Signalverarbeitung mit MATLAB® und Simulink®

Anwendungsorientierte Simulationen

von
Prof. Dr.-Ing. Josef Hoffmann,
Prof. Dr. Franz Quint

Oldenbourg Verlag München Wien

Prof. Dr.-Ing. Josef Hoffmann war Professor an der Fakultät Elektro- und Informationstechnik (EIT) der Hochschule Karlsruhe – Technik und Wirtschaft; er unterrichtet dort noch mit Lehrauftrag für die Fachgebiete Messtechnik, Technische Kommunikation, Netzwerke und Filter.

Prof. Dr. Franz Quint ist Professor an der Fakultät Elektro- und Informationstechnik (EIT) der Hochschule Karlsruhe – Technik und Wirtschaft; dort ist er zuständig für die Lehrveranstaltungen Nachrichtentechnik, Verarbeitung mehrdimensionaler Signale, Digitale Signalprozessoren und Informationstheorie und Codierung.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 450 51-0
oldenbourg.de

Lektorat: Anton Schmid
Herstellung: Anna Grosser
Coverentwurf: Kochan & Partner, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: Grafik + Druck, München
Bindung: Thomas Buchbinderei GmbH, Augsburg

ISBN 978-3-486-58427-1

Vorwort

Die digitale Signalverarbeitung hat in den letzten Jahrzehnten explosive Fortschritte sowohl in der Theorie als auch in den Anwendungen erfahren. Dies ist hauptsächlich der rasanten Entwicklung in den Technologien der digitalen Hardware und Rechnertechnik sowie der entsprechenden Software geschuldet. Was heutzutage alles in einem mobilen Kommunikationsgerät („*Handy*“) an Funktionalität integriert ist, kann dabei als Beispiel angesehen werden.

In allen elektrotechnischen, nachrichtentechnischen und den verwandten Studiengängen an Hochschulen werden Vorlesungen in digitaler Signalverarbeitung angeboten. Die Theorie der Lehrveranstaltungen wird dabei in vielen Fällen mit Simulationen in MATLAB begleitet. Es ist so möglich, Beispiele zu untersuchen, die praktisch relevant sind und über die einfachen, analytisch lösbaren Fälle früherer Zeiten hinausgehen.

In der Industrie hat sich die MATLAB-Produktfamilie in Forschung und Entwicklung zu einem Standardwerkzeug durchgesetzt. Sie wird von den wissenschaftlichen Voruntersuchungen über die Algorithmenentwicklung bis hin zur Implementierung auf einer dedizierten Hardware eingesetzt. Oftmals wird aus dem Simulationsprogramm des Verfahrens automatisch das in der Hardware zu implementierende Programm erzeugt. Diese durchgängige Entwicklungskette bietet den Vorteil effizienter Produktentwicklungszyklen, weil Abweichungen zwischen Simulation und auf der Hardware implementiertem Programm vermieden werden. Nachträglich erforderliche Änderungen können schneller implementiert und untersucht werden.

Damit ist die Verwendung der MATLAB-Produktfamilie in der Lehre nicht nur dem Verständnis der Theorie förderlich, sondern sie ermöglicht den Absolventen von Ingenieurstudiengängen auch einen raschen Zugang zur industriellen Praxis.

Das vorliegende Buch richtet sich vorwiegend an Ingenieurstudenten der Universitäten und Hochschulen für Technik, die eine Vorlesung zum Thema Signalverarbeitung hören. Mit Hilfe der Simulationen in vorliegendem Buch kann die Theorie mit praktischen Beispielen veranschaulicht und in Anwendungen eingesetzt werden.

Die vorgeschlagenen Simulationen wurden so gestaltet, dass sie häufig eingesetzten Anwendungen der Signalverarbeitungspraxis entsprechen. Die Studierenden können sie kreativ erweitern und eigene Konzepte untersuchen, bzw. ihre eigenen Fragen oder Schwierigkeiten beim Verstehen der Theorie mit ähnlichen Simulationen klären.

Das Buch richtet sich auch an Fachkräfte aus Forschung und Industrie, die im Bereich der Signalverarbeitung tätig sind und die MATLAB-Produktfamilie einsetzen oder einzusetzen beabsichtigen. Die leistungsfähigen Funktionen dieser Software werden mit typischen Verfahren der Signalverarbeitung dargestellt und untersucht. Die Simulationen gehen weiter als die Demonstrationsbeispiele aus der Begleitdokumentation von MATLAB, sind ausführlicher beschrieben und basieren auf der didaktischen Erfahrung der Autoren. Sie sind so gestaltet, dass man sie als Bausteine für weitere aufwändigere Anwendungen und Systeme verwenden kann.

Die MATLAB-Produktfamilie besteht aus der MATLAB-Grundsoftware, aus verschiedenen „Toolboxen“ und aus dem graphischen Simulationswerkzeug Simulink. MATLAB ist eine

leistungsfähige Hochsprache, die Grundfunktionen zur Manipulation von Daten, die in mehrdimensionalen Feldern gespeichert sind, enthält. Der Name MATLAB ist ein Akronym für "MATrix LABoratory" und bezieht sich auf die ursprünglich vorgesehene Anwendung, nämlich das Rechnen mit Matrizen und Datenfeldern.

Für verschiedene Fachgebiete gibt es Erweiterungen der MATLAB-Software in Form sogenannter „Toolboxen". Diese sind Sammlungen von MATLAB-Programmen, die zur Lösung spezifischer Aufgaben des entsprechenden Fachgebietes entwickelt wurden. Eine davon ist auch die *Signal Processing Toolbox*, die ein zentrales Thema dieses Buchs ist.

Eine Besonderheit dieses Buchs im Vergleich zu anderen MATLAB-Büchern besteht darin, dass auch Simulink als Erweiterung von MATLAB intensiv einsetzt wird. In Simulink werden mit relativ wenig Programmieraufwand Systemmodelle mit Hilfe von Blockdiagrammen, wie sie im Lehrbetrieb und in der Entwicklung üblich sind, erstellt. Funktionsblöcke aus verschiedenen Bibliotheken werden so verbunden, dass ein Modell des Systems entsteht.

Die Simulink-Bibliotheken stellen eine Vielzahl von Funktionsblöcken zur Verfügung: Blöcke für Signalquellen, für lineare und nichtlineare Funktionen, für zeitdiskrete Systemkomponenten, für Senken mit denen die Variablen der Modelle inspiziert werden können, usw. Ähnlich wie die Toolboxen MATLAB erweitern, sind in Simulink sogenannte „Blocksets" verfügbar, die spezifische Funktionen für verschiedene Fachgebiete bereitstellen. Im vorliegenden Buch liegt der Schwerpunkt auf den Blöcken des *Signal Processing Blockset*.

Das Simulink-Modell ist auch eine graphische Abbildung des Systems, die leicht zu verstehen, zu ändern und zu untersuchen ist. Die Erstellung von Simulink-Modellen ist in der Regel weniger fehleranfällig als die Programmierung mit MATLAB. Die notwendige Flexibilität gewinnt man durch die Erweiterung vorhandener oder die Programmierung neuer Blöcke. Zur übersichtlichen Gestaltung des Modells können Blöcke hierarchisch zu übergeordneten Blöcken zusammengefasst werden. Aus Simulink-Modellen können automatisch C- oder VHDL[1]-Programme generiert werden, die nach dem Übersetzen auf dedizierter Hardware lauffähig sind.

Das Buch enthält sieben Kapitel, in denen wichtige Aufgabestellungen in der Signalverarbeitung mit Simulationen begleitet werden. Bei einigen Themen werden auch die theoretischen Hintergründe erläutert, so dass man die Funktionen der MATLAB-Produktfamilie verstehen und einsetzen kann. Allerdings ist es nicht Ziel des Buchs, Lehrbuch für die Theorie der digitalen Signalverarbeitung zu sein. Vielmehr soll es der Vertiefung und dem Verständnis ausgewählter Verfahren dienen und Einsatz- und Anwendungsmöglichkeiten der MATLAB-Software in der Lehre und in Forschung und Entwicklung aufzeigen.

- Im ersten Kapitel werden Experimente zur Analyse und Synthese analoger Filter präsentiert. Diese Filter sind an der Schnittstelle zwischen der analogen Natur und der digitalen Signalverarbeitung notwendig. Es werden die von analogen Filtern verursachten Verzerrungen mit Simulationen untersucht und daraus praktische Aspekte abgeleitet. So ist z.B. die Amplitudenverzerrung durch die D/A-Wandler bei der Rekonstruktion zeitkontinuierlicher Signale aus den zeitdiskreten Abtastwerten dargestellt.

- Die Entwicklung digitaler Filter, ein Hauptthema der digitalen Signalverarbeitung, wird im zweiten und dritten Kapitel mit Hilfe der Werkzeuge aus der *Signal Processing Tool-*

[1] *Very High Speed Integrated Circuit Hardware Description Language*

box und aus der *Filter Design Toolbox* beschrieben. Die klassischen Entwicklungsverfahren wie z.B. das Parks-McClellan-Verfahren werden im zweiten Kapitel untersucht. Im dritten Kapitel werden spezielle Verfahren anhand von Simulationen erläutert. Dabei wird auch der für die Praxis wichtige Aspekt der Filter im Festkomma-Format ausführlich vorgestellt. Hierfür wird die *Fixed-Point Toolbox* einbezogen.

- Die Multiraten-Signalverarbeitung, die in vielen Anwendungen eingesetzt wird, wird im vierten Kapitel mit praktischen Beispielen begleitet. Zu Beginn werden die beiden Hauptthemen dieses Gebiets, die Dezimierung und die Interpolierung vorgestellt und mit Experimenten vertieft. Danach wird auf die Realisierung der Dezimierung und Interpolierung mit Polyphasenfiltern eingegangen. Auch Multiratenfilterbänke werden hier behandelt.

  Im letzten Teil dieses Kapitels werden die sogenannten CIC[2]- und IFIR[3]-Filter für die Dezimierung und Interpolierung zeitdiskreter Signale beschrieben und untersucht. CIC-Filter werden häufig in Hardware implementiert, weil sie ohne Multiplikationen realisierbar sind. Mit IFIR-Filtern werden Tiefpassfilter mit steilen Flanken und schmaler Bandbreite erzeugt, die sich ohne großen Aufwand implementieren lassen.

- Experimente zur Analyse und Synthese adaptiver Filter werden im fünften Kapitel vorgestellt. Dabei werden die Filter, die in Simulink-Blöcken und in den MATLAB-Objekten für adaptive Filter implementiert sind, eingesetzt.

- Das sechste Kapitel enthält eine der Thematik dieses Buches angepasste kompakte Darstellung von MATLAB. Es geht auf die Grundfunktionen von MATLAB und die Werkzeuge, die in den Experimenten eingesetzt werden, ein. Gute einführende Kenntnisse können auch über die Hilfe-Seiten *Getting Started* der MATLAB-Software erworben werden.

- Auf besonders beachtenswerte Aspekte von Simulink wird im letzten Kapitel eingegangen. Auch hier ist es ratsam, dass Simulink-Anfänger die Hilfe-Seiten *Getting Started* von Simulink vorher durcharbeiten.

Der Leser wird ermutigt, bei der Arbeit mit diesem Buch die vorgestellten Simulationen selbst in MATLAB oder Simulink durchzuführen, sie zu erweitern oder für seine Zwecke zu verändern. Hierfür kann er die Quellen aller für das Buch entwickelten MATLAB-Programme und Simulink-Modelle von der Internet-Seite des Oldenbourg-Verlages `http//www.oldenbourg-wissenschaftsverlag.de` beziehen.

Aus dem sehr großen Umfang der MATLAB-Produktfamilie werden in diesem Buch Funktionen folgender Werkzeuge eingesetzt: MATLAB und Simulink als Grundsoftware, *Signal Processing Toolbox*, *Signal Processing Blockset*, *Filter Design Toolbox*, *Fixed-Point Toolbox*, *Simulink Fixed Point* und *Wavelet Toolbox*. Hinzu kommen noch verschiedene mit MATLAB mitgelieferte Entwicklungs- und Analysewerkzeuge wie z.B. das *Filter Design and Analysis Tool*, kurz FDATool, für die Entwicklung und Analyse von Filtern oder das *Filter Visualization Tool* zur Visualisierung der Eigenschaften der Filter.

Die Handbücher der MATLAB-Produktfamilie sind im Internet als PDF-Dateien erhältlich (`http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html`).

[2] *Cascaded-Integrator-Comb*

[3] *Interpolated-Finite-Impulse-Response*

Allein für die MATLAB-Grundsoftware sind z.B. 19 Handbücher verfügbar. Aufgrund ihres Umfangs von mehreren tausend Seiten ist eine systematische Durcharbeitung von der ersten bis zur letzten Seite dem Anwender kaum möglich. Vielmehr sollten sie als Nachschlagewerk am Rechner verwendet werden, um so die Aufrufbeispiele der Funktionen oder die Demonstrationen gleich ausführen zu können. Ähnliche Dokumente gibt es auch für die anderen Komponenten der MATLAB-Familie. Der Einfachheit halber werden diese Quellen nicht zitiert. Sie stehen über die MATLAB-Oberfläche zur Verfügung und können immer nachgeschlagen werden.

Die Firma MathWorks bietet Unterstützung zur Produktfamilie auch unter der Adresse `http://www.mathworks.com` (oder `http://www.mathworks.de`) an. So findet man z.B. unter der Adresse `http://www.mathworks.com/matlabcentral` im Menü *File Exchange* eine Vielzahl von Programm- und Modellbeispielen. Weiterhin werden laufend Webinare[4], auch in deutscher Sprache angeboten, die man herunterladen und beliebig oft ansehen kann. Auch auf weltweit über 700 Buchtitel, die die MATLAB-Software beschreiben und einsetzen, wird auf den Internet-Seiten von MathWorks verwiesen.

Danksagung

Die Autoren möchten sich vor allem bei den Studierenden des Studiengangs Nachrichtentechnik der Hochschule Karlsruhe – Technik und Wirtschaft bedanken, die bei den entsprechenden Vorlesungen mit ihren Fragen dazu beigetragen haben, dass die Theorie mit passenden Simulationen in MATLAB und Simulink begleitet wurde. So sind die meisten Simulationen dieses Buches entstanden. Viele davon basieren auch auf Themen, mit denen unsere Studierenden in Diplomarbeiten, die sie in der Industrie ausarbeiten, konfrontiert wurden.

Gleichfalls möchten wir uns bei unseren Kollegen Prof. Kessler, Prof. Scherf und Prof. Beucher bedanken, die ebenfalls MATLAB und Simulink in ihren Vorlesungen einsetzen und über Gespräche dazu beigetragen haben, einige Simulationen didaktischer zu gestalten.

