Experiment No 04

BECS 32461

Paper D

IMPLEMENTATION OF Z-TRANSFORM AND INVERSE Z-TRANSFORM

Student Name: W. K. G. K. Jayawardana

Student No: EC/2021/006

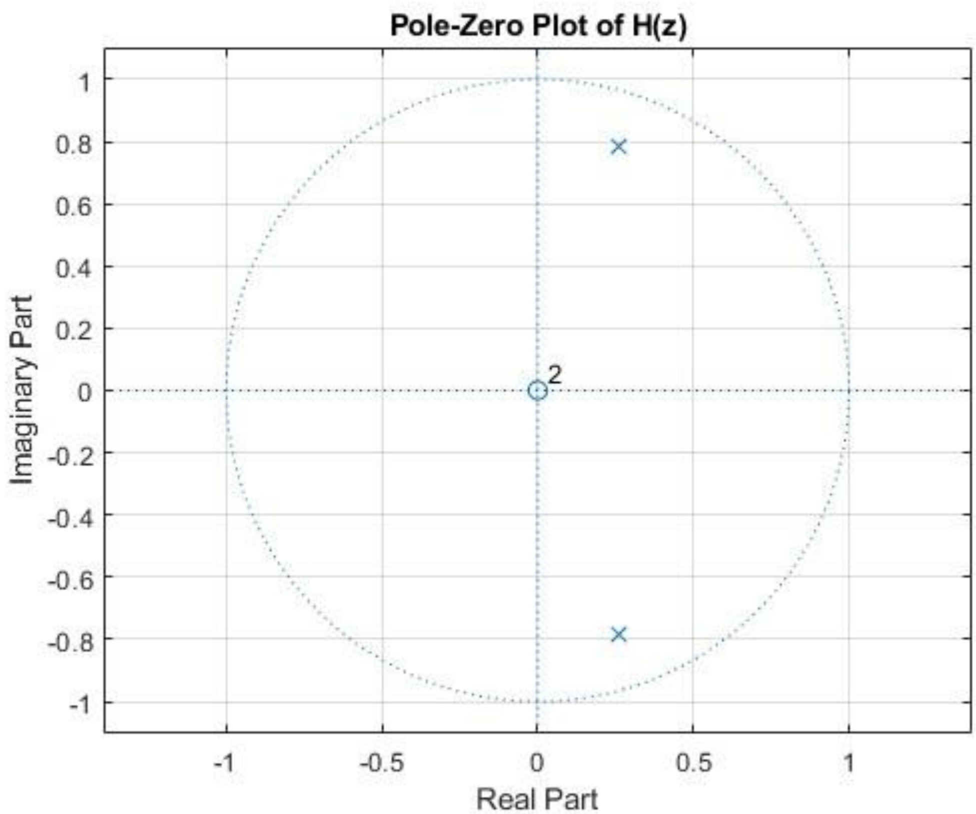
Date Performed: 2025/10/24

Date Submitted: 2025/10/24

PROCEDURE

F01.

