MSAS – Assignment #1: Simulation

Marcello Mutti, 220252

1 Implicit equations

Exercise 1

Let $\mathbf{f}$ be a two-dimensional vector-valued function $\mathbf{f}(\mathbf{x}) = (x_2^2 - x_1 - 2,\ -x_1^2 + x_2 + 10)^\top$, 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?

(3 points)

Given the dimension and order of the function $\mathbf{f}$, it's possible to deduce that there exist 4 zeros. In particular there exist two real solutions and two complex conjugate solutions that satisfy $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, namely:

$$\begin{aligned}
\mathbf{z}_1 &= (3.5140,\ 2.3482)^\top \\
\mathbf{z}_2 &= (2.7947,\ -2.1897)^\top \\
\mathbf{z}_3 &= (-3.1543 - 0.1708j,\ -0.0793 + 1.0773j)^\top \\
\mathbf{z}_4 &= (-3.1543 + 0.1708j,\ -0.0793 - 1.0773j)^\top
\end{aligned}$$

Such solutions can be found exploiting the Newton's method, which iteratively updates an initial guess $\mathbf{x}_0$, until the k^{th} iteration solution $\mathbf{x}_k$ satisfies predefined tolerance constraints:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{\partial\mathbf{f}(\mathbf{x}_k)}{\partial\mathbf{x}}\right)^{-1}\mathbf{f}(\mathbf{x}_k)$$

1) The function $\mathbf{f}$ is analytically differentiable, therefore it is possible to compute the Jacobian:

$$\frac{\partial\mathbf{f}(\mathbf{x})}{\partial\mathbf{x}} = \begin{bmatrix} -1 & 2x_2 \\ -2x_1 & 1 \end{bmatrix}$$

In order to correctly identify the different solutions of the zero-finding problem, attention has to be put on the initial conditions of the algorithm. It was found that the given initial conditions $\mathbf{x}_{0,i}$ are sufficiently separated to assure correct convergence to each solution $\mathbf{z}_i$:

$$\begin{aligned}
\mathbf{x}_{0,1} &= (4,\ 3)^\top & \mathbf{x}_{0,3} &= (-3,\ 2j)^\top \\
\mathbf{x}_{0,2} &= (3,\ -2)^\top & \mathbf{x}_{0,4} &= (-3 + j,\ -j)^\top
\end{aligned}$$

The algorithm halts once at least one of the following conditions is met:

$$AE = \|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq AbsTol = 1 \times 10^{-9}$$

$$RE = \frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|} \leq RelTol = 1 \times 10^{-12}$$

The maximum number of iterations is also constrained to to $N_{max} = 100$.
Below the solutions are reported in terms of number of iterations n_{it}^{an} to convergence and absolute error with respect to the analytical solution:

	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_4$
n_{it}^{an}	5	4	6	5
AE	1.0019×10^{-14}	8.8818×10^{-16}	5.0952×10^{-16}	5.0363×10^{-16}

Table 1: Number of iterations and absolute error, analytical Jacobian

2) The Jacobian of $\mathbf{f}$ can be estimated exploiting finite differences linear approximation. For example, using forwards differences (FD):

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i} \approx \left[\frac{\mathbf{f}(\mathbf{x}+\boldsymbol{\Delta}_1)-\mathbf{f}(\mathbf{x})}{\Delta} \quad \frac{\mathbf{f}(\mathbf{x}+\boldsymbol{\Delta}_2)-\mathbf{f}(\mathbf{x})}{\Delta} \right] \quad \text{with} \quad \boldsymbol{\Delta}_i = \Delta \begin{bmatrix} \delta_{1,j} \\ \delta_{2,j} \end{bmatrix}$$

The elements $\delta_{i,j}$ are Kronecker deltas. The FD step size is represented by Δ.

To compare the performance of finite differences approximation against analytical computation the problem was solved for different step sizes, given the same initial conditions and stopping criteria.

At first, the numbers of maximum iterations were reduced to match those of the analytical solution in Section 1 $n_{it} \leq n_{it}^{an}$, without regard for absolute and relative error.

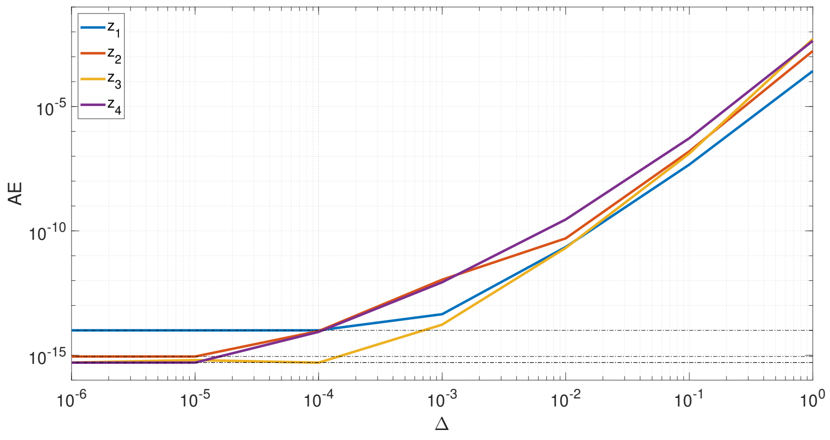

Figure 1: Absolute error AE in function of Δ, forward differences and constrained n_{it}.

As the step size decreases the solutions converge to those obtained using the analytical formula. It's worth noticing that the convergence is not monotonous.

The experiment is repeated without constraint on n_{it}, meaning the algorithm employing finite differences is allowed to reach convergence.

As expected the solutions' precision increases, even for low Δ. Again, as the step size decreases the analytical and approximated solutions become indistinguishable.

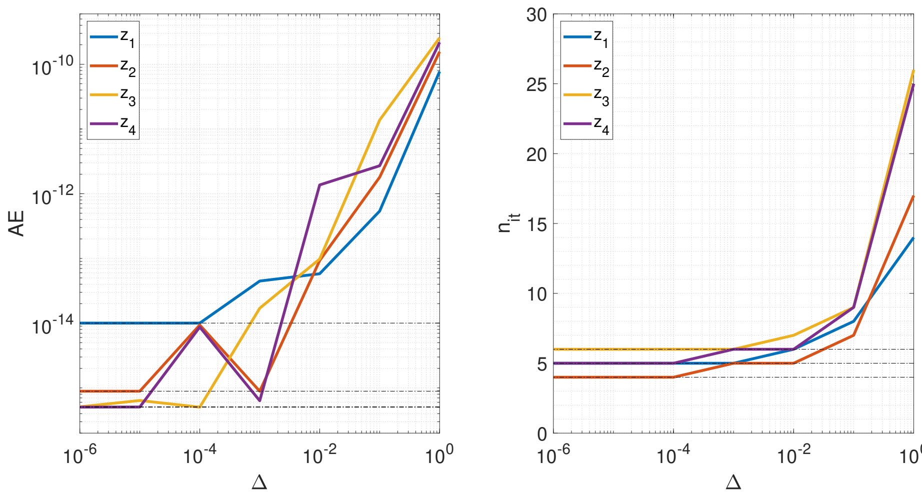

Figure 2: Absolute error AE in function of Δ (left), number of iterations n_{it} in function of Δ (right), forward differences.

2 Numerical solution of ODE

Exercise 2

The Initial Value Problem $\dot{x} = x - 2t^2 + 2$, $x(0) = 1$, has analytic solution $x(t) = 2t^2 + 4t - e^t + 2$. 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.

(4 points)

1) The fixed-step Heun's method is implemented as follows:

$$\begin{cases} \mathbf{x}^P = \mathbf{x}_k + h\mathbf{f}\left(\mathbf{x}_k, t_k\right) \\ \mathbf{x}_{k+1} = \mathbf{x}_k + \dfrac{h}{2}\left[\mathbf{f}\left(\mathbf{x}_k, t_k\right) + \mathbf{f}\left(\mathbf{x}^P, t_{k+1}\right)\right] \end{cases}$$

Where $t_k = t_0 + kh$, $\mathbf{x}_k = \mathbf{x}\left(t_k\right)$ and $k = 0 : N$, $N = \dfrac{t_f - t_0}{h}$.

