IMPRS GW Astronomy – Computational Physics 2025

Ordinary Differential Equations

Takami Kuroda

1 What are ODEs?

ODEs are equations as a function of $y(x)$ and its derivatives $y' \equiv dy/dx$, $y'' \equiv d^2y/dx^2$, $\cdots$, $y^{(n)'} \equiv d^ny/dx^n$ where $y(x)$ has only one argument x. ODEs are expressed in a general form as

$$g\left(x, y, y', y'', \cdots, y^{(n)'}\right) = 0 \tag{1}$$

1. **Example: Particle motion**

   The motion of a particle, whose mass is m, is expressed as

   $$F = ma(t), \tag{2}$$

   with F and $a(t)$ being the force acting on the particle and the acceleration of the particle at time t, respectively. Since the acceleration $a(t)$ is defined by the second time derivative of the position of the particle $x(t)$ as

   $$a(t) = \frac{d^2x(t)}{dt^2}, \tag{3}$$

   the ODE is expressed as

   $$g\left(t, x, \dot{x}, \ddot{x}\right) = F - m\ddot{x} = 0. \tag{4}$$

   Eq. (4) can be solved for given initial values at time $t = t_0$: $x(t_0) = x_0$ and $dx(t_0)/dt \equiv v(t_0) = v_0$. The solution of this ODE is

   $$F = ma(t) \Longleftrightarrow x(t) = x_0 + v_0 t + \frac{F}{2m}t^2. \tag{5}$$

1.1 Equivalence to first order differential equations

We next show that one ODE can be equivalent to first order simultaneous differential equations, with which we can solve the original ODE more easily. Let us first rewrite Eq. (1) as

$$y^{(n)'} = f\left(x, y, y', y'' \cdots, y^{(n-1)'}\right). \tag{6}$$

Then such n-th order differential equation can be reduced to

$$\begin{aligned}
\frac{dy}{dx} &= y' \\
\frac{dy'}{dx} &= y'' \\
&\vdots \\
\frac{dy^{(n-2)'}}{dx} &= y^{(n-1)'} \\
\frac{dy^{(n-1)'}}{dx} &= y^{(n)'} \left(= f\left(x, y, y', y'' \cdots, y^{(n-1)'}\right)\right). \qquad (7)
\end{aligned}$$

Therefore it indicates that a single n-th order differential equation is equivalent to the first order differential equations with n variables. Such first order simultaneous differential equations can be solved for given initial values:

$$y(x_0) = y_0, \quad y^{(n)'}(x_0) = a_n, \qquad (8)$$

making them to be termed as initial value problem (IVP).

2 Numerical methods for ODEs

How can we numerically solve a set of first order simultaneous equations (7)? In practice, we numerically solve such systems by means of numerical differentiation. Namely, for given right hand side (RHS) values $(y', y'', \ldots, y^{(n)'})$ in Eq. (7) at position $x = x_0$, which actually correspond to the slope (i.e. LHS), we estimate the next values $(y, y', \ldots, y^{(n-1)'})$ at $x = x_0 + h$. Here h denotes the step size. Afterward, we continue this procedure till reaching a desired final point $x = x_{\text{fin}}$. Therefore we should focus on solving each line of Eq. (7), taking a form of

$$y' = f(x, y). \qquad (9)$$

Here we assume for the simplicity that the function f has only two variables (x, y). However, the extension from two to more variables such as to $(x, y, y', y'', \cdots)$ is straightforward and we do not explicitly consider the existence of derivative terms in the following discussion. We also note that y as well as its derivatives y', y'', $\cdots$, $y^{(n)'}$ have only one argument x.

2.1 Numerical differentiation: Forward/backward/central finite differences

We begin with explaining the basic of *numerical differentiation*: the slope of function y. The slope of y at position x (i.e. $y' = dy/dx$) can be written as

$$\frac{dy(x)}{dx} = \lim_{h \to 0} \frac{y(x+h) - y(x)}{h}. \qquad (10)$$

Using the Taylor series expansion, $y(x+h)$ is expressed as

$$y(x+h) = y(x) + hy'(x) + \frac{1}{2}h^2y''(x) + \frac{1}{6}h^3y'''(x) + \cdots \,. \tag{11}$$

Plugging Eq. (11) into Eq. (10) yields the forward difference for a finite step size h

$$\frac{dy}{dx} \approx \frac{y(x+h)-y(x)}{h} = y'(x) + \frac{1}{2}hy''(x) + \frac{1}{6}h^2y'''(x) + \cdots \,. \tag{12}$$

