Numerical Solutions to ODEs and Runge-Kutta

Finite Difference Methods for ODE & PDE

June 2022

1 Numerical differentiation on data:

Again, recall finite difference approximations to $f'(x)$:

- **Forward Differences**:

$$\frac{df}{dx} \simeq \frac{f(x+\Delta x) - f(x)}{\Delta x}. \tag{1}$$

  Error: $O(\Delta x)$

- **Backward Differences**:

$$\frac{df}{dx} \simeq \frac{f(x) - f(x+\Delta x)}{\Delta x}. \tag{2}$$

  Error: $O(\Delta x)$

- **Central Differences**:

$$\frac{df}{dx} \simeq \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}. \tag{3}$$

  Error: $O(\Delta x^2)$

- **Second Derivative**:

$$f''(x) = \frac{f(x+\Delta x) + f(x-\Delta x) - 2f(x)}{\Delta x^2}. \tag{4}$$

  Error: $O(\Delta x^2)$

Higher accuracy schemes can be obtained by using more points.

Assume that we have two vectors of data, $\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$ and $\begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ f_n \end{bmatrix}$, how can we compute the derivative df/dx based on these two vectors?

We can use CD since it is the most accurate approximation. However, there are some cases where it cannot bes used:
1. In real-time.
2. At boundaries; as the first and last points do not have any points before or after, recpectively. So we use FD for the first, left most, point, and BD for the last, right most, point.

What if the x's were not evenly spaced?
The CD will look like,

$$f'_k = \frac{f_{k+1} - f_{k-1}}{x_{k+1} - x_{k-1}} \tag{5}$$

Coding

All the previously discussed was implemented here.

The preceding was merely the foundation for numerically integrating ODEs, which is the main point of this entire discussion.

For instance, let us consider this ODE: dx/dt = f, in a computer we can approximate this by FD as,

$$\frac{x_{k+1} - x_k}{\Delta t} = f(x_k) \implies x_{k+1} = x_k + \Delta t f(x_k), \tag{6}$$

where x_{k+1} is the future state, and x_k is the current state. So this is basically an integrator; we give it x_0 and it gives us x_1, now we have x_1, so it gives us x_2, and so on.

2 Numerical integration and numerical solutions to ODEs:

All of the following is very similar to numerical differentiation, so only a brief discussion will be provided.

Consider, for instance, the function f(x) that defines a particular curve. It is possible to approximate the area underneath this curve by dividing it into finite rectangles. The sum of the areas of all rectangles approximates the desired area under consideration.

$$\begin{aligned} \int_a^b f(x)dx &= \lim_{N\to\infty} \sum_{k=1}^{N} (f(a + \frac{b-a}{N}k)) \\ &= \lim_{\Delta x\to 0} \sum_{k=1}^{N} f(a + k\Delta x)\Delta x. \end{aligned} \tag{7}$$

This is the Riemann integral. However, it turns out that an integral called "Lebesque" integral can be used in cases where the Riemann Integral does not work. It is an integration with horizontal bars, instead of the vertical ones, that takes the limit of Δx goes to zero.

Just like what we did in FD, we are not going to take the limit. So, by following the same approach we will have,

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N}(f(a+\frac{b-a}{N}k)), \tag{8}$$

where N is the number of rectangles, and $\frac{b-a}{N}$ is the width (Δx).

We can use either the right-sided rectangle formula, or the left-sided rectangle formula. The former is,

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=1}^{N} f(x_k)\Delta x, \tag{9}$$

where the height of the rectangle is defined by the right point of its interval, and the later is,

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=0}^{N-1} f(x_k)\Delta x, \tag{10}$$

where the height of the rectangle is defined by the left point of its interval.

If Δx's were variable, then we write $x_k - x_k - 1$ instead of Δx.

2.1 Trapezoid Rule

Nevertheless, how accurate is the Riemann approach? In fact, it is possible for it to overshoot or undershoot. The undershooting happens everytime the function has a positive slope, while the overshooting happens everytime the function has a negative slope. To avoid these undershootings and overshootings and accordingly have better error properties, we could use the so-called "Trapezoid Rule", which states

$$\frac{\Delta x}{2}[f(x) + f(x+\Delta x)]. \tag{11}$$

Local Error: is the accumulated errors for every litthe rectangle.

Global Error: the total error that we get from adding up all of the local errors.

- **Right & Left Rectangles**
  - *Local error:* O(Δx^2)
  - *Global error:* O(Δx)
  - *How to approximate the error:*

$$\begin{aligned}\int_{x_o}^{x_o+\Delta x} f(x)dx &\approx \int_{x_o}^{x_o+\Delta x} [f(x) + (x-x_o)\frac{df}{dx}(x_o) + \frac{(x-x_o)^2}{2}\frac{d^2f}{dx^2}(x_o) + ...]dx \\ &\approx \left[f(x_o)\Delta x + \frac{(x-x_o)^2}{2}\frac{df}{dx}(x_o) + \frac{(x-x_o)^3}{3}\frac{d^3f}{dx^3}(x_o) + ...\right]_{x_o}^{x+\Delta x} \\ &\approx f(x_o)\Delta x + \frac{\Delta x^2}{2}\frac{df}{dx}(x_o) + ...\end{aligned} \tag{12}$$

- **Trapezoid Rule**
  - *Local error:* $\mathrm{O}(\Delta x^3)$
  - *Global error:* $\mathrm{O}(\Delta x^2)$

Coding

An implementation of this can be found here.