2) Heun's method (RK2) is used to propagate the initial condition on the given time interval. The integration is repeated for different step sizes h, and the result compared to the analytical solution in terms of absolute error AE. At each time instant t_k:

$$AE = \left|x(t_k) - x_k^{RK2}\right|$$

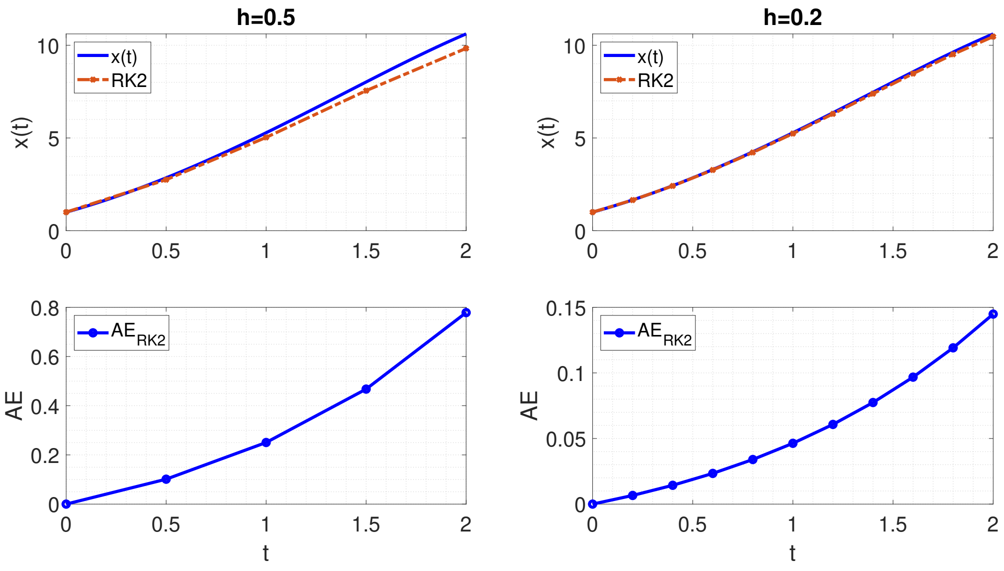

Figure 3: Analytical solution vs RK2.

Given the low order of the method ($n = 2$) for low step sizes the truncation error is significant. At $t = t_f$ the accumulation error is evident and for $h = 0.5$, almost comparable in order of magnitude to $x\left(t_f\right)$.
Decreasing the step size to $h = 0.01$ brings the error to a manageable order of magnitude. To further decrease the error without reducing h it's possible to increase the order n of the integration scheme.

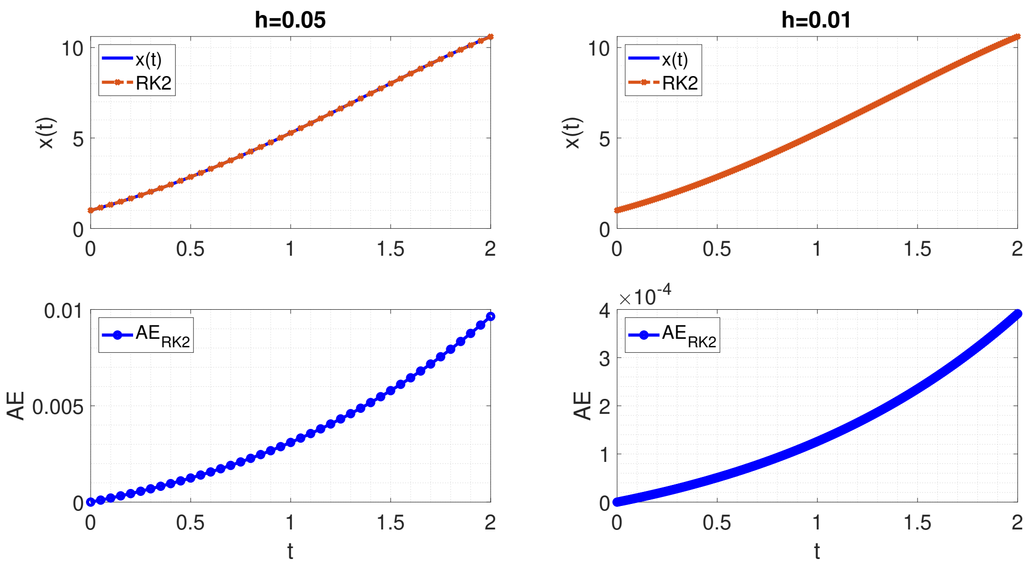

Figure 4: Analytical solution vs RK2.

3) The Runge-Kutta 4 method is implemented as follows:

$$\begin{cases} \mathbf{k}_1 = \mathbf{f}\left(\mathbf{x}_k, t_k\right) \\ \mathbf{k}_2 = \mathbf{f}\left(\mathbf{x}_k + \dfrac{h}{2}\mathbf{k}_1, t_{k+1/2}\right) \\ \mathbf{k}_3 = \mathbf{f}\left(\mathbf{x}_k + \dfrac{h}{2}\mathbf{k}_2, t_{k+1/2}\right) \\ \mathbf{k}_4 = \mathbf{f}\left(\mathbf{x}_k + h\mathbf{k}_3, t_{k+1}\right) \\ \mathbf{x}_{k+1} = \mathbf{x}_k + \dfrac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \end{cases}$$

It is used to evaluate the same IVP as per point 2.

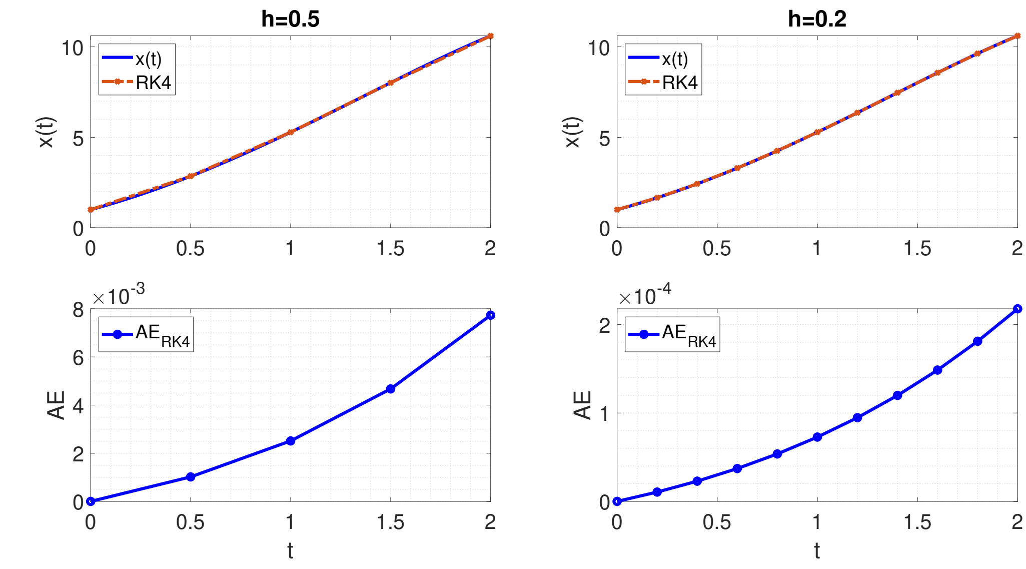

Figure 5: Analytical solution vs RK4.

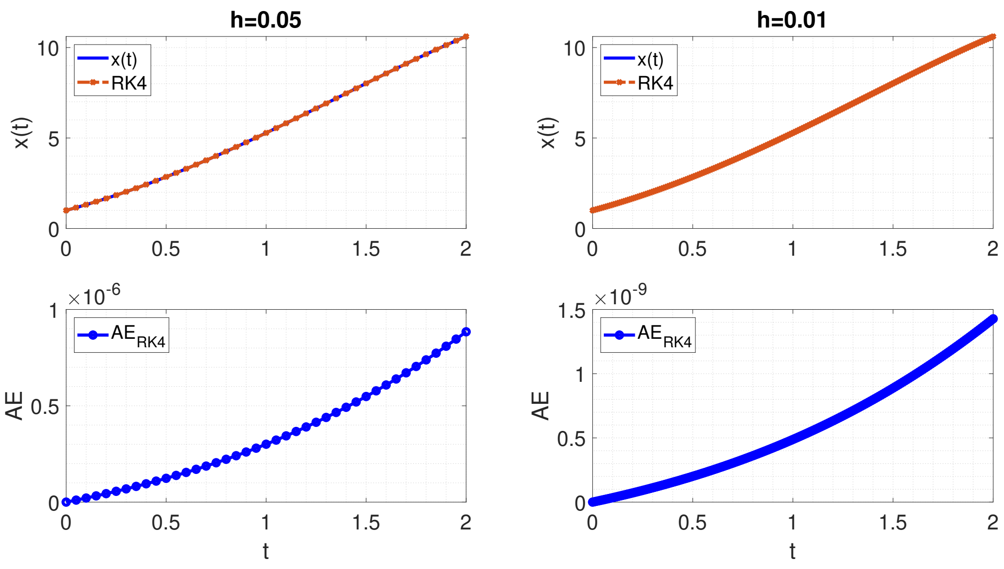

Figure 6: Analytical solution vs RK4.

