AME 40423 - Mechanisms and Machines
Drive System Trade Study

Abstract

The purpose of this Trade Study is to analyze a lifting four-bar mechanism in order to select a combination of gearbox and motor to be used for the mechanism's drive system, such that the system will meet all state variables and constraints set forth for the design as best as possible. In order to do this, the mechanism's kinematic characteristic were calculated such that the system was treated as an Inverse Dynamics Problem for a static case, in order to find the minimum torque needed to drive the system for the static case. Using the torque requirements, a motor and gearbox combination (BaneBot RS 395 motor and BaneBot P60 gearbox with 132:1 ratio) was then selected to create a gearmotor to use for the drive system. The torque-speed curve was of the gearmotor was then calculated, and the inertial effects of the motor's windings were factored into dynamic subsequent calculations. Manipulating the power equation, the angular acceleration of link 2 of the mechanism was then calculated, which was subsequently used with Euler's method to create a dynamic simulation of the machine's operation as a function of time. The results of the dynamic simulation were used to determine if the system satisfied the state variable constraint of the design. The original drive system satisfied two of the given constraints. The system was optimized in an attempt to satisfy the state variables with priority according to the order they were given. A drive system was created using a different gearmotor configuration (BaneBots RS 550 Motor and BaneBots P60 gearbox with 672:1 ratio) that was able to satisfy all but one of the state variables.

Engineering Analysis

The machine being studied is a four bar mechanism driven by a gearmotor, which lifts a steel block through an intermediate position and then continues to cycle for for a desired duration. The steel block is given to be an order of magnitude more massive than any other moving element in the machine, which thus means that the only inertial effects experienced by the system will be due by the steel block and the copper windings of the gearmotor. The gearmotor is a combination of a

Direct Current (DC) 12 Volt motor and a gearbox. There is no load torque acting on the system, thus the only toque acting on the system is the gearmotor's driving torque. For a full, detail, and clean statement of the Trade Study, see the attached document in the Appendix. The design variables for the trade study are

- *Design Variable 1*: Motor driving link 2
- *Design Variable 2*: Transmission driving link 2

The constraints on the design variables for the trade study are as follows

- The motor and transmission must come together as a single unit (i.e. a gearmotor) in order to ensure compatibility
- System power is 12 volts direct current

The state variables of the trade study are

- *State Variable 1*: Time required to move the block to the intermediate position.
- *State Variable 2*: Time required to reach steady-state operation.
- *State Variable 3*: Speed variation of the input at stead-state.
- *State Variable 4*: Cost of gearmotor.
- *State Variable 5*: Weight of gearmotor.

The constraints on these state variables are as follows

- Block must move to the intermediate position ($\theta_2 = -58.2°$) within 0.50 seconds.
- Machine should reach steady-state operation within 2 seconds.
- Coefficient of fluctuation of the input speed should be less than 10%.
- Gearmotor can cost no motre than \$60.

- Drive system can weigh no more than $2\ \mathrm{lb_f}$.

The state variables prioritized into a measure of merit are in the following order. To

- *State Variable 1*
- *State Variable 2*
- *State Variable 3*
- *State Variable 4*
- *State Variable 5*

In order to build the dynamic simulation, first a function was created that solved the position equation of the vector loop method using Newton's method, following the procedure outlined in Section 2.4 of the textbook [1]. The resulting solution of the position problem given by this function can be found in Figure 3 of Appendix A. The MATLAB code for this function can be found in the function FourBarNewton.m, which is listed in Appendix C.

Next, the results of the the position problem were used to calculate the Basic first and second order Kinematic Coefficients for links 3 and 4, with link 2 as the input. These kinematic coefficients were calculated according to Equation (4.9) on page 161 of the textbook [1]. The results of this can be found in Figures 4 and 5 of Appendix A. The MATLAB code for the function used to calculate these kinematic coefficients can be found in the function BasicKCs.m, which is listed in Appendix C.

The kinematic coefficients of points of interest were then calculated, in this case point G_3, the center of mass of the steel block attached to link 3, was the only point of interest. The x and y components of the first and second order coefficients of the vector locating G_3 were calculated as outlined by the procedure in Example 4.6 of the textbook [1]. The kinematic coefficient of elevation for the steel block, f_e is defined by Equation (7.9) on page 294 of the textbook [1], where θ_e is defined in Figure 2 of the problem statement given in Appendix D. The results are plotted in Figures 6, 7, and 8 of Appedix A. The MATLAB code for the function used to calculate these kinematic coefficients can be found in the function G3_KCs.m, which is listed in Appendix C.

Using the inertial properties of the system and the kinematic coefficients calculated above, the (ΣA) and (ΣB) of the system was calculated for each link according to Equation (7.5) on page 292 of the textbook [1]. Since the steel block was stated to be an order of magnitude larger than any other link in the mechanism, all links aside from link 3 were assumed to be mass-less, and thus have negligible As and Bs. Later, once a motor was decided, the copper windings of the motor would be incorporated, as they would have enough mass to have inertial effects on the system. The results are plotted in Figure 9 of Appendix A. The MATLAB code for the function used to calculate these kinematic coefficients can be found in the function sum_As_sum_Bs.m, which is listed in Appendix C.

The torque, T_2 required for static equilibrium was then calculated by using the power equation such that power is defined as

$$P = \overline{T}_2 \cdot \overline{\omega}_2 \tag{1}$$

Using this definition plus the definitions for power, change in kinetic energy, and change in potential energy due to elevation given by equations (7.2), (7.6), and (7.10) respectively. It is then assumed that there is no friction losses present, this gives the equation

$$T_2 \cdot \dot{\theta}_2 = (\Sigma A)\dot{\theta}_2\ddot{\theta}_2 + (\Sigma B)\dot{\theta}_2^3 + mgf_e\dot{\theta}_2 \tag{2}$$

Equation 2 was then solved for T_2 and solved for the static case, such that $\dot{\theta}_2$ and $\ddot{\theta}_2$ go to zero. This was then solved such that T_2 was negative for the initial lift. The resulting plot of the torque T_2 required for static equilibrium for one full revolution and for initial lift from the starting to intermediate positions can be found in Figures 10 and 11 in Appendix A respectively. The MATLAB code for the function used to calculate these the torque T_2 required for static equilibrium can be found in the function calc_T_2.m in Appendix C. These results were then used to estimate the minimum torque that would be required to lift the system. With a mimimum torque, a motor-gearbox combination were selected to make up the gearmotor drive system. The BaneBots RS 395 motor and P60 Gearbox with a 132:1 ration were selected. Since, the motor is a DC motor, the torque-speed curve of the gearmotor follows a linear behavior given by

$$T_{gm} = R \cdot T_{stall}\left(1 - \frac{\dot{\theta}_2 R}{\omega_{max}}\right) \tag{3}$$

Where R is the ratio of the gearbox used, T_{stall} and ω_{max} are the stall torque and maximum speed of the motor used, $\dot{\theta}_2$ is the given speed of the output, and T_{gm} is the resulting torque of the output. The attributes of the motor and gearbox were given by BaneBots [2]. The resulting plot of the torque-speed curve of the gearmotor can be found in Figure 12 of Appendix A, and the MATLAB code for the function used to calculate the torque-speed curve of the motor can be found in gearmotor_output.m in Appendix C. In order to finally produce a dynamic simulation, the power equation expression found in Equation 2 was solved for $\ddot{\theta}_2$ such that

$$\ddot{\theta}_2 = \frac{T_2 - (\Sigma B)\dot{\theta}_2^2 - mgf_e}{(\Sigma A)} \tag{4}$$

With this, an euler's method code was used wih a time step Δt such that

$$\Delta\theta_2 = \dot{\theta}_2\Delta t + \frac{1}{2}\ddot{\theta}_2\Delta t^2 \qquad \theta_2 = \theta_2 + \Delta\theta_2 \tag{5}$$

$$\Delta\dot{\theta}_2 = \ddot{\theta}_2\Delta t \qquad \dot{\theta}_2 = \dot{\theta}_2 + \Delta\dot{\theta}_2 \tag{6}$$

Estimates the θ_2 and $\dot{\theta}_2$ for that time step, which are then used with Equation (4) to estimate the $\ddot{\theta}_2$ for the next time step.

Results

The result of the dynamic simulation achieved using the steps outlined above were then used to determine if the design variables satisfied the above listed state variable constraints. In order to determine whether the chosen drive system satisfied the constraint on State Variable 1, a plot of θ_2 as a function of time was plotted in Figure 1 below.

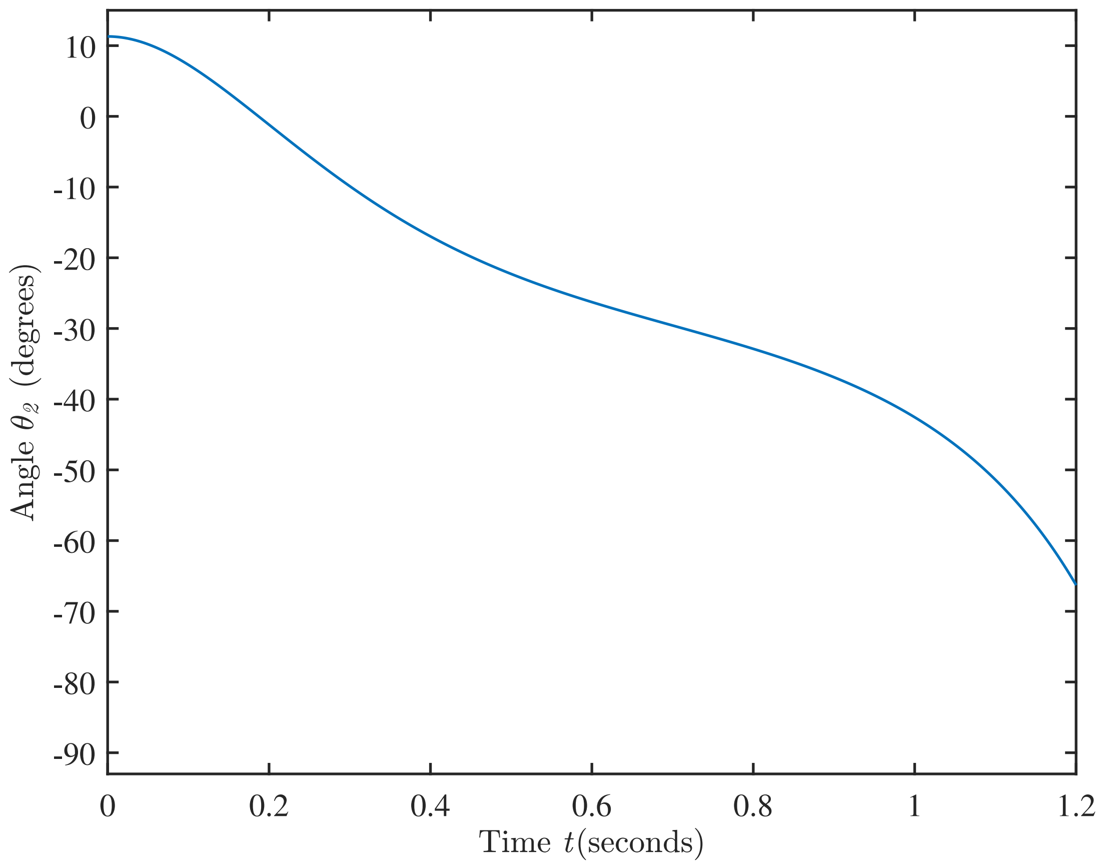

Figure 1: Plot of angle θ_2 as a function of time t for startup of mechanism.

