A-level Physics Teaching Aid

Alex Makelberge

May 2023

Contents

1 Analysis

1.1 Project Problem Definition

Client: Charterhouse School Physics department

- Thomas Marlow - Teacher

Contact: Thomas Marlow
Prince's Avenue
Godalming
Surrey
GU7 2DX
07514 674203

1.2 Background to problem

Charterhouse Physics department is a pivotal part of the school. All of Charterhouse's 1000 students learn physics at some point. There are 6 classes of A-level students, which teach over 100 students. Dr Marlow likes to visualise complex physics concepts using diagrams and descriptions.

Currently Dr Marlow draws diagrams on whiteboards and prints out diagrams on paper. This method can be quite ineffective when trying to explain moving objects such as planets and circular motion. Students only see snapshots and have to fill in the rest of the concept in their head. This can result in people getting the wrong idea and could potentially hinder their learning in the future.

The Physics department have asked for my help in making an interactive teaching aid that displays various different physics phenomena and models and asks exemplar questions based on the phenomena in play.

1.3 Interview

1.3.1 Dr Marlow - Teacher

What systems are you currently using to teach the A-level students?

TM: I often use Phet and myPhysicsLab to help visualise phenomena, with various other animations from other sources. I regularly use PASCO datalogging software too, when completing demos and practicals.

What are the benefits of the current method you are using?

TM: I really enjoy teaching using the animations as it is very visual and allows pupils to see abstract concepts in action, and also it promotes them to investigate phenomena by using such animation tools. Also there are often glitches or differences in approach or settings that can highlight misconceptions, again in a very visual way. I especially like animations that link to graphs as that also is a visual way to approach problems. It encourages the use of sketch diagrams in pupil's work which often is very useful!

What are the drawbacks of the current method you are using?

TM: It is perhaps less hands on, and perhaps sometimes gives pupils the illusion of understanding a phenomena (as they can see it in action) when perhaps they don't actually fully understand what is going on. There are some teachers that don't like using animations for exactly this reason. Additionally, I find it hard to find questions that integrate well with the animations at hand and I also struggle with manipulating the animations so that they do exactly what I want to demonstrate.

What different models/phenomena would be the most useful to model?

TM: I often struggle to find good animations of the double pendulum problem that highlight its chaotic nature, where a small change in initial conditions results in a very large change in the final conditions.

What other features would be useful for teaching?

TM: A video tutorial of someone using the animation and outlining the features of it. Also making it intuitive enough for pupils to use the simulation themselves. For longer topics that take multiple lessons to explain, it would be nice to be able to save my progress and settings of the simulations so I can quickly come back to them.

1.3.2 Conclusion

There are some good simulations mentioned such as Phet and myPhysicsLab that I am going to analyse as existing systems. Additionally, I am going to link the simulations to graphs and make the system as intuitive to use as possible. I could add questions add the end of the simulation to engage the students to aid them in a better understanding of the phenomena that they are learning.

1.4 Simulations to pick

There are many different physics phenomena that one can try to model.
Here are a few options:[1]

1. N-body problem
2. Double pendulum problem
3. Turbulent water flow
4. Plasma physics

This simulation is for A-level physics and therefore should be understandable for A-level physics students. The only simulations that are of that standard are the n-body problem and double pendulum problem. The n-body problem can be modelled with Newton's laws of gravity which is taught at A level, and the Double Pendulum problem is an extension of the single pendulum with periodic motion taught at physics. Mr Marlow made a preference for the Double Pendulum problem and therefore I will be focusing on that. There isn't enough time to make multiple simulations.

1.5 N-body problem definition

The double pendulum problem[5] is made from 2 pendulums attached end to end. The first pendulum hangs from a fixed point and has a mass at the end. The second pendulum hangs from the end of the first pendulum and has a mass at its end as well. The motion of the first pendulum depends on the motion of the second pendulum, and vice verse, making very nonlinear motion.

When modelling the double pendulum problem, typically this means solving the position of the pendulum over time which I will be doing. When set into motion the pendulum exhibits chaotic motion, i.e. it is very sensitive to initial conditions. This makes it impossible to predict in the long run what will happen, as small errors in numerical integration and equations can result in wildly different results.

1.6 Existing Systems

I had a look at some of the current teaching aids that are available online. There are lots of different ones available but I have narrowed it down to 2 based on the suggestions from Dr Marlow in the interview.

1. PhET: https://phet.colorado.edu/en/simulations/browse
2. myPhysicsLab: https://www.myphysicslab.com

1.6.1 PhET

'PhET provides fun, free, interactive, research-based science and mathematics simulations. We extensively test and evaluate each simulation to ensure educational effectiveness. These tests include student interviews and observation of simulation use in classrooms. The simulations are written in HTML5 (with some legacy simulations in Java or Flash), and can be run online or downloaded to your computer.' [2]

It currently works like this:

1. There is a menu [Figure 1] in which you can browse many different simulations. These are organised per subject that they teach. (Physics, Chemistry, Maths, etc.) Additionally, there is a filter [Figure 2] that will show select simulations based on if they fit the criteria that you are looking for. The criteria are based on which topics are covered, what education level, what coding language they are written in, the features they have, and what language they are written in.

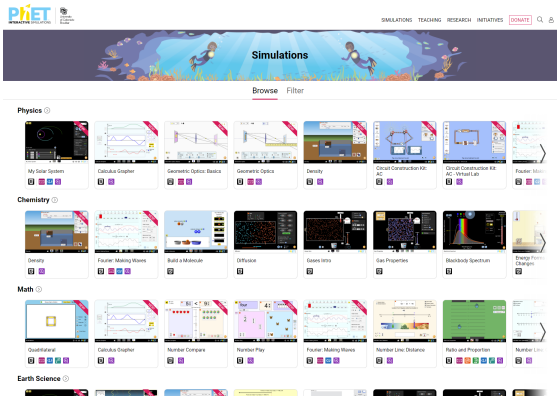


Figure 1: Browse Menu for PhET

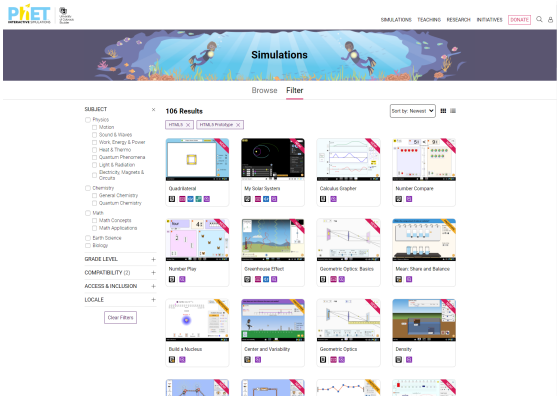


Figure 2: Filter Menu for PhET

2. Once a simulation has been selected. You click on it and it goes the the page of that simulation.[Figure 3] This page tells you some information about the simulation such as which topics it covers, and the features. It distinctly lacks any descriptions about the specific phenomena related to the simulation (i.e. a brief description of what it is).

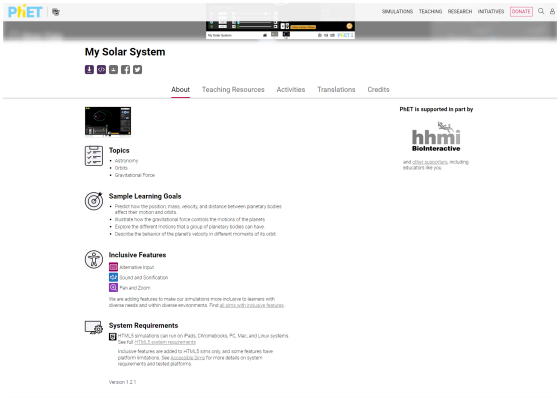


Figure 3: Description Page of the Simulation

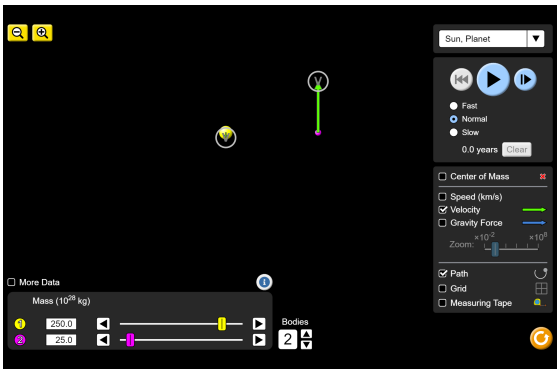


Figure 4: The Simulation Page

3. Within the simulation [Figure 4] there are numerous options:
   (a) Options to change the masses, position, and velocity of each body.
   (b) Choose the number of bodies in the simulation
   (c) A panel to display certain aspects of the simulation such as the centre of mass, the velocity, forces, path, grid, etc.
   (d) A panel to choose the speed at which the simulation runs.
   (e) A panel with some predefined options such as Sun with Planet, Ellipses, Hyperbolics, Binary Systems, etc.
   (f) You can drag and drop the bodies in the space provided and place them anywhere you would like.
   (g) You can change the initial velocities with arrows on the page. You can also change the velocity of the body midway through the simulation by pausing and moving the arrow.
   (h) An introductory simulation that simplifies to get the hang of things quicker.
   (i) A button to change the position of the camera so that it goes to the centre of mass of all the bodies in the system.

1.6.2 MyPhysicsLab

'I started developing this website as a way to practice what I was learning. I am now happy to use excellent tools such as HTML and JavaScript, and leave their development to others. I continue to work on physics simulations, with several new ones in development.' [3]

It currently works like this:

1. There is a menu [Figure 5] with many different simulations to choose from. They are all in an array where similar simulations are next to each other. There is a feature where you can use all the simulations in a series and click next on each one to bring you to the next simulation once you are done with it. Underneath the simulations are different sections of information about how to share the simulations and that they are all open-source on GitHub. [4]

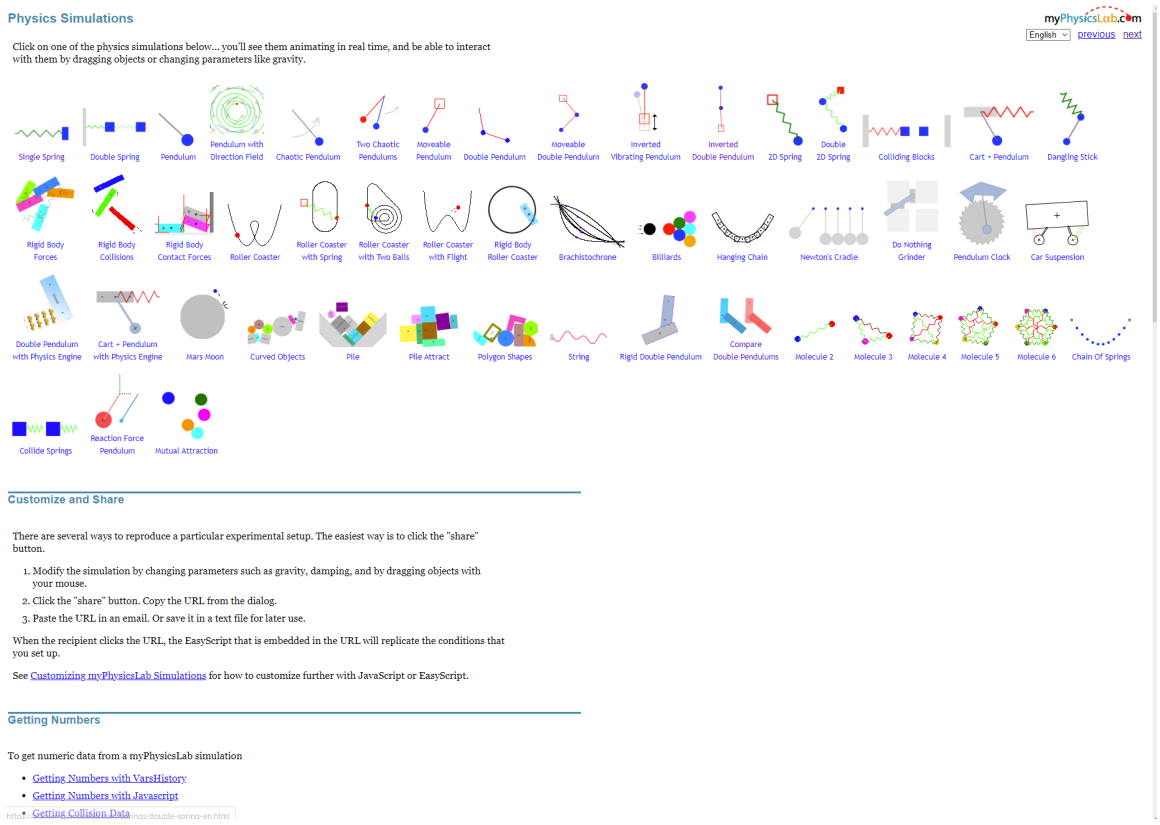


Figure 5: Browse Menu for PhET

2. Once a simulation has been selected. You click on it and it goes directly to the simulation running. [Figure 6] There are some initial parameters about the simulation that you can customise at it is running.

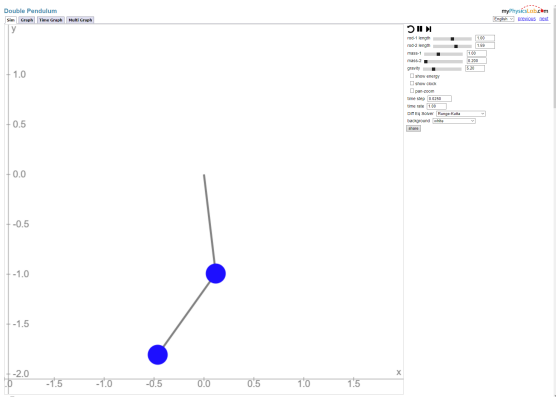


Figure 6: Simulation Page

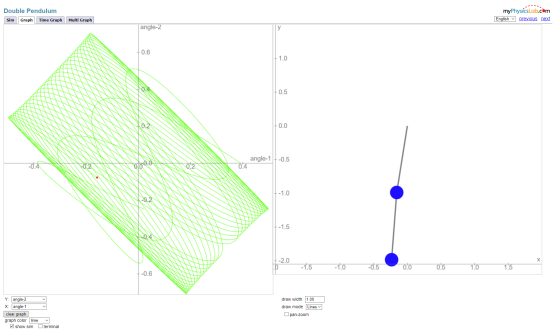


Figure 7: Simulation with graph

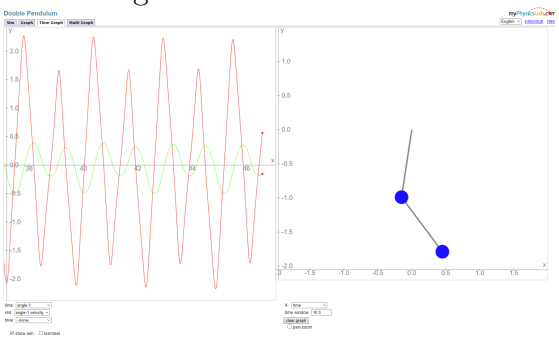


Figure 8: Simulation with Time Graph

3. Within the simulation [Figure 7][Figure 8] there are some different pages:

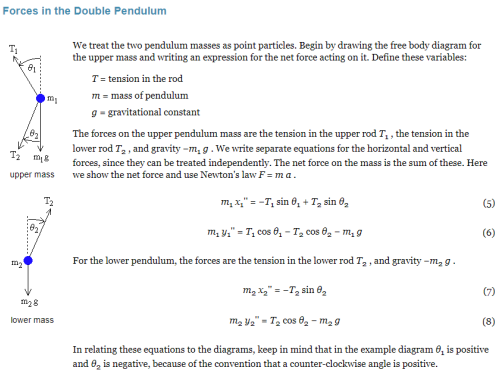
   (a) There is a general simulation page that has the simulation on full and all of the options are on the right.

   (b) A page which graphs the angles between the pendulum as it swings while the simulation is running (updates in real-time).

   (c) A page which graphs angles over time.

   (d) All the graphs can be customized to display anything you want.

   (e) There is a bar that displays the energies of the system.



Forces in the Double Pendulum

We treat the two pendulum masses as point particles. Begin by drawing the free body diagram for the upper mass and writing an expression for the net force acting on it. Define these variables:

T = tension in the rod
m = mass of pendulum
g = gravitational constant

The forces on the upper pendulum mass are the tension in the upper rod T_1, the tension in the lower rod T_2, and gravity $-m_1 g$. We write separate equations for the horizontal and vertical forces, since they can be treated independently. The net force on the mass is the sum of these. Here we show the net force and use Newton's law $F = m a$.

$$m_1 x_1'' = -T_1 \sin\theta_1 + T_2 \sin\theta_2 \quad (5)$$

$$m_1 y_1'' = T_1 \cos\theta_1 - T_2 \cos\theta_2 - m_1 g \quad (6)$$

For the lower pendulum, the forces are the tension in the lower rod T_2, and gravity $-m_2 g$.

$$m_2 x_2'' = -T_2 \sin\theta_2 \quad (7)$$

$$m_2 y_2'' = T_2 \cos\theta_2 - m_2 g \quad (8)$$

In relating these equations to the diagrams, keep in mind that in the example diagram θ_1 is positive and θ_2 is negative, because of the convention that a counter-clockwise angle is positive.

Figure 9: Mathematical description

4. Underneath the simulation is a descriptions and explanation of all of the maths [Figure 9] involved in the simulation and the physical phenomena. It goes heavily in depth an gives a complete understanding for the user.

1.6.3 Conclusion

There are some good ideas from these websites that would benefit my project. Features that will provide the best use:

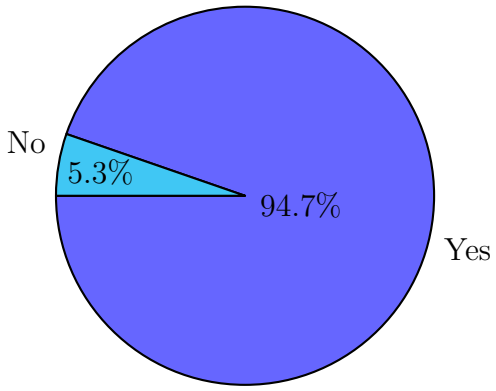
- Lots of customizability in the options within each simulation
- Being able to adjust the simulation in real-time as it is happening
- Graphs to display the data from within the simulation
- An in-depth explanation of the physics and maths behind all of the phenomena.

1.7 Questionnaire

Link to my form: https://forms.gle/7gabZBigmyTP8Nco6

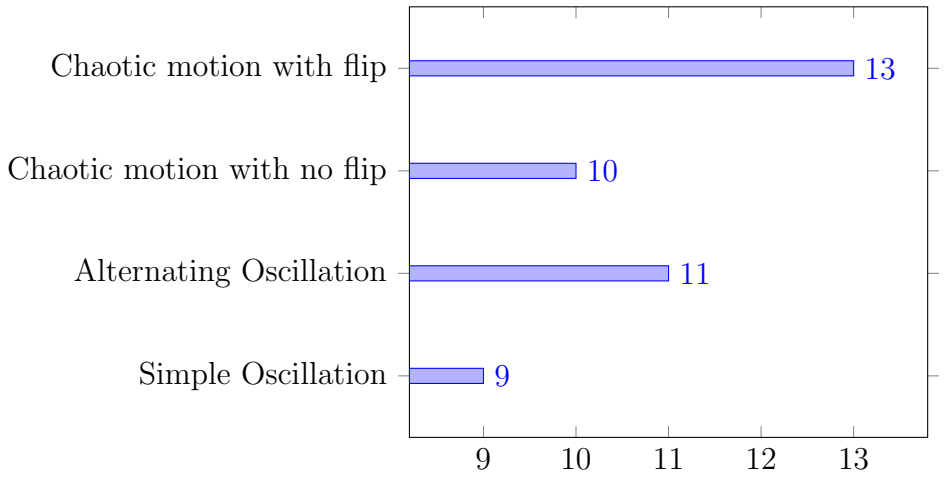
Questions:

- **Do you find visual aids useful when trying to learn about a topic?**



The vast majority find visual aids useful, therefore I need to make the simulation as interactive as possible and convey ideas through the simulation as much as possible. Additionally, in the explanations there should be lots of diagrams as visual aid for better learning.

