Digital Signal and Image Processing Notes for Continuation-exam 19.02.24.

Table of Contents

1. Signal Fundamentals

Folding/Nyquist Frequency = Half of Sampling Frequency

2. Signal Processing Fundamentales

2.1. Difference equation

Determining the difference equation of a circuit. Look at the circuit, X(z) is the input and Y(z) is the output. The difference equation is then a linear combination of a subset (or all) inputs and outputs.

2.2. Transfer function

Determine transfer function H(z) = Y(z) / X(z)

Given a difference equation, we can derive a z-transfer function which is equal to ratio the z-transform of the output Y(z) and (divided by) the z-transform of the input X(z)

2.3. Pole-Zero form

A transfer function can be turned to a pole-zero form by making sure there are no negative powers of z in the numerator and denominator. In other words, there must be positive powers of z in both the numerator and denominator. Then take the factored form of the new transfer function.

2.4. Impulse and Step Response

Determining the impulse response of a circuit h(n). This is found by taking the inverse z-transform of the difference equation $h(t) = Z^{-1}\{H(t)\}$

2.5. System Properties

2.5.1. Stability

The system stability is determined by the poles of the z-transfer function H(z). The zeros do not affect stability. To find stability the poles are placed on the z-plane unit circle, their location and order determines the stability. The different cases is described as follows.

Stable: If the outmost pole(s) is(are) inside the unit circle, then the system is stable.

Marginally stable: If the outmost pole(s) is(are) first-order pole(s) and on the unit circle, then the system is marginally stable.

Unstable: If the outmost pole(s) is(are) outside the unit circle on the z-plane pole-zero plot, the system is unstable. Also, if the outmost pole(s) is(are) multiple-order pole(s) and on the unit circle, then the system is unstable.

2.5.2. Causality

A system is causal if the current output only depends on the current input and it's past values. If the ouput depends on the future output, x(n+1) + x(n+2)+…, the system is non-causal. Non-causal systems cannot be realized.

2.5.3. Time Invariance

The system is time-invariant if the output y_1(n) due to input x_1(n). This means then that if the input is shifted x_1(n - n0) then the output is shifted y_1(n - n0).

To determine time-invariancy shifted y_2 should be equal to shifted y_1. For example,

2.5.4. Linearity

If y_1(n) is the output due to x_1(n), then a linear system will fulfill y_2(n) is the output due to x_2(n).

Linearity holds if neither input nor output includes any non-linear terms, such as sin, cos, exponentials, logarithms, squares, cubes, etc.

Determining this can be done by setting y(n) equal to weighted sums y_1 and y_2. $y(n) = \alpha y_1(n) + \beta y_2(n)$. Apply the same to the input x(n), then $x(n) = \alpha x_1(n) + \beta y_2(n)$.

If the function is y(n) = x^2(n), then it's nonlinear because y_1(n)= x_1^2(n) and y_2(n)=x_2^2(n), then $\alpha y_1(n) + \beta y_2(n) = \alpha x_1^2(n) + \beta x_2^2(n)$. But y(n) = $(\alpha x_1(n) + \beta x_2(n))^2 = \alpha^2 x_1^2(n) + 2\alpha\beta x_1(n)x_2(n) + \beta^2 x_2^2(n)$, these do NOT match...

2.5.5. Poles and Zeros

Given a z-transfer function in factored form, the zeros and poles of the function can be determined. The zeros define when the numerator is equal to 0. The poles define when the denominator is equal to 0. $(\frac{numerator}{denominator})$.

3. Digital Image Processing

3.1. Grayscale images

An 8-bit image has each pixel stored as an 8-bit number. The storage amount required is then W x H (x 1 byte) bytes.

3.2. RGB images

An RGB images has pixeled stored as 3 x 8-bit color values, meaning 24 bits per pixel. Then the amount of bytes required is equal to W x H x 3

3.3. Indexed images

An indexed image has 8-bit pixel values equal to indices for a colormap. The colormap has 256 entries, each elemente has 3 8-bit colors, then the colormap takes 3 x 256 bytes of storage. The indexed image then requires W x H + 3 x 256 bytes of storage.

