

# Real-Time Audio Equalizer mittels Matlab und Simulink

Studienarbeit WS02/03

Hochschule Rapperswil HSR

T. Balczarczyk      R. Guldener

18. Februar 2003

# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabenstellung</b>	<b>1</b>
<b>2</b>	<b>Grundlagen</b>	<b>2</b>
2.1	Psychoakustik . . . . .	2
2.2	Die schnelle Fouriertransformation FFT . . . . .	3
2.3	Diskrete lineare und zyklische Faltung . . . . .	3
2.4	Echtzeitverarbeitung durch Anwendung der schnellen Faltung	5
2.4.1	Segmentierung mit Overlap-Add . . . . .	6
2.4.2	Segmentierung mit Overlap-Save . . . . .	7
<b>3</b>	<b>Praktische Arbeiten I</b>	<b>9</b>
3.1	Grundprinzip der Schaltung . . . . .	9
3.2	Umsetzung in Simulink mit der Overlap-Add und -Save Methode	10
3.2.1	Overlap-Add . . . . .	10
3.2.2	Overlap-Save . . . . .	11
3.2.3	Erstellen der Übertragungsfunktion . . . . .	11
3.3	Benutzte Komponenten in Simulink . . . . .	13
3.3.1	From Wave Device . . . . .	13
3.3.2	To Wave Device . . . . .	13
<b>4</b>	<b>Praktische Arbeiten II</b>	<b>15</b>
4.1	Messungen mit der Soundkarte . . . . .	15
4.1.1	Messmittelliste . . . . .	15
4.1.2	Systemverzögerung . . . . .	15
4.2	Frequenzabtasttechnik . . . . .	16
4.3	Stereo FFT . . . . .	17
4.4	Loudness . . . . .	18
<b>5</b>	<b>Schlusswort</b>	<b>19</b>
<b>A</b>	<b>Messungen</b>	<b>21</b>
<b>B</b>	<b>Übertragungsfunktion</b>	<b>26</b>

# Abbildungsverzeichnis

2.1	Kurven gleicher Lautstärke . . . . .	2
2.2	<b>Lineare Faltung.</b> Die Länge der Impulsantwort $h[n]$ beträgt 5, diejenige von $x[n]$ ist 9. . . . .	4
2.3	Diskrete zyklische Faltung durch die FFT . . . . .	5
2.4	<b>Diskrete zyklische Faltung durch die FFT.</b> Beide Signale wurden vorhergehend noch einem <i>Zero-Padding</i> unterzogen. . . . .	6
2.5	<b>Overlap-Add Algorithmus.</b> Die Sequenz $y[n]$ ist das Resultat der Faltung von $x[n]$ mit einem Filter der Länge 5. In diesem Beispiel ist $h[n] = 0.2$ für $n = 0, \dots, 4$ . Die Blocklänge ist 10, der Overlap dadurch 4. . . . .	7
2.6	<b>Overlap-Save Algorithmus.</b> Die Sequenz $y[n]$ ist das Resultat der Faltung von $x[n]$ mit einem Filter der Länge 5. In diesem Beispiel ist $h[n] = 0.2$ für $n = 0, \dots, 4$ . Die Blocklänge ist 10, der Overlap dadurch 4. . . . .	8
3.1	<b>Grundprinzip des Equalizers.</b> Die <i>Window Function</i> ist für den Frequenzbereich gedacht als Gewichtung einzelner bestimmter Frequenzen, und somit nicht zu verwechseln mit der Problematik des <i>Leakage</i> . . . . .	9
3.2	<i>Unbuffer</i> und <i>Buffer</i> in Simulink . . . . .	10
3.3	Schaltung Overlap-Add . . . . .	10
3.4	Schaltung Overlap-Save . . . . .	11
3.5	<b>Auswirkungen des Zero-Padding.</b> Oberes Spektrum wurde falsch, unteres richtig mit Nullen aufgefüllt. Weitere Erklärungen auf S. 26f. . . . .	12
3.6	<i>From Wave Device</i> liest Daten in Echtzeit ab der Soundkarte. . . . .	13
3.7	Parameter-Einstellungen <i>From Wave Device</i> . . . . .	13
3.8	<i>To Wave Device</i> schreibt Daten in Echtzeit in den Ausgabebuffer der Soundkarte. . . . .	14
3.9	Parameter-Einstellungen <i>To Wave Device</i> . . . . .	14
4.1	<b>Systemverzögerung.</b> Einspeisung eines Sägezahnsignals aus dem Frequenzgenerator und messen der Systemverzögerung mit dem Oscilloscope. . . . .	16
4.2	Frequenzabtasttechnik . . . . .	17

A.1	Frequenzgang der Soundkarte . . . . .	21
A.2	<b>Summe der Messungen.</b> Da die Messwerte in dB gemessen wurden, kann man nicht einfach eine Addition durchführen. Die Berechnung muss im linearen Bereich durchgeführt werden. Somit ergibt sich ein Wert von 15,5 dB. . . . .	22
A.3	Frequenzgänge 90 Hz bis 500 Hz . . . . .	23
A.4	Frequenzgänge 1 kHz bis 8 kHz . . . . .	24
A.5	Frequenzgänge 16 kHz und 22 kHz . . . . .	25
B.1	<b>Herstellung des Übertragungsspektrums.</b> Oberes Spektrum ist mit 1024 Werte pro Frame, unteres mit 512 Werte pro Frame hergestellt worden. Beim oberen Spektrum sind doppelt so viele Frequenzstützpunkte vorhanden wie beim unteren Spektrum. Somit ist es möglich, einen Zwischenwert einzufügen, welcher ein <i>freundlicheres</i> Spektrum ergibt. . . . .	26
B.2	<b>Impulsantwort der Übertragungsfunktion.</b> Wenn die Impulsantwort der Übertragungsfunktion (oberstes Bild) am Ende mit Nullen aufgefüllt wird (mittleres Bild), entsteht ein <i>wildes</i> Spektrum (Abbildung 3.5 oberes Spektrum). Richtigerweise muss die Impulsantwort in der Mitte mit Nullen aufgefüllt werden, damit sie nicht unterbrochen wird. Um weiter das Gesetz der Overlap-Add/Save Methode zu erfüllen, darf die Impulsantwort nicht länger als $L$ Werte sein. Dies wird erreicht, indem der rechte Teil nach links an den Anfang geschoben und anschliessend das Frame mit Nullen aufgefüllt wird (unterstes Bild). . . . .	27

# Kapitel 1

## Aufgabenstellung

Es soll ein Echtzeit-Audio-Equalizer mit den Programmen Matlab und Simulink gebaut werden. Die Bezeichnung *Echtzeit* soll nicht allzu wörtlich genommen werden; Verzögerungen bis zu einer Sekunde liegen im Toleranzbereich.

Matlab und Simulink verfügen über geeignete Mittel zur Eingabe und Ausgabe von Audio-Streams; diese sollten auch verwendet werden. Der Kern der Aufgabenstellung besteht darin, die eingelesenen Audio-Streams zu bearbeiten, ausgewählte Frequenzbereiche zu verstärken oder zu dämpfen und anschliessend auf den Ausgang der Soundkarte zu senden. Am Eingang der Soundkarte ist ein CD-Player angeschlossen und ausgangsseitig ein Verstärker, welcher die Audio-Streams über Lautsprecher ausgibt.

