Warsaw University of Technology

FACULTY OF
ELECTRONICS AND INFORMATION TECHNOLOGY

Assignment C: Numerical methods for solving systems of IVPs.

Mahmoud Elshekh Ali
Student index number: 323930

June 11, 2024

Course ENUME spring semester 2023/2024

Supervisor dr inż. Jakub Wagner

Contents

1 Mathematical Symbols And Notations

- x -Error free independent scalar variable.
- $\mathbf{x}$ -Error free independent vector variable.
- y -Error free dependent scalar variable.
- $\mathbf{y}$ -Error free dependent vector variable.

2 Introduction

- The purpose of this experiment is to analyze a few methods for solving systems of initial value problems (IVPs) numerically. Differential equations play a very important role in the modelling of many systems such as mechanical systems, circuits, thermal systems, fluid dynamics, etc. However, the difficulty of working with differential equations is that sometimes they are either very difficult to solve or even outright unsolvable. For this reason, many numerical methods have been developed over time to obtain a numerical estimate of the solution to such equations; which is often sufficient for practical uses. This of course extends to systems of differential equations too. One of these such systems of equations is the famous three-body problem, wherein the motion of three bodies that all gravitationally attract one another (the gravitational force is described by Newton's law of gravitation) is modelled using the solution to a system of 9 ODEs. One of the reasons this problem is so famous is the fact that it is (as of the time of writing this document) unsolvable. In this experiment, a simplified version of the three-body problem, called the planar three-body problem, due to the fact that the motion of the three bodies is simplified to 2 dimensions, is solved using 4 different numerical methods. The set of ODEs that make up the planar three-body problem are:

$$\begin{cases} \frac{d^2x_1(t)}{dt^2} = -Gm_2\frac{x_1(t)-x_2(t)}{r_{12}^3(t)} - Gm_3\frac{x_1(t)-x_3(t)}{r_{31}^3(t)} \\ \frac{d^2y_1(t)}{dt^2} = -Gm_2\frac{y_1(t)-y_2(t)}{r_{12}^3(t)} - Gm_3\frac{y_1(t)-y_3(t)}{r_{31}^3(t)} \\ \frac{d^2x_2(t)}{dt^2} = -Gm_3\frac{x_2(t)-x_3(t)}{r_{23}^3(t)} - Gm_1\frac{x_2(t)-x_1(t)}{r_{12}^3(t)} \\ \frac{d^2y_2(t)}{dt^2} = -Gm_3\frac{y_2(t)-y_3(t)}{r_{23}^3(t)} - Gm_1\frac{y_2(t)-y_1(t)}{r_{12}^3(t)} \\ \frac{d^2x_3(t)}{dt^2} = -Gm_1\frac{x_3(t)-x_1(t)}{r_{31}^3(t)} - Gm_2\frac{x_3(t)-x_2(t)}{r_{23}^3(t)} \\ \frac{d^2y_3(t)}{dt^2} = -Gm_1\frac{y_3(t)-y_1(t)}{r_{31}^3(t)} - Gm_2\frac{y_3(t)-y_2(t)}{r_{23}^3(t)} \end{cases} \tag{1}$$

Where:

- t is the independent variable representing time.
- $x_k(t), y_k(t)$ are the x and y coordinates of the k-th body respectively, where $k = 1, 2, 3$.
- G is the gravitational constant.
- m_k is the mass of the k-th body, where $k = 1, 2, 3$.
- $r_{jk}(t) = \sqrt{[x_k(t) - x_j(t)]^2 + [y_k(t) - y_j(t)]^2}$ for $j, k = 1, 2, 3$.

3 Methodology and results of experimentation

In order to properly explain the numerical methods used in this experiment, a brief introduction into the numerical methods used to solve systems of IVPs is provided here:

- Numerical methods for use in solving systems of IVPs are recursive algorithms which return approximations of the solutions to such systems in the form of matrices, where the columns correspond to discrete points in a finite interval of the domain of the approximated solution, and the rows correspond to the approximate values of the solution at these discrete points in the domain. These algorithms use a combination of numercial differentiation methods and the vectors already known or calculated to find the next vector in the matrix, i.e. **[1]**:

$$\mathbf{y_n} = \xi(\mathbf{y}_{n-1}, \mathbf{y}_{n-2}, ...) \text{ for } n = 1, 2, ..., N.$$

$$\frac{d\mathbf{y}(x)}{dy} = \mathbf{f}(x, \mathbf{y}(x))$$

Where ξ is an operator describing the recursive algorithm, and $\mathbf{y_n}$ is the obtained estimate of $\mathbf{y}(x_n) = \mathbf{y}(n\Delta x)$. $x_n \in \{x_0, x_1, ..., x_N\}$ where $x_k - x_j = \Delta x, k = 1, ..., N$ and $j = 0, ..., N-1$. $\mathbf{f}$ is the vector of numerical differentiation formulas.

- Numerical methods for use in solving IVPs are classified into 2 general categories: single-step methods and multi-step methods. Single step methods are methods where the operator ξ is a function of $\mathbf{y}_{n-1}$ only. Multi-step methods are methods where the operator ξ is a function of $\mathbf{y}_{n-1}, \mathbf{y}_{n-2}, ..., \mathbf{y}_{n-K}$ where $K \geq 1$. Both of these categories can be sub-divided into two subcategories: explicit and implicit methods. Implicit methods are those where the operator ξ is also a function of $\mathbf{y}_n$, otherwise it is called explicit. The methods used in this document are all explicit methods.

3.1 Methodology, assumptions, limitations, etc.

- Within this experiment, 4 different explicit numerical methods are used to find the solution to the planar three-body problem:

1. The *MATLAB* environment built-in function ode45 (4th order Runge-Kutta method with adaptive step), with the absolute and relative tolerance set to 10^{-12}.

2. The 2nd order Adam-Bashford method, where:
$y_n = y_{n-1} + \frac{1}{2}\Delta t(3f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$

3. The Kutta method, where:
$y_n = y_{n-1} + \frac{1}{6}\Delta t(f_1 + 4f_2 + f_3)$
$f_1 = f(t_{n-1}, y_{n-1})$
$f_2 = f(t_{n-1} + \frac{1}{2}\Delta t, y_{n-1} + \frac{1}{2}\Delta t f_1)$
$f_3 = f(t_{n-1} + \Delta t, y_{n-1} - \Delta t f_1 + 2\Delta t f_2)$

4. The 4th order Runge-Kutta method ("classical" Runge-Kutta method), where:
$y_n = y_{n-1} + \frac{1}{6}\Delta t(f_1 + 2f_2 + 2f_3 + f_4)$
$f_1 = f(t_{n-1}, y_{n-1})$
$f_2 = f(t_{n-1} + \frac{1}{2}\Delta t, y_{n-1} + \frac{1}{2}\Delta t f_1)$
$f_3 = f(t_{n-1} + \frac{1}{2}\Delta t, y_{n-1} + \frac{1}{2}\Delta t f_2)$
$f_4 = f(t_{n-1} + \Delta t, y_{n-1} + \Delta t f_3)$

In order to use these methods however, the system defining the planar three-body problem must be in the form $\frac{d\mathbf{y}(x)}{dy} = \mathbf{f}(x, \mathbf{y}(x))$. To achieve this, we may simply define new variables $v_{x1}(t) = \frac{dx_1(t)}{dy}$, $v_{y1}(t) = \frac{dy_1(t)}{dy}$, $v_{x2}(t) = \frac{dx_2(t)}{dy}$, $v_{y2}(t) = \frac{dy_2(t)}{dy}$, $v_{x3}(t) = \frac{dx_3(t)}{dy}$, $v_{y3}(t) = \frac{dy_3(t)}{dy}$ such that the new system of IVPs is as follows:

$$\begin{cases} \frac{dx_1(t)}{dy} = v_{x1}(t) \\ \frac{dv_{x1}}{dt} = -Gm_2\frac{x_1(t)-x_2(t)}{r_{12}^3(t)} - Gm_3\frac{x_1(t)-x_3(t)}{r_{31}^3(t)} \\ \frac{dy_1(t)}{dy} = v_{y1}(t) \\ \frac{dv_{y1}(t)}{dt} = -Gm_2\frac{y_1(t)-y_2(t)}{r_{12}^3(t)} - Gm_3\frac{y_1(t)-y_3(t)}{r_{31}^3(t)} \\ \frac{dx_2(t)}{dy} = v_{x2}(t) \\ \frac{dv_{x2}(t)}{dt} = -Gm_3\frac{x_2(t)-x_3(t)}{r_{23}^3(t)} - Gm_1\frac{x_2(t)-x_1(t)}{r_{12}^3(t)} \\ \frac{dy_2(t)}{dy} = v_{y2}(t) \\ \frac{dv_{y2}(t)}{dt} = -Gm_3\frac{y_2(t)-y_3(t)}{r_{23}^3(t)} - Gm_1\frac{y_2(t)-y_1(t)}{r_{12}^3(t)} \\ \frac{dx_3(t)}{dy} = v_{x3}(t) \\ \frac{dv_{x3}(t)}{dt} = -Gm_1\frac{x_3(t)-x_1(t)}{r_{31}^3(t)} - Gm_2\frac{x_3(t)-x_2(t)}{r_{23}^3(t)} \\ \frac{dy_3(t)}{dy} = v_{y3}(t) \\ \frac{dv_{y3}(t)}{dt} = -Gm_1\frac{y_3(t)-y_1(t)}{r_{31}^3(t)} - Gm_2\frac{y_3(t)-y_2(t)}{r_{23}^3(t)} \end{cases} \tag{2}$$

Therefore, we have $\mathbf{f} = \begin{bmatrix} v_{x1}(t) \\ -Gm_2\frac{y_1(t)-y_2(t)}{r_{12}^3(t)} - Gm_3\frac{y_1(t)-y_3(t)}{r_{31}^3(t)} \\ v_{y1}(t) \\ -Gm_2\frac{y_1(t)-y_2(t)}{r_{12}^3(t)} - Gm_3\frac{y_1(t)-y_3(t)}{r_{31}^3(t)} \\ v_{x2}(t) \\ -Gm_3\frac{x_2(t)-x_3(t)}{r_{23}^3(t)} - Gm_1\frac{x_2(t)-x_1(t)}{r_{12}^3(t)} \\ v_{y2}(t) \\ -Gm_3\frac{y_2(t)-y_3(t)}{r_{23}^3(t)} - Gm_1\frac{y_2(t)-y_1(t)}{r_{12}^3(t)} \\ v_{x3}(t) \\ -Gm_1\frac{x_3(t)-x_1(t)}{r_{31}^3(t)} - Gm_2\frac{x_3(t)-x_2(t)}{r_{23}^3(t)} \\ v_{y3}(t) \\ -Gm_1\frac{y_3(t)-y_1(t)}{r_{31}^3(t)} - Gm_2\frac{y_3(t)-y_2(t)}{r_{23}^3(t)} \end{bmatrix}$

This experiment was performed entirely using the *MATLAB* environment. The figures used were generated using *MATLAB*'s extensive plotting capabilities.