As expected the higher order ($n = 4$) method yields much more satisfactory results even for the smallest step size.

4) The trade off between two methods of different order can be evaluated both in terms of computational time T_{CPU} over step size h, and maximum relative integration error $RE_{\max}$ over computational time T_{CPU}.

   T_{CPU} is computed as the mean of many realizations of `Matlab`'s function `timeit`.

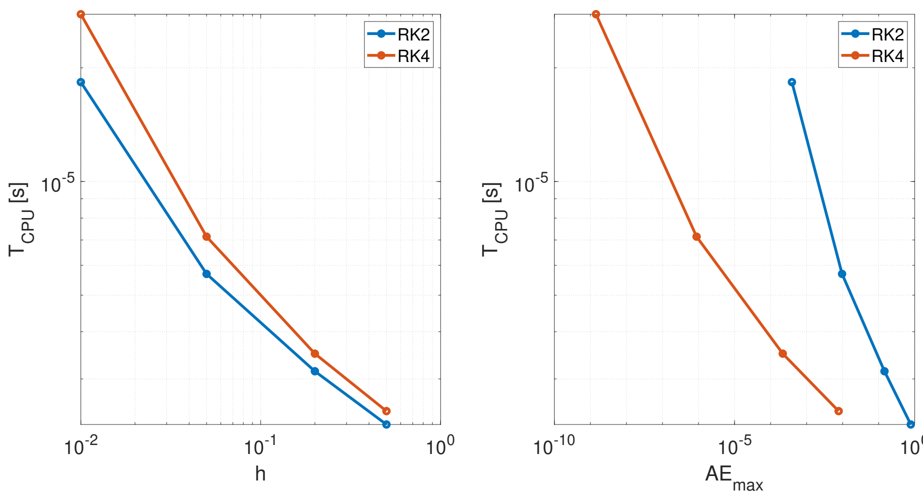

Figure 7: Computational time in function of h (left), and in function of $RE_{\max}$ (right).

Given an arbitrary step size h the computational time of an higher order method is expected to be greater, as it has to perform a greater number of function evaluations.

Analogously, given the same computational time T_{CPU}, an higher order method is expected to yield a more precise result.
In other words, if an admissible maximum error is fixed a higher order method is capable of meeting such requirement with a greater step size h, therefore lower computational time.

Exercise 3

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 $\mathrm{Re}\{\lambda\}$-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.

(5 points)

1) Starting from RK2 integration scheme and exploiting the linearity of $\mathbf{f}\,(\mathbf{x}, t, \alpha) = A\,(\alpha)\,\mathbf{x}$ it is possible to compute the linear operator:

$$F_{\mathrm{RK2}}(h, \alpha) = I + A\,(\alpha)\,h + \frac{1}{2}\,(A\,(\alpha)\,h)^2$$

2) For arbitrary α, the problem can be formulated as the zero-finding problem of the scalar function:

$$f_{SR}\,(h) = max\,(|eig\,(F_{\mathrm{RK2}}(h, \alpha))|) - 1$$

The solution is found using `MatLab`'s function `fzero`, which given an initial guess h_0 finds the solution of $f_{SR}\,(h) = 0$ through bisection and interpolation.

For $\alpha = \pi$, $h_\pi^{\mathrm{RK2}} = 2.0000$, $h\lambda_\pi^{\mathrm{RK2}} = -2.0000$.

3) Iteratively α is decreased from $\alpha = \pi$ to $\alpha = 0$, and the $f_{SR}\,(h)$ is updated accordingly. For sake of generality, the possibility of finding multiple solutions for the same value of α_i is taken into account: at each iteration, h_{α_i} is searched starting from different initial guesses ranging from $h_0 = 0$ and $h_{0v} \propto h_{\alpha_{i-1}}$.

A tolerance $Tol = 10^{-9}$ is set to avoid collecting solutions that differ by less than Tol. The algorithm also rejects solutions $h_{\alpha_i} < Tol$, as $h = 0$ is expected only for $\alpha = 0$ and added a posteriori. The algorithm starts at $\alpha = \pi$ to avoid stalling in the origin.
All the solutions are stored in an array collecting all the zeros found, for decreasing α, in the form of $h\lambda = he^{\alpha j}$.

```
% Solution for alpha=pi
h0=fzero(@(h) max(abs(eig(F_RK(pi,h))))-1,h0guess);
% Iterating on alpha from pi to 0
for i=1:m
    % h0 sweep vector
    h0v=linspace(h0,0,n);
    for j=1:n-1
        % f_SR zero-finding solution
        h=fzero(@(h) max(abs(eig(F_RK(alpha(i),h))))-1,h0v(j)+1);
        % First non-zero solution is collected
        if j==1 && h>Tol
            hh=[hh h];
            aa=[aa alpha(i)];
            h0=hh(end);
        % Different non-zero solutions are collected
        elseif abs(hh(end)-h)>Tol && h>Tol
            hh=[hh h];
            aa=[aa alpha(i)];
            h0=hh(end-1);
        end
    end
end
```

For the purpose of plotting the stability region, the solutions might need to be rearranged so that the ordering of $h\lambda$ is not based on α, but rather on the distance of each point $(Re\{h\lambda\}, Im\{h\lambda\})$ with respect to its neighbouring ones.

Such a problem can be interpreted as the search of the minimum length path for a Travelling Salesman Problem: starting from $(h\lambda)_\pi$, the points are therefore rearranged within the array with a simple Bubble Sort like algorithm based on the distance of each point with respect to the unordered ones.

```
hh % Collection of h solutions
hl % Collection of [Re(h*lambda) Im(h*lambda)]
% TSP ordering
for i=1:length(hh)-1
    % Reference distance d
    d=norm(hl(:,i)-hl(:,i+1));
    for j=i+1:length(hh)-1
        % If a point is found at distance < d they are switched
        dn=norm(hl(:,i)-hl(:,j+1));
        if dn<d
            temp=hl(:,j+1);
            hl(:,j+1)=hl(:,i+1);
            hl(:,i+1)=temp;
            % Reference distance d update
            d=norm(hl(:,i)-hl(:,i+1));
        end
    end
end
```

The Stability Region is reported in the following point.

4) Starting from RK4 integration scheme, the associated linear operator:

$$F_{\text{RK4}}(h,\alpha) = I + A\left(\alpha\right)h + \frac{1}{2}\left(A\left(\alpha\right)h\right)^2 + \frac{1}{6}\left(A\left(\alpha\right)h\right)^3 + \frac{1}{24}\left(A\left(\alpha\right)h\right)^4$$

The zero-finding problem is solved in the same fashion as the previous point. The solutions are ordered according to the previously presented algorithm.

For $\alpha = \pi$, $h_\pi^{\text{RK4}} = 2.7853$, $h\lambda_\pi^{\text{RK4}} = -2.7853$.

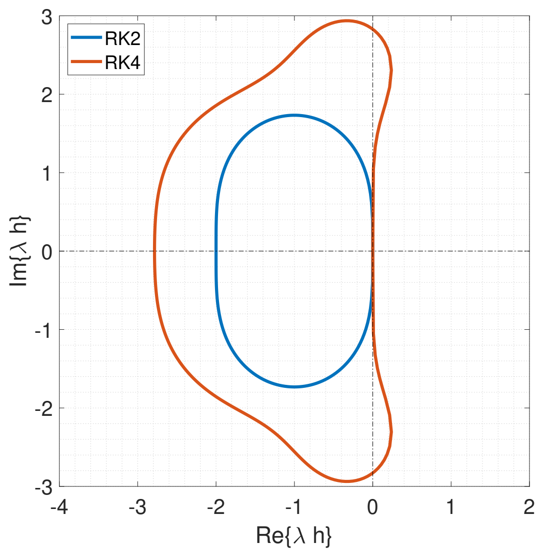

Figure 8: Stability Regions for RK2 and RK4.

Exercise 4

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 $\alpha \in [0, \pi]$ 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 four 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.

