Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Thorben Vornholz

Ein VST–Plugin zur adaptiven Geräuschunterdrückung in einem Arbeitsraum

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thorben Vornholz

Ein VST–Plugin zur adaptiven Geräuschunterdrückung in einem Arbeitsraum

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Wolfgang Fohl
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Eingereicht am: 4. Oktober 2012

Thorben Vornholz

Thema der Arbeit
Ein VST–Plugin zur adaptiven Geräuschunterdrückung in einem Arbeitsraum

Stichworte
VST–Plugin, VST, Virtual Studio Technology, Geräuschunterdrückung, Geräuschkompensation, adaptives Filter

Kurzzusammenfassung
Diese Bachelorarbeit befasst sich mit dem Thema der aktiven Geräuschunterdrückung. Ziel soll es sein, ein VST3–Plug-in zu entwickeln, welches durch adaptive Filterung Störgeräusche gezielt verringert. Notwendig dafür ist sowohl die Aufnahme des Stör- als auch des Fehlersignals, um entsprechenden Schall zu erzeugen, der das Störsignal im Raum kompensiert. Diese Arbeit bietet einen Einblick in die theoretische Filterleistung von einem adaptiven Filtersystem mit blockweiser Verarbeitung.

Des Weiteren wird anhand von praktischen Beispielen gezeigt, inwieweit Schall in einem Raum durch Überlagerung gedämpft werden kann.

Thorben Vornholz

Title of the paper
VST Plug-in for adaptive noise suppression in a working room

Keywords
VST Plug-in, VST, Virtual Studio Technology, noise suppression, noise compensation, adaptive filter

Abstract
This Bachelor thesis studies the topic of adaptive noise suppression and interference in a particular area. The main goal is the design of an VST Plug-in that is able to decrease interference by using an adaptive filter. The recording of both noise and error signal is necessary in order to produce an acoustic noise that compensates error signals inside a room.

This thesis also provides an insight into sound interference. Furthermore, it exemplifies to what extent interference can absorb noise in a room.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Stand der Technik

Die größten und wohl auch bekanntesten Einsatzgebiete, wenn es um aktive Geräuschunterdrückung geht, sind Kopfhörersysteme mit Active Noise Cancellation (ANC). Diese lassen sich sowohl in Hubschraubern oder Flugzeugen, als auch im Privatbereich finden. Zurückzuführen ist dieses System auf ein Patent aus dem Jahr 1936 von Paul Lueg [4]. Ihm kam die Idee, gegebene Schallwellen mit künstlich erzeugten Schallwellen zu überlagern, dass der durch den gegebenen Schall erzeugte Schallpegel vermindert wird. Dazu würde der vorhandene Störschall aufgenommen und im optimalen Fall so modifiziert werden, dass er invertiert wieder ausgegeben wird.

Abgesehen von Kopfhörersystemen gibt es noch weitere Einsatzgebiete, in denen aktive Geräuschunterdrückung zu finden ist. Inzwischen wird diese Technik von einigen Autoherstellern verwendet, um Motoren- und Windgeräusche zu unterdrücken [3] oder auch in Windkraftwerken, um diese mit einer höheren Drehzahl laufen zu lassen [6].

Es ist ebenfalls möglich, diese Technik auf Räume anzuwenden, in denen Störgeräusche vorhanden sind.

1.2. Ziele

Ziel dieser Arbeit soll zum einen sein, ein VST-Plug-in zu entwickeln, welches durch adaptive Filterung Störgeräusche in einem Arbeitsraum verringert. Dieses Plug-in ließe sich in Programmen wie Cubase oder Audacity einsetzen. Damit ließen sich sowohl bestimmte Geräusche vollständig unterdrücken als auch nur bestimmte Frequenzbereiche, deren passive Filterung besonders bei tiefen Frequenzen nur mit großem Aufwand realisierbar wäre.

Zum anderen soll geprüft werden, in wie weit sich Störsignale mit dieser Technik dämpfen lassen. Als Beispiel hierfür dienen das Geräusch einer Klimaanlage sowie ein Sinussignal und verschiedene synthetische Instrumente.

2. Theorie zur aktiven Geräuschkompensation

2.1. Eindimensionaler Ansatz

2.1.1. Physikalische Grundlagen

Um das Prinzip der Überlagerung etwas zu vereinfachen, wird zunächst die Überlagerung von Transversalwellen betrachtet.

Lässt man zwei Wellen mit gleicher Amplitude und Frequenz aufeinander treffen, entstehen sowohl konstruktive als auch destruktive Interferenzen. Abhängig davon, ob ein Wellenberg auf ein Wellental oder einen weiteren Wellenberg trifft, ist die Amplitude an der Stelle entweder Null oder das zweifache eines der Wellenberge/-täler.

Anders verhält es sich, wenn die Wellen aus der selben Richtung kommen. Bei gleicher Frequenz und invertierter Amplitude heben sich die Wellen gegenseitig auf (siehe Abbildung 2.1).

Bei realen Schallwellen lässt sich ebenfalls das Prinzip der destruktiven Interferenzen beobachten. Dabei gilt es zu beachten, dass sich Schallwellen in einem Raum sowohl kreisförmig als auch geradlinig ausbreiten können, abhängig davon, welcher Quelle sie entstammen. Diese Unterschiede spielen eine große Rolle bei destruktiven Interferenzen. In beiden Fällen ist es physikalisch nicht mehr möglich, Schallwellen aus der selben Richtung kommen zu lassen. Dadurch entstehen neben den destruktiven immer auch konstruktive Interferenzen. Bei Schallquellen, die geradlinigen Schall liefern, ist das Problem nicht so schwerwiegend, da die Interferenzmuster nicht so auffallend sind wie bei kreisförmigen Quellen.

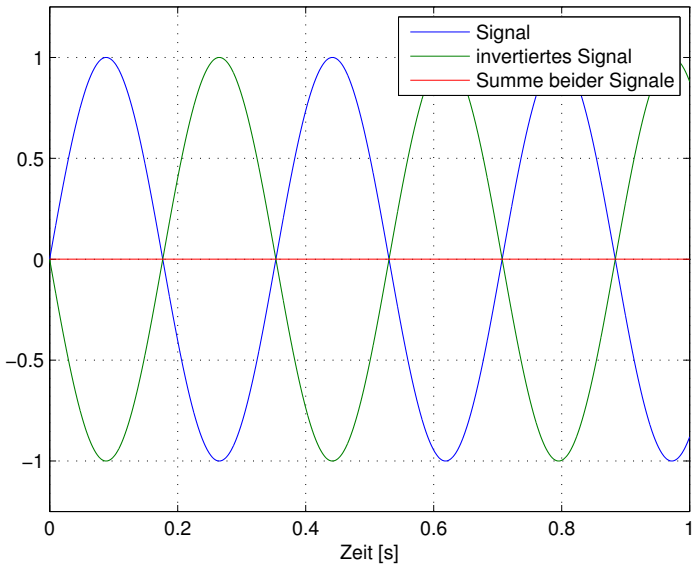

Abbildung 2.1.: Eine Welle und die Spiegelung an der X–Achse aus gleicher Richtung

2.1.2. Mathematische Grundlagen

Die mathematische Darstellung bezieht sich auf das Prinzip, wie das inverse Signal zu einem gegebenen Signal ausgerechnet werden kann. In der Theorie hat eine Welle mit einer bestimmten Frequenz folgende Form und ist abhängig von Ort und Zeit.

$$\xi(x,t) = \xi_0 \cdot \sin\left(\frac{2\pi}{\lambda}(x - \upsilon t)\right) \tag{2.1}$$

Diese lässt sich umformen in

$$\xi(x,t) = \xi_0 \cdot \sin\left[\omega\left(\frac{x}{c} - t\right)\right] \tag{2.2}$$

ξ_0 steht hierbei für die Amplitude, $\omega = 2\pi f$ für die Kreisfrequenz, $c = 343\frac{m}{s}$ für die Fortbewegungsgeschwindigkeit in der Luft, x und t jeweils für Ort und Zeit. Gesucht wird jetzt das Signal, dass $\xi(x,t)$ an der X–Achse spiegelt. Dieses Signal, in diesem Fall $\psi(x,t)$, lässt sich auf zwei Arten realisieren. Die erste Möglichkeit ist, die Amplitude $-\xi_0$ zu invertieren. Die zweite Möglichkeit ist, die Phase des Signals um $180°$ zu verschieben.

Bei der Verschiebung der Phase muss beachtet werden, dass die Verschiebung um $180°$ nicht bei jeder Art von Signal funktioniert. Voraussetzung dafür ist, dass die Signale periodisch und nach der halben Periode gespiegelt sind.

Invertierung der Amplitude

Um $\psi(x,t)$ anhand der Amplitude zu erzeugen, wird ein Signal gesucht, dass addiert zum Ursprungssignal $\xi(x,t)$, 0 ergibt.

$$\xi(x,t) + \psi(x,t) = 0 \tag{2.3}$$

Setzt man für $\xi(x,t)$ und $\psi(x,t)$ jeweils die Formel 2.1 ein, erhält man

$$\xi_0 \cdot \sin\left[\omega\left(\frac{x}{c} - t\right)\right] + \psi_0 \cdot \sin\left[\omega\left(\frac{x}{c} - t\right)\right] = 0 \tag{2.4}$$

Die beiden Faktoren mit $\sin$ lassen sich auf beiden Seiten kürzen

$$\xi_0 + \psi_0 = 0 \tag{2.5}$$

$$\Rightarrow \quad \psi_0 = -\xi_0 \tag{2.6}$$

Da $\psi_0 = -\xi_0$ ist, ergibt sich aus Gleichung 2.5

$$\xi_0 + (-\xi_0) = 0 \qquad \text{bzw.} \qquad \xi_0 - \xi_0 = 0 \tag{2.7}$$

Phasenverschiebung um 180 Grad

Als Alternative zur Invertierung der Amplitude ist es auch möglich, eines der Signale um $180°$ in der Phase zu verschieben.

$$\xi_0 \cdot \sin\left[\omega\left(\frac{x}{c} - t\right)\right] + \xi_0 \cdot \sin\left[\omega\left(\frac{x}{c} - t\right) + \alpha\right] = 0 \tag{2.8}$$

Gesucht wird der Winkel α, der dafür sorgt, dass Gleichung 2.9 0 ergibt.

$$\sin(\omega) + \sin(\omega + \alpha) = 0 \tag{2.9}$$

Gemäß den Regeln der Trigonometrie ist dies genau dann der Fall, wenn α ein Vielfaches von $180°$ ist.

$$\sin(\omega) + \sin(\omega + (n \cdot 180°)) = 0 \tag{2.10}$$

Setzt man für $n = 1$ ein, ergibt das

$$\sin(\omega) + \sin(\omega + 180^\circ) = 0 \tag{2.11}$$

Aus der Tatsche das

$$-\sin(\alpha) = \sin(\alpha + 180^\circ) \tag{2.12}$$

ist, ergibt sich aus Gleichung 2.11:

$$\sin(\omega) - \sin(\omega) = 0 \tag{2.13}$$

Beide Methoden funktionieren, solange man Wellen eindimensional betrachtet und die Anforderungen an Frequenz und Amplitude eingehalten werden.