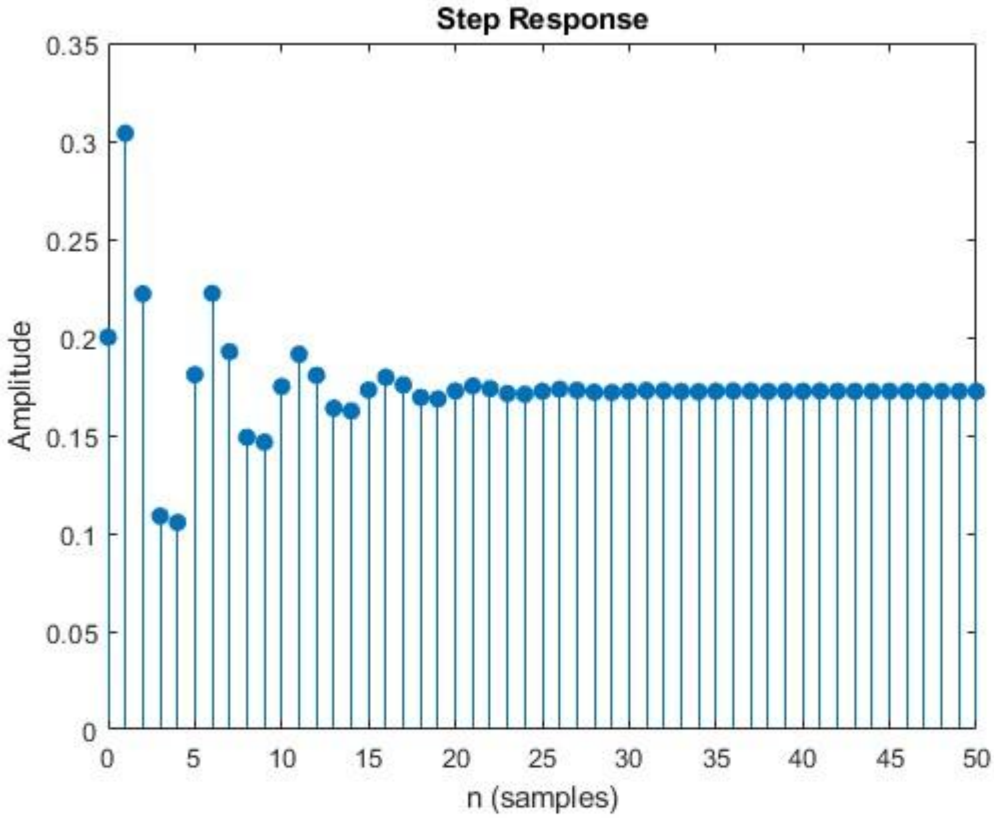
```
b = 0.2;
a = [1, -0.52, 0.68];

H = tf(b, a, -1);
disp('Transfer Function H(z):'); H

figure;
zplane(b, a);
title('Pole-Zero Plot of H(z)');
grid on;
```




```
>> F01
Transfer Function H(z):

H =

          0.2
  -------------------
  z^2 - 0.52 z + 0.68
```

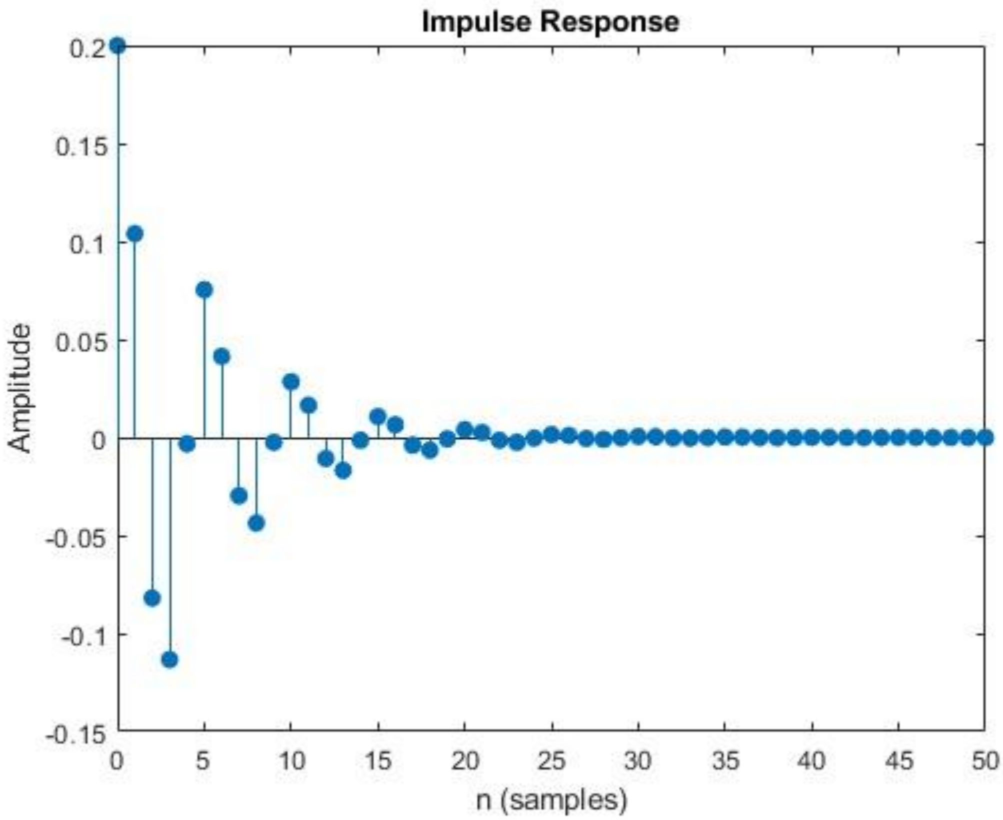
F02.

```
b = 0.2;
a = [1, -0.52, 0.68];

H = tf(b, a, -1);
disp('Transfer Function H(z):'); H

figure;
stepz(b, a);
```



