System Functions - LTI Systems

Arnav Goyal - 251244778

March 8, 2024

The Difference Equations

The systems shown in (1) and (2) below will be repeatedly referred to within this lab manual as the first system, and the second system respectively.

$$y(n) - 1.9y(n-1) + 1.16y(n-2) - 0.224y(n-3) = 3.5x(n) - 4.75x(n-1) + 1.58x(n-2) \tag{1}$$

$$y(n) - 0.3y(n-1) - 0.2y(n-2) + 0.35y(n-3) = 2.5x(n) - 0.6x(n-1) - 0.8x(n-2) \tag{2}$$

Analytical System Functions

This is technically question 2, but lets start by analytically obtaining th required results for each system. Let's start by obtaining the z-domain system function for (1) and (2).

Note: In the below equations the symbols Y and X are shorthand for $Y(z)$ and $X(z)$ respectively.

System 1

$$Y - 1.9Yz^{-1} + 1.16Yz^{-2} - 0.224Yz^{-3} = 3.5X - 4.75Xz^{-1} + 1.58Xz^{-2}$$

$$Y\left[1 - 1.9z^{-1} + 1.16z^{-2} - 0.224z^{-3}\right] = X\left[3.5 - 4.75z^{-1} + 1.58z^{-2}\right]$$

$$T_1(z) = \frac{Y}{X} = \frac{3.5 - 4.75z^{-1} + 1.58z^{-2}}{1 - 1.9z^{-1} + 1.16z^{-2} - 0.224z^{-3}} \tag{3}$$

System 2

$$Y - 0.3Yz^{-1} - 0.2Yz^{-2} + 0.35Yz^{-3} = 2.5X - 0.6Xz^{-1} - 0.8Xz^{-2}$$

$$Y\left[1 - 0.3z^{-1} - 0.2z^{-2} + 0.35z^{-3}\right] = X\left[2.5 - 0.6z^{-1} - 0.8z^{-2}\right]$$

$$T_2(z) = \frac{Y}{X} = \frac{2.5 - 0.6z^{-1} - 0.8z^{-2}}{1 - 0.3z^{-1} - 0.2z^{-2} + 0.35z^{-3}} \tag{4}$$

Thus we can see that systems (1) and (2) take the form of (3) and (4) respectively.

Inverse Transformation into the Discrete-Time Domain

Now that we have the system functions, we should inverse transform them into the discrete-time domain, and use those new equations to obtain the impulse and step response for each system.

Note: The inverse transformation will be done by hand after obtaining partial fraction expansion terms from `MATLAB` . This script will be shown at the end of this section

This is the resulting residue, pole and direct term values from the MATLAB script

```
SYSTEM 1
Residues:
0.500 + 0.000j
1.000 + 0.000j
2.000 + 0.000j

Poles:
0.800 + 0.000j
0.700 + 0.000j
0.400 + 0.000j

Direct Terms:

---------------
SYSTEM 2
Residues:
1.000 + 0.500j
1.000 + -0.500j
0.500 + 0.000j

Poles:
0.500 + 0.500j
0.500 + -0.500j
-0.700 + 0.000j

Direct Terms:

---------------
```

System 1

$$T_1(z) = \frac{0.5}{1-0.8z^{-1}} + \frac{1}{1-0.7z^{-1}} + \frac{2}{1-0.4z^{-1}}$$
$$T_1(n) = [0.5(0.8)^n + (0.7)^n + 2(0.4)^n]u(n) \tag{5}$$

System 2

$$T_2(z) = \frac{1+0.5j}{1-(0.5+0.5j)z^{-1}} + \frac{1-0.5j}{1-(0.5-0.5j)z^{-1}} + \frac{0.5}{1+0.7z^{-1}}$$
$$T_2(n) = \left[2.236(0.707)^n \cos(0.463+0.785n) + 0.5(-0.7)^n\right] u(n) \tag{6}$$

