Advanced Numerical Methods
SF2520

Fall term 2021

Computer Exercise 1

Initial Value Problems

David Stuhrmann

19980508-T271

`dast9356@student.su.se`

Sofia Nguyen

19980625

`song0575@student.su.se`

September 19, 2021

Part 1: Accuracy and stability of a Runge-Kutta method

We are given a Runge-Kutta method for solving a first order ODE of the form $y'(t) = f(t, y)$ where $y \in \mathbb{R}^d$ may be a vector. The method computes three auxiliary values k_1, k_2 and k_3. The time steps are assumed to be equidistant $t_n = nh$ of step size h and the points are numbered by $n = 0, 1, 2, \ldots, N$.

$$\begin{aligned} k_1 &= f\left(t_n, u_n\right), \\ k_2 &= f\left(t_n + h, u_n + hk_1\right), \\ k_3 &= f\left(t_n + h/2, u_n + hk_1/4 + hk_2/4\right), \\ u_{n+1} &= u_n + \frac{h}{6}\left(k_1 + k_2 + 4k_3\right) \end{aligned}$$

The values of $u_n \in \mathbb{R}^d$ approximate the values $y(t_n)$.

The ODE system which shall be examined in this exercise is of the form

$$\frac{\mathrm{d}m}{\mathrm{d}t} = a \times m + \alpha\, a \times (a \times m) \tag{1}$$

with the parameter $\alpha = 0.1$, the constant vector $a = (\frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{2}})^T$ and the initial condition $m(0) = (1, 0, 0)^T$.

As it is needed later the right hand side of equation (1) will be written as Am with $A \in \mathbb{R}^{3\times3}$. Note that the vector product $(a \times \_)$ can also be written as a matrix product with the matrix B.

$$B = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \tag{2}$$

Rewriting f:

$$\begin{aligned} f(t, m) &= a \times m + \alpha\, a \times (a \times m) && (3) \\ &= Bm + \alpha B^2 m && (4) \\ &= B(I + \alpha B)m && (5) \\ &= Am && (6) \end{aligned}$$

So the system matrix is $A = B(I + \alpha B)$, where I denotes the 3×3 identity matrix.

Our implementation of this Runge-Kutta method is done in a Matlab function.

```
[t, u] = rungeKutta(f, h, N, y0)
```

The arguments are the function `f` from the right hand side of equation (1), the step size `h`, the number `N` of steps such that $Nh = T$, the end time of the interval over which the ODE should be solved and the last is the initial condition `y0`. The `rungeKutta` function returns the time steps `t` and approximation values `u`.

We plot the three components of m over time $[0, 40]$ in figure 1. For the solution we used the Runge-Kutta method with $N = 80$ points, thus the step size was $h = 0.5$ and the initial condition $m_0 = (1, 0, 0)^T$. The sample points are marked with circles. From the plot it is visible that the components perform a damped oscillation resulting to match the direction of the given vector a.

More clearly it is shown in figure 2 that m becomes parallel to a. In the plot it is shown in orange the direction of m which is just $\frac{m}{\|m\|}$. We see that it spirals towards the tip of the blue line which is a reference from the origin into the direction of a.

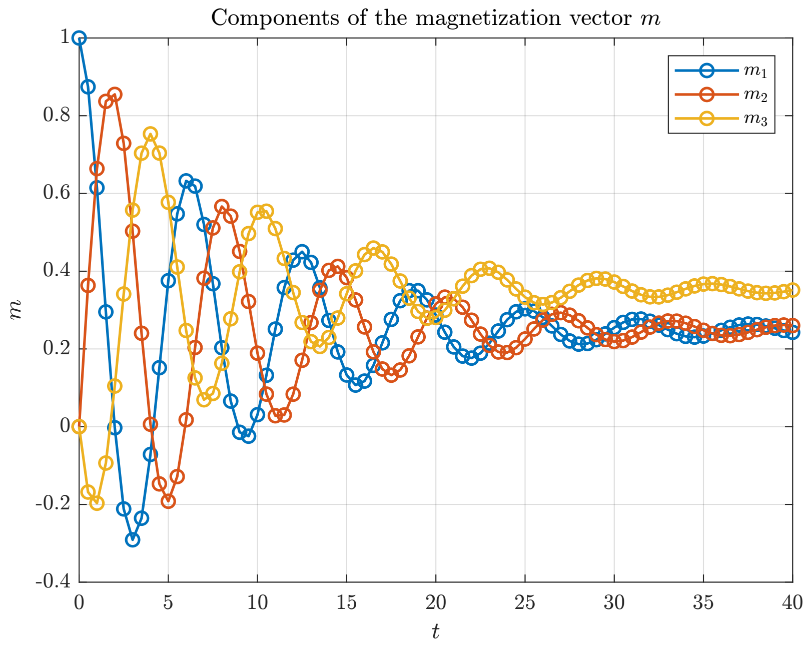


Figure 1: Numerical solution by Runge-Kutta to the magnetization ODE in the time interval $[0, 40]$ with initial condition $m_0 = (1, 0, 0)^T$. Shown are the three components of the magnetization vector m. The solver used $N = 80$ steps, so the step size was $h = 0.5$.