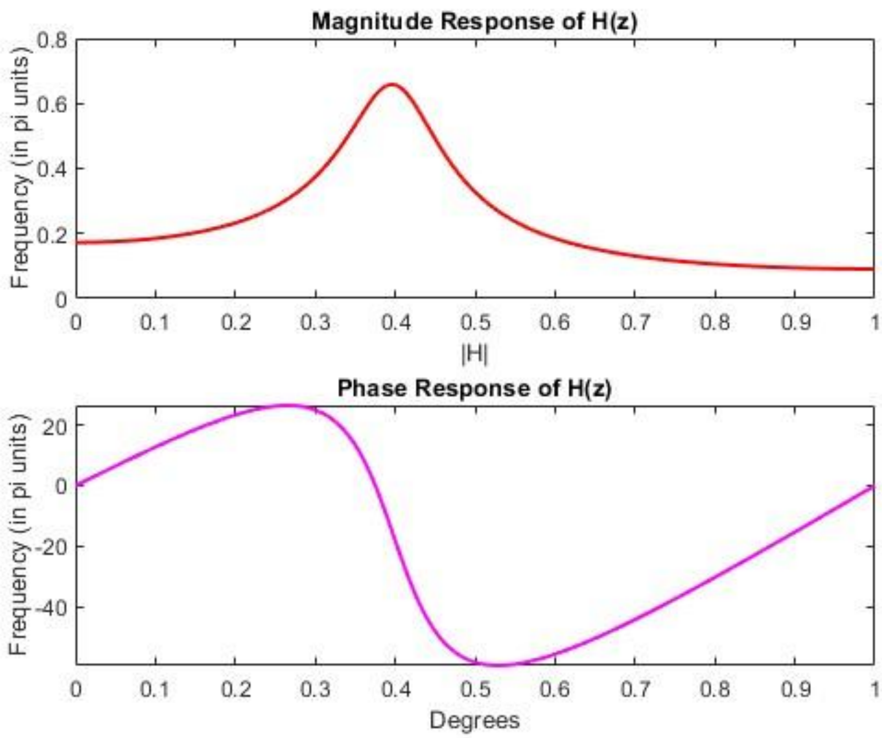
F03.

```
b = 0.2;
a = [1, -0.52, 0.68]; s

H = tf(b, a, -1);
disp('Transfer Function H(z):'); H

impz(b, a);
```



F04.

```
b = 0.2;
a = [1, -0.52, 0.68];

w = linspace(0, pi, 500);
[h, w] = freqz(b, a, w);
magH = abs(h);
phaH = angle(h) * 180/pi;

poles = roots(a);
disp('Poles of H(z):');
disp(poles);

subplot(2,1,1);
plot(w/pi,magH, 'r-', 'LineWidth', 1.5)
xlabel("|H|")
ylabel("Frequency (in pi units)")
title('Magnitude Response of H(z)');

subplot(2,1,2);
plot(w/pi,phaH, 'm-', 'LineWidth', 1.5)
xlabel("Degrees")
ylabel("Frequency (in pi units)")
title('Phase Response of H(z)');
```



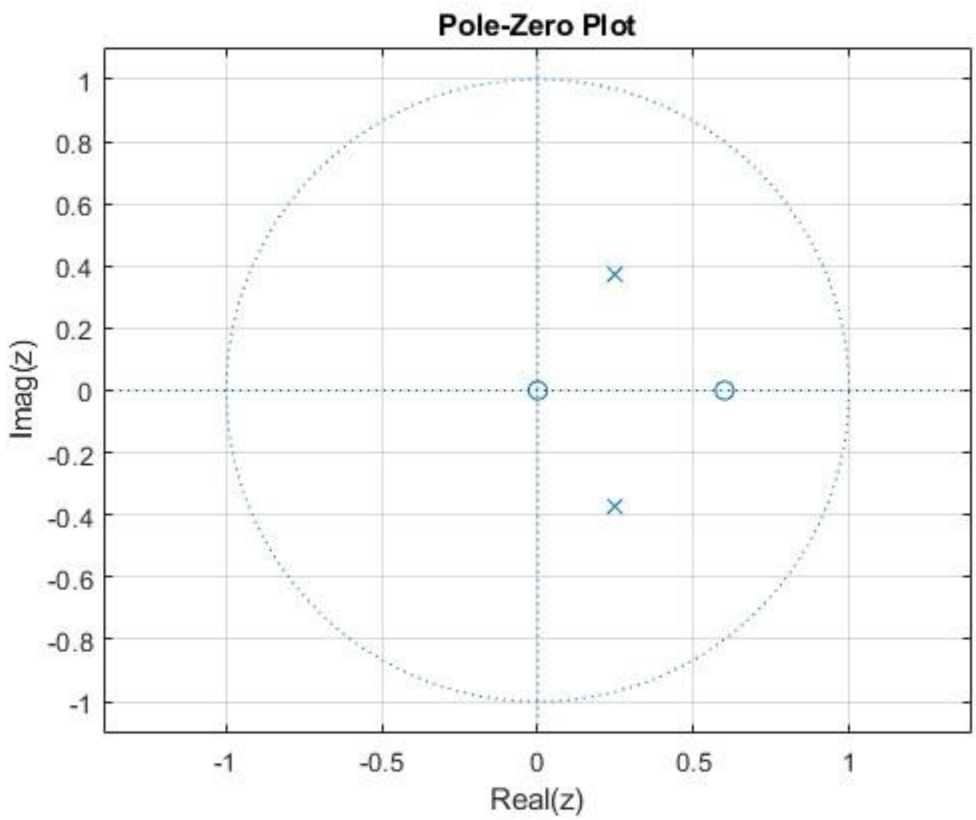

```
>> F04
Poles of H(z):
   0.2600 + 0.7826i
   0.2600 - 0.7826i
```

