MSAS – Assignment #1: Simulation

Gloria Castronovo, 992130

1 Implicit equations

Exercise 1

Given the function $f(x) = \sin x + x - 1$, guess a and b such that $f(a)f(b) < 0$. Find the zero(s) of f in $[a, b]$ with 1) the bisection method, 2) the secant method, 3) the regula falsi method. All solutions must be calculated with 8-digit accuracy. Which method requires the least number of function evaluations? Report the computational time required by each method.

In order to solve this root-finding problem, three different numerical methods have been implemented: the Bisection method, the Secant method and the Regula Falsi method, and they all require a starting interval for which the product of the function evaluated at its extremes is negative.

Figure 1 shows the behaviour of the function f(x), which intersects the x axis once, in correspondence of the root, and the starting points with the respecting function evaluations.

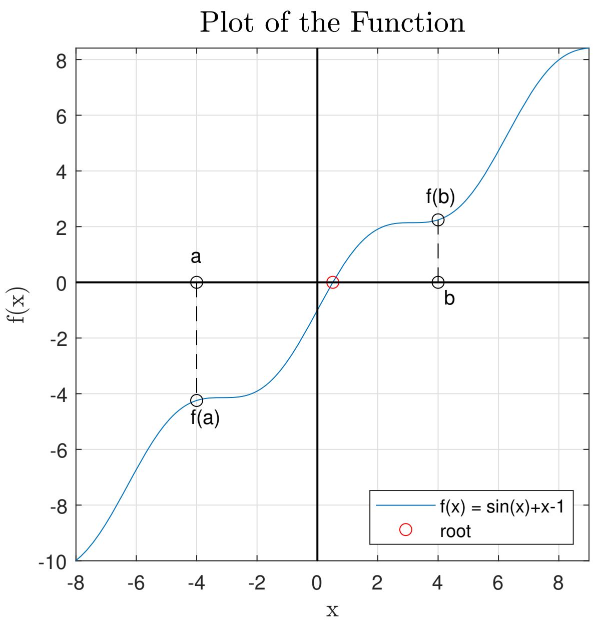


Figure 1: Function behaviour and root finding.

Since these are numerical methods, the root is obtained after a certain number of iterations, chosen according to a stopping criteria. This grants that the solution is close enough to the zero.

In particular, by setting a tolerance equal to 10^-9, the computation will stop as soon as the error is less than this number, and then the first 8 digits after the comma will be the same at these steps. In addition, the error has been computed at each step differently, since these methods work basing on different schemes, as Tab.1 shows, while Tab.2 summarizes all the results obtained.

Table 1: Error computation expressions.

Method	Error
Bisection	$\frac{\lvert \mathbf{x_{k+1}} - \mathbf{x_k} \rvert}{2}$
Secant	$\lvert \mathbf{x_{k+1}} - \mathbf{x_k} \rvert$
Regula Falsi	$\lvert \mathbf{x_{k+1}} - \mathbf{x_k} \rvert$

Table 2: Results obtained from the three methods using a toll $= 10^{-9}$.

Parameters/Methods	Bisection	Secant	Regula falsi
x_0	0.51097343	0.51097343	0.51097343
$N_{functionevaluation}$	90	16	42
CPU time [s]	$5.49x10^{-4}$	$6.58x10^{-5}$	$3.87x10^{-5}$

It's evident that the secant method converges faster to the solution with respect to the other two, especially compared to the bisection method, which requires an higher number of iterations, as well as function evaluations, before satisfying the minimum tolerance.

In addition, the CPU time has been computed as an average computational time.

Exercise 2

Let $\mathbf{f}$ be a two-dimensional vector-valued function $\mathbf{f}(\mathbf{x}) = (x_1^2 + x_2 - 5,\, x_2^2 - x_1)^\top$, where $\mathbf{x} = (x_1, x_2)^\top$. Find the zero(s) of $\mathbf{f}$ by using Newton's method with $\partial \mathbf{f}/\partial \mathbf{x}$ 1) computed analytically, and 2) estimated through finite differences. Which version is more accurate?

For this problem, since it can be considered as a system of equations, a vector of common solutions of the kind $\mathbf{x} = (x_1, x_2)$ is researched, and Fig.2 shows what it happens when the domain of solutions of the two functions, intersect.

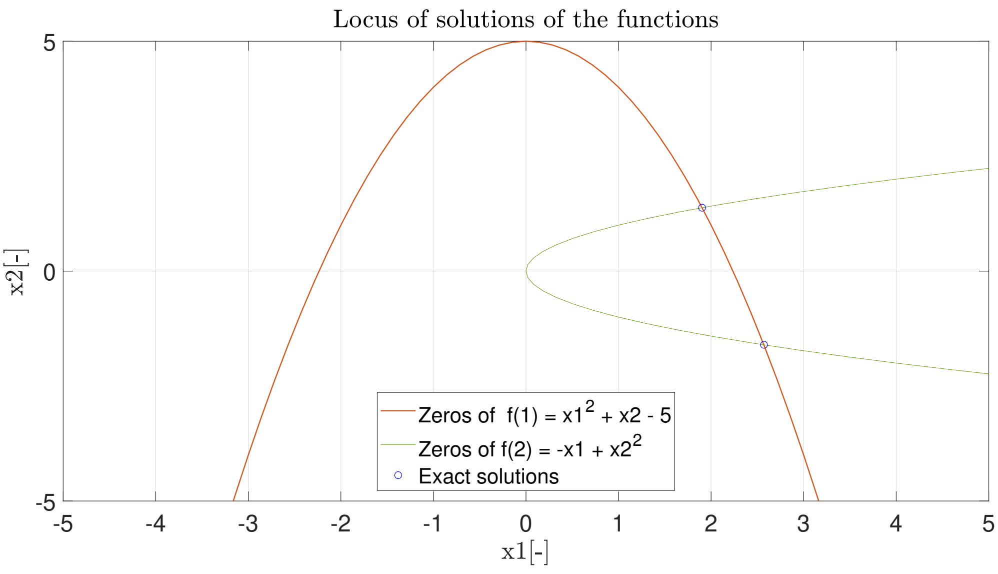


Figure 2: Locus of solutions of the functions.

