Physics 305: Ordinary differential equations (ODEs)

Vasileios Paschalidis[1]
[1] *Departments of Astronomy & Physics, University of Arizona, Tucson*

In this note we discuss methods for numerically solving ODEs.

I. ODES IN PHYSICS

Physics equations of motion require the solution of ordinary differential equations. While the vast majority of equations in Physics are partial differential equations, ordinary differential equations pop-up in scenarios where a high degree of symmetry and/or stationarity (time-independence) is imposed (e.g., stationary and spherically symmetric). For example, in Newtonian gravity the equation for the gravitational potential is

$$\nabla^2\Phi = 4\pi G\rho, \tag{1}$$

but in spherical symmetry this becomes an ODE

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\Phi\right) = 4\pi G\rho. \tag{2}$$

The Newtonian equation of motion is also a second-order ODE in time (acceleration=force/mass),

$$\frac{d^2x}{dt^2} = F/m \tag{3}$$

For a very complex density distribution or complicated (time-dependent) forces, these last equations are not easily solved analytically. So, in such cases we need to solve the ODEs numerically. There are multiple ways one can numerically solve ODEs, and we will review here some of these methods including the Euler, and Runge-Kutta methods. We will also discuss the distinction between initial value and boundary value problems.

II. INITIAL VALUE PROBLEMS: THEORETICAL BACKGROUND

An initial value problem (IVP) for ODEs is described by the following equations

$$\frac{dy}{dt} = f(t,y), a \le t \le b, \text{ Initial conditions: } y(a) = \alpha \tag{4}$$

for a single ODE, and

$$\frac{d\vec{y}}{dt} = \vec{F}(t,y), a \le t \le b, \text{ Initial conditions: } \vec{y}(a) = \vec{\alpha}, \tag{5}$$

where

$$\vec{y} = (y_1, y_2, \ldots y_n), \ \vec{F} = (f_1(t,\vec{y}), f_2(t,\vec{y}) \ldots f_n(t,\vec{y})). \tag{6}$$

for a system of coupled ODEs.

Before we decide to code any differential equation in a computer it is always important to be able to answer the following questions: 1) Does the ODE have a solution? 2) Is the ODE solution unique for a given initial conditions? 3) Does the solution depend continuously on the initial conditions? All three questions point to the concept of well-posedness of the IVP. If the answer is NO to any of these 3 questions, the IVP is not well-posed, and it is redundant to try to solve the ODE as an IVP, as the numerical algorithm will fail. In this first section we will introduce the mathematical background for answering the above questions.

DEFINITIONS

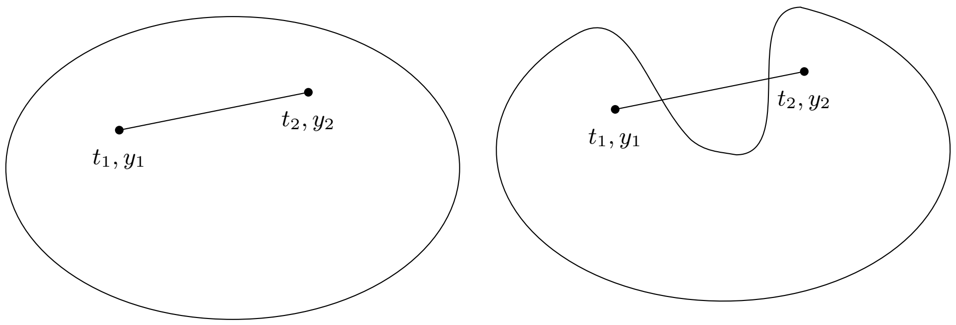

FIG. 1. Left: depiction of a convex set. Right: depiction of a non-convex set.

1. We say that $f(t,y)$ satisfies a <u>Lipschitz condition</u> (or is Lipschitz continuous) in the variable y on a set $D \subset \mathbb{R}^2$, if a constant $L > 0$ exists with $|f(t,y_1) - f(t,y_2)| \leq L|y_1 - y_2|$.

   For example, $f(t,y) = ty$, $D = \{(t,y)|1 \leq t \leq 2, -3 \leq y \leq 4\}$, then for (t,y_1), (t,y_2) we have $|f(t,y_1) - f(t,y_2)| = |t||y_1 - y_2| \leq 2|y_1 - y_2|$. Thus, $L = 2$ and $f(t,y) = ty$ is Lipschitz continuous.

