Runge-Kutta 4th Order Method (RK4)

Created by Haroon Sakhi

Introduction

The Runge-Kutta 4th Order Method (RK4) is a widely used numerical method for solving differential equations due to its accuracy and efficiency. This notebook provides:

- A concise mathematical formulation of RK4.
- Step-by-step implementation.
- A Mathematica code to compute RK4.
- A comparison with Euler's method.

```
"💡 Key Insight: RK4 is more accurate than
  Euler's method because it computes four intermediate slopes."
```

Derivation

Consider the **first-order differential equation:**

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

with an initial condition:

$$y(x_0) = y_0.$$

The goal is to approximate $y(x_0 + h)$ using a **fourth-order method.**

- #### Taylor Series Expansion

The exact value of y at $x_0 + h$ can be written as a **Taylor series:**

$$y(x_0 + h) = y_0 + h\, y'(x_0) + \frac{h^2}{2} y''(x_0) + \frac{h^3}{6} y'''(x_0) + \frac{h^4}{24} y^{(4)}(x_0) + O(h^5).$$

Since the equation is given as $y' = f(x, y)$, the next step is to compute higher derivatives of y to get a more accurate approximation.

Computing Higher Derivatives

Expanding each term:

$$y' = f(x, y),$$
$$y'' = \frac{d}{dx} f(x, y) = f_x + f_y \cdot f,$$
$$y''' = f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_y f_x + f_y^2 f,$$
$$y^{(4)} = f_{xxx} + 3 f_{xxy} f + 3 f_{xyy} f^2 + f_{yyy} f^3 + 3 f_{xy} f_x + 3 f_{xy} f_y f + 3 f_{yy} f f_x + 3 f_{yy} f_y f^2 + f_y f_{xx} + 3 f_y f_{xy} f + 3 f_y^2 f_x + 3 f_y^2 f_y f + f_y^3 f.$$

Computing these derivatives explicitly for every problem is impractical, so instead, the goal is to approximate the solution by using function evaluations at carefully chosen points.

Constructing the RK4 Approximation

Instead of relying on a single function evaluation (as in Euler's method), the RK4 method uses four function evaluations at different points. The general update formula is:

$$y_{n+1} = y_n + h(a_1 k_1 + a_2 k_2 + a_3 k_3 + a_4 k_4).$$

where k_1, k_2, k_3, k_4 are function evaluations at different points.

Defining the $k - Values$

The function evaluations are defined as follows:

$$k_1 = f(x_n, y_n),$$
$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right),$$
$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right),$$
$$k_4 = f(x_n + h, y_n + h k_3).$$

Finding the Coefficients a_1, a_2, a_3, a_4

Expanding these terms in a **Taylor series** and matching coefficients with the exact series gives:

$$a_1 = \frac{1}{6}, \quad a_2 = \frac{1}{3}, \quad a_3 = \frac{1}{3}, \quad a_4 = \frac{1}{6}.$$

Thus, the final **RK4 update formula** is:

$$\boxed{y_{n+1} = y_n + h\left(\frac{1}{6} k_1 + \frac{1}{3} k_2 + \frac{1}{3} k_3 + \frac{1}{6} k_4\right).}$$

"💡 Key Insight: This derivation provides a structured approach to the RK4 method, ensuring accuracy by incorporating multiple function evaluations."

Why RK4 is a Better Choice

"Feature"	"Euler Method"	"RK4 Method"
"Accuracy"	"Low"	"High"
"Computational Cost"	"Low"	"Moderate"
"Error per Step"	"O(h^2)"	"O(h^5)"

Extra Notes

✔ RK4 reduces truncation error significantly.
✔ It requires more computations but offers better stability.
✔ Used in physics simulations, engineering, and control systems.

Implementing RK4 Method

Euler's method follows these key steps:

◈ **Breaking Down RK4Step:** This function performs a single step of the **RK4 method** to update the solution.

◈ **Inputs:** It takes the function f (representing the ODE), the current values of x and y and the step size h .

◈ **Calculating Slopes:** The method computes four intermediate slopes (k_1, k_2, k_3, k_4)at different points within the interval.