Since the application of the Newton's method is required, its general mathematical expression is following.

$$\mathbf{x_{i+1}} = \mathbf{x_i} - \mathbf{J_i^{-1}}\mathbf{f}(\mathbf{x_i}) \tag{1}$$

It allows to compute the solution to the problem at each step from the knowledge of the previous step and the function evaluated at it, and the Jacobian matrix, which contains the partial derivatives of the function considered.

After giving a first initial guess of the solution to initialize the algorithm, which has been set to $\mathbf{x} = (2.5, 1.2)$, it's possible to start approaching the problem by computing the solution analytically, and the Jacobian matrix is obtained as follows

$$\mathbf{J} = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_2} \\ \\ \frac{\delta f_2}{\delta x_1} & \frac{\delta f_2}{\delta x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 & 1 \\ \\ -1 & 2x_2 \end{bmatrix} \tag{2}$$

The analytical solution has been computed after enough number of iterations such that the absolute error between the last and second to last x vector, was smaller than the tolerance, which a has been set to toll $= 10^{-6}$.

To estimate the solution through finite difference, the algorithm has been implemented in the same way as the analytical one but for this case, it's necessary to redefine the Jacobian matrices, that are formulated as

$$\mathbf{J_{forward}} = \begin{bmatrix} \frac{\mathbf{f}(x_1+\epsilon, x_2) - \mathbf{f}(x_1, x_2)}{\epsilon} & \frac{\mathbf{f}(x_1, x_2+\epsilon) - \mathbf{f}(x_1, x_2)}{\epsilon} \end{bmatrix} \tag{3}$$

$$\mathbf{J_{centered}} = \begin{bmatrix} \frac{\mathbf{f}(x_1, x_2) - \mathbf{f}(x_1-\epsilon, x_2)}{\epsilon} & \frac{\mathbf{f}(x_1, x_2) - \mathbf{f}(x_1, x_2-\epsilon)}{\epsilon} \end{bmatrix} \tag{4}$$

$$\mathbf{J_{backward}} = \begin{bmatrix} \frac{\mathbf{f}(x_1+\epsilon, x_2) - \mathbf{f}(x_1-\epsilon, x_2)}{2\epsilon} & \frac{\mathbf{f}(x_1, x_2+\epsilon) - \mathbf{f}(x_1, x_2-\epsilon)}{2\epsilon} \end{bmatrix} \tag{5}$$

Tab.3 shows the results obtained from the computations.

Table 3: Results obtained from the three methods using a toll $= 10^{-8}$.

Methods/Parameters	x	f(x)	CPU time [s]
Analytical	$\begin{pmatrix} 1.9027837 \\ 1.3794142 \end{pmatrix}$	$\begin{pmatrix} 0.910223x10^{-16} \\ 0 \end{pmatrix}$	1.88x10^{-4}
FORWARD	$\begin{pmatrix} 1.9027837 \\ 1.3794142 \end{pmatrix}$	$\begin{pmatrix} 1.110563x10^{-16} \\ 0 \end{pmatrix}$	$7.528x10^{-3}$
CENTRED	$\begin{pmatrix} 1.9027837 \\ 1.3794142 \end{pmatrix}$	$\begin{pmatrix} 4.4511561x10^{-16} \\ 0 \end{pmatrix}$	9.31x10^{-3}
BACKWARD	$\begin{pmatrix} 1.9027837 \\ 1.3794142 \end{pmatrix}$	$\begin{pmatrix} 5.1512151x10^{-16} \\ 0 \end{pmatrix}$	9.46x10^{-3}

It's possible to say that the analytical method is the most accurate one together with the Finite Difference Forward one and its CPU is a little bit lower than the CPU of the other three methods. The choice to use one Finite Difference method instead of the other is encouraged by the accuracy versus CPU ratio. If a fast computation is needed without having necessarily high accuracy, the best choice will be the Centered or Backward method. If the user attaches more importance to accuracy, he should opt for the Forward one.

