Individual Engineering Analysis

The Flying Squirrel Team

Motor Torque Analysis

Joseph Mathews

ME-476C-001

April 27, 2025

1. Introduction

The purpose of this report is to detail one of the five analyses that Dr. Razavian requested that the team perform for the Flying Squirrel robot. He requested that a structural analysis be performed on the top and bottom halves of the robot's design, an analysis on the torque all four motors will experience to output the desired force, as well as the speed the motors will need to turn to move the robot at the desires speed, and the total power the robot will need for all of its motors and other systems. The analysis I performed was the torque experienced by the motors, and the process used is detailed below. Several prediction calculations were performed, and a MATLAB script was created to determine the average and maximum torque values. These results were used as a factor in determining the appropriate motors for the Flying Squirrel.

2. Torque Estimate Analysis

2.1 Initial Estimate

For the second presentation, a torque estimation of the driving motors was performed for the mathematical modelling section. The first of the guiding assumptions used in these calculations were that the cable tension was at the maximum amount of force as detailed in our engineering requirements, which was 10N. The second was that it was only accounting for one motor, with the third assuming that the robot was in equilibrium. The last guiding assumption was that the motor's winch had a radius of 0.5 inches. A summary of this initial estimation is included below. The result of this calculation was that the maximum applied torque onto the motors would be 0.127 Nm.

$F = 10N$ - Maximum required tension of the wires

$r = 0.5\ in = 0.0127m$ - Radius of the winch

τ - Estimated maximum torque

$$\tau = F \times r = 0.127Nm \quad \text{(Eq 1)}$$

2.2 Safety Considerations

After presenting these results in a client meeting the following week, Dr. Razavian suggested a factor of safety of 2 should be applied to the estimate. The summary of that calculation is listed below. He told the team to work on a MATLAB script to calculate the torques in different positions the robot could be, and said that we should expect the values from the script to be similar to the adjusted estimation.

$\tau = 0.127\ Nm$ - Calculated torque

$F.O.S. = 2$ - Factor of safety

$\tau_{Adjusted}$ - Estimated maximum torque adjusted by the factor of safety

$$\tau_{Adjusted} = \tau \times F.O.S. = 0.254\ Nm \quad \text{(Eq 2)}$$

2.3 Secondary Estimations

While the MATLAB script was in development, another set of estimations were conducted for the third presentation's mathematical modelling section. These calculations are summarized below, and their guiding assumptions were that all three driving motors were being utilized, and the robot was static after movement, since all three motors would be employed in such a scenario. A winch diameter of 40mm was assumed, as was a distance between the robot and the anchors being 46.6 inches, both of which were derived from Justin's speed calculations for the same presentation. The last guiding assumptions were that the cables were coming out of the top of the robot, meaning that the forces would be at an angle, and that the driving motors would experience the maximum amount of torque when the robot was extended to its maximum height. These calculations were conducted at multiple locations the robot could move to in an effort to locate the highest torque needed, with the cable lengths and angles being provided by Justin or by trigonometry. The "Top," "Right," and "Left" indicators next to the answers below indicate the cable they belong to, relative to looking down from above the table.

For the robot being on the edge of a 6-inch radius area of play, with 2 cables being at equal lengths between the robot and the respective anchors (Determined to be where the highest torque will be experienced in the designated area of play):

$r = 20mm$ - Radius of the new winch

$L_{Top} = L_{Right} = 45.895\ in$ - Length of the cables on the top and right sides of the robot

$L_{Left} = 54.599\ in$ - Length of the cable on the left of the robot

$L_z = 19\ in$ - Vertical component of the cable's length

$h = 19\ in$ - Maximum height of the driving winches when the robot is fully extended

$\theta_1 = 133°$ - Angle between horizontal components of the top and right cables

$\theta_2 = 113.5°$ - Angle between horizontal components of the top and left cables

$\theta_3 = 113.5°$ - Angle between the horizontal components of the left and right cables

