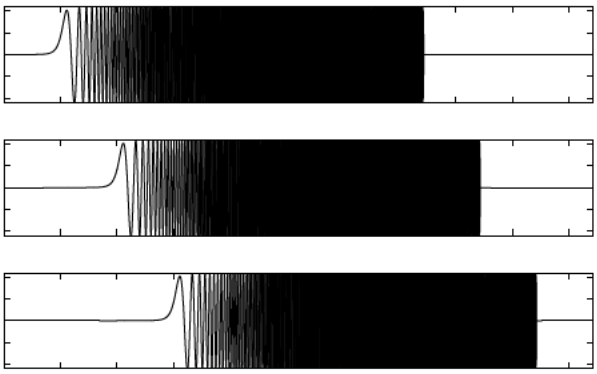
Technische Universität Berlin
Institut für Sprache und Kommunikation
Fachgebiet Audiokommunikation

Magisterarbeit

Entwicklung einer Software zur zeitoptimierten, akustischen Fehlerdetektion für vielkanalige Lautsprechersysteme



Christian Dietz

[redacted]

Betreuer:
Prof. Dr. Stefan Weinzierl
Prof. Dr. Anselm Görtz

Die selbstständige Anfertigung versichere ich an Eides statt.

Christian Dietz Berlin, den 25. November 2010

Inhaltsverzeichnis

1. Einleitung

Lautsprechersysteme, wie sie etwa zur Synthese von Schallfeldern zum Einsatz kommen, verfügen über eine Vielzahl von Lautsprecherkanälen. So ist das Wellenfeldsynthese-System der Technischen Universität zu Berlin, das im Jahr 2007 in einem Hörsaal im Rahmen dessen Renovierung eingerichtet wurde, mit 2704 Lautsprechern ausgestattet, die auf 832 Kanäle verteilt sind. Die Überprüfung des Funktionszustandes der einzelnen Kanäle eines Systems dieser Größenordnung ist per aures schwer zu bewerkstelligen und mit einem hohen Zeitaufwand verbunden.

In der vorliegenden Arbeit wird eine Methode zur automatisierten Überprüfung solcher Lautsprecheranlagen entwickelt und eine entsprechende Messsoftware vorgelegt. Es soll ein Messsystem konzipiert und realisiert werden, das sich problemlos in vorhandene Lautsprechersysteme integrieren lässt, auftretende Defekte an Lautsprecherkanälen zuverlässig identifiziert und zeitoptimiert arbeitet. Die daraus folgenden Konsequenzen und sich ableitenden Fragen, werden in den einzelnen Kapiteln behandelt. In Kapitel 2 wird ein geeignetes Messverfahren gesucht, welches die Basis einer solchen Software bilden kann, und geprüft, ob Möglichkeiten existieren, die sich daraus ergebenden Messzeiten zu reduzieren. Welche Defekte von der Software detektiert werden sollen, wird in Kapitel 3 anhand der Lautsprecher der WFS-Anlage der TU Berlin erörtert. Wie diese Defekte simuliert werden können, damit sie während der Entwicklung des Detektionsalgorithmus und der Evaluation des Messsystems verfügbar sind, wird in Kapitel 4 verhandelt. Die Integration des Messsystems in eine bestehende Wiedergabeanlage, ist Gegenstand von Kapitel 5. Außerdem wird die gewählte Programmierumgebung in diesem Kapitel vorgestellt. In Kapitel 6 wird der Aufbau und das Bedienkonzept der Messsoftware beschrieben. Den Kern der Arbeit bildet der Entwurf eines Algorithmus zur Identifizierung von Defekten und Defektkombinationen von Lautsprecherkanälen, der in Kapitel 8 Raum findet. Abschließend wird in Kapitel 9 dessen Zuverlässigkeit mittels systematischer Simulation fehlerhafter Betriebszustände eruiert.

2. Grundlagen

Da das zu entwickelnde Messsystem auch für schon installierte Lautsprecheranlagen ohne großen Aufwand nutzbar sein soll, ist ein Messverfahren erforderlich, das unabhängig von der Art der Signaldistribution ist und keine baumaßlichen Veränderungen an vorhandenen Komponenten notwendig macht. Aus diesem Grund kommt statt einem Verfahren, das auf Messungen der elektrischen Impedanz der Lautsprecher basiert, eines in Frage, welches mittels akustischer Messung Informationen über den Betriebszustand sammelt. Um den Messvorgang zeitlich zu optimieren, liegt es nahe, die in [Majdak u. a., 2007] vorgestellte *Multiple Exponential Sweep Method*, kurz MESM, zu nutzen. Die Voraussetzung hierfür ist ein FFT-basiertes Messverfahren mit Sweeps als Anregungssignalen, weshalb dieses die Grundlage des in dieser Arbeit entwickelten Messsystems bildet. In dem Folgenden wird dieses Verfahren und die MESM näher erläutert.

FFT-basierte Messverfahren bedienen sich der Theorie der Signale und Systeme[1]. Die zu messende Komponente wird als ideales System gesehen. Solche Systeme haben die Eigenheit, linear und zeitinvariant zu sein, weshalb sie *Linear Time-Invariant Systems*, kurz LTI-Systeme, genannt werden.

2.1. LTI-Systeme

Durch die Annahme, dass es sich bei einem Teil der Messstrecke um ein LTI-System handelt, vereinfachen sich Zusammenhänge und es stehen viele Methoden zur Analyse bereit. Ein System, wie in Abbildung 2.1 skizziert, transformiert ein Eingangssignal $x(n)$ in ein Ausgangssignal $y(n)$. Liegt ein LTI-System vor, wird es durch die Eigenschaften Linearität und Zeitinvarianz charakterisiert. Nach [Oppenheim und Schafer, 1999] ist ein System *linear*, wenn das Superpositionsprinzip gilt, also eine beliebige Linearkombination von Eingangssignalen zu einer ihr proportionalen Summe

[1] siehe Oppenheim und Willsky [1996]

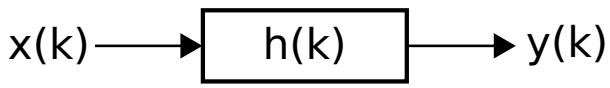


Abbildung 2.1.: Ein System

von Ausgangssignalen führt. *Zeitinvariant* ist ein System, wenn sich dessen Eigenschaften nicht zeitabhängig ändern. Die Reaktion auf ein Eingangssignal ist zu jedem Zeitpunkt gleich.

Ein LTI-System wird durch seine Impulsantwort $h(n)$, kurz IR für *Impulse Response*, vollständig beschrieben. Sie ist die Verknüpfung von Eingangssignal $x(n)$ und Ausgangssignal $y(n)$ der Länge und wird im zeitdiskreten digitalen Bereich durch die Faltungssumme

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = x(n) * h(n) \tag{2.1}$$

beschrieben [Kammeyer und Kroschel, 2006, Seite 13]. Diese wird durch das Symbol „$*$“ kenntlich gemacht. Wird ein LTI-System mit einem Dirac-Impuls angeregt, liegt am Ausgang direkt die Impulsantwort vor. Die Implementierung eines LTI-Systems fordert zudem, dass *Kausalität* gewährleistet ist. Dies bedeutet, dass die Reaktion eines Systems nicht vor der eigentlichen Anregung reagieren darf. Gleichung 2.1 wird so nach

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) = x(n) * h(n) \tag{2.2}$$

überführt [Feiten und Röbel, 1996, Seite 21]. Hat $x(n)$ die Länge N und $h(n)$ die Länge M beträgt die Länge der Faltungssumme $M + N - 1$.

Die Eigenschaft der Linearität ermöglicht es, das mathematische Werkzeug Fourier-Transformation auf LTI-Systeme anzuwenden. Die Fourier-Transformation ist in der Lage, ein Signal im Frequenzbereich darzustellen. Auf der digitalen Ebene wird sie *Discrete Fourier Transform*, kurz DFT, genannt. Für diskrete Signale ist das Spektrum immer kontinuierlich und periodisch mit 2π. Nach [Kammeyer und Kroschel, 2006, Seite 222] lässt sich das Spektrum $X(n)$ der *zeitdiskreten* Folge $x(k)$ der Länge N aus der Transformationsgleichung

$$X(n) = DFT\{x(k)\} = \sum_{k=0}^{N-1} x(k)e^{jk\frac{2\pi}{N}n} \tag{2.3}$$

ermitteln. Die Frequenzauflösung ist abhängig von der Länge N sowie der Abtastfrequenz f_A und beträgt $\frac{f_A}{N}$. Die IDFT, kurz für *Inverse Discrete Fourier Transform*, gestattet eine Rücktransformation in den Zeitbreich.

$$x(k) = IDFT\{X(n)\} = \frac{1}{N}\sum_{k=0}^{N-1} X(n)e^{-jk\frac{2\pi}{N}n} \tag{2.4}$$

Die beiden Gleichungen 2.3 und 2.4 bilden ein Transformationspaar.

Durch Einsetzen von Gleichung 2.3 in Gleichung 2.2 folgt, dass die Faltung im Zeitbereich einer Multiplikation im Frequenzbereich entspricht.

$$Y(e^{j\Omega}) = X(e^{j\Omega}) \cdot H(e^{j\Omega}) \tag{2.5}$$

$\Omega = \frac{\omega}{f_A}$ ist die auf die Abtastfrequenz f_A genormte Kreisfrequenz ω. Die Fouriertransformierte $H(e^{j\Omega})$ von $h[n]$ heißt Übertragungsfunktion oder *Transfer Function*, kurz TF. Ihr Betrag bildet den Betragsfrequenzgang $|H(e^{j\Omega})|$.

Mit der *Fast Fourier Transform*, kurz FFT, steht ein schneller Algorithmus für die Berechnung von Fourier-Transformationen zur Verfügung. Für die Berechnung muss eine Signalfolge die Länge einer Zweierpotenz besitzen. Dieses Werkzeug kann nun die mit hohem Rechenaufwand verbundene Faltungsoperation durch eine einfache Multiplikation der transformierten Signale und anschließender Rücktransformation ersetzen. Dies wird *schnelle* Faltung genannt. Bei der Implementierung einer solchen Faltung muss darauf geachtet werden, dass die beiden miteinander zu faltenden Signale mindestens die Länge $M + N - 1$ besitzen und diese eine Zweierpotenz ist. Dies kann durch das Anhängen von Nullen, das sogenannte *Zero-Padding*, erreicht werden.

2.2. FFT-basierte Messverfahren

Ziel eines Messsystems ist die Akquise der Impulsantwort eines Prüflings, mit deren Hilfe zum Beispiel Aussagen über dessen Spektrum, Phasengang, Dynamikumfang und nicht lineares Verhalten getroffen werden können. Neben der in dieser Arbeit vorgestellten Messmethode, existieren noch weitere Messtechniken, welche etwa in [Müller und Massarani, 2001] ausführlich beschrieben werden.

Bei der Annahme, dass es sich bei der Übertragungsstrecke, wie sie Abbildung 2.2 veranschaulicht, um ein LTI-System handelt, werden zeitinvariante Faktoren wie et-

wa Temperatur- oder Luftfeuchtigkeitsschwankungen und das nichtlineare Verhalten einzelner Komponenten zunächst nicht berücksichtigt. Dies bietet die Möglichkeit, die Werkzeuge der digitalen Signalverarbeitung für die Messung heranzuziehen. Die

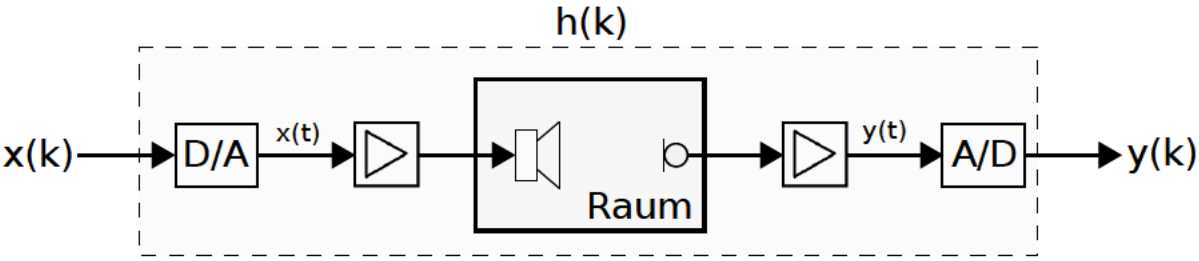


Abbildung 2.2.: Eine Messstrecke

Umstellung der Gleichung 2.5 zu

$$H(e^{j\Omega}) = \frac{Y(e^{j\Omega})}{X(e^{j\Omega})} \tag{2.6}$$

ist die fundamentale Rechenoperation sogenannter FFT-basierter Messsysteme und wird als Entfaltung bezeichnet. Werden mittels FFT das Anregungssignal $x(k)$ und das Ausgangssignal $y(k)$ in den Frequenzbereich transformiert, entspricht ihr *komplexer* Quotient der Übertragungsfunktion $H(e^{j\Omega})$. Die Impulsantwort $h(k)$ kann nun leicht durch eine anschließende IFFT gewonnen werden. Diese beinhaltet aber neben den Übertragungseigenschaften des Messobjekts Lautsprecher auch jene der bei einem softwarebasierten Messsystem notwendigen Wandlerkomponenten, des Mikrofons inklusive des Vorverstärkers, des Raumes und der Endstufe zum Betrieb des Lautsprechers. Die Einflüsse der Verstärker- und Wandlereinheiten können durch eine elektrische Referenzmessung getilgt werden, indem, wie in Abbildung 2.3 dargestellt, der Ausgang des Leistungsverstärkers über einen Pegelabschwächer direkt mit dem Eingang des Mikrofonvorverstärkers verbunden wird. In der so gemessenen Übertragungsfunktion $X^*(e^{j\Omega})$ sind die Übertragungseigenschaften der digitalen und analogen Signalverarbeitungsstufen mit Ausnahme des Lautsprechers und des Mikrofons enthalten. Handelt es sich bei dem Anregungssignal um einen deterministischen Stimulus, welcher sich bei jedem Messzyklus identisch verhält, muss die Referenzmessung nur einmal zu Beginn der Messungen getätigt und gespeichert werden. Damit nicht die Übertragungsfunktion des Mikrofons in das Messergebnis einbezogen wird, muss dessen Übertragungsfehler H_{Mic} durch einen separaten Ver-

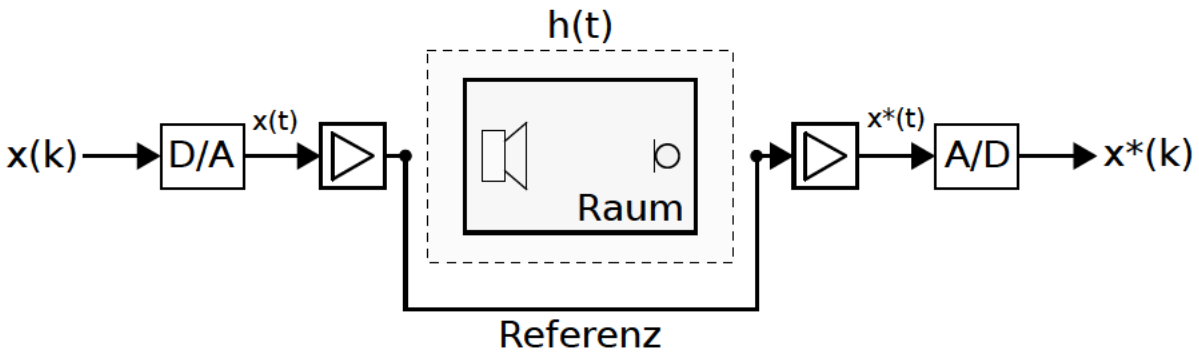


Abbildung 2.3.: Elektrische Referenzmessung

gleich mit einem Referenzmessmikrofon ermittelt werden. Ist dies geschehen, kann die Übertragungsfunktion zwischen zwei Punkten innerhalb eines Raumes aus

$$H(e^{j\Omega}) = \frac{Y(e^{j\Omega})}{X^*(e^{j\Omega})} \cdot \frac{1}{H_{Mic}(e^{j\Omega})} \tag{2.7}$$

berechnet werden. Sie enthält die Übertragungsfunktion des Raumes und die des Lautsprechers.

2.2.1. Raumimpulsantwort

Die zur Raumübertragungsfunktion gehörende Impulsantwort wird Raumimpulsantwort bzw. *Room Impulse Response*, kurz RIR, genannt. Abbildung 2.4[2] zeigt, dass sich eine solche Raumimpulsantwort in Direktsignal, erste Reflexionen und diffusen Nachhall aufteilen lässt. Das Direktsignal entspricht dem Teil des ausgesendeten Schalls, der ohne Umweg das Mikrofon erreicht. Es folgen die ersten Reflexionen an Raumbegrenzungen und Gegenständen innerhalb des Raumes. Der zeitliche Versatz zwischen ihnen und dem Direktschall wird durch die Raumeigenschaften, die gegenseitige Lage von Mikrofon und Lautsprecher sowie ihrer Positionen innerhalb des Raumes bestimmt. Mit der Zeit verdichten sie sich zunehmend, da sie wiederum an den verschiedenen Flächen reflektiert werden. Die Überlagerung all dieser Reflexionen mündet in einem statistisch weitgehend gleichverteilten Signalanteil, dem diffusen Nachhall. Der Schalldruck des Nachhalls nimmt mit der Zeit logarithmisch ab, was durch Dissipation der Luft und die schallabsorbierende Wirkung der einzelnen Flächen zu erklären ist.

[2] aus [Zölzer, 2005, Seite 198]

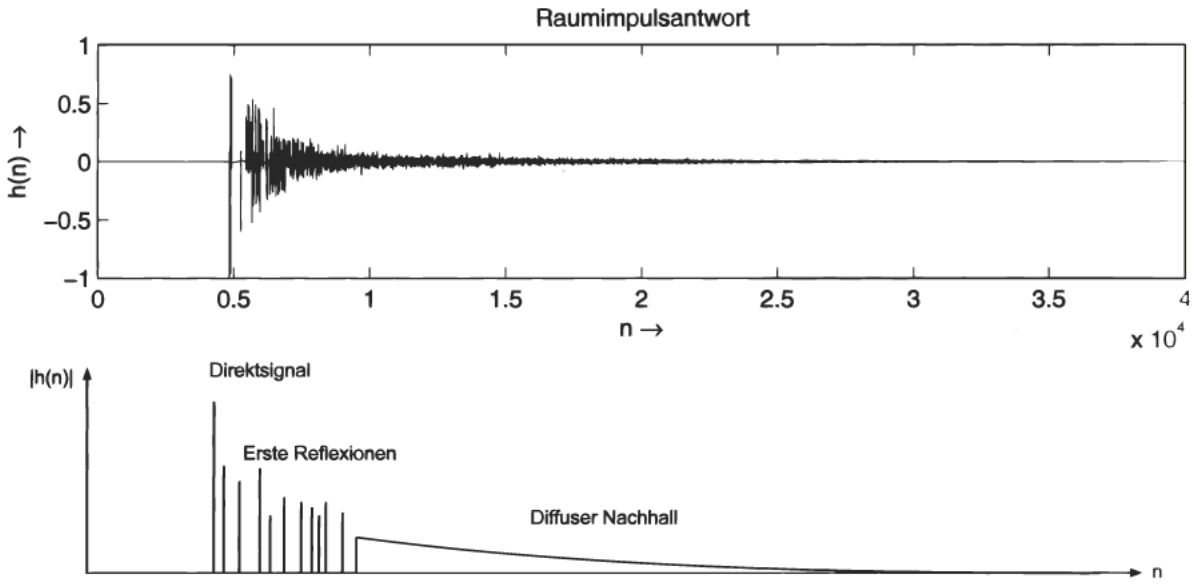


Abbildung 2.4.: Raumimpulsantwort

2.2.2. Sweeps als Anregungssignale

Welches Anregungssignal benutzt wird, hat maßgeblichen Einfluss auf die Qualität einer FFT-basierten Messung. Den Standard zur Akquise von Raumimpulsantworten stellen gegenwärtig Sweeps dar. Da sie auch für die MESM von elementarer Bedeutung sind, werden sie hier näher beschrieben. Einen Überblick über weitere Stimuli, wie etwa Dirac-Impulse und Rauschfolgen, bietet [Müller und Massarani, 2001].

Synthese von Sweeps

Ein Sweep ist ein sinusförmiges Signal, welches alle Frequenzen eines definierten Bereichs in einer bestimmten Zeit sukzessive durchläuft. Das Spektrum eines Sweeps wird durch die Veränderung der Frequenz in Abhängigkeit von der Zeit determiniert. Im Falle von linearen Sweeps erhöht sich die Frequenz mit der Dauer linear; das Resultat ist ein weißes Spektrum. Exponentielle Sweeps hingegen verfügen über ein rosa Spektrum, ihr Energiegehalt ist also pro Oktave konstant.

Sweeps können entweder im Zeitbereich oder wesentlich flexibler im Frequenzbereich durch Vorgabe eines beliebigen Betragsspektrums und einer frei wählbaren zeitlichen Hüllkurve erzeugt werden (siehe [Müller und Massarani, 2001]). Die Implementierung einer Sweepsynthese wird in [Giese, 2009, Seite 120-124] ausführlich beschrieben. Es ist zu beachten, dass dem Sweep Stille einer gewissen Dauer ange-

fügt werden muss, damit alle Frequenzanteile im Nachhall der Raumimpulsantwort vorhanden sind. Dieses in [Müller und Massarani, 2001] als Gap bezeichnete Zeitintervall wird durch die maximale Differenz der Nachhallzeiten aller vorkommenden Frequenzen und der von diesen abhängigen verbleibenden Dauer des Sweeps bestimmt. Hierbei ist die Nachhallzeit als die Zeit definiert, die der Abklingvorgang benötigt, um im Rauschteppich zu verschwinden. Bei raumakustischen Messungen nimmt die Nachhallzeit in der Regel zu höheren Frequenzen hin ab, weshalb es sinnvoll ist, Sweeps von tiefen nach hohen Frequenzen laufen zu lassen.

Signal-Rausch-Abstand

Der *Signal-To-Noise Ratio*, kurz SNR, ist ein Maß für das Verhältnis von Nutz- und Störsignalen einer Messung. Der Dynamikumfang einer Raumimpulsantwort wird durch eine Vielzahl an Faktoren beschränkt, wie dem Eigenrauschen des Prüflings, dem thermischen Rauschen der Verstärker, dem Quantisierungsrauschen der Wandler sowie den Hintergrundgeräuschen des Raumes. Wie groß der SNR sein sollte, ist gebunden an die jeweilige Anwendung. Hohe Ansprüche müssen an binaurale Impulsantworten gestellt werden, weshalb in [Lindau, 2006, Seite 113] ein SNR von 98 dB gefordert wird. Für die Frequenzgangsmessung von Lautsprechern hingegen werden in [Weinzierl, 2008, Seite 1156] 65 dB für ausreichend befunden.

Eigenschaften von Sweeps

Bei raumakustischen Messungen leistet die Wahl des Anregungssignal einen entscheidenden Beitrag zum maximal erreichbaren SNR. Zum einen bestimmt es die Höhe der Austeuerungsgrenze des Systems, zum anderen die Robustheit Störgeräuschen gegenüber.

Von den bekannten Anregungssignalen liefern Sweeps bei raumakustischen Messungen die besten Ergebnisse bezogen auf den SNR. Sie verfügen über einen niedrigen Crest-Faktor[3], sind gegenüber Zeitvarianzen wenig anfällig und besitzen schon nach einmaligem Abspielen ein glattes Spektrum. Sie können durch die flexible Synthese im Frequenzbereich an die jeweilige Messsituation angepasst werden. So gewährleistet die Anpassung des Frequenzgangs an das frequenzselektive Verhalten des Raumes einen konstanten SNR über den gesamten Frequenzbereich einer Messung. Die konstante zeitliche Hüllkurve garantiert eine maximale Leistung über die-

[3] Der Crest-Faktor beschreibt das Verhältnis von Scheitelwert zu Effektivwert einer Wechselgröße

sen Bereich. Sie kann aber auch an die Leistungsverträglichkeit eines Mehr-Wege-Systems adaptiert werden, wodurch jeder Weg mit der optimalen Leistung gespeist wird. Ein auch für die MESM entscheidendes Merkmal ist der Umgang mit Nichtlinearitäten, auch Klirrprodukte genannt, bedingt durch zu hohe Aussteuerung. Ist die Dauer des Sweeps größer als die der Raumimpulsantwort, werden diese harmonischen Verzerrungsprodukte durch den Entfaltungsvorgang vor die lineare Raumimpulsantwort projiziert. In Abbildung 2.5 ist dies dargestellt. Die Klirrprodukte kön-

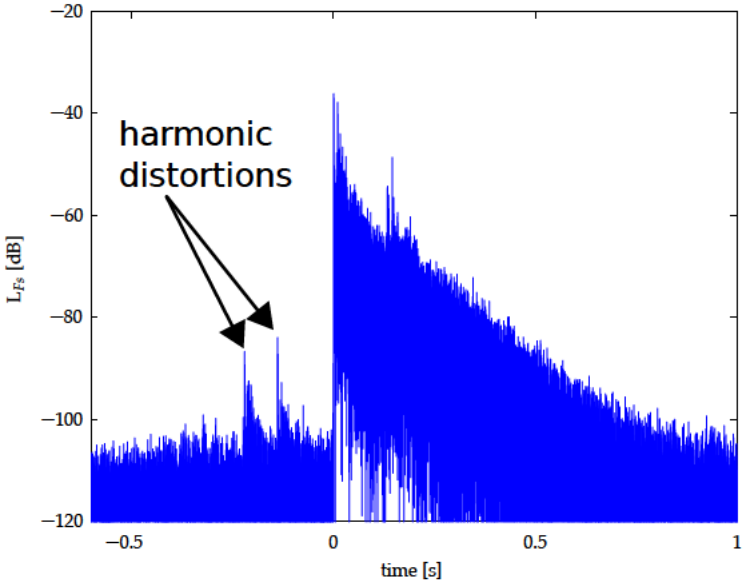


Abbildung 2.5.: Entfaltungsprodukt einer Messung mit exponentiellen Sweep

nen leicht ausgefenstert und die lineare Impulsantwort gewonnen werden. Diese Tatsache ermöglicht es, die Messkomponenten an ihrer Belastungsgrenze zu betreiben. Beschränkt ist die Höhe der Austeuerung dadurch, dass zu hohe Pegel breitbandige Geräusche, zum Beispiel bedingt durch lose Teile am Gehäuse oder im Raum, verursachen können. Diese bringen wiederum eine Verschlechterung des SNRs mit sich. Neben der Möglichkeit durch Anheben der Amplitude auf das Anregungssignal mehr Energie zu verteilen, um den SNR zu erhöhen, ist dies auch durch Mitteln mehrerer Messzyklen und/oder Verlängerung des Sweeps realisierbar. Hierbei geht allerdings der Vorteil der Unempfindlichkeit gegenüber Zeitvarianzen verloren.

2.3. MESM - Multiple Exponential Sweep Method

In [Majdak u. a., 2007] wurde die *multiple exponential sweep method*, kurz MESM, vorgestellt, mit deren Hilfe die Messzeit zur Akquise von binauralen Impulsantworten, kurz BRIRs, bzw. *Head-Related Transfer Functions*, kurz HRTFs, für mehrere Positionen reduziert werden kann. Eine HRTF ist eine zweikanalige Übertragungsfunktion einer Strecke zwischen einem Lautsprecher und zwei Mikrofonen, welche sich jeweils in einem der beiden Gehörgänge einer Person oder eines Kunstkopfes befinden. HRTFs sind von großer Bedeutung in der Binauraltechnik, da sie die für die Lokalisation eines Hörereignisses wichtigen Informationen des Außenohres, des Kopfes und des Rumpfes beinhalten. Für die dynamische Auralisation, die mögliche Kopfbewegungen für alle Quellpositionen einbezieht, sind eine Vielzahl von HRTF-Messungen notwendig. In [Lindau, 2006, Seite 159-162] wird zum Beispiel eine Messreihe von 13741 Einzelmessungen mit einem Kunstkopf beschrieben, die 33 Stunden beanspruchte. Hier ist sicherlich eine Verkürzung der Messzeit wünschenswert. Werden die BRIRs einer Person gemessen, kann es durch unvermeidliche Kopfbewegungen zu Messfehlern kommen, die durch die Reduzierung der Messdauer minimiert werden könnten.

Die MESM fußt auf der Annahme, dass es sich bei der Messkette um ein schwach nichtlineares System handelt. Sie bedient sich exponentieller Sweeps als Anregungssignale und kombiniert die Algorithmen *Interleaving* und *Overlapping*. Das Ergebnis der Entfaltung eines leicht nichtlinearen Systems, gemessen mit einem exponentiellen Sweep, ist in Abbildung 2.6[4] dargestellt. Die Verzerrungsprodukte sind als *Harmonic Impulse Responses*, kurz HIRs, zu identifizieren. Ihre Nummerierung steigt von rechts nach links. Die erste HIR entspricht der linearen Impulsantwort, kurz LIR, welche es durch Fensterung zu extrahieren gilt. Die Längen L_k der einzelnen Impulsantworten werden durch das Grundrauschen festgelegt. Der Abstand zwischen der k-ten HIR und der LIR trägt die Bezeichnung τ_k und kann nach [Farina, 2000] durch

$$\tau_k = T \cdot \frac{\ln k}{\ln(\frac{\omega_2}{\omega_1})} \tag{2.8}$$

[4] aus [Majdak u. a., 2007]

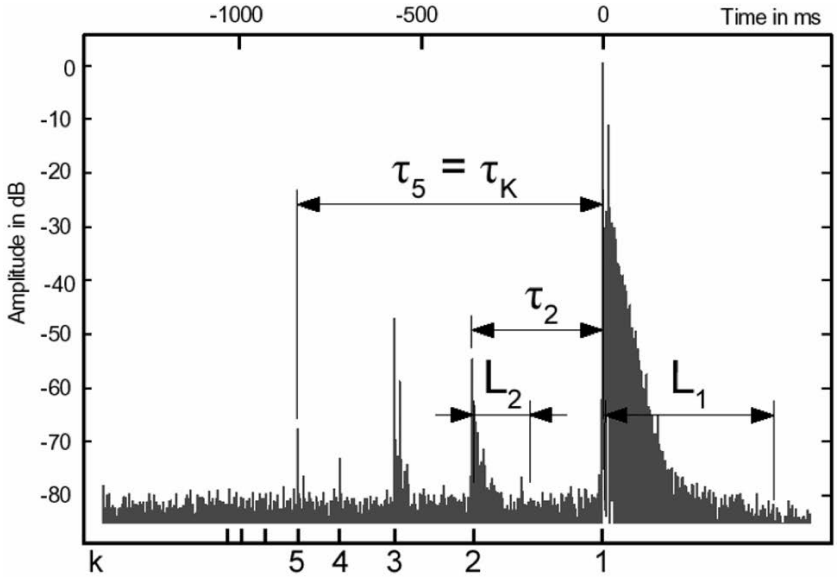


Abbildung 2.6.: Entfaltungsergebnis eines mit einem exponentiellen Sweep angeregten leicht nichtlinearen Systems

eindeutig bestimmt werden. T ist hierbei die Länge des Sweeps, ω_1 dessen Anfangs- und ω_2 dessen Endfrequenz. Die Zeit T_{ES}, die eine Messung von N Systemen beansprucht, wird von [Majdak u. a., 2007] mit

$$T_{ES} = (T + L_1)N \tag{2.9}$$

angegeben.

2.3.1. Interleaving

Dieses Verfahren bedient sich der Tatsache, dass die Verlängerung des Sweeps eine Vergrößerung der Abstände τ_k um denselben Faktor zur Folge hat. Der Zwischenraum τ_2 wird nun so weit gestreckt, dass eine bestimmte Anzahl η von LIRs in ihm Platz finden kann. Die Länge T' des benötigten Sweeps ist nach [Majdak u. a., 2007] durch

$$T' = [(\eta - 1)L_1 + L_2]\frac{\ln(\frac{\omega_2}{\omega_1})}{\ln 2} \tag{2.10}$$

gegeben. Die Messungen können nun quasi parallel stattfinden, indem die einzelnen Systeme durch Sweeps mit einer zeitlichen Verzögerung Δt_{INT} von

$$\Delta t_{INT} = (i - 1)L_1 \,, \qquad 0 < i \leq \eta \tag{2.11}$$

angeregt werden. Abbildung 2.7[5] zeigt das Entfaltungsergebnis einer Messung von vier Systemen. Die LIRs, zu erkennen an den höheren Peaks, sind zeitlich eindeutig

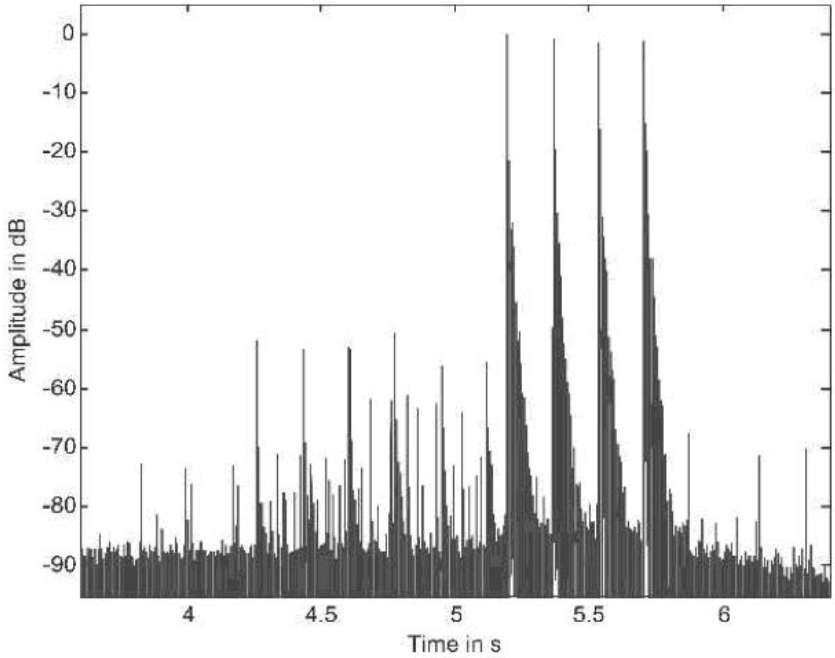


Abbildung 2.7.: Entfaltungsergebnis einer Messung mit Interleaving-Verfahren

getrennt von den HIRs, die nicht mehr den einzelnen Systemen zugeordnet werden können. Die Messdauer T_{INT} einer Messung von N Systemen, auf die in Gruppen von jeweils η Systemen das Interleaving-Verfahren angewandt wird, kann durch

$$T_{INT} = \frac{N}{\eta} T' + NL_1 \tag{2.12}$$

bestimmt werden. Der Vergleich von Gleichung 2.9 und 2.12 macht deutlich, dass nur dann eine Verbesserung der Messzeit erreicht werden kann, wenn das Verhältnis von T' und η kleiner als T ist.

2.3.2. Overlapping

Bei der Overlapping-Methode werden ebenso zueinander zeitlich verzögerte Sweeps verwendet. Die Verzögerung Δt_{OV}, gegeben durch

$$\Delta t_{OV} = \frac{\ln K}{\ln(\frac{\omega_2}{\omega_1})} T + L_1 \,, \tag{2.13}$$

[5] aus [Majdak u. a., 2007]

richtet sich aber hier nach dem größtmöglichen Abstand zwischen der LIR und der höchsten noch nicht im Rauschteppich verschwundenen HIR K. Gleichung

$$\Delta t_i = (i-1) \cdot \Delta t_{OV} \,, \qquad 0 < i \leq N \tag{2.14}$$

liefert die Anregungszeitpunkte der einzelnen Systeme. Das Ergebnis der Entfaltung einer Messung von vier Systemen mittels Overlapping-Verfahren ist in Abbildung 2.8[6] zu sehen. Die vier Systeme sind nicht verschachtelt, vielmehr kommt es zu einer Aneinanderreihung der einzelnen Entfaltungsergebnisse. Der zeitliche Aufwand T_{OV} ei-

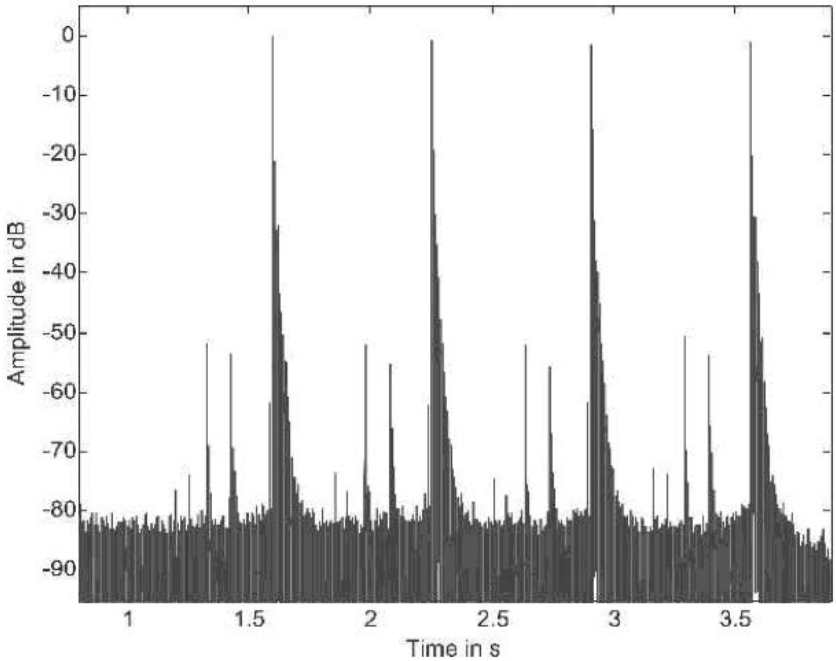


Abbildung 2.8.: Entfaltungsergebnis einer Messung mit Overlapping-Verfahren

ner Messung von N Systemen ist durch

$$T_{OV} = T + (N-1)(\tau_k + L_1) \tag{2.15}$$

bestimmt.

