Chapter 9 Analysis and Design of Digital Filter

Introduction

What designs have we done in this course?

What do we mean by filters here?

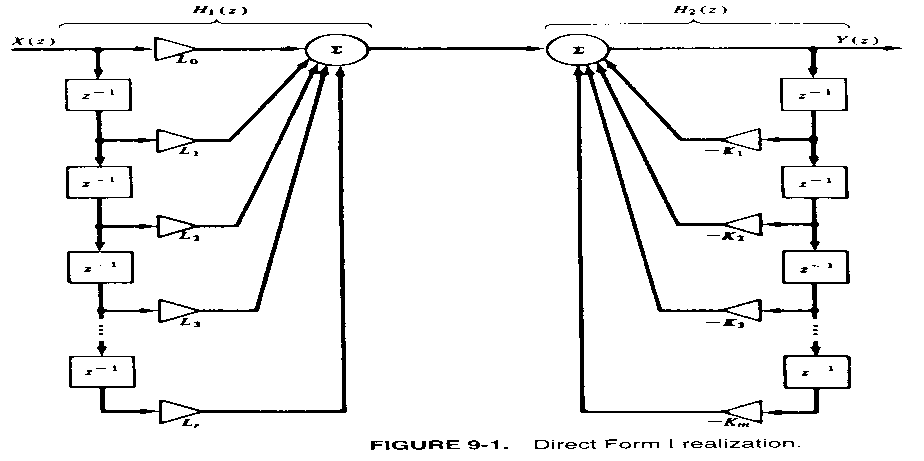
What do we mean by filters design?

Given specifications (requirements) => *H(z)*

Let’s see how we can implement a digital filter (processor) if its *H(z)* is given?

Structures of Digital Processors

1. Direct-Form Realization

The function is realized!

What’s the issue here?

Count how many memory elements we need!

Can we reduce this number?

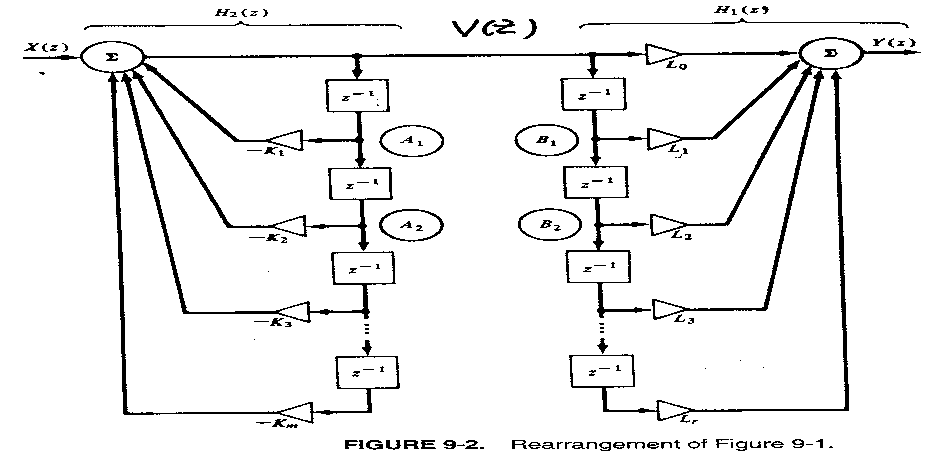
If we can, what is the concern?





Denote 

Implement *H2(z)* and then *H1(z)* ?



Why *H2* is implemented?

(1)(2)

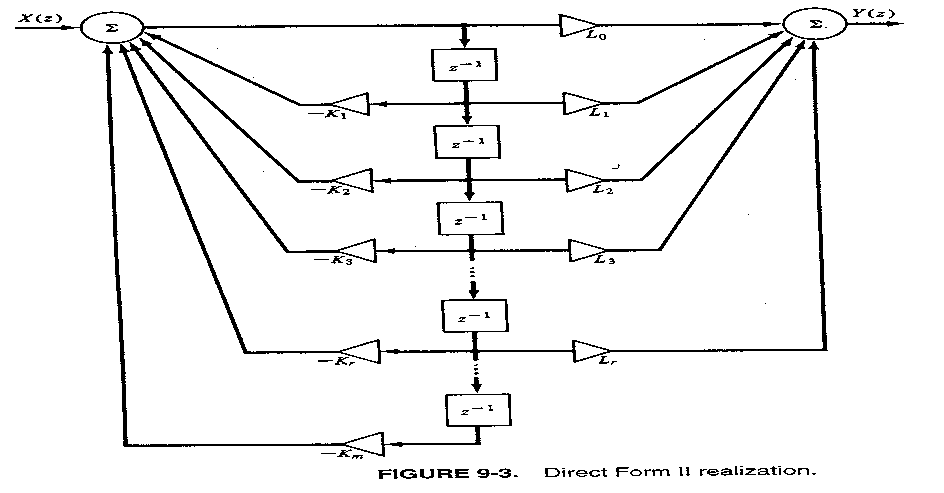


*H*2 is realized!

Can you tell why *H*1 is realized?

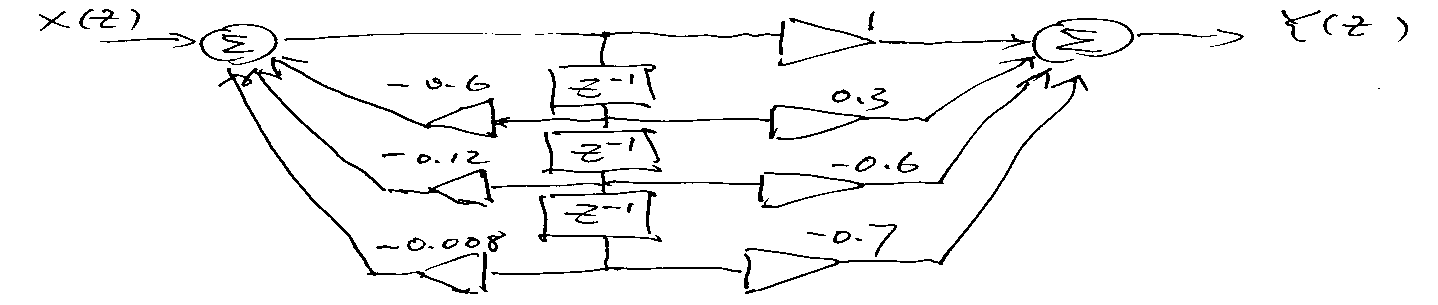
What can we see from this realization? Signals at  and : always the same

🡺 Direct Form II Realization



Example 

Solution: 

