Ordinary differential equations

Michał Sarosiek

ENUME

Warsaw University of Technology, Faculty of Electronics and Information Technology

Tutor: Jakub Wagner

10 June 2024

Warsaw, Poland

I declare that this piece of work, which is the basis for recognition of achieving learning outcomes in the Numerical Methods course, was completed on my own.

Table of Contents

Notation

- t – time
- $x_k(t), y_k(t)$ – coordinates of the kth object's position $k = 1, 2, 3$
- m_k – kth body's mass with $k = 1, 2, 3$
- G – gravitational constant
- $\frac{d^n y}{dt^n}$ – nth derivative of y
- $r_{jk}(t)$ – function of time characterizing the distance between objects j and k
- $f(\cdot)$ – function characterizing an ODE
- h – step size of methods for solving ODEs
- Δ_{y_2} – mean-square error characterizing y_2
- $\mathbf{p}$ – vector of velocities
- $[\cdot]^T$ – operator of transposition

Introduction

The subject of this report is the planar three-body problem, a classical problem in celestial mechanics that involves predicting the motion of three bodies moving under their mutual gravitational attraction in a two-dimensional plane. This system is governed by a set of nonlinear second-order ordinary differential equations (ODEs). The report consists of solving these ODEs using various numerical methods, exploring specific solutions under certain conditions, and analyzing the performance of these methods.

MATLAB Function ode45

Ode45 is a versatile ODE solver in MATLAB, which is based on a variable-step Runge-Kutta (4,5) formula, also known as the Dormand-Prince method [1]. It is well-suited for solving non-stiff differential equations and provides high accuracy with adjustable tolerance parameters.

Euler Explicit Method

The Euler method is the simplest and most straightforward numerical procedure for solving initial value problems. It is an explicit first-order method where the next value is determined by the current value plus the product of the step size and the derivative at the current value [2].

Adams-Bashforth Method

This method is a multi-step explicit method that uses the information from the current and the previous steps to compute the next value [2]. This report utilizes the Adams-Bashforth Method of order 2.

Gear Explicit Method

Similarly to the Adams-Bashforth Method, it is a multi-step explicit method that computes the next value using data from both the current and prior steps [2]. For the purpose of this report, the Gear explicit method of order 2 was analyzed.

Summary

This report focuses on solving the planar three-body problem. Various numerical methods are employed, including MATLAB's ode45, the Euler explicit method, the Adams-Bashforth method of order 2, and the Gear explicit method of order 2. The analysis involves solving the ODEs for given initial conditions, finding specific parameter values for particular solutions, analyzing the accuracy of the methods through error computation, and estimating initial velocities from provided data.

Methodology and results of experimentation

This report analyzes a set of the following set of nonlinear second-order differential equations modelling the two-dimensional movement trajectories of three objects which attract each other:

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

where:

- t is the time,
- $x_k(t)$ and $y_k(t)$ are the coordinates of the kth object's position $k = 1, 2, 3$,
- m_k is the kth body's mass with $k = 1, 2, 3$,
- G is the gravitational constant,
- $r_{jk}(t) = \sqrt{\left(x_k(t) - x_j(t)\right)^2 + \left(y_k(t) - y_j(t)\right)^2}$ for $j, k = 1, 2, 3$.

A second-order ordinary differential equation:

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right) \quad (2)$$

can be expressed as a pair of first-order equations by adopting $z(t) \equiv \frac{dy(t)}{dt}$:

$$\begin{cases} \frac{dy(t)}{dt} = z(t) \\ \frac{dz(t)}{dt} = f(t, y, z) \end{cases} \quad (3)$$

Such a form was used for the purpose of the analysis.

Analysis of the trajectory obtained through several methods

This section covers the solutions obtained through several algorithms. The analyzed methods include MATLAB Function ode45, Euler explicit method, Adams-Bashforth method of order 2 and Gear explicit method of order 2. The parameters used are the following: $m_1 = m_2 = m_3 = G = 1$, $t \in [0,32.584945]$. For all the methods (other than ode45 as it chooses it on its own), the step of 10^{-3} was used. Exemplary initial conditions used for the purpose of presentation are presented in Table 1.

Table 1 Exemplary initial conditions for the planar three-body problem.

k	1	2	3
$x_k(0)$	0.0132604844	1.4157286016	−1.4289890859
$y_k(0)$	0.0000000000	0.0000000000	0.0000000000
$\frac{dx_k}{dt}(0)$	0.0000000000	0.0000000000	0.0000000000
$\frac{dy_k}{dt}(0)$	1.0541519210	−0.2101466639	−0.8440052572

MATLAB Function ode45

The obtained solution was achieved with parameters of absolute error and total error set to 10^{-12}. The function ode45 requires parameters such as: the function constituting (1), time span, the initial conditions. Having fulfilled those requirements through transforming (1) according to (3), the trajectory was calculated. It can be seen in Figure 1.

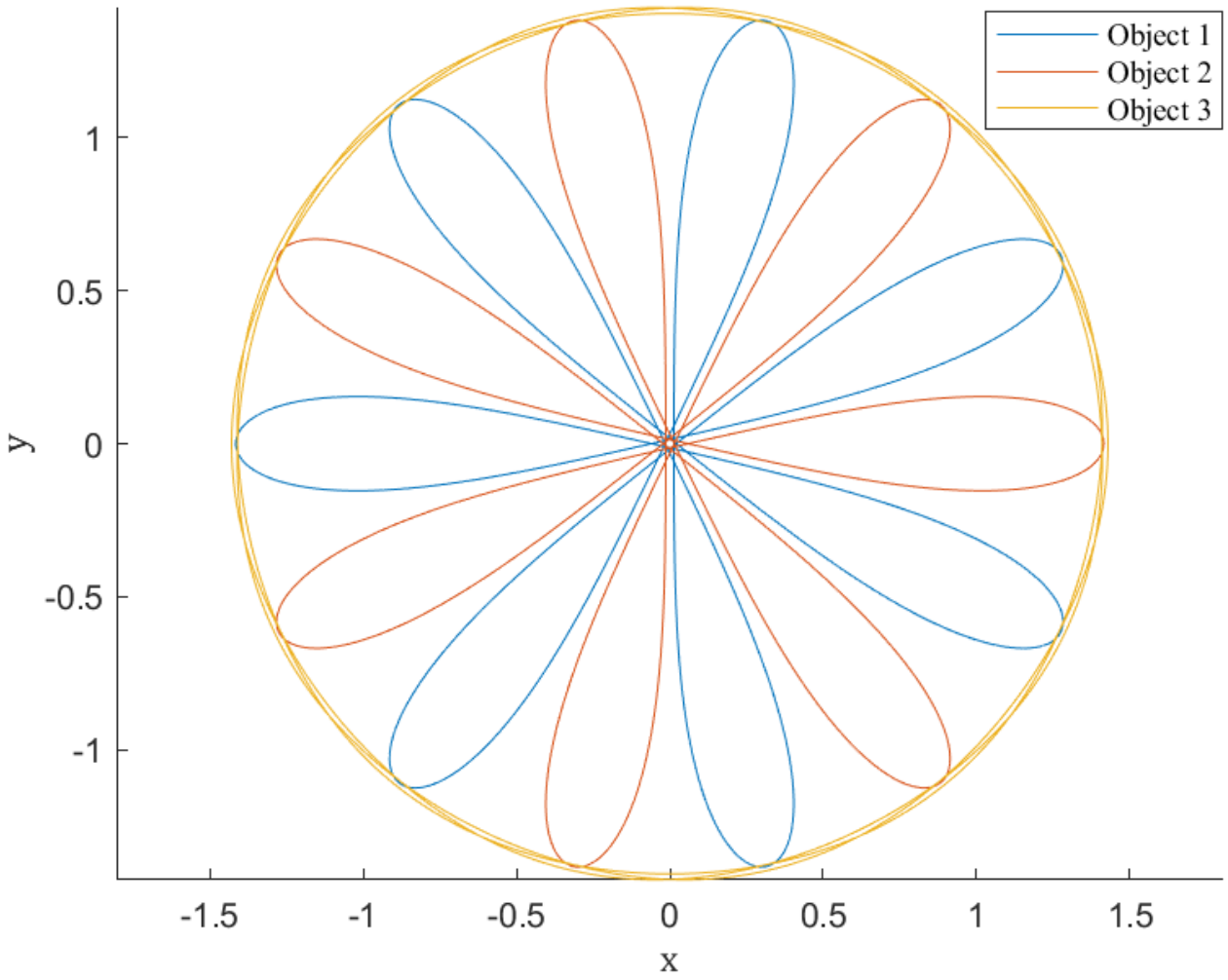

Figure 1. Position across $t \in [0,32.584945]$ calculated using function ode45.