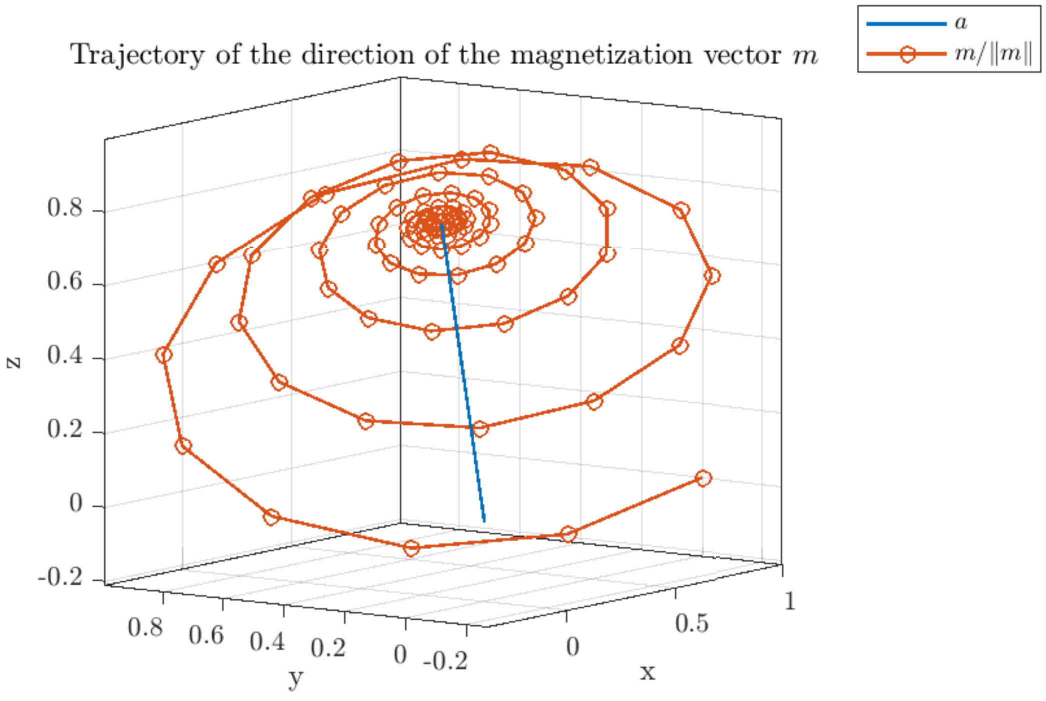


Figure 2: Trajectory (orange) of the normalized magnetization vector m in the time interval $[0, 40]$. The step size was $h = 0.5$ with $N = 80$ points. The blue line is the reference direction of the fixed vector $a \approx (0.500, 0.5000.707)^T$.

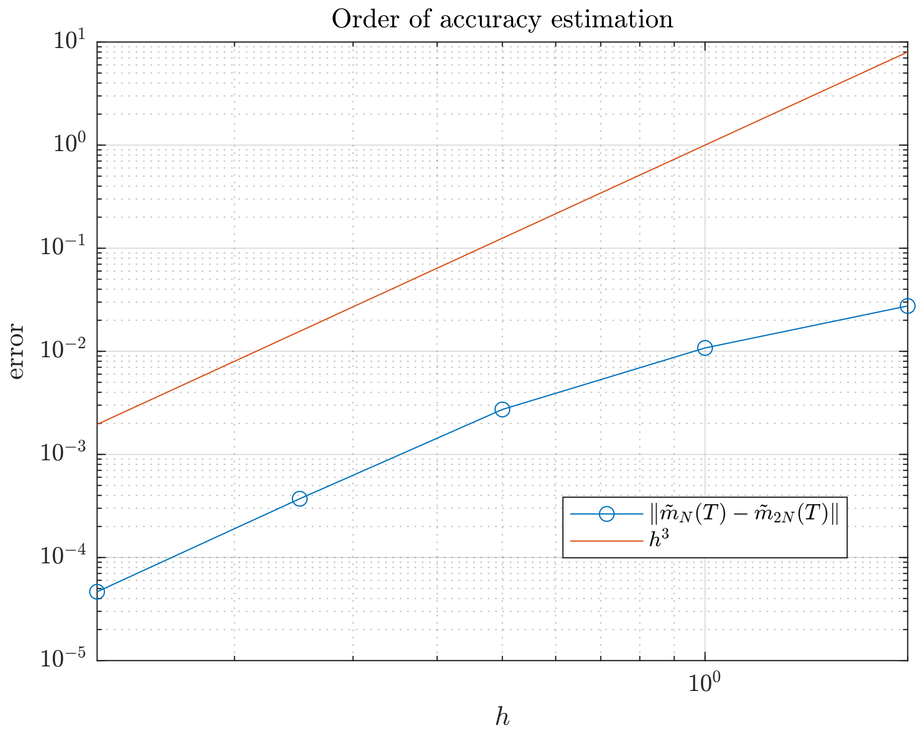


Figure 3: Estimation of the order of accuracy of the Runge-Kutta method. Double-logarithmic plot of the norm of the differences at the end of the time interval $[0, 40]$. The step size was decreased by factors of $\frac{1}{2}$. As Reference the line for the order h^3 is plotted in orange.

