Jieh Meinhold
12/14/2018
UCLA ID: 505225582

162A Final Project

Task 1

The selected mechanism design is shown in Figure 1 below. The width of all the links except for link 6 is 30mm, and the out of plane thickness of all links is 30mm. More detailed dimensions and the center of mass location of the table (link 6) are shown in Figure 2. Critical lengths (distances between revolute joints), and mass properties for each link are given in Table 1.

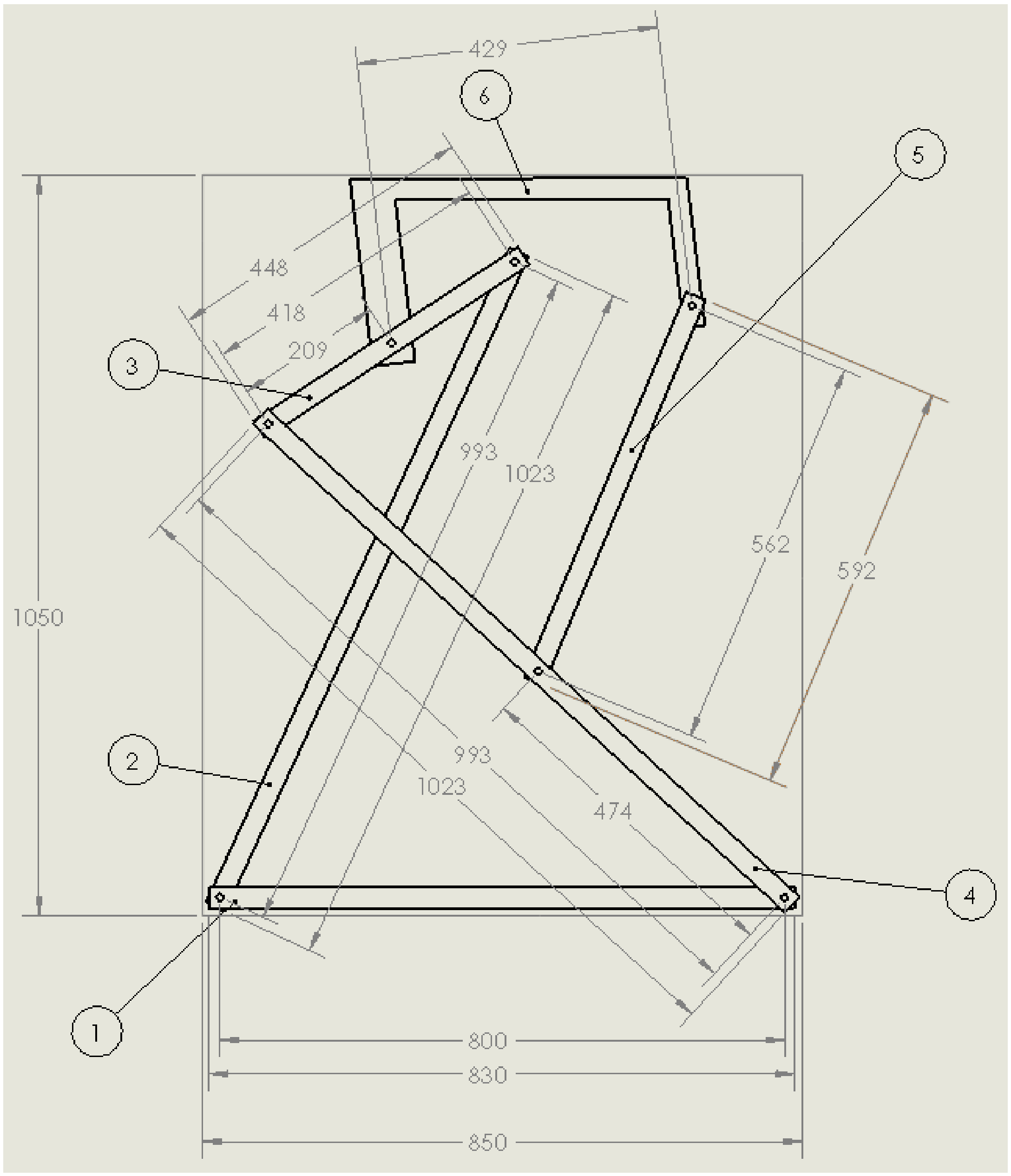


Figure 1: Mechanism dimensions (all dimensions in mm)

