Using Vpython to Analyze a Dynamic Equlibrium System

Katherine Kireeva

15-424

1 Introduction

The goal of this project is to use python's graphics library to animate a pulley and spring system for different initial conditions and constants. This process can be generalized for an arbitrary set of springs and masses with the goal of determining if a given system would stay in dynamic equilibrium, where all the masses remain within upper and lower bounds. This paper specifically looks at the case of a three masses, tow of which are connected by a weightless ideal spring.

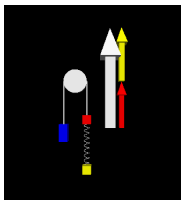


Figure 1: The arrows represent the forces acting on the red mass

2 Motivation

Spring and pulley problems are common in high school physics classes, and can be solved with good understanding of Hooke's Spring Law and Newton's Force Law. Even a combination of springs and pulleys is can be analyzed if the masses start out stationary (or moving at the same velocity) and all the forces balance perfectly, e.i. the system is in static equilibrium. However the relevant problems become a lot more complicated when the masses are either moving at different velocities, or start in positions where the spring, tension and gravitational forces acting on them don't cancel out. Animating a system in dynamic equilibrium is useful for gaining an intuition for how the differential equations governing it behave.

3 Review of the Physics Involved

For purposes of calculations, call the blue block weighing *2m*, A . Similarly, call the red and yellow blocks which both weigh m, B and C respectively. The relevant forces acting on the three masses are tension, gravity, and the spring force.

1. In a weak gravity field like earth's it is a safe approximation to say the the field is flat, e.i, not height dependent for changes in height that are small relative to earth's radius.

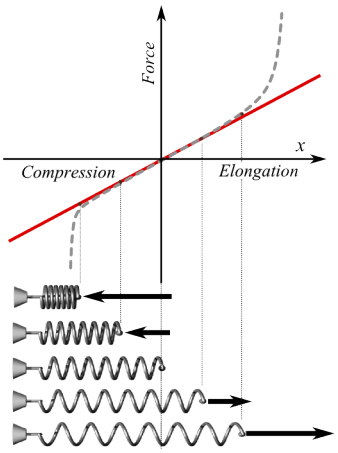
$$F = G\frac{(m * M_e)}{R_e^2} \approx mg$$

. To simplify calculations, the mass is set to 1, $m = 1$, since the strength of the gravitational field g can be changed for the same effect on the system. For $\Delta\ h << R_e$, the change in the force due to gravity is negligible compared to the magnitude.

2. The tension is distributed along the pulley cable and therefore acts equally on blocks A and B.

3. Hooke's Law states that the the force felt by an object connected to an extended or contracted spring is proportional to the change in length, x from the spring's natural length L, where the proportionality constant k varies with the spring. It's worth mentioning that in an real spring, the force follows an s-curve as the spring is stretched or compressed beyond it's operational limits.

