MSAS – Assignment #1: Simulation

Davide Lanza, 995599

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.

Function analysis

Before starting the resolution of the zero finding problem, the function has been plotted (figure 1) in order to choose an interval in which the function is continuous and $f(a)f(b) < 0$, so that the presence of a root is ensured; in particular $[-5, 5]$ has been considered the starting interval. From the plot it can be noticed that the function has only one root.

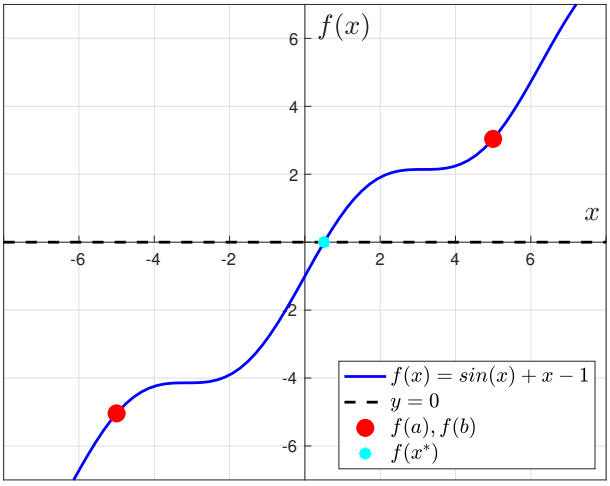


Figure 1: Plot of the function

Stopping criteria

The three methods have two stopping criteria: residual and error. The residual criterion is $|f(x^*| < Tol$ where x^* is the root found at each iteration and Tol the tolerance; the following criterion assures the result to be close enough to $f(x) = 0$. The error, strictly related to the accuracy, is computed as $|x_{k+1} - x_k| < 10^{-8}$: it guarantees 8-digit accuracy, as requested. It's important to remark the fact that the double criteria has been chosen in order to have both accurate and precise results. In the bisection method one theoretical relation can be used to estimate the minimum number of iterations needed to have the prescribed error ϵ: $k_{min} = \log_2(\frac{b-a}{\epsilon}) - 1$

Results and comparison

In figure 2 are compared the errors at each iteration of the three methods. The semi logarithmic scale has been used to emphasize the convergences of the methods: in particular can be noticed the linear convergence behaviour of the bisection method with respect to the super linear

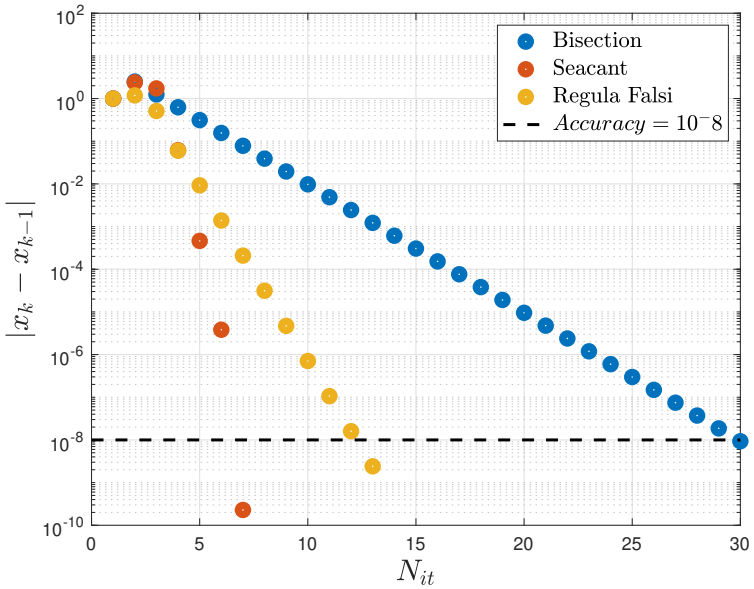


Figure 2: Errors comparison

one of the others. In all the methods two starting guesses are requested to solve the problem. Computational times are of the same order of magnitude. One last consideration is the number of function evaluations: in all the cases one evaluation per step is needed, plus an additional evaluation in the first step. Therefore the number of function evaluation is equal to $N_{it} + 1$. In the table below 1 are reported the numerical results.

Method	root	N_{it}	Computational time $[s]$
Bisection	0.51097344	29	0.00574679
Secant	0.51097343	6	0.00329508
Regula Falsi	0.51097343	12	0.00500604

Table 1: Results

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?

Function analysis

Before doing any further reasoning, it is important to graphically visualize the function, in order to understand the trend and how many roots it presents. In this case the function is two dimensional and multi-variable; it has two zeros as shown in figure 3. Now that the position of the zeros is clear it is possible to select a right initial guess to start the newton method.

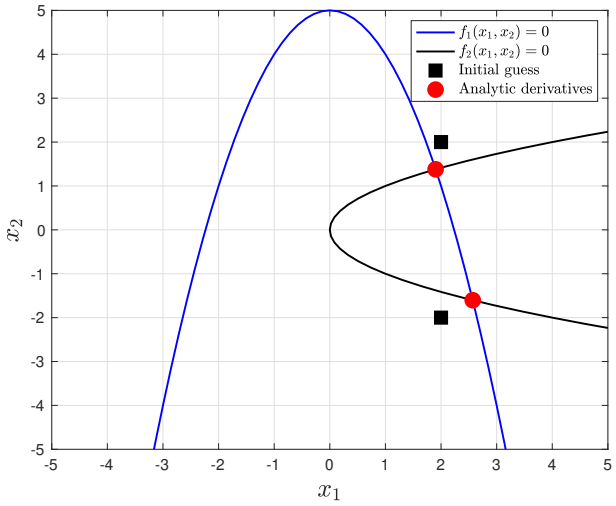


Figure 3: Function

Newton method and stopping criterion

Unlike the previous root-finding methods, Newton's method uses the derivatives of the function to acquire more information. Basically, the zero of the function is calculated by locally substituting its tangent line. In this particular case the function is two-dimensional, therefore the solution has been set up like this:

$$J(\mathbf{x}^{(k)})\delta\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)}) \tag{1}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)} \tag{2}$$

J is the Jacobian of the function. At each step it is requested to solve a linear system problem.
Matlab note: $\delta\mathbf{x}^{(k)} = -J\backslash\mathbf{f}$. The $\backslash$Matlab command has been used for efficiency issues.

