# Table of Contents



# Mayank Vanjani

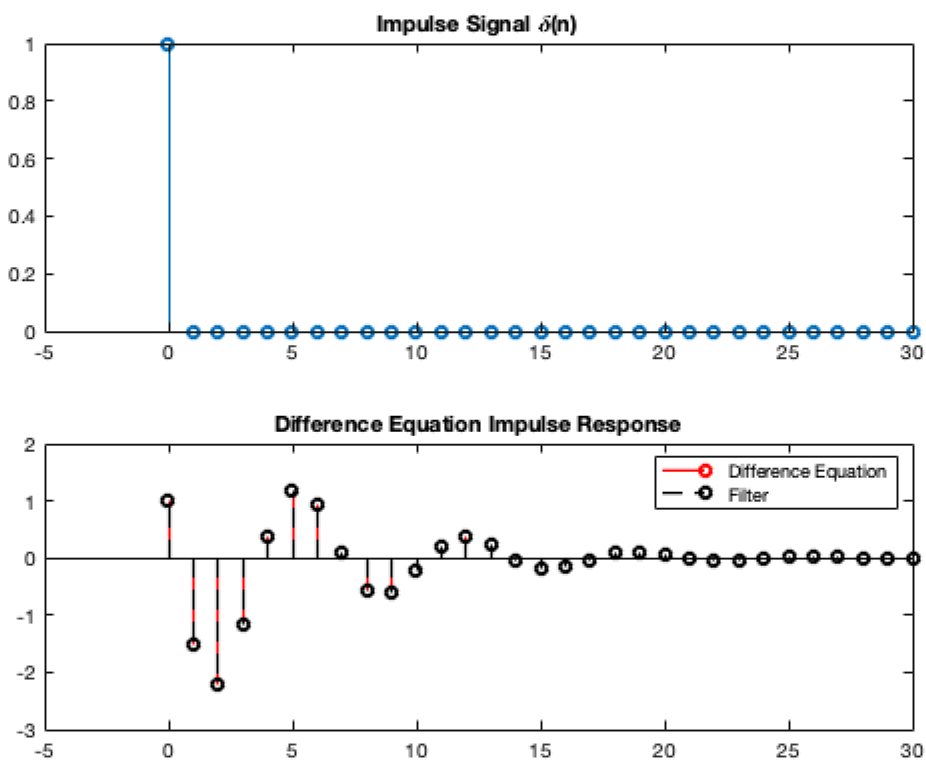
Lab 4: Complex Poles 11/16/18

```
clear;
clc;

step = @(n, t) n >= t;
delta = @(n,t) n == t;
ramp = @(n,t) (n-t).*(n >= t);

% myDiffeq function definition at the end of the script
```

# 1: Difference Equation

```
H(z) = Y/X
Result takes the form of a damped sinusoid

%{
        Derivation of Impulse Response

H(z) = (1 - 2.5z^(-1)) / (1 - z^(-1) + 0.7z^(-2))
X(n - m) <=> z^(-m)X(z)
y(s) = x(s) = 0 for s < 0

y(0) = x(0) - 2.5x(-1) + y(-1) -0.7y(-2) = x(0)
y(1) = x(1) - 2.5x(0) + y(0) -0.7y(-1) = x(1) - 2.5x(0) + y(0)
y(2) = x(2) - 2.5x(1) + y(1) -0.7y(0)
--> y(n) = x(n) - 2.5(n-1) + y(n-1) - 0.7y(n-2) for n >= 2
Above steps in myDiffeq function (underneath)

%}

n = 0:30;
impulse = delta(n,0);

h = myDiffeq(impulse,4);
len_h = length(h);

numerator = [1 -2.5];
denominator = [1 -1 0.7];
```

