Digital Signal Processing

Higher Order Recursive Filters
Butterworth and Chebyshev

Load these files from MyCourses and Start MATLAB

- There is a MATLAB Live Script available to find the recursion equation coefficients
  - <IIR_Designer.mlx>

- Located in MyCourses in the Content/MATLAB Tools sections

Today's Topics

- Review of single pole recursive filters
- Introduction to Butterworth and Chebyshev Filters
- Designing Butterworth and Chebyshev filters using MATLAB
- In Class Problem
  - Upload the MATLAB file from myCourses
    - IIR_Designer.mlx

Filter Classification

FILTER USED FOR:	FILTER IMPLEMENTED BY: Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average (Ch. 15)	Single pole (Ch. 19)
Frequency Domain *(separating frequencies)*	Windowed-sinc (Ch. 16)	Chebyshev (Ch. 20)
Custom *(Deconvolution)*	FIR custom (Ch. 17)	Iterative design (Ch. 26)

FIR Filter Review

- The transfer function of the FIR filter is

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

- The filter transfer function has all zeros

Recursive (IIR) Filters

- The transfer function of the recursive filter is

$$H(z) = \frac{\sum_{k=0}^{M-1} a_k z^{-1}}{1 - \sum_{k=0}^{N-1} b_1 z^{-1}}$$

Zeros are the roots of the numerator

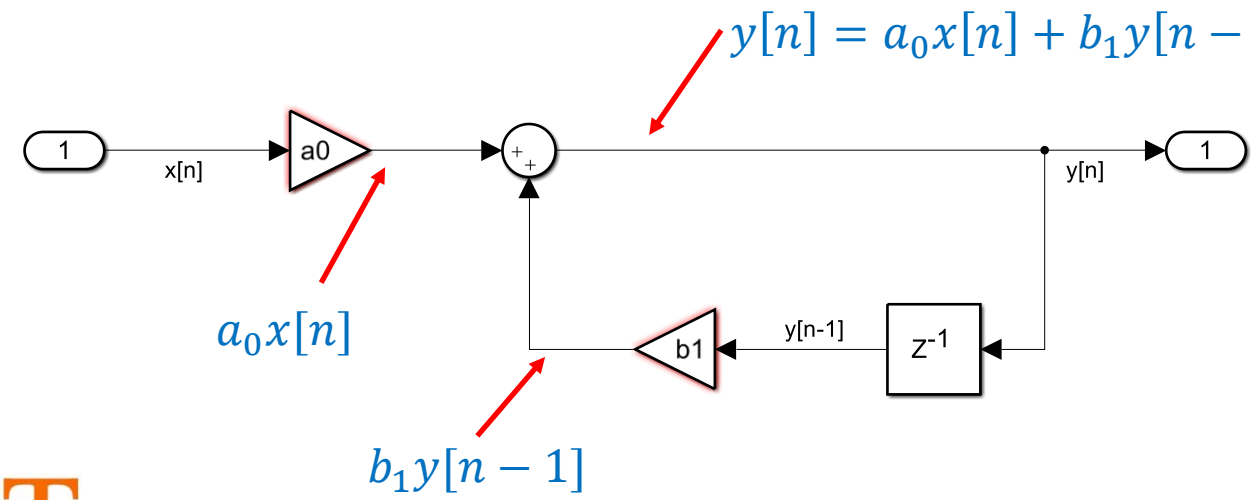
Poles are the roots of the denominator

- The recursive filter has both poles and zeros in its transfer function

The Single Pole Recursive Filter

- Transfer function of the single pole recursive low pass filter

$$H(z) = \frac{a_0}{1 - b_1 z^{-1}} \quad \Longleftrightarrow \quad y[n] = a_0 x[n] + b_1 y[n-1]$$



The Single Pole Recursive Filter

- The time constant of the filter is the value d in samples

$$x = e^{-1/d}$$

- We can relate this to the corner frequency of the filter

$$f_c = \frac{1}{2\pi d} \qquad x = e^{-2\pi f_c} \qquad \text{For the recursive filter}$$

$$f_c = \frac{1}{2\pi\tau} \qquad \text{For the analog RC filter}$$

Calculating the Filter Coefficients

- From the value of x we can easily find the single pole filter coefficients, a_0 and b_1

$$x = e^{-1/d} \qquad \text{or} \qquad x = e^{-2\pi f_c}$$

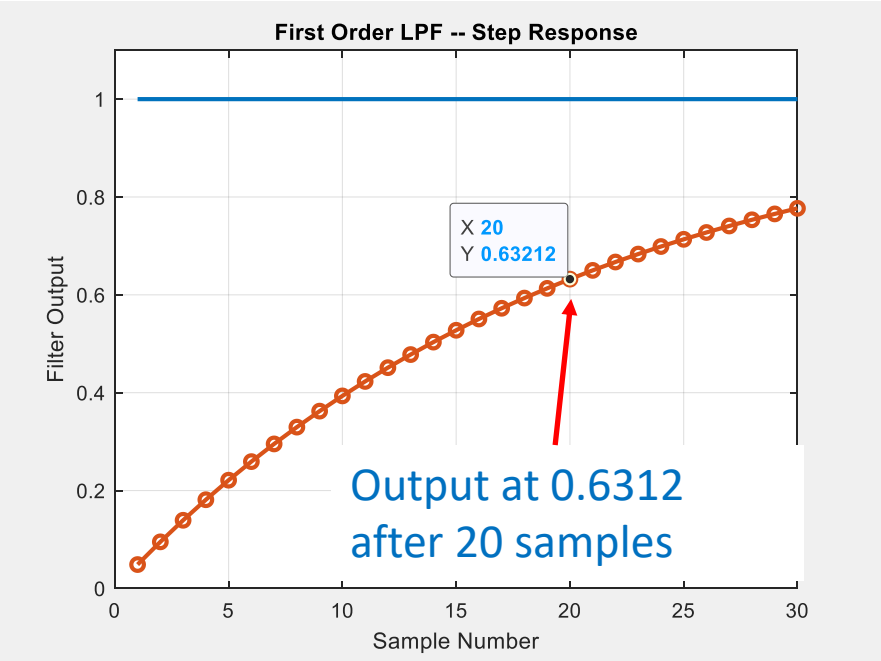
Then:

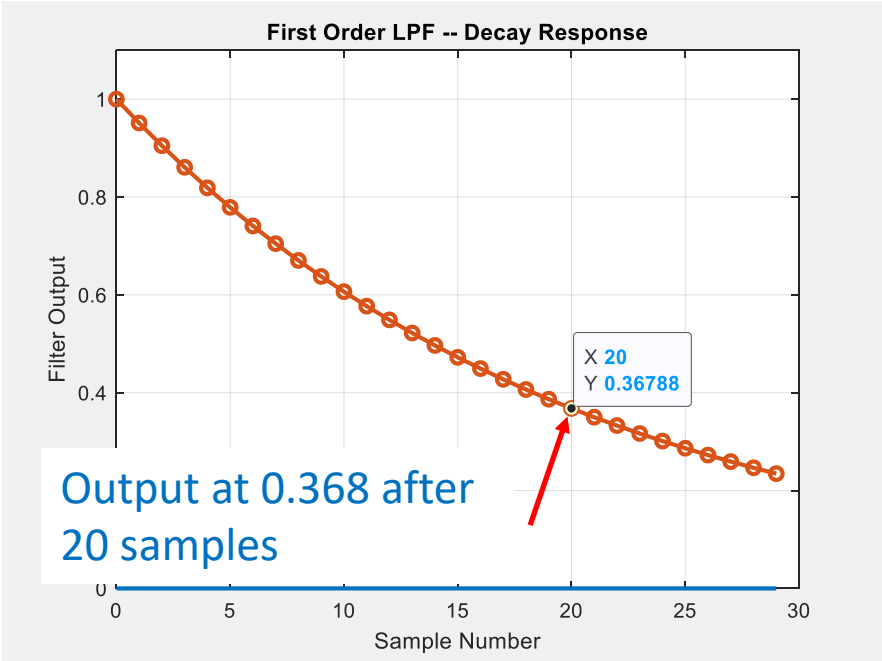
$$b_1 = x$$

$$a_0 = 1 - x$$

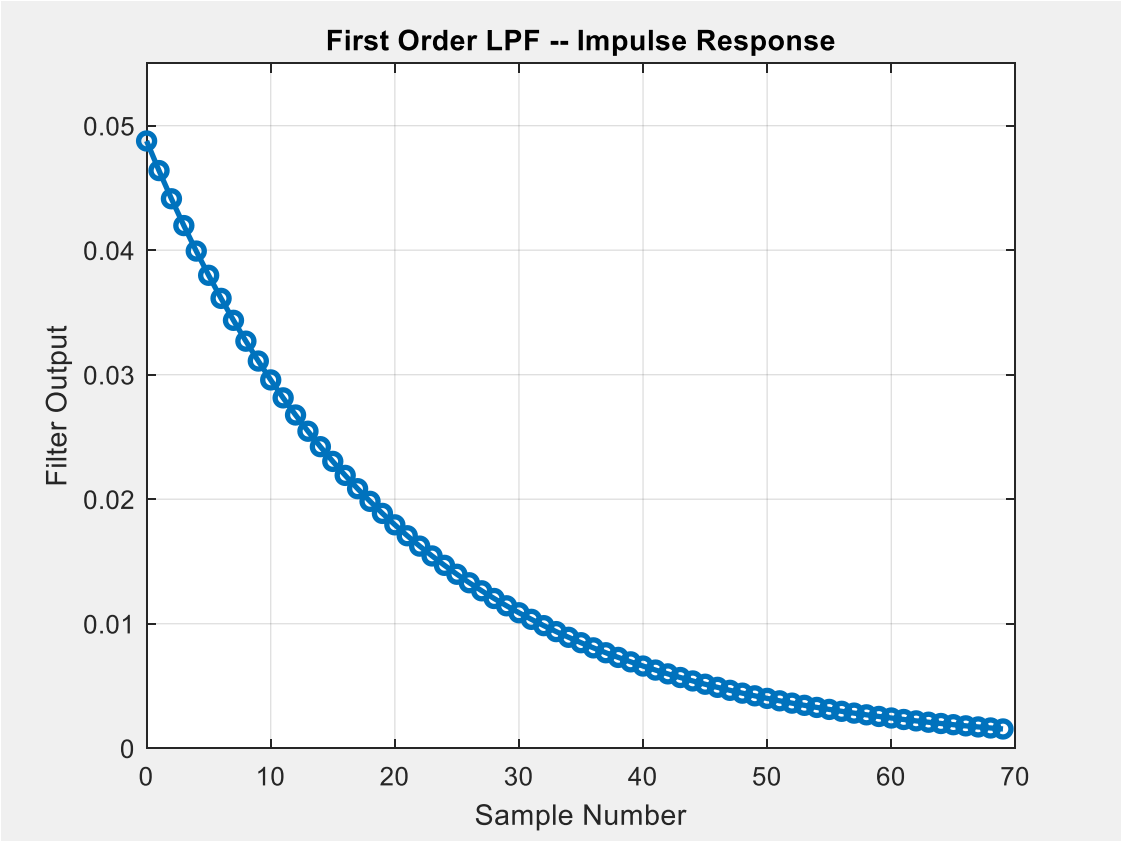
First Order Recursive Filter Example

- First order LPF Step and Decay responses for a filter with a time constant of 20 samples