$F = 10N$ - Tension of the cables

$\emptyset$ - Angle between the cables and the horizontal plane

L_{xy} - Horizontal component of the cable's length

$$\emptyset = \sin^{-1}\left(\frac{h}{L}\right) = 24.456°\ (Top);\ 24.456°\ (Right);\ 20.365°\ (Left) \quad \text{(Eq 3)}$$

$$F_{xy} = F * \cos(\emptyset) = 9.103N\ (Top);\ 9.103N\ (Right);\ 9.375N\ (Left) \quad \text{(Eq 4)}$$

$$F_z = F * \sin(\emptyset) = 4.140N\ (Top);\ 4.140N\ (Right);\ 3.480N\ (Left) \quad \text{(Eq 5)}$$

$$\tau_{xy} = F_{xy} \times r = 0.182\ Nm\ (Top);\ 0.182\ Nm\ (Right);\ 0.188\ Nm\ (Left) \quad \text{(Eq 1)}$$

$$\tau_z = F_z \times r = 0.0828\ Nm\ (Top);\ 0.0828\ Nm\ (Right);\ 0.0696\ Nm\ (Left) \quad \text{(Eq 1)}$$

$$\tau_{max} = \sqrt{\tau_{xy}^2 + \tau_z^2} = 0.2005\ Nm\ (Left) \quad \text{(Eq 6)}$$

The maximum torque found for this area of play was 0.2005 Nm, and was experienced in the left motor. The top and right motors both experience a torque just below that, with a value of 0.19995 Nm. These torques, when rounded to the nearest thousandth, are essentially equal, being 0.2 Nm. A diagram of this location and the robot's setup are added below, and were used as validation for this, showing an aerial view of the robot, with its initial position in grey and the edge of the area of play being shown with the darker sketch.

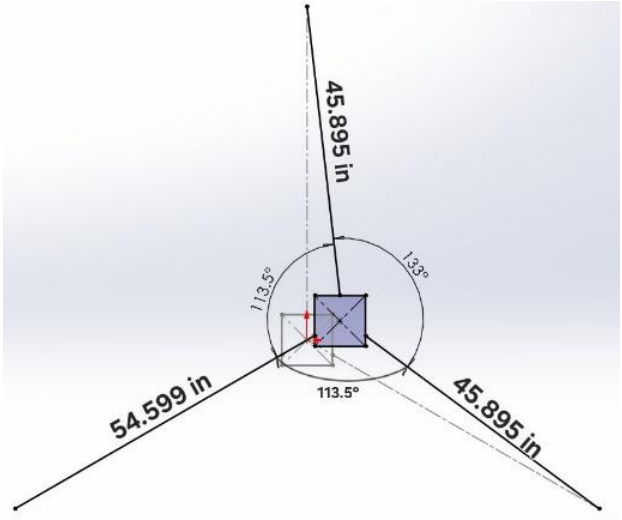

Figure 1: Cable length and angles when robot is at the edge of its area of play

3. Final Torque Analysis

3.1 MATLAB Script

The goal of this MATLAB script was to create an iterative loop that would designate a series of points the robot could move to within an area of play, and then calculate the change in each of its cable lengths, calculate the forces, and find the torques experienced by each motor. The code for this is included in Appendix A. The primary change in assumptions for this code was that the cables would be coming out of the bottom half of the robot and would be level with its connection to the anchors, eliminating any vertical components of force and torque.

The first primary feature of this code was setting up the robot. It started by establishing the robot's origin in vector form, then, after designating how far away each anchor was to be from the robot, it established the three anchor points, evenly spaced, in vector form. The area of play

was then established in the form of x and y coordinates. The code first plotted the three anchor points and the boundaries created by them. The loop containing the other calculations also included a line of code that would move the robot in 10mm intervals, plotting each location onto the graph to visualize where the robot moves. This is shown in figure 2 below.

