Using Computer Simulations to Test the Viability of a Spring Powered Launcher

Simon Abrelat

May 2019

002129-0004

IB Physics HL IA

Words:

Contents

1 Introduction

In the design process, there needs to be a way to vet interesting and alternative ideas. On my robotics team, one way to do this with computer simulations. To scale a 13 inch (33 cm) vertical step the idea of a spring powered launch was brought up, so its efficacy had to be determined. The idea was to angle the robot and have a pogo-inspired launcher spring out from the back of the robot. The goal of this simulation is to approximately test if this design possible, and, if its theoretically possible, are the materials reasonable.

1.1 Design Constraints

This robot is designed for the FIRST Robotics Competition (FRC), so there are a lot of rules that inform what is possible. There must be a padded bumper that is no more than 3 inch (7.6 cm) off of the ground. This greatly limits the angle we are able to tilt the robot, for this simulation a max angle of 50°. There is also a weight limit of 150 lbs (69 kg). This weight will be used for the simulation to prove that the design can work in a variety of situations and is more consistent. There are space constraints, but they are not being factored in for this simulation since those are more design specific implementation details over physic constraints.

2 Simulation

This simulation has 2 sections: acceleration and flight. The acceleration uses the spring to force the robot up. The flight is the projectile motion of the robot in the air. Given the physical constraints, there are simulation constants like a weight of 69 kg and angle of 50°.

2.1 Spring Acceleration

Acceleration has to be calculated from the force of the spring. The spring force is approximated by Hooke's Law (1). Hooke's Law states the force is proportional to the distance of compression of the spring. The slope of this proportion would then be the spring constant which is measured in Newtons per meter. So for each meter compressed the spring exerts that many Newtons of force.

$$F_s = -kx \tag{1}$$

F_s: Force of the spring, N

k: Spring constant, $\frac{N}{m}$

x: The displacement of the spring, m

The force is then used to describe the acceleration using Newton's second law (2). The vertical acceleration from the spring is then reduced by the force due to gravity giving the net acceleration on the object. This net force is described by equation 3.

$$\vec{F} = m\vec{a} \tag{2}$$

$\vec{F}$: Force vector, N

m: Mass, kg

$\vec{a}$: Acceleration vector, $\frac{m}{s^2}$

$$ma = -kx - mg \tag{3}$$

a: Net acceleration, $\frac{m}{s^2}$

k: Spring constant, $\frac{N}{m}$

m: Mass, kg

g: Acceleration due to gravity, $9.81\frac{m}{s^2}$

The acceleration of the object decreases the distance which then decreases the force and the net force is decreased by gravity (4), full derivation in Appendix 3. This generates a cyclical decay in force. For this simulation, this second degree differential equation would be approximated using Euler's method.

$$\ddot{x} = \frac{-kx}{m} - g \tag{4}$$

Equation 4 is calculated by stepping through and solving the equation small steps at a time. For this, the equation gives you acceleration given a position, the acceleration determines the velocity which moves the position as described in Algorithm 1.

Algorithm 1 Acceleration Solver

```
Require: $v, a = 0$
Require: $g = 9.81$
Require: $t = 0.001$
Require: $x =$ starting displacement
Require: $k =$ spring constant
  while $x \geq 0$ do
    $a \leftarrow \frac{-kx}{m} - g$
    $v \leftarrow v + a * t$
    $x \leftarrow x + v * t$
  end while
```

3 Appendix

List of Figures

Equation Derivation, Appendix 3

$$
\begin{gathered}
F = ma \\
F = -kx \\
\sum F = -kx - mg \\
ma = -kx - mg \\
m\ddot{x} = -kx - mg \\
m(\ddot{x} + g) = -kx \\
\ddot{x} + g = \frac{-kx}{m} \\
\ddot{x} = \frac{-kx}{m} - g
\end{gathered}
$$

Code

Listing 1: Spring Acceleration

```python
import matplotlib.pyplot as plt
from typing import List
import numpy as np
from math import sin, cos, hypot
from math import radians as rad

```

```python
# Simulation Constants
dt: float = .001  # time between physics iterations

# Physics Constants
m: float = 69  # mass: kg
k: float = 7000  # spring constant: N/m
g: float = 9.81  # gravitational accel: m/s^2
theta: float = rad(45)  # the angle the robot luaches at
e: float = .5  # effiency

# Spring data recording
x: float = -0.3  # inital compression
v: float = 0  # initial velocity
a: float = 0  # initial accel
ta: List[float] = []  # time
xa: List[float] = []  # position
va: List[float] = []  # velocity
aa: List[float] = []  # acceleration


def hookesAcc(x: float) -> float:
    return (((-k * x) / m) - g) * e
    #v = ((-k * x) / m) - g
    #return hypot(v*sin(theta) - g, v*cos(theta))


i = 0
while(x < 0):
    i += 1
    a = hookesAcc(x)
    v += a * dt
    x += v * dt
    if(x > 0): break

    t = i * dt

    ta.append(t)
    xa.append(x)
    va.append(v)
    aa.append(a)
    print(x, v, a, t)

print("max vel: ", v)
print("height: ", ((v**2) / (2*g)))
plt.plot(ta, xa)
plt.plot(ta, va)
plt.plot(ta, aa)
plt.show()
```