



AG Medizinische Physik
am FB 8, Physik

Dr. V. Hohmann
Prof. Dr. Dr. B. Kollmeier

Skript zum Blockpraktikum „Physikalische Meßtechnik und digitale Signalverarbeitung“

Auszug (Theorieteil)
zur Vorbereitung auf das Praktikum

Wintersemester 2002/2003

Oldenburg, Januar 2003

Vorwort

Das vorliegende Vorbereitungsskript zum Blockpraktikum „Physikalische Meßtechnik und digitale Signalverarbeitung“ stellt einen Auszug aus dem Skript zum gleichnamigen Blockpraktikum vom Wintersemester 98/99 und 99/2000 dar. Es beinhaltet den Theorieteil des Praktikums, das zur Vertiefung der Theorie angeboten wird. Nicht enthalten sind jedoch die Ausarbeitungen des experimentell orientierten Projektteils des Praktikums.

Anzumerken ist, daß der vorliegende Theorieteil von den damaligen Teilnehmerinnen und Teilnehmern geschrieben ist und daher nicht den Anspruch eines Lehrbuchs erfüllen kann. Es ist anzuraten, eines der unten zitierten Lehrbücher zur Vorbereitung hinzuzuziehen.

Oldenburg, Januar 2003

V. Hohmann und B. Kollmeier

Literatur:

Lüke, H.D. „Signalübertragung“, Springer, 1989.

Oppenheim, A.V., Schafer, W. „Digital Signal Processing“, Prentice Hall, 1975.

Stearns, S.D. „Digitale Verarbeitung analoger Signale“, Oldenbourg 1979.

Inhaltsverzeichnis

1	Allgemeine Signaltheorie und AD/DA-Wandlung	6
1.1	Einleitung	6
1.2	Fourieranalyse	6
1.2.1	Praktische Durchführung	6
1.2.2	Analytische Umsetzung eines analogen Filters	7
1.3	Diskrete Signale	8
1.3.1	Motivation	8
1.3.2	Mathematische Darstellung	8
1.4	Quantisierte Signale	9
1.5	Signalausgabe	10
1.5.1	Deltakamm	10
1.5.2	Rechteck	10
1.5.3	Dreieck	11
1.5.4	Sinc-Funktion	12
2	Spektralanalyse und DFT	13
2.1	Einleitung	13
2.2	Diskretisierung von Signalen	13
2.2.1	Diskretisierung im Zeitbereich	13
2.2.2	Diskretisierung im Frequenzbereich	15
2.3	Diskrete Fouriertransformation (DFT)	16
2.3.1	Fast Fouriertransformation (FFT)	16
2.4	Fensterung	16
2.5	Zeit-Frequenz-Analyse	18
2.5.1	Filterbank-Summation	19
2.5.2	Kurzzeitspektralanalyse	20

3	Digitale Filter	24
3.1	Allgemeine Einführung: Wozu dienen Filter?	24
3.2	LTI-Systeme	25
3.3	Elementaroperationen und Differenzengleichung	28
3.4	Übertragungsfunktion der Differenzengleichung	30
3.5	Die z-Transformation	31
3.6	Pol- und Nullstellenzerlegung	31
3.7	Definitionsbereich von $H(z)$	33
3.8	Implementation der Differenzengleichung auf dem Rechner	36
3.9	Schlußbemerkungen	37
3.9.1	Verringerung des Rechenaufwands	37
3.9.2	Vergleich von Polen und Nullstellen bezüglich des Rechenaufwands	37
4	Entwurf Digitaler Filter	39
4.1	Einleitung	39
4.2	Umsetzung analoger in digitale Filter	40
4.2.1	Einschub: Analoge Filter	40
4.2.2	Die z-Transformation	41
4.2.3	Impulsantwortinvariantes Design	42
4.2.4	Frequenzganginvariantes Design	43
4.2.5	Inverses Filter	45
4.3	Typen analoger Filter	45
4.3.1	Butterworth	45
4.3.2	Chebyshev	46
4.3.3	Bessel	46
4.4	Design digitaler FIR-Filter	46
4.4.1	Phasenlinearität von FIR-Filtern	47
4.4.2	Windowing-Design	47
4.4.3	Frequenzabtastfilter	48
4.4.4	Gradientenverfahren	50

5	Versuche zur AD/DA-Wandlung	51
5.1	Einleitung	51
5.2	Programmierung unter MATLAB	51
5.3	Methode	52
5.3.1	Meßapparatur	53
5.3.2	Meßsignal	53
5.3.3	Meßverfahren	54
5.3.4	Meßergebnisse	56
5.4	Diskussion	56
6	Experimente zur Spektralanalyse und DFT	58
6.1	Zusammenfassung und Einleitung	58
6.2	Versuchsaufbau	58
6.3	Versuchsdurchführung	58
6.4	Auswirkungen der Fenstertechnik	60
6.5	Sweep-Signal und Kurzzeitspektren	62
6.5.1	Begrenzung des Sweepsignals	62
6.5.2	Anwendung verschiedener Fensterfunktionen	66
6.5.3	Kurzzeitspektren des Sweepsignals (mit verschiedenen Fensterfunktionen)	69
7	Matlab - erste Schritte	71
7.1	Bedienung von Matlab	71
7.2	Hilfefunktionen	72
7.3	Variablen	72
7.4	Operatoren	73
7.5	Schleifen und Bedingungen	74
7.6	Generierung von Matrizen/Vektoren	75
7.7	Funktionen	75
7.8	Visualisierung	76
7.9	Eingabe und Ausgabe	77
7.10	Erstellung eigener Befehle („M-Files“)	77
7.11	Aufgaben	78

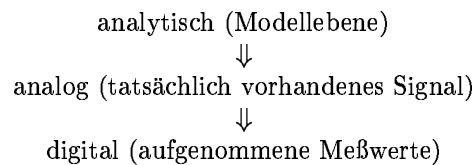
- Theoretische Grundlagen -

1 Allgemeine Signaltheorie und AD/DA-Wandlung

von Domenik Helms und Frank Dannemann

1.1 Einleitung

Um sich mit den allgemeinen Begriffen der Signaltheorie vertraut zu machen, sollte einführend zunächst die Frage gestellt werden, wie man Signale vernünftig messen kann bzw. was man überhaupt messen möchte. Bei diesem Prozeß werden drei Ebenen durchlaufen:



Ideale Voraussetzung wäre ein Signal, dessen Frequenzen auf der analytischen Modellebene bereits vor Beginn der Verarbeitung bekannt sind. Dies wird mittels der *Fourieranalyse* realisiert.

1.2 Fourieranalyse

Bei der Fourieranalyse wird ein Signal $s(t)$ aufgeteilt in eine Summe von Frequenzen:

$$\begin{aligned}
 \sum A(f) \cos(2\pi ft + \varphi(f)) &= \operatorname{Re}(S(f) \cdot e^{i2\pi ft}) \\
 S(f) &= A(f) \cdot e^{i\varphi(f)} \\
 \Rightarrow s(t) &\longleftrightarrow \int_{-\infty}^{\infty} s(t) e^{-i2\pi ft} dt = S(f)
 \end{aligned} \tag{1.1}$$

Möchte man aus dem transformierten Signal wieder das Originalsignal gewinnen, so geschieht dies mittels Rücktransformation:

$$\begin{aligned}
 s(t) &= \int_{-\infty}^{\infty} S(f) \cdot e^{2\pi i ft} df \\
 &= \int_0^{\infty} S(f) \cdot e^{2\pi i ft} \int_{-\infty}^0 S(f) e^{2\pi i ft} df \\
 &= \int_0^{\infty} S(f) \cdot e^{2\pi i ft} + \left[\int_0^{\infty} S^*(-f) e^{2\pi i ft} df \right]^*
 \end{aligned} \tag{1.2}$$

Wie aus Gleichung 1.2 ersichtlich ist, müssen auch negative Frequenzen zugelassen werden, die auch als *Spiegelfrequenzen* bezeichnet werden. Dies ist nötig um sicherzugehen, daß das Ergebnis wieder reell ist d.h. das ursprüngliche Signal wieder rekonstruiert werden kann. Durch die „Unschärferelation“ $\Delta T \Delta B \geq 1$ entstehen bei der Fouriertransformation jedoch Informationsverluste beim Eingangssignal, die mit jeder Filterung von $s(t)$ einhergeht.

1.2.1 Praktische Durchführung

Um ein analoges Signal in seine Frequenzen zu zerlegen, kann z.B ein Bandpaßfilter verwandt werden, wie er in Abbildung 1.1 dargestellt ist.

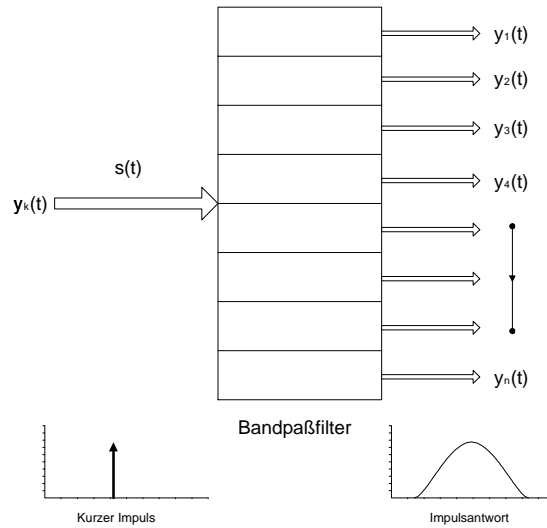


Abbildung 1.1: Analoge Umsetzung einer Fouriertransformation

Dieser ist zusammengesetzt aus einzelnen Resonatoren mit verschiedenen Resonanzfrequenzen: Wir wollen im Folgenden versuchen, dieses Filterverhalten analytisch zu simulieren.

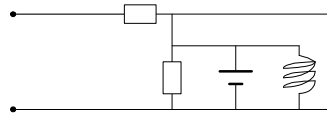


Abbildung 1.2: Mögliche Umsetzung eines Resonators in analoge technische Schaltungen, bestehend aus Ohmschen Widerstand, Spule und Kondensator

1.2.2 Analytische Umsetzung eines analogen Filters

Um aus einem Signal $s(t)$ bestimmte Frequenzen analytisch herauszufiltern, benötigen wir eine Übertragungsfunktion $H_k(f)$. Multipliziert man diese im Frequenzbereich mit $s(f)$, so entspricht dies einer Faltung mit dem Eingangssignal im Zeitbereich. Das Ergebnis wird als *Impulsantwort* bezeichnet. Anschaulich gibt diese die Antwort auf die Frage, wie das Signal auf bestimmte Frequenzen reagiert:

$$\begin{array}{lll} s(f) & \longrightarrow & s(f) \cdot H_k(f) = Y_k(f) \\ \circ \text{---} \bullet & s(t) & \longrightarrow s(t) * h_k(t) = y_k(t) \quad \text{Impulsantwort} \end{array}$$

Für $s(t) * h_k(t)$ können wir auch schreiben

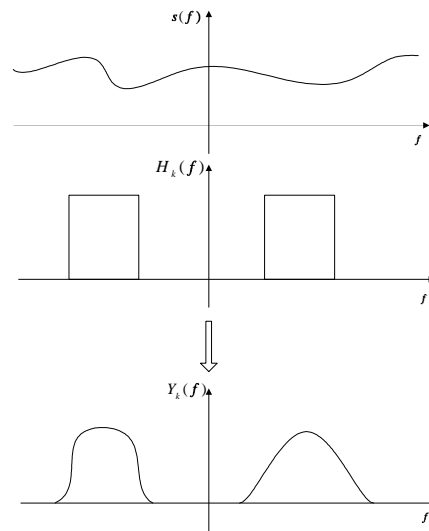
$$s(t) * h_k(t) = \int_{-\infty}^{\infty} s(t - \tau) h_k(\tau) d\tau \quad (1.3)$$

Hieraus ergibt sich der *Faltungssatz*:

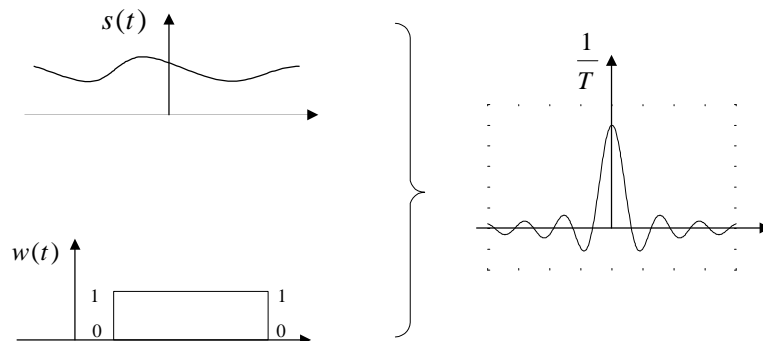
$$\boxed{\begin{array}{ll} \mathcal{F}(x(t) \cdot y(t)) & = \mathcal{F}(x(t)) * \mathcal{F}(y(t)) \\ \mathcal{F}(x(t) * y(t)) & = \mathcal{F}(x(t)) \cdot \mathcal{F}(y(t)) \end{array}} \quad \text{Faltungssatz}$$

Mit diesem Wissen ist es nun beispielsweise analytisch möglich, einen bestimmten Zeitbereich aus einer Funktion herauszuschneiden. Dazu benutzen wir als Übertragungsfunktion eine sogenannte *Fensterfunktion* $w(t)$:

$$Y(t) = s(t) \cdot w(t) \quad \circ \text{---} \bullet \quad Y(f) = S(f) * W(f) \quad (1.4)$$

Abbildung 1.3: Anwendung einer Übertragungsfunktion $H_k(f)$

Wählt man zunächst eine Rechteckfunktion und faltet diese mit dem Eingangssignal, so erhält man eine *sinc*-Funktion ($\frac{\sin(x)}{x}$) (Siehe Abb. 1.4). Da diese jedoch unendlich ausgedehnt ist, wäre es besser, z.B. eine Hamming-Funktion für die Faltung zu benutzen (s. nachfolgende Theorie-Teile).

Abbildung 1.4: *Windowing*: Das Signal $s(t)$ wird mit der Fensterfunktion $w(t)$ gefaltet

1.3 Diskrete Signale

1.3.1 Motivation

Um ein nicht mathematisch beschreibbares Signal zu bearbeiten, insbesondere zu fourieranalysieren, kann ein Computer verwendet werden. Da der Computer eine endlich lange Zeit braucht um einen Wert des Signals aufzunehmen, und er auch nur endlich viele verschiedene Werte unterscheiden kann, verarbeitet er keine stetigen, sondern nur diskrete, quantisierte Signale in Form von Samples. Deshalb ist es für die Veranschaulichung der Vorgänge im Computer notwendig, eine mathematische Darstellung für diskrete, quantisierte Signale zu finden. Mathematische Aussagen über die Fourieranalytierte solcher Signale gelten allgemein, also insbesondere auch für die Daten, die im Computer berechnet werden.

1.3.2 Mathematische Darstellung

Um ein diskretes Signal (z.B. ein digitalisiertes kontinuierliches Signal) mathematisch zu behandeln, kann man es als Produkt einer kontinuierlichen Funktion mit einem Deltakamm darstellen. Der Deltakamm ist definiert als

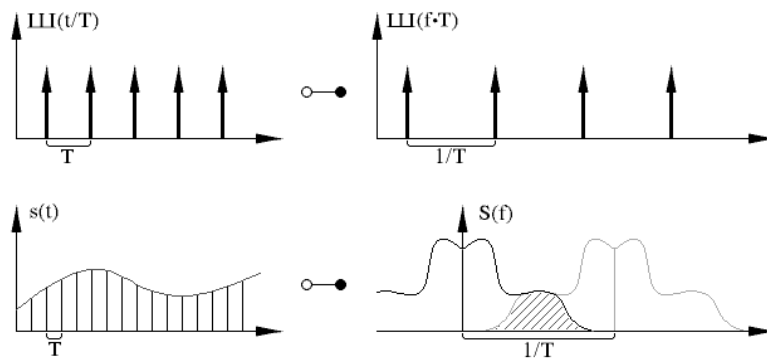
$\sqcup(x) = \sum_{n \in \mathbb{Z}} \delta(x - n)$. Soll eine Funktion $s(t)$ zu den Zeiten $n \cdot T, n \in \mathbb{N}$ dargestellt werden, multipliziert man sie mit einem skalierten Deltakamm $\sqcup(\frac{t}{T})$.

$$s_d(t) = s(t) \cdot \sqcup\left(\frac{t}{T}\right) \quad (1.5)$$

Was aus diesem Signal mittels Fourieranalyse wird, läßt sich leicht zeigen: Es gilt $\sqcup(\frac{t}{T}) \circ \bullet \sqcup(f \cdot T)$, so daß mit Gleichung 1.4 folgt:

$$s_d(t) = s(t) \cdot \sqcup\left(\frac{t}{T}\right) \circ \bullet S_d(f) = S(f) * \sqcup(f \cdot T) \quad (1.6)$$

Das fourieranalytierte Signal wird also durch die Faltung mit dem Deltakamm periodisch und zwar mit der Periode $\frac{1}{T} = f_s$, der Samplingfrequenz.



In der Grafik ist oben die Fourieranalyse eines Deltakamms dargestellt. Aus der Periode T wird in der Fourieranalytierten $\frac{1}{T} = f_s$. Darunter ein kontinuierliches Signal $s(t)$, das in Abständen von T ausgelesen wird. Da aber die Fourieranalytierte mit dem Deltakamm gefaltet und nicht multipliziert wird, wiederholt sie sich $\frac{1}{T}$ -periodisch. Dadurch kann es passieren, daß sich die einzelnen Graphen im $S(f)$ -Diagramm überschneiden. Die Fourieranalyse liefert hier die Summe der einzelnen Graphen und somit ein falsches Ergebnis, weil größere Frequenzen als $\frac{f_s}{2}$ von der nächsten Periode aufaddiert werden. Um derartige Fehler zu vermeiden ist es notwendig, das **Samplingtheorem** zu beachten, welches besagt, daß $\frac{f_s}{2}$ die größte im Signal zulässige Frequenz ist. Denn wenn es keine Frequenzen gibt, die größer sind als $\frac{f_s}{2}$ überschneiden sich die Graphen erst garnicht und addieren sich somit auch nicht falsch auf.

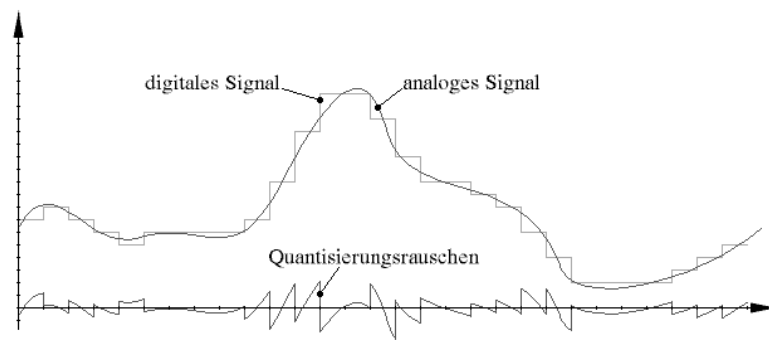
1.4 Quantisierte Signale

Da das vom Computer zu verarbeitende Signal wie oben erwähnt quantisiert ist, treten bei der Eingabe eines analogen Signals in den Computer Rundungsfehler auf, das sogenannte **Quantisierungsrauschen**, weil der Computer sich entscheiden muß, welcher digitalen Zahl er den analogen Wert zuordnet.

Bezeichnend für die Größe des Verlusts ist der **Signal-Rausch Abstand** ($\frac{S}{N}$) der das Verhältnis zwischen Signalpegel und Rauschpegel angibt. Bei einer binären Darstellung mit n Bit entspricht die größtmögliche Amplitude dem Wert $\frac{1}{2} \cdot 2^n$, da sowohl positive als auch negative Amplituden dargestellt werden müssen. Der größtmögliche Fehler bei korrekter Rundung entspricht dem Wert $\frac{1}{2}$. Es gilt also für den Signal-Rausch Abstand bei n -Bit Darstellung:

$$\frac{S}{N} = \frac{2^{n-1}}{\frac{1}{2}} = 2^n = 20 \cdot \log_{10}(2^n) \text{ dB} = n \cdot 20 \cdot \log_{10}(2) \text{ dB} \approx n \cdot 6 \text{ dB} \quad (1.7)$$

Pro Bit *Sampletiefe* steigt der $\frac{S}{N}$ also um etwa 6 dB . Das bedeutet z.B. für eine CD (16 Bit): $\frac{S}{N} = 16 \cdot 6 \text{ dB} = 96 \text{ dB}$

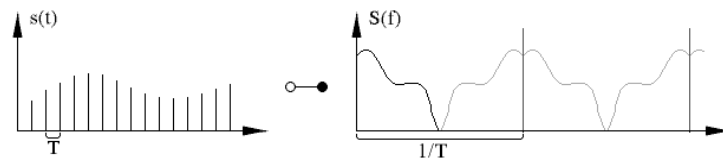


1.5 Signalausgabe

Um eine diskrete, quantisierte Funktion wieder analog auszugeben gibt es mehrere Modelle. Je aufwendiger das Signal rekonstruiert wird, desto mehr ähnelt es einem analogen, stetigen Signal. Der Rechenaufwand für den Computer steigt aber auch mit immer komplexeren Modellen.

1.5.1 Deltakamm

Würde das Signal einfach als Deltakamm ausgegeben werden, sollten sich die selben mathematischen Konsequenzen für die im Signal enthaltenen Frequenzen ergeben, wie beim Einlesen. Es gäbe dann bei jedem vielfachen von $\frac{1}{T}$ sogenannte Spiegelfrequenzen, die im Originalsignal (bei Einhaltung des Samplingtheorems) nicht vorhanden waren. Das die Ausgabe als Deltakamm trotzdem ein ähnliches Signal wie das Eingangssignal ausgibt hat zwei Gründe:



1.) Analoge Geräte können nur endlich hohe Frequenzen bearbeiten. Insbesondere in einem Lautsprecher werden (je nach Typ) sehr hohe Frequenzen nicht mehr ausgegeben. Außerdem kann das menschliche Ohr auch nur Frequenzen bis zu einer bestimmten Grenzfrequenz wahrnehmen. Ist also die Ausgabefrequenz hoch genug, können vom Ohr keine Spiegelfrequenzen mehr wahrgenommen werden.

2.) Es ist für einen Computer (so wie für jedes andere reale System) nicht möglich, wirkliche Deltaimpulse auszugeben. (Unter anderem wegen unendlich hoher Amplitude und Flankensteilheit.) Deshalb kann man das Ausgabesignal bestenfalls als Rechteckfunktion betrachten.

1.5.2 Rechteck

Eine weitere Möglichkeit der Ausgabe (die in vielen einfachen Systemen auch genutzt wird) ist es, den Wert der Amplitude über die gesamte Zeit T zu halten. Das ist digital sehr einfach realisierbar und liefert bessere Ergebnisse als die Ausgabe mit dem Deltakamm. Es entsteht eine Treppenfunktion mit der 'Stufenbreite' T , die sich mathematisch als Faltung des digitalen Signals mit der Rechteckfunktion darstellen lässt.

$$s(t) = s_d(t) * \text{rect}\left[-\frac{T}{2}, \frac{T}{2}\right] \quad (1.8)$$

Die Rechteckfunktion ist innerhalb des angegebenen Intervalls 1 und 0 sonst. Um das Frequenzspektrum zu bestimmen, benötigt man die Fourieranalyse der Rechteckfunktion:

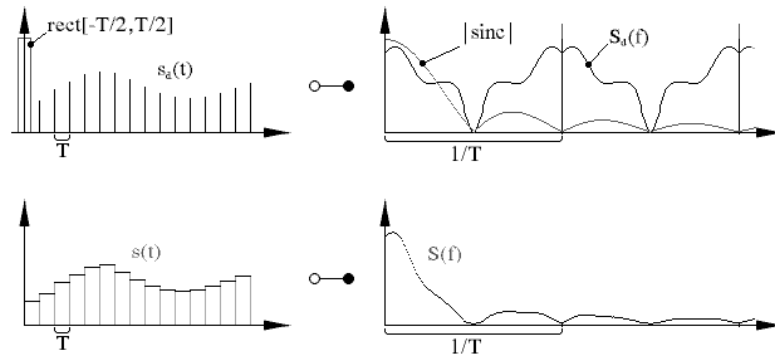
$$\mathcal{F}(\text{rect}[-\frac{T}{2}, \frac{T}{2}]) = \int_{-\infty}^{\infty} \text{rect}(t) \cdot e^{-2\pi i f t} dt = \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-2\pi i f t} dt = \frac{-1}{2\pi i f} \cdot (e^{-2\pi i f \frac{T}{2}} - e^{2\pi i f \frac{T}{2}}) \quad (1.9)$$

$$= \frac{1}{\pi f} \cdot \frac{(e^{\pi i f T} - e^{-\pi i f T})}{2i} = \frac{\sin(\pi f T)}{\pi f} = T \cdot \text{sinc}(\pi f T) \quad (1.10)$$

Deshalb gilt für die Fourieranalyse des Ausgangssignals:

$$s(t) = s_d(t) * \text{rect}[-\frac{T}{2}, \frac{T}{2}] \longleftrightarrow S(f) = S_d \cdot T \cdot \text{sinc}(\pi f T) \quad (1.11)$$

Das Spektrum wird also mit der sinc-Funktion skaliert. Diese hat ihren ersten Nulldurchgang bei $\pi f T = \frac{\pi}{2} \Rightarrow f = \frac{1}{2} \cdot \frac{1}{T} = \frac{f_s}{2}$. Das heißt, das Spiegelfrequenzen über $\frac{f_s}{2}$ mit $\frac{1}{f}$ gedämpft werden. Aber auch erwünschte Frequenzen leicht unter $\frac{f_s}{2}$ werden stark gedämpft.



1.5.3 Dreieck

Als Vorschlag aus dem Auditorium kam die Idee, zwischen den Samples linear zu interpolieren. Dieses Verfahren findet zwar selten Anwendung, weil es bei vergleichsweise hohem Aufwand wenig Qualitätsgewinn bedeutet, ist aber mathematisch interessant.

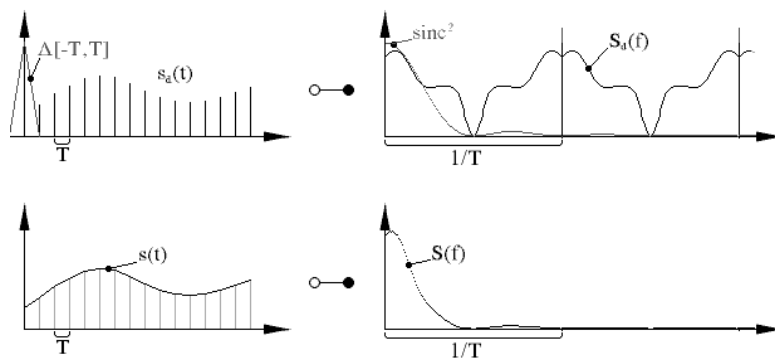
Die Idee ist, jeden einzelnen Peak als Dreieck zu interpretieren, das jeweils beim vorherigen Sample bei Null anfängt, bis zum aktuellen Sample auf den Wert der Amplitude ansteigt und dann bis zum nächsten Sample wieder auf Null abfällt. Ausgegeben wird dann die Summe der sich überlagernden Dreiecke wodurch sich insgesamt eine lineare Interpolation ergibt.

Ein Dreieck ist mathematisch darstellbar als Faltung zweier rect-Funktionen. Wie sich leicht zeigen lässt, gilt: $\text{rect}[-\frac{T}{2}, \frac{T}{2}] * \text{rect}[-\frac{T}{2}, \frac{T}{2}] = \Delta[-T, T]$ Dadurch folgt für das Spektrum der Ausgabe

$$s(t) = s_d(t) * \Delta = s_d(t) * \text{rect} * \text{rect} \longleftrightarrow S(f) = S_d(f) \cdot \text{sinc}^2(f) \quad (1.12)$$

Der Vorteil gegenüber der Rechteckausgabe ist, daß $\text{sinc}^2(x)$ unterhalb von $\frac{f_s}{2}$ 'rechteckiger' ist als $\text{sinc}(x)$ und oberhalb schneller (nämlich quadratisch) abfällt.

Die Dreiecksausgabe ist von der Ausgabequalität zwar besser als die Rechteckausgabe, dafür ist sie aber digital kaum zu realisieren. Es müßten Zwischenwerte zwischen den ursprünglichen Samples berechnet und ausgegeben werden, die dann die lineare Interpolation annähern. Dieses Verfahren wird **Oversampling** genannt.



1.5.4 Sinc-Funktion

Wenn Oversampling verwandt wird, können die zusätzlichen Ausgabewerte aber noch effizienter genutzt werden, als nur zur linearen Interpolation. Das beste Verfahren ist, über die Ausgabefunktion eine sinc-Funktion zu Falten und deren Ergebnisse als Zwischenwerte auszugeben.

$$s(t) = s_d(t) * \text{sinc} \quad \longleftrightarrow \quad S(f) = S_d(f) \cdot \text{rect} \quad (1.13)$$

Alle Frequenzen unterhalb von $\frac{f_s}{2}$ bleiben unverändert, alle Spiegelfrequenzen werden zu Null. Da die sinc-Funktion jedoch unendlich lang ist, kann in der Realität nur mit einem Ausschnitt von ihr gefaltet werden. Ist dieser Ausschnitt allerdings lang genug ist der Qualitätsverlust relativ gering.

2 Spektralanalyse und DFT

von Ralf Bruns und Michael Leißner

2.1 Einleitung

Die Verarbeitung und Analyse von Signalen erfolgt heute fast ausschließlich digital. Analoge Signale sind in der Regel zeit- und wertekontinuierlich. Sie müssen daher zunächst in eine diskrete Form überführt werden. Dabei ist es wünschenswert, daß neben dem Zeitsignal auch das Spektrum als eine Folge diskreter Werte vorliegt.

Im folgenden wollen wir daher die Diskretisierung von Signalen im Zeit- und Frequenzbereich betrachten. Abschließend werden wir auf die Problematik der Zeit-Frequenz-Analyse eingehen.

2.2 Diskretisierung von Signalen

2.2.1 Diskretisierung im Zeitbereich

Um aus einem kontinuierlichen aperiodischen Signal (Abbildung 2.1 a) mit einem kontinuierlichen aperiodischen Spektrum (2.1 b) eine diskrete Folge von Werten zu erhalten, die dann beispielsweise in einer Datei gespeichert werden kann, wird das Signal in konstanten Zeitabständen T abgetastet. T ist dabei durch die Samplingfrequenz f_s gegeben mit $T = \frac{1}{f_s}$. Zu jedem Abtastzeitpunkt wird ein ebenfalls diskreter Wert ermittelt, wobei die Anzahl n der zur Verfügung stehenden Quantisierungsstufen durch die Bitbreite gegeben ist: $n = 2^{Bits}$. Je höher die Bitbreite gewählt wird, umso kleiner wird der Quantisierungsfehler¹ q , denn es gilt $q \approx A \cdot 2^{-Bits}$, wobei A die Vollaussteuerung ist.

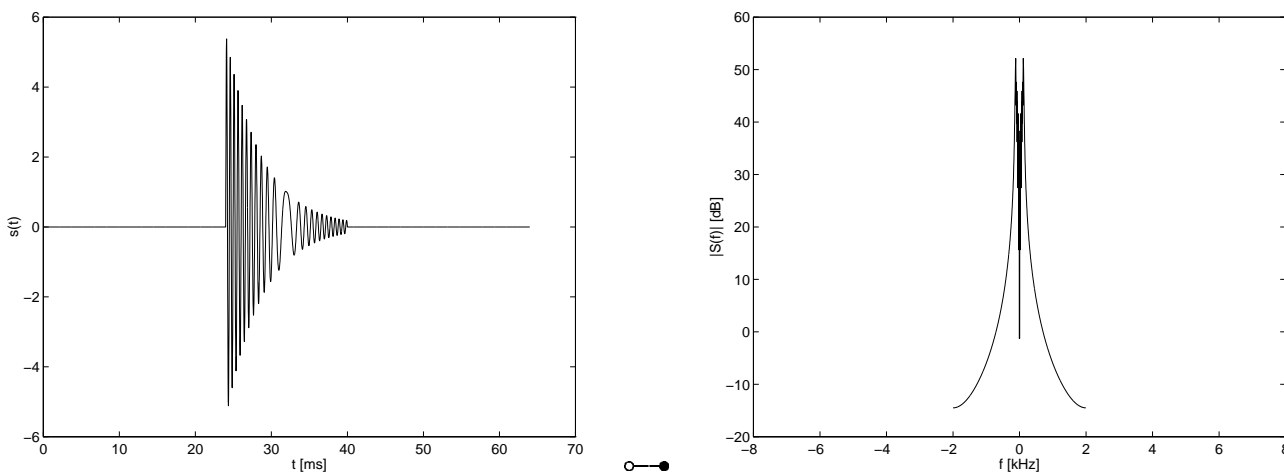


Abbildung 2.1: (a) Kontinuierliches aperiodisches Signal

(b) Kontinuierliches aperiodisches Spektrum

Mathematisch läßt sich die Diskretisierung im Zeitbereich am besten durch die Multiplikation des kontinuierlichen Signals mit einem δ -Kamm beschreiben. Als Ergebnis erhält man das gewünschte diskrete kontinuierliche Signal im Zeitbereich (Abbildung 2.3 a).

Gemäß dem Faltungssatz entspricht die Multiplikation zweier Funktionen der Faltung ihrer Fouriertransformierten. Da der Übergang vom Signal zum Spektrum gerade über die Fouriertransformation geschieht, bedeutet dies, daß wir das Spektrum (Abbildung 2.1 b), das wie das Signal kontinuierlich und aperiodisch ist, mit der Fouriertransformierten des δ -Kamms – einem mit dem Kehrwert der Periodizität umskalierten δ -Kamm mit der inversen Periodizität – falten müssen. Das Ergebnis dieser Faltung ist ein kontinuierliches periodisches Spektrum (Abbildung 2.2 b).

