ADVANCED DIGITAL SIGNAL PROCESSING

Lab 9: Design of IIR Filters

Objective

The objective of this lab is to design IIR filters by approximating analog transfer function.

1 Introduction

IIR filters have infinite-duration impulse responses, hence they can be matched to analog filters, all of which generally have infinitely long impulse responses. Therefore the basic technique of IIR filter design transforms well-known analog filters into digital filters using complex-valued mappings. The advantage of this technique lies in the fact that both analog filter design (AFD) tables and the mappings are available extensively in the literature. This basic technique is called the A/D (analog-to-digital) filter transformation. However, the AFD tables are available only for lowpass filters. We also want to design other frequency-selective filters (highpass, bandpass, bandstop, etc.). To do this, we need to apply frequency-band transformations to lowpass filters. These transformations are also complex-valued mappings, and they are also available in the literature.

2 Bilinear Transformation

The most widely used approach to IIR filter design is based on the *bilinear* transformation from s-plane to z-plane given by,

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) \Rightarrow z = \frac{1+{}^{sT}/_{2}}{1-{}^{sT}/_{2}}$$

where T is a parameter known as *sampling period*. Using the above transformation an analog transfer function $H_a(s)$ is converted into a digital transfer function $H(z)$ according to,

$$H(z) = H_a(s)\big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

For the *bilinear* transformation, the relation between the imaginary axis ($s = j\Omega$) in the s-plane and the unit circle ($z = e^{j\omega}$) in the z-plane is given by,

$$\Omega = \tan(\omega/2)$$

which maps the entire imaginary axis in the s-plane to the unit circle in the z-plane introducing a distortion in the frequency axis called *warping*. To develop a digital filter meeting a specified magnitude response, the analog equivalents (Ω_p and Ω_s) of the critical band-edge frequencies (ω_p and ω_s) of the digital filter are first obtained using the relation of $\Omega = \tan(\omega/2)$, the analog prototype $H_a(s)$ is then designed using the prewarped critical frequencies, and $H_a(s)$ is transformed using the *bilinear* transformation to obtain the desired digital filter transfer function $H(z)$.

2.1 Example

Transform $H_a(s) = \dfrac{s+1}{s^2+5s+6}$ into a digital filter using the bilinear transformation. Choose $T = 1$.

$$H(z) = H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\bigg|_{T=1}\right) = H_a\left(2\frac{1-z^{-1}}{1+z^{-1}}\right)$$

Simplifying,

$$H(z) = \frac{3+2z^{-1}-z^{-2}}{20+4z^{-1}} = \frac{0.15+0.1z^{-1}-0.05z^{-2}}{1+0.2z^{-1}}$$

MATLAB provides a function called `bilinear` to implement this mapping.

```
>> num = [1,1]; den = [1,5,6];
>> T = 1; Fs = 1/T;
>> [b,a]=bilinear(num,den,Fs)
```

b = 0.1500 0.1000 -0.0500
a = 1.0000 0.2000 0.0000

For example, it is required to design a digital filter to approximate the following normalized analog transfer function.

$$H(s) = \frac{1}{s^2+\sqrt{2}s+1}$$

Use the bilinear transformation method to obtain $H(z)$. Assume 3 dB cut-off frequency of 150 Hz and sampling frequency of 1.28 kHz.

```
clc, clear all, close all
Fs = 1280; % sampling frequency (samples per
    second)
T = 1/Fs; % sampling time
fp = 150; % passband cut-off freq in Hz
Wp = 2*pi*fp; % digital passband cut-off freq
    in rads/sec
% apply pre-warping transformation
OmegaP = tan(Wp*T/2); % prewarp prototype
    passband freq
[num,den]=butter(2,1,'s'); % 2nd order
    normalized analogue butterworth filter with
     cut-off 1 rad/sec
printsys(num,den)
[num1,den1]=lp2lp(num,den,OmegaP); % low-pass
    to low-pass transformation
[numd,dend]=bilinear(num1,den1,0.5); % bilinear
     transformation
printsys(numd,dend,'z')
fvtool(numd,dend)
```

ADVANCED DIGITAL SIGNAL PROCESSING

Lab 9: Design of IIR Filters

MATLAB's function `fvtool` displays the magnitude response of the digital filter. You can also use it to visualize phase response, group delay, impulse response, step response, pole-zero plot, and coefficients of the filter.