(4 points)

1) The Runge-Kutta 1 method coincides with Forward Euler:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}\left(\mathbf{x}_k, t_k\right)$$

Where $t_k = t_0 + kh$, $\mathbf{x}_k = \mathbf{x}\left(t_k\right)$ and $k = 0 : N$, $N = \dfrac{t_f - t_0}{h}$.
The associated linear operator:

$$F_{\text{RK1}}(h, \alpha) = I + A\left(\alpha\right) h$$

It is possible to exploit the linearity of the problem both for the computation of $\mathbf{x}_{\text{an}}(1)$ and $\mathbf{x}_{\text{RK1}}(1)$. In particular, for arbitrary α:

$$\mathbf{x}_{\text{an}}\left(1\right) = \mathbf{x}_{\text{an}}\left(t_f\right) = e^{A(\alpha)\left(t_f - t_0\right)}\mathbf{x}\left(t_0\right)$$

$$\mathbf{x}_{\text{RK1}}\left(1\right) = \mathbf{x}_{\text{RK1}}\left(t_f\right) = \left[F_{\text{RK1}}(h, \alpha)\right]^N \mathbf{x}\left(t_0\right)$$

The problem can be stated as the zero-finding problem of the scalar function:

$$f_{AR}\left(h\right) = \|e^{A(\alpha)\left(t_f - t_0\right)}\mathbf{x}\left(t_0\right) - \left[F_{\text{RK1}}(h, \alpha)\right]^{\left(t_f - t_0\right)/h} \mathbf{x}\left(t_0\right)\|_\infty - tol$$

For fixed values of α and tol, $f_{AR}\left(h\right) = 0$ is expected to yield a unique non-singular solution, found using `MatLab`'s function `fzero`.
Starting from $\alpha = 0$, the problem is solved iteratively for increasing values of α and fixed tol. The process is then repeated iterating on tol.

```
% Iterating on Tol
for i=1:length(Tol)
    % Iterating on alpha from 0 to pi
    for j=1:m
        % f_AR zero-finding solution
        [h,~,ex_flag]=fzero(@(h) norm(expm(A(alpha(j)))*x0+...
            -((F_RK(alpha(j),h))^((tf-t0)/h))*x0,'inf')+...
            -Tol(i),h0(i));
        % Convergence check
        if ex_flag<=0
            error('fzero error')
        end
        % Collection of solutions h
        H(i,j)=h;
        % Collection of lambda*h
        LH(i,j)=H(i,j)*exp(1i*alpha(j));
    end
end
```

Given the nature of the problem, `fzero` convergence is very sensible to the initial guess. Such guess is fixed over α and updated for each iteration of tol.

For RK1 the initial guesses used: $\left[1 \times 10^{-3},\, 1 \times 10^{-4},\, 1 \times 10^{-5},\, 1 \times 10^{-6}\right]$.

2) The loci of solutions, for different values of *tol*:

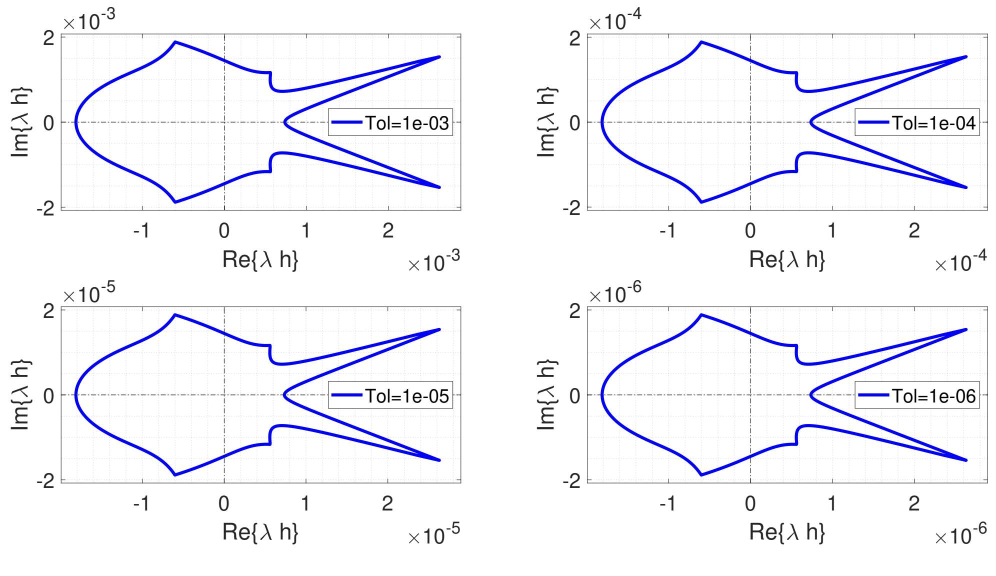

Figure 9: Accuracy Regions of RK1.

RK1 performs only one function evaluation per integration step, therefore at $\alpha = \pi$:

$$N_F^{\text{RK1}} = \lceil N \rceil$$

It ought to be noted that the integration scheme must perform an integer number of steps. Taking $\lceil N \rceil$ ensures that the last integration step evaluates $\mathbf{x}(t)$ at $t \geq t_f$, therefore ensuring the whole integration interval is covered.

The number of function evaluations is reported in the following point.

3) Using the previously defined linear operators $F_{\text{RK2}}(h, \alpha)$ and $F_{\text{RK4}}(h, \alpha)$ it is possible to compute the accuracy regions of both RK2 and RK4.

Given the initial guesses for RK2: $\left[4.5 \times 10^{-2},\, 1.5 \times 10^{-2},\, 5 \times 10^{-3},\, 1.6667 \times 10^{-3}\right]$:

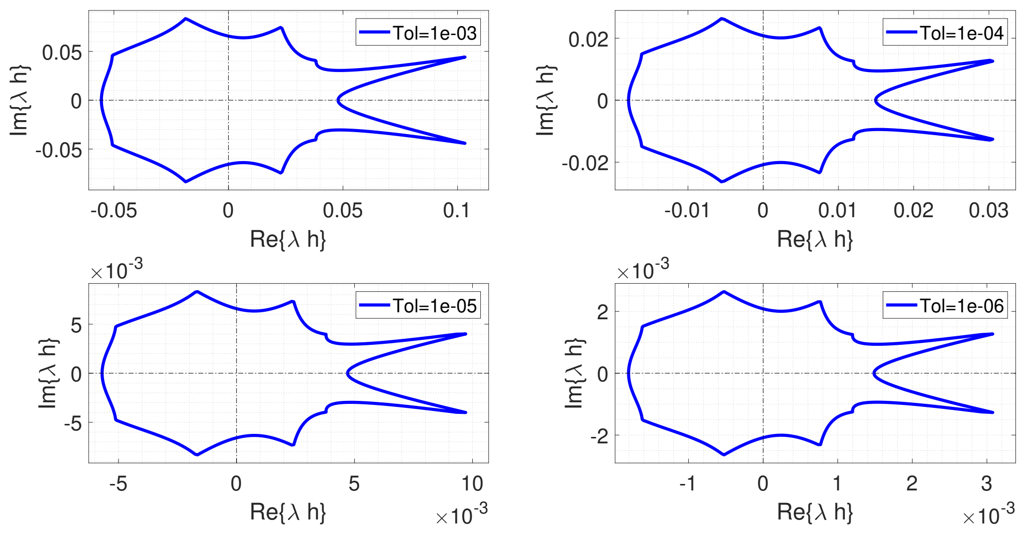

Figure 10: Accuracy Regions of RK2.

Given the initial guesses for RK4: $\left[6\times10^{-1},\, 3\times10^{-1},\, 1.5\times10^{-1},\, 7.5\times10^{-2}\right]$:

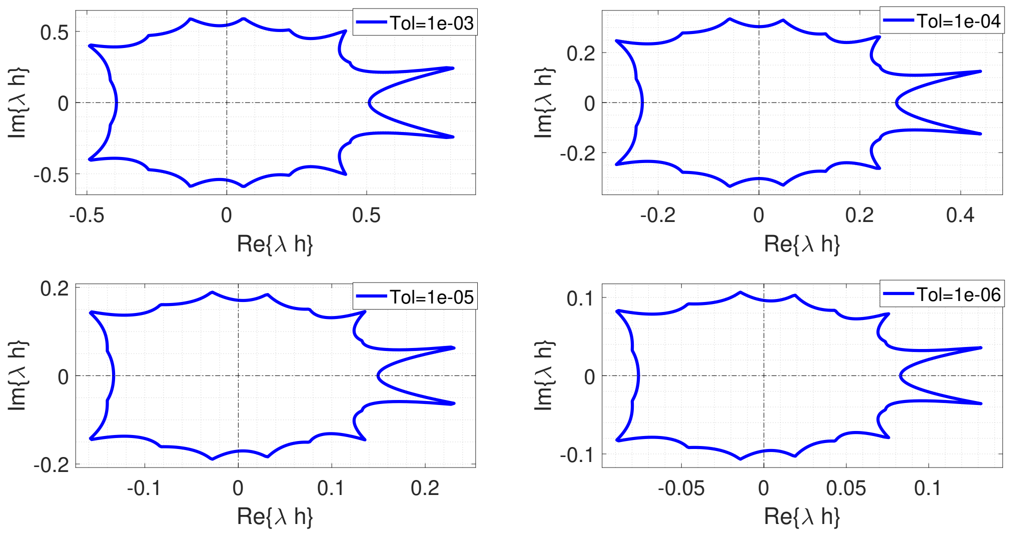

Figure 11: Accuracy Regions of RK4.