2. A set $D \subset \mathbb{R}$ is called <u>convex</u>, if for (t_1,y_1) and $(t_2,y_2) \in D$, then $[(1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2] \in D \; \forall \lambda \in [0,1]$

Theorem 1: If $f(t,y)$ is defined on a convex set $D \subset \mathbb{R}^2$, and there exists a postive $L > 0$ such that $\left|\frac{\partial f}{\partial y}(t,y)\right| \leq L$ $\forall (t,y) \in D$, then f is Lipschitz continuous on D, with Lipschitz constant L.

Theorem 2: If $D = \{(t,y) \mid a \leq t \leq b, -\infty < y < \infty\}$, and $f(t,y)$ is continuous on D, and $f(t,y)$ satisfies a Lipschitz condition on D in y, then the IVP

$$\frac{dy}{dt} = f(t,y), a \leq t \leq b, \text{ Initial conditions: } y(a) = \alpha \tag{7}$$

has a *unique* solution $y(t)$ for $a \leq t \leq b$.

For example, the IVP

$$\frac{dy}{dt} = f(t,y) = 1 + t\sin(ty), 0 \leq t \leq 2, \text{ Initial conditions: } y(a) = \alpha \tag{8}$$

satisfies

$$\frac{\partial f}{\partial y} = t^2\cos(ty) \leq 4, \tag{9}$$

Thus, by theorems 1 and 2, the IVP with $f(t,y) = 1 + t\sin(ty)$ has a unique solution.

Well-posedness of the IVP

The IVP

$$\frac{dy}{dt} = f(t,y), a \leq t \leq b, \text{ Initial conditions: } y(a) = \alpha \tag{10}$$

is *well-posed* if

- A unique solution $y(t)$ exists
- There exist constants $\epsilon_0 > 0$, $k > 0$ such that $\forall \epsilon > 0$ with $\epsilon_0 > \epsilon > 0$, whenever $\delta(t)$ is continuous with $|\delta(t)| < \epsilon$ $\forall t \in [a,b]$, and $\delta_0 < \epsilon$, the IVP

$$\frac{dz}{dt} = f(t,z) + \delta(t), a \leq t \leq b, \text{ Initial conditions: } z(a) = \alpha + \delta_0 \tag{11}$$

  has a unique solution $z(t)$ that satisfies

$$|z(t) - y(t)| < k\epsilon_0, \; \forall t \in [a,b]. \tag{12}$$

Theorem 3: Assume $D = \{(t, y) \mid a \le t \le b, -\infty < y < \infty\}$, where $f(t, y)$ is continuous and is Lipschitz continuous in y on D, then the IVP

$$\frac{dy}{dt} = f(t, y), a \le t \le b, \text{ Initial conditions: } y(a) = \alpha \tag{13}$$

is well-posed.

Well-posedness is a fundamental property of initial value problems for ordinary and partial differential equations. If a differential equation does not admit a well-posed initial value problem, then we cannot integrate it numerically. This is because, well-posedness is intimately related with the "stability" of the solution as stated in Eq. (12). This equation implies that the perturbed initial value problem must have a solution that is very close to the unperturbed one, i.e., the absolute difference between the solutions $z(t)$ and $y(t)$ is bounded. If a small perturbation either in initial conditions or in the right-hand-side of Eq. (11) leads the difference between the solution $z(t)$ and $y(t)$ to grow without bound, then this problem is ill-posed and cannot be integrated in a computer. The reason is that computers will always have finite error, i.e., will always apply small perturbations in the form of either truncation or round-off error. More specifically well-posedness implies continuous dependence on the initial data.

III. INITIAL VALUE PROBLEMS: NUMERICAL ALGORITHMS

We will now investigate numerical algorithms for solving IVPs for ODEs using first-order and higher-order accurate methods.

A. Euler's Method

For an IVP

$$\frac{dy}{dt} = f(t, y), a \le t \le b, \ y(a) = \alpha \tag{14}$$

We seek a solution $y(t)$ at various values of $t \in [a, b]$ called the mesh points. The solution at other t can be obtained by interpolation. Consider the following mesh points

$$t_i = a + ih, \ i = 0, 1, 2, \ldots, N, \tag{15}$$

where $h = (b-a)/N = t_{i+1} - t_i$ is called the step size. The simplest possible method for solving an IVP is the forward Euler method, which is derived from Taylor's theorem

$$y(t_{i+1}) = y(t_i) + h\frac{dy}{dt}\bigg|_{t_i} + \frac{h^2}{2}y''(\xi) \Rightarrow \tag{16}$$

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}y''(\xi) \tag{17}$$

Dropping the 2nd-order term, we have then the following algorithm

$$\begin{aligned} y_0 =& \alpha \\ y_{n+1} =& y_n + hf(t_n, y_n), \ n = 0, 1, 2, \ldots N-1. \end{aligned} \tag{18}$$

The last equation is called a difference equation, and we denote $y_n = y(t_n)$.

