BACHELOR

A numerical approach for the double pendulum

Dams, Ivo Servaas

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science

A numerical approach for the double pendulum

Bachelor Final Project

Ivo Servaas Dams

Supervisor:
Barry Koren

Eindhoven, Tuesday 23rd November, 2021

Contents

Chapter 1

Introduction

The clock is one of the most notable and perhaps most widely used inventions we know. Reading the time off a clock has become much easier the past few decades. A longer time ago, throughout the 18th and 19th centuries, clocks commonly made use of the dynamics of a pendulum. Their greater accuracy allowed for the faster pace of life which was necessary for the Industrial Revolution [1]. There is little doubt that Christiaan Huygens was the first to design an entire clock based on integral pendulum theory [2]. This pendulum had one rod or string and one mass.

While Huygens was able to find extensive analytical results for the simple pendulum, finding analytical results for the double pendulum is much more difficult and usually requires different methods. One might think that the behavior of the double pendulum can still be easily predicted, yet the double pendulum can exhibit distinct behavior: chaos [3]. This means that a tiny difference in its starting position may result in a completely different trajectory. This strong sensitivity to the initial conditions is popularly known as the "butterfly effect" [4]. One could say that the double pendulum could be a stepping stone to better understand, for example, the weather or the climate, which may contain comparable properties.

Because it is very complicated to compute and determine this intricate chaotic behavior analytically, a simulation with the help of a computer is made using numerical methods and analysis. Rather than exact symbolic answers, numerical analysis gives approximate solutions within specified error bounds. Minor mistakes are thus allowed for better accessibility and speed of solutions, but how fast and accurate are different numerical methods when applied to the double pendulum? When is the system predictable and stable or when does a pendulum in a simulation spin out of control? And why? Proper attention should be paid to the choice of method and its parameters.

At first, we will have a look at some properties and analytical solutions of the single pendulum. Then, the basics of the double pendulum will be analysed. An extensive simulation was written and used for this project. Not solely as a way of solving problems, but rather as a means or tool to be used in this field of mathematics. Changing variables are used to check when stability occurs and the reasonableness of the results is verified. Various methods and starting situations are compared and chaos can be demonstrated. The simulations enable an accurate prediction of the behavior of a double pendulum to some extent. Even a simplified version of the double pendulum can exhibit chaos, so it is imperative to understand how to apply numerical methods in such systems before trying them in more complex ones. For instance, the dynamics of the Milky Way, the stock market or other time-dependent problems [3].

Chapter 2

Pendulum

Since the single pendulum has only one rod or string and one mass, its path is quite predictable. In physical reality, such a pendulum generally swings back and forth with a fixed period until it eventually stops due to friction. But what would happen if we would attach another rod with a mass? To start understanding the double pendulum, we will begin with the basics of a single pendulum.

2.1 Single pendulum

We define a single pendulum as a system only containing a point mass attached to a support with a rod. The aim of this report is to find a simulation mainly working in terms of mathematics and not to find a perfect simulation in terms of physics. In order to understand both the single and double pendulum without making things too complicated physics-wise, a few assumptions are made. Some important basic properties of a situation with a single pendulum are in this case:

- There is a rod with length ℓ that is massless and rigid.
- We have a point mass with mass m.
- The pendulum is assumed to move in only two dimensions and not in three dimensions such as we know it in reality.
- There is a uniform gravitational field with constant gravitational acceleration $g = 9.81$ m/s^2.
- The rod is attached to a fixed unmovable support, which does not hinder the movement of the pendulum in any form. The pendulum is able to make a looping.
- There are no friction forces.

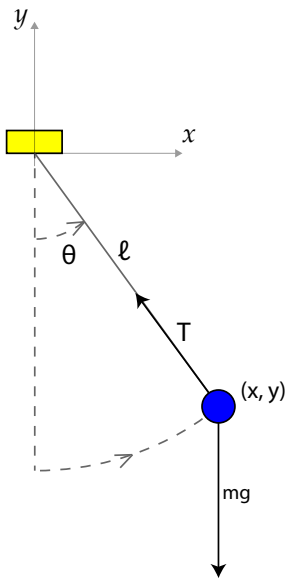

Figure 2.1: Forces acting on a point mass for a single pendulum.

In Figure 2.1 we see a basic schematic of a single pendulum with its basic forces. We have a Cartesian coordinate system with two dimensions x and y. There are two forces acting on the point mass in a single pendulum. The first force is the gravitational force mg pointing downward. The second force is a force in the rod from the point mass pointed towards the support, which we will call the tension, denoted by T. This force is unknown, but will be eliminated later. We define the angular displacement between the y-axis in the negative direction and the rod with variable θ. This angle will shift when the pendulum is moving.

2.1.1 Kinematics

We have a Cartesian coordinate system with two dimensions such that we have the following coordinates for the point mass:

$$x = \ell \sin\theta, \tag{2.1a}$$
$$y = -\ell \cos\theta. \tag{2.1b}$$

For the first- and second-order time derivatives, the velocity and acceleration components of the point mass in x- and y-direction, we find:

$$\dot{x} = \ell\dot{\theta}\cos\theta, \tag{2.2a}$$
$$\dot{y} = \ell\dot{\theta}\sin\theta, \tag{2.2b}$$

$$\ddot{x} = \ell(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta), \tag{2.3a}$$
$$\ddot{y} = \ell(\ddot{\theta}\sin\theta + \dot{\theta}^2\cos\theta). \tag{2.3b}$$

2.1.2 Dynamics

From Newton's second law [5] and using Figure 2.1 we find:

$$m\ddot{x} = -T\sin\theta, \tag{2.4a}$$
$$m\ddot{y} = T\cos\theta - mg. \tag{2.4b}$$

By multiplying equation (2.4a) by $\cos\theta$ and equation (2.4b) by $\sin\theta$ and subsequently adding the two corresponding equations we then find:

$$\ddot{x}\cos\theta + \ddot{y}\sin\theta = -g\sin\theta. \tag{2.5}$$

After substituting the equations (2.3) into (2.5) we arrive at the following equation:

$$\ddot{\theta} = -\frac{g}{\ell}\sin\theta. \tag{2.6}$$

This is the main differential equation used to describe this harmonic oscillator.

2.1.3 Linearization

For small θ we can linearize $\sin\theta$ such that: $\sin\theta \approx \theta$. Using equation (2.6) we then arrive at the following equation:

$$\ddot{\theta} = -\frac{g}{\ell}\theta, \tag{2.7}$$

with $\theta(0) = \theta_0$ (initial angular displacement) and with for instance $\dot{\theta}(0) = 0$. Consequently, the characteristic polynomial is:

$$\lambda^2 = -\frac{g}{\ell},$$

with solutions:

$$\lambda_1 = -i\sqrt{\frac{g}{\ell}}, \quad \lambda_2 = i\sqrt{\frac{g}{\ell}}. \tag{2.8}$$

Then the solution for θ is of the form: $\theta(t) = Ae^{\lambda_1 t} + Be^{\lambda_2 t}$ (for some constants A and B). Imposing the initial conditions yields:

$$\theta(t) = \theta_0 \cos\left(\sqrt{\frac{g}{\ell}}t\right). \tag{2.9}$$

From this equation, Christiaan Huygen's law for the period of an ideal mathematical pendulum can be derived, which has the property that it is independent of θ_0 and m.

2.1.4 Numerical stability analyses

Before considering the double pendulum problem, we will consider two numerical solution methods for the single pendulum problem and analyse the stability of both methods. We can rewrite equation (2.6) to a first-order differential equation. By introducing $p_1 \equiv \theta(t)$ and $p_2 \equiv \dot{\theta}(t)$, we find the following coupled first-order system:

$$\dot{\boldsymbol{p}} = \begin{bmatrix}\dot{p}_1 \\ \dot{p}_2\end{bmatrix} = \begin{bmatrix}p_2 \\ -\frac{g}{\ell}\sin p_1\end{bmatrix} = \begin{bmatrix}f_1(p_1,p_2) \\ f_2(p_1,p_2)\end{bmatrix} = \boldsymbol{f}(\boldsymbol{p}), \qquad t \geq 0. \tag{2.10}$$

In order to compute $p_1 \equiv p_1(t)$ and $p_2 \equiv p_2(t)$ we need the initial conditions $p_1(0) = \theta(0)$ and $p_2(0) = \dot{\theta}(0)$. To solve a first-order system numerically different methods can be used. Firstly, the well-known explicit Forward Euler method will be considered and numerical stability will be analysed in the case of a single pendulum. Forward Euler can be denoted by:

$$\boldsymbol{p}(t_{n+1}) = \boldsymbol{p}(t_n) + \tau\boldsymbol{f}(\boldsymbol{p}_n), \tag{2.11}$$

where $\tau = t_{n+1} - t_n$ is the time-step. In order to analyse the numerical stability of Forward Euler for the single pendulum we will linearize $\boldsymbol{f}(\boldsymbol{p}_n)$ about $\boldsymbol{p}(t_n)$. Then numerical stability is found

when $|Q(\lambda_j\tau)| \equiv |1+\lambda_j\tau| \leq 1$ for both eigenvalues λ_1 and λ_2 of the Jacobian matrix $\boldsymbol{J}$ of $\boldsymbol{f}$ [6]. In this case, the Jacobian matrix of $\boldsymbol{f}$ is the following matrix:

$$\boldsymbol{J}|_{(\boldsymbol{p}(t_n))} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell}\cos p_1 & 0 \end{bmatrix}. \tag{2.12}$$

Now, for $-\frac{\pi}{2} < p_1 < \frac{\pi}{2}$ the eigenvalues are:

$$\lambda_1 = -i\sqrt{\frac{g}{\ell}\cos p_1}, \quad \lambda_2 = i\sqrt{\frac{g}{\ell}\cos p_1} \tag{2.13}$$

and since $|Q(\lambda_1\tau)| = |Q(\lambda_2\tau)| = \sqrt{1+\frac{g}{\ell}\cos p_1\tau^2} > 1$ for $-\frac{\pi}{2} < p_1 < \frac{\pi}{2}$ already, the Forward Euler method is unstable for the single pendulum.

Other methods exist that can be stable. Since the eigenvalues are purely imaginary for $|p_1| < \frac{\pi}{2}$, a method with a stability region containing (part of) the imaginary axis is required. Therefore we will now consider the standard explicit 4-stage Runge-Kutta (RK4) method. This method can be expressed by:

$$\boldsymbol{k}_1 = \boldsymbol{f}(\boldsymbol{p}(t_n)), \tag{2.14a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(\boldsymbol{p}(t_n) + \frac{1}{2}\tau\boldsymbol{k}_1), \tag{2.14b}$$

$$\boldsymbol{k}_3 = \boldsymbol{f}(\boldsymbol{p}(t_n) + \frac{1}{2}\tau\boldsymbol{k}_2), \tag{2.14c}$$

$$\boldsymbol{k}_4 = \boldsymbol{f}(\boldsymbol{p}(t_n) + \tau\boldsymbol{k}_3), \tag{2.14d}$$

$$\boldsymbol{f}(\boldsymbol{p}(t_{n+1})) = \boldsymbol{f}(\boldsymbol{p}(t_n)) + \frac{1}{6}\tau(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4). \tag{2.14e}$$

For the RK4 method, numerical stability is found when [6]:

$$|Q(\lambda\tau)| = |1 + \lambda\tau + \frac{1}{2}(\lambda\tau)^2 + \frac{1}{6}(\lambda\tau)^3 + \frac{1}{24}(\lambda\tau)^4| < 1. \tag{2.15}$$

For the general eigenvalue λ, which is either λ_1 or λ_2, we find:

$$\begin{aligned}
|Q(\lambda\tau)| &= |1 + \lambda\tau + \frac{1}{2}(\lambda\tau)^2 + \frac{1}{6}(\lambda\tau)^3 + \frac{1}{24}(\lambda\tau)^4| \\
&= |1 \pm i\sqrt{\frac{g}{\ell}\cos p_1(t_n)}\tau - \frac{1}{2}\frac{g}{\ell}\cos p_1(t_n)\tau^2 \mp \frac{1}{6}i\left(\frac{g}{\ell}\cos p_1(t_n)\right)^{\frac{3}{2}}\tau^3 + \frac{1}{24}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^4| \\
&= \sqrt{\left(1 - \frac{1}{2}\frac{g}{\ell}\cos p_1(t_n)\tau^2 + \frac{1}{24}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^4\right)^2 + \left(\pm\sqrt{\frac{g}{\ell}\cos p_1(t_n)}\tau \mp \frac{1}{6}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^{\frac{3}{2}}\tau^3\right)^2} \\
&= \sqrt{\begin{aligned}&1 - \frac{g}{\ell}\cos p_1(t_n)\tau^2 + \frac{1}{4}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^4 + \frac{1}{12}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^4 - \frac{1}{24}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^3\tau^6 \\ &+ \frac{1}{576}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^4\tau^8 + \frac{g}{\ell}\cos p_1(t_n)\tau^2 - \frac{1}{3}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^4 + \frac{1}{36}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^3\tau^6\end{aligned}} \\
&= \sqrt{1 - \frac{1}{72}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^3\tau^6 + \frac{1}{576}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^4\tau^8}.
\end{aligned} \tag{2.16}$$

Note that $\frac{1}{576}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^4\tau^8$ is always positive. Therefore $|Q(\lambda\tau)| \leq 1$ when:

$$\begin{aligned}
&\frac{1}{576}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^4\tau^8 \leq \frac{1}{72}\left(\frac{g}{\ell}\cos p_1(t_n)\right)^3\tau^6 \\
\implies &\left(\frac{g}{\ell}\cos p_1(t_n)\right)^2\tau^2 \leq 8\left(\frac{g}{\ell}\cos p_1(t_n)\right) \\
\implies &\tau \leq \sqrt{\frac{8\ell}{g\cos p_1(t_n)}}.
\end{aligned} \tag{2.17}$$

Numerical stability only seems possible for $-\frac{\pi}{2} \leq p_1(t_n) \leq \frac{\pi}{2}$, otherwise one of the eigenvalues will turn into a purely real positive one. For $|p_1| = \frac{\pi}{2}$, the upper bound given in (2.17) for the time-step becomes infinitely large. However, since $Q(\lambda\tau)$ is based on linearization, we need to keep in mind the presumption that $\tau \ll 1$.