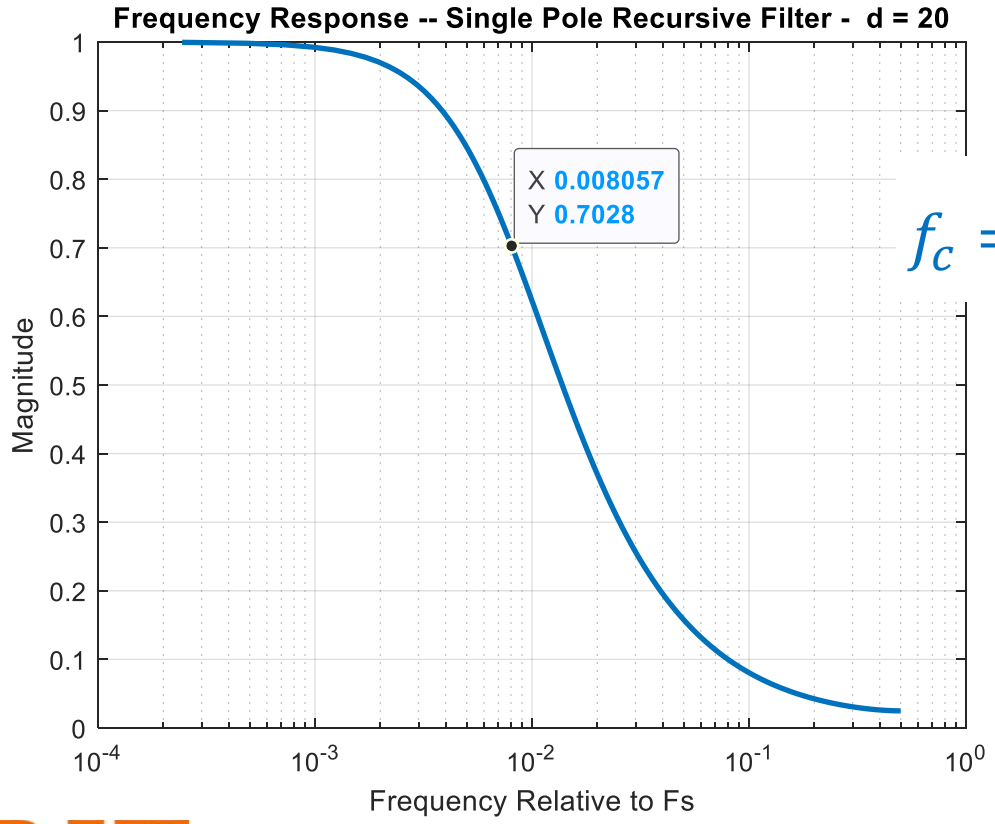


First Order Recursive Filter Impulse Response



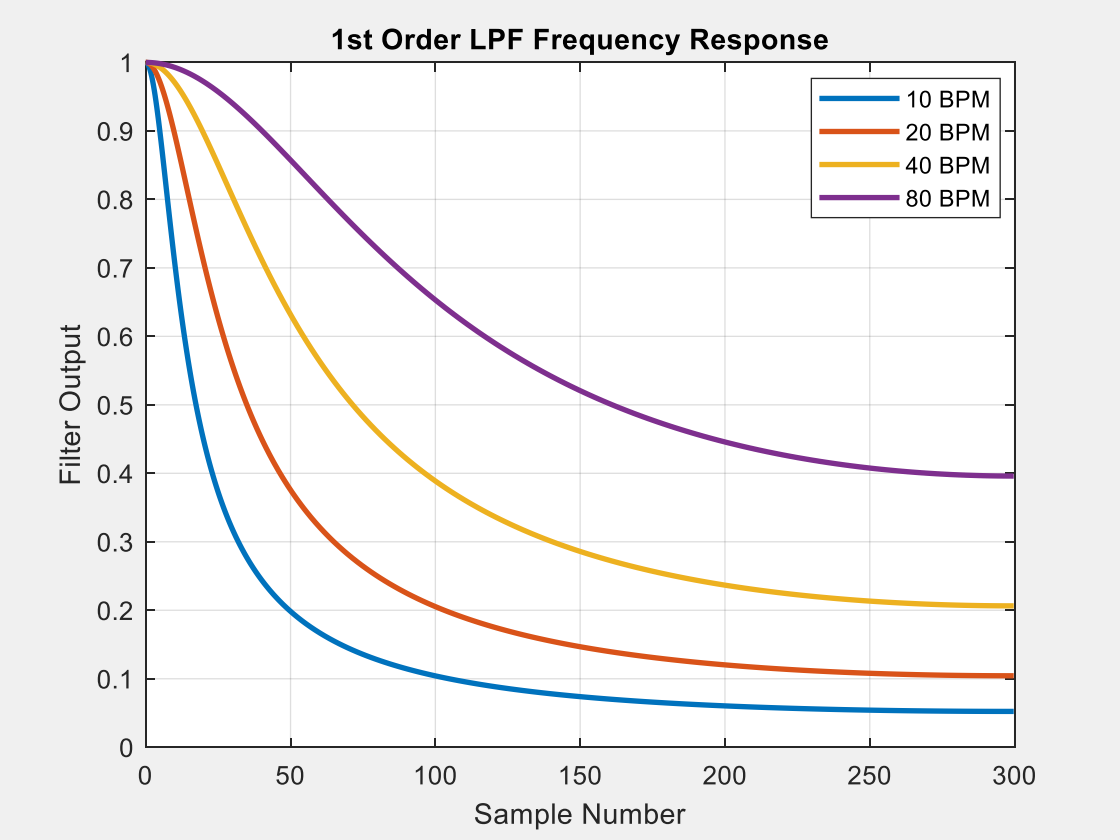
First order Recursive LPF Example

- Frequency Response of the single pole IIR filter



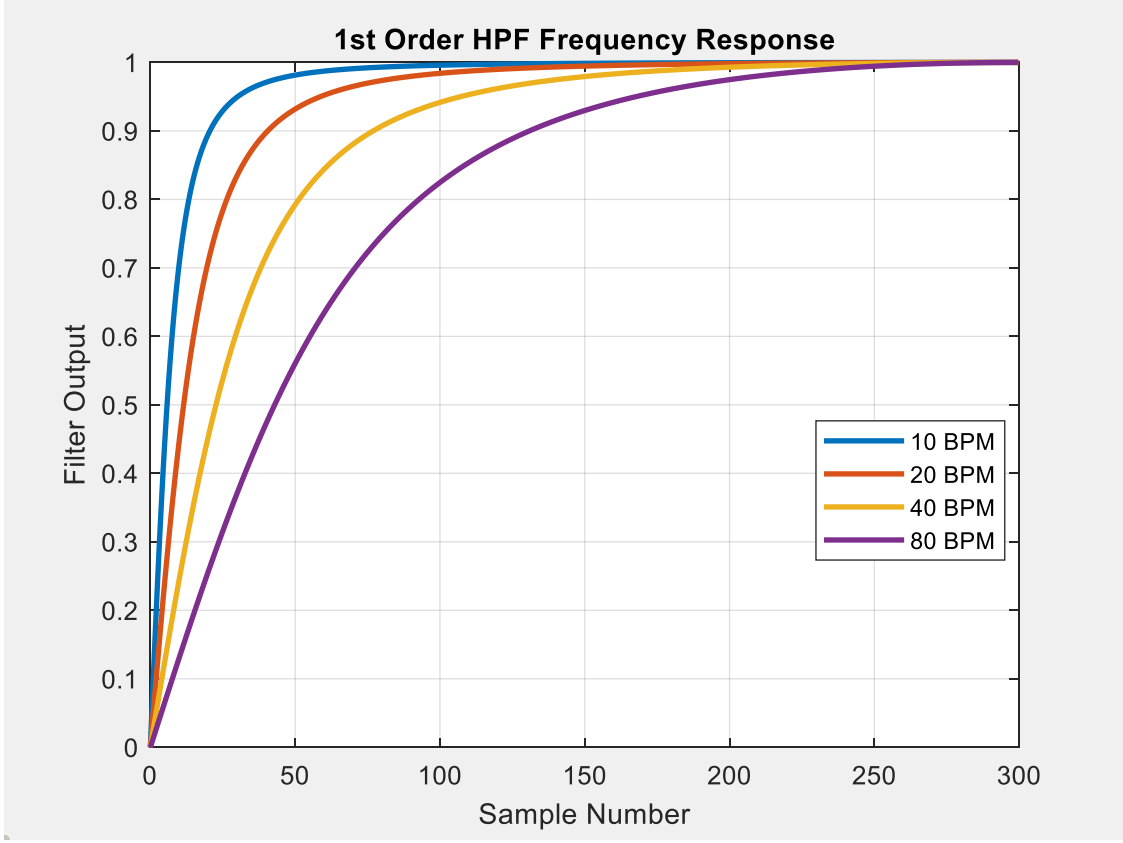
$$f_c = \frac{1}{2\pi d} = \frac{1}{2\pi\ 20} = 0.008$$

Comparing LPF Corner Frequencies



As I change the corner frequency the response gets lower in frequency but the relative attenuation stays the same

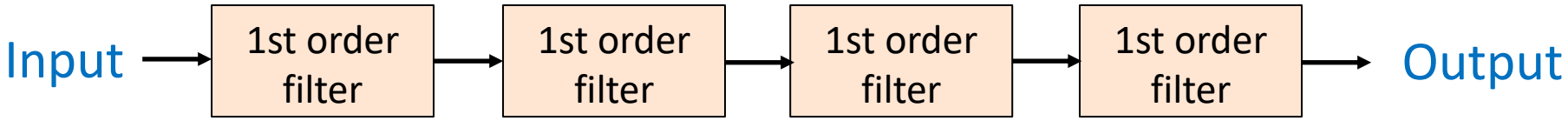
Comparing HPF Corner Frequencies

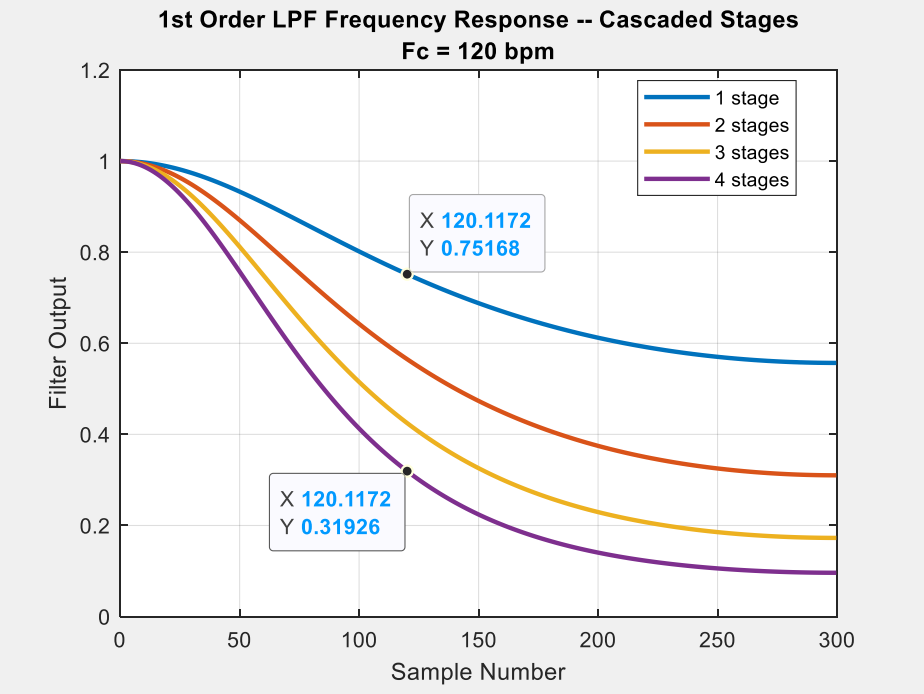


As the corner frequency gets lower the response gets lower in frequency but the relative attenuation stays the same