3.4. Intensity images

Intensity images use floating point values as pixel values, resulting in a grayscale image, then depending on the size of floats the storage requirement is W x H x sizeof(float)(in bytes)

4. FIR Filter Design

4.1. Fourier Transform Method

The Fourier transform method uses ideal impulse response functions to determine the coefficients. Table 7.1 describes these functions. These functions depend on normalized cutoff frequencies. Therefore, these must be found beforehand. The required frequencies depend on the type of filter being designed. The filter functions are symmetrical around n = 0 and must therefore be delayed by M, b(n)=h(n-M) for n = 0, …, 2M.

1. Calculate normalized cutoff frequencies
2. Lookup table 7.1 for the filter function
3. $h(n) = function(n,\ norm\ co\ freqs), for - M \leq n \leq M$
4. Delay the coefficients by M

$$b(n) = h(n - M), for\ n = 0, 1, \dots, 2M$$

The transfer function and following difference equation can the be found from the calculated b(n) coefficients.

This method results in the Gibbs effect, which shows ripple in the magnitude and phase responses of the filter, which is undesirable. The window method solves this problem by tapering the filter coefficients.

4.2. Window method

The window method applies a window-function to the filter coefficients. To get rid of the Gibbs effect, the filter coefficients are truncated and scaled. There exist several types of

Table 7.x shows these window functions. Some examples are rectangular, Bartlett, Hamming, Hanning and Blackman windows.

Table 7.7 shows the specifications of each window type.

Hamming window example

$$w_{ham}(n) = 0.54 + 0.46\cos\left(\frac{n\pi}{M}\right),\ -M \leq n \leq M$$

Then filter coefficient h(n) becomes:

$$h_{win}(n) = h(n) * w_{ham}(n)$$

Window specification may not be granted; thus we have to calculate it ourselves, defined in steps:

1. Identify the correct window depending on ripple and/or attenuation requirements.
2. Calculate $\Delta f = \frac{f_{stop} - f_{pass}}{f_s}$ to find required order $N = \frac{const}{\Delta f}$. Round up to an odd number.
3. Cutoff frequency $f_c = \frac{f_{pass} + f_{stop}}{2}$ and normalized $\omega_c = \frac{2\pi f_c}{f_s}$

4. Time delay in passband (linear phase) is $\frac{N}{2} * \frac{1}{f_s}$

Having found the desired window, the numerical windowing method is described next.

The window method is described in 3 steps as follows.

1. Obtain the FIR filter coefficients from the Fourier transform method.
2. Multiply the generated coefficients by the selected window function
$$h_w(n) = h(n)w(n),\ \ n = -M, \dots, 0, 1, \dots, M$$
3. Delay the windowed impulse sequence h_w(n) by M samples to get the windowed FIR filter coefficients.
$$b_n = h_w(n - M),\ \ for\ n = 0, 1, \dots, 2M$$

4.3. Frequency Sampling method

The sampling method is defined by 3 steps.

1. Given filter length (taps) N=2M+1, calculate the magnitude frequency response H_k for normalized frequency omega_k from 0 to PI.
$$H_k\ at\ \omega_k = \frac{2\pi k}{(2M+1)}\ for\ k = 0,1, \dots, M$$
2. Calcualte the FIR filter coeffficents
$$b_n = h(n) = \frac{1}{2M+1}\left\{H_0 + 2\sum_{k=1}^{M} H_k \cos\left(\frac{2\pi k(n-M)}{2M+1}\right)\right\} for\ n = 0,1, \dots, M$$

3. Use the following symmetry to find the rest of the coefficients.
$$h(n) = h(2M - n)\ for\ n = M+1, \dots, 2M$$

Code example

```
% Fig. 7.31 (Example 7.13)
% MATLAB program to create Fig. 7.31
fs=8000; % sampling frequency
H1=[1 1 1 1 1 1 1 0 0 0 0 0 0 0]; % magnitude specifications
B1=firfs(25,H1); % design filter
[h1,f]=freqz(B1,1,512,fs); % calculate magnitude frequency response
H2=[1 1 1 1 1 1 1 0.5 0 0 0 0 0 0]; % magnitude specifications
B2=firfs(25,H2); % Design filter
[h2,f]=freqz(B2,1,512,fs); % calculate magnitude frequency response
p1=180*unwrap(angle(h1))/pi;
p2=180*unwrap(angle(h2))/pi
subplot(2,1,1); plot(f,20*log10(abs(h1)),'-.',f,20*log10(abs(h2)));grid
```

```
axis([0 fs/2 -80 10]);
xlabel('Frequency (Hz)'); ylabel('Magnitude Response (dB)');
subplot(2,1,2); plot(f,p1,'-.',f,p2);grid
xlabel('Frequency (Hz)'); ylabel('Phase (degrees)');
```

4.4. Optimal design method

Given specified passbands/stopbands, ripple and attenuation, sample rate and filter order(?) we can use the Parks-McCellen algorithm.

