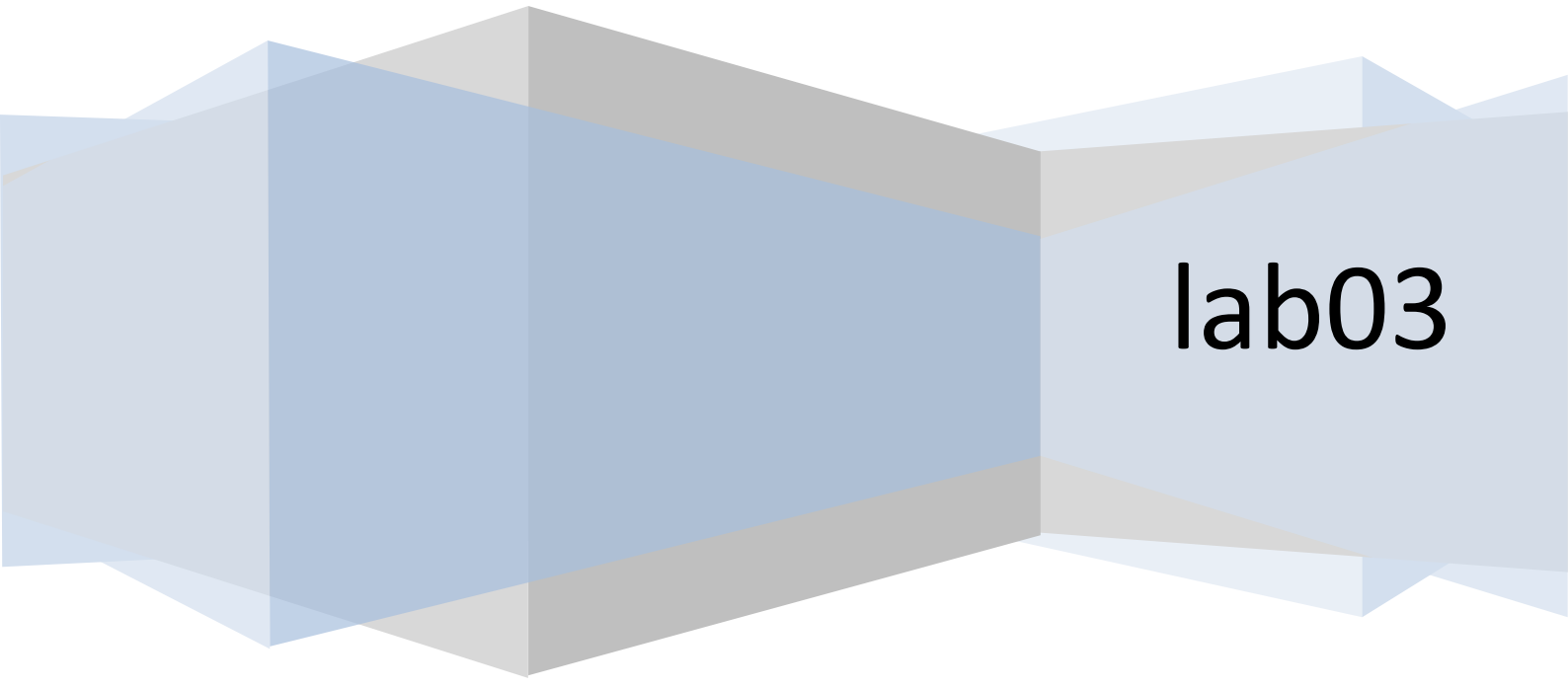
New York University

Digital Signal Processing LAB

lab submission

Drumil Mahajan, dm3804@nyu.edu



Submissions:

Problem Statement :

Modify filtering_paInt16_a.py to avoid run-time overflow errors even if gain is very high. Do this by inserting an if statement to verify if the sample value is in the allowed range. If the value is not in the allowed range, then set the sample value to its maximum (positive or negative) allowed value, before writing the sample value to the audio stream. Test your program by setting the gain to a high value. What effect does this have on the sound produced by the program?

Solution: File for the solution is filtering_paInt16_a.py.

Snapshot of the function :

```
def clip_n(y , n):
    '''
    This function takes two arguments.
    y is a floating point number
    n is a integer, which is equal to number of bits.

    This funtions bound the value y between max and min of n bit signed number
    '''
    max = 2**(n-1) - 1
    min = -2**(n)
    if y > max:
        return max
    elif y < min:
        return min
    else:
        return y
```

Figure 1

This function takes two parameters. One is y which is the value to be clipped between a maximum and a minimum. The other parameter is n which is the number of bits in the output signal. In the question mentioned. The value of n = 16.