In the next step we asses the order of accuracy of the Runge-Kutta method. For this we run the method for different step sizes corresponding to $N = 20, 40, 80, 160, 320, 640$ steps. In the end we always take the last approximation value which is $\tilde{m}_N(T)$. Because the the number of points is doubled in each successive run we can take the norm of the differences in the approximations $\|\tilde{m}_N(T) - \tilde{m}_{2N}(T)\|$. The error estimate this then plotted against $h(N)$ in a double logarithmic plot, see figure 3. For the smaller step sizes h we see that the blue points indicating the error estimate seem to lie on a straight line. As a reference the orange line shows the h^3 accuracy which seems to match the order of accuracy of this method.

In the last part of we look at the stability limit for the step size since we use an explicit method. Earlier we already found the system matrix $A = B(I + \alpha B)$ for the ODE $\frac{\mathrm{d}m}{\mathrm{d}t} = Am$. If we already know the eigenvectors of B then it follows that those are also eigenvectors of A. Let $Bv = \mu v$ then $Av = B(I + \alpha B)v = \mu(1 + \alpha\mu)v$. Thus the eigenvalues w. r. t. to A is $\lambda = \mu(1 + \alpha\mu)$. With the explicit form of B in terms of the components of a is is simple to compute the eigenvalues of B. Set up the characteristic polynomial.

$$\begin{aligned}
\det(\mu I - B) &= \begin{vmatrix} \mu & a_3 & -a_2 \\ -a_3 & \mu & a_1 \\ a_2 & -a_1 & \mu \end{vmatrix} && (7)\\
&= \mu^3 + a_1a_2a_3 - a_1a_2a_3 + a_1^2\mu + a_3^2\mu + a_2^2\mu && (8)\\
&= \mu(\mu^2 + \|a\|^2) && (9)
\end{aligned}$$

The zeros of the polynomial are $\mu = 0$ and $\mu = \pm\mathrm{i}\|a\|$. As these are three distinct values we know that there also exist three eigenvectors and that B can be diagonalized. Thus we also have all three eigenvectors of A. The eigenvalues are $\lambda = 0$, $\lambda = \pm\mathrm{i}\|a\| - \alpha\|a\|$. Since $\|a\| = 1$, the eigenvalues become simply $\{0, \mathrm{i} - \alpha, -\mathrm{i} - \alpha\}$. A quick check with Matlab's command `eig(A)` confirm the values $\{0, -0.1 + \mathrm{i}, -0.1 - \mathrm{i}\}$.

Next we have to identify the stability region S, which is the points in the complex plane for which initial errors of the test problem $y' = \lambda y$ will vanish in the limit. For this assume two approximations u and $\tilde{u}$ where $\tilde{u}_0 - u_0 = \delta$ a small disturbance. The difference between both approximations is $w_n = \tilde{u}_n - u_n$, so $w_0 = \delta$. The recursion for w_n can be calculated by applying the Runge-Kutta method.

$$\begin{aligned}
w_{n+1} &= \tilde{u}_{n+1} - u_{n+1} && (10)\\
&= \tilde{u}_n + \frac{h}{6}(\tilde{k}_1 + \tilde{k}_2 + 4\tilde{k}_3) - u_n - \frac{h}{6}(k_1 + k_2 + 4k_3) && (11)\\
&= \tilde{u}_n - u_n + \frac{h}{6}(\tilde{k}_1 - k_1 + \tilde{k}_2 - k_2 + 4(\tilde{k}_3 - k_3) && (12)\\
&= w_n + \frac{h}{6}\Big(\lambda w_n + \lambda(w_n + h\lambda w_n) + 4\Big(\lambda\big(w_n + \frac{h}{4}\lambda w_n + \frac{h}{4}\lambda(w_n + h\lambda w_n)\big)\Big)\Big) && (13)\\
&= \Big(1 + \frac{1}{6}h\lambda + \frac{1}{6}h\lambda(1 + h\lambda) + \frac{2}{3}\Big(h\lambda\big(1 + \frac{1}{4}h\lambda + \frac{1}{4}h\lambda + \frac{1}{4}(h\lambda)^2\big)\Big)\Big)w_n && (14)\\
&= \Big(1 + \frac{1}{6}h\lambda + \frac{1}{6}h\lambda + \frac{1}{6}(h\lambda)^2 + \frac{2}{3}h\lambda + \frac{1}{3}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3\Big)w_n && (15)\\
&= \Big(1 + h\lambda + \frac{1}{2}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3\Big)w_n && (16)\\
&= C(h\lambda)w_n && (17)
\end{aligned}$$

where $C(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3$. So the stability region is given by

$$S = \{z \in \mathbb{C} : |C(z)| < 1\} \tag{18}$$

To find the step size h such that the solution by the Runge-Kutta method is absolute stable means that $h\lambda \in S$ for all eigenvalues λ of A.

