraten, mit denen Audio-Signale erfasst werden. Bei zeitbasierten Verfahren geben diese die Auflösung vor. Die zeitliche Differenz der Ankunftszeit eines Signals zwischen den Mikrofonen (TDOA) muss bezogen auf die Abtastrate ganzzahlig sein [32, S. 36 f.]. Das bedeutet, dass die Abtastrate ein Raster vorgibt und der ermittelte Winkel des Sprechers in diesem Raster liegen muss. Die maximal mögliche Auflösung bei gegebener Arraygeometrie wird somit durch die Abtastrate bestimmt.
Wird in Gleichung 2.3

$$\tau = \frac{\kappa}{f_a} \tag{7.1}$$

gesetzt, so ergibt sich:

$$\kappa = \frac{d}{v} \cdot sin(\phi) \cdot f_a \tag{7.2}$$

Der detektierte Winkel in Abhängigkeit der ganzzahligen Verzögerung κ der Samples ergibt sich durch folgende Umformung [32, S. 37]:

$$\phi = asin\left(\frac{\kappa \cdot v}{d \cdot f_a}\right) \cdot \frac{180}{\pi} \tag{7.3}$$

Die Geschwindigkeit v ist im vorliegenden Fall die Schallgeschwindigkeit und wird als konstant angenommen. Es bleiben somit die Abtastrate und der Mikrofonabstand als maßgebliche Parameter, welche die Winkelauflösung bestimmen. Da κ ebenfalls im Argument des Arcussinus steht, muss es einen nichtlinearen Zusammenhang zwischen Winkelauflösung und Verzögerung der Samples geben. Abbildung 7.1 zeigt die mögliche Winkelauflösung als Funktion der Abtastrate und des Mikrofonabstandes für $\kappa = 1$.

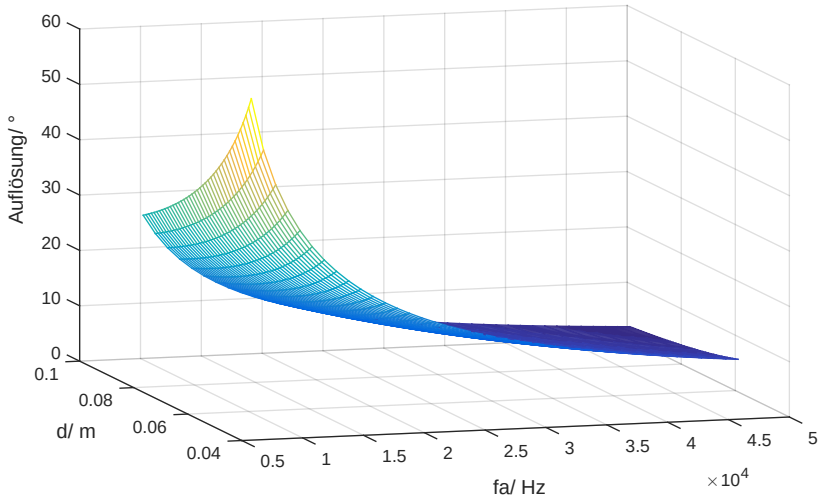


Abbildung 7.1.: Auflösung des MCCC in Abhängigkeit der Abtastrate und des Mikrofonabstands

Die folgende Tabelle zeigt die maximale Winkelauflösung für einen festen Mikrofonabstand bei gängigen Abtastraten.

Tabelle 7.1.: Maximale Auflösung der MCCC in Abhängigkeit der Abtastrate für $d = 5cm$

Abtastrate/ kHz	**Minimale Winkelauflösung/ °**
8	59,0928
16	25,4041
24	16,6186
48	8,2215

Wie bereits zuvor beschrieben findet dadurch, dass κ im Argument des Arcussinus steht, eine nichtlineare Quantisierung statt, wodurch die Auflösung in Abhängigkeit von κ variiert. Die folgende Abbildung zeigt die entstehende Quantisierungskennlinie für einen Mikrofonabstand von 5 cm und einer Abtastrate von 48 kHz.

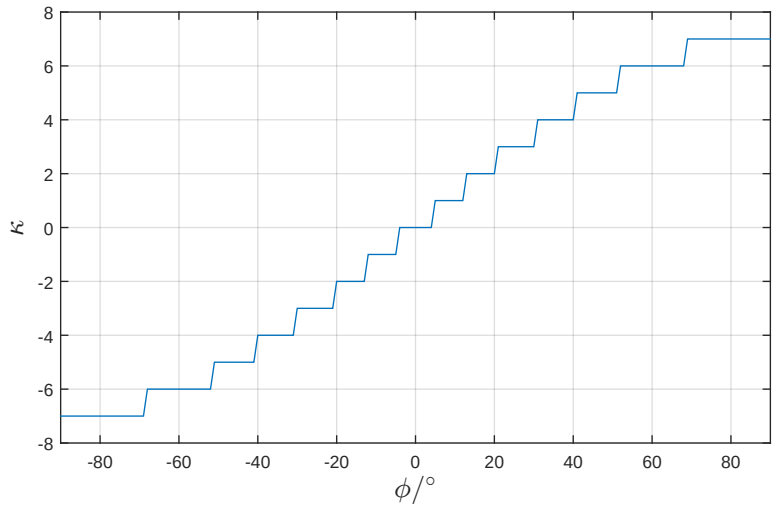


Abbildung 7.2.: Quantisierungskennlinie des Sprecherwinkels für zeitbasierte Verfahren mit einem Mikrofonabstand von $d = 5cm$ und einer Abtastrate von $f_a = 48kHz$

Es ist deutlich zu erkennen, dass die Stufen zwischen $-60°$ und $+60°$ annähernd gleich groß sind. In diesem Bereich entspricht die Winkelauflösung in etwa der maximalen Winkelauflösung aus Tabelle 7.1. Bei diesem Bereich handelt es sich um den annähernd linearen Bereich

der Arcussinufunktion. Außerhalb dieses Bereichs wird die Auflösung deutlich schlechter, da die Steigung der Arcussinusfunktion hier flacher wird. Es ergibt sich ein systematischer Fehler in Abhängigkeit des Winkels.

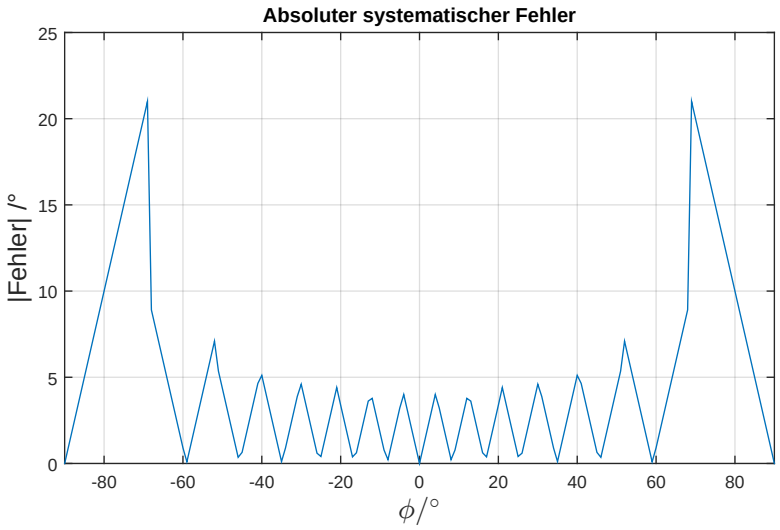


Abbildung 7.3.: Absoluter systematischer Fehler bei zeitbasierten Verfahren in Abhängigkeit des Winkels bei einem Mikrofonabstand von $d = 5cm$ und einer Abtastrate von $f_a = 48kHz$

Dieser Fehler ist jeweils halb so groß wie die korrespondierende Auflösung, da der tatsächliche Winkel im ungünstigsten Fall genau zwischen zwei möglichen Winkeln liegen kann.

7.1.2. Unterraumbasierte Verfahren

Unterraumbasierte Algorithmen, wie MUSIC, basieren, wie in Abschnitt 2.4 beschrieben, auf der Zerlegung der Kovarianzmatrix in Rausch- und Signalanteil und arbeiten im Frequenzbereich. Die maximal mögliche Winkelauflösung ist daher bei einer exakt ermittelten Kovarianzmatrix bei einer planaren Wellenfront ebenfalls exakt und unabhängig vom SNR [22]. Das bedeutet, dass die Winkelauflösung bei diesen Verfahren theoretisch unendlich genau ist.
Selbstverständlich ist das in der Realität nicht der Fall. Hier kommen Einflüsse wie numerische Ungenauigkeiten und statistische Einflüsse zum Tragen, sodass die Auflösung nie

perfekt sein kann. Die Genauigkeit wird maßgeblich von statistischen Zusammenhängen beeinflusst, welche in [22] hergeleitet werden. Die Ergebnisse sollen an dieser Stelle vorgestellt werden.
Wie zuvor erwähnt, liefert der MUSIC-Algorithmus, welcher hier stellvertretend für die unterraumbasierten Verfahren behandelt werden soll, exakte Ergebnisse, sofern eine exakte Kovarianzmatrix vorliegt. Da diese in den konkreten Anwendungen lediglich anhand einer bestimmten Anzahl von Samples geschätzt werden kann, liegt es nahe, dass die Genauigkeit des Algorithmus von der Genauigkeit der Kovarianzmatrix abhängt. Es resultiert eine Verteilungsfunktion für den Wert des Nullspektrums[3] bei einer Quelle, welche wie folgt lautet:

$$\overline{\hat{D}(\tau(\phi_i))} = \frac{\eta_i \cdot \sigma_v^2 \cdot (N-1)}{K \cdot (\eta_i - \sigma_v^2)^2} \tag{7.4}$$

Darin ist η_i der zur Quelle korrespondierende Eigenwert, N ist die Anzahl der Mikrofone, K die zur Schätzung der Kovarianzmatrix verwendete Anzahl von Samples und σ_v^2 die Varianz des Rauschens an den Mikrofonen.
Aus Gleichung 7.4 geht hervor, dass der Wert des Nullspektrums an der Stelle $\tau(\phi = \phi_i)$ bei einer sehr großen Anzahl K von Samples gegen null geht. Das bedeutet, dass das MUSIC-Spektrum an dieser Stelle einen Peak aufweist, der gegen unendlich geht. Mit kleiner werdender Anzahl von Samples und größer werdender Rauschamplitude wird dieser Peak immer weniger ausgeprägt, was in schlechteren Ergebnissen resultiert.
Demzufolge hängt die Genauigkeit des Algorithmus zwar von den Systemparametern ab, die theoretische Winkelauflösung ist jedoch unendlich.
Die Performance lässt ist sich am besten durch den Fehler beschreiben, welcher laut [39] normalverteilt und mittelwertfrei ist und dessen Varianz sich folgendermaßen beschreiben lässt:

$$E(\hat{\phi}_i - \phi_i) = \frac{\sigma_v}{2M} \cdot \frac{\sum_{k=1}^{D} \frac{\eta_k}{\sigma_v - \eta_k} |\boldsymbol{\varsigma}^H(\tau(\phi_i))\boldsymbol{q}_k|}{\sum_{k=D+1}^{N} |\boldsymbol{d}^H(\tau(\phi_i))\boldsymbol{q}_k|} \tag{7.5}$$

[3] Reziprok des MUSIC-Spektrums

Darin ist $\boldsymbol{d}(\tau(\phi_i))$ die erste Ableitung des Steering-Vektors nach dem Winkel, $\boldsymbol{q}_m$ sind die Eigenvektoren, von denen die ersten D dem Signal- und der Rest dem Rauschunterraum zuzuordnen sind. M ist die Anzahl der Zufallsexperimente. Aus Gleichung 7.5 geht hervor, dass die Varianz des Fehlers große Werte annimmt, wenn ein zu einer Quelle korrespondierender Eigenwert in der Größenordnung der Rauschvarianz liegt, was an einem kleinen SNR liegt. Dieser Fall tritt ebenfalls ein, wenn mehrere stark korrelierte Quellsignale vorhanden sind. Dabei kann es sich auch um dasselbe Quellsignal handeln, welches wie in Abschnitt 2.3.1 beschrieben über verschiedene Wege zum Mikrofonarray gelangt.
Obwohl die Winkelauflösung von MUSIC und anderen unterraumbasierten Algorithmen in der Theorie zwar unbeschränkt ist, kann sie nicht als ebenso genau angenommen werden, da diese durch Einflüsse der Umgebung fehlerbehaftet ist. Ein weiter wichtiger Aspekt ist die endliche zur Berechnung zur Verfügung stehende Zeit. Da für jeden möglichen Winkel eine Berechnung des Werts des MUSIC-Spektrum stattfinden muss, ist es unmöglich, eine unendlich feine Auflösung in endlicher Zeit zu realisieren. Vielmehr muss hier ein Weg gefunden werden, bei dem die Auflösung so gewählt wird, dass sie den Anforderungen entspricht und das System gleichzeitig echtzeitfähig bleibt. Zudem wäre es nicht sinnvoll, eine Auflösung zu wählen, die deutlich kleiner ist als die Streuung des Fehlers, da dieser die maximale Genauigkeit bestimmt.

7.2. Messungen im reflexionsarmen Schallmessraum

Nachdem die Algorithmen zunächst mit simulierten Daten getestet und die Ergebnisse anhand einer MATLAB-Simulation verifiziert wurden, sollte das Gesamtsystem ebenfalls getestet werden. Dafür wurden zunächst Messungen in einem sehr reflexionsarmen Raum durchgeführt. Auch wenn dieser nur bedingt die Realität widerspiegelt, lassen sich die Ergebnisse hieraus sehr gut verwerten. Für jeden der implementierten Algorithmen finden sich in der Literatur viele Methoden zur Signalvorverarbeitung, mit welchen die Algorithmen unter verschieden Umgebungseinflüssen, wie beispielsweise Nachhall, besser arbeiten. Diese hängen von den jeweiligen Algorithmen und Einflüssen ab und erfordern teilweise einen erheblichen Aufwand. Im reflexionsarmen Schallmessraum sind diese Umwelteinflüsse auf ein Minimum reduziert, sodass die Algorithmen auch ohne aufwendige Vor- bzw. Nachtverarbeitung untersucht werden können. Hierfür wurde folgende Versuchsanordnung installiert:

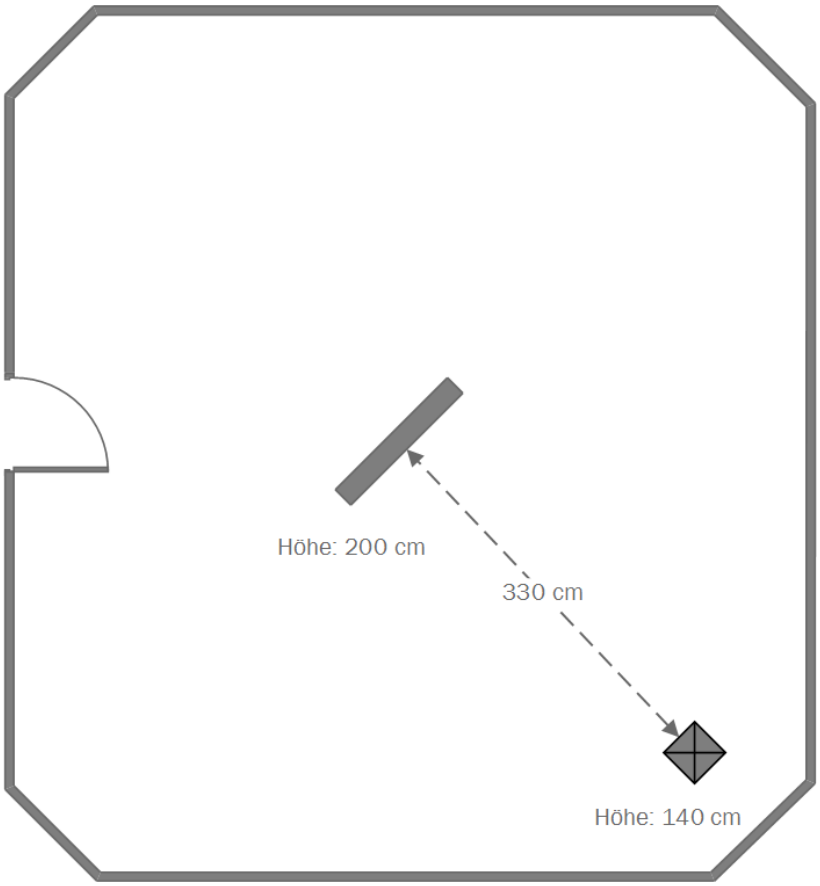
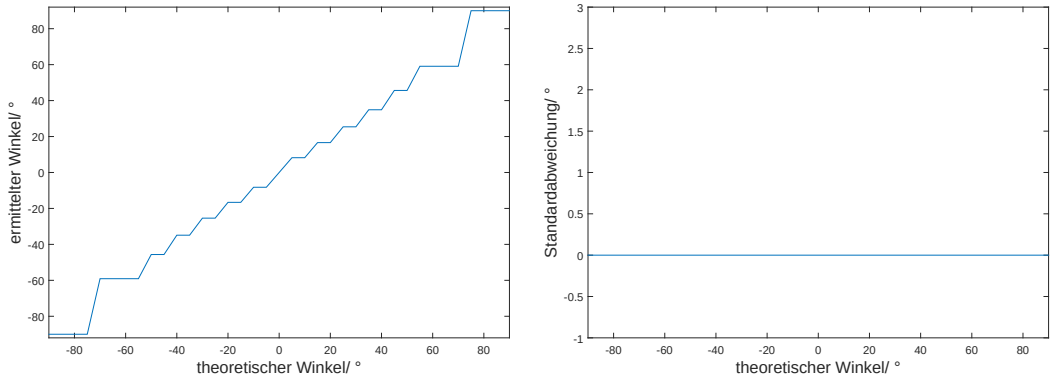