2.3.3. Kombination von Interleaving und Overlapping

Unter MESM wird das Zusammenspiel der beschriebenen Algorithmen verstanden. Sollen die IRs von N Systemen ermittelt werden, gilt es zunächst, Gruppen von je-

[6] aus [Majdak u. a., 2007]

weils η Systemen zu bilden, auf die Interleaving angewandt wird. Diese $\frac{N}{\eta}$ Gruppen werden nun durch Overlapping miteinander verbunden. Die Verzögerungszeiten t_i sind durch

$$t_i = L_1(i-1) + \left\lfloor \frac{i-1}{\eta} \right\rfloor \eta_k \,, \qquad 0 < i \leq N \tag{2.16}$$

gegeben. $\lfloor x \rfloor$ steht für die nächstkleinere ganze Zahl von x. Ein Entfaltungsergebnis von vier Systemen, die paarweise gruppiert wurden, zeigt Abbildung 2.9[7]. Die

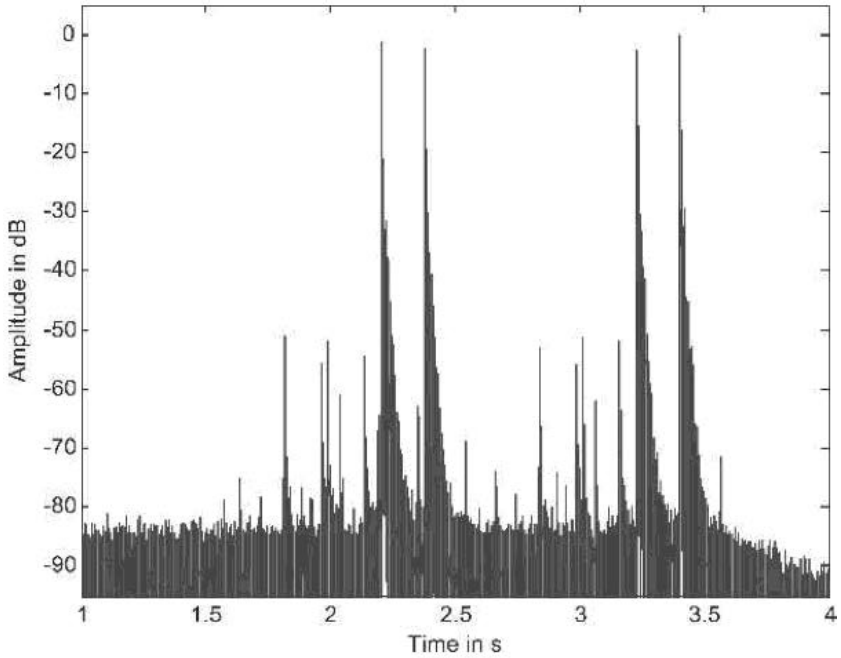


Abbildung 2.9.: Entfaltungsergebnis einer Messung mit MESM

Gesamtdauer T_{MESM} einer Messung mit MESM liefert

$$T_{MESM} = T' + \tau'_k \left(\frac{N}{\eta} - 1 \right) + NL_1 \quad . \tag{2.17}$$

Ob die Messzeit optimiert werden kann, hängt von dem durch c deklarierten Frequenzbereich des Sweeps, der höchsten detektierbaren Klirrkomponente K sowie den Nachhallzeiten L_1 und L_2 ab. Bevor eine Messung mit der MESM-Methode durchgeführt werden kann, sind die Werte dieser Parameter im Vorfeld messtechnisch zu bestimmen.

[7] aus [Majdak u. a., 2007]

2.3.4. MESM mit beliebig gefärbten Sweeps

In [Giese, 2009] wird eine Methode vorgestellt, die es möglich macht, die MESM mit arbiträr gefärbten Sweeps durchzuführen. Durch die spektrale Anpassung der Sweeps an die Umgebungsgeräusche kann der SNR einer Messung weiter erhöht werden. Eine Software wurde implementiert, welche die Overlapping-Methode für gefärbte Sweeps adaptiert. Solche Sweeps bedingen sinusartige Artefakte in der IR, weswegen das Maskierungskriterium der MESM diesbezüglich ergänzt wurde.

3. Lautsprechermodule

Die für die Wellenfeldsyntheseanlage des Raumes H0104 entwickelten Lautsprechermodule mussten diversen Anforderungen genügen. Neben den Voraussetzungen für die Wellenfeldsynthese spielten auch die Größe des Raumes und dessen verschiedene Nutzungsformen eine Rolle.

3.1. Anforderungen

Die örtlich diskrete Schallfeldsynthese bedingt orts- und frequenzabhängige Artefakte, die neben dem Einfallwinkel der virtuellen Quelle im hohen Maße vom Lautsprecherabstand abhängig sind. Diese Artefakte werden als *spatial aliasing* bezeichnet. In [Spors und Rabenstein, 2006] wird für Lautsprecherzeilen in Kombination mit ebenen Wellen die Gleichung

$$f_{pw} \leq \frac{c}{\Delta x(1 + |\cos \alpha_{pw}|)} \tag{3.1}$$

angegeben, mit der es möglich ist, die höchste noch artefaktfrei synthetisierbare Frequenz f_{pw} für den Lautsprecherabstand Δx zu bestimmen. c entspricht der Schallgeschwindigkeit und α_{pw} dem Einfallswinkel einer ebenen Welle (90° für zur Lautsprecherzeile orthogonale Einfallswinkel). Die Lautsprecher müssen also einen möglichst geringen Abstand zueinander aufweisen.

Dies bedeutet für große Räume, wie etwa den H0104, der einen Umfang von 86 m aufweist, dass eine Vielzahl von Lautsprecherkanälen notwendig ist. Das bleibt nicht ohne Konsequenzen für die Lautsprecherentwicklung: So soll die Signaldistribution platzsparend und übersichtlich geschehen sowie die Lautsprecherkonstruktion möglichst kostengünstig sein, wobei eine hohe Wiedergabequalität selbstverständlich gewährleistet sein muss.

Wie in [Behrens u. a., 2007] beschrieben, waren bei der Gestaltung der Raumakustik Kompromisse notwendig, wird der Raum doch nicht nur für die Wellenfeldsynthese, sondern auch als Hör-, Konferenz- und Konzertsaal genutzt. Für ein optimales

Ergebnis der Wellenfeldsynthese müsste der Raum reflexionsarm sein, für Vorlesungen aber über eine sprachoptimierte Nachhallzeit verfügen. Desweiteren konnten die Lautsprecher aus architektonischen Gründen nicht in einer konstanten Höhe aufgehängt werden. Das Abstrahlverhalten der Lautsprecher soll diesen Kompromissen bei der Akustik des Raumes entgegenwirken.

3.2. Aufbau

Die Lautsprecher wurden als achtkanalige Module realisiert [siehe Makarski u. a., 2008]. Abbildung 3.1 zeigt schematisch die Vorderseite eines Lautsprechermoduls. Werden mehrere solcher Module aneinandergereiht, ist die Anordnung der Kanäle

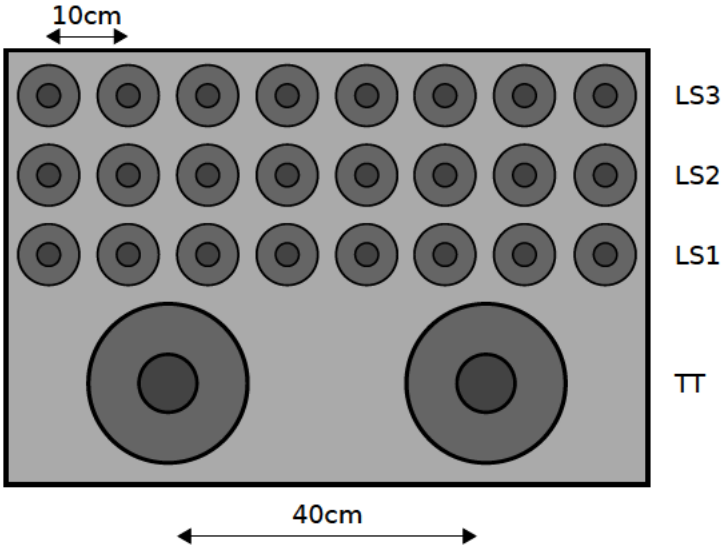


Abbildung 3.1.: Frontalansicht eines Lautsprechermoduls

äquidistant. Die Kanäle wurden als 2-Wege-Systeme konzipiert. Ein Kanal besteht aus drei Breitbandlautsprechern und teilt sich mit drei weiteren Kanälen einen Tieftöner. Durch die Verwendung der platzsparenden Breitbandlautsprecher konnte ein Abstand von 10 cm erreicht werden. Nach Gleichung 3.1 tritt so *spatial aliasing* bei einer ebenen Welle mit einem Einfallwinkel von 90° ab einer Frequenz von 3,4 kHz auf. So war es möglich, den Raum mit 832 Kanälen auszustatten.

Die Signalzuführung erfolgt für jedes Modul über eine ADAT[1]-Leitung. Der Signalfluss ist in Abbildung 3.2 für einen Kanal dargestellt. Über ein ADAT-Interface wer-

[1] wird zum Beispiel in [Weinzierl, 2008, Seite 1004] beschrieben

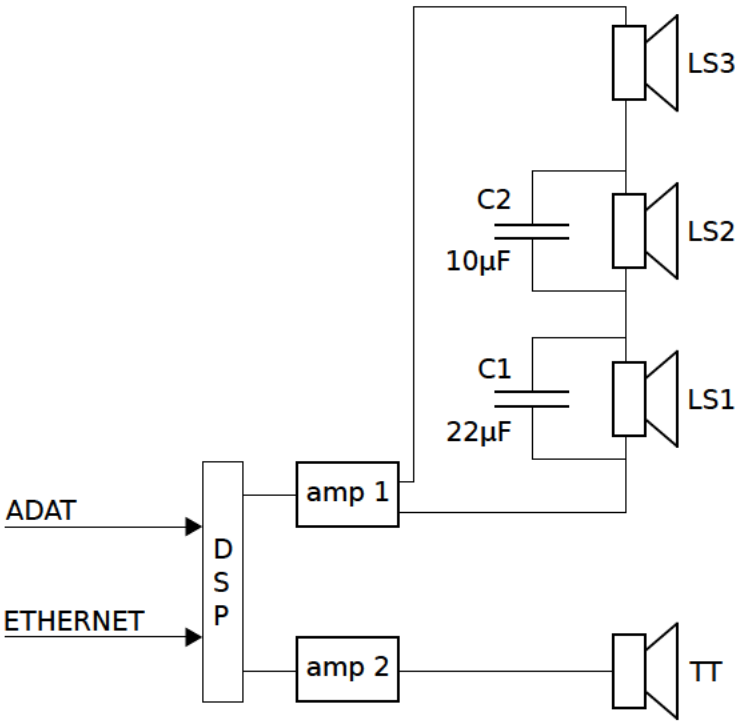


Abbildung 3.2.: Signalflussdiagramm eines Kanals

den die Audiosignale der DSP[2]-Einheit „HD2“[3] der Firma „Four Audio“ zugeführt. Diese erfüllt unterschiedliche Aufgaben und behandelt die acht Kanäle getrennt voneinander. Die Verstärkung der einzelnen Kanäle wird hier eingestellt. Vier Limiter pro Kanal verhindern Verzerrungen aufgrund von Übersteuerung der Endstufen und schützen die folgenden Komponenten vor Überlastung und Beschädigung durch zu hohe Pegel. IIR-Filter und FIR-Filter (siehe Kapitel 4.2) sind integriert. Letztere ermöglichen eine linearphasige Entzerrung oberhalb von 300 Hz und damit zum Beispiel eine kanalweise Freifeldentzerrung auf einen weitestgehend linearen Phasen- und Frequenzgang und/oder eine Raumentzerrung für die jeweilige Einbausituation. Die Überwachung aller DSP-Einheiten, Software-Updates sowie die Einstellungen der Parameter der beschriebenen Signalverarbeitung werden mittels einer bidirektionalen Verbindung mit einem zentralen Steuerrechner über ein lokales Netzwerk erreicht. Zu diesem Zweck verfügt jedes Modul über eine standardisierte Ethernet[4]-Schnittstelle. Die Endstufen für die Breitbandlautsprecher und Tieftöner besitzen 40 Watt bzw. 100 Watt. Die Trennfrequenz liegt etwa bei 200 Hz. Die Zeilenanordnung

[2] *digital signal processor*
[3] http://www.fouraudio.com/de/produkte/hd2.html
[4] siehe zum Beispiel in [Weinzierl, 2008, Seite 1022]

der drei Breitbandlautsprecher in Kombination mit einem einfachen passiven Netzwerk gewährleistet eine enge vertikale Schallbündelung ab einer Frequenz von 1 kHz, wodurch die Reflexionen der Schallwellen an der Decke und dem Boden minimiert werden.

3.3. Mögliche Defekte der Lautsprechermodule

Die Defekte eines Lautsprecherkanals, die neben dem Aussetzen kompletter Kanäle oder Module von dem Messsystem erkannt werden sollen, beruhen auf dem Ausfall einzelner Komponenten des Schaltkreises hinter den Verstärkern. Dieser ist in Abbildung 3.3 dargestellt. Er besteht aus einem Tieftöner TT und einer Serienschaltung von

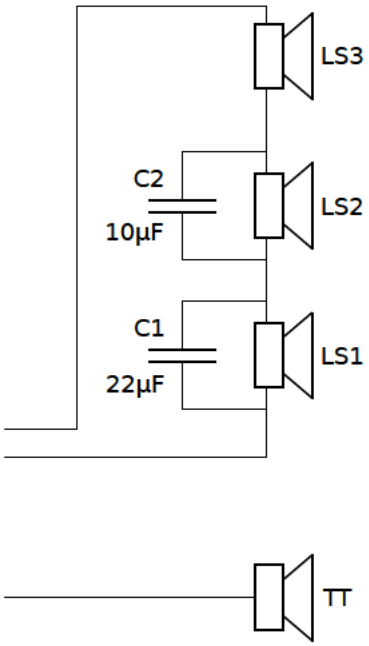


Abbildung 3.3.: Systematisches Schaltbild eines Lautsprecherkanals

drei Breitbandlautsprechern $LS1$, $LS1$ und $LS3$. Zu $LS1$ und $LS2$ sind die Kondensatoren $C1$ und $C2$ parallel geschaltet. Durch das Parallelschalten eines Kondensators wird dem Lautsprecher Tiefpassverhalten aufgeprägt. Bei $LS1$ beginnt dies bei circa 5,5 kHz und bei $LS2$ bei circa 10 kHz. Nur $LS3$ arbeitet ohne Tiefpassfilter bis zu den höchsten Frequenzen.

Wie sich der Ausfall einzelner oder mehrerer Bauteile im Frequenzgang eines Kanals widerspiegelt, wurde anhand von Messungen im „Studio 2“ des Fachbereichs *Audiokommunikation* bestimmt. Hier wurde eine 128 kanalige WFS-Anlage mit bau-

gleichen Lautsprechermodulen eingerichtet [siehe Goltz, 2010]. Als Messsystem kam ein für diese Arbeit entwickeltes System für die Messungen von einzelnen IRs zum Einsatz, außerdem ein RME Fireface 400[5] Audio-Interface mit integriertem Mikrofonvorverstärker und ein Behringer ECM8000[6] Messmikrofon. Die Abtastfrequenz f_s wurde auf 48 kHz gesetzt. Das Mikrofon wurde 2 m vor einem der installierten Module mittig auf Höhe des mittleren Breitbandlautsprechers aufgestellt. Alle nun folgenden Messungen fanden jeweils dreimal statt und wurden dann gemittelt. Zunächst erfolgte die Akquise des Frequenzganges eines funktionstüchtigen Kanals. Hiernach wurden alle möglichen Defekte am selben Kanal, wie zum Beispiel der Ausfall eines Lautsprechertreibers, manuell impliziert und deren Frequenzgänge gemessen, ohne die Position des Mikrofons, dessen Verstärkung und Einstellungen am Messsystem zu ändern. Der vom Normalzustand abweichende Frequenzgang ergibt sich aus dem Quotienten der komplexen Spektren eines Kanals mit und ohne Defekt. Die Spektren der Messstrecke, des Raumes, des Vorverstärkers und des Mikrofons, die in jeder Messung enthalten sind, werden durch diese Operation „herausgerechnet“. In Abbildung 3.4 ist dieser Vorgang für den Ausfall des Tieftöners dargestellt. Der ermittelte Ziel-Frequenzgang wurde anschließend mit einer Glättung von einer Oktave bearbeitet. Die gemessenen Frequenzgänge sind für die Darstellung ebenfalls geglättet worden.

3.3.1. Ausfall von Lautsprechern

Der Defekt eines Bauteils innerhalb der Serienschaltung hat die Änderung des Schaltkreises und damit der Übertragungseigenschaften des gesamten Kanals zur Folge. In Abbildung 3.5 sind die Schaltungen für Lautsprecherausfälle skizziert. Bei dem Ausfall von $LS1$ und/oder $LS2$ kommt es neben einem Leistungsverlust zu einer Serienschaltung des jeweiligen Kondensators und des nächsten Lautsprechers. Dies entspricht einem Hochpass erster Ordnung. Die Grenzfrequenz f_C von Filtern dieser Ordnung lässt sich mit

$$f_C = \frac{1}{2\pi CR} \tag{3.2}$$

berechnen. R ist hier ein reeler Widerstand. Da Lautsprecher über einen komplexen Widerstand verfügen, dient Gleichung 3.2 nur zur Orientierung. Die Grenzfrequenz f_C verhält sich antiproportional zu C. Je kleiner der Wert des Kondensators, desto hö-

[5] http://www.rme-audio.de/products_fireface_400.php
[6] http://www.behringer.com/DE/Products/ECM8000.aspx

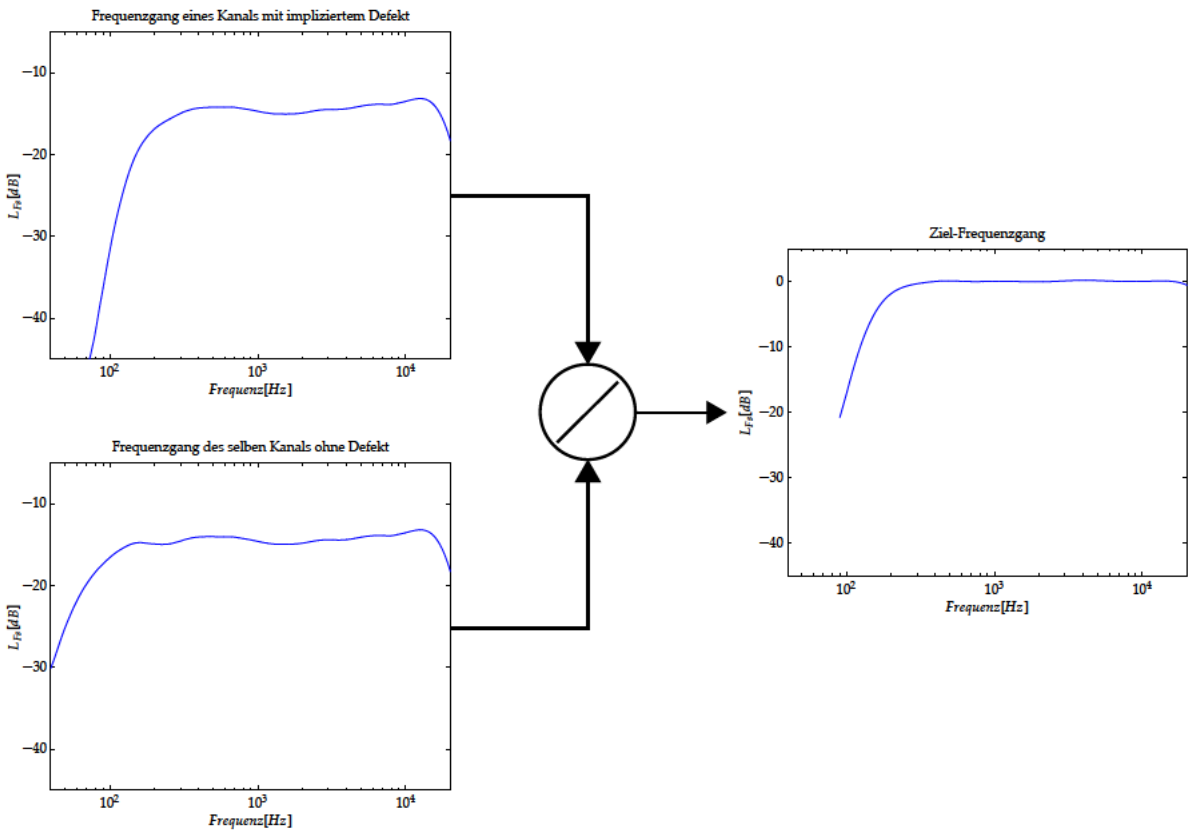
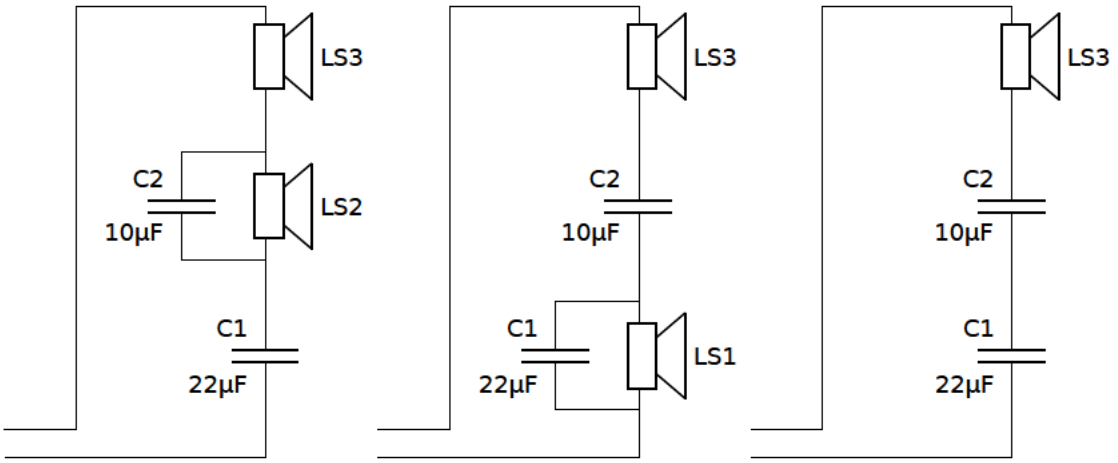


Abbildung 3.4.: Erzeugung des vom Normalzustand abweichenden Frequenzganges



(a) $LS1$ defekt (b) $LS2$ defekt (c) $LS1$ und $LS2$ defekt

Abbildung 3.5.: Ausfall von Lautsprechern

her ist also die Grenzfrequenz des Hochpasses. Abbildung 3.6 zeigt die Frequenzgänge der vier möglichen Lautsprecherausfälle. Das Hochpassverhalten ist gut an dem

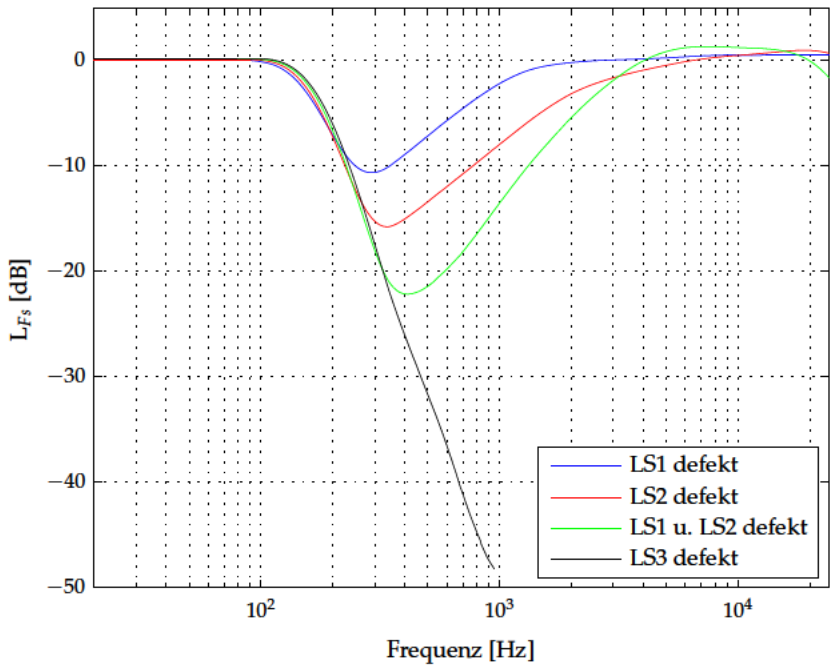


Abbildung 3.6.: Auswirkungen von defekten Lautsprechertreiber auf den Frequenzgang eines Lautsprecherkanals

Einbruch zwischen circa 120 Hz und 4 kHz zu erkennen. Unterhalb von 120 Hz wirkt der Tieftöner TT. Wie stark der Einbruch ist, hängt wie oben beschrieben mit der Größe des in Serie geschalteten Kondensators zusammen. Für den in Abbildung 3.5a dargestellten Fall addieren sich $C1$ und $C2$ zu $C_{ges} = 6{,}9\,\mu F$, was zur höchsten Grenzfrequenz und damit zum größten Einbruch führt. Fällt $LS3$ aus, wird der Schaltkreis unterbrochen und es klingt nur noch TT.

3.3.2. Ausfall von Kondensatoren

Auch die Kondensatoren $C1$ und $C2$ könnten ausfallen oder aber abreißen und den Schaltkreis, wie in Abbildung 3.7 zu sehen, verändern. Das Fehlen eines Kondensators hebt den Tiefpass auf, den er mit einem parallelgeschalteten Lautsprecher bildete, weswegen es zu einer Anhebung der oberen Frequenzen kommen müsste. In Abbildung 3.8 ist diese nur ansatzweise erkennbar und beträgt weniger als 1 dB. Auffällig ist, dass alle Frequenzgänge eine ähnliche Tiefpasscharakteristik aufweisen. Dies ist auch bei dem gleichzeitigen Ausfall von $LS1$ und $LS2$ in Abbildung 3.6 zu beobachten. In diesem Fall ist nur noch der Lautsprecher $LS3$ in Betrieb, zu dem

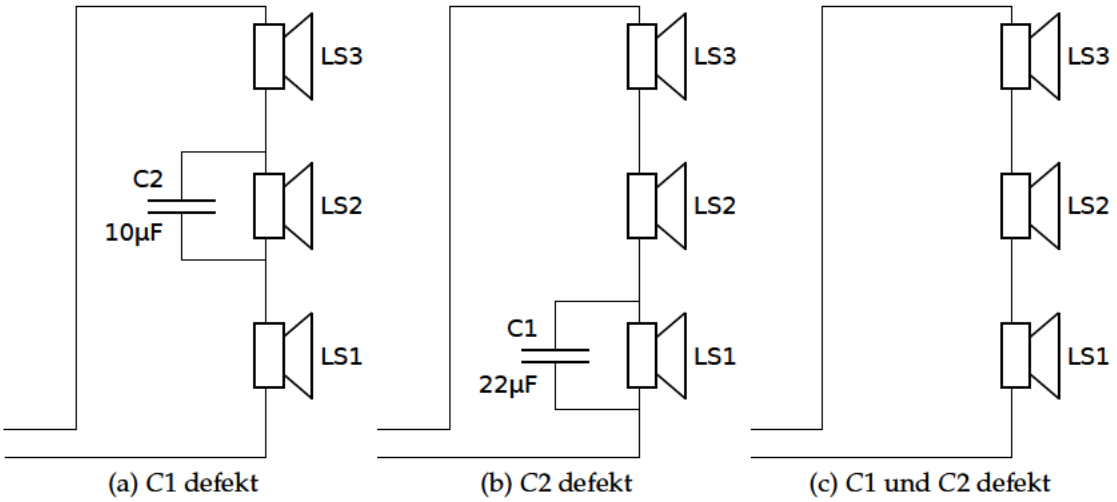


Abbildung 3.7.: Ausfall von Kondensatoren

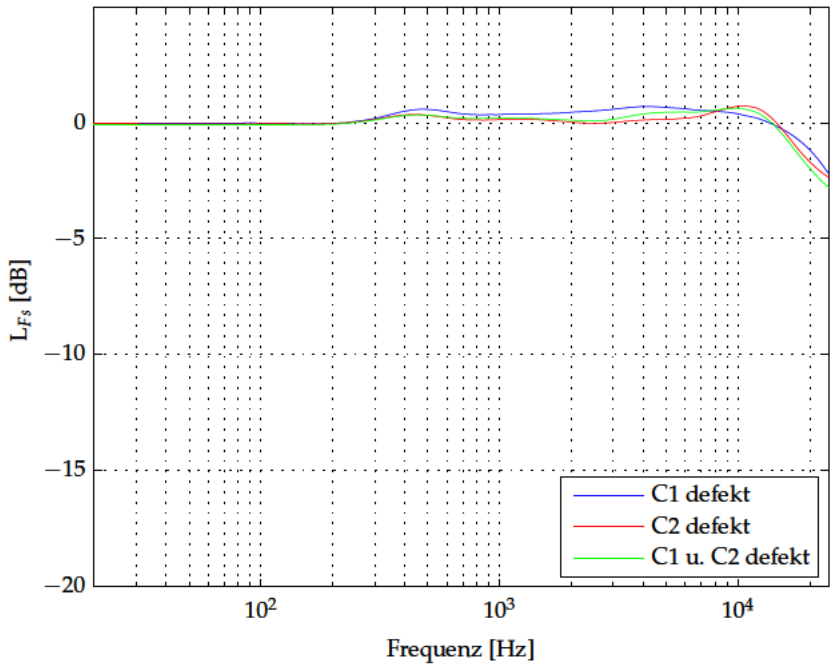


Abbildung 3.8.: Einflüsse von defekten Kondensatoren auf den Frequenzgang eines Lautsprecherkanals

kein Kondensator parallel geschaltet ist. Die Frequenzgänge der letztgenannten Verknüpfungen von Defekten werden durch das mit der Bauweise einhergehende Tiefpassverhalten geprägt.

3.3.3. Kombination aller Ausfälle

In den Tabellen 3.1a und 3.1b sind alle möglichen Kombinationen an Ausfällen von Bauteilen gelistet. 0 steht für eine defekte und 1 für eine funktionierende Komponente. Jede Zeile gibt die Kodierung eines Zustandsszenarios an. Diese wird im Laufe

TT	1.LS	2.LS	3.LS	1.Ko	2.Ko
1	1	1	1	1	1
1	1	0	1	1	1
1	0	1	1	1	1
1	0	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	0
1	1	1	1	0	0
1	1	0	1	0	1
1	0	1	1	1	0

(a) mit Tieftöner

TT	1.LS	2.LS	3.LS	1.Ko	2.Ko
0	1	1	1	1	1
0	1	0	1	1	1
0	0	1	1	1	1
0	0	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
0	1	1	1	0	0
0	1	0	1	0	1
0	0	1	1	1	0

(b) ohne Tieftöner

Tabelle 3.1.: Potentielle Schaltkreiszustände eines Lautsprecherkanals

der Arbeit weiter verwendet. Der Ausfall von $LS1$ wird zum Beispiel durch 101111 repräsentiert. Da der Ausfall von $LS3$ einen kompletten Ausfall der Breitbandlautsprecher bedeutet, sind die zusätzlichen Ausfälle von $LS1$ und/oder $LS2$ nicht in der Tabelle eingetragen. Zum Beispiel werden 110011, 111011 und 100011 durch 111011 repräsentiert.

3.3.4. Fazit

Es gibt zwanzig mögliche Schaltkreise. Der Ausfall von Kondensatoren hat nur geringen Einfluss auf den Frequenzgang eines Kanals. Für Messungen, die nicht unter Laborbedingungen stattfinden, kann eine zuverlässige Erkennung dieser Defekte ausgeschlossen werden. Kombinationen ausgefallener Lautsprechertreiber sind dagegen gut unterscheidbar, weswegen die Detektion solcher Defekte in dieser Arbeit weiter behandelt wird. Tabelle 3.2 fasst die zehn potentiellen Schaltkreise zusammen, die das Messsystem identifizieren soll.

TT	1.LS	2.LS	3.LS	1.Ko	2.Ko
1	1	1	1	1	1
1	1	0	1	1	1
1	0	1	1	1	1
1	0	0	1	1	1
1	1	1	0	1	1
0	1	1	1	1	1
0	1	0	1	1	1
0	0	1	1	1	1
0	0	0	1	1	1
0	1	1	0	1	1

Tabelle 3.2.: Zu detektierende Schaltkreiszustände eines Lautsprecherkanals

4. Bestimmung eines Filter-Sets zur Simulation der potentiellen Defekte

Um alle interessierenden Schäden eines Lautsprecherkanals im Raum hörbar zu machen und für die spätere Evaluation des Messsystems, wurde ein Set von digitalen Filtern erstellt, das es erlaubt, die möglichen Defekte zu simulieren. Ein manuelles Implizieren, also das Verändern des elektrischen Schaltkreises, ist mit einem hohen Aufwand verbunden und bei einem Lautsprecherssystem mit 832 Kanälen nicht praktikabel. So müsste das in der Wand fest eingelassene Lautsprechermodul mittels Leiter erreicht, das Schutzgitter abgeschraubt, das jeweilige oder die jeweiligen Lautsprecher-Chassis herausgeschraubt, abgesteckt und/oder Kondensatoren abgelötet und das Modul wieder zusammengebaut werden. Das Set wurde mit FIR-Filtern realisiert. Bevor in Kapitel 4.3 auf die konkrete Implementierung des Sets eingegangen wird, werden die für diese Aufgabe wichtigsten Sachverhalte der digitalen Signalverarbeitung in kurzer Form vorgestellt. Eine umfassende Beschreibung findet sich unter anderem in [Kammeyer und Kroschel, 2006; Oppenheim und Schafer, 1999; Feiten und Röbel, 1996].

4.1. Digitale Filter

Systeme, die frequenzabhängig die Amplitude und Phase von zeit- und wertdiskreten Signalen beeinflussen, werden als digitale Filter bezeichnet. Sie sind Teil einer wichtigen Unterklasse von LTI-Systemen, die durch die lineare Differentialgleichung beliebiger Ordnung mit konstanten Koeffizienten a_k und b_k

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=0}^{N} a_k y[n-k] \tag{4.1}$$

charakterisiert wird [Feiten und Röbel, 1996, Seite 22]. Mit $i \in N/0$ setzt sich das Ausgangssignal $y[n]$ aus den gewichteten Werten des aktuellen Eingangssignals $x[n]$ und je nach Ordnung den vorherigen Eingangssignalen $x[n-i]$ und Ausgangssignalen $y[n-i]$ zusammen. Sind die sogenannten Anfangsbedingungen $y[n_0-1], ..., y[n_0-N]$ bekannt, können alle folgenden Ausgangswerte iterativ berechnet werden. Da vorherige Ausgangswerte in die aktuelle Berechnung mit einbezogen werden, wird eine Gleichung wie Gleichung 4.1 *rekursiv* genannt. Filter dieser Art heißen auch *rekursive* oder *Infinite Impulse Response* (IIR)-Filter. Sie werden an dieser Stelle nicht näher erläutert.[1]

Ist $N = 0$ entfällt der *rekursive* Anteil von Gleichung 4.1 und es kann die *nichtrekursive* Gleichung

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{4.2}$$

abgeleitet werden. Die Ausgangswerte sind nur von dem aktuellen Eingangswert und dessen Vorgängern abhängig. Da die Gleichungen 4.2 und 2.1 identisch sind, entspricht die Folge der Koeffizienten b_k der Impulsantwort des entsprechenden LTI-Systems. Ihre Länge ist endlich, weshalb solche Systeme als FIR-Filter bezeichnet werden.

4.2. FIR-Filter

Ein FIR-Filter ist ein digitales, in der Regel nicht rekursives[2] LTI-System. Die Impulsantwort ist zeitlich beschränkt, was zur Folge hat, dass das BIBO[3]-Stabilitätskriterium stets erfüllt ist [vgl. Oppenheim und Schafer, 1999, Seite 14]. Auch unter Quantisierungseinflüssen ist dies der Fall.

Für diese Arbeit wurde ein Filter-Set nach einer in [Müller, 1999, Kapitel 4] beschriebenen Methode generiert. Die so entstandenen Filter weisen einen linearen Phasenverlauf, also eine konstante Gruppenlaufzeit, im Durchlassbereich auf. Im Gegensatz zu den erwähnten IIR-Filtern, die diese Eigenschaft nur aproximativ erreichen können, erfolgt so die Übertragung von Signalen verzerrungsfrei. Ausgangspunkt ist hier die Vorgabe des gewünschten reellen Frequenzganges des Filters. Es folgt die Transformation in den Zeitbereich mittels IDFT. Da die Phase und somit die Gruppen-

[1] Eine umfangreiche Beschreibung findet sich in [Oppenheim und Schafer, 1999].
[2] Eine Ausnahme bilden beispielsweise CIC-Filter; siehe dazu [Hogenauer, 1981].
[3] Bounded Input Bounded Output

laufzeit des Frequenzganges konstant Null gesetzt wurde, ist es notwendig, dass die Impulsantwort nun „um den halben Ausschnitt zyklisch vertauscht“ [Müller, 1999, Seite 157] wird, so dass sich der Peak dieser in der Mitte des Ausschnitts befindet. Durch Multiplikation mit einer Fensterfunktion werden nun die Filterkoeffizienten gewonnen. Mit der Länge L des Fensters wird die Filterordnung $N = L - 1$ festgelegt. Es wird eine gerade Filterordnung gewählt, so dass die Impulsantwort bezüglich $N/2$ symmetrisch ist. Nach [Meyer, 2006, Seite 288] entspricht dies dem Grundtyp 1 linearphasiger FIR-Filter. Filter dieses Typs können mit beliebigen Frequenzgängen konstruiert werden. Ihre Gruppenlaufzeit ist konstant und beträgt $N/2$ Werte.