The stability limit can now be computed by finding the zeros of the function $f_{\text{stab}}(h) = |C(h\lambda| - 1$ for each eigenvalue λ. Because if $f_{\text{stab}}(h_{\text{stab}}) = 0$ then we are just at the boundary of S which means $|C(h_{\text{stab}}\lambda)| = 1$. To find the zeros we used Matlab's build in function `fzero`, which finds zeros of non-linear functions, and the starting point $h_0 = 2$.

λ	h_{stab}
0	2
$-0.1 + \mathrm{i}$	2.1420
$-0.1 - \mathrm{i}$	2.1420

(19)

The eigenvalue $\lambda = 0$ can be ignored because in the time evolution $\mathrm{e}^{\lambda t} = \mathrm{e}^{0t} = 1$ is just constant so the initial error will in the direction of the corresponding eigenvector just stay the same for the entire solution. Now look at the value $h_{\text{stab}} \approx 2.1420$. Both eigenvalues are complex conjugates of each other so it is no surprise that they have the same stability limit.

The plots verifying the stability limits are shown in figures 4 and 5 for the now longer time interval $[0, 80]$ to show the behavior of the solution. With $h \approx 2.1053$, corresponding to 57 steps, the step size is just below the stability limit. Figure 4 which shows the components of m shows this stable solution. Note that the stability does not mean that the solution is accurate which you can see by comparing it with figure 1.

Now with 56 steps we have $h \approx 2.1429$ which is slightly above the stability limit and as it can be seen in figure 5. The components of m just oscillate with a large error and do not converge.

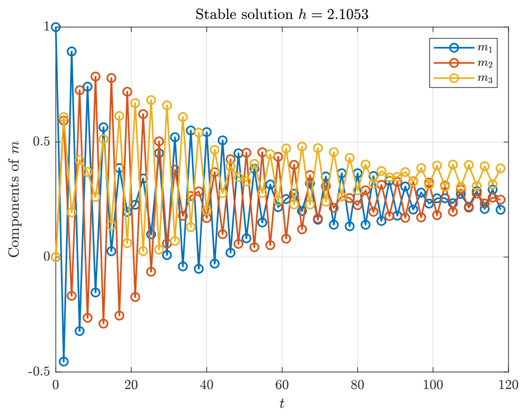


Figure 4: Stable solution just below the stability limit of $h = 2.1420$. Shown are the components of the vector m. The initial condition was $m_0 = (1, 0, 0)^T$ and $N = 57$ steps were used.

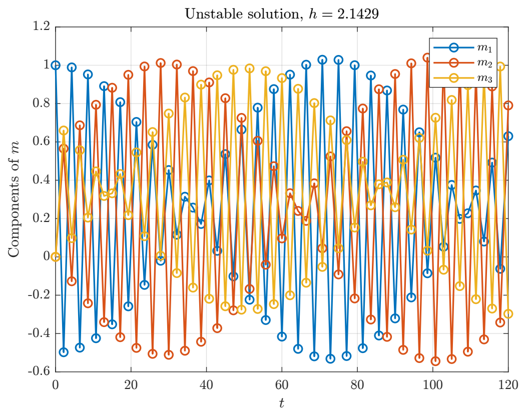


Figure 5: Unstable solution just above the stability limit of $h = 2.1420$. Shown are the components of the vector m. The initial condition was $m_0 = (1, 0, 0)^T$ and $N = 56$ steps were used.

Part 2: Multi step method for a three-body problem

In this problem we shall use the help of the previous method of Runge-Kutta and implement it to be able to use the Adam-Bashford 4 method, which is an explicit method needing three previous steps.

With the initial values given

$r_0 = \begin{pmatrix} -\mu \\ 0 \end{pmatrix}$, $r_1 = \begin{pmatrix} 1-\mu \\ 0 \end{pmatrix}$, $r(0) = \begin{pmatrix} 1.2 & 0 \end{pmatrix}^T$ and $r'(0) = \begin{pmatrix} 0 & -1 \end{pmatrix}^T$

we can rewrite the equation to vector form

$$\frac{d^2}{dt^2}\begin{pmatrix} x \\ y \end{pmatrix} = -(1-\mu)\frac{\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} -\mu \\ 0 \end{pmatrix}}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} -\mu \\ 0 \end{pmatrix} \right\|^3} - \mu \frac{\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 1-\mu \\ 0 \end{pmatrix}}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 1-\mu \\ 0 \end{pmatrix} \right\|^3} + 2\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \tag{20}$$