2.2 Simpson's Rule

Compared to the previous rules, this one is the most accurate. The formula of this rule is,

$$\int_{x_o}^{x_2} f(x)dx = \frac{\Delta x}{3}[f_o + 4f_1 + f_2] + \frac{\Delta x^5}{90}f^4(c). \tag{13}$$

3 Numerical solutions to ODEs (Forward and Backward Euler):

What is a vector field? Consider this ODE $\dot{x} = f(x)$, with the initial condition $x(0) = x_o$. f is a vector field, it gives a vector $\dot{x}$ for every x you evaluate.

- **Forward (explicit) Euler:**

  f may be nonlinear function, but oftenly we are going to have $\dot{x} = Ax \implies e^A t x_o$. Our interest here is to "numerically" solve the ODE by starting with x_o, and iterating to get $x_o \to x_1 \to x_2 \to ... \to x_N$. "Trajectory". This is all based on finite difference derivatives. To approximate $\dot{x}$, we can use FD:

$$\dot{x_k} \approx \frac{x_{k+1} - x_k}{\Delta t} = f(x_k) \implies x_{k+1} = x_k + \Delta t f(x_k) \tag{14}$$

  This is called Forward Euler. However, it is unstable. Stability means: does this trajectory do anything like what the actual solution does? and does it more close as we make Δt smaller? In the case of $\dot{x} = Ax$,

$$\begin{aligned} x_{k+1} &= x_k + \Delta t A x_k \\ &[I + \Delta t A]x_k, \end{aligned} \tag{15}$$

  where I is the identity matrix.

- **Backward (implicit) Euler:**
  We could use BD instead of FD, and get more accurate implicit approximation,

$$\dot{x}_{k+1} = \frac{x_{k+1} - x_k}{\Delta t} = f(x_{k+1}) \implies x_{k+1} = x_k + \Delta t f(x_{k+1}) \tag{16}$$

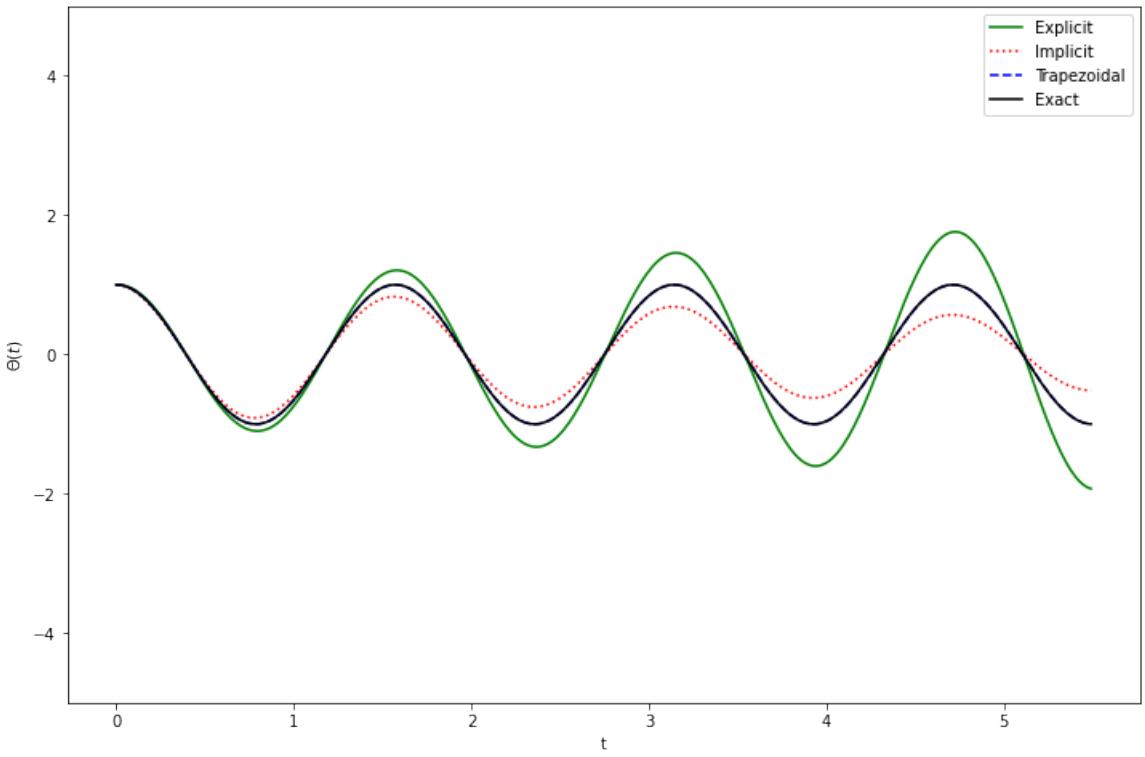
  Even though this method is sort of slow, it has better stability. For a linear system, $\dot{x} = Ax$,

$$\begin{aligned} x_{k+1} &= x_k + \Delta t A x_{k+1} \\ &= [I - \Delta t A]^{-1} x_k \end{aligned} \tag{17}$$

4 Accuracy and Stability:

- **Accuracy:**
  The capacity of a scheme to approach the exact solution, which is typically unknown, as a function of the step size h.
- **Stability:**
  A scheme's stability is measured by how well it can prevent the error from escalating as time is integrated forward. The scheme is stable if the error stays the same; if not, it is unstable. Some integration techniques are regarded to as unstable because they are stable for some values of h but unstable for others.

Here is an illustration of stability, a solution of pendulum equation.



Coding

The codes of the example can be found here.

5 Runge-Kutta integration of ODEs and the Lorenz equation: