APC523 Final Project Writeup

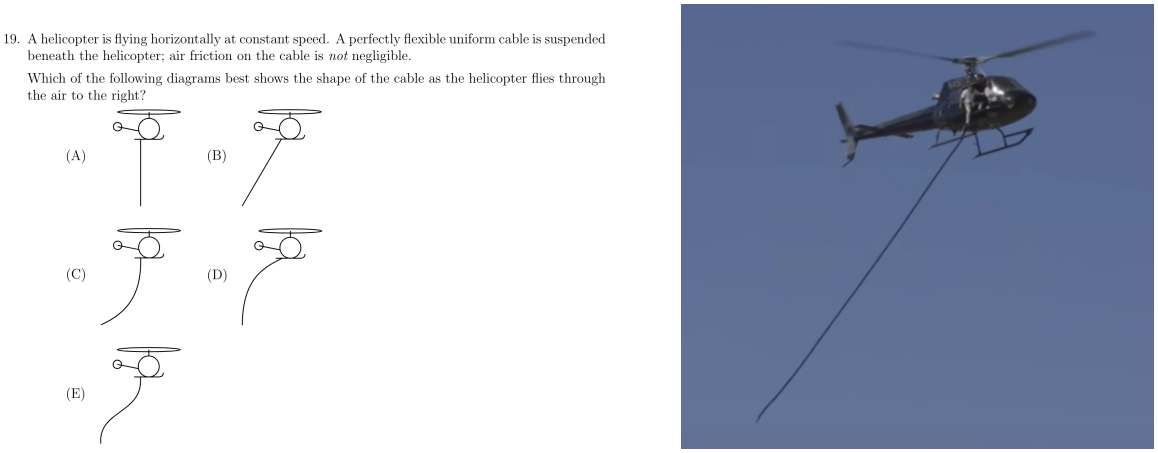
Atharva Pathak and Jello Zhou

May 7, 2024

1 Introduction

In our project, we have created a rope simulator to explore some effects we saw in YouTube videos.

In this video by Veritasium, he hangs a rope from a helicopter to answer the question in Figure 1a. He also tries adding an additional mass to the bottom of the rope and adding an additional parachute to the bottom of the rope. We have simulated all three cases.



(a) Problem from 2014 $F = ma$ competition that Veritasium resolves.

(b) Screenshot from Veritasium video showing helicopter towing a rope.

Figure 1: Setup and solution of Helicopter problem.

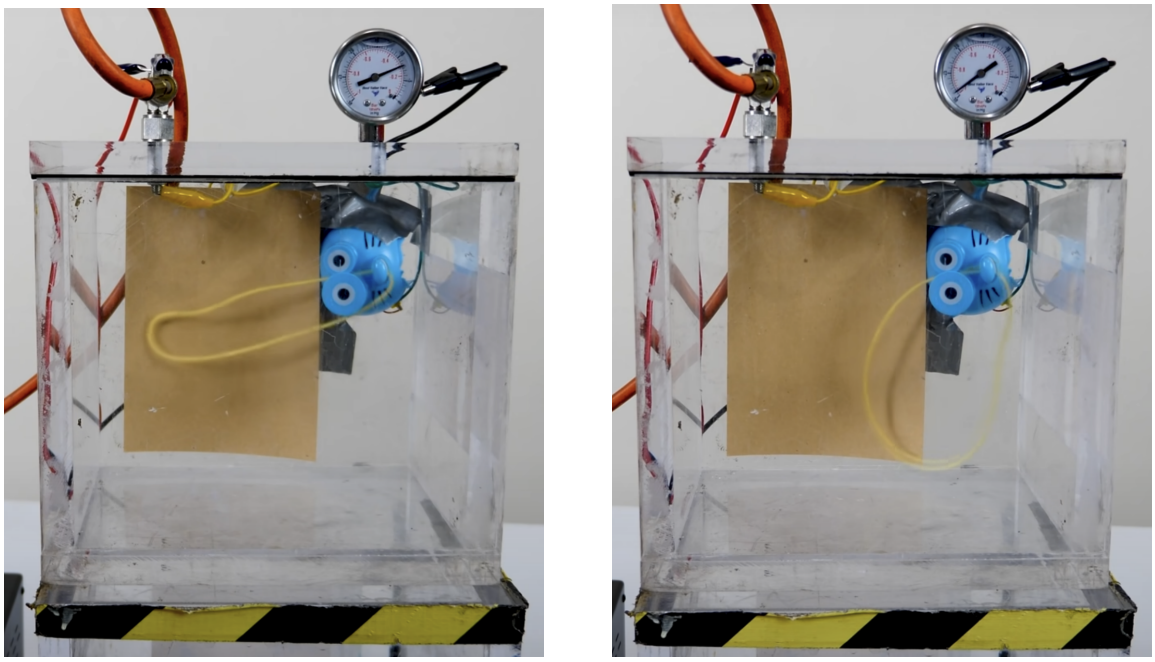
This video by the Action Lab uses a string shooter toy that drives a loop of string. He tests the toy in air and in a vacuum, and shows that the loops of string form different shapes. Screenshots from the video are in Figure 2.

