TMA4215 Numerical Mathematics

Project: Numerical Integrators for Charged Particles

Norwegian University of Science and Technology (NTNU)

Deadline for submission: November 17th, 2025.

Practical information

- This part of the project counts for 10% of the final mark. The work should be performed and handed in by groups of 2-3.
- Most of this subproject focuses on the implementation (and testing) of codes in Python. You will submit a report in form of a Jupyter notebook file. This file will contain the code as well as explanations and comments in text of your findings and of your numerical results. Make sure that your code contains a meaningful documentation, in particular a reasonable number of comments to explain what you have done. Also test the programs before submission and comment on all strange behaviour of the programs in your report.
- The Jupyter notebook report and the code should be submitted in Blackboard by one of the group members. Include the names of all group members in the report.
- There will be four exercise sessions devoted to the supervision of the project (November 5, 7, 10 and 14). See Blackboard for details on time and place as well as updated information.

1 Motivation

A *Tokamak* is a device creating a toroidal (donut-shaped) magnetic field that allows the confinement of charged particles—if the density and temperature are high enough, nuclear fusion reactions can be sustained. To control and optimize such devices, one needs accurate and efficient numerical methods for simulating the motion of charged particles in electromagnetic fields over long time intervals. In such cases, standard methods, such as Runge–Kutta 4, often display slow drift in energy leading to simulations that violate the physical properties of the system. Methods that are specifically designed to preserve such physical properties are often called *geometric numerical integrators*. In this project you will implement one standard method (RK4) and two different integrators designed to preserve the energy over longer time intervals in charged-particle dynamics: the Boris method and a fourth-order symmetric multistep method.

2 Governing equations

We consider a particle with position $x(t) \in \mathbb{R}^3$ and velocity $v(t) = \dot{x}(t)$ moving in a static magnetic field $B(x) = \nabla \times A(x)$ with vector potential $A(x)$ and an electrostatic potential $U(x)$. We assume unit mass and charge. Two equivalent forms (see [2] for a derivation of the latter) of second order differential equation are used:

$$\dot{x} = v, \qquad \dot{v} = v \times B(x) - \nabla U(x), \tag{1}$$

$$\ddot{x} = A'(x)^\top \dot{x} \; - \; \tfrac{\mathrm{d}}{\mathrm{d}t} A(x) \; - \; \nabla U(x). \tag{2}$$

The following energy is conserved along solutions of (1) and (2):

$$E(x, v) = \tfrac{1}{2}\|v\|_2^2 + U(x). \tag{3}$$

Notation. $\dot{x} := \frac{\mathrm{d}}{\mathrm{d}t}x$ denotes the time derivative of x and $\ddot{x}$ the second derivative. The cross product is denoted by $\times$, and $A'(x)$ denotes the Jacobian of A. We will use x_1, x_2, x_3 for the components of x and $x_{(n)} \approx x(t_n)$ for discrete approximations at time t_n.

3 Model problems

You will work with the two following systems:

3.1 2D charged particle (adapted from [2])

$$B(x) = \begin{bmatrix} 0 \\ 0 \\ \sqrt{x_1^2 + x_2^2} \end{bmatrix}, \qquad A(x) = \frac{1}{3}\begin{bmatrix} -x_2\sqrt{x_1^2 + x_2^2} \\ x_1\sqrt{x_1^2 + x_2^2} \\ 0 \end{bmatrix},$$
$$U(x) = \frac{1}{100\sqrt{x_1^2 + x_2^2}}, \qquad \nabla U(x) = \frac{1}{100}\begin{bmatrix} \frac{x_1}{(x_1^2+x_2^2)^{3/2}} \\ \frac{x_2}{(x_1^2+x_2^2)^{3/2}} \\ 0 \end{bmatrix}. \tag{4}$$

Use initial values $x_{(0)} = \begin{bmatrix}0.0,\ 1.0,\ 0.1\end{bmatrix}^T$, $v_{(0)} = \begin{bmatrix}0.09,\ 0.05,\ 0.20\end{bmatrix}^T$.

3.2 Tokamak magnetic field (adapted from [3])

$$B(x) = \begin{bmatrix} -\dfrac{2x_2 + x_1x_3}{2(x_1^2 + x_2^2)} \\ \dfrac{2x_1 - x_2x_3}{2(x_1^2 + x_2^2)} \\ \dfrac{\sqrt{x_1^2 + x_2^2} - 1}{2\sqrt{x_1^2 + x_2^2}} \end{bmatrix}, \qquad A(x) = \begin{bmatrix} 0 \\ \frac{1}{2}\Big(x_1 - \ln(\sqrt{x_1^2 + x_2^2} + x_1)\Big) \\ -\frac{1}{2}\Big(x_3 \arctan(x_2, x_1) + \ln(x_1^2 + x_2^2)\Big) \end{bmatrix}. \tag{5}$$

Let $U(x) = \nabla U(x) = 0$ and use initial values $x_{(0)} = \begin{bmatrix}1.2,\ 0.0,\ 0.0\end{bmatrix}^T$, $v_{(0)} = \begin{bmatrix}0,\ 4.816 \cdot 10^{-4},\ -2.059 \cdot 10^{-3}\end{bmatrix}^T$.