The design method is specified in 5 steps.

1. Acquire specifications, band edges, ripple, attenuation, filter order and sample rate.
2. Find normalized band edges with the Nyquist frequency and ideal magnitudes. Note: Edge frequencies in the passband have ideal magnitudes of 1, and frequencies in the stopband have ideal magnitudes of 0.
3. Find error weights W_s and W_p. Turn ripple and attenuation from dB to absolute values, by using inverse decibel equation, and find the ratio $\frac{\delta_p}{\delta_s} = \frac{W_s}{W_p}$.
4. Apply Remez algorithm to calculate the filter coefficients. This can be done using the Firpm function in MATLAB, see section 5.4.
5. If the filter is not satisfactory, increase the order and repeat steps 1-4.

Code example:

```
fs=8000;
f=[ 0 0.2 0.25 1]; % normalized edge frequencies
m=[ 1 1 0 0] ; % ideal magnitudes
w=[ 1 12 ]; % error weight factors -> order = W_p, W_s
n=53 %filter order
b=firpm(n,f,m,w); % (53+1)Parks-McClellen algorithm and Remez exchange
format long
freqz(b,1,512,fs) % plot the frequency response
axis([0 fs/2 -80 10]);
```

4.5. Filter Order & Poles & Zeros

The amount of poles and zeros in a FIR filter is equal to the filter order.

5. FIR Filters in MATLAB

5.1. Fir1

Fir1 is the method to design a WINDOW-METHOD FIR frequency filter

b = fir1(n, Wn, ftype, window, scaleopt)

- n = filter order.
- Wn =

- If scalar $0 \leq Wn \leq 1$ then the filter is a lowpass or highpass filter with cutoff freq Wn.

-[w1 w2] Two element vector designs a bandpass or bandstop filter with lower fc w1 and upper w2

-More than two elements creates a multiband filter, which will not be asked for

- Ftype =

- 'low' specifies a lowpass filter with cutoff frequency Wn. 'low' is the default for scalar Wn.

- 'high' specifies a highpass filter with cutoff frequency Wn.

- 'bandpass' specifies a bandpass filter if Wn is a two-element vector. 'bandpass' is the default when Wn has two elements.

- 'stop' specifies a bandstop filter if Wn is a two-element vector.

- Window = Bartlett, blackman, chebwin, hamming, hann, rectwin, barthannwin
- B = Output filter coefficients

Examples:

Bandpass FIR filter: The following bandpass filter with passband $0.35\pi\backslash sample \leq w \leq 0.65\pi\backslash sample$

```
b = fir1(48,[0.35 0.65]);

freqz(b,1,512)
```

Highpass FIR filter: The following highpass filter has cutoff frequency of 0.48 and a Chebyshev window with 30 dB of ripple

```
bhi = fir1(34,0.48,'high',chebwin(35,30));

freqz(bhi,1)

outhi = filter(bhi,1,y);
```

5.2. Fir2

Fir2 is the method to design a sampling-method FIR frequency filter

b = fir2(n,f,m) returns row vector b containing the n+1 coefficients of an order n FIR filter. The frequency-magnitude characteristics of this filter match those given by vectors f and m:

- f is a vector of frequency points in the range from 0 to 1, where 1 corresponds to the Nyquist frequency. The first point of f must be 0 and the last point 1. The frequency points must be in increasing order.

- m is a vector containing the desired magnitude response at the points specified in f.

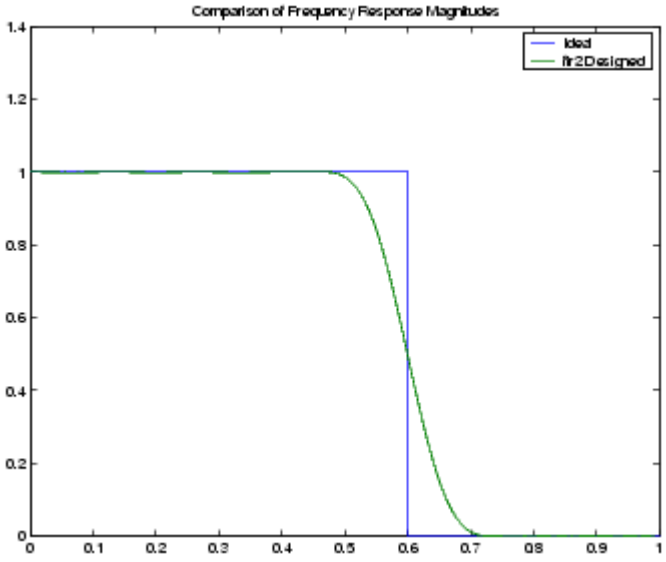
- f and m must be the same length.