The Stopping criterion used is the difference between two successive iterates. In particular: $||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|| < tol$. This criterion was chosen because the zeros are simple. Otherwise, one could have used the one from the residual: $||\mathbf{f}(\mathbf{x}^{(k)}|| < tol$.

Jacobian matrix

The Jacobian has been computed both analytically and through finite differences.

$$J_{an} = \begin{pmatrix} 2x_1 & 1 \\ -1 & 2x_2 \end{pmatrix} \tag{3}$$

The approximation through forward finite differences results in:

$$J_{approx} = \left(\frac{\mathbf{f}(\mathbf{x} + \epsilon_\mathbf{x}) - \mathbf{f}(\mathbf{x})}{||\epsilon_x||}, \frac{\mathbf{f}(\mathbf{x} + \epsilon_\mathbf{y}) - \mathbf{f}(\mathbf{x})}{||\epsilon_y||} \right) \tag{4}$$

The approximation through centered finite differences results in:

$$J_{approx} = \left(\frac{\mathbf{f}(\mathbf{x} + \epsilon_\mathbf{x}) - \mathbf{f}(\mathbf{x} - \epsilon_\mathbf{x})}{||2\epsilon_x||}, \frac{\mathbf{f}(\mathbf{x} + \epsilon_\mathbf{y}) - \mathbf{f}(\mathbf{x} - \epsilon_\mathbf{y})}{||2\epsilon_y||} \right) \tag{5}$$

where $\epsilon_\mathbf{x} = (\epsilon_x; 0)$ and $\epsilon_\mathbf{y} = (0; \epsilon_x)$. The size of the perturbation ϵ is:

$$\epsilon = \begin{cases} |x|\sqrt{eps}, & if|x| \geq 1 \\ \sqrt{eps}, & if|x| < 1 \end{cases} \tag{6}$$

Results

In table 2 are reported the solutions using the analytic Jacobian. Regarding the approximated solutions, the results are very close to the analytical one. The error reaches a very low asymptotic value in 4 iterations.

Initial guess	Root
(2; 2)	(1.90278368; 1.37941425)
(2; -2)	(2.56963170; -1.60300708)

Table 2: Analytical results

It can be observed from the results in figure 4 (it has just been compared analytic vs forward but with centred solution the trend is the same) that there is practically no difference in accuracy between the approximate and analytical methods, for the perturbation that has been chosen. This is due to the fact that the Jacobian results a two dimensional linear function with a low slope and so the finite differences approximate very well the variations of the function.

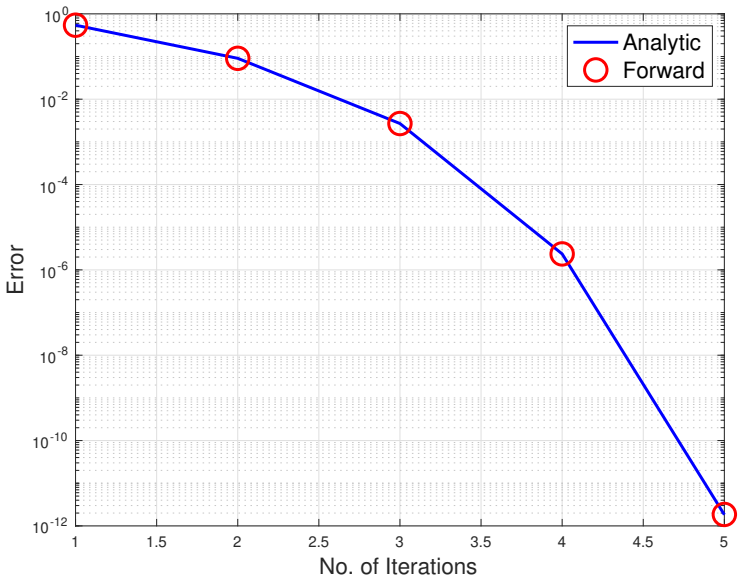


Figure 4: Analytic vs Forward

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.

Methods implementation

This section briefly reports the implementation of the two methods given the initial value problem and a discretized time vector: $\begin{cases} \dot{x} = f(x,t) \\ x(0) = 1 \end{cases}$

The Heun's method (RK2):

$$\begin{cases} x^p_{k+1} = x_k + hf(x_k, t_k) \\ x^c_{k+1} = x_k + \frac{h}{2}[f(x_k, t_k) + f(x^p_{k+1}, t_{k+1})] \end{cases} \tag{7}$$

The 4^{th} order Runge-Kutta method (RK4):

$$\begin{cases} k_1 = f(x_k, t_k) \\ k_2 = f(x_k + \frac{h}{2}k_1, t_k + \frac{h}{2}) \\ k_3 = f(x_k + \frac{h}{2}k_2, t_k + \frac{h}{2}) \\ k_4 = f(x_k + hk_3, t_k + h) \\ x_{k+1} = x_k + \frac{h}{2}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \tag{8}$$

Mapping control

Before integrating the initial value problem it's important to check if the analytic solution can be mapped properly with the considered method. To do so, the eigenvalue of the scalar problem has been studied considering that is a time-variant eigenvalue and is always a real number:

1. if t $\in [0, 1]$, $\lambda(t) > 0$ and so the unstable eigenvalue is always mapped as unstable in the λh right semi plane.
2. if t $\in [1, 2]$, $\lambda(t) < 0$ and the stable eigenvalue is mapped as stable in the left semi plane only if $h\lambda(t) > -2.8$ or -2 (RK4 and RK2 margin).

In our problem $h_{max}\lambda(t = 2) = -0.3$ and so we are inside the stability regions of both RK2 and RK4 for each negative real eigenvalue.