Abbildung 7.4.: Versuchsanordnung im reflexionsarmen Schallmessraum

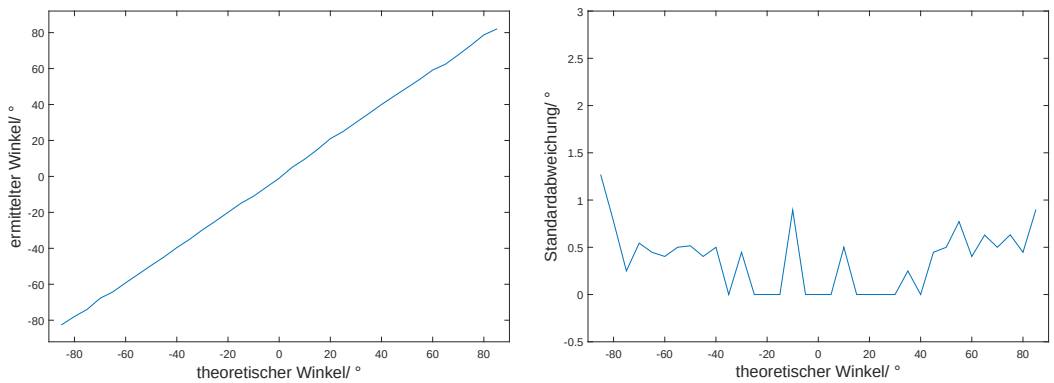
Das Mikrofonarray befindet sich in 3,3 m Abstand zu einem Lautsprecher. Dieser ist in 1,4 m Höhe auf einem Ständer montiert. Das Mikrofonarray wurde auf einem Drehteller angebracht, welcher an Drahtseilen an der Decke aufgehängt ist und befindet sich in dieser Anordnung in 2 m Höhe.
Zunächst sollten die Algorithmen über den gesamten Winkelbereich getestet werden. Dafür wurde bandbegrenztes Rauschen als Signalquelle verwendet. Das Mikrofonarray wurde auf dem Drehteller in Schritten von fünf Grad in beide Richtungen verdreht. Die Messungen wurden für beide Algorithmen über mehrere Blöcke aufgezeichnet, sodass neben dem durchschnittlich ermittelten Winkel auch die jeweilige Standardabweichung für jeden eingestellten Winkel ermittelt werden kann. Für die Auswertung wurden jeweils 100 Werte herangezogen und für jeden eingestellten Winkel sowohl Mittelwert auch als Standardabweichung ermittelt.

- Abtastfrequenz: 8 kHz (MUSIC) bzw. 48 kHz (MCCC)
- Fensterlänge: 1024 Samples
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Schmalbandrauschen



(a) Ermittelter und simulierter Winkel (b) Standardabweichung

Abbildung 7.5.: Ergebnisse der Messung im Schallmessraum für verschiedene Winkel mit dem MCCC



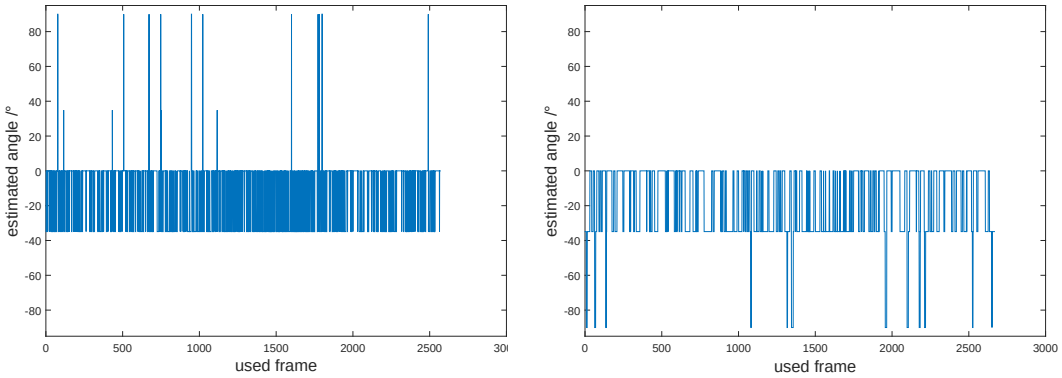
(a) Ermittelter und simulierter Winkel (b) Standardabweichung

Abbildung 7.6.: Ergebnisse der Messung im Schallmessraum für verschiedene Winkel mit MUSIC

Die Ergebnisse der Messungen zeigen zunächst, dass beide Verfahren funktionieren und plausible Ergebnisse liefern. Auch zeigt sich, dass die Ergebnisse des MCCC in das durch

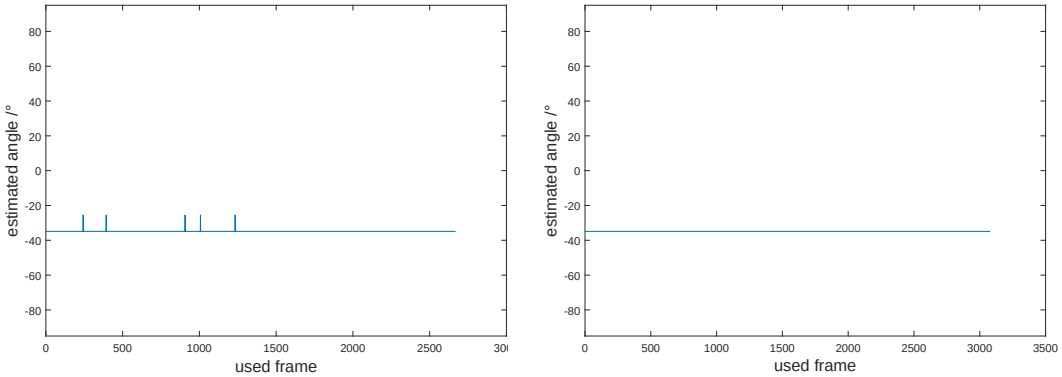
die Abtastrate vorgegebene Raster fallen. Die Standardabweichung des MCCC beträgt für alle eingestellten Winkel annähernd null. Die mit dem MUSIC-Algorithmus erzielten Ergebnisse sind ebenfalls plausibel. Hier wird das Raster durch die verwendeten Steering-Vektoren vorgegeben. Die gewählte Winkelauflösung bei dem vorliegenden Test betrug 1°. Auffällig ist hier, dass der Verlauf der Standardabweichung über dem Sollwinkel scheinbar keinem Muster folgt. Eine mögliche Erklärung dafür ist der nicht ideale Versuchsaufbau. Einerseits war das Mikrofonarray auf einem aufgehängten Drehteller montiert, welcher schwingen kann. Außerdem hat dieser lediglich eine sehr grobe Skalierung, weshalb die Winkeleinstellung zusätzliche Toleranzen mit sich bringt. Der Lautsprecher wurde mittels Halterung auf dem Boden des Messraums aufgestellt. Dieser besteht aus einem Drahtgitter, welches ebenfalls schwingen kann. Als Abhilfe wurde jede Winkeleinstellung mit großer Vorsicht vorgenommen und anschließend für mehrere Minuten gewartet, um sämtliche Schwingungen möglichst gut abklingen zu lassen. Messfehler lassen sich trotzdem nicht ausschließen.
Anschließend wurden die Algorithmen mit tatsächlichen Sprachsignalen getestet. Hierfür wurde die Aufnahme einer Rede vor dem deutschen Bundestag verwendet. Die Sprecherin ist weiblich und die Rede beinhaltet deutliche Sprechpausen und Applaus. Zunächst wurde der MCCC getestet. Die folgenden Abbildungen enthalten die Ergebnisse für verschiedene Blocklängen jeweils mit und ohne Medianfilter, wobei die sonstigen Einstellungen wie folgt lauten:

- Abtastfrequenz: 48 kHz
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Frauenstimme
- Winkel: -35°



(a) ohne Medianfilter (b) mit Medianfilter

Abbildung 7.7.: Ergebnisse des MCCC aus Messraum für eine Sequenz bei einer Blocklänge von 256

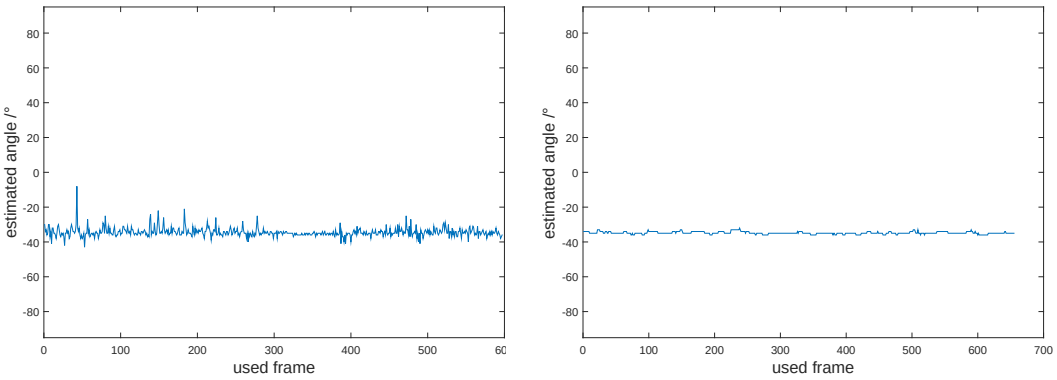


(a) ohne Medianfilter (b) mit Medianfilter

Abbildung 7.8.: Ergebnisse des MCCC im Messraum für eine Sequenz bei einer Blocklänge von 512

Es zeigt sich deutlich, dass der Algorithmus bei der vorliegenden Implementierung mit einer Blocklänge von 256 Samples ein sehr hohes Maß an Fehldetektionen aufweist. Mit einer Blocklänge von 512 Samples funktioniert der MCCC jedoch sehr gut. Die Ergebnisse entsprechen den Erwartungen. Diese Beobachtung konnte - wie zuvor beschrieben - bereits während der MATLAB-Simulation gemacht werden und soll im nächsten Abschnitt untersucht werden.
Zunächst wird jedoch das Ergebnis des Test mit dem MUSIC-Algorithmus vorgestellt. Die eingestellten Parameter lauten wie folgt:

- Abtastfrequenz: 8 kHz
- Anzahl der Mikrofone: 8
- Abstand zwischen den Mikrofonen: 5 cm
- Quellsignal: Frauenstimme
- Winkel: -35°



(a) ohne Medianfilter (b) mit Medianfilter

Abbildung 7.9.: Ergebnisse von MUSIC im Messraum für eine Sequenz bei einer Blocklänge von 512

Hier ist zu sehen, dass das Ergebnis ohne Medianfilter starken Schwankungen unterliegt. Diese konnten vor allem nach Sprechpausen beobachtet werden. Mit einem Medianfilter wurde hier eine deutliche Verbesserung erzielt.
Die folgende Tabelle beinhaltet eine Übersicht aller Standardabweichungen der hier gezeigten Ergebnisse.

Tabelle 7.2.: Standardabweichungen der implementierten Verfahren

Verfahren	**Standardabweichung** $\sigma/°$
MCCC ohne Median und 256 Samples Blocklänge	21,10
MCCC mit Median und 256 Samples Blocklänge	18,98
MCCC ohne Median und 512 Samples Blocklänge	0,41
MCCC mit Median und 512 Samples Blocklänge	$1.07 \cdot 10^{-12}$
MUSIC ohne Median	2,61
MUSIC mit Median	0,69

7.3. Untersuchung der Fehldetektionen des MCCC-Algorithmus

Bei den ersten Simulationen von dem MCCC-Algorithmus und auch im Schallmessraum hat sich ein zunächst unplausibel scheinendes Verhalten gezeigt. Wurde der SNR zu groß bzw. als unendlich gewählt, konnte beobachtet werden, dass die Determinante von $\boldsymbol{R}_a(0)$ minimal wurde, obwohl der simulierte Winkel nicht bei 0° lag. Die Fehler-Funktion $J_{MCCC}(p)$ hatte ihr Minimum somit unabhängig vom Winkel bei $p = 0$.
Die folgende Abbildung zeigt den Verlauf von $J_{MCCC}(p)$ für eine Abtastrate von 48000 Hz, eine Fensterlänge von 128 Samples, einen SNR von 200 dB und einem Winkel von -50°.

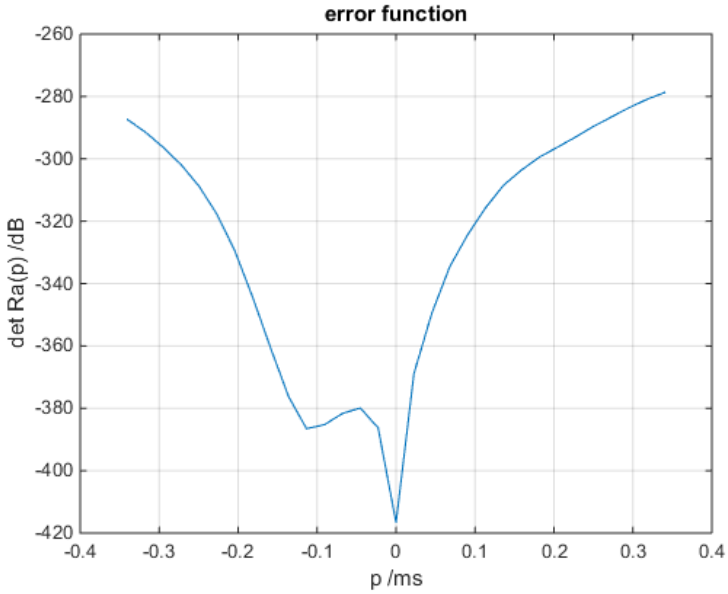


Abbildung 7.10.: Fehlerhaft bei 0° detektierter Winkel

Um diesen Sachverhalt zu ergründen, muss zunächst das Verfahren des MCCC schrittweise nachvollzogen werden. Dies basiert darauf, dass im ersten Schritt alle Signale der N Mikrofone miteinander korreliert werden, um anschließend eine Korrelationsmatrix in Abhängigkeit der zeitlichen Differenz der Eingangssignale zu erstellen. Bei einem ULA beträgt die zeitliche Verzögerung der Eingangssignale zwischen dem ersten und den zweitem Mikrofon

genau τ, zwischen dem ersten und dem dritten genau $2 \cdot \tau$ usw.

$$\boldsymbol{R}_a(p) = \begin{pmatrix} \sigma^2_{y_1} & r_{y_1,y_2}(p) & \cdots & r_{y_1,y_N}(p) \\ r_{y_2,y_1}(p) & \sigma^2_{y_2} & \cdots & r_{y_2,y_N}(p) \\ \vdots & \vdots & \ddots & \vdots \\ r_{y_N,y_1}(p) & r_{y_N,y2}(p) & \cdots & \sigma^2_{y_N} \end{pmatrix} \tag{7.6}$$

Hierbei entspricht $R_a(0)$ genau der Kovarianzmatrix $E[\boldsymbol{y}\boldsymbol{y}^T]$. Der Algorithmus ist so gestaltet, dass nun über alle Werte von p entsprechend -90° bis +90° die zugehörige Korrelationsmatrix erstellt und deren Determinante berechnet wird. Bildlich gesehen werden somit die Signale aller Mikrofone entsprechend ihres Abstandes zum Referenzmikrofon und des aktuellen Werts für p verschoben.

$$\boldsymbol{y}_a(k,p) = [y_1(k)\ y_2(k + \mathcal{F}_2(p))\ \cdots\ y_N(k + \mathcal{F}_N(p))]^T \tag{7.7}$$

Anschließend wird aus den verschobenen Signalen die Kovarianzmatrix gebildet.

$$\boldsymbol{R}_a(p) = E[\boldsymbol{y}_a(k,p)\boldsymbol{y}_a^T(k,p)] \tag{7.8}$$

Wenn der Wert für p gerade genau so gewählt wird, dass $p = \tau$ beträgt, also die Verschiebung genau der Zeitverzögerung der Signale zwischen zwei Mikrofonen entspricht, sind die Signale vollständig korreliert. Die Werte in der Kovarianzmatrix hätten in diesem Fall alle denselben Wert. Somit wären alle Vektoren der Matrix linear voneinander abhängig und folglich die Determinante null.
Dieser Fall tritt in der Realität aufgrund von Rauschen und Ungenauigkeiten im Abstand zwischen zwei Mikrofonen etc. nicht ein. Gesucht wird daher das Minimum der Determinante in Abhängigkeit der Zeitverschiebung.
Ein weiterer Fall, bei dem $det(\boldsymbol{R}_a) = 0$ ist, tritt auf, wenn das Rauschen nicht vorhanden ist und die Anzahl der Signalquellen kleiner der Anzahl der Mikrofone ist. Denn gilt:

$$rank(\boldsymbol{R}_a(0)) = rank(E[\boldsymbol{y}\boldsymbol{y}^T]) = D \tag{7.9}$$

Nur eine quadratische Matrix mit vollem Rang besitzt eine Determinante ungleich null.
Somit ist begründet warum der Wert von $det(\boldsymbol{R}_a(0))$ bei wenig bis gar keinem Rauschen sehr klein bzw. zu null wird.