2.2 Double pendulum

The system of the double pendulum can be defined similarly to that of the single pendulum. Basically, the double pendulum is the same as a single pendulum with an extra rod with a point mass attached to the first point mass. Most assumptions and conditions still hold, but for a double pendulum we define a few changes to its properties compared to the single pendulum:

- There are two point masses with masses m_1 and m_2 instead of one mass m.
- We have two massless rods with lengths ℓ_1 and ℓ_2 instead of one rod ℓ.
- We have angular displacements θ_1 and θ_2 instead of one θ.
- There are two tensions T_1 and T_2 in the rods instead of one tension T.

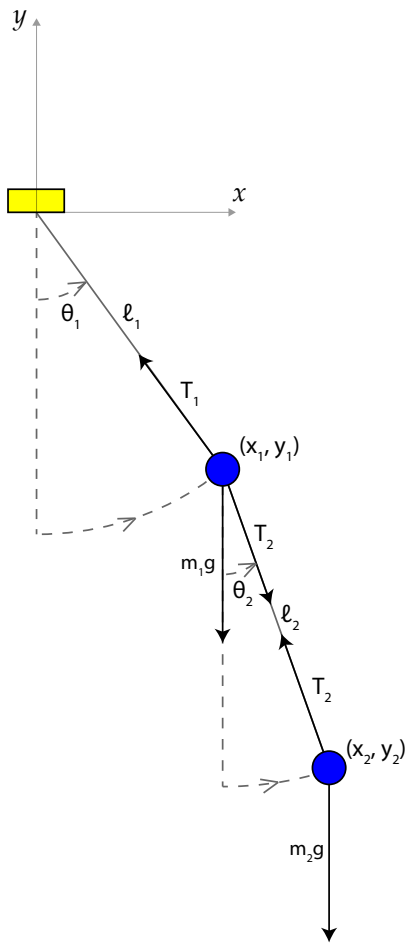

Figure 2.2: Forces acting on the two point masses for a double pendulum.

In Figure 2.2 we see a basic schematic of a double pendulum with its basic forces. The second tension T_2 from the first point mass pointed towards the second point mass is equal to the tension for the opposite situation. Again, the point masses have a gravitational force pointing downward and the tension forces will be eliminated later.

2.2.1 Kinematics

We have a Cartesian coordinate system with two dimensions such that we have the following coordinates for the two point masses:

$$x_1 = \ell_1 \sin\theta_1, \tag{2.18a}$$
$$y_1 = -\ell_1 \cos\theta_1, \tag{2.18b}$$
$$x_2 = \ell_1 \sin\theta_1 + \ell_2 \sin\theta_2, \tag{2.18c}$$
$$y_2 = -\ell_1 \cos\theta_1 - \ell_2 \cos\theta_2. \tag{2.18d}$$

For the first- and second-order time derivatives, we find:

$$\dot{x}_1 = \ell_1 \dot{\theta}_1 \cos\theta_1, \tag{2.19a}$$
$$\dot{y}_1 = \ell_1 \dot{\theta}_1 \sin\theta_1, \tag{2.19b}$$
$$\dot{x}_2 = \ell_1 \dot{\theta}_1 \cos\theta_1 + \ell_2 \dot{\theta}_2 \cos\theta_2, \tag{2.19c}$$
$$\dot{y}_2 = \ell_1 \dot{\theta}_1 \sin\theta_1 + \ell_2 \dot{\theta}_2 \sin\theta_2, \tag{2.19d}$$

$$\ddot{x}_1 = \ell_1(\ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1), \tag{2.20a}$$
$$\ddot{y}_1 = \ell_1(\ddot{\theta}_1 \sin\theta_1 + \dot{\theta}_1^2 \cos\theta_1), \tag{2.20b}$$
$$\ddot{x}_2 = \ell_1(\ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1) + \ell_2(\ddot{\theta}_2 \cos\theta_2 - \dot{\theta}_2^2 \sin\theta_2), \tag{2.20c}$$
$$\ddot{y}_2 = \ell_1(\ddot{\theta}_1 \sin\theta_1 + \dot{\theta}_1^2 \cos\theta_1) + \ell_2(\ddot{\theta}_2 \sin\theta_2 + \dot{\theta}_2^2 \cos\theta_2). \tag{2.20d}$$

Note that for all coordinates and derivatives, the first point mass is only influenced by ℓ_1 and (derivatives of) θ_1, while the second point mass is influenced by ℓ_1, ℓ_2 and (derivatives of) θ_1 and θ_2.

2.2.2 Dynamics

From Newton's second law and from Figure 2.2 we can find four equations with four unknown variables (after substituting equations (2.20)). Denoting $\theta_1 \equiv \theta_1(t)$, $\theta_2 \equiv \theta_2(t)$, $T_1 \equiv T_1(t)$ and $T_2 \equiv T_2(t)$ we start with the following equations:

$$m_1 \ddot{x}_1 = -T_1 \sin\theta_1 + T_2 \sin\theta_2, \tag{2.21a}$$
$$m_1 \ddot{y}_1 = T_1 \cos\theta_1 - T_2 \cos\theta_2 - m_1 g, \tag{2.21b}$$
$$m_2 \ddot{x}_2 = -T_2 \sin\theta_2, \tag{2.21c}$$
$$m_2 \ddot{y}_2 = T_2 \cos\theta_2 - m_2 g. \tag{2.21d}$$

We are not interested in variables T_1 and T_2 and will eliminate these in order to simplify the equations. First we add equations (2.21a) and (2.21c). Hereafter we also add equations (2.21b) and (2.21d) to obtain:

$$m_1 \ddot{x}_1 + m_2 \ddot{x}_2 = -T_1 \sin\theta_1, \tag{2.22a}$$
$$m_1 \ddot{y}_1 + m_2 \ddot{y}_2 = T_1 \cos\theta_1 - (m_1 + m_2) g. \tag{2.22b}$$

Adding equation (2.22a) multiplied by $\cos\theta_1$ to equation (2.22b) multiplied by $\sin\theta_1$ results in (2.23a). Similarly, by adding equation (2.21c) multiplied by $\cos\theta_2$ to equation (2.21d) multiplied by $\sin\theta_2$ we find (2.23b):

$$\cos\theta_1(m_1 \ddot{x}_1 + m_2 \ddot{x}_2) + \sin\theta_1(m_1 \ddot{y}_1 + m_2 \ddot{y}_2) = -(m_1 + m_2) g \sin\theta_1, \tag{2.23a}$$
$$m_2 \ddot{x}_2 \cos\theta_2 + m_2 \ddot{y}_2 \sin\theta_2 = -m_2 g \sin\theta_2. \tag{2.23b}$$

After substituting the equations (2.20) into (2.23a) we arrive at the following:

$$(m_1 + m_2)\ell_1\ddot{\theta}_1 + m_2\ell_2\ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2\ell_2\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) = -(m_1 + m_2)g \sin\theta_1. \tag{2.24a}$$

Similarly, substituting the equations (2.20) into (2.23b) yields:

$$m_2\ell_1\ddot{\theta}_1 \cos(\theta_1 - \theta_2) + m_2\ell_2\ddot{\theta}_2 - m_2\ell_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) = -m_2 g \sin\theta_2. \tag{2.24b}$$

Equations (2.24a) and (2.24b) are the equations of motion for the double pendulum. It can be verified that these equations are consistent with the results for the single pendulum. Suppose $m_2 = 0$. Then (2.24b) becomes the trivial equation $0 = 0$. Additionally, as expected, (2.24a) reduces to the equation of motion for a single pendulum, as in (2.6).

2.2.3 Consistency with conservation of total energy

Now we will show that equations (2.24a) and (2.24b) imply conservation of the total energy (kinetic energy plus potential energy) of the system. For a constant C for the conserved total energy we have:

$$\frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2) + m_1 g y_1 + m_2 g y_2 = C. \tag{2.25}$$

After filling in equations (2.18b), (2.18d) and equations (2.19) into (2.25) we obtain:

$$\frac{1}{2}(m_1 + m_2)\ell_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\ell_2^2\dot{\theta}_2^2 + m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) - (m_1 + m_2)g\ell_1 \cos\theta_1 - m_2 g\ell_2 \cos\theta_2 = C. \tag{2.26}$$

Differentiating this with respect to t we find:

$$\begin{aligned}
&(m_1 + m_2)\ell_1^2\ddot{\theta}_1\dot{\theta}_1 + m_2\ell_2^2\ddot{\theta}_2\dot{\theta}_2 \\
&+ m_2\ell_1\ell_2(\ddot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) + \dot{\theta}_1\ddot{\theta}_2 \cos(\theta_1 - \theta_2) - \dot{\theta}_1\dot{\theta}_2(\dot{\theta}_1 - \dot{\theta}_2) \sin(\theta_1 - \theta_2)) \\
&+ (m_1 + m_2)g\ell_1\dot{\theta}_1 \sin\theta_1 + m_2 g\ell_2\dot{\theta}_2 \sin\theta_2 = 0.
\end{aligned} \tag{2.27}$$

Notice that equation (2.27) is the same as equation (2.24a) multiplied by $\ell_1\dot{\theta}_1$ added to equation (2.24b) multiplied by $\ell_2\dot{\theta}_2$.

2.2.4 Rewriting to system of first-order differential equations

Now we will rewrite equations (2.24a) and (2.24b) to find expressions for $\ddot{\theta}_1$ and $\ddot{\theta}_2$. Subtracting equation (2.24b) multiplied by $\cos(\theta_1 - \theta_2)$ from equation (2.24a) yields:

$$\ddot{\theta}_1 = \frac{-m_2\ell_2\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - \frac{1}{2}m_2\ell_1\dot{\theta}_1^2 \sin(2\theta_1 - 2\theta_2) - (m_1 + m_2)g \sin\theta_1 + m_2 g \cos(\theta_1 - \theta_2) \sin\theta_2}{\ell_1(m_1 + m_2 \sin^2(\theta_1 - \theta_2))}. \tag{2.28}$$

Similarly, subtracting equation (2.24a) multiplied by $\cos(\theta_1 - \theta_2)$ from equation (2.24b) multiplied by $(m_1 + m_2)/m_2$ we find:

$$\ddot{\theta}_2 = \frac{(m_1 + m_2)\ell_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + \frac{1}{2}m_2\ell_2\dot{\theta}_2^2 \sin(2\theta_1 - 2\theta_2) - (m_1 + m_2)g(\sin\theta_2 - \cos(\theta_1 - \theta_2) \sin\theta_1)}{\ell_2(m_1 + m_2 \sin^2(\theta_1 - \theta_2))}. \tag{2.29}$$

To rewrite equations (2.28) and (2.29) to a first-order system we introduce the variables $p_1 = \theta_1$, $p_2 = \dot{\theta}_1$, $p_3 = \theta_2$ and $p_4 = \dot{\theta}_2$. Thus:

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \\ \dot{p}_4 \end{bmatrix} = \begin{bmatrix} p_2 \\ \frac{-m_2\ell_2(p_4)^2\sin(p_1-p_3)-\frac{1}{2}m_2\ell_1(p_2)^2\sin(2p_1-2p_3)-(m_1+m_2)g\sin p_1+m_2g\cos(p_1-p_3)\sin p_3}{\ell_1(m_1+m_2\sin^2(p_1-p_3))} \\ p_4 \\ \frac{(m_1+m_2)\ell_1(p_2)^2\sin(p_1-p_3)+\frac{1}{2}m_2\ell_2(p_4)^2\sin(2p_1-2p_3)-(m_1+m_2)g(\sin p_3-\cos(p_1-p_3)\sin p_1)}{\ell_2(m_1+m_2\sin^2(p_1-p_3))} \end{bmatrix}. \tag{2.30}$$

In order to compute $p_1 \equiv p_1(t)$, $p_2 \equiv p_2(t)$, $p_3 \equiv p_3(t)$ and $p_4 \equiv p_4(t)$ we need the initial conditions $\theta_1(0)$, $\dot{\theta}_1(0)$, $\theta_2(0)$ and $\dot{\theta}_2(0)$.

2.2.5 Numerical stability analyses

For the stability analysis of numerical solution methods for the double pendulum, similar steps can be followed like in Section 2.1.4. The system has already been rewritten to the first-order system (2.30). Now, the calculation of the Jacobian is more difficult, let alone the calculation of its eigenvalues. Even an attempt at computing the Jacobian in Wolfram Mathematica was not viable. The work to exactly compute the eigenvalues of the Jacobian is too extensive and difficult for this project. Therefore the Jacobian and its eigenvalues are computed approximately (numerically) as part of the numerical simulation instead of analytically.

Using RK4 as time integration method, each time-step, the four eigenvalues must satisfy $|Q(\lambda\tau)| < 1$ according to (2.15) to maintain stability.

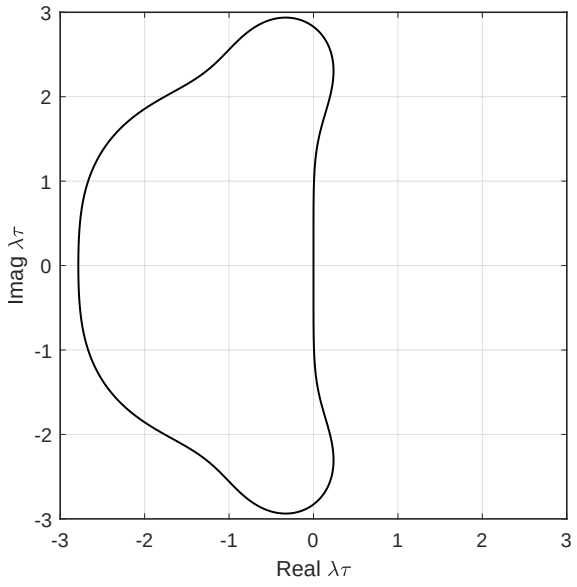

Figure 2.3: Stability region of RK4, where the inside is numerically stable and the outside is numerically unstable.