By inspecting the plot, the time it takes for the block to move to the intermediate position is $t = 1.558$ seconds, and thus state variable 1's constraint is not satisfied. A plot of $\dot{\theta}_2$ as a function of time was then plotted in order to determine State Variables 2 and 3. This plot can be seen in Figure 2 below.

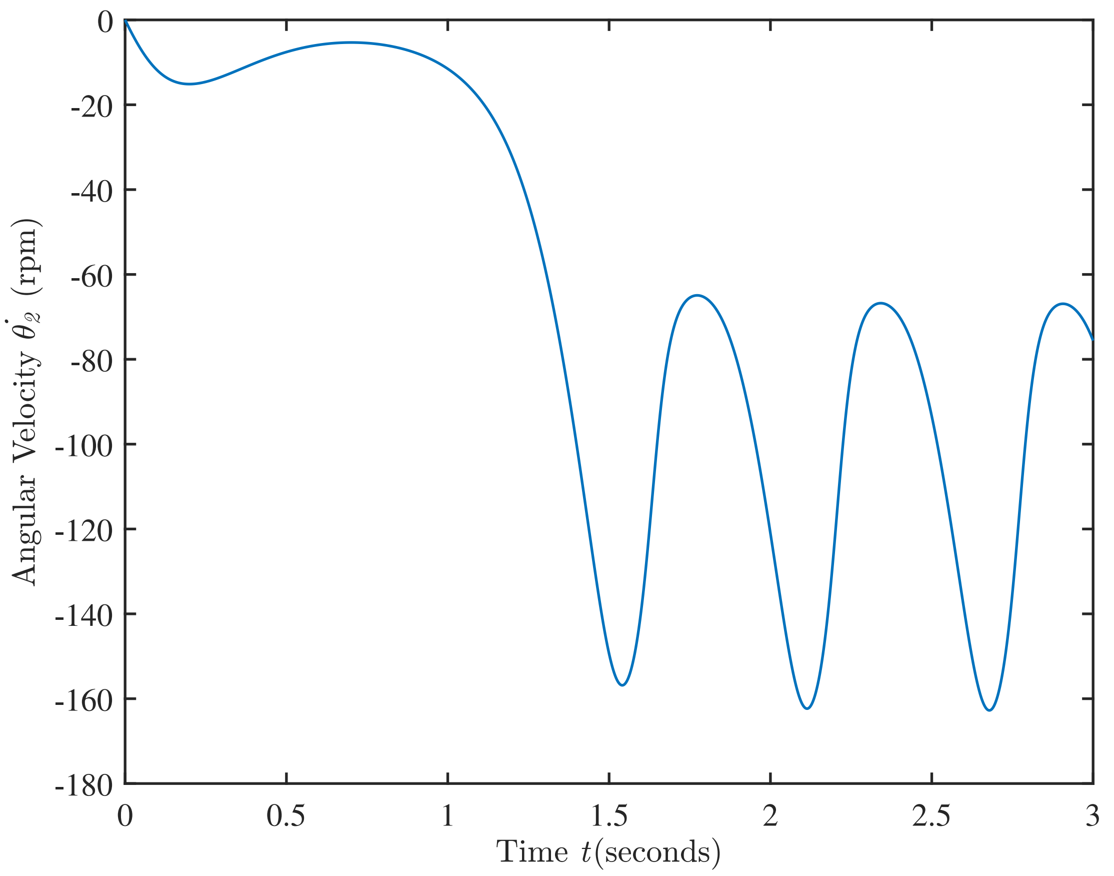

Figure 2: Plot of angular velocity $\dot{\theta}_2$ as a function of time t from start up to steady state of mechanism.

By inspection, the machine reaches steady state after approximately $t = 1.6$ seconds, thus the constraint for state variable 2 is satisfied. However, the coefficient of fluctuation, which is calculated in the code in Appendix C, is approximately $C_f = 85.7\%$, which does not satisfy the constraint on state variable 3. State variables that correspond to vendor specifications are the weight of the drive system and cost of the drive system. The overall cost of the drive system comes out to be $\$69.75$, which means that the constraint on state variable 4 is not satisfied. The weight of the drive system is $0.8125\ \mathrm{lb_f}$, which satisfies the constraint on state variable 5. The design variables are interconnected, and thus both design variables have effects on all state variables. In an attempt to improve on the design, a new motor-gearbox pair were selected. The

BaneBots RS 550 motor and P60 Gearbox with a ratio of $672 : 1$ were selected. This drive system was able to satisfy all state variable constraints except for state variable 4, which is almost impossible to satisfy. This drive system was able to bring the block to the intermediate position in $t = 0.4752$ seconds, was able to reach steady state operation in $t = 1$ second, had a coefficient of fluctuation of $C_f = 7.54\%$, had a cost of $\$85.2$, and a weight of $1.1938\ lb_f$. The plots showing the performance of this drive system can be found in Figures 13 and 14 in Appendix B.

References:

[1] Stanišić, Michael M. *Mechanisms and Machines: Kinematics, Dynamics, and Synthesis*. Cengage Learning, Stamford (2014).
[2] BaneBots LLC, 2018, “BaneBots.” from `http://www.banebots.com`.

Appendix A

Deliverable 1

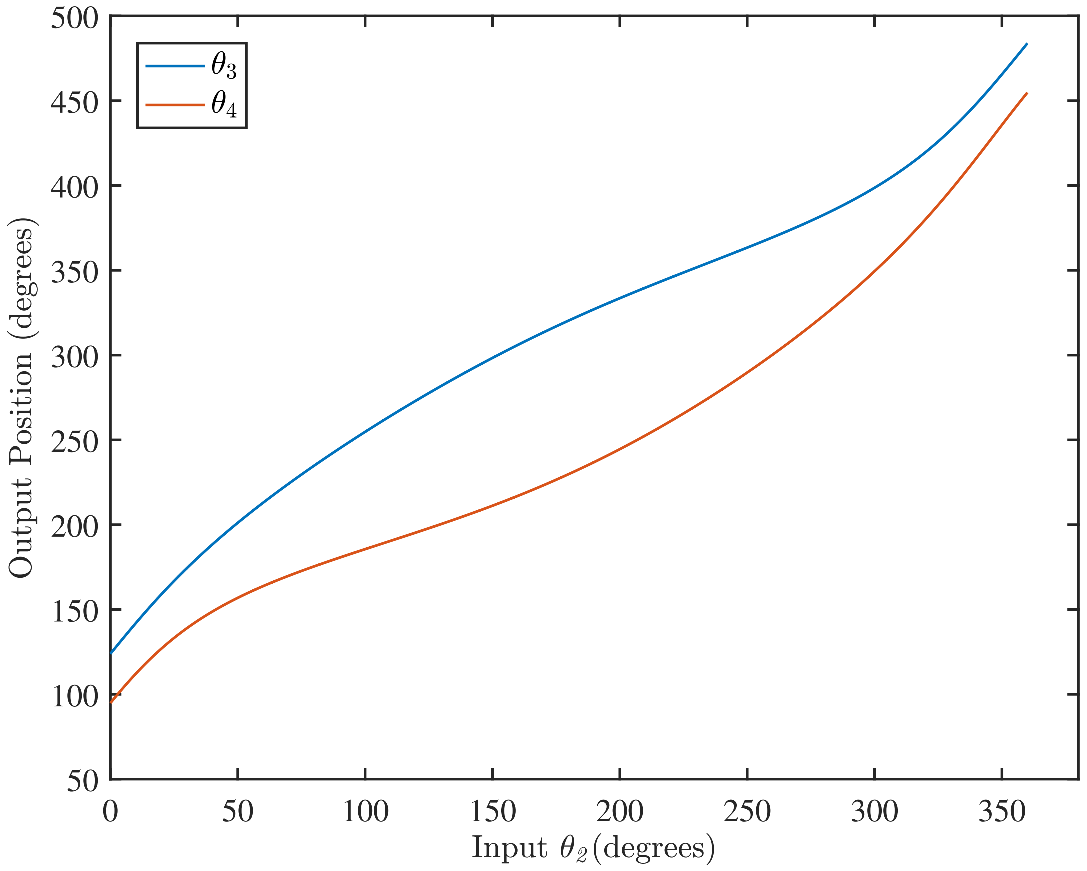

Figure 3: Plot of resulting output angles θ_3 and θ_4 as a function of the given input angle θ_2 solved through use of function that executes Newton's method.

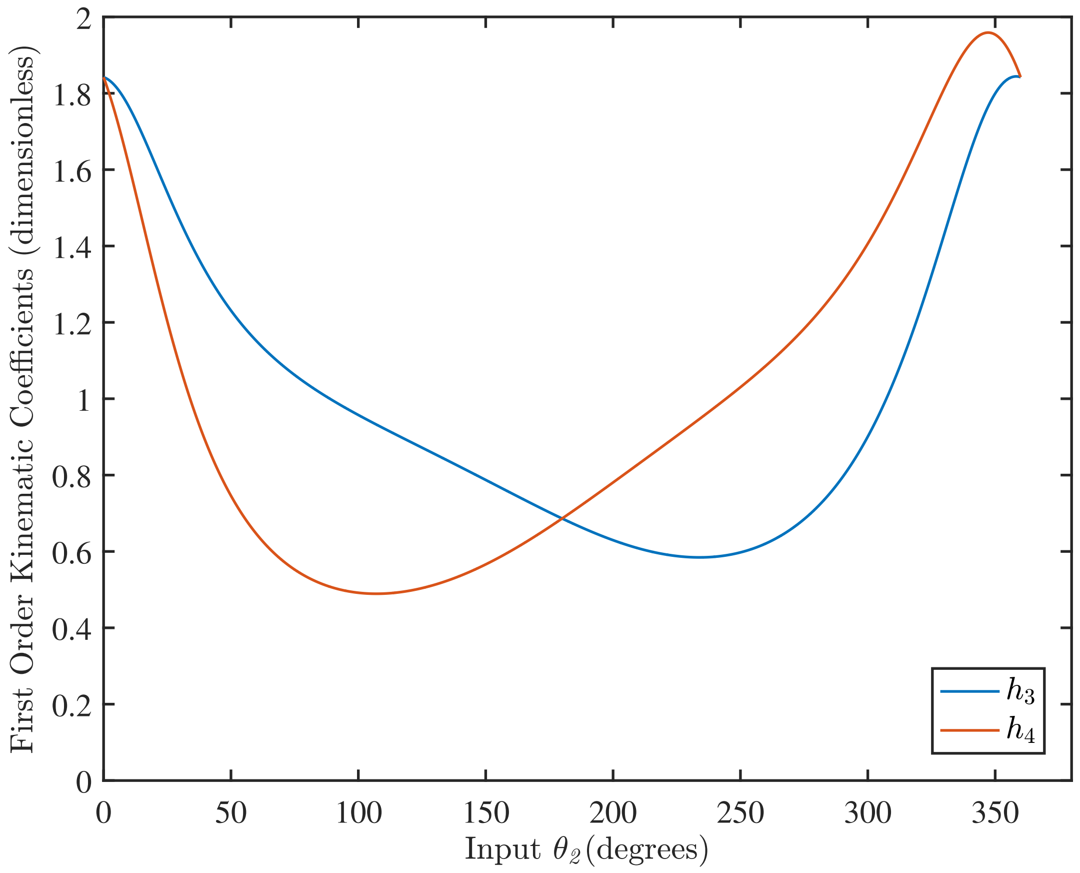

Figure 4: Plot of first order Kinematic Coefficients of links 3 and 4, h_3 and h_4 respectively, as a function of the given input angle θ_2.

