Technische Universität Braunschweig

Institut für Nachrichtentechnik

Praktikum für Nachrichtentechnik

Versuch 7: Digitale Filter

Betreuer: M.Sc. Marc-André Jung

Stand: 31. Oktober 2014

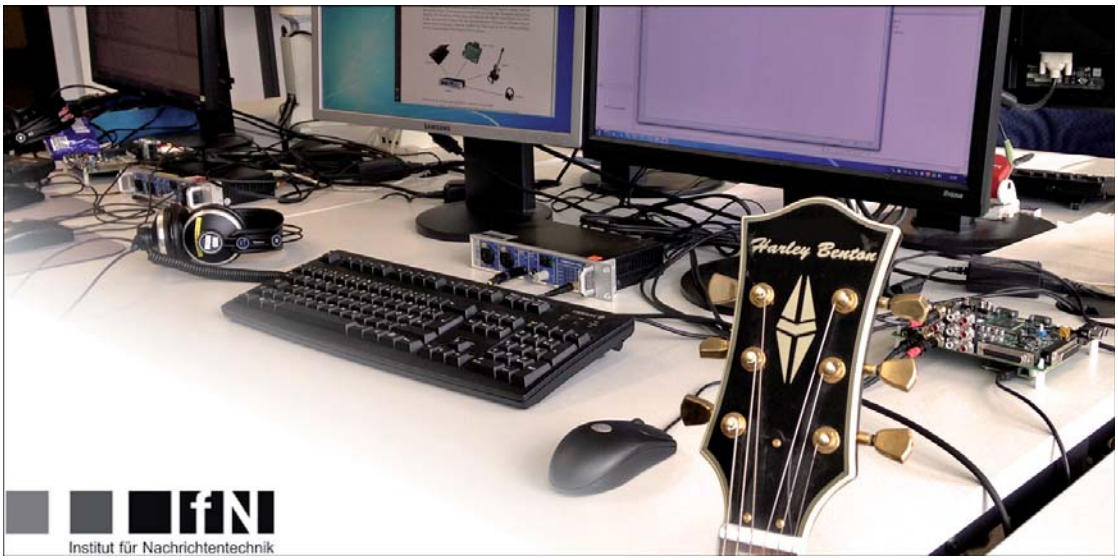
Skript erarbeitet von: Jung, Weiß, Franzen

Inhaltsverzeichnis

1 Einleitung

Die digitale Signalverarbeitung gewinnt im Zuge der fortschreitenden Digitalisierung stets an Bedeutung. Im Bereich von Audio und Sprache sind mit der „digitalen Revolution“ die Möglichkeiten von Anwendungen scheinbar unbegrenzt. Seien es klangliche Modifizierungen des Infotainmentsystems im Automobil, Effektgeräte für Musiker oder Sprachverarbeitung in der Telefonie, ohne digitale Filter wäre all das nicht mehr denkbar.

In diesem Praktikum werden drei anschauliche Versuche im Bereich der digitalen Filter durchgeführt. Dazu wird Ihr bereits erworbenes, aber auch neues Wissen über die theoretischen Grundlagen genutzt, um einen Einblick in die praktische Umsetzung zu erhalten. In den Versuchen selbst wird mit Matlab und einem digitalen Signalprozessor von Analog Devices gearbeitet. Zum Experimentieren und Auswerten der Ergebnisse wird außerdem das Programm Adobe Audition genutzt.



Dieses Skript ist in drei Teile geteilt. Den ersten Teil halten Sie bereits in den Händen. Er umfasst die theoretischen Grundlagen und Hintergründe, die für das Praktikum benötigt werden. Dieser Teil soll von ihnen vorbereitend zuhause erarbeitet werden. Zu Beginn des Praktikumstermins wird es ein kleines Kolloqium mit Verständnisfragen hierzu geben. Teil II und III bauen darauf auf. Sie beinhalten die Versuchsbeschreibungen und den dazugehörigen Versuchsaufbau. Diese beiden Teile werden Ihnen am Anfang des Praktikumstermins von ihrem Betreuer ausgehändigt und werden während des Termins bearbeitet.

Als weiterführende Literatur seien folgende Skripte bzw. Bücher empfohlen:

- T. Fingscheidt: Skript zur Vorlesung Digitale Signalverarbeitung. Im folgenden mit [Fi-DSV] referenziert.
- K.D. Kammeyer, K. Kroschel: Digitale Signalverarbeitung, Teubner-Verlag, 2002.
- A.V. Oppenheim, R.W. Schafer, J.R. Buck: Zeitdiskrete Signalverarbeitung, Pearson Studium, 2004.
- H.-W. Schüßler: Digitale Signalverarbeitung 1, Springer-Verlag, 1994.

2 Signale und Systeme

Für das Praktikum benötigen wir natürlich auch etwas Theorie. In diesem Kapitel werden die wichtigsten Grundlagen der Systemtheorie wiederholt und aufgefrischt.

2.1 Digitale Signale

Signale sind informationstragende Zeichen. Liegt ein analoges Signal vor, so liefert die zeitliche Abtastung mit anschliessender Quantisierung ein digitales Signal. Bei der weiteren Verarbeitung des digitalen Signals sind der Kreativität fast keine Grenzen gesetzt. Je nach Anwendung können, siehe Abbildung 1, auf verschiedenen Plattformen, wie z.B. dem PC, einem digitalen Signalprozessor oder einem Mikrocontroller, die gewünschten Verarbeitungsschritte realisiert werden. Mit den nächsten Kapiteln geht es nun etwas tiefer in die Theorie.

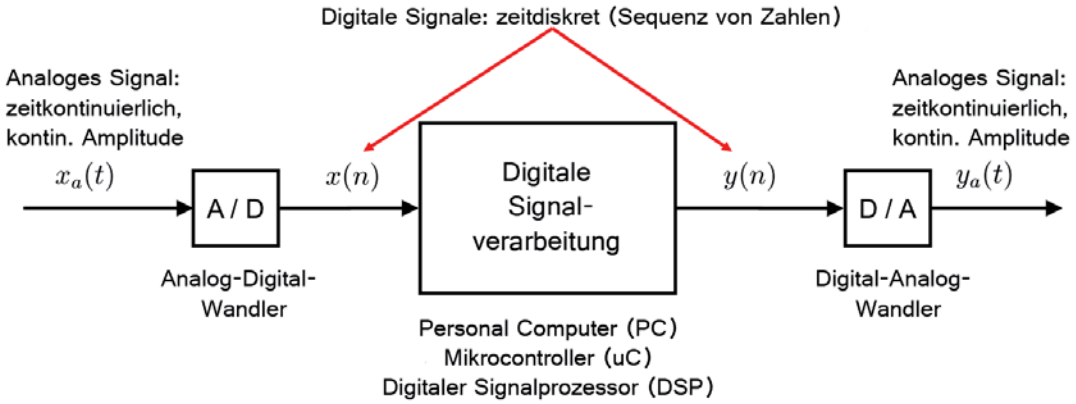


Abbildung 1: Zusammenhang analoge/digitale Signale und die DSV. Nach [Fi-DSV].

2.1.1 Abtastung

Um aus einem analogen Signal ein digitales Signal zu erhalten, wird das analoge Signal abgetastet. Dazu werden dem analogen Signal mit der Abtastperiode T Werteproben (*Samples*) entnommen. Die Abtastperiode T muss dabei sorgfältig gewählt sein. Um das analoge Signal fehlerfrei rekonstruieren zu können, muss nach dem **Abtasttheorem nach Shannon** folgendes gelten:

$$f_s \geq 2 \cdot f_g \quad \text{mit Abtastfrequenz } f_s = \frac{1}{T} \text{ und Grenzfrequenz } f_g \; . \tag{1}$$

Das heißt die Abtastfrequenz f_s des digitalen Signales muss mindestens zwei mal so groß sein, wie die höchste im Signal vorkommende Frequenz f_g (Grenzfrequenz). Die Frequenz $f_n = 2 \cdot f_g$ wird häufig auch als Nyquist-Frequenz bezeichnet. Sollte $f_s < f_n$ sein, ergibt sich ein verfälschtes abgetastetes Signal, aus dem das Original nicht mehr fehlerfrei rekonstruierbar ist.

Um die Abtastung mathematisch darzustellen benötigen wir zunächst den zeitkontinuierlichen Dirac-Stoß, der als

$$\delta_a(t) = \begin{cases} \infty & , t = 0 \\ 0 & , t \neq 0 \end{cases} \tag{2}$$

definiert ist. Der Dirac-Stoß besitzt die sogenannte Sieb- oder auch Ausblendeigenschaft, die den Wert des Signals $x(t)$ zum Zeitpunkt t_0 „aussiebt“, d.h. alle verbleibenden Werte nicht mehr berücksichtigt:

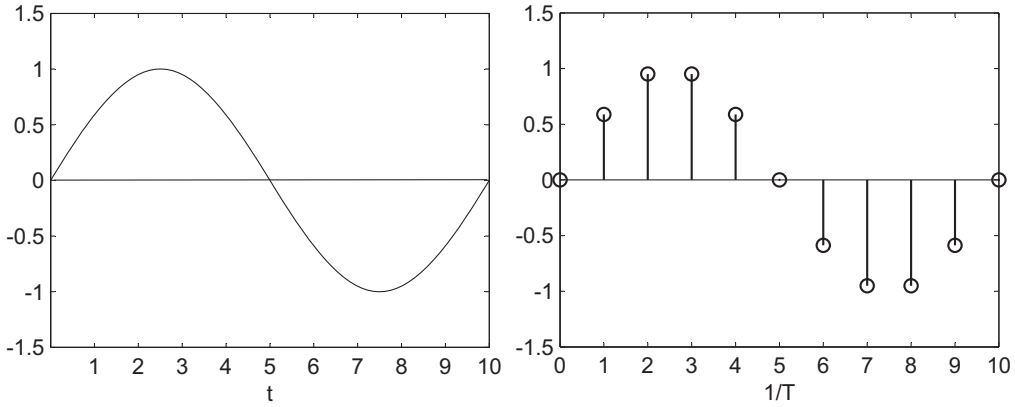
$$\int_{-\infty}^{\infty} x(t) \cdot \delta_a(t - t_0)\,\mathrm{d}t = x(t_0)\ . \tag{3}$$

Setzen wir in diese Formel für t_0 nun ein Vielfaches unserer Abtastperiode T ein, so erhalten wir den n-ten Abtastwert aus dem analogen Signal x_a über die Ausblendeigenschaft durch

$$x(n) = \int_{-\infty}^{\infty} x_a(t) \cdot \delta_a(t - nT)\,\mathrm{d}t = x_a(nT)\ , \tag{4}$$

wie in Abb. 2 beispielhaft gezeigt ist. Da wir im folgenden immer zeitdiskrete Signale betrachten, sei noch einmal betont, dass der Index n in ein Signal der ganzzahlige Werteindex in die Folge der Abtastwerte ist. Das heißt kurz:

$x(n)$ bezeichnet den n-ten Abtastwert von $x_a(t)$.



(a) kontinuerlich (t in s) (b) diskret (T = 1s)

Abbildung 2: Kontinuierliches und entsprechendes abgetastetes Signal.

Als zusätzliche Information sei noch gesagt, dass wir für Rechnungen im zeitdiskreten Bereich den Dirac-Stoß definieren als

$$\delta(n) = \begin{cases} 1 & , n = 0 \\ 0 & , n \neq 0 \end{cases}\ . \tag{5}$$

2.1.2 Quantisierung

Auch bei der Amplitude ist es so, dass die Abtastwerte (Samples) nicht jeden kontinuierlichen Wert annehmen können, sondern hier systembedingt auf feste, binär codierte Werte gerundet werden. Dies nennt man Quantisieren. Die Anzahl der Quantisierungsstufen ist davon abhängig, welche Auflösung man für die Darstellung eines Samples wählt. Ein mit 8 Bit quantisiertes Sample kann z.B. $2^8 = 256$ verschiedene Amplitudenwerte annehmen, ein mit 16 Bit quantisiertes Sample bereits $2^{16} = 65536$. Je größer die Anzahl der Quantisierungsstufen ist, umso geringer wird natürlich auch der Fehler zum ursprünglichen analogen Signal.

Um quantisierte und nicht-quantisierte Signale auseinanderhalten zu können, kennzeichnen wir quantisierte Signale mit dem zusätzlichen Index q. Also:

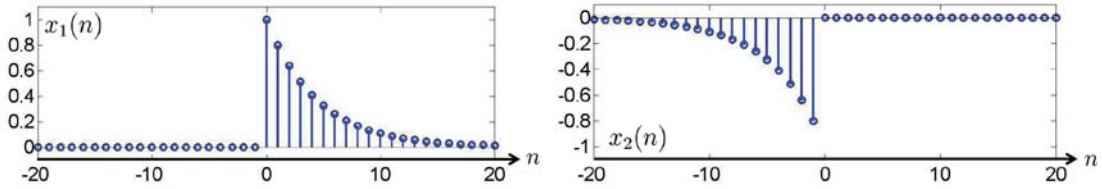
$$x_q(n) \text{ bezeichnet den n-ten quantisierten Abtastwert von } x_a(t).$$

2.1.3 Sequenzen

Ein zeitdiskretes Signal, also bei uns $x(n)$, wird häufig auch als Sequenz bezeichnet. Dabei nennt man eine Sequenz

$$\begin{aligned} \text{rechtsseitig:} \quad & x(n) = 0 \quad \text{für alle} \quad n < n_{\text{start}} \quad mit \quad n_{\text{start}} > -\infty \\ \text{linksseitig:} \quad & x(n) = 0 \quad \text{für alle} \quad n > n_{\text{end}} \quad mit \quad n_{\text{end}} < \infty \ . \end{aligned} \tag{6}$$

Eine Sequenz ist beidseitig, wenn sie weder links- noch rechtsseitig ist. In Abbildung 3 sind zwei Beispielsequenzen dargestellt.



(a) Rechtsseitige Beispielsequenz (b) Linksseitige Beispielsequenz

Abbildung 3: Beispiel: rechts- und linksseitige Exponentialsequenz.

2.2 LSI-Systeme

Ein System ist ein Operator oder eine (mehr oder weniger komplexe) Transformation, die einer Eingangssequenz eine Ausgangssequenz zuordnet. Die Abstrahierung in Systeme kann Berechnungen wesentlich vereinfachen und auch übersichtlicher gestalten. Es folgen einige grundlegende Dinge zu Systemen.

2.2.1 Grundlagen der Systemtheorie

Ein System ist definiert durch eine eindeutige Abbildung $\mathfrak{Tr}\{\}$ eines Eingangssignales $x(n)$ auf ein Ausganggsignal $y(n)$

$$y(n) = \mathfrak{Tr}\{x(n)\} \ . \tag{7}$$

Das System ist **linear** wenn

$$y(n) = \mathfrak{Tr}\left\{a \cdot x_1(n) + b \cdot x_2(n)\right\} = a \cdot \mathfrak{Tr}\{x_1(n)\} + b \cdot \mathfrak{Tr}\{x_2(n)\} \tag{8}$$

gilt, also das Prinzip der Superposition bzw. Überlagerung anwendbar ist. Ein System ist zudem **zeitinvariant**, wenn zeitliches Verschieben des Einganggsinals um t_0 eine auch um t_0 verschobene, aber anderweitig identische Antwort hervorruft. Im zeitdiskreten Bereich wird dieselbe Eigenschaft, dann mit Verschiebung n_0, **shift invariant** genannt:

$$y(n - n_0) = \mathfrak{Tr}\{x(n - n_0)\} \ . \tag{9}$$

Sind beide Eigenschaften erfüllt, haben wir es mit einem sogenannten LSI-System zu tun, einem „linear shift invariant system“.

Doch wie erhält man die Ausgangssequenz eines Systems, also $\mathfrak{Tr}\{x(n)\}$, auf eine beliebige Eingangssequenz $x(n)$?

$$\rightarrow \text{Durch Faltung mit der Impulsantwort des Systems!}$$

Dazu im nächsten Abschnitt mehr.

2.2.2 Eigenschaften und Beschreibung von LSI-Systemen

Die Antwort eines LSI-Systems auf den Dirac-Stoß $\delta(n)$ als Eingangssequenz heißt **Impulsantwort** $h(n)$.

$$h(n) = \mathfrak{Tr}\{\delta(n)\} \tag{10}$$

Die Impulsantwort charakterisiert ein System vollständig und erlaubt somit die Berechnung der Antwort auf eine beliebige Eingangssequenz, sofern sich das System zu Beginn im Ruhezustand befindet. Ist $h(n)$ bekannt, so berechnet sich die Antwort $y(n)$ des Systems auf die Eingangssequenz $x(n)$ zu

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k) \cdot h(k) = x(n) \star h(n) \tag{11}$$

Die Operation $\star$ nennt sich Faltung. Allgemein für zwei Signale $x_1(n)$ und $x_2(n)$:

$$y(n) = x_1(n) \star x_2(n) = \sum_{k=-\infty}^{\infty} x_1(n-k) \cdot x_2(k) = \sum_{k=-\infty}^{\infty} x_1(k) \cdot x_2(n-k) \tag{12}$$

Mit der Impulsantwort lassen sich einige wichtige Eigenschaften des Systems bestimmen: Ein System ist genau dann **stabil**, wenn gilt, dass

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty \text{ ist.} \tag{13}$$

Ein System ist genau dann **kausal**, wenn

$$h(n) = 0 \quad \text{für} \quad n < 0 \ . \tag{14}$$

Für den Entwurf von Filtern spielt die Unterscheidung von Systemen nach der FIR- oder IIR-Eigenschaft eine entscheidende Rolle. Ein System ist ein **FIR (finite impulse response)** System, genau dann wenn gilt

$$h(n) \neq 0 \quad \text{für nur endlich viele } n. \tag{15}$$

Ist dies nicht erfüllt, so ist es ein **IIR (infinite impulse response)** System.

2.3 Differenzengleichungen und Blockschaltbilder

Eine weitere Möglichkeit die Ausgangswerte eines LSI-Systems zu erhalten ist, sie als gewichtete Summe der letzten Eingangs- und der letzten Ausgangswerte aufzufassen. Die Gleichung, die dies beschreibt, heißt **Differenzengleichung**:

$$0 = \sum_{\nu=0}^{N_a} a_\nu \cdot y(n-\nu) + \sum_{\mu=0}^{N_b} b_\mu \cdot x(n-\mu) \ , \text{ mit} \tag{16}$$

N_a vorigen Ausgangswerten, N_b vorigen Eingangswerten und a_ν, b_μ als Gewichtungsfaktoren. Setzen wir (normierungsbedingt) $a_0 = -1$, so erhalten wir

$$y(n) = \sum_{\nu=1}^{N_a} a_\nu \cdot y(n-\nu) + \sum_{\mu=0}^{N_b} b_\mu \cdot x(n-\mu) \ , \tag{17}$$

das heißt $y(n)$ ergibt sich aus N_a vorigen Ausgangswerten und $N_b + 1$ Eingangswerten.