In Figure 2.3 we see the stability region of RK4. Here it can be seen that part of the imaginary axis $[-\alpha i, \alpha i]$, where $\alpha \in \mathbb{R}$, lies within the stability region and the stability region may contain some eigenvalues with positive real components. For an imaginary eigenvalue λ, for which the

eigenvalue is of the form $\lambda = bi$, where $b \in \mathbb{R}$, we find:

$$\begin{aligned}
|Q(\lambda\tau)| =& |1 + \lambda\tau + \frac{1}{2}(\lambda\tau)^2 + \frac{1}{6}(\lambda\tau)^3 + \frac{1}{24}(\lambda\tau)^4| \\
&= |1 + bi\tau + \frac{1}{2}(bi\tau)^2 + \frac{1}{6}(bi\tau)^3 + \frac{1}{24}(bi\tau)^4| \\
&= \sqrt{\left(1 - \frac{1}{2}b^2\tau^2 + \frac{1}{24}b^4\tau^4\right)^2 + \left(b\tau - \frac{1}{6}b^3\tau^3\right)^2} \\
&= \sqrt{1 - \frac{1}{72}b^6\tau^6 + \frac{1}{576}b^8\tau^8}.
\end{aligned}$$

Note that $-\frac{1}{72}b^6\tau^6$ is always negative and $\frac{1}{576}b^8\tau^8$ always positive. Therefore $|Q(\lambda\tau)| = |1 + \lambda\tau + \frac{1}{2}(\lambda\tau)^2 + \frac{1}{6}(\lambda\tau)^3 + \frac{1}{24}(\lambda\tau)^4| \leq 1$ when:

$$\begin{aligned}
&\frac{1}{576}b^8\tau^8 \leq \frac{1}{72}b^6\tau^6 \\
\implies\ & b^2\tau^2 \leq 8 \\
\implies\ & \tau \leq \frac{\sqrt{8}}{b}.
\end{aligned} \tag{2.31}$$

Assuming the eigenvalues are imaginary, we take the following for the time-step:

$$\tau \leq \frac{\sqrt{8}}{\max|b|}. \tag{2.32}$$

So we let the eigenvalue with the largest imaginary part determine the upper bound for the time-step that should be used.

Chapter 3

Results

In the upcoming chapter various results are shown and explained. In all cases $m_1 = m_2 = 1$ kilogram and $\ell_1 = \ell_2 = 1$ meter. Additionally in all cases $\dot{\theta}_1(0) = 0$ rad/s and $\dot{\theta}_2(0) = 0$ rad/s. These variables are chosen and used throughout the entire chapter. This choice of variables already provides a satisfactory amount of results and discussion for this report. It should be noted that for most simulations with plots, videos can be found in Appendix A.

3.1 Simulation with Forward Euler

In the simulation, the previously discussed methods will be tested and briefly compared. First, we will look at the Forward Euler method. For this method, there does not exist a time-step for which the simulation will be numerically stable, due to the fact that the stability region for Forward Euler does allow for any purely imaginary eigenvalue. The total energy of the system, as used in (2.25), should be conserved. In Figure 3.1 we see how swiftly the energy changes compared to its initial value.

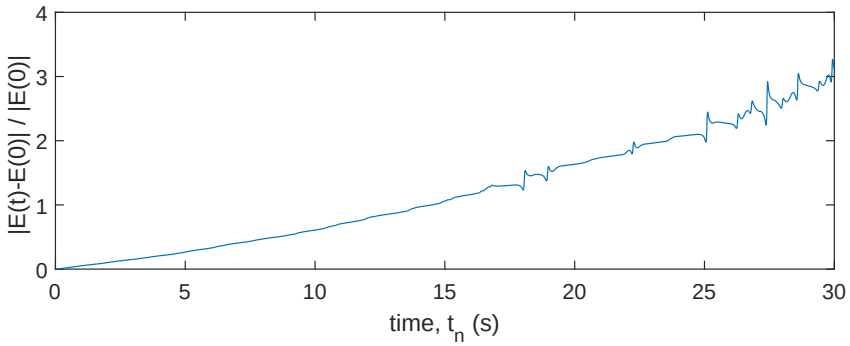

Figure 3.1: Evolution of relative energy error for Forward Euler, $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, $\tau = 10^{-2}$ s.

In Figure 3.1 we see the development of the relative error of the total energy of the double pendulum for Forward Euler. It can be seen that the error rapidly increases and after some time we can see a few growing spikes. For Forward Euler the numerical solution becomes increasingly more energetic for the simulation of the double pendulum and the pendulum will start moving faster. The simulation has difficulty coping with the increased speed of the pendulum and it starts making loopings more often, which the simulation struggles with, hence the spikes. Considering that we have a system without any friction forces, Forward Euler is not suited to simulate a double pendulum. Any attempt to demonstrate chaos for the double pendulum with Forward Euler is not trustworthy, since differences in trajectories could be ascribed to numerical instability.

3.2 Simulation and simulated eigenvalues of RK4

A better suited numerical method is RK4. This method is capable of maintaining stability for a wider range of eigenvalues, among which purely imaginary eigenvalues. To show results, frequently a fixed time-step will be used. This time-step is taken lower than the lowest time-step that is found using an adaptive time-step using (2.32). An example of the changes in the total energy using this adaptive time-step for RK4 is given in Figure 3.2.

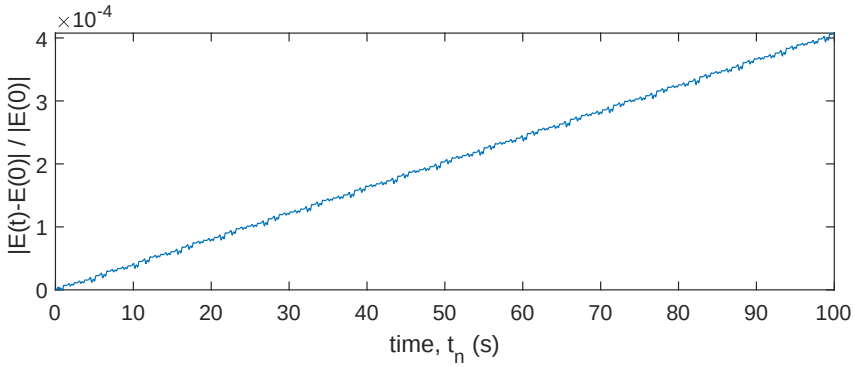

Figure 3.2: Evolution of relative energy error for RK4, $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, τ adaptive.

For RK4, it can be seen that the error in energy grows slowly, steadily and approximately linearly. This implies that the perturbation errors in the simulation are somewhat similar for every time-step. Since the angles of the initial solution are rather small, the time-step does not fluctuate strongly. The angles do not become significantly larger or smaller than these starting angles after running the simulation for a while.

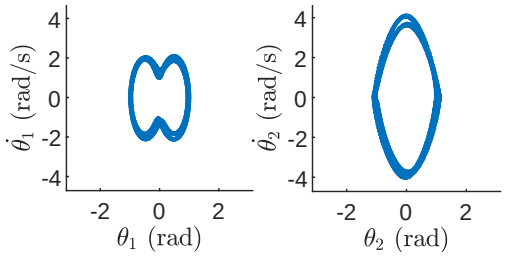

Figure 3.3: $\dot{\theta}_1(t)$ plotted against $\theta_1(t)$ (left) and $\dot{\theta}_2(t)$ plotted against $\theta_2(t)$ (right) for $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, τ adaptive.

In Figure 3.3 we see two plots with the angles versus angular momentum. It can be seen that, around $\theta_1(t) = 0$ and $\theta_2(t) = 0$, there is a local minimum for $\dot{\theta}_1(t)$ and a local maximum for $\dot{\theta}_2(t)$. This is caused by the fact that $\theta_2(t)$ cannot fully keep up with $\theta_1(t)$ at this point. In other words, $\theta_1(t)$ is faster at $\theta_1(t) = 0$, because its trajectory, mainly influenced by gravity, is shorter than that of $\theta_2(t)$. First, the second point mass slows down the first point mass, approximately until the point where $\theta_1(t) = 0$ and $\theta_2(t) = 0$ is reached, after which the fast second point mass pulls along the first point mass stronger. These two plots show a clear, structured region of motion. In situations where chaos occurs, this region is not well defined.

The eigenvalues that are computed are not always completely imaginary. Still, the imaginary component generally is significantly larger than the real component. We will now look at the ratio of the real and imaginary components of the eigenvalues to show this. This ratio is defined as: $|\frac{\mathrm{Re}(\lambda)}{\mathrm{Im}(\lambda)}|$.

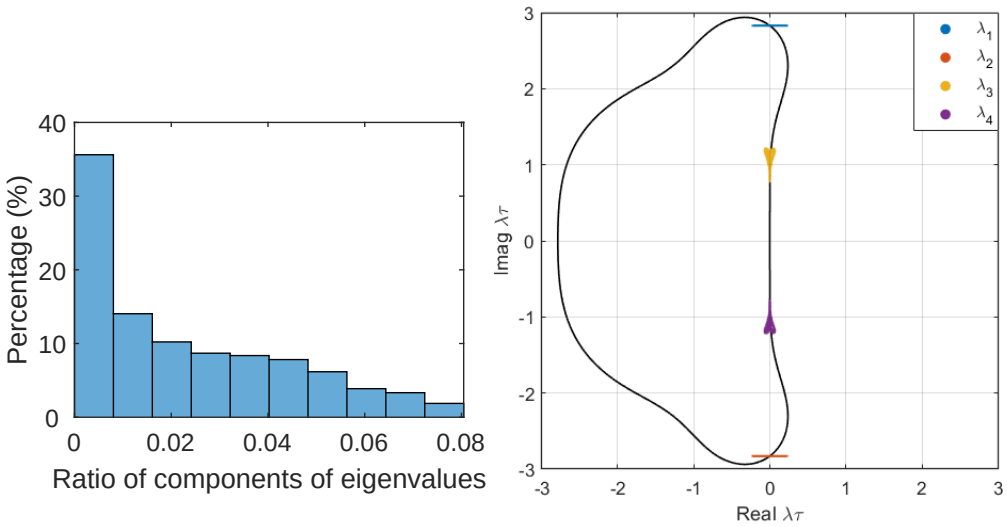

Figure 3.4: Left: Histogram of the ratio of the components of the eigenvalues.
Right: Stability range of RK4 plotted together with all computed eigenvalues as dots.
$\theta_1(0) = 1$ rad. $\theta_2(0) = 1$ rad, τ adaptive, $t \in [0, 100]$ s.

In Figure 3.4, on the left we see a histogram of the aforementioned ratio of the components of all the eigenvalues used in the entire time-step process. The percentage defines the probability ($\times 100\%$) that a ratio falls into a bar of the histogram. On the right we see the stability region for RK4 as in Figure 2.3, plotted together with all eigenvalues multiplied by their respective values of τ for a simulation to $t = 100$ s.

As we can see in the histogram, for this initial solution, the ratio $|\frac{\mathrm{Re}(\lambda)}{\mathrm{Im}(\lambda)}|$ does not become much larger than ≈ 0.08 and is much smaller in most cases. In the plot on the right we are able to see the adaptive time-step doing its work: Every largest imaginary component of the eigenvalues gets 'pulled' into the stability region. Since only the imaginary component is considered, a few eigenvalues with a small real component can fall slightly outside of the stability region. Because the time-step used for later simulations will be strictly smaller than any adaptive time-step used here, this will not be an issue for numerical stability, as long as the angles of the initial solution are sufficiently small. In the simulation we accept that the real component is ignored and only the largest imaginary component determines the next time-step. Also note that λ_1 is the complex conjugate eigenvalue of λ_2 and λ_3 is the complex conjugate eigenvalue of λ_4, as it should be.

We will now consider a different initial condition:

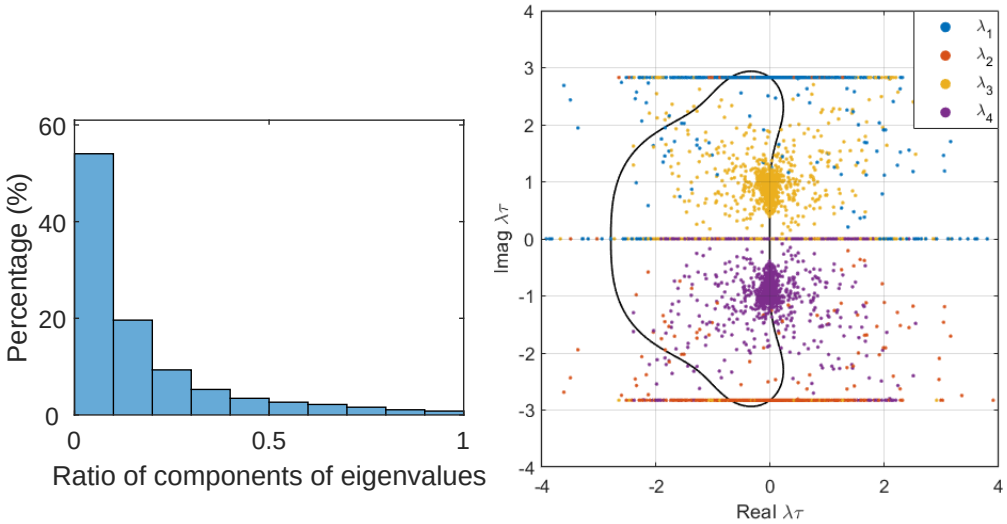

Figure 3.5: Left: Histogram of the ratio of the components of the eigenvalues.
Right: Stability range of RK4 plotted together with all computed eigenvalues as dots.
$\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad, τ adaptive, $t \in [0, 100]$ s.

In Figure 3.5, on the left we see a histogram of the ratio of the components of the eigenvalues, but now for $\theta_2(0) = 2$ instead of $\theta_2(0) = 1$. On the right we see the stability region for RK4, again plotted together with all eigenvalues multiplied by their respective τ for a simulation to $t = 100$. In Figure 3.5 it can be seen that for an initial condition with one larger starting angle than before, the ratio of the real over the imaginary components of the eigenvalues can become much larger. In fact, approximately 12% of the eigenvalues had a larger real component than its imaginary component and fell outside of the histogram shown in Figure 3.5. For these cases, even a completely real eigenvalue was not uncommon.

In the plot on the right, we can still see some clusters around the same location as in Figure 3.4, but the eigenvalues are much more spread out. A considerable amount falls far outside of the stability region for RK4. This suggests that RK4 becomes unstable more easily for large angles in the pendulum. Even the single pendulum had angles for which numerical stability was uncertain, so this is not unexpected to happen for the double pendulum as well. RK4 is still good to use for (non-stiff) problems with negative real eigenvalues. Nevertheless, this means that for initial solutions with large angles, the time-step needs to be considerably smaller to even approach an accurate simulation when compared to the situation in Figure 3.4. Some eigenvalues will never fall into the stability region (no matter how small τ), but can get extremely close.