Die Länge der Impulsantwort und die Art des Fensters haben direkten Einfluss auf den Frequenzgang des Filters, dessen Übertragungsfunktion sich aus der Faltung der Spektren des Fensters und der Impulsantwort ergibt. „Die Auflösung des FIR-Filters ist im *linearen* Frequenzmaßstab konstant“ [Müller, 1999, Seite 157]. Wird die Grenzfrequenz eines Tiefpassfilters bei gleicher Filterordnung verringert, nimmt also auch die Flankensteilheit des FIR-Filters im logarithmischen Frequenzmaßstab ab. Eine höhere Flankensteilheit kann durch eine längere Impulsantwort gewährleistet werden, wodurch sich die Gruppenlaufzeit und der Rechenaufwand vergrößern. In [Müller, 1999, Kapitel 4] wird eine Filterlänge vorgeschlagen, die mindestens der halben Wellenlänge der tiefsten zu beeinflussenden Frequenz entspricht. Für 20 Hz bei einer Abtastrate von 48 kHz sind dies 2400 Koeffizienten. Auch die Art des Fensters hat Einfluss auf die Flankensteilheit. Zudem definiert die Fensterfunktion, wie stark die Nebenmaxima gedämpft werden. Detailierte Informationen zu verschiedenen Fenstertypen und ihren Eigenschaften liefert [Harris, 1978].

4.3. Erstellen des Filter-Sets

Die Filtergenerierung erfolgt in zwei Schritten. Im ersten wird der Ziel-Frequenzgang definiert und im zweiten auf die oben geschilderte Weise die Impulsantwort erzeugt.

Für den ersten Schritt wird auf die in Kapitel 3.3 ermittelten Frequenzgänge aller möglichen Kombinationen von Komponentenausfällen zurückgegriffen. Diese dienen nun als Ziel-Frequenzgänge. Der zweite Schritt ist in Abbildung 4.1 skizziert. Um die Impulsantwort des Filters zu erhalten, wurde ein Kaiser-Bessel-Fenster verwendet. Die Form dieses Fensters kann über den Faktor α beeinflusst werden. Ein größeres α bedingt eine breitere Hauptkeule und eine höhere Sperrdämpfung. Für $\alpha = 0$ entspricht das Kaiser-Bessel-Fenster einem Rechteck-Fenster. Die Werte für $\alpha = 3{,}2$ und

die Länge $L = 2001$ Samples, die schließlich zum Einsatz kamen, wurden ermittelt, indem die Frequenzgänge der gefensterten Impulsantworten für verschiedene α und Fensterlängen mit den entsprechenden Ziel-Frequenzgängen verglichen wurden. Für $f_s = 48$ kHz bedeutet diese Filterlänge eine zeitliche Verzögerung von 20,8 ms. Die Auswirkungen der Impulsantwortlänge und Fensterfunktion sind im unteren Frequenzbereich gut zu erkennen. Die Flankensteilheit des Ziel-Frequenzgangs wird in ausreichendem Maße erreicht und die Dämpfung der ersten Nebenkeule beträgt circa 110 dB bei 30 Hz.

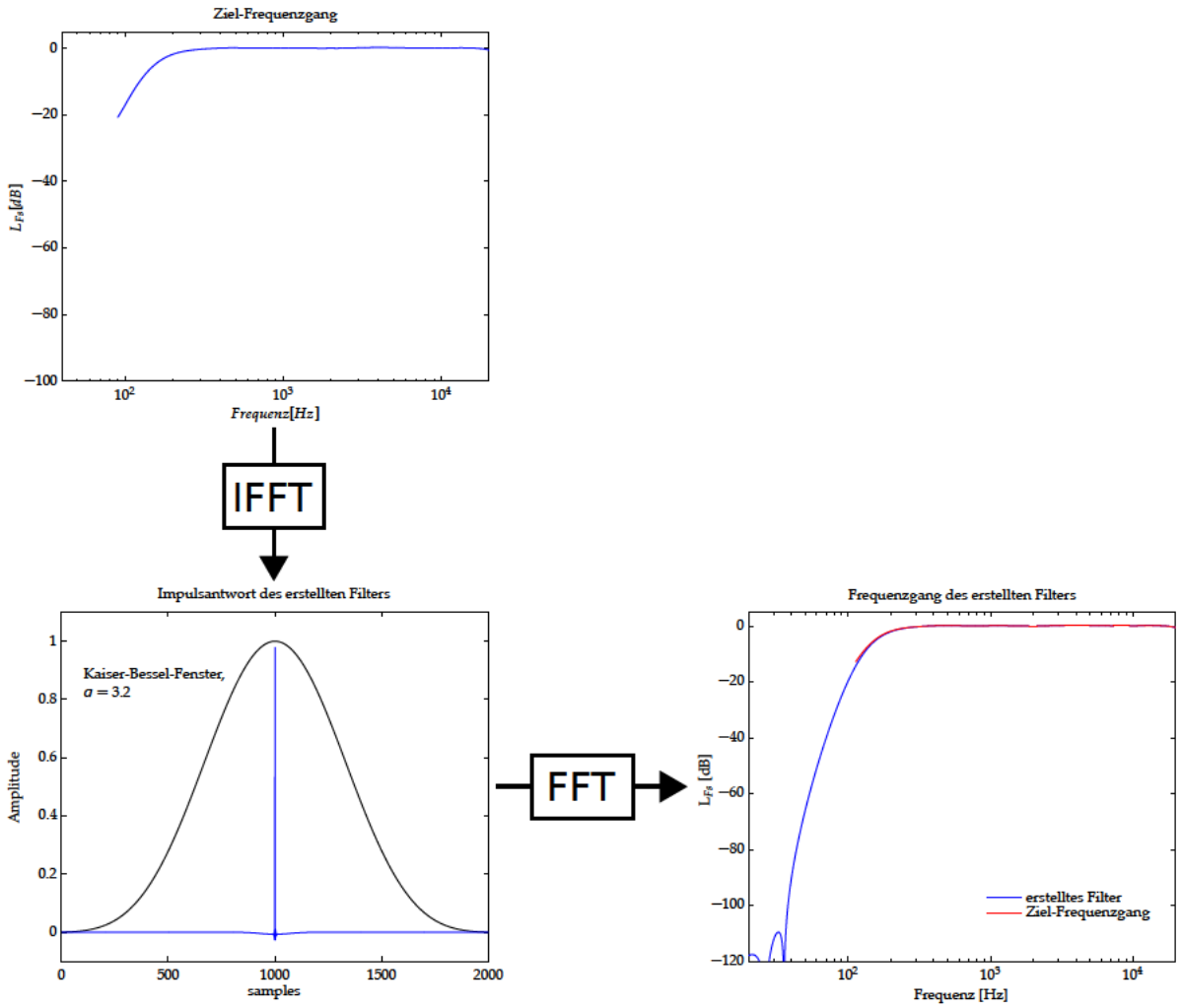


Abbildung 4.1.: Erzeugung der Filterkoeffizienten mit einem Kaiser-Bessel-Fenster der Länge 2001 Samples und $\alpha = 3.2$

5. Aufbau des Messsystems

5.1. Hardware-Komponenten

Das Messsystem ist auf eine Weise in das WFS-System im H0104 integriert, die es ermöglicht, Kontrollmessungen ohne zeitraubende Vorbereitungen durchzuführen. Änderungen der Hardware-Konfiguration oder Kabelwege sind nicht erforderlich. Für die Einrichtung des Systems kann gänzlich auf vorhandene Hardware-Komponenten zurückgegriffen werden. Folgende Geräte finden sich vor Ort:

- Beyerdynamic MC930[1]: Ein an der Rückseite des Hörsaales befestigtes Raummikrofon.
- Yamaha O2R96[2]: Ein mit einer MADI-Schnittstelle ausgestattetes Mischpult.
- 3 x RME MADI Bridge[3]: Eine MADI-Patchbay.
- 16 x RME ADI-648[4]: Ein bidirektionaler MADI-Formatkonverter mit integriertem 8-Kanal 16x16 Matrix Router.
- 104 x Lautsprechermodule[5]: Ein Panel mit jeweils acht Lautsprecherkanälen und ADAT-Eingang.
- 15 x Node: Rechner zur Berechnung der WFS, ausgestattet mit einem RME HDSPe MADI[6] Audiointerface.
- WFS-Control: Rechner mit installiertem Messsystem und RME HDSPe MADI Audiointerface.

[1] http://www.beyerdynamic.de/shop/mp/microphones/film-and-broadcasting/ambience-recordings/mc-930.html
[2] http://de.yamaha.com/products/music-production/mixers/02r96vcm/02r96vcm/?mode=model
[3] http://www.rme-audio.de/products_madi_bridge.php
[4] http://www.rme-audio.de/products_adi_648.php
[5] siehe Kapitel 3
[6] http://www.rme-audio.de/products_hdspe_madi.php

5.2. Audio-Routing

Im Regieraum, der über zwei MADI-Leitungen mit dem Rechnerraum, in dem die für die WFS notwendigen Einheiten stehen, verbunden ist, befindet sich ein Computer, der die Messsoftware beherbergt. Außerdem ist dort ein Mischpult eingebaut, das als Analog-Digital-Konverter und Vorverstärker des Messmikrofons fungiert. Als solches dient ein schon an die Rückwand des Hörsaals angebrachtes Raummikrofon. Abbildung 5.1 stellt die gesamte Hardware-Konfiguration sowie den Signalfluss dar. Das System erstreckt sich über drei Räume. Die Sample Rate und damit die Clock

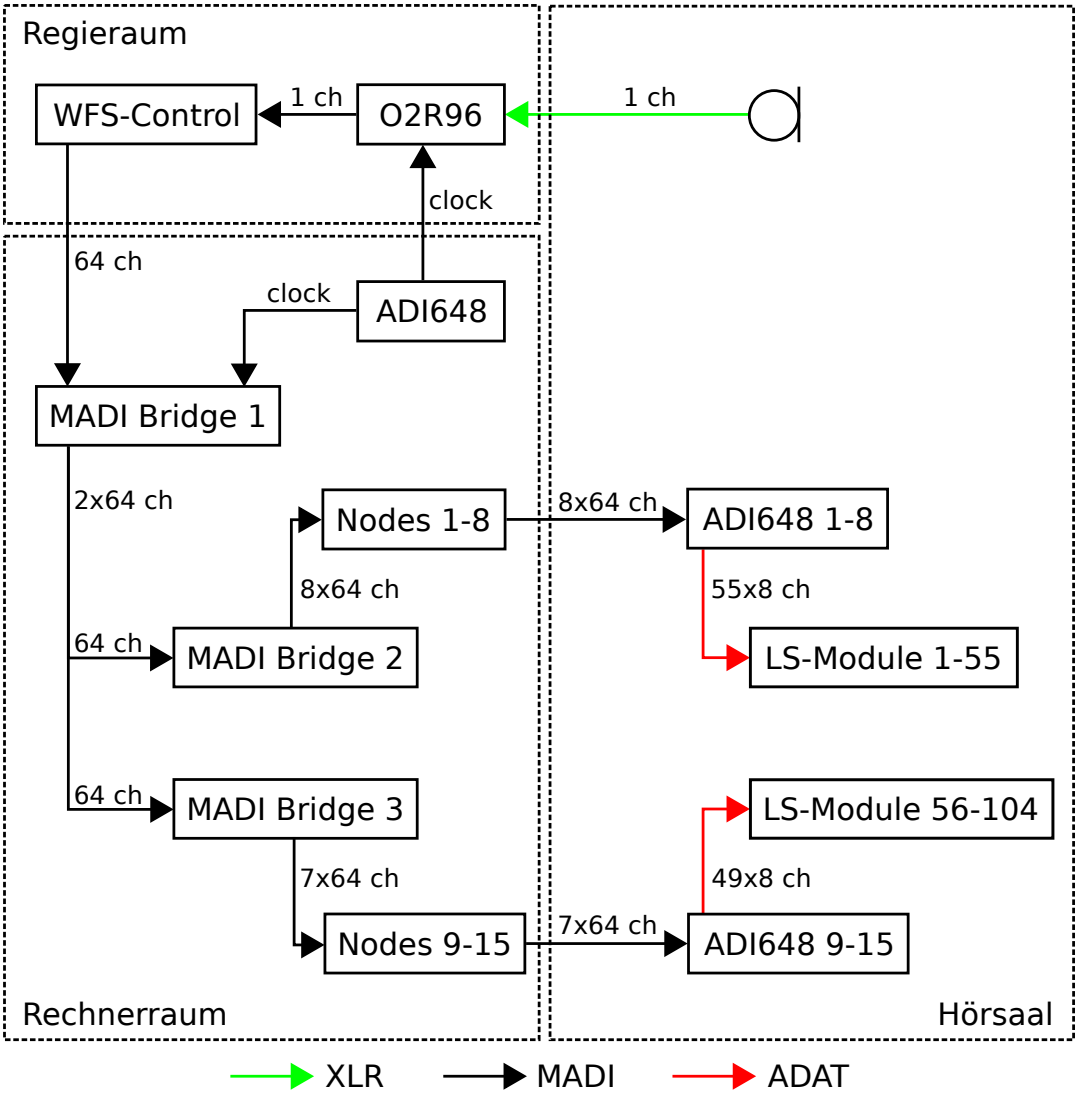


Abbildung 5.1.: Signalflussdiagramm des Messsystems im Raum H0104

des kompletten Systems wird durch das ADI-648 im Rechnerraum gegeben und über

sämtliche digitalen Leitungen übertragen. Das O2R96 leitet das vom Raummikrofon aufgenommene Signal über eine MADI-Leitung zu WFS-Control und damit in die Messsoftware. 64 Ausgangskanäle des Messrechners sind über eine MADI-Leitung mit der MADI Bridge 1 verbunden, von der sie auf die MADI Bridges 2 und 3 geroutet werden. Von dort führen sieben bzw. acht MADI-Wege jeweils 64 Kanäle in die entsprechenden Nodes, in denen die Signale mittels einer rechnerinternen Matrix den Ausgängen der Audiointerfaces zugeordnet werden. Sie sind via MADI mit den ADI-648 Einheiten verknüpft. Hier werden die sieben respektive acht MADI-Wege auf 49 bzw. 55 ADAT-Verbindungen verteilt und die insgesamt 104 Lautsprechermodule gespeist.

Jedwede Ausgangskanäle liegen an allen Nodes an, wie in Abbildung 5.2 zu sehen ist. Das Routing der Signale auf die einzelnen Lautsprechermodule und ihre Kanäle

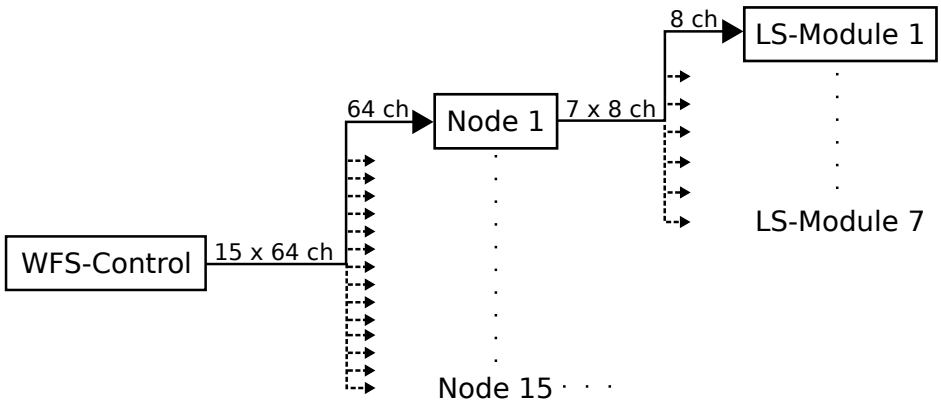


Abbildung 5.2.: Signalweg zwischen WFS-Control und einem Lautsprechermodul

wird in den Nodes mit Hilfe der Software-Matrix JACK[7] vollzogen. Mit ihr können jegliche Ein- auf Ausgangsverbindungen realisiert werden.

5.3. Programmierumgebung und zusätzliche Software-Komponenten des Messsystems

Das Messsystem nutzt mehrere Software-Komponenten, die im Folgenden kurz vorgestellt werden. Es wird zwischen der eigentlichen Programmierumgebung und zusätzlicher Software unterschieden.

[7] siehe Kapitel 5.3.2

5.3.1. GNU Octave

Die Programmierung des Messsystems erfolgte mittels der GPL lizensierten höheren Programmiersprache GNU Octave. Sie ist größtenteils kompatibel zum proprietären MATLAB® von Mathworks. Unterschiede wurden bei der Entwicklung des Messsystems beachtet, wodurch es ebenso möglich ist, das Messsystem unter MATLAB® zu nutzen. GNU Octave stellt eine interaktive Skriptsprache dar. Befehlssequenzen können als Skripte oder Funktionen abgespeichert und verwendet werden. Es stehen diverse Toolboxes für verschiedene Aufgabengebiete zur Verfügung. So existiert eine Signal-Processing-Toolbox, welche zum Beispiel die für die Realisierung des Messsystems wichtigen Funktionen der FFT- sowie Filter-Prozesse bereitstellt. Ein weiteres Funktionsmerkmal ist das dynamische Laden von in C++ oder C geschriebenen Modulen. Um GNU Octave als JACK-Client nutzen zu können, wurde das von Robert Humphrey entwickelte Modul Playrec (siehe [Humphrey]) herangezogen. Playrec gewährleistet eine Schnittstelle zum Audiointerface via des JACK Audioservers mit einer beliebigen Anzahl von Audiokanälen.

5.3.2. JACK - JACK Audio Connection Kit

JACK ist für die Betriebssysteme GNU/Linux, Mac OS X und MS Windows verfügbar und beinhaltet einen Audioserver sowie eine Programmierschnittstelle. Der Audioserver kann unter anderem die Treiber ALSA (Advanced Linux Sound Architecture) und Core Audio als Backend nutzen, wodurch eine Schnittstelle zum Audiointerface gewährleistet ist. Programme zur Audiosignalverarbeitung melden sich bei dem Server als JACK-Clients an, können sich über diesen mit der Audiohardware verbinden und/oder eine beliebige Anzahl von Audiokanälen untereinander synchron austauschen. Das Signalrouting kann über eine grafische Oberfläche oder ebenfalls enthaltende Kommandozeilenprogramme gesteuert werden.

5.3.3. Weitere Software für das Lautsprechersystem im Raum H0104

Aufgrund des Aufbaus des WFS-Systems im H0104 muss die Messsoftware auf WFS-Control in der Lage sein, über Netzwerk den JACK Audioserver auf den Nodes zu starten bzw. zu stoppen und das Routing der Audiosignale innerhalb der Nodes fernzusteuern. Abbildung 5.3 zeigt die Netzwerksituation für die netzwerkfähigen Kom-

ponenten des Messsystems. WFS-Control und Nodes befinden sich in verschiede-

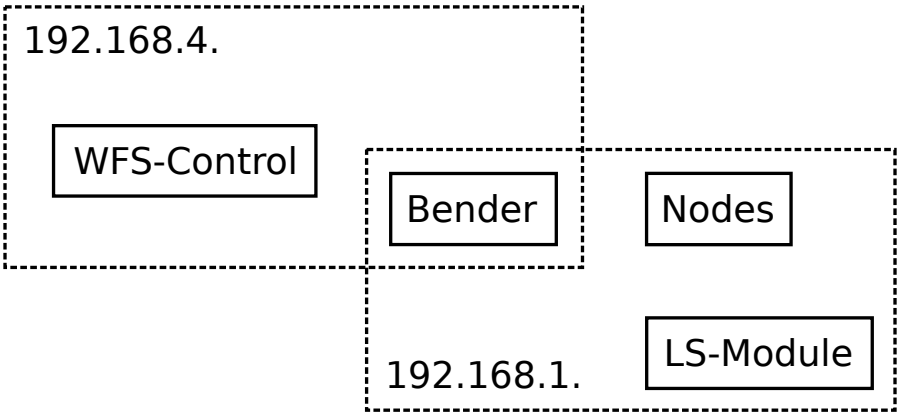


Abbildung 5.3.: Netzwerke des Messsystems im Raum H0104

nen Netzwerken, weswegen die Steuerbefehle den Umweg über den Computer Bender machen müssen, der in beide Netzwerke eingebunden ist. Bender ist die Hard- und Software-Schnittstelle zum WFS-System, dessen Kontrollsoftware zudem auf diesem Rechner installiert ist. Von WFS-Control aus werden Skripte auf Bender ausgeführt, die wiederum via Netzwerk die notwendigen Kommandozeilenprogramme mit entsprechenden Parametern auf den Nodes starten. Die Open Secure Shell, kurz OpenSSH, stellt Programme zur Verfügung, die verschlüsselte Verbindungen mittels des SSH-Protokolls ermöglichen und mit denen Skripte und Programme auf entfernten Rechnern ausgeführt werden können. Dies entfällt für vielkanalige Lautsprechersysteme, die wie das WFS-System im Raum EN325 mit einem Computer betrieben werden.

5.4. Audio-Routing innerhalb des Messsystem-Rechners

Anhand von WFS-Control wird das Signalrouting innerhalb des Rechners, auf welchem das Messsystem installiert wurde, erläutert. Abbildung 5.4 zeigt schematisch die Software-Komponenten, die für diese Aufgabe herangezogen werden. Da WFS-Control mit einem GNU/Linux-Betriebssystem ausgestattet ist, wird als Backend des JACK Audioservers der ALSA-Treiber verwendet. Über Playrec meldet sich GNU Octave als JACK-Client an. Das Messsystem verfügt nun über eine vom Audiointer-

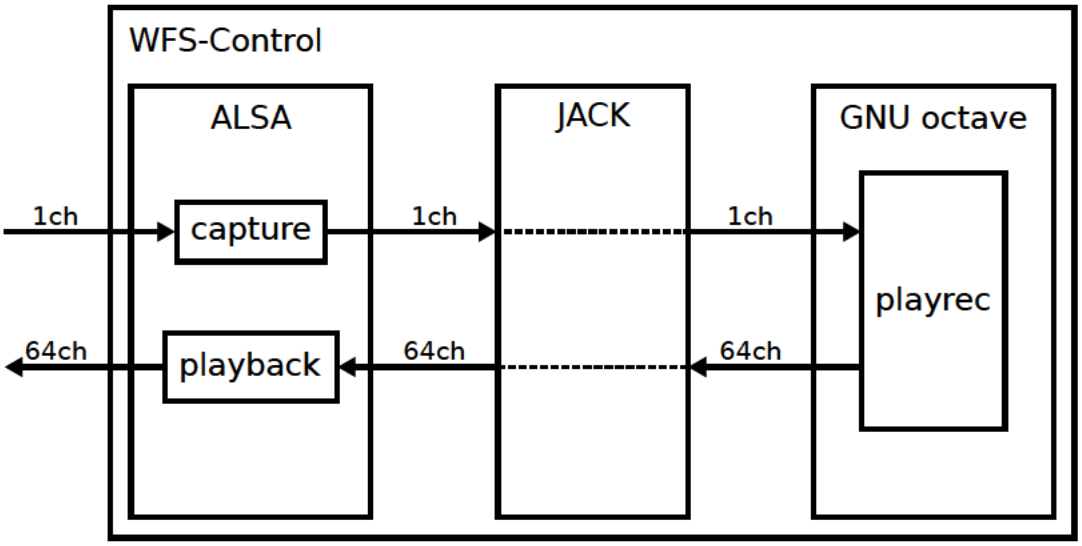


Abbildung 5.4.: Routing innerhalb von WFS-Control

face vorgegebene Anzahl von Ausgangskanälen sowie einen Eingangskanal für das Mikrofonsignal.

6. Messsoftware

Die Messoftware ist, wie in Kapitel 5.3.2 beschrieben, mit GNU Octave realisiert worden und vollständig kompatibel mit MATLAB®. Sie läuft unter den unixoiden Betriebssystemen GNU/Linux und Mac OS X. Zusätzlich benötigte Software ist JACK, als Schnittstelle zur Audio-Hardware und Routing-Matrix, und das GNU Octave- bzw. MATLAB®-Modul Playrec, welches die Messsoftware als JACK-Client beim JACK-Audioserver anmeldet.

6.1. Wahl der Messmethode

Um die Forderung nach einer möglichst kurzen Dauer des Messvorgangs zu erfüllen, wird die in Kapitel 2.3 vorgestellte MESM-Methode herangezogen. Für die Detektion potentieller Defekte spielt das Erreichen eines hohen SNR eine untergeordnete Rolle, weswegen auf die MESM-Erweiterung für die Messung mit arbiträr gefärbten Sweeps verzichtet wurde. Da die MESM-Methode für Binauralmessungen in Räumen mit vernachlässigbaren Nachhallzeiten entwickelt wurde, stellt sich die Frage, inwieweit sich die Messdauer in einem Raum, wie dem H0104 mit Nachhallzeiten von mehr als 600 ms, reduziert. Im Raum H0104 wurden Impulsantworten verschiedener Lautsprecherkanäle gemessen und die zur Berechnung der Messzeiten mittels Overlapping, Interleaving und MESM Parameter abgelesen und deren Mittelwerte gebildet. Mit diesen wurden dann die Messzeiten nach den Gleichungen 2.9 (sequentielle Messung), 2.12 (mittels Interleaving), 2.15 (mittels Overlapping) und 2.17 (mittels MESM) bestimmt. Die Ergebnisse sind in Tabelle 6.1 aufgelistet und machen deutlich, dass die Messung via Overlapping-Methode zeitlich am effektivsten ist. Daraufhin wurden die Berechnungen für das WFS-System im Raum EN325 wiederholt. Die Nachhallzeit beträgt dort etwa 200 ms. Tabelle 6.2 zeigt die resultierenden Messzeiten. Hier ist die Messdauer für die Messung mittels Overlapping-Methode ebenfalls am geringsten, wenn auch die zeitliche Differenz weniger eindeutig ist.

Methode	Messdauer
Sequentielle Messung	~30 Minuten
Mit Interleaving	~90 Minuten
Mit Overlapping	~11 Minuten
Mit MESM	~15 Minuten

Tabelle 6.1.: Theoretische Messzeiten für verschiedene Messmethoden im Raum H0104. Parameter: $N = 832$, $L_1 = 0.65\,s$, $K = 2$, $L_2 = 0.2\,s$, $L_1 = 0.65\,s$, $T = 1.49\,s$, $\omega_1 = 30\,Hz$, $\omega_2 = 20\,kHz$

Methode	Messdauer
Sequentielle Messung	~3.6 Minuten
Mit Interleaving	~4.4 Minuten
Mit Overlapping	~0.8 Minuten
Mit MESM	~0.9 Minuten

Tabelle 6.2.: Theoretische Messzeiten für verschiedene Messmethoden im Raum EN325. Parameter: $N = 128$, $L_1 = 0.2\,s$, $K = 2$, $L_2 = 0.2\,s$, $T = 1.49\,s$, $\omega_1 = 30\,Hz$, $\omega_2 = 20\,kHz$

Die Wahl der Messmethode fiel entsprechend auf die Overlapping-Methode mit exponentiellen Sweeps. Sie wurde in die Messsoftware implementiert.

6.2. Überblick

Die Messsoftware kann in vier unterschiedlichen Modi, definiert durch den angegebenen Messtyp, betrieben werden. In Tabelle 6.3 sind die Messtypen aufgelistet und kurz beschrieben. Der Typ *check* stellt hier die Standardeinstellung dar. Mit ihm

Name	Funktion
check	Überprüfung der Lautsprecherkanäle anhand der Referenzdaten
pre	Akustische Referenzmessung zur Akquise der für die MESM benötigen Parameter sowie der Referenzfrequenzgänge und Referenzmaximalamplituden
reference	Messung der Referenzfrequenzgänge und Referenzmaximalamplituden
typical	Einfache Messung der Impulsantworten

Tabelle 6.3.: Überblick über die vier möglichen Messtypen der Software

werden die Kanäle des Lautsprechersystems getestet und etwaige Defekte identifiziert. Bevor dies geschehen kann, muss mittels *pre* eine akustische Referenzmessung getätigt werden. Diese liefert die für zeitoptimierte Messungen nötigen Parameter zur Sweepsynthese sowie die Daten, welche dann zur Detektion potentieller Defekte herangezogen werden. Letztere können auch separat durch Messungen des Typs *reference* gewonnen werden. Ist die einfache Akquise von Impulsantworten von Interesse, ist der Typ *typical* vorzuziehen.

Abbildung 6.1 zeigt das Ablaufdiagramm der Software in zusammengefassten Arbeitsschritten. Alle Einstellungen der Messsoftware werden in einer *ini*-Datei vorge-

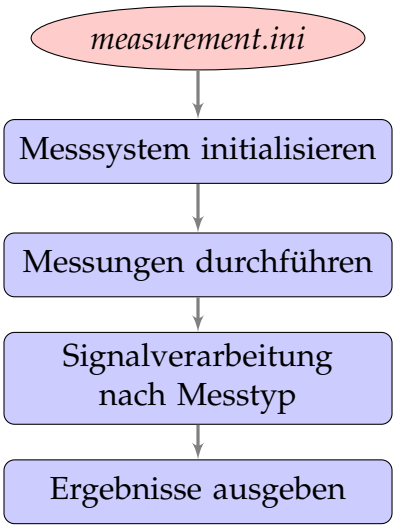


Abbildung 6.1.: Ablaufdiagramm der Messsoftware

nommen, die von der Messsoftware eingelesen wird. Die angegebenen Parameter werden sodann in das Struct »setup« geschrieben. Es folgt eine Initialisierung des Systems. Nach Beenden der Akquise der Systemreaktion auf die Sweep-Matrix wird eine Signalverarbeitung vorgenommen, die abhängig von dem definierten Messtyp ist. Schlussendlich werden die Ergebnisse ausgegeben. Ein *log*-File protokolliert während des Programmablaufs alle Textausgaben über die Kommandozeile. Kommt es zu einem unerwarteten Abbruch der Software, ist diese Datei im Systemordner zu finden. Andernfalls wird sie im für die Messung angelegten Ordner abgelegt.

6.3. Initialisierung des Messsystems

Die Initialisierung des Systems, wie in Abbildung 6.2 dargestellt, richtet sich nach den Parametern, die in dem *ini*-File *measurement.ini* festgelegt werden. Für das im

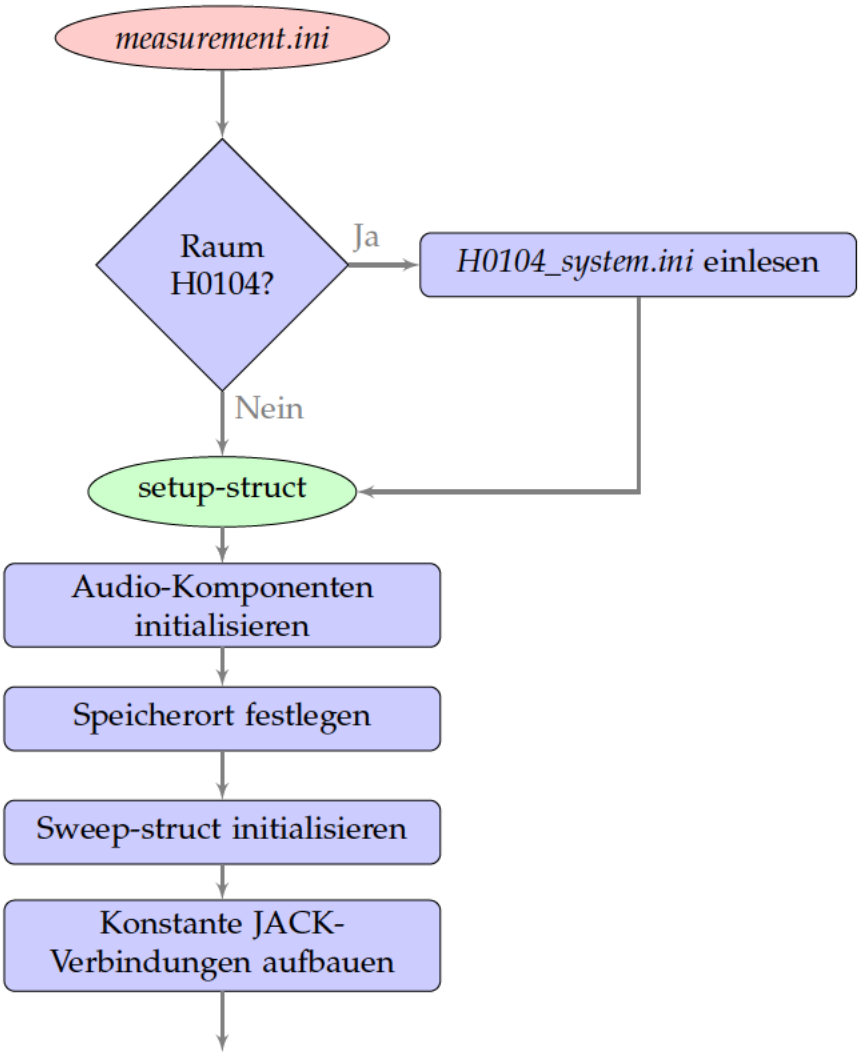


Abbildung 6.2.: Ablaufdiagramm der Systeminitialisierung

Raum H0104 installierte System wird zusätzlich noch die Datei *H0104_system.ini* geladen. In ihr ist definiert, wo sich der Ordner auf Bender befindet, in dem die Skripte zur Fernsteuerung der Nodes liegen, und wie die JACK-Parameter der Nodes lauten. Die Einstellungen werden in das Struct »setup« geschrieben und allen Softwarekomponenten als globale Variable zur Verfügung gestellt. Nachdem die Audio-Komponenten initialisiert wurden, werden der Ordner zur Speicherung der Teil- und Endergebnisse angelegt, die angegebene Sweep-Struct überprüft oder eine neue erstellt, sowie die JACK-Verbindungen etabliert, welche während des sich anschließenden Messvorgangs nicht dynamisch geändert werden.

6.3.1. Initialisierung der Audio-Komponenten

Der Programmablaufplan zur Initialisierung der Audio-Komponenten der Messsoftware ist in Abbildung 6.3 zu sehen. Im Normalfall besteht dieser Vorgang aus dem

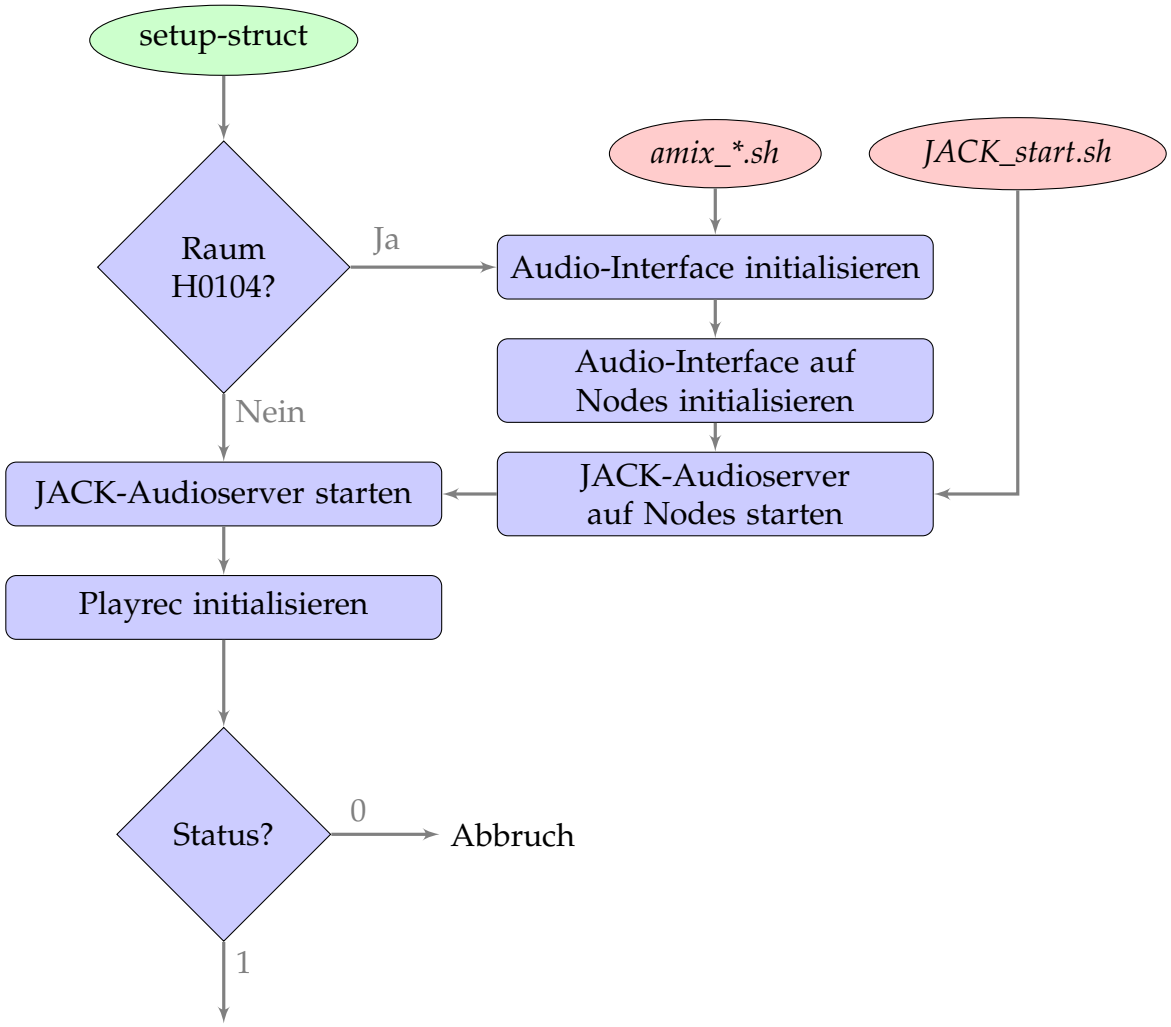


Abbildung 6.3.: Initialisierung der Audio-Komponenten

Starten des JACK-Audioservers und der Initialisierung des Playrec-Moduls mit den im Struct »setup« eingetragenen Parametern. Für das System im Raum H0104 ist es außerdem erforderlich, auf den Nodes JACK-Audioserver zu starten. Dies geschieht durch das Ausführen des Skripts *JACK_start.sh* auf Bender. Da alle am Messsystem im H0104 beteiligten Rechner mit RME HDSPe MADI Audio-Interfaces ausgestattet und deren Einstellungen über das Kommandozeilenprogramm amixer einseh- und veränderbar sind, werden diese von der Messsoftware überprüft und gegebenenfalls angepasst. Kann einer der genannten Arbeitsschritte nicht ordnungsgemäß ausge-

führt werden, wird die Status-Variable auf Null gesetzt. Die abschließende Statusabfrage hat dann einen Abbruch des Messvorgangs zur Konsequenz.