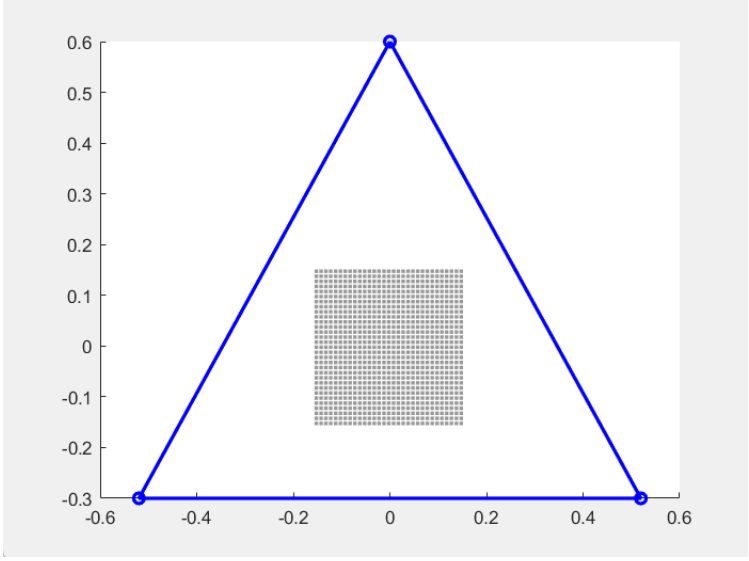

Figure 2: Anchors, area boundaries, and robot locations along the horizontal plane

The maximum force was defined to be 20N, which was defined by Dr. Razavian, as he wanted to ensure the team could find motors that would be suitable for greater applied forces than the required maximum force, for safety purposes. It also served another purpose, that being to ensure the motors used are not going to constantly run at their maximum capacity, which would damage the motors.

During a client meeting, Dr. Razavian also specified that the 20N of applied force would be from the user, not the cables, and instead the code needed to include the force having the ability to be applied in all directions, and the torques calculated would be the torques required for the robot to move against that applied force. Ryan assisted with this portion of the code, and helped incorporate the following lines of code into the MATLAB script. This allowed the force to be split into its x and y components, and allowed for those directions to change each iteration.

```
rad(i) = ((i-1)*0.01)*pi();

Fx(i) = cos(rad(i))*F;

Fy(j) = sin(rad(i))*F;

desiredForce(:,i,j) = [Fx(i);Fy(j);0];
```

The lines in the code following this calculate the change in cable length between the anchor points and the robot, calculate the unit vector for each cable, and then use the unit vectors to apply a modifier that will help solve for the forces. These steps were specified by Dr. Razavian during a client meeting. Following these steps, the modifiers were used with the “desiredForce”

variable from above, used with the "lsqnonneg" function to eliminate negative force values and find the real forces experienced in each cable. The calculated forces were used with a winch radius of 5mm to find the torques experienced by each motor at each location. Separate equations were used to separate the torques from the primary table into tables corresponding to each motor, for better visibility. The code also found the maximum torque, minimum torque, and summed the three torques in each position to plot on a surface plot. The surface plot is found in figure 3 in section 3.3. The code is not entirely perfect, but Dr. Razavian stated that it was good enough for the purposes the team needed it for and to move forward with what we have.

3.2 Dr. Razavian's MATLAB Script

Dr. Razavian created his own script as well and shared it with the team to assist us with our analysis. This code can be found in Appendix B. The code he wrote was far more complex, and included more calculations, including motor speed calculations. This proved to be helpful in this analysis and provided much needed direction to improve the team's code. He also instructed us to use this code alongside the team's code to find an answer for the torque each motor experiences to select a motor that will meet the expected torque.

3.3 Comparison of Results

The code I wrote to calculate the torque applied to the motors produced an average of around 0.10-0.13 Nm, with a maximum torque of 0.131 Nm. These are the results when the anchor radius is 0.6m away from the robot. The torque values change when the anchor radius changes, but 0.6m was used to compare the two code's results, since that was the value used in Dr. Razavian's code. The torque from all three motors were summed for each position and plotted on a surface plot, shown in figure 3.