3.3 Simulations with various amounts of double pendulums and initial solutions

3.3.1 Energy errors for halving τ

From now on we will fix τ (which will be equal to 10^{-2} in most cases). We start with a plot of the energy with the non-chaotic initial solution $\theta_1(0) = \theta_2(0) = 1$ rad. Here we can compare the ratios of the variation of the energy and the reference value $E(0)$ for τ, $\tau/2$ and $\tau/4$. As can be seen in the plots in Figure 3.6, the energy error becomes approximately 2^5 times as small when halving the time-step. This will be further elaborated on in Section 3.4. There is also relatively more noise for the energy error when the time-step gets smaller, although the absolute noise of the energy error still becomes smaller.

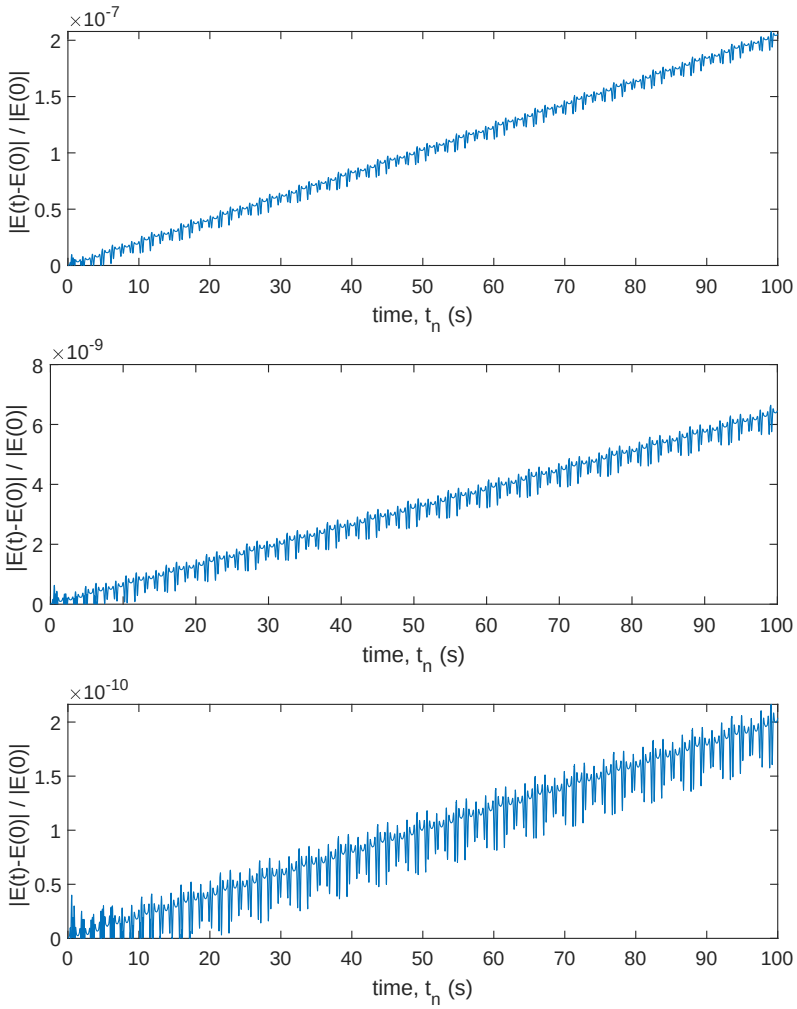

Figure 3.6: Evolution of relative energy errors for $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, $\tau = 10^{-2}$ s (top), $\tau/2 = 5.0 \cdot 10^{-3}$ s (middle) and $\tau/4 = 2.5 \cdot 10^{-3}$ s (bottom).

3.3.2 Chaotic

Now, how predictable is the trajectory of the double pendulum? First we will look at three simulations of the double pendulum with a small perturbation ϵ for the initial solution in $\theta_1(0)$. For each pendulum, visible by their colour, the perturbation is implemented as follows:

- Blue: $\theta_1(0) = \theta_1(0)$ rad.
- Red: $\theta_1(0) := \theta_1(0) - \epsilon$ rad.
- Yellow: $\theta_1(0) := \theta_1(0) - 2\epsilon$ rad.

In Figure 3.7 we see the three double pendulums plotted together at different time instances with the corresponding relative energy errors in Figure 3.8. After some time, the double pendulums start to follow completely different trajectories, as can be seen in the last plot on the right in Figure 3.7. The evolution of the total energy is also different for the three pendulums. It becomes unrecognizable that the pendulums had approximately the same initial solutions.

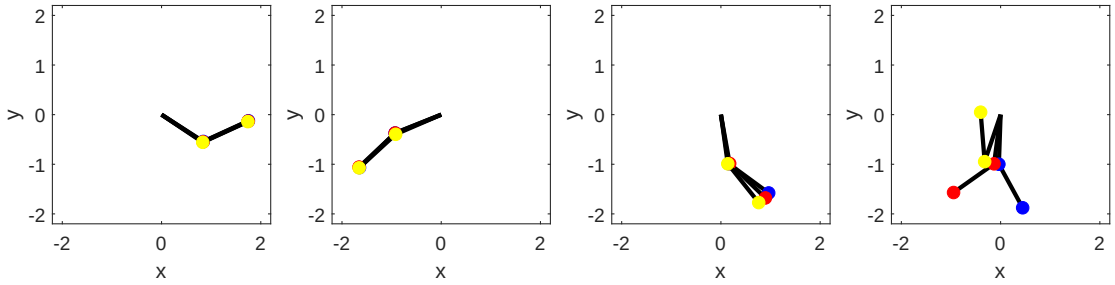

Figure 3.7: Four snapshots of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad. Snapshots at (left to right): $t = 0$ s, $t = 7.5$ s, $t = 15$ s, $t = 22.5$ s.

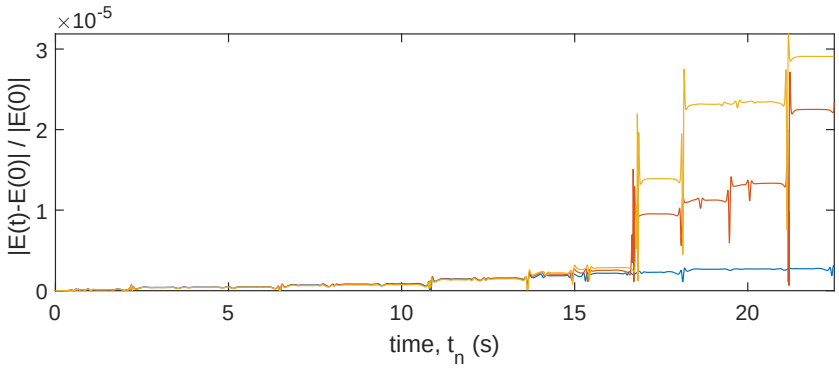

Figure 3.8: Evolution of relative energy errors of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad.

3.3.3 Chaotic with $\epsilon = 10^{-15}$ rad

In Figures 3.9 and 3.10 it can be seen that the double pendulum behaves significantly differently after some time for a small perturbation ϵ in initial solution $\theta_1(0)$, even for $\epsilon = 10^{-15}$ rad. This implies that, until a certain degree that the machine precision of Matlab allows, no matter how small ϵ, for this starting position the double pendulum will always become chaotic. The larger the perturbation, the sooner the differences occur.

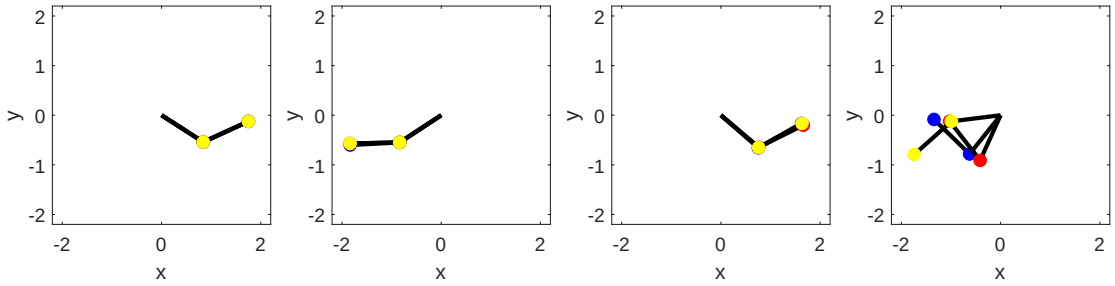

Figure 3.9: Four snapshots of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-15}$ rad. Snapshots at (left to right): $t = 0$ s, $t = 80$ s, $t = 90$ s, $t = 100$ s.

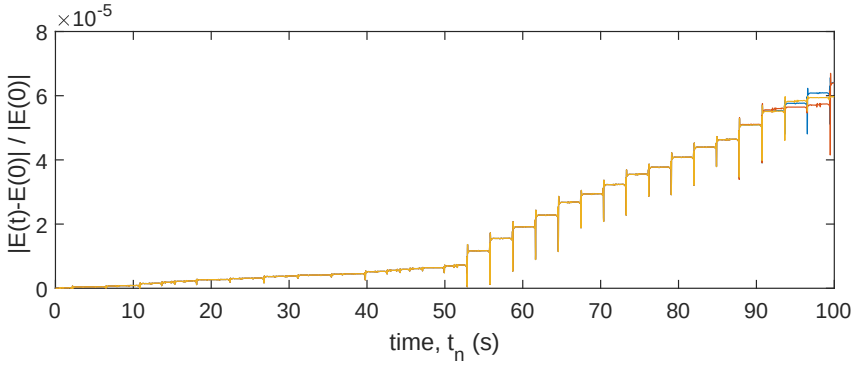

Figure 3.10: Evolution of relative energy errors of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-15}$ rad.

3.3.4 Non-chaotic (yet)

In Figure 3.11 it can be seen that for a system with a lower total amount of energy, still different angles for the initial solution, $\theta_1(0) \neq \theta_2(0)$, and $\epsilon = 10^{-2}$ rad, the three pendulums also start to differ in motion, but do not yet depart from each other. In Figure 3.12 we see the relative energy errors of the pendulums from Figure 3.11. The energy errors stay together to some extent, for the time considered. We do not (yet) see the same chaotic behaviour as in Figures 3.7 and 3.9. The relative energy errors slowly depart from each other, while staying fairly linear. In the case of a chaotic situation, these errors would have spiked much more strongly and more abruptly.

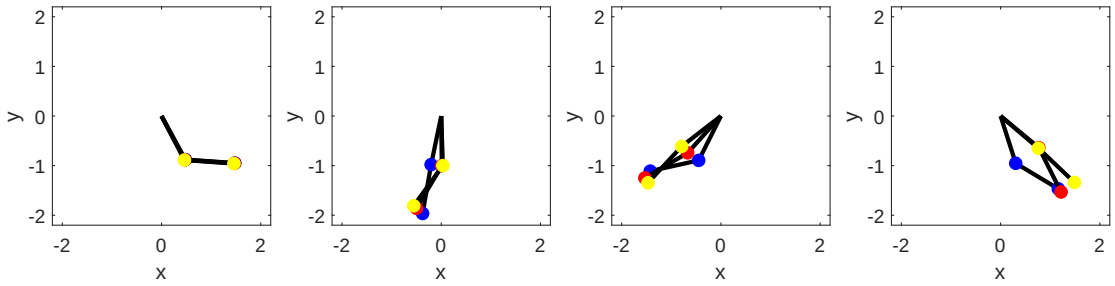

Figure 3.11: Four snapshots of three double pendulums for $\theta_1(0) = 0.5$ rad, $\theta_2(0) = 1.5$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad. Snapshots at (left to right): $t = 0$ s, $t = 20$ s, $t = 100$ s, $t = 200$ s.

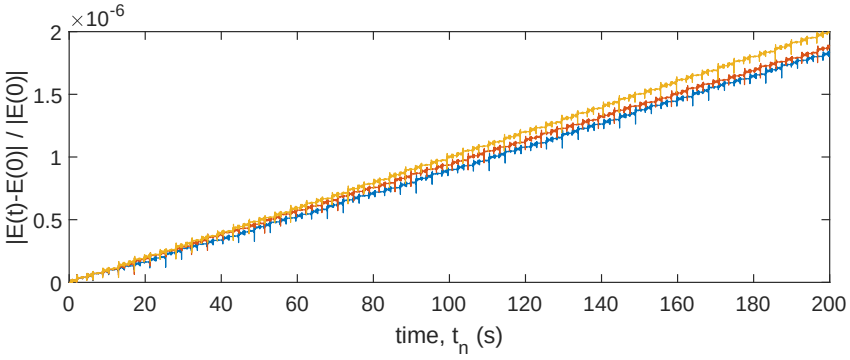

Figure 3.12: Evolution of relative energy errors of three double pendulums for $\theta_1(0) = 0.5$ rad, $\theta_2(0) = 1.5$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad.

3.3.5 Non-chaotic

In Figure 3.13 we see three double pendulums, just as in Figure 3.6, with equal angles for the initial solution for $\theta_1(0)$ and $\theta_2(0)$ with the corresponding relative energy errors in Figure 3.14. It can be seen that the double pendulum does not show significant differences after a long time for these initial solutions. $\theta_1(t)$ and $\theta_2(t)$ stay close to each other and only differences in phase between the three pendulums seem to occur for increasing t, contrary to the situations in Figures 3.7 and 3.9.

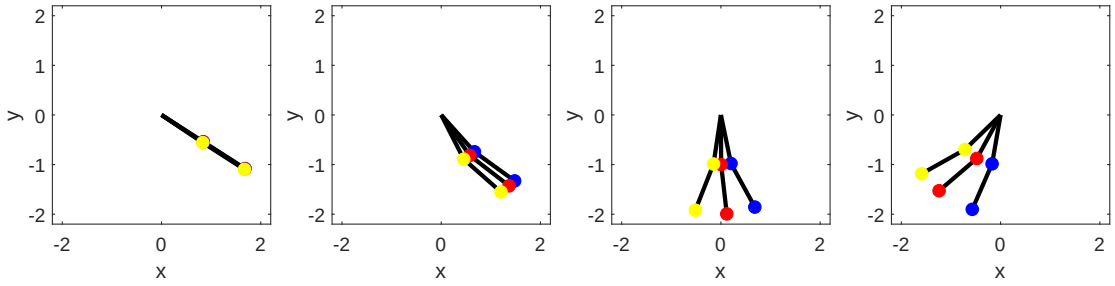

Figure 3.13: Four snapshots of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad. Snapshots at (left to right): $t = 0$ s, $t = 100$ s, $t = 200$ s, $t = 300$ s.

