MAE 162A Final Project: *Design of a horizontal-platform mechanism*

Duncan Di Mauro

UID: 805163177

Due: 6/11/2021

1. Design Requirements

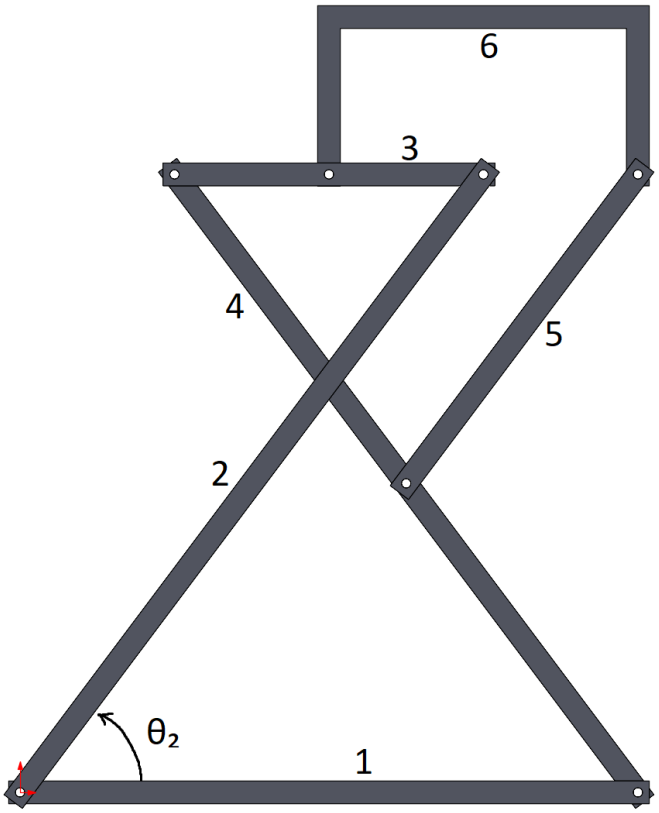

Figure 1: Horizontal-platform mechanism

The goal of this project is to design and simulate a mechanism which provides horizontal motion to a tabletop (link 6) with an input of θ_2. The design requirements are as follows:

1. The tabletop must be able to travel at least 850mm along the x-axis
2. The total vertical translation of the table's center of mass must be < 4mm
3. The total angular deviation of the tabletop must be less than 1° from the horizontal
4. For at least one position, the mechanism should fit in an 850mm x 1050mm area
5. No links must ever rise above the top surface of the tabletop

The main assumptions are as follows:

- All links lie on the same plane and can pass through each other
- Each link has a constant density which is that of aluminum (2,698.9kg/m^3)
- The width and out-of-plane thickness of each link including the tabletop is 30 mm
- The mechanism is operated when gravity is acting in the negative y-direction

2. Mechanism Design

The mechanism that was designed is a Chebyshev linkage, which converts rotational motion to approximate straight-line motion. In this design, link 1 is fixed and can be considered ground. The two vector loops used to analyze the mechanism and describe its links can be seen below:

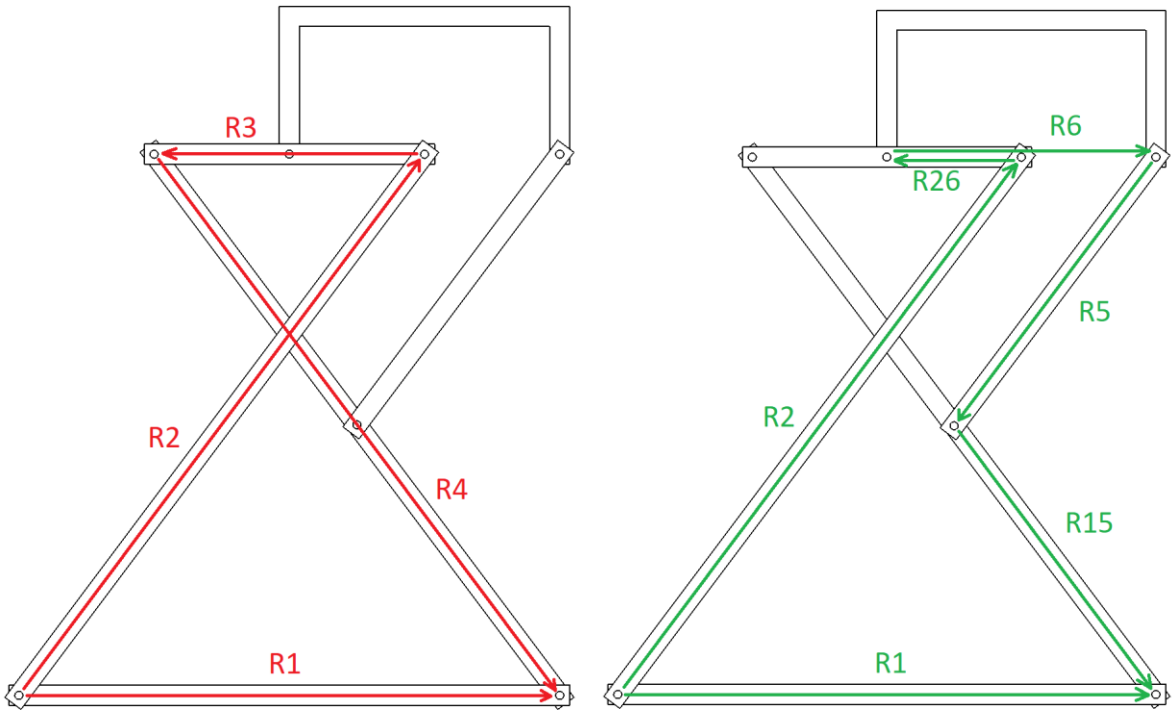

Figure 2: Vector loops used for analysis – loop 1 (left) and loop 2 (right)

