Runge-Kutta Method for Solving Ordinary Differential Equations

Author: John M. Cimbala, Penn State University
Latest revision: 26 September 2016

Consider a first-order ordinary differential equation (ODE) for y as a function of t,

$$\frac{dy}{dt} = B - Ay \tag{1}$$

Assume that the starting or initial condition $y(t_{start})$ at some time $t = t_{start}$ is known (t_{start} is often but not necessarily zero). If coefficients A and B are constants, Eq. 1 can be solved analytically. However, if A and/or B are functions of t and/or y, an analytical solution may be difficult, if not impossible to find. In such cases, the ***Runge-Kutta marching technique*** is useful for obtaining an *approximate* numerical solution of Eq. 1. Subroutines to perform Runge-Kutta marching are built into modern mathematical programs such as Matlab; nevertheless, readers should be familiar with how the method works. While leaving out much of the details, this learning module provides enough information about the algorithm so that readers can write a computer program to perform Runge-Kutta marching. Readers are encouraged to learn more about this technique by studying Press et al. (1986) or other books on numerical methods.

The most simple-minded algorithm for solving Eq. 1 numerically is the explicit Euler method, where one marches explicitly in time steps of duration Δt, incrementing y based on its slope with respect to time. Letting subscript n denote the time step at hand, one can march to the next time step, n+1, as follows:

$$y_{n+1} = y_n + \Delta t \left(\frac{dy}{dt} \right)_n \tag{2}$$

where the slope in Eq. 2 is evaluated from the right hand side of Eq. 1 at time step n,

$$\left(\frac{dy}{dt} \right)_n = B_n - A_n y_n \tag{3}$$

Once the value of y_{n+1} is known, Eq. 2 is re-solved at the new time step; this process is continued from the starting time, t_{start}, till some ending time, t_{end}. While this technique is simple to program on a computer, it is inherently unstable, only first-order accurate, and requires very small Δt in order to achieve reasonable results. The main problem here is that the slope is evaluated only at time step n, and is assumed to be constant throughout the time interval Δt, as illustrated (and exaggerated) in Figure 1.

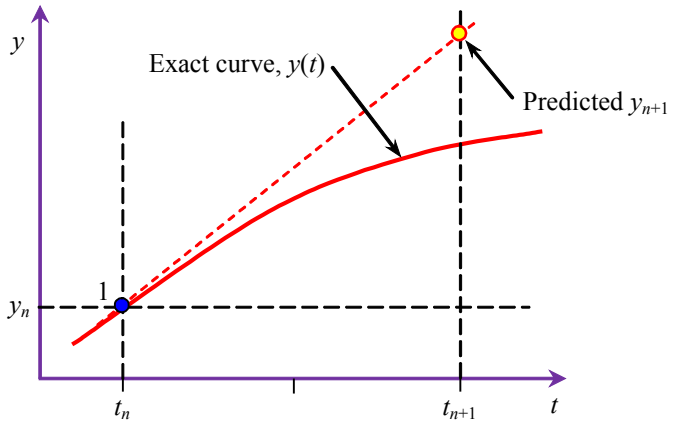

Figure 1 Illustration of the explicit Euler method in which the predicted value of y_{n+1} at time t_{n+1} is extrapolated from the slope (derivative) of y at time $t = t_n$..

In reality, as one marches towards time step n+1, the slope does *not* remain constant. Mathematicians and engineers have developed clever algorithms to modify the slope such that information is used from one or more values of t between t_n and t_{n+1}. These algorithms include the implicit Euler method and various kinds of predictor-corrector techniques, which can be formulated to first-, second-, or higher-order accuracy.

The **Runge-Kutta technique** is **fourth-order accurate**, and can be thought of as a kind of predictor-corrector technique in that the final value of y_{n+1} at $t = t_{n+1}$ is calculated as

$$y_{n+1} = y_n + \Delta y_{final} \tag{4}$$

where increment Δy_{final} is a ***weighted average*** of four "***trial increments***," namely Δy_1, Δy_2, Δy_3, and Δy_4, evaluated from slopes calculated at $t = t_n$, $t_{n+1/2}$, $t_{n+1/2}$, and t_{n+1}, respectively, as indicated in Figure 2.