4 Methods to implement

Implement all three methods below such that they can be used on both systems in Sections 3.1–3.2.

4.1 Boris method [1]

A second order method for (1) on a staggered grid (position and velocity are computed at different points in time). Given $(x_{(n)}, v_{(n-1/2)})$ and step size h:

$$v^- = v_{(n-1/2)} - \tfrac{h}{2}\nabla U(x_{(n)}), \qquad t = \tfrac{h}{2}B(x_{(n)}), \tag{6}$$
$$v^+ = v^- + v' \times s, \qquad s = \frac{2t}{1 + t \cdot t}, \tag{7}$$
$$v_{(n+1/2)} = v^+ - \tfrac{h}{2}\nabla U(x_{(n)}), \qquad v' = v^- + v^- \times t, \tag{8}$$
$$x_{(n+1)} = x_{(n)} + h\, v_{(n+1/2)}. \tag{9}$$

If only $(x_{(0)}, v_{(0)})$ is given, initialize $v_{(-1/2)}$ by a half-step backward using another method (e.g. `solve_ivp`). We can find the velocity at integer steps by averaging $v_{(n)} = \frac{1}{2}(v_{(n+1/2)} + v_{(n-1/2)})$.

4.2 Classical Runge–Kutta 4 on the extended system

On (1), define $f(x, v) = v \times B(x) - \nabla U(x)$. A time step from $(x_{(n)}, v_{(n)})$ with step size h is given by:

$$\begin{aligned}
k_1^v &= h\, f(x_{(n)}, v_{(n)}), & k_1^x &= h\, v_{(n)},\\
k_2^v &= h\, f(x_{(n)} + \tfrac{1}{2}k_1^x,\ v_{(n)} + \tfrac{1}{2}k_1^v), & k_2^x &= h(v_{(n)} + \tfrac{1}{2}k_1^v),\\
k_3^v &= h\, f(x_{(n)} + \tfrac{1}{2}k_2^x,\ v_{(n)} + \tfrac{1}{2}k_2^v), & k_3^x &= h(v_{(n)} + \tfrac{1}{2}k_2^v),\\
k_4^v &= h\, f(x_{(n)} + k_3^x,\ v_{(n)} + k_3^v), & k_4^x &= h(v_{(n)} + k_3^v),\\
v_{(n+1)} &= v_{(n)} + \tfrac{1}{6}\,(k_1^v + 2k_2^v + 2k_3^v + k_4^v),\\
x_{(n+1)} &= x_{(n)} + \tfrac{1}{6}\,(k_1^x + 2k_2^x + 2k_3^x + k_4^x).
\end{aligned}$$

4.3 Fourth-order symmetric multistep method [2, Section 4]

This is an explicit 9-point method for solving Equation (2). Let the five-point central difference centered in $y_{(j)}$ be

$$\Delta y_{(j)} := \tfrac{1}{12}\big(y_{(j-2)} - 8y_{(j-1)} + 8y_{(j+1)} - y_{(j+2)}\big).$$

Choose parameters $a_1 = -0.7, a_2 = 0.1, a_3 = 0.9$ as specified in [2] and use β and α coefficients given by

$$\begin{aligned}
\beta_0 &= \frac{1}{3}\,(20a_1a_2a_3 - 4(a_1a_2 + a_1a_3 + a_2a_3) - 28(a_1 + a_2 + a_3) - 52)\\
\beta_1 &= \frac{1}{3}\,(2a_1a_2a_3 + 14(a_1a_2 + a_1a_3 + a_2a_3) + 26(a_1 + a_2 + a_3) + 38)\\
(\alpha_{-4}, \ldots, \alpha_4) &= \left(1, -\frac{7}{5}, \frac{9}{25}, \frac{22}{125}, -\frac{34}{125}, \frac{22}{125}, \frac{9}{25}, -\frac{7}{5}, 1\right).
\end{aligned}$$

At time t_j for $j = n-4, \ldots, n+4$, let $x_{(j)} \approx x(t_j)$ with $t_j = t_0 + jh$. With

$$\begin{aligned}
v_{(j)} &:= \frac{\Delta x_{(j)}}{h}, \qquad A_j^\Delta := \frac{\Delta A(x_{(j)})}{h} = \frac{1}{12h}\big(A(x_{(j-2)}) - 8A(x_{(j-1)}) + 8A(x_{(j+1)}) - A(x_{(j+2)})\big)\\
F_j &:= A'(x_{(j)})^\top v_{(j)} - A_j^\Delta - \nabla U(x_{(j)}).
\end{aligned}$$

Note that you need to first compute the Jacobian $A'(x)$ either analytically or using automatic differentiation. The update for $x_{(n+4)}$ could be found by

$$\sum_{j=-4}^{4} \alpha_j\, x_{(n+j)} = h^2\Big(\beta_1 F_{n-1} + \beta_0 F_n + \beta_1 F_{n+1}\Big).$$

Starting values: You need $x_{(0)}, \ldots, x_{(7)}$. Generate them with a high-accuracy one-step method (e.g. `solve_ivp` RK45 with `rtol=1e-10`, `atol=1e-12`) on (1).