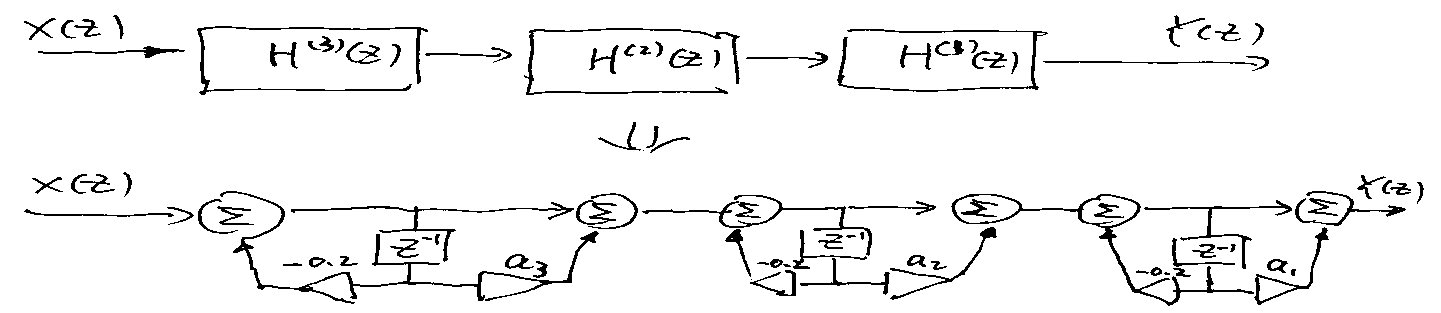


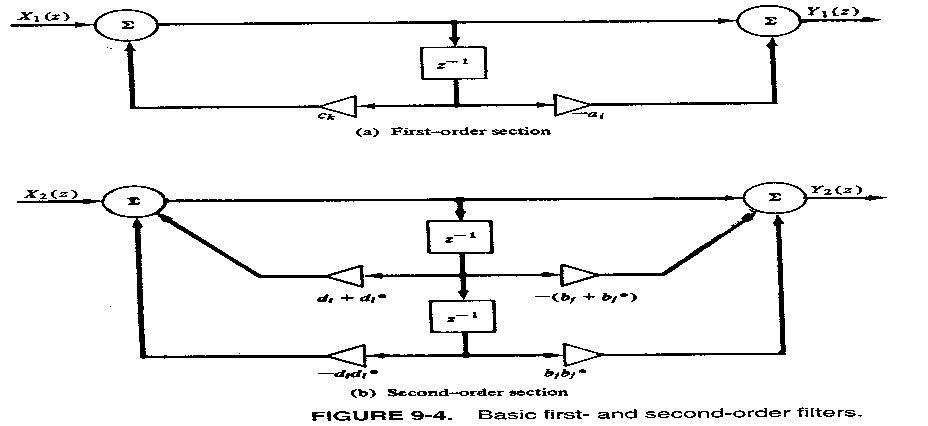
Important: H(z)=B(z)/A(z) (1) A: 1+…. (2) Coefficients in A: in the feedback channel

2. Cascade Realization

Factorize 





 General Form



Apply Direct II for each!

3. Parallel Realization (Simple Poles)



Example 9-1

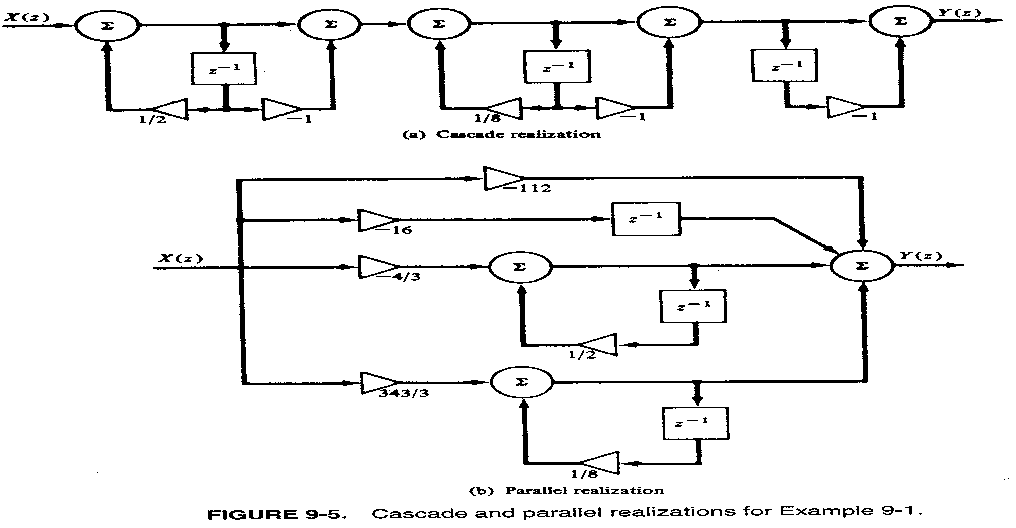


cascade and parallel realization!

Solution:

1. Cascade:





1. Parallel



In order to make deg(num)<deg(den)

Partial-Fraction Expansion for *s*









*z =* anything other than *0, ½, 1/8, 1🡺 B = -*112

For example, *z*=2



Example -2: System having a complex conjugate pole pair at 

🡺Transfer function





How do we calculate the amplitude response

 and  ?

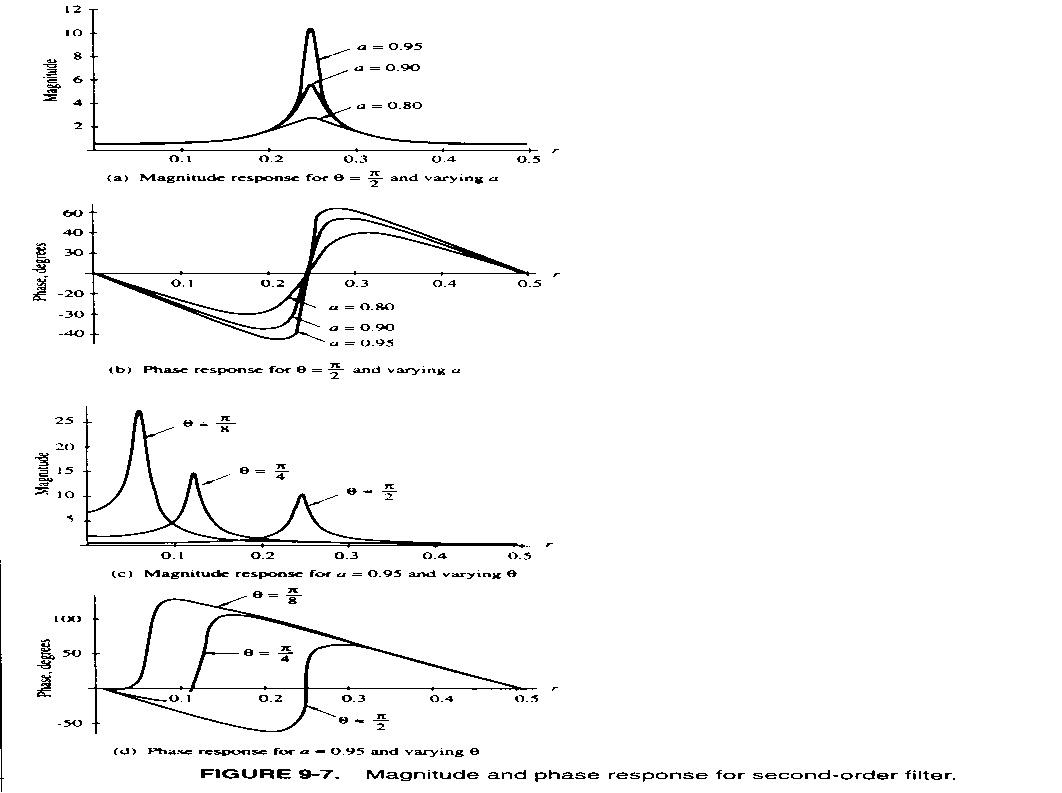
How the distance between the pole and the unit circle influence *|H|*  and *∠H* ?

