Project 2

Amanda Lind Davíðsdóttir <amanda19@ru.is>

Bjarmi Valentino B Del Negro <bjarmib19@ru.is>

Eric Ruge <eric22@ru.is>

Fanney Einarsdóttir <fanneye22@ru.is>

T-406-TOLU

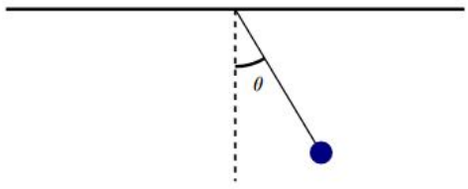
RU Science and Engineering

December 4, 2022

Numerical Analysis - Project 2

Simple pendulum

A simple pendulum is an easy but important classical example in mechanics:



Mynd 1: Einfaldur pendúll

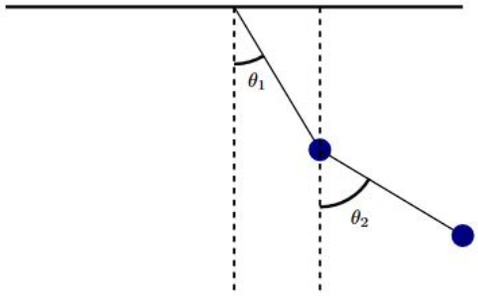
The rod has length L and we assume it is massless and perfectly stiff. According to Newton's laws the angle θ must satisfy the differential equation

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin(\theta) = 0 \tag{1}$$

where $g = 9.81\, m \cdot s^{-2}$. It is easy to approximately solve the equation by using $\sin(\theta) \simeq \theta$ but the equation is solvable as is, both analytically and numerically.

Double pendulum

When two penduli are connected together the situation becomes vastly more complex:



Mynd 2: Tvöfaldur pendúll

The rods have length L_1 and L_2 and the point masses are m_1 and m_2. One can show that the angles θ_1 and θ_2 satisfy the differential system

$$\begin{aligned}
\frac{d^2\theta_1}{dt^2} &= \frac{m_2 L_1 \omega_1^2 \sin(\Delta)\cos(\Delta) + m_2 g \sin(\theta_2)\cos(\Delta) + m_2 L_2 \omega_2^2 \sin(\Delta) - (m_1+m_2) g \sin(\theta_1)}{(m_1+m_2)L_1 - m_2 L_1 \cos^2(\Delta)} \\
\frac{d^2\theta_2}{dt^2} &= \frac{-m_2 L_2 \omega_2^2 \sin(\Delta)\cos(\Delta) + (m_1+m_2)\big(g\sin(\theta_1)\cos(\Delta) - L_1\omega_1^2\sin(\Delta) - g\sin(\theta_2)\big)}{(m_1+m_2)L_2 - m_2 L_2 \cos^2(\Delta)}
\end{aligned} \tag{2}$$

where $\Delta = \theta_2 - \theta_1$, $\omega_1 = \theta_1'$ and $\omega_2 = \theta_2'$.

Single pendulum

Problem 1

The differential equation (1) can be rewritten into a first-degree differential system with two variables, as described in the following. First, it is necessary to define $\bar{\theta}$ as a system (3).

$$\bar{\theta} = \begin{pmatrix} \theta \\ \theta' \end{pmatrix} \tag{3}$$

Second, the system is derived into $\bar{\theta}'$ (4).

$$\bar{\theta}' = \begin{pmatrix} \theta' \\ \theta'' \end{pmatrix} \tag{4}$$

The last step is to find the second derivative (5) of equation (1) and insert it into equation (4), which results in a first-degree differential system $F(t, \theta)$ with two variables (6).

$$\theta'' = -\frac{g \cdot \sin\theta}{L} \tag{5}$$

$$\bar{\theta}' = \begin{pmatrix} \theta' \\ -\frac{g \cdot \sin\theta}{L} \end{pmatrix} = F(t, \theta) \tag{6}$$

Problem 2

The simple pendulum from Problem 1 is solvable with the programming and numeric computing software Matlab. In the following sections, Euler's method is used to estimate the position of the pendulum. Euler's method uses small tangent lines (slopes) over small distance steps (h) to approximate the solution to an initial-value problem, as shown in equation (7).

$$y(i+1) = y(i) + f(t(i), y(i)) \cdot h \tag{7}$$

Euler's method is transferred into code and used in the simple pendulum problem. The step width h is calculated by the input of the total time and the number of steps. Afterward, the time is defined as t(1) = 0, and the system (3) is initialized. Subsequently, a for loop calculates the tangents in the defined step width using the inserted function of equation (8) named eulerstep by calculating the values as vectors. Calculating the values in vectors enables Matlab to plot the results in an animation, which is shown in Problem 3.

```
function [t y]=Euler_func_multi(x,n,T)
h=T/n;
t(1)=0;
y=[x(1); x(2)]; %To get the initial conditions

for i=1:n
    f = ydot(t(i),y(:,i)); %Now this is a vector
    y(:,i+1)=eulerstep(t(i),y(:,i),h);  %y(:,i)+f'.*h %switching f because its a column
    t(i+1)=t(i)+h;
end
end
```

Eulerstep transforms equation (7) into code by utilizing ydot as the system which needs to be solved. In this problem, it's equation (6).

```
function y=eulerstep(t,x,h)
y=x+h.*ydot(t,x);
end
```

Figure 1: Function eulerstep.m

The function ydot transforms equation (1) into the system of equation (7) and defines the length of the pendulum as well as the gravitational constant g.

```
function z=ydot(t,y)
g=9.81; L=2;
z(1,:) = y(2); %From problem 1 we can see that the first line in Z_vec is equal to y_prime
z(2,:) = -g/L*sin(y(1)); %From problem 1 we can see that the second line in Z_vec is equal to this
    ↪ equation
end
```

Figure 2: Function ydot.m

Problem 3

In Problem 3 the system which needs to be solved is defined as shown in table 1.

Table 1: Values of problem 3

$\theta(0)$	$\theta'(0)$	T	n
$\frac{\pi}{12}$	0	20	500

The functions described in Problem 2 are used to solve the system and are accessed by the following code 3.

```
clear all; close all; clc;

x=[pi/12;0]; %Initial conditions
T=20;
n=500;
L=2
[t, y]=Euler_func_multi(x,n,T);

plot_theta(t,y)%plot theta
hold off
figure
plot_pendulum(y,L) %plot the pendulum
```

Figure 3: Code to solve problem 3

The code consists of two additional functions to plot the pendulum and theta. The function plot_pendulum(y,L) uses the built-in Matlab function animatedline, addpoints, clearpoints and drawnow to create the animation. The function plot_theta(t,y) draws theta over time. These Matlab functions are shown in section A and B in the appendix. The created video of the moving pendulum can be seen in this video: link. The result of plotting theta is shown in the following figure 4.

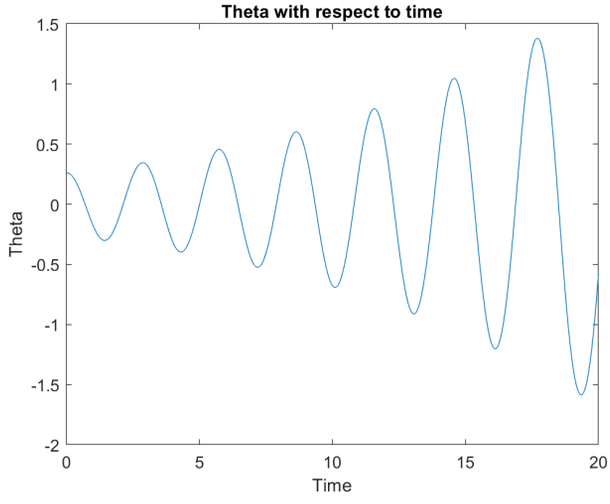


Figure 4: Resulting Theta over time

The figure and the video shows how Euler's method results in a wrong approximation of the system. Due to no friction in the system, the result should be a constant wave. Instead of a constant wave, Euler's method results in an infinite growing wave. This is because Euler's method can not improve itself. In the beginning, the error seems relatively small, but from there the error accumulates with each iteration. Making the system uncontrollable.

Problem 4

In this section problem, 3 was repeated, but the value of theta was changed, see in table 2. The code is only changed to set the new vale of $\theta(0)$ but uses the same functions as shown in section O.

Table 2: Values of problem 3

$\theta(0)$	$\theta'(0)$	T	n
$\frac{\pi}{2}$	0	20	500

$\theta(0)$ is the angle that the pendulum has at the time 0. In comparison to problem 3, the starting angle in this example is 90 degrees and therefore bigger. Consequently, the amplitude of θ is bigger from the beginning. As shown in the following figure 5 and animation: link,

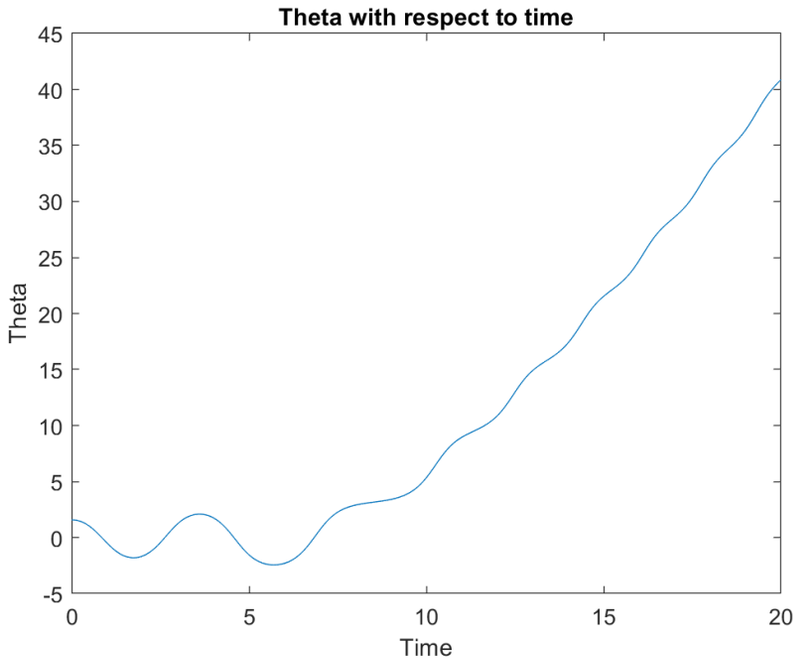


Figure 5: Resulting Theta over time in problem 4

the system becomes uncontrollable even faster than the system in Problem 3. This is because after four times crossing the midpoint at the bottom, the pendulum does not return, but starts spinning counter-clockwise. Euler's method is accumulating the error, making the pendulum go over the return point at swing nr. 4 and therefore the pendulum starts swinging in circles.

Problem 5

Euler's method was exchanged for Runge-Kutta's method and problems 3 and 4 were repeated. Fourth order Runge Kutta's method gives a good approximation of ordinary differential equations solutions. Instead of only calculating one slope per step (h) like in Euler's method, it calculates four different slopes per step (h) and uses them to calculate the weighted average.