- **Suppose there is a simulation of a double pendulum, what demos would you want?**

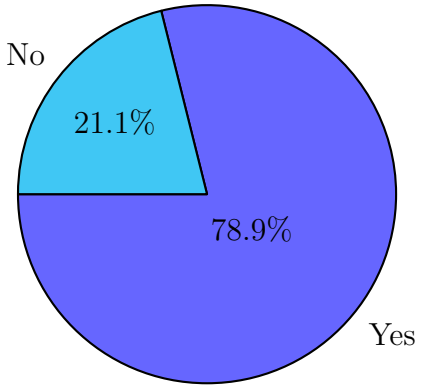


Extra suggestions:

- Spinning in a circle
- Not moving at the top
- Not moving at the bottom

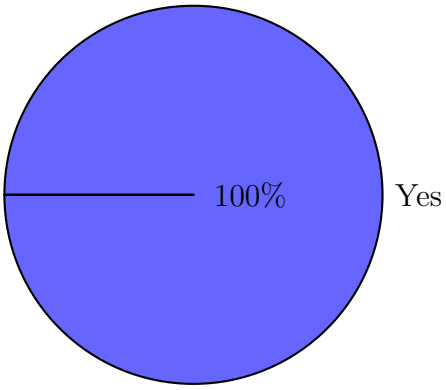
All of the simulations initially proposed were very popular and I think I shall incorporate all of them into the simulation. For the extra suggestions, I will initially add all of them but if a user wants to add a preset that they can't they should be able to. Therefore, I should add functionality to add their own presets to the simulation whenever they want.

- **Do you want to be able to click on the pendulum to set it where you want?**



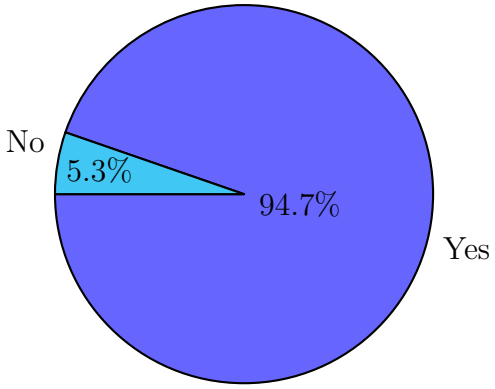
I will definitely add functionality to the simulation so that you can click on the pendulum to set its position.

- **Would you want graphs that describe the motion and get updated live as the simulation runs?**



As was suggested by Dr Marlow, live updated graphs definitely need to be incorporated into the simulation.

- **Do you want the graph to display the data so that you can interact with it?**



This is definitely wanted. I will add it so you can click on the graph and it will display a point when you click on it.

1.8 Features

This is a short breakdown of everything that is needed for the simulation itself and how it would get solved.

Features:

1. Correctly running pendulum
    (a) Differential
    (b) Numerical integration
    (c) Draw to the screen
2. Being able to click on the pendulum to set the position of it
    (a) Get where clicked
    (b) Calculate points where first pendulum can go
    (c) Calculate angle of pendulums
3. Control pendulum with dynamic variables
    (a) Every frame update variables from sliders
4. Draw the graph as it runs
    (a) Every frame redraw the graph
    (b) Draw a line of all the points there are
5. Be able to get closest point when clicking on graph
    (a) Get where clicked
    (b) Use KNN to get closest point
    (c) Display this point
6. Display energy bars
    (a) Every frame update the values for potential and kinetic energy
    (b) Graph them to the screen
7. Presets
    (a) Get preset from user
    (b) Store onto the database
    (c) Retrieve from the database

1.9 Technical Research

1.9.1 Double pendulum

The double pendulum consists of an upper pendulum and a lower pendulum, where each pendulum has its own mass and length. The upper pendulum is attached to a fixed point, and the lower pendulum is attached to the end of the upper pendulum. The masses of the pendulums are denoted as m_1 and m_2 with corresponding positions (x1, y1) and (x2, y2), while the lengths of the pendulums are denoted as l_1 and l_2. The gravitational acceleration is represented by g.

The behavior of the double pendulum can be described using the equations of motion. These equations involve the angles made by each pendulum with respect to a reference vertical line. Let θ_1 represent the angle of the upper pendulum with the vertical, and θ_2 represent the angle of the lower pendulum with respect to the upper pendulum.

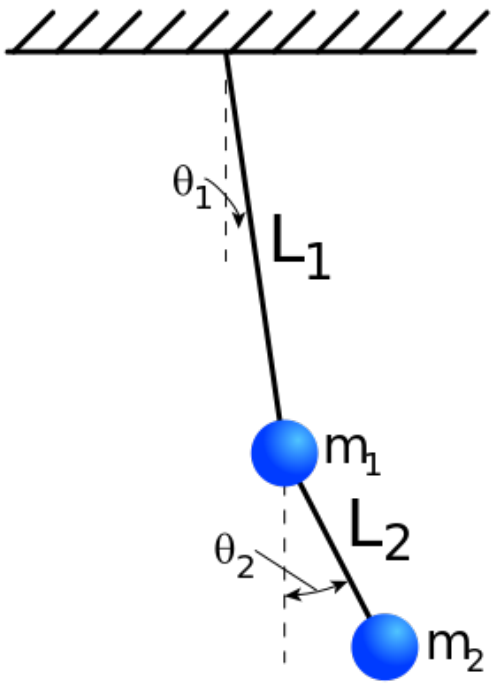


Figure 10: Diagram

For mass m_1, the forces acting on it are its weight ($m_1 \times g$) and the tension in the rod connecting it to mass m_2. The tension can be decomposed into horizontal and vertical components.

The positions of the masses are given by [5]:

$$x_1 = l_1 \sin\theta_1 \tag{1}$$

$$y_1 = -l_1 \cos\theta_1 \tag{2}$$

$$x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 \tag{3}$$

$$y_2 = -l_1 \cos\theta_1 - l_2 \cos\theta_2 \tag{4}$$

The potential energy of the system is then given by:

$$V = m_1 g y_1 + m_2 g y_2 \tag{5}$$

$$= -(m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \tag{6}$$

The kinetic energy is given by:

$$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \tag{7}$$

$$T = \frac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 [l_1^2 \dot{\theta}_1^2 + l_2^2 \dot{\theta}_2^2 + 2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2)] \tag{8}$$

The Lagrangian is then:

$$L \equiv T - V \tag{9}$$

$$= \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + (m_1 + m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos\theta_2 \tag{10}$$

Therefore, the momenta with θ_1 and θ_2 can be obtained from L:

$$p_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2) l_1^2 \dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{11}$$

$$p_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = m_2 l_2^2 \dot{\theta}_2 + m_2 l_1 l_2 \dot{\theta}_1 \cos(\theta_1 - \theta_2) \tag{12}$$

The equations of motion of the system are the Euler-Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = 0 \Longrightarrow \frac{dp_{\theta_i}}{dt} - \frac{\partial L}{\partial \theta_i} = 0 \text{ for } i = 1, 2 \tag{13}$$

Since:

$$\begin{aligned} \frac{dp_{\theta_1}}{dt} &= (m_1 + m_2)l_1^2\ddot{\theta}_1 + m_2 l_1 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &\quad -m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + m_2 l_1 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \end{aligned} \tag{14}$$

$$\begin{aligned} \frac{dp_{\theta_2}}{dt} &= m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) \\ &\quad -m_2 l_1 l_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) \end{aligned} \tag{15}$$

$$\frac{\partial L}{\partial \theta_1} = -m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - (m_1 + m_2) g l_1 \sin\theta_1 \tag{16}$$

$$\frac{\partial L}{\partial \theta_2} = m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - m_2 g l_2 \sin\theta_2 \tag{17}$$

Then plugging into equation (19), you get (after dividing by l_1 when $i = 1$ and by $m_2 l_2$ when $i = 2$):

$$\begin{aligned} (m_1 + m_2)l_1\ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \\ +m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + (m_1 + m_2) g \sin\theta_1 = 0 \end{aligned} \tag{18}$$

$$l_2\ddot{\theta}_2 + l_1\ddot{\theta}_1 \cos(\theta_1 - \theta_2) - l_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + g\sin\theta_2 = 0 \tag{19}$$

Equations (24) and (25) form a system of coupled second-order nonlinear differential equations. Dividing equation (24) by $(m_1 + m_2)l_1$ and equation (16) by l_2 and also moving all terms which do not involve $\ddot{\theta}_1$ and $\ddot{\theta}_2$ to the right-hand side, we obtain:

$$\ddot{\theta}_1 + \alpha_1(\theta_1, \theta_2)\ddot{\theta}_2 = f_1(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \tag{20}$$

$$\ddot{\theta}_2 + \alpha_2(\theta_1, \theta_2)\ddot{\theta}_1 = f_2(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \tag{21}$$

where:

$$\alpha_1(\theta_1, \theta_2) := \frac{l_2}{l_1}\left(\frac{m_2}{m_1 + m_2}\right)\cos(\theta_1 - \theta_2) \tag{22}$$

$$\alpha_2(\theta_1, \theta_2) := \frac{l_1}{l_2}\cos(\theta_1 - \theta_2) \tag{23}$$

and:

$$f_1(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) := -\frac{l_2}{l_1}\left(\frac{m_2}{m_1+m_2}\right)\dot{\theta}_2^2 \sin(\theta_1-\theta_2) - \frac{g}{l_1}\sin\theta_1 \tag{24}$$

$$f_2(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) := \frac{l_1}{l_2}\dot{\theta}_1^2 \sin(\theta_1-\theta_2) - \frac{g}{l_2}\sin\theta_2 \tag{25}$$

f_1 does not depend on $\dot{\theta}_1$ and f_2 does not depend on $\dot{\theta}_2$. Equations (26) and (27) can be combined into a single equation:

$$A\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{pmatrix} = \begin{pmatrix}1 & \alpha_1\\ \alpha_2 & 1\end{pmatrix}\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{pmatrix} = \begin{pmatrix}f_1\\ f_2\end{pmatrix} \tag{26}$$

where the matrix A depends on θ_1 and θ_2 since α_1 and α_2 depend on these variables. Being a 2 x 2 matrix, A can be inverted directly:

$$A^{-1} = \frac{1}{\det(A)}\begin{pmatrix}1 & -\alpha_1\\ -\alpha_2 & 1\end{pmatrix} = \frac{1}{1-\alpha_1\alpha_2}\begin{pmatrix}1 & -\alpha_1\\ -\alpha_2 & 1\end{pmatrix} \tag{27}$$

Notice that A is always invertible since:

$$\det(A) = 1-\alpha_1\alpha_2 = 1 - \left(\frac{m_2}{m_1+m_2}\right)\cos^2(\theta_1-\theta_2) > 0 \tag{28}$$

because $m_2/(m_1+m_2) < 1$ and $\cos^2(x) \leq 1$ for all real values of x. From equations (32) and (33) we obtain:

$$\begin{pmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{pmatrix} = A^{-1}\begin{pmatrix}f_1\\ f_2\end{pmatrix} = \frac{1}{1-\alpha_1\alpha_2}\begin{pmatrix}f_1-\alpha_1 f_2\\ -\alpha_2 f_1 + f_2\end{pmatrix} \tag{29}$$

Finally, letting $\omega_1 := \dot{\theta}_1$ and $\omega_2 := \dot{\theta}_2$, we can write the equations of motion of the double pendulum as a system of coupled first order differential equations on the variables $\theta_1, \theta_2, \omega_1, \omega_2$:

$$\frac{d}{dt}\begin{pmatrix}\theta_1\\ \theta_2\\ \omega_1\\ \omega_2\end{pmatrix} = \begin{pmatrix}\omega_1\\ \omega_2\\ g_1(\theta_1,\theta_2,\omega_1,\omega_2)\\ g_2(\theta_1,\theta_2,\omega_1,\omega_2)\end{pmatrix} \tag{30}$$

$$g_1 := \frac{f_1-\alpha_1 f_2}{1-\alpha_1\alpha_2} \quad g_2 := \frac{-\alpha_2 f_1 + f_2}{1-\alpha_1\alpha_2} \tag{31}$$

Equation (30) can be solved numerically using a numerical method.

1.9.2 Assumptions

When modelling something in the real world, it is extremely complex and therefore certain assumptions must be made to make the model feasable to run.[6]

Assumptions:

1. **Rigid rods:** The rods connecting the masses are assumed to be rigid, they do not bend, stretch, or compress. This simplifies the equations of motion as it eliminates the need to consider elastic deformations or other complexities associated with flexible rods.
2. **Massless Rods:** The rods are assumed to be massless compared to the masses connected to the rods. If the rods have significant mass, their distribution would need to be considered, complicating the equations.
3. **Frictionless:** There is assumed to be no friction, that includes friction between the pivots of the pendulum and air resistance. Introducing friction would make the system non-conservative and add a damping force.
4. **Two-dimensional motion:** Motion is assumed to be in 2 dimension, in x and y. This makes it easier to represent on a 2D screen and also reduces calculations.
5. **Point masses:** The masses at the end of the pendulums are assumed to be point masses through which gravity and other force act.

1.9.3 Numerical methods

These are methods to numerically approximate second order coupled differential equations which are required to find values for the next values of θ for the next frame in the simulation. Let:

$$\frac{dy}{dx} = f(x, y) \tag{32}$$

Where the initial conditions are given by:

$$y(x_0) = y_0 \tag{33}$$

and h is the step-size for the next x value.

Euler Method[7]:

Also known as a 1st order Runge-Kutta Method is not very accurate but is very simple. The formula is:

$$y_1 = y_0 + hf(x_0, y_0) \tag{34}$$

4th Order Runge-Kutta Method[8]:

This is a very accurate numerical method that solves very efficiently and correctly. The formula is:

$$\begin{aligned}
k_1 &= f(x_n, y_n) && (35)\\
k_2 &= f\left(x_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right) && (36)\\
k_3 &= f\left(x_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right) && (37)\\
k_4 &= f(x_n + h, y_n + hk_3) && (38)\\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) && (39)\\
x_{n+1} &= x_n + h && (40)
\end{aligned}$$

Euler vs RK4:

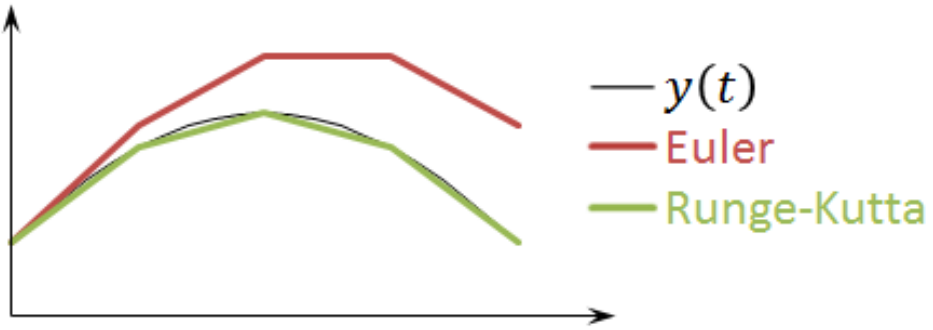


Figure 11: Direct comparison of RK4 and Euler

RK4 is considerably more accurate than the Euler method.

Conclusion

RK4 is a lot more accurate than Euler method and therefore should definitely be used in the simulation. What would be even better if you can change between Euler and RK4 and directly compare both of them next to each other with the same starting conditions.

1.9.4 Nearest Neighbour algorithm[10]

This algorithm will be used on the graph. When clicking on the graph, the closest point that is recorded will be displayed on the screen. For this an algorithm is needed to calculate which point to display.

Let (x_0, y_0) be the point where the mouse is clicked and (x_i, y_i) be an arbitrary point that is stored in the list of points on the graph.

Calculate the distance of all the points to the clicked point and store it with each point in the array:

$$d_1 = \sqrt{(x_i - x_0)^2 + (y_i - x_0)^2} \tag{41}$$

Sort the array my this distance measure in descending order and take the first element to get the smallest value.

If you wish to graph multiple point this is easy as you just take the next value in the array as it is sorted.

1.9.5 Energy bars

A bar that shows the current energy levels of the simulation will be on display next to the pendulum. The sum of energy should be constant.[9]

Kinetic energy

The general equation for kinetic energy is given by:

$$E_k = \frac{1}{2}mv^2 \tag{42}$$

For a system of masses the total kinetic energy is given where N is the index of each mass:

$$E_k = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \tag{43}$$

For the double pendulum there are two masses so the equation becomes:

$$E_k = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \tag{44}$$

Some values from the pendulum to substitute into the equation:

$$E_k = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \tag{45}$$

$$E_k = \frac{1}{2}m_1 (l_1\omega_1)^2 + \frac{1}{2}m_2 (l_2\omega_2)^2 \tag{46}$$

Which finally becomes:

$$E_k = \frac{1}{2}m_1 l_1^2\omega_1^2 + \frac{1}{2}m_2 l_1^2\omega_2^2 \tag{47}$$

Potential energy

The general equal for potential energy is given by:

$$E_p = mgh \tag{48}$$

For a system of masses, the total potential energy is given where N is the index of each mass:

$$E_p = \sum_{i=1}^{N} m_i g h_i \tag{49}$$

For the double pendulum there are two masses so the equation becomes:

$$E_p = m_1 g h_1 + m_2 g h_2 \tag{50}$$

If we define the bottom possible position of the pendulum as 0 height, then calculating the heights becomes:

$$\begin{aligned} h_1 &= l_1(1 - \cos\theta_1) + l_2 && (51)\\ h_2 &= h_1 - l_2 \cos\theta_2 && (52)\\ h_2 &= l_1(1 - \cos\theta_1) + l_2(1 - \cos\theta_2) && (53) \end{aligned}$$

Finally total potential energy becomes:

$$E_p = m_1 g(l_1(1 - \cos\theta_1) + l_2) + m_2 g(l_1(1 - \cos\theta_1) + l_2(1 - \cos\theta_2)) \tag{54}$$

1.9.6 Set the position of the pendulum[11]

When clicking on the pendulum it should put the bottom pendulum at the point where you clicked. Therefore it needs to calculate what angles make that possible.