RK2 performs two function evaluation per integration step, while RK4 performs four. Therefore at $\alpha = \pi$:

$$N_F^{\text{RK2}} = 2\lceil N \rceil$$
$$N_F^{\text{RK4}} = 4\lceil N \rceil$$

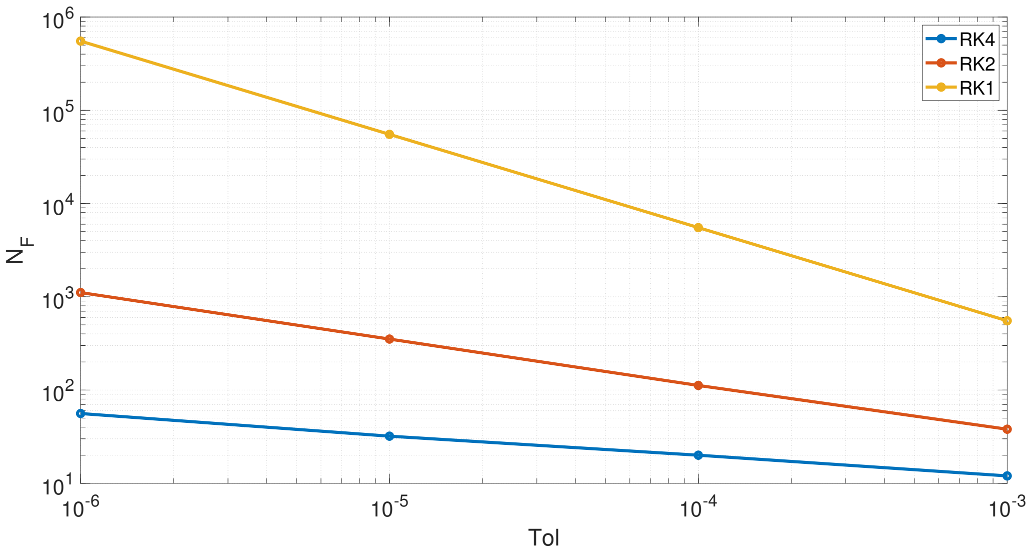

Figure 12: Number of function evaluations in function of *tol*.

Exercise 5

Consider the backinterpolation method 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 3, draw the stability domain of BI2$_{0.4}$ in the $(h\lambda)$-plane. 3) Derive the domain of numerical stability of BI2$_\theta$ for the values of $\theta = [0.1,\, 0.3,\, 0.7,\, 0.9]$.

(5 points)

1) The second order backinterpolation method BI2$_\theta$, $\theta \in [0,\, 1]$ is implemented as follows, combining both RK2 and BRK2:

$$\begin{cases} \mathbf{x}^{PL} = \mathbf{x}_k + \theta h \mathbf{f}\left(\mathbf{x}_k, t_k\right) \\ \mathbf{x}^{L}_{k+\theta} = \mathbf{x}_k + \theta \dfrac{h}{2}\left[\mathbf{f}\left(\mathbf{x}_k, t_k\right) + \mathbf{f}\left(\mathbf{x}^{PL}, t_{k+\theta}\right)\right] \\ \mathbf{x}^{PR} = \mathbf{x}_{k+1} - (1-\theta)\, h \mathbf{f}\left(\mathbf{x}_{k+1}, t_{k+1}\right) \\ \mathbf{x}^{R}_{k+\theta} = \mathbf{x}_{k+1} - (1-\theta)\dfrac{h}{2}\left[\mathbf{f}\left(\mathbf{x}_{k+1}, t_{k+1}\right) + \mathbf{f}\left(\mathbf{x}^{PR}, t_{k+\theta}\right)\right] \end{cases}$$

Where $t_k = t_0 + kh$, $\mathbf{x}_k = \mathbf{x}\left(t_k\right)$ and $k = 0 : N$, $N = \dfrac{t_f - t_0}{h}$.

The computation of the first two terms is fully explicit. The last two terms are computed integrating backwards from a guess of $\mathbf{x}_{k+1}$. The guess is updated until $\mathbf{x}^{R}_{k+\theta} = \mathbf{x}^{L}_{k+\theta}$.

The corresponding linear operator is computed as follows:

$$B_{\text{BI2}_\theta}(h,\alpha) = \left(I + (\theta - 1)\, hA\left(\alpha\right) + \frac{1}{2}\left((\theta-1)\, hA\left(\alpha\right)\right)^2\right)^{-1} F_{\text{RK2}}(\theta h, \alpha)$$

2) The Stability Region of BI2$_{0.4}$ is computed using the same algorithm of Exercise 3. The only modification is that the direction of α sweep is taken as function of θ:

- If $\theta < 0.5$, α is taken from 0 to π, as the stability domain is expected to belong to the first and fourth quadrants of the complex plane.
- If $\theta > 0.5$, α is taken from π to 0, as the stability domain is expected to belong to the second and third quadrants of the complex plane.

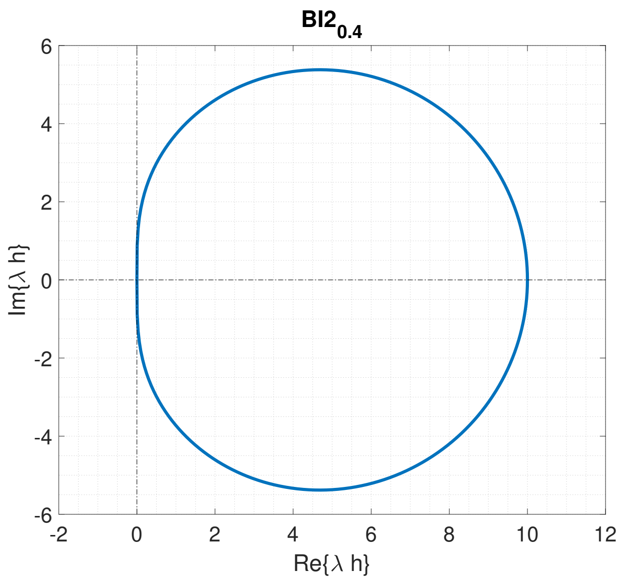

Figure 13: Stability Region for BI2$_{0.4}$.

3) Below the domains of numerical stability of BI2$_\theta$ for $\theta = [0.1, 0.3, 0.7, 0.9]$:

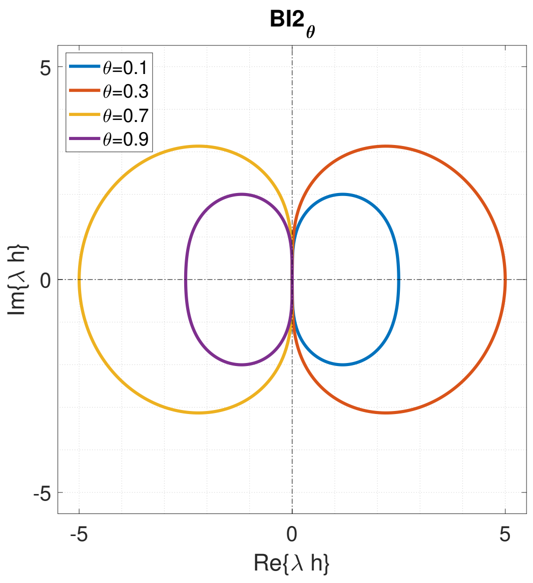

Figure 14: Stability Region for BI2$_\theta$.

It's worth noticing that the Stability Region of BI2$_\theta$ is symmetrical, in terms of contour, with respect to the imaginary axis to BI2$_{1-\theta}$.

It ought to be noted that the symmetry only applies to the contour of the domain: for $\theta > 0.5$ the stable domain is the region of space delimited by the contour, while for $\theta < 0.5$ the stable domain is the region of space outside the boundary.

Exercise 6

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.

