Table of Contents

ReadMe

ME 4133 Patrick Herke, Bailey Smoorenburg, Jill Bohnet, Connor McCarthy, Gavin Sheng

```
% Each section can be run individually
% They will call one of the functions listed at the bottom of the file
% kinematicData must be run before anything else
% Then run the section defining the speed and load
% After that the order shouldn't matter
% If issues arise please run the functions in order
```

```
% To change w2 and P4 simply adjust the values in the section Defining
 speed and
% load and re-run the section
```

Solving for Kinematic Postion Data

```
kinematicData();
```

Plotting Kinematic Data

```
plottingKinematicData();
```

Validating KCs by gradient function

```
validationByGrad();
```

Defining speed and load

```
w2 = 10*2*pi/60;
P4 = -20.0;
```

Finding Joint Forces by IDP (no friction)

```
IDP(w2, P4)
```

Finding Torque by Power Equation (no friction)

```
torqueConstant(w2,P4)
```

Finding Startup Torque (no friction)

```
[torDataMax,maxTorStartup] = torqueStartup(w2,P4)
```

Finding Speeds for Dynamic vs Static Forces Dominating

```
dynamicVsStatic()
```

Finding Joint Forces by IDP (with friction)

```
inc12 = false;
inc34 = false;
inc13 = false;
inc23 = true;
```

```
inc14 = false;
frictionIDP(inc12,inc34,inc13,inc23,inc14,w2,P4)
```

Functions

Function: kinematicData

```
function posData = kinematicData()
    clear; clc;

    % The values from A139 to A216 of excel sheet are not possible due
 to length of theta3

    % defining known lengths and angles
    R1 = 3.52;
    th1 = 90*pi/180;
    R2 = 0.82;
    th4 = 0*pi/180;
    R6 = 1.85;
    th6 = 63*pi/180;
    RCG3 = 2.28;

    % Remaining unkown lengths and angles are:
    th2 = 0*pi/180;
    th3 = 269.3*pi/180;
    R3 = 4.07;
    R4 = 0.87;
    R5 = 1.65;
    % th5 = th3;

    % defining and intitializing the table
    sz = [361 17];
    varNames =
 ["theta2","theta3","R3","R4","R5", "h3","f3","f4","f5", "h3p","f3p","f4p","f5p",
    varTypes = repmat("double",1,17);
    posData =
 table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);

    % generating the data for the position plots
    % the for loop uses the previous position data as a guess for the
 next part
    for j = 1:361;
        % calculating theta2 in radians and adding it to the table
        th2 = (j-1)*pi/180;
        posData.theta2(j) = th2;

        % setting the tolerance
        tol = 1e-8;
        % setting the correction factor to zero to start
        xNew = [0; 0; 0; 0;];

        % implementing newton rasphon up to 100 times
```

```
    for i = 1:100
        xOld = xNew;

        % defining the jacobian
        J = [-R3*sin(th3), cos(th3), 1, 0;
          R3*sin(th3), sin(th3), 0, 0;
          R5*sin(th3), 0, 0, -cos(th3);
         -R5*cos(th3), 0, 0, -sin(th3);];

        % defining the residual
        b = [-(0 + R4 + R3*cos(th3) - R2*cos(th2));
         -(R1 + 0 + R3*sin(th3) - R2*sin(th2));
         -(R2*cos(th2) - R5*cos(th3) - R6*cos(th6));
         -(R2*sin(th2) - R5*sin(th3) - R6*sin(th6));];

        % calculating the correction factor
        xNew = J\b;

        % updating the position variables
        th3 = th3 + xNew(1);
        R3 = R3 + xNew(2);
        R4 = R4 + xNew(3);
        R5 = R5 + xNew(4);

        % calculating the error
        relErr = norm(xNew - xOld) / norm(xOld);

        % ending the loop if the error is below the tolerance
        if relErr < tol
            break
        end
    end

    % adding the position data to the table
    posData.theta3(j) = th3;
    posData.R3(j) = R3;
    posData.R4(j) = R4;
    posData.R5(j) = R5;
end

%%First Order Kinematic Coefficients
for i=1:361
    %%creating variables for the data
    theta2 = posData.theta2(i);
    theta3 = posData.theta3(i);
    R3 = posData.R3(i);
    R4 = posData.R4(i);
    R5 = posData.R5(i);


    %%writing in the Jacobian
    J=[-R3*sin(theta3),cos(theta3),1,0;
       R3*cos(theta3),sin(theta3),0,0;
       R5*sin(theta3),0,0,-cos(theta3);
```

```
            -R5*cos(theta3),0,0,-sin(theta3)];

        %%writing in "b"
        b=[-R2*sin(theta2);
            R2*cos(theta2);
            R2*sin(theta2);
            -R2*cos(theta2)];

        %%calculating the first order KC's
        x=J\b;

        posData.h3(i) = x(1,1);
        posData.f3(i) = x(2,1);
        posData.f4(i) = x(3,1);
        posData.f5(i) = x(4,1);
    end

    %%First Order Kinematic Coefficients
    for i=1:361
        %%creating variables for the data
        %%position
        theta2 = posData.theta2(i);
        theta3 = posData.theta3(i);
        R3 = posData.R3(i);
        R4 = posData.R4(i);
        R5 = posData.R5(i);

        %%first order kinemtaics
        h3 = posData.h3(i);
        f3 = posData.f3(i);
        f4 = posData.f4(i);
        f5 = posData.f5(i);

        %%writing in the Jacobian, remains the same from previous
        J = [-R3*sin(theta3),cos(theta3),1,0;
           R3*cos(theta3),sin(theta3),0,0;
           R5*sin(theta3),0,0,-cos(theta3);
           -R5*cos(theta3),0,0,-sin(theta3)];

        %%writing in "b"
        b = [-(R2*cos(theta2) - R3*h3^2*cos(theta3) -
2*f3*h3*sin(theta3));
            -(R2*sin(theta2) - R3*h3^2*sin(theta3) +
2*f3*h3*cos(theta3));
            -(-R2*cos(theta2) + R5*h3^2*cos(theta3) +
2*f5*h3*sin(theta3));
            -(-R2*sin(theta2) + R5*h3^2*sin(theta3) -
2*f5*h3*cos(theta3))];

        %%calculating the first order KC's
        x = J\b;

        posData.h3p(i) = x(1,1);
        posData.f3p(i) = x(2,1);
```

```
            posData.f4p(i) = x(3,1);
            posData.f5p(i) = x(4,1);
        end

        for i=1:361
            %%creating variables for the data
            %%position
            theta2 = posData.theta2(i);
            theta3 = posData.theta3(i);
            RCG3 = 2.28;

            h3 = posData.h3(i);
            f4 = posData.f4(i);

            h3p = posData.h3p(i);
            f4p = posData.f4p(i);

            fg3x = f4 - RCG3*h3*sin(theta3);
            fg3y = RCG3*h3*cos(theta3);

            fg3xp = f4p + RCG3*(-h3p*sin(theta3) - h3^2*cos(theta3));
            fg3yp = RCG3*(h3p*cos(theta3) - h3^2*sin(theta3));

            posData.fg3x(i) = fg3x;
            posData.fg3y(i) = fg3y;
            posData.fg3xp(i) = fg3xp;
            posData.fg3yp(i) = fg3yp;
        end

        % writing the data to an excel file
        filename = 'kinematicData.xlsx';
        writetable(posData,filename,'Sheet',1,'Range','A1')
    end
```

