Hopp3r: a Planar 3 Degree of Freedom Hopping Robot

Alex Friedman
Zen Iwankiw
Dan Lynch
Greg Niederschulte
Andrew SaLoutos
Suhail Pallath Sulaiman
Zidong Xiao

June 12, 2018

Contents

Chapter 1

Project Definition

1.1 Motivation

Our sponsor, Dr. Paul Umbanhowar, has asked us to design a hopping robot for a NASA project that investigates legged locomotion on soft ground. Soft ground is a new frontier in robotic locomotion. It underlies many uses for robots, such as disaster response, search and rescue, military ground support, earth science, and extraterrestrial exploration. Given the multitude of legged animals that traverse dirt, sand, and snow with relative ease, legged robots are a promising alternative to wheeled or treaded robots, which often get stuck in or skid on these types of granular media.

To date, most robophysical studies of locomotion on/in granular media have examined purely vertical intrusion, with the exception of work on quasi-static penetration with angled flat plate intruders by Li et al. [2]. Real-world locomotion often requires horizontal foot motion and rotation during stance as well as vertical motion, so our team has developed a new robot, named Hopp3r, to research planar legged locomotion in GM.

1.2 Specifications

1.2.1 Kinematic specifications

1. The robot must have 3 coplanar degrees of freedom, all of which must be actuated. Hopp3r takes its name from these three degrees of freedom.
2. The robot must have a maximum extended height no greater than 40 cm.

1.2.2 Dynamic specifications

1. The robot must be able to jump in place and locomote forward/backward.
2. If the center of mass (CoM) height when the leg is fully extended is X cm above the ground, then the CoM must reach at least X + 10 cm when hopping continuously on rigid ground.
3. In steady state locomotion, the robot must have a maximum 25% stance duty cycle
4. The robot can *optionally* use a reaction wheel or other appendage to control body orientation.
5. The robot must be able to balance during stance by using the three actuators to control the pose of the body.
6. The robot must be capable of a standing broad jump (starting from rest and coming to rest without jumping again) of at least 15 cm.
7. The robot must be capable of a soft landing, using control to minimize the maximum force after impact.

1.2.3 Other mechanical specifications

1. The robot should weigh less than 5 kg.
2. The leg must accept different feet. For hard ground, the foot should be a hemisphere made of soft rubber or other soft, durable material.

1.2.4 Boom specifications

1. The robot must be mounted on a boom by a lockable revolute joint located approximately at the robot's center of mass.
2. The boom must have two rotational degrees of freedom (pitch and yaw). Yaw must be lockable.
3. The boom must be 1.2 - 1.5 m long.
4. The boom must not deflect more than 0.5% of its length when holding the robot at rest above ground.
5. The boom mass should be $< 10\%$ of the robots mass.
6. The boom must have an attachment point for counterweights.
7. The boom counterweight system, when unloaded, must only counter the weight of the boom. This requirement only applies to the performance tests: 1.2.2.2 (jump height) and 1.2.2.6 (broad jump).

1.2.5 Electromechanical specifications

1. Power can be offboard, provided by an umbilical cord (through boom).
2. Actuators must be onboard.
3. Amplifiers may be onboard or offboard.

1.2.6 Sensor specifications

1. The revolute joint connecting the robot to the boom must be equipped with a rotary encoder.
2. Each boom joint must be equipped with a rotary encoder (one for sensing pitch and one for sensing yaw).
3. Each of the robot's actuated joints must be equipped with an encoder.
4. The robot must be equipped with a force sensor that measures forces generated in the foot.
5. The robot must be equipped with an IMU to measure acceleration of the foot.
6. Each actuator must be equipped with a temperature sensor.

1.2.7 Analysis and simulation

1. Develop a dynamic model of both phases of the robot (stance & flight)
2. Develop a simulation of the robot using the dynamic model.

Chapter 2

Team Members

Sponsor: Dr. Paul Umbanhowar
Team Members:

- Alex Friedman
- Zen Iwankiw
- Dan Lynch
- Greg Niederschulte
- Andrew SaLoutos
- Suhail Pallath Sulaiman
- Zidong Xiao

Team Leader: Andrew
Mechanical Engineering: Alex, Zen, Suhail, Andrew, Greg
Production Engineering: Alex, Zen, Greg
Electrical Engineering: Dan, Tom, Andrew
Software Engineering: Dan, Suhail, Tom, Andrew

Chapter 3

System Diagrams

3.1 Mechanical

3.2 Electrical

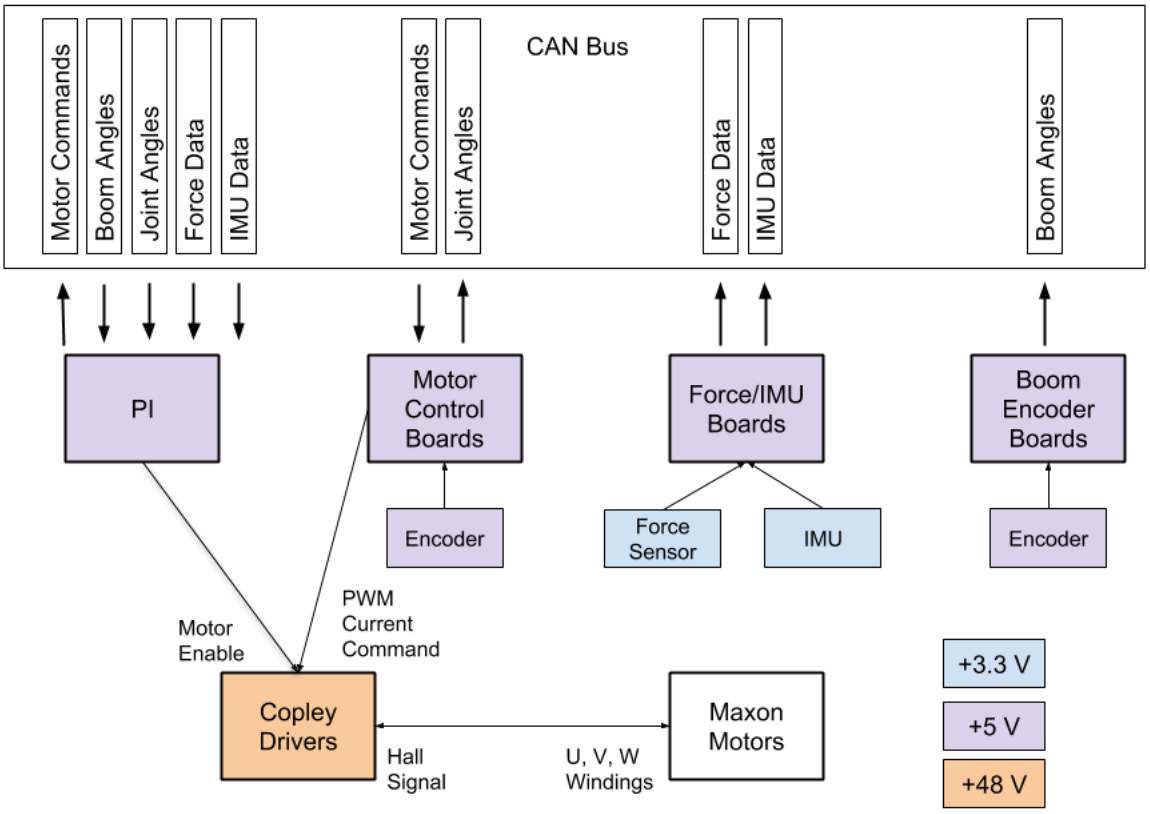

Figure 3.1: Block diagram of Hopp3r's electrical system.

3.3 Software

Chapter 4

Engineering Analyses

4.1 Kinematics

4.1.1 Motivation and mobility

Hopp3r is a closed-chain robot designed to study legged locomotion. Such a robot is "stronger" than a serial robot because closed kinematic chains allow multiple actuators to contribute to the motion of an end-effector. Additionally, this closed-chain legged robot has more mass concentrated at its top, compared to an open-chain robot, which is advantageous for legged locomotion for two reasons:

- Less mass near the foot increases agility and reduces actuation requirements at the hip.
- The robot behaves like an inverted pendulum during stance, so concentrating the mass at the top of the pendulum increases stability.

The remainder of this chapter derives a kinematic model of a particular planar 3-degree-of-freedom (DOF) closed-chain robot. Having 3 DOF is desirable because a planar legged robot with 3 DOF can fully control its position and orientation during stance and flight.[1]

These analyses closely follow [3], especially Chapters 6 ("Inverse Kinematics") and 7 ("Kinematics of Closed Chains").

We now provide a verification that the robot shown in Figure 4.1 has 3 degrees of freedom, using Grübler's formula:

$$\text{DOF} = m(N - 1 - J) + \sum_{i=1}^{J} f_i, \tag{4.1}$$

where $m = 3$ for planar mechanisms, N is the number of links (including the base), J is the number of joints, and f_i is the number of freedoms at the i^{th} joint. From Figure 4.1, $N = 8$, $J = 9$, and $f_i = 1$ for all i. Note that the lowest joint in the figure is really two joints; one joint connects the left coupler to the middle link, and the other connects the right coupler to the middle link. Thus, according to Grübler's formula, the robot has 3 degrees of freedom.

4.1.2 Forward kinematics

Closed-chain linkages can be decomposed into groups of open-chain linkages (hereafter referred to as "open sub-chains") whose motions are constrained by loop-closure equations. Our robot comprises three open sub-chains, shown in Figure 4.2[2]: the θ-chain is on the left, the ϕ-chain is in the middle, and the ψ-chain is on the right. We will first examine the forward kinematics for each open sub-chain and then develop loop-closure equations.

[1] The interface between the foot and ground can be viewed as an unactuated joint, inviting questions about underactuation and controllability for this particular robot.

[2] Note: this drawing depicts the three actuated joints as collinear, but the kinematic equations do not make this assumption.

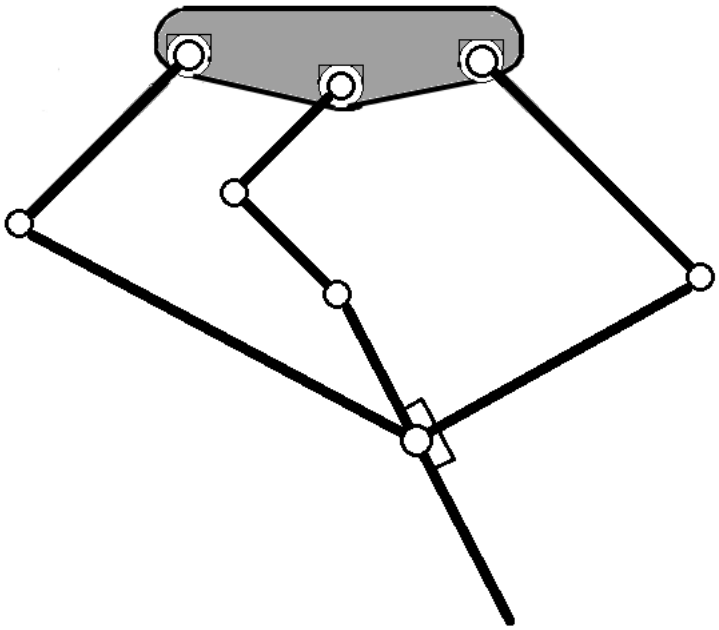

Figure 4.1: Schematic diagram of the planar robot.

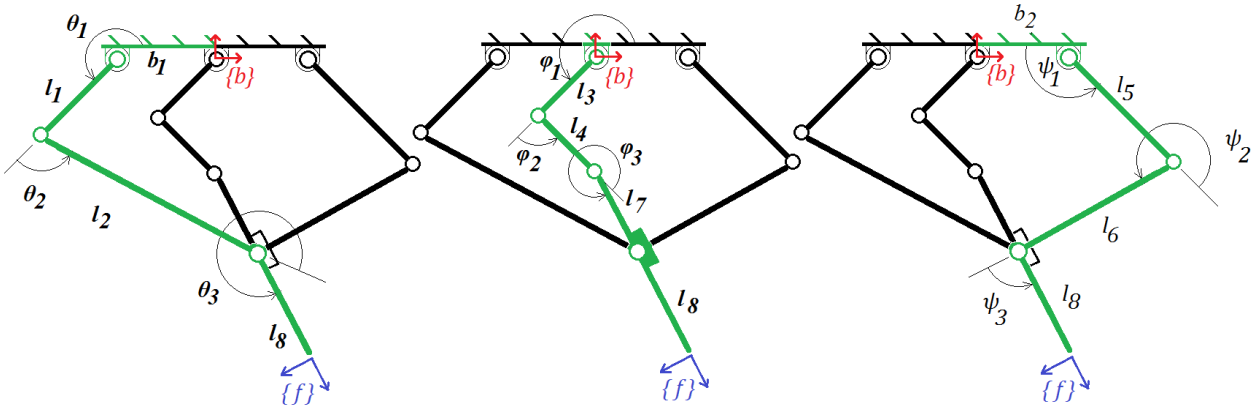

Figure 4.2: The planar robot comprises three open sub-chains, referred to as the θ-, ϕ-, and ψ-chains, from left to right.

Open sub-chain forward kinematics

The forward kinematics of each open sub-chain relate the foot frame $\{f\}$ to the base frame $\{b\}$. The forward kinematics of the θ-chain are

$$\begin{bmatrix} x_f \\ y_f \\ \measuredangle_f \end{bmatrix} = \begin{bmatrix} -b_{1,x} + l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_8 \cos(\theta_1 + \theta_2 + \theta_3) \\ b_{1,y} + l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_8 \sin(\theta_1 + \theta_2 + \theta_3) \\ \theta_1 + \theta_2 + \theta_3 \end{bmatrix}. \tag{4.2}$$

The forward kinematics of the ϕ-chain are

$$\begin{bmatrix} x_f \\ y_f \\ \measuredangle_f \end{bmatrix} = \begin{bmatrix} l_3 \cos\phi_1 + l_4 \cos(\phi_1 + \phi_2) + (l_7 + l_8) \cos(\phi_1 + \phi_2 + \phi_3) \\ l_3 \sin\phi_1 + l_4 \sin(\phi_1 + \phi_2) + (l_7 + l_8) \sin(\phi_1 + \phi_2 + \phi_3) \\ \phi_1 + \phi_2 + \phi_3 \end{bmatrix}. \tag{4.3}$$

Lastly, the forward kinematics of the ψ-chain are

$$\begin{bmatrix} x_f \\ y_f \\ \measuredangle_f \end{bmatrix} = \begin{bmatrix} b_{2,x} + l_5 \cos\psi_1 + l_6 \cos(\psi_1 + \psi_2) + l_8 \cos(\psi_1 + \psi_2 + \psi_3) \\ b_{2,y} + l_5 \sin\psi_1 + l_6 \sin(\psi_1 + \psi_2) + l_8 \sin(\psi_1 + \psi_2 + \psi_3) \\ \psi_1 + \psi_2 + \psi_3 \end{bmatrix}. \tag{4.4}$$

Thus, if we know the joint positions of any of the three open sub-chains, we can calculate the end-effector pose. Although this formulation is straightforward, it is not particularly useful on its own, because both the actuated and unactuated joint positions in that chain must be known in order to calculate the end-effector pose. It would be more useful to have a mapping from the positions of only the actuated joints to the end-effector pose.

Geometric forward kinematics

One way to determine the foot pose $(x_f, y_f, \measuredangle_f)$ from the actuated joint positions $q_a = (\theta_1, \phi_1, \psi_1)$ is to realize that each rotating link sweeps out a circle, so any two open sub-chains meet at a point that is the intersection of two circles (in the case of a spatial robot, these points are the intersections of spheres instead of circles). As shown in Figure 4.3, there are two intersections to consider:

- (x_A, y_A): the intersection of the θ-chain and the ψ-chain, and
- (x_{uA}, y_{uA}): the intersection of the ϕ-chain and the most distal link

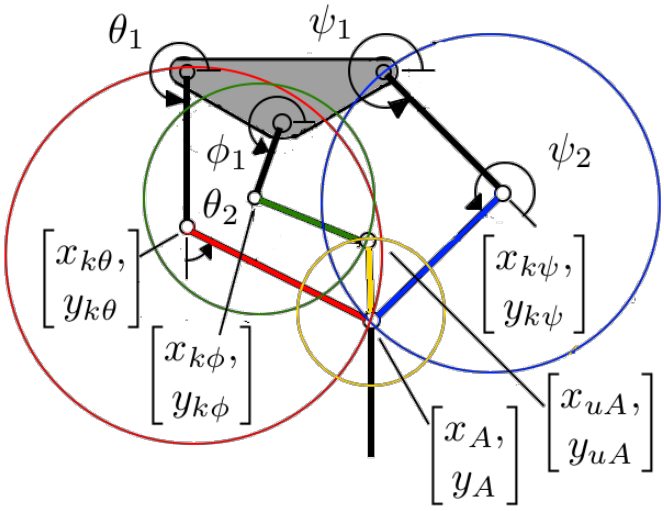

Figure 4.3: The forward kinematics can be solved by considering the intersection of circles.

Using the link lengths and angle conventions defined earlier, the first of these intersection points, (x_A, y_A), can be calculated:

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} \frac{b}{a}(x_{k\psi} - x_{k\theta}) \pm \frac{c}{a}(y_{k\psi} - y_{k\theta}) + x_{k\theta} \\ \frac{b}{a}(y_{k\psi} - y_{k\theta}) \mp \frac{c}{a}(x_{k\psi} - x_{k\theta}) + x_{k\theta} \end{bmatrix}, \text{ where} \tag{4.5}$$

- $(x_{k\theta}, y_{k\theta})$, $(x_{k\phi}, y_{k\phi})$, and $(x_{k\psi}, y_{k\psi})$ are calculated from $q_a = (\theta_1, \phi_1, \psi_1)$,
- $a = \sqrt{(x_{k\theta} - x_{k\psi})^2 + (y_{k\theta} - y_{k\psi})^2}$,
- $b = \frac{l_2^2 - l_6^2 + a^2}{2a} = \frac{l_2}{l_6}$ (for symmetric links, i.e. when $l_2 = l_6$), and

- $c = \sqrt{l_2^2 - b^2}$.

The same approach can be applied to find the other intersection point, (x_{uA}, y_{uA}). Combining these steps gives Algorithm 1, a compact geometric forward kinematics algorithm:

Algorithm 1 Analytic geometric closed-chain forward kinematics

function GEOMFK(q_a)
  Given θ_1 and ψ_1, solve for (x_A, y_A).
  Given ϕ_1 and (x_A, y_A), solve for (x_{uA}, y_{uA}).
  Given (x_A, y_A) and (x_{uA}, y_{uA}), solve for $(x_F, y_F, \measuredangle_F)$.
  Use subchain inverse kinematics to solve for q_u.
  return $T_{bf} = (x_f, y_f, \measuredangle_f)$ and q_u.
end function

Observe that equation 4.5 has one $\pm$ and one $\mp$ operator. Currently, these are implemented as `switch` statements, but they introduce jump discontinuities into the forward kinematic equations. Given the limited workspace of the robot, these discontinuities are encountered very rarely, but they warrant further analysis and possibly a more considered approach to the forward kinematics problem.

Numerical forward kinematics

Given $q_a \in \mathbb{R}^3$, the loop-closure equations can be solved directly to obtain $q_u \in \mathbb{R}^6$. Then, once all the joint positions are known, we can choose any of the three open sub-chains and use the corresponding forward kinematics equations to calculate the end-effector pose.

This approach requires solving a system of nonlinear equations (with potentially many solutions), so a numerical root-finding algorithm may be more appropriate than seeking an analytical solution. The simplest such algorithm is the Newton-Raphson method.

Consider a nonlinear equation $g(q) = \ldots$, for which we seek the roots, i.e., we wish to solve $g(q) = 0$. Linearize the equation by writing a Taylor series expansion of $g(q)$ about the point q_0:

$$g(q) = g(q_0) + \left.\frac{\partial g}{\partial q}\right|_{q_0} (q - q_0) + \text{h.o.t.} \tag{4.6}$$

Discarding the higher-order terms, the roots are approximately

$$q = q_0 - \left(\left.\frac{\partial g}{\partial q}\right|_{q_0}\right)^{-1} g(q_0) \quad . \tag{4.7}$$

Taking the new q as q_0, this method can be iterated until some convergence threshold criterion is satisfied.

It turns out that $\frac{\partial g}{\partial q}$ is the *constraint Jacobian* (derived in the next section), which shows up often in the analysis of closed kinematic chains. We can use it to develop Algorithm 2, an iterative numerical forward kinematics algorithm.

Algorithm 2 Newton-Raphson closed-chain forward kinematics

function NRFK(q_a, q_u^0, ε)
  Initialize $r > \varepsilon$ ▷ measure of convergence
  while $r > \varepsilon$ **do** ▷ convergence threshold
    $q_u^{new} = q_u^0 - J_c^{-1} g\left(q_a, q_u^0\right)$
    $r = \frac{\left|g(q_a, q_u^0) - g(q_a, q_u^{new})\right|}{|g(q_a, q_u^0)|}$ ▷ update measure of convergence
    $q_u^0 = q_u^{new}$
  end while ▷ we now have approximate values for q_u
  $\theta = \begin{bmatrix} q_a(1) & q_u^{new}(1) & q_u^{new}(2) \end{bmatrix}^{\mathrm{T}}$
  return $\begin{bmatrix} x_f & y_f & \measuredangle_f \end{bmatrix}^{\mathrm{T}} = T_{bf}(\theta)$ ▷ $T_{bf}(\theta)$ is the end-effector pose computed using the θ-chain
end function

In summary, this algorithm finds approximate unactuated joint positions from the actuated joint positions and uses the positions along one open sub-chain to compute the end-effector pose. Note that this algorithm uses the θ-chain to compute the end-effector pose, but the other open sub-chains are equally viable.

Pros and Cons of `geomFK` and `NRFK`

Each of the two forward kinematics methods described above has strengths and weaknesses, especially when considering their implementation on an embedded system. Iterative methods do not provide guaranteed execution times, so they are usually poor choices for real-time control. Additionally, matrix inversion is a computationally intensive task with unbounded execution time, so it is best avoided or at least replaced with LU decomposition.

Algorithm	Pros	Cons
`geomFK`	bounded execution time	jump discontinuities
`NRFK`	no jump discontinuities	unbounded execution time & matrix inversion

Based on these pros and cons, and considering that Hopp3r's relatively small workspace generally avoids jump discontinuities, we opted to implement `geomFK` instead of `NRFK`. Nevertheless, a better approach may exist, and this deserves exploration.

4.1.3 Inverse Kinematics

The inverse kinematics problem is to find joint angles that yield a particular end-effector pose (the foot pose, in the case of this robot). Whereas the forward kinematics for this robot (and closed-chain linkages in general) is difficult to compute, the inverse kinematics can be solved analytically, using basic geometry. In fact, for this particular robot, the foot pose $(x_f, y_f, \measuredangle_f)$ itself provides a lot of information: from it, the location of the most distal joint (the "ankle") , with respect to the base frame $\{b\}$, can be calculated as

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} x_f - l_8 \cos \measuredangle_f \\ y_f - l_8 \sin \measuredangle_f \end{bmatrix} , \tag{4.8}$$

and similarly, the location of the joint above (the "upper ankle") is

$$\begin{bmatrix} x_{uA} \\ y_{uA} \end{bmatrix} = \begin{bmatrix} x_f - (l_7 + l_8) \cos \measuredangle_f \\ y_f - (l_7 + l_8) \sin \measuredangle_f \end{bmatrix} . \tag{4.9}$$

These two points (the ankle and the upper ankle) are crucial to solving the inverse kinematics because they connect the open subchains, forming closed loops.

θ-chain inverse kinematics

Knowing the ankle's location, we can analytically solve for θ_1 and θ_2 by applying the same geometric inverse kinematics used with 2R robot arms. Figure 4.4 below displays most of the geometry relevant for computing θ_1 and θ_2.

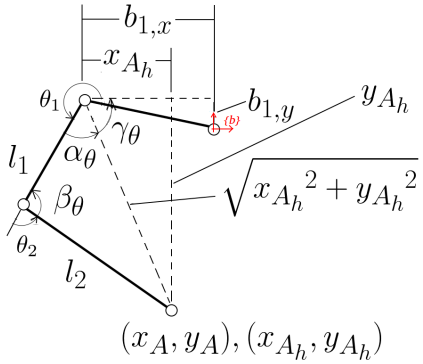

Figure 4.4: Geometric features relevant for computing inverse kinematics of the θ-chain

First, relative to $\{b\}$, the location of joint θ_1 (the "θ hip") is simply

$$\begin{bmatrix} x_{H,\theta} \\ y_{H,\theta} \end{bmatrix} = \begin{bmatrix} -b_1, x \\ b_1, y \end{bmatrix} .$$

To apply 2R inverse kinematics, we also need the location of the angle with respect to the θ hip:

$$\begin{bmatrix} x_{A_{h,\theta}} \\ y_{A_{h,\theta}} \end{bmatrix} = \begin{bmatrix} -x_{H,\theta} + x_A \\ y_{H,\theta} - y_A \end{bmatrix} .$$

If we draw a line connecting the ankle and the θ hip, the angle between this line and the base frame x-axis is simply

$$\gamma_\theta = \text{atan2}(y_{A_{h,\theta}}, x_{A_{h,\theta}}).$$

Now, the angle between this same line and link l_1 can be found using the Law of Cosines:

$$\alpha_\theta = \arccos\left(\frac{x_{A_{h,\theta}}^2 + y_{A_{h,\theta}}^2 + l_1^2 - l_2^2}{2l_1\sqrt{x_{A_{h,\theta}}^2 + y_{A_{h,\theta}}^2}} \right) ,$$

and the hip angle is

$$\theta_1 = -\gamma_\theta - \alpha_\theta . \tag{4.10}$$

The (x, y) location of the θ_2 joint (the "knee") can now be computed:

$$\begin{bmatrix} x_{K,\theta} \\ y_{K,\theta} \end{bmatrix} = \begin{bmatrix} x_{H,\theta} + l_1 \cos\theta_1 \\ y_{H,\theta} + l_1 \sin\theta_1 \end{bmatrix} .$$

We can also find θ_2 by applying the Law of Cosines again:

$$\beta_\theta = \arccos\left(\frac{l_1^2 + l_2^2 - x_{A_{h,\theta}}^2 - y_{A_{h,\theta}}^2}{2l_1 l_2} \right) ,$$

and the knee angle is simply

$$\theta_2 = \pi - \beta_\theta . \tag{4.11}$$

We can verify the inverse kinematics result by calculating the (x, y) location of the ankle using θ_1 and θ_2:

$$\begin{bmatrix} x_{A,\theta} \\ y_{A,\theta} \end{bmatrix} = \begin{bmatrix} x_{H,\theta} + l_1 \cos\theta_1 + l_2 \cos\theta_1 + \theta_2 \\ y_{H,\theta} + l_1 \sin\theta_1 + l_2 \sin\theta_1 + \theta_2 \end{bmatrix} .$$

Lastly, because we know θ_1, θ_2, and $\measuredangle_f$, we can find θ_3:

$$\theta_3 = \measuredangle_f - \theta_1 - \theta_2 . \tag{4.12}$$

ψ-chain inverse kinematics

The inverse kinematics of the ψ-chain mirror those of the θ-chain. Figure 4.5 below shows the relevant geometric features of the ψ-chain.