(4 points)

1) It is possible to exploit the linearity of the IVP to compute the analytical solution over the integration interval, given the initial condition $\mathbf{x}_0 = \mathbf{x}(0)$:

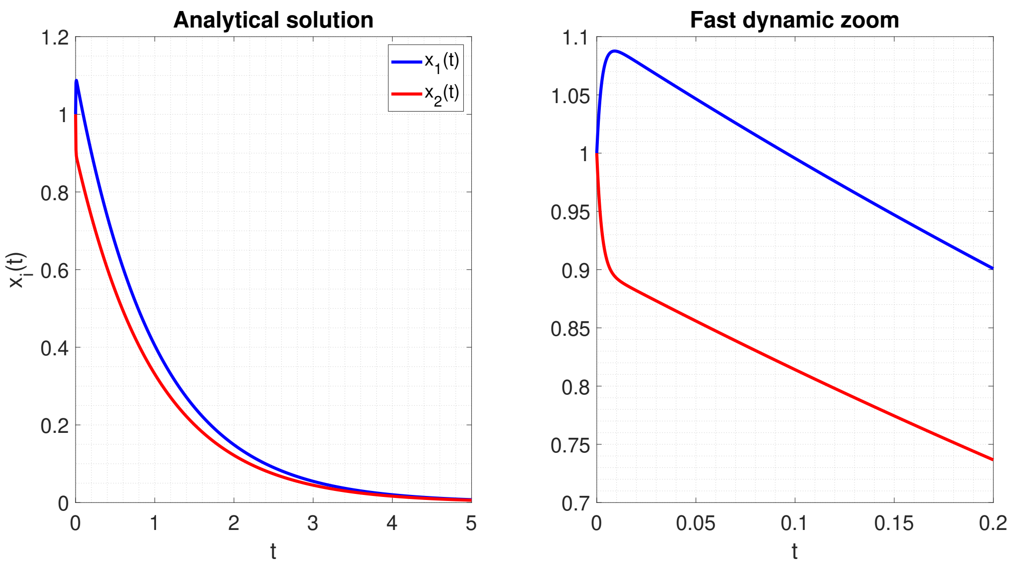

Figure 15: Analytical solution.

Both elements $x_i(t)$ of the solution manifest two dynamic transients: the slower one happening over the whole integration interval, the faster one settling in $t \approx 0.01\,s$.

The IVP is solved employing RK4 with step size $h = 0.1$:

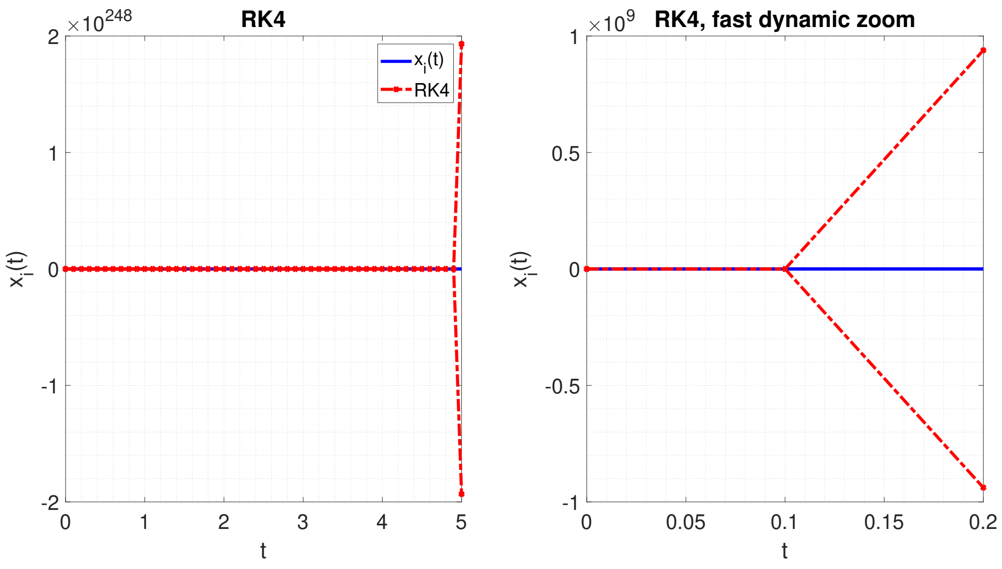

Figure 16: Analytical solution vs RK4.

2) The fourth order Implicit Extrapolation method (IEX4) is implemented as follows:

$$\begin{aligned}
&\mathbf{k}_1 = \mathbf{x}_k + h\mathbf{f}\left(\mathbf{k}_1, t_{k+1}\right) && \mathbf{k}_3 = \mathbf{k}_{3b} + \frac{h}{3}\mathbf{f}\left(\mathbf{k}_3, t_{k+1}\right)\\
&\mathbf{k}_{2a} = \mathbf{x}_k + \frac{h}{2}\mathbf{f}\left(\mathbf{k}_{2a}, t_{k+1/2}\right) && \mathbf{k}_{4a} = \mathbf{x}_k + \frac{h}{4}\mathbf{f}\left(\mathbf{k}_{4a}, t_{k+1/4}\right)\\
&\mathbf{k}_2 = \mathbf{k}_{2a} + \frac{h}{2}\mathbf{f}\left(\mathbf{k}_2, t_{k+1}\right) && \mathbf{k}_{4b} = \mathbf{k}_{4a} + \frac{h}{4}\mathbf{f}\left(\mathbf{k}_{4b}, t_{k+1/2}\right)\\
&\mathbf{k}_{3a} = \mathbf{x}_k + \frac{h}{3}\mathbf{f}\left(\mathbf{k}_{3a}, t_{k+1/3}\right) && \mathbf{k}_{4c} = \mathbf{k}_{4b} + \frac{h}{4}\mathbf{f}\left(\mathbf{k}_{4c}, t_{k+3/4}\right)\\
&\mathbf{k}_{3b} = \mathbf{k}_{3a} + \frac{h}{3}\mathbf{f}\left(\mathbf{k}_{3b}, t_{k+2/3}\right) && \mathbf{k}_4 = \mathbf{k}_{4c} + \frac{h}{4}\mathbf{f}\left(\mathbf{k}_4, t_{k+1}\right)
\end{aligned}$$

$$\mathbf{x}_{k+1} = -\frac{1}{6}\mathbf{k}_1 + 4\mathbf{k}_2 - \frac{27}{2}\mathbf{k}_3 + \frac{32}{3}\mathbf{k}_4$$

Where $t_k = t_0 + kh$, $\mathbf{x}_k = \mathbf{x}\left(t_k\right)$ and $k = 0 : N$, $N = \dfrac{t_f - t_0}{h}$.

It is used to solve the same IVP, with same step size $h = 0.1$:

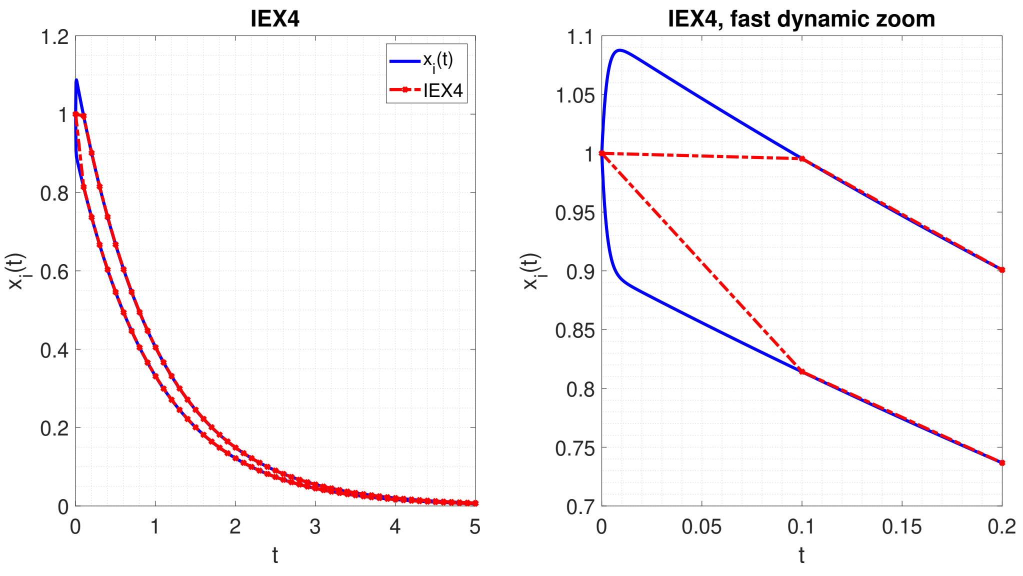

Figure 17: Analytical solution vs IEX4.