Function: plottingKinematicData

```
    function plottingKinematicData()
```

Program Setup

```
        clear; clc; % clear previous work

        % reading in the data
        fullTable = readtable('kinematicData.xlsx');

        % defining colors for the graphs
        graphColors = {'#BD3131','#CAC006','#3C7FE6','#40A72A'};
        rowMinMax = {'min','max'};
```

Position Graph

```
        % defining columns for the position graph data
```

```
    posColNames = {'theta3','R3','R4','R5'};
    % initializing the table for the minimums and maximums
    posMinMax = table([0;0],[0;0],[0;0],
[0;0],'VariableNames',posColNames,'RowNames',rowMinMax);
    % finding indices of the local minimums and maximums for the
 position graph
    for i=1:4
        posMinMax.(posColNames{i}) = [find(fullTable.(posColNames{i})
 == min(fullTable.(posColNames{i}))); find(fullTable.(posColNames{i})
 == max(fullTable.(posColNames{i})))];
    end

    % defining the figure
    figure('Name','Position','position',[10,10,1200,1000])
    % plotting theta3, R3, R4, R5 versus theta2
    for i=1:4
        plot(fullTable.theta2, fullTable.
(posColNames{i}),'-x','MarkerIndices',[posMinMax.
(posColNames{i}).'],'color',graphColors{i})
        hold on
    end
    hold off

    % adding plot title
    title('Position Analysis')
    % creating legend for plot
    legend('\theta3_{(rad)}','R3_{(in)}','R4_{(in)}','R5_{(in)}')
    % labeling the x & y axes
    xlabel('\theta2_{(rad)}')
    ylabel('Outputs')
    % setting xtick values and labels
    xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
    % adding gridlines
    grid on
    grid minor
    % setting the limits of the axis
    xlim([0,2*pi])

    % adding labels to the marks denoting the maximums and minimums
    for i=1:2
        for j=1:4
            % getting the x coordinate of the mark
            xPoint =
 fullTable.theta2(posMinMax{rowMinMax{i},posColNames{j}});
            % getting the y coordinate of the mark
            yPoint = fullTable.(posColNames{j})
(posMinMax{rowMinMax{i},posColNames{j}});
            % combining all the information into a sinlge formatted
 string for the label
            labelString = sprintf('local %s\n%s=%0.4f @ %s2=
%0.4f',rowMinMax{i},posColNames{j},yPoint,char(952),xPoint);
            % adding the text to the graph
```

```
            text(xPoint, yPoint,
labelString,'VerticalAlignment','top','HorizontalAlignment','center')
        end
    end

    % saving the graph
    ax = gca;
    exportgraphics(ax,'position.jpg')
```

1st Order Kinematic Coefficients Graph

```
    % defining columns for the First Order Kinematic Coefficients
 graph data
    firstOrderColNames = {'h3','f3','f4','f5'};
    % initializing the table for the minimums and maximums
    firstOrderMinMax = table([0;0],[0;0],[0;0],
[0;0],'VariableNames',firstOrderColNames,'RowNames',{'min','max'});
    % finding indices of the local minimums and maximums for the First
 Order Kinematic Coefficients graph
    for i=1:4
        firstOrderMinMax.(firstOrderColNames{i}) = [find(fullTable.
(firstOrderColNames{i}) == min(fullTable.(firstOrderColNames{i})));
 find(fullTable.(firstOrderColNames{i}) == max(fullTable.
(firstOrderColNames{i})))];
    end

    % defining figure
    figure('Name','1st Order','position',[10,10,1200,1000])
    % plotting theta3, R3, R4, R5 versus theta2
    % plotting the mins and maxes
    for i=1:4
        plot(fullTable.theta2, fullTable.
(firstOrderColNames{i}),'-x','MarkerIndices',[firstOrderMinMax.
(firstOrderColNames{i}).'],'color',graphColors{i})
        hold on
    end
    % plotting the zeros
    for i=1:4
        plot(fullTable.theta2(posMinMax.(posColNames{i})),
[0;0],'x','color',graphColors{i})
        hold on
    end
    hold off

    % adding plot title
    title('1st Order Kinematic Coefficients')
    % creating legend for plot
    legend('h3_{(-)}','f3_{(length)}','f4_{(length)}','f5_{(length)}')
    % labeling the x & y axes
    xlabel('\theta2_{(rad)}')
    ylabel('Outputs')
    % setting xtick values and labels
    xticks(0:pi/4:2*pi)
```

```
xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
    % adding gridlines
    grid on
    grid minor
    % setting the limits of the axis
    xlim([0,2*pi])
    ylim([-1.5, 2.5])

    % adding labels to the marks denoting the maximums and minimums
    % setting the alignments of the labels relative to the marks
    firstVertAlign =
{'top','top','top','bottom';'top','top','top','top'};
    for i=1:2
        for j=1:4
            % getting the x coordinate of the mark
            xPoint =
fullTable.theta2(firstOrderMinMax{rowMinMax{i},firstOrderColNames{j}});
            % getting the y coordinate of the mark
            yPoint = fullTable.(firstOrderColNames{j})
(firstOrderMinMax{rowMinMax{i},firstOrderColNames{j}});
            % combining all the information into a sinlge formatted
string for the label
            labelString = sprintf('local %s\n%s=%0.4f @ %s2=%0.4f
\n',rowMinMax{i},firstOrderColNames{j},yPoint,char(952),xPoint);
            % adding the text to the graph
            text(xPoint, yPoint,
labelString,'VerticalAlignment',firstVertAlign{i,j},'HorizontalAlignment','center
        end
    end

    % adding labels to the marks denoting the zeros
    % setting the alignments of the labels relative to the marks
    firstVertAlign =
{'bottom','top','top','bottom';'bottom','bottom','top','top'};
    firstHorizAlign =
{'center','center','center','center';'center','left','center','center'};
    for i=1:2
        for j=1:4
            % getting the x coordinate of the mark
            xPoint =
fullTable.theta2(posMinMax{rowMinMax{i},posColNames{j}});
            % setting the y to 0
            yPoint = 0;
            % combining all the information into a sinlge formatted
string for the label
            labelString = sprintf('\n%s=%0.1f @ %s2=%0.4f
\n',firstOrderColNames{j},yPoint,char(952),xPoint);
            % adding the text to the graph
            text(xPoint, yPoint,
labelString,'VerticalAlignment',firstVertAlign{i,j},'HorizontalAlignment',firstHo
        end
    end
```

```
    ylim([-1.5, 2.5])

    % saving the graph
    ax = gca;
    exportgraphics(ax,'1stOrderKinCoeff.jpg')
```

