Project 1
Numerical ODEs

1 Introduction

We've seen in class how systems of ODEs can be solved numerically on computer. In this project, you will explore a few numerical methods for solving ODEs. You will be asked to implement these methods in Python and use them to analyze famous equations in physics. The methods you will implement fall under the the category of **Runge-Kutta** (RK) methods and **symplectic** methods:

1. Runge-Kutta Methods
   a. Euler (aka RK1)
   b. RK2
   c. RK4
2. Symplectic Methods
   a. Symplectic Euler (aka Euler-Cromer)
   b. Stormer-Verlet (Optional)

2 Background and notation

Recall that a system of ODEs can be thought of as an equation for a single vector-valued function of a single real variable. In class, we used $\mathbf{x}$ for this function, but to avoid confusion later when the function we are looking for involves some components which are positions and other components which are velocities, we will denote our unknown vector valued function by $\boldsymbol{\xi}$. This function takes a given "time" t as its input and outputs a vector $\boldsymbol{\xi}(t)$ in d real dimensions. In components, this can be written as

$$\boldsymbol{\xi}(t) = \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \\ \vdots \\ \xi_d(t) \end{pmatrix} \tag{1}$$

In this project, we consider numerically solving initial value problems for first-order systems of the form:

$$\dot{\boldsymbol{\xi}}(t) = \mathbf{w}(\boldsymbol{\xi}(t)), \qquad \boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0 \tag{2}$$

where the initial time t_0, the “velocity" function $\mathbf{w}$, and the initial “position" $\boldsymbol{\xi}_0$ are all given at the outset. I have put the words “velocity" and “position" in quotes to emphasize the fact that the components of the function $\boldsymbol{\xi}$ could represent any quantities that are time evolving according to the equation (2), not just spatial positions.

We uniformly use the following notation for the information that needs to be input into the numerical method in addition to the velocity function $\mathbf{w}$:

$$t_0 = \text{the initial time} \tag{3}$$
$$\boldsymbol{\xi}_0 = \text{the initial “position"} \tag{4}$$
$$h = \text{the step size} \tag{5}$$
$$N = \text{the number of steps} \tag{6}$$

we assume that the solution will then be generated on a mesh of times $(t_0, t_1, \dots, t_N)$ defined by

$$t_n = t_0 + nh \tag{7}$$

and the solution will consist of a sequence of positions $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_N)$ generated according to some algorithm, and approximating the positions of the exact solution $\boldsymbol{\xi}$ evaluated at the times in the mesh:

$$\boldsymbol{\xi}_n = \boldsymbol{\xi}(t_n). \tag{8}$$

3 Runge-Kutta Methods

The Runge-Kutta methods can be specified by how a given numerical position $\boldsymbol{\xi}_{n+1}$ is generated in terms of the previous numerical position $\boldsymbol{\xi}_n$.

3.1 Euler's Method

Euler's method assumes makes the crude approximation that between steps, the system moves with constant velocity.

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + h \cdot \mathbf{w}(\xi_n), \tag{9}$$

To show how Euler's method is like RK2 and RK4 below, we can also describe it as first computing the following variable at a given step:

$$\mathbf{k}_{1,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n) \tag{10}$$

and then using this variable to step forward in the following way:

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + \mathbf{k}_{1,n} \tag{11}$$

where we have defined $\mathbf{w}_n = \mathbf{w}(\xi_n)$. Euler's method can be shown to be a first-order method.

3.2 RK2

RK2 Uses information from both the last step and halfway between the last step and the current step to move forward. For each step, we first compute two intermediate variables:

$$\mathbf{k}_{1,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n) \tag{12}$$
$$\mathbf{k}_{2,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n + \tfrac{1}{2}\mathbf{k}_1) \tag{13}$$

then we use these variables to step forward in the following way:

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + \mathbf{k}_{2,n} \tag{14}$$

RK2 can be shown to be a second-order method.