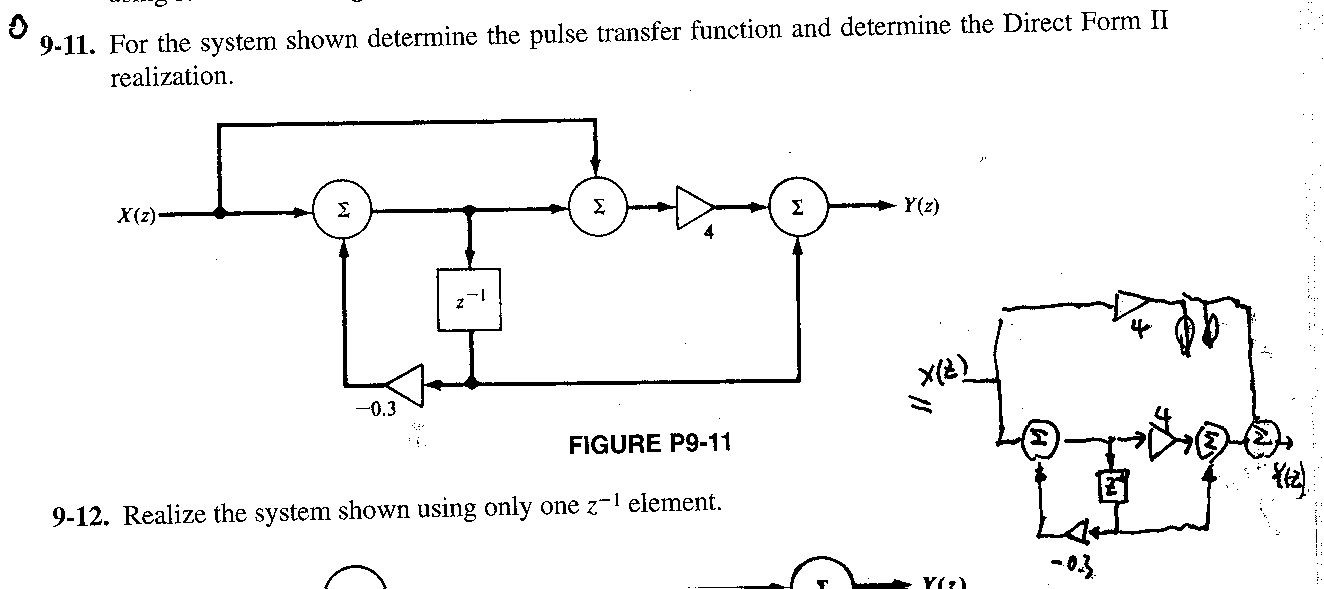
How the distance between the pole and the unit circle influence ?

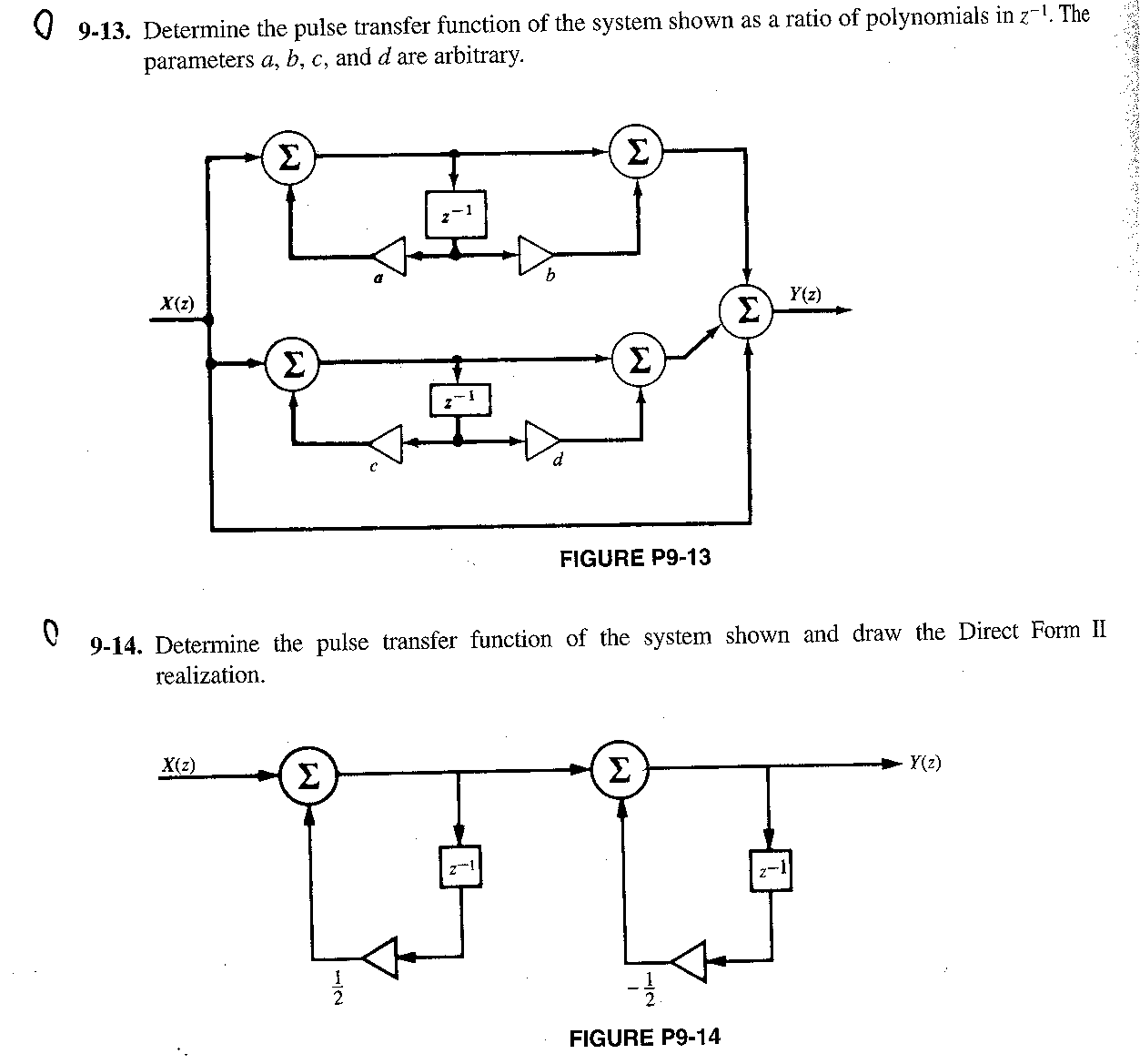


How the pole angle  influence  and ?