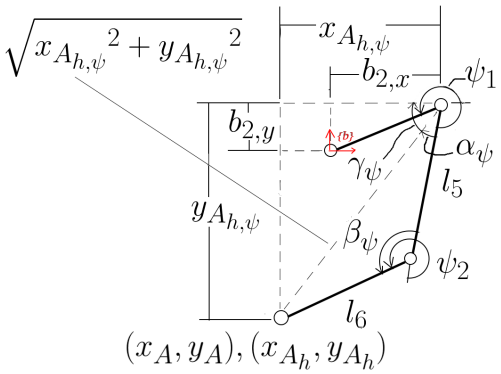

Figure 4.5: Geometric features relevant for computing inverse kinematics of the ψ-chain

In the $\{b\}$ frame, the origin of the ψ-chain is

$$\begin{bmatrix} x_{H,\psi} \\ y_{H,\psi} \end{bmatrix} = \begin{bmatrix} b_2, x \\ b_2, y \end{bmatrix} \ ,$$

and with respect to this point, the location of the "ankle" joint is

$$\begin{bmatrix} x_{A_{h,\psi}} \\ y_{A_{h,\psi}} \end{bmatrix} = \begin{bmatrix} x_{H,\psi} - x_A \\ y_{H,\psi} - y_A \end{bmatrix} \ .$$

The "hip" angle, ψ_1, is given by

$$\gamma_\psi = \text{atan2}(y_{A_{h,\psi}}, x_{A_{h,\psi}}),$$

$$\alpha_\psi = \arccos\left(\frac{x_{A_{h,\psi}}^2 + y_{A_{h,\psi}}^2 + l_5^2 - l_6^2}{2l_5\sqrt{x_{A_{h,\psi}}^2 + y_{A_{h,\psi}}^2}} \right) \ ,$$

$$\psi_1 = \pi + \gamma_\psi + \alpha_\psi \ . \tag{4.13}$$

The "knee" angle, ψ_2, is given by

$$\beta_\psi = \arccos\left(\frac{l_5^2 + l_6^2 - x_{A_{h,\psi}}^2 - y_{A_{h,\psi}}^2}{2l_5 l_6} \right) \ ,$$

$$\psi_2 = \pi + \beta_\psi \ . \tag{4.14}$$

The "ankle" angle, ψ_3, is simply

$$\psi_3 = \measuredangle_f - \psi_1 - \psi_2 \ . \tag{4.15}$$

ϕ-chain inverse kinematics

Just as a 2R arm has "elbow-up" and "elbow-down" configurations, this planar robot has a number of possible configurations for the same end-effector pose. One such example is the "elbow-left" or "elbow-right" configuration of the ϕ chain. The derivations below assume the "elbow-left" configuration, so the equations resemble those of the θ chain (the "elbow-right" configuration equations resemble those of the ψ-chain).

This model defines the origin of the $\{b\}$ frame at joint ϕ_1, so

$$\begin{bmatrix} x_{H,\phi} \\ y_{H,\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ .$$

The "ankle" of the ϕ chain is different from that of the θ and ψ chains and is consequently referred to as the "upper ankle". Repeating the same geometric procedure as done above for the other two chains, we first calculate the "hip" angle, ϕ_1:

$$\gamma_\phi = \text{atan2}(y_{uA}, x_{uA}),$$

$$\alpha_\phi = \arccos\left(\frac{x_{uA}^2 + y_{uA}^2 + l_3^2 - l_4^2}{2l_3\sqrt{x_{uA}^2 + y_{uA}^2}}\right),$$

$$\phi_1 = -\gamma_\phi - \alpha_\phi \ . \tag{4.16}$$

The "knee" angle, ϕ_2, is given by

$$\beta_\phi = \arccos\left(\frac{l_3^2 + l_4^2 - x_{uA}^2 - y_{uA}^2}{2l_3 l_4}\right) \ ,$$

$$\phi_2 = \pi - \beta_\phi \ . \tag{4.17}$$

The "ankle" angle, ϕ_3, is simply

$$\phi_3 = \measuredangle_f - \phi_1 - \phi_2 \ . \tag{4.18}$$

4.1.4 Actuator Jacobian

We would now like to solve the inverse dynamics problem: what motor speeds/torques are required to generate a desired end-effector speed/force? The solution can be approximated using the *actuator Jacobian*.

Loop-closure equations

Refer to Figures 4.1 and 4.2 and observe that the open sub-chains intersect at a common joint, measured by θ_3 and ψ_3. Consequently, there are three loops: the θ-ϕ loop, the ϕ-ψ loop, and the ψ-θ loop. We can describe these three loops with two sets of equations:

$$g(\theta, \phi, \psi) := \begin{bmatrix} T_{bf}(\theta) - T_{bf}(\phi) \\ -T_{bf}(\phi) + T_{bf}(\psi) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.19}$$

where $T_{bf}(\cdot)$ represents the twist from the base frame $\{b\}$ to the end-effector frame $\{f\}$ along the corresponding open sub-chain; this is just a compact representation of the earlier open sub-chain forward kinematic equations, so the left and right sides of the equation are 6×1 column vectors.

For a given set of actuated joint angles $q_a = [\theta_1, \phi_1, \psi_1]^{\mathrm{T}}$, we seek unactuated joint angles ($q_u = [\theta_2, \theta_3, \phi_2, \phi_3, \psi_2, \psi_3]^{\mathrm{T}}$), i.e., the roots of the loop-closure equations, so the iterative Newton-Raphson root-finding equation is

$$\left[\theta_2^{k+1}, \theta_3^{k+1}, \phi_2^{k+1}, \phi_3^{k+1}, \psi_2^{k+1}, \psi_3^{k+1}\right]^{\mathrm{T}} = q_u^{k+1} = q_u^k - J_c^{-1} g\left(q_a, q_u^k\right) \ , \tag{4.20}$$

where J_c, the *constraint Jacobian*, is the matrix of partial derivatives of the loop-closure expression with respect to the elements of q_u. Since the loop-closure equation was expressed as a 6×1 column vector, the constraint Jacobian is a square 6×6 matrix.

Constraint Jacobian

As mentioned earlier, the constraint Jacobian J_c is the matrix of partial derivatives of the loop-closure expression with respect to the unactuated joint positions q_u. Differentiating the loop-closure equation above, we get the following 6×6 matrix:

$$J_c = \begin{bmatrix} \frac{\partial T_{bf}(\theta)}{\partial \theta_2} & \frac{\partial T_{bf}(\theta)}{\partial \theta_3} & -\frac{\partial T_{bf}(\phi)}{\partial \phi_2} & -\frac{\partial T_{bf}(\phi)}{\partial \phi_3} & 0 & 0 \\ 0 & 0 & -\frac{\partial T_{bf}(\phi)}{\partial \phi_2} & -\frac{\partial T_{bf}(\phi)}{\partial \phi_3} & \frac{\partial T_{bf}(\psi)}{\partial \psi_2} & \frac{\partial T_{bf}(\psi)}{\partial \psi_3} \end{bmatrix} \ . \tag{4.21}$$

Actuator Jacobian derivation

The actuator Jacobian maps generalized actuator velocities and forces to end-effector velocities and forces. Returning to the loop-closure equation (equation 4.19) and differentiating with respect to all joint positions (q_a and q_u), we get

$$\begin{bmatrix} J_\theta & -J_\phi & 0 \\ 0 & -J_\phi & J_\psi \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = 0_{6\times 1} \ , \tag{4.22}$$

where each $J \in \mathbb{R}^{3\times 3}$ is the Jacobian of the corresponding open sub-chain. Expanding this equation is straightforward and will help with the next step. When fully expanded, equation 4.22 becomes

$$\begin{bmatrix} J_{\theta_{1,1}} & J_{\theta_{1,2}} & J_{\theta_{1,3}} & -J_{\phi_{1,1}} & -J_{\phi_{1,2}} & -J_{\phi_{1,3}} & 0 & 0 & 0 \\ J_{\theta_{2,1}} & J_{\theta_{2,2}} & J_{\theta_{2,3}} & -J_{\phi_{2,1}} & -J_{\phi_{2,2}} & -J_{\phi_{2,3}} & 0 & 0 & 0 \\ J_{\theta_{3,1}} & J_{\theta_{3,2}} & J_{\theta_{3,3}} & -J_{\phi_{3,1}} & -J_{\phi_{3,2}} & -J_{\phi_{3,3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & -J_{\phi_{1,1}} & -J_{\phi_{1,2}} & -J_{\phi_{1,3}} & J_{\psi_{1,1}} & J_{\psi_{1,2}} & J_{\psi_{1,3}} \\ 0 & 0 & 0 & -J_{\phi_{2,1}} & -J_{\phi_{2,2}} & -J_{\phi_{2,3}} & J_{\psi_{2,1}} & J_{\psi_{2,2}} & J_{\psi_{2,3}} \\ 0 & 0 & 0 & -J_{\phi_{3,1}} & -J_{\phi_{3,2}} & -J_{\phi_{3,3}} & J_{\psi_{3,1}} & J_{\psi_{3,2}} & J_{\psi_{3,3}} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\psi}_1 \\ \dot{\psi}_2 \\ \dot{\psi}_3 \end{bmatrix} = 0_{6\times 1} \ , \tag{4.23}$$

which can be rearranged into an actuated part, $H_a \in \mathbb{R}^{6\times 3}$, and an unactuated part, $H_u \in \mathbb{R}^{6\times 6}$, such that

$$\begin{bmatrix} H_a(q_a, q_u) & H_u(q_a, q_u) \end{bmatrix} \begin{bmatrix} \dot{q}_a \\ \dot{q}_u \end{bmatrix} = 0_{6\times 1} \ , \tag{4.24}$$

or equivalently,

$$\dot{q}_u = -H_u^{-1} H_a \dot{q}_a \ . \tag{4.25}$$

From equation 4.23, it is apparent that

$$H_a(q_a, q_u) = \begin{bmatrix} J_{\theta_{1,1}} & -J_{\phi_{1,1}} & 0 \\ J_{\theta_{2,1}} & -J_{\phi_{2,1}} & 0 \\ J_{\theta_{3,1}} & -J_{\phi_{3,1}} & 0 \\ 0 & -J_{\phi_{1,1}} & J_{\psi_{1,1}} \\ 0 & -J_{\phi_{2,1}} & J_{\psi_{2,1}} \\ 0 & -J_{\phi_{3,1}} & J_{\psi_{3,1}} \end{bmatrix} \text{, and} \tag{4.26}$$

$$H_u(q_a, q_u) = \begin{bmatrix} J_{\theta_{1,2}} & -J_{\phi_{1,3}} & -J_{\phi_{1,2}} & -J_{\phi_{1,3}} & 0 & 0 \\ J_{\theta_{2,2}} & -J_{\phi_{2,3}} & -J_{\phi_{2,2}} & -J_{\phi_{2,3}} & 0 & 0 \\ J_{\theta_{3,2}} & -J_{\phi_{3,3}} & -J_{\phi_{3,2}} & -J_{\phi_{3,3}} & 0 & 0 \\ 0 & 0 & -J_{\phi_{1,2}} & -J_{\phi_{1,3}} & J_{\psi_{1,2}} & J_{\psi_{1,3}} \\ 0 & 0 & -J_{\phi_{2,2}} & -J_{\phi_{2,3}} & J_{\psi_{2,2}} & J_{\psi_{2,3}} \\ 0 & 0 & -J_{\phi_{3,2}} & -J_{\phi_{3,3}} & J_{\psi_{3,2}} & J_{\psi_{3,3}} \end{bmatrix} \ , \tag{4.27}$$

which is, in fact, the constraint Jacobian J_c from the previous section (see equation 4.21), so the velocities of the unactuated joints are

$$\dot{q}_u = -J_c^{-1} H_a \dot{q}_a \ , \tag{4.28}$$

assuming J_c is invertible. Observe that $-J_c^{-1} H_a$ is a 6×3 matrix that maps actuated joint velocities to unactuated joint velocities. For example,

$$\dot{\theta}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \left(-J_c^{-1} H_a \right) \dot{q}_a \text{ , and} \tag{4.29}$$

$$\dot{\theta}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \left(-J_c^{-1} H_a \right) \dot{q}_a \ . \tag{4.30}$$