2 Implementation of Simulation

We implement rope by point masses of mass m connected to each other by stiff springs with rest length ℓ and spring constant k. Lines and loops of rope are both easily made by either enforcing or not enforcing boundary conditions between ends of the rope. We implement air resistance and gravity by a force $-bv$ and $-mg$ on each particle.

We implemented a rope stiffness term by adding weaker springs between every other point mass, and setting the boundary conditions appropriately when dealing with a line rather than loop. This term proved to be helpful in reducing bunching of particles in the string shooter case, but we did not find it necessary in the helicopter case.

Our main time integrator was symplectic Euler, where we use force to update velocity and use the updated velocity to update position. Since we were discretizing the rope with a first-order approximation,



(a) String shooter in air

(b) String shooter in vacuum

Figure 2: Screenshots from Action Lab video showing string shooter in atmosphere versus vacuum. The string is thrown to the left by rotating wheels, then fed back through a loop on the right. In air, the string forms a horizontal loop, but in a vacuum, the loop is more vertical.

i.e. connections between point masses are springs that can only flex in one direction, we did not need higher-order time integrators. We also implemented the fourth order Yoshida method, but did not observe any benefits.

Following the Libretexts resource linked below, for a harmonic oscillator with angular frequency ω we have

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} 1-\omega^2\Delta t^2 & \Delta t \\ -\omega^2\Delta t & 1 \end{bmatrix} \begin{bmatrix} x_n \\ v_n \end{bmatrix},$$

and the eigenvalues of the matrix are $\lambda = 1 - \frac{\omega^2\Delta t^2}{2} \pm \frac{\omega\Delta t\sqrt{\omega^2\Delta t^2-4}}{2}$. When $|\omega\Delta t| > 2$, these eigenvalues are both real and both have magnitude not equal to one, but when $|\omega\Delta t| < 2$ the eigenvalues are complex and lie on the unit circle. So for stability, we must have $|\omega\Delta t| < 2$ for symplectic Euler.

In our system, we have characteristic oscillation at $\omega \approx \sqrt{\frac{k}{m}}$, where k is the spring constant connecting the point masses and m is the mass of a particle. So for stability we must have $\Delta t \lesssim 2\sqrt{\frac{m}{k}}$. Indeed, in our experimentation, insufficiently small values of Δt led to wild oscillation and instability. Typical values in our simulation are $k = 5000\,\mathrm{N\,m^{-1}}$ and $m = 1\,\mathrm{kg}$, so $2\sqrt{\frac{m}{k}} = 0.028\,\mathrm{s}$ is the upper limit. We used $\Delta t = 0.01\,\mathrm{s}$ or $0.001\,\mathrm{s}$, both of which easily satisfy the stability constraint.

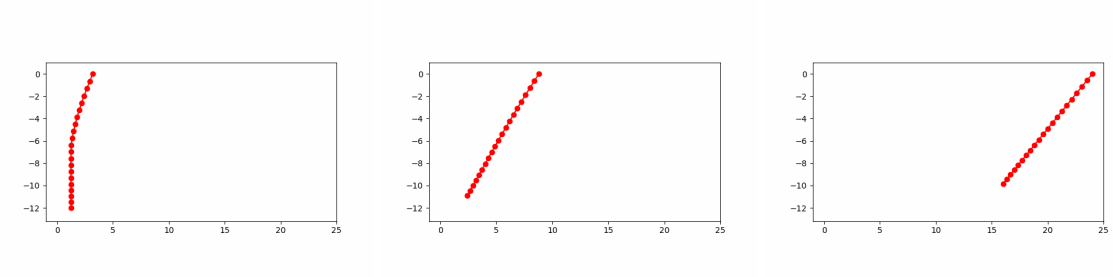
3 Helicopter Problem

To simulate a rope hanging from a helicopter, we assume an initial condition in which the string is dangling vertically with top node at the origin, and nodes separated by each spring's rest length. Further, we impose a constant initial velocity v to each node. This velocity is the velocity at which the helicopter moves. During each force update step, we set the force on the top node to zero such that it continues moving at v in order to model the force the helicopter exerts on the rope.