Use the following code to design digital bandpass butterworth filter using bilinear transformation method.

```
clc, clear all, close all
Fs = 2000; % sampling frequency (samples per
    second)
T = 1/Fs; % sampling time
fp = 200; % passband cut-off freq in Hz
fs = 300; % stopband cut-off freq in Hz
Wp = 2*pi*fp; % digital passband cut-off freq
    in rads/sec
Ws = 2*pi*fs; % digital stopband cut-off freq
    in rads/sec
% apply pre-warping transformation
OmegaP = tan(Wp*T/2); % prewarp prototype
    passband freq
OmegaS = tan(Ws*T/2); % prewarp prototype
    stopband freq
W = (OmegaP+OmegaS)/2; % central frequency
BW = OmegaS - OmegaP; % bandwidth
[num,den]=butter(1,1,'s'); % 2nd order
    normalized analogue butterworth filter with
     cut-off 1 rad/sec
printsys(num,den)
[num1,den1]=lp2bp(num,den,W,BW); % low-pass to
    band-pass transformation
[numd,dend]=bilinear(num1,den1,0.5); % bilinear
     transformation
printsys(numd,dend,'z')
fvtool(numd,dend)
```

To have better insight, lets say we want to design a lowpass digital butterworth filter with the following specifications using bilinear transformation method,

$$\omega_p = 0.2\pi \quad R_p = 1\,dB$$
$$\omega_s = 0.3\pi \quad A_s = 15\,dB$$

```
clc, clear all, close all
% digital filter specifications
Wp = 0.2*pi; % digital passband freq in rad
Ws = 0.3*pi; % digital stopband freq in rad
Rp = 1; % passband ripple in dB
As = 15; % stopband attenuation in dB
% analog prototype specifications: inverse
    mapping for frequencies
T = 1; Fs = 1/T; % set T=1
OmegaP = (2/T)*tan(Wp/2); % prewarp prototype
    passband freq
OmegaS = (2/T)*tan(Ws/2); % prewarp prototype
    stopband freq
ep = sqrt(10^(Rp/10)-1); % passband ripple
    parameter
Ripple = sqrt(1/(1+ep*ep)); % passband ripple
Attn = 1/(10^(As/20)); % stopband attenuation
N = ceil((log10((10^(Rp/10)-1)/(10^(As/10)-1)))
    /(2*log10(Wp/Ws))); % order of the
    butterworth filter
fprintf('\n*** Butterworth Filter Order = %2.0f
     \n',N)
OmegaC = Wp/((10^(Rp/10)-1)^(1/(2*N))); %
    cutoff frequency in radians/sec
[z,p,k] = buttap(N); % butterworth analog
    lowpass filter prototype
p = p*OmegaC;
k = k*OmegaC^N;
B = real(poly(z));
b0 = k; cs = k*B; ds = real(poly(p));
[b,a] = bilinear(cs,ds,Fs); % bilinear
    transformation
% frequency response of digital filter
[H,w] = freqz(b,a,1000,'whole');
H = (H(1:1:501))'; w = (w(1:1:501))'; %  w =
    501 frequency samples between 0 to pi
    radians
mag = abs(H); % absolute magnitude computed
    over 0 to pi radians
db = 20*log10((mag+eps)/max(mag)); % Relative
    magnitude in dB computed over 0 to pi
    radians
pha = angle(H); % Phase response in radians
    over 0 to pi radians
grd = grpdelay(b,a,w); % Group delay over 0 to
    pi radians
thresh1 = -Rp*ones(1,length(mag)); % plot(w/pi,
    thresh1,w/pi,thresh2);
thresh2 = -As*ones(1,length(mag));
subplot(2,2,1);
plot(w/pi,mag);
title('Magnitude Response')
xlabel('frequency in \pi units'); ylabel('|H|')
axis([0,1,0,1.1])
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',[0,Attn,
    Ripple,1]); grid
subplot(2,2,2);
plot(w/pi,pha/pi);
title('Phase Response')
xlabel('frequency in \pi units'); ylabel('\pi
    units')
axis([0,1,-1,1]);
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',[-1,0,1]);
     grid
subplot(2,2,3);
plot(w/pi,db);
title('Magnitude Response in dB')
xlabel('frequency in \pi units'); ylabel('
    Decibels')
axis([0,1,-40,5]);
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick'
    ,[-50,-15,-1,0]); grid
subplot(2,2,4);
plot(w/pi,grd);
title('Group Delay')
xlabel('frequency in \pi units'); ylabel('
    Samples')
axis([0,1,0,11])
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',0:2:10);
    grid
```

