ODE: Advanced Methods

May 6, 2020

RUNGE-KUTTA METHODS

Second Order Runge-Kutta

Our general ODE takes the form,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}(t), t) \tag{1}$$

where the vector $\mathbf{x}$(t) is the desired solution.

Consider the kepler problem of an orbit in the x-y plane.We have,

$$\mathbf{x}(t) = [r_x(t), r_y(t), v_x(t), v_y(t)]$$

and

$$\begin{aligned}\mathbf{f}(\mathbf{x}(t), t) &= \left[\frac{dr_x}{dt}, \frac{dr_y}{dt}, \frac{dv_x}{dt}, \frac{dv_y}{dt}\right] \\ &= [v_x(t), v_y(t), F_x(t)/m, F_y(t)/m]\end{aligned}$$

The first Runge-Kutta formula is,

$$x(t+\tau) = x(t) + \tau f\left(x^*\left(t+\frac{\tau}{2}\right), t+\frac{\tau}{2}\right) \tag{2}$$

where

$$x^*(t+\tau/2) = x(t) + \frac{\tau}{2} f(x(t), t)$$

To explain why we are doing this, we need to step back and look at the taylor expansion in one-variable case.

$$\begin{aligned}x(t+\tau) &= x(t) + \tau\frac{dx(\zeta)}{dt} \\ &= x(t) + \tau f(x(\zeta), \zeta)\end{aligned}$$

This expansion is exact for some value ζ between t and $t+\tau$.

1. The Euler formula takes $\zeta = t$.
2. Euler-Cormer uses $\zeta = t$ in the velocity and$\zeta = t+\tau$ in the position euation.

3. RungeKutta tries to use $\zeta = t + \tau/2$, since that is probably a better guess.However since $x(t+\frac{\tau}{2})$ is not known, we use an Euler step to compute $x * (t + \tau/2)$

Fourth-Order Runge-Kutta

In practice, the most commonly used method is the following fourth-order Runge-Kuta fomula:

$$x(t+\tau) = x(t) + \frac{1}{6}\tau[F_1 + 2F_2 + 2F_3 + F_4] \tag{3}$$

where

$$\begin{aligned} F_1 &= f(x,t) \\ F_2 &= f\left(x + \frac{1}{2}\tau F_1, t + \frac{1}{2}\tau\right) \\ F_3 &= \left(x + \frac{1}{2}\tau F_2, t + \frac{1}{2}\tau\right) \\ F_4 &= f(x + \tau F_3, t + \tau) \end{aligned}$$

Adaptive Runge-Kutta Method

In certain problems, we have to vary τ. We have a rough idea of what τ should be; now we have to selecta τ_{min} and τ_{max} and find a way to switch betwen them. If we do this manually, it would be worse than just doing the brute-force calculation with a small time step.

Adaptive programs monitor the time step such ythat user given accuracy is always mainatained. An example of such a methos is as follows. Suppose, you are going from the state $x(t)$ to $x(t+\tau)$, we can do it in two ways.

- Take a big step from $x(t)$ to $x(t+\tau)$. Call this $x_b(t+\tau)$.
- Now, repeat the same calculation but with a time step of $\tau/2$. Call this value you got from two jumps of $\tau/2$, $x_s(t+\tau)$

Now, find the difference between these two calculation. that will estimate the local truncation error. If the error is tolerable, we ggo with the timestep τ. otherwiise, we choose a lower time step and repeat the clauclation till the optimal time step is found. The estimated local truncation error can guife us selectung a new time step for the next iteration.

Call Δ the local truncation errror. we know that for RK4 method, $\Delta \propto \tau^5$. Suppose the current time step τ_{old} gave an error of $\Delta_c = |x_b - x_s|$. Given that we want the error to be less than or equal to the user-specified ideal error, Δ_i, then, the estimate for the new time step is:

$$\tau_{est} = \tau \| \frac{\Delta_i}{Delta_c} |^{1/5} \tag{4}$$

Since this is only an estimate, the new time step is $\tau_{new} = S_1 \tau_{est}$, where $S_1 < 1$. We also need a safety factor $S_2 > 1$ just to be sure that the program won't astly raise or lower the time step

CHAOS in Lorenz Model

(*still in development*)

$$\frac{dx}{dt} = \sigma(y - x) \tag{5}$$

$$\frac{dy}{dt} = rx - y - xz \tag{6}$$

$$\frac{dz}{dt} = xy - bz \tag{7}$$

T