Ordinary Differential Equations

Runge-Kutta Methods

Sachin Shanbhag

Department of Scientific Computing
Florida State University,
Tallahassee, FL 32306.

References

- S. Chapra and R. Canale, Numerical Methods for Engineers
- Carnahan and Wilkes, Applied Numerical Methods, University of Michigan, Class Notes, 1996.
- C. Moler, Numerical Computing with Matlab, available at `http://www.mathworks.com/moler/chapters.html`
- Pal, Numerical Analysis for Scientists and Engineers, 2007.
- M. Heath, Scientific Computing: An Introductory Survey
- wikipedia.org

Contents

Runge Kutta Methods

- First developed by German physicist and mathematician Carl Runge in 1895
- RK methods approach accuracy of higher order Taylor methods without calculating higher derivatives
- Generalized form:

$$\boxed{y_n = y_{n-1} + h\phi(t_{n-1}, y_{n-1}, h)} \tag{1}$$

where $\phi(t_{n-1}, y_{n-1}, h)$ is called the increment function which represents a judicious choice of "slope"

Runge Kutta Methods

The increment function is a weighted sum of "slopes" k measured over the interval $[t_{n-1}, t_n]$:

$$\phi = a_1 k_1 + a_2 k_2 + ... + a_s k_s \tag{2}$$

where the a's are constants and s is the number of stages.

Roughly speaking, s determines the order of the RK method

For the ODE $y' = f(t, y)$, the ks are obtained by evaluating $f(t, y)$ at several points.

$$k_i = f(t_{n-1} + \Delta t_i, y_{n-1} + \Delta y_i) \tag{3}$$

Generalized RK Methods

$$y_n = y_{n-1} + h\phi(t_{n-1}, y_{n-1}, h)$$
$$\phi = a_1 k_1 + a_2 k_2 + ... + a_s k_s$$

The k's are given by,

$$\begin{aligned} k_1 &= f(t_{n-1}, y_{n-1}) && (4) \\ k_2 &= f(t_{n-1} + p_2 h, y_{n-1} + q_{21} k_1 h) && (5) \\ k_3 &= f(t_{n-1} + p_3 h, y_{n-1} + q_{31} k_1 h + q_{32} k_2 h) && (6) \\ &\vdots \\ k_s &= f\left(t_{n-1} + p_s h, y_{n-1} + h \sum_{i=1}^{s-1} q_{s,i} k_i\right) && (7) \end{aligned}$$

The as ps and qs are constants which define a particular method from the family of s-stage RK methods.

Deriving RK methods: Strategy

So far we have forced RK methods to have a certain form through eqns 1, 2, and the definitions of the ks.

The overall strategy for an $s-$stage method is to:

(i) write down a Taylor series expansion around t_{n-1} as before

(ii) linearize the slopes k_i, and stick them into eqn. 1

(iii) compare the powers of step-size h to determine the constants of the method.

Thus, we end up doing a lot of up-front work to derive a particular method.

But once a method is derived it can be used on any $f(t, y)$ unlike the old Taylor series method.

RK methods of order 1

- Consider $m = 1$, then

$$\phi = a_1 k_1 = a_1 f(t_{n-1}, y_{n-1})$$

- Thus,

$$y_n = y_{n-1} + h a_1 f(t_{n-1}, y_{n-1})$$

- By comparing with a Taylor series expansion,

$$y_n = y_{n-1} + h f(t_{n-1}, y_{n-1}) + \frac{h^2}{2} y''(\xi)$$

  we find $a_1 = 1$.

- Thus, RK method of order 1 is the same as Euler's method

RK methods of order 2

Here we have,

$$y_n = y_{n-1} + h(a_1 k_1 + a_2 k_2), \tag{8}$$

with

$$\begin{aligned} k_1 &= f(t_{n-1}, y_{n-1}) \quad &(9) \\ k_2 &= f(t_{n-1} + p_2 h, y_{n-1} + q_{21} k_1 h) \quad &(10) \end{aligned}$$

Recall the Taylor series expansion:

$$y_n = y_{n-1} + hf(t_{n-1}, y_{n-1}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right) + \mathcal{O}(h^3), \tag{11}$$

We are going to linearize k_2 (10) and stick it in (8). Comparison of (8) and (11) will help us find the constants of the method.

RK methods of order 2

A function of two variables may be linearized by truncating the Taylor series expansion:

$$g(x+\Delta x, y+\Delta y) = g(x,y) + \Delta x \frac{\partial g}{\partial x} + \Delta y \frac{\partial g}{\partial y} + \ldots$$

Thus, k_2 may be linearized as:

$$\begin{aligned} f(t_{n-1}+p_2 h, y_{n-1}+q_{21}k_1 h) = f(t_{n-1}, y_{n-1}) + p_2 h \frac{\partial f}{\partial t} \\ + q_{21}k_1 h \frac{\partial f}{\partial y} + \mathcal{O}(h^2) \end{aligned} \qquad (12)$$

Sticking k_1 and the linearized k_2 (eqn. (12)) into equation (8)

RK methods of order 2

$$y_n = y_{n-1} + a_1 h f(t_{n-1}, y_{n-1}) + a_2 h f(t_{n-1}, y_{n-1}) + a_2 p_2 h^2 \frac{\partial f}{\partial t}$$
$$+ a_2 q_{21} h^2 f(t_{n-1}, y_{n-1}) \frac{\partial f}{\partial y} + \mathcal{O}(h^3)$$

This can be simplified by collecting powers of h as:

$$y_n = y_{n-1} + h f(t_{n-1}, y_{n-1}) (a_1 + a_2)$$
$$+ h^2 \left(a_2 p_2 \frac{\partial f}{\partial t} + a_2 q_{21} f(t_{n-1}, y_{n-1}) \frac{\partial f}{\partial y} \right) + \mathcal{O}(h^3) \quad (13)$$