The MATLAB script used to do this is titled 'agoyal57_LAb6_PFD.m', the code is shown below:

```
%% ensure clear data
clear;

%% define both system with their coefficients
num1 = [3.5, -4.75, 1.58];
den1 = [1, -1.9, 1.16, -0.224];
num2 = [2.5, -0.6, -0.8];
den2 = [1, -0.3, -0.2, 0.35];

%% obtain partial fractions & write output to file 'output.txt'
file = fopen('output.txt', 'w');
printSystem(num1, den1, 1, file);
printSystem(num2, den2, 2, file);

%% define write to output file function
function printArr(array, label, file)
fprintf(file, '%s:\n', label);
for i = 1:length(array)
fprintf(file, '%.3f + %.3fj\n', real(array(i)), imag(array(i)));
end
fprintf(file, '\n');
end

%% helper function for above
function printSystem(num, den, label, file)
fprintf(file, 'SYSTEM %d\n', label);
[r, p, k] = residuez(num, den);
printArr(r, 'Residues', file);
printArr(p, 'Poles', file);
printArr(k, 'Direct Terms', file);
fprintf(file, '---------------\n');

end
```

Analytically Obtaining Step & Impulse Response

I decided to do this by creating a `Python` script that implements the analytical algorithm of finding the impulse response and then the step response of each system. It performs the algorithm below.

$$\text{Impulse Response}[n] = T(n)$$

$$\text{Step Response}[0] = \text{Impulse Response}[0]$$
$$\text{Step Response}[n] = \text{Impulse Response}[n] + \text{Step Response}[n-1]$$

The `Python` script, titled '`agoyal57_Lab6_ImpStepResp.py`' is shown below.

```
import numpy as np

# define the analytic dt system functions
def sys1(n):
        res = 0.5 * (0.8) ** n + (0.7) ** n + 2 * (0.4) ** n
        return res
def sys2(n):
        res = 2.236 * (np.sqrt(2)/2) ** n * np.cos(0.463 + 0.785 * n)
                + 0.5 * (-0.7) ** n
        return res

# func to calculate impulse response
def impulse(system, iterations):
        resp = []
        for n in range(iterations):
        val = system(n)
        val_round = float(f'{val:.3f}')
        resp.append(val_round)
        return resp

# func to calculate step response
def step(system, iterations):
        resp = []
        imp = impulse(system=system, iterations=iterations)

        # the first element in step response is same as impulse resp
        resp.append(imp[0])
        for n in range(1, iterations):
        val = resp[n-1] + imp[n]
        val_round = float(f'{val:.2f}')
        resp.append(val_round)
```

```
        return resp

# func to get impulse and step response written in a file
def fileWrite(file, system, label):
        file.write(f'{label.upper()} - IMPULSE RESPONSE\n')
        file.write(str(impulse(system, 10)))
        file.write('\n')
        file.write(f'{label.upper()} - STEP RESPONSE\n')
        file.write(str(step(system, 10)))
        file.write('\n')

## BEGIN MAIN
file = open('responses.txt', 'w')
fileWrite(file, sys1, 'system 1')
fileWrite(file, sys2, 'system 2')
file.close()
## END MAIN
```

The output is shown below

```
SYSTEM 1 - IMPULSE RESPONSE
[3.5, 1.9, 1.13, 0.727, 0.496, 0.352, 0.257, 0.19, 0.143, 0.108]
SYSTEM 1 - STEP RESPONSE
[3.5, 5.4, 6.53, 7.26, 7.76, 8.11, 8.37, 8.56, 8.7, 8.81]
---
SYSTEM 2 - IMPULSE RESPONSE
[2.501, 0.152, -0.254, -0.921, -0.38, -0.21, 0.183, 0.146, 0.154, 0.011]
SYSTEM 2 - STEP RESPONSE
[2.501, 2.65, 2.4, 1.48, 1.1, 0.89, 1.07, 1.22, 1.37, 1.38]
```

