PHYS3071 Assignment 3

Ryan White
s4499039

September 18, 2022

0.1 Double Derivative Solution

To obtain a numerical solution of the double pendulum at some time t, we first need analytic expressions of the double derivatives of each mass with respect to the origin. Beginning with the information from the task sheet, we have that

$$-A(t) = (m_1 + m_2)l_1\theta_1''(t) + m_2l_2\theta_2''(t)\cos(\theta_1(t) - \theta_2(t)) \tag{1}$$
$$-B(t) = m_2l_2\theta_2''(t) + m_2l_1\theta_1''(t)\cos(\theta_1(t) - \theta_2(t)) \tag{2}$$

where

$$A(t) = m_2l_2(\theta_2'(t))^2\sin(\theta_1(t) - \theta_2(t)) + (m_1 + m_2)g\sin\theta_1(t) \tag{3}$$
$$B(t) = -m_2l_1(\theta_1'(t))^2\sin(\theta_1(t) - \theta_2(t)) + m_2g\sin\theta_2(t) \tag{4}$$

for masses m_1, m_2 and pendulum lengths l_1, l_2. The subscript 1 corresponds to the initial mass connected to the origin, where ths subscript 2 corresponds to the second mass connected to the first. We aim to obtain functions of $\theta''(t)$ (for both θ_1 and θ_2), and so we can simultaneously solve equations (1) and (2) using

$$\underline{C}\underline{X} = \underline{S} \tag{5}$$

where

$$\underline{C} = \begin{bmatrix} l_1(m_1 + m_2) & m_2l_2\cos(\theta_1 - \theta_2) \\ m_2l_1\cos(\theta_1 - \theta_2) & m_2l_2 \end{bmatrix}; \qquad \underline{X} = \begin{bmatrix} \theta_1'' \\ \theta_2'' \end{bmatrix}; \qquad \underline{S} = \begin{bmatrix} -A(t) \\ -B(t) \end{bmatrix} \tag{6}$$

A solution for $\underline{X}$ can be found by taking the inverse of $\underline{C}$: $\underline{X} = \underline{C^{-1}S}$, where

$$\underline{C}^{-1} = \frac{1}{\det C}\begin{bmatrix} C_{22} & -C_{12} \\ -C_{21} & C_{11} \end{bmatrix} = \frac{1}{m_2l_1l_2(m_1 + m_2) - m_2^2l_1l_2\cos^2(\theta_1 - \theta_2)}\begin{bmatrix} m_2l_2 & -m_2l_2\cos(\theta_1 - \theta_2) \\ -m_2l_1\cos(\theta_1 - \theta_2) & l_1(m_1 + m_2) \end{bmatrix} \tag{7}$$

And so, computing $\underline{C}$: $\underline{X} = \underline{C^{-1}S}$ gives

$$\theta_1'' = \frac{-m_2l_2A(t) + m_2l_2\cos(\theta_1 - \theta_2)B(t)}{m_2l_1l_2(m_1 + m_2) - m_2^2l_1l_2\cos^2(\theta_1 - \theta_2)} = \frac{-A(t) + \cos(\theta_1 - \theta_2)B(t)}{l_1(m_1 + m_2) - m_2l_1\cos^2(\theta_1 - \theta_2)} \tag{8}$$
$$\theta_2'' = \frac{m_2l_1\cos(\theta_1 - \theta_2)A(t) - l_1(m_1 + m_2)B(t)}{m_2l_1l_2(m_1 + m_2) - m_2^2l_1l_2\cos^2(\theta_1 - \theta_2)} = \frac{m_2\cos(\theta_1 - \theta_2)A(t) - (m_1 + m_2)B(t)}{m_2l_2(m_1 + m_2) - m_2^2l_2\cos^2(\theta_1 - \theta_2)} \tag{9}$$