We simulate all three cases described in the Introduction – rope of uniform density, rope with mass attached to bottom, and rope with parachute attached to bottom. In each case, we use the parameters

$v = 8$, $g = 9.81$, $k = 5000$, $l = 0.5$, and $N = 21$, and simulate to $t_f = 3$, where v is the velocity of the top particle, g is gravitational acceleration, k is the spring constant of the spring connecting adjacent particles, l is the rest length of the springs connecting adjacent particles, and N is the number of particles.

Rope of constant density. To simulate a rope of constant density, we use $b = 5$ and $m = 5$. Results are shown in Figure 3. It is possible to see that results support analytical predictions – the steady state of the

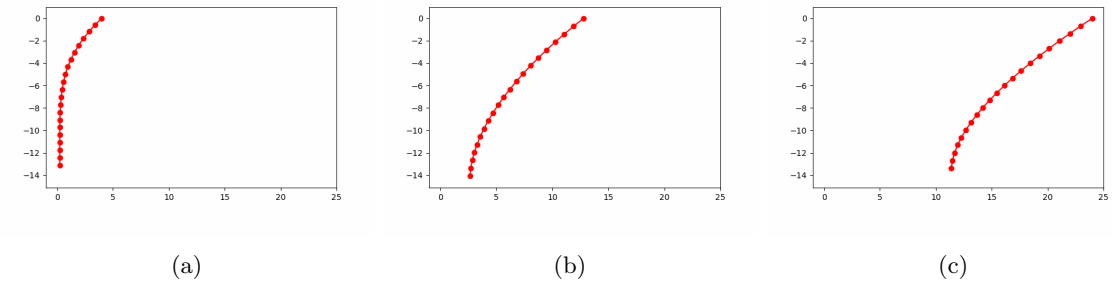


(a) (b) (c)

Figure 3: Selected frames of the helicopter problem with no additional mass or parachute.

system approaches a straight line. An intuitive argument for a straight line steady state is as follows. We consider the forces on a section of the rope with infinitesimal length $d\ell$. The angle of this rope is determined by the ratio of the gravitational and drag forces. Note that the gravitational force is proportional to mass, which is proportional to $d\ell$; further, the drag force is proportional to area, which is also proportional to $d\ell$. Hence, the ratio between gravity and drag remains constant over the rope, so the rope hangs at a constant angle.

Rope with mass attached to bottom. Here, we set the mass of every non-bottom node to be $m = 5$, and the mass of the bottom node to be $M = 100$. Further, we use the drag constant $b = 5$. A selection of frames is shown in Figure 4. In the figure, one can see that the rope relaxes to a curved shape, making the



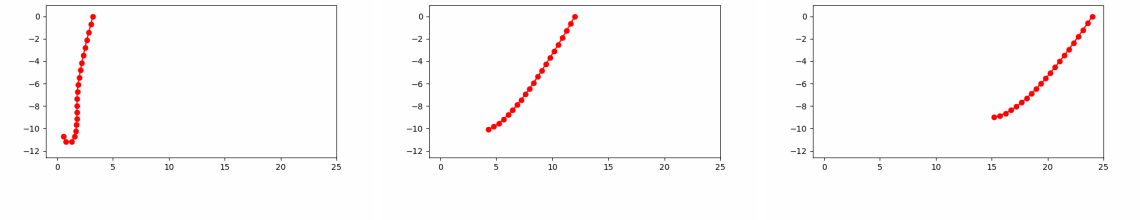
(a) (b) (c)

Figure 4: Selected frames of the helicopter problem with an additional mass at the bottom of the rope.

shape of option (D) of Figure 1a. This argument can be formulated as follows. Given any point on the rope, consider the forces acting on the part of the rope below it. Air resistance would be roughly equal to the case with no mass (i.e. it would still be proportional to the length of the rope below the point). However, there is additional gravitational force on the mass. As one moves to lower and lower points on the rope, the ratio of gravity to air resistance on the part of the row below the point would increase. Therefore, the angle of the rope would be steeper for lower points.