2nd Order Kinematic Coefficients Graph

```
    % defining columns for the Second Order Kinematic Coefficients
 graph data
    secondOrderColNames = {'h3p','f3p','f4p','f5p'};
    % initializing the table for the minimums and maximums
    secondOrderMinMax = table([0;0],[0;0],[0;0],
[0;0],'VariableNames',secondOrderColNames,'RowNames',{'min','max'});
    % finding indices of the local minimums and maximums for the
 secondOrderition graph
    for i=1:4
        secondOrderMinMax.(secondOrderColNames{i}) = [find(fullTable.
(secondOrderColNames{i}) == min(fullTable.(secondOrderColNames{i})));
 find(fullTable.(secondOrderColNames{i}) == max(fullTable.
(secondOrderColNames{i})))];
    end

    % defining figure
    figure('Name','2nd Order','position',[10,10,1200,1000])
    % plotting theta3, R3, R4, R5 versus theta2
    for i=1:4
        plot(fullTable.theta2, fullTable.
(secondOrderColNames{i}),'color',graphColors{i})
        hold on
    end
    % plotting the zeros
    for i=1:4
        plot(fullTable.theta2(firstOrderMinMax.
(firstOrderColNames{i})),[0;0],'x','color',graphColors{i})
        hold on
    end
    hold off

    % adding plot title
    title('2nd Order Kinematic Coefficients')
    % creating legend for plot

 legend("h3'_{(-)}","f3'_{(length)}","f4'_{(length)}","f5'_{(length)}")
    % labeling the x & y axes
    xlabel('\theta2_{(rad)}')
    ylabel('Outputs')
    % setting xtick values and labels
    xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
    % adding gridlines
    grid on
```

```
    grid minor
    % setting the limits of the axis
    xlim([0,2*pi])


    % adding labels to the marks denoting the maximums and minimums
    % setting the alignments of the labels relative to the marks
    secondVertAlign =
 {'bottom','top','bottom','bottom';'top','bottom','top','top'};
    secondHorizAlign =
 {'right','center','left','center';'center','left','center','left'};
    for i=1:2
        for j=1:4
            % getting the x coordinate of the mark
            xPoint =
 fullTable.theta2(firstOrderMinMax{rowMinMax{i},firstOrderColNames{j}});
            % setting the y to 0
            yPoint = 0;
            % combining all the information into a sinlge formatted
 string for the label
            labelString = sprintf('\n%s=%0.1f @ %s2=%0.4f
\n',secondOrderColNames{j},yPoint,char(952),xPoint);
            % adding the text to the graph
            text(xPoint, yPoint,
 labelString,'VerticalAlignment',secondVertAlign{i,j},'HorizontalAlignment',second
        end
    end

    % saving the graph
    ax = gca;
    exportgraphics(ax,'2ndOrderKinCoeff.jpg')

end
```

Function: validationByGrad

```
function validationByGrad()
    kinData = readtable('kinematicData.xlsx');
    % R3 values
    RCG3= 2.28;
    R3 = kinData.R3;
    R4 = kinData.R4;
    R5 = kinData.R5;

    % Theta values, converted into radians
    theta2 = kinData.theta2;
    theta3 = kinData.theta3;

    f4 = kinData.f4;
    f4p = kinData.f4p;
    h3 = kinData.h3;
    h3p = kinData.h3p;
    f3 = kinData.f3;
```

```
    f3p = kinData.f3p;
    f5 = kinData.f5;
    f5p = kinData.f5p;

    f3grad=gradient(R3,theta2);
    diffR3=abs((f3grad-f3)./f3);
    f3percent=sum(diffR3)/360*100

    f3pgrad=gradient(f3,theta2);
    diffR3p=abs((f3pgrad-f3p)./f3p);
    f3ppercent=sum(diffR3p)/360*100

    f4grad=gradient(R4,theta2);
    diffR4=abs(f4grad-f4)./f4;
    f4percent=sum(diffR4)/360*100

    f4pgrad=gradient(f4,theta2);
    diffR4p=abs((f4pgrad-f4p)./f4p);
    f4ppercent=sum(diffR4p)/360*100

    f5grad=gradient(R5,theta2);
    diffR5=abs((f5grad-f5)./f5);
    f5percent=sum(diffR5)/360*100

    f5pgrad=gradient(f5,theta2);
    diffR5p=abs((f5pgrad-f5p)./f5p);
    f5ppercent=sum(diffR5p)/360*100

    h3grad=gradient(theta3,theta2);
    diffth3=abs((h3grad-h3)./h3);
    h3percent=sum(diffth3)/360*100

    h3pgrad=gradient(h3,theta2);
    diffth3p=abs((h3pgrad-h3p));
    diffth3p./h3p;
    h3ppercent=sum(diffth3p)/360*100
end
```

Function: IDP

```
function ax = IDP(w2, P4)

    KinematicDataMatrix =
 readmatrix('kinematicData.xlsx', 'Range', 'A:Q');%readtable to ref
 wirh column names
    kinData = readtable('kinematicData.xlsx');

    % Givens - R values in inches
    R1 = 3.52;
    R2 = 0.82;
    R3 = KinematicDataMatrix(:,3);
    R5 = KinematicDataMatrix(:,5);
```

```
    % Theta values, converted into radians
    theta2 = KinematicDataMatrix(:,1);
    theta3 = KinematicDataMatrix(:,2);
    theta5 = theta3;

    %inputs for accelerations
    f_g3x = KinematicDataMatrix(:,14);
    f_g3y = KinematicDataMatrix(:,15);
    fp_g3x = KinematicDataMatrix(:,16);
    fp_g3y = KinematicDataMatrix(:,17);
    f_g4x = KinematicDataMatrix(:,8);
    fp_g4x = KinematicDataMatrix(:,12);
    h3 = KinematicDataMatrix(:,6);
    h3p = KinematicDataMatrix(:,10);

     g = 32.2; %in/s^2

%      m2 = 0; %slug
%      Ig_2 = 0; %slug*in^2
%
%      m3 = 0; %slug
%      Ig_3 = 0; %slug*in^2
%
%      m4 = 0; %slug

    m2 = 0.027/32.2; %slug
    Ig_2 = 0.0088/32.2; %slug*in^2

    m3 = 0.1135/32.2; %slug
    Ig_3 = 0.3786/32.2; %slug*in^2

    m4 = .0812/32.2; %slug

    Rcg3 = 2.28 %inches, pythag thm from pic

    iter = length(KinematicDataMatrix(:,1)');

    sz = [361 13];
    varNames =
 ["F12x","F12y","F23","F23x","F23y","F13","F13x","F13y","F34x","F34y","F14","R7F14
    varTypes = repmat("double",1,13);
    forcesIDP =
 table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);

    for i = 1:iter
        %accelerations
        alpha2 = 0;
        alpha3 = h3(i)*alpha2 + h3p(i)*w2^2;

        a_g2x = 0;
        a_g2y = 0;

        a_g3x = (f_g3x(i)*alpha2 + fp_g3x(i)*w2^2)/12; % ft/s^2
        a_g3y = (f_g3y(i)*alpha2 + fp_g3y(i)*w2^2)/12; % ft/s^2
```

```
        a_g3x = (f_g4x(i)*alpha2 + fp_g4x(i)*w2^2)/12; % ft/s^2
        a_g3y = 0;

        a_g4y = 0;
        a_g4x = (f_g4x(i)*alpha2 + fp_g4x(i)*w2^2)/12; % fp_g4x = fp4
 % ft/s^2

        % A = 9x9
        A = [1, 0, -cos(theta3(i)-(3*pi/2)),                0,
   0,    0, 0, 0, 0;
             0, 1, -sin(theta3(i)-(3*pi/2)),                0,
   0,    0, 0, 0, 0;
             0, 0, cos(theta3(i)-(3*pi/2)),     cos(theta3(i)-
(3*pi/2)), -1,   0, 0, 0, 0;
             0, 0, sin(theta3(i)-(3*pi/2)),     sin(theta3(i)-
(3*pi/2)), 0,   -1, 0, 0, 0;
             0, 0,         0,                                0,
   1,    0, 0, 0, 0; %why did you have a 1 in this line for F_13?
             0, 0,         0,                                0,
   0,    1, 1, 0, 0;
             0, 0, -R2*sin(theta2(i)-theta3(i) + pi/2),     0,
   0,    0, 0, 0, 1; %I don't think you need that +pi/2
             0, 0,        R3(i),                       R3(i)-R5(i),
 0,    0, 0, 0, 0; %ccw vs cw
             0, 0,         0,                                0,
   0,    0, 0, 1, 0];

        J = [m2*a_g2x;
             m2*a_g2y + m2*g; %center of gravity is at pin + m2*g;
             m3*a_g3x;
             m3*a_g3y + m3*g;
             m4*a_g4x - P4;
             m4*a_g4y + m4*g;
             Ig_2*alpha2/12;
             Ig_3*alpha3/12 + m3*Rcg3*(cos(theta3(i))*a_g3y -
 sin(theta3(i))*a_g3x + m3*g*cos(theta3(i)));
             0]; %change you can't do this without also accounting for
 the other forces on 4

        x = A\J;

        forcesIDP.F12x(i) = x(1);
        forcesIDP.F12y(i) = x(2);

        forcesIDP.F23(i) = x(3);
        forcesIDP.F23x(i) = x(3)*cos(theta3(i)-(3*pi/2));
        forcesIDP.F23y(i) = x(3)*sin(theta3(i)-(3*pi/2));

        forcesIDP.F13(i) = x(4);
        forcesIDP.F13x(i) = x(4)*cos(theta3(i)-(3*pi/2));
        forcesIDP.F13y(i) = x(4)*sin(theta3(i)-(3*pi/2));

        forcesIDP.F34x(i) = x(5);
```

```
        forcesIDP.F34y(i) = x(6);

        forcesIDP.F14(i) = x(7);
        forcesIDP.R7F14(i) = x(8);

        forcesIDP.T2(i) = x(9);

    end

    filename = 'forcesIDP.xlsx';
    writetable(forcesIDP,filename,'Sheet',1,'Range','A1')
```