# Kapitel 2

## Grundlagen

### 2.1 Psychoakustik

Das Gebiet der Psychoakustik beschreibt, wie das menschliche Gehör die Lautheit von Tönen empfindet, was durch Hörtests ermittelt werden kann. Das Gehör reagiert auf Frequenzen zwischen 20 Hz und 20 kHz, am empfindlichsten ist es zwischen 2 kHz und 4 kHz. Abbildung 2.1 zeigt diesen Zusammenhang deutlich. Das menschliche Gehör besitzt die Eigenschaft, dass leise

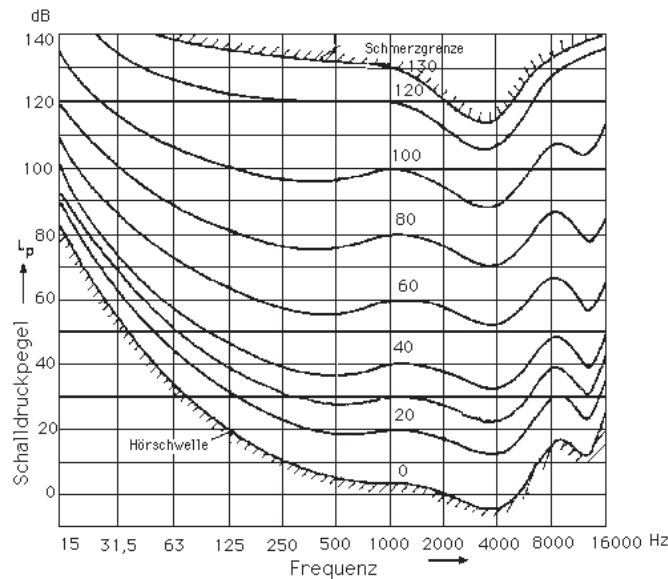


Abbildung 2.1: Kurven gleicher Lautstärke

Töne in der Nähe eines lauten Tones nicht mehr so gut gehört werden, wie wenn der laute Ton nicht vorhanden wäre. Dies kann so weit führen, dass leise Töne vom Ohr gar nicht mehr wahrgenommen werden. Diese Eigen-

schaft wird *Maskierung* genannt und wurde bei Experimenten beobachtet. Dabei zeigte sich auch, dass die Frequenzauflösung des menschlichen Gehörs beschränkt ist. Diese variiert von wenigen 100 Hz bei niedrigen Frequenzen bis über 4 kHz bei hohen Frequenzen. Dadurch kann das hörbare Spektrum in kritische Bänder unterteilt werden, die das Auflösungsvermögen des Ohres widerspiegeln. Tabelle 2.1 gibt eine Übersicht der kritischen Bänder.

MPEG/Audio nutzt zur Komprimierung die Eigenschaften des menschlichen Gehörs aus, das heisst auch, dass MPEG/Audio eine verlustbehaftete Komprimierungsmethode ist.

Band Nummer	Frequenz in Hz	Band Nummer	Frequenz in Hz	Band Nummer	Frequenz in Hz
0	50	9	940	18	3840
1	95	10	1125	19	4690
2	140	11	1265	20	5440
3	235	12	1500	21	6375
4	330	13	1735	22	7690
5	420	14	1970	23	9375
6	560	15	2340	24	11625
7	660	16	2720	25	15375
8	800	17	3280	26	20250

Tabelle 2.1: Kritische Bänder des menschlichen Gehörs. Die Frequenzen sind am oberen Ende des Bandes.

## 2.2 Die schnelle Fouriertransformation FFT

Bis anhin gilt die Annahme, dass die schnelle Fouriertransformation (FFT=Fast Fourier Transform) perfekt ist; sie liefert für zeitdiskrete Signale der Länge  $n$  fouriertransformierte der Länge  $N$ . Gemäss der Theorie liefert die inverse Fouriertransformation (IFFT=Inverse FFT) wieder das ursprüngliche Signal. Bei Verwendung eines Computers gibt es jedoch Rundungsfehler, die die Rechengenauigkeit beeinträchtigen. Dann liefert die IFFT nicht mehr das ursprüngliche Signal, sondern ein Signal mit einem komplexen Anteil, dessen Imaginärteile für die Weiterverarbeitung für uns nicht von Interesse sind.

## 2.3 Diskrete lineare und zyklische Faltung

Die Faltung von nicht-zeitdiskreten Signalen kann man schreiben als

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau \quad .$$

Sind die Signale dagegen zeit-diskret, vereinfacht sich das Integral zu einer Summe:

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

Betrachten wir nun 2 Signale,  $h1[n]$  und  $x1[n]$ , wie in Abbildung 2.2 dargestellt. Die Auswertung von (2.1) ergibt  $y1[n]$ , das lässt sich mit ein wenig Aufwand auch auf dem Papier bewerkstelligen. Bei kurzen Signalen, deren Länge kleiner als 50 ist, kann man eine solche Faltung in der Zeit ohne Bedenken ausführen. Wenn man hingegen einen kontinuierlichen Datenstrom verarbeiten muss, wie in dieser Studienarbeit, dann wählt man vorzugsweise eine grössere Blocklänge, schliesslich will man den Bus nicht zu sehr strapazieren. Bei langen Blocklängen wird solch eine Berechnung immer länger dauern, denn die Komplexität dieses Algorithmus beträgt  $O(M^2)$ , wenn  $M$  die Blocklänge ist. Daher überlegt man sich bald einmal den Einsatz der FFT und IFFT, da deren Komplexität sich im Bereich  $O(M \cdot \log_2(M))$  befindet. Vergleicht man Abbildung 2.2 mit Abbildung 2.3, dann stellt man

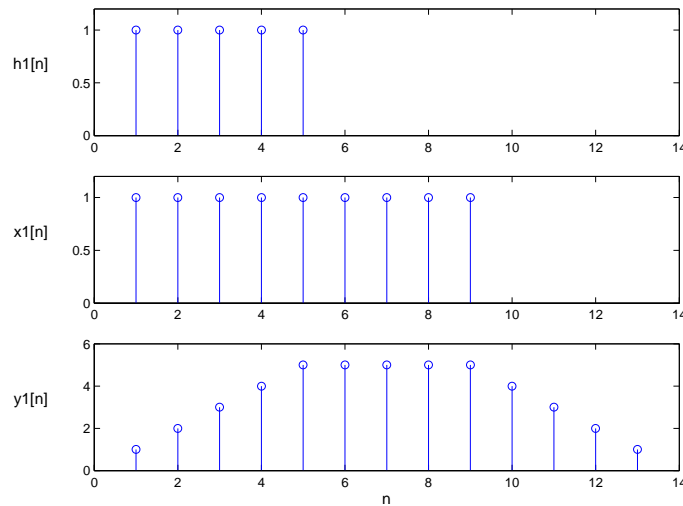


Abbildung 2.2: **Lineare Faltung**. Die Länge der Impulsantwort  $h[n]$  beträgt 5, diejenige von  $x[n]$  ist 9.

einige Unterschiede fest. In Abbildung 2.3 wurde über den Frequenzbereich das Ausgangssignal  $y[n]$  berechnet; was dabei herausgekommen ist nennt man *zyklische Faltung*. Der Teil, der nach hinten zu lang ist, wird vorne in das Signal eingeschoben bzw. in den Anfang addiert. Abbildung 2.3 illustriert diesen Vorgang sehr schön. Die diskrete zyklische Faltung wird für zwei Signal wie folgt definiert:

$$y[n] = x[n] \oplus h[n]$$

Ganz allgemein lässt sich sagen:

Die Multiplikation diskreter Frequenzfunktionen entspricht im Zeitbereich einer diskreten zyklischen Faltung.<sup>1</sup>

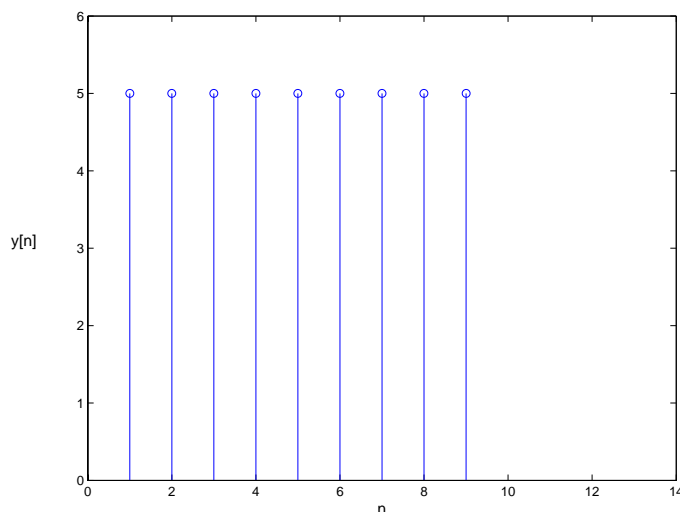


Abbildung 2.3: Diskrete zyklische Faltung durch die FFT

Solch eine Signalverformung, wie in Abbildung 2.3 dargestellt, kann dadurch vermieden werden, indem dem Signal und der Impulsantwort Nullen angefügt werden, bis die Signale  $N + L - 1$  lang sind. Die Rechnung entspricht dann immer noch der einer zyklischen Faltung, aber mit dem Unterschied, dass die an den Anfang addierte Sequenz Null ist und demnach das Signal nicht verändert. Abbildung 2.4 auf Seite 6 zeigt diesen Zusammenhang deutlich.