3.2 Goals of the experiment

The experiment is split into 4 tasks, which are as follows:

1. Task1: Compare the results of the numerical solution to the planar three-body problem for all 4 methods previously mentioned. Assume that $G = m_1 = m_2 = m_3 = 1$ and $t \in [0, 5.226525]$. The vector of initial conditions is given as:

$$\mathbf{y_0} = \begin{bmatrix} x_1(0) \\ v_{x1}(0) \\ y_1(0) \\ v_{y1}(0) \\ x_2(0) \\ v_{x2}(0) \\ y_2(0) \\ v_{y2}(0) \\ x_3(0) \\ v_{x3}(0) \\ y_3(0) \\ v_{y3}(0) \end{bmatrix} = \begin{bmatrix} 0.8083106230 \\ 0.0000000000 \\ 0.0000000000 \\ 0.9901979166 \\ -0.4954148566 \\ 0.0000000000 \\ 0.0000000000 \\ -2.7171431768 \\ -0.3128957664 \\ 0.0000000000 \\ 0.0000000000 \\ 1.7269452602 \end{bmatrix}$$

2. Task 2: Find r_0 satisfying the following set of equations forming a particular solution of the planar three-body problem:

$$\begin{cases} x_1(t) = r_0 sin(t - \frac{\pi}{2}) \\ y_1(t) = r_0 cos(t - \frac{\pi}{2}) \\ x_2(t) = r_0 sin(t + \frac{\pi}{6}) \\ y_2(t) = r_0 cos(t + \frac{\pi}{6}) \\ x_3(t) = r_0 sin(t + \frac{5\pi}{6}) \\ y_3(t) = r_0 cos(t + \frac{5\pi}{6}) \end{cases} \tag{3}$$

   Assume $G = 1$ and $m_1 = m_2 = m_3 = \sqrt{3}$.

3. Task 3: calculate the mean-square error in the approximation using the solution obtained in task 2 for each of the 4 methods previously mentioned, according to the following formula: $e_y = \frac{1}{N}\Sigma_{n=1}^{N}[y_n - y(t_n)]^2$. Determine the dependence of this error on Δt for each method.

4. Task 4: Using the measured data of the coordinates provided in the file "data_13.csv", use the methods previously mentioned to approximate the masses of the three bodies. Assume that the trajectories of the bodies satisfy the planar three-body problem and that $G = 1$.

3.3 Results of experimentation

3.3.1 Results obtained for task 1

Figures 1 - 7 illustrate the obtained trajectories of the 3 bodies:

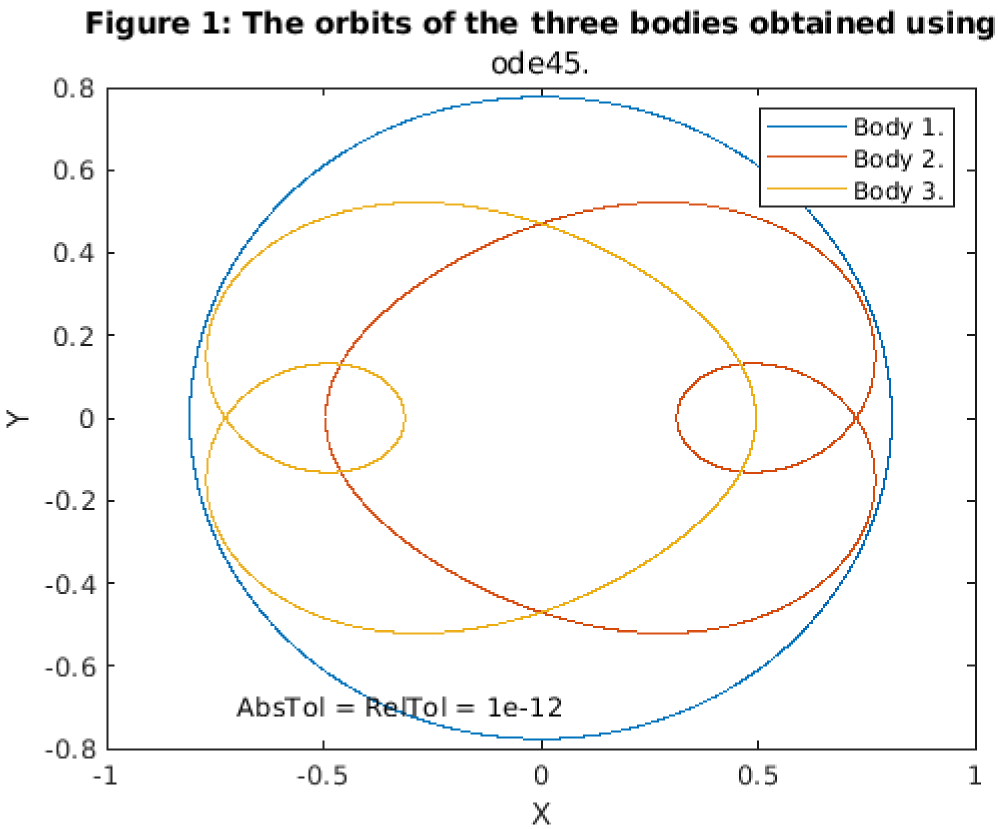

Figure 1: Plot of the obtained trajectories approximated using ode45.

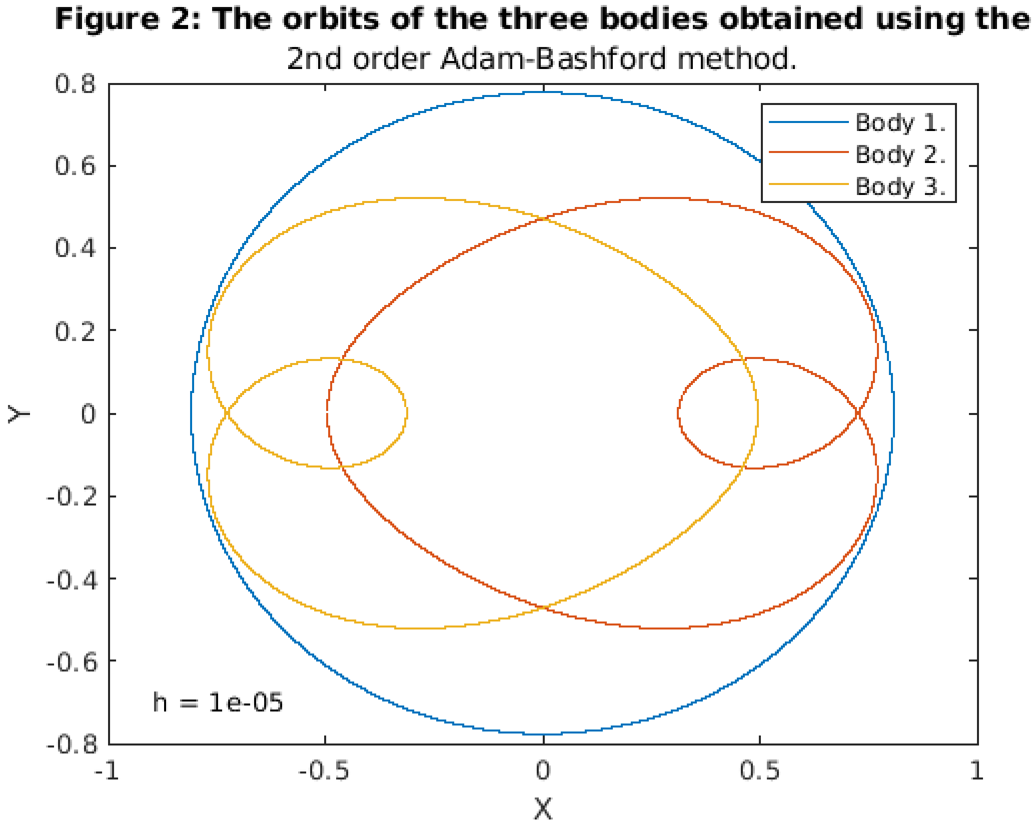

Figure 2: Plot of the obtained trajectories approximated using the 2nd order Adam-Bashford method.

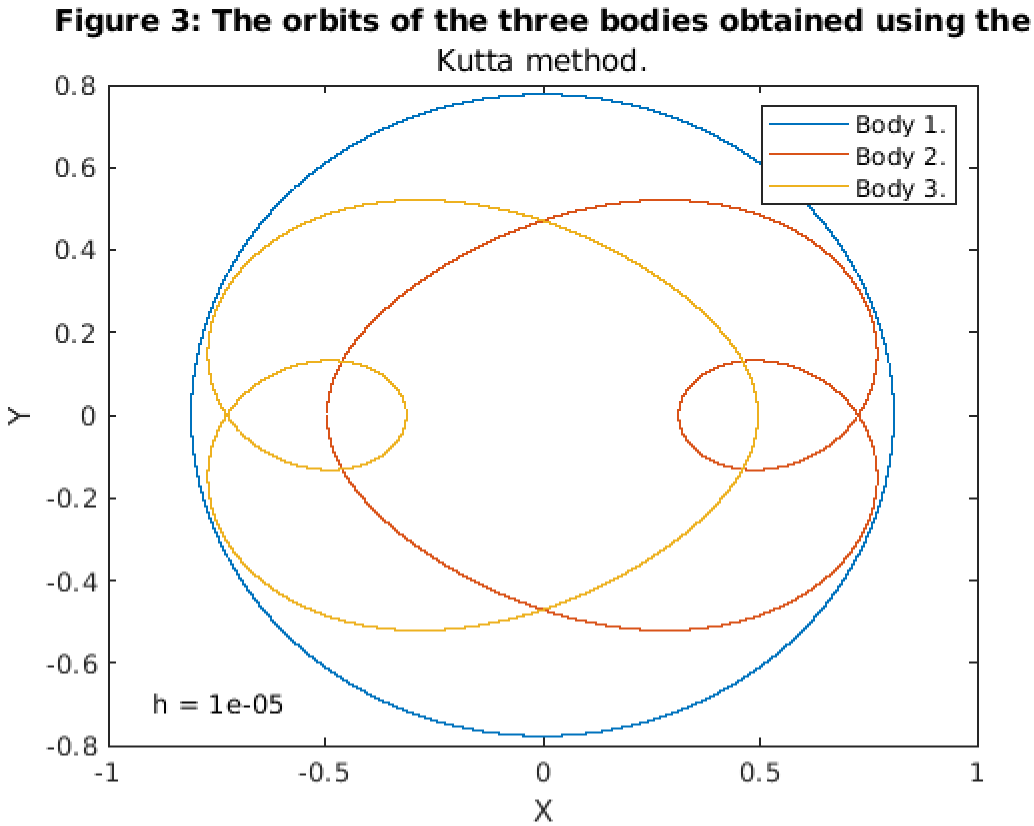

Figure 3: Plot of the obtained trajectories approximated using the Kutta method.

