Computational Astrophysics

E. Larrañaga

Observatorio Astronómico Nacional
Universidad Nacional de Colombia

May 25, 2019

Outline

Ordinary Differential Equations

A system of first-order ordinary differential equations (ODEs) is a relationship between an unknown (vectorial) function $y(x)$ and its derivative $y'(x)$. The general system of first-order ODEs has the form

$$y'(x) = f(x, y(x)) \,. \tag{1}$$

A solution to the differential equation is, obviously, any function $y(x)$ that satisfies it.

Ordinary Differential Equations

There are two general classes of first-order ODE problems:

1. Initial value problems: $y(x_i)$ is given at some starting point x_i.
2. Two-point boundary value problems: y is known at two ends ("boundaries") of the domain and these "boundary conditions" must be satisfied simultaneously.

Reduction to First-Order ODE

Any ODE can be reduced to first-order form by introducing additional variables.

Example

$$y''(x) + q(x)y'(x) = r(x) \; . \tag{2}$$

Introducing a new function $z(x)$ this can be written as

$$\begin{aligned} &(1) \quad y'(x) = z(x)\,, \\ &(2) \quad z'(x) = r(x) - q(x)z(x) \; . \end{aligned}$$

Errors and ODEs

All procedures to solve numerically an ODE consist of transforming a continuous differential equation into a discrete iteration procedure that starts from the initial conditions and returns the values of the dependent variable $y(x)$ at points $x_m = x_0 + m * h$, where h is the discretization step size assumed to be constant here).

Errors and ODEs

Two kinds of errors can arise in this procedure:

1. **Round-off error**. Due to limited float point accuracy. The global round-off is the sum of the local float point errors.
2. **Truncation error**.
   Local: The error made in one step when we replace a continuous process (e.g. a derivative) with a discrete one (e.g., a forward difference).
   Global: If the local truncation error is $\mathcal{O}(h^{n+1})$, then the global truncation error must be $\mathcal{O}(h^n)$, since the number of steps used in evaluating the derivatives to reach an arbitrary point x_f, having started at x_0, is $\frac{x_f - x_0}{h}$.

Euler's Method

We want to solve $y' = f(x, y)$ with $y(x_0) = y_0$. We introduce a fixed stepsize h and we first obtain an estimate of $y(x)$ at $x_1 = x_0 + h$ using Taylor's theorem:

$$\begin{aligned} y(x_1) = y(x_0 + h) &= y(x_0) + y'(x_0)h + \mathcal{O}(h^2)\,, \\ &= y(x_0) + hf(x_0, y(x_0)) + \mathcal{O}(h^2)\,. \end{aligned} \tag{3}$$

By analogy, we obtain that the value y_{n+1} of the function at the point $x_{n+1} = x_0 + (n+1)h$ is given by

$$y_{n+1} = y(x_{n+1}) = y_n + hf(x_n, y(x_n)) + \mathcal{O}(h^2)\,. \tag{4}$$

This is called the *forward Euler Method*.

Euler's Method

Euler's method is extremely simple, but rather inaccurate and potentially unstable.
The error scales $\propto h^2$ locally. However, if L is the length of the domain, then $h = L/N$, where N is the number of points used to cover it. Since we are taking N integration steps, the global error is $\propto Nh^2 = NL^2/N^2 = LL/N \propto h$.

Hence, forward Euler is a first-order accurate method.

Stability of Euler's Method

Forward Euler is an *explicit* method. This means that y_{n+1} is given explicitly in terms of known quantities y_n and $f(x_n, y_n)$.

Explicit methods are simple and efficient, but the drawback is that the step size must be small for stability.

Stability of Euler's Method

Example

$$y' = -ay \;, \qquad \text{with} \quad y(0) = 1\,,\; a > 0\,,\; y' = \frac{dy}{dt}\;. \tag{5}$$

The exact solution to this problem is $y = e^{-at}$, which is stable and smooth with $y(0) = 1$ and $y(\infty) = 0$.
Applying forward Euler,

$$y_{n+1} = y_n - a\,h\,y_n = (1 - ah)y_n \tag{6}$$

$$y_{n+1} = (1 - ah)^2 y_{n-1} = \cdots = (1 - ah)^{n+1} y_0\;. \tag{7}$$

Stability of Euler's Method

This implies that in order to prevent any potential amplification of errors, we must require that $|1 - ah| < 1$.