## 2.4 Echtzeitverarbeitung durch Anwendung der schnellen Faltung

Das Kapitel wurde teilweise aus [1] S. 8-2ff übernommen. Wird die Faltung mittels FFT implementiert, d.h. Eingangssignal und Übertragungsfunktion in den Frequenzbereich transformiert und anschliessend multipliziert und wieder in den Zeitbereich zurücktransformiert, so erhält man die zyklische Faltung des Eingangssignals mit der Übertragungsfunktion. Man muss noch beachten, dass für eine Multiplikation im Frequenzbereich das Eingangssignal und die Übertragungsfunktion auf die gleiche Länge gebracht werden müssen, da sonst die Matrizen/Vektoren nicht miteinander multipliziert werden können.

Für die Echtzeitverarbeitung eines kontinuierlichen Datenstroms ist das so entstandene Ausgangssignal unbrauchbar, die zyklische Faltung ist für eine solche Verarbeitung nicht zu gebrauchen. Warum?

---

<sup>1</sup>Aus [2], S. 5-28

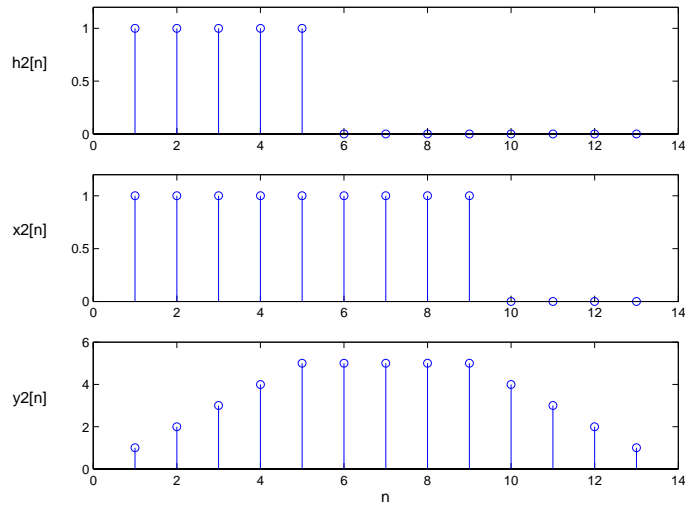


Abbildung 2.4: **Diskrete zyklische Faltung durch die FFT.** Beide Signale wurden vorhergehend noch einem *Zero-Padding* unterzogen.

Die FFT weist eine Periodizität<sup>2</sup> auf. Will man eine Faltung mittels FFT realisieren, so erhält man eine zyklische Faltung, was heisst, dass das Ausgangssignal falsche Werte enthält. Das zeigt sich darin, dass das Ende des Ausgangssignals an dessen Anfang addiert wird; für eine Filterung, Faltung oder Korrelation ist dieser Effekt nicht erwünscht.

Man kann diesen Effekt beseitigen, indem man dem Eingangssignal und der Übertragungsfunktion Nullen hinzufügt, und zwar genau  $L + N - 1$ . Dabei ist  $L$  die Länge des Filters oder eben der Übertragungsfunktion und  $N$  ist die Länge des Eingangssignals. Wenn man jetzt wieder die Faltung mittels der FFT durchführt, erhält man wieder eine zyklische Faltung aber mit dem Effekt, dass die an den Anfang addierte Sequenz Null ist und demnach das Ausgangssignal nicht verändert wird; man spricht auch von einer nicht-zyklischen oder linearen Faltung.

Die Nachteile dieses Verfahrens ist, dass man die FFT mit längeren Sequenzen rechnen muss. Aber solange die Signale eine Länge von  $2^x$  haben, ist die FFT sehr effizient.

### 2.4.1 Segmentierung mit Overlap-Add

Weil die Faltung jetzt eine lineare Faltung ist, kann durch Addition der Ausgangsblöcke, die aus den Eingangsblöcken errechnet wurden, die Ausgangssequenz berechnet werden. Einziges Problem ist, dass die Ausgangsblöcke länger sind als deren Eingangsblöcke. Man löst das Problem, indem man das Ende des vorherigen Ausgangsblocks mit dem Anfang des aktuellen Ausgangsblocks überlappt. Der Name *Overlap-Add* ist offensichtlich. Mit einer

<sup>2</sup>Ein Theorem der Abtastung besagt: abtasten  $\circ \rightarrow \bullet$  periodisch fortsetzen.

Blocklänge  $N$  und einer Filterlänge  $L$  ( $N > L$ ) beträgt die Überlappung  $L + N - 1$ . Abbildung 2.5 illustriert, wie die Overlap-add Methode funktioniert;  $N = 10$  und  $L = 5$ .

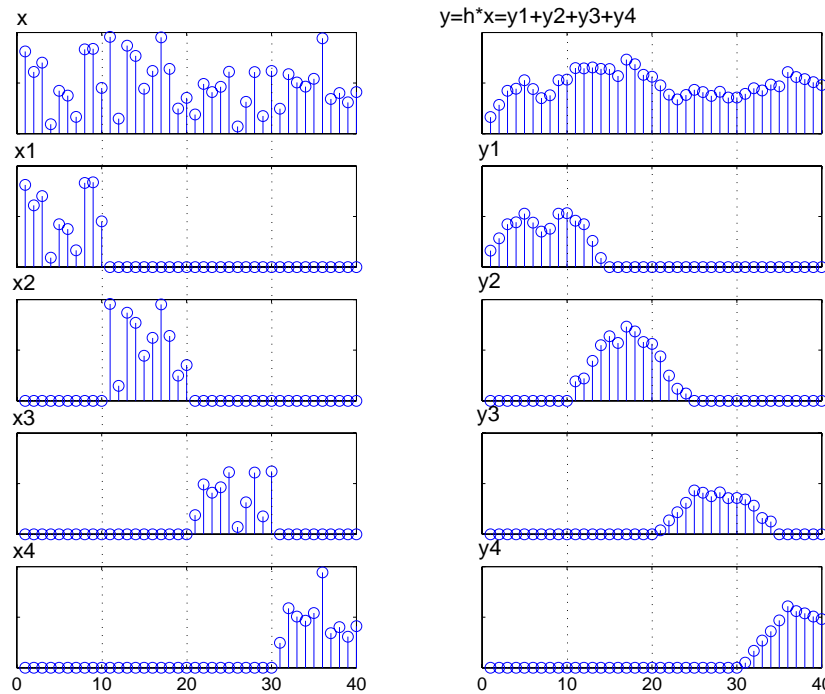


Abbildung 2.5: **Overlap-Add Algorithmus**. Die Sequenz  $y[n]$  ist das Resultat der Faltung von  $x[n]$  mit einem Filter der Länge 5. In diesem Beispiel ist  $h[n] = 0.2$  für  $n = 0, \dots, 4$ . Die Blocklänge ist 10, der Overlap dadurch 4.

### 2.4.2 Segmentierung mit Overlap-Save

Die Idee dieses Verfahrens besteht darin, dass man die Eingangsblöcke überlappt, speichert, den Ausgangsblock berechnet, einige Werte vom Anfang und vom Ende des Ausgangsblocks streicht, und zwar genau die Overlap-Länge. Abbildung 2.6 zeigt die Arbeitsweise des Overlap-Save Algorithmus.

