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 $(t_0, t_1, \ldots, t_N)$ defined by

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

and the solution will consist of a sequence of positions $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_N)$ 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:

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

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

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:

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

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

Both symplectic Euler methods are first order methods.

4.2 Stormer-Verlet

This is a second-order symplectic solver abbreviated as SV.

$$\mathbf{p}_{n+1/2} = \mathbf{p}_n - \frac{h}{2} \cdot \frac{\partial V}{\partial \mathbf{q}}(\mathbf{q}_n) \tag{36}$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h \cdot \frac{\partial T}{\partial \mathbf{p}}(\mathbf{p}_{n+\frac{1}{2}}) \tag{37}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_{n+1/2} - \frac{h}{2} \cdot \frac{\partial V}{\partial \mathbf{q}}(\mathbf{q}_{n+1}) \tag{38}$$

5 The Kepler Problem

In the 17th century Johannes Kepler formulated the following three statements for planetary motion:

1. Planets move in elliptical orbits with the Sun at one focus of the ellipse.
2. The amount of area per unit time swept out by a line joining a planet to the Sun is constant.
3. The period of the orbit T is proportional to $a^{3/2}$, where a is the semi-major axis of the ellipse.

Let us take a closer look at the Kepler problem starting with the equations of motion for a planet in the gravitational potential of a sun. The gravitational force motion of two bodies can always be reduced to an equivalent one-body problem by writing down the equations in the center-of-mass frame (see for example Ref. [**?**]). Let us consider the motion of a planet with mass m in the gravitational potential of a sun with mass M (Fig. 1). The equations of motion for the reduced problem are

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= -\frac{G(M+m)\,\mathbf{r}}{r^3}, \end{aligned} \tag{39}$$

where $\mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1$ is the position of the planet relative to the position of the sun, and G is the gravitational constant.

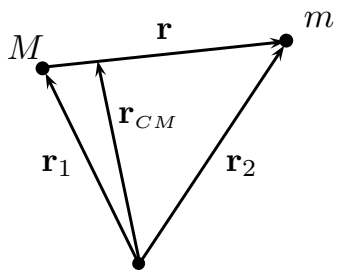

Figure 1: Various coordinates for the Kepler problem. The vectors $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_{CM}$, and $\mathbf{r}$ denote the positions of the sun, the planet, the center of mass, and the relative distance, respectively.

The Kepler problem has four constants of motion: three components of the angular momentum

$$\mathbf{L} = \mathbf{r} \times \mu\mathbf{v} \tag{40}$$

and the total energy

$$E = \frac{\mu v^2}{2} - \frac{G\mu(M+m)}{r}, \tag{41}$$

where $\mu = Mm/(M+m)$ is the reduced mass. The conservation of $\mathbf{L}$ is in fact equivalent to Kepler's second law and it also implies that the motion is restricted to a plane perpendicular to the direction of $\mathbf{L}$. The constants of motion can be used to obtain r as a function of the polar angle θ in the plane of motion [**?**]:

$$r\big(\theta(t)\big) = \frac{C}{1 - \epsilon\cos\big(\theta(t) - \theta_0\big)}. \tag{42}$$

The above expression defines a conic (a curve obtained from the intersection of a cone with a plane). The parameter ϵ is known as the *eccentricity*. The orbit type depends on ϵ according to the following rule:

- $\epsilon = 0$: circle,
- $\epsilon < 1$: ellipse,
- $\epsilon = 1$: parabola,
- $\epsilon > 1$: hyperbola.

Inverting the relation between θ and t to determine the motion as a function of time is generally not an easy task. For this purpose it is usually more convenient to integrate the equations of motion (39) numerically.

References

[1] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics.* Addison Wesley, San Francisco, 3rd edition, 2002.