See Fig. 9-7







Discrete-Time Integration

A method of Discrete-time system Design: Approximate continuous-time system

Integrator  ← a simple system

↑ ↑ system input

Output

Discrete-time approximation of this system: discrete-time Integrator

1. Rectangular Integration



change of *y* from *t0* to *t* :

*t= nT, t0 = nT- T*



Constant 

*T* small enough => 



*y(nT)* : System output

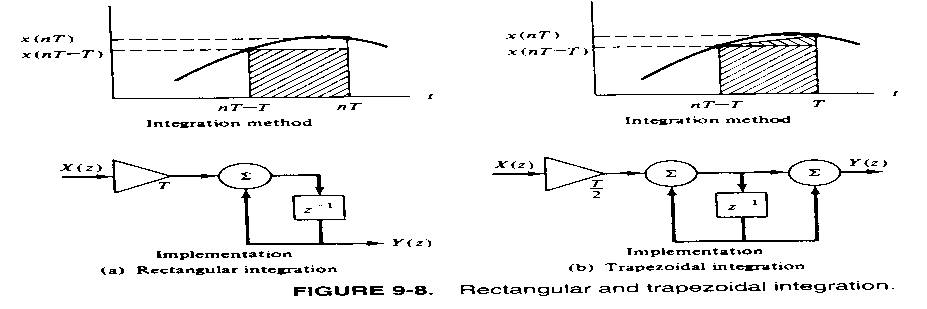
*x(nT)* : System input , to be integrated

Constant



A discrete-time integrator: rectangular integrator





Constants

1. Trapezoidal Integration





or 



1. Frequency Characteristics
   1. Rectangular Integrator



Frequency Response 

Or  

Amplitude Response



Phase Response



* 1. Trapezoidal Integrator



Frequency Response



Amplitude: 

Phase:

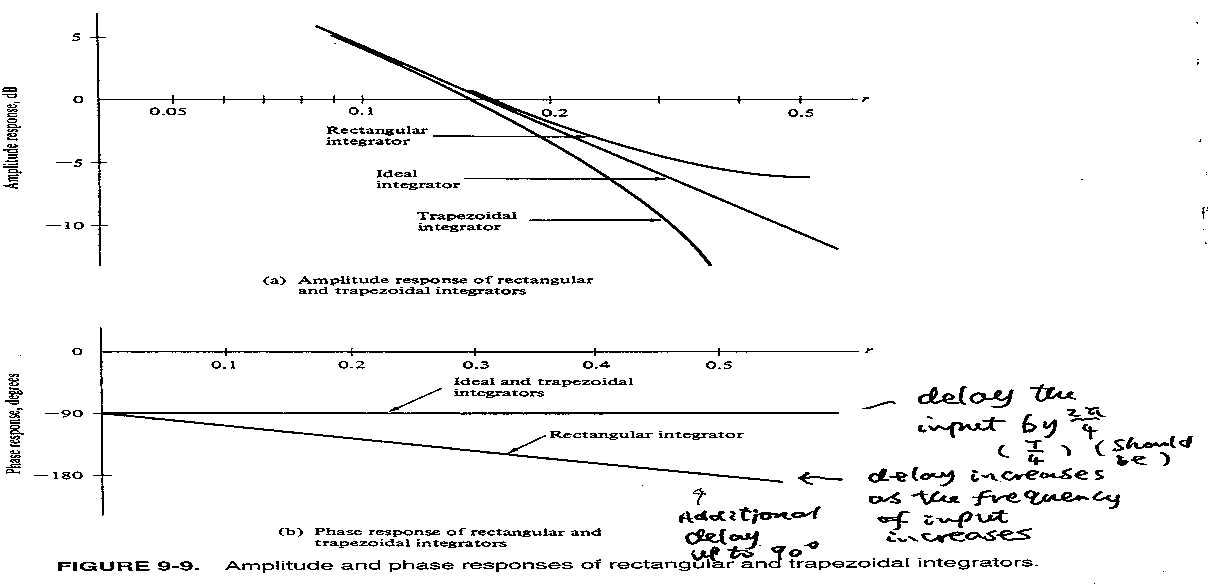


* 1. Versus Ideal Integrator

Ideal (continuous-time ) Integrator



when *T=*1 second (Different plots and relationships will result if *T* is different.)



* Low Frequency Range 

(Frequency of the input is much lower than the sampling frequency:

It should be!)



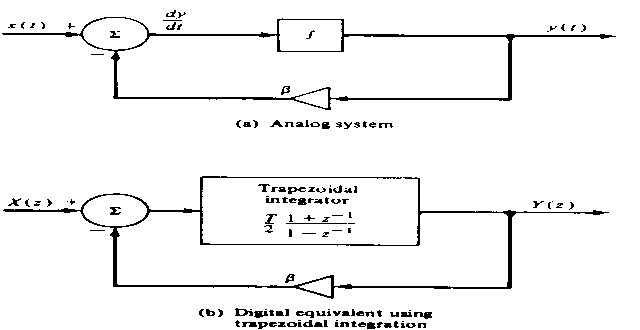


* High Frequency: Large error (should be)

Example 9-4 Differential equation (system)



Determine a digital equivalent.



Solution

(1) Block Diagram of the original system

(2) An equivalent

(3) Transfer Function Derivation

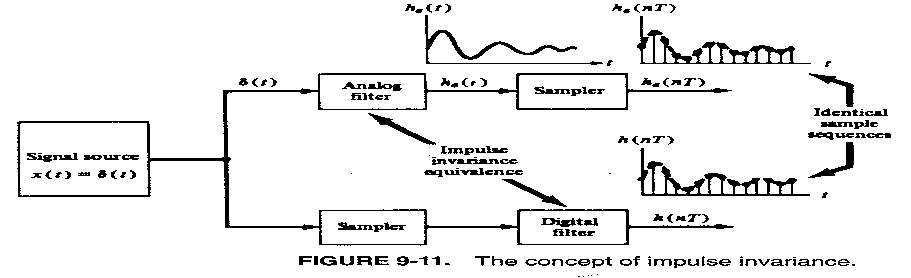


**Infinite Impulse Response (IIR) Filter Design**

(Given *H(s) → Hd(z)* )

**A Synthesis in the Time-Domain: Invariant Design**

1. Impulse – Invariant Design
2. Design Principle



1. Illustration of Design Mechanism (Not General Case)

Assume:

1. Given analog filter (Transfer Function)

 (a special case)

(2) Sampling Period *T* (sample *ha(t)* to generate *ha(nT)*)

Derivation:

1. Impulse Response of analog filter



1. *ha(nT)*: sampled impulse response of analog filter



1. z-transform of *ha(nT)*

Sampled impulse response of analog filter



1. Impulse-Invariant Design Principle



↑

Digital filter is so designed that its impulse response *h(nT)*

equals the sampled impulse response of the analog filter *ha(nT)*

Hence, digital filter must be designed such that



z-transfer function  of the digital filter. Of course,

the z-transfer function of its impulse response.

(5)  (scaling)

=>

(3) Characteristics

(1)  when *T→* 0

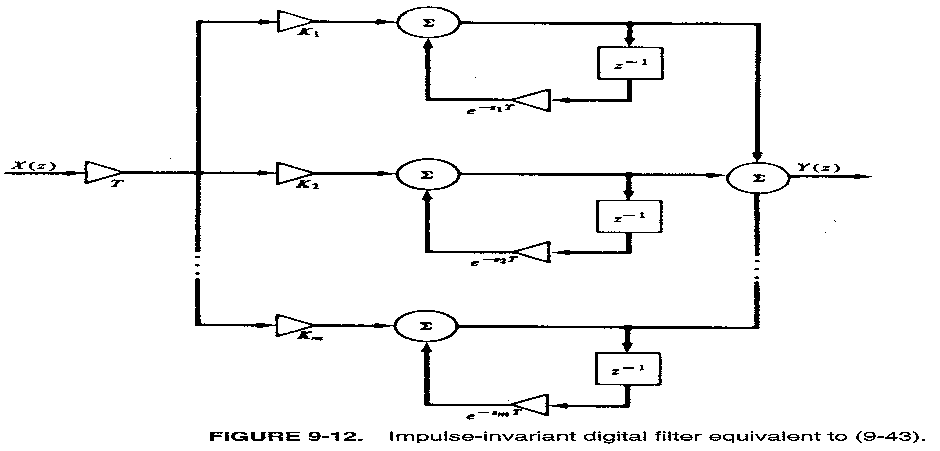
↑

frequency response of digital filter

1. 
2. Design: Optimized for *T =* 0

Not for *T* ≠ 0 (practical case) (due to the design principle)

(4) Realization: Parallel

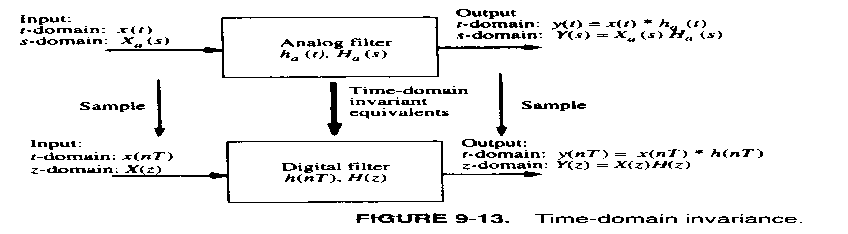


(5) Design Example 

Solution: 



1. General Time – Invariant Synthesis
2. Design Principle



1. Derivation

Given: *Ha(s)* transfer function of analog filter

*Xa(s)* Lapalce transform of input signal of analog Filter

*T* sampling period

Find *H(z)* z-transfer function of digital filter

1. Response of analog filter *xa(t)*



(2) *ya(nT)* sampled signal of analog filter output



(3) z-transform of *ya(nT)*



(4) Time – invariant Design Principle



↑

Digital filter is so designed

that its output equals the sampled

output of the analog filter

Incorporate the scaling :



↑

*T*

z-transfer function of digital filter

(5) Design Equation



special case *X*(*z*)*=*1*, Xa*(*s*) *=* 1 (impulse)

=>

(6) Design procedure

A: Find  (output of analog filter)

B: Find 

C: Find 

D: 

Example 9-5 

Find digital filter H(z) by impulse - invariance.

Solution of design:

1. Find 





1. Find 



1. Find 

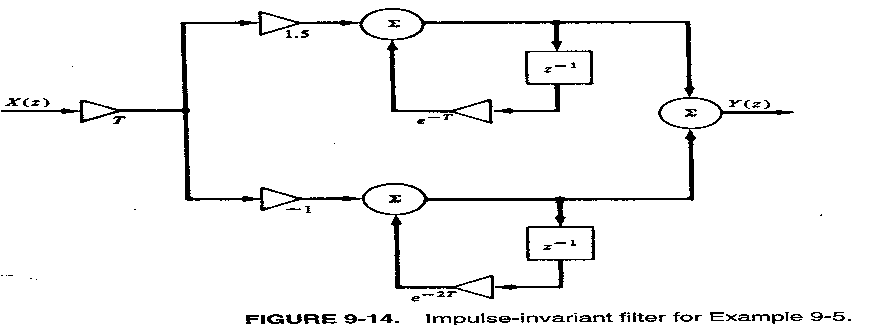


1. Find z-transfer function of the digital filter



use G = T



1. Implementation

Characteristics

(1) Frequency Response equations: analog and digital

Analog : 

Digital : 

(2) dc response comparison ()

Analog: 

Digital: 

Varying with T (should be)

 , 



 for example 

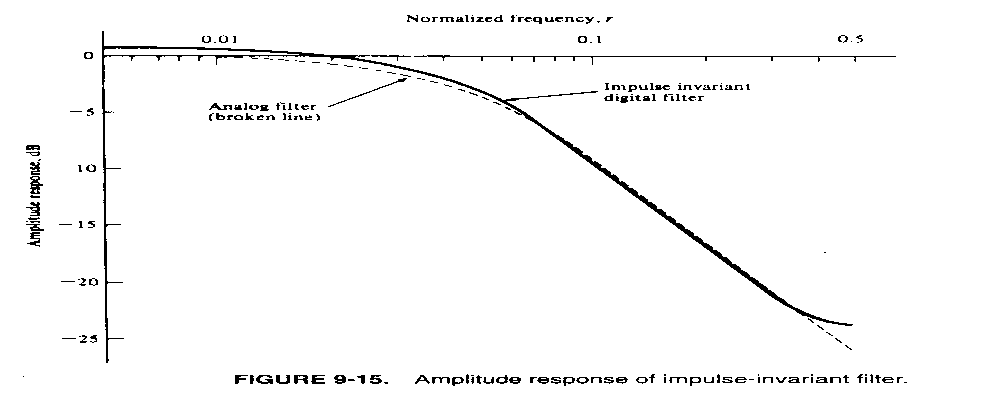
, 

 good enough

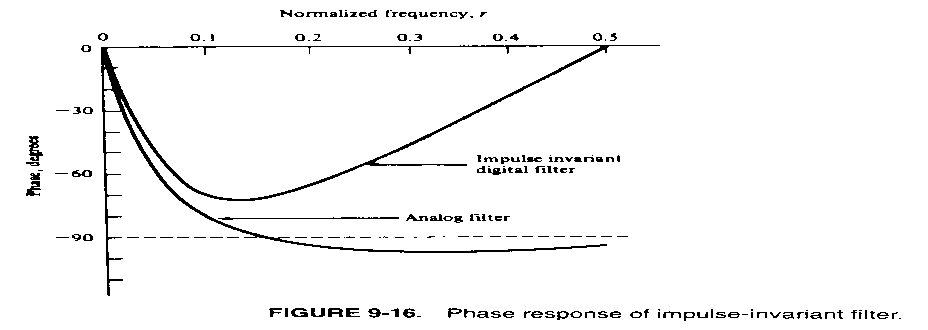
(3)  versus  : 

Using normalized frequency 





(4)  versus 



(5) Gain adjustment when 

 => frequency response inequality

adjust G =>  at a special 

for example 

If *G = T =* 0.3142 => 

If selecting *G = T/*1.0745 => 

1. Step – invariance synthesis



Example 9-6 . Find its step-invariant equivalent.

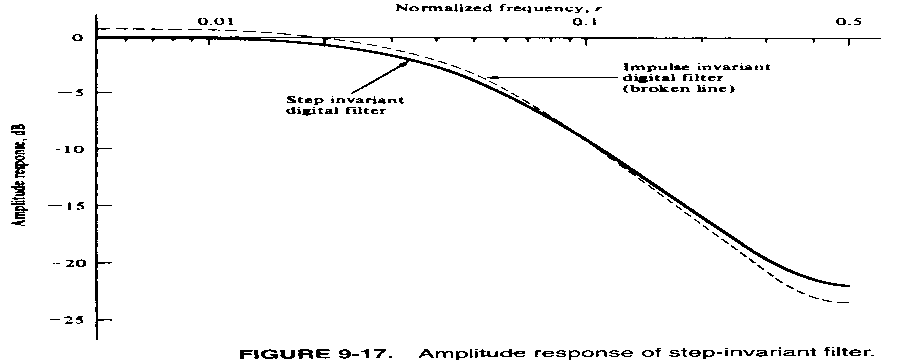
Solution of Design







Comparison with impulse-invariant equivalent.