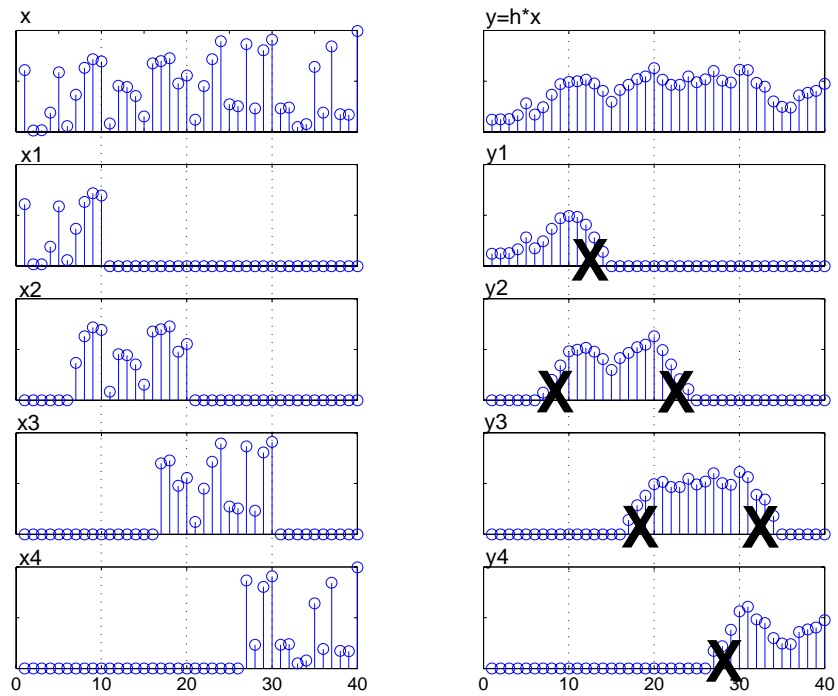


Abbildung 2.6: **Overlap-Save Algorithmus.** Die Sequenz  $y[n]$  ist das Resultat der Faltung von  $x[n]$  mit einem Filter der Länge 5. In diesem Beispiel ist  $h[n] = 0.2$  für  $n = 0, \dots, 4$ . Die Blocklänge ist 10, der Overlap dadurch 4.

# Kapitel 3

## Praktische Arbeiten I

### 3.1 Grundprinzip der Schaltung

Ein Audiosignal wird zuerst durch die Soundkarte des Computers digitalisiert. Anschliessend werden die Daten in den Frequenzbereich mittels einer Fast-Fourier-Transform (FFT) überführt. Im Frequenzbereich wird das Signal mit einem Frequenzspektrum multipliziert und danach mit der Inverse-Fourier-Transform (IFFT) wieder in den Zeitbereich überführt. Anschliessend werden die Daten wieder in ein analoges Signal gewandelt. Abbildung 3.1 zeigt das dazugehörige Prinzip.

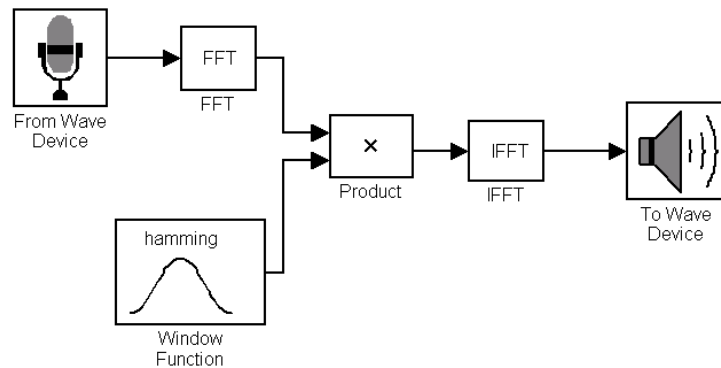


Abbildung 3.1: **Grundprinzip des Equalizers.** Die *Window Function* ist für den Frequenzbereich gedacht als Gewichtung einzelner bestimmter Frequenzen, und somit nicht zu verwechseln mit der Problematik des *Leakage*.

## 3.2 Umsetzung in Simulink mit der Overlap-Add und -Save Methode

### 3.2.1 Overlap-Add

Bei der Overlap-Add Methode wird dem Datenframe vor der FFT eine Anzahl Nullen angehängt. Dies geschieht hier indem die Frames im *Unbuffer* (Abbildung 3.2) wieder zu einem kontinuierlichen Bitstrom zusammengesetzt werden und anschliessend im *Buffer* erneut zu einem Frame der doppelten Länge der ursprünglichen zusammengesetzt werden. Der *Unbuffer* setzt die zweite Hälfte des Frames nicht auf Null, daher muss dies noch im *Overwrite Values* Block (Abbildung 3.3) nachgeholt werden.

Nach der Rücktransformation werden die verschiedenen Frames gemäss der Theorie addiert und ausgegeben.

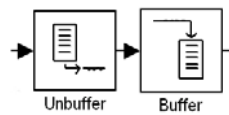


Abbildung 3.2: *Unbuffer* und *Buffer* in Simulink

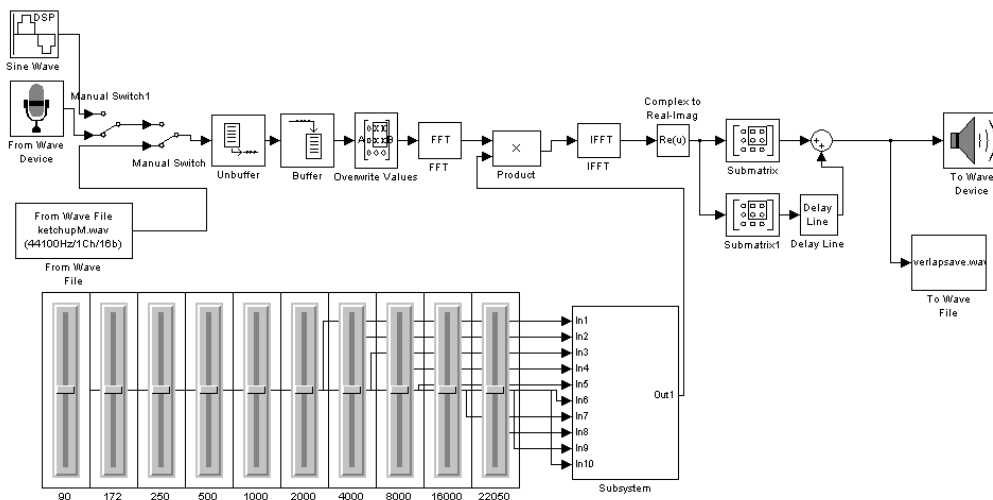


Abbildung 3.3: Schaltung Overlap-Add

### 3.2.2 Overlap-Save

Bei der Overlap-Save Methode wird ausgenutzt, dass die Information des vorhergehenden Frames benutzt wird um die Framegrösse zu verdoppeln. Dies geschieht im *Unbuffer* und *Buffer* Block. Das Originalframe ist 512 Werte gross. Nach dem *Buffer* haben wir eine Grösse von 1024 Werten. Dabei werden die zweiten 512 Werte mit den ersten 512 des vorhergehenden Signals gefüllt. Wegen der zyklischen Faltung werden aber nach der Rücktransformation die ersten 512 Werte unbrauchbar und werden nun im Block *Submatrix* (Abbildung 3.4) gelöscht.

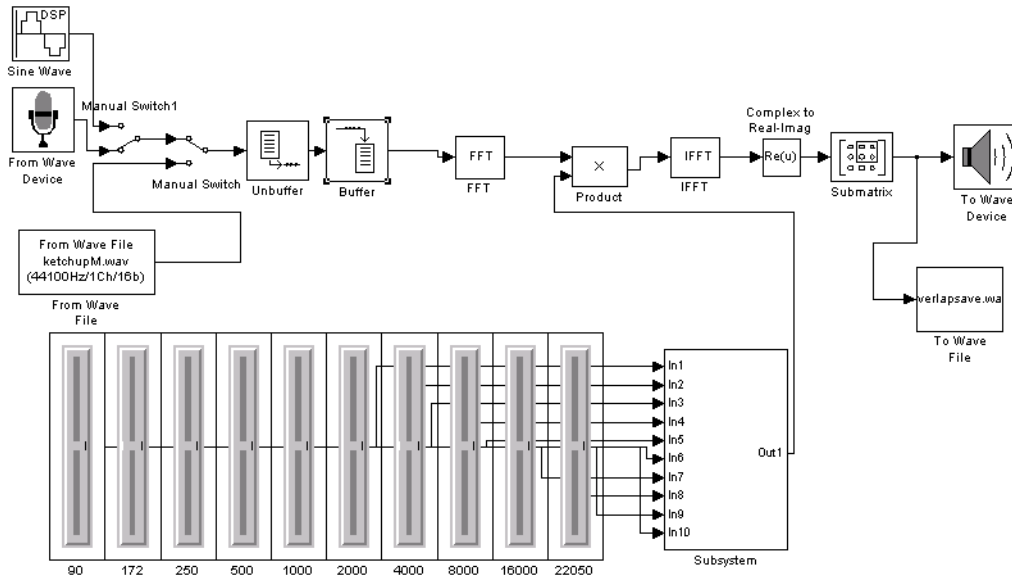


Abbildung 3.4: Schaltung Overlap-Save

### 3.2.3 Erstellen der Übertragungsfunktion

Die Regler liefern eine kontinuierliche Spannung, wobei diese zuerst digitalisiert werden muss. Jede Sekunde wird ein Wert digitalisiert. Anschliessend wird der Reglerwert (0-1000) auf  $\pm 20$  dB umgerechnet. Die zehn Reglerwerte werden nun in ein Array gepackt und mit einer Matrix multipliziert um die Übertragungsfunktion zu erhalten. Die Matrix ist so gewählt, dass die Summe jeder Spalte eins gibt. Es wurde auch darauf geachtet, dass keine abrupten Änderungen entstehen. Dies wurde durch Interpolation mit der Sinusfunktion erreicht.