Effect of increasing the gain on the sound produced by the program :

After applying the clipping function which limits the value of the output between the maximum and minimum of a 16 bit sample which is 2^15 - 1 and -2^15 respectively, the output value cannot cross these limits else it would result in an error.

Although when high gain is applied, a lot of values who were supposed to be higher or lower than the MAX or MIN are clipped to the MAX and MIN respectively and hence it introduces DISTORTION in the sound signal.

It is also noticed that the even after applying high gain, the speech signal is preserved as it depends n the zero crossing but there is a lot of DISTORITION introduced.

Problem Statement:

Describe how to design two second-order filters (with same resonant frequency f1) so that the rise-time and decay-time of the impulse response are different? (The two filters will have different pole radii.) Implement two filters in cascade in Matlab and in real-time in Python/PyAudio, that generates an impulse response with short rise-time and slow decay-time.

Solution: Files attached as :

Python file: filtering_paInt16_a.py : The filter is implemented as two second order filter in series. The output of the first filter which is y0 is used as an input for the second filter.

Snapshot of the filter :

```
for n in range (0, N):

# Use impulse as input signal
    if n == 0:
        x0 = 1.0
    else :
        x0 = 0.0

    # Difference equation
    y0 = x0 - a1 * y1 - a2 * y2

    # Passing the output of first difference equation as the
    # input to the second difference equation.

    z0 = y0 - a1 * z1 - a2 * z2

    # Delays
    y2 = y1
    y1 = y0
    z2 = z1
    z1 = z0


    # Output
    out = clip_n( gain * y0 , 16 )
    print out
    str_out = struct . pack ('h', out) # 'h' for 16 bits
    stream . write ( str_out )
```

Figure 2

Matlab File: lab03_fourth_order_filter.m : The filter is implemented as two second order filter in series. The output of the first filter is the input of the second filter. The rise and fall time of the filter is controlled by changing the pole radii. As the pole radii is increased towards unity, the fall time/decay time increases.

The first filter has a pole radii of 0.999 and second filter has pole radii of 0.998. The decay time or the time constant is determined by the maximum of the both pole radii.

Snapshot of filters in matlab:

Filter 1:

```
%% Implementing the recursive filter

Fs = 8000 ;    % Sampling frequency.
F1 = 400;      % Frequency of the signal we want. The sampling
               % frequency must be more than twice of frequency we want to
               % sample. Nyquist theorm
f1 = F1/Fs;    % Normalized frequency. Ratio of Frequency/ And sampling fre
om1 = 2 * pi * f1 ; % angular frequency
r1 = 0.999;    % Radius of the pole
a1 = [1 -2*r*cos(om1) r1^2];
b1 = 1;
y = filter(b1, a1, x);
```