2 Numerical solution of ODE

Exercise 3

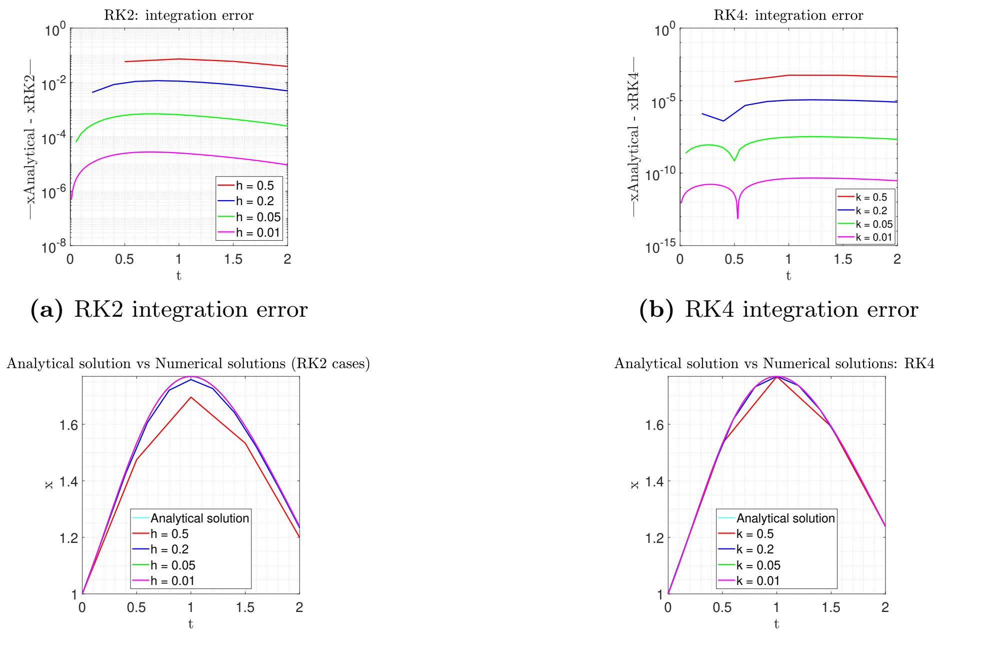
The Initial Value Problem $\dot{x} = -\frac{t^2-1}{t^2+1}x$, $x(0) = 1$, has analytic solution $x(t) = e^{2\arctan(t)-t}$. 1) Implement a general-purpose, fixed-step Heun's method (RK2); 2) Solve the IVP in $t \in [0, 2]$ for $h_1 = 0.5$, $h_2 = 0.2$, $h_3 = 0.05$, $h_4 = 0.01$ and compare the numerical vs the analytical solution; 3) Repeat points 1)–2) with RK4; 4) Trade off between CPU time & integration error.

In order to implement Heun's method (RK2) and RK4, it is required to define f(x, t) such that f(x, t) $= \dot{x} = -\frac{t^2-1}{t^2+1}x$. After this, the RK2 methods has been implemented with the explicit mid-point rule as shown in Eq.6, and similarly for the RK4 method, it's possible to obtain Eq.7

$$\begin{cases} x_p = x_k + \frac{1}{2}hf(x_k, t_k) \\ x_{k+1} = x_k + hf(x_p, t_k + \frac{1}{2}h) \end{cases} \tag{6}$$

$$\begin{cases} k_1 = f(x_k, t_k) \\ k_2 = f(x_k + \frac{h}{2}k_1, t_k\frac{1}{2}h) \\ k_3 = f(x_k + \frac{h}{2}k_2, t_k\frac{1}{2}h) \\ k_4 = f(x_k + hk_3, t_k + h) \\ x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \tag{7}$$

Then, for different step sizes h $= [0.5,0.3,0.05,0.01]$, it is plotted the absolute error $\epsilon_{\mathbf{abs}} = |\mathbf{x_{Analytical}} - \mathbf{x_{RK}}|$ for both propagation methods (Fig 3a and 3b), and the locus of solutions with the respect to the analytical one (Fig 3c and 3d).



(a) RK2 integration error

(b) RK4 integration error

(c) RK2 approx. of the solution

(d) RK4 approximation of the solution

Figure 3: Integration error and solutions for RK2 and RK4, at different time steps

The error is computed as the absolute value of the difference between the solution at the iteration considered, computed analytically, and the one at the same step computed numerically through the two methods. The graphs show that RK4 method, always gives a better approximation of the solution, at each time step. Regardless this, the computational time oscillates for both Runge Kutta computations, around the order of magnitude of 10^{-4}, so it's possible to say that within the same time interval, RK4 behaves definitely better in terms of accuracy than RK2 does.

Exercise 4

Let $\dot{\mathbf{x}} = A(\alpha)\mathbf{x}$ be a two-dimensional system with $A(\alpha) = [0, 1; -1, 2\cos\alpha]$. Notice that $A(\alpha)$ has a pair of complex conjugate eigenvalues on the unit circle; α denotes the angle from the Re$\{\lambda\}$-axis. 1) Write the operator $F_{\mathrm{RK2}}(h,\alpha)$ that maps $\mathbf{x}_k$ into $\mathbf{x}_{k+1}$, namely $\mathbf{x}_{k+1} = F_{\mathrm{RK2}}(h,\alpha)\,\mathbf{x}_k$. 2) With $\alpha = \pi$, solve the problem "Find $h \geq 0$ s.t. $\max(|\mathrm{eig}(F(h,\alpha))|) = 1$". 3) Repeat point 2) for $\alpha \in [0,\pi]$ and draw the solutions in the $(h\lambda)$-plane. 4) Repeat points 1)–3) with RK4 and represent the points $\{h_i\lambda\}$ of Exercise 3. What can you say?

The operators $F_{\mathrm{RK2}}(h,\alpha)$ and $F_{\mathrm{RK4}}(h,\alpha)$ that map $\mathbf{x}_k$ into $\mathbf{x}_{k+1}$ such that $\mathbf{x}_{k+1} = F_{\mathrm{RK2}}(h,\alpha)\,\mathbf{x}_k$, have been found respectively by pushing until the second and fourth order the Taylor Series Expansion, and they are shown in Eq.8 and Eq.9.

$$F_{RK2}(h,\alpha) = I + hA(\alpha) + \frac{h^2}{2}A(\alpha)^2 \tag{8}$$

$$F_{RK4}(h,\alpha) = I + hA(\alpha) + \frac{h^2}{2}A(\alpha)^2 + \frac{h^3}{6}A(\alpha)^3 + \frac{h^4}{4}A(\alpha)^4 \tag{9}$$

To solve the problem $\max(|\mathrm{eig}(F(h,\alpha))|) = 1$, it's necessary to make α values vary, within an interval which start at 0 and ends at π. In this way, the step size are computed for any angle contained in the interval.