2.2. Dreidimensionaler Ansatz

Schallüberlagerung im realen Raum findet in 3 Dimensionen statt. Für jede dieser Dimensionen gilt die oben beschriebene physikalische und mathematische Überlagerung aus 2.1.1 und 2.1.2. Durch Reflektionen von Wänden, Böden und Decken sowie der Tatsache, dass sich Schall von Lautsprechern aus kreisförmig entfernt, können Interferenzen entstehen, die für aktive Geräuschkompensation in einem Raum hinderlich sind. Es gilt zu beachteten, dass Schall, der erzeugt wird um Störgeräusche zu dämpfen, so reflektiert werden kann, dass er das eigentliche Störgeräusch nur noch weiter verstärkt. Zusätzlich bewirkt die kreisförmige Ausstrahlung aus einem Lautsprecher, dass es nur einen gewissen Bereich gibt, in dem sich der Schall aktiv dämpfen lässt. Im restlichen Bereich enstehen entweder durch die Ausstrahlung der Lautsprecher oder durch Reflektionen konstruktive Interferenzen, die dafür sorgen, dass der Schall nicht gedämpft wird.

Vergleichen lässt sich dies mit Interferenzmustern von Wellen im Wasser. Wirft man zwei identische Steine in ein Becken mit Wasser, entstehen an der Stelle, an der sich die Wellen treffen, Interferenzmuster. Diese sind vergleichbar mit den Mustern, die bei Schallüberlagerung entstehen.

3. Adaptive Filterung

3.1. Grundlagen

Zur aktiven Geräuschkompensation wird ein adaptives Filtersystem verwendet. Dieses Filtersystem besteht aus zwei Komponenten: einem FIR-Filter und einem Adaptionsalgorithmus (vgl. Abbildung 3.1).

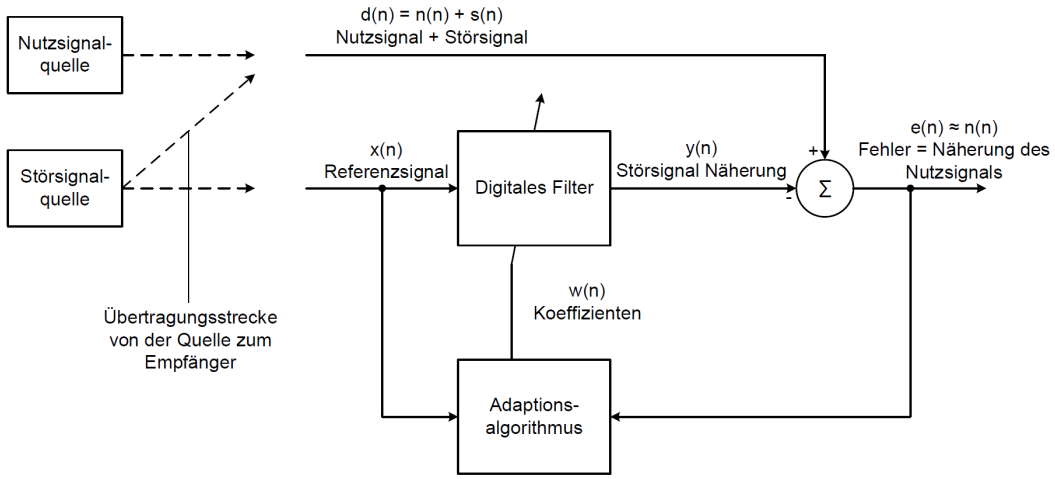

Abbildung 3.1.: Adaptives Filtersystem

3.1.1. FIR–Filter

Das FIR–Filter bekommt ein Eingangssignal $x(n)$ zugeführt, welches mit den Filterkoeffizienten $w(n)$ gefiltert wird. Bei $x(n)$ handelt es sich um das Referenzsignal des aufgenommenen Störsignals $s(n)$, dass sich nicht 1:1 in dem System abbilden lässt. Als Ergebnis der Filterung erhält man mit dem Signal $y(n)$ eine Annäherung an das Störsignal. Diese Annäherung subtra-

hiert man von $d(n)$ (der Summe aus Nutzsignal $n(n)$ und Störsignal $s(n)$), um das Fehlersignal $e(n)$ zu erhalten.

$$e(n) = d(n) - y(n) \tag{3.1}$$

$$e(n) = n(n) + s(n) - y(n) \tag{3.2}$$

Im idealen Fall entspricht die Störsignalannäherung $y(n)$ dem Störsignal $s(n)$,

$$e(n) = n(n) + y(n) - y(n) \tag{3.3}$$

womit sich in der Gleichung 3.3 $y(n)$ aufhebt

$$e(n) \approx n(n) \tag{3.4}$$

und das Fehlersignal $e(n)$ dem Nutzsignal $n(n)$ entspricht.

3.1.2. Adaptionsalgorithmus

Der Adaptionsalgorithmus errechnet aus dem Fehlersignal $e(n)$ und dem Referenzsignal $x(n)$ einen neuen Satz Filterkoeffizienten $w(n)$ für das digitale Filter.

Ziel des Adapitonsalgorithmus ist es, das Fehlersignal $e(n)$ zu minimieren. Die Anpassung der Filterkoeffizienten geschieht so lange, bis das Signal $y(n)$ keinerlei zeitliche Korrelation mit dem Fehlersignal $e(n)$ aufweist. Dies gelingt durch die Formel 3.5, die in [5] ab Kapitel 3.2.1 näher beschrieben wird.

$$w(n+1) = w(n) + \mu e(n)x(n) \tag{3.5}$$

Der Faktor μ sorgt für die Konvergenz des Algorithmus. Je kleiner μ gewählt wird, desto länger dauert es, bis der Algorithmus konvergiert und die optimalen Filterkoeffizienten gefunden hat. Entsprechend gut ist dafür das Ergebnis bei einfachen Eingangssignalen. Wird μ zu groß gewählt, kann das eine ungenaue Filterung zur Folge haben oder das System in Schwingung versetzten, sodass $w(n)$ und $y(n)$ unkontrolliert größer werden.

Kleine Werte für μ können gewählt werden, wenn sich die Eingangssignale nur langsam oder gar nicht ändern, also eine gleichbleibende mittlere Signalenergie aufweisen. Hier kann teilweise sogar ein konstanter Wert für μ gewählt werden.

Größere Werte für μ sollten gewählt werden, wenn sich die Eingangssignale häufiger ändern und ihre mittlere Signalenergie nicht konstant ist. Dadurch kann sich der Algorithmus in kurzer Zeit anpassen und neue Filterkoeffizienten berechnen. Bei der Wahl des Wertes für μ ist zu beachten, dass immer Bedingung 3.6 erfüllt sein muss, damit es zur Konvergenz kommt. Für weitere Erläuterungen siehe [5, Seite 26, ab Gleichung 39]

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{3.6}$$

Für die dynamische Berechnung von μ gilt

$$\mu(n) = \frac{\beta}{\gamma + \mathbf{x}^T(n)\mathbf{x}(n)}, \tag{3.7}$$

wobei β so zu wählen ist, dass $0 < \beta < 2$ gilt und γ eine kleine positive Konstante ist, die verhindert, dass μ zu groß wird, wenn das Produkt aus $\mathbf{x}^T(n)\mathbf{x}(n)$ zu klein ist.

3.2. Besonderheiten der blockweisen Filterung

Um ein adaptives Filter mit blockweiser Verarbeitung zu entwerfen (Abbildung 3.2), muss sowohl das adaptive Filter als auch der Adaptionsalgorithmus angepasst werden. Wichtig ist hierbei, dass die Ein- und Ausgangssignale blockweise vorliegen (siehe Beispiel für das Referenzsignal 3.8) und nicht mehr elementweise. Die Unterschiede zwischen der seriellen Verarbeitung, wie sie in [5] verwendet wird, und der blockweisen Verarbeitung werden im folgenden Abschnitt aufgezeigt.

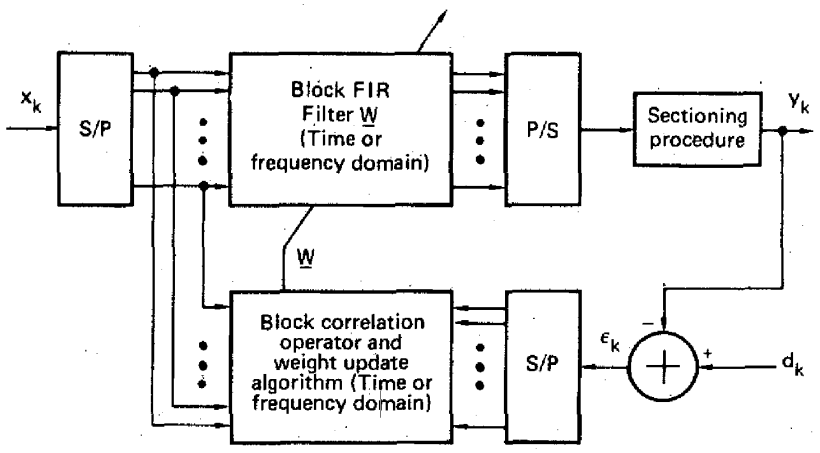

Abbildung 3.2.: Adaptives Filtersystem mit blockweiser Verarbeitung

3.2.1. Blockweise FIR–Filterung

Das FIR–Filter verrechnet das Eingangssignal

$$\mathbf{X}_n = \begin{pmatrix} x_{1+nL} \\ x_{2+nL} \\ \vdots \\ x_{L+nL} \end{pmatrix} \tag{3.8}$$

mit den Filterkoeffizienten. Dies geschieht mittels schneller Faltung.

$$\mathbf{Y}_n = [\mathbf{X}_n * \mathbf{W}_n]_{0 \ldots L-1} \tag{3.9}$$

Das Ergebnis dieser Rechnung ist ein Block der Größe $2L-1$ des Ausgangssignals $\mathbf{Y}_n$, dessen erste L-Elemente verwendet werden.