We can develop a mapping from the velocities of the actuated joints to the end-effector velocity, using the θ-chain:

$$\begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ \dot{\measuredangle}_f \end{bmatrix} = J_\theta \left(q_a, q_u\right) \dot{\theta} = J_\theta \begin{bmatrix} & & & \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} & & \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \end{bmatrix} \dot{q}_a \ , \tag{4.31}$$

or more simply,

$$\begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ \dot{\measuredangle}_f \end{bmatrix} = J_{a,\theta} \left(q_a, q_u\right) \dot{q}_a \ , \tag{4.32}$$

where $J_{a,\theta} \in \mathbb{R}^{3\times 3}$ is the θ-chain *actuator Jacobian*. Similar expressions can be derived for $J_{a,\phi} \in \mathbb{R}^{3\times 3}$ and $J_{a,\psi} \in \mathbb{R}^{3\times 3}$ by returning to equation 4.28 and deriving expressions for $\dot{\phi}_2$, $\dot{\phi}_3$, $\dot{\psi}_2$, and $\dot{\psi}_3$. Together, the actuator Jacobians are

$$J_{a,\theta} = J_\theta \begin{bmatrix} & & & \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} & & \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \end{bmatrix} \tag{4.33}$$

$$J_{a,\phi} = J_\phi \begin{bmatrix} & & & \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} & & \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \end{bmatrix} \tag{4.34}$$

$$J_{a,\psi} = J_\psi \begin{bmatrix} & & & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} & & \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & & & & & \begin{pmatrix} -J_c^{-1} H_a \end{pmatrix} \end{bmatrix} . \tag{4.35}$$

The relationship between joint speeds and end-effector speed is given above in equation 4.32, but in general, the relationship is

$$V = J_a \dot{\theta} \quad \text{and } \dot{\theta} = J_a^{-1} V \ . \tag{4.36}$$

Similarly, the relationship between joint torques and end-effector force is

$$\tau = J_a^{\mathrm{T}} F \quad \text{and } F = J_a^{-\mathrm{T}} \tau \tag{4.37}$$

4.1.5 Link geometry optimization for jumping

Having developed the forward and inverse kinematics and the actuator Jacobian, it is now possible to optimize link geometry for virtually compliant locomotion. Consider two limit cases:

1. if the robot is too small, the distance over which a virtual spring can compress is too short, so the robot is not very compliant;

2. conversely, if the robot is too large, generating an arbitrary end-effector wrench will require enormous motor torques.

One way to optimize link geometry for virtually compliant locomotion is to maximize both the linkage's mechanical advantage and the end-effector's workspace. The workspace is the reachable subset of the task space ($\mathcal{W} \subseteq SE(2)$). Maximizing the linkage's mechanical advantage allows selection of a lower-torque motor to apply a given end-effector wrench. Maximizing the "volume" of the end-effector's workspace is important not just because it maximizes not only the robot's mobility but also the "travel" over which a virtual spring can compress and extend.

The nonlinear kinematic equations and actuator Jacobian result in a feature-rich parameter space with potentially many local minima, so a nonlinear optimization method such as simulated annealing or MATLAB's `fmincon` is more suitable than gradient descent for finding the optimizer. After selecting an optimization method, the problem reduces to designing an objective function that captures the qualities of the optimizer (large mechanical advantage and large workspace). Algorithm 3 computes the value of such a quadratic objective function that captures these two qualities.

Algorithm 3 Link length objective function

function ObjectiveFunction(l) ▷ l is a list of the leg lengths and actuated joint locations.
  for each $(x_{f,i}, y_{f,i}, \measuredangle_{f,i}) \in \mathcal{W}$ **do**
    if IK solvable & $q_{min} \leq q \leq q_{max}$ **then**
      append $(x_{f,i}, y_{f,i}, \measuredangle_{f,i})$ **to** $\mathcal{W}_{reachable}$
      Given $l, (x_{f,i}, y_{f,i}, \measuredangle_{f,i}), q$, compute $\tau = J_a^{\mathrm{T}} F$
      append $\|\tau\|$ **to** τ_{list}
    end if
    $V_{workspace} = \mathrm{vol}(\mathcal{W})$
    $\tau_{max} = \max(\tau_{list})$
  end for
  return $J = w_1 \tau_{max}^2 + w_2 V_{workspace}^2$.
end function

We used MATLAB's `fmincon` to minimize this objective function, and the resulting dimensions are listed in Table 4.1.

Dimension	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	$b_{1,x}$	$b_{1,y}$	$b_{2,x}$	$b_{2,y}$
Length (cm)	6.57	15.2	8.43	8.56	6.57	15.2	6.91	10.2	5.73	0.82	5.73	0.82

Table 4.1: Optimized link lengths

After optimizing these dimensions, we chose to drive the left and right actuated joints with timing belts, in order to achieve a 1.5:1 gear reduction. Timing belts allow for a gear reduction without much backlash and make it possible to place the motors such that the robot's CoM is located on the boom's roll axis.

4.2 Motor selection

With the actuator Jacobian and optimized link lengths, we can select motors based on the specifications outlined in Section 1.2.

4.2.1 Torque and speed requirements

Torque requirement

Given the specified maximum weight and jump height, the motors must provide roughly 5 J. Assuming a lossless jump, the robot's liftoff velocity must be 1.41 $\frac{\mathrm{m}}{\mathrm{s}}$. If jumping from rest, the change in momentum is $5\mathrm{kg} \times 1.41 \frac{\mathrm{m}}{\mathrm{s}} = 7.07 \frac{\mathrm{N}}{\mathrm{s}}$, and to satisfy the 25% stance time specification, the robot has roughly 0.1 s to apply a 70 N force that will result in this change in momentum. If jumping repeatedly, this time window is halved, so the robot must apply 141.4 N for 0.05 s. Assuming the robot begins applying this force in a "compressed" configuration $q_a = (-135°, -152°, -45°)$, the required joint torques are

$$\tau_{jump} = J_a(q)F = J_a(q) \begin{bmatrix} 0 \\ -141.4 \\ 0 \end{bmatrix} = \begin{bmatrix} 6.4 \\ 0 \\ -6.4 \end{bmatrix} \text{ Nm}, \tag{4.38}$$

which, due to the timing belt gearing, are reduced to ±4.3 Nm.

Speed requirement

To determine motor speed requirements, we consider the task of "sweeping the leg" during flight, i.e. moving from a foot pose (−5 cm, −25 cm, −105°) to a foot pose (+5 cm, −25 cm, −75°). Given the jump height specification, the robot has roughly $dt = 0.3$ s to accomplish this task. The actuated joint angles corresponding to the initial and final foot poses are $q_{a,i} = (-146.1°, -154.5°, -61.7°)$ and $q_{a,f} = (-118.3°, -146.7°, -33.9°)$, respectively, so the motors must spin at $\frac{q_{a,i} - q_{a,f}}{dt} = (15.4, 4.3, 15.5)$ rpm.

Based on this speed requirement and the torque requirement calculated above, and bearing in mind the mass of each motor could not exceed $\frac{5 \text{ kg}}{3} = 1.6$ kg, we selected the Maxon 244789 brushless DC (BLDC) motor [4] for each of Hopp3r's actuators.

4.2.2 Thermal analysis

Thermal model

The motivation for deriving a thermal model of Hopp3r's motors is to determine how long they can operate before reaching their thermal limit. Based on the thermal model, it may also be possible to push the motors beyond their rating. We begin by modeling the motor with three control volumes (the windings, the housing, and the backplate to which all three motors are mounted), depicted in Figure 4.6.

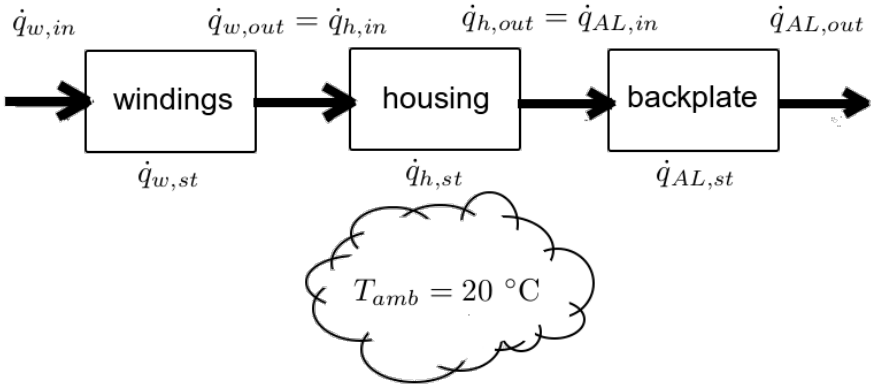

Figure 4.6: Control volumes used to derive thermal model of each motor

The heat stored in each control volume is the difference between the heat delivered and the heat lost:

$$\dot{q}_{w,st} = \dot{q}_{w,in} - \dot{q}_{w,out}, \tag{4.39a}$$

$$\dot{q}_{h,st} = \dot{q}_{h,in} - \dot{q}_{h,out}, \text{ and} \tag{4.39b}$$

$$\dot{q}_{AL,st} = \dot{q}_{AL,in} - \dot{q}_{AL,out}. \tag{4.39c}$$

The heat delivered to the windings comes from Joule heating, so $\dot{q}_{w,in} = I^2 R$, where I is the current through the motor windings and R is the resistance of the motor windings; current is approximated as torque times the motor's speed constant: $I = k_v \tau$. For each control volume, heat stored is given by $mc_p\dot{T}$ and heat loss is given by $hA\Delta T$. This heat loss occurs via conduction between the windings and the housing and between the housing and the backplate; heat is lost from the backplate to the surrounding ambient air by free convection (*free* because we are not actively cooling the motors). Expanding Equation 4.39 yields the following linear-affine thermal model:

$$\frac{d}{dt}\begin{bmatrix} T_w \\ T_h \\ T_{AL} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m_w c_w R_{T,wh}} & \frac{1}{m_w c_w R_{T,wh}} & 0 \\ \frac{1}{m_h c_h R_{T,wh}} & -\frac{1}{m_h c_h R_{T,wh}} - \frac{1}{m_h c_h R_{T,hAL}} & \frac{1}{m_h c_h R_{T,hAL}} \\ 0 & \frac{1}{m_{AL} c_{AL} R_{T,hAL}} & -\frac{1}{m_{AL} c_{AL} R_{T,hAL}} - \frac{h_{AL} A_{AL}}{m_{ALh} c_{AL}} \end{bmatrix} \begin{bmatrix} T_w \\ T_h \\ T_{AL} \end{bmatrix} + \begin{bmatrix} \frac{I^2 R_w}{m_w c_w} \\ 0 \\ \frac{h_{AL} A_{AL}}{m_{AL} c_{AL}} T_{amb} \end{bmatrix} \tag{4.40}$$

The thermal properties of the Maxon 244879 BLDC motor and the aluminum backplate are listed in Table 4.2.

Parameter	Units	Value
k_v	$\frac{\text{A}}{\text{Nm}}$	4.62
R	Ω	2.28
$T_{w,max}$	°C	125
m_w	kg	0.3
m_h	kg	0.3
m_{AL}	kg	0.7
c_w	$\frac{\text{J}}{\text{kg}\cdot^\circ\text{C}}$	390
c_h	$\frac{\text{J}}{\text{kg}\cdot^\circ\text{C}}$	910
c_{AL}	$\frac{\text{J}}{\text{kg}\cdot^\circ\text{C}}$	910
$R_{T,wh}$	$\frac{^\circ\text{C}}{\text{W}}$	2.6
$R_{T,hAL}$	$\frac{^\circ\text{C}}{\text{W}}$	1.91
h_{AL}	$\frac{\text{W}}{\text{m}^2\cdot^\circ\text{C}}$	182.2
A_{AL}	m^2	0.086

Table 4.2: Parameters for thermal model of motors

The derivation for the heat transfer coefficient h_{AL} for convection from the aluminum backplate to the surrounding ambient air is from [1] and is as follows:

$$h_{AL} = \frac{Nu_L k_{air}}{L} \ , \tag{4.41}$$

where L, the characteristic length approximated by the backplate's surface area divided by its perimeter, is 0.112 m. The Nusselt number for *free* convection at a vertical wall, given this characteristic length, is given by

$$0.68 + \frac{0.67 \ \text{Ra}_L^{\frac{1}{4}}}{\left[1 + \left(\frac{0.492}{\text{Pr}}\right)^{\frac{9}{16}}\right]^{\frac{4}{9}}} \ , \tag{4.42}$$

where Ra_L and Pr are the Rayleigh and Prandtl numbers, respectively. Here, $\text{Pr} = 0.705$ and the Rayleigh number is given by

$$\text{Ra}_L = \frac{g\beta \left(T_s - T_{amb}\right)\left(L^3\right)}{\nu\alpha} \ , \tag{4.43}$$

where

- g (gravitational acceleration) = 9.81 $\frac{\text{m}}{\text{s}^2}$,
- β (thermal expansion coefficient) = 0.003 $^\circ\text{C}^{-1}$,
- T_s (surface temperature) is approximated as 100 °C,
- T_{amb} (ambient temperature) is assumed to be 20 °C, i.e. room temperature,
- ν (kinematic viscosity) = $19 \times 10^{-6} \frac{\text{m}^2}{s}$, and
- α (thermal diffusivity) = $27 \times 10^{-6} \frac{\text{m}^2}{s}$.