Diese Zusammenhänge sollen exemplarisch für drei zeitverschobene Sinusfunktionen demonstriert werden. Dazu wird die Korrelationsfunktion für zwei Sinusfunktionen gleicher Frequenz mit unterschiedlichen Phasenverschiebungen und - wie für den MCCC notwendig - einer Funktion für die angenommene Zeitverzögerung $\mathcal{F}_n(p)$ zunächst allgemein berechnet.

$$f_1(t) = sin(\omega t + \phi_1 + \mathcal{F}_1(p)) \tag{7.10}$$

$$f_2(t) = sin(\omega t + \phi_2 + \mathcal{F}_2(p)) \tag{7.11}$$

Die Korrelationsfunktion berechnet sich wie folgt:

$$\begin{aligned}
r_{f_1,f_2}(\tau,p) &= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} f_1(t) \cdot f_2(t-\tau) dt \\
&= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} sin(\omega t + \phi_1 + \mathcal{F}_1(p)) \cdot sin(\omega t - \omega\tau + \phi_2 + \mathcal{F}_2(p)) dt \\
&= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \frac{1}{2} cos(\omega t + \phi_1 + \mathcal{F}_1(p) - (\omega t - \omega\tau + \phi_2 + \mathcal{F}_2(p))) - \\
&\qquad \frac{1}{2} cos(\omega t + \phi_1 + \mathcal{F}_1(p) + (\omega t - \omega\tau + \phi_2 + \mathcal{F}_2(p))) dt \\
&= \frac{1}{2} cos(\omega\tau + \phi_1 + \mathcal{F}_1(p) - \phi_2 - \mathcal{F}_2(p)) - \\
&\qquad \underbrace{\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \frac{1}{2} cos(2\omega t - \omega\tau + \phi_1 + \mathcal{F}_1(p) + \phi_2 + \mathcal{F}_2(p))}_{=0}
\end{aligned} \tag{7.12}$$

Da der Cosinus mittelwertfrei ist, ergibt sich für das Integral in den Grenzen von $-\infty$ bis ∞ der Wert Null. Somit vereinfacht sich der Ausdruck zu:

$$r_{f_1,f_2}(\tau,p) = \frac{1}{2} cos(\omega\tau + \phi_1 + \mathcal{F}_1(p) - \phi_2 - \mathcal{F}_2(p)) \tag{7.13}$$

Nun wird die Korrelationsmatrix der drei phasenvershobenen Sinusfunktionen aufgestellt.

Die Funktionen sollen wie folgt lauten:

$$f_1(t) = sin(\omega t) \tag{7.14}$$

$$f_2(t) = sin\left(\omega t - \frac{\pi}{3} + F_2(p)\right) \tag{7.15}$$

$$f_3(t) = sin\left(\omega t - \frac{2 \cdot \pi}{3} + F_3(p)\right) \tag{7.16}$$

Für ein lineares Mikrofonarray lauten die Funktionen $\mathcal{F}_n(p)$:

$$\mathcal{F}_1(p) = 0 \cdot p \tag{7.17}$$

$$\mathcal{F}_2(p) = 1 \cdot p \tag{7.18}$$

$$\mathcal{F}_3(p) = 2 \cdot p \tag{7.19}$$

Die Korrelationsmatrix wird an der Stelle $\tau = 0$ gebildet und lautet somit für den vorliegenden Fall wie folgt:

$$\begin{aligned} \boldsymbol{R}_a(p) &= \frac{1}{2} \cdot \begin{pmatrix} 1 & cos\left(\frac{\pi}{3} - p\right) & cos\left(\frac{2\cdot\pi}{3} - 2 \cdot p\right) \\ cos\left(-\frac{\pi}{3} + p\right) & 1 & cos\left(-\frac{\pi}{3} + \frac{2\cdot\pi}{3} + p - 2 \cdot p\right) \\ cos\left(-\frac{2\cdot\pi}{3} + 2 \cdot p\right) & cos\left(-\frac{2\cdot\pi}{3} + \frac{\pi}{3} + 2 \cdot p - p\right) & 1 \end{pmatrix} \\ &= \frac{1}{2} \cdot \begin{pmatrix} 1 & cos\left(\frac{\pi}{3} - p\right) & cos\left(\frac{2\cdot\pi}{3} - 2 \cdot p\right) \\ cos\left(-\frac{\pi}{3} + p\right) & 1 & cos\left(\frac{\pi}{3} - p\right) \\ cos\left(-\frac{2\cdot\pi}{3} + 2 \cdot p\right) & cos\left(-\frac{\pi}{3} + p\right) & 1 \end{pmatrix} \end{aligned} \tag{7.20}$$

Wird nun für p der Wert Null eingesetzt, ergibt sich folgende Korrelationsmatrix:

$$\boldsymbol{R}_a(p = 0) = \frac{1}{2} \cdot \begin{pmatrix} 1 & cos\left(\frac{\pi}{3}\right) & cos\left(\frac{2\cdot\pi}{3}\right) \\ cos\left(-\frac{\pi}{3}\right) & 1 & cos\left(\frac{\pi}{3}\right) \\ cos\left(-\frac{2\cdot\pi}{3}\right) & cos\left(-\frac{\pi}{3}\right) & 1 \end{pmatrix} \tag{7.21}$$

Mit der dazugehörigen Determinante:

$$
\begin{aligned}
&2 \cdot det(\boldsymbol{R}_a(0)) = \\
&1 + cos\left(-\frac{\pi}{3}\right) \cdot cos\left(-\frac{\pi}{3}\right) \cdot cos\left(\frac{2\dot{\pi}}{3}\right) + cos\left(\frac{\pi}{3}\right) \cdot cos\left(\frac{\pi}{3}\right) \cdot cos\left(-\frac{2\dot{\pi}}{3}\right) \\
&\quad - \left[cos\left(-\frac{2\dot{\pi}}{3}\right) \cdot cos\left(\frac{2\dot{\pi}}{3}\right) + cos\left(\frac{\pi}{3}\right) \cdot cos\left(-\frac{\pi}{3}\right) + cos\left(\frac{\pi}{3}\right) \cdot cos\left(-\frac{\pi}{3}\right)\right] \\
&= 1 - \frac{1}{8} - \frac{1}{8} - \frac{1}{4} - \frac{1}{4} - \frac{1}{4} \\
&= 0
\end{aligned}
\tag{7.22}
$$

Die Korrelationsmatrix an der Stelle $p = \frac{\pi}{3}$ lautet wie folgt:

$$
\begin{aligned}
\boldsymbol{R}(p = \frac{\pi}{3}) &= \frac{1}{2} \cdot \begin{pmatrix} 1 & cos\,(0) & cos\,(0) \\ cos\,(0) & 1 & cos\,(0) \\ cos\,(0) & cos\,(0) & 1 \end{pmatrix} \\
&= \frac{1}{2} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}
\end{aligned}
\tag{7.23}
$$

Mit der zugehörigen Determinante :

$$
\begin{aligned}
2 \cdot det(\boldsymbol{R}(0)) &= 1 + 1 + 1 - 1 - 1 - 1 \\
&= 0
\end{aligned}
\tag{7.24}
$$

Es zeigt sich somit, dass die Determinante an der Stelle $p = 0$ im Fall nicht verrauschter Signale immer zu null wird. Warum der Algorithmus bei verrauschten Signalen trotzdem funktionieren kann, ist in der Art der linearen Abhängigkeit von $\boldsymbol{R}_a(p)$ begründet. Dazu muss zunächst veranschaulicht werden, welche Bedeutung die Determinante einer Matrix hat. Für den hier exemplarisch diskutierten Fall ist der Wert der Determinante das sogenannte Spatprodukt. Dieses gibt das Volumen des in der Matrix enthaltenen Spats an. Folglich muss das Volumen hier in beiden Fällen gleich null sein. Der Unterschied ist aber, dass die Vektoren im Falle $\boldsymbol{R}_a(p = \frac{\pi}{3})$ alle identisch sind. Im Falle von $\boldsymbol{R}_a(p = 0)$ liegen die Vektoren zwar so, dass von ihnen kein Volumen jedoch eine Fläche aufgespannt wird. In diesem Fall sind die Vektoren zusammen linear abhängig. Das bedeutet, dass sich einer durch die

anderen beiden darstellen lässt. Die folgenden Abbildungen zeigen die Vektoren des vorgestellten Falls:

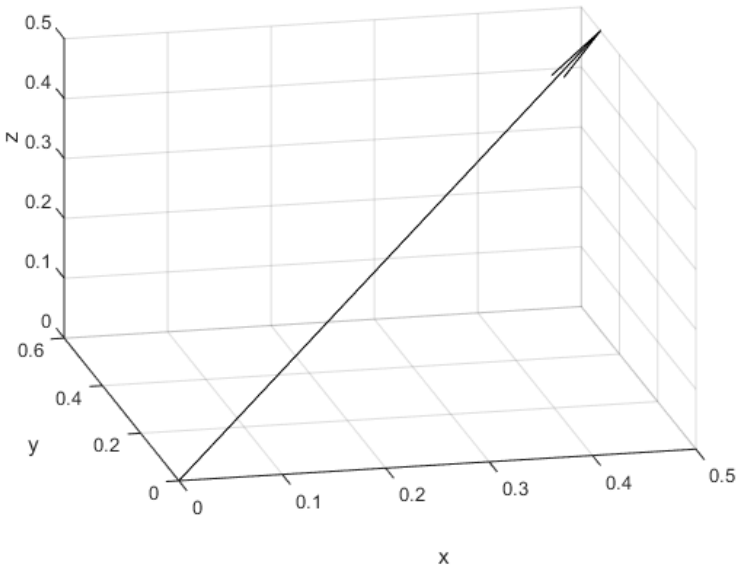


Abbildung 7.11.: Vektoren der Matrix $\boldsymbol{R}_a(p = \frac{\pi}{3})$

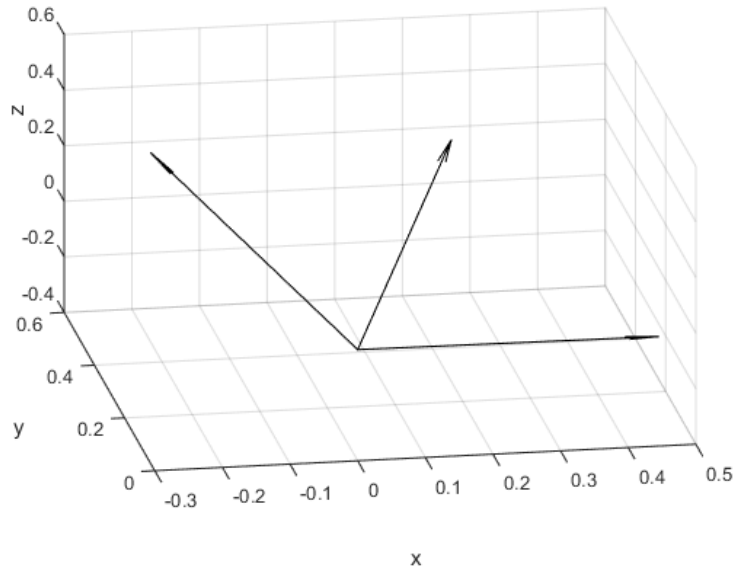


Abbildung 7.12.: Vektoren der Matrix $\boldsymbol{R}_a(p = 0)$

Im Fall $\boldsymbol{R}_a(p = 0)$ ist - wie schon beschrieben - eine lineare Abhängigkeit vorhanden,

die alle Vektoren einbezieht. Sobald ein Parameter geändert wird, wächst das aufgespannte Volumen rapide. Im Falle von $\boldsymbol{R}_a(p = \frac{\pi}{3})$ macht die Änderung eines Parameters nichts aus. Es müssen für die gleiche Änderung des Volumens schon deutlich stärkere Änderungen der Parameter vorgenommen werden. Diese Parameter werden durch das Rauschen beeinflusst. Dieses hat demnach einen deutlich stärkeren Einfluss auf $det\left(\boldsymbol{R}_a(p = 0)\right)$ als auf $det\left(\boldsymbol{R}_a(p = \frac{\pi}{3})\right)$.

Im nächsten Schritt wurden die Korrelationsfunktionen betrachtet. Dabei fiel auf, dass die Werte der Amplitudenmaxima der einzelnen Korrelationsfunktionen deutliche Abweichungen voneinander aufweisen. Da es sich bei den Signalen der Mikrofone im rauschfreiem Fall lediglich um phasenverschobene identische Signale handelt, müssten die Korrelationen ebenfalls bis auf eine Phasenverschiebung identisch sein. Dass sie es nicht sind, liegt daran, dass für die Bildung der Korrelationsfunktionen lediglich ein Ausschnitt der Signale herangezogen wird. Wenn ein diskretes Signal der Länge K mit sich selbst korreliert wird, hat die entstehende Funktion ein Maximum bei τ = 0. Wird ein diskretes Signal der Länge K mit demselben um p verschobenen Signal korreliert, hat die Korrelationsfunktion ein Maximum bei $\tau = p$. Allerdings wird das Maximum vom Wert her kleiner als das der Autokorrelationsfunktion. Der Grund liegt in der endlichen Länge der betrachteten Signalausschnitte. Bei der Korrelation wird davon ausgegangen, dass alle Werte links und rechts des Zeitfensters null sind. Daher finden an der Stelle $\tau = p$ mehr Multiplikationen mit 0 statt, je größer die Zeitverschiebung des Signals ist. Deutlich wird dieser Zusammenhang bei der Betrachtung der Faltung zweier identischer Rechtecke. Es entsteht dabei bekanntermaßen ein Dreieck mit doppelter Länge. Bei der Faltung zweier Signale endlicher Länge findet ebenso eine Faltung des Fensters statt, was bedeutet, dass das Ergebnissignal mit dem aus der Faltung der Rechteckfenster entstehenden Dreieck gewichtet ist. Die folgende Abbildung zeigt die Korrelationsfunktionen des diskutieren Fall.

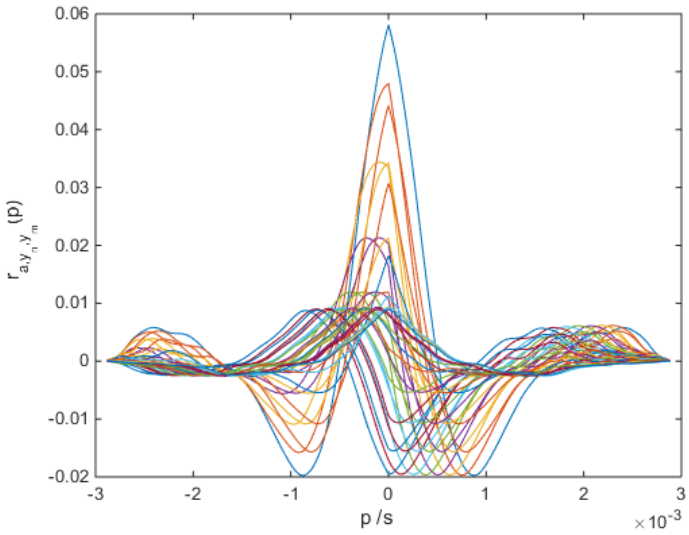


Abbildung 7.13.: Korrelationsfunktionen bei fehlerhafter Winkeldetektion

Deutlich zu erkennen ist der Amplitudenunterschied der Maxima der verschiedenen Korrelationsfunktionen. Aus den beschriebenen Beobachtung folgt, dass dieser Amplitudenunterschied geringer werden müsste, wenn entweder die Abtastrate reduziert oder die Blocklänge erhöht wird.
Die folgende Abbildung zeigt die Korrelationsfunktionen mit der doppelten Blocklänge.