0.2 ODE Systems

Now, to find the position of the masses at each point in time, the angles of each mass (with respect to the origin) need to be found given values at the previous time step. For Euler's method, we have

$$\theta_i(t + \Delta t) = \theta_i(t) + \Delta t\theta_i'(t) \tag{10}$$
$$\theta_i'(t + \Delta t) = \theta_i'(t) + \Delta t\theta_i''(t) \tag{11}$$

and for the Runge-Kutta4 method, we have

$$\begin{aligned} k_1 &= \Delta t\theta_i'(t, \theta_i) & k_1' &= \Delta t\theta_i''(t, \theta_i) \\ k_2 &= \Delta t\theta_i'(t, \theta_i + k_1/2) & k_2' &= \Delta t\theta_i''(t, \theta_i + k_1'/2) \\ k_3 &= \Delta t\theta_i'(t, \theta_i + k_2/2) & k_3' &= \Delta t\theta_i''(t, \theta_i + k_2'/2) \\ k_4 &= \Delta t\theta_i'(t, \theta_i + k_3) & k_4' &= \Delta t\theta_i''(t, \theta_i + k_3') \\ \theta_i(t + \Delta t) &= \theta_i(t) + \tfrac{1}{6}\left(k_1 + 2(k_2 + k_3) + k_4\right) & \theta_i'(t + \Delta t) &= \theta_i'(t) + \tfrac{1}{6}\left(k_1' + 2(k_2' + k_3') + k_4'\right) \end{aligned} \tag{12}$$

where the (x, y) positions of each mass (subscript 1 for the inner mass and 2 for the outer mass) are given by

$$\begin{aligned} x_1 &= l_1 \sin\theta_1 & x_2 &= l_1 \sin\theta_1 + l_2 \sin\theta_2 \\ y_1 &= -l_1 \cos\theta_1 & y_2 &= -l_1 \cos\theta_1 - l_2 \cos\theta_2 \end{aligned} \tag{13}$$

The above equations [(3-4) and (8-13)] are exactly those which were implemented in the C++ program to numerically solve the double pendulum. The energy of the system at some time was calculated given its instantaneous velocity and the positions of the masses via

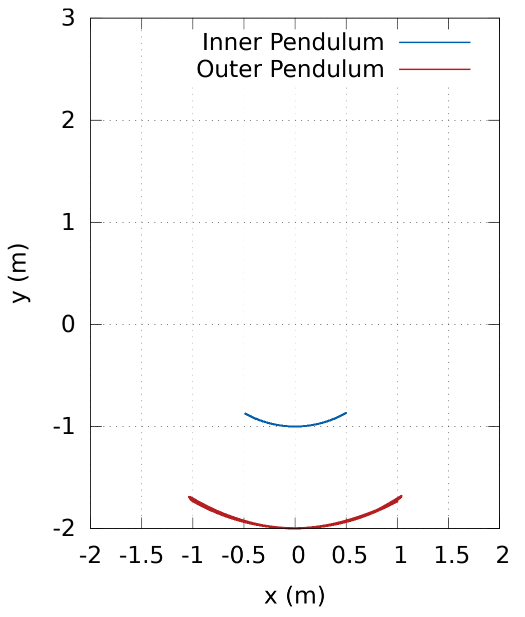
$$E = T + V \approx \frac{1}{2}\left(m_1 v_1^2 + m_2 v_2^2\right) - g\left(m_1 y_1 + m_2 y_2\right) \tag{14}$$

where the approximately symbol arises from the 'instantaneous velocity' approximation, which took into account the change in position over two time steps.

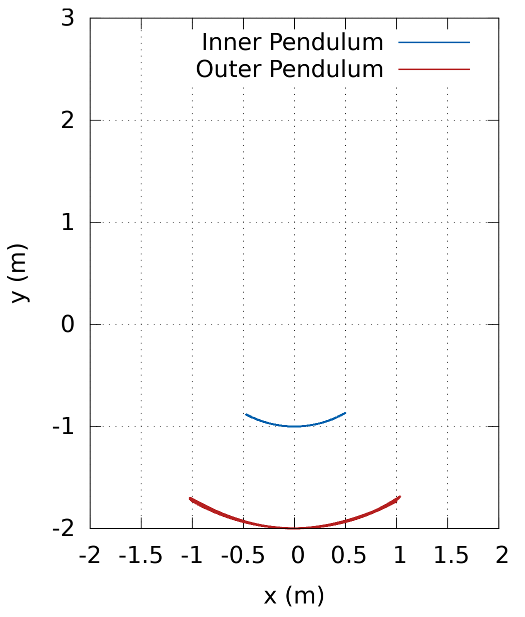
0.3 Double Pendulum Examples and their Energies

This section serves as an answer to both c. and d. in the task sheet.