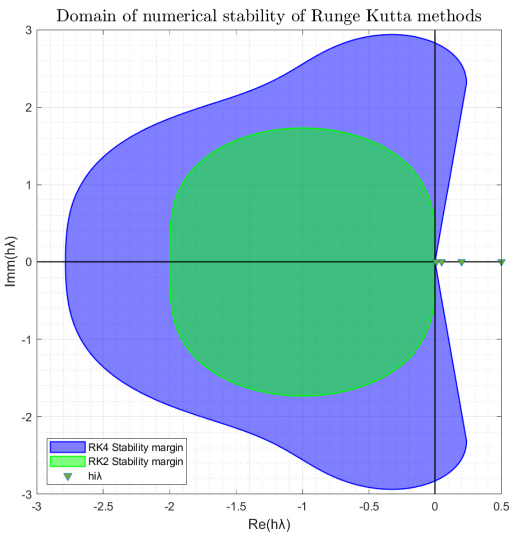


Figure 4: Domain of numerical stability of Runge Kutta methods.

Fig.4 shows the domain of stability for Runge Kutta 2 and 4, and their contours are found by multiplying the values of h found from the solution to the problem, and the eigenvalues of the A matrix, which since is alpha-dependent, it changes for each angle,and so do the eigenvalues . In addition to this domains, the scatters show the $\{h_i\lambda\}$ points representing the eigenvalues of the IVP of exercise 3 at time t = 0 mapped into the $\{h_i\lambda\}$ plane for the various step sizes h used in this specific exercise. Knowing that λ= 1, the time continuous system is unstable. As Runge Kutta is an explicit method, the domain of stability belongs to the coloured regions. In our case, the mapped eigenvalue remains in the unstable region of the plot whatever the step size is,because it is a purely real eigenvalue and then it remains on the right part of the x axis.

Exercise 5

Consider the IVP $\dot{\mathbf{x}} = A(\alpha)\mathbf{x}$, $\mathbf{x}(0) = [1, 1]^T$, to be integrated in $t \in [0, 1]$. 1) Take $\alpha \in [0, 2\pi]$ and solve the problem "Find $h \geq 0$ s.t. $\|\mathbf{x}_{\text{an}}(1) - \mathbf{x}_{\text{RK1}}(1)\|_\infty = \text{tol}$", where $\mathbf{x}_{\text{an}}(1)$ and $\mathbf{x}_{\text{RK1}}(1)$ are the analytical and the numerical solution (with RK1) at the final time, respectively, and $\text{tol} = \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. 2) Plot the five locus of solutions in the $(h\lambda)$-plane; plot also the function evaluations vs tol for $\alpha = \pi$. 3) Repeat points 1)–2) for RK2 and RK4.

To solve this exercise, it's necessary to evaluate the analytical solution of the IVP, considering the expression $x_{an} = x_0 e^{A(\alpha)t}$ and evaluating at the final time. By looking at its the espression, it's clearly dependent on the angle α, which is variable and which goes from 0 to 2π. Similarly to what has been done in the previous exercise, by imposing this condition inside a solver of system of non linear equations, it's possible to compute all the values of the step size with which the integrations are performed, as function of the angle and the tolerance, by starting from four initial guesses of h, since the tolerances to be considered are four. The choice of fixed values of h, comes from the fact that by looking at the theoretical graphs of the stability regions, the values chosen are close to the ones for which the graphs intersect the real axis. Moreover, with the aim to plot the regions of stability, the eigenvalues of A(α) are computed and progressively multiplied by the values of the time step found at each iteration. This last procedure described, has been executed keeping in mind that the locus of solutions are symmetrical with respect to the real axis, so it's easier to just multiply the h for just one of the two components of the eigenvalues vector found at each step, still considering it has a real and an imaginary part, and mirror it with respect to the real axis. The operators of $F_{\text{RK2}}(h, \alpha)$ and $F_{\text{RK4}}(h, \alpha)$ are presented in Eq.8 and Eq.9 of Exercise 4, while $F_{\text{RK1}}(h, \alpha)$ formal expression is recalled in Eq. 10, and derived by stopping at the first order the Taylor Series expansion

$$F_{RK1}(h, \alpha) = I + hA(\alpha) \tag{10}$$

Fig.5 and Fig.6a and Fig.6b show the stability domain of the Runge Kutta methods, obtained from this computation, at the four tolerances given, while Fig.6c represents the number of function evaluations needed when the tolerance increases.

