```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kaleb Coleman, Matthew Day, Scott Meyers, Andrew Peterson
% October 19, 2021
% ME4133
% Project III
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Preliminaries
clear; clc; close all;

%Inital guess for Position Analysis
%Theta 3 is equal to Theta 5
R1=3.52;
t1=(90*(pi/180));
R2=.82;
R3=4;
t3=(-110*(pi/180));
R4=1;
t4=0;
R5=2;
t5=(-101*(pi/180));
R6=1.85;
t6=(63*(pi/180));

%Import .txt data file as a single matrix
File_Data = readmatrix('data1');

%Define matrix values to variables to sync two theta and calculate %
 error
%for data validation
theta2_Pre = File_Data(:,1');
theta3 = File_Data(:,2) ;
Provided_R3 =File_Data(:,3);
Provided_R4 =File_Data(:,4);
Provided_R5 =File_Data(:,5);

%Data Imported from excel for First Order %error calculation
Provided_H3 =File_Data(:,6);
Provided_F3 =File_Data(:,7);
Provided_F4 =File_Data(:,8);
Provided_F5 =File_Data(:,9);

%Data Imported from excel for Second Order %error calculation
%H3P =File_Data(:,10)';
%F3P =File_Data(:,11)';
%F4P =File_Data(:,12)';
%F5P =File_Data(:,13)';

%Position Analysis
 -------------------------------------------------------

%Container for position solutions
```

```
M = [];

%Iterators
I = 0;
r = 0;

%Theta 2 setup
theta2 = theta2_Pre';

%Theta 2 iteration from 0 to 2pi
for t2 = theta2;
    r = r+1;

    %Newton Raphson method
    while (.00005 < R1*exp(1i*(pi/2))+R4*exp(1i*0)+R3*exp(1i*t3)-
R2*exp(1i*t2)) | (.00005 < R2*exp(1i*t2)-R5*exp(1i*t3)-R6*exp(1i*t6))
        %^^^^ check if current estimate is close enough using VLE's
        I = I+1;

        % A position matrix
        A = [-R3*sin(t3) cos(t3) 1 0;
          R3*cos(t3) sin(t3) 0 0;
            R5*sin(t3) 0 0 -cos(t3);
            -R5*cos(t3) 0 0 -sin(t3)];
        %b position matrix
        b= -[0+R4+R3*cos(t3)-R2*cos(t2);
            R1+0+R3*sin(t3)-R2*sin(t2);
            R2*cos(t2)-R5*cos(t3)-R6*cos(t6);
            R2*sin(t2)-R5*sin(t3)-R6*sin(t6)];
        j= A\b; %Jacobian

        %add deltas to last estimate to get better estimate
        t3 = t3 + j(1);
        R3 = R3 + j(2);
        R4 = R4 + j(3);
        R5 = R5 + j(4);
    end
    %Results stored in matrix
    M(r,1:5) = [t2,t3,R3,R4,R5];
end
M;
% First Order
 Coefficients-----------------------------------------------
% b1= -[R2*sin(t2);
%      -R2*cos(t2);
%      -R2*sin(t2);
%      R2*cos(t2)];
%iterate through each data set
for iter = 1:size(M,1)
    % A position matrix
    A = [-M(iter,3)*sin(M(iter,2)) cos(M(iter,2)) 1 0;
        M(iter,3)*cos(M(iter,2)) sin(M(iter,2)) 0 0;
        M(iter,5)*sin(M(iter,2)) 0 0 -cos(M(iter,2));
        -M(iter,5)*cos(M(iter,2)) 0 0 -sin(M(iter,2))];
```

```
    %b first order matrix (partial diff. wrt theta 2)
    b1= -[R2*sin(M(iter,1));
        -R2*cos(M(iter,1));
        -R2*sin(M(iter,1));
        R2*cos(M(iter,1))];

    M(iter,6:9) = A\b1;
end
%---------------------------------------------------------------------------

%Data Validation by checking our calculations against the provided
 Excel
%Sheet data, in the form of percent error
Delta_t2 = (abs(M(:,1) - theta2_Pre) / theta2_Pre) * 100;
Delta_t3 = (abs(M(:,2) - Provided_R3) / Provided_R3) * 100;
Delta_R3 = (abs(M(:,3) - Provided_R3) / Provided_R3) * 100;
Delta_R4 = (abs(M(:,4) - Provided_R4) / Provided_R4) * 100;
Delta_R5 = (abs(M(:,5) - Provided_R5) / Provided_R5) * 100;

%First Coefficients validation
Delta_H3 = (abs(M(:,2) - Provided_H3) / Provided_H3) * 100;
Delta_F3 = (abs(M(:,3) - Provided_F3) / Provided_F3) * 100;
Delta_F4 = (abs(M(:,4) - Provided_F4) / Provided_F4) * 100;
Delta_F5 = (abs(M(:,5) - Provided_F5) / Provided_F5) * 100;

%Second Coefficients validation
%Delta_H3 = (abs(M(:,2) - Provided_H3) / Provided_H3) * 100;
%Delta_F3 = (abs(M(:,3) - Provided_F3) / Provided_F3) * 100;
%Delta_F4 = (abs(M(:,4) - Provided_F4) / Provided_F4) * 100;
%Delta_F5 = (abs(M(:,5) - Provided_F5) / Provided_F5) * 100;

 %Position anaylsis graph-------------------------------------------