$$F = -k * x$$



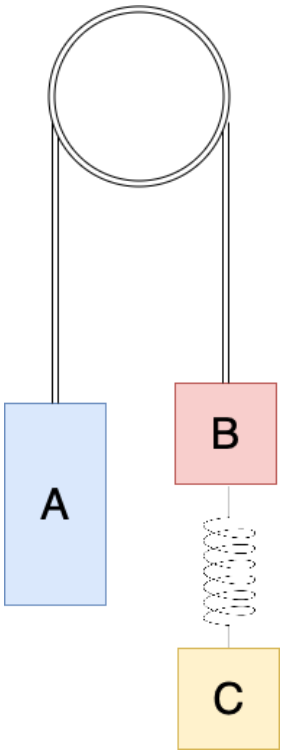


Figure 2: Spring shorter than L

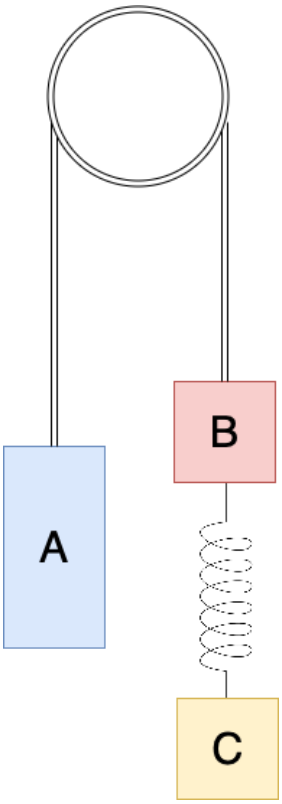


Figure 3: Spring is natural length, L

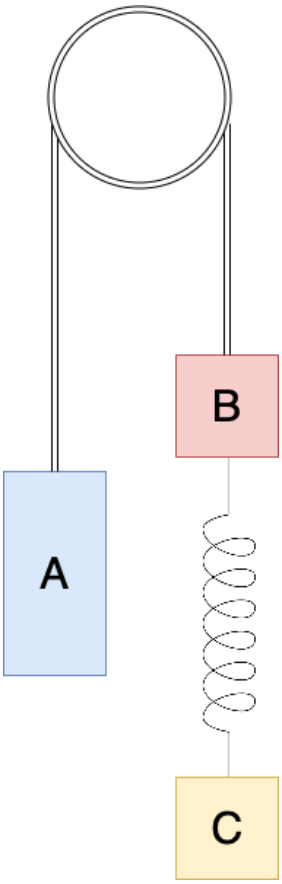


Figure 4: Spring longer than L

4 Differential Equations Describing the motion

- L - the natural length of the spring
- k - the spring constant
- g - acceleration due to gravity per unit of mass
- T - Tension
- F_S - Spring force
- x_A, x_B, x_C - positions of the blocks

and down was chosen to be the positive direction. Another assumption is that the blocks are point masses and the spring connects blocks B and C perfectly. Define the spring force as follows

$$F_S = -k * (L - (xB - xC))$$

The mass times acceleration of a body is equal to the sum of the forces acting on it. Referring to the free body diagrams below, we can write down the three equations of acceleration.

$$2 * x_A'' = T - 2g$$
$$x_B'' = T - g - F_S$$
$$x_C'' = F_S - g$$

Because blocks A and B are connected by a string hung across a pulley, we know their accelerations are equal and opposite.

$$x_A'' = -x_B''$$

We can now solve for tension that will be used

$$\frac{T}{2} - g = -T + g + F_S$$

$$\frac{3T}{2} = 2g + F_S$$

$$\frac{T}{2} = 2g + F_S$$

$$T = \frac{4g}{3} + \frac{2}{3}F_S$$

The oscillation rate of the system will be

$$w = \sqrt{\frac{k}{g}}$$

Forces.png

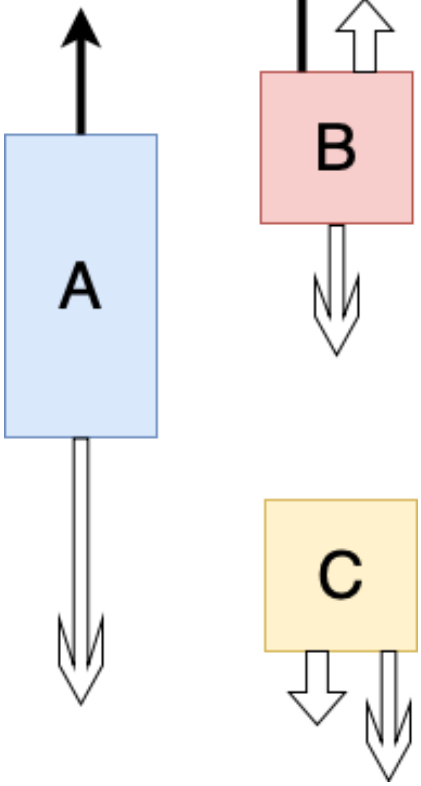


Figure 5: Spring shorter than L

Forces.png

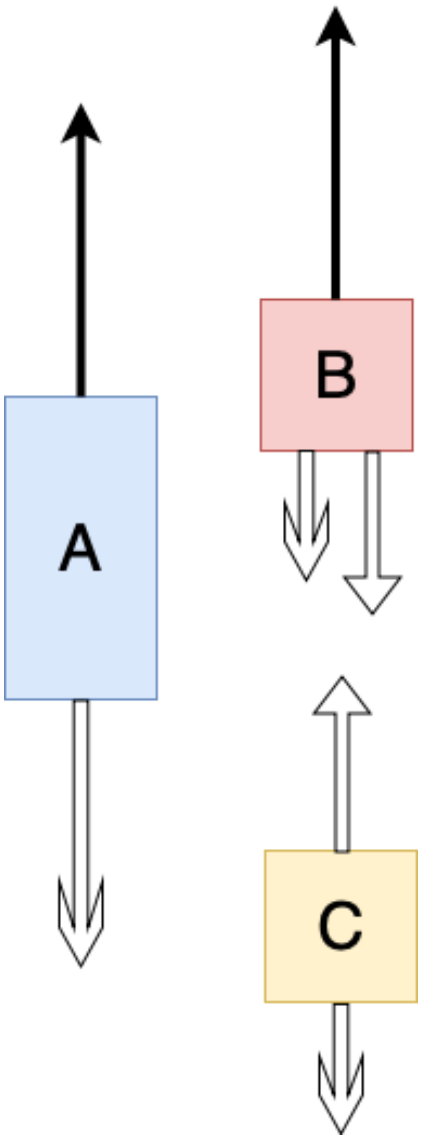


Figure 6: Spring longer than L

Figure 7: Free Body Diagrams for the blocks. Black arrow represents tension, pointed arrow represents gravity, triangular arrow represents spring force. Notice that the yellow block extends much lower down than it rises up. Also notice how the force of tension increases when the spring force is pulling the red block down.

