Kangaroo Robot

Brandon Simoncic & Eddie Doemer

December 15, 2018

1 Background

As the desire for discovery increases, so does the innovation needed to make those discoveries possible. With the Curiosity rover nearing the end of its use, a new rover must take its place to take on the next challenge; analyzing the entire equator of mars in 45 days. This new rover must abandon traditional rover design to meet the speed and versatility of the task. Considering the environmental characteristics of Mars, the proposed rover design takes into consideration the dynamics of walking and rolling along with the energy requirements to complete its task.

1.1 Environment of Mars

The climate of Mars is dry and sterile. Small dust particles are suspended in the air which can find itself into actuators and short circuitry. The temperature may fluctuate between 20 C and -70 C which is important to consider when determining how to store energy on the rover. The wind should be no greater than 60 MPH with a density of 1% that of the earth. Since the density is so low, it makes flying rovers very ineffective on Mars. The wind should not produce any particularly large forces on mechanical structures of the rover. The greatest danger of the wind is the dust that may be carried with it. To combat the dust, flexible covers should be applied over the joints of the robot, or closed bearings are used for a rotating component. If using solar panels as energy, we must consider the amount of sunlight that can reach the solar panels during periods of dust storms. Since the objective is to measure the terrain around the equator, the rover cannot deviate far to avoid the maximum 1-meter rocks in the way. Therefore the rover must be able to climb, jump, or roll over the rocks. A traditional rover would have significant difficulty traversing 1 [m] high obstacles without significant size or flexibility. Jumping robots may have too high of a failure rate. A walking robot may not be able to keep the 12 MPH average velocity needed to meet the mission time frame. Therefore, a hybrid option is proposed below.

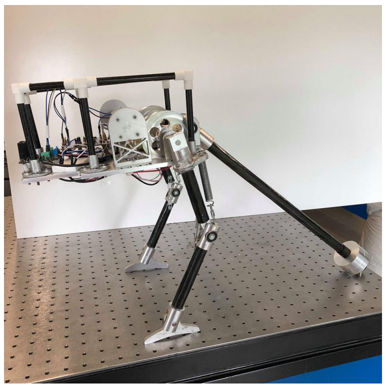

Figure 1: This the the second iteration prototype developed at LTU by Hamid Vejdani and Austin Curtis

2 Jumping Morphism

We propose the morphism of a kangaroo to meet the proper requirements of speed and height constraints.

2.1 Design

Figure 2.1 displays an image of the second iteration prototype of a kangaroo robot that our design is primarily based off of. Looking furthermore into figure 2.1 the frame,tail, and legs are all constructed from carbon fiber and high grade aluminum. In theory, the robot will be able to leap 1 [m] off of the ground and maintain parallel symmetry to the ground by using the tail and it's legs as a counter balance in flight. The total mass of the robot is 3.60 [Kg], the body weighing in at 2.X [Kg], the tail at 0.52 [Kg], and each leg weighing in at 0.25 [Kg]. The robot itself is 29.21 [cm] tall, the tail is at a length of 43.18 [cm], with the body centroid at 10 [cm] from the tail's pivot point.

2.2 Design Goals

Our design parameters of 12 [m/s] and jump clearance 1 [m] requires a further calculation. Using 1 and 2, determining ω_n and ζ became trivial.Next was to determine the angle of flight, the design parameters led to a preferred angle of 77.125deg. All of these parameters would then be used to calculate the poles and their respective pole placement. Although we did not use these parameters within the simulation, we used the Matlab function Ldqr(), this will later be explained why in section 3.3.

$$t_s = \frac{4}{\zeta\omega_n} \tag{1}$$

$$\omega_n = \frac{4t_s}{\zeta} \tag{2}$$

3 Equation of motion

Once we found the equations of motion for the system, we needed to linearize them to correctly tune a controller. Using the small angle approximation on the previous equation 6, the distance is proportional to the angle from the center point. The same small angle approximation was applied to equation 7, even though larger angles are expected. A very heavy tail could be added to the robot to keep the angles of movement small, however, we decided to try and tune a controller assuming it is linear, and check if it works with the non-linear system. Since the small angle approximation for equation 8 would not linearize the system, we assumed a constant small and linear change depending on theta. Again, the controller would need to be tuned more to work with a non-linear system even if it worked with the linear simulation. Equations of inertia 3,4, and 5 are shown below.