3.3 RK4

RK4 is like RK2, but is computes more intermediate information before stepping forward. We first compute the following variables at a given step:

$$\mathbf{k}_{1,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n) \tag{15}$$
$$\mathbf{k}_{2,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n + \tfrac{1}{2}\mathbf{k}_1) \tag{16}$$
$$\mathbf{k}_{3,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n + \tfrac{1}{2}\mathbf{k}_2) \tag{17}$$
$$\mathbf{k}_{4,n} = h \cdot \mathbf{w}(\boldsymbol{\xi}_n + \mathbf{k}_3) \tag{18}$$

then we use these variables to step forward in the following way:

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + \frac{1}{6}\mathbf{k}_{1,n} + \frac{1}{3}\mathbf{k}_{2,n} + \frac{1}{3}\mathbf{k}_{3,n} + \frac{1}{6}\mathbf{k}_{4,n} \tag{19}$$

RK4 can be shown to be a fourth-order method.

4 Symplectic Methods

Symplectic ODE solvers are particularly well-suited to computing solutions to differential equations for Hamiltonian Systems like those we encounter in Hamiltonian mechanics. In such cases, the evolution of the system is described motion of a point in the **phase space** of the system: the set of all admissible $(2k)$-dimensional vectors

$$\boldsymbol{\xi} = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_k \\ q_1 \\ \vdots \\ q_k \end{pmatrix} = \begin{pmatrix} \xi_1 \\ \xi_2 \vdots \\ \xi_{2k} \end{pmatrix} \tag{20}$$

of (generalized) coordinates and their corresponding canonical momenta. The evolution of Hamiltonian systems is governed by Hamilton's equations which are coupled, first-order differential equations for the coordinates and momenta;

$$\dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}} \qquad \dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}. \tag{21}$$

If we define the **symplectic matrix** J as

$$J = \begin{pmatrix} 0 & I_k \\ -I_k & 0 \end{pmatrix} \tag{22}$$

where I_k is the $k \times k$ identity matrix, and if we define

$$\nabla H = \begin{pmatrix} \frac{\partial H}{\partial p_1} \\ \vdots \\ \frac{\partial H}{\partial p_k} \\ \frac{\partial H}{\partial q_1} \\ \vdots \\ \frac{\partial H}{\partial q_k} \end{pmatrix} = \begin{pmatrix} \frac{\partial H}{\partial \xi_1} \\ \frac{\partial H}{\partial \xi_2} \\ \vdots \\ \frac{\partial H}{\partial \xi_{2k}} \end{pmatrix} \tag{23}$$

then we can write Hamilton's equations in the form of our general first order system (2):

$$\dot{\boldsymbol{\xi}}(t) = J^T \nabla H(\boldsymbol{\xi}(t)). \tag{24}$$

It's possible to show that the time evolution that results from Hamilton's equation is **symplectic**. Here's what that means. For each time t we define a transformation Φ_t on phase space as the function that time evolves the initial state $\boldsymbol{\xi}(t)$ of the system to its state $\boldsymbol{\xi}(t)$ at time t, namely

$$\Phi_t(\boldsymbol{\xi}_0) = \boldsymbol{\xi}(t) \tag{25}$$

The Jacobian matrix

$$\Phi_t' = \begin{pmatrix} \frac{\partial(\Phi_t)_1}{\partial\xi_1} & \cdots & \frac{\partial(\Phi_t)_1}{\partial\xi_{2k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial(\Phi_t)_{2k}}{\partial\xi_1} & \cdots & \frac{\partial(\Phi_t)_{2k}}{\partial\xi_{2k}} \end{pmatrix} \tag{26}$$

satisfies

$$(\Phi_t')^T J \Phi_t' = J \tag{27}$$

for all times t. Another way of saying this is that Φ_t' preserves (or leaves invariant) the symplectic matrix J. A numerical ODE solver is **symplectic** provided it has the property that just like the true time-evolution, it preserves the symplectic matrix J This means that if ϕ is the transformation that evolves the state of the system from one discrete point in phase space to the other,

$$\phi(\boldsymbol{\xi}_n) = \boldsymbol{\xi}_{n+1} \tag{28}$$

then

$$(\phi')^T J \phi' = J. \tag{29}$$

This property turns out to make symbplectic solvers significantly more stable than many other ODE solvers when applied to Hamiltonian systems. In this project, we consider only Hamiltonian systems for which the Hamiltonian can be written in the form

$$H(\mathbf{p}, \mathbf{q}) = T(\mathbf{p}) + V(\mathbf{q}). \tag{30}$$

In this case, Hamilton's equations reduce to

$$\dot{\mathbf{p}} = -\frac{\partial V}{\partial \mathbf{q}}, \qquad \dot{\mathbf{q}} = \frac{\partial T}{\partial \mathbf{p}} \tag{31}$$