¹Quantisierungsfehler = Differenz von Original- und quantisiertem Signal

Signal	Spektrum
$s(t)$	$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-2\pi i f t} dt$
\bullet	$*$
$\sqcup_T(t)$	$\frac{1}{T} \sqcup_{\frac{1}{T}}(f)$
$=$	$=$
$s'_d(t) = s(t) \cdot \sqcup_T(t)$	$\tilde{S}(f) = \frac{1}{T} \cdot S(f) * \sqcup_{\frac{1}{T}}(f)$
	$= \sum_{n=-\infty}^{+\infty} \frac{1}{T} S(f - \frac{n}{T})$

(2.1)

(2.2)

Die Definitionsmenge lässt sich einschränken, da das Signal diskret ist. Es gilt: $s'_d(t) = 0 \quad \forall t \neq n \cdot T, \quad n \in \mathbf{Z}$. Die neue Definitionsmenge ist daher $n \cdot T, \quad n \in \mathbf{Z}$. Durch Umskalieren setzen wir $s_d(n) = s'_d(n \cdot T)$. Mathematisch nicht ganz korrekt verwenden wir im folgenden die Schreibweise $s_d(n) = s(t) \cdot \sqcup_T(t)$.

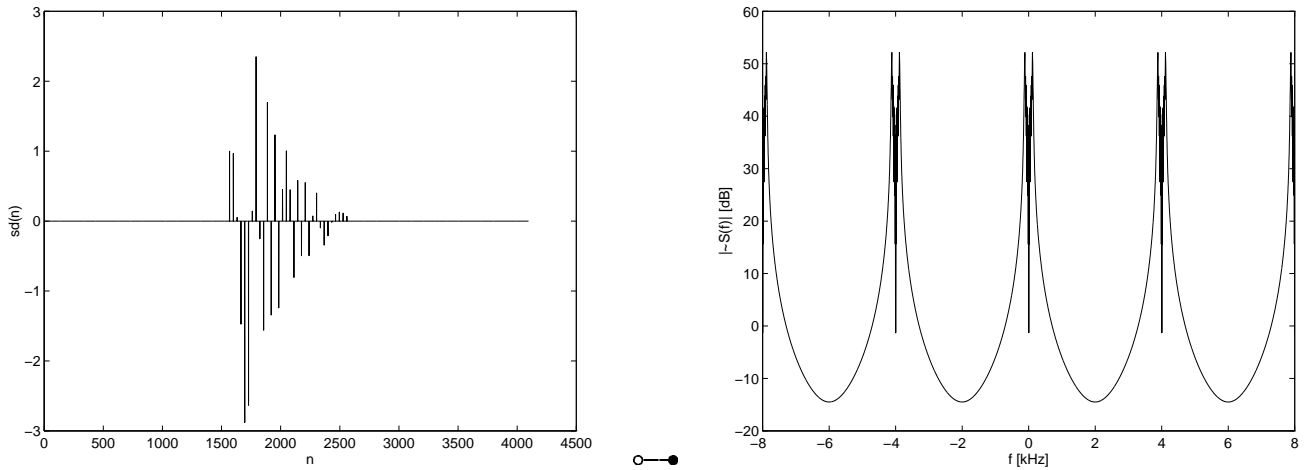


Abbildung 2.2: (a) Diskretes aperiodisches Signal

(b) Kontinuierliches periodisches Spektrum

Da das Spektrum periodisch ist, könnte es zu einem Überlapp der einzelnen Perioden und somit zu Aliasing-Effekten bei der späteren Rekonstruktion des Signals kommen. Um dies zu verhindern, muß die Abtastfrequenz f_s mindestens doppelt so hoch sein, wie die höchste im Spektrum vorkommende Frequenz, die Grenzfrequenz f_g . Dies ist die Bedingung des sogenannten **Abtasttheorems**:

$$f_s = \frac{1}{T} \geq 2 \cdot f_g \quad (2.3)$$

In der Praxis stellt man die Einhaltung des Abtasttheorems sicher, indem man eine Tiefpaßfilterung unterhalb der halben Abtastfrequenz durchführt, die im allgemeinen fest vorgegeben ist. Beträgt die Abtastfrequenz beispielsweise 44100 Hz wie bei einer CD, so müssen alle Frequenzen, die über 22100 Hz liegen, zur Erfüllung des Abtasttheorems aus dem Spektrum herausgefiltert werden. Da das menschliche Gehör Frequenzen in dieser Größenordnung ohnehin nicht mehr wahrnehmen kann, ist dies bei einem Audiosignal nicht mit einem für den Menschen hörbaren Qualitätsverlust verbunden.

2.2.2 Diskretisierung im Frequenzbereich

Es liegt nun ein diskretes Signal und ein kontinuierliches periodisches Spektrum vor. Dieses läßt sich nun analog zur oben behandelten Diskretisierung im Zeitbereich durch Multiplikation mit einem δ -Kamm diskretisieren. Auch hierbei ist wieder der Zusammenhang mit dem Zeitsignal über die Fouriertransformation zu beachten. Denn nun muß eine Faltung im Zeitbereich mit einem δ -Kamm durchgeführt werden, was zwangsläufig dazu führt, daß das Zeitsignal periodisch wird (Abbildung 2.3 a).

Signal		Spektrum	
$s_d(n) = s(t) \cdot \sqcap_T(t)$	○—●	$\tilde{S}(f) = \frac{1}{T} S(f) * \sqcap_{\frac{1}{T}}(f)$	(2.4)
		$= \sum_{n=-\infty}^{+\infty} \frac{1}{T} S(f - \frac{n}{T})$	
*		●	
$\frac{1}{F} \sqcap_{\frac{1}{F}}(t)$	○—●	$\sqcap_F(f)$	
=		=	
$\tilde{s}_d(n) = \frac{1}{F} s_d(n) * \sqcap_{\frac{1}{F}}(t)$	○—●	$\tilde{S}_d(k) = \tilde{S}(f) \cdot \sqcap_F(f)$	(2.5)

Bei der Faltung auf der linken Seite von Gl. 2.5 kann zirkuläres Aliasing offenbar nur vermieden werden, wenn für die „Abtastfrequenz“ $\frac{1}{F}$ gilt (T' ist hierbei die Länge von $s_d(n)$):

$$\frac{1}{F} \geq T' \quad (2.6)$$

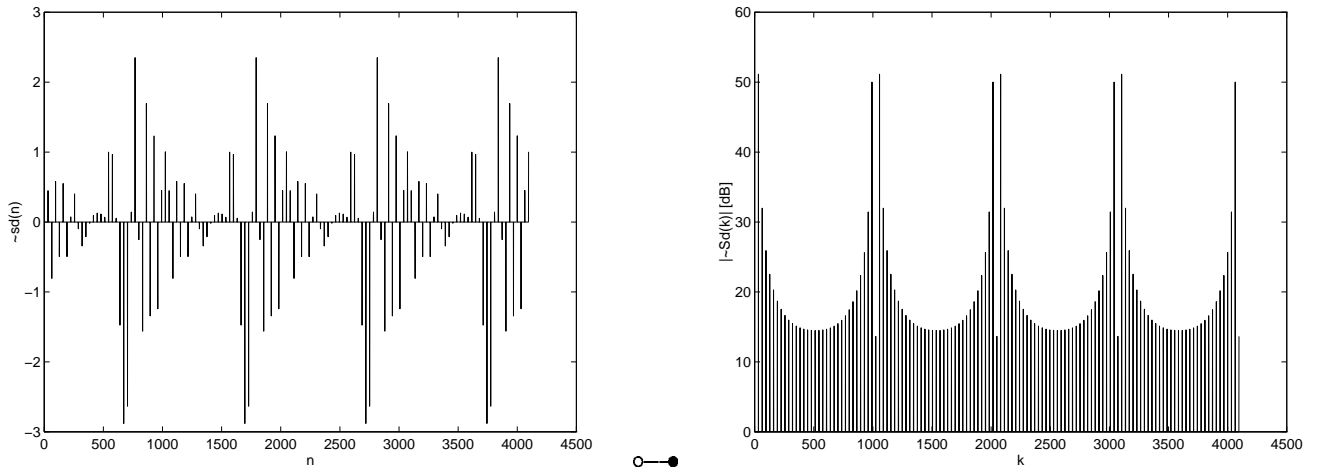


Abbildung 2.3: (a) Diskretes periodisches Signal

(b) Diskretes periodisches Spektrum

Insgesamt erhalten wir nach den beiden Diskretisierungen im Zeit- und Frequenzbereich ein diskretes periodisches Signal und ein diskretes periodisches Spektrum.

Die Feinstruktur (T) des Signals ist dabei zur Grobstruktur (Periodizität $\frac{1}{T}$) des Spektrums geworden, während die Grobstruktur des Zeitsignals (Periodizität $\frac{1}{F}$) zur Feinstruktur (F) des Spektrums geworden ist.

Die Beziehung zwischen einer Periode des diskreten periodischen Signals und einer Periode des diskreten periodischen Spektrums ist durch die Diskrete Fouriertransformation (DFT) gegeben.

2.3 Diskrete Fouriertransformation (DFT)

Wie bereits beschrieben hat ein Zeitsignal mit in F gequanteltem Spektrum die Periodizität $\frac{1}{F}$. Da das Zeitsignal in T gequantelt ist, muß gelten:

$$\frac{1}{F} = N \cdot T, \quad N \in \mathbb{N} \quad (2.7)$$

Damit sind nun **DFT** und inverse DFT definiert als:

$$\begin{array}{ccc} \text{inverse DFT} & & \text{DFT} \\ s(n) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) e^{2\pi i \frac{n k}{N}} & \longleftrightarrow & S(k) = \sum_{n=0}^{N-1} s(n) e^{-2\pi i \frac{n k}{N}} \end{array} \quad (2.8)$$

Die DFT stellt die Beziehung zwischen einer Periode des diskreten periodisch fortgesetzten Signals und einer Periode des diskreten periodischen Spektrums her.

Die DFT sieht formal ähnlich aus wie die Fouriertransformation und hat auch fast die gleichen Eigenschaften: Im Unterschied zur Fouriertransformation wird jedoch nicht über die gesamte Zeit integriert, sondern über eine Periode des diskreten Signals summiert. Der Normierungsfaktor $\frac{1}{N}$ wird benötigt, damit nach der Hintereinanderausführung von Hin- und Rücktransformation wieder das ursprüngliche Signal vorliegt. Dieser Faktor ist beliebig auf DFT und inverse DFT verteilbar, die in Gl. 2.8 angegebene Verteilung ist rein willkürlich gewählt.

Ein weiterer Unterschied wird beim Betrachten des Faltungsproduktes zweier diskreter Signale deutlich. Es ist über die DFT mit dem Produkt der zugehörigen Spektren verbunden. Da diese aber nur auf eine Periode des periodischen Spektrums Anwendung findet, gilt der Faltungssatz hier auch nur für eine zyklische Faltung. Bei der Rücktransformation in den Zeitbereich erhält man daher periodisch fortgesetzte Zeitfunktionen, was unter Umständen zu zirkulärem Aliasing führen kann.

Dies läßt sich jedoch verhindern, indem man an das periodische Signal im Zeitbereich Nullen anfügt und das Fenster somit vergrößert (siehe Abschnitt 2.4). Zudem wird dadurch die Auflösung aufgrund einer besseren Abtastung im Frequenzbereich erhöht.

Der Rechenaufwand bei der Durchführung der DFT liegt normalerweise bei M^2 , da zur Berechnung eines Spektralwertes M Multiplikationen mit M Additionen ausgeführt werden müssen.

Er läßt sich durch Anwendung der Fast Fouriertransformation verringern:

2.3.1 Fast Fouriertransformation (FFT)

Die Fast Fouriertransformation ist ein etwas komplizierterer Algorithmus, der durch geschickten Umgang mit Zwischenergebnissen einige redundante Rechenschritte in der DFT eliminiert. Bei der FFT verringert sich der Rechenaufwand, wenn M in wenige Primfaktoren zerfällt. Im günstigsten Fall ist M eine Zweierpotenz. In dem Fall ist der Rechenaufwand proportional zu $M \cdot \log_2 M$. Durch das Anfügen von Nullen läßt sich das zu transformierende Signal immer auf eine passende Größe $N = 2^l$, $l \in \mathbb{N}$ bringen. Aufgrund der für große M erheblich geringeren Rechenarbeit wird die FFT normalerweise für die Berechnung der DFT herangezogen.

2.4 Fensterung

Soll ein Signal mit dem Rechner bearbeitet werden, so muß es nicht nur diskretisiert sondern auch endlich sein. Liegt ein unendlich langes Signal vor, oder soll eine Zeit-Frequenz-Analyse durchgeführt werden, so muß ein Teil des Signals isoliert werden. Dies geschieht durch Multiplikation des Zeitsignals $s(t)$ mit einer sog. Fensterfunktion $w(t)$, welche nur in einem begrenzten Zeitbereich T' ungleich Null ist. Dieser gefensterte Bereich des Signals wird als eine Periode eines periodischen Signals betrachtet, damit das Spektrum diskret wird. Zusammen mit der

notwendigen Diskretisierung erhält man aus dem kontinuierlichen aperiodischen Signal $s(t)$ das diskrete periodische Signal $s_w(t)$:

$$s_w(t) = \underbrace{\left(\underbrace{s(t) \cdot \sqcup\sqcup_T(t)}_{\text{diskretisiert}} \cdot w(t) \right)}_{\text{gefenstert}} * \sqcup\sqcup_{T'}(t) \quad (2.9)$$

Bei der Multiplikation von $s(t)$ und $w(t)$ werden die Spektren gemäß des Faltungssatzes gefaltet. Damit das Signalspektrum möglichst wenig verfälscht wird, ist also eine Fensterfunktion $w(t)$ erwünscht, deren Spektrum $W(f)$ einer δ -Distribution möglichst ähnlich ist; eine endliche Zeitfunktion mit einem exakt δ -förmigen Spektrum gibt es nicht. In der Praxis wird häufig eine der beiden folgenden Fensterfunktionen verwendet:

$$\text{Hanning-Fenster der Länge } T: \quad w(t) = \begin{cases} 0.5 + 0.5 \cos(2\pi \cdot \frac{t}{T}), & -\frac{T'}{2} \leq t \leq \frac{T'}{2} \\ 0, & |t| > \frac{T'}{2} \end{cases} \quad (2.10)$$

$$\text{Hamming-Fenster der Länge } T: \quad w(t) = \begin{cases} 0.54 + 0.46 \cos(2\pi \cdot \frac{t}{T}), & -\frac{T'}{2} \leq t \leq \frac{T'}{2} \\ 0, & |t| > \frac{T'}{2} \end{cases} \quad (2.11)$$

Oder, diskretisiert (hier ähnelt die Fourier-Transformierte natürlich einem δ -Kamm):

$$\text{Hanning-Fenster aus } M \text{ Samples:} \quad w_d(n) = \begin{cases} 0.5 + 0.5 \cos(2\pi \cdot \frac{n}{M-1}), & -\frac{M}{2} \leq n \leq \frac{M}{2} \\ 0, & |n| > \frac{M}{2} \end{cases} \quad (2.12)$$

$$\text{Hamming-Fenster aus } M \text{ Samples:} \quad w_d(n) = \begin{cases} 0.54 + 0.46 \cos(2\pi \cdot \frac{n}{M-1}), & -\frac{M}{2} \leq n \leq \frac{M}{2} \\ 0, & |n| > \frac{M}{2} \end{cases} \quad (2.13)$$

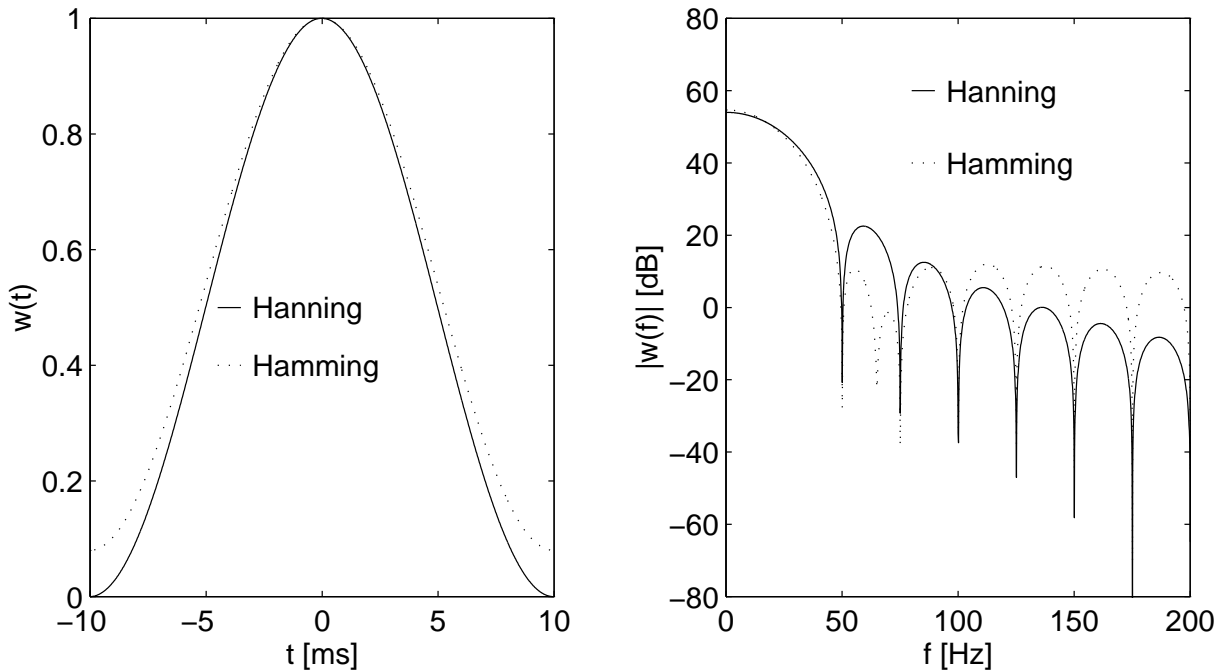


Abbildung 2.4: Hanning- und Hammingfenster im Zeit- und Frequenzbereich

Die Abbildung 2.4 stellt die beiden Fensterfunktionen in Zeit- und Frequenzbereich für eine bei der Analyse von Sprache übliche Fensterlänge von 20 ms dar. Die ersten Nullstellen, die aufgrund der endlichen Auflösung nur angedeutet als solche zu erkennen sind, liegen für ein M Samples langes Fenster bei $\frac{2}{M}$. Im Frequenzbereich ist

die logarithmische Darstellung zu beachten.

Der Pegel im Frequenzbereich ist frei normierbar und hängt davon ab, wie der Normierungsfaktor $\frac{1}{N}$ auf die DFT und die inverse DFT verteilt ist (siehe Abschnitt 2.3). Die der Abb. 2.4 zugrunde liegende Normierung ist die in Gleichung 2.8 beispielhaft angegebene.

Damit ein diskretes Spektrum entsteht, muß wie oben beschrieben das gefensterter Signal als eine Periode eines periodischen Signals betrachtet werden. Eine verbesserte Frequenzauflösung kann erreicht werden, wenn die DFT-Länge N größer gewählt wird als die Fensterlänge M . Dadurch werden die Perioden T' länger und die Frequenzauflösung besser: Der (unveränderte) Frequenzbereich $-\frac{f_s}{2}$ bis $\frac{f_s}{2}$ wird dann N -mal abgetastet. Beim Vergrößern der DFT-Länge werden die fehlenden Samples mit Nullen aufgefüllt. Dies wird als „zero-padding“ bezeichnet und kann wie in Abschnitt 2.3.1 erwähnt auch benutzt werden, um eine schnelle FFT zu ermöglichen. Es gibt noch einen weiteren Grund dafür, $N > M$ zu wählen: Wird das Signal gefiltert, so kann es sich z.B. bei einem FIR-Filter um die Länge der Impulsantwort verlängern. Durch das Einfügen von hinreichend vielen Nullen kann zirkuläres Aliasing vermieden werden. Wir erhalten so die Bedingung:

$$N \geq M + (\text{Länge der Impulsantwort}) \quad (2.14)$$

Durch die Vergrößerung der DFT-Länge kann nur die Abtastung des bereits mit der Fourier-Transformierten des Fensters verschmierten Spektrums verbessert werden; neue Informationen werden dabei nicht gewonnen. Damit das Spektrum originalgetreuer abgetastet werden kann, muß die Fensterlänge M vergrößert werden; dadurch nähert sich das Spektrum des Fensters mehr einer δ -Funktion an. Gemäß der Unschärferelation verliert man dadurch aber Informationen im Zeitbereich. Das gewonnene Spektrum ist in der Zeit weniger scharf lokalisiert.

Eine optimale Zeit- und Frequenzauflösung zu finden ist eine Aufgabe der Zeit-Frequenz-Analyse.

2.5 Zeit-Frequenz-Analyse

Die Zeit-Frequenz-Analyse ist mit dem prinzipiellen Problem der Unschärferelation behaftet. Zeitpunkt und Frequenz können prinzipiell nicht gleichzeitig beliebig genau gemessen werden. Es gilt $\Delta f \cdot \Delta t \geq \frac{1}{2}$. Hierbei sind Δf und Δt als Varianzen von Meßreihen zu verstehen.

Üblicherweise wird als Zeit-Frequenz-Verteilungsfunktion die Kurzzeit-Fourier-Analyse gewählt:

$$S(f, t) = \int_{-\infty}^{+\infty} s(t') \cdot w(t' - t) \cdot e^{-2\pi i f t'} dt' \quad (2.15)$$

bzw. diskretisiert:

$$S(f_k, t_m) = \sum_n s(nT) \cdot w_d(n - \frac{t_m}{T}) \cdot e^{-2\pi i \frac{n k}{N}} \quad (2.16)$$

Das Betragsquadrat von Gl. 2.15 bzw. Gl. 2.16 liefert das sog. Spektrogramm. Zur grafischen Darstellung wird f über t aufgetragen und $|S|^2$ farb- bzw. graustufenkodiert. Die Zeit-Frequenz-Unschärfe zeigt sich dabei in Form von sog. Heisenberg-Ellipsen (siehe Abb. 2.5).

Es sei an dieser Stelle nur kurz auf die Wigner-Ville-Verteilung $WVD(t, f) = \int_{-\infty}^{+\infty} s(t + \frac{\tau}{2}) \cdot s^*(t - \frac{\tau}{2}) \cdot e^{-2\pi i f \tau} d\tau$ hingewiesen, die zwar eine bessere Zeit-Frequenz-Auflösung als das Spektrogramm liefert, aber nicht mehr direkt als Energiedichte interpretierbar ist. Sie liefert zwar für reelle Signale nur reelle Werte, allerdings können diese auch negativ sein. Sie kann außerdem Artefakte liefern, und zwar in Zeit- und Frequenzbereichen, in denen kein Signal vorliegt. Je nach Zielsetzung kann es zwar günstiger sein, die WVD zu benutzen, aus den eben genannten Gründen beschäftigen uns aber im folgenden nur mit der Kurzzeit-Fourier-Analyse.

² f_s : Samplingfrequenz

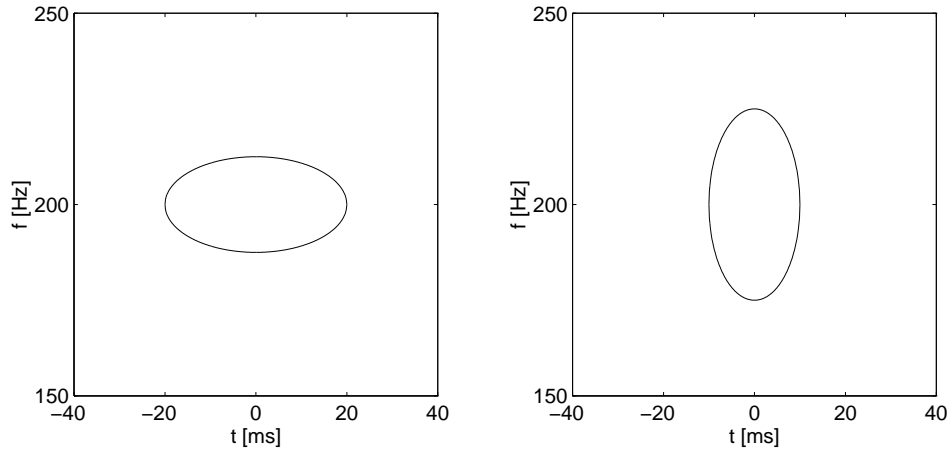


Abbildung 2.5: Heisenberg-Ellipsen bei Fensterlängen von 40 bzw. 20 ms

Werfen wir noch einmal einen Blick auf Gl. 2.15. Da das Fenster symmetrisch ist, gilt:

$$\begin{aligned}
 S(f, t) &= \int_{-\infty}^{+\infty} \underbrace{s(t') \cdot e^{-2\pi i f t'}}_{=: g(t')} \cdot w(t - t') dt' \\
 &= g(t) * w(t) \\
 &= \mathcal{F}^{-1}(G(f) \cdot W(f))
 \end{aligned}$$

Da $W(f)$ möglichst δ -förmig sein soll, ist $S(f, t)$ für ein festes f in Abhängigkeit von t ein Tiefpaßsignal. Es genügt demnach, es mit der doppelten Grenzfrequenz abzutasten. Bei einem Hanning- oder Hammingfenster kann in etwa die erste Nullstelle bei $\frac{2}{M}$ als Grenzfrequenz angesehen werden. Die $S(f, t)$ sind dann mit einer Frequenz

$$F_{Fenster} \geq \frac{4}{M} \cdot f_s \quad (2.17)$$

abzutasten. Das entspricht einem Fenstervorschub von maximal $\frac{1}{4}$ oder einem Überlapp von mindestens $\frac{3}{4}$. Diese Bedingung ist einzuhalten, wenn es gilt, bei sukzessiven Kurzzeit-Fourier-Analysen möglichst wenige Informationen zu verlieren. Gibt man sich aber z.B. damit zufrieden, daß die Grenzfrequenz nur um 10 dB abgeschwächt statt völlig unterdrückt wird, so genügt bereits ein Überlapp von $\frac{1}{2}$ Fenster.

Eine Kurzzeit-Fourier-Transformation der Länge N bei einer Fensterlänge M kann offenbar durch eine Filterbank beschrieben werden, bei der N Bandpaß-Filter einen tiefpaßgefilterten Ausgang $S(f_k, t_m)$ haben.

Wir werden nun zwei verschiedene Möglichkeiten betrachten, das Zeitsignal aus der Kurzzeit-Fourier-Transformierten zu rekonstruieren.

2.5.1 Filterbank-Summation

Ein Ansatz zur Rekonstruktion des Original-Signals ist, zu jedem beliebigen Rekonstruktions-Zeitpunkt $t_m = mT$ die formale Fourier-Rücktransformation zu bilden:

$$\begin{aligned}
s_{\text{rekonstruiert}}(t_m) &:= \frac{1}{N} \sum_{k=0}^{N-1} S(f_k, t_m) \cdot e^{2\pi i \frac{k m}{N}} \\
&= \frac{1}{N} \sum_{k=0}^{N-1} \sum_n s(nT) \cdot w(n - \frac{t_m}{T}) \cdot e^{-2\pi i \frac{n k}{N}} \cdot e^{2\pi i \frac{k m}{N}} \\
&= \frac{1}{N} \sum_n s(nT) \cdot w_d(n - m) \cdot \underbrace{\sum_{k=0}^{N-1} e^{\frac{2\pi i k}{N} (m-n)}}_{=N \cdot \delta_{mn}} \\
&= s(mT) \cdot \underbrace{w_d(0)}_{=1} \\
&= s(t_m)
\end{aligned} \tag{2.18}$$

Ein Nachteil der Filterbank-Summationsmethode ist der hohe Rechenaufwand. Das rekonstruierte Signal soll wieder mit der Original-Samplingfrequenz vorliegen, d.h. für jedes Sample muß eine komplett neue Kurzzeit-Fourier-Analyse durchgeführt werden; das Analyse-Fenster wird immer nur um ein Sample weitergesetzt. Es ist eine möglichst kleine Abtastfrequenz F erwünscht, die die Bedingung 2.17 erfüllt, bei der Filterbank-Summation ist jedoch F maximal, nämlich $F = f_s$. Ein Vorteil dieser hohen Abtastfrequenz ist jedoch, daß sich etwaige spektrale Modifikationen sofort auswirken und sehr präzise zeitlich steuern lassen. Eine weniger rechenintensive Methode, die durch ein knappes Einhalten der Bedingung 2.17 die benötigte Rechenleistung verringert, ist das Overlap-Add-Verfahren.

2.5.2 Kurzzeitspektralanalyse

(von Monika Kordus und Mark Stamminger)

Signale, deren Spektren sich in der Zeit ändern, werden mit der Kurzzeitspektralanalyse bearbeitet. Dazu wird das verschobene Zeitsignal mit einem Fenster $w(t)$ multipliziert und das Produkt wird Fourier transformiert:

$$S(f, t') = \int_{-\infty}^{\infty} s(t + t') w(t) e^{-2\pi i f t} dt \tag{2.19}$$

Da die Fensterfunktion $w(t)$ idealerweise überall Null ist, außer in einem begrenzten Zeitbereich, gleitet dieses Fenster über das Zeitsignal, wenn t' variiert wird, und man bekommt zu jedem t' den momentanen Frequenzgehalt des Signals.

Für die digitalisierte Bearbeitung benutzen wir die DFT:

$$S(f_k, t_m) = \sum_{n=0}^{N-1} s(nT + t_m) w(n) e^{-2\pi i n k / N} \tag{2.20}$$

Für jeden Zeitpunkt t_m bekommen wir dann in einer Spalte des Spektrogrammes den momentanen Frequenzgehalt des Signals. Wir wollen das Spektrogramm anhand eines Beispiels mit MATLAB veranschaulichen. Abbildung 2.6 zeigt den zeitlichen Verlauf einer gesprochenen Ziffer 'Null'. In einem Spektrogramm trägt man die Frequenz auf der senkrechten Achse und die Zeit auf der waagerechten Achse auf. Abbildungen 2.7 und 2.8 sind senkrechte Ausschnitte aus dem Spektrogramm: sie zeigen den Frequenzgehalt des Signals jeweils bei den Zeitpunkten 400ms und 500 ms. Die Abtastfrequenz f_s in diesem Spektrogramm beträgt 16000 Hz und die DFT Länge $N = 256$ gleicht der Fensterlänge M . In Sekunden, ist die Fensterbreite daher 16 ms. Da der Fenstervorschub der halben Fensterlänge beträgt, ergibt sich alle 8 ms ein solches momentanes Frequenzbild des Signals.

Wir haben folgende Variablen:

1. Die Fensterlänge M

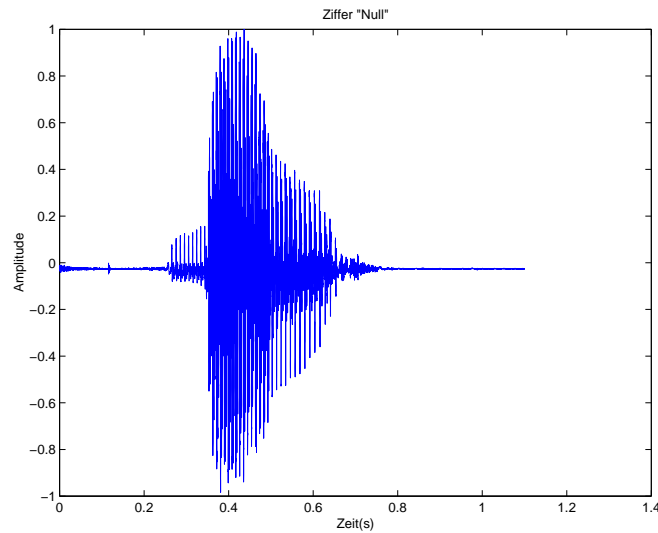
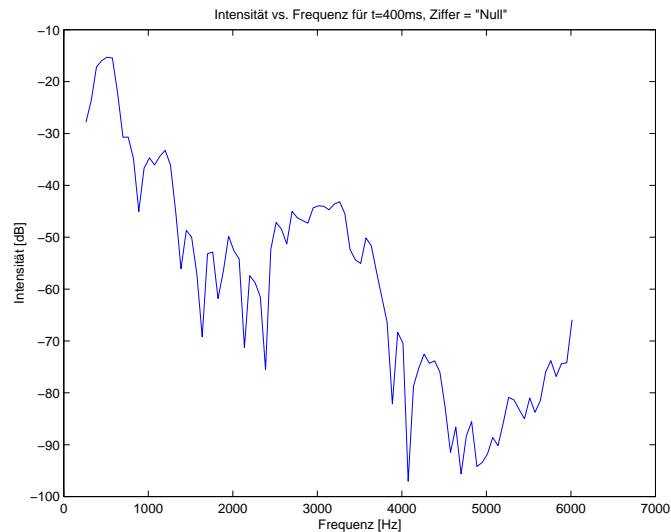


Abbildung 2.6: Gesprochene Ziffer Null

Abbildung 2.7: Frequenzgehalt, $t = 400$ ms, gesprochene Ziffer Null

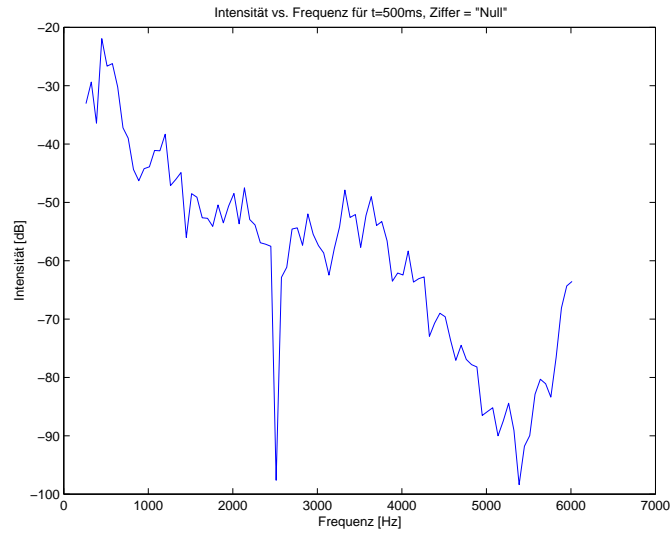
2. Die Länge der DFT N
3. Die Abtastfrequenz $f_s = \frac{1}{T}$, wo T der Zeitabstand zwischen Abtastpunkten ist
4. Die Framerate F

Wir haben dabei folgende Bedingungen für die Beziehungen zwischen Variablen:

1. Die Fensterlänge M ist eine frei wählbare Größe. Ein großer Wert für M bedeutet eine hohe Frequenzauflösung aber eine niedrige Zeitauflösung. Umgekehrt, ein niedriger Wert für M bedeutet eine hohe Zeitauflösung aber eine niedrige Frequenzauflösung.
2. Die Länge der DFT N muß mindestens so groß sein wie die Länge des Fensters M . Wenn das Spektrum gefiltert wird, und anschließend in ein Zeitsignal umgewandelt werden soll, besteht die Möglichkeit, daß dieses Zeitsignal länger als das ursprüngliche Signal wird. Dies wird wie folgt klar:

Das gefilterte Signal S' im Frequenzbereich ist

$$S'(f_k, t_m) = H(f_n)S(f_k, t_m) \quad (2.21)$$

Abbildung 2.8: Frequenzgehalt, $t = 500$ ms, gesprochene Ziffer Null

wobei H die Übertragungsfunktion des Filters ist.