Results and comparison

The approximated solutions have been compared with the analytic solution through the error defined as $|x_{analytic} - x_{RK}|$ at each time step t_h. In figure 5 and 6 are shown the solutions for four different time steps. It can be noticed by the figures that the RK4 method approximates better the exact solution for every t_h used with respect to the other method. This was to be expected, since the number of function evaluations requested by RK4 method is four per step compared to the two requested by the Heun method.

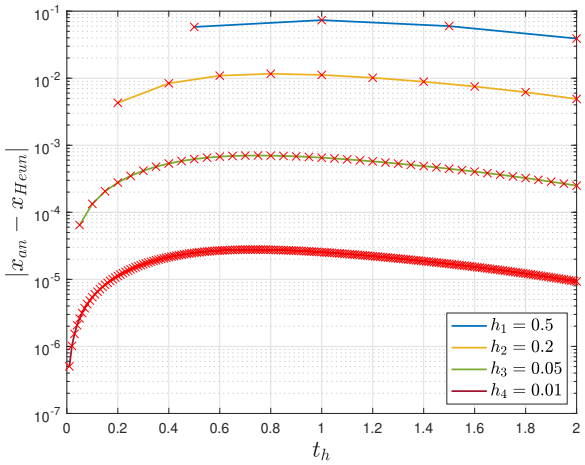


Figure 5: RK2 integration error

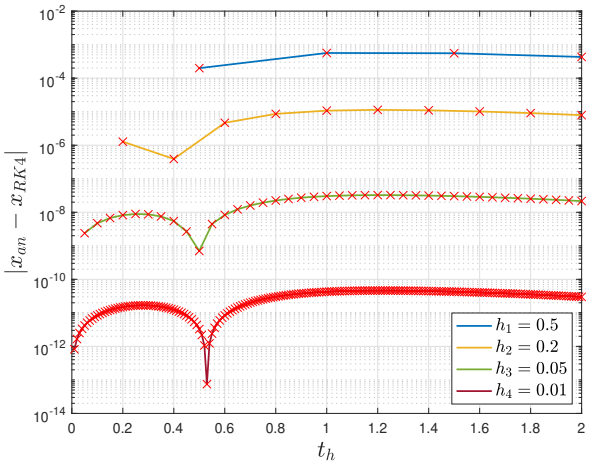
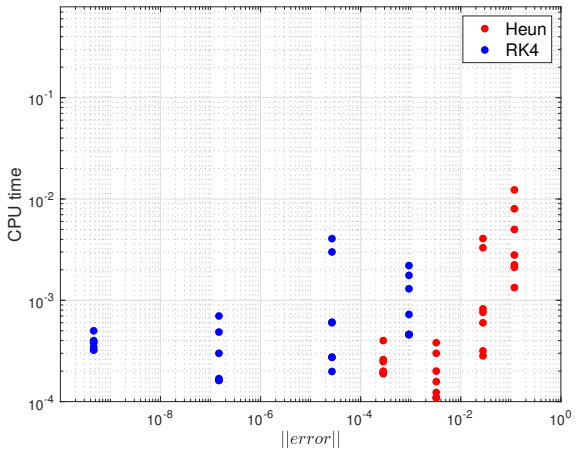


Figure 6: RK4 integration error

Computational time vs integration error



An additional analysis has been done by comparing the computational times of the two methods under consideration. In order to have more consistent and appreciable results, the simulation has been repeated several times. The figure on the left shows that the computational time does not systematically differ significantly for the two integration methods whereas the accuracy of the solution is completely different, even several orders of magnitude.

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 $F_{\text{RK2}}(h, \alpha)$ that maps $\mathbf{x}_k$ into $\mathbf{x}_{k+1}$, namely $\mathbf{x}_{k+1} = F_{\text{RK2}}(h, \alpha)\,\mathbf{x}_k$. 2)With $\alpha = \pi$, solve the problem "Find $h \geq 0$ s.t.max $(|\text{eig}(F(h, \alpha))|) = 1$". 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?

F-operators

Stating the dynamic problem as $\dot{\mathbf{x}} = A(\alpha)\mathbf{x}$ the F-operators, which maps $\mathbf{x_k}$ into $\mathbf{x_{k+1}}$, are:

$$F_{RK2}(h, \alpha) = I^{(2)} + hA(\alpha) + \frac{(A(\alpha)h)^2}{2} \tag{9}$$

$$F_{RK4}(h, \alpha) = I^{(2)} + hA(\alpha) + \frac{(A(\alpha)h)^2}{2} + \frac{(A(\alpha)h)^3}{3!} + \frac{(A(\alpha)h)^4}{4!} \tag{10}$$

Functional analysis

The evaluation of the requested solution has been made through the fsolve Matlab function, which requires a guess to be initialized. This section explains the procedure followed to determine the initial guess h_{guess}.

The problem is generically stated as: "Find h ≥ 0 s.t. $max(|eig(F(h, \alpha)|) = 1$ ". The idea is to plot the functional of the problem defined as:

$$g(h) = max(|eig(F(h, \alpha)|) - 1$$

and find graphically h such that $g(h) = 0$. Solutions will often be more than one: unfeasible ones will be discarded ($h < 0$), the others will be given as input depending on the α angle.

Let's take an example imposing $\alpha = \pi$. Figure 7 shows the functional solutions using RK2 method. As expected from the theory a proper initial guess is 2.

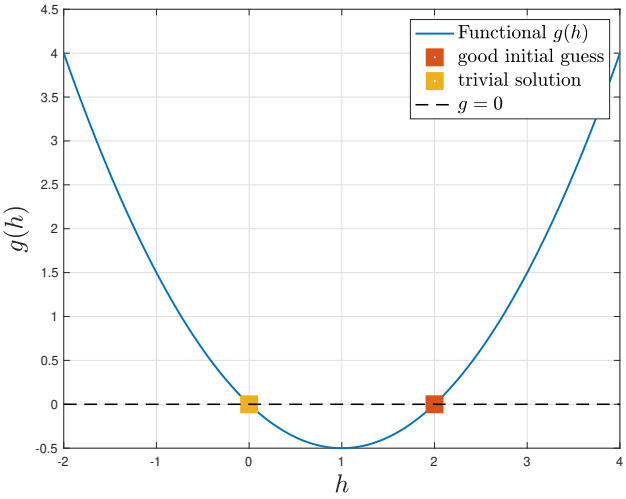


Figure 7: RK2: Functional for $\alpha = \pi$

A further step can be done considering now an interval of α angles varying from 0 to π. There will be a set of curves as shown in figure 8 and 9 with different zeros. For the sake of clarity a smaller number of curves have been plotted. Rectangles in the graphs highlight the range in which good value of h_{guess} can be selected. An additional confirmation comes from theory which demonstrates that the values are around 2 for RK2 and around 2.8 for RK4.