Exercise 1: Consider the following IVP

$$\frac{dy}{dt} = -y, 0 \le t \le 3, \ y(0) = 1. \tag{19}$$

Is the problem well-posed? Use Euler's method to solve the IVP for $N = 100, N = 500, N = 1000, N = 5000, N = 10000$ and compare $y(3)$ with the analytic (exact) solution. How quickly does the relative error decay as a function of h?

B. Higher-order Taylor methods

Euler is only a first-order accurate method and is not used in practise. There are higher-order accurate methods that are preferred instead. Before we discuss standard methods we will first overview higher-order Taylor methods.

Consider the IVP

$$y' = f(t, y), a \leq t \leq b,\ y(a) = \alpha \tag{20}$$

where a prime $'$ implies differentiation with respect to t. Expand

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \ldots + \frac{h^k}{n!}y^{(k)}(t_n) + \frac{h^{k+1}}{(k+1)1}y^{(k+1)}(\xi_n),\ \xi_n \in (t_n, t_{n+1}). \tag{21}$$

But, from the ODE

$$y'(t) = f(t, y(t)), y''(t) = f'(t, y(t)), \ldots y^{(k)}(t) = f^{(k-1)}(t, y(t)) \tag{22}$$

By use of Eq. (22), Eq. (21) becomes

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}f'(t_n, y(t_n)) + \ldots + \frac{h^k}{n!}f^{(k-1)}(t_n, y(t_n)) + \frac{h^{k+1}}{(k+1)1}f^{(k)}(\xi_n, y(\xi_n)). \tag{23}$$

Equation (23) motivates the algorithm of the Taylor method of order m, i.e.,

$$\begin{aligned} y_0 =& \alpha \\ y_{n+1} =& y_n + hT^{(m)}(t_n, y_n),\ n = 0, 1, \ldots N-1, \end{aligned} \tag{24}$$

where

$$T^{(m)}(t_n, y_n) = f(t_n, y_n) + \frac{h}{2}f'(t_n, y_n) + \ldots + \frac{h^{(m-1)}}{m!}f^{(m-1)}(t_n, y_n) \tag{25}$$

Euler is a Taylor method of order 1.

C. Runge-Kutta Methods

Runge-Kutta (RK) methods are widespread, efficient, and almost the standard. We will derive the second-order accurate RK2 method, and simply present the 3rd (RK3) and 4th-order accurate RK4 methods.

First, let us consider Taylor's theorem in 2 variables. Let $f(t, y)$ be C^{n+1} on $D = \{(t, y) \mid a \leq t \leq b,\ c \leq y \leq d\}$ and $(t_0, y_0) \in D$. Then $\forall\ (t, y) \in D$, there exists ξ between (t, t_0) and (y, y_0) such that

$$f(t, y) = P_n(t, y) + R_n(t, y), \tag{26}$$

where the Taylor polynomial of degree n is

$$\begin{aligned} P_n(t) =& f(t_0, y_0) + \left[(t - t_0)\frac{\partial f}{\partial t}(t_0, y_0) + (y - y_0)\frac{\partial f}{\partial y}(t_0, y_0)\right] \\ &+ \left[\frac{(t - t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0, y_0) + (t - t_0)(y - y_0)\frac{\partial^2 f}{\partial t \partial y}(t_0, y_0) + \frac{(y - y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0, y_0)\right] \\ &+ \ldots \left[\frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}(t - t_0)^{n-j}(y - y_0)^j\frac{\partial^n f}{\partial t^{n-1}\partial y^j}(t_0, y_0)\right] \end{aligned} \tag{27}$$

and the remainder is

$$R_n(t, y) = \frac{1}{(n+1)!}\sum_{j=0}^{n+1}\binom{n+1}{j}(t - t_0)^{n+1-j}(y - y_0)^j\frac{\partial^{(n+1)} f}{\partial t^{n+1-j}\partial y^j}(\xi, \mu), \tag{28}$$

for some $\xi \in (t_0, t)$, and $\mu \in (y_0, y)$.

