CLASS: TE E&TC SUB: DSP

EXP NO. -07

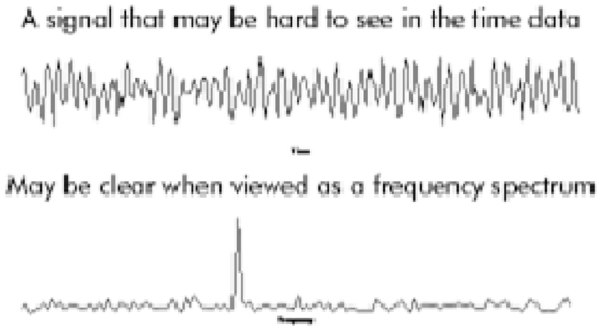
PERFORMANCE DATE	SUBMISSION DATE	SIGN	REMARK

TITLE : IIR Filter Design

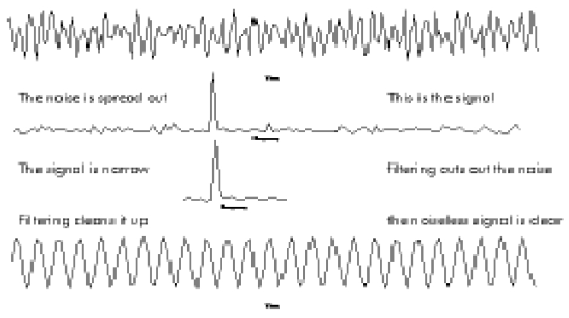
OBJECTIVE :

1. To design Low pass IIR Butterworth filter using Bilinear Z transform.
2. To design Low pass IIR Butterworth filter using Impulse Invariance Method

Filtering is a process of selecting, or suppressing, certain frequency components of a signal. Let's take A coffee filter which allows small particles to pass while trapping the larger grains. A digital filter does a similar thing, but with more subtlety. The digital filter allows to pass certain frequency components of the signal: in this it is similar to the coffee filter, with frequency standing in for particle size. But the digital filter can be more subtle than simply trapping or allowing through: it can attenuate, or suppress, each frequency components by a desired amount. This allows a digital filter to shape the frequency spectrum of the signal. Filtering is often, though not always, done to suppress noise. It depends on the signal's frequency spectrum being different from that of the noise:



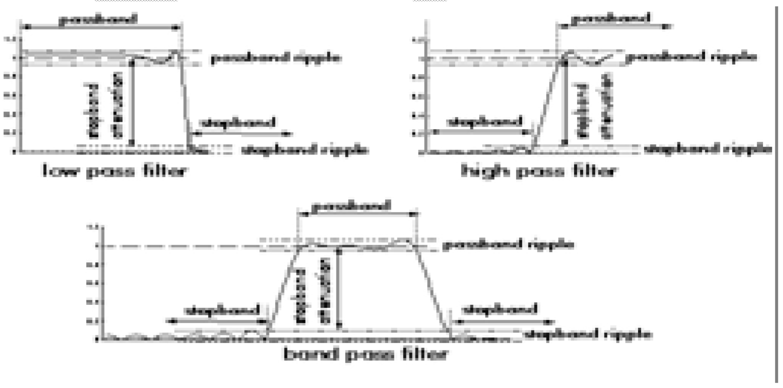
The diagram shows how a noisy sine wave viewed as a time domain signal cannot be clearly distinguished from the noise. But when viewed as a frequency spectrum, the sine wave shows as a single clear peak while the noise power is spread over a broad frequency spectrum. By selecting only frequency components that are present in the signal, the noise can be selectively suppressed:



The diagram shows how a noisy sine wave may be 'cleaned up' by selecting only a range of frequencies that include signal frequency components but exclude much of the noise:

- the noisy sine wave (shown as a time signal) contains narrow band signal plus broad band noise
- the frequency spectrum is modified by suppressing a range outside the signal's frequency components
- the resulting signal (shown in the time domain again) looks much cleaner

Digital filters can be more subtly specified than analogue filters, and so are specified in a different way:



Whereas analogue filters are specified in terms of their '3dB point' and their 'roll off', digital filters are specified in terms of desired attenuation, and permitted deviations from the desired value in their frequency response:

Pass Band

The band of frequency components that are allowed to pass

Stop Band

The band of frequency components that are suppressed

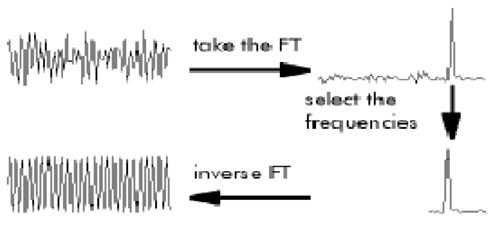
Pass band Ripple

The maximum amount by which attenuation in the pass band may deviate from nominal gain

Stop band Attenuation

The minimum amount by which frequency components in the stop band are attenuated

The pass band need not necessarily extend to the 3 dB point: for example, if pass band ripple is specified as 0.1 dB, then the pass band only extends to a point at which attenuation has increased to 0.1 dB. Between the pass band and the stop band lies a transition band where the filter's shape may be unspecified. Note that the stop band attenuation is formally specified as the attenuation to the top of the first side lobe of the filter's frequency response. Digital filters can also have an 'arbitrary response': meaning, the attenuation is specified at certain chosen frequencies, or for certain frequency bands. Filtering can be done directly in the frequency domain, by operating on the signal's frequency spectrum:

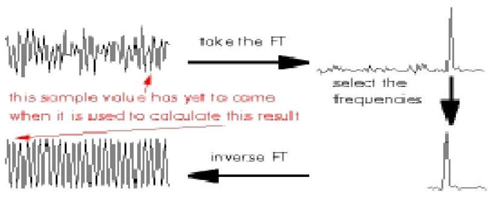


The diagram shows how a noisy sine wave can be cleaned up by operating directly upon its frequency spectrum to select only a range of frequencies that include signal frequency components but exclude much of the noise:

- The noisy sine wave (shown as a time signal) contains narrow band signal plus broad band noise

- The frequency spectrum is calculated
- The frequency spectrum is modified by suppressing a range outside the signal's frequency components
- The time domain signal is calculated from the frequency spectrum
- The resulting signal (shown in the time domain again) looks much cleaner

Filtering in the frequency domain is efficient, because every calculated sample of the filtered signal takes account of all the input samples. Filtering in the frequency domain is sometimes called 'a causal 'filtering because (at first sight) it violates the laws of cause and effect.


take the FT
select the frequencies
this sample value has yet to come when it is used to calculate this result
inverse FT

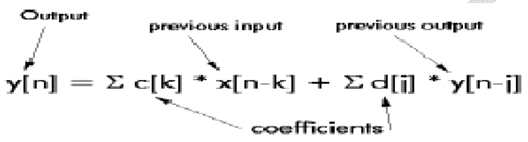
Because the frequency spectrum contains information about the whole of the signal - for all time values - samples early in the output take account of input values that are late in the signal, and so can be thought of as still to happen. The frequency domain filter 'looks ahead' to see what the signal is going to do, and so violates the laws of cause and effect. Of course this is nonsense - all it means is we delayed a little until the whole signal had been received before starting the filter calculation - so filtering directly in the frequency domain is perfectly permissible and in fact often the best method. It is often used in image processing.

There are good reasons why we might not be able to filter in the frequency domain:

- We might not be able to afford to wait for future samples - often, we need to deliver the next output as quickly as possible, usually before the next input is received
- We might not have enough computational power to calculate the Fourier transform
- We might have to calculate on a continuous stream of samples without the luxury of being able to chop the signal into convenient lumps for the Fourier transform.

- We might not be able to join the edges of the signals smoothly after transforming back from the frequency domain

None of the above reasons should make us ignore the possibility of frequency domain filtering, which is very often the best method. It is often used in image processing, or certain types of experiment where the data necessarily comes in bursts, such as NMR or infra red spectroscopy. Output from a digital filter is made up from previous inputs and previous outputs, using the operation of convolution:



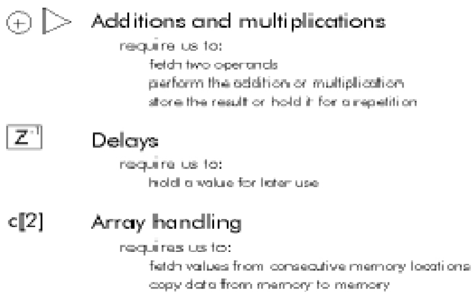
Two convolutions are involved: one with the previous inputs, and one with the previous outputs. In each case the convolving function is called the filter coefficients. The filter can be drawn as a block diagram: Two convolutions are involved: one with the previous inputs, and one with the previous outputs. In each case the convolving function is called the filter coefficients. The filter can be drawn as a block diagram:

Types of Filter

Analog Filter & Digital Filter

	Analog Filter	Digital Filter
1	Analog Filter processes analog inputs and generates analog outputs	A digital filter processes and generates digital data
2	Analog filter are constructed from active or passive electronic components	A digital filter consists of elements like, adder, multiplier and delay unit.
3	Analog filter is described by a differential equation	Digital filter is described by a difference equation
4	The frequency response of an analog filter can be modified by changing the components	The frequency response can be changed by changing the filter coefficients.

The filter diagram can show what hardware elements will be required when implementing the filter:



Additions and multiplications
require us to:
fetch two operands
perform the addition or multiplication
store the result or hold it for a repetition

z^{-1} Delays
require us to:
hold a value for later use

c[2] Array handling
requires us to:
fetch values from consecutive memory locations
copy data from memory to memory

The design of a Digital Filter is usually specified in terms of the characteristics of the signals to be passed through the filter. In many cases, the signals are described in terms of their frequency content. Based upon their Frequency Selective nature, Filter can be classified as-

Ideal filter types

Based on Frequency Response

Low pass

Attenuates frequencies above cut off frequency, letting frequencies below cut off (*f*C) through, see Figure 1.

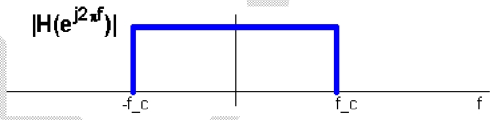


Figure: An ideal Low Pass filter.

High pass

High pass filters stops low frequencies, letting higher frequencies through, see Figure 2.

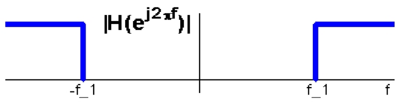


Figure: An ideal High Pass filter.

Band pass

Letting through only frequencies in a certain range, see Figure 3.

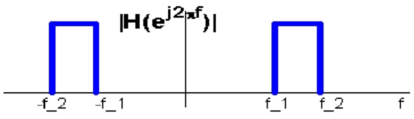


Figure: An ideal band pass filter.

Band stop

Stopping frequencies in a certain range , see Figure 4.

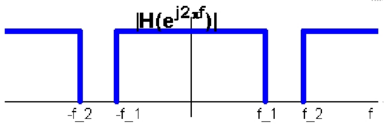


Figure: An ideal band stop filter

Depending upon the Impulse Response of the Filters, these can be classified as-

IIR Filter – Infinite Impulse Response Filter

FIR Filter – Finite Impulse Response Filter

The IIR filter can realize both the poles and zeroes of a system because it has a rational transfer function, described by polynomials in z in both the numerator and the denominator:

$$H(z)\frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=1}^{N} a_k Z^{-k}}$$

The difference equation for such a system is described by the following:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \ + \ \sum_{k=1}^{N} a_k y(n-k)$$

Where, M and N are order of the two polynomials

bk and ak are the filter coefficients.

Designing a Digital Filter is **finding out Coefficients** in Numerator and Denominator. IIR filters can be expanded as infinite impulse response filters. In designing IIR filters, cutoff frequencies of the filters should be mentioned. The order of the filter can be estimated using butter worth polynomial. That’s why the filters are named as butter worth filters. Filter coefficients can be found and the response can be plotted.

Methods of Designing IIR Filter

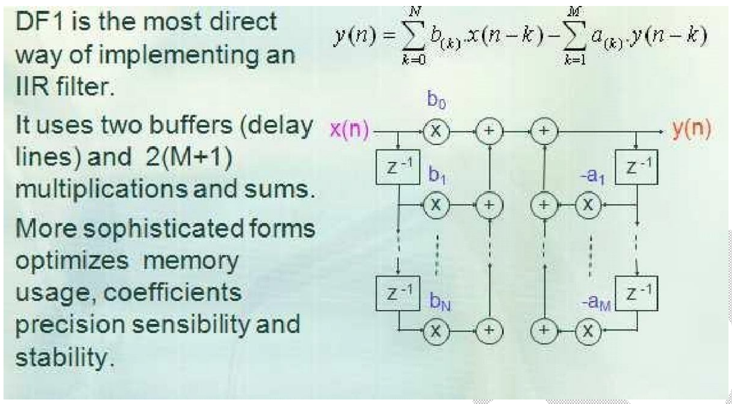
	Impulse Invariance Method	Bilinear Transformation Method
1	Poles are transferred by using equation $( ) = \frac{Ck}{1 - e^{PkT} \; Z^{-1}}$ $=1$	Poles are transferred by using equation $^{-1}$ $-$
2	Mapping is Many to One	Mapping is One to One
3	Aliasing Effect is Present	Aliasing Effect is not Present
4	It is not suitable to design high pass Filter and band Reject Filter	High Pass Filter and Band Reject can be designed.
5	Only poles of the system can be mapped	Poles as well as Zeros can be mapped
6	No Frequency Warping Effect	Frequency Warping Effect is Present

Digital Filter Structures:

For realization of IIR filters following structures are used

1. Direct form structure
2. Cascade form structure
3. Parallel form structure

Direct form structures are classified in two types:



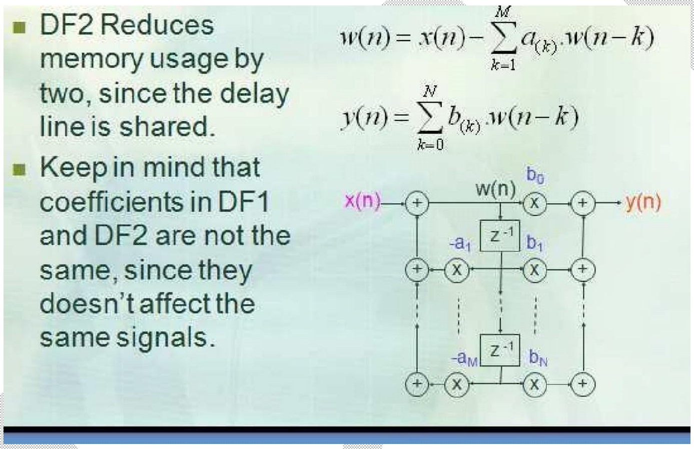
DF1 is the most direct way of implementing an IIR filter.

It uses two buffers (delay lines) and 2(M+1) multiplications and sums.

More sophisticated forms optimizes memory usage, coefficients precision sensibility and stability.

$$y(n)=\sum_{k=0}^{N} b_{(k)}.x(n-k)-\sum_{k=1}^{M} a_{(k)}.y(n-k)$$

Direct form I Structure

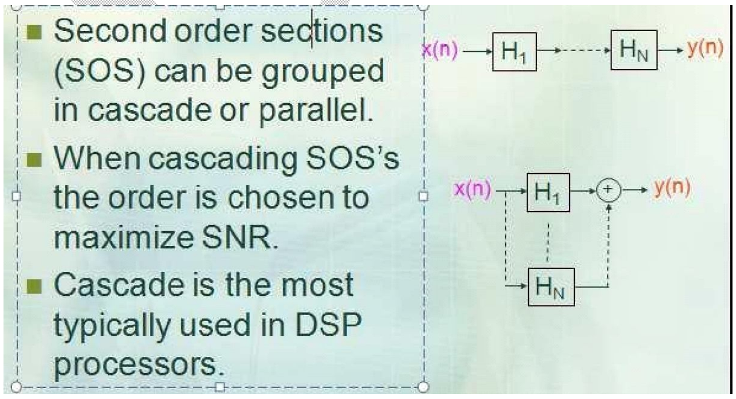


- DF2 Reduces memory usage by two, since the delay line is shared.
- Keep in mind that coefficients in DF1 and DF2 are not the same, since they doesn't affect the same signals.

$$w(n)=x(n)-\sum_{k=1}^{M} a_{(k)}.w(n-k)$$

$$y(n)=\sum_{k=0}^{N} b_{(k)}.w(n-k)$$

Direct form-II structure



- Second order sections (SOS) can be grouped in cascade or parallel.
- When cascading SOS's the order is chosen to maximize SNR.
- Cascade is the most typically used in DSP processors.

Cascade form structure

ALGORITHM:

A. To design Low pass IIR Butterworth filter using Impulse Invariance

B. To design Low pass IIR Butterworth filter using Bilinear Z transform.

CONCLUSION:

REFERENCES:

1. Digital Signal Processing – A Computer Based Approach, “S.K. Mitra”

2. Digital Signal Processing, “Ifeacher & Jervis”

3. Digital Signal Processing, “Proakis”