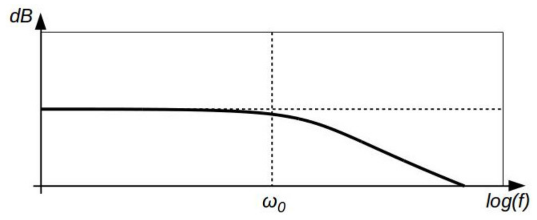
Programming Assignment #6

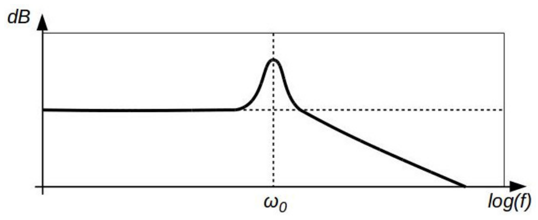
CS 246, Fall 2018

Due Wednesday, November 7

In this assignment you will modify a higher order Butterworth low pass filter to incorporate resonance near the cutoff frequency.



Without Resonance



With Resonance

You will also compare the gain of your filter with the theoretical filter gain. Specifically, you are to implement the following public class interface.

```
class LowPassRes : public Filter {
  public:
    LowPassRes(float f=0, float r=0, float R=44100);
    void setFrequency(float f);
    void setResonance(float r);
    float operator()(float x);
    float theoreticalGain(float f);
};
```

`LowPassRes(f,r,R)` — (constructor) creates a digital resonant low pass filter with cutoff frequency f, resonance factor r, and sampling rate R.

`setFrequency(f)` — (re)sets the cutoff frequency of the filter to f.

`setResonance(r)` — (re)sets the resonance factor of the filter to r. This value should assumed to be in the range $[0, 1]$ (no error checking needs to be performed).

`operator()(x)` — returns the output of the filter for the given input value x.

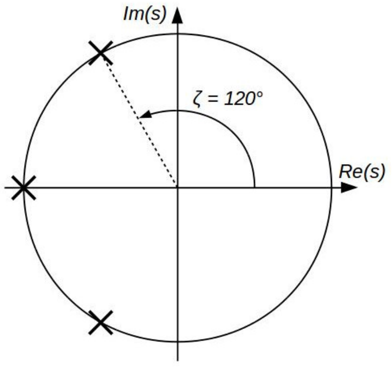
`theoreticalGain(f)` — returns the theoretical gain of the filter at the frequency f, as determined by the s–space transfer function of the filter.

You are to implement a modified Butterworth low pass filter of order at least third order. If you implement a third order filter, you will receive a maximum grade of 93%. To receive full credit, you should implement a fourth order filter (or higher).

Your submission for this assignment should consist of two files: (1) an interface file named `LowPassRes.h`, and (2) an implementation file named `LowPassRes.cpp`. The *public* portion of the `LowPassRes` class must be exactly as dictated above. You may include the header file `LowPassRes.h` as well as any *standard* C++ header file.

Third Order Butterworth Filter

The normalized third order Butterworth filter has three poles in the s–plane: $s = -1$ and $s = \cos\zeta \pm i \sin\zeta$, where $\zeta = \frac{2\pi}{3}$. That is, the normalized transfer function is



$$H(s) = \frac{1}{(s+1)(s^2 - 2s\cos\zeta + 1)}$$

Recall that a normalized transfer function has cutoff frequency of $\omega_0 = 1\,\text{rad/s}$. To convert a normalized transfer function to a general one, we make the replacement

$$s \leftarrow s/\omega_0$$

That is, the above normalized transfer function becomes the general transfer function

$$\boxed{H(s) = \frac{\omega_0^3}{(s+\omega_0)(s^2 - 2\omega_0 s\cos\zeta + \omega_0^2)}}$$

for a third order Butterworth low pass filter with cutoff frequency ω_0.

Of course if $\zeta = \frac{2\pi}{3}$, then $\cos\zeta = -\frac{1}{2}$. However, we will incorporate resonance by letting ζ be closer to the imaginary axis. That is we take

$$\boxed{\zeta = \tfrac{\pi}{6}(4 - r), \quad \text{where } 0 \le r \le 1}$$

where r is the *resonance factor*. For $r = 0$, we get the usual third order Butterworth filter. Larger values of r will increase the gain near the cutoff frequency. A value of $r = 1$ gives (theoretically) infinite gain at the cutoff frequency.

Our usual procedure to get a digital filter from a transfer function in s–space is to (1) find the corresponding differential equation, and (2) apply the discretization rules to get a recurrence relation for the filter output. However, the differential equation in this case will involve the third derivative of the output, $\frac{d^3y}{dt^3}$. Here is the discretization rule for the third derivative:

$$\boxed{\frac{d^3y}{dt^3} \to R^3(y_n - 3y_{n-1} + 3y_{n-2} - y_{n-3})}$$

Note that to compute the theoretical gain of our modified Butterworth filter at the frequency f, we only need to compute the value of

$$|H(i2\pi f)| = \left| \frac{\omega_0^3}{(i2\pi f + \omega_0)[(i2\pi f)^2 - 2\omega_0(i2\pi f)\cos\zeta + \omega_0^2]} \right|$$

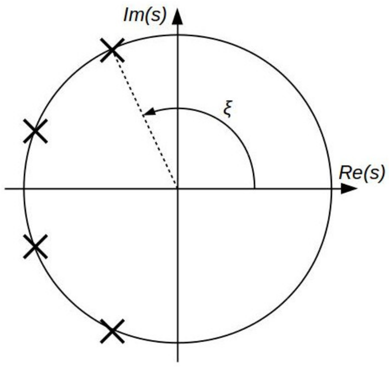
While it is possible to simplify this expression, you actually do not need to. The C++ STL `complex` class will do the work for you. For example, to compute $|i2\pi f + \omega_0|$, you can write

```
float f, omega0;  // assume these are given meaningful values
const complex<float> I(0,1);
const float PI = 4.0f*atan(1.0f);
complex<float> H = I*2.0f*PI*f + omega0;
float absH = abs(H);
```

Here the (overloaded) `abs` function computes the complex modulus.

Fourth Order Butterworth Filter

The normalized fourth order Butterworth filter has four poles in the s–plane: $\cos\frac{7\pi}{8} \pm i\sin\frac{7\pi}{8}$ and $\cos\xi \pm i\sin\xi$, where $\xi = \frac{5\pi}{8}$.



We can add resonance by taking ξ closer to the imaginary axis, namely

$$\xi = \tfrac{\pi}{8}(5 - r) \quad \text{where } 0 \leq r \leq 1$$

To complete the description of the filter, you will need to find the general transfer function from the above information. And to find the recurrence relation of the digital filter, you will need to discretize the corresponding differential equation. The discretization rule for the fourth derivative is:

$$\frac{d^4y}{dt^4} \to R^4(y_n - 4y_{n-1} + 6y_{n-2} - 4y_{n-3} + y_{n-4})$$