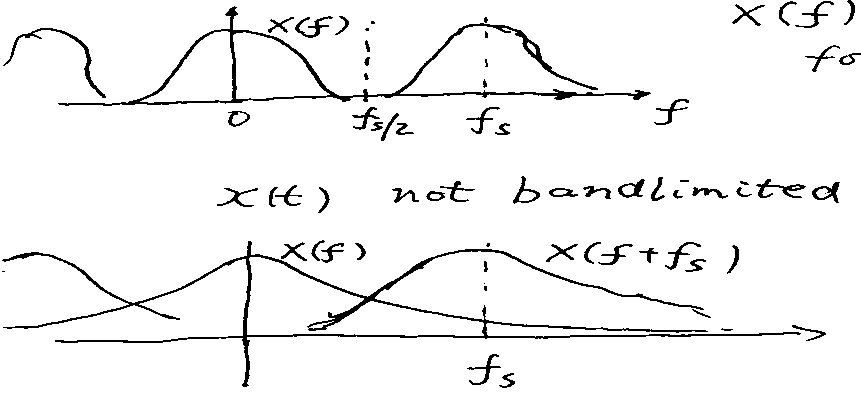
**Design in the Frequency Domain --- The Bilinear z-transform**

1. Motivation (problem in Time Domain Design)

Introduced by sampling, undesired!

*x*(*t*) bandlimited (





for 



for 

Consider digital equivalent of an analogy filter *Ha*(*f*): )

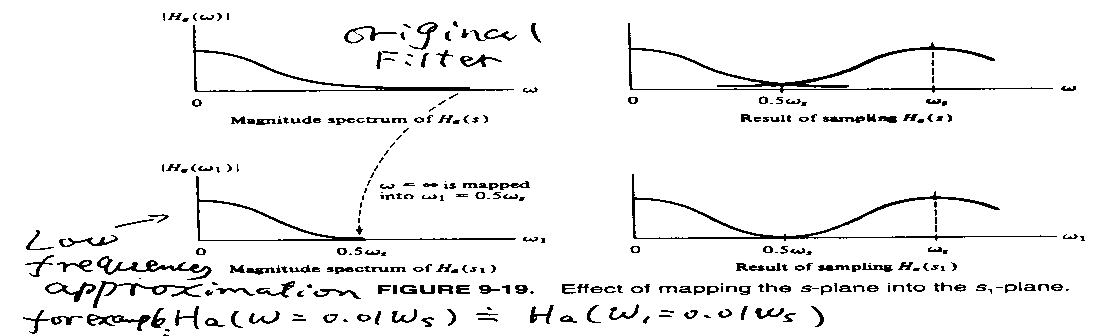
*Ha*(*f*): bandlimited => can find a *Hd*(*z*)

*Ha*(*f*): not bandlimited => can not find a *Hd*(*z*) Such that 

1. Proposal: from  axis to  axis

 (: given sampling frequency)

(*s* plane to *s*1 plane)



Observations: (1) Good accuracy in low frequencies

(2) Poor accuracy in high frequencies

(3) 100% Accuracy at 

a given specific number such that 0.01

Is it okay to have poor accuracy in high frequencies? Yes! Input is bandlimited!

What do we mean by good, poor and 100% accuracy?

Assume (1) (originally given analogy filter)

(2) The transform is 

Then, is a function of . Denote .

Good accuracy:



Poor Accuracy



100% Accuracy (Equal)



Is  bandlimited? That is, can we find a  such that =0?

Yes, . 🡺 We have no problem to find a digital equivalent

 without aliasing!

Let’s use as a number (for example 0.2) representing any low frequency,

Then, because  is a good approximation of ,

 should be a good approximation.

A digital filter can thus be designed for an analogy filter  which is not bandlimted!

Two Step Design Procedure:

Given: analogy filter 

(1) Find an bandlimited analogy approximation () for 

(2) Design a digital equivalent  for the bandlimited filter .

Because of the relationship between () for ,

 is also digital equivalent of .

The overlapping (aliasing) problem is avoided!

The designed digital filter can approximate  (for  and  take the

same value) at low frequency.

3.  axis to  axis (*s* plane to *s1* plane) transformation

Requirement :  ( is given sampling frequency.)

Proposed transformation :

Variable in  domain



Variable in  domain

Effect of C:

Constant

We want the transformation map

 (for example, ) to  

=> 

i.e. when the sampling period *T* is given, *C* is the only parameter

which determines what  will be mapped into  axis with the

same value.

Example: 

 not bandlimited

If we want to map  to 

1. 



Hence, for any given *T* or 



is bandlimited as a function of  by 

 when  at low frequencies.

Further 

Exactly holds!

How to select  or sampling frequency  at which ?

(1)  should be small?

why?  ,

(The accuracy should be good at low frequencies)

(2) When  is given or determined by application,  should be large

enough such that  to ensure the accuracy in the frequency

range including 

When 

 since  for small *x*.

4. Design of Digital Filter using bilinear z-transform

1. A procedure: (1) 

(not bandlimited, (bandlimited,

original analog) analog)

or 

(2) 

(Transfer replace  by *z*

function of

digital filter)

\* Question: Can we directly obtain *Hd(z)* from *Ha(s)* ? Yes! (But how?)

1. Bilinear z-transform

Preparation : (1) 

(2) 

Hence, 

Replace  by *s* ,  by *s*1 ()



Replace  for digital filter

 direct transformation from *s* to *z* (bypass *s*1)

Bilinear z – transformation

Example 

Digital Filter



C: only undetermined parameter in the digital filter.

To determine C: (1) 

(2)  (related to the frequency range of interest)

Example 9-7 

 : break frequency

Take 

Consider 



🡺 C determined => *Hd*(*z*) determined

🡺 

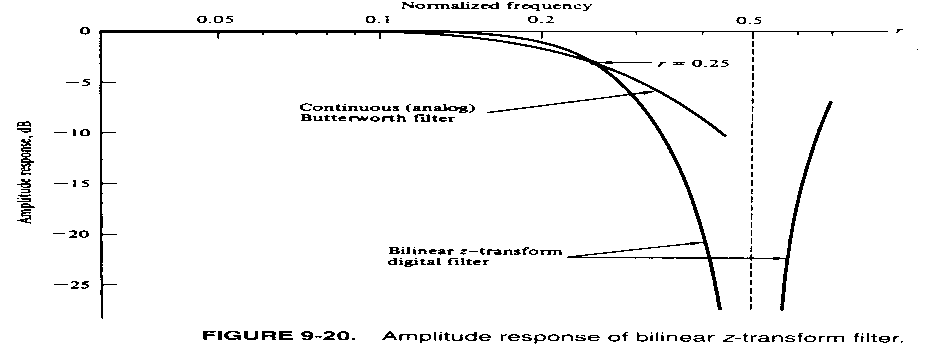
🡺 , , 

To compare the frequency response with the original analog filter *Ha* :



( replace *s* by )

🡺 , 



Too low *fs*  => poor accuracy in *fc*.