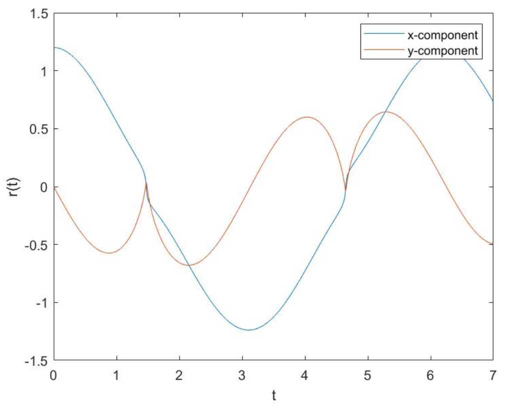
Rewrite this second ODE by $\frac{d\mathbf{r}}{dt} = \mathbf{u}$ and thus $\frac{d^2\mathbf{r}}{dt^2} = \frac{d\mathbf{y}}{dt}$.

We then achieve the final four column vector for
f($\mathbf{r}$)=

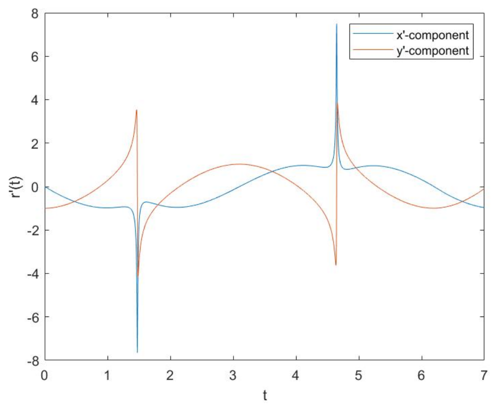
$$\frac{du}{dt} = \begin{pmatrix} u_x \\ u_y \\ -(1-\mu)\frac{x+\mu}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} -\mu \\ 0 \end{pmatrix} \right\|^3} - \mu \frac{(x-1+\mu)}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 1-\mu \\ 0 \end{pmatrix} \right\|^3} + 2\frac{dy}{dt} + x \\ -(1-\mu)\frac{y}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} -\mu \\ 0 \end{pmatrix} \right\|^3} - \mu \frac{(y)}{\left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 1-\mu \\ 0 \end{pmatrix} \right\|^3} - 2\frac{dx}{dt} + y \end{pmatrix} \tag{21}$$

Figure 6 is plotted with $N = 7000$ and $h = 0.001$ i.e. the time $t = [0, 7]$ where we have plotted the position coordinates $\mathbf{r}(t) = (x(t), y(t)))$, the velocity $\mathbf{r}'(t) = (x'(t), y'(t))$. In the last subplot we have the position coordinates of the satellite i.e. $x(t)$ against $y(t)$ where the position of the moon and earth are shown.

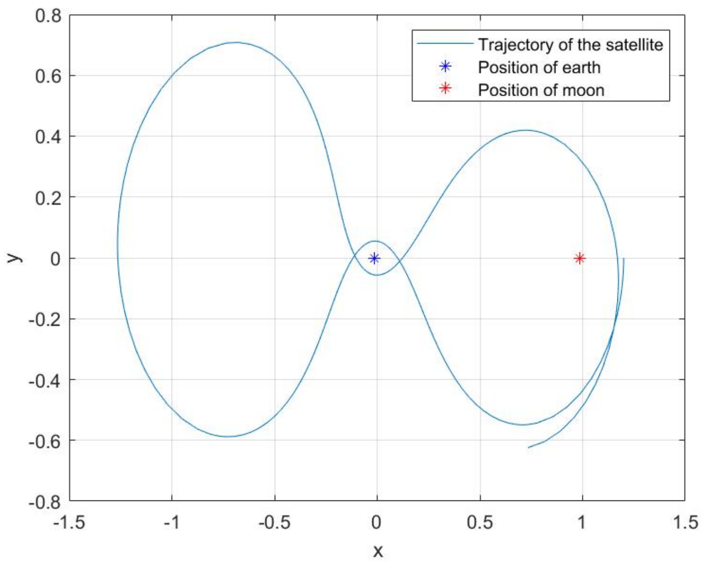
Using three different methods of Explicit Euleur, Runge-Kutta and the Adam-Bashford method, we analyse the accuracy obtained during computing. We specifically want to have an accuracy where $|\mathbf{r}_h(t) - \mathbf{r}_{h/2}(t)| \leq TOL$ where we set $TOL = 0.25$. The results can be seen in table 1.