To transform Runge-Kutta's method into code the function RK-method-single-pendulum.m, as seen in figure 6 was made to calculate the four slopes (k1, k2, k3, k4) for the four lines. Slope 1 is an Euler estimate at y=0, slope 2 is calculated at the midpoint $\frac{h}{2}$, slope 3 is an update at the midpoint, and slope 4 is the average to find the estimate at y=h.

```
function [t y] = RK_method_single_pendulum(x,n,T)

h=T/n;
t(1)=0;
y=x; %To get the initial conditions


for i=1:n

    k1 = ydot(t(i),y(:,i)); %slope 1
    k2 = ydot(t(i)+h/2,y(:,i)+(h/2).*k1); %slope 2
    k3 = ydot(t(i)+h/2,y(:,i)+(h/2).*k2); %slope 3
    k4 = ydot(t(i)+h,y(:,i)+h.*k3); %slope 4
    y(:,i+1)=RK_step(y(:,i),h,k1,k2,k3,k4);
    t(i+1)=t(i)+h;

end



end
```

Figure 6: Function RK-method-single-pendulum.m

To calculate the weighted average of the four slopes used in the RK-method-single-pendulum, a new function RK-step, seen in figure 7, was made. The standard weights for the method were used in the calculations.

```
function y = RK_step(x,h,k1,k2,k3,k4)
y = x + h*((k1/6)+(k2/3)+(k3/3)+(k4/6)); %weighted average
end
```

Figure 7: Function RK-step.m

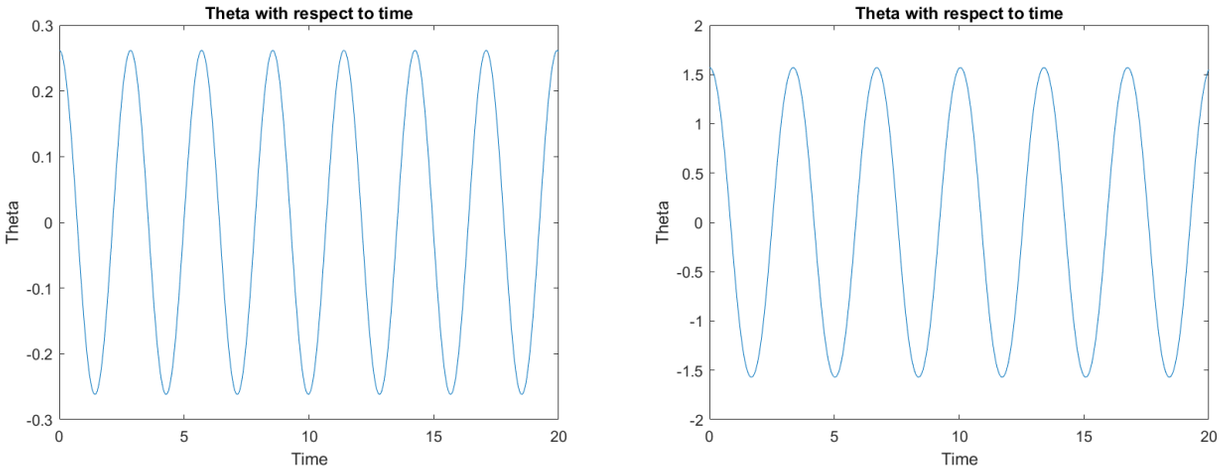
The main code for this problem can be seen in figure 8.

```
1  clear all; close all; clc;
2
3  % Problem 5.1
4  x=[pi/12;0]; %Initial conditions
5  T=20;
6  n=500;
7  L = 2;
8
9  [t, y] = RK_method_single_pendulum(x,n,T);
10
11 plot_theta(t,y)
12 figure
13 plot_pendulum(y,L)
14
15 %% Problem 5.2
16 x=[pi/2;0]; %Initial conditions
17 T=20;
18 n=500;
19 L = 2;
20
21 [t, y] = RK_method_single_pendulum(x,n,T);
22
23 plot_theta(t,y)
24 figure
25 plot_pendulum(y,L)
```

Figure 8: Function problem5.m

The code consists of two additional functions to plot the pendulum and theta as explained before in problem 3. The first part shows when Runge-Kutta's method is used with the values from problem 3 (starts in line 3 in figure 8), and the second part with the values from problem 4 (starting in line 15 in figure 8).

When comparing the two methods together, the animations: link, and the figures 9a and 9b with figures 4 and 5 it can be seen that Runge-Kutta's method has a much better approximation than Euler's method.



(a) $\theta(0) = \frac{\pi}{12}$

(b) $\theta(0) = \frac{\pi}{2}$

Figure 9: Resulting Theta over time with Runge-Kutta's method.

If the two figures 9a and b, are compared, it can be seen that the range of θ is much higher in (b) when the

angle is bigger. That is because when $\theta = \frac{\pi}{2}$, the pendulum starts higher up, and therefore the wavelengths are longer and the amplitude is higher since there is more potential energy stored in the system and that results in greater kinetic energy when the pendulum has reached the bottom.

As mentioned in problem 3, due to no friction in the system, the result should be a constant wave, and that can be seen in figure 9 when using Runge-Kutta's method. Therefore, it can be said that Euler's method is very inaccurate and gives a high error. When using Runge-Kutta's method the error decreases significantly and gives more accurate results than Euler's method.

Double pendulum - Implementation and errors

Problem 6

The differential equation (2) can be rewritten into a first-degree differential system for four variables, as described in the following. First, it is necessary to define $\bar{\theta}$ as a system (8).

$$\bar{\theta} = \begin{pmatrix} \theta_1 \\ \theta_1' \\ \theta_2 \\ \theta_2' \end{pmatrix} \tag{8}$$

Second, the system is derived into $\bar{\theta}'$ (9).

$$\bar{\theta}' = \begin{pmatrix} \theta_1' \\ \theta_1'' \\ \theta_2' \\ \theta_2'' \end{pmatrix} \tag{9}$$

The last step is to find the second derivative of θ_1 (10) and θ_2 (11) from equation (2) and insert it into equation (9), which results in a first-degree differential system $F(t, \theta)$ with four variables (12).

$$\theta_1'' = \frac{m_2 L_1 \omega_1^2 \sin(\Delta)\cos(\Delta) + m_2 g \sin(\theta_2)\cos(\Delta) + m_2 L_2 \omega_2^2 \sin(\Delta) - (m_1+m_2) g \sin(\theta_1)}{(m_1+m_2) L_1 - m_2 L_1 \cos^2(\Delta)} \tag{10}$$

$$\theta_2'' = \frac{-m_2 L_2 \omega_2^2 \sin(\Delta)\cos(\Delta) + (m_1+m_2)(g \sin(\theta_1)\cos(\Delta) - L_1 \omega_1^2 \sin(\Delta) - g \sin(\theta_2))}{(m_1+m_2) L_2 - m_2 L_2 \cos^2(\Delta)} \tag{11}$$

$$\bar{\theta}' = \begin{pmatrix} \theta_1' \\ \frac{m_2 L_1 \omega_1^2 \sin(\Delta)\cos(\Delta) + m_2 g \sin(\theta_2)\cos(\Delta) + m_2 L_2 \omega_2^2 \sin(\Delta) - (m_1+m_2) g \sin(\theta_1)}{(m_1+m_2) L_1 - m_2 L_1 \cos^2(\Delta)} \\ \theta_2' \\ \frac{-m_2 L_2 \omega_2^2 \sin(\Delta)\cos(\Delta) + (m_1+m_2)(g \sin(\theta_1)\cos(\Delta) - L_1 \omega_1^2 \sin(\Delta) - g \sin(\theta_2))}{(m_1+m_2) L_2 - m_2 L_2 \cos^2(\Delta)} \end{pmatrix} = F(t, \theta) \tag{12}$$

where $\Delta = \theta_2 - \theta_1$, $\omega_1 = \theta_1'$ and $\omega_2 = \theta_2'$.

Problem 7

In this section, the code from the simple pendulum is modified as it solves a double pendulum and draws the motion of both pendulums in real-time.

The system consists of the constant mass and lengths of the pendulums

$$L_1 = L_2 = 2 \qquad m_1 = m_2 = 1$$

and the initial values

Table 3: Initial values of problem 7

$\theta_1(0)$	$\theta_1'(0)$	$\theta_2(0)$	$\theta_2'(0)$	T	n
$\frac{\pi}{3}$	0	$\frac{\pi}{6}$	0	20	350

The variable n is chosen by experimenting with the value from 50 to 1000 in steps of 50. The value 350 gives a smooth and stable animation with a reasonable speed. Low values of n result in a very inaccurate and fast system, while large values give a slow animation due to the enormous amount of calculations that the computer has to perform. The variables and constants are included in the main code to run the system as shown in the following figure 10.

```
clear all; close all; clc;
L1=2; L2=2; m1=1; m2=1;T=20;

y0 = [pi/3;0;pi/6;0];
n = 350;
T = 20;

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
plot_double_pendulum(y,L1,L2)
```

Figure 10: Function problem7.m

The Runge-Kutta method solves the system, which is explained in problem 5. RK_method_double_pendul is working with the same principle but uses ydot_double_penulum to calculate the k's, as shown in section C. Figure 11 shows the new ydot_double_pedulum function, which consists of the system equations from problem 6 and the gravitation constant g.

```
function z = ydot_double_pedulum(t,y,L1,L2,m1,m2)
g=9.81;
delta = y(3)-y(1);
w1 = y(2);
w2 = y(4);
theta1 = y(1);
theta2 = y(3);

z(1,:) = w1;
z(2,:) = (m2*L1*w1^2*sin(delta)*cos(delta)+m2*g*sin(theta2)*cos(delta)+m2*L2*w2^2*sin(delta)-(m1+m2)
    ↪ *g*sin(theta1))/((m1+m2)*L1-m2*L1*cos(delta)^2);
z(3,:) = w2;
z(4,:) = (-m2*L2*w2^2*sin(delta)*cos(delta)+(m1+m2)*(g*sin(theta1)*cos(delta)-L1*w1^2*sin(delta)-g*
    ↪ sin(theta2)))/((m1+m2)*L2-m2*L2*cos(delta)^2);

end
```

Figure 11: Function ydot_double_pedulum.m

The results are plotted with plot_double_pendulum. The function works as the plot function explained in previous sections, but is adjusted to plot both pendulums at the same time and draws an animation that shows both pendulums' trajectories, as shown in section D. The variable path is created to draw the lines where the pendulum went. It is an animated line that gets updated in the for-loop by the position of the pendulum, it is representing.

The animation of the described system with the initial values of table 3 is shown in this video: link. The double pendulum starts on the right, where the mass closer to the attachment point (red) is higher than the mass on the bottom of the pendulum (blue). When the pendulum is released, red reaches the return point faster than blue. When red stops and tries to turn around, blue is taking over going to the left and pulling red even further. This results in blue moving in a small loop before turning around. The animation shows that red travels on one radius-shaped path, whereas blue travels on different paths each time.

Problem 8

In the following section, the system of problem 7 is analyzed by comparing different initial values as shown in the following table 4. The starting angles are changed, so the double pendulum is raised higher at the start until both balls are in a vertical position at the top in number 6 ($\theta = \pi$). The time interval $T = 20$, the iterations of $n = 350$ and $\theta_1' = 0, \theta_2' = 0$ remain the same as in problem 7. Table 4 shows the initial values of the six scenarios, which are also stored in video files here: link.

Table 4: Initial values of problem 8

No.	$\theta_1(0)$	$\theta_2(0)$
1	$\frac{\pi}{2}$	$\frac{\pi}{2}$
2	$\frac{\pi}{2}$	$\frac{\pi}{6}$
3	$\frac{\pi}{1.5}$	$\frac{\pi}{6}$
4	$\frac{\pi}{3}$	$\frac{\pi}{4}$
5	$\frac{\pi}{3}$	π
6	π	π

No. 1

The double pendulum starts at 90 degrees to the right-hand side in a straight line. Compared to problem 7 this brings more potential energy into the system since this angle is larger. After the first time, crossing the midpoint, the outcome is similar. While the red ball travels in a circular motion, the lower ball (blue) returns in a small loop. When the blue ball is performing the loop, the red ball starts returning to the right side. Right before the red ball reaches the lowest point, the blue ball overtakes and reaches the highest point below the starting point, whereas the red ball exceeds the starting point of 90 degrees. When the balls reach the left side the second time, the blue ball exceeds the last height, whereas the red ball stays lower than the last time. This change of potential energy between the two balls courses the blue ball to overshoot the red ball and move in a circular motion around it in swing 6. When the blue ball swings around the red, the red ball changes the direction of motion. After the blue ball circled the red once, the system is quickly changing directions and becomes more unpredictable.