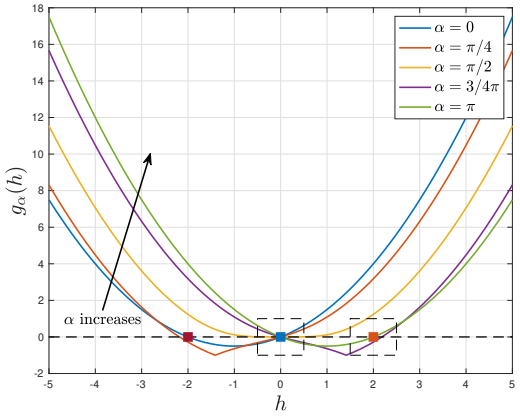


Figure 8: RK2: Functional for $\alpha \in [0, \pi]$

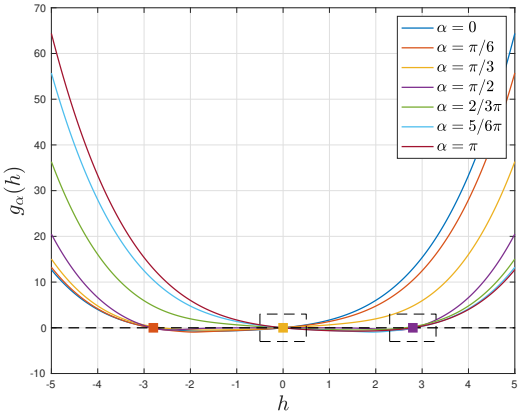


Figure 9: RK4: Functional for $\alpha \in [0, \pi]$

Stability regions

Once all the initial guesses have been determined they can be used in the fsolver to retrieve the solution of the problem. The results can be drawn in the $(h\lambda)$-plane to obtain the stability domain of the integration methods as shown in figure 10. RK4 stability domain is bigger with respect to RK2 but it has to be remarked the fact that some unstable eigenvalues are mapped in stables ones, therefore the stepsize choice has to be done carefully.

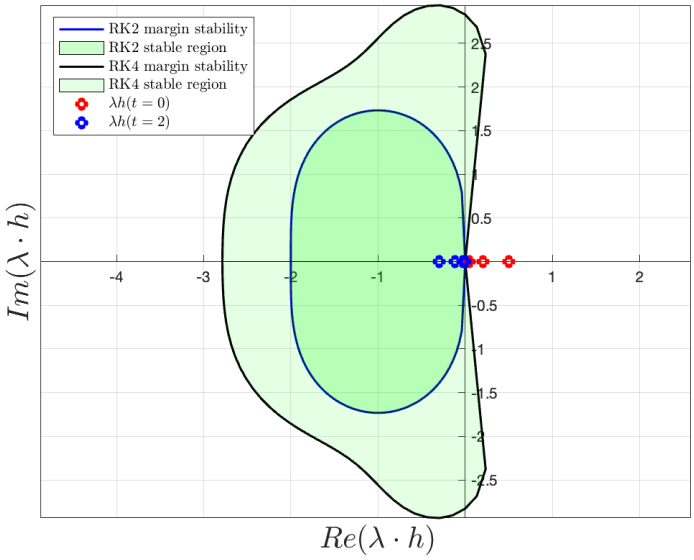


Figure 10: RK2 and RK4 stability regions

Eigenvalues representation

In figure 10 are represented red and blue points which are $(h_i\lambda)$ from exercise 3 at initial and final time. As already said in exercise 3, all the eigenvalues are mapped correctly: in particular at t = 0, the IVP has $\lambda = 1$ (unstable) and in fact the mapped value is $h_i\lambda > 0$ for all the prescribed h (unstable). At $t = 2$, the IVP has $\lambda = -0.6$ (stable) and all the mapped values are within RK2 and RK2 stability regions.

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.

F-operators

This accuracy problem has been solved by using the F-operators that map $\mathbf{x_k}$ into $\mathbf{x_{k+1}}$. The biggest issue was to ensure that the last step of the integration interval exactly matched the final time. In order to do that the operator has been decomposed in two contributes. In particular below are reported all the three operator. For RK1:

$$
\begin{aligned}
F_{RK1_a}(h,\alpha) &= I^{(2)} + rem\left(\frac{\Delta t}{h}\right) A \\
F_{RK1_b}(h,\alpha) &= I^{(2)} + hA \\
\Rightarrow F_{RK1}(h,\alpha) &= F_{RK1_a} F_{RK1_b}^n
\end{aligned} \tag{11}
$$

where $n = floor\left(\frac{\Delta t}{h}\right)$.

The same method has been applied to RK2:

$$
\begin{aligned}
F_{RK2_a}(h,\alpha) &= I^{(2)} + rem\left(\frac{\Delta t}{h}\right) A + \frac{(rem(\Delta t/h)A)^2}{2} \\
F_{RK2_b}(h,\alpha) &= I^{(2)} + hA + \frac{(Ah)^2}{2} \\
\Rightarrow F_{RK2}(h,\alpha) &= F_{RK2_a} F_{RK2_b}^n
\end{aligned} \tag{12}
$$

And lastly, for RK4:

$$
\begin{aligned}
F_{RK4_a}(h,\alpha) &= I^{(2)} + rem\left(\frac{\Delta t}{h}\right) A + \frac{(rem(\Delta t/h)A)^2}{2} + \frac{(rem(\Delta t/h)A)^3}{3!} + \frac{(rem(\Delta t/h)A)^4}{4!} \\
F_{RK4_b}(h,\alpha) &= I^{(2)} + hA + \frac{(Ah)^2}{2} + \frac{(Ah)^3}{3!} + \frac{(Ah)^4}{4!} \\
\Rightarrow F_{RK4}(h,\alpha) &= F_{RK4_a} F_{RK4_b}^n
\end{aligned} \tag{13}
$$

Note:

- Floor() rounds the elements to the nearest integers towards minus infinity.
- Rem() returns the remainder after division.