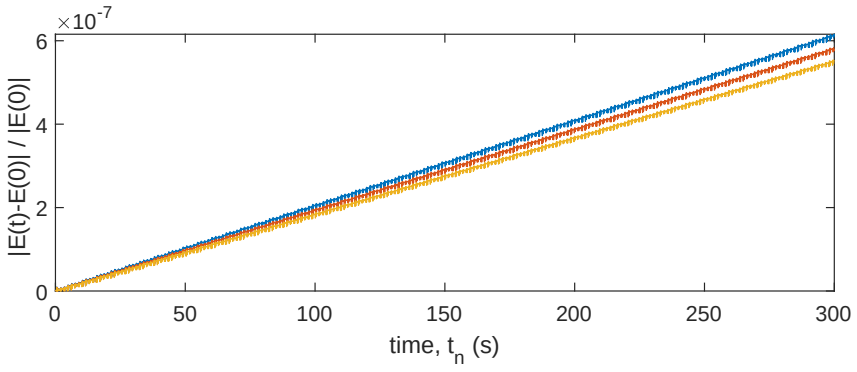

Figure 3.14: Evolution of relative energy errors of three double pendulums for $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad, $\tau = 10^{-2}$ s, $\epsilon = 10^{-2}$ rad.

3.3.6 Checks for the simulation

In Figure 3.15 we can see two mirrored pendulums that follow an exactly identical mirrored path for up to at least $t = 10^5$ s. For comparison, this is longer than a full day on earth. Notice that a perturbation of $\epsilon = 10^{-15}$ rad already shows chaotic behaviour after 100 seconds in Figure 3.9. The results in Figure 3.15 imply that the rounding errors are either extremely small when using Matlab, or, more likely, Matlab does not make a distinction when rounding values with an opposite sign in this simulation.

A similar simulation has been performed on two double pendulums with exactly the same initial solution. Again, this results in two double pendulums that move exactly the same throughout the entire simulation. This also happens when performing the same simulation twice in a row. So, while the simulation usually has small errors or chaotic behaviour, as long as the initial solution is equivalent (or mirrored), two double pendulums have the exact same trajectory. One could say that the simulation is able to make the exact same mistakes and errors repeatedly. It should be noted that this is not confirmed for the simulation for $t \to \infty$, as the simulation has only been run up to $t = 10^5$ s.

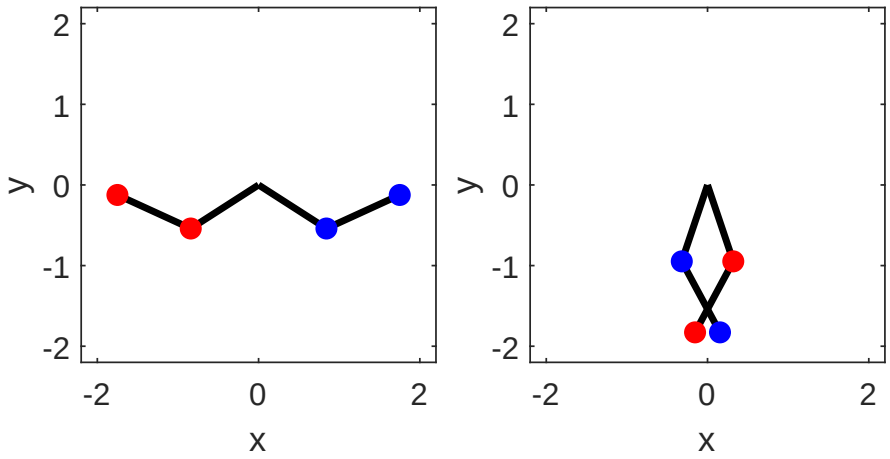

Figure 3.15: Two snapshots of two mirrored double pendulums for $\theta_1(0) = \pm 1$ rad, $\theta_2(0) = \pm 2$ rad, $\tau = 10^{-2}$ s. Snapshots at (left to right): $t = 0$ s, $t = 10^5$ s.

In Figure 3.16 we see one double pendulum starting at 180 degrees. This is the initial solution for which the system contains the most energy out of every initial solution where $\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$ rad. We will call this reference point $E_{\max}$. The double pendulum should theoretically stay in this position. However, due to rounding errors in Matlab, at approximately $t \approx 23.5$ rad the pendulum tips over.

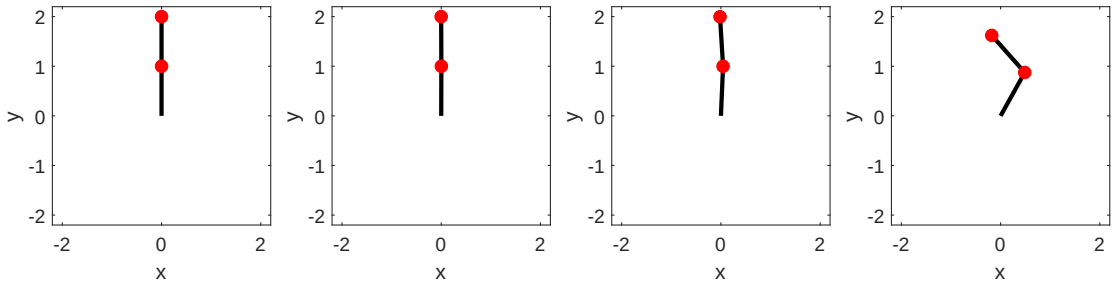

Figure 3.16: Four snapshots of a double pendulum for $\theta_1(0) = \pi$ rad, $\theta_2(0) = \pi$ rad, $\tau = 10^{-2}$ s. Snapshots at (left to right): $t = 23$ s, $t = 23.5$ s, $t = 24$, $t = 24.5$.

3.3.7 Change in energy for five initial solutions

In Figure 3.17 we see the change in energy relative to $E_{\max}$ in two plots that are essentially the same, except for the y-axis. Generally, the larger the angles in the initial solution, the faster the error increases, although the differences are not the same each time. While the change in energy is non-linear for small time intervals, it is approximately linear in the long run. In terms of energy, for $\theta_1(0) = \theta_2(0)$, RK4 seems to be suitable for this simulation, since the change in energy relative to $E_{\max}$ takes extremely long to become significantly larger.

The two smallest angles have no chaotic properties, but the three largest do, which results in a larger error as we can see in the right plot. This further substantiates that the simulations work better for non-chaotic situations.

The same simulation has also been performed for a time-step of $\tau = 10^{-4}$ s. The corresponding plot contains some anomalies (downward spikes) and has therefore been placed in Appendix B. Yet, the plot shows errors that decrease slowly occasionally. Possibly this time-step is sufficiently low to ensure numerical stability, but the spikes may also indicate that Matlab does not handle extremely small time-steps well or that the noise is simply supposed to become relatively larger.

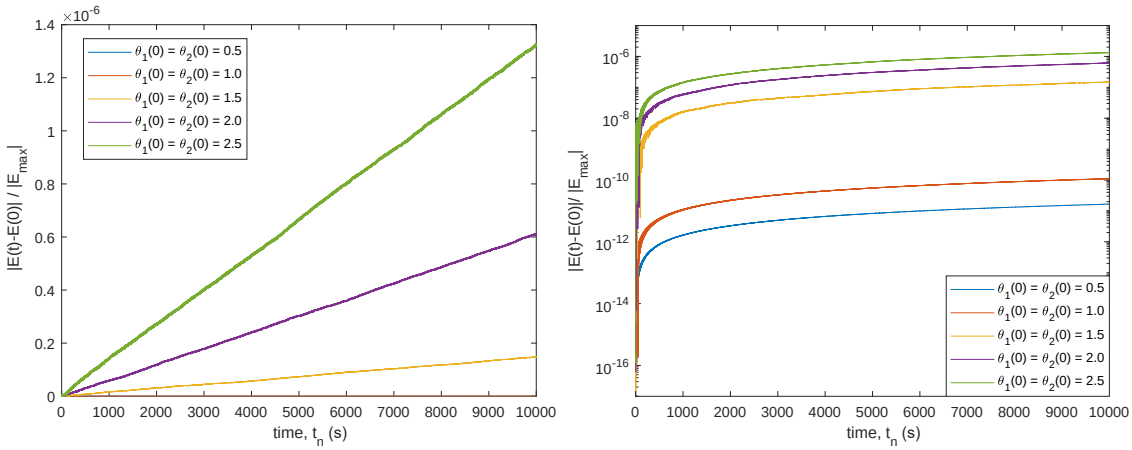

Figure 3.17: Two plots of the change in energy relative to $E_{\max}$ for five initial solutions (in radians) with different y-axes for $\tau = 10^{-3}$ s. Left: linear y-axis. Right: logarithmic y-axis.

3.4 Richardson extrapolation

We combine the RK4 method with Richardson extrapolation to verify the order of accuracy of the method. Let p^τ denote the numerical approximation of $p(t_n)$ with n steps of size τ. For an r-th order convergent numerical method (with $2n$ steps for $p^{\tau/2}$ and $4n$ steps for $p^{\tau/4}$) we have:

$$\left(\frac{1}{2}\right)^r \approx \frac{p^{\tau/4} - p^{\tau/2}}{p^{\tau/2} - p^\tau}. \tag{3.1}$$

We now run the simulation multiple times for different halved time-steps. In this case we look at the initial solution where $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad and we look at the differences of $\theta_1(t)$ at $t = 1$ s. For $\tau = 0.01$ s (and consequently $\tau/2 = 0.005$ s and $\tau/4 = 0.0025$ s), we find the following orders of accuracy:

τ (s)	r
0.01	3.8615
0.005	3.9360
0.0025	3.9691
0.00125	3.9851

Table 3.1: Orders of accuracy determined through Richardson extrapolation with initial solution $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad ending at $t = 1$ s.

It can be observed that the difference between 4 and r approximately halves after each halved time-step in this example.

RK4 should be fourth-order accurate, so these values for r that get increasingly closer to 4 for smaller time-steps confirm that the method used in the simulation is indeed fourth-order convergent. This indicates that for simulations with angles for the initial condition that are relatively short (with an ending time at $t = 1$ s), halving the time-step makes a numerical method about 16 times more accurate. Note that this is an arbitrary example. Nonetheless, looking at the differences for $\theta_2(t)$ or even the angular momentum yields similar results.

When choosing another ending time, for instance $t = 100$ s with the same initial solution, we see that convergence to fourth-order accuracy is slower:

τ (s)	r
0.01	3.1031
0.005	3.6998
0.0025	3.8705
0.00125	3.9453

Table 3.2: Orders of accuracy determined through Richardson extrapolation with initial solution $\theta_1(0) = 1$ rad, $\theta_2(0) = 1$ rad ending at $t = 100$ s.

If we take another initial solution, such as when $\theta_1(0) = 1$ rad and $\theta_2(0) = 2$ rad, we see that with an ending time of $t = 1$ s, the order of accuracy is still close to 4:

τ	r
0.01	3.9271
0.005	3.9667
0.0025	3.9840
0.00125	3.9949

Table 3.3: Orders of accuracy determined through Richardson extrapolation with initial solution $\theta_1(0) = 1$ rad, $\theta_2(0) = 2$ rad ending at $t = 1$ s.

Yet, we know that the double pendulum can become chaotic quickly for this initial solution. Computing the accuracy for the same situation, but now up to and including $t = 100$ s, we find accuracies not even remotely close to 4 that do not make sense. To clarify, some values for r become complex because the differences are so large. This is caused by the chaotic behaviour. While the earlier examples had fairly similar trajectories for the double pendulum, the last example has not and the differences in $\theta_1(t)$ cannot be compared the same way. The double pendulum can be considered to not be fourth-order convergent for RK4 in the long run in terms of angles of the pendulum. This information is useful and it may be used as an indicator or as a measure for chaos.

Now let us re-evaluate Figure 3.17.

$\theta_1(0) = \theta_2(0)$ (rad)	r
0.5	5.1847
1.0	4.9506
1.5	4.8848
2.0	4.9664
2.5	4.9826

Table 3.4: Orders of accuracy determined through Richardson extrapolation for different initial solutions with $\tau = 10^{-3}$ ending at $t = 10^4$ s.

In Table 3.4 we can see orders of accuracy for the energy of the system for different initial solutions as in Figure 3.17. Only τ, $\tau/2$ and $\tau/4$ are computed, so only one order of accuracy is shown per initial solution. Contrary to the previous tables, a large ending time has been taken to make sure that the energy error was not influenced significantly by noise. While the order of accuracy generally tends to converge to 4 for an angle or angular momentum, these results seem to indicate that the energy converges with an order of accuracy of 5. Thus every time the time-step halves, the energy error becomes about 32 times smaller.

Chapter 4

Conclusion

First, the single pendulum was approached analytically. Most of the assumptions that were made were necessary to simplify the problem, but could also be removed to make the problem more realistic, at the cost of being more complicated. Note however that an assumption like air friction could have proven to make the problem less sensitive to chaos and hence simpler. A non-linear differential equation (2.6) was derived and linearized. Then numerical stability was checked, which showed that even for the single pendulum, there would already be a constraint for stability, like an upper bound for the time-step, which had an extra restriction based on linearization. Also, RK4 appeared more promising than Forward Euler.

The analysis of the double pendulum requires more steps and is more extensive than that of the single pendulum. A first-order system could be constructed, but computing the eigenvalues of the Jacobian appeared to be cumbersome. A more time-consuming approach using more computations to find these eigenvalues could have provided more extensive analytical results to be used in the simulation for the double pendulum. With the assumption that every eigenvalue was completely imaginary, a simple upper bound for the time-step was still obtained, to be used for an adaptive time-step.

Forward Euler and RK4 have been the only numerical methods that have been used in the simulation. While most methods, like Backward Euler or RK2, would not yield more accurate results than RK4, other methods, like Adams–Bashforth, Adams–Moulton or a symplectic method (e.g. Verlet integration) could have been more accurate or faster. There could be different or more sophisticated methods, with a more fitting stability region.

