Final Project –hybrid dynamics of a robotic hopper

We declare that the project was solely done on our owns. And the main source of resource is based on the lecture notes.

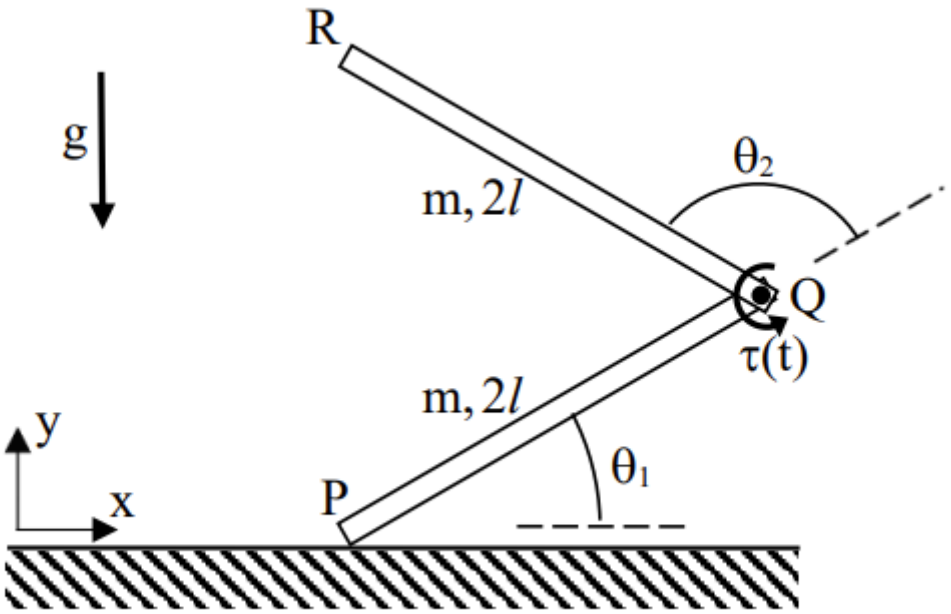
Name	ID
Jonathan Oh	941170383
Xingkun Huang	941164840

ALL THE CODE IS AT GITHUB (Downloadable):

garlicbutter/HybridDynamicsFinal: MATLAB code and the final report for the class "036087 - Hybrid Dynamics in Mechanical Systems - Spring" (github.com)

Configuration

Our robot has following configuration. It's a simplified model of a robotic hopper composed of two identical links connected by a rotational joint which is actuated by a torque.



We choose the generalized coordinates to be $\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta_1}, \boldsymbol{\theta_2})$ where x, y is the position of the endpoint P and θ_i are the relative angle between the links. During the entire motion, the actuation torque τ satisfies the bound $\tau \leq \tau_{max}$. The physical values for the simulations are: $m = 0.3kg,\ g = 10\,{}^{m}/_{s^2},\ l = 0.15m,\ \mu = 0.3,\ \tau_{max} = 10Nm$

Our objective of the project is to formulate the hybrid dynamics of the robot in all possible contact states, the conditions for transitions between contact states, and the impact law. Also, we will write MATLAB code functions for dynamic simulations of "jumping forward from rest" under a prescribed actuation torque. Finally, we will choose time-profile of actuation torque in attempt to maximize the jump distance d.

Task

1. For generalized coordinates $\mathbf{q}=(x,y,\theta_1,\theta_2)$, write a detailed derivation of the robot's constrained equations of motion in matrix form: $\mathbf{M(q)\ddot{q}} + \mathrm{B}(\mathbf{q},\dot{\mathbf{q}}) + \mathrm{G}(\mathbf{q}) = \mathrm{F_q} + \mathbf{W(q)}^T\boldsymbol{\lambda}$. The coordinates (x, y) which denote position of the endpoint P, are constrained during contact. Write the conditions for consistency of each contact state (stick, slip, separation) and conditions for transitions between contact states.

First, we lay out the kinematics of the system. We find the center of gravity of each link. We denote the lower link as link number 1 and the upper one as 2. CG is the center of gravity of the robot.

$$\boldsymbol{r_{1c}} = \begin{bmatrix} x + lc_1 \\ y + ls_1 \end{bmatrix} \; where \; c_i = \cos(\theta_i), s_i = \sin(\theta_i)$$

$$\boldsymbol{r_Q} = \begin{bmatrix} x + 2lc_1 \\ y + 2ls_1 \end{bmatrix}$$

$$\boldsymbol{r_{2c}} = \boldsymbol{r_Q} + \begin{bmatrix} lc_{12} \\ ls_{12} \end{bmatrix} = \begin{bmatrix} x + 2lc_1 + lc_{12} \\ y + 2ls_1 + ls_{12} \end{bmatrix} \; where \; c_{12} = cos(\theta_1 + \theta_2)$$

$$\boldsymbol{r_{CG}} = \frac{\begin{bmatrix} x + lc_1 \\ y + ls_1 \end{bmatrix} + \begin{bmatrix} x + 2lc_1 + lc_{12} \\ y + 2ls_1 + ls_{12} \end{bmatrix}}{2} = \begin{bmatrix} x + \frac{3}{2}lc_1 + \frac{1}{2}lc_{12} \\ y + \frac{3}{2}ls_1 + \frac{1}{2}ls_{12} \end{bmatrix}$$