Solutions

At this point the problem can be solved using the operator to map from the initial condition to the iteration at the final time, as requested by the problem. By varying α from 0 to 2π and varying the tolerance, the results are shown in figure 11, 12, 13.

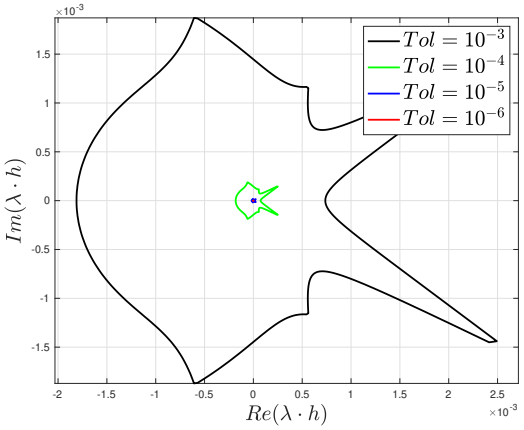


Figure 11: RK1

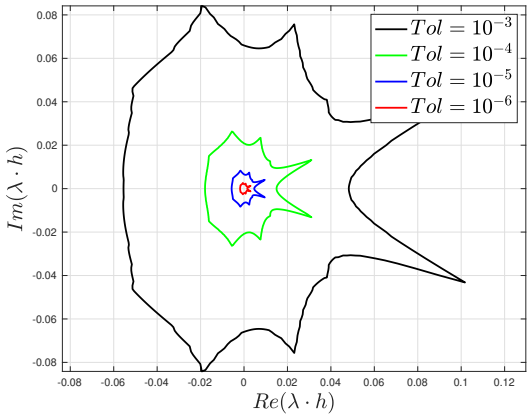


Figure 12: RK2

These three charts represents the five locus of solutions in the $(h\lambda)$-plane. It can be noticed that the accuracy regions get smaller as the tolerance decreases. That makes sense since smaller time step must be used to meet tighter tolerances. Moreover, as expected, increasing the order of the integration method the accuracy region becomes bigger being equal the tolerance.

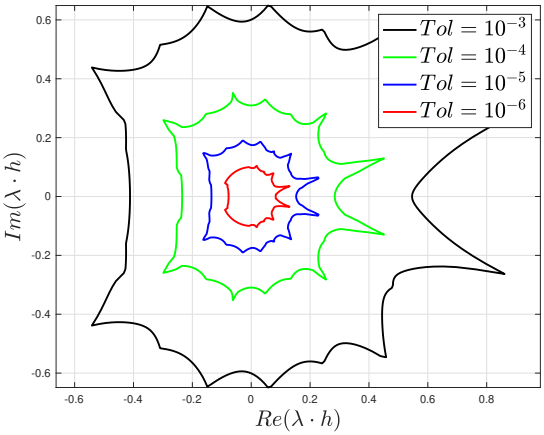


Figure 13: RK4

Function evaluations

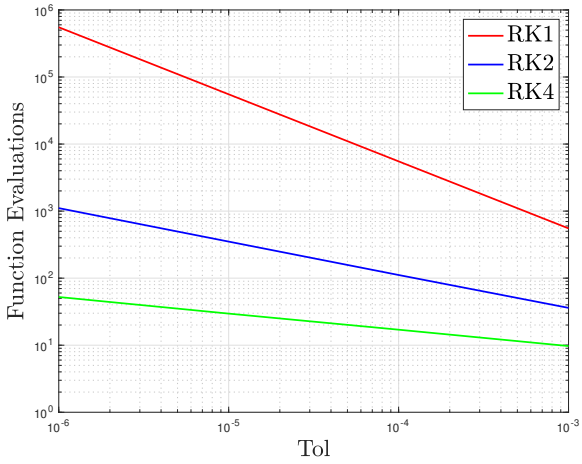


Figure 14: No. of function evaluations: $\alpha = \pi$

In figure 14 has been plotted the number of function evaluations with respect to the tolerance, which equals the number of stages of the method multiplied by the number of steps performed during the simulation as a function of the achieved accuracy. It turns out that RK4 was cheaper than RK2 and RK1 for all the tolerance values. By cutting the step size in half, we double the number of function evaluations needed to complete the simulation run. For the same ''price" we could have kept the step size the same and instead doubled the accuracy order. Thereby we would have gained two orders of magnitude in improved accuracy as opposed to only one order by reducing the step size.

Exercise 6

Consider the backinterpolation method $\text{BI2}_{0.4}$. 1) Derive the expression of the linear operator $B_{\text{BI2}_{0.4}}(h, \alpha)$ such that $\mathbf{x}_{k+1} = B_{\text{BI2}_{0.4}}(h, \alpha)\mathbf{x}_k$. 2) Following the approach of point 3) in Exercise 5, draw the stability domain of $\text{BI2}_{0.4}$ in the $(h\lambda)$-plane. 3) Derive the domain of numerical stability of BI2_θ for the values of $\theta = [0.1, 0.3, 0.7, 0.9]$.

Operator derivation

This section presents the derivation of B_{BI2_θ} operator that maps $\mathbf{x_k}$ into $\mathbf{x_{k+1}}$. Let's consider a linear time invariant system: $\dot{\mathbf{x}} = A\mathbf{x}$.