We model this as finding the points of intersection of 2 circles. There is a circle of places one line can go, and a circle where the other line can go, therefore we want where both coincide. First you need to check whether it is possible to set the pendulum to where has been clicked.
Let (x_0, y_0) be the top point of the pendulum and (x_1, y_1) be the point clicked. Find the distance between these points:

$$R = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{55}$$

It is only possible if this condition is met:

$$|l_1 - l_2| \leq R \leq l_1 + l_2 \tag{56}$$

We want to find the distance between the centre of our circle and the line that joins the points of intersection. Let this be a, it is calculated by:

$$a = \frac{l_1^2 - l_2^2 + R^2}{2R} \tag{57}$$

Then we want the distance h which is the shortest distance from a point of intersection to the line that connects the two centres of the circles. h is given by:

$$h^2 = l_1^2 - a^2 \tag{58}$$

The point at which the line that connects the points of intersections and the line that connects the centres intersect is needed:

$$x_2 = x_0 + \frac{a(x_1 - x_0)}{d} \tag{59}$$

$$y_2 = y_0 + \frac{a(y_1 - y_0)}{d} \tag{60}$$

Then the points of intersection become:

$$x_3 = x_2 \pm \frac{h(y_1 - y_0)}{d} \tag{61}$$

$$y_3 = y_2 \pm \frac{h(x_1 - x_0)}{d} \tag{62}$$

We don't need both points so we arbitrarily take the positive point so our (x_3, y_3) becomes:

$$x_3 = x_2 + \frac{h(y_1 - y_0)}{d} \tag{63}$$

$$y_3 = y_2 + \frac{h(x_1 - x_0)}{d} \tag{64}$$

Our pendulum is in terms of the anti-clockwise angles that the pendulums make with the negative y-axis. Therefore once we have the intersection point we need to calculate these angles and update them on the simulation.

Here is the calculation of θ_1 and θ_2 given the points:

$$\theta_1 = \tan^{-1} \frac{y_3 - y_2}{x_3 - x_2} \tag{65}$$

$$\theta_2 = \tan^{-1} \frac{y_2 - y_1}{x_2 - x_1} \tag{66}$$

We set the angular velocity to zero:

$$\omega_1 = 0 \tag{67}$$

$$\omega_2 = 0 \tag{68}$$

1.9.7 Databases

A database is needed to store all of the presets/demos and I want people to be able to add their own presets to the system.

Benefits of an online database:[12]

1. **Minimal maintenance:** The service provider manages the infrastructure, security patches, backups, and up-time, which makes it very keep the database up and running.
2. **Ease of setup:** No database needs to be downloaded to run the application on a server which makes it significantly easier to run on less-powerful machines.
3. **Scalability:** If the project ever gets bigger and more complex data structures are needed, it is very easy to add these.

Benefits of a local database:

1. **Control:** Having everything stored locally gives you full control over the security of the database allowing data to stay very secure.
2. **Performance:** No need to wait for an external database to respond which will increase performance as just a local lookup is needed.

I will use an online database because the minimal maintenance and ease of setup are particularly valuable to me to allow me to run the project on very simple hardware. No sensitive data will be stored on the database, therefore having control over the security doesn't make a big difference. Equally, the database will only be queried when you load the website or send a preset, which isn't very often. Also, traffic will be low on the website and therefore performance isn't as big an issue. I am going to use MongoDB for my online database as it is what I am most familiar with and it gives you free storage for small databases.

1.10 IPSO (Input Process Storage Output)

Differential

Input	Process
g, m_1, m_2, l_1, l_2 $\theta_1, \theta_2, \omega_1, \omega_2$	Calculate the value given by the coupled second order differential equation.
Storage	Output
N/A	$\omega_1, \omega_2, \dot{\omega}_1, \dot{\omega}_2$

RK4

Input	Process
h, θ_1, θ_2 ω_1, ω_2	Numerically integrate the next values of $\theta_1, \theta_2, \omega_1, \omega_2$ based on h by RK4
Storage	Output
N/A	$\theta_1, \theta_2, \omega_1, \omega_2$

Euler method

Input	Process
h, θ_1, θ_2 ω_1, ω_2	Numerically integrate the next values of $\theta_1, \theta_2, \omega_1, \omega_2$ based on h by Euler
Storage	Output
N/A	$\theta_1, \theta_2, \omega_1, \omega_2$

Draw pendulum

Input	Process
$m_1, m_2, l_1, l_2, \theta_1, \theta_2$ width, height	Draw the pendulum to the screen
Storage	Output
N/A	Lines on screen

Nearest Neighbour algorithm

Input	Process
x, y, points	get the point on the graph closest to where was clicked
Storage	Output
N/A	x, y of the closest point

Draw closest point

Input	Process
x, y of the closest point	Draw the point on the graph
Storage	Output
N/A	An outline of the point on the graph

Calculate Kinetic Energy

Input	Process
$m_1, m_2, l_1, l_2, \omega_1, \omega_2$	Calculate the total kinetic energy
Storage	Output
N/A	E_k

Calculate Potential energy

Input	Process
$g, m_1, m_2, l_1, l_2, \theta_1, \theta_2$	Calculate total potential energy
Storage	Output
N/A	E_p

Draw energy bars

Input	Process
E_k, E_p	Draw the energy bars to the screen
Storage	Output
N/A	Coloured bars on the screen

Normalise angle

Input	Process
θ	Clamps angle between $[\pi, -\pi]$
Storage	Output
N/A	θ

Find circle intersections

Input	Process
$x_1, y_1, r_1, x_2, y_2, r_2$	Calculate points where two circles intersect
Storage	Output
N/A	x_3, y_3

Set pendulum position

Input	Process
$\theta_1, \theta_2, \omega_1, \omega_2$	Set the pendulum to these values
Storage	Output
Change pendulum variables	N/A

Get preset

Input	Process
N/A	Get all the presets from the database
Storage	Output
Access database and retrieve values	Values from database

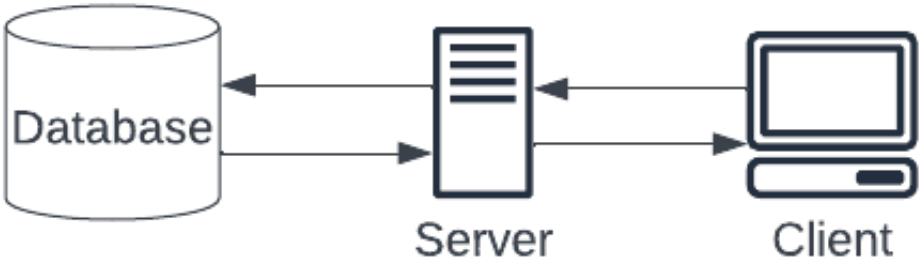
Insert preset

Input	Process
Preset	Put the preset into the database
Storage	Output
Access database and put values	N/A

1.11 Data flow diagram

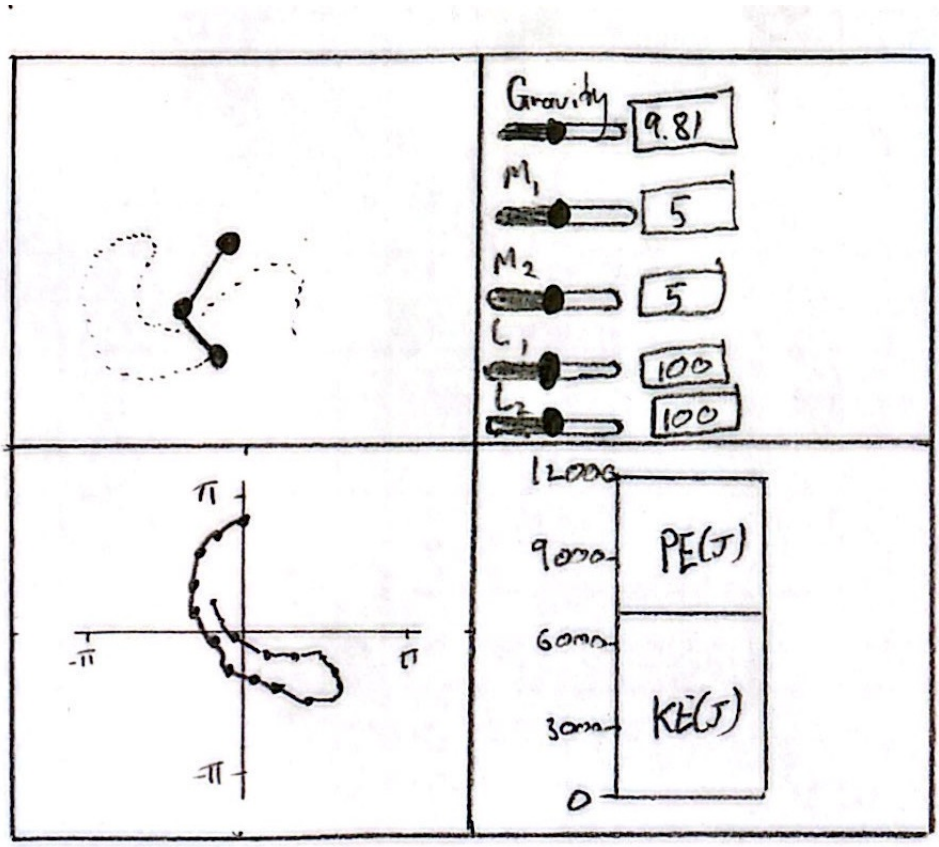
When the client wants the page, the server will get the presets from the database, then send the data with the page to the client. Then when the client wants to enter a preset into the database, they send it to the server which then puts in on the database, then the server resends the data and page back to the client to ensure the client's page is completely up to date.

UML diagram of architecture:



1.12 GUI

Here is roughly what the GUI should look like when the final this is done. The top left quadrant is the pendulum itself. Then the top right quadrant it the control section where you can change all the variables of the simulation. Then the bottom left quadrant is the graph where everything will be displayed, and the bottom right quadrant is the energy bars.



1.13 Users and Limitations

There would be numerous users of the system (all students and teachers in the physics department) with Mr Marlow being the primary user and using it to demonstrate in front of the entire class. All the teachers and students are computer literate to a level where the complexity of the system won't be a limiting factor. It is still important that the students and teachers spend as little time troubleshooting issues and learning how the system works, it must be adequately simple and intuitive to use so it doesn't waste time in lessons.

The limitations:

- Hardware - The computers in the physics department have significant processing limitations and therefore the system should be sufficiently efficient that it runs smoothly on these devices. One way to solve this would be to offload simulation calculations onto a server.
- Software - The IT department has put significant limitations to which applications can be installed on the school PCs, therefore an installation would likely not work. Having everything running on a website would not require any installations and therefore make it more accessible to everyone using it.
- Time - The project needs to be finished by March as those are my schools's internal deadlines.
- My skills - the problem can't be too complex that it becomes impossible for me to solve within the allotted time.

1.14 Objectives

1. General

1.1. The program must run as a website.

1.2. It must have a GUI

1.2.1. It must be completely controllable by a mouse

1.2.2. You should be able to fine tune values by typing them in.

1.2.3. You should be able to get the data of the pendulum.

1.3. The website must be able to run on school computers.

1.3.1. It must not have any pop-ups.

1.3.2. It must not use any banned websites/data.

1.3.3. It must not contain any code that needs to be downloaded to the user's computer.

1.3.4. It must work on any browser engines which could include WebKit, Blink, and Gecko.

1.3.5. It must be efficient and run quickly on low-performance computers.

2. Graphs

2.1. There must be graphs that dynamically update while the program runs.

2.1.1. Choose any 2 variables to plot against each other.

2.1.2. Be able to wipe the graph.

2.1.3. Be able to pause and stop the recording of data.

2.1.4. Be able to click on a piece of data and it shows its values.

3. Energy bars

3.1. The values must dynamically update as the simulation runs

3.2. The calculated values must be correct and accurate to the simulation

3.3. It must be easy to read off the bars to see the values of the kinetic energy.

4. Double Pendulum problem

4.1. It must use the correct equations.

4.2. It must run accurately.

4.3. User must be able to dynamically change the simulation as it runs.

4.3.1. Change gravity.

4.3.2. Independently change the length of each rod.

4.3.3. Independently change the mass of each part.

4.3.4. Drag the pendulum and move it to any starting position you want.

4.3.5. Change the numerical method used to solve it.

4.3.6. Change the integration step height

4.4. Have options to view each of the modes of the double pendulum

4.4.1. Simple oscillation (like the single pendulum).

4.4.2. Alternating oscillation (top to the left, bottom to right, mean position stays constant).

4.4.3. Chaotic motion without enough energy to flip over.

4.4.4. Chaotic motion with enough energy to flip over.

4.5. Be able to add their own presets to the simulation

4.6. Be able to load any of the added presets

2 Design

2.1 Language

I am most familiar with Python and have been programming with it the most throughout my life. But I am not able to use it as I need the program to run efficiently on websites. I will therefore definitely need HTML, CSS, and JavaScript. Simulations will be written in JavaScript, web-pages will be built in HTML, and CSS to make the front-end nice. When I initially started coding some prototypes, they were in Python due to me being familiar with them but then I quickly transitioned to using JavaScript for all sims.

2.2 File structure

```
.
|-- app.js
|-- public
|   |-- css
|   |   |-- style.css
|   |-- js
|       |-- graph.js
|       |-- main.js
|       |-- pendulum.js
|-- routes
|   |-- index.js
|-- views
    |-- index.pug
```

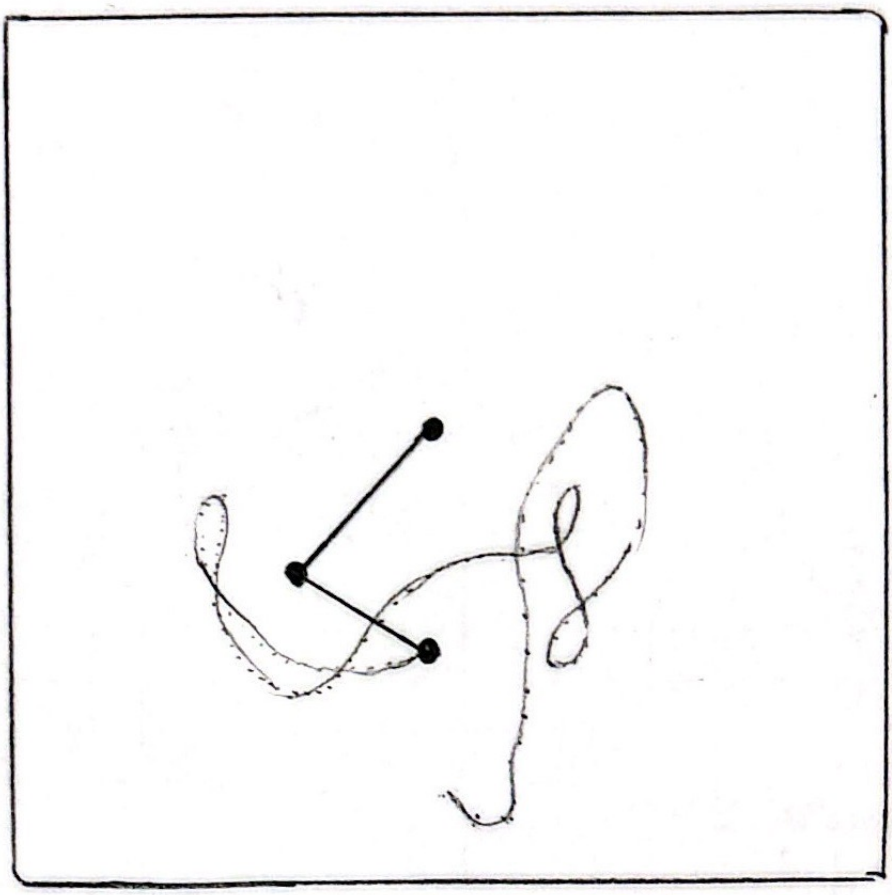
2.3 Libraries

- **express:**
  - Web framework for Node.js
  - Makes server-side logic
  - Manages HTTP requests and responses
- **routing:**
  - Importing from routes.js
  - routes handles the logic of the HTTP requests and responses
  - separated for more clarity and makes code easier to maintain.
- **path:**
  - Utilities for working with file and directory paths
  - Makes file handling it platform-independent
  - Makes sure file paths are correctly constructed and interpreted
- **url:**
  - Utilities for parsing and resolving URL strings.
  - 'fileURLToPath' used to convert URL of request to file path to specify files.
- **mongodb:**
  - Client for interactions with MongoDB database
  - MongoDB is NoSQL database
  - Used to insert and get presets from the database
  - Keep database online so that web app can be hosted on any server.
- **readline:**
  - Interface for reading inputs from the terminal line
  - Used to input the password for the MongoDB to securily connect to it
  - Means you don't need to hardcode the password which would be insecure.

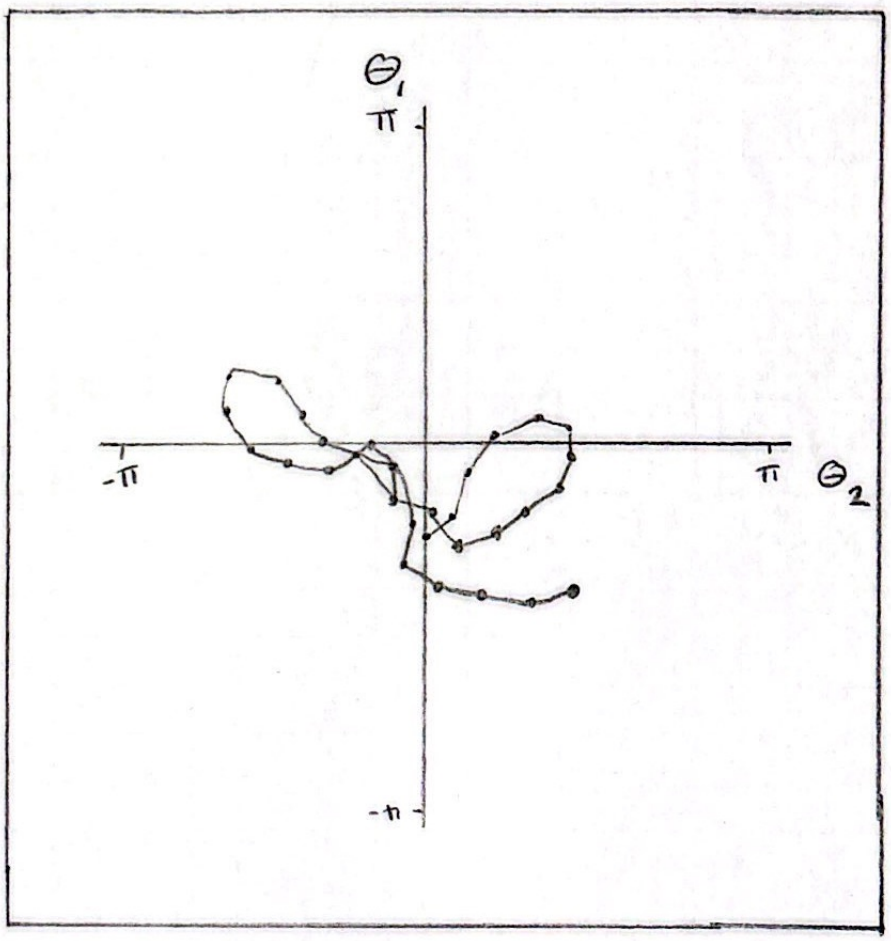
2.4 GUI

I drew different parts of the GUI on paper and then scanned them and put them into the document. These are what all of the components will look like and overall shows where they will all be placed and how they look together.

