HAW HAMBURG

MASTER THESIS
Arthur Haarring

Entwicklung eines Demonstrators für akustische Echounterdrückung auf einem digitalen Signalprozessor

FAKULTÄT TECHNIK UND INFORMATIK
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Arthur Haarring

Entwicklung eines Demonstrators für akustische Echounterdrückung auf einem digitalen Signalprozessor

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informations- und Kommunikationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Lutz Leutelt
Zweitgutachter: Prof. Dr. Matthias Kronauge

Eingereicht am: 27. September 2024

Arthur Haarring

Thema der Arbeit

Entwicklung eines Demonstrators für akustische Echounterdrückung auf einem digitalen Signalprozessor

Stichworte

Akustische Echounterdrückung, Adaptive Filter, Sprachsignalverarbeitung, Fast-Least-Mean-Squares, Subband AEC

Kurzzusammenfassung

Inhalt dieser Arbeit ist die Beschreibung, Simulation, Implementierung und Evaluation eines adaptiven Filters für Acoustic Echo Cancellation, in Form eines Demonstrators für Lehrzwecke. Der Fast-Least-Mean-Squares (FLMS) und der Subband AEC Algorithmus wurden zu diesem Zweck untersucht und simuliert. Umgesetzt wurde der FLMS Algorithmus, welcher in der Paxis Echodämpfungen von über 30 dB erreichte.

Arthur Haarring

Title of Thesis

Development of a Demonstrator for Acoustic Echo Cancellation on a Digital Signal Processor

Keywords

Speech Signal Processing, Acoustic Echo Cancellation, Adaptive Filtering, Fast-Least-Mean-Squares, Subband AEC

Abstract

This thesis includes the description, simulation, implementation, and evaluation of an adaptive filter for Acoustic Echo Cancellation, in the form of a demonstrator for educational purposes. The Fast-Least-Mean-Squares (FLMS) and the Subband AEC algorithms were studied and simulated for this purpose. The FLMS algorithm was then implemented, achieving echo suppression of over 30 dB in practice.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Kommunikation über weite Distanzen mittels Telefon- oder Videokonferenzen ist heutzutage nicht mehr wegzudenken. Die Arbeitskräfteerhebung des Statistischen Bundesamtes zeigt, dass im Jahr 2022 mit 24,3 % fast ein Viertel aller Erwerbstätigen in Deutschland von zu Hause gearbeitet hat [44]. Viele von ihnen sind darauf angewiesen, dass die Kommunikation mit ihren Kolleginnen und Kollegen einwandfrei und zuverlässig funktioniert. Verschiedene Faktoren spielen hierbei eine Rolle: die Zuverlässigkeit des Kanals zwischen den Kommunizierenden, die Qualität der Audio- und Videoaufnahmegeräte und nicht zuletzt auch die Signalverarbeitung, welche eventuelle Qualitätsmängel des Signals unterdrücken oder zumindest reduzieren kann. Eine besondere Bedeutung kommt dabei dem zentralen und wichtigsten Kommunikationsmittel, der Sprache, zu. Eine Vielzahl von Technologien und Algorithmen der Sprach- bzw. Audiosignalverarbeitung soll dafür sorgen, dass beide Seiten sich möglichst rauscharm verstehen und eventuelle unerwünschte Effekte unterdrückt werden.

Einer dieser unerwünschten Effekte ist Thema dieser Arbeit: Das Echo, welches bei Kommunikation via Mikrofon und Lautsprecher auftritt. Es hat zur Folge, dass die miteinander kommunizierenden Parteien sich mit einer leichten Verzögerung jeweils selbst hören. Eine Lösung dieses Problems ist das sogenannte Acoustic Echo Cancelling (AEC). Diese Art des Echos ist allerdings abzugrenzen von dem Echo, welches durch Reflexion des Signals in Telefonleitungen entsteht und durch einen Echo Suppressor reduziert wird [5]. Eine der frühesten Beschreibungen eines Echo-Cancellers stammt von Bell-Labs im Jahr 1966 im Zuge der Einführung des ersten Telekommunikationssatelliten Telstar [42]. Durch die langen Verzögerungen der Satelliten (bis zu 600 ms) notwendig geworden, entwickelten Sonhi et al. [41] den folgenden Lösungsansatz: die Verwendung eines adaptiven Filters, welcher das zu erwartende Echo modelliert und vom Sendesignal abzieht [41]. Dieser Ansatz ist noch heute die Grundlage einer Vielzahl von weiterentwickelten und optimierten Algorithmen zur Echounterdrückung [2] und findet auch in dieser Arbeit Anwendung.

Algorithmen zur akustischen Echounterdrückung und adaptive Filter beinhalten vielfältige Techniken und Verfahren der klassischen Signalverarbeitung, wie sie Gegenstand in der Lehre der Informations- und Kommunikationstechnik sind und bieten daher eine Möglichkeit, das theoretisch angeeignete Wissen zu kombinieren und innerhalb eines realen Systems anzuwenden. Dies führt zum Ziel dieser Arbeit: Die Entwicklung eines Demonstrators für akustische Echounterdrückung auf einem digitalen Signalprozessor. Der Demonstrator soll es Studierenden ermöglichen, Algorithmen und Methoden der klassischen Signalverarbeitung praxisnah anzuwenden, zu kombinieren und auf einem realen System zu implementieren.

Dazu wird in Kapitel 2 zunächst an die Theorie von Acoustic Echo Cancelling und adaptiven Filtern allgemein herangeführt, bevor in Kapitel 2.3 und 2.4 zwei Algorithmen exemplarisch vorgestellt und mathematisch beschrieben werden. Am Ende des Kapitels werden die Anforderungen an den zu entwickelnden Demonstrator definiert. Es folgt die Konzeption und Dokumentation der Umsetzung in Kapitel 3. Zuletzt wird der entwickelte Demonstrator in Kapitel 4 im Hinblick auf die zuvor definierten Anforderungen evaluiert.

2 Analyse

Dieses Kapitel stellt die theoretischen Grundlagen dieser Arbeit dar. Es wird zunächst das grundlegende Prinzip von AEC erläutert, angefangen bei einer Definition, möglichen Anwendungsbereichen, Herangehensweisen und Qualitätsmaßen. Es folgt die Einführung in die Theorie von adaptiven Filtern und die exemplarische Beschreibung von zwei adaptiven Filteralgorithmen. Dabei werden gewisse Grundlagen, wie beispielsweise die Anwendung von FIR-Filtern, die diskrete Fouriertransformation (DFT) in Form der Fast-Fourier-Transformation (FFT) und die Frequenzmodulation vorausgesetzt. Das Kapitel schließt mit der Definition der Anforderungen, welche die im weiteren Verlauf der Arbeit beschriebene Umsetzung erfüllen sollte.

2.1 Acoustic Echo Cancellation

Die Störung, deren Reduktion Acoustic Echo Cancellation zum Ziel hat, besteht darin, dass sich bei einer Telefon- oder Videokonferenz eine oder mehrere der Kommunikationsparteien mit kurzer zeitlicher Verzögerung selbst hören. Das Echo stört die Kommunikation und kann in extremen Fällen sogar jegliche Verständigung unterbinden. Aus diesem Grund ist AEC im Bereich der Telekommunikation und besonders bei Vollduplex-Übertragungen von entscheidender Bedeutung und stellt bis heute eine Herausforderung für die Signalverarbeitung dar [9].

Um die Entstehung des Echos zu visualisieren, zeigt Abbildung 2.1 ein Schema, welches den Weg des akustischen Signals und die Entstehung des Echos an einem Ende der Kommunikation darstellt. Dabei spielen folgende Signale eine zentrale Rolle:

- *Far Speech Signal* $x[k]$: Das Sprachsignal vom entfernten Ende des Kommunikationskanals, welches auf den Lautsprechern ausgegeben wird.
- *Echo Signal* $\tilde{x}[k]$: Das vom Mikrofon empfangene Echo des *Far Speech Signal*, welches durch den Raum reflektiert wurde.

- *Near Speech Signal* $s[k]$: Das Sprachsignal am nahen Ende des Kommunikationskanals, welches ebenfalls vom Mikrofon empfangen wird.

Das Echo Signal und das Near Speech Signal sind zusammen im Mikrofonsignal $y[k]$ enthalten und liegen in der nicht getrennt voneinander vor.

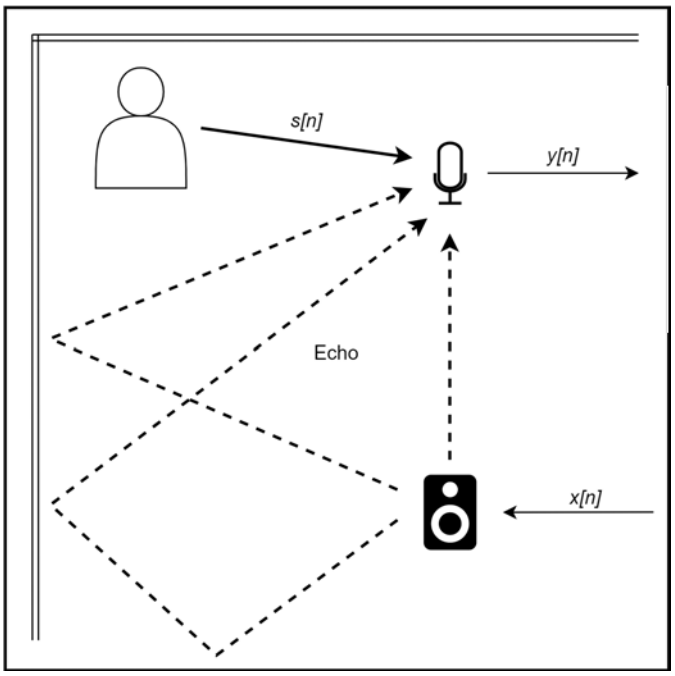

Abbildung 2.1: Schema der Entstehung des Echos mit den Signalbezeichnungen für Far Speech Signal $x[k]$, Near Speech Signal $s[k]$ und Mikrofonsignal $y[k]$

Die Herausforderung und Aufgabe von AEC ist es, den akustischen Pfad vom Lautsprecher zum Mikrofon zu modellieren und das erwartete Echo zu simulieren, damit es aus dem Mikrofonsignal entfernt werden kann. Im Allgemeinen wird die Aufgabe, ein unbekanntes System (hier den akustischen Pfad) nachzubilden, als *Systemidentifikation* bezeichnet. Die Wahl des Algorithmus, sowie seine Komplexität und sein Aufwand hängen dabei maßgeblich von der Umgebung ab, das heißt von den Eigenschaften des akustischen Pfades. Das System, bestehend aus Lautsprecher, Übertragungsweg und Mikrofon, wird als LEM System (Loudspeaker-Enclosure-Microphone System) oder auch LRM System (Lautsprecher-Raum-Mikrofon System) bezeichnet.

2.1.1 Anwendungsbereiche

AEC findet bei einer Vielzahl von Telekommunikationssystemen Anwendung. Neben dem Haus- oder Mobiltelefon, welches bei Aktivierung der Lautsprecher AEC einsetzen muss, ist die Echounterdrückung essenziell für Konferenzsysteme, welche standardmäßig eine Vollduplex-Kommunikation ermöglichen [51]. Ein weiterer prominenter Anwendungsbereich sind die Kommunikationsanlagen von Kraftfahrzeugen, welche aus Gründen der Verkehrssicherheit ebenfalls ausschließlich die Kommunikation per Lautsprecher zulassen [51]. Da bei den verschiedenen Anwendungsgebieten signifikante Unterschiede in Raumakustik, Signalqualität, Störgeräuschen und verfügbarer Rechenleistung vorherrschen, existiert eine große Bandbreite von Algorithmen, welche für ihr jeweiliges Anwendungsgebiet optimiert sind [11]. So weist beispielsweise die akustische Umgebung in einem Auto eine vergleichsweise kurze Impulsantwort von etwa 30 - 100 ms auf, welche jedoch von den Interaktionen der Fahrgäste abhängig ist und daher zeitlichen Variationen unterliegt [11]. Zusammen mit den in dieser Umgebung herrschenden Störgeräuschen bildet dieser Anwendungsbereich größere Herausforderungen für AEC. Die Umgebung, welche dieser Arbeit als Grundlage dient, ist die einer Audio- oder Videokonferenz via Desktop Computer (siehe Abbildung 2.2) [11]. Der akustische Pfad ist dabei in der Regel weniger Variationen ausgesetzt, als in Autos oder bei mobilen Endgeräten. Es kann davon ausgegangen werden, dass es sich bei der Umgebung um einen geschlossenen Raum mit vergleichsweise niedrigem Störgeräuschpegel handelt. Herausforderungen sind hier in der Praxis variierende Computerarchitekturen, Betriebssysteme und Audiohardware sowie Latenz, Geschwindigkeit und Stabilität der Netzwerkverbindung [11].

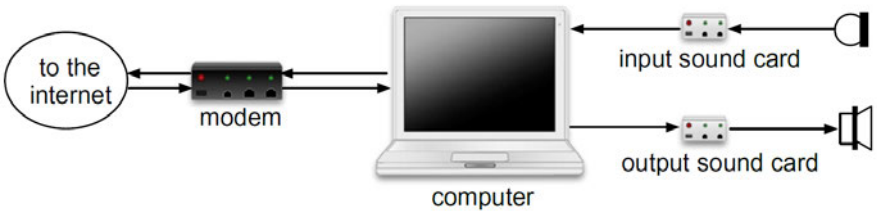

Abbildung 2.2: Schema einer Audiokonferenz via Desktop Computer [11]

2.1.2 Herangehensweisen

Für die Aufgabe der Modellierung eines unbekannten Systems existieren unterschiedliche Herangehensweisen, angefangen bei klassischer Signalverarbeitung mit linearen adaptiven

Filtern, über den Einsatz von speziellen Kalman Filtern hin zu Machine Learning. Zwei wesentliche Argumente sprechen für den Einsatz von linearen adaptiven Filtern in dieser Arbeit.

(1) Da bei adaptiven Filtern eine große Vielfalt von fortgeschrittenen Methoden der klassischen Signalverarbeitung Anwendung findet und die zugrundeliegenden Verfahren, wie FIR-Filter, Optimierungsverfahren und die DFT vorausgesetzt werden können, sind diese aus didaktischer Sicht eine sinnvolle Wahl, um bereits Gelerntes anzuwenden, zu kombinieren und darauf aufbauend neue Methoden kennenzulernen.
(2) Aufgrund der Vielfältigkeit von optimierten adaptiven Filtern zur Echounterdrückung, besteht eine breite Auswahl von Algorithmen, welche auch auf leistungsschwächeren Systemen eingesetzt werden können, sodass bei der Umsetzung auf möglicherweise bereits bestehendes Equipment zurückgegriffen werden kann. So wird eine höhere Flexibilität im Bezug auf den spezifischen Einsatzbereich des Demonstrators erreicht.

Dennoch sollen an dieser Stelle alternative Herangehensweisen und auch aktuelle Entwicklungen im Bereich AEC nicht unerwähnt bleiben. Eine Methode, welche klassische adaptive Filter teilweise übertrifft, sind die Kalman Filter. Diese basieren auf dem stochastischen Zustandsraummodell und werden angewandt, um Zustände auf Basis linearer dynamischer Systeme zu schätzen [24]. Kalman Filter, auch Optimalfilter genannt, sind den linearen adaptiven Filter ähnlich. Vor allem die Beziehung zwischen den adaptiven Filteralgorithmen Recurrent-Least-Squares (RLS) und Frequency-Domain-Adaptive Filter (FDAF) zu Kalman Filtern zeigt deren nahe Verwandtschaft [35], [27]. Eine Einführung in die Verwendung eines General Kalman Filter (GKF) für die Echounterdrückung geben Paleologu et al. [35]. Enzner & Vary [12] stellen einen effizienten und robusten adaptiven Kalman Filter zur Echounterdrückung vor, welcher mit einem stochastischen Zustandsraummodell des akustischen Echopfades arbeitet und die Berechnungen vollständig im Frequenzbereich durchführt (Frequency-Domain Adaptive Kalman Filter). Die Anwendung dieses Algorithmus wäre zwar auch für diese Arbeit denkbar, dagegen spricht allerdings seine erhöhte mathematische Komplexität im Vergleich zu klassischen adaptiven Filteralgorithmen. Aktuelle Entwicklungen steigern die Performance und Robustheit von Kalman Filtern durch die Kombination mit neuronalen Netzen zu Neuronalen Kalman Filtern (NKF), welche die Kovarianz des Zustandsrauschens und des Beobachtungsrauschens modellieren [56].

Neuronale Netze und Deep Learning bringen nicht nur in der Kombination mit Kalman Filtern signifikante Vorteile. Wettbewerbe wie die Acoustic Echo Cancellation Challenge

[8], [36], [9] zeigen, dass Algorithmen, welche neuronale Netze verwenden, den Methoden der klassischen Signalverarbeitung überlegen sind, vor allem in Hinblick auf die Robustheit gegenüber schnellen Variationen der Raumimpulsantwort und die Qualität der Echodämpfung. Die Anwendungen von Machine Learning reichen von der Kombination klassischer Signalverarbeitung und einem Deep-Neural-Network als zusätzlicher Maßnahme zur Reduzierung des Restechos [55], hin zu Kombinationen in Form der bereits erwähnten Neuronalen Kalman Filtern [56] und Deep Adaptive AEC [57]. Die große Anzahl unterschiedlicher Herangehensweisen zeigt, dass AEC auch heute noch Gegenstand der Forschung ist und selbst die aktuellsten Lösungsansätze noch Raum für Verbesserungen lassen [9].

2.1.3 Qualitätsmaße

Um die Qualität der Echokompensation objektiv zu beurteilen, stehen verschiedene Qualitätsmaße zur Auswahl, welche im Folgenden erläutert werden. Neben der Qualität der Kompensation sind allerdings auch weitere Faktoren, wie Konvergenzzeit, Nachführverhalten und Rechenaufwand entscheidend (siehe Kapitel 2.2.1).

Systemabstand

Der Systemabstand $D[k]$ bezeichnet das Maß für die Güte der Systemidentifikation und wird in Dezibel angegeben [51]. Er misst den Grad der Abweichung der vom adaptiven Filter modellierten Impulsantwort $\mathbf{h}[k]$ zum (diskreten) Zeitpunkt k zur Impulsantwort des realen Systems $\mathbf{g}$, welche als zeitinvariant angenommen wird. Dazu muss zunächst der Systemabstandvektor $\mathbf{d}[k]$ bestimmt werden:

$$\mathbf{d}[k] = \mathbf{g} - \mathbf{h}[k] \tag{2.1}$$

$$\frac{D[k]}{dB} = 10 \cdot log_{10} \frac{|\mathbf{d}[k]|^2}{|\mathbf{g}|^2} \tag{2.2}$$

Das Problem dieses Qualitätsmaßes ist, dass die Impulsantwort realer Systeme nicht bekannt ist und die Berechnung des Systemabstandes somit in der Praxis nicht möglich ist. Ist die Impulsantwort jedoch vorgegeben, beispielsweise bei einem Simulationsexperiment, so kann der Systemabstand durchaus sinnvoll eingesetzt werden [51].

ERLE

Echo Return Loss Enhancement (ERLE) ist ein Qualitätsmaß, welches spezifisch auf AEC Algorithmen angewendet wird. Es wird, wie der Systemabstand, in Dezibel angegeben und beschreibt das Verhältnis der Signalleistung vor und nach der Echokompensation ohne Near Speech Signal [51]. Es gibt an, wie viel das Echosignal $\tilde{x}[k]$ durch das vom Algorithmus geschätzte Echosignal $\hat{x}[k]$ gedämpft wurde.

$$\frac{ERLE[k])}{dB} = 10 \cdot log_{10}\Big(\frac{E\{\tilde{x}[k]^2\}}{E\{\tilde{x}[k]^2 - \hat{x}[k]^2\}}\Big) \tag{2.3}$$

Die Erwartungswerte der Signale werden in der praktischen Umsetzung durch Kurzzeit-Erwartungswerte $\hat{E}\{\}$ mit begrenzter zeitlicher Auflösung ersetzt [51]. Aufgrund der relativ simplen Berechnung wird dieses Maß im weiteren Verlauf dieser Arbeit verwendet, auch wenn es, genau wie PESQ, nicht gut mit subjektiven Sprachqualitätstests in realistischen Umgebungen korreliert [43].

PESQ

Perceptual Evaluation of Speech Quality (PESQ) ist ein weiteres konventionelles Maß für die Qualität von Sprachsignalen in der Telekommunikation [43], [28]. Es basiert auf dem Vergleich des ungestörten Signals (Referenzsignal) zum gestörten Signal. Wie in Abbildung 2.3 zu sehen, werden zunächst die Pegel der Signale angepasst, sodass diese einem Standard-Hörpegel entsprechen. Der Eingangsfilter filtert das Signal entsprechend der Impulsantwort eines Standard-Telefonhandgerätes, anschließend werden die Signale zeitlich ausgerichtet und transformiert. Die Transformation bewirkt eine Umwandlung der Signale, sodass diese die psychoakustisch empfundenen Lautstärken in Zeit- und Frequenzbereich repräsentieren [38]. Dies geschieht über mehrere Verarbeitungsstufen mit Hilfe eines psychoakustischen Modells. In den letzten beiden Verarbeitungsschritten werden schließlich gewichtete Differenzen der transformierten Signale gebildet, wobei psychoakustische Effekte wie Masking, Deletion und Asymmetry berücksichtigt werden [38]. Das Ergebnis der PESQ wird schließlich auf eine Skala mit der Bezeichnung Mean Opinion Score (MOS) von 1,0 (schlechte Qualität) bis 4,5 (keine Störung) abgebildet [38].