Euler Explicit Method

The Euler Explicit Method can be denoted by the following formula:

$$y_n = y_{n-1} + hf(t_{n-1}, y_{n-1}) \tag{4}$$

where f denotes a function characterizing the ODE. As mentioned before, the used step (h) is equal to 10^{-3}. Having used the form of (3), the solution was directly obtained through the application of (4). It is presented in Figure 2.

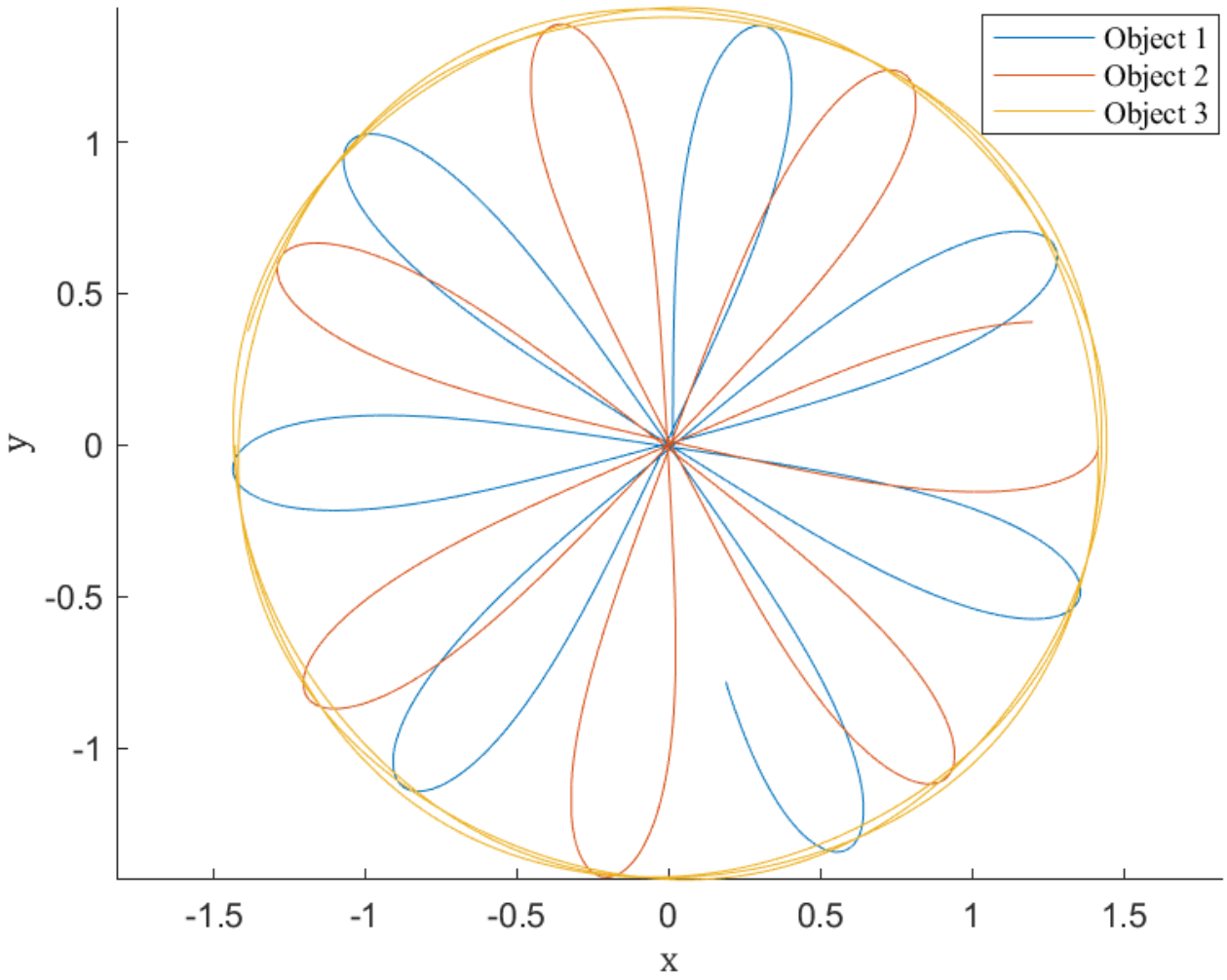

Figure 2. Position across $t \in [0,32.584945]$ calculated using the Euler Explicit Method.

Adams-Bashforth Method

The Adams-Bashforth Method of order 2 can be denoted by the following formula:

$$y_n = y_{n-1} + \frac{h}{2}[3f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2})] \tag{5}$$

Having directly applied (5), the results were calculated. They are presented in Figure 3.