3.2.2. Blockweise Adaption

Bei der seriellen Verarbeitung sieht die Formel zum Berechnen der neuen Filterkoeffizienten wie folgt aus:

$$w(n+1) = w(n) + \mu e(n) x(n) \tag{3.10}$$

Für die blockweise Adaption (siehe [2, Seite 746, Gleichung 19]) ändert sich die Formel auf folgende Art:

$$\mathbf{W}_{n+1} = \mathbf{W}_n + \frac{2\mu_B}{L} \cdot \Phi_n \tag{3.11}$$

Φ_n beschreibt die Korrelation zwischen dem Störsignal und Fehlersignal und lässt sich entweder durch Multiplikation der transponierten Eingangsmatrix $\mathcal{X}_n^T$ mit dem Fehlersignal $\mathbf{E}_n$ (siehe Gleichung 3.12) oder mittels schneller Faltung lösen (siehe Gleichung 3.13).

Für die schnelle Faltung gilt ebenso wie für Gleichung 3.9, dass der Ergebnisvektor eine Größe von $2L-1$ hat. Hiervon werden jedoch nur die hinteren L-Elemente verwendet.

$$\Phi_n = \mathcal{X}_n^T \mathbf{E}_n = \begin{pmatrix} x_{1+nL} & 0 & 0 & 0 \\ \vdots & x_{1+nL} & 0 & 0 \\ x_{L-1+nL} & \vdots & \ddots & 0 \\ x_{L+nL} & x_{L-1+nL} & \cdots & x_{1+nL} \end{pmatrix}^T \cdot \begin{pmatrix} e_{1+nL} \\ e_{2+nL} \\ \vdots \\ e_{L+nL} \end{pmatrix} \tag{3.12}$$

$$\Phi_n = [\text{invert}(\mathbf{X}_n) * \mathbf{E}_n]_{L \,\ldots\, 2L-1} \tag{3.13}$$

μ_B beeinflusst hier ebenfalls die Konvergenzgeschwindigkeit des Algorithmus und lässt sich unter der Annahme errechnen, dass sowohl die Konvergenzgeschwindigkeit als auch die Konvergenzgenauigkeit für die serielle und blockweise Implementierung gleich sind. Die Geschwindigkeit wird für die blockweise Implementierung als

$$T_{p\text{MSE}} = \frac{L}{4\mu_B \lambda_p} \tag{3.14}$$

definiert und die Genauigkeit als

$$\mathcal{M} = \frac{\mu_B}{L} \operatorname{tr} R \tag{3.15}$$

Für die serielle Implementierung gilt

$$\tau_{p\text{MSE}} = \frac{1}{4\mu\lambda_p} \tag{3.16}$$

als Formel für die Konvergenzgeschwindigkeit und

$$M = \mu \operatorname{tr} R \tag{3.17}$$

als Formel für die Genauigkeit.

Unter der zuvor genannten Annahme ergibt sich aus dem Verhältnis von Geschwindigkeit und Genauigkeit der beiden Implementierungsarten jeweils 1.

$$\frac{T_{p\text{MSE}}}{\tau_{p\text{MSE}}} = 1 \qquad \frac{\mathcal{M}}{M} = 1 \tag{3.18}$$

Durch Einsetzen der Formeln 3.14 bis 3.17 in 3.18 erhält man

$$\frac{\frac{L}{4\mu_B\lambda_p}}{\frac{1}{4\mu\lambda_p}} = 1 \qquad \text{und} \qquad \frac{\frac{\mu_B}{L}\operatorname{tr} R}{\mu \operatorname{tr} R} = 1 \tag{3.19}$$

Aus beiden Formeln lässt sich μ_B ableiten.

$$\mu_B = \mu \cdot L \tag{3.20}$$

Ebenso wie in Formel 3.6 darf μ_B nicht die folgende Bedingung verletzen.

$$0 < \mu_B < \frac{1}{\lambda_{\max}} \tag{3.21}$$

Hierbei steht $\lambda_{\max}$ für den größten Eigenwert aus der Matrix $R = E\left[\mathbf{X}_n \mathbf{X}_n^T\right]$, welche für die mittlere Eingangsleistung des Störsignals steht und durch

$$\sum_{i=0}^{L-1} [\mathbf{X}_n(i)^2] \tag{3.22}$$

beschrieben werden kann.

3.3. Besonderheiten bei der Anwendung des Algorithmus auf das Plug-in

Bei der Anwendung des Adaptionsalgorithmus ist zu beachten, dass die Addition der Näherung des Störsignals und der Überlagerung aus Nutz- und Störsignal (vgl. Formel 3.1) außerhalb des Plug-in geschieht. Das Plug-in errechnet ein Ausganssignal, welches über einen Lautsprecher ausgegeben wird und im Raum auf das Störsignal trifft. Die Addition der Signale findet daher durch die physikalische Überlagerung von Schallwellen im Raum statt. Das Resultat dieser Addition ist das Fehlersignal und wird, über ein Mikrofon aufgenommen, dem Algorithmus zugeführt.

4. Software und Geräte

4.1. Tascam US–122

Als Schnittstelle zwischen Lautsprechern und Mikrofon mit dem PC dient das Tascam US-122. Es bietet eine Abtastrate von 44,1 kHz oder 48 kHz bei einer Auflösung von 16– oder 24–Bit [7, Seite 21]. Diese Einstellungsmöglichkeiten werden in diesem Fall allerdings nicht vollständig genutzt. Die Auflösung bleibt bei 16–Bit und die Abtastrate bei 44,1 kHz.

Abbildung 4.1.: Tascam US–122 Audiointerface

Für diese Arbeit ist jedoch die Möglichkeit der Einstellung für die Puffergröße interessant. Diese lässt sich zwischen 256, 512, 1024 und 2048 wählen [7, Seite 18]. Genutzt wird eine Puffergröße von 1024, da die Filterung im Plug-in nur bis zu dieser Größe flüssig funktioniert. Eine Puffergröße von 2048 wäre mit einem schnelleren Prozessor wahrscheinlich möglich.

4.2. Mikrofon AKG C 3000 B

Bei dem Mikrofon zum Aufnehmen des Störgeräusch handelt es sich um das AKG C 3000 B. Damit ist es möglich, Frequenzen von 20 Hz bis 20 kHz bei fast linearem Frequenzgang aufzunehmen. Diese Eigenschaften sind besonders wichtig, um zum einen den vom Menschen hörbaren Bereich abzudecken, zum anderen um das Störgeräusch möglichst unverfälscht aufzunehmen [1, Seite 4].

4.3. Cubase LE 5

Cubase ist ein umfangreiches Programm zur Erstellung von Musik. Es unterstützt VST3–Plug-ins und kann ASIO–Treibern verwenden; beides Eigenschaften, die während dieser Bachelorarbeit zum Tragen kommen.

4.4. VST–Plug-in

VST (Virtual Studio Technology) ist eine Schnittstelle für Software im Audiobereich. Die Schnittstelle stellt eine Verbindung zwischen dem VST-Host, in diesem Fall die Software Cubase, und einem Effekt-Plug-in her. Es wird durch eine DLL (Dynamic Link Library) realisiert, die von dem Host geladen werden kann. Das Plug-in besteht aus zwei Teilen, dem Processor und dem Controller.

4.4.1. Controller

Der Controller bietet hierbei das Interface zum Anwender und ermöglicht die Steuerung des Plug-ins. Es beinhaltet einen Regler, mit dem sich das μ_B des Plug-ins beeinflussen lässt.

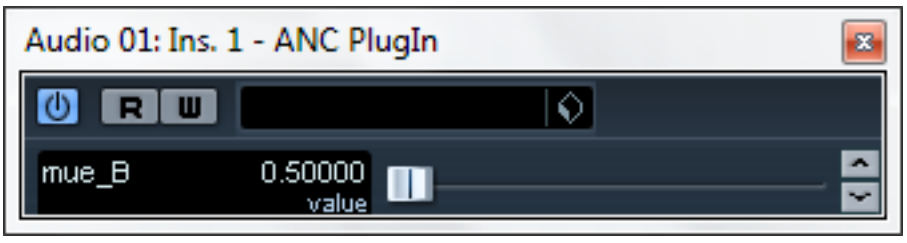

Abbildung 4.2.: Oberfläche des eingeschalteten Plug-in; der Konvegenzfaktor ist auf 0,5 gestellt

4.4.2. Processor

Der Processor nimmt die eigentliche Berechnung vor, verarbeitet Eingabesignale, erzeugt Ausgabesignale und nimmt Eingabewerte von dem Controller entgegen. Die Ein- und Ausgabe kann nur blockweise bei einer Blockgröße, die in Zweierpotenzen zwischen 256 und 2048 wählbar ist, geschehen. Alternativ lässt sich die Blockgröße auch durch eine Angabe in Milisekunden festlegen. Diese Alternative ist für das Plug-in jedoch ungeeignet, da die schnelle Faltung am besten mit Zweierpotenzen arbeitet.

Im Fall dieser Arbeit soll eine Blockgröße von 1024 gewählt werden, um geringe Latenz zu haben. Bei dieser Blockgröße ist es noch möglich, niedrige Frequenzen zu filtern. Eine Auflistung von Latenzzeit und minimaler Frequenz bei einer Abtastrate von 44,1 kHz lässt sich in Tabelle 4.1 nachlesen.

Blockgröße	Latenz	minmale Frequenz
128	2,925 ms	341,86 Hz
256	5,828 ms	171,60 Hz
512	11,633 ms	85,96 Hz
1024	23,243 ms	43,02 Hz
2048	46,463 ms	21,52 Hz
4096	92,902 ms	10,76 Hz

Tabelle 4.1.: Tabelle mit Latenzzeit und minimal filterbarer Frequenz bei verschiedenen Blockgrößen und einer Abtastrate von 44,1 kHz

4.5. ASIO–Treiber

ASIO (Audio Stream Input/Output) beschreibt ein Audiotransfer–Protokoll zum Zugriff auf Audiogeräte, um sehr geringe Latenzwerte zu erreichen. Diese spielen besonders in der Echtzeitverarbeitung von Audiosignalen eine wichtige Rolle.