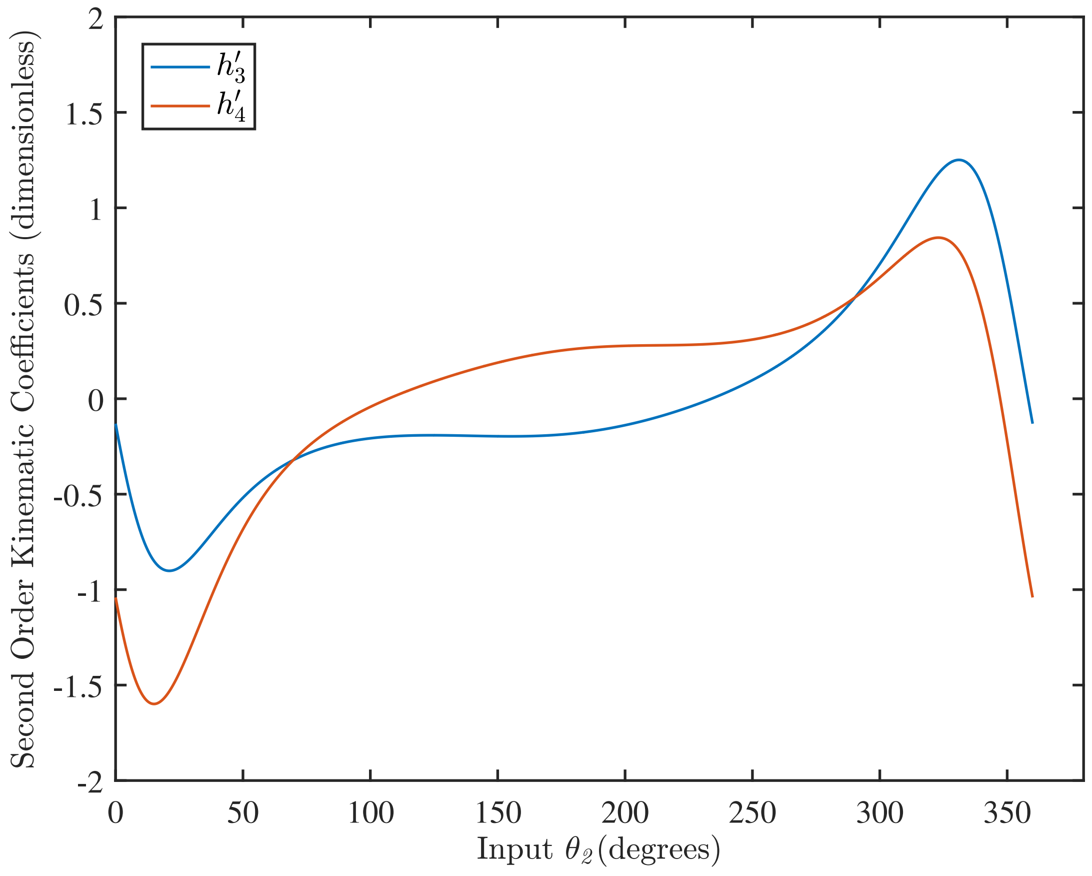

Figure 5: Plot of second order Kinematic Coefficients of links 3 and 4, h_3' and h_4' respectively, as a function of the given input angle θ_2.

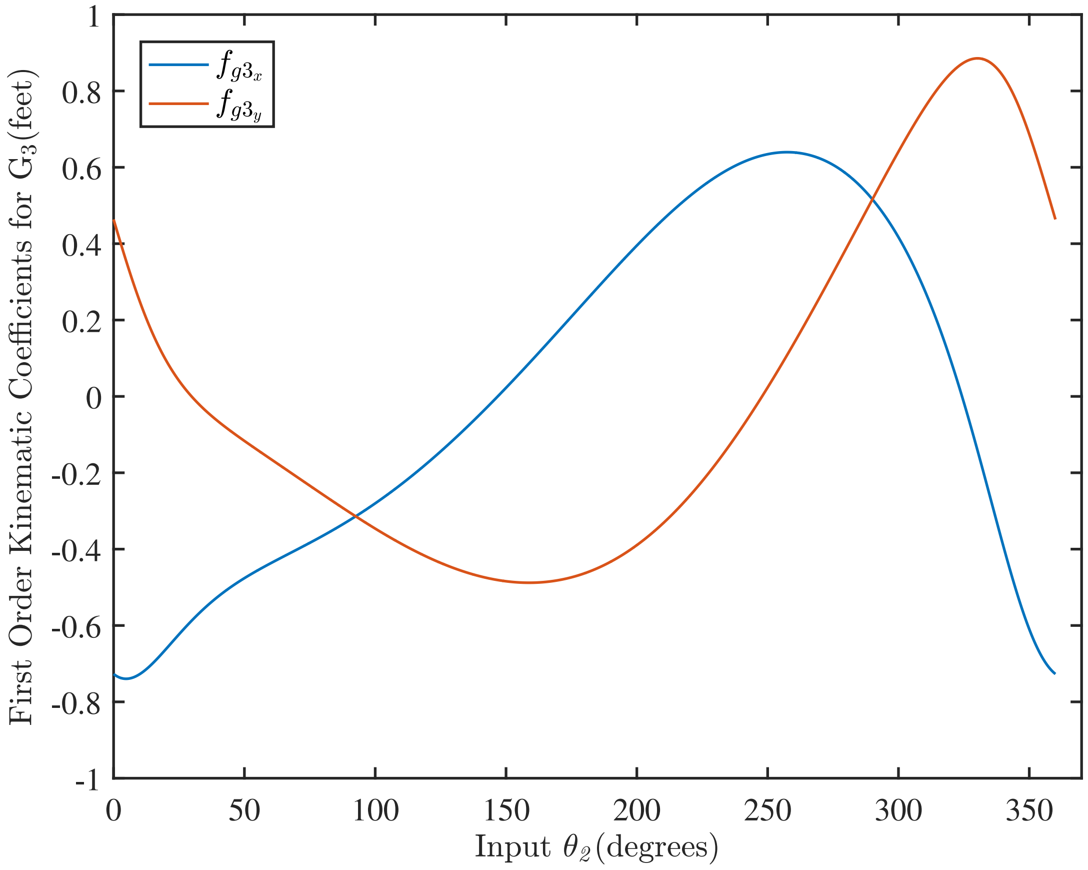

Figure 6: Plot of x-component and y-component first order kinematic coefficients, f_{g3_x} and f_{g3_y} respectively, for center of gravity G_3 of block connected to link 3 as a function of input angle θ_2.

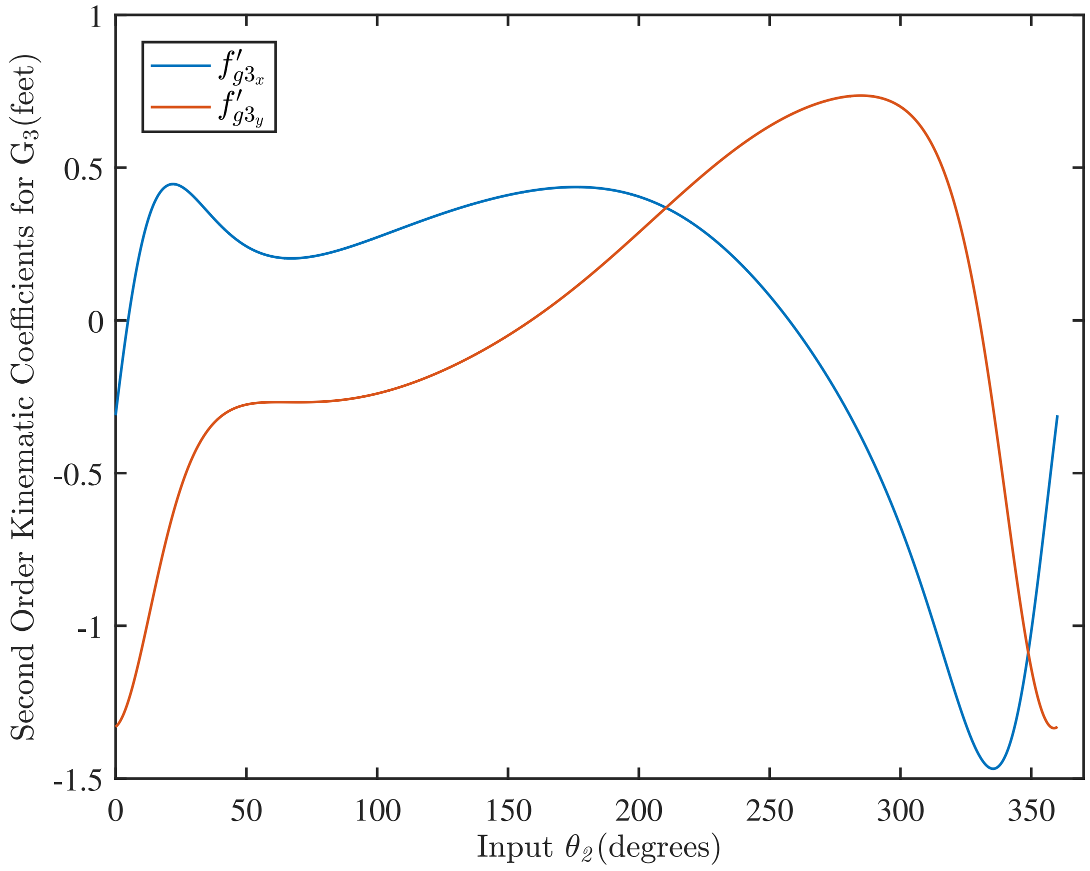

Figure 7: Plot of x-component and y-component second order kinematic coefficients,f'_{g3_x} and f'_{g3_y} respectively, for center of gravity G_3 of block connected to link 3 as a function of input angle θ_2.

