# Pouncing Pogo

[redacted]

MENG 443 Project Proposal

October 2022

## 1 Introduction

We propose to simulate the hopping behavior of a bouncing stick robot, with relevant parameters being its spring constant, incident angle from the ground, mass and length. Based on the Spring Loaded Inverted Pendulum (SLIP) problem in robot locomotion, this project hopes to further implement a genetic algorithm optimizing the parameters for fastest travel behavior.

While a relatively simple model of locomotion, this "pouncing" has real-world analogs in legged animal motion and even the airbag landing systems used by early Mars rovers [2, 4]. The spring models the role of muscles in animal legs that absorb and release kinetic energy upon impact with the ground.

## 2 Theory

### 2.1 Pogo architecture

Pogo is a one degree of freedom robot that moves by compressing and releasing a spring (Fig. 1). The main control input is the incident angle $\theta_I$; the release angle will be determined by physics in our model. The spring is characterized by Hooke's Law with spring constant $k$. Since Pogo must swing forward while

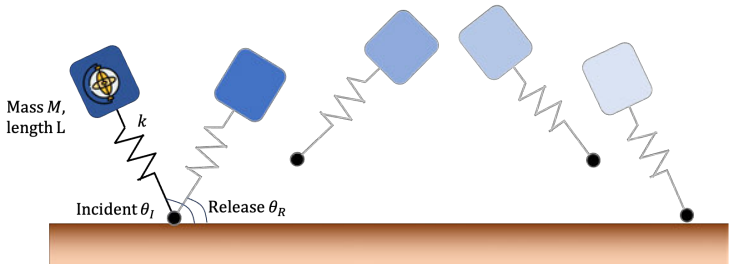


Figure 1: Illustration of proposed Pogo robot; $M$ is the tip (flywheel) mass, $k$ the spring constant, $\theta_I$ is the incident angle.

airborne, we can imagine a flywheel mechanism housed in the top to manage the angular momentum. It might make sense to relate Pogo's mass, length and spring constants per biological examples (bigger animals have bigger muscles) [1]. Pogo's motion can be separated into a sequence of "flight" and "stance" phases respectively.

During its flight phase, Pogo follows a ballistic trajectory under the influence of gravity alone (with $g$ as gravitational acceleration). Following the notation in [2], the equations of motion for the mass are:

$$\ddot{x}(t) = 0 \tag{1}$$

$$\ddot{y}(t) = -g \tag{2}$$

In the literature, the highest point reached by the top mass during flight is called the apex and defined when the vertical velocity is zero [2, 3]. Upon impact with the ground, the spring compresses until it reaches the minimum compression point, then starts expanding. Take-off occurs when the spring reaches its equilibrium point on the expansion cycle.

Relative to the origin (pivot) of the ground contact point, Pogo's tip mass is like an inverted pendulum with coordinates given by:

$$x(t) = l(t)\cos\theta(t) \tag{3}$$

$$y(t) = l(t)\sin\theta(t) \tag{4}$$

Using the kinetic, gravitational potential and spring elastic potential energies we can write the Lagrangian as:

$$L = \frac{M}{2}(l^2\dot{\theta}^2 + \dot{l}^2) - Mgl\sin\theta - \frac{k}{2}(l_k - l_{k,0})^2 \tag{5}$$

Applying the Euler-Lagrange equation to coordinates $l, \theta$ (instead of x, y) to obtain the equations of motion:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \frac{\partial L}{\partial q} \tag{6}$$

$$\Rightarrow \ddot{l} = -\frac{k}{M}(l_k - l_{k,0}) - g\sin(\theta) + l\dot{\theta}^2 \tag{7}$$

$$\Rightarrow \ddot{\theta} = -2\frac{\dot{l}}{l}\dot{\theta} - \frac{g}{l}\cos\theta \tag{8}$$

Eq. 7 and 8 will be numerically simulated to solve for the stance phase. The acceleration of the total leg length equals that of the spring i.e. $\ddot{l} = \ddot{l_k}$. We will set the spring length parameter $l_k \leq l$ at the start of the simulation. Due to the need to check for ground impact, the translational coordinates and derivatives of the center of mass, together with rotation information $\theta(t)$ and $\dot{\theta}(t)$, need to be computed and stored per timestep as Pogo's state vector.