(a) The position coordinates as a function of time



(b) The velocities as a function of time



(c) The trajectory of the satellite plotted wit position of the moon and earth in a time interval of t=[0,7]

Figure 6

Methods	Time accuracy (T_{acc})
Explicit Euler	2.089
Runge-Kutta	10.995
Adam-Bashford	19.729

Table 1: The time accuracy for the three different methods where h = 0.001 and the tolerance is0.25

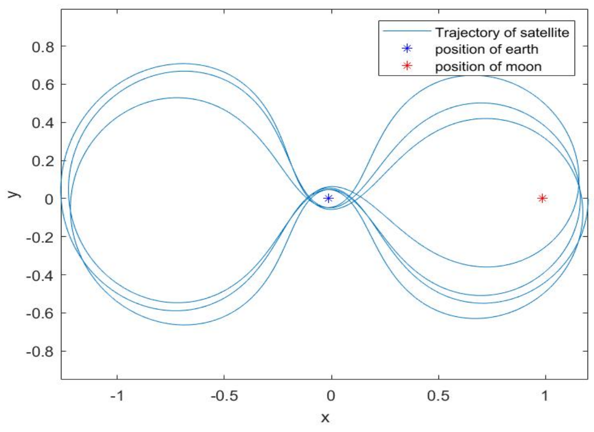


Figure 7: Trajectory of the satellite up to T_{acc} for the Adam-Bashford 4 method

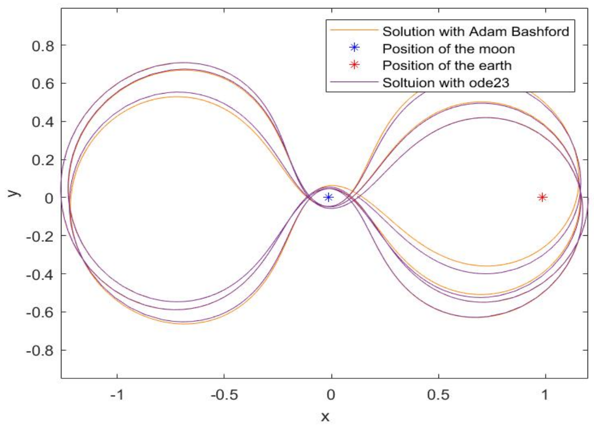


Figure 8: Trajectory of the satellite up to T_{acc} using ode23 in MATLAB compared to the solution from the Adam Bashford method.

Runge-Kutta is a third-order method, which means that the truncation error satisfies an order of $O(h^3)$, meanwhile the Explicit Euler is a first order method i.e. the truncation error being of order $O(h^1)$. It can be noted from table 1 that the time accuracy holds longer for the Adam-Bashford which are expected due to the Adam-Bashford being a fourth order method i.e. satisfies $O(h^4)$

When solving the function in Equation (21) with the *ode* function for $T_{acc} = 19.729$ we get trajectory given in Figure 8 which is relatively equal. This is used as verifying that our solution with the Adam-Bashford method is valid.

The number of steps used when solving with the in-built ode23 function is 979 compared to 19729 steps using the Adam-Bashford method. The biggest and the smallest time steps found when using the ode23 function is 0.1009 and 0.002 respectively.

The different sizes of time-steps used by the ODE to T_{acc} are plotted in figure 9 below.

The larger steps can be explained by smaller values of the derivative of the position coordinates because then the error is not so large. The smaller time-steps in the trajectory plot in figure 8 can thus be identified as the positions where the satellite is making a turn i.e. the lowest and highest positions in terms of y positions and also when crossing the position of earth because then the time derivatives are quite large.

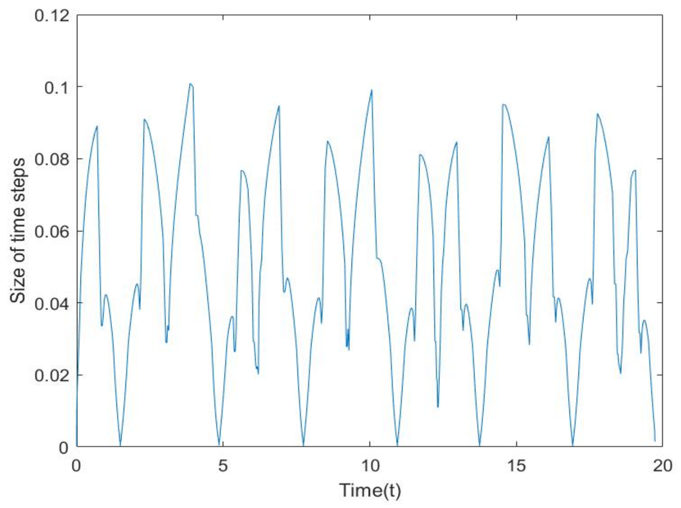


Figure 9: Trajectory of the satellite up to T_{acc} using ode23 in MATLAB

Part 3: A stiff system

The part is about a stiff system which means that we usually need a small step size to get a stable solution when using an explicit method.

The Robertson's problem describes three reactions which change the concentrations x_1, x_2 and x_3 of three substances. We group them together into the vector $x = (x_1, x_2, x_3)^T$. Then the non-linear system of ODEs is described by

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = F(x) = \begin{pmatrix} -r_1x_1 + r_2x_2x_3 \\ r_1x_1 - r_2x_2x_3 - r_3x_2^2 \\ r_3x_2^2 \end{pmatrix}, \quad x(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{22}$$

The rate constants given are $r_1 = 0.04$, $r_2, 10^4$ and $r_3 = 3 \cdot 10^7$.