Jetzt haben wir aber erst das einseitige Spektrum. Das Frame wird nun verdoppelt, indem es mit Nullen gefüllt wird. Anschliessend wird es gespiegelt

und mit dem Ungespiegelten addiert. Nun ist die gewünschte Übertragungsfunktion entstanden. Für die Multiplikation muss aber das Spektrum auf ein Frame von 1024 Werten gewandelt werden. Eine lineare Verteilung kann hier nicht vorgenommen werden, weil die zyklische Faltung das Datenframe unbrauchbar macht. Es muss ein Frame entstehen, das 1024 Werte hat, aber einer Übertragungsfunktion von 512 Werten entspricht. Darum wird das Spektrum rücktransformiert, mit Nullen gefüllt und wieder transformiert. Der Wert, der die Gleichspannung repräsentiert, muss separat behandelt werden.

Auf S. 26f sind weitere Erkenntnisse zur Erstellung der Übertragungsfunktion dargestellt.

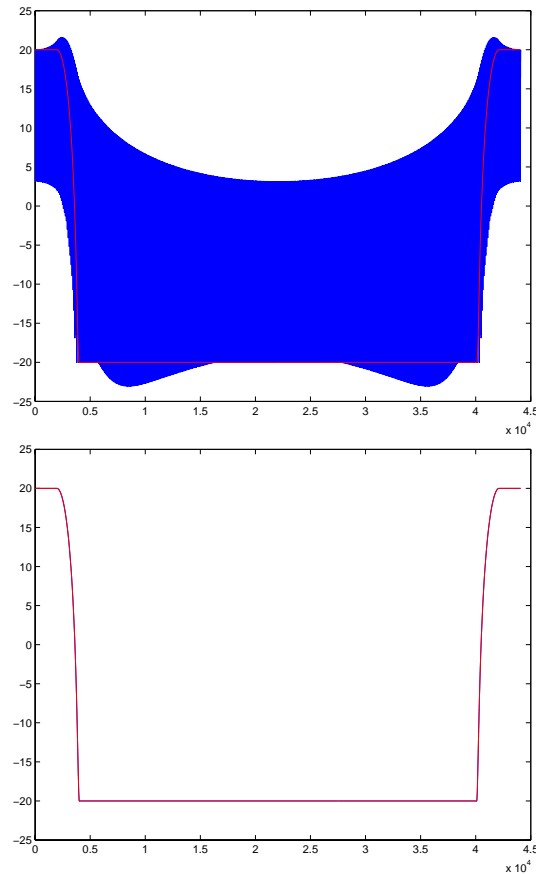


Abbildung 3.5: **Auswirkungen des Zero-Padding.** Oberes Spektrum wurde falsch, unteres richtig mit Nullen aufgefüllt. Weitere Erklärungen auf S. 26f.

## 3.3 Benutzte Komponenten in Simulink

### 3.3.1 From Wave Device

Unter Windows steuert dieser Block direkt die Soundkarte an. Für CD- Qualität wird eine Sample-Rate von 44,1 kHz und eine Signalaufösung von 16 Bit benötigt. *Samples per Frame* gibt an, wie viele Samples in ein Frame gepackt werden. Je grösser dies gewählt wird, desto länger ist die Verzögerung. In unserem Fall macht dies bei einer Sample-Rate von 44,1 kHz und einer Framelänge von 512 etwa 10 ms. Die *Queue duration* ist der Eingangsbuffer. Dieser wird gefüllt, wenn das System überlastet ist.



Abbildung 3.6: *From Wave Device* liest Daten in Echtzeit ab der Soundkarte.

Unter *Audio Device* kann noch die Soundkarte ausgewählt werden, wenn das System mehr als eine besitzt.

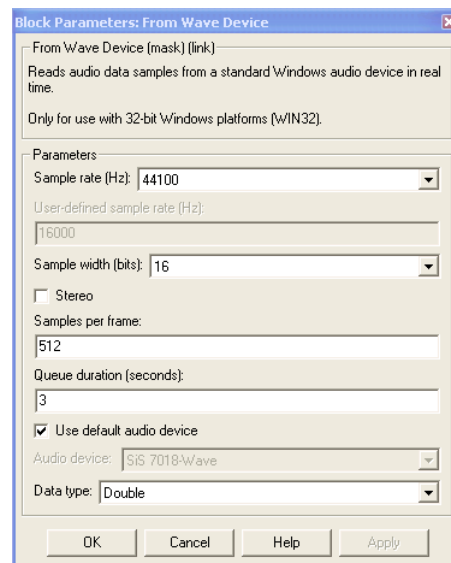


Abbildung 3.7: Parameter-Einstellungen *From Wave Device*.

### 3.3.2 To Wave Device

Dieses Tool ist das Gegenstück zum *From Wave Device*. Die *Queue duration* ist hier der Ausgangsbuffer. *Initial output delay* gibt an, um wieviel Sekunden

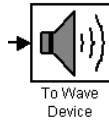


Abbildung 3.8: *To Wave Device* schreibt Daten in Echtzeit in den Ausgangsbuffer der Soundkarte.

der Ausgangsbuffer gefüllt wird, bevor ein Signal ausgegeben wird. Somit wird hier eine Überlast des Computers von einer Sekunde überbrückt, was aber die Verzögerung auf mindestens eine Sekunde erhöht.

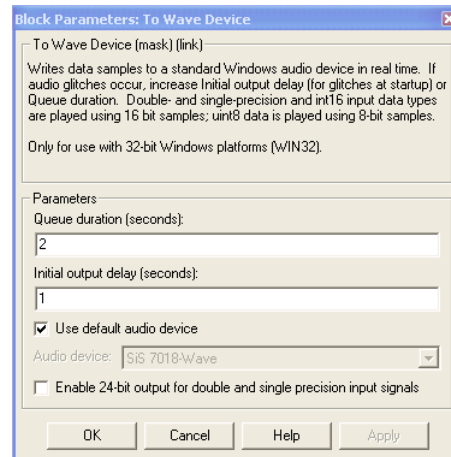


Abbildung 3.9: Parameter-Einstellungen *To Wave Device*.

# Kapitel 4

## Praktische Arbeiten II

### 4.1 Messungen mit der Soundkarte

#### 4.1.1 Messmittelliste

1. Digital Real-Time Oscilloscope  
Tektronix TDS 360, Inv.-Nr. 302.95.007
2. Frequenzgenerator  
Hewlett Packard 33120A
3. Spectrumanalyser  
Hewlett Packard Spectrum Analyzer HP3589A (10 Hz - 150 MHz),  
In.-Nr. 0311.94.002

#### 4.1.2 Systemverzögerung

In der Abbildung 4.1 ist die Systemverzögerung erkennbar:

1. Wie kommt die Systemverzögerung zustande?  
Mit einer Sample-Rate von 44,1 kHz wird alle  $23\mu\text{s}$  ein Wert eingelesen.  
Bis das erste Frame von 1024 Werten komplett ist vergehen etwa 23 ms.
2. Bis das Frame komplett berechnet wurde und am Ausgang zur Verfügung steht vergehen nochmals etwa 120 ms.
3. Mit dem *Initial output delay* im Block *To Wave Device* kann der Ausgangsbuffer der Soundkarte gefüllt werden. Dieser wurde möglichst klein eingestellt, damit für die Messung die Verzögerung möglichst klein ist.

All diese Gründe zusammengefasst ergibt sich eine Systemverzögerung von etwa 160 ms.