No. 2

In this example, red starts again at 90 degrees, whereas blue starts significantly lower at 30 degrees from the perpendicular of the red ball. This means that more potential energy is stored in red. The red ball accelerates fast on the circular path, whereas blue drops vertically in the beginning. The blue ball gets then pulled by red transferring the momentum in an almost linear motion until blue overtakes. Blue then continues in an almost vertical motion until reaching nearly the height of the attachment point. From there blue and red are changing quickly directions, ending up with blue circling around the red ball. After this happens, it seems like the system is stabilizing, meaning the blue ball goes into parallel circular motion with the red ball.

No. 3

In no. 3, red starts even higher than 90 degrees while blue remains at the same angle as before in no. 2. Compared to no. 2 this results in blue circling the red ball after the first time the double pendulum crosses the midline. From there, the blue ball spins multiple times around red, spending more time right under the attachment point than on the left or right side.

No. 4

In the following example no. 4, both balls start at a smaller angle than in the examples before. With less potential energy in the system, the energy transfer between the two balls is a lot smaller, resulting in a stable system, as seen in the animation. The balls overtake each other slowly in each swing by remaining in an almost circular motion around the attachment point.

No. 5

In number 5 the blue ball is vertically in line with the red ball, while the position of red is the same as in no. 4. When the double pendulum is released, the blue ball moves in a parable motion for the first swing. When returning from the left side, the parable motion is flattening out. Blue then continues with big loops at the returning points, ending with looping around the red ball once. From there, blue continues with bouncing motions while red is moving fast from left to right.

No. 6

The last example no. 6 is different from the others. In this example, both balls start vertically in line at the top of the attachment point. In reality, both balls would not move as long as both balls are perfectly aligned. In this simulation, the Runge-Kutta method is used, which results in a small error after every iteration. This small error accumulates, which makes the double pendulum move after 14 sec (approx 245 iterations) in the simulation. The red ball circles the attachment point while the blue ball gets dragged along.

Problem 9

In this section, the error of the Runge-Kutta method is analyzed and displayed by using a modified version of the code from problem 7 and the initial values no.1 to no.5 from table 4. Instead of drawing the animation, the code will return the position of the pendulum at the end of the time interval.

The program is run with n (100, 200, 400, 800, 1600, 3200, 6400) and returns a vector describing the double pendulum at $T = 20$. As seen in the main code in figure 12 the error is calculated by the function RK_calc_errors(y0, n, T, L1, L2, m1, m2).

```
%% Problem 9
clear all; close all; clc;

L1=2; L2=2; m1=1; m2=1;T=20;

y0 = [pi/3;0;pi/6;0]; %Initial values like in problem 7
%rand    upphafsgildin max og min pi/2
n = 100;
T = 20;

[error,n_vec] = RK_calc_errors(y0, n, T, L1, L2, m1, m2);

% Do best fit on errors on loglog plot
coefficients = polyfit(log(n_vec),log(error),1);
y1 = polyval(coefficients,log(n_vec));

plot(log(n_vec),log(error),'*')
hold on
plot(log(n_vec),y1)
xl = xlim;
yl = ylim;
xt = 0.7 * (xl(2)-xl(1)) + xl(1);
yt = 0.75 * (yl(2)-yl(1)) + yl(1);
caption = sprintf('y = %.2f * x + %.2f', coefficients(1), coefficients(2));
text(xt, yt, caption, 'FontSize', 9, 'Color', 'k');
ylabel('log(Error)')
xlabel('log(n)')
legend('Errors','Fitted line')
```

Figure 12: Main code of problem 9

The function in figure 13 calculates one very precise approximation with 30000 iterations and then compares the final value of θ with approximations of n iterations. The n is doubled in each iteration of the for loop resulting in the values (100, 200, 400, 800, 1600, 3200, 6400) as mentioned before. To display the results a logarithm scale (ln,ln) is used on both axes, it enables a linear presentation of the data.

```
function [error n_vec] = RK_calc_errors(y0,n,T,L1,L2,m1,m2);
n_theoretical = 30000; %many intervals so very precise values
[t y_true_val] = RK_method_double_pendulum(y0,n_theoretical,T,L1,L2,m1,m2); %y_true_val is the true
    ↪ value (theoretical)
y_true_val = y_true_val'; %transpose the vectors

true_val = y_true_val(end,:); %we take the last line and all the colums to get the biggest error
for i = 1:7
    [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
    y = y'; %transpose the vector
    error(i) = norm(y(end,:) - true_val); %error of the norm of the all the final values of the
    ↪ thetas
    n_vec(i) = n; % keep all the different n values that are used
    n = n*2;
end
end
```

Figure 13: Calculation of the error when using the Runge-Kutta method

Figure 14 shows how the error decreases with the number of iterations in every example. The error in no 1 and 2 is decreasing the fastest, with a slope of -4.39 and -4.44. The error in no. 3 and 5 are decreasing much slower, with a slope of -2.28 and -2.7. Example number 4 is with a slope of -3.76 close to the theoretical value of -4. The theoretical value comes from a mathematical proof, which shows that the used Runge-Kutta method is a fourth degree method [1].

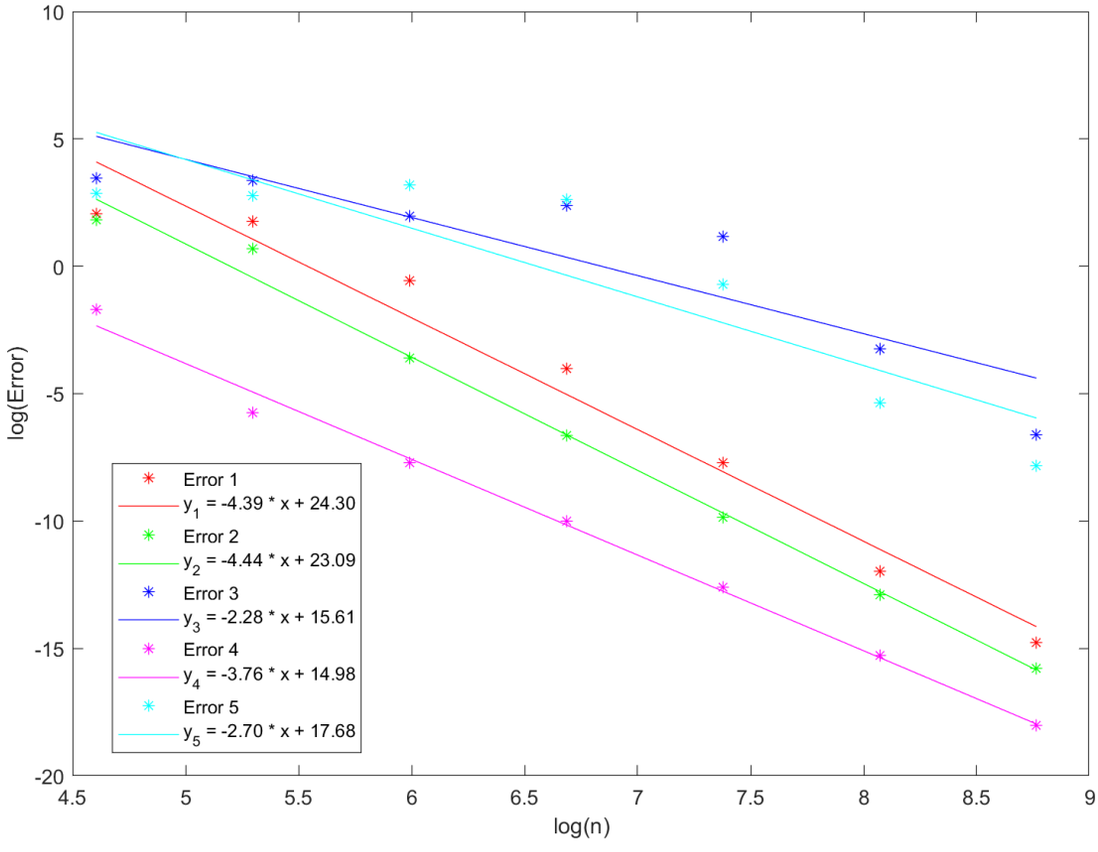


Figure 14: The error of the position from the pendulum at the end of the time interval with the initial values from no.1 to no.5 of table 4

The reason why the slopes vary from the theory is because of the first points of each graph. The first points are not accurate enough due to the small number of iterations, which is then influencing the fitted graph through the points. Analyzing n from (800, 1600, 3200, 6400, 12800, 25600, 51200) in figure 14 shows how the fitted slopes get closer to the theory.

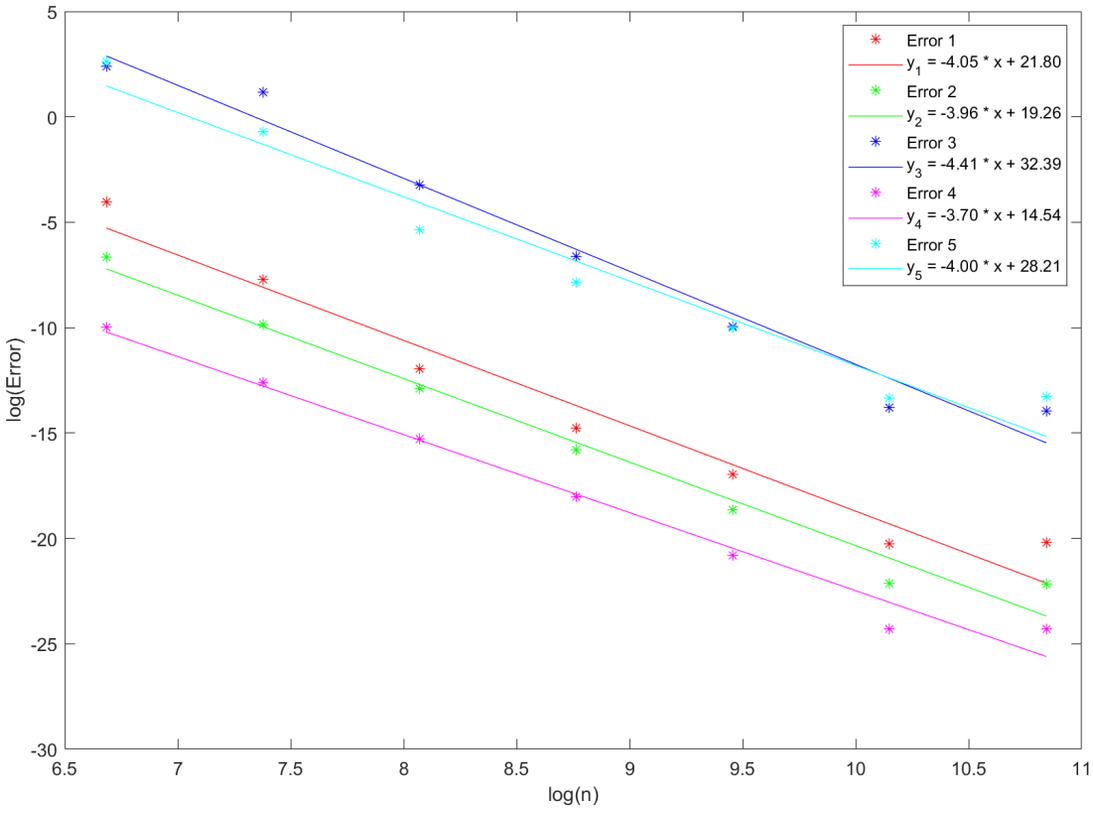


Figure 15: The error of the position from the pendulum at the end of the time interval with the initial values from no.1 to no.5 of table 4 and more iterations.

