Numerical Optimization Details

Samuel Pfrommer

June 20, 2017

Abstract

This is an overview of the technical details describing how I optimized SLIP trajectories for my paper "Key Control Strategies Emerge in Spring Loaded Inverted Pendulum Traversal of Slippery Terrain."

Parameter	Value
Max Length	1m
Min Length	0.5m
Hip Mass	1kg
Toe Mass	0.1kg
Spring coefficient	20N/m
Damping coefficient	0.5
Max hip torque	± 1
Max $\ddot{r_a}$	± 1
Gravity	$1m/s^2$
Slip friction coefficient	0.05

Table 1: A table describing various parameters for the SLIP hopper simulations.

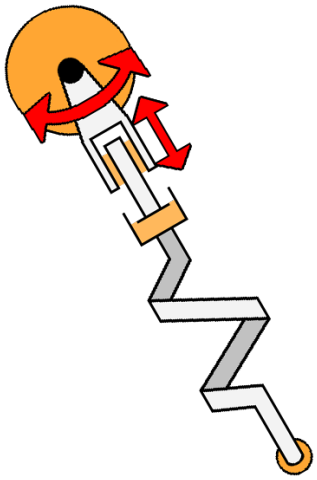

Figure 1: The Spring Loaded Inverted Pendulum (SLIP) model used in this experiment, with a spring and damper. The two red arrows represent actuation—one the hip torque, and the other the second derivative of the neutral spring length.

1 SLIP Model

The SLIP used in this experiment featured a damper, linear actuator (controls second derivative of neutral leg length, or $\ddot{r_a}$), and hip torque actuator (Figure 1). The parameters for the Spring Loaded Inverted Pendulum are listed in Table 1.

Parameters for the SLIP hopper are roughly proportional to human locomotion parameters. The slip phase friction coefficient was chosen manually to roughly approximate contact between an ice patch and a rubbery foot.

2 SLIP Equations of Motions

We will now review the equations of motions for the SLIP. Let (x,y) be the hip position, x_{toe} be the toe horizontal position, k be the spring coefficient, c be the damping coefficient, r be the SLIP length, r_a be the actuated SLIP length (or neutral spring length), τ be the hip torque, m_{hip} and m_{toe} be the hip and toe masses, respectively, and g be the gravitational constant. r can then be written as:

$$r = \sqrt{(x - x_{toe})^2 + y^2} \tag{1}$$

Differentiating by the chain rule, we get the following equation for the time derivative of r:

$$\dot{r} = \frac{(x - x_{toe}) \cdot (\dot{x} - \dot{x_{toe}}) + y\dot{y}}{r} \tag{2}$$

The force exerted by the spring-damper combination can then be expressed as follows:

$$F_s = k(r_a - r) + c(\dot{r_a} - \dot{r}) \tag{3}$$

While the force exerted by the hip torque on the ground is simply:

$$F_t = \frac{\tau}{r} \tag{4}$$

Realizing that the cosine of the angle between the SLIP and the ground is equal to $\frac{x - x_{toe}}{r}$ and the sine is equal to $\frac{y}{r}$, it is possible to express the acceleration of the hip:

$$\ddot{x} = \frac{1}{m_{hip}} \cdot (F_s \frac{x - x_{toe}}{r} + F_t \frac{y}{r}) \tag{5}$$

$$\ddot{y} = \frac{1}{m_{hip}} \cdot (F_s \frac{y}{r} - F_t \frac{x - x_{toe}}{r} - m_{hip} \cdot g) \tag{6}$$

However, it is also necessary to know the acceleration of the toe on the ice. We first determine the ground reaction force as follows:

$$GRF = F_s \frac{y}{r} - F_t \frac{x - x_{toe}}{r} + m_{toe} g \tag{7}$$

The force of friction on the toe is then given as:

$$F_f = -\mu \cdot \tanh(50\dot{x_{toe}}) \cdot GRF \tag{8}$$

Where the $\tanh(50\dot{x_{toe}})$ serves as a smooth approximation of a sign function, whose discontinuities create convergence difficulties for the optimizer.