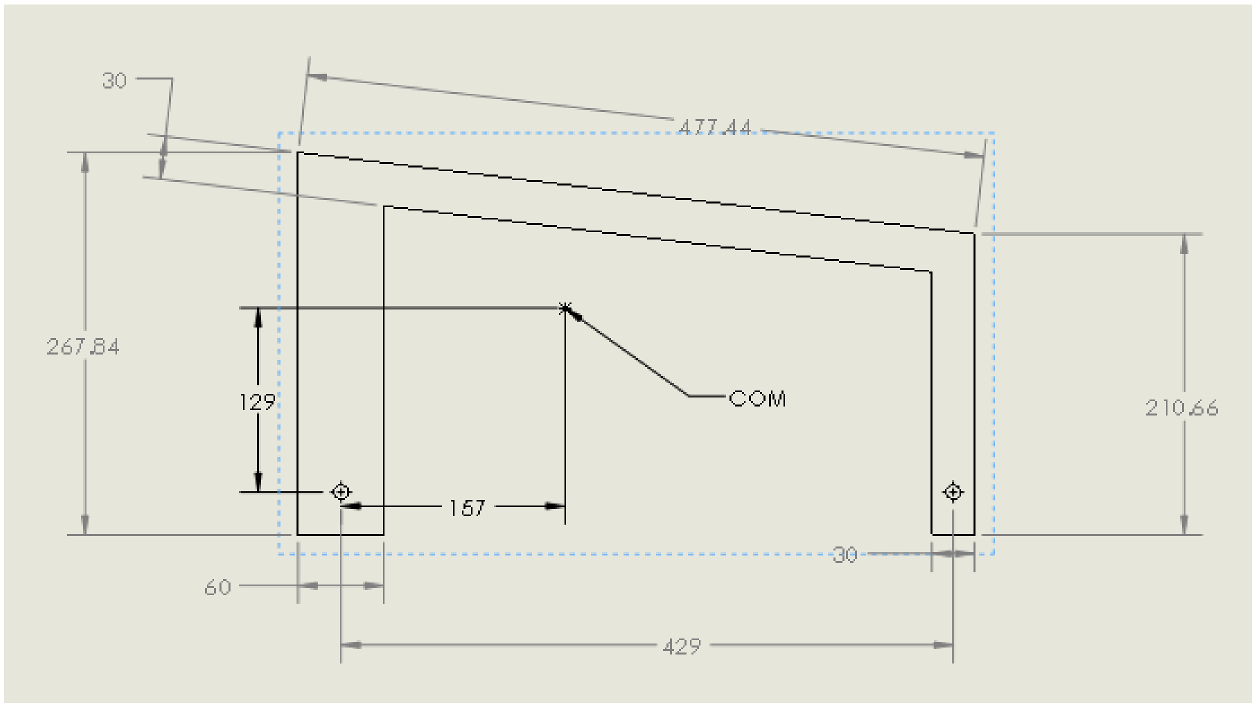


Figure 2: Link 6 (table) dimensions and center of mass location (all dimensions in mm)

Table 1: Link properties

Link #	D_1 (mm)	D_2 (mm)	MOI ($Kg*m^2$)	Mass (Kg)
1	800	NA	0.11390	2.00418
2	993	NA	0.21385	2.47317
3	418	209	0.01773	1.06955
4	993	474	0.21384	2.46680
5	562	NA	0.04112	1.42584
6	429	429	0.10079	2.72755

Note that D_1 is the distance between the furthest two revolute joints, and D_2 is the distance between the closest 2 revolute joints (if there are more than 2). The moments of inertia and masses were found using Solidworks. Moments of inertia listed are about the center of mass of the corresponding link. The mass properties are used for the power equation in task 5.

To design this mechanism, rough estimates of the link lengths were obtained using a simple Solidworks sketch (shown in Figure 3).

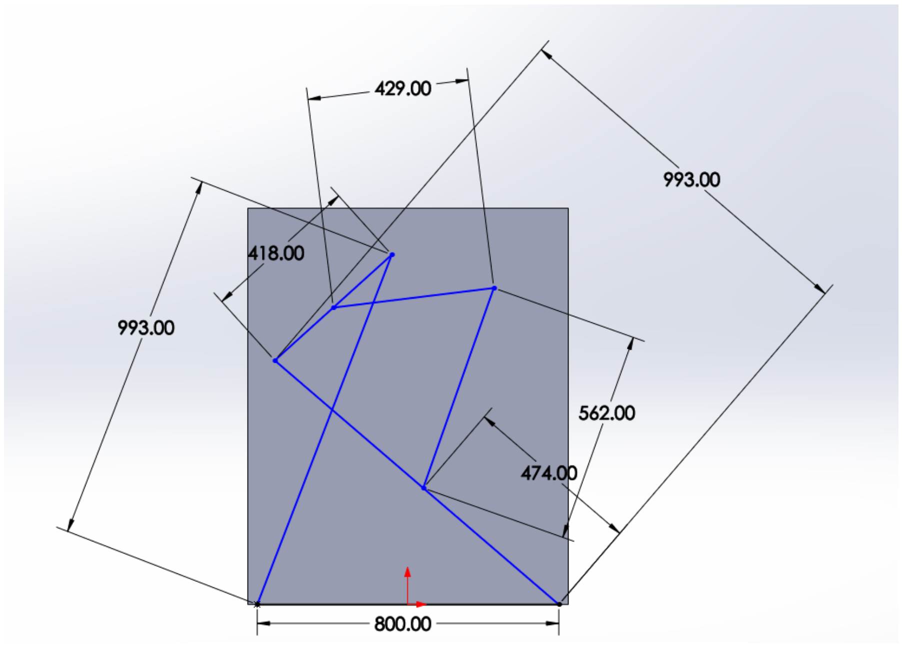


Figure 3: Initial Solidworks sketch (all dimension in mm)

Note that while this sketch shows the final dimensions, these were input after the MATLAB position analysis. Solidworks only allowed for near final dimensions because it is manual trial and error method. This sketch gave a visual representation of how the different links interacted. Links could be manually adjusted, and the resulting motion could be seen easily. It was also used to generate initial guesses for the unknowns in the vector loop equations so that Newton-Raphson functions would converge.

Two vector loops were used (Figure 4). The component forms were analyzed (appendix A1) to generate Jacobian matrices which were used to create MATLAB functions which perform Newton-Raphson's method to find the unknowns in each vector loop (MATLAB in Appendix A2 and A3).

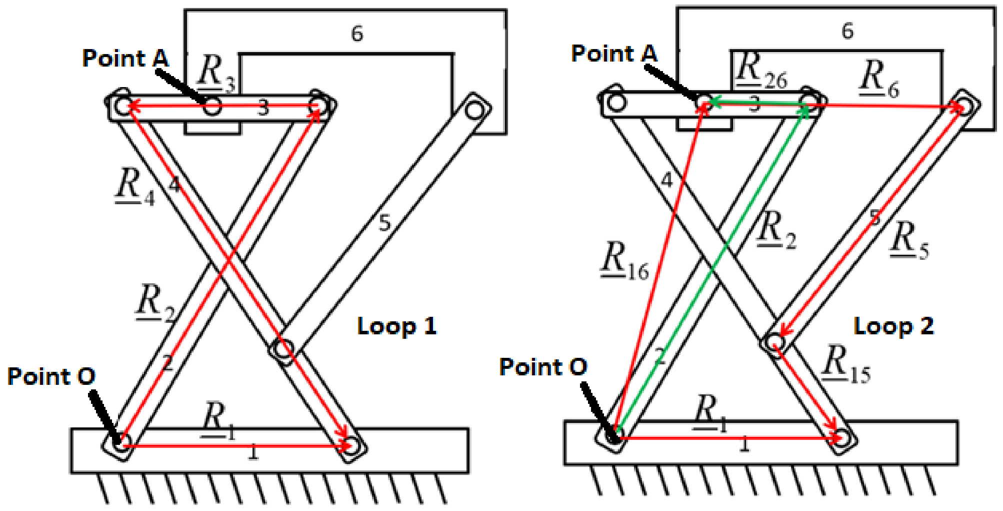


Figure 4: Vector loops 1 and 2

After this was completed, position analysis was performed at a range of input angles, while iterating over a range of link lengths centered around the estimates from Solidworks (see Appendix A4 for MATLAB code). After position analysis was performed for each combination of link lengths and input angle, the peak to peak x and y displacement of tables center of mass (from point O) was found using the vectors formed in the vector loops and the vector pointing from point A to the table center of mass R_{cm}. Since the angle between R_6 and R_{cm} is constant this analysis was simple to perform in MATLAB

For each link length combination, if the peak to peak Y displacement was less than 4mm, the peak to peak angular displacement of R6 was < 1 degree and the peak to peak X displacement was greater than 850mm, the link lengths were stored in an array. After this analysis was performed, one of the resulting combinations was chosen for the final design. This selected combination forms the design shown in Figure 1. For more detail on these computations, please see the provided code in appendix A4.