- Duplicate frequency points are allowed, corresponding to steps in the frequency response.

Use plot(f,m) to view the filter shape.

Example:

```
f = [0 0.6 0.6 1];

m = [1 1 0 0];

n = 30 % Order

b = fir2(n,f,m);

[h,w] = freqz(b,1,128);

plot(f,m,w/pi,abs(h))

legend('Ideal','fir2
Designed')

title('Comparison of Frequency
Response Magnitudes')
```



5.3. Firfs

The function Firfs uses the sampling method to generate filter coefficients.

B = firfs(n, h)

N = filter order, must be odd for M (n=2M+1) to be an integer.

H = The calculated magnitude frequency response (from Step 1 of Frequency sampling method).

Code example:

```
% Fig. 7.31 (Example 7.13)
% MATLAB program to create Fig. 7.31
fs=8000; % sampling frequency
H1=[1 1 1 1 1 1 0 0 0 0 0 0]; % magnitude specifications
B1=firfs(25,H1); % design filter
[h1,f]=freqz(B1,1,512,fs); % calculate magnitude frequency response
H2=[1 1 1 1 1 1 0.5 0 0 0 0 0]; % magnitude specifications
B2=firfs(25,H2); % Design filter
[h2,f]=freqz(B2,1,512,fs); % calculate magnitude frequency response
p1=180*unwrap(angle(h1))/pi;
p2=180*unwrap(angle(h2))/pi
subplot(2,1,1); plot(f,20*log10(abs(h1)),'-.',f,20*log10(abs(h2)));grid
axis([0 fs/2 -80 10]);
xlabel('Frequency (Hz)'); ylabel('Magnitude Response (dB)');
subplot(2,1,2); plot(f,p1,'-.',f,p2);grid
xlabel('Frequency (Hz)'); ylabel('Phase (degrees)');
```

5.4. Firpm

The firpm function implements the Parks-McCellen + Remez algorthim.

Note section 4.4 for explonation of this method.

Firpm(n, f, m, w)

N = filterorder

F = normalized edge frequecies

M = ideal magnitudes

W = Error Weights W_p and W_s

Code example:

```
fs=8000;
f=[ 0 0.2 0.25 1]; % edge frequencies
m=[ 1 1 0 0] ; % ideal magnitudes
w=[ 1 12 ]; % error weight factors W_p, W_s
n=53 %filter order
b=firpm(n,f,m,w); % (53+1)Parks-McClellen algorithm and Remez exchange
format long
freqz(b,1,512,fs) % plot the frequency response
axis([0 fs/2 -80 10]);
```

5.5. Check Gain

The gain at different frequencies can be found via the freqz function. Code example:

```
B=fir1(N,wn,'high'); % filter coeffisients, Hamming window as default
freqz(B,1,1000,fs);
[H f] = freqz(B,1,1000,fs); % frequency response
gain_at_2kHz = 20*log10(abs(H(200)))
plot(f,20*log10(abs(H))); grid; xlabel('Frequency (Hz)'); ylabel('dB');
```

5.6. Apply Filter

We have generated a filter b. We have signal y.

Y_filtered = filter(b, 1, y)

Where Y_filtered is the filtered signal.

6. IIR Filter Design

Infinite impulse response filters use an infinite count of filter coefficients, hence the infinite name.

An IIR filter is described the following difference equation.