## 2.2 Bounce termination rules

In the design shown in Fig. 1, Pogo will be launched with a fixed initial velocity and allowed to bounce repeatedly. Without a driving energy source, energy is fully conserved in this current model (if time permits we can add in energy dissipation terms). The maximum distance travelled will be recorded as the fitness function. We set two termination conditions (how Pogo falls over):

- **Fall back:** During ground contact, if Pogo's angular velocity $\dot{\theta}(t) \geq 0$ while $\theta(t) > 90°$, it would have lost forward momentum while pivoting. A continuous torque from the top mass will then cause it to fall backwards.
- **Trip:** When airborne and moving upwards, if Pogo's foot height falls to zero (swinging like a pendulum), it would trip over having not jumped high enough.

$$y_{foot} = y_M(t) - L * \sin(\theta(t)) \leq 0 \tag{9}$$

In the airborne phase, we only consider translational parabolic motion of the top mass (also the center of mass, assuming negligible spring and foot mass), subject to the trip-over condition. This basic model ignores air resistance and supposes the flywheel enables Pogo to reach whatever desired $\theta_I$ incident angle (e.g. via a kinematic controller). This makes sense in the context of a land-based animal leg (vs a swimming or flying mechanism, for instance).

As Pogo hits the ground, we transition to the contact phase. The spring compresses to arrest the vertical velocity component via conservation of energy, while the horizontal inertia forces the top mass forward. We assume sufficient friction at the foot (point of contact pivot) to keep Pogo from sliding.

## 2.3 Hopping simulation framework

Pogo will be instantiated as an object in Python, with various methods such as rotate-in-air, contact checking and ground pivoting. Equations of motion 7 and 8 will be computed using the Runge-Kutta 4 numerical integrator.

An outline of the pseudocode for the classic SLIP model is given below:

```
Initialize Pogo object, start state
while time < T:
    while Pogo is airborne:
        Forward propagate (Runge-Kutta 4) ballistic path per Eq. 1 and 2
        Check for contact (foot height)
        Increment distance travelled
        Increment time
    while Pogo is in contact:
        Forward propagate stance motion per Eq. 7 and 8
        Compress spring and check extension
        Increment time
```

```
        If spring is back to original length:
            Pogo goes airborne
    return distance traveled
```

## 2.4 Expected motion

Intuitively, we want the release angle $\theta_R$ to be small so as to impart sufficient forward horizontal velocity, but also not too small lest Pogo's foot trip. Also, since Pogo will fall over if the mass does not travel past the point of contact in time during a bounce, we expect sufficiently large incident angle $\theta_I$. We hope to reach these parameters through a genetic algorithm evolution.

A genetic algorithm is a biologically-inspired method to generate high-quality solutions to some fitness function, by mutating and evolving the properties ("genotype") of candidate solutions [5, 3]. In this project, the fitness function can be the maximum distance covered in a fixed interval of time without falling over, and genotypes will be the spring constant, mass, and incident angle (in binary form). From the initial population of sample Pogo parameters, the more successful versions (faster hopping) will undergo crossover and individual genes mutated. In each round, the least successful set will be replaced and this evolution continues until the overall results (hopping speed) converge.

The pseudocode for the genetic evolution portion is briefly outlined below:

```
Generate initial population with various k, mass and length
for each Pogo:
    Compute fitness score via motion simulation
while fitness scores not converged:
    Select high score individuals for mating
    Crossover genes
    Mutate
    Compute entire batch's fitness scores again
    Check for convergence
return fastest Pogo variant
```

It is possible that some evolved Pogo variants can travel fast for certain incident angles but are unstable at others (e.g. due to insufficient clearance from the ground). In that case, we can test with multiple input angles and take the average score from all trials to fine-tune the fitness function.

The key research questions are:

- When is it ideal for the robot to land (angle of attack $\theta_I$)?
- What is the ideal spring constant?
- Does a consistent set of evolved robots emerge?

## 3 Potential pitfalls and fall back plans

In the event the genetic algorithm does not work, we can revert to a simple point mass physics derivation to find the optimum bounce angle subject to the termination rules. The bounce mechanism can also be taken as a direct maximum coefficient-of-restitution calculation in the next timestep if the contact dynamics are difficult to implement. On the other hand, if there is sufficient time and the genetic algorithm works well, a more complex SLIP model can be considered with spring damping or resistive torque.

## References


[1] Ha D. (2019) Reinforcement learning for improving agent design. Artif Life; 25 (4): 352–365. doi: https://doi.org/10.1162/artl_a_00301

[2] Piovan G., Byl K. (2016) Approximation and control of the SLIP model dynamics via partial feedback linearization and two-element leg actuation strategy

[3] Walker K., Hauser H. (2018) Evolving optimal learning strategies for robust locomotion in the spring-loaded inverted pendulum model. IROS.

[4] NASA, Mars Exploration Rovers. https://mars.nasa.gov/mer/mission/overview/ Accessed 18 Oct. 2022.

[5] Geeksforgeeks. Genetic Algorithms. https://www.geeksforgeeks.org/genetic-algorithms/ Accessed 18 Oct. 2022.