Their respective velocities are:

$$\boldsymbol{v_{1c}} = \begin{bmatrix} \dot{x} - l\dot{\theta}_1 s_1 \\ \dot{y} + l\dot{\theta}_1 c_1 \end{bmatrix}$$

$$\boldsymbol{v_{2c}} = \begin{bmatrix} \dot{x} - 2l\dot{\theta}_1 s_1 - l(\dot{\theta}_1 + \dot{\theta}_2)s_{12} \\ \dot{y} + 2l\dot{\theta}_1 c_1 + l(\dot{\theta}_1 + \dot{\theta}_2)c_{12} \end{bmatrix}$$

Then we can obtain the kinetic energy of the system.

$$T = \sum_{i}^{n=2} \frac{1}{2}(m_i v_{ic}^T \cdot v_{ic}) + \frac{1}{2}I_c\dot{\theta}_1{}^2 + \frac{1}{2}I_c(\dot{\theta}_1 + \dot{\theta}_2)^2$$

The expression is long, and we won't display it. The exact expression can be calculated via MATLAB. As for the potential energy,

$$V = 2mgr_{CG}\hat{\boldsymbol{y}} = 2mg(y + \frac{ls_{12}}{2} + \frac{3ls_1}{2})$$

Then we can write the EOM in a matrix form via Lagrange method.

$$M = \frac{\partial^2 T}{\partial \dot{q}_i \partial \dot{q}_j}$$

$$B = \left(\sum_j^n \frac{\partial^2 T}{\partial \dot{q}_i \partial q_j} \dot{q}_j \right) - \frac{\partial T}{\partial q_i}$$

$$G = \frac{\partial V}{\partial q_i}$$

$$M = \begin{bmatrix} 2m & 0 & -lm(s_{12} + 3s_1) & -lms_{12} \\ 0 & 2m & lm(c_{12} + 3c_1) & lmc_{12} \\ -lm(s_{12} + 3s_1) & lm(c_{12} + 3c_1) & \frac{4}{3}l^2 m(3c_2 + 5) & \frac{2ml^2}{3}(3c_2 + 2) \\ -lms_{12} & lmc_{12} & \frac{2ml^2}{3}(3c_2 + 2) & \frac{4}{3}ml^2 \end{bmatrix}$$

$$B = \begin{bmatrix} -lm(\dot{\theta_1}^2 c_{12} + \dot{\theta_2}^2 c_{12} + 3\dot{\theta_1}^2 c_1 + 2\dot{\theta_1}\dot{\theta_2} c_{12}) \\ -lm(\dot{\theta_1}^2 s_{12} + \dot{\theta_2}^2 s_{12} + 3\dot{\theta_1}^2 s_1 + 2\dot{\theta_1}\dot{\theta_2} s_{12}) \\ -2ml^2 \dot{\theta_2} s_2 (2\dot{\theta_1} + \dot{\theta_2}) \\ 2ml^2 \dot{\theta_1}^2 s_2 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ 2mg \\ lgm(c_{12} + 3c_1) \\ lgmc_{12} \end{bmatrix}$$

As for the constraint matrix $\boldsymbol{W}$, we would require the velocity at point P.

$$r_P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$v_P = \frac{\partial r_P}{\partial t} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \dot{q}$$

$$W = \begin{bmatrix} W_t \\ W_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\lambda = \begin{bmatrix} \lambda_t \\ \lambda_n \end{bmatrix}$$

$$\lambda = (WM^{-1}W^T)^{-1}(WM^{-1}(B + G) - \dot{W}\dot{q})$$

$$\lambda_{n,slip} = \frac{\beta}{\alpha} = \frac{W_n M^{-1}(B + G) - \dot{W}_n \dot{q}}{W_n M^{-1}(W_n - \sigma\mu W_t)^T}$$

Condition for transition:

No-slip	$v_n = v_t = 0, \quad \lvert\lambda_t\rvert \le \mu\lambda_n$
No-slip to slippage	$\lvert\lambda_t\rvert = \mu\lambda_n$
Slipping	$v_n = 0, \quad v_t \neq 0, \quad \lambda_n \ge 0$
Slipping to separation	$\lambda_n = 0, \quad \dot{\lambda_n} < 0$
Separation	$\lambda_n = \lambda_t = 0$
Separation to re-impact	$d = 0, \quad \dot{d} < 0$

2. Write MATLAB functions for calculating the state equation $\dot{\mathbf{X}} = (\dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \mathbf{f}(\mathbf{X}, t)$ under each of the contact states, and write event functions for detecting the conditions for contact state transitions. These functions make use of another external function which prescribes the actuation torque $\tau(t)$ and has the syntax: function tau=calc_tau(t) .