%Define Y-points of graph (t3,r3,r4,r5,h3,f3,f4,f5)
t3=M(:,2)';
R3=M(:,3)';
R4=M(:,4)';
R5=M(:,5)';
H3=M(:,6)';
F3=M(:,7)';
F4=M(:,8)';
F5=M(:,9)';
%Local minimums and maximums for theta3, R3, R4, and R5 which are also
 the
%link limits for the unknowns

t3_min = islocalmin(t3);
t3_max = islocalmax(t3);

R3_min = islocalmin(R3);
R3_max = islocalmax(R3);

R4_min = islocalmin(R4);
R4_max = islocalmax(R4);
```

```
R5_min = islocalmin(R5);
R5_max = islocalmax(R5);

%Local minimums and maximums for H3, F3, F4, and F5

H3_min = islocalmin(H3);
H3_max = islocalmax(H3);

F3_min = islocalmin(F3);
F3_max = islocalmax(F3);

F4_min = islocalmin(F4);
F4_max = islocalmax(F4);

F5_min = islocalmin(F5);
F5_max = islocalmax(F5);

%Allows multiple data lines on one figure
hold on

%Plot the Position Analysis data
plot(theta2, t3, 'k', theta2, R3, 'b', theta2, R4, 'm', theta2,
 R5, 'r')

%Plot the local mins and maxes for data validation (used to compare
 roots on the First Order Graph)
plot(theta2(t3_min), t3(t3_min), 'k*', theta2(t3_max),
 t3(t3_max), 'k*', theta2(R3_min), R3(R3_min), 'b*', theta2(R3_max),
 R3(R3_max), 'b*', theta2(R4_min), R4(R4_min), 'm*', theta2(R4_max),
 R4(R4_max), 'm*',  theta2(R5_min), R5(R5_min), 'r*', theta2(R5_max),
 R5(R5_max), 'r*')

%Add legend to make the different lines distinguishable
legend('theta3 (rad)', 'R3 (in)','R4 (in)','R5 (in)','Local min or max
 (all lines)')

%Define what the graph is
title('Position Analysis')

%Label the x-axis
xlabel('\theta_2 (rad)')

%Label the y-axis
ylabel('Outputs')

%Add grids to make data easier to read
grid on
grid minor

%Used to convert the numerical radian to the simplified version
 original
%Sets the increment of values
set(gca,'XTick',0:pi/4:2*pi) ;
```

```
%Defines what the major x-axis grid line should be called
set(gca,'XTickLabel',
{'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'})



%First order Kinematic Coefficient vs input
 graph-----------------------
figure(2) %To setup different data on different plots
hold on   %Allows multiple data lines on one figure

%Plot the First Order Kinematic Coefficients
plot(theta2, H3,'k', theta2, F3, 'b', theta2, F4, 'm', theta2,
 F5, 'r')

%Plot the roots that correspond to the local mins and maxes on the
 Position Analysis Graph
 %theta2(t3_max), 0, 'k*', theta2(R4_min), 0, 'm*',
plot(theta2(t3_min), 0, 'k*',theta2(R3_min), 0, 'b*', theta2(R3_max),
 0, 'b*', theta2(R4_max), 0, 'm*',  theta2(R5_min), 0, 'r*',
 theta2(R5_max), 0, 'r*')

%Plot the local mins and maxes for data validation of the Second Order
 Graph
%
plot(theta2(H3_min), H3(H3_min), 'ks', theta2(H3_max),
 H3(H3_max), 'ks',theta2(F3_min), F3(F3_min), 'bs', theta2(F3_max),
 F3(F3_max), 'bs', theta2(F4_min), F4(F4_min), 'ms', theta2(F4_max),
 F4(F4_max), 'ms',  theta2(F5_min), F5(F5_min), 'rs', theta2(F5_max),
 F5(F5_max), 'rs')
%Add legend to make the different lines distinguishable
legend('H3 (-)', 'F3 (in)','F4 (in)','F5 (in)','Local min or max (all
 lines)');
%Define what the graph is
title('First Order Kinematic Coefficient vs Input')
%Label the x-axis
xlabel('\theta_2 (rad)')
%Label the y-axis
ylabel('Outputs')
%Add grids to make data easier to read
grid on
grid minor
%Used to convert the numerical radian to the simplified version
 original
%Sets the increment of values
set(gca,'XTick',0:pi/4:2*pi)
%Defines what the major x-axis grid line should be called
set(gca,'XTickLabel',
{'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'})

%---------------------------------------------------------------------
%Displaying link limits
fprintf('Theta 3 Lower Limit %s\n', (min(M(:,2))))
```

```
fprintf('Theta 3 Upper Limit %s\n', (max(M(:,2))))

fprintf('Length 3 Lower Limit %s\n',(min(M(:,3))))
fprintf('Length 3 Upper Limit %s\n',(max(M(:,3))))

fprintf('Length 4 Lower Limit %s\n',(min(M(:,4))))
fprintf('Length 4 Upper Limit %s\n',(max(M(:,4))))

fprintf('Length 5 Lower Limit %s\n',(min(M(:,5))))
fprintf('Length 5 Upper Limit %s\n',(max(M(:,5))))
```

Published with MATLAB® R2020a