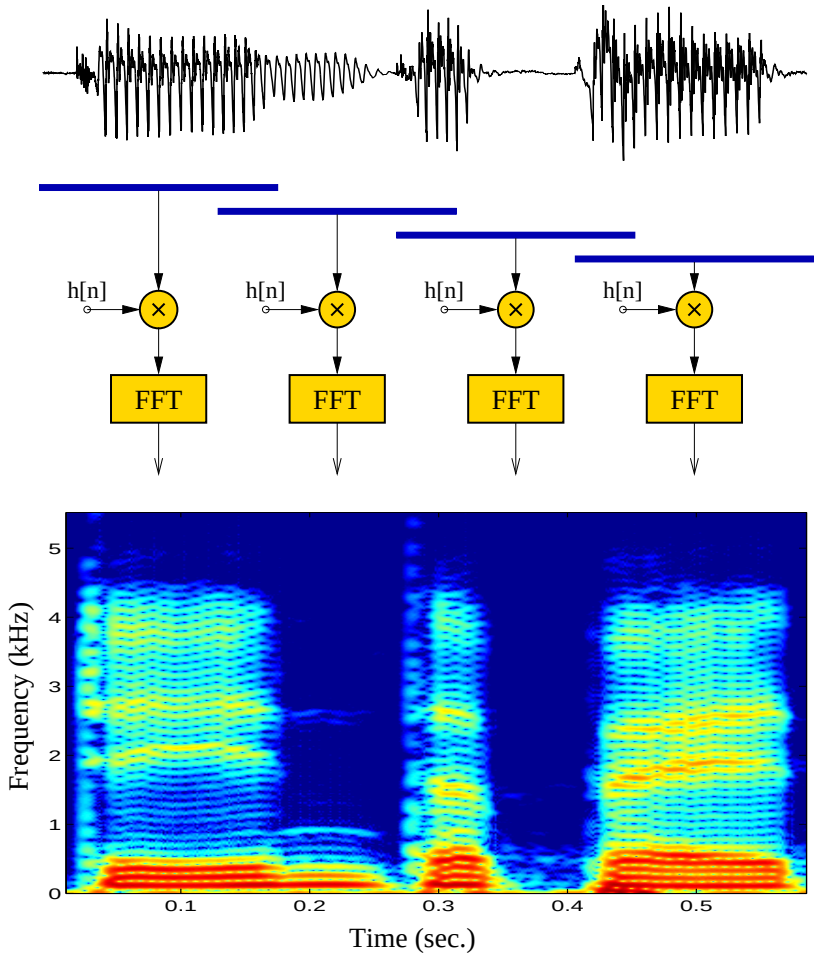
G. Doblinger TU-Wien

MATLAB®-Programmierung

in der digitalen Signalverarbeitung



Aufgabensammlung

MATLAB®-Programmierung in der digitalen Signalverarbeitung
Aufgabensammlung, Stand Oktober 2007

Dr. Gerhard Doblinger
Institut für Nachrichtentechnik und Hochfrequenztechnik
Technische Universität Wien
Gusshausstr. 25/389
A-1040 Wien

Tel. 58801 38927, Fax 58801 38999
Email: Gerhard.Doblinger@tuwien.ac.at
Internet: www.nt.tuwien.ac.at/about-us/staff/gerhard-doblinger/

Inhaltsverzeichnis

1 Zeitdiskrete Signale

Beispiel 1.1

Für die folgenden zeitdiskreten Signale soll der Zeitbereich $n \in [-20, 20]$ verwendet werden. Stellen Sie alle Signale graphisch am Bildschirm dar.

$$
\begin{aligned}
x_1(n) &= \delta(n) \quad \text{(Einsimpuls)} \\
x_2(n) &= \sigma(n) \quad \text{(Sprungfunktion)} \\
x_3(n) &= (-1)^n \\
x_4(n) &= \begin{cases} 0 & n \leq 0 \\ n & n > 0 \end{cases} \\
x_5(n) &= \begin{cases} -5 & n \leq -5 \\ n & -5 < n < 5 \\ 5 & n \geq 5 \end{cases}
\end{aligned}
$$

Probieren Sie insbesondere bei den zweistufigen Signalen mehrere Lösungen aus.

Beispiel 1.2

Die zeitdiskrete Cosinusschwingung mit der "Frequenz" θ sei gegeben durch

$$x(n) = \cos \theta n \quad \text{mit } n \in [-20, 20].$$

Für welche Werte θ ist $x(n)$ periodisch? Wie hängt θ mit der Anzahl der Signalwerte pro Periode zusammen? Variieren Sie θ im Intervall $[0, 2\pi]$ und kommentieren Sie das Ergebnis.
Welche Bedeutung haben die Werte $\theta = \pi$ und $\theta = 2\pi$, wenn $x(n)$ durch Abtastung des analogen Signals $x_a(t) = \cos \omega t$ zustande kommt, d.h. $x(n) = x_a(nT)$?

Beispiel 1.3

Verwenden Sie die bisher vorgestellten MATLAB®-Befehle, um die folgenden Signale zu erzeugen. Dabei soll $n \in [-20, 20]$ sein und α in $[-1, 1]$ liegen.

$$
\begin{aligned}
x_1(n) &= \alpha^n \sigma(n) \\
x_2(n) &= (n+1)\alpha^n \sigma(n) \\
x_3(n) &= \alpha^{|n|} \\
x_4(n) &= \begin{cases} 0 & n < 0 \\ 1-\alpha^n & 0 \le n < 10 \\ 1-\alpha^{10} & n \ge 10 \end{cases} \\
x_5(n) &= \sigma(n) \sum_{k=0}^{n} \alpha^k
\end{aligned}
$$

Die Berechnung einer Summe mit MATLAB® haben Sie noch nicht kennengelernt, so dass Sie diese spezielle Summe eben anders berechnen müssen!

Beispiel 1.4

Für das zu betrachtende Zeitintervall wird wieder $n \in [-20, 20]$ gewählt. Beachten Sie, dass bei elementweisen arithmetischen Verknüpfungen von Vektoren die Vektorlängen und die Art (Zeilen- oder Spaltenvektor) übereinstimmen müssen. Mit Ausnahme von Addition und Subtraktion muss der Punktoperator (also `.*` oder `./` oder `.^`) für die elementweisen Operationen angewendet werden.

$$
\begin{aligned}
x_1(n) &= \sigma(n+4) - \sigma(n-4) \\
x_2(n) &= e^{-0.1(n+10)} \sigma(n+10) \\
x_3(n) &= \frac{1}{2}\Big(x_2(n) \pm x_2(-n)\Big) \\
x_4(n) &= x_2(-n-10) \\
x_5(n) &= x_1((n-4) \oplus 14)
\end{aligned}
$$

Beispiel 1.5

Entwickeln Sie eine MATLAB®-Funktion zur Faltung von Signalen, indem Sie eine Datei `conv_lin.m` mit folgendem Inhalt anlegen:

```
function y = conv_lin(h,x)

Nx = length(x);
Nh = length(h);
Ny = ...

% compute convolution matrix H
...
% compute result vector
y = ...
```

Als Alternative können Sie die Faltung auch mit FOR-Schleifen berechnen. Zum Vergleich der Rechenzeiten beider Funktionen fügen Sie am Beginn der Funktion `T0 = clock;` ein. Nach den Rechenoperationen erhalten Sie mit `etime(clock,T0)` die Rechenzeit. Wenden Sie Ihr Programm auf folgende Signale an:

a) $h(n) = 1/10,\ 0 \leq n \leq 9,\ x(n) = \sin \frac{\pi}{5} n,\ 0 \leq n < 50$

b) $\mathbf{h} = (1, -1, 1)^T,\ x(n) = \sin \frac{\pi}{3} n,\ 0 \leq n < 30$

c) $h(n) = \delta(n) - \delta(n-1),\ x(n) = \sigma(n).$

Interpretieren und vergleichen Sie die Ergebnisse mit jenen der MATLAB®-Funktion `conv()`.