Dank gebührt auch Frau Courtnay von der Firma The MathWorks USA, die als Betreuerin der Autoren von MATLAB-Büchern uns regelmäßig mit der Anfrage *What is the status of your book project?* angespornt hat und uns gleichzeitig mit neuen Versionen und Vorankündigungen der Software versorgt hat. Ebenfalls bedanken wir uns beim Support-Team von The MathWorks Deutschland in München und besonders bei Herrn Schäfer, der sehr professionell und zeitnah unsere Anfragen beantwortet hat.

Nicht zuletzt bedanken wir uns bei unseren Familien, die viel Verständnis während unserer Arbeit am Buch aufgebracht haben.

Josef Hoffmann (josef.hoffmann@hs-karlsruhe.de)
Franz Quint (franz.quint@hs-karlsruhe.de)

Juli 2007

[4] Multimediale Seminare über das Internet

Inhaltsverzeichnis

1 Entwurf und Analyse analoger Filter

Die Bestrebungen, die Signalverarbeitung so weit wie möglich digital zu realisieren, hat die analogen Filter ein wenig in den Hintergrund treten lassen. Da die Natur aber analog ist, bleiben die analogen Filter an der Schnittstelle zur digitalen Welt sicher erhalten. Der große Aufwand, mit dem die Chip-Hersteller die Eigenschaften der integrierten analogen Filter ständig verbessern, ist ein Beweis dafür, dass analoge Filter weiterhin ihren wohlverdienten Platz in der Signalverarbeitung bewahren werden. Darüberhinaus werden die Entwurfsmethoden für analoge Filter auch bei der Erstellung digitaler IIR-Filter[1] angewandt.

Der Entwurf und die Analyse analoger Filter ist umfangreich in der Literatur behandelt [68], [34], [56], [38]. Eine gute Einführung in die Thematik erhält man auch aus der Literatur zur Digitalen Signalverarbeitung, und zwar in den Kapiteln, die sich mit den Entwurfsmethoden von IIR-Filtern beschäftigen. Beispielhaft erwähnt seien hier die Bücher [26], [55], [52], die ebenfalls MATLAB-Begleitprogramme anbieten.

1.1 Entwurf und Analyse mit Funktionen der *Signal Processing Toolbox*

Lineare zeitinvariante Systeme können im Zeitbereich oder im Bildbereich (mit Hilfe der Laplace-Transformation) beschrieben werden [34]. Die Beschreibung im Zeitbereich erfolgt oft mit Hilfe eines Systems von Differentialgleichungen erster Ordnung, den sogenannten Zustandsgleichungen. Deswegen spricht man auch von einer Beschreibung im Zustandsraum. Filter sind Systeme mit einem Eingang und einem Ausgang (SISO-Systeme[2]) und ihre Beschreibung im Zustandsraum ist gemäß Gl. (1.1).

$$\begin{aligned} \frac{d\mathbf{x}_s(t)}{dt} &= \mathbf{A}\mathbf{x}_s(t) + \mathbf{B}u(t) \\ y_s(t) &= \mathbf{C}\mathbf{x}_s(t) + Du(t) \end{aligned} \tag{1.1}$$

Die erste Gleichung ist ein System von Differentialgleichungen in vektorieller Notation und stellt den Zusammenhang zwischen dem Zustandsvektor $\mathbf{x}_s(t)$ und der Eingangsgröße $u(t) = x(t)$ dar[3]. Die Länge n des Zustandsvektors $\mathbf{x}_s(t)$ gibt die Ordnung des Systems (Filters) an. Die zweite, algebraische Gleichung ist die sogenannte Ausgangsgleichung und stellt die Abhängigkeit der Ausgangsgröße $y_s(t) = y(t)$ vom Zustandsvektor und von der Eingangsgröße dar. Die Eigenschaften (Parameter) des Systems werden durch die Matrizen $\mathbf{A}, \mathbf{B}, \mathbf{C}$ und

[1] *Infinte-Impulse-Response* oder rekursive Filter

[2] *Single Input Single Output*

[3] In der Zustandsbeschreibung ist es üblich, die Eingangsgröße mit $u(t)$ zu bezeichnen. In der Welt der digitalen Signalverarbeitung wird die Eingangsgröße üblicherweise mit $x(t)$ bezeichnet und darf nicht mit dem Zustandsvektor $x_s(t)$ verwechselt werden.

$D = d$ festgelegt. Diese Matrizen müssen auch in MATLAB zur Beschreibung des Systems im Zustandsraum (Englisch: *state-space*) angegeben werden.

Im Bildbereich, mit s als Variablen der Laplace-Transformation, wird ein lineares zeitinvariantes System durch die Übertragungsfunktion $H(s)$ beschrieben [34]. Die Übertragungsfunktion (Englisch: *Transfer Function*) ist eine rationale Funktion mit jeweils einem Polynom im Zähler und im Nenner, gemäß Gl. (1.2). Dabei sind $Y(s)$ die Laplace-Transformierte des Ausgangssignals $y(t)$ und $X(s)$ die des Eingangssignals $x(t)$.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^n + b_1 s^{n-1} + b_2 s^{n-2} \cdots + b_n}{s^n + a_1 s^{n-1} + a_2 s^{n-2} \cdots + a_n} \tag{1.2}$$

In MATLAB kann diese Form der Übertragungsfunktion mit Hilfe der Koeffizienten des Zähler- und des Nennerpolynoms spezifiziert werden. Die Koeffizienten werden in zwei Vektoren angegeben, wobei der Koeffizient der höchsten Potenz eines Polynoms an erster Stelle steht. Diese Art der Beschreibung wird in MATLAB als *Transfer-Function*-Form bezeichnet.

Wenn die Polynome der Übertragungsfunktion mit Hilfe der Nullstellen des Zählers und Nenners ausgedrückt werden, erhält man die Null-Polstellen-Form [34]:

$$H(s) = \frac{Y(s)}{X(s)} = b_0 \frac{(s - z_1)(s - z_2) \ldots (s - z_n)}{(s - p_1)(s - p_2) \ldots (s - p_n)} \tag{1.3}$$

Die Werte $z_1, z_2, \ldots, z_n$ sind die Nullstellen der Übertragungsfunktion und die Nullstellen des Nennerpolynoms $p_1, p_2, \ldots, p_n$ sind ihre Pole. Der Faktor b_0 wird im Englischen *Gain* genannt. Die Form der Übertragungsfunktion nach Gl. (1.3) wird in MATLAB als *Zero-Pole-Gain*-Form bezeichnet und wird durch zwei Vektoren, die die Nullstellen und Pole enthalten, sowie durch einen Skalar für die Verstärkung[4] spezifiziert.

Für das System mit der Übertragungsfunktion nach Gl. (1.2) gibt es eine Beschreibung im Zustandsraum mit folgenden Matrizen:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \ldots & . & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & 1 \\ -a_n & -a_{n-1} & . & . & -a_2 & -a_1 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 1 \end{pmatrix} \tag{1.4}$$

$$\mathbf{C} = \begin{pmatrix} (b_n - b_0 a_n) & (b_{n-1} - b_0 a_{n-1}) & \ldots & (b_1 - b_0 a_1) \end{pmatrix} \qquad \mathbf{D} = d = b_0 \tag{1.5}$$

In MATLAB können die drei unterschiedlichen Beschreibungsformen mit den Funktionen **`ss2tf`**, **`tf2ss`**, **`ss2zp`**, **`zp2ss`**, **`tf2zp`** und schließlich **`zp2tf`** ineinander umgewandelt werden. Dabei steht das Kürzel **`ss`** für die Zustandsraum-Beschreibung, **`zp`** für die Null-

[4] Dabei handelt es sich nicht um eine Verstärkung des Eingangssignals beim Durchgang durch das System (diese ist prinzipiell frequenzabhängig), sondern lediglich um eine Bezeichnung für den Koeffizienten des höchstgradigen Gliedes, der zur Darstellung in der angegebenen Form als Faktor auszuklammern ist.

Polstellen-Beschreibung und **tf** für die Beschreibung in der *Transfer-Function*-Form. Die Beschreibung im Zustandsraum ist die stabilste gegen numerische Ungenauigkeiten, gefolgt von der Null-Polstellen-Form.

Die *Signal Processing Toolbox* enthält mehrere Gruppen von Funktionen zur Entwicklung von Analogfiltern. In Tabelle 1.1 sind die Funktionen zum Entwurf von Prototyp-Filtern im Tiefpassbereich gemäß unterschiedlicher Approximationsschemata (wie z.B. Bessel, Butterworth usw.) angegeben. Mit Hilfe der in Tabelle 1.2 angegebenen Transformationsfunktionen können die Prototypen dann in die gewünschten Bandpass-, Bandsperr-, Hochpass- oder Tiefpassfilter transformiert werden.

Tabelle 1.1: Analoge Tiefpassprototyp-Filter

besselap	Bessel Tiefpassprototyp-Filter
buttap	Butterworth Tiefpassprototyp-Filter
cheby1ap	Tschebyschev Typ I Tiefpassprototyp-Filter (mit Welligkeit im Durchlassbereich)
cheby2ap	Tschebyschev Typ II Tiefpassprototyp-Filter (mit Welligkeit im Sperrbereich)
ellipap	Elliptisches Tiefpassprototyp-Filter

Tabelle 1.2: Transformationen der Analogprototypen

lp2bp	Transformation Tiefpass- zu Bandpassfilter
lp2bs	Transformation Tiefpass- zu Bandsperrfilter
lp2hp	Transformation Tiefpass- zu Hochpassfilter
lp2lp	Transformation Tiefpass- zu Tiefpassfilter (mit anderer Durchlassfrequenz)

In einem Beispiel soll die Anwendung der Funktionen aus Tabelle 1.1 und 1.2 veranschaulicht werden. Es wird ein elliptisches Bandsperrfilter 8. Ordnung mit einem Sperrbereich von 1000 Hz bis 2000 Hz, mit einer Welligkeit im Durchlassbereich von 0.1 dB und mit einer Dämpfung von 60 dB im Sperrbereich ermittelt. Zur Spezifikation der Filterkenngrößen im Frequenzbereich wird auf Abschnitt 1.5.1 verwiesen. Da sich bei der Transformation vom Tiefpassbereich in den Bandpassbereich die Filterordnung verdoppelt, wird zunächst ein Tiefpassfilter der Ordnung vier entworfen. Mit

```
Ap = 0.1;       % 0,1 dB Welligkeit im Durchlassbereich
As = 60;        % 60 dB Dämpfung im Sperrbereich
nord = 4;       % Ordnung des Tiefpassprototyp-Filters (8/2)
[z,p,k] = ellipap(nord,Ap,As); % Null- Polstellen des Prototyps
```

wird die Übertragungsfunktion des Tiefpassprototyp-Filters in der *Zero-Pole-Gain*-Form ermittelt. Die Vektoren z und p enthalten die Null- und Polstellen der Übertragungsfunktion. Für das Tiefpassprototyp-Filter vierter Ordnung erhält man 4 Nullstellen und 4 Polstellen. Die Koeffizienten der Zähler- und Nennerpolynome der Übertragungsfunktion erhält man mit:

```
[b,a] = zp2tf(z,p,k);   % Koeffizienten der Übertragungsfunktion
```

In den Vektoren b und a sind die fünf Koeffizienten der Polynome des Zählers und des Nenners der Übertragungsfunktion gemäß Gl. (1.2) enthalten.

Den Frequenzgang des Prototypfilters im Frequenzbereich 10^{-1} Hz bis 10^{1} Hz mit 500 logarithmisch skalierten Punkten ermittelt man mit:

```
[Hap,wap]=freqs(b,a,2*pi*logspace(-1,1,500)); % Frequenzgang
% des Prototyps
fap = wap/(2*pi);
```

Zur Transformation des Prototypfilters in das gewünschte Bandsperrfilter verwendet man die Funktion **lp2bs**:

```
f1 = 1000;          f2 = 2000;
fm = sqrt(f1*f2);   B = f2 - f1;
% fm = (f1 + f2)/2;   B = f2 - f1;
% ------- Das Bandsperrfilter
[zaehler, nenner] = lp2bs(b,a,fm*2*pi,B*2*pi);
% ------- Frequenzgang des Bandsperrfilters
[Hbs,wbs] = freqs(zaehler, nenner, 2*pi*logspace(2,4,500));
fbs = wbs/(2*pi);
```

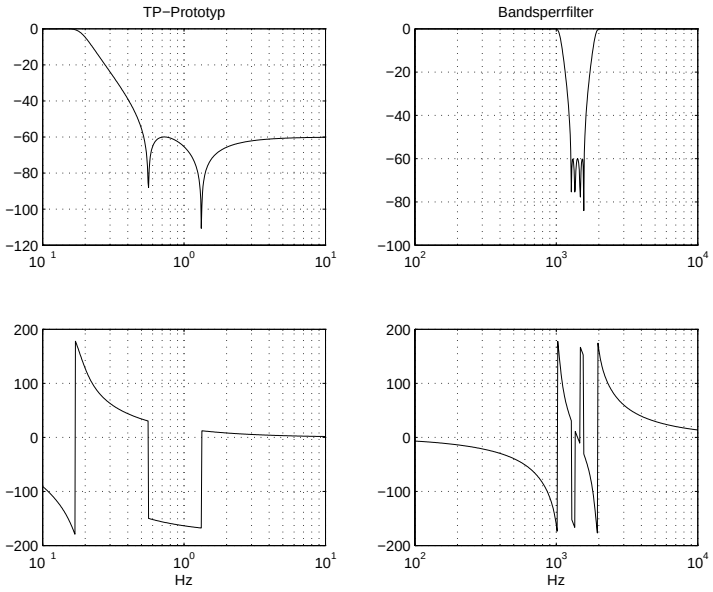

*Abb. 1.1: Frequenzgänge von Prototyptiefpass und Bandsperrfilter (*band_sperr1.m)

Die Ordnung des Bandsperrfilters verdoppelt sich wegen der Transformation aus der Tiefpass- in die Bandpasslage und ist somit acht. Damit hat die Übertragungsfunktion neun Koeffizienten im Zähler und neun Koeffizienten im Nenner. Für die Transformationen Tiefpass- in

Tiefpassfilter und Tiefpass- in Hochpassfilter bleibt die Ordnung erhalten . Die Frequenzgänge des Tiefpassprototyp- und Bandsperrfilters sind in Abb. 1.1 dargestellt und wurden mit der Funktion **freqs** ermittelt. Das Programm band_sperr1.m enthält alle Befehle zur Berechnung und Darstellung des Bandsperrfilters.