Compare it with the Taylor series expansion

$$y_n = y_{n-1} + h f(t_{n-1}, y_{n-1}) + \frac{h^2}{2} \left(\frac{\partial f}{\partial t} + f \frac{\partial f}{\partial y} \right) + \mathcal{O}(h^3), \quad (11)$$

RK methods of order 2

- A term by term comparison yields

$$a_1 + a_2 = 1$$
$$a_2 p_2 = \frac{1}{2}$$
$$a_2 q_{21} = \frac{1}{2}$$

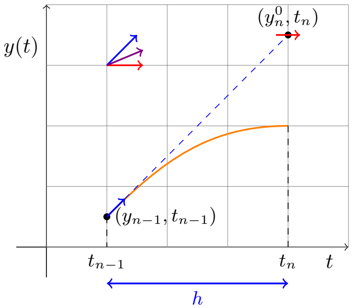
- We have 3 equations and 4 unknowns. There is a family (with infinite members) of RK order 2 methods.
- For a given choice of a particular variable, say a_2, the others can be uniquely determined. In fact, we have already met some of these methods before.

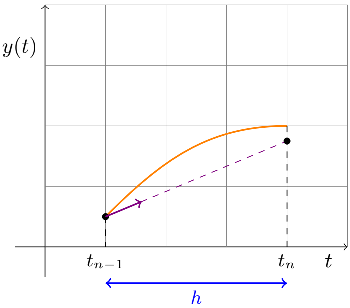
RK methods of order 2: Heun's method

Set $a_2 = 1/2; \implies a_1 = 1/2,\ p_2 = q_{21} = 1.$

$$y_n = y_{n-1} + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)$$

where $k_1 = f(t_{n-1}, y_{n-1})$, and $k_2 = f(\underbrace{t_{n-1} + h}_{t_n}, \underbrace{y_{n-1} + k_1 h}_{y_n^0})$.

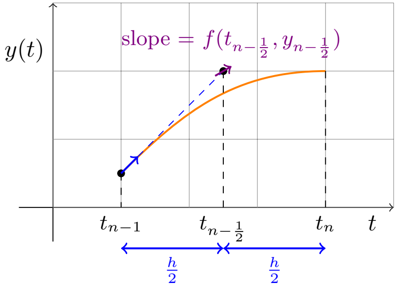


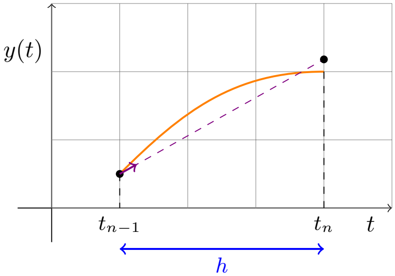


RK methods of order 2: Midpoint Method

Set $a_2 = 1; \implies a_1 = 0, p_2 = q_{21} = 1/2.$

$$y_n = y_{n-1} + hf\left(\underbrace{t_{n-1} + \frac{1}{2}h}_{t_{n-1/2}}, \underbrace{y_{n-1} + \frac{1}{2}f(t_{n-1}, y_{n-1})h}_{y_{n-1/2}}\right)$$





RK methods of order 2, 3 and 4

- Ralston's method,

  $$a_2 = 3/4 \implies a_1 = 1/4,\ p_2 = q_{21} = 2/3.$$

  This method provides the minimum bound on the truncation error for second order RK methods.
- For $s = 3$, we can perform a similar Taylor series analysis, and end up with 6 equations and 8 unknowns.
- As expected, the global error for third order RK methods scales as $\mathcal{O}(h^3)$
- The most popular RK methods are of order 4. As before there are an infinite number of methods.
- The most commonly used is the *classical* 4th order RK method.

Classical 4th order RK Method

$$
\begin{aligned}
y_n &= y_{n-1} + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\
t_n &= t_{n-1} + h
\end{aligned}
$$

where

$$
\begin{aligned}
k_1 &= f(t_{n-1}, y_{n-1}), \\
k_2 &= f(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}k_1), \\
k_3 &= f(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}k_2), \\
k_4 &= f(t_{n-1} + h, y_{n-1} + k_3).
\end{aligned}
$$

Note that when $f(t, y) = f(t)$, this method reduces to Simpson's (1/3) rule for integration.

Adaptive RK Methods

- The idea here is to change the step size, using a smaller h when the function changes quickly, and a larger h when it changes slowly
- These methods use estimates of the LTE to determine how adequate the current step size is, and adjust it to insure a specified LTE
- There are two broad ways of doing this automatically:
  - Use the same RK method with two different step sizes (say h and $h/2$)
  - Use the same step size, but use two different RK methods (say a 2nd order and a 4th order)

Step-Halving

- At a particular step, take the full step (h) and compute the estimate $y(h)$ as y_1
- Repeat with two independent half steps $(h/2)$and compute the estimate $y(h)$ as y_2
- The estimate of the local error $\Delta = y_2 - y_1$ allows you to figure out whether the step size is consistent with the accuracy you demand
- In addition, this information can also be used to refine the estimate
- For example, with RK4, we can estimate a 5th order correction by setting

$$y_2 \leftarrow y_2 + \frac{\Delta}{15}$$

Runge-Kutta Fehlberg

- If we naively use a RK4 and RK5 method with the same h, we need to perform 10 separate function computations over each step.
- Runge-Kutta Fehlberg is a clever method that reuses many of the points computed from RK4, and requires only 6 separate function computations over each step.
- Such RK methods are called embedded-RK methods
- These are extremely popular and are used in Matlab's "ode45" function, for example