Der Faltungssatz besagt

$$s'(n + t_m) = h(n) * (s(n + t_m)w(n)) \quad (2.22)$$

Das gefilterte Zeitsignal wird also durch die Faltung mit der Impulsantwort $h(n)$ breiter.

Daher kommt das 2. Abtasttheorem:

$$N \Rightarrow M + l(h) \quad (2.23)$$

where $l(h) = \text{Länge von } h(n)$. Für den Fall, daß das Signal nicht gefiltert wird, reicht eine DFT Länge, die genau der Fensterlänge entspricht, also $N = M$.

3. Die dritte Bedingung, die man bei der Kurzzeitspektalanalyse einhalten muß, verknüpft die Framerate F der Analysen, die Abtastrate f_s und die Länge des Fensters M .

In jedem Frequenzkanal des Spektrogrammes hat man ein Tiefpaßsignal. Die obere Grenzfrequenz des Tiefpaßsignals ist für ein Hanning Fenster

$$f_{\text{grenz}} = \frac{2f_s}{M} \quad (2.24)$$

Gemäß dem 1. Abtasttheorem (Nyquist Theorem) muß das Tiefpaßsignal mit einer Frequenz abgetastet werden, die mindestens zweimal so groß ist, wie f_{grenz} .

$$F \Rightarrow \frac{4f_s}{M} \quad (2.25)$$

Der Vorschub des Analysefensters T_{vor} ist der Umkehrwert der Framerate F .

$$T_{\text{vor}} = < \frac{M}{4f_s} \quad (2.26)$$

$$= < \frac{MT_s}{4} \quad (2.27)$$

wo T_s der Zeitabstand zwischen Abtastungen ist. Wir sehen, daß der Vorschub T_{vor} $\frac{1}{4}$ der Fensterbreite nicht überschreiten darf, wenn das Signal anschließend vollständig rekonstruierbar sein soll. Wenn man einen größeren Vorschub nimmt, zum Beispiel $T_{vor} = \frac{MT_s}{2}$, dann hat man die Framerate $F = \frac{2}{MT_s}$. Damit kann man nur Signale vollständig rekonstruieren, die eine obere Frequenz von $f_{grenz} = \frac{1}{MT_s}$ haben. Dies entspricht einer Abschwächung von etwa 10 dB in der Übertragungsfunktion des Filters. Wenn ein größerer Fehler zugelassen werden kann bzw. wenn das Zeitsignal nicht rekonstruiert werden muß, wird oft ein Vorschub von $T_{vor} = \frac{MT_s}{2}$ verwendet.

Literatur

- [1] KOLLMEIER, B.: *Lehrbrief Physikalische Meßtechnik und Digitale Signalverarbeitung*. Universität Oldenburg, 1999

3 Digitale Filter

von Marco Ohm und Hendrik Sroka

3.1 Allgemeine Einführung: Wozu dienen Filter?

Filter werden im Bereich der Signalverarbeitung im großen Maße eingesetzt. Sie dienen dabei verschiedensten Zwecken:

- **Signale trennen:**
Filter können zum Trennen von verschiedenen Signalkomponenten verwendet werden. Dabei ist einer der häufigsten Bereiche das Trennen von Nutz- und Störsignal.
- **Bandbreitenbeschränkung:**
Dieses wird bei Anti-Aliasing-Filtern und Rekonstruktionsfiltern verwendet.
- **Frequenzabhängige Verstärkung:**
Das wohl bekannteste Einsatzgebiet von Filtern. Als Beispiele wären hier der Equalizer bzw. die Loudness-Schaltung in vielen Hifi-Anlagen zu nennen. Ein weiteres Anwendungsgebiet findet sich in der Hörgeräteakustik.

Filter können auf unterschiedlichste Weise verwirklicht werden. Die klassische Variante ist das analoge Filter, welches man nochmals in zwei Unterkategorien einteilt:

- Passive Filter realisiert als Netzwerke aus Widerstand, Kondensator und Spule (RCL-Netzwerke).
- Aktive Filter, die z.B. einen Operationsverstärker als aktives Element verwenden.

Analoge Filter haben einige Beschränkungen, die sich jedoch teilweise durch die Nutzung von digitalen Filtern umgehen lassen:

- Die passiven Bauelemente, aus denen analoge Filter aufgebaut werden, können teilweise sehr teuer werden und einen großen Platzbedarf haben. Spulen z.B. sind teuer in der Produktion und interferieren leicht mit anderen Bauteilen.
- Um temperaturstabile analoge Filter zu konstruieren ist häufig ein großer schaltungstechnischer Aufwand notwendig, da die verwendeten Komponenten alle unterschiedlich auf Temperaturveränderungen reagieren.
- Einige Filter sind nur unter extremen Aufwand oder gar nicht realisierbar. Um Filter mit steilen Übertragungsfunktionen zu konstruieren, ist auch ein wachsender Bauteilaufwand nötig.
- Da die verwendeten Induktivitäten häufig nichtlineare Eigenschaften aufweisen, sind lineare Filter nur schwer realisierbar.

Eine weitere Möglichkeit ist das digitale Filter, das gegenüber dem analogen Filter einige Vorzüge besitzt:

- Es ist flexibler und kann leichter neuen Anforderungen angepaßt werden.
- Digitale Filter sind durch die Anpassung der Filterkoeffizienten leicht so zu konstruieren, daß sie stabil sind. Allerdings muß man hier gewisse Einschränkungen im Verlauf der Übertragungsfunktion in Kauf nehmen.
- Eine Phasenlinearität ist bei digitalen Filtern konstruierbar und eine Dispersion des gefilterten Signals ist im Gegensatz zu analogen Filtern durch Verwendung einer synthetischen Impulsantwort vermeidbar.

Natürlich haben auch digitale Filter ihre Nachteile. Einer der größten ist z.B., daß das Signal durch Rauschen aus mehr Quellen beeinträchtigt werden kann, als dieses bei analogen Filtern der Fall ist. Dazu zählen vor allem Rauschquellen wie das Quantisierungsrauschen und Rechenungenauigkeiten, welche sich auch in Rauschanteilen zeigen.

3.2 LTI-Systeme

Digitale Filter werden im allgemeinen als LTI-Systeme beschrieben. LTI steht dabei als Abkürzung für „Linear Time Invariant“. Alle weiteren Betrachtungen im folgenden Text beziehen sich auf LTI-Systeme. Wendet man einen Filter H auf ein Signal x an so erhält man ein Ausgangssignal y . Für den kontinuierlichen bzw. diskreten Fall läßt sich das wie folgt ausdrücken:

$$\begin{aligned} x(t) &\xrightarrow{H} y(t) \\ x_d(n) &\xrightarrow{H_d} y_d(n) \end{aligned}$$

Der diskrete Fall bedeutet dabei, daß das Signal aus einer Folge von Werten besteht.

Anforderungen an das System:

I) Linearität: $H(\alpha x + \beta y) = \alpha H(x) + \beta H(y)$

II) Zeitinvarianz: $H(x(t)) = y(t) \implies H(x(t - \tau)) = y(t - \tau)$

Ein diskretes Eingangssignal läßt sich als Summe der einzelnen Eingangswerte schreiben. Hierbei ist die δ -Funktion im diskreten Fall als Kronecker-delta zu verstehen; d.h.

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{sonst} \end{cases}$$

$$x(n) = \sum_{k=-\infty}^{\infty} x(k) \underbrace{\delta(n-k)}_{\text{Delta-Funktion}}$$

Jetzt wenden wir einen Filter H auf das Eingangssignal $x(n)$ an und erhalten ein Ausgangssignal $y(n)$.

$$\begin{aligned} H : y(n) &= H(x(n)) \\ &= H\left(\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right) \\ &\stackrel{\text{I)}}{=} \sum_{k=-\infty}^{\infty} x(k) \underbrace{H(\delta(n-k))}_{H(\delta(n))=:h(n)} \\ &\stackrel{\text{II)}}{=} \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ y(n) &= x(n) \star h(n) \end{aligned}$$

Als Resultat erhält man eine Faltung (\star) aus dem Eingangssignal und der sogenannten Impulsantwort $h(n)$.

Daraus folgt, daß LTI-Systeme eindeutig durch die Impulsantwort definiert sind

$$h(t), h_d(n) \longleftrightarrow H$$

Durch Fouriertransformation läßt sich jetzt die Übertragungsfunktion bestimmen:

$$y(t) \circ \bullet Y(f) = X(f)H(f) \iff H(f) = \frac{Y(f)}{X(f)}$$

Die Amplitude ist der Betrag von H , die Phasenverschiebung ist das Argument von H . Das System ist also *eindeutig* durch die Impulsantwort definiert.

Die folgenden zwei Formeln beschreiben den Filter daher schon vollständig

$$\begin{aligned} \longrightarrow y(n) &= x(n) \star h(n) \\ &= \sum_k x(k)h(n-k) \\ H(f) &= \mathcal{F}(h(n)) \end{aligned}$$

Es tritt jedoch ein Problem auf: Die Impulsantwort $h(n)$ kann unendlich lang sein.

Als Beispiel betrachte man die Rechteckfunktion im Frequenzbereich (siehe Abbildung 3.1). Durch die Fouriertransformation erhält man eine *sinc*-Funktion, die im Zeitbereich unendlich ausgedehnt ist. Man benötigt also eine Beschreibungsmöglichkeit, die auch unendliche lange Impulsantworten berücksichtigt bzw. erfaßt.

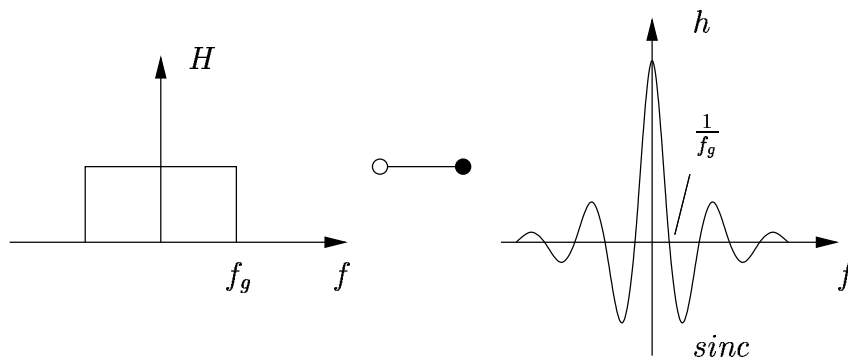


Abbildung 3.1: Ideales Tiefpaßfilter wird durch Fouriertransformation zu einer Sinc-Funktion

Als weiterer wichtiger Punkt wird die *Kausalität* für ein System gefordert. Darunter versteht man, daß das Ausgangssignal nicht von Eingangswerten abhängen darf, die erst später am Eingang vorliegen, das System also quasi in die Zukunft schauen würde. Daraus folgt allerdings, daß die Impulsantwort erst ab der Zeit $n = 0$ existieren darf.

Damit folgt die letzte Anforderung an unser System:

III) Kausalität: $h(n) = 0 \quad \forall n < 0$

Beispiel für ein nicht-kausales System: Die Impulsantwort $h(n)$ des Filters sei bekannt (siehe Abbildung 3.2) und diskret über die gesamte Zeit definiert. Insbesondere gelte: $h(n < 0) \neq 0$. Allgemein gilt:

$$\begin{aligned} y(n) &= x(n) \star h(n) \\ &= \sum_k x(k)h(n-k) \end{aligned}$$

Mit einem δ -Impuls als Eingangssignal ($x(n) = \delta(n)$) folgt:

$$y(n) = h(n)$$

So wie die Impulsantwort oben angenommen wird, stellt sie eine Art „Zeitmaschine“ dar, da sie schon auf Impulse antwortet, die noch gar nicht am Eingang anliegen.

Als Beispiel diene hier ein Tiefpaßfilter, der in analoger Schaltungstechnik als einfaches RC-Glied realisiert wird (Abbildung 3.3). Dieses Beispiel wird später noch einmal als digitales Analogon aufgegriffen.

Die Übertragungsfunktion erhält man auch hier als Quotient von Ausgangs- zu Eingangssignal.

$$H(\omega) = \frac{U_a(\omega)}{U_e(\omega)}$$

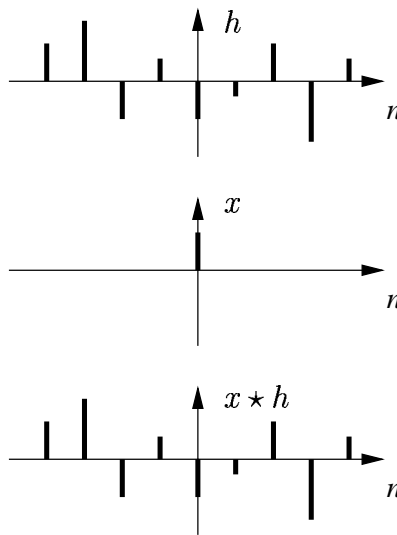
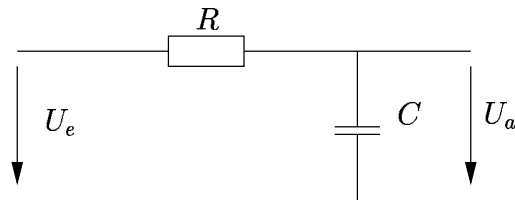
Abbildung 3.2: Faltung einer Impulsantwort h mit einem δ -Impuls.

Abbildung 3.3: Analoger Tiefpaß (RC-Glied)

Nach der Maschenregel gilt für die Eingangs- bzw. Ausgangsspannungen

$$U_e(\omega) \sim R + \frac{1}{i\omega C}$$

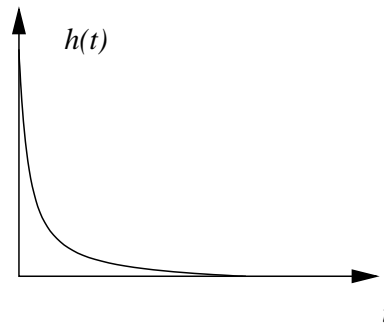
$$U_a(\omega) \sim \frac{1}{i\omega C}$$

Daraus folgt für die Übertragungsfunktion

$$H(\omega) = \frac{U_a(\omega)}{U_e(\omega)} = \frac{\frac{1}{i\omega C}}{R + \frac{1}{i\omega C}} = \frac{1}{1 + i\omega RC}$$

Die Impulsantwort ergibt sich als Integral über die zugehörige Differentialgleichung und ist proportional einer abklingenden e -Funktion (siehe Abbildung 3.4).

$$\Rightarrow h(t) \sim e^{-\frac{t}{RC}}$$

Abbildung 3.4: Impulsantwort als abklingende e -Funktion

3.3 Elementaroperationen und Differenzengleichung

Da digitale Filter ein LTI-System darstellen, folgen sofort einige Elementaroperationen, die man einzeln oder in Kombination als Filter-Funktionen verwenden kann:

$$\begin{array}{llll}
 \text{I)} & x_1(n), x_2(n) & \xrightarrow{\oplus} & x(n) = x_1(n) \oplus x_2(n) & \circ \bullet & X(n) = X_1(n) \oplus X_2(n) \\
 \text{II)} & \alpha, x_1(n) & \xrightarrow{\otimes} & x(n) = \alpha x_1(n) & \circ \bullet & X(n) = \alpha X_1(n) \\
 \text{III)} & x_1(n) & \longrightarrow & x_1(n-1) & \circ \bullet & X(n) = \underbrace{e^{-2\pi i \frac{f}{f_s} n}}_{\text{Phasenfaktor}} X_1(n) \\
 & & & & & = e^{-2\pi i f T} X_1(n)
 \end{array}$$

Die Operation I) beschreibt eine Addition von zwei Signalen, die Operation II) die Multiplikation mit einer Konstanten. Durch die Fourier-Transformation bleiben diese Operationen erhalten. Die Operation III) ist eine Verschiebung im Zeitbereich. Diese Verschiebung schlägt sich nach der Fouriertransformation als Phasenfaktor nieder. Die Herleitung für diesen Faktor befindet sich am Ende dieses Kapitels.

Benutzt man alle drei Elementaroperationen, so läßt sich ein Ausgangssignal $y(n)$ folgendermaßen darstellen:

$$\begin{aligned}
 y(n) = H(x(n)) &= \alpha_0 x(n) + \alpha_1 x(n-1) + \alpha_2 x(n-2) + \dots \\
 &\quad + \beta_1 y(n-1) + \beta_2 y(n-2) + \beta_3 y(n-3) + \dots
 \end{aligned}$$

Bzw. in Summenschreibweise

$$\boxed{y(n) = \sum_{k=0}^M \alpha_k x(n-k) + \sum_{j=1}^N \beta_j y(n-j)} \quad \text{Differenzengleichung} \quad (3.1)$$

Die obige Gleichung stellt eine Art diskretisierte Differentialgleichung dar. Dabei führt die Struktur der Gleichung nicht automatisch zu stabilen Filtern. Dieses ist insbesondere dann der Fall, wenn β -Koeffizienten ungleich null vorhanden sind.

Beispiel für 2 Koeffizienten:

$$\alpha_0 = 1 \text{ und } \beta_1 = 0.9$$

Mit diesen Koeffizienten erhält man die folgende Differenzengleichung:

$$y(n) = x(n) + 0.9y(n-1)$$

Betrachtet man als Eingangssignal eine Delta-Funktion, so folgt als Ausgangssignal direkt die Impulsantwort.

$$x(n) = \delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{sonst} \end{cases}$$

Entwicklung der Ausgangsfunktion

$$\begin{aligned}
 y(0) = h(0) &= 1 + 0 = 1 \\
 h(1) &= 0 + 0.9h(0) = 0.9 \\
 h(2) &= 0.9h(1) = 0.9^2 \\
 h(3) &= 0.9^2h(2) = 0.9^3 \\
 &\vdots \\
 h(n) &= 0.9^n \quad \leftarrow \text{Bew. z.B. durch vollständige Induktion}
 \end{aligned}$$

Für dieses Beispiel konvergiert die Impulsantwort, da für β Werte kleiner 1 gewählt wurden (siehe dazu auch Abbildung 3.5).

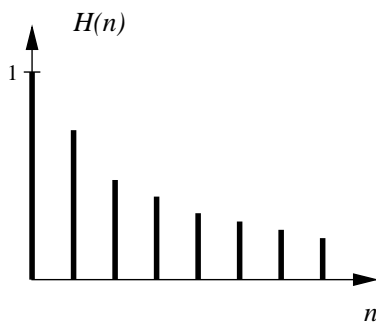


Abbildung 3.5: Impulsantwort für $\beta < 1$ (qualitativ)

Wählt man für β Werte größer 1, so divergiert die Impulsantwort z.B.

$$\beta_1 = 1.1 \implies h(n) = 1.1^n \implies \text{Divergenz}$$

Bei stabilen Filtern muß verlangt werden, daß die Impulsantwort nicht divergiert, wobei jedoch die Impulsantwort nicht in endlicher Zeit abklingen muß. Voraussetzung dafür ist, daß das Betragsquadrat von $x(t)$ quadratintegrabel ist.

$$\int_{-\infty}^{\infty} |x(t)|^2 dt < \infty \implies \lim_{t \rightarrow \infty} x(t) = 0$$

Man kann prinzipiell zwei verschiedene Filtertypen unterscheiden:

- 1) $\forall \beta_i = 0 \implies$ nicht rekursives Filter (FIR - Finite Impulse Response),
d.h. die Impulsantwort $h(x)$ ist finit (endlich lang).
- 2) $\exists \beta_i \neq 0 \implies$ rekursives Filter (IIR - Infinite Impulse Response),
d.h. die Impulsantwort $h(x)$ ist infinit (unendlich lang).

Bemerkung: Mit einem IIR-Filter läßt sich die Ordnung eines FIR-Filters stark reduzieren, allerdings besteht hier die Gefahr der Instabilität.

Abbildung 3.6 visualisiert die Differenzengleichung als Direkte Form I. Die „T“-Kästen kennzeichnen dabei die diskretisierten Zeitintervalle. Diese reichen wie ein Schieberegister die Eingangswerte in ihrer Kette weiter. Die jeweiligen Zwischenwerte werden mit jeweiligen α - bzw. β -Koeffizienten verknüpft und aufaddiert. An den β -Koeffizienten erkennt man hierbei auch deutlich die Rekursion.

Zum Unterpunkt III) soll nun noch kurz darauf eingegangen werden, wie man den Phasenfaktor erhält:

$$x_1(n) \longrightarrow x_1(n-1) = x_1(t-\tau) \circ \bullet \mathcal{F}(x_1(t-\tau))$$

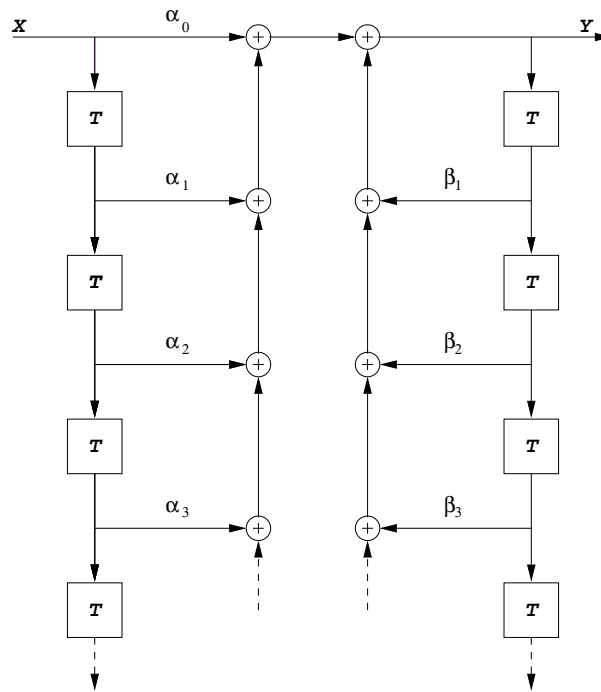


Abbildung 3.6: Direkte Form I

$$\begin{aligned}
 \mathcal{F}(x_1(t - \tau)) &= \int_{-\infty}^{\infty} x_1(t - \tau) e^{-2\pi i f t} dt \\
 &= e^{-2\pi i f \tau} \cdot \int_{-\infty}^{\infty} x_1(t - \tau) e^{-2\pi i f (t - \tau)} dt \\
 &= e^{-2\pi i f \tau} \cdot \int_{-\infty}^{\infty} x_1(t) e^{-2\pi i f t} dt \\
 &= e^{-2\pi i f \tau} \cdot \mathcal{F}(x_1(n))
 \end{aligned}
 \quad \square$$

3.4 Übertragungsfunktion der Differenzgleichung

Durch eine Fouriertransformation der Differenzgleichung (3.1) und anschließendes Umformen erhält man die Übertragungsfunktion $H(f)$ für das Filter.

$$\begin{aligned}
 Y(f) &= \sum_{k=0}^M \alpha_k e^{-2\pi i \frac{f}{f_s} k} X(f) + \sum_{j=1}^N \beta_j e^{-2\pi i \frac{f}{f_s} j} Y(f) \\
 Y(f) - Y(f) \sum_{j=1}^N \beta_j e^{-2\pi i \frac{f}{f_s} j} &= X(f) \sum_{k=0}^M \alpha_k e^{-2\pi i \frac{f}{f_s} k} \\
 H(f) = \frac{Y(f)}{X(f)} &= \frac{\sum_{k=0}^M \alpha_k e^{-2\pi i \frac{f}{f_s} k}}{1 - \sum_{j=1}^N \beta_j e^{-2\pi i \frac{f}{f_s} j}}
 \end{aligned}$$

H ist in f periodisch, da sich H auch als

$$H = H\left(e^{2\pi i \frac{f}{f_s}}\right)$$

ausdrücken läßt. Dies führt zur z -Transformation.

3.5 Die z -Transformation

Die z -Transformation kann zunächst als Umformung der Fouriertransformation verstanden werden mit der Variablensubstitution

$$z = e^{2\pi i f T} = e^{2\pi i \frac{f}{f_s}} \quad \text{z-Transformation}$$

$$\Rightarrow H(f) = H(z) = \frac{\sum_{k=0}^M \alpha_k z^{-k}}{1 - \sum_{j=1}^N \beta_j z^{-j}}$$

Durch die z -Transformation bildet man die eindimensionale Frequenzachse auf den komplexen Einheitskreis ab. Um flexibler zu sein, wird jedoch der Definitionsbereich auf die komplexe Ebene ausgedehnt, was eine Vereinfachung der Übertragungsfunktion schafft.

Dabei ist die Fouriertransformation jedoch nur auf dem Einheitskreis definiert (siehe Abbildung 3.7).

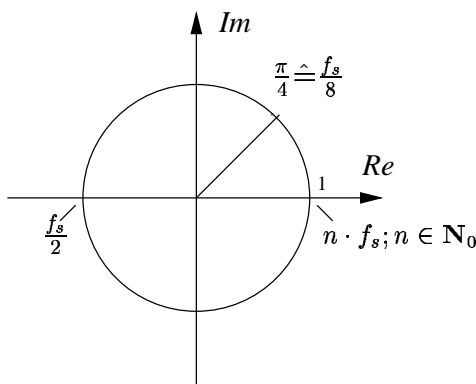


Abbildung 3.7: z -Transformierte

Beispiel: $\alpha_0 = 1, \beta_1 = 0,9 \Rightarrow h(n) = \beta_1^n$

$$H(z) = \frac{1}{1 - \beta_1 z^{-1}}$$

Ein weiterer entscheidender Vorteil der komplexen Zahlenebene ist, daß man die Methoden der Funktionentheorie anwenden kann.

In der Physik geht man häufig von reellen Größen zu komplexen Größen über, um auch die Phaseninformationen zu berücksichtigen. Dieses wird z.B. in der Optik und der Elektrotechnik angewendet. Auf die reellen Größen schließt man, indem man den Betrag und das Argument der komplexen Größen bestimmt.

3.6 Pol- und Nullstellenzerlegung

Die Übertragungsfunktion ist in dieser Darstellung ein Quotient zweier Polynome

$$H(z) = \frac{\sum_{k=0}^M \alpha_k z^{-k}}{1 - \sum_{j=1}^N \beta_j z^{-j}} \quad z \in \mathbb{C}$$

Ein Polynom k -ten Grades hat in der komplexen Darstellung auch k Nullstellen. Aufgrund des Hauptsatzes der Algebra lässt sich dieses Polynom somit auch in der Produktdarstellung schreiben. Die z_{0k} stellen dabei die k Nullstellen des Zähler-Polynoms und die z_{0j} die j Nullstellen des Nenner-Polynoms dar. Man erhält hiermit folgende Darstellung der Übertragungsfunktion.

$$H(z) = \alpha_0 z^{N-M} \frac{\prod_{k=1}^M (z - z_{0k})}{\prod_{j=1}^N (z - z_{0j})} \quad (3.2)$$

Diese Darstellung ist die Pol-Nullstellenzerlegung und wird in Abbildung 3.8 veranschaulicht.

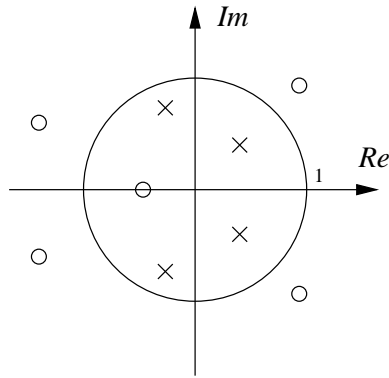


Abbildung 3.8: Pole (x) und Nullstellen (o) in der z-Ebene

3.7 Definitionsbereich von $H(z)$

Aus der Funktionentheorie ist bekannt, daß der Konvergenzradius einer Funktion wie $H(z)$ durch den Abstand der äußersten Polstelle gegeben ist. Außerhalb dieses Radius herrscht Konvergenz. Liegen alle Pole innerhalb des Einheitskreises, so befindet sich auch die Frequenzachse, die durch die z -Transformation auf den komplexen Einheitskreis abgebildet wird, im Konvergenzgebiet. Der Filter ist somit konvergent, also stabil (siehe Abbildung 3.9).

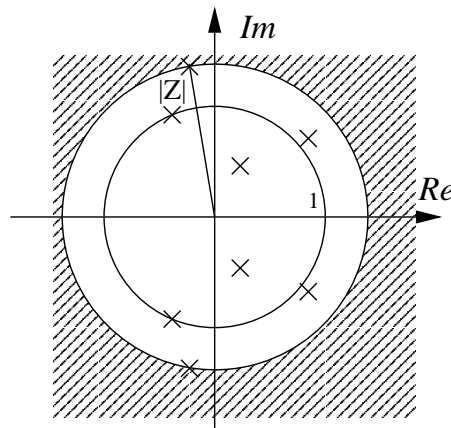


Abbildung 3.9: Konvergenzradius $R \geq |z|$ für den am weitest entfernten Pol

Beispiel (siehe dazu auch Abbildung 3.10):

$$H(z) = \frac{1}{1 - 0.9z^{-1}} = \frac{z}{z - 0.9} \Rightarrow \text{Nullstelle bei } z = 0, \text{ Pol bei } z = 0.9$$

\Rightarrow Konvergenz, da der Pol innerhalb des Einheitskreises liegt.

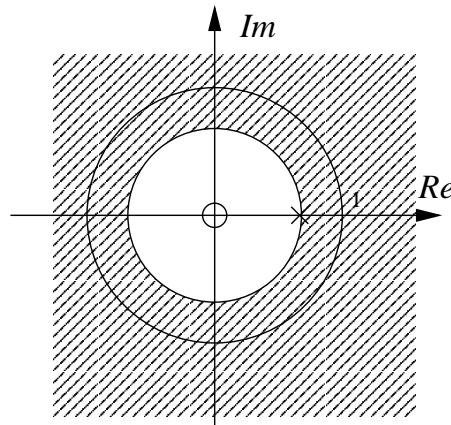


Abbildung 3.10: Konvergenzbereich für den Filter mit einem Pol ($z=0.9$) und einer Nullstelle ($z=0$)

Es stellt sich jetzt die Frage, wie die Übertragungsfunktion auf dem Einheitskreis von der Lage der Pole und Nullstellen abhängt. Allgemein läßt sich sagen, daß in der Nähe von Polen die Übertragungsfunktion erhöht wird und es in der Nähe von Nullstellen Einbrüche gibt. Siehe dazu Abbildungen 3.11 und 3.12.

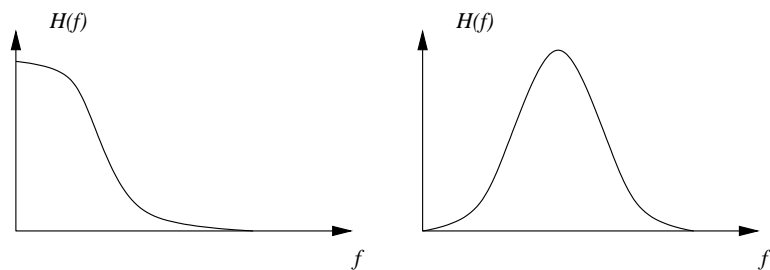


Abbildung 3.11: Verschiebt man den Pol auf dem Kreis mit Radius 0.9, so wandert das Maximum der Übertragungsfunktion zu höheren Frequenzen.

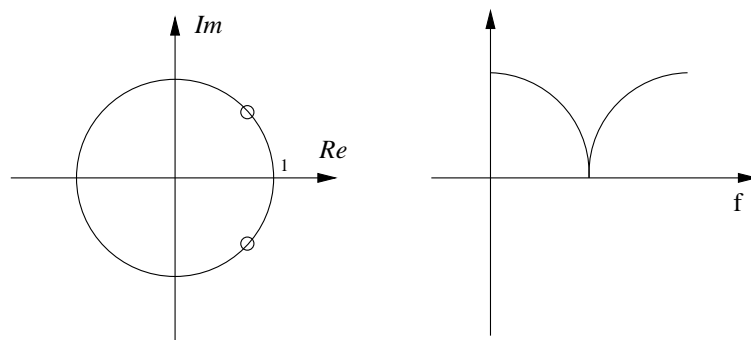


Abbildung 3.12: Nullstelle auf dem Einheitskreis

Zur Verdeutlichung macht man häufig eine geometrische Interpretation (siehe hierzu Abbildung 3.13):

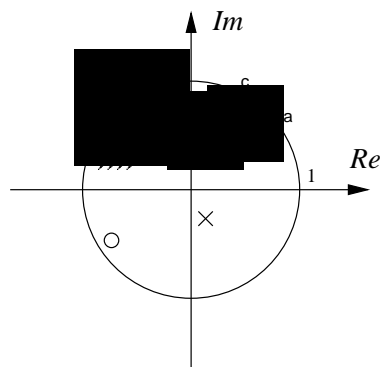


Abbildung 3.13: Geometrische Interpretation des Abstandes zwischen z und Nullstellen bzw. Polen

Betrachtet man Gleichung 3.2 auf Seite 32, so tauchen z.B. im Zähler die Produkte des Abstandes zwischen z und der jeweiligen Nullstelle auf. Ist die Frequenz dicht an der Nullstelle (z an der Position a in Abbildung 3.13), so ist der Einfluß der Nullstelle groß, da der Abstand klein und somit die Differenz des Zählers in Gleichung 3.2 ebenfalls klein ist. Entfernt man sich weiter (z.B. z an der Position d bzw. b), so wird der Einfluß der Nullstelle auf die Übertragungsfunktion geringer bei dieser Frequenz, da der Abstand größer wird.