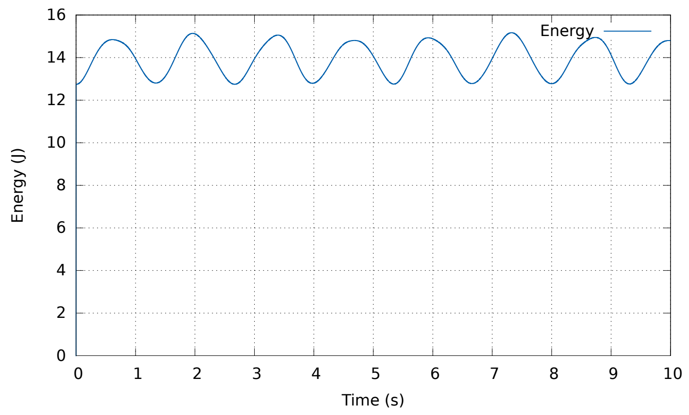
As part of the task requirements, we implemented the Euler and Runge-Kutta4 methods as per the system of equations shown in the previous subsection. The two initial condition states were plotted for each method and are shown in Figures 1 and 2.



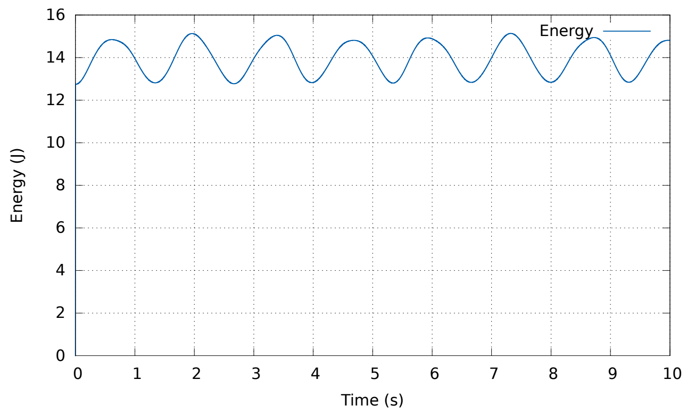
(a) Euler method implementation.



(b) Runge-Kutta4 implementation.

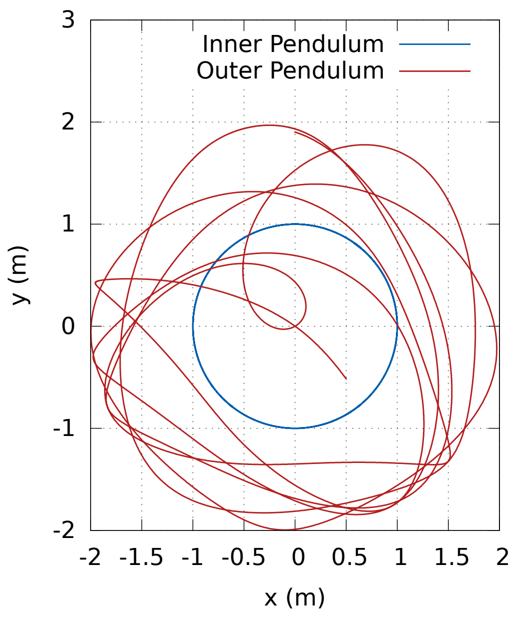


(c) Energy-time graph for the Euler implementation.

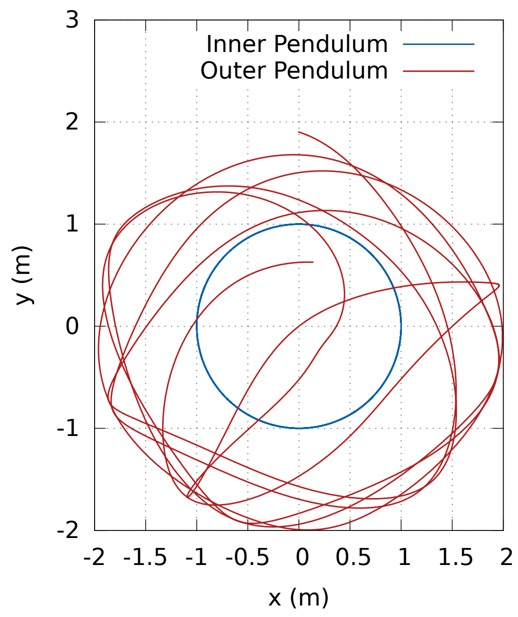


(d) Energy-time graph for the RK4 implementation.