At first we try to solve the problem with an explicit method in the time interval $[0, 1]$. The runs with $N = 125, 250, 500, 1000, 2000$ reveal that for 1000 steps we obtain a stable solution but for 500 steps it was not the case.

The solution with $N = 1000$ steps is shown in figure 10. From left to right the plots show the concentrations x_1, x_2 and x_3 in a double logarithmic plot over time.

In the next part we investigate the different build-in methods of Matlab. First we use the explicit solver `ode23` on the same time interval $[0, 1]$. We specify different relative tolerances `RelTol` and absolute tolerances `AbsTol` $=$ `RelTol`$/1000$. As the method is an adaptive one the number of steps can change. The second solver we compare with is the implicit solver `ode23s`. Here the time interval is choosen to be $[0, 1000]$. The method is also adaptive which means it has a variable step size. Table 2 lists the number of steps that the two solver used for the four tested relative tolerances.

Table 2: Number of steps used by the solver `ode23`/`ode23s` for different relative tolerances `RelTol` and absolute tolerances `AbsTol` $=$ `RelTol`$/1000$.

`RelTol`	steps `ode23` in $[0, 1]$	steps `ode23s` in $[0, 1000]$
10^{-3}	867	31
10^{-4}	873	53
10^{-5}	879	115
10^{-11}	3360	58425

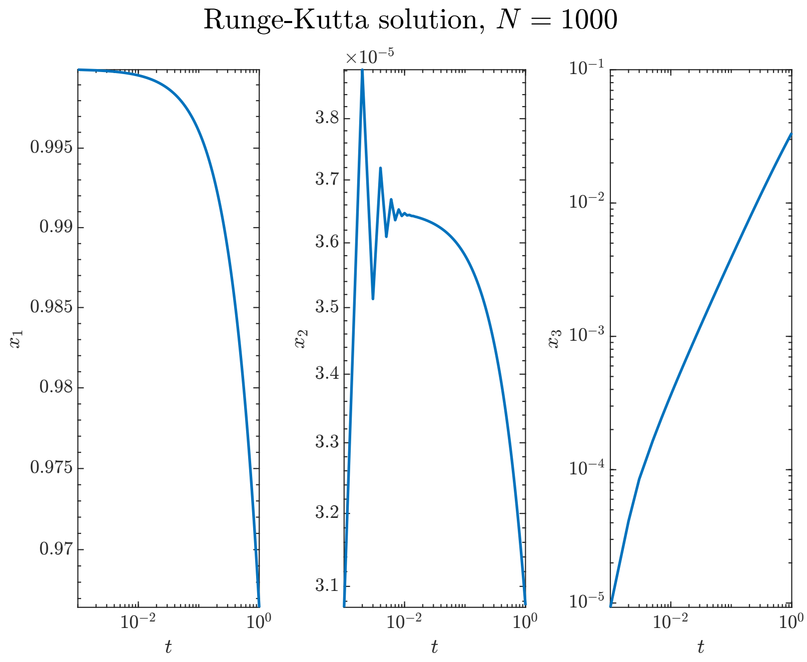


Figure 10: Solution by Runge-Kutta to the Robertson's problem in the interval $[0, 1]$. The number of steps was $N = 1000$, so $h = 0.001$. Shown are the concentrations x_1, x_2 and x_3 in a double logarithmic plot over time.

When comparing the number of steps it becomes visible that by `ode23` the step size stays around 870 for the relative tolerances 10^{-3}, 10^{-4} and 10^{-5} while for `ode23s` it increases significantly from 31 to 115 steps. Then for the much smaller rel. tolerance $1-^{-11}$ the number of steps increases quite a lot compared to the earlier results.

The behavior of the steps size is typical for a comparison of an explicit and implicit solver on a stiff system because for a stiff system the stability limit for the explicit method is quite small. That means that also for a lower accuracy the step size must be chosen small in order to get below the stability limit. We can guess that the stability limit for `ode23` is around $h = 1/867 \approx 0.0012$. For the implicit solver there is usually not a stability limit so as we saw for `ode23s` the step size scales with the desired accuracy.

Finally we also show the plots of the steps sizes for the accuracies 10^{-5} (left plot) and 10^{-7} (right plot). In figure 11 is is shown for the solver `ode23` which covered the time interval $[0, 1]$. You can see in both plots that the step size is the smallest around time $t = 0$ which is because there the rate of change is the largest. And as the solution evolves the step size gets larger but in the case 10^{-5} we see that it has an upper bound which is the stability limit.

For the solver `ode23s` the step size plots are shown in figure 12. Again the step size is smallest around $t = 0$ and then increases towards the end time of now $t = 1000$. In the rel. tolerance 10^{-5} case we see no upper boundary for the steps size which is because the solver being an implicit method has no stability limit.