From this we can estimate the error value as

$$err \approx \frac{y(x+h)-y(x)}{h} - y'(x) = \frac{1}{2}hy''(x) + \frac{1}{6}h^2y'''(x) + \cdots = \mathcal{O}(h), \tag{13}$$

indicating that this finite difference method is first-order accuracy. Analogously, the backward difference results in the same order of accuracy with the forward one as

$$err \approx \frac{y(x)-y(x-h)}{h} - y'(x) = -\frac{1}{2}hy''(x) + \frac{1}{6}h^2y'''(x) - \cdots = \mathcal{O}(h). \tag{14}$$

Finally if we take the central difference as follows

$$\frac{dy}{dx} \approx \frac{y(x+h)-y(x-h)}{2dh} = y'(x) + \frac{1}{6}h^2y'''(x) + \frac{1}{120}h^4y^{(5)}(x)\cdots \,, \tag{15}$$

the estimated error becomes

$$err \approx \frac{y(x+h)-y(x-h)}{2h} - y'(x) = \frac{1}{6}h^2y'''(x) + \cdots = \mathcal{O}(h^2). \tag{16}$$

It implies that we can get one order of magnitude smaller error, i.e., the second-order accuracy.
Like these, each method as well as other various methods that will be mentioned later give different numerical accuracy and one should carefully choose which is the most suitable one for one's purpose. We also have to pay attention for the numerical cost: Generally the higher the numerical accuracy is, the slower the computational speed is.

2.2 Stiffness and stability

Before going to the introduction of several major numerical methods, we shortly touch the *stiffness* of the equation(s) that we are going to solve and the *stability* of numerical methods. Although there are no concrete definition for the stiffness of the system, we can intuitively understand that the system (or equation(s)) is stiff, if the source term (RHS of Eq. (7)) changes quite *rapidly* during an integration path considered or there are significantly large differences between the source terms of each equation (7). From the mathematical point of view, the latter condition can be understood as follows. If the system is non-linear, then we can take a local linear approximation. Anyhow, let us consider a generalized linear system of Eq. (7) ($\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$). The equation is then expressed as

$$\mathbf{y}' = \mathbf{A}x + \mathbf{B}\mathbf{y}, \tag{17}$$

where $\mathbf{A}$ and $\mathbf{B}$ are vector of size n and $n \times n$ matrix, respectively. A general solution to this equation is

$$\mathbf{y} = \int dx \mathbf{A} x + \sum_{i=1}^{n} b_i e^{\lambda_i x}. \tag{18}$$

From this, λ_i can be read as eigen values. The stiffness is roughly defined by a ratio of the fastest and slowest propagation mode of the system

$$s = \frac{\max |\Re \lambda_i|}{\min |\Re \lambda_i|}, \tag{19}$$

and if s is large, typically $s \gtrsim 10^{4-5}$, then the system can be said as stiff. In the stiff system, if we do not appropriately choose the step size, the deviation of numerically estimated value from the true solution tends to become large. Moreover, the deviation sometimes diverges, i.e. numerical instability, meaning that $|y| \to \infty$, and we cannot continue the simulation.

The stability of numerical methods are often discussed by applying a test function

$$y' = \lambda y, \tag{20}$$

where $\lambda \in \mathbb{C}$. Eq. (20) can be analytically solved and one gets the solution

$$y(x) = e^{\lambda x}. \tag{21}$$

At the same time in the finite difference expression this solution can be alternatively expressed as

$$y_{i+1} = e^{\lambda(x_i + h)} = e^{\lambda h} e^{\lambda x_i} \equiv R(\lambda h) y_i = (R(\lambda h))^{i+1} y_0, \tag{22}$$

here y_i is the value at i-th point x_i and the finite difference h is defined by $h = x_{i+1} - x_i$. In the equation, $R(z)$ with $z \in \mathbb{C}$ is the so-called stability function. The stability of a method is then discussed when $y_i \to 0$ is achieved for $i \to \infty$. This means a step size h, which satisfies $R(\lambda h) < 1$, is considered to be a safe step size without entering an instability regime. On the complex plane, the region $R(\lambda h) < 1$ is termed as the region of absolute stability. The actual form of $R(z)$ differs from method to method, and those having a larger stable region are more stable methods.