Tabelle 1.3: Direkte Entwicklung von Analogfiltern

besself	Bessel Analogfilter
butter	Butterworth Analogfilter
cheby1	Tschebyschev Typ I Analogfilter (mit Welligkeit in Durchlassbereich)
cheby2	Tschebyschev Typ II Analogfilter (mit Welligkeit im Sperrbereich)
ellip	Elliptisches Analogfilter

Die Mittenfrequenz fm des Bandsperrfilters kann, je nachdem ob man eine Symmetrie im linearen oder logarithmierten Frequenzgang wünscht, als das arithmetische oder das geometrische Mittel der beiden Eckfrequenzen berechnet werden.

MATLAB bietet auf einer höheren Abstraktionsebene auch Funktionen an, in denen der dargestellte Weg zum Filterentwurf bereits integriert ist. Diese Funktionen sind in Tabelle 1.3 angegeben und mit ihnen können auch digitale IIR-Filter entworfen werden. Das Filter aus dem vorherigen Beispiel erhält man unter Verwendung dieser Funktionen mit den folgenden Befehlen:

```
f1 = 1000;          f2 = 2000;    % Bandsperrbereich
Ap = 0.1;           As = 60;      % Welligkeit im Durchlass-
                                  % bzw. Sperrbereich
nor = 4;                          % Ordnung des TP-Prototyps
[zaehler, nenner] = ellip(nor, Ap, As, [f1, f2]*2*pi,...
    'stop','s');                  % Das Bandsperrfilter
```

Der Parameter 'stop' gibt an, dass der Frequenzbereich für ein Bandsperrfilter gedacht ist und der Parameter 's' gibt an, dass ein analoges Filter zu berechnen ist. Dieselbe Funktion dient (wie schon erwähnt) auch zur Ermittlung eines elliptischen digitalen IIR-Filters, wobei dann das Argument 's' entfällt.

1.2 Verzerrungen durch analoge Tiefpassfilter

Analoge Tiefpassfilter spielen auch in der digitalen Signalverarbeitung eine besondere Rolle, und zwar als Bandbegrenzungsfilter vor der Analog-Digital-Wandlung (Abb. 1.2). Entsprechend ihrer Aufgabe werden sie als *Antialiasing*-Filter bezeichnet [38]. Auch bei der Digital-Analog-Wandlung ist zur Rekonstruktion des zeitkontinuierlichen Signals ein analoges Tiefpassfilter erforderlich, welches in Abb. 1.2 als Glättungsfilter bezeichnet ist.

Wird ein Signal mit der Frequenz f_s abgetastet, so sollte das *Antialiasing*-Filter idealerweise die Bandbreite des zeitkontinuierlichen Signals auf $f_s/2$ scharf und ohne Verzerrungen begrenzen. Das Filter müsste einen konstanten Amplitudengang bis $f_s/2$ haben und danach unmittelbar in einen Sperrbereich mit großer Dämpfung übergehen. Der Phasengang sollte idea-

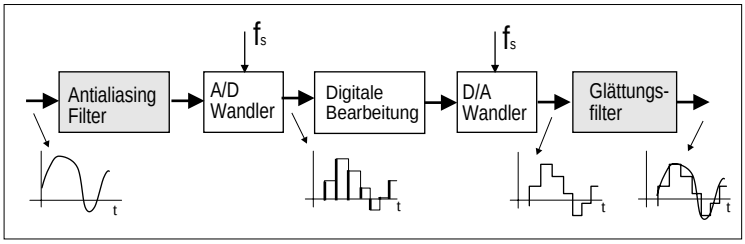

Abb. 1.2: Digitale Bearbeitung zeitkontinuierlicher Signale

lerweise Null oder linear fallend mit der Frequenz sein, dabei aber bei der Frequenz $f = 0$ den Wert $\varphi(0) = \pm 2k\pi, k = 0, 1, ...$ besitzen.

Dass ein in Abhängigkeit von der Frequenz linearer Phasenverlauf keine Verzerrungen hervorruft, kann man sich leicht folgendermaßen überlegen. Wenn am Eingang des Filters im stationären Zustand eine Komponente der Form

$$x(t) = \hat{x}\, cos(\omega t) \tag{1.6}$$

anliegt, dann ist der Ausgang durch

$$\begin{gathered} y(t) = \hat{x}\, A(\omega)\, cos(\omega t + \varphi(\omega)) = \hat{x}\, A(\omega)\, cos(\omega(t + \varphi(\omega)/\omega)) = \\ \hat{x}\, A(\omega)\, cos(\omega(t - \tau(\omega))) \end{gathered} \tag{1.7}$$

gegeben. Dabei sind $A(\omega)$ der Amplitudengang und $\varphi(\omega)$ der Phasengang des Filters. Damit jede Komponente der beliebigen Kreisfrequenz ω mit derselben Verzögerung am Ausgang ankommt, muss $\tau(\omega)$ eine Konstante τ sein. Das bedeutet:

$$\varphi(\omega)/\omega = -\tau(\omega) = -\tau \qquad \text{oder } \varphi(\omega) = -\tau\omega \tag{1.8}$$

Wegen der Periodizität der Sinus- oder Cosinusfunktionen ist auch eine Phase der Form

$$\varphi(\omega) = -\tau\omega - 2k\pi, \qquad k = 0, 1, 2, 3, ... \tag{1.9}$$

als ideal zu betrachten.

Der ideale Amplitudengang mit $A(\omega) = Konstante$ im Durchlassbereich kann annähernd bei analogen Filtern ohne allzu großen Aufwand realisiert werden. Dagegen ist eine annähernd lineare Phase nach Gl. (1.9) nicht so leicht zu realisieren.

Man definiert die Gruppenlaufzeit [9] als die negative Ableitung des Phasengangs nach der Kreisfrequenz

$$G_r(\omega) = -\frac{d\varphi(\omega)}{d\omega}\,, \tag{1.10}$$

wobei das negative Vorzeichen eingeführt wird, um positive Werte für die Gruppenlaufzeit zu erhalten. Keine Verzerrungen beim Durchgang durch das Filter erleiden Signale, in deren Frequenzbereich die Gruppenlaufzeit annähernd konstant ist.

Im Gegensatz zur Gruppenlaufzeit wird die Variable $\tau_p(\omega)$

$$\tau_p(\omega) = -\tau(\omega) = -\varphi(\omega)/\omega \tag{1.11}$$

als Phasenlaufzeit bezeichnet und stellt die Verzögerung einer Komponente der Frequenz ω im stationären Zustand dar.

Es ist berechtigt zu fragen, was geschieht, wenn der Phasengang linear aber nicht null bei $\omega = 0$ ist:

$$\varphi(\omega) = -G_r\,\omega - \theta_0 \tag{1.12}$$

Hier ist G_r die konstante Gruppenlaufzeit (die sich aus dem linearen Phasenverlauf ergibt) und θ_0 ist der Wert der Phase bei $\omega = 0$. Eine Summe von sinus- oder cosinusförmigen Komponenten

$$x(t) = \sum_{k=1}^{N} \hat{x}\; sin(\omega_k t) \tag{1.13}$$

im stationären Zustand wird durch das Filter mit konstantem (auf $A = 1$ normiertem) Amplitudengang und Phasengang nach Gl. (1.12) in

$$y(t) = \sum_{k=1}^{N} \hat{x}\; sin(\omega_k (t - G_r) - \theta_0) \tag{1.14}$$

umgewandelt. Weil $sin(\alpha - \beta) = sin(\alpha)cos(\beta) - sin(\beta)cos(\alpha)$ ist, kann der Ausgang $y(t)$ wie folgt geschrieben werden:

$$y(t) = cos(\theta_0) \sum_{k=1}^{N} \hat{x}\; sin(\omega_k (t - G_r)) - sin(\theta_0) \sum_{k=1}^{N} \hat{x}\; cos(\omega_k (t - G_r)) \tag{1.15}$$

Der zweite Term bildet eine Verzerrung wegen des *Offsets* θ_0. Nur wenn $\theta_0 = \pm 2k\pi$ ist, dann wird $sin(\theta_0) = 0$ und $cos(\theta_0) = 1$ und es entstehen keine Verzerrungen. Das sinusförmige Ausgangssignal ist eine mit G_r verzögerte Version des Eingangssignals.

Zu beachten ist, dass in vielen Anwendungen die Signale im stationären Zustand nicht sinus- oder cosinusförmig sind, im Sinne der Fourier-Analyse aber als eine (unendliche) Summe von Sinus- und Cosinusschwingungen dargestellt werden können. Insofern behalten die prinzipiellen Überlegungen dieses Abschnittes ihre Gültigkeit. Zur konkreten Auswirkung von nichtidealen Amplituden- und Phasengängen auf die Signalform empfiehlt sich jedoch auch eine Untersuchung an anderen Signalformen, wie sie nachfolgend an Beispielen gezeigt wird. Ein gutes Testsignal ist dabei bandbegrenztes Rauschen.

Im Programm `analog_tp1.m` werden die fünf möglichen analogen Tiefpassfiltertypen für gleiche Spezifikationen entworfen und ihre Frequenzgänge bzw. Gruppenlaufzeiten dargestellt (Abb. 1.3). Der Frequenzparameter in den Argumenten des Befehls zur Berechnung des Tschebyschev-Filters Typ II stellt die Frequenz dar, bei der man den Sperrbereich erreicht und nicht, wie bei den anderen Filtern, die Grenze des Durchlassbereichs.

Bei allen Filtern erhält man annähernd konstante Gruppenlaufzeiten für Frequenzen, die viel kleiner als die Durchlassfrequenz sind. In der Nähe der Durchlassfrequenz (1000 Hz) sind die Verläufe unterschiedlich.

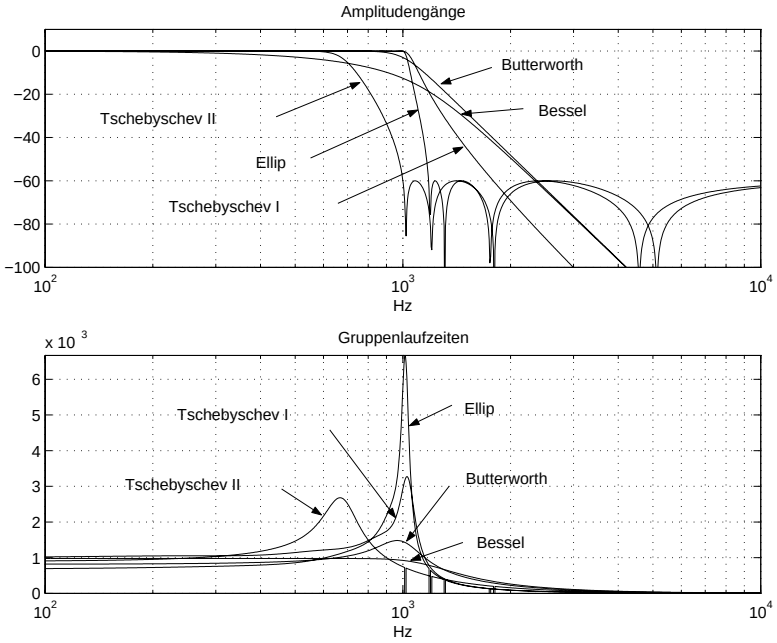

*Abb. 1.3: Amplitudengänge und Gruppenlaufzeiten der fünf Tiefpassfilter (*analog_tp1.m)

Früh ändert sich die Gruppenlaufzeit des Tschebyschev-Filters Typ II, weil bei diesem die Frequenz von 1000 Hz nicht die Durchlassfrequenz sondern die Sperrfrequenz ist. Die Durchlassfrequenz ist kleiner (ca. 500 Hz).

Sehr stark ändert sich die Gruppenlaufzeit bei dem Elliptischen und dem Tschebyschev-Filter vom Typ I. Für den Maßstab des Bildes sieht man kaum eine Änderung beim Bessel-Filter. Dieses Filter hat aber einen sehr flachen Verlauf des Amplitudengangs beim Übergang vom Durchlassbereich in den Sperrbereich und ergibt somit in diesem Bereich Amplitudenverzerrungen. Einen guten Kompromiss stellt das Butterworth-Filter dar.

Kurze Erläuterungen einiger Ausschnitte des Programms `analog_tp1.m` sollen zum besseren Verständnis führen und zum Experimentieren ermutigen. Die Koeffizienten der Zähler und Nenner der Übertragungsfunktionen werden in den Matrizen `b` und `a` gespeichert und einzeln berechnet, wie z.B. für das Tschebyschev-Filter Typ I:

```
Ap = 0.1;       % Welligkeiten im Durchlassbereich
[b(3,:),a(3,:)] = cheby1(nord, Ap, w_3dB, 's');
```

Anschließend wird der Frequenzbereich festgelegt und mit der Funktion **`freqs`** werden in der **`for`**-Schleife die Frequenzgänge der fünf Filter berechnet und in den Zeilen der Matrix `H` gespeichert:

```
alpha_min = log10(f_3dB/10);
alpha_max = log10(f_3dB*10);
```

```
fmin = 10^(floor(alpha_min));
fmax = 10^(ceil(alpha_max));
f = logspace(log10(fmin), log10(fmax), 1000);
w = 2*pi*f;

H = zeros(5,length(w));    % Frequenzgang
for k = 1:5
    H(k,:) = freqs(b(k,:), a(k,:), w);
end;
```

Da für analoge Filter in der *Signal Processing Toolbox* keine Funktion zur Berechnung der Gruppenlaufzeiten vorliegt, werden die Gruppenlaufzeiten der Filter mit Annäherungen über Finite-Differenzen berechnet:

```
Gr = -diff(unwrap(angle(H.'))./[diff(w')*ones(1,5)];
```

Die Übertragungsfunktionen des Elliptischen und des Tschebyschev-Filters Typ II besitzen Nullstellen auf der imaginären Achse der komplexen Variablen $s = \sigma + j\omega$ und dadurch sind im Amplitudengang Unstetigkeiten vorhanden. An diesen Stellen erhält man sehr kleine Werte für die Gruppenlaufzeiten, die in den Darstellungen durch die Wahl der Achsen (im Befehl **axis**) abgeschnitten werden:

```
subplot(212), semilogx(f(1:end-1), Gr);
La = axis;          axis([La(1:2), 0, max(max(Gr))]);
title('Gruppenlaufzeiten');
xlabel('Hz');       grid;
```

Die Zeilen der Matrix `Gr` enthalten die Gruppenlaufzeiten der fünf Filter und haben wegen der Differenzbildung eine um eins kleinere Länge als die Länge des Vektors der Frequenzen `f`, daher die Form des Befehls **semilogx**`(f(1:end-1), Gr);`.

Experiment 1.1: Verzerrungen wegen des Phasengangs

Zur Untersuchung der vorgestellten Filter wird das Modell `analog_tp_1.mdl` aus Abb. 1.4 benutzt. Die Filter werden in dem zuvor besprochenen Programm `analog_tp1.m` berechnet. Die Filterkoeffizienten sind damit auch im Simulink-Modell bekannt.