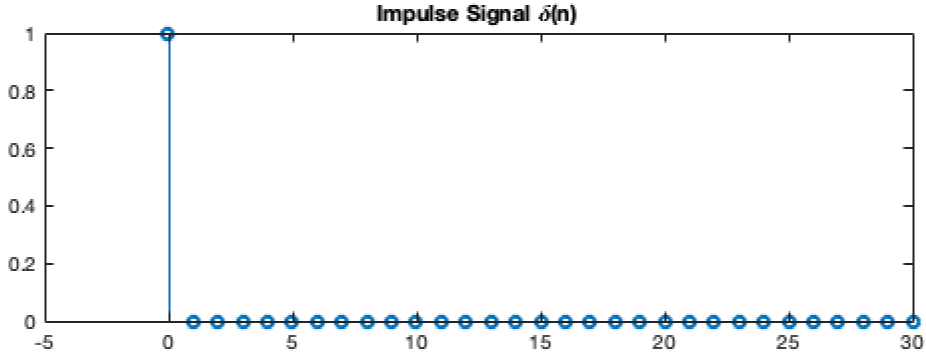
```
h_filter = filter(numerator, denominator, impulse);

figure(1); clf;
subplot(2,1,1);
stem(0:len_h-1,impulse);
xlim([-5 30]);
title("Impulse Signal \delta(n)");

subplot(2,1,2);
% See the same response when using the filter command and difference
 equation
stem(0:len_h-1,h,'r',"Linewidth", 1.25);
hold on;
stem(0:len_h-1,h_filter,'k--',"Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter");
title("Difference Equation Impulse Response");
% Difference equation and calculated impulse responses are the same!
```

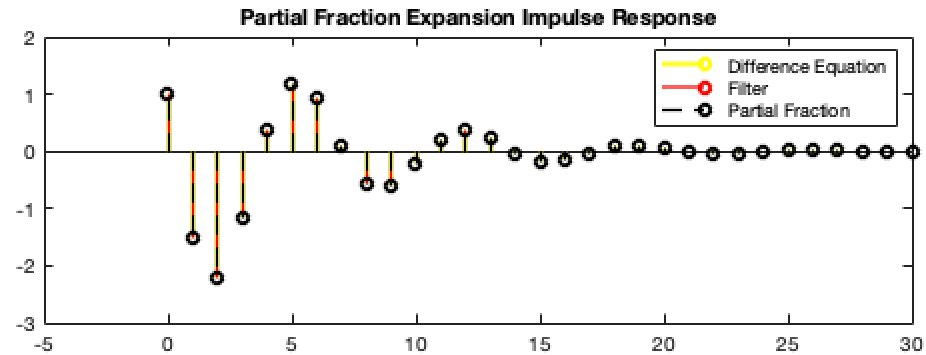



# 2: Partial Franction Expansion

```
Using MATLAB Residue Command
h(n) = C1p1^nu(n) + C2p2^nu(n)

numerator = [1 -2.5];
```

```
denominator = [1 -1 0.7];
[r,p,k] = residue( numerator, denominator );

% Complex Conjugate pairs that cancel into a real impulse response
% Seen from real plot of reconstructed h(n)
C1 = r(1); C2 = r(2);
p1 = p(1); p2 = p(2);
step_funct = step(n,0);

h_partial = (C1 * p1.^n .* step_funct) + (C2 * p2.^n .* step_funct);

figure(2); clf;
subplot(2,1,1);
stem(0:len_h-1,impulse);
xlim([-5 30]);
title("Impulse Signal \delta(n)");

subplot(2,1,2);
stem(0:len_h-1,h,'y', "Linewidth", 2);
hold on;
stem(0:len_h-1,h_filter,'r',"Linewidth", 1.25);
stem(0:len_h-1,h_partial,'k--',"Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter", "Partial Fraction");
title("Partial Fraction Expansion Impulse Response");
% Impulse response using Partial Fraction Expansion is the same as
 before!
```