In fact, there are 3 cases,

(i)	$0 < 1 - ah < 1$	:	$(1 - ah)^{n+1}$ decays (good!).
(ii)	$-1 < 1 - ah < 0$	:	$(1 - ah)^{n+1}$ oscillates (not so good!).
(iii)	$1 - ah < -1$	:	$(1 - ah)^{n+1}$ oscillates and diverges (bad!).

This gives the stability criterion of $0 < h < \frac{2}{a}$.

Predictor-Corrector Method

Consider the modification

$$y_{n+1} = y_n + h\,\frac{f(x_n, y_n) + f(x_{n+1}, y_{n+1})}{2}\ . \tag{8}$$

This may be a better estimate as it is using the "average slope" of y. However, we don't know y_{n+1} yet.

Predictor-Corrector Method

We can get around this problem by using forward Euler to estimate y_{n+1} and then use Eq. (8) for a better estimate:

$$\begin{aligned} y_{n+1}^{(\mathrm{P})} &= y_n + hf(x_n, y_n)\ , && \text{(predictor)} \\ y_{n+1} &= y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(\mathrm{P})})\right]\ . && \text{(corrector)} \end{aligned} \tag{9}$$

One can show that the error of the predictor-corrector method decreases locally with h^3, but globally with h^2. One says it is *second-order accurate* as opposed to the Euler method, which is first-order accurate.

Runge-Kutta Methods

The idea behind Runge-Kutta (RK) methods is to match the Taylor expansion of $y(x)$ at $x = x_n$ up to the highest possible order.

Second Order RK Method

For

$$\frac{dy}{dx} = f(x, y) \ , \tag{10}$$

we have

$$y_{n+1} = y_n + ak_1 + bk_2 \ , \tag{11}$$

with

$$\begin{aligned} k_1 &= h\,f(x_n, y_n) \ , \\ k_2 &= h\,f(x_n + \alpha h, y_n + \beta k_1) \ . \end{aligned} \tag{12}$$

Second Order RK Method

The four parameters a, b, α, β will be fixed so that Eq. (11) agrees as well as possible with the Taylor series expansion of $y' = f(x, y)$:

$$\begin{aligned} y_{n+1} &= y_n + hy'_n + \frac{h^2}{2} y''_n + \mathcal{O}(h^3) \ , \\ &= y_n + hf(x_n, y_n) + \frac{h^2}{2} \frac{d}{dx} f(x_n, y_n) + \mathcal{O}(h^3) \ , \\ &= y_n + hf_n + h^2 \frac{1}{2} \left(\frac{\partial f_n}{\partial x} + \frac{\partial f_n}{\partial y} f_n \right) + \mathcal{O}(h^3) \ , \end{aligned} \tag{13}$$

where $f_n = f(x_n, y_n)$.

Second Order RK Method

Using Eq. (11),

$$y_{n+1} = y_n + ahf_n + bhf(x_n + \alpha h, y_n + \beta hf_n) \ . \tag{14}$$

Now we expand the last term of Eq. (14) in a Taylor series to first order in terms of (x_n, y_n),

$$y_{n+1} = y_n + ahf_n + bh\left[f_n + \frac{\partial f}{\partial x}(x_n, y_n)\alpha h + \frac{\partial f}{\partial y}(x_n, y_n)\beta hf_n\right] \ , \tag{15}$$

and can now compare this with Eq. (11) to read off:

$$a + b = 1 \ , \qquad \alpha b = \frac{1}{2} \qquad \beta b = \frac{1}{2} \ . \tag{16}$$

Second Order RK Method

So there are only 3 equations for 4 unknowns and we can assign an arbitrary value to one of the unknowns. Typical choices are:

$$\alpha = \beta = \frac{1}{2}\ , \qquad a = 0\ , \qquad b = 1\ . \tag{17}$$

With this, we have for RK2:

$$k_1 = hf(x_n, y_n)\ , \tag{18}$$

$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)\ , \tag{19}$$

$$y_{n+1} = y_n + k_2 + \mathcal{O}(h^3)\ . \tag{20}$$

This method is locally $\mathcal{O}(h^3)$, but globally only $\mathcal{O}(h^2)$.