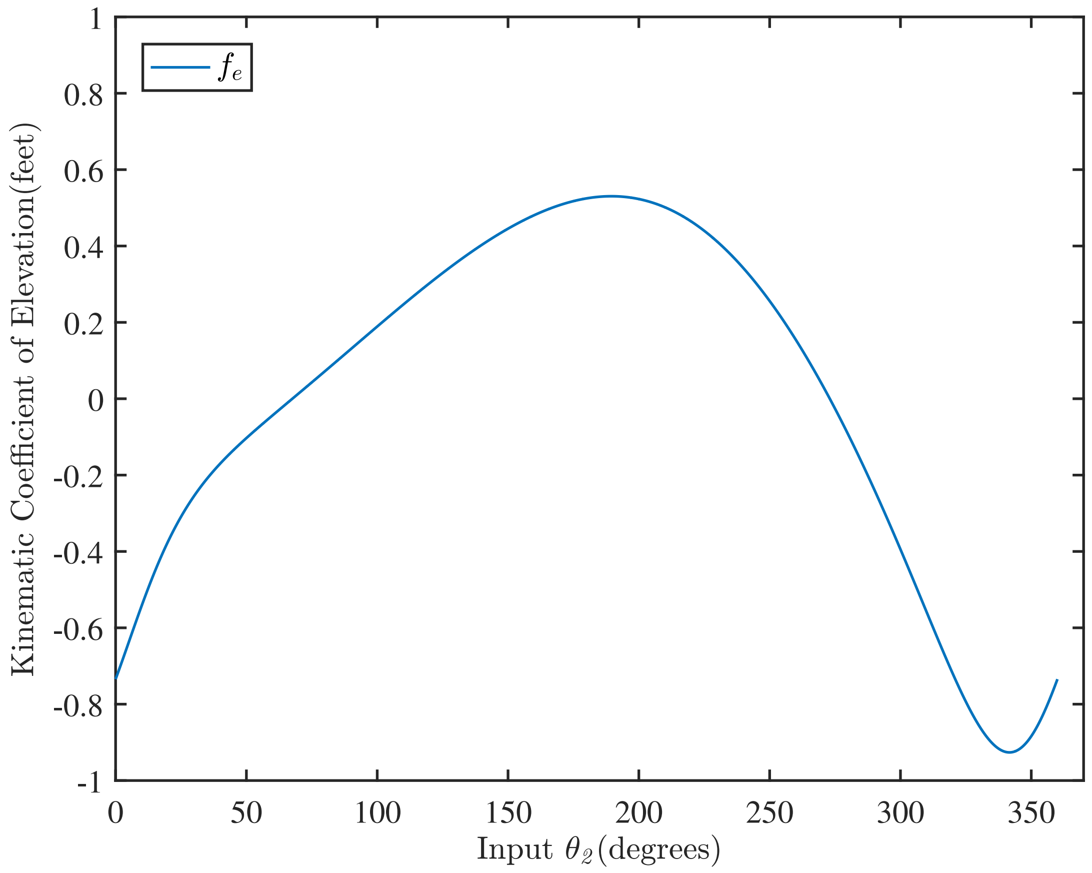

Figure 8: Plot of kinematic coefficient of elevation, f_e, for link 3 as a function of input angle θ_2.

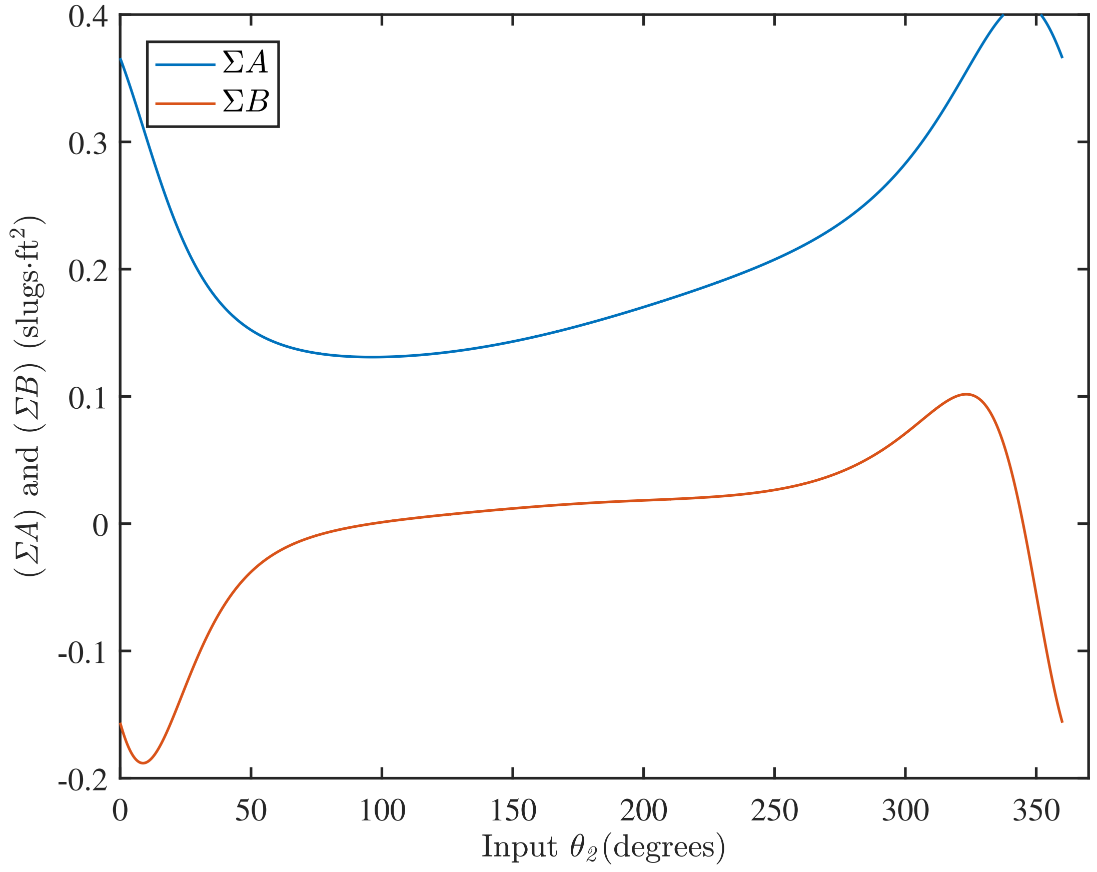

Figure 9: Plot of (ΣA) and (ΣB) for mechanism as a function of input angle θ_2.

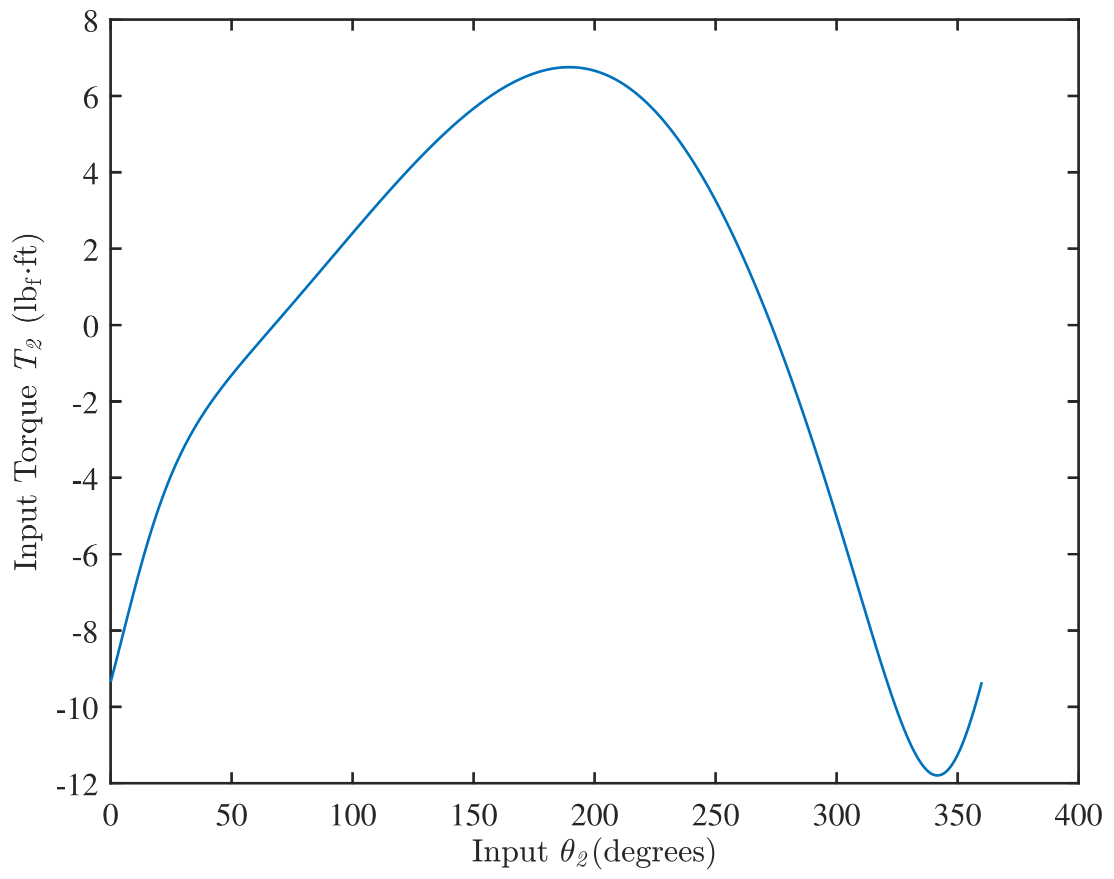

Figure 10: Plot of required torque, T_2 required for static equilibrium as a function of input angle θ_2 for one full revolution.