Bei der Durchführung der Messungen musste darauf geachtet werden, dass

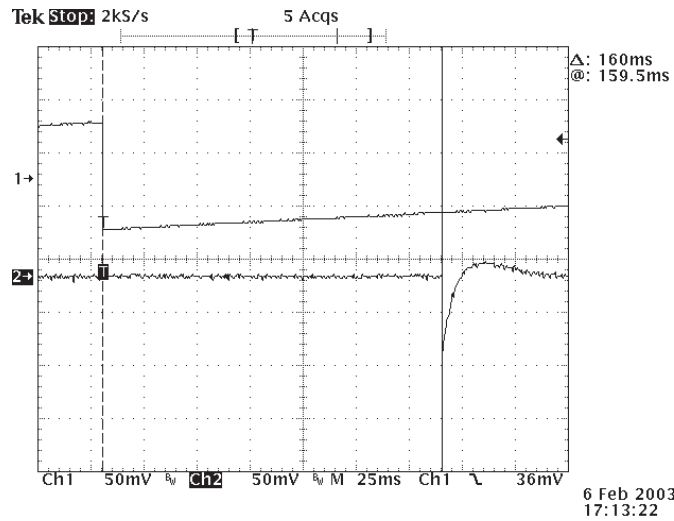


Abbildung 4.1: **Systemverzögerung.** Einspeisung eines Sägezahnsignals aus dem Frequenzgenerator und messen der Systemverzögerung mit dem Oszilloscope.

die Messzeit langsam eingestellt wurde. Der Grund hierfür war, dass die Signalquelle des Spectrum Analyzers und dessen Messeingang synchron sind und unser System (Equalizer, Soundkarte) eine gewisse Verzögerung hat. Es musste also die Messzeit so gross gewählt werden, dass unsere Systemverzögerung nicht ins Gewicht fiel.

## 4.2 Frequenzabtasttechnik

Hier ist die Frage, was zwischen zwei gegebenen Werten mit einem Frequenzabstand von  $\Delta f$  bei der Übertragungsfunktion geschieht. Aufschluss gibt [2] Kap. 6.3.3 auf S. 6-24..28.

Gemäss [2] Formel (6.34) hat ein üblicher *diskreter* Tiefpass folgende Stossantwort:

$$h_0(t) = \Delta f (H_0(0) + 2 \sum_{m=1}^{(M-1)/2} H(m\Delta f) \cos(2\pi m\Delta f t)) \quad , |t| = \tau/2$$

Anhand dieser Formel lässt sich sagen, dass die Stossantwort des Tiefpasses aus einer endlichen Anzahl von Cosinus-Schwingungen im Bereich  $|t| = \tau/2$  besteht.  $|t| = \tau/2$ , weil die Frequenzabtastwerte einen Abstand von  $\Delta f$  haben und die Zeitbegrenzung sich aus der fundamentalen Formel  $\tau = 1/\Delta f$  ergibt.

Mit einigen wenigen Umformungen erhält man dann die Fourierkoeffizienten der Cosinus-Komponenten und den Gleichanteil, welche in [2] auf S. 6-26 Formel (6.35) dargestellt sind. Auch hier wurde mit fundamentalen Ideen über Fourierreihen, Fouriertransformation und Abtastung gearbeitet, wie z.B. der

Zusammenhang *abtasten*  $\circ \text{---} \bullet$  *periodisch fortsetzen*.

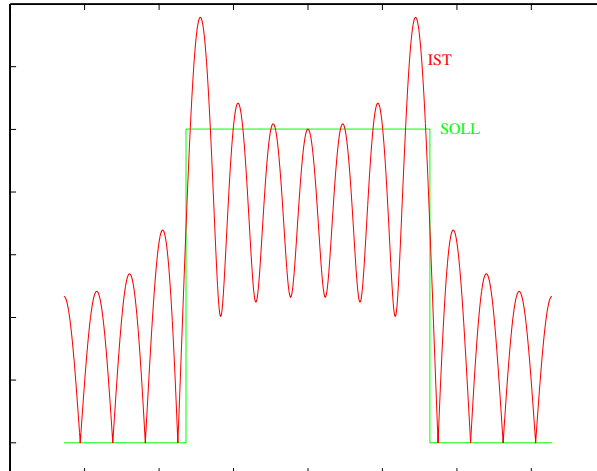


Abbildung 4.2: Frequenzabtasttechnik

### 4.3 Stereo FFT

Hier besteht die Idee darin, die FFT für den linken sowie den rechten Kanal in einem Schritt durchzuführen. Es sind in diesem Zusammenhang einige interessante Details mit dem FFT-Block aus Simulink aufgetaucht. Je nachdem, ob der Eingang beim FFT-Block reell oder komplex ist und ob man den Ausgang in linearer oder *bit-reversed* Form haben will, kommt einer oder mehr von den genannten Algorithmen zum Zug:

1. *Radix-2 decimation-in-time* Algorithmus
2. *Radix-2 decimation-in-frequency* Algorithmus
3. *half-length* Algorithmus
4. *double-signal* Algorithmus

Besonders interessant ist der *half-length* Algorithmus; merkt der FFT-Block, dass der Eingang nur aus reellen Werten besteht, so zerschneidet er die Eingangssequenz in zwei gleich lange Teile  $a$  und  $b$  und bildet dann den FFT von  $c = a + jb$ . Somit führt der FFT-Block eine FFT halber Länge aus und dadurch wird wieder Zeit gespart. Für die Trennung des transformierten Signals  $c$  ist der IFFT-Block gedacht. Es besteht die Möglichkeit, den Eingang als *input is conjugate symmetric* zu definieren, dann führt der IFFT-Block eine *Radix-2 decimation-in-time* in Verbindung mit dem *half-length* Algorithmus

aus.

Ausführliche Herleitung zum *half-length* Algorithmus sind in [3] S. 16ff ersichtlich.

## 4.4 Loudness

Die *Loudness*-Taste wird bei manchen Equalizern eingebaut und dient zur gehörrichtigen Lautstärkeentzerrung. Dieses Filter passt sich bei zunehmend aufgedrehten Lautstärkeregler dem flacher werdenden Verlauf der Gehörkurve an und hebt hohe bzw. tiefe Frequenzen vom Pegel her an (oder senkt die Mitten-Frequenzen entsprechend ab). Die Loudness-Taste ist demnach kein Bassverstärker und sollte nur bei geringer Abhörlautstärke aktiviert werden. Wir haben die Loudness-Taste nicht implementiert, weil das in dieser kurzen Zeit vom Januar bis Februar zuviel Aufwand bedeutet hätte. Eine mögliche Implementation könnte darin bestehen, dass man einige Gehörkurven bei bestimmten Maskierungen aus Tabellen herauslesen würde. Man muss sie herauslesen, weil die Gehörkurven schon früher von anderen Leuten empirisch bestimmt worden sind, da nicht jedes Ohr gleich hört. Diese Implementierung weiter zu verfolgen würde für diese Studienarbeit zuviel Aufwand für so eine kleine Funktion bedeuten.

# Kapitel 5

## Schlusswort

Diese Studienarbeit vermittelte ein umfassendes Verständnis der FFT/IFFT, diskrete lineare und zyklische Faltung, Leakage und Frequenzabtasttechnik. Letzteres bezieht sich vor allem auf das Erstellen der Übertragungsfunktion. Der Umgang mit Matlab und Simulink wurde gefestigt und die Möglichkeiten der beiden Programme, allen voran Simulink, ausprobiert.

Unser Equalizer basiert auf der FFT und IFFT, weil diese Transformationen in Simulink *intelligent* programmiert sind und sich dadurch interessante Möglichkeiten ergeben. Ein anderer Weg um einen Equalizer zu entwerfen wäre sicher derjenige über *herkömmliche* Filter (Bandpässe) gewesen, so wie Equalizer normalerweise auch gebaut sind. Am Anfang dieser Studienarbeit stellte sich schnell heraus, dass mit solch einer Implementation (Bandpässe) die Anforderung an die Rechenleistung des Computers stark zunahm. Ein Grund dafür waren die Filterlängen in Simulink, die eine Länge von etwa 30 haben mussten um einigermaßen *anständige* Filterflanken zu erhalten. Somit entschieden wir uns für ein Design mittels FFT und IFFT.

