Spring-Loaded Inverted Pendulum
Simple Control Strategy to Hop on Rough Terrain

Noe Gonzalez[1], Giulia Piovan[2], Katie Byl[2]

[1]Electrical Engineering Major, Santa Barbara City College
[2]Center for Controls, Dynamical Systems and Computation, University of California, Santa Barbara

UCSB Robotics
robotics.ece.ucsb.edu

Project Background

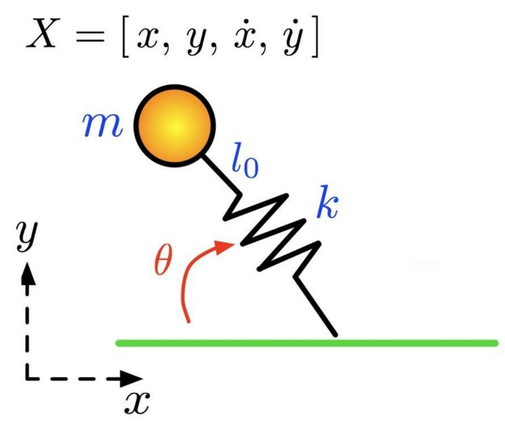
The main focus of legged locomotion is to develop and control robots that can undergo different types of natural environments. In particular, we are interested on a variety of uneven terrain (e.g. gravel, rock field, stairs, etc.), for which wheeled robots are not suitable. This technology has several applications, which include search and rescue, exploration in remote and/or hazardous area, transportation of supplies.

We decided to focus our attention on the Spring-Loaded Inverted Pendulum (SLIP), because it is the simplest model to effectively describe bouncing gaits (such as running and hopping) for bipeds and legged animals. For this reason, it is has often been used as a model for legged robot design.

Goal

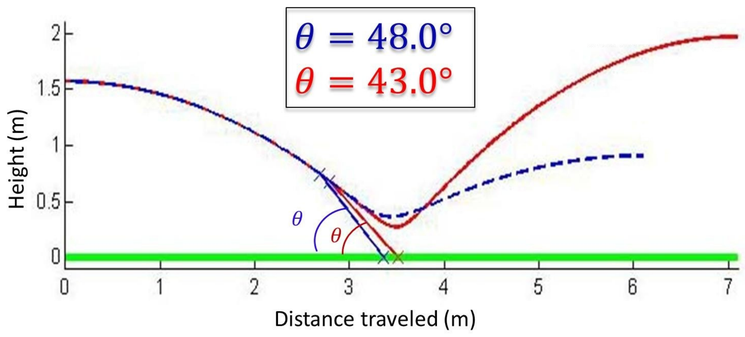
Our goal is to implement a strategy for leg positioning, to ensure that the model can perform successful jumps on rough terrain, even in the presence of faulty measurements of the terrain height.

SLIP Model



The SLIP model consists of a point mass attached to a massless spring leg. The trajectory is divided into two phases: the flight phase, where the body follows a ballistic trajectory, and the stance phase where the mass dynamics are affected by the compression and decompression of the spring. The **apex state** is defined to be the highest point of the flight phase: it is characterized by a zero vertical velocity ($\dot{y} = 0$). We define *jump* the transition between one apex state to the next.

Controlling the touch down angle

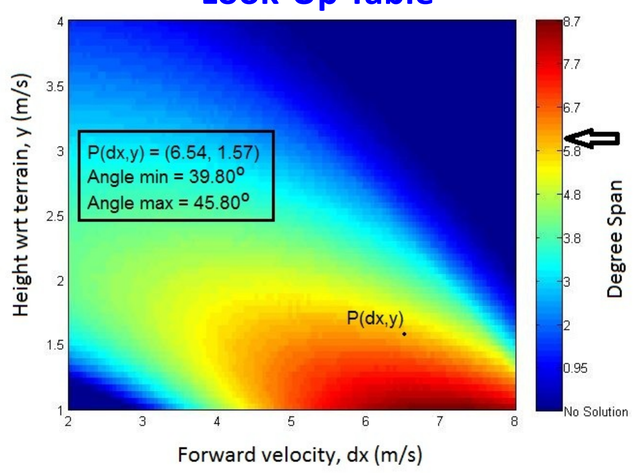


The only control action we have access to consists of the angle at which the leg touches the ground.

Given an initial apex state, there can be multiple touch-down angles that allow the completion of a successful jump. Therefore, for each initial apex state, we compute the range of all feasible touch-down angles.

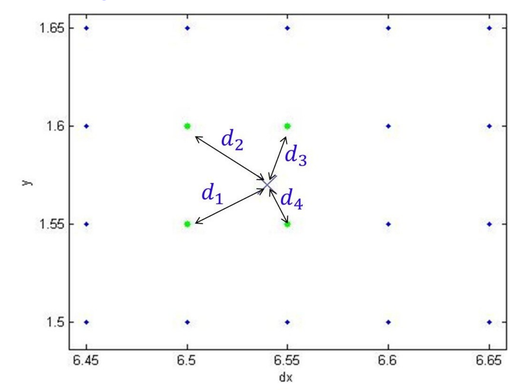
Successful jump: at the next apex state, the mass height must be bigger than the leg length, and the forward velocity must be positive (to avoid a backwards jump).