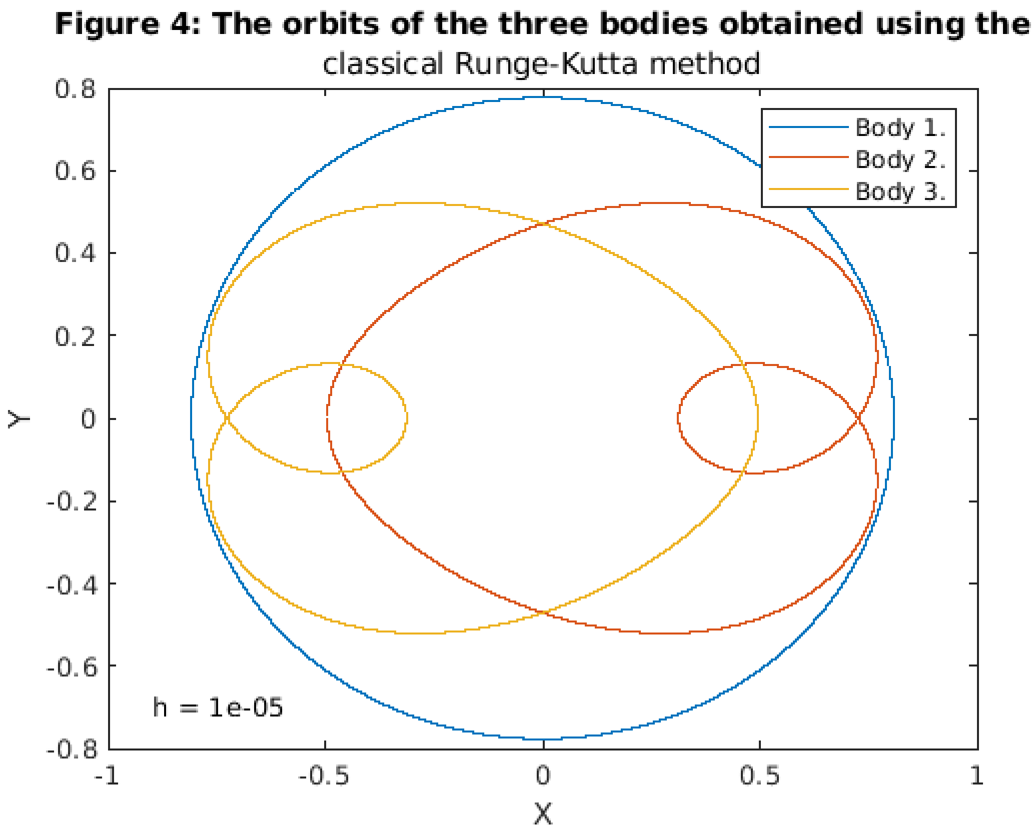

Figure 4: Plot of the obtained trajectories approximated using the classical Runge-Kutta method.

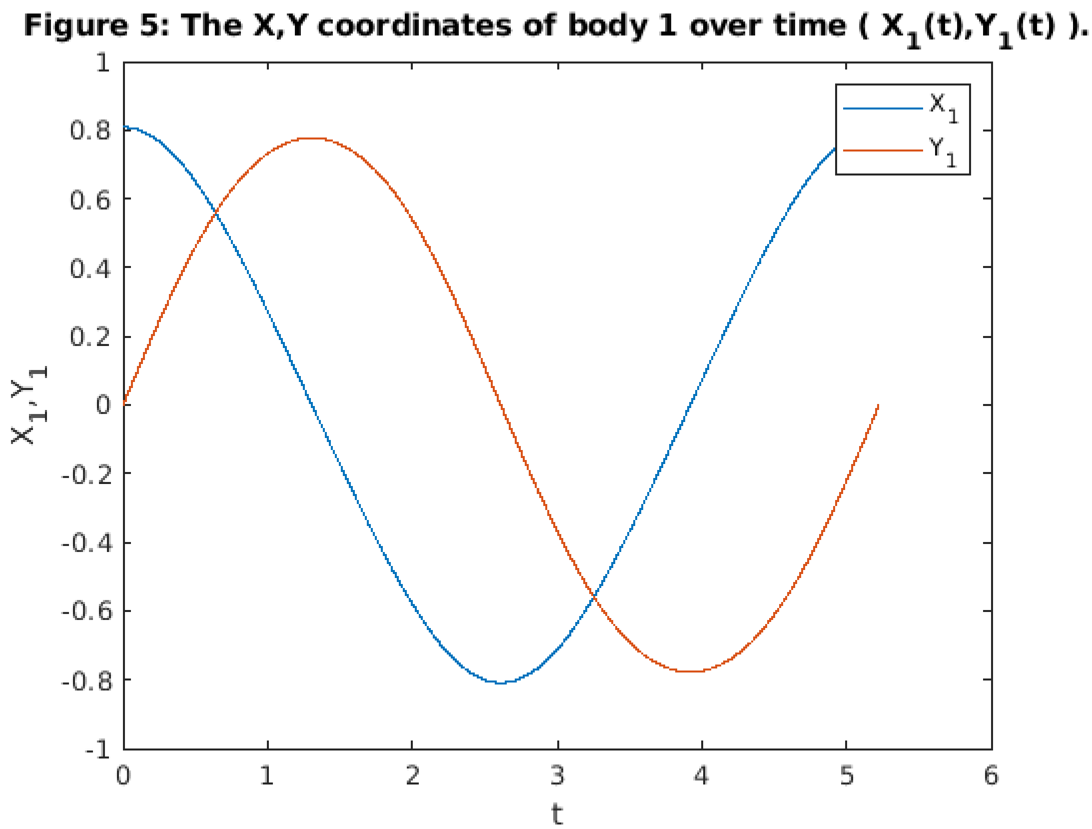

Figure 5: Plot of the obtained coordinates for the 1st body over time.

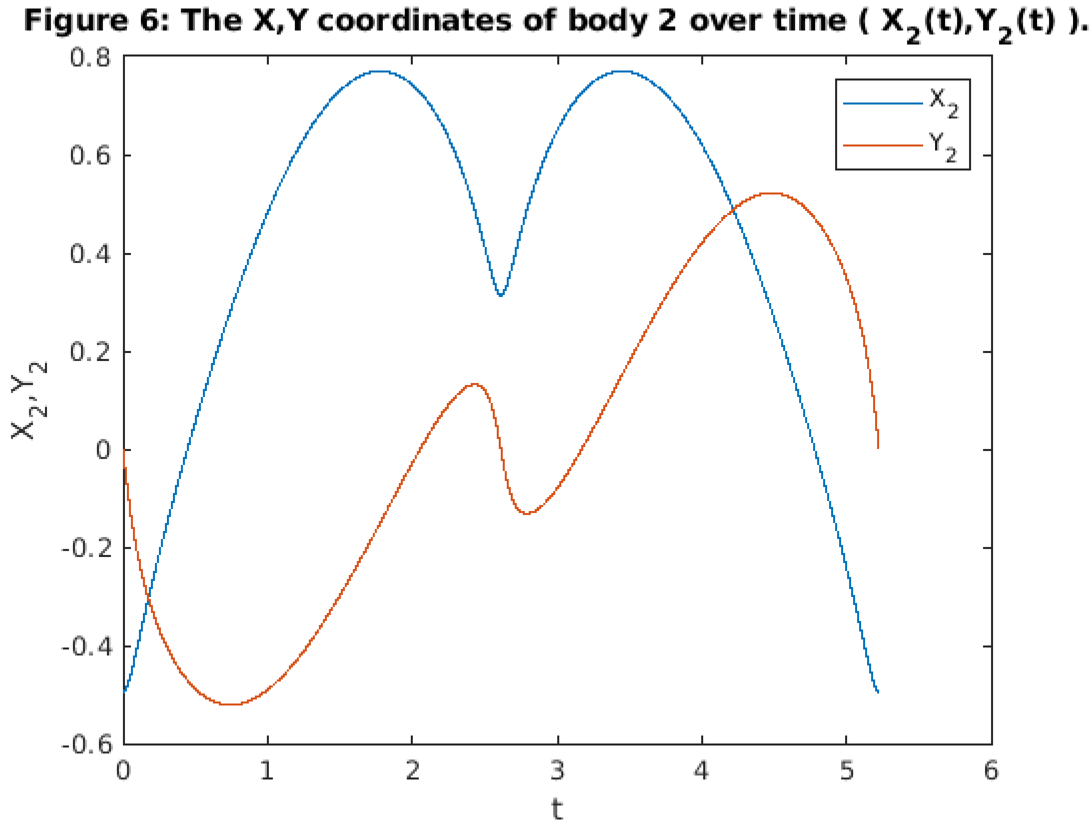

Figure 6: Plot of the obtained coordinates for the 2nd body over time..