Cascading Multiple Stages

- Single pole filters can be cascaded, but their roll-off doesn't improve dramatically





1st Order Response of 1-4 cascaded stages

Other Filter Types

- Bandpass and bandstop (notch) filters can also be made with recursive filters (see text)
- Higher order filters can be designed with plentiful design tools
- MATLAB will be used in Lab 09 to create Butterworth and Chebyshev higher order filters

Higher Order Recursive (IIR) Filters

- Adding poles and zeros to the transfer function can improve the frequency response of the filter

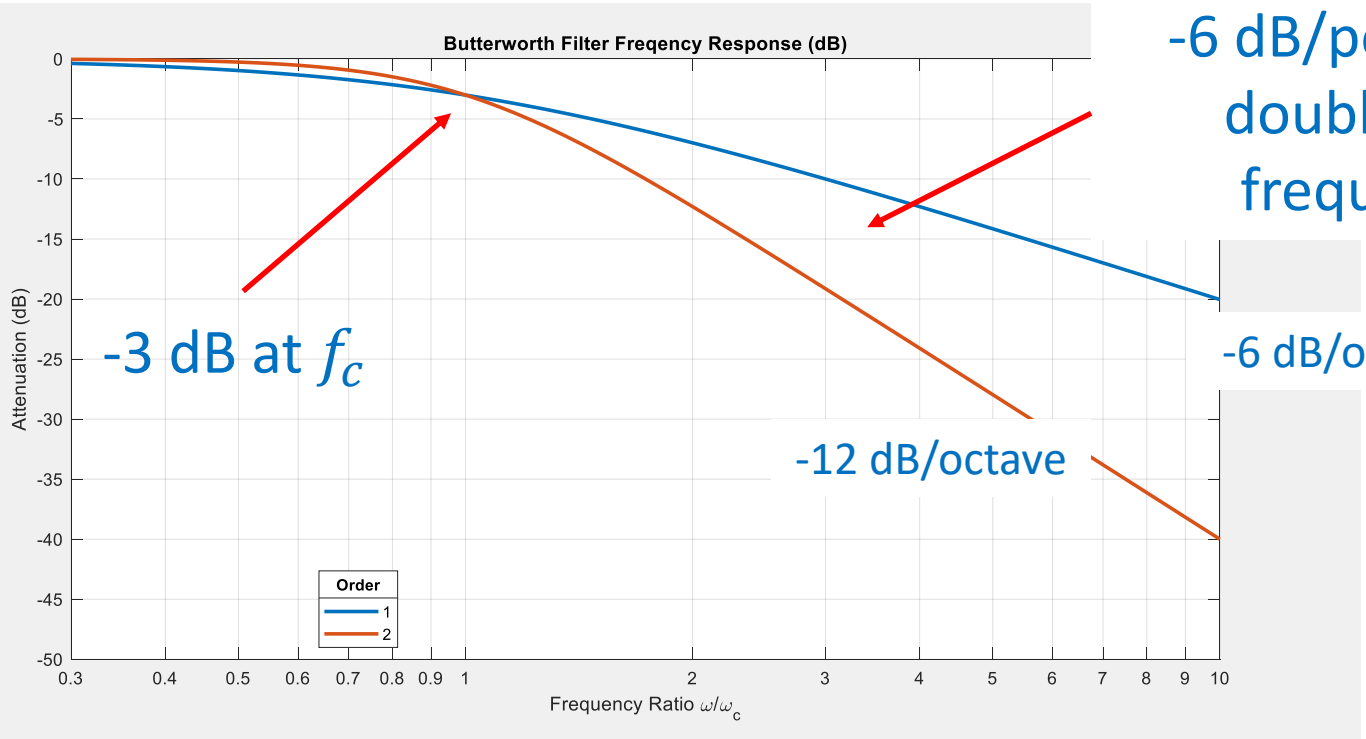
Zeros are the roots of the numerator

$$H(z) = \frac{\sum_{k=0}^{M-1} a_k z^{-1}}{1 - \sum_{k=0}^{N-1} b_1 z^{-1}}$$

Poles are the roots of the denominator

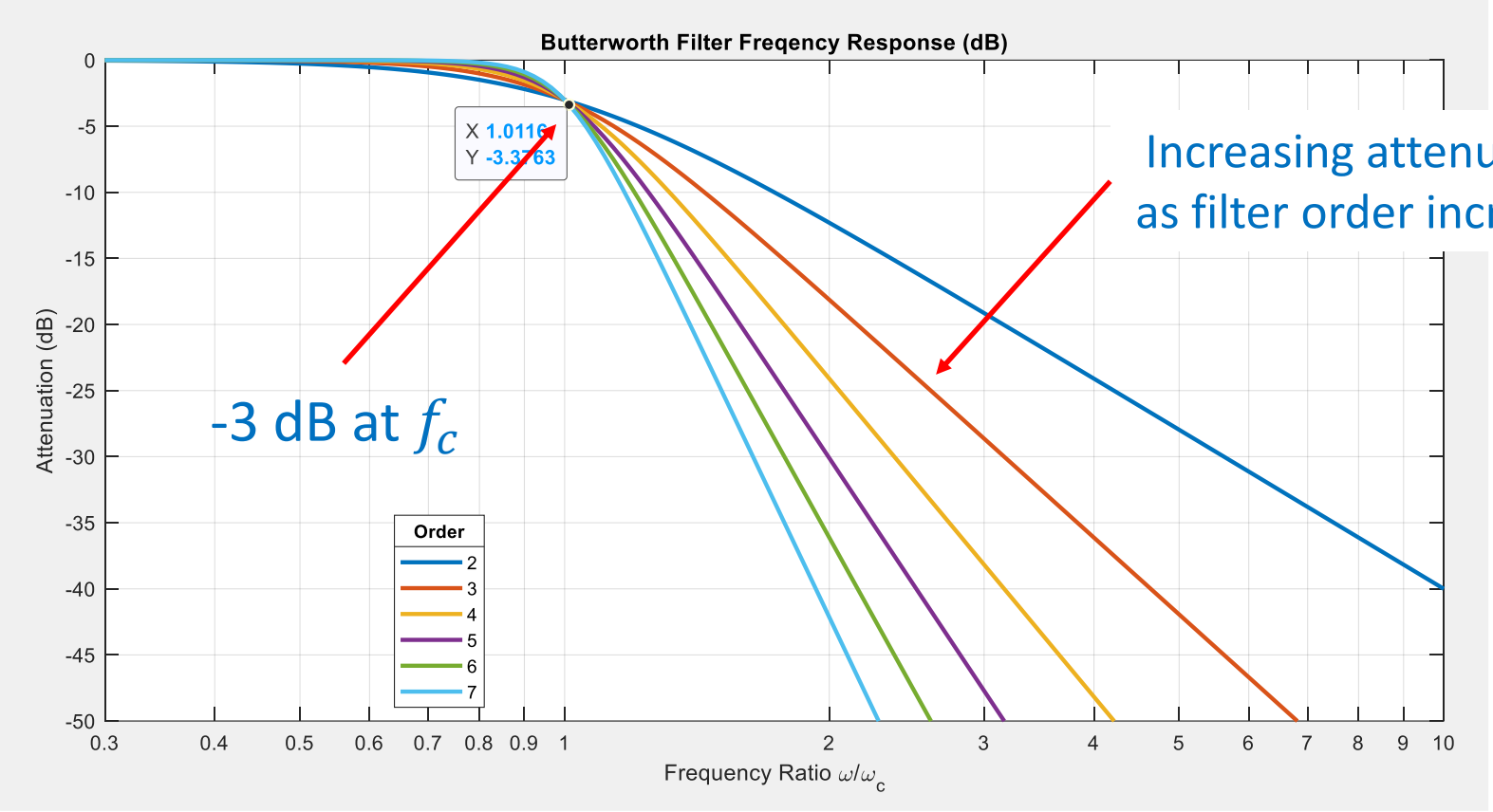
The Butterworth Filter

- The frequency response (dB scale) of a 1 and 2 pole Butterworth filter is shown here



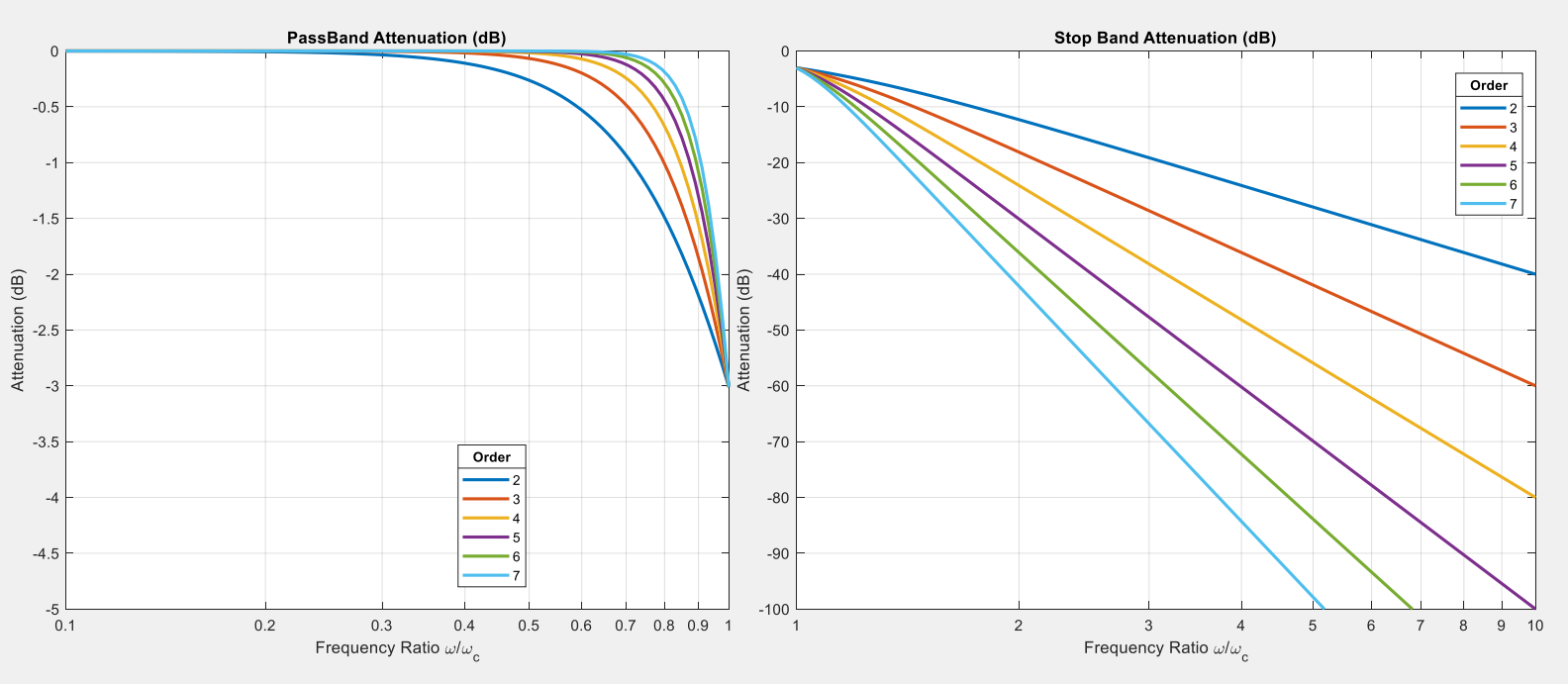
Frequency Response of the Butterworth Filter