The equation for the first vector loop is: R1+ R2 + R3 + R4 – R1 = 0. The unknowns in this loop include θ_3 and θ_4. The equation for the second vector loop is: R2 + R26 + R6 + R5 + R15 – R1 = 0. The unknowns in this loop are θ_5 and θ_6, with the constraints being $\theta_{26} = \theta_3$ and $\theta_{15} = \theta_4$. It is important to note that all vectors used are from one joint (pin) to the other. This means that the total lengths of each link are longer than the lengths of their corresponding vectors. The ratios between links were found using the specific proportions of a Chebyshev linkage along with Chebyshev's Lambda Mechanism and are as follows:

$$R_1 : R_2 : R_3 : R_4 : R_5 : R_6 : R_{26} : R_{15} = 1 : \frac{5}{4} : \frac{1}{2} : \frac{5}{4} : \frac{5}{8} : \frac{1}{2} : \frac{1}{4} : \frac{5}{8}$$

In order to meet the design requirements, the final length of R_1 (which determines all other lengths) was determined to be $R_1 = 805\ mm$. The only measurement which R_1 does not influence is the height of Link 6. This height was instead decided based on the 4th and 5th design requirements. All dimensions are shown in the images on the following pages:

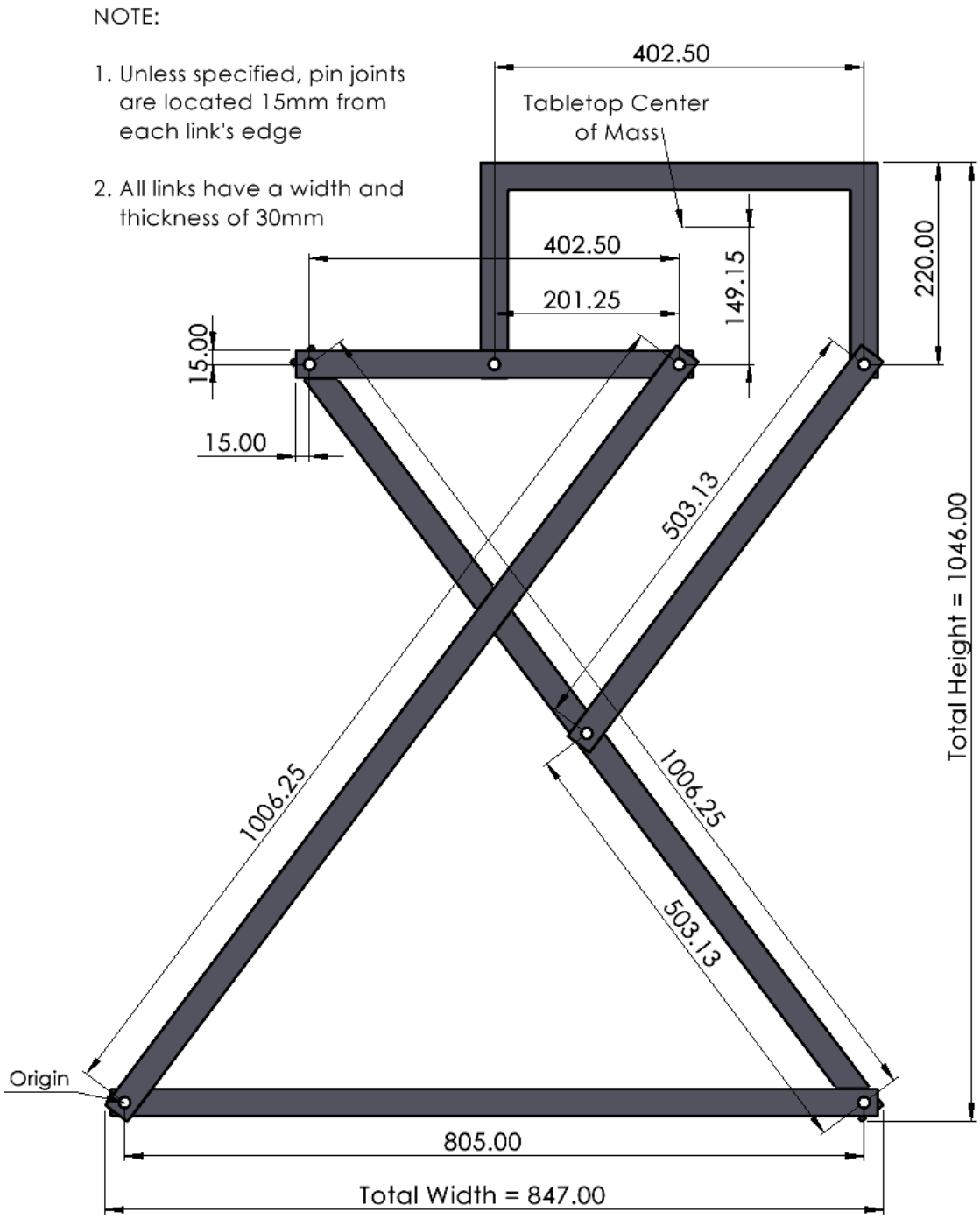

Figure 3: Final Dimensions of Mechanism (all in millimeters)

All measurements were found using SolidWorks after creating a complete design with all the appropriate mates and restrictions. SolidWorks was also used to observe the motion of the mechanism and decide on the height of link 6 such that the 5th design requirement was met.