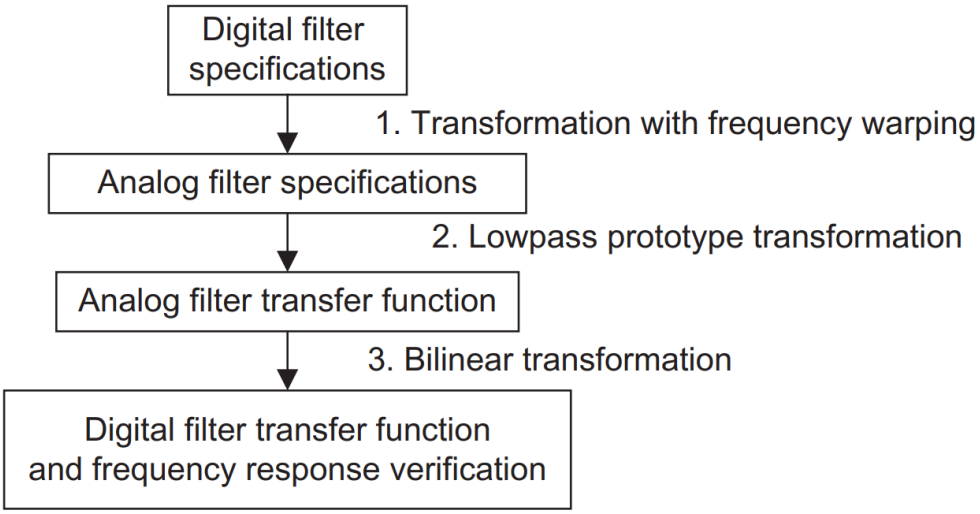
$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M)$$
$$- a_1 y(n-1) - \cdots - a_N y(n-N)$$

The transfer function is as follows.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

The b_i (M+1) numerator and a_i (N) denominator are the filter coefficients. Y(z) and X(z) are respectively the z-transform functions of input x(n) and output y(n).

6.1. Bilinear Transformation Method



$$y(t) = \int_0^t x(t)\,dt$$

Laplace transform yields the following.

$$Y(s) = \frac{X(s)}{s}$$

The Laplace transfer function is then

$$G(s) = \frac{Y(s)}{X(s)} = \frac{1}{s}$$

Using a numerical approximation of the integral yields the following.

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

Applying the z-transform to this equation yields:

$$Y(z) = z^{-1}Y(z) + \frac{T}{2}(X(z) + z^{-1}X(z))$$

Solving the ratio of Y(z)/X(z) gives the z-transfer function:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$$

It then follows:

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

This is a method to map or transform the points on the s-plane to the z-plane.

Given an analog filter transfer function H(s)=1/s it can be converted to a digital filter transfer function using the BLT method.

This is achieved by substituting s into H(s).
Into the given transfer function H(s) then it follows:

$$H(z) = H(s)|_{s=\frac{2z-1}{Tz+1}} = \frac{1}{\frac{2}{T}\frac{z-1}{z+1}}$$

Solving for T = 1/fs and simplifying the equation yields the difference equation. The digital filter transfer function is then defined by the difference equation H(z)=Y(z)/X(z).

Next the frequency mapping between s-plane and z-plane is done. The analog frequency w_a is on the jw axis on the s-plane. W_d is the digital frequency on the unit circle in the z-plane.

S=jw_a and z=e^jw_a*T are substituted into the BLT function.

$$jw_a = \frac{2}{T}\frac{e^{jw_dT} - 1}{e^{jw_dT} + 1}$$

Simplifying yields:

$$w_a = \frac{2}{T}\tan\left(\frac{w_dT}{2}\right)$$

And the inverse is

$$w_d = \frac{2}{T}\tan^{-1}\left(\frac{w_aT}{2}\right)$$

Analog frequencies can now be converted to digital frequencies, and vice-versa.
Then the analog lowpass filter H(s) can be found using prewarped analog frequency w_a and the lowpass prototype.

$$H(s) = H_p(s)|_{s=\frac{s}{w_a}} = H_p\left(\frac{s}{w_a}\right)$$

Then using BLT substitution:

$$H(z) = H(s)|_{s=\frac{2z-1}{Tz+1}}$$

The method is defined by 3 steps.

1. Given digital filter frequency specifications in radians per second (w_d = 2pi*f= 2f r/s), prewarp the digital frequency specifications to analog frequency specifications.