- Filter orders 2 through 7 shown



Pass band and Stop Band of the Butterworth Filter

- The passband is maximally flat
- The stopband is also maximally flat

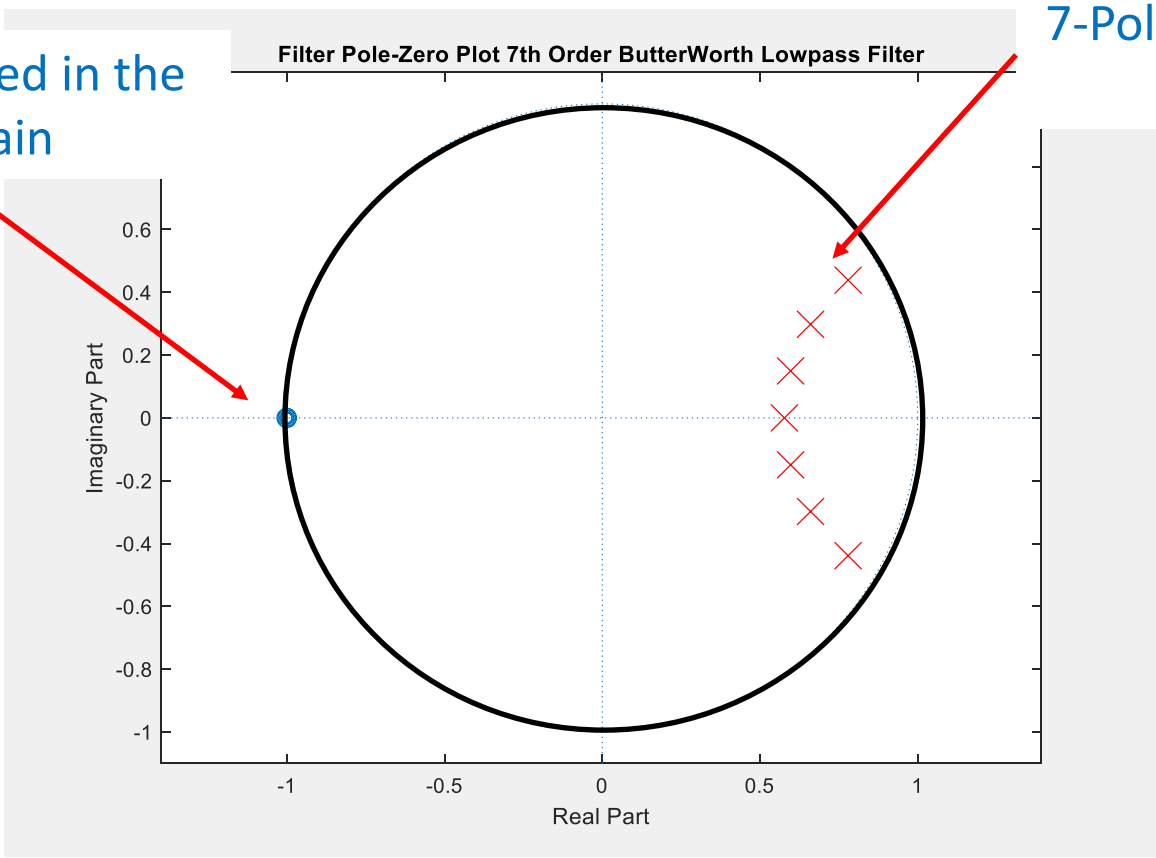


Butterworth Filter Pole-Zero Diagram

- 7-poles and 7 zeros
- Location of the poles and zeros determines the filter response

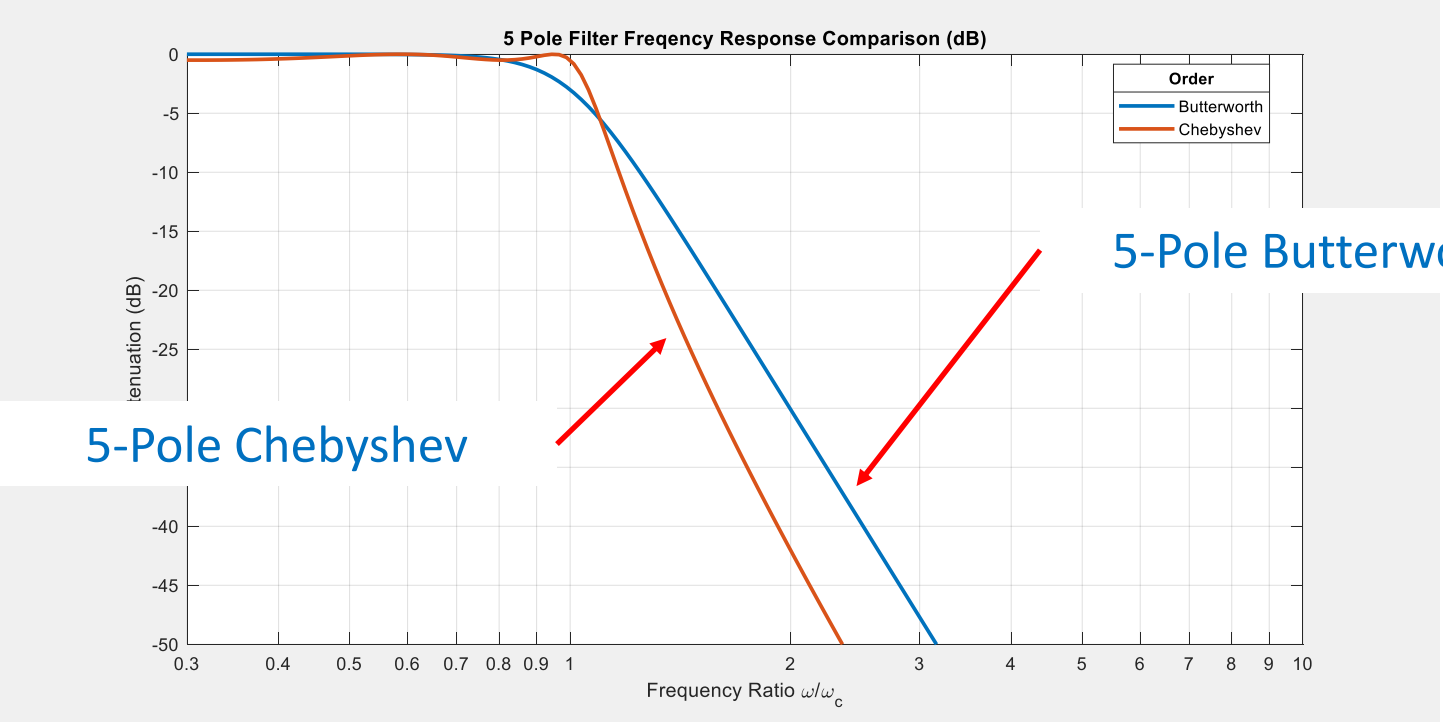
7-Zeros located in the Z-domain

7-Poles locations in the Z-domain



The Chebyshev Filter

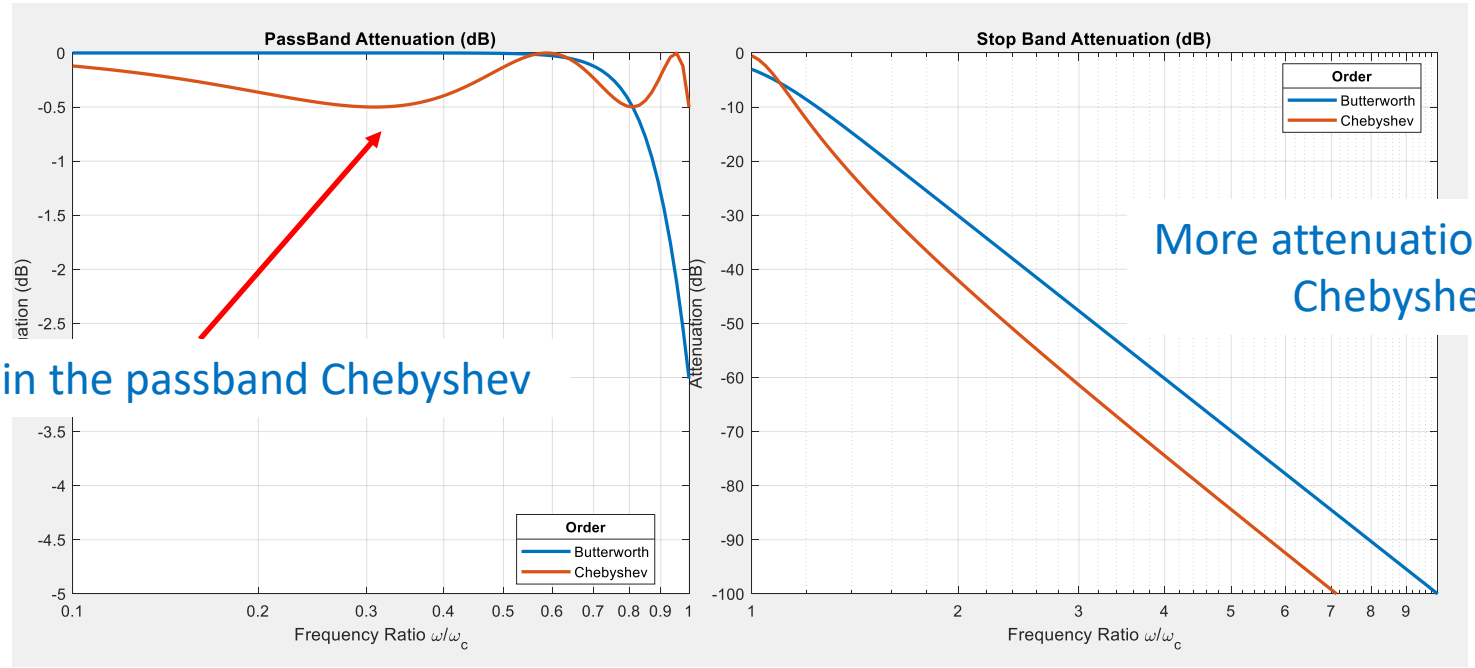
- The Chebyshev has more roll-off than a comparable Butterworth filter



