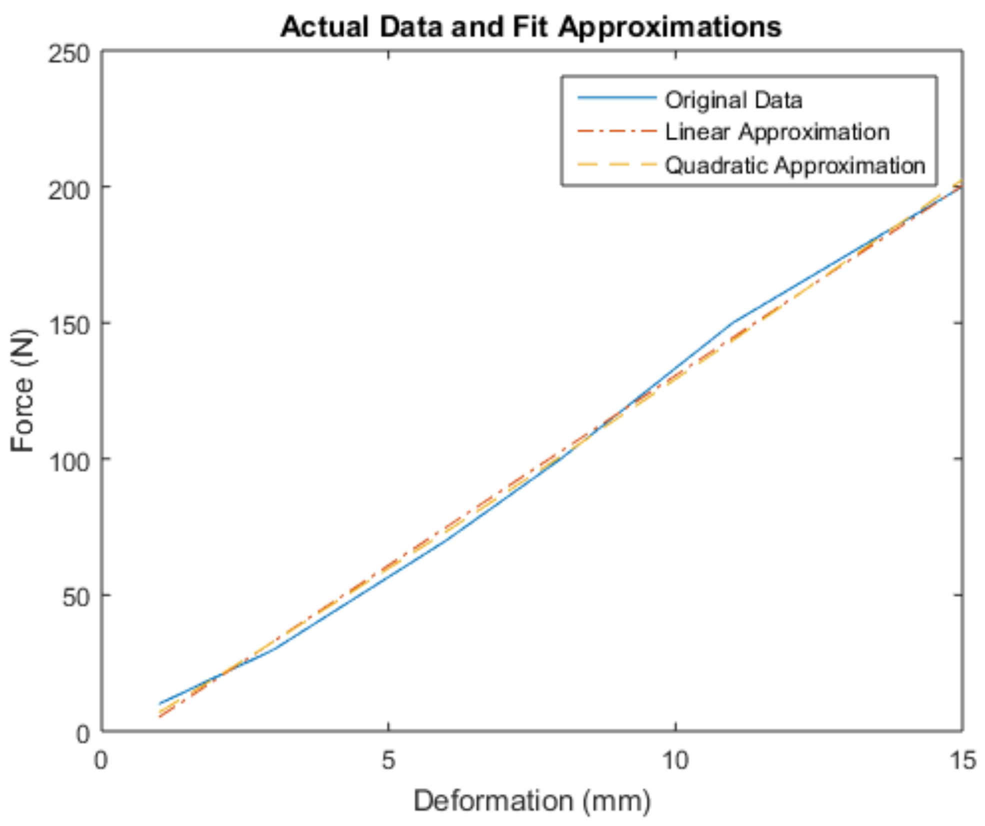
Contents

Problem 2 b

```
Force = [ 10, 20, 30, 70, 100, 150, 200 ]; % Force in newtons
Deformation = [ 1, 2, 3, 6, 8, 11, 15 ]; % Deformation in mm

Alin = [1 1; 1 2; 1 3; 1 6; 1 8; 1 11; 1 15]; % Deformation as column
% Deformation as Column vector with squares for quad
Aquad = [1 1 1; 1 2 4; 1 3 9; 1 6 36; 1 8 64; 1 11 121; 1 15 225];
% b vector from forces
b = [10; 20; 30; 70; 100; 150; 200];

% leas squares linear
xoptlin = Alin \ b;
% least squares quadratic
xoptquad = Aquad \ b;

% returns values flipped in wrong direction so flip them
xoptquad = flipud(xoptquad);
xoptlin = flipud(xoptlin);

% graph the fit lines and original data
ylin = polyval(xoptlin, Deformation);
yquad = polyval(xoptquad, Deformation);
figure(1)
plot(Deformation, Force)
hold on
plot(Deformation, ylin, '-.')
plot(Deformation, yquad, '--')
hold off
title('Actual Data and Fit Approximations')
xlabel('Deformation (mm)')
ylabel('Force (N)')
legend('Original Data', 'Linear Approximation', 'Quadratic Approximation')
linmax = max(abs(ylin - Force));
quadmax = max(abs(yquad - Force));
fprintf('Maximum Linear Error: %d\r\n', linmax);
fprintf('Maximum Quadratic Error: %d\r\n', quadmax);
```