Rope with parachute attached. To simulate a parachute on the bottom node of a rope, we leave mass unchanged from the standard case and increase the air resistance on the bottom node. The parameters we use are $m = 5$, $b = 3$, and $B = 30$ where B is drag on the bottom node. Selected frames of the simulation are

shown in Figure 5. This figure displays a steady state similar to option (B) of Figure 1. The explanation for



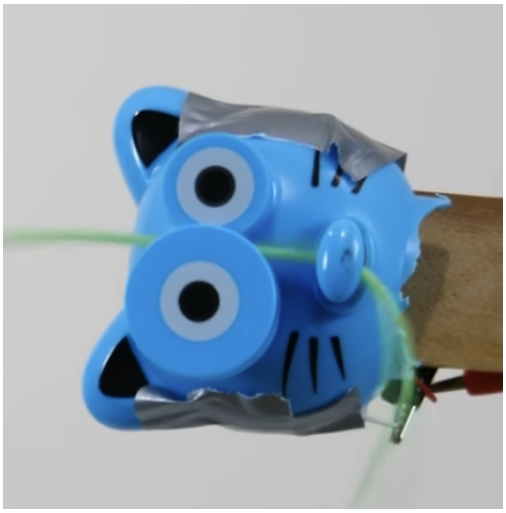
(a) (b) (c)

Figure 5: Selected frames of the helicopter problem with an additional mass at the bottom of the rope.

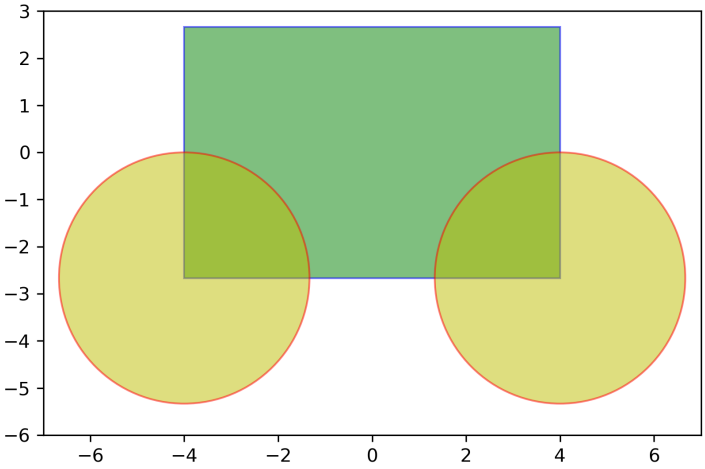
this shape is very similar to that for the additional mass case. For points lower on the rope, there would be a higher ratio of air resistance to gravity since the bottom node exerts disproportionately high air resistance. As such, the rope becomes less steep at lower points.

Our simulations confirm analytical predictions, as well as observations in Veritasium, suggesting that the algorithms are robustly implemented.

4 String Shooter Problem



(a) Closeup of string shooter, showing how string gets fed up through the nose and gets thrown by spinning eyes.



(b) How we model the stringshooter.

Figure 6: Closeup of stringshooter and simulation's model of stringshooter.

To model the string shooter, we initially looked at rigid body physics simulation to describe the string being forced around the nose, as shown in Figure 6a. However, we decided it would be unnecessary for our purposes. Instead, the string shooter in our model consists of three components, shown in Figure 6b.

For particles inside the rectangle, we provide a forward horizontal force $F_x = b_{\text{thrower}}(v_{\max} - v_x)$, where $v_{\max}$ denotes a maximum theoretical speed to get the string up to. We still keep the ambient air resistance active for particles inside the thrower, so the fastest speed attainable is $b_{\text{thrower}}(v_{\max} - v) - bv = 0 \implies v = \frac{b_{\text{thrower}} v_{\max}}{b_{\text{thrower}} + b}$. We also provide a vertical force $F_y = mg - k_c y - b_c \dot{y}$, where the subscript c stands for "constraint". The first term is to cancel gravity, and the other two terms are a damped harmonic oscillator

that try to keep particles close to the middle of the thrower. We have tuned b_c and k_c to form a critically damped harmonic oscillator for a balance between minimizing oscillation and allowing particles to get towards the middle. Further, we tune ω_0 such that the time scale of decay is smaller than the time scale of points travelling from one side of the rectangle to the other. This means that $2\pi/\omega_0 < L_{\text{box}}/v_{\max}$. To be safe, we set $\omega_0 = 2 \cdot (2\pi v_{\max}/L_{\text{box}})$.