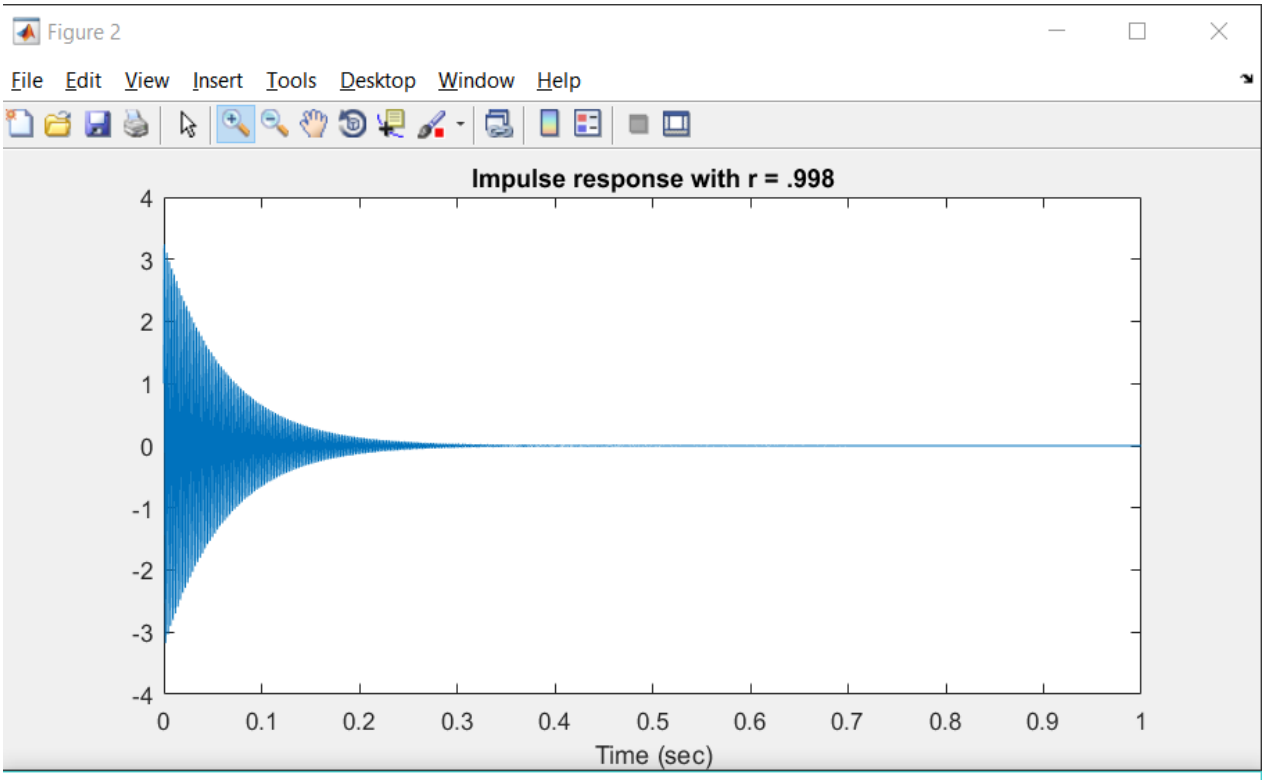
Filter 2:

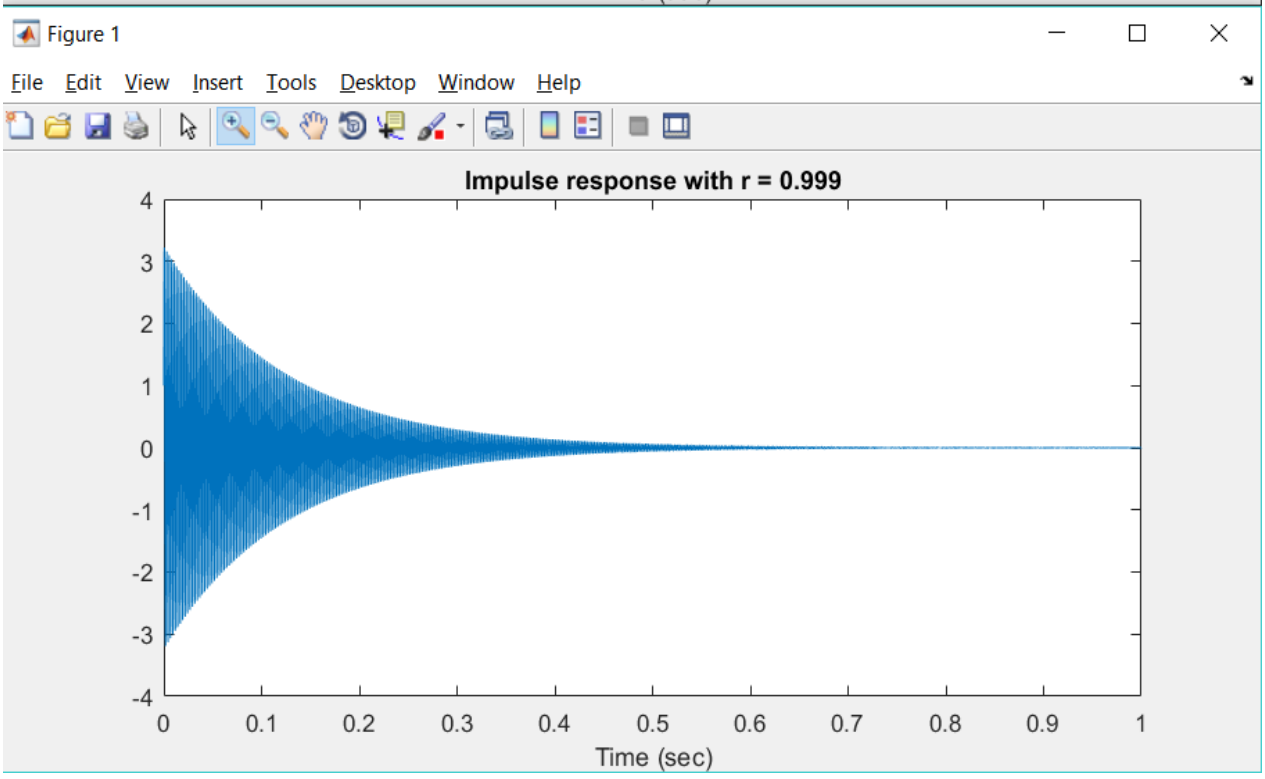
```
%% Implementing fourth order recursive filter by passing the output of firs

% The input has already been filtered once so passing hte output through
% the filter again will serve the purpose.
% The pole radus of the two filters is kept different.

F1 = 400;      % Frequency of the signal we want. The sampling
               % frequency must be more than twice of frequency we want to
               % sample. Nyquist theorm
f1 = F1/Fs;    % Normalized frequency. Ratio of Frequency/ And sampling fre
om1 = 2 * pi * f1 ; % angular frequency
r2 = 0.998;    % Radius of the pole
a2 = [1 -2*r*cos(om1) r2^2];
b2 = 1;
z = filter(b2, a2, y);
```