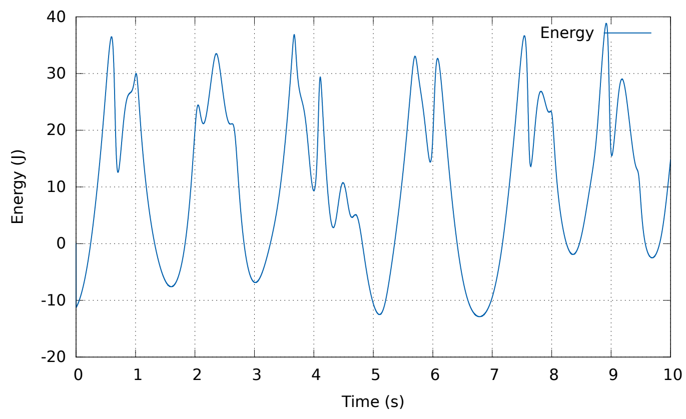
Figure 1: Euler and Runge-Kutta4 methods for numerically solving a double pendulum. The pendulum was simulated over 10 seconds, with 10^4 steps, making a step size of $\Delta t = 10^{-3}$s. The pendulum began with $\theta_1(0) = \theta_2(0) = \pi/6$, and $\theta_1'(0) = \theta_2'(0) = 0$. The left two graphs show the Euler method, while the right two show the RK4 method.



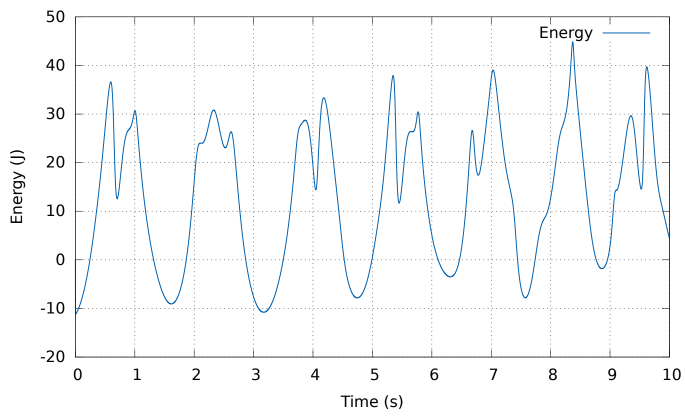
(a) Euler method implementation.



(b) RK4 implementation.



(c) Energy-time graph for the Euler implementation.



(d) Energy-time graph for the RK4 implementation.

Figure 2: Euler and Runge-Kutta4 methods for numerically solving a double pendulum. The pendulum was simulated over 10 seconds, with 10^4 steps, making a step size of $\Delta t = 10^{-3}$s. The pendulum began with $\theta_1(0) = 9\pi/10$, $\theta_2(0) = -9\pi/10$, and $\theta_1'(0) = -\pi$, $\theta_2'(0) = 0$.

Looking at subfigures (c) and (d) in both figures 1 and 2 we clearly see that energy is not conserved (given by the quasi-periodic behaviour of the energy). The fluctuation in energy is not significant for the simple dynamics given in figure 1, but fluctuates on two orders of magnitude in the more complicated case shown in figure 2.
In both cases, we see that the Euler and RK4 methods agree on energy (and hence position of the masses) for the first ~ 2 seconds, and then diverge apart. Upon successive testing, this agreement was found to be heavily dependent on the number of steps simulating the 10 second period, with more steps corresponding with intuitively more consistent results (i.e. a divergence in energies at a later time).