Plotting

```
    % defining colors for the graphs
    graphColors =
 {'#B58900','#cb4b16','#dc322f','#d33682','#6c71c4','#268bd2','#2aa198','#859900'}
    rowMinMax = {'min','max'};
```

Reaction Graph

```
    % defining columns for the position graph data

    % initializing the table for the minimums and maximums
    sz = [2 8];
    posColNames =
 {'F12x','F12y','F23','F13','F34x','F34y','F14','T2'};
    varTypes = repmat("double",1,8);
    posMinMax =
 table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',posColNames, 'RowNames

    % finding indices of the local minimums and maximums for the
 position graph
    for i=1:8
        maxes = (find(forcesIDP.(posColNames{i}) == max(forcesIDP.
(posColNames{i}))));
        mins = (find(forcesIDP.(posColNames{i}) == min(forcesIDP.
(posColNames{i}))));
        posMinMax.(posColNames{i}) = [mins(1); maxes(1)];
    end

    % defining the figure
    figure('Name','Position','position',[10,10,1200,1000])
    % plotting theta3, R3, R4, R5 versus theta2
    for i=1:8
        plot(kinData.theta2, forcesIDP.
(posColNames{i}),'-x','MarkerIndices',[posMinMax.
(posColNames{i}).'],'color',graphColors{i})
        hold on
    end
    hold off
    disp(forcesIDP.T2(78))
```

```
    % adding plot title
    tit = sprintf('Joint Force Analysis at w2 = %.2f and P4 = %.2f',
 w2, P4)
    title(tit)
    % creating legend for plot
    legend(posColNames)
    % labeling the x & y axes
    xlabel('\theta2_{(rad)}')
    ylabel('Joint Forces')
    % setting xtick values and labels
    xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
    % adding gridlines
    grid on
    grid minor
    % setting the limits of the axis
    xlim([0,2*pi])

    % adding labels to the marks denoting the maximums and minimums
    % for i=1:2
    %     for j=1:8
    %         % getting the x coordinate of the mark
    %         xPoint =
 kinData.theta2(posMinMax{rowMinMax{i},posColNames{j}});
    %         % getting the y coordinate of the mark
    %         yPoint = kinData.(posColNames{j})
(posMinMax{rowMinMax{i},posColNames{j}});
    %         % combining all the information into a sinlge formatted
 string for the label
    %         labelString = sprintf('local %s\n%s=%0.4f @ %s2=
%0.4f',rowMinMax{i},posColNames{j},yPoint,char(952),xPoint);
    %         % adding the text to the graph
    %         text(xPoint, yPoint,
 labelString,'VerticalAlignment','top','HorizontalAlignment','center')
    %     end
    % end

    % saving the graph
    ax = gca;
    saveName = sprintf('jointForces_w2_%.2f_P4_%.2f.jpg', w2, P4)
    exportgraphics(ax,saveName)

end
```

Function: torqueConstant

```
function torqueConstant(w2,P4)
```

Program Setup

```
    % reading in the data
    fullTable = readtable('kinematicData.xlsx');
```

```
    % defining and intitializing the table
    sz = [361 2];
    varNames = ["theta2","T2"];
    varTypes = repmat("double",1,2);
    torDataConst =
table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);

    % in lbm and convert to slugs
    m2 = 0.027/32.2;
    m3 = 0.1135/32.2;
    m4 = 0.0812/32.2;

    % in lbm*in^2 and convert to slug*ft*in
    Ig2 = 0.0088/32.2/12;
    Ig3 = 0.3786/32.2/12;

    g = 32.2 % ft/s^2

    for i = 1:361
        h2 = 1;
        h2p = 0;

        h3 = fullTable.h3(i);
        h3p = fullTable.h3p(i);
        fg3x = fullTable.fg3x(i);
        fg3y = fullTable.fg3y(i);
        fg3xp = fullTable.fg3xp(i);
        fg3yp = fullTable.fg3yp(i);

        fg4x = fullTable.f4(i);
        fg4xp = fullTable.f4p(i);

    %     if fg4x*w2 > 0
    %         P4 = -1.5;
    %     else
    %         P4 = 0;
    %     end

        dIe_dt = 2*(Ig2*h2*h2p) + 2*(m3*(fg3x*fg3xp + fg3y*fg3yp)/12 +
Ig3*h3*h3p) + 2*(m4*fg4x*fg4xp)/12;

        T2 = 0.5*dIe_dt*w2^2 + m3*g*fg3y - P4*fg4x;

        torDataConst.theta2(i) = (i-1)*pi/180;
        torDataConst.T2(i) = T2;
    end

    figure(3)
    plot(torDataConst.theta2,torDataConst.T2)
    xlabel('theta2 (rad)')
    ylabel('torque (lbs)')
    legend('torque')

    % writing the data to an excel file
```

```
    filename = 'torqueConstant.xlsx';
    writetable(torDataConst,filename,'Sheet',1,'Range','A1')

end
```

Function: torqueStartup

```
function [torDataMax,maxTorStartup] = torqueStartup(dw2,P4)
```