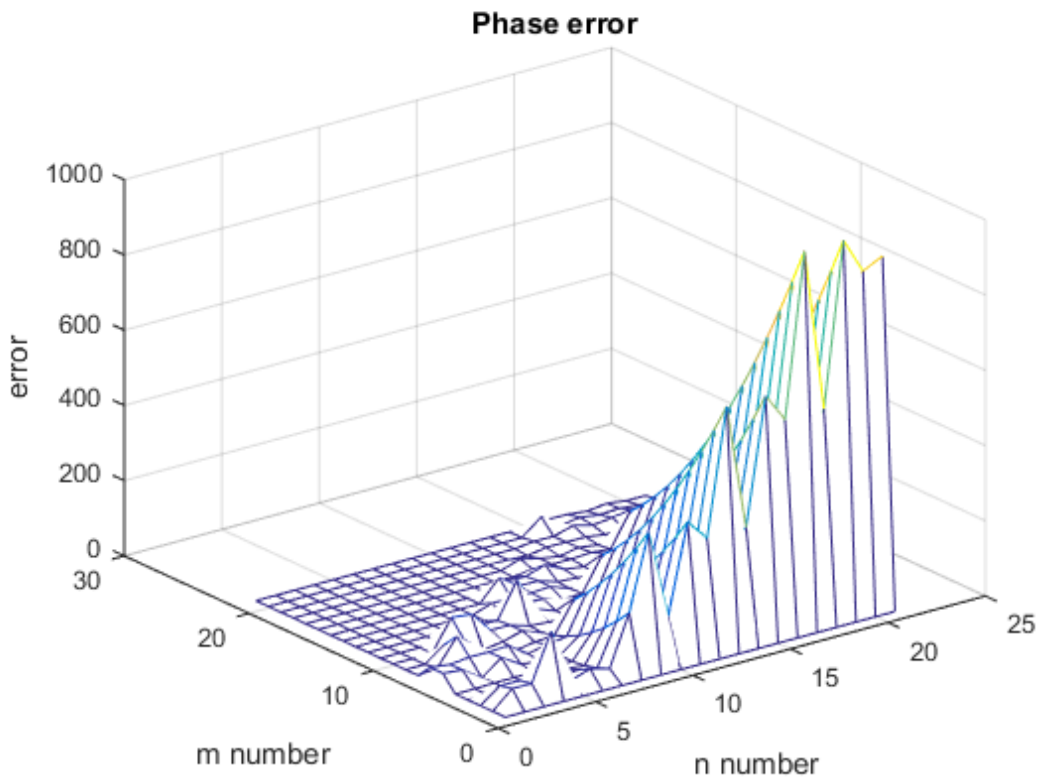
```
Maximum Linear Error: 5.407609e+00
Maximum Quadratic Error: 6.468645e+00
```