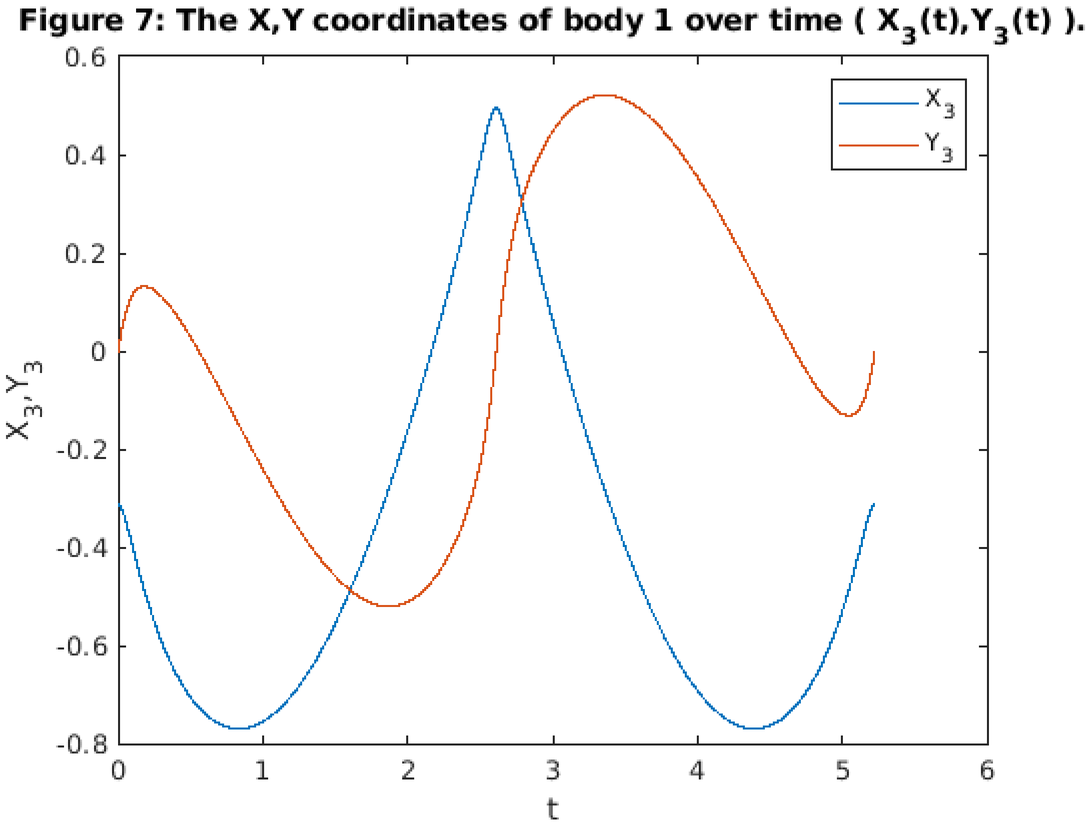

Figure 7: Plot of the obtained coordinates for the 3rd body over time.

3.3.2 results obtained for task 2

To find r_0, we can use the first equation in the planar three-body problem, setting $t = 0$ and plugging in the particular solution and the parameters from the system in equation (3):

$$\frac{d^2x_1(0)}{dt^2} = -Gm_2\frac{x_1(0) - x_2(0)}{r_{12}^3(0)} - Gm_3\frac{x_1(0) - x_3(0)}{r_{31}^3(0)}$$

$$x_1(t) = r_0 sin(t - \frac{\pi}{2}) \Rightarrow \frac{dx_1(t)}{dt} = r_0 cos(t - \frac{\pi}{2}) \Rightarrow \frac{d^2x_1(t)}{dt^2} = -r_0 sin(t - \frac{\pi}{2})$$

$$x_1(0) = -r_0, \quad x_2(0) = \frac{1}{2}r_0, \quad x_3(0) = \frac{1}{2}r_0, \quad \frac{d^2x_1(0)}{dt^2} = r_0,$$

$$y_1(0) = 0, \quad y_2(0) = \frac{\sqrt{3}}{2}r_0, \quad y_3(0) = -\frac{\sqrt{3}}{2}r_0$$

$$r_{jk}(t) = \sqrt{[x_k(t) - x_j(t)]^2 + [y_k(t) - y_j(t)]^2} \text{ for } j, k = 1, 2, 3 \Rightarrow$$

$$r_{12}^3(0) = [(\frac{1}{2}r_0 - (-r_0))^2 + (\frac{\sqrt{3}}{2}r_0 - 0)^2]^{\frac{3}{2}} = [\frac{9}{4}r_0^2 + \frac{3}{4}r_0^2]^{\frac{3}{2}} = 3^{\frac{3}{2}}r_0^3$$

$$r_{31}^3(0) = [(-r_0 - \frac{1}{2}r_0)^2 + (0 - (-\frac{\sqrt{3}}{2}r_0)^2)]^{\frac{3}{2}} = [\frac{9}{4}r_0^2 + \frac{3}{4}r_0^2]^{\frac{3}{2}} = 3^{\frac{3}{2}}r_0^3$$

Finally, we have:

$$r_0 = -\sqrt{3}\frac{-r_0 - \frac{1}{2}r_0}{3^{\frac{3}{2}}r_0^3} - \sqrt{3}\frac{-r_0 - \frac{1}{2}r_0}{3^{\frac{3}{2}}r_0^3} \Rightarrow r_0 = \sqrt{3}\frac{r_0 + \frac{1}{2}r_0}{3^{\frac{3}{2}}r_0^3} + \sqrt{3}\frac{r_0 + \frac{1}{2}r_0}{3^{\frac{3}{2}}r_0^3} \Rightarrow$$

$$r_0 = 2\sqrt{3}\frac{\frac{3}{2}r_0}{3^{\frac{3}{2}}r_0^3} \Rightarrow r_0 = \frac{3\sqrt{3}}{3^{\frac{3}{2}}r_0^2} \Rightarrow r_0 = \frac{1}{r_0^2} \Rightarrow r_0^3 = 1 \Rightarrow r_0 = 1 \quad (4)$$

3.3.3 Obtained results for task 3

Figures 8 - 11 show the obtained results for the dependence of the mean-square error on the time-step:

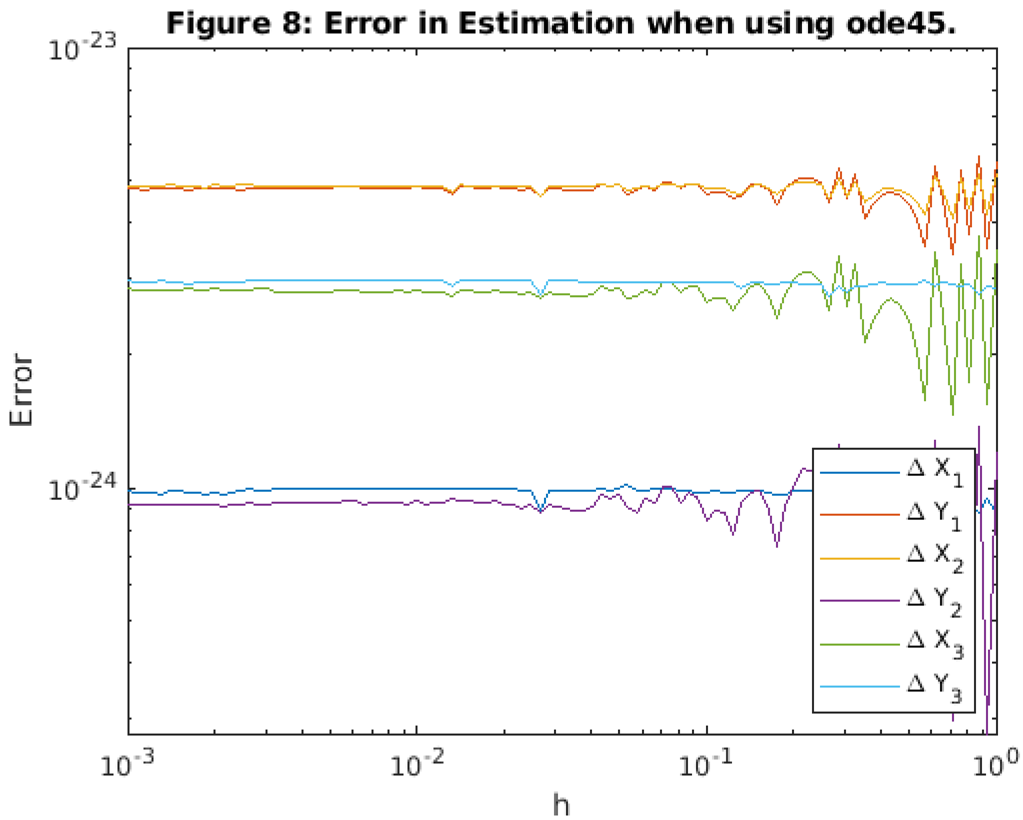

Figure 8: Obtained dependence of the mean-square error on the time-step in the results of ode45.

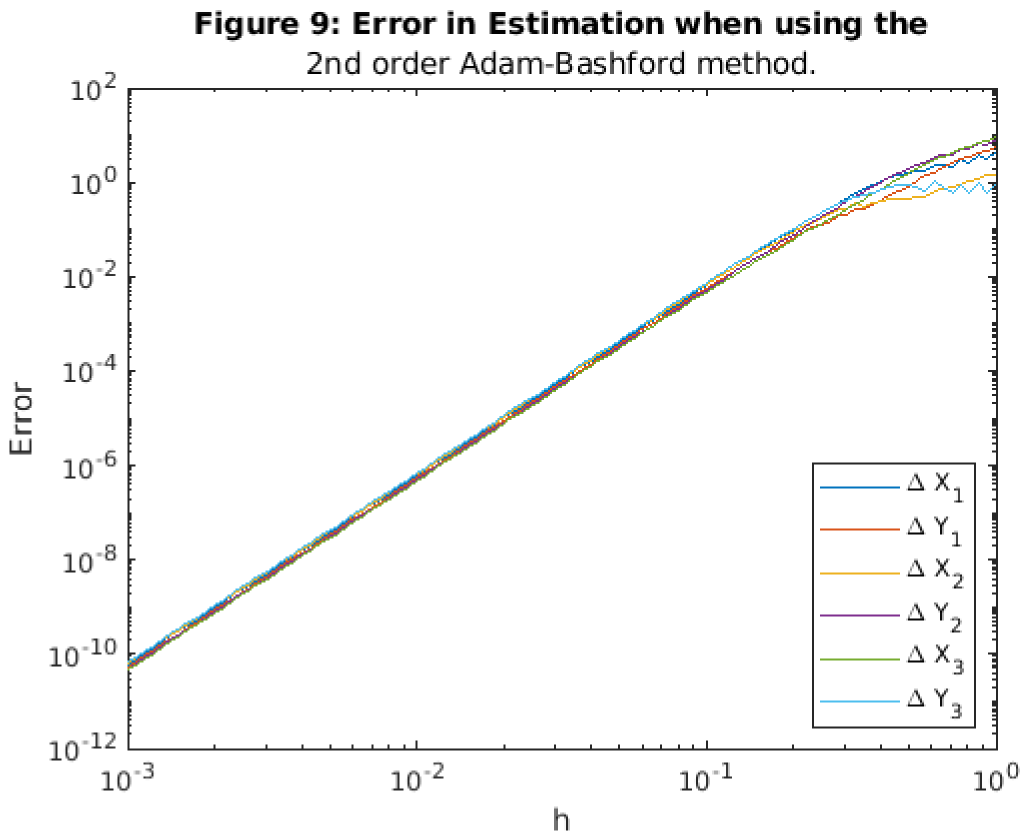

Figure 9: Obtained dependence of the mean-square error on the time-step in the results of the 2nd order Adam-Bashford method.