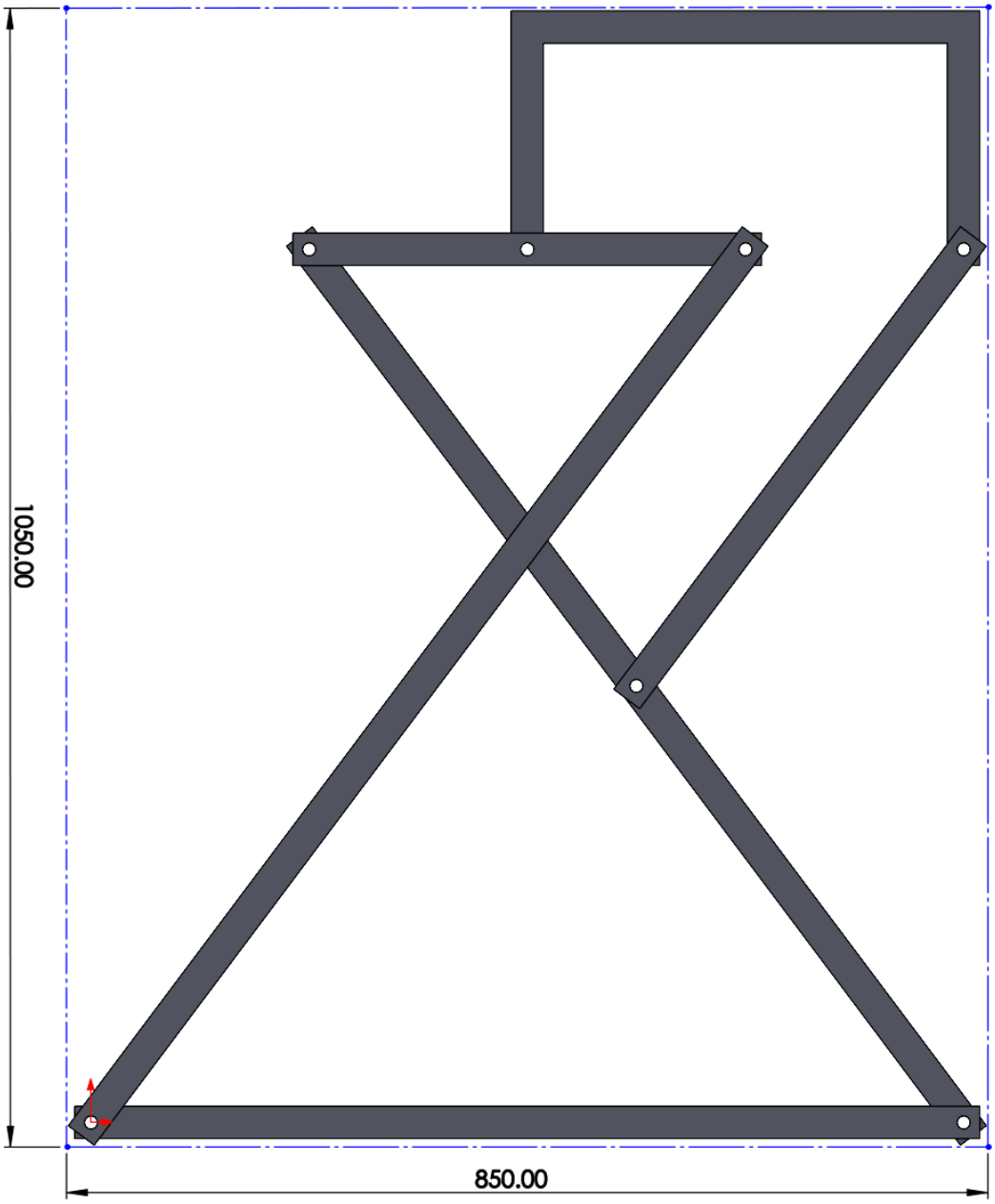

Figure 4: Mechanism meeting the 850 x 1050 mm design requirement

SolidWorks was also used to measure the horizontal and vertical displacement of Link 6 throughout the motion of the mechanism. The minimum and maximum angle of θ_2 were decided based off this initial analysis and were found to be:

$$\theta_{2,min} = 36.88° \quad \text{and} \quad \theta_{2,max} = 96.8°.$$

In order to confirm that the measurements and minimum/maximum angles met all the design requirements throughout the entire motion of the mechanism, analysis was performed in MATLAB.

3. Analysis to Confirm Design Requirements

All graphs in this section were produced through analysis in MATLAB. In order to measure the horizontal and vertical displacement of the tabletop's center of mass throughout the entire motion of the mechanism ($\theta_{2,min}$ to $\theta_{2,max}$), we first need to perform position analysis to understand the complete motion of all links. The Newton-Raphson method was performed on the first vector loop to find the corresponding θ_3 and θ_4 for all θ_2 values, and was then applied to the second vector loop to obtain values for θ_5 and θ_6. Then, center of mass calculations were performed and the position of the tabletop's center of mass, along with its angular deviation, could be described.