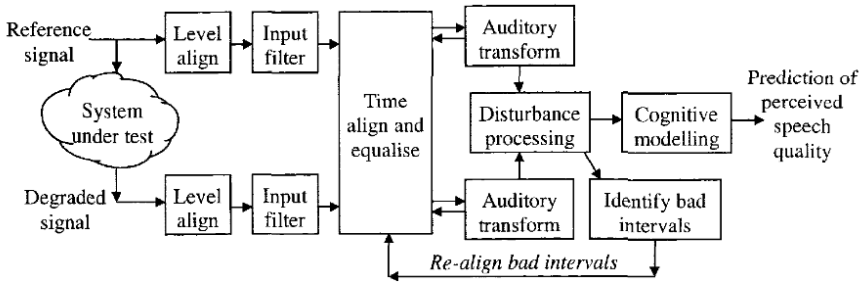

Abbildung 2.3: Schema des PESQ Qualitätsmaßes [38]

PESQ kann zwar ebenfalls für diesen Anwendungsfall verwendet werden, zielt aber hauptsächlich auf die Bewertung von Verzerrungen, Rauschen und Segmentierungen eines Sprachsignals ab [28]. Außerdem ist die Berechnung verglichen mit dem ERLE-Maß wesentlich komplexer. Einen tieferen Einblick in das Thema der Evaluation von Sprachsignalen gibt Loizou [28].

2.2 Adaptive Filter

Dieses Kapitel stellt adaptive Filter als Methode der klassischen Signalverarbeitung vor, sowie ihre Verwendung für AEC. Es legt die theoretische Basis für die in Kapitel 2.3 und 2.4 exemplarisch vorgestellten adaptiven Filteralgorithmen. Dabei ist zu beachten, dass sich die Theorie und Anwendung in dieser Arbeit auf lineare, FIR-basierte adaptive Filter beschränkt, auch wenn dies nicht immer explizit erwähnt wird. Aufgrund ihrer Stabilität und einfachen mathematischen Handhabung sind FIR-basierte adaptive Filter am weitesten verbreitet [31]. Adaptive IIR-Filter können allerdings auch Vorteile bringen, hauptsächlich die signifikante Reduktion der notwendigen Rechenleistung bei gleichbleibender Qualität des Sprachsignals [53], was bereits durch einige Studien demonstriert wurde [30], [33], [7]. Eine Einführung in nicht-lineare adaptive Filter, sowie die Möglichkeiten ihrer Verwendung für AEC ist in Zhao & Chen [58], sowie Hänsler & Schmidt [22] zu finden.

Grundlegende Aufgabe von adaptiven Filtern ist die Extraktion von Informationen aus einem oder mehreren Signalen [31]. Anders als bei zeitinvarianten Filtern, sind die Koeffizienten von adaptiven Filtern nicht entsprechend einer gewünschten Soll - Übertragungsfunktion festgelegt, sondern sind variabel und werden während der Laufzeit entsprechend eines Optimierungsalgorithmus angepasst. Ein adaptives Filter besteht demnach aus zwei

wesentlichen Bestandteilen: einem linearen Filter und einem Adaptionsalgorithmus. Anstelle einer Soll - Übertragungsfunktion wird ein Soll - Ausgangssignal $y[k]$ vorgegeben, welches mit dem Eingangssignal über ein unbekanntes System zusammenhängt. Dieses soll durch das adaptive Filter linear modelliert werden, sodass der Ausgang des Filters $\hat{x}[k]$ einen Schätzwert des Soll - Ausgangssignals, also des Ausgangs des unbekannten System darstellt [31]. Dabei ist das Eingangssignal $x[k]$ des Filters und des unbekannten Systems identisch. Die Abweichung des Filtersignals vom Soll - Ausgangssignal wird als Schätzfehler oder Fehlersignal $e[k]$ bezeichnet und dient sowohl als Grundlage des Optimierungsalgorithmus, als auch zur Evaluation der Filterqualität (siehe Kapitel 2.1.3) [31]. Das Schema eines adaptiven Filters ist Abbildung 2.4 zu entnehmen.

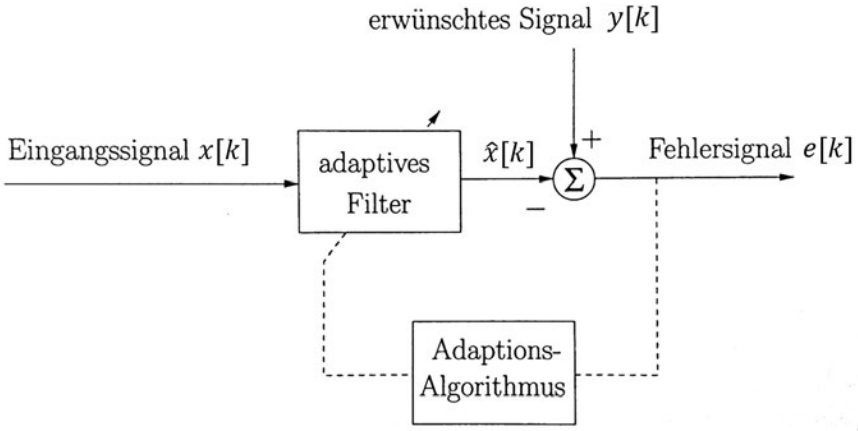

Abbildung 2.4: Schema eines adaptiven Filters. Veränderte Darstellung nach [31]

Je nach Anwendungsbereich des adaptiven Filters sind verschiedene Parameter und Signale von Bedeutung. Bei der Systemidentifikation sind die Filterkoeffizienten von zentralem Interesse, da diese das lineare Modell des unbekannten Systems repräsentieren. Dabei ist es auch möglich, dass die Koeffizienten lediglich während einer Lern- bzw. Trainingsphase adaptiert werden. Dies setzt allerdings voraus, dass das unbekannte System zeitinvariant ist oder sich nur sehr langsam ändert [31]. Weitere Einsatzgebiete umfassen unter anderem die inverse Modellierung, welche bei der Datenentzerrung Anwendung findet, die lineare Prädiktion, welche den zukünftigen Verlauf eines Signals abschätzt und bei der Linear Predictive Coding Analyse (LPC) von Sprache eingesetzt wird, sowie die Entfernung von Störungen aus einem Nutzsignal [31].

Wie bereits in Kapitel 2.1.2 erläutert, besteht die Aufgabe des adaptiven Filters bei der Echounterdrückung darin, den akustischen Pfad zwischen Lautsprecher und Mikrofon zu

modellieren. Es handelt sich also um den Anwendungsbereich der Systemidentifikation. Wie in Abbildung 2.5 dargestellt, entspricht das Eingangssignal des adaptiven Filters dem Far Speech Signal $x[k]$. Dieses wird parallel mit der Impulsantwort des Filters $h[k]$ und des akustischen Pfades g gefaltet und bildet das Echo Signal $\tilde{x}[k]$), welches zusammen mit dem Near Speech Signal $s[k]$ und Rauschen $n[k]$ im Mikrofonsignal $y[k]$ enthalten ist [50].

$$y[k] = s[k] + n[k] + \tilde{x}[k] \tag{2.4}$$

Das Ausgangssignal des adaptiven Filters $\hat{x}[k]$ wird nun vom Mikrofonsignal subtrahiert, um das Echo zu entfernen. Im Nutzsignal $\hat{s}[k]$ enthalten sind additives Rauschen $n[k]$, das Near Speech Signal $s[k]$ und das Fehlersignal $e[k]$ [50].

$$e[k] = \tilde{x}[k] - \hat{x}[k] \tag{2.5}$$

$$\hat{s}[k] = s[k] + n[k] + e[k] \tag{2.6}$$

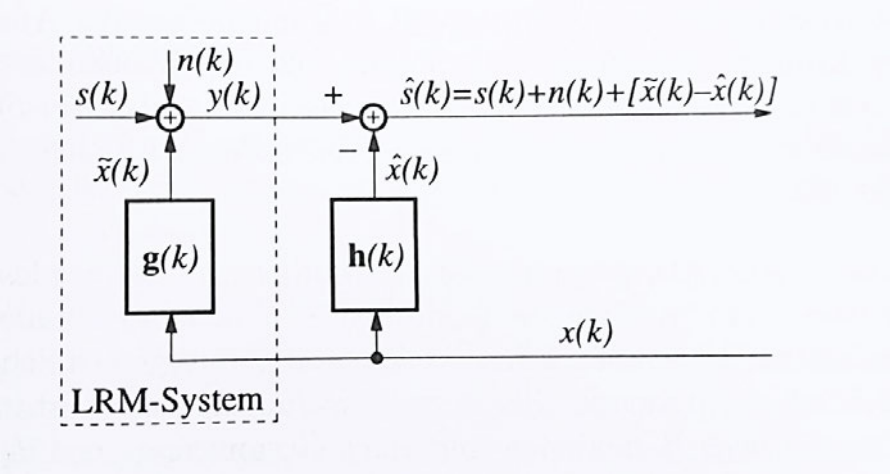

Abbildung 2.5: Blockdiagramm eines adaptiven Filters für AEC [50]

Es wird deutlich, dass das Fehlersignal in einem realen AEC-System nicht isoliert zugänglich ist, jedoch für den Adaptionsalgorithmus benötigt wird. Um dieses Problem zu lösen existieren unterschiedliche Herangehensweisen, welche in Abschnitt 2.2.2 erläutert

werden. Zunächst wird davon ausgegangen, dass nur das Echo Signal im Mikrofonsignal enthalten ist, also der Fall mit $s[k] = 0$ und $n[k] = 0$.

Jede Iteration eines adaptiven Filters mit Filterlänge N besteht im Allgemeinen aus den folgenden Schritten [51]:

1) Berechnung des Filterausgangs $\hat{x}[k]$:
   Dazu wird der Eingangssignalvektor

$$\mathbf{x}[k] = \big(x[k], x[k-1], ..., x[k-N+1]\big)^T \tag{2.7}$$

   mit dem Vektor der Filterkoeffizienten

$$\mathbf{h}[k] = \big(h_0[k], h_1[k], ..., h_{N-1}[k]\big)^T \tag{2.8}$$

   multipliziert (bzw. das Skalarprodukt berechnet):

$$\hat{x}[k] = \mathbf{h}^T[k]) \cdot \mathbf{x}[k] \tag{2.9}$$

2) Berechnung des Fehlersignals:
   Mit $s[k] = 0$ und $n[k] = 0$ lässt sich das Fehlersignal aus der Differenz von Mikrofonsignal $y[k] = \tilde{x}[k]$ und Filterausgang $\hat{x}[k]$ berechnen:

$$e[k] = y[k] - \hat{x}[k] \tag{2.10}$$

3) Anpassung der Filterkoeffizienten mittels Fehlersignal $e[k]$ und Adaptionsvorschrift $\mathcal{F}_\mathcal{A}$:

$$\mathbf{h}[k+1] = \mathcal{F}_\mathcal{A}\{\mathbf{h}[k], e[k], ...\} \tag{2.11}$$

Auch wenn jeder adaptive Filter diese Schritte durchlaufen muss, kann sich die Verarbeitungsstruktur verschiedener Algorithmen stark voneinander unterscheiden. Darauf und auf die Funktionsweise der Adaptionsalgorithmen wird den Kapiteln 2.2.2 und 2.2.3 näher eingegangen.

2.2.1 Beurteilungskriterien

Moschytz & Hofbauer [31] definieren sechs allgemeine Kriterien, nach welchen adaptive Filteralgorithmen beurteilt werden:

- Konvergenzzeit (Wie schnell sich die Filterkoeffizienten an das Optimum annähern)
- Fehleinstellung: (Wie nahe kommt der Algorithmus an das theoretische Optimum heran)
- Nachführverhalten (Wie gut reagiert der Algorithmus auf Änderungen des unbekannten Systems)
- Rechenaufwand pro Iteration
- Numerische Robustheit (Wie sensibel reagiert der Algorithmus auf Rundungs- bzw. Quantisierungsfehler)

Je nach Einsatzgebiet wird ein Algorithmus nach unterschiedlichen dieser Kriterien optimiert und beurteilt. Für die Echounterdrückung bzw. die Systemidentifikation allgemein sind die Konvergenzzeit und die Fehleinstellung von besonderer Bedeutung, da diese in der Regel nicht zugleich optimiert werden können. Für AEC wird die Fehleinstellung durch das bereits in Kapitel 2.1.3 eingeführte Qualitätsmaß ERLE bzw. bei Simulationen durch den Systemabstand repräsentiert. Näheres zum Trade-Off zwischen Konvergenzzeit und Fehleinstellung ist in Kapitel 2.2.2 erläutert. Auch das Nachführverhalten und der Rechenaufwand spielen je nach Einsatzgebiet und Aufgabe des AEC-Algorithmus wichtige Rollen.

2.2.2 Grundlegende Adaptionsalgorithmen

Der Adaptionsalgorithmus ist der Kern eines adaptiven Filters und bestimmt maßgeblich seine Eigenschaften. Das Ziel der Adaption ist die Minimierung des Adaptionsfehlers $e[k]$, welcher laufend neu berechnet wird. Um an das Prinzip der Adaption heranzuführen, wird zunächst der Least-Mean-Squares (LMS) Algorithmus erläutert. Es wird weiterhin davon ausgegangen, dass am nahen Ende weder Rauschen noch Sprache auftritt ($s[k] = 0$ und $n[k] = 0$).

Least-Mean-Squares

Ziel des LMS Algorithmus ist die Minimierung des mittleren quadratischen Adaptionsfehlers [50].

$$E\{e^2[k]\} = E\{\big(\tilde{x}[k] - \hat{x}\big)^2\} \tag{2.12}$$

Da es sich um statistische Zusammenhänge handelt, wird jeweils mit den Erwartungswerten $E\{\}$ der Signale gerechnet. Durch die Berechnung des Gradienten $\nabla[k]$ des mittleren quadratischen Fehlers kann dieser reduziert werden, indem die Koeffizienten des Filters $\mathbf{h}[k]$ in Richtung des negativen Gradienten angepasst werden [51], [50]. Der Gradient beschreibt folglich die Beziehung der Änderung des Kompensationsfehlers zur Änderung des Koeffizientenvektors. Dies entspricht der Gradient-Decent Methode, einem iterativen Optimierungsalgorithmus mit dem Ziel, das lokale Minimum einer Funktion zu finden [31]. Die Berechnung des Gradienten in Vektorform erfolgt durch die partielle Ableitung des mittleren quadratischen Fehlers nach den Filterkoeffizienten. Aus (2.12) folgt:

$$\nabla[k] = \frac{\partial E\{e^2[k]\}}{\partial \mathbf{h}[k]} \tag{2.13a}$$

$$= E\Big\{\frac{\partial(\tilde{x}[k] - \mathbf{h}^T[k]\mathbf{x}[k])^2}{\partial \mathbf{h}[k]}\Big\} \tag{2.13b}$$

$$= E\Big\{2 \cdot \big(\tilde{x}[k] - \mathbf{h}^T[k]\mathbf{x}[k]\big) \cdot \frac{\partial(\tilde{x}[k] - \mathbf{h}^T[k]\mathbf{x}[k])}{\partial \mathbf{h}[k]}\Big\} \tag{2.13c}$$

$$= -2 \cdot E\{e[k] \cdot \mathbf{x}[k]\} \tag{2.13d}$$

Der LMS Algorithmus ersetzt den Gradienten durch einen Momentangradienten

$$\hat{\nabla}(k) = -2e[k]\mathbf{x}[k] \tag{2.14}$$

und führt einen Schrittweitenfaktor $\beta[k]$ ein. Daraus ergibt sich die Adaptionsvorschrift des LMS Algorithmus zu [51]:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \beta[k]e[k]\mathbf{x}[k] \tag{2.15}$$

Damit der Algorithmus konvergiert, muss der Schrittweitenfaktor folgende Bedingung erfüllen [31], [17]

$$0 \leq \beta[k] \leq \frac{2}{\lambda_{max}} \tag{2.16}$$

λ_{max} bezeichnet den Größten der Eigenwerte der Autokorrelationsmatrix

$$R_{xx} = E\{\mathbf{x}^T[k]\mathbf{x}[k]\} \tag{2.17}$$

des Eingangssignalvektors $\mathbf{x}[k]$. Laut Moschytz & Hofbauer [31] muss $\beta[k]$ allerdings bis zu zwei Größenordnungen kleiner gewählt werden als die Obergrenze $\beta_{max} = \frac{2}{\lambda_{max}}$, um eine monoton abnehmende Konvergenz zu gewährleisten. Der Effekt der Schrittweite auf das Konvergenzverhalten des Algorithmus ist in Abbildung 2.6 zu sehen. Dargestellt ist die parabelförmige Krümmung des mittleren quadratischen Fehlers der Filterkoeffizienten für den eindimensionalen Fall $E\{e[k]^2\}$. Der FIR-Filter hat also nur einen Koeffizienten, welcher iterativ optimiert wird. h^0 entspricht dabei dem theoretischen optimalen linearen Filter, der sogenannten Wiener-Lösung, für welche der mittlere quadratische Fehler minimal wird [31].

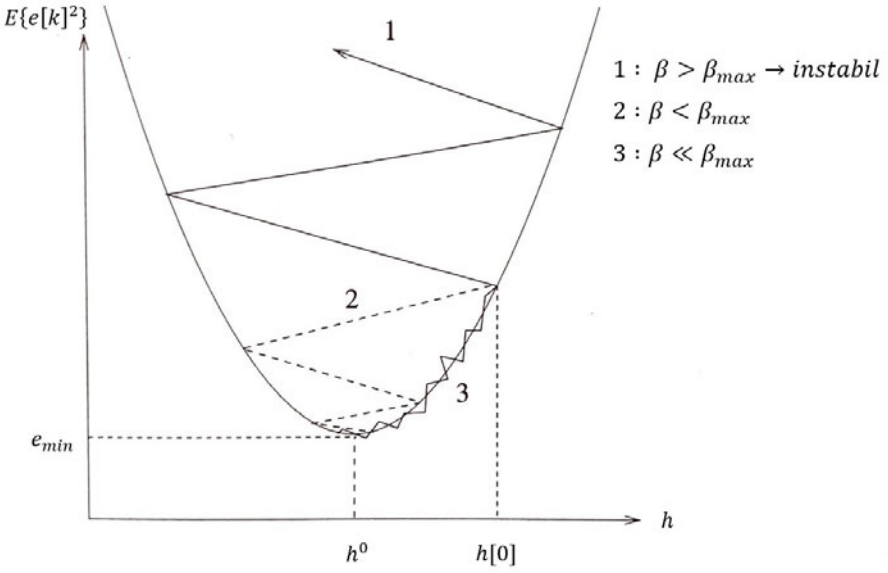

Abbildung 2.6: Konvergenz des LMS Algorithmus in Abhängigkeit der Schrittweite β. Konvergenzpfad 1 zeigt instabiles Verhalten, 2 zeigt eine schnelle und 3 eine langsame Konvergenz. e_{min} bezeichnet den kleinsten erreichbaren quadratischen Fehler. Veränderte Darstellung nach [31]

Normalized LMS

Da der größte Eigenwert λ_{max} der Autokorrelationsmatrix R_{xx} in der Praxis schwierig zu bestimmen ist, stellt der Normalized Least-Mean-Squares (NLMS) Algorithmus eine Möglichkeit dar, die obere Grenze der Schrittweite durch einfach zu bestimmende Größen abzuschätzen [31]. Es lässt sich zeigen, dass eine sichere und konservative obere Grenze der Schrittweite durch die Bestimmung der mittleren Eingangsleistung möglich ist [31], [50]:

$$0 < \beta[k] < \beta_{max} = \frac{2}{|\mathbf{x}[k]|^2} \tag{2.18}$$

Die mittlere Eingangsleistung wird durch das Skalarprodukt

$$|\mathbf{x}[k]|^2 = \mathbf{x}[k]^T \cdot \mathbf{x}[k] \tag{2.19}$$

abgeschätzt.

Im nächsten Schritt wird ein konstanter normierter Schrittweitenfaktor α und eine Sicherheitskonstante γ eingeführt, wodurch sich die Schrittweite des NLMS-Algorithmus ergibt [51], [31]:

$$\beta[k] = \frac{2\alpha}{\gamma + |\mathbf{x}[k]|^2} \tag{2.20}$$

Für α gilt nun die Stabilitätsbedingung

$$0 < \alpha < 1 \tag{2.21}$$

Die Sicherheitskonstante γ wird eingeführt, um zu verhindern, dass die Schrittweite zu groß wird, sollte die Eingangsleistung sehr klein werden [31].

Schließlich ergibt sich die Adaptionsvorschrift des NLMS Algorithmus aus (2.15) und (2.20) [31]:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \frac{2\alpha}{\gamma + |\mathbf{x}[k]|^2} \cdot e[k]\mathbf{x}[k] \quad , \quad 0 < \alpha < 2 \tag{2.22}$$

Der Effekt des normalisierten Schrittweitenparameters α ist in Abbildung 2.7 anhand einer Simulation dargestellt. $D[k]$ bezeichnet den Systemabstand (siehe 2.1.3) nach Itera-

tion k des NLMS Algorithmus. Als Eingangssignal dient in diesem Fall weißes Rauschen. Hier ist deutlich der Trade-Off zwischen großer Schrittweite und schneller Konvergenz auf der einen Seite und kleiner Schrittweite und hoher Echodämpfung bzw. kleinem Systemabstand auf der anderen Seite zu beobachten. Eine der Stärken des NLMS Algorithmus ist ein schnelles Nachführverhalten, also eine schnelle Detektion von Änderungen des LEM-Systems. Dies folgt daraus, dass der NLMS Algorithmus über keinerlei Speicher verfügt, welcher sich einer Änderung des LEM-Systems erst anpassen muss. Allerdings hat dies auch zur Folge, dass der Filter für stärker korrelierte Signale, wie beispielsweise Sprache, nur langsam konvergiert [3]. Eine mögliche Herangehensweise ist die Filterung des Eingangssignals durch einen Dekorrelationsfilter. In Kapitel 2.2.3 wird außerdem kurz der AP Algorithmus (Affine Projection) vorgestellt, welcher dieses Problem auf andere Weise löst.