$$I_B = \frac{M_B * L_B^2}{3} \tag{3}$$

$$I_T = (\frac{M_T}{3} + M_M) * L^2 \tag{4}$$

$$I_L = \frac{M_L * L_L^2}{3} \tag{5}$$

$$\ddot{\theta_B} = \frac{\tau_T + \tau_L}{I_B} - \frac{P_B * \theta_B}{I_B} \tag{6}$$

$$\ddot{\theta_T} = \frac{\tau_L}{I_T} - \frac{P_T * \theta_T}{I_T} \tag{7}$$

$$\ddot{\theta_L} = \frac{\tau_L}{I_L} \tag{8}$$

3.1 State Space Formulation

Once the equations of motion were linearized, we put them into our A and B matrices(3.1 & 3.1). The coefficients for the torques were placed in matrix 3.1, while the coefficients for the velocities were placed in the A matrix. The value 1 was placed where necessary for the other rows. After a state-space representation was found, it was discretized with the c2d() function using Zero-order hold and a sampling time of Ts = 0.01.

$$\begin{bmatrix} 0 & \frac{-P_B}{I_B} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{P_T}{I_T} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{0.1*P_L}{I_L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & \frac{1}{I_B} & 0 & \frac{1}{I_L} & 0 & 0 \\ 0 & \frac{1}{I_B} & 0 & 0 & 0 & \frac{1}{I_T} \end{bmatrix}$$

3.2 State Space Representation

Once our state-space system was found, using LQR from the function dlqr(), we could design our controller. First, our Q matrix 3.2 was set up to penalize the controller if the body moved away from its set point. The velocity of the body was also penalized to avoid fast fluctuations around the desired position. Both the legs and tail were left very low, while the tail's velocity was decreased even more. For the R matrix 3.2, we set the element corresponding to T_T to 1 to allow the tail to use as much energy as needed. The element corresponding to T_L was set extremely high to keep the controller from using it to orient the body. This should set the gains for the second row of your k matrix close to 0, suppressing control input to the leg. We then used the equation 9 to obtain our closed loop system with controller added. Using the new G, a state-space representation for the whole system was found.

$$\begin{bmatrix} 100K & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & 10K & & & & \vdots \\ \vdots & & 1 & & & \vdots \\ \vdots & & & 1 & & \vdots \\ \vdots & & & & 1 & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 100K \end{bmatrix}$$

$$G_k = G - H * k \tag{9}$$

3.3 Simulation

Using the lsim() function, we simulated the response of the linearized system and controller using no input for the tail and a cosign force input for the leg. The Frequency of the cosign input would be more than what's necessary to move the leg during flight phase. After simulating the response, the body was

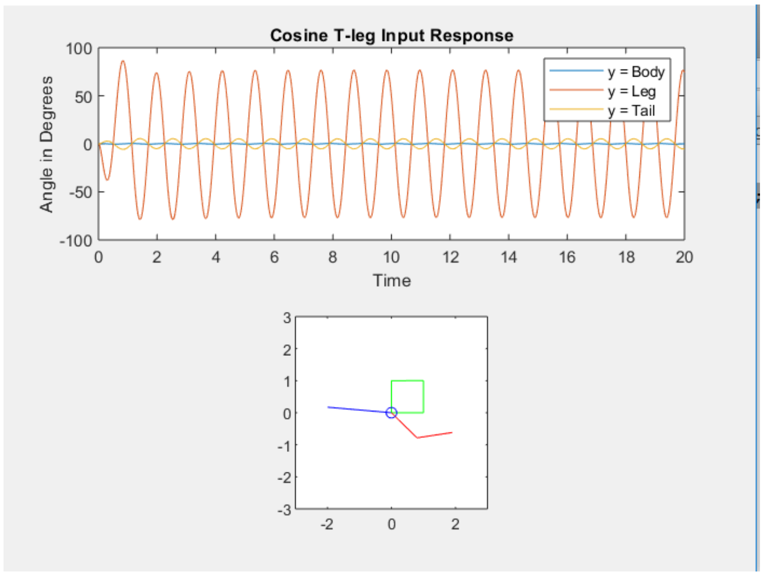

Figure 2: This the the second iteration prototype developed at LTU by Hamid Vejdani and Austin Curtis

controlled to be very close to 0 degrees 3.3. These could then be plotted and used to create a simulation.

3.4 Visualization

To visualize the actuation of the tail, leg, and body are a set of point clouds with their own distinct color and shape to resemble the robot in figure 2.1. Common practice of visualization and simulation of moving point clouds is to translate back to the origin, rotate about that, then to translate the image back to its original center. The lower half of the controllers code is dedicated to the two-dimensional simulation of the kangaroo robot.

3.5 Errors in calculation

Our biggest issue of this project was that we were unable to zero-out the last row of our ldqr() matrix. This intern meant that we were unable to fully implement foot-forward input pulses. We also struggled with ode45 function of Matlab to achieve a full understanding and simulation of the non-linear system. In future iterations we hope to simulate the non-linear system with foot forward input pulses.