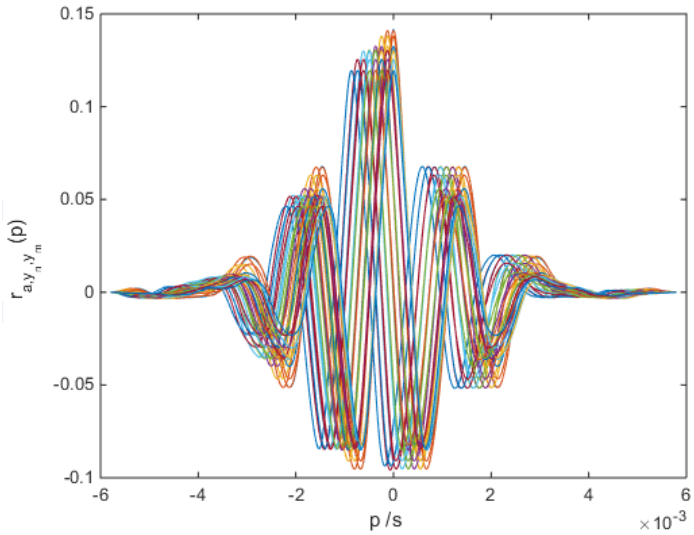


Abbildung 7.14.: Korrelationsfunktionen bei korrekter Winkeldetektion

Nun sind die Unterschiede der Maxima zueinander schon deutlich geringer.
Es zeigt sich, dass bei schrittweiser Erhöhung der Blocklänge der Wert der Determinante der Kovarianzmatrix ($\boldsymbol{R}_a(0)$) in derselben Größenordnung bleibt. Dies ist logisch, da lediglich die Anzahl der Samples zur Schätzung der Kovarianzmatrix erhöht wird. Jetzt lässt sich allerdings beobachten, dass der Wert der Determinante der Korrelationsmatrix an der Stelle $p = \tau$ kleiner wird und irgendwann den Wert der Kovarianzmatrix ($\boldsymbol{R}_a(0)$) unterschreitet. Der Grund hierfür ist, dass nun - wie zuvor beschrieben - geringere Unterschiede in den Amplituden der einzelnen Kreuzkorrelationen vorliegen und der durch die Blocklänge entstehende Fehler geringer wird.

7.4. Einfluss von Nachhall

7.4.1. Grundlegende Betrachtung von Nachhall

Algorithmen zur Sprecherlokalisierung finden oft in Räumen Anwendung, sodass Nachhall für diese relevant sein könnte. Nachhall ist die Reflexion von Schallwellen an Wänden und verschiedenen Gegenständen. Im Bezug auf die konkrete Anwendung bedeutet das, dass ein Signal nicht nur auf dem direkten, sondern auch auf verschiedenen indirekten Wegen am Mikrofonarray angelangt. Durch Absorption an den reflektierenden Oberflächen und durch Dämpfung in der Luft sind die Schallwellen, die über einen indirekten Pfad am Mikrofonarray angelangen, eine gedämpfte und zeitverzögerte Version der Schallwellen, die auf dem direkten Weg von der Quelle zum Mikrofonarray gelangt sind.
Je nach Art des Quellsignals und Blocklänge der Eingansdaten der Algorithmen bzw. Pfadlänge können hier verschiedene Effekte auftreten, welche zunächst unabhängig von den verschiedenen Verfahren betrachtet werden sollen. Dafür wird ein möglichst einfaches Szenario konstruiert, bei dem eine Schallquelle Schallwellen aussendet, die auf zwei Wegen zu einem Mikrofonarray gelangen. Ein Weg ist der direkte Pfad. Der andere Weg führt über eine Wand, an der die Schallwellen reflektiert werden zum Mikrofonarray.

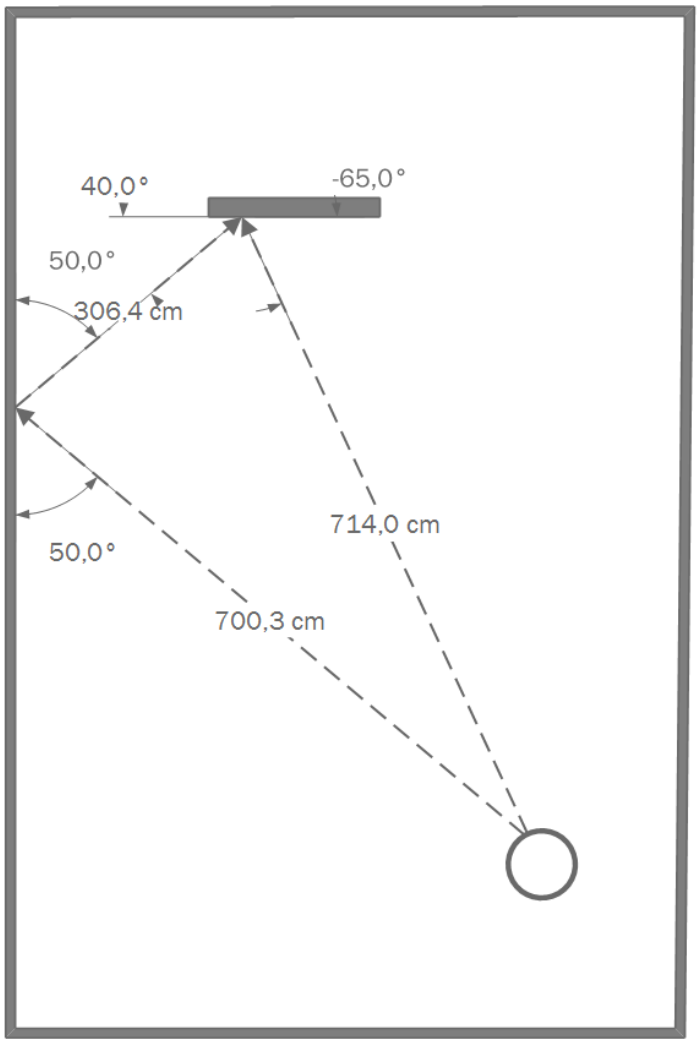


Abbildung 7.15.: Einfaches Multipfad-Szenario

Zunächst wird davon ausgegangen, dass die Wand ideal reflektiert und die Schallwellen ungedämpft zum Mikrofonarray gelangen. Die Geschwindigkeit, mit der sich die Schallwellen ausbreiten, beträgt $343,2\ m/s$. Die Zeit, welche die Schallwellen benötigen, um von der Quelle bis zum Mikrofonarray zu gelangen, lässt sich wie folgt berechnen:

$$Zeit = \frac{Weg}{Geschwindigkeit} \tag{7.25}$$

Somit ergeben sich für die beiden unterschiedlichen Pfade die in Tabelle 7.3 aufgeführten Laufzeiten.

Tabelle 7.3.: Zeitdauer bis zur Ankunft von Schallwellen bei verschiedenen Pfaden

Pfad	**Abstand /m**	**Laufzeit /ms**
direkt	7,14	20,80
indirekt	10,067	29,33

Die Zeitdifferenz der Ankunft der Schallwellen beträgt somit $8.53\ ms$. Angenommen die Quelle gibt lediglich einen konstanten Ton aus, der sich zum Beispiel anhand einer Sinusfunktion beschreiben lässt, denn würde sich das am ersten Mikrofon des Mikrofonarrays empfangende Signal wie folgt beschreiben lassen:

$$\begin{aligned} y_1(t) &= sin(2\pi \cdot f \cdot (t - \tau_1)) + sin(2\pi \cdot f \cdot (t - \tau_2)) \\ &= 2 \cdot cos(\pi \cdot f \cdot (\tau_2 - \tau_1)) \cdot sin\left(2\pi \cdot f \cdot \left(t - \frac{\tau_1 + \tau_2}{2}\right)\right) \end{aligned} \tag{7.26}$$

Um das Signal am zweiten Mikrofon zu beschreiben, muss zu den jeweiligen Verzögerungen eine vom Winkel abhängige Verzögerung addiert werden.

$$\begin{aligned} y_2(t) = 2 \cdot cos\left(\pi \cdot f \cdot \left(\left(\tau_2 + \frac{d}{v} \cdot sin(\phi_2)\right) - \left(\tau_1 + \frac{d}{v} \cdot sin(\phi_1)\right)\right)\right) \\ \cdot sin\left(2\pi \cdot f \cdot \left(t - \frac{\left(\tau_1 + \frac{d}{v} \cdot sin(\phi_1)\right) + \left(\tau_2 + \frac{d}{v} \cdot sin(\phi_2)\right)}{2}\right)\right) \end{aligned} \tag{7.27}$$

Würde das Signal lediglich über den direkten Weg empfangen werden, wäre die Zeit, die die Schallwellen von der Quelle bis zum zweiten Mikrofon benötigt:

$$\tau_{Quelle->Mic2} = \tau_1 + \frac{d}{v} \cdot sin(\phi_1) = 20,8ms + (-0,062ms) \tag{7.28}$$

Der Abstand zwischen den Mikrofonen beträgt 5 cm. Die Zeitverzögerung der Schallwellen auf dem direkten Pfad beträgt hier 0,062 ms. Für den indirekten Pfad ergibt sich folgende Zeitverzögerung:

$$\Delta\tau_{indirekt} = \frac{d}{v} \cdot sin(\phi_2) = 0,112ms \tag{7.29}$$

Bei zwei Pfaden lautet das am zweiten Mikrofon empfangende Signal für den konkreten

Fall:

$$\begin{aligned} y_2(t) &= 2 \cdot cos\left(\pi \cdot f \cdot ((29,33ms + 0,112ms) - (20,80ms - 0,062ms))\right) \\ &\quad \cdot sin\left(2\pi \cdot f \cdot \left(t - \frac{(20,80ms - 0,062ms\cdot) + (29,33ms + 0,112ms)}{2}\right)\right) \\ &= 2 \cdot cos\left(\pi \cdot f \cdot (8,704ms)\right) \cdot sin\left(2\pi \cdot f \cdot (t - 25,09ms)\right) \end{aligned} \tag{7.30}$$

Wohingegen das am ersten Mikrofon empfangende Signal

$$\begin{aligned} y_1(t) &= 2 \cdot cos(\pi \cdot f \cdot (29,33ms - 20,80ms)) \\ &\quad \cdot sin\left(2\pi \cdot f \cdot \left(t - \frac{20,80ms + 29,33ms}{2}\right)\right) \\ &= 2 \cdot cos(\pi \cdot f \cdot (8,53ms)) \cdot sin\left(2\pi \cdot f \cdot (t - 25,065)\right) \end{aligned} \tag{7.31}$$

lautet. Die Zeitverzögerung der beiden Signale beträgt somit:

$$\Delta\tau_{Mic1,Mic2} = 25,09ms - 25,065ms = -0,025ms \tag{7.32}$$

Dieser Wert entspricht einem Winkel von $9,88°$. Des Weiteren wurde die Amplitude ebenfalls beeinflusst.
In der Realität kann davon ausgegangen werden, dass die Schallwellen, welche auf dem direkten Weg zum Mikrofonarray gelangen, am wenigsten gedämpft sind und die Schallwellen, die eine längere Wegstecke zurücklegen müssen, je nach Strecke und Anzahl der Reflexionen sowie Oberfläche, an der sie reflektiert wurden, stärker gedämpft sind.
In so einem Fall lässt sich die resultierende Schallwelle wie folgt berechnen [17, S. 120]:

$$\begin{aligned} y(t) &= \hat{y_1} \cdot sin(\omega t + \phi_1) + \hat{y_2} \cdot sin(\omega t + \phi_2) \\ &= \hat{y} \cdot sin(\omega t + \phi) \end{aligned} \tag{7.33}$$

mit

$$\hat{y} = \sqrt{\hat{y_1}^2 + 2\hat{y_1}\hat{y_2}cos(\phi_1 - \phi_2) + \hat{y_2}} \tag{7.34}$$

und

$$tan(\phi) = \frac{\hat{y_1} sin(\phi_1) + \hat{y_2} sin(\phi_2)}{\hat{y_1} cos(\phi_1) + \hat{y_2} cos(\phi_2)} \tag{7.35}$$

Die Dämpfung bei einer Reflexion lässt sich für einen einfachen Fall ebenfalls angeben. Ein Einfluss ist der Absorptionsgrad des Materials, an dem die Schallwellen reflektiert werden. Dieser beträgt bei einer 15 cm starken Betonwand etwa 0,02 [43, S.140] und ist außerdem leicht frequenzabhängig. Das bedeutet, dass 98% der Leistung reflektiert werden. Eine weitere wesentliche Rolle spielt die Dämpfung des Schallpegels in Abhängigkeit der Entfernung. Der Schalldruck nimmt mit der Entfernung im Verhältnis $1/r$ ab. Dies soll nun für das hier verwendete Szenario verdeutlicht werden. Angenommen der Schalldruckpegel von menschlicher Sprache beträgt im Abstand von einem cm gemessen von der Quelle 96 dB, denn lässt sich der Schalldruck an einem zweitem Messpunkt wie folgt berechnen [14, S. 439]:

$$L_{s,2} = L_{s,1} - 20 \cdot log\left(\frac{r_2}{r_1}\right) \tag{7.36}$$

Gelangen die Schallwellen auf dem direkten Pfad zum Mikrofonarray, so beträgt der Schalldruck am ersten Mikrofon 38,93 dB. Der Schalldruck, der in die Wand einfallenden Welle beträgt 39,11 dB. Der der ausfallenden Welle beträgt 38,33 dB. Nachdem die Schallwelle sich weitere 306,4 cm durch den Raum bewegt hat, beträgt der Schalldruck am ersten Mikrofon 35,18 dB. Das bedeutet, dass die Amplitude der reflektierten Welle etwa um den Faktor 0,65 im Bezug auf die Amplitude des direkten Pfads gedämpft ist. Es ist somit mit nicht unerheblichen Interferenzen zu rechnen. Wie stark die über die verschiedenen Pfade eingehenden Signale interferieren können, lässt sich durch die Kohärenz bestimmen. Die Kohärenz zweier Signale wird wie folgt berechnet [10]:

$$\gamma_{xy}(f) = \frac{G_{xy}(f)}{\sqrt{G_{xx}(f) G_{yy}(f)}} \tag{7.37}$$

Darin ist $G_{xy}(f)$ das Kreuzleistungsspektrum mit

$$G_{xy}(f) = \int_{-\infty}^{\infty} r_{xy}(\tau) \mathrm{e}^{j2\pi\tau} d\tau \tag{7.38}$$

Schallwellen, die über verschiedene Pfade am Mikrofonarray ankommen, sind, sofern die

Phasenlage konstant ist, sehr stark kohärent. Allerdings wird in der konkreten Anwendung jeweils mit Blöcken gearbeitet, wodurch die Stärke der Kohärenz von zwei Faktoren abhängt:

Blocklänge in Kombination mit Abtastrate Angenommen die Blocklänge eines Systems beträgt 512 Samples und es wird mit 48 kHz abgetastet. Ein Block wäre in diesem System 10,67 ms lang. Das ist lediglich 2,14 ms länger als der Laufzeitunterschied, der im Beispiel betrachteten Pfade. Handelt es sich bei der Quelle um ein nicht periodisches Signal, ist der korrelierte Teil in einem Block bei ca. 25%. Bei einer Abtastrate von 8 kHz beträgt eine Blocklänge 64 ms. Mit der Laufzeitdifferenz zwischen den beiden Pfaden von 8,53 ms beträgt der Anteil korrelierter Signale 86,67 %.

Periodizität des Quellsignals Handelt es sich bei dem Quellsignal um ein periodisches Signal, befinden sich unabhängig von der Blocklänge korrelierte Anteile, welche über die verschiedenen Pfade eintreffen, in den Signalblöcken.

7.4.2. Einfluss von Nachhall auf den MUSIC-Algorithmus

Der MUSIC-Algorithmus ist in der Lage mehrere Quellen zu detektieren, sodass ein Signal, welches über einen anderen Pfad am Mikrofonarray angelangt, theoretisch als solches erkannt werden kann. Dies hängt stark von der Kohärenz ab. Sind die Signale stark kohärent, ist es kaum noch möglich, die jeweiligen Signal-Eigenvektoren zu finden, da die zugehörigen Eigenwerte sich betragsmäßig nicht stark genug von den Rausch-Eigenwerten unterscheiden. Wenn a priori bekannt wäre, wie viele Signale aus verschiedenen Richtungen vorhanden sind, kann eine korrekte Trennung von Signal- und Rauschunterraum erfolgen. Da dies gerade bei Multipfad-Szenarien nicht der Fall ist, muss ein Schwellwert für die Zuordnung der Eigenwerte definiert werden. Eine Vermischung von Signal- und Rauschunterraum ist somit häufig die Folge, was zu einer deutliche Verschlechterung der Ergebnisse bis hin zum totalen Versagen des Algorithmus führen kann.
Die folgende Abbildung zeigt die MUSIC-Spektren für vier verschiedene Frequenzen sowie das kombinierte Breitband-MUSIC-Spektrum in Schwarz für das zuvor beschriebene Szenario mit einer Quelle und zwei Pfaden.