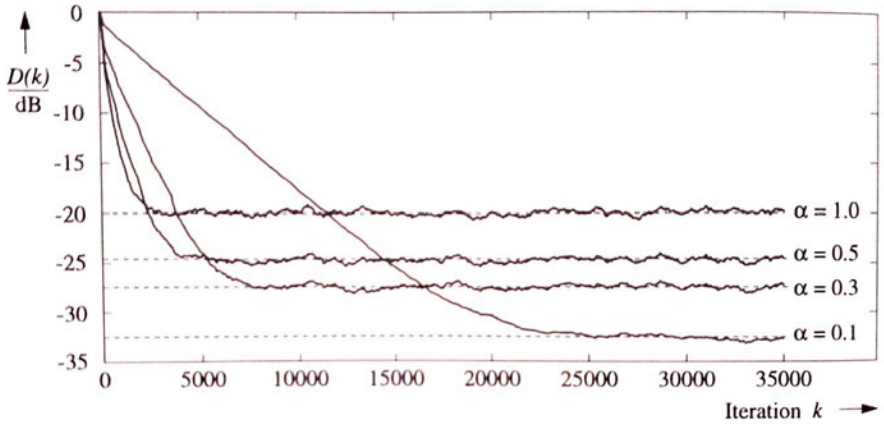

Abbildung 2.7: Konvergenz des NLMS Algorithmus in Abhängigkeit des normalisierten Schrittweitenparameters α. Als Eingangssignal dient weißes Rauschen [51]

Double Talk und adaptive Schrittweite

Bisher wurde vereinfachend angenommen, dass das Mikrofon lediglich das Echo Signal enthält und dieses somit isoliert zur Verfügung steht. In der Realität enthält das Mikrofonsignal allerdings zusätzlich zum Echo Signal das Near Speech Signal und additives Rauschen. Mit Gleichung 2.4 lautet die Adaptionsvorschrift des NLMS Algorithmus

[51]:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \frac{2\alpha}{\gamma + |\mathbf{x}[k]|^2} \cdot y[k]\mathbf{x}[k] \quad , \quad 0 < \alpha < 1 \tag{2.23}$$

Es ergibt sich nun das Problem, dass der LMS (bzw. NLMS) Algorithmus tatsächlich die Leistung des Mikrofonsignals $y[k]$ minimiert und nicht nur die des Fehlersignals $e[k]$ [51]. Die Filterkoeffizienten müssen daher 'eingefroren' werden, sobald Near Speech auftritt. In dem Szenario, dass nur Near Speech vorhanden ist ($x[k] = 0$), ist das kein Problem, da ohne Far Speech Signal keine Adaption stattfinden kann (siehe Formel (2.22)). In dem Fall, dass beide Kommunikationsparteien zeitgleich sprechen (Double Talk), wird die Adaption allerdings gestört. Da aber die Leistung des Mikrofonsignals im Double Talk Szenario in der Regel deutlich erhöht ist, lässt sich dies durch Messung der Momentanleistung des Mikrofonsignals detektieren und die Adaption kann gestoppt werden.

Alternativ kann auch ein adaptiver Schrittweitenfaktor eingesetzt werden, welcher die optimale Schrittweite pro Adaptionsschritt schätzt. Dieser verkleinert sich bei der Präsenz von Near Speech so weit, dass praktisch keine Adaption des Filters mehr stattfindet [51]. Es muss demnach derjenige Schrittweitenfaktor gefunden werden, für den der mittlere Systemabstand minimal wird. Um das lokale Minimum des mittleren Systemabstandes zu finden, wird die mittlere Änderung des Systemabstandes eines Adaptionsschrittes bestimmt, diese partiell nach dem Schrittweitenfaktor abgeleitet und auf Null gesetzt [51]. Dieses Verfahren beruht auf der Schätzung des Systemabstandes und erhöht zusätzlich die Latenz des adaptiven Filters. Außerdem müssen zusätzliche Maßnahmen getroffen werden, um Änderungen der Systemantwort des realen Systems zu detektieren [50].

Komplexere Verfahren der optimalen Schrittweitenbestimmung nutzen im Frequenzbereich vorliegende Signale, um die Schrittweite β sowohl in Abhängigkeit des Zeit- als auch des Frequenzverhaltens des Eingangssignals $x[k]$ zu optimieren [50]. Dies bringt nicht nur Vorteile in Bezug auf Double Talk, sondern verbessert auch das Verhalten in Bezug auf Interferenzen und variierende LEM-Impulsantworten [18] [50]. Bismor et al. [4] vergleichen 17 unterschiedliche Verfahren der Schrittweitenanpassung von adaptiven Filtern anhand des LMS Algorithmus und kommen dabei zu dem Schluss, dass die Effektivität einzelner Algorithmen stark von der jeweiligen Anwendung abhängt. Einen Überblick über verschiedene Verfahren der Schrittweitenbestimmung speziell für AEC-Filter geben Mader et al. [29]. Aktuelle Verfahren basieren auf Deep Learning, was zwar mehr Res-

sourcen verbraucht [29], dafür aber signifikante Verbesserungen, besonders für variable akustische Umgebungen, mit sich bringt [15] [57].

2.2.3 Adaptive Filter für AEC

Auf Grundlage des LMS und NLMS Algorithmus existieren verschiedene adaptive Filter. Diese lassen sich grob unterteilen nach der Art des Verarbeitung (Fullband-, Block- oder Subband Processing), sowie nach Zeit- und Frequenzbereichsverfahren [3]. In den Kapiteln 2.3 und 2.4 werden exemplarisch der Fast-Least-Mean-Squares Algorithmus (Block-Processing, Frequenzbereich) und der Subband-AEC Algorithmus mit NLMS (Subband-Processing, Zeitbereich) im Detail erläutert. Beide Algorithmen wurden im Hinblick auf ihre Effizienz sowie auf die angewandten Methodiken der klassischen Signalverarbeitung ausgewählt. Um ein vollständigeres Bild der Möglichkeiten zu bieten, werden im Folgenden zwei weitere adaptive Filteralgorithmen erwähnt, welche für diese Anwendung in Betracht gezogen, jedoch aus Gründen der Komplexität und Ineffizienz nicht weiter verfolgt wurden.

Der Affine Projection (AP) Algorithmus [34] ist eine Verallgemeinerung des NLMS Algorithmus und vermeidet das Problem, dass die Geschwindigkeit der Konvergenz von den Eigenschaften des Eingangssignals abhängt [3]. Dazu wird der Gradient zur Anpassung der Filterkoeffizienten nicht alleine mittels des aktuellen Eingangsvektors bestimmt, sondern innerhalb einer mehrdimensionalen Ebene (Hyperplane), welche von den letzten $M-1$ Eingansvektoren aufgespannt wird. Dieser Algorithmus besitzt zwar eine schnellere Konvergenz als der NLMS Algorithmus, allerdings erfordert er die Inversion einer $M \times M$ Matrix für jeden Adaptionsschritt und ist damit deutlich rechenintensiver [3].

Der Recursive Least Squares (RLS) Algorithmus dekorreliert das Eingangssignal, indem er die Autokorrelationsmatrix rekursiv abschätzt, invertiert und mit dem Eingangssignal multipliziert. Es muss also ebenfalls für jeden Adaptionsschritt eine Matrix invertiert werden. Durch die Rekursivität lässt sich dies jedoch beschleunigen [50], [3]. Der RLS-Algorithmus besitzt damit eine vergleichsweise hohe Konvergenzgeschwindigkeit, ist dafür aber numerisch sehr komplex und teilweise instabil [50], [3].

2.3 Fast Least Mean Squares

Der Fast Least Mean Squares (FLMS) Algorithmus wird auch als Frequency Domain Adaptive Filter (FDAF) [50] oder Frequency Domain - LMS [31] bezeichnet. Er beruht auf dem Block-LMS Algorithmus, nutzt aber die Eigenschaften der DFT, um den Rechenaufwand mittels FFT und schneller Faltung zu reduzieren [50]. Der Block-LMS Algorithmus und damit auch der FLMS Algorithmus gleichen in ihrem Konvergenzverhalten dem LMS Algorithmus, da der Gradient zwar durch die Mittlung genauer geschätzt wird, gleichzeitig aber die Koeffizienten nur alle L Takte nachgestellt werden [31]. Das Ablaufdiagramm des Algorithmus ist in Abbildung 2.8 abgebildet.

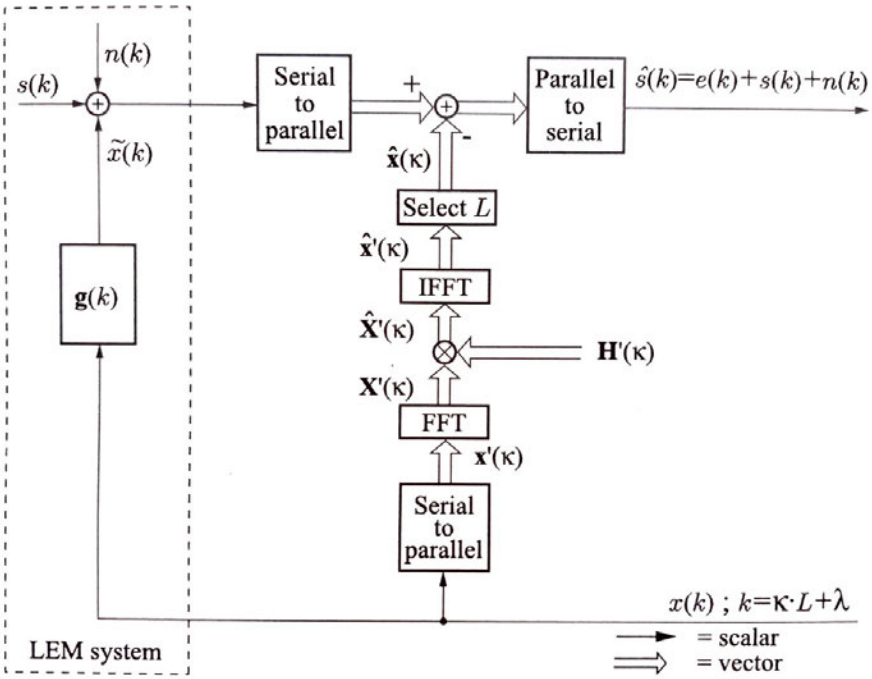

Abbildung 2.8: Blockdiagramm des FLMS Algorithmus [50]

Zunächst wird der bisher verwendete Zeitindex k durch einen Ausdruck mit Blockindex κ, Blocklänge L und Sampleindex λ ersetzt. [50].

$$k = \kappa L + \lambda; \qquad \lambda = 0, 1, ..., L-1 \tag{2.24}$$

2.3.1 Filterung des Eingangssignals

Im ersten Schritt wird das Eingangssignal mit den Filterkoeffizienten gefaltet. Dies erfolgt durch die *Overlap-Save* Methode. Auch die Anwendung des ähnlichen *Overlap-Add* Algorithmus ist denkbar, allerdings für die Anwendung des FLMS Algorithmus weniger effizient [13]. Die *Overlap-Save* Methode ist ein Algorithmus zur linearen, zirkulären Faltung. Um bei einer Impulsantwort der Länge m L valide Ausgangssamples zu erhalten, muss die Länge der FFT

$$M = m - 1 + L \tag{2.25}$$

gewählt werden [50]. Nur die letzten L Samples des Ergebnisses der Faltung sind valide, mit denen der regulären linearen Faltung übereinstimmende Samples. Die restlichen Werte sind aufgrund der zirkulären Faltung verfälscht [13]. Für die FFT der Länge M muss der Koeffizientenvektor $h[\kappa]$ mit $L-1$ Nullen auf dieselbe Länge erweitert werden (zero-padding)[31].

$$\mathbf{h}'[\kappa] = \Big[h_0[\kappa], ..., h_{m-1}[\kappa], 0, ..., 0\Big]^T \tag{2.26}$$

Das Eingangssignal muss ebenfalls M Samples lang sein und besteht entsprechend des *Overlap* - Prinzips aus L neuen und $m-1$ alten Samples. Hierbei ist zu beachten, dass der Signalvektor $\mathbf{x}'[\kappa]$ im Vergleich zum vorherigen Signalvektor $\mathbf{x}[k]$ gespiegelt ist, das aktuellste Sample steht am Ende des Vektors [50].

$$\mathbf{x}'[\kappa] = \Big[x'_0[\kappa], ..., x'_{M-1}[\kappa]\Big]^T \tag{2.27a}$$

$$= \Big[x[\kappa L - m + 1], ..., x[\kappa L], x[\kappa L + 1], ..., x[\kappa L + L - 1]\Big]^T \tag{2.27b}$$

Signal- und Koeffizientenvektor werden in den Frequenzbereich transformiert

$$\mathbf{H}'[\kappa] = FFT\{\mathbf{h}'[\kappa]\} \tag{2.28}$$

$$\mathbf{X}'[\kappa] = FFT\{\mathbf{x}'[\kappa]\} \tag{2.29}$$

und elementweise multipliziert [31].

$$\hat{\mathbf{X}}'[\kappa] = \mathbf{H}'[\kappa] \odot \mathbf{X}'[\kappa] \tag{2.30}$$

Der Operator $\odot$ bezeichnet hier die elementweise Multiplikation zweier Vektoren.

Das Ergebnis der Rücktransformation

$$\hat{\mathbf{x}}'[\kappa] = FFT^{-1}\{\hat{\mathbf{X}}'[\kappa]\} \tag{2.31}$$

entspricht der linearen zyklischen Faltung. Durch einfache Fensterung können nun die ersten $m-1$ Werte entfernt werden. Das Ergebnis ist ein Signalvektor der Länge L welcher mit dem Ergebnis der linearen Faltung von Eingangssignalblock und Filterimpulsantwort übereinstimmt [51].

$$\hat{\mathbf{x}}[\kappa] = \mathbf{w} \odot \hat{\mathbf{x}}'[\kappa] \tag{2.32a}$$

$$= \mathbf{x}[\kappa] \circledast \mathbf{h}[\kappa] \tag{2.32b}$$

Das dazu benötigte Fenster $\mathbf{w}$ wird beschrieben durch:

$$w_i = \begin{cases} 0 \ ; \ i = 0, ..., m-2 \\ 1 \ ; \ i = m-1, ..., M-1 \end{cases} \tag{2.33}$$

2.3.2 Adaption der Filterkoeffizienten

Auch die Adaption der Filterkoeffizienten kann in den Frequenzbereich verlagert werden, um Rechenaufwand zu sparen. Dazu wird zunächst die Adaptionsvorschrift für den Block-LMS Algorithmus betrachtet [50]:

$$\mathbf{h}[k+L] = \mathbf{h}[k] + \beta[k] \sum_{\lambda=0}^{L-1} e[k+\lambda]\mathbf{x}[k+\lambda] \tag{2.34}$$

So werden alle L Takte die Filterkoeffizienten mit den zwischengespeicherten Fehlerwerten $e[k], ..., e[k+L-1]$ angepasst. Durch die Einführung des mittleren Gradienten

$$\hat{\nabla}[k+L-1] = \frac{1}{L} \sum_{\lambda=0}^{L-1} e[k+\lambda]\mathbf{x}[k+\lambda] \tag{2.35}$$

lässt sich die Vorschrift in der Form

$$\mathbf{h}[k+L] = \mathbf{h}[k] + L\beta[k]\hat{\nabla}[k+L-1] \tag{2.36}$$

darstellen [50].

Für die Adaptionsvorschrift des FLMS Algorithmus wird nun (2.36) mit Berücksichtigung von (2.26) in die Blocknotation überführt [51].

$$\mathbf{h}'[\kappa+1] = \mathbf{h}'[\kappa] + \beta[\kappa]\hat{\nabla}'[\kappa] \tag{2.37}$$

Die einzelnen Elemente des Gradientenvektors $\hat{\nabla}'[\kappa]$ berechnen sich wie folgt:

$$\hat{\nabla}_i'[\kappa] = \begin{cases} \sum_{\lambda=0}^{L-1} e[\kappa L+\lambda]x[\kappa L+\lambda-i] & ;\ i = 0,1,...,m-1 \\ 0 & ;\ i = m, m+1, ..., M-1 \end{cases} \tag{2.38}$$

Es wird deutlich, dass die Elemente $\hat{\nabla}_i'[\kappa]$ durch Korrelation des Fehlersignals $\mathbf{e}[\kappa]$ mit dem Eingangssignal $\mathbf{x}[\kappa]$ berechnet werden [50]. Da die Korrelation sich lediglich durch die Spiegelung eines der Signale von der Faltung unterscheidet, kann auch dieser Vorgang durch den *Overlap-Save* Algorithmus beschleunigt werden. Dazu werden die Signalblöcke analog zu Kapitel 2.3.1 Fourier-transformiert und elementweise multipliziert [51].

$$\mathbf{H}'[\kappa] = FFT\{\mathbf{h}'[\kappa]\} \tag{2.39}$$

$$\hat{\nabla}'[\kappa] = FFT\{\hat{\nabla}'[\kappa]\} \tag{2.40}$$

$$\mathbf{H}'[\kappa+1] = \mathbf{H}'[\kappa] + \beta[\kappa]\hat{\nabla}'[\kappa] \tag{2.41}$$

Um den *Overlap-Save* Algorithmus anwenden zu können, muss zunächst der Fehlervektor analog zu (2.26) mit Nullen auf die Länge $M = m - 1 + L$ erweitert werden. Da es sich hier um die schnelle Korrelation handelt, müssen die Nullen vorne angefügt werden. Anschließend wird der Fehlersignalvektor $\mathbf{e}[\kappa]$, genauso wie der Eingangssignalvektor $\mathbf{x}'[\kappa]$ Fourier-transformiert.

$$\mathbf{e}'[\kappa] = \Big[0, ..., 0, e_0[\kappa], ..., e_L[\kappa]\Big] \tag{2.42}$$

$$\mathbf{E}'[\kappa] = FFT\{\mathbf{e}'[\kappa]\} \tag{2.43}$$

$$\mathbf{X}'[\kappa] = FFT\{\mathbf{x}'[\kappa]\} \tag{2.44}$$

Der transformierte Eingangssignalvektor $\mathbf{X}'[\kappa]$ wird vor der elementweisen Multiplikation komplex konjugiert, was einer Spiegelung im Zeitbereich entspricht.

$$\tilde{\nabla}[\kappa] = \mathbf{E}'[\kappa] \odot \mathbf{X}'^{*}[\kappa] \tag{2.45}$$

Anschließend wird das Ergebnis invers Fourier-transformiert, um die letzten $L-1$ Werte mittels Fensterung auf Null zu setzen und so den Ausdruck aus (2.38) zu erreichen.

$$\hat{\nabla}'[\kappa] = \tilde{\mathbf{w}} \cdot FFT^{-1}\{\tilde{\nabla}[\kappa]\} \tag{2.46}$$

$$\tilde{w}_i = \begin{cases} 1 & ; \ i = 0, 1, ..., m-1 \\ 0 & ; \ i = m, m+1, ..., M-1 \end{cases} \tag{2.47}$$

Das Blockdiagramm der Adaptionsvorschrift ist in Abbildung 2.9 dargestellt.

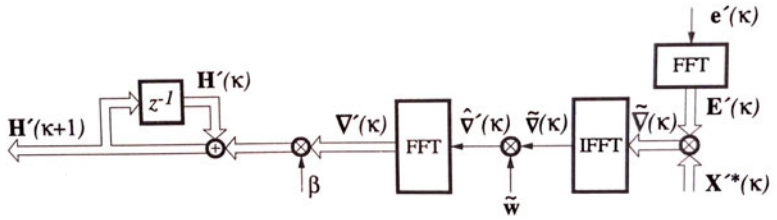

Abbildung 2.9: Blockdiagramm der Adaptionsvorschrift des FLMS Algorithmus [51]

Da die Adaption der Koeffizienten im Frequenzbereich stattfindet, wird der Schrittweitenfaktor $\beta[\kappa]$ auf die geschätzte spektrale Leistungsdichte normiert. Er hängt demnach von der geschätzten Leistung der jeweiligen Frequenz (bzw. des jeweiligen Frequency-Bins) ab [40].

$$\beta[\kappa] = \frac{2\alpha}{\gamma + \mathbf{P}[\kappa]} \tag{2.48}$$

$$\mathbf{P}[\kappa] = \lambda \mathbf{P}[\kappa - 1] + (1 - \lambda)|\mathbf{X}[\kappa]| \odot |\mathbf{X}[\kappa]| \quad , 0 < \lambda < 1 \tag{2.49}$$

Die Leistungsdichte wird durch einen exponentiell geglätteten Durchschnitt geschätzt. Dabei bestimmt der 'Forgetting Factor' λ das Gedächtnis der Mittelung, also wie stark aktuelle Werte gewichtet werden [40]. Bei der Initialisierung muss eine initiale Leistung P_0 definiert werden.

2.4 Subband AEC

Der Subband AEC bezeichnet eine Gruppe von Algorithmen, welche sich die Methode der Zerlegung des Signalspektrums in Teilbänder zunutze machen. Jedes des Teilbänder (Subbänder) hat seinen eigenen adaptiven Filter [13]. Auf Grund der durch Unterabtastung reduzierten Abtastfrequenz und Breite des Teilsignalspektrums genügt für die einzelnen Subbänder ein wesentlich kleineres Filter, wodurch Ressourcen gespart werden und der Algorithmus schneller konvergiert [13]. Der in den Subbändern verwendete AEC Algorithmus ist der in Kapitel 2.2.2 erläutere NLMS Algorithmus. Abbildung 2.10 zeigt das Ablaufdiagramm des Subband Algorithmus.

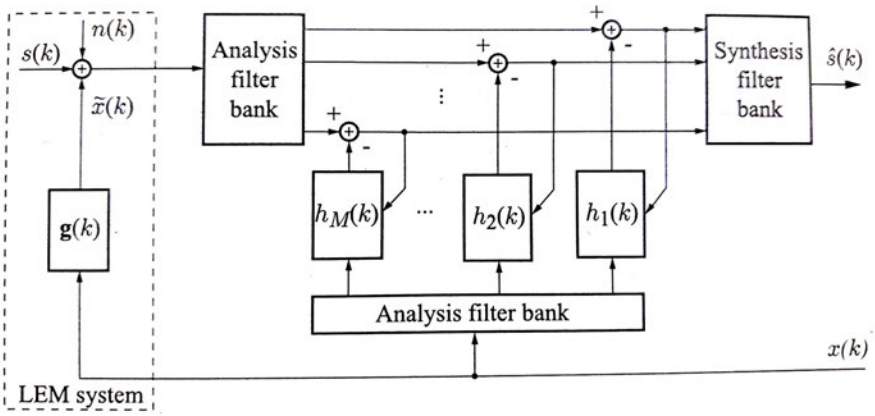

Abbildung 2.10: Ablaufdiagramm des Subband AEC Algorithmus. $h_1[k], ..., h_M[k]$ sind die adaptiven Filterkoeffizienten der einzelnen Subbänder [50]

Im Folgenden wird die Aufteilung eines Signals in Teilbänder (Subband Decomposition) sowie die Zusammenführung der Teilsignale zu einem Ausgangssignal mittels Filterbänken erläutert.