Es stehen verschiedene Quellen zur Verfügung, die man mit den *Gain*-Blöcken dem Summierer zuschalten kann. Der *Multiport Switch*-Block schaltet den Ausgang des gewählten Filters zum *Mux*-Block, an dem auch das über den *Transport Delay*-Block verzögerte Eingangssignal der Filter anliegt. So kann man Eingangs- und Ausgangssignal des Filters zeitrichtig überlagert mit dem Block *Scope* betrachten. Für jedes Filter muss diese Verzögerung durch Experimentieren neu eingestellt werden. Als Richtwert kann dessen Gruppenlaufzeit, die man aus Abb. 1.3 entnehmen kann (ca. 1 ms), dienen.

Zuerst sollen die Verzerrungen der Amplituden sinusförmiger Komponenten untersucht werden. Dazu stellt man die Frequenz eines Generators auf einen Wert im Durchlassbereich des entsprechenden Filters ein und schaltet nur diesen Generator auf den Eingang des Summierers. Vergrößert man die Frequenz zu Werten, die immer näher an der Durchlassfrequenz von 1000 Hz liegen, wird man die Frequenzgrenze ermitteln können, die zu Amplitudenverzerrungen führt.

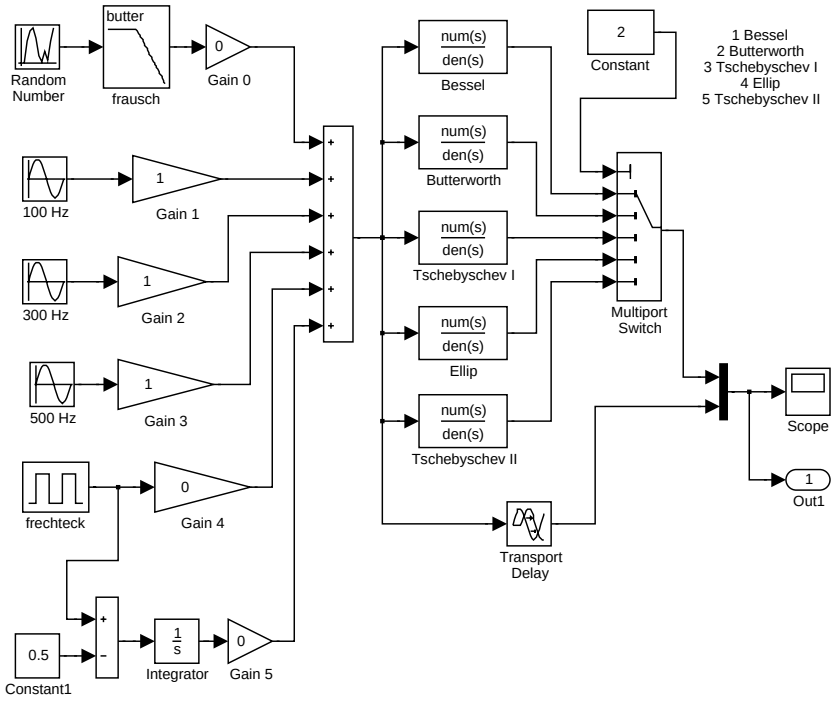

*Abb. 1.4: Simulink-Modell zum Testen der Filter (*analog_tp_1.mdl, analog_tp1.m)

Die Verzerrungen wegen eines nichtlinearen Phasenverlaufs können an einem einzelnen Sinussignal naturgemäß nicht beobachtet werden. Wir wählen zur Beobachtung der Verzerrungen ein Signal, das sich aus der Überlagerung von drei Sinussignalen ergibt. Die drei Generatoren werden mit Frequenzen von 100 Hz, 300 Hz und 500 Hz und Amplituden von 1, 1/3 bzw. 1/5 parametriert. Sie entsprechen somit den ersten drei Komponenten der Fourier-Reihe eines rechteckförmigen Signals mit dem Tastverhältnis 1:1 und ohne Mittelwert. Das Überlagerungssignal ist also eine symmetrische Annäherung des rechteckförmigen Signals und die Abweichung eines Filterausgangssignals von diesem symmetrischen Verlauf ist ein Maß für die Verzerrungen.

Es werden die Auswirkungen von vier analogen Filtern (ohne das Tschebyschev-Filter Typ II) mit Hilfe des Programms `analog_tp11.m` und des Modells `analog_tp_11.mdl` untersucht. Dabei wird die Simulation mit dem Modell für jeden Filtertyp einmal ausgeführt. Um die Programme einfach und verständlich zu halten, sind die Blöcke des Modells interaktiv zu initialisieren.

Im Programm `analog_tp11.m` wird zuerst das Programm `analog_tp1.m` aufgerufen um die Filter zu entwerfen. Die Bezeichnungen der Filter sind in einem *Cell*-Feld `text_` abgelegt, um sie bei den Darstellungen im Titel einzusetzen:

```
text_{1} = 'Bessel';
text_{2} = 'Butterworth';
text_{3} = 'Tschebyschev I';
text_{4} = 'Ellip';
```

Zu beachten ist, dass **text** eine Funktion in MATLAB ist, mit deren Hilfe Texte in Bilder geschrieben werden können, daher die Bezeichnung `text_` mit Unterstrich zur Unterscheidung. Für jedes Filter werden die korrekten Verzögerungen für den *Transport Delay*-Block im Vektor `verz` initialisiert, um die bestmögliche Überlagerung des Eingangs- und Ausgangssignals zu erzielen. In der anschließenden **for**-Schleife wird das Simulink-Modell für jedes Filter aufgerufen und die Ergebnisse in verschiedenen **subplot**-Fenstern dargestellt (Abb. 1.5):

```
figure(3);      clf;
for Typ = 1:4
    delay = verz(Typ);              % Festlegung der Verzögerung
    sim('analog_tp_11', [0, 0.025]);    % Aufruf der Simulation
    subplot(2,2,Typ), plot(tout, yout)
    title(text_{Typ});  grid
    La = axis;      axis([min(tout), max(tout), La(3:4)]);
    xlabel('Zeit in s')
end;
```

Die Unterschiede zwischen Eingangs- und Ausgangssignal werden in den vergrößerten Signalausschnitten aus Abb. 1.6 deutlich. Diese Darstellungen werden ebenfalls in einer **for**-Schleife erzeugt. Mit dem Befehl **axis** wird der darzustellende Ausschnitt gewählt:

```
nt = length(tout);
nd = fix(nt*0.43):fix(nt*0.65);    % Eingeschwungener Bereich
figure(4);      clf;
for Typ = 1:4
    delay = verz(Typ);
    sim('analog_tp_11', [0, 0.025]);
    subplot(2,2,Typ), plot(tout(nd), yout(nd,:))
    title(text_{Typ});  grid
    La = axis;      axis([min(tout(nd)), max(tout(nd)), ...
            0.65, max(max(yout(nd,:)))]);
    xlabel('Zeit in s')
end;
```

Die Variablen `yout`, `tout` sind die Signale, die nach der Simulation der MATLAB-Umgebung übertragen werden, gemäß der Initialisierung der Parameter *Data Import/Export* und der Option *Save to workspace* aus dem Fenster *Configuration Parameters* (Abb. 1.7). Dieses Fenster wird über das Menü *Simulation/Configuration Parameters* des Modells geöffnet.

Der *Outport*-Block *Out1* des Modells zeigt, welche Signale als *Output*-Signale mit `yout` bezeichnet sind. Über den Bereich *Save options* des gleichen Fensters (Abb. 1.7) wird `yout` als Feld (*Array*) mit zwei Spalten, welches die letzten 2000 Abtastwerte des Signals am Eingang des *Mux*-Blocks enthält, parametriert.

Die Ergebnisse aus Abb. 1.6 zeigen, dass das Besselfilter die drei sinusförmigen Komponenten in korrekter zeitlicher Relation zusammensetzt und somit keine Verzerrungen wegen der Phase hinzufügt. Der Unterschied zwischen Eingangs- und Ausgangssignal besteht wegen des frühen Abfalls des Amplitudengangs (siehe Abb. 1.3), der zu einer Dämpfung der Sinusschwingung mit der Frequenz $f = 500$ Hz führt. Bei den anderen Filtern sind die Verzerrungen hauptsächlich wegen der Abweichung des Phasengangs von einem linearen Verlauf gegeben. Sehr stark ist das beim Elliptischen Filter zu beobachten. Insgesamt den besten Kompromiss erreicht man mit dem Butterworth-Filter.

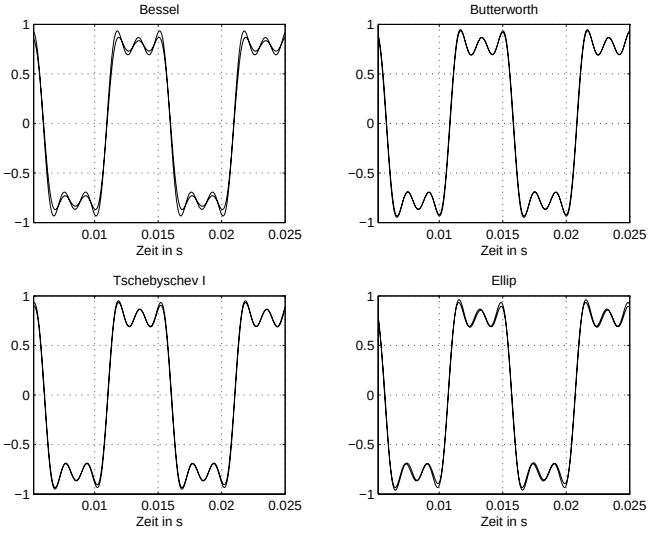

*Abb. 1.5: Wiedergabe der drei Komponenten mit 100 Hz, 300 Hz, 500 Hz und Amplituden 1, 1/3 bzw. 1/5 (*analog_tp11.m, analog_tp_11.mdl)

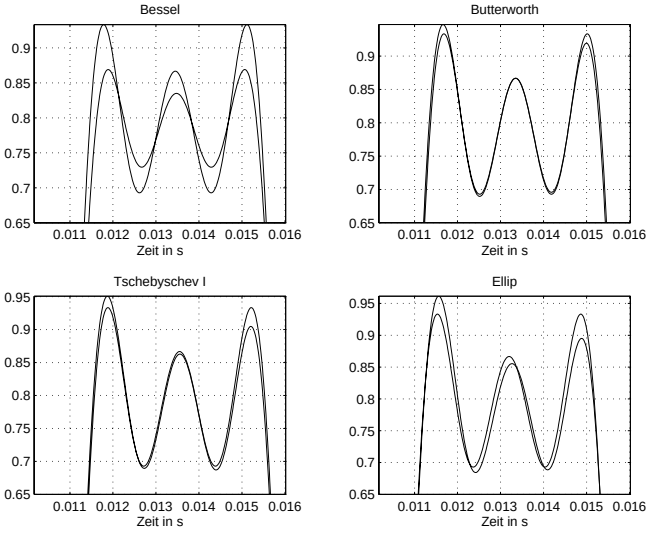

Abb. 1.6: Ausschnitte aus Abb. 1.5

In einem weiteren Versuch sollen die Filter mit einem bandbegrenzten Rauschsignal angeregt werden. Dafür werden, der Einfachheit halber, das vorherige Programm und das vorherige Modell unter den Namen `analog_tp12.m` beziehungsweise `analog_tp_12.mdl` gespeichert und entsprechend geändert. Im Modell werden alle Quellen mit Ausnahme der Rauschquelle (links oben in Abb. 1.4) durch die *Gain*-Blöcke gesperrt.

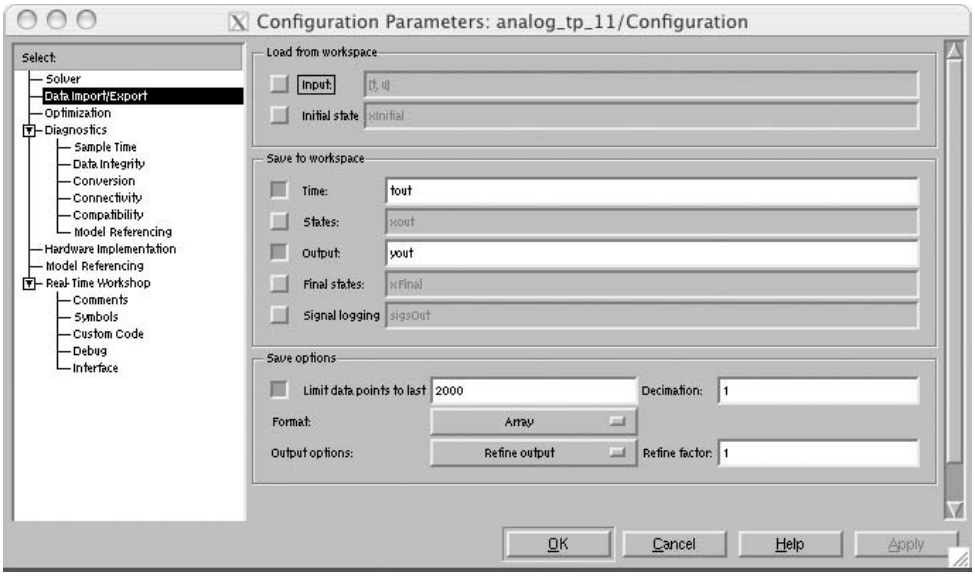

Abb. 1.7: 'Configuration Parameters'-Fenster aus dem Modell-Menü 'Simulation'

Zuerst wird die Bandbreite der Rauschquelle mit dem Tiefpassfilter, das an der Quelle angeschlossen ist, auf 500 Hz begrenzt. Wie aus der Darstellung der Frequenzgänge aus Abb. 1.3 zu erwarten war, sind keine großen Verzerrungen zu beobachten, da B=500 Hz noch in dem Bereich liegt, in dem die Amplitudengänge und Gruppenlaufzeiten konstant sind. Wenn man die Bandbreite des Rauschsignals auf 1000 Hz anhebt, dann ergeben sich erhebliche Fehler.

Experiment 1.2: Verzerrung von rechteckförmigen Pulsen

Mit diesem Experiment soll der Einfluss der Tiefpassfilter auf rechteckförmige Pulse unterschiedlicher Dauer untersucht werden. Dafür werden das zuvor verwendete Programm und Modell unter den Dateinamen `analog_tp13.m` beziehungsweise `analog_tp_13.mdl` gespeichert und dem Experiment entsprechend angepasst. Das Rechtecksignal wird durch eine von null verschiedene Verstärkung im Block *Gain 4* dem Filter zugeführt, während die anderen Signalquellen gesperrt werden. Der Generator wird mit der gewünschten Periode und Pulsdauer über das MATLAB-Programm initialisiert.