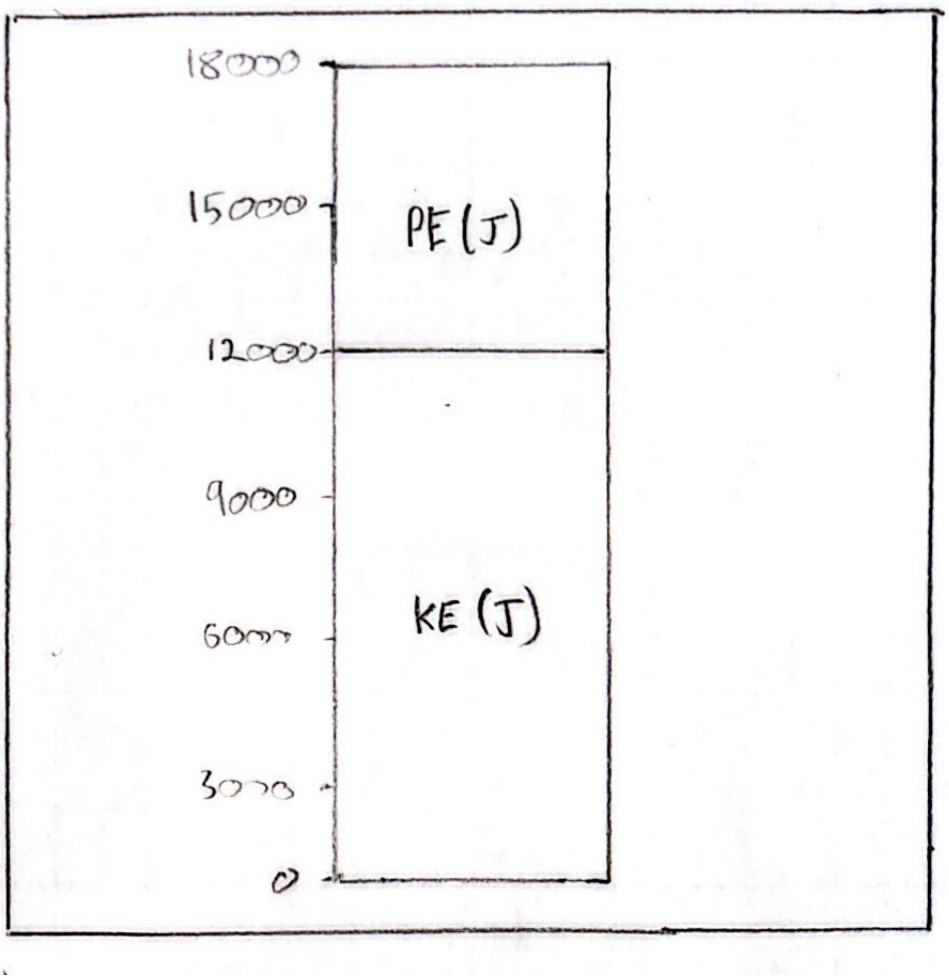
2.4.1 Pendulum



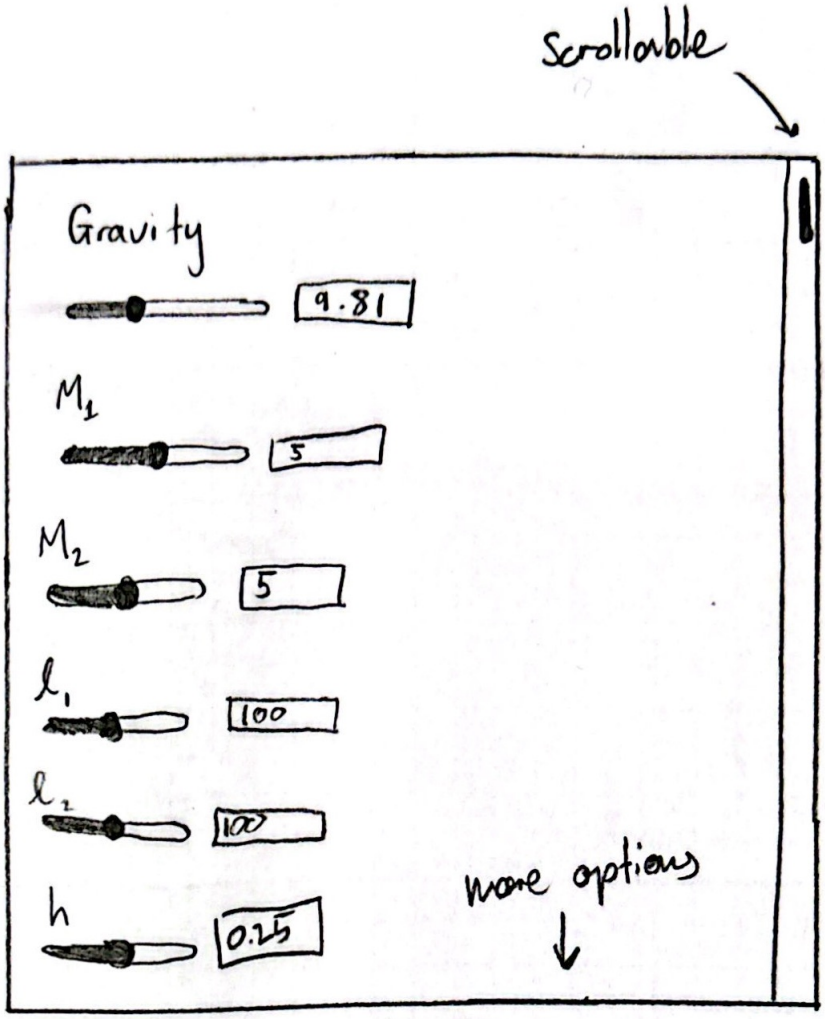
2.4.2 Graph



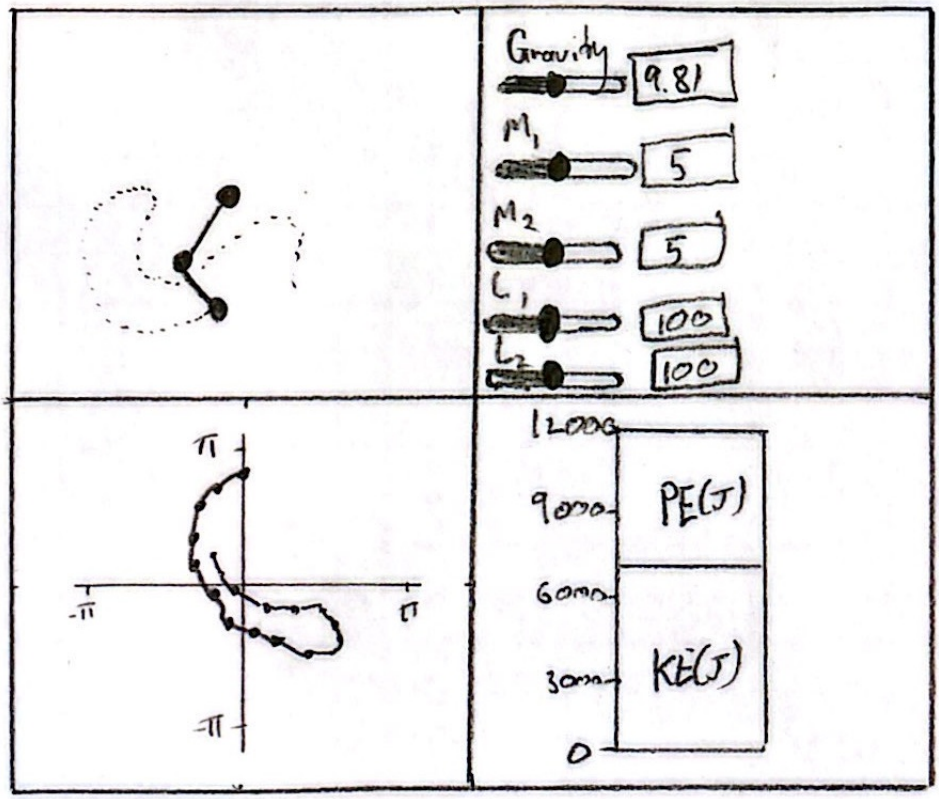
2.4.3 Energy bars



2.4.4 Controls



2.4.5 Overall


Gravity
9.81
M1
5
M2
5
L1
100
L2
100
π
-π
π
-π
12000
9000
6000
3000
0
PE(J)
KE(J)

2.5 Navigation

Pendulum

When just looking at it, it will animate and move around with a trace of the path of where it's been. Then if you click on the pendulum somewhere, it will put the pendulum to that position with zero angular velocity and angular acceleration.

Graph

There will be a graph that displays different variables against each other while the pendulum runs. In the control section you will be able to choose which variables are put on which axes. When the simulation is paused, you will be able to click on the graph and it will point out the closest point that you clicked on.

Energy bars

Will display the potential energy and kinetic energy of the simulation. When a variable is changed in someway, the maximum energy will update to reflect the new maximum energy.

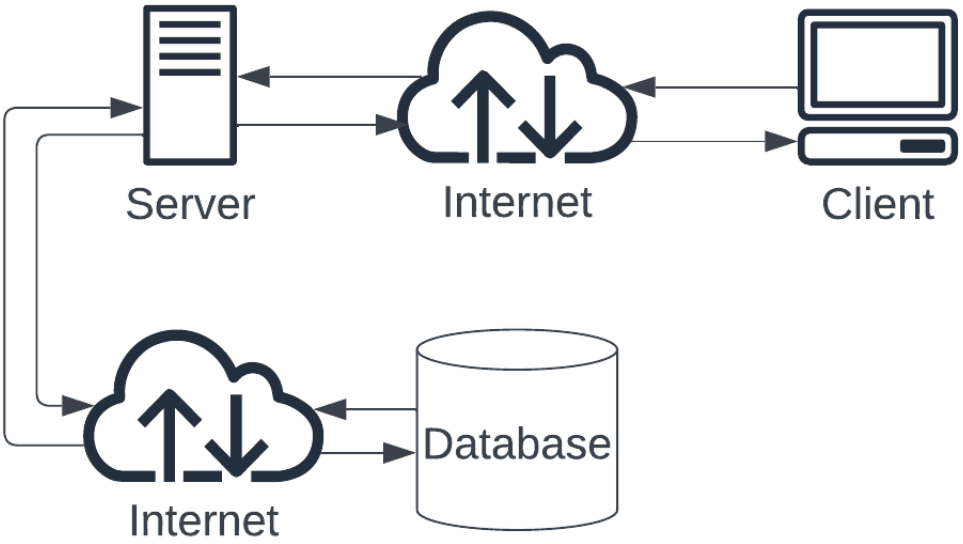
Controls

This is where you control the simulation. There are sliders that will change the variables as the simulation runs. Underneath that you will be able to add the presets to the database. Then, you can select a preset from the database of presets. Next, you can choose what 2 variables will be represented on the graph on each axis (time is only on the x-axis). You can also change what integration method is used to update the pendulum, choosing between RK4 and Euler. Then there are 3 buttons, one to pause the simulation, one to reset just the graph, and one to reset the entire simulation to how it starts when you first load the website.

2.6 Client-Server-Database architecture

When the client connects, it sends a get request to the server. The server then receives this request. The server sends a get request to the online database to get all of the saved presets. The database sends these presets to the server. The server then sends the HTML, CSS, JS, and database files all to the client. The client's web-browser then receives these and interprets them and displays them on the display.
When the client sends a preset to the server it sends a POST request to the server. The server then sends the data to be updated on the database. Then the database sends all the data back, which the server forwards with a new copy of all of the code. i.e. a POST-GET request.

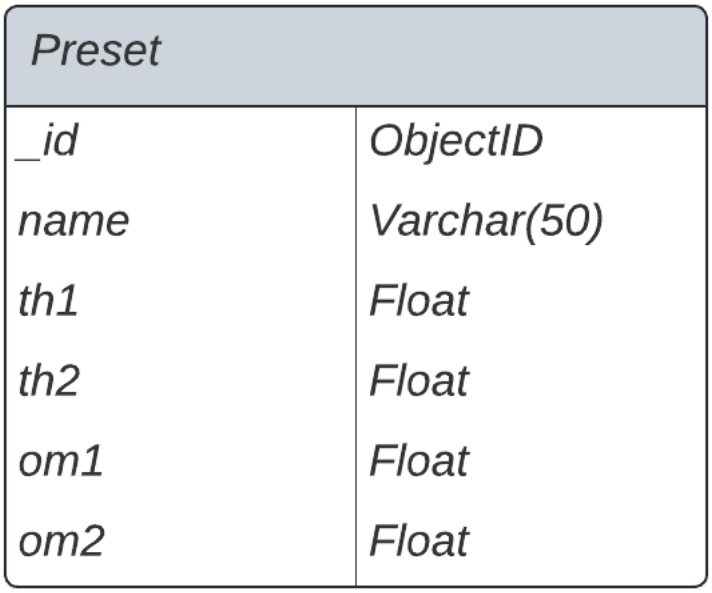
UML diagram of architecture:



2.7 Permanent data storage

Data will be securely stored on the database which will contain the data about presets. There won't be any data/cookies stored on the user's computer as that would overcomplicate things.

UML diagram of data structure:



2.8 Data structures

Item	Data type	Validation	Sample
θ_1	float	In range $[\pi, -\pi]$	1.42
θ_2	float	In range $[\pi, -\pi]$	-1.50
ω_1	float	In range $[\pi, -\pi]$	3.04
ω_2	float	In range $[\pi, -\pi]$	-2.89
m_1	float	In range $[2, 20]$	10.0
m_2	float	In range $[2, 20]$	10.0
l_1	float	In range $[5, 500]$	100.0
l_2	float	In range $[5, 500]$	100.0
g	float	In range $[0, 30]$	9.81
h	float	In range $[0.01, 1]$	0.25
width	integer	N/A	400
height	integer	N/A	400
xData	array	len <= 200	[3, 2.9, 2.8, ...]
yData	array	len <= 200	[1.0, 1.2, 1.4, ...]
historyTh1	array	len <= 200	[3, 2.9, 2.8, ...]
historyTh2	array	len <= 200	[3, 2.9, 2.8, ...]
historyOm1	array	len <= 200	[3, 2.9, 2.8, ...]
historyOm2	array	len <= 200	[3, 2.9, 2.8, ...]
historyTime	array	len <= 200	[3, 2.9, 2.8, ...]
x_0	integer	0 <= x_0 <= width	103
y_0	integer	0 <= y_0 <= height	205

2.9 Pseudocode algorithms

2.9.1 Differential

This is the differential equation of the pendulum that needs to be solved, it is a function that returns the $\omega_1, \omega_2, \dot{\omega}_1, \dot{\omega}_2$ given $\theta_1, \theta_2, \omega_1, \omega_2$. The equations are: (22), (23), (24), (25), (30), and (31)

procedure DYDX($\theta_1, \theta_2, \omega_1, \omega_2, g, m_1, m_2, l_1, l_2$)

$a_1 = (l_2/l_1) * (m_2/(m_1 + m_2)) * \cos(\theta_1 - \theta_2)$

$a_2 = (l_1/l_2) * \cos(\theta_1 - \theta_2)$

$f_1 = -(l_2/l_1) * (m_2/(m_1 + m_2)) * (\omega_2 * *2) * \sin(\theta_1 - \theta_2) - (g/l_1) * \sin(\theta_1)$

$f_2 = (l_1/l_2) * (\omega_1 * *2) * \sin(\theta_1 - \theta_2) - (g/l_2) * \sin(\theta_2)$

return $\omega_1, \omega_2, \dot{\omega}_1, \dot{\omega}_2$

end procedure

2.9.2 RK4

This is an algorithm that inputs a step height - h and current values to calculate the next value of the function with differential of dydx. This is used to numerically integrate the pendulum differential. The equations are: (35 - 40)

procedure RK4($h, \theta_1, \theta_2, \omega_1, \omega_2$)

$k_1 = dydx(\theta_1, \theta_2, \omega_1, \omega_2)$

$k_2 = dydx(\theta_1 + 0.5 * h * k_1[0], \theta_2 + 0.5 * h * k_1[1], \omega_1 + 0.5 * h * k_1[2], \omega_2 + 0.5 * h * k_1[3])$

$k_3 = dydx(\theta_1 + 0.5 * h * k_2[0], \theta_2 + 0.5 * h * k_2[1], \omega_1 + 0.5 * h * k_2[2], \omega_2 + 0.5 * h * k_2[3])$

$k_4 = dydx(\theta_1 + h * k_3[0], \theta_2 + h * k_3[1], \omega_1 + h * k_3[2], \omega_2 + h * k_3[3])$

$\theta_1 = \theta_1 + (h/6) * (k_1[0] + 2 * k_2[0] + 2 * k_3[0] + k_4[0])$

$\theta_2 = \theta_2 + (h/6) * (k_1[1] + 2 * k_2[1] + 2 * k_3[1] + k_4[1])$

$\omega_1 = \omega_1 + (h/6) * (k_1[2] + 2 * k_2[2] + 2 * k_3[2] + k_4[2])$

$\omega_2 = \omega_2 + (h/6) * (k_1[3] + 2 * k_2[3] + 2 * k_3[3] + k_4[3])$

return $\theta_1, \theta_2, \omega_1, \omega_2$

end procedure

2.9.3 Euler method

This is the algorithm for the Euler integration method. Another integration method that will be used to integrate the pendulum differential. The equation is: (34)

procedure EULERMETHOD($h, \theta_1, \theta_2, \omega_1, \omega_2$)

$\omega_1, \omega_2, \dot{\omega}_1, \dot{\omega}_2 = \text{dydx}(\theta_1, \theta_2, \omega_1, \omega_2)$

$\theta_1 = \theta_1 + h\omega_1$

$\theta_2 = \theta_2 + h\omega_2$

$\omega_1 = \omega_1 + h\dot{\omega}_1$

$\omega_2 = \omega_2 + h\dot{\omega}_2$

return $\theta_1, \theta_2, \omega_1, \omega_2$

end procedure

2.9.4 Draw pendulum

This draws the pendulum to the screen.

procedure DRAWPENDULUM(θ_1, θ_2, m_1, m_2, l_1, l_2, width, height)

clearScreen()

colour = black

x_0 = width / 2

y_0 = width/2

$x_1 = x_0 + l_1 \sin\theta_1$

$y_1 = y_0 + l_1 \cos\theta_1$

$x_2 = x_1 + l_2 \sin\theta_2$

$y_2 = y_1 + l_2 \cos\theta_2$

circle(x_1, y_1, m_1)

circle(x_2, y_2, m_2)

line((x_0, y_0), (x_1, y_1))

line((x_1, y_1), (x_2, y_2))

end procedure

2.9.5 Draw graph

This will draw the graph to the screen with all of the points.

procedure DRAWGRAPH(xData, yData, width, height)

clearScreen

yScale = height / (2*π)

xScale = width / (2*π)

colour = black

line((0, height/2), (width, height/2))

line((width/2, 0), (width/2, height))

for i **do** in 1 to len(xData)

line((xData[i-1], yData[i-1]), (xData[i], yData[i]))

end for

end procedure

2.9.6 Nearest Neighbour algorithm

This is the algorithm used to get the closest point to where you click so that it can be displayed on the graph. This is from equation (41).

procedure KNN(x_0, y_0, xData, yData)

x_{rad} = -(x_0/xScale - π)

y_{rad} = -(y_0/yScale - π)

distances = []

for i **do** in 1 to len(xData)

$d = \sqrt{(x_{rad} - x_0)^2 + (y_{rad} - y_0)^2}$

distances.append((xData, yData,d))

end for

sort distances by d

return distances[0]

end procedure

2.9.7 Calculate Kinetic energy

This algorithm calculates the kinetic energy of the pendulum to display on the energy bars. This comes from equation (47).

procedure KINETICENERGY($\omega_1, \omega_2, m_1, m_2, l_1, l_2$)

$E_k = \frac{1}{2}m_1\omega_1^2 l_1^2 + \frac{1}{2}m_2\omega_2^2 l_2^2$

return E_k

end procedure

2.9.8 Calculate Potential energy

This algorithm calculates the potential energy of the pendulum to display on the energy bars. This comes from equations: (51 - 54)

procedure POTENTIALENERGY($\theta_1, \theta_2, g, m_1, m_2, l_1, l_2$)