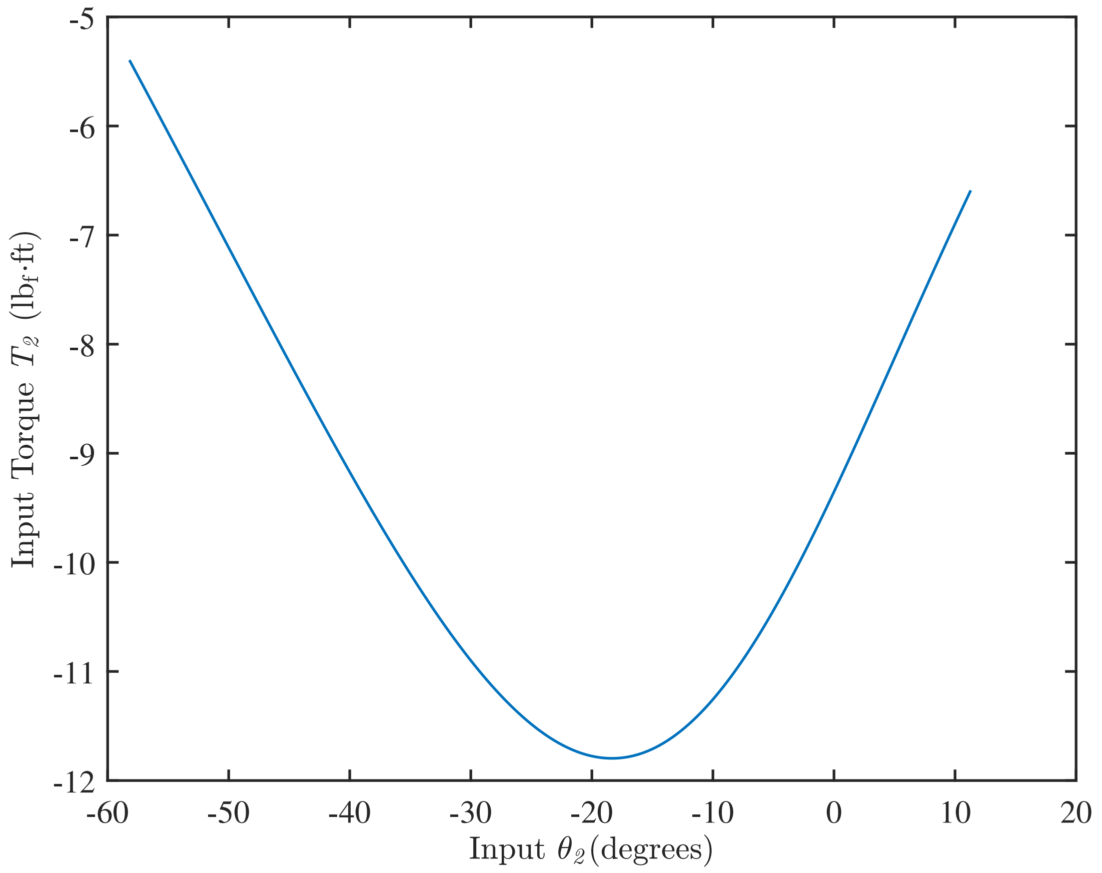

Figure 11: Plot of required torque, T_2 required for static equilibrium as a function of input angle θ_2 for initial lift from starting position to defined intermediate position.

Deliverable 5

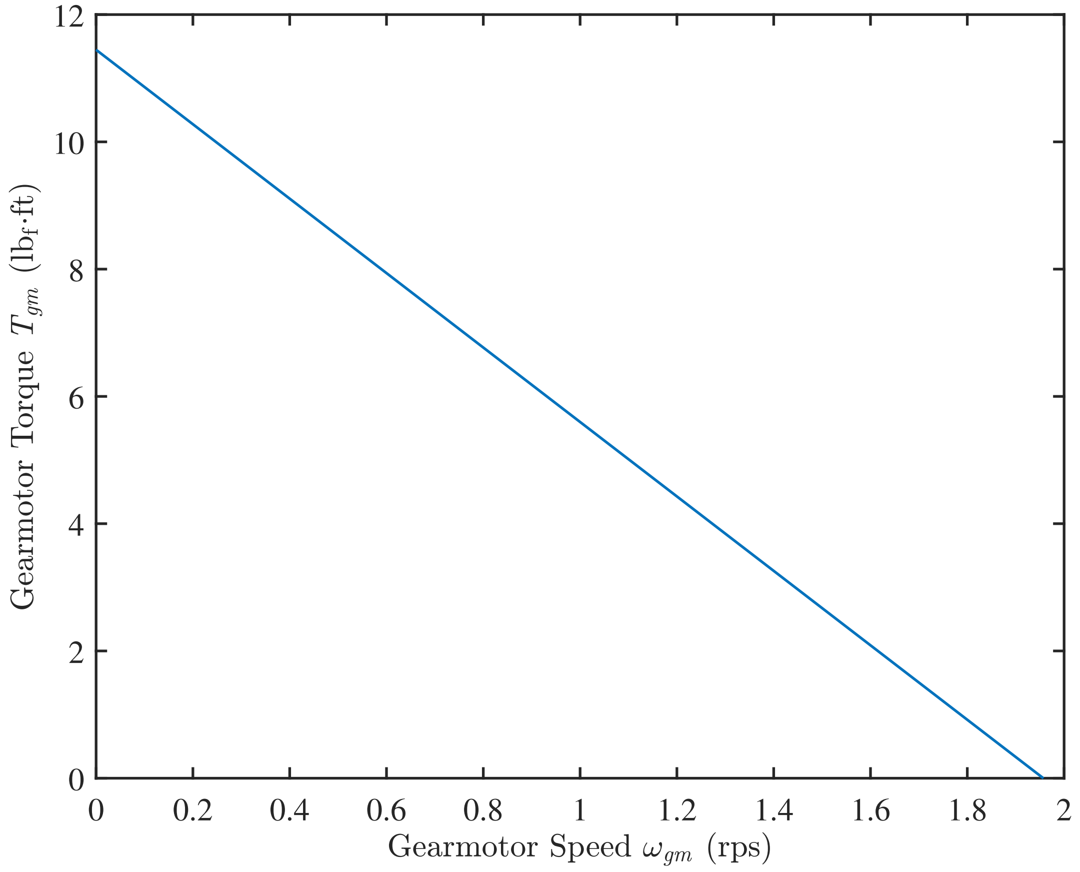

Figure 12: Plot of gearmotor Torque, T_{gm} for motor-gearbox combination as a function of gearmotor Speed ω_{gm}.

Deliverable 6 The resulting value of $\ddot{\theta}_2$ at $t = 0$ was calculated to be.

$$\ddot{\theta}_2 = -16.4738 \frac{\text{rad}}{\text{s}^2} \tag{7}$$

Appendix B

Results of optimized system using BaneBot RS 550 12V DC Motor and BaneBot P60 672:1 500 series gearbox.

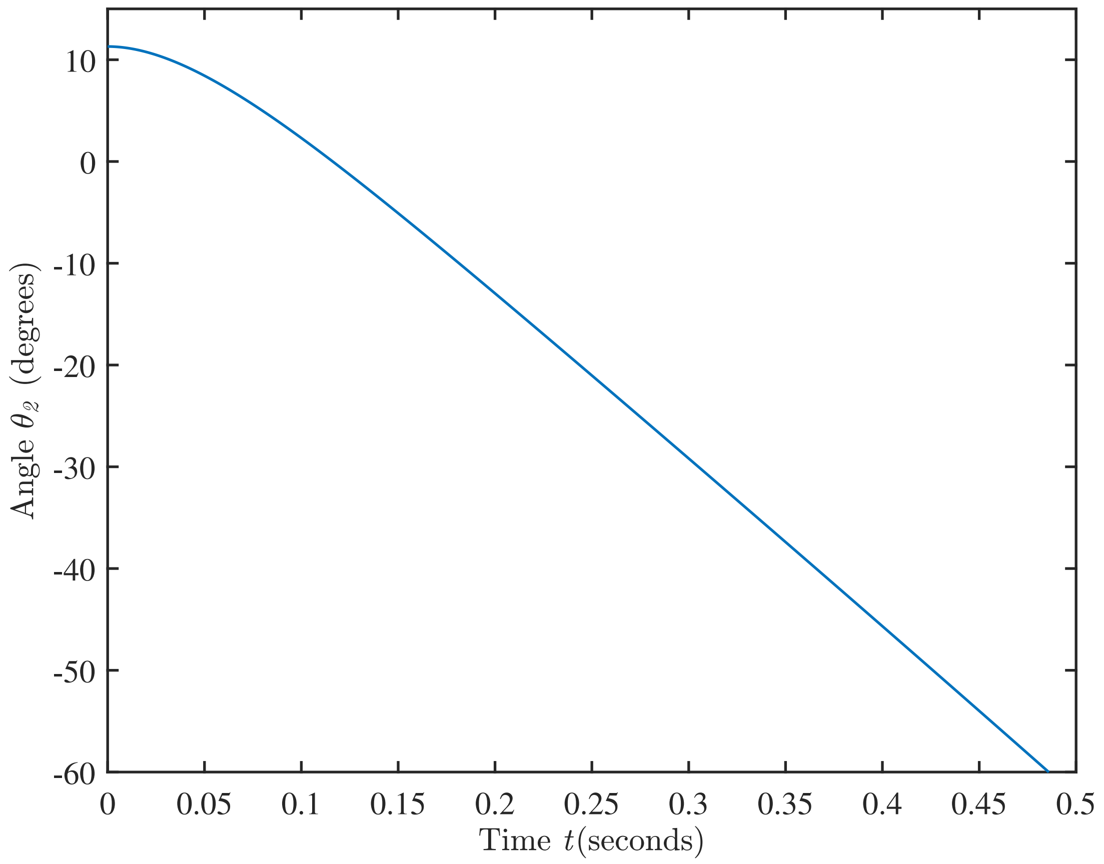

Figure 13: Plot of angle θ_2 as a function of time t for startup of mechanism with optimized drive system.

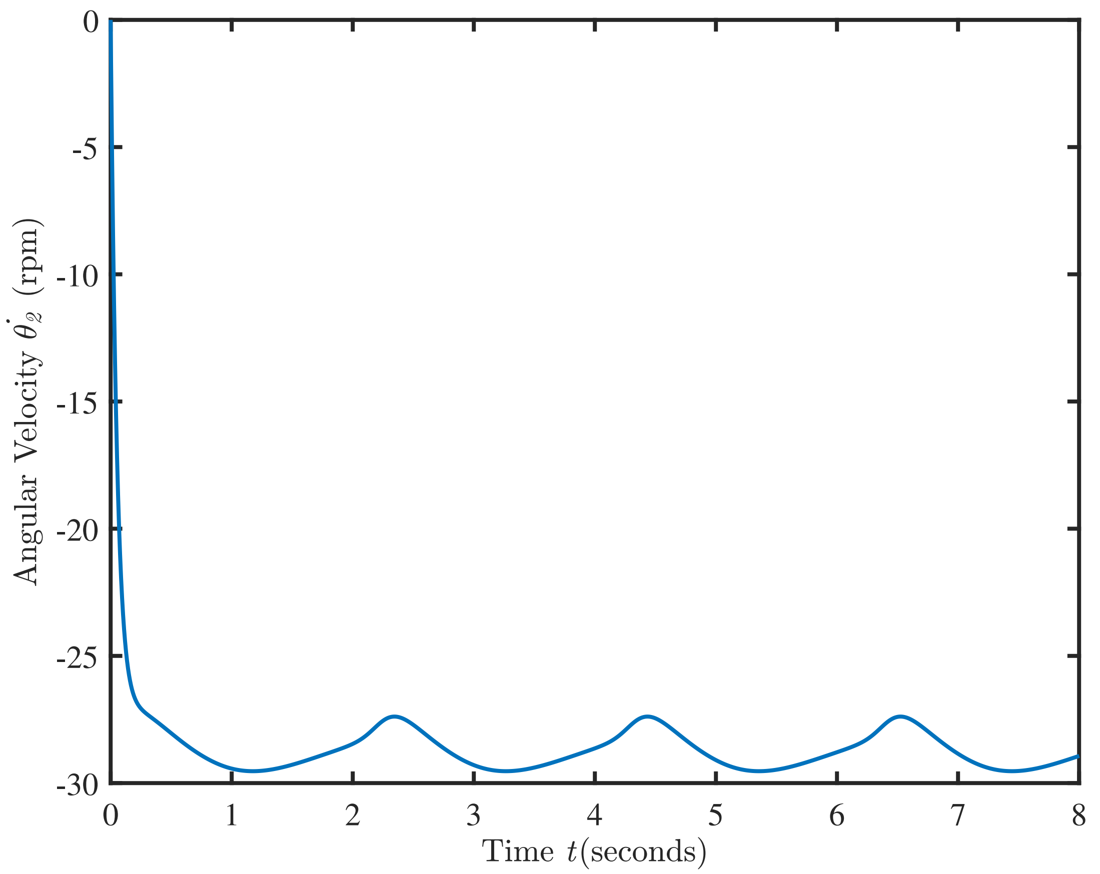

Figure 14: Plot of angular velocity $\dot{\theta}_2$ as a function of time t from start up to steady state of mechanism with optimized drive system.