There are three phases for this configuration. To use ODE 45, we need equation of motion for each phase. For each ODE45, we chose an appropriate RelError and AbsError.

- Stick:

$$EOM: \ddot{q} = M^{-1}(W^T\lambda + F_q - G - B)$$

- Slip:

$$EOM: \ddot{q} = M^{-1}((W_n - \sigma\mu W_t)^T\lambda_{slip,n} + F_q - G - B)$$
$$\lambda_t = -\sigma\mu\lambda_{slip.n}$$

- Contact separation:

$$\ddot{q} = M^{-1}(F_q - B - G)$$

For MATLAB code, we need event function to detect when the transition occurs, and we can terminate the phase calculation and move to a new state.

- Event function for sticking motion:
  1. $\lambda_t - \mu\lambda_n$: for transition to slipping
  2. $-\lambda_t - \mu\lambda_n$: for transition to slipping
  3. Height of point R : for detecting the termination condition for the simulation
  4. Height of point Q: for detecting the termination condition for the simulation
  5. λ_n : for transition to separation

- Event function for slipping motion:
  1. $\lambda_{n,slip}$: for separation
  2. v_t : for checking reverse condition

3. Height of point R: for detecting the termination condition for the simulation
4. Height of point Q: for detecting the termination condition for the simulation

- Event function for flying motion:
  1. Height of point P : to check if P landed first
  2. Height of point R : to check if R landed before P
  3. Height of point Q : to check if R landed before Q
  4. Angle constraint : $60°$ constraint of line PR and y axis
  5. Folding constraint : $\theta_2 \neq \pi$
  6. R is higher than Q
  7. Q is higher than P

3. Write MATLAB function of the form Xnew=impact_law(Xold) for calculating the change in the state vector due to collision of the link's endpoint P with the ground. Use Chatterjee's impact law and assume a frictional fully-plastic collision (e_n=e_t=0). Explain.

According to what we learned in the class, we write the function Xnew=impact_law(Xold) which Xold offers us the system's coordinates at collision time q_c, therefore we can obtain W_c and $M(q_c)$. The Impulse-momentum balance during collision is written as:

$$\mathbf{M}(\mathbf{q}_c)\Delta\dot{\mathbf{q}} = \mathbf{W}_c{}^T\boldsymbol{\Lambda} = \mathbf{w}_t(\mathbf{q}_c)^T\Lambda_t + \mathbf{w}_n(\mathbf{q}_c)^T\Lambda_n$$

And A is defined as:

$$\mathbf{A} = \mathbf{W}_c\mathbf{M}_c^{-1}\mathbf{W}_c^T$$

And Λ is the impulse vector and it's defined as follows:
For frictionless impact:

$$\Lambda^I = [0 \quad -\frac{v_n^-}{A_{22}}]^T$$

For completely plastic impulse:

$$\Lambda^{II} = -A^{-1}v^-$$

We can generate the first and second impulse vector from A, and after that we can obtain the new desired impulse under the frictional bound:

$$\boldsymbol{\Lambda} = (1+e_n)\boldsymbol{\Lambda}^{\mathbf{I}} + \kappa(\boldsymbol{\Lambda}^{\mathbf{II}} - \boldsymbol{\Lambda}^{\mathbf{I}})$$

Where:

$$. \kappa = \begin{cases} 1+e_t & \left|\hat{\Lambda}_t\right| \le \mu \hat{\Lambda}_n \\ \dfrac{\mu(1+e_n)\Lambda_n^I}{\left|\Lambda_t^{II}\right| - \mu(\Lambda_n^{II} - \Lambda_n^I)} & \text{else} \end{cases}$$

Finally, the resulting post-impact velocities can be obtained by substituting into the relations:

$$\Delta\dot{\mathbf{q}} = \mathbf{M}_c^{-1}\mathbf{W}_c^{\ T}\boldsymbol{\Lambda}$$

Which composites part of Xnew as the result of the function impact_law.

> **4.** Write MATLAB function main_jump which runs a simulation of the hybrid dynamics for given initial conditions $\theta_1(0),\theta_2(0)$ starting from rest $\dot{\mathbf{q}}(0)=0$, under a prescribed time-profile of actuation torque $\tau(t)$. The simulation includes motion until contact separation followed by free flight, impact, and additional motion in contact. The simulation is terminated when another endpoint Q or R collides with the ground. The function is required to calculate the jump distance d as described below in item **5**, and to generate the following graphs:

First, we divide the simulations into two parts. Ground motion (Sticking, Slipping) and Aerial motion (Flying). At the beginning of the simulation, we need to check the initial condition to determine the state of the robots. Then we start running the ground simulation **(ground_simulation)** until the separation condition is met. Then the hopper will enter into the aerial simulation(**air_simlation.m**). Upon impact, we need to apply the impact law and also check if the nominal force is positive. If the nominal force is negative, we need to resolve the Painleve's paradox by separating the hopper and execute air simulation until we can safely land on the ground.