The Chebyshev Filter

- The Chebyshev trades roll-off for ripple in the passband

Flat passband -- Butterworth

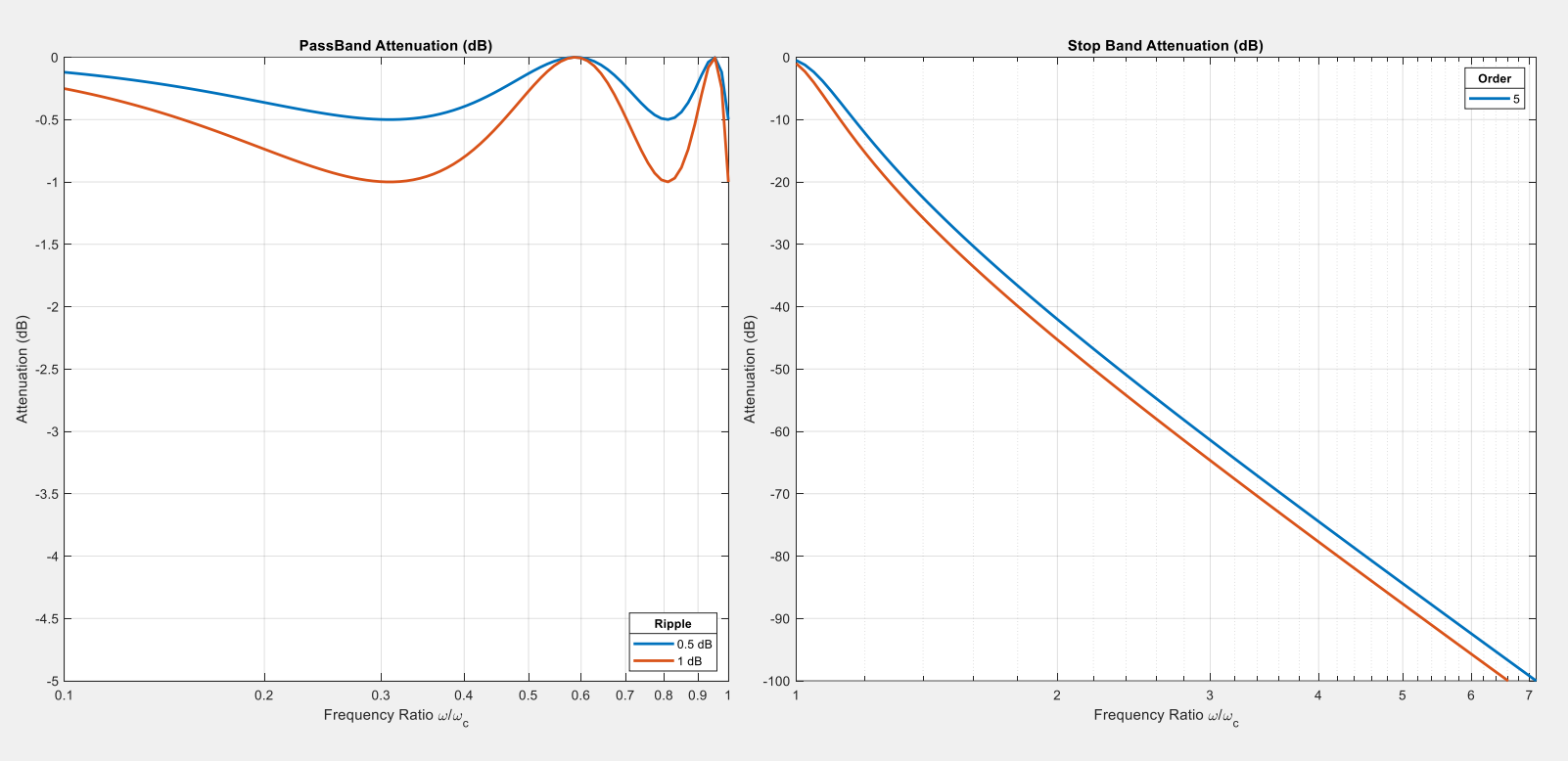


Ripple in the passband Chebyshev

More attenuation for the Chebyshev

Passband Ripple

- The Chebyshev filter trades off passband ripple for attenuation

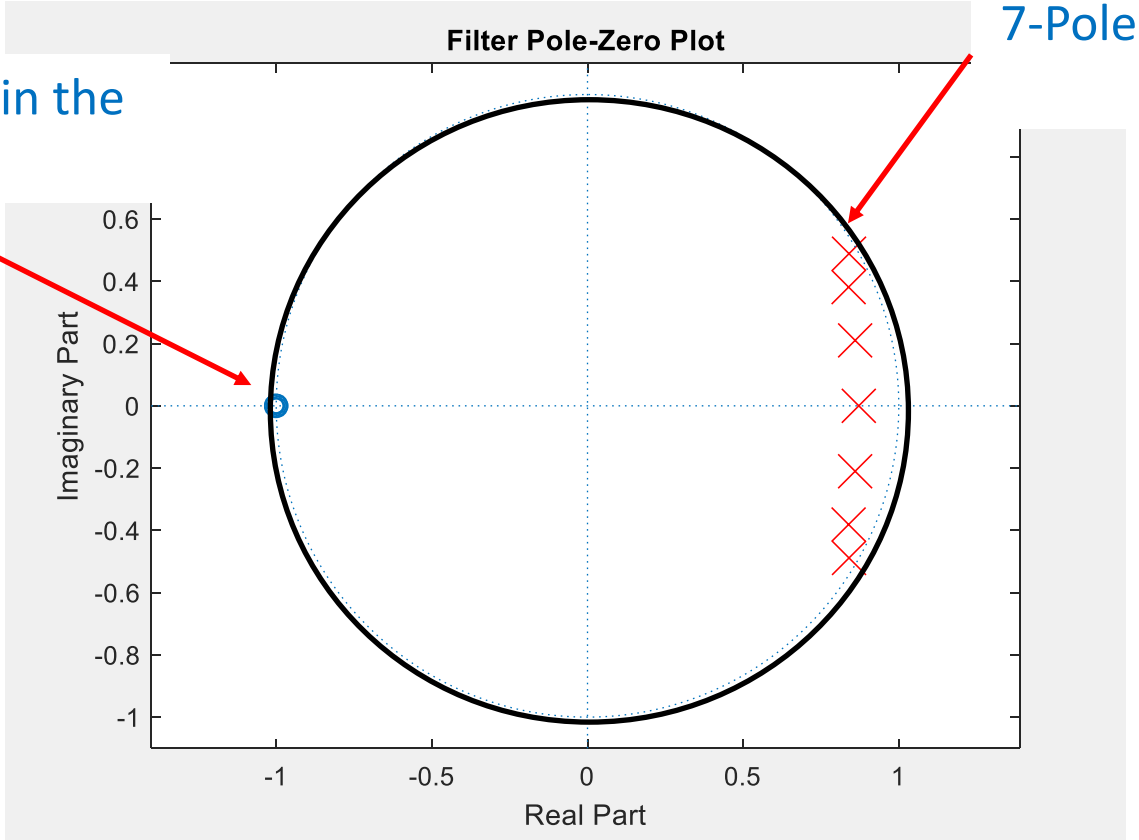


Chebyshev Filter Pole-Zero Diagram

- 7-poles and 7 zeros
- Location of the poles and zeros determines the filter response

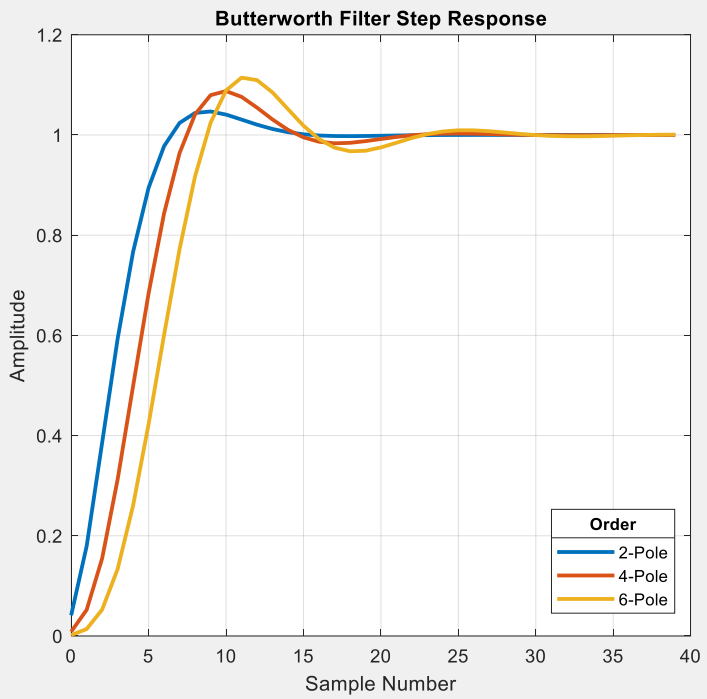
7-Zeros located in the Z-domain

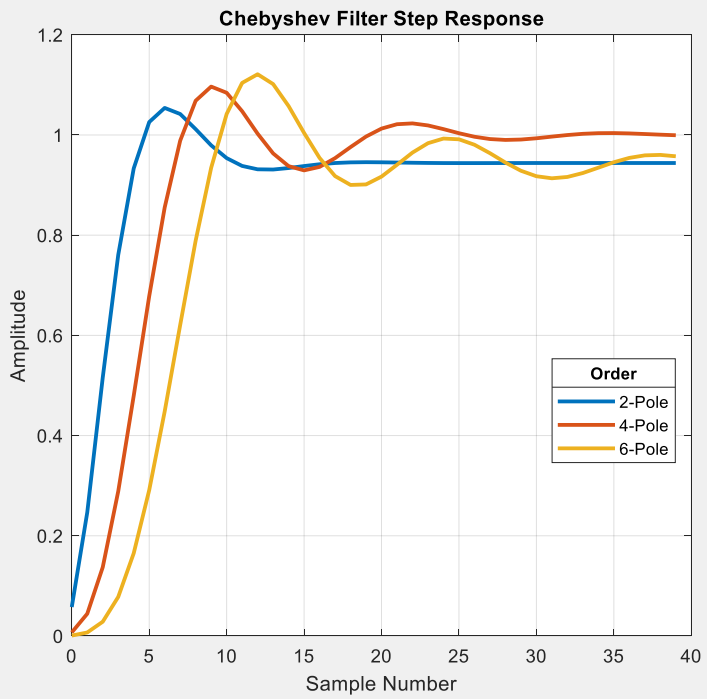
7-Poles locations in the Z-domain



Step Response Comparison

- Comparison of Butterworth and Chebyshev filter step response for 2, 4 and 6 poles