Appendix C

Main code

```
%% AME 40423 Design Set 2

% ID: 37084
% Date Due: 12/5/18
% designSet2.m

clear
clc
clf

%% Enter Knowns

%scalar knowns
r1 = 3.20; %inches
theta1 = 0; %radians
r2 = 7.00; %inches
r3 = 7.80; %inches
r4 = 6.50; %inches
r3prime = 4.80; %inches
phi3 = -24.5*(pi/180);
theta_e = (360 - 64.1)*(pi/180);

resolution = 10000;
range = 69.5;

% input stats for gearmotor
%Banebot RS 395
T_m_stall = 16.65*0.0625*(1/12); %lbf*ft
w_m_max = 15500/60; %rev/s
motor_l = 1.83; %inches
motor_d = 1.08; %inches
motor_cost = 5.25; %$
motor_weight = 3.4; %oz

%banebot P60 gearbox with ratio of 132:1
R = 132; %gear ratio of 132:1
gearbox_cost = 64.50; %$
gearbox_weight = 9.6; %oz

drive_system_cost = motor_cost + gearbox_cost;
drive_system_weight = (motor_weight+gearbox_weight)...
```

```
    *0.0625;%lbf

%banebot P60 gearbox with ratio of 132:1
R = 132; %gear ratio of 132:1

l = 5/12; %feet
w = 3/12; %feet
h = 3/12; %feet
volume = l*w*h; %ft^3
density_steel = 0.283; %lbm/in^3 per Prof Stanisic
density_steel = density_steel*((12^3)/32.2); %slugs/ft^3
m_block = volume*density_steel; %slugs
g = 32.2;

%% Find all deliverables
% Estimate values of unknown scalars (x_0)
theta2 = 0;
theta3_0 = 123 * (pi/180); %radians
theta4_0 = 94* (pi/180); %radians

%create vector of size 361 to hold theta2 of 0 to 360 degrees
%and the theta3 and theta4 values that correspond with
%that to graph
theta2_graph_revolution = zeros(1,((360*resolution)+1));
theta3_graph = zeros(1,((360*resolution)+1));
theta4_graph = zeros(1,((360*resolution)+1));
h3_graph = zeros(1,((360*resolution)+1));
h4_graph = zeros(1,((360*resolution)+1));
h3prime_graph = zeros(1,((360*resolution)+1));
h4prime_graph = zeros(1,((360*resolution)+1));
f_g3x_graph = zeros(1,((360*resolution)+1));
f_g3y_graph = zeros(1,((360*resolution)+1));
f_g3x_prime_graph = zeros(1,((360*resolution)+1));
f_g3y_prime_graph = zeros(1,((360*resolution)+1));
f_e_graph = zeros(1,((360*resolution)+1));
sumA_graph = zeros(1,((360*resolution)+1));
sumB_graph = zeros(1,((360*resolution)+1));
T_2_1revolution_graph = zeros(1,((360*resolution)+1));


%for one revolution (steady state) omega2_double_dot (for
%static equilibrium) is 0
theta2_doubledot = 0;
theta2_dot = 0;
```

```
for n = 1:((360*resolution)+1)
[theta3, theta4] = FourBarNewton(r1, r2, r3, r4, ...
    theta2, theta3_0, theta4_0);
[h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
    theta2, theta3, theta4);
[f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
    G3_KCs(r2, r3prime, theta2, theta3, theta_e,...
    phi3, h3, h3prime);
[sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x, ...
    f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
    motor_l, motor_d);
[T_2] = calc_T_2(sumA,sumB,f_e,theta2_dot,...
    theta2_doubledot, m_block);

%store theta values for this loop
theta2_graph_revolution(n) = theta2 * (180/pi); %degrees
theta3_graph(n) = theta3 * (180/pi); %degrees
theta4_graph(n) = theta4 * (180/pi); %degrees
h3_graph(n) = h3; %dimensionless
h4_graph(n) = h4; %dimensionless
h3prime_graph(n) = h3prime; %dimensionless
h4prime_graph(n) = h4prime; %dimensionless
f_g3x_graph(n) = f_g3x; %feet
f_g3y_graph(n) = f_g3y; %feet
f_g3x_prime_graph(n) = f_g3x_prime; %feet
f_g3y_prime_graph(n) = f_g3y_prime; %feet
f_e_graph(n) = f_e; %feet
sumA_graph(n) = sumA; %slugs*ft^2
sumB_graph(n) = sumB; %slugs*ft^2
T_2_1revolution_graph(n) = T_2; %lbf*ft

%increment to next theta2 for next loop
theta2 = theta2 + ((pi/180)*(1/resolution));

%set initial x0 values for next loop to be x values of
%current loop
theta3_0 = theta3; %radians
theta4_0 = theta4; %radians
end

theta2 = 11.3 * (pi/180);
theta3_0 = 144.3 *(pi/180);
theta4_0 = 114.2 * (pi/180);

theta2_startup_graph = zeros(1,((range*resolution)+1));
```

```
T_2_startup_graph = zeros(1,((range*resolution)+1));

%for for startup and static equilibrium conditions
%omega2_doubledot is equal to a constant negative
%acceleration
for n = 1:((range*resolution)+1)
[theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
    theta2, theta3_0, theta4_0);
[h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
    theta2, theta3, theta4);
[f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
    G3_KCs(r2, r3prime, theta2, theta3, theta_e,...
    phi3, h3, h3prime);
[sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x,...
    f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
    motor_l, motor_d);
[T_2] = calc_T_2(sumA,sumB,f_e,theta2_dot,...
    theta2_doubledot, m_block);

%store theta values for this loop
theta2_startup_graph(n) = theta2 * (180/pi); %degrees
T_2_startup_graph(n) = T_2; %lbf*ft

%increment to next theta2 for next loop
theta2 = theta2 - ((pi/180)*(1/resolution));

%set initial x0 values for next loop to be x values of
%current loop
theta3_0 = theta3; %radians
theta4_0 = theta4; %radians
end

w_gm_graph = zeros(1,((ceil(w_m_max*resolution*2*pi))+1));
T_gm_graph = zeros(1,((ceil(w_m_max*resolution*2*pi))+1));
w_gm = 0;
for i = 1:((ceil(w_m_max*resolution*2*pi))+1)
    [T_gm] = gearmotor_output(w_gm,w_m_max,T_m_stall,R);

    w_gm_graph(i) = w_gm;
    T_gm_graph(i) = -1*T_gm;

    w_gm = w_gm + (1/resolution);
end

%% Euler's method
```

```
%initially, the system is at rest, so theta2_dot is zero
theta2_dot_0 = 0;
theta2_dot = theta2_dot_0;

%calculate initial theta2_ddot
theta2 = 11.3 * (pi/180);
theta3_0 = 144.3 *(pi/180);
theta4_0 = 114.2 * (pi/180);

[theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
    theta2, theta3_0, theta4_0);
[h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
    theta2, theta3, theta4);
[f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
    G3_KCs(r2, r3prime, theta2, theta3, theta_e,...
    phi3, h3, h3prime);
[sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x,...
    f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
    motor_l, motor_d);
[T_2] = gearmotor_output(theta2_dot,w_m_max,...
    T_m_stall,R);

T_2_0 = T_2;

theta2_ddot_0 = (T_2_0 - (sumB*(theta2_dot^2))...
    -(m_block*g*f_e))/sumA;
theta2_ddot = theta2_ddot_0;

%initialize time variables
dt = 1/resolution; %seconds
t_0 = 0;
t_max = 3;
t = t_0;

%initialize graphing vectors
t_graph = zeros(1, ((t_max*resolution)+1));
theta2_graph_time = zeros(1, ((t_max*resolution)+1));
theta2_dot_graph = zeros(1, ((t_max*resolution)+1));

for n = 1:((t_max*resolution)+1)

    t_graph(n) = t;
    theta2_graph_time(n) = theta2;
    theta2_dot_graph(n) = theta2_dot;
    %kinematic calculations
    delta_theta2 = (theta2_dot*dt) +...
        (0.5* theta2_ddot *(dt^2));
    theta2 = theta2 + delta_theta2;

    delta_theta2_dot = theta2_ddot*dt;
    theta2_dot = theta2_dot + delta_theta2_dot;

    t = t+dt;

    %calculate using functions
    [theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
        theta2, theta3_0, theta4_0);
    [h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
        theta2, theta3, theta4);
    [f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
        G3_KCs(r2, r3prime, theta2, theta3, theta_e,...
        phi3, h3, h3prime);
    [sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x,...
        f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
        motor_l, motor_d);
    [T_2] = gearmotor_output(theta2_dot,w_m_max,...
        T_m_stall,R);
    theta2_ddot = (T_2 - (sumB*(theta2_dot^2))-...
        (m_block*g*f_e))/sumA;

    %set initial x0 values for next loop to be x
    %values of current loop
    theta3_0 = theta3; %radians
    theta4_0 = theta4; %radians
end

%convert results to proper units
theta2_graph_time = theta2_graph_time*(180/pi);
theta2_dot_graph = theta2_dot_graph*(60/(2*pi));

%find coefficient of fluctuation
%determine start of steady state by inspection
%starts @ t = 1.6 find indices for 1.6sec and 6s
i_start = 16001;
i_stop = 60001;
%find max speed in this range
theta2_dot_max = ...
    max(theta2_dot_graph(i_start:i_stop));
theta2_dot_min = ...
```

```
    min(theta2_dot_graph(i_start:i_stop));
theta2_dot_avg = (theta2_dot_max+...
    theta2_dot_min)/2;
C_f_original = abs((theta2_dot_max-theta2_dot_min)/...
    theta2_dot_avg);
disp('Coefficient of Fluctuation for link 2 in system =')
disp(num2str(C_f_original))

%% Deliverable 1
figure(11)
plot(theta2_graph_revolution, theta3_graph)
hold on
plot(theta2_graph_revolution, theta4_graph)
ylabel('Output Position $\displaystyle\it \rm (degrees)$',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$\theta_3$','$\theta_4$', 'location',...
    'northwest');
set(leg1,'Interpreter','latex');
axis([0,380,50,500])
hold off
figure(12)
plot(theta2_graph_revolution, h3_graph)
hold on
plot(theta2_graph_revolution, h4_graph)
ylabel('First Order Kinematic Coefficients (dimensionless)',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$h_3$','$h_4$', 'location', 'southeast');
set(leg1,'Interpreter','latex');
axis([0,380,0,2])
hold off
figure(13)
plot(theta2_graph_revolution, h3prime_graph)
hold on
plot(theta2_graph_revolution, h4prime_graph)
ylabel('Second Order Kinematic Coefficients (dimensionless)',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$h_3''$','$h_4''$', 'location', 'northwest');
set(leg1,'Interpreter','latex');
axis([0,380,-2,2])
```

```
hold off

%% Deliverable 2
figure(21)
plot(theta2_graph_revolution, f_g3x_graph)
hold on
plot(theta2_graph_revolution, f_g3y_graph)
ylabel('First Order Kinematic Coefficients for $\displaystyle G_3 (feet)$',
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$f_{g3_{x}}$','$f_{g3_{y}}$', 'location',...
    'northwest');
set(leg1,'Interpreter','latex');
axis([0,370,-1,1])
hold off
figure(22)
plot(theta2_graph_revolution, f_g3x_prime_graph)
hold on
plot(theta2_graph_revolution, f_g3y_prime_graph)
ylabel('Second Order Kinematic Coefficients for $\displaystyle G_3 (feet)$'
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$f_{g3_{x}}''$','$f_{g3_{y}}''$', 'location',...
    'northwest');
set(leg1,'Interpreter','latex');
axis([0,370,-1.5,1])
hold off
figure(23)
plot(theta2_graph_revolution, f_e_graph)
hold on
ylabel('Kinematic Coefficient of Elevation$\displaystyle\it \rm (feet)$',..