Programmatically Finding the Impulse, Step Response & Z-Plane Plots

To do this, a `MATLAB` script titled '`agoyal57_Lab6_Plots.m`' was written, the code is shown below.

```
%% ensure clear data
clear;

%% define both system with their coefficients
num1 = [3.5, -4.75, 1.58];
den1 = [1, -1.9, 1.16, -0.224];
num2 = [2.5, -0.6, -0.8];
den2 = [1, -0.3, -0.2, 0.35];

%% get imp/step responses
```

```
imp1 = filter(num1, den1, [1, zeros(1, 50)]);
step1 = filter(num1, den1, ones(1, length(imp1)));
imp2 = filter(num2, den2, [1, zeros(1, 50)]);
step2 = filter(num2, den2, ones(1, length(imp2)));

%% Plot responses of system 1
figure;
subplot(2,1,1);
stem(0:50, imp1);
title('Impulse Response - System 1');
xlabel('n');
ylabel('Amplitude');

subplot(2,1,2);
stem(0:50, step1);
title('Step Response - System 1');
xlabel('n');
ylabel('Amplitude');

%% Plot responses of system 2
figure;
subplot(2,1,1);
stem(0:50, imp2);
title('Impulse Response - System 2');
xlabel('n');
ylabel('Amplitude');

subplot(2,1,2);
stem(0:50, step2);
title('Step Response - System 2');
xlabel('n');
ylabel('Amplitude');

%% Plot Zero-Pole of System 1
figure;
zplane(num1, den1);
legend('Zero', 'Pole')
title('Zero Pole Plot - System 1');

%% Plot Zero-Pole of System 2
figure;
zplane(num2, den2);
legend('Zero', 'Pole')
title('Zero Pole Plot - System 2');
```

These are the Impulse/Step Responses, and Zero-Pole (z-plane) Plots for Each System: We can see that our plotted results and our analytical results are essentially the same. This makes sense because this is probably what the `MATLAB` function does under the hood.