Implementing the High Order Recursive Filters

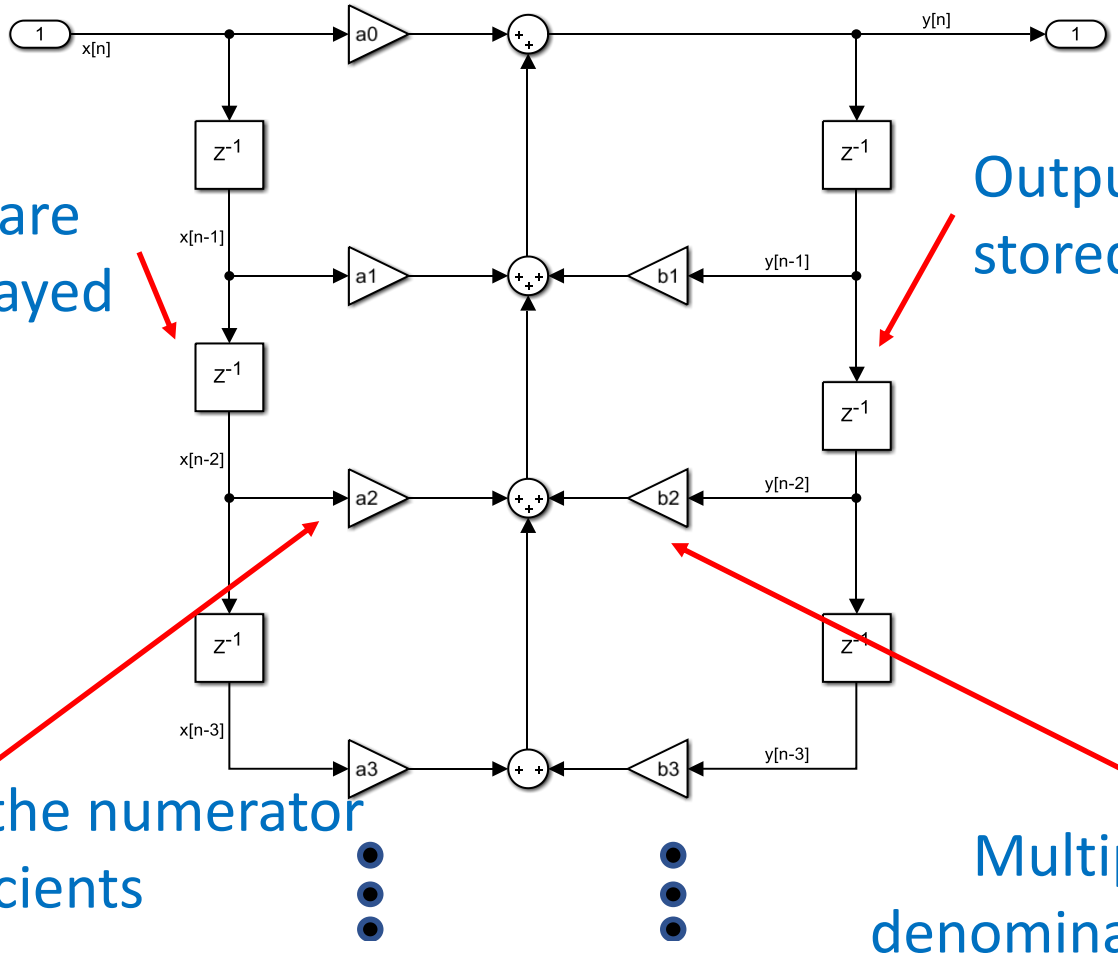
- Recursive filters compute the next output using:
  - The filter input values $x[n], x[n-1]$ ...
  - *and* the past filter output values $y[n-1], y[n-2]$, ...

- The difference equation is:

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + \cdots$$
$$b_1 y[n-1] + b_2 y[n-2] + b_3 y[n-3] + \cdots$$

- This is referred to as the direct form of the IIR

High Order Recursive Filters Block Diagram



Input samples are stored and delayed

Output samples are stored and delayed

Multiplied by the numerator coefficients

Multiplied by the denominator coefficients

Where Do You Locate the Poles and Zeros?

- The poles and zeros are found by solving the polynomials for the Butterworth and Chebyshev equations
  - Complicated and tedious
  - Tables with poles and zero locations

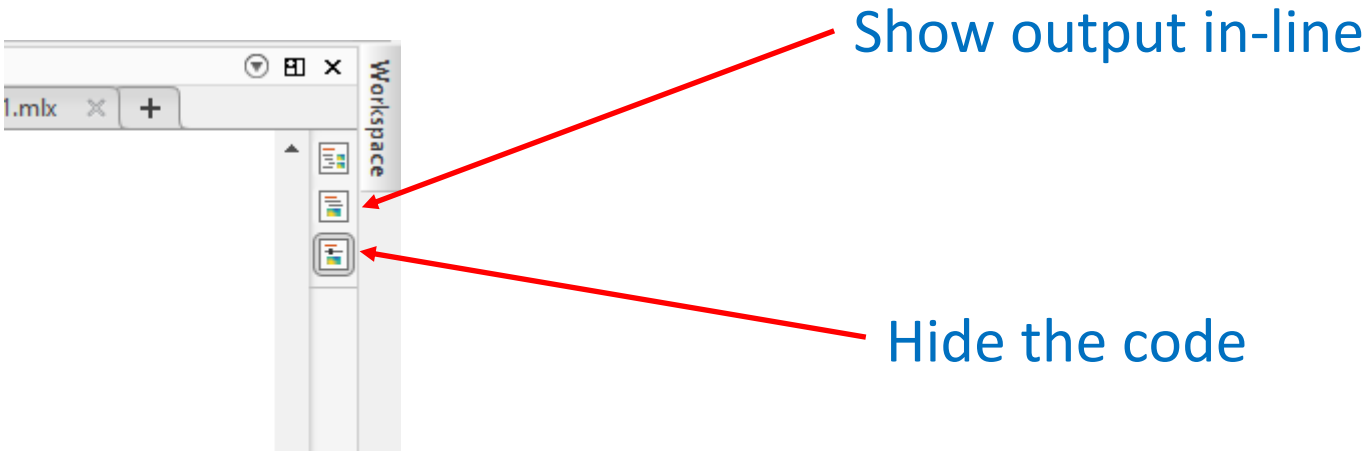
- There is an easier way
  - Filter design software
  - MATLAB

Using MATLAB to Find Filter Coefficients

- There is a MATLAB Live Script available to find the recursion equation coefficients
  - <IIR_Designer.mlx>

- Located in MyCourses in the Content/MATLAB Tools sections

How to Use the MATLAB Code



For easier viewing you can hide the underlying code

How to Use the MATLAB Code

Typically the breathing rate system is using a sampling rate of $10Hz$ or $600BPM$.

samplingFreq 600

Choose the filter type (Butterworth or Chebyshev) and the filter topology (LPF, HPF, BPF, BSF)

filterType Butterworth

filterTopology Lowpass

An additional parametr for the Chebyshev filter is the amount of ripple in the passband. If Chebyshev is the filter type selected then set the amount of ripple in decibels (dB).

rippleDb 0.5

The order of the filter will determine the number of poles in the filter. Th ases the filter com

filterOrder 5

Choose the corner frequency (sometimes called the cutoff frequency). This determines the location of the passband of the filter. In the case of the lowpass filter you only need to sp
frequency and the upper corner frequency. The figures below describe these values. The units for the corner frequency will be the same as the sampling frequency.

If the filter is a LPF or HPF, enter the corner frequency below

cornerFreq 70

If the filter is a BPF or BSF, enter both the upper and the lower corner frequecies below

lower_bpf_bsf_cornerFreq 50

upper_bpf_bsf_cornerFreq 80

Sample rate set to 600 bpm

Define the filter type and topology

If Chebyshev define the ripple (dB)

Set the filter order

How to Use the MATLAB Code

Typically the breathing rate system is using a sampling rate of $10Hz$ or $600BPM$.

samplingFreq 600

Choose the filter type (Butterworth or Chebyshev) and the filter topology (LPF, HPF, BPF, BSF). Here are some examples of various filter types and topologies.

filterType Butterworth

filterTopology Lowpass

An additional parametr for the Chebyshev filter is the amount of ripple dB).

rippleDb 0.5

The order of the filter will determine the number of poles in the filter. The gre lter com

filterOrder 5

Choose the corner frequency (sometimes called the cutoff frequency). This ed to sp
frequency and the upper corner frequency. The figures below describe thes

If the filter is a LPF or HPF, enter the corner frequency belo

cornerFreq 70

If the filter is a BPF or BSF, enter both the upper and the lower corner frequecies below

lower_bpf_bsf_cornerFreq 50

upper_bpf_bsf_cornerFreq 80

If LPF or HPF set the corner frequency (same units as sampling frequency)

If BPF or BSF set the upper and lower Corner frequency (same units as sampling frequency)

IIR Designer Results

Here is the z-transform of the filter:

```
Tfilt =

  0.00246 + 0.0123 z^-1 + 0.0246 z^-2 + 0.0246 z^-3 + 0.0123 z^-4 + 0.00246 z^-5
  ---------------------------------------------------------------------------
       1 - 2.641 z^-1 + 3.12 z^-2 - 1.954 z^-3 + 0.641 z^-4 - 0.08709 z^-5

Sample time: 0.1 seconds
Discrete-time transfer function.
```

Z-transform

The complex Zeros of the transfer function (roots of the numerator) are shown below:

```
zeros_b = 5×1 complex
   -1.0008 + 0.0006i
   -1.0008 - 0.0006i
   -0.9997 + 0.0009i
   -0.9997 - 0.0009i
   -0.9990 + 0.0000i
```

Complex zero-locations

The complex Poles of the transfer function (roots of the denominator) are shown below:

```
poles_a = 5×1 complex
    0.6158 + 0.5273i
    0.6158 - 0.5273i
    0.4821 + 0.2552i
    0.4821 - 0.2552i
    0.4452 + 0.0000i
```