EXERCISE

E01.

a.

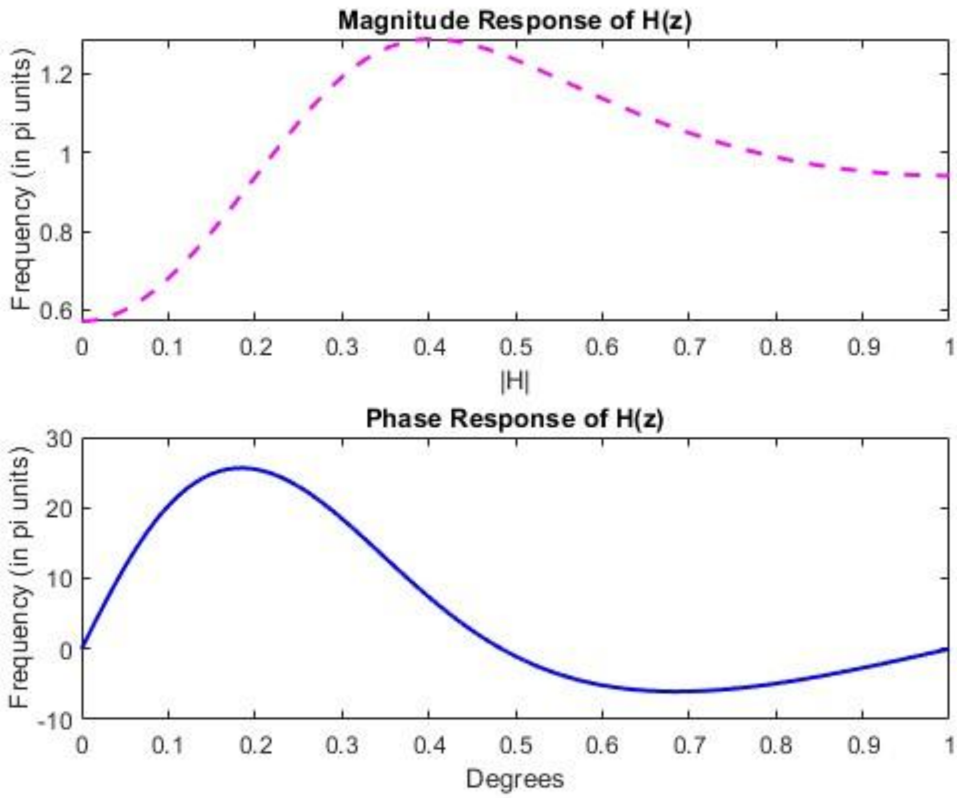
```
b = [1, -0.6,0];
a = [1, -0.5, 0.2];

figure;
zplane(b, a);
title('Pole-Zero Plot');
xlabel('Real(z)'); ylabel('Imag(z)');
grid on;
```



b.