File explanation:
animation.m: main_jump that runs the simulation and generates animation.

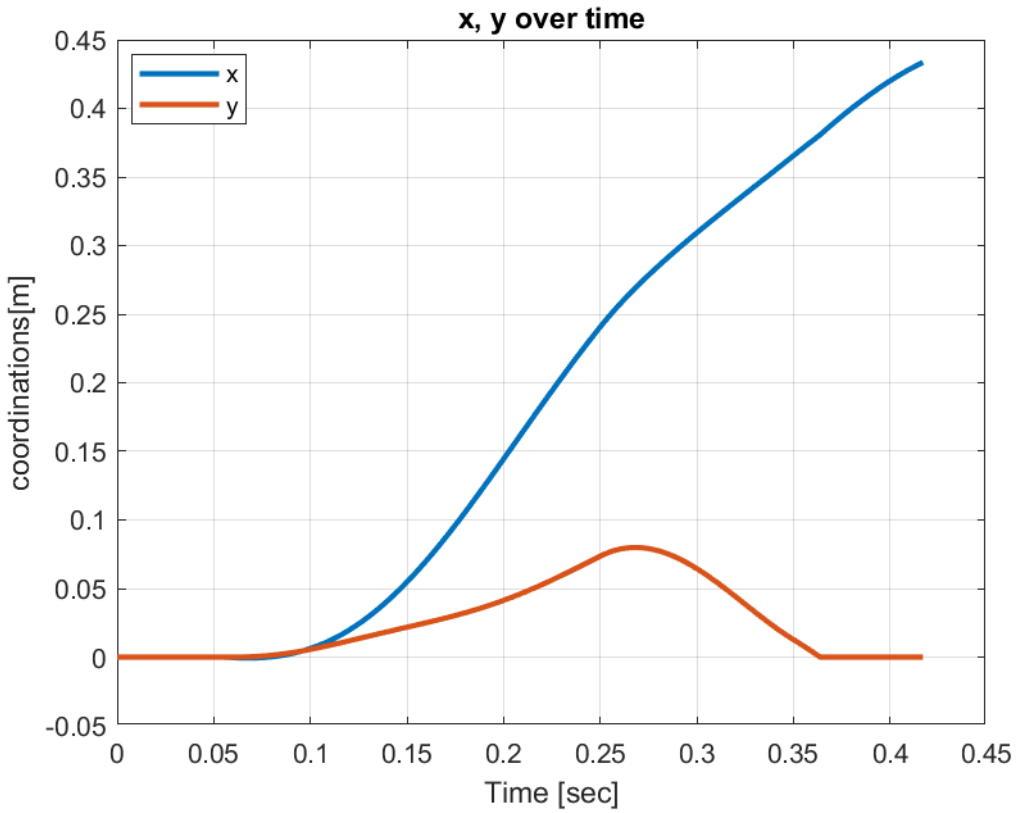
plotter: plot the figures as well as saving them.

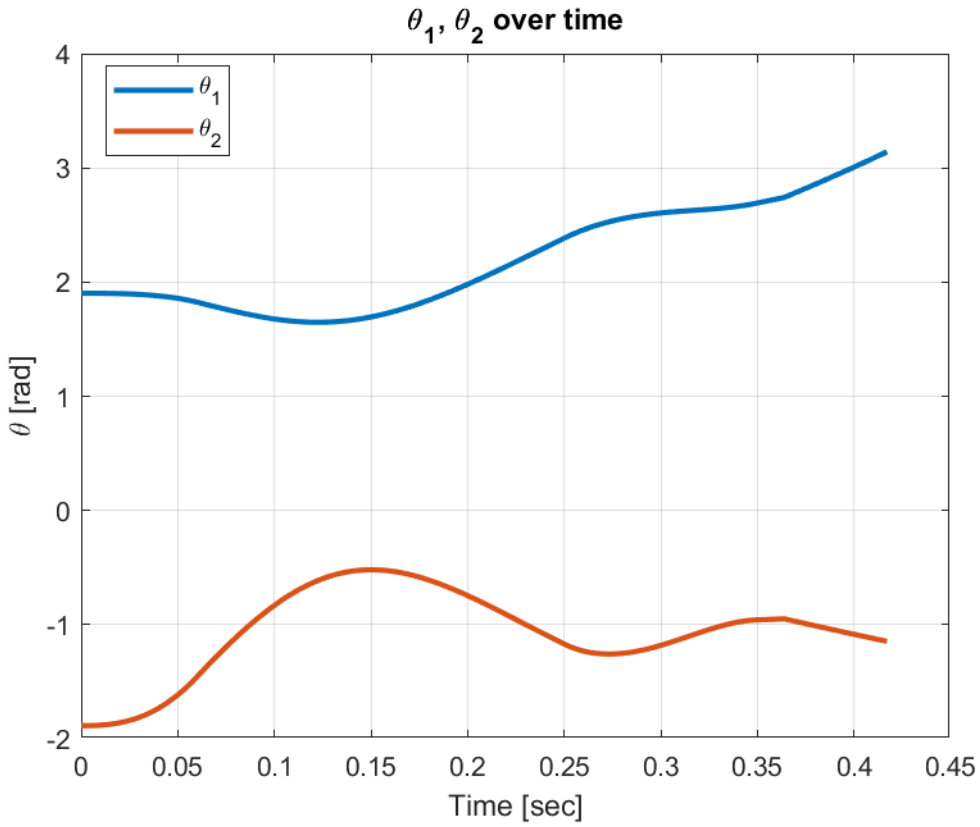
all the file that starts with temp : Automatic function generated by the Symbolic Math Toolbox version 8.6

tau_trial_and_error: log file for all the custom tau_calc that we tried.