Das System ist durch die Differenzengleichung und den Anfangszustand des Systems vollständig beschrieben. Die Anfangswerte sind notwendig, damit für die ersten $max(N_a, N_b)$ Ausgangswerte die Werte $x(n-\mu)$ und $y(n-\nu)$ definiert sind.

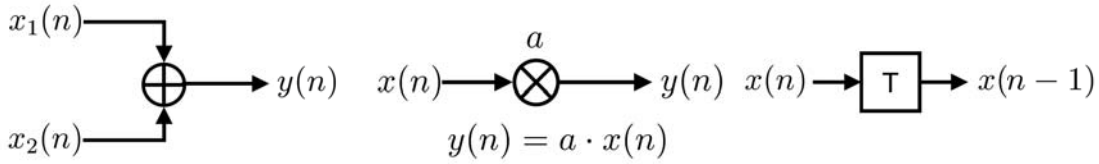
Wenn $N_a \neq 0$ ist, vorige Ausgangswerte also in die Berechnung des aktuellen Ausgangswerts eingehen, so ist das System **rekursiv**. Gilt hingegen $N_a = 0$, so ist das System nicht-rekursiv. Nicht-rekursive Systeme sind immer FIR-Systeme mit:

$$y(n) = \sum_{\mu=0}^{N_b} \frac{b_\mu}{-a_0} \cdot x(n-\mu) \tag{18}$$

Ein positiver Nebeneffekt ist, dass man aus Gleichung (18) direkt die Impulsantwort eines FIR-Systems ablesen kann:

$$h(n) = \begin{cases} \frac{b_n}{-a_0}, & n = 0, 1, 2, ..., N_b \\ 0, & \text{sonst} \end{cases} . \tag{19}$$

Differenzengleichungen haben noch einen weiteren Vorteil. Aus ihnen kann man sehr leicht anschauliche **Blockschaltbilder** gewinnen. Dies sind die graphischen Darstellungen der Differenzengleichungen. In ihnen werden die Pfade, die ein Signal im System durchläuft, deutlich. Dabei nutzen wir die in Abbildung 4 gezeigten Grundbausteine:



(a) Addition (b) Multiplikation mit a (c) Verzögerung um einen Abtastwert

Abbildung 4: Grundelemente eines Blockschaltbildes. Nach [Fi-DSV].

Das Eingangssignal (meist $x(n)$) erkennt man an dem eingehenden Pfeil. Am Ende des Blockschaltbildes fließen alle Pfeile zum Ausgangssignal $y(n)$ zusammen. Das Additionsglied (4a) akzeptiert zwei oder mehr Signale und leitet das Summensignal weiter. Die Multiplikation mit einem Faktor (4b) wird angezeigt durch das Multiplikationsglied mit dem Faktor daneben. Das Verzögerungsglied T (4c) verzögert ein Signal um einen Abtastwert (T steht für eine Abtastperiode). Führt man das Signal $x(n)$ durch k hintereinandergeschaltete Verzögerungsglieder, erhält man $x(n-k)$.

Ganz allgemein sieht unsere Differenzengleichung dann also aus wie in Abbildung 5 dargestellt. Für ein nicht-rekursives System vereinfacht es sich zu Abbildung 6.

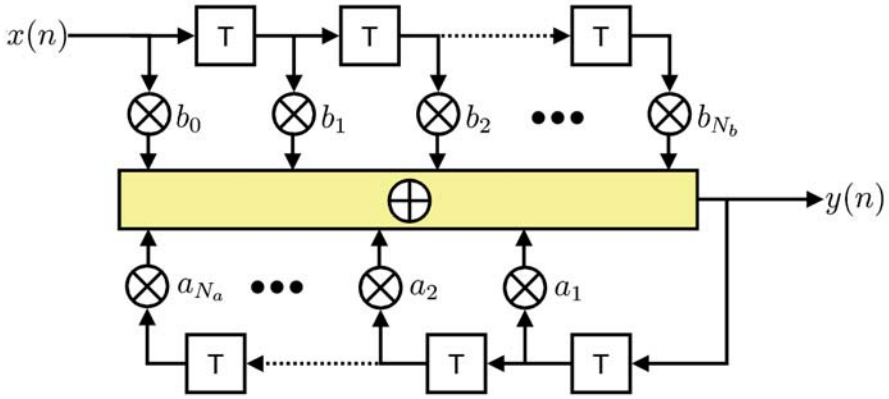


Abbildung 5: Blockschaltbild eines rekursiven Systems

$$y(n) = \sum_{\nu=1}^{N_a} a_\nu \cdot y(n-\nu) + \sum_{\mu=0}^{N_b} b_\mu \cdot x(n-\mu) \ .$$

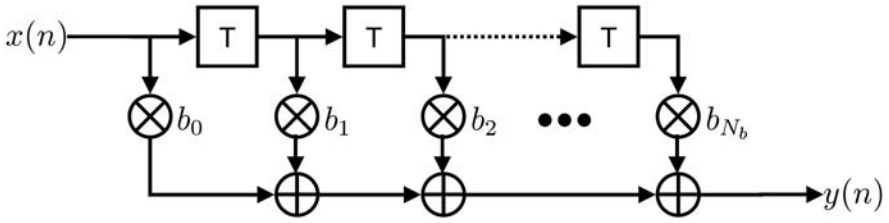


Abbildung 6: Blockschaltbild eines nicht-rekursiven Systems

$$y(n) = \sum_{\mu=0}^{N_b} b_\mu \cdot x(n-\mu) \ .$$

3 Systemanalyse im Frequenzbereich

Im folgenden wollen wir einen Überblick darüber gewinnen, welche Mittel uns zur Analyse von Systemen im Frequenzbereich zur Verfügung stehen.

3.1 Fourier-Transformation

Die Fourier-Transformation ist das wichtigste Werkzeug für die Analyse von Systemen im Frequenzbereich. Bei zeitkontinuierlichen Signalen liefert sie uns zunächst eine Analyse der harmonischen Komponenten des Signals. Das bedeutet sie „zerlegt“ ein Signal in seine harmonischen Komponenten und stellt dieses resultierende kontinuierliche Spektrum im Frequenzbereich dar. Sie ist definiert durch

$$\mathfrak{F}\{x_a(t)\} = \int_{-\infty}^{\infty} x_a(t) e^{-j\omega t}\,\mathrm{d}t\ . \tag{20}$$

3.1.1 Eigenschaften und Frequenzgänge

Der Vollständigkeit halber sind die Eigenschaften bzw. Rechenregeln der Fourier-Transformation in der folgenden Tabelle aufgeführt und denen des Zeitbereichs gegenübergestellt.

even odd

$$x_a(t) = x_a^{(e)}(t) + x_a^{(o)}(t) \quad \circ\!\!-\!\!\bullet \quad \mathfrak{F}\{x_a(t)\} = X_a(j\omega) = \mathrm{Re}\{X_a(j\omega)\} + j\mathrm{Im}\{X_a(j\omega)\}$$

Property	time domain $\circ\!\!-\!\!\bullet$ frequency domain	
Transform	$x_a(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X_a(j\omega)e^{j\omega t}d\omega$	$X_a(j\omega) = \int_{-\infty}^{+\infty} x_a(t)e^{-j\omega t}dt$
Even part	$x_a^{(e)}(t) = \frac{1}{2}x_a(t) + \frac{1}{2}x_a^*(-t)$	$\mathrm{Re}\{X_a(j\omega)\}$
Odd part	$x_a^{(o)}(t) = \frac{1}{2}x_a(t) - \frac{1}{2}x_a^*(-t)$	$j\mathrm{Im}\{X_a(j\omega)\}$
Conjugation	$x_a^*(t)$	$X_a^*(-j\omega)$
Real-valued $x_a(t)$	$x_a(t) = x_a^*(t)$	$X_a(j\omega) = X_a^*(-j\omega)$
Time Shift	$x_a(t - t_0)$	$X_a(j\omega)\cdot e^{-j\omega t_0}$ $\quad t_0 \in \mathbb{R}$
Frequency Shift	$x_a(t)\cdot e^{j\omega_0 t}$	$X_a(j(\omega-\omega_0))$ $\quad \omega_0 = 2\pi f_0$
Time & Freq. Mirroring	$x_a(-t)$	$X_a(-j\omega)$
Time & Freq. Scaling	$x_a(c\cdot t)$	$\frac{1}{\lvert c\rvert}\cdot X_a(j\frac{\omega}{c})$ $\quad c\in\mathbb{R}, c\neq 0$
Superposition	$c_1\cdot x_a(t) + c_2\cdot y_a(t)$	$c_1\cdot X_a(j\omega) + c_2\cdot Y_a(j\omega)$
Convolution	$x_a(t) * y_a(t)$	$X_a(j\omega)\cdot Y_a(j\omega)$
Modulation	$x_a(t)\cdot y_a(t)$	$\frac{1}{2\pi}\cdot X_a(j\omega) * Y_a(j\omega)$
Parseval's Theorem	$\int_{-\infty}^{+\infty} \lvert x_a(t)\rvert^2 dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \lvert X_a(j\omega)\rvert^2 d\omega$	

Abbildung 7: Tabelle der Eigenschaften der Fourier-Transformation. Aus [Fi-DSV].

Wofür können wir die Fourier-Transformation nun konkret nutzen? Zum Beispiel um den Frequenzgang eines Systems zu berechnen. Er beschreibt das frequenzabhängige Verhalten eines Systems (z.B. eines Filters). Das heißt aus ihm kann man unter anderem ablesen, wie stark eine Frequenz des Eingangssignal durch das System gedämpft wird und folglich im Ausgangssignal auftritt. Mathematisch gesehen ist der Frequenzgang $H(j\omega)$ die Fourier-Transformierte der Impulsantwort $h(n)$ und lässt sich wie folgt ermitteln:

$$y(n) = x(n) \star h(n) \circ\!\!-\!\!\bullet\, Y(j\omega) = X(j\omega) \cdot H(j\omega) \Longleftrightarrow H(j\omega) = \frac{Y(j\omega)}{X(j\omega)} \ . \tag{21}$$

Der Frequenzgang gibt für jede Frequenz $j\omega$ das Verhalten des Filters an. Die Darstellung in Polarkoordinaten ist

$$H(j\omega) = |H(j\omega)| \cdot e^{j\varphi(j\omega)} \ , \tag{22}$$

wobei $|H(j\omega)|$, der **Amplitudengang**, ein zentrales Element der Beschreibung von Filtern ist! Die Phase des Systems wird durch $\varphi(j\omega)$ dargestellt.

3.1.2 Bedeutung von Abtastung und Nyquistkriterium im Frequenzbereich

Wenn man den Dirac-Stoß unendlich oft mit dem Zeitabstand T wiederholt, so erhält man einen Dirac-Kamm. Als Formel, mit dazugehöriger Fourier-Transformierter und Abtastfrequenz ω_s:

$$w_a(t) = \sum_{n=-\infty}^{\infty} \delta_a(t - nT) \circ\!\!-\!\!\bullet\, W_a(j\omega) = \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta_a(j(\omega - n\omega_s)) \tag{23}$$

Somit kann die Abtastung eines Signals $x_a(t)$ auch als Multiplikation mit dem Dirac-Kamm dargestellt werden.

$$x_{a,s}(t) = x_a(t) \cdot w_a(t) \tag{24}$$

Die Transformation in den Frequenzbereich liefert

$$x_{a,s}(t) \circ\!\!-\!\!\bullet\, X_{a,s}(j\omega) = \frac{1}{2\pi} X_a(j\omega) \star W_a(j\omega) \ . \tag{25}$$

Und mit Einsetzen der Fourier-Transformierten des Dirac-Kamms folgt

$$X_{a,s}(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X_a(j(\omega - n\omega_s)) \ . \tag{26}$$

Das heißt, dass *das Abtasten eines Signales mit der Abtastfrequenz f_s direkt dazu führt, dass sich das Spektrum des Signals mit der Periode $\omega_s = 2\pi f_s$ wiederholt!* Beispielhaft ist das in Abbildung 8 dargestellt: das Spektrum $X_a(j\omega)$ unseres analogen Signals sei a), die Fourier-Transformierte $W_a(j\omega)$ des Dirac-Kamms ist b). Damit ergibt sich das Spektrum $X_{a,s}(j\omega)$ des abgetasteten Signals zu c). Betrachten wir nun noch den Fall, dass das Nyquistkriterium bei der Abtastung nicht erfüllt ist. Das bedeutet der Abstand zwischen den Spektren, also die Abtastrate f_s, ist zu klein gewählt. Durch die periodische Wiederholung des Spektrums im Frequenzbereich kommt es zur Überlappung der einzelnen Spektren. Dies ist als sogenanntes **Aliasing** zu sehen in d).

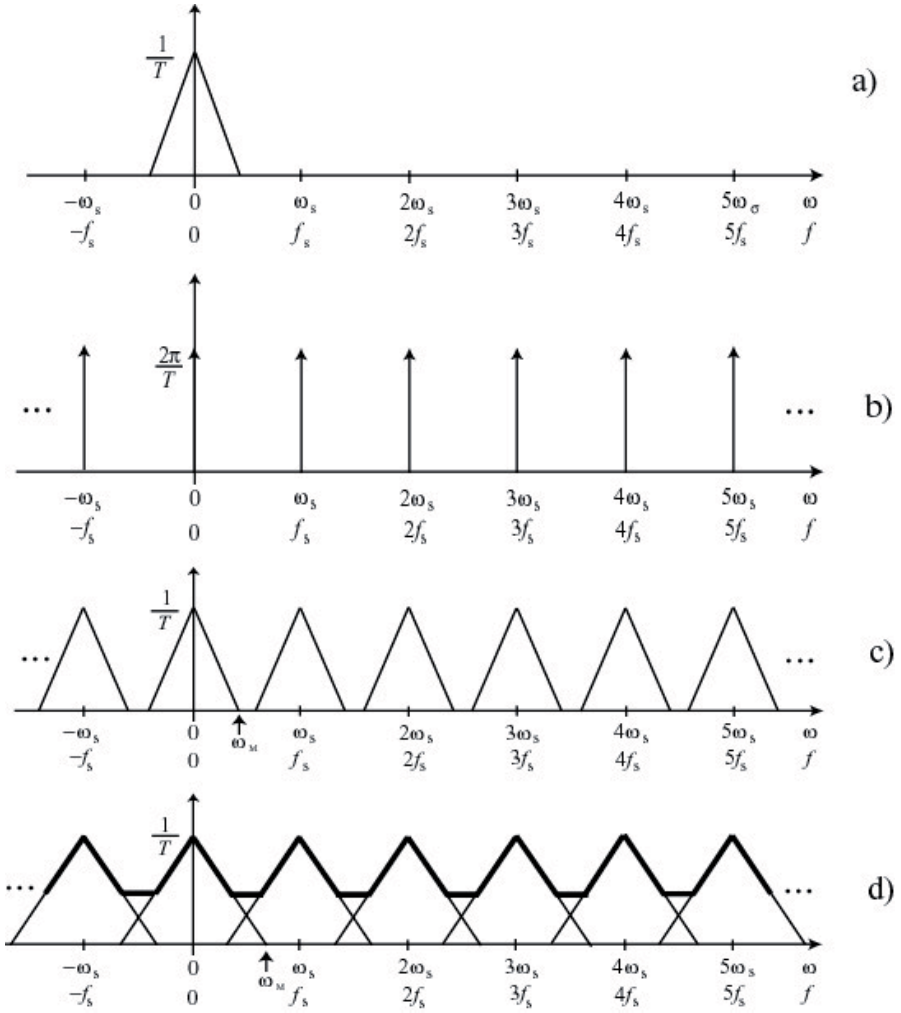


Abbildung 8: Auswirkung der Abtastung im Frequenzbereich. Ist die Abtastrate zu klein, ergeben sich verfälschte Spektren (d)). Die Rückgewinnung des analogen Signals ist nicht mehr möglich.

Wie in c) zu sehen ist, beinhaltet die periodische Wiederholung der Spektren keinerlei neue Information. Es reicht also eigentlich den Bereich von einem Spektrum zu betrachten. Dazu normalisieren wir die Frequenz, wie in Abbildung 9, auf den Bereich einer Abtastperiode:

$$\Omega = \omega T = 2\pi \frac{f}{f_s} \ . \tag{27}$$

Mit dieser normalisierten Frequenz wird die Fourier-Transformation von zeitdiskreten Signalen (DTFT, Discrete Time Fourier Transformation) wie folgt angegeben:

$$X(e^{j\Omega}) = \mathrm{DTFT}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j\Omega n} \tag{28}$$

Achtung! DTFT ≠ DFT (Diskrete Fourier-Transformation)

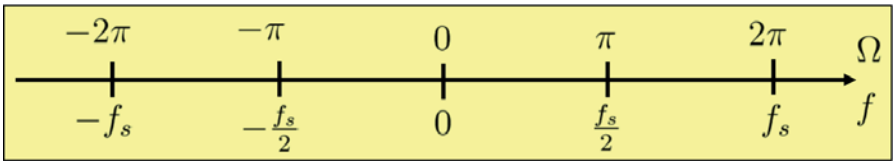


Abbildung 9: Normierte Kreisfrequenz Ω und lineare Frequenz f. Aus [Fi-DSV].

3.2 Z-Transformation

Eine Problematik der zeitdiskreten Fourier-Transformation ist, dass sie für Funktionen wie z.B. die Sprungfunktion nicht konvergiert. Fügen wir allerdings eine exponentielle Gewichtung, in Analogie zur Laplace-Transformation, hinzu, so kommen wir zur Z-Transformation. Als Erinnerung die Laplace-Transformation:

$$\mathfrak{L}\{x_a(t)\} = X_a(s) = \int_{-\infty}^{\infty} x_a(t) e^{-st}\,\mathrm{d}t, \quad \text{mit} \quad s = \sigma + j\omega\ . \tag{29}$$

Der Unterschied zur Fourier-Transformation: $e^{-s} = e^{-\sigma} \cdot e^{-j\omega}$. Der exponentielle Gewichtungsfaktor $e^{-\sigma}$ erzwingt Konvergenz für viele Funktionen. Hieraus erhalten wir also mit $X_a(s = j\omega)$ die Fourier Transformation. Damit haben wir für abgetastete Signale wieder den Bezug zur Abtastfrequenz. Wir definieren:

$$z = e^{sT}\ . \tag{30}$$

Damit ergibt sich die Z-Transformation als

$$\mathfrak{Z}\{x(n)\} = X(z) = \sum_{n=-\infty}^{\infty} x(n) \cdot z^{-n}\ . \tag{31}$$

Der Zusammenhang zwischen Laplace- und Z-Ebene ist in Abbildung 10 veranschaulicht.

3.2.1 Konvergenzbereich

Die Z-Transformation ergibt eine diskrete, komplexe Reihe, genauer eine Laurent-Reihe. Wie aus der Funktionentheorie bekannt sein sollte, konvergiert diese nur für bestimmte $z \in \mathbb{C}$. Der Bereich in $\mathbb{C}$, in dem die Reihe konvergiert, ist der sogenannte *Konvergenzbereich* (engl. region of convergence, ROC). Dieser Hängt nur von $r = |z|$ ab.