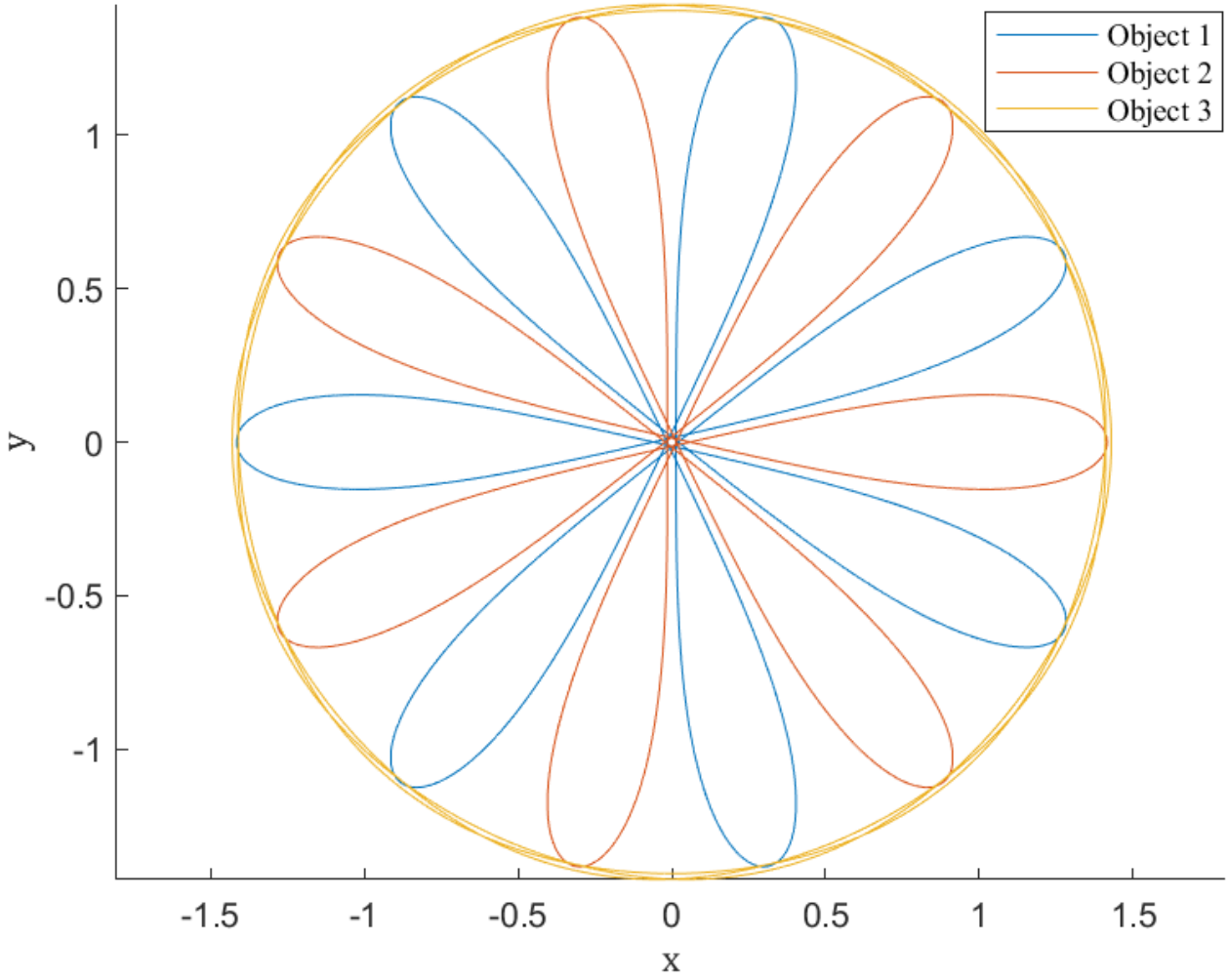

Figure 3. Position across $t \in [0,32.584945]$ calculated using the Adams-Bashforth Method of order 2.

Gear Explicit Method

The Gear Explicit Method of order 2 can be denoted by the following formula:

$$y_n = y_{n-2} + 2hf(t_{n-1}, y_{n-1}) \tag{6}$$

The results were obtained through the direct application of (6). They are depicted in Figure 4.

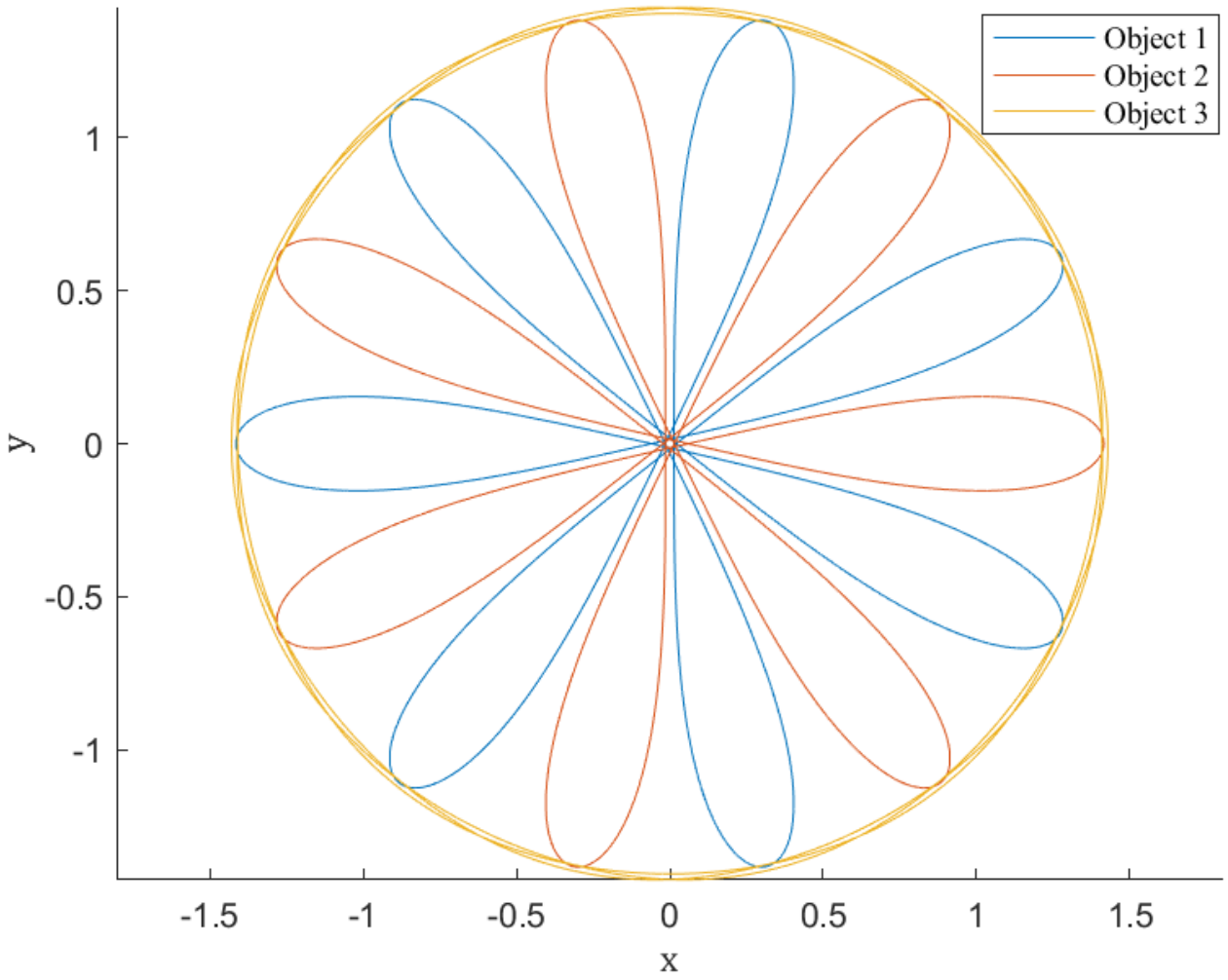

Figure 4. Position across $t \in [0,32.584945]$ calculated using the Gear Explicit Method of order 2.

Calculation of mass for the desired solution form

The desired solution form is the following:

$$\left.\begin{cases} x_1(t) = r_0 \sin\left(t + \frac{\pi}{6}\right) \\ x_2(t) = r_0 \sin\left(t + \frac{5}{6}\pi\right) \\ x_3(t) = r_0 \sin\left(t - \frac{\pi}{2}\right) \end{cases}\right| \text{ and } \begin{cases} y_1(t) = r_0 \cos\left(t + \frac{\pi}{6}\right) \\ y_2(t) = r_0 \cos\left(t + \frac{5}{6}\pi\right) \\ y_3(t) = r_0 \cos\left(t - \frac{\pi}{2}\right) \end{cases} \quad (7)$$

if $G = r_0 = 1$ and $m_1 = m_2 = m_3 = m$ Calculations used for obtaining the solution are presented below:

$$r_{12}(t) = \sqrt{\left(x_1(t) - x_2(t)\right)^2 + \left(y_1(t) - y_2(t)\right)^2}$$

$$r_{23}(t) = \sqrt{\left(x_2(t) - x_3(t)\right)^2 + \left(y_2(t) - y_3(t)\right)^2}$$

$$r_{31}(t) = \sqrt{\left(x_3(t) - x_1(t)\right)^2 + \left(y_3(t) - y_1(t)\right)^2}$$

Given $r_0 = 1$:

$$x_1(t) = \sin\left(t + \frac{\pi}{6}\right), \quad y_1(t) = \cos\left(t + \frac{\pi}{6}\right)$$

$$x_2(t) = \sin\left(t + \frac{5}{6}\pi\right), \quad y_2(t) = \cos\left(t + \frac{5}{6}\pi\right)$$

$$x_3(t) = \sin\left(t - \frac{\pi}{2}\right), \quad y_3(t) = \cos\left(t - \frac{\pi}{2}\right)$$

The second derivatives are:

$$\frac{d^2x_1(t)}{dt^2} = -\sin\left(t + \frac{\pi}{6}\right), \quad \frac{d^2y_1(t)}{dt^2} = -\cos\left(t + \frac{\pi}{6}\right)$$

$$\frac{d^2x_2(t)}{dt^2} = -\sin\left(t + \frac{5}{6}\pi\right), \quad \frac{d^2y_2(t)}{dt^2} = -\cos\left(t + \frac{5}{6}\pi\right)$$

$$\frac{d^2x_3(t)}{dt^2} = -\sin\left(t - \frac{\pi}{2}\right), \quad \frac{d^2y_3(t)}{dt^2} = -\cos\left(t - \frac{\pi}{2}\right)$$