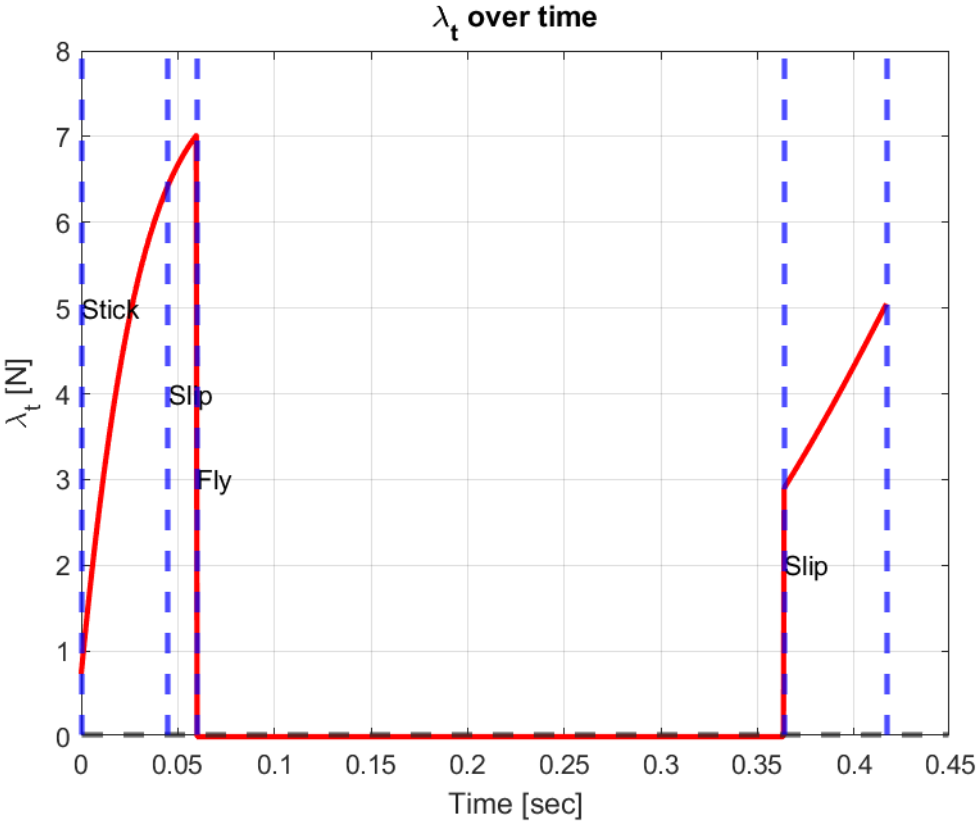
- Time plots of the coordinates **q**(t) as a function of time during the entire motion, (x,y) overlaid on one plot and (θ_1,θ_2) on another plot.

We can see from the y that we only separated from the ground once.