Dr. Razavian's code produced an average torque of about 0.1-0.2 Nm, with a maximum torque of around 0.3 Nm. Dr. Razavian also plotted his results on a surface plot, which is shown in figure 4.

Comparing the results, the torque experienced by the motor ranges between 0.1-0.2 Nm of torque on average. Dr. Razavian's code calculates the motor torque for the robot's location within the entire triangle boundary created by the anchor points, while ours only calculates the torque within a small area within the boundary. Along the edges of the surface plot provided to us by Dr. Razavian, the torque spikes by the edges of the boundaries, which is where we see the torque range of 0.22 and above. The locations where these torque values are found are locations that would either be very difficult for the robot to get to, or would be impossible for it to move there, as the triangle created by the three anchor points are not the true boundary of the robot's movement, since it has a maximum angle the wires can reach, which is below 180 degrees. Therefore, the values along the edges can be largely ignored. Comparing both the data tables produced by MATLAB and the surface plots, a large majority of the torque values fall between 0.10-0.15 Nm, with many values being smaller than that range. Therefore, the maximum torque the motors would realistically experience would be around 0.2 Nm, with the nominal torque being much less than that.

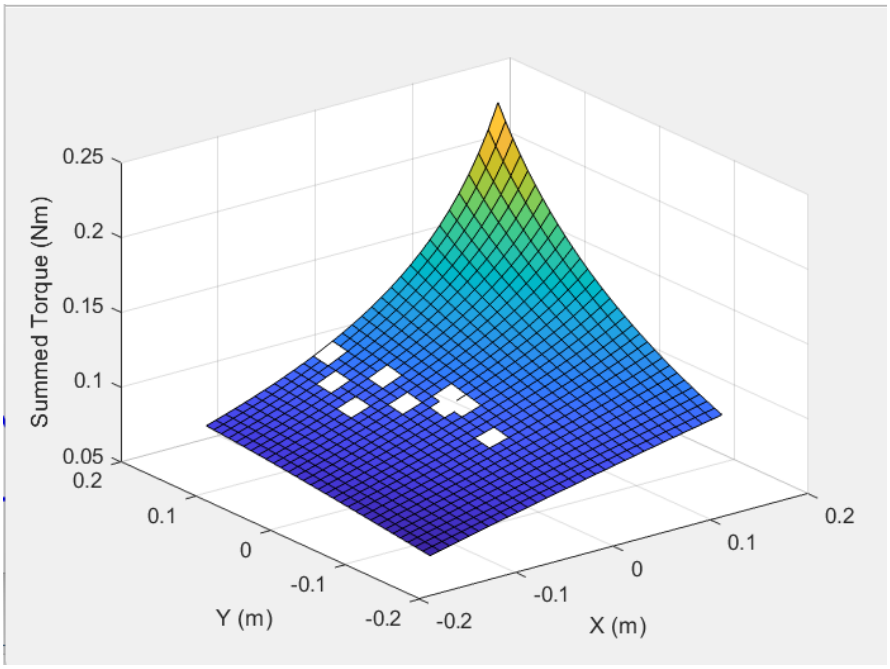

Figure 3: Surface plot of summed torques from the team's code

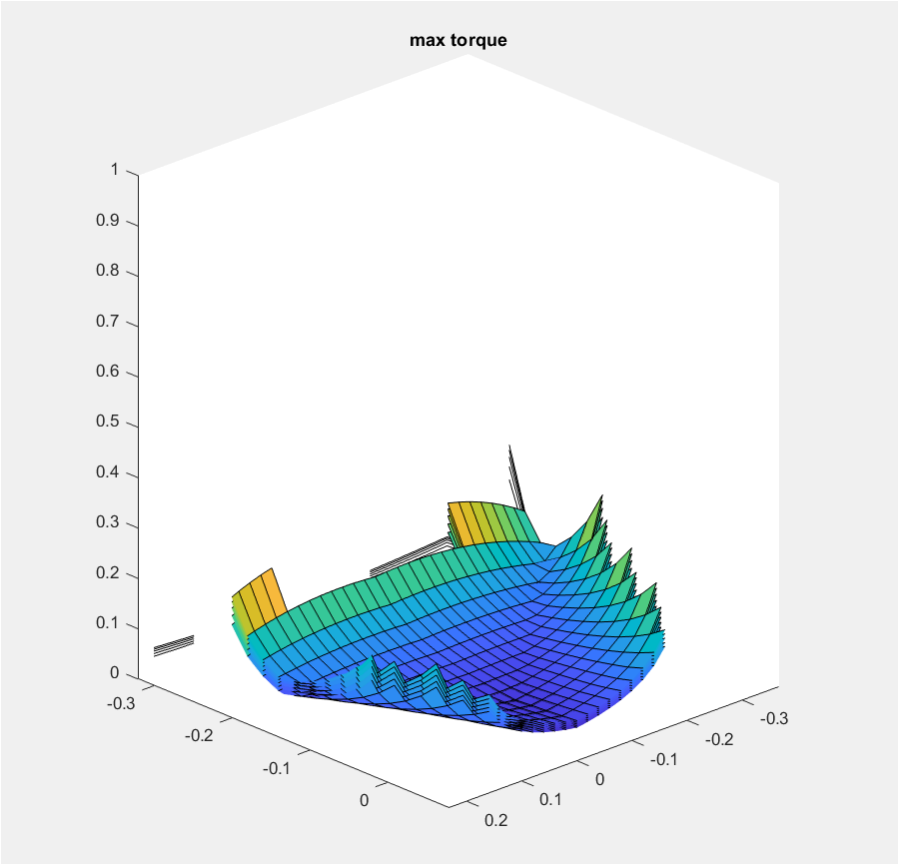

Figure 4: Surface plot of summed torques from Dr. Razavian's code