Der *Transport Delay*-Block aus dem ursprünglichen Modell wurde hier entfernt, so dass man die Verzögerungen vom Eingang zum Ausgang des Filters verfolgen kann. Die Simulation wird mit der Schrittweite `dt` ausgeführt.

```
period = 20e-3;     % Periode der rechteckigen Pulse
tau = 10;           % Dauer der Pulse in % der Periode
tau_r = period*tau/100; % Dauer der Pulse in s
dt = 1e-5;       % Schrittweite der Simulation
```

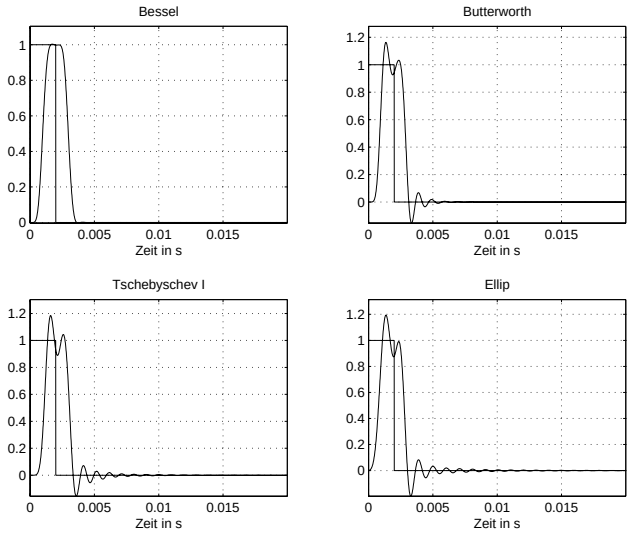

*Abb. 1.8: Antwort der Filter auf rechteckige Pulse (*analog_tp13.m, analog_tp_13.mdl)

```
figure(3);      clf;
for Typ = 1:4
    sim('analog_tp_13',[0:dt:2*period-dt]); % Aufruf
                                            % der Simulation
    subplot(2,2,Typ), plot(tout, yout)
    title(text_{Typ});  grid
    La = axis;      axis([min(tout), max(tout), La(3:4)]);
    xlabel('Zeit in s')
end;
```

Eine ähnliche Programmsequenz speichert einen Signalausschnitt bestehend aus einer Periode in den Variablen `yout` bzw. `tout` und stellt sie dar. Abb. 1.8 zeigt diesen Ausschnitt für ein Signal der Periode 20 ms und der Pulsdauer 2 ms.

In Abb. 1.9 ist im oberen Teil das Spektrum des rechteckigen Eingangspulses dargestellt, während im unteren Teil das Spektrum des Ausgangssignals für ein Butterworth-Filter mit einer Grenzfrequenz $f_g = 1000$ Hz dargestellt ist. Es ist verständlich, dass die Verzerrung des Ausgangssignals um so stärker sein wird, je geringer die Bandbreite des Filters im Vergleich zur Bandbreite des Signals ist. Wählt man die Pulsdauer des rechteckförmigen Signals z.B. als 1 ms, so wird das Spektrum des Pulses seine erste Nullstelle bei $f = 1000$ Hz haben und das gewählte Filter wird alle Komponenten (im Frequenzbereich) außer der Hauptkeule aus dem Ausgangssignal entfernen. Um diese Werte einzustellen, sind in dem Programm die Variablen `period` und `tau` mit den Werten `20e-3` (20 ms) bzw. `5` (5 % von 20 ms) zu initialisieren.

Der Programmabschnitt mit welchem die Spektren ermittelt werden, beginnt mit der Wahl des Filters. Danach wird die Simulation aufgerufen und die in der Variablen `yout` gespeicherten Abtastwerte werden mittels FFT Fourier-transformiert.

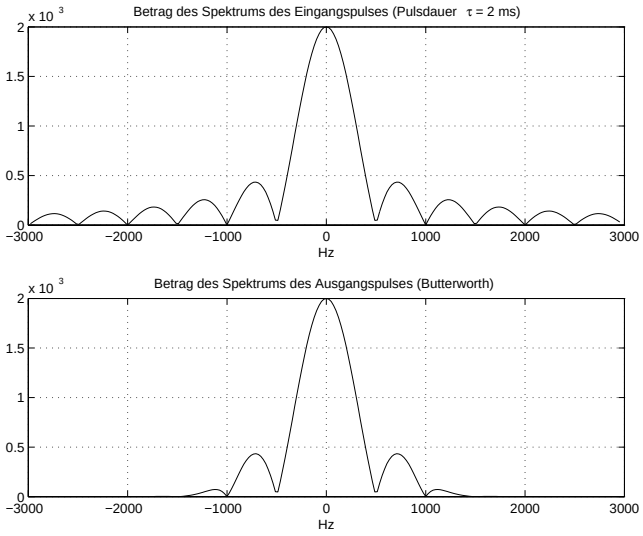

*Abb. 1.9: Betrag des Spektrums des Eingangs- und Ausgangspulses (*analog_tp13.m, analog_tp_13.mdl)

```
Typ = 2;            % Typ des Filters für die Spektraldichte
sim('analog_tp_13',[0:dt:period-dt]);
[n,m] = size(yout);
nfft = max(2^nextpow2(n), 4096);
H = dt*abs(fft(yout,nfft));  % Annäherung des Betrags der
% Fourier-Transformation
H = fftshift(H);
```

Bei der benutzen Simulationsschrittweite `dt = 1e-5`, dehnt sich der Frequenzbereich der FFT von 0 bis 100 kHz oder von -50 kHz bis 50 kHz. Der Frequenzbereich der hier interessiert, ist viel kleiner, so dass nur ein Ausschnitt notwendig ist. Die Programmsequenz, mit der ein Ausschnitt von -3000 Hz bis 3000 Hz der FFT selektiert und dargestellt wird, ist:

```
n_3000 = 3000*nfft*dt;     % Index für einen
          % Frequenzbereich von -3000 Hz bis 3000 Hz
ndfft = -fix(n_3000):fix(n_3000)-1;
fd = ndfft/(nfft*dt);

figure(5);      clf;
subplot(211), plot(fd,...
H(nfft/2-fix(n_3000)+1:nfft/2+fix(n_3000),1));
title(['Fourier-Transformation des Eingangspulses',...
       ' (Pulsdauer  \tau = ',num2str(tau_r*1000),' ms)']);
xlabel('Hz');   grid;
```

```
subplot(212), plot(fd, ...
H(nfft/2-fix(n_3000)+1:nfft/2+fix(n_3000),2));
title(['Fourier-Transformation des Ausgangspulses (',...
          text_{Typ},')']);
xlabel('Hz');   grid;
```

In dem Programm `analog_tp2.m` und dem Modell `analog_tp2.mdl` werden weitere Möglichkeiten zum Experimentieren gezeigt. Alle Parameter des Modells werden über Variablen aus dem Programm initialisiert. Solche Programme sollte man realisieren, nachdem Versuche mit einfachen Programmen und Modellen, die interaktiv parametriert werden, erfolgreich waren.

Studenten stellen oft die Frage, weshalb die Antwort der Filter auf rechteckförmige, periodische Signale nicht so aussieht, wie in der Zusammensetzung der periodischen Harmonischen aus Abb. 1.5. Die Filter unterdrücken doch die Harmonischen ab einer bestimmten Ordnung und somit müssten die verbliebenen Harmonischen die gezeigte Form ergeben. Dieses ist in der Tat der Fall bei digitalen FIR-Filtern[5], die einen linearen Phasengang haben, falls die Filterkoeffizienten, und damit auch die Impulsantwort, symmetrisch sind. Im Fall der zeitkontinuierlichen analogen Filter lässt sich eine symmetrische Impulsantwort jedoch nur annähernd realisieren.

In Abb. 1.10 sind die Impulsantworten derselben vier Filter dargestellt, die im Programm `analog_tp1.m` berechnet und untersucht wurden. Sie wurden mit dem Programm `einh_puls1.m` bzw. Simulink-Modell `einh_puls_1.mdl` erzeugt. Ein Puls der Amplitude eins und der Dauer `dt` wird als Eingangssignal den Filtern zugeführt. Die Antwort wird dann auf die Fläche des Pulses normiert, um die Impulsantwort anzunähern:

```
% ------- Impulsantwort
k1 = 1;     k2 = 0;
dt = 1e-5;      % Dauer des Pulses
d_t = 1e-6;     % Maximale Schrittweite der Simulation;
my_options = simset('MaxStep', d_t);

figure(3);      clf;
for Typ = 1:4
    sim('einh_puls_1', [0:dt:0.01], my_options);
    subplot(2,2,Typ),plot(tout, yout/dt)
                  % Darstellung mit Normierung yout/dt
    title(text_{Typ});  grid
    La = axis;      axis([min(tout), max(tout), La(3:4)]);
    xlabel('Zeit in s')
end;
```

Wie Abb. 1.10 zeigt, ist nur die Impulsantwort des Bessel-Tiefpassfilters annähernd symmetrisch und für dieses Filter ist die Antwort auf einen rechteckförmigen Puls (Abb. 1.8 links oben) für die steigende Flanke gleich der Antwort für die fallende Flanke.

[5] *Finite Impulse Response* oder nichtrekursive Filter

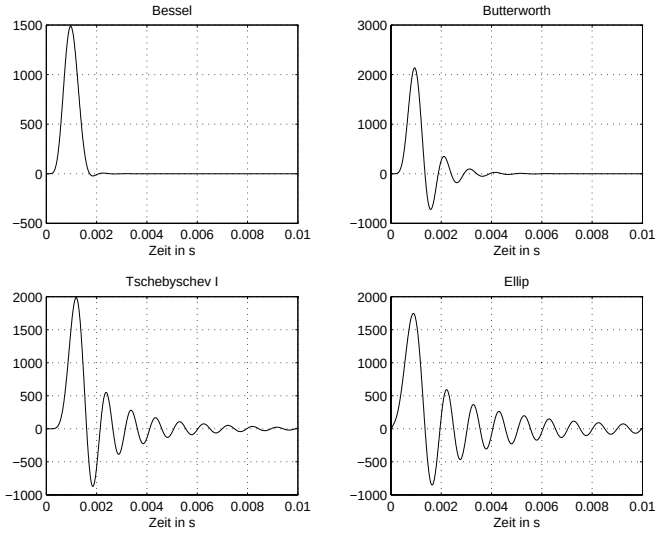

*Abb. 1.10: Impulsantworten der Filter (*einh_puls1.m, einh_puls_1.mdl)

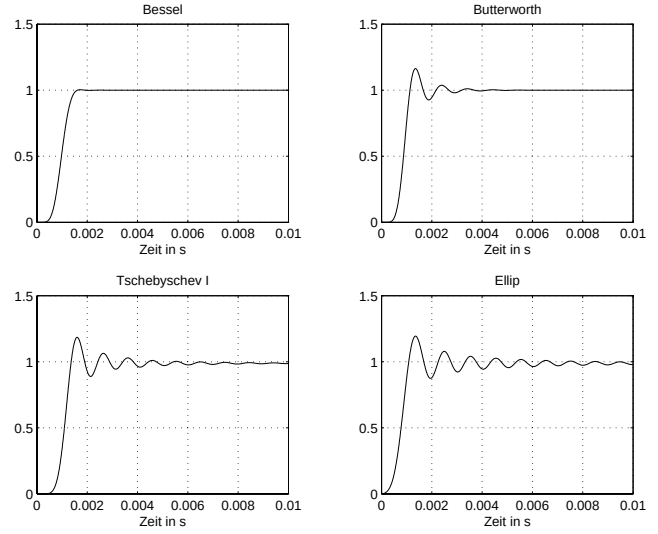

*Abb. 1.11: Sprungantworten der Filter (*einh_puls1.m, einh_puls_1.mdl)

Im Programm werden auch die Sprungantworten durch Integration der Impulsantworten berechnet, um die geschätzten Impulsantworten zu überprüfen. Die Integration wird mit der Funktion **cumsum** numerisch berechnet:

```
figure(4);      clf;
for Typ = 1:4
    sim('einh_puls_1', [0:dt:0.01]);
    subplot(2,2,Typ),plot(tout, cumsum(yout/dt)*dt)
                    % Annäherung des Integrals
    title(text_{Typ});  grid
    La = axis;      axis([min(tout), max(tout), La(3:4)]);
    xlabel('Zeit in s')
end;
```

Experimentell kann man die Sprungantwort mit dem besprochenen Simulink-Modell ermitteln, indem Sprünge auf den Eingang der Filter geschaltet werden (Abb. 1.11). Diese kann man mit den Sprungantworten, die über die Integration mit **cumsum** erhalten wurden, vergleichen.