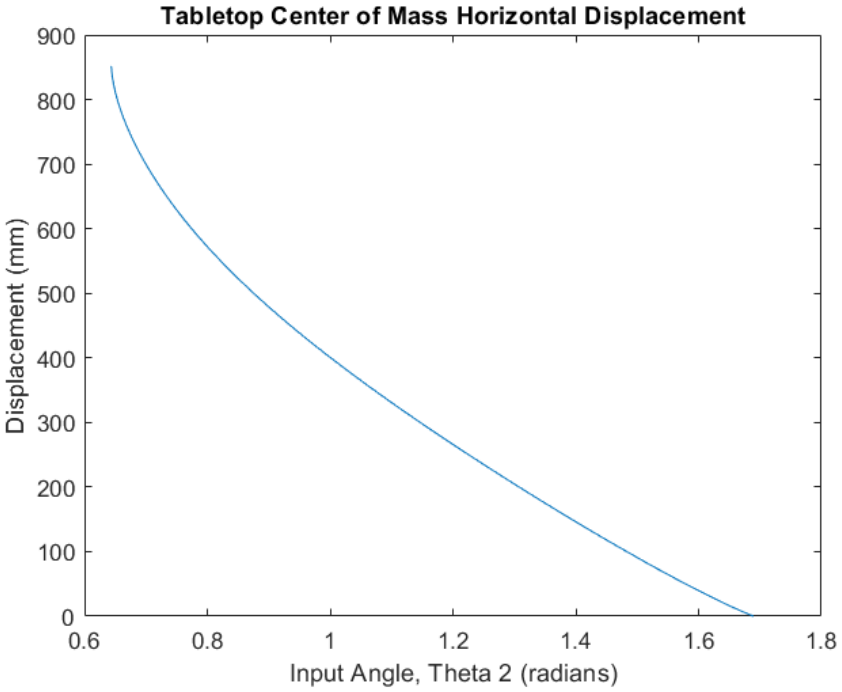

Figure 5: Horizontal displacement of tabletop's center of mass throughout the entire motion of the mechanism. The tabletop starts at x = 852 mm and ends at x = 0 mm.

As shown in the figure above, the 1st design requirement is met as the tabletop travels a total of 852 mm along the x-axis.

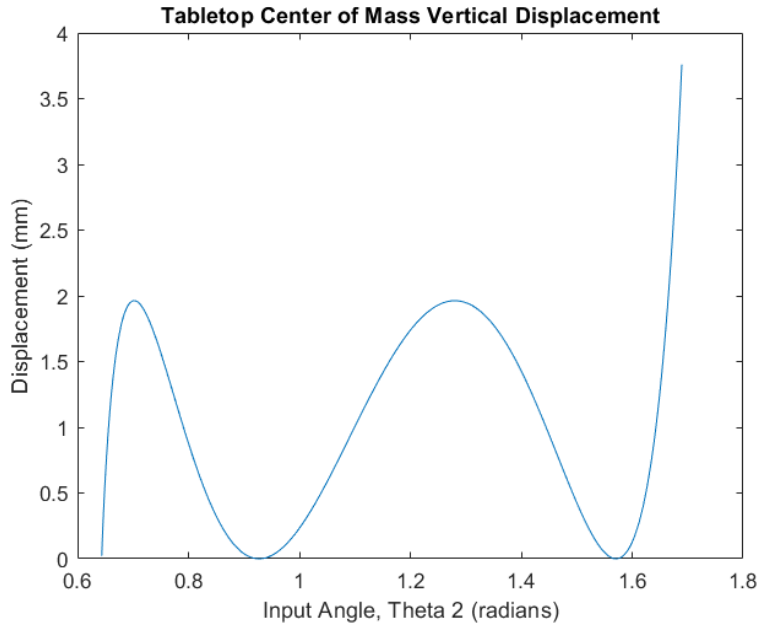

Figure 6: Vertical displacement of tabletop's center of mass throughout the entire motion of the mechanism. The vertical displacement ranges from 0 to 3.75866 mm.

As shown in figure 6 and 7, the 2nd and 3rd design requirement are met. The tabletop has a total vertical translation of 3.75866 mm which is < 4 mm. The data was normalized so that the lowest vertical displacement value would be 0. Additionally, the angular deviation of the tabletop from the horizontal is in the magnitude of 10^{-14} radians, which is much less than 1°.

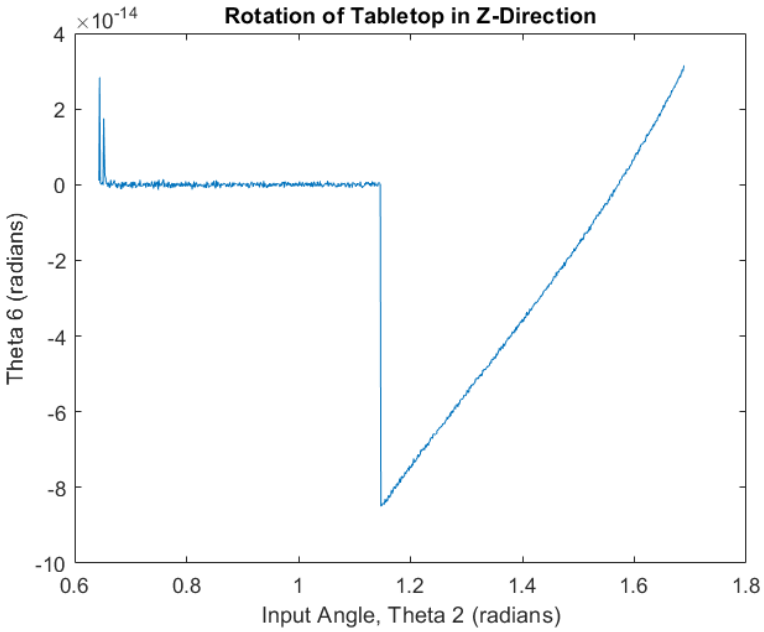

Figure 7: Angular rotation of the tabletop in the z-direction throughout the entire motion of the mechanism. The rotation is in the degree of 10^{-14} radians.