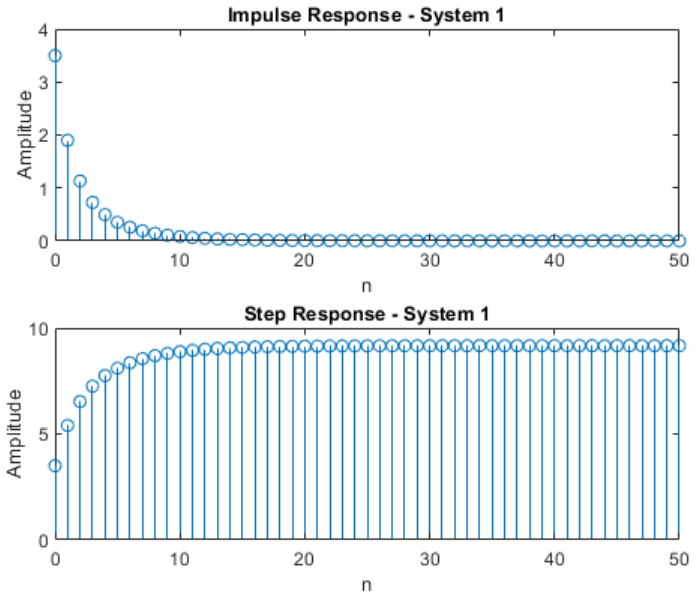

Figure 1: Impulse and Step Response Plots of System 1

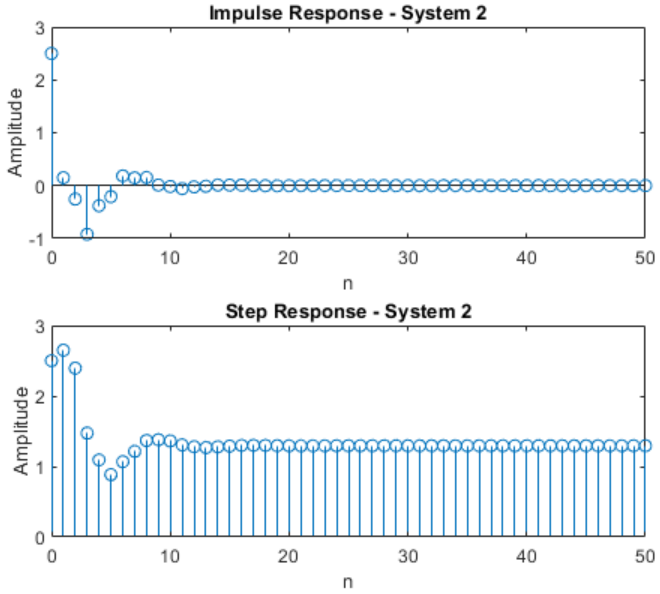

Figure 2: Impulse and Step Response Plots of System 2

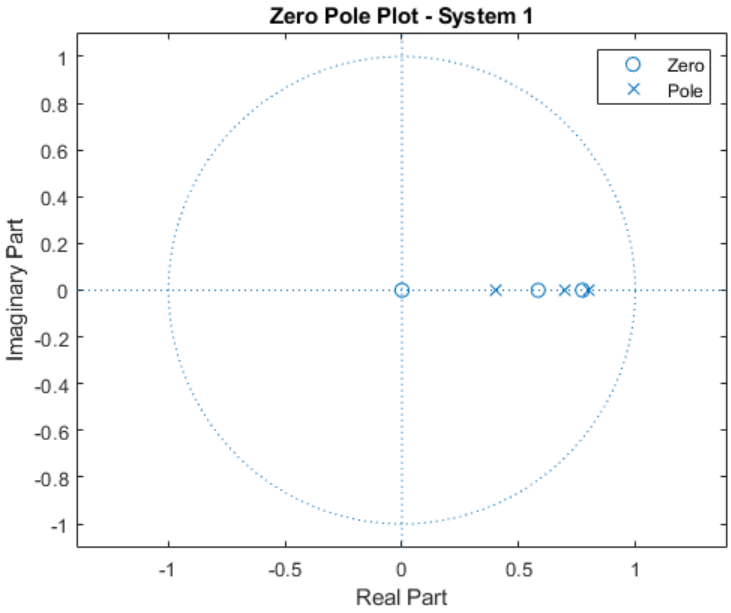

Figure 3: Z-Plane Plot of System 1

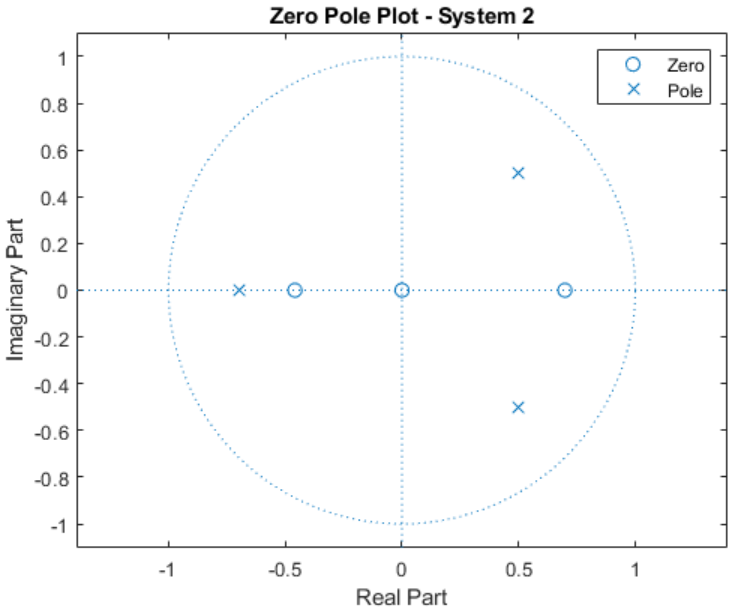

Figure 4: Z-Plane Plot of System 2