The distances (solved utilizing the Pythagorean identity):

$$r_{12}(t) = \sqrt{\left(\sin\left(t + \frac{\pi}{6}\right) - \sin\left(t + \frac{5}{6}\pi\right)\right)^2 + \left(\cos\left(t + \frac{\pi}{6}\right) - \cos\left(t + \frac{5}{6}\pi\right)\right)^2} = \sqrt{3}$$

$$r_{23}(t) = \sqrt{\left(\sin\left(t + \frac{5}{6}\pi\right) - \sin\left(t - \frac{\pi}{2}\right)\right)^2 + \left(\cos\left(t + \frac{5}{6}\pi\right) - \cos\left(t - \frac{\pi}{2}\right)\right)^2} = \sqrt{3}$$

$$r_{31}(t) = \sqrt{\left(\sin\left(t - \frac{\pi}{2}\right) - \sin\left(t + \frac{\pi}{6}\right)\right)^2 + \left(\cos\left(t - \frac{\pi}{2}\right) - \cos\left(t + \frac{\pi}{6}\right)\right)^2} = \sqrt{3}$$

Given $G = 1$ and $m_1 = m_2 = m_3 = m$

$$\frac{d^2x_1(t)}{dt^2} = -m\left(\frac{x_1(t) - x_2(t)}{r_{12}^3(t)} + \frac{x_1(t) - x_3(t)}{r_{31}^3(t)}\right)$$

$$-\sin\left(t + \frac{\pi}{6}\right) = -m\left(\frac{\sin\left(t + \frac{\pi}{6}\right) - \sin\left(t + \frac{5}{6}\pi\right)}{\left(\sqrt{3}\right)^3} + \frac{\sin\left(t + \frac{\pi}{6}\right) - \sin\left(t - \frac{\pi}{2}\right)}{\left(\sqrt{3}\right)^3}\right)$$

$$m = \frac{\sin\left(t+\frac{\pi}{6}\right)}{\frac{\sin\left(t+\frac{\pi}{6}\right)-\sin\left(t+\frac{5}{6}\pi\right)}{\left(\sqrt{3}\right)^3}+\frac{\sin\left(t+\frac{\pi}{6}\right)-\sin\left(t-\frac{\pi}{2}\right)}{\left(\sqrt{3}\right)^3}}$$

$$m = \frac{3\sqrt{3}}{2-\frac{\sin\left(t+\frac{5}{6}\pi\right)+\sin\left(t-\frac{\pi}{6}\right)}{\sin\left(t+\frac{\pi}{6}\right)}}$$

$$m = \frac{3\sqrt{3}}{2-\frac{\sin\left(t+\frac{\pi}{6}\right)\cos\left(\frac{2}{3}\pi\right)}{\sin\left(t+\frac{\pi}{6}\right)}}$$

$$m = \frac{3\sqrt{3}}{2+1} = \sqrt{3} \tag{8}$$

Analysis of accuracy of methods

For this section the calculated mass (8) was used alongside the initial conditions resulting from (7). The calculated initial positions are the following:

$$\begin{cases} x_1(0) = \sin\left(\frac{\pi}{6}\right) = \frac{1}{2} \\ x_2(0) = \sin\left(\frac{5}{6}\pi\right) = \frac{1}{2} \\ x_3(0) = \sin\left(-\frac{\pi}{2}\right) = -1 \end{cases} \text{ and } \begin{cases} y_1(0) = \cos\left(\frac{\pi}{6}\right) = \frac{\sqrt{3}}{2} \\ y_2(0) = \cos\left(\frac{5}{6}\pi\right) = -\frac{\sqrt{3}}{2} \\ y_3(0) = \cos\left(-\frac{\pi}{2}\right) = 0 \end{cases} \tag{9}$$

The initial velocities were calculated as such:

$$\begin{cases} \left.\frac{dx_1(t)}{dt}\right|_{t=0} = \cos\left(0+\frac{\pi}{6}\right) = \cos\left(\frac{\pi}{6}\right) = \frac{\sqrt{3}}{2} \\ \left.\frac{dx_2(t)}{dt}\right|_{t=0} = \cos\left(\frac{5}{6}\pi\right) = -\frac{\sqrt{3}}{2} \\ \left.\frac{dx_3(t)}{dt}\right|_{t=0} = \cos\left(-\frac{\pi}{2}\right) = 0 \end{cases} \text{ and } \begin{cases} \left.\frac{dy_1(t)}{dt}\right|_{t=0} = -\sin\left(\frac{\pi}{6}\right) = -\frac{1}{2} \\ \left.\frac{dy_2(t)}{dt}\right|_{t=0} = -\sin\left(\frac{5}{6}\pi\right) = -\frac{1}{2} \\ \left.\frac{dy_3(t)}{dt}\right|_{t=0} = -\sin\left(-\frac{\pi}{2}\right) = 1 \end{cases} \tag{10}$$

In this part $t \in [0,2\pi]$.

The analysis pertains y_2 and all previously used methods (excluding ode45). As a metric of accuracy, mean-square error was used. It is denoted by Δ_{y_2} and its definition is as follows:

$$\Delta_{y_2} = \frac{1}{N}\sum_{n=1}^{N}\left[\hat{y}_{2,n} - y_2(t_n)\right]^2 \tag{11}$$

where $\hat{y}_{2,1}, \ldots, \hat{y}_{2,N}$ are the estimates of $y_2(t_1), \ldots, y_2(t_N)$, obtained numerically and $y_2(t)$ is the reference solution, defined by one of the equations in the set (7).

For presentation of results, $h \in [10^{-3}, 10^{0}]$ was used.