Task 2

Using the MATLAB script given in appendix B, the X displacement of the tables center of mass for each input angle in a range from 35.5 to 100 degrees was found. The input angle corresponding to 850mm displacement from its starting point (starting input angle of 35.5) was found. The resulting input angle range for this mechanism to achieve the desired motion is:

$$\theta_{2min} = 35.5 \text{ degrees } (0.6196 \text{ Radians})$$
$$\theta_{2max} = 97.1 \text{ degrees } (1.6946 \text{ Radians})$$

Task 3

Using the MATLAB script given in appendix B, the X displacement of the tables center of mass (from point O) was plotted against the input angle for the range found in task 2:

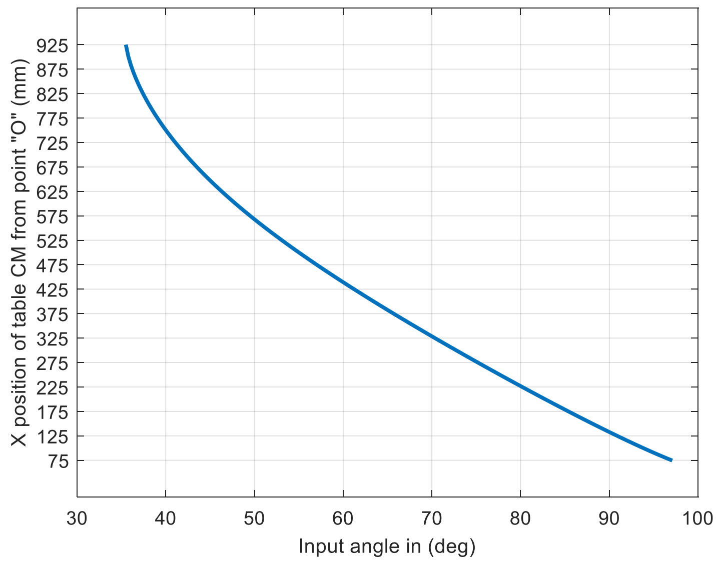


Figure 5: Table COM X displacement vs input angle

Note that the X displacement covers a range or 850mm over this input interval

Task 4

Using the MATLAB script given in appendix B, the Y displacement of the tables center of mass (from point O) was plotted against the input angle:

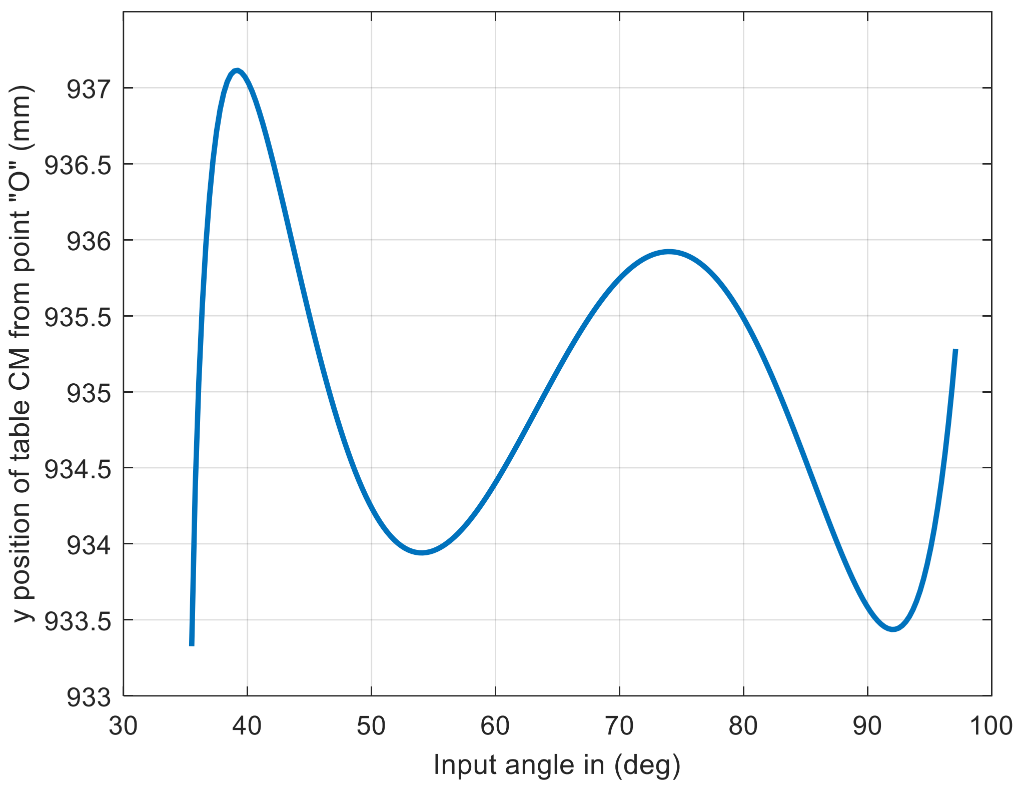


Figure 6: Table COM X displacement vs input angle

Note that the Y displacement covers a range of less than 4mm over this input interval

The peak to peak y deviation on this interval is 3.79 mm.