$h_1 = l_1 + l_2 - l_1 \cos\theta_1$

$h_2 = h_1 - l_2 \cos\theta_2$

$E_p = m_1 g h_1 + m_2 g h_2$

return E_p

end procedure

2.9.9 Normalise angle

Used when adding an angle to the angle histories to make sure it can be displayed on the graph properly for if the pendulum flips over itself.

procedure NORMALISEANGLE(θ)

while $\theta > \pi$ **do**

$\theta = \theta - 2\pi$

end while

while $\theta < -\pi$ **do**

$\theta = \theta + 2\pi$

end while

return θ

end procedure

2.9.10 Find circle intersections

This algorithm returns where the first pendulum can be put when you click on the pendulum to set its position. The equations comes from (55 - 68).

procedure FINDCIRCLEINTERSECTIONS($x_1, y_1, r_1, x_2, y_2, r_2$)

$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

if ($d >= r_1 + r_2$) or ($d <= |r_1 - r_2|$)) **then**

return False

end if

$\theta = \arctan \frac{y_2 - y_1}{x_2 - x_1}$

$a = \frac{r_1^2 - r_2^2 + d^2}{2d}$

$h = \sqrt{r_1^2 - a^2}$

$x_3 = x_1 + a \cos \theta$

$y_3 = y_1 + a \sin \theta$

$x_4 = x_3 + h \cos \left(\theta + \frac{\pi}{2}\right)$

$y_4 = y_3 + h \sin \left(\theta + \frac{\pi}{2}\right)$

return x_4, y_4

end procedure

2.9.11 Main run loop

This should be in the main loop and will run every frame to update the pendulum.

procedure RUN

updateVariables()

drawPendulum()

updatePendulum()

updateHistories()

drawEnergyBar()

drawGraph()

end procedure

2.10 Prototype

I have made a quick prototype of the core simulation of the n-body problem and the double-pendulum problem to demonstrate the core movement of the objects within it. I initially wrote in Python as it was easier for me to make a prototype, but the project itself will be in HTML, CSS, and JavaScript.

```
from scipy import constants
import math
import pygame

g = constants.G
pi = constants.pi
white = (255,255,255)
black = (0,0,0)
width = 800
height = 400 #screen size
h = 0.2 #step size
stepsPerFrame = 1
timeScale = 100 #speed of simulation

m1 = 5;   #mass first bob
m2 = 5;   #mass second bob
l1 = 100 #length first rod
l2 = 100 #length second rod
th1 = 90 * (pi / 180) #angle first
th2 = 30 * (pi / 180) #angle second
om1 = 0  #angular velocity first
om2 = 0  #angular velocity second

x0 = width/2 #base points
y0 = height/5

#clamps angles for propper size
def normalizeAngle(angle):
    angle %= (2*pi)
    if angle > pi:
        angle -= 2*pi
    elif angle < pi:
        angle += 2*pi
    return angle

#differential equation
def differential(th1, th2, om1, om2):
    delta = th2-th1
    den1 = (m1 + m2) * l1 - m2 * l1 * math.cos(delta) * math.cos(
   delta)
```

```
    den2 = (l2 / l1) * den1
    dotom1 = ((m2 * l2 * om2 * om2 * math.sin(delta) * math.cos(
    delta)
        + m2 * g * math.sin(th2) * math.cos(delta)
        + m2 * l2 * om2 * om2 * math.sin(delta)
        - (m1 + m2) * g * math.sin(th1))
        / den1)
    dotom2 = ((-l1 / l2) * om1 * om1 * math.sin(delta) * math.cos(
    delta)
        + (m1 + m2) * g * math.sin(th1) * math.cos(delta)
        - (m1 + m2) * l1 * om1 * om1 * math.sin(delta)
        - (m1 + m2) * g * math.sin(th2))/ den2
    return [om1, om2, dotom1, dotom2]

#numerical estimation of next value
def rk4Integration(h, th1, th2, om1, om2):
    k1_th1, k1_th2, k1_om1, k1_om2 = differential(th1, th2, om1, om2
    )
    k2_th1, k2_th2, k2_om1, k2_om2 = differential(
        th1 + 0.5 * h * k1_th1,
        th2 + 0.5 * h * k1_th2,
        om1 + 0.5 * h * k1_om1,
        om2 + 0.5 * h * k1_om2)
    k3_th1, k3_th2, k3_om1, k3_om2 = differential(
        th1 + 0.5 * h * k2_th1,
        th2 + 0.5 * h * k2_th2,
        om1 + 0.5 * h * k2_om1,
        om2 + 0.5 * h * k2_om2)
    k4_th1, k4_th2, k4_om1, k4_om2 = differential(
        th1 + h * k3_th1,
        th2 + h * k3_th2,
        om1 + h * k3_om1,
        om2 + h * k3_om2)

    dth1 = (h / 6) * (k1_th1 + 2 * k2_th1 + 2 * k3_th1 + k4_th1)
    dth2 = (h / 6) * (k1_th2 + 2 * k2_th2 + 2 * k3_th2 + k4_th2)
    dom1 = (h / 6) * (k1_om1 + 2 * k2_om1 + 2 * k3_om1 + k4_om1)
    dom2 = (h / 6) * (k1_om2 + 2 * k2_om2 + 2 * k3_om2 + k4_om2)

    print(dth1, dth2, dom1, dom2)

    return dth1, dth2, dom1, dom2

#initialises pygame
pygame.init()
screen = pygame.display.set_mode((width,height)) #instantiates the
    display
clock = pygame.time.Clock()
```

```
frameRate = 60
run = True

lastTime = 0
while run:
    clock.tick(frameRate)
    screen.fill(white)

    #allows for more accurate running
    stepsForThisFrame = int(stepsPerFrame * timeScale)
    for x in range(stepsForThisFrame):
        (dth1, dth2, dom1, dom2) = rk4Integration(h, th1, th2, om1, om2)
        th1 += dth1
        th2 += dth2
        om1 += dom1
        om2 += dom2

    #calculates position of pendulum arms
    x1 = x0 + l1 * math.sin(normalizeAngle(th1))
    y1 = y0 + l1 * math.cos(normalizeAngle(th1))
    x2 = x1 + l2 * math.sin(normalizeAngle(th2))
    y2 = y1 + l2 * math.cos(normalizeAngle(th2))

    #draws to the screen
    pygame.draw.lines(screen, black, False, [(x0,y0),(x1,y1),(x2,y2)])

    for event in pygame.event.get():#stops simulation if you exit the screen
        if event.type == pygame.QUIT:
            run = False

    pygame.display.flip()
pygame.quit()
```

3 Technical Solution

3.1 Technical skills

Group	Model	Algorithm	Page number
A	Complex scientific model	Pendulum differential	76/77
		Kinetic energy calculations	80
		Potential energy calculations	80/81
		Real-time variable updates	73/74
A	Complex mathematical model	Numerical integration	77/78
		KNN algorithm	84/85
		Graph scaling	84/85/86
		Circle intersections	79/80
		Calculate bearings	80
A	Complex client-server model	Request and response scripting	69/70
		Request routing	68/70
		Parsing JSON	71/74/75
		Servicing JSON	69
A	List structures	List operations	76/83/84
		Storing graph values	83
A	Complex file storage	Data stored on online database	69
C	Single table database	Non-SQL table access	69

3.2 Coding styles

Style	Characteristic	Page number
Excellent	Modules with appropriate interfaces	77/79/80
	Cohesive modules	80
8	Good exception handling	79
	Modules	76-82/83-88
	Loosely coupled modules	80/81/85
Good	Well-designed user interface	64-67
	Good use of local variables	77/78
	Use of constants	68/79/80
	Appropriate indentation	63-89
	Self-documenting code	78-82
	File path parameterised	68
	Modularisation of code	76-81
	Consistent style throughout	68-88
Basic	Meaningful identifier names	68-88
	Annotations used effectively	68-88

3.3 Directory

```
.
|-- node_modules
|-- prototypes
|-- public
|   |-- css
|   |   |-- style.css
|   |-- js
|       |-- graph.js
|       |-- main.js
|       |-- pendulum.js
|-- tex
    |-- media
    |-- main.tex
|-- routes
|   |-- index.js
|-- views
    |-- index.pug
|-- app.js
|-- package-lock.json
|-- package.json
|-- README.md
|-- TODO.md
```

This is the entire file directory from my github repo where I stored and saved everything. "node-modules" contains the modules needed for node to properly run for the JavaScript. "prototypes" is the earlier versions of the code while I was still trying to get everything to work. "tex" is the folder containing all of the LaTeX and media to make the PDF for my project. "package-lock.json" and "package.json" are JSON files that tell node what dependencies to install before running. "README.md" is the github README file which outlines how to use the code. "TODO.md" is just a todo list so I can keep track of tasks.

3.4 Source code

3.4.1 index.pug

Pug is a JavaScript language that compiles into HTML. It makes it simpler and easier to write. Instead of using closing tags it uses indentation to close opening tags. This is the main web-page which is displayed. It contains a grid of all the different elements.

```
doctype html
head
    meta(charset='UTF-8')
    meta(name='viewport' content='width=device-width, initial-scale=1.0')
    title Double Pendulum Problem
    link(rel='stylesheet' href='/css/style.css')

.grid-container
    .grid-header
        h1 Double pendulum

    canvas.grid-pendulum#canvasPendulum(width='900px' height='900px')


    canvas.grid-energy#canvasEnergy(width='250px' height='500px')

    .grid-control
        h2 Gravity
        input#gravSlider(type='range' min='0' max='30' step='0.01' value='9.81' oninput='gravAmount.value=gravSlider.value')
        input#gravAmount(type='text' min='0' max='30' step='0.01' value='9.81' oninput='gravSlider.value=gravAmount.value')
        h2 m1
        input#m1Slider(type='range' min='2' max='20' step='0.01' value='5' oninput='m1Amount.value=m1Slider.value')
        input#m1Amount(type='text' min='2' max='20' step='0.01' value='5' oninput='m1Slider.value=m1Amount.value')
        h2 m2
        input#m2Slider(type='range' min='2' max='20' step='0.01' value='5' oninput='m2Amount.value=m2Slider.value')
        input#m2Amount(type='text' min='2' max='20' step='0.01' value='5' oninput='m2Slider.value=m2Amount.value')
        h2 l1
        input#l1Slider(type='range' min='5' max='500' step='0.01' value='100' oninput='l1Amount.value=l1Slider.value')
        input#l1Amount(type='text' min='5' max='500' step='0.01' value='100' oninput='l1Slider.value=l1Amount.value')
```

```
30          h2 l2
31          input#l2Slider(type='range' min='5' max='500' step='0.01'
    value='100' oninput='l2Amount.value=l2Slider.value')
32          input#l2Amount(type='text' min='5' max='500' step='0.01'
    value='100' oninput='l2Slider.value=l2Amount.value')
33          h2 h
34          input#hSlider(type='range' min='0.01' max='1' step='0.01'
    value='0.25' oninput='hAmount.value=hSlider.value')
35          input#hAmount(type='text' min='0.01' max='1' step='0.01'
    value='0.25' oninput='hSlider.value=hAmount.value')
36
37          div(class="addPreset")
38              h2 Add a preset
39              form(action="/addPreset", method="post")
40
41                  .formGroup
42                      label(for="name") Name:
43                      input(type="text", id="name", placeholder="name"
    , name="name")
44
45                  .formGroup
46                      label(for="th1") th1:
47                      input(type="text", id="th1", placeholder="th1",
    name="th1")
48
49                  .formGroup
50                      label(for="th2") th2:
51                      input(type="text", id="th2", placeholder="th2",
    name="th2")
52
53                  .formGroup
54                      label(for="om1") om1:
55                      input(type="text", id="om1", placeholder="om1",
    name="om1")
56
57                  .formGroup
58                      label(for="om2") om2:
59                      input(type="text", id="om2", placeholder="om2",
    name="om2")
60
61                  button(type="submit") Submit
62
63          .presetSelect
64              h2 Select a Preset
65              select(name="presetSelect" id="presetSelect")
66                  each preset in presets
67                      option(value=preset) #{preset.name}
68
```

```
69
70          h2 y-variable
71          select#yGraphSelect
72              option(value='graphHistoryTh1') Th1
73              option(value='graphHistoryTh2') Th2
74              option(value='graphHistoryOm1') Om1
75              option(value='graphHistoryOm2') Om2
76          h2 x-variable
77          select#xGraphSelect
78              option(value='graphHistoryTh2') Th2
79              option(value='graphHistoryTh1') Th1
80              option(value='graphHistoryOm1') Om1
81              option(value='graphHistoryOm2') Om2
82              option(value='graphHistoryTime') Time
83          h2 Integration method
84          select#integrationSelect
85              option(value='RK4') RK4
86              option(value='Euler') Euler
87          h2 Pause
88          input#pauseCheck(type='checkbox')
89          h2 Restart Graph
90          input#restartButton(type='button')
91          h2 Restart Variables and Graph
92          input#resetButton(type='button')
93
94
95      canvas.grid-graph#canvasGraph(width='600px' height='350px')
96
97
98  script(type='module' src='/js/graph.js')
99  script(type='module' src='/js/pendulum.js')
100 script(type='module' src='/js/main.js')
```

3.4.2 style.css

This is the style sheet for the web-page. It defines the sizes of all the grids and makes the controlling group scrollable so it is always on the page.

```
.formGroup{
    margin-bottom: 10px;
}
.grid-container {
    display: grid;
    column-gap: 20px;
    row-gap: 20px;
}
.grid-header {
    grid-column: 1 / 8;
    grid-row: 1 / span 1;
}
.grid-pendulum {
    grid-column: 1 / 5;
    grid-row: 2 / 7;
}
.grid-energy {
    grid-column: 4 / 6;
    grid-row: 2 / 5;
}
.grid-control {
    height: 500px;
    overflow: auto;
    border: 1px solid;
    grid-column: 6 / 8;
    grid-row: 2 / 5;
    padding-left: 10px;
    padding-bottom: 10px;
}
.grid-graph {
    grid-column: 4 / 8;
    grid-row: 5 / 7;
}
```

3.4.3 app.js

This is the entry point of the web page on the server-side. It compiles the pug page, sends GET and POST requests to index.js, and defines which port the web-page will run on.

```
1
2 import express from 'express';
3 import path from 'path';
4 import { fileURLToPath  } from 'url';
5 import { dirname } from 'path';
6 import indexRouter from './routes/index.js';
7
8 // Set dirname to configure requests
9 const __filename = fileURLToPath(import.meta.url);
10 const __dirname = dirname(__filename);
11
12 // Initialise parameters for app
13 const app = express();
14 app.use(express.urlencoded({ extended: true }));
15 app.use(express.json());
16 app.set('view engine', 'pug');
17 app.use(express.static(path.join(__dirname, '/public')));
18 app.use('/', indexRouter);
19
20 // Set the port used
21 const PORT = process.env.PORT || 3000;
22
23 // Confirm server is running
24 app.listen(PORT, function() {
25     console.log('Server is running on port '+ PORT);
26 });
```

3.4.4 index.js

This connects to the MongoDB and handles all of the POST and GET requests. GET requests just return the index.pug page with the presets loaded on top. POST requests add the presets to the database and send the user back to the page.

```
1
2 import express from 'express';
3 import { MongoClient } from 'mongodb';
4 import readline from 'readline';
5
6 // Interface for reading password
7 const read = readline.createInterface({
8     input: process.stdin,
9     output: process.stdout
10 });
11
12 // Reads from the terminal
13 async function query() {
14     return await new Promise(resolve => read.question("Password: ",
    resolve));
15 }
16
17 // Gets password
18 async function getPassword(){
19     try {
20         const answer = await query();
21         return answer;
22     } catch (error) {
23         console.log(`Error: ${error}`);
24     } finally {
25         read.close();
26     }
27 }
28
29 // Get the password
30 const password = await getPassword();
31
32 // Set up client with mongoDB database
33 const client = new MongoClient(`mongodb+srv://dbo:${password}
    @physteach.dl3xqy3.mongodb.net/?retryWrites=true&w=majority`)
34
35 // Set up router
36 const router = express.Router();
37
38 // Gets the presets from the database
39 async function GetPresets() {
```

```
40      await client.connect();
41      const dbo = client.db('presets');
42      return await dbo.collection('preset-collection').find({}).
    toArray();
43  }
44
45  // Adds presets to database
46  async function InsertPreset(name, th1, th2, om1, om2) {
47      await client.connect();
48      let dbo = client.db('presets');
49      await dbo.collection('preset-collection').insertOne({name, th1,
    th2, om1, om2});
50  }
51
52  // General get request handling
53  router.get('/', async function (req, res) {
54      let presets = await GetPresets();
55      res.render('index', {presets});
56  });
57
58  // Handles post requests for adding the presets
59  router.post('/addPreset', function (req, res){
60      (async function () {
61          await InsertPreset(req.body.name, req.body.th1, req.body.th2
    , req.body.om1, req.body.om2);
62          res.redirect('/');
63      })();
64  });
65
66  // Export router to app.js
67  export default router;
```