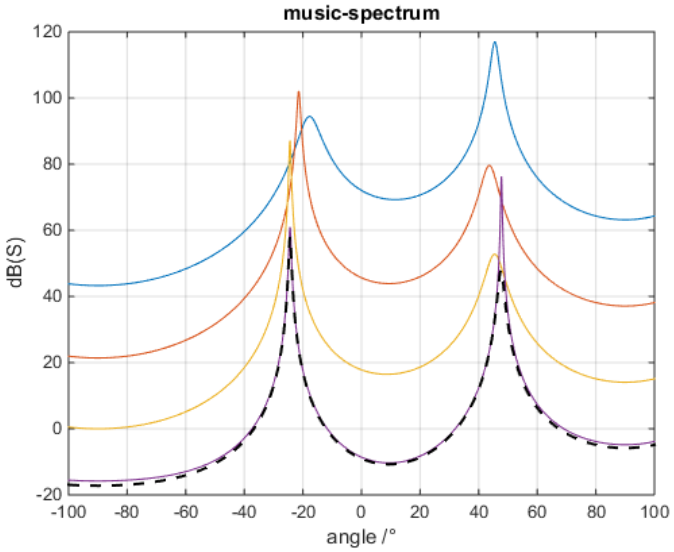


Abbildung 7.16.: MUSIC-Spektrum im Multipfad-Szenario bei der Annahme, dass zwei Quellen existieren

Wird nun davon ausgegangen, dass lediglich eine Quelle existiert, so wird der zweite Signal-Eigenvektor dem Rauschunterraum zugeordnet. Das Ergebnis ist in der folgenden Abbildung zu sehen.

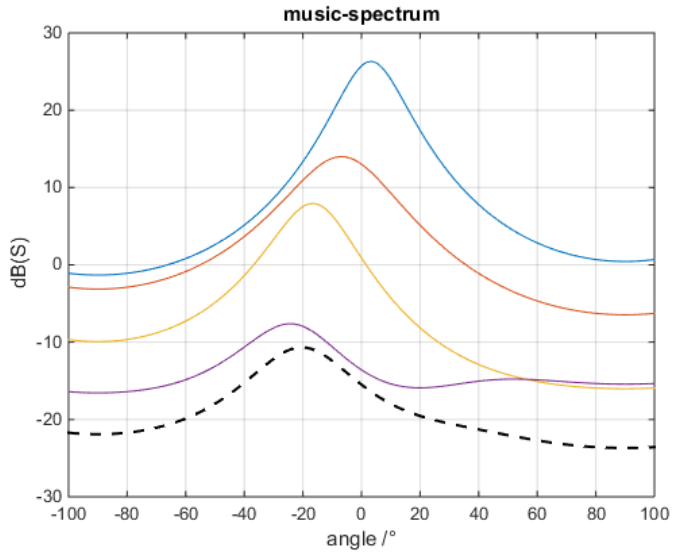


Abbildung 7.17.: MUSIC-Spektrum im Multipfad-Szenario bei der Annahme, dass eine Quelle existiert

Es ist festzustellen, dass sämtliche MUSIC-Spektren für die einzelnen Frequenzen deutlich schlechter definiert sind und teilweise stark vom korrekten Wert abweichen.
Bei noch mehr Pfaden in einer verhallten Umgebung ist von gravierenderen Einflüssen auszugehen.

7.4.3. Einfluss von Nachhall auf den MCCC-Algorithmus

Genau wie MUSIC ist auch der MCCC-Algorithmus in der Lage mehrere Quellen gleichzeitig zu lokalisieren [5, S 21].
Wird der Algorithmus auf das zuvor beschriebene Szenario angewendet, ergibt sich der folgende Verlauf für $det(\boldsymbol{R}_a(p))$.

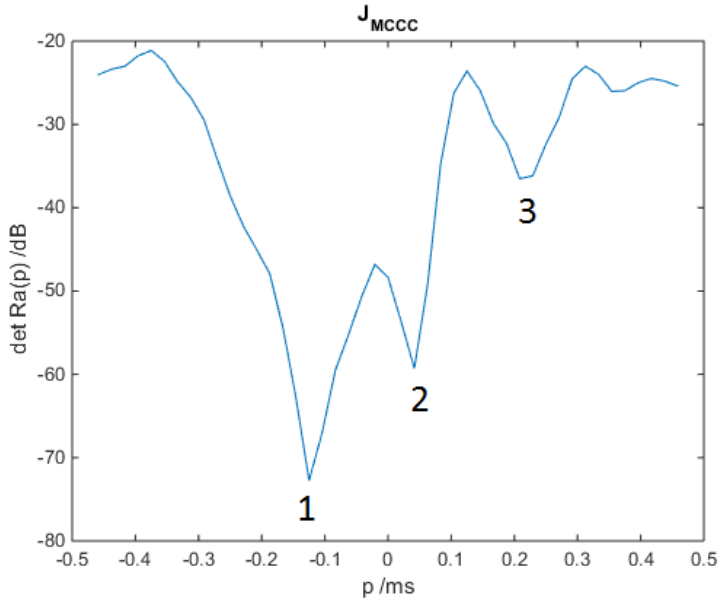


Abbildung 7.18.: Verlauf der Fehlerfunktion des MCCC-Algorithmus im Multipfad-Szenario

Bei der Annahme, dass lediglich eine Quelle vorhanden ist, liefert der Algorithmus ein korrektes Ergebnis (Minimum Nr. 1). Das über den zweiten Pfad empfangende Signal erzeugt das Minimum Nr. 3. Das betragsmäßig zweit größte Minimum gehört zu keiner Quelle. Dieses entsteht durch Fehlereinflüsse bedingt durch die Fensterung und durch periodische Wie-

derholung der Maxima der Korrelationsfunktionen, sodass bei der gegebenen Konstellation eine weitere Stelle entstanden ist, an der die Determinante minimal wird. Um tatsächlich mehrere Quellen sicher zu orten, könnte zum Beispiel eine bessere Frequenzauflösung oder Verfahren zur Signalvorverarbeitung, wie beispielsweise Prewhitening angewendet werden, sodass der gezeigte Effekt nicht mehr eintritt.
Für den an dieser Stelle thematisierten Einfluss von Nachhall bei einer Quelle lässt sich jedoch feststellen, dass der Algorithmus für das exemplarisch betrachtete Szenario ein korrektes Ergebnis liefert.

8. Gegenüberstellung der Verfahren

In den vorherigen Kapiteln wurden die verschiedenen Verfahren zunächst theoretisch betrachtet und simuliert. Anschließend wurden sie auf einer Hardware-Plattform implementiert und getestet. Außerdem wurden verschiedene andere Untersuchungen durchgeführt. In diesem Kapitel soll basierend auf den Ergebnissen der bisherigen Arbeit eine Gegenüberstellung bzw. ein Vergleich der beiden verschiedenen Ansätze stattfinden, sodass schlussendlich beurteilt werden kann, welcher Ansatz sich besser zur Sprecherlokalisierung eignet oder ob möglicherweise eine derartige Aussage gar nicht möglich ist und eine genauere Betrachtung der Anwendungsfälle notwendig ist.

8.1. Gegenüberstellung der theoretischen Ansätze

Die theoretischen Ansätze geben jeweils Aufschluss über die Grenzen und Komplexität der Methoden. Die Komplexität hat hierbei zweierlei Auswirkungen. Zum einen lässt sich ein Verfahren, welches leicht nachvollziehbar ist, leichter implementieren und ist somit schneller einsatzbereit. Auch mögliche Fehler lassen sich bei einem einfacheren Verfahren für gewöhnlich schneller finden. Zum anderen kann die Komplexität eine Auswirkung auf die benötigte Rechenzeit haben. Die Gegenüberstellung des Ressourcenaufwands findet allerdings an einer anderen Stelle statt. Die theoretischen Ansätze können außerdem Aufschluss über die maximale Performance geben.
Die zeitbasierten Verfahren beruhen darauf, die Zeitverschiebung zwischen zwei oder mehr Eingangssignalen zu bestimmen. Dafür werden diese miteinander korreliert und anhand der Korrelationsfunktionen auf verschiedene Weisen eine mögliche Zeitverschiebung ermittelt. Beim MCCC, der stellvertretend für die zeitbasierten Verfahren genauer untersucht wurde, werden diese Funktionen zum Beispiel genutzt, indem eine parametrierbare Korrelationsmatrix in Abhängigkeit einer angenommenen Zeitverschiebung erstellt wird, von welcher

jeweils die Determinante bestimmt werden muss. Unterraumbasierte Verfahren arbeiten mit der Trennung von Signal- und Rauschunterraum. Diese haben die Eigenschaft orthogonal zueinander zu sein. Der Signalunterraum der Kovarianzmatrix im Frequenzbereich enthält die Information der Verschiebung der Signale zueinander. Für die Gruppe der unterraumbasierten Verfahren wurde der MUSIC-Algorithmus näher untersucht. Dieser arbeitet mit dem Rauschunterraum, indem ein orthogonaler Vektor gefunden wird, welcher anhand der Arraygeometrie und einem angenommenen Winkel berechnet wird. Die maximal mögliche Performance bzw. Auflösung ist, wie bereits in Abschnitt 7.1 beschrieben, nur durch statistische Einflüsse beschränkt, wohingegen diese bei zeitbasierten Verfahren hauptsächlich von der Abtastrate und dem Mikrofonabstand abhängt. Für die gängige Abtastrate von 48 kHz wird bei einem Mikrofonabstand von 5 cm lediglich eine maximale Auflösung von 8,2215° erreicht. Die unterraumbasierten Verfahren sind in der Praxis trotz der möglichen unendlich feinen Auflösung durch zwei Faktoren begrenzt. Der erste Faktor ist die Genauigkeit. Verschiedene Störeinflüsse sorgen dafür, dass die Verfahren ungenaue Ergebnisse liefern. Dies macht sich anhand einer Varianz der Ergebnisse bei einer statischen Quelle bemerkbar. Wenn das Ergebnis bei einer statischen Quelle selbst mit entsprechender Signalvor- und Nachverarbeitung innerhalb eines gewissen Bereichs variiert, ist es nicht sinnvoll, eine deutlich feinere Auflösung zu wählen. Der zweite Faktor ist der Rechenaufwand. Je feiner die Auflösung, desto höher ist das Maß an benötigter Rechenzeit. Diese ist gerade in Echtzeitsystemen oftmals stark begrenzt, sodass es sinnvoll ist, eine nicht unnötig hohe Auflösung zu wählen. Dennoch lässt sich sagen, dass unterraumbasierte Verfahren, sofern keine sehr hohe Abtastrate gewählt wurde bzw. realisierbar ist, eine höhere Auflösung erzielen.

8.2. Gegenüberstellung der Integrierbarkeit

In vielen Anwendungen werden Algorithmen zur Sprecherlokalisierung nicht allein implementiert, sondern sind Teil eines Gesamtsystems. Ein mögliches Beispiel hierfür ist eine Freisprecheinrichtung, welche über mehrere Mikrofone verfügt. So kann der Sprecher mittels Lokalisierungs-Algorithmus ausfindig gemacht werden und ein Beamformer jeweils auf diesen ausgerichtet werden.

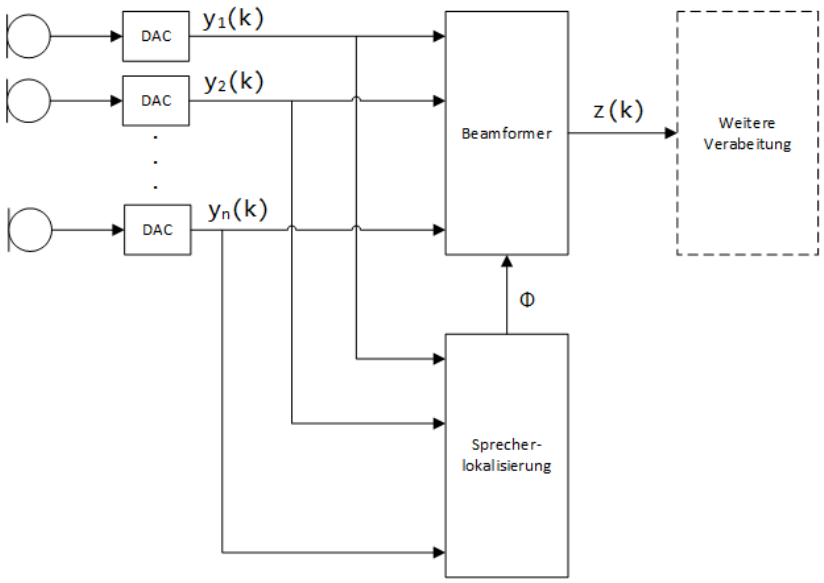


Abbildung 8.1.: Freisprecheinrichtung mit Mikrofonarray

Eine wichtige Anforderung in solchen Systemen ist die Flexibilität der Algorithmen. In dem vorliegenden Beispiel sind die Eingangssignale von dem Lokalisierungsalgorithmus und dem Beamformer dieselben. Das Ausgangssignal des Beamformers wird zur weiteren Verarbeitung verwendet. Es wäre somit praktisch, wenn alle Teilsysteme dieselbe Abtastrate nutzen würden. Da die Abtastrate bei zeitbasierten Verfahren bekanntermaßen in einem direkten Zusammenhang mit der Auflösung steht, muss im Einzelfall geprüft werden, ob die Auflösung bei gegebener Abtastrate und Arraygeometrie den Anforderungen genügt. Ist dies nicht der Fall, kann durch die Verwendung von Multiratensystemen[4] Abhilfe geschaffen werden. Diese sorgen allerdings für eine höhere Komplexität des Gesamtsystems. Unterraumbasierte Verfahren sind nicht auf eine bestimmte Abtastrate angewiesen. In der vorliegenden Arbeit wurde bei der Implementierung auf dem DSP mit einer SVD gearbeitet. Hierfür wurde jeweils der Bereich des Spektrums um die jeweilige Testfrequenz verwendet. Hier ist eine möglichst hohe Frequenzauflösung der FFT sinnvoll, welche entweder durch eine niedrigere Abtastrate erzielt wird oder durch eine größere Blocklänge. Die Frequenzauflösung hat jedoch keinen Einfluss auf die theoretische Winkelauflösung, sondern sorgt für eine genauere Repräsentation des Spektrums an der betrachteten Frequenzstützstelle und verbessert somit die Genauigkeit.

[4]Multiratensysteme sind digitale Systeme, in denen innerhalb der Verarbeitungskette unterschiedliche Abtastraten zum Einsatz kommen. Diese werden durch digitale Abtastratenumsetzer erzielt.

Wie in dem konkreten Beispiel muss für jede Anwendung geprüft werden, ob eine Verwendung von zeitbasierten Systemen bezüglich der Winkelauflösung möglich ist. Unterraumbasierte Verfahren bieten mehr Freiheitsgrade, da die Winkelauflösung nicht direkt von der Abtastrate abhängt. Eine Verbesserung der Frequenzauflösung lässt sich auf verschiedene Arten realisieren. Entweder werden längere Signalblöcke verwendet, eine niedrigere Abtastrate eingesetzt oder die Länge der FFT mittels Zeropadding vergrößert.

8.3. Gegenüberstellung von Schwachstellen und Störanfälligkeit

Ein wesentlicher Aspekt bei der Wahl eines Algorithmus ist dessen Anfälligkeit gegenüber Störungen. Bei den hier betrachteten Algorithmen ist dabei die einflussreichste Störungsart der Nachhall, welcher in Abschnitt 7.4 behandelt wurde. Dabei wurde festgestellt, dass der MUSIC-Algorithmus als Vertreter für unterraumbasierte Verfahren empfindlich auf Nachhall reagiert. Dies liegt daran, dass aufgrund der Kohärenz des Nutzsignals, mit dem auf verschiedenen Wegen reflektierten Signal, Rausch- und Signalunterraum nur noch schwer voneinander zu trennen sind. Es gibt allerdings Methoden der Signalvorverarbeitung, welche dazu da sind, dieses Problem zu umgehen. In der vorliegenden Arbeit wurden die Algorithmen jedoch ohne individuelle Vorverarbeitung betrachtet, sodass diese Verfahren nicht implementiert wurden. Der MCCC als Vertreter von den zeitbasierten Verfahren ist hingegen weniger empfindlich gegenüber Nachhall. Allerdings ist dieser sehr empfindlich bei einer zu geringen Blocklänge in Verbindung mit einem geringen Mikrofonrauschen. Hier kann es passieren, dass eine Fehldetektion bei 0° stattfindet, da die Korrelationsmatrix an dieser Stelle eine lineare Abhängigkeit aufweist, sofern alle Mikrofone rauschfrei sind. An dieser Stelle geht die Determinante der Korrelationsmatrix ebenfalls gegen Null, was der gesuchten Bedingung entspricht. Die Blocklänge spielt in diesem Fall ebenfalls eine Rolle, denn in der Realität ist immer ein Rauschen vorhanden. Jedoch sorgt das Fenster, welches bei der Korrelation durch die Blocklänge definiert wird, dafür, dass der durch die Vektoren der Korrelationsmatrix aufgespannte Raum an der Stelle, an der die angenommene Zeitverschiebung Null beträgt, geringer ist als an der Stelle, an der die angenommene Zeitverschiebung am dichtesten an der tatsächlichen Zeitverschiebung liegt. Auch hierfür existieren Methoden der Signalvorverarbeitung, die diese Fehler unterbinden. Auch eine entsprechend

größere Blocklänge oder das Herausrechnen des Einflusses des Fensters können Abhilfe schaffen.
Es wurde gezeigt, dass ein gewisses Maß an Rauschen für die Funktionalität des MCCC notwendig ist. Es lässt sich jedoch anhand der in dieser Arbeit durchgeführten MATLAB-Simulationen ebenfalls erkennen, dass genau wie bei MUSIC eine größer werdende Rauschamplitude zu schlechteren Ergebnissen führt.
Bei der Betrachtung der Störanfälligkeit wird der Empfindlichkeit gegenüber Nachhall die größte Bedeutung zugeschrieben, da sich ein großer Teil der möglichen Anwendungen in geschlossenen Räumen befindet. Hier ist der MCCC als Vertreter der zeitbasierten Verfahren weniger empfindlich als MUSIC als Vertreter der unterraumbasierten Verfahren.