Task 6

Using the MATLAB script given in appendix B, the angular displacement vector R6. This vector tracks the motion of link 6. The angular displacement was plotted against the input angle:

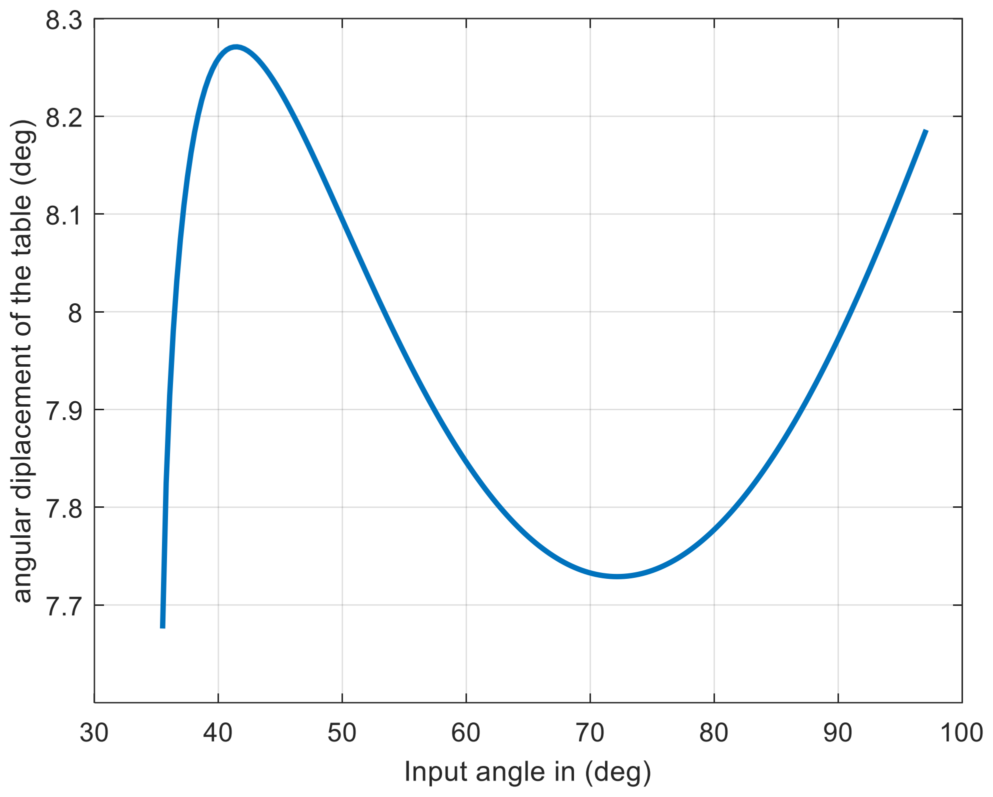


Figure 7: Table angular displacement vs input angle

Note that the angular displacement covers a range of less than 1 degree over this input interval

The peak to peak angular deviation on this interval is 0.6196 degrees

Task 5

MATLAB was used to simulate this mechanism for 3 cycles (1080 seconds) of the given sinusoidal input:

$$\theta_2 = \frac{(\theta_{2\max} - \theta_{2\min})}{2}\sin(\omega t) + \frac{(\theta_{2\max} + \theta_{2\min})}{2},$$

Where θ_{2min} and θ_{2max} correspond to the angles found in task 2.

This analysis gave the angle of each link at each discrete input angle. The x and y positions of the center of mass for each link was tracked using the vectors formed in the vector loops (the

COM of the rectangular links is merely the center of the link). This information was arranged into an array where the discrete differentiation command and the time discrete time step were used to find the angular velocity/acceleration of each link as well as the translational X and Y velocity/acceleration of each link. These were converted into first and second order kinematic coefficients using the input angular velocity and acceleration. Once this was completed the power equation was used to find the total power at each discrete time point. Note that friction was neglected, so the dissipative power term was 0. Dividing by the input velocity gives the input torque. Please see the commented MATLAB code in Appendix C for more information on these computations. The input torque was plotted against time. The input angle was also plotted for comparison:

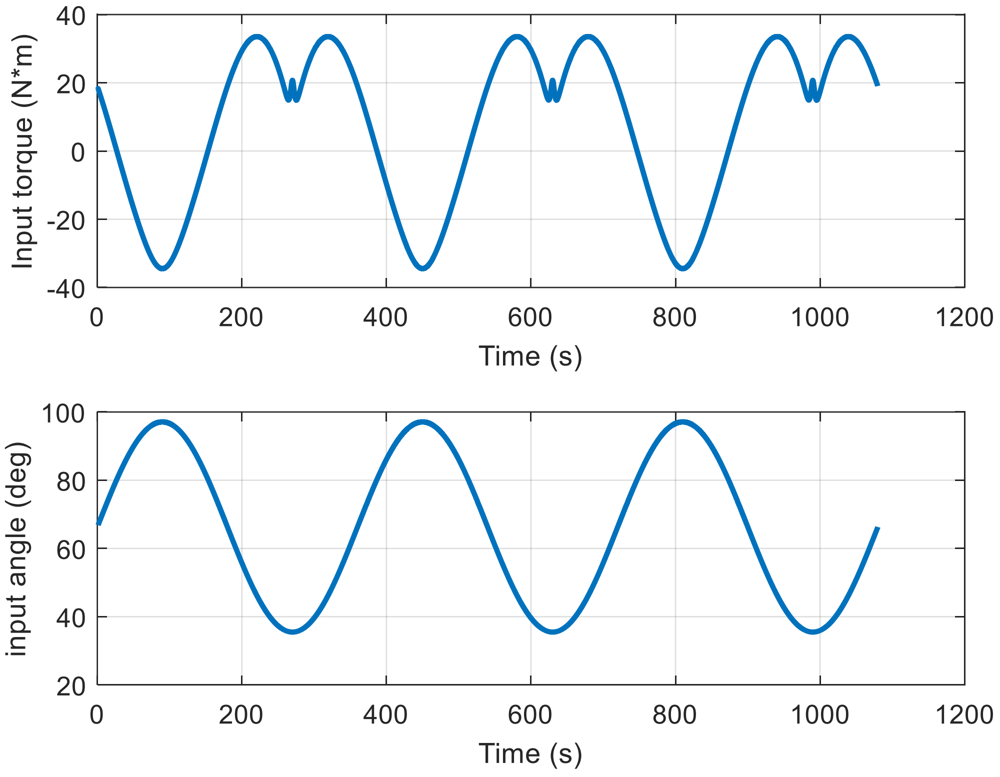
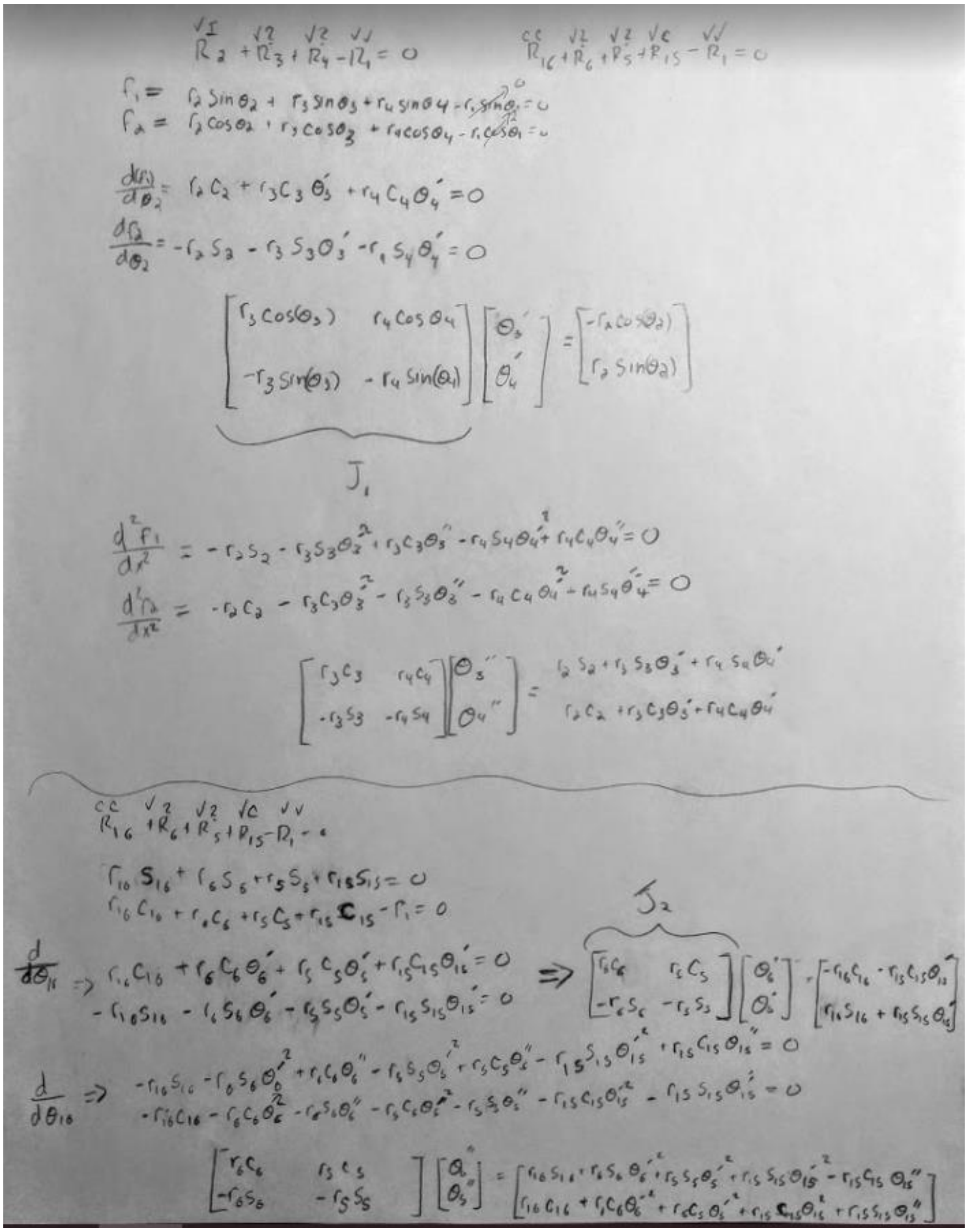


Figure 8: input torque vs time

One point that needed special consideration during this analysis was that while the convention for angular displacement is CCW from the positive X-axis, when using the diffs command in MATLAB, an angle that crosses 0 will suddenly jump to 2π. This results in an impulse in the time derivatives, To avoid this, the wrapToPi() function in MATLAB was used for θ_3 which crosses 0 during this cycle.

Appendix A-1: Hand calculations to find Jacobian matrices



$$\overset{\checkmark I}{R_2} + \overset{\checkmark ?}{R_3} + \overset{\checkmark ?}{R_4} - \overset{\checkmark\checkmark}{R_1} = 0 \qquad \overset{cc}{R_{16}} + \overset{\checkmark ?}{R_6} + \overset{\checkmark ?}{R_5} + \overset{\checkmark c}{R_{15}} - \overset{\checkmark\checkmark}{R_1} = 0$$

$$f_1 = r_2 \sin\theta_2 + r_3 \sin\theta_3 + r_4 \sin\theta_4 - \cancel{r_1 \sin\theta_1} = 0$$
$$f_2 = r_2 \cos\theta_2 + r_3 \cos\theta_3 + r_4 \cos\theta_4 - \cancel{r_1 \cos\theta_1} = 0$$

$$\frac{d(f_1)}{d\theta_2} = r_2 C_2 + r_3 C_3 \theta_3' + r_4 C_4 \theta_4' = 0$$
$$\frac{df_2}{d\theta_2} = -r_2 S_2 - r_3 S_3 \theta_3' - r_4 S_4 \theta_4' = 0$$

$$\underbrace{\begin{bmatrix} r_3\cos(\theta_3) & r_4\cos\theta_4 \\ -r_3\sin(\theta_3) & -r_4\sin(\theta_4) \end{bmatrix}}_{J_1} \begin{bmatrix} \theta_3' \\ \theta_4' \end{bmatrix} = \begin{bmatrix} -r_2\cos(\theta_2) \\ r_2\sin(\theta_2) \end{bmatrix}$$

$$\frac{d^2 f_1}{dx^2} = -r_2 S_2 - r_3 S_3 \theta_3'^2 + r_3 C_3 \theta_3'' - r_4 S_4 \theta_4'^2 + r_4 C_4 \theta_4'' = 0$$
$$\frac{d^2 f_2}{dx^2} = -r_2 C_2 - r_3 C_3 \theta_3'^2 - r_3 S_3 \theta_3'' - r_4 C_4 \theta_4'^2 - r_4 S_4 \theta_4'' = 0$$