While the eigenvalues can be pulled inside the stability region of RK4 for simple non-chaotic situations of the double pendulum (Figure 3.4), this is certainly not possible for every situation (Figure 3.5). Further research could expand on the (dis)advantages and specific properties of the stability region. RK4 is generally good to use for non-stiff problems with negative eigenvalues. However, the real components of the eigenvalues are large in some cases, which should have affected the chosen time-step, but this is not the case. Thus ignoring the real component of the eigenvalue likely had a negative effect on the accuracy of the simulation.

The plots in Section 3.3 show that for the double pendulum chaos usually occurs quickly or not at all. Even the smallest differences in starting position can result in chaos, although later than for large differences. Further research could investigate whether similar results are found for small perturbations on θ_2 or other properties of the double pendulum. Non-chaotic situations occurred, where the double pendulum behaved similarly for similar starting positions, but also where the double pendulums move differently, while staying together to some extent. Further research might yield specific conditions for which chaos occurs or not.

Determining the exact same starting situation for the simulation as that of an experiment with a double pendulum in physical reality would most likely be impossible without any errors. Therefore the predictability of the double pendulum is only short-lived in chaotic situations, since even the smallest errors in starting position have been demonstrated to already cause a different trajectory quickly. Further research could learn whether predicting a chaotic double pendulum is possible at all. In non-chaotic situations, we are able to predict the actual trajectory of the double pendulum

to some extent. Perturbation errors produce small alterations in the trajectory, but in non-chaotic situations, the pendulum is predictable provided that the time-step is sufficiently small.

Our simulation is consistent and makes possible errors consistently, such that each equivalent starting situation yields the exact same results repeatedly for multiple simulations. The simulation is not perfect in an equilibrium, as Figure 3.16 shows that the double pendulum tips over. How and why this happens at this point in time could be investigated further. The simulation has been verified in terms of order of accuracy. For RK4, convergence to fourth-order accuracy cannot be found in every chaotic situation. The smaller the time-step, the more accurate the simulation. Lastly, in terms of energy, which always changed approximately linearly in the long-term for non-chaotic situations, the order of accuracy of RK4 appears to be 5. Further research might learn why this occurs.

We conclude that predicting the motion of a double pendulum is feasible to some extent. The double pendulum still has many research questions. A tool has been made and showcased that it is able to explore some of the mathematical properties of the double pendulum. Even a simple system like a double pendulum can exhibit substantial chaos. It is imperative to understand how to apply numerical methods in such systems before applying them to more complex ones.

Bibliography

[1] Wikipedia contributors, "Pendulum clock — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Pendulum_clock&oldid=1028897143, 2021. [Online; accessed 4-August-2021].

[2] S. Whitestone, "Christian Huygens lost and forgotten pamphlet of his pendulum invention," *Annals of Science*, vol. 69, no. 1, pp. 91–104, 2012.

[3] A. M. Calvão and T. J. Penna, "The double pendulum: A numerical study," *European Journal of Physics*, vol. 36, no. 4, 2015.

[4] M. Gitterman, *The Chaotic Pendulum*. World Scientific, 2010.

[5] I. Newton, B. Cohen, A. M. Whitman, and J. T. A. T. T. Budenz, *The Principia : Mathematical Principles of Natural Philosophy*. University of California Press, 2016.

[6] C. Vuik, F. J. Vermolen, M. B. van Gijzen, and M. J. Vuik, *Numerical Methods for Ordinary Differential Equations*. Delft Academic Press, 2015.

Appendix

Appendix A: Videos of simulations

In this appendix, some videos of the simulations of the double pendulum can be found. All videos can be accessed via Google Drive:
https://drive.google.com/drive/folders/1Bv5dFgiDSPE5J4o6IBbfZ-B51QCdRJod?usp=sharing

They can also be accessed via Youtube through the channel with the username: I.S. Dams. All videos were published on the 6th of October 2021. This channel can be accessed through:
https://www.youtube.com/channel/UCJYNp-Bq-Y29QR0MyvGwzYQ
Below, the most important features of each simulation, relevant figures and a clickable link of each video can be found:

Forward Euler. Figure 3.1. https://youtu.be/Cfiug7aCBYM

RK4, adaptive τ. Figures 3.2 & 3.3. https://youtu.be/-P1uU8RCoHE

RK4, non-chaotic. Figure 3.6 (only top plot). https://youtu.be/o79LPz37G-s

RK4, 3 pendulums, chaotic. Figures 3.7 & 3.8. https://youtu.be/fs-IdCVttXY

RK4, 3 pendulums, extremely small ϵ, chaotic. Figures 3.9 & 3.10. https://youtu.be/p4CXSmbFzGA

RK4, 3 pendulums, non-chaotic (yet). Figures 3.11 & 3.12. https://youtu.be/f7RDJaYEb1E

RK4, 3 pendulums, non-chaotic. Figures 3.13 & 3.14. https://youtu.be/BEpZY-l8hJE

RK4, 180 degrees. Figure 3.16. https://youtu.be/6sr3OLUC0sk

Appendix B: Figure of change in relative energy for $\tau = 10^{-4}$

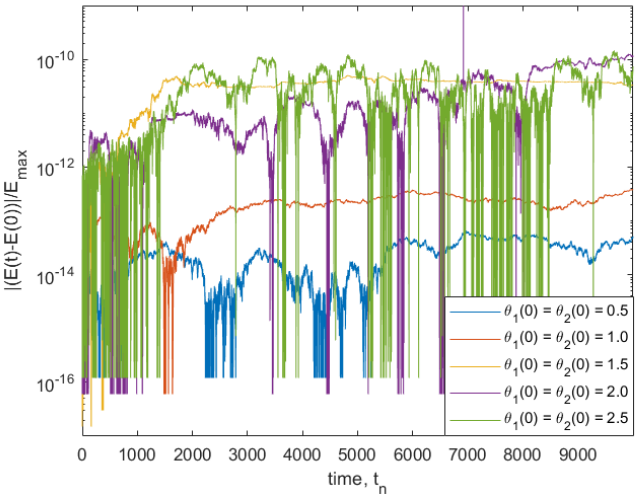

Figure 5.1: A plot of the change in energy relative to E_{max} for five starting positions (in radians) with a logarithmic y-axis for $\tau = 10^{-4}$ s.

Appendix C: Main code

Main Matlab code (version R2019b) for the simulation.

```
clear all
% model parameters
g = 9.81;
m1 = 1;
m2 = 1;
l1 = 1;
l2 = 1;

% time parameters
tau_adaptive = false;
tau_initial = 10^-2;
Tstart = 0;
Tend = 100;
Tstartfig = 0;

% pendulum simulation parameters
epsilon = 10^-2;
%epsilon = 0.5;
simulations = 3;
richardson = false; %1,/2,/4,/8,/16,/32,/64,/128
if tau_adaptive == true || richardson == true
    simulations = 1;
    epsilon = 0;
end

% miscellaneous parameters
make_video = false;
picture1 = 100;
picture2 = 200;
picture3 = 300;

%for index = [1,2,4,8,16,32,64,128]
for index = [1]
    N = Tend/tau_initial*index;
    N2 = ((Tend-Tstartfig)/tau_initial)*index;
    tau = tau_initial/index;

    for i = 1:simulations
        w1(i) = 1 - (i-1)*epsilon; % theta1
        w2(i) = 1; % theta2
        % w1(i) = 0.5 + (i-1)*epsilon;
        % w2(i) = 0.5 + (i-1)*epsilon;
        w3(i) = 0; % theta1'
        w4(i) = 0; % theta2'
        u1{i} = zeros(N+1,1);
        u2{i} = zeros(N+1,1);
        v1{i} = zeros(N+1,1);
        v2{i} = zeros(N+1,1);
        energy{i} = zeros(N+1,1);
        u1{i}(1) = w1(i);
        u2{i}(1) = w2(i);
        v1{i}(1) = w3(i);
        v2{i}(1) = w4(i);
        energy{i}(1) = 1/2*(m1 + m2)*l1^2*v1{i}(1)^2 + 1/2*m2*l2^2*v2{i}(1)^2 + m2*
    l1*l2*v1{i}(1)*v2{i}(1)*cos(u1{i}(1) - u2{i}(1)) - (m1 + m2)*g*l1*cos(u1{i}(1))
     - m2*g*l2*cos(u2{i}(1));
        t = zeros(N+1,1);
        test = zeros(N+1,1);
        test1 = zeros(N+1,1);
        test2 = zeros(N+1,1);
        E_max =  (m1 + m2)*g*l1 + m2*g*l2;
    end