3.4.5 main.js

The main code that runs on the pendulum on the client side. This is sent with the html and other JavaScript program. It holds all the variables and the functions that control the variables. It also initialises the canvases and it has the main running loop calling functions from Pendulum.js and Graph.js.

```
1
2 // Import functions from other files
3 import {drawGraph, resetHistories, updateHistories, drawClosestPoint
    } from "./graph.js";
4 import {drawPendulum, updatePendulum, setPendulumPosition,
    drawEnergyBar, updatePosHistories, getMaxEnergy,
    resetPendHistories} from "./pendulum.js";
5
6 // Declare variables for exporting
7 let g, m1, m2, l1, l2;
8
9 function start(){
10     //Initialise inputs
11     const gravSlider = document.getElementById('gravSlider');
12     const m1Slider = document.getElementById('m1Slider');
13     const m2Slider = document.getElementById('m2Slider');
14     const l1Slider = document.getElementById('l1Slider');
15     const l2Slider = document.getElementById('l2Slider');
16     const hSlider = document.getElementById('hSlider');
17     const gravAmount = document.getElementById('gravAmount');
18     const m1Amount = document.getElementById('m1Amount');
19     const m2Amount = document.getElementById('m2Amount');
20     const l1Amount = document.getElementById('l1Amount');
21     const l2Amount = document.getElementById('l2Amount');
22     const hAmount = document.getElementById('hAmount');
23     const pauseCheck = document.getElementById('pauseCheck');
24     const restartButton = document.getElementById('restartButton')
25     const resetButton = document.getElementById('resetButton');
26     const yGraphSelect = document.getElementById('yGraphSelect');
27     const xGraphSelect = document.getElementById('xGraphSelect');
28     const integrationSelect = document.getElementById('
    integrationSelect');
29     const presetSelect = document.getElementById('presetSelect');
30
31     //Run reset scrips when these buttons are clicked on the html
32     restartButton.addEventListener('click', RestartGraph);
33     resetButton.addEventListener('click', RestartVariablesAndGraph);
34
35     presetSelect.addEventListener("change", function(e){
36         const preset = JSON.parse(e.target.value);
```

```
37          setAngleParams(preset.th1, preset.th2, preset.om1, preset.
    om2);
38          maxLabel = getMaxEnergy(th1, th2, om1, om2);
39      });
40
41      //Initialise starting parameters
42      g = parseFloat(gravSlider.value);
43      m1 = parseFloat(m1Slider.value);
44      m2 = parseFloat(m2Slider.value);
45      l1 = parseFloat(l1Slider.value);
46      l2 = parseFloat(l2Slider.value);
47      let h = parseFloat(hSlider.value);
48      let th1 = 90 * (Math.PI / 180);
49      let th2 = 30 * (Math.PI / 180);
50      let om1 = 0;
51      let om2 = 0;
52      let maxLabel = getMaxEnergy(th1, th2, om1, om2);
53
54      //Changes maxLabel when a slider is changed
55      gravSlider.addEventListener('change', function(e){
56          maxLabel = getMaxEnergy(th1, th2, om1, om2);
57      });
58      m1Slider.addEventListener('change', function(e){
59          maxLabel = getMaxEnergy(th1, th2, om1, om2);
60      });
61      m2Slider.addEventListener('change', function(e){
62          maxLabel = getMaxEnergy(th1, th2, om1, om2);
63      });
64      l1Slider.addEventListener('change', function(e){
65          maxLabel = getMaxEnergy(th1, th2, om1, om2);
66      });
67      l2Slider.addEventListener('change', function(e){
68          maxLabel = getMaxEnergy(th1, th2, om1, om2);
69      });
70      hSlider.addEventListener('change', function(e){
71          maxLabel = getMaxEnergy(th1, th2, om1, om2);
72      });
73
74
75      //Canvas setup for pendulum
76      const desiredFPS = 60;
77      let canvasPendulum = document.querySelector('#canvasPendulum');
78      let ctxPendulum = canvasPendulum.getContext('2d');
79      let widthPendulum = canvasPendulum.width;
80      let heightPendulum = canvasPendulum.height;
81
82      // Changes position of pendulum when you click on it
83      canvasPendulum.addEventListener("click", function(event){
```

```
84          [th1, th2, om1, om2] = setPendulumPosition(canvasPendulum,
    event, th1, th2, om1, om2);
85          maxLabel = getMaxEnergy(th1, th2, om1, om2);
86
87      })
88
89      //Canvas setup for graph
90      let canvasGraph = document.querySelector('#canvasGraph');
91      let ctxGraph = canvasGraph.getContext('2d');
92      ctxGraph.font = "16px Arial";
93      ctxGraph.textAlign = "center";
94      let widthGraph = canvasGraph.width;
95      let heightGraph = canvasGraph.height;
96
97      // Checks when you click on the graph and does a function
98      canvasGraph.addEventListener("click", function(event){
99          if (pauseCheck.checked == true){
100             drawClosestPoint(canvasGraph, event, xGraph, yGraph,
    widthGraph, heightGraph);
101
102         }
103     });
104
105     //Axes selection variables
106     let yGraph = yGraphSelect.value;
107     let xGraph = xGraphSelect.value;
108
109     // Canvas setup for energy bar
110     let canvasEnergy = document.querySelector('#canvasEnergy');
111     let ctxEnergy = canvasEnergy.getContext('2d');
112     ctxEnergy.font = "16px Arial";
113     ctxEnergy.textAlign = "center";
114     let widthEnergy = canvasEnergy.width;
115     let heightEnergy = canvasEnergy.height;
116
117     //Reset and restart functions
118     function RestartGraph(){
119         th1 = 90 * (Math.PI / 180);
120         th2 = 30 * (Math.PI / 180);
121         om1 = 0;
122         om2 = 0;
123         maxLabel = getMaxEnergy(th1, th2, om1, om2);
124         resetHistories()
125         resetPendHistories()
126         }
127
128     function RestartVariablesAndGraph(){
129         g = 9.81;
```

```
130         gravSlider.value = 9.81;
131         gravAmount.value = 9.81;
132         h = 0.25;
133         hSlider.value = 0.25;
134         hAmount.value = 0.25;
135         m1 = 5;
136         m1Slider.value = 5;
137         m1Amount.value = 5;
138         m2 = 5;
139         m2Slider.value = 5;
140         m2Amount.value = 5;
141         l1 = 100;
142         l1Slider.value = 100;
143         l1Amount.value = 100;
144         l2 = 100;
145         l2Slider.value = 100;
146         l2Amount.value = 100;
147         RestartGraph()
148         }
149
150     // Updates variables to sliders on the page
151     function updateVariables(){
152         g = gravSlider.value;
153         m1 = m1Slider.value;
154         m2 = m2Slider.value;
155         l1 = l1Slider.value;
156         l2 = l2Slider.value;
157         h = hSlider.value;
158         yGraph = yGraphSelect.value;
159         xGraph = xGraphSelect.value;
160         widthGraph = canvasGraph.width;
161         heightGraph = canvasGraph.height;
162         widthPendulum = canvasPendulum.width;
163         heightPendulum = canvasPendulum.height;
164     }
165
166     // Updates angles
167     function setAngleParams(newTh1, newTh2, newOm1, newOm2){
168         th1 = parseFloat(newTh1) * (Math.PI / 180);
169         th2 = parseFloat(newTh2) * (Math.PI / 180);
170         om1 = parseFloat(newOm1);
171         om2 = parseFloat(newOm2);
172     };
173
174     // Variables for running
175     const timePerFrame = 1000/desiredFPS;
176     let totalFrames = 0;
177     let previousTime = 0;
```

```
178
179     function run(currentTime){
180
181         requestAnimationFrame(run);
182
183         const deltaTime = currentTime - previousTime;
184
185         if (deltaTime >= timePerFrame) {
186
187             previousTime = currentTime;
188
189             updateVariables();
190
191             let [x, y] = drawPendulum(ctxPendulum, th1, th2,
    widthPendulum, heightPendulum);
192
193             if (pauseCheck.checked == false){
194
195                 [th1, th2, om1, om2] = updatePendulum(
    integrationSelect.value, h, th1, th2, om1, om2);
196
197                 totalFrames++;
198
199                 updateHistories(th1, th2, om1, om2, totalFrames);
200
201                 updatePosHistories(x, y);
202
203             }
204
205             drawEnergyBar(ctxEnergy, widthEnergy, heightEnergy, th1,
     th2, om1, om2, maxLabel);
206
207             drawGraph(ctxGraph, xGraph, yGraph, widthGraph,
    heightGraph, pauseCheck);
208
209         }
210     }
211
212     requestAnimationFrame(run);
213
214 };
215
216 // Only start running the file when teh DOM has finished loading.
217 document.addEventListener('DOMContentLoaded', start);
218
219 // Get and set pendulum parameters
220 export function getParams() {
221     g = parseFloat(g);
```

```
222     m1 = parseFloat(m1);
223     m2 = parseFloat(m2);
224     l1 = parseFloat(l1);
225     l2 = parseFloat(l2);
226     return { g, m1, m2, l1, l2 };
227 };
228 
229 // Set the parameters within main.js
230 export function setParams({ newG, newM1, newM2, newL1, newL2 }) {
231     g = newG;
232     m1 = newM1;
233     m2 = newM2;
234     l1 = newL1;
235     l2 = newL2;
236 };
```

3.4.6 pendulum.js

This has all the code for the pendulum canvas and the energy bar canvas. It also stores the histories of x and y positions to draw the path that the pendulum has gone. It has the differential equation, numerical integration, energy calculation, and drawing functions.

```
1
2 import { getParams } from "./main.js";
3
4 // Constants for pendulum
5 const stepsPerFrame = 10;
6 const timeScale = 100;
7 const maxPosPoints = 200;
8
9 // Constants for energy bars
10 const Xpadding = 75;
11 const Ypadding = 10;
12 const numYPoints = 5;
13
14 // Pendulum position histories
15 let xHistories = [];
16 let yHistories = [];
17
18 export function resetPendHistories(){
19     xHistories = [];
20     yHistories = [];
21 }
22
23 // Update histories of positions
24 export function updatePosHistories(x,y){
25     xHistories.push(x);
26     yHistories.push(y);
27
28     // Limit histoy to maxPosPoints
29     while (xHistories.length > maxPosPoints){
30         xHistories.shift();
31         yHistories.shift();
32     }
33 }
34
35 function differential(th1, th2, om1, om2){
36
37     const {g, m1, m2, l1, l2} = getParams()
38
39     function a1(th1, th2){
40         return (l2/l1)*(m2/(m1+m2))*Math.cos(th1-th2);
```

```
41      }
42      function a2(th1, th2){
43          return (l1/l2)*Math.cos(th1-th2);
44      }
45      function f1(th1, th2, om1, om2){
46          return -(l2/l1)*(m2/(m1+m2))*(om2**2)*Math.sin(th1-th2) - (g
    /l1)*Math.sin(th1);
47      }
48      function f2(th1, th2, om1, om2){
49          return (l1/l2)*(om1**2)*Math.sin(th1-th2) - (g/l2)*Math.sin(
    th2);
50      }
51
52      let dotom1 = (f1(th1, th2, om1, om2) - a1(th1, th2)*f2(th1, th2,
     om1, om2))/(1-a1(th1,th2)*a2(th1,th2));
53      let dotom2 = (f2(th1, th2, om1, om2) - a2(th1, th2)*f1(th1, th2,
     om1, om2))/(1-a1(th1,th2)*a2(th1,th2));
54
55      return [om1, om2, dotom1, dotom2];
56
57  }
58
59  //Integration method
60  function Integration(method, h, th1, th2, om1, om2) {
61
62      if (method === 'RK4'){
63          let [k1_th1, k1_th2, k1_om1, k1_om2] = differential(th1, th2
    , om1, om2);
64          let [k2_th1, k2_th2, k2_om1, k2_om2] = differential(
65              th1 + 0.5 * h * k1_th1,
66              th2 + 0.5 * h * k1_th2,
67              om1 + 0.5 * h * k1_om1,
68              om2 + 0.5 * h * k1_om2
69          );
70          let [k3_th1, k3_th2, k3_om1, k3_om2] = differential(
71              th1 + 0.5 * h * k2_th1,
72              th2 + 0.5 * h * k2_th2,
73              om1 + 0.5 * h * k2_om1,
74              om2 + 0.5 * h * k2_om2
75          );
76          let [k4_th1, k4_th2, k4_om1, k4_om2] = differential(
77              th1 + h * k3_th1,
78              th2 + h * k3_th2,
79              om1 + h * k3_om1,
80              om2 + h * k3_om2
81          );
82
```

```
83          th1 += (h / 6) * (k1_th1 + 2 * k2_th1 + 2 * k3_th1 + k4_th1)
    ;
84          th2 += (h / 6) * (k1_th2 + 2 * k2_th2 + 2 * k3_th2 + k4_th2)
    ;
85          om1 += (h / 6) * (k1_om1 + 2 * k2_om1 + 2 * k3_om1 + k4_om1)
    ;
86          om2 += (h / 6) * (k1_om2 + 2 * k2_om2 + 2 * k3_om2 + k4_om2)
    ;
87
88          return [th1, th2, om1, om2];
89
90
91      } else if (method === 'Euler'){
92          let [dth1, dth2, dom1, dom2] = differential(th1, th2, om1,
    om2);
93
94          th1 += h*dth1;
95          th2 += h*dth2;
96          om1 += h*dom1;
97          om2 += h*dom2;
98
99          return [th1, th2, om1, om2];
100     }
101 }
102
103 //Draws the pendulum to the canvas
104 export function drawPendulum(ctxPendulum, th1, th2, widthPendulum,
    heightPendulum) {
105
106     const { g, m1, m2, l1, l2 } = getParams();
107
108     ctxPendulum.clearRect(0,0, widthPendulum, heightPendulum);
109
110     //Calculate positions of cirlces
111     let x0 = widthPendulum /2;
112     let y0 = heightPendulum / 2;
113     let x1 = x0 + l1 * Math.sin(th1);
114     let y1 = y0 + l1 * Math.cos(th1);
115     let x2 = x1 + l2 * Math.sin(th2);
116     let y2 = y1 + l2 * Math.cos(th2);
117
118     //Draw circles
119     ctxPendulum.beginPath();
120     ctxPendulum.arc(x1, y1, m1, 0, 2* Math.PI);
121     ctxPendulum.arc(x2, y2, m2, 0, 2 * Math.PI);
122     ctxPendulum.fillStyle = "#000";
123     ctxPendulum.fill();
124
```

```
125     //Draw lines connecting them
126     ctxPendulum.beginPath();
127     ctxPendulum.moveTo(x0, y0);
128     ctxPendulum.lineTo(x1, y1);
129     ctxPendulum.moveTo(x1, y1);
130     ctxPendulum.lineTo(x2, y2);
131     ctxPendulum.strokeStyle = "#000";
132     ctxPendulum.stroke();
133 
134     // Draw lines of the path
135     ctxPendulum.strokeStyle = "red";
136     ctxPendulum.beginPath();
137     ctxPendulum.moveTo(xHistories[0], yHistories[0]);
138     for (let i = 0; i <= xHistories.length; i++) {
139         ctxPendulum.lineTo(xHistories[i], yHistories[i]);
140         ctxPendulum.moveTo(xHistories[i], yHistories[i]);
141     }
142     ctxPendulum.stroke();
143     ctxPendulum.strokeStyle = "black";
144 
145     return [x2, y2];
146 }
147 
148 //Updates pendulum based on how long has passed
149 export function updatePendulum(method, h, th1, th2, om1, om2){
150 
151     [th1, th2, om1, om2] = Integration(method, h, th1, th2, om1, om2
    );
152 
153     return [th1, th2, om1, om2];
154 }
155 
156 // Returns the points of intersection of 2 circles to find middle
    point
157 function findCircleIntersections(x1,y1,r1, x2,y2,r2) {
158 
159     const d = Math.hypot(x2-x1, y2-y1);
160 
161     if ((d >= (r1 + r2)) || (d <= Math.abs(r1 - r2))) {
162         return false; // No intersection
163     }
164 
165     const theta = Math.atan2(y2 - y1, x2 - x1);
166     const a = (r1 * r1 - r2 * r2 + d * d) / (2 * d);
167     const h = Math.sqrt(r1 * r1 - a * a);
168 
169     const x3 = x1 + a * Math.cos(theta);
170     const y3 = y1 + a * Math.sin(theta);
```

```
171
172     const x4 = x3 + h * Math.cos(theta + Math.PI / 2);
173     const y4 = y3 + h * Math.sin(theta + Math.PI / 2);
174
175     // Arbitrarily returns to positive value and doesn't calculate
    other intersection
176     // As it is not needed
177     return [x4, y4];
178   }
179
180 export function setPendulumPosition(canvas, event, th1, th2, om1,
    om2){
181     const { g, m1, m2, l1, l2 } = getParams();
182
183     // Calculates the x and y position of where has been clicked
184     const xClick = event.offsetY;
185     const yClick = event.offsetX;
186
187     const origX = canvas.width/2;
188     const origY = canvas.height/2;
189
190     const point = findCircleIntersections(origX, origY, l1, xClick,
    yClick, l2);
191
192     // Calculates th1 and th2 that results in these points being
    chosen
193     if (point != false){
194         th1 = Math.atan2(point[1]-origY, point[0]-origX);
195         th2 = Math.atan2(yClick-point[1], xClick-point[0]);
196         return [th1, th2, 0, 0];
197     } else {
198     return [th1,th2, om1, om2];
199     };
200 }
201
202 // Returns sum of kinetic energies of both pendulums
203 function getKineticEnergy(om1, om2){
204     const { g, m1, m2, l1, l2 } = getParams();
205
206     let KE = 0.5 * m1 * om1*om1*l1*l1 + 0.5 * m2 * om2*om2*l2*l2;
207
208     return KE;
209 }
210
211 // Returns sum of potential energies of both pendulums
212 function getPotentialEnergy(th1, th2){
213     const { g, m1, m2, l1, l2 } = getParams();
214
```

```
215     const h1 = l1 + l2 - l1 * Math.cos(th1);
216     const h2 = h1 - l2 * Math.cos(th2);
217 
218     const PE = m1 * g * h1 + m2 * g * h2;
219 
220     return PE;
221 }
222 
223 // Returns the maximum energy for given parameters (updates
    everytime parameters change)
224 export function getMaxEnergy(th1, th2, om1, om2){
225     return getKineticEnergy(om1, om2) + getPotentialEnergy(th1, th2)
    ;
226 }
227 
228 // Draws the energy bar to the screen
229 export function drawEnergyBar(ctxEnergy, widthEnergy, heightEnergy,
    th1, th2, om1, om2, maxLabel){
230     const { g, m1, m2, l1, l2 } = getParams();
231     ctxEnergy.clearRect(0,0, widthEnergy, heightEnergy);
232 
233     const paddedHeightEnergy = heightEnergy - 2 * Ypadding;
234     const paddedWidthEnergy = widthEnergy - 2 * Xpadding;
235 
236     const KE = getKineticEnergy(om1, om2);
237     const PE = getPotentialEnergy(th1, th2);
238 
239     const maxE = KE + PE;
240 
241     // Draw the boxes to display energy
242     ctxEnergy.fillStyle = "red";
243     ctxEnergy.fillRect(Xpadding, paddedHeightEnergy*(1-KE/maxE) +
    Ypadding, paddedWidthEnergy, paddedHeightEnergy*KE/maxE);
244     ctxEnergy.fillStyle = "blue";
245     ctxEnergy.fillRect(Xpadding, Ypadding, paddedWidthEnergy,
    paddedHeightEnergy*PE/maxE);
246 
247     // Draw surrounding frame
248     ctxEnergy.strokeRect(Xpadding, paddedHeightEnergy*(1-KE/maxE) +
    Ypadding, paddedWidthEnergy, paddedHeightEnergy*KE/maxE)
249     ctxEnergy.strokeRect(Xpadding, Ypadding, paddedWidthEnergy,
    paddedHeightEnergy*PE/maxE);
250 
251     // Draw labels on each bar
252     ctxEnergy.fillStyle = "black";
253     ctxEnergy.fillText("KE (J)", Xpadding/2, Xpadding);
254     ctxEnergy.fillText("PE (J)", widthEnergy - Xpadding/2, Xpadding)
    ;
```

```
255
256     for (let i = 0; i <= numYPoints; i++) {
257
258         // Get the position and value for labels
259         let yPos = i*paddedHeightEnergy/numYPoints;
260         let Elabel = +((numYPoints-i)*maxLabel/numYPoints).
    toPrecision(2);
261
262         // Draw tick marks on both sides
263         ctxEnergy.beginPath();
264         ctxEnergy.moveTo(Xpadding, yPos + Ypadding);
265         ctxEnergy.lineTo(Xpadding-5, yPos + Ypadding);
266         ctxEnergy.stroke();
267
268         // Draw data points next to tick marks
269         ctxEnergy.textAlign = "right";
270         ctxEnergy.fillText(Elabel, Xpadding-10, yPos + Ypadding +5);
271         ctxEnergy.textAlign = "center";
272
273     }
274
275 }
```