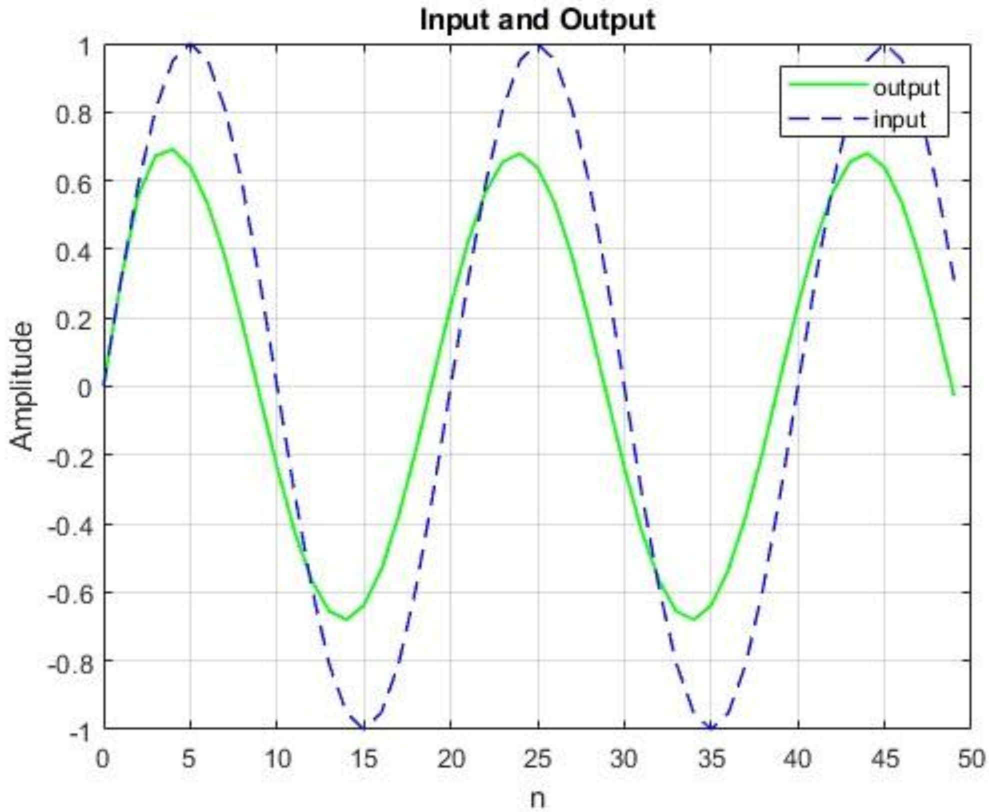
```
b = [1, -0.6 ,0];
a = [1, -0.5, 0.2];

w = linspace(0, pi, 500);
[h, w] = freqz(b, a, w);
magH = abs(h);
phaH = angle(h) * 180/pi;

subplot(2,1,1);
```

```
plot(w/pi,magH, 'm--', 'LineWidth', 1.5)
xlabel("|H|")
ylabel("Frequency (in pi units)")
title('Magnitude Response of H(z)');

subplot(2,1,2);
plot(w/pi,phaH, 'b-', 'LineWidth', 1.5)
xlabel("Degrees")
ylabel("Frequency (in pi units)")
title('Phase Response of H(z)');
```



c.

```
b = [1, -0.6 ,0];
a = [1, -0.5, 0.2];

n = 0:49;
x = sin(0.1*pi*n);
y = filter(b, a, x);

figure;
plot(n, y, 'g-', 'LineWidth', 1.2); hold on;
plot(n, x, 'b--', 'LineWidth', 1.0);
title('Input and Output');
xlabel('n');
ylabel('Amplitude');
legend('output','input');
grid on;
```



E02.