For particles inside the circles and outside the rectangle, we provide a radial force analogous to F_y for the rectangle, so as to push particles outward. For simplicity, we also provide an additional mg force upwards for all particles inside the circle to cancel gravity and help particles above the circles stay outside. The radial force is $-b_c v_r - k_c(r - R_{\text{circle}}))$.

We impose initial conditions of the rope dangling at rest in a square loop, with side length 1/4 of the total rope rest length and top edge lying at the centre of the rectangle, as shown in Figure 7. We run the

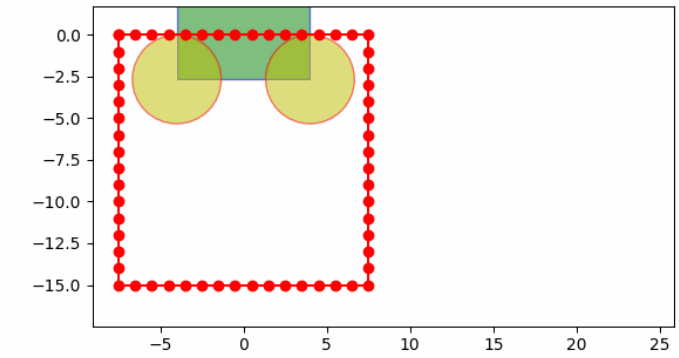


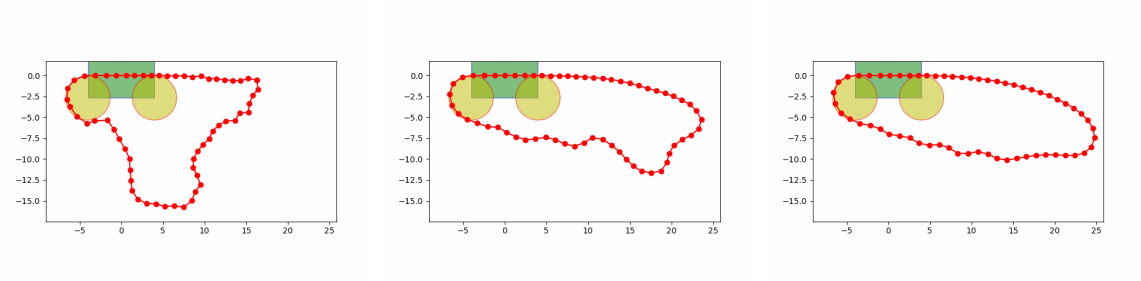
Figure 7: Initial condition for string shooter.

simulation with the parameters $N = 60, m = 1, k = 5000, k_2 = 500, l = 1, 9 = 9.8, L_{\text{box}} = 8, W_{\text{box}} = 8/3$, and $v_{\max} = 5$. We run three cases of varying air resistance – $b = 1$ for high air resistance, $b = 0.7$ for medium air resistance, and $b = 0.1$ for low air resistance, shown in Figures 8, 9, 10. It can be seen that the inclination angle of the loop is higher for higher air resistance, matching experimental observations. Further, at later times, propagating waves can be observed on the lower end of the string in the high and medium air resistance cases and on both ends of the string in the low air resistance case, matching results from the video.

Our model of a damped harmonic oscillator attracting the rope to the circle has physical inaccuracies. Namely, it is possible for the rope to get "stuck" on the circles even when it is in contact, but moving away, from them, as is the case for the circle on the right. This behaviour can be seen in the case of low air resistance – in the `low_air_resistance.gif` file included with this write-up, one can see that the rope "bunches up" and gets stuck near the circle on the right. A couple ways of correcting for this behaviour is to turn off the harmonic oscillator force from the circles for sections of the rope moving with positive radial velocity or moving the whole circle down. An example of the second approach is shown in Figure 11.