Ein ähnliches Verhalten ergibt sich bei den Polstellen. Da hier aber der Abstand der Polstellen von der Frequenz z auf dem Einheitskreis im Nenner von Gleichung 3.2 auftritt, ist das Verhalten genau umgekehrt: Liegt ein Pol in der Nähe von z , so ist der Abstand klein und somit auch der Nenner klein. Man erhält also eine große Amplitude der Übertragungsfunktion.

Liegt ein Pol oder eine Nullstelle in der Nähe des Mittelpunktes des Einheitskreises, so ist der Einfluß auf alle Frequenzen nahezu gleich groß, da sich der Abstand kaum ändert (siehe Punkte *a* und *c* in Abbildung 3.13).

Die Koeffizienten α_k und β_j müssen alle reell sein. Daraus folgt, daß alle Pole und Nullstellen als komplex-konjugierte Paare auftreten müssen. Nur wenn die Pole und Nullstellen auf der reellen Achse liegen, kommen sie nicht als Paare vor.

Dieses läßt sich an einer einfachen Rechnung verdeutlichen. Man betrachte den Zähler bzw. den Nenner von Gleichung 3.2 (Seite 32):

$$\prod_{j=1}^N (z - z_{0j}) = (z - z_{01})(z - z_{02})(z - z_{03})(z - z_{04}) \cdots (z - z_{0N}) \quad (3.3)$$

$$= [(z - z_{01})(z - z_{01}^*)][(z - z_{03})(z - z_{03}^*)] \cdots \quad (3.4)$$

$$z := x + iy$$

$$= [(x + iy) - (x_{01} + iy_{01})][(x + iy) - (x_{01} - iy_{01})] \cdot [(z - z_{03})(z - z_{03}^*)] \cdots \quad (3.5)$$

$$= [(x + iy)^2 - (x + iy)(x_{01} - iy_{01}) - (x + iy)(x_{01} + iy_{01}) + (x_{01} + iy_{01})(x_{01} - iy_{01})][(z - z_{03})(z - z_{03}^*)] \cdots \quad (3.6)$$

$$= \left[(x + iy)^2 - (x + iy)(x_{01} - iy_{01} + x_{01} + iy_{01}) + x_{01}^2 + y_{01}^2 \right] \cdot [(z - z_{03})(z - z_{03}^*)] \cdots \quad (3.7)$$

$$= [z^2 - z(2 \cdot x_{01}) + x_{01}^2 + y_{01}^2][(z - z_{03})(z - z_{03}^*)] \cdots \quad (3.8)$$

$$= [z^2 - z(2 \cdot x_{01}) + x_{01}^2 + y_{01}^2][z^2 - z(2 \cdot x_{03}) + x_{03}^2 + y_{03}^2] \cdots \quad (3.9)$$

Zunächst wird das Produkt in Gleichung 3.3 ausgeführt. Wir setzen nun komplex-konjugierte Paare für Pole bzw. Nullstellen voraus. Somit ergibt sich die Darstellung in Gleichung 3.4. Da z eine komplexe Zahl ist, wird sie jetzt als algebraische Summe von Real- und Imaginärteil substituiert. Diese Substitution ist in Gleichung 3.5 zu sehen, wobei die Umformung nur für das erste Produkt ausgeführt wird; alle anderen Terme sind entsprechend zu behandeln. In den Gleichungen 3.6 und 3.7 wird das erste Produkt ausmultipliziert und neu zusammengefaßt. Dieses ergibt nach einer Resubstitution Gleichung 3.8, an der zu sehen ist, daß in ihr keine komplexen Koeffizienten vorhanden sind. Da sich dieses für jeden Produktterm entsprechend zerlegen läßt, sind auch bei Polynomen höherer Ordnung alle Koeffizienten reell, wenn die Pol- und Nullstellen in komplex-konjugierten Paaren auftreten.

3.8 Implementation der Differenzengleichung auf dem Rechner

Aus der Direkten Form I (siehe Abbildung 3.14) folgt durch lineare Umformung die Direkte Form II (Abbildung 3.15). Man macht sich dabei die Tatsache zunutze, daß die Differenzengleichung wegen ihrer Kommutativität in beliebige Teilsummen zerlegt werden kann.

$$\begin{aligned} y(n) &= \sum_{k=0}^M \alpha_k x(n-k) + \sum_{j=1}^N \beta_j y(n-j) \\ &= \sum_{j=1}^N \beta_j y(n-j) + \sum_{k=0}^M \alpha_k x(n-k) \end{aligned}$$

Anders ausgedrückt: Im Frequenzbereich entspricht dieses einer Vertauschung der Multiplikationsreihenfolge:

$$\begin{aligned} H(z) &= \left(\sum_{k=0}^M \alpha_k z^{-k} \right) \cdot \frac{1}{1 - \sum_{j=1}^N \beta_j z^{-j}} \\ &= \frac{1}{1 - \sum_{j=1}^N \beta_j z^{-j}} \cdot \left(\sum_{k=0}^M \alpha_k z^{-k} \right) \end{aligned}$$

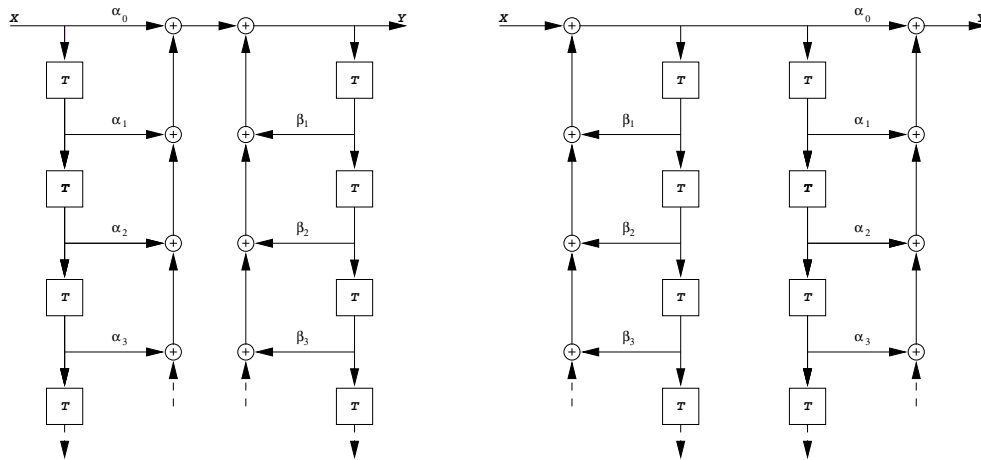


Abbildung 3.14: Kommutation der Differenzengleichung von Direkte Form I

Die Direkte Form II bietet den Vorteil, mit weniger Speicher auszukommen und gleichzeitig etwas weniger Rechenzeit zu verbrauchen, da nur eine Speicherkette existiert, die verschoben werden muß. Der „Nachteil“ ist eine höhere Abstraktion des Filtervorgangs, da ein Zwischensignal z im Speicher existiert, das keinen realen Bezug zum Signal besitzt.

Gewöhnlich sind Filter hintereinandergeschaltet, also seriell angeordnet. Durch eine Partialbruchzerlegung der Übertragungsfunktion kommt man zu einer parallelen Anordnung. Bei der seriellen Anordnung (siehe Abbildung 3.16) potenzieren sich die Fehler, d.h. ein Fehler, der im Filter H_1 entsteht (z.B. Rauschen) wird in Filter H_2 weiterverarbeitet. Dieses ist bei der parallelen Anordnung nicht der Fall. Hier werden die Fehler erst am Ende summiert und beeinflussen sich nicht gegenseitig. Der Nachteil von der parallelen Anordnung ist, daß man mehr Speicher braucht.

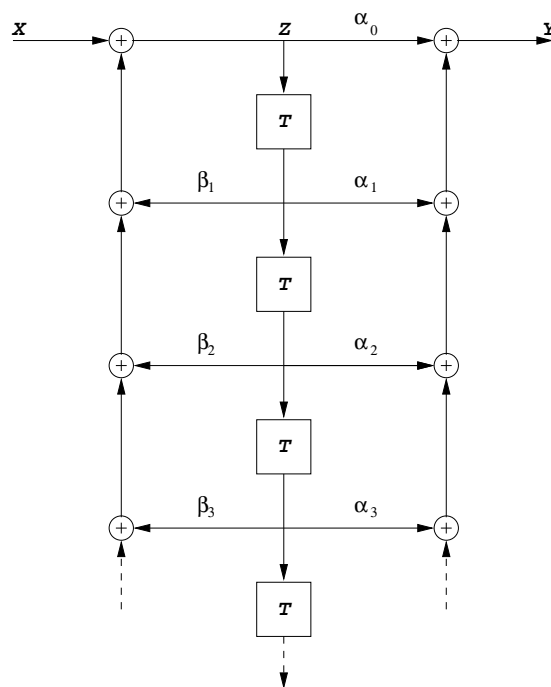


Abbildung 3.15: Direkte Form II

Serielle Anordnung:

$$H = H_1 \cdot H_2$$



Parallele Anordnung:

$$H = H_1 + H_2$$

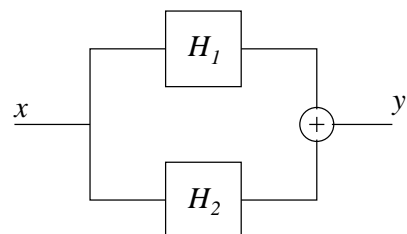


Abbildung 3.16: Filter in serieller und paralleler Anordnung

3.9 Schlußbemerkungen

3.9.1 Verringerung des Rechenaufwands

Neben der linearen Umformung von oben besteht eine weitere Möglichkeit den Rechenaufwand zu verringern darin, indem man eine Rückkopplung vornimmt. Dabei handelt es sich um ein grundsätzliches Design-Kriterium, bei dem man schon ausgerechnete Werte ein weiteres Mal verwendet und man sich so die Arbeit spart, den gleichen Rechenschritt immer wieder auszuführen. Dieses ist jedoch nur bei IIR-Filtern anwendbar, da bei FIR-Filtern das Ausgangssignal nur direkt vom Eingangssignal abhängt und somit keine Rückkopplung existiert.

3.9.2 Vergleich von Polen und Nullstellen bezüglich des Rechenaufwands

Man braucht viel mehr Nullstellen um gleiche Filtereigenschaften zu erzeugen, als wenn man Pole verwenden würde. Pole erzeugen wesentlich steilere Übertragungsfunktionen als Nullstellen.

Bei einem IIR-Filter braucht man wesentlich weniger Ordnungen, um sehr lange Impulsantworten zu erzeugen. Im Prinzip reicht schon ein IIR-Filter 1. Ordnung aus, um eine unendlich lange Impulsantwort zu erhalten. Selbst

in dem Fall, daß der rekursive Teil nach einer endlichen Zeit keinen Beitrag mehr liefert, braucht man bei einem FIR-Filter trotzdem mehr Ordnungen, um eine ähnliche Übertragungsfunktion zu erzeugen.

Ein Filter mit Nullstellen in der Nähe des Einheitskreises (wie zuvor schon einmal dargestellt) würde man aufgrund der Eigenschaften des Gehörs gar nicht heraushören. Diese Eigenschaft des Gehörs ist notwendig, da auch in normalen Räumen „Nullstellen“ durch Reflexionen an den Raumwänden und Gegenständen auftreten (siehe Abbildung 3.17).

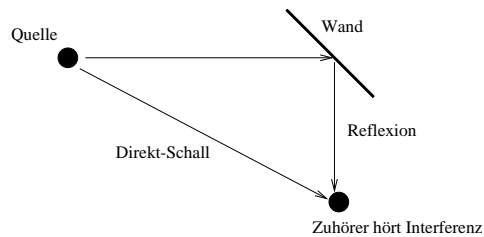


Abbildung 3.17: Quelle, Reflexion und Zuhörer

Würde man solche Filtereigenschaften von Räumen heraushören, das Gehör also linear arbeiten, so würde sich der Klangeindruck extrem verändern, wenn sich die Schallquelle oder der Zuhörer bewegt.

4 Entwurf Digitaler Filter

von Rainer Beutelmann und Daniel Carl

4.1 Einleitung

Die Aufgabe eines digitalen Filters ist es, eine vorgegebene Übertragungsfunktion zu approximieren. Bisher wurde nur der Fall betrachtet, daß Pole und Nullstellen der Differenzengleichung, und damit die Gleichung selbst, gegeben sind. Daraus ließen sich Übertragungsfunktion $H(f)$ und Impulsantwort $h(t)$ berechnen. Das Design eines digitalen Filters stellt gerade die Umkehrung dieses Problems dar. Gesucht sind Pole und Nullstellen der Differenzengleichung zu einer vorgegebenen Übertragungsfunktion oder Impulsantwort. Dies ist in der Praxis immer dann wichtig, wenn digital abgespeicherte Daten auf eine bestimmte Art gefiltert werden sollen [1]. Dabei wird die gewünschte Filterfunktion innerhalb bestimmter Toleranzgrenzen vorgegeben, was in (Abb. 4.1) veranschaulicht ist.

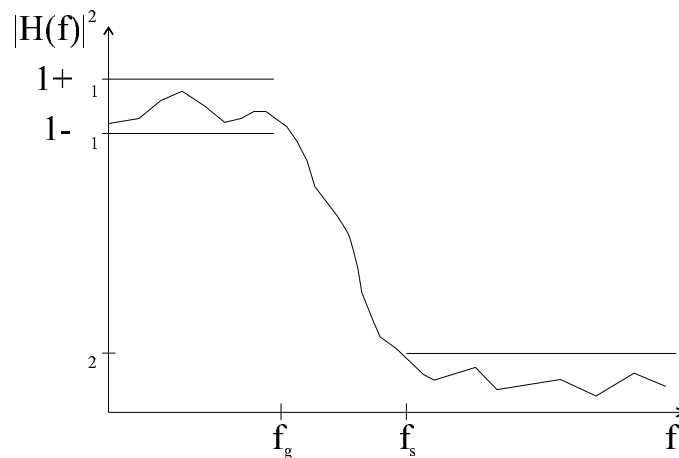


Abbildung 4.1: Toleranzbereich der Übertragungsfunktion

Die Frequenzachse in (Abb. 4.1) wird in drei wesentliche Bereiche unterteilt:

- Den Durchlaßbereich $0 < f < f_g$. In diesem Bereich darf der Betrag der Übertragungsfunktion um den zuvor festgelegten Toleranzparameter δ_1 schwanken. Die Grenzfrequenz f_g gibt an, wann $|H(f)|^2$ um $3dB$ abgefallen ist.
- Den Übergangsbereich $f_g < f < f_s$. In diesem Bereich sinkt der Betrag der Übertragungsfunktion stetig rasch ab. Die Breite dieses Bereiches kann aufgrund der Kausalität nicht zu Null werden.
- Das Stopppband $f_s < f$. In diesem Bereich muß $|H(f)|$ unter einem zweiten zuvor definierten Toleranzparameter δ_2 bleiben, und darf demnach auch nur um maximal diesen Wert schwanken.

In der Regel werden also der Betrag der Übertragungsfunktion im Durchlaß- und im Sperrbereich mit gewissen Toleranzen und ein Frequenzbereich des Übergangs zwischen diesen beiden Bereichen vorgegeben.

Die Frage ist nun, wieviele Pole und Nullstellen benötigt werden und wo sie in der z -Ebene liegen müssen, um die gegebenen Bedingungen möglichst gut zu erfüllen.

Man geht nun nach den folgenden Prinzipien vor:

1. Umsetzung analoger in digitale Filter \implies IIR-Filter
 - Impulsantwort invariant
 - Frequenzgang invariant
2. Optimierung eines digitalen Filters \implies FIR-Filter
 - Windowing Design
 - Frequenzabtastfilter
 - Gradientenverfahren

Ein IIR-Filter hat vor allem den Vorteil eines geringen Implementierungsaufwandes bei hoher Filterwirkung durch die direkte Übertragung "klassischen" Filterdesigns. Nachteilig ist bei einem IIR-Filter, daß er ggf. numerisch instabil ist, vor allem, wenn die Pole zu dicht am Einheitskreis liegen. Durch Rechnerungenauigkeiten bei den Filterkoeffizienten und den zu verarbeitenden digitalen Daten können hier exponentiell anwachsende Impulsantworten (instabil) oder Grenzzyklen auftreten. Außerdem kann ein IIR-Filter nicht phasenlinear sein, d.h. es tritt Dispersion auf.

Ein FIR-Filter ist dagegen numerisch stabil und bei symmetrischer Impulsantwort phasenlinear.

Die Filterfunktion eines FIR-Filters hat keine rekursiven Anteile, also nur Nullstellen und keine Pole. Die Filterwirkung, die durch Nullstellen erreicht wird, ist geringer, als die durch Pole. Dies hat einen wesentlich größeren Implementierungsaufwand zur Folge, d.h. es werden mehr Koeffizienten benötigt, was natürlich ein Nachteil gegenüber dem IIR-Filter darstellt.

4.2 Umsetzung analoger in digitale Filter

Bei dieser Art des Filterdesigns macht man sich die Kenntnisse zunutze, die man über das Design analoger Filter hat. Zum Verständnis der folgenden Überlegungen zum impulsantwortinvarianten und frequenzganginvarianten Design sind einige Grundkenntnisse über analoge Filter wichtig. Diese werden in Kapitel 4.2.1 vermittelt.

4.2.1 Einschub: Analoge Filter

Analoge Filter sind *LTI-Systeme*, und damit durch ihre Impulsantwort bzw. Übertragungsfunktion vollständig charakterisiert.

$$h(t) \circ \longrightarrow H(f) = \int_{-\infty}^{+\infty} h(t) \cdot e^{2\pi i f t} dt \quad (4.1)$$

Führt man jetzt eine Generalisierung der Koordinaten durch $s = \sigma + i\omega = \sigma + i2\pi f$, erweitert man damit den Definitionsbereich der Filterfunktion $H(f)$ auf die ganze komplexe Ebene. Außerdem beschränkt man sich auf kausale Systeme, d.h. $h(t) = 0$ für $t < 0$. Man nennt die in Gleichung (4.2) dargestellte Transformation dann nicht mehr Fouriertransformation sondern Laplacetransformation. Den Definitionsbereich der Laplacetransformierten wird *s-Ebene* genannt.

$$H(s) = \int_0^{+\infty} h(t) \cdot e^{st} dt \quad (4.2)$$

Das analoge Filter läßt sich jetzt durch eine rationale Funktion in der *s-Ebene* beschreiben. Jede rationale Funktion wiederum läßt sich eindeutig durch die Nullstellen des Zähler- und Nennerpolynoms darstellen.

$$H(s) = \frac{\prod_{k=1}^M (s - s_{0k})}{\prod_{j=1}^N (s - s_{pj})} \quad (4.3)$$

Bsp.1: RC-Glied

Die Übertragungsfunktion des RC-Gliedes (analoger Filter) ist als Gleichung (4.4) bekannt.

$$H(f) = \frac{1}{1 + i2\pi fRC} \quad (4.4)$$

Die Laplacetransformation liefert:

$$H(s) = \frac{1}{1 + sRC} = \frac{1}{(s + \frac{1}{RC})} \quad (4.5)$$

Die durch die Laplacetransformation erhaltene Übertragungsfunktion besitzt keine Nullstellen aber einen Pol bei $s = -\frac{1}{RC}$. Die Angabe des Poles in der Laplace-Ebene beschreibt die Filterfunktion des RC-Gliedes vollständig.

Bsp.2: Butterworthfilter 2. Ordnung

Ein Butterworth-Tiefpaß mit n energiespeichernden Elementen (Induktivitäten und Kapazitäten) hat die Übertragungsfunktion [2]:

$$|H(f)| = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_0}\right)^{2n}}} \quad (4.6)$$

Für dieses Tiefpaßfilter ($N = 2$) erhält man durch Laplacetransformation die Übertragungsfunktion (4.7), die zwei Pole besitzt. Die Pole sind zueinander komplex konjugiert und liegen in der linken Halbebene auf dem Einheitskreis. Das Filter ist also stabil. Das Stabilitätskriterium für analoge Filter besagt, daß alle Pole in der linken Halbebene der s -Ebene liegen müssen.

$$H(s) = \frac{1}{(s - s_{p1}) \cdot (s - s_{p2})} \quad (4.7)$$

$$s_{p1/2} = -\frac{\omega_0}{\sqrt{2}} \pm i \frac{\omega_0}{\sqrt{2}}$$

4.2.2 Die z-Transformation

Es sei daran erinnert das sich die Übertragungsfunktion eines *LTI Filters* durch seine Filterkoeffizienten α_i und β_k wie in Gleichung (4.8) darstellen läßt.

$$H(f) = \frac{Y(f)}{X(f)} = \frac{\sum_{k=0}^M \alpha_k e^{-i2\pi \frac{f}{f_s} k}}{1 - \sum_{j=0}^N \beta_j e^{-i2\pi \frac{f}{f_s} j}} \quad (4.8)$$

In Gleichung (4.8) ist zu sehen, daß die Übertragungsfunktion eine Funktion von $H(e^{-i2\pi \frac{f}{f_s}})$ ist, also periodisch mit der Abtastfrequenz f_s . Die Periodizität ist eine Folge der Abtastung des Signales im Zeitbereich. Ein diskretes Signal besitzt immer ein periodisches Spektrum. Führt man in Gleichung (4.8) die Ersetzung $z := e^{-i2\pi \frac{f}{f_s}}$ durch, kann man die Übertragungsfunktion als Funktion von z schreiben.

$$H(z) = \frac{\sum_{k=0}^M \alpha_k z^{-k}}{1 - \sum_{j=0}^N \beta_j z^{-j}} \quad (4.9)$$

Diese Transformation kann man auf die Fouriertransformation übertragen. So erhält man die z -Transformierte der δ -Impulsantwort. Diese Darstellung der Übertragungsfunktion hat sich für die Beschreibung digitaler Filter als sehr praktisch erwiesen.

$$H(z) = \sum_{k=-\infty}^{+\infty} h(k) \cdot z^{-k} \quad (4.10)$$

Der Definitionsbereich von z wird auf die ganze komplexe Ebene erweitert, da so eine bessere mathematische Beschreibung durch die Funktionentheorie möglich ist. Der Frequenzachse entspricht in der z -Ebene der Einheitskreis.

Für die Übertragung eines analogen Filters in ein digitales Filter stellt sich die Aufgabe eine *konforme Abbildung*³ von der s -Ebene in die z -Ebene $H(s) \rightarrow H(z)$ zu finden, so daß die imaginäre Achse auf den Einheitskreis abgebildet wird. Außerdem muß die linke Halbebene der s -Ebene in den Einheitskreis abgebildet werden. Ein analoges Filter ist stabil, wenn seine Pole in der linken Halbebene liegen. Bei digitalen Filtern gilt demnach als Stabilitätskriterium, daß die Pole im Einheitskreis liegen müssen. Für eine solche Art der Abbildung gibt es mehrere Möglichkeiten, von denen im Folgenden zwei vorgestellt werden.

4.2.3 Impulsantwortinvariantes Design

Wie der Name schon vermuten läßt, wird bei dieser Art des digitalen Filterdesign die Impulsantwort des analogen Filters “übernommen“. Bildet man die Übertragungsfunktion des analogen Filters $H(s)$ mit der konformen Transformation (4.11) in die z -Ebene ab, so wird die imaginäre Achse auf den Einheitskreis und die linke Halbebene in den Einheitskreis abgebildet.

$$z = e^{\frac{s}{f_s}} = e^{sT} \quad (4.11)$$

Allerdings müssen dabei einige Dinge beachtet werden. Schaut man sich die Transformation (4.11) genauer an, so stellt man fest, daß die linke Halbebene in horizontalen Streifen periodisch in den Einheitskreis abgebildet wird. Erst einmal wird der Streifen $-i\pi f_s < f < i\pi f_s$ in den Einheitskreis abgebildet. Alle oberhalb und unterhalb dieses Streifens liegenden Frequenzbereiche werden aber ebenfalls in Intervallen von $2\pi T$ additiv in den Einheitskreis abgebildet, d.h. sie überlagern sich dort.

Um diesen *Aliasing-Effekt* zu vermeiden, muß folgende Bedingung an $H(s)$ gestellt werden: Die Übertragungsfunktion muß im Bereich oberhalb der halben Samplingfrequenz weit genug abgefallen sein ($|H(s)| \approx 0$ für $|f| \geq \frac{1}{2}f_s$). Auf gar keinen Fall dürfen Pole und Nullstellen in diesem Frequenzbereich auftreten, da diese ebenfalls in den unteren Frequenzbereich abgebildet werden würden. In (Abb. 4.2) ist der *Aliasing-Effekt* veranschaulicht. Im Frequenzbereich $-i\pi f_s < f < i\pi f_s$ befinden sich zwei Pole. Zusätzlich zu diesen beiden Polen werden die beiden Pole und Nullstellen außerhalb des schraffierten Bereiches auch in den Einheitskreis abgebildet. Das so entstandene digitale Filter bekommt dadurch natürlich völlig andere Eigenschaften als das zugrundegelegte analoge Filter.

Impulsinvariantes Filterdesign kann also nur auf bandbegrenzte analoge Filter angewendet werden.

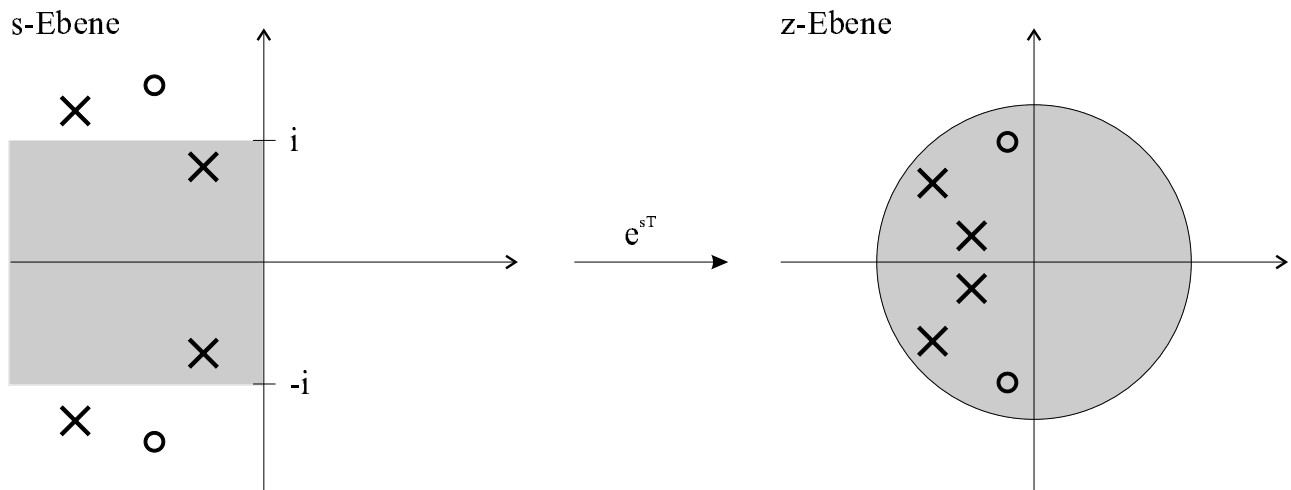


Abbildung 4.2: Aliasing bei impulsinvariantem Filterdesign

Wenden wir nun einmal die z -Transformation auf den in Kapitel 4.2.1 Bsp.1 beschriebenen RC-Tiefpaß an. Die Laplacetransformierte dieses analogen Filters ist in Gleichung (4.5) gegeben. Die z -Transformation liefert für das

³Eine Abbildung heißt genau dann konform (oder winkeltreu), wenn der Schnittwinkel zweier durch einen Punkt gehender stetig differenzierbarer Kurven bei der Abbildung erhalten bleibt (einschließlich des Richtungssinns).

zugehörige digitale impulsinvariante Filter die Übertragungsfunktion (4.12)

$$H(z) = \frac{1}{z - e^{-\frac{T}{RC}}} \quad (4.12)$$

mit dem Filterkoeffizienten (4.13) und δ -Impulsantwort (4.14).

$$\beta_1 = e^{-\frac{T}{RC}} \quad (4.13)$$

$$h(n) = \beta_1^n = e^{-\frac{T}{RC} \cdot n} \quad (4.14)$$

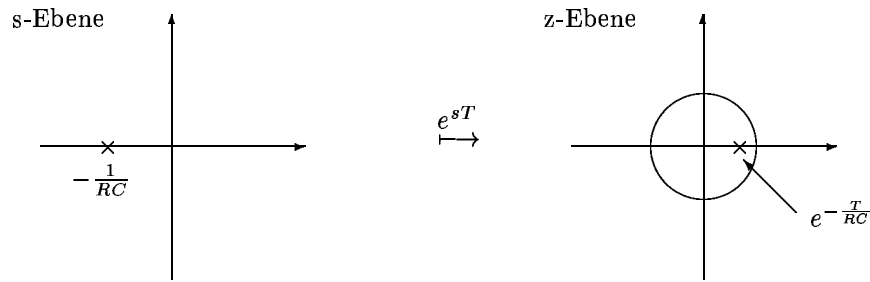


Abbildung 4.3: z-Transformation beim RC-Glied

In (Abb. 4.3) ist noch einmal die z-Transformation für das RC-Glied grafisch veranschaulicht. Das Filter hat nur eine Nullstelle auf der reellen Achse. Es treten bei der z-Transformation keine Aliasing Probleme auf, vorausgesetzt die Größen R und C sind so gewählt, daß $|H(s)|$ für $|\Im(s)| > \pi T$ weit genug abgefallen ist.

Erinnern wir uns noch einmal an die Impulsantwort des RC-Gliedes $h(t) = e^{-\frac{t}{RC}}$ so stellt man fest, daß Gleichung (4.14) die mit $f_s = \frac{1}{T}$ abgetastete Impulsantwort $h(t)$ des analogen Filters ist, daher der Name *impulsantwortinvariant*.

In (Abb. 4.4) sind noch einmal in übersichtlicher Form die Zusammenhänge beim *impulsantwortinvarianten* Filterdesign dargestellt.

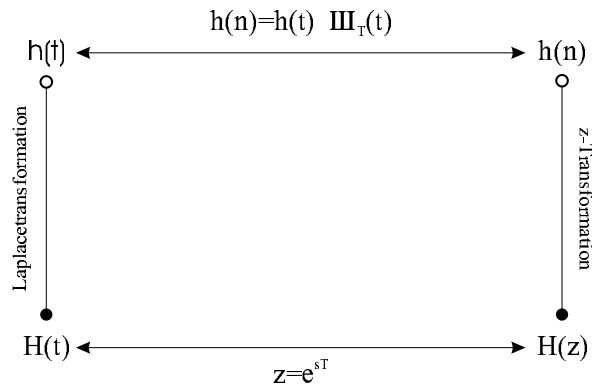


Abbildung 4.4: Impulsantwortinvariantes Filterdesign

4.2.4 Frequenzganginvariantes Design

Eine weitere Möglichkeit ein analoges Filter in ein digitales zu übertragen ist *frequenzganginvariantes* Filterdesign. Man sucht eine Abbildung von der s -Ebene in die z -Ebene, die die gesamte imaginäre Achse auf den Einheitskreis und die linke Halbebene in den Einheitskreis abbildet.

Diese Bedingungen erfüllt die *Bilineartransformation* (Gleichung (4.15)). Die Rücktransformation ist in Gleichung

(4.16) dargestellt.

$$z = \frac{1 + \frac{T}{2} \cdot s}{1 - \frac{T}{2} \cdot s} \quad (4.15)$$

$$\Leftrightarrow s = \frac{2}{T} \cdot \frac{z - 1}{z + 1} \quad (4.16)$$

Die entscheidende Frage ist jetzt, wie die unendlich lange imaginäre Achse der s-Ebene auf den endlichen Einheitskreis der z-Ebene abgebildet wird. Ein linearer Zusammenhang ist wegen “unendlich” von vornherein auszuschließen. Dieser Frage soll jetzt rechnerisch nachgegangen werden.

$$\begin{aligned} z &= e^{i2\pi f_d/f_s} =: e^{i\phi} \\ \Rightarrow \phi &= 2\pi f_d/f_s = 2\pi f_d T \\ s &= \frac{2}{T} \cdot \frac{z - 1}{z + 1} = \frac{2}{T} \cdot \frac{e^{i\phi} - 1}{e^{i\phi} + 1} \\ &= \frac{2}{T} \cdot \frac{e^{i\phi/2} - e^{-i\phi/2}}{e^{i\phi/2} + e^{-i\phi/2}} = \frac{2}{T} \cdot \frac{i \cdot \sin(\phi/2)}{\cos(\phi/2)} \\ &= \frac{2i}{T} \cdot \tan(\phi/2) \\ s &= \sigma + i\omega \Rightarrow \omega = 2\pi f_a = \frac{2}{T} \cdot \tan(\phi/2) \\ \Rightarrow f_a &= \frac{1}{\pi T} \cdot \arctan\left(\frac{\pi f_d}{f_s}\right) \end{aligned} \quad (4.17)$$

Gleichung (4.17) ist in (Abb. 4.5) für die Samplingfrequenz $f_s = 22050 \text{ Hz}$ aufgetragen. Gut zu erkennen ist ein annähernd linearer Verlauf bei den unteren Frequenzen bis zu $\frac{1}{4}f_s$. Oberhalb dieses Bereiches ist die Frequenzabbildung sehr stark gestaucht. Daran sollte man bei *frequenzganginvariantem* Filterdesign immer denken.