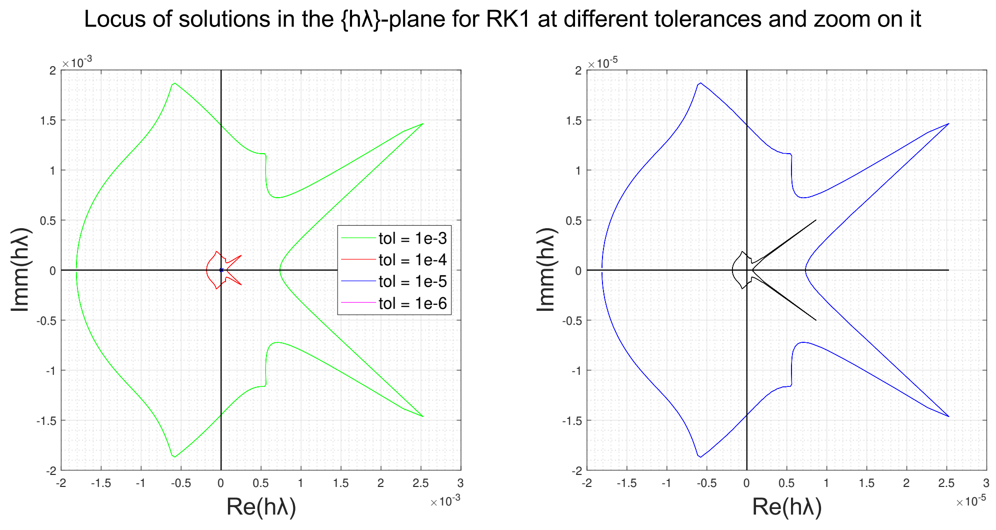
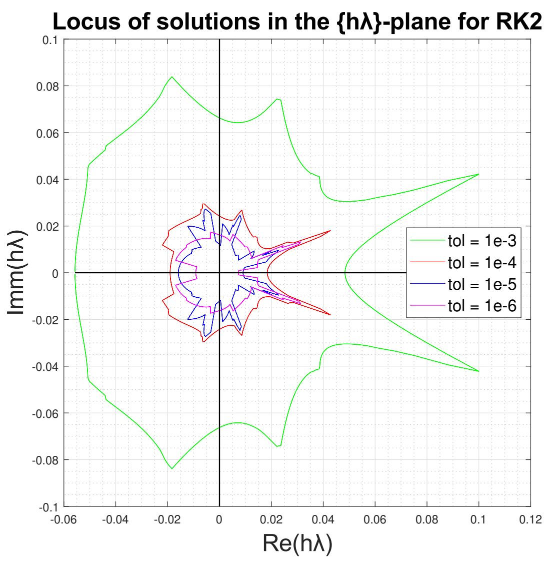
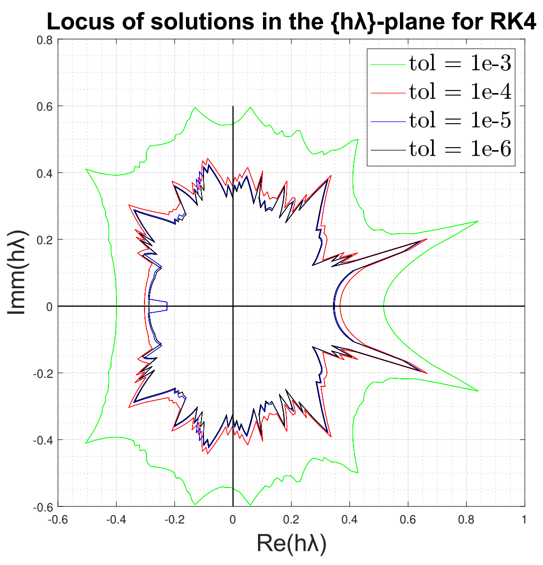


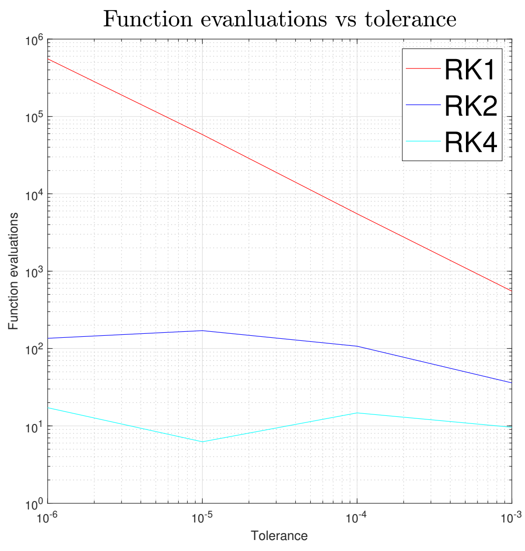
Figure 5: Locus of solutions for RK1.



(a) Locus of solutions for RK2.



(b) Locus of solutions for RK4.



(c) Function evaluations vs.tolerance .

Figure 6

Now by looking at the graphs, despite the region of numerical stability for RK1 come close to what was expected to see and same for RK2 and RK4 at the first tolerances, the remaining curves of the plot show that there is an error due to the fact that the first approach given to the problem, wasn't considering that, since the solution don't grant that the integration interval is perfectly subdivided by the time step h, the last step must be adapted to solve this trouble, such that the operator should have been instead$F_{RK1}(h,\alpha) = F_1 * F_2$with $F_1(h,\alpha) = I + rem\left(\frac{\Delta t}{h}\right) A(\alpha)$ and $F_1(h,\alpha) = I + rem\left(\frac{\Delta t}{h}\right) A(\alpha)$, and the same criterion should rule the successive orders of RK used.

The error is then propagated as shown in the graph of the function evaluations vs. the tolerance, where the alpha value was set to π. Although this error may have ruined the trend of these curves for RK2 and RK4, Fig.10 still shows that the number of function evaluations for RK1 is higher than RK2 and RK4, which as expected are more stable in terms of number of function evaluations and computational time, when the tolerance is changing. Lastly, is important to point out that the graphs shown are the ones computed with the first approach since, regardless of the effort to correct the code and considering the approaching deadline, the new graphs obtained were going further from the one expected, caused by new errors generated in the code, in the effort to change and correct the previous one.

Exercise 6

Consider the backinterpolation method $\mathrm{BI2}_{0.4}$. 1) Derive the expression of the linear operator $B_{\mathrm{BI2}_{0.4}}(h,\alpha)$ such that $\mathbf{x}_{k+1} = B_{\mathrm{BI2}_{0.4}}(h,\alpha)\mathbf{x}_k$. 2) Following the approach of point 3) in Exercise 5, draw the stability domain of $\mathrm{BI2}_{0.4}$ in the $(h\lambda)$-plane. 3) Derive the domain of numerical stability of $\mathrm{BI2}_\theta$ for the values of $\theta = [0.1,\ 0.3,\ 0.7,\ 0.9]$.