8.4. Gegenüberstellung von Performance und des Ressourcenbedarfs

In Abschnitt 6 wurden die Algorithmen hinsichtlich des benötigten Rechenaufwands und des Speicherplatzbedarfs untersucht. Gerade der Bedarf an Rechenaufwand hat hierbei eine enorme Relevanz für die Verwendung der Algorithmen in Echtzeitsystemen. Häufig kommen hier eingebettete Systeme zum Einsatz, in denen oft vergleichsweise wenig interner Speicher zur Verfügung steht.
Da Rechenaufwand und Speicherplatz durch die Herangehensweise bei der Implementierung beeinflusst werden, können die hier ermittelten Ergebnisse lediglich als Tendenz gewertet werden. Die unterraumbasierten Verfahren benötigen eine Eigen- bzw. eine Singulärwertzerlegung, welche hauptsächlich von der Matrixgröße abhängt. Diese wird wiederum durch die Anzahl der Mikrofone definiert. Ein weiterer wichtiger Faktor ist die gewählte Auflösung. Beim MCCC wurde, um das zuvor dargestellte Problem der Fehldetektionen zu umgehen, eine größere Blocklänge gewählt. Dies sorgt sowohl für einen erhöhten Speicherplatzbedarf als auch für einen höheren Rechenaufwand. Bei Benutzung von Methoden, die eine geringere Blocklänge zulassen, könnte der benötigte Speicherplatz reduziert werden. Ob eine Reduktion der Rechenzeit erzielt wird, hängt von der durch die Methoden zusätzlich benötigten Rechenzeit ab. Bei den hier vorliegenden Implementierungen der Algorithmen ist die benötigte Rechenzeit des MCCC trotz der relativ großen Blocklänge deutlich geringer als die des MUSIC-Algorithmus. Ein weiterer Vorteil ist, dass die zeitbasierten

Verfahren den gesamten Frequenzbereich der Eingangssignale nutzen. Unterraumbasierte Verfahren arbeiten jeweils in einem schmalen Frequenzband. Hier wurde eine inkohärente Breitbandversion implementiert, die vier Frequenzstützstellen nutzt. Das Ergebnis ließe sich bei der Verwendung von noch mehr Stützstellen verbessern. Bei dem Speicherplatz ist der Bedarf bei MUSIC geringer als bei dem MCCC. Eine geringere Blocklänge würde zwar beim MCCC Abhilfe schaffen, jedoch sind je nach Anzahl der Mikrofone sehr viele KKF durchzuführen, deren Ergebnisse im Speicher zur Verfügung stehen müssen. Ein möglicher Ansatz, um den Speicherplatz zu reduzieren, ist es, die KKF auf konventionelle Art zu berechnen und nicht über den Ansatz der schnellen Faltung. So müssten von jeder KKF auch wirklich nur die für den Algorithmus relevanten Stellen berechnet und gespeichert werden. Dies hätte eine Minimierung des benötigten Speicherplatz zur Folge. Da - wie schon angemerkt - die jeweilige Implementierung ausschlaggebend für den Bedarf an Ressourcen ist, lässt sich nur eine Tendenz angeben. Die vorliegenden Messungen zeigen, dass die Rechenzeit für MUSIC im Breitbandfall deutlich länger ist als für den MCCC trotz einer großen Blocklänge und einer damit einhergehenden aufwendigen Berechnung der KKF. Auf der anderen Seite benötigt der MCCC mehr Speicherplatz, welcher allerdings durch verschiedene Ansätze reduziert werden kann.
Die Messungen im reflexionsarmen Schallmessraum zeigen, dass MUSIC wie erwartet eine deutlich feinere Auflösung ermöglicht als der MCCC. Es konnte ermittelt werden, dass die Standardabweichung bei MUSIC bei einer Sprachsequenz mit 0,69° größer ist als die des MCCC mit 1,07° $\cdot 10^{-12}$. Jedoch hängt dies stark davon ab, an welcher Stelle des Rasters sich der eingestellte Winkel gerade befindet. Ist ein Winkel genau zwischen zwei Positionen des Rasters, ist die Wahrscheinlichkeit für eine Detektion beider Rasterpositionen gleich groß. Eine höhere Standardabweichung wäre die Folge
Was Performance anbelangt, haben wiederum beide Algorithmen ihre Vorteile. Wenn es um den Ressourcenbedarf geht, ist der MCCC etwas im Vorteil, da dieser weniger Rechenzeit benötigt. Der benötigte Speicherplatz lässt sich durch verschiedene Maßnahmen deutlich reduzieren. Der Vorteil von MUSIC liegt wiederum darin, dass wie bereits aus dem theoretischen Ansatz deutlich wird, eine bessere Auflösung erzielt werden kann.

8.5. Gegenüberstellung der Fähigkeit, mehrere Quellen zu lokalisieren

Generell ist es mit beiden Methoden möglich, mehrere Quellen zu lokalisieren. Der MCCC basiert auf der parametrierbaren Korrelationsmatrix. Durch die verschiedenen Fehlereinflüsse, wie die Fensterung und periodische Wiederholungen der Maxima der Korrelationsfunktionen, können allerdings Fehler entstehen. MUSIC kann ebenfalls mehrere Quellen orten. Dabei ist die Anzahl der Quellen lediglich durch die Anzahl der Mikrofone begrenzt. Bei einem Mikrofonarray mit N Mikrofonen gilt für die Anzahl der erlaubten Quellen D:

$$D < N \tag{8.1}$$

Die Herausforderung hierbei ist es, die zu den Quellen gehörigen Eigenwerte ausfindig zu machen, um eine korrekte Trennung zwischen Signal- und Rauschunterraum vornehmen zu können. Handelt es sich bei den Quellen um vollständig unkorrelierte Signale, ist diese Trennung leicht möglich, da der Wertebereich der zu den Signalen gehörigen Eigenwerte deutlich höher liegt als der der Rauscheigenwerte. Bei teilweise korrelierten Signalen ist dies nicht der Fall. Eine klare Unterscheidung zwischen Raus- und Signaleigenwerten ist ohne Weiteres nicht mehr möglich. Allerdings existieren diverse Ansätze, um dieses Problem zu lösen. Zwei von diesen werden in dem folgenden Kapitel als Ausblick für kommende Arbeiten benannt.

8.6. Zusammenfassung und Beurteilung der Ergebnisse

In der vorliegenden Arbeit wurden zwei sehr unterschiedliche Methoden untersucht, welche beide dazu dienen Geräuschquellen zu lokalisieren. Beide Methoden haben Vor- und Nachteile gegenüber der jeweils anderen, sodass abschließend keine der beiden als besser angesehen werden kann. Vielmehr muss für jede Anwendung im Einzelfall geprüft werden, welche der Methoden sich besser eignet. Dafür ist es notwendig, verschiedene Anforderungen zu definieren und zu prüfen, welche Methode diese mit dem geringsten Aufwand erfüllt. Die Anforderungen lassen sich anhand folgender Punkte ausmachen:

- Benötigte Winkelauflösung

- Gegebenheiten der Umgebung
- Existierende Hardware

  Verwendete Abtastrate

  Rechenleistung

  Speicherplatz

Grundsätzlich lässt sich sagen, dass für ein System, welches mit einer relativ groben Winkelauflösung auskommt, ein zeitbasiertes Verfahren zu bevorzugen ist, sofern die gegebene Abtastrate und die gegebene Arraygeometrie diese Auflösung zulassen. Der Vorteil ist die Robustheit sowie die simplere Implementierung und die geringere Rechenzeit. Soll eine feinere Auflösung erzielt werden, ist es sinnvoll, ein unterraumbasiertes Verfahren einzusetzen. Diese lassen sich durch die größere Anzahl von Freiheitsgraden außerdem leichter in ein vorgegebenes System integrieren. Bei der Notwendigkeit mehrere Quellen gleichzeitig zu lokalisieren, empfiehlt es sich ebenfalls auf ein unterraumbasiertes Verfahren zurückzugreifen.

9. Zusammenfassung und Ausblick

Ziel der vorliegenden Arbeit war es, zwei verschiedene Ansätze zur Sprecherlokalisierung mit Mikrofonarrays gegenüberzustellen. Dazu wurden zunächst verschiedene Algorithmen, die jeweils auf einem dieser Ansätze beruhen, theoretisch betrachtet. Anschließend wurden die zuvor betrachteten Algorithmen in MATLAB simuliert und hinsichtlich verschiedener Gesichtspunkte untersucht. Dazu wurde Wert darauf gelegt die Simulationen realitätsnah zu gestalten. Daraufhin wurde jeweils ein zeit- und ein unterraumbasiertes Verfahren gewählt, um diese auf einem DSP-basierten Echtzeitsystem zu implementieren und weiter zu untersuchen. Neben einer Performance-Untersuchung bezüglich Rechengeschwindigkeit und Speicherbedarf wurden auch Messungen im reflexionsarmen Schallmessraum durchgeführt. Außerdem wurden die Algorithmen bezüglich ihrer theoretischen Genauigkeit, ihrer Empfindlichkeit gegenüber Nachhall und ihrer Schwachstellen untersucht. Zum Schluss wurden die Ergebnisse aus den vorangegangenen Abschnitten herangezogen, um einen Vergleich der beiden Methoden anzustellen.

Am Anfang der Arbeit war ein Ziel herauszufinden, ob eine Methode besser ist bzw. sich für den praktischen Einsatz in einem Echtzeitsystem eignet. Jedoch wurde schnell klar, dass diese Methoden viele verschiedene Eigenschaften aufweisen, die eine solch pauschale Aussage nicht zulassen. Vielmehr hat sich gezeigt, dass für jeden Einsatzzweck eine eigene Überprüfung stattfinden muss, um herauszufinden, welches Verfahren sich besser dafür eignet.

Für die durchgeführten Untersuchungen wurde ein Echtzeitsystem aufgebaut, welches sich von dem bis dato für Arbeiten in diesem Bereich verwendeten System unterscheidet. Dieses System bietet sehr viel Potenzial zur Weiterentwicklung der implementierten Verfahren. Aufgrund der begrenzten Zeit wurde sowohl der MCCC, als auch Breitband-MUSIC in einer einfachen Form ohne zusätzliche Signalvorverarbeitung bzw. Nachverarbeitung abgesehen von einem Medianfilter implementiert. Durch die leistungsstarke Hardware in Verbindung mit der entwickelten Schnittstelle und der grafischen Oberfläche auf einem PC ist es nun

möglich, die vorliegende Implementierung zu nutzen, um mögliche Methoden der besagten Signalvorverarbeitung zu ergänzen und so bessere Ergebnisse in verhallter Umgebung zu erzielen. Dies wäre vor allem bei den unterraumbasierten Verfahren von Interesse. In [7, S. 65ff.] werden dafür zwei verschiedene Methoden vorgestellt. Eine davon ist das so genannte Forward-Backward Averaging. Die andere Methode heißt Spatial Smoothing, welche sich besser für Multipfad-Szenarien eignet. Beide Methoden sind dazu da, die Signale, welche über verschiedenen Wege empfangen wurden, zu dekorrelieren, was letztendlich dafür sorgt, dass eine ordnungsgemäße Trennung von Signal- und Rauschunterraum vorgenommen werden kann.

Durch die leistungsstarke Hardware ist es auch möglich, das bestehende System für eine konkrete Anwendung zu erweitern. Denkbar wäre beispielsweise ein adaptiver Beamformer. Dabei käme ein Algorithmus zur Sprecherlokalisierung zum Einsatz, welcher als Ergebnis einen Winkel liefert. Darauf basierend werden Parameter für einen Beamformer berechnet, dessen Beampattern so ausgelegt ist, dass die größte Verstärkung in der Richtung des Sprechers liegt. So ein System könnte zum Beispiel für eine Freisprecheinrichtung genutzt werden.

Literaturverzeichnis

[1] ADVE, Raviraj: *Direction of Arrival Estimation*. – URL `http://www.comm.utoronto.ca/~rsadve/Notes/DOA.pdf`. – Zugriffsdatum: 07.04.2015

[2] ALIYAZICIOGLU, Zekeriya ; HWANG, H.K. ; GRICE, Marshall ; YAKOVLEV, Anatoly: Sensitivity Analysis for Direction of Arrival Estimation using a Root-MUSIC Algorithm. In: *Engineering Letters, VOL. 16, ISSUE 3* (2008)

[3] BAUMANN, Wolf: *Optimierung frequenzvarianter Nullbeamformer für akustische Signale mittels Statistik höherer Ordnung - Anwendung in Kfz und in Büroräumen*, Technische Universität Berlin, Dissertation, 2005

[4] BENESTY, Jacob ; HUANG, Yiteng ; CHEN, Jingdong: *Robust Time Delay Estimation Exploting Spatial Correlation*. 2003

[5] BENESTY, Jacob ; HUANG, Yiteng ; CHEN, Jingdong: *Microphone Array Signal Processing*. Berlin Heidelberg New York : Springer-Verlag, 2008. – ISBN 978-3-540-78611-5

[6] BRANDSTEIN, Michael ; WARD, Darren: *Microphone Arrays- Signal Processing Techniques and Applications*. Berlin Heidelberg New York : Springer-Verlag, 2001. – ISBN 3-540-41953-5

[7] CHEN, Zhizhang ; GOKEDA, Gopal K. ; YU, Yiqiang: *Introduction to Direction-of-Arrival Estimation* -. Norwood : Artech House, 2010. – ISBN 978-1-596-93090-2

[8] CHENG, Y. ; LI, B.: *DSP based Audio Source Localization*, Hochschule für Angewandte Wissenschaften Hamburg, Master-Thesis, 2004

[9] CLAUDIO, Elio D. D. ; PARISI, Raffaele: Weighted Average of Signal Subspace for Robust Wideband Direction Finding. In: *IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 49, NO. 10* (10.10.2001)

[10] CLIFFORD, Carter G.: Coherence and Time Delay Estimation. In: *PROCEEDINGS OF THE IEEE, VOL. 75, NO. 2* (02.02.1987)

[11] CONNOR, Frank R.: *Signale - Typen, Übertragung und Verarbeitung elektrischer Signale*. Berlin Heidelberg New York : Springer-Verlag, 2013. – ISBN 978-3-322-86216-7

[12] DAY, David M. ; HEROUX, Michael A.: Solving complex-valued linear systems via equivalent real formulations. In: *SIAM Journal of Scientific Computing* (2000). – URL `http://www.osti.gov/scitech/servlets/purl/756121`

[13] DMOCHOWSKI, Janek ; BENESTY, Jacob ; AFFES, Sofiène: Direction of Arrival Estimation Using the Parameterized Spatial Correlation Matrix. In: *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 15, NO. 4* (04.05.2007)

[14] DOBRINSKI, Paul ; KRAKAU, Gunter ; VOGEL, Anselm: *Physik für Ingenieure -*. Berlin Heidelberg New York : Springer-Verlag, 2013. – ISBN 978-3-322-93887-9

[15] DOERR, Benjamin: *Mathematik für Informatiker II - Singulärwertzerlegung*. – URL `http://resources.mpi-inf.mpg.de/departments/d1/teaching/ss10/MFI2/kap47.pdf`. – Zugriffsdatum: 21.07.2015

[16] EHRHARD, Stephan ; FISCHER, Moritz ; FUHR, Manuel ; MOUAZZEN, Mortada ; MÜLLER, Marcus ; SCHULZ, Marc L.: *Spektralschätzung mit MUSIC und ESPRIT*. – URL `digbib.ubka.uni-karlsruhe.de/volltexte/documents/1678008`. – Zugriffsdatum: 20.04.2015

[17] EICHLER, Jürgen: *Physik Grundlagen für das Ingenieurstudium - kurz und prägnant*. Wiesbaden : Vieweg-Verlag, 2007. – ISBN 978-3-8348-0223-1

[18] ERNST, Hartmut ; SCHMIDT, Jochen ; BENEKEN, Gerd: *Grundkurs Informatik - Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung*. Berlin Heidelberg New York : Springer-Verlag, 2015. – ISBN 978-3-658-01628-9

[19] GOLUB, Gene H. ; KAHAN, William: Calculating the Singular Values and Pseudo-Inverse of a Matrix. In: *IEEE TRANSACTIONS ON ACOUSTICS SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-34, NO. 2* (02.04.1986)

[20] HEUPELN, Orchan: *Ereignisgesteuerte Programmierung unter Qt Signal/Slot Mechanismus*, Fachhochschule Aachen, Seminararbeit, 2009

[21] HOHENESTER, Ulrich: *Computational-Physics 1 - Vorlesungs-Skript*. – URL `http://physik.uni-graz.at/~uxh/teaching/computational-physics1/kapitel5.pdf`. – Zugriffsdatum: 16.07.2015