5. Realisierung von Prototypen in MATLAB

Die MATLAB-Prototypen dienen dem Vergleich zwischen der seriellen und blockweisen Implementierung des Algorithmus. Des Weiteren soll das Verhalten von Cubase unter realen Bedingungen simuliert werden.

Vorteil der Prototypen ist, dass MATLAB sehr gute Möglichkeiten zum Debuggen bietet, die sich in C++ nicht ergeben. Außerdem lässt sich in MATLAB der Idealfall des Plug-ins simulieren. Daraus ergibt sich die Möglichkeit, die Leistung der blockweisen Implementierung vorher zu messen und mögliche Definzite festzustellen.

5.1. Serielle Implementierung

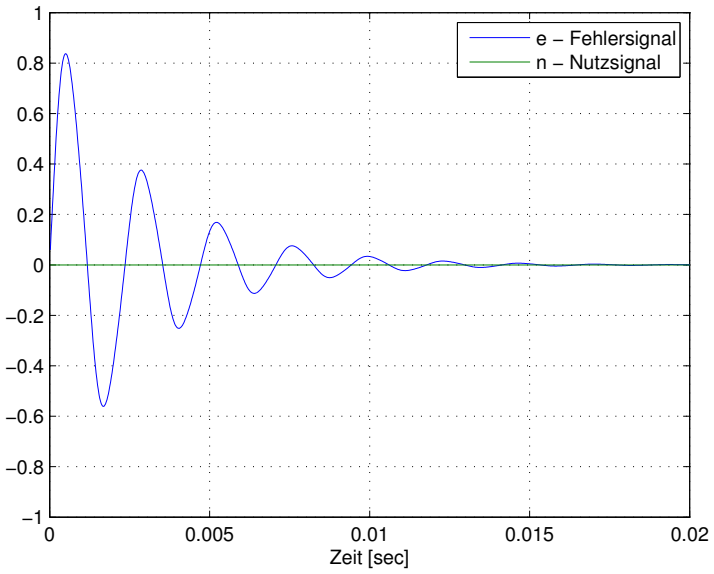

Abbildung 5.1.: Dämpfung eines Sinusstörgeräuschs mit 424 Hz bei einem Konvergenzfaktor von 0,01533 mittels serieller Verarbeitung, das Fehlersignal nähert sich immer weiter dem Nutzsignal

Das Skript „`test_LMS.m`“ (siehe Anhang B) zeigt die Funktionsweise der adpativen Filterung, wie sie in der Masterarbeit von Jörn Matthies [5, Kapitel 3] verwendet wird.

5.1.1. Sinus auf Stille

Für den ersten Vergleich wird ein Sinussignal als Störgeräusch verwendet. Ein Nutzsignal gibt es in diesem Fall noch nicht. Das Sinussignal hat eine Frequenz von $\sqrt{2} \cdot 300\,\text{Hz} = 424\,\text{Hz}$ bei einer Abtastrate von 44,1 kHz und einer Filtergröße von 1024. Diese Einstelleungen werden für alle weiteren Tests gleich bleiben. Die Grafik 5.1 verdeutlicht, wie schnell der Algorithmus bei einem μ von 0,01533 konvergiert und das Störgeräusch dämpft. Innerhalb von 15 Milisekunden dämpft der Algorithmus das Störsignal um -20 dB.

5.1.2. Klimaanlagengeräusch auf Stille

Für den zweiten Vergleich wird das Geräusch der Klimaanlage benutzt. Hier liegt der Konvergenzfaktor bei 0,002. Zu erkennen ist auf Abbildung 5.2, dass hier die Filterung deutlich länger dauert als bei dem Sinussignal. Während bei dem Sinussignal schon nach 15 Milisekunden eine Dämpfung von -20 dB erreicht wird, dauert es in diesem Fall 0,5 Sekunden.

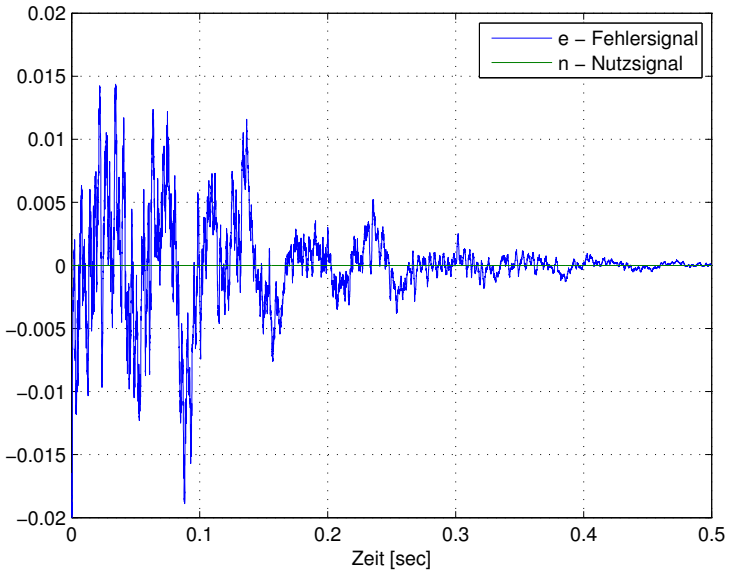

Abbildung 5.2.: Dämpfung des Geräuschs der Klimaanlage mittels serieller Verarbeitung bei einem Konvergenzfaktor von 0,002; das Fehlersignal nähert sich immer weiter dem Nutzsignal

5.2. Blockweise Implementierung

Das Skript „`test_BLMS.m`“ (siehe Anhang B) zur blockweisen Filterung soll veranschaulichen, wie Leistungsstark es gegenüber der seriellen Implementierung ist. Getestet wird hier jeweils das Verhalten in drei Fällen. In zwei dieser Fälle ist kein Nutzsignal vorhanden, es soll jeweils Stille erzeugt werden. Im letzten Fall ist das Nutzsignal ein Gitarrenstück.

Für die Versuche mit Stille werden, wie bei der seriellen Implementierung, wieder ein Sinussignal und eine Probeaufnahme der Klimaanlage verwendet.

5.2.1. Sinus auf Stille

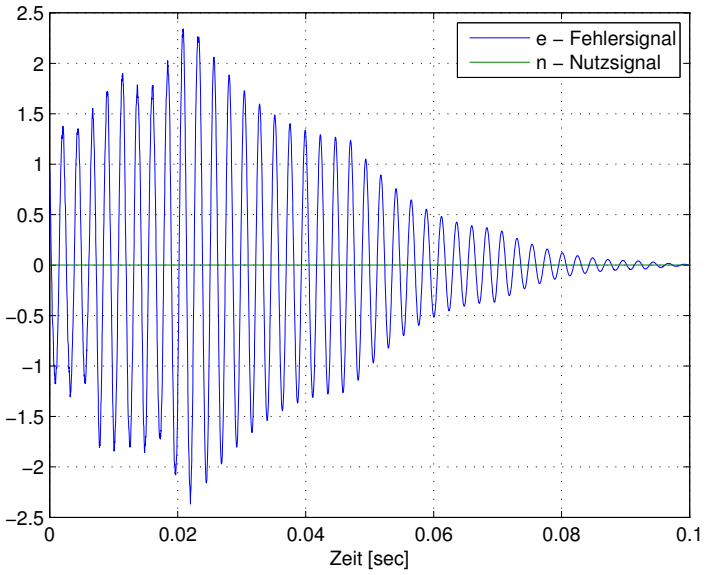

Abbildung 5.3.: Dämpfung eines Sinusstörgeräuschs mit 424 Hz mittels blockweiser Filterung, Konvergenzfaktor = 0,0025

Die Abbildung 5.3 zeigt das Verhalten bei blockweiser Filterung und zufällig gewählten Startkoeffizienten zwischen 0 und 0,1. Zu sehen ist, dass innerhalb der ersten 60 ms die größte Anpassung stattfindet, während danach nur noch Anpassungen an die Phase geschehen. Eine Dämpfung des Störsignals um ca. -20 dB wird jedoch erst nach 0,1 Sekunde erreicht. Die lange Konvergenzdauer ist abhängig davon, wie die Startkoeffizienten und der Konvergenzfaktor für den Algorithmus gewählt werden. Der Konvergenzfaktor liegt mit 0,0025 unter dem für die serielle Implementierung gewählten Wert von 0,01533. Nach Gleichung 3.20 muss μ_B jedoch

bei 15,7 liegen, was wiederum Gleichung 3.21 stark verletzen würde, da $\mu_B < 0,002$ sein muss, um Konvergenz zu erreichen. Bei Werten, die nur gering über dem Maximum liegen, konvergiert der Algorithmus meist auch noch.

Aufeinanderfolgende Tests mit unterschiedlichen Startkoeffizienten zeigten, dass die Filterleistung nach 0,1 Sekunde um $\pm$ 5 dB abweichen kann.

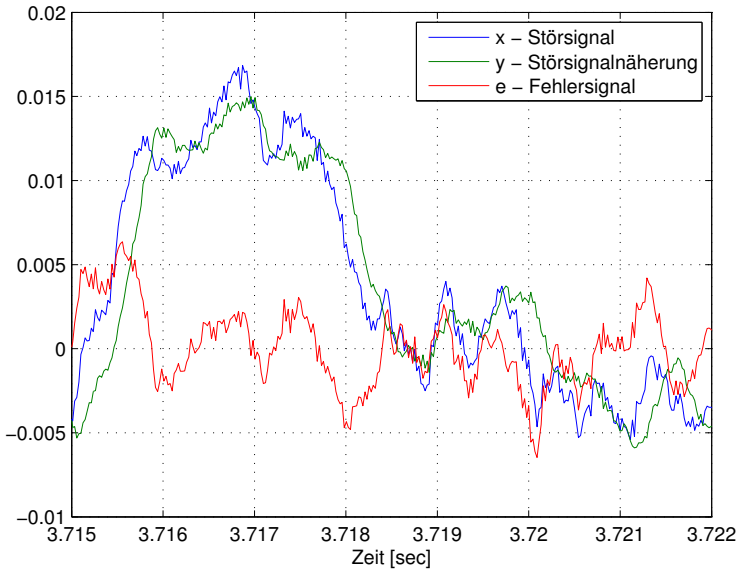

Abbildung 5.4.: Dämpfung des Geräuschs der Klimaanlage bei blockweiser Verarbeitung; zu sehen ist das Fehlersignal (Differenz aus Störsignal und Störsignalnäherung)

5.2.2. Klimaanlagegeräusch auf Stille

Bei dem Geräusch der Klimaanlage liefert der Algorithmus ein akzeptables Ergebnis. Der Konvergenzfaktor μ_B liegt hier bei 20, die Startkoeffizienten zwischen 0 und 0,1 werden zusätzlich mit einem von–Hann–Fenster multipliziert.