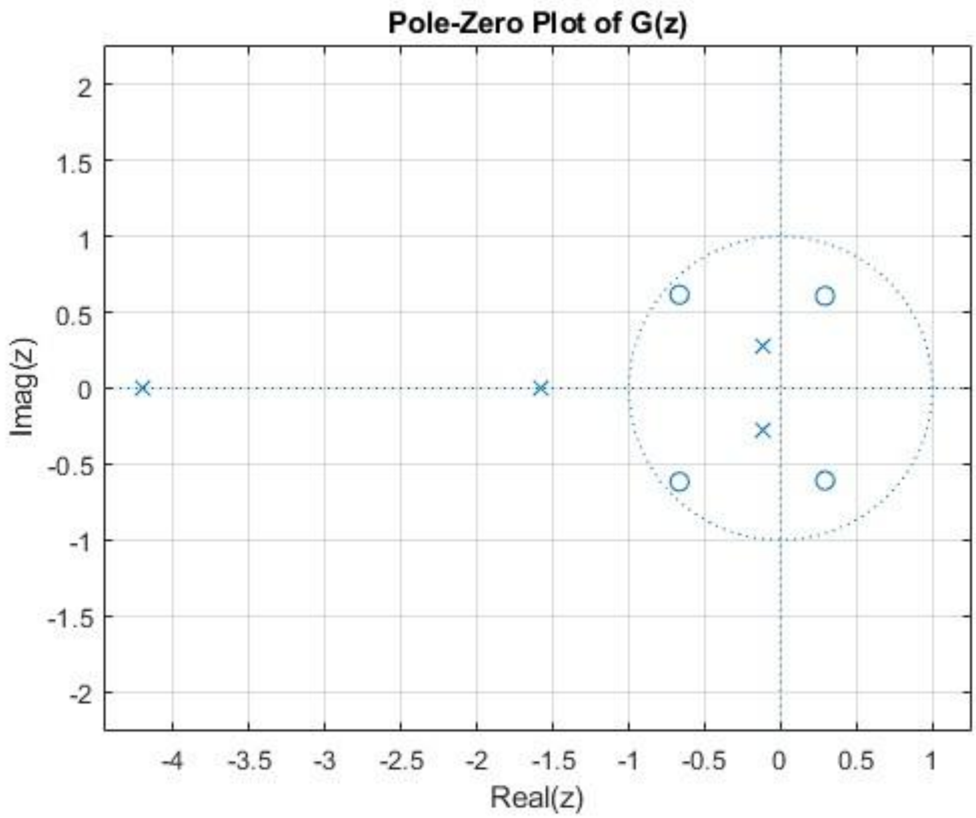
a.

```
b = [4, 3, 2, 0.5, 1.5];
a = [1, 6, 8, 2, 0.6];
[R, P, K] = residuez(b, a);

fprintf('Residues (R):\n'); disp(R);
fprintf('Poles (P):\n'); disp(P);
fprintf('Direct terms (K):\n'); disp(K);
```

```
>> E02a
Residues (R):
   5.7111 + 0.0000i
  -2.0243 + 0.0000i
  -1.0934 + 0.7915i
  -1.0934 - 0.7915i

Poles (P):
  -4.2009 + 0.0000i
  -1.5761 + 0.0000i
  -0.1115 + 0.2796i
  -0.1115 - 0.2796i

Direct terms (K):
    2.5000
```

b.

```
b = [4, 3, 2, 0.5, 1.5];
a = [1, 6, 8, 2, 0.6];

figure;
zplane(b, a);
title('Pole-Zero Plot of G(z)');
xlabel('Real(z)'); ylabel('Imag(z)');
grid on;
```



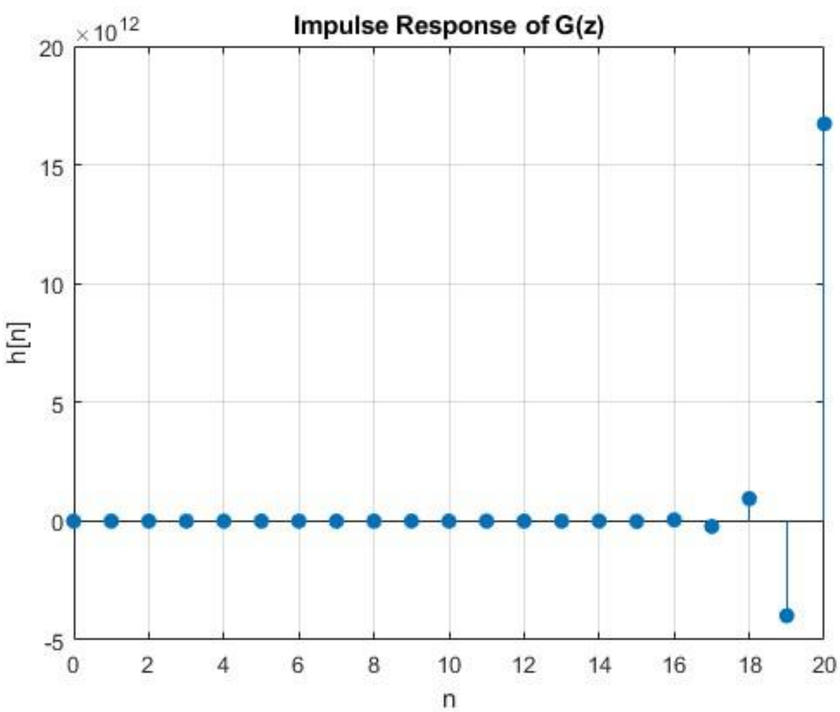
c.

```
b = [4, 3, 2, 0.5, 1.5];
a = [1, 6, 8, 2, 0.6];

p = roots(a);
mag_p = abs(p);

fprintf('Poles:\n'); disp(p);
fprintf('Magnitudes:\n'); disp(mag_p);

if all(mag_p < 1)
    fprintf('\nResult: System is STABLE (all |p| < 1).\n');
elseif any(mag_p > 1)
    fprintf('\nResult: System is UNSTABLE (some |p| > 1).\n');
else
    fprintf('\nResult: MARGINAL (some pole(s) lie on the unit circle: |p| == 1).\n');
```

```
end
>> E02c
Poles:
  -4.2009 + 0.0000i
  -1.5761 + 0.0000i
  -0.1115 + 0.2796i
  -0.1115 - 0.2796i

Magnitudes:
    4.2009
    1.5761
    0.3010
    0.3010


Result: System is UNSTABLE (some |p| > 1).
```