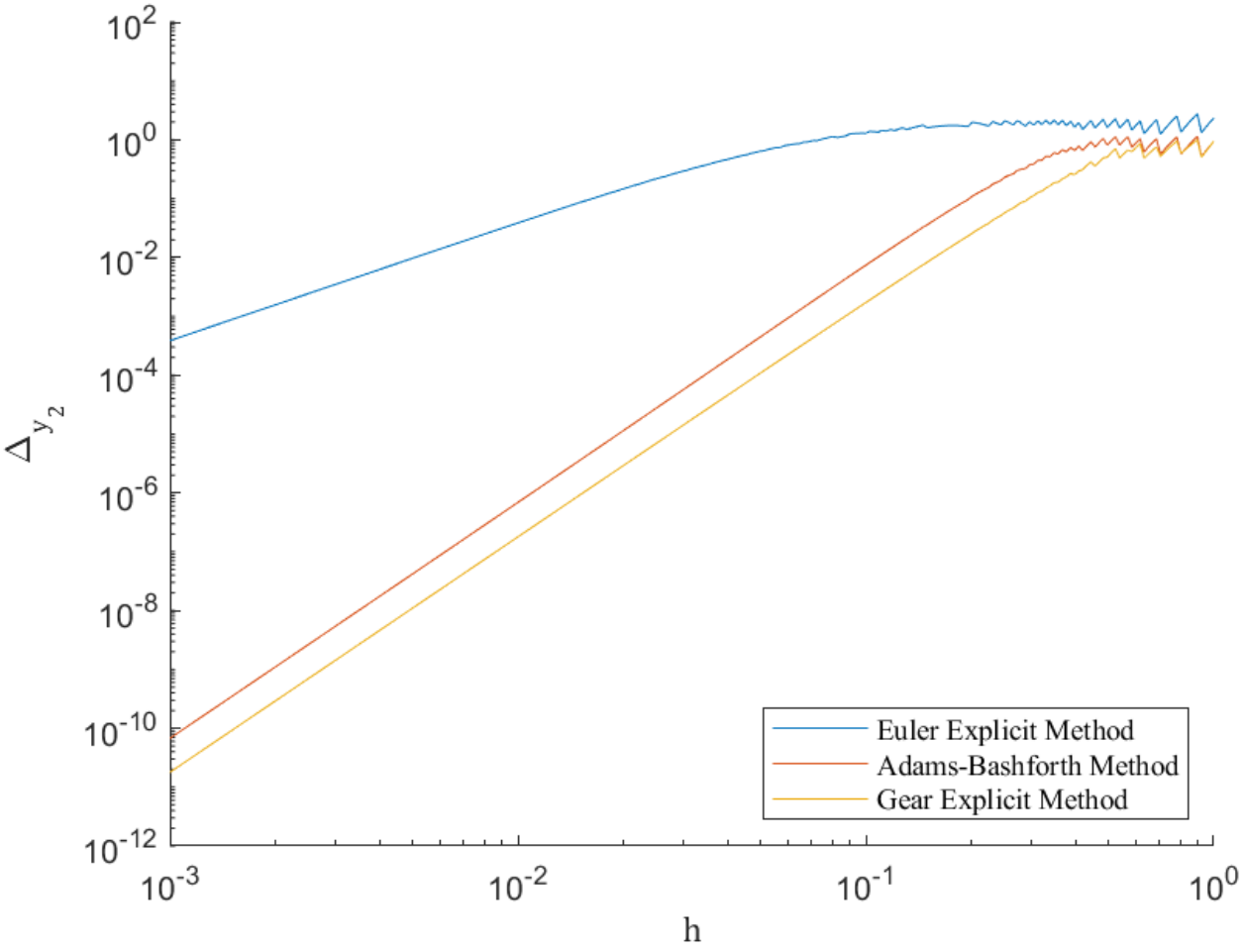

Figure 5. Dependence of the mean-square error on the step of integration for Euler Explicit Method, Adams-Bashforth Method of order 2 and Gear Explicit Method of order 2.

Estimation of initial velocities

This section covers the estimation of initial velocities given a set of positions in time instances. The analysis was done for $G = m_1 = m_2 = m_3 = 1$.

It is important to note that, the initial velocities can't be accurately estimated through the following equation:

$$\frac{dx(t)}{dt} = \frac{x_{n+1} - x_n}{t_{n+1} - t_n} \tag{12}$$

as such an estimation yields erroneous trajectory. Therefore, it was made through the minimization of the following criterion:

$$\begin{aligned} J(p) = &\sum_{n=1}^{N} \sqrt{\left[\hat{x}_1(t_n; p) - \tilde{x}_{1,n}\right]^2 + \left[\hat{y}_1(t_n; p) - \tilde{y}_{1,n}\right]^2} \\ &+ \sum_{n=1}^{N} \sqrt{\left[\hat{x}_2(t_n; p) - \tilde{x}_{2,n}\right]^2 + \left[\hat{y}_2(t_n; p) - \tilde{y}_{2,n}\right]^2} \\ &\sum_{n=1}^{N} \sqrt{\left[\hat{x}_3(t_n; p) - \tilde{x}_{3,n}\right]^2 + \left[\hat{y}_3(t_n; p) - \tilde{y}_{3,n}\right]^2} \end{aligned} \tag{13}$$

where **p** is the vector of velocities to be estimated:

$$\mathbf{p} = \left[\frac{dx_1(0)}{dt} \frac{dy_1(0)}{dt} \frac{dx_2(0)}{dt} \frac{dy_2(0)}{dt} \frac{dx_3(0)}{dt} \frac{dy_3(0)}{dt}\right]^T \tag{14}$$

and $\hat{x}_k$ and $\hat{y}_k$ are functions modelling the coordinates of the trajectory of kth body obtained through solving (1) with the parameters **p**.

The process of minimization was done using MATLAB's function fminsearch which utilized (13) to minimize (14) starting from the points resulting from (12). The results of this process can be seen in Table 2.

Table 2. Estimated initial velocities.

$\frac{dx_1(0)}{dt}$	0.725359552559566
$\frac{dy_1(0)}{dt}$	0.00343891969394009
$\frac{dx_2(0)}{dt}$	-0.373901025070656
$\frac{dy_2(0)}{dt}$	-0.629540103929511
$\frac{dx_3(0)}{dt}$	-0.371860831386569
$\frac{dy_3(0)}{dt}$	0.640514560568173

The given set of coordinates, trajectory resulting from (12) and the trajectory with initial velocities contained in Table 2, for objects 1, 2, 3 can be seen in Figure 6, Figure 7 and Figure 8 accordingly.

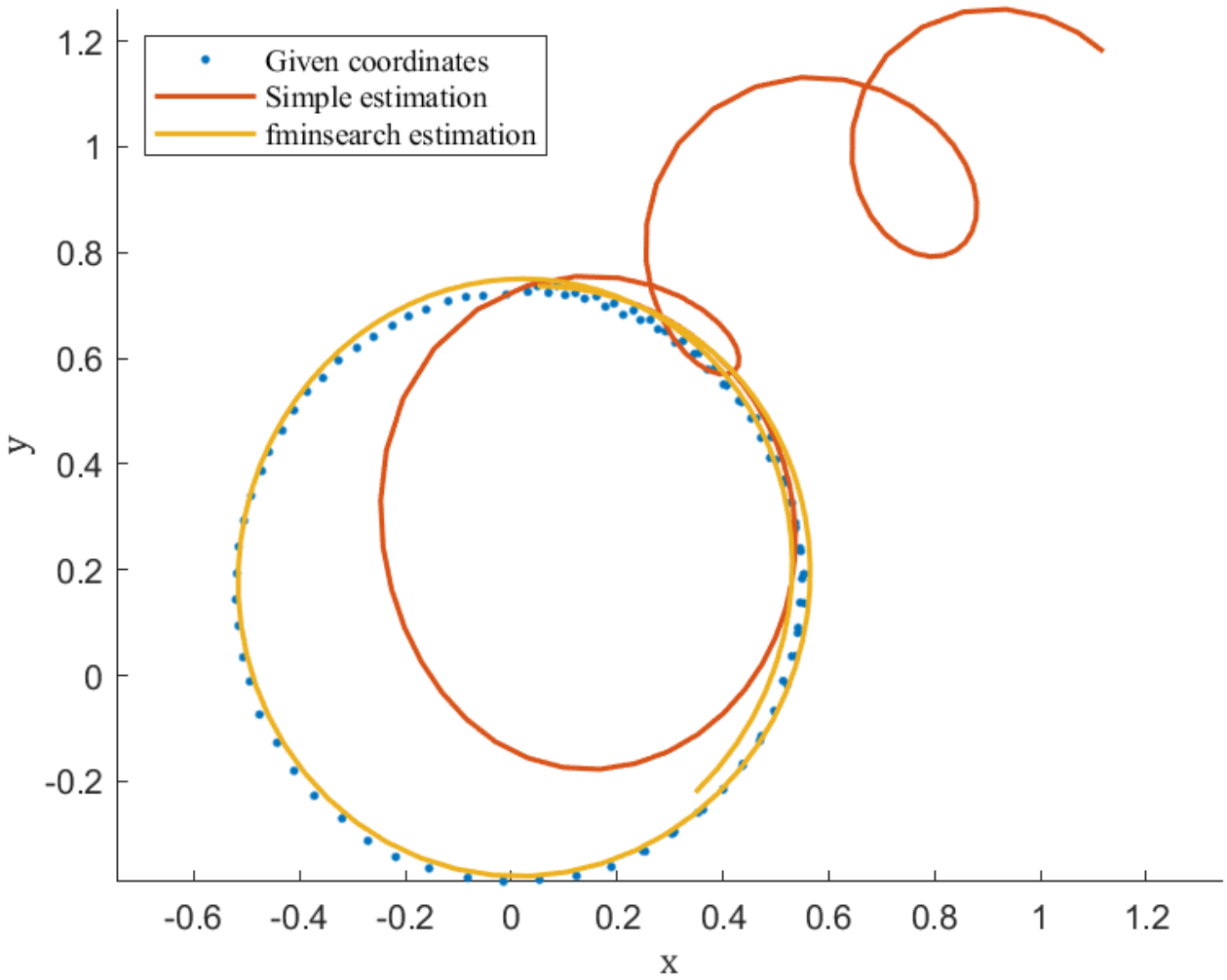

Figure 6. Object 1: The given coordinates, the trajectory resulting from (12) and the trajectory resulting from fminsearch estimation.