Program Setup

```
    % Code takes approximately 2 minutest to finish
    % requires input of desired w2 value in radians
    % and the applied load as P4

    % reading in the kinematic data
    fullTable = readtable('kinematicData.xlsx');

    % in lbf
    % P4 = 1.5;

    % dw2 = 10*2*pi/60;
    f = 2*pi;
    m = dw2*(f/(2*pi));
    tSpan = 1;

    % defining the function for a smooth acceleration
    accel = @(x) ( m*(1 - cos(f*x)) );
    % also defines what the velocity and displacement should look like
    % given continuos integration of the acceleration function
    vel = @(x) ( m*(x - (1/f)*sin(f*x)) );
    displacement = @(x) ( m*(0.5*x^2 + (1/f^2)*cos(f*x) - 1/f^2) );
```

Finding Worst Startup Position

```
    sz = [361 5];
    varNames =
{'theta2Init','theta2MaxTor','timeMaxTor','maxTor','sheetName'};
    varTypes = repmat("double",1,5);
    varTypes(5) = "string";
    maxTorStartup =
table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);
    startTor = 0;

    for j = 1:361
        theta2 = j-1; % degrees
        maxTorStartup.theta2Init(j) = theta2*pi/180;
        [theta2MaxTor, timeMaxTor, maxTor, sheetName, torData] =
startupSim(fullTable,accel,vel,displacement,theta2,dw2,P4);

        maxTorStartup.theta2MaxTor(j) = theta2MaxTor;
        maxTorStartup.timeMaxTor(j) = timeMaxTor;
```

```
        maxTorStartup.maxTor(j) = maxTor;
        maxTorStartup.sheetName(j) = sheetName;

        disp(abs(maxTor) >= abs(startTor))
        if abs(maxTor) >= abs(startTor)
            torDataMax = torData;
            startTor = maxTor;
        end
    end
```

Writing the data for worst position to an excel file

```
    index = find(maxTorStartup.maxTor == startTor);
    disTheta2 = maxTorStartup.theta2Init(index);
    sheetName = maxTorStartup.sheetName(index);

    maxTorSheetName = sprintf('iTh2=%.4f,w2=%.2f,P4=
%.2f',disTheta2,dw2,P4)
    filename = 'maxTor.xlsx';

 writetable(maxTorStartup,filename,'Sheet',maxTorSheetName,'Range','A1')

    filename = 'torqueStartup.xlsx';
    writetable(torDataMax,filename,'Sheet',sheetName,'Range','A1')
```

Graphing data for startup torque

```
    %sheetName = 'initTh2 = 48.0000';
    graphStartup(sheetName,maxTorSheetName,index,dw2,P4)

    ran = true;
```

Startup Simulation Function

```
    function [theta2MaxTor, timeMaxTor, maxTor, sheetName, torData] =
 startupSim(fullTable,accel,vel,displacement,theta2,dw2,P4)
        sheetName = sprintf('iTh2=%.4f,w2=%.2f,P4=%.2f',theta2,dw2,P4)
        theta2 = theta2*pi/180; % convert to radians

        % defining and intitializing the table
        sz = [1501 11];
        varNames =
 {'t','T2','theta2Sim','theta2Fun','w2Sim','w2Fun','a2','Ie','dIe_dt','deg','pos'}
        varTypes = repmat("double",1,11);
        torData =
 table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);

        % in lbm and convert to slugs
        m2 = 0.027/32.2;
        m3 = 0.1135/32.2;
```

```
        m4 = 0.0812/32.2;

        % in lbm*in^2 and convert to slug*in^2
        Ig2 = 0.0088/32.2;
        Ig3 = 0.3786/32.2;

        % in rad/s
        w2 = 0;
        a2 = 0;
        t = 0;

        % in rad
        dth2 = 1*pi/180;

        % interpolation of KCs
        interp = @(col, deg, pos) ((fullTable.(col)(pos+1) -
fullTable.(col)(pos))*(deg+1-pos) + fullTable.(col)(pos));

        i = 0;
        while t < 1
            i = i + 1;
            if abs(w2) > 100
                fprintf('Stopped at %d iterations and w2 = %f', i, w2)
                break
            end

            deg = theta2*180/pi;
            while deg >= 360
                deg = mod(deg,360);
            end
            while deg < 0
                deg = 360 - mod(-deg,360);
            end

            pos = floor(deg+1);

            torData.deg(i) = deg;
            torData.pos(i) = pos;

            h2 = 1;
            h2p = 0;

            h3 = interp("h3", deg, pos);
            h3p = interp("h3p", deg, pos);
            fg3x = interp("fg3x", deg, pos);
            fg3y = interp("fg3y", deg, pos);
            fg3xp = interp("fg3xp", deg, pos);
            fg3yp = interp("fg3yp", deg, pos);

            fg4x = interp("f4", deg, pos);
            fg4xp = interp("f4p", deg, pos);

            Ie = (Ig2*h2^2) + (m3*(fg3x^2 + fg3y^2) + Ig3*h3^2) +
(m4*fg4x^2);
```

```
            dIe_dt = 2*(Ig2*h2*h2p) + 2*(m3*(fg3x*fg3xp + fg3y*fg3yp)
 + Ig3*h3*h3p) + 2*(m4*fg4x*fg4xp);

            % in seconds
            % dt = fzero(@(t)(displacement(t)-
(theta2+1*pi/180)),tSpan) - t;
            a2 = accel(t);
            if a2 > 0
                dt = max(roots([0.5*a2, w2, -dth2]));
                if dt > 0.01
                    dt = 0.01;
                end
            else
                dt = 0.001;
            end

            T2 = Ie*a2 + 0.5*dIe_dt*w2^2 + m3*32.2/12*fg3y - P4*fg4x;

            theta2 = theta2 + w2*dt + 0.5*a2*dt^2;
            w2 = w2 + a2*dt;
            t = t + dt;

            torData.t(i) = t;
            torData.T2(i) = T2;
            torData.theta2Sim(i) = theta2;
            torData.theta2Fun(i) = displacement(t);
            torData.w2Sim(i) = w2;
            torData.w2Fun(i) = vel(t);
            torData.a2(i) = a2;
            torData.Ie(i) = Ie;
            torData.dIe_dt(i) = dIe_dt;
        end
        torData = torData(1:i,
{'t','T2','theta2Sim','theta2Fun','w2Sim','w2Fun','a2','Ie','dIe_dt','deg','pos'})

        sMaxTor = max(torData.T2);
        sMinTor = min(torData.T2);
        if abs(sMaxTor) > abs(sMinTor)
            maxTor = sMaxTor;
        else
            maxTor = sMinTor;
        end

        index = find(torData.T2 == maxTor);
        timeMaxTor = torData.t(index);
        theta2MaxTor = torData.theta2Sim(index);
    end
```