4. Driving Link 2 with a Motor

The final portion of this project involved analyzing the mechanism when link 2 was driven with a motor described by a sinusoidal displacement of:

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

where ω is 1°/sec and t is time. The derivations of this equation provide values for ω_2 and α_2 which were used in calculations. The goal in this portion of the project was to plot the input torque of the motor as a function over time for three full cycles of its oscillating stroke. The equation that was used to find torque was the power equation:

$$P = \tau_2\omega_2 = \frac{dT}{dt} + \frac{dU}{dt} + \frac{dW_{dis}}{dt} = \frac{dT}{dt} + \frac{dU_{gr}}{dt}$$

This was simplified since the value for $\frac{dW_{dis}}{dt}$ = 0 and potential energy is only coming from gravity. The kinetic energy was found with:

$$\frac{dT}{dt} = \sum_{j=2}^{6} A_j \cdot \omega_2\alpha_2 + \sum_{j=2}^{6} B_j\omega_2^3, \text{ with}$$

$$A_j = m_j\left(x'^2_{G_j} + y'^2_{G_j}\right) + I_{G_j}\theta'^2_j, \qquad B_j = m_j\left(x'_{G_j}x''_{G_j} + y'_{G_j}y''_{G_j}\right) + I_{G_j}\theta'_j\theta''_j$$

The gravitational potential energy was found using:

$$\frac{dU_{gr}}{dt} = \sum_{j=2}^{6} m_j g y'_{G_j}\omega_2$$

In order to find the torque, we needed to solve for all the variables used in these equations. Position analysis was performed using the Newton-Raphson method, and first and second kinematic coefficients were found by finding the Jacobian and solving matrix equations. The center of masses and moments of inertia were found in MATLAB.

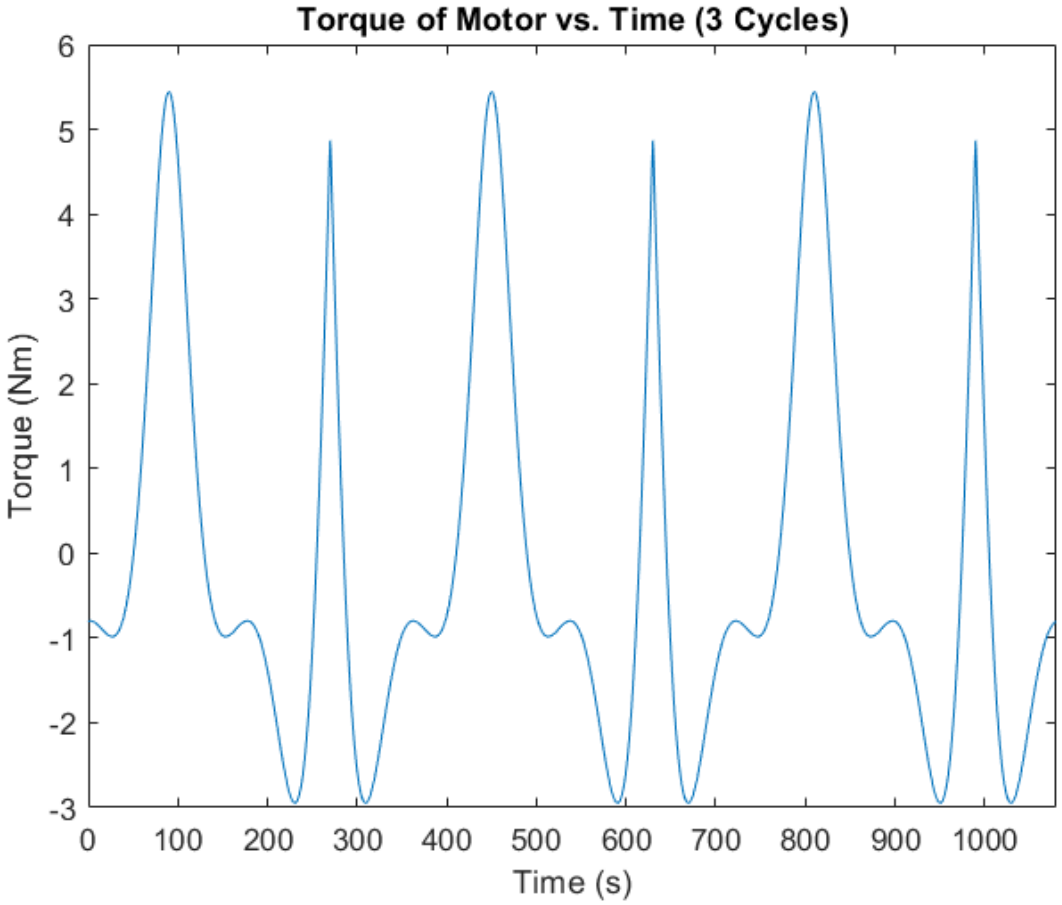

Figure 8: Input torque of the motor as a function of time for three full cycles of its oscillating stroke

As shown in the figure, the torque peaks when the mechanism changes direction, or when $t = 90 + 180n$ seconds for all integer values of n.