Furthermore as a useful index of strong stability, there is a terminology "*A-stable*". The numerical integration scheme is called A-stable when its stable function $R(z)$ covers the whole complex region with $\Re z < 0$. Recall that $z = \lambda h$, this means that for any step size the A-stable method allows us to eventually reach a *non-divergent* (usually 0) asymptotic value for the result ($y = e^{\lambda x}$) of the test function $y' = \lambda y$. Therefore, the A-stable methods are generally considered to be the ideal method for solving quite stiff equation.

2.3 Euler method

The Euler method is the simplest one-step method to solve IVP $y' = f(x, y)$ and to obtain a series of y_i at each discretized position x_i for given initial values y_0 and $f(x_0, y_0)$ at x_0. There are two types of the Euler method: explicit and implicit one. Below we discuss their basic concept especially focusing on their stability.

2.3.1 Explicit method

Based on the forward finite difference form (Eq. (11)) and neglecting higher order terms than h^2, we can express the approximate value at the next discretization point x_{i+1} as

$$y(x_{i+1}) \approx y(x_i) + hf(x_i, y_i), \tag{23}$$

where $h = x_{i+1} - x_i$ is the step size. As can be seen, the explicit Euler method evaluates the value at $(i+1)$-th step simply applying the current slope $f(x_i, y_i)$. This makes the scheme simple and quite easy to implement.

It is also important to keep in mind the local truncation error, which measures the deviation of numerical result from the exact solution. As for the Euler method, the next step value is obtained via Eq. (23). Meanwhile the exact solution can be read as

$$y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \mathcal{O}(h^3). \tag{24}$$

Therefore the expected deviation is given by

$$y(x_i + h) - y(x_{i+1}) = \frac{h^2}{2}y''(x_i) + \mathcal{O}(h^3). \tag{25}$$

This indicates that the Euler method is expected to produce an error of the order of $\mathcal{O}(h^2)$ at every time step.

As a drawback of its simple expression, the Euler method is known to often suffer from numerically instability. It sometimes returns quantitatively different values $y(x+h)$ and consequently resulting in an inaccurate sequence of solutions $y(x+2h)$, $y(x+3h)$, $\cdots$ (i.e. the overall behavior of $y(x)$ becomes qualitatively incorrect).

If we apply the test function ($y' = \lambda y$) to the explicit Euler method, one obtains

$$y_{i+1} = y_i + hy'(x_i) = y_i + \lambda h y_i = (1 + \lambda h)y_i. \tag{26}$$

Therefore the stability function becomes $R(z) = 1 + z$ and $|1 + z| < 1$ is the region of stability for the Euler method. Such a narrow stability region enforces us to employ a sufficiently small time step, leading generally to *time consuming* numerical simulation.

2.3.2 Implicit method

To overcome the quite narrow stability region of the explicit Euler method, one can alternatively solve the IVP *implicitly*. This is equivalent to rewrite Eq. (23) as

$$y(x_{i+1}) \approx y(x_i) + hf(x_{i+1}, y_{i+1}), \tag{27}$$

thus evaluating the slope $y'(= f)$ at $(i+1)$-th position instead of at i. The more stable nature of this method in comparison to the explicit method can be understood as follows. Again applying the former test equation $y' = \lambda y$, this implicit Euler method yields

$$y_{i+1} = y_i + hy'(x_{i+1}) = y_i + \lambda h y_{i+1}. \tag{28}$$

It can be rewritten as

$$y_{i+1} = \frac{1}{1 - \lambda h} y_i. \tag{29}$$

Therefore, the stability function becomes

$$R(z) = \frac{1}{1-z}. \tag{30}$$

As a consequence, the stability region ($|R(z)| < 1$) appears for almost all complex numbers z (except $|1-z| < 1$), which is quite a larger domain compared to the one with the previous explicit Euler method.
Note that ensuring the numerical stability is not equivalent to achieving sufficient accuracy in numerical results. For instance, the explicit methods sometimes give more accurate results such as for the shock wave propagation. As already mentioned, however, the explicit methods should employ significantly shorter (time-)step size, meaning that the total simulation time to reach the desired final simulation time can sometimes be an order of magnitude or even more longer than the implicit models. Otherwise, the numerical instability appears in the explicit scheme and would eventually crush the simulation.