3.4.7 graph.js

This code governs how the graph is drawn and stores the variables for all the different graphs that you can choose. It also draws the graph and controls when the graph is clicked to draw the closest point.

```

// Graph constants
const padding = 5;
const numXTimePoints = 5;
const maxGraphPoints = 200;

//Graph histories
let graphHistoryTh1 = [];
let graphHistoryTh2 = [];
let graphHistoryOm1 = [];
let graphHistoryOm2 = [];
let graphHistoryTime = [];
let drawPoint = false;
let closestPoint = [0,0];

//Clamp angles between pi and -pi
function normaliseAngle(angle){
    angle = angle % (2 * Math.PI);
    if (angle > Math.PI) {
        angle -= 2 * Math.PI;
    } else if (angle < -Math.PI) {
        angle += 2 * Math.PI;
    }
    return angle;
}

// Update the history lists to add the latest values
export function updateHistories(th1, th2, om1, om2, totalFrames){
    graphHistoryTh1.push(normaliseAngle(th1));
    graphHistoryTh2.push(normaliseAngle(th2));
    graphHistoryOm1.push(normaliseAngle(om1));
    graphHistoryOm2.push(normaliseAngle(om2));
    graphHistoryTime.push(totalFrames);

    // Limit the history to maxGraphPoints
    while (graphHistoryTh1.length > maxGraphPoints) {
        graphHistoryTh1.shift();
        graphHistoryTh2.shift();
        graphHistoryOm1.shift();
        graphHistoryOm2.shift();
        graphHistoryTime.shift();
```

```
    }
}

// Empty the histories to reset the graph
export function resetHistories(){
    graphHistoryTh1 = [];
    graphHistoryTh2 = [];
    graphHistoryOm1 = [];
    graphHistoryOm2 = [];
}

// Gets the smallest value in an array of dictionaries by a certain key
const minBy = (arr, key) => arr.reduce((a,b) => a[key] < b[key] ? a: b, {});

// Uses a KNN to get cloest point and returns it.
export function drawClosestPoint(canvasGraph, event, xGraph, yGraph, widthGraph, heightGraph){

    // Calculates the x and y position of where has been clicked
    const rect = canvasGraph.getBoundingClientRect();
    const xClick = event.clientX - rect.left;
    const yClick = event.clientY - rect.top;

    //Scaling
    let yScale = heightGraph / (2 * Math.PI);
    let xScale = widthGraph / (2 * Math.PI);

    // Get xy data
    let xData = eval(xGraph);
    let yData = eval(yGraph);

    if (xGraph != 'graphHistoryTime'){
        let xPoint = -(xClick / xScale - Math.PI);
        let yPoint = -(yClick / yScale - Math.PI);

        let distances = [];
        for (let i = 0; i < xData.length; i++){
            let distance = Math.sqrt((xPoint-xData[i])**2 + (yPoint-yData[i])**2);
            distances.push({'distance': distance, 'point': [xData[i], yData[i]]});
        }

        let closest = minBy(distances, 'distance')

        closestPoint = closest;
```

```
85
86          drawPoint = true;
87
88      };
89
90
91
92
93  }
94
95  // Draws the graph to the canvas
96  export function drawGraph(ctxGraph, xGraph, yGraph, widthGraph,
        heightGraph, pauseCheck) {
97
98      ctxGraph.clearRect(0,0, widthGraph, heightGraph);
99
100     // Calculate padding so graph items don't go over the edge
101     let paddedHeightGraph = heightGraph * (1 - padding/100);
102     let paddedWidthGraph = widthGraph * (1 - padding/100);
103
104     //Scaling
105     let yScale = paddedHeightGraph / (2 * Math.PI);
106     let xScale = paddedWidthGraph / (2 * Math.PI);
107
108     // Get xy data
109     let xData = eval(xGraph);
110     let yData = eval(yGraph);
111
112     // Draw axes
113     ctxGraph.beginPath();
114     ctxGraph.moveTo(0, heightGraph / 2);
115     ctxGraph.lineTo(widthGraph, heightGraph / 2);
116     ctxGraph.moveTo(widthGraph / 2, 0);
117     ctxGraph.lineTo(widthGraph / 2, heightGraph);
118     ctxGraph.strokeStyle = "#000000";
119     ctxGraph.stroke();
120
121     //+-pi on y scale
122     ctxGraph.fillText("\u03C0", widthGraph/2-15, heightGraph/2 -
        Math.PI*yScale + 5);
123     ctxGraph.fillText("-\u03C0", widthGraph/2-15, heightGraph/2 +
        Math.PI*yScale + 5);
124
125     //Tick marks for y axes
126     ctxGraph.beginPath();
127     ctxGraph.moveTo(widthGraph/2 - 5, heightGraph/2 - Math.PI*yScale
        );
128     ctxGraph.lineTo(widthGraph/2, heightGraph/2 - Math.PI*yScale);
```

```
129     ctxGraph.moveTo(widthGraph/2 - 5, heightGraph/2 + Math.PI*yScale
    );
130     ctxGraph.lineTo(widthGraph/2, heightGraph/2 + Math.PI*yScale);
131     ctxGraph.stroke();
132 
133     //x axes for angle variables
134     if (xGraph != 'graphHistoryTime'){
135         //+-pi on x scale
136         ctxGraph.fillText("\u03C0", widthGraph/2 + Math.PI*xScale,
    heightGraph/2 + 20);
137         ctxGraph.fillText("-\u03C0", widthGraph/2 - Math.PI*xScale,
    heightGraph/2 + 20);
138 
139         //tick marks for x axes
140         ctxGraph.beginPath();
141         ctxGraph.moveTo(widthGraph/2 - Math.PI*xScale, heightGraph/2
     + 5);
142         ctxGraph.lineTo(widthGraph/2 - Math.PI*xScale, heightGraph
    /2);
143         ctxGraph.moveTo(widthGraph/2 + Math.PI*xScale, heightGraph/2
     + 5);
144         ctxGraph.lineTo(widthGraph/2 + Math.PI*xScale, heightGraph
    /2);
145         ctxGraph.stroke();
146 
147         //Loop through graph history and plot
148         for (let i = 0; i <= xData.length; i++) {
149 
150             ctxGraph.beginPath();
151 
152             let prevY = yData[i];
153             let currY = yData[i+1];
154             let prevX = xData[i];
155             let currX = xData[i+1];
156 
157             ctxGraph.moveTo(widthGraph/2 - prevX*xScale, heightGraph
     / 2 - prevY * yScale);
158             ctxGraph.lineTo(widthGraph/2 - currX*xScale, heightGraph
     / 2 - currY * yScale);
159             ctxGraph.arc(widthGraph/2 - currX*xScale, heightGraph /
    2 - currY * yScale, 1, 0, 2* Math.PI);
160 
161             //Check if points on opposite side of graph and makes
    line invisible
162             if ((Math.abs(prevY - currY) > Math.PI) || (Math.abs(
    prevX - currX) > Math.PI)){
163                 ctxGraph.strokeStyle = "#00000000"; //Invisible if
    angle greater than 2PI
```

```
164             }else{
165                 ctxGraph.strokeStyle = "#FF0000";  // Red for angle
    th1
166             };
167             ctxGraph.stroke();
168         }
169      } else {
170 
171         // x-axis labels and tick marks
172         for (let i = 0; i <= numXTimePoints; i++){
173             let xPos = i * (paddedWidthGraph/numXTimePoints);
174             let time = xData[Math.floor(i*maxGraphPoints/
    numXTimePoints)]/100;
175             if (time == undefined) {time = 100;}
176             time = time.toFixed(2);
177             ctxGraph.fillText(time, xPos, heightGraph/2 + 30);
178 
179             ctxGraph.beginPath();
180             ctxGraph.moveTo(xPos, heightGraph/2);
181             ctxGraph.lineTo(xPos, heightGraph/2 + 10);
182             ctxGraph.stroke();
183         }
184 
185         let pixelPerDataPoint = paddedWidthGraph / graphHistoryTh1.
    length;
186         for (let i = 1; i < graphHistoryTh1.length; i++){
187 
188             ctxGraph.beginPath();
189 
190             let prevY = yData[i-1];
191             let currY = yData[i];
192 
193             ctxGraph.moveTo((i - 1) * pixelPerDataPoint, heightGraph
     / 2 - prevY * yScale);
194             ctxGraph.lineTo(i * pixelPerDataPoint, heightGraph / 2 -
     currY * yScale);
195 
196             if (Math.abs(prevY - currY) > Math.PI){
197                 ctxGraph.strokeStyle = "#00000000"; //Invisible if
    angle greater than 2PI
198             }else{
199                 ctxGraph.strokeStyle = "#FF0000";  // Red for angle
    th1
200             }
201 
202             ctxGraph.stroke();
203 
204         }
```

```
205
206     }
207
208     // Draws circle around closest point clicked on and shows the
    values of the point
209     if (drawPoint && pauseCheck.checked == true) {
210         let closestCoords = closestPoint.point;
211         ctxGraph.beginPath();
212         ctxGraph.arc(widthGraph/2 - closestCoords[0]*xScale,
    heightGraph / 2 - closestCoords[1] * yScale, 10, 0, 2* Math.PI);
213         ctxGraph.fillText(`${closestCoords[0].toFixed(2)}, ${
    closestCoords[1].toFixed(2)}`, widthGraph/2 - closestCoords[0]*
    xScale + 10, heightGraph / 2 - closestCoords[1] * yScale - 10);
214         ctxGraph.stroke();
215     }
216
217 }
```

3.4.8 package.json

This is the JSON that has the information about the whole project. It stores all of the dependencies and their versions to keep version control consistent. Dependencies are downloaded to the server that runs the webpage.

```
{
  "name": "physteach",
  "version": "1.0.0",
  "description": "Physics Teach Aid",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Alex Makelberge",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "mongodb": "^6.3.0",
    "pug": "^3.0.2"
  },
  "keywords": []
}
```

4 Testing

4.1 Test strategy

I am going with three different testing strategies. For the functions I am going to test them by inputting normal, boundary, and erroneous data and checking if their outputs are correct. For the: differential function, drawing of the pendulum, drawing of the graph, drawing of the energy bars, finding circle intersections, main run loop, nearest neighbour I will be making a testing video demonstrating me using every part to show that they work. For the numerical integration I will use a simple differential equation to compare their results to a known solution and seeing how their outputs vary with step height.

4.2 Functions

4.2.1 Tests

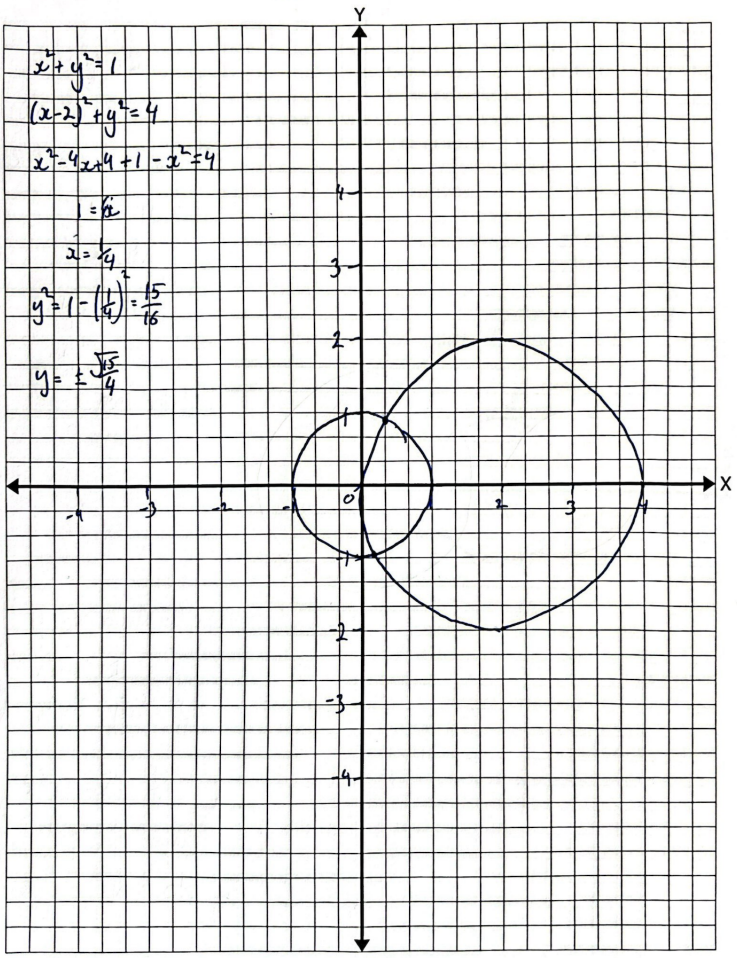
Function	Type	Input	Output	Passed?	Objective
NormaliseAngle	Normal	8.62	2.34	y	2.1 & 4.2
NormaliseAngle	Boundary	π	π	y	2.1 & 4.2
getKineticEnergy	Normal	0.24, 0.45	6502.5	y	3.2
getKineticEnergy	Boundary	π, π	493480	y	3.2
getPotentialEnergy	Normal	$\pi/2, \pi/2$	19620	y	3.2
getPotentialEnergy	Boundary	π, π	34335	y	3.2
findCircleIntersection	Normal	0,0,1,2,0,2	0.25, 0.9682	y	2.1.4
findCircleIntersection	Boundary	0,0,1,2,0,1	1,0	y	2.1.4
findCircleIntersection	Erroneous	0,0,1,3,0,1	no solution	y	2.1.4

4.2.2 findCircleIntersections:

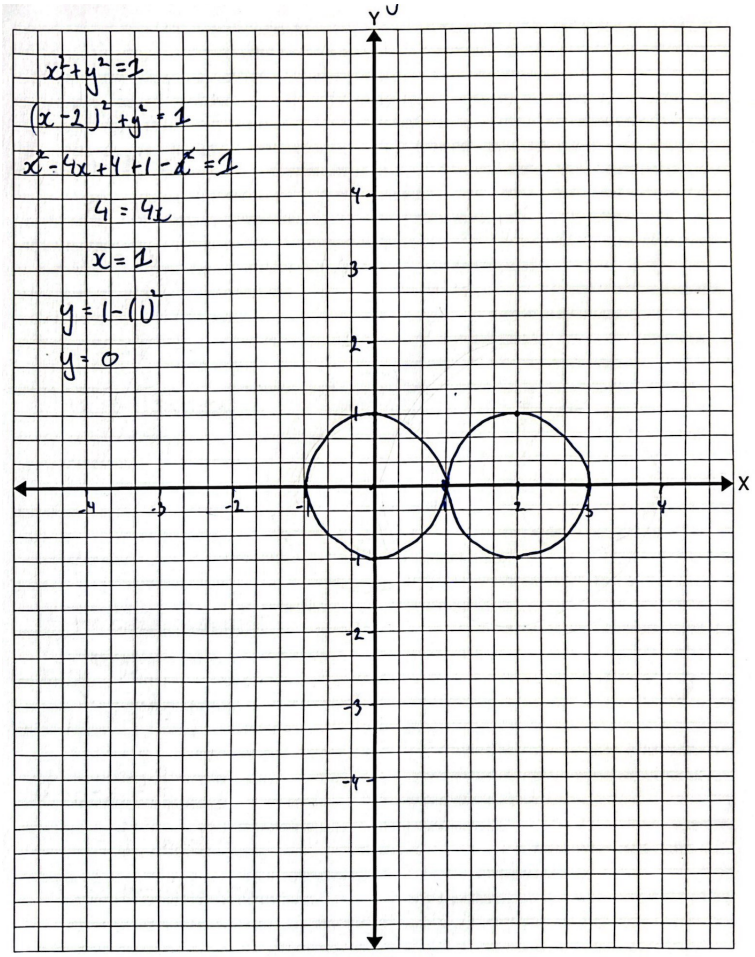
Results from algorithm:

1. **Normal:** $[0.25000000000000006, 0.9682458365518543]$
2. **Boundary:** $[1, 0]$
3. **Erroneous:** false, false

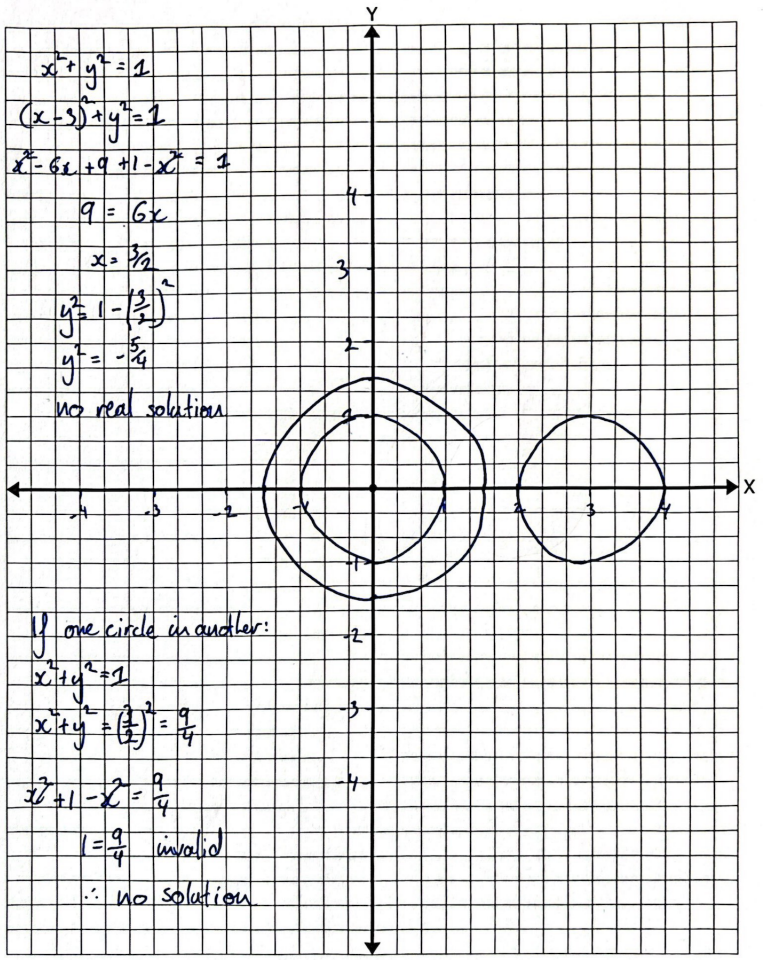
Normal



Boundary



Erroneous



Conclusion

The program is working as intended with all inputs aligning with the hand calculated values. For erroneous values it returns false which the rest of the program handles to not move the pendulum to that position.

4.3 Testing video

Here is the link to my testing video:

`https://youtu.be/wpPRPoB_Hpc`

4.4 Numerical integration

4.4.1 Differential equation

To test my numerical integration I ran it on a very simple differential equation that is easy to find the solution to:

$$\frac{dy}{dx} = y \tag{69}$$

Which solves like this:

$$\int \frac{1}{y} dy = \int 1 dx \tag{70}$$

$$\ln y = x + c \tag{71}$$

$$y = e^{x+c} \tag{72}$$

$$y = Ae^{x} \tag{73}$$

For the purpose of testing I am letter $A = 1$.

