MSAS – Assignment #1: Simulation

Niccolò Bardazzi, 10800456

1 Implicit equations

Exercise 1

Given the function $f(x) = \cos x - x$, 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.

$f(x) = \cos x - x$ is a continuous function monotonically decreasing thus it has only one zero x^* such that $x^* \in [0,\ 1]$, since $-1 \leq \cos x \leq 1$ and for $0 \leq x \leq \frac{\pi}{2}$ both of them are positive, thus $0 \leq x \leq 1$.
The zero of $f(x)$ must be achieved with 8-digit accuracy, hence it has been set a tolerance of 10^{-8} between 2 successive steps.

1) **Bisection method**: 0.73908513|0393505
2) **Secant method**: 0.73908513|3215161
3) **Regula falsi method**: 0.73908513|3171071

All the methods require 1 function evaluation per step plus the ones needed to start the method: 1 for bisection and secant, 2 for falsi regula. From Table 1 it is possible to see that the most efficient method is the secant one since it required the least number of function evaluations and time as well (these results have been compared with [1], demonstrating a perfect match).

Table 1: Root finding methods comparison.

Algorithm	N° of function evaluations	Computational time, s
Bisection	28	$1.038 \cdot 10^{-4}$
Secant	7	$1.698 \cdot 10^{-5}$
Falsi Regula	10	$2.038 \cdot 10^{-5}$

In Figure 1 the errors' trend for each method have been shown in the semi-log plane proving the linear convergence of falsi regula and bisection methods, while an almost parabolic convergence of the secant method (the analytical solution, needed for error's definition, has been retrieved from `fsolve` with 10^{-16} tolerance).

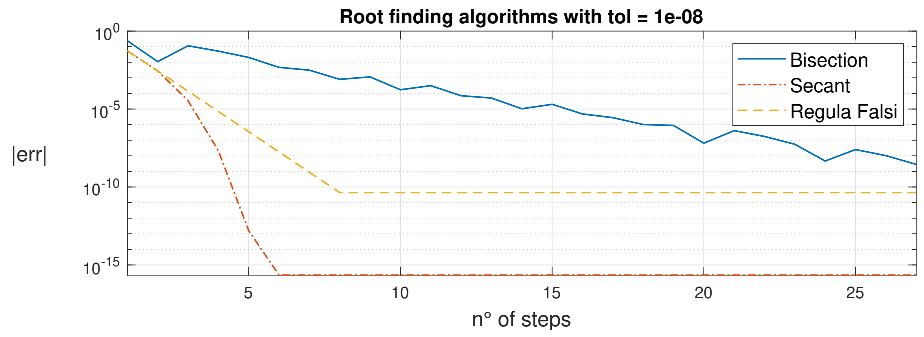

Figure 1: Semi-logarithmic error trend in the three algorithms.

Exercise 2

Let $\mathbf{f}$ be a two-dimensional vector-valued function $\mathbf{f}(\mathbf{x}) = (x_1^2 - x_1 - x_2,\, x_1^2/16 + x_2^2 - 1)^\top$, 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?

There are 2 zeros whose initial guesses have been searched studying the function obtained by substituting x_2 from the second equation into the first one: $x_1^4 - 2x_1^3 + \frac{17}{16}x_1^2 - 1$ (Figure 2). The respective values of x_2 associated to the two zeros have been computed substituting x_1 to one of the equations of $\mathbf{f}(\mathbf{x})$, bringing to $\mathbf{x_1^{(0)}} = [1.5\ 0.75]^\mathrm{T}$ and $\mathbf{x_2^{(0)}} = [-0.5\ 0.75]^\mathrm{T}$.

In order to compare the accuracy, a finite number of iterations ($n = 4$) has been given as input for the two version of Newton's solver.

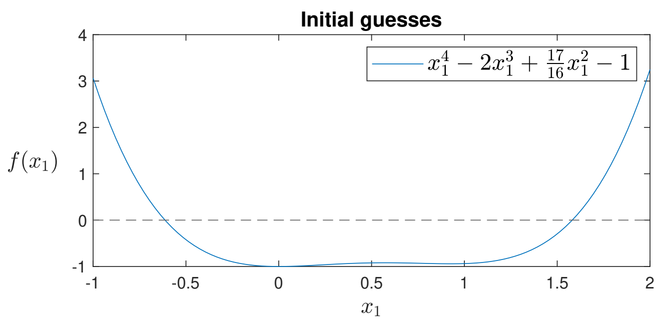

Figure 2: Initial guesses for Newton's method.

1-2) The analytical computation has been performed with symbolic manipulation to achieve $\mathbf{f}'(\mathbf{x_k})$ using `jacobian` function of MATLAB, while the numerical one has been performed both with forward finite differences (FFD) and central finite differences (CFD). In Table 2 the iteration values at each step for the zero $\mathbf{x_1}$ have been displayed.

Table 2: Newton Method firsts 4 steps with analytical and numerical jacobian for $\mathbf{x_1}$.

Jacobian	1st	2nd	3rd	4th
Analytical	1.593137254901961	1.581144874445963	1.581005561415158	1.581005546629657
	0.936274509803922	0.918730422351774	0.918573004402380	0.918572991844084
FFD	1.593137253160341	1.581144874560267	1.581005561417638	1.581005546629658
	0.936274506776922	0.918730422568257	0.918573004405453	0.918572991844084
CFD	1.593137254901961	1.581144874445963	1.581005561415158	1.581005546629657
	0.936274509803922	0.918730422351774	0.918573004402380	0.918572991844084

As it is possible to notice, there are no differences between CFD version and the one with the analytical jacobian since $\mathbf{f}(\mathbf{x})$ is a second order vectorial equation and central difference method matches the TSE till the 2nd order. On the other hand, FFD version is less accurate than the one with analytical jacobian, however symbolic manipulation in MATLAB is $\sim$ 100 - 1000 times slower than regular computations.

$$\mathbf{x_1} = \begin{bmatrix} 1.581005546629657 \\ 0.918572991844084 \end{bmatrix} \qquad \mathbf{x_2} = \begin{bmatrix} -0.612743189208374 \\ 0.988197405129624 \end{bmatrix}$$

$$\mathbf{x_1}_{\mathbf{FFD}} = \begin{bmatrix} 1.581005546629657 \\ 0.918572991844084 \end{bmatrix} \qquad \mathbf{x_2}_{\mathbf{FFD}} = \begin{bmatrix} -0.612743189208374 \\ 0.988197405129624 \end{bmatrix}$$

$$\mathbf{x_1}_{\mathbf{CFD}} = \begin{bmatrix} 1.581005546629657 \\ 0.918572991844084 \end{bmatrix} \qquad \mathbf{x_2}_{\mathbf{CFD}} = \begin{bmatrix} -0.612743189208374 \\ 0.988197405129624 \end{bmatrix}$$

2 Numerical solution of ODE

Exercise 3

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

1-2-3) Heun's method and RK4 for a general purpose have been both implemented on MATLAB. RK4 method provides much more accurate results than Heun's method because the TSE has been matched till the 4th order instead of 2nd order. In addition, since the integration error has the following form: $\frac{d^n \mathbf{x}}{dt^n}(t^*) \cdot \frac{h^n}{n!}$ therefore not only an increment on the order of the method (n) brings benefits but also reduction of the stepsize (h). Both improvements have been displayed in Figure 3.