6.3.2. Initialisierung des Speicherpfads

Neben einem generellen Speicherpfad für die Messergebnisse, definiert durch den Eintrag *measurement* → *globalSaveToPath* im *ini*-File, bekommt jede Messung einen Namen über *measurement* → *name* im *ini*-File zugewiesen. Daraufhin wird ein gleichnamiger Ordner im Speicherpfad angelegt. Alle während der Messung erzeugten Dateien werden dort hineingeschrieben. Wird ein schon existierender Name angegeben, ist die Messsoftware mittels Nummerierung in der Lage, einen noch nicht vorhandenen Namen zu generieren.

6.3.3. Initialisierung der Sweep-Matrix

Um Messungen durchführen zu können, muss der Messoftware ein verarbeitbares Struct namens »swp« vorliegen. Es enthält unter anderem die Sweep-Matrix, dessen Reihen die Sweeps für die jeweiligen Kanäle, repräsentiert durch die Spalten, enthalten. Abbildung 6.4 zeigt eine Sweep-Matrix für drei Kanäle mit einer Verzögerungszeit von 200 ms. Der Grundbaustein zur Generierung einer solchen Matrix

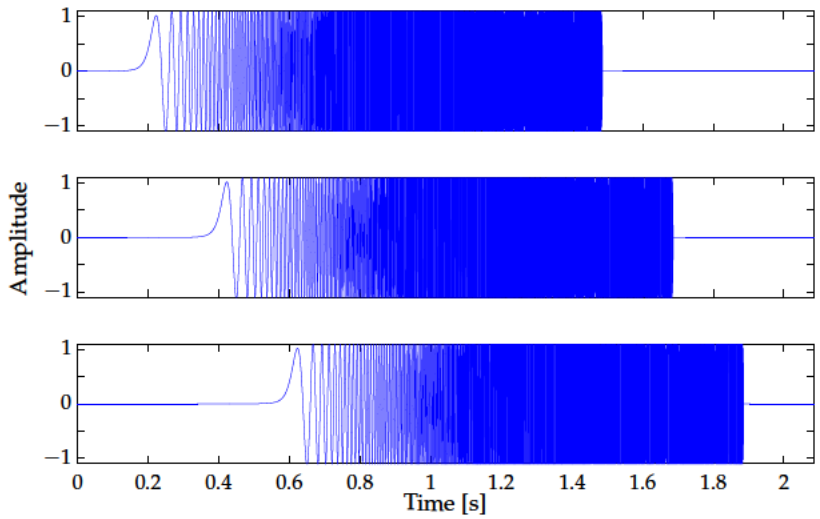


Abbildung 6.4.: Sweep-Matrix für die Messung von drei Lautsprecherkanäle.

stellt ein exponentieller Sweep dar, auf dem die in [Majdak u. a., 2007] vorgestellte

MESM-Methode fußt. Er wird mit einem von André Giese[1] realisierten MATLAB®-Skript erzeugt. Der zeitliche Abstand zwischen den Sweeps ergibt sich aus den in einer akustischen Referenzmessung ermittelten zeitlichen Ausdehnungen der Klirrkomponenten und Nachhallzeiten für alle Lautsprecherkanäle sowie den akustischen Laufzeiten zwischen diesen und dem Messmikrofon. Die Erfassung dieser Parameter mittels akustischer Referenzmessung und die Berechnung der zeitlichen Verzögerung zwischen den Sweeps wird in Kapitel 7 beschrieben.

Initialisierungsmethoden

Die Initialisierung des Structs »swp« ist abhängig vom Messtyp und kann auf drei Arten geschehen. Diese Initialisierungsmethoden sind *build*, *specific* und *file*. Die Parameter, die für die verschiedenen Initialisierungsmethoden im *ini*-File unter *Sweep* definiert werden können, sind in Tabelle 6.4 aufgelistet.

Feld	*build*	*specific*	*file*
amp	Austeuerung der Sweeps in dB_{Fs}		
name		Name der »swp«-Struct-Datei Mittels * können mehrere Dateien gewählt werden	
maxChans	Maximale Anzahl sich überlappender Sweeps		
tGap	Pfad zu und Name der »tGap«-Struct-Datei oder Einzahlwert für feste Verzögerungszeiten		Wert in Sekunden
runtimeFile	Pfad zu und Name der »runtime«-Struct-Datei oder Einzahlwert für konstante akustische Laufzeiten		
path		Pfad zu den Sweep-Dateien	

Tabelle 6.4.: Felder des Abschnitts *Sweep* im *ini*-File und ihre Bedeutung für die unterschiedlichen Initialisierungstypen

Die Initialisierungsmethoden *specific* und *build* generieren für jeden zu messenden Kanal einen spezifischen Sweep innerhalb der Sweep-Matrizen. Bei einer Messung von 832 Kanälen, bei 32 sich überlappenden Sweeps, kommen beispielsweise 26

[1] siehe [Giese, 2009, Seite 120-124]

»swp«-Structs zum Einsatz. Variierende akustische Laufzeiten, Nachhallzeiten und zeitliche Ausdehnungen der Klirrkomponenten zwischen den Kanälen können so berücksichtigt werden. Dieser Initialisierungsmethoden bedient sich die Software bei Messungen mittels der Messtypen *check*, *reference* oder *typical*. Die Methode *specific* setzt voraus, dass zuvor Sweep-Dateien für die zu messende Lautsprecheranlage generiert wurden. Bei der Initialisierung wird geprüft, ob die Dateien existieren und die Kanäle mit den zu messenden Kanälen übereinstimmen. Gegebenenfalls wird eventuell die Aussteuerung der Sweeps angepasst. Die Dateien werden dann lediglich geladen, was einen schnellen Ablauf der Initialisierung gewährleistet. Sind die angegebenen Sweep-Dateien nicht vorhanden oder werden Unstimmigkeiten bei den Kanälen erkannt, bedient sich die Software der Initialisierungsmethode *build*, die es ermöglicht, Sweep-Dateien bei der Initialisierung zu generieren. Hierbei können die gemessenen akustischen Laufzeiten und die Gap-Zeiten mit einbezogen werden, indem die entsprechenden Dateien nebst Speicherort im *ini*-File angegeben werden. Auf diese Weise können Sweep-Dateien erstellt werden, falls nicht das gesamte Lautsprechersystem gemessen werden soll. Optional können auch Einzahlwerte in Sekunden für konstante akustische Laufzeiten bzw. Gap-Zeiten eingetragen werden. In diesem Fall wird die Ordnung der Sweeps von der Software so gewählt, dass die Sweep-Dauer größer als die angegebene Gap-Zeit ist. Ansonsten ist die Sweep-Ordnung durch das entsprechende Feld im Struct »tGap« festgelegt. Die maximale Anzahl sich überlappender Sweeps pro Datei kann ebenfalls definiert werden, wird aber durch die Anzahl der Ausgänge des Audio-Interface begrenzt. Eine geringere Anzahl ist zu wählen, falls es bei der Messung bedingt durch einen zu kleinen Arbeitsspeicher zu Ausfällen bei der Audioübertragung kommt. Als Speicherort für Sweep-Dateien, die während der System-Initialisierung erzeugt werden, dient ein Ordner namens *temp*, der im Ordner der jeweiligen Messung automatisch angelegt wird.

Ein dritter Typ der Initialisierung ist *file*. Er wird indirekt über die Wahl des Messtyps *pre* gewählt. Bei der dadurch stattfindenden akustischen Referenzmessung werden ohne Anwendung der MESM-Methode die Lautsprecherkanäle sequentiell vermessen. Im Gegensatz zu den Typen *specific* und *build* wird bei *file* nur eine Sweep-Datei angelegt, deren »swp«-Struct nur eine einkanalige Sweep-Matrix mit fester Gap-Zeit enthält. Laufzeiten und Gap-Zeiten der einzelnen Kanäle können nicht definiert werden, da diese ja gerade durch den Messtyp *pre* ermittelt werden sollen. Nur die *ini*-File-Einträge *amp* und *tGap* haben in diesem Fall Einfluss auf die Initialisierung, wobei hier *tGap* ein Einzahlwert in Sekunden sein muss. Die Software berech-

net abhängig von diesem die Sweep-Ordnung und setzt die Initialisierungsmethode auf *file*.

Aufbau des Structs »swp«

Die einzelnen Felder des Structs »swp« zeigt Abbildung 6.5. Neben der Sweep-Matrix

Name	Feld	Funktion
swp	mesmSwp	Sweep-Matrix
	channels	Kanäle
	N	Sweep-Ordnung
	amp	Aussteuerung der Sweeps in dB_{Fs}
	singleSwp	Einzelner Sweep ohne Gap-Zeit
	tGap	Gap-Zeiten der einzelnen Kanäle
	runtimes	Kanalbezogene akustische Laufzeiten
	tCompensate	Zeitliche Korrekturen durch Laufzeitkompensation
	startTimes	Startzeiten der einzelnen Sweeps
	errors	Defektbezeichnungen für Testmessungen

Tabelle 6.5.: Aufbau des Structs »swp«

sind dort die zu den Sweeps gehörenden Kanäle; ein einzelner Sweep, der für die elektrische Referenzmessung herangezogen wird; Bezeichnungen implizierter Defekte für Testmessungen, sowie Informationen zur Erzeugung der Sweep-Matrix gespeichert.

6.3.4. Aufbau konstanter JACK-Verbindungen

Den Abschluss der Systeminitialisierung bildet der Aufbau der statischen Verbindungen innerhalb der JACK-Routing-Matrix. Dazu gehört immer die Verbindung des Kanals der Audio-Hardware, an dem das Messmikrofon angeschlossen ist, mit dem Eingangskanal der Messsoftware. Enthält die Sweep-Datei alle zu messenden Kanäle, werden auch schon sämtliche Ausgänge der Messsoftware mit jenen der Audio-Hardware verbunden. Bei Messungen in mehreren Etappen, werden die JACK-Verbindungen später dynamisch aufgebaut. Eine Ausnahme bildet das Messystem im Raum H0104. Das eigentliche Routing auf den Nodes geht dynamisch vonstatten, während die Ausgänge der Messsoftware mit der Audio-Hardware konstant kurzgeschlossen sind.

6.4. Messungen

Die Entfaltung ist das mathematische Werkzeug FFT-basierter Messsysteme, mit dem die Übertragungsfunktion eines Prüflings gewonnen wird. Wie in Kapitel 2.2 beschrieben, ist hierfür zum einen eine elektrische Referenzmessung durchzuführen und zum anderen die Reaktion des Prüflings auf den selben Stimulus aufzunehmen. Neben diesen Arbeitsschritten wird im vorliegenden Kapitel die Software-Komponente besprochen, die das Messsystem auf Übersteuerung des Eingangspegels überprüft.

6.4.1. Elektrische Referenzmessung

Die Messsoftware bietet drei Möglichkeiten, die Daten einer elektrischen Referenzmessung in Form eines »reference«-Structs bereitzustellen. Der Eintrag $reference \rightarrow measurement$ im *ini*-File kontrolliert, wie dies geschieht. Wird in diesem eine Datei angegeben, die ein »reference«-Struct enthält, wird diese geladen. Lautet der Eintrag 0, wird die zuvor initialisierte Sweep-Datei geöffnet, aus dem »swp«-Struct der einzelne Sweep, der die Grundlage der Sweep-Matrix bildet, extrahiert und in das »reference«-Struct gespeichert. Vor ihm wird noch ein Stück Stille eingefügt, dessen Länge sich aus den Buffer-Zeiten von JACK berechnet. Wird *measurement* gleich 1 gesetzt, erfolgt eine Messung mit den übrigen Parametern, die in $reference$ angegeben werden können. Hier gilt es, über *device* den zu verwendenden JACK-Client und über *input* bzw. *output* dessen Ein- und Ausgang zu wählen. Nach Überprüfung der Parameter wird dann eine Messung durchgeführt.

Da das vorgestellte Messsystem vornehmlich zur Identifikation potentieller Defekte von Lautsprechersystemen dient und dessen Detektionsalgorithmus keine exakten Messergebnisse benötigt, ist die Verwendung des Sweeps als Referenzwert ausreichend.

6.4.2. Kontrolle des Eingangspegels

Die Übersteuerung des Mikrofonsignals hat Einfluss auf die gemessenen Frequenzgänge und sollte vermieden werden. Aus diesem Grund gilt es, den Eingangspegel diesbezüglich zu überprüfen. Dazu wird ein Sweep mit den Eigenschaften der initialisierten Sweeps über einen oder sequentiell über mehrere Lautsprecher abgespielt und die maximale Amplitude am Eingang des Messsystems erfasst. Unter Be-

rücksichtigung der Anzahl sich überlappender Sweeps wird dann berechnet, ob eine Übersteuerung möglich ist. Über den Eintrag *measurement* → *gainCheck* im *ini*-File können die Lautsprecherkanäle übergeben werden. Wird der Eintrag freigelassen, findet keine Überprüfung statt.

6.4.3. Aussenden des Stimulus und Aufnahme der Reaktion

Der Algorithmus zum Abspielen des Sweep-Structs und Aufnehmen der Systemreaktion ist in Abbildung 6.5 dargestellt. Bedingt durch Szenarien, wie beispielsweise eine hohe Anzahl an Lautsprecherkanälen oder einen zu kleinen Arbeitsspeicher, kann es notwendig sein, die Messungen schrittweise durchzuführen. Hierfür bestimmt der Algorithmus zunächst die Zahl der notwendigen Durchläufe und legt sie in der Variablen *turns* ab. Der folgende Ablauf ist abhängig von dem im *ini*-File definierten Sweep-Typ. Für die Sweep-Typen *specific* und *build*, welche die individuellen Gap-Zeiten und akustischen Laufzeiten der Lautsprecherkanäle berücksichtigen, werden die Sweep-Structs bei jedem Durchlauf dynamisch geladen, während für den Sweep-Typ *files*, welcher nur ein Sweep-Struct vorsieht, dieses vor dem Beginn der Iteration geladen wird. Die anschließende Schleife beginnt mit der Abfrage nach dem Sweep-Typ, dem Aufbau der notwendigen Verbindungen in JACK, dem Aussenden des Sweep-Structs und Aufnehmen der Reaktion, dem Abbau der Verbindungen, dem Speichern der Aufnahme und der Prüfung, ob ein weiterer Durchlauf vonnöten ist. Der Auf- und Abbau der notwendigen JACK-Verbindungen beinhaltet auch die Frage, ob es sich um eine Messung des Lautsprechersystems im H0104 handelt. Ist dies der Fall, wird das Routing der Audiosignale auf den Nodes mit Hilfe von Skripten, die auf Bender liegen, über das Netzwerk gesteuert. Die aquirierten Daten werden in dem Struct »mess« abgelegt und als **_temp_*.mat* auf die Festplatte gespeichert. Das Struct »mess« enthält die in Tabelle 6.6 eingetragenen Felder. Die in Feld *x* gespeicherte Aufnahme wird für die folgende Entfaltung herangezogen, die Daten der Felder *tGap*, *initRuntime*, *channelsPerTurn* und *globalChannelsPerTurn* sind für die spätere Ausfensterung der IRs von Bedeutung und das Feld *channels* für die Zuordnung der IRs zu ihren Kanälen. Die in dem Feld *errorInSweep* abgelegten Informationen sind für die abschließende Evaluation wichtig, während die übrigen Felder allgemeine Fakten zur Messung enthalten.

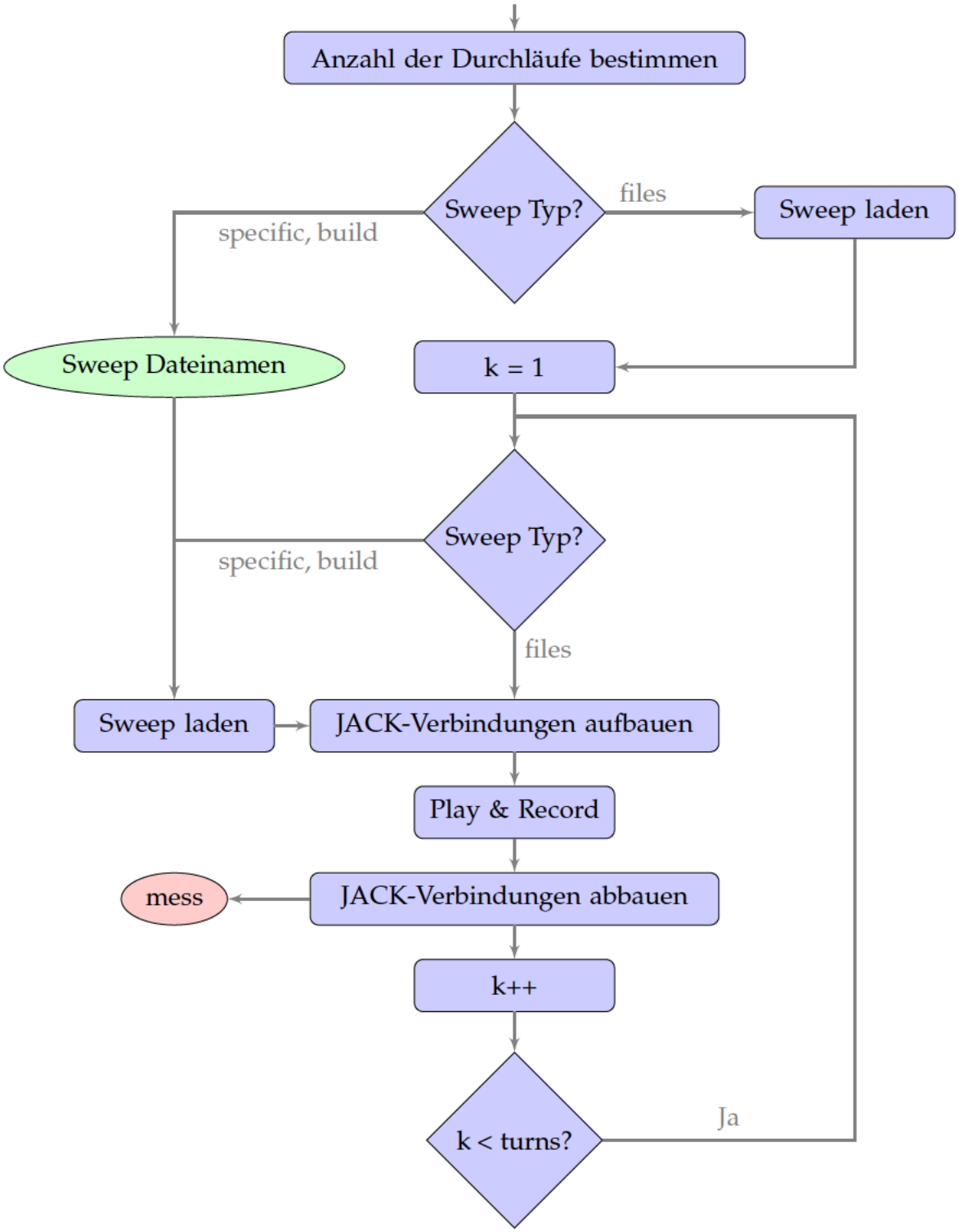


Abbildung 6.5.: Algorithmus zum Abspielen der Sweep-Matrix und Aufnahme der Systemreaktion

Name	Feld	Funktion
mess	x	Aufnahme
	IR	Reserviert für das Ergebnis der Entfaltung
	channels	Kanäle, die in der Aufnahme erfasst wurden
	globalChannelsPerTurn	Anzahl der Kanäle pro Durchgang
	channelsPerTurn	Anzahl der Kanäle für den Fall, dass der letzte Durchlauf weniger Kanäle beinhaltete
	fs	Abtastfrequenz
	tGap	Kanalbezogene Gap-Zeiten
	initRuntime	Akustische Laufzeit des ersten im Sweep-struct vorkommenen Kanals
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar
	errorInSweep	Defektbezeichnungen für Testmessungen

Tabelle 6.6.: Aufbau des Structs »mess«

6.5. Signalverarbeitung in Abhängigkeit von dem Messtyp

Die Signalverarbeitung, die sich anschließt, unterscheidet sich je nachdem, welcher der vier möglichen Messtypen zu Beginn der Messung gewählt wurde.

6.5.1. Entfaltung

Allen Messtypen gemein ist die Durchführung der Entfaltung, wie sie in Kapitel 2.2 beschrieben wurde. Das Ablaufdiagramm der zuständigen Software-Komponente zeigt Abbildung 6.6. Zunächst werden die für die Entfaltung nach Gleichung 2.7 benötigten Daten geladen. Dies ist zum einen das Struct »reference«, gespeichert in der Datei *referenceTemp.mat*, und zum anderen das Struct »mess«, abgelegt in **_*.mat*. Die Längen beider werden durch Zero-Padding, das Anhängen von Nullen, angeglichen und verdoppelt, bevor sie mittels FFT in den Frequenzbereich überführt werden. Nach durchgeführter Entfaltung und Rücktransformation in den Zeitbereich wird der Messtyp abgefragt. Der Typ *pre* bedingt das sequentielle Messen der einzelnen Lautsprecherkanäle, was bedeutet, dass die Entfaltungsprodukte jeweils nur eine IR nebst Klirrkomponenten besitzen. Die Ergebnisse der Entfaltung werden im Struct

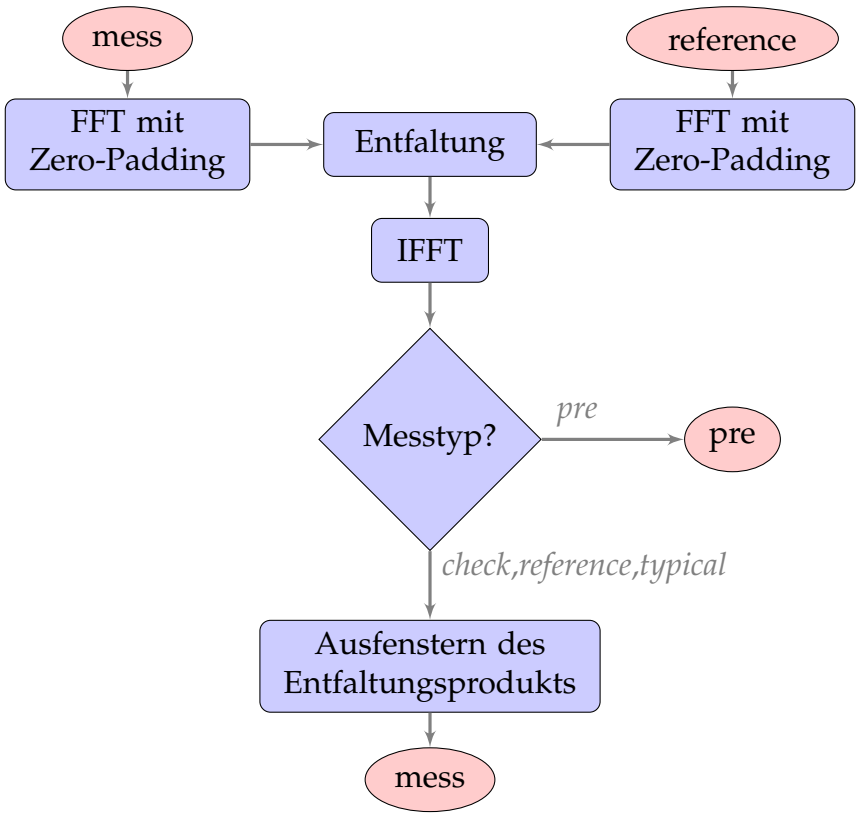


Abbildung 6.6.: Ablauf des Programmteils zur Entfaltung

»pre« unter *_pre_*.mat* gespeichert und für einen sich anschließenden Analyseprozess zur Berechnung der MESM-Parameter und Referenzdaten herangezogen. Die anderen Messtypen können aufgrund der Messungen mit quasi-parallelen Sweeps mehrere IRs in ihren Entfaltungsprodukten enthalten. Bevor sie im Feld *IR* des Structs »mess« gespeichert werden, wird jeweils der Bereich mit den aneinandergereihten Impulsantworten ausgefenstert.

6.5.2. Ausfenstern der Impulsantworten

Nachdem das Struct »mess« geladen wurde, werden die einzelnen Impulsantworten mittels Fensterung extrahiert und den jeweiligen Kanälen zugeordnet. Dies wird mit Rechteckfenstern, welche die im Feld *tGap* des Structs »mess« angegebenen Längen für die entsprechenden Kanäle aufweisen, realisiert. Mittels eines zeitlichen Offsets, definiert im Feld *initRuntime* von »mess«, wird die akustische Laufzeit vor der ersten Impulsantwort herausgenommen. Die Impulsantworten werden in das Struct »IRs« abgelegt, dessen Aufbau in Tabelle 6.7 dargestellt ist. Für den Fall, dass die

Name	Feld	Funktion
IRs	IR	Impulsantwort
	channel	Lautsprecherkanal
	fs	Abtastfrequenz
	maxAmp	Maximale Amplitude
	tGap	Kanalbezogene Gap-Zeiten
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar
	errorInSweep	Defektbezeichnungen für Testmessungen

Tabelle 6.7.: Aufbau des Structs »IRs«

Messung mehrere »mess«-Structs zur Erfassung aller Lautsprecherkanäle benötigt, werden alle ermittelten Impulsantworten in dem Struct »IRs« zusammengefasst. Um den Speicherbedarf des Structs zu kontrollieren, kann im *ini*-File über den Eintrag *memoryThreshold* die Größe von »IRs« beschränkt werden. Größere »IRs« werden dann in mehrere Dateien mit den Namen **_IRs_*.mat* gespeichert. Neben der Extraktion der Impulsantworten wird an dieser Stelle noch das Struct »maxAmp« erstellt, welches Informationen zum Status und der maximalen Amplitude jedes Kanals innehat. Tabelle 6.8 zeigt dessen Aufbau. Zur Bestimmung des Status eines Lautsprecherka-

Name	Feld	Funktion
maxAmp	amp	Maximale Amplitude
	channel	Lautsprecherkanal
	fs	Abtastfrequenz
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar
	errorInSweep	Defektbezeichnungen für Testmessungen

Tabelle 6.8.: Aufbau des Structs »maxAmp« bzw. »refMaxAmp«

nals wird zunächst die Energie-Zeit-Kurve der jeweiligen Impulsantwort berechnet. Danach wird der Abstand des maximalen und minimalen Wertes mit einem Threshold verglichen. Bei Unterschreiten des Schwellenwertes gilt der Kanal als komplett ausgefallen. Das Feld *amp* des Structs »maxAmp« wird auf -1 gesetzt. Anderenfalls enthält es den Betrag der maximalen Amplitude der Impulsantwort. Handelt es sich

um eine Messung des Typs *reference*, wird das Struct »maxAmp« in »refMaxAmp« umbenannt und statt unter **_maxAmp.mat* in **_refMaxAmp.mat* abgespeichert.

6.5.3. Berechnung der Betragsfrequenzgänge

Zur Berechnung der Betragsfrequenzgänge wird zunächst das Struct »IRs« geladen. Die folgende Signalverarbeitung ist abhängig von den Angaben im *ini*-File. Der Eintrag *TFN* bestimmt die FFT-Länge 2^{TFN} und *smooth* die Art der Glättung, wobei Werte verschieden von Null, den Bruchteil einer Oktave angeben. Für *smooth* gleich Null wird keine Glättung ausgeführt. Die berechneten Frequenzgänge werden mittels Bandpass gefiltert und solange nacheinander in ein Struct geschrieben, bis dieses den *memoryThreshold* überschreitet und dadurch eine Speicherung in mehrere Dateien veranlasst wird. Das entstehende Struct »TFs« ist in Tabelle 6.9 skizziert. Der Messtyp

Name	Feld	Funktion
TFs	TF	Betragsfrequenzgang
	channel	Lautsprecherkanal
	fs	Abtastfrequenz
	TFN	FFT-Ordnung
	smooth	Glättungsintervall
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar
	errorInSweep	Defektbezeichnungen für Testmessungen

Tabelle 6.9.: Aufbau des Structs »TFs« bzw. »refTFs«

reference erwirkt ein Umbenennen von »TFs« in »refTFs«. Die Structs werden abschließend in die Dateien **_TFs.mat* bzw. **_refTFs.mat* geschrieben.

6.5.4. Weitere Signalverarbeitung

Für Messungen des Typs *typical* werden die Impulsantworten in 24 Bit **.wav*-files mit einer der Messung entsprechenden Abtastrate gewandelt. Für *check* und *pre* kommt es zu weiteren Verarbeitungsschritten, welche in Kapitel 8 bzw. Kapitel 7 erläutert werden.

6.6. Datei-Management

Die Messsoftware ist modular aufgebaut und speichert die Ergebnisse jedes Arbeitsschrittes separat auf die Festplatte. Kommt es zu einem unerwarteten Abbruch, kann so auf die Zwischenresultate zugegriffen werden. Nachdem die Messung abgeschlossen ist, muss entschieden werden, welche der gespeicherten Daten behalten und welche verworfen werden können. Dies wird je nach Messtyp über den Eintrag *deleteFiles* im *ini*-File bestimmt. Ist *deleteFiles* aktiviert, besteht die Ausgabe des Resultats für Messungen des Typs *check* lediglich aus einem *txt*-File, das Informationen zu identifizierten Defekten beinhaltet. Der Messtyp *typical* liefert die Impulsantworten der einzelnen Kanäle als *wav*-Files und der Typ *reference* die Dateien **_refMaxAmp.mat* und **_refTFs.mat*. Letztere werden auch bei Messungen des Typs *pre* ausgegeben, ebenso wie die Dateien, die Informationen zur Synthese von sich überlappenden Sweeps enthalten. Bei nicht aktiviertem *deleteFiles*, werden alle während der Messung angelegten Ordner und Dateien mit *temp* im Namen nach Abschluss gelöscht.

6.7. Ordnerstruktur

Die Verzeichnissstruktur der Messsoftware stellt Abbildung 6.7 dar. Im Hauptordner *Messsystem* befindet sich neben den Unterordnern *ini_Files* und *components* die Datei *startMeasurement.m*, welche die Messprozedur startet. Der Unterordner *ini_Files* beinhaltet das *ini*-File, in dem die Messeinstellungen getätigt werden können. Im Ordner *components* sind alle Dateien gespeichert, die, ist das System einmal eingerichtet, keiner Bearbeitung mehr bedürfen. Betriebsystemspezifische Dateien finden sich in den Ordnern *_macOs* und *_linux*. Hier sind die für diese Arbeit in C geschriebenen Programme *jack_disconnect_all*, für das schnelle Trennen aller JACK-Verbindungen, und *jack_check*, zur Statusabfrage von JACK, sowie das Playrec-Modul, alle kompiliert für das jeweilige System, abgelegt. Da Einstellungen am Audiointerface HDSPe MADI von RME über das Kommandozeilenprogramm amixer unter GNU/Linux getätigt werden können, werden diese mittels Skripte von der Messsoftware übernommen, welche sich ebenfalls im Ordner *_linux* befinden. Aufgrund des speziellen Aufbaus des WFS-Systems im Raum H0104 ist ein Ordner *_h104* angelegt worden, in dem neben den erwähnten Dateien für GNU/Linux noch *ini*-Files für die Nodes und Bender gespeichert sind. Alle GNU Octave *m*-Files, die entweder für die Messsoftware geschrieben wurden oder nicht Teil der Standarddistribution von GNU Octave sind, lie-

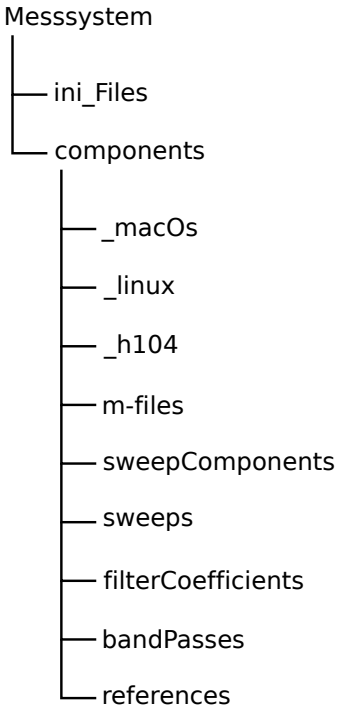


Abbildung 6.7.: Ordnerstrukur der Messsoftware

gen im Ordner *m-files*. Messungen mit der MESM-Methode fordern eine auf die Messumgebung angepasste Matrix von Sweeps. Diese ist abhängig von der Anzahl der Lautsprecherkanäle, den Gap-Zeiten, die sich aus den Nachhallzeiten und den zeitlichen Ausdehnungen der Klirrkomponenten zusammensetzen, und den akustischen Laufzeiten zwischen den Lautsprechern und dem Mikrofon. Bei der Einrichtung des Messsystems wird die Matrix, die alle Lautsprecherkanäle einschließt, in dem Ordner *sweeps* gespeichert. Soll nur ein Teil der Kanäle getestet werden, wird vor der Messung dynamisch eine Sweep-Matrix erstellt. Die Bausteine hierfür sind ein einkanaliger Sweep sowie die Gap- und akustischen Laufzeiten. Diese Komponenten liegen im Ordner *sweepComponents*. Es existieren Sweeps für verschiedene Abtastfrequenzen und Ordnungen. Das Filter-Set, das die Filterkoeffizienten aller zu erkennenden Defekte einschließt, befindet sich in dem Ordner *filterCoefficients*. Aus Gründen der doch vorhandenen Unterschiede zwischen GNU Octave und MATLAB® wurde auf die dynamische Generierung von benötigten Bandpässen verzichtet. Stattdessen liegen im Ordner *bandPasses* erstellte Frequenzgänge von Bandpässen für verschiedene Abtastfrequenzen und FFT-Ordnungen. Die Grundlage des Detektionsalgorithmus bilden Referenzfrequenzgänge aller Lautsprecherkanäle eines Systems in einwand-

freiem Zustand. Bei der Einrichtung des Messsystems werden sie erfasst und in dem Ordner *references* abgelegt.

7. Akustische Referenzmessung

Messungen mit Hilfe der Overlapping-Methode erfordern eine akustische Referenzmessung, durch die alle erforderlichen Daten gewonnen werden, um die Gap-Zeiten des Sweep-Arrays zu berechnen. Es werden die Nachhallzeiten und die zeitliche Ausdehnung der Verzerrungsprodukte benötigt. Bei einer Messsituation mit unterschiedlichen akustischen Laufzeiten zwischen den Lautsprechern und dem Messmikrofon müssen außerdem die Laufzeiten detektiert werden, um eine Laufzeitkompensation durchführen zu können.

Die Referenzmessung wird kanalweise vorgenommen und dient auch zur Akquise der Frequenzgänge und Maximalamplituden der einzelnen Kanäle, mit denen die Ergebnisse späterer Messungen verglichen werden. Um eine solche Messung zu bewerkstelligen, muss in dem *ini*-File der Messtyp *pre* eingetragen werden. Ein Zwischenergebnis der folgenden Messung sind die Dateien **_pre_*.mat*, in welchen jeweils das Entfaltungsprodukt eines Lautsprecherkanals gespeichert ist. Aus diesen Entfaltungsprodukten werden alle benötigten Parameter berechnet. Abbildung 7.1 zeigt den Aufbau des Analyseteils der akustischen Referenzmessung.

7.1. Gate

Zunächst werden die Entfaltungsprodukte daraufhin geprüft, ob ein Lautsprecherkanal Schall emittiert oder nicht. Dies geschieht über eine partielle RMS-Berechnung. Die Differenz von maximalem und minimalem RMS-Pegel wird mit einem Threshold, der mittels des Eintrages *detection* $\rightarrow$ *gateThreshold* im *ini*-File definiert wird, verglichen. Bei dessen Überschreitung spielt der entsprechende Kanal bei den anknüpfenden Kalkulationen keine Rolle mehr.