Problem 10

A program was written that draws an animated parametrized curve $\mathbb{R}^2$ such that

$$x = \theta_1(t), y = \theta_2(t)$$

where θ_1 and θ_2 are the position angles.

The same constants were used from Problem 7, where the system consists of the constant mass and lengths of the pendulums

$$L_1 = L_2 = 2 \qquad m_1 = m_2 = 1$$

and the initial values

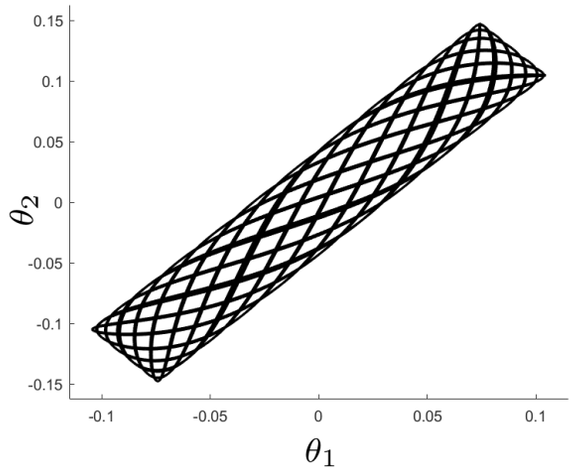
Table 5: Initial values of problem 7

$\theta_1(0)$	$\theta_1'(0)$	$\theta_2(0)$	$\theta_2'(0)$	T	n
$\frac{\pi}{3}$	0	$\frac{\pi}{6}$	0	50	10000

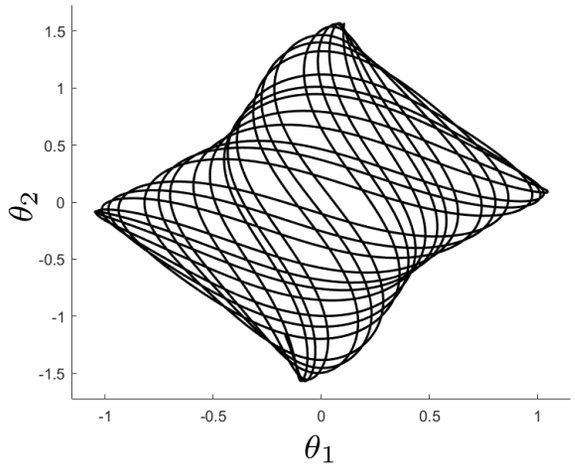
The code for this problem is similar to problem 7 in figure 10 except for different values of T and n. The same RK_method_double_pendulum.m is used but this time another plot function is used, animated_curve.m, since the plot is showing the correlation between θ_1 and θ_2.

The code was run for various initial values and they were chosen by experimenting to see a pattern. When the angles of both θ_1 and θ_2 are decreased, the pattern is more cyclic and it results in an almost perfect rectangle as can be seen in figure 16a. But as the angles increase, the pattern becomes less and less cyclic until they almost become just random. This development can be seen in figures 16a-f. The animations when the correlation is cyclic 16a and not cyclic at all 16f and also the animation of the double pendulum at the same initial angles, can be seen here: link.

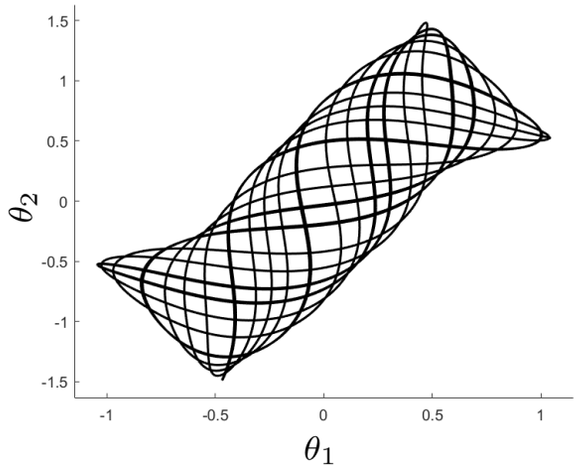
As seen in the animations, in (a) the position angles go similar paths and get closer and closer with time, and in the end, it nearly goes exactly in the same path again and again to make an almost perfect rectangle. This makes sense since when the double pendulum animation is looked at, the double penduli is almost in line the whole time, just a small change but it is structured, confirming the parametrized curve. However, in (f) the position angles are constantly trying to search for a path but the final outcome results in a random graph that is not very structured. That is also confirmed when looking at the double pendulum animation, since the two position angles are much higher and start at different angles, the double pendulum isn't structured at all and goes all over the place.



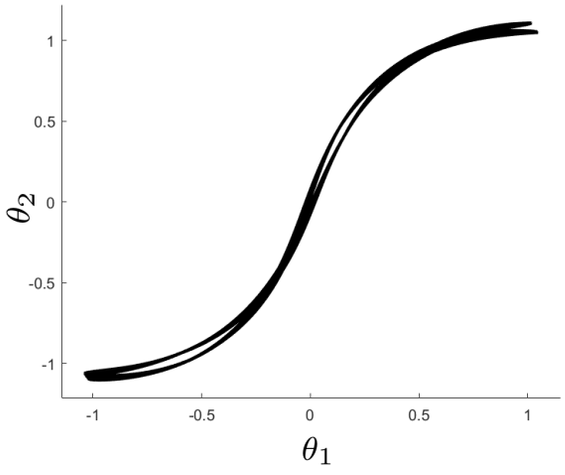
(a) $\theta_1 = \frac{\pi}{30}, \theta_2 = \frac{\pi}{30}$



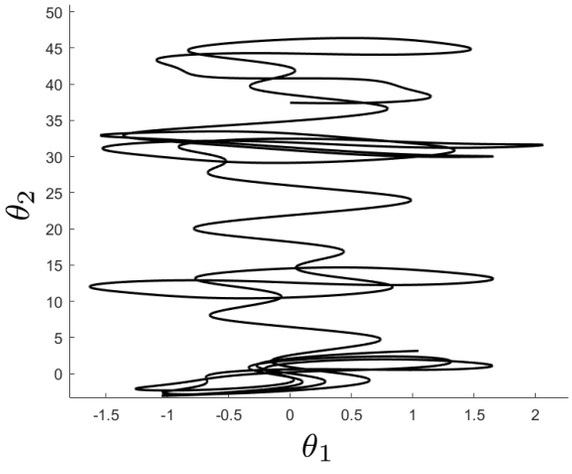
(b) $\theta_1 = \frac{\pi}{30}, \theta_2 = \frac{\pi}{2}$



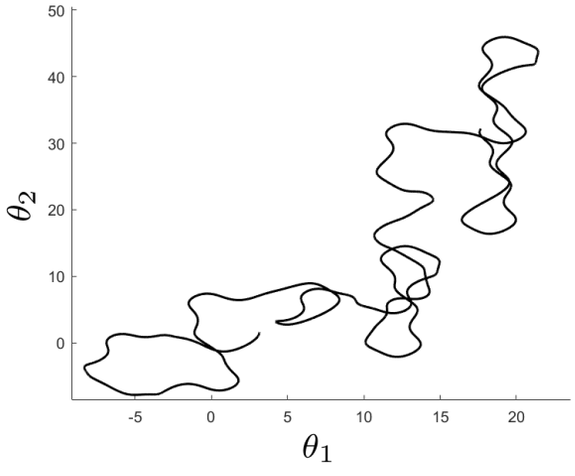
(c) $\theta_1 = \frac{\pi}{3}, \theta_2 = \frac{\pi}{6}$



(d) $\theta_1 = \frac{\pi}{3}, \theta_2 = \frac{\pi}{3}$



(e) $\theta_1 = \frac{\pi}{3}, \theta_2 = \pi$



(f) $\theta_1 = \pi, \theta_2 = \frac{\pi}{2}$

Figure 16: Final outcome of animated parametrized curves $\mathbb{R}^2$ when $x = \theta_1(t), y = \theta_2(t)$.

Problem 11

In this problem, the two double penduli were studied with slightly different initial values:

Pendulum 1:

Table 6: Initial values for Pendulum 1

$\theta_1(0)$	$\theta_1'(0)$	$\theta_2(0)$	$\theta_2'(0)$
$\frac{2\pi}{3}$	0	$\frac{\pi}{6}$	0

Pendulum 2:

Table 7: Initial values for Pendulum 2

$\theta_1(0)$	$\theta_1'(0)$	$\theta_2(0)$	$\theta_2'(0)$
$\frac{2\pi}{3}+\varepsilon$	0	$\frac{\pi}{6}+\varepsilon$	0

where $\varepsilon = 10^{-k}$ and $k \in (1,2,3,4,5)$ on the time interval [0,40]. At $t = 0$ both double penduli are at rest and almost at the same position in space.

The same constants were used from Problem 7 and 10, where the system consists of the constant mass and lengths of the pendulums

$$L_1 = L_2 = 2 \qquad m_1 = m_2 = 1$$

The variable n was chosen to be 2000 since the value gave a smoother and more stable animation with a reasonable speed than if n was 1000. When n was set to a larger value, it gave a slow animation due to the enormous amount of calculations that the computer had to perform and it took a really long time. The variables and constants are included in the main code to run the system as shown in the following figure 17.

```
%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

for k = 1:5
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)
end

%% Showing the theta with respect to time % and mean displacement lika
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

y1 = y1';
for k = 1:5
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);
y2=y2';

figure
subplot(2,1,1)
plot(t1,abs(y1(:,1)-y2(:,1)),'LineWidth',1.2) % plotting the difference between the actual theta1
    ↪ value and the theta1 with the epsilon error
ylabel('Displacement on \theta_1 angles')
xlabel('Time[s]')
legend('\theta_1 - (\theta_1 + \epsilon)','Location','best')

subplot(2,1,2)
plot(t2,abs(y1(:,3)-y2(:,3)),'r','LineWidth',1.2) % plotting the difference between the actual
    ↪ theta2 value and the theta2 with the epsilon error
ylabel('Displacement on \theta_2 angles')
xlabel('Time[s]')
legend('\theta_2 - (\theta_2 + \epsilon)','Location','best')
end
```

Figure 17: Function for problem 11

The main function is split into two sections, the first one was used for showing the graphical representation. The animations were plotted with the plot_chaos_double_pendulum.m function that can be seen in section E. The animations for each k value can be seen on this link.

The second section was used for plotting a graph comparing the displacement between the actual θ values and θ with the ε error for both angles of θ, figure 17 (line 36 and 42).

The Runge-Kutta method explained in problem 5, solves the system as done in problem 7. RK_method_double-_pendulum is working with the same principle but uses ydot_double_pedulum to calculate the ks, as shown in section C. Figure 11 shows the ydot_double_pedulum function, which consists of the system equations from problem 6 and the gravitation constant g.

To calculate the weighted average of the four slopes used in the RK-method-double-pendulum, the function RK-step, seen in figure 7, was used like in problem 5. The standard weights for the method were used in the calculations.

The graphs in figure 18 show the displacement of both θ angles with respect to time and the mean displacement as well during the time interval with different values of k. The plot above, colored in blue, shows the difference between the actual θ_1 value and the θ_1 with the ε error. And the plot below, colored in red, shows the actual θ_2 value and the θ_2 with the ε error.