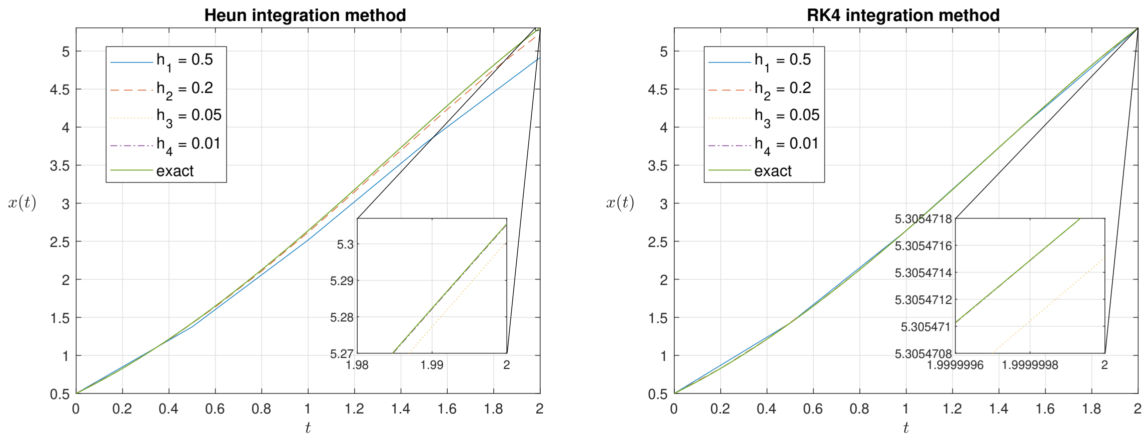

Figure 3: Integration of $\dot{x}$ with Heun's method and RK4 method.

4) However the gain in accuracy leads to a trade-off with the CPU time, which has been shown in Figure 4 (the integration error has been taken with respect to the analytical solution at the last point of the span analyzed).

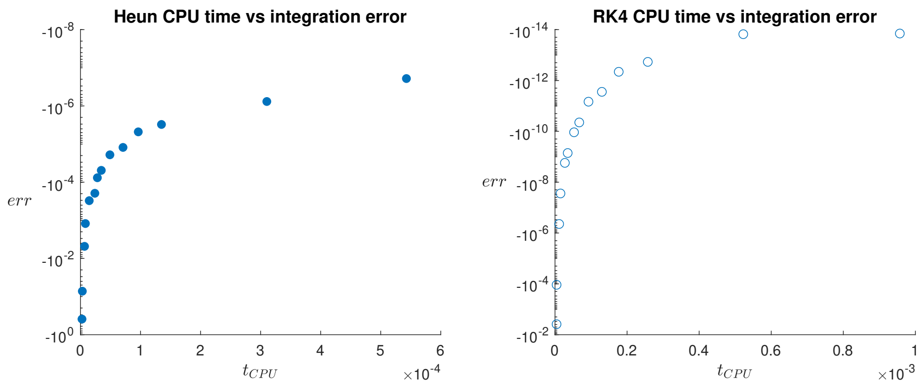

Figure 4: Trade-off between CPU time and integration error.

The plots show an asymptotic graph in the semi-logarithmic plane with a bigger advantage at the very beginning of the plot, since the error decreases with almost no time increment. Then the error stalls with only CPU time increasing, leading to a waste of computational time.

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 with $t = 0$. What can you say?

1-2-3-4) The problem of finding $h \geq 0$ s.t. $\max(|\mathrm{eig}(F(h,\alpha))|) = 1$ is equal to map the stability regions of $F(h,\alpha)$ which is the forward operator that, in case of a LTI system and for RK$_2$ and RK$_4$ schemes, is respectively:

$$\mathbf{F}_{\mathbf{RK_2}}(h,\alpha) = \mathbf{I}^{(\mathbf{n})} + \mathbf{A}(\alpha)\cdot h + \frac{(\mathbf{A}(\alpha)\cdot h)^2}{2}$$

$$\mathbf{F}_{\mathbf{RK_4}}(h,\alpha) = \mathbf{I}^{(\mathbf{n})} + \mathbf{A}(\alpha)\cdot h + \frac{(\mathbf{A}(\alpha)\cdot h)^2}{2} + \frac{(\mathbf{A}(\alpha)\cdot h)^3}{3!} + \frac{(\mathbf{A}(\alpha)\cdot h)^4}{4!}$$

The stability maps have recovered using the bisection method due to its high stability and they have been displayed in Figure 5, where eigenvalues λ_i of $\mathbf{A}(\alpha)$ have not been computed with `eig` but just:

$$\lambda_i = \cos\alpha \pm j\cdot\sin\alpha \tag{1}$$

since they are placed on the unitary circle.