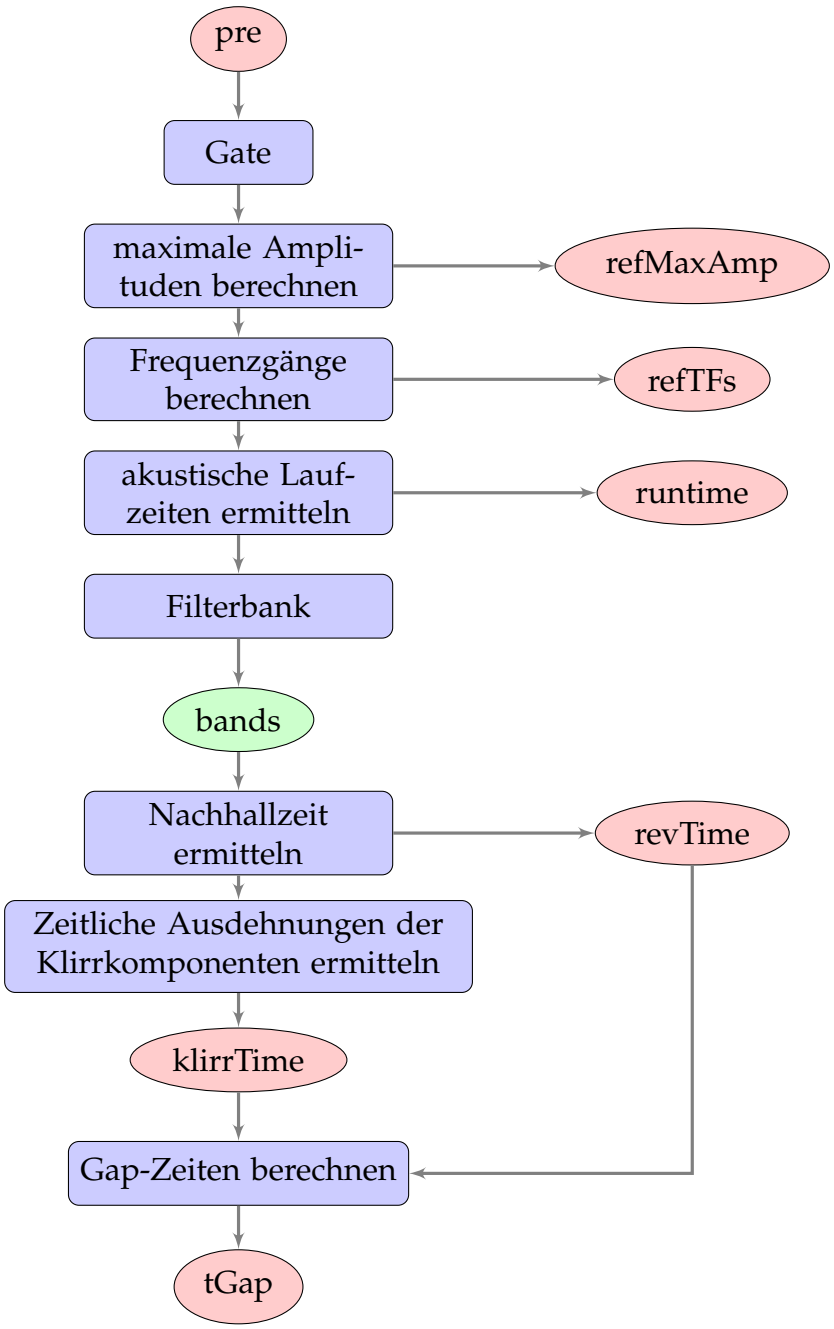


Abbildung 7.1.: Ablauf des Analyseteils der akustischen Referenzmessung

7.2. Berechnung der Betragsfrequenzgänge und Maximalamplituden der Impulsantworten

In den nächsten Arbeitsschritten werden die Structs »refMaxAmp« und »refTFs« ermittelt und gespeichert. Abbildung 7.2 stellt die maximalen Pegel aller Lautsprecher-

kanäle im Raum H0104 dar. Hierfür wurde das Gate deaktiviert. Es ist gut zu erken-

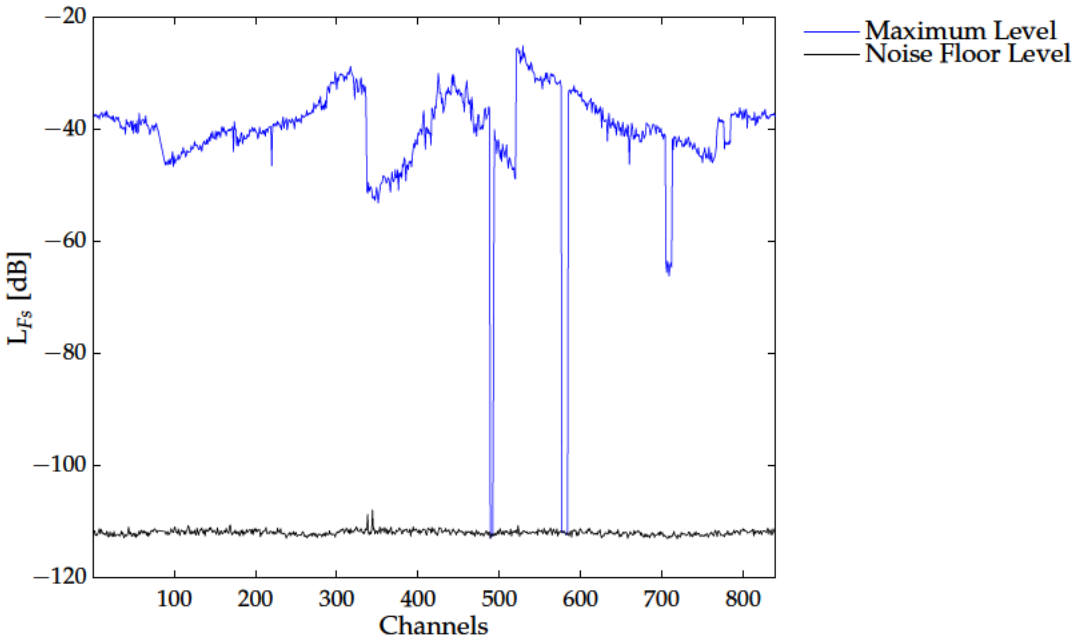


Abbildung 7.2.: Maximal- und Grundgeräusch-Pegel aller Kanäle im Raum H0104

nen, auf welche Weise die Position des Mikrofons Einfluss auf den Verlauf nimmt. Durch die Montage des Mikrofons in mittlerer Position an der Rückwand des Raumes kommt es zu Unstetigkeiten im Bereich der Lautsprecherkanäle 520 und 340, die sich nahe an den Ecken von Rückwand und Seitenwänden befinden. Von dort aus steigt der Pegel bis etwa Kanal 440 an, auf dessen Höhe das Mikrofon angebracht ist. In der Richtcharakteristik des Mikrofons findet sich der Grund, warum das Maximum des Pegelverlaufs nicht an dieser Stelle ist. Des Weiteren sind an anhand des Verlaufs auch Kanalausfälle und Defekte einzelner Lautsprechertreiber zu erkennen. Das Lautsprechermodul mit den Kanälen 577 bis 584 sowie die vier Kanäle 489 bis 493 eines Moduls sind stumm. Darüber hinaus sind Ausfälle von einzelnen Lautsprecher-Wegen bei dem Modul mit den Kanälen 705 bis 712 zu verzeichnen.

Mittels FFT werden die Betragsfrequenzgänge bestimmt. Der FFT-Grad richtet sich nach dem Eintrag *detection* → *TFN* im *ini*-File. Soll eine Glättung absolviert werden, ergibt sich deren Schrittgröße aus der Angabe des Bruchteils einer Oktave im Feld *smooth*. Die Frequenzgänge werden in dem Struct »refTFs« festgehalten. Die Structs »refTFs« und »refMaxAmp« werden als **_refTFs.mat* respektive **_refMaxAmp.mat* gespeichert.

7.3. Akustische Laufzeiten

Schallwellen benötigen Zeit, um eine gewisse Distanz zurückzulegen. Bei der Messung eines Lautsprechers mit einem Mikrofon ist diese Zeit als akustische Laufzeit vor der IR zu erkennen. Die Länge der Laufzeit hängt von der Distanz zwischen Lautsprecher und Mikrofon sowie den Eigenschaften des Umgebungsmediums ab.

7.3.1. Detektion akustischer Laufzeiten

Die Laufzeit einer IR kann nicht über deren maximalen Pegel erfolgen, da der Direktschall, welcher den Beginn einer IR darstellt, nicht zwingend die Komponente mit dem höchsten Pegel ist. Abbildung 7.3 zeigt einen solchen Fall. Die gemessene RIR

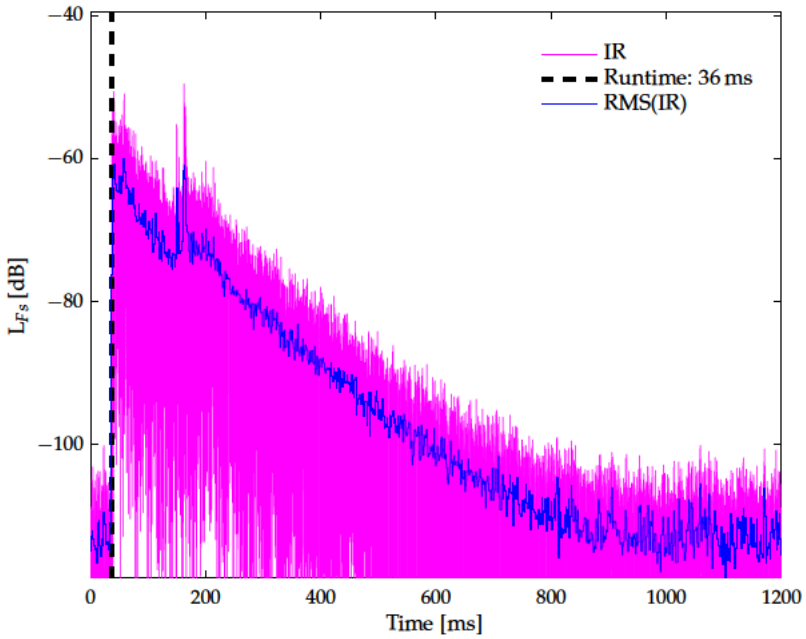


Abbildung 7.3.: Laufzeitdetektion für Kanal 355 der WFS-Anlage im H0104

des Kanals 355 der WFS-Anlage im Raum H0104 hat ihr Maximum -49,6 dB bei etwa 160 ms. Die Verwendung des höchsten Pegels zur Detektion würde hier eine Laufzeit von 160 ms ergeben und etwa 125 ms der IR ignorieren. Bei genauer Betrachtung, siehe Abbildung 7.4, ist aber selbst der Peak mit dem zweithöchsten Pegel von 51 dB nicht der Anfang der IR. Die Methode, die für diese Arbeit herangezogen wurde, wird in [Defrance u. a., 2008] erläutert und dort als *mean over time* bezeichnet. Zunächst wird die RMS-Funktion $E(n)$ der IR mit einer gewissen Fensterbreite gebildet.

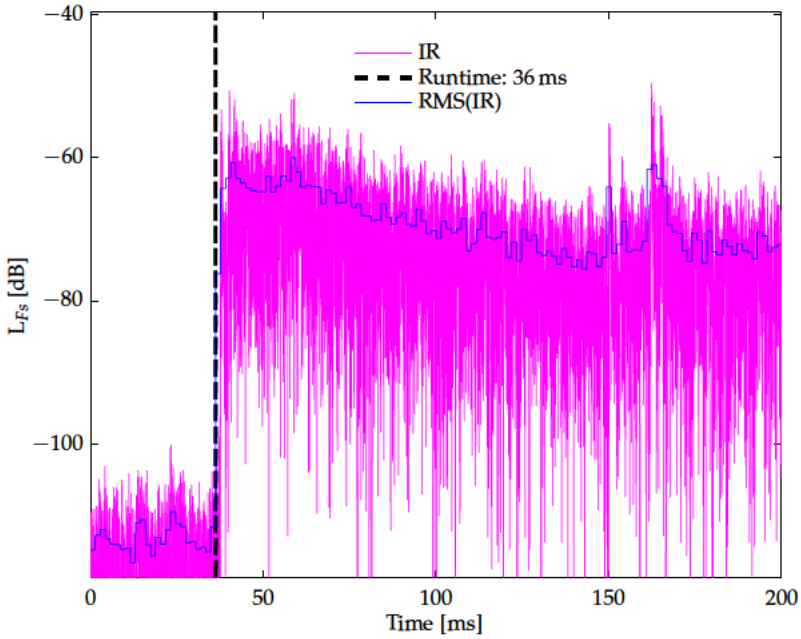


Abbildung 7.4.: Vergrößerung des Ausschnitts aus Abbildung 7.3

Diese hängt von der gewünschten zeitlichen Auflösung ab und kann beliebig gesetzt werden. Nun wird der Index des höchsten Pegels der geglätteten Funktion ermittelt. Dieser begrenzt den Bereich der folgenden Analyse von $E(n)$. Der Algorithmus durchläuft $E(n)$ mit einer Schrittweite, die der Fensterbreite der RMS-Funktion entspricht, in zeitlich positiver Richtung und bildet das Verhältnis des aktuellen und des vorherigen Wertes von $E(n)$. Der Index des größten Quotienten wird als Onset der IR und damit der Laufzeit ausgegeben. Das Ergebnis dieses Detektions-Algorithmus ist als gestrichelte Linie in den Abbildungen 7.3 und 7.4 dargestellt. Die Fensterbreite der RMS-Funktion beträgt hier 64 samples, die erkannte Laufzeit 36 ms. Die Resultate der Laufzeitanalyse werden in das Struct »runtime« geschrieben und in der entsprechenden Datei *_*runtime.mat* gespeichert. Die Felder des Structs sind in Tabelle 7.1 aufgelistet.

Abbildung 7.5 zeigt die Laufzeiten zwischen den einzelnen Lautsprecherkanälen der WFS-Anlage im Raum H0104 und dem Raummikrofon. Es ist gut zu erkennen, dass das Mikrofon mittig an der Raumrückwand montiert ist. Die Kanäle in diesem Bereich weisen die niedrigsten Laufzeiten auf. Die Laufzeitenkurve verläuft symmetrisch. Die höchsten Laufzeiten resultieren aus der Distanz zwischen dem Mikrofon und den Lautsprechermodulen im vorderen Bereich des Raumes. Auffällig ist der

Name	Feld	Funktion
runtime	seconds	Akustische Laufzeit in Sekunden
	channel	Lautsprecherkanal
	fs	Abtastrate
	room	Ortsangabe
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar

Tabelle 7.1.: Aufbau des Structs »runtime«

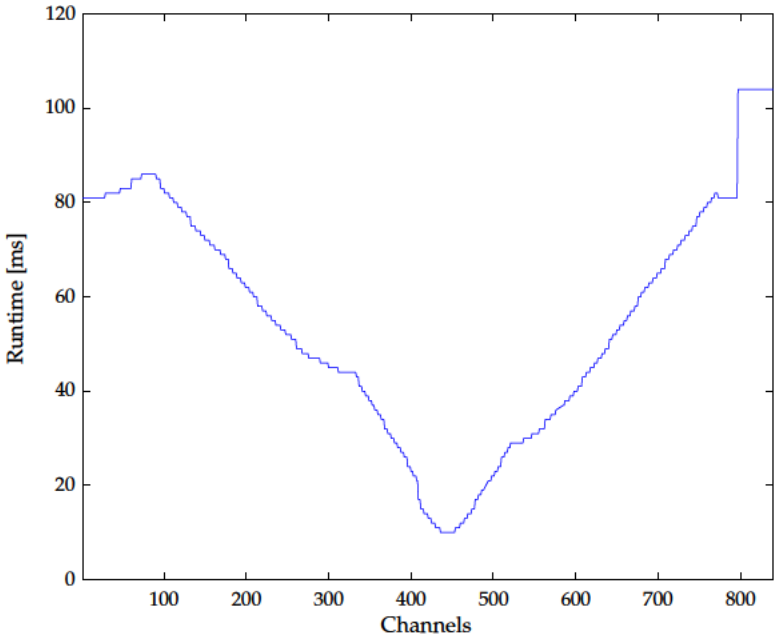


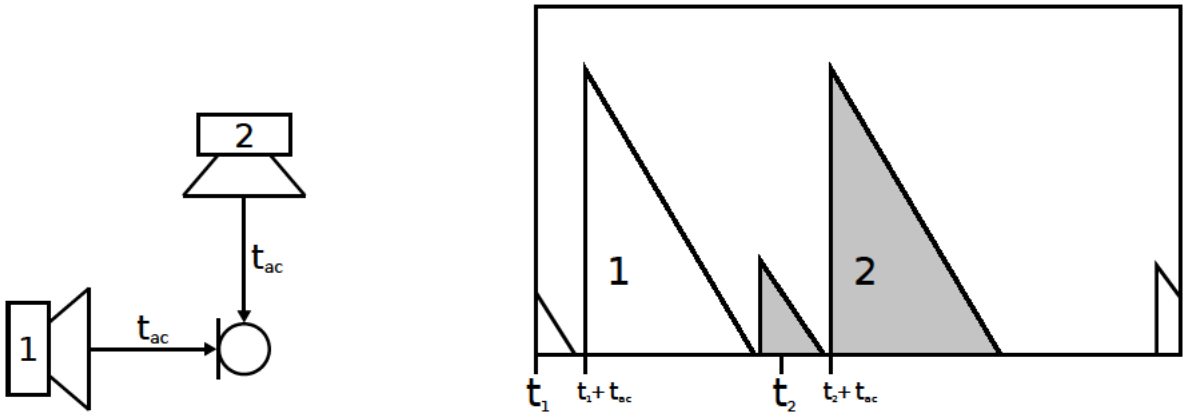
Abbildung 7.5.: Laufzeiten zwischen den einzelnen Kanälen der WFS-Anlage im H0104 und dem Raummikrofon

Sprung von 81 ms bei Kanal 796 auf 104 ms bei den folgenden Kanälen, der im Gegensatz zu den Laufzeiten für die ersten 100 Kanäle, die erwartungsgemäß zu den Ecken des Raumes hin zunehmen, steht. Eine Erklärung hierfür könnte sein, dass die DSP-Konfigurationen der Kanäle 797 bis 840 von jenen der übrigen Kanäle abweichen.

7.3.2. Overlapping vs. akustische Laufzeiten

Eine Messsituation mit unterschiedlichen akustischen Laufzeiten, wie sie hier vorliegt, hat Konsequenzen für die Implementierung der Overlapping-Methode in das Messsystem.

Das in Kapitel 2.3 beschriebene Overlapping-Verfahren setzt eine äquidistante Verteilung der Lautsprecher um das Mikrofon voraus, wie sie in Abbildung 7.6a zu sehen ist. Die akustischen Laufzeiten t_{ac} zwischen den Lautsprechern und dem Mikrofon sind identisch. Aus der quasisimultanen Messung mit den Anregungszeitpunkten t_1 und t_2 nach Gleichung 2.15 resultiert das in Abbildung 7.6b schematisch dargestellte Entfaltungsprodukt. Die LIR beider Systeme nebst ihren Klirrkomponenten sind um den Wert t_{ac} verschoben und können mittels Ausfensterung extrahiert werden.



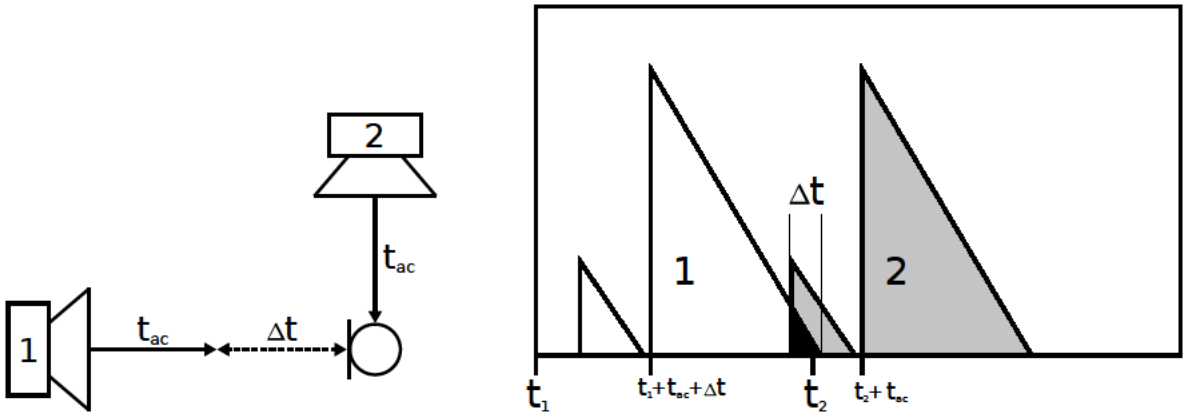
(a) Messsituation (b) resultierendes Entfaltungsprodukt

Abbildung 7.6.: Zwei Systeme mit identischen akustischen Laufzeiten

Liegt eine ungleichmäßige Verteilung der Lautsprecher vor, kommt es bei dem dazugehörigen Entfaltungsprodukt je nach Laufzeitdifferenzen zu Überlappungen von IRs bzw. freien Stellen zwischen ihnen. In Abbildung 7.7a ist eine Anordung dargestellt, bei der das System 1 eine um Δt längere Laufzeit als System 2 aufweist. Bei dem Entfaltungsprodukt gemäß Abbildung 7.7b kommt es zu einer Überlappung der LIR des Systems 1 und der zweiten HIR des Systems 2. Die zeitliche Überlagerung entspricht Δt.

7.3.3. Laufzeitkompensation und ihre Implementierung

Um die Überdeckung zweier oder mehrerer IRs zu verhindern, müssen die Anregungszeitpunkte an die jeweiligen akustischen Laufzeiten angepasst werden. In der



(a) Messsituation (b) resultierendes Entfaltungsprodukt

Abbildung 7.7.: Zwei Systeme mit unterschiedlichen akustischen Laufzeiten

durch die Abbildungen 7.7a und 7.7b illustrierten Messanordnung müsste zum Beispiel System 2 zum Zeitpunkt $t_2 + \Delta t$ angeregt werden.

Die Laufzeit des ersten Systems einer arbiträren Anordnung von Systemen kann nicht kompensiert werden, da dies eine Anregung des Systems zu einem negativen Zeitpunkt bedeuten würde. In [Giese, 2009, Seite 80] wird die Gleichung

$$t_{c_i} = t_i + t_{diff_i} = t_i + (t_{ac_1} - t_{ac_i}) \tag{7.1}$$

angegeben, mit der die laufzeitkompensierten Anregungszeitpunkte t_{c_i} aller Systeme berechnet werden können. Wie groß die jeweilige zeitliche Korrektur t_{diff_i} ausfällt, wird durch die Differenz der akustischen Laufzeiten des ersten und des zu berechnenden Systems festgelegt.

Für die Anwendung von Gleichung 7.1 wird die Kenntnis aller Systemlaufzeiten vorausgesetzt. In einer ersten Messung werden diese ermittelt und gespeichert. Sie können dann dem Programmmodul zur Synthese von sweep-Arrays übergeben werden. Dieses berechnet zunächst die Anregungszeitpunkte unter Berücksichtigung der frequenzabhängigen Werte der Nachhallzeiten und zeitlichen Ausdehnungen der Klirrkomponenten nach Gleichung 2.14. Es folgt die Korrektur der Anregungszeitpunkte mit Hilfe von Gleichung 7.1.

Evaluation der implementierten Laufzeitkompensation

Zur Veranschaulichung und Kontrolle der implementierten Laufzeitkompensation wird eine Messsituation mit vier Quellen an unterschiedlichen Positionen simuliert.

Die Messsoftware wird den zu messenden Kanälen entsprechend rechnerintern mit einer Sequenzersoftware verbunden. Dort wird für jeden Eingangskanal eine Spur mit einem einfachen Delay-Plugin angelegt, mit welchem die Laufzeit eingestellt werden kann. Die Ausgangssignale der Spuren werden aufsummiert und nachdem sie ein Hall-Plugin durchlaufen haben, zur Messsoftware zurückgeführt.

Es werden drei Messungen vorgenommen. Die erste liefert die benötigten Daten für die laufzeitkompensierte Messung mittels Overlapping. Sie sind in Tabelle 7.2 eingetragen. Es folgt eine Messung qua Overlapping mit den Anregungszeitpunkten

System	1	2	3	4
t_{rev}	100	100	100	100
t_{klirr}	0	0	0	0
t_i	0	100	200	300
t_{diff_i}	0	-91	-31	-51
t_{c_i}	0	9	169	249
t_{ac_i}	15	106	46	66

Tabelle 7.2.: Daten der Messung in [ms]

t_i. Diese sind für Quellen in gleicher Entfernung zum Messmikrofon adäquat. Das Ergebnis der Entfaltung veranschaulicht Abbildung 7.8. Zwischen System 1 und 2 existiert eine zeitliche Lücke von 91 ms. Die Systeme 3 und 4 liegen 20 ms auseinander, während sich die IRs der Systeme 2 und 3 um 60 ms überlagern und nicht weiter verwendbar sind. Die dritte Messung bedient sich der laufzeitkompensierten Anregungszeitpunkte t_{ac_i}. Das Entfaltungsprodukt zeigt Abbildung 7.9. Im Gegensatz zur Abbildung 7.8 sind die Abstände zwischen den einzelnen Systemen gleich den jeweiligen Summen aus Nachhallzeit t_{rev} und zeitlicher Ausdehnung der Klirrkomponenten t_{klirr}, was in diesem Fall eine konstante Distanz von 100 ms bedeutet. Die Zeitspanne zwischen der Position der IR des Systems i und der des Systems 1 ist exakt so groß wie der Wert der nicht laufzeitkompensierten Anregungszeit t_i des betrachteten Systems. Der zeitliche Offset um t_{ac_1} bleibt bestehen und kann, wie zuvor beschrieben, nicht kompensiert werden. Da sämtliche IRs autark voneinander sind, können sie für folgende Signalverarbeitungsprozesse herangezogen werden.

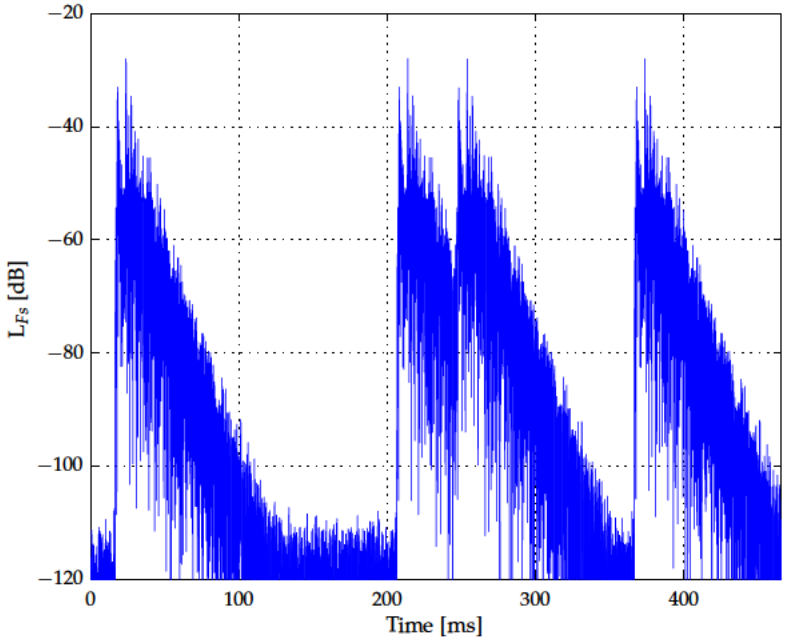


Abbildung 7.8.: Entfaltungsprodukt einer Messung mittels MESM bei arbiträrer Quellenanordnung

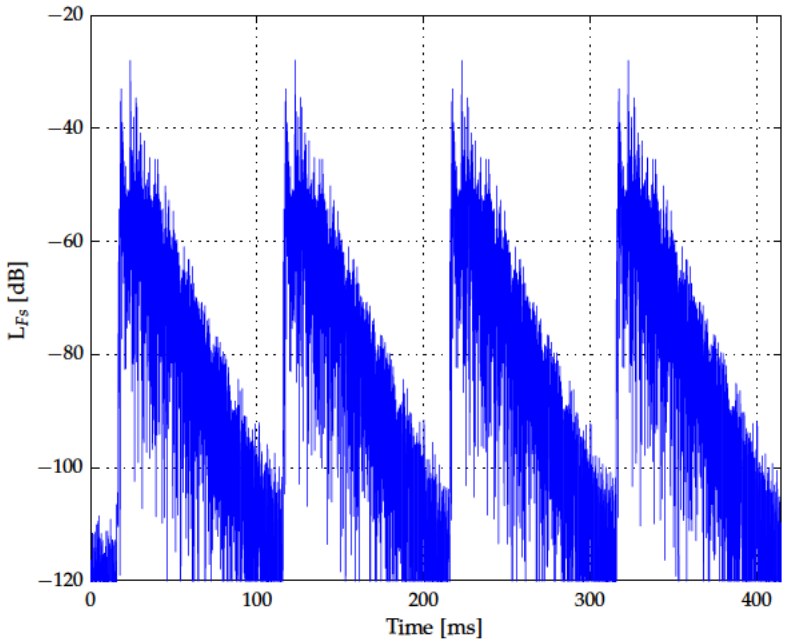


Abbildung 7.9.: Entfaltungsprodukt einer laufzeitkompensierten Messung mittels Overlapping bei arbiträrer Quellenanordnung

7.4. Filterbank

Die Bestimmung der Nachhallzeit und der zeitlichen Ausdehnung der Klirrkomponenten soll frequenzabhängig erfolgen. Hierfür wird eine Terz- bzw. Oktavfilterbank implementiert, die MATLAB®-Funktionen von Christophe Couvreur enthält. Die digitalen IIR-Filter folgen dem Standard ANSI S1.1-1986 (ASA 65-1986). Abbildung 7.10 zeigt die Frequenzbänder der Oktavfilterbank. Die gefilterten Entfaltungsprodukte

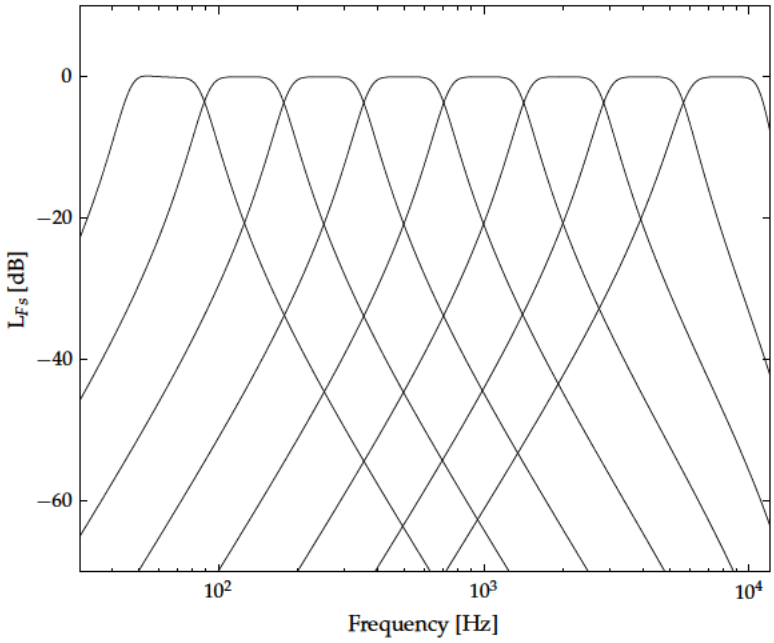


Abbildung 7.10.: Frequenzbänder der Oktavfilterbank

werden in das Struct »bands« geschrieben, welches im weiteren Verlauf der Berechnungen zum Zuge kommt. Im Gegensatz zu den anderen Structs, die während des Analyseteils erzeugt werden, wird es nicht auf die Festplatte gespeichert.

7.5. Bestimmung der Nachhallzeit, des SNR und des Grundgeräuschpegels

Zur Berechnung der Gap-Zeiten des Sweep-Arrays müssen die Nachhallzeiten aus den IRs aller interessierenden Kanäle eruiert werden. Die hier vorgestellte Software bietet zwei Methoden, die Nachhallzeit automatisch zu berechnen. Sie unterscheiden sich, neben der verwendeten Algorithmen, in ihrer Definition der Nachhallzeit.

7.5.1. Die Nachhallzeit als Schnittpunkt von Nachhallkurve und Grundgeräuschpegel

Zur Bestimmung dieses Zeitpunktes wurde die Methode nach Lundeby gewählt, wie sie in [Karjalainen u. a., 2002] beschrieben ist. Es handelt sich um ein iteratives Verfahren, das sich in folgende Arbeitsschritte gliedern lässt:

1. Die quadratische IR wird über Zeitintervalle im Bereich von 10 - 50 ms gemittelt.
2. Der Grundgeräuschpegel wird aus den letzten 10% der IR geschätzt.
3. Die Nachhallkurve wird mittels linearer Regression dem Amplitudenverlauf in dem Zeitraum zwischen dem Maximum der IR und 5 dB über dem zuvor bestimmten Grundgeräuschpegel angenähert.
4. Ermittlung des Schnittpunktes der Reggressionsgeraden und des geschätzten Pegels des Grundrauschens.
5. Berechnung des Grundgeräuschpegels aus dem Bereich zwischen 5 dB-Abfall der Regressionsgeraden nach dem Schnittpunkt und dem Ende der IR. Wenn dieser Zeitraum kleiner als 10% der Länge der IR ist, werden die letzten 10% der IR benutzt.
6. Lineare Regression im Dynamikbereich von 20 dB, beginnend ab 5 dB über dem Grundgeräuschpegel.
7. Erneute Kalkulation des Schnittpunktes von geschätzter Nachhallkurve und Grundrauschen.

Die letzten drei Schritte werden so oft wiederholt, bis sich der Pegel des Grundrauschens nicht mehr verändert. Ist dies nach fünf Iterationen nicht der Fall, wird die Berechnung abgebrochen. Der auf solche Weise implementierte Algorithmus liefert neben der Nachhallzeit auch den Rauschpegel und den SNR einer gemessenen IR. Als Erweiterung zum vorgestellten Algorithmus besteht die Möglichkeit, einen Offset-Pegel auf den ermittelten Grundgeräuschpegel zu addieren. Die Nachhallzeit ist dann der Schnittpunkt des so entstandenen Pegels und des Verlaufs der quadratischen Mittelwerte der IR. In Abbildung 7.11 sind die Ergebnisse des Algorithmus für die IR des Kanals 400 des WFS-Systems im Raum H0104 dargestellt. Der hier durch eine grüne Linie repräsentierte *Noise Floor*-Pegel setzt sich aus Grundgeräuschpegel und Offset-Pegel zusammen, weshalb letzterer, um das Resultat der

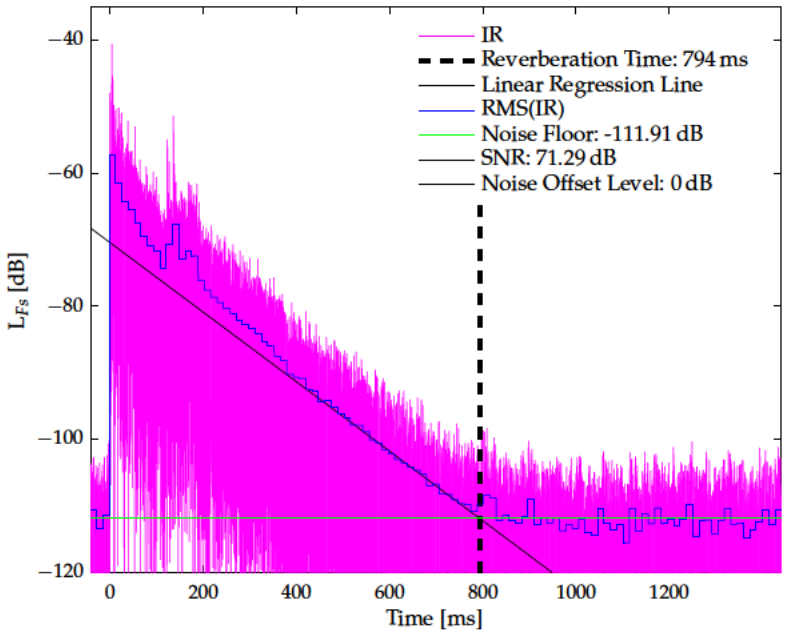


Abbildung 7.11.: Berechnung der Nachhallzeit nach der Lundeby-Methode

Grundgeräuschpegel-Detektion zu veranschaulichen, bei der Berechnung dieser Grafik auf 0 dB gesetzt wurde.

7.5.2. Die Nachhallzeit als Schnittpunkt von Nachhallkurve und vorgegebenem Schwellenwert

Dieser Algorithmus ermöglicht es, die Nachhallzeit über einen Threshold in dB, welcher sich auf das Maximum der IR bezieht, zu bestimmen. Der Zeitpunkt, zu dem die quadratisch gemittelte IR gleich diesem Schwellenwert ist, wird als Nachhallzeit ausgegeben. Abbildung 7.12 zeigt die Ergebnisse für Kanal 400. Die Berechnung des SNR über den Grundgeräuschpegel und den Spitzenpegel der IR erfolgt über die zuvor erläuterte Methode nach Lundeby. Die Abweichung beim SNR um ein hundertstel dB lässt sich mit unterschiedlichen Größen der Mittelungsintervalle erklären.