[22] KAVEH, Mostafa ; BARABELL, Arthur J.: The Statistical Performance of the MUSIC and the Minimum-Norm Algorithms in Resolving Plane Waves in Noise. In: *IEEE TRANSACTIONS ON ACOUSTICS SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-34, NO. 2* (02.04.1986)

[23] KÖHLER, Bert-Uwe: *Konzepte der statistischen Signalverarbeitung*. Berlin Heidelberg New York : Springer-Verlag, 2005

[24] KLEMENZ, Adolf ; NÖLKER, Norbert: *D.Module.C6713 Technical Data Sheet*. 2012. – URL `http://www.dsignt.de/files/dsignt/media/datasheets/tdd6713R3.pdf`. – Zugriffsdatum: 11.07.2015

[25] KLEMENZ, Adolf ; NÖLKER, Norbert: *D.Module.C6657 Data Sheet*. 2014. – URL `http://www.dsignt.de/files/dsignt/media/datasheets/tdd2c6657.pdf`. – Zugriffsdatum: 11.07.2015

[26] KLEMENZ, Adolf ; NÖLKER, Norbert: *D.Module.PCM3003 Technical Data Sheet*. 2014. – URL `http://www.dsignt.de/files/dsignt/media/datasheets/tddpcm3003.pdf`. – Zugriffsdatum: 11.07.2015

[27] LEIBNIZ-INSTITUT FÜR NEUROBIOLOGIE MAGDEBURG: *Hauptkomponentenanalyse – Principal Component Analysis (PCA)*. – URL `ftp://ftp.ifn-magdeburg.de/pub/MBLehre/sv06_130509-ftp.pdf`. – Zugriffsdatum: 29.03.2015

[28] LIBERTY, Edo: *Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)*. – URL `http://www.cs.yale.edu/homes/el327/datamining2012aFiles/06_singular_value_decomposition.pdf`. – Zugriffsdatum: 21.07.2015

[29] MILDENBERGER, Otto: *Informationstechnik kompakt - Theoretische Grundlagen*. Berlin Heidelberg New York : Springer-Verlag, 2013. – ISBN 978-3-322-90262-7

[30] PANASONIC: *PANASONIC type WM, MICROPHONE INSERTS (Electret)*. – URL `http://www.anglia.com/product_guide/sensors/686_689.pdf`. – Zugriffsdatum: 12.07.2015

[31] PAPULA, Lothar: *Mathematik für Ingenieure und Naturwissenschaftler 2 - Ein Lehr- und Arbeitsbuch für das Grundstudium*. Berlin Heidelberg New York : Springer-Verlag, 2009. – ISBN 978-3-834-80564-5

[32] PIKORA, Kolja: *Korrelationsalgorithmen zur Sprecherlokalisierung mittels DSP-basiertem Echtzeitsystem und Mikrofonarray*, Hochschule für Angewandte Wissenschaften Hamburg, Masterthesis, 2011

[33] PRESS, William H. ; TEUKOLSKY, Saul A. ; VETTERLING, William T. ; FLANNERY, Brian P.: *Numerical Recipes 3rd Edition - The Art of Scientific Computing*. 3. Aufl. Cambridge : Cambridge University Press, 2007. – ISBN 978-0-521-88068-8

[34] REERMANN, Jens: *Echtzeit-Lokalisierung von Audioquellen mittels zirkularen Mikrofonarrays und des UCA-ESPRIT-Algorithmus*, Hochschule für Angewandte Wissenschaften Hamburg, Masterthesis, 2013

[35] ROY, Richard ; KAILATH, Thomas: ESPRIT-Estimation of Signal Parameters Via Rotational Invariance Techniques. In: *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, VOL. 37, NO. 7* (1989)

[36] SAXENA, Anshul K.: *Wideband Audio Source Localization using Microphone Array and MUSIC Algorithm*, Hochschule für Angewandte Wissenschaften Hamburg, Master-Thesis, 2009

[37] SCHLUNDT, Johannes: *Schriftliche Ausarbeitung zum Vortrag Monte-Carlo-Simulation*. 2013. – URL `http://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2012_2013/ms-1213-schlundt-montecarlosimulation-ausarbeitung.pdf`. – Zugriffsdatum: 26.04.2015

[38] SCHOLZ, Matthias: *Nichtlineare Hauptkomponentenanalyse auf Basis neuronaler Netze*, Humboldt-Universität zu Berlin, Diplomarbeit, 2002

[39] STOICA, Petre ; NEHORAI, Arye: MUSIC, Maximum Likelihood, and Cramer-Rao Bound. In: *J-Siam Numer. Anal. Ser. B Vol. 2,No.2* (1965)

[40] TEXAS INSTRUMENTS: *C674x DSPLIB*. – URL `http://processors.wiki.ti.com/index.php/C674x_DSPLIB#DSPF_sp_fftSPxSP_.28Mixed_Radix_Forward_FFT_with_Bit_Reversal.29`. – Zugriffsdatum: 16.07.2015

[41] TUNCER, T. E. ; FRIEDLANDER, Benjamin: *Classical and Modern Direction-of-Arrival Estimation* -. 1. Aufl. Amsterdam, Boston : Academic Press, 2009. – ISBN 978-0-080-92307-9

[42] VESA, Andy: Sensitivity Analysis for Direction of Arrival Estimation using MUSIC ans Root-MUSIC Algorithm. In: *18thTelecommunications forum TELFOR* (2010)

[43] ZÜRCHER, Christoph ; FRANK, Thomas: *Bauphysik - Leitfaden für Planung undPraxis*. Berlin Heidelberg New York : Springer-Verlag, 1998. – ISBN 978-3-322-92789-7

A. Mathematische Grundlagen

In der vorliegenden Arbeit werden mathematische Verfahren verwendet, die einigen Lesern nicht bekannt sein könnten. Diese sollen hier kurz vorgestellt werden, um zum Verständnis der behandelten Themen beizutragen.

A.1. Grundlagen und Begriffe der statistischen Signalverarbeitung

In der vorliegenden Arbeit werden einige Methoden der statistischen Signalverarbeitung angewendet. Die Grundlagen der hierzu werden zum Beispiel in [23] beschrieben. Die folgenden Ausführungen sind daran angelehnt.

A.1.1. Erwartungswerte

Der Begriff Erwartungswert ist an den alltäglichen Begriff Durchschnitt oder auch Mittelwert angelehnt. Der Erwartungswert einer Zufallsvariablen beschreibt die Zahl, die die Zufallsvariable im Mittel annimmt. Falls diese einer diskreten Verteilung unterliegt, lässt sich der Erwartungswert wie folgt berechnen:

$$E[x] = \mu = \sum_{i=1}^{N} P(x_i) \cdot x_i \tag{A.1}$$

Der Erwartungswert einer Funktion $g(x)$ ist folgendermaßen definiert:

$$E[g(x)] = \mu = \int_{-\infty}^{\infty} g(x) \cdot f_X(x) dx \tag{A.2}$$

A.1.2. Verbunderwartungswerte

Der Verbunderwartungswert wird aus der Verbundverteilungsdichte für den diskreten Fall wie folgt gebildet:

$$E[g(x,y)] = \sum_i \sum_j g(x_i, y_i) \cdot P(x_i, y_i) \tag{A.3}$$

Im kontinuierlichen Fall gilt:

$$E[g(x,y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x_i, y_i) \cdot f_{XY}(x,y) dx dy \tag{A.4}$$

Ein Spezialfall von Gleichung A.4 ist die sogenannte Autokorrelationsfunktion (AKF), welche die statistischen Gemeinsamkeiten einer Funktion zum Zeitpunkt t mit derselben Funktion zu einem anderen Zeitpunkt $t + \tau$ beschreibt.

$$r_{xx}(t,\tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(t) \cdot x(t+\tau) \cdot f[x(t), x(t+\tau)] dx_t dx_{t+\tau} \tag{A.5}$$

Sofern $x(t)$ und $x(t + \tau)$ statistisch unabhängig voneinander sind, gilt:

$$r_{xx}(t,\tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(t) \cdot f[x(t)] \cdot x(t+\tau) \cdot f[x(t+\tau)] dx_t dx_{t+\tau} \tag{A.6}$$

$$= E[x(t)] \cdot E[x(t+\tau)] \tag{A.7}$$

Für den Vergleich von zwei unterschiedlichen Zufallsvariablen lässt sich die Kovarianzfunk-

tion angeben.

$$c_{xy}(t,\tau) = E[x(t) - \mu_x] \cdot E[y(t+\tau) - \mu_y] \tag{A.8}$$

Sofern es sich um ergodische[5] Prozesse handelt, lassen sich die Kovarianzfunktion und die Autokorrelationsfunktion wie folgt berechnen:

$$c_{xy}(\tau) = \int_{-\infty}^{\infty} [x(t) - \mu_x] \cdot [y(t+\tau) - \mu_y] dt \tag{A.9}$$

$$r_{xx}(t,\tau) = \int_{-\infty}^{\infty} x(t) \cdot x(t-\tau) dt \tag{A.10}$$

A.1.3. Erwartungswerte und Verbunderwartungswerte von Matrizen und Vektoren

Handelt es sich bei der Zufallsvariablen um einen Vektor $\mathbf{x} = [x_1, x_2, \cdots, x_N]$, ist der Erwartungswert wie folgendermaßen definiert:

$$E[\boldsymbol{x}] = \boldsymbol{\mu_x} = \begin{pmatrix} E[x_1] \\ E[x_2] \\ \vdots \\ E[x_N] \end{pmatrix} \tag{A.11}$$

Analog gilt für Matrizen:

$$E\left[\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{pmatrix}\right] = \begin{pmatrix} E[x_{11}] & E[x_{12}] & \cdots & E[x_{1M}] \\ E[x_{21}] & E[x_{22}] & \cdots & E[x_{2M}] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_{N1}] & E[x_{N2}] & \cdots & E[x_{NM}] \end{pmatrix} \tag{A.12}$$

[5] Lässt sich das vorliegende Signal durch eine Schar modellieren, welche aus unendlich vielen Signalen besteht, die zu jedem Zeitpunkt der gleichen Statistik unterliegen, gilt ein stochastischer Prozess als ergodisch, wenn der Scharrerwartungswert zu einem Zeitpunkt t mit dem Zeiterwartungswert übereinstimmt.

Ist **x** ein Vektor von Signalen ($\mathbf{x} = [x_1, x_2, \cdots, x_N]$) ergibt sich unter Berücksichtigung von Gleichung A.8 die Kovarianzmatrix:

$$E[(\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{x} - \boldsymbol{\mu_x})] = \begin{pmatrix} c_{X_1X_1}(0) & c_{X_1X_2}(0) & \cdots & c_{X_1X_N}(0) \\ c_{X_2X_1}(0) & c_{X_2X_2}(0) & \cdots & c_{X_2X_N}(0) \\ \vdots & \vdots & \ddots & \vdots \\ c_{X_NX_1}(0) & c_{X_NX_2}(0) & \cdots & c_{X_NX_N}(0) \end{pmatrix} \tag{A.13}$$

A.2. Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (Engl. Principal Component Analysis - PCA) ist für einen Teil der hier vorgestellten und auch implementierten Verfahren ein zentrales Element und soll darum einmal grundsätzlich vorgestellt werden. Die folgenden Ausführungen zur PCA beziehen sich auf [27].

Bei mehrdimensionalen Datensätzen scheint es oft sinnvoll, diese unter Einbehaltung der Hauptmuster zu reduzieren. Die Vorteile liegen dabei auf der Hand, denn ein reduzierter Datensatz bedeutet, dass es weniger Daten zu verarbeiten gibt, wodurch wiederum entschiedene Rechenzeit eingespart werden kann. In vielen Fällen - wie auch in dieser Arbeit - erfolgt eine Reduktion bzw. eine Aufteilung in Signal- und Rauschanteil eines Datensatzes.

Es stellt sich die Frage, welche die wichtigen und welche die irrelevanten Dynamiken sind, die dem Datensatz zugrunde liegen. Daher wird die Annahme getroffen, dass die Richtung mit der größten Varianz, die interessante Dynamik eines Systems beschreibt.

Das Verfahren der PCA soll nun anhand eines zweidimensionalen Datensatzes exemplarisch erläutert werden.

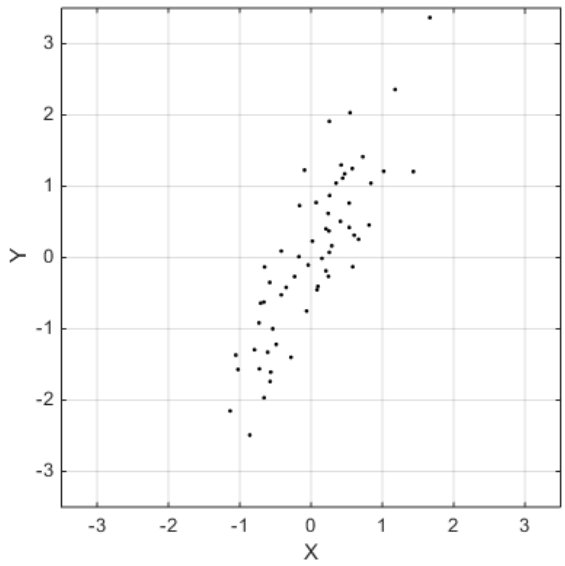


Abbildung A.1.: Zweidimensionaler Datensatz

Zunächst muss ein Einheitsvektor mit maximaler Varianz gefunden werden. Dies führt auf ein Eigenwertproblem zurück, wobei der größte Eigenwert λ und der zugehörige Eigenvektor $\boldsymbol{x}$ der Kovarianzmatrix $\boldsymbol{R}$ gefunden werden soll.

$$\boldsymbol{Rq} = \lambda \boldsymbol{q} \tag{A.14}$$

Sei $\boldsymbol{R}$ eine Matrix der Dimension N x N, so ist jeder Vektor $\boldsymbol{q}(\boldsymbol{q} \neq 0)$, für den mit einer geeigneten Zahl λ die o. g. Beziehung gilt, ein Eigenvektor von $\boldsymbol{R}$. λ ist der dazugehörige Eigenwert. Weiterhin gelten folgende Aussagen:

- Eigenvektoren existieren ausschließlich für quadratischen Matrizen.
- Nicht jede quadratische Matrix besitzt Eigenvektoren.
- Besitzt eine quadratische N x N Matrix Eigenvektoren, so besitzt sie genau N Eigenvektoren.
- Eigenvektoren sind zueinander orthogonal.

Gleichung A.14 lässt sich umstellen zu:

$$(\boldsymbol{R} - \lambda \boldsymbol{E})\boldsymbol{q} = 0 \tag{A.15}$$

mit der nichttrivialen Lösung ($\boldsymbol{q} \neq 0$):

$$det(\boldsymbol{R} - \lambda \boldsymbol{I}) = 0$$

$$det\left[\begin{pmatrix} c_{X_1X_1}(0) - \lambda & c_{X_1X_2}(0) & \cdots & c_{X_1X_N}(0) \\ c_{X_2X_1}(0) & c_{X_2X_2}(0) - \lambda & \cdots & c_{X_2X_N}(0) \\ \vdots & \vdots & \ddots & \vdots \\ c_{X_NX_1}(0) & c_{X_NX_2}(0) & \cdots & c_{X_NX_N}(0) - \lambda \end{pmatrix}\right] = 0 \tag{A.16}$$

Ein vom Nullvektor verschiedener Vektor ist ein Eigenvektor, wenn dessen Richtung durch Multiplikation mit der zugrunde liegenden Matrix nicht verändert wird. Er wird lediglich um einen Skalierungsfaktor λ gestreckt. Dieser ist Eigenwert der Matrix. Das bedeutet eine vom Nullvektor verschiedene Lösung des Gleichungssystems A.16, welches auch charakteristische Gleichung der Matrix $\boldsymbol{R}$ genannt wird, ist Eigenvektor von $\boldsymbol{R}$. Die Richtungen der extrahierten Eigenvektoren entsprechen den zugrunde liegenden Komponenten des Datensatzes. Die PCA liefert diese Richtungen (Merkmale) geordnet. Eine Reduzierung um Merkmale, deren Varianzen die kleinsten Werte aufweisen, bedeutet eine Reduktion des Datensatzes mit dem geringstmöglichen Informationsverlust. Das bedeutet, dass die ersten K Merkmale einen linearen Unterraum der Dimension K aufspannen, worin die Varianz der Daten am größten ist. Eine wichtige Eigenschaft der PCA ist die Orthogonalität. Alle Unterräume von dem original Datensatz sind orthogonal zueinander. [38, S. 5]
Grundsätzlich besteht die Aufgabe bei der PCA folglich darin eine Matrix $\boldsymbol{X}$ zu finden, mit welcher sich die Kovarianzmatrix diagonalisieren lässt.

$$\boldsymbol{X}^{-1}\boldsymbol{R}\boldsymbol{X} = \boldsymbol{D} \tag{A.17}$$

$\boldsymbol{D}$ ist hier die Diagonalmatrix der Eigenwerte von $\boldsymbol{R}$ und $\boldsymbol{X}$ ist die Eigenvektormatrix.
Die folgende Abbildung zeigt den ursprünglichen Datensatz mit den zugehörigen skalierten Eigenvektoren:

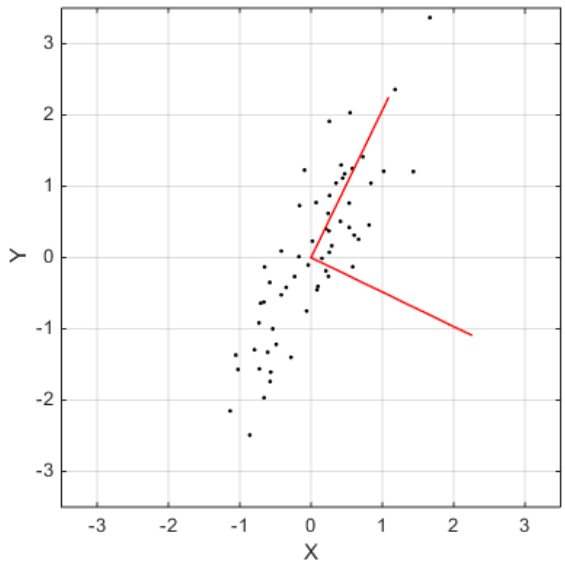


Abbildung A.2.: Zweidimensionaler Datensatz mit Eigenvektoren

Anschließend gilt es den ursprünglichen Datensatz mithilfe der Eigenvektoren genau um diese zu rotieren.

$$\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{A} \tag{A.18}$$

Wobei $\boldsymbol{Q}$ die Matrix aus Eigenvektoren und $\boldsymbol{A}$ der ursprüngliche Datensatz ist. Für das hier vorgestellte Beispiel eines zweidimensionalen Datensatz lautet diese Rechnung wie folgt:

$$\begin{aligned}\boldsymbol{Y} &= \boldsymbol{Q}\boldsymbol{A} \\ &= \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{pmatrix} \\ &= \begin{pmatrix} q_{11}a_{11} + q_{12}a_{21} & \cdots & q_{11}a_{1n} + q_{12}a_{2n} \\ q_{21}a_{11} + q_{22}a_{21} & \cdots & q_{21}a_{1n} + q_{22}a_{2n} \end{pmatrix}\end{aligned} \tag{A.19}$$

Das Ergebnis ist ein rotierter Datensatz, wie in Abbildung A.3 dargestellt.