3) The numerical approximations of each component x_i are compared to the analytical solutions in term of absolute error AE at each time instant t_k:

$$AE_i = \left|x_i(t_k) - x_{i,k}^{\text{num}}\right|$$

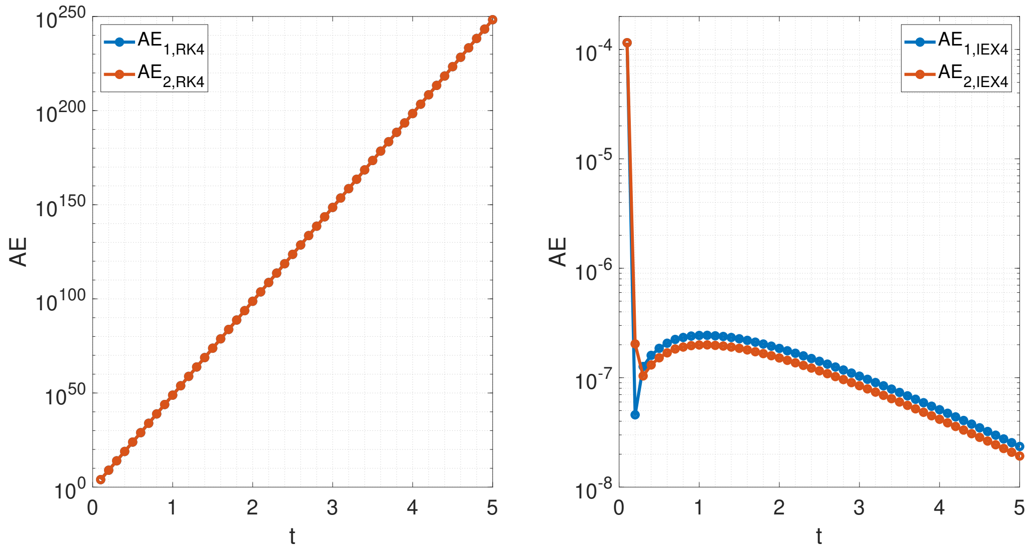

Figure 18: Absolute Errors AE_i of RK2 (left), and IEX4 (right).

4) The problem eigenvalues λ_i are computed as follows:

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -400 \end{pmatrix} = eig\left(B\right)$$

The stability region associated to IEX4 is computed according to the method presented in Exercise 3, given the linear operator:

$$\begin{aligned} F_{\text{IEX4}}(h,\alpha) \quad &= \quad -\frac{1}{6}\left(I - hA\left(\alpha\right)\right)^{-1} + 4\left(I - \frac{h}{2}A\left(\alpha\right)\right)^{-2} + \\ & \quad -\frac{27}{2}\left(I - \frac{h}{3}A\left(\alpha\right)\right)^{-3} + \frac{32}{3}\left(I - \frac{h}{4}A\left(\alpha\right)\right)^{-4} \end{aligned}$$

Given the step size $h = 0.1$, the eigenvalues are represented in the complex plane:

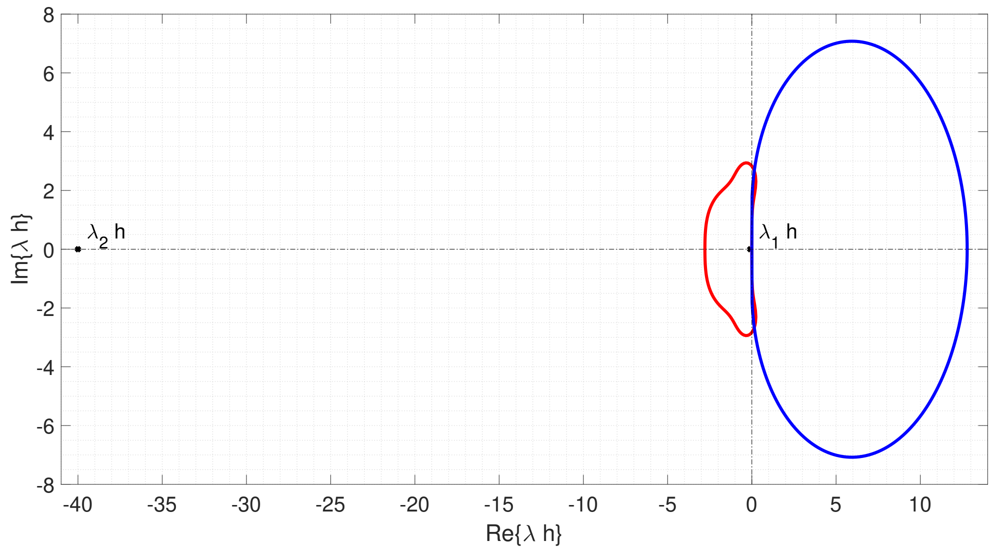

Figure 19: IVP eigenvalues and stability domains of RK4, IEX4.

5) The behaviour of the dynamic transients is confirmed by the computation of the eigenvalues of the system, and consequently the settling times as τ_i:

$$\begin{cases} \tau_1 = \dfrac{\ln 100}{|\lambda_1|} \approx 4.6052 \\ \tau_2 = \dfrac{\ln 100}{|\lambda_2|} \approx 0.0115 \end{cases}$$

RK4 is not capable of capturing the behaviour of either modes, as the eigenvalues of the problem combined with the selected step size $h = 0.1$ do not both belong to the stability domain. In particular for λ_2:

$$-2.7853 = h\lambda_\pi^{\text{RK4}} \gg h\lambda_2 = -40$$

RK4 in fact diverges instantly from the expected solution, outputting a meaningless result.

To achieve convergence a smaller step size is required:

$$h^{\text{RK4}} \leq \frac{\left|h\lambda_\pi^{\text{RK4}}\right|}{|\lambda_2|} \approx 6.963\,233\,91 \times 10^{-3}$$

IEX4 is an $\mathcal{A}$-stable method, therefore it is able to globally approximate the analytical solution with good precision. It is however not capable of capturing the fast transient associated to λ_2. This is due to the fact that the step size $h = 0.1$ is much larger than the settling time of the fast dynamic transient τ_2.

Reducing the step size would allow for a better approximation of the of the first transient, but could result in the over discretization of the integration interval once the fast transient has settled.

Such problem could be eased with the implementation of step size control, therefore allowing for the use of smaller step sizes at $t \approx 0$ and larger step sizes elsewhere.

Exercise 7

Consider the two-dimensional IVP:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-\frac{5}{2}\left[1+8\sin(t)\right]x_1\\ (1-x_1)x_2+x_1\end{bmatrix}, \qquad \begin{bmatrix}x_1(t_0)\\ x_2(t_0)\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}$$

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.

(5 points)

---

1) The third order Adams-Bashforth method (AB3) is implemented as follows:

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

Where $t_k = t_0 + kh$, $\mathbf{x}_k = \mathbf{x}\left(t_k\right)$ and $k = 0 : N$, $N = \dfrac{t_f - t_0}{h}$.

It is used to solve the IVP, with same step size $h = 0.1$.

Since the analytical solution is unknown, the results obtained using AB3 are compared to the results obtained using a higher order integration method. The reference solution $\mathbf{x}^{\text{ref}}$ is computed using `MatLab`'s `ode113` integrator.

It ought to be noted that since AB3 is a multi-step integration scheme, it requires knowledge of 3 past samples. It was chosen to employ the same order Runge-Kutta method to ensure correct startup of AB3.

RK3 is implemented as follows:

$$\begin{cases}\mathbf{k}_1 = \mathbf{f}\left(\mathbf{x}_k, t_k\right)\\ \mathbf{k}_2 = \mathbf{f}\left(\mathbf{x}_k + \dfrac{h}{2}\mathbf{k}_1, t_{k+1/2}\right)\\ \mathbf{k}_3 = \mathbf{f}\left(\mathbf{x}_k + h\left(-\mathbf{k}_1 + 2\mathbf{k}_2\right), t_{k+1}\right)\\ \mathbf{x}_{k+1} = \mathbf{x}_k + \dfrac{h}{6}\left(\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3\right)\end{cases}$$

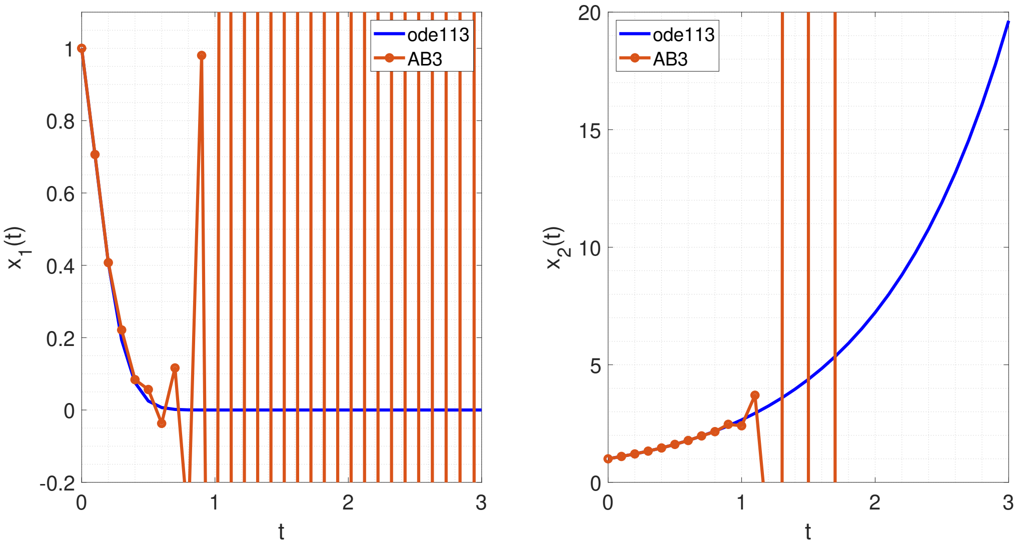

Figure 20: AB3 solution vs `ode113`

2) The third order Adams-Moulton method (AM3) is implemented as follows:

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

The third order Adams-Bashforth-Moulton method (ABM3) exploits AB3 and AM3 to build a totally explicit predictor-corrector scheme. It is implemented as follows:

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

The third order Backwards Differences Formulae method (BDF3) is implemented as follows:

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11}\mathbf{f}\left(\mathbf{x}_{k+1}, t_{k+1}\right)$$

All the aforementioned methods are multi-step methods. ABM3 and BDF3 require knowledge of previous 3 samples, AM3 requires knowledge of 2. Again, RK3 is employed to take care of the startup problem.