1.3 Verzerrungen durch analoge Hochpassfilter

Man kann Hochpassfilter aus Tiefpassfiltern durch eine mathematische Transformation der Frequenzvariablen [9] erhalten. Wenn gewünscht wird, dass die Transformation die in Abb. 1.12 dargestellte Symmetrie im logarithmischen Frequenzgang haben soll, dann sind folgende Beziehungen zu erfüllen:

$$log(\omega_0) - log(\omega_{TP}) = log(\omega_{HP}) - log(\omega_0) \tag{1.16}$$

oder

$$\omega_{TP} = \frac{{\omega_0}^2}{\omega_{HP}}. \tag{1.17}$$

Wenn die komplexen Variablen $j\omega$ betrachtet werden, dann ist die Transformation durch

$$j\omega_{TP} = \frac{{\omega_0}^2}{j\omega_{HP}} \tag{1.18}$$

gegeben.
Exemplarisch wird diese Transformation für ein Tiefpassfilter zweiter Ordnung

$$H(j\omega_{TP}) = \frac{1}{(j\omega_{TP})^2/{\omega_0}^2 + (j\omega_{TP})2\zeta/\omega_0 + 1} \tag{1.19}$$

angewandt. Die Parameter des Filters, welche die Bandbreite und den Typ (Bessel, Butterworth, usw.) bestimmen, sind ω_0 als charakteristische Frequenz und 2ζ als Dämpfungsfaktor. Durch Einsetzen der Frequenztransformation erhält man die Übertragungsfunktion des Hochpassfilters:

$$H(j\omega_{HP}) = H(j\omega_{TP})\big|_{j\omega_{TP} = {\omega_0}^2/(j\omega_{HP})} \tag{1.20}$$

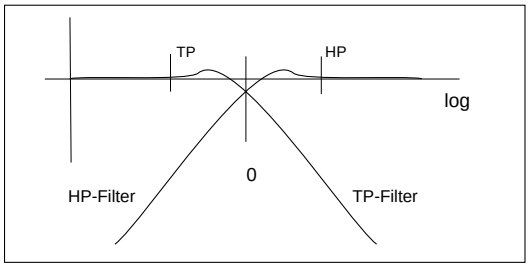

Abb. 1.12: Transformation der TP-Filter in HP-Filter

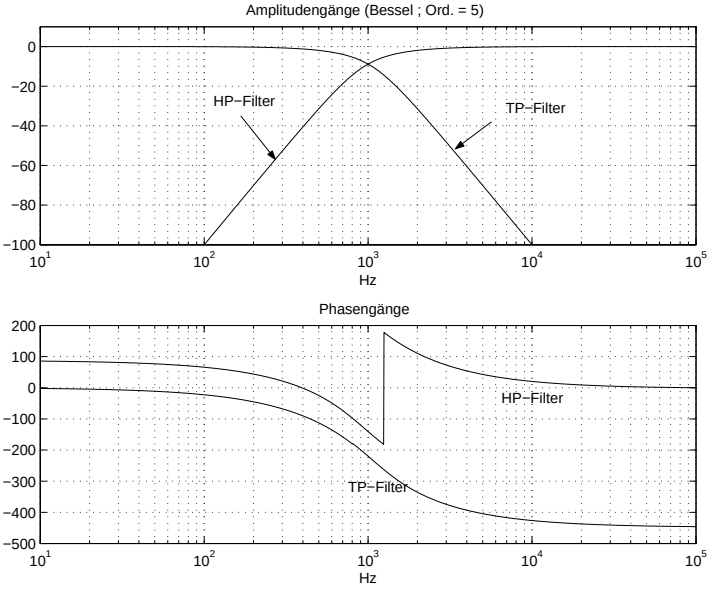

*Abb. 1.13: Amplitudengang und Phasengang des TP- und HP-Filters (*tp2hp1.m, tp2hp_1.mdl)

oder

$$H(j\omega_{HP}) = \frac{(j\omega_{HP})^2/{\omega_0}^2}{(j\omega_{HP})^2/{\omega_0}^2 + (j\omega_{HP})2\zeta/\omega_0 + 1} \tag{1.21}$$

Diese Transformation wird in den MATLAB-Funktionen verwendet, um die Tiefpassfilter in Hochpassfilter umzuwandeln. Da sich der Charakter des Filters ändert, werden sich durch die Transformation die Einschwingeigenschaften der ursprünglichen Tiefpassfilter ändern. Dieser Sachverhalt wird mit dem Programm `tp2hp1.m` veranschaulicht, in dem der Frequenzgang und die Sprungantwort eines Tiefpassfilters und des entsprechenden Hochpassfilters ermittelt und dargestellt werden.

In Abb. 1.13 sind die Frequenzgänge des gewählten Filtertyps dargestellt und die erwartete Symmetrie der Amplitudengänge in der logarithmischen Darstellung ist ersichtlich.

Die Sprungantworten werden mit Hilfe des Simulink-Modells `tp2hp_1.mdl`, das aus dem MATLAB-Programm aufgerufen wird, ermittelt. Sie zeigen, dass die Einschwingvorgänge sich stark unterscheiden. Am besten zu sehen ist dies beim Bessel-Filter. Für das gewählte Bessel-Filter 5. Ordnung besitzt das Tiefpassfilter praktisch kein Überschwingen, während beim entsprechenden Hochpassfilter es zu ca. 30 % Überschwingung kommt (welche sich bei einem Hochpassfilter als eine „Unterschwingung“ des stationären Endwertes null darstellt).

Die ideale Phase der Hochpassfilter ist gleich null, ein Wert der sich bei höheren Frequenzen im Durchlassbereich einstellt (Abb. 1.13). Die Verzerrungen durch Hochpassfilter, die mit der Transformation nach Gl. (1.18) erhalten werden, können ähnlich wie bei den Tiefpassfiltern untersucht werden. Die dort gezeigten Programme sind leicht für diese Filter anzupassen.

Im Programm `tp2hp3.m` werden die üblichen fünf Typen von Hochpassfiltern für gegebene Parameter (Durchlassfrequenz 1000 Hz, Ordnung 8, usw.) ermittelt und ihre Frequenzgänge dargestellt (Abb. 1.14).

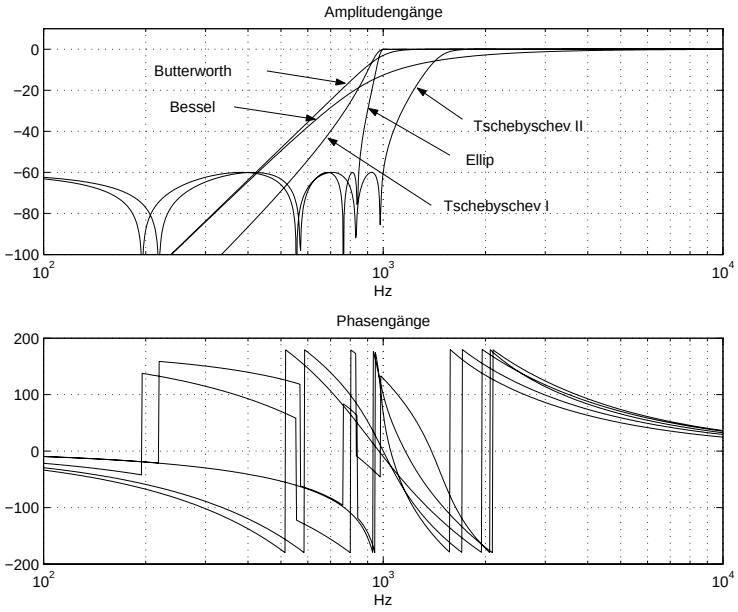

*Abb. 1.14: Frequenzgänge der typischen Hochpassfilter (*tp2hp3.m)

Der folgende Programmabschnitt berechnet beispielhaft das Tschebyschev-Filter Typ I:

```
....
% ------- Entwicklung eines Tschebyschev-Filters Typ I
Ap = 0.1;           % Welligkeiten in Durchlassbereich
[b(3,:),a(3,:)] = cheby1(nord, Ap, w_3dB,'high','s');
....
```

Die Zeichenkette `'high'` gibt an, dass ein Hochpassfilter zu berechnen ist und `'s'` zeigt, dass ein analoges Filter gewünscht wird. Wie bereits erwähnt, dient dieselbe Funktion auch zur Entwicklung von digitalen IIR-Filtern ausgehend von den entsprechenden analogen Filtern.

Aus der Darstellung in Abb. 1.14 ist zu entnehmen, dass der ideale Amplitudengang von konstant 0 dB im Durchlassbereich relativ leicht zu erhalten ist. Nur das Bessel-Filter hat einen flachen Übergang vom Sperrbereich in den Durchlassbereich und das Tschebyschev-Filter Typ II, mit Welligkeit im Sperrbereich, hat für die charakteristische Frequenz eine andere Definition. Sie stellt die Grenze des Sperrbereichs dar und dadurch ist die Durchlassfrequenz größer als 1000 Hz.

Der Phasengang beginnt bei allen Filtern bei einem positiven Wert gleich der Ordnung mal $\frac{\pi}{2}$ (in diesem Fall bei $8 \times \frac{\pi}{2} = 4\pi$). Wegen der Vieldeutigkeit der Phase in 2π beginnt die Darstellung bei dem Phasenwert 0 (das ist 4π mod 2π). Es ist zu beachten, dass die Ordinate des Phasengangs in Abb. 1.14 nicht in Radian, sondern in Grad skaliert ist.

Bei den Hochpassfiltern ist der ideale Phasengang gleich null, ein Wert der sich im Durchlassbereich leider erst weit oberhalb der Durchlassfrequenz einstellt . Es gibt hier keine Annäherung eines Phasenverlaufs, die keine Verzerrungen hervorruft. Man erinnere sich, dass bei den Tiefpassfiltern ein linearer Phasenverlauf mit dem Wert null (oder einem Vielfachen von 2π) bei $\omega = 0$ ideal ist. Bei den Hochpassfiltern sind somit größere Verzerrungen wegen ihres Phasengangs zu erwarten.

Das zuvor verwendete Programm (`tp2hp3.m`) wird leicht abgewandelt in eine Funktion (`tp2hp31.m`), die als Argument die Durchlassfrequenz für die zu entwerfenden Hochpassfilter besitzt. Die Funktion liefert als Ergebnis die Koeffizienten der Zähler und Nenner der Filter.

Um die Verzerrungen wegen des Phasenverlaufs zu untersuchen, werden wie bei den Tiefpassfiltern drei sinusförmige Signale am Eingang überlagert, so dass sie eine periodischen Folge von rechteckförmigen Pulsen annähern. Das Simulink-Modell aus der Datei `analog_hp_11.mdl`, das jetzt aus dem Programm `analog_hp11.m` aufgerufen wird, ist dem Modell aus Abb. 1.4, das für die Tiefpassfilter benutzt wurde, ähnlich. Der Unterschied besteht darin, dass jetzt die Überlagerung des Eingangs- und des Ausgangssignals für jedes Filter mit einer Verzögerung des Ausgangssignals erzwungen werden kann. Das sinusförmige Ausgangssignal im stationären Zustand ist bei Hochpassfiltern voreilend, im Gegensatz zu Tiefpassfiltern, wo es, wie gesehen, nacheilt.

Mit dem Programm `analog_hp12.m` und Modell `analog_hp_12.mdl` wird der Einfluss der Hochpassfilter auf bandbegrenztes Rauschen untersucht . Es wird weißes Rauschen mit einem Bandpassfilter gefiltert, so dass man Spektralkomponenten im Bereich von 1000 Hz bis 5000 Hz erhält. Der Simulink-Block dieses Filters (*Analog Filter Design*) wird aus dem *DSP-Blockset* über *Filtering, Filter Design* entnommen.

Wenn die Durchlassfrequenz der Hochpassfilter 10 Hz ist (zwei Dekaden kleiner als die kleinste im Signal vorkommende Frequenz), sind die Unterschiede zwischen dem Eingangs- und Ausgangssignal sehr klein. Dagegen sind die Unterschiede bei einer Durchlassfrequenz von 100 Hz schon viel größer. Bei 1000 Hz Durchlassfrequenz sind die Verzerrungen so groß, dass man das Ausgangssignal dem Eingangssignal nicht mehr zuordnen kann.

1.4 Verzerrungen modulierter Signale durch Bandpassfilter

Wie schon bekannt ist, führen Filter mit einem linearen Phasengang, der bei $\omega = 0$ nicht null oder kein Vielfaches von 2π ist, zu Verzerrungen. Diese Verzerrungen (in der Literatur als

Phase-Intercept-Distortion bekannt) sind jedoch vielmals nur von akademischem Interesse und sollten für jede Anwendung auf Relevanz untersucht werden.

In diesem Abschnitt werden die Verzerrungen von amplituden- und frequenzmodulierten Signalen beim Durchgang durch Bandpassfilter untersucht. Es wird gezeigt, dass das modulierte Bandpasssignal zwar durch die *Phase-Intercept-Distortion* verzerrt wird, aber das nachrichtentragende modulierende Signal lediglich mit der Gruppenlaufzeit des Filters verzögert, aber nicht verzerrt wird.

Wir betrachten unser Filter als ein System mit konstantem Amplitudengang und mit einem linearem Phasengang der Form

$$\varphi(\omega) = -G_r\,\omega - \theta_0, \tag{1.22}$$

wobei G_r die konstante Gruppenlaufzeit ist. Die Phasenlaufzeit, definiert durch

$$\tau_p = -\varphi(\omega)/\omega = G_r + \theta_0/\omega, \tag{1.23}$$

stellt die Verzögerung einer sinus- oder cosinusförmigen Komponente der Kreisfrequenz ω beim Durchgang durch das Bandpassfilter dar.

1.4.1 Verzerrung amplitudenmodulierter Signale

Ähnlich dem Vorgehen in Abschnitt 1.2 betrachten wir die Auswirkungen des nichtidealen Phasengangs anhand harmonischer Schwingungen. Eine Trägerschwingung $x_c(t) = cos(\omega_c t)$ sei mit einer harmonischen Schwingung der Kreisfrequenz ω_m amplitudenmoduliert, so dass eine Zweiseitenbandamplitudenmodulation mit Träger vorliegt. Das amplitudenmodulierte Signal (kurz AM-Signal) hat dann die Form:

$$x(t) = [1 + m\,cos(\omega_m t)]cos(\omega_c t) = e(t)\,cos(\omega_c t), \tag{1.24}$$

wobei $e(t) = 1 + m\,cos(\omega_m\,t)$ die Hülle des Modulationssignals bildet, ω_c bzw. ω_m die Träger- und Modulationsfrequenz sind und m den Modulationsindex darstellt $(0 \leq m \leq 1)$. Durch einfache mathematische Umformungen kann das Signal auch in folgender Form dargestellt werden:

$$x(t) = cos(\omega_c t) + \frac{m}{2}\,cos[(\omega_c - \omega_m)t] + \frac{m}{2}\,cos[(\omega_c + \omega_m)t] \tag{1.25}$$

Sie zeigt, dass das Signal aus dem Trägersignal und zwei Seitenbändern der Frequenzen $\omega_c - \omega_m$ bzw. $\omega_c + \omega_m$ besteht.

Die Antwort $y(t)$ des Filtersystems besteht aus der Superposition der Antworten auf diese drei Komponenten, wobei jede Komponente die ihr entsprechende Phasenverschiebung erfährt:

$$\begin{gathered} y(t) = cos[\omega_c(t - Gr) - \theta_0] + \\ \frac{m}{2}\,cos[(\omega_c - \omega_m)(t - Gr) - \theta_0] + \frac{m}{2}\,cos[(\omega_c + \omega_m)(t - Gr) - \theta_0] = \\ cos[\omega_c(t - Gr) - \theta_0] + m\,cos[\omega_c(t - G_r) - \theta_0]\,cos[\omega_m(t - G_r)] \end{gathered} \tag{1.26}$$

Die Eigenschaften dieser Antwort werden sichtbar, wenn sie wie folgt ausgedrückt wird:

$$y(t) = \{1 + m\,cos[\omega_m(t - G_r)]\}cos\{\omega_c[t - (G_r + \theta_0/\omega_c)]\} \tag{1.27}$$

Die Trägerfrequenz wird mit der Phasenlaufzeit $\tau_p(\omega_c) = G_r + \theta_0/\omega_c$ und die Hülle mit der Gruppenlaufzeit G_r verzögert. Das Signal $y(t)$ ist verzerrt. Allerdings wird die Information, die in der Hülle enthalten ist, nicht verzerrt, sondern nur mit G_r verzögert. Somit wird das modulierende Signal nicht verzerrt, wenn der Amplitudengang und die Gruppenlaufzeit des Bandpassfilters im Frequenzbereich des modulierten Signals konstant bleiben. In der Praxis versucht man diese Bedingung annähernd zu erfüllen.