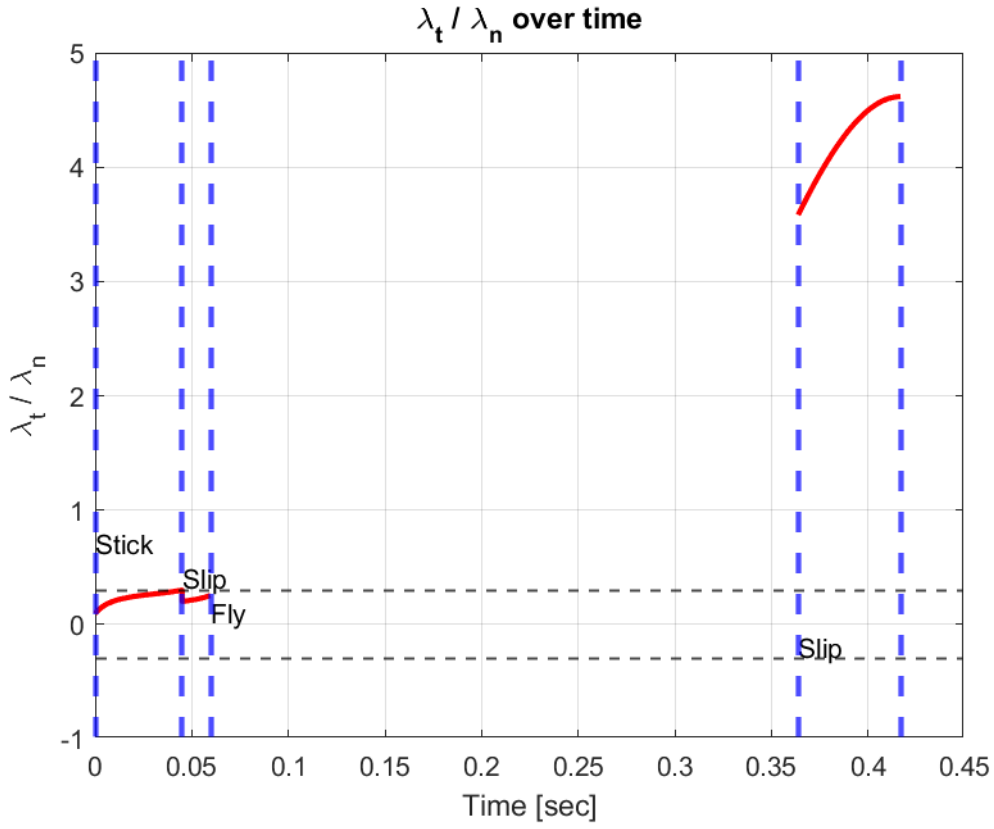


- Time plot of the normal contact force $\lambda_n(t)$ overlaid with a dashed horizontal line at height 0.

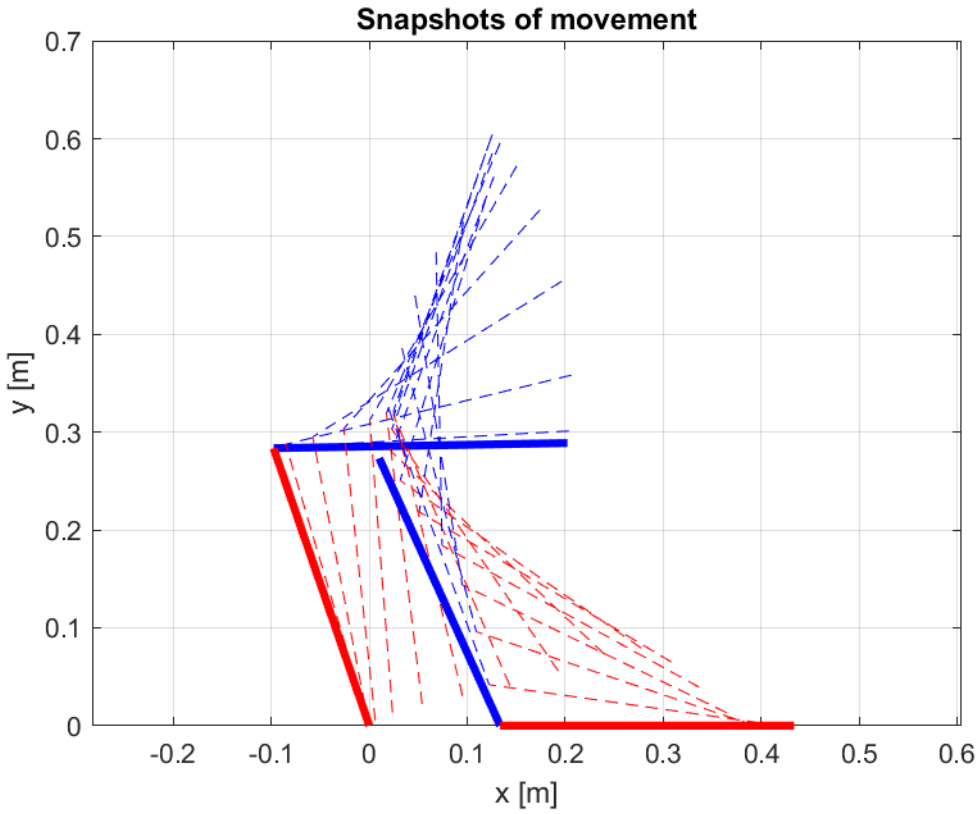


- Plot of the force ratio λ_t/λ_n as a function of time t, overlaid with two dashed horizontal lines at values $\pm\mu$. In all the time plots mentioned above, mark by text labels the time intervals of different contact states, and the transition times by dashed vertical lines.

We can see when the hopper is sticked to the ground, the ratio of λ is between μ.



- A plot of the robot's links at initial time, the contact separation time, two equally-spaced intermediate times during free flight, first impact time, two equally-spaced intermediate times during the second single-contact phase, and at the time of final impact at Q or R. (Use command axis equal for ensuring correct aspect ratio between axes).



5. Choose initial conditions $\theta_1(0),\theta_2(0)$ and actuation torque $\tau(t)$ that attempt to maximize the horizontal jumping distance d. The jump distance is defined as $d=X_{end}-X_{start}$, where X_{start} is the maximal horizontal displacement of the link's endpoint P during the first contact phase until separation. X_{end} is the minimal horizontal displacement of P during the contact phase after the first impact, until the second collision. If Q or R hits the ground behind P, then its horizontal displacement should be used in calculating X_{end} instead of P.