Wie die Grafik 5.4 zeigt, stimmen Nutzsignal und die Näherung des Nutzsignals fast überein. Die Unterschiede haben hier Auswirkung auf den Frequenzbereich ab 400 Hz, in dem kaum noch Filterung statt findet.

Ein weiterer Versuch mit zufälligen Startkoeffizienten zwischen 0 und 0,1 zeigte, dass sich das Filterverhalten im niederfrequenten Bereich stark verändert. Abbildung 5.5 veranschaulicht das Verhalten des Algorithmus bei unterschiedlichen Startkoeffizienten.

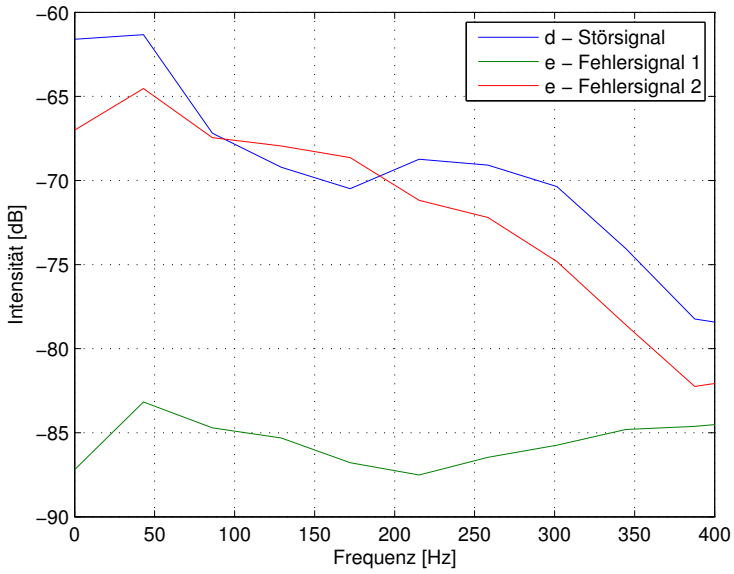

Abbildung 5.5.: Unterschied in der Filterleistung; Startkoeffizienten von Fehlersignal 1 mit von–Hann–Fenster, Fehlersignal 2 ohne

5.2.3. Klimaanlagengeräusch auf Gitarrenklang

Bei dem letzten Beispiel handelt es sich bei dem Störgeräusch erneut um das Rauschen einer Klimaanlage. Anders als bei den beiden vorangegangenen Tests wird hier als Nutzsignal ein Gitarrensolo verwendet, um auch die Leistung unter realen Bedinungen besser einschätzen zu können. Die Filterkoeffizienten werden hier ebenfalls mit einer Fensterfunktion multipliziert.

Vergleicht man hier das Nutzsignal mit dem Fehlersignal (Abbildung 5.6), stellt man fest, dass diese ebenfalls fast übereinander liegen. Die Unterschiede machen sich hier im hochfrequenten Bereich bemerkbar.

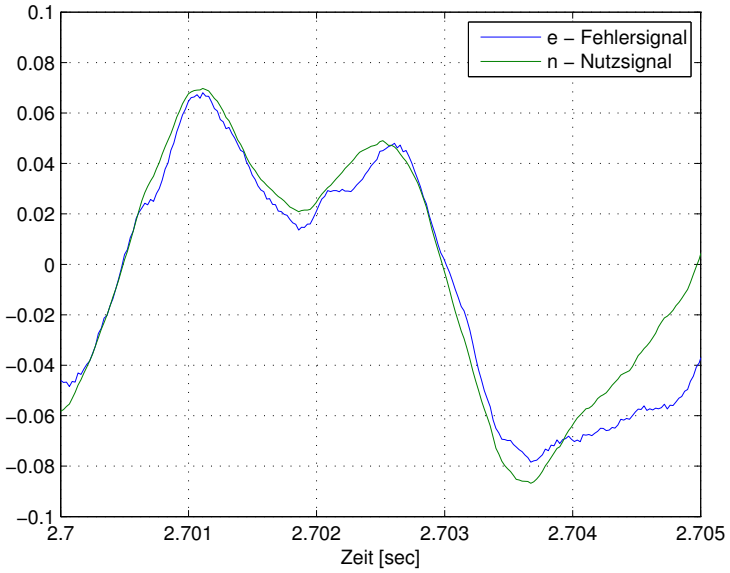

Abbildung 5.6.: Dämpfung des Geräuschs der Klimaanlage, welches ein Gitarrensolo verunreinigt hat; das Fehlersignal nähert sich dem Nutzsignal

5.2.4. Verhaltenssimulation von Cubase unter realen Bedingungen

Das letzte Skript „`test_BLMS_cubase.m`“ (siehe Anhang B) soll das Verhalten von Cubase unter realen Bedingungen simulieren. Diese sind gegeben, wenn das Plug-in sowohl das Störgeräusch als auch das Fehlersignal über ein Mikrofon aufnimmt. Die größten Unterschiede, die sich daraus ergeben, sind die Puffer, die die Ein- und Ausgabe verzögern sowie die Addition von Schall im Raum. Letzteres lässt sich nicht simulieren, da die Überlagerung von Schall im Raum von zu vielen Faktoren abhängt.

Um das Resultat dieses Skriptes mit der einfachen blockweisen Implementierung zu vergleichen (siehe 5.2), werden hier als Eingangssignale nur das Klimaanlagengeräusch und das Gitarrensolo verwendet. Die Startkoeffizienten wurden hier ebenfalls gefenstert, μ_B liegt bei 10.

Vergleicht man auch hier das Nutzsignal mit dem Fehlersignal, stellt man fest, dass das Resultat unter gleichen Bedingungen vergleichbar ist mit der blockweisen Implementierung ohne Verzögerung.

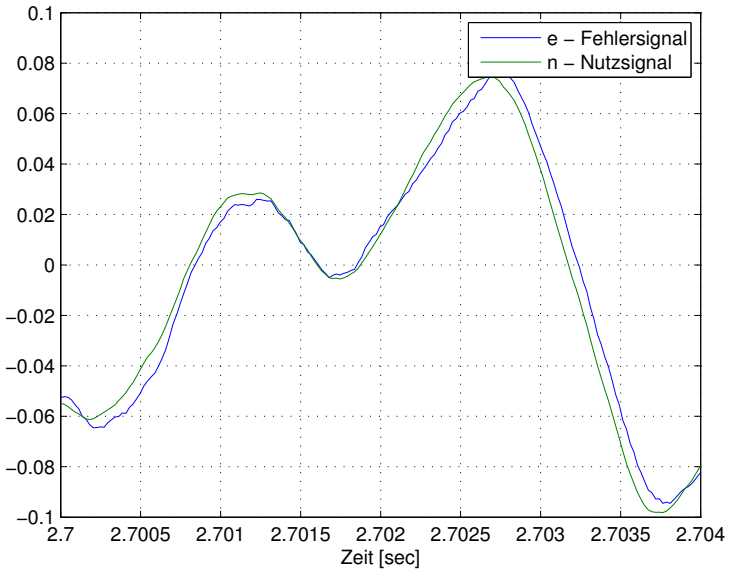

Abbildung 5.7.: Erwartetes Verhalten des Plug-in im realen Anwendungsfall; das Fehlersignal nähert sich dem Nutzsignal

6. Test des VST–Plug-in unter realen Bedinungen

6.1. Einleitung

Um die Funktionsfähigkeit des Plug-ins zu testen, wurden mehrere Testszenarien erstellt. Getestet wird das Verhalten, ähnlich wie bei den MATLAB-Skripten in Kapitel 5, mit jeweils einem Sinussignal und einer Probeaufnahme der Klimaanlage. Des Weiteren werden Tests mit komplexeren Sinussignalen und synthetischen Instrumenten als Störsignal durchgeführt. Nutzsignal ist hier entweder ein Audiosignal oder Stille.

6.2. I/O und Funktionalität des Plug-in

Das Plug-in verfügt über einen Stereo Ein- und ausgang. Über den Eingang wird auf dem rechten Kanal das Störsignal zugefügt, über den linken Kanal mittels Mikrofon das Fehlersignal. Der linke Kanal des Stereoausgangs gibt das Kompensationssignal wieder.

Zusätzlich bietet das Plug-in eine Einstellmöglichkeit für den Konvergenzwert μ_B. Dieser kann durch einen Regler im Bereich von 0,0005 bis 50 eingestellt werden. Um eine Übersteuerung zu vermeiden, sollte der Regler nur langsam bewegt werden. Werte unter 1 lassen sich nur manuell eintragen. Eine dynamische Berechnung des Konvergenzwertes ist in diesem Plug-in nicht implementiert.

6.3. Notwendige Geräte

Für Versuche mit auswertbaren Resultaten wird Folgendes benötigt:

- PC oder Laptop mit einem Stereo–Audioausgang
- ein Audiointerface mit sehr geringer Latenz und einstellbarer Blockgröße
- eine Audiosoftware, die das Fehlersignal separat aufnehmen kann, um es nach dem Versuch zu analysieren

- Cubase oder ähnlicher Software, die VST3–Plug-ins und ASIO–Treiber unterstützt
- Mikrofon und zwei Lautsprecher mit möglichst linearem Frequenzgang

Für Versuche mit einem Audiosignal als Nutzsignal wird noch ein weiterer Lautsprecher bzw. eine weitere Wiedergabequelle benötigt.