When k increases the value of ε decreases since $\varepsilon = 10^{-k}$, and the error decreases. As seen in figure 18 the time it takes the two pendulums to go separate ways increases when k increases. The time was approximated and read from the graphs and inserted into table 8 to see it in numbers but the graph explains it really well also.

Table 8: Approximate time it takes the pendulums to separate

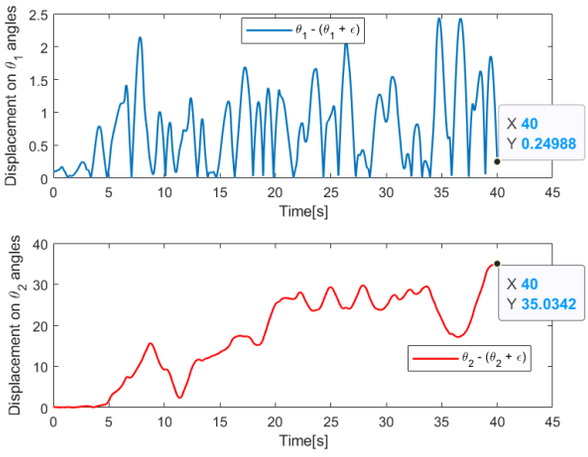
k	θ_1	θ_2
1	0 s	2.5 s
2	3 s	5 s
3	4 s	6 s
4	7 s	11 s
5	13 s	14 s

These results make sense since in reality, if $\varepsilon = 0$ there would be no difference between the two pendulums and they would follow each other the whole time. When the value of k is less, the error accumulates and at some point is so much that the penduli goes into chaos. The difference between the angles of the two pendulums is much higher, affecting each pendulum's trajectory.

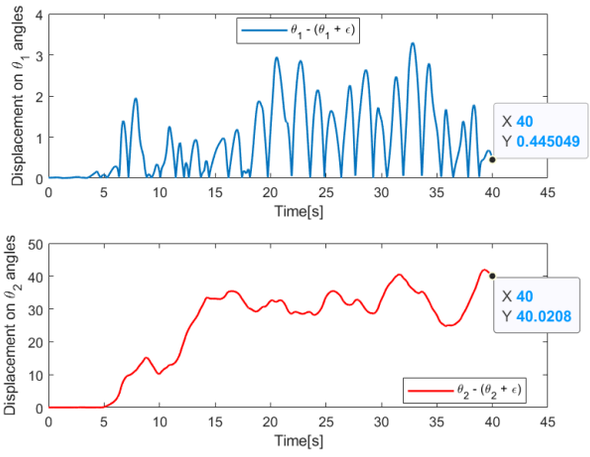
The displacement on θ (Y-value) when $T = 40s$ (X-value) can be read in the box on the right of each graph. The results have been inserted into table 9. The graph shows a much better explanation of what is happening but it is interesting to see in the table that the displacement angle θ_1 increases when k goes from 1 to 4 but the displacement of θ_2 decreases when k goes from 2 to 5 when $T = 40s$.

Table 9: Displacement of θ angles at T=40

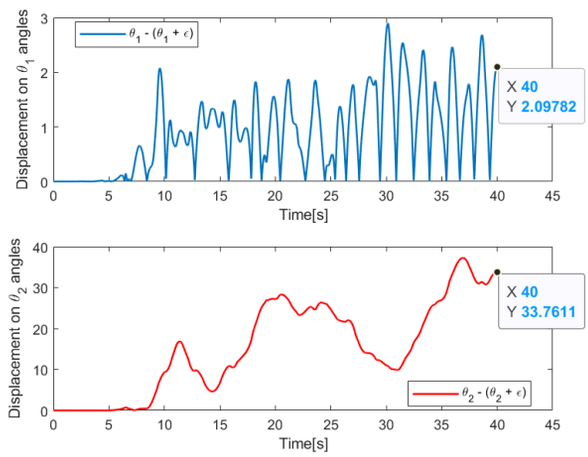
k	θ_1	θ_2
1	0.24988	35.0342
2	0.445049	40.0208
3	2.09782	33.7611
4	3.29928	26.2635
5	2.12347	10.1584



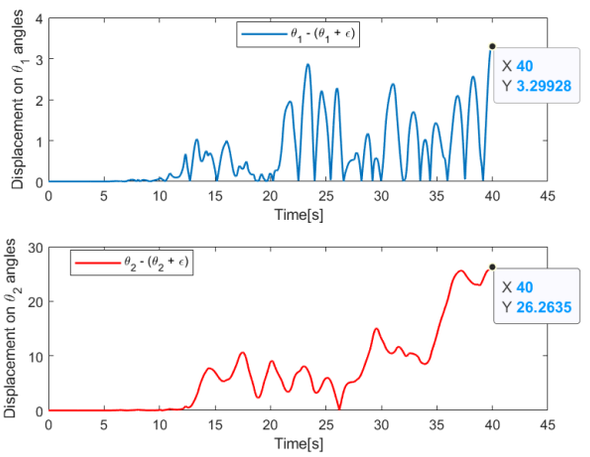
(a) k = 1



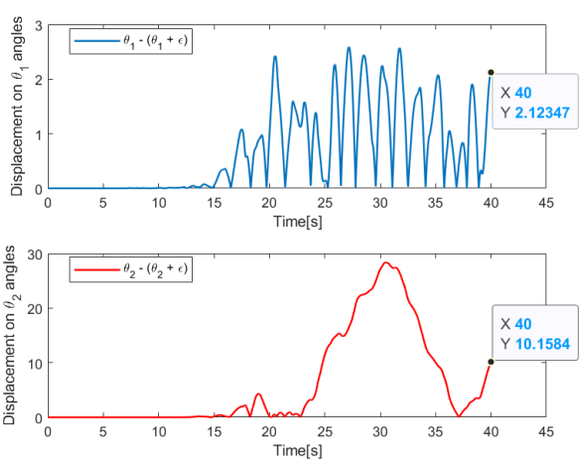
(b) k = 2



(c) k = 3



(d) k = 4



(e) k = 5

Figure 18: Displacement on both θ angles with respect to time.

Problem 12

Here, the goal was to look at the difference between both pendulum's positions at $T = 40$ when $\varepsilon = 10^{-k}$ and $k \in (1, 2, ..., 12)$ and look at the effects of three different variables: the time interval length, method precision, and the initial values.

When looking at the time interval length, the constant mass and length were the same as in previous examples, and the same initial values used as in problem 11, see tables 6 and 7. The value of n was set to 2000 and the value of T was set to 45. The same function of RK_method_double_pendulum.m was used like in previous examples and can be seen in section C.

As seen in the results in figure 19, it shows the time interval T when the displacement of the pendulums happens with respect to all values of k given. The value of k controls the ε error and when k increases the value of the error decreases and as seen in the figure, the value of the time interval, T, increases. Since the ε error decreases, the displacement of the pendulums occurs at a higher value of T as the ε decreases.

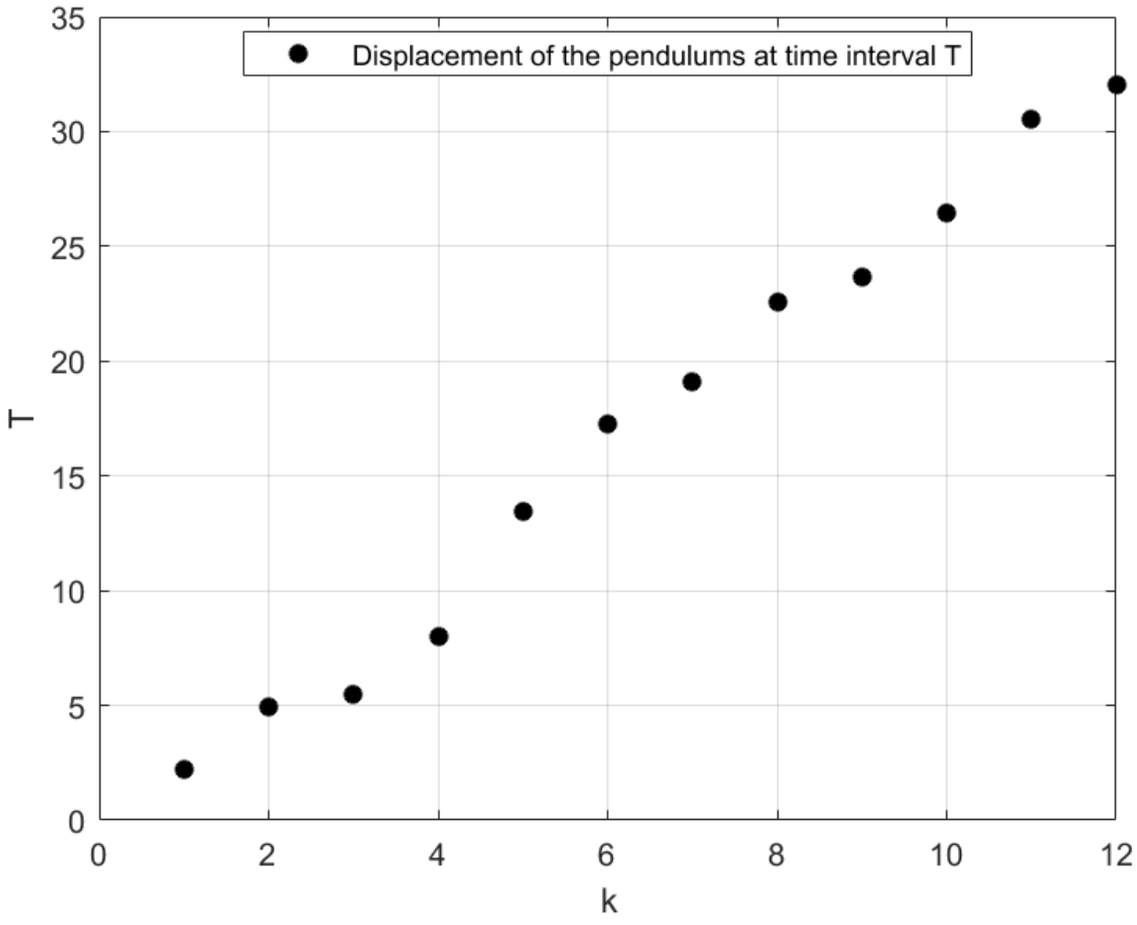


Figure 19: Displacement of the pendulums at time interval T

Next, the effects of the method precision were looked at by changing the value of n. A prediction made is that a smaller value of n results in more unreliable calculations. An experience was made with the following n values: [1000 10000 50000 100000].

The results are shown in figure 20, which shows the correlation between the value of k with respect to T for different values of n as seen in the title of each graph. As predicted, when n is increased, the result gives more reliable calculations and accurate data. The value of T was read from the figures in Matlab when $k = 12$ for each value of n and the results were put in table 10. From the table, it can be interpreted that when the value

of n increases, the time, T, it takes the two pendulums to separate increases since it is more precise.