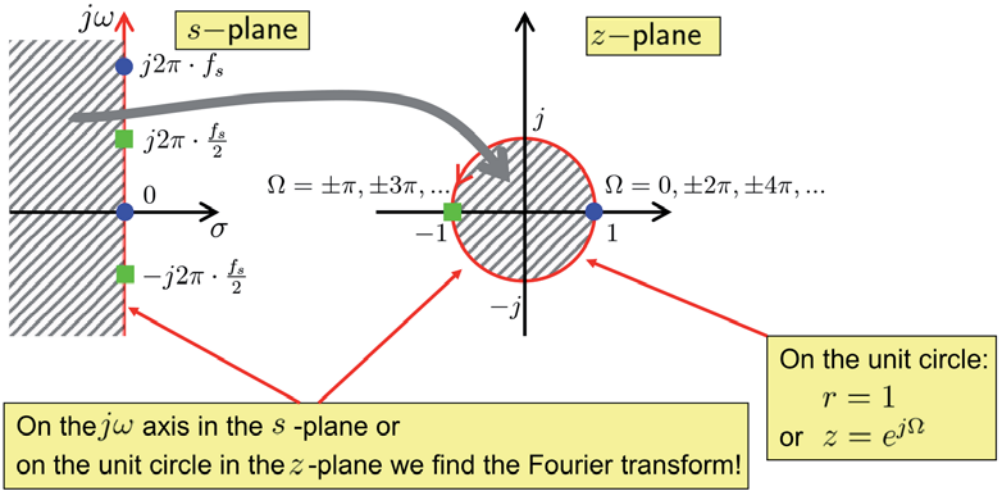


Abbildung 10: Zusammenhang zwischen Laplace- und Z-Ebene. Aus [Fi-DSV].

3.2.2 Eigenschaften

Für die Rechnungen im Z-Bereich gelten ähnliche Regeln wie für die im Fourier-Bereich. Auch hier zur Vollständigkeit eine Tabelle mit den wichtigsten Eigenschaften bzw. Rechenregeln.

Property	time domain ○—● z-domain	
Transform	$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz$	$\mathfrak{Z}\{x(n)\} = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$ $\mathfrak{Z}_1\{x(n)\} = \sum_{n=0}^{\infty} x(n) z^{-n}$
Conjugation	$x^*(n)$	$X^*(z^*)$
Real-valued $x(n)$	$x(n) = x^*(n)$	$X(z) = X^*(z^*)$
Time Shift	$x(n - n_0)$	$X(z) \cdot z^{-n_0}$ $\quad n_0 \in \mathbb{N}_0$
	$x(n + n_0)$	$\mathfrak{Z}\{x(n+n_0)\} = X(z) \cdot z^{n_0}$ $\quad n_0 \in \mathbb{N}_0$
		$\mathfrak{Z}_1\{x(n+n_0)\} = X(z) \cdot z^{n_0} - \sum_{n=0}^{n_0-1} x(n) z^{n_0-n}$
Time Mirroring	$x(-n)$	$X(1/z)$
Subsampling	$x(n \cdot L)$	$X(z^{1/L})$ $\quad L \in \mathbb{N}$
Linear Weighting	$n \cdot x(n)$	$-z \frac{d}{dz} X(z)$
Exp. Weighting	$c^n \cdot x(n)$	$X(z/c)$ $\quad c \in \mathbb{C}$
Superposition	$c_1 \cdot x(n) + c_2 \cdot y(n)$	$c_1 \cdot X(z) + c_2 \cdot Y(z)$
Convolution	$x(n) * y(n)$	$X(z) \cdot Y(z)$
Modulation	$x(n) \cdot y(n)$	$\frac{1}{2\pi j} \oint_C X(v) Y(z/v) v^{-1} dv$

Abbildung 11: Tabelle der Eigenschaften der Z-Transformation. Aus [Fi-DSV].

3.2.3 Systeme im Z-Bereich

Nun wollen wir anhand der Z-Transformation Systeme analysieren, insbesondere rekursive und nicht-rekursive Filter klassifizieren. Dazu benötigen wir die Z-Übertragungsfunktion des Systems. Sie wird durch Z-Transformation der Impulsantwort gewonnen:

$$H(z) = \sum_{n=-\infty}^{\infty} h(n) \cdot z^{-n} \; . \tag{32}$$

Die Verzögerung um einen Abtastwert entspricht im Z-Bereich der Multiplikation mit z^{-1}. Damit ergibt sich die Z-Transformierte der allgemeinen Differenzengleichungen zu:

$$\begin{aligned} 0 &= \sum_{\nu=0}^{N_a} a_\nu \cdot y(n-\nu) + \sum_{\mu=0}^{N_b} b_\mu \cdot x(n-\mu) \\ &\quad \downarrow \\ 0 &= \sum_{\nu=0}^{N_a} a_\nu \cdot Y(z) \cdot z^{-\nu} + \sum_{\mu=0}^{N_b} b_\mu \cdot X(z) \cdot z^{-\mu} \end{aligned} \tag{33}$$

Setzen wir dies in

$$H(z) = \frac{Y(z)}{X(z)} \tag{34}$$

ein, so erhalten wir die Z-Übertragungsfunktion zu

$$\begin{aligned} H(z) &= \frac{\sum_{\mu=0}^{N_b} b_\mu \cdot z^{-\mu}}{-\sum_{\nu=0}^{N_a} a_\nu \cdot z^{-\nu}} \overset{-a_0=1}{=} \frac{\sum_{\mu=0}^{N_b} b_\mu \cdot z^{-\mu}}{1 - \sum_{\nu=1}^{N_a} a_\nu \cdot z^{-\nu}} = z^{N_a - N_b} \cdot \frac{\sum_{\mu=0}^{N_b} b_\mu \cdot z^{N_b-\mu}}{z^{N_a} - \sum_{\nu=1}^{N_a} a_\nu \cdot z^{N_a-\nu}} \\ &= \underbrace{z^{N_a-N_b}}_{\text{Pol oder Nullstelle der Ordnung} |N_a - N_b|} \cdot b_0 \frac{\prod_{\mu=1}^{N_b}(z - z_{0\mu})}{\prod_{\nu=1}^{N_a}(z - z_{\infty\nu})} = b_0 \frac{\prod_{\mu=1}^{N_b}(1 - z_{0\mu} z^{-1})}{\prod_{\nu=1}^{N_a}(1 - z_{\infty\nu} z^{-1})} \; . \end{aligned} \tag{35}$$

Mit folgender Benennung fällt es nun leicht rekursive und nicht-rekursive Filter zu klassifizieren:

$$\begin{aligned} H(z) &= \frac{\sum_{\mu=0}^{N_b} b_\mu \cdot z^{-\mu}}{1 - \sum_{\nu=1}^{N_a} a_\nu \cdot z^{-\nu}} = \frac{B(z)}{1 - A(z)} \quad \text{mit} \\ B(z) &= \sum_{\mu=0}^{N_b} b_\mu \cdot z^{-\mu} \quad \text{und} \quad A(z) = \sum_{\mu=1}^{N_a} a_\nu \cdot z^{-\nu} \end{aligned} \tag{36}$$

Das heißt ein Filter ist

$$\begin{aligned} \text{nicht-rekursiv, wenn} \quad & H(z) = B(z) \\ \text{rekursiv, wenn} \quad & H(z) = \frac{b_0}{1 - A(z)} \end{aligned} \tag{37}$$

Zu beachten ist, dass ein System durch seine Z-Übertragungsfunktion **nur** zusammen mit dem Konvergenzbereich vollständig beschrieben ist.

3.3 Pol-Nullstellen Diagramme

Die Form der Z-Übertragungsfunktion aus Gleichung (35) erlaubt eine Zerlegung in Pol- und Nullstellen. Es gelten folgende Zusammenhänge:

1. Ist $h(n)$ eine rechtsseitige Sequenz, liegt der Konvergenzbereich außerhalb des Kreises durch die vom Ursprung am weitesten entfernte Polstelle.

2. Ist ein System kausal und stabil, so liegen alle Pole innerhalb des Einheitskreises.

3. Damit die Impulsantwort eines Systems reell ist, müssen alle Pol- und Nullstellen komplex konjugierte Paare sein.

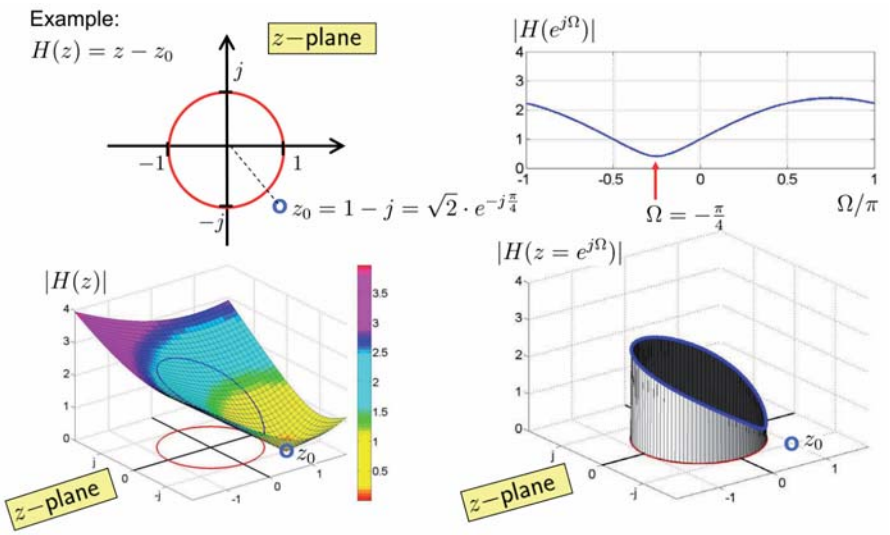


Abbildung 12: Analyse einer Beispiel Z-Übertragungsfunktion $H(z) = z - z_0$. Aus [Fi-DSV].

Den Frequenzgang erhalten wir durch $H(e^{j\Omega}) = H(z = e^{j\Omega})$, d.h. anschaulich entspricht der Amplitudengang $|H(e^{j\Omega})|$ dem *Ablaufen des Einheitskreises in der Z-Ebene* (siehe Abbildungen 12). Befindet sich ein Pol in der Nähe des Einheitskreises ist im Amplitudengang an dieser Stelle ein Anstieg zu sehen. Bei einer Nullstelle nahe des Einheitskreises ist entsprechend ein Abfall zu beobachten.

Achtung! Dies lässt bisher keine Aussage über den Phasengang zu. Ein nicht-linearer Phasenverlauf (d.h. frequenzabhängige Laufzeitunterschiede) kann Signale verzerren.

3.4 Diskrete Fourier-Transformation

Die oben eingeführte DTFT arbeitet mit einer gesamten (unendlich langen) Sequenz und liefert für diskrete Signale ein nach wie vor kontinuierliches Spektrum. Für die Verarbeitung mit digitaler Hardware ist es allerdings notwendig ein diskretes Spektrum zu

erlangen. Zusätzlich ist es für viele Anwendungen hilfreich, nicht das gesamte Eingangssignal verarbeiten zu müssen, sondern das Spektrum einzelner Blöcke zu betrachten. Hierfür wird die diskrete Fourier-Transformation (DFT) genutzt: wir erhalten das diskrete Spektrum (als Funktion von k) von einem Block der Länge K eines diskreten Eingangssignals. Die mathematische Definition ist

$$\begin{aligned} \mathrm{DFT}\{x(n)\} = X(k) &= \sum_{n=0}^{K-1} x(n) e^{-j2\pi \frac{nk}{K}} \\ &\updownarrow \\ \mathrm{IDFT}\{X(K)\} = x(n) &= \frac{1}{K} \sum_{k=0}^{K-1} X(k) e^{j2\pi \frac{nk}{K}} \ , \end{aligned} \tag{38}$$

wobei n der Index in die Samplewerte im Zeitbereich und k der Index in die Frequenzwerte im nun diskreten Frequenzbereich ist. Als Beispiel, in Abbildung 13, die DTFT und die DFT mit $K = 16$ für das Signal x(n) zum Vergleich:

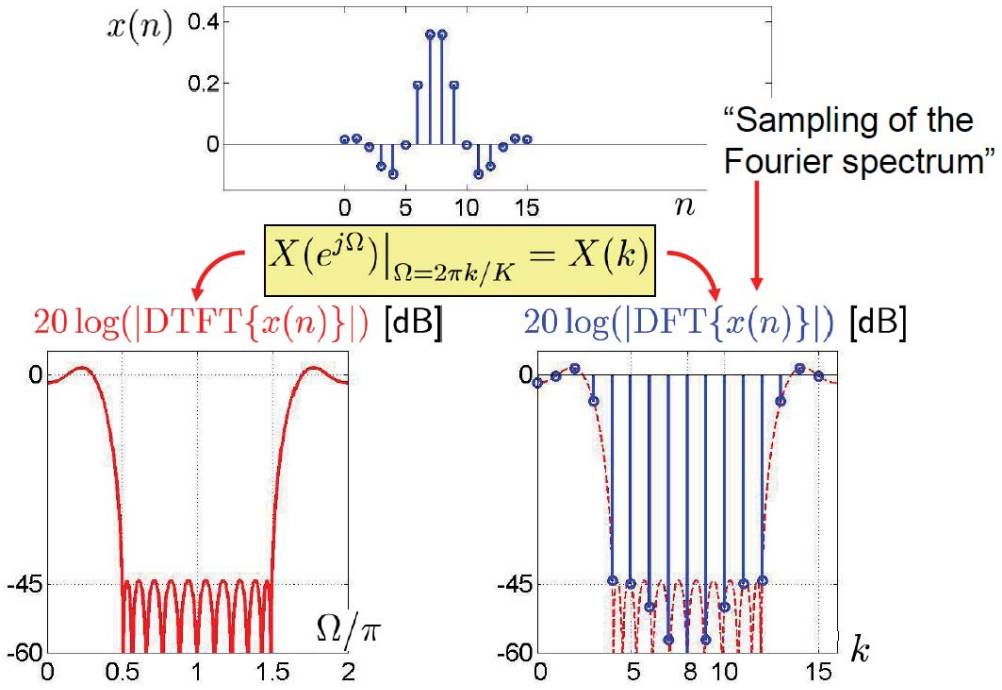


Abbildung 13: Beispiel zur diskreten Fourier-Transformation. Aus [Fi-DSV].

Abschließend die wichtigsten Rechenregeln der DFT als Tabelle in Abbildung 14.

$x(n) \circ\!\!-\!\!\bullet \mathrm{DFT}\{x(n)\} = X(k) = \mathrm{Re}\{X(k)\} + j\mathrm{Im}\{X(k)\}$

Periodic continuation	$\tilde{x}(\nu) = x(\nu \mod K), \quad \nu \in \mathbb{Z}$	$\tilde{X}(\kappa) = X(\kappa \mod K), \quad \kappa \in \mathbb{Z}$
Property	time domain $\circ\!\!-\!\!\bullet$ frequency domain $a(n) = ...\|_{n=0,1,...,K-1} \circ\!\!-\!\!\bullet A(k) = ...\|_{k=0,1,...,K-1}$	
Transform	$x(n) = \frac{1}{K}\sum_{k=0}^{K-1} X(k)e^{j2\pi\frac{nk}{K}}$ $n = 0, 1, ..., K-1$	$X(k) = \sum_{n=0}^{K-1} x(n)e^{-j2\pi\frac{nk}{K}}$ $k = 0, 1, ..., K-1$
Even part	$\tilde{x}^{(e)}(n) = \frac{1}{2}\tilde{x}(n) + \frac{1}{2}\tilde{x}^*(-n)$	$\mathrm{Re}\{X(k)\}$
Odd part	$\tilde{x}^{(o)}(n) = \frac{1}{2}\tilde{x}(n) - \frac{1}{2}\tilde{x}^*(-n)$	$j\mathrm{Im}\{X(k)\}$
Conjugation	$x^*(n)$	$\tilde{X}^*(-k) = \tilde{X}^*(K-k)$
Real-valued $x(n)$	$x(n) = x^*(n)$	$X(k) = \tilde{X}^*(K-k)$
Time Shift	$\tilde{x}(n-n_0)$	$X(k)\cdot e^{-j2\pi\frac{n_0 k}{K}} \quad n_0 \in \mathbb{Z}$
Frequency Shift	$x(n)\cdot e^{j2\pi\frac{nk_0}{K}}$	$\tilde{X}(k-k_0) \quad k_0 \in \mathbb{Z}$
Time & Freq. Mirroring	$\tilde{x}(-n)$	$\tilde{X}(-k) = \tilde{X}(K-k)$
Superposition	$c_1\cdot x(n) + c_2\cdot y(n)$	$c_1\cdot X(k) + c_2\cdot Y(k)$
Circular Convolution	$x(n)\otimes y(n)$	$X(k)\cdot Y(k)$
Modulation	$x(n)\cdot y(n)$	$\frac{1}{K}\cdot X(k)\otimes Y(k)$
Parseval's Theorem	$\sum_{n=0}^{K-1}\|x(n)\|^2 = \frac{1}{K}\sum_{k=0}^{K-1}\|X(k)\|^2$	

$n = 0, 1, ..., K-1$ $\qquad$ $k = 0, 1, ..., K-1$

Abbildung 14: Tabelle der Eigenschaften der diskreten Fourier-Transformation. Aus [Fi-DSV].