6.4. Versuchsaufbau

Die zwei Lautsprecher werden parallel nebeneinander gestellt. Ihre Fronten werden so ausgerichtet, dass ein Schnittpunkt in ca. 1 m Entfernung entsteht. Das Mikrofon wird im Abstand von ca. 1 m mittig vor den beiden Lautsprecher platziert.

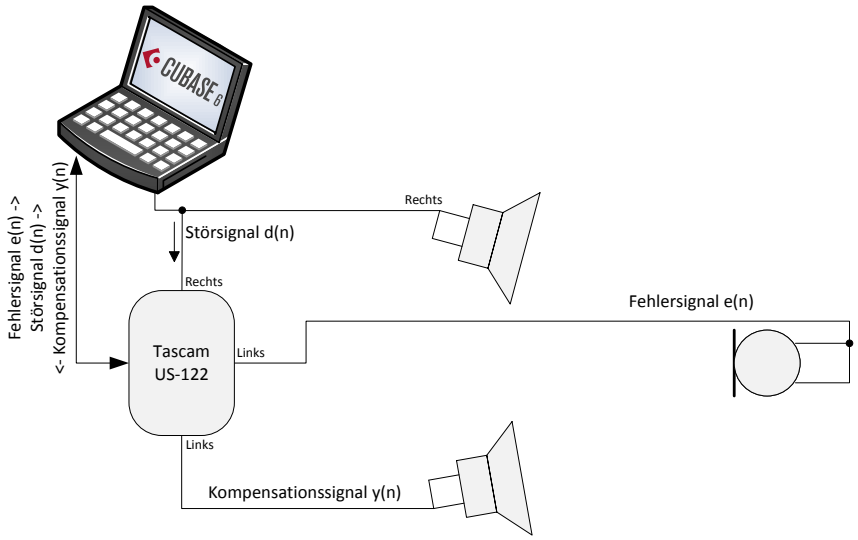

Abbildung 6.1.: Versuchsaufbau mit zwei Lautsprechern und einem Mikrofon

Das Störgeräusch wird mit einer beliebigen Audiosoftware über den Stereoausgang des Laptops/PCs wiedergegeben. Zu beachten ist, dass das Störgeräusch in Stereo vorliegen muss. Hierbei wird der linke Kanal mit dem rechten Audioeingang des Tascam–Interface und der rechte Kanal direkt mit einem der Lautsprecher verbunden. Wird das Störgeräusch nicht manuell erzeugt, wird anstelle des Stereoausgangs ein weiteres Mikrofon an den rechten Audioeingang des Tascam–Interface angeschlossen, um das Störsignal aufzunehmen.

Das Kompensationssignal wird aus dem Plug-in generiert und über den linken Kanal an das Tascam–Interface weitergereicht. Von dort geht es über den linken Audioausgang an den anderen Lautsprecher.

Das Fehlersignal, aufgenommen über das Mikrofon, gelangt über den linken Kanal durch das Tascam–Interface in das Plug-in. Für spätere Analysen des Signals ist es sinnvoll, dass Signal mittels separater Software (z.B. Audacity) aufzuzeichnen.

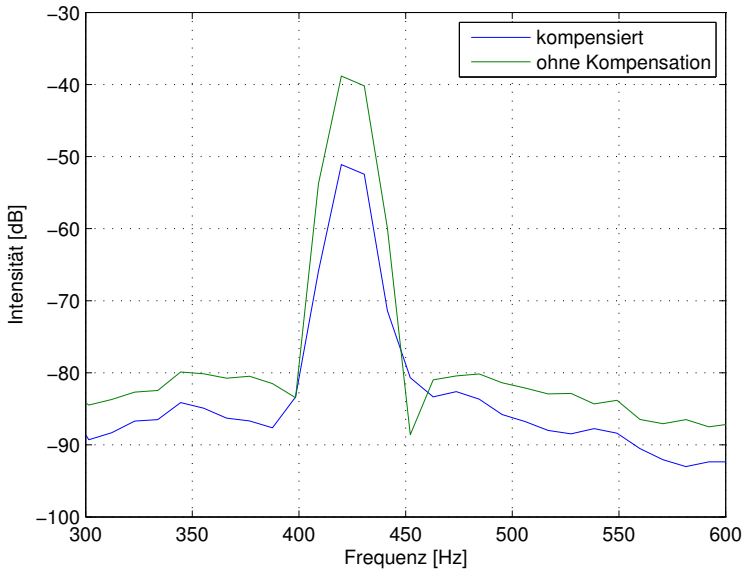

Abbildung 6.2.: Optimale Überlagerung bei einem Sinussignal in einem Raum, wenn das Mikrofon so mittig wie möglich zwischen den Lautsprecher positioniert wird

6.5. Optimalfall

Um eine Einschätzung zu bekommen, in welchem Maß Störgeräusche generell in einem Raum gedämpft werden können, bedarf es ebenfalls zweier Lautsprecher und eines Mikrofons. Diese werden wie in 6.4 angeordnet. Über einen der beiden Lautsprecher wird ein Audiosignal

abgespielt, über den zweiten Lautsprecher das an der Zeitachse gespiegelte Audiosignal. Mit dem Mikrofon kann nun aufgenommen werden, wie gut sich der Schall im Raum überlagert und kompensiert.

6.5.1. Optimale Überlagerung bei einem Sinussignal

Das Beispiel mit dem Sinussignal aus 5.1.1 zeigt, dass sich der Schall im Raum sowohl hör- als auch messbar verringern lässt. Die Grafik 6.2 veranschaulicht, dass das Sinussignal um mehr als -10 dB abgeschwächt wird.

6.5.2. Optimale Filterung des Klimaanlagensignals

Ein Versuch mit dem Klimaanlagensignal zeigt, dass sich selbst im optimalen Fall, wenn das Mikrofon im Bereich zwischen den Lautsprechern steht, nicht alle Frequenzen dämpfen lassen (siehe Abbildung 6.3).

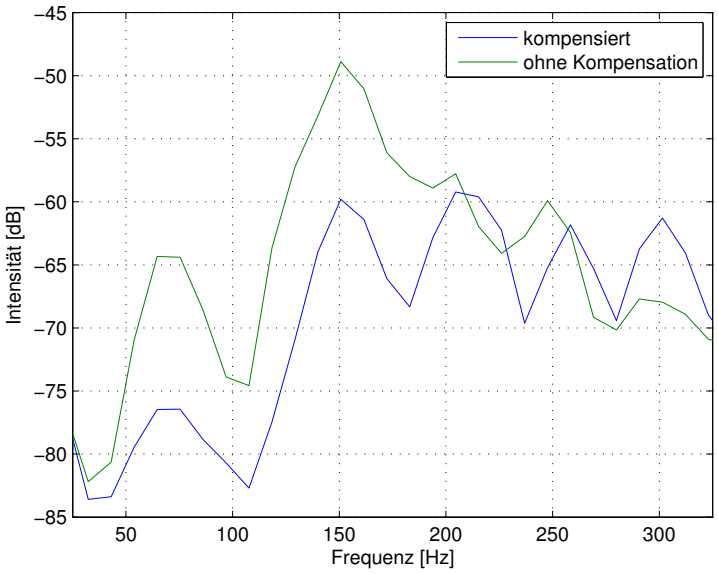

Abbildung 6.3.: Kompensation des Klimaanlagensignals im Raum unter optimalen Bedingungen; zu erkennen ist die Verstärkung bei z.B. 150 Hz und die Abschwächung bei 300 Hz durch die Kompensation

Reflektionen von Wänden, deren Laufzeit und Stärke variieren können, haben zur Folge, dass bestimmte Frequenzbereiche unverändert bleiben oder verstärkt werden. In diesem Versuch macht sich das z.B. in dem Bereich um 300 Hz bemerkbar. Hier wird das Störsignal um ca. +5 dB angehoben, während es zwischen 50 und 200 Hz gedämpft wird.

6.6. Ergebnis mit dem Cubase–Plug-in

In diesem Abschnitt werden die Tests mit dem Cubase–Plug-in beschrieben. Diese zeigen in unterschiedlichen Anwendungsfällen, was das Plug-in leisten kann.

6.6.1. Filterleistung mit einem Sinussignal

Der Test mit einem Sinussignal erreicht ein gutes Maß an Dämpfung von -15 dB bis -30 dB (siehe Abbidlung 6.4). Dieses ist vergleichbar mit der Filterleistung aus 5.2.1, wo eine Dämpfung von -20 dB innerhalb weniger Milisekunden erreicht wird. Das Plug-in bedarf für die gleiche Leistung zwischen 20 und 30 Sekunden, abhängig von der Größe des μ_B.

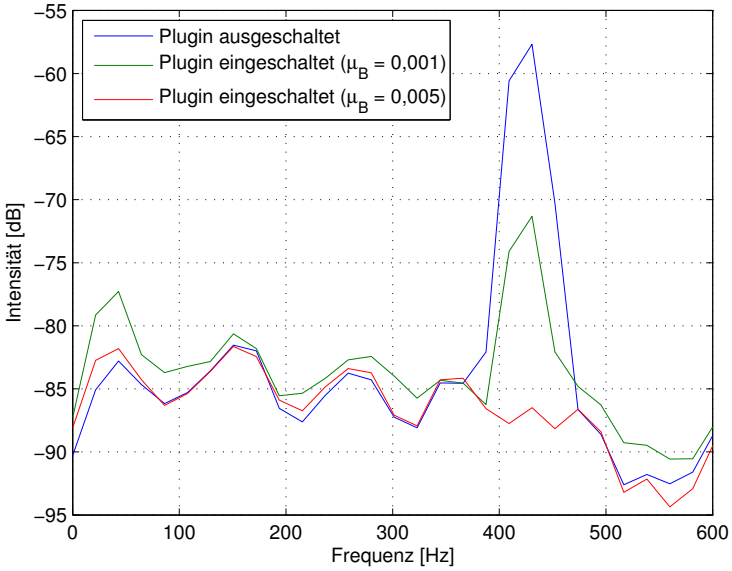

Abbildung 6.4.: Vergleich der Filterleistungen des Plug-ins bei unterschiedlichem Konvergenzfaktor

Die zeitliche Differenz kann sich aus unterschiedlichen Bedingungen für Plug-in und Protoyp ergeben. Während sich für die Prototypen eindeutig definieren lässt, welches Signal mit wieviel

Leistung in den Algorithmus fließt, ist dies mit dem aktuellen Aufbau des Versuchs kaum realisierbar.