This section comprised of several optimization problem. First, we will have to decide an appropriate initial condition. We notice that we can translate the constraint to the mathematical expression.

- The angle between the line PR to the y axis must not exceed 60o in absolute value. This one is trickier to see but utilizing the characteristic of equilateral triangles we get

$$30 < \theta_1 + \frac{\theta_2}{2} < 150$$

This inequality is helpful for later on since this is one of the strictest constraints.

- The joint Q is always above the endpoint P and below the endpoint R

$$0 < \theta_1 < \pi$$

$$0 < \theta_1 + \theta_2 < \pi$$

These inequalities are useful for choosing a valid initial condition.

- The joint angle θ_2 can never reach the values of ±180º (folded links) during the entire motion

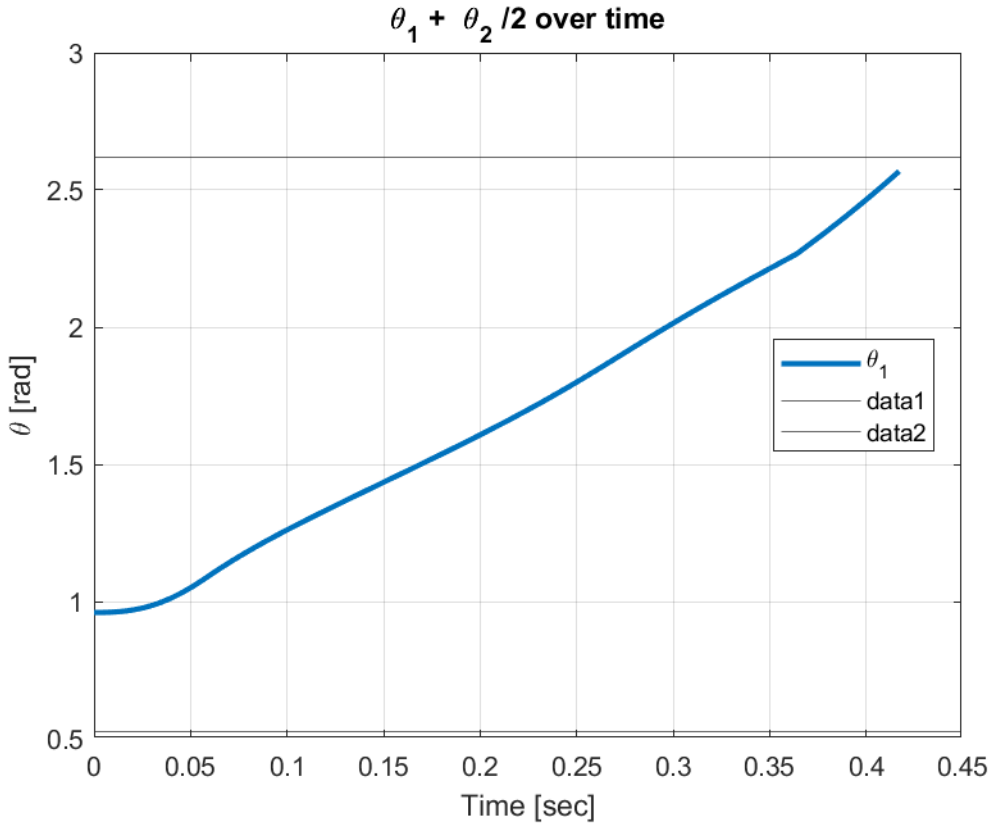
$$\theta_2 \neq \pm 180$$

This restricts on how we can land since many profiles will lead to folded links upon landing.

For the actuation planning, we chose the broad jump as a reference. After analyzing the motion of the athlete, we use the movement of his thigh and calf to simulate the two rods in our project. As it shows in the figure, the movement contains 4 stages: Firstly a negative torque is applied on the joint so that the rods would eject out until the two rods are almost parallel. Then a positive torque is applied on the joint in order to clamp the angle between the two rods while it is flying. This would help extend the flying time of the rods. Before the rods landing, a negative torque should be applied on the rods again which enlarge the angle between the two rods. And in this way, the lower rod could reach as far as possible. In The end of the movement, a negative torque should be applied in order to clamp the angle so that the rod on the left hand side would come closer to the rod on the right hand side as much as possible. When we are measuring the distance of jumping, we should count the last point on the left-hand side, therefore the less the distance between the landing point of the two rods, the better.

To sum, we need to manipulate the motion so that we can change our shape with respect to the center of gravity.