7.5.3. Aufbau des Structs »revTime«

Die ermittelten Nachhallzeiten werden in dem Struct »revTime« erfasst und als Datei **_revTime.mat* gespeichert. Tabelle 7.3 zeigt den Aufbau des Structs. Neben den für die weiteren Schritte der Signalverarbeitung relevanten Informationen, welche

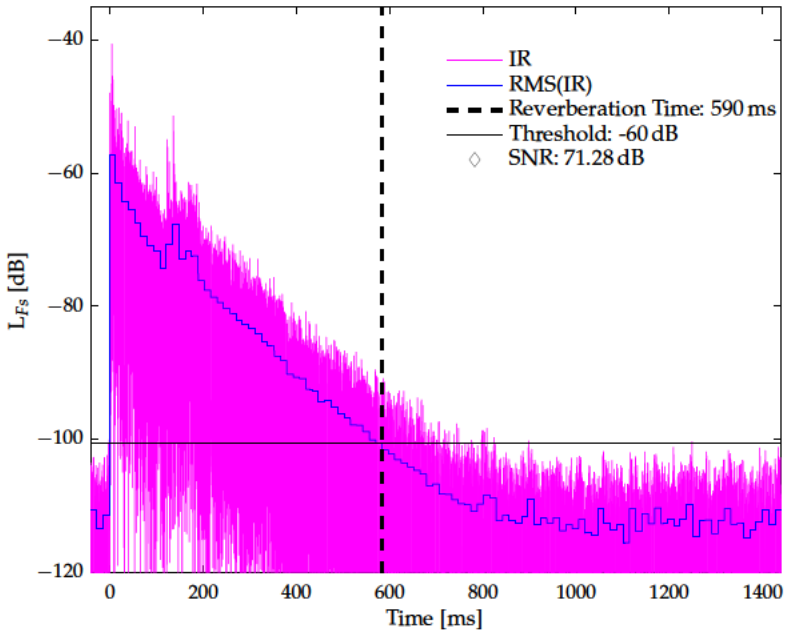


Abbildung 7.12.: Berechnung der Nachhallzeit mittels definiertem Threshold

Name	Feld	Funktion
revTime	seconds	Nachhallzeiten
	channel	Lautsprecherkanal
	fc	Mittenfrequenzen der Frequenzbänder
	fs	Abtastrate
	noiseRMS	Mittlerer Pegel der Umgebungsgeräusche
	peakValue	Maximalamplitude
	SNR	Signal-Rausch-Abstand
	room	Ortsangabe
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar

Tabelle 7.3.: Aufbau des Structs »revTime«

Nachhallzeiten in welchen Frequenzbändern für welchen Lautsprecherkanal vorliegen, werden weitere akustische Kenngrößen, die während der Berechnung bestimmt wurden, sowie Ort, Datum und Beschreibung der Messung gespeichert.

Einen Überblick über die frequenzunabhängigen Nachhallzeiten aller Lautsprecherkanäle im Raum H0104 liefert Abbildung 7.13. Hierfür wurden die Nachhallzeiten mit der Lundeby-Methode berechnet.

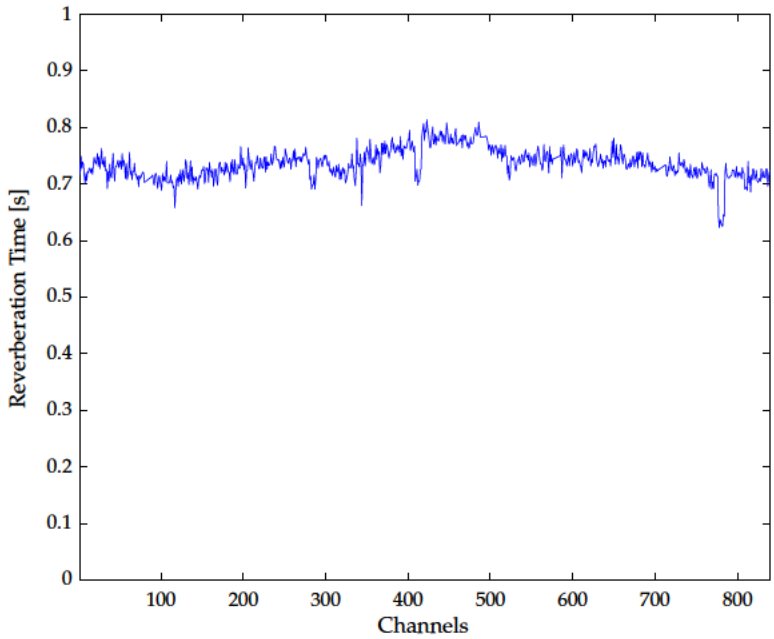


Abbildung 7.13.: Nachhallzeiten aller Lautsprecherkanäle im Raum H0104

7.6. Bestimmung der zeitlichen Ausdehnung der Verzerrungsprodukte

Nichtlineare Verzerrungen, die zum Beispiel durch Übersteuerung der Messkette entstehen, haben Einfluss auf die Gap-Zeiten des Sweep-Arrays bei der Messung mittels MESM. Die Methode, welche zur automatischen Detektion des Zeitraumes aller Verzerrungskomponenten herangezogen wird, soll im Folgenden erläutert werden.

Zunächst wird ein Maskierungskriterium festgelegt, welches den Pegel beschreibt, ab dem die Klirrkomponenten nicht mehr in die Rechnung einbezogen werden. Dieser Schwellenwert kann auf zwei Arten definiert werden. Zum einen kann er sich aus dem Grundgeräuschpegel und einem frei wählbaren Offset-Pegel zusammensetzen. Zum anderen kann er ein manuell einzustellender threshold unter dem Maximum der IR sein. Die IR wird in Zeitintervalle mit der Größe einer definierten Anzahl von

Samples unterteilt. Aus diesen Zeitfenstern werden quadratische Mittelwerte gebildet. Hiernach wird der Zeitpunkt des LIR-Anfangs bestimmt. Er definiert den zeitlichen Nullpunkt. Von diesem ausgehend wird zuletzt der größtmögliche Abstand in negativer zeitlicher Richtung detektiert, zu dem die IR das Maskierungskriterium erfüllt. In Abbildung 7.14 ist das Resultat der Berechnung für den Kanal 426 zu sehen. Der Schwellenwert liegt 6 dB über dem Pegel des Grundrauschens. Die Länge der

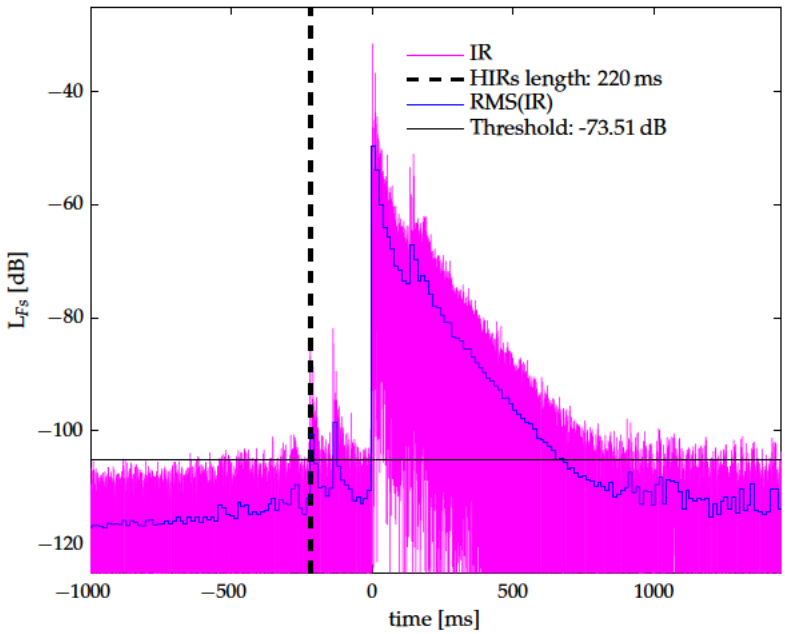


Abbildung 7.14.: Bestimmung der zeitlichen Länge der Verzerrungsprodukte einer IR

harmonischen Verzerrungen beträgt 220 ms. Zum Abschluss werden die detektierten Werte im Struct »distTime« abgelegt und unter dem Namen **_distTime.mat* gespeichert. Die Gliederung des Structs zeigt Tabelle 7.4.

7.7. Bestimmung der Gap-Zeiten

Die Berechnung der in Kapitel 2.2.2 dargelegten Gap-Zeiten ist neben der Ordnung und der Abtastrate des zu erzeugenden Sweeps abhängig von der Summe der frequenzabhängigen zeitlichen Ausdehnung des Nachhalls und der Klirrprodukte. Sie stehen als »revTime« und »distTime« zur Verfügung. Die Gleichung

$$T_{Gap} = max(T(f(t)) - (T_S - t)) \,, \qquad 0 < t \leq T_S \tag{7.2}$$

Name	Feld	Funktion
distTime	seconds	Zeitliche Ausdehnung der Klirrkomponenten
	channel	Lautsprecherkanal
	fc	Mittenfrequenzen der Frequenzbänder
	fs	Abtastrate
	rmsNoise	Mittlerer Pegel der Umgebungsgeräusche
	room	Ortsangabe
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar

Tabelle 7.4.: Aufbau des Structs »distTime«

mit den Variablen $T(f)$ für die Summe von Nachhall und Klirrkomponenten sowie T_S für die Dauer des Sweeps erlaubt die Bestimmung der Gap-Zeit T_{Gap}. Die Gap-Zeiten werden in dem Struct »tGap«, dessen Aufbau Tabelle 7.5 zeigt, abgelegt und unter **_tGap.mat* abgespeichert.

Name	Feld	Funktion
tGap	seconds	Akustische Laufzeit in Sekunden
	channel	Lautsprecherkanal
	fs	Abtastrate
	room	Ortsangabe
	day	Tag
	time	Uhrzeit
	name	Name der Messung
	comment	Kommentar

Tabelle 7.5.: Aufbau des Structs »tGap«

8. Algorithmus zur Detektion der möglichen Defekte

Die Detektion von defekten Lautsprechern oder von Kombinationen von Lautsprecherausfällen eines Kanals geschieht über den Vergleich der gemessenen Betragsfrequenzgänge $|H|_{dB}$ mit ihren Referenzbetragsfrequenzgängen $|H_{Ref}|_{dB}$, welche der $|H|_{dB}$ des Kanals im funktionstüchtigen Zustand entsprechen. Um Ausfälle kompletter Kanäle zu erkennen und eine Pegelanpassung durchzuführen, werden zudem die maximalen Amplituden A_{Ref}, der IRs als Referenz herangezogen. Das Referenzpaar ist in den Dateien *_*refTFs.mat*, für die Frequenzgänge, und *_*refMaxAmp.mat*, für die maximalen Amplituden aller Lautsprecherkanäle, gespeichert. Beide werden entweder während der akustischen Referenzmessung zu Beginn der Inbetriebnahme des Messsystems ermittelt und gespeichert oder in einem separaten Messvorgang erfasst. Die Daten der zu untersuchenden Messung sind in den Dateien *_*TFs.mat* und *_*maxAmp.mat* auf der Festplatte abgelegt. Die Parameter des Detektionsalgorithmus werden zusammen mit anderen messrelevanten Einstellungen in der Datei *measurement.ini* zu Beginn einer Messung definiert und stehen, gespeichert in der Struktur setup, während des gesamten Programmablaufs als globale Variable allen Funktionen zur Verfügung.

8.1. Der Algorithmus im Überblick

Eine Visualisierung der Funktionsweise des Detektionsalgorithmus bietet Abbildung 8.1. Nachdem die Messroutine die IRs der zu überprüfenden Lautsprecherkanäle festgestellt hat, werden sie mit Hilfe der Structs »refMaxAmp« und »maxAmp« im nächsten Arbeitsschritt normalisiert, ihre $|H|_{dB}$ berechnet und diese im Struct »TFs« als *_*TFs.mat* gespeichert. Daraufhin wird das Struct »analyse« erzeugt, in dem zu jedem Lautsprecherkanal Informationen über Pegeldifferenz und identifizierte Defekte abgelegt werden, und die Datei *_*analyse.mat* auf die Festplatte geschrieben. Aus dem

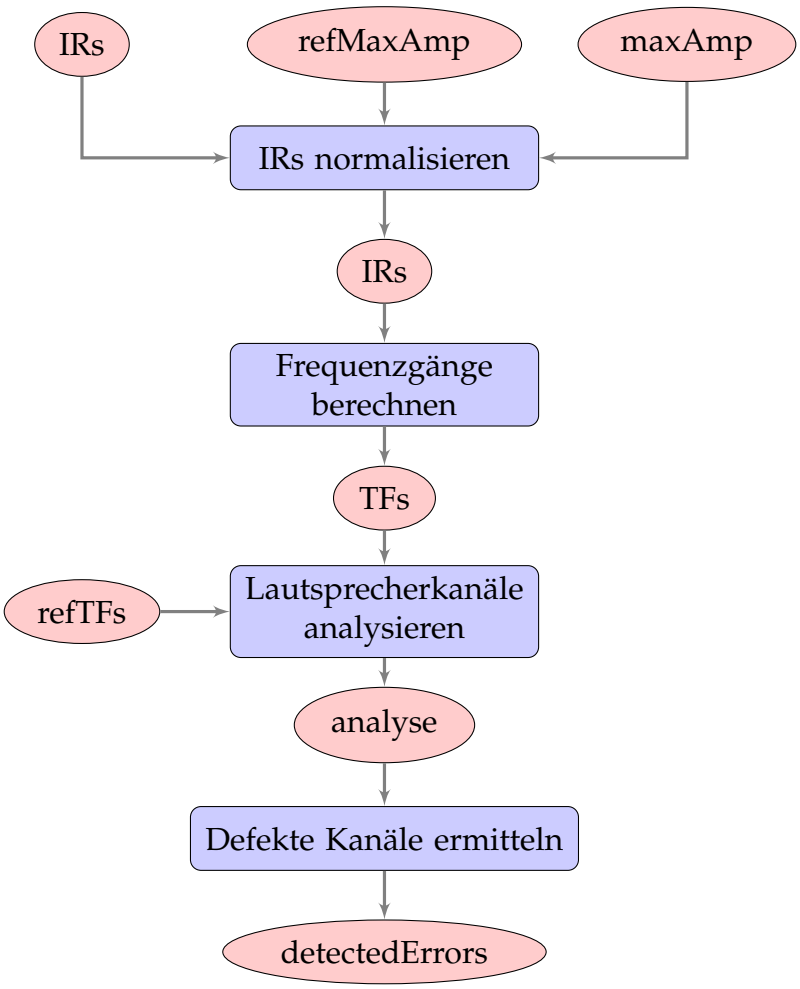


Abbildung 8.1.: Algorithmus zur Identifikation von Defekten eines Lautsprecherkanals

»analyse«-Struct werden abschließend die Kanäle gefiltert, bei denen entweder Lautsprecherausfälle oder Pegeldifferenzen diagnostiziert wurden. Das Ergebnis des Algorithmus wird in das Struct »detectedErrors« abgelegt und als **_detectedErrors.mat* gesichert.

8.1.1. Mittelwert der Summe der quadratischen Abweichungen

Bevor die einzelnen Schritte des Algorithmus erläutert werden, wird kurz das mathematische Werkzeug *Mean Squared Error*, kurz MSE, beschrieben, das von großer Bedeutung für die Implementierung der hier vorgestellten Detektion ist. Der MSE ei-

ner diskreten Messreihe m mit N Samples ergibt sich aus dem Mittelwert der Summe der quadrierten Abstände von $m(k)$ und $ref(k)$ einer Referenzmessreihe ref.

$$MSE = \frac{1}{N} \sum_{k=1}^{N} [ref(k) - m(k)]^2 \tag{8.1}$$

Das Messsystem nutzt den MSE für zwei Aufgaben. Zum einen können quantitative Aussagen zum Unterschied zweier Kurven getroffen werden und zum anderen werden mit Hilfe des MSE Kurvenanpassungen vorgenommen.

8.1.2. Kurvenangleichung mittels MSE

Der Algorithmus, der die Angleichung an eine Referenzkurve vornimmt, verändert die Verstärkung der anzupassenden Kurve so lange, bis der kleinstmögliche MSE gefunden ist. Dazu wird zunächst die Verstärkungsrichtung ermittelt. Sie gibt an, ob die Werte der Kurve angehoben oder abgesenkt werden müssen. Nun wird die Kurve mit einem festgelegten Faktor gemäß der Verstärkungsrichtung iterativ verändert. Bei jedem Schritt wird die Verstärkungsrichtung geprüft. Ändert sie sich, wird der Verstärkungsfaktor halbiert. Die anzupassende Kurve oszilliert also in immer kleineren Abständen um die Referenzkurve. Dies geschieht, bis der aktuelle MSE dem vorherigen MSE entspricht oder eine definierte Schrittanzahl erreicht ist.

8.2. Normalisierung der gemessenen Impulsantworten und Berechnung der Übertragungsfunktionen

Da die Funktion des Mikrofonvorverstärkers im Raum H0104 von einem Mischpult übernommen wird, das nicht ausschließlich für das Messsystem genutzt wird, gilt die Annahme einer nicht konstanten Verstärkung des Mikrofonsignals zwischen nicht aufeinanderfolgenden Messreihen. Aus diesem Grund muss ein konstanter Verstärkungsfaktor berechnet werden, der es ermöglicht, die Amplituden der IRs an die Amplituden der Referenzmessung anzupassen, bevor die FFT der IRs durchgeführt wird. Eine herkömmliche Normalisierung kommt nicht in Frage, da sich diese nur auf einen Wert, in diesem Fall die maximale Amplitude eines Kanals, bezieht. Handelt es sich hierbei um einen defekten Kanal, dessen Maximalamplitude von der eines intakten Kanals abweicht, wird ein fehlerhafter Verstärkungsfaktor erkannt.

Abbildung 8.2 gibt einen Überlick über den angewandten Algorithmus. Werden

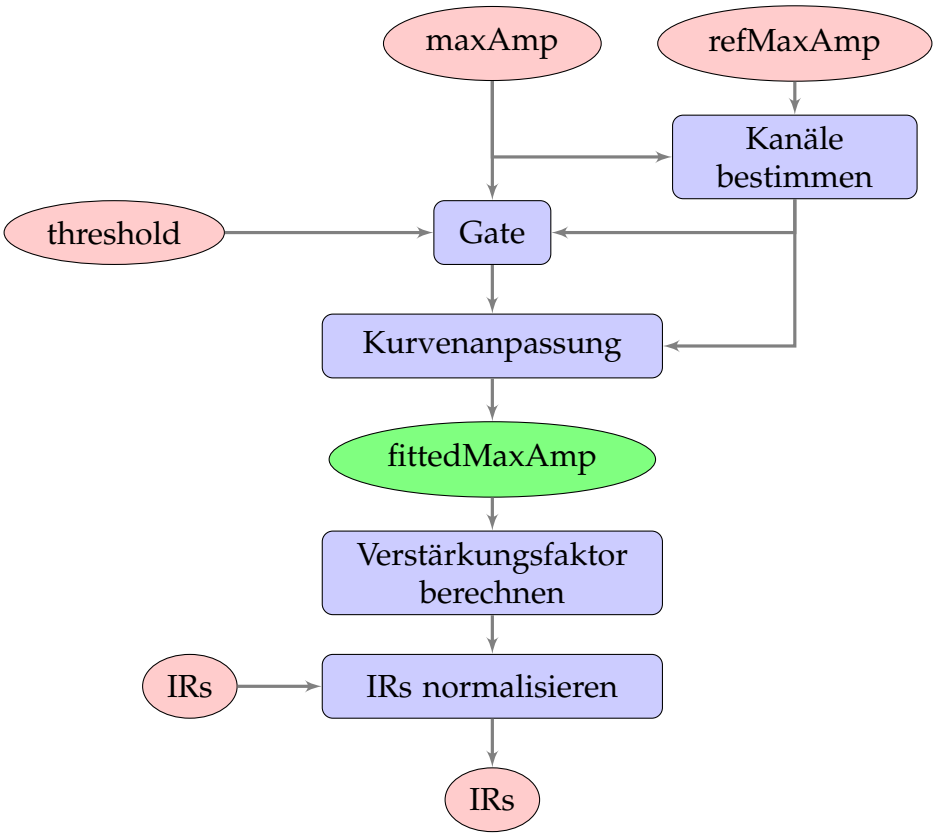


Abbildung 8.2.: Algorithmus zur Normalisierung der IRs

nicht alle Kanäle des Lautsprechersystems geprüft, muss das Struct »refMaxAmp« dahingehend angepasst werden, dass nur die relevanten Kanäle in ihm enthalten sind. Ein Gate stellt fest, ob die Pegeldifferenz von A und A_{Ref} eines Kanals einen vorgegebenen Threshold überschreitet. In einem solchen Fall wird dieser Kanal bei den anknüpfenden Berechnungen nicht weiter berücksichtigt. Im nächsten Arbeitschritt wird eine Kurvenanpassung der A in »maxAmp« an die A_{Ref} in »refMaxAmp« durchgeführt, wie sie in Kapitel 8.1.2 beschrieben ist. Die Amplitudendifferenz von der so ermittelten angepassten Amplitude A_{Fitted} und A eines Kanals ergibt den Verstärkungsfaktor, mit welchem die IRs normalisiert werden. Im Anschluss werden sie mittels FFT in den Frequenzbereich transformiert, die resultierenden $|H|_{dB}$ geglättet, in das Struct »TFs« geschrieben und in der Datei *_*TFs.mat* gespeichert.

In Abbildung 8.3 ist das Ergebnis der Amplitudenangleichung mit deaktiviertem Gate für eine Messung im Raum H0104 dargestellt. Bei dieser Messung wurden Defekte an einigen Kanälen simuliert, was Einfluss auf die Amplituden der jeweiligen

IRs zur Folge hatte. Es ist zu erkennen, dass durch das Einbeziehen der Amplituden aller Kanäle einer Messung von A_{Ref} abweichende Werte ausgeglichen werden.

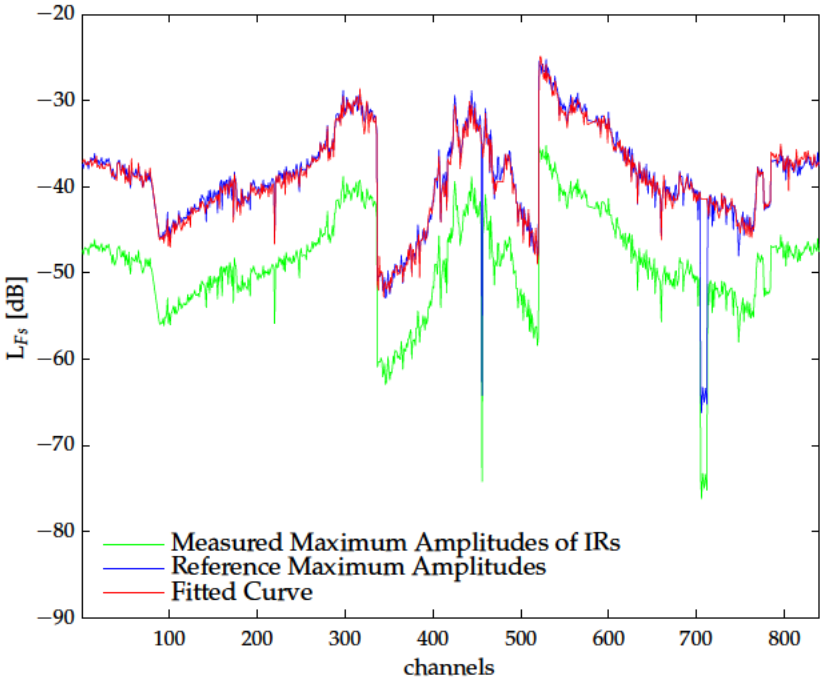


Abbildung 8.3.: Angleichung von A in »maxAmp« an die A_{Ref} in »refMaxAmp«

8.3. Analyse der Lautsprecherkanäle

Nachdem die $|H|_{dB}$ berechnet wurden, wird eine Analyse der Lautsprecherkanäle vollzogen. Der in Abbildung 8.4 dargestellte Algorithmus überprüft die $|H|_{dB}$ der einzelnen Kanäle auf Abweichungen bezüglich der $|H_{Ref}|_{dB}$ und identifiziert gegebenenfalls Defekte. Desweiteren werden die Pegeldifferenzen zwischen A und A_{Ref} festgestellt. Die Informationen aller gemessenen Kanäle werden abschließend in das Struct »anlayse« geschrieben und als **_analyse.mat* gespeichert.

8.3.1. Prüfung des Kanalstatus und Berechnung der Pegeldifferenz

Der Kanalstatus gibt an, ob ein Lautsprecherkanal Schall emittiert oder nicht. Die Berechnungen hierfür finden bei der Erzeugung des »IRs«- und »maxAmp«-Structs

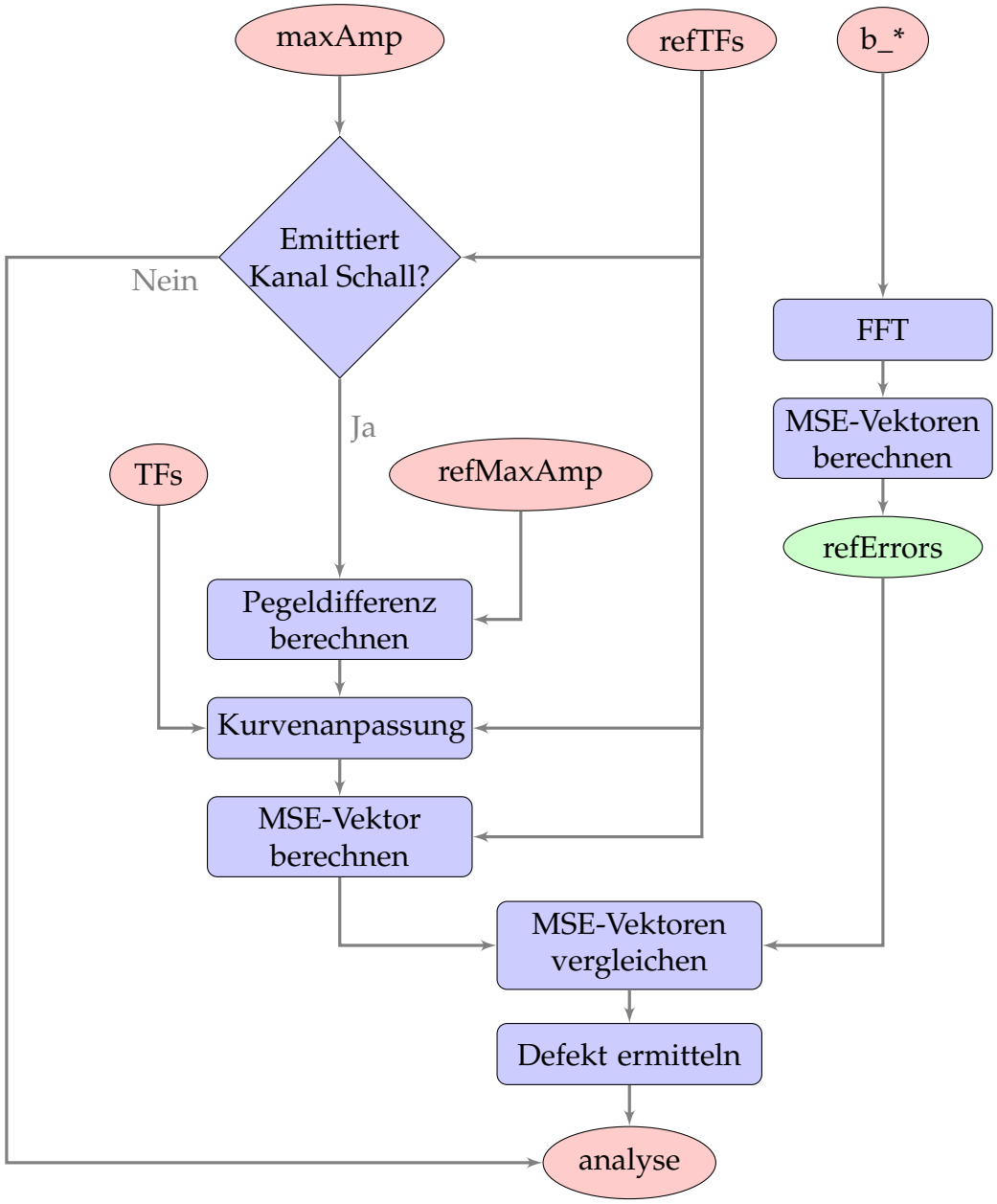


Abbildung 8.4.: Algorithmus zur Analyse der einzelnen Lautsprecherkanäle

statt. Im Struct »maxAmp« wird der Kanalstatus vermerkt. »maxAmp» wird nun ausgelesen und der Status entsprechend im Struct »analyse« vermerkt. Sendet ein Kanal Schall aus, wird im nächsten Arbeitsschritt die Amplitudendifferenz von A und A_{Ref} gebildet. Es besteht die Möglichkeit einen Toleranzpegel anzugeben, der zur Folge hat, dass eine Differenz nur dann gespeichert wird, wenn dieser überschritten wird.

8.3.2. Anpassung der gemessenen Betragsfrequenzgänge an die entsprechenden Referenzbetragsfrequenzgänge

Der Ausfall von Lautsprechertreibern hat Konsequenzen für den Verlauf der $|H|_{dB}$ eines Kanals und wurde in Kapitel 3.3.1 untersucht. In dem nächsten Schritt des Detektionsalgorithmus wird die $|H|_{dB}$ der $|H_{Ref}|_{dB}$ angenähert.

Hierfür wurde der zuvor beschriebene Angleichungsalgorithmus dahingehend erweitert, dass er abhängig von Frequenzbändern die Anpassung vornimmt. Durch die Übergabe der Anfangs- und Endwerte beliebig vieler Frequenzbänder wird dieser Modus aktiviert. Zunächst wird das Frequenzband bestimmt, in dem der Verlauf der $|H|_{dB}$ am ehesten dem der $|H_{Ref}|_{dB}$ entspricht. Dazu werden die MSE-Werte für alle Frequenzbänder einzeln gebildet. Das Frequenzband, das gesucht wird, weist den niedrigsten MSE auf. Sodann wird die Kurvenanpassung auf die in Kapitel 8.2 erläuterte Weise bewerkstelligt, wobei nur der zuvor eruierte Frequenzbereich beachtet wird.

Abbildung 8.5 zeigt das Ergebnis der Frequenzgangsanpassung anhand einer Messung eines Lautsprecherkanals mit impliziertem Defekt. Der simulierte Ausfall von

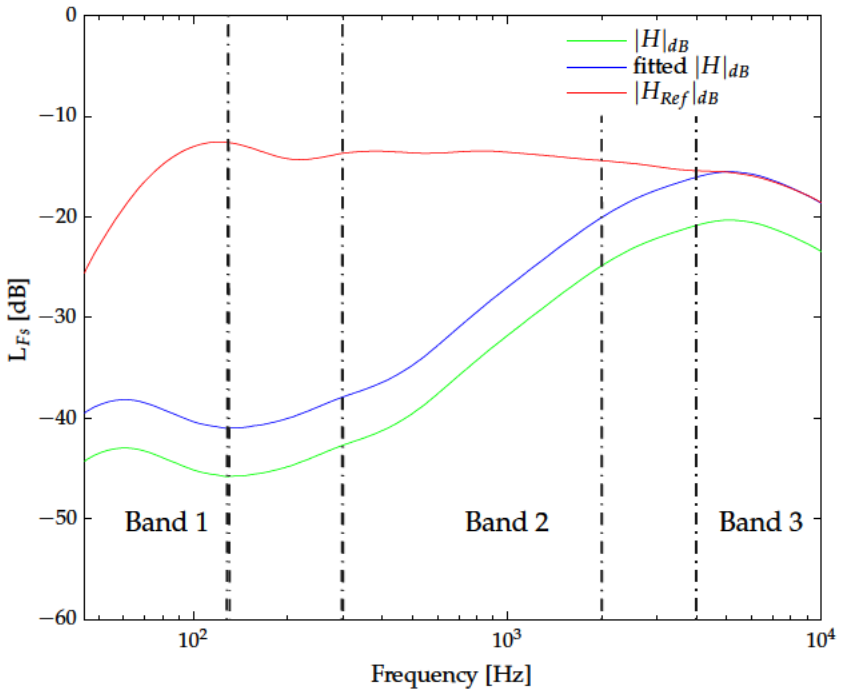


Abbildung 8.5.: Anpassung eines gemessenen Frequenzgangs an einen Referenzfrequenzgang unter Berücksichtigung von Frequenzbändern

TT und $LS2$ bewirkt eine deutlich erkennbare Abweichung des Verlaufs der $|H|_{dB}$ des gemessenen Kanals von dem der $|H_{Ref}|_{dB}$. Für die Anpassung der $|H|_{dB}$ wurden die drei Frequenzbänder mit den Eckfrequenzen 40 Hz und 130 Hz, 300 Hz und 2 kHz sowie 4 kHz und 12 kHz gewählt. Bei der Anpassung orientierte sich der Algorithmus offensichtlich an Band 3. Die Parameter der Frequenzbänder müssen mit Bedacht gewählt werden. Sie können aus den in Kapitel 3.3.1 gemessenen Frequenzgängen abgeleitet werden. Wie aus Abbildung 3.6 ersichtlich, kann der Frequenzgang eines Kanals des WFS-Lautsprechermoduls in drei Bereiche eingeteilt werden. Der untere Frequenzbereich wird durch TT bestimmt. Während Ausfälle der Breitbandlautsprecher $LS1$ und $LS2$ den mittleren Bereich beeinflussen, wird der obere durch Defektkombinationen mit ausgefallenem $LS3$ tangiert. Die von den gewählten Frequenzbändern nicht erfassten Frequenzbereiche stellen Übergangszonen dar, in denen der Einfluss von Defekten noch uneindeutig ist und deren Einbeziehen das Ergebnis der Anpassung verschlechtert.

8.3.3. MSE-Vektor-Berechnung

Nachdem die $|H|_{dB}$ bearbeitet wurden, wird ein Vektor erstellt, der MSE-Werte für einzelne Frequenzbereiche der $|H|_{dB}$ enthält. Mittels dieser Art der Merkmalsextraktion wird die $|H|_{dB}$ eines Lautsprecherkanals fortan durch wenige Werte repräsentiert. Die Frequenzbereiche werden in *measurement.ini* festgelegt. Sie richten sich nach den bei der Kurvenanpassung verwendeten, müssen aber für die jeweilige Messsituation optimiert werden. Für die WFS-Anlage im Raum H0104 stellte sich die Verwendung von vier Frequenzbändern als geeignet heraus. Das mittlere Frequenzband wurde in zwei Bänder unterteilt. Abbildung 8.6 zeigt die Aufgliederung der $|H|_{dB}$ und $|H_{Ref}|_{dB}$ in die einzelnen Frequenzbereiche zur Berechnung des MSE-Vektors. Die MSE-Werte sind in den jeweiligen Bändern eingetragen. Bei der Erstellung des MSE-Vektors kann, wie es hier der Fall ist, ein Threshold angegeben werden, bei dessen Überschreitung der MSE-Wert nicht weiter steigt. In den Berechnungen für Abbildung 8.6 wurde der Threshold auf den Wert 100 gesetzt.

8.3.4. Erstellung des Structs »refErrors«

Um aus einem MSE-Vektor Schlüsse auf das Vorhandensein eines bestimmten Defekts zu ziehen, werden MSE-Vektoren benötigt, mit denen dieser verglichen wird und die die etwaigen Defekte repräsentieren. Diese Referenz-MSE-Vektoren werden

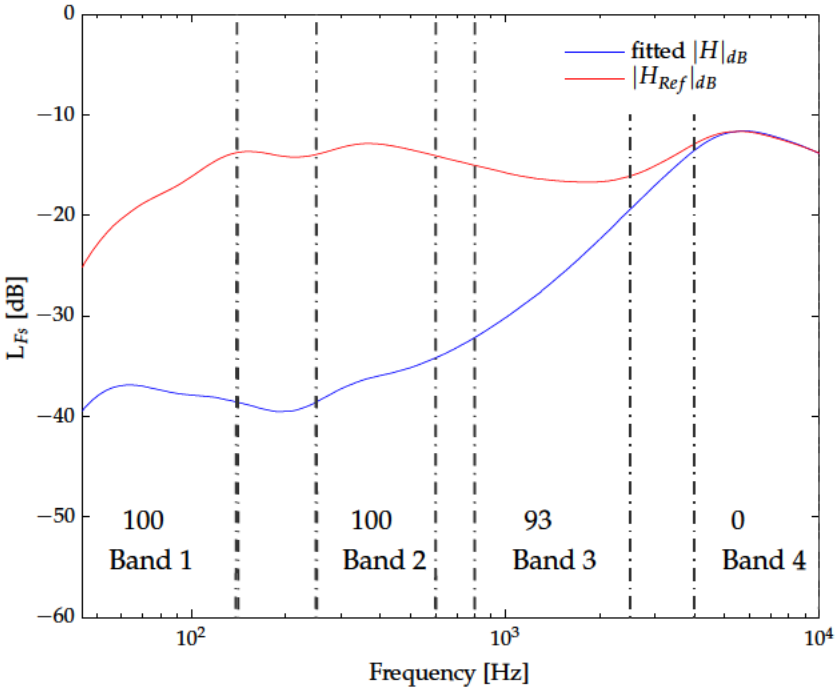


Abbildung 8.6.: Berechnung des MSE-Vektors eines gemessenen Lautsprecherkanals. Die Eckfrequenzen der vier Frequenzbänder sind 40 Hz und 140 Hz, 250 Hz und 600 Hz, 800 Hz und 2,5 kHz sowie 4 kHz und 12 kHz

aus den Filterkoeffizienten *b_*.mat* des Filter-Sets gewonnen, dessen Generierung in Kapitel 4 erläutert wurde.