2.4 Runge-Kutta method

However, their accuracy (local truncation error) is of the order of $\mathcal{O}(h^2)$, which is usually not so high. To achieve a higher order accuracy, there are also multi-step methods. The Runge-Kutta methods are one of them and are iterative methods to solve IVP $y' = f(x, y)$. In these methods, the value at next step y_{i+1} is obtained after s-stages by

$$y_{i+1} = y_i + h \sum_{n=1}^{s} b_n k_n, \tag{31}$$

where

$$k_n = f(x_i + c_n h, y_i + h \sum_{l=1}^{s} a_{nl} k_l), \quad n = 1, \ldots, s. \tag{32}$$

From a condition of the Taylor expansion

$$h \sum_{n=1}^{s} b_n = h, \tag{33}$$

$\sum_{n=1}^{s} b_n$ must be unity. Furthermore, the following condition is often used to determine the coefficients.

$$\sum_{l=1}^{s} a_{nl} = c_n, \quad n = 1, \ldots, s, \tag{34}$$

with $c_1 = 0$.

To more easily understand the coefficients appearing in the Runge-Kutta method, the following *Butcher tableau* is commonly used.
Note that for the explicit Runge-Kutta methods, the upper triangle (i.e. a_{ij} with $j \geq i$) becomes always 0. This means for calculating k_n, we need the values only for k_m with $m < n$. Therefore we can evaluate k_1, k_2, $\cdots$, k_s in a sequential manner. Meanwhile

$$\begin{array}{c|ccccc}
c_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_s
\end{array}$$

for the implicit method, a_{ij} with $j \geq i$ are not always 0. Therefore when calculating for instance k_1, we need the values of k_2, k_3, $\cdots$, which makes the implementation of such implicit scheme more complicated. In the following, we focus only on the explicit Runge-Kutta methods.

2.4.1 Stability function of Runge-Kutta methods

Applying again the test function $y' = \lambda y$ to Runge-Kutta methods, one immediately obtains an alternative expression to Eq. (32) as follows

$$k_n = \lambda \left(y_i + h \sum_{l=1}^{n-1} a_{nl} k_l \right) , \tag{35}$$

where we use a condition $a_{nl} = 0$ for $l \geq n$ of the explicit Runge-Kutta methods. Plugging this equation into Eq. (31) yields

$$y_{i+1} = R(\lambda h) y_i \tag{36}$$

with

$$R(z) = 1 + z \sum_i b_i + z^2 \sum_{i,j} b_i a_{ij} + z^3 \sum_{i,j,k} b_i a_{ij} a_{jk} + \cdots . \tag{37}$$

Recalling that a_{ij} is $s \times s$ lower triangular matrix, $a^s = 0$. Therefore, the stability function $R(z)$ is a polynomial of degree $\leq s$. If we discuss whether the Runge-Kutta methods are A-stable or not, since

$$|R(z)| = \infty, \quad \text{for } z \to -\infty, \tag{38}$$

they are not A-stable. indicating we have to choose some appropriate step size.

2.4.2 Accuracy of Runge-Kutta methods

For smaller numbers of stages ($s \leq 4$), it is known that the order of numerical accuracy p is the same with the number of stages, i.e., $p = s$. This means, if one needs a fourth-order accuracy code, Runge-Kutta with at least four stages is required. However, to achieve a higher order accuracy beyond 5th-order, $s \geq p + 1$ is known to be necessary, though the actual minimum number of stages is not well understood.

2.4.3 Second-order methods with two stages ($s = 2$)

One example of Butcher tableau for Runge-Kutta methods with two stages ($s = 2$) is
Here α is a parameter and the method is called “midpoint method” for $\alpha = 1/2$ and

$$
\begin{array}{c|cc}
0 & & \\
\alpha & \alpha & \\
\hline
 & 1-\frac{1}{2\alpha} & \frac{1}{2\alpha}
\end{array}
$$

“Heun’s method” for $\alpha = 1$. The next step value can be explicitly written as

$$
y_{i+1} = y_i + h\left[\left(1 - \frac{1}{2\alpha}\right) f(x_i, y_i) + \frac{1}{2\alpha} f(x_i + \alpha h, y_i + \alpha h f(x_i, y_i))\right]. \tag{39}
$$

2.4.4 RK4: Runge-Kutta with four stages $(s = 4)$

Generally the Runge-Kutta method refers to this four-stage method, which is fourth-order accuracy. The Butcher tableau of RK4 is denoted as

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Therefore the value at the next step y_{i+1} is explicitly given by

$$
y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{40}
$$

where

$$
\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f(x_i + h/2, y_i + hk_1/2) \\
k_3 &= f(x_i + h/2, y_i + hk_2/2) \\
k_4 &= f(x_i + h, y_i + hk_3).
\end{aligned} \tag{41}
$$