**Bilinear z-Transform Bandpass Filter**

1. Construction Mechanism
2. From an analog low-pass filter *Ha(s)*

to analog bandpass filter 

i.e., replace *s* by  to form a bandpass filter

In the low pass filter

For example  low-pass

 band-pass

Why? Original low-pass





Low => High Gain

High => Low Gain

After Replacement



high  => high => low gain

low  => high => low gain

1. From analog to digital

Replace *s* in  by 

🡺 

for example



digital bandpass filter

2. Bilinear z-transform equation

Analog Low-pass Bandpass (analog)

s 



Direct Transformation

s (in low-pass)



s (in low-pass)



with 

3. How to select ( ) for bandpass filter

(design)

Important parameters of bandpass

1. center frequency 
2.  upper critical frequency
3.  low critical frequency

Selection of  for bandpass: 

Design of ( )

We want , Also want 

one parameter C => impossible

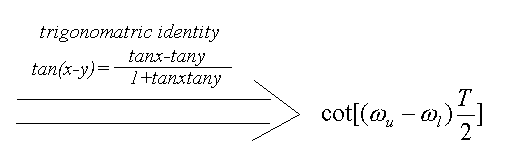
1. solution 
2. bandwidth 

Hence, A and B can be determined to perform the transform.

4. Convenient design equation



why no C?









In low-pass

Example :

5. In the normalized frequency

Reference frequency: sampling frequency 

=> , 

=>

s => 

In low-pass

Example 9-9 Lowpass 

Transfer function of bandpass digital filter

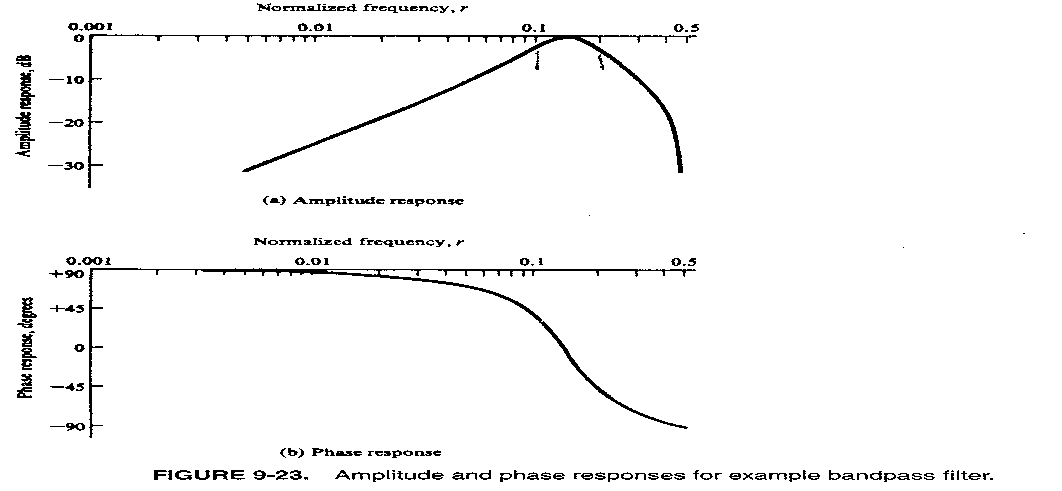


A and B? Determined by design requirements.

 sampling frequency *fs =* 5000*Hz*

1. 
2. 
3. 

, , 



**Butterworth Filter Design**

In the previous filter tutorials we looked at simple first-order type low and high pass filters that contain only one single resistor and a single reactive component (a capacitor) within their RC filter circuit design.

In applications that use filters to shape the frequency spectrum of a signal such as in communications or control systems, the shape or width of the roll-off also called the “transition band”, for a simple first-order filter may be too long or wide and so active filters designed with more than one “order” are required. These types of filters are commonly known as “High-order” or “nth-order” filters.

The complexity or filter type is defined by the filters “order”, and which is dependant upon the number of reactive components such as capacitors or inductors within its design. We also know that the rate of roll-off and therefore the width of the transition band, depends upon the order number of the filter and that for a simple first-order filter it has a standard roll-off rate of 20dB/decade or 6dB/octave.

Then, for a filter that has an nth number order, it will have a subsequent roll-off rate of 20n dB/decade or 6n dB/octave. So a first-order filter has a roll-off rate of 20dB/decade (6dB/octave), a second-order filter has a roll-off rate of 40dB/decade (12dB/octave), and a fourth-order filter has a roll-off rate of 80dB/decade (24dB/octave), etc, etc.

High-order filters, such as third, fourth, and fifth-order are usually formed by cascading together single first-order and second-order filters.

For example, two second-order low pass filters can be cascaded together to produce a fourth-order low pass filter, and so on. Although there is no limit to the order of the filter that can be formed, as the order increases so does its size and cost, also its accuracy declines.

**Decades and Octaves**

One final comment about *Decades* and *Octaves*. On the frequency scale, a **Decade** is a tenfold increase (multiply by 10) or tenfold decrease (divide by 10). For example, 2 to 20Hz represents one decade, whereas 50 to 5000Hz represents two decades (50 to 500Hz and then 500 to 5000Hz).

An **Octave** is a doubling (multiply by 2) or halving (divide by 2) of the frequency scale. For example, 10 to 20Hz represents one octave, while 2 to 16Hz is three octaves (2 to 4, 4 to 8 and finally 8 to 16Hz) doubling the frequency each time. Either way, *Logarithmic* scales are used extensively in the frequency domain to denote a frequency value when working with amplifiers and filters so it is important to understand them.

Since the frequency determining resistors are all equal, and as are the frequency determining capacitors, the cut-off or corner frequency ( ƒC ) for either a first, second, third or even a fourth-order filter must also be equal and is found by using our now old familiar equation:



As with the first and second-order filters, the third and fourth-order high pass filters are formed by simply interchanging the positions of the frequency determining components (resistors and capacitors) in the equivalent low pass filter. High-order filters can be designed by following the procedures we saw previously in the Low Pass filter and High Pass filter tutorials. However, the overall gain of high-order filters is **fixed** because all the frequency determining components are equal.

**Filter Approximations**

So far we have looked at a low and high pass first-order filter circuits, their resultant frequency and phase responses. An ideal filter would give us specifications of maximum pass band gain and flatness, minimum stop band attenuation and also a very steep pass band to stop band roll-off (the transition band) and it is therefore apparent that a large number of network responses would satisfy these requirements.

Not surprisingly then that there are a number of “approximation functions” in linear analogue filter design that use a mathematical approach to best approximate the transfer function we require for the filters design.

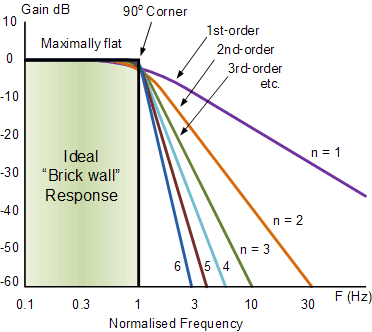
Such designs are known as **Elliptical**, **Butterworth**, **Chebyshev**, **Bessel**, **Cauer** as well as many others. Of these five “classic” linear analogue filter approximation functions only the **Butterworth Filter** and especially the *low pass Butterworth filter* design will be considered here as its the most commonly used function.

**Low Pass Butterworth Filter Design**

The frequency response of the **Butterworth Filter** approximation function is also often referred to as “maximally flat” (no ripples) response because the pass band is designed to have a frequency response which is as flat as mathematically possible from 0Hz (DC) until the cut-off frequency at -3dB with no ripples. Higher frequencies beyond the cut-off point rolls-off down to zero in the stop band at 20dB/decade or 6dB/octave. This is because it has a “quality factor”, “Q” of just 0.707.

However, one main disadvantage of the Butterworth filter is that it achieves this pass band flatness at the expense of a wide transition band as the filter changes from the pass band to the stop band. It also has poor phase characteristics as well. The ideal frequency response, referred to as a “brick wall” filter, and the standard Butterworth approximations, for different filter orders are given below.