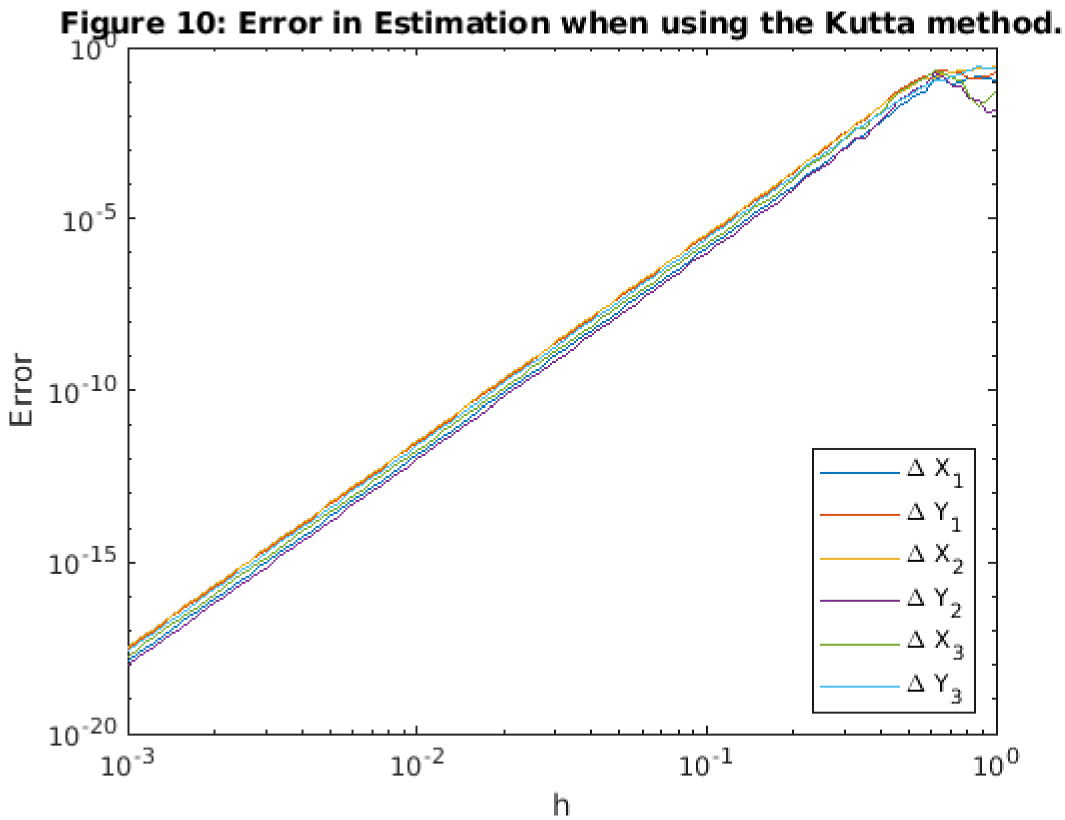

Figure 10: Obtained dependence of the mean-square error on the time-step in the results of the Kutta method.

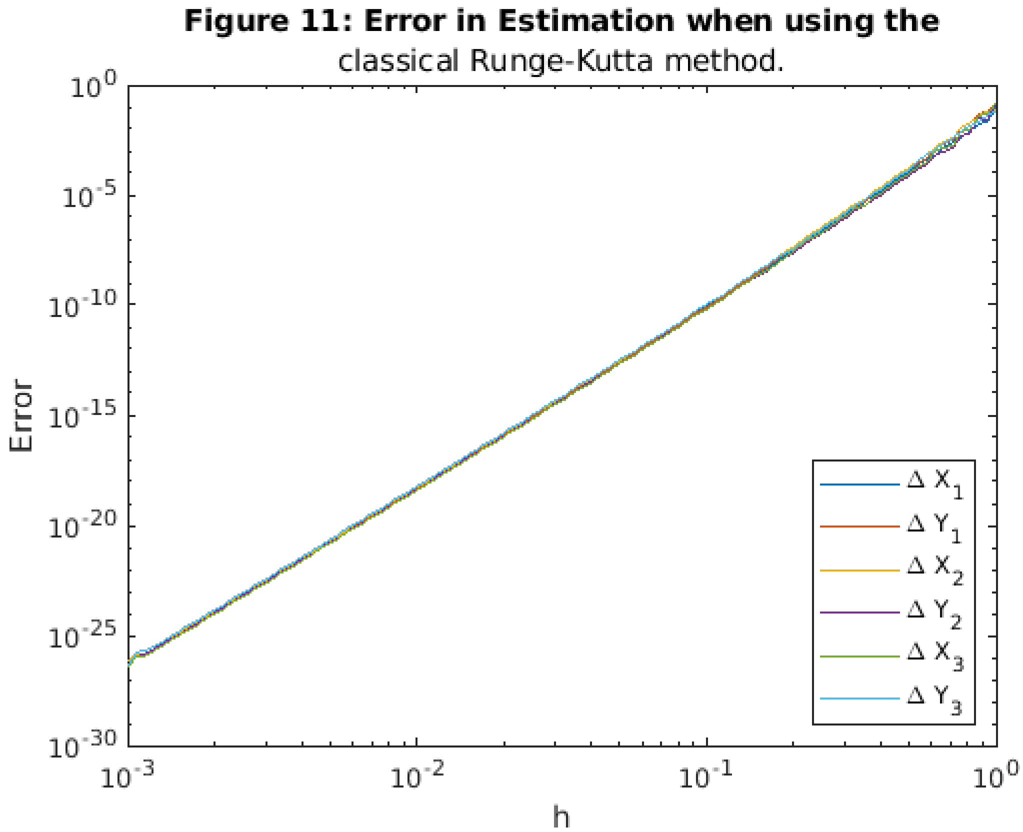

Figure 11: Obtained dependence of the mean-square error on the time-step in the results of the 4th order Runge-Kutta method.

3.3.4 Obtained results for task 4

Figures 12, 13, 14 show the obtained results for the approximated trajectories. The trajectories are calculated with the initial guess of $m_1 = m_2 = m_3 = 1$, and the data in "$data\_13.csv$", using ode45. The parameters are then optimized by finding the set of parameters which minize the following criterion:

$$J(\mathbf{m}) = \Sigma_{n=1}^{N} \sqrt{[\hat{x_1}\ (t_n; \mathbf{m}) - \widetilde{x}_{1,n}]^2 + [\hat{y_1}\ (t_n; \mathbf{m}) - \widetilde{y}_{1,n}]^2} +$$
$$\Sigma_{n=1}^{N} \sqrt{[\hat{x_2}\ (t_n; \mathbf{m}) - \widetilde{x}_{2,n}]^2 + [\hat{y_2}\ (t_n; \mathbf{m}) - \widetilde{y}_{2,n}]^2} +$$
$$\Sigma_{n=1}^{N} \sqrt{[\hat{x_3}\ (t_n; \mathbf{m}) - \widetilde{x}_{3,n}]^2 + [\hat{y_3}\ (t_n; \mathbf{m}) - \widetilde{y}_{3,n}]^2}$$

Where:

$$\mathbf{m} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix},$$

$\hat{x_i}\ (t_n; \mathbf{m})$ is the numerical solution approximating the x coordinates of the i-th body at t_n.

$\hat{y_i}\ (t_n; \mathbf{m})$ is the numerical solution approximating the y coordinates of the i-th body at t_n.

$\widetilde{x}_{i,n}$ is the measured x coordinate of the i-th body at t_n.

$\widetilde{y}_{i,n}$ is the measured y coordinate of the i-th body at t_n.

The obtained optimized parameters are:
$m_1 = 1.1648$,
$m_2 = 1.1409$,
$m_3 = 1.0826$.

4 Discussion

4.0.1 Obtained results for task 1

- looking at figures 1, 2, 3, 4, it can be seen that all methods return roughly equivalent results. An interesting feature of the obtained results is the fact that the bodies all occupy periodic orbits, which is not guranteed in choatic systems such as the three-body problem. Furthermore, due to the fact that the obtained orbits are quite smooth and predictable, the uniformity of the

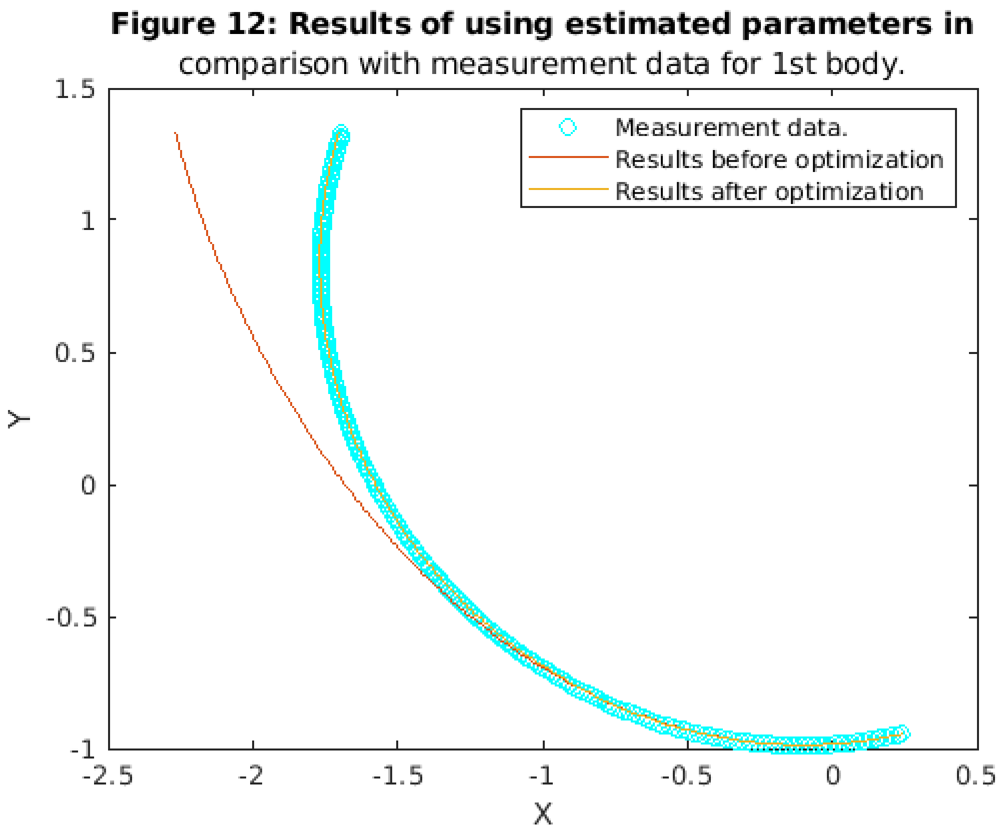

Figure 12: Obtained dependence of the mean-square error on the time-step in the results of the 2nd order Adam-Bashford method.