   For lowpass and highpass filters

   $$w_a = \frac{2}{T}\tan\left(\frac{w_dT}{2}\right)$$

   For bandpass and bandstop:

   $$w_{al} = \frac{2}{T}\tan\left(\frac{w_lT}{2}\right)$$

   $$w_{ah} = \frac{2}{T}\tan\left(\frac{w_hT}{2}\right)$$

   $$w_0 = \sqrt{w_{al}w_{aj}}$$

   $$W = w_{ah} - w_{al}$$

2. Perform the prototype transformation using the lowpass prototype H_p(s)

$$From\ lowpass\ to\ lowpass: H(s) = H_p(s)|_{s=\frac{s}{w_a}}$$

$$From\ lowpass\ to\ highpass: H(s) = H_p(s)|_{s=\frac{w_a}{s}}$$

$$From\ lowpass\ to\ bandpass: H(s) = H_p(s)|_{s=\frac{s^2+w_0^2}{sW}}$$

$$From\ lowpass\ to\ bandpass: H(s) = H_p(s)|_{s=\frac{sW}{s^2+w_0^2}}$$

3. Subsitutie the BLT to obtain the digital filter

$$H(z) = H(s)|_{s=\frac{2z-1}{Tz+1}}$$

6.1.1. Butterworth

1. Given specifications, turn the digital frequency specifications to radians per second. Apply for both passband ripple (A_p) and stopband attenuation (A_s).

$$w_d = 2\pi f = 2f\ rad/sec$$

2. BLT method can begin. First apply the warping equation to turn the digital frequencies (in rad/sec) into analog frequencies.

$$w_{ap}\ (and\ w_{as}) = \frac{2}{T}\tan\left(\frac{w_d T}{2}\right)$$

Using table 8.6 find the Lowpass prototype specification for the filter type being designed.

$$v_p = 1$$

$$v_s = (for\ example)\frac{w_{as}}{w_{ap}}$$

And compute the filter order (unless otherwise specified):

$$\epsilon^2 = 10^{0.1*A_p} - 1$$

$$n \geq \frac{\log_{10}\frac{10^{0.1*A_s}-1}{\epsilon^2}}{2*\log_{10}(v_s)}$$

3. Given n (integer), table 8.3 (for 3dB Butterworth prototype) yields the transfer function. The first order 3dB ripple Butterworth prototype transfer function is given:

$$H_p(s) = \frac{1}{s+1}$$

Then apply prototype transformational, see end of previous (section 6.1) section. Lowpass to lowpass is given as example:

$$H(s) = H_p(s)|_{\frac{s}{w_{ap}}}$$

4. Apply BLT transformation.

$$H(z) = H(s)|_{s=\frac{2z-1}{Tz+1}}$$

6.1.2. Chebyshev Type 1

The Chebyshev filter follow much the same procedure as the Butterworth, however with different lookup tables, specifically prototype transfer functions.

6.2. Cascade Method

I kinda very much doubt this is coming

The Cascade method uses a different lookup table for lowpass prototype transfer functions. From there it is exactly the same, although requires more difficult algebraic simplifications.

However, this eventually ends up in a difference equation that is a factor of 2 “typical” difference equations. Therefore this method is best applied in MATLAB, where each of the factors is

The fourth order lowpass Butterworth prototype transfer function:

$$\frac{1}{(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)}$$

Example 8.14 shows the BLT + bilinear process that yields this difference equation:

$$H(z) = \frac{0.5108 + 1.0215z^{-1} + 0.5108z^{-2}}{1 + 0.5654z^{-1} + 0.4776z^{-2}} * \frac{0.3730 + 0.7460z^{-1} + 0.3730z^{-2}}{1 + 0.4129z^{-1} + 0.0790z^{-2}}$$

Code, note that the MATLAB script outputs the correct coefficients compared with the difference equation above. Note also the filters are combined using the conv() function:

```
fs=8000; % Sampling rate
[B1 A1]=lp2lp([1][1 0.7654 1], 2.3946*10^4) % Complete step 2
[b1 a1]=bilinear(B1,A1,fs) % complete step 3
[B2 A2]=lp2lp([1][1 1.8478 1], 2.3946*10^4) % Complete step 2
[b2 a2]=bilinear(B2,A2,fs) % complete step 3
% Plot the magnitude and phase responses jH(z)j
% b1=[0.5108 1.0215 0.5108]; a1=[1 0.5654 0.4776]; coefficients from MATLAB
% b2=[0.3730 0.7460 0.3730]; a2=[1 0.4129 0.0790]; coefficients from MATLAB
freqz(conv(b1,b2),conv(a1,a2),512,fs); % Combined filter responses
```

```
axis([0 fs/2 -40 10]);
```

6.3. Impulse Invariant Method

The impulse invariant design method uses the inverse Laplace transform of analog transfer function to generate the impulse response. Then the impulse response is sampled using T = 1/fs. Then the digital filter transfer function H(z) is found via the z-transform of the sampled impulse response.