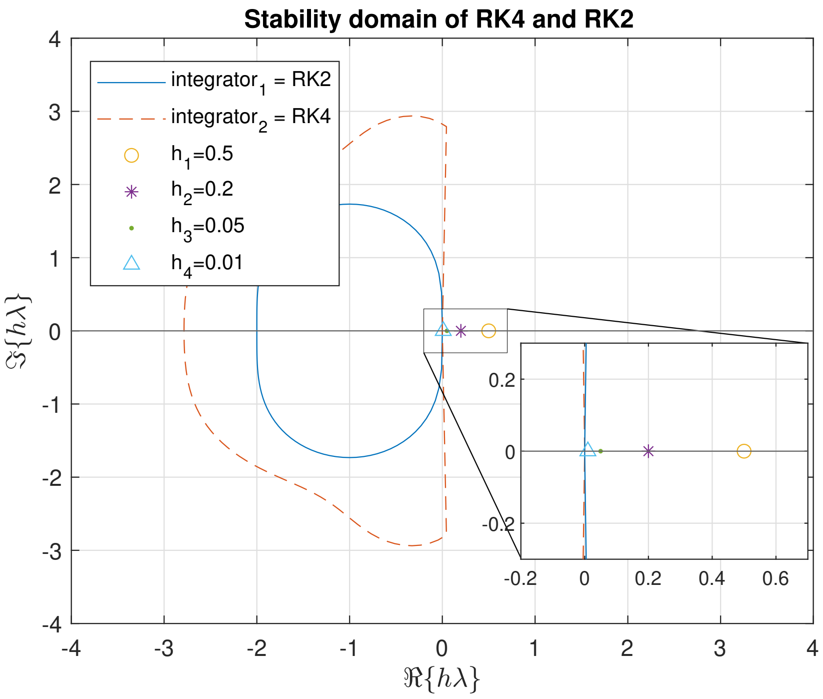

Figure 5: $\{h_i\lambda\}$ points in the stability domain of RK4 and RK2.

The dynamics' matrix of Exercise 3 with $t = 0$ has only $\lambda = 1$, thus $\{h_i\lambda\}$ coincide with the stepsizes provided: $h_1 = 0.5$, $h_2 = 0.2$, $h_3 = 0.05$, $h_4 = 0.01$. As also highlighted from the zoom Figure 5, all the $\{h_i\lambda\}$ are obviously in the right plane, this means that they will be mapped as unstable eigenvalues which is in perfect agreement with the analytical solution of the problem that stays in right plane as well.

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.

1-2-3) The problem of finding $h \geq 0$ s.t. $\|\mathbf{x}_{\text{an}}(1) - \mathbf{x}_{\text{RK1}}(1)\|_\infty = \text{tol}$, is equal to find the trust accuracy region whose bounds on the stepsize satisfy the prescribed error defined as tol and computed with norm infinity. As in Exercise 4, it has been used again the bisection method due to high speed computations and the eigenvalues of $\mathbf{A}(\alpha)$ have been computed with Equation 1. In order to obtain the accuracy domains, shown in Figure 6, the stepsize in RKs must be a divisor of the interval, however this would lead to high computational time to find the right step, therefore the number of steps has been approximated to the lower closest integer number and the last step has been added as the remainder of the span $t \in [0, 1]$. In addition, the maximum allowed error on the function $\|\mathbf{x}_{\text{an}}(1) - \mathbf{x}_{\text{RK1}}(1)\|_\infty - \text{tol} = 0$ has been set to 1% of tolerance.