The desired filter is a 6^{th}-order Butterworth filter

whose system function $H(z)$ is given in the parallel form,

$$H(z) = \frac{1.8587 - 0.6304z^{-1}}{1 - 0.9973z^{-1} + 0.257z^{-2}} + \frac{-2.1428 + 1.1454z^{-1}}{1 - 1.0691z^{-1} + 0.3699z^{-2}} + \frac{0.2871 - 0.4463z^{-1}}{1 - 1.2972z^{-1} + 0.6449z^{-2}}$$

You can use MATLAB's function `fdatool` to open the Filter Design and Analysis Tool (FDATool) to design and analyze filters.

```
≫ fdatool
```

Play around and try to verify the outcomes using this tool.

3 Impulse Invariance

If we want to preserve the shape of the impulse response from analog to digital filter, then we obtain a technique called impulse *invariance transformation*. In this design method we want the digital filter impulse response to look "similar" to that of a frequency-selective analog filter. Hence we sample $h_a(t)$ at some sampling interval T to obtain $h(n)$; that is,

$$h(n) = h_a(nT)$$

The parameter T is chosen so that the shape of $h_a(t)$ is "captured" by the samples. Since this is a sampling operation, the analog and digital frequencies are related by,

$$\omega = \Omega T \quad \text{or} \quad e^{j\omega} = e^{j\Omega T}$$

Since $z = e^{j\omega}$ on the unit circle and $s = j\Omega$ on the imaginary axis, we have the following transformation from the s-plane to the z-plane.

$$z = e^{sT}$$

The system functions $H(z)$ and $H_a(s)$ are related through the frequency-domain aliasing formula $\omega = \Omega T_s$, where $T_s = \frac{1}{Fs}$

$$H(z) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a\left(s - j\frac{2\pi}{T}k\right)$$

3.1 Design Procedure

Given the digital lowpass filter specifications ω_p, ω_s, R_p, and A_s, we want to determine $H(z)$ by first designing an equivalent analog filter and then mapping it into the desired digital filter. The steps required for this procedure are,

1. Choose T and determine the analog frequencies,

$$\Omega_p = \frac{\omega_p}{T_p} \quad \text{and} \quad \Omega_s = \frac{\omega_s}{T}$$

2. Design an analog filter $H_a(s)$ using the specifications Ω_p, Ω_s, R_p, and A_s. This can be done using any one of the three (Butterworth, Chebyshev, or elliptic) prototypes.

3. Using partial fraction expansion, expand $H_a(s)$ into,

$$H_a(s) = \sum_{k=1}^{N} \frac{R_k}{a - p_k}$$

4. Now transform analog poles $\{p_k\}$ into digital poles $\{e^{p_k T}\}$ to obtain the digital filter:

$$H(z) = \sum_{k=1}^{N} \frac{R_k}{1 - e^{p_k T}z^{-1}}$$

3.2 Example

Transform $H_a(s) = \dfrac{s+1}{s^2+5s+6}$ into a digital filter $H(z)$ using the impulse invariance technique in which $T = 0.1$.

We first expand $H_a(s)$ using partial fraction expansion,

$$H_a(s) = \frac{s+1}{s^2+5s+6} = \frac{2}{s+3} - \frac{1}{s+2}$$

The poles are at $p_1 = -3$ and $p_2 = -2$. Using $T = 0.1$ in the equation $H(z) = \sum_{k=1}^{N} \frac{R_k}{1-e^{p_k T}z^{-1}}$, we have,

$$H(z) = \frac{2}{1 - e^{-3T}z^{-1}} - \frac{1}{1 - e^{-2T}z^{-1}} = \frac{1 - 0.8966z^{-1}}{1 - 1.5595z^{-1} + 0.6065z^{-2}}$$