6.6.2. Filterleistung mit ansteigenden und abfallenden Sinussignalen

Dieser Test soll zeigen, wie schnell sich das Plugin an Lautstärkeänderungen anpassen kann. Als Störsignal dient hierbei erneut das Sinussignal mit 424 Hz, welches linear für 30 Sekunden ansteigt und anschließend wieder abfällt. Auffällig sind die Amplituden der Störsignalnäherung am Anfang und Ende sowie in der Mitte der Abbildung 6.5. Diese treten auf, wenn das Störsignal seine Frequenz ändert oder es zu Lautstärkunterschieden kommt, wie es in Abbildung 6.5 zwischen Sekunden 200 und 210 der Fall ist.

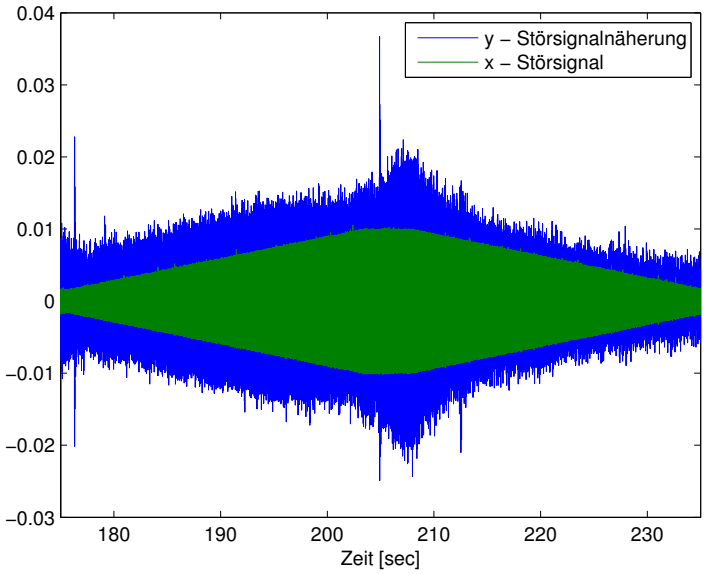

Abbildung 6.5.: Anpassungsfähigkeit des Plug-ins bei Lautstärkeänderung

6.6.3. Filterleistung mit mehrtönigen Sinussignalen

Tests mit mehrtönigen Sinussignalen zeigen in Bezug auf die Filterleistung unterschiedliche. Bei Signalen, die sich aus zwei oder drei Frequenzen zusammensetzen, werden nicht alle Frequenzanteile im gleichen Maß gefiltert.

Abbildung 6.6 zeigt das Verhalten bei einem Signal, bestehend aus zwei Sinusfrequenzen (141,37 Hz und 424,26 Hz) und einem μ_B von 0,005. Zu erkennen ist, dass tiefere Signale um -30 dB gedämpft, höhere wiederum gerinfügig verstärkt werden.

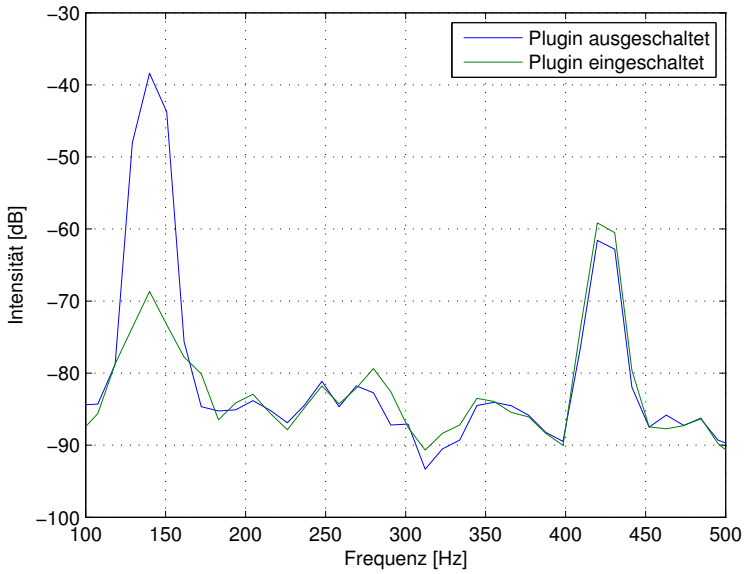

Abbildung 6.6.: Leistung des Plug-ins bei Mehrfrequenzsignalen, bei denen nur der niederfrequente Anteil gefiltert wird

Bei einem Signal aus drei Frequenzen (333 Hz, 440 Hz und 888 Hz) zeigt sich bei gleichbleibendem Konvergenzfaktor sogar, dass das tiefste Signal, anders als zuvor, um 10 dB angehoben wird und die beiden höheren Signale geringfügig abgeschwächt werden (siehe Abbildung 6.7).

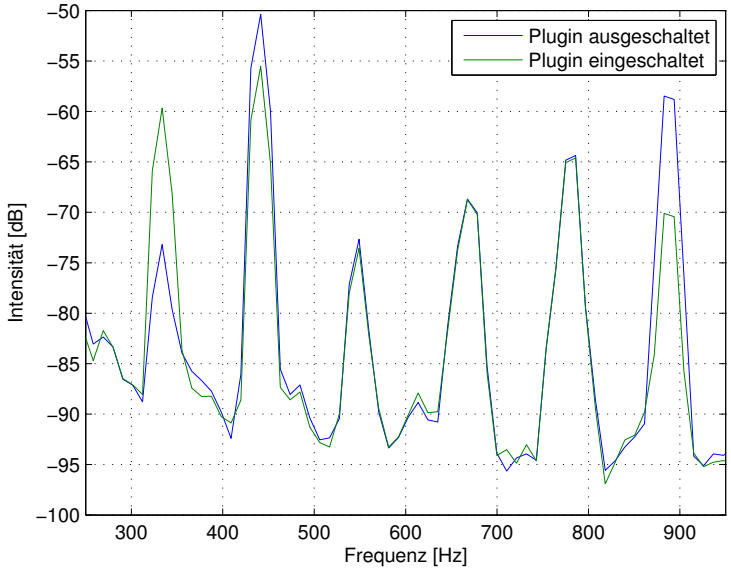

Abbildung 6.7.: Leistung des Plug-ins bei Mehrfrequenzsignalen, bei denen nur der hochfrequente Anteil geringfügig gefiltert und der Frequenzbereich um 300 Hz deutlich verstärkt wird

6.6.4. Filterleistung mit Instrumenten

Für die folgenden Versuche dienen synthetische, sinusähnliche Instrumente als Störsignalquelle. Diese Störsignale sind komplexer als die mehrtönigen Sinussignale aus Punkt 6.6.3.

Flöte

Bei der Flöte liegt der Dämpfungsgrad in der Mitte der drei Instrumente. Frequenzen um 500 Hz und 1570 Hz werden um bis zu -7 dB gedämpft. Frequenzen um den Bereich 1050 Hz wiederum um ca. 7 dB angehoben (siehe Abbildung 6.8).

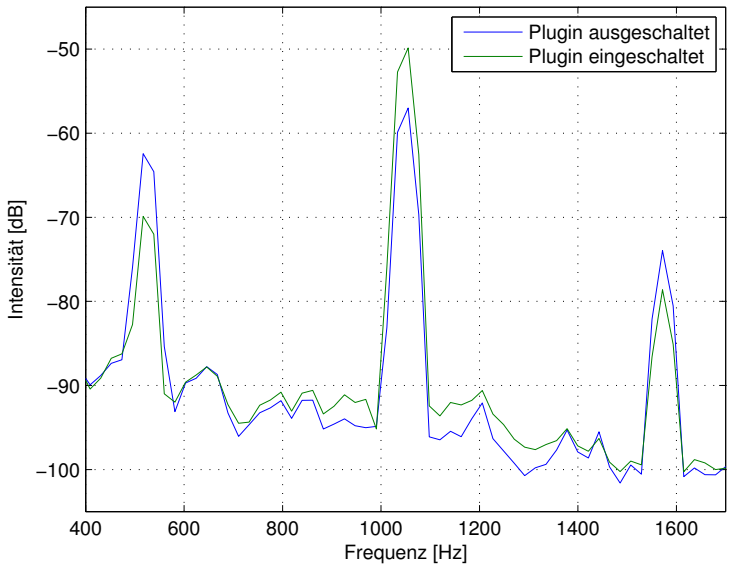

Abbildung 6.8.: Filterleistung mit Flöte als Störsignal

Trompete

Die Trompete lässt sich von den drei Instrumenten am besten kompensieren, auch wenn die Kompensationsleistung lediglich -8 dB bei einer Frequenz von 800 Hz beträgt. Wie schon bei der Flöte (Abbildung 6.8) wird hier das Signal um den Bereich von 1000 Hz um ca. 7 dB angehoben (siehe Abbildung 6.9).

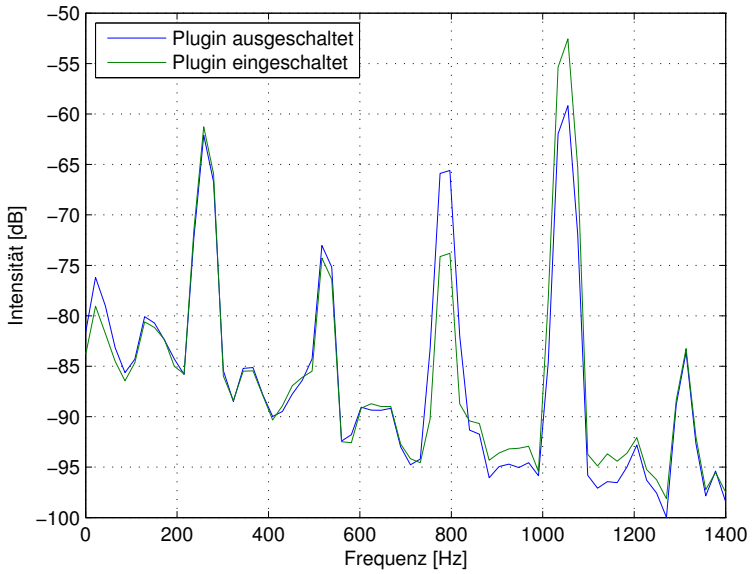

Abbildung 6.9.: Filterleistung mit Trompete als Störsignal

Bratsche

Die Bratsche lässt sich von den drei Instrumenten am schlechtesten kompensieren. Das Maximum an Dämpfung liegt hier bei ca. -2 dB im Bereich von 525 Hz und 775 Hz (siehe Abbildung 6.10). Grund dafür könnte sein, dass das Frequenzspektrum der Bratsche am breitesten von allen 3 Instrumenten ist und Frequenzen bis zu 10 kHz beinhaltet. Positiv zu vermerken ist, dass es bei diesem Test kaum zu konstruktiven Interferenzen kam.