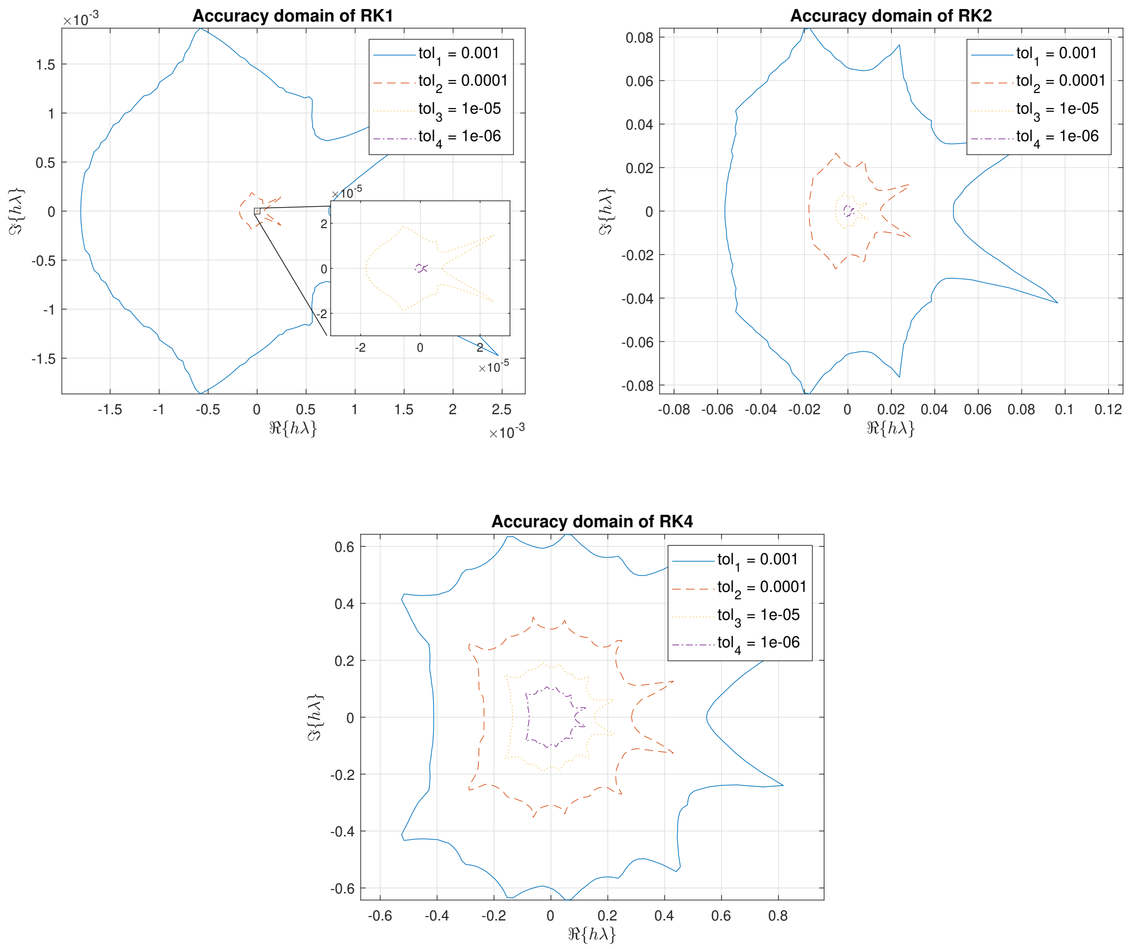

Figure 6: Accuracy domains of RKs.

These results highlight a huge reduction in the accuracy domain of RK1 with increasing the tolerance (till $\sim 10^{-6}$) while a less emphasized one for RK4 (till $\sim 10^{-1}$). This difference of 5 order of magnitude, is related to the truncation error $(\frac{d^n\mathbf{x}}{dt^n}(t^*) \cdot \frac{h^n}{n!})$ that decreases more rapidly for stepsize decrements when the order is higher. This has also been proven in Figure 7 where for error decreasing, the number of function evaluation (related to the inverse of the step) increases with less slope for RK4 than RK2 in the log-log plane.

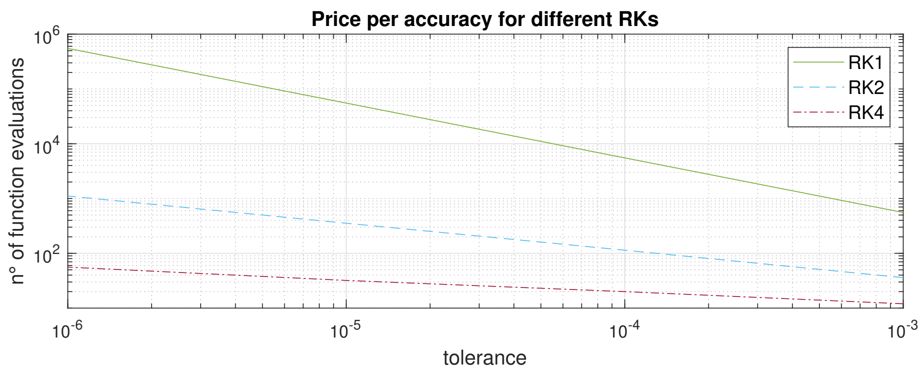

Figure 7: Trade-off between tolerance and number of function evaluations for $\alpha = \pi$.