3.4.1 Zirkulare Faltung

Die lineare (normale) Faltung ist sehr rechenintensiv. Mit der DFT eröffnet sich nun eine wesentlich effizientere Möglichkeit der Faltung: die *zirkulare* oder *zyklische Faltung*. Sie ergibt sich aus der Multiplikation der DFT-Spektren zweier Signale:

$$y(n) = x(n) \star h(n) \quad \underrightarrow{\text{DFT}} \quad Y(k) = X(k) \cdot H(k) \quad \multimap \bullet \quad y'(n) = \underbrace{x(n) \otimes h(n)}_{Zirk.Faltung} \; . \tag{39}$$

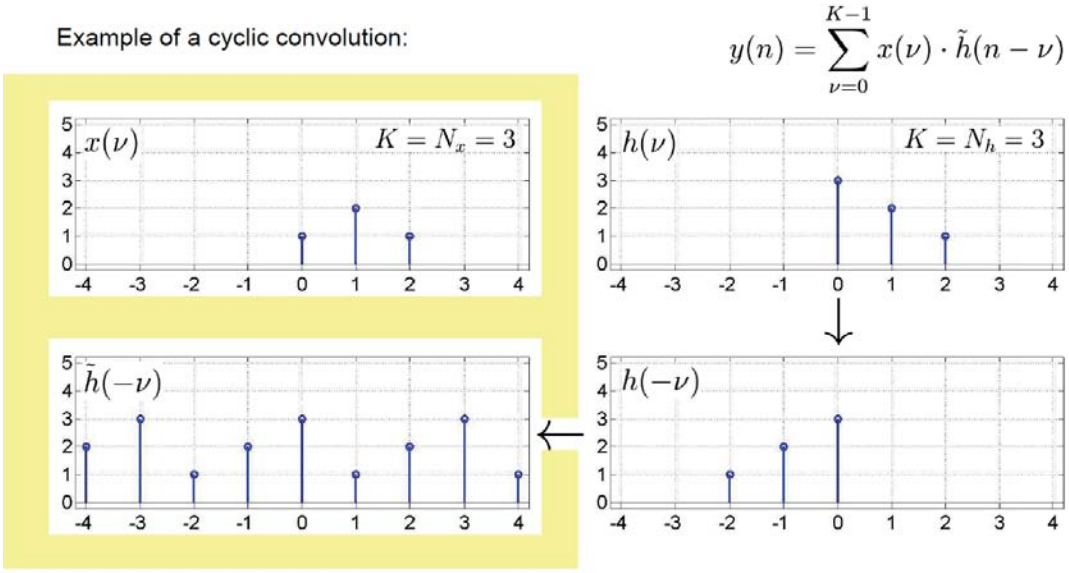
Die Multiplikation im DFT-Bereich entspricht also der zyklischen Faltung im Zeitbereich. Für die direkte Definition im zeitdiskreten Bereich ist es wichtig, dass die Signale mit der Blocklänge K *periodisch fortgesetzt* werden und K nicht zu klein gewählt wird. In der Benennung wird die periodische Fortsetzung durch eine Tilde gekennzeichnet. Aus dem Signal $h(n)$ wird also das Signal $\tilde{h}(n)$. Die Definition im Zeitbereich:

$$y(n) = x(n) \otimes h(n) = \sum_{\nu=0}^{K-1} \tilde{x}(n) \cdot \tilde{h}(n-\nu) = \sum_{\nu=0}^{K-1} x(n) \cdot \tilde{h}(n-\nu) \tag{40}$$

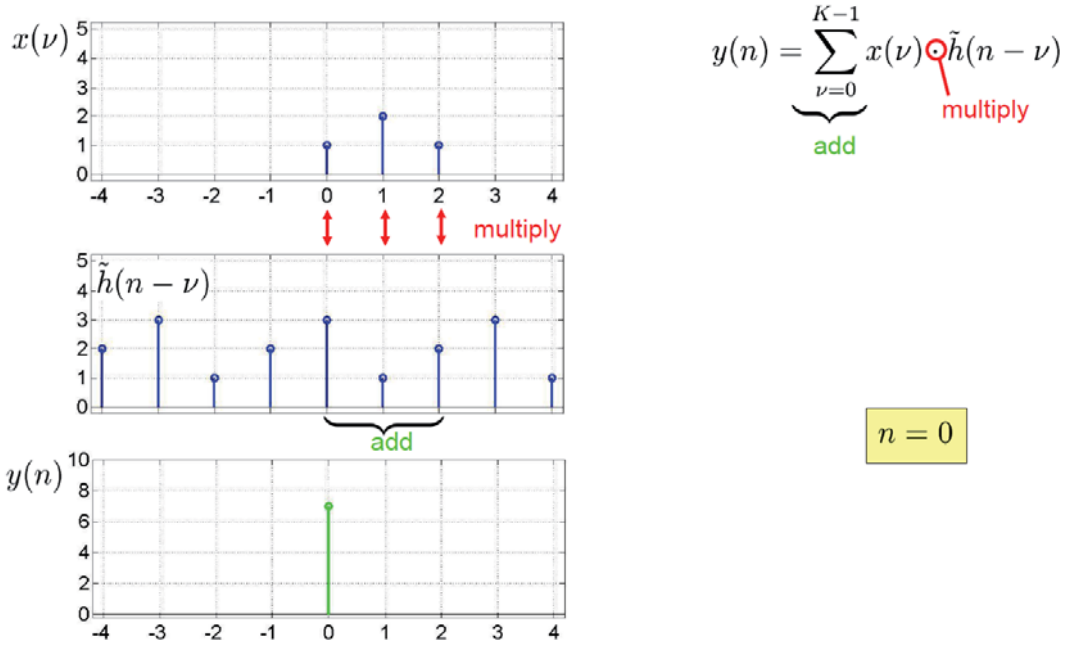
Die lineare (normale) und die zirkuläre Faltung sind für $n = 0, 1, ..., K-1$ gleich, wenn für die Länge K der DFT gilt:

$$K \geq N_x + N_h - 1 \tag{41}$$

wobei $N_{x,h}$ die Länge der Sequenzen $x(n)$ bzw. $h(n)$ ist. Dies kann man immer erreichen, indem man K entsprechend wählt, und die Signale soweit nötig mit Nullen auffüllt (zero padding). Abbildungen 15 bis 17 zeigen ein Beispiel zur zirkulären Faltung.

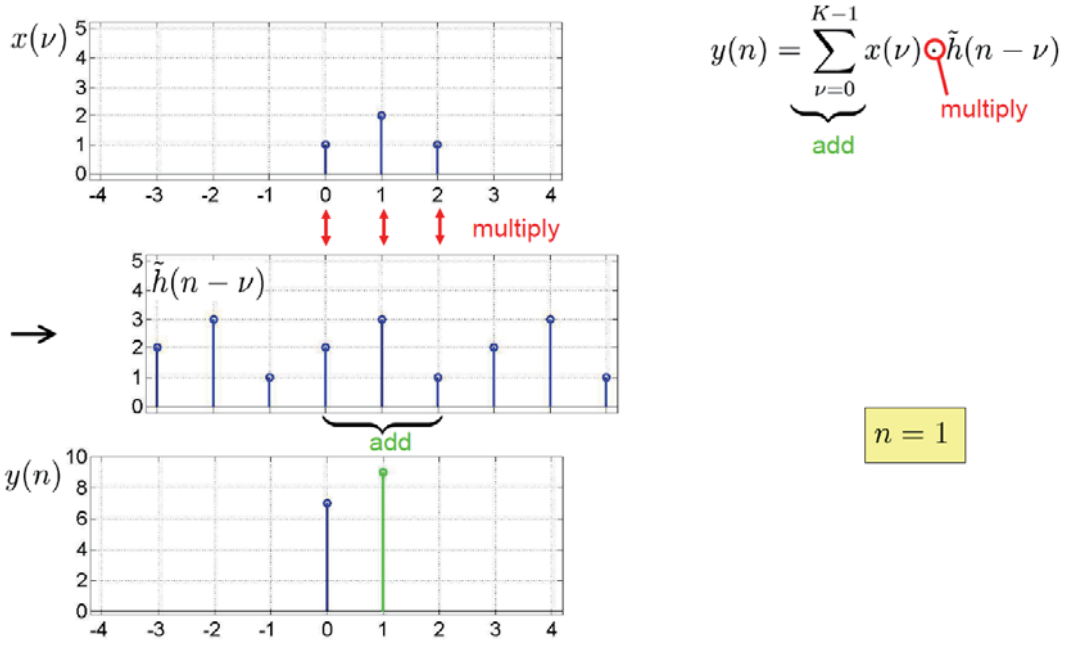


(a) „Vorbereitung“

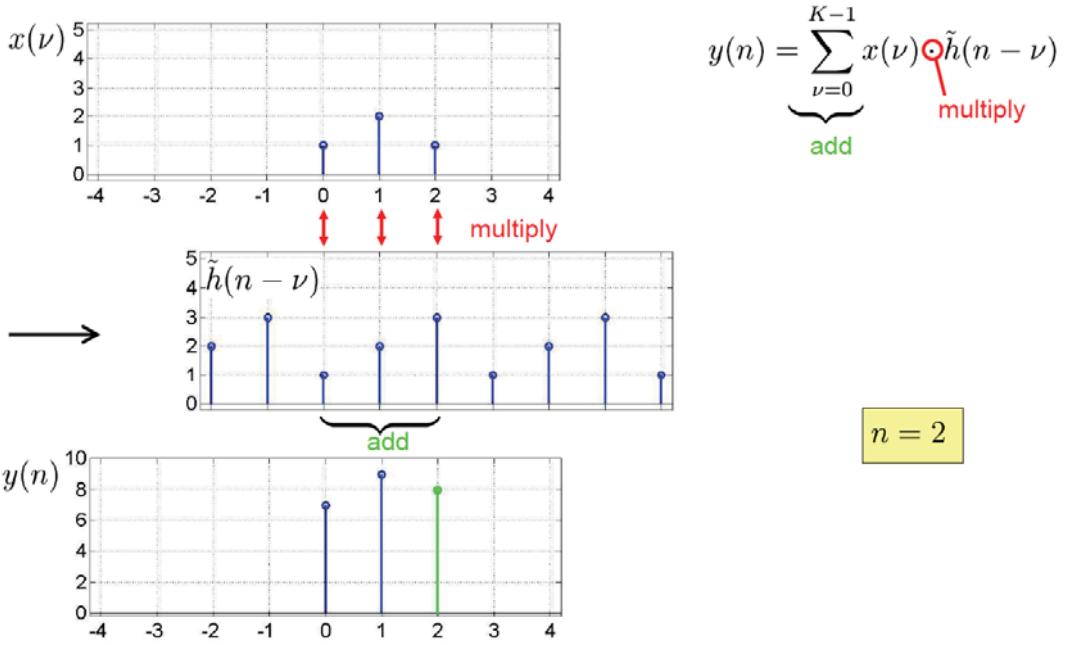


(b) Schritt 1

Abbildung 15: Beispiel zur zirkulären Faltung. Aus [Fi-DSV].



(a) Schritt 2



(b) Schritt 3

Abbildung 16: Beispiel zur zirkulären Faltung. Aus [Fi-DSV].

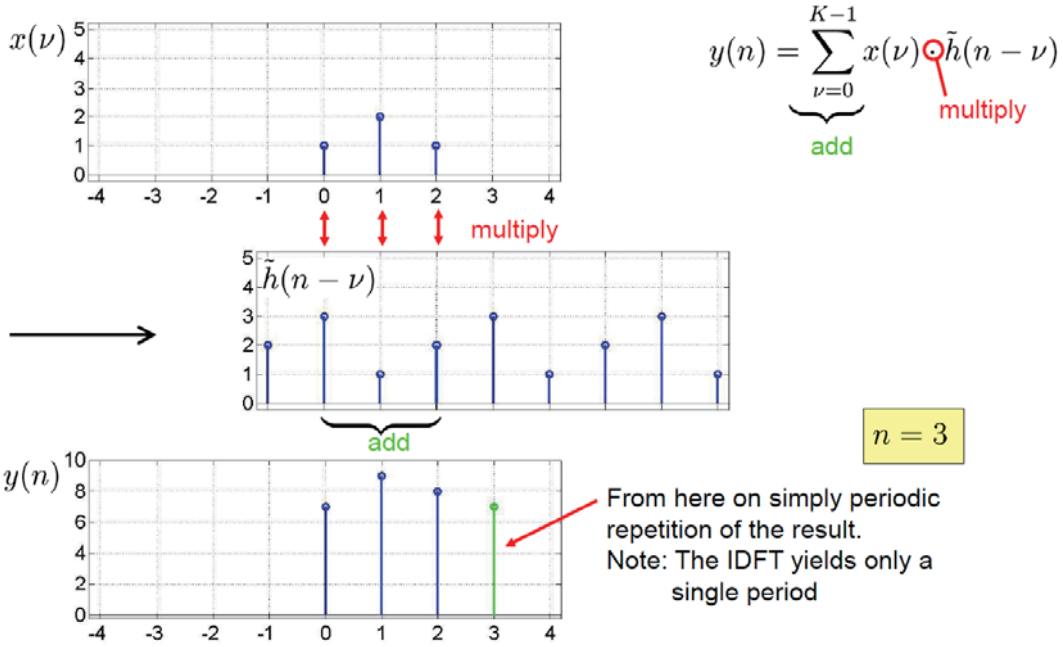


Abbildung 17: Beispiel zur zirkulären Faltung (Schritt 4). Aus [Fi-DSV].

3.4.2 Schnelle Faltung mit Overlap-Add

In der heutigen Signalverarbeitung werden Verfahren gefordert, die in der Lage sind, Signale bereits während sie eigentlich noch empfangen werden zu falten. Zum Beispiel wenn eine feste, endliche Filter-Impulsantwort mit einem (theoretisch) unendlich langen Eingangssignal gefaltet werden soll. Eine schnelle und effiziente Lösung hierfür ist das *Overlap-Add*-Verfahren. Beim Overlap-Add wird das Eingangssignal in Blöcke partitioniert. Jeweils einer der Blöcke und die Impulsantwort werden für die schnelle Fourier-Transformation auf die Länge K mit Nullen aufgefüllt und im Frequenzbereich multipliziert. Das Faltungsergebnis ist dann länger als der ursprüngliche ausgeschnittene Block des Eingangssignals. Der „Überhang“ wird auf das Ergebnis der Faltung des nächsten Blockes addiert. In Abbildung 18 ist ein anschauliches Beispiel dargestellt.

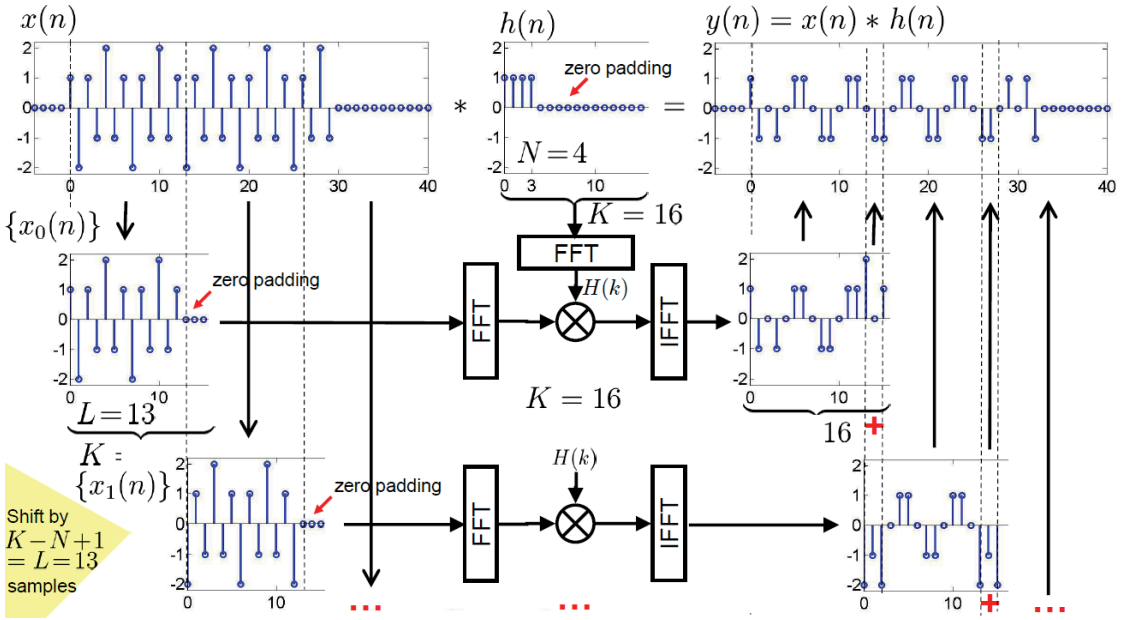


Abbildung 18: Overlap-Add Faltung von beliebig langem Signal $x(n)$ mit der Filter-Impulsantwort $h(n)$. Aus [Fi-DSV].

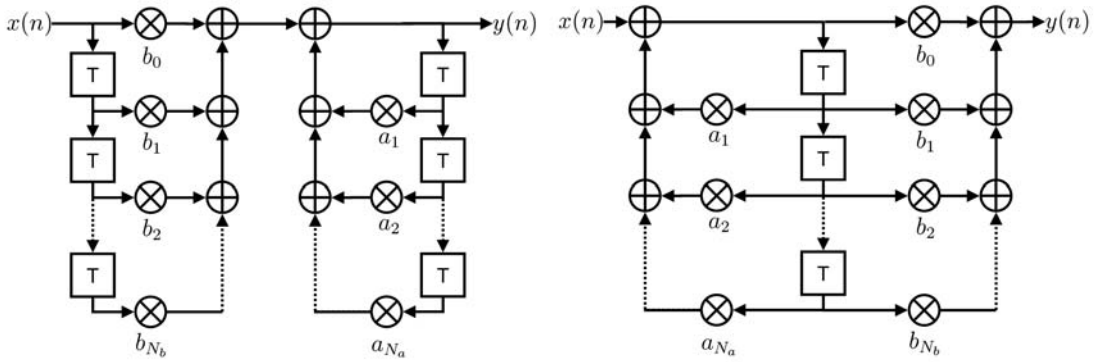
4 Digitale Filter

Filter begegnen uns in der Signalverarbeitung in verschiedensten Formen und vor allem für unterschiedlichste Anwendungen. Die beiden bekanntesten Formen sind wohl das Tiefpassfilter, welches tiefe Frequenzen durchlässt und höhere Frequenzen dämpft, sowie das Hochpassfilter, welches entsprechend nur hohe Frequenzen durchlässt. Ein anschauliches Beispiel (allerdings meist mit analogen Filtern) findet sich beim Mehr-Wege-Lautsprecher: besteht ein Lautsprecher aus einem Tief-/Mitteltöner und einem Hochtöner, so kann man nun mit zwei Filtern den beiden Tönern nur die Signalkomponenten zukommen lassen, die sie auch wirklich wiedergeben sollen.

Digitale Filter arbeiten entsprechend nur mit digitalen Signalen. Sie werden nicht durch einzelne elektrische Bauteile realisiert, sondern, beispielsweise von einem Signalprozessor, durch mathematische Operationen auf das digitale Signal angewandt. Sie lassen sich in zwei Gruppen aufteilen: zum einen Filter mit endlicher Impulsantwort (Finite Impulse Response, *FIR*) und zum anderen Filter mit unendlicher Impulsantwort (Infinite Impulse Response, *IIR*).

4.1 Filterstrukturen

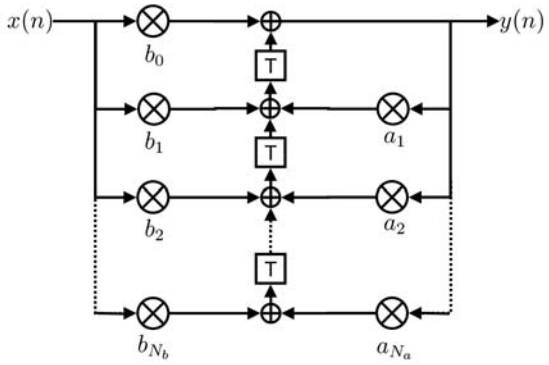
In Bezug auf Gleichung 36 in Kapitel 3.2.3 sind hier noch die wichtigsten Strukturen von Blockschaltbildern dargestellt:



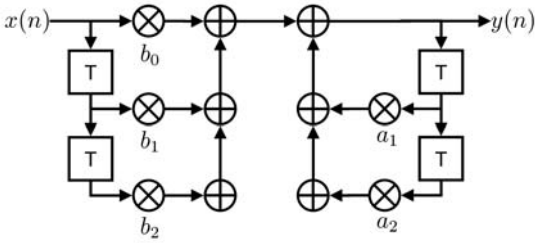
(a) Direktform I (b) Direktform II

Abbildung 19: Direktformen der Filterstruktur. Aus [Fi-DSV].