RK2

We wish to find a_1, α_1, β_1 such that $a_1 f(t+\alpha_1, y+\beta_1)$ approximates $T^{(2)}$, i.e.,

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2} f'(t,y) \tag{29}$$

with no error greater than $O(h^2)$. Since,

$$f'(t,y) = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt}, \text{ and } \frac{dy}{dt} = f, \tag{30}$$

we have

$$T^{(2)}(t,y) = f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t} + \frac{h}{2}\frac{\partial f}{\partial y} \cdot f. \tag{31}$$

We also have from the Taylor expansion in 2 variables

$$a_1 f(t+\alpha_1, y+\beta_1) = a_1 f(t,y) + a_1\alpha_1 \frac{\partial f}{\partial t} + a_1\beta_1 \frac{\partial f}{\partial y} + a_1 \underbrace{R_1(t+a_1, y+\beta_1)}_{\text{residual}} \tag{32}$$

where the residual is $O(h^2)$ if 2nd-order derivatives of f are bounded. If we match Eq. (32) (without its residual) to Eq. (31) we obtain the following set of algebraic equations

$$\begin{aligned} a_1 &= 1 \\ a_1\alpha_1 &= \frac{h}{2} \\ a_1\beta_1 &= \frac{h}{2} f(t,y) \end{aligned} \tag{33}$$

that is

$$a_1 = 1, \ \alpha_1 = \frac{h}{2}, \ \beta_1 = \frac{h}{2} f(t,y). \tag{34}$$

Thus,

$$T^{(2)}(t,y) = f\left(t + \frac{h}{2}, y + \frac{h}{2} f(t,y)\right) - R_1 \tag{35}$$

We have therefore arrived at the RK2 algorithm

$$\begin{aligned} y_0 &= \alpha \\ y_{n+1} &= y_n + hf\left(t_n + \frac{h}{2}, y_n + \frac{h}{2} f(t_n, y_n)\right), \ n = 0, 1, \ldots N-1, \end{aligned} \tag{36}$$

To avoid the nested functioning we can rewrite this as follows

$$\begin{aligned} y_0 &= \alpha \\ k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \\ y_{n+1} &= y_n + k_2, \ n = 0, 1, \ldots N-1, \end{aligned} \tag{37}$$

RK3

The RK3 (Heun's) algorithm is the following

$$\begin{aligned} y_0 =& \alpha \\ k_1 =& hf(t_n, y_n) \\ k_2 =& hf(t_n + \frac{h}{3}, y_n + \frac{1}{3}k_1) \\ k_3 =& hf(t_n + \frac{2h}{3}, y_n + \frac{2}{3}k_2) \\ y_{n+1} =& y_n + \frac{1}{4}(k_1 + 3k_3) \end{aligned} \tag{38}$$

for each $n = 0, 1, \dots N-1$. The method has local truncation error $O(h^3)$, as long as $y(t)$ is at least C^4.

RK4

The RK4 algorithm is the following

$$\begin{aligned} y_0 =& \alpha \\ k_1 =& hf(t_n, y_n) \\ k_2 =& hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \\ k_3 =& hf(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \\ k_4 =& hf(t_{n+1}, y_n + k_3) \\ y_{n+1} =& y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{39}$$

for each $n = 0, 1, \dots N-1$. The method has local truncation error $O(h^4)$, as long as $y(t)$ is at least C^5.

Algorithm for RK4:

Input: a, b, N, initial condition α, $f(t, y)$

- Step 1: Set $h = (b-a)/N$

  $t = a$

  $y = \alpha$

  output(t,y)

- Step 2: For $n = 1, 2, \dots N$ do Steps 3-5

- Step 3:

$$\begin{aligned} k_1 =& hf(t, y) \\ k_2 =& hf(t + h/2, y + k_1/2) \\ k_3 =& hf(t + h/2, y + k_2/2) \\ k_4 =& hf(t + h, y + k_3) \end{aligned} \tag{40}$$

- Step 4:

$$y = y + (k_1 + 2k_2 + 2k_3 + k_4)/6 \tag{41}$$

$$t = a + n \cdot h \tag{42}$$

- Step 5:

  output(t,y)

Exercise 2: Consider the following IVP

$$\frac{dy}{dt} = -2ty, 0 \le t \le 2,\ y(0) = 1. \tag{43}$$

Solve the ODE analytically. Follow the RK4 algorithm to implement instead the RK2 method and solve the IVP for $N = 100, N = 200, N = 400, N = 800, N = 1600$ and compare $y(3)$ with the analytic (exact) solution and the Euler solution. How quickly does the relative error decay as a function of h for RK2? Is RK2 more accurate than Euler? To make the code modular define a function for the right-hand-side of the equation, i.e., $f(t, y)$. Define a function RK2 that does the integration from t to $t + h$ and to which you will pass a function pointer that will point to $f(t, y)$. This way if you change $f(t, y)$, the code is invariant. The RK2 function will take as input t, h, the previous value of y, the pointer to the function and will return the new value of y, at $t + h$.