$$\begin{bmatrix} r_3 C_3 & r_4 C_4 \\ -r_3 S_3 & -r_4 S_4 \end{bmatrix} \begin{bmatrix} \theta_3'' \\ \theta_4'' \end{bmatrix} = \begin{matrix} r_2 S_2 + r_3 S_3 \theta_3' + r_4 S_4 \theta_4' \\ r_2 C_2 + r_3 C_3 \theta_3' + r_4 C_4 \theta_4' \end{matrix}$$

$$\overset{cc}{R_{16}} + \overset{\checkmark ?}{R_6} + \overset{\checkmark ?}{R_5} + \overset{\checkmark c}{R_{15}} - \overset{\checkmark\checkmark}{R_1} = 0$$

$$r_{16} S_{16} + r_6 S_6 + r_5 S_5 + r_{15} S_{15} = 0$$
$$r_{16} C_{16} + r_6 C_6 + r_5 C_5 + r_{15} C_{15} - r_1 = 0$$

$$\frac{d}{d\theta_{16}} \Rightarrow r_{16} C_{16} + r_6 C_6 \theta_6' + r_5 C_5 \theta_5' + r_{15} C_{15} \theta_{15}' = 0$$
$$-r_{16} S_{16} - r_6 S_6 \theta_6' - r_5 S_5 \theta_5' - r_{15} S_{15} \theta_{15}' = 0$$

$$\Rightarrow \overbrace{\begin{bmatrix} r_6 C_6 & r_5 C_5 \\ -r_6 S_6 & -r_5 S_5 \end{bmatrix}}^{J_2} \begin{bmatrix} \theta_6' \\ \theta_5' \end{bmatrix} = \begin{bmatrix} -r_{16} C_{16} - r_{15} C_{15} \theta_{15}' \\ r_{16} S_{16} + r_{15} S_{15} \theta_{15}' \end{bmatrix}$$

$$\frac{d}{d\theta_{16}} \Rightarrow -r_{16} S_{16} - r_6 S_6 \theta_6'^2 + r_6 C_6 \theta_6'' - r_5 S_5 \theta_5'^2 + r_5 C_5 \theta_5'' - r_{15} S_{15} \theta_{15}'^2 + r_{15} C_{15} \theta_{15}'' = 0$$
$$-r_{16} C_{16} - r_6 C_6 \theta_6'^2 - r_6 S_6 \theta_6'' - r_5 C_5 \theta_5'^2 - r_5 S_5 \theta_5'' - r_{15} C_{15} \theta_{15}'^2 - r_{15} S_{15} \theta_{15}'' = 0$$

$$\begin{bmatrix} r_6 C_6 & r_5 C_5 \\ -r_6 S_6 & -r_5 S_5 \end{bmatrix} \begin{bmatrix} \theta_6'' \\ \theta_5'' \end{bmatrix} = \begin{bmatrix} r_{16} S_{16} + r_6 S_6 \theta_6'^2 + r_5 S_5 \theta_5'^2 + r_{15} S_{15} \theta_{15}'^2 - r_{15} C_{15} \theta_{15}'' \\ r_{16} C_{16} + r_6 C_6 \theta_6'^2 + r_5 C_5 \theta_5'^2 + r_{15} C_{15} \theta_{15}'^2 + r_{15} S_{15} \theta_{15}'' \end{bmatrix}$$

Appendix A-2: MATLAB function for Newton Raphson on vector loop 1

```
function [r1, r2, r3, r4, th2, th3, th4] = N_R_1(r1,r2,r3,r4,th2,th30,th40)

dth3 = 10;
dth4 = 10;
D = [dth3;dth4];

n = 0;
th3 = th30;
th4 = th40;
while norm(D,1)>.00001
n = n + 1;

f1 = r2*sin(th2) + r3*sin(th3) + r4*sin(th4);
f2 = r2*cos(th2) + r3*cos(th3) + r4*cos(th4) - r1;
F = [f1;f2];

%Evaluate Jacobian
J11 = r3*cos(th3);
J12 = r4*cos(th4);
J21 = -r3*sin(th3);
J22 = -r4*sin(th4);
J = [J11 J12; J21 J22];

%return dx, dy
D = -J\F;
dth3 = D(1);
dth4 = D(2);

%update guesses
th3 = th3 + dth3;
th4 = th4 + dth4;
end

theta3 = th3;
theta4 = th4;
th30 = wrapToPi(theta3);
th40 = wrapTo2Pi(theta4);

th4 = th40;
th3 = th30;
```

Appendix A-3: MATLAB function for Newton Raphson on vector loop 2

```
function [th6, th5] = N_R_2(r1, r16, r6, r5, r15, th16, th15, th60, th50)
% performs newton-Raphson on second vector loop

dth6 = 10;
dth5 = 10;
D = [dth6;dth5];


n = 0;
th6 = th60;
th5 = th50;
while norm(D,1)>.001
n = n + 1;

f1 = r16*sin(th16) + r6*sin(th6) + r5*sin(th5) + r15*sin(th15);
f2 = r16*cos(th16) + r6*cos(th6) + r5*cos(th5) + r15*cos(th15)- r1;
F = [f1;f2];

%Evaluate Jacobian
J11 = r6*cos(th6);
J12 = r5*cos(th5);
J21 = -r6*sin(th6);
J22 = -r5*sin(th5);
J = [J11 J12; J21 J22];

%return dx, dy
D = -J\F;
dth6 = D(1);
dth5 = D(2);
%update guesses
th6 = th6 + dth6;
th5 = th5 + dth5;
end

theta3 = th6;
theta4 = th5;
th60 = wrapTo2Pi(theta3);
th50 = wrapTo2Pi(theta4);

theta3 = wrapTo2Pi(theta3);
theta4 = wrapTo2Pi(theta4) ;
t3 = theta3/pi()*180;
t4 = theta4/pi()*180;
th6 = th60;
th5 = th50;
```