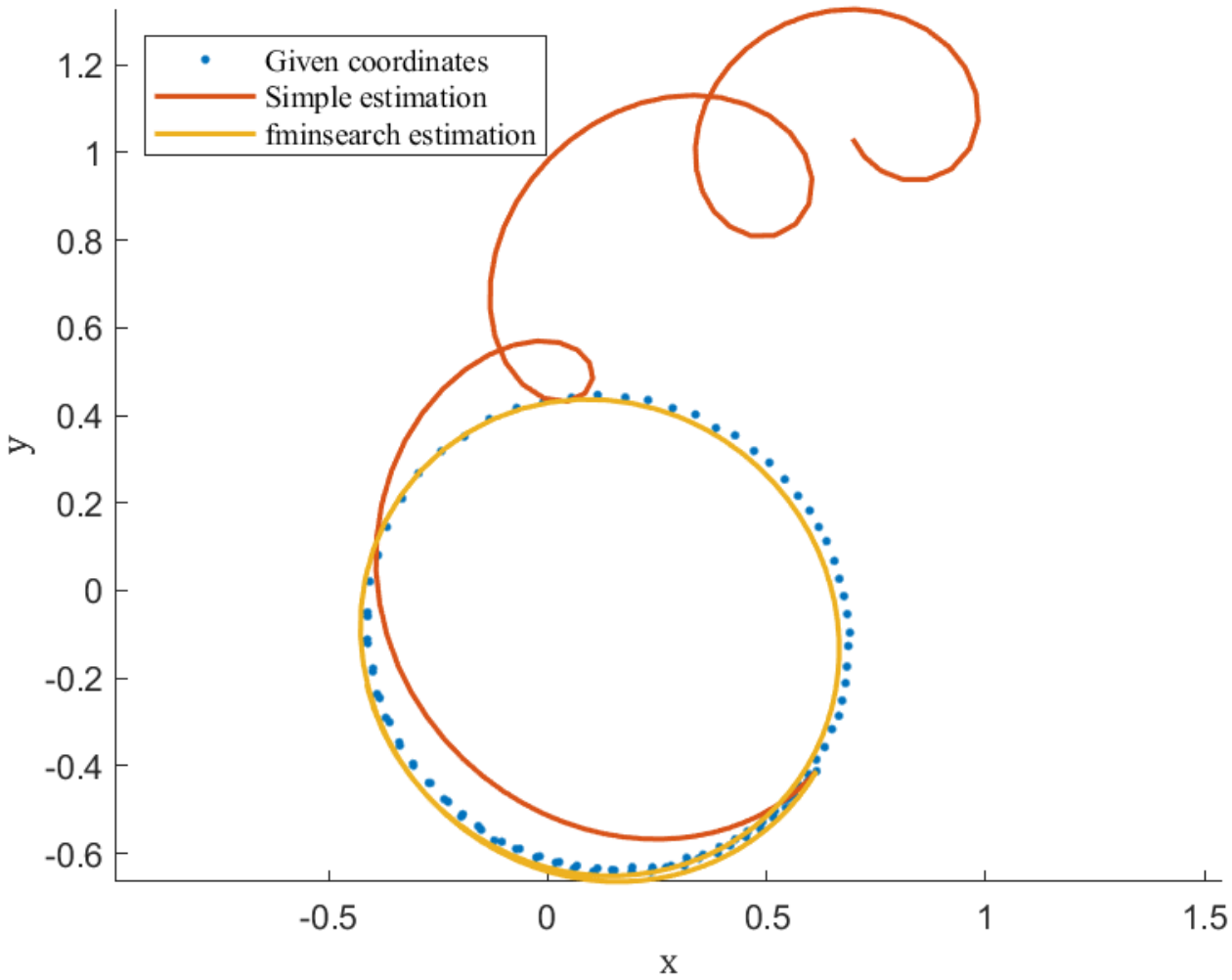

Figure 7. Object 2: The given coordinates, the trajectory resulting from (12) and the trajectory resulting from fminsearch estimation.

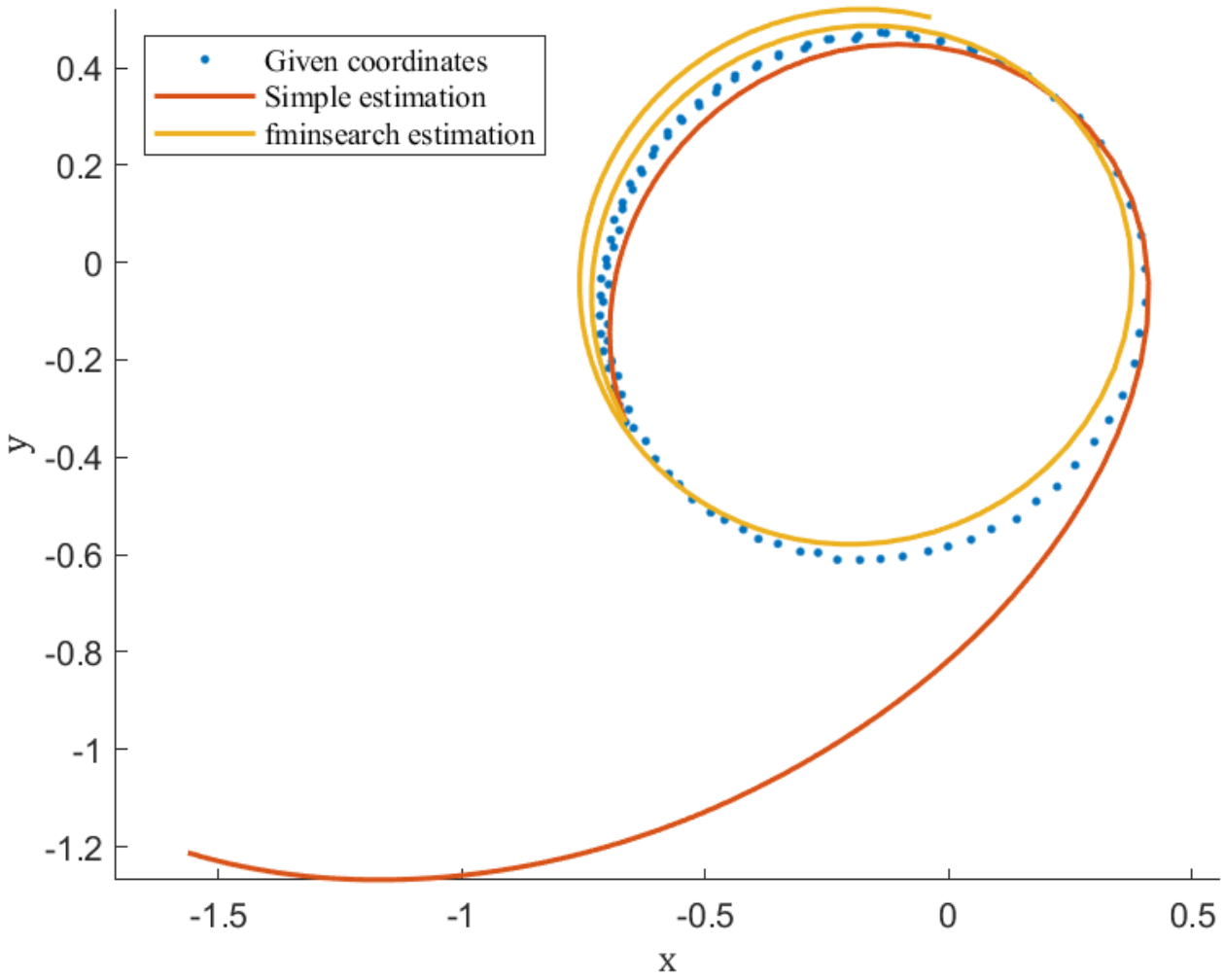

Figure 8. Object 3: The given coordinates, the trajectory resulting from (12) and the trajectory resulting from fminsearch estimation.

