Digital Filter:

Description:

Digital filters refers to the hard ware and software implementation of the mathematical algorithm which accepts a digital signal as input and produces another digital signal as output whose wave shape, amplitude and phase response has been modified in a specified manner.
Digital filter play very important role in DSP. Compare with analog filters they are preferred in number of application due to following advantages.

- ✓ Truly linear phase response
- ✓ Better frequency response
- ✓ Filtered and unfiltered data remains saved for further use.

There are two types of digital filters.

- FIR (finite impulse response) filter
- IIR (infinite impulse response) filter

Design of FIR filters using Matlab commands.

FIR1:

FIR filters design using the window method. B = FIR1(N,Wn) designs an N'th order low pass FIR digital filter and returns the filter coefficients in length N+1 vector B. **The cutoff frequency Wn must be between 0 < Wn < 1.0, with 1.0 corresponding to half the sample rate**. The filter B is real and has linear phase. The normalized gain of the filter at Wn is -6 dB.
B = FIR1(N,Wn,'high') designs an N'th order highpass filter. You can also use B = FIR1(N,Wn,'low') to design a lowpass filter. If Wn is a two-element vector, Wn = [W1 W2], FIR1 returns an order N bandpass filter with passband W1 < W < W2.
B = FIR1(N,Wn,'stop') is a bandstop filter if Wn = [W1 W2]. You can also specify If Wn is a multi-element vector, Wn = [W1 W2 W3 W4 W5 ... WN], FIR1 returns an order N multiband filter with bands 0 < W < W1, W1 < W < W2, ..., WN < W < 1.
B = FIR1(N,Wn,'DC-1') makes the first band a passband.
B = FIR1(N,Wn,'DC-0') makes the first band a stopband.

By default FIR1 uses a Hamming window. Other available windows, including Boxcar, Hann, Bartlett, Blackman, Kaiser and Chebwin can be specified with an optional trailing argument. For example, B = FIR1(N,Wn,kaiser(N+1,4)) uses a Kaiser window with beta=4. B = FIR1(N,Wn,'high',chebwin(N+1,R)) uses a Chebyshev window.
For filters with a gain other than zero at Fs/2, e.g., highpass and bandstop filters, N must be even. Otherwise, N will be incremented by one. In this case the window length should be specified as N+2.
By default, the filter is scaled so the center of the first pass band has magnitude exactly one after windowing. Use a trailing 'noscale' argument to prevent this scaling, e.g.
B = FIR1(N,Wn,'noscale')
B = FIR1(N,Wn,'high','noscale')

B = FIR1(N,Wn,wind,'noscale').
You can also specify the scaling explicitly, e.g. FIR1(N,Wn,'scale'), etc.

FREQZ Digital Filter Frequency Response.

[H,W] = FREQZ(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter: given numerator and denominator coefficients in vectors B and A. The frequency response is evaluated at N points equally spaced around the upper half of the unit circle. If N isn't specified, it defaults to 512.
[H,W] = FREQZ(B,A,N,'whole') uses N points around the whole unit circle.
H = FREQZ(B,A,W) returns the frequency response at frequencies designated in vector
W, in radians/sample (normally between 0 and pi).
[H,F] = FREQZ(B,A,N,Fs) and [H,F] = FREQZ(B,A,N,'whole',Fs) return frequency vector F (in Hz), where Fs is the sampling frequency (in Hz).
H = FREQZ(B,A,F,Fs) returns the complex frequency response at the frequencies designated in vector F (in Hz), where Fs is the sampling frequency (in Hz).
[H,W,S] = FREQZ(...) or [H,F,S] = FREQZ(...) returns plotting information to be used with FREQZPLOT. S is a structure whose fields can be altered to obtain different frequency response plots. For more information see the help for FREQZPLOT.
FREQZ(B,A,...) with no output arguments plots the magnitude and unwrapped phase of the filter in the current figure window.

Designing a Low Pass Filter:

Suppose out target is to pass all frequencies below 1200 Hz

```
fs=8000; % sampling frequency
n=50; % order of the filter
w=1200/ (fs/2);
b=fir1(n,w,'low'); % Zeros of the filter
freqz(b,1,128,8000); % Magnitude and Phase Plot of the filter
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);
```

Designing High Pass Filter:

Now our target is to pass all frequencies above 1200 Hz

```
fs=8000;
n=50;
w=1200/ (fs/2); b=fir1(n,w,'high');
freqz(b,1,128,8000);
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
```

```
zplane(b,1);
```

Designing High Pass Filter:

```
fs=8000;
n=50;
w=1200/ (fs/2);
b=fir1(n,w,'high');
freqz(b,1,128,8000);
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);
```

Designing Band Pass Filter:

```
fs=8000;
n=40;
b=fir1(n,[1200/4000 1800/4000],'bandpass');
freqz(b,1,128,8000)
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);
```

Designing Band stop Filter:

```
fs=8000;
n=40;
b=fir1(n,[1200/4000 2800/4000],'stop');
freqz(b,1,128,8000)
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);
```

Designing Notch Filter

```
fs=8000;
n=40;
b=fir1(n,[1500/4000 1550/4000],'notch');
freqz(b,1,128,8000)
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
```

```
figure(3)
zplane(b,1);
```

Designing Multi Band Filter

```
n=50;
w=[0.2 0.4 0.6];
b=fir1(n,w);
freqz(b,1,128,8000)
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);
```

Designing of IIR filters by Matlab commands.

Matlab contains various routines for design and analyzing digital filter IIR. Most of these are part of the signal processing tool box. A selection of these filters is listed below.

- ✓ Buttord (for calculating the order of filter)
- ✓ Butter (creates an IIR filter)
- ✓ Ellipord (for calculating the order of filter)
- ✓ Ellip (creates an IIR filter)
- ✓ Cheb1ord (for calculating the order of filter)
- ✓ Cheyb1 (creates an IIR filter)

Explanation of the Commands for Filter Design:

Buttord:

Butterworth filter order selection.

[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Butterworth filter that loses no more than Rp dB in the pass band and has at least Rs dB of attenuation in the stop band.

Wp and Ws are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample). For example

Low pass: Wp = .1, Ws = .2

High pass: Wp = .2, Ws = .1

Band pass: Wp = [.2 .7], Ws = [.1 .8]

Band stop: Wp = [.1 .8], Ws = [.2 .7]

BUTTORD also returns Wn, the Butterworth natural frequency (or, the "3 dB frequency") to use with BUTTER to achieve the specifications.

[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs, 's') does the computation for an analog filter, in

which case Wp and Ws are in radians/second. When Rp is chosen as 3 dB, the Wn in BUTTER is equal to Wp in BUTTORD.

Butter:

Butterworth digital and analog filter design.
[B,A] = BUTTER(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.
If Wn is a two-element vector, Wn = [W1 W2], BUTTER returns an order 2N bandpass filter with passband W1 < W < W2.
[B,A] = BUTTER(N,Wn,'high') designs a highpass filter.
[B,A] = BUTTER(N,Wn,'stop') is a bandstop filter if Wn = [W1 W2].
When used with three left-hand arguments, as in [Z,P,K] = BUTTER(...), the zeros and poles are returned in length N column vectors Z and P, and the gain in scalar K. When used with four left-hand arguments, as in [A,B,C,D] = BUTTER(...), state-space matrices are returned.
BUTTER(N,Wn,'s'), BUTTER(N,Wn,'high','s') and BUTTER(N,Wn,'stop','s') design analog Butterworth filters. In this case, Wn is in [rad/s] and it can be greater than 1.0.

Buttord and Butter Filter:

Designing IIR Low Pass Filter:

Suppose our target is to design a filter to pass all frequencies below 1200 Hz with pass band ripples = 1 dB and minimum stop band attenuation of 50 dB at 1500 Hz. The sampling frequency for the filter is 8000 Hz;

```
fs=8000;
[n,w]=buttord(1200/4000,1500/4000,1,50); % finding the order of the filter
[b,a]=butter(n,w); % finding zeros and poles for filter
figure(1)
freqz(b,a,512,8000);
figure(2)
[h,q] = freqz(b,a,512,8000);
plot(q,abs(h)); % Normalized Magnitude plot
grid
figure(3)
f=1200:2:1500;
freqz(b,a,f,8000) % plotting the Transition band
figure(4)
zplane(b,a) % pole zero constellation diagram
```

Designing IIR High Pass Filter:

We will consider same filter but our target now is to pass all frequencies above 1200 Hz

```
[n,w]=buttord(1200/5000,1500/5000,1,50);
[b,a]=butter(n,w,'high');
figure(1)
freqz(b,a,512,10000);
```

```
figure(2)
[h,q] = freqz(b,a,512,8000);
plot(q,abs(h)); % Normalized Magnitude plot
grid
figure(3)
f=1200:2:1500;
freqz(b,a,f,10000)
figure(4)
zplane(b,a)
```

Designing IIR Band Pass Filter:

Now we wish to design a filter to pass all frequencies between 1200 Hz and 2800 Hz with pass band ripples = 1 dB and minimum stop band attenuation of 50 dB. The sampling frequency for the filter is 8000 Hz;

```
[n,w]=buttord([1200/4000,2800/4000],[400/4000, 3200/4000],1,50);
[b,a]=butter(n,w,'bandpass');
figure(1)
freqz(b,a,128,8000)
figure(2)
[h,w]=freqz(b,a,128,8000);
plot(w,abs(h))
grid
figure(3)
f=600:2:1200;
freqz(b,a,f,8000); % Transition Band
figure(4)
f=2800:2:3200;
freqz(b,a,f,8000); % Transition Band
figure(5)
zplane(b,a)
```

Designing IIR Band Stop Filter:

```
[n,w]=buttord([1200/4000,2800/4000],[400/4000, 3200/4000],1,50);
[b,a]=butter(n,w,'stop');
figure(1)
freqz(b,a,128,8000)
[h,w]=freqz(b,a,128,8000);
figure(2)
plot(w,abs(h));
grid
figure(3)
f=600:2:1200;
freqz(b,a,f,8000); % Transition Band
figure(4)
f=2800:2:3200;
freqz(b,a,f,8000); % Transition Band
```

```
figure(5)
zplane(b,a);
```

Design an IIR filter to suppress frequencies of 5 Hz and 30 Hz from given signal.

Creating a signals ‘s’ with three sinusoidal components (at 5,15,30 Hz) and a time vector ‘t’ of 100 samples with a sampling rate of 100 Hz, and displaying it in the time domain.

The Matlab commands are shown below.

```
fs=100;
t=(1:100)/fs;
s=sin(2*pi*t*5)+sin(2*pi*t*15)+sin(2*pi*t*30);
plot(t,s)
grid
```

Now we design a filter to keep the 15 Hz sinusoid and eliminate the 5 and 30 Hz sinusoids. We use the functions ellipord and ellip to create an infinite impulse response (IIR) filter with a pass band from 10 to 20 Hz. The ellipord function requires the specification of pass band corner frequencies, minimum transition band frequencies near the pass band corner frequencies, the maximum pass band ripple in decibels (dB), and the minimum stop band attenuation in dB. In this example, we choose a transition frequency to be ±5 Hz near the pass band corners, with a maximum of 0.1 dB ripple in the pass band, and a minimum of 40 dB attenuation in the stop bands. We start by determining the minimum order (pass band and stop band frequencies are normalized to the Nyquist frequency):

```
wp1 = 10/50;
wp2 = 20/50;
ws1 = 5/50;
ws2 = 25/50;
wp = [Wp1 Wp2];
ws = [Ws1 Ws2];
rp = 0.1;
rs = 40;
[n,wn] = ellipord(wp,ws,rp,rs);
```

ellipord returns an order of 5, the minimum possible order for a low pass prototype that will meet the constraints upon transformation to a band pass filter. When we apply this order to the ellip function, internally we transform the low pass prototype to a band pass filter using the function lp2bp. This doubles the order, making n = 10. Next we use n, the order, and Wn, the pass band corner frequencies, to actually design the filter. We also use freqz, a tool for computing and displaying the frequency response of the descriptive transfer function. When called with no left-hand-side arguments (i.e., return values), freqz displays the magnitude and phase response of the filter normalized to the Nyquist frequency.

```
[b,a] = ellip(n,.1,40,w);
freqz(b,a,128,100)
[h,w]=freqz(b,a,128,100);
plot(w,abs(h));
grid
title(‘Normalized Magnitude Response’);
axis([0 50 0 1.2]);

figure(4)
sf=filter(b,a,s); % Time domain Response of the Filter
plot(t,sf)
grid
xlabel('Time (seconds)');
ylabel('Signal Amplitude');
title('Filtered Signal only 15 Hz frequency');
```