Beispiel 1.6

```
h = h(:);
h1 = [h(2:N) ; h];
ic = [0:N-1]';
ir = N:-1:1;
Hc = ic(:,ones(N,1)) + ir(ones(N,1),:);
Hc(:) = h1(Hc);
```

Verwenden Sie diese MATLAB®-Befehle für ein M-File `conv_cyc.m` zur Implementierung der zyklischen Faltung und verwenden Sie zum Testen folgende Signale:

a) $N = 10$, $h(n) = (-1)^n$, $0 \leq n \leq N-1$, $x(n) = 1$, $0 \leq n \leq N-1$

b) Wiederholen Sie Punkt a) für $N = 11$

c) $N = 10$, $h(n) = \delta(n) - \delta(n-1)$, $0 \leq n \leq N-1$, $x(n) = n$, $0 \leq n \leq N-1$. Vergleichen Sie hier das Ergebnis der zyklischen Faltung mit jenem der linearen Faltung.

Beispiel 1.7

```
Nx = 200;
SNR = -5;                       % SNR in dB
                                % (SNR = 20*log20(R))
w0 = 2*pi/Nx*5;                 % frequency of sinusoid
x = sin(w0*[0:Nx-1]);
r = randn(size(x));             % create gaussian noise
R = norm(x)/norm(r);            % compute actual SNR
gain = R * 10^(-SNR/20);        % gain factor needed for
                                % given SNR
xn = x + gain*r;                % add noise to create
                                % noisy measurement data
```

Bestimmen Sie anhand der Autokorrelationsfunktion des verrauschten Sinussignals die Periodendauer (oder die Frequenz) in Abhängigkeit vom Signal-Rausch-Abstand SNR. Die automatische Berechnung der Periodendauer kann im vorliegenden Fall einfach durch Zählen der Nulldurchgänge des Signals erfolgen.

a) Ab welchem SNR nimmt der Messfehler drastisch zu?

b) Wie hängt die Messgenauigkeit mit der Messdauer (Signallänge `Nx`) zusammen?

Vergleichen Sie das Ergebnis mit den Messungen der Periodendauer direkt am verrauschten Sinussignal.
Hinweis: Die Bestimmung der Nulldurchgänge ist in MATLAB® denkbar einfach. Wandeln Sie die Autokorrelationsfunktion in ein Rechtecksignal um. Durch Differenzbildung benachbarter Werte erhalten Sie Impulse an den Positionen der Nulldurchgänge. Die Positionen werden dann mit dem FIND-Befehl bestimmt.

2 Zeitdiskrete Systeme

Beispiel 2.1

Verwenden Sie MATLAB® zur Analyse der folgenden Systeme, d.h. klären Sie, ob die Systeme kausal, stabil, zeitinvariant und linear sind. Die "unbekannten" Systeme sind in Form von MATLAB®-Befehlen gegeben (das Eingangssignal $x(n)$ hat die Länge N):

a) **System 1:** `y = 2*x + 0.5;`

b) **System 2:** `y = [x(1) ; x(2:N)-x(1:N-1)];`

c) **System 3:**
`y = conv([1 1 1 1 1],x); y = min(y,1); y = max(y,-1);`

d) **System 4:** `y(1:2:N) = x(1:2:N); y(2:2:N) = -x(2:2:N);`

e) **System 5:** `y = [0:N-1]' .* x(:);`

f) **System 6:** `y = filter(1,[1 -0.9],x,0.5);`[1]

Schreiben Sie dazu eine MATLAB®-Funktion `blackbox.m` mit der die einzelnen Systeme durch das Argument `systype` ausgewählt werden können:

```
function y = blackbox(systype,x)
x = x(:);
N = length(x);
if     systype == 1,   y = 2*x + 0.5;
elseif systype == 2,   y = [x(1) ; x(2:N)-x(1:N-1)];
elseif ...
end
```

Setzen Sie dazu verschiedene Eingangssignale `x` in `blackbox()` ein. Testen Sie die Systeme nicht nur mit Einsimpulsen!

Beispiel 2.2

Versuchen Sie analytisch und experimentell mit MATLAB® herauszufinden, ob bei den folgenden Systemen eine Vertauschung der Teilsysteme $h_1(n)$, $h_2(n)$ bei Kettenschaltung zulässig ist, d.h.

$$\sum_{k=-\infty}^{\infty} h_1(k)\, h_2(n-k) = \sum_{k=-\infty}^{\infty} h_1(n-k)\, h_2(k), \quad \forall n.$$

a) $h_1(n) = \sigma(n)$ (Sprungfunktion) und $h_2(n) = \delta(n) - \delta(n-1)$
Zum Testen mit MATLAB® verwenden Sie $x(n) = \sigma(n)$.

b)

$$h_1(n) = \begin{cases} \frac{1}{N} & 0 \le n \le N-1 \\ 0 & \text{sonst} \end{cases} \qquad h_2(n) = \sin \frac{2\pi}{N} n \, \sigma(n).$$

Das Testsignal soll in diesem Fall $x(n) = \cos \frac{2\pi}{N} n$ sein.

Dokumentieren Sie die Probleme, die bei der Behandlung dieser Beispiele mit MATLAB® auftreten.

Beispiel 2.3

Finden Sie analytisch und experimentell mit MATLAB® die Impulsantworten $h_i(n)$ der inversen Systeme zu den gegebenen Impulsantworten $h(n)$. Von den Impulsantworten $h(n)$ wird dazu nur ein endlicher Ausschnitt mit $0 \le n < N$ betrachtet.

a) $h(n) = (0,8)^n \sigma(n)$

b) $h(n) = (0,8)^n \cos \frac{3\pi}{10} n \, \sigma(n)$

c)

$$h(n) = \begin{cases} 0,5 \left(1 + \cos \frac{2\pi}{N+1}(n+1)\right) & 0 \le n \le N-1 \\ 0 & \text{sonst} \end{cases}$$

Welche dieser Systeme sind überhaupt invertierbar? Experimentieren Sie mit verschiedenen Werten für N und N_i. Gibt es Unterschiede in den Lösungen der beiden Methoden? Testen Sie die Qualität der inversen Filterung durch Vergleich von Eingangs- und Ausgangssignal. Als Eingangssignal kann die Sprungfunktion $x(n) = \sigma(n)$ eingesetzt werden.

Beispiel 2.4

Bestimmen Sie mit MATLAB® die Antworten $y(n)$ folgender Systeme auf ein Eingangssignal der Form $x(n) = \cos \theta_0 n$ für verschiedene Frequenzwerte θ_0.

a) $h(n) = (0,9)^n \cos \frac{2\pi}{32} n \, \sigma(n)$

b) $h(n) = (0.9)^{|n|}$

c) $H\left(e^{j\theta}\right) = \dfrac{1}{1 - e^{j\theta}}$

Beispiel 2.5

Verwenden Sie die MATLAB®-Funktionen `filter` und `filtic` zur Bestimmung von $y(n)$ für folgende Differenzengleichungen:

a) $y(n) = 0.5y(n-1) + x(n) - 0.5x(n-1)$
für $x(n) = \sigma(n)$ und der Anfangsbedingung $y(-1) = 1$.

b) $y(n) - 0.6y(n-1) - 0.16y(n-2) = 5x(n)$
für $x(n) = \delta(n)$ und $y(-2) = \frac{25}{4}$, $y(-1) = 0$.

Zeigen Sie, dass sich die Systemantwort jeweils zusammensetzt aus

$$y(n) = y_h(n) + y_p(n)$$

($y_h(n)$ ist die Systemantwort für $x(n) \equiv 0$ und $y_p(n)$ jene für verschwindende Anfangsbedingungen.)