Sind diese Bedingungen grob verletzt, so sind die Verzerrungen, die das modulierende Signal erfährt, von der Art der Demodulation abhängig. Bei kohärenter Demodulation entstehen lineare Verzerrungen so, dass das demodulierte Signal dem Faltungsergebnis des modulierenden Signals mit der Impulsantwort der äquivalenten Basisbanddarstellung des Bandpassfilters entspricht. Wird jedoch Hüllkurvendemodulation durchgeführt, so entstehen im allgemeinen Fall auch nichtlineare Verzerrungen. Lediglich wenn das Bandpassfilter eine gerade Symmetrie bezüglich der Trägerfrequenz aufweist, entstehen nur dieselben linearen Verzerrungen wie bei kohärenter Demodulation [35].

1.4.2 Verzerrung frequenzmodulierter Signale

Ein mit einer harmonischen Schwingung der Kreisfrequenz ω_m frequenzmodulierter Träger hat die Form:

$$x(t) = cos[\phi(t)] = cos[\omega_c t + \mu\ sin(\omega_m t)], \tag{1.28}$$

wobei ω_c die Trägerfrequenz ist, $\mu = \Delta\omega/\omega_m$ der Modulationsindex und $\Delta\omega$ die größte Frequenzabweichung der Modulationsfrequenz ω_m. Die momentane Phase des Signals ist $\phi(t) = \omega_c t + \mu\ sin(\omega_m t)$ und die Momentanfrequenz ist:

$$\omega(t) = \frac{d\phi(t)}{dt} = \omega_c + \Delta\omega\ cos(\omega_m t) \tag{1.29}$$

Die Darstellung dieses Signals als Fourier-Reihe führt zu

$$x(t) = \sum_{k=-\infty}^{\infty} J_k(\mu) cos[(\omega_c + k\ \omega_m)t], \tag{1.30}$$

wobei $J_k(\mu)$ die Bessel-Funktionen erster Art der Ordnung k sind. Das Spektrum besteht somit aus der Trägerfrequenz und aus unendlich vielen Spektrallinien mit Vielfachen von ω_m als Abstände zur Trägerfrequenz.

Die Antwort des Filtersystems ist wieder eine Superposition dieser harmonischen Signale, verschoben jeweils um die entsprechenden Phasen:

$$y(t) = \sum_{k=-\infty}^{\infty} J_k(\mu) cos[(\omega_c + k\ \omega_m)(t - G_r) - \theta_0] \tag{1.31}$$

Auch hier ist das Ausgangssignal $y(t)$ verzerrt, weil θ_0 nicht immer ein Vielfaches von 2π ist. Die Antwort (1.31) kann auch als frequenzmoduliertes Signal (wie in 1.28) geschrieben werden:

$$y(t) = cos\{\omega_c(t - G_r) - \theta_0 + \mu\ sin[\omega_m(t - G_r)]\} \tag{1.32}$$

Die Augenblicksphase $\phi(t)$ ist

$$\phi(t) = \omega_c(t - G_r) - \theta_0 + \mu\ sin[\omega_m(t - G_r)] \tag{1.33}$$

und die Momentanfrequenz ist als Ableitung der Phase gegeben durch:

$$\omega(t) = \frac{d\phi(t)}{dt} = \omega_c + \Delta\omega\ cos[\omega_m(t - G_r)] \tag{1.34}$$

Man bemerkt also, dass bei konstanter Gruppenlaufzeit und konstantem Amplitudengang die nachrichtentragende Momentanfrequenz nicht verzerrt, sondern lediglich um die Gruppenlaufzeit G_r verzögert wird. Ein Versatz θ_0 ist dabei belanglos.

Im allgemeinen Fall eines nichtidealen Frequenzgangs des Bandpassfilters sind die Auswirkungen jedoch schwerwiegender als bei der Amplitudenmodulation. Eine frequenzabhängige Gruppenlaufzeit führt nicht nur zu Phasenverzerrungen, sondern auch zu einer Amplitudenmodulation des modulierten Signals. Und andererseits führt ein nichtidealer Amplitudengang nicht nur zu Verzerrungen in der Amplitude des Signals (die bei idealer Demodulation für frequenzmodulierte Signale irrelevant sind), sondern auch zu nichtlinearen Verzerrungen der Phase, und damit des modulierenden Signals [35]. Insoweit sind die Anforderungen an Amplituden- und Phasengang der Filter beim Einsatz mit frequenzmodulierten Signalen strenger als beim Einsatz mit amplitudenmodulierten Signalen.

Experiment 1.3: Frequenzgang und Gruppenlaufzeit von Bandpassfiltern

Bandpassfilter werden aus Tiefpassfilter mit folgender Transformation erhalten:

$$\begin{aligned} j\omega_{BP} &= (j\omega_{TP}/\omega_0 + \omega_0/\omega_{TP})/\gamma \\ \gamma &= (\omega_2 - \omega_1)/\omega_0 \end{aligned} \tag{1.35}$$

Hier sind ω_1 , ω_2 die untere und obere Grenze des Durchlassbereichs und die Frequenz ω_0 ist durch $\omega_0 = \sqrt{\omega_1\ \omega_2}$ definiert.

Wenn die relative Bandbreite γ klein ist, kann man ω_0 als Mittenfrequenz bezeichnen und durch das arithmetische Mittel $\omega_0 = (\omega_1 + \omega_2)/2$ annähern.

In MATLAB werden die Tiefpassprototyp-Filter, die über die Funktionen aus Tabelle 1.1 berechnet werden, mit Hilfe der Funktionen aus Tabelle 1.2 in Bandpassfilter transformiert.

Im Programm `analog_bp1.m` werden Bandpassfilter aus Tiefpassprototyp-Filtern entwickelt und deren Frequenzgang bzw. Gruppenlaufzeit dargestellt (Abb. 1.15). Über die Variable `Typ` wird der Typ des Filters (Bessel, Butterworth, Tschebyschev I oder Elliptisch) gewählt und danach werden die Prototyp- bzw. die entsprechenden Bandpassfilter berechnet:

```
% -------- Wahl des Filtertyps
Typ = 2;        % 1 = Bessel; 2 = Butterworth;
% 3 = Tschebyschev I; 4 = Ellip
% ------- Felder der Koeffizienten
btp = zeros(4, nord+1);   % TP-Koeffizienten
atp = zeros(4, nord+1);
```

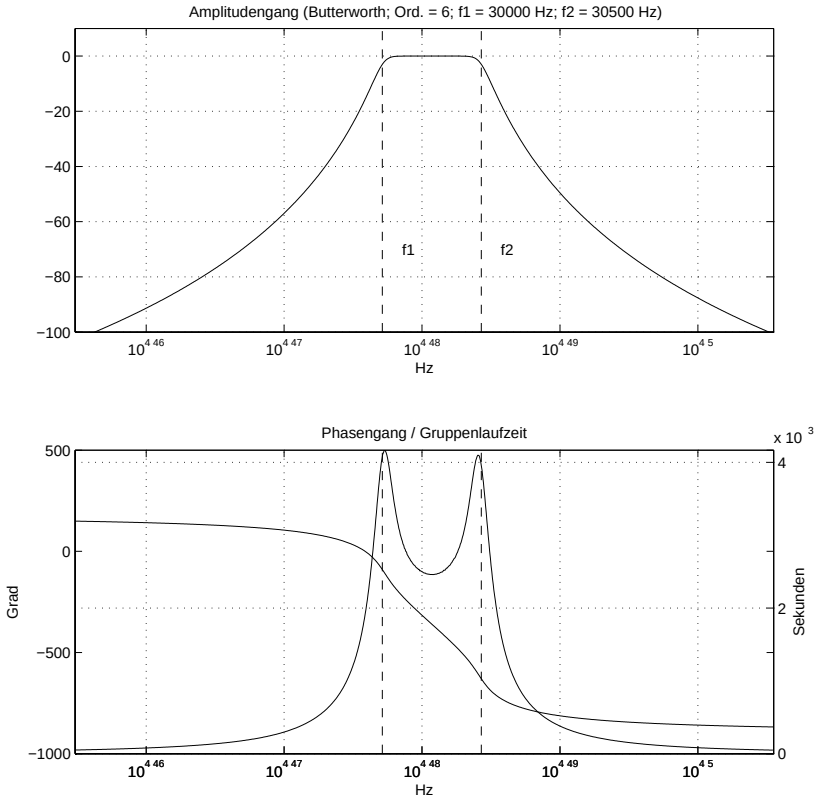

*Abb. 1.15: Frequenzgang und Gruppenlaufzeit des BP-Filters mit f1 = 30000 Hz und f2 = 35000 Hz (*analog_bp1.m)

```
bbp = zeros(4, 2*nord+1); % BP-Koeffizienten
abp = zeros(4, 2*nord+1);

% ------- Entwicklung eines Bessel TP- und BP-Filters
[z,p,k] = besselap(nord);
[btp(1,:),atp(1,:)] = zp2tf(z,p,k);  % TP_Prototyp
[bbp(1,nord+1:end),abp(1,:)]=lp2bp(btp(1,:),atp(1,:),w0,Bw);
                                % BP-Bessel-Filter

% ------- Entwicklung eines Butterworth TP- und BP-Filters
[z,p,k] = buttap(nord);
[btp(2,:),atp(2,:)] = zp2tf(z,p,k);  % TP-Prototyp
[bbp(2,nord+1:end),abp(2,:)]=lp2bp(btp(2,:),atp(2,:),w0,Bw);
                                % BP-Butterworth-Filter
.......
```

Der Frequenzgang und die Gruppenlaufzeit werden ähnlich wie bei den TP- oder HP-Filtern ermittelt. Für die Darstellung des Phasengangs und der Gruppenlaufzeit wurde die Funktion **plotyy** verwendet, die es erlaubt, zwei Variablen mit einem sehr unterschiedlichen Wertebereich im selben Fenster darzustellen. Die Funktionsachse für den Phasengang ist links und für die Gruppenlaufzeit rechts (Abb. 1.15 unten). Die benutzte Form des Befehls ist:

```
[haxis, hline1, hline2] = plotyy(x1,y1,x2,y2,@semilogx);
```

Dabei sind `haxis,hline1,hline2` die Zeiger für die graphischen Achsen (Objekt `axes`) für die erste ($y_1(x_1)$) bzw. die zweite ($y_2(x_2)$) darzustellende Funktion. Mit **@semilogx** wird in diesem Beispiel die MATLAB-Funktion, die für die Darstellung verwendet werden soll, angegeben. Für jede darzustellende Kurve kann eine andere Darstellfunktion gewählt werden.

Der Zeiger `haxis` ist ein Vektor mit zwei Komponenten, so dass `haxis(1)` bzw. `haxis(2)` die Zeiger auf die Achsen der beiden Darstellungen sind. Mit dem Befehl **axes**`(haxis(1))` werden z.B. die Achsen der ersten Darstellung gewählt und mit den nachfolgenden Befehlen werden diese dann entsprechend gestaltet. Ähnlich werden die Achsen der zweiten Darstellung mit **axes**`(haxis(2))` gewählt und ebenfalls parametriert. Hier werden auch die zwei vertikalen Linien zur Abgrenzung des Durchlassbereichs erzeugt. Das Programm `analog_bp2.m` unterscheidet sich von `analog_bp1.m` nur dadurch, dass zur Berechnung von ω_0 der arithmetische anstatt des geometrischen Mittelwerts verwendet wird.

Die Funktion `analog_bp11` ist eine Erweiterung von `analog_bp1.m`, um für weitere Experimente beliebige Filter (Filterkoeffizienten) zu berechnen:

```
function [b,a]=analog_bp11(f1,f2,nord,Typ)
```

Die Argumente sind im Kontext dieses Abschnitts leicht zu verstehen. Die Frequenzen `f1` und `f2` sind die Grenzen des Durchlassbereichs, und `nord` und `Typ` sind die Ordnung bzw. der Typ des Filters. In den Vektoren `b` bzw. `a` werden die Koeffizienten des gewünschten Filters geliefert.

1.5 Rekonstruktion zeitkontinuierlicher Signale

In diesem Abschnitt wird mit Hilfe eines Experiments die Rekonstruktion eines zeitkontinuierlichen Signals aus seinen Abtastwerten untersucht (Abb. 1.16). Dieser Vorgang ist in der Praxis unter dem Namen Digital/Analog-Wandlung bekannt.[6]

Zeitdiskrete Signale werden in den folgenden Kapiteln ausführlich behandelt. Einige Grundkenntnisse, soweit sie für das Verständnis dieses Experiments notwendig sind, werden hier vorweggenommen.

Experiment 1.4: Tiefpassfilter als Glättungsfilter

Zeitdiskrete Signale $x[nT_s]$ als die Werte des zeitkontinuierlichen Signals $x(t)$ zu den diskreten Zeitpunkten nT_s mit $n = ... -2, -1, 0, 1, 2, ...$ sind als Zahlen in einem Speicher eines Rechenwerkes gespeichert. Die Zeitdauer T_s ist der Abstand zwischen den Zeitpunkten, zu

[6] Streng genommen beziehen sich die Begriffe *digital* und *analog* nur auf den Wertebereich eines Signals (digital = wertdiskret und analog = wertkontinuierlich). Häufig wird in der Praxis, gerade im Zusammenhang mit A/D- und D/A-Wandlung, oftmals der Begriff *digital* für wert- *und* zeitdiskrete Signale verwendet, während *analog* für wert- *und* zeitkontinuierliche Signale steht.

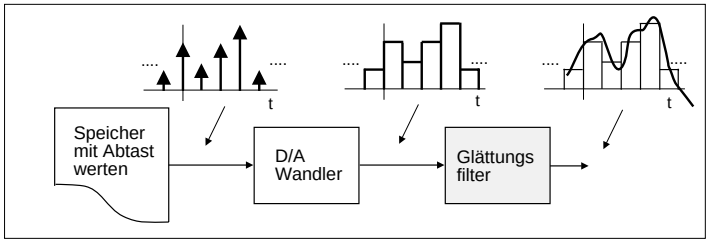

Abb. 1.16: Digital/Analog-Wandlung

denen die Abtastwerte aus dem zeitkontinuierlichen Signal entnommen werden und wird Abtastperiode genannt. Ihr Kehrwert ist die Abtastfrequenz $f_s = 1/T_s$.

Da Speicher immer nur eine endliche Genauigkeit haben können, ist auch der Wertebereich der Abtastwerte diskret. Dieser Aspekt der Quantisierung wertkontinuierlicher Größen soll hier außer Acht bleiben und es wird angenommen, dass diese digital mit großer Auflösung (mit sehr vielen Bit) dargestellt sind.