2.4.1 Quadrature-Mirror-Filterbank

Grundlage der Zerlegung und Zusammenführung der Teilbänder ist eine Quadrature-Mirror-Filterbank (QMF), bestehend aus Analyse- und Synthesefilterbank (siehe Abbildung 2.11) [25]. Die Filter H_i der Analysefilterbank sind Bandpassfilter und haben die Aufgabe, das Eingangssignal in N Teilsignale X_i zu zerlegen, sodass jedes der Teilsignale einen Teil des Frequenzbandes des Eingangsignals repräsentiert [25]. Die Filter werden so

entworfen, dass sich ihre Aliasing-Komponenten gegenseitig auslöschen (Perfect Reconstruction), weshalb QMFs auch als Aliasing-frei bezeichnet werden [25]. Da die Breite der Frequenzbänder der Teilsignale nur noch $1/N$ des originalen Frequenzbandes beträgt, ist es nun möglich, die Teilsignale mit einem Faktor von bis zu $1/N$ zu dezimieren und so bei einer niedrigeren Abtastfrequenz zu verarbeiten [13]. Die Dezimation um den maximalen Faktor N wird als kritische Dezimation bezeichnet [25].

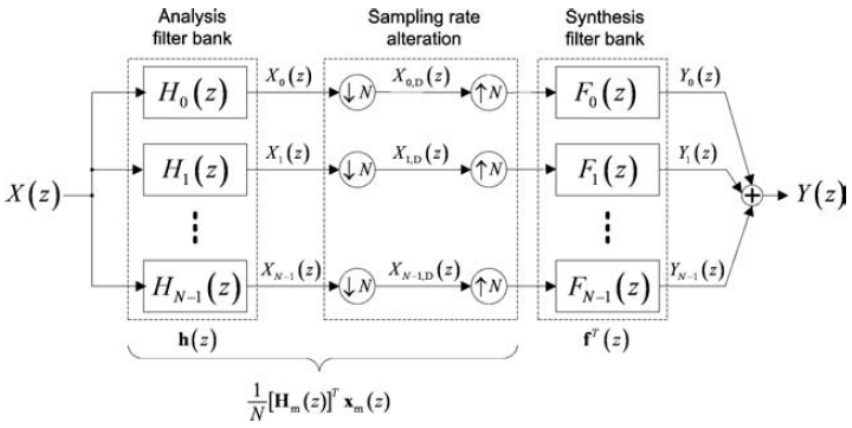

Abbildung 2.11: Schema einer kritisch dezimierten QMF [25]

Anschließend werden die Teilsignale mit dem Faktor N interpoliert, um die ursprüngliche Abtastrate wiederherzustellen. Die Filter F_i der Synthesefilterbank sorgen anschließend dafür, dass die Teilbandsignale zu einem Ausgangssignal kombiniert werden. Die Ausgangs- Eingangsrelation der QMF wird beschrieben durch:

$$Y(z) = \Big[\frac{1}{N}\sum_{i=0}^{N-1} F_i(z)H_i(z)\Big]X(z) \tag{2.50}$$

2.4.2 Polyphasenzerlegung

Die Effizienz der Filterbank kann durch den Einsatz von Polyphasen gesteigert werden, sodass auch die Analyse- und Synthesefilter auf der um $1/N$ dezimierten Abtastfrequenz berechnet werden [13]. Jeder Analysefilter H_i wird dazu in jeweils N Polyphasenkompo-

nenten $E_{i,r}$ zerlegt (eine kritische Dezimation mit Faktor $1/N$ vorausgesetzt) [25].

$$H_i(z) = \sum_{r=0}^{N-1} E_{i,r}(z^N) \cdot z^{-r}, \quad i = 0, 1, ..., N-1 \tag{2.51}$$

$$E_{i,r}(z) = \sum_{k=0}^{K-1} h_i(kN + r) \cdot z^{-k} \tag{2.52}$$

$E_{i,r}$ repräsentiert dabei die r-te Polyphasenkomponente des i-ten Analysefilters H_i. $h_i(k)$ ist die Impulsantwort des jeweiligen Filters $H_i(z)$. Die Länge der Impulsantworten beträgt $L = K \cdot N$, K ist die Länge der Polyphasenkomponenten.

Unter Ausnutzung der Noble Identities lässt sich nun die Dezimation vor den Filter schieben, sodass pro Takt der schnellen Abtastrate nicht mehr der gesamte Filter, sondern lediglich eine Polyphasenkomponente berechnet werden muss. Die Ergebnisse der Polyphasenkomponenten werden addiert [25]. So gilt für einen Analysefilter $H_i(z^{-N})$ bei einer um N dezimierten Abtastfrequenz:

$$H_i(z^{1/N}) = \sum_{r=0}^{N-1} E_{i,r}(z) \cdot z^{-r/N} \tag{2.53}$$

Der Term $z^{-r/N}$ repräsentiert eine Verzögerungskaskade, welche den Polyphasenkomponenten vorgelagert ist.

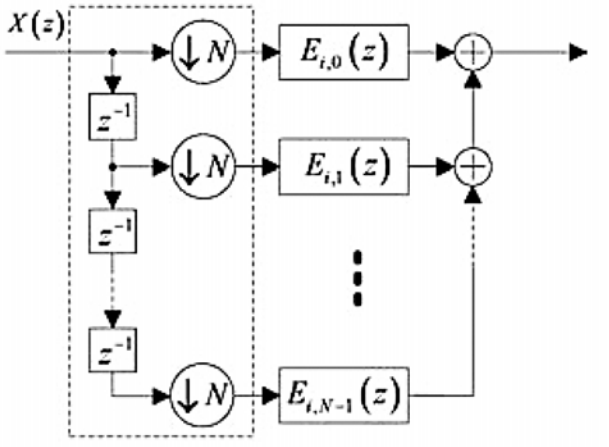

Abbildung 2.12: Polyphasenzerlegung eines Analysefilters H_i als Blockdiagramm [25]

Jedes Verzögerungsglied der Kaskade entspricht der Verzögerung des Signals um einen Takt der schnellen Abtastfrequenz, weshalb der Dezimator in der Praxis nicht bis hinter die Kaskade geschoben wird (siehe Abbildung 2.12). Laut Lee et al. [25] wird diese Art der Umwandlung als Polyphasenzerlegung *Typ I* bezeichnet.

Analog wird mit den Synthesefiltern F_i verfahren, wobei nach Lee et al. [25] die Polyphasenzerlegung *Typ II* angewandt wird. Diese zeichnet sich dadurch aus, dass sie die Zerlegung beim letzten Koeffizienten der Impulsantwort beginnt, also eine Permutation der Polyphasenzerlegung *Typ I* ist: $R_r(z) = E_{N-1-r}(z)$. Damit ergibt sich für die Synthesefilter F_i:

$$F_i(z) = \sum_{r=0}^{N-1} R_{i,r}(z^N) \cdot z^{-(N-1-r)}, \quad i = 0, 1, ..., N-1 \tag{2.54}$$

$$R_{i,r}(z) = \sum_{n=0}^{K-1} f_i(nN + N - 1 - r) \cdot z^{-n} \tag{2.55}$$

Nun werden die Interpolatoren nach vorne durch die Filter bewegt, wodurch diese auf der niedrigeren Abtastfrequenz berechnet werden (siehe Abbildung 2.13).

$$F_i(z^{1/N}) = \sum_{r=0}^{N-1} R_{i,r}(z) \cdot z^{-(N-1-r)/N} \tag{2.56}$$

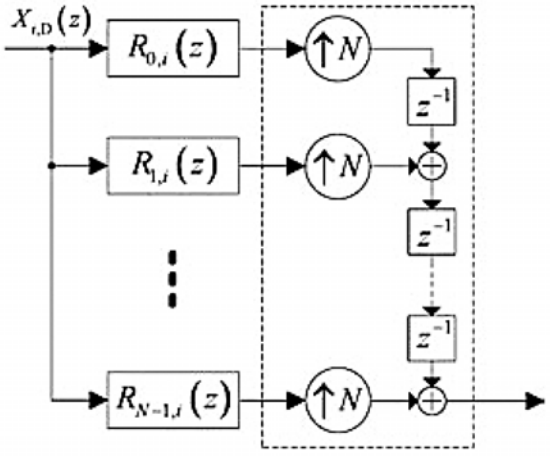

Abbildung 2.13: Polyphasenzerlegung eines Synthesefilters F_i als Blockdiagramm [25]

Wird diese Umwandlung auf jeden der Analyse- und Synthesefilter angewandt lässt sich gesamte Filterbank vereinfacht darstellen (siehe Abbildung 2.14). Dazu werden Polyphasenkomponenten der Filter in jeweils einer Analyse- und Synthesematrix zusammengefasst [25].

$$E(z) = \begin{bmatrix} E_{0,0}(z) & E_{0,1}(z) & \dots & E_{0,N-1}(z) \\ E_{1,0}(z) & E_{1,1}(z) & \dots & E_{1,N-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ E_{N-1,0}(z) & E_{N-1,1}(z) & \dots & E_{N-1,N-1}(z) \end{bmatrix} \tag{2.57}$$

$$R(z) = \begin{bmatrix} R_{0,0}(z) & R_{0,1}(z) & \dots & R_{0,N-1}(z) \\ R_{1,0}(z) & R_{1,1}(z) & \dots & R_{1,N-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ R_{N-1,0}(z) & R_{N-1,1}(z) & \dots & R_{N-1,N-1}(z) \end{bmatrix} \tag{2.58}$$

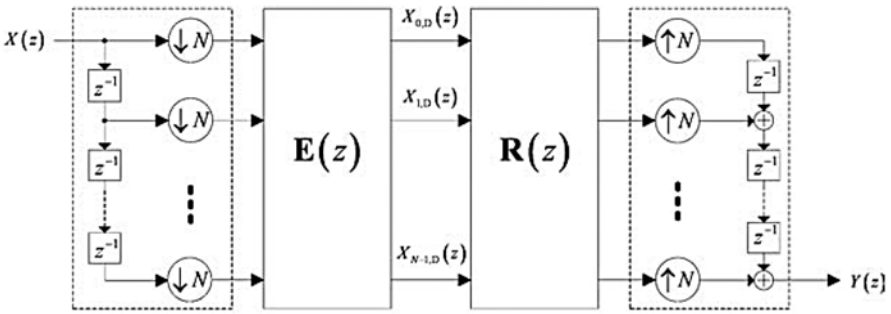

Abbildung 2.14: Vereinfachtes Blockdiagramm der Filterbank [25]

2.4.3 Entwurf der Filterbank

Der Entwurf der Analyse- und Synthesefilter lässt sich zu Entwurf, Modulation und Transformation eines einzigen Tiefpassfilters vereinfachen [13]. Grundlage ist ein Prototypfilter $P(z)$ der Länge L mit der Eckfrequenz $\frac{\pi}{2N}$ (siehe Abbildung 2.15), wobei N die Anzahl der Subbänder darstellt.

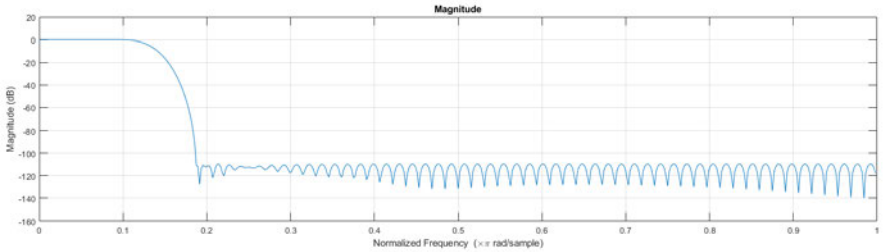

Abbildung 2.15: Beispiel des Frequenzgangs eines Tiefpass-Prototypfilters mit Eckfrequenz $\frac{\pi}{2N}$ und $N = 8$

Die Analysefilter werden durch Modulation des Protoypfilters erzeugt [25].

$$H_i(z) = \alpha_i P(z \cdot W_{2N}^{(i+0.5)}) + \alpha_i^* P(z \cdot W_{2N}^{-(i+0.5)}), \quad i = 0, 1, ..., N-1 \tag{2.59}$$

Der Modulationsfaktor $W_{2N} = e^{-j\pi/N}$ bestimmt, wie weit der Filter moduliert wird, der Faktor α_i berechnet sich wie folgt [25]:

$$\alpha_i = exp\Big\{ j\big(\theta_i - \frac{\pi}{N}(i+0.5)\frac{L-1}{2}\big)\Big\} \quad , \theta_i = (-1)\frac{\pi}{4} \tag{2.60}$$

Im Zeitbereich entspricht dies der Multiplikation mit einem Cosinussignal:

$$h_i(n) = 2p(n)cos\Big[\frac{\pi}{N}(i+0.5)\big(n - \frac{L-1}{2}\big) + \theta_i\Big] \tag{2.61}$$

Abbildung 2.16 zeigt die modulierten Prototypfilter der Analysefilterbank für $N = 4$ Subbänder.

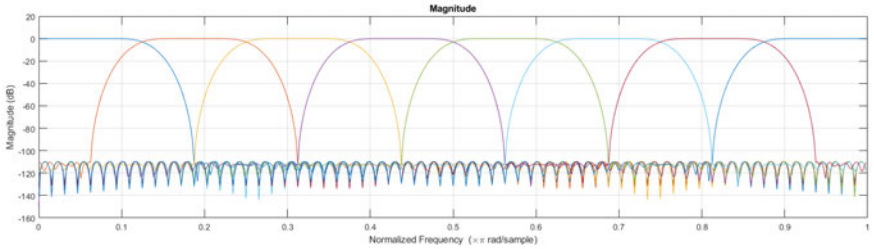

Abbildung 2.16: Beispiel der aus dem Prototypfilter erzeugten Analysefilter für $N = 8$ Subbänder

Die Synthesefilter sind durch folgende Transformation aus den Analysefiltern berechenbar [25]

$$F_i(z) = z^{-L+1} H_i(z^-1) \tag{2.62}$$

Eine so entworfene Filterbank wird als Pseudo-QMF Filterbank bezeichnet und kann durch bestimmte Designbedingungen des Prototypfilters eine annähernd perfekte Rekonstruktion des Signals erreichen. Annähernd perfekt bedeutet in diesem Fall, dass strukturell nur die Aliasingkomponenten der benachbarten Frequenzbänder ausgelöscht werden. Die Aliasingkomponenten aus entfernten Bändern werden als ausreichend gedämpft angenommen [25]. Für die annähernd perfekte Rekonstruktion werden zwei wichtige Designbedingungen an den Prototypfilter gestellt.

Um die Aliasingkomponenten der benachbarten Bänder auszulöschen, muss folgende, als *Flatness constraint* bezeichnete Bedingung bestmöglich erfüllt werden.

$$|P(e^{j\omega})|^2 + |P(e^{j(\omega-\pi/N)})|^2 \tilde{=} 1 \quad , 0 < \omega < \pi/N \tag{2.63}$$

Um die Aliasingkomponenten weiter entfernter Bänder zu unterdrücken, muss die Dämpfung des Stopbands so hoch wie möglich sein.

$$|P(e^{j\omega})| \tilde{=} 0 \quad , \omega > \pi/N \tag{2.64}$$

2.5 Ergänzende Maßnahmen

In der praktischen Anwendung kann ein AEC mit linearem adaptivem Filter das auftretende Echo nicht immer ausreichen dämpfen. Beispielsweise in Szenarien, bei welchen sich die LEM Impulsantwort schlagartig ändert oder die zur Verfügung stehenden Ressourcen für längere Filter nicht ausreichen [50] [51]. Zwei der zusätzlichen Maßnahmen, welche für die weitere Reduktion des Echos zuständig sind, werden im Folgenden kurz erläutert. Daneben existieren noch eine Reihe weitere, komplexere Herangehensweisen. Dies können beispielsweise frequenzselektive Postfilter sein, welche psychoakustische Effekte ausnutzen, um die Wahrnehmung des Restechos zu reduzieren. Außerdem kann ein Postfilter dazu genutzt werden, das Leistungsdichtespektrum des Restechos abzuschätzen, um so die Schrittweite des Adaptionsalgorithmus genauer zu steuern [32]. Aktuellere Methoden verwenden dazu meist auf Deep Learning basierende Ansätze [16].

2.5.1 Center Clipper

Ein Center Clipper wird definiert durch einen Schwellwert A. Liegt der Betrag des Ausgangssignals $\hat{s}[k]$ darunter, wird das Signal unterdrückt, um die Reste des Echos zu entfernen [51].

$$\tilde{s}[k] = \begin{cases} \hat{s}[k] & ; \hat{s}[k] > +A \\ 0 & ; |\hat{s}[k]| \leq +A \\ \hat{s}[k] & ; \hat{s}[k] < -A \end{cases} \tag{2.65}$$

Der Schwellwert sollte dabei möglichst niedrig sein, um die Kommunikation nicht zu beeinträchtigen. Auch eine adaptive Einstellung der Schwelle ist möglich [50].

2.5.2 Pegelwaage

Eine Pegelwaage kann auch dann eingesetzt werden, wenn das Restecho noch einen höheren Pegel hat, beispielsweise bei der Initialisierung des adaptiven Filters [51]. Sie funktioniert nach dem Prinzip, dass immer das Signal, welches einen niedrigeren Pegel hat, gedämpft wird. In Verbindung mit einem AEC werden zunächst Ein- und Ausgangssignal gleich gedämpft. Steigt nun der Pegel am Eingangssignal, so wird die Dämpfung am Eingang (a_x) reduziert und am Ausgang (a_y) erhöht. So hat der adaptive Filter Zeit, sich an die LEM-Impulsantwort anzupassen. Mit steigender Dämpfung des AEC kann die Dämpfung der Pegelwaage reduziert werden [51].

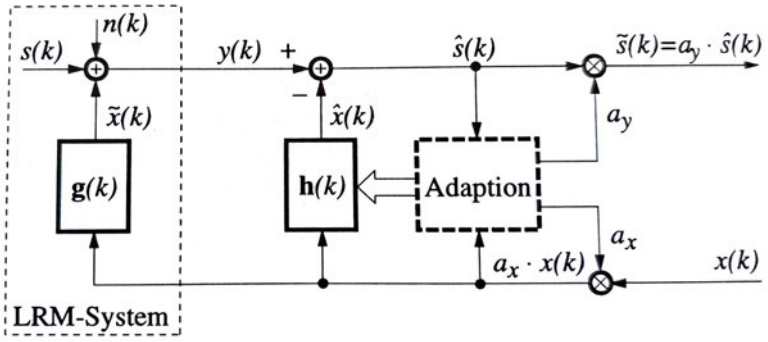

Abbildung 2.17: Schema eines Echokompensators mit Pegelwaage [51]

2.6 Anforderungen

Anschließend an die theoretischen Grundlagen sollen nun die Anforderungen definiert werden. Zusammengefasst ist die Anforderung an diese Arbeit die Realisierung eines Demonstrators für akustische Echounterdrückung, welcher dem Einsatz in der Lehre im Rahmen eines Laborprojektes dienen soll. Er soll Studierenden im Rahmen eines Projektes die Möglichkeit bieten, erlernte und neue Methoden der Signalverarbeitung in der Praxis umzusetzen und zu demonstrieren. Daher sind neben Anforderungen, wie der Höhe der Echodämpfung oder der Effizienz, auch didaktische Aspekte zu berücksichtigen, beispielsweise die Übereinstimmung der verwendeten Methoden mit denen, welche in den entsprechenden Modulhandbüchern [19], [20], [21] aufgeführt sind. Die einzelnen Anforderungen werden in Tabelle 2.1 ausgeführt.

Tabelle 2.1: Auflistung der Anforderungen

Qualität	• Es soll ein signifikanter Effekt akustisch eindeutig wahrnehmbar und in der Signalverarbeitung analytisch eindeutig messbar sein. • Sowohl die subjektiv empfundene als auch die instrumentell gemessene Echodämpfung soll spezifiziert werden.
Latenz	• Das System soll eine ausreichend hohe Latenz aufweisen, sodass das akustische Echo eindeutig wahrgenommen werden kann. Hierbei ist zu berücksichtigen, dass Echos, welche nach weniger als 25 ms eintreffen, im Allgemeinen nicht wahrgenommen werden [39] und Latenzen von über 400 ms (One-way) für Echtzeitkommunikation nicht akzeptabel sind [23]. • Die entsprechenden Latenzzeiten sind zu quantifizieren.
Hardware	• Das System soll auf der Hardware, welche den Studierenden im Labor zur Verfügung steht, implementierbar und lauffähig sein. Die zur Verfügung stehende Hardware ist in Tabelle 2.2 aufgeführt.

Software	• Verwendete Software und Programmiersprachen sollen den Studierenden bereits bekannt sein.
Algorithmen	• Die verwendeten Methoden sollen in den durch die Modulhandbücher festgelegten Lehrinhalten der klassischen Signalverarbeitung enthalten sein, oder darauf aufbauen. • Es sollen möglichst viele durch die Lehre bereits bekannte Methoden der klassischen Signalverarbeitung Anwendung finden. • Es sollen keine der klassischen Signalverarbeitung fremden Methoden verwendet werden (beispielsweise Machine Learning).

Tabelle 2.2: Zur Verfügung stehende Hardware

Menge	Bezeichnung
2	Prozessor-Board UniDAQ2.DSP-ADDA (D.SignT) [10]
2	EM280 Electret Cardioid Microphone (Yo-tronics)
1	4 Kanal Mikrofonverstärker
2	2.4 GHz Wireless Transmitter Model 2266T (Jesmay)
2	2.4 GHz Wireless Receiver Model 2266R (Jesmay)
2	XDS200 USB Debug Probe (Spectrum Digital)
2	Mini Cubes 2.0 Lautsprecher (Ultron)

3 Konzeption und Design

In diesem Kapitel wird unter Berücksichtigung der in Kapitel 2 definierten Anforderungen und Grundlagen zunächst ein Konzept entworfen und anschließend dessen Simulation und Umsetzung beschrieben.