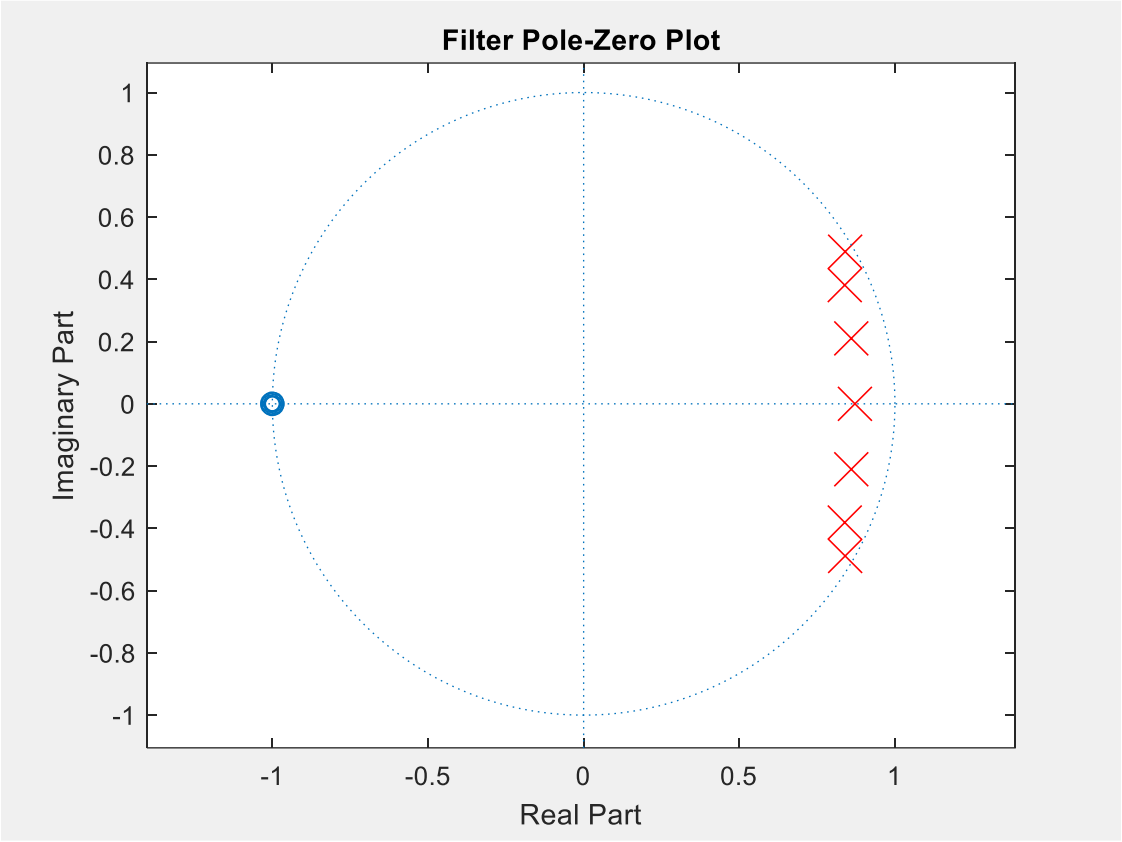
Complex pole-locations

The complex Poles of the transfer function can also be expressed in terms of frequency by computing the magnitude of the complex poles:

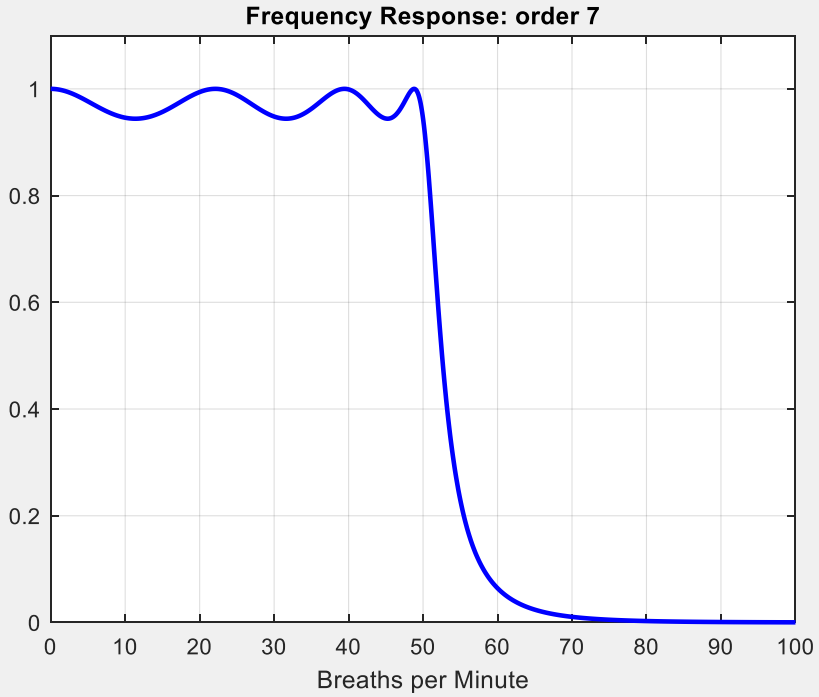
```
pole_wn = 5×1
    0.8107
    0.8107
    0.5455
    0.5455
    0.4452
```

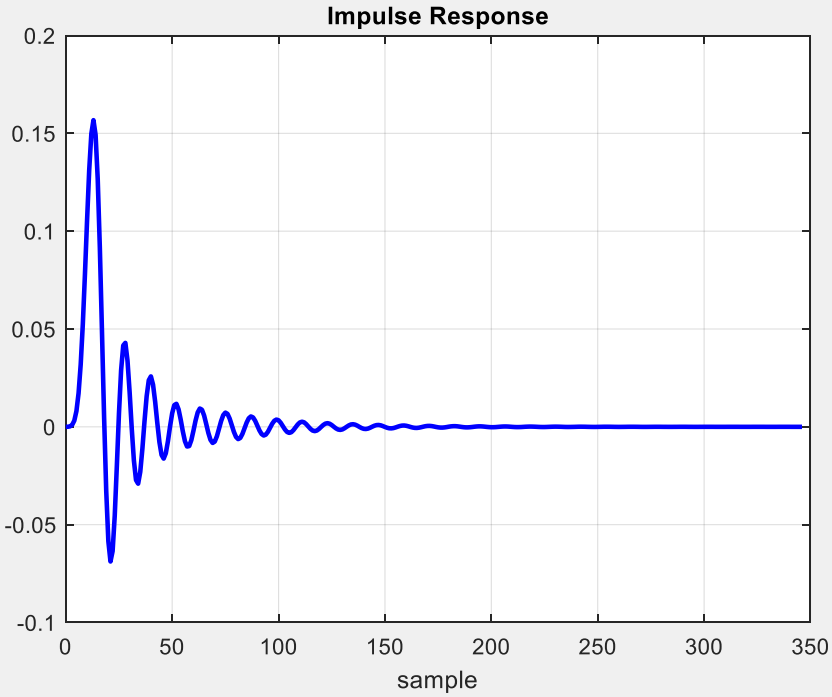
Pole-locations in terms of frequency

Pole Zero Plots



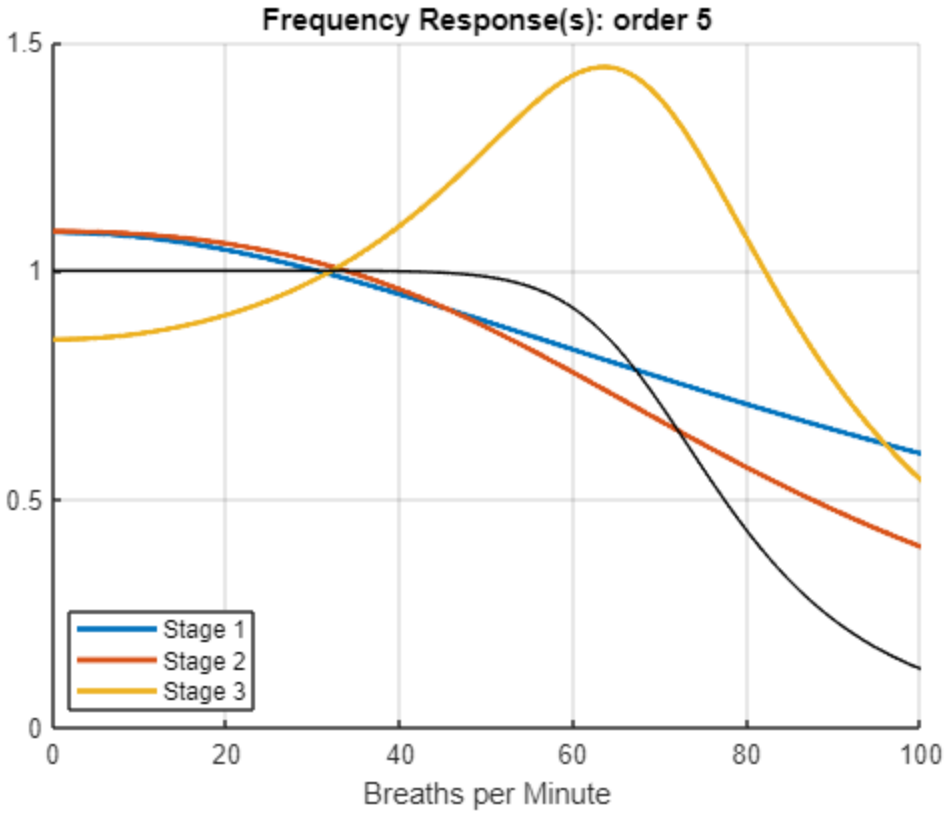
Frequency and Impulse Response Graphs





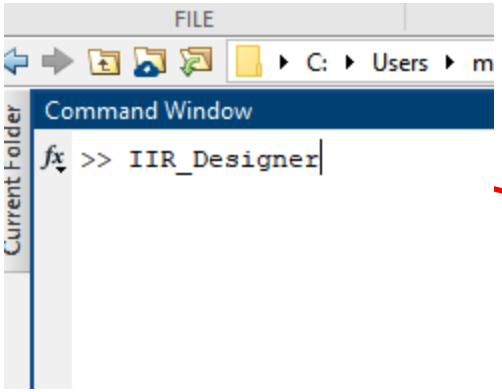
Frequency and Impulse Response Graphs

- Frequency response of individual stages



Coefficients and Header Text

- In order to copy from MATLAB to the Arduino IDE
  - Set parameters as before in Live Script
  - Run IIR_Designer in the command window



Run “IIR_Designer”
in the command window

Coefficients and Header Text

First print the C-code for the Direct form of the IIR Filter

```
****  C Code for Direct IIR Filter -- Copy to Arduino IDE ****

  //BWRTH low, order 5, 70 BPM

  const int MFILT = 6;
  static float GAIN = 0.0245967;
  static float b[] = {0.1000000, 0.5000000, 1.0000000, 1.0000000, 0.5000000, 0.1000000};
  static float a[] = {1.0000000, -2.6411375, 3.1202049, -1.9542490, 0.6409775, -0.0870864};
```

Direct form

Print the C-Code for the cascaded Second Order Sections (SOS). This may be copied and placed in window.