Appendix A-4: MATLAB script to iterate link lengths

```
clear all; close all; clc

th2_min = 35.5*pi/180;
th2_max = 98*pi/180;
r1 = 800;
th30 = 97*pi/180; th40 = 265*pi/180; th60 = 3*pi/180; th50 = pi+.5; %initial guesses for unknowns
a = 1;
t = 1;
lim1 = 4;
lim2 = 1;
lim3 = 850;
th2_range =  th2_min:.02:th2_max;
for r2 = 987:3:1000
    b = 1
    for r3 = 418:2:426
        r4 = r2;
        r6 = 350;
        r5 = 515;
        Xcm6 = 145; %with respect to point A on link 6
        Ycm6 = 136; %with respect to point A on link 6
        rcm = sqrt((Xcm6^2)+ (Ycm6^2));

        r6_min = 425; r6_max = 460;
        r5_min = 560; r5_max = 600;
        r15_min = 450; r15_max = 475;
        m = 1
        for r6 = r6_min:2:r6_max  %iterate through different r6 values
            k =1;
            for r5 = r5_min:2:r5_max  %iterate through different r5 values
                j = 1;
                for r15 = r15_min:2:r15_max  %iterate through different r15 values
                    i = 1;
                    for th2 = th2_range  %check these link lengths over range of inputs
                        [r1, r2, r3, r4, th2, th3(i), th4(i)] = N_R_1(r1,r2,r3,r4,th2,th30,th40);

                        % to track find R16
                        r26 = r3/2;
                        th26(i) = th3(i);
                        x16(i) = r2*cos(th2)+r26*cos(th26(i));
                        y16(i) = r2*sin(th2)+r26*sin(th26(i));

                        r16(i) = sqrt((y16(i)^2) + (x16(i)^2));
                        th16(i) = asin(y16(i)/r16(i));
                        th15(i) = th4(i);
                        % check this case
                        if x16(i) <0
                            th16(i) = pi-asin(y16(i)/r16(i));
                        end

                        [th6(i), th5(i)] = N_R_2(r1,r16(i),r6,r5,r15,th16(i),th15(i),th60,th50);
                        th60 = th6(i); th50 = th5(i);
                        th30 = th3(i); th40 = th4(i);
                        i = i+1;
                    end
                    DEV(j,k,m) = peak2peak(th6*180/pi); %find maximum deviation over input range

                    Xcm6 = 50; %with respect to point A on link 6
                    Ycm6 = 100; %with respect to point A on link 6
                    rcm = sqrt((Xcm6^2)+ (Ycm6^2));
                    thcm6 = atan(Ycm6/Xcm6);%with respect to point R6
                    % for i = 1:length(th6)
                    thCM = wrapTo2Pi(thcm6+th6);
                    thCM = wrapTo2Pi(thcm6+th6);
                    CMy = r16.*sin(th16) + rcm*sin(thCM);
                    CMx = r16.*cos(th16) + rcm*cos(thCM);
                    DEV2(j,k,m) = peak2peak(CMy);

                    if(DEV2(j,k,m)<lim1 && DEV(j,k,m)<lim2 && peak2peak(CMx)>lim3)
```

```
                        r_5(t) = r5
                        r_6(t) = r6
                        r_15(t) = r15
                        r_2(t) = r2
                        r_3(t) = r3
                        t = t+1
                        lim1 = lim1-.05;
                        lim2 = lim2-.05;
                        lim3 = lim3+1;
                    end
                    th60 = th6(1); th50 = th5(1);
                    th30 = th3(1); th40 = th4(1);
                    j = j+1;
                end
                k = k+1;
            end
            m = m+1;
        end

        b = b+1;
    end
    a = a+1;
end
```

Appendix B: MATLAB to analyze selected design

```
clear all; close all; clc;

r1 = 800; r2 = 993; r3 = 418; r4 = 993; r5 = 570; r6 = 435; r15 = 474;
%selected design lengths
th30 = 97*pi/180; th40 = 265*pi/180; th60 = 3*pi/180; th50 = pi+.5; %initial
guesses for unknowns

th2_min = 35.5*pi/180;
th2_max = 100*pi/180;
i = 1;
th2_range =  th2_min:.005:th2_max;
for th2 = th2_range  %check these link lengths over range of inputs
[r1, r2, r3, r4, th2, th3(i), th4(i)] = N_R_1(r1,r2,r3,r4,th2,th30, th40);

% to find r16 and th16 from other links
r26 = r3/2;
th26(i) = th3(i);
x16(i) = r2*cos(th2)+ r26*cos(th26(i));
y16(i) = r2*sin(th2)+ r26*sin(th26(i));

r16(i) = sqrt((y16(i)^2) + (x16(i)^2));
th16(i) = asin(y16(i)/r16(i));
th15(i) = th4(i);
% check this case
if x16(i) <0
    th16(i) = pi-asin(y16(i)/r16(i));

end
[th6(i), th5(i)] = N_R_2(r1, r16(i),r6,r5,r15,th16(i),th15(i),th60,th50);
th60 = th6(i); th50 = th5(i);
th30 = th3(i); th40 = th4(i);
i = i+1;
end

Xcm6 = 157; %with respect to point A on link 6
Ycm6 = 129; %with respect to point A on link 6
rcm = sqrt((Xcm6^2)+ (Ycm6^2));
thcm6 = atan(Ycm6/Xcm6);%with respect to point R6
thCM = wrapTo2Pi(thcm6+th6);
CMy = r16.*sin(th16) + rcm*sin(thCM);
CMx = r2*cos(th2_range)+ r3*cos(th3)/2 + rcm*cos(thCM);


% Print out max deviations to verify design requirements
[m,ind] = min(abs(CMx - CMx(1)+ 850));
ind = ceil(ind);
X_deviation = peak2peak(CMx(1:ind))
Y_deviation = peak2peak(CMy(1:ind))
angular_deviation = peak2peak(th6(1:ind)*180/pi)
th2_min = th2_min
th2_max = th2_range(ind)
```

```
th2_min_deg = th2_min*180/pi
th2_max_deg = th2_max*180/pi

figure(1)
plot(th2_range(1:ind)*180/pi,CMx(1:ind),'Linewidth',2)
yticks([round(min(CMx(1:ind))):50:ceil(max(CMx(1:ind)))])
grid on
xlabel('Input angle in (deg)')
ylabel('X position of table CM from point "O"')

figure(2)
plot(th2_range(1:ind)*180/pi,CMy(1:ind),'Linewidth',2)
yticks([round(min(CMy(1:ind))):.5:max(CMy(1:ind))])
grid on
xlabel('Input angle in (deg)')
ylabel('y position of table CM from point "O"')

figure(3)
plot(th2_range(1:ind)*180/pi,th6(1:ind)*180/pi,'Linewidth',2)
yticks([floor(min(th6(1:ind)*180/pi)):.1:ceil(max(th6(1:ind)*180/pi))])
grid on
xlabel('Input angle in (deg)')
ylabel('angular displacement of R6 (table) in degrees')
```