Abbildung 20 zeigt unter anderem ein sogenanntes *Biquad-Filter* (abgekürzt für biquadratic), ein Filter bei dem Zähler und Nenner jeweils Polynome zweiter Ordnung sind. Aus Biquad-Filtern lassen sich Filter größerer Ordnung konstruieren. Einzelne Parameter sind dann justierbar ohne dass das komplette Filter geändert werden muss. Sie werden in der SHARC-DSP-Bibliothek verwendet, welche wir auch in diesem Praktikum nutzen.



(a) Transponierte Direktform II



(b) Rekursives Biquad-Filter

Abbildung 20: Weitere Formen der Filterstruktur. Aus [Fi-DSV].

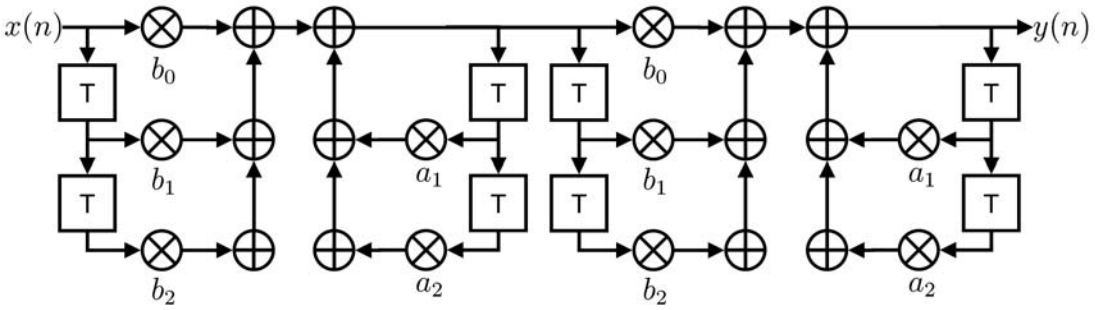


Abbildung 21: Biquad-Filter Kaskade. Aus [Fi-DSV].

4.2 Filterentwurf

Wenn man einen digitalen Filter konstruieren möchte, durchläuft man die folgenden Hauptschritte:

1. *Spezifikation*
2. *Design* eines zeitdiskreten und kausalen Filters
3. *Implementierung* eines Filters mit endlicher Rechengenauigkeit

Die Spezifikation ist zum größten Teil durch die Anwendung bestimmt, in der das Filter genutzt werden soll. Es ergeben sich aber noch weitere Entscheidungen. Zum einen bei der Charakterisierung des Amplitudengangs, zum anderen bei der Frage, ob das Filter minimal-, linear- oder beliebigphasig sein soll, und letztlich ob es ein FIR- oder IIR-Filter sein soll. Abbildung 22 zeigt ein übersichtliches Flussdiagramm zum Filterdesign. Die Grafik ist dem Skript zur Vorlesung „Digitale Signalverarbeitung“ entnommen, die **Kapitelnummern beziehen sich daher nicht auf das Skript zu diesem Praktikum!**

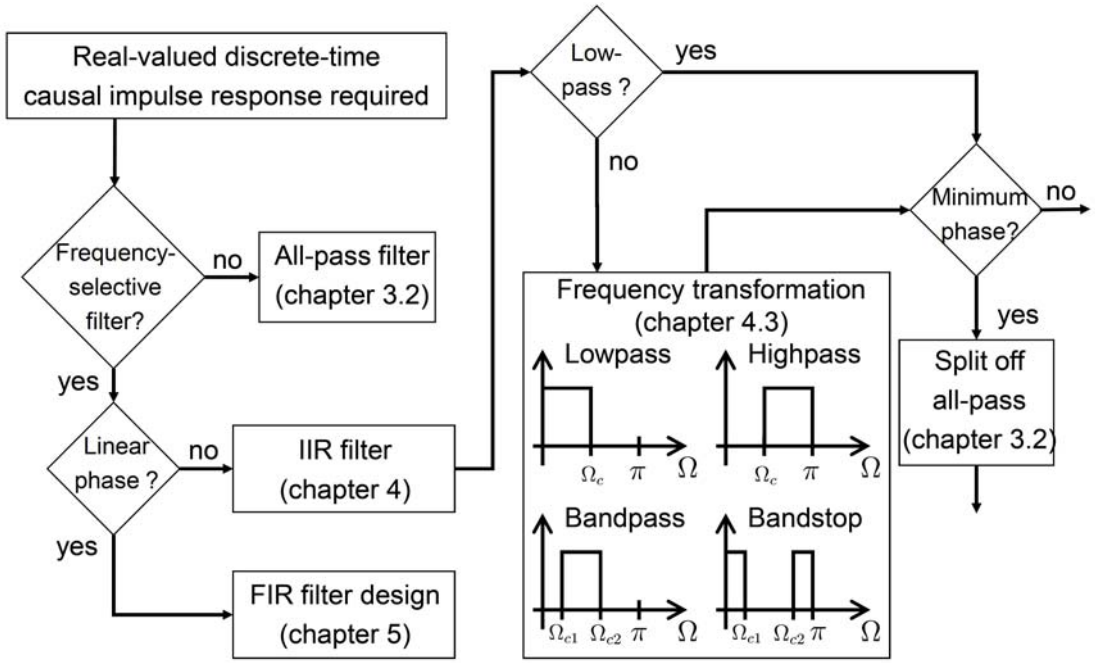


Abbildung 22: Flussdiagramm Filterdesign. Aus [Fi-DSV].
ACHTUNG: Die Kapitelnummern beziehen sich nicht auf dieses Skript!

4.3 IIR-Entwurf

Betrachten wir zunächst den Entwurf von IIR-Filtern. In Abschnitt 4.4 folgt dann der Entwurf von FIR-Filtern. Wir beginnen mit der Spezifikation. Die Richtlinien ergeben sich hier aus dem gewünschten Frequenzverhalten: Durchlassfrequenz Ω_p und Sperrfrequenz Ω_{st}, siehe Beispiel in Grafik 23. Dazugehörige Gütemaße sind definiert:

$$\begin{aligned} \text{Welligkeit im Durchlassbereich:} \quad & R_p = -20\,\log(1-\delta_p) \quad [\text{dB}] \\ \text{Sperrdämpfung:} \quad & d_{st} = -20\,\log(\delta_{st}) \quad [\text{dB}] \end{aligned} \tag{42}$$

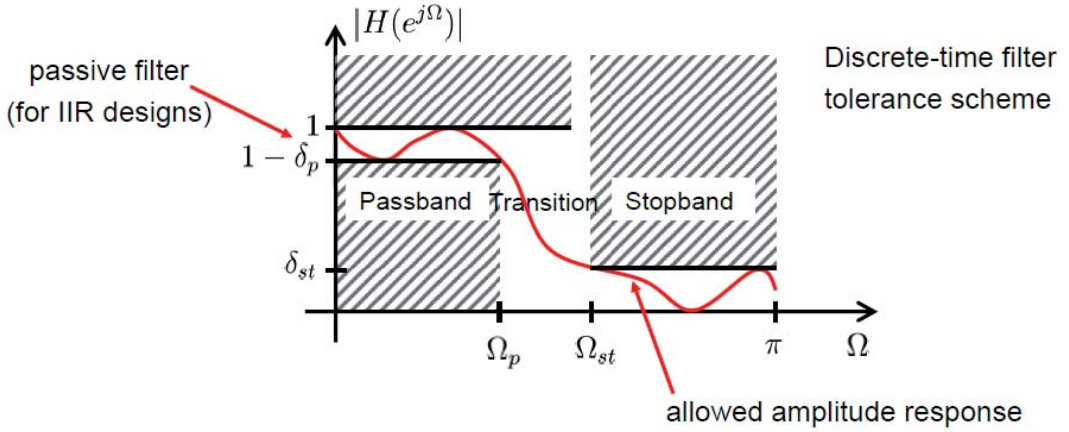


Abbildung 23: Toleranzschema und beispielhafter Amplitudenverlauf zum IIR-Filterentwurf. Aus [Fi-DSV].

In den nächsten Kapiteln werden einige wichtige analoge IIR-Filter vorgestellt. Die Überführung in den diskreten Bereich ist über die anschließend vorgestellte bilineare Transformation (Kapitel 4.3.4) möglich.

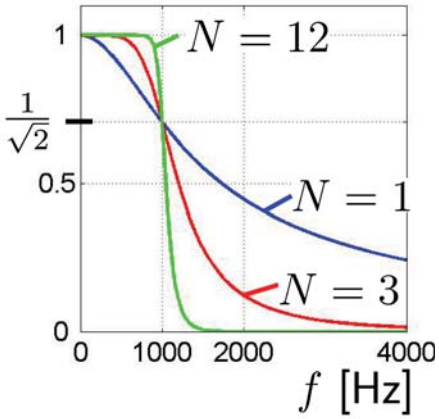
4.3.1 Butterworth-Filter

Anforderungen an Entwurf:

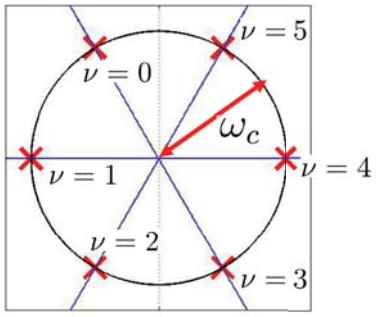
- flaches Amplitudenspektrum im Durchlassbereich bei $\omega = 0$
- die ersten $2 \cdot N - 1$ Ableitungen von $|H_a(j\omega)|^2$ sind null (bei $\omega = 0$)
- Tiefpass-Entwurf: monoton abnehmender Amplitudengang

Lösung: Butterworth mit Ordnung N

$$|H_a(j\omega)|^2 = \frac{1}{1 + \left(\frac{j\omega}{j\omega_c}\right)^{2N}} \tag{43}$$



(a) Frequenzgang $|H_a(j\omega)|$ abh. von N



(b) Pol-Nullst.-Diagr. in S-Ebene, $N = 3$

Abbildung 24: Amplitudengang und Pol-Nullstellen-Diagramm eines analogen Butterworth Entwurfs mit $f_c = 1000\,\text{Hz}$. Aus [Fi-DSV].

4.3.2 Chebyshev-Typ-I-Filter

Anforderungen an Entwurf:

- verteile Näherungsfehler gleichmässig über das Durchlassband
- gleichmässige Welligkeit („Equiripple“) im Durchlassband und monotones Verhalten im Sperrband

Lösung: Chebyshev-Typ-I

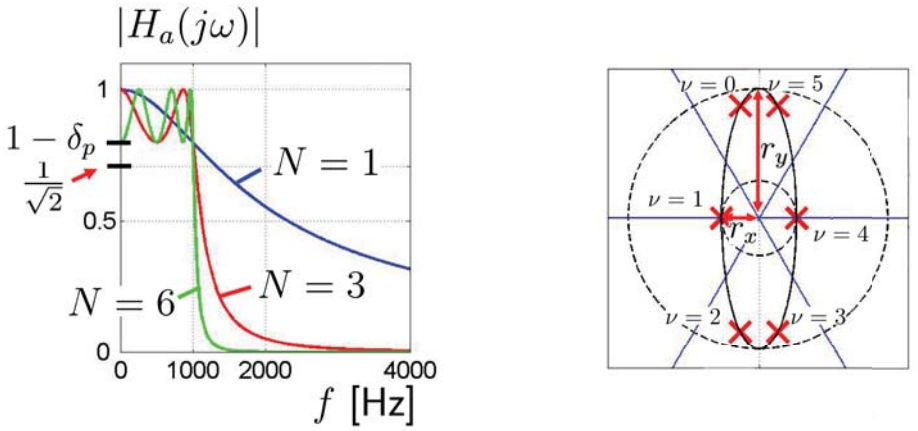
$$|H_a(j\omega)|^2 = \frac{1}{1 + a^2 \cdot V_N^2\left(\frac{j\omega}{j\omega_c}\right)} \tag{44}$$

Dabei ist

$$V_N\left(\frac{j\omega}{j\omega_c}\right) = \cos\left(N \cdot \arccos\left(\frac{j\omega}{j\omega_c}\right)\right) \quad \text{mit} \quad \left|\frac{j\omega}{j\omega_c}\right| \leq 1 \tag{45}$$

und über die Welligkeit im Durchlassbereich ergibt sich

$$a = \sqrt{\frac{1}{(1-\delta_p)^2} - 1} \quad , \quad 0 \leq \Omega \leq \Omega_c \tag{46}$$



(a) Frequenzgang $|H_a(j\omega)|$ abh. von N (b) Pol-Nullst.-Diagr. in S-Ebene, $N=3$

Abbildung 25: Amplitudengang und Pol-Nullstellen-Diagramm eines analogen Chebyshev-Typ-I-Entwurfs mit $f_c = 1000\,\text{Hz}$. Aus [Fi-DSV].

Nebenbemerkung: wird die Welligkeit statt in das Durchlassband gleichmäßig in das Sperrband gelegt, so erhalten wir ein Chebyshev-Typ-II-Filter.

4.3.3 Cauer-Filter

Anforderungen an Entwurf:

- verteile Näherungsfehler gleichmässig über das Durchlassband **und** Sperrband
- sehr steiler Übergang Durchlass- zu Sperrband
- Entwürfe mit niedrigster Ordnung möglich
- starke Phasenverzerrung tolerierbar

Lösung: Cauer-Filter

Welligkeit in Durchlass- und Sperrband unabhängig einstellbar

- Welligkeit im Sperrband gegen Null: Filter wird zu Chebyshev-Filter vom Typ I
- Welligkeit im Durchlassband gegen Null: Chebyshev-Typ-II-Filter
- Beide gegen Null: Butterworth Filter

4.3.4 Bilineare Transformation

Die in den vorigen Kapiteln vorgestellten Filter sind zeitkontinuierliche Filter. Um ein kontinuierliches Filter $H_a(s)$ ins Zeitdiskrete (dann $H(z)$) zu überführen, nutzen wir die bilineare Transformation. Als Erinnerung: das Spektrum eines abgetasteten Signales wiederholt sich periodisch, die Wiederholungen sind für uns aber nicht relevant. Darum bilden wir $-\infty < j\omega < \infty$ auf $-\pi \leq \Omega \leq \pi$ ab.

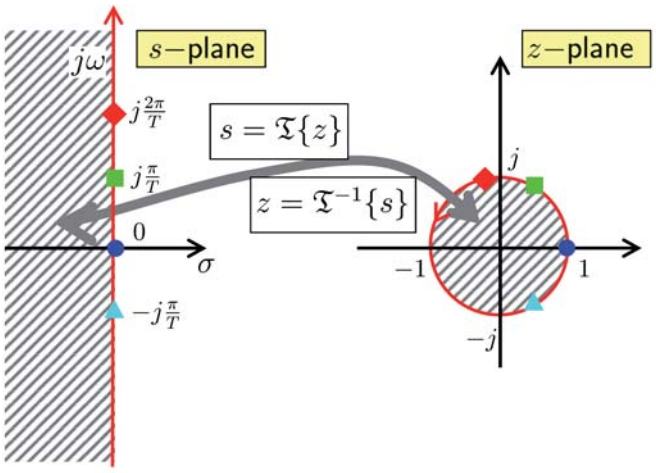


Abbildung 26: Die bilineare Transformation bildet die linke Halbebene der S-Ebene in den Einheitskreis der Z-Ebene ab. Aus [Fi-DSV].

S-Ebene	Z-Ebene
s = 0	z = +1
$j\omega \to \infty$	$z = -1$
$j\omega \to -\infty$	$z = -1$

Tabelle 1: Anforderungen an Transformation.

Tabelle 1 zeigt die Anforderungen, die die Transformation erfüllen muss. Es handelt sich um eine nichtlineare Abbildung, die somit zu einer Frequenzverzerrung (außer an der Stelle v, s.u.) führt! Eine Möglichkeit der Transformation ergibt sich mit

$$s = v \cdot \frac{z-1}{z+1} \quad \Longleftrightarrow \quad z = \frac{v+s}{v-s} \quad \text{, wobei } [v]_{SI} = \text{Hz} \ . \tag{47}$$

Daraus folgt unmittelbar die bilineare Transformation als

$$H(z) = H_a(s)|_{s=v\cdot\frac{z-1}{z+1}} \tag{48}$$

Da $z = e^{j\Omega}$ und $s = \sigma + j\omega$ ist, folgt

$$s = v \cdot \frac{e^{j\Omega}-1}{e^{j\Omega}+1} = v \cdot \frac{e^{\frac{j\Omega}{2}} \cdot \left(e^{\frac{j\Omega}{2}} - e^{-\frac{j\Omega}{2}}\right) \cdot 2}{e^{\frac{j\Omega}{2}} \cdot \left(e^{\frac{j\Omega}{2}} + e^{-\frac{j\Omega}{2}}\right) \cdot 2} = v \cdot \frac{j \cdot \sin\left(\frac{\Omega}{2}\right)}{\cos\left(\frac{\Omega}{2}\right)} = jv \cdot \tan\left(\frac{\Omega}{2}\right) \tag{49}$$

Setzt man nun $\sigma = 0$ (Übergang von Laplace zu Fourier), folgen die Transformationsvorschriften für die Frequenzachsen:

$$\begin{aligned} \omega &= v \cdot \tan\left(\frac{\Omega}{2}\right) \\ \Omega &= 2 \cdot \arctan\left(\frac{w}{v}\right) \end{aligned} \tag{50}$$

Neben der Stelle $\omega = 0$, welche auf $\Omega = 0$ abgebildet wird, erlaubt die freie Wahl des Parameters v exakt eine weitere Frequenz $\omega' \neq 0$, welche die Transformation eindeutig festlegt und bei der die Filterfunktionen übereinstimmen. Setze dazu

$$v = \frac{\omega'}{\tan\left(\frac{\Omega'}{2}\right)} \tag{51}$$

für genau eine beliebige Frequenz $\omega' \neq 0$. Ein Beispiel ist in Abbildung 27 zu sehen.