Lastly, the thermal conductivity of air, k_{air}, is 0.0288 $\frac{\text{W}}{\text{m}^\circ\text{C}}$.

Simulation results

Two tasks, balancing and jumping, were simulated. In balancing, the motors continuously applied torques $\tau_{balance}$ required to keep the robot upright, calculated using the Jacobian transpose method:

$$\tau_{balance} = J_a(q)^{\mathrm{T}} F \ , \tag{4.44}$$

where $J_a(q)$ is the configuration-dependent actuator Jacobian derived in Section 4.1.4 and the end-effector wrench F is simply $[0, mg, 0]^{\mathrm{T}}$, i.e. a force opposing the robot's weight. As shown in Figure 4.7, the robot can balance for over two minutes before the motor windings reach their critical temperature $T_{w,max}$.

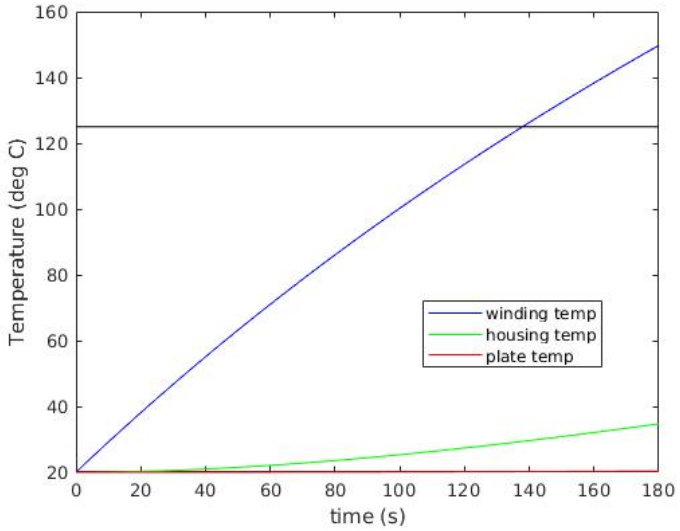

Figure 4.7: Temperature profiles of motor winding, motor housing, and backplate during a balancing task

In the second simulation, the robot jumps with period 0.4 seconds, applying the stall current (21 A) for 0.1 s and applying 0 A for the remaining 0.3 s. As shown in Figure 4.7, the robot can jump like this for about one minute before the motor windings reach their critical temperature. Note that this jumping behavior is a very crude approximation of the current profile for *intelligent* periodic jumping, which might use impedance control to absorb impacts upon landing, resulting in more intricate current profiles.

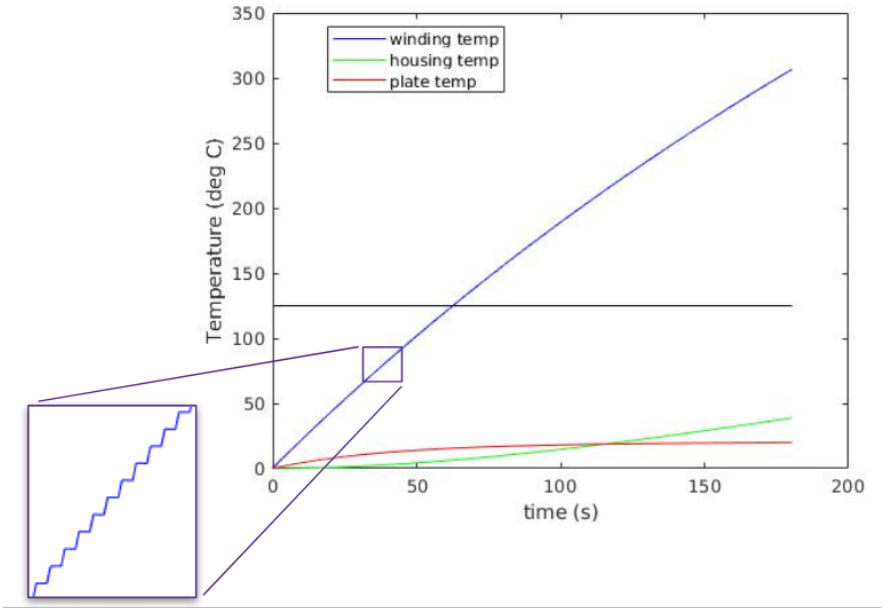

Figure 4.8: Temperature profiles of motor winding, motor housing, and backplate during steady state jumping. The close-up shows the sharp rise in temperature during each jump.

Observability

The stall current of a motor is typically derived from how much heat the windings (or, more specifically, the winding insulation) can dissipate. Aggravatingly, directly measuring the winding temperature ranges anywhere from impractical to impossible, depending on motor construction and placement. Thus, the ability to estimate the winding temperature is desirable, and *observability* is a prerequisite for this ability. Since Equation 4.40 is linear-affine, determining observability is easy. Let $x = [T_w, T_h, T_{AL}]^{\mathrm{T}}$ represent the state of the thermal system, so that the state-space representation of the system is

$$\dot{x} = Ax + Bu \tag{4.45a}$$

$$y = Cx + Du \tag{4.45b}$$

where A and Bu are given in Equation 4.40, and $y = Cx + Du$ is the output of the system. Let $D = 0$ (the current through the motor does not show up the system's output) and consider two cases:

1. there is a temperature sensor mounted on the motor housing, so we can directly measure the housing temperature. In this case, $C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$.
2. as above, we can directly measure the housing temperature, and with an additional sensor mounted on the backplate, we can also measure the backplate temperature. In this case, $C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

For a linear system with n-th dimensional state, observability is determined by the rank of the observability matrix:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{4.46}$$

The system is observable if $rank(\mathcal{O}) = n$.

For the thermal model in Equation 4.40, it turns out that for either choice of C enumerated above, $rank(\mathcal{O}) = 3$, so, whether we choose to measure the backplate temperature or not, the system is observable! All that remains is to actually create the estimator; given that the system is linear, a Kalman filter seems like a reasonable starting point.

4.3 Transmission design and analysis

4.3.1 Belt selection

See"Belts" document for more detailed analysis and walk-through with links to the proper diagrams.

Assume 4 Nm of torque and 300 rpm needed for the big pulley, with a safety factor of 2 and gearing ratio of 1.5. This means a torque of 2.7 Nm and 450 rpm. Therefore design for 8 Nm (4 Nm * FOS=2) and 450 rpm. Chose the GT2/3 series because we are willing to pay up for the extra strength. Based on charts provided by SDP/SI, we need a 5mm GT3 or 5mm HDT. Using their online calculator and product catalog, we selected a center distance of 140 mm and 98 tooth belt, and 32/48 teeth on the pulleys. This means the small pulley gets a 14.4 Nm torque rating, a length ratio of .92, a width ratio of .6, which means an adjusted torque rating of 7.95, which is close enough to a FOS of 2. Finally, using their calculators, we have sufficient teeth engagement (15 and 25), which is much more than required. The tension should be $18\text{lbf} \times \text{FOS}(= 2) = 36\text{lbf}$.

4.3.2 Shaft load calculations

See "Belts" document for links to instructional documents with diagrams and step by step instructions.

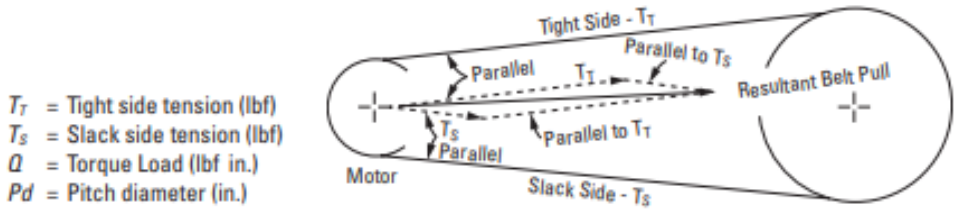

Figure 4.9: Belt pull vector diagram

Tensions and resulting forces

Figure 4.9 illustrates the tensions in our belt drive. For a torque of 8 Nm = 71 lbf-in. with pitch diameters (P_d) of ~ 2 in. and ~ 3 in. for driving and driven pulleys respectively, we get four tensions. We then add 36 lbf to each one because we are using the belt as a registration drive:

$$T_{t,driving} = 2.5 \times \frac{71}{2} = 88.75 \text{ lbf} + 36 \text{ lbf} = 124.75 \text{ lbf}, \tag{4.47a}$$

$$T_{s,driving} = 0.5 \times \frac{71}{2} = 17.75 \text{ lbf} + 36 \text{ lbf} = 53.75 \text{ lbf}, \tag{4.47b}$$

$$T_{t,driven} = 2.5 \times \frac{71}{3} = 59.16 \text{ lbf} + 36 \text{ lbf} = 95.16 \text{ lbf}, \text{ and} \tag{4.47c}$$

$$T_{s,driven} = 0.5 \times \frac{71}{3} = 11.83 \text{ lbf} + 36 \text{ lbf} = 47.83 \text{ lbf}. \tag{4.47d}$$

We then calculate the vector sum correction factor by comparing the center-to-center distance (144 mm or 5.7 in.) of the pulleys to the difference in pitch diameter (3 and 2 inches). The equation is $\frac{D-d}{c} = \frac{3-2}{5.7} = 0.18$ which, from Figure 4.10, correlates to a vector sum correction factor of approximately 0.99.

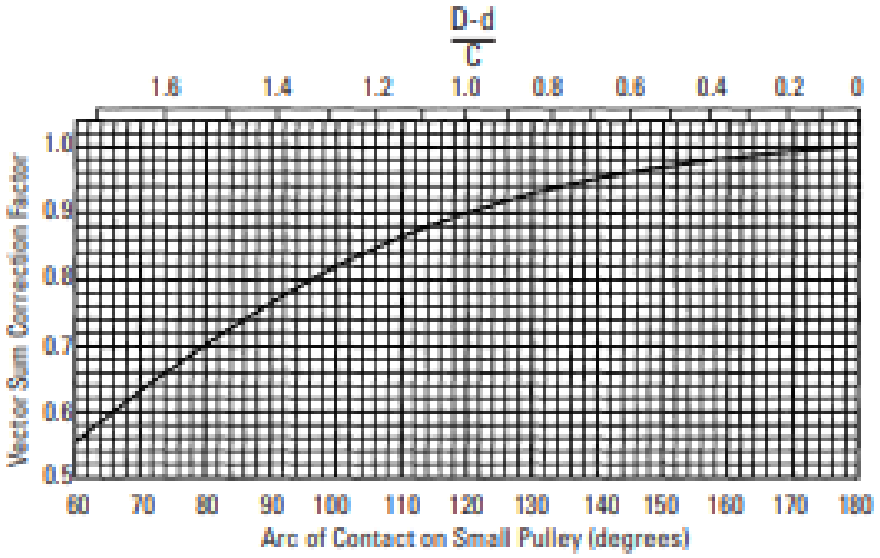

Figure 4.10: Vector sum correction factor

We then add the two force vectors together to get the total force on the shaft and multiply by the vector sum factor. This gives us a driving force of $(124.75 + 53.75) \times 0.99 = 176.7$ lbf and a driven force of $(95.16 + 47.83) \times 0.99 = 141.6$ lbf. These are the forces on each shaft applied at the pulley axially.

If the bearings are equally spaced on the outsides of the pulleys, then the radial force on the bearings are simply $\frac{1}{2}$ the load. If they are both on one side of the pulley in a cantilever, the forces are amplified.

4.3.3 Shaft failure calculations

The shafts in the upper part of the robot were expected to transmit 4 Nm of torque to the legs for jumping. The belt and pulley system was designed for this load case according to SDP/SI's recommendations. For our specific torque requirement, considered as $\tau = 8$ Nm in each shaft for a minimum safety factor of 2, and required belt tension $F_{belt\ tension} = 36$ lbf (160 N), the calculated radial force $F_{radial\ equiv}$ due to appropriate belt tension and transmitted torque on the driving (P101) and driven (P103) shafts are 353 N and 264 N, respectively. Due to the higher effective force of 353 N, shaft P101 (directly coupled to the motors and the small pulleys) are considered most critical. The three remaining shafts are under lighter loads, and therefore should be considered safe if designed similarly to the top shafts.