Basically it's a two stage procedure: in the first stage it's taken a θh step with an explicit 2^{nd} order method, in the second stage it's taken the remaining $(1-\theta)h$ step with an implicit 2^{nd} order method.

$$\begin{cases} 1^{st} stage : \mathbf{x}^l_{k+\theta} = \mathbf{x}_k + \theta h A \mathbf{x}_k + \frac{1}{2}(\theta h A)^2 \mathbf{x}_k \\ 2^{nd} stage : \mathbf{x}_{k+1} = \mathbf{x}^l_{k+\theta} + (1-\theta) h A \mathbf{x}_{k+1} + \frac{1}{2}((1-\theta)hA)^2 \mathbf{x}_{k+1} \end{cases}$$

The system can be rearranged as follow:

$$\begin{cases} 1^{st} stage : \mathbf{x}^l_{k+\theta} = (I^{(2)} + \theta h A + \frac{1}{2}(\theta h A)^2)\mathbf{x}_k \\ 2^{nd} stage : \mathbf{x}_{k+1} = (I^{(2)} - (1-\theta) h A - \frac{1}{2}((1-\theta)hA)^2)^{-1} \mathbf{x}^l_{k+\theta} \end{cases}$$

By substituting the first equation into the second, we obtain the mapping operator:

$$B_{BI2_\theta}(h, \alpha) = (I^{(2)} - (1-\theta)hA - \frac{1}{2}((1-\theta)hA)^2)^{-1}(I^{(2)} + \theta hA + \frac{1}{2}(\theta hA)^2) \tag{14}$$

Substituting in this last expression the value $\theta = 0.4$ gives the operator you were looking for.

Stability domains

To determine the stability domain, we proceed as we have already done in exercise 4. Figure 15 shows the initial guess to be given to the fsolve Matlab function. Figure 16 represents the region of stability of the method in the $(h\lambda)$-plane (Note: The red area is the unstable part). As expected this method results in very a nice stability domain with a large unstable region in the right half plane.

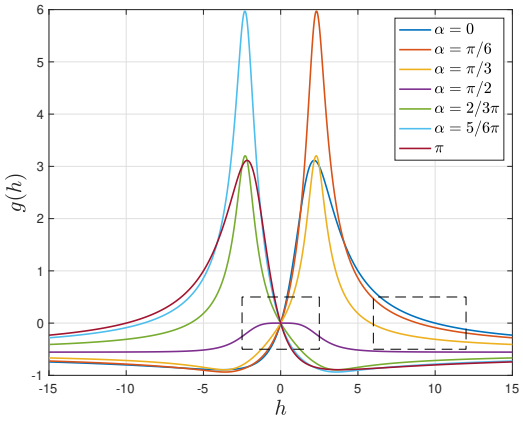


Figure 15: $BI2_{0.4}$: Functional

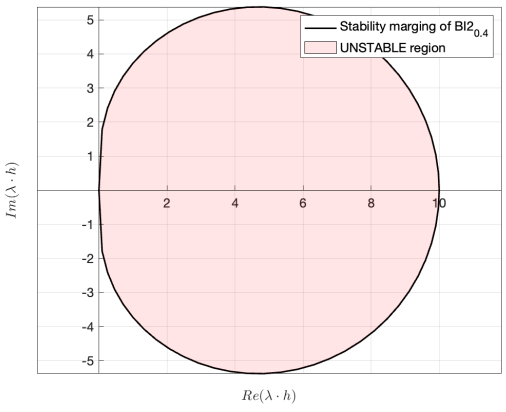
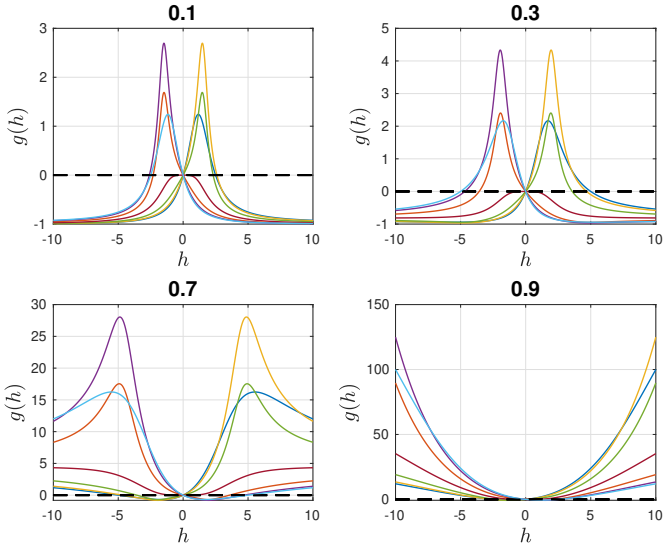


Figure 16: $BI2_{0.4}$ stability region



Let's now extend the problem to different value of θ. As always it's important to have a valid guess to be given to the Matlab solver. In the chart on the left are reported the functional plots for the given $\theta = [0.1, 0.3, 0.7, 0.9]$.

Figure 17: $BI2_\theta$: Functional

From figure 18 it can be noticed that the stability domain changes drastically depending on θ. The graph shows what we expected: for values of $\theta > 0.5$ the stability domains turn over on the other side of the semi plane. In fact that value of θ splits the regions that come from BE and FE. Backward Runge-Kutta methods are special cases of this new class with $\theta = 0$, and the explicit Runge-Kutta algorithms are special cases of this class $\theta = 1$.

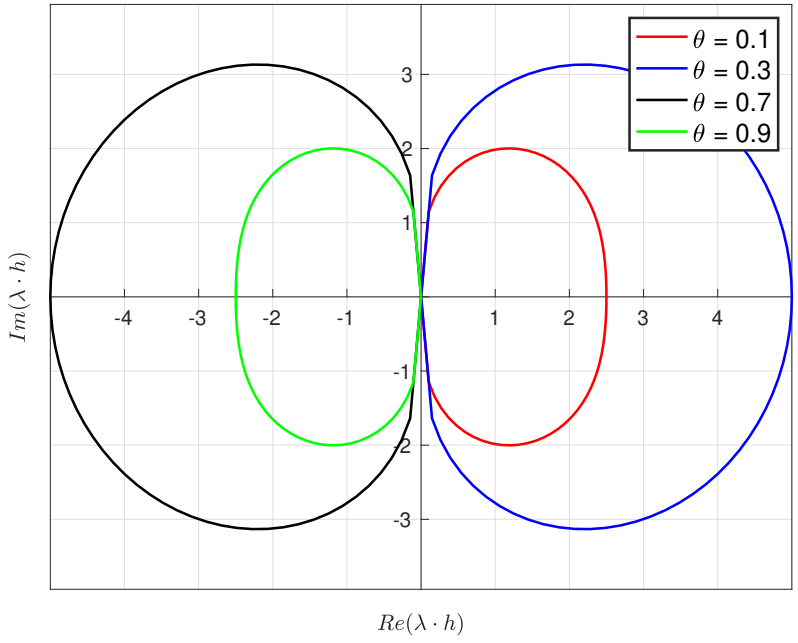


Figure 18: $BI2_\theta$: Stability regions

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.