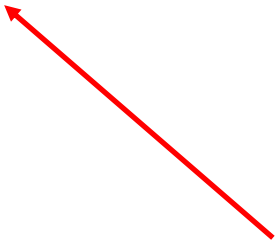
```
 ****  C Code for SOS Filter -- Copy to Arduino IDE ****

//Filter specific variable declarations
const int numStages = 3;
static float G[numStages];
static float b[numStages][3];
static float a[numStages][3];


//BWRTH low, order 5, 70 BPM

G[0] = 0.1349871;
b[0][0] = 1.0000000; b[0][1] = 0.9990472; b[0][2]= 0.0000000;
a[0][0] = 1.0000000; a[0][1] =  -0.4452287; a[0][2] =  0.0000000;
G[1] = 0.1349871;
b[1][0] = 1.0000000; b[1][1] = 2.0015407; b[1][2]= 1.0015416;
a[1][0] = 1.0000000; a[1][1] =  -0.9642853; a[1][2] =  0.2975739;
G[2] = 0.1349871;
b[2][0] = 1.0000000; b[2][1] = 1.9994122; b[2][2]= 0.9994131;
a[2][0] = 1.0000000; a[2][1] =  -1.2316235; a[2][2] =  0.6573129;
```

Cascaded stages

Coefficients and Header Text

Print the coefficients so that they can easily be copied into MATLAB

```
Direct IIR Filter Coefficients for easy copy to MATLAB

b = [0.0024597, 0.0122984, 0.0245967, 0.0245967, 0.0122984, 0.0024597];
a = [1.0000000, -2.6411375, 3.1202049, -1.9542490, 0.6409775, -0.0870864];
```



Coefficients for copying into MATLAB

Butterworth Filter ICP

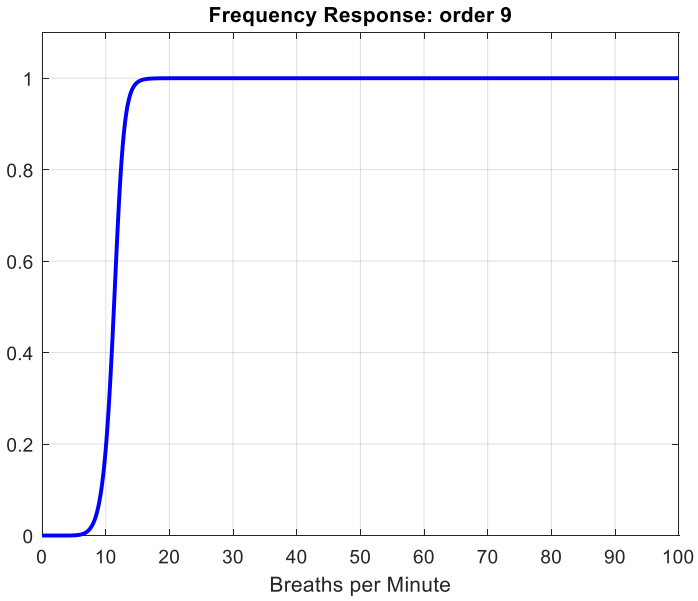
- Using the MATLAB code for Butterworth and Chebyshev filters, design a filter with the following parameters

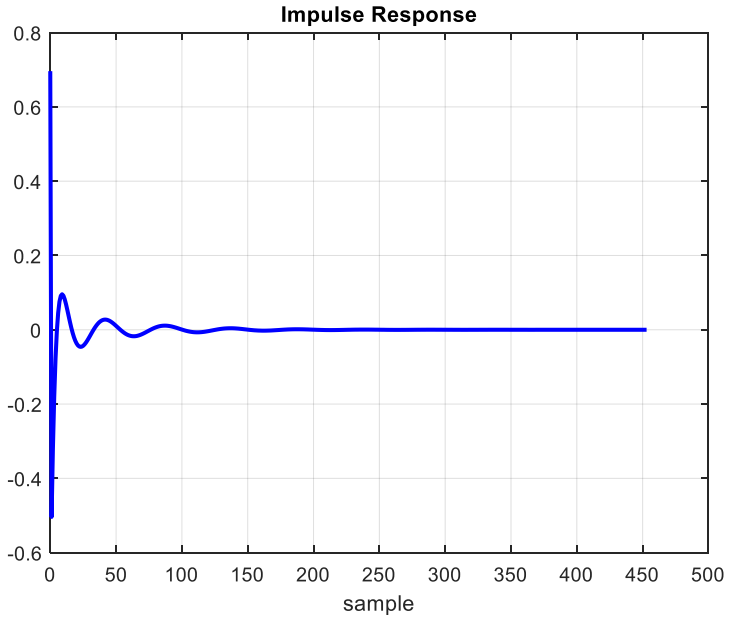
- Butterworth filter
- High pass
- 7 poles
- Corner Frequency – 12 BPM

Butterworth Filter ICP

NO PRINT

- Frequency and impulse response





Butterworth Filter ICP

NO PRINT

- C-Code variable declarations

**** C Code for Direct IIR Filter -- Copy to Arduino IDE ****

```
// BWRTH HIGH, order 9, 12 BPM
const int MFILT = 10;
static float GAIN = 87.7188;
static float b[] = {0.0079365, -0.0714286, 0.2857143, -0.6666667, 1.0000000, -1.0000000, 0.6666667, -0.2857143, 0.0714286, -0.0079365};
static float a[] = {1.0000000, -8.2763713, 30.4708077, -65.4968459, 90.5808889, -83.5828216, 51.4579469, -20.3816702, 4.7127337, -0.4846682};
```

C-Code Variable Declaration

- The MATLAB script will print out variable declarations for the filter design

```
****  C Code for Direct IIR Filter -- Copy to Arduino IDE ****

 // CHEBY LOW, order 7, R = 0.5, 50 BPM
 const int MFILT = 8;
 static float GAIN = 0.000101653;
 static float b[] = {0.0285714, 0.2000000, 0.6000000, 1.0000000, 1.0000000, 0.6000000, 0.2000000, 0.0285714};
 static float a[] = {1.0000000, -5.9489241, 15.5949386, -23.3080572, 21.4230142, -12.0992813, 3.8862757, -0.5475942};
```

Numerator coefficients (b's)

Denominator coefficients (a's)

NOTE: The MATLAB output calls the numerator coefficient b and denominator coefficient a. Opposite of the text

Paste in the filter parameters from MATLAB

- The variable declarations for the filter recursion coefficients can be copied into the C-code

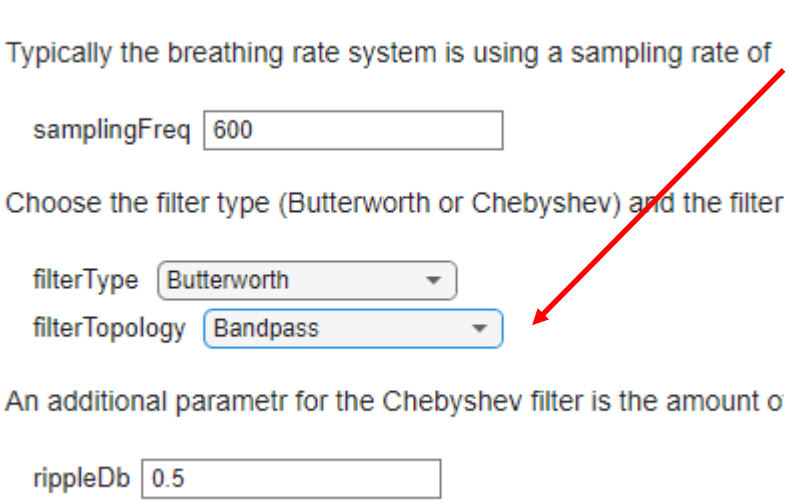
```
//*****************************************************************************
float IIR_DIRECT(float xv)
{

  //  Data to define the filter coefficients for the direct form IIR filter


  // CHEBY LOW, order 5, R = 0.5, 50 BPM
  const int MFILT = 6;
  static float GAIN = 0.0016669;
  static float b[] = {0.1000000, 0.5000000, 1.0000000, 1.(
  static float a[] = {1.0000000, -4.0763511, 6.9540630, -(


  //-------------------------------------------------------------
  //  Two arrays to contain the input and output sequences in time
  static float xM[MFILT] = {0.0}, yM[MFILT] = {0.0};
```

Paste in the declarations from MATLAB for the filter designs

Making Bandpass and Bandstop Filters

- Use the same IIR_Designer.mlx Live Script

Making Bandpass and Bandstop Filters



Change the filter topology to BPF Or BSF

If BPF or BSF set the upper and lower Corner frequency (same units as sampling frequency)

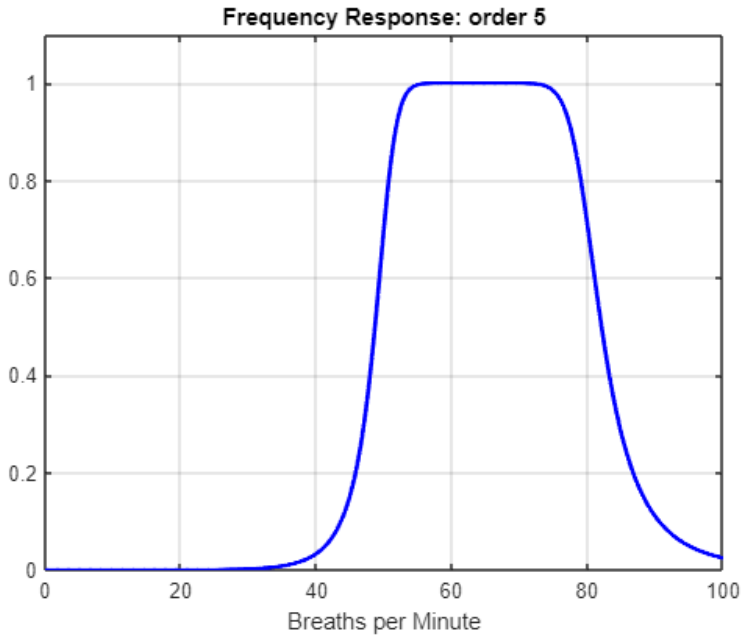
Making Bandpass and Bandstop Filters

If BPF or BSF set the upper and lower Corner frequency (same units as sampling frequency)

If the filter is a BPF or BSF, enter both the upper and the lower corner frequecies below

lower_bpf_bsf_cornerFreq 50

upper_bpf_bsf_cornerFreq 80



Why Use Chebyshev Filters?

- Can provide sharper attenuation than Butterworth filters
  - Not as sharp as windowed-sinc filter, but they are more than adequate for many applications.

- Chebyshev filters are fast
  - Typically more than an order of magnitude faster than the windowed-sinc.
  - Because they are implemented using recursion rather than convolution.

Round off Error Limitations of Recursive Filters

- Round off error of single precision floating point limits the number of poles that can be used in higher order recursive filters
- The range of the smallest recursion coefficients and the largest coefficients changes with corner frequency
- Exceeding these limits of the data type results in excessive passband ripple, poor stopband attenuation and worst of all instability!

Cutoff frequency	0.02	0.05	0.10	0.25	0.40	0.45	0.48
Maximum poles	4	6	10	20	10	6	4

Summary

- Review of single pole recursive filters
- Introduction to Butterworth and Chebyshev Filters
- Designing Butterworth and Chebyshev filters using MATLAB
- Limitations of higher order recursive filters.