Zunächst werden die Frequenzgänge der möglichen Defekte aus den Filterkoeffizienten via FFT berechnet. Die FFT-Parameter gleichen denen, welche auch für die Erzeugung der $|H|_{dB}$ gewählt und vor der Messung in der Datei *measurement.ini* definiert wurden. Nun werden aus den Frequenzgängen und einem weißen Referenzfrequenzgang die Referenz-MSE-Vektoren berechnet und zusammen mit den jeweiligen Defektbezeichnungen in dem Struct »refErrors« gespeichert. Die Frequenzbänder entsprechen den in Kapitel 8.3.3 verwendeten. In Tabelle 8.1 findet sich eine Auflistung der Referenz-MSE-Vektoren für dieses Beispiel.

8.3.5. Vergleich der MSE-Vektoren

Um mögliche Defekte zu identifizieren, wird der Referenz-MSE-Vektor in dem Struct »refErrors« gesucht, dessen Werte am nächsten an denen des MSE-Vektors der zu überprüfenden Messreihe sind. Hierfür werden die MSE-Werte aus dem MSE-Vektor

Defekt	MSE-Vektor			
111111	0	0	0	0
000111	100	100	94	0
001111	100	80	3	0
010111	100	100	36	0
011111	100	0	0	0
100111	0	100	94	0
101111	0	87	3	0
110111	0	100	36	0
111011	0	100	100	100

Tabelle 8.1.: Aus dem Filter-Set berechnete Referenz-MSE-Vektoren der potentiellen Defekte für Frequenzbänder mit den Eckfrequenzen: 40 Hz und 140 Hz, 250 Hz und 600 Hz, 800 Hz und 2,5 kHz sowie 4 kHz und 12 kHz

und den einzelnen Referenz-MSE-Vektoren berechnet. Der Referenz-MSE-Vektor, der den kleinsten Wert verursacht, bestimmt den erkannten Defekt. In Tabelle 8.2 sind die MSE-Werte, die bei dem Vergleich entstehen, für mehrere Kanäle, die denselben Defekt aufweisen, eingetragen. Die kleinsten Werte geben den Defekt an und

Defekt	MSE-Werte für Defekt 101111 auf Kanal:	61	242	418	575
111111		743	763	1128	946
000111		5162	5090	4734	4960
001111		2645	2603	2407	2554
010111		3304	3246	2917	3218
011111		3216	3205	3482	3404
100111		2676	2635	2366	2488
101111		**265**	**253**	**100**	**163**
110111		819	793	550	657
111011		5484	5441	5169	5292

Tabelle 8.2.: MSE-Werte des Vergleichs von MSE-Vektor und Referenz-MSE-Vektoren für vier Lautsprecherkanäle

sind vergrößert dargestellt. Der Defekt wird bei allen Lautsprecherkanälen eindeutig identifiziert. Tabelle 8.3 zeigt ausschnittsweise weitere Ergebnisse einer Messung mit simulierten Defekten. Auch in diesem Fall werden alle Defekte erkannt. Die Detektion scheint verlässlich zu funktionieren. Es sind aber noch weitere Testmessungen erforderlich, um die Reliabilität des Algorithmus zu eruieren.

Kanal	Defekt	MSE-Vektor				erkannt
56	111111	0	0	0	1	111111
61	101111	0	55	1	0	101111
171	010111	100	100	22	0	010111
196	011111	100	0	0	0	011111
206	000111	100	100	82	0	000111
216	100111	0	100	80	0	100111
346	111111	0	0	0	0	111111
416	000111	100	100	93	0	010111
541	011111	100	0	0	0	011111
556	001111	100	64	2	0	001111
566	000111	100	100	89	0	000111
586	110111	0	100	26	0	110111
646	111111	0	0	0	0	111111
716	100111	0	100	75	0	100111
831	111011	0	100	100	100	111011

Tabelle 8.3.: MSE-Vektoren und detektierte Defekte für verschiedene Lautsprecherkanäle mit unterschiedlichen implizierten Defekten

Die Informationen, die zur Prüfung des Kanalstatus ermittelt wurden, werden in das Struct »analyse« geschrieben, dessen Felder in Tabelle 8.4 verzeichnet sind.

Name	Feld	Funktion
analyse	MSE	MSE-Vektor
	channel	Lautsprecherkanal
	fs	Abtastfrequenz
	errorDetected	Detektierte Defektkombination
	levelDifference	Pegeldifferenz
	day	Tag
	time	Uhrzeit
	errorInSweep	Simulierte Defektkombination

Tabelle 8.4.: Aufbau des Structs »analyse«

8.4. Extraktion der als defekt eingestuften Lautsprecherkanäle

Im letzten Arbeitsschritt des Detektionsalgorithmus werden sämtliche defektbehafteten Kanäle aus dem Struct »analyse« extrahiert, in das Struct »detectedErrors« geschrieben und als **_detectedErrors.mat* gespeichert. Dies geschieht analog zu dem in Abbildung 8.7 dargestellten Flussdiagramm. Zunächst wird geprüft, ob ein Defekt

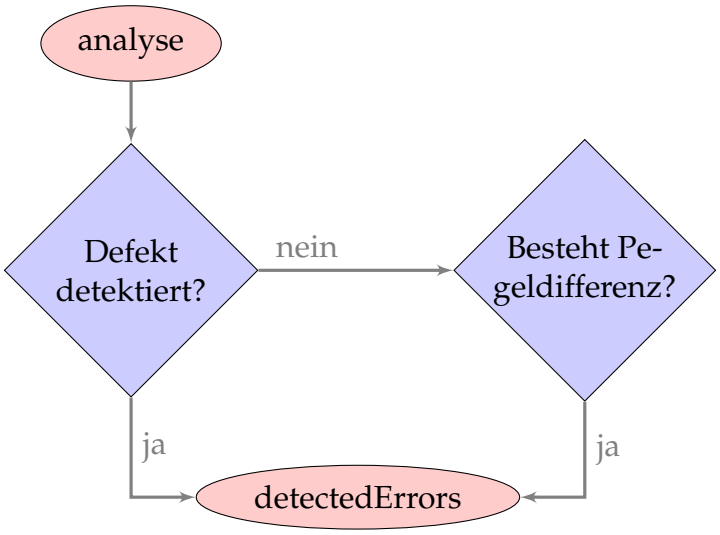


Abbildung 8.7.: Verarbeitung der Analysedaten

vorliegt, der durch einen Lautsprecherausfall oder eine Kombination von Ausfällen bedingt ist. Ist dies nicht der Fall, wird die Pegeldifferenz inspiziert. Ist eine solche in dem »analyse«-Struct vermerkt, wird auch dies als Defekt gewertet. Den Aufbau des Structs »detectedErrors« zeigt Tabelle 8.5.

Name	Feld	Funktion
detectedErrors	MSE	MSE-Vektor
	channel	Lautsprecherkanal
	fs	Abtastfrequenz
	errorDetected	Detektierte Defektkombination
	levelDifference	Pegeldifferenz
	day	Tag
	time	Uhrzeit
	errorInSweep	Simulierte Defektkombination

Tabelle 8.5.: Aufbau des Structs »detectedErrors«

8.5. Ausgabe der Detektionsergebnisse

Die Resultate der Detektion werden als *txt*-File gespeichert. Am Beispiel von Messungen im H0104 wird ein solches im Folgenden veranschaulicht.

```
MEASUREMENT RESULTS


Date: 25-Oct-2010


  -----------------
 | o o o o o o o o | 3rd Loudspeaker
 | o o o o o o o o | 2nd Loudspeaker
 | o o o o o o o o | 1st Loudspeaker
 |                 |
 | O O O O O O O O | Sub
 |_________________|


3 Channels are not working correctly!
-------------------------------------------------------
-------------------------------------------------------


Channel            7
Node, Channel      1, 7
Not working        2nd Loudspeaker
-------------------------------------------------------

Channel            37
Node, Channel      1, 37
Not working        Sub
                   1st Loudspeaker
                   2nd Loudspeaker
-------------------------------------------------------

Channel            48
Node, Channel      1, 48
Not working        3rd Loudspeaker
```

9. Evaluation des Messsystems

Die Evaluation des Messsystems erfolgt im Hinblick auf zwei Aspekte. Zum einen soll die Zuverlässigkeit der Detektion und zum anderen die Geschwindigkeit gegenüber dem herkömmlichen Messverfahren mit einzelnen Sweeps überprüft werden.

9.1. Messungen zur Verifikation des Detektionsalgorithmus

Um die Reliabilität des Messsystems zu untersuchen, wurden im Raum H0104 fünf Testmessungen mit implizierten Defekten durchgeführt. Hierfür wurden während der Erstellung der »swp«-Structs einzelne Sweeps der Sweep-Matrix mittels des in Kapitel 4 vorgestellten Filter-Sets gefiltert. Die Simulation des Ausfalls eines ganzen Lautsprecherkanals wurde durch ein Auf-Null-Setzen der Amplituden erreicht. Wie in Tabelle 3.2 notiert, ergeben sich zehn mögliche Zustände, die von dem Messsystem erkannt werden sollen.

9.1.1. Referenzmessung

Zunächst wurde eine Referenzmessung durchgeführt, mit der die notwendigen Parameter für die Erzeugung der Sweep-Matrizen sowie die für die Detektion erforderlichen Referenzfrequenzgänge mit den dazugehörigen maximalen Amplituden gewonnen wurden.

Für die Referenzmessung wurde ein einzelner Sweep sequentiell den 832 Lautsprecherkanälen im H0104 zugeführt. Die Gap-Zeit musste hier so gewählt sein, dass der Abklingvorgang vollständig in der IR enthalten war. Sie wurde auf 1,2 s gesetzt. Die Forderung, dass die Dauer des Sweeps größer als die Nachhallzeit sein muss, wurde mittels der Sweep-Ordnung 16 bei einer Abtastrate von 44,1 kHz gewährleistet. Es folgten die in Kapitel 7 erläuterten Arbeitsschritte. Zur Bestimmung der Nachhallzeit wurde die Methode nach Lundeby verwendet. Da einige Lautsprecherkanäle

des WFS-Systems schon defektbehaftet sind, wurden deren Werte durch eine Mittlung der erhobenen Daten der nächstliegenden Kanäle ersetzt. Die gemessenen akustischen Laufzeiten entsprechen denen, die in Abbildung 7.5 dargestellt sind. Abbildung 9.1 zeigt die laufzeitkorrigierten Gap-Zeiten für das Messsystem im H0104.

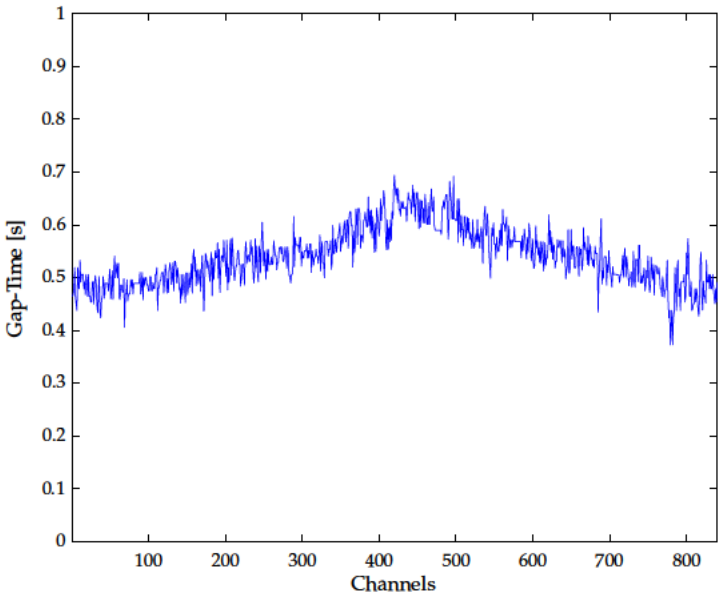


Abbildung 9.1.: Laufzeitkorrigierte Gap-Zeiten für das Messsystem im Raum H0104

9.1.2. Erzeugung von »swp«-Structs mit implizierten Defekten

Messungen im Raum H0104 werden mit mehreren Sweep-Structs vorgenommen. Zum einen können nur maximal 64 Kanäle von WFS-Control zu den Nodes geschickt werden und zum anderen limitiert dessen Arbeitspeicher die potentielle Anzahl der Kanäle. Vormessungen ergaben, dass Messungen mit 32-kanalige Sweep-structs zuverlässig funktionieren. Eine höhere Anzahl paralleler Sweeps führte zu Ausfällen. Im Gegensatz dazu ist es im WFS-Labor im Raum EN325 der TU Berlin möglich, 128-kanalige Sweep-structs zu nutzen. Statt den 4 GB Arbeitsspeicher von WFS-Control, besitzt der Computer des WFS-Labors 32 GB.

Für die Testmessungen wurden zunächst die 10 % der 832 Lautsprecherkanäle zufällig gewählt, welche die zu detektierenden Defekte aufweisen sollten. Die neun einzelnen Ausfallszenarien wurden dann in beliebiger Reihenfolge, aber in ihrer Anzahl gleichmäßig auf die 10 % verteilt. Diese Informationen wurden als Struct gespeichert

und der Funktion übergeben, die für die Generierung der Sweep-Structs zuständig ist. Bei der Erzeugung wurden die Simulationen möglicher Defektverknüpfungen mittels der FIR-Filterkoeffizienten des erstellten Filter-Sets realisiert. Für die spätere Auswertung der Messergebnisse wurden die implizierten Defekte in den jeweiligen Sweep-Structs vermerkt.

9.1.3. Testmessungen

Welchen Lautsprecherkanälen welcher Defekt innewohnt, wurde bei jeder der fünf Messungen variiert. Zudem wurde nach jeder Testmessung die Vorverstärkung des Messmikrofons verändert. Nachdem die Testmessungen durchgeführt waren, wurde der Detektionsalgorithmus noch optimiert, indem die Anzahl der Frequenzbänder und deren Eckfrequenzen feinjustiert wurden. In Tabelle 9.1 sind die Daten der für die WFS-Lautsprecher-Module verwendeten Frequenzbänder eingetragen.

Band	$f_u[Hz]$	$f_o[Hz]$
1	70	140
2	250	600
3	800	2k
4	4k	10k

Tabelle 9.1.: Frequenzbänder des Detektionsalgorithmus für die WFS-Lautsprecher-Module

9.1.4. Ergebnisse

Bevor die Auswertung erfolgen kann, muss berücksichtigt werden, dass einige Lautsprecherkanäle zum Zeitpunkt der Messungen Defekte aufweisen. So funktioniert bei Kanal 493 nur der Tieftöner, bei den Kanälen 701 bis 704 hingegen nur die Mittelhochtöner. Das Modul mit den Kanälen 577 bis 584 ist aufgrund einer defekten ADAT-Leitung abgeschaltet. Wurde einer dieser Kanäle zusätzlich mit einem simulierten Defekt versehen, ist diese Möglichkeit bei der Auswertung der Messergebnisse miteinzubeziehen. Die Identifizierung eines kompletten Kanalausfalls von Kanal 577, muss dementsprechend als richtig bewertet werden, auch wenn dem Sweep ein anderer Defekt impliziert wurde. In Tabelle 9.2 sind die Ergebnisse der fünf Messreihen erfasst, nachdem der Detektionsalgorithmus optimiert wurde. Die Resultate

Defekt	Messung 1		Messung 2		Messung 3		Messung 4		Messung 5		Treffer
	sim.	erk.	sim.	erk.	sim.	erk.	sim.	erk.	sim.	erk.	
111111	736	736	739	739	737	737	737	737	736	736	100%
110111	9	9	9	9	10	10	10	10	9	9	100%
101111	9	9	9	9	9	9	9	9	9	9	100%
100111	10	10	9	9	9	9	10	10	9	9	100%
111011	11	11	10	10	10	10	10	10	11	11	100%
011111	12	12	11	11	12	12	11	11	13	13	100%
010111	9	9	10	10	9	9	8	8	9	9	100%
001111	9	9	9	9	8	8	8	8	9	9	100%
000111	9	9	9	9	8	8	10	10	8	8	100%
011011	17	17	17	17	17	17	17	17	17	17	100%

Tabelle 9.2.: Ergebnisse der fünf Testmessungen

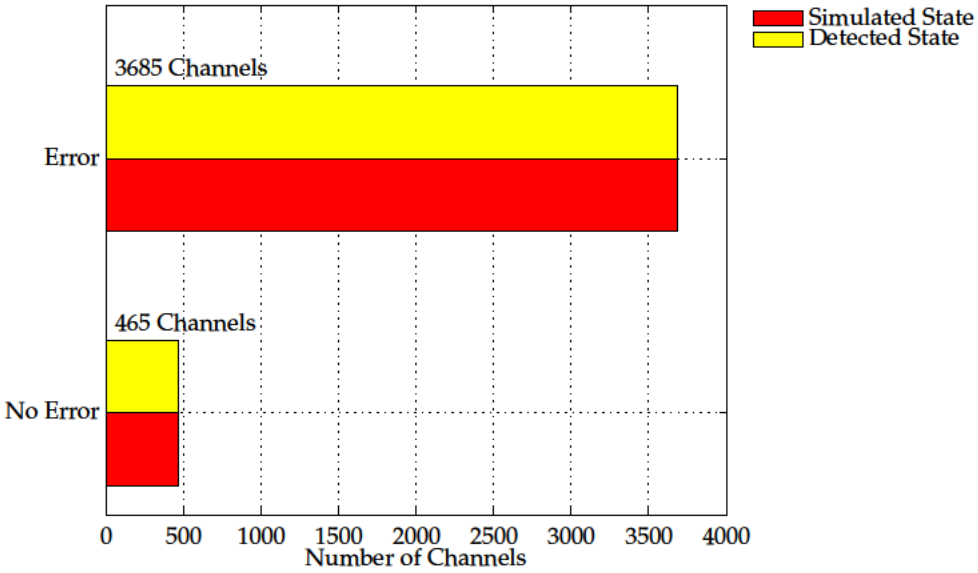


Abbildung 9.2.: Zusammenfassung der drei Messungen

der Messreihen sind in den Abbildungen 9.2 und 9.3. zusammengefasst. Die ausgeglichene Streuung der potentiellen Defektkombinationen ist gut zu erkennen. Die etwas größere Anzahl von 111011 (Mittelhochtöner defekt) und 011111 (Tieftöner defekt) sowie die deutlich größere von 011011 (ganzer Kanal ausgefallen) erklären sich durch die erwähnten tatsächlich vorliegenden Defekte. Die Resultate bezeugen dem Detektionsalgorithmus eine fehlerfreie Identifizierung aller in diesen Messreihen vorkommenden Defektkombinationen.

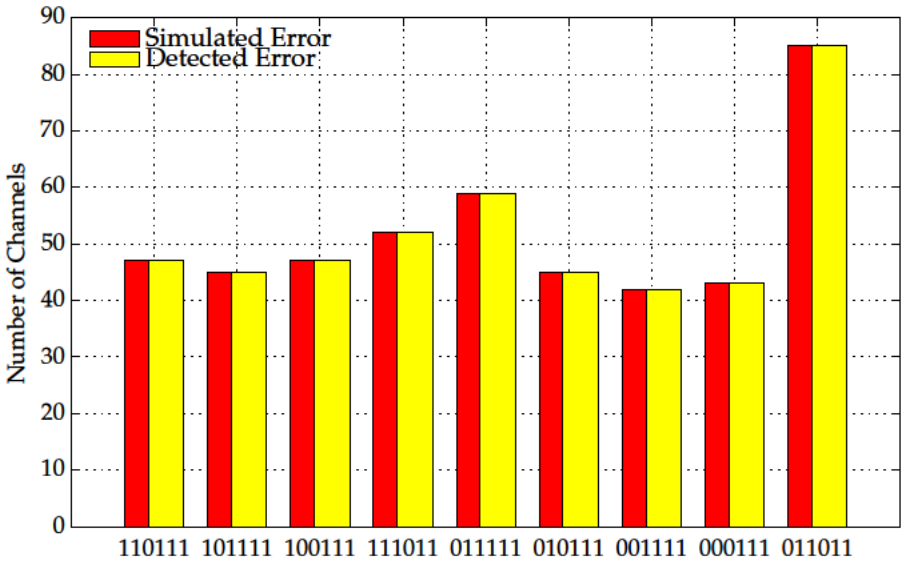


Abbildung 9.3.: Identifizierte und nicht identifizierte Defekte der drei Messungen

9.2. Vergleich der Messgeschwindigkeiten

Es wurden die zeitlichen Differenzen des in Kapitel 6.4.3 erläuterten Vorgangs für Messungen mit und ohne Overlapping-Methode ermittelt, sowie die daraus folgenden zeitlichen Konsequenzen für den gesamten Messvorgang erfasst. Der betrachtete Software-Abschnitt beinhaltet das Laden des Sweep-Structs, den Aufbau der nötigen JACK-Verbindungen, das Abspielen und Aufnehmen des Sweep-Structs, den Abbau der JACK-Verbindungen und das Speichern der Aufnahmen. Für die Testmessungen mit der Overlapping-Methode kamen 32-kanalige Sweep-Structs zum Einsatz. Die Gap- und akustischen Laufzeiten zur Generierung der Sweep-Structs wurden von der Referenzmessung aus Kapitel 9.1.1 übernommen. Die Gap-Zeiten wurden ebenfalls für die Messungen ohne Overlapping-Methode verwendet, welche mit einkanaligen Sweep-Structs durchgeführt wurden.

Jeweils drei Messreihen wurden im Raum H0104 vorgenommen. Neben den benötigten Zeiten für den Vorgang des Abspielens und Aufnehmens, wurden die Zeiten erfasst, die für das Laden, Speichern und Routen beansprucht werden. Darüber hinaus wurden die Gesamtmessdauern bestimmt. Anschließend wurden die Zeiten gemittelt. In Tabelle 9.3 sind die Ergebnisse eingetragen. Die Abspiel- und Aufnehmdauer korrespondiert mit den zeitlichen Längen der jeweiligen Sweep-Structs. Messungen mit dem Overlapping-Verfahren erfordern längere Zeiten für das Laden der

	Abspielen & Aufnehmen	Laden, Speichern, JACK-Verbindungen	Gesamter Messvorgang
Mit Overlapping (32 Kanal)	8 Min.	7 Min.	22 Min.
Ohne Overlapping	28 Min.	5 Min.	42 Min.

Tabelle 9.3.: Vergleich der benötigten Messzeiten für Messungen mit und ohne Overlapping-Methode

Sweep-Structs, Routen der Audio-Wege und Abspeichern des Aufgenommenen, was sich mit den größeren Datenmengen erklären lässt, die bearbeitet werden müssen. Von der Initialisierung des Messsystems bis zur Ergebnisausgabe kann die Messdauer durch die Overlapping-Methode um circa 48 % gesenkt werden.

10. Fazit

10.1. Zusammenfassung

Es wurde ein softwarebasiertes akustisches Messsystem entwickelt, das es erlaubt, den Funktionszustand einzelner Kanäle vielkanaliger Lautsprecheranlagen automatisiert zu überprüfen. Die Software wurde in GNU Octave realisiert, ist vollständig kompatibel mit Mathworks MATLAB® und läuft auf den Betriebssystemen Mac OS X und GNU/Linux. Die Grundlage bildet ein FFT-basiertes Messverfahren zur Akquise von Impulsantworten, das exponentielle Sweeps als Anregungssignale nutzt. Um die Messdauer zu optimieren, wurde die Overlapping-Methode implementiert, mit der quasi-parallele Messungen mit sich überlappenden Sweeps ermöglicht werden. Die Messsoftware wurde mit Funktionen ausgestattet, die es erlauben, die für die Overlapping-Methode benötigten Parameter automatisiert zu bestimmen. Ein Algorithmus wurde vorgeschlagen, der durch die Interpretation von Frequenzgangs- und Amplitudenunterschieden in der Lage ist, ausgefallene Teile eines Lautsprecherkanals zu identifizieren. Am Beispiel der Lautsprechermodule der WFS-Anlage der TU Berlin, wurden die Einflüsse von defekten Komponenten auf den Frequenzgang eines Kanals untersucht und definiert, dass neben Pegelabweichungen ausgefallene Lautsprechertreiber von der Software erkannt werden sollen. Dem Detektionsalgorithmus müssen sowohl ein Referenzpaar, das die spezifischen Frequenzgänge und Maximalamplituden jedes einzelnen Kanals in einwandfreiem Betriebszustand beinhaltet, als auch die Frequenzgangsunterschiede für die potentiellen Defekte vorliegen. Letztere müssen nur einmal pro Lautsprechertyp ermittelt werden und können dann für alle Wiedergabesysteme, die sich dieses Typs bedienen, verwendet werden. Für die WFS-Lautsprechermodule wurden diese, im Zuge der Erzeugung eines Sets von FIR-Filtern zur Simulation der potentiellen Defekte, gewonnen. Das Referenzpaar wird bei der Einrichtung des Messsystems bestimmt. Die Abweichung eines Frequenzgangs von dessen Referenzfrequenzgang wird von dem Detektionsalgorithmus mit den Frequenzgängen der möglichen Defekte verglichen und so der Ausfall

eines Lautsprechertreibers erkannt. Ist dies nicht der Fall, wird geprüft, ob sich die gemessene Maximalamplitude von deren Referenzamplitude unterscheidet.

Das Messsystem wurde in die WFS-Anlage der TU Berlin integriert und vollständig eingerichtet. Mit diesem System wurden Testmessungen vollzogen, die zum einen die Zuverlässigkeit des Detektionsalgorithmus mittels systematischer Simulation fehlerhafter Betriebszustände und zum anderen die Zeitoptimierung durch die Anwendung der Overlapping-Methode eruieren sollten. Sämtliche implizierten Defekte wurden von der Messsoftware erkannt, weswegen dem Algorithmus eine hohe Reliabilität bescheinigt werden kann. Im Vergleich zur sequentiellen Vermessung der Lautsprecherkanäle, konnten die Messzeiten für das WFS-System durch die Verwendung der Overlapping-Methode etwa halbiert werden. Das Messsystem ist imstande, die 832 Kanäle der WFS-Anlage der TU Berlin innerhalb von circa 20 Minuten verlässlich auf Lautsprecherausfälle und Pegeldifferenzen hin zu überprüfen.

10.2. Ausblick

Mit dem entwickelten Messsystem wurde die WFS-Anlage der TU Berlin um ein Werkzeug erweitert, mit welchem der Funktionszustand aller Lautsprecherkanäle in kurzer Zeit zuverlässig festgestellt werden kann. Durch die sichere Detektion von Lautsprechertreiberausfällen und die Implementierung der Overlapping-Methode, werden Wartungsarbeiten erleichtert und deren Dauer reduziert.

Das Messsystem ist auf andere Wiedergabeanlagen, die aus mehreren Lautsprechern eines Typs bestehen, übertragbar. Hierbei wäre es von Nutzen, eine Datenbank von Filter-Sets anzulegen, die die Frequenzgänge potentieller Defekte verschiedener Lautsprechermodelle nebst den dazugehörigen Informationen zu den Frequenzbändern, die eine optimale Detektion gewährleisten, beinhaltet. Eine solche Datenbank würde das Einrichten des Messsystems weiter vereinfachen, da kein Filter-Set erstellt werden müsste und der Detektionsalgorithmus Zugriff auf die bestmöglichen Parameterwerte hätte, ohne vor Ort justiert werden zu müssen. Die entsprechenden Daten für die WFS-Lautsprechermodule wurden in dieser Arbeit erfasst und können für zukünftige Installationen des Messsystems herangezogen werden.

Eine Erweiterung der Messsoftware dahingehend, Lautsprechersysteme zu unterstützen, bei denen mehrere Lautsprechertypen zum Einsatz kommen, erscheint sinnvoll und würde die Übertragbarkeit noch verallgemeinern.

Für die Identifikation von ausgefallenen Lautsprechertreibern ist ein hoher SNR unerheblich, weswegen das Messsystem dahingehend nicht optimiert wurde. Für die Detektion anderer Defekte könnte dies relevant werden, weswegen eine Ergänzung der Software um die Verwendung von Sweeps mit arbiträren Spektren als Anregungssignale zur Verbesserung des SNR interessant ist.

Desweiteren ist eine Portierung der Software auf das Betriebssystem MS Windows denkbar.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Behrens u. a. 2007] BEHRENS, Tobias ; AHNERT, Wolfgang ; MOLDRZYK, Christoph: Raumakustische Konzeption von Wiedergaberäumen für Wellenfeldsynthese am Beispiel eines Hörsaals der TU Berlin. In: *DAGA*, 2007

[Defrance u. a. 2008] DEFRANCE, Guillaume ; DAUDET, Laurent ; POLACK, Jean-Dominique: Finding the onset of a room impulse response: Straightforward? In: *Journal of the Acoustical Engineering Society* Bd. 124, 2008, S. EL248–254

[Farina 2000] FARINA, Angelo: Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique. In: *108th Convention of the Audio Engineering Society*, 2000

[Feiten und Röbel 1996] FEITEN, B. ; RÖBEL, A.: *Einführung in die digitale Signalverarbeitung*. Technische Unversität Berlin, 1996

[Giese 2009] GIESE, André: *ein optimiertes Messverfahren für Raumimpulsantworten*, Technische Universität Berlin, Magisterarbeit, 2009

[Goltz 2010] GOLTZ, Florian: *Planung und Konfiguration eines Labors zur Wellenfeldsynthese*, Technische Universität Berlin, Magisterarbeit, 2010

[Harris 1978] HARRIS, Fredric J.: On the Use of Windows for Harmonic Analysis with Discrete Fourier Transform. In: *Proc.IEEE* (1978), S. 51–83

[Hogenauer 1981] HOGENAUER, Eugene: An economical class of digital filters for decimation and interpolation. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 29 (1981), April, Nr. 2, S. 155–162

[Humphrey] HUMPHREY, Robert: *playrec: Multi-channel Matlab Audio*. – URL http://www.playrec.co.uk. – [Online; Stand 24. Juni 2010]

[Kammeyer und Kroschel 2006] KAMMEYER, Karl-Dirk ; KROSCHEL, Kristian: *Digitale Signalverarbeitung - Filterung und Spektralanalyse mit MATLAB-Übungen*. 6. Teubner, 2006

[Karjalainen u. a. 2002] KARJALAINEN, Matti ; ANTSALO, Poju ; MÄKIVIRTA, Aki ; PELTONEN, Timo ; VÄLIMÄKI, Vesa: Estimation of Modal Decay Parameters from Noisy Response Measurements. In: *Journal of the Acoustical Engineering Society* Bd. 50, 2002, S. 867–878

[Lindau 2006] LINDAU, Alexander: *Ein Instrument zur softwaregestützten Messung binauraler Raumimpulsantworten in mehreren Freiheitsgraden*, Technische Universität Berlin, Magisterarbeit, 2006

[Majdak u. a. 2007] MAJDAK, Piotr ; BALAZS, Peter ; LABACK, Bernhard: Multiple Exponential Sweep Method for Fast Measurement of Head-Related Transfer Functions. In: *Journal of the Acoustical Engineering Society* Bd. 55, 2007, S. 623–637

[Makarski u. a. 2008] MAKARSKI, Michael ; GOERTZ, Anselm ; WEINZIERL, Stefan ; MOLDRZYK, Christoph: Zur Entwicklung von Lautsprechern für die Wellenfeldsynthese. In: *25. Tonmeistertagung – VDT international convention*, 2008

[Meyer 2006] MEYER, Martin: *Signalverarbeitung*. 4. Friedr. Vieweg & Sohn Verlag, 2006

[Müller 1999] MÜLLER, Swen: *Digitale Signalverarbeitung für Lautsprecher*, Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 1999

[Müller und Massarani 2001] MÜLLER, Swen ; MASSARANI, Paulo: Transfer-Function Measurement with Sweeps. In: *Journal of the Acoustical Engineering Society*, 2001, S. 443–471

[Oppenheim und Schafer 1999] OPPENHEIM, Alan V. ; SCHAFER, Ronald W.: *Discrete-Time Signal Processing*. 2. Prentice-Hall, Inc., 1999

[Oppenheim und Willsky 1996] OPPENHEIM, Alan V. ; WILLSKY, Alan S.: *Signals and Systems*. 2. Prentice-Hall, Inc., 1996

[Spors und Rabenstein 2006] SPORS, Sascha ; RABENSTEIN, Rudolf: Spatial Aliasing Artifacts Produced by Linear and Circular Loudspeaker Arrays used for Wave Field Synthesis. In: *120th Convention of the Audio Engineering Society*, 2006

[Weinzierl 2008] WEINZIERL, Stefan: *Handbuch der Audiotechnik*. Springer Berlin Heidelberg, 2008 (VDI-Buch)

[Zölzer 2005] ZÖLZER, Udo: *Digitale Audiosignalverarbeitung*. 3. Teubner, 2005

A. Dateien der Messsoftware

A.1. Konfigurationsdatei der Messsoftware

Alle Einstellungen der Messsoftware werden in der Datei *measurement.ini* im Ordner *iniFiles/* vorgenommen. Die Felder des *ini*-Files sind in Tabelle A.6 zu sehen.

Feld	Beschreibung
[measurement]	
type	Messtyp: *check, pre, reference, typical*
name	Frei wählbarer Name für die Messung
chansToMeasure	Zu messende Kanäle
globalSaveToPath	Ordner für die Messergebnisse
comment	Optionale Beschreibung
gainCheck	Für 0 wird Eingangskanal nicht auf Übersteuerung geprüft
	Sonst werden hier die Kanäle eingetragen
deleteFiles	0 oder 1. Für 0 werden Dateien mit Teilergebnisse behalten
[testDevice]	
name	Name des zu messenden JACK-Clients
inputs	Zu messende Kanäle
outputs	Mikrofoneingang
[detection]	
TFN	FFT-Länge: 2^{TFN}
smooth	Bruchteil einer Oktave zur Glättung der Frequenzgänge
	Für 0 findet keine Glättung statt
path	Ordner der Referenzwerte
refName	Name der Referenzdatei
memoryThreshold	Maximale Dateigröße in MByte
gateThreshold	Threshold des Gates in dB
bandFrequencies	Vektor mit Eckfrequenzen der Frequenzbänder

[sweep]	
type	Sweep-Typ: *specific, build, file*
amp	Maximale Amplitude in dB(Fs)
name	Name der Sweep-Dateien
maxChans	Maximale Anzahl paralleler Sweeps
tGap	Einzahlwert in s für die Gap-Zeit
	Pfad zur Datei, die Struct »tGap« enthält
runtimeFile	Pfad zur Datei, die Struct »runtime« enthält
path	Speicherort, der Sweep-Dateien
[reference]	
device	Name des JACK-Clients für Referenzmessung
file	Pfad zu Datei, die Struct »reference« enthält
input	Eingangskanal für Referenzmessung
measurement	0 Sweep wird als Referenz verwendet
	1 Referenzmessung wird durchgeführt
	-1 Datei wird geladen
output	Ausgangskanal für Referenzmessung
[JACK]	
driver	Treiber zu Starten des JACK-Audioservers
fs	Abtastrate
period	Buffergröße in Samples
inputDevice	Eingangs-Device, mit dem JACK initialisiert wird
name	Bezeichnung des Device, mit dem JACK initialisiert wird
nperiods	*nperiod* Parameter
outputDevice	Eingangs-Device, mit dem JACK initialisiert wird
realtime	0 oder 1. 1 für Realtime-Modus
realtime_priority	*realtime_priority* Parameter
path	Ordner der JACK-Programme
[systemDevice]	
name	Name, mit dem sich die Messsoftware bei JACK anmeldet
	PortAudio oder *octave*

Tabelle A.1.: *components/iniFiles/measurement.ini*

A.2. Datei zum Starten des Messvorgangs

Im Messsystemordner liegt die Datei *startMeasurement.m*. Über den Kommandozeilenaufruf *octave startMeasurement.m* innerhalb dieses Ordners wird eine Messung gestartet.

A.3. Dateien in components/_linux/, _macOs/ und _h104/

In diesen Ordnern befinden sich Dateien, die für den Ablauf der Messsoftware auf dem jeweiligen System benötigt werden. Tabelle A.2 zeigt die Dateien, die in allen Ordnern vorhanden sind, aber für die verschiedenen Systeme kompiliert wurden. Die Messsoftware ist auf GNU/Linux in der Lage die Einstellungen von RME Audio-

Datei	Beschreibung
playrec.mex*	Matlab, Octave Schnittstelle zum JACK-Audioserver
jack_check	Programm zur Bestimmung der Sample Rate und Buffer Size des JACK-Audioservers
jack_disconnect_all	Programm zum Abbau aller JACK-Verbindungen

Tabelle A.2.: *_linux*, *_macOs* und *_h104*

Hardware zu setzen. Hierfür sind Skripte und ein *ini*-File nötig, die in Tabelle A.3 aufgelistet sind.

Datei	Argument	Beschreibung
amixGetVolume.sh	Device ID	Bestimmt Verstärkung aller Kanäle
amixSetVolume.sh	Device ID	Setzt Verstärkung aller Kanäle
amixInputSelect.sh	Device ID	Bestimmt *Input Select* Status
amixMADILock.sh	Device ID	Bestimmt *MADI Lock* Status
amixPreferredSyncReference.sh	Device ID	Bestimmt *preferred sync reference*
amixSampleClockSource.sh	Device ID	Bestimmt *sample clock source*
amixSamplerate.sh	Device ID	Bestimmt *sample rate*
amixSyncReference.sh	Device ID	Bestimmt *autosync reference*
amixSystemClockMode.sh	Device ID	Bestimmt *system clock mode*
alsa.ini		Enthält Einstellungen für Messablauf

Tabelle A.3.: Dateien in *_linux* und *_h104* zur Steuerung der RME Audio-Hardware

A.4. Dateien in components/singleSweeps/

Zur Generierung der Sweep-Matrizen sind im Ordner *components/singleSweeps/* einzelne exponentielle Sweeps unterschiedlicher Länge und Abtastrate hinterlegt. Sie sind jeweils in einem *mat*-File gespeichert. Tabelle A.4 zeigt die vorhandenen Dateien.