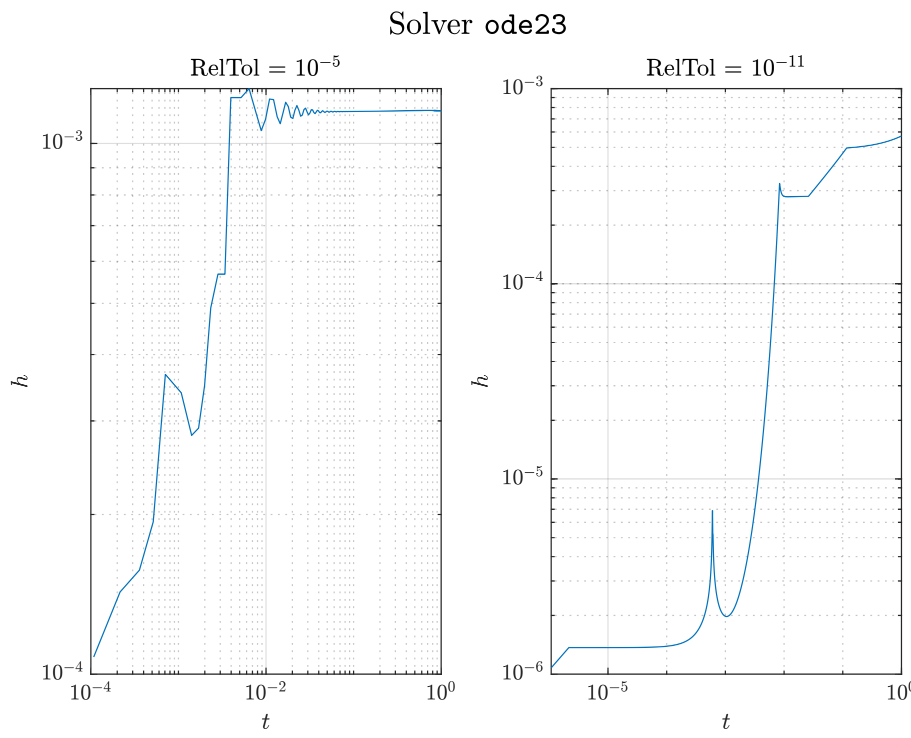


Figure 11: Plot of the step sizes used by the solver `ode23` in the interval $[0, 1]$. The relative tolerances were 10^{-5} (left) and 10^{-11} (right). The absolute tolerance was set to one thousands of the relative tolerance.

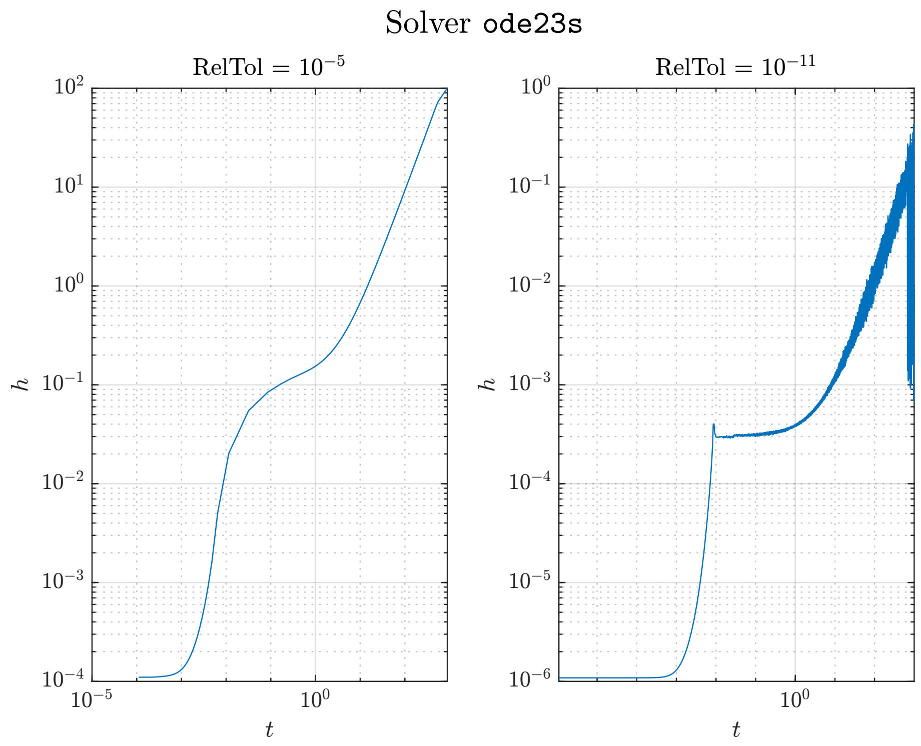


Figure 12: Plot of the step sizes used by the solver `ode23s` in the interval $[0, 1000]$. The relative tolerances were 10^{-5} (left) and 10^{-11} (right). The absolute tolerance was set to one thousands of the relative tolerance.