Appendix – MATLAB Code

All MATLAB scripts used in this project are shown below. Functions are provided first, and then the main script.

```
function [t3,t4] = pos_L1(r1, r2, r3, r4, t2, formation)
%First Vector Loop Position Analysis - find t3 and t4 for all t2

%Note that "L1" means loop 1

%Preallocating Space
t3 = ones(1, length(t2));
t4 = ones(1, length(t2));

if formation == 1 %without the motor
    t3_0 = 0.5*pi;
    t4_0 = 1.75*pi;
end

if formation == 2 %with the motor
    t3_0 = 3.7526;
    t4_0 = 5.5278;
end

for i = 1:length(t2)
    %set dt3 and dt4 to randomly large number
    dt3 = 100;
    dt4 = 100;
    D = [dt3;dt4];

    %Set up variables for iterations
    n = 0;
    t3_1 = t3_0;
    t4_1 = t4_0;

    while norm(D,1)>10^(-6)

        n = n + 1;
        %Evaluate f1, f2
        f1 = r2*cos(t2(i)) + r3*cos(t3_1) + r4*cos(t4_1) - r1;
        f2 = r2*sin(t2(i)) + r3*sin(t3_1) + r4*sin(t4_1);
        F = [f1;
             f2];

        %Evaluate Jacobian

        J = [-r3*sin(t3_1), -r4*sin(t4_1);
              r3*cos(t3_1), r4*cos(t4_1)];

        %return dt3, dt3
```

```
        D = -J\F;
        dt3 = D(1);
        dt4 = D(2);

        %update t3_1, t4_1
        t3_1 = t3_1 + dt3;
        t4_1 = t4_1 + dt4;

    end

    %Extract t3_1 and t4_1 for solution
    t3(i) = t3_1;
    t4(i) = t4_1;

    %Update guess for next iteration
    t3_0 = t3_1;
    t4_0 = t4_1;
end

end


function [t5,t6] = pos_L2(r2, r26, r6, r5, r15, r1, t2, t3, t4,
formation)
%Second Vector Loop Position Analysis - find t5 and t6 for all t2

%Note that "L2" means loop 2

%Preallocating Space
t5 = ones(1, length(t2));
t6 = ones(1, length(t2));

if formation == 1 %without the motor
    t5_0 = 1.21*pi;
    t6_0 = 0;
end

if formation == 2 %with the motor
    t5_0 = 4.30817;
    t6_0 = 0;
end

for i = 1:length(t2)
    %set dt5 and dt6 to randomly large number
    dt5 = 100;
    dt6 = 100;
    D = [dt5;dt6];

    %Set up variables for iterations
    n = 0;
    t5_1 = t5_0;
    t6_1 = t6_0;
```

```
    while norm(D,1)>10^(-6)

        n = n + 1;
        %Evaluate f1, f2
        f1 = r2*cos(t2(i)) + r26*cos(t3(i)) + r6*cos(t6_1) +
r5*cos(t5_1) + r15*cos(t4(i)) - r1;
        f2 = r2*sin(t2(i)) + r26*sin(t3(i)) + r6*sin(t6_1) +
r5*sin(t5_1) + r15*sin(t4(i));
        F = [f1;
             f2];

        %Evaluate Jacobian

        J = [-r5*sin(t5_1), -r6*sin(t6_1);
              r5*cos(t5_1), r6*cos(t6_1)];

        %return dt5, dt6
        D = -J\F;
        dt5 = D(1);
        dt6 = D(2);

        %update t5_1, t6_1
        t5_1 = t5_1 + dt5;
        t6_1 = t6_1 + dt6;

    end

    %Extract t5_1 and t6_1 for solution
    t5(i) = t5_1;
    t6(i) = t6_1;

    %Update guess for next iteration
    t5_0 = t5_1;
    t6_0 = t6_1;
end

end


%MAE 162A Final Project
%Duncan Di Mauro
%UID: 805163177
%Due: 6/11/2021 10:00 AM

clear all; close all; clc;

%% Establishing Constants

r1 = 0.805; %meters
r2 = r1*(5/4);
r3 = r1*(1/2);
```

```
r4 = r1*(5/4);
r5 = r1*(5/8);
r6 = r1*(1/2);
r26 = r1*(1/4);
r15 = r1*(5/8);

%Note that t2 means theta2. Same with other thetas.
t2_min_degrees = 36.88;
t2_min = t2_min_degrees*(pi/180);
t2_max_degrees = 96.8;
t2_max = t2_max_degrees*(pi/180);

%Making the t2 input array, goes from the minimum to maximum in
radians
t2 = linspace(t2_min, t2_max, 1000);


%% Position Analysis - find t3, t4, t5, and t6 for all t2

[t3,t4] = pos_L1(r1, r2, r3, r4, t2, 1);
[t5,t6] = pos_L2(r2, r26, r6, r5, r15, r1, t2, t3, t4, 1);

figure(2);
plot(t2, t6)
title('Rotation of Tabletop in Z-Direction');
xlabel('Input Angle, Theta 2 (radians)');
ylabel('Theta 6 (radians)');


%% Link 6 Center of Mass Displacement Calculations