D. Higher order ODEs and systems of ODEs

Many ODEs in physics are of higher than first-order for which the methods we discussed above apply. For example, a simple harmonic oscillator satisfies

$$m\frac{d^2x}{dt^2} = -kx \tag{44}$$

Any higher order ODE which can be solved for the highest order derivative can be written as a system of first order ODEs. For example, to obtain a first-order form for Eq. (44) we let $y = dx/dt$, and then the equation becomes the system of two first-order ODEs for the functions $y(t)$ and $x(t)$

$$\begin{aligned}\frac{dx}{dt} &= y\\ \frac{dy}{dt} &= -\frac{k}{m}x\end{aligned} \tag{45}$$

Thus, a large set of higher-order ODEs can be cast to first-order form as a system of first-order ODEs. This is why we will discuss here how we can solve numerical systems of coupled first-order ODEs. Thus, we will here concern ourselves primarily with generalizing our methods to systems of first order ODEs. The IVP for a general first-order system of m equations can be written as

$$\frac{d\vec{u}}{dt} = \vec{F}(t, \vec{u}),\ a \le t \le b,\ \vec{u}(t = a) = \vec{\alpha}, \tag{46}$$

where the column vector of unknown functions $\vec{u}(t) = (u_1(t), u_2(t), u_3(t), \ldots u_m(t))$, the column vector of "sources" $\vec{F}(t, \vec{u}) = (f_1(t, u_1, u_2, \ldots, u_n), f_2(t, u_1, u_2, \ldots, u_n), \ldots, f_m(t, u_1, u_2, \ldots, u_n))$., and the m initial values $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m$. The goal is to find $\vec{u}(t)$. First, we need to generalize our definitions for *Lipschitz condition* and our theorems for uniqueness of solutions.

Definition: For a function $f(t, u_1, \ldots, u_m)$ defined on $D = \{(t, u_1, \ldots, u_m) \mid a \le t \le b,\ -\infty < u_i < \infty,\ i = 1, 2, \ldots m\}$, we say that f satisfies a Lipschitz condition on D in the variables $u_1, \ldots, u_m$, if there exists a constant $L > 0$ such that

$$|f(t, u_1, \ldots u_m) - f(t, z_1, \ldots z_m)| \le \sum_{i=1}^{m} |u_i - z_j|, \text{ for all } (t, u_1, \ldots, u_m) \text{ and } (t, z_1, \ldots, z_m) \text{ in } D. \tag{47}$$

By virtue of the mean value theorem it can be shown that if f and its first partial derivatives are continuous in D, and if

$$\left|\frac{\partial f(t, u_1, u_2, \ldots u_m)}{\partial(u_1, u_2, \ldots u_m)}\right| \le L, \text{ for } i = 1, 2, \ldots, m \text{ and all } (t, u_i) \in D, \tag{48}$$

then f is Lipschitz continuous on D.

Theorem:
Let $f_i(t, u_1, \ldots u_m),\ i = 1, \ldots m$ be defined on $D = \{(t, u_1, \ldots, u_m) \mid a \le t \le b,\ -\infty < u_i < \infty,\ i = 1, 2, \ldots m\}$. If all $f_i(t, u_1, \ldots u_m)$ are Lipschitz continuous on D, then the IVP of Eq. (46) has a unique solution $u_1(t), u_2(t), \ldots u_m(t)$ for $a \le t \le b$.

1. RK4 for systems of ODEs

Here we will simply generalize RK4 for systems of ODEs of the type of Eq. (46). Let $N > 0$, such that $h = (b-a)/N$ is the integration step size. As usually we will define the mesh points

$$t_n = a + n * h, \; n = 0, 1, 2, \dots N. \tag{49}$$

We will use the notation $u_{i,n}$ to designate the function i at time t_n, i.e., $u_{i,n} = u_i(t_n)$ We have the initial conditions

$$u_{1,0} = \alpha_1, u_{2,0} = \alpha_2, \dots, u_{m,0} = \alpha_m, \tag{50}$$

The RK4 algorithm then becomes as follows.