The surface plot our code produced would be around the center of the surface plot Dr. Razavian generated. Since both are around 0.1Nm or under, it is reasonable to assume that the nominal torque can be lower than the predicted torque.

4. Conclusion

The results from this analysis show that the maximum torque for the driver motors is about the same as the torque that was initially estimated. Dr. Razavian has expressed that he wants gimbal motors to be used as our driver motors, and using the results from this analysis, he has requested that the team choose a motor that has a nominal torque of 0.15Nm. As for the maximum torque, he requested that the team choose the maximum value, which has been chosen as a range between 0.2 and 0.25 Nm, with the other specifications taking priority, so as long as the maximum torque falls within that range with the rest of the specifications meeting the other requirements, that motor will be deemed appropriate for use in this project. The robot will also be incorporating a fourth motor, which will be driving the two lead screws to raise and lower the robot. There's no analysis for that motor, because Dr. Razavian has already given us the specifications to use when researching motors. The nominal torque and RPM values he has instructed the team to use are 0.15 Nm and 6000 RPM, respectively. The team is researching gimbal motors and other motor types that will provide the necessary torque and speed required, while not drawing a significant amount of power. There have been several promising discoveries that will be reviewed in the next meeting. Moving forward, the team will be running extensive tests on any motor that is ordered, and if the torque requirements change, these results will be updated accordingly.