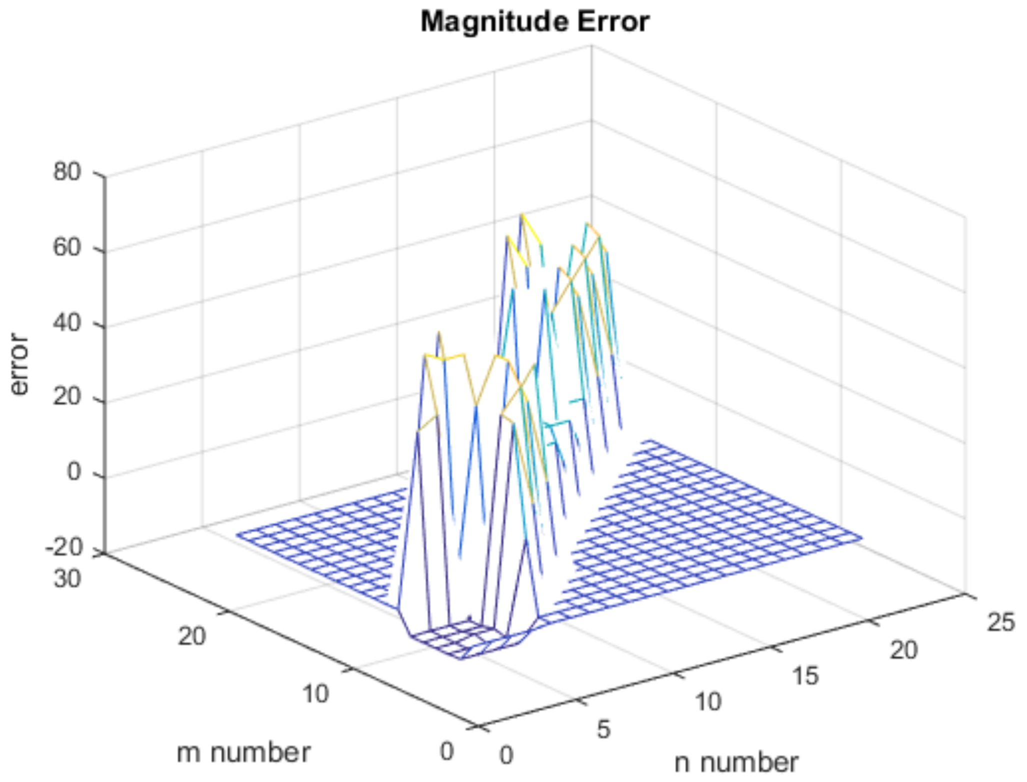


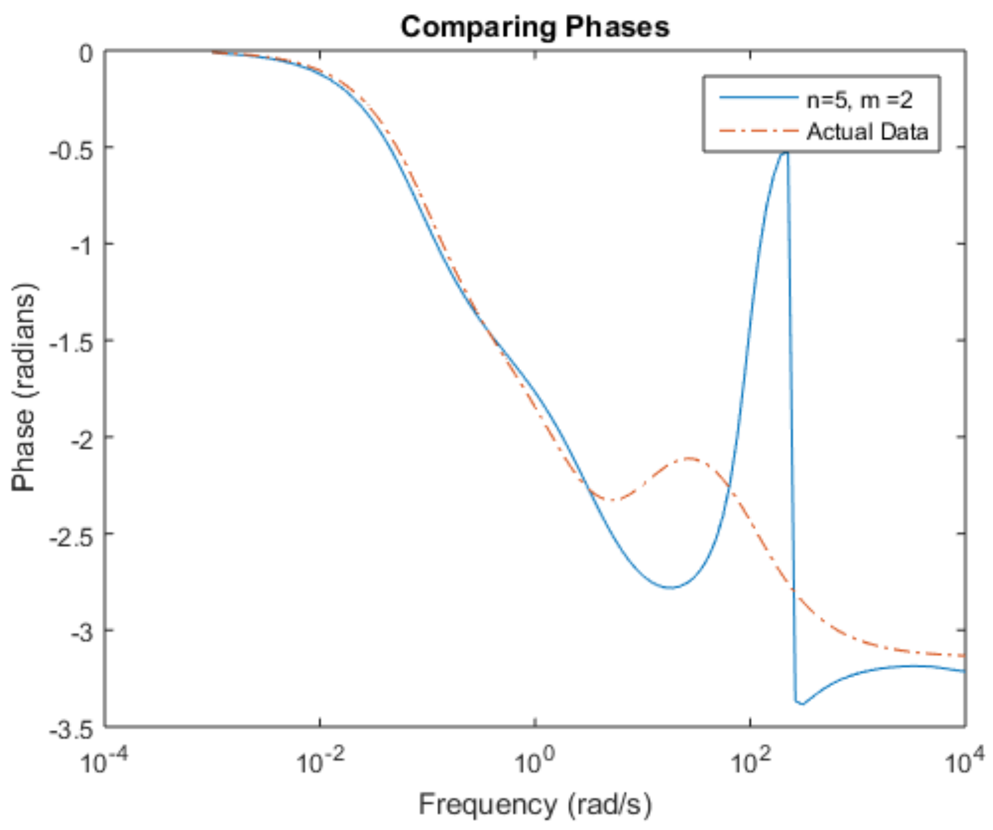
Problem 4

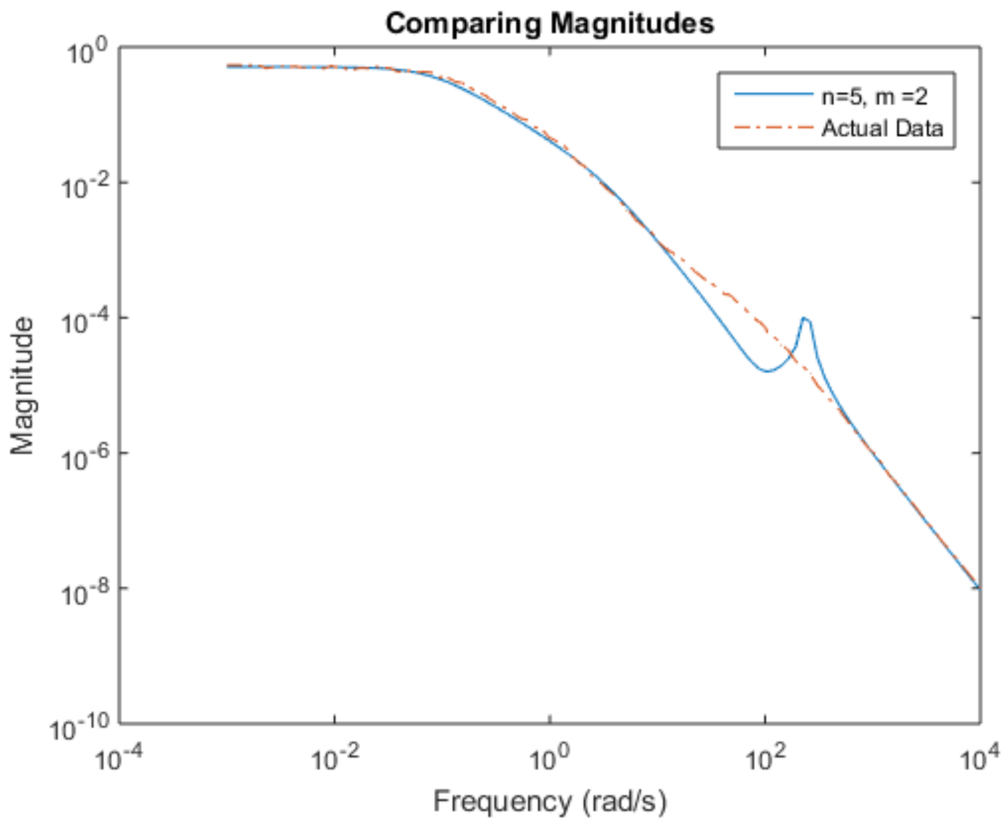
```
load('data_for_freqresp_Prob4_HW2')
% get real and imaginary parts in rectangular form
real = MAG_M.*cos(PHASE_RAD_M);
imag = MAG_M.*sin(PHASE_RAD_M);
% set range of n and m
n = 20;
m = 20;
% step through n's and m's and average error
for nn = 1:n
    for mm = 0:m
        % get optimal x values using least squares from myLS function
        xopts = myLS(nn, mm, W, real, imag);
        % get the numerator from the last m values of xopts
        num = fliplr(xopts(nn+1:nn+1+mm)');
        % get the denominator from the first n
        den = cat(2, 1, fliplr(xopts(1:nn)'));
        % create the transfer function
        sys = tf(num, den);
        % get the phase and magnitude from the bode plot
        [magt, phaset, wt] = bode(sys, W);
        % record the average error of the magnitude and phase