Algorithm for RK4:
Input: a, b, N, initial condition α, $f(t, y)$

- Step 1: Set $h = (b-a)/N$
  $t = a$
  $u_1 = \alpha_1, u_2 = \alpha_2, \dots, u_m = \alpha_m,$
  output(t,u_i)
- Step 2: For $n = 1, 2, \dots N-1$ do Steps 3-4
- Step 3:

$$\begin{aligned}
&\text{For } i = 1, 2, \dots, m\\
&\quad k_{1,i} = h f_i(t_n, u_1, \dots, u_m)\\
&\text{For } i = 1, 2, \dots, m\\
&\quad k_{2,i} = h f_i(t + \frac{h}{2}, u_1 + \frac{k_{1,1}}{2}, u_2 + \frac{k_{1,2}}{2}, \dots u_m + \frac{k_{1,m}}{2})\\
&\text{For } i = 1, 2, \dots, m\\
&\quad k_{3,i} = h f_i(t + \frac{h}{2}, u_1 + \frac{k_{2,1}}{2}, u_2 + \frac{k_{2,2}}{2}, \dots u_m + \frac{k_{2,m}}{2})\\
&\text{For } i = 1, 2, \dots, m\\
&\quad k_{4,i} = h f_i(t + h, u_1 + k_{3,1}, u_2 + k_{3,2}, \dots u_m + k_{3,m})\\
&\text{For } i = 1, 2, \dots, m\\
&\quad\; u_i = u_i + (k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i})/6
\end{aligned} \tag{51}$$

- Step 4:

$$t = a + n \cdot h \tag{52}$$

- Step 5:
  output(t,$\vec{u}$)

Exercise 3: Generalize the RK3 algorithm to a system of m first-order ODEs.

Exercise 4: Non-dimensionalize the ODE of the simple harmonic oscillator, and cast it to first-order form.

$$m\ddot{x} = -k(x - x_0). \tag{53}$$

To achieve a general solution (independent of m and k) we want to non-dimensionalize the equation first. To achieve this let us perform a dimensional analysis. We will denote the unit of time by $[T]$, and the unit of length $[L]$. Then above equation becomes

$$m\frac{[L]}{[T]^2} = -k[L] \Rightarrow m\frac{1}{[T]^2} = -k \Rightarrow \sqrt{\frac{m}{k}} = [T], \tag{54}$$

in other words, $\sqrt{\frac{m}{k}}$ defines a timescale in our problem. So, we will introduce a new dimensionless time $\hat{t} = t/\sqrt{\frac{m}{k}}$. Then, Eq. (53) becomes

$$\frac{d^2x}{d\hat{t}^2} = -x. \tag{55}$$

We can further introduce the length of the spring at the relaxed position x_0 as a lengtscale in the problem to non-dimensionalize x, i.e., $\hat{x} = x/x_0$. The fully non-dimensional ODE then becomes

$$\frac{d^2\hat{x}}{d\hat{t}^2} = -(\hat{x} - 1). \tag{56}$$

The benefit of this form is that all we need is to solve this problem once, and then by choosing x_0 and $\sqrt{\frac{m}{k}}$ we can rescale our time and length to solve the problem for any value of x_0 and $\sqrt{\frac{m}{k}}$ (for our choice of initial conditions). Since we can rename our variables, we can remove the hats to obtain the final non-dimensional form of the harmonic oscillator

$$\frac{d^2x}{dt^2} = -(x - 1). \tag{57}$$

E. Stability of numerical schemes for ODE IVP

We have so far seen that round-off error instabilities can arise that destroy the stability of a numerical integration scheme. Numerical schemes for ODEs can also be unstable, and they have to be analyzed for each ODE separately. Let us consider for example the simple IVP

$$\frac{du}{dt} = -\lambda u, \; u(0) = u_0, \; t > 0. \tag{58}$$

and let us consider the forward Euler method as the solution scheme, i.e.

$$u_{n+1} = u_n - \lambda u_n h = (1 - \lambda h)u_n \tag{59}$$

Let us now consider that we have an exact solution of the algebraic Eqs. (59), u_n, and that we perturbed it $u_n \to u_n + \delta u_n$ (essentially the perturbation is seeded by round off error), then the above equation becomes

$$\delta u_{n+1} = (1 - \lambda h)\delta u_n \tag{60}$$

which we can rewrite as

$$\delta u_{n+1} = (1 - \lambda h)^n \delta u_0 \tag{61}$$

from which we can write

$$\left| \frac{\delta u_{n+1}}{\delta u_0} \right| = |1 - \lambda h|^n \tag{62}$$

Thus, the term $|1 - \lambda h|$ is an amplification factor. If $|1 - \lambda h| > 1$, then the magnitude of the perturabation grows in time, if $|1 - \lambda h| \leq 1$, the perturbation does not grow. Thus, we have stability if

$$|1 - \lambda h| \leq \Rightarrow -1 \leq 1 - \lambda h \leq 1 \tag{63}$$

The right side of the inequality is always satisfied for $\lambda > 0$ (h is defined to be > 0), but the left side of the inequality is satisfied only if $h \leq 2/\lambda$! Thus, if the step size is greater than $2/\lambda$, the forward Euler method is numerically unstable!