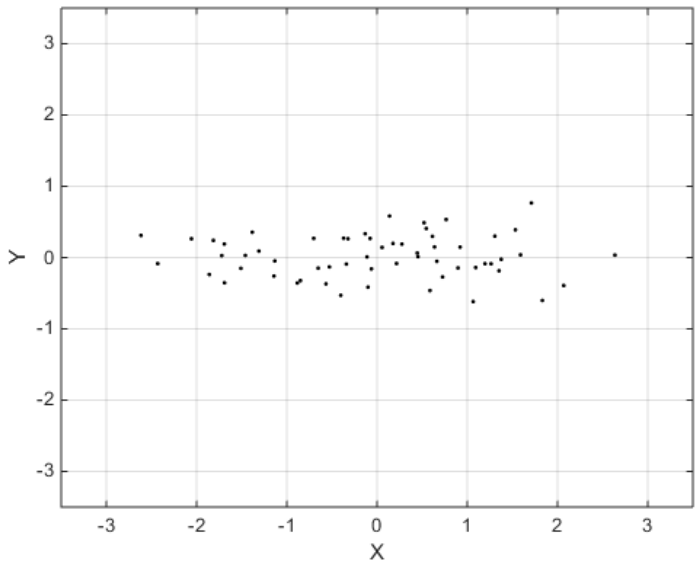


Abbildung A.3.: Rotierter zweidimensionaler Datensatz

Nun kann der Datensatz durch das Entfernen einer oder mehrerer Komponenten reduziert werden. Die folgende Abbildung zeigt den Beispieldatensatz, der bereits rotiert wurde in Schwarz dargestellt und dazu den um die Y-Komponente reduzierten Datensatz in Blau dargestellt.

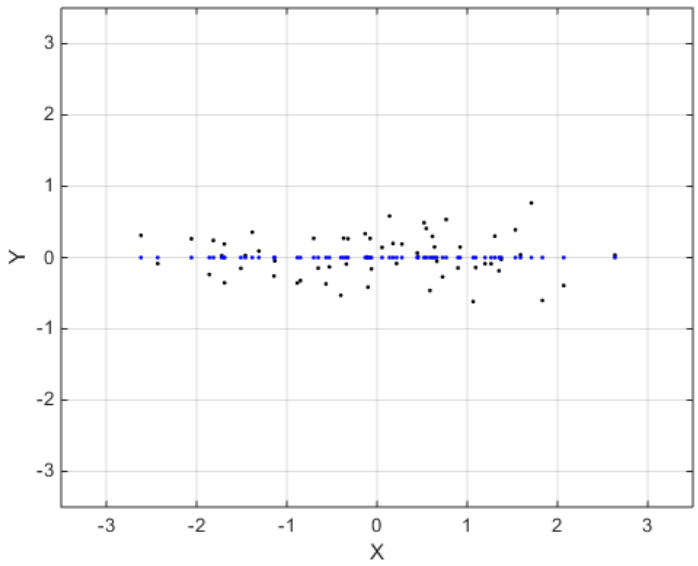


Abbildung A.4.: Rotierter zweidimensionaler Datensatz mit auf eine Dimension reduziertem Datensatz

B. Funktionsweise von MUSIC anhand eines Beispiels

Um die Funktionsweise von MUSIC besser nachvollziehen zu können, soll diese anhand eines simplen Beispiels beschrieben werden. Dafür wird ein Mikrofonarray mit drei Mikrofonen betrachtet, die in einem Abstand von 5 cm angeordnet sind. Es existiert eine Quelle, welche relativ zu dem Mikrofonarray in einem 45 Grad Winkel angeordnet ist. Das Signal der Quelle beinhaltet lediglich eine Frequenz, welche 480 Hz beträgt. Das Signal lässt sich somit wie folgt beschreiben:

$$s(t) = cos(2\pi \cdot 480 \cdot t) \circ\!\!-\!\!\bullet\, S(f) = \frac{1}{2} \cdot [\delta(f - 480) + \delta(f + 480)] \tag{B.1}$$

Die an den Mikrofonen empfangenden Signale lassen sich in kompakter Schreibweise folgendermaßen beschreiben:

$$\begin{pmatrix} Y_1(f,\phi) \\ Y_2(f,\phi) \\ Y_3(f,\phi) \end{pmatrix} = \begin{pmatrix} 1 \\ \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi))} \\ \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi))} \end{pmatrix} \cdot S(f) + \begin{pmatrix} \Upsilon_1(f) \\ \Upsilon_2(f) \\ \Upsilon_3(f) \end{pmatrix} \tag{B.2}$$

$$= \begin{pmatrix} 1 \\ \mathrm{e}^{-j2\pi \cdot 480 \cdot \frac{0{,}05}{343{,}2} \cdot sin(45 \cdot \frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi \cdot 480 \cdot 2 \cdot \frac{0{,}05}{343{,}2} \cdot sin(45 \cdot \frac{\pi}{180})} \end{pmatrix} \cdot \frac{1}{2} \cdot [\delta(f - 480) + \delta(f + 480)] + \begin{pmatrix} \Upsilon_1(f) \\ \Upsilon_2(f) \\ \Upsilon_3(f) \end{pmatrix} \tag{B.3}$$

wobei Υ_n das jeweilige an den Mikrofonen vorhandene Rauschen beschreibt und für die Funktion $\mathcal{F}_n(\tau(\phi))$

$$\mathcal{F}_n(\tau(\phi)) = (n-1) \cdot \frac{d}{v} \cdot sin(\phi) \tag{B.4}$$

gilt. Darin ist d der Abstand zwischen zwei Mikrofonen und v die Geschwindigkeit, für welche die Schallgeschwindigkeit von 343,2 $\frac{m}{s}$ angenommen wird.
Von diesem Signal wird nun die Kovarianzmatrix benötigt, welche nach Gleichung 2.56 wie folgt definiert ist:

$$\boldsymbol{R}_a(f,\phi) = \boldsymbol{\Sigma}(\tau(\phi))\boldsymbol{A}_s(f)\boldsymbol{\Sigma}^H(\tau(\phi)) + \sigma^2\boldsymbol{I} \tag{B.5}$$

Für das konkrete Beispiel ergibt sich daraus:

$$\begin{aligned}
\boldsymbol{R}_a(f,\phi) &= \begin{pmatrix} 1 \\ \mathrm{e}^{-j\omega\mathcal{F}_2(\tau(\phi))} \\ \mathrm{e}^{-j\omega\mathcal{F}_3(\tau(\phi))} \end{pmatrix} \cdot E\left[|S(f)|^2\right] \cdot \begin{pmatrix} 1 & \mathrm{e}^{j\omega\mathcal{F}_2(\tau(\phi))} & \mathrm{e}^{j\omega\mathcal{F}_3(\tau(\phi))} \end{pmatrix} + \sigma^2\boldsymbol{I} \\
&= E\left[|S(f)|^2\right] \\
&\cdot \begin{pmatrix} 1 & \mathrm{e}^{j\omega\mathcal{F}_2(\tau(\phi))} & \mathrm{e}^{j\omega\mathcal{F}_3(\tau(\phi))} \\ \mathrm{e}^{j\omega[\mathcal{F}_1(\tau(\phi))-\mathcal{F}_2(\tau(\phi))]} & 1 & \mathrm{e}^{j\omega[\mathcal{F}_3(\tau(\phi))-\mathcal{F}_2(\tau(\phi))]} \\ \mathrm{e}^{j\omega[\mathcal{F}_1(\tau(\phi))-\mathcal{F}_3(\tau(\phi))]} & \mathrm{e}^{j\omega[\mathcal{F}_2(\tau(\phi))-\mathcal{F}_3(\tau(\phi))]} & 1 \end{pmatrix} + \sigma^2\boldsymbol{I} \\
&= 0,5 \\
&\cdot \begin{pmatrix} 1 & \mathrm{e}^{j2\pi\cdot480\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} & \mathrm{e}^{j2\pi\cdot480\cdot2\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi\cdot480\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} & 1 & \mathrm{e}^{j2\pi\cdot480\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi\cdot480\cdot2\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} & \mathrm{e}^{-j2\pi\cdot480\cdot\frac{0,05}{343,2}\cdot sin(45\cdot\frac{\pi}{180})} & 1 \end{pmatrix} \\
&+ \sigma^2\boldsymbol{I}
\end{aligned} \tag{B.6}$$

Für den Fall, dass $\sigma^2 = 0$ ist, existiert genau ein Eigenwert, welcher sich über das charakte-

ristische Polynom bestimmen lässt.

$$0 = det\left(\boldsymbol{R}_a(f,\phi) - \lambda \boldsymbol{I}\right) \tag{B.7}$$

$$= det\begin{pmatrix} 1-\lambda & \mathrm{e}^{j2\pi\cdot 480\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} & \mathrm{e}^{j2\pi\cdot 480\cdot 2\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi\cdot 480\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} & 1-\lambda & \mathrm{e}^{j2\pi\cdot 480\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi\cdot 480\cdot 2\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} & \mathrm{e}^{-j2\pi\cdot 480\cdot \frac{0{,}05}{343{,}2}\cdot sin(45\cdot\frac{\pi}{180})} & 1-\lambda \end{pmatrix} \tag{B.8}$$

$$= -8\lambda^3 + 12\lambda^2 \tag{B.9}$$

Es ergibt sich genau eine von Null verschiedene Lösung für λ, welche den Wert $1,5$ hat. Diese ist dem Signal zuzuordnen. Für die zwei Nulleigenwerte müssen nun die passenden Eigenvektoren gefunden werden. Diese ergeben sich aus dem folgendem Gleichungssystem:

$$\left(\boldsymbol{R}_a(f,\phi) - \lambda \cdot \boldsymbol{I}\right)\boldsymbol{q}_m = \boldsymbol{0} \tag{B.10}$$

Für das vorliegende Beispiel wurden die zu dem Nulleigenwerten passenden Eigenvektoren mit MATLAB berechnet. Diese lauten wie folgt:

$$\boldsymbol{q}_{n1} = \begin{pmatrix} 0.8005 + j0.0255 \\ -0.5379 + j0.0075 \\ -0.2631 \end{pmatrix} \tag{B.11}$$

$$\boldsymbol{q}_{n2} = \begin{pmatrix} -0.1579 - j0.0181 \\ -0.6142 - j0.0108 \\ 0.7729 \end{pmatrix} \tag{B.12}$$

Diese werden nun zusammengefasst zu der Matrix $\boldsymbol{Q}_n$, welche per Definition orthogonal zu dem zum Signal passenden Steering-Vektor ist.

$$\boldsymbol{Q}_n = \begin{pmatrix} 0.8005 + j0.0255 & -0.1579 - j0.0181 \\ -0.5379 + j0.0075 & -0.6142 - j0.0108 \\ -0.2631 & 0.7729 \end{pmatrix} \tag{B.13}$$

Im letzten Schritt werden verschiedene Steering-Vektoren ausprobiert, um genau den orthogonalen zu finden.

$$S_{MUSIC}(\phi) = \frac{1}{\boldsymbol{\varsigma}(\tau(\phi))^H \cdot \boldsymbol{Q}_n \cdot \boldsymbol{Q}_n^H \cdot \boldsymbol{\varsigma}(\tau(\phi))} \tag{B.14}$$

Im konkreten Beispiel ist das genau der Steering-Vektor, mit dem bereits die Verzögerung zwischen den Signalen an den verschiedenen Mikrofonen in Gleichung B.3 beschrieben wurde.

$$\boldsymbol{\varsigma}(\tau(\phi)) = \begin{pmatrix} 1 \\ \mathrm{e}^{-j2\pi \cdot 480 \cdot \frac{0{,}05}{343{,}2} \cdot sin(45 \cdot \frac{\pi}{180})} \\ \mathrm{e}^{-j2\pi \cdot 480 \cdot 2 \cdot \frac{0{,}05}{343{,}2} \cdot sin(45 \cdot \frac{\pi}{180})} \end{pmatrix} \tag{B.15}$$

In der Praxis wird der passende Steering-Vektor gesucht, indem der Ausdruck aus Gleichung B.14 für verschiedene Winkel berechnet wird. Der Winkel, an dem dieser Ausdruck maximal ist, ist der gesuchte Winkel bzw. liegt am nahestehen daran, da die Auflösung aufgrund begrenzter Rechenzeit nicht unendlich fein sein kann.
Das beschrieben Maximum entsteht dadurch, dass das Produkt von zueinander orthogonalen Vektoren bzw. Unterräumen zu Null wird. Das so genannte Pseudospektrum nimmt an dieser Stelle theoretisch den Wert unendlich an. Durch Rauschen und nummerische Ungenauigkeiten tritt dieser Fall allerdings nie ein, sodass der Wert lediglich sehr groß wird. Die folgende Abbildung zeigt das Pseudospektrum zu dem behandelten Beispiel.

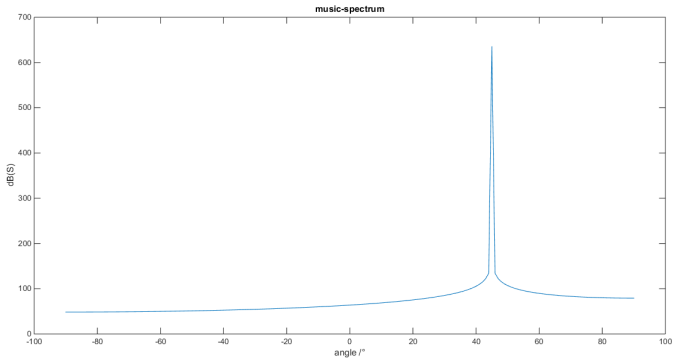


Abbildung B.1.: Pseudospektrum des MUSIC Verfahrens bei einer Quelle ohne Rauschen

C. Inhalt des Datenträgers

Zu dem Druckexemplar dieser Arbeit gehört ein Datenträger, dessen Inhalte nachfolgend aufgelistet werden. Der Datenträger liegt den Prüfern Prof. Dr. Ulrich Sauvagerd von der Hochschule für Angewandte Wissenschaften Hamburg und Prof. Dr. Hans-Dieter Schütte von der Fachhochschule Westküste vor und kann nach Absprache eingesehen werden.

01_PDF Digitale Version dieser Arbeit im pdf-Format.

02_MATLAB Sämtliche MATLAB-Skripte, die zur Simulation der Verfahren in dieser Arbeit genutzt wurden.

03_DSP_Software Softwareprojekt, welches auf dem DSP mit Code Composer Studio 6 entwickelt wurde.

04_PC_Software Softwareprojekt in QT, welches zu Visualisierung und Aufzeichnung der Ergebnisse dient.

Versicherung über die Selbstständigkeit

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 habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 28. August 2015	
Ort, Datum	Unterschrift