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$','interpreter',..
    'latex');
leg1 = legend('$f_{e}$', 'location', 'northwest');
set(leg1,'Interpreter','latex');
axis([0,370,-1,1])
hold off

%% Deliverable 3
figure(31)
plot(theta2_graph_revolution, sumA_graph)
hold on
```

```
plot(theta2_graph_revolution, sumB_graph)
ylabel('$\displaystyle\it (\Sigma{A}) \rm$ and $\displaystyle\it '...
    '(\Sigma{B})$ $\displaystyle\rm (slugs{\cdotp}ft^{2})$',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
leg1 = legend('$\Sigma{A}$','$\Sigma{B}$', 'location',...
    'northwest');
set(leg1,'Interpreter','latex');
axis([0,370,-0.2,0.4])
hold off

%% Deliverable 4
figure(41)
plot(theta2_graph_revolution, T_2_1revolution_graph)
hold on
ylabel('Input Torque $\displaystyle\it T_{2} \rm$'...
    '$\displaystyle\rm (lb_{f}{\cdotp}ft)$',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
%axis([0,370,-0.2,0.4])
hold off
figure(42)
plot(theta2_startup_graph, T_2_startup_graph)
hold on
ylabel('Input Torque $\displaystyle\it T_{2} \rm$'...
    '$\displaystyle\rm (lb_{f}{\cdotp}ft)$',...
    'interpreter','latex');
xlabel('Input  $\displaystyle \it \theta_2 \rm (degrees)$',...
    'interpreter','latex');
%axis([0,370,-0.2,0.4])
hold off

%% Deliverable 5
figure(51)
plot((w_gm_graph/(2*pi)), T_gm_graph)
hold on
ylabel('Gearmotor Torque $\displaystyle\it T_{gm}$'...
    '$\displaystyle\rm (lb_{f}{\cdotp}ft)$',...
    'interpreter','latex');
xlabel('Gearmotor Speed $\displaystyle\it {\omega}_{gm}$'...
    '$\displaystyle\rm (rps)$','interpreter','latex');
axis([0,2,0,12])
hold off
```

```
%% Deliverable 6
disp(['theta2_doubledot = ', num2str(theta2_ddot_0) ''])

%% Deliverable 7
figure(71)
plot(t_graph,theta2_graph_time)
hold on
ylabel('Angle $\displaystyle\it \theta_{2} \rm$'...
    '$\displaystyle\rm (degrees)$','interpreter','latex');
xlabel('Time $\displaystyle \it t \rm (seconds)$',...
    'interpreter','latex');
axis([0,1.2,-93,15])
hold off
figure(72)
plot(t_graph,theta2_dot_graph)
hold on
ylabel('Angular Velocity $\displaystyle\it \dot{\theta_{2}}$' ...
    '$\displaystyle\rm (rpm)$','interpreter','latex');
xlabel('Time $\displaystyle \it t \rm (seconds)$',...
    'interpreter','latex');
axis([0, t_max,-180,0])
hold off

%% Optimize the drive system

% input stats for gearmotor
%Banebot RS 550
T_m_stall = 69.16*0.0625*(1/12); %lbf*ft
w_m_max = 19300/60; %rev/s
motor_l = 2.24; %inches
motor_d = 1.41; %inches
motor_cost_optimize = 7.25;
motor_weight_optimize = 7.7; %oz

%banebot P60 gearbox with ratio of 672:1
R = 672; %gear ratio of 672:1
gearbox_cost_optimize = 77.95;
gearbox_weight_optimize = 11.4; %oz

drive_system_cost_optimize = ...
    motor_cost_optimize + gearbox_cost_optimize;
drive_system_weight_optimize = (motor_weight_optimize+...
    gearbox_weight_optimize)*0.0625;%lbf
```

```
%for one revolution (steady state) omega2_double_dot (for static
%equilibrium) is 0
theta2_dot_0 = 0;
theta2_dot = theta2_dot_0;

%calculate initial theta2_ddot
theta2 = 11.3 * (pi/180);
theta3_0 = 144.3 *(pi/180);
theta4_0 = 114.2 * (pi/180);

[theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
    theta2, theta3_0, theta4_0);
[h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
    theta2, theta3, theta4);
[f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
    G3_KCs(r2, r3prime, theta2, theta3, theta_e, phi3, h3, h3prime);
[sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x,...
    f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
    motor_l, motor_d);
[T_2] = gearmotor_output(theta2_dot,w_m_max,...
    T_m_stall,R);

T_2_0 = T_2;

theta2_ddot_0 = (T_2_0 - (sumB*(theta2_dot^2))-...
    (m_block*g*f_e))/sumA;
theta2_ddot = theta2_ddot_0;

%initialize time variables
dt = 1/resolution; %seconds
t_0 = 0;
t_max = 5;
t = t_0;

%initialize graphing vectors
t_graph_optimize = zeros(1, ((t_max*resolution)+1));
theta2_graph_time_optimize = zeros(1, ((t_max*resolution)+1));
theta2_dot_graph_optimize = zeros(1, ((t_max*resolution)+1));

for n = 1:((t_max*resolution)+1)

    t_graph_optimize(n) = t;
    theta2_graph_time_optimize(n) = theta2;
    theta2_dot_graph_optimize(n) = theta2_dot;
    %kinematic calculations
    delta_theta2 = (theta2_dot*dt) + ...
        (0.5* theta2_ddot *(dt^2));
    theta2 = theta2 + delta_theta2;

    delta_theta2_dot = theta2_ddot*dt;
    theta2_dot = theta2_dot + delta_theta2_dot;

    t = t+dt;

    %calculate using functions
    [theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
        theta2, theta3_0, theta4_0);
    [h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4,...
        theta2, theta3, theta4);
    [f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] =...
        G3_KCs(r2, r3prime, theta2, theta3, theta_e,...
        phi3, h3, h3prime);
    [sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x,...
        f_g3y, f_g3x_prime, f_g3y_prime, R, m_block, l, w,...
        motor_l, motor_d);
    [T_2] = gearmotor_output(theta2_dot,w_m_max,T_m_stall,R);
    theta2_ddot = (T_2 - (sumB*(theta2_dot^2))-...
        (m_block*g*f_e))/sumA;

    %set initial x0 values for next loop to be x values of current loop
    theta3_0 = theta3; %radians
    theta4_0 = theta4; %radians
end

%convert results to proper units
theta2_graph_time_optimize = ...
    theta2_graph_time_optimize*(180/pi);
theta2_dot_graph_optimize = ...
    theta2_dot_graph_optimize*(60/(2*pi));

%find coefficient of fluctuation
%determine start of steady state by inspection
%starts @ t = 1sec find indices for 1sec and 6s
i_start = 10001;
i_stop = 60001;
%find max speed in this range
theta2_dot_max_optimize = ...
    max(theta2_dot_graph_optimize(i_start:i_stop));
theta2_dot_min_optimize = ...
```

```
    min(theta2_dot_graph_optimize(i_start:i_stop));
theta2_dot_avg_optimize = (theta2_dot_max_optimize+...
    theta2_dot_min_optimize)/2;
C_f_optimize = abs((theta2_dot_max_optimize-...
    theta2_dot_min_optimize)/theta2_dot_avg_optimize);
disp('Coefficient of Fluctuation for link 2 in optimized system =')
disp(num2str(C_f_optimize))


figure(81)
plot(t_graph_optimize,theta2_graph_time_optimize)
hold on
ylabel('Angle $\displaystyle\it \theta_{2} \rm$'...
    '$\displaystyle\rm (degrees)$',...
    'interpreter','latex');
xlabel('Time $\displaystyle \it t \rm (seconds)$',...
    'interpreter','latex');
axis([0,0.4,-60,15])
hold off
figure(82)
plot(t_graph_optimize,theta2_dot_graph_optimize)
hold on
ylabel('Angular Velocity $\displaystyle\it \dot{\theta_{2}}$'...
    '$\displaystyle\rm (rpm)$','interpreter','latex');
xlabel('Time $\displaystyle \it t \rm (seconds)$',...
    'interpreter','latex');
axis([0, 8,-30,0])
hold off

%% determine state variable 1
%find what time t the original system crosses intermediate position
i_state_1 = find(theta2_graph_time < -58.2, 1);
t_state_1 = t_graph(i_state_1);
state_1_met = t_state_1 <= 0.5;
disp('Original System:')
disp(['Time required to move block to intermediate position ='])
disp([num2str(t_state_1),' seconds'])
if state_1_met
    disp('State Variable 1 Constraint met')
else
    disp('State Variable 1 Constraint not met')
end
disp([' '])
%optimized system
i_state_1_opt = find(theta2_graph_time_optimize < -58.2, 1);
```

```
t_state_1_opt = t_graph_optimize(i_state_1_opt);
state_1_met_opt = t_state_1_opt <= 0.5;
disp('Optimized System:')
disp(['Time required to move block to intermediate position ='])
disp([num2str(t_state_1_opt),' seconds'])
if state_1_met_opt
    disp('State Variable 1 Constraint met')
else
    disp('State Variable 1 Constraint not met')
end

disp([' '])
disp([' '])

%% determine state variable 2
%determined by inspection
t_state_2 = 1.6;
state_2_met = t_state_2 <= 2;
disp('Original System:')
disp(['Time required to move block to intermediate position ='])
disp([num2str(t_state_2),' seconds'])
if state_2_met
    disp('State Variable 2 Constraint met')
else
    disp('State Variable 2 Constraint not met')
end
disp([' '])
%optimized system
t_state_2_opt = 1;
state_2_met_opt = t_state_2_opt <= 2;
disp('Optimized System:')
disp(['Time required to move block to intermediate position ='])
disp([num2str(t_state_2_opt),' seconds'])
if state_2_met_opt
    disp('State Variable 2 Constraint met')
else
    disp('State Variable 2 Constraint not met')
end

disp([' '])
disp([' '])

%% determine state variable 3

state_3_met = C_f_original < 0.1;
```

```
disp('Original System:')
disp(['Coefficient of fluctuation of input speed ='])
disp([num2str(C_f_original),' '])
if state_3_met
    disp('State Variable 3 Constraint met')
else
    disp('State Variable 3 Constraint not met')
end
disp([' '])
state_3_met_opt = C_f_optimize < 0.1;
disp('Optimized System:')
disp(['Coefficient of fluctuation of input speed ='])
disp([num2str(C_f_optimize),' '])
if state_3_met_opt
    disp('State Variable 3 Constraint met')
else
    disp('State Variable 3 Constraint not met')
end

disp([' '])
disp([' '])
%% determine state variable 4

state_4_met = drive_system_cost < 60;
disp('Original System:')
disp(['Cost of drive system ='])
disp(['$',num2str(drive_system_cost)])
if state_4_met
    disp('State Variable 4 Constraint met')
else
    disp('State Variable 4 Constraint not met')
end
disp([' '])
state_4_met_opt = drive_system_cost_optimize < 60;
disp('Optimized System:')
disp(['Cost of drive system ='])
disp(['$',num2str(drive_system_cost_optimize)])
if state_4_met_opt
    disp('State Variable 4 Constraint met')
else
    disp('State Variable 4 Constraint not met')
end

disp([' '])
disp([' '])
```

```
%% determine state variable 5

state_5_met = drive_system_weight < 2;
disp('Original System:')
disp(['Weight of drive system ='])
disp(['$',num2str(drive_system_weight),' lbf'])
if state_5_met
    disp('State Variable 5 Constraint met')
else
    disp('State Variable 5 Constraint not met')
end
disp([' '])
state_5_met_opt = drive_system_weight_optimize < 2;
disp('Optimized System:')
disp(['Weight of drive system ='])
disp([num2str(drive_system_weight_optimize),' lbf'])
if state_5_met_opt
    disp('State Variable 5 Constraint met')
else
    disp('State Variable 5 Constraint not met')
end
```