Ein weiteres Problem war das Ausdrucken der gespeicherten Frequenzgänge, die der HP3589A Spectrum Analyzer aufgenommen hatte. Das Datei-Format der gespeicherten Daten war uns nicht bekannt und es bestand die Möglichkeit, die Frequenzgänge mittels einer Digitalkamera aufzunehmen und dann weiter zu verwenden. Ein grosses Dankeschön geht an dieser Stelle an den Laborassistenten Peter Roffler, der sich um das Ausdruck-Problem gekümmert hat und es geschafft hat, vom Spectrum Analyzer Bilder auf einen Datenträger zu bekommen, die wir dann weiter verwenden konnten.

Ein weiterer Dank an dieser Stelle geht an unseren Betreuer, Prof. Dr. G. Schuster, für die vielen Tipps & Tricks und die gute Zusammenarbeit.

# Literaturverzeichnis

- [1] Vijay K. Madisetti, Douglas B. Williams: The digital signal processing handbook. CRC Press, 1998.
- [2] A. Schüeli: Digitale Signalverarbeitung, Skript zum Vertiefungsfach Digitale Signalverarbeitung an der Abteilung Elektrotechnik der HSR.
- [3] Winthrop W. Smith, Joanne M. Smith: Handbook of Real-Time Fast Fourier Transforms. Algorithms to Product Testing. IEEE Press, 1995.
- [4] Emmanuel C. Ifeachor, Barrie W. Jervis: Digital Signal Processing. A Practical Approach. Addison-Wesley Publishing Company, 1993.

# Anhang A

## Messungen

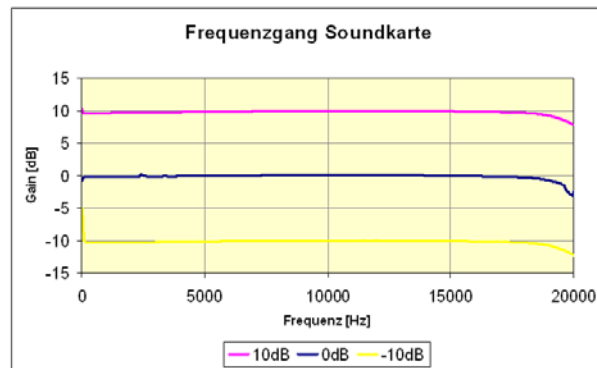


Abbildung A.1: Frequenzgang der Soundkarte

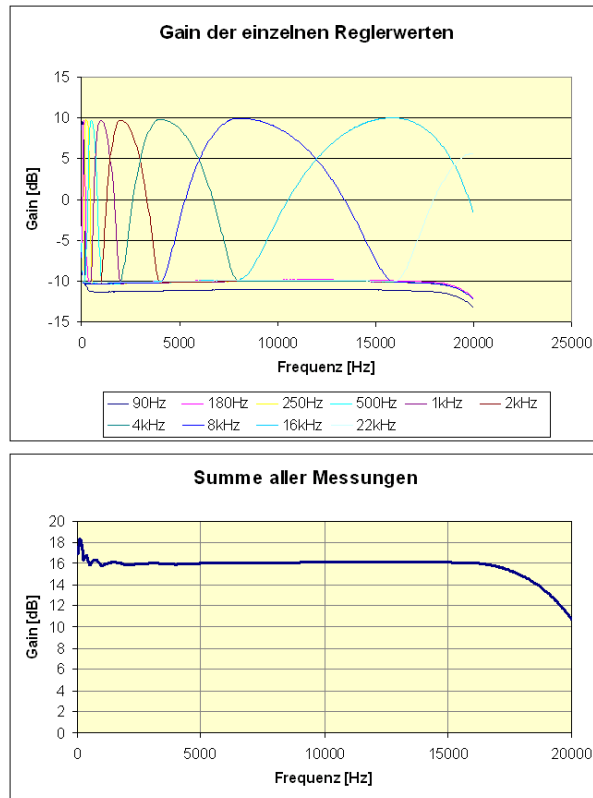


Abbildung A.2: **Summe der Messungen**. Da die Messwerte in dB gemessen wurden, kann man nicht einfach eine Addition durchführen. Die Berechnung muss im linearen Bereich durchgeführt werden. Somit ergibt sich ein Wert von 15,5 dB.