Beispiel 2.6

Entwickeln Sie eine MATLAB®-Funktion `[H,h,p0,z0] = eval_sys(b,a)` mit der ausgehend vom gegebenen Zähler- und Nennerpolynom `b,a` von $H(z)$ die Systemeigenschaften Frequenzgang `H` (Betrag und Phase), kausale Impulsantwort `h`, Pole und Nullstellen `p0,z0` berechnet und graphisch dargestellt werden können. Prüfen Sie anhand der Pole und Nullstellen die vorher angegebenen Zusammenhänge. Beobachten Sie auch den Einfluss der Pol/Nullstellenkonstellation auf die Impulsantwort.

a) $H(z) = \dfrac{\alpha z^2 + (1+\alpha)\beta z + 1}{z^2 + (1+\alpha)\beta z + \alpha}$ mit $\alpha = 0,8$ und $\beta = 0,5$.

b) $H(z) = \dfrac{2z(3z+17)}{(z-1)(z^2 - 6z + 25)}$

c) $H(z) = \dfrac{1 - z^{-6}}{1 - z^{-1}}$

d) $y(n) + 2y(n-1) + \frac{3}{2}y(n-2) + \frac{1}{2}y(n-3) = x(n) + x(n-1)$

Beispiel 2.7

In dieser Aufgabe soll die Koeffizientenempfindlichkeit der direkten Form mit jener der Kaskadenform verglichen werden. Die Quantisierung der Filterkoeffizienten `C` auf eine Wortlänge von `Nbits` wird mit

```
scale = 2^(Nbits-1);
Cq = 1/scale*round(scale*C);
```

vorgenommen. Verwenden Sie diese Quantisierung bei die Programmierung einer MATLAB®-Funktion `compare_filt(Nbits)`, mit der die Frequenzgänge des exakten Filters und der beiden quantisierten Filterstrukturen dargestellt werden. Testen Sie das Systemverhalten für Wortlängen im Bereich von 10 bis 32 Bits. Als Testfilter soll ein elliptisches Tiefpassfilter dienen, das Sie beispielsweise mit

```
N = 7;                                % N = filter order
[z0,p0,k] = ellip(N,0.1,70,1/8);      % get poles and zeros
[b,a] = ellip(N,0.1,70,1/8);          % or numerator and denominator
                                      % coefficients
```

entwerfen. Variieren Sie dabei die Filterordnung `N`.

Beispiel 2.8

Erweitern Sie die Funktion `compare_filt(Nbits)` von Aufgabe 2.7 für die Untersuchung der Koeffizientenempfindlichkeit der Parallelform digitaler Filter. Berechnen Sie auch für direkte Form, Kaskaden- und Parallelform den Fehler der Impulsantwort durch die Koeffizientenquantisierung. Als Fehlermaß wird dabei

$$\varepsilon^2 = \frac{\sum_{n=0}^{\infty}(h(n) - h_q(n))^2}{\sum_{n=0}^{\infty} h^2(n)}$$

eingesetzt. Die obere Summengrenze muss natürlich auf einen vernünftigen Wert beschränkt werden. Welche der drei Filterstrukturen hat den kleinsten Fehler ε (in Abhängigkeit von `Nbits`)?
Hinweis: Die Wurzel aus der Summe der Quadrate kann in MATLAB® mit der Funktion `norm()` ausgewertet werden.

Beispiel 2.9

Blockzustandsraumdarstellung:

$$\mathbf{x}(n+M) = \mathbf{A}_M \mathbf{x}(n) + \mathbf{B}_M \mathbf{u}(n),$$

mit der $N \times N$ Matrix

$$\mathbf{A}_M = \mathbf{A}^M$$

und der $N \times M$ Matrix

$$\mathbf{B}_M = \left(\mathbf{A}^{M-1}\mathbf{B}, \mathbf{A}^{M-2}\mathbf{B}, \ldots, \mathbf{AB}, \mathbf{B}\right).$$

$$\mathbf{y}(n) = \mathbf{C}_M \mathbf{x}(n) + \mathbf{D}_M \mathbf{u}(n),$$

mit der $M \times N$ Matrix

$$\mathbf{C}_M = \begin{pmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{M-1} \end{pmatrix}$$

und der $M \times M$ unteren Dreiecksmatrix

$$\mathbf{D}_M = \begin{pmatrix} \mathbf{D} & 0 & 0 & \ldots & 0 \\ \mathbf{CB} & \mathbf{D} & 0 & \ldots & 0 \\ \mathbf{CAB} & \mathbf{CB} & \mathbf{D} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{M-2}\mathbf{B} & \mathbf{CA}^{M-3}\mathbf{B} & \mathbf{CA}^{M-4}\mathbf{B} & \ldots & \mathbf{D} \end{pmatrix}.$$

Die Aufstellung der Matrizen für die Zustandsraumdarstellung bei Blockverarbeitung ist eine besonders empfehlenswerte Übung für die Matrizenverarbeitung mit MATLAB®. Entwickeln Sie daher eine MATLAB®-Funktion `tf2blockss()` zur Berechnung der Matrizen $\mathbf{A}_M$, $\mathbf{B}_M$, $\mathbf{C}_M$ und $\mathbf{D}_M$. Als Funktionsparameter werden die Koeffizientenvektoren `a` und `b` der Differenzengleichung und die Blocklänge M verwendet.
Zum Testen kann die folgende blockweise Signalverarbeitung mit MATLAB® eingesetzt werden:

```
M = 40;                              % block length
Nu = 5*M;                            % signal length is a multiple of M
u = ones(Nu,1);                      % input signal
[b,a] = ellip(5,0.1,70,1/8);         % design an elliptic low pass filter
[Am,Bm,Cm,Dm] = tf2blockss(b,a,M);   % block state-space matrices
Ns = max(length(a),length(b))-1;     % state vector length
```

```
x = zeros(Ns,1);                  % initial state vector
y = zeros(size(u));

for n = 1:M:Nu                    % block filter operation
   nb = n:n+M-1;                  % block time indices
   y(nb) = Cm*x + Dm*u(nb);
   x = Am*x + Bm*u(nb);
end
```

Beispiel 2.10

```
[b,a] = ellip(7,0.1,100,1/8); % elliptic filter design
[A1,B1,C1,D1] = tf2ss(b,a);   % compute state-space decomposition
[V,L] = eig(A1);              % compute eigenvalue decomposition
A1 = diag(L);                 % A1 = vector of diagonal elements of L
B1 = inv(V)*B1;
C1 = C1*V;

% compute block state-space matrices Am, Bm, Cm, Dm

N = length(A1);               % state-space dimension
M = round(sqrt(2*N));         % optimum block length
Bm = zeros(N,M);
Cm = zeros(M,N);
Am = 1;
for n = M-1:-1:1
   Am = Am .* A1;
   Bm(:,n) = Am .* B1;
   Cm(M-n+1,:) = C1 .* (Am.');
end
Am = Am .* A1;
Bm(:,M) = B1;
Cm(1,:) = C1;
Dm = toeplitz([D1 ; real(Cm(1:M-1,:)*B1)],[D1 zeros(1,M-1)]);
```

Verwenden Sie diese MATLAB®-Anweisungen zur Entwicklung einer Funktion `y = blockss_filt(b,a,x)` zur Signalfilterung mit der Blockzustandsraumdarstellung. Beachten Sie, dass durch die Eigenwertzerlegung die Matrizen komplexwertig sind. Dabei kann durch Vermeidung des Rechnens mit konjugiert komplexen Elementen der Rechenaufwand reduziert werden. Vergleichen

Sie die Arithmetikoperationen pro Abtastintervall mit jenen der Kaskaden- und Parallelform.

3 Diskrete Fouriertransformation

Beispiel 3.1

Das periodische Signal $\tilde{x}(n)$ mit der Periodendauer N soll durch eine Fourierreihe angenähert werden, die weniger als N Koeffizienten hat:

$$\tilde{x}_M(n) = \frac{1}{N} \sum_{k=0}^{M-1} c_k e^{j\frac{2\pi}{N}nk},$$

mit $M \leq N$. Berechnen Sie den Fehler

$$\varepsilon(M) = \sum_{k=0}^{N-1} |\tilde{x}(n) - \tilde{x}_M(n)|^2$$

in Abhängigkeit von M für verschiedene periodische Signale mit der Periode N. Überprüfen Sie außerdem die Parsevalsche Gleichung

$$\sum_{n=0}^{N-1} |\tilde{x}(n)|^2 = \sum_{k=0}^{N-1} |c_k|^2.$$

Hinweis: In MATLAB® können Sie die Fourierreihenkoeffizienten am einfachsten mit `c = fft(x,M)` berechnen. Für die Summe der Quadrate kann wieder die Funktion `norm()` eingesetzt werden.