Four Bar Position Function

```
function [theta3, theta4] = FourBarNewton(r1, r2, r3, r4,...
    theta2, theta3_0, theta4_0)

%% AME 40423 Programming Set 2 Newton’s Method Function Open Four Bar

% ID: 37084
% FourBarNewton.m

%uses newton’s method to solve for values of output angles theta3 and
%theta4 based off of known lengths of r1,r2,r3,r4 and the input value
%theta3 as well as estimated values of theta3_0 and theta4_0

%% function start
tolerance = 0.001;
error = 1;
%reset initial x values to 0
theta3 = 0;
theta4 = 0;
%make dummy counter i
i = 0;

x_0 = [theta3_0; theta4_0;];

while (error>tolerance)

%% make x0 equal to previous x if this is not first loop
if i > 0
x_0 = x;
theta3_0 = x_0(1);
theta4_0 = x_0(2);
end

%% compute fbar computed at x0

f1 = r1 + (r4*cos(theta4_0))- (r3*cos(theta3_0)) - (r2*cos(theta2));
f2 = (r4*sin(theta4_0)) - (r3*sin(theta3_0)) - (r2*sin(theta2));

%define fbar
fbar = [f1; f2;];

%% compute jacobian at x0 (A)

dfdtheta3 = [(r3)*sin(theta3_0); (-1*r3)*cos(theta3_0);];
```

```
dfdtheta4 = [(-1*r4)*sin(theta4_0); (r4)*cos(theta4_0);];

%define A
A = [dfdtheta3 dfdtheta4];

%% solve for x

x = x_0 - inv(A)*fbar;

theta3 = x(1);
theta4 = x(2);

i = i+1;

%% check error
error = norm(x-x_0);
end
```

Four Bar Kinematic Coefficients Function

```
function [h3,h3prime,h4,h4prime] = BasicKCs(r2, r3, r4, theta2,...
    theta3, theta4)

%% AME 40423 Design Set 2 Kinematic Coefficient Calculator
% ID: 37084
% BasicKCs.m

%Calculate values of kinematic coefficients from solved position problem

%% function start

h3 = (r2*r4*sin(theta4-theta2))/(r3*r4*sin(theta3-theta4));
h4 = (r2*r3*sin(theta3-theta2))/(r3*r4*sin(theta3-theta4));

B = [(-r2*cos(theta2))-(r3*cos(theta3)*(h3^2))+(r4*cos(theta4)*(h4^2));...
    (-r2*sin(theta2))-(r3*sin(theta3)*(h3^2))+(r4*sin(theta4)*(h4^2));];
J = [(r3*sin(theta3)), (-r4*sin(theta4)); (-r3*cos(theta3)),...
    (r4*cos(theta4));];

hprime = inv(J)*B;

h3prime = hprime(1);
h4prime = hprime(2);
```

G_3 Kinematic Coefficients Function

```
function [f_g3x, f_g3y, f_g3x_prime, f_g3y_prime, f_e] = G3_KCs(r2,...
    r3prime, theta2, theta3, theta_e, phi3, h3, h3prime)

%% AME 40423 Design Set 2 Kinematic Coefficient Calculator
% ID: 37084
% G3_KCs.m

%Calculate values of kinematic coefficients for G3 from solved position
%problem

%% function start

% define r_g3x and r_g3y
r_g3x = (r2*cos(theta2)) + (r3prime*cos(theta3+phi3));
r_g3y = (r2*sin(theta2)) + (r3prime*sin(theta3+phi3));

%calculate first order KCs
f_g3x = (-1*r2*sin(theta2)) - (r3prime*sin(theta3+phi3)*h3);
f_g3y = (r2*cos(theta2)) + (r3prime*cos(theta3+phi3)*h3);

%calculate second order KCs
f_g3x_prime = (-1*r2*cos(theta2)) - (r3prime*cos(theta3+phi3)*(h3^2)) ...
    - (r3prime*sin(theta3+phi3)*h3prime);
f_g3y_prime = (-1*r2*sin(theta2)) - (r3prime*sin(theta3+phi3)*(h3^2)) ...
    + (r3prime*cos(theta3+phi3)*h3prime);

f_e = (f_g3x*cos(theta_e)) + (f_g3y*sin(theta_e));

%convert resulting KCs to feet
f_g3x = f_g3x/12;
f_g3y = f_g3y/12;
f_g3x_prime = f_g3x_prime/12;
f_g3y_prime = f_g3y_prime/12;
f_e = f_e/12;
```

(ΣA) and (ΣB) Calculation Function

```
function [sumA, sumB] = sum_As_sum_Bs(h3, h3prime, f_g3x, f_g3y,...
    f_g3x_prime, f_g3y_prime, R, m_block, l, w, motor_l, motor_d)

%% AME 40423 Design Set 2 Kinematic Coefficient Calculator
% ID: 37084
% sum_As_sum_Bs.m

%Calculate sum of As and sum of Bs

%% function start

%find mass of block
g = 32.2;
density_copper = 0.324; %lbm/in^3
density_copper = density_copper * ((12^3)/32.2); %slugs/ft^3
%takes assumptions from previous motor and used them to find the
%approximate dimensions of new motor coils, assumes it scales
length_coils = ((1.2/1.83)*motor_l)/12;
d_coils = ((3/4/1.08)*motor_d)/12;
volume_coils = length_coils*((d_coils/2)^2)*pi;
m_coils = volume_coils*density_copper;
h_coils = R;

%block is assumed to be so much heavier than the links that they
%simply go
%to zero

%calculate As and Bs of links 1, 2, and 4 (in this case 0)
A_1 = 0;
A_2 = 0;
A_4 = 0;

B_1 = 0;
B_2 = 0;
B_4 = 0;


% calculate A and B of block/link3
I_g3 = (1/12)*m_block*((l^2)+(w^2)); %slugs*ft^2
A_3 = (m_block*((f_g3x^2)+(f_g3y^2))) + (I_g3*(h3^2)); %slugs*ft^2
B_3 = (m_block*((f_g3x*f_g3x_prime)+(f_g3y*f_g3y_prime))) +...
    (I_g3*(h3*h3prime)); %slugs*ft^2
```

```
% calculate A and B of motor coils, assume that the fg and fg_prime
%of the coils is 0
I_motor = 0.5*m_coils*((d_coils/2)^2);
A_motor = (I_motor*(h_coils^2));
B_motor = 0;

%sum all the As and Bs
sumA = A_1 + A_2 + A_3 + A_4 + A_motor;
sumB = B_1 + B_2 + B_3 + B_4 + B_motor;
```

T_2 calculation function

```
function [T_2] = calc_T_2(sumA,sumB,f_e,theta2_dot,...
    theta2_doubledot, m_block)

%% AME 40423 Design Set 2 Kinematic Coefficient Calculator
% ID: 37084
% calc_T_2.m

%Calculate T_2 for static equilibrium
% NOTE that in this case we are assuming that T_2 is positive, so motor
% torque is negative

%% function start

%because it is being calculated for static equilibrium, the velocity
%of the link is assumed to be zero
g = 32.2;

%block is assumed to be so much heavier than the links that they simply
%go to zero
T_2 = ((sumA*theta2_doubledot) + (sumB*(theta2_dot^2)) + ...
    (m_block*g*f_e));
```

Gearmotor curve calculation function

```
function [T_2] = gearmotor_output(theta2_dot,w_m_max,T_m_stall,R)

%% AME 40423 Design Set 2 gearmotor output
% ID: 37084
% gearmotor_output.m

%Calculate output of gearmotor from motor/gearbox assembly

%% function start
T_gm_stall = T_m_stall * R * -1;
w_gm_max = (w_m_max/R)*2*pi;
theta2_dot = abs(theta2_dot);

T_2 = T_gm_stall*(1-(theta2_dot/w_gm_max));
```