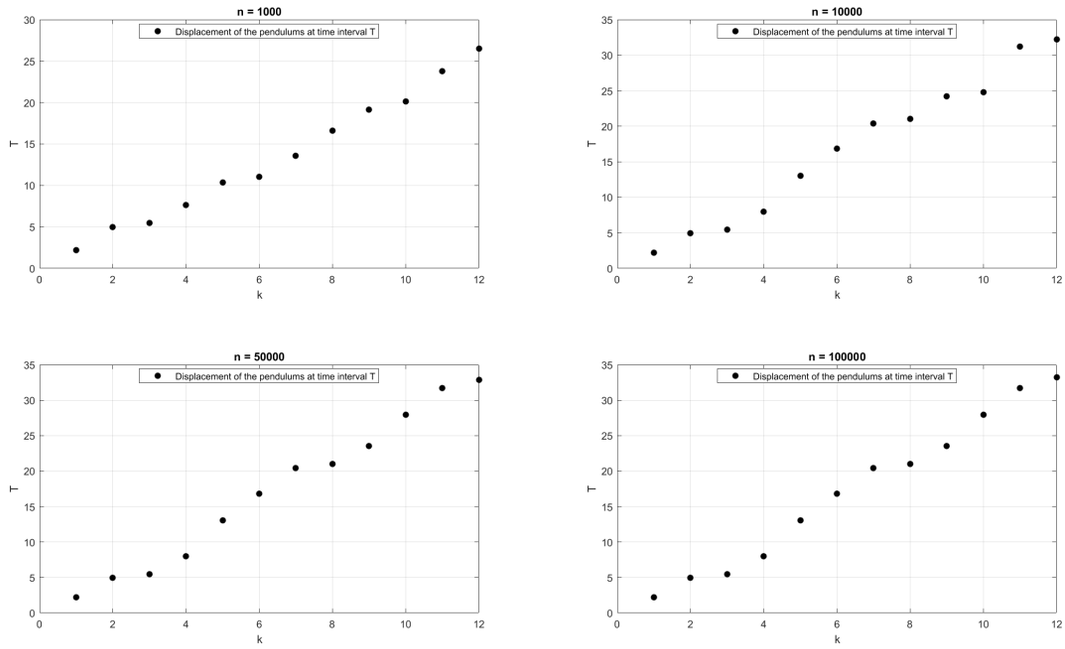


Figure 20: Displacement of the pendulums at time interval T at different values of n

Table 10: Time of displacement when $k = 12$ for different values of n

n	T
1000	26.505 s
10000	32.247 s
50000	32.8887 s
100000	33.2222 s

In the third experiment, the effect when different initial conditions were interpreted. A vector with four different initial values was used:

Table 11: Different initial values

Vector	$\theta_1(0)$	$\theta_1'(0)$	$\theta_2(0)$	$\theta_2'(0)$
1	$\frac{\pi}{2}$	0	$\frac{\pi}{6}$	0
2	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$	0
3	π	0	$\frac{\pi}{2}$	0
4	1.1π	0	π	0

and $n = 2000$ and $T = 40$.

A displacement vector was made in the code for when $k = 12$, the values of θ displacement were held which were at the time interval $T = 40s$. These values of the θ displacement were printed for each initial value vector and the results can be seen in table 12.

Table 12: Displacement of θ for different initial values

Vector	Displacement of θ
1	3.1121e-10
2	1.9565
3	15.4888
4	19.8425

As can be seen when comparing the results from table 12 with 11, when the position angles are increased as they do down the table, the displacement of θ increases. That seems right since, as θ increases, there is more energy in the system at time=0 and the harder it gets to approximate how the double pendulum behaves.

Independent work

In this section, the influence of the length and mass on the behavior of the double pendulum is determined. The initial values of θ remain at $\theta_1 = \frac{\pi}{3}$ and $\theta_2 = \frac{\pi}{4}$ while the length L_1 and L_2 and mass m_1 and m_2 are changed to the according values of table 13.

Table 13: Different initial values

No.	L_1	L_2	m_1	m_2
1	2	2	1	1
2	4	2	1	1
3	2	4	1	1
4	4	2	50	1
5	2	4	1	50

The code for this section is explained in Problem 8 and uses the same angles as Problem 8 No.4. The animation of the movement is plotted for each section including the path line as in Problem 11 and can be seen under the following link. The main code to run the parameters can be seen in section W.

No. 1

In the first example, the system is not changed to Problem 8 No.4. The double pendulum swings smoothly, with blue and red overtaking each other slowly. The system is stable by having relatively small angles because not much energy is transferred or stored inside the system. The path of the blue ball is always changing on both returns, while the position around the midpoint seems to look very similar in each round. This stable characteristic of the system is also represented by the following figure 21.

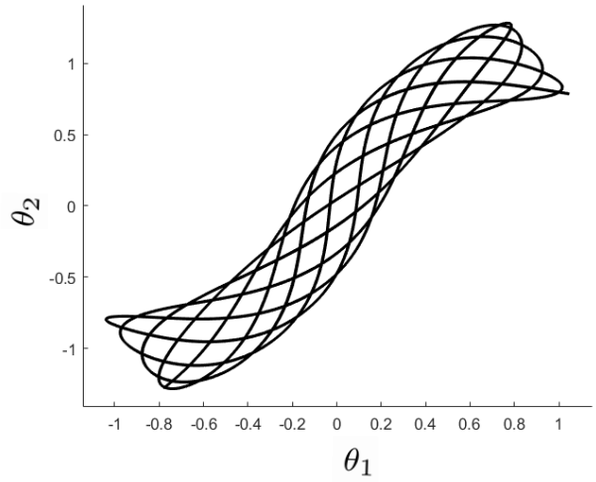


Figure 21: Parametrized curve with initial values from no.1

No. 2

In the second example, red and blue are remaining the same distance from each other while the length of the rope attaching red to the central rotation point is doubled. Making this distance larger results in a smoother swing motion. The motion of both balls is very periodic, meaning even blue is almost following one path. The path is concentric to red's radius in the middle section and seems almost linear to both sides with a little radius at the return points. This is because blue gets first pulled by red and then overtakes after the middle point in each repetition. The periodic characteristic of the system is also shown in figure 22.

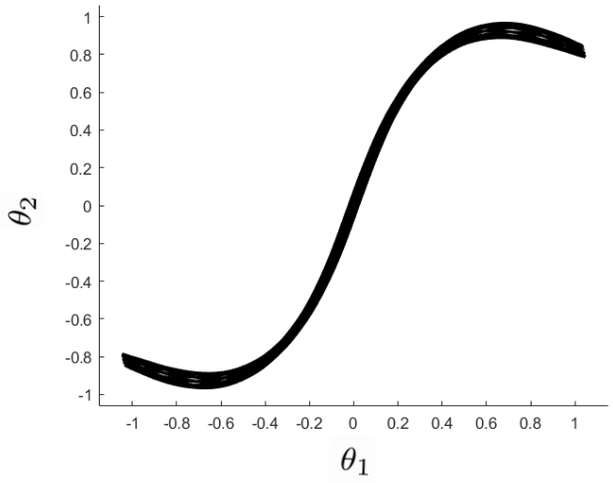


Figure 22: Parametrized curve with initial values from no.2

No. 3

In No. 3, the distance between the balls is doubled, while the distance to the attachment point remains the same as in No.1. Changing this distance leads to a similar motion as described in No.2 with blue getting first pulled by red and then overtaking after the middle. However, with a greater distance between the balls, blue's path looks different. On both sides, there is one loop connected by the crossing point, which is shifted slightly to the left. The bottom section of the path seems to have a circular shape, while both the top section occurs when the blue ball gets pulled along and overtakes faster than in the previous examples. The periodic characteristic of the system is also shown in figure 23.

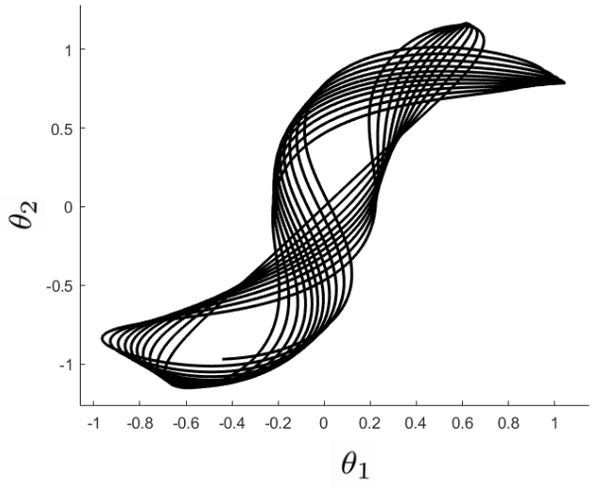


Figure 23: Parametrized curve with initial values from no.3

No. 4

The fourth example is with double the distance to the attachment point and 50 times heavier red ball. In comparison with No. 2, the change in weight influenced the system dramatically. The red ball remains stable, almost unaffected by the blue ball, while the blue circles red multiple times and even changes directions. When the movement seems random at first, the pattern repeats itself, shifting the motion of the blue ball marginally to the left. This is because of the steady motion of the red ball, with minor changes because of blue. The effect of the added weight can be seen when comparing figure 24 to the previous figures, this one being the most chaotic.

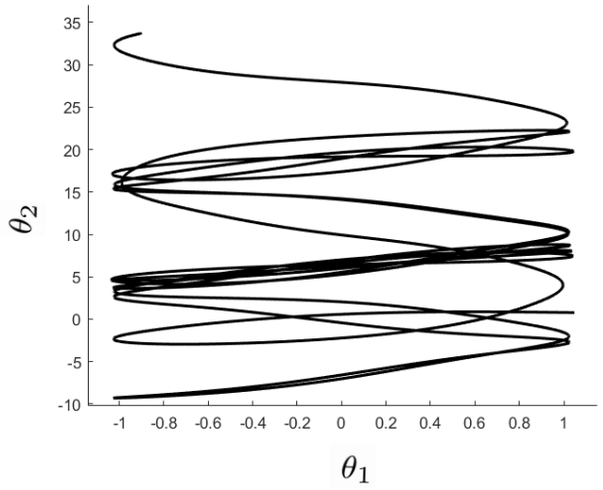


Figure 24: Parametrized curve with initial values from no.4

No. 5

In the last example, the distance between the two balls is doubled and the weight of the blue is 50 times higher than the red. This setup leads to a system where the blue ball is constantly moving forward, whereas the red ball is behaving like a spring. The red ball moves back and forth causing the blue ball to accelerate and stop in small steps back and forth. This motion creates two almost concentric circles of red and blue. As seen in the video the blue path is thicker than the red path, this is because of the noise which is created when slowing down and accelerating. Both rods are then not parallel causing blue to be closer to the attachment point. The system in figure 25 is also periodic but the angles change more frequently compared to previous figures.