◈ **Final Update:** It combines these slopes using a weighted average to determine the next value y_{n+1}

Example: Applying RK4 Method

Defining the Differential Equation

In[∘]:=
```
dydx[x_, y_] := -2 x y
Solution[x_] := 4 Exp[-x^2]
```

☑ The exact solution is $y = 4\, e^{-x^2}$, which we compare against RK4 approximation.

Writing the RK4 Method

In[∘]:=

```
RK4[start_, end_, stepSize_, initialConditions_] := Module[{x, y, X, Y, K1, K2, K3, K4},
  {x, y} = initialConditions;
  X = {x}; Y = {y};
  While[x < end,
    K1 = stepSize*dydx[x, y];
    K2 = stepSize*dydx[x + stepSize/2, y + K1/2];
    K3 = stepSize*dydx[x + stepSize/2, y + K2/2];
    K4 = stepSize*dydx[x + stepSize, y + K3];

    y = y + (K1 + 2 K2 + 2 K3 + K4)/6;
    x = x + stepSize;
    AppendTo[X, x];
    AppendTo[Y, y];
    ];
  Transpose[{X, Y}]]
```

Key Features :

- Starts with initial values .
- Iterates using RK4 formula .
- Stores computed values for visualization .

Running the Method & Comparing with the Exact Solution

In[∘]:=

```
xi = -2;
xf = 2;
dx = 0.01;
IC = {-2, 0.07326};

(* Compute numerical solution *)
RK4Results = RK4[xi, xf, dx, IC];
xSol = Subdivide[xi, xf, 10000];
ySol = Solution /@ xSol;
```

Key Insights :

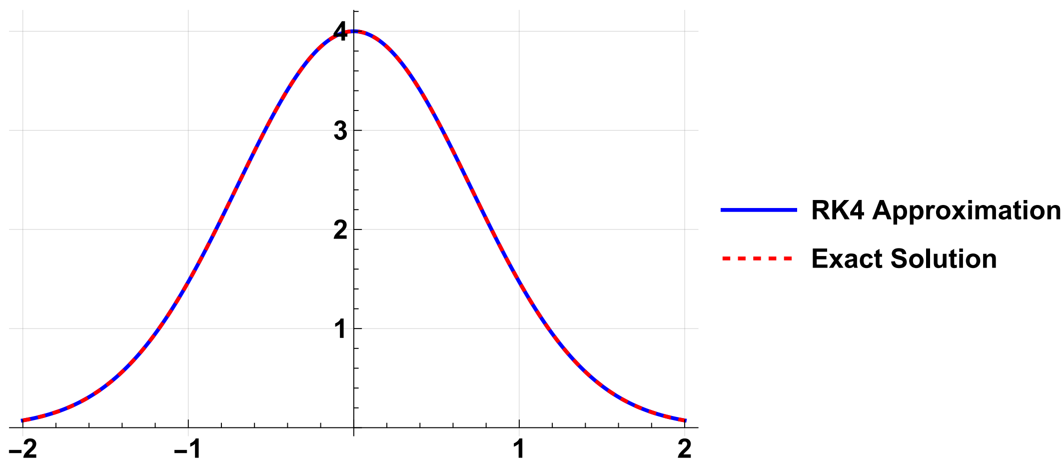
- **Step size:** 0.01
- **Comparison: RK4** vs.exact solution

Visualizing Results

RK4 Approximation vs. Exact Solution

In[ ]:=
```
ListLinePlot[{RK4Results, Transpose[{xSol, ySol}]},
  PlotStyle → {Blue, {Dashed, Red}},
  GridLines → Automatic,
  PlotLegends → {"RK4 Approximation", "Exact Solution"},
  FrameLabel → {"x", "y"}, LabelStyle → {Bold, 14}
]
```

Out[ ]=



- **Blue Line :** RK4 approximation
- **Dashed Line :** Exact solution
- **Goal :** Assess accuracy visually

Conclusion

☑ RK4 provides significantly higher accuracy compared to Euler’s method with a manageable computational cost.

By incorporating four slope evaluations per step, RK4 minimizes truncation errors.

It achieves better stability and precision, making it ideal for solving ODEs efficiently.

While slightly more complex, RK4 balances accuracy and efficiency, making it a preferred choice for many applications.