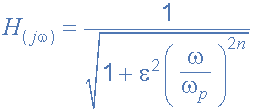
**Ideal Frequency Response for a Butterworth Filter**



Note that the higher the Butterworth filter order, the higher the number of cascaded stages there are within the filter design, and the closer the filter becomes to the ideal “brick wall” response.

In practice however, Butterworth’s ideal frequency response is unattainable as it produces excessive passband ripple.

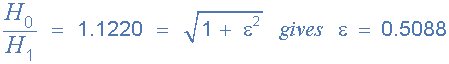
Where the generalised equation representing a “nth” Order Butterworth filter, the frequency response is given as:



Where: n represents the filter order, Omega ω is equal to 2πƒ and Epsilon ε is the maximum pass band gain, (Amax). If Amax is defined at a frequency equal to the cut-off -3dB corner point (ƒc), ε will then be equal to one and therefore ε2 will also be one. However, if you now wish to define Amax at a different voltage gain value, for example 1dB, or 1.1220 (1dB = 20logAmax) then the new value of epsilon, ε is found by:

|  |  |
| --- | --- |
| pass band gain formula | * Where: * H0 = the Maximum Pass band Gain, Amax. * H1 = the Minimum Pass band Gain. |

Transpose the equation to give:



The **Frequency Response** of a filter can be defined mathematically by its **Transfer Function** with the standard Voltage Transfer Function H(jω) written as:

|  |  |
| --- | --- |
| voltage transfer function | * Where: * Vout = the output signal voltage. * Vin  = the input signal voltage. * j   = to the square root of -1 (√-1) * ω  = the radian frequency (2πƒ) |

Note: ( jω ) can also be written as ( s ) to denote the **S-domain.** and the resultant transfer function for a second-order low pass filter is given as:

s-domain transfer function

**Normalised Low Pass Butterworth Filter Polynomials**

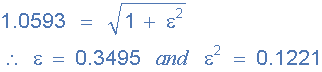
To help in the design of his low pass filters, Butterworth produced standard tables of normalised second-order low pass polynomials given the values of coefficient that correspond to a cut-off corner frequency of 1 radian/sec.

|  |  |
| --- | --- |
| **n** | **Normalised Denominator Polynomials in Factored Form** |
| 1 | (1+s) |
| 2 | (1+1.414s+s2) |
| 3 | (1+s)(1+s+s2) |
| 4 | (1+0.765s+s2)(1+1.848s+s2) |
| 5 | (1+s)(1+0.618s+s2)(1+1.618s+s2) |
| 6 | (1+0.518s+s2)(1+1.414s+s2)(1+1.932s+s2) |
| 7 | (1+s)(1+0.445s+s2)(1+1.247s+s2)(1+1.802s+s2) |
| 8 | (1+0.390s+s2)(1+1.111s+s2)(1+1.663s+s2)(1+1.962s+s2) |
| 9 | (1+s)(1+0.347s+s2)(1+s+s2)(1+1.532s+s2)(1+1.879s+s2) |
| 10 | (1+0.313s+s2)(1+0.908s+s2)(1+1.414s+s2)(1+1.782s+s2)(1+1.975s+s2) |

**Filter Design – Butterworth Low Pass**

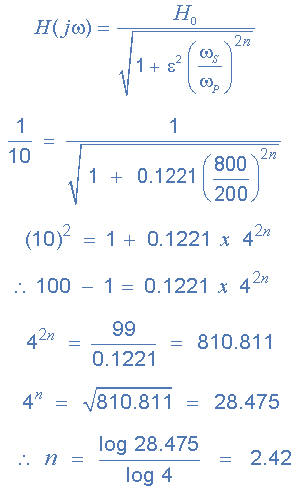
Find the order of an active low pass Butterworth filter whose specifications are given as: Amax = 0.5dB at a pass band frequency (ωp) of 200 radian/sec (31.8Hz), and Amin = 20dB at a stop band frequency (ωs) of 800 radian/sec. Also design a suitable Butterworth filter circuit to match these requirements.

Firstly, the maximum pass band gain Amax = 0.5dB which is equal to a gain of **1.0593**(0.5dB = 20log A) at a frequency (ωp) of 200 rads/s, so the value of epsilon ε is found by:



Secondly, the minimum stop band gain Amin = 20dB which is equal to a gain of **10** (20dB = 20log A) at a stop band frequency (ωs) of 800 rads/s or 127.3Hz.

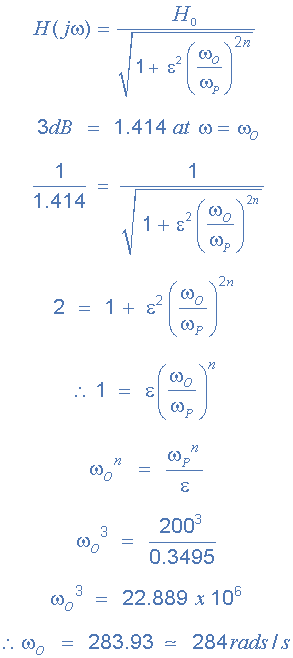
Substituting the values into the general equation for a Butterworth filters frequency response gives us the following:



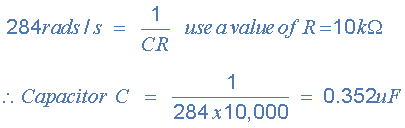
Since n must always be an integer ( whole number ) then the next highest value to 2.42 is n = 3, therefore a **“a third-order filter is required”** and to produce a third-order **Butterworth filter**, a second-order filter stage cascaded together with a first-order filter stage is required.

From the normalised low pass Butterworth Polynomials table above, the coefficient for a third-order filter is given as (1+s)(1+s+s2) and this gives us a gain of 3-A = 1, or A = 2. As A = 1 + (Rf/R1), choosing a value for both the feedback resistor Rf and resistor R1 gives us values of 1kΩ and 1kΩ respectively, ( 1kΩ/1kΩ + 1 = 2 ).

We know that the cut-off corner frequency, the -3dB point (ωo) can be found using the formula 1/CR, but we need to find ωo from the pass band frequency ωp then,



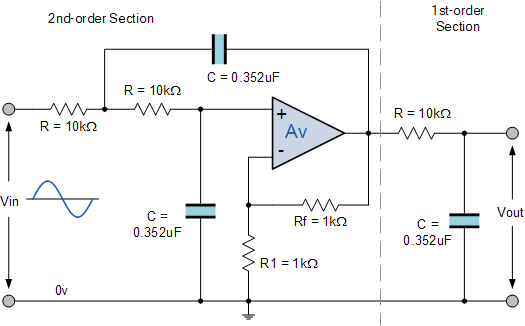
So, the cut-off corner frequency is given as 284 rads/s or 45.2Hz, (284/2π) and using the familiar formula 1/CR we can find the values of the resistors and capacitors for our third-order circuit.



Note that the nearest preferred value to 0.352uF would be 0.36uF, or 360nF.

**Third-order Butterworth Low Pass Filter**

and finally our circuit of the third-order low pass **Butterworth Filter** with a cut-off corner frequency of 284 rads/s or 45.2Hz, a maximum pass band gain of 0.5dB and a minimum stop band gain of 20dB is constructed as follows.



So for our 3rd-order Butterworth Low Pass Filter with a corner frequency of 45.2Hz, C = 360nF and R = 10kΩ