Given these loads and the geometry of the design, the bearing reaction forces and shaft moments were calculated. The following calculations summarize the key steps of computing the Goodman's line safety factor for the most critical shaft (P101).

$$F_{torque\ equiv} = \frac{\tau}{r_{pulley}} = \frac{8\text{ Nm}}{25.4\text{ mm}} = 315\text{ N}, \tag{4.48}$$

$$F_{radial\ equiv} = \sqrt{F_{belt\ tension}^2 + F_{torque\ equiv}^2} = 353\text{ N}. \tag{4.49}$$

Given maximum shaft bending moment M_{max} of 5.22 Nm (due to the bearing's reaction force) and a shaft diameter $d = 12$ mm, the maximum bending stress is

$$\sigma_{max} = \frac{32M_{max}}{\pi d^3} = 30.8\text{ MPa}, \tag{4.50}$$

and given a stress concentration factor $K_f = 1.26$, the bending stress amplitude is $\sigma_{ba} = K_f\sigma_{max} = 38.8$ MPa.

For a torque $\tau = 8$ Nm, the shear stress amplitude is

$$\tau_a = \frac{16\tau}{\pi d^3} = 23.6\text{ MPa}, \tag{4.51}$$

and the Von Mises stress is

$$\sigma_a^{'} = \sqrt{\sigma_{ba}^2 + 3\tau_a^2} = 56.4 \text{ MPa}. \tag{4.52}$$

To determine the safety factor, we first calculate the endurance limit S_e given an ultimate tensile strength $S_{ut} = 310$ MPa:

$$S_e = \text{Correction Factor} \times 0.5 \times S_{ut} = 0.772 \times 0.5 \times 310 \text{ MPa} = 119.7 \text{ MPa}. \tag{4.53}$$

Finally, the safety factor is

$$n_s = \frac{S_e}{\sigma_a^{'}} = 2.12. \tag{4.54}$$

This conservative analysis shows that even for twice our expected operating torque, the Goodman's line safety factor is still 2. We means that we can expect to be able to run these shafts at about 16 Nm (about 4 times beyond our expected operating region) before failure.

4.3.4 Hard stops

To implement hard stops on the legs, we opted for a simple design of a bolt and nut. This allowed for easy replacement of the stops, should there be wear or failure. We assumed a worse case scenario of a 4 Nm torque being applied on a leg. We then designed the hard stops to be 1.5 inches (0.0381 m) away from the center of the output shafts. By dividing the torque by the radius, we get a force of 105 N.

Next, we calculate the moment I of the hard stops, approximating them as cylindrical bars, so that

$$I = \frac{1}{3}m(3r^2 + L^2). \tag{4.55}$$

The impact on the bolt is 1.3 inches away from the front face of the robot, so $L = 1.3$ inches. After a few iterations, we ended up selecting a bolt size of $\frac{5}{16}$", so $r = 0.15625$ inches. From the length L and radius r, the volume is 1.63×10^{-6} m^3, and given a density (steel) of $\rho = 7800$ kg/m^3, the bolt has mass $m = 0.0127$ kg, so from the moment equation above, $I = 0.0075$ kg $\cdot$ m^3.

To calculate the mass stress in the system, we start with the bending moment I_b, which is 23.6 lbf (105 N from before) times the length (1.3 inches), or 30.68 in·lbf. The centroidal distance is simply the radius of the bolt ($r = 0.15625$ in). The moment of inertia I_c can then be calculated: $I_c = \frac{\pi r^4}{4} = 4.7 \times 10^{-4}$ in^4.

The maximum bending stress is

$$\sigma = I_b \frac{r}{I_c} = 10.24 \text{ ksi}. \tag{4.56}$$

A grade 8 steel bolt has a tensile strength of 150 ksi, so the hard stops have a safety factor of almost 15. Although these bolts are suitable as hard stops, they are evidently excessive and can be reduced in size; for example, switching to 10-24 bolts, used throughout the robot, results on a safety factor of 3.3.

4.3.5 Link FEA

Finite element analysis (FEA) was performed on several waterjetted 6061 aluminum (2.76×10^8 N/m^2 yield strength) parts of the robot. For all analyses, the load case was a 250 N force directed into the bottom of the foot. This load approximates a 5g impact force.

Left and right upper legs

Load case 1:

- *Assumption 1*: Assume half of the load goes into the two side legs equally, with none in the middle leg.
- *Assumption 2*: Assume a worst case scenario where the load goes in perpendicular to the centerlines.

See "Force calculations for legs" document for geometry and math for loads. This results in a load of 163 N, and the link passes with a safety factor close to 10. See Figure 4.11.

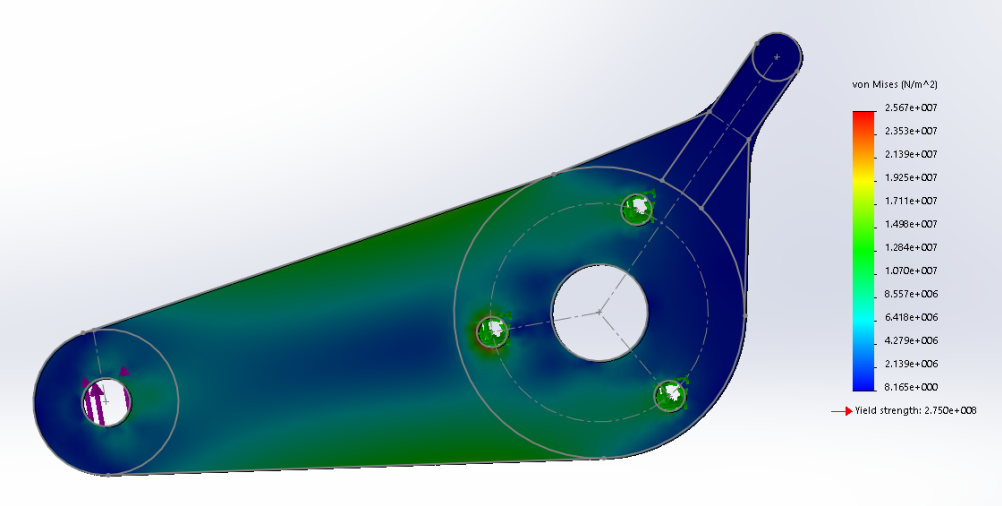

Figure 4.11: Load case 1 for leg A

Load case 2:

Using the assumed torque of 4 Nm (from the pulley calculations above) applied at the three holes and the hard stop face locked, we get a safety factor of roughly 10. See Figure 4.12. This would not be a shock load, but a load applied after this link is already pressed against the bolt. Given the FOS of 10, failure due to this load case is unlikely.

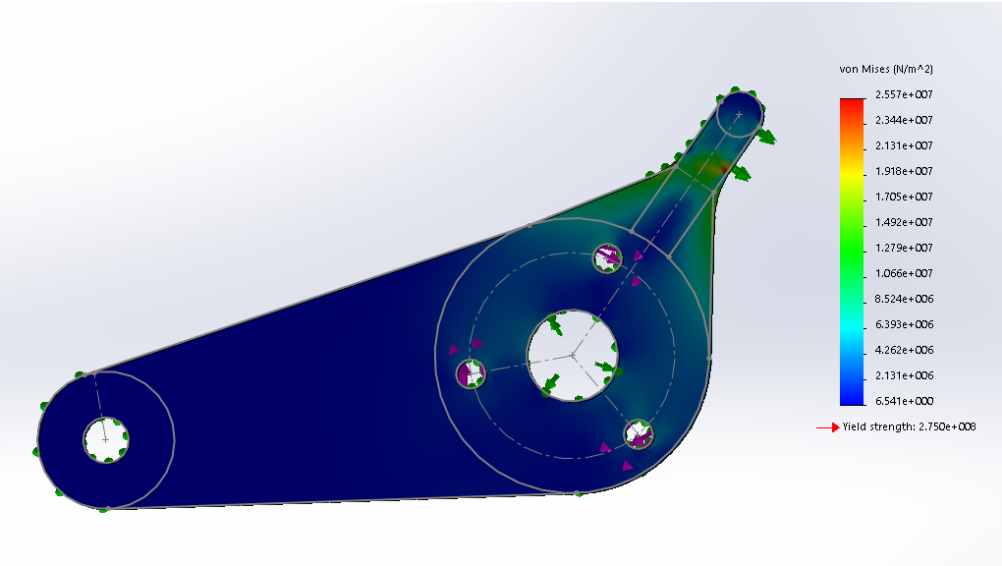

Figure 4.12: Load case 2 for leg A

Middle upper leg

Load case 3:

- *Assumption 1*: Assume all the load goes into only the center portion of the legs
- *Assumption 2*: Assume similar geometry to previous load case, i.e. a load of 325 N tangent to the centerlines.

Locked the three holes to simulate a stalled motor. The link passes this simulation with a safety factor of about 1.5. See Figure 4.13.

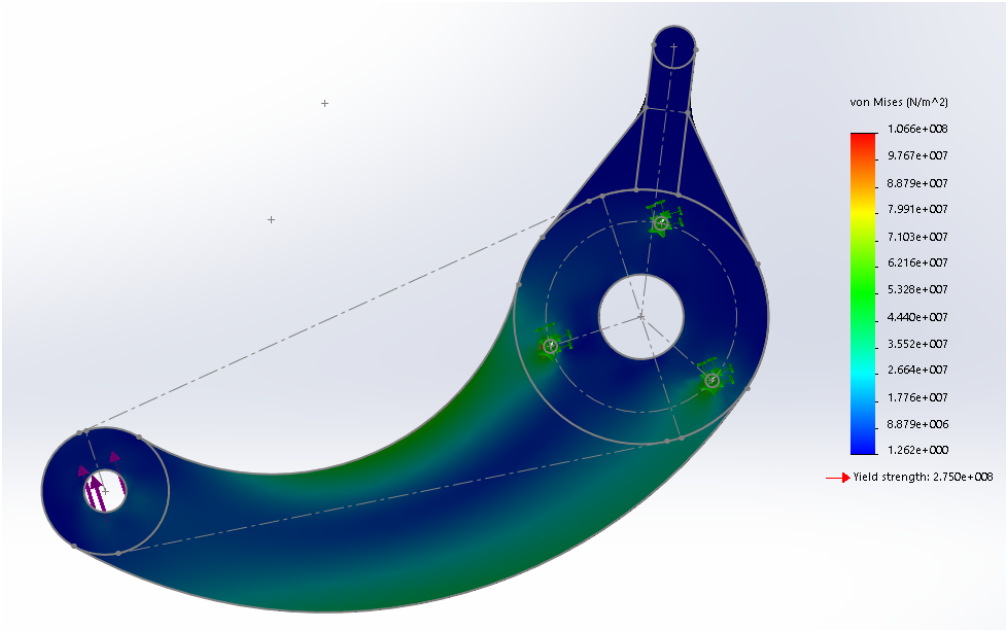

Figure 4.13: Load case 3 for leg C

Support plate

Load case 4: Using the loads found in the "Belts" document in ideal loading, the support plate will have a FOS of 49. Given this FOS, the plate can withstand most un-ideal load cases.

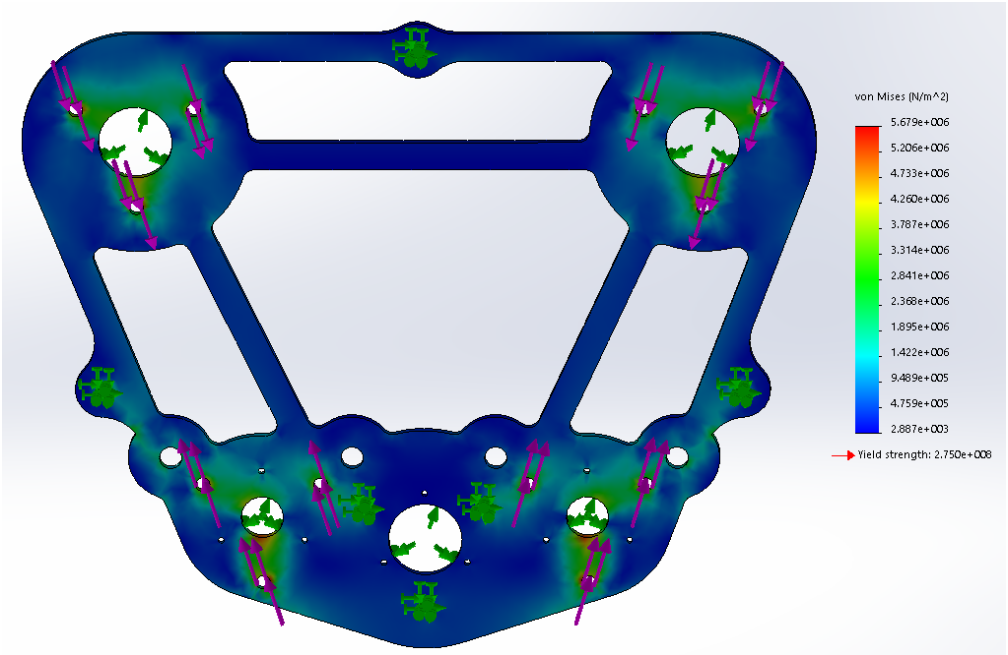

Figure 4.14: Load case 4 for the support plate

4.4 Boom selection

Recall from Section 1.2.4, the initial critical specifications for the choice of boom material and cross section were:

- the boom must be 1.2 - 1.5 m long. This is based on the radius of the circular arc to travel from corner to corner in the existing testbed.
- the boom must not deflect more than 0.5% of its length when holding the robot at rest. Assuming the maximum boom length and the maximum 5 kg mass of the robot, the boom must not deflect more than 7.5 mm for with a 50 N load on its end. Deflection in the boom can be modeled as a spring-type dynamic component - thus large boom deflections would further complicate the challenging dynamics of the full system.
- the boom mass should be less than 10% of the robot's mass. Again assuming the maximum robot mass, this limits the boom's mass to 500 g. This ensures that the boom's mass and inertia are negligible compared to that of the robot.