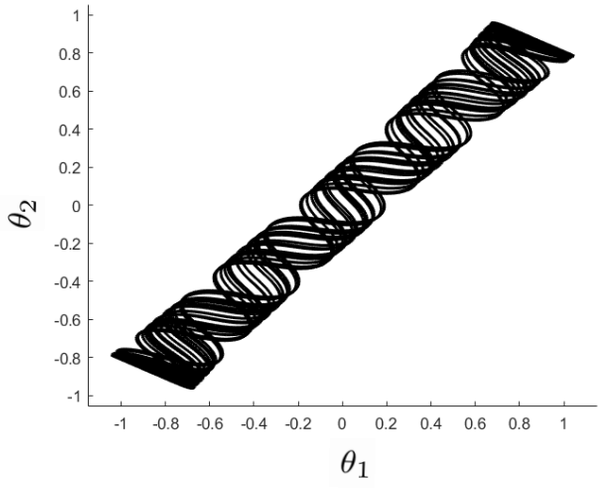


Figure 25: Parametrized curve with initial values from no.5

References

[1] Mathematics Stack Exchange, “ordinary differential equations - explanation and proof of the 4th order runge-kutta method - mathematics stack exchange,” 02.12.2022. [Online]. Available: https://math.stackexchange.com/questions/528856/explanation-and-proof-of-the-4th-order-runge-kutta-method

A Plot theta

```
function plot_theta(t,y)

plot(t,y(1,:))
title('Theta with respect to time')
xlabel('Time')
ylabel('Theta')

end
```

B Plot pendulum

```
function plot_pendulum(y,L)
theta = y(1,:);

x_0 = 0; %We want to pendulum to be fixed at (0,0)
y_0 = 0;
b = animatedline('Color','b','Marker','.','MarkerSize',35);
h = animatedline('Color','k','LineWidth',3);

axis([-3 3 -3 3])
x_1 = L.*sin(theta); %trigonometric functions
w_1 = -L.*cos(theta);
hold on
title('Pendulum animation')
plot([-0.5,0.5],[0,0],'linewidth',8, 'color','k')

for i=1:length(theta)

    addpoints(b,x_1(i),w_1(i))
    addpoints(h,[x_0 x_1(i)],[y_0 w_1(i)])
    drawnow
    clearpoints(h)
    clearpoints(b)

end
end
```

C Runge-Kutta method with double pendulum

```
function [t y] = RK_method_double_pendulum(x,n,T,L1,L2,m1,m2)
h=T/n;
t(1)=0;
y=x; %To get the initial conditions


for i=1:n

    k1 = ydot_double_pedulum(t(i),y(:,i),L1,L2,m1,m2); %slope 1
    k2 = ydot_double_pedulum(t(i)+h/2,y(:,i)+(h/2).*k1,L1,L2,m1,m2); %slope 2
    k3 = ydot_double_pedulum(t(i)+h/2,y(:,i)+(h/2).*k2,L1,L2,m1,m2); %slope 3
    k4 = ydot_double_pedulum(t(i)+h,y(:,i)+h.*k3,L1,L2,m1,m2); %slope 4
    y(:,i+1)=RK_step(y(:,i),h,k1,k2,k3,k4);
    t(i+1)=t(i)+h;
```

```

end

end
```

D Plot double pendulum

```
function plot_double_pendulum(y,L1,L2)
theta1 = y(1,:);
theta2 = y(3,:);

x_0_ball_1 = 0; %We want to pendulum to be fixed at (0,0)
y_0_ball_1 = 0;

line_1 = animatedline('Color','k','LineWidth',3);
line_2 = animatedline('Color','k','LineWidth',3);
ball_2 = animatedline('Color','b','Marker','.','MarkerSize',35);
path_2 = animatedline('Color','b');
ball_1 = animatedline('Color','r','Marker','.','MarkerSize',50);
path_1 = animatedline('Color','r');

axis([-5 5 -5 5])
x_1 = L1*sin(theta1); %trigonometric functions
w_1 = -L1*cos(theta1);

x_2 = L1*sin(theta1)+L2*sin(theta2); %trigonometric functions
w_2 = -L1*cos(theta1)-L2*cos(theta2);

hold on
title('Double pendulum animation')

plot(0,0,'.','MarkerSize',35, 'color','k')

for i=1:length(theta1)

    addpoints(line_1,[x_0_ball_1 x_1(i)],[y_0_ball_1 w_1(i)])
    addpoints(ball_1,x_1(i),w_1(i))

    addpoints(line_2,[x_1(i) x_2(i)],[w_1(i) w_2(i)])
    addpoints(ball_2,x_2(i),w_2(i))
    addpoints(path_2,x_2(i),w_2(i))

    addpoints(path_1,x_1(i),w_1(i))

    drawnow
    clearpoints(line_1)
    clearpoints(ball_1)
    clearpoints(line_2)
    clearpoints(ball_2)

end

end
```

E Plot chaos double pendulum

```
function plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)

theta1 = y1(1,:);
theta2 = y1(3,:);

theta11 = y2(1,:);
theta22 = y2(3,:);

figure %open figure for animation

x_0_ball_1 = 0; %We want to pendulum to be fixed at (0,0)
y_0_ball_1 = 0;

%%%%%%%%% FOR PENDULI 1 %%%%%%%%%%
line_1 = animatedline('Color','k','LineWidth',3);
line_2 = animatedline('Color','k','LineWidth',3);
ball_2 = animatedline('Color','b','Marker','.','MarkerSize',35);
path_2 = animatedline('Color','b');
ball_1 = animatedline('Color','r','Marker','.','MarkerSize',50);
path_1 = animatedline('Color','r');

%%%%%%%%% FOR PENDULI 2 %%%%%%%%%%
line_11 = animatedline('Color','k','LineWidth',3);
line_22 = animatedline('Color','k','LineWidth',3);
ball_22 = animatedline('Color','b','Marker','.','MarkerSize',35);
path_22 = animatedline('Color','b');
ball_11 = animatedline('Color','r','Marker','.','MarkerSize',50);
path_11 = animatedline('Color','r');

%%%%%%%%% FOR PENDULI 1 %%%%%%%%%
axis([-5 5 -5 5])
x_1 = L1*sin(theta1); %trigonometric functions
w_1 = -L1*cos(theta1);
x_2 = L1*sin(theta1)+L2*sin(theta2); %trigonometric functions
w_2 = -L1*cos(theta1)-L2*cos(theta2);

%%%%%%%%% FOR PENDULI 2 %%%%%%%%%
axis([-5 5 -5 5])
x_11 = L1*sin(theta11); %trigonometric functions
w_11 = -L1*cos(theta11);
x_22 = L1*sin(theta11)+L2*sin(theta22); %trigonometric functions
w_22 = -L1*cos(theta11)-L2*cos(theta22);

hold on
title('Double pendulum animation')

plot(0,0,'.','MarkerSize',35, 'color','k')


%vidObj = VideoWriter('Chaos');
%vidObj.FrameRate = n/T; %frame rate set to number of steps/time interval
%open(vidObj) %open the video object

for i=1:length(theta1)

    addpoints(line_1,[x_0_ball_1 x_1(i)],[y_0_ball_1 w_1(i)])
    addpoints(ball_1,x_1(i),w_1(i))
```

```
    addpoints(line_2,[x_1(i) x_2(i)],[w_1(i) w_2(i)])
    addpoints(ball_2,x_2(i),w_2(i))
    addpoints(path_1,x_1(i),w_1(i))
    addpoints(path_2,x_2(i),w_2(i))

    addpoints(line_11,[x_0_ball_1 x_11(i)],[y_0_ball_1 w_11(i)])
    addpoints(ball_11,x_11(i),w_11(i))
    addpoints(line_22,[x_11(i) x_22(i)],[w_11(i) w_22(i)])
    addpoints(ball_22,x_22(i),w_22(i))
    addpoints(path_11,x_11(i),w_11(i))
    addpoints(path_22,x_22(i),w_22(i))
    drawnow
    %frame = getframe(gcf); %capture figure as a movie frame
    %writeVideo(vidObj, frame) %writes the movie frames from the getframe function

    %%% 1 %%%
    clearpoints(line_1)
    clearpoints(ball_1)
    clearpoints(line_2)
    clearpoints(ball_2)

    %%% 2 %%%
    clearpoints(line_11)
    clearpoints(ball_11)
    clearpoints(line_22)
    clearpoints(ball_22)

end

%close(vidObj) %close the video object

end
```

F Eulers function

```
function [t y]=Euler_func_multi(x,n,T)
h=T/n;
t(1)=0;
y=[x(1); x(2)]; %To get the initial conditions

for i=1:n
    f = ydot(t(i),y(:,i)); %Now this is a vector
    y(:,i+1)=eulerstep(t(i),y(:,i),h);  %y(:,i)+f'.*h %switching f because its a column
    t(i+1)=t(i)+h;
end
end
```

G Eulers step function

```
function y=eulerstep(t,x,h)
y=x+h.*ydot(t,x);
end
```

H Code of Runge-Kutta error calculation

```
function [error n_vec] = RK_calc_errors(y0,n,T,L1,L2,m1,m2);
n_theoretical = 30000; %many intervals so very precise values
[t y_true_val] = RK_method_double_pendulum(y0,n_theoretical,T,L1,L2,m1,m2); %y_true_val is the true
    ↪ value (theoretical)
y_true_val = y_true_val'; %transpose the vectors

true_val = y_true_val(end,:); %we take the last line and all the colums to get the biggest error
for i = 1:7
    [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
    y = y'; %transpose the vector
    error(i) = norm(y(end,:) - true_val); %error of the norm of the all the final values of the
    ↪ thetas
    n_vec(i) = n; % keep all the different n values that are used
    n = n*2;
end
end
```

I Code of Runge-Kutta double pendulum calculation

```
function [t y] = RK_method_double_pendulum(x,n,T,L1,L2,m1,m2)
h=T/n;
t(1)=0;
y=x; %To get the initial conditions


for i=1:n

    k1 = ydot_double_pedulum(t(i),y(:,i),L1,L2,m1,m2); %slope 1
    k2 = ydot_double_pedulum(t(i)+h/2,y(:,i)+(h/2).*k1,L1,L2,m1,m2); %slope 2
    k3 = ydot_double_pedulum(t(i)+h/2,y(:,i)+(h/2).*k2,L1,L2,m1,m2); %slope 3
    k4 = ydot_double_pedulum(t(i)+h,y(:,i)+h.*k3,L1,L2,m1,m2); %slope 4
    y(:,i+1)=RK_step(y(:,i),h,k1,k2,k3,k4);
    t(i+1)=t(i)+h;

end

end
```

J Code of Runge-Kutta single pendulum calculation

```
function [t y] = RK_method_single_pendulum(x,n,T)

h=T/n;
t(1)=0;
y=x; %To get the initial conditions


for i=1:n

    k1 = ydot(t(i),y(:,i)); %slope 1
    k2 = ydot(t(i)+h/2,y(:,i)+(h/2).*k1); %slope 2
    k3 = ydot(t(i)+h/2,y(:,i)+(h/2).*k2); %slope 3
    k4 = ydot(t(i)+h,y(:,i)+h.*k3); %slope 4
```

```
    y(:,i+1)=RK_step(y(:,i),h,k1,k2,k3,k4);
    t(i+1)=t(i)+h;

end



end
```

K Code of y prime (single pendulum) calculations

```
function z=ydot(t,y)
g=9.81; L=2;
z(1,:) = y(2); %From problem 1 we can see that the first line in Z_vec is equal to y_prime
z(2,:) = -g/L*sin(y(1)); %From problem 1 we can see that the second line in Z_vec is equal to this
    ↪ equation
end
```

L Code of y prime (double pendulum) calculations

```
function z = ydot_double_pedulum(t,y,L1,L2,m1,m2)
g=9.81;
delta = y(3)-y(1);
w1 = y(2);
w2 = y(4);
theta1 = y(1);
theta2 = y(3);

z(1,:) = w1;
z(2,:) = (m2*L1*w1^2*sin(delta)*cos(delta)+m2*g*sin(theta2)*cos(delta)+m2*L2*w2^2*sin(delta)-(m1+m2)
    ↪ *g*sin(theta1))/((m1+m2)*L1-m2*L1*cos(delta)^2);
z(3,:) = w2;
z(4,:) = (-m2*L2*w2^2*sin(delta)*cos(delta)+(m1+m2)*(g*sin(theta1)*cos(delta)-L1*w1^2*sin(delta)-g*
    ↪ sin(theta2)))/((m1+m2)*L2-m2*L2*cos(delta)^2);

end
```

M Code of animating the curve

```
function animated_curve(y,n,T) %we need to call n and T for the FrameRate

figure %open figure for animation
path = animatedline('Color','k','LineWidth',1.5);

axis([min(y(1,:)) max(y(1,:)) min(y(3,:)) max(y(3,:))]*1.1) %to get always the right fixed axis

%vidObj = VideoWriter('Parametrized curve'); %create a video object, it will create an AVI file
%vidObj.FrameRate = n/T; %frame rate set to number of steps/time interval
%open(vidObj) %open the video object

for i = 1:length(y(1,:))
    addpoints(path,y(1,i),y(3,i));
    %pause(0.05)
```

```
    drawnow

    %frame = getframe(gcf); %capture figure as a movie frame
    %writeVideo(vidObj, frame) %writes the movie frames from the getframe function

end
%close(vidObj) %close the video object
end
```