Discussion

Analysis of the trajectory obtained through several methods

Based on Figure 1, Figure 2, Figure 3 and Figure 4 we can see that the initial conditions presented in Table 1 result in a stable pattern akin to a flower. Therefore, the initial conditions are part of a periodic solution to the presented planar three-body problem.

MATLAB Function ode45 as well as Adams-Bashforth Method and Gear Explicit Method resulted in trajectories of great enough accuracy to satisfy the periodic solution in the allotted time span. Euler Explicit Method, on the other hand, resulted in visible inconsistencies with the desired shape.

One can conclude that the chosen step of $h = 10^{-3}$ was appropriate for the two step methods but insufficient for the Euler Explicit Method. As such, the two step methods seem vastly more accurate. MATLAB Function ode45 also proves to be a reliable way of solving ODEs.

Analysis of accuracy of methods

Magnitude of the mean-square error presented in Figure 5 confirms the correctness of mass calculations made in (8).

The obtained characteristics confirm the assumption of accuracy made solely based on the resulting trajectory. The Euler Explicit Method proved to be significantly less accurate compared to the Adams-Bashforth Method and the Gear Explicit Method.

Each of the methods' mean-square error seems to result in a linear characteristic with relation to step size. The accuracy decreases with the decrease in h. This behavior continues up to a certain point where the step size becomes large enough for the error to become more unpredictable. This threshold seems to be based on the accuracy of the method as it appeared the earliest in the Euler Explicit Method and the latest in the Gear Explicit Method.

Estimation of initial velocities

The resulting trajectories presented in Figure 6, Figure 7 and Figure 8 prove the inaccuracy of estimated velocities obtained through (12). This can be explained through the imperfect nature of the measurements.

The minimization of the criterion (13) using fminsearch proves to be a significantly more accurate solution. The resulting trajectory follows the data points relatively close. Nonetheless, the estimation is not perfect which can be noticed in the deviations from the desired positions.

When choosing an estimation algorithm, it is important to account for imperfections of the model. As such, utilization of more robust methods of calculation is advised.

References

[1] R. Z. Morawski, Numerical Methods (ENUME), Lecture notes for Spring Semester 2023/2024.

[2] MathWorks, "ode45," [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode45.html. [Accessed 9 June 2024].

Listing of the developed programs

file: Task1.m

```
clearvars
%{
m1 = 1;
m2 = 1;
m3 = 1;
G = 1;
%}
obj1_xy = [0.0132604844, 0];
obj2_xy = [1.4157286016, 0];
obj3_xy = [-1.4289890859, 0];
obj1_dt = [0, 1.0541519210];
obj2_dt = [0, -0.2101466639];
obj3_dt = [0, -0.8440052572];
z0 = [obj1_xy, obj2_xy, obj3_xy, obj1_dt, obj2_dt, obj3_dt];
tspan = [0, 32.584945];

%ode45
options = odeset('AbsTol', 1e-12, 'RelTol', 1e-12);
[t, z] = ode45(@odeFun, tspan, z0, options);
figure(1)
clf
hold on;
plot(z(:,1), z(:, 2));
plot(z(:,3), z(:, 4));
plot(z(:,5), z(:, 6));
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Object 1', 'Object 2', 'Object 3', 'FontName', 'Times New Roman');
axis equal
hold off;

%euler explicit
h = 0.001;
t_steps = tspan(1):h:tspan(2);
z_euler = zeros(length(z0), length(t_steps));
z_euler(:,1) = z0;
for n = 2:(length(t_steps))
    z_euler(:, n) = z_euler(:,n-1) + h*(odeFun(t_steps(n-1), z_euler(:,n-1)));
end
figure(2)
clf
hold on;
plot(z_euler(1, :), z_euler(2, :));
plot(z_euler(3, :), z_euler(4, :));
plot(z_euler(5, :), z_euler(6, :));
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Object 1', 'Object 2', 'Object 3', 'FontName', 'Times New Roman');
axis equal
hold off;
```

```
%Adams-Bashforth of order 2
z_adams = zeros(length(z0), length(t_steps));
z_adams(:,1) = z0;
z_adams(:, 2) = z_adams(:,1) + h*(odeFun(t_steps(1), z_adams(:,1)));
for n = 3:(length(t_steps))
    z_adams(:, n) = z_adams(:,n-1) + (h/2)*(3*odeFun(t_steps(n-1), z_adams(:,n-1)) -
odeFun(t_steps(n-2), z_adams(:,n-2)));
end
figure(3)
clf
hold on;
plot(z_adams(1, :), z_adams(2, :));
plot(z_adams(3, :), z_adams(4, :));
plot(z_adams(5, :), z_adams(6, :));
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Object 1', 'Object 2', 'Object 3', 'FontName', 'Times New Roman');
axis equal
hold off;

%Gear explicit of order 2
z_gear = zeros(length(z0), length(t_steps));
z_gear(:,1) = z0;
z_gear(:, 2) = z_gear(:,1) + h*(odeFun(t_steps(1), z_gear(:,1)));
for n = 3:(length(t_steps))
    z_gear(:, n) = z_gear(:,n-2) + h*(2*odeFun(t_steps(n-1), z_gear(:,n-1)));
end
figure(4)
clf
hold on;
plot(z_gear(1, :), z_gear(2, :));
plot(z_gear(3, :), z_gear(4, :));
plot(z_gear(5, :), z_gear(6, :));
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Object 1', 'Object 2', 'Object 3', 'FontName', 'Times New Roman');
axis equal
hold off;

function vector = odeFun(t, w)
    xy1 = w(1:2);
    xy2 = w(3:4);
    xy3 = w(5:6);
    dt1 = w(7:8);
    dt2 = w(9:10);
    dt3 = w(11:12);
    r12=norm(xy1 - xy2);
    r31=norm(xy3 - xy1);
    r23=norm(xy2 - xy3);

    a1 = (-1)*(xy1 - xy2)/(r12)^3 - (xy1 - xy3)/(r31)^3;
    a2 = (-1)*(xy2 - xy3)/(r23)^3 - (xy2 - xy1)/(r12)^3;
    a3 = (-1)*(xy3 - xy1)/(r31)^3 - (xy3 - xy2)/(r23)^3;
```

```
    vector = [dt1; dt2; dt3; a1; a2; a3];
end
```