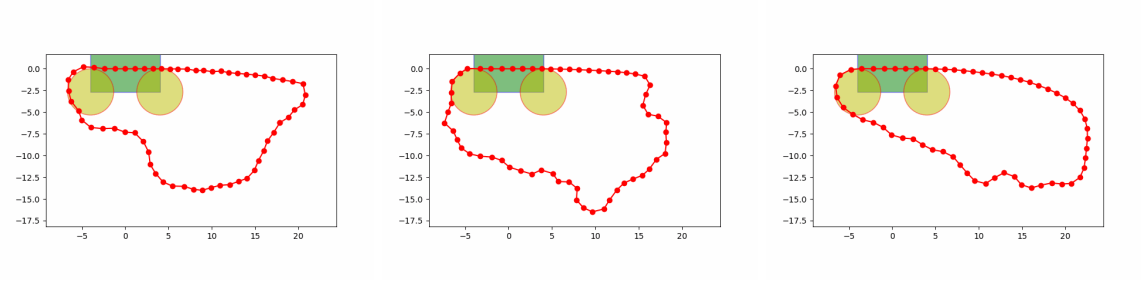
5 How to Run the Code

All code is in a Jupyter notebook. We built the code using the Deepnote platform, which allows us to collaborate on a Jupyter notebook simultaneously and run it. The longest simulations, using $N = 60$, $t_{\text{final}} = 5$, $\Delta t = 0.001$, took less than 5 seconds, and the longest animation exports using Matplotlib took around 30 seconds.



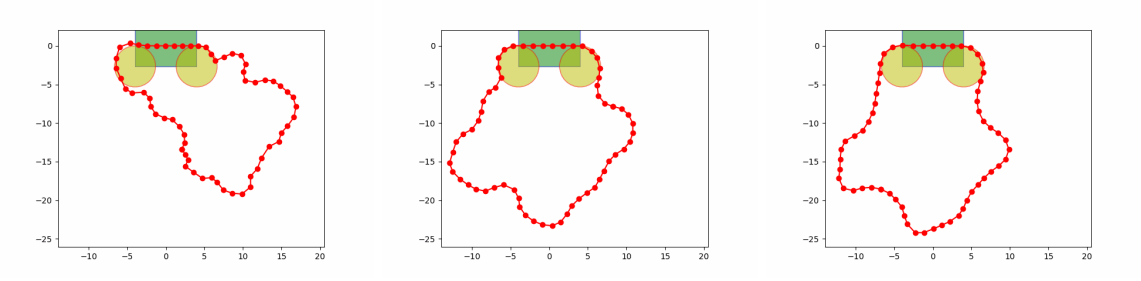
(a) (b) (c)

Figure 8: Selected frames of the string shooter problem with high air resistance.



(a) (b) (c)

Figure 9: Selected frames of the string shooter problem with medium air resistance.



(a) (b) (c)

Figure 10: Selected frames of the string shooter problem with low air resistance.

6 Our Contributions

We worked on pretty much everything together, both collaboratively and in parallel, on generating ideas for how to implement certain things like the string shooter, writing the code itself, and putting together this writeup.

7 Some resources

- https://youtu.be/q-_7y0WUnW4
- https://youtu.be/2ANnMsyunkk
- https://math.libretexts.org/Bookshelves/Differential_Equations/Numerically_Solving_Ordinary_Differential_Equations_(Brorson)/01%3A_Chapters/1.07%3A_Symplectic_integrators

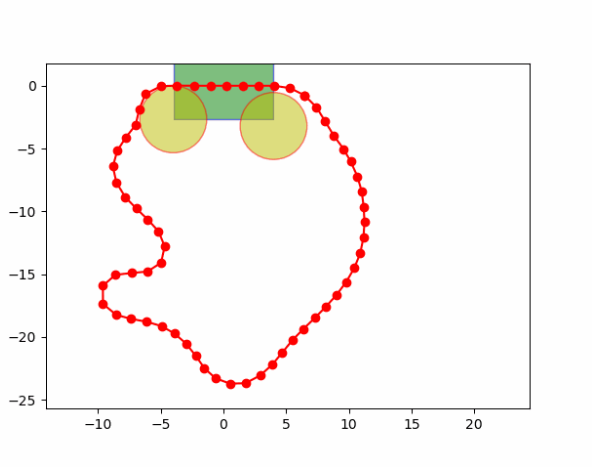


Figure 11: Frame from low air resistance case with right constraint circle moved down to avoid spurious interaction.

- https://www.aapt.org/physicsteam/2015/upload/exam1-2014-2-2.pdf
- https://www.owlree.blog/posts/simulating-a-rope.html
- https://www.toptal.com/game/video-game-physics-part-iii-constrained-rigid-body-simulation
- https://medium.com/@szewczyk.franciszek02/rope-simulator-in-c-a595a3ef956c
- https://www.cs.cmu.edu/afs/cs/academic/class/15462-s13/www/lec_slides/Jakobsen.pdf
- https://matthias-research.github.io/pages/publications/posBasedDyn.pdf
- https://pybullet.org/Bullet/phpBB3/viewtopic.php?t=1585
- https://www.algorithm-archive.org/contents/verlet_integration/verlet_integration.html