3.1 Konzept

Der konzeptionelle Aufbau des Systems (siehe Abbildung 3.1) sieht die Verwendung von zwei Kommunikationsstationen vor, welche durch eine Funkverbindung miteinander verbunden sind. Jede der Stationen ist von identischem Aufbau und setzt sich aus jeweils einem DSP-Board, Mikrofon, Lautsprecher sowie Funksender und -empfänger zusammen. Jede der Stationen entfernt das an ihrem Ende entstehende akustische Echo. Die DSPs sind jeweils mit einem PC verbunden, von welchem die Algorithmen entsprechend der Anforderungen zur Laufzeit gesteuert werden können.

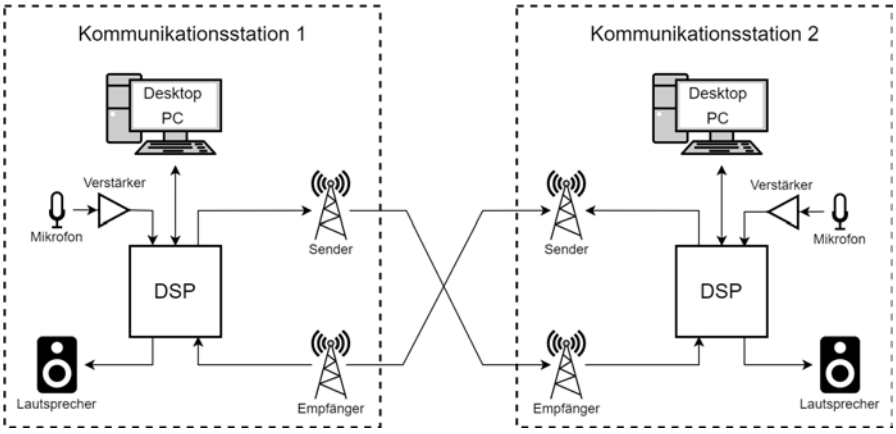

Abbildung 3.1: Konzeptioneller Aufbau des Demonstrators im Doppelbetrieb

Die Demonstration des Systems ist auch mit nur einer einzelnen Kommunikationstation möglich, wodurch die Arbeit im Labor vereinfacht wird. Dazu sind jedoch jeweils zwei Mikrofone und Lautsprecher notwendig (siehe Abbildung 3.2).

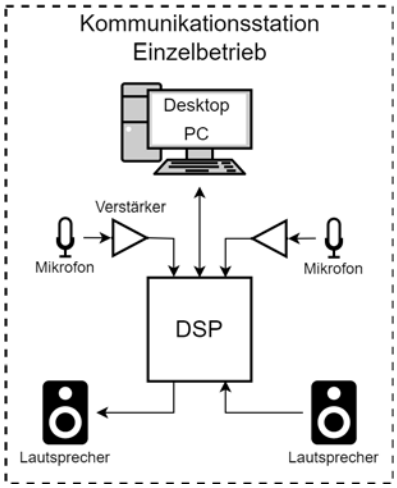

Abbildung 3.2: Konzeptioneller Aufbau des Demonstrators im Einzelbetrieb

In den Kapiteln 2.3 und 2.4 wurden zwei AEC Algorithmen exemplarisch vorgestellt, welche sich den Anforderungen gemäß beide für die Umsetzung eignen. Der Subband AEC Algorithmus verwendet bekannte Methoden der Signalverarbeitung wie Unterabtastung und Subband Dekomposition und Polyphasenzerlegung. Der FLMS Algorithmus verwendet ebenfalls bekannte Methoden wie die FFT, die Filterung von Signalen im Frequenzbereich durch die schnelle Faltung (Overlap-Save Algorithmus), die Fensterung von Signalen und allgemein die Blocksignalverarbeitung. Daher werden beide Algorithmen zunächst in Matlab simuliert, um anschließend einen der beiden auf der Hardware zu implementieren. Die Simulationen sollen nicht nur dem Zweck der Auswahl des passenden Algorithmus dienen, sondern sollen den Studierenden auch die Möglichkeit geben, den Algorithmus in einer Umgebung zu untersuchen und zu testen, welche portabel und leicht zu debuggen ist. Für die Simulationen werden mit der Zielhardware Testsignale aufgenommen und in Matlab importiert.

Um Komplexität und Zeitrahmen der Implementierungen zu verringern, wird außerdem nach bereits bestehenden Lösungen recherchiert. Für den FDAF (bzw. FLMS) existiert bereits die Matlab-Funktion *dsp.FrequencyDomainAdaptiveFilter*, als Teil der DSP System Toolbox [49]. Für diese Funktion existiert ein Beispiel, welches die Verwendung dieses adaptiven Filters zur akustischen Echounterdrückung demonstriert [47]. Auch wenn die

Funktion nicht Open-Source ist, eignet sich dieses Beispiel als Ausgangspunkt der Simulation. Es existieren auch weitere Implementierungen des FDAF. Hervorzuheben ist hier Xu [54], welcher neben FDAF auch viele weitere adaptive Filteralgorithmen in Python implementiert und dokumentiert. Für Subband AEC existieren ebenfalls bereits Implementierungen in Matlab. Als Grundlage und Orientierung dient hier eine Umsetzung auf Grundlage von Gilloire & Vetterli [14], welche verschiedene Filterbänke und Implementierungen des Subband Algorithmus zur Verfügung stellt [1]. Erwähnenswert ist auch die Implementierung eines Subband Kalman AEC Filters in Matlab [6]. Da trotz intensiver Recherche keine offen zugänglichen Implementierungen der Algorithmen in C für DSPs gefunden wurden, dient der Matlab-Code als Vorlage, mit Hilfe derer der entsprechende Algorithmus auf dem DSP implementiert werden soll.

Ein mit der zur Verfügung gestellten Hardware aufgenommenes Testsignal soll dazu dienen, die Eignung der Algorithmen mittels Simulation möglichst realitätsnah festzustellen.

3.2 Simulation

Wie bereits in Kapitel 3.1 beschrieben, ist das Ziel der Simulation zum einen, die beiden in Kapitel 2 exemplarisch erläuterten Algorithmen mit Testsignalen zu untersuchen und zu entscheiden, welcher der beiden auf dem DSP implementiert werden soll. Zum anderen dient die Simulation als Vorlage für die Studierenden, welche diese dazu Nutzen sollen, den Algorithmus zu untersuchen und den Einfluss verschiedener Parameter zu testen. Entsprechend den Anforderungen dient Matlab R2023b als Entwicklungsumgebung der Simulation.

Als Testsignale wurden mit dem in Abbildung 3.1 dargestellten Aufbau auf einem der DSPs Signale von 3 Sekunden Länge und mit einer Abtastrate von 16 kHz aufgezeichnet. Dabei handelt es sich das ungestörte Lautsprechersignal und das durch das Echo gestörte Mikrofonsignal. Die aufgezeichneten Signale wurden anschließend in eine von Matlab interpretierbare Datei gespeichert. Da sowohl Mikrofonverstärker als auch Funkempfänger unterschiedliche Gleichspannungen einbringen, werden die Testsignale vor ihrer Verwendung mittelwertbefreit. Außerdem werden die Signalamplituden auf das Intervall $[-1, 1]$ normalisiert. Als Qualitätsmaß wird das ERLE-Maß verwendet, wobei vor der Berechnung jeweils 1000 Samples gemittelt werden, um eine ausreichende Glättung zu erreichen.

3.2.1 FLMS Matlabsimulation

Ausgangspunkt der Simulation des FLMS Algorithmus ist das Matlab-Beispiel zu Acoustic Echo Cancellation [47]. Nachfolgend wurden die in Matlab integrierten Funktionen und Operationen durch eigene Implementierungen ersetzt, mit dem Ziel, einen möglichst einfach nach C zu überführenden Code zu erstellen. Das Ablaufdiagramm der FLMS Simulation ist in Abbildung 3.3 dargestellt. Im Folgenden wird näher auf das zugrundeliegende Matlab Beispiel sowie die Implementierungen des FLMS und der FFT eingegangen.

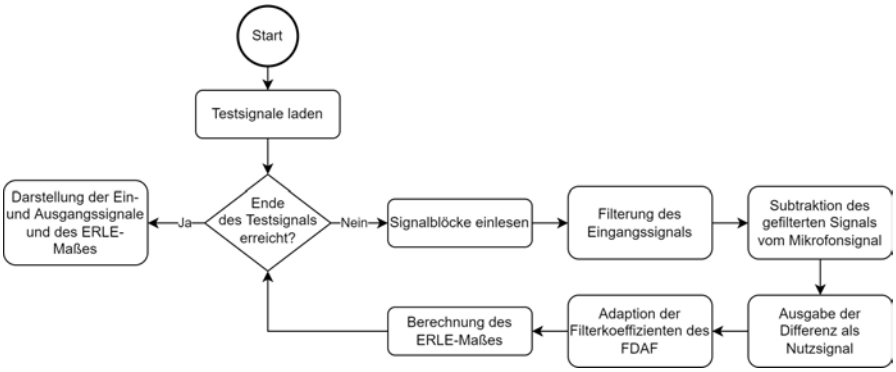

Abbildung 3.3: Ablaufdiagramm der FLMS Simulation

AEC Matlab-Beispiel

Das AEC Matlab-Beispiel demonstriert die Anwendung der Matlab-Funktion *dsp.FrequencyDomainAdaptiveFilter* [49] mit einem 30-sekündigen Testsignal der Abtastfrequenz von 16 kHz, wobei das Echo nicht real, sondern künstlich mit Hilfe einer zuvor erzeugten Raumimpulsantwort generiert wird. Die relevanten Parameter des adaptiven Filters sind auf die in Tabelle 3.1 aufgeführten Parameter voreingestellt.

Das Ergebnis der Echounterdrückung in Abbildung 3.4 zeigt deutlich die Effektivität des adaptiven Filters: das errechnete ERLE-Maß zeigt Echodämpfungen von über 40 dB an. Ebenso gut erkennbar ist die Konvergenz des Filters, welcher das nach der Initialisierung auftretende Echo nur wenig dämpft, sich aber mit der Zeit immer genauer an die Raumimpulsantwort adaptiert und so das ERLE-Maß mit der Zeit steigt.

Tabelle 3.1: Relevante Parameter des FDAF des AEC Matlab-Beispiels

Parameter	Beschreibung	Wert
Length	Länge des FIR-Filters	2048
StepSize	Schrittweite der Adaption der Koeffizienten	0.025
InitialPower	Initiale Signalleistung für die Berechnung der normierten Schrittweite	0.01
AveragingFactor	Forgetting factor des exponentiell geglätteten Durchschnitts der Leistungsdichte (siehe Formel (2.49))	0.98

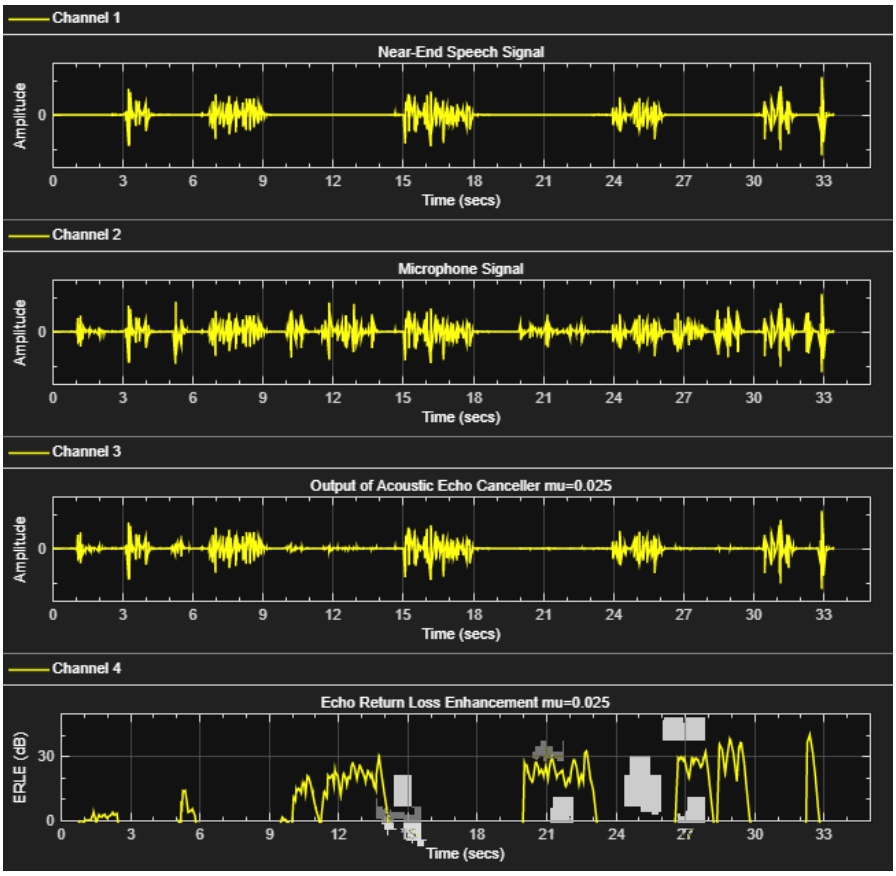

Abbildung 3.4: Ergebnis des FLMS Matlab Beispiels mit voreingestellten Parametern

Im nächsten Schritt wurden die Testsignale von Matlab durch die eigens zu Simulationszwecken aufgezeichneten Testsignale ersetzt. Um die Ergebnisse vergleichbar zu machen, wurden die 3-sekündigen Signale so kombiniert, dass sowohl am nahen Ende als auch am entfernten Ende gesprochen wird. Außerdem werden die Signale auf eine Länge von ca. 33 Sekunden erweitert. Die beiden Testsignale sind oben in Abbildung 3.5 dargestellt und repräsentieren das Far Speech Signal $x[k]$, welches aus den Lautsprechern kommt und das Mikrofonsignal $y[k]$, welches das Near Speech Signal sowie das Echo enthält.

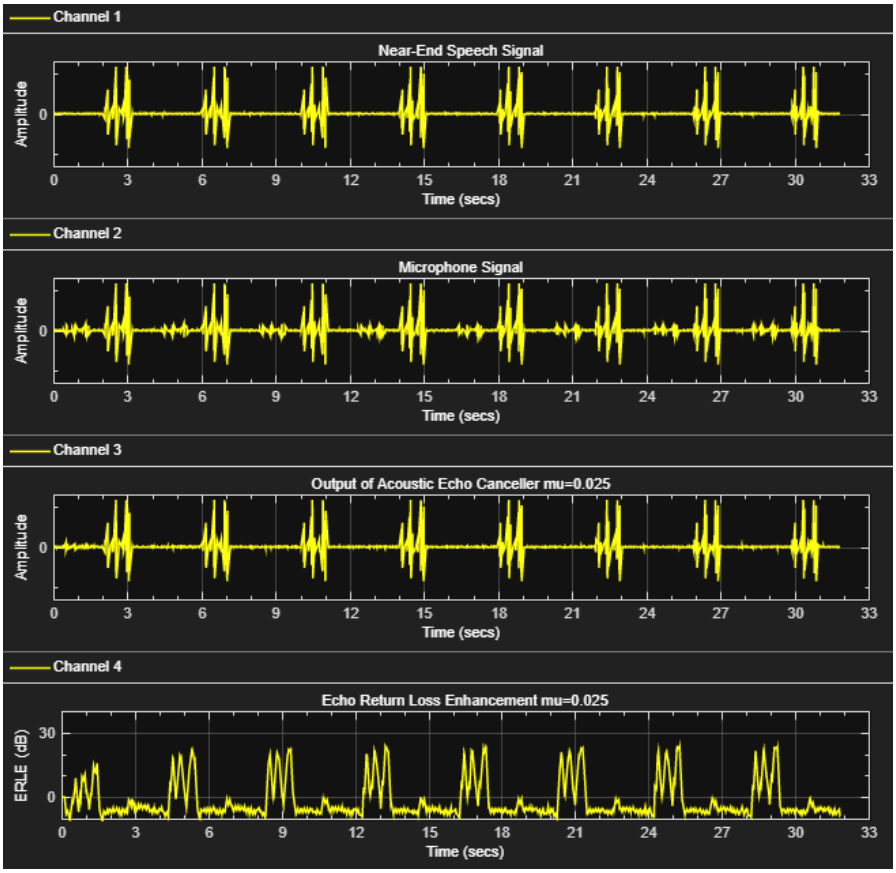

Abbildung 3.5: Ergebnis des FLMS Matlab Beispiels mit eigenem Testsignal

Abbildung 3.5 zeigt eine deutliche Echoreduktion, ebenso wie die zunehmende Konvergenz des adaptiven Filters. Die Echodämpfung steigt hier von anfänglich etwa 10 dB bis knapp über 20 dB an. Das Ergebnis ist allerdings nur begrenzt mit dem des Matlab-Testsignals vergleichbar, da es einen deutlich niedrigeren Pegel und weniger Echo hat. Außerdem handelt es sich um eine reale Umgebung und nicht um ein künstlich erzeugtes

Echo. Dennoch wird die Eignung des adaptiven Filters für die reale akustische Umgebung deutlich.

FLMS Implementierung

Im nachfolgenden Schritt wurde die Matlab-Funktion *dsp.FrequencyDomainAdaptiveFilter* durch eine eigene Implementierung des FLMS Algorithmus ersetzt. Für die Berechnung der FFT bzw. IFFT werden die Matlab Funktionen *fft* bzw. *ifft* eingesetzt, für Vektor- und Matrixberechnungen werden die in Matlab integrierten Operatoren verwendet. Testsignal und Parameter bleiben unverändert.

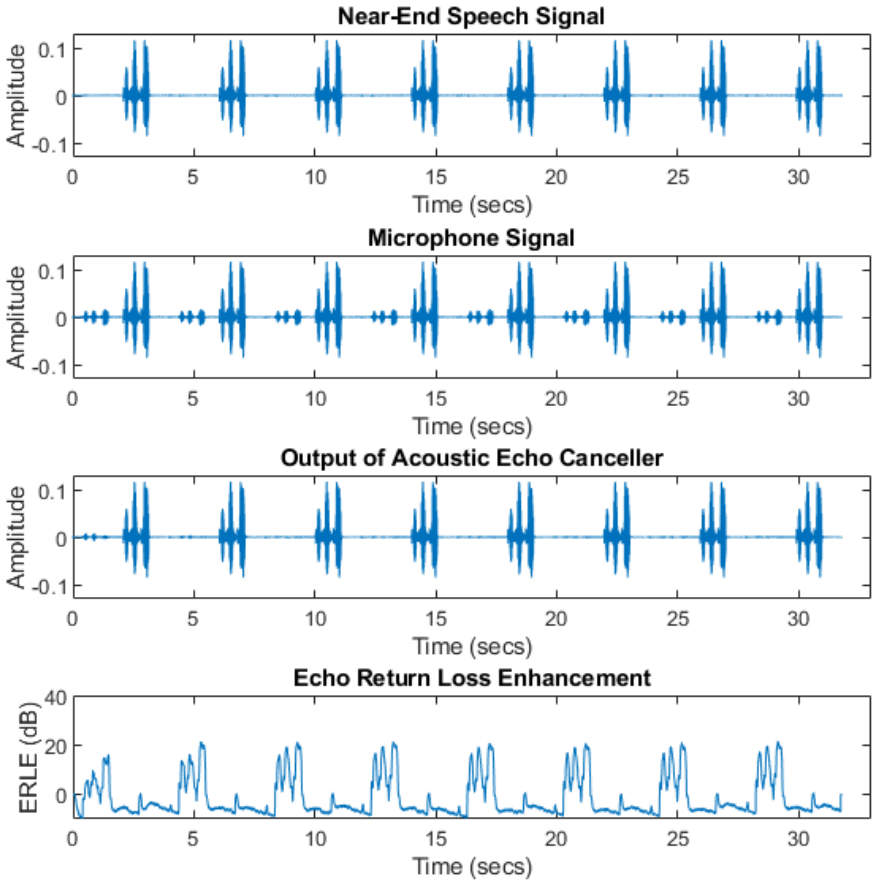

Abbildung 3.6: Ergebnis der Simulation nach Implementierung des FLMS Algorithmus

Abbildung 3.5 zeigt das Ergebnis der FLMS Implementierung. Diese erreicht, genau wie die implementierte Funktion des Matlab-Beispiels, Echodämpfungen von über 20 dB.

Komplexe Datentypen und Operationen

Zur Vorbereitung auf die FFT und später die C-Implementierung wurden alle Variablen des Matlab Datentyps *complex double* durch Datenstrukturen des Datentyps *double* ersetzt. Dabei werden Real- und Imaginärteil der komplexen Werte im Speicherblock jeweils hintereinander gespeichert. In Folge wurden daher komplexe Multiplikationen und Divisionen manuell implementiert.

Fast-Fourier-Transformation

Zuletzt wurde die externe Matlab-Funktion *fft* durch eine eigene Implementierung ersetzt. Als Vorlage diente dazu die von Press et al. in Kapitel 12.2.1 beschriebene FFT [37]. Diese verwendet das Danielson-Lanczos Lemma und basiert auf von N.M. Brenner ursprünglich in FORTRAN geschriebenem Code [37].

Es handelt sich um eine In-Place-FFT, welche die ursprünglichen Daten ersetzt und dadurch entsprechend ressourcenschonend ist. Außerdem erlaubt diese Implementierung die Berechnung der inversen Fourier Transformation durch den zusätzlichen Parameter *isign*. Die Implementierung der FFT kann in diesem Anwendungsfall noch weiter optimiert werden, indem man die Berechnung der Twiddlefaktoren nur einmal durchführt. Dies ist möglich, da diese lediglich von der Länge der FFT abhängen, welche während der Laufzeit des Algorithmus konstant bleibt. Außerdem ist die Auslagerung des Bitreversing durch Erstellung einer Adressmap möglich, sodass die Samples bereits vorher in die entsprechende Reihenfolge gebracht werden.

Die Optimierungen wurden, genau wie eine Fixpunkt-Implementierung, zu Testzwecken simuliert. Dabei stellte sich jedoch heraus, dass die Auflösung der FFT im Fixpunktformat auf der Hardware nicht ausreichend wäre. Da sich die Performance der nichtoptimierten FFT als ausreichend herausstellte, wurden die Optimierungsmöglichkeiten nicht weiter verfolgt.