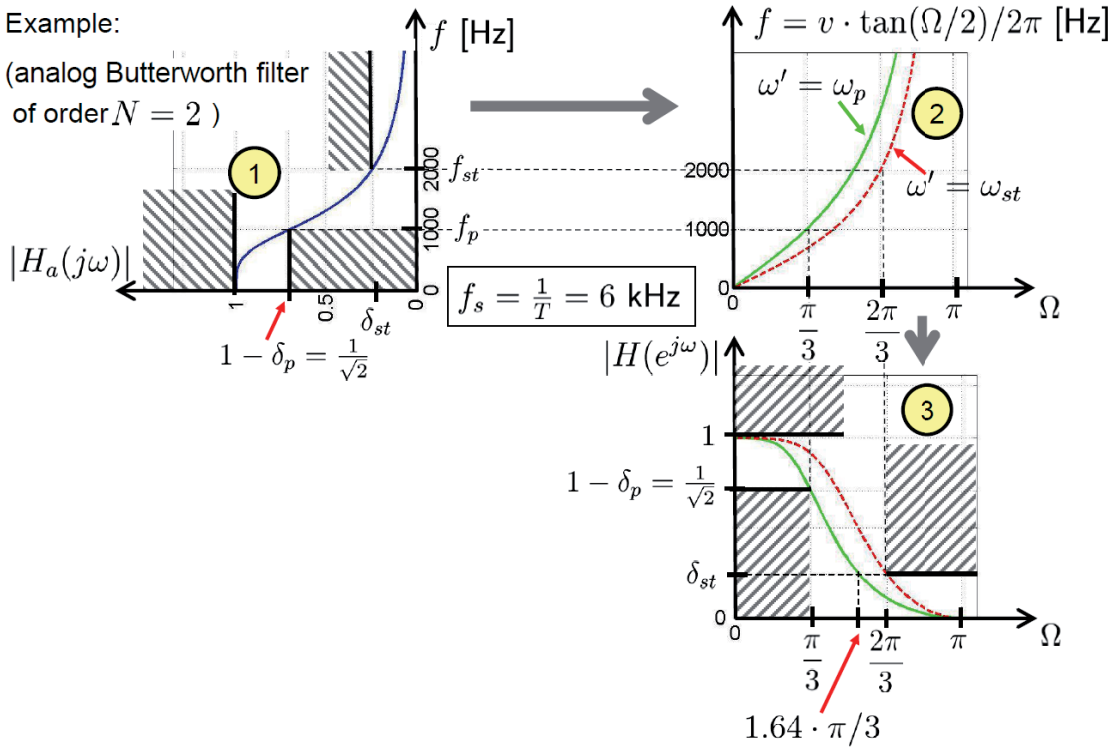


Abbildung 27: Beispiel zur bilinearen Transformation. Einmal mit $\omega' = \omega_p$ (grün) und einmal mit $\omega' = \omega_{st}$ (rot gestrichelt). Aus [Fi-DSV].

4.4 FIR-Entwurf

Auch bei den FIR-Filtern beginnt der Entwurf mit der Spezifikation. Durchlassfrequenz Ω_p und Sperrfrequenz Ω_{st} bleiben wie gehabt (siehe Beispiel in Grafik 28). Bei den dazugehörigen Gütemaßen ist die Definition der Sperrdämpfung ebenfalls analog zu der bei IIR-Filtern, die Welligkeit im Durchlassbereich jedoch ist anders definiert!

$$\begin{aligned} \text{Welligkeit im Durchlassbereich:} \quad & R_p = 20\ \log(1+\delta_p) - 20\ \log(1-\delta_p) \quad [dB] \\ \text{Sperrdämpfung:} \quad & d_{st} = -20\ \log(\delta_{st}) \quad [dB] \end{aligned} \tag{52}$$

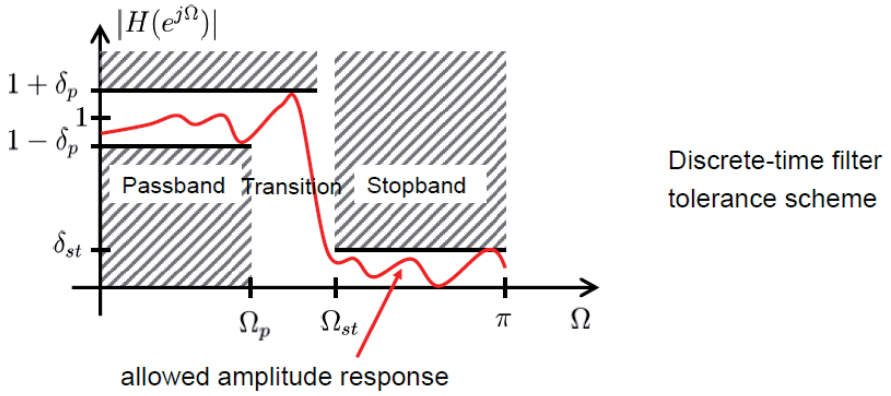


Abbildung 28: Toleranzschema und beispielhafter Amplitudenverlauf zum FIR-Filterentwurf. Aus [Fi-DSV].

FIR-Filter können **linearphasig** sein. Das bedeutet dann, dass die Gruppenlaufzeit $\tau = -\frac{d\phi(\omega)}{d\omega}$ konstant ist. Es treten somit keine nicht-linearen Phasenänderungen (Verzerrungen) für einzelne Frequenzen auf, was besonders für im Audiobereich sehr interessant ist.

Modifizierte Fourier-Approximation

Um einen FIR-Filter zu entwerfen, entwickeln wir diesen, wie den IIR-Filter, im Frequenzbereich. Dabei ergibt sich allerdings eine unendlich ausgedehnte Impulsantwort. Der gewünschten (unendlichen) Impulsantwort $h_{ideal}(e^{j\Omega})$ nähert man sich nun an, indem man eine endliche Impulsantwort (FIR) $h(n)$ sucht, welche den Fehler zur idealen Impulsantwort minimiert. Dazu wird eine **Fensterfunktionen** $w(n)$ verwendet, um aus der unendlichen Impulsantwort eine endliche zu machen.

$$\begin{aligned} h(n) &= h_{ideal}(n) \cdot w(n) \\ &\updownarrow \\ H(e^{j\Omega}) &= H_{ideal} \star W(e^{j\Omega}) \end{aligned} \tag{53}$$

Zu beachten ist, dass sich die Fensterfunktion auf den Frequenzgang eines Filters auswirkt. Die Auswirkungen sind dabei je nach verwendetem Fenster unterschiedlich. Die einfachste und wohl bekannteste Fensterfunktion ist die Rechteckfunktion. Da das Multiplizieren mit dem Rechteck einer Faltung mit der SI-Funktion im Frequenzbereich entspricht, erkennt man recht schnell, dass hierbei Verzerrungen entstehen. Insbesondere ist die Sperrdämpfung nahe des Durchlassbereichs für viele Anwendungen zu gering. In der Realität werden Fensterfunktionen mit geringere Flankensteilheit genutzt, welche eine bessere Sperrdämpfung aufweisen. Die Geläufigsten sind das *Hann*-, *Hamming*-, *Blackman*- und *Kaiser*-Fenster. Dabei erlaubt das Kaiser-Fenster das Einstellen der Sperrdämpfung durch den Parameter β.

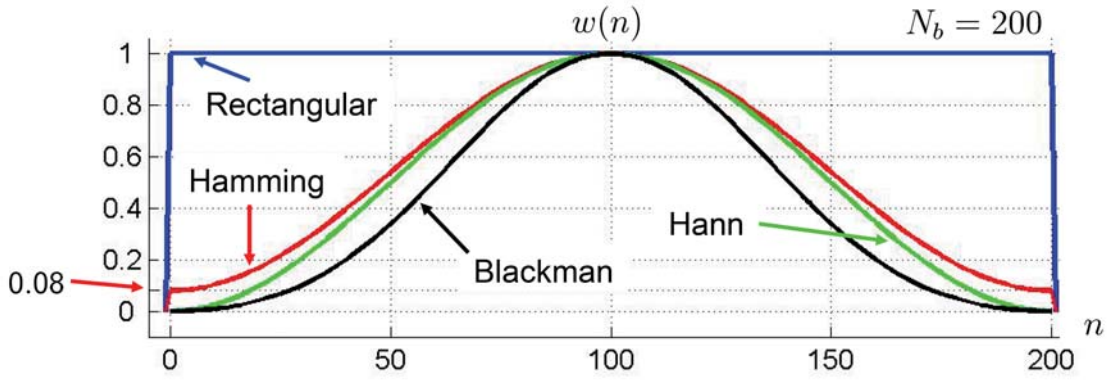


Abbildung 29: Geläufige Fensterfunktionen in der digitale Signalverarbeitung. Aus [Fi-DSV].

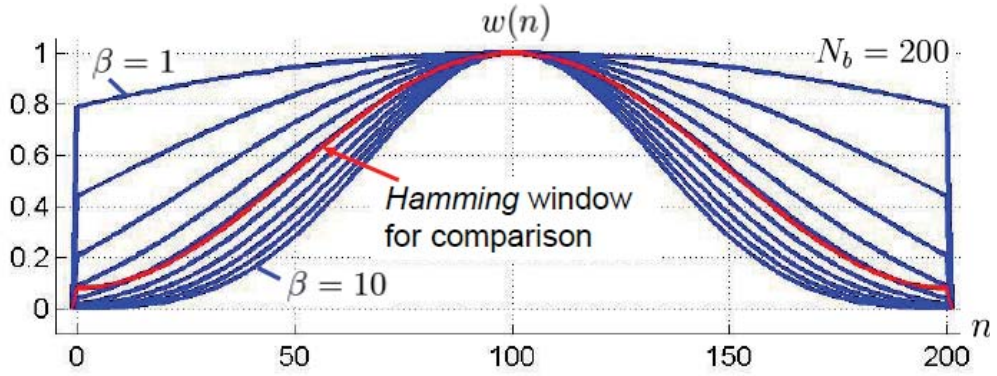
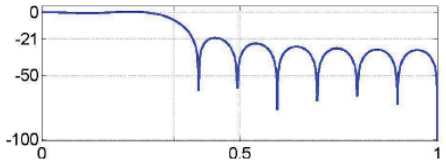
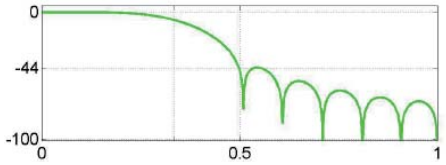
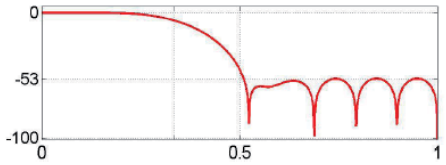
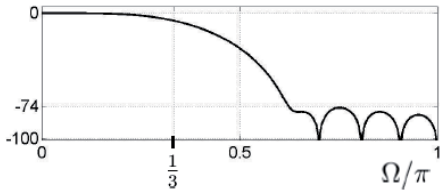
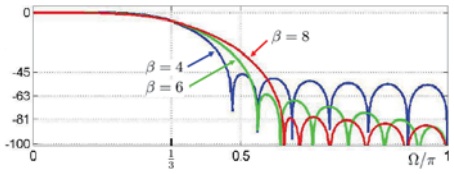


Abbildung 30: Das Kaiser-Fenster. Aus [Fi-DSV].

Fenster	Formel (n = 0, 1, ..., N_b)	Verlauf der Filterdämpfung in dB ($N_b = 19$, $N = 20$, $\Omega_c = \frac{\pi}{3}$)
Rechteck	$\omega(n) = 1$	
Hann	$\omega(n) = 0.50 - 0.50 \cdot cos(\frac{2\pi n}{N_b})$	
Hamming	$\omega(n) = 0.54 - 0.46 \cdot cos(\frac{2\pi n}{N_b})$	
Blackman	$\omega(n) = 0.42 - 0.50 \cdot cos(\frac{2\pi n}{N_b}) + 0.08 \cdot cos(\frac{4\pi n}{N_b})$	
Kaiser	$\omega(n) = \frac{I_0(\beta\sqrt{1-(1-\frac{2}{N_b}n)^2})}{I_0(\beta)}$	

Anmerkung zum Kaiser-Fenster
Folgende Zusammenhänge sollten beachtet werden. Es sei

$$d = -20\log(\min\{\delta_p, \delta_{st}\}) \text{ [dB]} \tag{54}$$

und

$$\Delta\Omega = \Omega_{st} - \Omega_p \tag{55}$$

Dann ist die Filterordnung mit

$$N_b \geq \frac{\frac{d}{dB} - 7.95}{2.29 \cdot \Delta\Omega} \tag{56}$$

und β mit

$$\beta = \begin{cases} 0 & , d < 21 \text{ dB} \\ 0.5842 \cdot (\frac{d}{dB} - 21)^{0.4} + 0.07886 \cdot (\frac{d}{dB} - 21) & , 21 \text{ dB} \leq d \leq 50 \text{ dB} \\ 0.1102 \cdot (\frac{d}{dB} - 8.7) & , d > 50 \text{ dB} \end{cases} \tag{57}$$

zu wählen.

4.5 Zusammenfassung: Digitale Filter

Die Zusammenfassung der IIR- und FIR-Filter als Vergleich bei gleicher Spezifikation.

IIR-Filter		FIR-Filter
• niedrigere Ordnung möglich für gleiche Frequenzgangkriterien		• immer stabil
• allgemein keine lineare Phase	$\Longleftrightarrow$	• linearphasiger Entwurf möglich
• kleinere Gruppenlaufzeit, aber frequenzabhängig	$\Longleftrightarrow$	• frequenzunabhängige (dafür aber größere) Gruppenlaufzeit
• weniger Rechenleistung und Speicher nötig	$\Longleftrightarrow$	• Rechen- und Speicheraufwand können sehr groß sein
• empfindlich für Rundungsfehler da begrenzte Rechengenauigkeit	$\Longleftrightarrow$	• nicht so anfällig für begrenzte Rechengenauigkeit

5 Die Praktikumsumgebung

Im Verlauf des Praktikums kommen wir mit drei verschiedenen Programmen bzw. Geräten in Berührung. Zum einen wird *Matlab* für den Filterentwurf und für die Anwendung des Filters auf Audiodateien genutzt. Um eine Echtzeit-Anwendung zu implementieren arbeiten wir mit einem digitalen Signalprozessor von *Analog Devices*, welcher in ein dazugehöriges Entwicklungsboard integriert ist. Parallel wird uns *Adobe Audition* behilflich sein, Audiosignale wiederzugeben und deren Spektren zu betrachten.

5.1 Entwicklungsboard und DSP

Das Herz des Praktikums bildet der digitale Signalprozessor (DSP) „ADSP-21489“ der Sharc-Serie von Analog Devices. Er ist mit 400 MHz getaktet, bietet 5 Mbits internen SRAM und ist für Hochgeschwindigkeitsberechnungen im Bereich von Audioanwendungen ausgelegt.

(a) DSP-Chip (b) Entwicklungsboard (c) Audio Interface

Abbildung 31: Verwendetes Hardware Equipment

Der Prozessor ist in das oben dargestellte Entwicklungsboard EZ-Kit Lite integriert, welches viele Funktionen bietet, die für die Audioverarbeitung sehr nützlich sind. In diesem Praktikum sind hiervon erst einmal nur einige wenige Funktionen wichtig.

Das Board besitzt vier Cinch-Buchsen als Schnittstelle für eingehende analoge Audiosignale und acht Buchsen für die Ausgangssignale. Für die Analog/Digital-Wandlung (bzw. Digital/Analog-Wandlung) der Signale ist der Audio-Coder-Decoder AD1939 zuständig. Zwischen Audio-Codec und DSP werden die digitalen Signale direkt übertragen. Des Weiteren sind in Abbildung 31b unten links acht kleine weiße Kästchen zu sehen. Dies sind einzeln ansteuerbare LEDs. Direkt über den LEDs sind vier Taster integriert. Je nach Programmierung kann man über sie zum Beispiel zwischen den Eingangskanälen wechseln oder andere Funktionen ein- bzw. ausschalten. Letztlich befindet sich unter dem Board ein sogenannter Debug Agent. Über ihn wird das Board via USB an den Computer angeschlossen, um es direkt aus der Entwicklungsumgebung zu programmieren.

5.2 Messaufbau

Die Kommunikation der Audiosignale zwischen DSP und PC geschieht über ein externes Audio Interface. Es stellt in beide Richtungen die A/D- bzw. D/A Wandlung zur Verfügung und ermöglicht dadurch, dass Audiosignale vom PC an den DSP geleitet und vom DSP bearbeitete Signale am PC wieder aufgenommen werden können. Auch Instrumente müssen zunächst zur Vorverstärkung (und teilweise zur Digitalisierung) in das Audio Interface geführt werden.

Als Audio Interface nutzen wir hier eine **Fireface 400 von RME**. Durch ihren Firewire Anschluss wird eine Datenrate erreicht, die 196 kHz Abtastfrequenz und 24 Bit Worttiefe für die Audiokanäle erlaubt. Es stehen acht Ein- und Ausgänge und zwei Mikrofoneingänge mit Vorverstärkung und einzeln schaltbarer 48 V Phantomspeisung zur Verfügung. Bei den Ein- und Ausgängen wird das Signal meist symmetrisch Übertragen. Statt wie bei der unsymmetrischen Übertragung nur einen einzelnen Leiter und als Gegenpol die Masse zu nutzen, wird bei der symmetrischen Übertragung auf einem weiteren Leiter das Nutzsignal invers angelegt. Der Empfänger bildet dann die Differenz der beiden Signale und erhält wieder das eigentliche Signal. Da sich leitungsinduzierte Gleichtaktstörungen auf beiden Leitern nahezu gleich einkoppeln, wird der Großteil der Störungen durch die Differenzbildung bei der symmetrischen Übertragung wieder entfernt.

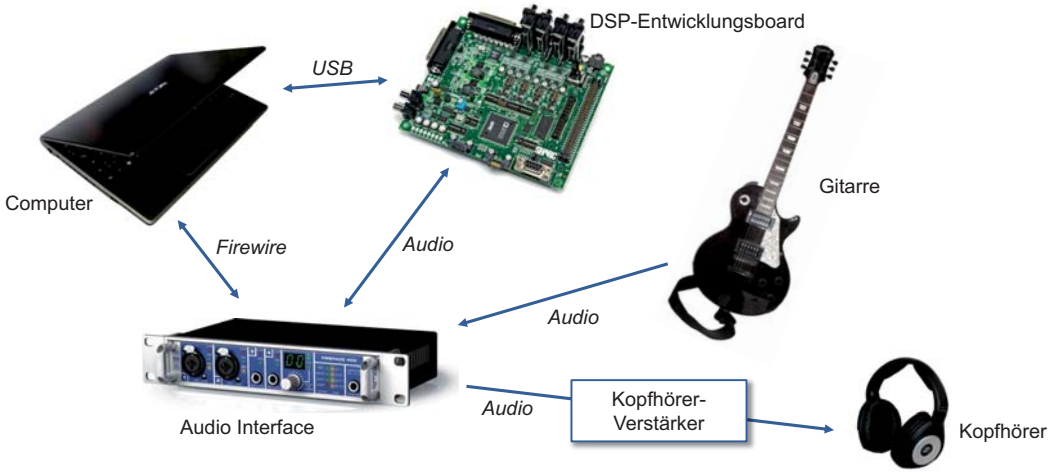


Abbildung 32: Aufbau und Signalfluss. (Quelle Gitarrenbild: http://upload.wikimedia.org/wikipedia/commons/5/57/E-Guitare-horiz.png)

5.3 Matlab und Signal Processing Toolbox

Als Software für den Filterentwurf nutzen wir Matlab. Alle oben vorgestellten Verfahren und Filter sind hier komfortabel zu nutzen. Die Signal Processing Toolbox (SPT) in Matlab erlaubt die Angabe von Entwurfsrichtlinien (z.B. Ordnung, Dämpfungen usw.) und berechnet Filter nach der ausgesuchten Entwurfsmethode ($\rightarrow$ Befehl *fdatool* (filter design and analysis tool)). Mit diesem Werkzeug kann man Filterkoeffizienten bestimmen und Pol-Nullstellen Diagramme, Frequenzgang, Phasengang etc. analysieren.