Let us know consider the RK2 algorithm for the same ODE (58), for which we have $f(t, u) = -\lambda u$. The RK2 algorithm is

$$u_{n+1} = u_n + hf\left(t_n + \frac{h}{2}, u_n + \frac{h}{2} f(t_n, u_n)\right) \tag{64}$$

Using $f(t,u) = -\lambda u$, we can rewrite this as

$$u_{n+1} = u_n + hf\left(t_n + \frac{h}{2}, u_n - \lambda\frac{h}{2}u_n)\right) \tag{65}$$

and

$$u_{n+1} = u_n - \lambda h(u_n - \lambda\frac{h}{2}u_n) = u_n\left(1 - \lambda h(1 - \lambda\frac{h}{2})\right) \tag{66}$$

or

$$u_{n+1} = u_0\left(1 - \lambda h + \frac{(\lambda h)^2}{2}\right)^n \tag{67}$$

and again we obtain

$$\left|\frac{u_{n+1}}{u_0}\right| = \left|1 - \lambda h + \frac{(\lambda h)^2}{2}\right|^n \tag{68}$$

with the amplification factor this time being $\left|1 - \lambda h + \frac{(\lambda h)^2}{2}\right|$, which for stability must satisfy

$$\left|1 - \lambda h + \frac{(\lambda h)^2}{2}\right| \leq 1. \tag{69}$$

or

$$-1 \leq 1 - \lambda h + \frac{(\lambda h)^2}{2} \leq 1. \tag{70}$$

The right inequality becomes

$$\lambda h(\frac{\lambda h}{2} - 1) \leq 0 \Rightarrow \frac{\lambda h}{2} \leq 1 \Rightarrow h \leq \frac{2}{\lambda}, \tag{71}$$

which is the same restriction on the step size as in the Euler method. The left inequality in (70) requires

$$0 \leq 2 - \lambda h + \frac{(\lambda h)^2}{2} \tag{72}$$

If we set $x = \lambda h$, the right-hand-side in the last equation becomes $\frac{1}{2}x^2 - x + 2$, whose discriminant is $\Delta = 1 - 4\frac{1}{2}2 = -3 < 0$. Thus, $\frac{1}{2}x^2 - x + 2 > 0$. Hence, inequality (72) is always satisfied. Thus, RK2 has the same restriction on the timestep as the Euler method, i.e.,

$$h \leq \frac{2}{\lambda}. \tag{73}$$

Thus, we cannot choose an arbitrarily large h, because the numerical scheme will become unstable!

As a rule of thumb, when solving IVP usually there is an associated time (or length) scale with the problem, that we must resolve, i.e, the step must be smaller than this scale. The ODE (58), has as characteristic time scale $1/\lambda$. Thus, to be numerically stable (and to have an accurate solution), we must resolve this time scale, i.e.,

$$h < 1/\lambda, \tag{74}$$

which is a restriction very close to the one we obtained with a much more elaborate perturbation analysis. So, always try to think about the problem you are trying to solve. Determine the characteristic scales, and these will determine what step size you should use. Thinking, can save us a lot of time from trying to figure out why our codes may fail/crash.

Exercise 5: Let us now consider a system of ODEs

$$\begin{aligned} \frac{dx}{dt} &= -\frac{2}{3}(x+y) \\ \frac{dy}{dt} &= \frac{1}{3}(2x - 7y) \end{aligned} \tag{75}$$

What is the numerical stability condition when employing the forward Euler method?