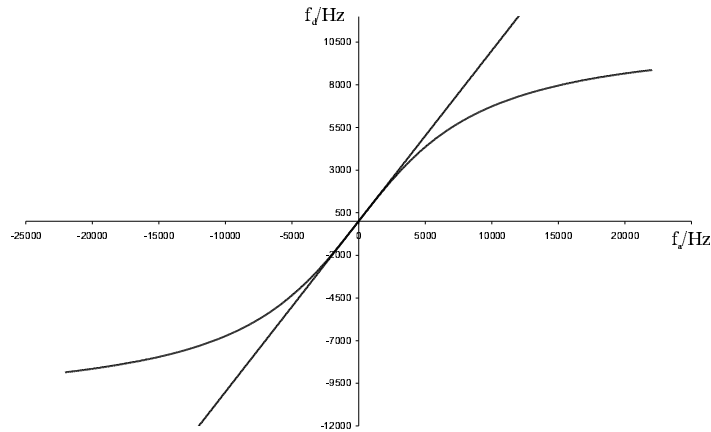


Abbildung 4.5: Abbildung Frequenzen: analoger Filter \mapsto digitaler Filter

Betrachten wir wieder das Beispiel des RC-Gliedes und wenden die Bilineartransformation (Gleichung (4.15)) auf die Übertragungsfunktion des analogen Filters (Gleichung (4.5)) an. Man erhält nach einigen Umformungen folgende digitale Übertragungsfunktion:

$$H(z) = \frac{z + 1}{1 - \frac{2}{T}RC + z \cdot \left(1 + \frac{2}{T}RC\right)} \quad (4.18)$$

4.2.5 Inverses Filter

Ein digitales Filter wird durch die Übertragungsfunktion $H(z)$ beschrieben, die ihren Definitionsbereich in der gesamten z -Ebene hat. Wir erinnern uns daran, daß die digitale Übertragungsfunktion eine rationale Funktion ist und deshalb wie in Gleichung (4.19) durch ihre Pole z_{pj} und Nullstellen z_{0k} dargestellt werden kann.

$$H(z) = \frac{\prod_{k=1}^M (z - z_{0k})}{\prod_{j=1}^N (z - z_{pj})} \quad (4.19)$$

Gesucht ist nun das *inverse* Filter H^{-1} . Die Hintereinanderausführung des Filters H und des inversen Filters H^{-1} soll das ursprüngliche Eingangssignal wiederherstellen $H^{-1} \cdot H \cdot X = X \Rightarrow H^{-1} \cdot H = 1$.

Das inverse Filter ist der Kehrwert von Gleichung (4.19).

$$H^{-1}(z) = \frac{\prod_{j=1}^N (z - z_{pj})}{\prod_{k=1}^M (z - z_{0k})} \quad (4.20)$$

Trotz dieser einfachen Darstellung existiert das inverse Filter i.A. nicht. Besitzt das Filter H Nullstellen außerhalb des Einheitskreises, werden diese bei H^{-1} zu Polen. Das Stabilitätskriterium besagt aber, daß das Filter keine Pole außerhalb des Einheitskreises besitzen darf.

4.3 Typen analoger Filter

Im Bereich der Analogtechnik existieren schon eine Reihe von optimierten Filtertypen, deren Eigenschaften man bei der Umwandlung in digitale Filter natürlich erhalten möchte. Sie werden hier der Vollständigkeit halber erwähnt.

4.3.1 Butterworth

Butterworth-Filter sind im Durchlaßbereich flach, der Übergang in den Sperrbereich verläuft steil und die Übertragungsfunktion knickt im Bereich der Grenzfrequenz scharf ab (Abb. 4.6a). Dafür ist der Phasenverlauf nicht linear, und damit die Gruppenlaufzeit vor allem im Bereich um die Grenzfrequenz nicht näherungsweise konstant (Abb. 4.6b). Ein großer Nachteil dieses Filtertyps besteht darin, daß er nicht sehr stabil ist, d.h. bei Stoßanregung kann sich schnell eine Schwingung mit wachsender Amplitude aufschaukeln. Butterworth-Filter besitzen nur komplex konjugierte Pole, die in der s -Ebene auf dem Einheitskreis in der reell-negativen Hälfte sowie im Ursprung liegen.

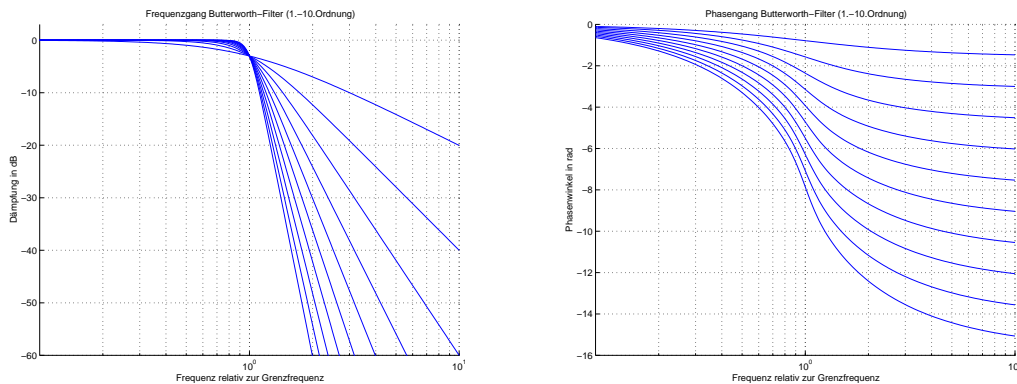


Abbildung 4.6: Übertragungsfunktionen analoger Butterworth-Filter

4.3.2 Chebyshev

Chebyshev-Filter (auch: Tschebyscheff) fallen noch steiler in den Sperrbereich ab als Butterworth-Filter, das erkauft man sich jedoch durch eine Welligkeit im Durchlaßbereich mit konstanter Amplitude, die um so größer wird, je steiler die Flanke sein soll (Abb. 4.7a). Auch der Phasenverlauf ist wellig und die Gruppenlaufzeit weicht noch stärker vom konstanten Verlauf ab als die eines Butterworth-Filters (Abb. 4.7b).

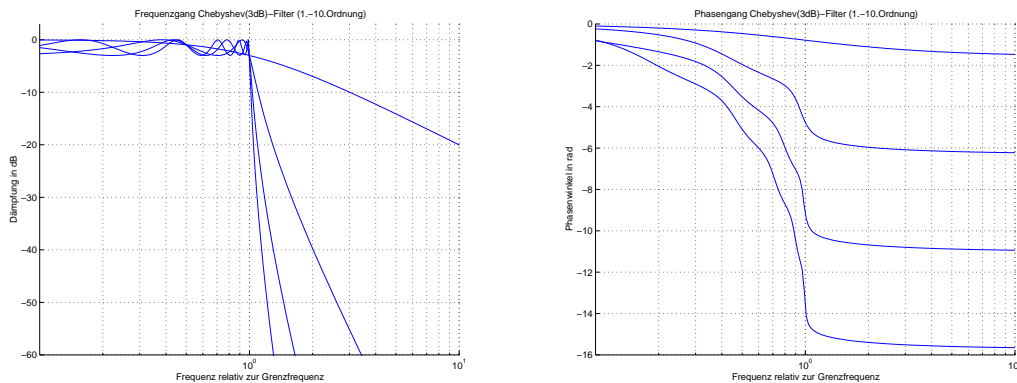


Abbildung 4.7: Übertragungsfunktion analoger Chebyshev-Filter mit 3dB Welligkeit im Durchlaßbereich

4.3.3 Bessel

Bessel-Filter sind für eine möglichst konstante Gruppenlaufzeit unterhalb der Grenzfrequenz optimiert, d.h. annähernd linearen Phasenverlauf (Abb. 4.8b). Damit sind sie vor allem für die unverzerrte Übertragung z.B. von Rechteckimpulsen geeignet. Im Gegensatz zu Butterworth- oder Chebyshev-Filtern verläuft der Übergang vom Durchlaß- in den Sperrbereich deutlich flacher (Abb. 4.8a).

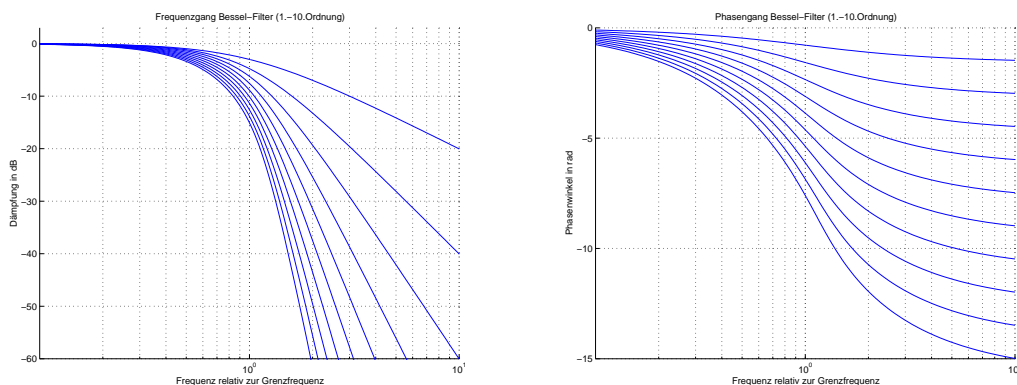


Abbildung 4.8: Übertragungsfunktion analoger Bessel-Filter

4.4 Design digitaler FIR-Filter

Wie schon anfangs erwähnt, besitzen FIR-Filter Vorteile gegenüber IIR-Filtern, nämlich daß sie numerisch immer stabil sind und man auf einfache Weise Phasenlinearität erreichen kann. Allerdings ist der Aufwand, sprich: die Anzahl der benötigten Filterkoeffizienten, bei vergleichbarer Filterwirkung (d.h. z.B. Flankensteilheit) erheblich höher.

Zwei der gebräuchlichen Methoden zur Optimierung von FIR-Filtern, das "Windowing-Design" und das Frequenzabtastfilter, werden im folgenden vorgestellt, das Gradientenverfahren wird hier nur als weitere Möglichkeit erwähnt, aber nicht detailliert behandelt. Davor zeigen wir noch, wie man die Phasenlinearität von FIR-Filtern erreicht.

4.4.1 Phasenlinearität von FIR-Filtern

Hinreichende Bedingung für die Phasenlinearität eines Filters ist eine symmetrische Impulsantwort. Dies impliziert ein FIR-Filter, denn eine symmetrische, aber unendliche Impulsantwort würde nicht der Kausalitätsforderung genügen.

$$\begin{aligned} h(n) &= h(N-n) & n = 0, \dots, \frac{N}{2} - 1 & \quad \text{für } N \text{ gerade} \\ & & n = 0, \dots, \frac{N-1}{2} & \quad \text{für } N \text{ ungerade} \end{aligned} \quad (4.21)$$

Wir nehmen an, N sei gerade, der Beweis für ungerades N verläuft analog. Aus $h(n)$ berechnen wir die z -Transformierte und zeigen mit ihr unter Ausnutzung der Symmetrie, daß der Phasenwinkel unabhängig von weiteren Bedingungen linear verläuft:

$$\begin{aligned} H(z) &= \sum_{n=0}^{N-1} h(n)z^{-n} = \sum_{n=0}^{\frac{N}{2}-1} \left(h(n)z^{-n} + h(N-n)z^{-(N-n)} \right) \\ &= z^{-\frac{N}{2}} \left[\sum_{n=0}^{\frac{N}{2}-1} h(n) \left(z^{\frac{N}{2}-n} + z^{-(\frac{N}{2}-n)} \right) \right] = z^{-\frac{N}{2}} \underbrace{\sum_{n=0}^{\frac{N}{2}-1} h(n) \cdot 2\operatorname{Re}(z^{\frac{N}{2}-n})}_{\text{reell}} \end{aligned} \quad (4.22)$$

Für die Berechnung der Phase relevant bleibt $z^{-\frac{N}{2}}$. Mit der Definition von z (s. 4.2.2) ergibt sich:

$$\begin{aligned} \arg \left(z^{-\frac{N}{2}} \right) &= \arg \left((e^{-2\pi i \frac{f}{f_s}})^{-\frac{N}{2}} \right) \\ &= \frac{\pi N}{f_s} \cdot f \sim f \end{aligned} \quad (4.23)$$

Der Phasenwinkel ist proportional zu f und damit die Gruppenlaufzeit, definiert als Ableitung des Phasenwinkels nach der Frequenz, konstant.

4.4.2 Windowing-Design

Grundidee dieses Verfahrens ist, daß man sich zunutze macht, ein Filter vollständig durch seine Impulsantwort beschreiben zu können. Dazu transformiert man die vorliegende Übertragungsfunktion in die zugehörige Impulsantwort. Diese ist aber im allgemeinen unendlich lang und kontinuierlich und kann deshalb nicht digital verarbeitet werden. Außerdem muß die Kausalität beachtet werden, d.h. die Impulsantwort muß für $t < 0$ verschwinden.

Man geht deshalb folgendermaßen vor:

Zuerst wird die Impulsantwort $h(t)$ mit einer geeigneten Funktion $w(t)$ gefenstert, wobei hier je nach gewünschter Übertragungsfunktion und Aufwand verschiedene Fensterfunktionen optimal sein können. Die Multiplikation mit der Fensterfunktion im Zeitbereich entspricht eine Faltung der jeweiligen Fouriertransformierten, d.h. eine "Versmierung" der ursprünglich gewünschten Übertragungsfunktion, der Übergang vom Durchlaß- in den Sperrbereich wird breiter (Abb. 4.10b). An dieser Stelle wird deutlich, daß man durch die Wahl eines breiteren Fensters den Verschmierungseffekt verringern kann (weil die Fouriertransformierte des Fensters sich damit der idealen Deltafunktion nähert), aber dafür mehr Filterkoeffizienten und damit mehr Rechenaufwand hat.

$$\begin{aligned} \tilde{h}(t) &= w(t) \cdot h(t) \\ \Downarrow \\ \tilde{H}(f) &= W(f) * H(f) \end{aligned} \quad (4.24)$$

Dann wird die gefensterte Impulsantwort diskretisiert, d.h. abgetastet. Die Abtastwerte ungleich null entsprechen genau den Filterkoeffizienten des FIR-Filters (Gleichung (4.26)), wobei sie wegen der Kausalität erst so verschoben werden müssen, daß der erste nichtverschwindende Wert bei $t = 0$ liegt. Diese Operation ändert das Amplitudenverhalten des Filters nicht, dafür jedoch den Phasenverlauf, da eine Zeitverschiebung $t' = t + \tau$ einer Multiplikation mit dem reinen Phasenfaktor $e^{2\pi i f \tau}$ im Frequenzbereich äquivalent ist (Abb. 4.9).

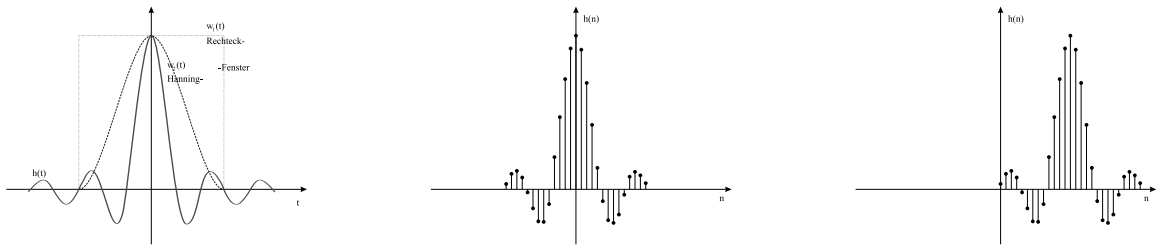


Abbildung 4.9: Fensterung, Abtastung und Verschiebung im Zeitbereich

Definition der diskreten Deltafunktion (Impuls):

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{sonst} \end{cases} \quad (4.25)$$

Einsetzen in die Differenzengleichung eines FIR-Filters (Koeffizienten $\alpha_k, k = 0, \dots, N$) liefert :

$$\sum_{k=0}^N \alpha_k \delta(n - k) = \begin{cases} 0 & n < 0 \\ \alpha_n & 0 \leq n \leq N \\ 0 & n > N \end{cases} \quad (4.26)$$

Die Abtastung, im Zeitbereich äquivalent der Multiplikation mit einem Deltakamm, setzt die "gefensterte" Übertragungsfunktion mit der Samplingfrequenz $f_s = \frac{1}{T_s}$ periodisch fort (Abb. 4.10c).

$$\begin{aligned} h(n) &= \sqcup_{T_s}(t) \cdot w(t) \cdot h(t) \\ \Updownarrow \\ H(z) &= \sqcup_{f_s}(f) * W(f) * H(f) \end{aligned} \quad (4.27)$$

Dies sieht schematisch ungefähr so aus:

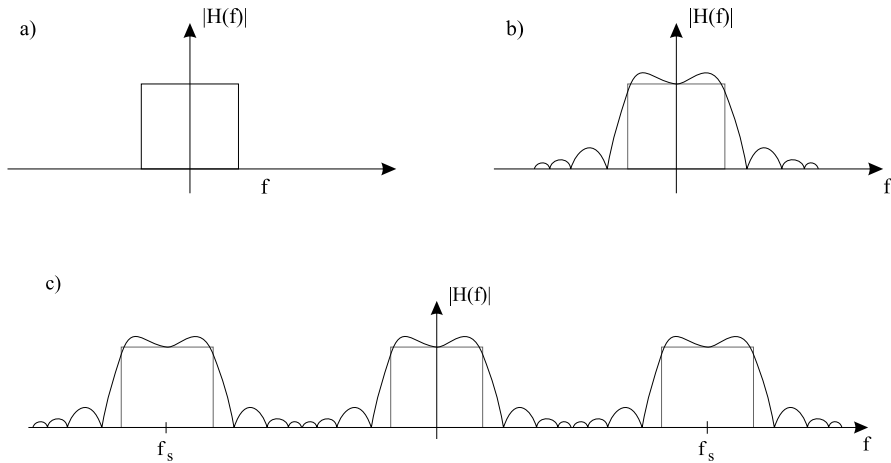


Abbildung 4.10: Übertragungsfunktion(a) nach Fensterung(b) und Abtastung(c)

4.4.3 Frequenzabtastfilter

Die Übertragung des "Windowing"-Designprinzips auf den Frequenzbereich liefert der Frequenzabtastfilter. Statt der gewünschten Impulsantwort arbeitet man mit der Übertragungsfunktion und tastet diese ab. Von der abgetasteten Übertragungsfunktion $\tilde{X}(k)$ bildet man die inverse DFT, um die Impulsantwort zu bekommen, allerdings ist diese mit N periodisch, also unter anderem unendlich lang, und nicht für ein FIR-Filter geeignet. Da aber im

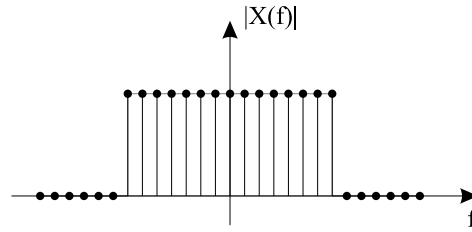


Abbildung 4.11: Abtastung der Übertragungsfunktion

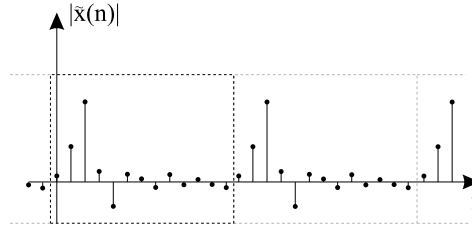


Abbildung 4.12: Periodische Impulsantwort

Prinzip fast alle Information in einer Periode der Impulsantwort steckt, nimmt man eine Periode und setzt alle anderen Werte auf Null. Die erhaltenen Werte einer Periode können nun direkt als Filterkoeffizienten eingesetzt werden.

$$\begin{aligned}\tilde{x}(n) &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{2\pi i \frac{kn}{N}} \\ x(n) &= \begin{cases} \tilde{x}(n) & 0 \leq n \leq N \\ 0 & \text{sonst} \end{cases}\end{aligned}\quad (4.28)$$

Der Haken an dieser Idee ist, daß man so nur das Verhalten des Filters an den Abtaststellen genau definiert. Wie die tatsächliche Übertragungsfunktion aussieht, zeigt die z-Transformation der Impulsantwort $x(n)$:

$$\begin{aligned}X(z) &= \sum_{n=0}^{N-1} x(n) z^{-n} = \sum_{n=0}^{N-1} \left[\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{2\pi i \frac{kn}{N}} \right] z^{-n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) \sum_{n=0}^{N-1} e^{2\pi i \frac{kn}{N}} z^{-n} = \dots \text{(Eigenschaften der geometrischen Reihe)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) \frac{1 - z^{-N}}{1 - e^{2\pi i \frac{k}{N}} z^{-1}} \\ &= \underbrace{\frac{1 - z^{-N}}{N}}_{X_0(z)} \sum_{k=0}^{N-1} \underbrace{\frac{\tilde{X}(k)}{1 - e^{2\pi i \frac{k}{N}} z^{-1}}}_{H_k(z)}\end{aligned}\quad (4.29)$$

In der letzten Form sieht man, daß $X_0(z)$ ein FIR-Filter (nur Nullstellen) mit den Koeffizienten $\alpha_0 = 1$, $\alpha_1 = \dots = \alpha_{N-1} = 0$, $\alpha_N = -1$ ist, die $H_k(z)$ dagegen IIR-Filter (All-Pol-Filter) sind, letztere mit den Koeffizienten $\alpha_0 = \tilde{X}(k)$ und $\beta_1 = e^{2\pi i \frac{k}{N}}$. Daraus folgt, daß man ein beliebiges FIR-Filter als Hintereinanderschaltung eines FIR-Filters $X_0(z)$ und der Summe (d.h. Parallelschaltung) mehrerer IIR-Filter darstellen kann.

Daß das Filter insgesamt wieder nur eine endliche Impulsantwort zeigt, ergibt sich aus der Tatsache, daß die Nullstellen von $X_0(z)$ und die einzelnen Polstellen von $H_k(z)$ jeweils an denselben Stellen auf dem Einheitskreis

liegen und sich gewichtet mit $\tilde{X}(k)$ gegenseitig aufheben:

$$\begin{aligned} X_0(z) &= \frac{1 - z^{-N}}{N} = \prod_{k=0}^{N-1} \left(z - e^{2\pi i \frac{k}{N}} \right) \quad \text{Nullstellen bei } e^{2\pi i \frac{k}{N}} \\ H_k(z) &= \frac{\tilde{X}(k)}{1 - e^{2\pi i \frac{k}{N}} z^{-1}} \quad \text{Polstellen bei } e^{2\pi i \frac{k}{N}} \end{aligned} \quad (4.30)$$

Die gewünschte Übertragungsfunktion wird praktisch interpoliert, zwischen den Stütz-(Abtast-)stellen kann die tatsächliche Übertragungsfunktion zum Teil erhebliche Abweichungen aufweisen.

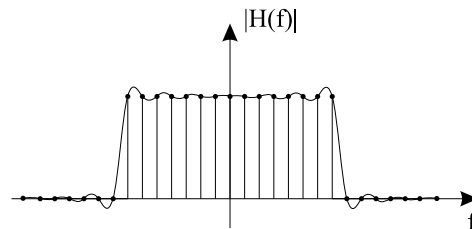


Abbildung 4.13: Interpolation der Übertragungsfunktion

4.4.4 Gradientenverfahren

Wie schon erwähnt, können wir hier keine detaillierte Information zu interaktiven Verfahren des Filterdesigns liefern, da für eine Behandlung dieses Themas während des Praktikums keine Zeit blieb. Wir verweisen hier als Beispiel auf das Projekt “Adaptive Filter“ in der zweiten Praktikumswoche, in dem unter anderem eine Methode zur digitalen Modellierung unbekannter Filter vorgestellt wurde.

Literatur

- [1] KOLLMEIER, B.: *Skript Signalverarbeitung Teil IV: Digitale Signalverarbeitung*, Oldenburg, September 1995
- [2] LÜKE, H.D.: *Signalübertragung*, Springer, 7. Auflage, 1998
- [3] REMMERT: *Funktionentheorie 1*, Springer, 4. Auflage, 1995
- [4] TIETZE, U. / SCHENK, CH.: *Halbleiter-Schaltungstechnik*, Springer, 1993
- [5] ZEIDLER, E.: *Teubner-Taschenbuch der Mathematik*, B.G. Teubner Stuttgart, Leipzig

5 Versuche zur AD/DA-Wandlung

von Roman Kehle und Diane Hemmen-Schüler

5.1 Einleitung

Die Digitalisierung analoger Signale bietet viele Vorteile hinsichtlich der Speicherung, der Übertragung und Bearbeitung von Musik oder Sprache. Das bei der AD-Wandlung auftretende Problem der Quantisierung des Signals tritt nicht bei der DA-Wandlung auf, d.h. es gehen keine Informationen durch die Konvertierung verloren und im Idealfall erhält man das ursprüngliche Signal zurück. Dieses ist natürlich für die Datenübertragung sehr wichtig. Ein zweiter Effekt der bei der AD-Wandlung auftreten kann, ist der des Aliasing. Dieses ist durch den Einsatz von Filtern in den Wandlern in den Griff zu bekommen. Digitale Signale lassen sich mit einem Rechner und der entsprechenden Software leicht bearbeiten, um diese Effekte zu simulieren. Ein dafür geeignetes Programmpaket stellt MATLAB dar.

5.2 Programmierung unter MATLAB

MATLAB ist ein Command-Line-Interpreter. Einzelne Programme werden als Skripten, sogenannte M.File mit der Extension .m, im MATLAB-Editor MEDTIT.EXE erzeugt. Die ganzen Funktionen MATLABs selbst sind in Skripten abgefaßt. Für Graphiken wird ein eigenes Ausgabefenster geöffnet.

Um den Umgang mit MATLAB zu erleichtern, sei auf folgendes hingewiesen:

- Variablen werden dynamisch als Matrix angelegt.
- Funktionen können als Gleichungen angegeben werden.
- Hilfe läßt sich mit help BEFEHL aufrufen und ist auch im HTML-Format vorhanden

Vorteile bei der Arbeit mit MATLAB sind:

- Typumwandlungen entfallen.
- Einfache Bearbeitung von Sounddateien.
- Visualisierung ist für den wissenschaftlichen Gebrauch zweckmäßig.
- Der mathematische Funktionsumfang ist erweiterbar.

Nachteile gibt's auch:

- Strukturiertes Programmieren ist nicht klar erkennbar. Die Befehle sind nicht transparent genug.
- Das Arbeitsverzeichnis des Interpreters und des Editors stimmen nicht überein.
- Die Hilfe könnte ausführlicher sein.
- LaTeX oder HTML wird nicht unterstützt.

Einige Beispiele, die als Aufgaben zu bearbeiten waren:

1. Datengenerierung und -manipulation

- Sinus (1 kHz, Abtastfrequenz 10 kHz, Länge 10 ms)

```
x=0:10; y=sin(2*pi*(1000/10000)*x); plot(x,y);
```

- Sinus (*Zeitbasis generieren und Sinus darüber plotten*)
gemeinsame Zeitbasis: (kgV) ergibt 100
Sinus : in 10 ms erhalten wir 10 Schwingungen
Abtastfrequenz : in 10 ms erhalten wir 100 Schwingungen

```
x=0:100;y=sin(2*pi*(1000/10000)*x); plot(x,y);
```

2. Umgang mit Matrizen

- Skalarprodukt oder Inneres Produkt

```
a=[1 2 3]; b=[4 5 6]; c = a * b'
```

Wir multiplizieren einen Zeilen- mit einem Spaltenvektor.

- Vektorprodukt oder Äußeres Produkt

```
a=[1 2 3]; b=[4 5 6]; c = cross(a,b)
```

- Komponentenweises Multiplizieren

```
a=[1 2 3]; b=[4 5 6]; c = a .* b
```

3. Fourieranalyse

- Plotte das Leistungsspektrum eines Rauschsignals (*1000 Samples, Varianz 1*)

```
x=rand(1,1000); y=abs(fft(x-0.5)); plot(x,y);
```

Durch die kleine Verschiebung, die bei Gleichverteilung nicht nötig wäre, erhalten wir einen anderen Wertebereich. Das Spektrum ist periodisch und um die halbe Abtastfrequenz symmetrisch.

- Plotte logarithmisches Leistungsspektrum (doppellogarithmisch) (*in dB : $20 * \log_{10}(\text{abs}(fft(x)))$*)

```
x=20-log10(abs(fft(rand1,1000))); y=1:1000; semilogy(x,y);
```

5.3 Methode

Im folgenden wird die Meßapparatur beschrieben und gezeigt, wie die Software MATLAB zur digitalen Signalverarbeitung eingesetzt werden kann. Mit digitalen Signalen wird so verfahren, daß die Wortbreite von anfänglichen 16-Bit verringert wird, um das Quantisierungsrauschen hörbar zu machen. Ähnlich wird verfahren, um die etwas höher gewählte Abtastrate des abgetasteten Signals zu senken, damit ein Aliasingfehler simuliert werden kann. Da die AD-Wandlung analoger Signale bereits durch die Eigenschaft der verwendeten Wandler eine optimale Wiedergabe zuläßt (d.h. die Wandler haben auf die jeweilige Abtastfrequenz keine optimierten Anti-Aliasing-Filter), haben wir uns vergegenwärtigt, welche Auswirkungen diese Faktoren haben könnten.

5.3.1 Meßapparatur

Ein Mikrophon wurde mit der Soundkarte eines Computers verbunden. Die verwendete Software des Digitalen Signalverarbeitungs-Praktikums DSP99 besteht aus drei Programmen:

- **Cool Edit 96** [Syntrillium]
- **Mixer** [Microsoft]
- **MatLab** [Mathworks]

Cool Edit 96

Wir nutzen Cool Edit 96 zur Aufnahme von Sounddateien im Wave-Format. Im Programm wird dieses Format als „Windows-PCM“ mit der Extension .wav bezeichnet. Es bietet die Funktionalität eines Aufnahmestudios für zuhause. So ist z.B. die Nachbearbeitung einer Aufnahme zum Entfernen von Kratzern, Brummen und Rauschen möglich.

Mixer

Hinter dem Lautsprecher-Symbol in der Taskleiste versteckt sich die Lautstärkeregelung. Diese wird so angepaßt, bis das Eingangssignal voll ausgesteuert wird.

MATLAB

Diese Bezeichnung steht für MATrix LABoratory. Ein interaktives das System, das numerische Berechnungen und wissenschaftliche Visualisierung in sich vereinigt. MATLAB ist programmierbar und es können auch C-Programme eingebunden werden. Darüberhinaus verfügt MATLAB über eine Reihe von anwenderspezifischen Lösungen, die sogenannten Toolboxes, wie z.B. der Signalverarbeitung.

Wir benutzen MATLAB um unsere aufgenommene Sounddatei zu manipulieren und sehen uns verschiedene Signalverläufe im Zeit- und Frequenzspektrum an.

5.3.2 Meßsignal

Die Aussteuerung

Wichtig ist das Eingangssignal gut ausgesteuert wird. Dies wird in Cool Edit 96 im unteren Skalenbereich graphisch dargestellt. Dabei muß für die oben genannten Werte bei 90dB Maximalausschlag gewählt worden sein. Wenn die Versuchsperson bei jedem Test ungefähr gleich laut spricht, kann man die Amplitude mit dem Mixer modifizieren.

Die Aufnahme

Ein gesprochener Satz wie z.B. „Das ist ein Testsatz.“ wird in Cool Edit 96 aufgenommen und abgespeichert. Dabei werden einige Angaben erwartet. Zum Beispiel nehmen wir den Satz mono auf. Im Gegensatz dazu kann in stereo nacheinander auf dem jeweils selektierten Kanal etwas aufgenommen und gleichzeitig wiedergegeben werden. Die DA-Wandlung des Eingangssignals soll mit der Abtastfrequenz 22050 Hz bei 16-Bit Wortbreite erfolgen. Diese Werte sollten für die nachfolgende Bearbeitung gewählt werden, dabei hört sich das Aufgenommene dem Orginal ziemlich ähnlich an und die Dateien sind nicht unnötig zu groß geraten.

In Abbildung 5.1 ist zu sehen, was Cool Edit 96 anzeigt.



Abbildung 5.1: Cool Edit 96

5.3.3 Meßverfahren

Die Sounddatei soll mit Hilfe von MATLAB bearbeitet werden. Das für uns vorbereitete MATLAB-Skript FTOF.M beinhaltet die Funktion WAVREAD, die diese Datei als Matrix einliest und die Anzahl der Samples übergibt. Wir fügen einen eigenen Programmteil in FTOF.M ein um die Matrix zu bearbeiten. Danach wird mittels der Funktion WAVWRITE mit der bearbeiteten Matrix eine neue Sounddatei generiert. Im Aufruf des Skripts muss der Name der alten und der neuen Sounddatei übergeben werden.

```
FTOF('Dateiname1','Dateiname2');    % die Angabe des Pfades
                                     % ist moeglich}
```

Anschließend hören wir uns die neue Sounddatei mit Cool Edit 96 an.

Quantisierungsrauschen bei verschiedenen Wortbreiten

Die mit einer Wortbreite von 16-Bit aufgenommenen Samples besitzen bei voller Aussteuerung die höchst mögliche Information von $2^{16} = 65536$. Das heißt wir erhalten Werte im Bereich $I = [-32768, +32768]$. Die Werte in unserem Eingangs-Array IN mit den Samples liegen im Bereich $I = [-1, 1]$, sind Dezimalzahlen und besitzen praktisch die gleiche Information. Die Wortbreite zu verringern bedeutet, daß die Werte ungenauer gemacht werden. Dies könnte durch Runden zum nächsten ganzzahligen Wert oder durch Runden zum nächsten, kleinsten ganzzahligen Wert erfolgen.

Der von uns eingefügte Programmteil in FTOF.M:

```
OUT = IN * 2 ^ 15;           % Verschieben um 15 Stellen nach links
BIT = 2;                     % beispielsweise um 2-BIT Wortbreite
OUT = FLOOR(OUT*2^(BIT-16)); % zurueckschieben, Runden
OUT = OUT*2^(16-BIT);        % und wieder nach links schieben
OUT = OUT*2^(-15);           % endgueltig nach rechts verschieben
```

mit

BIT : neue Wortbreite

IN : Eingangs-Array (Matrix) mit Samples

OUT: Ausgangs-Array (Matrix) mit Samples

Es ist zu bemerken, falls mit der Student Edition von MATLAB gearbeitet wird, daß die Matrix nur 16384 Elemente aufweisen darf, was der gleichen Anzahl Samples und ungefähr einer Aufnahmedauer von einer halben Sekunde entspricht.

Die Funktion $\text{floor}(x + 0.5) = \text{round}(x)$ wie Abbildung 5.2 zeigt.