6 Versuch I - Störgeräusch

Ein Journalist hat den Gesprächsverlauf einer Telefonkonferenz mitgeschnitten, um später anhand der Aufzeichnung seinen Artikel zu schreiben. Als er sich abends seinem Artikel und auch der Aufnahme zuwendet, stellt er schockiert fest, dass sich ein sehr unangenehmes Störgeräusch mit eingeschlichen hat. Verzweifelt schickt er Ihnen einen knappen Ausschnitt der Aufnahme und bittet Sie um Hilfe.

Hören Sie sich die zugesendete Datei „Snippet.wav“ in Adobe Audition an. Nutzen Sie in Adobe Audition auch die Ansicht des „Spectral Frequency Displays“ (unterhalb der „Waveform“-Anzeige). Ermitteln Sie zunächst einige grundlegende Eckdaten:

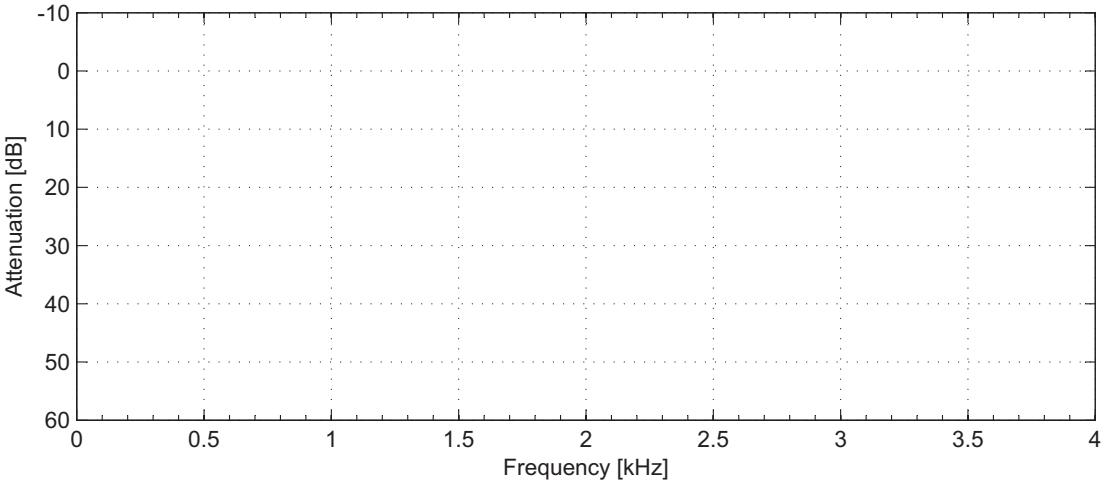
Abtastfrequenz _____

höchste darstellbare Frequenz _____

Störgeräusch bei ca. (Hz) _____

Breite des Störgeräuschs ca. (Hz) _____

Machen Sie sich nun Gedanken, welche Art von Filter Sie nutzen können, um das Störgeräusch zu entfernen. Tiefpass, Hochpass, Bandpass oder Bandsperre? Skizzieren Sie das Filter in das Diagramm. Im Sperrbereich soll die Amplitude um 60 dB gedämpft sein. Kennzeichnen Sie auch Durchlass- und Sperrfrequenzen.



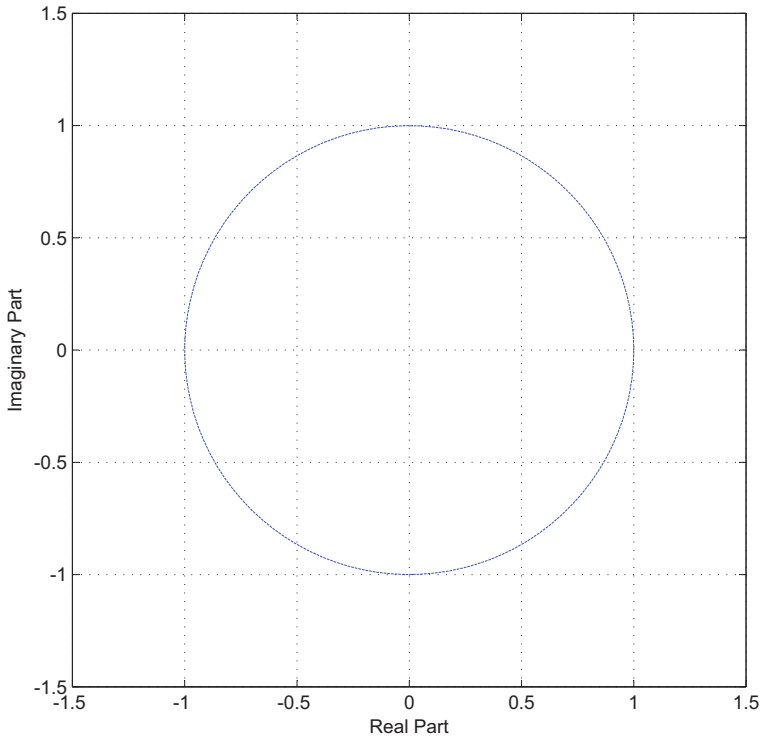
Sind Durchlass- und Sperrfrequenzen dieselben? Begründen Sie.

_____

_____

Öffnen Sie nun das *filter design and analysis tool* in Matlab, durch Tippen des Befehls **fdatool** in der Konsole. Entwerfen Sie ein IIR-Filter zweiter Ordnung. Orientieren Sie sich an den von Ihnen ermittelten Grenzfrequenzen und sorgen Sie dafür, dass das Filter ein flaches Amplitudenspektrum im Durchlassbereich hat. Welches Filter haben Sie gewählt? Warum?

Betrachten Sie das entworfene Filter in der Pol-Nullstellen-Darstellung. Wechseln Sie die obere Ansicht auf „Magnitude and Phase Responses“. Übertragen Sie die Pol- und Nullstellen in das untenstehende Diagramm.



Ist das Filter linearphasig?

Verschieben Sie nun die Pol- und Nullstellen manuell. Wie verändert sich dadurch der Amplitudenverlauf des Filters?

Wann wird das Filter instabil?

Kehren Sie zur „Design Filter“-Ansicht des *filter design and analysis tools* zurück und generieren das ursprüngliche Filter nochmals. Über **File** → **Generate Matlab Code** → **Filter Design Function** können Sie eine Funktion erzeugen, die ihnen den weiteren Zugriff auf das Filter ermöglicht.

Schreiben Sie eine Funktion, die Ihr Filter auf „Snippet.wav“ anwendet und als Ausgabe eine neue „.wav“-Datei erzeugt, die Sie dem Journalisten zusenden können. Hören und sehen Sie sich das Ergebnis in Adobe Audition an. Ist die Filterordnung ausreichend? Nützliche Befehle: **audioread**, **audiowrite**, **filter**.

Generieren Sie nun ein IIR-Filter (mit flachem Amplitudenspektrum) und ein FIR-Filter (gleichmäßige Verteilung des Näherungsfehlers auf Durchlass- und Sperrband) jeweils mit „minimum order“ aber ansonsten gleichen Eigenschaften. Wenden Sie auch diese Filter auf die Sprachdatei an und nennen Sie wichtige Unterschiede zwischen den beiden Filtern.

7 Versuch II - Faltungshall

In diesem Versuch soll ein Faltungshall auf dem DSP implementiert werden. Als Eingangssignal wird eine Gitarre angeschlossen. Der DSP fügt dem Gitarrensignal Hall hinzu und gibt es anschließend wieder an die Fireface zurück.

Der Faltungshall ist ein beliebter Effekt in der Musikproduktion. Einem trockenen Schallsignal wird, durch Faltung mit der Impulsantwort eines halligen Raumes, ein künstlicher Räumlichkeitseindruck verliehen. Die Impulsantwort beschreibt dabei die statische Akustik an einer Position des Raums. Sie lässt sich ermitteln, indem man z.B. einen „Sweep“ (also einen „Durchlauf“) über die hörbaren Frequenzen (etwa 20 Hz bis 20 kHz) mit einem Lautsprecher wiedergibt und gleichzeitig die Ausgabe mit einem Mikrophon an anderer Stelle im Raum aufnimmt. Errechnet man aus dem „Eingangssignal“ (in den Raum) und dem „Ausgangssignal“ (aus dem Raum, d.h. ins Mikrophon) nun die Übertragungsfunktion, so entspricht sie der Raumimpulsantwort. Zu beachten ist, dass die Raumimpulsantwort von Wiedergabe- und Aufnahmeposition abhängt. Durch Veränderung der Positionen ergeben sich also theoretisch unendlich viele Systeme für einen Raum.

Anhand der Impulsantwort eines Raumes kann man sich eine Vorstellung des Raumes machen. Hilfreich ist hierbei zum Beispiel die Sabinesche Nachhallzeit T_{60}. Dies ist die Zeit, nach der die anfängliche Energie der Impulsantwort um 60 dB abgefallen ist und anhand derer man die Größe des Raumes abschätzen kann. Weiterhin lässt sich die Raumimpulsantwort, wie in Abbildung 33 dargestellt, zeitlich in drei Phasen unterscheiden: den Direktschall, die frühen Reflexionen und den diffusen Nachhall.

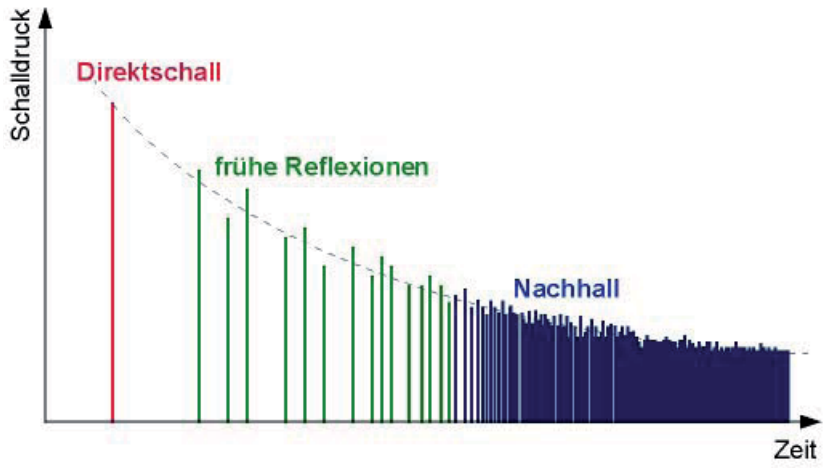


Abbildung 33: Aufbau einer Raumimpulsantwort

Hören Sie sich zunächst in **Adobe Audition** die bereits aufgenommene Raumimpulsantwort „RaumImpulsAntwort_-3dB.wav“ an. Schätzen sie mit der „Waveform“-Anzeige

grob die Nachhallzeit ab. Wie lang ist T_{60} etwa? Würden Sie diese Raumimpulsantwort intuitiv eher einer Kirche, einem Hörsaal oder einem kleineren Konferenzraum zuordnen? Begründen Sie.

Aus der Raumimpulsantwort soll nun mit Matlab ein FIR-Filter für den DSP erzeugt werden. Hierfür sind einige Arbeitsschritte notwendig, die nacheinander durchlaufen werden:

1. Die Filterung auf dem DSP soll in Echtzeit geschehen. Das bedeutet hier, dass jedes einzelne Eingangssample gefiltert wird bevor es wieder ausgegeben wird. Um dabei die Rechenkapazität des DSPs nicht zu überschreiten, begrenzen wir die Länge der Impulsantwort und damit auch die Ordnung des Filters. Bei einer Abtastrate von 48 kHz wollen wir nur die erste Achtelsekunde der Raumimpulsantwort für den Filter nutzen. Wieviele Samples müssen verwendet werden?

_____ Samples

2. Die soeben bestimmte Zahl der Samples entspricht genau der Anzahl an sogenannten Taps (den Verzögerungs- bzw. Speicherelementen) des FIR-Filters. Ein FIR-Filter der Ordnung N hat $N+1$ Taps. Der große Vorteil ist nun: die Sample-Werte der .wav-Datei können direkt als Filterkoeffizienten verwendet werden. Sie entsprechen den Koeffizienten des Terms $B(z)$ aus Kapitel 3.2.3.

```
-1.919555664062500e-02,
-2.157592773437500e-02,
-2.178955078125000e-02,
...
```

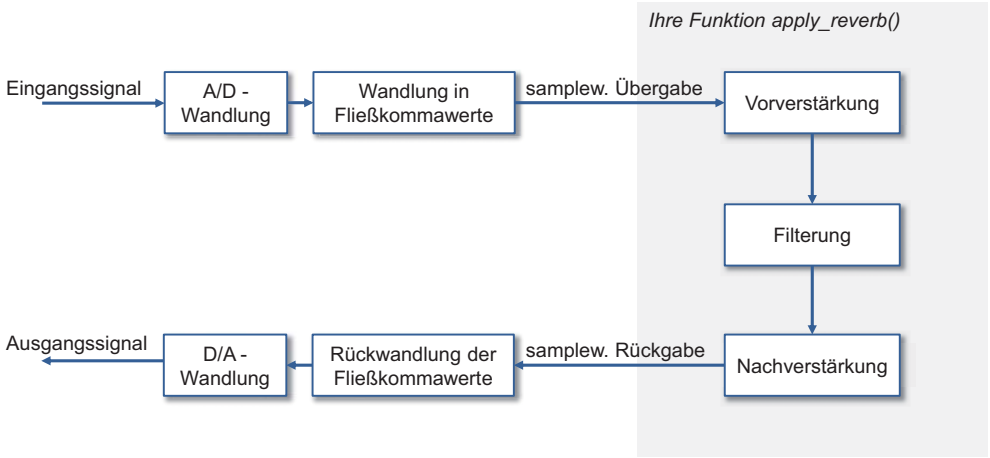
Tabelle 2: Ausschnitt der Koeffizientendatei „celldata.dat“.

Mit einem Matlab-Skript sollen die Filterkoeffizienten in eine eigene Datei geschrieben werden. Öffnen Sie das Skript „FIR_Filter_from_WAV_studs.m“ und vervollständigen Sie es an den gekennzeichneten Stellen. Die Filterkoeffizienten müssen in umgekehrter Reihenfolge geschrieben werden, damit sie vom DSP richtig ausgelesen werden. Mit „F5“ können Sie das Skript dann ausführen. Wenn alles richtig ist, sollte sich in dem Ordner nun die Datei „celldata.dat“ befinden und wie in Tabelle 2 aussehen.

3. Öffnen Sie die Datei „celldata.dat“ mit **Notepad++** und speichern Sie sie erneut unter dem Namen „ImpulseResponse_FIR_coeffs.h“. Schieben Sie die neue Datei in den Ordner des DSP-Projekts. Die Filterkoeffizienten stehen dem DSP-Programm nun zur Verfügung.

4. Für die Implementierung auf dem DSP ist bereits ein Framework in der Programmiersprache C für Sie vorbereitet. Es liefert Ihnen Sample für Sample die Eingangssignalwerte zur weiteren Verarbeitung. Die Eingangswerte werden als Fließkommawerte im Wertebereich zwischen -1.0 und 1.0 dargestellt, wobei -1.0 bzw. 1.0 eine Vollaussteuerung bedeutet. **Beachten Sie im folgenden unbedingt den in Kapitel 9 beschriebenen Versuchsaufbau!**

5. Öffnen Sie das **Visual DSP++**-Projekt „DAFX – Reverb – Sample-based“. Es beinhaltet das vorbereitete Framework. In der Datei **Reverb.c** soll in der Funktion **apply_reverb()** die Filterung realisiert werden. Ihre Funktion soll sich an dem abgebildeten Signalflussdiagramm und den nachstehenden Erklärungen orientieren.



Der Funktion **apply_reverb()** wird mit jedem Aufruf ein Sample **audio_data** übergeben. In dem Schritt der Vorverstärkung soll der Pegel gesenkt werden, um während der Berechnungen innerhalb des gültigen Wertebereichs zu bleiben. Der Wert von **audio_data** soll um 30 dB gesenkt werden. Mit Hilfe der Tabelle 3 entspricht dies einem Faktor von

$$\text{pre\_gain} = \frac{}{} .$$

Zum Filtern des Signals wird die Funktion **fir()** aus den Bibliotheken des DSPs verwendet. Übergeben werden müssen dieser Funktion ein Eingangssample, die Filterkoeffizienten als Array von Fließkommawerten, ein Zustandsarray um Werte überlagen zu können und die Anzahl der Filtertaps. Mit den im Framework verwendeten Variablennamen muss der Aufruf wie folgt lauten:

audio_data = fir(audio_data, coeffs, state, TAPS);

Die Nachverstärkung soll den Pegel wieder grob an den des Eingangssignals anpassen. Dafür soll der Pegel um 20 dB angehoben werden, dies entspricht einem Faktor von

$$\text{post\_gain} = \text{\_\_\_\_\_} .$$

Leistungsgrößen P_1/P_2	Feldgrößen F_1/F_2	Dezibel
100	10	20 dB
10	3.16	10 dB
4	2	6 dB
2	1.41	3 dB
1	1	0 dB

Tabelle 3: Hilfstabelle für Dezibelumrechnungen.

6. Ist Ihre Funktion fertig geschrieben und VisualDSP++ mit dem ADSP-21489 verbunden (wie im Aufbau beschrieben), können Sie durch Drücken von F7 das Programm kompilieren und auf das DSP-Board laden. Mit F5 wird das Programm gestartet und läuft bis Sie es mit Shift+F5 anhalten. Um es erneut zu aktivieren sollten Sie vorher auch immer neu kompilieren (wieder mit F7).

WICHTIG: Das Framework ist so ausgelegt, dass man zwischen verhalltem und klarem Sound hin- und herwechseln kann. Leuchtet keine der LEDs auf dem Board, wird das Signal unverändert an den Ausgang gegeben (klarer Gitarrensound). Erst durch Drücken des ersten Pushbuttons "SW8" wird Ihre Funktion „**apply_reverb**" auf das Signal angewendet und es leuchten die mittleren beiden LEDs. Nochmaliges Drücken schaltet den Effekt wieder aus.

Testen Sie Ihr Programm mit der Gitarre! Vergleichen Sie auch das klare und das bearbeitete Signal miteinander! Zeigen Sie das Ergebnis dem Praktikumsbetreuer.

7. Nun soll das Ergebnis mit Adobe Audition betrachtet werden. Öffnen Sie **Adobe Audition** und klicken Sie auf **Multitrack**. Erstellen Sie eine neue Session mit 48000 Hz Abtastrate. Ziehen Sie die Datei „dry_guitar_normalized_to_-18dB.wav" in die erste Spur und schieben Sie sie an den Anfang der Spur.

Klicken Sie auf das „R" links neben der **zweiten** Spur, dadurch wird die Spur für die Aufnahme ausgewählt, das „R" sollte nun rot hinterlegt sein. Wählen Sie als Input für die zweite Spur „[01M] Analog (5+6) (2- RME Fireface 400) 1" aus. Stellen Sie die Fireface Matrix, wie im Aufbau beschrieben, für das Wiedergeben und Aufnehmen in Audition ein.