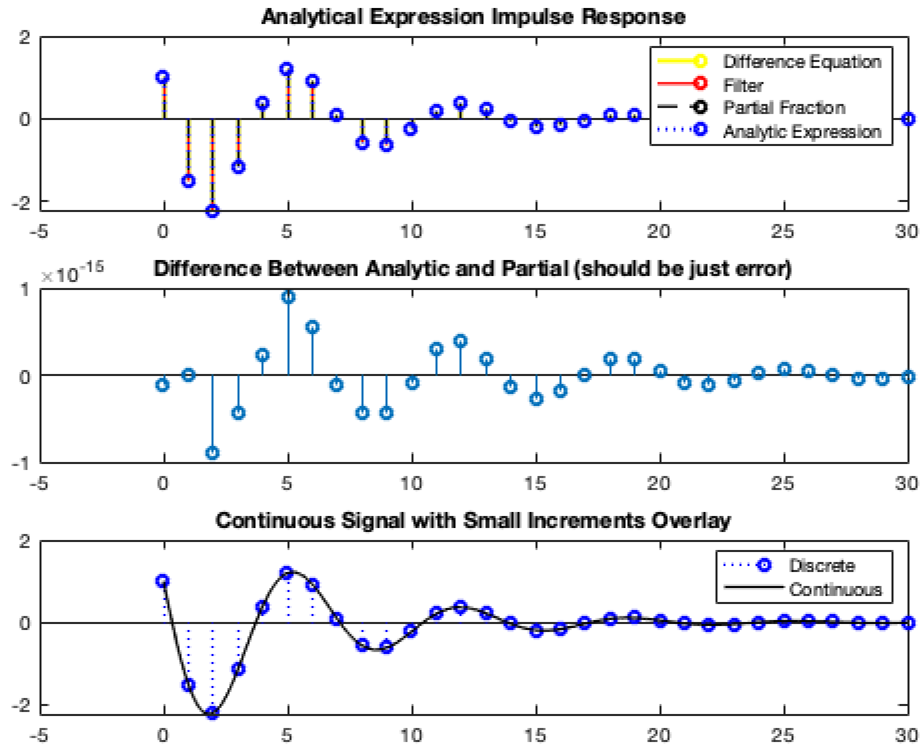



# 3: Expression

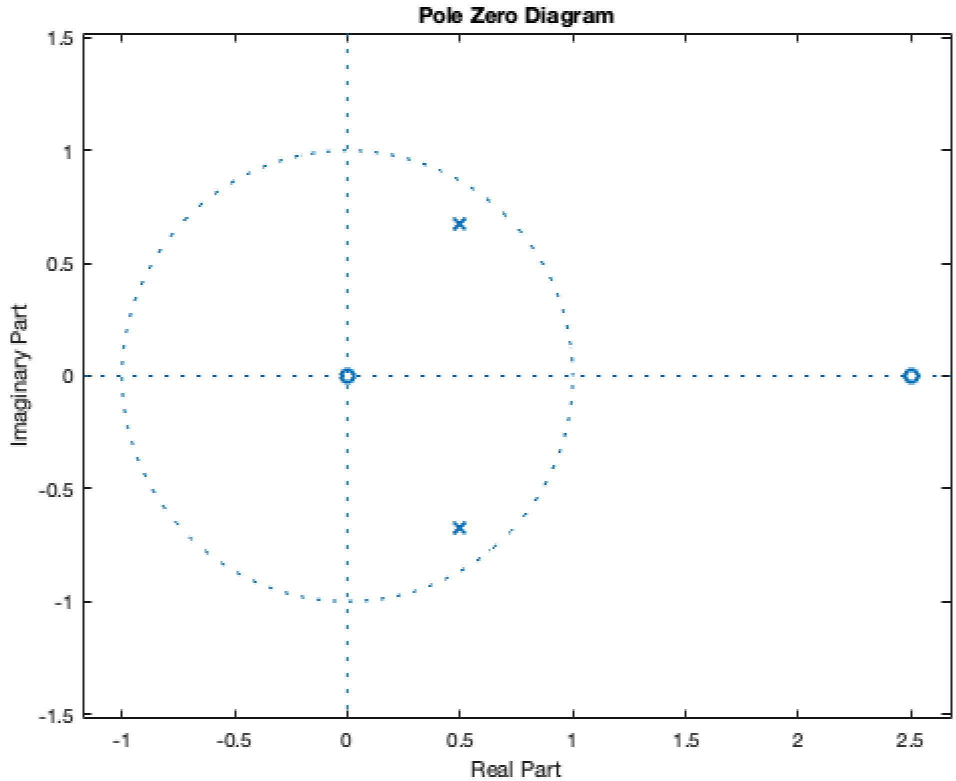
```
Non-Complex Impulse Response
h(n) = Ar^ncos(wn+theta)u(n)
C = Re^(ja)    p = re^(jB)

% Complex to Polar conversion
R1 = abs(C1); R2 = abs(C2);
a1 = angle(C1); a2 = angle(C2);
r1 = abs(p1); r2 = abs(p2);
B1 = angle(p1); B2 = angle(p2);

%{
if (R2 == R1)
    disp("R2 == R1 True");
end
if (a2 == -1*a1)
    disp("a2 == -a1 True");
end
if (r2 == r1)
    disp("r2 == r1 True");
end
if (B2 == -1*B1)
    disp("B2 == -B1 True");
end
%}
```

```
% Analytic Expression Variables and Calculation
% h(n) = 2R1 * r1^n * cos(B1n+a1) * u(n)
A = 2 * R1;
w0 = B1;
theta0 = a1;
h_analytic = A * r1.^n .* cos(w0*n + theta0) .* step(n,0);

figure(3); clf;
% Enable Fullscreen Figure (small graphs otherwise)
% set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1,
 0.96]);
subplot(3,1,1);
stem(0:len_h-1,h,'y',"Linewidth", 2);
hold on;
stem(0:len_h-1,h_filter,'r',"Linewidth", 1.25);
stem(0:len_h-1,h_partial,'k--',"Linewidth", 1.25);
stem(0:len_h-1,h_analytic,'b:',"Linewidth", 1.75);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter", "Partial Fraction", "Analytic
 Expression");
title("Analytical Expression Impulse Response");

subplot(3,1,2);
stem(0:len_h-1,(h_analytic - h_partial));
%ylim([-2 2]);
xlim([-5 30]);
title("Difference Between Analytic and Partial (should be just
 error)");

subplot(3,1,3);
t = 0:0.01:30;
h_analytic_cont = A * r1.^t .* cos(w0*t + theta0) .* step(t,0);
stem(0:len_h-1,h_analytic,'b:', "Linewidth", 1.25);
hold on;
plot(t, h_analytic_cont, 'k', "Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Discrete", "Continuous");
title("Continuous Signal with Small Increments Overlay");
% Once again same impulse response
```




# 4: Z-Plane

```
Z-Plane/Transform shows stability
Poles inside unit circle = Stable

numerator = [1 -2.5];
denominator = [1 -1 0.7];

figure(4); clf;
zplane( numerator, denominator );
title("Pole Zero Diagram");
```