Remark: This method uses the vector potential $A(x)$ and its Jacobian $A'(x)$ instead of the magnetic field $B(x)$. For the Tokamak field, one will run into problems if $x_2 = 0$ and $x_1 \le 0$ due to the logarithm in $A(x)$. However, with the given initial values, the particle should not cross this singularity.

5 Experiments & deliverables

Run the following for both systems in Sections 3.1–3.2.

A. Plotting trajectories and error over long time

Implement all three methods such that they can be used on both systems in Sections 3.1–3.2. Run simulations up to time $T = 10^5$ with step size $h = 0.1$. Produce the following plots:

1. Plot the trajectories of the two first coordinates of the solutions, (x_1, x_2) on the x, y-axis, over time, for the different methods for both systems. For the Tokamak system, also plot $R = \sqrt{x_1^2 + x_2^2}$ on the x-axis and x_3 on the y-axis. You should then see something called a *banana orbit*, which means the particle is confined in a certain region of space. See for instance Figure 6 in [3] as a reference.
2. Plot the error over time for all methods with $\|x_{(n)}^{\text{method}} - x_{\text{ref}}(t_n)\|_2$ on the y axis and t_n on the x-axis. Use log scale on the y-axis (`plt.semilogy()`). Compute the reference solution for instance using `solve_ivp` from `scipy.integrate` (RK45, tolerances around 10^{-12}).
3. Plot the change in energy (Equation (3)) over time: $e_n = |E(x_{(n)}, v_{(n)}) - E(x_{(0)}, v_{(0)})|$ against time for each method, for one choice of step size h_i, and use log scale on the y-axis (`plt.semilogy()`). For the symmetric multistep method, see if you obtain the expected result stated in Theorem 3.1 of [2].

Comment on the results and if you observed the expected behaviour the different methods. If not, try to explain why.

B. Order verification and work precision diagrams

For decreasing step sizes $h_i = \frac{h}{2^i}$, $i = 0, 1, 2$ with $h = 0.1$, compute global errors

$$e_i^{\text{method}} = \|x_{(N_i)}^{\text{method}} - x_{(N_i)}^{\text{ref}}\|_2,$$

at a fixed end-time $T = h_i N_i$, with $x_{(N_i)} \approx x(T)$, by comparison with a reference solution. You may use shorter time intervals in this exercise. Use `time.perf_counter()` to measure the time τ_i^{method} each method uses to compute a numerical solution with step size h_i. Produce the following plots:

1. Plot $(h_i, e_i^{\text{method}})$ in a log–log plot (`plt.loglog()`) for each method against reference lines (h_i, h_i^p) with $p = 2, 4$ to see if the expected orders are achieved.
2. Make a log–log plot of error vs. time, $(e_i^{\text{method}}, \tau_i^{\text{method}})$ for each method. This is sometimes called a work–precision diagram. Which method is most efficient for a given error tolerance?

Comment on the results and if you managed to verify the expected orders of convergence. Was the efficiency results as expected?

C. Exploratory exercise (Counts 20% of the subproject score)

Do something you find interesting related to the project. For instance, you could try to implement other methods (e.g. implicit midpoint, Stormer–Verlet, other Boris variants, higher-order symmetric multistep methods), or investigate long-time conservation of other invariants (e.g. momentum as mentioned in [2]). Another interesting numerical experiment is to verify numerically that the symmetric multistep method is actually symmetric. You could try to reproduce Figure 6 in [3] (the breakdown of RK4) by using the initial conditions given there and comparing only Boris against RK4. You could also explore different initial values or other magnetic fields or electrostatic potentials. Discuss your findings and motivate why the given experiment is interesting in this setting.

References

[1] J. P. Boris et al. Relativistic plasma simulation-optimization of a hybrid code. In *Proc. Fourth Conf. Num. Sim. Plasmas*, pages 3–67, 1970.

[2] E. Hairer and C. Lubich. Symmetric multistep methods for charged-particle dynamics. *The SMAI Journal of computational mathematics*, 3:205–218, Oct. 2017. ISSN 2426-8399. doi: 10.5802/smai-jcm.25. URL https://smai-jcm.centre-mersenne.org/articles/10.5802/smai-jcm.25/.

[3] Y. He, Y. Sun, J. Liu, and H. Qin. Volume-preserving algorithms for charged particle dynamics. *Journal of Computational Physics*, 281:135–147, Jan. 2015. ISSN 00219991. doi: 10.1016/j.jcp.2014.10.032. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999114007141.