Schalten Sie den Halleffekt auf dem DSP ein und drücken Sie dann in Audition „Record" (unten). Drücken Sie „Stop", wenn die Datei vollständig abgespielt wurde.

8. In der zweiten Spur ist das verhallte Signal aufgenommen worden. Betrachten Sie beide Signale im Frequenzbereich (Doppelklick auf die jeweilige Spur), zoomen Sie auch näher an bestimmte Abschnitte ran. Beschreiben Sie die Unterschiede und begründen Sie.

8 Versuch III - Simulation analoger Audiohardware

Die moderne Rockmusik ist ohne eine ordentlich verzerrte Gitarre nicht denkbar. Leider ist einem befreundeten Gitarristen sein analoges Effektgerät geklaut worden – und das, obwohl er doch morgen einen Auftritt hat. „Das große T“ (ein bekannter Online-Versand für Musikprodukte) kann das Effektgerät so kurzfristig nicht mehr liefern. Helfen Sie Ihrem Freund, indem Sie die analoge Audiohardware auf dem DSP nachbilden.

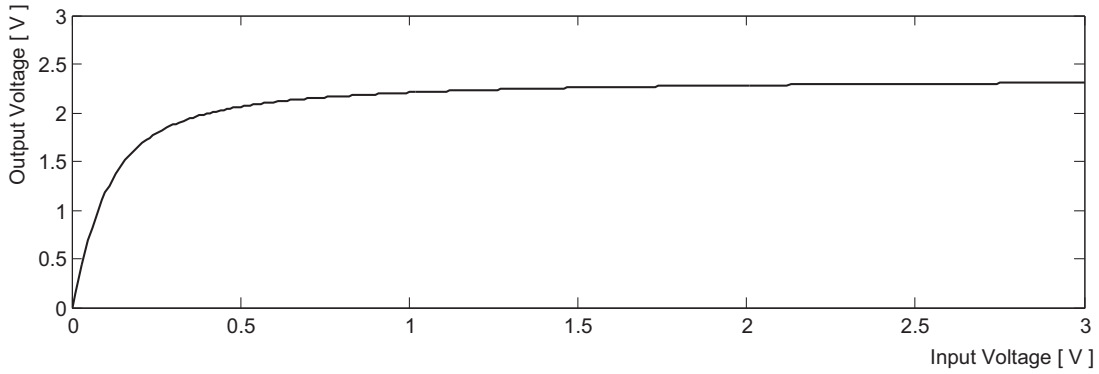
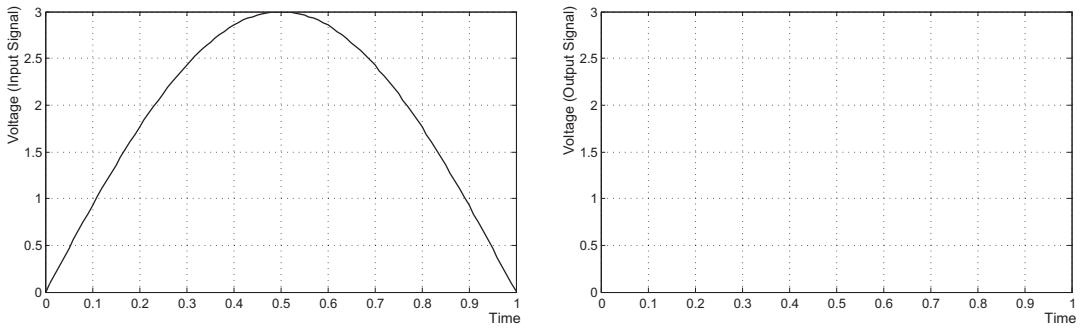


Abbildung 34: Gemessene Kennlinie des Effektgerätes.

Zunächst wollen Sie die Funktionsweise des Verzerrungs-Effektgeräts verstehen. Betrachten Sie dazu die oben gezeigte Kennlinie des Effektgeräts und vervollständigen Sie das folgende Diagramm: das linke Signal wird als Eingangssignal an das Effektgeräts gegeben. Zeichnen Sie rechts das dazugehörende Ausgangssignal ein.



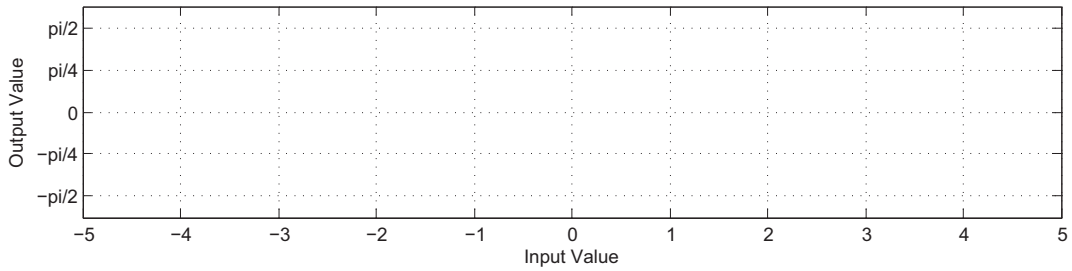
Wodurch kommt es zur Verzerrung des Eingangssignals?

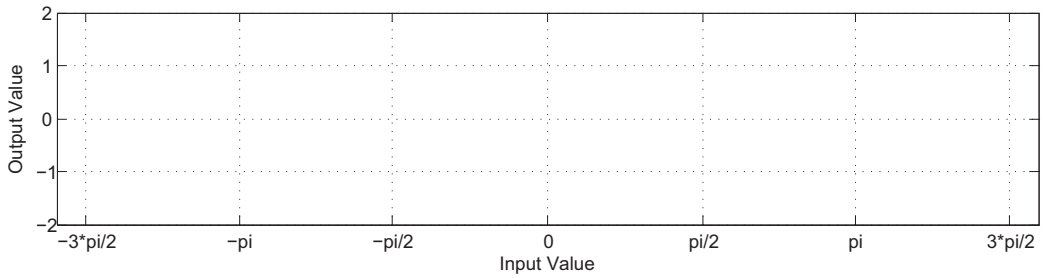
Die Kennlinie des Effektgeräts ist nur für positive Aussteuerungen gezeigt. Audiosignale haben jedoch immer positive, wie auch negative Aussteuerungen; man denke z.B. an ein einfaches Sinussignal. Gehen Sie davon aus, dass sich das Effektgerät für negative Aussteuerung in gleicher Weise verhält.

Bei den Überlegungen zu Ihrer eigenen Implementierung wollen Sie statt eines klassischen Filterentwurfs die nichtlineare Kennlinie nachbilden. Dazu sollen die einzelnen Eingangswerte durch eine mathematische Funktion auf die entsprechenden Ausgangswerte abgebildet werden.

Wählen Sie aus den folgenden sechs Funktionen jene aus, welche sich zum Nachbilden der Kennlinie am besten eignet. Skizzieren Sie die Funktion in **eines** der beiden Diagramme; wählen Sie das sinnvollere!

1. **Sinus** 2. **Cosinus** 3. **Tangens** 4. **Arcussinus** 5. **Arcuscosinus** 6. **Arcustangens**

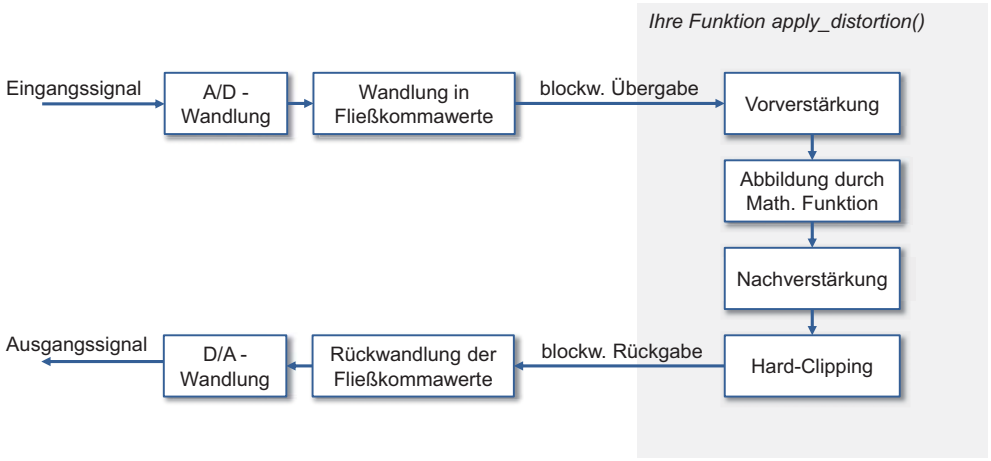




Für welche Funktion haben Sie sich entschieden? Warum?

Auf dem DSP steht Ihnen für Ihre Implementierung ein bereits vorbereitetes Framework zur Verfügung. Es liefert Ihnen blockweise die Eingangssignalwerte. Blockweise bedeutet, dass das Framework immer 256 Eingangswerte „aufsammelt“ und diese als Block in einem Array zur Bearbeitung bereitstellt. Die Eingangswerte werden als Fließkommawerte im Wertebereich zwischen -1.0 und 1.0 dargestellt, wobei -1.0 bzw. 1.0 eine Vollaussteuerung bedeutet.

Öffnen Sie das **Visual DSP++**-Projekt „DAFX – Distortion“. Es beinhaltet das vorbereitete Framework. Schreiben Sie in der Datei **Distortion.c** die Funktion **apply_distortion()**. Ihre Funktion soll sich an dem abgebildeten Signalflussdiagramm und den untenstehenden Erklärungen orientieren.



Vorverstärkung: Ein konstanter Faktor c. Soll genutzt werden, um die Eingangswerte in den gewünschten Wertebereich der mathematischen Funktion zu bringen. Sinnvolle Werte hierfür können Sie z.B. anhand ihrer Skizze der mathematischen Funktion ermitteln. Welcher Wert hat hier gute Ergebnisse erzielt?

$$c = \_\_\_\_\_$$

Abbildung Math. Funktion: nützliche Befehle sind hier die Fließkommaimplementierungen der math. Funktionen. **sinf()**, **cosf()**, **tanf()**, **asinf()**, **acosf()**, **atanf()**.

Nachverstärkung: Ein konstanter Faktor, der sich aus zwei Werten (d und e) zusammensetzt: zum einen sollen die Ausgangswerte der mathematischen Funktion wieder in den Wertebereich von -1.0 bis 1.0 gebracht werden. Welcher Faktor muss dafür gewählt werden?

d = _____

Zum anderen muss die Lautstärke des Ausgangssignals wieder grob an die des Eingangssignals angepasst werden. Dazu soll der Pegel um **ca. 14 dB** gesenkt werden. Welcher Faktor muss hierfür gewählt werden?

e = _____

Hard-Clipping: Soll den Wertebereich des Ausgangssignals auf -0.90 bis 0.99 beschränken. Durch dieses asymmetrische „Hard-Clipping“ wird das Signal zusätzlich verzerrt.

WICHTIG: Das Framework ist so ausgelegt, dass man zwischen Verzerrung und klarem Sound hin- und herwechseln kann. Leuchtet keine der LEDs auf dem Board, wird das Signal unverändert an den Ausgang gegeben (klarer Gitarrensound). Erst durch Drücken des ersten Pushbuttons „SW8“ wird Ihre Funktion „**apply_distortion**“ auf das Signal angewendet und es leuchten die mittleren beiden LEDs. Nochmaliges Drücken schaltet die Verzerrung wieder aus.

Testen Sie Ihren Verzerrungseffekt mit der Gitarre, wie Sie es in Versuch II getan haben!

Zuletzt soll das Ergebnis in Audition betrachtet werden. Spielen Sie dazu die Datei „dry_guitar_normalized_to_-3dB.wav“ in Audition ab und nehmen Sie das verzerrte Signal wieder auf. Der Ablauf ist derselbe wie in Versuch II Schritt 7, nur mit anderer Sound-Datei.

Vergleichen Sie das klare und das verzerrte Gitarrensignal im Zeit- sowie im Frequenzbereich. Beschreiben Sie die Unterschiede.

9 Aufbau: Versuch II und Versuch III

Genutzt werden:

- PC mit Audobe Audition, Matlab und Visual DSP++
- Gitarre
- ADSP-21489 Evaluation Board
- Fireface 400

1. Fireface

- Die Fireface wird über Firewire mit dem PC verbunden und extern mit Strom versorgt.
- Am PC wird über das orangene **Fireface Settings** Symbol in der Taskleiste eine Einstellung vorgenommen: **Rechtsklick** → **Settings** → **Reiter „Analog“**. Bei **Instrument/Line Inputs** für **3** den Haken auf **Inst** setzen (dadurch wird die Eingangsimpedanz der Fireface für die Gitarre hochgesetzt).

2. DSP-Board

- Das DSP-Board wird über USB mit dem PC verbunden und extern mit Strom versorgt.
- In **Visual DSP++** wird die Verbindung zum Board hergestellt: „**Connect to Target**“ oben links.

3. Adobe Audition

- Über **Edit** → **Preferences** → **Audio Hardware** wird
  - **Analog (5+6)** als **Default Input**
  - **Lautsprecher** als **Default Output** gewählt.
- Über **File** → **New** → **Multitrack Session** wird eine mehrspurige Session geöffnet. Eine Spur wird für das klare Gitarrensignal genutzt, eine für das bearbeitete.

4. **Verkabelung**:

- Die Gitarre wird über ein Instrumentenkabel mit **Input 3** (Vorderseite) der Fireface verbunden.
- **Balanced Line Output 5 + 6** der Fireface (Rückseite) werden mit **Input 1** des DSP-Boards (links unten, 1. und 2. Cinch-Buchse) verbunden.
- **Output 1** des DSP-Boards (links mitte, 1. und 2. Cinch-Buchse) werden mit **Balanced Line Input 5 + 6** der Fireface verbunden (Rückseite).
- Der Kopfhörer wird an **Monitor Ch. 7/8** der Fireface (Vorderseite) angeschlossen.

5. **Fireface Matrix**

- **Linksklick** auf das **TotalMix FX** Symbol unten rechts in der Taskleiste:
- Zum Spielen und Aufnehmen der Gitarre wird die Matrix wie in Bild 35 eingestellt.
- Zum Abspielen in Audition, Weitergeben an das DSP und dann wieder Aufnehmen in Audition wird die Matrix wie in Bild 36 eingestellt.

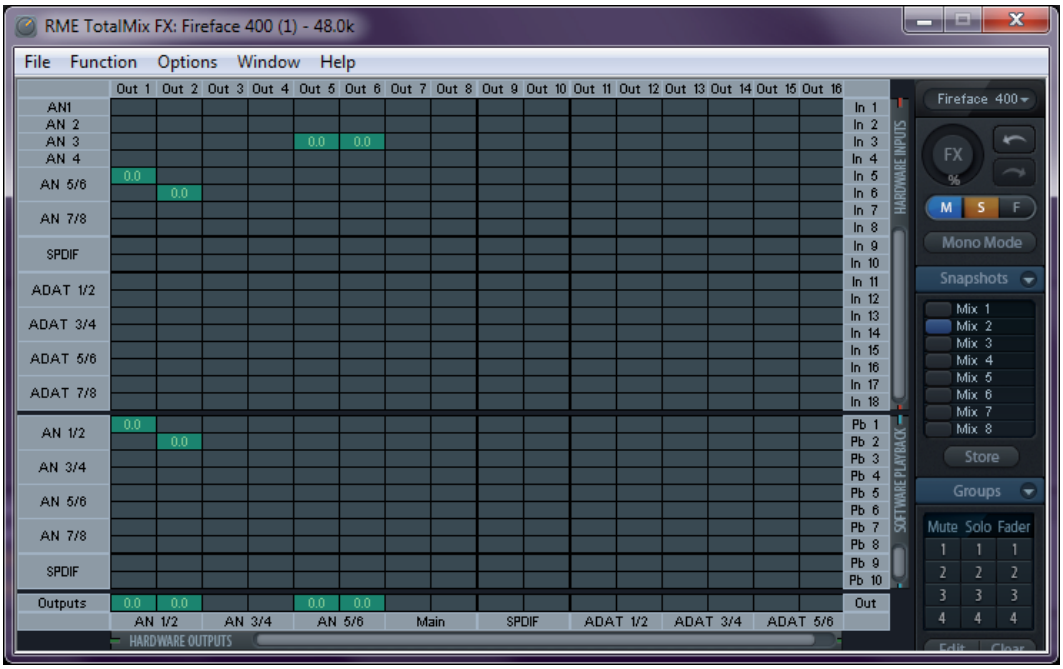


Abbildung 35: Matrix Einstellung zum Spielen und Aufnehmen der Gitarre.

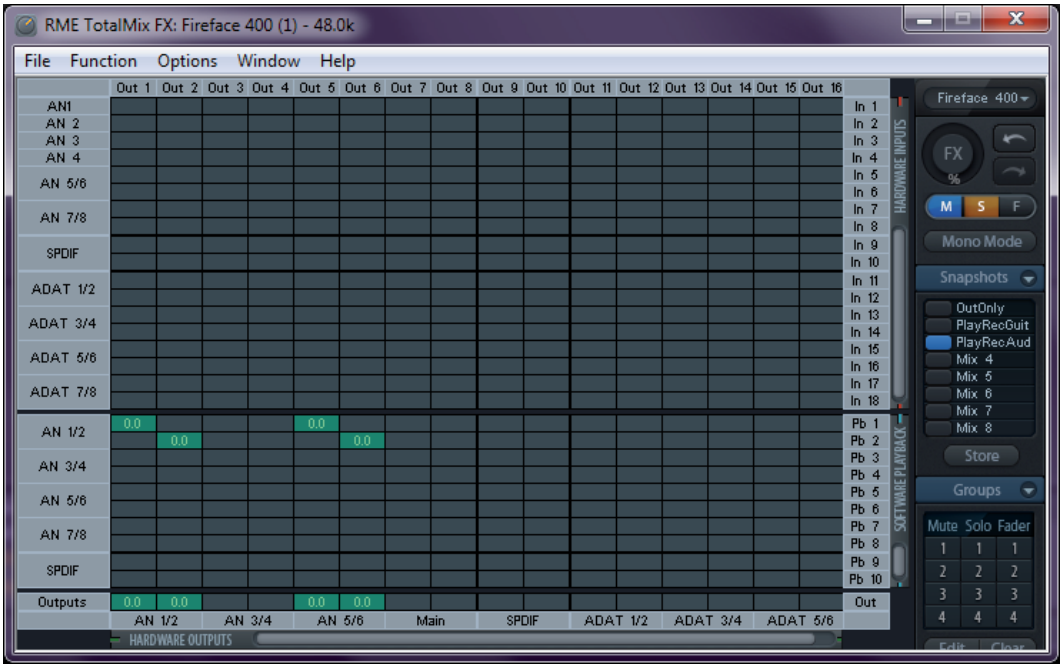


Abbildung 36: Matrix Einstellung Abspielen in Audition und Aufnehmen nach DSP Verarbeitung.