The higher the order of RKs, the more efficiently function evaluations have been used, therefore increasing the order of RK methods brings only benefits.
With the aim of speeding up computations, it has been observed that the operator F is of an LTI system, hence it has been performed F^{N} instead of repeating the multiplication N times.

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 $\theta = [0.1, 0.3, 0.7, 0.9]$.

---

1) Taking two "substeps" $(h\theta)$, $(h(1-\theta))$ to integrate the domain, firstly explicitly and then implicitly, the RK2 operator will be:

$$\mathbf{F_{RK_2}}(h,\theta) = \left[\mathbf{I^{(n)}} + \mathbf{A}\theta h + \frac{(\mathbf{A}\theta h)^2}{2}\right]$$

while the IRK2 operator:

$$\mathbf{B_{IRK_2}}(h,\theta) = \left[\mathbf{I^{(n)}} - \mathbf{A}(1-\theta)h + \frac{(\mathbf{A}(1-\theta)h)^2}{2}\right]^{-1}$$

Whatever is the choice of θ, the condition to be satisfied is: $\mathbf{x^{(l)}_{k+1}} = \mathbf{x^{(r)}_{k+1}}$ which means that the solution at the half way point reached explicitly $\mathbf{x^{(l)}_{k+1}}$ (l stands for 'from left'), has to match the solution of the same point reached implicitly solution $\mathbf{x^{(r)}_{k+1}}$ (r stands for 'from right'). Therefore:

$$\mathbf{x^{(l)}_{k+1}} = \mathbf{F_{RK2}}(h,\theta)\ \mathbf{x_k} \quad \text{and} \quad \mathbf{x_{k+1}} = \mathbf{B_{IRK2}}(h,\theta)\ \mathbf{x^{(r)}_{k+1}}$$

But due to the constraint $\mathbf{x^{(l)}_{k+1}} = \mathbf{x^{(r)}_{k+1}}$:

$$\mathbf{x_{k+1}} = \mathbf{B_{IRK2}}(h,\theta)\ \mathbf{F_{RK2}}(h,\theta)\ \mathbf{x_k} = \mathbf{B_{BI_2}}(h,\theta)\ \mathbf{x_k}$$

where:

$$\mathbf{B_{BI_2}}(h,\theta) = \left[\mathbf{I^{(n)}} - \mathbf{A}(1-\theta)h + \frac{(\mathbf{A}(1-\theta)h)^2}{2}\right]^{-1} \cdot \left[\mathbf{I^{(n)}} + \mathbf{A}\theta h + \frac{(\mathbf{A}\theta h)^2}{2}\right]$$

2-3) For the stability region on BI2 method it has been used the same algorithm of Exercise 4 with an additional change in the error sign inside the `while` loop whenever $\theta < 0.5$ because the method starts behaving like an explicit method, therefore the eigenvalues of $\mathbf{B_{BI_2}}(h,\theta)$ are on the other side of the imaginary plane as shown in Figure 8.

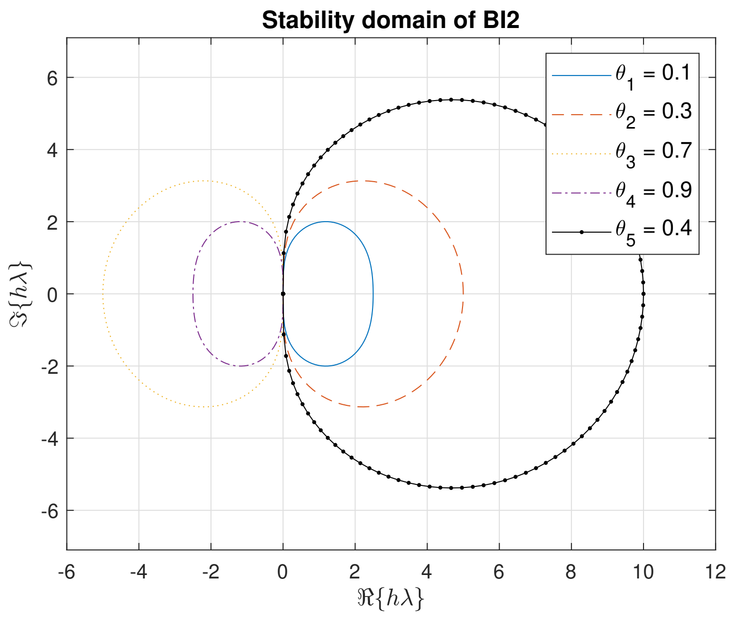

Figure 8: BI2_θ stability domains.