For torque planning, we chose a heuristic approach. We tried to follow the general principle of broad jumping. First, we need a strong torque on the first section to lift the body up in the air. This will be the only source of work done on the hopper since it will leave the ground immediately. We chose a quadratic polynomial to concentrate the maximum torque at the moment of separation. For rest of the part, we tried to maintain balance on the hopper, so we won't breach the constraints. As we mentioned before, we gave our attention to the strictest constraint that is : angle between line PR and y axis. We realized that once the hopper has left the ground, it is hard to revert the increasing trend of the $\theta_1 + \frac{\theta_2}{2}$. We chose the initial condition to have as lower thetas possible and we tried to land on the ground before it passes $\theta_1 + \frac{\theta_2}{2} > 150°$.



Also, another import factor is that we don't want too large angular momentum of the hopper upon contact separation. Finally, we realized that by rewriting the state equation to $q = [x, y, \theta_1, \theta_1 + \theta_2]$, we could obtain the following result.

$$a = \frac{lm}{3}(6ls_2\dot{\theta_1}^2 + 3gc_{12} + 3\ddot{y}c_{12} - 3\ddot{x}s_{12} + 6l\ddot{\theta_1}c_2)$$

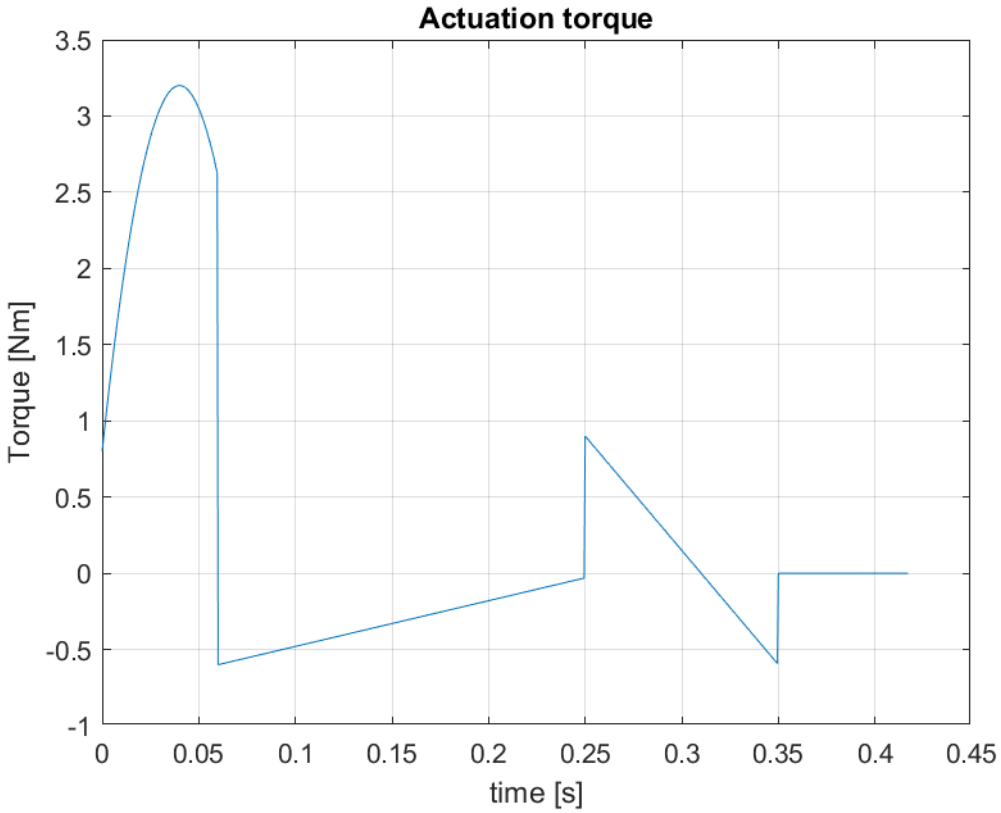
$$b = \frac{4l^2m}{3}$$

$$\tau = b * \ddot{\theta}_{12,desired} + a$$

Since we got the relation between $\ddot{\theta_{12}}$ and τ, we could run a close loop control to obtain a better result. However, this wasn't in the scope of the course. Below is our obtained result.

$$d = 0.1336[m]$$

```
%final result
t0 = 0
t1 = 0.06
t2 = 0.25
t3 = 0.35
f1 = 3.2-1500*(t-0.04)^2
f2 = -0.6+3*t
f3 = 0.9-15*t
f4 = 0
```



6. Write a short paragraph of summary, discussion and conclusion (4-6 sentences).

In this project, we utilized all the tools we have learned from the course and implemented to practical example. From the falling cat to rocking horse, we learned the principles of Lagrange equation and other impact theory so that we can apply them to our hybrid dynamical system. We also saw that the constraint of the motion can alter the path of optimizing the planning.
Chatterjee's impact law was helpful for solve any kinds of impact that we encountered. Although the final distance we obtained is not optimal, it certainly spiced our inspiration and let us search for other approach to resolve the challenge. If given enough time and material, we could finish our feedback control on the hopper and then we can fit the control signal to an analytical function. We estimate from the conservation of energy, that theoretically we can reach at least five times further distance than we have achieved. Finally, it was intriguing to construct a MATLAB program that can handle a dynamical system and we would like to explore into this area.