3.2.2 Subband AEC Matlabsimulation

Vorlage der Simulation des Subband AEC Algorithmus ist die Implementierung von Aironi [1]. Wie in Kapitel 2.4 beschrieben, bildet der Prototypfilter die Grundlage des

Subband AEC Algorithmus. Nach dessen Entwurf und Optimierung wurde dieser entsprechend zu den Analyse- und Synthesefiltern der Filterbank moduliert und transformiert. Die so erhaltenen Filter wurden anschließend in ihre Polyphasen zerlegt. Zuletzt wurde für jeden der Kanäle ein adaptiver Filter mit dem NLMS Algorithmus implementiert. Das Ablaufdiagramm der Subband AEC Simulation ist in Abbildung 3.7 dargestellt, Tabelle 3.2 zeigt die für die Simulation verwendeten Parameter. Das Ergebnis der Simulation ist in Abbildung 3.8 abgebildet. Anschließend wird näher auf den Entwurf des Prototypfilters und der Filterbänke eingegangen.

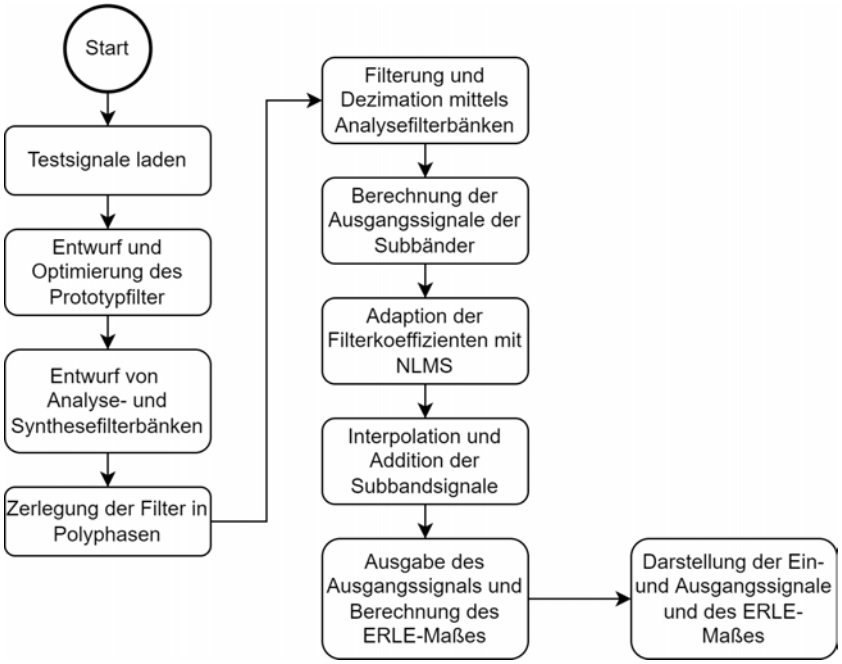

Abbildung 3.7: Ablaufdiagramm der Subband AEC Simulation

Tabelle 3.2: Parameter der Subband AEC Matlabsimulation. Angepasst nach [1]

Parameter	Wert
Länge des Prototypfilters	32
Anzahl der Subbänder	4
Länge der adaptiven Filter	144
Schrittweite	0.4
Averaging Factor (s. Tab. 3.1)	0.5

Die Parameter wurden teilweise von der Vorlage [1] übernommen. Um die Rechenzeit zu reduzieren wurde die Anzahl der Subbänder, die Länge des Prototypfilters, sowie die Länge der adaptiven Filter angepasst.

Das Ergebnis der Simulation (Abbildung 3.8) zeigt Echodämpfungen zwischen 10 und 20 dB. Auffällig ist außerdem die schnelle Konvergenz der adaptiven Filter. Dies ist daran zu erkennen, dass sich ihre Echodämpfung mit der Zeit nicht merklich steigert (Differenz < 1 dB). Dies war aufgrund der großen Schrittweite und der kurzen Filterlängen zu erwarten.

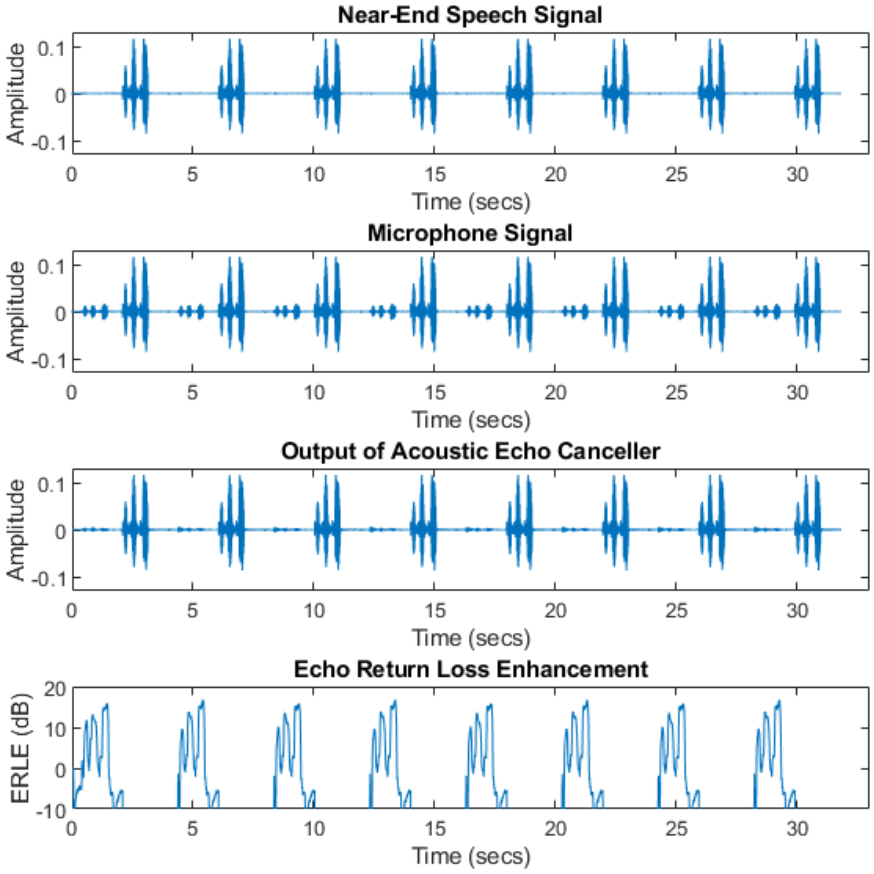

Abbildung 3.8: Ergebnis der Simulation des Subband AEC Algorithmus

Entwurf des Prototypfilters

Der Prototypfilter wurde mit dem in Matlab integrierten Parks-McClellan Algorithmus entworfen und anhand des in Kapitel 2.4 erläuterten *Flatness Constraint* iterativ opti-

miert. Dazu wurde eine bereits existierende Funktion der Vorlage [1] verwendet. Abbildung 3.9 zeigt den Frequenzgang des Filters.

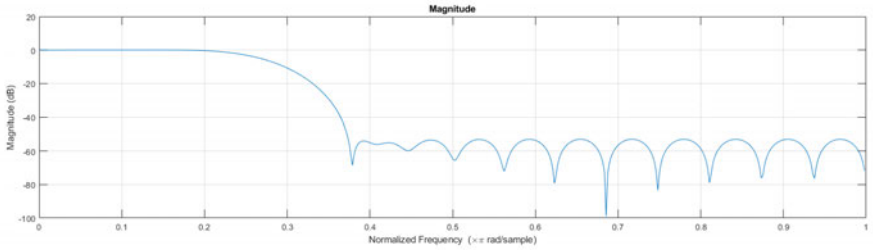

Abbildung 3.9: Frequenzgang des optimierten Prototypfilters

Entwurf der Filterbänke

Analyse- und Synthesefilter wurden durch Modulation aus den Koeffizienten des Prototypfilters gewonnen (siehe Abbildung 3.10). Auch für diesen Vorgang kann auf die Vorlage [1] zurückgegriffen werden. Anschließend werden diese in ihre Polyphasenkomponenten zerlegt und in einem dreidimensionalen Array gespeichert. Durch Zero-Padding wird sichergestellt, dass alle Polyphasenkomponenten die gleiche Anzahl an Koeffizienten haben.

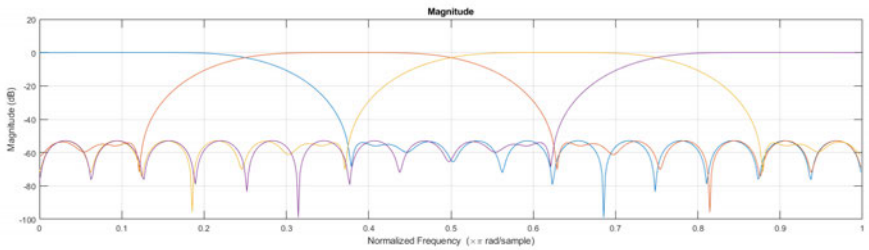

Abbildung 3.10: Frequenzgänge der Analysefilter

3.2.3 Vergleich von FLMS und Subband AEC Simulation

Ein Vergleich der Simulationsergebnisse zeigt zunächst, dass zwar beide Algorithmen signifikante Echodämpfungen erreichen. Allerdings wird anhand der Abbildung 3.11 deutlich, dass der FLMS Algorithmus konsistent höhere ERLE-Werte erreicht, als der Subband AEC, aber auch, dass der Subband AEC Algorithmus deutlich schneller konvergiert.

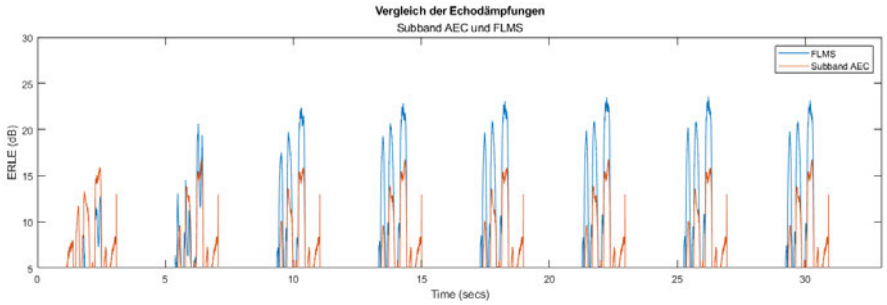

Abbildung 3.11: Vergleich der Echodämpfungen von FLMS und Subband AEC

Auffällig ist auch ein deutlicher Unterschied in Bezug auf den Verbrauch an Ressourcen, vor allem an Rechenzeit. Während der FLMS Algorithmus für die Verarbeitung des Testsignals Rechenzeiten im Bereich von 320 Millisekunden aufweist, benötigt der Subband AEC Algorithmus rund 870 Millisekunden.

3.3 Umsetzung

In diesem Kapitel wird zunächst die Simulation in C und anschließend die Implementierung des FLMS Algorithmus auf dem DSP beschrieben. Die Simulation in C soll den Übergang zwischen Matlab und C Code vereinfachen. Sie bietet außerdem eine Möglichkeit, den Code ohne zusätzliche Hardware zu testen, da diese den Studierenden unter Umständen nicht immer zur Verfügung steht. Dazu wird auch die Verwendung von mex-Files untersucht. Außerdem wird auf Hardware- und Board Setup sowie Speichermanagement und Interrupt Service Routinen eingegangen. Als Entwicklungsumgebung für die C-Simulation diente Visual Studio 2022, für die Implementierung auf dem DSP wurde Code Composer Studio 12.5 verwendet.

3.3.1 Verwendung von mex-Files

Mex-Files bzw. mex-Functions sind eine Möglichkeit, ein C/C++ Programm aus Matlab aufzurufen [48]. Um den Übergang zwischen Matlab und C Code zu vereinfachen, bietet sich die Verwendung von mex-Functions daher an. Die erstellte Funktion muss dabei der von Matlab definierten API entsprechen und die abstrakte Klasse *matlab::mex::Function* erben und erweitern. Es ist demnach möglich, einzelne Funktionen, wie beispielsweise die FFT, in C zu schreiben und mit der bereits bestehenden Matlab-Simulation zu testen.

Allerdings erwies sich diese Möglichkeit in der Praxis als zu komplex, um den Übergang zu vereinfachen. Viele der C Funktionen greifen direkt auf Speicherblöcke zu und arbeiten mit Pointern, was durch die mex-API und die Matlab spezifischen Datentypen und -strukturen erschwert wird. Daher wurde entschieden, den Matlab-Code direkt nach C zu übertragen.

3.3.2 FLMS C-Simulation

Die Übertragung des Algorithmus von Matlab nach C wird dadurch vereinfacht, dass die komplexen Datentypen und externen Funktionen bereits im Matlab Code durch eigene Implementierungen ersetzt wurden (siehe Kapitel 3.2.1).

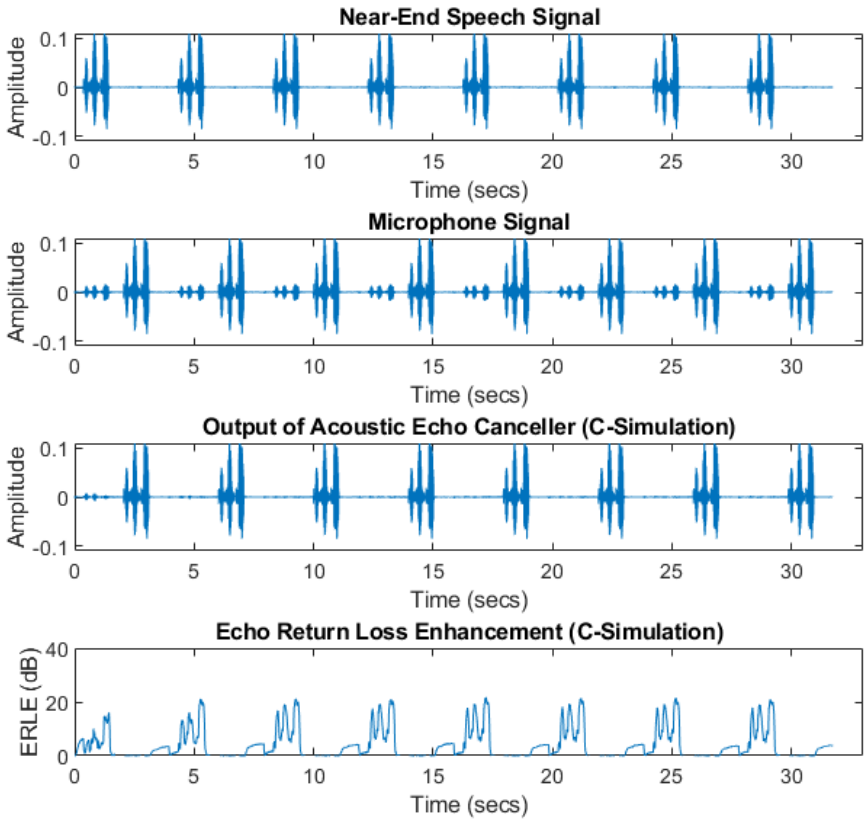

Abbildung 3.12: Ergebnis der C-Simulation des FLMS Algorithmus

Da das vorrangige Ziel der C Simulation die Umsetzung des Algorithmus ist, wird auf die Simulation von Interrupt-Service-Routinen verzichtet, zudem werden die Testsignale

analog zur Matlab Implementierung aus der Datei eingelesen und blockweise verarbeitet. So kann der in Abbildung 3.3 dargestellte Ablauf übernommen werden. Die einzige Herausforderung stellte die Anpassung der Speicherindizes zwischen Matlab und C dar.

Abbildung 3.12 zeigt, dass die C-Simulation genau wie die Matlab Simulation ERLE-Werte von deutlich über 20 dB erreicht. Diese Umsetzung des Algorithmus wird für die Implementierung auf dem DSP übernommen.

3.3.3 Hardware Setup

Das Hardware Setup besteht vollständig aus der zur Verfügung gestellten Hardware (siehe Tabelle 2.2). Abbildung 3.13 zeigt den Schaltplan des Setups.

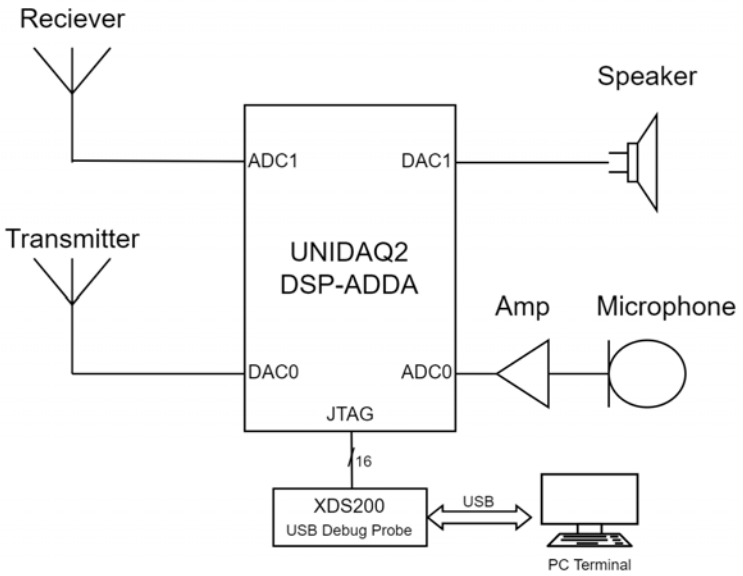

Abbildung 3.13: Schaltplan des Hardware Setups des Demonstrators

3.3.4 DSP-Board Setup und Initialisierung

Für den Setup des DSP-Boards wird auf eine bereits im Laborbetrieb verwendete und erprobte Projektvorlage zurückgegriffen, welche die Interrupt-Service-Routinen (ISR) und die Kommunikation mit dem Terminal zu Debug-Zwecken initialisiert. Die Abtastrate wird auf 16 kHz festgelegt. Nach erfolgreichem Setup werden die ISRs aktiviert und

der Algorithmus ausgeführt. Der Algorithmus wird mit den bereits in den Simulationen verwendeten Parametern initialisiert (siehe Tabelle 3.1), mit Ausnahme der Filterlänge, welche auf Grund der begrenzten Ressourcen des DSPs auf 1024 Koeffizienten reduziert werden musste.

3.3.5 Memorymanagement und Interrupt-Service-Rountinen

Der wesentliche Unterschied zwischen der C Simulation und der Implementierung auf dem DSP besteht in der Datenakquise. Während das Testsignal komplett geladen und anschließend in Blöcke zerlegt und verarbeitet werden kann, müssen die Samples auf dem DSP einzeln vom ADC eingelesen und abgespeichert werden. Erst wenn alle Samples eines Blocks eingelesen wurden, kann der Block verarbeitet und parallel ein neuer Block eingelesen werden. Anschließend müssen die Samples des Ausgangsblockes wieder serialisiert werden und mittels DAC ausgegeben werden. Dies geschieht in den Interrupt Service Routinen (ISR) von ADC und DAC, welche im Folgenden näher betrachtet werden.

Die ADC ISR speichert die Samples in dem für den neuen Block reservierten Speicherbereich. Dabei werden die Samples mittels einer Blockindex Variable gezählt und bei Erreichen des letzten Samples ein entsprechendes Flag (ADC-Flag) gesetzt. Das Flag verhindert, das weitere Samples eingelesen werden und signalisiert der Hauptroutine, dass der Block komplett eingelesen wurde. Treffen Samples ein, obwohl das Flag gesetzt ist, werden diese verworfen. Außerdem werden alle Samples eines Blockes aufaddiert, um die Entfernung des Mittelwertes zu vereinfachen. Der Ablauf der ISR des ADC ist in Abbildung 3.14 dargestellt.

Die DAC ISR serialisiert den zuletzt verarbeiteten Speicherblock und schreibt die Samples in der entsprechenden Frequenz auf den spezifizierten DAC Ausgang. Dazu wird die Blockindex Variablen der ADC ISR verwendet. Setzt die ADC ISR das Flag, dass ein Block komplett eingelesen wurde, wird der DAC Ausgang auf Null gesetzt. Um den Effekt des Echos zu verstärken, wird in der DAC Routine außerdem eine variable Verzögerung eingebaut. Abbildung 3.15 zeigt den Ablauf der DAC ISR.