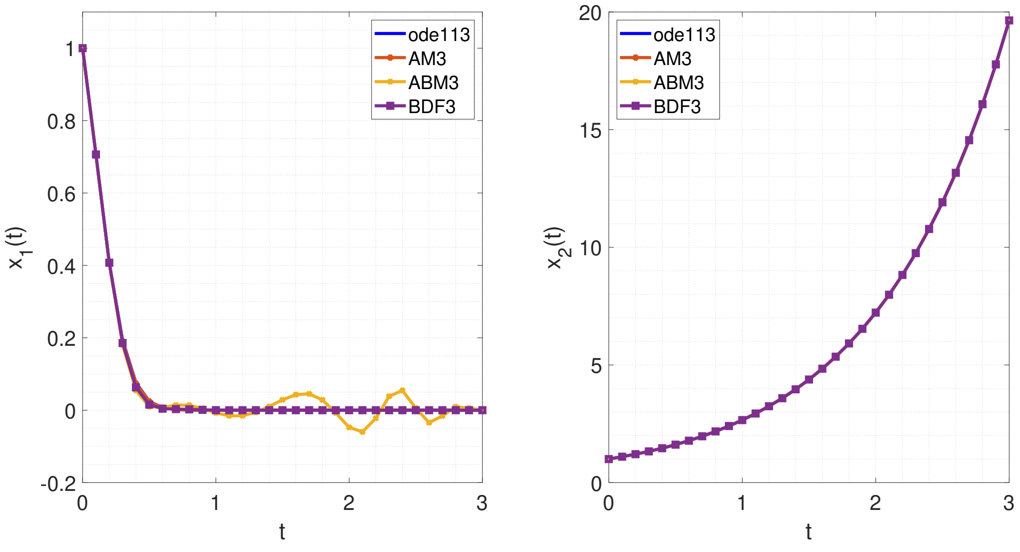

Figure 21: AM3, ABM3, BDF3 solutions vs `ode113`

3) The given IVP is non-linear, therefore its not possible to discuss the methods' stability in terms of eigenvalues. In first approximation the methods' stability is evaluated in function of the step size h only.

With the given step size $h = 0.1$ AB3 manages to perform a few succesful steps before diverging from the reference solution. It's worth noticing that AB3 stability domain is very restricted, even for linear systems.

Below the comparison of AM3, ABM3, BDF3 solutions x_i^{num} with respect to the reference `ode113` solutions x_i^{ref}, in terms of absolute error AE.
At each time instant t_k:

$$AE_i = \left|x_{i,k}^{\text{ref}} - x_{i,k}^{\text{num}}\right|$$

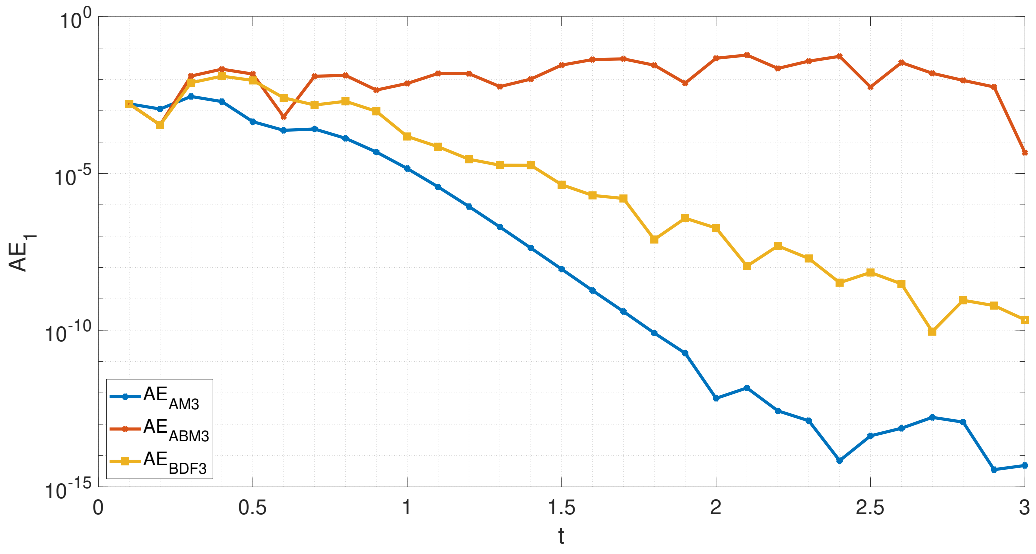

Figure 22: AM3, ABM3, BDF3 absolute error AE on x_1^{ref}.

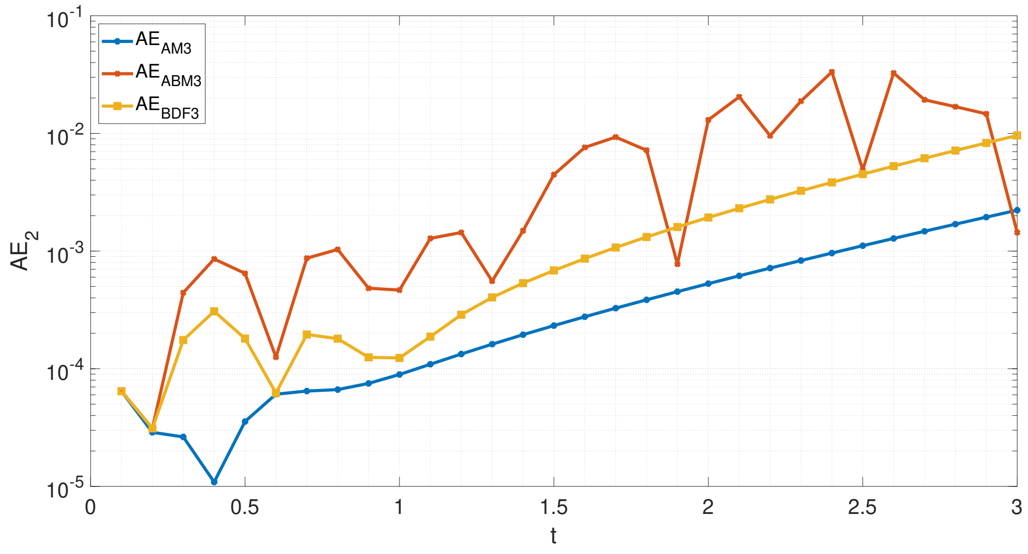

Figure 23: AM3, ABM3, BDF3 absolute error AE on x_2^{ref}.

Both ABn and AMn methods are constructed on the basis on Newton-Gregory backwards polynomials, with the difference that AB methods are expicit and AM methods implicit. Qualitatively, for the same order n, AMn method's expected to perform better.
This is indeed the case, as not only AM3 converges given the same step size h, but it is also the best performing method amongst the ones employed.

As previously stated, ABM3 is obtained by combination of AB3 and AB3. It can therefore be hypothesized that it inherits the properties of both: the divergence of the predictor is reasonably kept under control by the implicit correction.
Qualitatively x_2^{ABM3} performance is comparable to those of AM3 and BDF3. However x_1^{ABM3} is not able to converge, despite the stationarity of the reference solution x_1^{ref}. Possibly a small decrease of h can ease this problem.

It ought to be noted that given the fact that all the presented method are based on extrapolation techniques, they are all subject to extrapolation instability. Its effect is proportional to the order n of the method. This source of error, combined with truncation error can cause substantial accumulation error, leading to inaccurate results.

That is the case for x_2^{num}: as t_k increases, so does AE_2.
$AE_{2,ABM3}$ has an oscillating behaviour, but still shows secular drift.

On the other hand x_1^{AM3} and x_1^{BDF3} perform much better in terms of AE_1. Possibly, this is due to the stability of the reference solution: the extrapolation error becomes negligible, allowing for convergence.