Appendix A: Team's Motor Torque MATLAB Script

```
clear; clc; close all;
% Robot Position
R = [0;0;0]; % Robot Starting Postion
R0 = R.*[1;1;0]; % Robot Base Position
% Winch Radius:
r = 0.005; % m
% Magnitude of Desired Force:
F = 20; % N

% Anchor point locations
% AnchorCircleRadius = 0.4572; % m (18in radius)
% AnchorCircleRadius = 0.6096; % m (24in radius)
AnchorCircleRadius = 0.6;
A1 = AnchorCircleRadius*[cosd(90); sind(90); 0]; % Top Anchor
A2 = AnchorCircleRadius*[cosd(210); sind(210);0]; % Left Anchor
A3 = AnchorCircleRadius*[cosd(-30); sind(-30);0]; % Right Anchor

X = -0.1524:0.01:0.1524;
Y = -0.1524:0.01:0.1524;
Z = 0;

TorqueSurface = zeros(length(Y), length(X));

hold on
plot([A1(1),A3(1),A2(1),A1(1)],[A1(2),A3(2),A2(2),A1(2)],'bo-','LineWidth',2);
plot([R(1)],R(2))


for i=1:length(X)
```

```
for j=1:length(Y)
    R = [X(i);Y(j);Z];
    plot(R(1), R(2), '.', 'Color', [0.6 0.6 0.6]);

    % User applied force
    rad(i) = ((i-1)*0.01)*pi();
    Fx(i) = cos(rad(i))*F;
    Fy(j) = sin(rad(i))*F;
    desiredForce(:,i,j) = [Fx(i);Fy(j);0];

    V1 = A1-R;
    u1 = V1/norm(V1);
    V2 = A2-R;
    u2 = V2/norm(V2);
    V3 = A3-R;
    u3 = V3/norm(V3);
    c1 = u1.*(1+[1;1;0]);
    c2 = u2.*(1+[1;1;0]);
    c3 = u3.*(1+[1;1;0]);
    c4 = [0;0;1];
    C = [c1 c2 c3 c4];

    wireForces(:,i,j) = lsqnonneg(C,desiredForce(:,i,j));
    torques = wireForces(:,i,j).*r;
    torque1 = torques(1);
    torque2 = torques(2);
    torque3 = torques(3);
    totaltorque = torque1+torque2+torque3;
    TorqueSurface(j,i) = norm(totaltorque);

    maxTension = max(wireForces(:,i,j));
```

```
        minTension = min(wireForces(:,i,j));
        maxTorque = r.*maxTension;

    end
end
V = 1; % m/s

RPM = (V*60)/(pi()*r);

[Xgrid, Ygrid] = meshgrid(X, Y);
figure(2);
surf(Xgrid, Ygrid, TorqueSurface);
xlabel('X (m)');
ylabel('Y (m)');
zlabel('Summed Torque (Nm)');
```