%Finding center of mass locations relative to the tail of r6
r6_cm_x = r6*(1/2);
r6_cm_y = 0.14915; %in meters, found from SolidWorks
r6_cm_mag = sqrt(r6_cm_x^2 + r6_cm_y^2);

t_r6_cm = atan(r6_cm_y/r6_cm_x); % = radians counterclockwise from r6

%Finding center of mass locations relative to the origin of the system
xg6_1 = r2*cos(t2) + r26*cos(t3) + r6_cm_mag*cos(t6 + t_r6_cm);
yg6_1 = r2*sin(t2) + r26*sin(t3) + r6_cm_mag*sin(t6 + t_r6_cm);

%Finding the ranges of xg6_1 and yg6_1
range_xg6_1 = max(xg6_1) - min(xg6_1)
range_yg6_1 = max(yg6_1) - min(yg6_1)

% Making the displacement vectors have a minimum value of 0
xg6_1 = xg6_1 - min(xg6_1);
yg6_1 = yg6_1 - min(yg6_1);

figure(3);
plot(t2,xg6_1*1000);
```

```
title('Tabletop Center of Mass Horizontal Displacement');
xlabel('Input Angle, Theta 2 (radians)');
ylabel('Displacement (mm)');

figure(4);
plot(t2,yg6_1*1000);
title('Tabletop Center of Mass Vertical Displacement');
xlabel('Input Angle, Theta 2 (radians)');
ylabel('Displacement (mm)');


%% Motor Driving Link 2 Setup and Position Analysis

w = 1*(pi/180); %degrees/sec

%Creating time array for 3 full cycles
%seconds needed = 360*3 = 180
t = linspace(0, 1080, 1081);

%Modeling theta 2 based on the motor (t2m means theta 2 motor)
t2m = (t2_max - t2_min)*0.5*sin(w*t) + (t2_max + t2_min)*0.5;

%Finding 1st and 2nd derivatives of t2m (w2 and a2)
w2 = (1/2)*w*(t2_max - t2_min)*cos(w*t);
a2 = -(1/2)*w^2*(t2_max - t2_min)*sin(w*t);

%Finding angles when motor is applied
[t3m, t4m] = pos_L1(r1,r2,r3,r4,t2m,2);
[t5m, t6m] = pos_L2(r2,r26,r6,r5,r15,r1,t2m,t3m,t4m,2);

%% Finding first order coefficients

% First Order Kinematic Coefficient Analysis, Loop 1
% Find t3p and t4p (theta 3 prime and theta 4 prime)

%Preallocating Space
t3mp = ones(1,length(t2m));
t4mp = ones(1,length(t2m));

for i = 1:length(t2m)

    J = [-r3*sin(t3m(i)), -r4*sin(t4m(i));
          r3*cos(t3m(i)), r4*cos(t4m(i))];

    RightSide = [r2*sin(t2m(i));-r2*cos(t2m(i))];

    Solution = J\RightSide;

    t3mp(i) = Solution(1,1);
    t4mp(i) = Solution(2,1);
end
```

```
% First Order Kinematic Coefficient Analysis, Loop 2
% Find t5p and t6p (theta 5 prime and theta 6 prime)

%Preallocating Space
t5mp = ones(1,length(t2m));
t6mp = ones(1,length(t2m));

for i = 1:length(t2m)

    J = [-r5*sin(t5m(i)), -r6*sin(t6m(i));
          r5*cos(t5m(i)), r6*cos(t6m(i))];

    RightSide =
[r2*sin(t2m(i))+r26*sin(t3m(i))*t3mp(i)+r15*sin(t4m(i))*t4mp(i);
                -r2*cos(t2m(i))-r26*cos(t3m(i))*t3mp(i)-
r15*cos(t4m(i))*t4mp(i)];

    Solution = J\RightSide;

    t5mp(i) = Solution(1,1);
    t6mp(i) = Solution(2,1);
end

%% Finding second order coefficients

% Second Order Kinematic Coefficient Analysis, Loop 1 - find t3pp and
t4pp
% (theta 3 double prime theta 4 double prime)

%Preallocating Space
t3mpp = ones(1,length(t2m));
t4mpp = ones(1,length(t2m));

for i = 1:length(t2m)

    J = [-r3*sin(t3m(i)), -r4*sin(t4m(i));
          r3*cos(t3m(i)), r4*cos(t4m(i))];

    RightSide =
[r2*cos(t2m(i))+r3*cos(t3m(i))*t3mp(i)^2+r4*cos(t4m(i))*t4mp(i)^2;

r2*sin(t2m(i))+r3*sin(t3m(i))*t3mp(i)^2+r4*sin(t4m(i))*t4mp(i)^2];

    Solution = J\RightSide;

    t3mpp(i) = Solution(1,1);
    t4mpp(i) = Solution(2,1);
end

% Second Order Kinematic Coefficient Analysis, Loop 2 - find t5pp and
t6pp
% (theta 5 double prime theta 6 double prime)
```

```
%Preallocating Space
t5mpp = ones(1,length(t2m));
t6mpp = ones(1,length(t2m));

for i = 1:length(t2m)

    J = [-r5*sin(t5m(i)), -r6*sin(t6m(i));
          r5*cos(t5m(i)), r6*cos(t6m(i))];

    RightSide =
[r2*cos(t2m(i))+r26*sin(t3m(i))*t3mpp(i)+r26*cos(t3m(i))*t3mp(i)^2+r6*
cos(t6m(i))*t6mp(i)^2+...

