Overview of Runge-Kutta Methods

Runge-Kutta Methods

Runge-Kutta (RK) methods are a family of numerical techniques for solving ordinary differential equations (ODEs) of the form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{1}$$

where $\mathbf{y}$ is a vector (for systems of ODEs) or scalar (for a single ODE), $\mathbf{f}(t, \mathbf{y})$ defines the dynamics, and $\mathbf{y}_0$ is the initial condition. These methods approximate the solution at discrete times $t_{n+1} = t_n + h$, where h is the step size, by combining multiple slope evaluations.

General Framework

An s-stage Runge-Kutta method approximates $\mathbf{y}_{n+1}$ from $\mathbf{y}_n$ as:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i \tag{2}$$

where the stage slopes are:

$$\mathbf{k}_i = \mathbf{f}\left(t_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j\right) \tag{3}$$

The coefficients a_{ij}, b_i, and c_i are defined in a Butcher tableau:

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

Explicit methods have $a_{ij} = 0$ for $j \geq i$, while implicit methods require iterative solutions.

Common Runge-Kutta Methods

First-Order (Euler's Method)

Single stage ($s = 1$):

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n) \tag{4}$$

Error per step: $O(h^2)$, global error: $O(h)$. Simple but less accurate.

Second-Order RK (RK2)

Example (midpoint method):

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n) \tag{5}$$

$$\mathbf{k}_2 = \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_1) \tag{6}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2) \tag{7}$$

Error per step: $O(h^3)$, global error: $O(h^2)$.

Fourth-Order RK (RK4)

Four stages:

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n) \tag{8}$$

$$\mathbf{k}_2 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right) \tag{9}$$

$$\mathbf{k}_3 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right) \tag{10}$$

$$\mathbf{k}_4 = \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_3) \tag{11}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{12}$$

Error per step: $O(h^5)$, global error: $O(h^4)$. Widely used for its accuracy and simplicity.

Higher-Order and Adaptive Methods

Methods like RK5 or Runge-Kutta-Fehlberg (RKF45) use more stages or adaptive step sizes. Implicit methods (e.g., Gauss-Legendre RK) handle stiff systems.

Application to the Differential Equation

For the equation $mg - \beta\dot{x}^2 - k(x - x_0) = m\ddot{x}$, we use the substitution $y = x - x_0$:

$$m\ddot{y} + \beta\dot{y}^2 + ky = mg \tag{13}$$

Define $y_1 = y$, $y_2 = \dot{y}$:

$$\frac{dy_1}{dt} = y_2 \tag{14}$$

$$\frac{dy_2}{dt} = \frac{mg - \beta y_2^2 - ky_1}{m} \tag{15}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} y_2 \\ \frac{mg - \beta y_2^2 - ky_1}{m} \end{bmatrix}$$

Initial conditions: $y_1(0) = 0$, $y_2(0) = v_0$. RK4 iteratively applies the four-stage update to compute $\mathbf{y}(t)$, yielding $x(t) = y_1(t) + x_0$, $\dot{x}(t) = y_2(t)$.

Advantages and Limitations

- **Advantages**:
  - High accuracy (e.g., RK4 has $O(h^4)$ global error).
  - Easy to implement for non-stiff systems.
  - Flexible for systems of ODEs.
- **Limitations**:
  - Fixed-step methods are inefficient for stiff systems.
  - Implicit methods are stable but computationally intensive.
  - Higher-order methods increase computational cost.

Practical Considerations

- **Step Size (h)**: Smaller h improves accuracy but increases computation. Adaptive methods adjust h dynamically.
- **Stability**: RK4 is stable for small h; stiff systems (e.g., large k/m) may require implicit or adaptive methods.
- **Implementation**: RK4 is straightforward to code using libraries like NumPy.

Summary

Runge-Kutta methods solve ODEs $\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y})$ by combining multiple slope evaluations. The RK4 method, with four stages, offers $O(h^4)$ global error and is applied to the system:

$$\dot{y}_1 = y_2 \tag{16}$$

$$\dot{y}_2 = \frac{mg - \beta y_2^2 - ky_1}{m} \tag{17}$$

to numerically solve the given equation, providing accurate position and velocity solutions.