Appendix B: Dr. Razavian's Analysis Code

```
%%
clearvars

v = 1; % desired speed
a = 2; % desired acceleration
F = 20; % magnitude of the desired force

T = 0; % external rotating moment on robot;
Omega = 0; % rotational velcoity of the robot

phaseshift = pi; % the angle between robot's acceleration and hand force. maximum torques at pi

AnchorCircleRadius = 0.6;
A1 = AnchorCircleRadius*[cosd(90); sind(90); 0];
A2 = AnchorCircleRadius*[cosd(210); sind(210);0];
A3 = AnchorCircleRadius*[cosd(-30); sind(-30);0];

RobotCircleRadius = 0.1;
C1_u = [RobotCircleRadius*[cosd(90);  sind(90)] ; 0.1]; % the anchor points relative to robot's
handle
C2_u = [RobotCircleRadius*[cosd(210); sind(210)]; 0.1];
C3_u = [RobotCircleRadius*[cosd(-30); sind(-30)]; 0.1];

R = [0;0;0.1]; % robot's handle position.

r_pulley = 0.005; % radius of of pulley
screwAngle = 0.01; % some sort of pitch, like tangent of screw angle.
```

```
hf999 = figure(999);
clf(hf999)
plot_FSquirrleBases(R,C1_u, C2_u, C3_u, A1, A2, A3, figure(999))



%%
m = 5;

gearRatio = 1;

phiRange = 0:0.05:2*pi;

xRange = -0.3:0.02:0.3;
yRange = -0.3:0.02:0.1;
zRange = 0.1:0.1:0.5;



[tau_F_only_max,tau_F_only_min,omega_pulley_max,omega_pulley_min] = ...
   deal(nan(length(xRange),length(yRange),length(zRange)));

for i = 1:length(xRange)
   for j = 1:length(yRange)
      for k = 1:length(zRange)
         R = [xRange(i); yRange(j); zRange(k)];


         J_inv =
FSquirrle_Jacobian_inverse(R(1),R(2),R(3),A1(1),A1(2),A2(1),A2(2),A3(1),A3(2),C1_u(1),C1_u(2),C
1_u(3),C2_u(1),C2_u(2),C2_u(3),C3_u(1),C3_u(2),C3_u(3),r_pulley);
         J =
FSquirrle_Jacobian(R(1),R(2),R(3),A1(1),A1(2),A2(1),A2(2),A3(1),A3(2),C1_u(1),C1_u(2),C1_u(3),C
2_u(1),C2_u(2),C2_u(3),C3_u(1),C3_u(2),C3_u(3),r_pulley);
```

```
    [B,~,B_l] =
FSquirrle_bases(R(1),R(2),R(3),A1(1),A1(2),A2(1),A2(2),A3(1),A3(2),C1_u(1),C1_u(2),C1_u(3),C2_
u(1),C2_u(2),C2_u(3),C3_u(1),C3_u(2),C3_u(3));


    if min([dot(B_l(:,1),B_l(:,2)) , dot(B_l(:,1),B_l(:,3)) , dot(B_l(:,2),B_l(:,3))])<cosd(165)
       continue
    end


    omega_pulley = [];
    tau_F_only = [];
    % clf(hf999)
    % plot_FSquirrleBases(R,C1_u, C2_u, C3_u, A1, A2, A3, hf999)
    for phi = phiRange
       Fx = F*cos(phi);
       Fy = F*sin(phi);
       Fz = 0;


       handForce = -[Fx;Fy;Fz];


       lineForces = lsqnonneg(B,handForce);
       tau_F_only = [tau_F_only, lineForces.*[r_pulley; r_pulley; r_pulley; screwAngle]];


       % ax = a*cos(phi+phaseshift);
       % ay = a*sin(phi+phaseshift);
       % robotAcc = [ax;ay;0];
       % tau_a_only = [tau_a_only, J' * (M*robotAcc)];
       %
       vx = v*cos(phi);
       vy = v*sin(phi);
       vz = 0;


       V = [vx;vy;vz];
```

```
            omega_pulley = [omega_pulley, J_inv*V];
        end

        omega_pulley_max(i,j,k) = max(omega_pulley(:));
        omega_pulley_min(i,j,k) = min(omega_pulley(:));
        tau_F_only_max(i,j,k) = max(tau_F_only(1:3,:),[],'all');
        tau_F_only_min(i,j,k) = min(tau_F_only(1:3,:),[],'all');
      end
   end
end

%%
% tau_combined = tau_a_only + tau_F_only;

figure(1); clf
[xx,yy] = meshgrid(xRange,yRange);

for i = 1:size(tau_F_only_max,3)
   subplot(1,2,1)
   surface(xx, yy, tau_F_only_max(:,:,i)'/gearRatio)
   hold all
   zlim([0,1])
   title('max torque')

   subplot(1,2,2)
   surface(xx, yy, tau_F_only_min(:,:,i)'/gearRatio)
   hold all
   zlim([-1,0])
   title('min torque')

end
```

```
%%
figure(2); clf
for i = 1:size(omega_pulley_max,3)
   subplot(1,2,1)
   surface(xx, yy, omega_pulley_max(:,:,i)'/gearRatio)
   hold all
   title('max motor speed')

   subplot(1,2,2)
   surface(xx, yy, omega_pulley_min(:,:,i)'/gearRatio)
   hold all
   title('min motor speed')

end

max(omega_pulley)
```

References:

[1] K. Smith, "Preventing Torque Overloads with Mechanical Limiters," *Machinedesign.com*, Aug. 07, 2020. https://www.machinedesign.com/automation-iiot/article/21138651/preventing-torque-overloads-with-mechanical-limiters

[2] "Motor Sizing Calculations," *Oriental Motor U.S.A. Corp.* https://www.orientalmotor.com/technology/motor-sizing-calculations.html