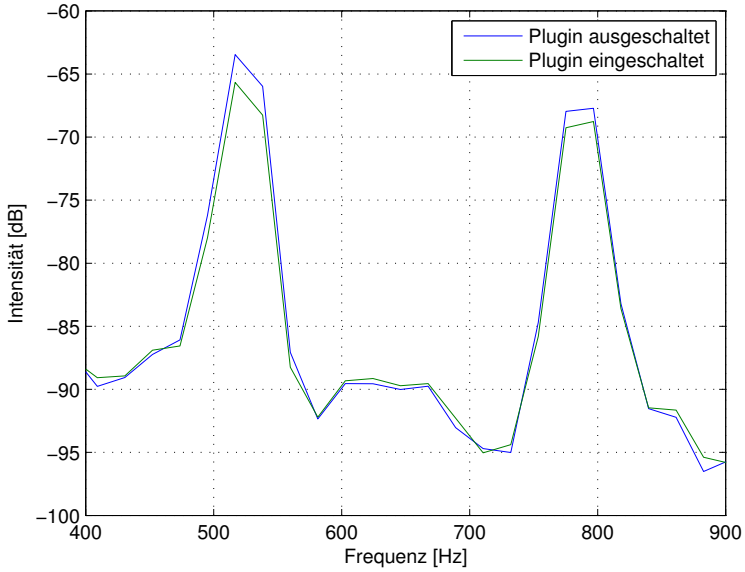

Abbildung 6.10.: Filterleistung mit Bratsche als Störsignal

Grund für die mangelhafte Kompensation sowohl bei den Instrumenten als auch bei den mehrfrequenten Signalen kann der Konvergenzfaktor μ_B sein, der für komplexere Signale größer gewählt werden muss. Das Plug-in erreicht jedoch keine Konvergenz, wenn μ_B deutlich größer als 0,005 ist.

6.6.5. Filterleistung mit der Klimaanlagenaufnahme

Das Beispiel mit dem Klimaanlagensample zeigt, dass sich das Signal der Klimaanlage in diesem Aufbau lediglich um 2 dB im Bereich zwischen 600 Hz und 1600 Hz senken lässt.

Grund hierfür können die in 5.5 erwähnten Startkoeffizienten sein, die bei diesem Test zwischen 0 und 0,1 lagen. Ein weiterer Grund kann auch hier der Versuchsaufbau sein, der schon bei dem Test mit dem Sinussignal 6.6.1, bezogen auf die Konvergenzzeit, für Schwierigkeiten gesorgt hat.

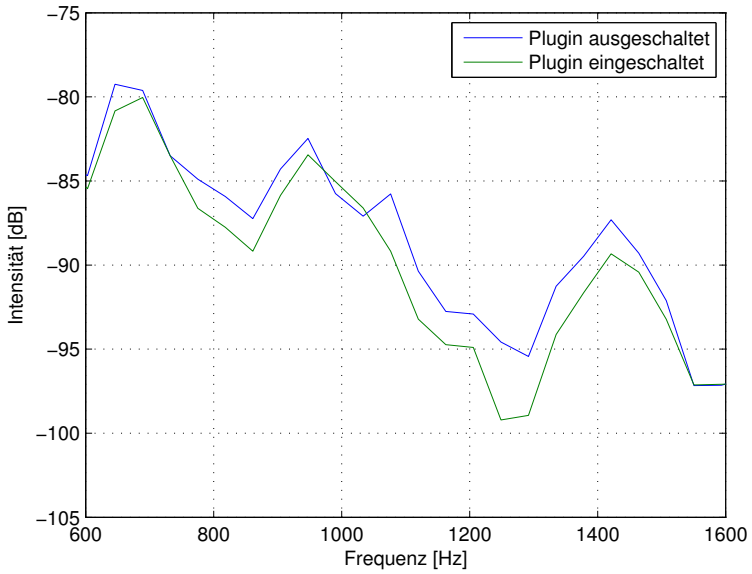

Abbildung 6.11.: Filterleistung des Plug-in bei der Klimaanlagenaufnahme und einem Konvergenzfaktor von 1

7. Bekannte Probleme

7.1. Interferenzen

Das größte Problem ist die in 2.1.1 genannte Überlagerung von Schallwellen, die zwangsläufig entstehen und ein Interferenzmuster erzeugen. Folge dieses Musters ist, dass sich Schallwellen durch andere Schallwellen nie vollständig reduzieren lassen. Das ist energietechnisch auch nicht möglich, da Energie nur umgewandelt wird. Entsprechend wird Schall nur an bestimmten Orten verringert und an anderen Orten wiederum verstärkt.

Generell ist es einfacher, geradlinige Schallquellen zu kompensieren, da dort größere Flächen zur Kompensation entstehen.

7.2. Schallreflektion

Ein weiteres Problem ist die Schallreflektion, die dafür sorgt, dass selbst im optimalen Fall konstruktive und destruktive Interferenzen entstehen. Diese erschweren die Kompensation zusätzlich.

7.3. Implementierung

Es bestehen Unterschiede zwischen der seriellen und blockweisen Implementierung des Algorithmus. Wie in [2, Seite 748, Abschnitt IV.] beschrieben, verhalten sich beide Implementierungen identisch bezogen auf Filterleistung, Geschwindigkeit und Genauigkeit, wenn $\mu_B = \mu \cdot L$ ist. Dies gilt, solange 3.6 und 3.21 dabei nicht verletzt werden.

Dennoch sind in allen drei Bereichen Unterschiede festzustellen, die ein Hinweis auf eine fehlerhafte Implementierung sein können.

7.4. Cubase LE 5 und VST3–Plug-in API

Während der Verwendung des Plug-ins kam es zu Problemen. Es traten Situationen auf, in denen das zweimalige Übersetzen des Plug-ins ohne zwischenzeitliche Änderungen im Quellcode zu deutlichen Laufzeitunterschieden führte.

Ebenso gab es Fehler während der Initialisierung des Plug-ins, die auf eine fehlerhafte Nutzung des Speichers hindeuten. Folge dieses Fehlers war das Beenden von Cubase mit einer Fehlermeldung.

Weitere Probleme äußerten sich durch Anhalten des Betriebssystems, hier Windows 7 Home Premium. Mögliche Ursache könnten die Treiber des Tascam Audiointerface sein, die nur für Windows Vista ausgelegt sind.

8. Fazit

Die Arbeit hat gezeigt, dass es generell möglich ist, mit der blockweisen Implementierung eines adaptiven Filters, Störgeräusche zu filtern. Versuche mit verschiedenen Signalarten zeigten die Unterschiede zwischen der blockweisen und seriellen Implementierung auf, die sich unter anderem in Rechengeschwindigkeit, Filterleistung und Konvergenzgeschwindigkeit bemerkbar machten. Anders als bei der seriellen Implementierung ist es bei der blockweisen möglich, den Algorithmus in Echtzeit an einem PC oder Laptop einzusetzen. Dennoch erreichte die serielle Implementierung bessere Ergebnisse in Filterleistung und Konvergenzverhalten. Entsprechend ließen sich bei Signalen wie z.B. dem der Klimaanlage höchstens tiefe Frequenzen filtern.

Auch die Versuche zur Schallkompensation im Raum haben gezeigt, dass mehrere Dinge beachtet werden müssen. Die Tatsache, dass der Schall aus der gleichen Richtung kommt, reicht nicht immer aus, um eine optimale Kompensationsleistung zu erhalten. Es zeigte sich, dass selbst bei optimaler Ausrichtung von Lautsprecher und Mikrofon kein Punkt in einem Raum zu finden war, an dem das komplexe Signal einer Klimaanlage nahezu vollständig kompensiert werden konnte.

Abschließend bleibt zu sagen, dass es noch weiterer Lösungsansätze bedarf, um Störgeräusche wie das einer Klimaanlage oder Ähnliche in einer solchen Dimension effektiv zu kompensieren.

A. Quellenverzeichnis

- Abbildung 3.1 aus [5, Seite 19]
- Abbildung 3.2 aus [2, Seite 745]
- Abbildung 4.1 abgerufen am 16.08.2012
  http://tascam.com/content/images/universal/product_detail/311/medium/US-122_right.jpg
- Aus Abbildung 6.1 abgerufen am 28.09.2012
  http://upload.wikimedia.org/wikipedia/commons/5/52/Cubase_6_logo.svg

B. Inhalt der CD

Die CD beinhaltet

- das VST3–Plug-in
- den Quellcode für das VST3–Plug-in
- sämtliche Matlabskripte zur Auswertung der Algorithmen
- Audiodateien für die Matlabskripte

Literaturverzeichnis

[1] AKG (Veranst.): *C 3000B Bedienungshinweis.* – URL http://www.akg.com/mediendatenbank2/psfile/datei/69/c3000b4055c43a7e2e7.pdf

[2] Clark, G.: Block Implementation of Adaptive Digital Filters. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 29 (1981). – URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1163603

[3] Honda: *Official Honda Legend Site | Specifications.* – URL http://legend.honda.com.au/about-the-car_specifications.aspx

[4] Lueg, P.: *Process of Silencing Sound Oscillations.* Juni 1936. – URL http://www.google.com/patents/US2043416.pdf. – Patent US 2043416

[5] Matthies, J.: *Aktive Geräuschkompensation mit einer FPGA-basierten Signalverarbeitungsplattform*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, Februar 2008. – URL http://opus.haw-hamburg.de/volltexte/2008/528/pdf/MA_I_Matthies_15.02.2008.pdf

[6] Pressetext: *"Antischall" erlaubt Windrädern höhere Stromausbeute.* – URL http://www.pressetext.com/news/20080801020

[7] Tascam (Veranst.): *US–122 Benutzerhandbuch.* – URL http://tascam.com/content/downloads/products/311/US122_Eng.pdf

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 4. Oktober 2012 Thorben Vornholz