1 Numerical Integration Methods

1.1 Leapfrog

The Leapfrog method is a numerical method commonly used to solve ordinary differential equations (ODEs) that involve second-order time derivatives. Such an ODE is shown below:

$$\ddot{x} = \frac{d^2x}{dt^2} = f(x) \tag{1}$$

The Leapfrog method is a variant of the finite difference method, and it approximates the solution of an ODE by discretizing both time and space. The method gets its name from the way it calculates the values of the solution at each time step, which resembles a "leapfrogging" motion. It is a simple and efficient algorithm that is often used in simulations of physical systems, such as celestial mechanics or molecular dynamics. The idea is straight forward; in the time interval Δt,

$$\begin{aligned} a(t_0) &= f(x_0) \\ x(t_0+\Delta t) &= x(t_0) + v(t_0)\Delta t + a(t_0)\frac{\Delta t^2}{2} \qquad (2) \\ v(t_0+\Delta t) &= v(t_0) + \{a(t_0) + a(t_0+\Delta t)\}\frac{\Delta t}{2} \qquad (3) \end{aligned}$$

For more stability, this version can be rearranged to what is called 'kick-drift-kick' form,

$$\begin{aligned} v(t_0+\Delta t/2) &= v(t_0) + a(t_0)\frac{\Delta t}{2} \\ x(t_0+\Delta t) &= x(t_0) + v(t_0+\Delta t/2)\Delta t \qquad (4) \\ v(t_0+\Delta t) &= v(t_0+\Delta t/2) + a(t_0+\Delta t)\frac{\Delta t}{2} \end{aligned}$$

This version provides more time resolution to our calculation; however, it increases the number of calculations needed by about 50%.

1.2 Runge Kutta

The Runge-Kutta methods, named after the German mathematicians Carl Runge and Martin Wilhelm Kutta, family of numerical methods used to solve ordinary differential equations (ODEs) that are in the following form

$$\frac{\partial y}{\partial x} = f(x, y) \tag{5}$$

The basic idea behind the Runge-Kutta method is to approximate the solution of an ODE by taking small steps and using a weighted average of function evaluations at different points within each step.

$$y_{n+1} = y_n + \delta x \sum_{i=1}^{m} b_i k_i \qquad x_{n+1} = x_n + \delta x \tag{6}$$

where

$$k_i = f(x_n + c_i \delta x, y_n + \delta x \sum_{j=1}^{i-1} a_{ij} k_j) \tag{7}$$

These *equations 6 and 7* define the family of methods. To specify a perticular method, order m, coefficients a_{ij}, b_i and c_i should be provided. The coefficients of any Runge-Kutta method can be visualized by a tableau called *Butcher Tableau*.

$$\begin{array}{c|cccc} 0 & & & & \\ c_1 & a_{21} & & & \\ c_2 & a_{31} & a_{32} & & \\ \vdots & & \cdots & & \\ c_m & a_{m1} & a_{m2} & \cdots & a_{m,m-1} \\ \hline & b_1 & b_2 & \cdots & b_m \end{array}$$

Figure 1: Butcher Tableau

The simplest Runge-Kutta method is the *Euler's method*. Its *Butcher tableau* is as follows.

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

The most commonly used version of the Runge-Kutta method is the fourth-order Runge-Kutta method, also known as RK4. The RK4 method involves four function evaluations per step and has an error term that is proportional to the step size raised to the fifth power. Two of the most used *Butcher tableaus* for *RK4* given below.

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \qquad \begin{array}{c|cccc} 0 & & & & \\ 1/3 & 1/3 & & & \\ 2/3 & -1/3 & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array}$$

Figure 2: Butcher Tableau for RK4

The second tableau in *figure 2* is called the *"3/8 rule"*. Its main advantage is that its error coefficients are smaller than the other. But it costs more floating point operations per step. Resulting in slower calculations.

References

[1] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulations*, McGraw-Hill Book Company, 1985, p. 56.