4.1 Symplectic Euler

There are two variants of the so-called symplectic Euler method. We'll call the first one **SE1**, and it works as follows:

$$\mathbf{p}_{n+1} = \mathbf{p}_n - h \cdot \frac{\partial V}{\partial \mathbf{q}}(\mathbf{q}_n) \tag{32}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h \cdot \frac{\partial T}{\partial \mathbf{q}}(\mathbf{p}_{n+1}) \tag{33}$$

Notice that unlike with Runge-Kutta methods, the order of these equations matters since the second "position update" equation uses $\mathbf{p}_{n+1}$ from the first "momentum update" equation. The second variant of symplectic Euler, which we'll call **SE2** is different essentially only in that it updates position first and then momentum:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h \cdot \frac{\partial T}{\partial \mathbf{q}}(\mathbf{p}_n) \tag{34}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_n - h \cdot \frac{\partial V}{\partial \mathbf{q}}(\mathbf{q}_{n+1}) \tag{35}$$

Both symplectic Euler methods are first order methods.

4.2 Stormer-Verlet (Optional)

If you are interested in learning about this method, you are encourages to look it up. It's a second-order symplectic solver.

5 Assignments

Tip: When you're solving ODEs numerically on computer, you can often vastly simplify your work by first non-dimensionalizing any system of equations you encounter before implementing them on computer. This means normalizing time, and the components of ξ in such a way that all terms in the equations become dimensionless.

1. **The harmonic oscillator** The one-dimensional harmonic oscillator is described by the equation

$$\ddot{x}(t) = -\omega^2 x(t), \qquad \omega > 0. \tag{36}$$

   a. Determine the general solution to the initial value problem for the harmonic oscillator with initial conditions

$$x(0) = x_0, \qquad \dot{x}(0) = 0, \qquad \dot{x}_0 \neq 0. \tag{37}$$

   Comment on what the general solution looks like in phase space, and make a few phase space plots for some fixed ω and various x_0.

   b. Write down the Hamiltonian and derive Hamilton's equations for the simple harmonic oscillator of mass m and frequency ω.

   c. Solve the initial value numerically using Euler's Method, RK2, RK4, SE1, and SE2 for various step sizes. Make both plots of the position as a function of time and phase space plots comparing the numerical solution to the exact solution. Comment on your results.

d. Check energy conservation for each solver by graphing total energy (kinetic plus potential) versus time and comment on your results.

2. **The two-body problem**

   a. Write down the system of equations satisfied by planets interacting gravitationally according to Newton's Universal Law of Gravitation (the two-body problem), and show in the standard way (see your analytic mechanics textbook) that the system of equations can be reduced to that of a single body executing a planar orbit around a fixed force center at the origin.

   b. Show how the reduced two-body problem is solved exactly, and write down explicit expressions for the closed orbits.

   c. Write down the Hamiltonian and the corresponding Hamilton's equations for the reduced two-body problem.

   d. Show that Kepler's Second Law (equal areas in equal times) is equivalent to angular momentum conservation.

   e. Use either SE1 or SE2 to numerically solve the reduced two-body problem for various initial conditions that form closed orbits, and plot the orbits against the corresponding exact orbits.

   f. Confirm Kepler's Second and Third Laws using your numerical solutions. Note that since the Second Law is equivalent to angular momentum conservation, its confirmation reduces to confirming angular momentum conservation. To confirm the 3rd law, determine the time period T and the semi-major axis a of the ellipse a for each numerically obtained orbit. Then, plot $\ln T$ versus $\ln a$ for various initial conditions. What should the slope be? Verify that the plot has the correct slope using linear regression.

3. **Further Investigation (Optional)**

   a Prove that the symplectic Euler methods are symplectic.

   b Prove that the symplectic Euler methods exactly conserve angular momentum in the reduced two-body problem.

   c Numerically study a modification of the two-body problem in which the Gravitational force deviates from an inverse square force by a small amount. Do the orbits close in this case? Refer to Bertrand's Theorem, and compare the statement of this theorem with your results.

   d Read up on "backward error analysis" and investigate how it can be used to gain insight into why the symplectic Euler methods are so stable over such long periods of time (something you should have noticed while completing the assignments above.