Graphing Function

```
    function ran =
 graphStartup(sheetName,maxTorSheetName,index,dw2,P4)
        torDataGr = readtable('torqueStartup.xlsx','Sheet',sheetName);
```

```
        maxTorStartupGr =
 readtable('maxTor.xlsx','Sheet',maxTorSheetName);
        tspan = 1:height(torDataGr);

        figName = sprintf('w2 vs Time - w2 = %.4f and P4 =
 %.2f',dw2,P4);
        figure('Name',figName)
        plot(torDataGr.t(tspan), torDataGr.w2Sim(tspan))
        xlabel('time (s)')
        ylabel('w2 (rad/s)')
        title(figName)
        % adding gridlines
        grid on
        grid minor
        exportgraphics(gca,[figName,'.jpg'])

        figName = sprintf('Theta2 vs Time - w2 = %.4f and P4 =
 %.2f',dw2,P4);
        figure('Name',figName)
        plot(torDataGr.t(tspan), torDataGr.theta2Sim(tspan))
        xlabel('t (s)')
        ylabel('theta2 (rad)')
        title(figName)
        % adding gridlines
        grid on
        grid minor
        exportgraphics(gca,[figName,'.jpg'])

        figName = sprintf('Alpha2 vs Time - w2 = %.4f and P4 =
 %.2f',dw2,P4);
        figure('Name',figName)
        plot(torDataGr.t(tspan), torDataGr.a2(tspan))
        xlabel('t (s)')
        ylabel('a2 (rad/s^2)')
        title(figName)
        % adding gridlines
        grid on
        grid minor
        exportgraphics(gca,[figName,'.jpg'])

        figName = sprintf('Max Torque vs Initial Theta2 - w2 = %.4f
 and P4 = %.2f',dw2,P4);
        figure('Name',figName)
        plot(maxTorStartupGr.theta2Init, maxTorStartupGr.maxTor,'-
x','MarkerIndices',index)
        xlabel('theta2 (rad)')
        ylabel('torque (lbf*in)')
        title(figName)
        % seting xlims
        xlim([0,2*pi])
        % setting xtick values and labels
        xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
```

```
            % adding gridlines
            grid on
            grid minor
            exportgraphics(gca,[figName,'.jpg'])

            ran = 1;
        end

    end
```

Function: dynamicVsStatic

```
    function dynamicVsStatic()
        P=9; %lbf
        t2=7;
        g=32.2;

        m2= .027;
        m3=.1135; %lb
        m4= .0812; %lb
        RCG3= 1.3;
        IG3= .3717;
        IG2=.0086;

        w=1;

        kinData = readtable('kinematicData.xlsx');
        % R3 values
        R3 = kinData.R3;
        R4 = kinData.R4;
        R5 = kinData.R5;

        % Theta values, converted into radians
        theta2 = kinData.theta2;
        theta3 = kinData.theta3;

        %inputs for power
        fg3x = kinData.fg3x;
        fg3y = kinData.fg3y;
        fg3xp = kinData.fg3xp;
        fg3yp = kinData.fg3yp;
        f4 = kinData.f4;
        f4p = kinData.f4p;
        h3 = kinData.h3;
        h3p = kinData.h3p;

        f3 = kinData.f3;
        f3p = kinData.f3p;
        f5 = kinData.f5;
        f5p = kinData.f5p;

        D = abs(m3.*(f4-(RCG3.*cos(theta3)).*(f4p-RCG3.*h3p.*sin(theta3)-
    h3.^2.*cos(theta3))+IG3.*h3.*h3p + m4.*(f4.*f4p)).*w^2);
```

```
    U= abs((m2*g).*w + (m3*g).*f3*w + (m4*g).*f4*w);

    subplot (3,1,1)
    plot(theta2, U, 'r', theta2, D,'b')
    grid on %%adds major grid lines (every tik mark)
    grid minor%% adds 5 minor grid lines between each major grid line
    legend("Static Forces", "Dynamic Forces")

    %%add title and axis labels
    title('Static Forces Versus Dynamic Forces for Low Omega Values
 (1rad/sec)')
    xlabel('\theta2 (radians)')
    ylabel('Energy')

    w=1000;
    D = abs(m3.*(f4-(RCG3.*cos(theta3)).*(f4p-RCG3.*h3p.*sin(theta3)-
h3.^2.*cos(theta3))+IG3.*h3.*h3p + m4.*(f4.*f4p)).*w^2);
    U= abs((m2*g).*w + (m3*g).*f3*w + (m4*g).*f4*w);

    subplot (3,1,2)
    plot(theta2, U, 'r', theta2, D,'b')
    grid on %%adds major grid lines (every tik mark)
    grid minor%% adds 5 minor grid lines between each major grid line
    legend("Static Forces", "Dynamic Forces")

    %%add title and axis labels
    title('Static Forces Versus Dynamic Forces for High Omega
 Values(1000rad/sec)')
    xlabel('\theta2 (radians)')
    ylabel('Energy')
    w=35;
    D = abs(m3.*(f4-(RCG3.*cos(theta3)).*(f4p-RCG3.*h3p.*sin(theta3)-
h3.^2.*cos(theta3))+IG3.*h3.*h3p + m4.*(f4.*f4p)).*w^2);
    U= abs((m2*g).*w + (m3*g).*f3*w + (m4*g).*f4*w);

    subplot (3,1,3)
    plot(theta2, U, 'r', theta2, D,'b')
    grid on %%adds major grid lines (every tik mark)
    grid minor%% adds 5 minor grid lines between each major grid line
    legend("Static Forces", "Dynamic Forces")

    %%add title and axis labels
    title('Neither Static Forces nor Dynamic Forces Dominant(35rad/
sec)')
    xlabel('\theta2 (radians)')
    ylabel('Energy')
end
```

Function: frictionIDP

```
function frictionIDP(inc12,inc34,inc13,inc23,inc14,w2,P4)

    torqueConstant(w2,P4)
```