```

```
62      set(gcf,'color',[1 1 1]);
63
64      subplot(221);
65      for i = 1:simulations
66          xbob1(i) = l1*sin(u1{i}(1));
67          ybob1(i) = -l1*cos(u1{i}(1));
68          xbob2(i) = l1*sin(u1{i}(1))+l2*sin(u2{i}(1));
69          ybob2(i) = -l1*cos(u1{i}(1))-l2*cos(u2{i}(1));
70      end
71      h_rod1(1) = plot([0,xbob1(1)],[0,ybob1(1)],'k-','linewidth',2);
72      hold on;
73      h_rod2(1) = plot([xbob1(1),xbob2(1)],[ybob1(1),ybob2(1)],'k-','linewidth',2);
74      hold on;
75      h_bob1(1) = plot(xbob1(1),ybob1(1),'b.','markersize',20);
76      hold on;
77      h_bob2(1) = plot(xbob2(1),ybob2(1),'b.','markersize',20);
78      hold on;
79      if simulations >= 2
80          h_rod1(2) = plot([0,xbob1(2)],[0,ybob1(2)],'k-','linewidth',2);
81          hold on;
82          h_rod2(2) = plot([xbob1(2),xbob2(2)],[ybob1(2),ybob2(2)],'k-','linewidth'
    ,2);
83          hold on;
84          h_bob1(2) = plot(xbob1(2),ybob1(2),'r.','markersize',20);
85          hold on;
86          h_bob2(2) = plot(xbob2(2),ybob2(2),'r.','markersize',20);
87          hold on;
88      end
89      if simulations >= 3
90          h_rod1(3) = plot([0,xbob1(3)],[0,ybob1(3)],'k-','linewidth',2);
91          hold on;
92          h_rod2(3) = plot([xbob1(3),xbob2(3)],[ybob1(3),ybob2(3)],'k-','linewidth'
    ,2);
93          hold on;
94          h_bob1(3) = plot(xbob1(3),ybob1(3),'y.','markersize',20);
95          hold on;
96          h_bob2(3) = plot(xbob2(3),ybob2(3),'y.','markersize',20);
97      end
98      xlabel 'x'
99      ylabel 'y'
100     set(gca,'xlim',[-2.2 2.2],'ylim',[-2.2 2.2]);
101     axis square;
102
103     subplot(212);
104     for i = 1:simulations
105         energyPlot(i) = plot(t(1),abs(energy{i}(1)-energy{i}(1))/abs(energy{i}(1)),
    '-');
106         hold on;
107     end
108     set(gca,'xlim',[Tstartfig Tend]);
109     xlabel 'time, t_n (s)'
110     ylabel '|E(t)-E(0)| / |E(0)|'
111
112     subplot(243);
113     set(gca,'xlim',[-pi pi],'ylim',[-1.5*pi 1.5*pi]);
114     hold on;
115     h_phase1 = plot(u1{1}(1)+u2{1}(1),v1{1}(1)+v2{1}(1),'o','MarkerSize',1);
116     axis square;
117     xlabel ('$\theta_1$ (rad)','interpreter','latex');
118     ylabel ('$\dot{\theta_1}$ (rad/s)','interpreter','latex');
119
120     hold on
121     subplot(244)
122     set(gca,'xlim',[-pi pi],'ylim',[-1.5*pi 1.5*pi]);
123     hold on;
124     h_phase2 = plot(u1{1}(1)+u2{1}(1),v1{1}(1)+v2{1}(1),'o','MarkerSize',1);
125     axis square;
```

```
126     xlabel ('$\theta_2$ (rad)','interpreter','latex');
127     ylabel ('$\dot{\theta_2}$ (rad/s)','interpreter','latex');
128 
129     %% Create Movie:
130     if make_video == true
131         %set(gcf, 'Position', get(0, 'Screensize'));
132         if isunix % for linux
133            pathVideoAVI = '~/someVideo.avi'; % filename, used later to generate mp4
134         elseif ispc % fow windows
135            pathVideoAVI = 'video'; % filename, used later to generate mp4
136         end
137     video = VideoWriter(pathVideoAVI,'MPEG-4');
138     %video = VideoWriter(pathVideoAVI,'Uncompressed AVI');
139     %video.Quality=100;
140     if tau_adaptive == true
141         video.FrameRate = 2026/Tend; %only correct for Figure 3.2 and 3.3
142     else
143         video.FrameRate = 1/tau_initial;
144     end
145     open(video);
146     frame = getframe(gcf);
147     writeVideo(video,frame);
148     end
149 
150     if tau_adaptive == true
151         syms a b c d;
152         f = [c, d, (-m2*l2*(d)^2*sin(a-b)-1/2*m2*l1*(c)^2*sin(2*a-2*b)-(m1+m2)*g*
        sin(a)+m2*g*cos(a-b)*sin(b))/(l1*(m1+m2*(sin(a-b)^2))), ((m1+m2)*l1*(c)^2*sin(a
        -b)+1/2*m2*l2*(d)^2*sin(2*a-2*b)-(m1+m2)*g*(sin(b)-cos(a-b)*sin(a)))/(l2*(m1+m2
        *(sin(a-b)^2)))];
153         v = [a, b, c, d];
154         A=jacobian(f,v);
155     end
156 
157     sympref('FloatingPointOutput',true);
158     n = 0;
159     stop = false;
160     stop1 = false;
161     stop2 = false;
162     stop3 = false;
163     t(1) = 0;
164 
165     for i = 1:simulations
166         u1new{i} = w1(i);
167         u2new{i} = w2(i);
168         v1new{i} = w3(i);
169         v2new{i} = w4(i);
170     end
171 
172     saveas(gcf,'figure0.pdf');
173 
174     for n=1:N-N2
175 
176         t(n+1) = t(n) + tau;
177 
178         for i = 1:simulations
179             u1old{i} = u1new{i};
180             u2old{i} = u2new{i};
181             v1old{i} = v1new{i};
182             v2old{i} = v2new{i};
183 
184             %% Runge Kutta 4:
185             k11 = v1old{i};
186             k12 = v2old{i};
187             k13 = (-m2*l2*(v2old{i})^2*sin(u1old{i}-u2old{i})-1/2*m2*l1*(v1old{i})
        ^2*sin(2*u1old{i}-2*u2old{i})-(m1+m2)*g*sin(u1old{i})+m2*g*cos(u1old{i}-u2old{i
        })*sin(u2old{i}))/(l1*(m1+m2*(sin(u1old{i}-u2old{i})^2)));
```

```
188             k14 = ((m1+m2)*l1*(v1old{i})^2*sin(u1old{i}-u2old{i})+1/2*m2*l2*(v2old{
    i})^2*sin(2*u1old{i}-2*u2old{i})-(m1+m2)*g*(sin(u2old{i})-cos(u1old{i}-u2old{i
    })*sin(u1old{i})))/(l2*(m1+m2*(sin(u1old{i}-u2old{i})^2)));
189             k21 = v1old{i} + 1/2*tau*k13;
190             k22 = v2old{i} + 1/2*tau*k14;
191             k23 = (-m2*l2*(v2old{i}+1/2*tau*k14)^2*sin(u1old{i}+1/2*tau*k11-u2old{i
    }-1/2*tau*k12)-1/2*m2*l1*(v1old{i}+1/2*tau*k13)^2*sin(2*u1old{i}+tau*k11-2*
    u2old{i}-tau*k12)-(m1+m2)*g*sin(u1old{i}+1/2*tau*k11)+m2*g*cos(u1old{i}+1/2*tau
    *k11-u2old{i}-1/2*tau*k12)*sin(u2old{i}+1/2*tau*k12))/(l1*(m1+m2*(sin(u1old{i
    }+1/2*tau*k11-u2old{i}-1/2*tau*k12)^2)));
192             k24 = ((m1+m2)*l1*(v1old{i}+1/2*tau*k13)^2*sin(u1old{i}+1/2*tau*k11-
    u2old{i}-1/2*tau*k12)+1/2*m2*l2*(v2old{i}+1/2*tau*k14)^2*sin(2*u1old{i}+tau*k11
    -2*u2old{i}-tau*k12)-(m1+m2)*g*(sin(u2old{i}+1/2*tau*k12)-cos(u1old{i}+1/2*tau*
    k11-u2old{i}-1/2*tau*k12)*sin(u1old{i}+1/2*tau*k11)))/(l2*(m1+m2*(sin(u1old{i
    }+1/2*tau*k11-u2old{i}-1/2*tau*k12)^2)));
193             k31 = v1old{i} + 1/2*tau*k23;
194             k32 = v2old{i} + 1/2*tau*k24;
195             k33 = (-m2*l2*(v2old{i}+1/2*tau*k24)^2*sin(u1old{i}+1/2*tau*k21-u2old{i
    }-1/2*tau*k22)-1/2*m2*l1*(v1old{i}+1/2*tau*k23)^2*sin(2*u1old{i}+tau*k21-2*
    u2old{i}-tau*k22)-(m1+m2)*g*sin(u1old{i}+1/2*tau*k21)+m2*g*cos(u1old{i}+1/2*tau
    *k21-u2old{i}-1/2*tau*k22)*sin(u2old{i}+1/2*tau*k22))/(l1*(m1+m2*(sin(u1old{i
    }+1/2*tau*k21-u2old{i}-1/2*tau*k22)^2)));
196             k34 = ((m1+m2)*l1*(v1old{i}+1/2*tau*k23)^2*sin(u1old{i}+1/2*tau*k21-
    u2old{i}-1/2*tau*k22)+1/2*m2*l2*(v2old{i}+1/2*tau*k24)^2*sin(2*u1old{i}+tau*k21
    -2*u2old{i}-tau*k22)-(m1+m2)*g*(sin(u2old{i}+1/2*tau*k22)-cos(u1old{i}+1/2*tau*
    k21-u2old{i}-1/2*tau*k22)*sin(u1old{i}+1/2*tau*k21)))/(l2*(m1+m2*(sin(u1old{i
    }+1/2*tau*k21-u2old{i}-1/2*tau*k22)^2)));
197             k41 = v1old{i} + tau*k33;
198             k42 = v2old{i} + tau*k34;
199             k43 = (-m2*l2*(v2old{i}+tau*k34)^2*sin(u1old{i}+tau*k31-u2old{i}-tau*
    k32)-1/2*m2*l1*(v1old{i}+tau*k33)^2*sin(2*u1old{i}+2*tau*k31-2*u2old{i}-2*tau*
    k32)-(m1+m2)*g*sin(u1old{i}+tau*k31)+m2*g*cos(u1old{i}+tau*k31-u2old{i}-tau*k32
    )*sin(u2old{i}+tau*k32))/(l1*(m1+m2*(sin(u1old{i}+tau*k31-u2old{i}-tau*k32)^2))
    );
200             k44 = ((m1+m2)*l1*(v1old{i}+tau*k33)^2*sin(u1old{i}+tau*k31-u2old{i}-
    tau*k32)+1/2*m2*l2*(v2old{i}+tau*k34)^2*sin(2*u1old{i}+2*tau*k31-2*u2old{i}-2*
    tau*k32)-(m1+m2)*g*(sin(u2old{i}+tau*k32)-cos(u1old{i}+tau*k31-u2old{i}-tau*k32
    )*sin(u1old{i}+tau*k31)))/(l2*(m1+m2*(sin(u1old{i}+tau*k31-u2old{i}-tau*k32)^2)
    ));
201             u1new{i} = u1old{i} + 1/6*tau*(k11+2*k21+2*k31+k41);
202             u2new{i} = u2old{i} + 1/6*tau*(k12+2*k22+2*k32+k42);
203             v1new{i} = v1old{i} + 1/6*tau*(k13+2*k23+2*k33+k43);
204             v2new{i} = v2old{i} + 1/6*tau*(k14+2*k24+2*k34+k44);
205             energy{i}(n+1) = 1/2*(m1 + m2)*l1^2*v1new{i}^2 + 1/2*m2*l2^2*v2new{i}^2
     + m2*l1*l2*v1new{i}*v2new{i}*cos(u1new{i} - u2new{i}) - (m1 + m2)*g*l1*cos(
    u1new{i}) - m2*g*l2*cos(u2new{i});
206         end
207     end
208
209     for i = 1:simulations
210         u1{i}(n+1) = u1new{i};
211         u2{i}(n+1) = u2new{i};
212         v1{i}(n+1) = v1new{i};
213         v2{i}(n+1) = v2new{i};
214         %energy{i}(n+1) = 1/2*(m1 + m2)*l1^2*v1new{i}(1)^2 + 1/2*m2*l2^2*v2new{i
    }(1)^2 + m2*l1*l2*v1new{i}(1)*v2new{i}(1)*cos(u1new{i}(1) - u2new{i}(1)) - (m1
    + m2)*g*l1*cos(u1new{i}(1)) - m2*g*l2*cos(u2new{i}(1));
215     end
216
217     for n=(N-N2+1):N
218
219         if tau_adaptive == true
220             a_new = u1{1}(n);
221             b_new = u2{1}(n);
222             c_new = v1{1}(n);
223             d_new = v2{1}(n);
224             A_new = A;
```

```
225             A_new = subs(A_new, a, a_new); A_new = subs(A_new, b, b_new);
226             A_new = subs(A_new, c, c_new); A_new = subs(A_new, d, d_new);
227             A_new = double(A_new);
228             E = eig(A_new);
229             delta_t = sqrt(8)/max(abs(imag(E)));
230             test(n) = delta_t;
231             for j=1:4
232                 test1(4*(n-1) + j) = real(E(j));
233                 test2(4*(n-1) + j) = imag(E(j));
234             end
235
236             safety_factor = 0.1;
237             tau = safety_factor*delta_t/index;
238
239             if tau > 0.1
240                 tau = 0.1;
241             end
242             if t(n) + tau > Tend
243                 tau = Tend - t(n);
244                 stop = true;
245             end
246         end
247         if stop == true
248             break;
249         end
250
251         t(n+1) = t(n) + tau;
252
253         for i = 1:simulations
254
255             %% forward Euler:
256 %             u1{1}(n+1) = u1{1}(n) + tau*v1{1}(n);
257 %             u2{1}(n+1) = u2{1}(n) + tau*v2{1}(n);
258 %             v1{1}(n+1) = v1{1}(n) + tau*(-m2*l2*(v2{1}(n))^2*sin(u1{1}(n)-u2{1}(n
    ))-1/2*m2*l1*(v1{1}(n))^2*sin(2*u1{1}(n)-2*u2{1}(n))-(m1+m2)*g*sin(u1{1}(n))+m2
    *g*cos(u1{1}(n)-u2{1}(n))*sin(u2{1}(n)))/(l1*(m1+m2*(sin(u1{1}(n)-u2{1}(n)))^2))
    );
259 %             v2{1}(n+1) = v2{1}(n) + tau*((m1+m2)*l1*(v1{1}(n))^2*sin(u1{1}(n)-u2
    {1}(n))+1/2*m2*l2*(v2{1}(n))^2*sin(2*u1{1}(n)-2*u2{1}(n))-(m1+m2)*g*(sin(u2{1}(
    n))-cos(u1{1}(n)-u2{1}(n))*sin(u1{1}(n))))/(l2*(m1+m2*(sin(u1{1}(n)-u2{1}(n)))
    ^2)));
260 %             energy{1}(n+1) = 1/2*(m1 + m2)*l1^2*v1{1}(n+1)^2 + 1/2*m2*l2^2*v2{1}(
    n+1)^2 + m2*l1*l2*v1{1}(n+1)*v2{1}(n+1)*cos(u1{1}(n+1) - u2{1}(n+1)) - (m1 + m2
    )*g*l1*cos(u1{1}(n+1)) - m2*g*l2*cos(u2{1}(n+1));
261
262             %% backward Euler: *unused*
263
264 %             u1_1 = u1(n) + tau*max(n);
265 %             u2_1 = u2(n) + tau*v2(n);
266 %             v1_1 = v1(n) + tau*(-m2*l2*(v2(n))^2*sin(u1(n)-u2(n))-1/2*m2*l1*(v1(n
    ))^2*sin(2*u1(n)-2*u2(n))-(m1+m2)*g*sin(u1(n))+m2*g*cos(u1(n)-u2(n))*sin(u2(n))
    )/(l1*(m1+m2*(sin(u1(n)-u2(n)))^2)));
267 %             v2_1 = v2(n) + tau*((m1+m2)*l1*(v1(n))^2*sin(u1(n)-u2(n))+1/2*m2*l2*(
    v2(n))^2*sin(2*u1(n)-2*u2(n))-(m1+m2)*g*(sin(u2(n))-cos(u1(n)-u2(n))*sin(u1(n))
    ))/(l2*(m1+m2*(sin(u1(n)-u2(n)))^2)));
268 %             for iter = 1:maxIter,
269 %                 rth1 = u1_1 - u1(n) - tau*v1_1;
270 %                 rth2 = u2_1 - u2(n) - tau*v2_1;
271 %                 rv1 = v1_1 - v1(n) - tau*(-m2*l2*(v2_1)^2*sin(u1_1-u2_1)-1/2*m2*
    l1*(v1_1)^2*sin(2*u1_1-2*u2_1)-(m1+m2)*g*sin(u1_1)+m2*g*cos(u1_1-u2_1)*sin(u2_1
    ))/(l1*(m1+m2*(sin(u1_1-u2_1)^2)));
272 %                 rv2 = v2_1 - v2(n) - tau*((m1+m2)*l1*(v1_1)^2*sin(u1_1-u2_1)+1/2*
    m2*l2*(v2_1)^2*sin(2*u1_1-2*u2_1)-(m1+m2)*g*(sin(u2_1)-cos(u1_1-u2_1)*sin(u1_1)
    ))/(l2*(m1+m2*(sin(u1_1-u2_1)^2)));
273 %                 u1_1 = u1_1 - rth1;
274 %                 u2_1 = u2_1 - rth2;
275 %                 if(norm([rth1,rv1],inf)<tol) break; end;
```

```
276 %                 if(norm([rth2,rv2],inf)<tol) break; end;
277 %             end
278 %             %if iter>=maxIter, disp('convergence failed in backward Euler'); end
279 %             u1(n+1) = u1_1;
280 %             u2(n+1) = u2_1;
281 %             v1(n+1) = v1_1;
282 %             v2(n+1) = v2_1;
283
284
285           %% Runge Kutta 4:
286               k11 = v1{i}(n);
287               k12 = v2{i}(n);
288               k13 = (-m2*l2*(v2{i}(n))^2*sin(u1{i}(n)-u2{i}(n))-1/2*m2*l1*(v1{i}(
    n))^2*sin(2*u1{i}(n)-2*u2{i}(n))-(m1+m2)*g*sin(u1{i}(n))+m2*g*cos(u1{i}(n)-u2{i
    }(n))*sin(u2{i}(n)))/(l1*(m1+m2*(sin(u1{i}(n)-u2{i}(n))^2)));
289               k14 = ((m1+m2)*l1*(v1{i}(n))^2*sin(u1{i}(n)-u2{i}(n))+1/2*m2*l2*(v2
    {i}(n))^2*sin(2*u1{i}(n)-2*u2{i}(n))-(m1+m2)*g*(sin(u2{i}(n))-cos(u1{i}(n)-u2{i
    }(n))*sin(u1{i}(n))))/(l2*(m1+m2*(sin(u1{i}(n)-u2{i}(n))^2)));
290               k21 = v1{i}(n) + 1/2*tau*k13;
291               k22 = v2{i}(n) + 1/2*tau*k14;
292               k23 = (-m2*l2*(v2{i}(n)+1/2*tau*k14)^2*sin(u1{i}(n)+1/2*tau*k11-u2{
    i}(n)-1/2*tau*k12)-1/2*m2*l1*(v1{i}(n)+1/2*tau*k13)^2*sin(2*u1{i}(n)+tau*k11-2*
    u2{i}(n)-tau*k12)-(m1+m2)*g*sin(u1{i}(n)+1/2*tau*k11)+m2*g*cos(u1{i}(n)+1/2*tau
    *k11-u2{i}(n)-1/2*tau*k12)*sin(u2{i}(n)+1/2*tau*k12))/(l1*(m1+m2*(sin(u1{i}(n)
    +1/2*tau*k11-u2{i}(n)-1/2*tau*k12)^2)));
293               k24 = ((m1+m2)*l1*(v1{i}(n)+1/2*tau*k13)^2*sin(u1{i}(n)+1/2*tau*k11
    -u2{i}(n)-1/2*tau*k12)+1/2*m2*l2*(v2{i}(n)+1/2*tau*k14)^2*sin(2*u1{i}(n)+tau*
    k11-2*u2{i}(n)-tau*k12)-(m1+m2)*g*(sin(u2{i}(n)+1/2*tau*k12)-cos(u1{i}(n)+1/2*
    tau*k11-u2{i}(n)-1/2*tau*k12)*sin(u1{i}(n)+1/2*tau*k11)))/(l2*(m1+m2*(sin(u1{i
    }(n)+1/2*tau*k11-u2{i}(n)-1/2*tau*k12)^2)));
294               k31 = v1{i}(n) + 1/2*tau*k23;
295               k32 = v2{i}(n) + 1/2*tau*k24;
296               k33 = (-m2*l2*(v2{i}(n)+1/2*tau*k24)^2*sin(u1{i}(n)+1/2*tau*k21-u2{
    i}(n)-1/2*tau*k22)-1/2*m2*l1*(v1{i}(n)+1/2*tau*k23)^2*sin(2*u1{i}(n)+tau*k21-2*
    u2{i}(n)-tau*k22)-(m1+m2)*g*sin(u1{i}(n)+1/2*tau*k21)+m2*g*cos(u1{i}(n)+1/2*tau
    *k21-u2{i}(n)-1/2*tau*k22)*sin(u2{i}(n)+1/2*tau*k22))/(l1*(m1+m2*(sin(u1{i}(n)
    +1/2*tau*k21-u2{i}(n)-1/2*tau*k22)^2)));
297               k34 = ((m1+m2)*l1*(v1{i}(n)+1/2*tau*k23)^2*sin(u1{i}(n)+1/2*tau*k21
    -u2{i}(n)-1/2*tau*k22)+1/2*m2*l2*(v2{i}(n)+1/2*tau*k24)^2*sin(2*u1{i}(n)+tau*
    k21-2*u2{i}(n)-tau*k22)-(m1+m2)*g*(sin(u2{i}(n)+1/2*tau*k22)-cos(u1{i}(n)+1/2*
    tau*k21-u2{i}(n)-1/2*tau*k22)*sin(u1{i}(n)+1/2*tau*k21)))/(l2*(m1+m2*(sin(u1{i
    }(n)+1/2*tau*k21-u2{i}(n)-1/2*tau*k22)^2)));
298               k41 = v1{i}(n) + tau*k33;
299               k42 = v2{i}(n) + tau*k34;
300               k43 = (-m2*l2*(v2{i}(n)+tau*k34)^2*sin(u1{i}(n)+tau*k31-u2{i}(n)-
    tau*k32)-1/2*m2*l1*(v1{i}(n)+tau*k33)^2*sin(2*u1{i}(n)+2*tau*k31-2*u2{i}(n)-2*
    tau*k32)-(m1+m2)*g*sin(u1{i}(n)+tau*k31)+m2*g*cos(u1{i}(n)+tau*k31-u2{i}(n)-tau
    *k32)*sin(u2{i}(n)+tau*k32))/(l1*(m1+m2*(sin(u1{i}(n)+tau*k31-u2{i}(n)-tau*k32)
    ^2)));
301               k44 = ((m1+m2)*l1*(v1{i}(n)+tau*k33)^2*sin(u1{i}(n)+tau*k31-u2{i}(n
    )-tau*k32)+1/2*m2*l2*(v2{i}(n)+tau*k34)^2*sin(2*u1{i}(n)+2*tau*k31-2*u2{i}(n)
    -2*tau*k32)-(m1+m2)*g*(sin(u2{i}(n)+tau*k32)-cos(u1{i}(n)+tau*k31-u2{i}(n)-tau*
    k32)*sin(u1{i}(n)+tau*k31)))/(l2*(m1+m2*(sin(u1{i}(n)+tau*k31-u2{i}(n)-tau*k32)
    ^2)));
302               u1{i}(n+1) = u1{i}(n) + 1/6*tau*(k11+2*k21+2*k31+k41);
303               u2{i}(n+1) = u2{i}(n) + 1/6*tau*(k12+2*k22+2*k32+k42);
304               v1{i}(n+1) = v1{i}(n) + 1/6*tau*(k13+2*k23+2*k33+k43);
305               v2{i}(n+1) = v2{i}(n) + 1/6*tau*(k14+2*k24+2*k34+k44);
306               energy{i}(n+1) = 1/2*(m1 + m2)*l1^2*v1{i}(n+1)^2 + 1/2*m2*l2^2*v2{i
    }(n+1)^2 + m2*l1*l2*v1{i}(n+1)*v2{i}(n+1)*cos(u1{i}(n+1) - u2{i}(n+1)) - (m1 +
    m2)*g*l1*cos(u1{i}(n+1)) - m2*g*l2*cos(u2{i}(n+1));
307
308           % output:
309           xbob1(i) = l1*sin(u1{i}(n+1));
310           ybob1(i) = -l1*cos(u1{i}(n+1));
311           xbob2(i) = l1*sin(u1{i}(n+1))+l2*sin(u2{i}(n+1));
312           ybob2(i) = -l1*cos(u1{i}(n+1))-l2*cos(u2{i}(n+1));
```

```
313             set(h_rod1(i),'xdata',[0 xbob1(i)],'ydata',[0 ybob1(i)]);
314             set(h_rod2(i),'xdata',[xbob1(i) xbob2(i)],'ydata',[ybob1(i) ybob2(i)]);
315             set(h_bob1(i),'xdata',xbob1(i),'ydata',ybob1(i));
316             set(h_bob2(i),'xdata',xbob2(i),'ydata',ybob2(i));
317             set(energyPlot(i),'xdata',t(1:n+1),'ydata',abs((energy{i}(1:n+1)-energy
    {i}(1))/energy{i}(1)));
318         end
319         set(h_phase1,'xdata',wrapToPi(u1{1}(1:n+1)),'ydata',v1{1}(1:n+1));
320         set(h_phase2,'xdata',wrapToPi(u2{1}(1:n+1)),'ydata',v2{1}(1:n+1));
321
322         drawnow;
323
324         %% movie and pictures:
325         if make_video == true
326             frame = getframe(gcf);
327             writeVideo(video,frame);
328         end
329
330         if t(n+1) >= picture1 && stop1 == false
331             saveas(gcf,'figure1.pdf');
332             stop1 = true;
333         end
334         if t(n+1) >= picture2 && stop2 == false
335             saveas(gcf,'figure2.pdf');
336             stop2 = true;
337         end
338         if t(n+1) >= picture3 && stop3 == false
339             saveas(gcf,'figure3.pdf');
340             stop3 = true;
341         end
342     end
343     saveas(gcf,'figure4.pdf');
344
345     if richardson == true
346         if index == 1
347             R1 = energy{1}(:);
348         elseif index == 2
349             R2 = energy{1}(:);
350         elseif index == 4
351             R4 = energy{1}(:);
352         elseif index == 8
353             R8 = energy{1}(:);
354         elseif index == 16
355             R16 = energy{1}(:);
356         elseif index == 32
357             R32 = energy{1}(:);
358         elseif index == 64
359             R64 = energy{1}(:);
360         else
361             R128 = energy{1}(:);
362         end
363     end
364
365 end
366
367 if richardson == true
368     ord_conv_1 = -log((R4(end)-R2(end))/(R2(end)-R1(end)))/log(2)
369     ord_conv_2 = -log((R8(end)-R4(end))/(R4(end)-R2(end)))/log(2)
370     ord_conv_3 = -log((R16(end)-R8(end))/(R8(end)-R4(end)))/log(2);
371     ord_conv_4 = -log((R32(end)-R16(end))/(R16(end)-R8(end)))/log(2)
372     ord_conv_5 = -log((R64(end)-R32(end))/(R32(end)-R16(end)))/log(2)
373     ord_conv_6 = -log((R128(end)-R64(end))/(R64(end)-R32(end)))/log(2)
374 end
375
376 if make_video == true
377     close(video);
378     pathVideoMP4 = regexprep(pathVideoAVI,'\.avi','.mp4'); % generate mp4 filename
```

```
    if isunix % for linux
        [~,~] = system(sprintf('ffmpeg -i %s -y -an -c:v libx264 -crf 0 -preset
    slow %s',pathVideoAVI,pathVideoMP4)); % for this to work, you should have
    installed ffmpeg and have it available on PATH
    elseif ispc % for windows
        [~,~] = system(sprintf('ffmpeg.exe -i %s -y -an -c:v libx264 -crf 0 -preset
     slow %s',pathVideoAVI,pathVideoMP4)); % for this to work, you should have
    installed ffmpeg and have it available on PATH
    end
end

if tau_adaptive == true
    min(test(2:n))
end

test3 = test1./test2;
test4 = abs(test3(test3>=-1 & test3<=1));
subplot(222);
% First Histogram
% histogram(test4,10,'Normalization','probability','BinWidth',max(test4)/10);
% ytix = get(gca, 'YTick');
% set(gca, 'YTick',ytix, 'YTickLabel',ytix*100);
% xlabel ('Ratio of components of eigenvalues');
% ylabel ('Percentage (%)');
% ylim([0 0.4]);
% xlim([0 max(test4)]);
% Second Histogram
% histogram(test4,10,'Normalization','probability','BinWidth',1/10);
% ytix = get(gca, 'YTick');
% set(gca, 'YTick',ytix, 'YTickLabel',ytix*100);
% xlabel ('Ratio of components of eigenvalues');
% ylabel ('Percentage (%)');
% ylim([0 0.61]);
% xlim([0 1]);

% figure(3)
% for i = 1:simulations
% plot(t(2:n+1),abs((energy{i}(2:n+1)-energy{i}(1))/E_max));
% hold on;
% end
% xlim([0 t(n+1)]);
% legend('\theta_1(0) = \theta_2(0) = 0.5','\theta_1(0) = \theta_2(0) = 1.0','\
    theta_1(0) = \theta_2(0) = 1.5','\theta_1(0) = \theta_2(0) = 2.0', '\theta_1(0)
     = \theta_2(0) = 2.5');
% xlabel 'time, t_n (s)'
% ylabel '|E(t)-E(0)| / |E_m_a_x|'
```