Preliminary considerations

This section aims to show the main theoretical differences between the two methods, in order illustrate what is expected from the integration results.

- RK4 is an explicit method while IEX4 is implicit.
- RK4 is L-stable while IEX4 is A-stable (particular case of L-stability).
- RK4 requires four function evalutations while IEX4 ten.

Another important consideration concerns the eigenvalues of the matrix B which are:

$$\lambda = \begin{pmatrix} -1 \\ -400 \end{pmatrix}$$

As can be observed $\lambda_2 << \lambda_1$ therefore the system seems to be stiff. RK4 is not the right choice for this kind of problem, since it has a limited stability region on the left-hand semi-plane. It is reasonable to use integrators having the left-hand semi-plane as part of their domain of numerical stability, such as A-stable methods.

Integration solutions

To show the integration results, it has been chosen to compare the analytical solution of the problem $\mathbf{x}(t) = e^{Bt}\mathbf{x}(0)$ with the approximate solutions at each discretized time instant. Therefore the error has been defined as $|\mathbf{x}_{analytical} - \mathbf{x}_{approx}|$. If figure 19 are shown the results.

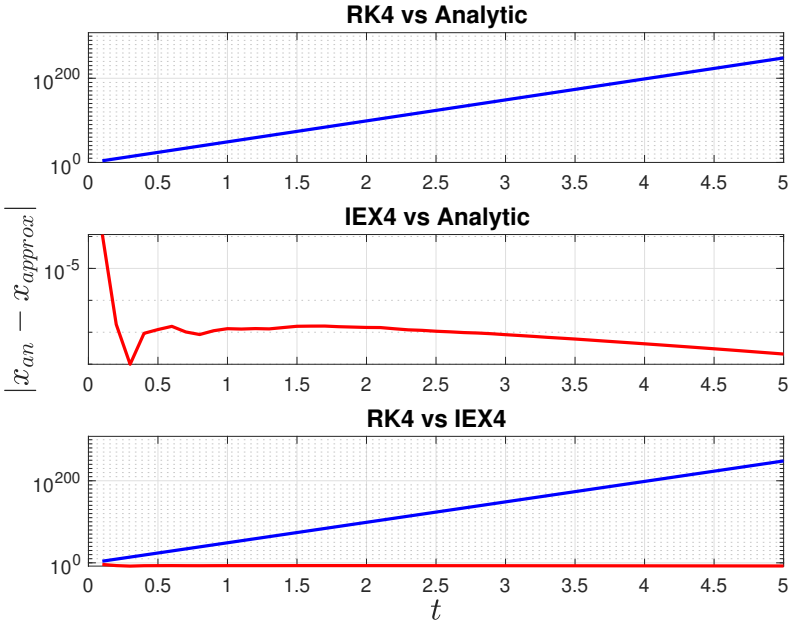


Figure 19: Integration errors

The plots are semi-logarithmic since they show results more clearly. In particular, as we predicted, the error between RK4 and the analytic diverges; this highlights the fact that the method is not suitable or in any case the time step should be drastically reduced, resulting in an increased computational load. Instead the error between IEX4 and the analytic solution is contained.

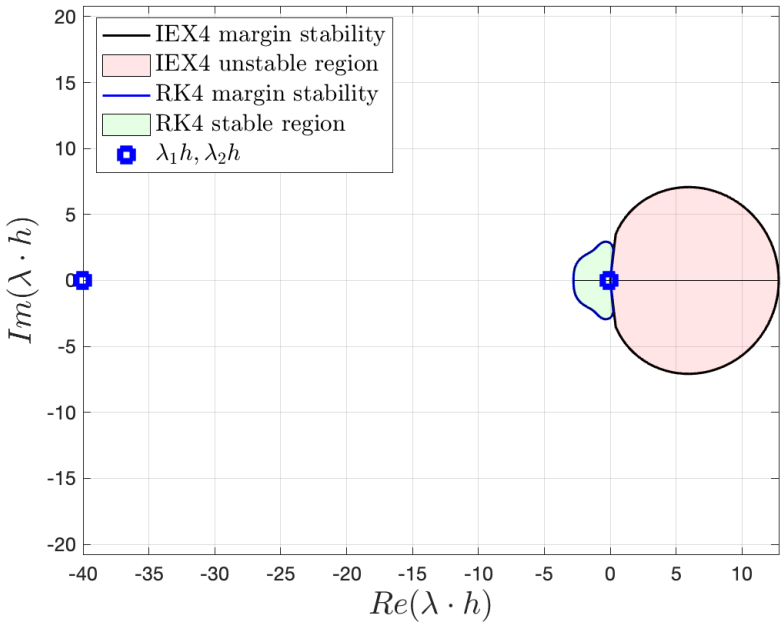


Figure 20: Eigenvalues representation

In order to better understand the problem related to stiff systems it may be useful to look at the figure 20. The two blue dots represent on the λh-plane the eigenvalues associated to the problem:

$$\lambda h = \begin{pmatrix} -0.1 \\ -40 \end{pmatrix}$$

The first eigenvalue is part of both region of stability instead the second belongs only to IEX4 stability region.
In conclusion it can be said that for a fixed step size h = 0.1, RK4 is not suitable and one possible solution it's to reduce the step size to the point where $h\lambda = -2.8$.