file: Task3.m

```
clearvars
obj1_xy = [0.5, sqrt(3)/2];
obj2_xy = [0.5, -sqrt(3)/2];
obj3_xy = [-1, 0];
obj1_dt = [sqrt(3)/2, -1/2];
obj2_dt = [-sqrt(3)/2, -1/2];
obj3_dt = [0, 1];
z0 = [obj1_xy, obj2_xy, obj3_xy, obj1_dt, obj2_dt, obj3_dt];
tspan = [0, 2*pi];

figure(1)
clf
hold on;
yscale('log');
xscale('log');
h = logspace(-3,0,256);

%euler explicit
z_euler_delta = zeros(1, length(h));
for i = 1:length(h)
    t_steps = tspan(1):h(i):tspan(2);
    z_euler = zeros(length(z0), length(t_steps));
    z_euler(:,1) = z0;
    for n = 2:(length(t_steps))
        z_euler(:, n) = z_euler(:,n-1) + h(i)*(odeFun(t_steps(n-1), z_euler(:,n-1)));
    end
    z_euler_y2_ref = zeros(1, length(t_steps));
    z_euler_y2_ref(1, :) = cos(t_steps + (5/6)*pi);
    z_euler_delta(i) = 1/(length(t_steps))*sum((z_euler(4, :) - z_euler_y2_ref).^2);
end
plot(h, z_euler_delta);

%Adams-Bashforth of order 2
z_adams_delta = zeros(1, length(h));
for i = 1:length(h)
    t_steps = tspan(1):h(i):tspan(2);
    z_adams = zeros(length(z0), length(t_steps));
    z_adams(:,1) = z0;
    z_adams(:, 2) = z_adams(:,1) + h(i)*(odeFun(t_steps(1), z_adams(:,1)));
    for n = 3:(length(t_steps))
        z_adams(:, n) = z_adams(:,n-1) + (h(i)/2)*(3*odeFun(t_steps(n-1),
z_adams(:,n-1)) - odeFun(t_steps(n-2), z_adams(:,n-2)));
    end
    z_adams_y2_ref = zeros(1, length(t_steps));
    z_adams_y2_ref(1, :) = cos(t_steps + (5/6)*pi);
    z_adams_delta(i) = 1/(length(t_steps))*sum((z_adams(4, :) - z_adams_y2_ref).^2);
end
plot(h, z_adams_delta);
```

```
%Gear explicit of order 2
z_gear_delta = zeros(1, length(h));
for i = 1:length(h)
    t_steps = tspan(1):h(i):tspan(2);
    z_gear = zeros(length(z0), length(t_steps));
    z_gear(:,1) = z0;
    z_gear(:, 2) = z_gear(:,1) + h(i)*(odeFun(t_steps(1), z_gear(:,1)));
    for n = 3:(length(t_steps))
        z_gear(:, n) = z_gear(:,n-2) + h(i)*(2*odeFun(t_steps(n-1), z_gear(:,n-1)));
    end
    z_gear_y2_ref = zeros(1, length(t_steps));
    z_gear_y2_ref(1, :) = cos(t_steps + (5/6)*pi);
    z_gear_delta(i) = 1/(length(t_steps))*sum((z_gear(4, :) - z_gear_y2_ref).^2);
end
plot(h, z_gear_delta);
ylabel('\Delta_{y_2}', 'FontName', 'Cambria Math');
xlabel('h', 'FontName', 'Cambria Math');
legend('Euler Explicit Method', 'Adams-Bashforth Method', 'Gear Explicit Method',
'FontName', 'Times New Roman', 'Location','southeast');

hold off;

function vector = odeFun(t, w)
    xy1 = w(1:2);
    xy2 = w(3:4);
    xy3 = w(5:6);
    dt1 = w(7:8);
    dt2 = w(9:10);
    dt3 = w(11:12);
    r12=norm(xy1 - xy2);
    r31=norm(xy3 - xy1);
    r23=norm(xy2 - xy3);

    a1 = (-sqrt(3))*(xy1 - xy2)/(r12)^3 - (sqrt(3))*(xy1 - xy3)/(r31)^3;
    a2 = (-sqrt(3))*(xy2 - xy3)/(r23)^3 - (sqrt(3))*(xy2 - xy1)/(r12)^3;
    a3 = (-sqrt(3))*(xy3 - xy1)/(r31)^3 - (sqrt(3))*(xy3 - xy2)/(r23)^3;

    vector = [dt1; dt2; dt3; a1; a2; a3];
end
```

file: Task4.m

```
clearvars;
table = readtable("data_42.csv");
obj1_xy = [table.x1(1), table.y1(1)];
obj2_xy = [table.x2(1), table.y2(1)];
obj3_xy = [table.x3(1), table.y3(1)];
t_change = table.t(2)-table.t(1);
obj1_dt = [(table.x1(2)-table.x1(1))/t_change, (table.y1(2)-table.y1(1))/t_change];
obj2_dt = [(table.x2(2)-table.x2(1))/t_change, (table.y2(2)-table.y2(1))/t_change];
obj3_dt = [(table.x3(2)-table.x3(1))/t_change, (table.y3(2)-table.y3(1))/t_change];
p_ini = [obj1_dt, obj2_dt, obj3_dt];
z0 = [obj1_xy, obj2_xy, obj3_xy, p_ini];
```

```
options = odeset('AbsTol', 1e-12, 'RelTol', 1e-12);
[t, z] = ode45(@odeFun, table.t, z0, options);

%parameters = fminsearch(@minimisation, p_ini);
%Approximated parameters
parameters = [0.725359552559566, 0.00343891969394009,      -0.373901025070656, -
0.629540103929511,  -0.371860831386569, 0.640514560568173];
z0_approx = [obj1_xy, obj2_xy, obj3_xy, parameters];
[t, z_approx] = ode45(@odeFun, table.t, z0_approx, options);

figure(1);
clf;
hold on;
axis equal
p = plot(table.x1, table.y1, '.');
p.MarkerSize = 8;
plot(z(:,1), z(:, 2), LineWidth = 1.5);
plot(z_approx(:,1), z_approx(:, 2), LineWidth = 1.5);
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Given coordinates', 'Simple estimation', 'fminsearch estimation', 'FontName',
'Times New Roman', 'Location', 'northwest');


hold off;

figure(2);
clf;
hold on;
axis equal
p = plot(table.x2, table.y2, '.');
p.MarkerSize = 8;
plot(z(:,3), z(:, 4), LineWidth = 1.5);
plot(z_approx(:,3), z_approx(:, 4), LineWidth = 1.5);
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Given coordinates', 'Simple estimation', 'fminsearch estimation', 'FontName',
'Times New Roman', 'Location', 'northwest');
hold off;

figure(3);
clf;
hold on;
axis equal
p = plot(table.x3, table.y3, '.');
p.MarkerSize = 8;
plot(z(:,5), z(:, 6), LineWidth = 1.5);
plot(z_approx(:,5), z_approx(:, 6), LineWidth = 1.5);
ylabel('y', 'FontName', 'Cambria Math');
xlabel('x', 'FontName', 'Cambria Math');
legend('Given coordinates', 'Simple estimation', 'fminsearch estimation', 'FontName',
'Times New Roman', 'Location', 'northwest');
hold off;

function vector = odeFun(t, w)
```

```
    xy1 = w(1:2);
    xy2 = w(3:4);
    xy3 = w(5:6);
    dt1 = w(7:8);
    dt2 = w(9:10);
    dt3 = w(11:12);
    r12=norm(xy1 - xy2);
    r31=norm(xy3 - xy1);
    r23=norm(xy2 - xy3);

    a1 = (-1)*(xy1 - xy2)/(r12)^3 - (1)*(xy1 - xy3)/(r31)^3;
    a2 = (-1)*(xy2 - xy3)/(r23)^3 - (1)*(xy2 - xy1)/(r12)^3;
    a3 = (-1)*(xy3 - xy1)/(r31)^3 - (1)*(xy3 - xy2)/(r23)^3;

    vector = [dt1; dt2; dt3; a1; a2; a3];
end

function J = minimisation(p)
    table = readtable("data_42.csv");
    xy1_ini = [table.x1(1), table.y1(1)];
    xy2_ini = [table.x2(1), table.y2(1)];
    xy3_ini = [table.x3(1), table.y3(1)];
    initial_parameters = [xy1_ini, xy2_ini, xy3_ini, p];
    options = odeset('AbsTol', 1e-12, 'RelTol', 1e-12);
    [t, z] = ode45(@odeFun, table.t, initial_parameters, options);
    xy1 = z(:, 1:2);
    xy2 = z(:, 3:4);
    xy3 = z(:, 5:6);

    J = sum(norm(xy1 - [table.x1, table.y1])) + sum(norm(xy2 - [table.x2, table.y2]))
+ sum(norm(xy3 - [table.x3, table.y3]));

end
```