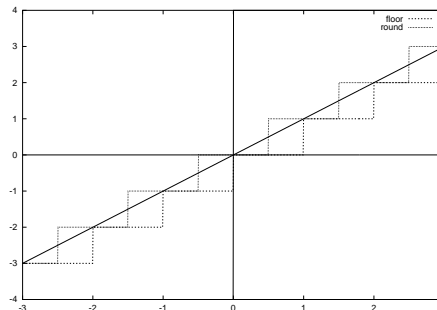


Abbildung 5.2: Floor und Round

Abtasttheorem/Aliasing

Bei der Änderung der Abtastrate ist zu beachten, daß sowohl die Anzahl der Samples, als auch die Abtastrate selbst entsprechend zu modifizieren ist. Beispielsweise erhalten wir für unseren Satz „Das ist ein Testsatz.“ entsprechend der Abtastrate von 22050 kHz bei seiner Länge von ungefähr 1863 ms genau 41091 Samples. Diese Werte stammen von Cool Edit 96. Zu fast dem selben Ergebnis kommen wir, indem wir die Abtastrate berechnen: $\text{Samples} = \text{Abtastrate} \cdot \text{Dauer} = 22050 \text{ 1/ms} / 1000 \cdot 1863 \text{ ms} = 41079$. Die Dauer ist wohl eher 1863.5 ms, da die Anzeige bzw. die Zeitmessung von Cool Edit 96 nicht genauer ist. Es ergibt sich ein Fehler von $\pm 1 \text{ ms}$.

Wir benutzen wieder FTOF.M, jedoch entfernen wir einfachheitshalber $\text{OUT} = \text{IN}$, da weniger Elemente in der neuen Matrix sein sollen. Will man die Anzahl der Samples ermitteln auch noch $\text{out_laenge} = \text{in_laenge}$ weglassen.

Der von uns eingefügte Programmteil in FTOF.M:

```
ZAEHLER = 1;
SCHRITT = 2;                                % Nur jedes 2.Sample wird
FOR I = 1:SCHRITT:IN_LAENGE,
    OUT(ZAEHLER) = IN(I);                    % in die Ausgangs-Matrix geschrieben
    ZAEHLER = ZAEHLER + 1;
END
SFREQ = SFREQ / SCHRITT;                     % und entsprechend die Abtastrate halbiert.
OUT_LAENGE = ZAEHLER - 1;
```

mit

IN : Eingangs-Array (Matrix) mit Samples
OUT : Ausgangs-Array (Matrix) mit Samples
OUT LAENGE: Anzahl der Samples des Ausgangs-Arrays
SCHRITT : Ordnungszahl Element N
SFREQ : Abtastrate
ZAEHLER : Zählt die Samples

Selbstverständlich ist dieses Vorgehen auch in Cool Edit 96 möglich. Einfach das Aufgenommene kopieren und als Neu mit anderer Abtastrate einfügen.

5.3.4 Meßergebnisse

An den Messungen nahmen eine männliche und eine weibliche Versuchsperson teil. Die übereinstimmend zu folgendem Ergebnis kamen:

- **Quantisierungsrauschen bei verschiedenen Wortbreiten**

Der Testsatz ist bis 6 Bit noch ziemlich ohne Rauschen zu hören. Wird die Wortbreite weiter verringert nimmt das Rauschen immermehr zu. Das Gesprochene ist trotzdem auch bei einem Bit noch verständlich. Die Lautstärke ausgenommen des Rauschens ist konstant. **In der gemeinsamen Besprechung wird festgestellt:**

- Bei 5-6 Bit ist das Rauschen nicht mehr als störend zu empfinden.
- Die Anwendung der Funktionen Round oder Floor ist nicht zu unterscheiden.
- In höheren Tönen ist mit abnehmender Bitzahl mehr Rauschen zu hören als tieferen Tönen, da Sprache ein 1/f-Spektrum aufweist, das Quantisierungsrauschen aber flach ist.
- Je höher die Bitzahl, desto besser ist das Signal durchmoduliert.
- Das Rauschen setzt sich aus relativ kleinen unregelmäßigen Signalen zusammen.
- **Abtasttheorem/Aliasing**

Die Abtastrate des Testsatzes ist sowohl per MATLAB-Prozedur als auch mit Hilfe von Cool Edit 96 zu ändern. Der Testsatz hörte sich erst bei 1000 Hz deutlich verfremdet an. Das Gesprochene war kaum noch zu verstehen. Die Stimme war leiser.

5.4 Diskussion

- **Quantisierungsrauschen bei verschiedenen Wortbreiten**

Das Verringern der Wortbreite bedeutet, daß Samples, die zwischen zwei zulässigen diskreten Werten liegen, auf- oder abgerundet werden. Die dabei entstehenden Fehler sind zufällig. Das bei einer bestimmten Wortbreite entstehende Rauschen hat eine uniforme Wahrscheinlichkeitsdichtefunktion. Das Quantisierungsrauschen ist proportional zur Bitzahl. Anders betrachtet, steht bei 16 Bit gerade $6\text{dB} \cdot 16 = 96\text{dB}$ rauschfreier Bereich zur Verfügung. Im Prinzip sind es noch weniger da durch Übersteuerung 10dB, einen höheren Störpegel 14dB und ein Sicherheitsabstand von 20dB reserviert sind, damit Verzerrungen vermieden werden wie in dem Themenbeitrag „Digital Audio“ [1] bemerkt wird. Bei 6 Bit sind schon ungefähr 54dB Rauschen vorhanden. Somit rückt das Rauschen immer mehr störend in den Vordergrund. Es verdeckt jedoch das Signal nicht, weil dafür der spektrale Unterschied zu groß ist.

- **Abtasttheorem/Aliasing**

Das Signal wird durch das Auftreten des Aliasing verfälscht. Das an der halben Abtastfrequenz gespiegelte Frequenzband überlappt sich mit dem ursprünglichem Signalband. Dieses wiederholt sich bei allen ganzzahligen Vielfachen der Abtastfrequenz. Die Amplitude hat sich in dem Bereich, in dem sich die Bänder überlappen, erhöht. Verhindert wird Aliasing, wenn die höchste im Signal vorkommende Frequenz nicht größer als die halbe Abtastfrequenz wird, wie das Abtasttheorem von Shannon besagt. —[2]— Bei einer Frequenzanalyse des Testsatzes mit

Cool Edit 96 ist zu sehen, daß nur Frequenzen bis zur halben Abtastfrequenz vorkommen, da ein Antialiasingfilter (Tiefpaßfilter) im AD-Wandler eingebaut ist. Bei der DA-Wandlung braucht man ein Rekonstruktionsfilter um die Frequenzen zu entfernen, die durch die Spiegelung des Signals an der Abtastfrequenz dazu gekommen sind. Diese waren ja im ursprünglichem Signal nicht vorhanden. Sie liegen oberhalb der Nyquist-Frequenz, die die Obergrenze der durch die Fourier-Transformation —[3]— noch erfaßbaren Frequenzen darstellt. Dadurch war diese metallisch klingende Verzerrung in den höheren Frequenzen bemerkbar.

Literaturverzeichnis

Literatur

- [1] STUCKERT, R.: *Digital Audio*. <http://www.darmstadt.gmd.de/mobile/courses/digitalvideo/ss97/seminar/reports/ralf.stuckert/index.html>, 1997
- [2] BÜLL, I.: *Physik mit dem Computer*. S. 129 - 134, 1998
- [3] WORTMANN, U.: *Fourier-Transformation*. <http://www.spoc.ethz.ch/uli/papers/diss/main-node23.htm>, 1999

6 Experimente zur Spektralanalyse und DFT

von Johannes Wisch und Daniel Reddig

6.1 Zusammenfassung und Einleitung

Am zweiten Praktikumstag bestand unsere Aufgabe darin, die diskrete Fouriertransformation genauer kennenzulernen. Speziell sollten die Eigenschaften der verschiedenen Fenstertechniken untersucht werden. Hierfür wurde uns ein spezielles Matlab-Skript zur Verfügung gestellt, welches insgesamt 36 Demoplots zu verschiedenen Zeitfunktionen und deren Spektren darstellte. Als ein Anwendungsbereich der Fourieranalyse sollten auch die AD-gewandelten Sprachsignale vom ersten Praktikumstag analysiert werden. Dabei ging es darum, eine möglichst optimale Fensterung sowie optimale Anpassung der Samplingfrequenz zu wählen, um im Spektrum die Formanten der verschiedenen Vokale deutlich zu erkennen.

Im folgenden werden die Demoplots einzeln vorgestellt und erläutert warum die DFT zu einer Zeitfunktion genau das Spektrum erzeugt welches man im Demoprogramm sieht.

6.2 Versuchsaufbau

Als Hilfsmittel standen uns die Computer mit Soundkarte zur AD-DA Wandlung des CIP-Raumes zur Verfügung. An jeden Computer waren außerdem ein Mikrophon und ein Kopfhörer angeschlossen. Weiterhin verwendeten wir folgende Programme :

- TAG2: Demoprogramm mit 36 Demoplots zur DFT
- MIXTER: Einstellung der Wiedergabe-Lautstärke.
- MATLAB: Softwarepaket zur Signalverarbeitung

6.3 Versuchsdurchführung

Die erste zu untersuchende DFT hatte als Zeitsignal einen Sinus mit 341Hz, und wurde mit einer Frequenz von 10kHz abgetastet (Abb. 1.1). Dieses Zeitsignal ist genau deshalb ein Sinus, weil der Signalwert zum Start und Ende

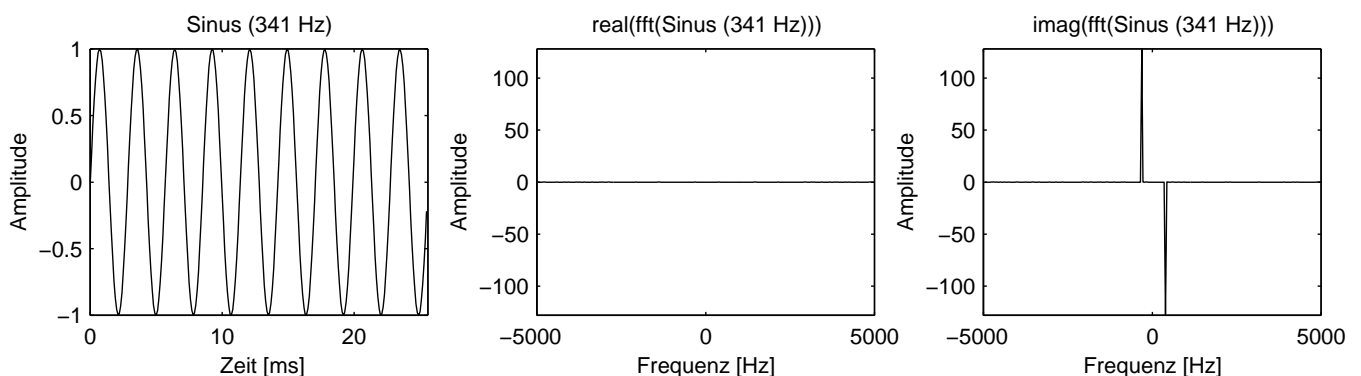


Abbildung 6.1: Ein Sinus mit 341Hz

des DFT-Fensters Null ist und außerdem ein ganzzahliges Vielfaches einer Sinusschwing enthält. Wie in der Theorie am Vormittag besprochen, hat die DFT eines Sinus keinen Real-Teil, sondern nur zwei Peaks punktsymmetrisch um den Nullpunkt im Imaginärteil des Frequenzspektrums. Die zweite zu untersuchende DFT hatte als Zeitsignal einen Kosinus mit 341Hz (Abb. 1.2) Man kann die Argumentation für das Sinussignal analog auf den Kosinus

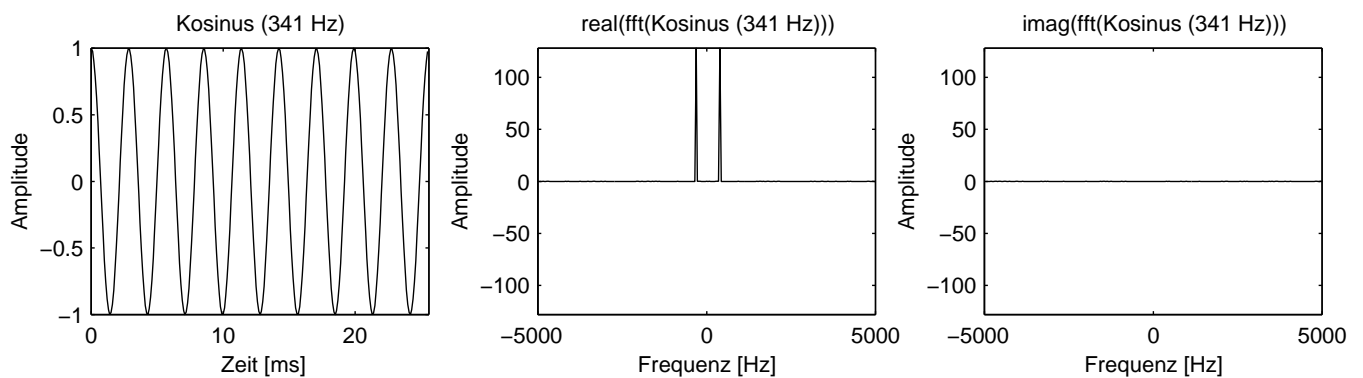


Abbildung 6.2: Ein Kosinus mit 341Hz

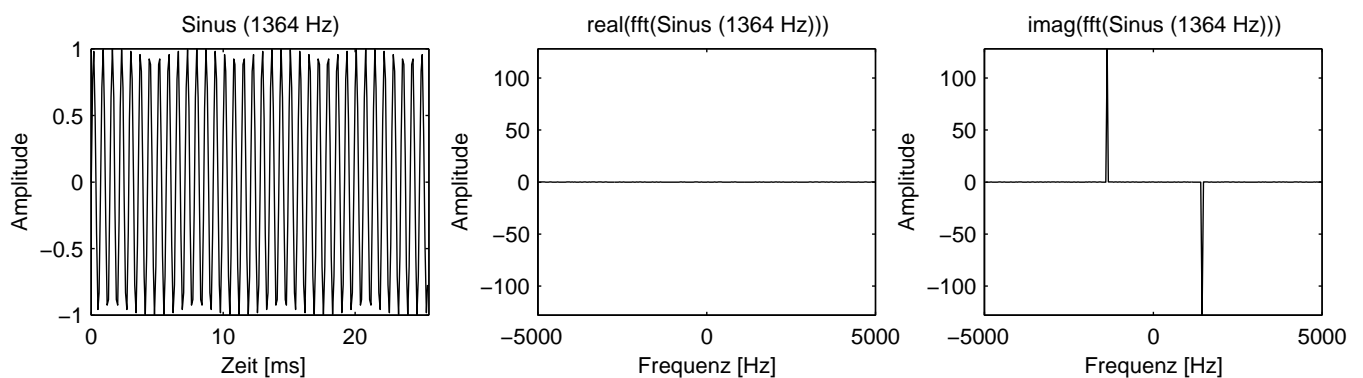


Abbildung 6.3: Ein Sinus mit 1364Hz

übertragen. Wie erwartet, haben wir nun zwei Peaks achsensymmetrisch um die Amplitudenachse im Realteil des Frequenzspektrums. Dafür haben wir keine Werte im Imaginärteil. Die dritte zu untersuchende DFT hatte als Zeitsignal einen Sinus mit 1364Hz (Abb. 1.3) Es trifft dieselbe Argumentation zu wie für den ersten Sinus. Man rechnet jedoch leicht nach, daß die Länge der Periode nur ein Viertel des ersten Sinus mit 341Hz hat. Die beiden Peaks liegen folglich viermal soweit auseinander (vergl. Theorie). Im vierten und fünften Bild sehen wir die DFT zweier Rechteckfunktionen die anscheinend spiegelsymmetrisch um die Amplitudenachse sind. Wie zu erwarten

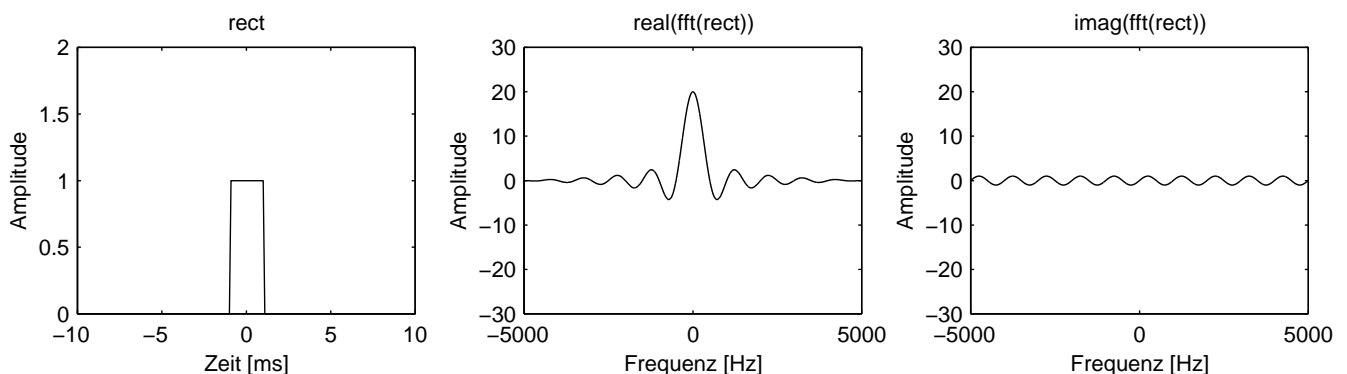


Abbildung 6.4: Rechteckfunktion 1

war, sehen wir im Realteil des Frequenzspektrums für beide Rechteckfunktionen eine Sincfunktion. Wir hätten im imaginären jedoch nichts erwartet, sehen nun aber einen Sinus. Wie ist das zu erklären? Offensichtlich sind die Rechteckfunktionen doch nicht so genau um den Nullpunkt zentriert wie wir dachten. Das hängt damit zusammen, daß die Anzahl der DFT Schritte gerade ist, d.h. es gibt keine exakte Mitte, sondern nur rechts oder links von der Mitte. Für unsere Rechteckfunktion bedeutet das, daß sie um einen halben DFT-Schritt aus der Mitte verschoben

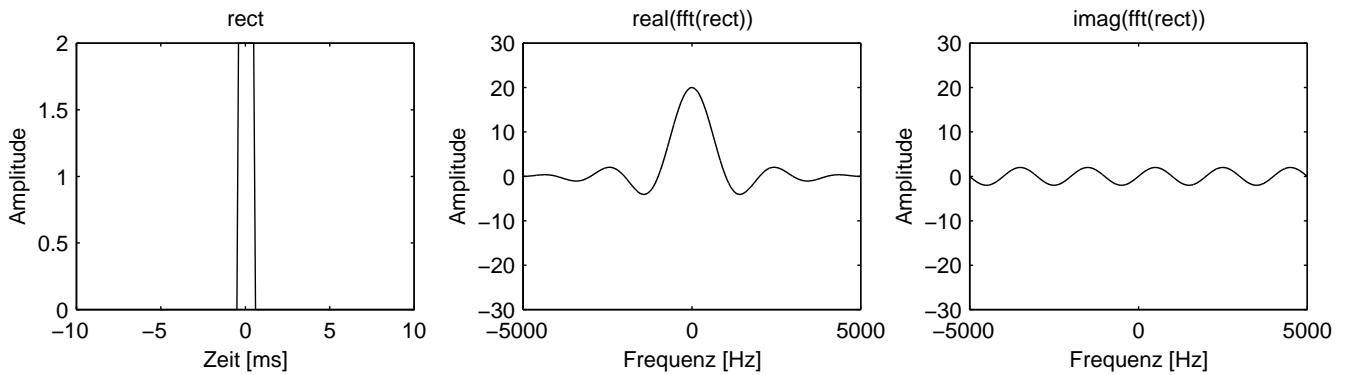


Abbildung 6.5: Rechteckfunktion 2

ist. Daher haben wir eine sehr flach abfallende Sincfunktion im imaginärteil des Frequenzspektrums. Je mehr eine Rechteckfunktion aus der Mitte verschoben ist, um so größer wird dieser Anteil. Ein Beispiel für zwei gleich große Real- und Imaginärteile sehen wir im nächsten Bild. In den folgenden drei Bildern sehen wir einen Deltapuls und

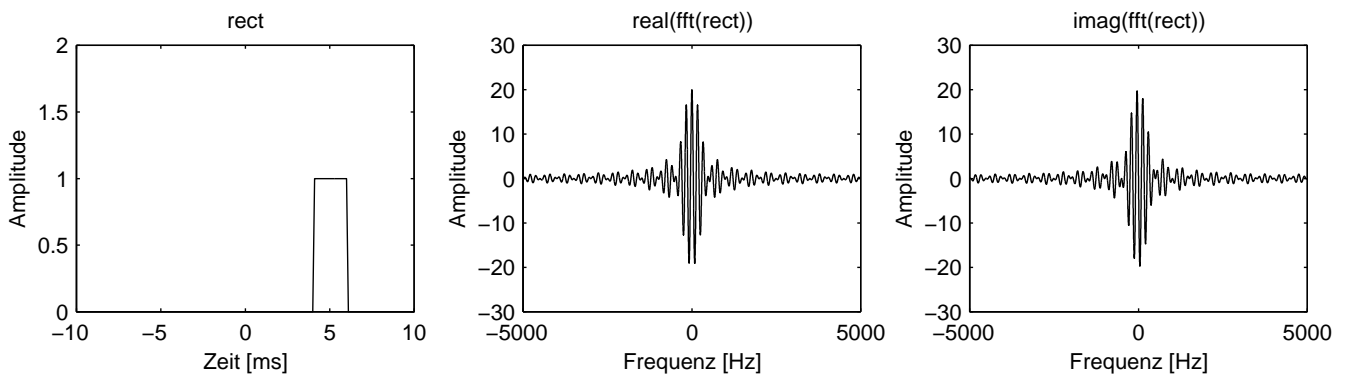


Abbildung 6.6: Rechteckfunktion 3

zwei Pulsfolgen mit unterschiedlicher Pulsweite. Die DFT der beiden Pulsfolgen ist klar und entspricht genau dem was wir erwartet hätten. Warum sehen wir für den Deltapuls aber eine konstante Funktion im Frequenzbereich? Da wir eine DFT mit endlicher Anzahl von Schritten haben, ist erst einmal eine Pulsfolge im Frequenzbereich mit der inversen Fensterlänge des Zeitbereichs als Periode zu erwarten. Das ist genau die Schrittlänge der DFT. Folglich bekommen wir für jeden DFT Schritt einen Deltapuls. Bei der graphischen Darstellung dieser Pulse werden die Maxima durch eine Linie verbunden.

6.4 Auswirkungen der Fenstertechnik

Oft interessiert uns bei einem längeren Zeitsignal die Frequenz des Signals bzw. das Amplitudenspektrum zu einem bestimmten Zeitpunkt. Da die Bestimmung der Frequenz zu einem genauen Zeitpunkt nicht möglich ist, müssen wir einen Ausschnitt geeigneter Länge herausgreifen, und auf diesen eine diskrete Fouriertransformation anwenden. (Wir gehen hier davon aus, dass das Signal digitalisiert mit einer bestimmten Abtastrate f_s und kontinuierlichen Abtastwerten vorliegt. Im Rechner sind die Abtastwerte natürlich auch quantisiert, z.B. als floating point Zahlen.) Bei der diskreten Fouriertransformation eines endlichen Signalstücks also eines nichtperiodischen Signals, erhalten wir als Ergebnis immer die Fouriertransformation einer Periode des Signals, das periodisch zu einem unendlichen Signal zusammengesetzt ist. Beim Hintereinanderhängen stellen wir im allgemeinen fest, dass das Ende und der Anfang des Signalstücks nicht zueinander passen. Es entsteht also eine Sprungstelle. Diese führt im Frequenzbereich zu einer Verbreiterung des Spektrums. Wir erhalten also Spektralanteile, die in dem betrachteten Ausschnitt des längeren Signals in Wirklichkeit nicht vorhanden sind. Um dies zu vermeiden, müssen wir dafür sorgen, dass die Enden der Signalstücke sanft auf Null gehen, so dass keine Sprungstelle mehr im periodischen Signal vorhanden ist. Dies erreichen wir durch die Multiplikation einer Wichtungsfunktion – auch Fensterfunktion

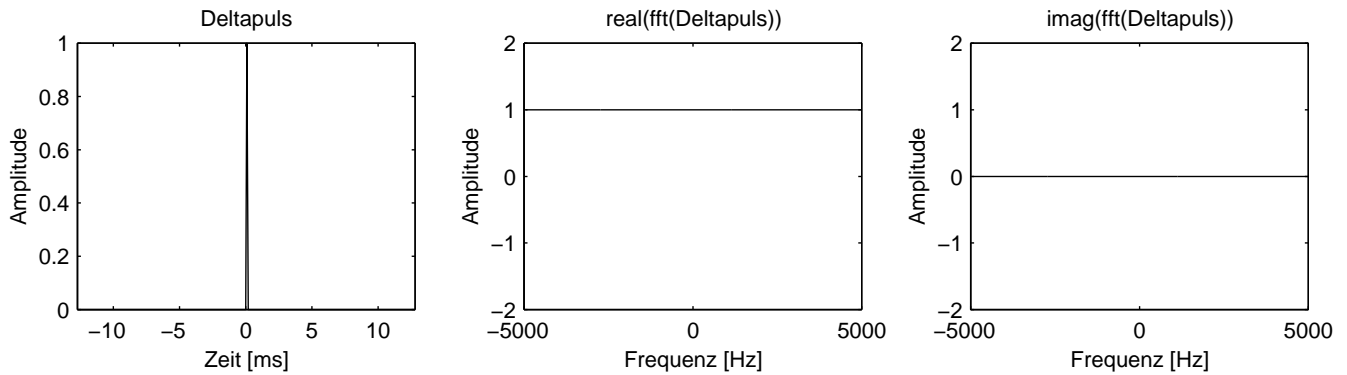


Abbildung 6.7: Deltapuls

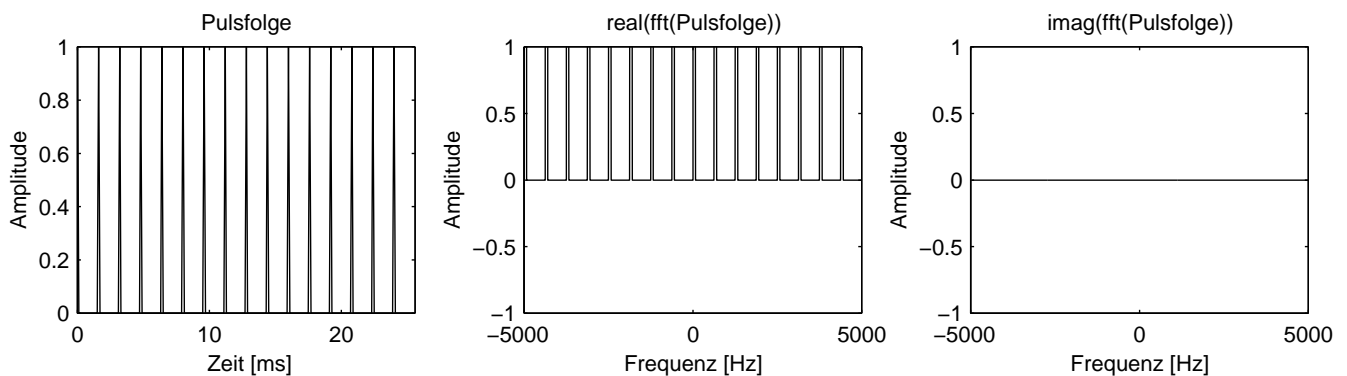


Abbildung 6.8: Pulsfolge mit 16 Pulsen im Fenster

oder kurz Fenster genannt – mit dem Signalausschnitt vor Ausführung der Fouriertransformation. Das einfachste Fenster stellt dabei das Rechteckfenster dar, das heißt, es wird einfach überhaupt kein Fenster benutzt und die Fouriertransformation wird direkt auf den Signalausschnitt angewandt, was zu den erwähnten Nachteil führt. Andererseits bedeutet die Anwendung von Wichtungsfunktionen, wie das Hammingfenster (Gleichung 6.1) oder das Hanningfenster (Gleichung 6.2) aber auch, dass das Signal selbst verändert wird, was wir insbesondere dann beachten müssen, wenn wir das Signal aus den Spektren der Signalausschnitte wieder zurücktransformieren und zusammensetzen (resynthetisieren) wollen. Abbildung 6.11 zeigt die Fensterfunktionen und deren Amplitudenspektren. Die Signaldauer beträgt 5 ms und wir haben 100000 Hz als Abtastfrequenz verwendet. Man nennt die Fouriertransformierte der Fensterfunktion auch den „Kernel“ der Fensterfunktion. Da sich das Spektrum des gefensterten Signalausschnitts durch Faltung der Fouriertransformierten des ungensterten Signals mit dem Kernel ergibt, können wir ihn uns auch als Impulsantwort im Frequenzbereich vorstellen.

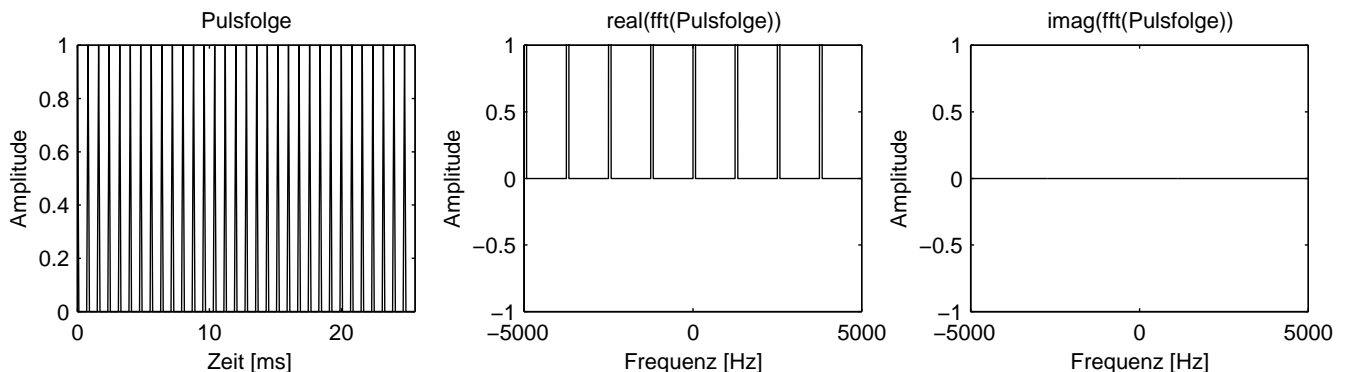


Abbildung 6.9: Pulsfolge mit 32 Pulsen im Fenster

Um die Wichtungsfunktionen miteinander vergleichen zu können betrachten wir folgende Eigenschaften, Sie bestimmen hauptsächlich die Qualität bei der Auflösung nahe beieinanderliegender Frequenzanteile.

- 6dB-Bandbreite (6 dB bandwidth)
- der Absolutbetrag des ersten Nebenmaximums (sidelobe level)
- die Abfallrate der Nebenmaxima (fall off)

Für die Darstellung eines Spektrums müssen wir die erreichbare Frequenzauflösung beachten. Bei der diskreten Fouriertransformation erhalten wir für einen Signalausschnitt mit N Abtastwerten N Werte im Frequenzbereich. Der maximal darstellbare Frequenzbereich zwischen $-\frac{f_s}{2}$ und $+\frac{f_s}{2}$ ist also in N sogenannte Frequenzbins aufgeteilt. Das bedeutet daß wir bei kurzen Signalausschnitten, also bei hoher Auflösung im Zeitbereich, eine geringe Auflösung im Frequenzbereich erhalten. Eine höhere Frequenzauflösung, ohne die spektrale Charakteristik des Signalausschnitts zu verändern, erhalten wir, indem wir an den gefensterten Signalausschnitt Nullen anfügen. Somit haben wir mehr Abtastwerte im Zeitbereich und damit auch mehr Frequenzbins im Frequenzbereich. Dieses Verfahren nennt man Zeropadding. Um es zu demonstrieren, haben wir an einem Rechteckfenster mit einer Breite von 5 ms, einmal 50 Nullen auf jeder Seite (Abbildung 6.10 a)) und einmal 1000 Nullen auf jeder Seite (Abbildung 6.10 c)) hinzu gefügt. Daneben sind die jeweiligen Amplitudenspektren zu sehen. In Bild 6.10 b) ist die Frequenzauflösung schlechter als in Bild 6.10 d), da viel weniger Frequenzbins (durch Kreuze dargestellt) zur Verfügung stehen. Zum Beispiel ist die erste Nullstelle die theoretisch bei 200 Hz zu erwarten ist, in Bild 6.10 b) gar nicht vorhanden.

Den sidelobe level haben wir grafisch mit Hilfe eines Ausdrucks wie in Bild 6.11 bestimmt. Die Signallänge beträgt 5 ms, die Abtastfrequenz ist 100000 Hz. In der Grafik entspricht 1 mm etwa 0,9 dB. Der Ablesefehler ist kleiner als 1 dB. Den fall off haben wir ebenfalls grafisch bestimmt. Dazu haben wir das Amplitudenspektrum nochmals bis 8000 Hz geplottet, so dass wir den Abfall über 3 Oktaven (1000 bis 8000 Hz) ablesen können (Abbildung 6.12).

Tabelle 6.1 zeigt die ermittelten Werte im Vergleich mit den Literaturwerten [2].

Fensterfunktion	Rechteck		Hanning		Hamming	
	Messung	Literatur	Messung	Literatur	Messung	Literatur
6 dB bandwidth (DFT bins)	1,22	1,21	2,02	2,00	1,84	1,81
sidelobe level (dB)	-13,4	-13	-30,0	-32	-44,6	-43
fall off (dB/Oktave)	-6,0	-6	-18,4	-18	-5,3	-6

Tabelle 6.1: Eigenschaften der Fensterfunktionen im Frequenzbereich

Zusammenfassung

Beim Rechteckfenster wird der Signalvektor im Zeitbereich nicht verändert. Im Windowkernel liegt die Amplitude des ersten Nebenmaximums bei -13 dB. Das kann dazu führen, dass Signalanteile, die etwa den gleichen Frequenzabstand wie Haupt- und Nebenmaximum haben, nicht mehr getrennt werden können. Am besten schneidet hierbei das Hammingfenster ab, der langsame Abfall der weiteren Nebenmaxima führt allerdings dazu, dass Störsignale mit breitem Spektrum (z.B. Rauschen) relativ gut mitverstärkt werden. Diese werden daher durch das Hanningfenster wesentlich besser gedämpft.