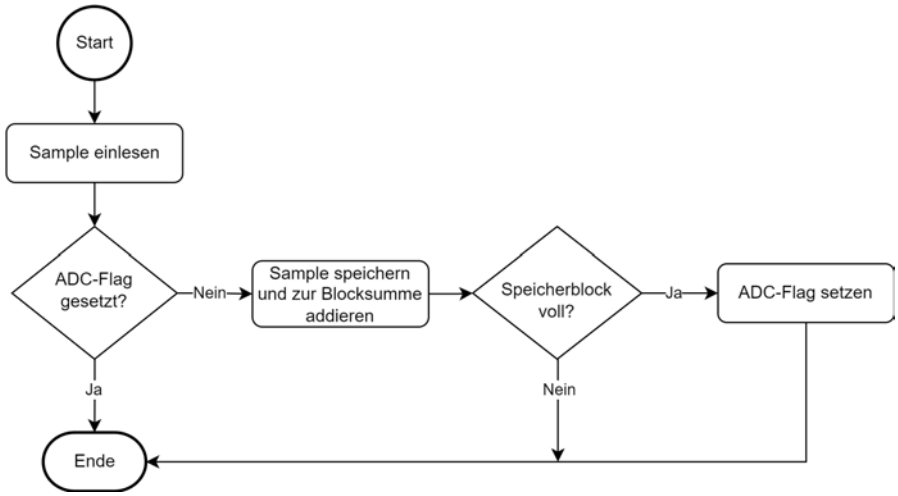

Abbildung 3.14: Ablaufdiagramm der ADC Interrupt Service Routine

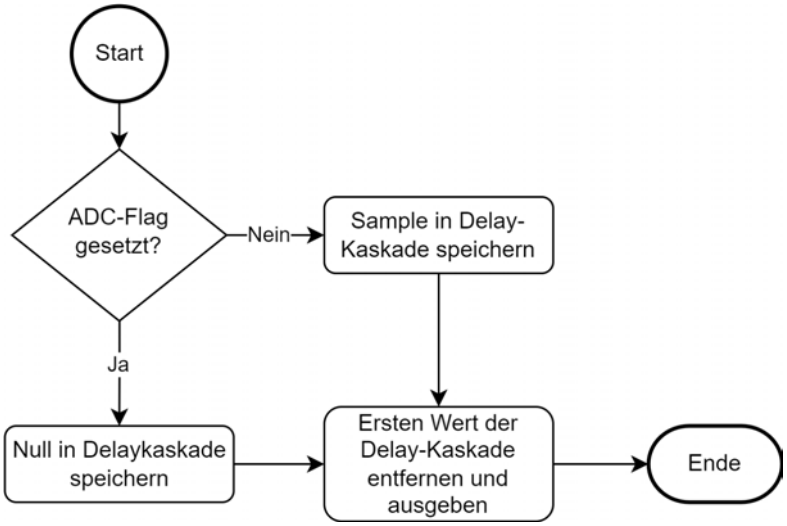

Abbildung 3.15: Ablaufdiagramm der DAC Interrupt Service Routine

Da der Algorithmus den Overlap-Save Algorithmus anwendet, wird neben dem aktuellen Block des Far Speech Signals auch der zuvor eingelesene benötigt. Um die Verarbeitung zu vereinfachen, werden außerdem Speicherplätze für die komplexen Werte reserviert. Es werden daher für das Far Speech Signal drei Speicherbereiche mit jeweils der vierfachen Blocklänge erstellt: Einer, welchen der Algorithmus für die Verarbeitung verwendet und zwei, in dem die aktuellen Samples des ADC gespeichert werden. Für das Mikrofonsignal und das Ausgangssignal müssen dagegen nur jeweils zwei Speicherbereiche mit einfacher

Blocklänge reserviert werden. Ist ein neuer Speicherblock eingelesen und der Algorithmus fertig mit der Datenverarbeitung, müssen diese Speicherblöcke (bzw. ihre Pointer) getauscht werden. Dies ist schematisch in Abbildung 3.16 dargestellt.

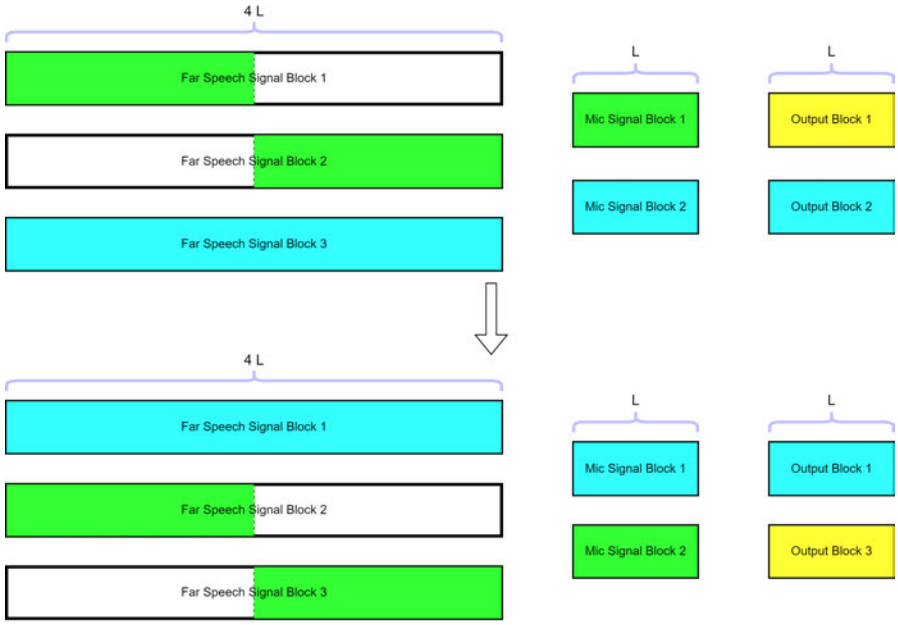

Abbildung 3.16: Schematische Darstellung des Memorymanagements. Darstellung der Speicherbereiche und ihrer Verwendung vor (oben) und nach (unten) dem Tausch nach Verarbeitung eines Datenblocks. Grün markierte Speicherbereiche werden vom ADC beschrieben, gelb markierte werden vom DAC serialisiert und blau markierte werden vom AEC-Algorithmus verarbeitet

4 Evaluation

Ziel der Evaluation ist es, den in Kapitel 3.3 beschriebenen Demonstrator in Bezug auf die in Kapitel 2.6 definierten Anforderungen zu untersuchen und zu bewerten.

4.1 Qualität

Um die Qualität der Echounterdrückung auf dem Zielsystem zu evaluieren, wird das zuvor aufgenommene Testsignal (siehe Kapitel 3.2) als Eingangssignal (Far Speech Signal) auf das Zielsystem gegeben und das Ausgangssignal (Nutzsignal) des AEC-Demonstrators aufgezeichnet. Das Testsignal wurde aufgrund des begrenzten Speicherplatzes auf $1,25$ Sekunden gekürzt. Um das ERLE-Maß bestimmen zu können, wird jede Messung zweimal durchgeführt, um sowohl das AEC-Ausgangssignal, als auch das Mikrofonsignal aufzeichnen zu können. Es wird einmal zu Beginn und einmal nach viermaligen Durchlauf des Testsignals gemessen. Der Messaufbau gleicht dem in Abbildung 3.13 dargestellten Aufbau. Die Ergebnisse sind Abbildung 4.1 zu entnehmen. Die Messergebnisse zeigen wie erwartet zu Beginn der Adaption niedrige Echodämpfungen von ca. 10 dB, welche bereits nach ca. 1 Sekunde auf 20 dB ansteigen. Nach viermaliger Wiederholung des Testsignals erreicht das adaptive Filter Echodämpfungen von bis zu 30 dB. Dass im realen Test höhere Dämpfungen erreicht werden, als in den Simulationen, ist auf den Pegel des Mikrofonssignals zurückzuführen, welcher im realen Test etwa doppelt so hoch war wie der des in den Simulationen verwendeten Testsignals. Der Mikrofonsignalpegel ist vermutlich auf Grund der erhöhten Lautstärke des Lautsprechers erhöht.

Eine Dämpfung von 30 dB ist psychoakustisch wirksam genug, um unerwünschte Signale durch den Verdeckungseffekt zu eliminieren [52]. Damit sind die Anforderungen an die Qualität des AEC Demonstrators erfüllt. Lediglich das Double Talk Szenario wurde in der Umsetzung aus Zeitgründen nicht berücksichtigt, was allerdings die Effektivität des Demonstrators nicht beeinträchtigt.

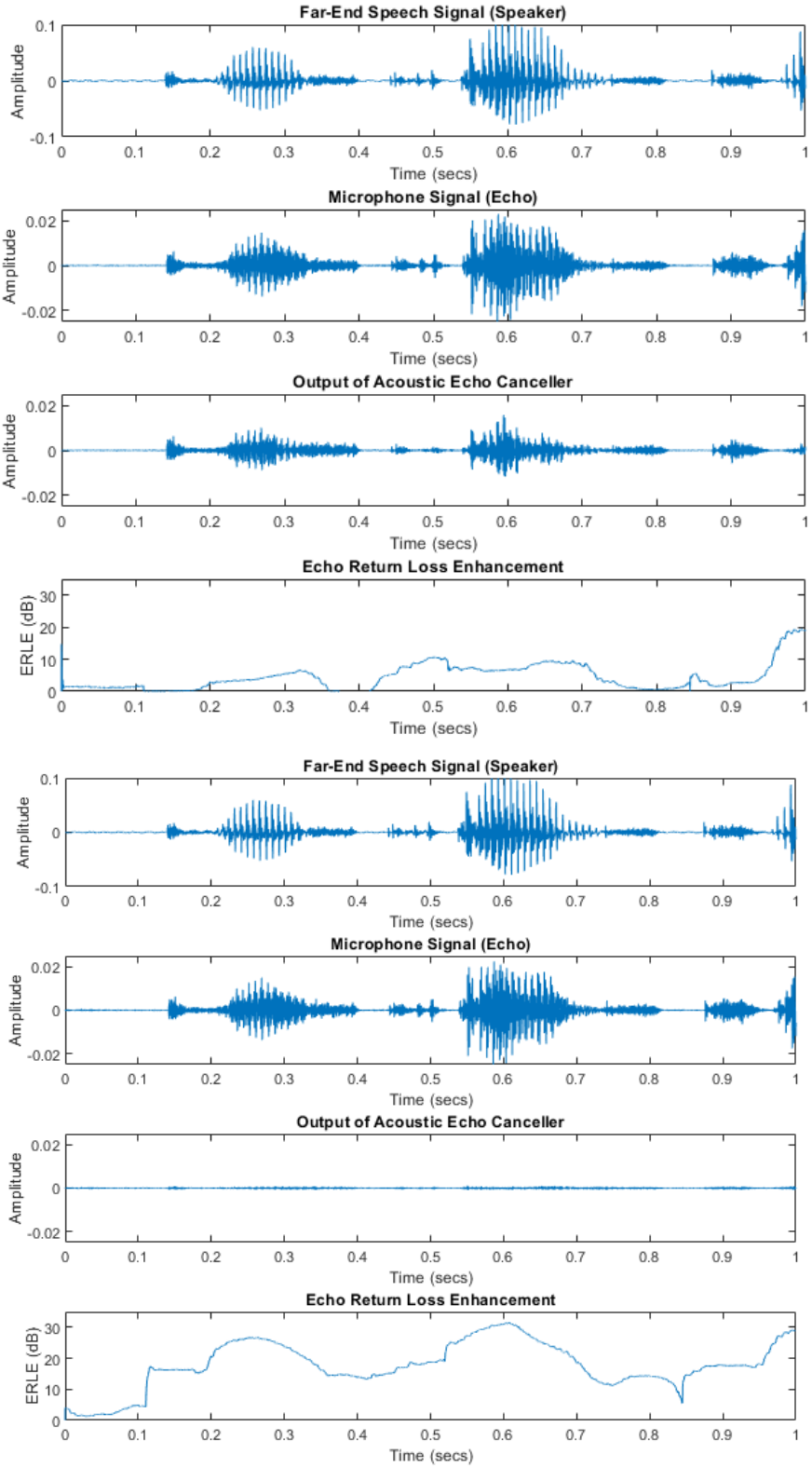

Abbildung 4.1: Echodämpfung des Demonstrators mit Testsignal zu Beginn der Adaption (oben) und nach 4 Durchläufen des Testsignals (unten)

4.2 Performance

Um die Performance des Algorithmus zu ermitteln, wird die Amplitude am DAC0 während der Rechenzeit auf Null gezogen, sodass die diese mittels Oszilloskop gemessen werden kann. Zeitgleich wird nach dem Einlesen eines Signalblocks der Ausgang am DAC1 getoggelt. Da die Performance vom Optimierungslevel des Compilers abhängt [46], wird diese Messung mit jedem der Optimierungslevel wiederholt. Höhere Optimierungslevel entsprechen dabei einer höheren Optimierung. Auch der jeweilige Speicherbedarf wird ermittelt. Alle weiteren Optimierungsparameter bleiben während der Messung konstant. Abbildung 4.2 fasst die Messergebnisse zusammen, die Messdaten sind Tabelle A.1 im Anhang zu entnehmen.

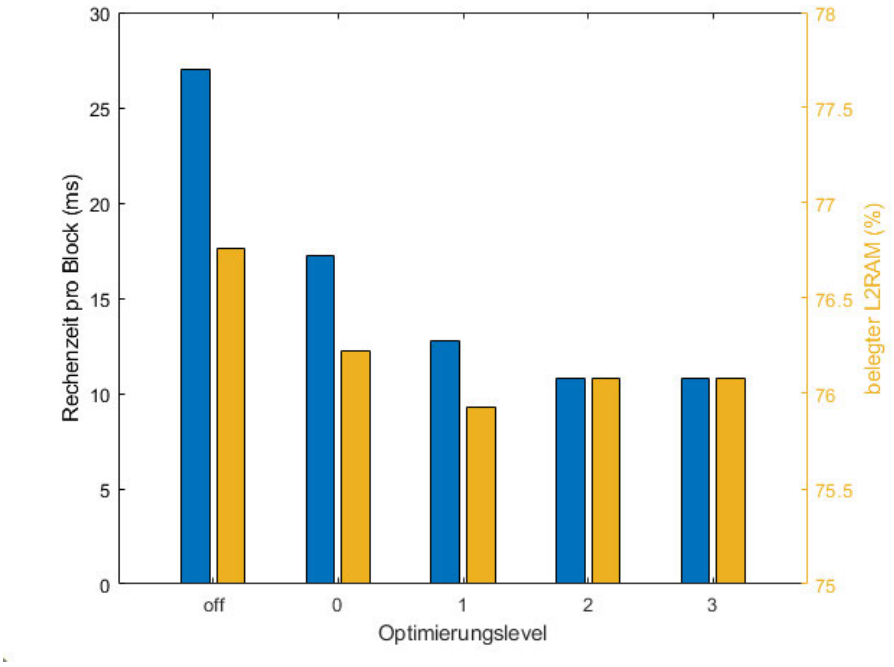

Abbildung 4.2: Performance des Algorithmus in Abhängigkeit des Optimierungslevels. Gemessen wurde der Einfluss des Compilerparameters $--opt\_level$. Die weiteren Optimierungsparameter blieben dabei konstant ($--opt\_for\_speed = 2$ und $--opt\_for\_space = 1$)

Während bei der Rechenzeit eine deutliche Verkürzung mit zunehmendem Optimierungslevel zu erkennen ist, bleibt die Speicherauslastung annähernd gleich. Ein Nebeneffekt zeigte sich bei Optimierungslevel 2 und höher, bei welchen eine Variable, welche in einer

der ISR-Routinen gelesen (aber nicht geschrieben) wurde und nicht als *volatile* deklariert war, scheinbar durch den Compiler entfernt worden war.

Die Messung (siehe Tabelle A.1) ergab außerdem, dass der Parameter $--opt\_for\_speed$, welcher in Code Composer Studio als *Speed vs. size trade-off* bezeichnet wird, weder auf die Rechengeschwindigkeit, noch auf den Speicherbedarf einen Einfluss hat. Für den Parameter $--opt\_for\_space$ war ebenfalls nur ein minimaler Einfluss messbar. Abbildung 4.3 zeigt den Einfluss des Parameters bei verschiedenen Optimierungslevels. Der maximale durch Erhöhung des Parameters eingesparte Speicherbedarf liegt bei lediglich 0.33 % (Optimierungslevel 3). Auch diese Messwerte sind im Anhang zu finden (siehe Tabelle A.2).

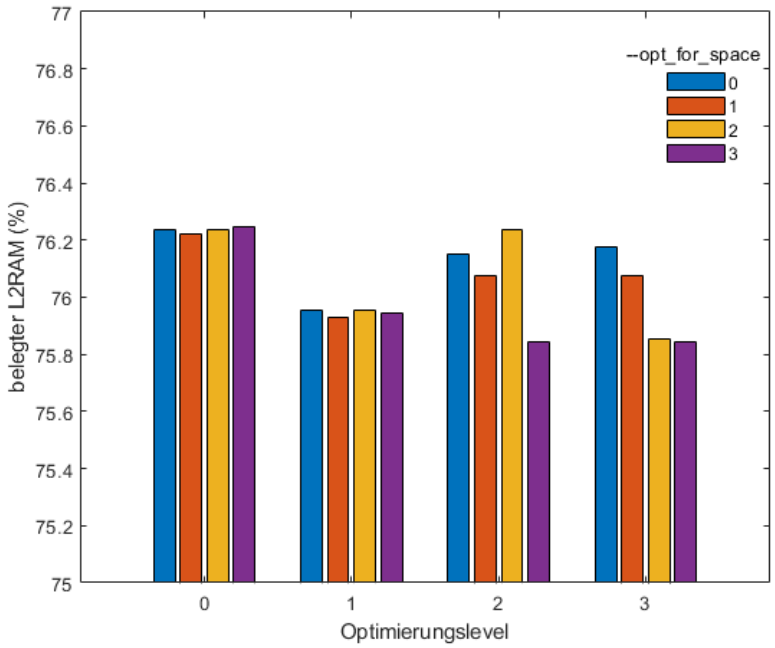

Abbildung 4.3: Speicherbedarf des Algorithmus in Abhängigkeit des Optimierungslevels $--opt\_level$ und des Parameters $--opt\_for\_space$. Die weiteren Optimierungsparameter blieben dabei konstant ($--opt\_for\_speed = 2$)

Die Rechenzeit des Algorithmus liegt mit je nach Optimierungslevel 10,75 - 27 ms deutlich unter dem Maximum von 64 ms, der Dauer eines Signalblocks. Allerdings ist der Programmspeicher des DSP mit 76 - 77 % so weit ausgelastet, dass eine weitere Erhöhung

der Filterlänge auf 2048 Koeffizienten nicht möglich ist. Dennoch ist der Demonstrator auf der Zielhardware lauffähig und erfüllt somit diese Anforderung.

4.3 Latenz

Wie bereits in Kapitel 2.6 erläutert, spielt die Latenz ebenfalls eine wichtige Rolle bei der Beurteilung des Demonstrators. Sie sollte niedrig genug liegen, dass die Kommunikation uneingeschränkt möglich ist, aber nicht so niedrig sein, dass der Echoeffekt kaum mehr wahrnehmbar ist. Da eine anpassbare künstliche Verzögerung des Signals implementiert wurde, liegt das Augenmerk der Evaluation auf der Einhaltung der oberen Grenze der Echtzeitkommunikation von 400 ms [22].

Da die Verzögerung bedingt durch die Blockverarbeitung des Algorithmus konstant 64 ms beträgt und dagegen die Verzögerungen der Funkstrecke vernachlässigbar klein sind, ist auch diese Anforderung erfüllt. Um einen deutlich wahrnehmbaren Echo-Effekt mit dem Demonstrator zu erreichen, sollte die künstliche Verzögerung auf mindestens 300 ms gesetzt werden.

4.4 Lehreinsatz

Zuletzt wird die didaktische Eignung des Demonstrators untersucht. Dazu wird auf die verwendete Hardware, Software und Programmiersprachen, sowie die angewandten Algorithmen eingegangen und geprüft, ob diese den Voraussetzungen entsprechen.

4.4.1 Hardware

Die zur Verfügung gestellte Hardware (siehe Kapitel 2.6) wurde vollständig und ohne Einsatz von zusätzlicher Technik verwendet. Der entwickelte Demonstrator ist auf der Hardware implementiert worden und darauf lauffähig (siehe Kapitel 4.2). Somit eignet sich dieser unter diesem Aspekt für die Anwendung in der Lehre und im Labor.

4.4.2 Software

Die Simulationen der beiden AEC Algorithmen wurden vollständig mittels Matlab (Version R2023b) durchgeführt, die Implementierung auf dem zur Verfügung gestellten DSP erfolgte mit Code Composer Studio (Version 12.5.0). Die zusätzliche C-Simulation wurde mit Microsoft Visual Studio 2022 (Version 17.6.5) durchgeführt. Es wurden keine zusätzliche Software oder Programmiersprachen verwendet. Die Anwendung sämtlicher verwendeter Software und Programmiersprachen kann nach den in Kapitel 2.6 erwähnten Modulhandbüchern vorausgesetzt werden. Folglich erfüllt der entwickelte Demonstrator diese Anforderungen.

4.4.3 Algorithmen

Die schließlich umgesetzte Version des Demonstrators verwendet unter Anderem die folgenden Methoden und Algorithmen der digitalen Signalverarbeitung:

- Parallelisierung und Serialisierung von Signalen, Blockverarbeitung
- Fast-Fourier-Transformation
- Schnelle Faltung und schnelle Korrelation, Overlap-Save Algorithmus
- FIR Filter
- Fensterung von Signalen
- Leistungsdichtespektrum

Sämtliche angewandten Konzepte und Methoden können nach den Modulhandbüchern ([19], [20], [21]) als bekannt vorausgesetzt werden und sind Teil der klassischen Signalverarbeitung. Die Herausforderung besteht darin, das bereits bekannte Wissen in einem neuen Konzept, dem adaptiven Filter, zu vereinen. Die Anforderung an die verwendeten Methoden gilt somit als erfüllt.

5 Fazit

In dieser Arbeit wurde die zugrundeliegende Theorie, Konzeption, Umsetzung und Evaluation eines adaptiven Filters zur akustischen Echounterdrückung beschrieben. Ziel war es, diesen in Form eines Demonstrators für den Einsatz in der Hochschullehre zu entwickeln, welcher es ermöglicht, bereits bekanntes Wissen der klassischen Signalverarbeitung zu kombinieren und praktisch anzuwenden. Um dies zu erreichen, wurden zunächst grundlegende Konzepte von Acoustic Echo Cancellation und adaptiven Filtern beschrieben. Zwei konkrete adaptive Algorithmen wurden exemplarisch im Detail erläutert (siehe Kapitel 2): Der FLMS Algorithmus, eine nach Performance optimierte Version des Least-Mean-Squares Algorithmus, macht sich die Eigenschaften der Signalverarbeitung im Frequenzbereich zu nutze, um effizient zu filtern und das Fehlersignal zu analysieren. Dabei kommt unter Anderem die FFT und der Overlap-Save Algorithmus zum Einsatz. Der Subband AEC Algorithmus spart dagegen Rechenzeit und Konvergenzzeit, indem er das Signal in mehrere Frequenzbänder aufteilt und in jedem Band jeweils einen (kürzeren) adaptiven Filter (hier NLMS) anwendet. Dabei kann das adaptive Filter durch Dezimation auf einer niedrigeren Abtastfrequenz berechnet werden. Da für die einzelnen Bänder kürzere Filterlängen ausreichen, wird außerdem die Konvergenzzeit beschleunigt. Das Ende des Kapitels 2 bildet die Definition der Anforderungen, welche die Umsetzung erfüllen soll.