Note that using $a = b = 1/2$ and $\alpha = \beta = 1$ we recover the predictor-corrector method!

RK3

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1)\ , \\
k_3 &= hf(x_n + h, y_n - k_1 + 2k_2)\ , \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 4k_2 + k_3) + \mathcal{O}(h^4)\ . \qquad (21)
\end{aligned}
$$

RK4

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \ , && (22) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \ , \\
k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \ , \\
k_4 &= hf(x_n + h, y_n + k_3) \ , \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5) \ . && (23)
\end{aligned}
$$

RK Methods with Adaptive Step Size

The RK methods above require choosing a fixed step size h but, how should one choose this parameter?
It would be better to choose an *error tolerance* and let h be chosen automatically to satisfy this error tolerance.
This implies that we need

1. A method for estimating the error.
2. A way to adjust the stepsize h, if the error is too large (or too small).

Embedded Runge-Kutta Formulae

Embedded RK formulae provide an error estimator. Now we will present the scheme for 3rd/4th order embedded RK (Bogaki and Shampine)

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \ , \\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \ , \\
k_3 &= hf(x_n + \frac{3}{4}h, y_n + \frac{3}{4}k_2) \ , \\
y_{n+1} &= y_n + \frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3 + \mathcal{O}(h^4) \\
k_4 &= hf(x_n + h, y_{n+1}) \\
y^*_{n+1} &= y_n + \frac{7}{24}k_1 + \frac{1}{4}k_2 + \frac{1}{3}k_3 + \frac{1}{8}k_4 + \mathcal{O}(h^3) \ .
\end{aligned}
\tag{24}
$$

The error is

$$
\delta y_{n+1} = y_{n+1} - y^*_{n+1} \ . \tag{25}
$$

Embedded Runge-Kutta Formulae

In this scheme, k_4 of step n is the same as k_1 of step $n+1$.
Therefore, k_1 does not need to be recomputed on step $n+1$; simply save k_4 and re-use it on the next step.
This trick is called FSAL (First Same As Last).

Adjusting the Step Size h

Given the error estimate $\delta y_{n+1} = y_{n+1} - y^*_{n+1}$ we want that it says smaller than some tolerance, $|\delta y_{n+1}| \leq \epsilon$ by adjusting h.
Usually, one sets

$$\epsilon = \underbrace{\epsilon_a}_{\text{absolute error tolerance}} + |y_{n+1}| \underbrace{\epsilon_r}_{\text{relative error tolerance}} . \tag{26}$$

Adjusting the Step Size h

We define

$$\Delta = \frac{|\delta y_{n+1}|}{\epsilon} \ , \tag{27}$$

and we want $\Delta \approx 1$.

Note that for a p-th-order formula, $\Delta \sim \mathcal{O}(h^p)$. So if you took a step h and got a value Δ, then you change the step to h_{desired},

$$h_{\text{desired}} = h \left| \frac{\Delta_{\text{desired}}}{\Delta} \right|^{\frac{1}{p}} \ , \tag{28}$$

to get the new $\Delta_{\text{desired}} = 1$.

Adjusting the Step Size h

The algorithm to adjust h can be written as follows:

1. Take step h, measure Δ.
2. If $\Delta > 1$ (error too large), then
   - set $h_{\text{new}} = h \left| \frac{1}{\Delta} \right|^{\frac{1}{p}} S$, where S is a fudge factor (~ 0.9 or so).
   - *reject* the old step, redo with h_{new}.
3. If $\Delta < 1$ (error too small), then
   - set $h_{\text{new}} = h \left| \frac{1}{\Delta} \right|^{\frac{1}{p}} S$.
   - accept old step, take next step with h_{new}.

Next Class

Ordinary Differential Equations. Boundary Value Problems