It is easy to develop a MATLAB function to implement the impulse invariance mapping. Given a rational function description of $H_a(s)$, we can use the `residue` function to obtain its pole-zero description. Then each analog pole is mapped into a digital pole using $z = e^{sT}$. Finally, the `residuez` function can be used to convert $H(z)$ into rational function form. A similar function called `impinvar` is available in the SP toolbox of MATLAB.

```
clc, clear all, close all
c = [1,1]; % numerator polynomial in s of the
    analog filter
d = [1,5,6]; % denominator polynomial in s of
    the analog filter
T = 0.1; % sampling (transformation) parameter
Fs = 1/T; % sampling frequency
[R,p,k] = residue(c,d); % partial-fraction
    expansion (residues)
```

Lab 9: Design of IIR Filters

```
p = exp(p*T);
[b,a] = residuez(R,p,k); % z-transform partial-
    fraction expansion
% b =   1.0000   -0.8966
% a =   1.0000   -1.5595    0.6065
b = real(b); a = real(a);
num = b; den = a;
maxLength = max([length(num), length(den)]);
num(length(num)+1:maxLength) = 0;
den(length(den)+1:maxLength) = 0;
printsys(num,den,'z')
% impulse response of the analog filter
t = 0:0.01:3; subplot(2,1,1); impulse(c,d,t);
axis([0,3,-0.1,1]);hold on
n0 = 0; n1 = 0; n2 = 3/T;
x = [zeros(1,(n0-n1)), 1, zeros(1,(n2-n0))]; %
    generates impulse sequence
% impulse response of the digital filter
n = 0:1:3/T; hn = filter(b,a,x);
stem(n*T,hn); xlabel('time in sec'); title ('
    Impulse Responses');
hold off
% magnitude response of the digital filter
[Hd,wd] = freqz(b,a,1000,'whole');
Hd = (Hd(1:1:501))'; wd = (wd(1:1:501))'; % 501
     frequency samples between 0 to pi radians
magd = abs(Hd); % absolute magnitude computed
    over 0 to pi radians
% magnitude response of the analog filter
wmax = 2*pi*Fs; % maximum frequency in rad/sec
    over which response is desired
ws = [0:1:500]*wmax/500; % array of 500
    frequency samples between [0 to wmax]
Hs = freqs(c,d,ws); % computation of s-domain
    frequency response
mags = abs(Hs); % absolute magnitude over [0 to
     wmax]
subplot(2,1,2); plot(ws/(2*pi),mags*Fs,wd/(2*pi
    )*Fs,magd)
xlabel('frequency in Hz'); title('Magnitude
    Responses');
ylabel('Magnitude');
text(1.3,.5,'Analog filter'); text(1.5,1.5,'
    Digital filter')
```

The digital filter is,

$$H(z) = \frac{1 - 0.8966z^{-1}}{1 - 1.5595z^{-1} + 0.6065z^{-2}}$$

as expected. Looking at the impulse responses and the magnitude responses (plotted up to the sampling frequency $1/T$) of the analog and the resulting digital filter. Clearly, the aliasing in the frequency domain is evident.

Lets re-design a digital filter to approximate the following normalized analog transfer function.

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}$$

Use the impulse invariance method to obtain $H(z)$. Assume 3 dB cut-off frequency of 150 Hz and sampling frequency of 1.28 kHz.

```
clc, clear all, close all
Fs = 1280; % sampling frequency (samples per
    second)
T = 1/Fs; % sampling time
fp = 150; % passband cut-off freq in Hz
Wp = 2*pi*fp; % digital passband cut-off freq
    in rads/sec
[num,den]=butter(2,1,'s'); % 2nd order
    normalized analogue butterworth filter with
     cut-off 1 rad/sec
printsys(num,den)
[numd,dend]=impinvar(num,den,0.5); % impulse
    invariance method
printsys(numd,dend,'z')
fvtool(numd,dend)
```

How about we re-design the lowpass digital butterworth filter with the following specifications using impulse invariance method.