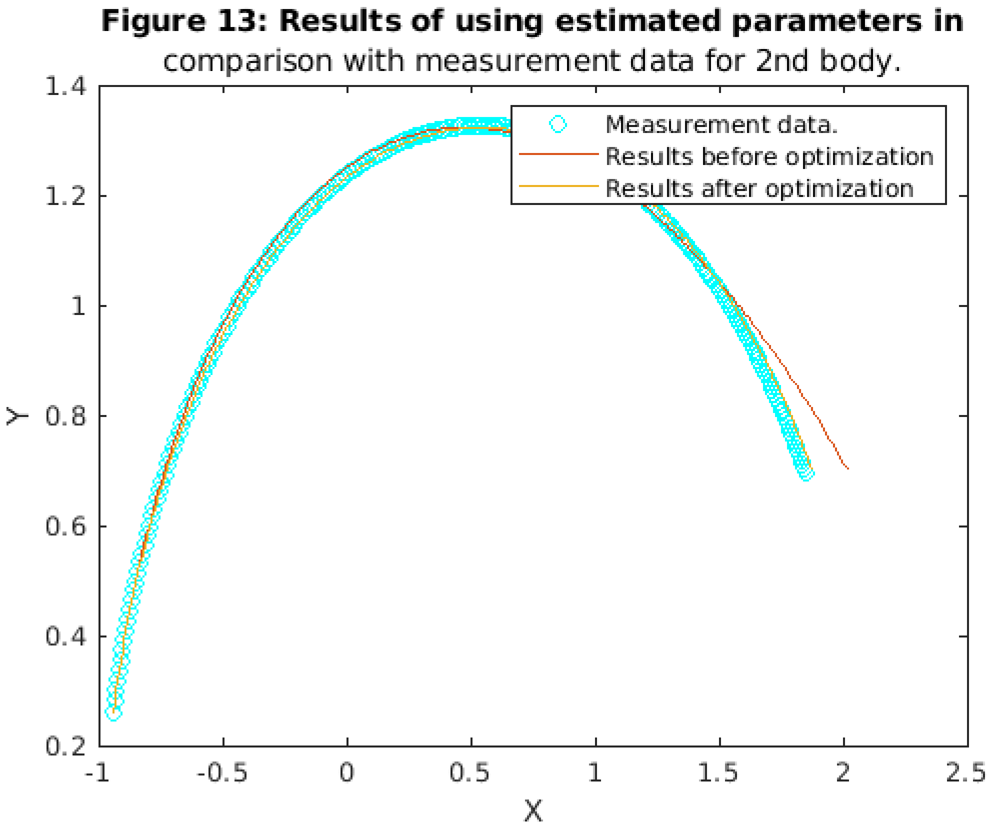

Figure 13: Obtained dependence of the mean-square error on the time-step in the results of the Kutta method.

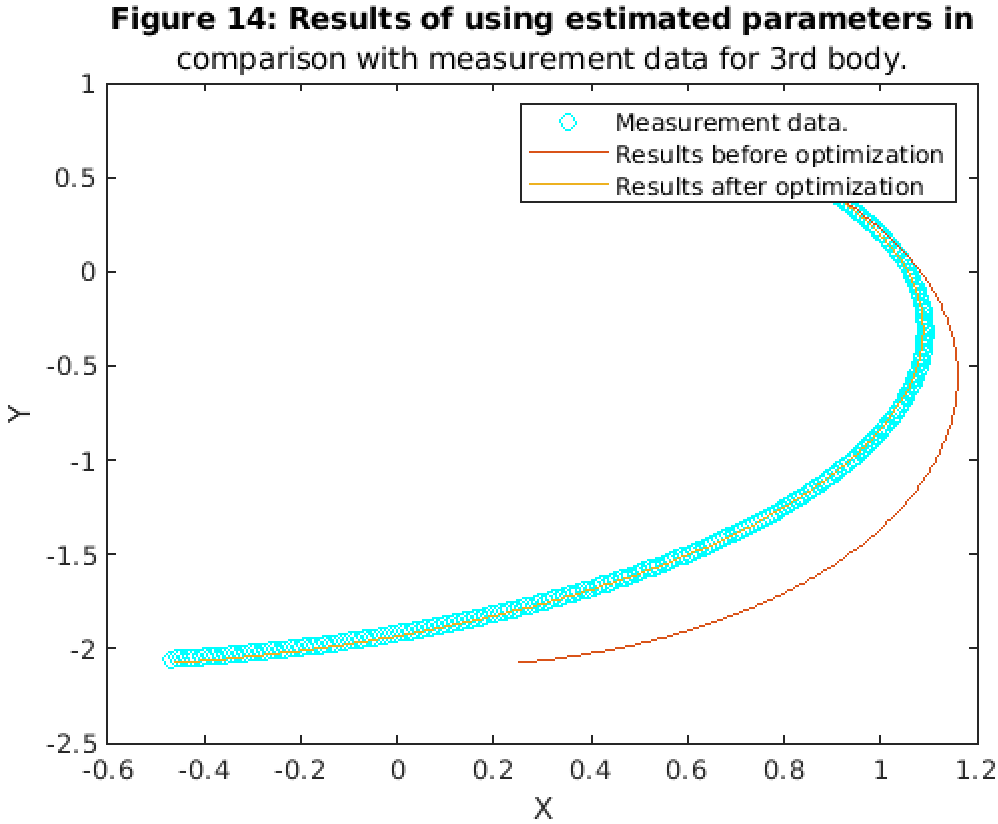

Figure 14: Obtained dependence of the mean-square error on the time-step in the results of the 4th order Runge-Kutta method.

obtained results is not surprising . This explains why the obtained results when using very accurate and stable numerical methods **[1]** such as ode45 return very similar results to methods which are much less accurate and stable such as the 2nd order Adam-Bashford method.

4.0.2 Obtained results for task 2

- Similarly to the first task, it can be seen from the particular solution that the new initial conditions still provide periodic orbits.

4.0.3 Obtained results for task 3

- Looking at Figures 9, 10, 11, the important conclusion can be drawn that the mean-square error in the estimation when using each of the provided methods depends heavily on the time-step h, so much so that the dependence is almost linear. This, however, is not observed in figure 8. The reason for the unique behaviour in the dependence obtained when using ode45 is because of the way that ode45 decides the-time step when given an interval of time. Even when ode45 is given specific points in time to find the solution at, ode45 ignores the time-step between those points and decides its own time-step according to the adaptive-step algorithm, which decides a new time-step every iteration according to the size of the local error. ode45 then interpolates the obtained solutions at the specific points given to it **[2]**. This leads to the mean-square error not changing regardless of the time-step between the vector of time points given to it.

- Finally, reading the values from figures 8, 9, 10, 11, the following order can be given for the methods, ordered from the lowest to the largest mean-square error given the same values of the time-step h:

1. ode45.
2. 4th order Runge-kutta method.
3. Kutta method.
4. 2nd order Adam-bashford method.

4.0.4 Obtained results for task 4

- A comparison of the results before and after optimization clearly show that the optimization succeeds in providing much better estimates to the measured data. It also shows that numerical methods can be used to obtain reliable estimates of unknown parameters in problems which are difficult to solve analytically.

5 Conclusion

- Numerical methods are robust tools for obtaining solutions to systems that are difficult to solve analytically.
- In terms of accuracy and stability, the best of the tested methods is ode45, and the 2nd order Adam-Bashford method is the worst. The other two methods fall somewhere in between.

6 References

[1] R. Z. Morawski, lecture notes for the course *Numerical Methods*, Warsaw University of Technology, Faculty of Electronics and Information Technology, spring semester 2023/24.

[2] `https://www.mathworks.com/help/matlab/ref/ode45.html?s_tid=doc_ta` [accessed in 11/06/2024]