Beispiel 3.2

In dieser Aufgabe können Sie Ausführzeit und Genauigkeit der direkten Implementierung der DFT mit den Werten bei der FFT vergleichen. Entwerfen Sie dazu eine Funktion `dft_compare(N)` die sinusförmige Testsignale `x = sin(2*pi/N*M*[0:N-1])` und Zufallssignale `x = randn(N,1)` verwendet und damit die DFT und IDFT berechnet. Als Fehler dient dabei die maximale Betragsabweichung zwischen Eingangssignal und dem Ergebnis der kombinierten

Anwendung DFT + IDFT.
Hinweis: Für die Zeitmessung kann die Funktion `etime()` eingesetzt werden, über die Sie mit `help etime` mehr erfahren.

Beispiel 3.3

Interpolation im Frequenzbereich:

$$X\left(e^{j\theta}\right) = \sum_{n=0}^{N-1} x(n) e^{-j\theta n}$$
$$= \sum_{k=0}^{N-1} X(k) \underbrace{\frac{1}{N} \sum_{n=0}^{N-1} e^{-j\left(\theta - \frac{2\pi}{N}k\right)n}}_{G_N\left(e^{j\left(\theta-\frac{2\pi}{N}k\right)}\right)} = \sum_{k=0}^{N-1} X(k)\, G_N\left(e^{j\left(\theta-\frac{2\pi}{N}k\right)}\right).$$

Stellen Sie die Interpolationsfunktion $G_N\left(e^{j\theta}\right)$ für verschiedene Werte von N nach Betrag und Phase graphisch mit MATLAB® dar. Welches Zeitsignal $g_N(n)$ gehört als inverse Fouriertransformation zu $G_N\left(e^{j\theta}\right)$?
Bestimmen Sie die DFTs der Länge $L = 32$ von den folgenden Signalen der Länge $N = 10$, einmal mit der Interpolationsformel und zum Vergleich durch Anfügen von Nullen.

a) $x(n) = \delta(n), \quad n = 0, 1, \ldots, 9$

b) $x(n) = \delta(n-1), \quad n = 0, 1, \ldots, 9$

c) $x(n) = e^{-0{,}1n^2}, \quad n = 0, 1, \ldots, 9$

Beispiel 3.4

Mit MATLAB® kann eine dezimierte DFT z.B. in folgender Weise realisiert werden:

```
x = x(:);                  % signal stored as column vector
Nx = length(x);
M = ceil(Nx/L);            % L = number of frequency points
N = M*L;                   % N is a multiple of L
x = [x ; zeros(N-Nx,1)];   % append zeros, if necessary
```

```
Xb = zeros(L,M);          % matrix for storage of M signal blocks
                          % of length L
Xb(:) = x;                % columns of Xb are subsequent signal
                          % blocks of length L
xb = sum(Xb.');           % compute row sums of Xb
X = fft(xb);              % decimated FFT of length L
```

Untersuchen Sie anhand verschiedener Signale (z.B. kurze Sequenzen abgetasteter Sprachsignale) wie die Dezimation im Frequenzbereich das Erscheinungsbild des Signalspektrums beeinflusst.
Hinweis: Ein Audiosignal, als Datei im WAV-Format (z.B. `voice.wav`) gespeichert, kann mit `x = wavread('voice.wav')` als Vektor `x` in die Arbeitsumgebung von MATLAB® geladen werden. Im Falle eines Stereosignals ist `x` eine $N_x \times 2$ Matrix.

Beispiel 3.5

Es ist das Spektrum eines Signals zu untersuchen, das aus der Überlagerung zweier Sinusschwingungen besteht:

$$x(n) = \sin \theta_1 n + \sin \theta_2 n.$$

Beobachten Sie das Spektrum von $x_w(n) = w(n)x(n)$ in Abhängigkeit von der Differenzfrequenz $\Delta\theta = \theta_2 - \theta_1$ für verschiedene Fensterfunktionen $w(n)$. Achten Sie insbesondere auf die Beeinflussung der Größe und Form des spektralen Hauptbeitrags durch die Nebenbeiträge der anderen Frequenzkomponente (Leakage). Welche Fensterfunktion liefert die beste Trennung der Spektralkomponenten? Welche ist am besten zur Messung der spektralen Amplituden geeignet?

Beispiel 3.6

Entwerfen Sie eine MATLAB®-Funktion `[yl,yc] = convolve(x,h)` zur Berechnung der normalen (linearen) und der zyklischen Faltung mit der DFT für Signale unterschiedlicher Längen. Testen und vergleichen Sie die Ergebnisse beider Faltungsoperationen mit folgenden Signalen:

a)

$$x(n) = 1,\ n = 0, \ldots, 9 \quad h(n) = 1,\ n = 0, \ldots, 24$$

b)

$$x(n) = \begin{cases} 1 & 0 \le n \le 9 \\ 0 & 10 \le n \le 15 \end{cases} \qquad h(n) = \begin{cases} 1 & 0 \le n \le 24 \\ 0 & 25 \le n \le 31 \end{cases}$$

c)

$$x(n) = \begin{cases} 1 & 0 \le n \le 9 \\ 0 & 10 \le n \le 15 \end{cases} \qquad h(n) = \delta(n) - \delta(n-10),\ 0 \le n \le 15$$

Versuchen Sie in allen Fällen die Unterschiede zwischen normaler und zyklischer Faltung zu erklären.

Beispiel 3.7

Das MATLAB®-Beispiel zeigt die Implementierung der Overlap-Save Methode:

```
x = x(:);
h = h(:);
Nx = length(x);
Nh = length(h);
M = L - Nh + 1;                    % block shift (L = FFT length)
Nx1 = M*ceil(Nx/M);                % make length of x a multiple of M
x = [x ; zeros(Nx1-Nx,1)];         % pad with zeros
H = fft(h,L);                      % filter transfer function
x = [zeros(Nh-1,1) ; x];           % add leading zeros to input signal
Nx1 = Nx + Nh - 1;                 % new input signal length
y = zeros(Nx1,1);                  % init. output signal vector

for n = 1:M:Nx1-L+1                % block processing loop
   y1 = ifft(fft(x(n:n+L-1),L).*H);  % FFT - multiply by H - IFFT
   y(n:n+M-1) = y1(Nh:L);          % discard first Nh samples
end

y = y(1:Nx);
if ~any(imag(x)) & ~any(imag(h))   % real-valued signals x, h ?
   y = real(y);                    % avoid tiny imaginary parts
end
```

Entwickeln Sie aufbauend auf dem Musterbeispiel für die Overlap-Save Methode ein MATLAB®-Programm `y = ovadd(h,x)` für die Overlap-Add Methode. Die Filterimpulsantwort `h` eines FIR-Tiefpassfilters der Länge `Nh = 64` können Sie z.B. mit `h = fir1(64,1/8)` entwerfen. Vergleichen Sie für verschiedene

Eingangssignale das Ausgangssignal Ihres Programms mit jenem der Filterfunktion `y = filter(h,1,x)`. Messen Sie die Rechenzeit des Overlap-Add Programms mit `etime()` für verschiedene FFT-Längen `L`.