The expression of the linear operator $B_{\mathrm{BI2}_{0.4}}(h,\alpha)$ for a general case, is derived as follows, taking into account that a Backward Runge Kutta method of step size h is no more that a Frontward Runge Kutta of step size h, and it's presented as follows:

$$BI2_{0.4}(h,\alpha) = [I - (1-\theta)hA(\alpha) + \frac{((1-\theta)h)^2}{2}A^2(\alpha)]^{-1}[I + \theta hA(\alpha) + \frac{(\theta h)^2}{2}A^2(\alpha)] \qquad (11)$$

For different values of θ, the domain of stability of $B_{\mathrm{BI2}_{0.4}}$ is represented in Fig.7 for $\theta = 0.4$ and for $\theta = [0.1,\ 0.3,\ 0.7,\ 0.9]$ in Fig.8.

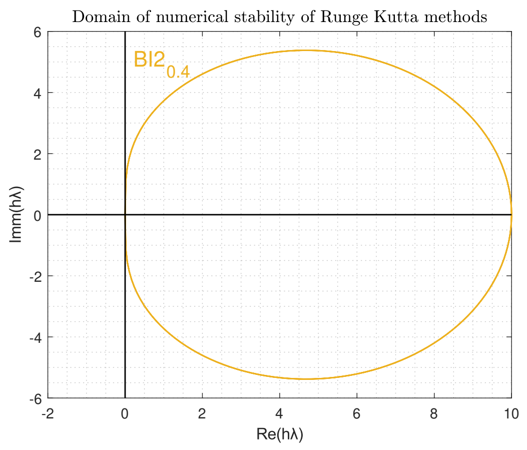


Figure 7: Domain of numerical stability of $BI2_\theta$ for $\theta = 0.4$.

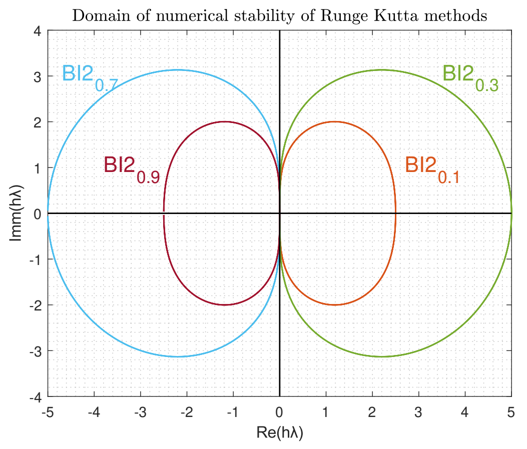


Figure 8: Domain of numerical stability of $BI2_\theta$ for $\theta = [0.1,\ 0.3,\ 0.7,\ 0.9]$.

As in Ex.4, the domain of numerical stability has been obtained for changing values of the step size h, and once computed the eigenvalues of the A(α) matrix of the operator, which has been considered again as $A(\alpha) = [0, 1; -1, 2\cos\alpha]$, the product between the step size and the eigenvalues for different angles, gives the points of the domain, i.e. the locus of the solution. It's now necessary to point out that now, in contrast to the Runge Kutta stability domain, here the regions internal to the boundaries of the domain, represent the region of instability, while the external ones the region of stability, which is clearly much wider. It's noteworthy that the domains are symmetric with respect to the imaginary axis.

Exercise 7

Consider the IVP $\dot{\mathbf{x}} = B\mathbf{x}$ with $B = [-180.5, 219.5; 179.5, -220.5]$ and $\mathbf{x}(0) = [1, 1]^T$ to be integrated in $t \in [0, 5]$. Notice that $\mathbf{x}(t) = e^{Bt}\mathbf{x}(0)$. 1) Solve the IVP using RK4 with $h = 0.1$; 2) Repeat point 1) using implicit extrapolation technique IEX4; 3) Compare the numerical results in points 1) and 2) against the analytic solution; 4) Compute the eigenvalues associated to the IVP and represent them on the $(h\lambda)$-plane both for RK4 and IEX4; 5) Discuss the results.

To solve this IVP problem, firstly the analytical solution is found within the time interval of [0,5] and the eigenvalues of the B matrix are computed. They result to be λ_1 = -1 and λ_2 = -400. This basically tells that this can be considered as an A-stiff system since $|\lambda_2|$ » $|\lambda_1|$. In this particular kind of systems it is recommended to use propagation methods whose stability domains cover the entire left part of the hλ-plane. If so, stable systems, which have eigenvalues with negative real part, will be mapped into stable domain. It is the case for IEX4 propagation method but not for RK4 method. In fact, since IEX4 is an A-stable method, it's solution is coherent with the analytical one, while a RK4, with h = 0.1, does not provide a mapping of λ_2 into stable region. That is why our method diverges. Fig.9 and Fig.10 show the behaviour of the solutions of the two methods in comparison with the analytical one, within the time interval given, while Fig.11 and Fig.12 the absolute error between the solutions.

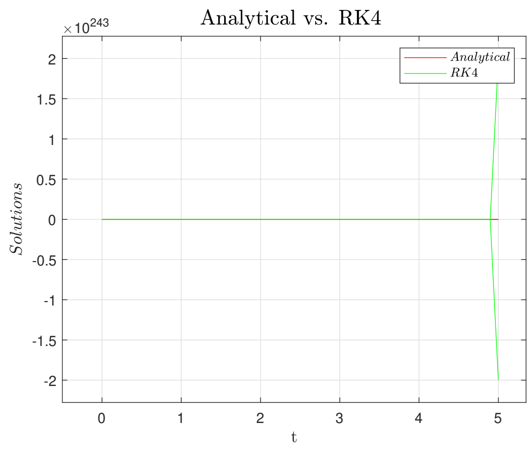


Figure 9: Analytical solution vs.RK4.