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 BI$2_{0.1}$; 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 BI$2_{0.1}$; 5) Discuss the results.

1-2-3) The system has been integrated both with BI2 and RK4 schemes. The respective errors have been displayed in Figure 9, highlighting a small error of BI2 solution while a total disagreement with the one of RK4.

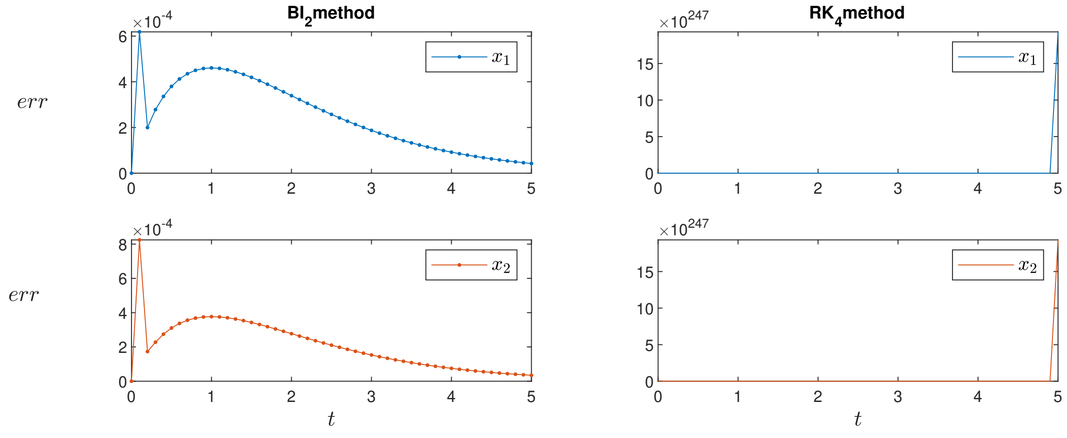

Figure 9: Stiff problem integration errors.

4-5) This behaviour is due to the fact that the problem is a stiff problem which means that the eigenvalues of B have some order of magnitude of difference, $eig(B) = [-1, -400]$. Even if they are both negative, hence the real system has a stable dynamics, RK4 integrator maps it as an unstable one because one of the two eigenvalues falls out of RK4 stability region (Figure 10). Therefore since the dynamics is coupled, both x_1 and x_2 components diverge from the analytical solution, which brings to a huge error with respect to the real system.

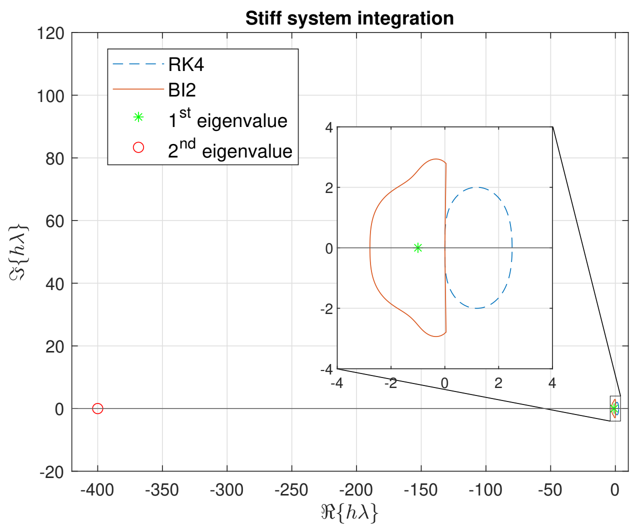

Figure 10: RK4, BI2 stability region and eigenvalues of the dynamics' matrix.

In order to have nice match with RK4, the stepsize should be reduced at least $\sim \frac{400}{2.8}$ times. For BI2 instead, there are no problems since it maps all the stability region into a stable one, indeed the error is very limited.

References

[1] J. Ehiwario and S. Aghamie. "Comparative study of bisection, Newton-Raphson and secant methods of root-finding problems". In: *IOSR Journal of Engineering* 4.4 (2014), pp. 01–07.