Beispiel 3.8

Vergleichen Sie mit MATLAB® die Ausführzeiten τ_A der Chirp-DFT mit jenen der FFT, wenn beide für die Spektralanalyse in den folgenden Frequenzintervallen $[f_1, f_2]$ und mit Frequenzauflösungen Δf eingesetzt werden sollen (die Abtastfrequenz ist in allen Fällen 16 kHz):

f_1 (Hz)	f_2 (Hz)	Δf (Hz)	τ_A Chirp-DFT	τ_A FFT
0	8000	100		
0	8000	10		
0	8000	1		
1000	2000	100		
1000	2000	10		
1000	2000	1		

4 Entwurf digitaler Filter

Beispiel 4.1

Gegeben ist das folgende, einfache Programm für einen FIR-Tiefpassfilterentwurf:

```
N1 = 40;                    % filter length minus 1
n = 1:N1/2;
fc = 1/10;                  % cutoff frequency normalized to Fs/2
h = sin(pi*fc*n)./(pi*n);   % one-sided discrete-time sinc-function
h = [h(N1/2:-1:1), fc, h];  % include symmetric part and midpoint
plot(abs(fft(h)));
zplane(h,1);                % plot poles and zeros
```

Modifizieren Sie die obigen MATLAB®-Befehle, um alle vier Fälle von FIR-Filtern graphisch zu veranschaulichen. Achten Sie insbesondere auf die Unterschiede zwischen gerader und ungerader Filterlänge. Stimmen die Frequenzgänge und die Lage der Pole und Nullstellen mit Ihren Vorstellungen überein? Um die Untersuchung flexibel zu gestalten, schreiben Sie am besten eine M-Funktion der Art `h = firsym(N,fc,even_odd)`, wobei `N` die Filterlänge, `fc` die Grenzfrequenz normiert auf die halbe Abtastfrequenz ist und `even_odd` zur Unterscheidung zwischen gerader (z.B. `even_odd = 'e'`) und ungerader Symmetrie von $h(n)$ dient. Der Prototyp der M-Funktion könnte dann so aussehen:

```
function h = firsym(N,fc,even_odd)
N2 = fix(N/2);
n = 1:N2;
if rem(N,2) == 1
   h = sin(pi*fc*n)./(pi*n);
   if strcmp(upper(even_odd),'E')
      h = ...
   else
      h = ...
   end
else
   ...
end
```

Beispiel 4.2

Gegeben ist folgendes Programm für die Frequenzabtastungsmethode:

```
function [h,Hid] = firfs(N,fc)

N2 = fix((N-1)/2);
Np = fix(fc*N/2);
Hmag = zeros(1,N);
Hmag(1:Np+1) = ones(1,Np+1);      % magnitude response from 0 to pi
                                  % (ideal low pass filter)
Hmag(N-N2+1:N) = Hmag(N2+1:-1:2); % magnitude resp. from pi to 2*pi,
                                  % using (even) symmetry property
% create linear phase response

phi = zeros(1,N);
phi(1:N2+1) = -pi*(N-1)*(0:N2)/N; % phase response from 0 to pi
```

```
phi(N-N2+1:N) = -phi(N2+1:-1:2);  % phase response from pi to 2*pi,
                                  % using (odd) symmetry property

% compute ideal frequency response and FIR filter impulse response

Hid = Hmag .* exp(j*phi);
h = real(ifft(Hid));        % real() avoids a very small imaginary
                            % part caused by rounding errors
```

Die Approximation steiler Frequenzgänge ist nicht optimal mit der Frequenzabtastungsmethode zu erreichen. Mit `firfs()` werden Sie feststellen, dass die Approximation des Rechtecktiefpassfilters nur eine geringe Sperrdämpfung ergibt. Eine Idee zur Verbesserung liegt nun darin, dass im Referenzbetragsverlauf eine Übergangszone zwischen Durchlass- und Sperrbereich vorgesehen wird. Dadurch entsteht ein rampenförmiger Betragsverlauf, der besser approximiert werden kann.
Modifizieren Sie `firfs()`, so dass im gewünschten Betragsverlauf ein bis drei Frequenzpunkte für den Übergangsbereich vorsehen werden, wo der Betragsverlauf Werte zwischen 0 und 1 annimmt. Optimieren Sie experimentell diese Zwischenwerte in bezug auf eine maximale Sperrdämpfung.

Beispiel 4.3

Schreiben Sie eine MATLAB®-Funktion `h_min = lin2min(h_lin)`, die ausgehend von der Impulsantwort `h_lin` eines linearphasigen Filters zunächst mit `roots()` die Nullstellen der Übertragungsfunktion berechnet, dann die außerhalb des Einheitskreises liegenden Nullstellen mit `find()` findet und diese durch die invertierten Nullstellen ersetzt. Abschließend wird mit `poly()` die Übertragungsfunktion des neuen (minimalphasigen) Filters gebildet und die Filterkoeffizienten bestimmt. Korrigieren Sie auch die Änderung der Filterverstärkung bedingt durch die Nullstelleninversion.
Vergleichen Sie graphisch die Frequenzgänge des so gewonnenen minimalphasigen FIR-Filters mit dem ursprünglichen Filter. Gibt es Unterschiede im Betragsverlauf? Beobachten Sie die Änderung der Impulsantwort. Bis zu welcher Filterlänge funktioniert diese Methode?

Beispiel 4.4

Das MATLAB®-Beispiel zeigt den FIR-Filterentwurf mit der Methode der kleinsten Quadrate (LS-Entwurf).

```
L = length(f);                          % number of frequency points
if rem(N,2) == 0                        % even filter length N ?
   M = N/2;
   S = 2*cos(pi*f(:)*(M - 0.5 - [0:M-1])); % system matrix
else
   M = (N-1)/2;
   S = [2*cos(pi*f(:)*(M - [0:M-1])) , ones(L,1)];
end
h = S \ m(:);                           % solve LS-problem
h = [h ; h(M:-1:1)];                    % append symmetric part
```

Modifizieren Sie das MATLAB®-Beispiel zum LS-Entwurf von FIR-Filtern mit ungerader symmetrischer Impulsantwort. Entwerfen Sie danach einen sogenannten Hilberttransformator mit dem gewünschten Frequenzgang

$$H\left(e^{j\theta}\right) = e^{-j\theta\frac{N-1}{2}} e^{-j\frac{\pi}{2}} \operatorname{sign}\theta$$

(sign x ist die Vorzeichenfunktion).
Welche Abweichungen vom idealen Betragsverlauf sind zu beobachten? Ist der Hilberttransformator gerader oder ungerader Länge besser? Wie sieht die Lage der Pole und Nullstellen in der komplexen z-Ebene aus? Können Sie einen minimalphasigen Hilberttransformator entwerfen?

Beispiel 4.5

Verwenden Sie die vorgestellte LS-Methode zum Entwurf

a) eines FIR-Filters mit gaußförmigem Betragsfrequenzgang,

b) eines "Differenzierers" (Betragsverlauf proportional zur Frequenz),

c) eines FIR-Filters zur Formung von weißem Rauschen zu $1/f$-Rauschen (Leistungsdichtespektrum verkehrt proportional zur Frequenz) mit einer vorgegebenen unteren Grenzfrequenz f_c,

d) eines FIR-Filters zur Approximation des Frequenzgangs der mittleren Hörschwelle des menschlichen Gehörs, deren Betragsverlauf in Dezibel durch die Funktion

$$T_q(f) = 3.64\left(\frac{f}{1000}\right)^{-0.8} - 6.5\, e^{-0.6\left(\frac{f}{1000}-3.3\right)^2} + 0.001\left(\frac{f}{1000}\right)^4$$

angenähert werden kann (Frequenz f in Hertz).

Beispiel 4.6

Entwickeln Sie eine MATLAB®-Funktion `h = firkw(fc,fr,ar)` zum Entwurf eines FIR-Tiefpassfilters mit dem Kaiser-Fenster. Die Argumente von `firkw()` sind die Grenzfrequenz `fc`, die Sperrfrequenz `fr` (beide normiert auf die halbe Abtastfrequenz) und die Sperrdämpfung `ar`. Beachten Sie dabei, dass bei der Anwendung des Kaiser-Fensters auf die Impulsantwort eines idealen Tiefpassfilters bei der Grenzfrequenz f_c eine Dämpfung von 3 dB auftritt. Durch die Modifikation der gegebenen Grenzfrequenz gemäß $f_c \rightarrow f_c + (f_r - f_c)/2$ kann dieser Effekt jedoch näherungsweise korrigiert werden.
Sie können Ihre MATLAB®-Funktion auch allgemeiner gestalten, indem Sie die optionale Übergabe der Filterlänge `N` zulassen, d.h. wenn `N` fehlt, wird die Filterlänge in der Funktion geschätzt. Die Funktionsdeklaration sollte dann `h = firkw(fc,fr,ar,N)` sein. Die Anzahl der Funktionsargumente kann mit `nargin` festgestellt werden.