        errorM(nn + 1, mm + 1) = mean((MAG_M - magt(:)).^2);
        errorP(nn + 1, mm + 1) = mean((PHASE_RAD_M - phaset(:) * pi / 180).^2);
    end
end

figure(2)
mesh(errorP)
```

```
title('Phase error')
xlabel('n number')
ylabel('m number')
zlabel('error')
figure(3)
mesh(log(errorM))
title('Magnitude Error')
xlabel('n number')
ylabel('m number')
zlabel('error')

% do it again for favorite
nn = 5;
mm = 2;
xopts = myLS(nn, mm, W, real, imag);
num = fliplr(xopts(nn+1:nn+1+mm)');
den = cat(2, 1, fliplr(xopts(1:nn)'));
sys = tf(num, den);
[magt, phaset, wt] = bode(sys, W);

figure(4)
semilogx(wt, phaset(:)*pi/180)
hold on
semilogx(W, PHASE_RAD_M, '-.')
hold off
title('Comparing Phases')
xlabel('Frequency (rad/s)')
ylabel('Phase (radians)')
legend('n=5, m =2', 'Actual Data')
figure(5)
loglog(wt, magt(:))
hold on
loglog(W, MAG_M, '-.')
title('Comparing Magnitudes')
xlabel('Frequency (rad/s)')
ylabel('Magnitude')
legend('n=5, m =2', 'Actual Data')
hold off
```