$$\omega_p = 0.2\pi \quad R_p = 1\,dB$$
$$\omega_s = 0.3\pi \quad A_s = 15\,dB$$

Use the following MATLAB code.

```
clc, clear all, close all
% digital filter specifications
Wp = 0.2*pi; % digital passband freq in rad
Ws = 0.3*pi; % digital stopband freq in rad
Rp = 1; % passband ripple in dB
As = 15; % stopband attenuation in dB
% analog prototype specifications: inverse
    mapping for frequencies
T = 1; Fs = 1/T; % set T=1

OmegaP = Wp*T; % prototype passband freq
OmegaS = Ws*T; % prototype stopband freq
ep = sqrt(10^(Rp/10)-1); % passband ripple
    parameter
Ripple = sqrt(1/(1+ep*ep)); % passband ripple
Attn = 1/(10^(As/20)); % stopband attenuation
N = ceil((log10((10^(Rp/10)-1)/(10^(As/10)-1)))
    /(2*log10(Wp/Ws))); % order of the
    butterworth filter
fprintf('\n*** Butterworth Filter Order = %2.0f
     \n',N)
OmegaC = Wp/((10^(Rp/10)-1)^(1/(2*N))); %
    cutoff frequency in radians/sec
[z,p,k] = buttap(N); % butterworth analog
    lowpass filter prototype
p = p*OmegaC;
k = k*OmegaC^N;
B = real(poly(z));
b0 = k; cs = k*B; ds = real(poly(p));
% impulse invariance transformation
[R,p,k] = residue(cs,ds); % partial-fraction
    expansion (residues)
p = exp(p*T);
[b,a] = residuez(R,p,k); % z-transform partial-
    fraction expansion
b = real(b); a = real(a);
num = b; den = a;
maxLength = max([length(num), length(den)]);
num(length(num)+1:maxLength) = 0;
den(length(den)+1:maxLength) = 0;
printsys(num,den,'z')
% frequency response of digital filter
[H,w] = freqz(num,a,1000,'whole');
```

```
H = (H(1:1:501))'; w = (w(1:1:501))'; %  w =
    501 frequency samples between 0 to pi
    radians
mag = abs(H); % absolute magnitude computed
    over 0 to pi radians
db = 20*log10((mag+eps)/max(mag)); % Relative
    magnitude in dB computed over 0 to pi
    radians
pha = angle(H); % Phase response in radians
    over 0 to pi radians
grd = grpdelay(b,a,w); % Group delay over 0 to
    pi radians
thresh1 = -Rp*ones(1,length(mag)); % plot(w/pi,
    thresh1,w/pi,thresh2);
thresh2 = -As*ones(1,length(mag));
subplot(2,2,1);
plot(w/pi,mag);
title('Magnitude Response')
xlabel('frequency in \pi units'); ylabel('|H|')
axis([0,1,0,1.1])
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',[0,Attn,
    Ripple,1]); grid
subplot(2,2,2);
plot(w/pi,pha/pi);
title('Phase Response')
xlabel('frequency in \pi units'); ylabel('\pi
    units')
axis([0,1,-1,1]);
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',[-1,0,1]);
     grid
subplot(2,2,3);
plot(w/pi,db);
title('Magnitude Response in dB')
xlabel('frequency in \pi units'); ylabel('
    Decibels')
axis([0,1,-40,5]);
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick'
    ,[-50,-15,-1,0]); grid
subplot(2,2,4);
plot(w/pi,grd);
title('Group Delay')
xlabel('frequency in \pi units'); ylabel('
    Samples')
axis([0,1,0,11])
set(gca,'XTickMode','manual','XTick'
    ,[0,0.2,0.3,1]);
set(gca,'YTickmode','manual','YTick',0:2:10);
    grid
```

The desired filter is a 6^{th}-order Butterworth filter whose system function $H(z)$ is given in the parallel form,

$$H(z) = \frac{1.8587 - 0.6304z^{-1}}{1 - 0.9973z^{-1} + 0.257z^{-2}} + \frac{-2.1428 + 1.1454z^{-1}}{1 - 1.0691z^{-1} + 0.3699z^{-2}} + \frac{0.2871 - 0.4463z^{-1}}{1 - 1.2972z^{-1} + 0.6449z^{-2}}$$

4 Exercise

1. Design a second order high pass digital filter with cut-off frequency of 30 Hz and a sampling frequency of 150 Hz using bilinear and impulse invariance method. Sketch magnitude response, phase response, impulse response, step response and pole-zero diagram of the filter and compare the results. **Hint:** replace the command `lp2lp` to `lp2hp`

2. A discrete time band-stop filter with butterworth characteristics meeting the specifications given below is required.

stopband	$200 - 300$ Hz
sampling frequency	2 kHz
filter order	2

Obtain the coefficients of the filter using the bilinear transformation method.