IDP for Project Mechanism

Bailey Smoorenburg, Connor McCarthy, Gavin Sheng, Jill Bohnet, Patrick Herke

```
%function ax = frictionIDP(w2, P4)
    tol = 1e-8;
    mu = 0.2;
    R = 0.2; % in
    rw = 1.7/2 % in % half the width of the slider
    rh = 1.0/2 % in % half the height of the slider

    kinData = readtable('kinematicData.xlsx');

    % Givens - R values in inches
    R1 = 3.52;
    R2 = 0.82;
    R3 = kinData.R3;
    R5 = kinData.R5;

    % Theta values, converted into radians
    th2 = kinData.theta2;
    th3 = kinData.theta3;

    %inputs for accelerations
    fg3x = kinData.fg3x;
    fg3y = kinData.fg3y;
    fg3xp = kinData.fg3xp;
    fg3yp = kinData.fg3yp;
    f4 = kinData.f4;
    f4p = kinData.f4p;
    h3 = kinData.h3;
    h3p = kinData.h3p;

    % additional inputs for friction
    f3 = kinData.f3;
    f5 = kinData.f5;

    g = 32.2; %ft/s^2

%     m2 = 0; %slug
%     Ig_2 = 0; %slug*in^2
%
%     m3 = 0; %slug
%     Ig_3 = 0; %slug*in^2
%
%     m4 = 0; %slug

    m2 = 0.027/32.2; %slug
    Ig_2 = 0.0088/32.2/12; %slug*ft*in

    m3 = 0.1135/32.2; %slug
    Ig_3 = 0.3786/32.2/12; %slug*ft*in
```

```
       m4 = .0812/32.2; %slug

       Rcg3 = 2.28; %inches, pythag thm from pic

       % initializing table to store values
       sz = [361 28];
       varNames =
['theta2',"F12x","F12y","F23","F23x","F23y","F13","F13x","F13y","F34x","F34y","F1
       varTypes = repmat("double",1,28);
       forcesIDP =
table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',varNames);

       F12 = 0;
       F34 = 0;
       F23n = 0;
       F13n = 0;
       F14 = 0;
       R7F14 = 0;

       for i = 1:361
           stop = 0;
   %         if f4*w2 > 0
   %             P4 = -10;
   %         else
   %             P4 = 0;
   %         end

           %accelerations
           alpha2 = 0;
           alpha3 = h3(i)*alpha2 + h3p(i)*w2^2; %rad/s^2

           a_g2x = 0;
           a_g2y = 0;

           a_g3x = (fg3x(i)*alpha2 + fg3xp(i)*w2^2)/12; %ft/s^2
           a_g3y = (fg3y(i)*alpha2 + fg3yp(i)*w2^2)/12; %ft/s^2

           a_g3x = (f4(i)*alpha2 + f4p(i)*w2^2)/12; %ft/s^2
           a_g3y = 0;

           a_g4y = 0;
           a_g4x = (f4(i)*alpha2 + f4p(i)*w2^2)/12; %ft/s^2

           fric = ones(5,1);

           % input directional indicator
           Din = sign(w2);

           % pin joint directional indicators
           D12 = Din;
           D34 = sign(h3(i))*Din;
           % pin in slot directional indicators
           if F23n > 0
               D23 = sign(f3(i) + R*(h3(i) - 1))*Din;
```

```
            else
                D23 = sign(f3(i) - R*(h3(i) - 1))*Din;
            end

            if F13n > 0
                D13 = sign((f3(i)-f5(i)) + R*h3(i))*Din;
            else
                D13 = sign((f3(i)-f5(i)) - R*h3(i))*Din;
            end

            % slider in slot directional indicator
            D14 = sign(-f4(i))*Din;

                F12 = 0;
                F34 = 0;
                F23n = 0;
                F13n = 0;
                F14 = 0;
                R7F14 = 0;

            for j = 1:1000
                % T12 = mu*R*|F12|*D12 acting on link 1
                T12 = mu*R*abs(F12)*D12; %lbf*in
                % T34 = mu*R*|F34|*D34 acting on link 4
                T34 = mu*R*abs(F34)*D34; %lbf*in
                % T13 = mu*R*F13*D13 acting on link 1
                T13 = mu*R*F13n*D13; %lbf*in
                % f13 = mu*F13n acting on link 1
                f13 = mu*abs(F13n)*D13; %lbf
                f13x = f13*cos(th3(i) - pi);
                f13y = f13*sin(th3(i) - pi);
                % T23 = mu*R*F23*D13 acting on link 2
                T23 = mu*R*F23n*D23; %lbf*in
                % f23 = mu*F23n acting on link 2
                f23 = mu*abs(F23n)*D23; %lbf
                f23x = f23*cos(th3(i) - pi);
                f23y = f23*sin(th3(i) - pi);
                % f14 acting on link 4
                N14 = [1 1; rw -rw]\[F14; R7F14]; %lbf
                f14 = mu*(abs(N14(1)) + abs(N14(2)))*D14; %lbf
                % the book swaps the signs on T14 depending on >0 but
I don;t
                % think that's necessary
                % it also is used if height above and below the block
C pin
                % aren't the same
                if N14(1) > 0
                    T14(1) = mu*rh*N14(1)*D14; %lbf*in
                else
                    T14(1) = mu*rh*N14(1)*D14; %lbf*in
                end

                if N14(2) > 0
                    T14(2) = mu*rh*N14(2)*D14; %lbf*in
```

```
                else
                    T14(2) = mu*rh*N14(2)*D14; %lbf*in
                end

                % storing the old friction values
                fricOld = fric;
                fricStore{i,j} = fric;

                if inc12
                    fric(1) = (1/sqrt(1/mu^2 + 1))*F12;
                end
                if inc34
                    fric(2) = (1/sqrt(1/mu^2 + 1))*F34;
                end
                if inc13
                    fric(3) = f13;
                end
                if inc23
                    fric(4) = f23;
                end
                if inc14
                    fric(5) = f14;
                end

%                 fric = [(1/sqrt(1/mu^2 + 1))*F12;
%                         (1/sqrt(1/mu^2 + 1))*F34;
%                         f13;
%                         f23;
%                         f14;]


                relError = norm(fric - fricOld)/norm(fricOld);
                if relError < tol || all(fric == fricOld)
                    fricStore{i,j+1} = fric;
                    stop = 1;
                    break
                end

                % A = 9x9
                A = [1, 0, -cos(th3(i)-(3*pi/2)),                0,
        0,    0, 0, 0, 0;
                     0, 1, -sin(th3(i)-(3*pi/2)),                0,
        0,    0, 0, 0, 0;
                     0, 0, cos(th3(i)-(3*pi/2)),     cos(th3(i)-
(3*pi/2)), -1,   0, 0, 0, 0;
                     0, 0, sin(th3(i)-(3*pi/2)),     sin(th3(i)-
(3*pi/2)), 0,   -1, 0, 0, 0;
                     0, 0,          0,                          0,
        1,    0, 0, 0, 0;
                     0, 0,          0,                          0,
        0,    1, 1, 0, 0;
                     0, 0, -R2*cos(th3(i)-th2(i)),     0,           0,
    0, 0, 0, 1;
```

```
                    0, 0,         R3(i),                      R3(i)-
R5(i),  0,    0, 0, 0, 0;
                    0, 0,          0,                              0,
           0,    0, 0, 1, 0];

                jKinem = [m2*a_g2x;
                        m2*a_g2y;
                        m3*a_g3x;
                        m3*a_g3y;
                        m4*a_g4x;
                        m4*a_g4y;
                        Ig_2*alpha2;
                        Ig_3*alpha3 + m3*Rcg3*(cos(th3(i))*a_g3y -
 sin(th3(i))*a_g3x + g*cos(th3(i)));
                        0];

   %            jForce = [f23x;
   %                    -m2*g + f23y;
   %                    -f13x + -f23x;
   %                    -m3*g + -f13y + -f23y;
   %                    P4 + f14;
   %                    -m4*g;
   %                    -T12 + T23  + R2*cos((th2(i)+pi/2) -
 (th3(i)-pi))*f23;
   %                    -T34 + -T13 + -T23;
   %                    T34 + T14(1) + T14(2)];

                jForce = [0;
                        -m2*g;
                        0;
                        -m3*g;
                        P4;
                        -m4*g;
                        0;
                        0;
                        0];

                jF12 = [0;
                        0;
                        0;
                        0;
                        0;
                        0;
                        -T12;
                        0;
                        0];

                jF34 = [0;
                        0;
                        0;
                        0;
                        0;
                        0;
                        0;
```

```
        -T34;
        T34];

jF13 = [0;
        0;
        -f13x;
        -f13y;
        0;
        0;
        0;
        -T13;
        0];

jF23 = [f23x;
        f23y;
        -f23x;
        -f23y;
        0;
        0;
        T23 + R2*cos((th2(i)+pi/2) - (th3(i)-pi))*f23;
        -T23;
        0];

jF14 = [0;
        0;
        0;
        0;
        f14;
        0;
        0;
        0;
        T14(1) + T14(2)];

J = jKinem - jForce;

if inc12
    J = J - jF12;
end
if inc34
    J = J - jF34;
end
if inc13
    J = J - jF13;
end
if inc23
    J = J - jF23;
end
if inc14
    J = J - jF14;
end

% J = jKinem - jForce - jF12 - jF34 - jF13 - jF14 -
jF23;
```

```
            x = A\J;

            F12 = sqrt(x(1)^2 + x(2)^2); % F12 = sqrt(F12x^2 +
F12y^2)
            F34 = sqrt(x(5)^2 + x(6)^2); % F34 = sqrt(F34x^2 +
F34y^2)
            F13n = x(4);
            F23n = x(3);
            F14 = x(7);
            R7F14 = x(8);

   %          fricOld = fric;
   %          fric = (1/sqrt(1/mu^2 + 1)).*[F12; F34; F13; F23;
F14];

            % x = [  1,    2,    3,    4,    5,    6,    7,    8,
  9]
            % x = [F12x, F12y, F23n, F13n, F34x, F34y, F14, R7F14,
T2]
        end

        forcesIDP.theta2(i) = th2(i);
        if inc12
            forcesIDP.f12(i) = fric(1);
            forcesIDP.T12(i) = T12;
        end
        if inc34
            forcesIDP.f34(i) = fric(2);
            forcesIDP.T34(i) = T34;
        end
        if inc13
         forcesIDP.f13(i) = fric(3);
            forcesIDP.T13(i) = T13;
        end
        if inc23
         forcesIDP.f23(i) = fric(4);
            forcesIDP.T23(i) = T23;
        end
        if inc14
         forcesIDP.f14(i) = fric(5);
            forcesIDP.T14_1(i) = T14(1);
            forcesIDP.T14_2(i) = T14(2);
        end

        forcesIDP.F12x(i) = x(1);
        forcesIDP.F12y(i) = x(2);

        forcesIDP.F23(i) = x(3);
        forcesIDP.F23x(i) = x(3)*cos(th3(i)-(3*pi/2));
        forcesIDP.F23y(i) = x(3)*sin(th3(i)-(3*pi/2));

        forcesIDP.F13(i) = x(4);
        forcesIDP.F13x(i) = x(4)*cos(th3(i)-(3*pi/2));
        forcesIDP.F13y(i) = x(4)*sin(th3(i)-(3*pi/2));
```

```
        forcesIDP.F34x(i) = x(5);
        forcesIDP.F34y(i) = x(6);

        forcesIDP.F14(i) = x(7);
        forcesIDP.R7F14(i) = x(8);

        forcesIDP.T2(i) = x(9);

        forcesIDP.N14_1(i) = N14(1);
        forcesIDP.N14_2(i) = N14(2);

        forcesIDP.iter(i) = j;
    end

    disp(max(forcesIDP.iter))
    filename = 'forcesIDP.xlsx';
    writetable(forcesIDP,filename,'Sheet',1,'Range','A1')
```