7 Appendix

7.1 *MATLAB* implementation of assignment C

```
% Beggining of script, clearing all variables from previous script runs.
clearvars

% Task 1 ****************************************

% Defining the vector describing the position, as well as the velocities for each
% body. it is initialized with the initial conditions. The vector is of the
% following form: z = [x1, y1, vx1, vy1, x2, y2, vx2, vy2, x3, y3, vx3, vy3]
%                      1   2    3    4   5   6    7    8   9   10   11   12
```

```
z = [ 0.8083106230 ; 0 ; 0 ; 0.9901979166 ; -0.4954148566 ; 0 ; 0 ; -2.7171431768 ; -0.3128957664 ; 0 ; 0 ; 1.7269452602 ];

% Defining the time span and time step. The time span is chosen according to
% the assignment description.
T = 5.226525;
p = -5;
dt = 10^p;

% Defining known constants used in the set of equations.
G = 1;
m1 = 1;
m2 = 1;
m3 = 1;

% Defining the function describing the distances between the bodies.
r12 = @(Z) sqrt((Z(1) - Z(5))^2 + (Z(2) - Z(6))^2);
r23 = @(Z) sqrt((Z(5) - Z(9))^2 + (Z(6) - Z(10))^2);
r13 = @(Z) sqrt((Z(1) - Z(9))^2 + (Z(2) - Z(10))^2);

% Defining the set of ODEs to be solved
f = @(t,Z) [
 Z(3);
 Z(4);
 -G*m2*(Z(1) - Z(5))/r12(Z)^3 - G*m3*(Z(1) - Z(9))/r13(Z)^3;
 -G*m2*(Z(2) - Z(6))/r12(Z)^3 - G*m3*(Z(2) - Z(10))/r13(Z)^3;
 Z(7);
 Z(8);
 -G*m3*(Z(5) - Z(9))/r23(Z)^3 - G*m1*(Z(5) - Z(1))/r12(Z)^3;
 -G*m3*(Z(6) - Z(10))/r23(Z)^3 - G*m1*(Z(6) - Z(2))/r12(Z)^3;
 Z(11);
 Z(12);
```

```
 -G*m1*(Z(9) - Z(1))/r13(Z)^3 - G*m2*(Z(9) - Z(5))/r23(Z)^3;
 -G*m1*(Z(10) - Z(2))/r13(Z)^3 - G*m2*(Z(10) - Z(6))/r23(Z)^3;
];

% Obtaining the matrix of numerical values representing the solution of the
% set of ODEs in the time interval [0,T]. It is obtained using ode45. The absolute and relative tolerance is
% set to 10^-12 using odeset.
options = odeset('RelTol',1e-12,'AbsTol',1e-12);
[ts,R0] = ode45(f,[0 T],z,options);

% Obtaining the matrix of numerical values representing the solution of the
% set of ODEs in the time interval [0,T]. It is obtained using the second
% order Adam-Bashford method.
z_AB = AB2(f,dt,T,z);

% Obtaining the matrix of numerical values representing the solution of the
% set of ODEs in the time interval [0,T]. It is obtained using the Kutta method.
z_K = Kut(f,dt,T,z);

% Obtaining the matrix of numerical values representing the solution of the set of
% ODEs in the time interval [0,T]. It is obtained using the classical Runge-Kutta method.
z_RK = RK4(f,dt,T,z);

% Code for generation of figure 1,2,3,4,5,6,7.
txt = "h = "+dt;

comet(R0(:,1),R0(:,2))
hold on;
comet(R0(:,5),R0(:,6))
comet(R0(:,9),R0(:,10))
```

```
title("Animated orbits of the three bodies in order
   ", "obtained using ode45.")
xlabel("X")
ylabel("Y")
text(-0.7,-0.7,"AbsTol = RelTol = 1e-12");
hold off;

figure(1)
clf
plot(R0(:,1),R0(:,2))
hold on;
plot(R0(:,5),R0(:,6))
plot(R0(:,9),R0(:,10))
legend("Body 1.","Body 2.","Body 3.");
title("Figure 1: The orbits of the three bodies
   obtained using", "ode45.")
xlabel("X")
ylabel("Y")
text(-0.7,-0.7,"AbsTol = RelTol = 1e-12");
hold off;

figure(2)
clf
plot(z_AB(1,:),z_AB(2,:))
hold on;
plot(z_AB(5,:),z_AB(6,:))
plot(z_AB(9,:),z_AB(10,:))
legend("Body 1.","Body 2.","Body 3.");
title("Figure 2: The orbits of the three bodies
   obtained using the", "2nd order Adam-Bashford
   method.")
xlabel("X")
ylabel("Y")
text(-0.9,-0.7,txt);
hold off;

figure(3)
clf
plot(z_K(1,:),z_K(2,:))
hold on;
plot(z_K(5,:),z_K(6,:))
```

```
plot(z_K(9,:),z_K(10,:))
legend("Body 1.","Body 2.","Body 3.");
title("Figure 3: The orbits of the three bodies
   obtained using the", "Kutta method.")
xlabel("X")
ylabel("Y")
text(-0.9,-0.7,txt);
hold off;

figure(4)
clf
plot(z_RK(1,:),z_RK(2,:))
hold on;
plot(z_RK(5,:),z_RK(6,:))
plot(z_RK(9,:),z_RK(10,:))
legend("Body 1.","Body 2.","Body 3.");
title("Figure 4: The orbits of the three bodies
   obtained using the", "classical Runge-Kutta
   method")
xlabel("X")
ylabel("Y")
text(-0.9,-0.7,txt);
hold off;

figure(5)
clf
plot(ts,R0(:,1));
hold on;
plot(ts,R0(:,2));
title("Figure 5: The X,Y coordinates of body 1 over
   time ( X_{1}(t),Y_{1}(t) ).");
legend("X_{1}","Y_{1}")
xlabel("t")
ylabel("X_{1},Y_{1}")
hold off;

figure(6)
clf
plot(ts,R0(:,5));
hold on;
plot(ts,R0(:,6));
```

```
legend("X_{2}","Y_{2}")
title("Figure 6: The X,Y coordinates of body 2 over
   time ( X_{2}(t),Y_{2}(t) ).");
xlabel("t")
ylabel("X_{2},Y_{2}")
hold off;

figure(7)
clf
plot(ts,R0(:,9));
hold on;
plot(ts,R0(:,10));
legend("X_{3}","Y_{3}")
title("Figure 7: The X,Y coordinates of body 1 over
   time ( X_{3}(t),Y_{3}(t) ).");
xlabel("t")
ylabel("X_{3},Y_{3}")
hold off;

% Task 3 ****************************************
% Setting value of constant r0. It is set to the
   value obtained
% analytically in task 2 using the particular
   analytical solution.
r0 = -1;

% Defining the time span and time step. The time
   span is chosen according to
% the assignment description.
T = 2*pi;
N = 100;
dt_values = logspace(-3,0,N);

% Setting the values of the constant G and the
   masses.
G = 1;
m1 = sqrt(3);
m2 = sqrt(3);
m3 = sqrt(3);
```

```
% Refefining f to update the values of the constant
   G and the masses.
f = @(t,Z) [
 Z(3);
 Z(4);
 -G*m2*(Z(1) - Z(5))/r12(Z)^3 - G*m3*(Z(1) - Z(9))/
    r13(Z)^3;
 -G*m2*(Z(2) - Z(6))/r12(Z)^3 - G*m3*(Z(2) - Z(10))/
    r13(Z)^3;
 Z(7);
 Z(8);
 -G*m3*(Z(5) - Z(9))/r23(Z)^3 - G*m1*(Z(5) - Z(1))/
    r12(Z)^3;
 -G*m3*(Z(6) - Z(10))/r23(Z)^3 - G*m1*(Z(6) - Z(2))/
    r12(Z)^3;
 Z(11);
 Z(12);
 -G*m1*(Z(9) - Z(1))/r13(Z)^3 - G*m2*(Z(9) - Z(5))/
    r23(Z)^3;
 -G*m1*(Z(10) - Z(2))/r13(Z)^3 - G*m2*(Z(10) - Z(6))
    /r23(Z)^3;
];

% Defining the new initial conditions.
z = [
r0*sin(-pi/2) ;
r0*cos(-pi/2) ;
r0*cos(-pi/2) ;
-r0*sin(-pi/2) ;
r0*sin(pi/6) ;
r0*cos(pi/6) ;
r0*cos(pi/6) ;
-r0*sin(pi/6) ;
r0*sin(pi*5/6) ;
r0*cos(pi*5/6) ;
r0*cos(pi*5/6) ;
-r0*sin(pi*5/6) ;
];

% Defining the function represeting the particular
   solution to the set of
```

```
% ODEs.
PS = @(t) [
   r0*sin(t-pi/2) ;
   r0*cos(t-pi/2) ;
   r0*sin(t+pi/6) ;
   r0*cos(t+pi/6) ;
   r0*sin(t+(5*pi)/6) ;
   r0*cos(t+(5*pi)/6) ;
];

%initializing variables used for the storing of
   error
timespan = 0:dt_values(1):T;
M = size(timespan,2);

ms_OD = zeros(6,N);
ms_AB = zeros(6,N);
ms_K = zeros(6,N);
ms_RK = zeros(6,N);

% Error when using ode45 to obtain a numerical
   estimate of the trajectories.
for i1 = 1:N
    timespan = 0:dt_values(i1):T;
    M = size(timespan,2);

    r = PS(timespan);
    ErrOD = zeros(6,M);

    [ts,R] = ode45(f,timespan,z,options);
    EstOD = R';
    ErrOD(1,:) = EstOD(1,:) - r(1,:);
    ErrOD(2,:) = EstOD(2,:) - r(2,:);
    ErrOD(3,:) = EstOD(5,:) - r(3,:);
    ErrOD(4,:) = EstOD(6,:) - r(4,:);
    ErrOD(5,:) = EstOD(9,:) - r(5,:);
    ErrOD(6,:) = EstOD(10,:) - r(6,:);

    ErrOD = ErrOD.^2;
    for i2 = 1:6
        ms_OD(i2,i1) = (1/M)*sum(ErrOD(i2,:));
```

```
    end
end

% Error when using the 2nd order Adam-Bashford to
  obtain a numerical estimate of the trajectories.
for j1 = 1:N
    timespan = 0:dt_values(j1):T;
    M = size(timespan,2);

    r = PS(timespan);
    ErrAB = zeros(6,M);

    EstAB = AB2(f,dt_values(j1),T,z);
    ErrAB(1,:) = EstAB(1,:) - r(1,:);
    ErrAB(2,:) = EstAB(2,:) - r(2,:);
    ErrAB(3,:) = EstAB(5,:) - r(3,:);
    ErrAB(4,:) = EstAB(6,:) - r(4,:);
    ErrAB(5,:) = EstAB(9,:) - r(5,:);
    ErrAB(6,:) = EstAB(10,:) - r(6,:);

    ErrAB = ErrAB.^2;
    for j2 = 1:6
        ms_AB(j2,j1) = (1/M)*sum(ErrAB(j2,:));
    end
end

% Error when using the kutta method to obtain a
  numerical estimate of the trajectories.
for l1 = 1:N
    timespan = 0:dt_values(l1):T;
    M = size(timespan,2);

    r = PS(timespan);
    ErrK = zeros(6,M);

    EstK = Kut(f,dt_values(l1),T,z);
    ErrK(1,:) = EstK(1,:) - r(1,:);
    ErrK(2,:) = EstK(2,:) - r(2,:);
    ErrK(3,:) = EstK(5,:) - r(3,:);
    ErrK(4,:) = EstK(6,:) - r(4,:);
    ErrK(5,:) = EstK(9,:) - r(5,:);
```

```
    ErrK(6,:) = EstK(10,:) - r(6,:);

    ErrK = ErrK.^2;
    for l2 = 1:6
        ms_K(l2,l1) = (1/M)*sum(ErrK(l2,:));
    end
end

% Error when using the Runge-Kutta method to obtain
   a numerical estimate of the trajectories.
for k1 = 1:N
    timespan = 0:dt_values(k1):T;
    M = size(timespan,2);

    r = PS(timespan);
    ErrRK = zeros(6,M);

    EstRK = RK4(f,dt_values(k1),T,z);
    ErrRK(1,:) = EstRK(1,:) - r(1,:);
    ErrRK(2,:) = EstRK(2,:) - r(2,:);
    ErrRK(3,:) = EstRK(5,:) - r(3,:);
    ErrRK(4,:) = EstRK(6,:) - r(4,:);
    ErrRK(5,:) = EstRK(9,:) - r(5,:);
    ErrRK(6,:) = EstRK(10,:) - r(6,:);

    ErrRK = ErrRK.^2;
    for k2 = 1:6
        ms_RK(k2,k1) = (1/M)*sum(ErrRK(k2,:));
    end
end

% Code for generation of figure 8,9,10,11.
figure(8)
clf
loglog(dt_values,ms_OD(1,:));
hold on
for i = 2:6
    loglog(dt_values,ms_OD(i,:));
end
title("Figure 8: Error in Estimation when using
   ode45.");
```

```
legend("\Delta X_{1}","\Delta Y_{1}","\Delta X_{2}","\Delta Y_{2}","\Delta X_{3}","\Delta Y_{3}",'Location','southeast');
xlabel("h")
ylabel("Error")
hold off;

figure(9)
clf
loglog(dt_values,ms_AB(1,:));
hold on
for i = 2:6
    loglog(dt_values,ms_AB(i,:));
end
title("Figure 9: Error in Estimation when using the ", "2nd order Adam-Bashford method.");
legend("\Delta X_{1}","\Delta Y_{1}","\Delta X_{2}","\Delta Y_{2}","\Delta X_{3}","\Delta Y_{3}",'Location','southeast');
xlabel("h")
ylabel("Error")
hold off;

figure(10)
clf
loglog(dt_values,ms_K(1,:));
hold on
for i = 2:6
    loglog(dt_values,ms_K(i,:));
end
title("Figure 10: Error in Estimation when using the Kutta method.");
legend("\Delta X_{1}","\Delta Y_{1}","\Delta X_{2}","\Delta Y_{2}","\Delta X_{3}","\Delta Y_{3}",'Location','southeast');
xlabel("h")
ylabel("Error")
hold off;

figure(11)
clf
```

```
loglog(dt_values,ms_RK(1,:));
hold on
for i = 2:6
    loglog(dt_values,ms_RK(i,:));
end
title("Figure 11: Error in Estimation when using the ", "classical Runge-Kutta method.");
legend("\Delta X_{1}","\Delta Y_{1}","\Delta X_{2}","\Delta Y_{2}","\Delta X_{3}","\Delta Y_{3}",'Location','southeast');
xlabel("h")
ylabel("Error")
hold off;

% Task 4 ****************************************
% Reading the table of data13, and from it the time span used in the measurements.

j = @J;

p = fminsearch(j,[1,1,1]);

data13 = readtable("Documents/MATLAB/enume-2024-mahmoud-elshekh-ali-323930/Assignment C/data_13.csv");
data13 = data13{:,:};
timespan = 0:0.02:5;

G = 1;
m1 = p(1);
m2 = p(2);
m3 = p(3);

z = [
    data13(1,2);
    data13(1,3);
    (data13(2,2) - data13(1,2))/0.02;
    (data13(2,3) - data13(1,3))/0.02;
    data13(1,4);
    data13(1,5);
    (data13(2,4) - data13(1,4))/0.02;
```

```
    (data13(2,5) - data13(1,5))/0.02;
    data13(1,6);
    data13(1,7);
    (data13(2,6) - data13(1,6))/0.02;
    (data13(2,7) - data13(1,7))/0.02;
    ];

f = @(t,Z) [
 Z(3);
 Z(4);
 -G*m2*(Z(1) - Z(5))/r12(Z)^3 - G*m3*(Z(1) - Z(9))/r13(Z)^3;
 -G*m2*(Z(2) - Z(6))/r12(Z)^3 - G*m3*(Z(2) - Z(10))/r13(Z)^3;
 Z(7);
 Z(8);
 -G*m3*(Z(5) - Z(9))/r23(Z)^3 - G*m1*(Z(5) - Z(1))/r12(Z)^3;
 -G*m3*(Z(6) - Z(10))/r23(Z)^3 - G*m1*(Z(6) - Z(2))/r12(Z)^3;
 Z(11);
 Z(12);
 -G*m1*(Z(9) - Z(1))/r13(Z)^3 - G*m2*(Z(9) - Z(5))/r23(Z)^3;
 -G*m1*(Z(10) - Z(2))/r13(Z)^3 - G*m2*(Z(10) - Z(6))/r23(Z)^3;
];

[ts,R1] = ode45(f,timespan,z);

G = 1;
m1 = 1;
m2 = 1;
m3 = 1;

f = @(t,Z) [
 Z(3);
 Z(4);
 -G*m2*(Z(1) - Z(5))/r12(Z)^3 - G*m3*(Z(1) - Z(9))/r13(Z)^3;
```

```
 -G*m2*(Z(2) - Z(6))/r12(Z)^3 - G*m3*(Z(2) - Z(10))/
    r13(Z)^3;
 Z(7);
 Z(8);
 -G*m3*(Z(5) - Z(9))/r23(Z)^3 - G*m1*(Z(5) - Z(1))/
    r12(Z)^3;
 -G*m3*(Z(6) - Z(10))/r23(Z)^3 - G*m1*(Z(6) - Z(2))/
    r12(Z)^3;
 Z(11);
 Z(12);
 -G*m1*(Z(9) - Z(1))/r13(Z)^3 - G*m2*(Z(9) - Z(5))/
    r23(Z)^3;
 -G*m1*(Z(10) - Z(2))/r13(Z)^3 - G*m2*(Z(10) - Z(6))
    /r23(Z)^3;
];

[ts,R2] = ode45(f,timespan,z);

% Code for generation of figure 12,13,14.
figure(12)
clf
plot(data13(:,2),data13(:,3),'oc','LineWidth',0.1)
hold on;
plot(R2(:,1),R1(:,2));
plot(R1(:,1),R1(:,2));
xlabel("X")
ylabel("Y")
title("Figure 12: Results of using estimated
   parameters in", "comparison with measurement data
    for 1st body.");
legend("Measurement data.","Results before
   optimization","Results after optimization");

figure(13)
clf
plot(data13(:,4),data13(:,5),'oc','LineWidth',0.1)
hold on;
plot(R2(:,5),R1(:,6));
plot(R1(:,5),R1(:,6));
xlabel("X")
ylabel("Y")
```

```
title("Figure 13: Results of using estimated parameters in", "comparison with measurement data for 2nd body.");
legend("Measurement data.","Results before optimization","Results after optimization");

figure(14)
clf
plot(data13(:,6),data13(:,7),'oc','LineWidth',0.1)
hold on;
plot(R2(:,9),R1(:,10));
plot(R1(:,9),R1(:,10));
xlabel("X")
ylabel("Y")
title("Figure 14: Results of using estimated parameters in", "comparison with measurement data for 3rd body.");
legend("Measurement data.","Results before optimization","Results after optimization");
```