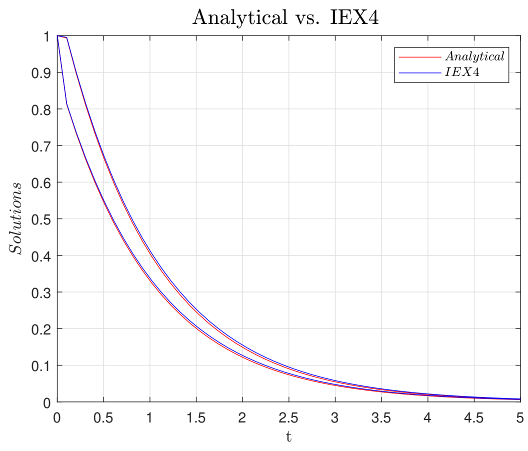


Figure 10: Analytical solution vs. IEX4.

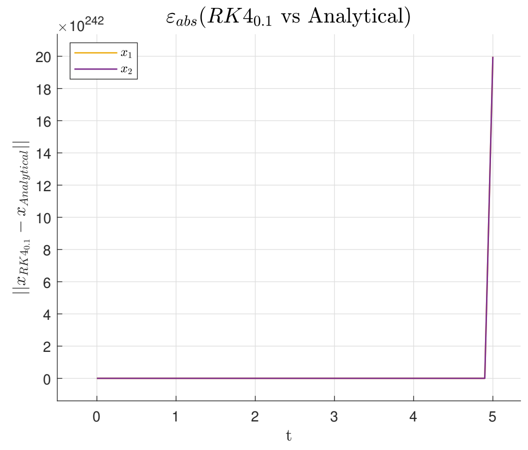


Figure 11: Absolute error between RK4 and analytical one.

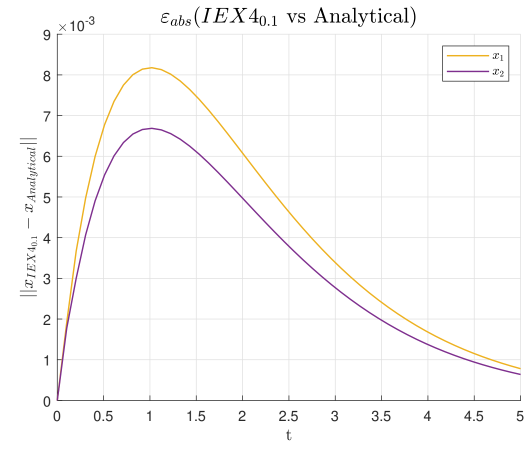


Figure 12: Absolute error between IEX4 and analytical one.

In Fig.13, it can be observed the fact that for RK4, λ_2 is not mapped into a stable region even though λ_1 is. On the opposite, for IEX4, it is observed that both eigenvalues don't belong to the unstable region which is the one highlighted in yellow.

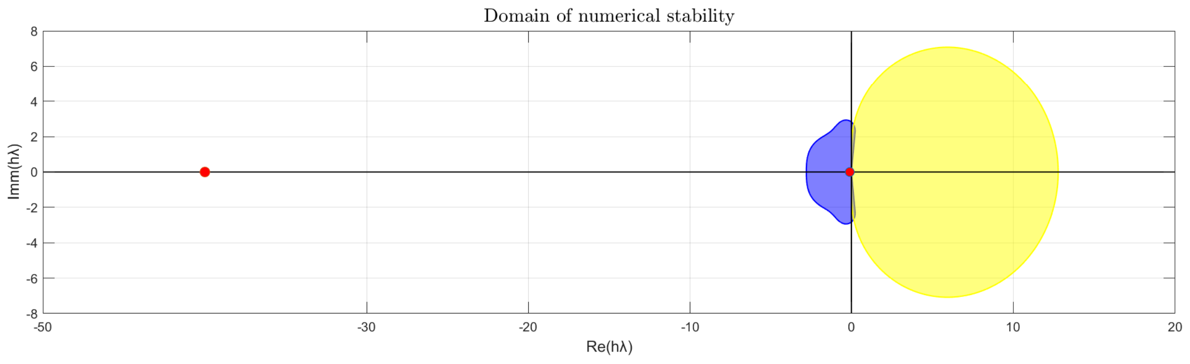


Figure 13: Domain of numerical stability and eigenvalues.

Again, the stability has been mapped by considering the matrix of the problem containing the same elements of the A matrix of the previous exercises, letting the α angle assume values starting from 0 and ending to π, so that for each angle considered, the eigenvalues computed are multiplied for the step size, so to build the stability domains from their product. The step size at each alpha is again computed in such a way to solve the problem $\max(|\text{eig}(F(h,\alpha))|) = 1$, where the F functional are evaluated.

Exercise 8

Consider the two-dimensional IVP

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}-\frac{5}{2}\left[1+8\sin(t)\right]x\\ (1-x)y+x\end{bmatrix}, \qquad \begin{bmatrix}x(t_0)\\ y(t_0)\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}$$

1) Solve the IVP using AB3 in $t \in [0,3]$ for $h = 0.1$; 2) Repeat point 1) using AM3, ABM3, and BDF3; 3) Discuss the results.