Der D/A-Wandler bildet aus den zeitdiskreten Werten ein zeitkontinuierliches Signal, mit konstanten Werten zwischen den Abtastzeitpunkten. Der Wandler verhält sich wie ein Halteglied nullter Ordnung mit einer Impulsantwort, die in Abb. 1.17 oben dargestellt ist [68], [56]. Die zeitdiskreten Abtastwerte aus dem Speicher, betrachtet als Dirac-Funktionen (oder Dirac-Impulse), gefaltet mit der Impulsantwort des Haltegliedes nullter Ordnung, führen zum treppenförmigen Signal am Ausgang des Wandlers.

Sicher gibt es im Speicher keine Dirac-Funktionen, diese sind nur als Vorstellung oder Modell notwendig, um die zeitkontinuierliche Faltung auch für die zeitdiskreten Signale anwenden zu können. Wichtig ist, dass dieses Modell eine mathematische Beschreibung des Wandlers ermöglicht, welche sein treppenförmiges Ausgangssignal ergibt.

Der Frequenzgang $X(f)$ des D/A-Wandlers als Fourier-Transformierte der Impulsantwort ist [34]:

$$X(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt = \int_{0}^{T_s} e^{-j2\pi ft} dt \tag{1.36}$$

oder

$$X(f) = e^{-j\pi fT_s} \frac{T_s}{2} \frac{sin(\pi fT_s)}{\pi fT_s} \tag{1.37}$$

Der Phasengang wird hauptsächlich durch den komplexen Drehzeiger $e^{-j\pi fT_s}$ bestimmt und ist somit linear, bis auf Sprünge von π an den Stellen, an denen die Sinusfunktion aus Gl. (1.37) ihr Vorzeichen wechselt. Der Phasengang des Haltegliedes führt also nicht zu Verzerrungen des Signals. Der Amplitudengang, als Betrag des komplexen Frequenzgangs, entspricht dem Betrag der $sin(x)/x$-Funktion, kurz sinc-Funktion. Die Nullstellen dieser Funktion für positive Werte der Frequenz liegen bei $k/T_s, k = 1, 2, 3, ...$

In Abb. 1.17 ist unten der Amplitudengang im Frequenzbereich $f \in [0, 2f_s]$ dargestellt. Er wurde mit dem Programm `zero_order_hold.m` erzeugt. Der Frequenzbereich von 0

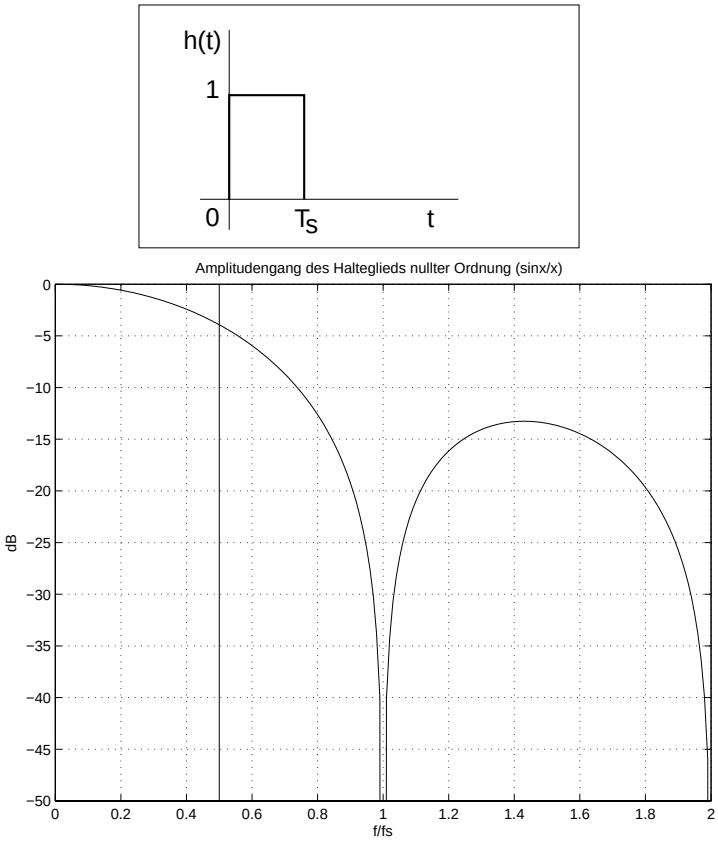

Abb. 1.17: Impulsantwort und Amplitudengang des Halteglieds nullter Ordnung (zero_order_hold.m)

bis $f_s/2$, auch als Nyquist-Bereich bekannt, ist der interessierende Frequenzbereich des abgetasteten Signals, den das nachgeschaltete ideale Glättungsfilter extrahiert. Aufgrund dieses Amplitudengangs des Wandlers entstehen im Nyquist-Bereich Amplitudenverzerrungen bis zu ca. -4 dB (Faktor $\cong 0.63$).

Das ideale Glättungsfilter sollte als Durchlassbereich den Nyquist-Bereich haben und dann unmittelbar in den Sperrbereich, beginnend bei $f_s/2$, mit unendlich hoher Dämpfung übergehen. Da dieses Verhalten, genau wie beim Antialiasing-Filter vor der A/D-Wandlung, nicht realisierbar ist, ist es ratsam eine Überabtastung des zeitkontinuierlichen Signals anzuwenden. Wenn die maximale Frequenz eines Signals f_{max} ist, dann sollte man eine Abtastfrequenz f_s wählen, die durch

$$f_s = k_u(2f_{max}) \tag{1.38}$$

gegeben ist, wobei k_u den Überabtastfaktor darstellt. Der Wert $k_u = 1$ (keine Überabtastung) entspricht dem Wert gemäß Abtasttheorem. In der Praxis werden je nach Anwendung Überabtastfaktoren zwischen 1.5 und 5 benutzt.

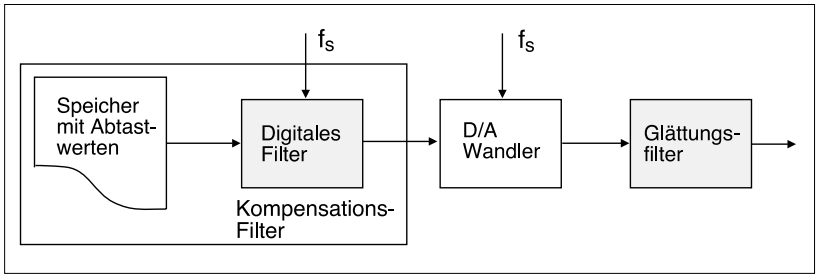

Abb. 1.18: Struktur des Ausgangs eines D/A-Wandlers mit digitalem Kompensationsfilter

In diesem Experiment wird der Rekonstruktionsfehler als Folge des Frequenzgangs des D/A-Wandlers und des Glättungsfilters untersucht. Dazu wird ein zeitkontinuierliches Signal abgetastet und anschließend wird mit einen D/A-Wandler (simuliert als Halteglied nullter Ordnung) und einem Glättungsfilter das ursprüngliche Signal rekonstruiert. Durch Differenzbildung wird die Qualität der Rekonstruktion bestimmt. Es wird ebenfalls untersucht, wie die Amplitudenverzerrung des D/A-Wandlers mit einem digitalen Filter vor der Wandlung kompensiert werden kann. Abb. 1.18 zeigt die neue Struktur der Rekonstruktion eines zeitkontinuierlichen Signals mit Kompensationsfilter, D/A-Wandler und analogem Glättungsfilter.

Das Kompensationsfilter soll als Amplitudengang eine der sinc-Funktion inverse Charakteristik aufweisen und linearphasig sein. In der Literatur [3] werden als Näherung dieser Charakteristik zwei sehr einfache digitale Kompensationsfilter vorgeschlagen. Die Lösung mit FIR-Filter entspricht folgender Differenzengleichung:

$$y[nT_s] = (-1/16)x[nT_s] + (9/8)x[(n-1)T_s] + (-1/16)x[(n-2)T_s] \quad (1.39)$$

Für die Realisierung als IIR-Filter wird die Differenzengleichung

$$y[nT_s] = (9/8)x[nT_s] - (1/8)y[(n-1)T_s] \quad (1.40)$$

empfohlen. In der kompakten Schreibweise der z-Transformation werden diese Filter mit folgenden Übertragungsfunktionen beschrieben:

$$H_{FIR}(z) = -(1/16) + (9/8)z^{-1} - (1/16)z^{-2} \quad (1.41)$$

und

$$H_{IIR}(z) = \frac{9/8}{1 + (1/8)z^{-1}} \quad (1.42)$$

Zur Simulation wird das Simulink-Modell (`rekonstr_1.mdl`) aus Abb. 1.19 verwendet, das mit dem Programm `rekonstr1.m` parametriert und aufgerufen wird. Das Eingangssignal der Simulation wird aus weißem Rauschen, das mit einem Tiefpassfilter (*Analog Filter Design1*-Block) bandbegrenzt wird, generiert. Als Bandbreite des Signals wurde `frausch`=250 Hz gewählt und die Abtastfrequenz wurde auf `fs`=1000 Hz festgelegt. Damit erhält man einen Überabtastfaktor von $k_u = f_s/(2f_{max}) = 2$.

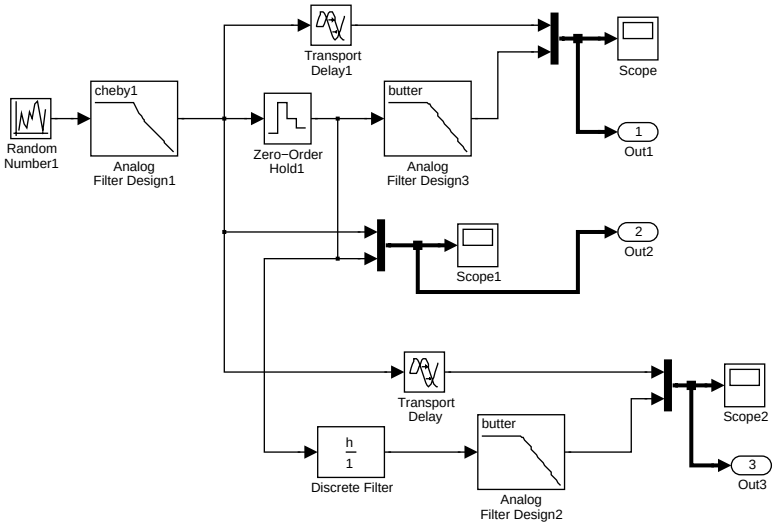

*Abb. 1.19: Simulation der Rekonstruktion zeitkontinuierlicher Signale (*rekonstr_1.mdl, rekonstr1.m)

Der D/A-Wandler wird mit dem Block *Zero-Order Hold1* (Halteglied nullter Ordnung) simuliert. Als Glättungsfilter wird ein Tiefpassfilter von Typ Butterworth (*Analog Filter Design3*-Block) eingesetzt.

Auf dem Block *Scope1* sind das zeitkontinuierliche und das mit Halteglied nullter Ordnung abgetastete Signal (wie in Abb. 1.20 gezeigt) überlagert dargestellt. Auf dem Block *Scope* werden das verzögerte zeitkontinuierliche Signal und das rekonstruierte Signal dargestellt. Die Verzögerung *Transport Delay1* ist so einzustellen, dass die Laufzeit des Glättungstiefpasses kompensiert wird. Man erreicht dies durch Versuche unter Beobachtung der zeitrichtigen Überlagerung der beiden Signale auf dem Block *Scope*.

Zur numerischen Evaluation des Rekonstruktionsfehlers wird die erforderliche Verzögerung in dem Programm `rekonstr1.m` durch Minimierung der Summe der Beträge der Differenzen der beiden Signale automatisch eingestellt.

```
......
% Bestimmung der optimalen Verzögerung
nv = 200;    % Anfangsindex
diff_y = zeros(2*nv,1);

y1 = yout(nv:end-nv,1);
for k = 1:2*nv
    diff_y(k) = sum(abs(yout(k:length(y1)+k-1,2) - y1));
end;
min_diff_1 = min(diff_y);
p1 = find(diff_y == min_diff_1);
......
```

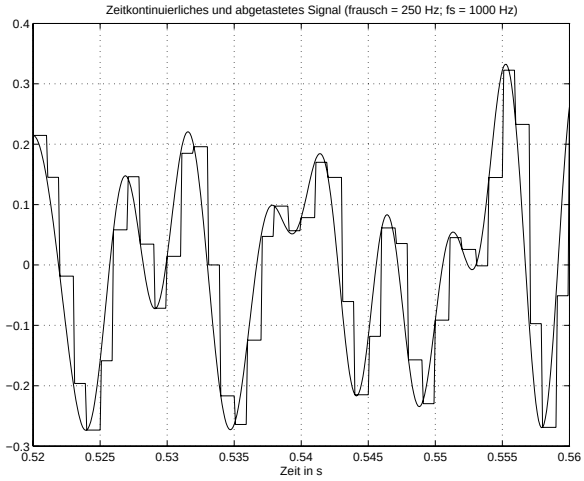

*Abb. 1.20: Zeitkontinuierliches und mit einem Halteglied nullter Ordnung abgetastetes Signal (*rekonstr_1.mdl, rekonstr1.m)

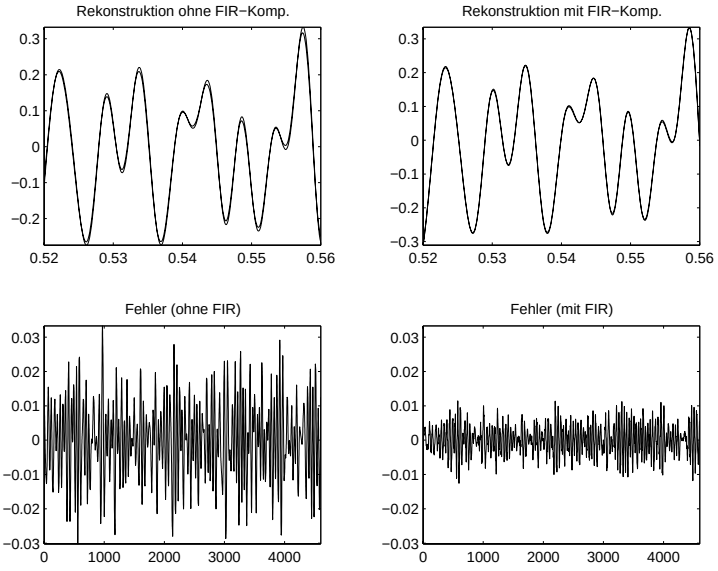

*Abb. 1.21: Rekonstruktion mit und ohne FIR-Kompensation (*rekonstr_1.mdl, rekonstr1.m)

Hierfür werden in der Senke *Out1* die beiden Signale als Spalten des Feldes `yout` (`yout(:,1)` und `yout(:,2)`) gespeichert. Ebenso wird im Vektor `tout` die Simulationszeit gespeichert. Diese Einstellungen werden mit der Option *Data Import/Export* des Dialogs