6.5 Sweep-Signal und Kurzzeitspektren

6.5.1 Begrenzung des Sweep signals

In [1] wird ein sweep-signal definiert als ein Signal, das einen bestimmten Spektralbereich umfasst und im Zeitbereich so strukturiert ist, dass die Energie gleichmäßig verteilt ist.

Um solch ein Signal zu erzeugen, benutzen wir zunächst eine Sinusfunktion $y = \sin(2\pi f(t)t)$. Die Frequenz $f(t)$ soll dabei kontinuierlich mit der Zeit t innerhalb eines gewünschten Frequenzintervalls ansteigen oder abfallen. Zur Darstellung des Signals haben wir zunächst folgendes MATLAB Programm benutzt:

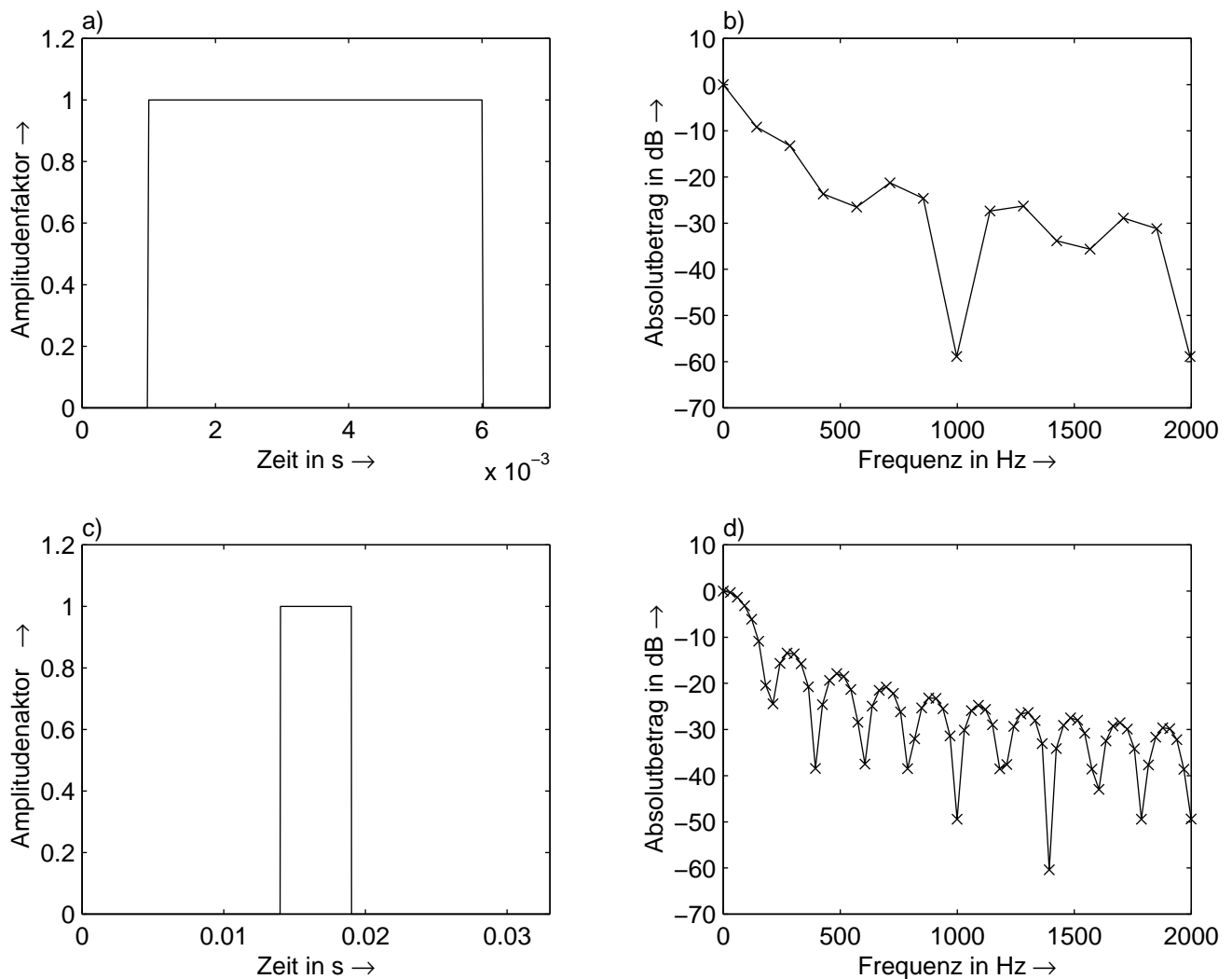


Abbildung 6.10: a) Rechteckfenster mit 50 Nullen beidseitig, b) Amplitudenspektrum dieses Fensters, c) Rechteckfenster mit 1000 Nullen auf jeder Seite, d) Amplitudenspektrum des Fensters in Abbildung 6.10c dargestellten Fensters

```
function y = sweep1(start,end,Abtastrate,Dauer);
%
%
% Die Dauer des Signals wird durch T [s] festgelegt.
% Die Frequenz f in Hz steigt linear mit der Zeit an.
% f l"auft genau von start bis end.
% y enth"alt den Signalvektor

t=0:1/Abtastrate:Dauer;      % Zeitvektor
tL=length(t);                % Laenge des Zeitvektors
f=start+(end-start)*t/Dauer; % Frequenz
y=sin(2*pi*f.*t);            % Signalvektor y
```

Das Spektrum dieses Signals ist breiter als das durch den Frequenzvektor f vorgegeben. Abbildung 6.13 zeigt

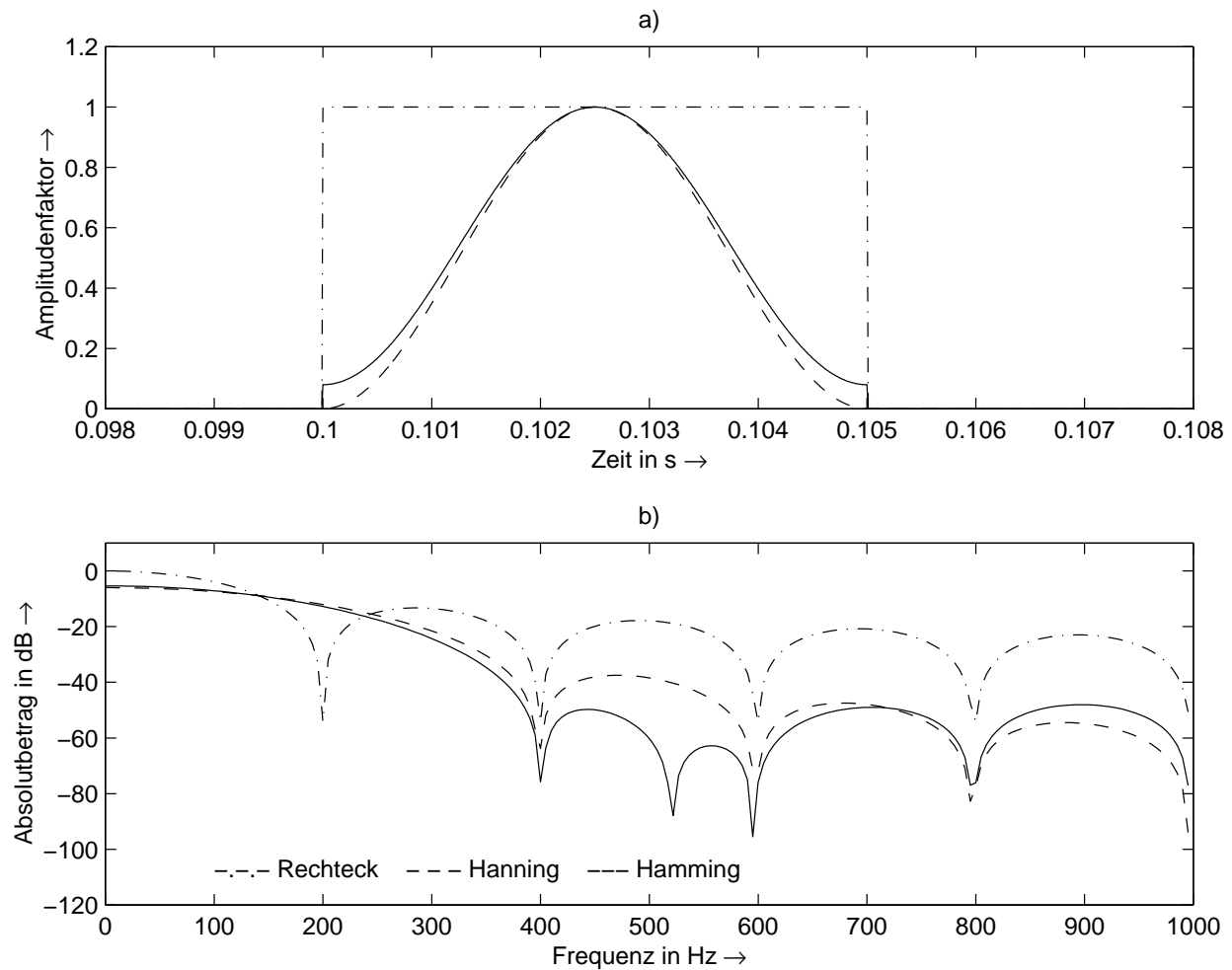


Abbildung 6.11: a) Hanningfenster, Hammingfenster und Rechteckfenster b) die DFT der Fensterfunktionen

die diskrete Fouriertransformierte des gesamten Signals mit gewählten Frequenzen für start (400 Hz) und end (2000 Hz). Die Signaldauer beträgt 1 s. Die Amplitude der Signalenergie ist in dB skaliert. Als Bezugspunkt dient die Mittenfrequenz zwischen start und end, für die wir 0 dB festlegen. Es ergibt sich ein Bereich zwischen der Frequenz start und der Frequenz end plus Intervalllänge, in dem die Amplitude größer als -6 dB bleibt. Also im Beispiel zwischen 400 Hz und 3600 Hz. Diesen Bereich wollen wir Spektralbereich nennen; von der Breite des Spektrums sprechen wir, wenn wir die Breite dieses Bereichs auf der Frequenzachse meinen. Die genauen Werte $f_l = 400 \text{ Hz}$ und $f_r = 3602 \text{ Hz} \pm 1 \text{ Hz}$ hängen unter anderen von der Signaldauer ab. Sie sind ebenfalls der Abbildung zu entnehmen. Sie werden von der Anzeigefunktion `spec01.m` geliefert (siehe Anhang). Außerhalb des Spektralbereichs ist die Signalenergie um kleiner als -6 dB und fällt für Frequenzen, die gegen Null oder unendlich gehen, schnell weiter ab. Innerhalb dieses Bereichs bilden sich natürlich auch noch Schwankungen der Amplitude aus: Da das Sweep signal zeitlich begrenzt ist, kann man es im Zeitbereich als ein unendliches Sweep signal multipliziert mit einer Rechteckfunktion analysieren. Im Frequenzbereich ergibt das dann eine Sinc-Funktion gefaltet mit der Fouriertransformierten des unendlichen Sweeps.

Um einen Sweep mit dem gewünschten Frequenzspektrum (400 Hz bis 2000 Hz) zu erhalten, muß der Vektor `f` entsprechend angepasst werden. Im gewählten Beispiel sollte `f` also von 400 Hz bis 1200 Hz laufen. Das Programm ändern wir daher folgendermaßen ab:

```
function y = sweep12(start,end,Abtastrate,Dauer);
%
%
```

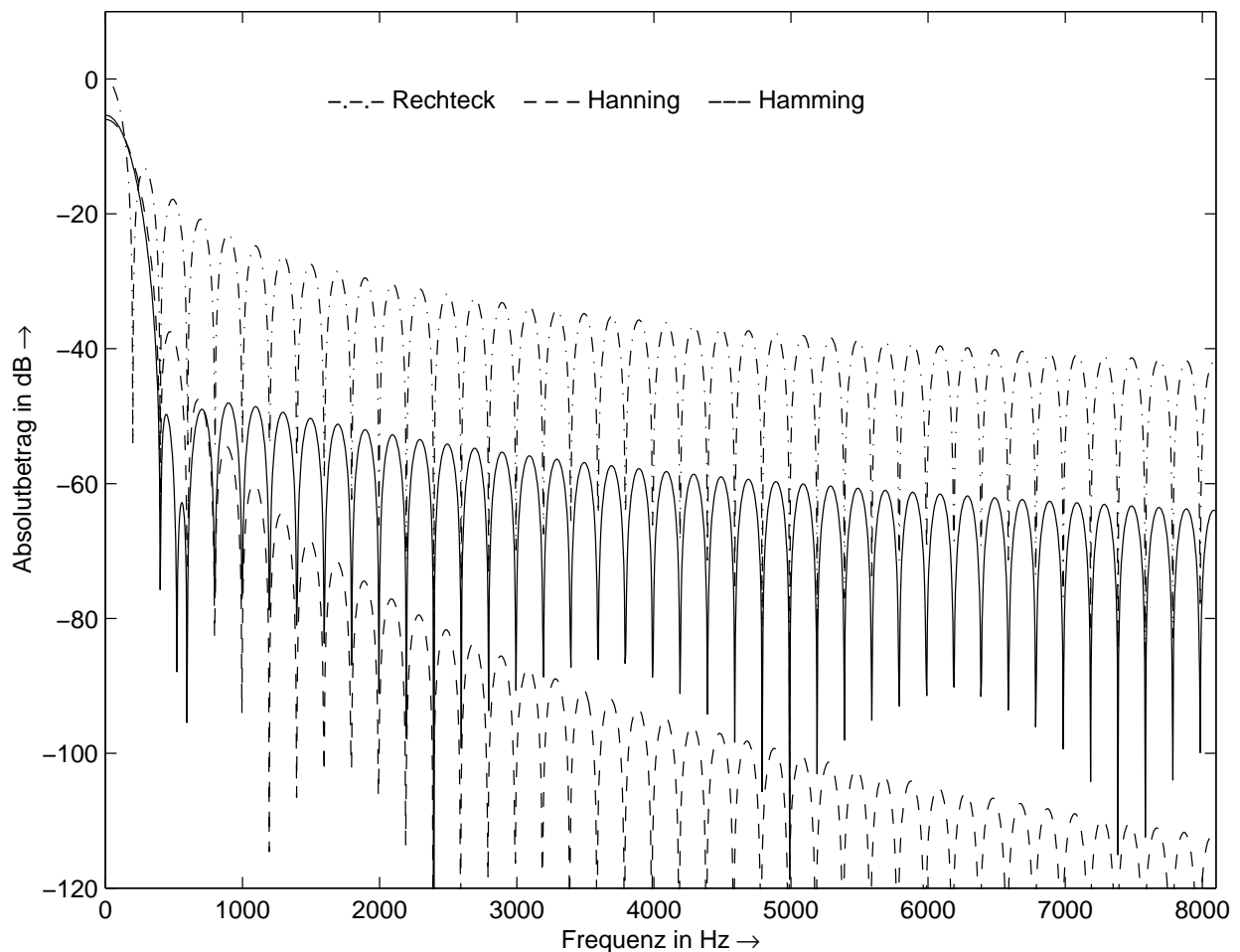


Abbildung 6.12: Nochmal ein Amplitudenspektrum aller 3 Fensterfunktionen, Abtastfrequenz: 100000 Hz, Fensterbreite: 5 ms

```
% Die Dauer des Signals wird durch T [s] festgelegt.
% Die Frequenz f in Hz steigt linear mit der Zeit an.
% f l"auft genau von start bis end.
% y enth"alt den Signalvektor

t=0:1/Abtastrate:Dauer;          % Zeitvektor
tL=length(t);                    % Laenge des Zeitvektors
f=start+(end-start)*t/(2*Dauer); % Frequenz laeuft nur noch bis start+1/2*(end-start)
y=sin(2*pi*f.*t);                % Signalvektor y
```

Wir erzeugen wieder einen Sweep von 1s Dauer und erhalten ein Signal mit einem Spektralbereich zwischen $400\text{ Hz} \pm 1\text{ Hz}$ und $2002\text{ Hz} \pm 1\text{ Hz}$ (Abbildung 6.13). Dieses Programm erzeugt also ein Sweepsignal mit dem gewünschtem Spektralbereich. Dies trifft genau genommen nur für relativ lange Signale zu. Wird die Signaldauer verringert, verbreitert sich natürlich das Spektrum. Die Abbildung 6.15 zeigt das Spektrum eines Sweeps von 0,1s Dauer. Die Abbildung 6.16 zeigt das Spektrum eines Sweeps der 1 ms dauert. Es wurden wieder die gleichen Werte für start und end verwendet. Man erkennt, dass das Spektrum des Signals immer mehr „verschmiert“ je kürzer es ist. Es fällt schon innerhalb des -6dB-Spektralbereichs schneller ab und hat größere Signalamplituden außerhalb dieses Bereichs. Das ist zu erwarten, denn faßt man das Signal wieder als Rechteckfunktion der Zeit

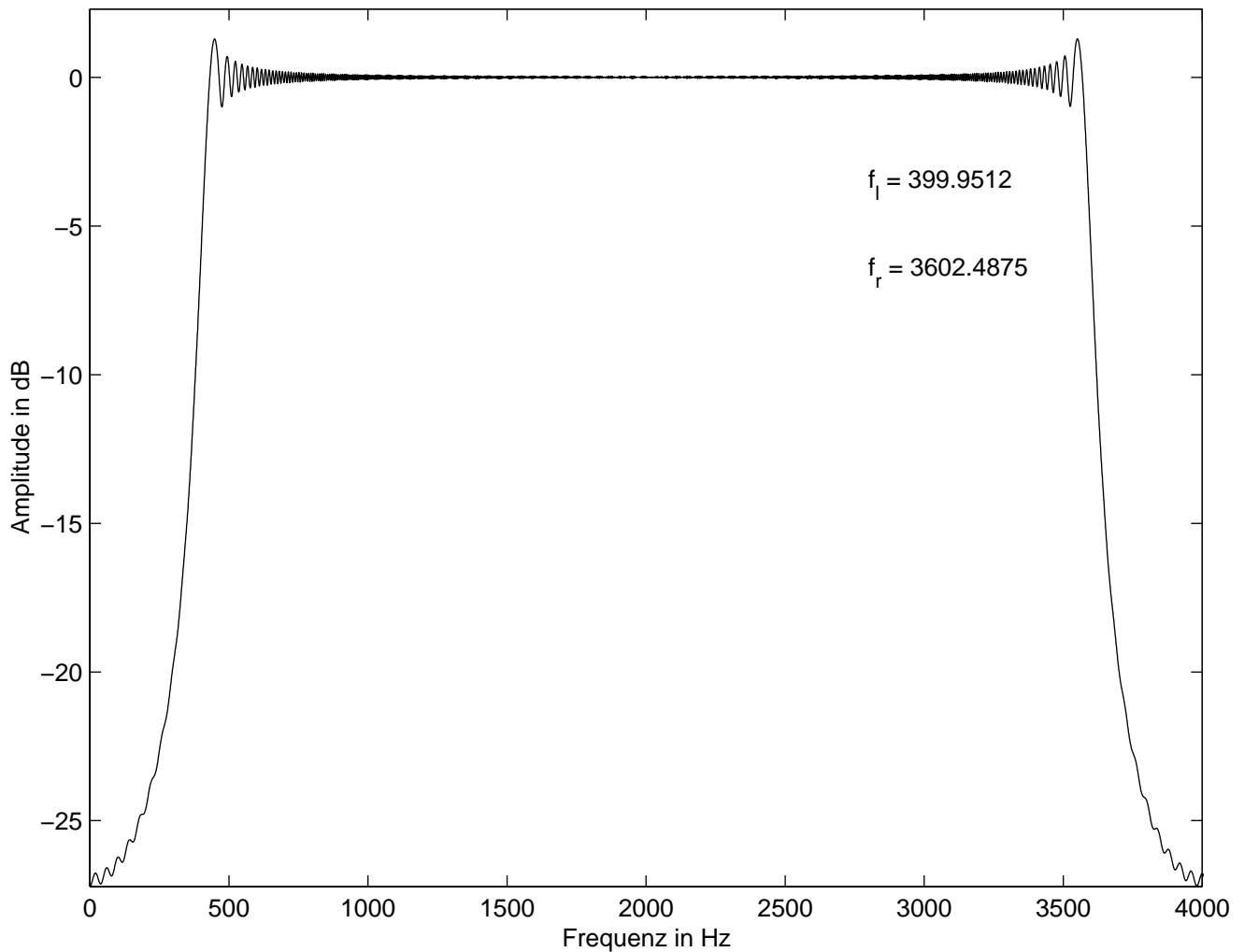


Abbildung 6.13: Fouriertransformierte von $y = \sin(2\pi f(t)t)$ $f(t) = 400 \dots 2000$ $t = 0 \dots 1$

multipliziert mit einem unendlichem Sweep auf, so wird bei kürzerer Signaldauer die Fouriertransformierte der Rechteckfunktion (also die Sinc-Funktion) immer breiter, d. h. die Nullstellen liegen weiter auseinander auf der Frequenzachse. Durch die Faltung ergibt sich damit insgesamt eine Verschmierung im Frequenzbereich.

6.5.2 Anwendung verschiedener Fensterfunktionen

In Abschnitt 6.5.1 haben wir ausschließlich ein Rechteckfunktion als Fenster für das gesamte Signal verwendet. Jetzt wollen wir zusätzlich betrachten, wie sich ein Hamming-Fenster und ein Hanning-Fenster im Spektrum auswirkt. Diese Funktionen sind folgendermaßen für ein Fenster mit N Samples definiert:

Hamming-Funktion :

$$hm(k) = \frac{1 - \cos\left(2\pi \frac{k}{N+1}\right)}{2} \quad k = 1, \dots, N \quad (6.1)$$

Hanning-Funktion :

$$hn(k) = 0,45 - 0,64 \cos\left(2\pi \frac{k-1}{N}\right) \quad k = 1, \dots, N \quad (6.2)$$

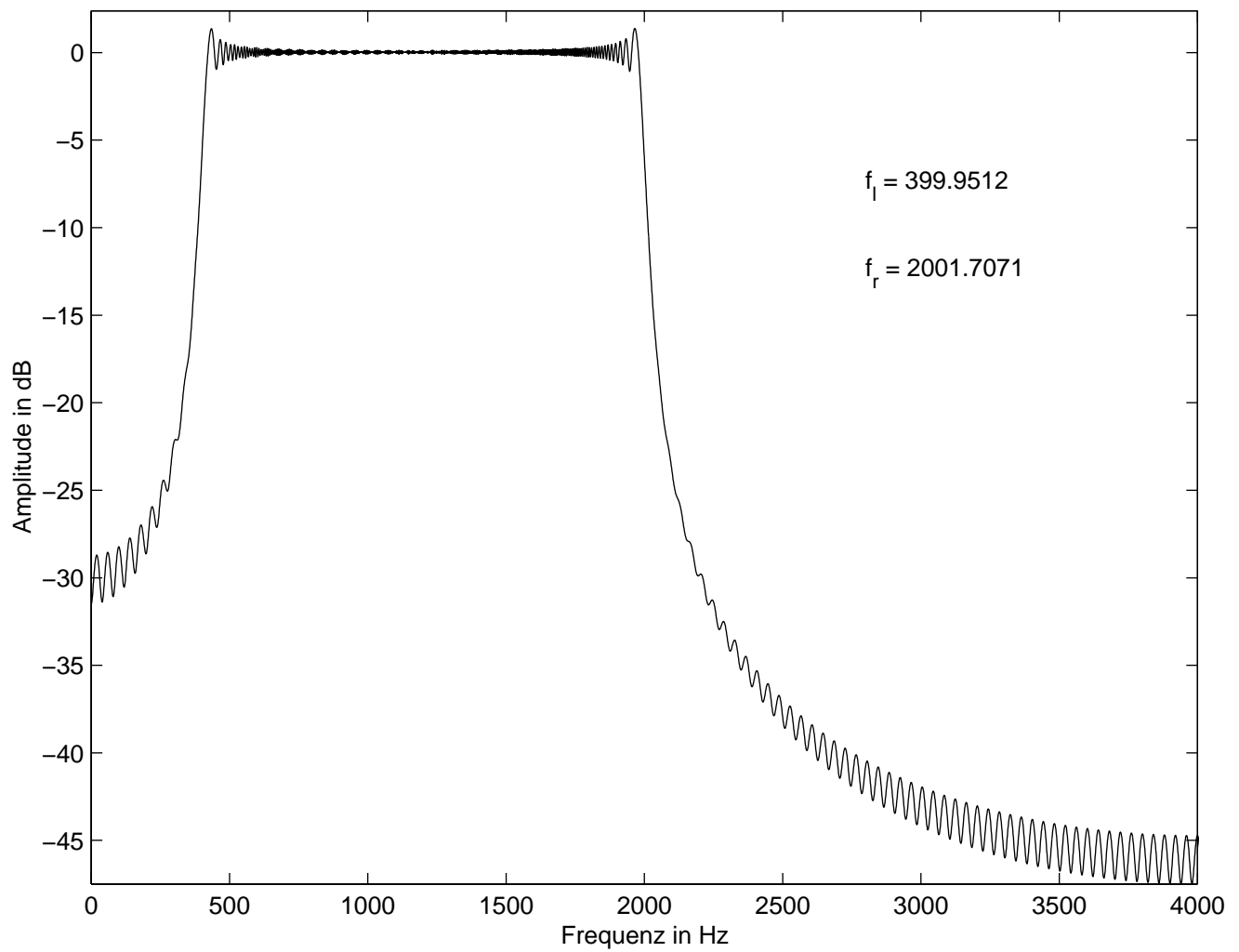


Abbildung 6.14: Fouriertransformierte von $y = \sin(2\pi f(t)t)$ $f(t) = 400 \dots 1200$ $t = 0 \dots 1$

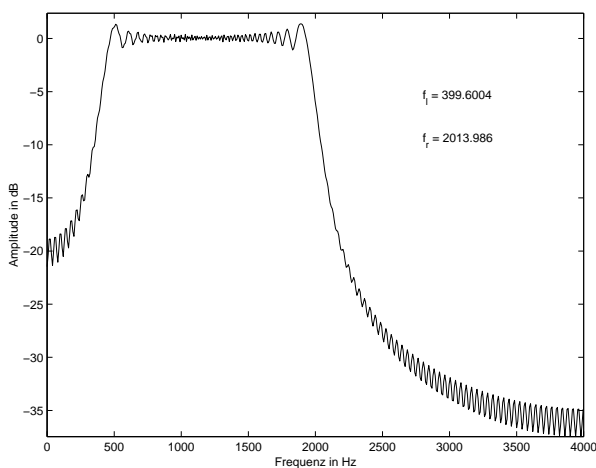


Abbildung 6.15: Spektrum eines Sweeps, Signaldauer 100ms

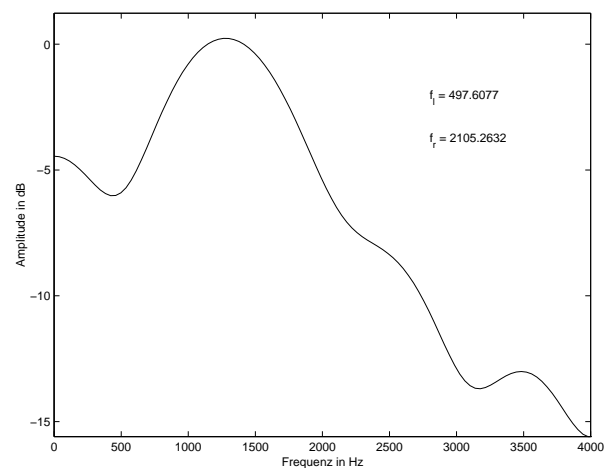


Abbildung 6.16: Spektrum eines Sweeps, Signaldauer 1ms

Diese Funktionen wenden wir jetzt auf das gesamte Sweepsignal (Zeitdauer 1s) mit dem Spektrum von 400Hz bis 2000Hz an. Das Ergebnis ist in Abbildung 6.17 zu sehen.

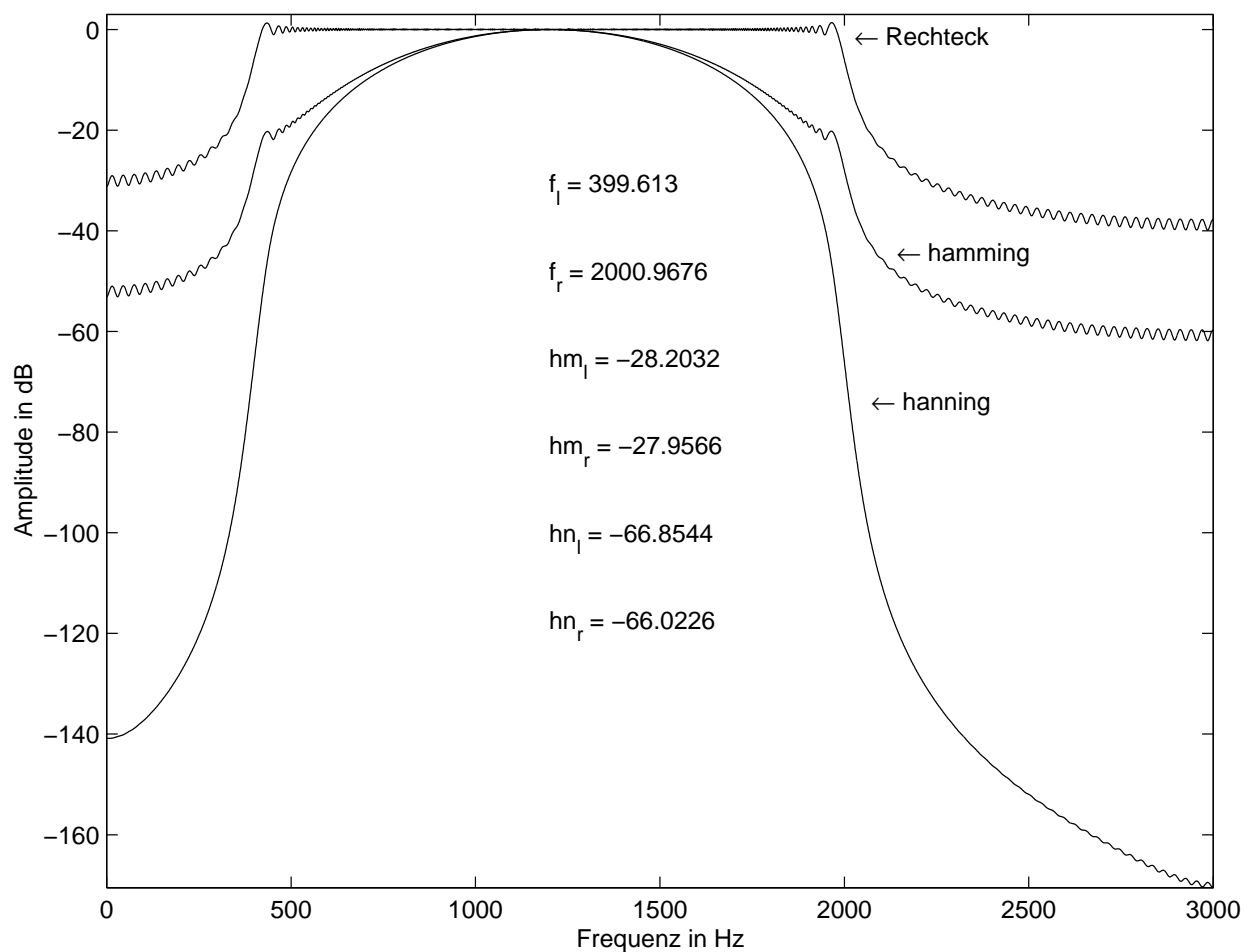


Abbildung 6.17: DFT mit Fenster eines Sweeps von 400 Hz bis 2000 Hz, Zeitdauer 1 s. Für die Bedeutung der Zahlenwerte siehe Text

In der Abbildung 6.17 ist der Spektralbereich wieder durch die Frequenzwerte in Hz f_l und f_r angegeben. Zum Vergleich der Fensterfunktionen betrachten wir die Amplitude(in dB) zu diesen Frequenzen beim Hamming- und beim Hanningfenster. Diese sind in der Abbildung ebenfalls mit hn_l und hn_r für das Hanningfenster und hm_l und hm_r für ein Hammingfenster angegeben. Man sieht daran, dass die Amplitude außerhalb des Spektralbereichs des Rechteckfensters beim Hammingfenster kleiner als -27dB und beim Hanningfenster kleiner als -66dB ist. Allerdings fällt die Amplitude innerhalb des Spektralbereichs des Rechteckfensters ziemlich stark ab; beim Hammingfenster weniger stark als beim Hanningfenster. Das Hanningfenster behält dabei in etwa seine Cosinusform aus dem Zeitbereich bei, während bei der Hammingfunktion stärkere Welligkeiten für Frequenzen nahe den Grenzen des Spektralbereichs auftreten.

6.5.3 Kurzzeitspektren des Sweepsignals (mit verschiedenen Fensterfunktionen)

Das bisher als Beispiel verwendete Sweepsignal wurde mit 6000 Samples pro Sekunde erzeugt. Der Sweep mit einer Sekunde Dauer hat also 6000 Samples. Wir wollen nun ein Kurzzeitspektrum dieses Signal erzeugen, also die Länge N der Fensterfunktionen variieren.