Impulse responses :





Problem Statement:

MATLAB Graphical User Interface (GUI). Make a MATLAB GUI that allows the use to control the pole positions of a second-order difference equation. The user should be able to control the pole radius and pole angle by sliders. The GUI should display
_ pole/zero diagram
_ impulse response
_ frequency response (magnitude)

Comment on the relationship between the pole position and the impulse response and the frequency response.

Solution : Files attached is matlab_GUI.m

Relationship between pole position and the impulse response and frequency response.

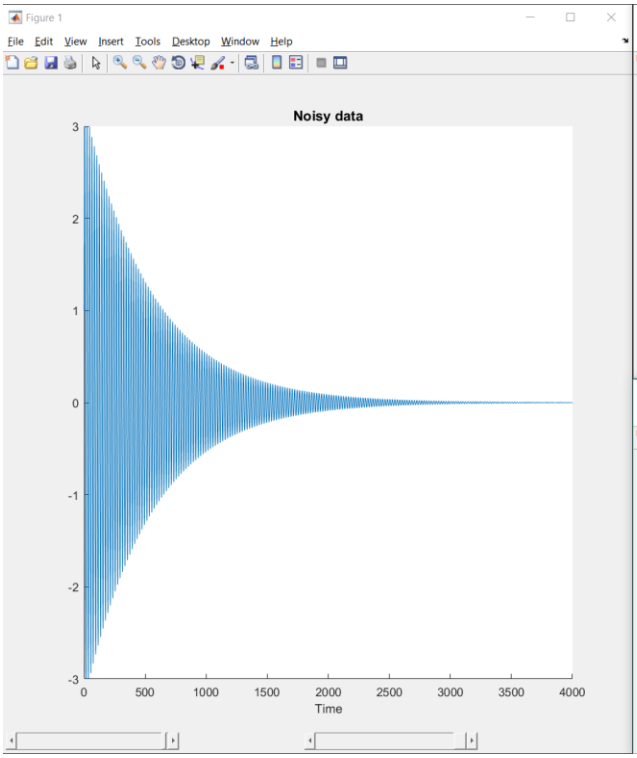
For any pole in a z-plane, there are two parameters associated to it. One is the pole radius and another is the angle the pole makes with the real axis. The angle the pole makes with the real axis determines the frequency and the radius determines the decay time.

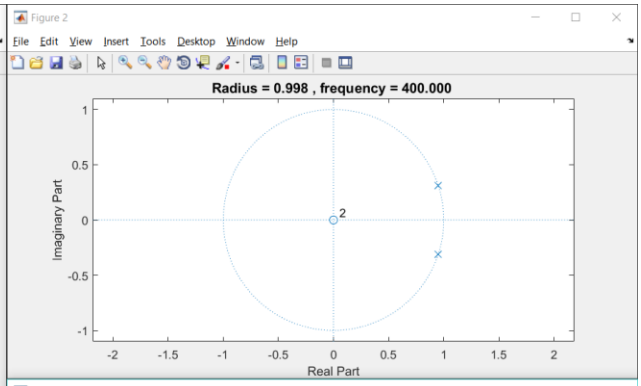
- As the angle of the pole increases, the frequency increases.
- As the radius of the pole increases, the decay time increases.