Finally, the acceleration of the toe can be expressed as follows:

$$\ddot{x_{toe}} = \begin{cases} 0 & \text{if stick phase} \\ \frac{1}{m_{toe}} \cdot (-F_s \frac{x - x_{toe}}{r} - F_t \frac{y}{r} + F_f) & \text{if slip phase} \end{cases} \tag{9}$$

This concludes the SLIP dynamics during stance phase. Flight phase dynamics initiate when the ground reaction force intersects zero and are governed by the canonical equations for ballistic trajectories.

3 Experimentation

217 experiments were performed. In each case, a random ice patch was generated with a width of between 0.8 and 1.8 meters. The SLIP x position was initialized to a random location over the ice patch, allowing for at least a 0.3 meter margin from each side. Since the robot was assumed to be running in the positive x direction upon unexpectedly encountering the ice patch, the initial hip x velocity was randomly distributed between 0m/s and 0.5m/s, and the initial hip y velocity was randomly distributed between 0m/s and -0.4m/s. The toe started off contacting the ice a random distance within 0.2 meters in front of the hip. The initial y value was inferred from the toe x and hip x using the assumption that the leg is fully extended at the moment of touchdown. Additionally, at touchdown $r_a = r$ and $\dot{r_a} = 0$.

Two candidate step sequences were then considered: immediately stepping from the slippery surface to the ground on the right of the patch (two stance phases), or first stepping to the ground on the left and using the energy storage capabilities of the SLIP spring to launch over to the other side of the patch. Both cases were presented to the numerical trajectory optimizer (described in the next section), which found minimum-work trajectories for both step sequences. The optimal step sequence from the optimizer was then compared with the sliding mass model predictions.

Parameter	Value
Nodes per phase	10
Min stance time	0.1
Max stance time	2
Min flight time	0
Max flight time	5
Max GRF	10N

Table 2: A table of parameters for the numerical trajectory optimizer.

4 Optimization Problem

To calculate the minimum work required for each step sequence, the trajectory optimization problem was transcribed into a function optimization problem using direct collocation. Each stance phase was discretized into ten evenly-spaced nodes, with an additional parameter for the time duration of each stance phase. Dynamics between nodes were enforced using trapezoidal quadrature. To link the end of one stance phase with the beginning of the next, a flight time parameter was included. The end hip position of the ballistic trajectory was then constrained to equal the hip position at the start of the next phase. The two candidate step sequences were formulated as separate optimization problems. After transcription, the programming problem was solved using MATLAB's *fmincon* function and the Sequential Quadratic Programming algorithm.

To facilitate more reliable convergence, the optimization was performed in three steps. First, the optimizer was given a constant objective function—in other words, it was instructed to find any trajectory that satisfied the constraints. The output trajectory was then fed into an impulse-squared objective function, which is relatively fast and reliable to optimize. Finally, the output of the impulse objective was fed into a work objective function. This method converged reliably from random initial guesses when solutions existed, usually in under two minutes.

The following parameters were used for the optimizer:

4.1 Objective Function

The impulse cost function is as follows:

$$C_{imp} = \sum_{n=1}^{N_{phase}} \sum_{i=1}^{N_{grid}-1} [0.5 \cdot \Delta t_n \cdot (\ddot{r}_{a_{n,i}}^{\,2} + \ddot{r}_{a_{n,i+1}}^{\,2} + \tau_{n,i}^2 + \tau_{n,i+1}^2)] \tag{10}$$

Where N_{phase} is the number of stance phases, N_{grid} is the number of discretization nodes per stance phase, and Δt_n is equal to the $\mathrm{n_{th}}$ stance phase time T_n divided by N_{grid}.

The work cost function is written as:

$$\sum_{n=1}^{N_{phase}} \left[\int_0^{T_n} |F_s \cdot \dot{r}_a| \, dt + \int_{\theta_{td}}^{\theta_{to}} |\tau d\theta| \right] \tag{11}$$

Since the absolute value function is inherently nonsmooth—creating optimizer convergence problems—a smooth approximation was employed for both cost functions.

$$|x| \approx \sqrt{x^2 + \epsilon^2} - \epsilon \tag{12}$$

Where ϵ is about 10^{-3}.