Beispiel 4.7

Erweitern Sie Ihre MATLAB®-Funktion `firkw()` für verschiedene Filtertypen, indem Sie ein weiteres Argument `typ` vorsehen, das die einzelnen Filtertypen unterscheidet (`typ = 'lp'` oder `'hp'` oder `'bp'` oder `'bs'` oder `'di'` oder `'hi'`). Welche Einschränkung ergibt sich hinsichtlich der Filterlänge?

Beispiel 4.8

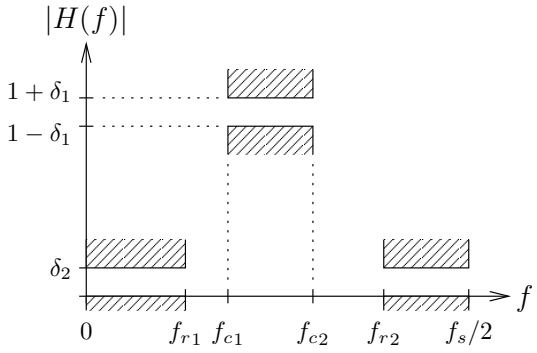
Entwickeln Sie die Funktion `h = firrem(fc,fr,ar,rp,N)` für das Equi-Ripple Design eines FIR-Tiefpassfilters mit den Eckfrequenzen `fc` und `fr`, der Sperrdämpfung `ar` und der Welligkeit im Durchlassbereich `rp` (beide Angaben in dB). Die Filterlänge `N` ist optional, d.h. wird sie nicht angegeben, dann wird `N` geschätzt. Verwenden Sie die Gewichtungsmöglichkeit (Argument `w` bei der Funktion `remez()` bzw. `firpm()`), um die Approximation im Sperrbereich mit dem Faktor δ_1/δ_2 zu bevorzugen. Vergleichen Sie die Entwurfsergebnisse mit jenen der Kaiserfenstermethode (`firkw()`).

Beispiel 4.9

In dieser Aufgabe werden Sie Bandpassfilter mit dem Remez-Algorithmus entwerfen und dabei auch ein Problem dieses Verfahrens kennenlernen. Zunächst ist ein symmetrisches Bandpassfilter mit folgenden Daten zu entwerfen:

$$f_{r1} = 0.5, f_{c1} = 0.6, f_{c2} = 0.8, f_{r2} = 0.9, f_s = 10, a_r = 60, r_p = 0.2$$

(Alle Frequenzangaben in kHz, Sperrdämpfung und Welligkeit im Durchlassbereich in dB, $a_r = -20\log_{10}\delta_2$ und $r_p = 20\log_{10}\frac{1+\delta_1}{1-\delta_1}$.)



Verändern Sie anschließend die obere Sperrfrequenz auf $f_{r2} = 1$ kHz und beobachten Sie den jeweiligen Frequenzgang. Versuchen Sie das Ergebnis zu interpretieren!

Beispiel 4.10

Gegeben ist folgendes Beispiel eines Low-Delay FIR Filter Designs:

```
N = 101;                    % filter length
Ndoff = 20;                 % delay should be smaller by Ndoff samples
                            % than for linear phase filter
Nd = (N-1)/2 - Ndoff;       % desired filter delay
Lpass = 10*N;               % number of frequency point in pass band
Lstop = 10*N;               % number of frequency point in stop band
fc = 2/8;                   % cutoff frequency normalized to Fs/2
fr = 2.5/8;                 % stop band edge normalized to Fs/2
Wpass = 1;                  % pass band weighting factor
Wstop = 50;                 % stop band weighting factor

theta = pi*[linspace(0,fc,Lpass), linspace(fr,1,Lstop)];
W = [Wpass*ones(1,Lpass), Wstop*ones(1,Lstop)];
```

```
D = [exp(-j*theta(1:Lpass)*Nd) zeros(1,Lstop)];

h = lslevin(N,theta,D,W);
```

(Die Funktion `lslevin()` ist in der Programmsammlung zum Buch enthalten). Variieren Sie im vorigen Beispiel die Parameter `Ndoff` im Bereich $[-30, 30]$ und beobachten Sie den Betrags- und Phasenverlauf des FIR-Filters. Wie groß sind in Abhängigkeit von `Ndoff` die maximalen Fehler für Betrag (in dB) und Gruppenlaufzeit (in Samples)? Vergleichen Sie den Betragsverlauf des Low-Delay FIR-Filters (`Ndoff = 20`) mit einem exakt linearphasigen Filter, dessen Betragsverlauf die gleichen Spezifikationen erfüllt.

Beispiel 4.11

Verwenden Sie die Funktion `lslevin()` aus der der Programmsammlung zum Buch für den Entwurf von FIR-Allpassfiltern mit einem gewünschten Frequenzgang $D\left(e^{j\theta}\right) = e^{-j\phi(\theta)}$ und

a) für eine sinusförmige Gruppenlaufzeit, d.h. der gewünschte Phasenverlauf ist

$$\phi(\theta) = \frac{N-1}{2}\theta + 2\pi(1 - \cos\theta)$$

mit $N = 61$.

b) für ein Chirp-Allpassfilter mit dem Phasenverlauf

$$\phi(\theta) = \frac{N-1}{2}\,\theta - \eta\left(\theta - \frac{\pi}{2}\right)^2$$

und $N = 61$, $N = 101$ bzw. $\eta = \frac{8}{2\pi}$, $\eta = \frac{16}{2\pi}$.

Welche Anwendungen können Sie sich für diese Filter vorstellen?

Beispiel 4.12

Verwenden Sie `lerner_lp()` aus der der Programmsammlung zum Buch zum Entwurf von digitalen Lernerfiltern und stellen Sie deren Übertragungsfunktionen nach Betrag und Phase

$$H(j\theta) = A(\theta)e^{j\phi(\theta)},$$

sowie die Gruppenlaufzeit $\tau_g(\theta) = -\frac{d\phi(\theta)}{d\theta}$ graphisch dar. Vergleichen Sie Lerner-Tiefpassfilter mit FIR-Filtern annähernd gleicher Filtersteilheit. Wie groß sind die Unterschiede im Implementierungsaufwand (Anzahl der Multiplikationen, Additionen und Verzögerungselemente)? Ist die Signalverzögerung beim Lerner- oder beim FIR-Filter geringer?

Beispiel 4.13

Erweitern Sie die Funktion `lerner_lp()` aus der der Programmsammlung zum Buch für den Entwurf digitaler Bandpassfilter. Dabei werden beim analogen Lerner-Bandpassfilter die Polstellen einfach in die Bandpasslage verschoben. Beachten Sie die korrekte Wahl der Vorzeichen der Residuen bei den konjugiert komplexen Polen. Auch für diese Aufgabe lohnt sich der Vergleich mit linearphasigen FIR-Filtern.

Beispiel 4.14

Verwenden Sie die Funktion MATLAB®-Funktion `prony()` zum Entwurf rekursiver Filter

a) mit dreieckförmiger Impulsantwort,

b) mit gaußförmiger Impulsantwort,

c) zum Nachbilden der Impulsantwort eines Hilberttransformators,

d) zum Nachbilden der Impulsantwort eines Differenzierers.

Variieren Sie in allen Fällen die Parameter N und M und sehen Sie sich mit `zplane()` die Pol/Nullstellenkonstellation an.

Beispiel 4.15

In dieser Aufgabe sollen Sie die vier Standardapproximationen für digitale IIR-Tiefpassfilter vergleichen. Dazu bestimmen Sie jeweils Pole, Nullstellen und den Verstärkungsfaktor für die folgende Spezifikation:

$$f_c = 0.800 \text{ kHz},\ f_r = 1.2 \text{ kHz},\ f_s = 10 \text{ kHz},\ r_p = 0.5 \text{ dB},\ a_r = 60 \text{ dB}.$$

Passen Sie die jeweilige Filterordnung so an, dass Filter gleicher Steilheit und Welligkeiten entstehen. Betrachten Sie den Frequenzgang von Dämpfung und Gruppenlaufzeit. Welches Filter hat die größte Verzerrung der Gruppenlaufzeit? Wie sieht die Lage der Pole und Nullstellen mit der Funktion `zplane()` aus?