N Code of problem 3

```
clear all; close all; clc;

x=[pi/12;0]; %Initial conditions
T=20;
n=500;
L=2
[t, y]=Euler_func_multi(x,n,T);

plot_theta(t,y)%plot theta
hold off
figure
plot_pendulum(y,L) %plot the pendulum
```

O Code of Problem 4

```
%% Problem 4
clear all; close all; clc;

x=[pi/2;0]; %Initial conditions
T=20;
n=500;
L=2;

[t,y]=Euler_func_multi(x,n,T)

plot_theta(t,y)%plot theta
hold off
figure
plot_pendulum(y,L) %plot the pendulum
```

P Code of problem 5

```
clear all; close all; clc;

% Problem 5.1
x=[pi/12;0]; %Initial conditions
T=20;
n=500;
L = 2;

[t, y] = RK_method_single_pendulum(x,n,T);

```

```
plot_theta(t,y)
figure
plot_pendulum(y,L)

%% Problem 5.2
x=[pi/2;0]; %Initial conditions
T=20;
n=500;
L = 2;

[t, y] = RK_method_single_pendulum(x,n,T);

plot_theta(t,y)
figure
plot_pendulum(y,L)
```

Q Code of problem 7 and 8

```
clear all; close all; clc;
L1=2; L2=2; m1=1; m2=1;T=20;

y0 = [pi/3;0;pi/6;0];
n = 350;
T = 20;

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
plot_double_pendulum(y,L1,L2)
```

R Code of problem 9

```
%% Problem 9
clear all; close all; clc;

L1=2; L2=2; m1=1; m2=1;T=20;

y0 = [pi/3;0;pi/6;0]; %Initial values like in problem 7
%rand    upphafsgildin max og min pi/2
n = 100;
T = 20;

[error,n_vec] = RK_calc_errors(y0, n, T, L1, L2, m1, m2);

% Do best fit on errors on loglog plot
coefficients = polyfit(log(n_vec),log(error),1);
y1 = polyval(coefficients,log(n_vec));

plot(log(n_vec),log(error),'*')
hold on
plot(log(n_vec),y1)
xl = xlim;
yl = ylim;
xt = 0.7 * (xl(2)-xl(1)) + xl(1);
yt = 0.75 * (yl(2)-yl(1)) + yl(1);
caption = sprintf('y = %.2f * x + %.2f', coefficients(1), coefficients(2));
text(xt, yt, caption, 'FontSize', 9, 'Color', 'k');
```

```
ylabel('log(Error)')
xlabel('log(n)')
legend('Errors','Fitted line')
```

S Code of problem 10

```
%% problem 10   Run for different initial values and analyse
close all; clear all;clc;

L1=2; L2=2; m1=1; m2=1;T=50;

y0 = [pi/30;0;pi/30;0];   %almost a perfect rectangle
n = 10000;


[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate
%plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate



%% 10.1a    %increase theta2
y0 = [pi/30;0;pi/10;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.1b    %increase theta2 more
y0 = [pi/30;0;pi/2;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.2a   %increase theta1
y0 = [pi/10;0;pi/30;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.2b    %increase theta1 more
y0 = [pi/2;0;pi/30;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.3   %bigger angles
y0 = [pi/3;0;pi/6;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.4
y0 = [pi/1.5;0;pi/6;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

```

```
%% 10.5
y0 = [pi/3;0;pi/3;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.6
y0 = [pi/3;0;pi;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate

%% 10.7
y0 = [pi;0;pi/2;0];

[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
animated_curve(y,n,T) %we need to call n and T for the FrameRate
%plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
```

T Code of problem 11

```
%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

for k = 1:5
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)
end

%% Showing the theta with respect to time % and mean displacement lika
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

y1 = y1';
for k = 1:5
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);
y2=y2';

figure
subplot(2,1,1)
```

```
plot(t1,abs(y1(:,1)-y2(:,1)),'LineWidth',1.2) % plotting the difference between the actual theta1
    ↪ value and the theta1 with the epsilon error
ylabel('Displacement on \theta_1 angles')
xlabel('Time[s]')
legend('\theta_1 - (\theta_1 + \epsilon)','Location','best')

subplot(2,1,2)
plot(t2,abs(y1(:,3)-y2(:,3)),'r','LineWidth',1.2) % plotting the difference between the actual
    ↪ theta2 value and the theta2 with the epsilon error
ylabel('Displacement on \theta_2 angles')
xlabel('Time[s]')
legend('\theta_2 - (\theta_2 + \epsilon)','Location','best')
end
```

U Code of problem 11 creating the video

```
%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION when k=1
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

k = 1;
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)


%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION when k=2
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

k = 2;
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)

%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION when k=3
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

```

```
[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

k = 3;
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)

%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION when k=4
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

k = 4;
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)

%% Problem 11 SHOWING THE GRAPHICAL REPRESENTATION when k=5
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 40;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

k = 5;
y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
[t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);

plot_chaos_double_pendulum(y1,y2,L1,L2,n,T)
```

V Code of problem 12

```
%% Problem 12 - TIME INTERVAL LENGTH
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = 2000;
T = 45;

[t1 y1] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);

y1 = y1';
counter = 1;

for k = 1:12
    y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0];
```

```
    [t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);
    y2=y2';

    for i = 1:length(y2(:,1))
        if abs(y1(i,3)-y2(i,3)) > 0.1 %there is always a minor displacement so we decided to detect
    ↪ when the displacement of the angles is more than 0.1
            time_vec(counter) = t2(i);
            counter = counter + 1;
            break;
        end
    end
    k_vec(k) = k;
end


plot(k_vec,time_vec,'k*','LineWidth',2)
grid on
xlabel('k')
ylabel('T')
legend('Displacement of the pendulums at time interval T','Location','Best')

%% PROBLEM 12 - METHOD PRECISION (CHANGING THE VALUE OF N)
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [2*pi/3;0;pi/6;0];
n = [1000 10000 50000 100000];
T = 45;


counter = 1;
for j = 1:4
    [t1 y1] = RK_method_double_pendulum(y0,n(j),T,L1,L2,m1,m2); % Pendulum nr 1
    y1 = y1';
    for k = 1:12
        y00 = [(2*pi/3)+10^(-k); 0; (pi/6)+10^(-k); 0]; %Pendulum nr 2
        [t2 y2] = RK_method_double_pendulum(y00,n(j),T,L1,L2,m1,m2);
        y2=y2';

        for i = 1:length(y2(:,1))
            if abs(y1(i,3)-y2(i,3)) > 0.1 %there is always a minor displacement so we decided to
    ↪ detect when the displacement of the angles is more than 0.1
                time_vec(counter) = t2(i);
                counter = counter + 1;
                break;
            end
        end
        k_vec(k) = k;
    end
    subplot(2,2,j)
    plot(k_vec, time_vec,'k*','LineWidth',2)
    counter = 1;
    grid on
    xlabel('k')
    ylabel('T')
    legend('Displacement of the pendulums at time interval T','Location','Best')
    title(['n = ',num2str(n(j))]) %prints the title with each n used
end

```

```
%% Problem 12 - WITH DIFFERENT INITIAL POSITIONS
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;
y0 = [pi/2 0 pi/6 0 ; pi/2 0 pi/2 0 ; pi 0 pi/2 0; 1.1*pi 0 pi 0]';
n = 2000;
T = 40;


counter = 1;
for j = 1:4
[t1 y1] = RK_method_double_pendulum(y0(:,j),n,T,L1,L2,m1,m2);
y1 = y1';
    for k = 1:12
        y00 = [(y0(1,j))+10^(-k); 0; (y0(3,j))+10^(-k); 0];
        [t2 y2] = RK_method_double_pendulum(y00,n,T,L1,L2,m1,m2);
        y2=y2';
        k_vec(k) = k;
        if k == 12
            displacement_vec(counter) = abs(y2(end,3)-y1(end,3))
        end

    end
    counter = counter + 1;
end
```

W Independent Task Main code

```
%% 1. Independent assignment NORMAL
close all; clear all; clc;

L1=2; L2=2; m1=1; m2=1;T=50;

y0 = [pi/3;0;pi/4;0];
n = 2000;


[t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
animated_curve(y,n,T)

%HERE IS A DEMONSTRATION ON HOW THE DOUBLE PENDULUM BEHAVES IF THE INITIAL VALUES ARE THE SAME AS IN
    ↪ EXAMPLE 8.4
%AND THE MOVEMENT IS NOT VERY CHAOTIC AND SINCE THE INITIAL ANGLE IS NOT TO
%BIG THE MOVEMENT IS PREDICTABLE

%IN THE NEXT EMAPLES WE ALLWAYS USE THE SAME INITIAL VALUES ON THE POSITION
%% 2. Independent assignment with L1 longer than L2
close all; clear all; clc;

L1=4; L2=2; m1=1; m2=1;T=50;

y0 = [pi/3;0;pi/4;0];
n = 2000;
```

```
26
27
28 [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
29 plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
30 animated_curve(y,n,T)
31
32 %HERE THE MOVEMENT OF THE PENDULUM IS SHOWN WITH THE L1 LONGER THAN L2.
33 %THE MOVEMENT OF THE DOUBLE PENDULUM IS NOT AS CHAOTIC AS THE PENDULUM WITH
34 %THE NORMAL INITIAL VALUES AND IS VERY PERIODIC AS SHOWN IN THE GRAPH
35 %(NEARLY GOES ALLWAS IN THE SAME PATH)
36
37
38 %% 3. Independent assignment with L2 longer than L1
39 close all; clear all; clc;
40
41 L1=2; L2=4; m1=1; m2=1;T=50;
42
43 y0 = [pi/3;0;pi/4;0];
44 n = 2000;
45
46
47 [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
48 plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
49 animated_curve(y,n,T) %we need to call n and T for the FrameRate
50
51 %HERE THE MOVEMENT OF THE PENDULUM IS SHOWN WITH THE L2 LONGER THAN L1. THE
52 %MOVEMENT OF THE DOUBLE PENDULU BECOMES MORE CHAOTIC THAN IF L1 IS LONGER
53 %THAN THE L2 BUT STILL IT IS PREDICTABLE AND HAS A NICE PERIODIC MOVEMENT
54 %AS SHOWN IN THE GRAPH (ANIMATED CURVE)
55
56 %% 4. Independent assignment with L1 longer than L2 AND m1 has more weight than m2
57 close all; clear all; clc;
58
59 L1=4; L2=2; m1=50; m2=1;T=50;
60
61 y0 = [pi/3;0;pi/4;0];
62 n = 5000;
63
64
65 [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
66 plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
67 animated_curve(y,n,T)
68
69 %MORE CHAOS HAPPENS AND THE LOWER PENDULUM GOES IN MANY CIRCLES MORE
70 %UNPREDICTABLE
71
72 %% 5. Independent assignment with L2 longer than L1 AND m2 has more weight than m1
73 close all; clear all; clc;
74
75 L1=2; L2=4; m1=1; m2=50;T=50;
76
77 y0 = [pi/3;0;pi/4;0];
78 n =5000;
79
80
81 [t y] = RK_method_double_pendulum(y0,n,T,L1,L2,m1,m2);
82 plot_double_pendulum(y,L1,L2,n,T) %we need to call n and T for the FrameRate
83 animated_curve(y,n,T) %we need to call n and T for the FrameRate
84
```

```
85 %PERIODIC AND TRYES TO BE LIKE A STICK WITH NO LEGAMENT....
```