r5*cos(t5m(i))*t5mp(i)^2+r15*sin(t4m(i))*t4mpp(i)+r15*cos(t4m(i))*t4mp
(i)^2;
                  r2*sin(t2m(i))-
r26*cos(t3m(i))*t3mpp(i)+r26*sin(t3m(i))*t3mp(i)^2+r6*sin(t6m(i))*t6mp
(i)^2+...
                  r5*sin(t5m(i))*t5mp(i)^2-
r15*cos(t4m(i))*t4mpp(i)+r15*sin(t4m(i))*t4mp(i)^2];

    Solution = J\RightSide;

    t5mpp(i) = Solution(1,1);
    t6mpp(i) = Solution(2,1);
end

%% Center of mass location, velocity, and acceleration of each link

%Finding center of mass for each link
xg2 = 0.5*r2*cos(t2m);
yg2 = 0.5*r2*sin(t2m);

xg3 = r2*cos(t2m) + 0.5*r3*cos(t3m);
yg3 = r2*sin(t2m) + 0.5*r3*sin(t3m);

xg4 = r2*cos(t2m) + r3*cos(t3m) + 0.5*r4*cos(t4m);
yg4 = r2*sin(t2m) + r3*sin(t3m) + 0.5*r4*sin(t4m);

xg5 = xg3 + r6*cos(t6m) + 0.5*r5*cos(t5m);
yg5 = yg3 + r6*sin(t6m) + 0.5*r5*sin(t5m);

xg6 = r2*cos(t2m) + r26*cos(t3m) + r6_cm_mag*cos(t_r6_cm + t6m);
yg6 = r2*sin(t2m) + r26*sin(t3m) + r6_cm_mag*sin(t_r6_cm + t6m);

% Finding velocity for each center of mass
xg2p = -0.5*r2*sin(t2m);
yg2p = 0.5*r2*cos(t2m);

xg3p = -r2*sin(t2m) - 0.5*r3*sin(t3m).*t3mp;
yg3p = r2*cos(t2m) + 0.5*r3*cos(t3m).*t3mp;
```

```
xg4p = -r2*sin(t2m) - r3*cos(t3m).*t3mp - 0.5*r4*sin(t4m).*t4mp;
yg4p = r2*cos(t2m) + r3*cos(t3m).*t3mp + 0.5*r4*cos(t4m).*t4mp;

xg5p = xg3p - r6*sin(t6m).*t6mp - 0.5*r5*sin(t5m).*t5mp;
yg5p = yg3p + r6*cos(t6m).*t6mp + 0.5*r5*cos(t5m).*t5mp;

xg6p = -r2*sin(t2m) - r26*sin(t3m).*t3mp - r6_cm_mag*sin(t_r6_cm +
t6m).*t6mp;
yg6p = r2*cos(t2m) + r26*cos(t3m).*t3mp + r6_cm_mag*cos(t_r6_cm +
t6m).*t6mp;

% Finding accelerations for each center of mass
xg2pp = -0.5*r2*cos(t2m);
yg2pp = -0.5*r2*sin(t2m);

xg3pp = -r2*cos(t2m) - 0.5*r3*cos(t3m).*t3mp.^2 -
0.5*r3*sin(t3m).*t3mpp;
yg3pp = -r2*sin(t2m) - 0.5*r3*sin(t3m).*t3mp.^2 +
0.5*r3*cos(t3m).*t3mpp;

xg4pp = -r2*cos(t2m) - r3*cos(t3m).*t3mp.^2 - r3*sin(t3m).*t3mpp -
0.5*r4*cos(t4m).*t4mp.^2 - 0.5*r4*sin(t4m).*t4mpp;
yg4pp = -r2*sin(t2m) - r3*sin(t3m).*t3mp.^2 + r3*cos(t3m).*t3mpp -
0.5*r4*sin(t4m).*t4mp.^2 + 0.5*r4*cos(t4m).*t4mpp;

xg5pp = xg3pp - r6*cos(t6m).*t6mp.^2 - r6*sin(t6m).*t6mpp -
0.5*r5*cos(t5m).*t5mp.^2 - 0.5*r5*sin(t5m).*t5mpp;
yg5pp = yg3pp - r6*sin(t6m).*t6mp.^2 + r6*cos(t6m).*t6mpp -
0.5*r5*sin(t5m).*t5mp.^2 + 0.5*r5*cos(t5m).*t5mpp;

xg6pp = -r2*cos(t2m) - r26*cos(t3m).*t3mp.^2 - r26*sin(t3m).*t3mpp -
r6_cm_mag*cos(t_r6_cm + t6m).*t6mp.^2 - r6_cm_mag*sin(t_r6_cm +
t6m).*t6mpp;
yg6pp = -r2*sin(t2m) - r26*sin(t3m).*t3mp.^2 + r26*cos(t3m).*t3mpp -
r6_cm_mag*sin(t_r6_cm + t6m).*t6mp.^2  + r6_cm_mag*cos(t_r6_cm +
t6m).*t6mpp;

%Calculating mass of each link
rho = 2698.9; %Density of aluminum in kg/m^3
m1 = r1*rho*0.03^2;
m2 = r2*rho*0.03^2;
m3 = r3*rho*0.03^2;
m4 = r4*rho*0.03^2;
m5 = r5*rho*0.03^2;
m6 = r6*rho*0.03^2;


% Potential Energy Calculation:
Ugr = 9.81*w2.*(m2*yg2p + m3*yg3p +m4*yg4p + m5*yg5p + m6*yg6p);
```

```
%Kinetic Energy Calculations
I1 = m1/12*(0.03^2 + r1^2);
Ig2 = m2/12*(0.03^2 + r2^2);
Ig3 = m3/12*(0.03^2 + r3^2);
Ig4 = m4/12*(0.03^2 + r4^2);
Ig5 = m5/12*(0.03^2 + r5^2);
Ig6 = m6/12*(0.03^2 + r6^2);

A = m2*(xg2p.^2+yg2p.^2) + Ig2 + m3*(xg3p.^2+yg3p.^2) + Ig3*t3mp.^2 +
...
    m4*(xg4p.^2+yg4p.^2) + Ig4*t4mp.^2 + m5*(xg5p.^2+yg5p.^2) + ...
    Ig5*t5mp.^2 + m6*(xg6p.^2 + yg6p.^2) + Ig6*t6mp.^2;

B = m2*(xg2p.*xg2pp + yg2p.*yg2pp) + m3*(xg3p.*xg3pp + yg3p.*yg3pp) +
...
    Ig3*t3mp.*t3mpp + m4*(xg4p.*xg4pp + yg4p.*yg4pp) + Ig4*t4mp.*t4mpp
+ ...
    m5*(xg5p.*xg5pp + yg5p.*yg5pp) + Ig5*t5mp.*t5mpp +...
    m6*(xg6p.*xg6pp + yg6p.*yg6pp) + Ig6*t6mp.*t6mpp;

KE = A.*w2.*a2 + B.*(w2.^3);

%Finding Torque
torque = (KE + Ugr)./w2;
figure(5);

plot(t,torque);
title('Torque of Motor vs. Time (3 Cycles)');
axis([0,1080,-3,6])
xlabel('Time (s)');
ylabel('Torque (Nm)');
```