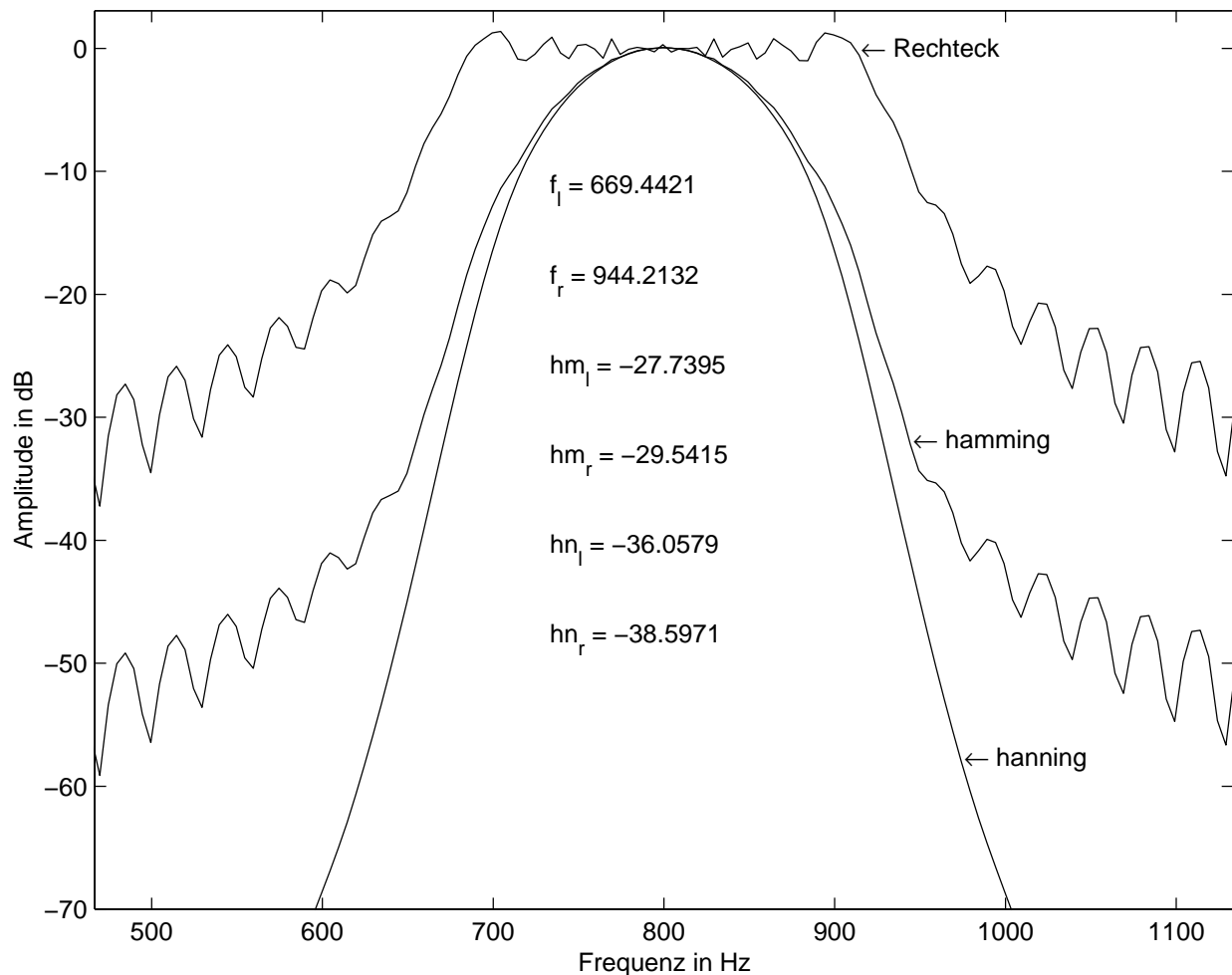


Abbildung 6.18: Spektrum eines verkürzten Ausschnitts (1000 Samples) des Sweepsignals. Insgesamt ist es 6000 Samples lang

Abbildung 6.18 zeigt die Frequenzanalyse für ein Fenster mit der Länge $N = 1000$. Das Fenster geht von Sample 1000 bis 2000 des Sweepsignals; das entspricht einem Frequenzbereich von 666,7 Hz bis 933,3 Hz. Durch Vergleich mit f_l und f_r in Abbildung 6.18 sieht man, dass auch hier bei einem kürzeren Signalausschnitt der Spektralbereich etwas größer wird. Es sind wieder die gleichen Größen wie in 6.5.2 eingetragen. Je kleiner die Fensterlänge ist, umso flacher werden die Flanken des Rechteckfensters im Spektrum. Außerdem wird die Amplitude außerhalb des Spektralbereichs des Rechteckfensters beim Hamming- und beim Hanningfenster (hm und hn) weniger stark abgeschwächt. Der Unterschied zwischen Hamming- und Hanningfenster nimmt auch ab, wie der Vergleich mit Abbildung 6.19 zeigt. Dort haben wir ein Spektrum eines Signalausschnitts mit der Fensterlänge $N = 500$ gezeichnet.

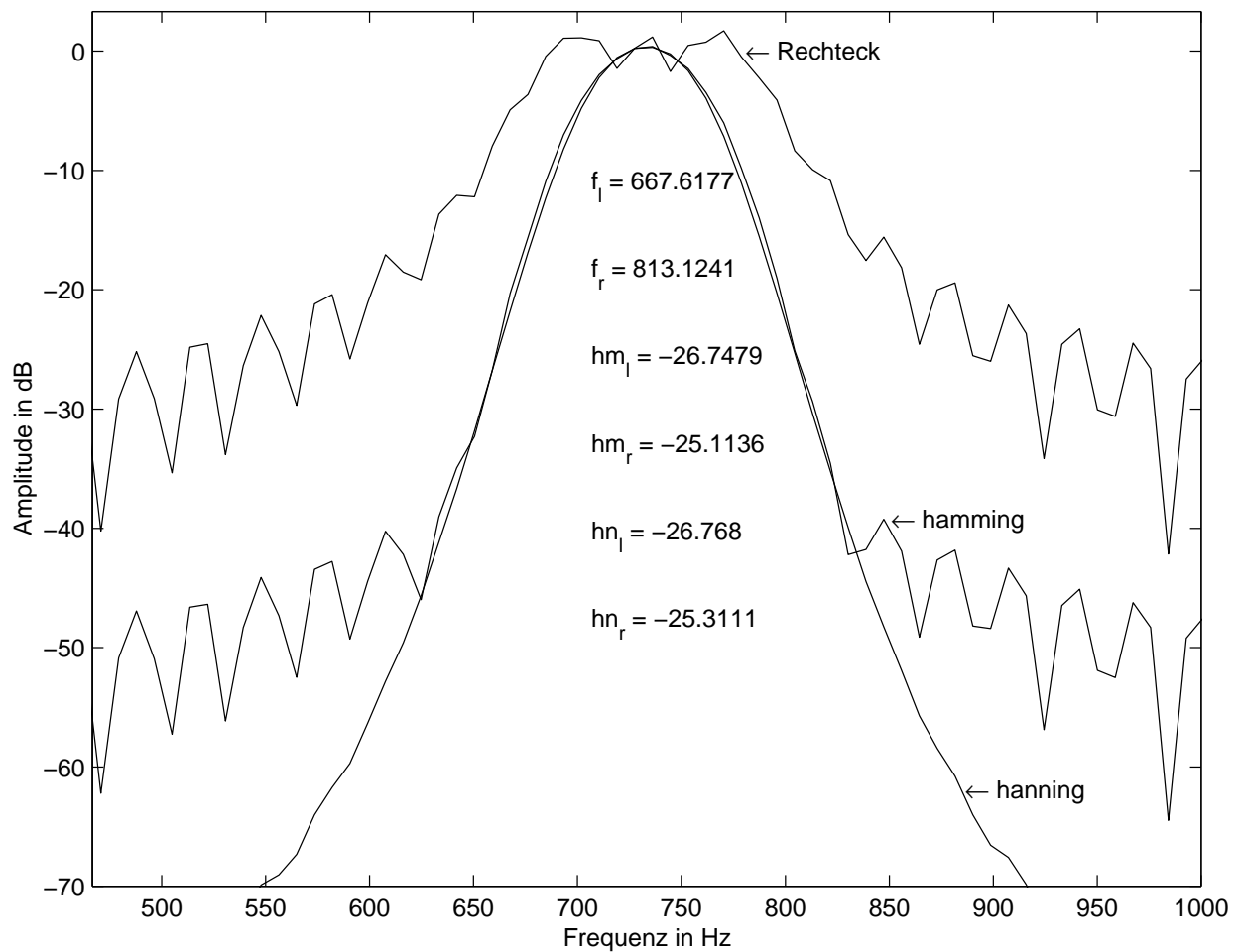


Abbildung 6.19: Spektrum eines verkürzten Ausschnitts (500 Samples) des Sweepsignals. Insgesamt ist es 6000 Samples lang

Literatur

- [1] AG MEDIZINISCHE PHYSIK FB PHYSIK UNIVERSITÄT OLDENBURG : *Anleitung für das DSP – Praktikum*
- [2] FORDWATER: *Advanced DSP* <http://www.bores.com>

7. Matlab - erste Schritte

Zur Durchführung der Übungen wird Matlab als spezielles Programm für die numerische Mathematik eingesetzt, das einfach zu bedienen ist und einen schnellen Einstieg in die Welt der numerischen Mathematik ermöglicht. Matlab stellt eine sehr gute Ergänzung zu den bekannten Programmen der symbolischen (analytischen) Mathematik (Maple, Mathematica) dar und kommt immer dann zum Einsatz, wenn keine geschlossenen analytischen Lösungen eines Problems angegeben werden können. Die grundlegende Datenstruktur von Matlab sind zweidimensionale Matrizen (daher auch der Name: MATrix LABoratory). Matlab kann mathematische Berechnungen in einer den Programmiersprachen C bzw. FORTRAN ähnlichen Syntax ausführen. Alle grundlegenden mathematischen Funktionen sowie spezielle Funktionen (Bessel etc.) sind enthalten. Die wesentlichen aus der numerischen Mathematik bekannten Algorithmen (Lösung linearer Gleichungssysteme, Eigenwerte/Eigenvektoren, Differentialgleichungen etc.) sind implementiert. Weiterhin beinhaltet das Programm Möglichkeiten zur grafischen Darstellung von Funktionen und Datensätzen. Es kann zweidimensionale xy-Plots, Plots in Polarkoordinaten sowie dreidimensionale Darstellungen erzeugen. Für spezielle Anwendungen (Bildverarbeitung, Signalverarbeitung, etc.) stehen sog. Toolboxes mit weiteren Funktionen zur Verfügung, die nicht im Grundpaket enthalten sind.

Die folgende Tabelle zeigt die Vor- und Nachteile von Matlab gegenüber Programmiersprachen wie C oder Fortran:

Vorteile	Nachteile
Schnelle Programmentwicklung	Unter Umständen langsamere Programmausführung im Vergleich zu optimierten C-Programmen
Sehr gute Grafikmöglichkeiten	
Viele wichtige Algorithmen sind bereits enthalten	
Portable Programmierung	

Im CIP-Raum des Fachbereichs Physik steht Matlab in der Version 5.3 mit der Signalverarbeitungs-Toolbox zur Verfügung. Das Programm läuft unter Windows NT. Da Matlab an Lizenzen gebunden ist, kann es nicht auf andere Rechner transferiert werden! Zur Nutzung ist ein Login für das Windows NT-Netzwerk des Rechenzentrums sowie die Zugehörigkeit zur Nutzergruppe fb8 notwendig (Details siehe <http://www.physik.uni-oldenburg.de/docs/cip/start.html>)

Im folgenden werden die grundlegende Struktur von Matlab sowie die wichtigsten Befehle erläutert. Daran anschließend finden sich kleinere Aufgaben, anhand derer der Umgang mit Matlab praktisch erprobt werden kann.

7.1 Bedienung von Matlab

Nach Aufruf von Matlab erscheint ein Kommandozeileneditor, in dem Kommandos eingegeben werden, die nach Drücken der Return-Taste (↵) vom Programm abgearbeitet werden. Bereits vorher abgearbeitete Befehlszeilen können durch Drücken der Pfeiltasten (↑↓) wieder hervorgeholt werden. Gibt man einen Anfangsbuchstaben ein und bedient dann die Pfeiltasten, erscheinen nur die vorherigen Befehle, die mit diesem Buchstaben beginnen. Die Kommandozeile kann so editiert werden, wie es aus Textverarbeitungsprogrammen bekannt ist (Markieren von Zeichen/Wörtern mit der Maus, Ersetzen und Einfügen von Zeichen/Wörtern, etc.).

Matlab sucht die auf der Kommandozeile eingegebenen Befehle in den im Suchpfad angegebenen Verzeichnissen. Mit dem Befehl `path` kann der Suchpfad ergänzt werden, falls z.B. ein eigenes Verzeichnis mit selbstgeschriebenen Kommandos in den Suchpfad aufgenommen werden soll.

Durch Eingabe des Befehls `diary dateiname` merkt sich Matlab alle eingegebenen Befehle sowie die „Antworten“ von Matlab in der angegebenen Datei. Damit kann die Abfolge der Befehle und Ergebnisse nachträglich nachvollzogen werden.

Matlab merkt sich die Werte aller Variablen, die während des Programmlaufs erzeugt werden im Speicher (dies wird als „Workspace“ bezeichnet). Mit den Befehlen `save` und `load` kann der Workspace in eine Datei abgespeichert und zur Fortsetzung der Arbeit später wieder aus der Datei geladen (rekonstruiert) werden. Mit dem Befehl `clear` können die Variablen im Workspace gelöscht werden. Die Befehle können durch Anhängen

von Variablennamen auf diese beschränkt werden. Zum Beispiel würde der Befehl `save test.dat x y` nur die Variablen `x` und `y` in der Datei `test.dat` speichern. Die Befehle `who` und `whos` listen die aktuell im Workspace vorhandenen Variablen auf, wobei `whos` genauere Informationen liefert.

Die Grafik erscheint in zusätzlichen Fenstern auf dem Bildschirm. Mit dem Befehl `figure` wird ein neues Grafikfenster erzeugt, so daß mehrere Grafiken erzeugt werden können.

7.2 Hilfefunktionen

Hilfe zu den einzelnen Befehlen erhält man durch Eingabe von `help Befehlsname`, wobei `help` ohne weitere Argumente eine Liste von möglichen Hilfetemen liefert. Ist der Name eines Befehls nicht bekannt, so empfiehlt sich die Suche unter relevanten Hilfetemen sowie die Benutzung des Befehls `lookfor Stichwort`, wobei das Stichwort etwas mit dem gesuchten Befehl zu tun haben sollte (in Englisch!).

Zugriff auf die gesamte Dokumentation erhält man, wenn der Befehl `helpdesk` eingegeben wird und ein Internet-Browser (Netscape) gestartet ist. Dann erscheint die Dokumentation im HTML-Format. Interessant für Anfänger ist insbesondere das Handbuch „Getting started“, das eine detaillierte Einführung in Matlab enthält.

Weiterhin können „Examples and Demos“ vom Help-Menü aus aufgerufen werden. Dort finden sich Beispiele nach Themengebieten geordnet. Meist werden auch die benutzten Matlab-Befehle mit angezeigt, so daß die Beispiele als Vorlage für eigene Anwendungen dienen können.

7.3 Variablen

Namen für Variablen können frei vergeben werden. Groß- und Kleinschreibung wird dabei berücksichtigt. Variablen werden durch Zuweisung eingeführt und müssen nicht deklariert werden. Der Typ der Variablen (Skalar, Vektor, Matrix) wird von Matlab anhand der Zuweisung festgestellt. Beispiele:

```
a = pi           % a wird der Wert Pi=3.1415... zugewiesen
b = [1 2]        % b wird der (Zeilen-)Vektor (1, 2) zugewiesen
c = [1 ; 2]       % c wird der (Spalten-)Vektor (1, 2) zugewiesen
d = [[1 2 3];[4 5 6i]] % d wird eine 2X3 Matrix zugewiesen
```

(Hinweise: Der Text nach dem %-Zeichen wird als Kommentar gewertet.

Die Variable `pi` ist fest vordefiniert.

Die Variable $i = \sqrt{-1}$ ist zur Eingabe komplexer Größen fest vordefiniert)

Der Wert einer Variablen kann jederzeit angezeigt werden, indem der Name der Variablen ohne Zusätze in der Kommandozeile eingegeben wird. Beispiele:

```
a           % Eingabe der Variablen
a = 3.14     % Antwort von MATLAB (Skalar)

b           % Eingabe der Variablen
b = 1  2     % Antwort von MATLAB (Zeilenvektor)

c           % Eingabe der Variablen
c = 1
    2       % Antwort von MATLAB (Spaltenvektor)

d           % Eingabe der Variablen
d = 1 2 3
    4 5 6i  % Antwort von MATLAB (2X3 Matrix)
```

Es lassen sich auch Elemente oder Bereiche von Matrizen und Vektoren anzeigen bzw. an Variablen zuweisen.
Beispiele:

```
d(1,3)          % Anzeigen der ersten Zeile, dritte Spalte
d = 3          % Antwort von MATLAB

d(1,2:3)       % Anzeigen der ersten Zeile, Elemente 2 bis 3
d = 2 3       % Antwort von MATLAB

z = d(2,1:3)   % Zuweisung von Zeile 2, Spalten 1 bis 3 an z
z = 4 5 6i    % Antwort von MATLAB

d(2,2) = 3     % Zuweisung des Wertes 3 an die Matrix d (Zeile 2, Spalte 2)
d = 1 2 3     % Antwort von MATLAB (2X3 Matrix)
    4 3 6i

d(1,:)         % Adressierung der ersten Zeile
= 1 2 3       % Antwort von MATLAB

d(1,1:2:3)     % Adressierung der ersten Zeile, jedes 2. Element)
= 1 3         % Antwort von MATLAB

d(1,[3 1])    % Adressierung der ersten Zeile, 3. und 1. Element)
= 3 1         % Antwort von MATLAB
```

(Hinweise: Jede Operation veranlaßt Matlab seine Arbeit auf dem Bildschirm zu dokumentieren. Dies kann unterbunden werden, indem einem Befehl ein Semikolon (;) nachgestellt wird.
Die Indizierung von Elementen von Vektoren/Matrizen beginnt immer bei Index 1.)

7.4 Operatoren

Der Hochkomma-Operator ' (**Achtung:** nicht mit einfachen oder doppelten Anführungsstrichen verwechseln!) führt eine komplexe Konjugation durch:

```
y = c';        % Wandle c in einen Zeilenvektor um
y              % Ergebnis zeigen
y = 1 2        % Antwort von Matlab

y = d';        % berechne komplex konjugierte Matrix
y              % Ergebnis zeigen
y = 1 4        % Antwort von MATLAB (3X2 Matrix)
    2 5
    3 -6i
```

Der Operator .' führt dagegen eine Transposition durch:

```
y = d.';       % berechne transponierte Matrix
y              % Ergebnis zeigen
y = 1 4        % Antwort von MATLAB (3X2 Matrix)
    2 5
    3 6i
```

Die bekannten Operatoren +,-,*,/ funktionieren unter Matlab wie bei einem Taschenrechner. Die Anwendung dieser Operatoren auf Matrizen und Vektoren kann jedoch zu Verwirrungen führen, da Matlab die Variablen links und rechts der Operatoren analysiert und die Operatoren 'intuitiv' einsetzt, d.h. je nach Dimension der Variablen skalare oder Matrixoperationen einsetzt. Beispiele:

```
y = a * a;     % multipliziere zwei Skalare
y              % Ergebnis zeigen
y = 9.8696     % Antwort von Matlab
```

```

y = b * c;      % multipliziere Zeilen und Spaltenvektor
y              % Ergebnis ist ein Skalarprodukt
y = 5

y = b .* c'     % multipliziere komponentenweise zwei Zeilenvektoren
y = 1 4        % Ergebnis ist wieder ein Zeilenvektor

y = d.^d        % komponentenweise Potenzierung
y = d^3        % Potenzierung der Matrix

```

In ähnlicher Weise lassen sich auch Matrizen untereinander und Matrizen mit Vektoren multiplizieren, addieren usw. Falls eine Operation komponentenweise durchgeführt werden soll, die auch als Matrixoperation möglich wäre, muß immer ein Punkt vor den Operator gesetzt werden. Bei allen Operationen nimmt es Matlab sehr genau mit den Dimensionen von Matrizen und Vektoren und gibt bei Unstimmigkeiten die Fehlermeldung "matrix dimensions must agree" aus.

7.5 Schleifen und Bedingungen

Wiederholte Operationen werden innerhalb von Schleifen durchgeführt. Die folgende Schleife berechnet einen Zeilen- und einen Spaltenvektor:

```

for i=1:5          % Beginn der Schleife
    p(i) = i^2;    % Einträge in einen Zeilenvektor
    q(i,1) = i^3;  % Einträge in einen Spaltenvektor
end               % Ende der Schleife

```

Es können mehrere Schleifen ineinander geschachtelt werden. Das Index-Inkrement kann durch den Doppelpunkt-Operator eingestellt werden:

```

for i=1:2:5        % Beginn der Schleife über ungerade Indizees
    q(i) = i;      % Einträge für ungerade Indizees
    q(i+1) = -i;   % Einträge für gerade Indizees
end               % Ende der Schleife

```

Wird der Befehl `break` innerhalb einer Schleife verwendet, so bricht die Ausführung der Schleife ab. Es gibt auch Schleifen, die ausgeführt werden, solange eine bestimmte Bedingung erfüllt ist:

```

x=100;
while( x > 1)      % Beginn der Schleife
    x = x/2;
end               % Ende der Schleife

```

(Hinweis: Schleifen in Matlab sind relativ langsam. Wann immer möglich sollten sie durch Vektoroperationen ersetzt werden.).

Folgende Operatoren stehen zur Erstellung von Relationen zur Verfügung:

Operator	Funktion
<code>==</code>	gleich
<code>~=</code>	ungleich
<code>></code>	größer
<code>>=</code>	größer gleich
<code><</code>	kleiner
<code><=</code>	kleiner gleich
<code>&</code>	logisches „und“
<code> </code>	logisches „oder“
<code>~</code>	logisches „nicht“

Die bedingte Ausführung von Befehlen erfolgt über sog. If-Konstruktionen:

```

if( x == 0 | x == 1 )           % Beginn der If-Konstruktion
    status = 1;
elseif( x < 0 & x ~= -1 )      % Achtung: elseif muß in einem Wort geschrieben werden
    status = -1;
else
    status = 0;
end                             % Ende der If-Konstruktion

```

7.6 Generierung von Matrizen/Vektoren

Der Doppelpunkt-Operator (:) dient zur Adressierung von Teilen von Matrizen/Vektoren (siehe Abschnitt *Variablen*), aber auch zur Generierung von Matrizen/Vektoren. Beispiele:

```

x = [1:10];                    % Erzeuge Zeilenvektor mit den ganzen Zahlen zwischen 1 und 10
x                                     % Ergebnis zeigen
x = 1 2 3 4 5 6 7 8 9 10      % Antwort von Matlab

x = [1:3 ; 1 3 4];            % Erzeuge (2X3)-Matrix
x                                     % Ergebnis zeigen
x = 1 2 3                      % Antwort von Matlab
    1 3 4

```

Es kann auch eine Schrittweite angegeben werden:

```

x = [0:pi/4:pi];              % Erzeuge Zeilenvektor mit Zahlen von 0 bis Pi,
                                % Schrittweite Pi/4.
x                                     % Ergebnis zeigen
x = 0 0.7854 1.5708 2.3562 3.1416  % Antwort von Matlab

```

Zur Generierung von Matrizen/Vektoren stehen weiterhin folgende Funktionen zur Verfügung:

```

zeros - generiert Matrizen/Vektoren und füllt sie mit Nullen
ones   - generiert Matrizen/Vektoren und füllt sie mit Einsen
rand   - generiert Matrizen/Vektoren und füllt sie mit gleichverteilten Zufallszahlen im
          Intervall (0,1)
randn  - generiert Matrizen/Vektoren und füllt sie mit normalverteilten Zufallszahlen
          (Mittelwert 0, Varianz 1);

```

Als Argumente werden jeweils die gewünschte Anzahl der Zeilen und Spalten übergeben. Beispiel:

```

x = randn(2,3); % erzeuge (2X3)-Matrix mit Zufallszahlen (normalverteilt)

```

7.7 Funktionen

Matlab enthält alle möglichen mathematischen und numerischen Funktionen. Alle Funktionen werden auf den gesamten Vektor/Matrix angewandt. Beispielsweise erzeugt folgender Befehl einen Vektor, der die Werte der Sinusfunktion zwischen 0 und 2 Pi mit einer Auflösung von Pi/10 enthält:

```

x = sin([0:pi/10:2*pi]);

```

Die folgende Tabelle enthält grundlegende und spezielle mathematische Funktionen:

Funktion	Beschreibung
sqrt(x)	Quadratwurzel
sin(x)	Sinus
asin(x)	Arcussinus

sinh(x)	Sinus Hyperbolicus
asinh(x)	Arcussinus Hyperbolicus
cos(x)	Cosinus
tan(x)	Tangens
exp(x)	Exponentialfunktion
expm(X)	Matrix-Exponentialfunktion
log(x)	natürlicher Logarithmus
logm(x)	Matrix-Logarithmus
log10(x)	Zehnerlogarithmus
rem(i,n)	Modulofunktion
round(x)	Runden
floor(x)	Abrunden
ceil(x)	Aufrunden
inv(X)	Inverse einer Matrix berechnen
fft(x)	schnelle Fouriertransformation
abs(x)	Absolutbetrag
real(x)	Realteil
imag(x)	Imaginärteil
Sum	Summe über Vektorelemente
Prod	Produkt der Vektorelemente
Max	Maximumssuche
Min	Minimumssuche
Mean	Mittelwert
Std	Standardabweichung
Cov	(Ko-)Varianz
Median	Medianwert
Bessel	Besselfunktionen
Erf	Gaussche Fehlerfunktion
Gamma	Gammafunktion
Size	berechne Größe einer Matrix/Vektor
Hist	Histogramm
Sort	Sortieralgorithmus

7.8 Visualisierung

Zur grafischen Darstellung von Daten stehen die verschiedensten Visualisierungsmöglichkeiten zur Verfügung. Die einfachste Möglichkeit sind xy-Plots, bei denen Datenpunkte (x,y) in einem kartesischen Koordinatensystem dargestellt werden:

```
x = [0:pi/10:2*pi];
y = sin(x);
plot(x,y);
```

Diese Befehle erzeugen eine Periode der Sinusfunktion als Grafik. Die folgende Tabelle enthält weitere für die Visualisierung wesentliche Befehle:

Funktion	Beschreibung
plot(x,y)	xy-Plot, lineare Achsen
loglog(x,y)	xy-Plot, doppelt logarithmische Darstellung
Semilogx(x,y)	xy-Plot, logarithmische x-Achse
Semilogy(x,y)	xy-Plot, logarithmische y-Achse
title('text')	Setzen des Bildtitels
xlabel('text')	Setzen der x-Achsenbeschriftung
ylabel('text')	Setzen der y-Achsenbeschriftung
Axis	Setzen der Achsen-Wertebereichs
polar(theta,rho)	Polarplot
Contour(z)	2D-Höhenliniendarstellung der Matrix z

image(z)	2D-Farbdarstellung der Matrix z
imagesc(z)	wie image, jedoch mit Skalierung der Daten
pcolor(z)	ähnlich wie image(z)
Colormap	Setzen der Farbskala bei Farbdarstellung
plot3(x,y,z)	3D-Darstellung von Vektoren
Contour3(z)	3D-Höhenliniendarstellung der Matrix z
mesh(z)	3D-Gitterplot der Matrix z
surf(z)	3D-Gitterplot der Matrix z mit Farbdarstellung

Zur Erprobung der 2D- und 3D-Darstellungen kann mit dem Befehl **peaks** eine Testmatrix erzeugt werden. Beispielsweise plottet der Befehl `imagesc(peaks)` eine 2-D-Farbdarstellung der von **peaks** erzeugten Matrix.

7.9 Eingabe und Ausgabe

Der Befehl `input` liest eine Eingabe von der Tastatur:

```
x = input('Wert von x eingeben: ')
```

Dieser Befehl druckt den angegebenen Text auf den Bildschirm und wartet auf eine Eingabe von der Tastatur. Die Eingabe wird der Variablen `x` zugewiesen.

Der Befehl `disp` druckt Texte und Werte von Variablen auf dem Bildschirm:

```
disp('Der Wert von x ist: ')
disp(x)
```

Diese Befehle drucken den Text sowie den Wert der Variablen `x`.

Für die formatierte Ausgabe steht auch der Befehl `fprintf` zur Verfügung wie er aus der Programmiersprache C bekannt ist. Ebenso stehen die Befehle `fopen`, `fread` und `fwrite` und `fclose` zum Lesen von bzw. Schreiben in Dateien zur Verfügung (Hinweis: nicht mit den Befehlen `load` und `save` zu verwechseln, diese speichern den Workspace in einem speziellen Matlab-eigenen Datenformat ab).

7.10 Erstellung eigener Befehle („M-Files“)

In Matlab lassen sich auf einfache Weise eigene Programme und Funktionen selbst schreiben. Hierfür schreibt man ein sog. **Script**. Das ist eine Textdatei, in der Matlab-Befehle zusammengefaßt werden. Der Name der Datei wird von Matlab als Name des Befehls interpretiert, die Extension `.m` der Datei ist festgelegt. Daher heißen diese Dateien auch M-Files. Wie solche M-Files aufgebaut sein müssen, soll im folgenden anhand eines Beispiels erläutert werden. Für weitere Informationen hierzu kann auch die Matlab Hilfe-Funktion verwendet werden. Dazu muß `help script` oder `help function` eingegeben werden.

(Hinweis: Damit Matlab die selbstgeschriebenen M-Files findet, muß das Verzeichnis, in dem sich die Dateien befinden, im Suchpfad sein (Kommando `path`) oder die Dateien müssen im aktuellen Verzeichnis sein (Kommando `cd`).)

Ein Script wird dadurch erstellt, daß die gewünschten Befehle zeilenweise in die Textdatei geschrieben werden. Dazu kann ein normaler Texteditor (z.B. `notepad`) verwendet werden, aber auch der Matlab-eigene Texteditor (Aufruf des Editors durch Auswählen von **Open** oder **New** im Menüpunkt **File**). Das Skript wird durch Eingabe des Dateinamens auf der Kommandozeile gestartet. Matlab arbeitet dann alle Befehle ab und behält die im Skript erzeugten Variablen im Workspace.

(Hinweis: Wenn ein M-File geändert wird, liest Matlab es u.U. nicht neu ein, es wird dann die alte Version ausgeführt (Fehler bei Netzwerk-Installation von Matlab, z.B. im CIP-Raum). Das Neu-Einlesen wird erzwungen, wenn man den Befehl `clear all` eingibt, was aber den Nebeneffekt hat, daß alle Variablen im Workspace gelöscht werden.

Man kann aus einem Skript eine Funktion machen, die sich wie eine Matlab-eigene Funktion verhält. Dazu sind bestimmte Formalia einzuhalten, die im folgenden anhand eines Beispiels erläutert werden sollen:

```
function [mean,stdev] = stat(x)
%
% [mean,stdev] = stat(x)
%
% Diese Funktion berechnet Mittelwert und Standardabweichung eines Datensatzes
%
% Übergabeparameter:
%   x - Datensatz als Vektor
%
% Rückgabewerte:
%   mean - Mittelwert
%   stdev - Standardabweichung
%

n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);
```

Die erste Zeile beginnt mit dem Schlüsselwort `function` und deklariert den Namen der Funktion (hier: `stat`), die Übergabeparameter in runden Klammern (hier: `x`) sowie die Rückgabewerte (hier: `mean` und `stdev`). Der nachfolgende Kommentarblock wird von Matlab ausgegeben, wenn die Hilfefunktion für dieses Kommando aufgerufen wird (`help stat`). Danach erfolgt die eigentliche Funktion, die aus beliebigen Matlab-Kommandos aufgebaut sein kann. Wichtig ist, daß dabei die Rückgabewerte aus den Übergabeparametern berechnet werden. Dabei können auch Hilfsvariablen generiert werden (hier: `n`). Die Funktion verwaltet ihren eigenen Workspace, d.h. der Funktion sind nur die Übergabewerte bekannt und es werden nur die Rückgabewerte an den übergeordneten Workspace zurückgegeben. Alle Hilfsvariablen sind nach Beendigung der Funktion gelöscht. Die Funktion kann nun folgendermaßen von der Kommandozeile aus aufgerufen werden:

```
noise = randn(1,100);           % Zufallszahlen generieren
[mnoise,snoise] = stat(noise);  % Statistik berechnen
disp('Mittelwert und Standardabweichung sind:');
mnoise
snoise
```

7.11 Aufgaben

Anmerkung: Speichere die Lösungen aller Aufgabenstellungen in Skript-Dateien ("M-Files").

I. Datengenerierung

- generiere einen Zeitvektor von 0ms bis 10 ms bei einer Abtastfrequenz von 10 kHz.
- generiere über der Zeitbasis einen Sinus mit einer Frequenz von 1 kHz,.
- plotte Sinus aus b. über der Zeitbasis aus a.
- wie c., jedoch nur jeden 2. Abtastwert und jeden 4. Abtastwert.
- speichere die generierten Daten in einem Datenfile (Hinweis: benutze dazu den Befehl "save")
- ersetze die erste Beispiel-Schleife im Abschnitt *Schleifen und Bedingungen* durch Vektoroperationen.

II. Umgang mit Matrizen

- definiere die Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

und die Vektoren $b = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$ und $d = \begin{pmatrix} 1 & 2 \end{pmatrix}$ als Zeilenvektoren.

- b. berechne Ab^T ; bA ; $b * c$ als Skalarprodukt und (!) komponentenweise.
- c. löse das lineare Gleichungssystem $Ax=b^T$ (Hinweis: "help slash") und überprüfe die Lösung durch Einsetzen.
- d. berechne $A_2 d^T$, wobei $A_2 = \begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix}$ (A_2 als Untermatrix von A schreiben!).

III. 2D- und 3D-Darstellung von Matrizen

- a. generiere eine 2-dimensionale Gauss-Glocke mit Varianz von 8 auf einer (25x25)-Matrix.
- b. plote die Matrix in 2-D-Farbdarstellung mit Höhenlinien.
- c. wie b, jedoch 3D-Farbdarstellung.
- d. **Optionale Aufgabe:** Wie läßt sich a. ohne FOR-Schleife lösen?

Die Beispiele im Menu ,Help->Examples and Demos' sind sehr instruktiv und man kann dort viele Tricks und Kniffe der Matlab-Programmierung lernen (die Quelltexte für alle Beispiele sind enthalten!). Die Durchsicht dieser Beispiele wird empfohlen.