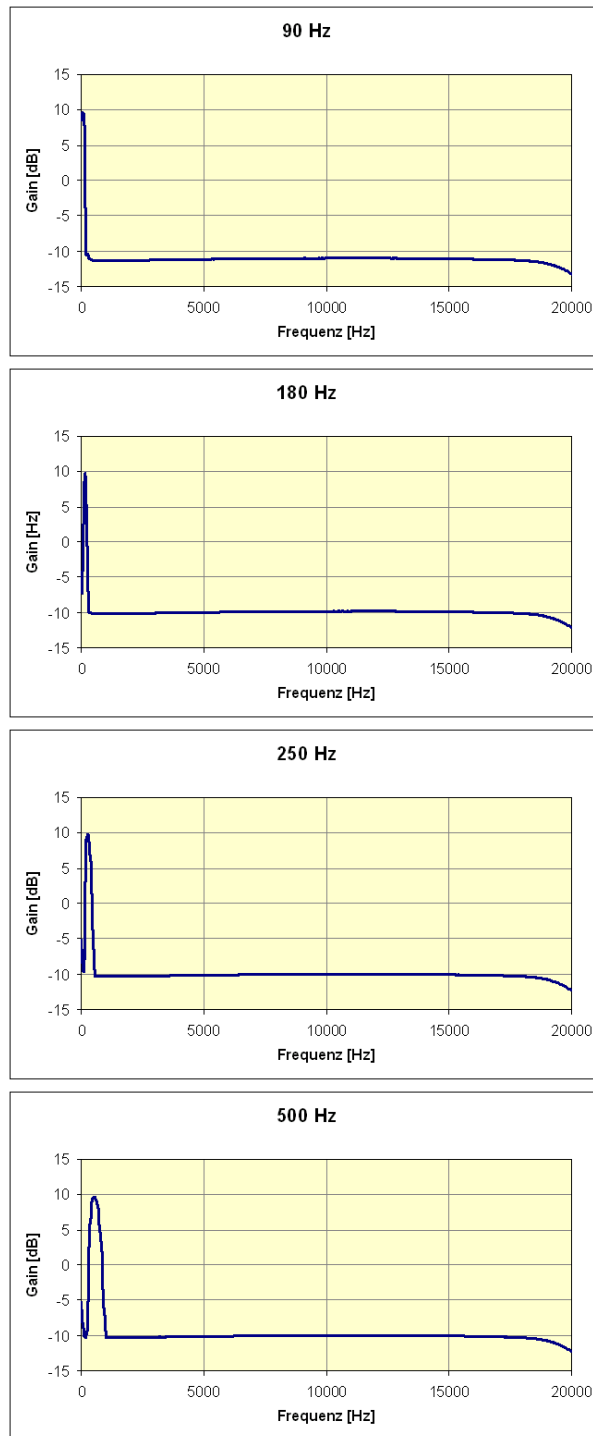


Abbildung A.3: Frequenzgänge 90 Hz bis 500 Hz

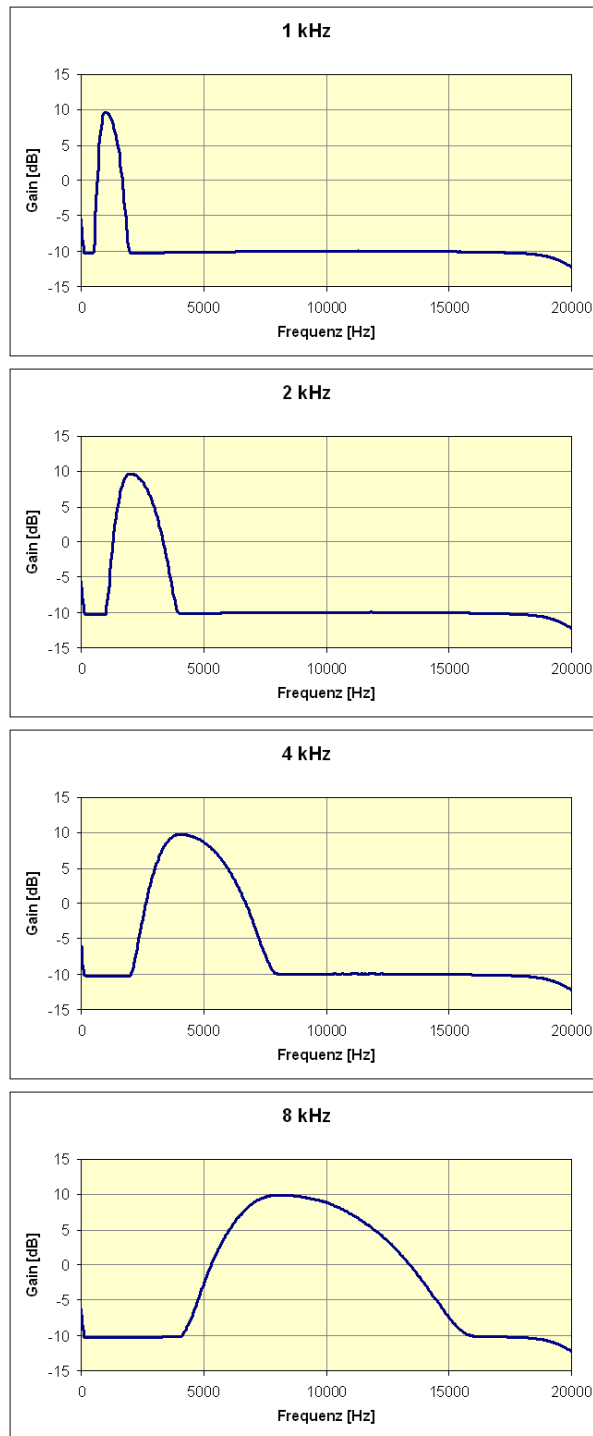


Abbildung A.4: Frequenzgänge 1 kHz bis 8 kHz

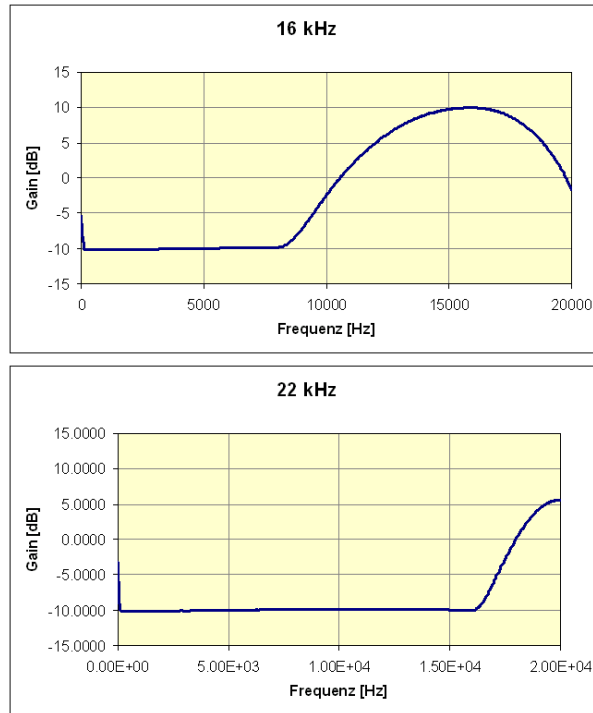


Abbildung A.5: Frequenzgänge 16 kHz und 22 kHz

# Anhang B

## Übertragungsfunktion

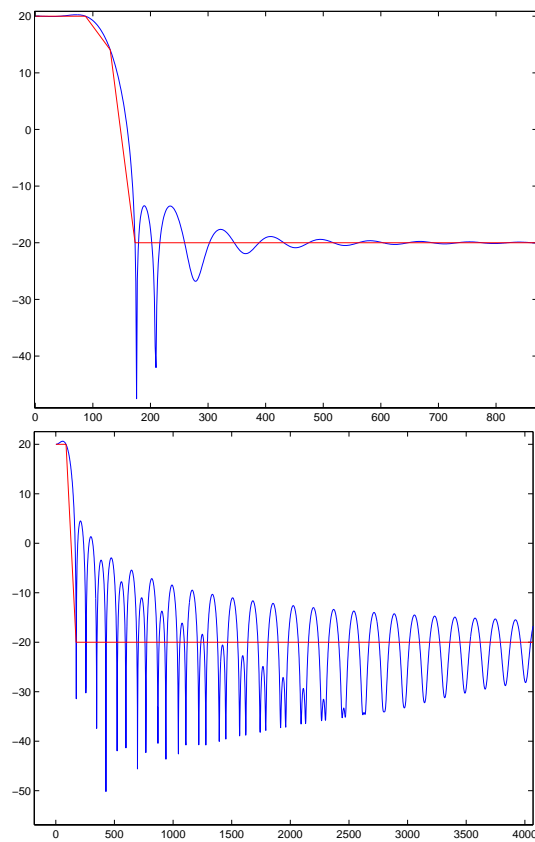


Abbildung B.1: **Herstellung des Übertragungsspektrums.** Oberes Spektrum ist mit 1024 Werte pro Frame, unteres mit 512 Werte pro Frame hergestellt worden. Beim oberen Spektrum sind doppelt so viele Frequenzstützpunkte vorhanden wie beim unteren Spektrum. Somit ist es möglich, einen Zwischenwert einzufügen, welcher ein *freundlicheres* Spektrum ergibt.

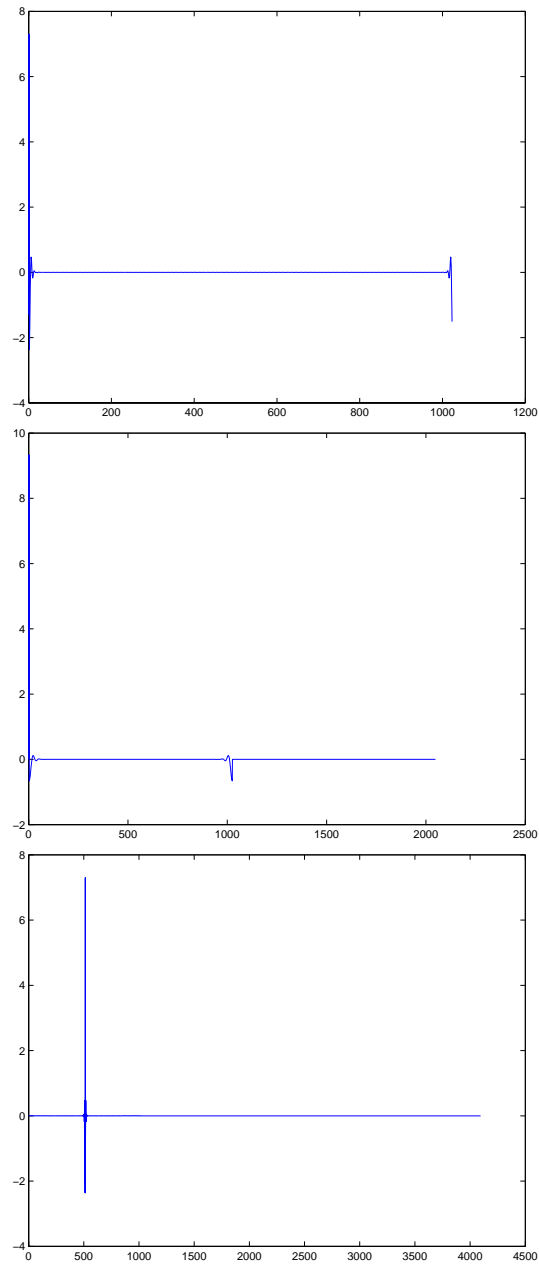


Abbildung B.2: **Impulsantwort der Übertragungsfunktion.** Wenn die Impulsantwort der Übertragungsfunktion (oberstes Bild) am Ende mit Nullen aufgefüllt wird (mittleres Bild), entsteht ein *wildes* Spektrum (Abbildung 3.5 oberes Spektrum). Richtigerweise muss die Impulsantwort in der Mitte mit Nullen aufgefüllt werden, damit sie nicht unterbrochen wird. Um weiter das Gesetz der Overlap-Add/Save Methode zu erfüllen, darf die Impulsantwort nicht länger als  $L$  Werte sein. Dies wird erreicht, indem der rechte Teil nach links an den Anfang geschoben und anschliessend das Frame mit Nullen aufgefüllt wird (unterstes Bild).