4.4.2 Code

I have copied the integration method directly from the code. I used Graph.js for the graphing functionality to print it to the screen.

```
1
2 function differential(x,y,z,w){
3     return [x,y,z,w]
4 }
5
6 //Integration method
7 function Integration(method, h, th1, th2, om1, om2) {
8     if (method === 'RK4'){
9         let [k1_th1, k1_th2, k1_om1, k1_om2] = differential(th1, th2
    , om1, om2);
10         let [k2_th1, k2_th2, k2_om1, k2_om2] = differential(
11             th1 + 0.5 * h * k1_th1,
12             th2 + 0.5 * h * k1_th2,
13             om1 + 0.5 * h * k1_om1,
14             om2 + 0.5 * h * k1_om2
15         );
16         let [k3_th1, k3_th2, k3_om1, k3_om2] = differential(
17             th1 + 0.5 * h * k2_th1,
18             th2 + 0.5 * h * k2_th2,
19             om1 + 0.5 * h * k2_om1,
20             om2 + 0.5 * h * k2_om2
21         );
```

```
        let [k4_th1, k4_th2, k4_om1, k4_om2] = differential(
            th1 + h * k3_th1,
            th2 + h * k3_th2,
            om1 + h * k3_om1,
            om2 + h * k3_om2
        );

        th1 += (h / 6) * (k1_th1 + 2 * k2_th1 + 2 * k3_th1 + k4_th1);
        th2 += (h / 6) * (k1_th2 + 2 * k2_th2 + 2 * k3_th2 + k4_th2);
        om1 += (h / 6) * (k1_om1 + 2 * k2_om1 + 2 * k3_om1 + k4_om1);
        om2 += (h / 6) * (k1_om2 + 2 * k2_om2 + 2 * k3_om2 + k4_om2);

        return [th1, th2, om1, om2];


    } else if (method === 'Euler'){
        let [dth1, dth2, dom1, dom2] = differential(th1, th2, om1, om2);

        th1 += h*dth1;
        th2 += h*dth2;
        om1 += h*dom1;
        om2 += h*dom2;

        return [th1, th2, om1, om2];
    }
}

function solution(t, A=1){
    return A*Math.exp(t);
}

const h = 0.1;
const finalVal = 10;
const steps = finalVal/h;
let x=1, y=1, z=1, w=1;
let a=1, b=1, c=1, d=1;
let solutionX = 0;
let dataEuler = [];
let dataSolution = [];
let dataRK4 = [];

for (let t = 0; t <= steps; t++){
    let time = t*h;
```

```
65
66      dataSolution.push({x:solutionX, time:time});
67      solutionX = solution(time);
68
69      dataRK4.push({x:a, time:time});
70      [a, b, c, d] = Integration('RK4', h, a, b, c, d);
71
72      dataEuler.push({x:x, time:time});
73      [x, y, z, w] = Integration('Euler', h, x, y, z, w);
74
75  }
76
77  var ctx = document.getElementById('testCanvas').getContext('2d');
78  var chart = new Chart(ctx, {
79      type: 'line',
80      data: {
81          labels: dataRK4.map(data => data.time.toFixed(2)),
82          datasets: [{
83              label: 'RK4',
84              borderColor: 'rgb(255, 99, 132)',
85              data: dataRK4.map(data => data.x),
86              fill: false,
87          }, {
88              label: 'Euler',
89              borderColor: 'rgb(99, 255, 132)',
90              data: dataEuler.map(data => data.x),
91              fill: false,
92          }, {
93              label: 'Solution',
94              borderColor: 'rgb(132, 99, 255)',
95              data: dataSolution.map(data => data.x),
96              fill: false,
97          }]
98      },
99      options: {
100         scales: {
101             x: {
102                 title: {
103                     display: true,
104                     text: 'Time'
105                 }
106             },
107             y: {
108                 title: {
109                     display: true,
110                     text: 'Displacement'
111                 }
112             }
```

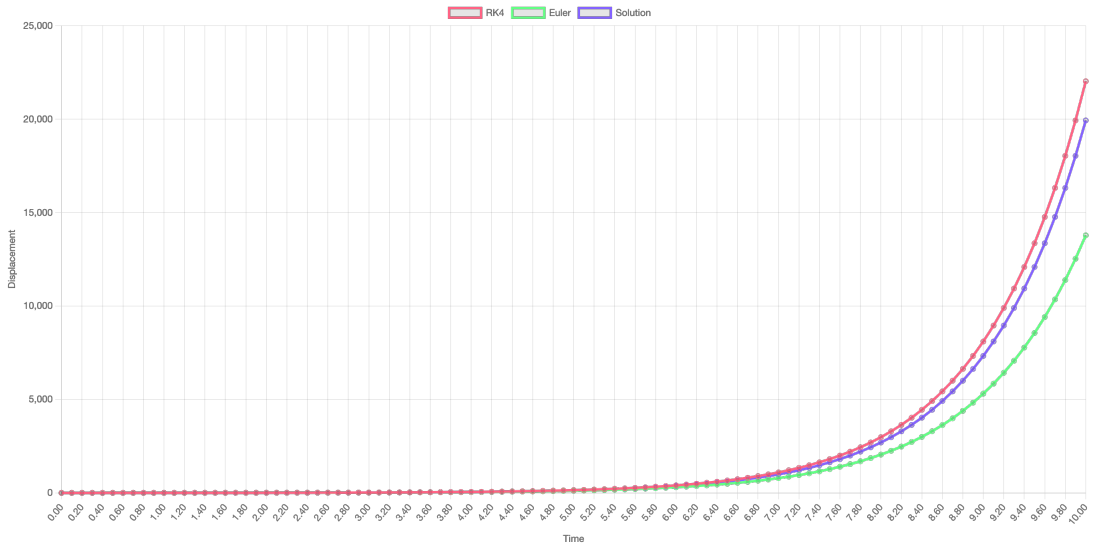
```
        }
    }
});
```

HTML:

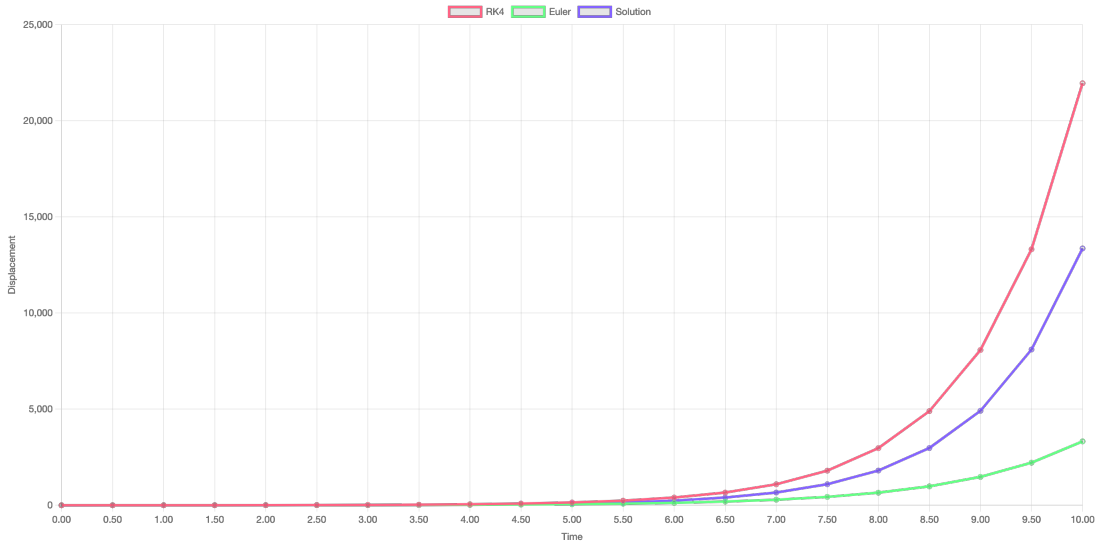
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
    =1.0">
    <title>RK4 and Euler comparison</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <canvas id="testCanvas" width="800" height="400"></canvas>
    <script src="test.js"></script>
</body>
</html>
```

4.4.3 Results

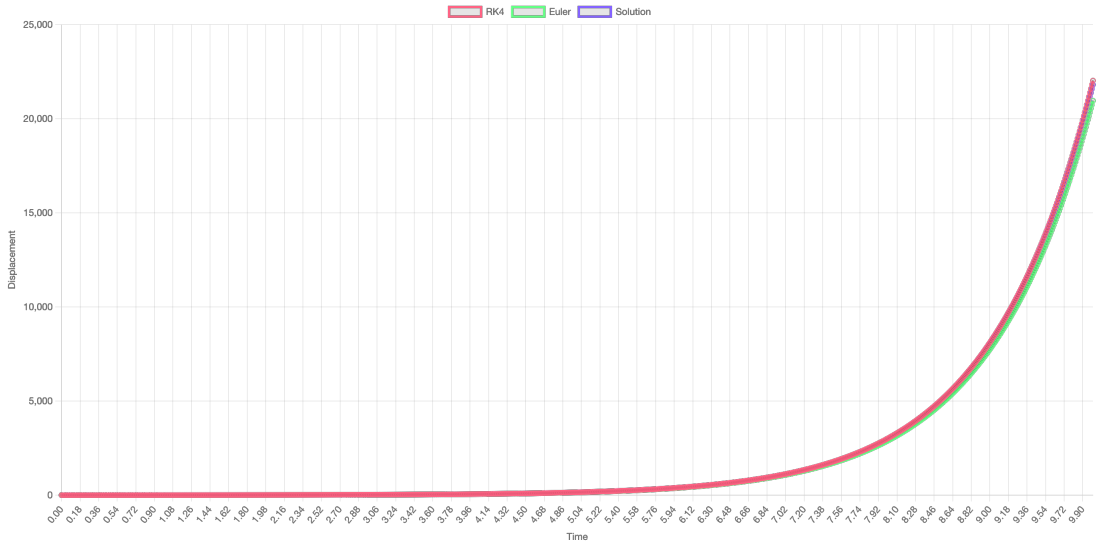
Step height set to 0.1:



Step height set to 0.5:



Step height set to 0.01:



It is clear to see that when you change the step height, the estimated values are a lot closer to the true values, this shows that the integration function is working as expected.

5 Evaluation

5.1 Objectives

These are a copy of my objectives to make it easier to navigate to and from them.

1. **General**

   1.1. The program must run as a website.

   1.2. It must have a GUI

      1.2.1. It must be completely controllable by a mouse

      1.2.2. You should be able to fine tune values by typing them in.

      1.2.3. You should be able to get the data of the pendulum.

   1.3. The website must be able to run on school computers.

      1.3.1. It must not have any pop-ups.

      1.3.2. It must not use any banned websites/data.

      1.3.3. It must not contain any code that needs to be downloaded to the user's computer.

      1.3.4. It must work on any browser engines which could include WebKit, Blink, and Gecko.

      1.3.5. It must be efficient and run quickly on low-performance computers.

2. **Graphs**

   2.1. There must be graphs that dynamically update while the program runs.

      2.1.1. Choose any 2 variables to plot against each other.

      2.1.2. Be able to wipe the graph.

      2.1.3. Be able to pause and stop the recording of data.

      2.1.4. Be able to click on a piece of data and it shows its values.

3. **Energy bars**

   3.1. The values must dynamically update as the simulation runs

   3.2. The calculated values must be correct and accurate to the simulation

   3.3. It must be easy to read off the bars to see the values of the kinetic energy.

4. **Double Pendulum problem**

   4.1. It must use the correct equations.

4.2. It must run accurately.

4.3. User must be able to dynamically change the simulation as it runs.

- **4.3.1**. Change gravity.
- **4.3.2**. Independently change the length of each rod.
- **4.3.3**. Independently change the mass of each part.
- **4.3.4**. Drag the pendulum and move it to any starting position you want.
- **4.3.5**. Change the numerical method used to solve it.
- **4.3.6**. Change the integration step height

4.4. Have options to view each of the modes of the double pendulum

- **4.4.1**. Simple oscillation (like the single pendulum).
- **4.4.2**. Alternating oscillation (top to the left, bottom to right, mean position stays constant).
- **4.4.3**. Chaotic motion without enough energy to flip over.
- **4.4.4**. Chaotic motion with enough energy to flip over.

4.5. Be able to add their own presets to the simulation

4.6. Be able to load any of the added presets

5.2 Client's evaluation

Do you think the simulation had a complete GUI where everything was controllable by a mouse and it was easy to fine tune values?

TM: Yes, I did like the user interface and felt that it was comprehensive and completed everything that I would want it to. I particularly liked the cleanliness of it and how it was uncluttered. Also you'd thought about making it fool proof and not braking when you tried to put the pendulum in an impossible position, which was excellent and would easily have annoyed someone if it glitched at that point. It was really easy to fine tune, save a preset and to trial the presets already available.

Do you think the program ran accurately and was faithful to the double pendulum problem?

TM: I feel it was faithful to the kind of dynamics you would see and I particularly liked the unstable equilibrium demonstration and was impressed there was enough error/random nudging in the code method to mean that it can caused it to fall from the unstable equilibrium. It was particularly interesting that the two different integration methods produced wildly different results though, and would be interesting to think about why that is in more detail (or be shown by your program!)

Do you think there were enough presets and was adding a preset easy enough?

TM: I think the range of presets was good and can't think of any more on the top of my head that I would want, similarly adding a preset looked intuitive.

What was not as effective?

TM: I think if someone were to click onto the page without knowing a little it perhaps was sparse in terms of instructions, and maybe a few little boxes just highlighting that you could click on the picture to change the starting point, that you could click on the graph to find values, and maybe just a box defining the angles as anticlockwise from the negative y axis could have been useful, and labelling the axes. Saying all that I liked that it was sparse and uncluttered so I feel bad suggesting this as a criticism. So less the program and just the presentation of the instructions, or even just a link to a page of 5 top tips for using the animation or something so that you would then keep the uncluttered nature of the animation but someone could easily access the instructions and or go into more depth about the research that you have done. You could even link in your key findings of your analysis so that a user could use the presets to

visualise your results. A tutorial button would be even cooler, one that lead you through the key presets and pointed out things (like the unstable equilibrium, or what things you could do with the app).

What would you want to be improved?

TM: I would quite like the graph to have a convenient equal interval scale on the left (that would have to change with each run). Also it might be visually nice to have it as a line graph over time as opposed to a bar chart for energy, so you could see the total energy staying constant but you can visualise them both going up and down. Might be cool to have a pie chart option too, so the size of the whole pie would represent the total energy and then pizza slices could grow and shrink. Would give a visual sense of the total energy changing with the change in start point. Also might look pretty and be able to see patterns in the energy data easier, especially with a time series graph (e.g. if it got into an oscillating mode or something). Would it be easy to push the code to a triple pendulum? Or one where the bars were stiff springs?

5.2.1 Conclusion

Dr Marlow clearly thinks overall the simulation runs accurately and faithfully to the double pendulum and that all of the features work correctly. However, he did mention that it might not be obvious for someone who has never used the website to see all of the features so a tutorial guide is needed. Additionally he wanted a convenient equal interval scale on the left of the graph to more clearly display the values on the graph, and more clear labelling of what $\theta_1, \theta_1, \omega_1, \omega_2$ mean and how they are measured from the negative y-axis.

5.3 Completed objectives

Objective	Completed?	Proof
1.1	y	Testing video
1.2	y	Client Evaluation
1.2.1	y	Testing video
1.2.2	y	Testing video
1.2.3	y	Testing video
1.3	y	Testing video
1.3.1	y	Client evaluation
1.3.2	y	Testing video
1.3.3	y	Testing video
1.3.4	y	Testing video
1.3.5	y	Testing video
2.1	y	Testing video
2.1.1	y	Testing video
2.1.2	y	Testing video
2.1.3	y	Testing video
2.1.4	y	Testing video
3.1	y	Testing video
3.2	y	Subsection 4.2.1
3.3	y	Client evaluation
4.1	y	Section 4.2.1
4.2	y	Section 4.4 Client evaluation
4.3	y	Testing video
4.3.1	y	Testing video
4.3.2	y	Testing video
4.3.3	y	Testing video
4.3.4	y	Testing video
4.3.5	y	Testing video
4.3.6	y	Testing video
4.4	y	Testing video
4.4.1	y	Testing video
4.4.2	y	Testing video
4.4.3	y	Testing video
4.4.4	y	Testing video
4.5	y	Client evaluation
4.6	y	Client evaluation

5.3.1 Group 1 objectives

I feel all of these objectives have been properly achieved. Arguably objective 1.3.3 is slightly not achieved as to host the website the server needs to download lots of dependencies such as NodeJS; on the other hand, users themselves don't need to download anything and can just type in the site's address.

5.3.2 Group 2 objectives

These have all clearly been met. The graph is very interactive and all of the features are accurate and work well. From Dr Marlow's suggestions the graph could be improved by adding an equal interval scale on the left of the graph to make it easier to read from the graph.

5.3.3 Group 3 objectives

These objectives have also been met. From Dr Marlow, objective 3.3 could be improved by adding a line chart and pie chart to show how the kinetic and potential energy change over time and show how their sum remains constant.

5.3.4 Group 4 objectives

The double pendulum objectives have also been met. 4.3 could be improved by adding more control such as adding a third pendulum to the simulation of changing the material of the pendulum.

5.4 Overall evaluation

Overall I believe the simulation gives a faithful representation of the double pendulum problem and will be a useful teaching aid for the Charterhouse physics department to help in the teaching of chaotic motion. Dr Marlow believes the simulation did all the things needed for an effective tool to help teach students about the double pendulum. I believe it is a good foundation for a very ambitious teaching aid that could incorporate many different aspects of physics and could be used to help teaching physics from GCSE to beyond A-level material.

5.5 What to improve

Based on feedback from Dr Marlow, the analysis and my own ideas here are some things that I could do to further improve the teaching aid. **Improvements:**

- **Add more simulations**

  As mentioned by Dr Marlow, he questioned whether the program could incorporate a triple pendulum or when the rods are stiff springs. These would have to be separate simulations but would definitely be nice to have. This could link into having many pages of simulations. This could be even further extended to build an API where users can build their own simulations with fundamentals of the system.

- **Add more rigorous physics diagrams**

  Dr Marlow suggested adding labels for the angles and angular velocities to make them more obvious what they are. He also wanted potentially show the angles on the pendulum with a negative y-axis displayed. This could be a toggle in the control section that changes the drawPendulum function.

- **Add tutorial**

  Dr Marlow mentioned that without some prior knowledge, the simulation could be quite tricky to use especially with some of the features not being obvious. On the README.md of my github repository I have written a brief tutorial but I should make it more accessible and additionally have a more interactive tutorial which guides you to using the whole thing. Here is the url to my github: `https://github.com/AMakelberge/PhysTeachNew`

- **Add damping**

  Add a feature to introduce frictional forces into the system that reduces the energy of the bars and make them slow down. This would make the differential equation significantly more complicated but would be possible.

- **Online hosting**

  Like the other physics simulation websites, it would be nice to set up a server to have the website run on there. This would require having all inputs secured against potential SQL attacks or other cyber-security attacks. Also this would link nicely with the first improvement, it would be a nice platform for more simulations to be added.

6 References

References

[1] Britannica, March 2024, *chaos theory*, accessed 20 March 2024, https://www.britannica.com/science/chaos-theory

[2] Kathy Perkins, 2023, *About PhET*, accessed 31 May 2023, https://phet.colorado.edu/en/about

[3] Erik Neumann, April 2001, *About the Author*, accessed 31 May 2023, https://www.myphysicslab.com/index-en.html

[4] Erik Neumann, April 2001, *Open Source Software*, accessed 31 May 2023, https://www.myphysicslab.com/index-en.html

[5] Diego Assencio, 38 February 2014, *The double pendulum: Lagrangian formulation*, accessed 19 August 2023, https://diego.assencio.com/?index=1500c66ae7ab27bb0106467c68feebc6

[6] Liu Y / Yu H / Cang S, 2012, *Modelling and motion control of a double-pendulum driven cart*, accessed 20 March 2024, https://journals.sagepub.com/doi/10.1177/0959651811414507

[7] Wikipedia, 1st September 2023, *Euler method*, accessed 15 September 2023, https://en.wikipedia.org/wiki/Euler_method

[8] Wikipedia, 15th August 2023, *Runge-Kutta methods*, accessed 15 September 2023, https://en.wikipedia.org/wiki/RungeKutta_methods

[9] Double Pendulum, Drew Baden May 2016, *Double pendulum*, accessed 28 February 2024, http://www.physics.umd.edu/hep/drew/pendulum2.html

[10] Wikipedia, 7 March 2024, *k-nearest neighbours algorithm*, accessed 20 March 2024, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[11] Wolfram MathWorld, 15 March 2024, *Circle-Circle Intersection*, accessed 12 February 2024, https://mathworld.wolfram.com/Circle-CircleIntersection.html

[12] Simbla, May 2022, *AI-Generated cloud CRM*, accessed 22 January 2024, https://www.simbla.com/advantagesanddisadvantagesofonlinedatabasebuilder