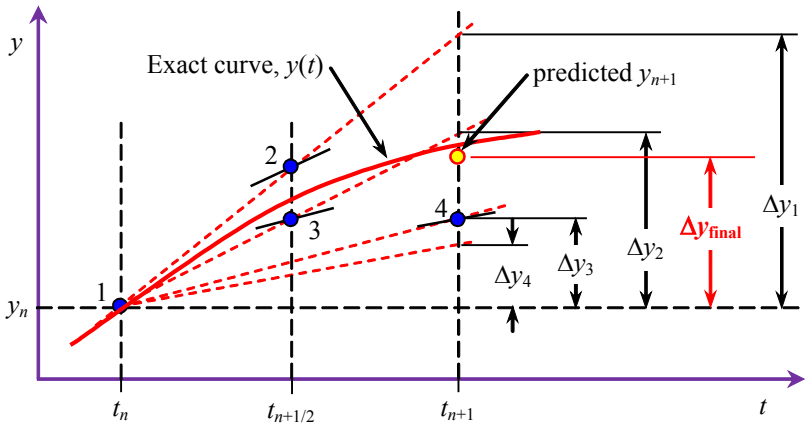

Figure 2 Illustration of the four points where trial increments Δy_1, Δy_2, Δy_3, and Δy_4 are calculated for the fourth-order Runge-Kutta marching technique; point 1 is the original point at $t = t_n$, and points 2, 3, and 4 are trial points at $t = t_{n+1/2}$, $t = t_{n+12}$, and $t = t_{n+1}$, respectively.

The final predicted value of y_{n+1} at $t = t_{n+1}$ is calculated as

$$\Delta y_{\text{final}} = \frac{\Delta y_1}{6} + \frac{\Delta y_2}{3} + \frac{\Delta y_3}{3} + \frac{\Delta y_4}{6} \tag{5}$$

In Eq. 5 the first trial increment, Δy_1, is evaluated as in the explicit Euler method, using the slope evaluated at point 1, at $t = t_n$,

$$\Delta y_1 = \Delta t \left(\frac{dy}{dt} \right)_{t=t_n,\, y=y_n} \tag{6}$$

This slope is used to predict a trial midpoint (point 2) at $t = t_{n+1/2} = t_n + \Delta t/2$ (halfway to the next time step),

$$y_2 = y_n + \frac{\Delta y_1}{2}$$

The second slope is calculated at point 2, which is used to predict the second trial increment (Δy_3) and the second trial midpoint (point 3),

$$\Delta y_2 = \Delta t \left(\frac{dy}{dt} \right)_{t=t_n+\frac{\Delta t}{2},\, y=y_2} \tag{7}$$

$$y_3 = y_n + \frac{\Delta y_2}{2}$$

The third slope is calculated at point 3, which is used to predict the third trial increment (Δy_4) and the third trial point (point 4), this time at $t = t_{n+1}$:

$$\Delta y_3 = \Delta t \left(\frac{dy}{dt} \right)_{t=t_n+\frac{\Delta t}{2},\, y=y_3} \tag{8}$$

$$y_4 = y_n + \Delta y_3$$

The fourth slope is calculated at trial point 4, as predicted by the third slope.

$$\Delta y_4 = \Delta t \left(\frac{dy}{dt} \right)_{t=t_n+\Delta t,\, y=y_4} \tag{9}$$

Finally, a weighted average of all four slopes is used to predict y_{n+1}, using Eqs. 4 and 5. As seen in Eq. 5, increments Δy_2 and Δy_3 have twice the importance (weight) as the other two. Another way to say this is that the two inner slopes are counted as twice as important as the two outer slopes.

The technique can be applied to more than one differential equation simultaneously. In addition, an n^{th}-order ODE can be solved by the Runge-Kutta method by splitting it into n first-order ODEs.