Solution: The forward Euler method becomes

$$\begin{aligned} x_{n+1} =& x_n - \frac{2}{3}(x_n + y_n)h \\ y_{n+1} =& y_n + \frac{1}{3}(2x_n - 7y_n)h \end{aligned} \tag{76}$$

Let consider Harmonic perturbations to our variables, i.e.

$$x_n = x_0 \exp(inh),\ y_n = y_0 \exp(inh), \tag{77}$$

where i is the imaginary unit. If we plug Eq. (77) in Eq. (78) we obtain

$$\begin{aligned} x_0 \exp(ih) =& x_0 - \frac{2}{3}(x_0 + y_0)h \\ y_0 \exp(ih) =& y_0 + \frac{1}{3}(2x_0 - 7y_0)h \end{aligned} \tag{78}$$

This is the following eigenvalue problem

$$\begin{bmatrix} 1 - 2h/3 & -2h/3 \\ 2h/3 & 1 - 7h/3 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \lambda \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{79}$$

where $\lambda = \exp(ih)$ is the eigenvalue. For the system to be numerically stable the eigen values of the matrix

$$M = \begin{bmatrix} 1 - 2h/3 & -2h/3 \\ 2h/3 & 1 - 7h/3 \end{bmatrix} \tag{80}$$

must be smaller than unity. To find the eigenvalue of M we need to solve the characteristic polynomial

$$\det|M - \lambda I| = 0 \Rightarrow (1 - 2h/3 - \lambda)(1 - 7h/3 - \lambda) + 4h^2/9 = 0. \tag{81}$$

This equation has the following solutions $\lambda_1 = 1 - h$, $\lambda_2 = 1 - 2h$. For the numerical integration system to be stable the **absolute value** of both eigenvalues must be smaller than unity, i.e., we obtain the following stability requirements $|1 - h| < 1$, and $|1 - 2h| < 1$, which yield $-1 < 1 - h < 1$ and $-1 < 1 - 2h < 1$, and $h < 2$ and $h < 1/2$, respectively.

Exercise 6: Implement the RK4 algorithm for the following system of ODEs

$$\begin{aligned} \frac{dx}{dt} =& -\frac{2}{3}(x + y) \\ \frac{dy}{dt} =& \frac{1}{3}(2x - 7y) \end{aligned} \tag{82}$$

This system is exactly solvable. The help you compare your algorithm the analytic solution of the system is the following

$$\begin{aligned} x =& \frac{2}{3}e^{-t}(2x_0 - y_0) - \frac{1}{3}e^{-2t}(x_0 - 2y_0) \\ y =& \frac{1}{3}e^{-t}(2x_0 - y_0) - \frac{2}{3}e^{-2t}(x_0 - 2y_0) \end{aligned} \tag{83}$$

where $x_0 = x(0)$ and $y_0 = y(0)$ are the initial conditions.

IV. BOUNDARY VALUE PROBLEMS AND STIFF ODES

In many occasions in physics we need to specify not only an initial condition for the ODE, but also a "final" condition. These types of problems are called *boundary value problems*. A first-order ODE cannot have a boundary value problem because all first-order ODEs require is the value of the variable at one point, e.g., the value at one "boundary". Thus, second or higher order ODEs are the only ones that can have boundary value problems. For example, consider the electrostatic potential between two metal spheres of different radii. We can ask what is the potential between the two if the potential of the inner sphere is kept at potential V_{inner} and the outer one sphere is kept at potential 0. The potential between the sphere is governed by Laplace's equation, which in spherical symmetry reduces to

$$\frac{d^2V}{dr^2} + \frac{2}{r}\frac{dV}{dr} = 0,\ V(R_{\text{inner}} =) = V_{\text{inner}}, V(R_{\text{outer}}) = 0. \tag{84}$$

The reason why higher order systems can be cast as boundary value problems is that a higher order system requires more than one initial condition to determine the solution. In the example above, the second order equation requires that we know the value of V and the value of dV/dr at a point to solve it completely. Thus, it requires two conditions. Instead of providing the derivative, one can provide the value of the function at another point, making it a boundary value problem.

The above discussion should have already prepared the ground for one of the most common methods for solving boundary value problems – *the shooting method.* In the shooting method one casts the boundary value problem as an initial value problem where one would set $V(R_{\text{inner}} =) = V_{\text{inner}}$, and then set the derivative to some value solve the initial value problem, and if $V(R_{\text{outer}} = 0) \neq 0$, go back change the value of the derivative and iterate until $V(R_{\text{outer}} = 0) = 0$, to within a certain tolerance. This can be done for linear and non-linear equations, but we will not expand further on this topic here.

In addition to boundary value problems there exist *stiff* ordinary differential equations, which typically have part of their solution a transient solution (signal) changing on a rapid time scale, and a more "stready-state" part which changes on a slower time scale. These problems require special numerical schemes to be solved. We only mention stiff systems here, in case you ever run into a problem that has such behavior, and you need to resort to these special stiff solvers.