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.

Methods implementation

This section briefly reports the implementation of the methods given the two-dimensional initial value problem and a discretized time vector: $\begin{cases}\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t)\\ \mathbf{x}(0) = [1,1]^T\end{cases}$ to highlight implicit methods, explicit methods and how to start the integration process.

$$AB3 : \mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right) \tag{15}$$

$$AM3 : \mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1}\right) \tag{16}$$

$$ABM3 : \begin{cases}\mathbf{x}(t_{k+1})^P = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right)\\ \mathbf{x}(t_{k+1})^c = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}(\mathbf{x}^P_{k+1}, t_{k+1}) + 8\mathbf{f}_k - \mathbf{f}_{k-1}\right)\end{cases} \tag{17}$$

$$BDF3 : \mathbf{x}_{k+1} = \frac{18}{12}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{\mathbf{k-2}} + \frac{6}{11}h\mathbf{f}_{k+1} \tag{18}$$

As can be seen from the equations above, AM3 and BDF3 are implicit methods, instead ABM3 and AB3 are explicit methods. Of course, as it was done before, implicit methods will be solved through the matlab fsolve function.

Startup problem

The problem now is how the integration process is started in the first place. Usually, the initial state vector is given at time t_0, but no back values are available. AB3, ABM3, BDF3 need two values before t_0, AM3 one. In this exercise, RK4 was used to evaluate the function at previous instants. 1^{st} and 2^{nd} order method have been excluded for accuracy problem; to satisfy certain accuracy requirements you should have used a very small step size which is not worth. Instead, with RK4, also the early steps are of the correct order and a decent step size can be used from the first steps. One caution to have is to check that we are not integrating a stiff system, as in our case.

Results and comparison

In figure 21 and 22 are shown the integration solutions divided per components for each method. It can be seen instantly that both components of AB3 diverge completely as we could have expected. To better understand why the different methods behave like this it is appropriate to make a quick study of the eigenvalues associated to the problem under consideration. In doing so it has just been considered the x-components for matters of simplicity, but it could have been done with both the components.

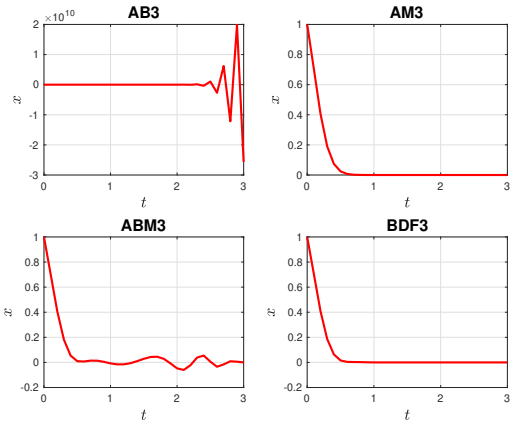


Figure 21: Integration results: x component

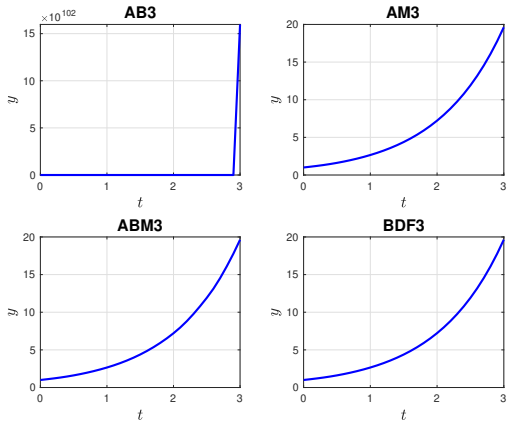


Figure 22: Integration results: y component

In figure 23 we notice that $h\lambda(t)$ is less than zero at each considered instant and so it can already be stated that by integrating with BDF3 the mapping process will not encounter any problems as it's stable in the left-hand semi-plane. With the same immediacy, it can be seen that integrating with AB3 almost at every time instant the stable eigenvalue will be mapped into unstable, as the stability region limit, reported in figure 24, is abundantly exceeded. The AM3 integration is appropriate, since the stability region limit is never achieved and so the mapping process is effective. Lastly, let's consider ABM3 that, as we can see in the x-axis plot, is oscillating around 0. This phenomenon can be explained considering that λh crosses the region's limit of stability zone when the solution is already trending toward zero, and so it cannot explode as in AB3.

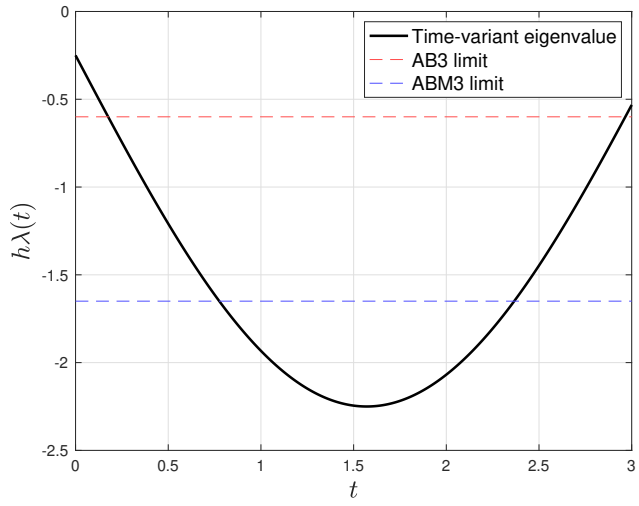


Figure 23: Time-variant eigenvalue

Method	λh limit
AB3	-0.55
AM3	-6
ABM3	-1.7
BDF3	+7

Figure 24: Limits of stability

Strategies for step size and order control are: decreasing h (as in single-step methods) or increasing the order of the integration method. Just be sure to recompute the samples.

References

- Woods, Lawrence, Modeling and Simulation of Dynamic Systems, Prentice Hall, 1997
- A. Quarteroni, F. Saleri, P. Gervasio, Calcolo Scientifico, Springer, 2010
- Lecture notes from the MSc Course 'Modeling and Simulation', F. Topputo