Phase error
error
1000
800
600
400
200
0
30
20
10
0
m number
0
5
10
15
20
25
n number


Magnitude Error
error
80
60
40
20
0
-20
30
20
10
0
m number
0
5
10
15
20
25
n number

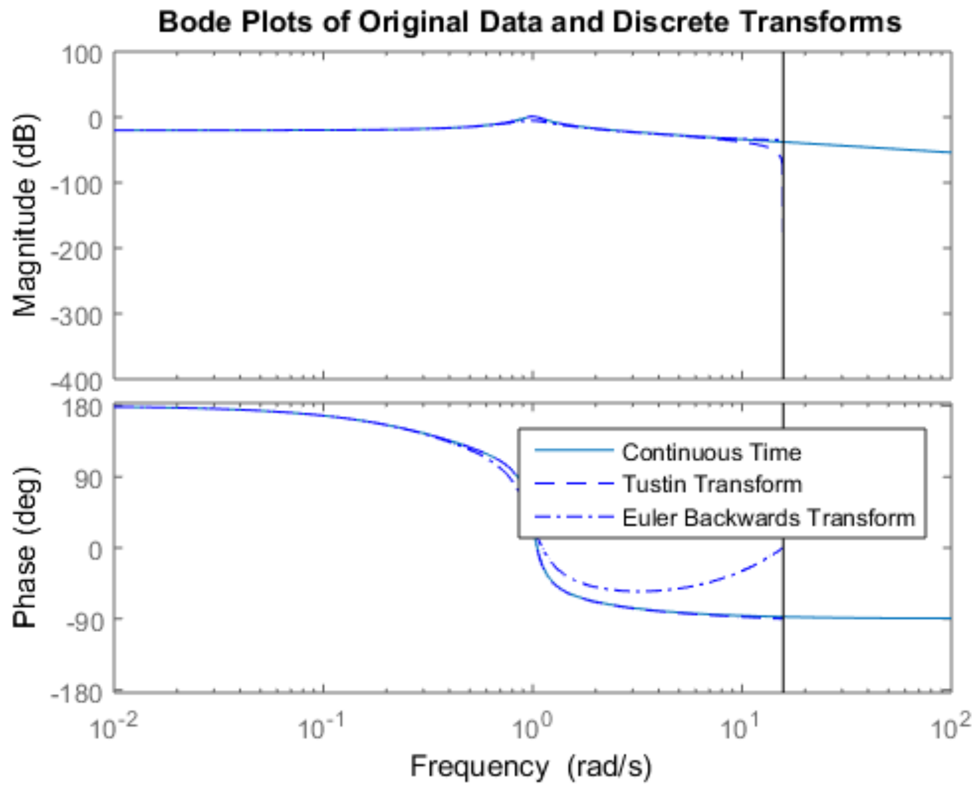




Problem 5d

```
% create the 3 transfer functions
% Continuous Time
```

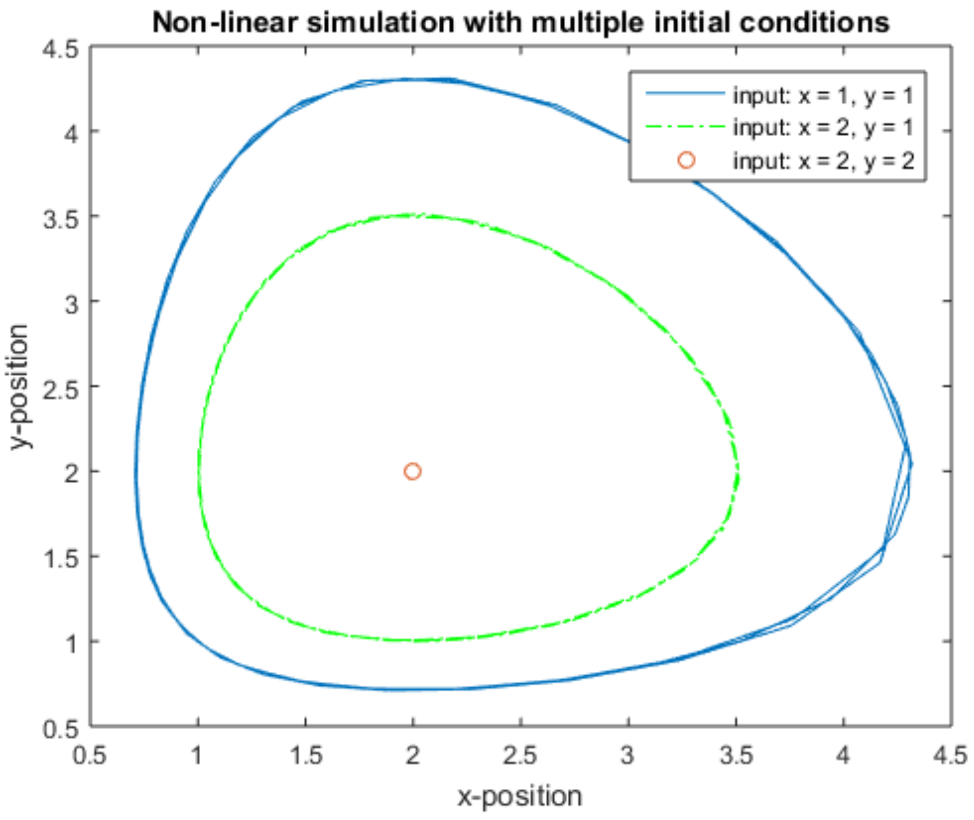
```
sysC = tf([0.2, -0.1], [1, 0.2, 1]);
% Tustin transform
sysT = filt([19, -2, -21], [1030, -1980, 990], 0.2);
% Reverse Euler
sysBE = filt([0.9, -1], [27, -51, 25], 0.2);

% plot the systems together
figure(6)
bode(sysC)
hold on
bode(sysT, '--')
bode(sysBE, '-.')
title('Bode Plots of Original Data and Discrete Transforms')
legend('Continuous Time', 'Tustin Transform', 'Euler Backwards Transform')

hold off
```



Problem 6d

```
% set final time
t = 10;
% set first initial conditions
yinput = [1; 1];
% solve with ode45 and prob6d function
[out_t, out_z] = ode45(@prob6d, [0, 10], yinput);

figure(7)
plot(out_z(:,1), out_z(:, 2))
hold on
```

```
% second initial conditions and solve
yinput = [2; 1];
[out_t, out_z] = ode45(@prob6d, [0, 10], yinput);

plot(out_z(:,1), out_z(:, 2), 'g-.')
% third intitial conditions and solve
yinput = [2; 2];
[out_t, out_z] = ode45(@prob6d, [0, 10], yinput);

scatter(out_z(:,1), out_z(:, 2))
legend('input: x = 1, y = 1', 'input: x = 2, y = 1', 'input: x = 2, y = 2')
title('Non-linear simulation with multiple initial conditions')
xlabel('x-position')
ylabel('y-position')
hold off
```



Published with MATLAB® R2015a