7.2 Supporting functions used in the implementation

7.2.1 Supporting function *AB2.m*

```
% This function returns the numerical solution to a given system of IVPs
% using the 2nd order Adam-bashford method.
function z_AB = AB2(f,dt,T,Z)

timespan = 0:dt:T;

z_AB = zeros(12,size(timespan,2));
z_AB(:,1) = Z;
z_AB(:,2) = Z + dt*f(0,Z);

for i = 3:size(timespan,2)
    z_AB(:,i) = z_AB(:,i-1) + dt*((3/2)*f(0,z_AB(:,i-1)) - (1/2)*f(0,z_AB(:,i-2)));
end
```

7.2.2 Supporting function *Kut.m*

```
% This function returns the numerical solution to a given system of IVPs
% using the Kutta method.
function z_K = Kut(f,dt,T,Z)

timespan = 0:dt:T;

z_K = zeros(12,size(timespan,2));
z_K(:,1) = Z;

for j = 2:size(timespan,2)
    f1 = f(0,z_K(:,j-1));
    f2 = f(0,z_K(:,j-1)+(dt/2)*f1);
    f3 = f(0,z_K(:,j-1)-dt*f1+2*dt*f2);
    z_K(:,j) = z_K(:,j-1) + (dt/6)*(f1+4*f2+f3);
end
```

7.2.3 Supporting function *RK4.m*

```
% This function returns the numerical solution to a given system of IVPs
% using the 4th order Runge-Kutta method.
function z_RK = RK4(f,dt,T,Z)

timespan = 0:dt:T;

z_RK = zeros(12,size(timespan,2));
z_RK(:,1) = Z;

for k = 2:size(timespan,2)
    f1 = f(0,z_RK(:,k-1));
    f2 = f(0,z_RK(:,k-1)+(dt/2)*f1);
    f3 = f(0,z_RK(:,k-1)+(dt/2)*f2);
    f4 = f(0,z_RK(:,k-1)+dt*f3);
    z_RK(:,k) = z_RK(:,k-1) + (dt/6)*(f1+2*f2+2*f3+f4);
end
```

7.2.4 Supporting function $J.m$

```
% This function is for use in finding the approximation criterion of the
% numerical solution given the vector of parameters P.
function j = J(P)

% Importing data from the file "data_13.csv".
data13 = readtable("Documents/MATLAB/enume-2024-mahmoud-elshekh-ali-323930/Assignment C/data_13.csv");
data13 = data13{:,:};
timespan = 0:0.02:5;

% Defining the parameters used in the planar three-body problem.
G = 1;
m1 = P(1);
m2 = P(2);
m3 = P(3);

% Defining the initial conditions for the system of IVPs used in the planar
% three-body problem.
z = [
    data13(1,2);
    data13(1,3);
    (data13(2,2) - data13(1,2))/0.02;
    (data13(2,3) - data13(1,3))/0.02;
    data13(1,4);
    data13(1,5);
    (data13(2,4) - data13(1,4))/0.02;
    (data13(2,5) - data13(1,5))/0.02;
    data13(1,6);
    data13(1,7);
    (data13(2,6) - data13(1,6))/0.02;
    (data13(2,7) - data13(1,7))/0.02;
    ];

```

```
% Defining the functions describing the distances between the bodies
% for use in the system of IVPs used in the planar three-body problem.
r12 = @(Z) sqrt((Z(1) - Z(5))^2 + (Z(2) - Z(6))^2);
r23 = @(Z) sqrt((Z(5) - Z(9))^2 + (Z(6) - Z(10))^2);
r13 = @(Z) sqrt((Z(1) - Z(9))^2 + (Z(2) - Z(10))^2);

% Defining the set of IVPs which make up the planar three-body
% problem.
f = @(t,Z) [
    Z(3);
    Z(4);
    -G*m2*(Z(1) - Z(5))/r12(Z)^3 - G*m3*(Z(1) - Z(9))/r13(Z)^3;
    -G*m2*(Z(2) - Z(6))/r12(Z)^3 - G*m3*(Z(2) - Z(10))/r13(Z)^3;
    Z(7);
    Z(8);
    -G*m3*(Z(5) - Z(9))/r23(Z)^3 - G*m1*(Z(5) - Z(1))/r12(Z)^3;
    -G*m3*(Z(6) - Z(10))/r23(Z)^3 - G*m1*(Z(6) - Z(2))/r12(Z)^3;
    Z(11);
    Z(12);
    -G*m1*(Z(9) - Z(1))/r13(Z)^3 - G*m2*(Z(9) - Z(5))/r23(Z)^3;
    -G*m1*(Z(10) - Z(2))/r13(Z)^3 - G*m2*(Z(10) - Z(6))/r23(Z)^3;
    ];

% Using ode45 to obtain a numerical solution using the parameters P
% to compare with the data imported from the file "data_13.csv".
[ts,R] = ode45(f,timespan,z);

% Finding the sought after approximation criterion.
j = sum( sqrt( (R(:,1) - data13(:,2)).^2 ) + sqrt( (R(:,2) - data13(:,3)).^2 ));
```

```
j = j + sum( sqrt( (R(:,5) - data13(:,4)).^2 ) +
    sqrt( (R(:,6) - data13(:,5)).^2 ) );
j = j + sum(sqrt( (R(:,9) - data13(:,6)).^2) + sqrt(
     (R(:,10) - data13(:,7)).^2 ) );

end
```