Appendix C: MATLAB to find input torque

```
clear all; close all; clc;

r1 = 800; r2 = 993; r3 = 418; r4 = 993; r5 = 570; r6 = 435; r15 = 474;
%selected design lengths
th30 = 97*pi/180; th40 = 265*pi/180; th60 = 3*pi/180; th50 = pi+.5; %initial
guesses for unknowns

th2_min = 35.5*pi/180;
th2_max = 97.1*pi/180;
i = 1;
wo = 1*pi/180  %angular frequency in radians/s

t_range = 1:.05:(360*3);
th2_range =  ((th2_max-th2_min)/2)*sin(wo*t_range) + (th2_max+th2_min)/2;
% th2_min:.0005:th2_max;
for t = t_range  %check these link lengths over range of inputs
th2 = ((th2_max-th2_min)/2)*sin(wo*t) + (th2_max+th2_min)/2;

[r1, r2, r3, r4, th2, th3(i), th4(i)] = N_R_1(r1, r2, r3, r4, th2, th30,
th40);

% to find r16 and th16 from other links
r26 = r3/2;
th26(i) = th3(i);
x16(i) = r2*cos(th2)+r26*cos(th26(i));
y16(i) = r2*sin(th2)+r26*sin(th26(i));

r16(i) = sqrt((y16(i)^2) + (x16(i)^2));
th16(i) = asin(y16(i)/r16(i));
th15(i) = th4(i);

if x16(i) <0
    th16(i) = pi-asin(y16(i)/r16(i));
end

[th6(i), th5(i)] = N_R_2(r1, r16(i), r6, r5, r15, th16(i), th15(i), th60,
th50);
th60 = th6(i); th50 = th5(i);
th30 = th3(i); th40 = th4(i);
i = i+1;
end

Xcm6 = 156; %with respect to point A on link 6
Ycm6 = 128; %with respect to point A on link 6

rcm = sqrt((Xcm6^2)+ (Ycm6^2));
thcm6 = atan(Ycm6/Xcm6);%with respect to point R6
% for i = 1:length(th6)
    thCM = wrapTo2Pi(thcm6+th6);
% end
CMy = r16.*sin(th16) + rcm*sin(thCM);
CMx = r2*cos(th2_range)+ r3*cos(th3)/2 + rcm*cos(thCM);
```

```
% Mass properties (meters and kilograms)
m(1) = 2.00418; m(2) = 2.47317; m(3) = 1.06955; m(4) = 2.46680; m(5) =
1.42584; m(6) = 2.72755;
I(1) = 0.11390; I(2) = 0.21385; I(3) = 0.01773; I(4) = 0.21384; I(5) =
0.04112; I(6) = 0.10079;

%%%%%%%%******make sure to find inertia about right point for link 6!!!
I(6) = I(6) + m(6)*rcm^2; %from Parallel axis thrm

% positions of the centers of each link
x(2,:) = r2*cos(th2_range);
x(3,:) = r2*cos(th2_range)+r3*cos(th3)/2;
x(4,:) = r2*cos(th2_range)+r3*cos(th3)+r4*cos(th4)/2;
x(5,:) = x(3,:)+r6*cos(th6)+r5*cos(th5)/2;
x(6,:) = CMx;

y(2,:) = r2*sin(th2_range);
y(3,:) = r2*sin(th2_range)+r3*sin(th3)/2;
y(4,:) = r2*sin(th2_range)+r3*sin(th3)+r4*sin(th4)/2;
y(5,:) = y(3,:)+r6*sin(th6)+r5*sin(th5)/2;
y(6,:) = CMy;

%angular displacements
th(2,:) = th2_range;
th(3,:) = th3;
th(4,:) = th4;
th(5,:) = th5;
th(6,:) = th6;


% Converting to meters
x = x/1000;
y = y/1000;


dt = t_range(2)-t_range(1); %time step

% Find time derivatives
for i = [2,3,4,5,6]
    dx(i,:) = diff(x(i,:))/dt;
    dy(i,:) = diff(y(i,:))/dt;
    dth(i,:) = diff(th(i,:))/dt;

    ddx(i,:) = diff(dx(i,:))/dt;
    ddy(i,:) = diff(dy(i,:))/dt;
    ddth(i,:) = diff(dth(i,:))/dt;
end
x = x(:,1:(end-2));
y = y(:,1:(end-2));
th = th(:,1:(end-2));

dx = dx(:,1:(end-1));
dy = dy(:,1:(end-1));
dth = dth(:,1:(end-1));
```

```
% angular velocity and angular acceleration of input (link t)
w = diff(th2_range)/dt;
a = diff(w)/dt;
w = w(:,1:(end-1));

% kinematic coefficients
xp = dx./w;
yp = dy./w;
thp = dth./w;

xpp = (ddx-xp.*a)./(w.^2);
ypp = (ddy-yp.*a)./(w.^2);
thpp = (ddth-thp.*a)./(w.^2);

%to evaluate the power equation
for i = [2,3,4,5,6]
    A(i,:) =  m(i)*((xp(i,:).^2)+(yp(i,:).^2))+I(i)*thp(i,:).^2;
    B(i,:) =  m(i)*((xp(i,:).*xpp(i,:))+(yp(i,:).*ypp(i,:)))+...
        I(i)*thp(i,:).*thpp(i,:);
    AA(i,:) = A(i,:).*w.*a;
    BB(i,:) = B(i,:).*w.^3;

    G(i,:) = m(i)*9.8*yp(i,:).*w;
end

DTt = sum(AA+BB);  %% KE temr
DUt = sum(G);   %%PE term


P = DTt+DUt; %total power
T = P./w; % torque
%% Plots
subplot(2,1,1)
plot(t_range(1:end-2),T,'Linewidth',2)
xlabel('Time (s)')
ylabel('Input torque (N*m)')
grid on


subplot(2,1,2)
plot(t_range,th2_range*180/pi,'Linewidth',2)
xlabel('Time (s)')
ylabel('input angle (deg)')
grid on
```