The code used for generating the animations is given below. It uses VPython for the visual image libraries to create arbitrary shapes, and the Python Imaging Library for image capture. After generating the images in a pre-existing folder ffmpeg is used to render the mp4 file.

Vpython Download: http://vpython.org/contents/doc.html PIL Download: https://pypi.org/project/Pillow/ ffmpeg Download : https://www.ffmpeg.org

```
from visual import *
from visual.graph import *
from visual.controls import *
from PIL import ImageGrab

scene.title="Dynamic Equilibrium Simulation"
scene.range = 10
scene.autocenter = 1
scene.autoscale = 1

#image grab stuff
image = False
padding = 4
directory = "movie-images"

#dimensions
rope_length = 5.0
spring_length = 1.0
spring_start = 4.0
size = 0.75
scale = 4

#constants
g = 4.0
k = 1.0
dt = 0.05
time = 40

#pulley and strings for visual purposes
pulley = cylinder(pos=(0,0,-0.25), axis=(0,0,0.5), radius=1)
left_string = cylinder(pos=(-1,0,0), axis=(0,-rope_length,0), radius=0.05)
right_string = cylinder(pos=(1,0,), axis=(0,-rope_length,0), radius=0.05)

#spring and boxes objects
box_a = box(pos = left_string.pos + left_string.axis,
            length = size, width = size, height = size*2, color = color.blue)
box_b= box(pos = right_string.pos + right_string.axis,
            length = size, width = size, height = size, color = color.red)
```

```
spring = helix(pos = box_b.pos, axis = (0, -spring_start, 0),
               radius = 0.25, coils = 10, thickness = 0.05)
box_c = box(pos = spring.pos + spring.axis,
            length = size, width = size, height = size, color = color.yellow)

#initial conditions
t = 0.0
box_a.vel = 0.0
box_b.vel = 0.0
box_c.vel = 0.0

Fs = 0.0
T = 0.0

#force comparisons on the red box, scaled down
T_force = arrow(pos = box_b.pos + (2, 0, 0))
G_force = arrow(pos = box_b.pos + (3, 0, 0), color = color.red)
G_force.axis = (0,g/scale, 0)
S_force = arrow(pos = G_force.pos + G_force.axis, color = color.yellow)

#graph of speeds of the red and blue box
gdisplay(title='Position', xtitle='Time', ytitle='Position',
            xmax=time, ymax=0, ymin=-rope_length * 3, x=0, y=450, width=500, hei
line_a = gcurve(color=color.blue)
line_b = gcurve(color=color.red)
line_c = gcurve(color=color.yellow)

scene.autocenter = 0
while t < time:
    #tells computer how fast to animate
    rate(50)

    #calculating the forces
    dL = (spring_length - (box_b.pos - box_c.pos).y)
    Fs = -k * dL
    T = 4.0 * g / 3.0 + 2.0 * Fs / 3.0

    #drawing force arrows, scaled
    S_force.axis = (0, Fs/scale, 0)
    T_force.axis = (0,T/scale,0)

    if(image):
        im = ImageGrab.grab((40,50,500,520))
        timestamp = int(t/dt)
        im.save(directory + "%05d" % timestamp  +'.png')
```

```
#velocity
box_a.vel += dt * (T / 2.0  - g)
box_b.vel += dt * (T - Fs - g)
box_c.vel += dt *(-g + Fs)

#position
box_a.pos.y += dt * box_a.vel
box_b.pos.y += dt * box_b.vel
box_c.pos.y += dt * box_c.vel

#springs and pulley ropes
left_string.axis.y = box_a.pos.y
right_string.axis.y = box_b.pos.y
spring.pos.y = box_b.pos.y
spring.axis = box_c.pos - box_b.pos

#plots of positions
line_a.plot(pos=(t, box_a.y))
line_b.plot(pos=(t, box_b.y))
line_c.plot(pos=(t, box_c.y))

t += dt
```

In the directory with the generated images (depending on what *dt* you use this may take a lot of time and disk space) run ffmpeg with an appropriate frame rate. This command uses images named img-xxxx.png from 0000 to 9999 at a 100 frames/second to generate the file movie.mp4.

```
ffmpeg -r 100 -i img-%4d.png -vf format=yuv420p movie.mp4
```