Appendix D: Code for stability range of RK4

Matlab code used for plotting the stability range of RK4 including eigenvalues found in previously used Matlab code in Appendix C.

```
% Specify x range and number of points
x0 = -4;
x1 = 4;
Nx = 301;
% Specify y range and number of points
y0 = -4;
y1 = 4;
Ny = 301;
% Construct mesh
xv = linspace(x0,x1,Nx);
yv = linspace(y0,y1,Ny);
```

```
[x,y] = meshgrid(xv,yv);
% Calculate z
z = x + 1i*y;
% 4th order Runge-Kutta growth factor
g = 1 + z + 0.5*z.^2 + 1/6*z.^3 + 1/24*z.^4;
% Calculate magnitude of g
gmag = abs(g);
% Plot contours of gmag
cmap = [1 1 1];  % white

contourLevels = [-60 -30 -20 0 20 30 60];

figure();
contour(gmag, contourLevels);

% Use the custom colormap
colormap(cmap);

%colorbar()
set(gca, 'clim', [-60 60],'linewidth',1.75);
set(gca,'linewidth',6)
contourf(x,y,gmag,[1 1],'k-','linewidth',1,'HandleVisibility','off');
axis([x0,x1,y0,y1]);
axis('square');
axis on;
ax = gca;
ax.XRuler.Axle.LineWidth = 0.8;
ax.YRuler.Axle.LineWidth = 0.8;
%xline(0);
%yline(0);
xlabel('Real \lambda\tau');
ylabel('Imag \lambda\tau');
grid on;
hold on;
%for i=1:length(test)
%    for j=1:4
%        plot(test1(4*(i-1)+j)*test(i),test2(4*(i-1)+j)*test(i),'.','MarkerSize',5)
    ;
%    end
%end
%setting legend dot size and color
plot(0,123,'.','MarkerSize',20,'Color',[0 0.4470 0.7410]);
plot(0,123,'.','MarkerSize',20,'Color',[0.8500 0.3250 0.0980]);
plot(0,123,'.','MarkerSize',20,'Color',[0.9290 0.6940 0.1250]);
plot(0,123,'.','MarkerSize',20,'Color',[0.4940 0.1840 0.5560]);
for i=1:length(test(test>0))
    plot(test1(4*(i-1)+1)*test(i),test2(4*(i-1)+1)*test(i),'.','MarkerSize',5,'
    Color',[0 0.4470 0.7410]);
    plot(test1(4*(i-1)+2)*test(i),test2(4*(i-1)+2)*test(i),'.','MarkerSize',5,'
    Color',[0.8500 0.3250 0.0980]);
    plot(test1(4*(i-1)+3)*test(i),test2(4*(i-1)+3)*test(i),'.','MarkerSize',5,'
    Color',[0.9290 0.6940 0.1250]);
    plot(test1(4*(i-1)+4)*test(i),test2(4*(i-1)+4)*test(i),'.','MarkerSize',5,'
    Color',[0.4940 0.1840 0.5560]);
end
legend('\lambda_1','\lambda_2','\lambda_3','\lambda_4');
% for i=1:length(test(test>0))
%    for j=1:4
%        plot(test1(4*(i-1)+j)*test(i),test2(4*(i-1)+j)*test(i),'.','MarkerSize
    ',5,'Color',[1 1-(i-1)/(length(test)-1) 0]);
%    end
% end
```

Appendix E: Code for plotting multiple energy errors

Matlab code used for plotting multiple energy errors compared to E_{max} with a logarithmic y-axis using previously used Matlab code in Appendix C.

```
figure(3);
for i = 1:simulations
    semilogy(t(2:n+1),abs((energy{i}(2:n+1)-energy{i}(1))/E_ref));
    hold on;
end
xlim([0 t(n+1)]);
ylim([10^-17 10^-5])
legend('\theta_1(0) = \theta_2(0) = 0.5','\theta_1(0) = \theta_2(0) = 1.0','\
    theta_1(0) = \theta_2(0) = 1.5','\theta_1(0) = \theta_2(0) = 2.0', '\theta_1(0)
     = \theta_2(0) = 2.5');
xlabel 'time, t_n'
ylabel '|(E(t)-E(0))|/E_m_a_x'
```