Beispiel 4.16

Von Interesse ist auch ein Vergleich der Impulsantworten der einzelnen IIR-Filter. Dazu gehen Sie wie folgt vor:

a) Bestimmen Sie Pole `p`, Nullstellen `z` und Verstärkungsfaktor `k` des jeweiligen IIR-Filters.

b) Transformieren Sie Pole und Nullstellen der Übertragungsfunktion mit `b = k*real(poly(z))` und `a = real(poly(p))` in die Darstellung als rationale Funktion mit dem Koeffizientenvektor `b` des Zählerpolynoms bzw. `a` des Nennerpolynoms (`real()` eliminiert einen kleinen Imaginärteil, der durch Rundungsfehler entstehen kann).

c) Berechnen Sie mit `h = filter(b,a,[1 zeros(1,Ntime)])` wie gewohnt die Impulsantwort.

Welches der untersuchten Filter ist für die Bandbegrenzung bei Datenübertragungskanälen am besten geeignet? Welche Filter sind streng minimalphasig? Versuchen Sie, mit dem LS-Verfahren, aus der Impulsantwort die Polynom- bzw. Pol/Nullstellendarstellung der Übertragungsfunktion zu bestimmen. Gibt es Abweichungen zur ursprünglichen Darstellung? Ab welcher Filterordnung versagt diese Methode? Versuchen Sie, die Fehlerquellen zu lokalisieren.

Beispiel 4.17

Entwerfen Sie mit $H_{hp}(z) = H_{lp}(-z)$ aus einem beliebigen Tiefpassfilter ein Hochpassfilter und wenden Sie dann die einfache Transformation auf die Pol/-Nullstellendarstellung von $H(z)$ an. Beobachten Sie auch die Änderung der Pole und Nullstellen mit `zplane()`.

Beispiel 4.18

Die Transformation $z \to \pm z^2$ kann z.B. dadurch verallgemeinert werden, dass z in $H(z)$ durch $\pm z^L$ (L ganzzahlig) ersetzt wird. Überlegen Sie sich mit den Eigenschaften der Fouriertransformation den Einfluss einer solchen Transformation, d.h. $e^{j\theta} \to e^{j\theta L}$ auf den Frequenzgang $H(e^{j\theta})$. Entwerfen Sie danach ein schmalbandiges Bandpassfilter (oder Tiefpassfilter) und kontrollieren Sie Ihre theoretischen Resultate mit MATLAB®.

Beispiel 4.19

Es soll ein IIR-Filter entworfen werden, dessen Dämpfungsverlauf dem Frequenzgang der mittleren Hörschwelle eines gesunden menschlichen Gehörs entspricht. Eine Näherungsformel für diesen Verlauf ist bereits auf Seite 18 in Aufgabe 4.5 angegeben. Bestimmen Sie mit `yulewalk()` ein IIR-Filter, das den Frequenzverlauf der Hörschwelle nachbildet. Entwerfen Sie dann das inverse Filter zur Kompensation dieses Frequenzgangs und testen Sie Ihren Entwurf, indem Sie beide Filter in Kette schalten. Vergleichen Sie das Ergebnis mit jenem des FIR-Filters als Lösung von Aufgabe 4.5.

5 Multiratenfilterbänke und Wavelets

Beispiel 5.1

Es ist mit `[h,g] = unicmfb(8,256,1/32,1e5,256)` aus der Programmsammlung zum Buch eine Filterbank zu entwerfen. Schreiben Sie eine MATLAB®-Funktion `[y,Y] = unifib(h,g,x)`, die mit dem Entwurfsergebnis die komplette, kritisch abgetastete Filterbank für verschiedene Eingangssignale berechnet. Das Ausgangssignal wird im Vektor `y` gespeichert. Die Matrix `Y` enthält die Teilbandsignale als Spaltenvektoren. Als Eingangssignale verwenden Sie zeitverschobene Einsimpulse oder ein Zufallssignal `x = randn(Nx,1)`.

Beispiel 5.2

Verwenden Sie die Funktionen `unicmfb()` und `nuftrans()` aus der Programmsammlung zum Buch für das Design einer nichtäquidistanten Filterbank. Dazu entwerfen Sie zunächst zwei äquidistante Filterbänke mit $N_1 = 8$ und $N_2 = 4$ Kanälen. Die FIR-Filterlängen sind $L_1 = 96$ (8-Kanalfilterbank) und $L_2 = 40$ (4-Kanalfilterbank). Dann wird mit `nuftrans()` der Tiefpassprototyp des Übergangsfilter ($L_t = 96$) entworfen. Das Bandpassübergangsfilter erhalten Sie dann durch eine cosinusförmige Modulation des Prototyps.

Beispiel 5.3

Verwenden Sie `fft_fib()` aus der Programmsammlung zum Buch für das Testen der Analyse/Synthese-Filterbank mit Sprachsignalen. Dazu erweitern Sie diese MATLAB®-Funktion für Blockverarbeitung, Achten Sie darauf, dass beim Aneinanderfügen der Blöcke keine Stoßstellen auftreten! Die Audiosignale können beispielsweise als WAV-Dateien (oder in binärer Form als RAW-Datei) gespeichert sein. Variieren Sie die Blocklänge N und den Dezimationsfaktor M. Für welche Werte M sind Abweichungen des rekonstruierten Signals vom ursprünglichen Signal deutlich wahrnehmbar?

Beispiel 5.4

Es ist ein Equalizer für Audiosignale zu entwerfen, mit dem Verzerrungen des Frequenzgangs (z.B. von Lautsprechern) kompensiert werden können. Eine andere Einsatzmöglichkeit wäre die Realisierung von Klangbewertungsfiltern (oder Klangreglern). Verwenden Sie Ihr Filterbankprogramm aus dem vorhergehenden Beispiel, in dem Sie in der Teilbandverarbeitung den Betrag der FFT-Komponenten mit einem reellwertigen Faktor $0 \leq g_k \leq 1,\ k = 1, 2, \ldots, N$ modifizieren. Beachten Sie dabei die Symmetriebedingungen der FFT für reellwertige Signale.

Beispiel 5.5

Entwickeln Sie ein MATLAB®-Programm `wtfib(N,h_l,h_h)`, mit dem ausgehend von den Impulsantworten der Zweikanalfilter $h_L(n)$ und $h_H(n)$ die Frequenzgänge der Teilbandfilter einer DTWT der Länge N dargestellt werden können.

Beispiel 5.6

Implementierung der Zweikanalpolyphasenfilterbank:

```
hg = h(1:2:end);          % polyphase filters of low pass prototyp
hu = h(2:2:end);
x = x(:);                 % input signal

% analysis filter bank

u1 = filter(hg,1,x(1:2:end));
u2 = filter(hu,1,[0 ; x(2:2:end-1)]);
y1 = u1 + u2;             % subband signals y1, y2
y2 = u1 - u2;

% synthesis filter bank

v1 = y1 - y2;
v2 = y1 + y2;
y = zeros(size(x));
y(1:2:end) = filter(hg,1,v1);
y(2:2:end) = filter(hu,1,v2);
```

Kontrollieren Sie zunächst, ob das vorgeschlagene Programm für die Zweikanalfilterbank bei geraden und ungeraden Signallängen richtig arbeitet. Berechnen Sie dann mit dem Programm `unicmbf()` der Programmsammlung zum Buch die Prototypimpulsantwort $h(n)$ für eine Zweikanalfilterbank (z.B. mit `h = unicmfb(2,40,1/6,1e5,80);`). Testen Sie danach mit verschiedenen Eingangssignalen das Verhalten der Zweikanalpolyphasenfilterbank und bestimmen Sie den Rekonstruktionsfehler. Wie groß ist die Zeitverzögerung der Filterbank zwischen Eingang und Ausgang?