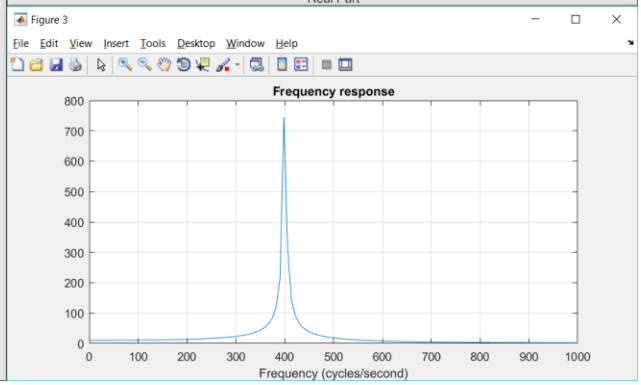
So as we increase the pole radius using the slider, the decay time of the impulse response increases.

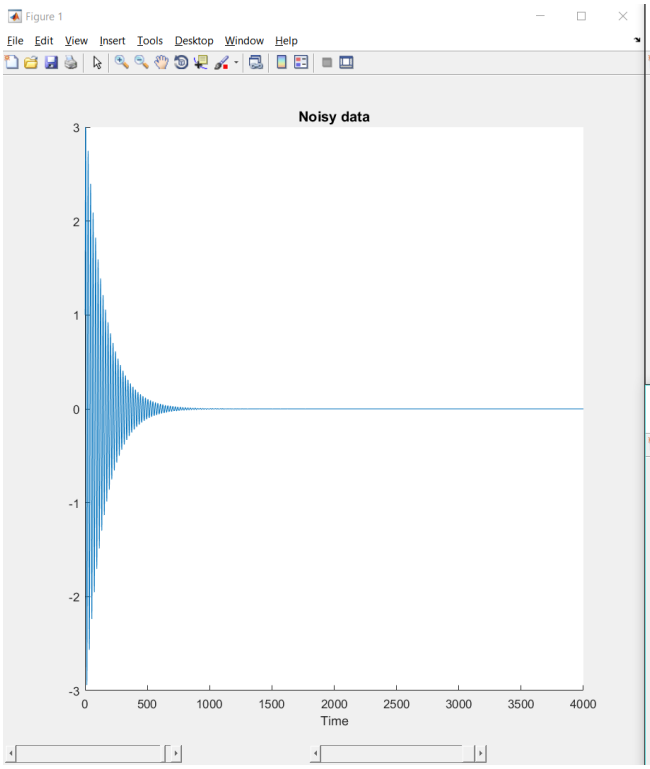
Plots on the next page:

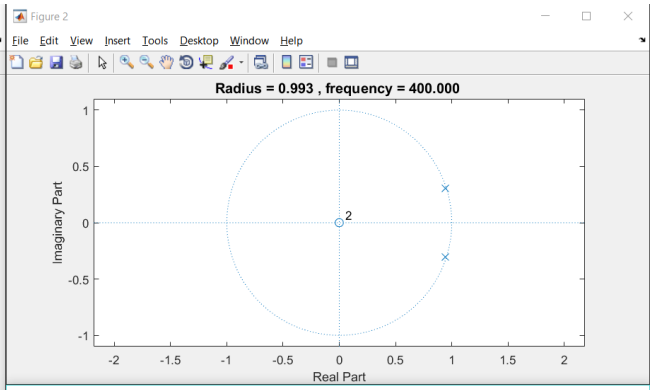
When pole radius decreases, the decay time decreases but the frequency response remains same as the frequency response does not depend on the pole radius.

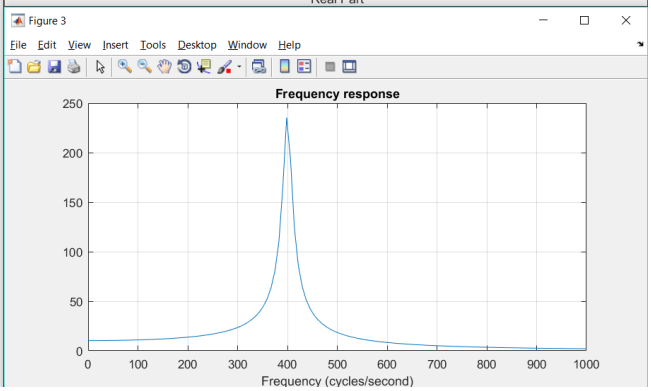












When pole angle is decreased :The frequency response changes as the frequency depends on the angle of the pole but the decay time remains constant.