# 5: New Equation

```
Complex Conjugates and Stable = Decaying Sinusoid
Copied steps from previous equation except an extra component/term

% 1
step = @(n, t) n >= t;
delta = @(n,t) n == t;
n = 1:30;
impulse = delta(n,1);

h = myDiffeq(impulse,5);
len_h = length(h);

numerator = [1 -0.6];
denominator = [1 -2.1 1.6 -0.4];
h_filter = filter(numerator, denominator, impulse);

figure(5); clf;
subplot(2,1,1);
stem(0:len_h-1,impulse);
xlim([-5 30]);
title("Impulse Signal \delta(n)");

subplot(2,1,2);
stem(0:len_h-1,h,'r',"Linewidth", 1.25);
```

```
hold on;
stem(0:len_h-1,h_filter,'k--',"Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter");
title("Difference Equation Impulse Response NEW");

% 2
[r,p,k] = residue( numerator, denominator );

C1 = r(1); C2 = r(2); C3 = r(3);
p1 = p(1); p2 = p(2); p3 = p(3);
step_funct = step(n,0);

h_partial = (C1 * p1.^n .* step_funct) + ...
    (C2 * p2.^n .* step_funct) + ...
    (C3 * p3.^n .* step_funct);

figure(6); clf;
subplot(2,1,1);
stem(0:len_h-1,impulse);
xlim([-5 30]);
title("Impulse Signal \delta(n)");

subplot(2,1,2);
stem(0:len_h-1,h,'y', "Linewidth", 2);
hold on;
stem(0:len_h-1,h_filter,'r',"Linewidth", 1.25);
stem(0:len_h-1,h_partial,'k--',"Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter", "Partial Fraction");
title("Partial Fraction Expansion Impulse Response NEW");

% 3
R1 = abs(C1); R2 = abs(C2); R3 = abs(C3);
a1 = angle(C1); a2 = angle(C2); a3 = angle(C3);
r1 = abs(p1); r2 = abs(p2); r3 = abs(p3);
B1 = angle(p1); B2 = angle(p2); B3 = angle(p3);

% h(n) = 3R1 * r1^n * cos(B1n+a1) * u(n)
A = R1 * 2;
w0 = B1;
theta0 = a1;
h_analytic = (A * r1.^n .* cos(w0*n + theta0) .* step(n,0)) + (R3 *
 r3.^n .* step(n,0));
% (R3 * r3.^n .* step(n,0)) Extra component

figure(7); clf;
% Enable Fullscreen Figure (small graphs otherwise)
% set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1,
 0.96]);
subplot(3,1,1);
stem(0:len_h-1,h,'y',"Linewidth", 2);
```

```
hold on;
stem(0:len_h-1,h_filter,'r',"Linewidth", 1.25);
stem(0:len_h-1,h_partial,'k--',"Linewidth", 1.25);
stem(0:len_h-1,h_analytic,'b:',"Linewidth", 1.75);
hold off;
xlim([-5 30]);
legend("Difference Equation", "Filter", "Partial Fraction", "Analytic
 Expression");
title("Analytical Expression Impulse Response NEW");

subplot(3,1,2);
stem(0:len_h-1,(h_analytic - h_partial));
xlim([-5 30]);
title("Difference Between Analytic and Partial (should be just
 error)");

subplot(3,1,3);
t = 0:0.01:29;
h_analytic_cont = A * r1.^t .* cos(w0*t + theta0) .* step(t,0);
stem(1:len_h,h_analytic,'b:', "Linewidth", 1.25);
hold on;
plot(t, h_analytic_cont, 'k', "Linewidth", 1.25);
hold off;
xlim([-5 30]);
legend("Discrete", "Continuous");
title("Continuous Signal with Small Increments Overlay NEW");

% 4
numerator = [1 -0.6];
denominator = [1 -2.1 1.6 -0.4];
figure(8); clf;
zplane( numerator, denominator );
title("Pole Zero Diagram NEW");
```

# myDiffeq Difference Equation Function

```
Used in part 1 of both transfer functions

function y = myDiffeq(x, case_val)

N = length(x);
y = zeros(1,N);

switch case_val

    case 1
        y(1) = x(1);
        for n = 2:N-1
            y(n) = x(n)+2*x(n-1)-0.95*y(n-1);
        end

    case 2
        y(1) = x(1);
```

```
        for n = 2:N-1
            y(n) = x(n)+2*x(n-1);
        end

    case 3
        y(1) = x(1);
        for n = 2:N-1
            y(n) = x(n)+2*x(n-1)-1.1*y(n-1);
        end

    case 4
        y(1) = x(1);
        y(2) = x(2) - 2.5*x(1) + y(1);
        for n = 3:N-1
            y(n) = x(n) - 2.5*x(n-1) + y(n-1) - 0.7*y(n-2);
        end

    case 5
        y(1) = x(1);
        y(2) = x(2) - 0.6*x(1) + 2.1*y(1);
        y(3) = x(3) - 0.6*x(2) + 2.1*y(2) - 1.6*y(1);
        for n = 4:N-1
            y(n) = x(n) - 0.6*x(n-1) + 2.1*y(n-1) - 1.6*y(n-2) +
 0.4*y(n-3);
        end
    end

end
```

*Published with MATLAB® R2018a*