Plotting

```
    % defining colors for the graphs
    graphColors =
{'#B58900','#cb4b16','#dc322f','#d33682','#6c71c4','#268bd2','#2aa198','#859900'}
    rowMinMax = {'min','max'};
```

Joint Forces

```
    % defining columns for the position graph data

    % initializing the table for the minimums and maximums
    sz = [2 8];
    posColNames =
{'F12x','F12y','F23','F13','F34x','F34y','F14','T2'};
    varTypes = repmat("double",1,8);
    posMinMax =
table('Size',sz, 'VariableTypes',varTypes, 'VariableNames',posColNames, 'RowNames

    % finding indices of the local minimums and maximums for the
position graph
 %    for i=1:8
 %        maxes = (find(forcesIDP.(posColNames{i}) ==
max(forcesIDP.(posColNames{i}))));
 %        posMinMax.(posColNames{i}) = [find(forcesIDP.
(posColNames{i}) == min(forcesIDP.(posColNames{i}))); maxes(1)];
 %    end

    % defining the figure
    figure('Name','Joint Forces','position',[10,10,1200,1000])
    % plotting theta3, R3, R4, R5 versus theta2
    for i=1:8
        plot(kinData.theta2, forcesIDP.
(posColNames{i}),'color',graphColors{i})
```

```
            hold on
        end
        hold off

        % adding plot title
        tit = sprintf('Joint Force Analysis at w2 = %.2f and P4 =
%.2f', w2, P4)
        title(tit)
        % creating legend for plot
        legend(posColNames)
        % labeling the x & y axes
        xlabel('\theta2_{(rad)}')
        ylabel('Joint Forces')
        % setting xtick values and labels
        xticks(0:pi/4:2*pi)

xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
        % adding gridlines
        grid on
        grid minor
        % setting the limits of the axis
        xlim([0,2*pi])

        % saving the graph
        ax = gca;
        saveName = sprintf('jointReactionForces_w2_%.2f_P4_%.2f.jpg',
w2, P4)
        exportgraphics(ax,saveName)

   %     figure(20)
   %     plot(kinData.theta2, f4*w2, '-')
```

Friction Forces

```
    if inc12 || inc34 || inc13 || inc23 || inc14
        posColNames =
{'f12','f34','f13','f23','f14','T12','T34','T13','T23','T14_1','T14_2'};

        % defining the figure
        figure('Name','Friction','position',[10,10,1200,1000])
        % plotting theta3, R3, R4, R5 versus theta2
        for i=1:8
            plot(kinData.theta2, forcesIDP.
(posColNames{i}),'color',graphColors{i})
            hold on
        end
        hold off

        % adding plot title
        tit = sprintf('Joint Force Analysis at w2 = %.2f and P4 =
%.2f', w2, P4)
        title(tit)
        % creating legend for plot
```

```
        legend(posColNames)
        % labeling the x & y axes
        xlabel('\theta2_{(rad)}')
        ylabel('Joint Forces')
        % setting xtick values and labels
        xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
        % adding gridlines
        grid on
        grid minor
        % setting the limits of the axis
        xlim([0,2*pi])

        % saving the graph
        ax = gca;
        saveName = sprintf('jointFrictionForces_w2_%.2f_P4_%.2f.jpg',
 w2, P4)
        exportgraphics(ax,saveName)
    end
```

Graph T2 with and without friction

```
    noFric = readtable('torqueConstant.xlsx');

    figure('Name','compare')
    tit = sprintf('Torque for Constant w2 - w2 = %.2f and P4 = %.2f',
 w2, P4)
    plot(noFric.theta2,noFric.T2,...
        forcesIDP.theta2,forcesIDP.T2)
    legend('T2_{noFric}','T_{Fric}')
    title(tit)
    ylabel('torque (lbf*in)')
    xlabel('theta2 (rad)')
    minDiff = min(forcesIDP.T2-noFric.T2)
    maxDiff = max(forcesIDP.T2-noFric.T2)
    % seting xlims
    xlim([0,2*pi])
    % setting xtick values and labels
    xticks(0:pi/4:2*pi)

 xticklabels({'0','\pi/4','\pi/2','3\pi/4','\pi','5\pi/4','3\pi/2','7\pi/4','2\pi'
    % adding gridlines
    grid on
    grid minor
    dim = [.675 .5 .3 .3];
    str = sprintf('T2_{Fric} - T2_{noFric}\nmin diff: %f\nmax diff:
 %f', minDiff, maxDiff);
    annotation('textbox',dim,'String',str,'FitBoxToText','on');
    exportgraphics(gca,[tit,'.jpg'])

end
```

Published with MATLAB® R2020a