We suspect that the fluctuation in energy is a result of the approximations around the velocity component in equation (14). As the double pendulum is inherently chaotic, the error in the energies would quickly compound over multiple steps, which could explain, at least in part, the observed large scale energy fluctuations. Having the current system energy as a parameter in the functions, and requiring that the energy at the end of the solution be equal to the energy at the start (perhaps by boosting or slowing the velocities of the masses) could help solve this issue.

0.4 A Cool Looking Trajectory

As part of the task description, we generated an interesting system by setting the initial conditions so that the first pass was parallel with the positive x plane, and the second mass was situated at the origin. The first mass had an initial velocity essentially in the negative y direction, and the second mass in the positive y direction. The results of this are shown in figure 3.

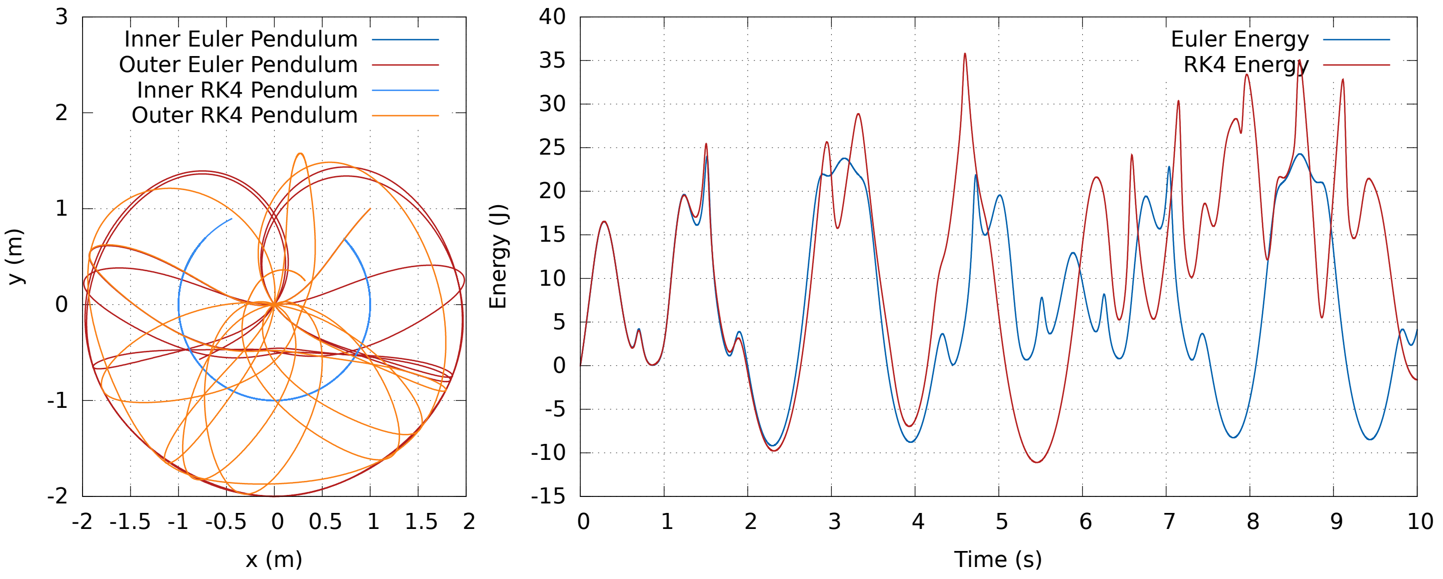


Figure 3: Euler and Runge-Kutta4 methods for numerically solving a double pendulum. The pendulum was simulated over 10 seconds, with 10^4 steps, making a step size of $\Delta t = 10^{-3}$s. The pendulum began with $\theta_1(0) = \pi/2$, $\theta_2(0) = 3\pi/2$, and $\theta_1'(0) = -\pi$, $\theta_2'(0) = \pi$. Left: the positions of each pendulum mass (for each method) over time. Right: the energy associated with each method over time.

The energy is clearly fluctuating and quasi-periodic as in the prior two examples. Similarly, the methods are seen to diverge in solution at around the ~ 2 second mark as before, with the trajectories showing the chaotic divergence as expected. Perhaps a more fitting trajectory is the one shown in the footnote[1].

[1]Me trying to code this in C++