Datei	Abtastrate	Länge
singleSweep_fs44100_N15.mat	44,1 kHz	2^{15} Samples
singleSweep_fs44100_N16.mat	44,1 kHz	2^{16} Samples
singleSweep_fs44100_N17.mat	44,1 kHz	2^{17} Samples
singleSweep_fs44100_N18.mat	44,1 kHz	2^{18} Samples
singleSweep_fs48000_N15.mat	48 kHz	2^{15} Samples
singleSweep_fs48000_N16.mat	48 kHz	2^{16} Samples
singleSweep_fs48000_N17.mat	48 kHz	2^{17} Samples
singleSweep_fs48000_N18.mat	48 kHz	2^{18} Samples

Tabelle A.4.: Dateien in *components/singleSweeps/*

A.5. Dateien in components/filterCoefficients/

Hier sind die Structs »coeff« in Form von *mat*-Files gespeichert, die die Filterkoeffizienten zur Simulation der potentiellen Defekte enthalten. Die Dateinamen haben die Form *b*_(Defektbezeichnung)*.mat*. In Tabelle A.5 ist der Aufbau von »coeff« gezeigt. Im Falle der WFS-Lautsprechermodule sind hier acht *mat*-Files zu finden.

Name	Feld	Funktion
coeff	b	Filterkoeffizienten
	fs	Abtastfrequenz
	error	Defektbezeichnung

Tabelle A.5.: Aufbau des Structs »coeff«

A.6. Dateien in components/sweeps/

Es wurden für das WFS-Lautsprechersystem im Raum H0104 spezifische »swp«-Structs berechnet. Sie sind unter den Namen *mesm_N16_amp-16_Fs44100_*.mat* im

Ordner *components/sweeps/* gespeichert. Die enthaltenen Sweep-Matrizen haben eine Abtastrate von 44,1 kHz, die Länge 2^{16} Samples und eine Amplitude von -16 dB(Fs).

A.7. Dateien in components/bandpasses/

Die während der Messung benötigten Bandpässe sind hier in Form von Frequenzgängen mit unterschiedlichen FFT-Längen und Abtastraten in jeweils einem *mat*-File gespeichert. Sie sind in Tabelle A.6 zu sehen.

Datei	Abtastrate	Länge
bandpass_N15_44100.mat	44,1 kHz	2^{15} Samples
bandpass_N16_44100.mat	44,1 kHz	2^{16} Samples
bandpass_N17_44100.mat	44,1 kHz	2^{17} Samples
bandpass_N18_44100.mat	44,1 kHz	2^{18} Samples
bandpass_N15_48000.mat	48 kHz	2^{15} Samples
bandpass_N16_48000.mat	48 kHz	2^{16} Samples
bandpass_N17_48000.mat	48 kHz	2^{17} Samples
bandpass_N18_48000.mat	48 kHz	2^{18} Samples

Tabelle A.6.: Dateien in *components/bandpasses/*

A.8. Dateien in components/references/

Für das WFS-System im Raum H0104 wurden bereits die Referenzfrequenzgänge nebst Referenzmaximalamplituden und die zur Generierung der Sweep-Matrizen benötigten Parameter mittels akustischer Referenzmessung gewonnen. Die entsprechenden Dateien sind im Ordner *components/references/* zu finden und in Tabelle A.7 verzeichnet.

Datei	Inhalt
reference_h0104_refMaxAmp.mat	Struct »refMaxAmp«
reference_h0104_refTFs.mat	Struct »refTFs«
reference_h0104_runtime.mat	Struct »runtime«
reference_h0104_tGap_N16_lundeby3dB.mat	Struct »tGap«

Tabelle A.7.: Dateien in *components/references/*

A.9. Dateien in components/m-files/

Unter *components/m-files/* sind die *m*-Files abgelegt, aus denen sich die Messsoftware zusammensetzt. Es folgt eine Auflistung dieser Dateien.

```
STATUS = AMIXCHECKANDSET(DEVICE,VERBOSE,BENDER) checks if the audio interface
is running with the parameters defined in the global variable SETUP.
DEVICE            ID number of device according to amixer
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
BENDER            0 or 1. if 1 the audio interface on bender is checked
STATUS            0 or 1. if 0 an error occured
dependencies:     'amixer', 'amixPreferredSyncReference.sh',
                  'amixInputSelect.sh', 'amixSampleClockSource.sh'
global variables: 'setup', 'compPath'

STATUS=AMIXCHECKVOL(ZERODB,COMPUTER) checks if all amixer channels have the
value of argument ZERODB.
ZERODB     32768 is equal to 0dB
COMPUTER   0 for local computer, 1 for bender and 2 for nodes
STATUS:    0 or 1. if 0 an error occured

STATUS = AMIXSETVOL(DEVICE,ZERODB,BENDER) sets the volume of all amixer
channels to value of argument ZERODB.
DEVICE              number of device according to amixer
ZERODB              value for which amixer is tested. 32768 is equal to 0dB
BENDER              0 or 1. if 1 the bender's audio interface is checked
STATUS              0 or 1. if 0 an error occured
dependencies:       'amixSetVolume.sh', 'amixCheckVol.m'
global variables:   'setup', 'compPath'

STATUS = AMIXNODESINIT(VERBOSE) initializes the audio interfaces on nodes.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'nodesSetVolumeToZeroDb.sh', 'amixCheckVol.m'
global variables: 'setup', 'compPath'
```

```
[STATUS FS] = AMIXINIT(VERBOSE) initializes the audio interfaces of all
involved computers if they are equipped with RME audio hardware.
VERBOSE            'verbose' or []. if 'verbose' informations are displayed
STATUS             0 or 1. if 0 an error occured
FS                 detected sample rate
dependencies:      'amixCheckAndSet.m', 'amixCheckVol.m', 'amixSetVol.m',
                   'amixNodesInit.m', 'amixGetSampleRate.m'
global variables:  'setup', 'compPath'

FS = AMIXGETSAMPLERATE(DEVICE,BENDER) detects audio interface's sample rate.
DEVICE            number of device according to amixer
BENDER            0 or 1. if 1 bender's interface is checked
FS                detected sample rate
dependencies:     'amixSamplerate.sh'
global variables: 'setup', 'compPath'

[ANALYSE] = ANALYSEIT(VERBOSE,BANDFREQUENCIESFITTING) compares the measured
frequency responses saved in the TFs struct with their reference frequency
responses saved in refTFs and detects if a loudspeaker channel is defective.
VERBOSE                 'verbose' or []. if set 'verbose' informations
                         will be displayed
BANDFREQUENCIESFITTING   matrix that defines frequency bands to
                         perform curve fitting
ANALYSE                  struct that contains all informations of the analysis
function dependencies:   'onOrOff.m', 'getMSE.m', 'fitCurve.m', 'decideIt.m'
global variables:        'setup'

TFS = BANDPASSTFS(TFS,INDB) filters a frequency response saved in struct TFs
with a bandpass saved in bandpass_N*_*.mat located in components/bandpasses/.
TFS        struct that contains frequency response and additional informations
INDB       0 or 1. if 1 frequency response is given in dB.
TFS        filtered frequency response
global variables:        'setup'
```

```
[] = ANALYSEPRESTRUCTS(DOREFTFS,DOREFMAXAMP,DORUNTIME,DOFILTERING,DOREVTIME,
DODISTTIME) calculates data that is necessary for overlapping method.
DOREFTFS          0 or 1. if 1 refTFs is calculated
DOREFMAXAMP       0 or 1. if 1 refMaxAmp calculated
DORUNTIME         0 or 1. if 1 acoustical runtimes are calculated
DOFILTERING       0 or 1. if 1 calculations for terz-bands
DOREVTIME         0 or 1. if 1 reverberation times are calculated
DODISTTIME        0 or 1. if 1 highest HIR's durations are calculated
global variable:  'setup'

CHS = CHAN2NODE(CHANNELS) converts given CHANNELS to a struct CHS that
contains node and channel informations.

DEVICESTATUS = DEVICESCHECK(VERBOSE) acertains if all devices defined in
global variable SETUP are set up.
VERBOSE            'verbose' or []. if 'verbose' informations are displayed
DEVICESTATUS       0 or 1. if 0 an error occured
dependencies:      'JACKGetPorts.m'
global variables:  'setup'

[] = DECONVOLUTEIT(DELETETEMP, KEEPTF, SAVEPREIR,VERBOSE) performs de-
convolution with the loudspeakers' responses to the sweep the electric
reference measurement.
DELETETEMP        0 or 1. if 1 temp-files are deleted
KEEPTF            0 or 1. if 0 TF field in the struct mess is deleted
SAVEPREIR         0 or 1. if 1 resulting impulse response is not trimmed
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variables: 'setup'

ERROR = DECIDEIT(MSE,REFERRORS) detects if a loudspeaker channel is defective
by comparing a given MSE-vector with reference MSE-vectors
MSE           MSE-vector
REFERRORS     struct that contains reference MSE-vectors
ERROR         detected error
dependencies: 'getMSE.m'
```

```
Y = DBSUM(X) calculates sum of dB values stored in vector X.

L = dBFs(x) caluclates dB(Fs) value of amplitude X.

FILENAME = ERRORS2TXT() reads the file detectedErrors.mat and outputs content
as txt-file named FILENAME.
global variables: 'setup'

Y = FRACFILTER(X,FS,FRAC) applies fractional octave band smoothing on
multichannel spectrum.
X              spectrum
FS             sample rate
FRAC           fraction of octave for smoothing
Y              smoothed spectrum
dependencies:  'fract_oct_smooth.m'

STATUS = FOLDERCLEANUP(DELETETEMPS, VERBOSE) deletes files created during
measurement process depending on setup.type and DELETETEMPS.
DELETETEMPS      0 or 1 to delete files.
VERBOSE          'verbose' or []. if 'verbose' informations are displayed
STATUS           0 or 1. if 0 an error occured
global variable: 'setup'

[STATUS EXISTINGFILES] = FOLDERCHECK(FOLDER) looks if a given FOLDER exists and
checks its content.
FOLDER         folder to be checked
STATUS         if -1 folder doesn't exist, if 0 folder is empty and if 1 not
EXISTINGFILES  files in folder

FITTEDCURVE = FITCURVE(CURVE, REFCURVE, FS, FREQS, STEPSIZE) fits a CURVE to a
reference curve by calculating the smallest MSE value.
CURVE          curve to be fitted to reference curve
REFCURVE       reference curve
FS             sample rate
FREQS          matrix that defines frequency bands to perform curve fitting
STEPSIZE       initial step value
FITTEDCURVE    fitted curve
```

```
dependencies:  'getMSE.m'

[NOISEFLOOR PEAKINDEX] = FINDNOISEFLOOR(IR,N,THRESHOLD,NOISETHRESH,VERBOSE)
detects the noisefloor of impulse response.
IR            impulse response
N             time window length in samples to calculate RMS of signal
THRESHOLD     in dB. if difference of peak value and minimum of IR's RMS is
              below noiseThresh noisefloor isn't calculated.
              instead all outputs will be -1
NOISETHRESH   in dB. this function defines the noisefloor as the area around
              the minimum of the IR's RMS that is below the noiseThresh
NOISEFLOOR    part of signal detected as noisefloor
PEAKINDEX     IR's peak position represented as sample
dependencies: 'RMS.m', 'getIRStart.m'

BANDS = FILTERBANK(SIG,FS,TYPE) filters signal with digital one-third-octave
or octave filters.
SIG           signal to be filtered
FS            sample rate
TYPE          'terz' or 'octave'
BANDS         struct that contains results
dependencies: 'oct3dsgn.m'

NOERRORCHANS = GETWORKINGCHANS(ANALYSEFILE, VERBOSE) looks for working channels
in struct analyse and writes them into vector NOERRORCHANS
ANALYSEFILE        analyse struct
VERBOSE            'verbose' or []. if 'verbose' informations are displayed
NOERRORCHANS       vector that contains working channels
global varaiables: 'setup'

[] = GETTFS(FRAC,INTER,INDB,ISREF,VERBOSE) calculates a FFT of all IRs saved
in setup.saveToPath. resulting frequency responses are saved in struct TFs.
FRAC          spectra are smoothed with fraction frac. no smoothing for frac=0
INTER         0 or 1. if 1 spectra are converted to integer values
INDB          0 or 1. if 1 spectra are saved in dB.
VERBOSE       if set to 'verbose' status messages are displayed
global variables: 'setup'
```

```
RUNTIME = GETRUNTIMESTRUCT(IRS,PRE,N,PLOTTEN) creates a struct that contains
the acoustical runtimes of all impulse responses stored in struct IRs.
IRS            struct that contains impulse responses
PRE            'pre' or ''. if 'pre' IRs must be a pre struct
N              window size of RMS calculation
PLOTTEN        0 or 1. if 1 results are plotted
RUNTIME        struct that contains acoustical runtimes
dependencies:  'getIRStart.m'

[REVTIME THRESHOLDABS NOISERMS PEAKVALUE SNR T REVINDEX] = GETREVTIMETHRESH
(IR,FS,N,THRESHOLD,PLOTTITLE) detects an impulse response's reverberation time.
IR                impulse response
FS                sample rate
N                 time window length in samples to calculate RMS of a signal
NOISERMSOFFSET    in dB. an offset is added to the RMS value of the noisefloor
PLOTTITLE         as string. a given plotTitle will lead to a plot
REVTIME           reverberation time in seconds
NOISERMS          IR's RMS value of its noisefloor
PEAKVALUE         value of IR's peak
SNR               signal to noise ratio. defined as ratio of IR's peak
                  value and its noisefloor's RMS value
T                 time vector in seconds. 0s is position of IR's peak
REVINDEX          represented as sample.
dependencies:     'lundebyRevTime.m', 'getIRStart.m', 'RMS.m', 'dBFs.m'

[REVTIMES] = GETREVTIMESTRUCT(BANDS,N,THRESHKIND,THRESHOROFFSET,PLOTTEN)
calculates the reverberation times of the filtered IR supplied by the
struct bands.
BANDS             struct that contains filtered impulse response
N                 time window length in samples to calculate RMS of signal
THRESHKIND        'thresh' for threshold or 'noise' for noisefloor
THRESHOROFFSET    in dB. if threshKind = 'thresh' it's the threshold. if
                  'noise' an offset can be added to the RMS value of the
                  noisefloor. the RMS value of the noisefloor is used to
                  detect the reverberation time. see getRevTime.m
PLOTTEN           if 1 informations of the band with the highest
```

```
                  reverberation time will be plotted
REVTIMES          struct that contains all band informations like
                  fs          sample rate
                  fc          filter's center frequency
                  seconds     bands' reverberation times
                  noiseRMS    RMS value of band's noisefloor
                  peakValue   value of band's peak
                  SNR         band's signal to noise ratio
dependencies:     'lundebyRevTime.m', 'getRevTime.m', 'getRevTimeThresh.m',
                  'plotTerzBand.m'

[REVTIME NOISERMS PEAKVALUE SNR T REVINDEX] = GETREVTIME(IR,FS,N,
NOISERMSOFFSET, PLOTTITLE) calculates an impulse response's reverberation time.
it is defined as the time the IR's RMS reaches the RMS value of its noisefloor.
IR                impulse response
FS                sample rate
N                 time window length in samples to calculate RMS of a signal
NOISERMSOFFSET    in dB. an offset can be added to the noisefloor's RMS value.
PLOTTITLE         as string. a given plotTitle will lead to a plot
REVTIME           reverberation time in seconds
NOISERMS          IR's RMS value of its noisefloor
PEAKVALUE         value of IR's peak
SNR               signal to noise ratio. defined as ratio of IR's peak
                  value and its noisefloor's RMS value
T                 time vector in seconds. 0s is position of IR's peak
REVINDEX          represented as sample.
dependencies:     'findNoisefloor.m', 'RMS.m'

REFERRORS = GETREFERRORS(FS,N,BANDFREQUENCIES,THRESHOLD) creates the struct
REFERRORS that includes the MSE vector of each possible loudspeaker channel
error given by filter coefficients saved in 'components/filterCoefficients/'.
the MSE vector's length depends on the number of frequency bands defined in
BANDFREQUENCIES. if a MSE value exeeds THRESHOLD it will be set to THRESHOLD.
FS and N are needed to perform the FFT of the filter coefficients.
dependencies: 'bandpassIt.m', 'getMSE.m'
```

```
STATUS = GETREFERENCE(VERBOSE) creates a reference file.
it's saved in setup.saveToPath.
if setup.reference.measurement is 1 a measurement will be done
if setup.reference.measurement is -1 a reference file will be loaded
if setup.reference.measurement is 0 the singleSweep is taken as reference
the system latency is then added
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'referenceMeasurement.m'
global variables: 'setup'

MSE = GETMSE(REF,X,LOWERTHRESHOLD) calculates the mean squared error of
functions REF and X. if the absolute value of the difference of REF(k) and X(k)
exeeds LOWERTHRESHOLD it will be set 0.

MAXAMP = GETMAXAMP(X) detects highest amplitude MAXAMP of X in dB(Fs).

[] = GETLEVELDIFFERENCE(THRESHOLD,MAXAMP,REFMAXAMP,VERBOSE) calculates
level difference of amplitudes stored in MAXAMP and REFMAXAMP saves it
if it exceeds THRESHOLD.
global variable: 'setup'

IRSTART = GETIRSTART(IR,N,PLOTTEN,FS) detects an impulse response's onset.
IR                impulse response
N                 window size for RMS calculation
PLOTTEN           0 or 1. if 1 results are plotted
FS                sample rate to perform plot
IRSTART           onset in samples
dependencies:     'dBFs.m', 'RMS.m'

[] = GETIRS(NORM,DELETETEMPS,ISREF,VERBOSE) extracts all impulse responses by
windowing deconvolution result that is stored in struct mess. IRs are saved
in struct IRs. every IR is analysed to check if channel is working at all.
a struct will be created that contains the maximum amplitudes of each IR. if a
channel is not working the amplitude is set -1.
NORM          0 or 1. if 1 impulse responses are normalised
DELETETEMPS   0 or 1. if 1 all temp files are deleted after calculation
```

```
ISREF         0 or 1. if 1 the struct maxAmp is renamed to refMaxAmp
VERBOSE       'verbose' or []. if 'verbose' informations are displayed
dependencies: 'isChannelWorking.m'

ERRORMATRIX = GETERRORMATRIX(CHANNELS,PERCENT,ERRORS,PLOTTEN) is needed
evaluating the measurement system. it creates a error vector that contains
a status for all given loudspeaker channels. the number of errors depends
on a given percentage.
CHANNELS           vector that contains channels
PERCENT            percentage of number of occuring errors
ERRORS             all possible error values
PLOTTEN            0 or 1. if 1 results will be plotted
ERRORMATRIX        vector with the length of channel vector. every index
                   stands for an error status

DISTTIMES = GETDISTTIMESTRUCT(BANDS,THRESHORNOISE,THRESHOLD,PLOTTEN) detects
the durations of the highest HIRs of the filtered deconvolution results stored
in struct BANDS depending on a given threshold.
BANDS         struct that contains filtered deconvolution results
THRESHORNOISE 'thresh' or 'noise'
THRESHOLD     sets threshold to detect duration if 'thresh' mode
              sets an offset if 'noise' mode
PLOTTEN       0 or 1. if 1 results will be plotted
DISTTIMES     struct that contains results
dependencies: 'lundebyRevTime.m', 'findNoisefloor.m', 'dBFs.m', 'getDistTime.m'

DISTTIME = GETDISTTIME(IR,FS,THRESHOLD,N,PLOTTEN) detects the duration
of the highest HIR of the deconvolution result IR depending THRESHOLD.
IR            deconvolution result in time domain
FS            sample rate
THRESHOLD     threshold to detect duration
N             window size for RMS calculation
PLOTTEN       0 or 1. if 1 results will be plotted
DISTTIME      duration of highest HIR
dependencies: 'getIRStart.m', 'dBFs.m', 'RMS.m', 'decideIt.m'

DIFFERENCE = GETDISTANCE(REF,X) calculates the sum of differences of REF and X.
```

```
[] = GETDETECTEDERRORS(VERBOSE) checks the struct analyse if loudspeaker
channel errors were detected. in this case it writes the channels' informations
into struct DETECTEDERRORS and saves it as *_detectedErrors.mat.
VERBOSE          'verbose' or []. if 'verbose' informations are displayed
global variable: 'setup'

DELAYS = GETDELAYS(CHANNELS,RUNTIME,TGAP) calculates the compensated delay
times for the mesm sweep synthesis of given channels.
channels          channels to be excited by sweeps
runtime           struct created by getRuntimeStruct.m
tGap              delay time that has to be compensated
delays            struct that contains the compensated delay times
global variables: 'setup'

STATUS = GAINCHECK(VERBOSE) performs measurements of the channels given by
setup.gainCheckChans and checks if the level of incomming audio doesn't
overload the input depending on how many sweeps are played parallel because
of the overlapping method. channels will be measured sequentially with the
sweep saved in *gainCheckSweep.mat.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'whatThreshold.m', 'playAndRecord.m', 'JACKConnect.m',
                  'JACKConnectNode.m', 'JACKDisconnect.m', 'chan2node.m',
                  'JACKDisconnectNode.m'
global variables: 'setup'

PRE = ISPREWORKING(PRE,THRESHOLD) checks if a channel is working at all.
the RMS of a given deconvolution result is calculated. if the difference
of its maximum and minimum values is beneath a threshold the channel is
detected as not working.
PRE             deconvolution result in time domain
THRESHOLD       threshold for detection
STATUS          0 or 1. if 0 channel is not working
dependencies:   'RMS.m'
```

```
STATUS = ISCHANNELWORKING(IR,THRESHOLD) checks if a channel is working at
all. the RMS of a given impulse response is calculated. if the difference
of its maximum and minimum values is beneath a threshold the channel is
detected as not working.
IR            impulse response
THRESHOLD     threshold for detection
STATUS        0 or 1. if 0 channel is not working
dependencies: 'RMS.m'

[] = IRS2WAVS(IRS,BIT,VERBOSE) saves the impulse responses stored in struct
IRs as wav-files.
IRS               struct that contains impulse responses
BIT               16, 24 or 32 bit are possible
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variables: 'setup'

STATUS = JACKINIT(VERBOSE) checks if JACK is running with the parameters
defined in struct SETUP. if not it kills JACK and starts it with the right
settings.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'JACKCheck.m', 'JACKStart.m', 'JACKStop.m'

[STATUS DEVICESCHECK] = JACKCHECK(VERBOSE) verifies if JACK is running
with all devices that are defined in struct SETUP.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
DEVICECHECK       status of used devices. 0 or 1. if 0 an error occured
dependencies:     'JACKStatus.m', 'JACKGetPorts.m'
global variables: 'setup'

[] = JACKSTART(VERBOSE) starts JACK with settings defined in struct SETUP.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variables: 'setup'

[] = JACKSTOP(VERBOSE) stops JACK.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
```

```
[STATUS FS BUFFERSIZE] = JACKSTATUS(BENDER) checks if JACK is running.
BENDER              0 or 1. if 1 JACK on bender is checked
STATUS              0 or 1. if 1 an error occured
FS                  JACK's sample rate
BUFFERSIZE          JACK's buffer size
global variables:   'setup', 'compPath'

TREE = JACKGETPORTS(BENDER) detects JACK clients.
BENDER              0 or 1. if 1 JACK on bender is checked
TREE(device_index)                                  device infos
TREE(device_index).outputs{output_index}            outputs of device
TREE(device_index).outputsLong{outputLong_index}    device:output
TREE(device_index).inputs{input_index}              inputs of device
TREE(device_index).inputsLong{inputLong_index}      device:input
global variables:   'setup'

[AVAILABLEDEVICES] = JACKAVAILABLEDEVICES(DRIVER) checks which devices
are available to start JACK depending on a given driver.
DRIVER              'alsa', 'coreaudio' etc.
AVAILABLEDEVICES    available devices
global variables:   'setup'

[] = JACKCONNECT(OUTDEVICE,INDEVICE, OUTCHANS,INCHANS,VERBOSE) sets up
JACK connections.
OUTDEVICE          JACK output device
INDEVICE           JACK input device
OUTCHANS           JACK output channels
INCHANS            JACK input channels
VERBOSE            'verbose' or []. if 'verbose' informations are displayed
dependencies:      'JACKGetPorts.m'
global variables: 'setup'

[] = JACKDISCONNECT(OUTDEVICE,INDEVICE, OUTCHANS,INCHANS,VERBOSE)
releases JACK connections.
OUTDEVICE          JACK output device
INDEVICE           JACK input device
```

```
OUTCHANS          JACK output channels
INCHANS           JACK input channels
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
dependencies:     'JACKGetPorts.m'
global variables: 'setup'

[] = JACKDISCONNECTALL(VERBOSE) disconnects all JACK connections
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variables: 'compPath'

[STATUS] = JACKBENDERINIT(VERBOSE) checks if the JACK audio server is
running on bender with parameters provided by struct SETUP. if JACK is
running with wrong parameters JACK will be stopped and started again with
right settings.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'readJACKInitFile.m', 'JACKBenderCheck.m',
                  'JACKBenderStart.m', 'JACKBenderStop.m'

[STATUS DEVICECHECK] = JACKBENDERCHECK(VERBOSE) verifies if JACK is
running on Bender with devices defined in struct SETUP.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
DEVICECHECK       status of used devices. 0 or 1. if 0 an error occured
dependencies:     'JACKStatus.m', 'JACKGetPorts.m'
global variables: 'setup'

STATUS = JACKBENDERSTART(VERBOSE) starts JACK audio server on bender.
VERBOSE           'verbose' or []. if set 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
global variables: 'setup'

[] = JACKBENDERSTOP(VERBOSE) stops JACK audio server on bender.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variables: 'setup'
```

```
[] = JACKBENDERCONNECT(OUT,IN,VERBOSE) sets up JACK connections on bender.
OUT                 input channels
IN                  output channels
VERBOSE             'verbose' or []. if 'verbose' informations are displayed
dependencies:       'JACKStatus.m'
global variables: 'setup'

[STATUS] = JACKNODESINIT(VERBOSE) checks if JACK is running on nodes with
parameters defined in struct SETUP. if not it stops JACK and starts it
with right settings.
VERBOSE             'verbose' or []. if 'verbose' informations are displayed
STATUS              0 or 1. if 0 an error occured
dependencies:       'JACKNodesCheck.m', 'JACKNodesStart.m', 'JACKNodesStop.m'

STATUS = JACKNODESSTART(VERBOSE) starts JACK on nodes with parameters
defined in struct SETUP.
VERBOSE             'verbose' or []. if 'verbose' informations are displayed
STATUS              0 or 1. if 0 an error occured
global variables: 'setup'

STATUS = JACKNODESSTOP(VERBOSE) stops JACK on nodes.
VERBOSE             'verbose' or []. if 'verbose' informations are displayed
STATUS              0 or 1. if 0 an error occured
global variables: 'setup'

STATUS = JACKNODESCHECK() checks if JACK is running on all nodes.
STATUS     0 or 1. if 1 JACK is running on all nodes

[] = JACKSINGLENODECONNECT(NODE,OUT,IN) connects an input and an
output on a node computer.
NODE          1-15. node
OUT           output channel
IN            input channel

[] = JACKSINGLENODEDISCONNECT(NODE,OUT,IN) disconnects an input and an
output on a node computer.
NODE          1-15. node
```

```
OUT          output channel
IN           input channel

[] = JACKCONNECTNODE(OUT,IN) sets up JACK connections on node computers
of the WFS-system in room H0104.
OUT               output channels
IN                struct with node and input channel informations
dependencies:     'JACKSingleNodeConnect.m'

[] = JACKDISCONNECTNODE(OUT,IN) releases JACK connections on node computers
of the WFS-system in room H0104.
OUT               output channels
IN                struct with node and input channel informations
dependencies:     'JACKSingleNodeDisconnect.m'

[] = JACKCONSTANTCONNECTIONS(VERBOSE) sets up constant JACK connections
of the measurement system. in general the microphone channel is connected
with the system's input.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
dependencies:     'JACKConnect.m', 'JACKBenderConnect.m'

[REVTIME NOISE SNR] = LUNDEBYREVTIME(IR,FS,NOISEOFFSET,PLOTTEN) calculates
reverberation time of an impulse response according to the lundeby method.
IR                impulse response
FS                sample rate
N                 time window length in samples to calculate RMS of a signal
NOISERMSOFFSET    in dB. an offset can be added to the noisefloor's RMS value
PLOTTEN           0 or 1. if 1 results are plotted
REVTIME           reverberation time in seconds
NOISE             RMS value of noisefloor
SNR               signal to noise ratio. defined as ratio of IR's peak
                  value and its noisefloor's RMS value
dependencies:     'linearRegression.m', 'getIRStart.m', 'RMS.m', 'dBFs.m'

[A B] = LINEARREGRESSION(Y) performs a linear regression with given
points in vector Y. a solution for f(x;a,b) := ax + b is calculated.
```

```
X = MAKECOL(X) changes X to a column vector.

X = MAKEROW(X) changes X to a row vector.

[] = MEASUREMENT(VERBOSE) controls measurement operation depending on which
setup.type and setup.sweep.type is defined. the results are stored in struct
MESS and saved as '*_temp_*.mat'.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
dependencies:     'chan2node.m', 'JACKConnectNode', 'JACKConnect',
                  'JACKDisconnectNode', 'JACKDisonnect', 'playAndRecord'
global variables: 'setup'

SIGNAL = NORMALIZE(X,TOWHAT) normalizes a signal to a given value.
X         signal
TOWHAT    highest amplitude in dB
SIGNAL    normalized signal

CHANNELS = NODE2CHAN(NODES, NODECHANNELS) calculates CHANNELS for given
NODES and NODECHANNELS.

[] = NORMALIZEIT(GATE,GATETHRESHOLD,STEPSIZE,VERBOSE) normalizes impulse
responses in structs IRs by comparing curve of maximum amplitudes stored
in struct maxAmp and refMaxAmp.
GATE              0, no gate. 1, gate with gateThreshold
GATETHRESHOLD     gate's threshold
STEPSIZE          initial step size for curve fitting process
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
dependencies:     'getMSE.m'
global variables: 'setup'

[] = PROGRESSBAR(TIME,BEATSPERSECOND) displays a progress bar to look at
while something calculations are paused.
TIME              duration in seconds
BEATSPERSECOND    dots to be displayed per second
```

```
[REFTFS,F] = PRE2REFTFS(PRE,N,FRAC,INTER,INDB) calculates frequency response
of a deconvolution result stored in struct PRE.
PRE              struct that contains result of deconvolution operation
N                FFT length: 2^N
FRAC             fraction of octave in case octave-width smoothing
                 is chosen. if frac is 0 no smoothing is processed
INTER            0 or 1. if 1 results are casted to 16 bit
INDB             0 or 1. if 1 the resukting frequency response is in dB
REFTFS           struct that contains results
F                frequency vector to plot frequency responses
dependencies:    'fract_oct_smooth.m'

REFMAXAMP = PRE2REFMAXAMP(PRE) calculates the maximum amplitude of a
deconvolution result stored in struct PRE.
PRE              struct that contains result of deconvolution operation
REFMAXAMP        struct that contains results
dependencies:    'dBFs.m', 'RMS.m'

[] = PLOTTERZBAND(TERZ, INDEX,N) plots the informations in struct TERZ
created with the function getRevTimeTerzBands.m.
TERZ      struct
INDEX     index of struct to plot
N         time window length in samples to calculate RMS of signal

[] = PLAYRECSTOP(VERBOSE) stops playrec
VERBOSE       'verbose' or []. if 'verbose' informations are displayed
dependencies: 'playrec.mex'

STATUS = PLAYRECSTART(VERBOSE) initializes playrec with parameters defined
in struct SETUP.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'playrec.mex', 'playrecSelectDevice.m'
global variables: 'setup', 'compPath'
```

```
[DEVICEID, INCHAN, OUTCHAN, FS, STATUS] = PLAYRECSELECTDEVICE(DEVICE) gets
informations about a given playrec device.
DEVICE         device to check
DEVICEID       device's playrec ID
INCHAN         number of input channels
OUTCHAN        number of output channels
FS             sample rate
STATUS         0 or 1. if 0 an error occured
dependencies:  'playrec.mex'

STATUS = PLAYRECINIT(VERBOSE) checks if playrec is running with parameters
set in struct SETUP. if not it kills playrec and starts it with right settings.
VERBOSE        'verbose' or []. if 'verbose' informations are displayed
STATUS         0 or 1. if 0 an error occured
dependencies:  'playrecCheck.m'

PLAYRECSTATUS = PLAYRECCHECK(VERBOSE) checks if playrec is already initialized.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
PLAYRECSTATUS     0 or 1. if 0 an error occured
dependencies:     'playrec.mex'

X = PLAYANDRECORD(OUTCHANNELS,INCHANNELS,SWEEP) sends out a signal to given
output channels and simultanously records incoming audio from input channels.
OUTCHANNELS       output channels
INCHANNELS        input channels
SWEEP             signal to be played
X                 recorded audio signal
dependencies:     'playrec.mex'

Y = RMS(X,N) devides a signal into segments and calculates their RMS values.
x        signal
N        window length in samples to calculate RMS of signal
Y        RMS vector of x. RMS's length corresponds to x's length.
         if N is not given RMS's length is 1
```

```
[STATUS, RECORDING] = REFERENCEMEASUREMENT(VERBOSE) performs electric reference
measurement
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
RECORDING         recorded audio signal
dependencies:     'JACKConnect.m', 'JACKDisconnect.m', 'playAndRecord.m'

SETUP = READINITFILE(INITFILE) reads INITFILE file and writes parameters
into struct SETUP.

ALSA = READALSAINITFILE(INITFILE) reads INITFILE file and writes parameters
into struct ALSA.

[] = READSYSTEMINITFILE(INITFILE) reads INITFILE file and writes parameters
into global variable SETUP.

JACK = READJACKINITFILE(INITFILE) reads INITFILE file and writes parameters
into struct JACK.

[] = SAVETOINIT(VERBOSE) creates folder into which all files of measurement
will be saved to. folder name is defined in struct SETUP. if it already exist
a new folder will be created by consecutively numbering the given folder name.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
global variable: 'setup'

STATUS = SYSTEMINIT(VERBOSE) initializes audio components of the measurement
system. if OS is linux and RME audio interface is used the audio interface's
parameter are set up, JACK is started and payrec.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 a component couldn't been started
dependencies:     'amixInit.m', 'readSystemInitFile.m', 'JACKBenderInit.m',
                  'JACKInit.m', 'JACKNodesInit.m', 'playrecInit.m',
                  'JACKDisconnectAll.m','devicesCheck.m'
global variables: 'setup', 'compPath'
```

```
STATUS = SYSTEMSTOP(VERBOSE) stops playrec and JACK.
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
STATUS            0 or 1. if 0 an error occured
dependencies:     'playrec.mex', 'playrecStop.m', 'playrecCheck.m',
                  'JACKBenderStop.m', 'JACKNodesCheck.m', 'JACKNodesStop.m'
global variable:  'setup'

[] = SWEEPINIT(VERBOSE) checks an existing struct SWP or builds one depending
on given parameters in struct SETUP.
NORM              0 or 1. if 1 impulse responses are normalised
DELETETEMPS       0 or 1. if 1 all temp files are deleted after calculation
ISREF             0 or 1. if 1 the struct maxAmp is renamed to refMaxAmp
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
dependencies:     'isChannelWorking.m'
global variables: 'setup'

SINGLESWP = SWEEPSTRUCTINIT(CHANNELS,TGAP,AMP,RUNTIME,ERRORS,MAXCHANS,
SAVETOPATH,N,FS,VERBOSE) creates sweep matrices depending on given parameters,
puts them into swp structs and saves them as .mat-files.
CHANNEL           vector that contains all channels
TGAP              value or a tGap struct that contains gap times
AMP               amplitude of sweeps in dB(Fs)
RUNTIME           value or runtime struct
ERRORS            for testing the measurement system. vector that
                  contains numbers representing errors
MAXCHANS          maximum number of parallel sweeps
SAVETOPATH        path where all swp structs are saved in
N                 sweep length: 2^n
FS                sample rate
VERBOSE           'verbose' or []. if 'verbose' informations are displayed
SINGLESWP         single sweep that is used for building matrices
dependencies:     'normalize.m', 'getErrorMatrix.m', 'getDelays.m',
                  'sweepCreateStruct.m'
global variables: 'setup'
```

```
[SWP, SWEEP] = SWEEPCREATESTRUCT(SWEEP,CHANNELS,DELAYS,AMP,ERRORFIELD)
creates sweep matrix depending on given parameters.
CHANNELS          vector that contains all channels
DELAYS            struct that contains delay times
AMP               amplitude of sweeps in dB(Fs)
ERRORFIELD        vector that contains all channels' stati
SWP               struct that contains sweep matrix
SWEEP             single sweep that is used for building matrix
dependencies:     'normalize.m'

THRESHOLD = WHATTHRESHOLD(SWEEP,MAXCHANS,TGAP,FS) calculates upper threshold
for gain check measurement.
SWEEP            single sweep that was used to create sweep matrix
MAXCHANS         number of parallel sweeps
TGAP             shortest possible delay between sweeps
FS               sample rate
THRESHOLD        upper threshold
dependencies     'dBSum.m', 'getMaxAmp.m'
```