d.

```
b = [4, 3, 2, 0.5, 1.5];
a = [1, 6, 8, 2, 0.6];

n = 0:20;
[h, nout] = impz(b, a, n);

figure;
stem(nout, h, 'filled');
title('Impulse Response of G(z)');
xlabel('n');
ylabel('h[n]');
grid on;
```



E03.

a.

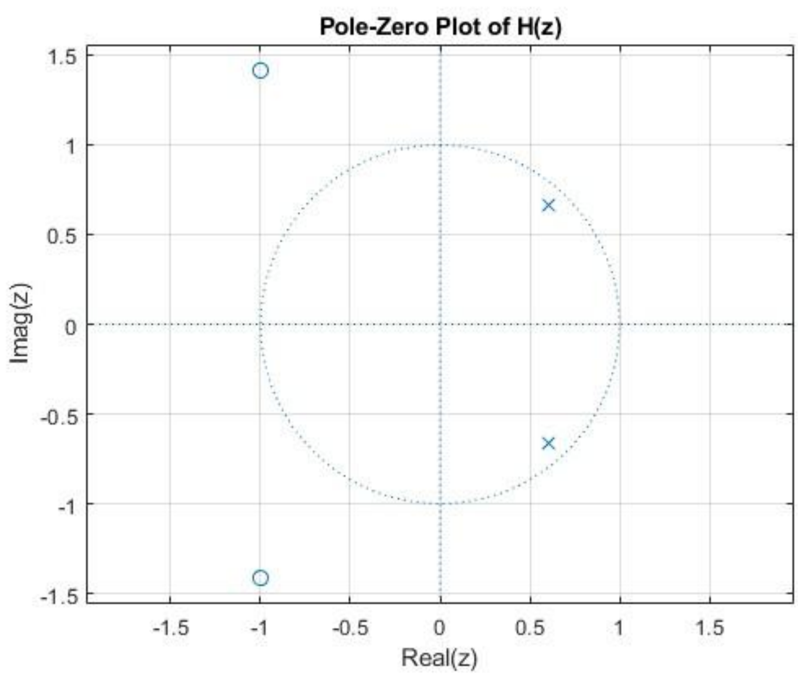
```
syms z n

Hz = (1 + 2*z^-1 + 3*z^-2) / (1 - 1.2*z^-1 + 0.8*z^-2);

x_n = iztrans(Hz, z, n);
x_n_simpl = simplify(x_n);

fprintf('Symbolic inverse Z-transform x[n] = \n');
disp(x_n_simpl);
```

Symbolic inverse Z-transform x[n] =

(15*kroneckerDelta(n, 0))/4 + (16*(-1)^n*4^n*cos(n*(pi - acos((3*5^(1/2))/10))))/(3*(2*5^(1/2))^n) + ((-1)^n*5^(1 - n)*11^(1/2)*(- 3 - 11^(1/2)*1i)^(n - 1)*97i)/66 - ((-1)^n*5^(1 - n)*11^(1/2)*(- 3 + 11^(1/2)*1i)^(n - 1)*97i)/66

b.

```
b = [1, 2, 3];
a = [1, -1.2, 0.8];

figure;
zplane(b, a);
title('Pole-Zero Plot of H(z)');
xlabel('Real(z)'); ylabel('Imag(z)');grid on;
```



c.

```
b = [1, 2, 3];
a = [1, -1.2, 0.8];

[R, P, K] = residuez(b, a);
fprintf('Residues (R):\n'); disp(R);
fprintf('Poles (P):\n'); disp(P);
fprintf('Direct terms (K):\n'); disp(K);

[num, den] = residuez(R, P, K);
fprintf('Reconstructed numerator (z^-1 coeffs):\n'); disp(num);
fprintf('Reconstructed denominator (z^-1 coeffs):\n'); disp(den);
```