6.4. Pole-Zero Placement Method

Only remaining thing

Quite simple calculations.

Poles are complex conjuagtes with magnitude r and angle Theta.

Zeros are placed at z=1 corresponding to DC(zero frequency), and z=-1, corresponding at the folding frequency.

R controls bandwidth

Theta controls location of central frequency

Location of poles and zeros controls magnitude

Location of poles controls stability

Number of zeros controls phase linearity

Theta = f0 / fs (central frequency / sampling rate)

Usually, a pair of complex conjugate zeros are placed on the unit circle in the z-plane with angle θ, yields a numerator of $(z - e^{j\theta})(z - e^{-j\theta})$ om the transfer function. The magnitude contribution to the frequency response at $z = e^{j\theta}$ is $(e^{j\omega} - e^{j\theta})(e^{j\omega} - e^{-j\theta})$. Therefore, if $\omega = \theta$ the magnitude reaches zero.

When a pair of complex conjugate poles are placed within the unit circle, the denominator equals $(z - re^{j\theta})(z - re^{-j\theta})$, where r is the chosen radius less than and close to 1 which places the poles inside the unit circle. The magnitude contribution to the frequency response at $\omega = \theta$ will rise to a large magnitude, because the first factor $\left(e^{j\theta} - re^{j\theta}\right) = (1 - r)e^{j\theta}$ gives a small magnitude of $1 - r$ which is the length between the pole and unit circle at angle $\omega = \theta$. Note the magnitude of $e^{j\theta}$ is 1.

Therefore magnitude response can be reduced using zero placement and increase using pole placement. A combination of poles and zeros will ersult in different frequency responses, such as lp, hp, bp, bs. The

method is intiuative and approximate, and is easy to compute for simple IIR filters. The method has good performance when the bandpass and bandstop filters have very narrow bandwidth requirements and when the lowpass and highpass filters have either very low cutoff frequencies close to DC or very high cutoff frequencies close to the folding frequency.

6.4.1. Second Order Bandpass

The design process is described in 4 steps:

1. $\theta = \left(\frac{f_0}{f_s}\right) * 360\ degrees$
2. $r \approx 1 - \left(\frac{BW_{3dB}}{f_s}\right) * \pi, \quad for\ 0.9 \leq r < 1$
3. $H(z) = \frac{K(z-1)(z+1)}{(z-re^{j\theta})(z-re^{-j\theta})} = \frac{K(z^2-1)}{(z^2-2rz*\cos(\theta)+r^2}$
4. $K = \frac{(1-r)\sqrt{1-2r*\cos(2\theta)+r^2}}{2|\sin(\theta)|}$

6.4.2. Second Order Bandstop (Notch)

1. $\theta = \left(\frac{f_0}{f_s}\right) * 360\ degrees$
2. $r \approx 1 - \left(\frac{BW_{3dB}}{f_s}\right) * \pi, \quad for\ 0.9 \leq r < 1$
3. $H(z) = \frac{K(z-e^{j\theta})(z+e^{-j\theta})}{(z-re^{j\theta})(z-re^{-j\theta})} = \frac{K(z^2-2z*\cos(\theta)+1}{(z^2-2rz*\cos(\theta)+r^2}$
4. $K = \frac{1-2r*\cos(\theta)+r^2}{2-2\cos(\theta)}$

6.4.3. First Order Lowpass

1. $f_c < \frac{f_s}{4}, \quad a \approx 1 - 2 * \frac{f_c}{f_s} * \pi, \quad for \quad 0.9 \leq a < 1$
2. $f_c > \frac{f_s}{4}, \quad a \approx -\left(1 - \pi + 2 * \frac{f_c}{f_s} * \pi\right), \quad for \ -1 < a \leq -0.9$
3. $H(z) = \frac{K(z+1)}{z-a}$
4. $K = \frac{1-a}{2}$

6.4.4. First Order Highpass

5. $f_c < \frac{f_s}{4}, \quad a \approx 1 - 2 * \frac{f_c}{f_s} * \pi, \quad for \quad 0.9 \leq a < 1$
6. $f_c > \frac{f_s}{4}, \quad a \approx -\left(1 - \pi + 2 * \frac{f_c}{f_s} * \pi\right), \quad for \ -1 < a \leq -0.9$
7. $H(z) = \frac{K(z-1)}{z-a}$
8. $K = \frac{1+a}{2}$

7. IIR Filters in MATLAB

Butterworth

Input Arguments

n — Filter order

Filter order, specified as an integer scalar. For bandpass and bandstop designs, n represents one-half the filter order.