Nach der Vorstellung des Systemkonzeptes in Kapitel 3.1 werden beide Algorithmen zunächst in Matlab simuliert, um sie bezüglich ihrer Komplexität, Performance und Eignung für die Implementierung auf dem DSP zu untersuchen. Um dabei möglichst aussagekräftige und realitätsnahe Ergebnisse zu erhalten, wurden mit der zur Verfügung gestellten Hardware ein Testsignal und das entsprechende Echo aufgenommen. Beide Algorithmen erreichten deutlich messbare Dämpfungen des Echos (siehe Abbildung 3.11). Allerdings wies der FLMS Algorithmus nicht nur eine bessere Echoreduktion, sondern auch eine signifikant bessere Performance auf und wurde daher für die Umsetzung auf dem DSP ausgewählt. Vor der Implementierung auf dem DSP ist der Algorithmus allerdings zunächst in C Simuliert worden, mit dem Ziel den Übergang zwischen Matlab und dem

DSP zu erleichtern. Zu diesem Zweck wurde auch die Verwendung von mex-Files untersucht, einer Möglichkeit, C/C++ Funktionen aus Matlab aufzurufen. Allerdings stellte sich die mex-API für diesen Anwendungsfall als zu komplex heraus. Schließlich wurde die Implementierung auf dem DSP beschrieben, angefangen beim konkreten Aufbau der Hardware (siehe Abbildung 3.13), über das Setup des verwendeten DSP-Boards, bis hin zur Implementierung der Interrupt-Service-Routinen.

Im Kapitel 4 wurde der Demonstrator in Bezug auf die zuvor definierten Anforderungen untersucht, welche ohne Einschränkungen erfüllt wurden. Hierbei wurde das Testsignal auf dem Zielsystem dazu verwendet, die reale Echodämpfung sowie die Rechenzeit des Algorithmus in Abhängigkeit des Compiler-Optimierungslevels zu bestimmen (siehe Abbildungen 4.1 und 4.2). Auf der Hardware wird eine Echodämpfung von über 30 dB erreicht, welche ausreicht, um das Echo durch die eigene Sprache zu überdecken [52]. Die Rechenzeit des Algorithmus beträgt je nach Optimierungslevel zwischen 10,75 und 27 ms, deutlich weniger als die für die Ein- und Ausgabe eines Signalblocks benötigten 64 ms. Dies entspricht der durch die Blockverarbeitung bedingten Latenz des Systems. Die Übertragung kann damit als Echtzeitkommunikation klassifiziert werden [23]. Um für Demonstrationszwecke ein deutliches Echo zu erzeugen, wird das Signal zusätzlich um 300 ms verzögert. Schlussendlich ergab die Auswertung der verwendeten Methoden und Konzepte der klassischen Signalverarbeitung, sowie der verwendeten Hard- und Software, dass diese bereits aus vorhergehenden Modulen bekannt seien und lediglich ihre Kombination zu einem adaptiven Filter nicht Teil der entsprechenden Modulhandbücher ist.

Der Anhang zur Arbeit (insbesondere der Quellcode) befindet sich auf CD und kann beim Erstgutachter eingesehen werden.

5.1 Ausblick

Das in dieser Arbeit entwickelte System bietet in verschiedenen Bereichen Potential zur Optimierung und Erweiterung. Auf einige Aspekte, welche sich im Verlauf der Entwicklung abgezeichnet haben, wird im Folgenden näher eingegangen.

Die beiden theoretisch erörterten und simulierten Algorithmen FLMS und Subband AEC bieten beide verschiedene Parameter, welche Performance und Qualität maßgeblich beeinflussen. Die in Kapitel 3.2 für die Simulation und auch in Kapitel 3.3 für die Umsetzung auf dem DSP eingestellten Parameter stammen aus bereits vorhandenen Bei-

spielen und sorgen bei beiden Algorithmen für eine erfolgreiche, signifikante Reduktion des Echos. Dies ist zwar für Demonstrationszwecke ausreichend, allerdings bietet sich hier die Möglichkeit der Optimierung, um die Parameter bestmöglich an die Einsatzumgebung anzupassen. Für die Optimierung des Subband AEC Algorithmus beschreiben Lee et al. [25] einige zu optimierende Kostenfunktionen, welche unter Umständen zu einer Reduktion des Rechenaufwandes führen können. Li et al. [26] stellen ebenfalls einen vielversprechenden Designansatz vor, wobei das Prototypfilter komplex moduliert wird (DFT Filterbank) und nicht mit einem Kosinussignal.

Auch eine Verbesserung der Performance des implementierten FLMS Algorithmus ist unter Umständen möglich. Der verwendete Prozessor kann pro Takt zwei 32-Bit Floating-Point Operationen, oder vier 16-Bit Fixpoint Operationen rechnen [45], eine Fixpunktimplementierung würde die Performance demnach deutlich erhöhen. In der Simulation, stellte sich allerdings heraus, dass die verwendete Implementierung der FFT dabei so viel an Genauigkeit verliert, dass keine signifikante Echoreduktion mehr möglich war. Die Implementierung der FFT müsste also entsprechend angepasst werden. Eine Verbesserung der Performance der FFT ist ebenfalls durch die Auslagerung der Berechnung der Twiddlefaktoren und des Bit-Reversing möglich, was bereits erfolgreich Simuliert wurde (siehe Kapitel 3.2.1).

Des Weiteren würde die Benutzerfreundlichkeit des Demonstrators erheblich von einer Anpassung der Hardware profitieren. Beispielsweise wäre durch Installation eines Buttons an einem der GPIO-Pins das Ein- und Ausschalten des adaptiven Filters während der Laufzeit denkbar, was eine bessere Demonstrationswirkung erzielen würde. Ferner wäre die Anpassung einiger Filterparameter während der Laufzeit denkbar, um so ihren Effekt direkt beobachtbar zu machen. Ob die Leistung des DSPs für eine Terminalverbindung zum PC via UART bei gleichzeitiger Verarbeitung der Signale ausreicht, ist dabei genauso zu untersuchen, wie die Möglichkeit, diese Funktion mittels mehrerer Buttons zu implementieren. Auch die Implementierung einer angepassten Pegelwaage (siehe Kapitel 2.5.2) wäre sinnvoll, um ein gleichzeitiges Sprechen (Double Talk) möglich zu machen, ohne dass sich der adaptive Filter dabei verstellt.

Zuletzt ist es denkbar, beide Algorithmen gemeinsam zu implementieren, in dem Sinne, dass der FLMS Algorithmus in den jeweiligen Subbändern berechnet wird. Laut Vary et al. [50] ist der FLMS Algorithmus ab einer Filterlänge von über 32 Koeffizienten dem LMS Algorithmus überlegen, was die Rechengeschwindigkeit angeht. Da in den einzelnen Subbändern jedoch meist ein kurzes Filter ausreicht und der Ressourcenverbrauch

bei längeren Filtern um ein Vielfaches erhöht wird, ist fraglich, ob dies insgesamt zu einer Verbesserung der Qualität führt. Allerdings wäre diese Kombination aus didaktischer Perspektive interessant, da so wesentlich mehr Konzepte der Signalverarbeitung zur Anwendung kommen, wobei natürlich auch die Komplexität entsprechend erhöht wird. Beispielsweise wäre eine Aufteilung denkbar, bei welcher verschiedene Gruppen von Studierenden jeweils einen der beiden Algorithmen entwickeln, sich gegenseitig vorstellen und diese anschließend kombinieren.

Literaturverzeichnis

[1] Aironi, Carlo: *SubbandAdaptiveX*. 2024. – URL https://github.com/viduzz84/SubbandAdaptiveX. – Zugriffsdatum: 2024-09-12

[2] Albu, Iuliana ; Anghel, Cristian ; Paleologu, Constantin: Adaptive filtering in acoustic echo cancellation systems — A practical overview. In: *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2017, S. 1–6

[3] Benesty, Jacob ; Huang, Yiteng (Hrsg.): *Adaptive Signal Processing*. Berlin : Springer, 2003

[4] Bismor, Dariusz ; Czyz, Krzysztof ; Ogonowski, Zbigniew: Review and Comparison of Variable Step-Size LMS Algorithms. In: *The International Journal of Acoustics and Vibration* 21 (2016), S. 24–39

[5] Brady, P. T. ; Helder, G. K.: Echo suppressor design in telephone communications. In: *The Bell System Technical Journal* 42 (1963), Nr. 6, S. 2893–2917

[6] Changxuding: *Subband Kalman AEC*. 2024. – URL https://github.com/changxuding/Subband_Kalman_AEC. – Zugriffsdatum: 2024-09-12

[7] Craney, Edward P.: *Adaptive IIR filtering for acoustic echo cancellation on a mobile handset*, University of Southampton, Dissertation, 2004

[8] Cutler, Ross ; Saabas, Ando ; Panarmaa, Tanel ; Loide, Markus ; Sootla, Sten ; Purin, Marju ; Gamper, Hannes ; Braun, Sebastian ; Aichner, Robert ; Srinivasan, Sriram: INTERSPEECH 2021 Acoustic Echo Cancellation Challenge. In: *Interspeech 2021*, 2021

[9] Cutler, Ross ; Saabas, Ando ; Parnamaa, Tanel ; Purin, Marju ; Gamper, Hannes ; Braun, Sebastian ; Sørensen, Karsten ; Aichner, Robert: ICASSP 2022 Acoustic Echo Cancellation Challenge. In: *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, S. 9107–9111

[10] D.SignT: *UniDAQ2.DSP-ADDA Prozessor Board.* 2018. – URL https://www.dsignt.de/files/dsignt/media/datasheets/dsunidaq2_dsp-adda.pdf. – Zugriffsdatum: 10.09.2024

[11] Enzner, Gerald ; Buchner, Herbert ; Favrot, Alexis ; Kuech, Fabian: Chapter 30 - Acoustic Echo Control. In: Trussell, Joel (Hrsg.) ; Srivastava, Anuj (Hrsg.) ; Roy-Chowdhury, Amit K. (Hrsg.) ; Srivastava, Ankur (Hrsg.) ; Naylor, Patrick A. (Hrsg.) ; Chellappa, Rama (Hrsg.) ; Theodoridis, Sergios (Hrsg.): *Academic Press Library in Signal Processing: Volume 4* Bd. 4. Elsevier, 2014, S. 807–877

[12] Enzner, Gerald ; Vary, Peter: Frequency-domain adaptive Kalman filter for acoustic echo control in hands-free telephones. In: *Signal Processing* 86 (2006), Nr. 6, S. 1140–1156

[13] Farhang-Boroujeny, Behrouz: *Adaptive Filters: Theory and Applications.* 1st ed. Wiley, 2013 (New York Academy of Sciences Series)

[14] Gilloire, A. ; Vetterli, M.: Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation. In: *IEEE Transactions on Signal Processing* 40 (1992), Nr. 8, S. 1862–1875

[15] Haubner, Thomas ; Brendel, Andreas ; Kellermann, Walter: End-to-End Deep Learning-Based Adaptation Control for Linear Acoustic Echo Cancellation. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32 (2024), S. 227–238

[16] Haubner, Thomas ; Halimeh, Mhd. M. ; Brendel, Andreas ; Kellermann, Walter: A Synergistic Kalman- and Deep Postfiltering Approach to Acoustic Echo Cancellation. In: *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, S. 990–994

[17] Haykin, Simon: *Adaptive Filter Theory.* 5. Pearson Education Limited, 2014

[18] Heckmann, Martin ; Vogel, Julia ; Kroschel, Kristian: Frequency Selective Step-Size Control For Acoustic Echo Cancellation. In: *European Signal Processing Conference* 2015 (2000)

[19] Hochschule für Angewandte Wissenschaften Hamburg: *Modulhandbuch des BA-Studiengangs Elektrotechnik und Informationstechnik 2024.* 2024. – URL https://www.haw-hamburg.de/fileadmin/zentrale_PDF/TI/Modulhandb%C3%BCcher/Informations-_und_Elektrotechnik/MDH-

BA-IuE-EuI/MDH-BA-IuE-EuI-2024_final.pdf. – Zugriffsdatum: 2024-09-12

[20] Hochschule für Angewandte Wissenschaften Hamburg: *Modulhandbuch des MA-Studiengangs Informations- und Kommunikationstechnik 2021.* 2024. – URL https://www.haw-hamburg.de/fileadmin/zentrale_PDF/TI/Modulhandb%C3%BCcher/Informations-_und_Elektrotechnik/MDH-MA-IuE-IuK/MDH-MA-IuE-IuK-2021.pdf. – Zugriffsdatum: 2024-09-12

[21] Hochschule für Angewandte Wissenschaften Hamburg: *Modulhandbuch des MA-Studiengangs Mikroelektronische Systeme 2018.* 2024. – URL https://www.haw-hamburg.de/fileadmin/zentrale_PDF/TI/Modulhandb%C3%BCcher/Informations-_und_Elektrotechnik/MDH-MA-IuE-MeS/MDH-MA-IuE-MeS-2018.pdf. – Zugriffsdatum: 2024-09-12

[22] Hänsler, Eberhard ; Schmidt, Gerhard: *Topics in Acoustic Echo and Noise Control.* 1. Springer Berlin, Heidelberg, 2006

[23] (ITU), International Telecommunication U.: *ITU-T Recommendation G.114: One-way transmission time.* 2003

[24] Kim, Youngjoo ; Bang, Hyochoong: *Introduction to Kalman Filter and Its Applications.* In: *Introduction and Implementations of the Kalman Filter*, IntechOpen, 2019

[25] Lee, Kong-Aik ; Gan, Woon-Seng ; Kuo, Sen-Maw: *Subband adaptive filtering: Theory and implementation.* 1. Wiley, 2009. – ISBN 978-0-470-51694-2

[26] Li, Qin ; Chen, Wei-Ge ; He, Chao ; Malvar, Henrique S.: Design of oversampled DFT modulated filter banks optimized for acoustic echo cancellation. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, S. 197–200

[27] Lippuner, Dani ; Moschytz, George S.: The Kalman filter in the context of adaptive filter theory. In: *International Journal of Circuit Theory and Applications* 32 (2004), Nr. 4, S. 223–253

[28] Loizou, Philipos C.: *Speech Quality Assessment.* S. 623–654. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011

[29] Mader, Andreas ; Puder, Henning ; Schmidt, Gerhard U.: Step-size control for acoustic echo cancellation filters – an overview. In: *Signal Processing* 80 (2000), Nr. 9, S. 1697–1719

[30] Mboup, M. ; Bonnet, M.: IIR filtering for acoustic echo cancellation. In: *Conference Record of the Twenty-Fifth Asilomar Conference on Signals, & Systems Computers*, 1991, S. 203–206

[31] Moschytz, George S. ; Hofbauer, Markus: *Adaptive Filter*. 1. Springer Berlin, Heidelberg, 2000

[32] Myllylä, Ville: Residual echo filter for enhanced acoustic echo control. In: *Signal Processing* 86 (2006), Nr. 6, S. 1193–1205. – Applied Speech and Audio Processing

[33] Naylor, P.A. ; Tanrikulu, O. ; Constantinides, A.G.: Subband adaptive filtering for acoustic echo control using allpass polyphase IIR filterbanks. In: *IEEE Transactions on Speech and Audio Processing* 6 (1998), Nr. 2, S. 143–155

[34] Ozeki, Kazuhiko ; Umeda, T.: An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. In: *Electronics and Communications in Japan Part I-communications* 67 (1984), S. 19–27

[35] Paleologu, C. ; Benesty, J. ; Ciochina, S.: Study of the General Kalman Filter for Echo Cancellation. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21 (2013), August, Nr. 8, S. 1539–1549

[36] Peng, Renhua ; Cheng, Linjuan ; Zheng, Chengshi ; Li, Xiaodong: ICASSP 2021 Acoustic Echo Cancellation Challenge: Integrated Adaptive Echo Cancellation with Time Alignment and Deep Learning-Based Residual Echo Plus Noise Suppression. In: *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, S. 146–150

[37] Press, W. H. ; Teukolsky, S. A. ; Vetterling, W. T. ; Flannery, B. P.: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007

[38] Rix, A.W. ; Beerends, J.G. ; Hollier, M.P. ; Hekstra, A.P.: Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)* Bd. 2, 2001, S. 749–752

[39] Sakhnov, Kirill ; Verteletskaya, Ekaterina ; Simak, Boris: *Perceptual Echo Control and Delay Estimation*. In: Garcia, Lino (Hrsg.): *Adaptive Filtering Applications*, IntechOpen, 2011

[40] Shynk, J.J.: Frequency-domain and multirate adaptive filtering. In: *IEEE Signal Processing Magazine* 9 (1992), Nr. 1, S. 14–37

[41] Sondhi, M. M. ; Presti, A. J.: A self-adaptive echo canceller. In: *The Bell System Technical Journal* 45 (1966), Nr. 10, S. 1851–1854

[42] Sondhi, Man M.: The History of Echo Cancellation. In: *IEEE Signal Processing Magazine* 95 (2006)

[43] Sridhar, Kusha ; Cutler, Ross ; Saabas, Ando ; Parnamaa, Tanel ; Loide, Markus ; Gamper, Hannes ; Braun, Sebastian ; Aichner, Robert ; Srinivasan, Sriram: ICASSP 2021 Acoustic Echo Cancellation Challenge: Datasets, Testing Framework, and Results. (2020)

[44] Statistisches Bundesamt: *Erwerbstätige, die von zu Hause aus arbeiten*. 2024. – URL https://www.destatis.de/DE/Themen/Arbeit/Arbeitsmarkt/Qualitaet-Arbeit/Dimension-3/home-office.html. – Zugriffsdatum: 31.05.2024

[45] Texas Instruments Inc.: *TMS320C6747 7 Fixed- and Floating-Point Digital Signal Processor*. 2008. – URL https://www.ti.com/lit/ds/symlink/tms320c6747.pdf. – Zugriffsdatum: 2024-09-19

[46] Texas Instruments Inc.: *TMS320C28x Optimizing C/C++ Compiler*. 2024. – URL http://downloads.ti.com/docs/esd/SPRU514/. – Zugriffsdatum: 2024-09-19

[47] The Mathworks, Inc.: *Acoustic Echo Cancellation (AEC)*. 2024. – URL https://www.mathworks.com/help/audio/ug/acoustic-echo-cancellation-aec.html. – Zugriffsdatum: 2024-09-12

[48] The Mathworks, Inc.: *Call C/C++ MEX Functions from MATLAB*. 2024. – URL https://www.mathworks.com/help/matlab/call-mex-file-functions.html?s_tid=CRUX_lftnav. – Zugriffsdatum: 2024-09-12

[49] The Mathworks, Inc.: *Frequency Domain Adaptive Filter*. 2024. – URL https://www.mathworks.com/help/dsp/ref/dsp.

frequencydomainadaptivefilter-system-object.html. – Zugriffsdatum: 2024-09-12

[50] Vary, Peter: *Digital speech transmission: Enhancement, coding, and error concealment*. Wiley, 2005

[51] Vary, Peter ; Heute, Ulrich ; Hess, Wolfgang: *Digitale Sprachsignalverarbeitung: Mit 30 Tabellen*. Teubner, 1998

[52] Weinzierl, Stefan (Hrsg.): *Handbuch der Audiotechnik*. Springer Vieweg Berlin, Heidelberg, 2020

[53] Williamson, Geoffrey A.: *Adaptive IIR Filters*. In: Madisetti, Vijay K. (Hrsg.) ; Williams, Douglas B. (Hrsg.): *The Digital Signal Processing Handbook*, CRC Press LLC, 1999

[54] Xu, Ewan: *PyAEC*. 2024. – URL https://github.com/ewan-xu/pyaec. – Zugriffsdatum: 2024-09-12

[55] Xu, Shiyun ; He, Changjun ; Yan, Bosong ; Wang, Mingjiang: A Multi-Stage Acoustic Echo Cancellation Model Based on Adaptive Filters and Deep Neural Networks. In: *Electronics* 12 (2023), Nr. 15

[56] Yang, Dong ; Jiang, Fei ; Wu, Wei ; Fang, Xuefei ; Cao, Muyong: Low-Complexity Acoustic Echo Cancellation with Neural Kalman Filtering. In: *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023

[57] Zhang, Hao ; Kandadai, Srivatsan ; Rao, Harsha ; Kim, Minje ; Pruthi, Tarun ; Kristjansson, Trausti: Deep Adaptive Aec: Hybrid of Deep Learning and Adaptive Acoustic Echo Cancellation. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, S. 756–760

[58] Zhao, Haiquan ; Chen, Badong: *Efficient Nonlinear Adaptive Filters: Design, Analysis and Applications*. 1. Springer International Publishing and Imprint Springer, 2023

A Anhang

A.1 Messdaten

Tabelle A.1: Messdaten der Performance und Speicherauslastung des Algorithmus in Abhängigkeit der Optimierungsparameter −−opt_level und −−opt_for_speed. Der Parameter −−opt_for_space blieb während der Messung konstant auf dem Wert 1

−−opt_level	−−opt_for_speed	Rechenzeit (ms)	belegter L2RAM (%)
off	–	27	76.76
0	0	17.2	76.22
0	1	17.2	76.22
0	2	17.2	76.22
0	3	17.2	76.22
0	4	17.2	76.22
0	5	17.2	76.22
1	0	12.8	75.93
1	1	12.8	75.93
1	2	12.8	75.93
1	3	12.8	75.93
1	4	12.8	75.93
1	5	12.8	75.93
2	0	10.75	76.08
2	1	10.75	76.08
2	2	10.75	76.08
2	3	10.8	76.08
2	4	10.8	76.08
2	5	10.8	76.08
3	0	10.8	76.08
3	1	10.75	76.08
3	2	10.75	76.08
3	3	10.8	76.08
3	4	10.75	76.08
3	5	10.75	76.08

Tabelle A.2: Messdaten der Performance und Speicherauslastung des Algorithmus in Abhängigkeit der Optimierungsparameter −−opt_level und −−opt_for_space. Der Parameter −−opt_for_speed blieb während der Messung konstant auf dem Wert 2

−−opt_level	−−opt_for_space	belegter L2RAM (%)
off	–	27
0	0	76.236
0	1	76.224
0	2	76.236
0	3	76.248
1	0	75.956
1	1	75.931
1	2	75.955
1	3	75.943
2	0	76.151
2	1	76.077
2	2	75.856
2	3	75.844
3	0	76.175
3	1	76.077
3	2	75.856
3	3	75.844

Erklärung zur selbstständigen Bearbeitung

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

Ort Datum Unterschrift im Original