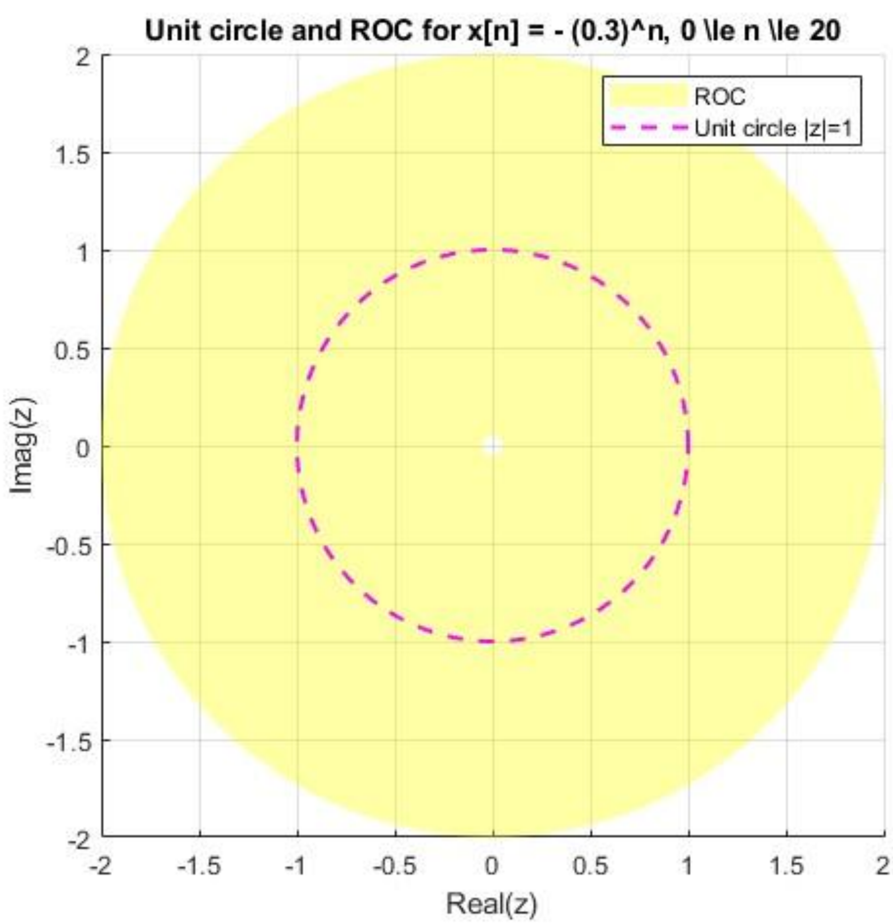
```
>> E03c
Residues (R):
  -1.3750 - 3.6558i
  -1.3750 + 3.6558i

Poles (P):
   0.6000 + 0.6633i
   0.6000 - 0.6633i

Direct terms (K):
    3.7500

Reconstructed numerator (z^-1 coeffs):
     1     2     3

Reconstructed denominator (z^-1 coeffs):
    1.0000   -1.2000    0.8000
```

d.

System is STABLE (all |p| < 1)

E04.

```
r_outer = 2.0;

x_inner = r_inner * cos(theta); y_inner = r_inner * sin(theta);
x_outer = r_outer * cos(theta); y_outer = r_outer * sin(theta);
ux = cos(theta); uy = sin(theta);

figure;
hold on;
axis equal;
fill([x_inner, fliplr(x_outer)], [y_inner, fliplr(y_outer)], 'y','FaceAlpha', 0.35,
'EdgeColor', 'none');

plot(ux, uy, 'm--', 'LineWidth', 1.5);

xlabel('Real(z)');
ylabel('Imag(z)');
title('Unit circle and ROC for x[n] = - (0.3)^n, 0 \le n \le 20');
legend('ROC','Unit circle |z|=1');
grid on;
xlim([-r_outer r_outer]); ylim([-r_outer r_outer]);
```



The Region of Convergence (ROC) is the set of complex z-values where the Z-transform sum converges to a finite value. Its shape (outside a radius, inside, or an annulus) depends on whether the sequence is finite, right-sided, or left-sided, and the ROC never includes poles where the transform blows up. ROC matters for stability because an LTI system is BIBO-stable only if its ROC includes the unit circle ($|z| = 1$), which guarantees the frequency response exists and bounded inputs produce bounded outputs.