This analysis was performed by modeling the boom as a cantilevered Bernoulli-Euler beam with a concentrated load P at its free end. From Bernoulli-Euler tables, the maximum deflection δ for a beam of length L, moment of inertia I, and modulus of elasticity E is

$$\delta = \frac{PL^3}{3EI}. \tag{4.57}$$

It is clear that for a given material, increasing the moment of inertia I decreases the deflection, but often at the cost of increased mass. This was the key tradeoff in this analysis. We analyzed booms of different common materials (steel, aluminum, and carbon fiber) and different simple cross sections (hollow square and tube). Due to its high strength to weight ratio, we chose a carbon fiber tube, with dimensions $D = 1\frac{5}{16}$" (OD) and $d = 1\frac{3}{16}$" (ID). The following calculations show the deflection δ and mass m of a length $L = 1.5$ m boom supporting a 5 kg robot:

$$P = 5 \text{ kg} \times 9.81 \frac{\text{m}}{\text{s}^2} = 49 \text{ N}, \tag{4.58a}$$

$$I = \frac{\pi \left(D^4 - d^4\right)}{64} = 1.99 \times 10^{-8} \text{ m}^4 , \tag{4.58b}$$

$$\delta = \frac{PL^3}{3EI} = 12.1 \text{ mm}, \tag{4.58c}$$

$$m = \rho V = \rho AL = \rho \frac{\pi}{4}(D^2 - d^2)L = 0.476 \text{ kg}, \tag{4.58d}$$

where, for carbon fiber, $E = 228$ GPa and $\rho = 2000 \frac{\text{kg}}{\text{m}^3}$.

Though even this choice of boom does not meet the deflection specification in the worst case length and load scenario, this selection was deemed acceptable by the team and its sponsor, as the calculated deflection is still 0.8% of the total length.

4.5 Counterweight design

The boom counterweight system needed to be able to counter the weight of the boom at the minimum and incrementally counter additional loads up to the entire weight of system (boom and robot combined). The counterweight system needed to be designed with the following important considerations in mind:

- Counterweight adjustments are incremental with a variety of resolution scales so that the counterweight can be adjusted with precision
- Counterweight system should be as compact as possible and allow for maximum clearance underneath its pivot so that it does not limit the height to which the robot can jump
- Counterweight system should be easily adjustable so that iterations of weighting schemes can be quickly tested and adjusted as necessary
- Counter weight system should have clamping mechanisms built in so that the weights do not oscillate during jumping and cause vibrational inaccuracies in collected data

The calculation involved was a simple balancing of moments about the fulcrum point, where

$$M(x) = F \times x \tag{4.59}$$

The moment $M(x)$ at a given location x away from the fulcrum is dependent on the load F at that location. A simplifying assumption that was made was to consider distributed loads such as the weight of the boom to be acting as a point load at a distance halfway between the beginning and end of the load. The resulting free body diagram and calculated loads for the extremes of the required load conditions are shown in Figure 4.15.

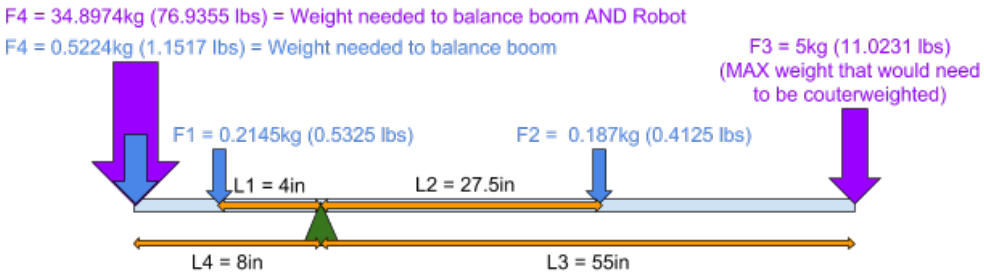

Figure 4.15: Free body diagram used for designing counterweight system

From these calculations, we designed and built a counterweight system that was able to handle plate weights ranging from 0.5 kg to 10 kg as well as precision adjustment scale weights ranging from 5 g to 50 g. We purchased three precision weight sets and one plate weight set so that we are able to adjust in a range of 0 to 40 kg with up to 5 g precision.

Chapter 5

Design Evolution

5.1 Mechanical

At the Midterm presentation in Winter Quarter, we had a vague idea of what the robot would look like and how it would function (see Figure 5.1). The purpose of the design at this point was to get the concept of the robot down and discuss the system as a whole. Since then, every single mechanical component has been redesigned in a significant way.

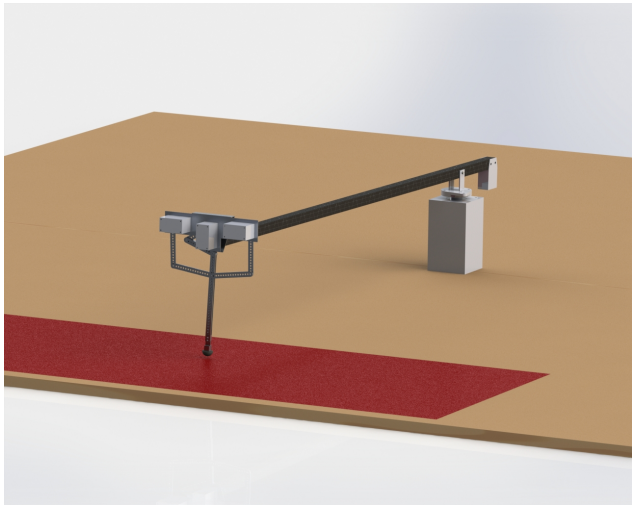

Figure 5.1: CAD rendering of system design as of Winter quarter midterm

At the Winter Quarter Final, the base and boom holder assemblies were fully designed with the robot incomplete (see Figure 5.2). We had made $\frac{3}{4}$ of the base and boom holder assembly. We have since finished making it, including some minor redesigns of components like the counterweight holder and base block where we switched from two roller bearings to a roller and a thrust bearing.

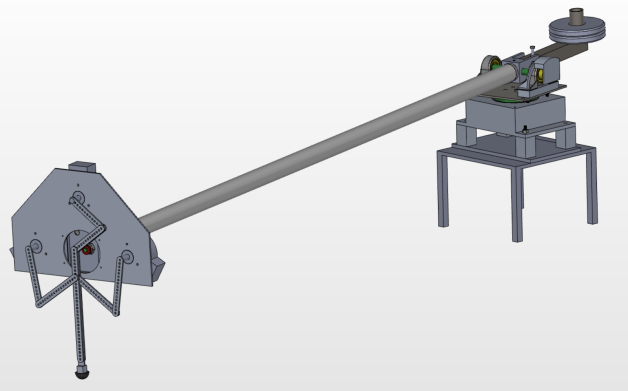

Figure 5.2: CAD rendering of system design as of Winter quarter final

At the Spring Quarter Midterm, the entire system was designed in a way that was supposed to be final (see Figure 5.3). We had finished the construction of the entire base and boom holder assembly with the exception of the counterweight system. We had the entire robot manufactured and assembled as well, but we have since redesigned many of those components such as changing from L brackets to square tubing for the legs.

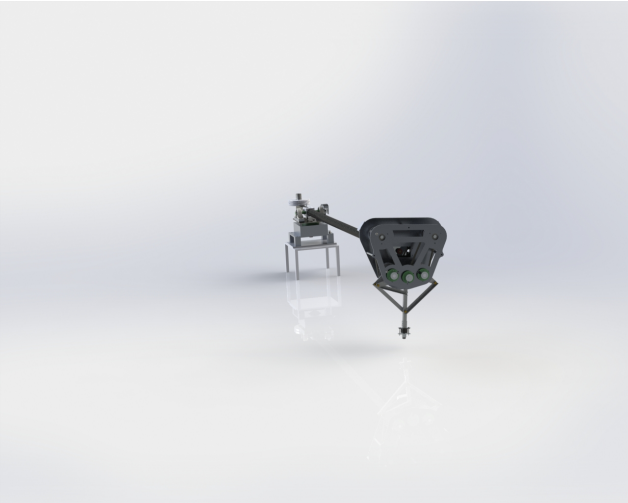

Figure 5.3: CAD rendering of system design as of Spring quarter midterm

5.2 Electrical

The layout of the electrical system took shape quickly during the start of winter quarter, when we decided to use a CAN bus and determined that our nodes would be a single-board computer, three motor control boards, three boom encoder boards, and one board for communicating with the IMU and the force sensor. As we went through the motor selection process, we also decided to use the Copley Accelus motor controller panels to control our motors. By the end of winter quarter, we had designed and ordered the first versions of each of the custom Tiva boards and were set on the Copley amplifiers.

At the start of spring quarter, we went through two quick iterations of our custom Tiva boards to fix circuit bugs and make sure that we had the exact functionality we wanted out of the boards. Overall, there were no significant additions or subtractions from the design of the custom boards. We also decided to use a set of four 12 V batteries in series to power our motors and the Copley amplifiers, and had to design the power electronics to make sure that the system was still safe.

From the midterm of Spring quarter to the end of Spring quarter, no major changes were made to the electrical design. The only additions were a second switch in the positive voltage line of the battery bank, as

an additional safety precaution. At the end of spring quarter, all of the pieces of the electrical system were assembled and functioning as intended.

5.3 Software

Over the course of both quarters, there was roughly linear progress in terms of building the necessary software for our robot. As the mechanical design came further and further along, we introduced new functionality into the firmware of the Tiva boards and developed the CAN bus message hierarchy used by the Tiva boards and the Raspberry Pi. At the end of Spring quarter, we had high-functioning firmware for each Tiva board and could send and receive messages to and from the Raspberry Pi.

As Tiva firmware development progressed, we simultaneously developed a multi-threaded real-time control program on the Raspberry Pi, with separate threads for reading the CAN bus, calculating control inputs and writing them to the CAN bus, and streaming operational data back to the client program over UART.

Chapter 6

Final Design Description

6.1 Mechanical

The system, shown in Figure 6.1, is separated into two primary systems, the robot and the holder, each with their own subsystems.

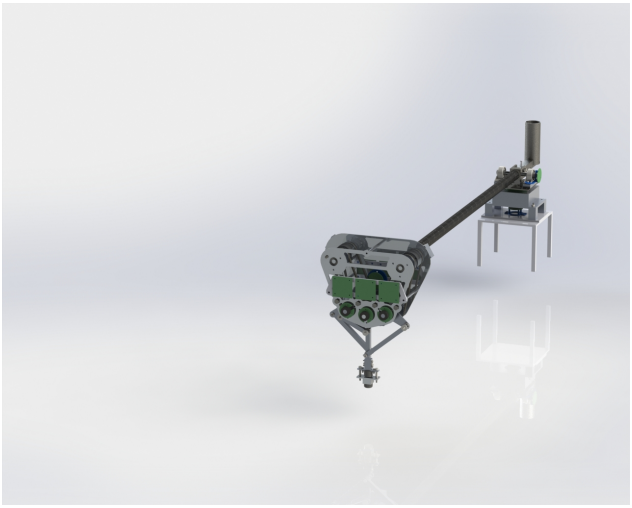

Figure 6.1: CAD rendering of system design as of Spring quarter final

6.1.1 Robot

The robot contains three subsystems: the body, the legs and the foot.

Body

The robot body's primary function is to hold all the components which transfer the energy from the motors to the legs. It also houses electronics including the leg encoders and the IMU. Finally, it connects to the boom assembly through the roll bearing housing.

Legs

The robot legs' primary function is to support the weight of the robot while transferring the rotation of the shafts into linear motion, allowing jumping and balancing. The lengths of the legs were carefully selected through an optimization program to allow for the most power and motion, while needed the least power.

Foot

The robot foot's primary functions are to hold the bumper and force sensor. The foot system allows for a softer landing and easier control through the use of a round rubber bumper. The foot system also passes the majority of the in axis force from the jumping through the force sensor, allowing for a rough tracking and collection of forces. The force sensor will eventually be replaced by a much nicer one currently on the old single axis system, so the foot system is only a temporary solution and will be redesigned.

6.1.2 Holder

The holder contains three subsystems: the boom, the base, and the counterweight.

Boom

The boom's primary function is to connect the robot with the base in the lightest way possible. Carbon fiber is used due to its high strength to weight ratio.

Base

The base's primary function is to allow for the movement of the robot and boom in a controlled manner with the option of locking any of the 3 axis (roll, pitch, and yaw).

Counterweight

The counterweight's primary function is to offset the weight of the boom, with the option of adding more weight to simulate different gravities. The design allows for both fine and coarse adjustment through the addition of both small and large weights.