To solve this two dimensional IVP within a time interval equal to t = [0,3] for h = 0.1, it's possible to adopt different approaches, whether they are implicit or explicit, as the ones here listed, and implemented for this case.

$$\textbf{AB3}: \mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}) \tag{12}$$

$$\textbf{AM3}: \mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{12}(5\mathbf{f}_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1}) \tag{13}$$

AMB3:

$$\begin{cases}\textbf{AB}_{\textbf{predictorstage}}: x^p_{k+1} = \mathbf{x}_k + \frac{h}{2}(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2})\\ \textbf{AM}_{\textbf{predictorstage}}: x^c_{k+1} = \mathbf{x}_k + \frac{h}{12}(5\mathbf{f}_{x^p_{k+1},t_{k+1}} + 8\mathbf{f}_k - \mathbf{f}_{k-1})\end{cases} \tag{14}$$

$$\textbf{BDF3}: \mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1}\frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11}h\mathbf{f}_{k+1} \tag{15}$$

As it's possible to see from all of the four equations, we need to start the iteration from the third step, and this requires the evaluation of the solution to the problem at the first and second time step, since at t = 0, the x is already given, and they are evaluated by considering the runge kutta methods. Moreover, AM3 and BDF3 are implicit methods, so in order to know the solution at the current step, the function evaluation at the current step is needed, so by substituting the expression of the function, flipping it and expliciting x_{k+1}, y_{k+1} is found.

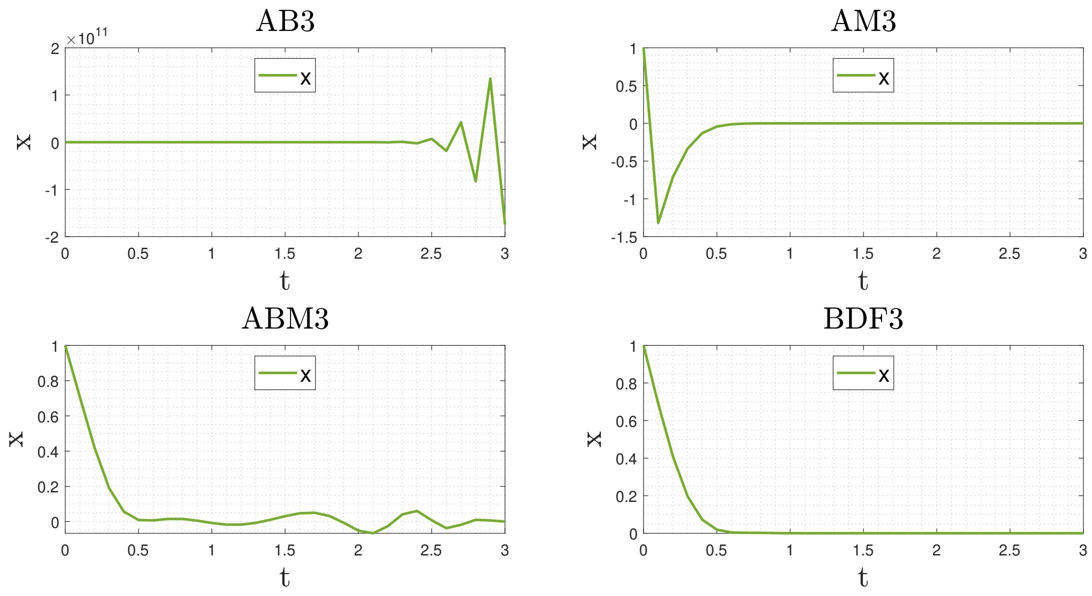


Figure 14: .x solutions

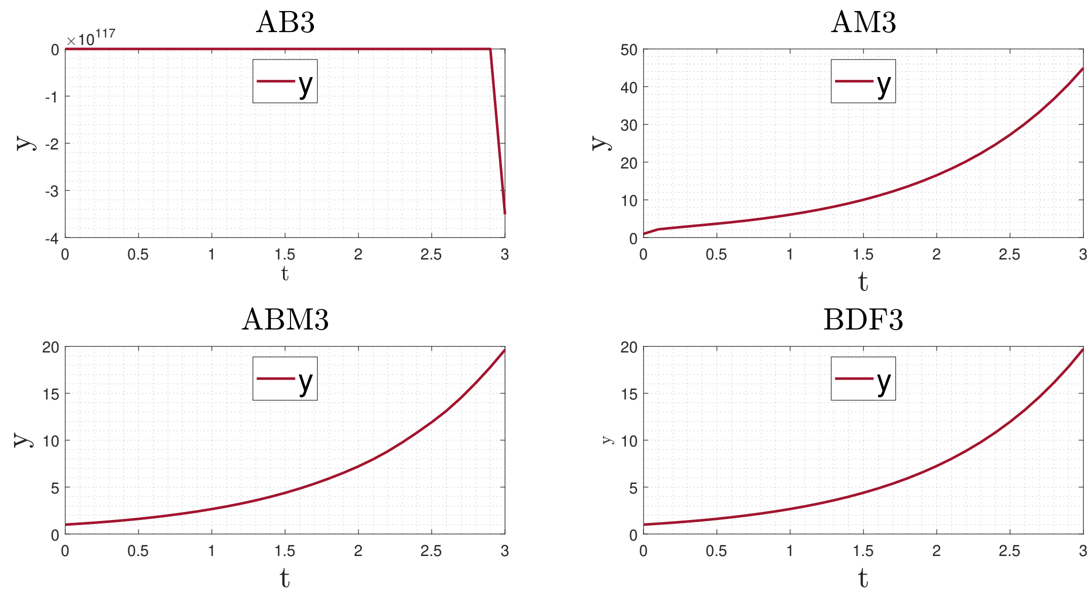


Figure 15: y solutions.

Having a look on the graphs of the solutions for x and y, it's possible to see that some solutions oscillate or don't converge, this is because the one eigenvalue of the first function for example, which is a time invariant one, doesn't fal in the regions of stability for these methods.