Look-Up Table



Varying the apex height and forward velocity (taken from a range of reasonable values with respect to the model specifics), we computed a look up table that shows the touch-down angle degree span for a set of initial apex states (height and forward velocity). The red zone is where we have the widest choice of touch-down angles, whereas the blue zone corresponds to all the initial apex states from which no angle allows the completion of a successful jump.

Finding angle range for point outside our mesh

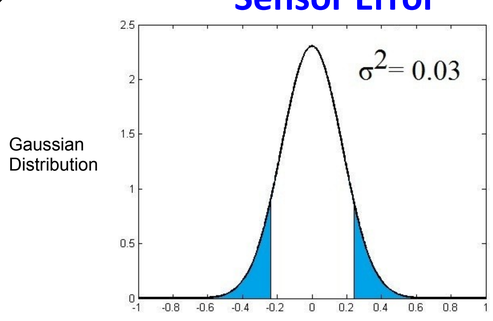


P(dx, y): (6.54, 1.57) → $W_{avg} = \sum_{i}^{4} W_i d_i$ → Angle Range: 39.80° - 45.80°

We have created a mesh of values that varies at 0.05 intervals, and for each point of the mesh we have computed the range of successful touch-down angles. If the current apex state that is not a point of the pre-computed mesh, we find its angle range as follows:

- We locate the 4 mesh points that are the closest to the current apex state.
- We compute a weighted average of the angle range associated to each of the 4 mesh points. The weights are inversely proportional to the distance from the apex point to the mesh points: the highest weight will be associated to the closest point, etc.
- Once we have the range of angles we choose the middle point in order to stay in a good zone.

Sensor Error

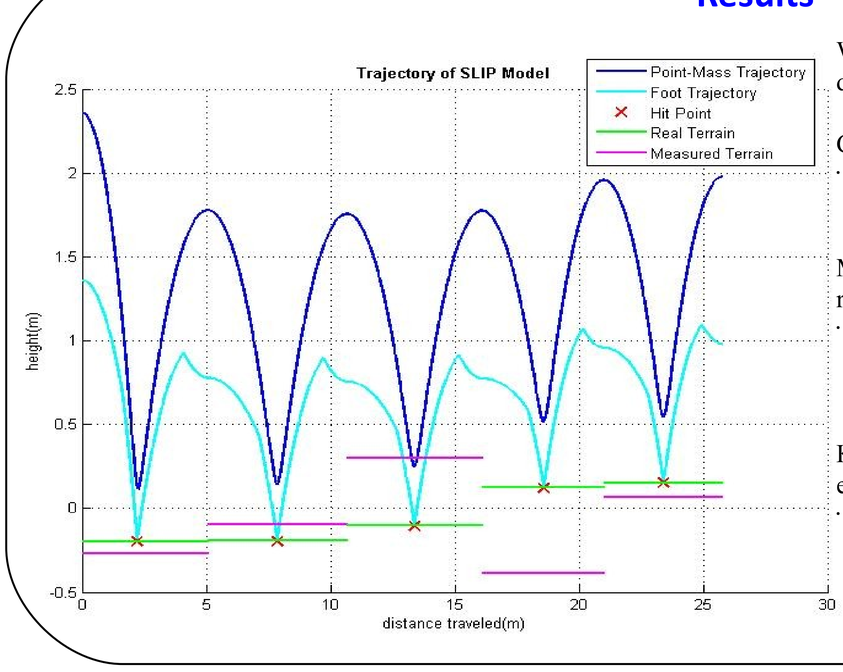


The robot will also carry a sensor that detects the terrain height. We model the noise associated to the sensor as a Gaussian distribution, with zero mean and variance σ^2.

In the picture above the variance is $\sigma^2 = 0.03$, which means that the measurement noise is bigger than 25% of the leg length with probability ~0.15 (see area in figure).

We adjust the angle range of the current apex state by taking into consideration the faulty measurement, i.e. we compute the probability of the real terrain to higher or lower than the measurement. We then calculate the respective new distances of the apex state from the terrain, and the respective angle ranges. Then, we compute a weighted average of such ranges, where the weight relative to each point will be given by the associated error probability.

Results



We were able to build a control strategy that enabled us to do the following:

Choose an initial apex state to start our simulation from.
- By defining where we start we can find the degree span that will allow a successful jump.

Maximize the probability of performing successful jumps remaining in a desired region.
- By choosing a specific angle from our degree span we can have favorable conditions that will allow us to keep jumping successfully.

Keep jumping in the presence of rough terrain and noise error from our sensor.
- We tested several σ^2 = .01, .02, .03, .04, etc... on a moderately rough terrain and found that our strategy works well for values of σ^2 up to 0.03.

Acknowledgments:

INSET: Jens Kuhn, Nick Arnold, Arica Lubin,
Mentor & Faculty Advisor: Giulia Piovan, Katie Byl