6.2 Electrical

Parts of the electrical system are on the robot itself, on the base of the boom, and separate from the mechanical system entirely. On the robot, there are three motor control Tiva boards, one of the boom encoder Tiva boards (to capture the roll of the robot relative to the boom), and the Tiva board for the force sensor and IMU. At the base of the boom, there are two more of the boom encoder Tiva boards, to capture the pitch and yaw of the boom. Away from the base of the boom, there is a Raspberry Pi, which runs the main controller, as well as the three Copley Accelus motor drivers, the bank of 12 V batteries, and the power switches for the 48 V and 5 V supplies. The CAN bus runs from the Raspberry Pi to the base of the boom and out along the boom to the other group of boards, along with +5 V power, ground, and PWM signals for the Copley amplifiers. Cables for hall signals and motor phases also run along the boom between each motor and its corresponding amplifier.

Each of the custom Tiva boards uses a TM4C123G microcontroller and serves as a node on the CAN bus of the robot. The three boom encoder boards communicate over SPI with AEAT-6600 magnetic encoders to track the angles of the three degrees of freedom of the boom, and then report the angles back over the CAN bus to the Raspberry Pi. In a similar manner, the board for the IMU and force sensor communicates over I2C with the IMU and reads an analog signal from the force sensor, and then reports the data back over the CAN bus. The three motor control boards communicate over SPI with RLS Orbis magnetic encoders to track the positions of the output shafts for each of the three motors. Each board also sends a PWM signal to the one of the Copley amplifiers that controls the current that the the Copley tries to put through the motor. The boards can use the PWM output to the Copley amplifiers to close a position control loop using reference positions received from the Raspberry Pi over the CAN bus and the encoder signals, or enter a current control mode based on commanded currents which are also received from the Raspberry Pi over the CAN bus.

The Tiva boards and the Raspberry Pi are powered from a 5 V, 10 A wall supply. The only switch in the 5 V power line is a small barrel jack. The Copley amplifiers, and thus the motors, are powered from a group of 4 $\times$ 12 V, 35 Ah batteries that are wired in series. There is one large switch in the +48 V power line, as well as an 80 A circuit breaker, which acts as a second switch. In accordance with the Copley amplifier datasheet, there are fuses on the power input and two of the three motor phase outputs. In the current

configuration, these fuses are all 35 A fuses, which would protect the amplifiers in the case of a short but are well above the stall current of the motors.

6.3 Software

The firmware on the robot is an example of distributed hierarchical control. A high-level task space controller runs at 500 Hz in one thread of the main program, `main.c` on the Raspberry Pi, and it sends commands to lower-level torque/position controllers running at 1 kHz on the motor control Tivas.

6.3.1 Raspberry Pi firmware

The main program on the Raspberry Pi, `main.c`, uses a number of libraries, including

- `pthreads`[1] for multithreading,
- SocketCAN[2] for interfacing with the CAN bus,
- GNU Scientific Library[3] (GSL) for matrix math and interpolation, and
- WiringPi[4] to make the Pi's GPIO pins accessible to userspace programs like `main.c`.

The motivation for making `main.c` a multithreading program is to separate timing-critical control computations from slower and less crucial tasks like communicating with the client via UART. These threads share a circular buffer; the control thread puts data in the buffer and the UART thread reads data from the buffer.

Should computation speed become a limiting factor, we recommend replacing GSL with Automatically-Tuned Linear Algebra Software[5] (ATLAS), which has already been set up on the Pi. Building ATLAS on the Pi required us to disable CPU throttling, effectively overclocking the Pi: by default, it runs at 700 MHz; we are running it at *1.2 GHz*, so it is essential to actively cool the Pi.

6.3.2 Tiva firmware

Each Tiva program resembles a simplified version of the Pi's `main.c`. The firmware is built on Texas Instruments' `Tivaware`[6] driver library.

Motor control firmware

The motor control firmware is built around a 1 kHz timer-driven interrupt. An asynchronous process reads from the CAN bus and stores received commands (either position or current references)in a global variable. The interrupt service routine (ISR) checks the global variable, reads from a RLS Orbis encoder over SPI, and calculates a current required to track the reference signal. This current command is sent to the appropriate Copley Accelus amplifier via a 25 kHz PWM signal.

Boom encoder firmware

Like the motor control firmware, the boom encoder firmware is also built around a timer-driven interrupt, currently set to 10 Hz. The ISR reads from an AEAT-6600 encoder over SPI, converts the reading to tenths of a degree, and writes the converted value to the CAN bus. If the robot is using boom angles in any control computations, the boom interrupt frequency will need to increase.

[1] `http://man7.org/linux/man-pages/man7/pthreads.7.html`
[2] `https://www.kernel.org/doc/Documentation/networking/can.txt`
[3] `https://www.gnu.org/software/gsl/`
[4] `http://wiringpi.com/`
[5] `http://math-atlas.sourceforge.net/`
[6] `http://www.ti.com/tool/SW-TM4C`

IMU and force sensor firmware

The firmware for the IMU and force sensor Tiva is also built around a timer-driven interrupt, currently set to 100 Hz. The ISR reads from an analog-output force sensor and (for now) from a STM LSM6DS33 IMU over I2C. The ISR then writes the measured force and accelerations to the CAN bus. If the robot is using these values in any control computations, this interrupt frequency will also need to be increase.

6.3.3 Client

The client program is very minimal, with the understanding that the end user's research tasks will dictate most of the structure of the client. At present, the client provides a serial interface with the Raspberry Pi and some basic plotting functions to visualize data received from the Pi.

Chapter 7

Testing Procedures and Performance Results

By the end of the winter quarter, it was evident to the team and the instructors that high level control of the robot before the spring final was a very ambitious task. Instead we shifted our focus to developing the robust low level functionality that would be required to build up the desired high level behavior in the future. At this point, the primary goal for the end of spring quarter was to do an open loop jump to demonstrate the potential of the robot in the future upon development of high level controllers.

Before going straight to a jump, we wanted to validate our kinematic models and motor selection by finding the current required in the motors to support the robot's full 5 kg weight. This requires a 50 N force at the foot, mapping through the actuator Jacobian to about 1.5 Nm in each of the two outside legs. Due to the 1.5:1 pulley ratio between the legs and the motors, this means we expected the outside two motors to supply 1 Nm of torque, corresponding to 4.6 A.

We tested this by putting the robot in position hold mode, commanding a fairly extended leg position, setting the full weight of the robot on the leg, and reading the current flowing through the windings. In this setup, the outside motors required between 4.0 - 5.0 A, verifying our expected torque requirements.

Once we were confident in our kinematic model and our motor selection, we began our first attempts at a feed-forward jump by commanding a current impulse to the two outside motors. We began in position hold mode in a squatted position, then switching to current control mode, increasing the command magnitude and duration conservatively until a satisfactory jump was achieved. Ultimately, when reaching an command of 19 A over 0.15 s, we decided that this was the maximum power we were comfortable providing at this point without any additional control or safeties. Nonetheless, as shown in Figure 7.1, slow motion video footage of one of these jumps shows a maximum jump height of about 3.5 inches (9 cm), narrowly missing our target of 10 cm.

Figure 7.1: Still frame showing maximum height (9 cm) achieved in an open-loop jump

The team is optimistic about this result for several reasons. First, our feedforward current impulses are perhaps the least controlled and most unintelligent way to achieve a jump. The actuator Jacobian that maps the actuated joint torques to wrenches at the foot is configuration dependent, and therefore it is difficult to predict and control the jump and the applied forces without any kind of control strategy.

Second, our jumps were done using only the two outside motors - the center motor was powered off during the jump tests. While it is convenient to think of this center motor as only controlling the angle of the foot while the two outside motors provide the power for the jump, this is an oversimplification - utilizing the center motor would surely add to the jump height.

Even a simple controller for this motor would greatly help the jump, as we noticed that some of the motors' power went towards a small component of horizontal velocity, rather than all towards the vertical velocity as desired. We think that even a simple position control of this motor to keep the bottom leg vertical while the two outside motors are producing high torque would help keep the leg forces vertical, resulting in a higher jump.

From our kinematic model developed during the first quarter, we had expected to require 12.6 A from each of the two outside motors to achieve a 10 cm jump, but our jumps required 19 A - about a 50% difference. We cannot definitively account for this discrepancy, but it is likely that the factors above are responsible. It is encouraging that the robot is able to jump using a current within the same order of magnitude as our prediction, but this should be reevaluated once a better jumping strategy is implemented.

Chapter 8

Suggested Next Steps

The following considerations and next steps must be taken to ensure both the short term and long term success of the design.

8.1 Mechanical

8.1.1 Modified motor shaft/drive shaft coupling

As we began increasing the loads on our motors, we found that the Maxon motor shafts would often decouple from the drive shafts. We attribute this failure to our use of set screws to transmit these high torques. Several different levels of modifications should be made to improve this coupling:

- Add a second set screw: The timing pulleys use two set screws, and we have not yet seen these decouple from the shaft. Perhaps this may be the easy solution.
- Add flats to the motor shaft: Flats would increase the gripping strength of the set screw.
- Use a key and keyway: This is the most rigorous mechanical solution, but would require non-trivial modifications to the two coupled shafts.

8.1.2 Integrate new force sensor

The force sensor currently on our robot is a simple and cheap 1-axis sensor. It can be useful in the short term as a ground detection device, but ultimately it should get replaced by the 6-axis force sensor that is currently on the 1-DOF hopping robot in the NxR lab.

8.1.3 Design for assembly

Currently, mechanical assembly is a two-person job and takes about 2 hours to complete. Several DFA considerations could be taken to improve the assembly process, such as designing a jig for belt tensioning.

8.1.4 Harness for supporting the robot

Executing a jump currently is also at least a two person process - one to man the controls and emergency stop, and another (or two) to support the robot and keep it upright. Until the robot is intelligent enough to stand, balance, jump, and land on its own, a harness is required to keep the robot from crashing into the floor. This would allow the robot to be operated by one person only.

8.1.5 Patch the carbon fiber tube

An unfortunate accident during assembly resulted in about a 1 inch long crack in the carbon fiber boom. This was patched temporarily with duct tape, but this should be properly repaired with a commercial carbon fiber patch.

8.2 Electrical

8.2.1 Future board iterations

Version 4 of each of the custom PCBs will be required for several necessary upgrades and luxury items:

- Prepare for an upgraded IMU and force sensor
- Break out additional digital and analog I/O pins, for flexibility in future upgrades
- Add +5 V and GND connections to the programming headers to streamline the programming process. This would remove the need to power all boards while programming, and gives a convenient location to connect common ground between the custom board and the programming Tiva.

8.2.2 Robustify cable connectors

Most our cables use either screw terminal blocks or 0.1" headers to connect and transmit signals. These were convenient for us to use in the short term, but are not the most robust and they have the potential for the user to plug in a cable backwards. It would be nice to use polarizing Molex connectors at these locations.

8.2.3 Cooling the Raspberry Pi

The temporary fan/heatsink assembly on the Raspberry Pi should be replaced with a more professional active cooling system. Better cooling can also be achieved by attaching the Pi's CAN adapter by running wires instead of mounting it directly over the Pi's hot processor.

8.3 Software

8.3.1 Motor control Tiva boards

- Implement configuration dependent PID gains for each controller, or a different position controller altogether (such as a task-space pose controller). Due to the closed linkage, it is possible for the motors to fight each other while moving to a position. A controller should be implemented that ensures cooperation between the motors in this mode.
- Estimate the motor winding temperatures. With our thermal model, this can be estimated with a Kalman filter (see Section 4.2.2).

8.3.2 Raspberry Pi

The software on the Raspberry Pi is primarily used to implement the robot controllers. Short term goals include writing stable controllers for balancing and landing a jump. These are prerequisites for developing more sophisticated controllers required for locomotion on soft ground.

Should computation speed become a limiting factor, we recommend replacing GSL (see Section 6.3.1) with Automatically-Tuned Linear Algebra Software (ATLAS), which has already been set up on the Pi and is available at `http://math-atlas.sourceforge.net/`

From Section 4.1.2, we recommend searching or devising a forward kinematics algorithm that is free from the jump discontinuities of `geomFK` and the unbounded execution time of `NRFK`.

8.3.3 Client

- Full data logging. The user should be able to read any recorded data from a experiment into a high level analysis software like MATLAB.
- Interface with the dynamic model. As more complex controllers are written, it would be useful to use `trep`[1] to validate some behaviors before executing them on the robot.

[1] `http://murpheylab.github.io/trep/`

- (optional) Add a second serial interface to allow the end user to connect to any of the Tivas' UART modules for debugging, data logging, and visualization.
- (optional) Interface with Robot Operating System (ROS). This will be easier if the client program is ported to Python.

Chapter 9

Other Documentation

- Assembly Instructions
- User Guide
- Software Documentation
- Quickstart Instructional Video
- “Belts” calculations
- “Shafts” calculations

Bibliography

[1] Frank P. Incropera. *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, Inc., USA, 2006.

[2] C. Li, T. Zhang, and D. I. Goldman. A terradynamics of legged locomotion on granular media. *Science*, 339(6126):1408–1412, 2013.

[3] K. M. Lynch and F. C. Park. *Modern Robotics*. Cambridge University Press.

[4] Maxon Motor. 244879. `https://www.maxonmotor.com/medias/sys_master/root/8825435389982/17-EN-271.pdf`, 2017. Datasheet from Maxon's online catalog.