Wn — Cutoff frequency

Cutoff frequency, specified as a scalar or a two-element vector. The cutoff frequency is the frequency at which the magnitude response of the filter is $1 / \sqrt{2}$.

- If Wn is scalar, then butter designs a lowpass or highpass filter with cutoff frequency Wn.
- If Wn is the two-element vector [w1 w2], where w1 < w2, then butter designs a bandpass or bandstop filter with lower cutoff frequency w1 and higher cutoff frequency w2.
- For digital filters, the cutoff frequencies must lie between 0 and 1, where 1 corresponds to the Nyquist rate—half the sample rate or π rad/sample.
- For analog filters, the cutoff frequencies must be expressed in radians per second and can take on any positive value.

ftype — Filter type
'low' | 'bandpass' | 'high' | 'stop'

Filter type, specified as one of the following:

'low' specifies a lowpass filter with cutoff frequency Wn. 'low' is the default for scalar Wn.

'high' specifies a highpass filter with cutoff frequency Wn.

'bandpass' specifies a bandpass filter of order 2n if Wn is a two-element vector. 'bandpass' is the default when Wn has two elements.

'stop' specifies a bandstop filter of order 2n if Wn is a two-element vector.

Output Arguments

[b a] = Transfer function coefficients. N + 1 for lowpass and highpass. 2N + 1 for bandpass and bandstop.

Or [z p k] = Zeros, poles and gain. N for lowpass highpass. 2N for bandpass bandstop.

7.1. Bilinear Transformation Design

Lowpass to lowpass.

$$[B, A] = lp2lp(Bp, Ap, wa)$$

Lowpass to highpass.

$$[B, A] = lp2hp(Bp, Ap, wa)$$

Lowpass to bandpass.

$$[B, A] = lp2bp(Bp, Ap, w0, W)$$

Lowpass to bandstop.

$$[B, A] = lp2bs(Bp, Ap, w0, W)$$

Bilinear transform to get digital filter coefficients.

$$[b, a] = bilinear(B, A, fs)$$

Plot the magnitude and phase.

$$freqz(b, a, 512, fs)$$

7.2. Butterworth

7.2.1. Buttord & Butter

N, Wn= Buttord(Wp, Ws, Rp, Rs)

Wp = Normalized passband edge frequency. For passband or bandstop filters the normalized frequency edges of the passband can be input as [x, y].

Ws = Normalized stopband edge frequency. For passband or bandstop filters the normalized frequency edges of the stopband can be input as [x, y].

Rp = Required passband ripple dB.

Rs = Required passband attenuation dB.

B, A = Butter(n, Wn)

N = Order of filter.

Wn = Cutoff frequency.

7.2.2. Lp2 & Bilinear

1. Using table 8.3 (first order 3dB Butterworth lowpass prototype transfer functions) coefficients are determined.

[B, A] = lp2(lp/hp/bp/bs)(Bp, Ap, wa / w0, W).

Bp are the coefficients of the numerator of the lowpass prototype transfer function, found in table 8.3.

Ap are the coefficients of the denominator of the lowpass prototype transfer function, found in table 8.3.

For example, the second order prototype transfer function:

$$H_p(s) = \frac{1}{s^2 + 1.4142s + 1}$$

Gives Bp = [1] and Ap = [1, 1.4142, 1].

Wa is the cutoff frequency of the lowpass or highpass analog filter, computed in the first step of the Butterworth procedure.

W0 and W are also computed in the first steps of the Butterworth procedure if the we're designing is a bandpass or bandstop filter.

2. The

[b, a] = bilinear(B, A, fs)

B and A are found in the previous step.

Fs is the sampling rate.

7.3. Chebyshev Type 1

7.3.1. Cheby1ord & Cheby1

N, Wp = Cheby1ord(Wp, Ws, Rp, Rs)

Wp = Normalized passband edge frequency.

Ws = Normalized stopband edge frequency.

Rp = Required passband ripple dB.

Rs = Required passband attenuation dB.

B, A = Cheby1 (n, Rp, Wp)

N = Order of filter.

Rp = Peak-to-Peak passband ripple

Wp = Normalized passband edge frequency