Beispiel 5.7

Gegeben ist folgende MATLAB®-Funktion für die DTWT/IDTWT:

```
function y = DTWT_IDTWT(N,h,x)
% function y = DTWT_IDTWT(N,h,x)
% compute DTWT and IDWT to show reconstruction property
%
% N    number of DTWT coefficients = number of filter bank channels
% h    low pass filter impulse response (linear phase filter)
% x,y  input, and output signal vectors
%
% Note: do not use unicmfb() to design filter because linear phase
%       filters are required, use e.g. h = qmf_2(50,0.65,0.1)
%       (qmf_2() can be obtained from the author's homepage)
%
% G. Doblinger, TU-Wien, 02-2001

Nx2 = ceil(length(x)/2);   % maximum length of DTWT coefficients
L = length(h);
p1 = h(1:2:end);           % polyphase filters of low pass prototyp
p2 = h(2:2:end);

DTWT = zeros(Nx2+L,N);     % matrix of N DTWT coefficients
yl = x;
for n = 1:N-1
   [yl,yh] = polyafib2(p1,p2,yl);
   DTWT(1:length(yh),n) = yh;
end
DTWT(1:length(yl),N) = yl;

% compute IDTWT = output y of synthesis filter bank

y = DTWT(1:length(yl),N);
for n = N-1:-1:1
   y = polysfib2(p1,p2,y,DTWT(1:length(y),n));
end

% ------------------------------------------------------

function [yl,yh] = polyafib2(p1,p2,x)

% compute analysis filter bank (polyphase filters)

u1 = conv(p1,x(1:2:end));
u2 = conv(p2,[0 ; x(2:2:end-1)]);
yl = u1 + u2;
```

```
yh = u1 - u2;

% ------------------------------------------------------

function y = polysfib2(p1,p2,yl,yh)

% compute synthesis filter bank (polyphase filters)

L = 2*length(p1);
Ly = 2*length(yl)+L-2;
v1 = yl - yh;
v2 = yl + yh;
y = zeros(Ly,1);
y(1:2:Ly) = conv(p1,v1);
y(2:2:Ly) = conv(p2,v2);
y = y(L:Ly+3-L);   % compensate delay of 2 channel filter bank
```

Verwenden Sie den angegeben MATLAB®-Code zur Bestimmung der DTWT folgender Signale:

a) Zeitdiskreter, verzögerter Rechteckimpuls. Wie ändert sich die DTWT mit der Verzögerungszeit des Rechteckimpulses?

b) Sinusförmiges Chirp-Signal $x(n) = \sin(\alpha n^2)$.

c) Modulierter Gaußimpuls $x(n) = e^{-\alpha n^2} \cos(\theta_0 n)$.

Als Prototypfilterimpulsantwort `h` verwenden Sie jene aus der vorhergehenden Aufgabe. Überprüfen Sie die Rekonstruktionseigenschaften der IDTWT, indem Sie (wie im angegebenen Programm) die DTWT und IDTWT zusammenschalten.

Beispiel 5.8

Wenn Sie sich selbst ein Bild von der Leistungsfähigkeit des Wavelet-Schrumpfens machen wollen, dann sind Experimente mit der Funktion `wt_dn()` (siehe Programmsammlung zum Buch) und verschiedenen Signalen und Störpegeln empfehlenswert. Versuchen Sie die folgenden, mit weißem Rauschen überlagerten Signale zu entstören. Variieren Sie dabei die Kanalanzahl N und die Leistung des Störsignals.

a) Vokalische Sprachlaute oder eine Singstimme.

b) Signale mit Sprungstellen.

c) Tiefpass- oder bandpassgefiltertes weißes Rauschen.

d) Random Walk Signal:

$$x(n) = x(n-1) + r(n)\sigma(n), \quad n = 0, 1, 2, \ldots$$

($x(-1) = 0$, $r(n)$ ist ein mittelwertfreies, gaußsches Rauschen, $\sigma(n)$ ist die zeitdiskrete Sprungfunktion).

6 Anwendungen aus dem Bereich der Audiotechnik

Beispiel 6.1

Die vorgestellten Algorithmen zur Pitch- und Zeitskalierung (`pitch_scale()` und `time_scale()` der Programmsammlung zum Buch) bieten ein weites Feld für Experimente und Verbesserungen. Bei den Tests mit Sprachsignalen werden Sie feststellen, dass neben der Zeit- oder Frequenzskalierung auch ein geringer Nachhall zu hören ist. Bei Musiksignalen fällt dieser Effekt nicht so stark auf. Der Effekt wird deutlicher, wenn die Länge des Zeitfensters erhöht wird. Um die Ursachen dieser Beobachtung aufzuspüren, bestimmen Sie die Impulsantworten der Systeme. Da die Systeme zeitvariant sind, müssen dabei die Antworten auf $\delta(n-k)$-Impulsen mit verschiedenen Verzögerungszeiten k gemessen werden, um die **zweidimensionale Impulsantwort** $h(n,k)$ zu erhalten.
Testen Sie die Systeme mit Sinussignalen und prüfen Sie, ob die Amplituden der Ausgangssignale konstant sind oder eine Modulation aufweisen. Wiederholen Sie dieses Experiment mit Chirp-Signalen und versuchen Sie eine optimale Zeitfensterlänge zu bestimmen. Versuchen Sie die Ergebnisse der Experimente zu interpretieren.

Beispiel 6.2

Mit dem angegebenen Signalentstörungsverfahren (`enhance()` der Programmsammlung zum Buch)können eine Reihe von Experimenten ausgeführt werden. Speziell die richtige Wahl der Parameter hat einen starken Einfluss auf die Qualität des entstörten Signals. Die Parameterwahl sollte mit einer Auswahl einiger Musik- oder Sprachsignalproben vorgenommen werden. Abtastfrequenzen von 11.025 kHz, 16 kHz oder maximal 22.05 kHz sind gegenüber 44.1 kHz

zu bevorzugen, um die Signallängen (und damit die Rechenzeit) relativ kurz zu halten. Bei Musiksignalen wird man ein Zeitfenster im Bereich von 40 ms bis 60 ms wählen, während bei Sprachsignalen die Dauer des Zeitfensters um 20 ms liegt. Dementsprechend sind die Fensterlängen N an das Testmaterial anzupassen. Folgende Untersuchungen können als Anregung dienen:

a) Bei welchem SNR am Eingang des Systems verschlechtert sich das Systemverhalten drastisch?

b) Berechnen Sie mit der MATLAB®-Funktion `specgram()` die Spektrogramme des verrauschten Eingangssignals und des entstörten Signals. Beobachten Sie dabei die zeitvariante Filterfunktion des Systems.

c) Wie ist das Systemverhalten, wenn Sie ähnlich wie mit der Wavelet-Transformation ein verrauschtes Chirp-Signal verarbeiten?

Beispiel 6.3

In dieser Aufgabe sollen drei Experimente zur Bestimmung der Verformung des entstörten Signals durch den Rauschunterdrückungsalgorithmus durchgeführt werden. Diese Untersuchungen sind insbesondere bei der Restaurierung gestörter Musikaufnahmen von Bedeutung. Die Ergebnisse sollen in erster Linie der richtigen Wahl der Parameter N, α und ρ des Algorithmus dienen.

a) Verwenden Sie verrauschte Sinussignale mit konstanter Amplitude und Frequenzen von 400 Hz, 1 kHz und 3 kHz zur Bestimmung der jeweiligen Grenze, ab der das System diese Töne unterdrückt, obwohl sie im verrauschten Signal noch hörbar sind. Versuchen Sie eine optimale Wahl der Parameter zu finden und testen Sie das Ergebnis mit leiser Musik (z.B. zarte Geigen- oder langgezogene Flötentöne).

b) Mit diesem Experiment soll das Einschwingverhalten des Entstörverfahrens analysiert werden. Dazu werden eingeschaltete, verrauschte Sinussignale mit den Frequenzen aus der vorhergehenden Teilaufgabe eingesetzt. Messen Sie das Ein- und Ausschwingverhalten (Dauer) des Systems in Abhängigkeit von der Amplitude des Sinussignals und der Fensterdauer N. Testen Sie die Hörbarkeit der transienten Verzerrungen mit hart angeschlagenen Klavier- oder Gitarrenklängen.

c) Untersuchen Sie die Modulation von Sinussignalen mit dem im betreffenden Filterbankkanal liegenden Rauschen. Ab welchem SNR am Systemeingang wird eine Klangänderung durch das Restrauschen hörbar?