

1 Week 04: 16 and 18 September 2003

We now begin the definition and construction of Runge-Kutta methods. These one-step methods are essentially always stable, but designing Runge-Kutta methods which are consistent to high order can be difficult. This theory is presented in Sec. II.2 of [HNW93], Sec. 4.1 of [Sha94], and Chap. 5 of [Lam91].

2 Examples

We have already seen several examples of Runge-Kutta methods: explicit and implicit Euler, the implicit midpoint rule, the explicit midpoint rule with Euler predictor, and so forth. Another example is the trapezoidal rule with Euler predictor: in Runge-Kutta notation it looks like

$$\begin{aligned}k_1 &= f(t_n, u_n) \\k_2 &= f(t_n, u_n + hk_1) \\u_{n+1} &= u_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)\end{aligned}$$

The basic idea is to build a series of “stages” k_i which approximate $y' = f$ at various points, using samples of f from other stages, and approximate $u_{n+1} - u_n$ by a linear combination of the stages. Since each stage can involve the others, the right-hand side f ends up being evaluated in a complicated nonlinear way.

The most famous example is the classical four-stage fourth-order scheme

$$\begin{aligned}k_1 &= f(t_n, u_n) \\k_2 &= f\left(t_n + \frac{1}{2}h, u_n + h\frac{1}{2}k_1\right) \\k_3 &= f\left(t_n + \frac{1}{2}h, u_n + h\frac{1}{2}k_2\right) \\k_4 &= f(t_n + h, u_n + hk_3) \\u_{n+1} &= u_n + h\left(\frac{1}{6}k_1 + \frac{2}{6}k_2 + \frac{2}{6}k_3 + \frac{1}{6}k_4\right).\end{aligned}$$

3 Form and classification

A general Runge–Kutta method has s stages:

$$\begin{aligned} k_1 &= f(t_n + c_1 h, u_n + h(a_{11}k_1 + a_{12}k_2 + \cdots + a_{1s}k_s)) \\ k_2 &= f(t_n + c_2 h, u_n + h(a_{21}k_1 + a_{22}k_2 + \cdots + a_{2s}k_s)) \\ &\dots \\ k_s &= f(t_n + c_s h, u_n + h(a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{ss}k_s)) \\ u_{n+1} &= u_n + h(b_1k_1 + b_2k_2 + \cdots + b_s k_s), \end{aligned}$$

conveniently summarized in a *Butcher array*

$$\frac{c \mid A}{\mid b^T}$$

consisting of a column vector c , an $s \times s$ matrix A and a column vector b . For example, the 2–stage trapezoidal method above has Butcher array

$$\frac{0 \mid 0 \ 0}{\frac{1}{2} \mid \frac{1}{2} \ 0} \\ \mid \frac{1}{2} \ \frac{1}{2}$$

while the classical scheme has

$$\frac{0 \mid 0 \ 0 \ 0 \ 0}{\frac{1}{2} \mid \frac{1}{2} \ 0 \ 0 \ 0} \\ \frac{\frac{1}{2} \mid 0 \ \frac{1}{2} \ 0 \ 0}{1 \mid 0 \ 0 \ 1 \ 0} \\ \mid \frac{1}{6} \ \frac{2}{6} \ \frac{2}{6} \ \frac{1}{6}$$

Various useful properties of the method can be expressed with reference to the matrix A , giving rise to acronyms: for example, an explicit Runge–Kutta (ERK) method is one for which A is strictly lower triangular ($a_{ij} = 0$ for $i \leq j$). All others are implicit (IRK), but there are several useful classes of IRKs.

In general, IRK methods are useful because they can achieve high order with few stages—good results cheap—and they are good for stiff problems. (It’s essentially a theorem that an effective method for stiff problems must be implicit.) ERK methods with s stages can achieve at best order $p = s$,

and for high p even less. For example, a 5th-order ERK must have 6 stages, a 7th-order ERK requires 9 stages, an 8th-order ERK requires 11 stages, and so forth. This is because the number of terms in the Taylor series which must be matched for high order increases with p faster than the number $s(s-1)/2+2s$ of parameters available in an s -stage ERK method. These facts are proved by laborious calculations with the aid of graph theory and sometimes computer algebra [But87]. By contrast, an s -stage IRK method has s^2+2s parameters and can achieve order $p=2s$ easily. (Order higher than $2s$ is impossible with s stages, because it would yield numerical integration with higher-order accuracy than Gauss quadrature when $f(t, y)$ is independent of y .)

The “Butcher theory” of attainable order for explicit Runge–Kutta methods contains essentially three theorems:

Theorem 1 *All Runge–Kutta methods of order up to 4 have the same order on IVPs whether y is a scalar or a vector. For order greater than 4, the order for a system may be lower than for a scalar equation.*

Theorem 2 *Any s -stage ERK has order less than or equal to s .*

Theorem 3 *An ERK method of order p must have at least $p+1$ stages if $p > 4$, $p+2$ stages if $p > 6$, $p+3$ stages if $p > 7$.*

There is a $p=10$ th order method with 17 stages, useful for computations requiring more than about 15-digit accuracy.

The disadvantage of IRK methods is their expense. For a d -dimensional IVP $y' = f(t, y)$ where $y(t) \in \mathbf{R}^d$ for each t , we are resigned to having to solve a $d \times d$ nonlinear system (probably by Newton’s method or a variation of it) to advance any implicit method. But because an IRK method links together s stages k_i , each a vector of dimension d , it requires solution of an $sd \times sd$ nonlinear system. Since each iteration of Newton’s method usually involves solving an $N \times N$ linear system by Gaussian elimination at cost $O(N^3)$, an IRK method requires $O(s^3d^3)$ work per time step. The extra factor of $O(s^3)$ can be substantial if $s \geq 2$ or so.

This suggests the introduction of a class of methods called DIRK for “diagonally implicit Runge–Kutta”. For DIRK methods, A is lower triangular (but not strictly), so each k_i depends only on itself and previously computed k_j ’s with $j \leq i$. Thus we can solve for the stages k_i in sequence rather than all at once. Total cost per time step is reduced to $O(sd^3)$ from $O(s^3d^3)$, a factor of 16 for a four-stage method.

In many situations, the nonlinear equations are solved by modified versions of Newton's method where the Jacobian is frozen for several steps at a time. This saves effort if the linear systems are solved by LU-factorization of the Jacobian J (an equivalent form of Gaussian elimination) because the LU-factors can be reused to solve $Jx = b$ for a new right-hand side b in $O(N^2)$ instead of $O(N^3)$. Since the nonlinear system we solve for k_i has Jacobian $J = I - ha_{ii}Df(t_n + c_i h, u_n + h(\sum_j a_{ij}k_j))$, it is considerably cheaper if all the diagonal matrix elements a_{ii} are identical. Such methods are called singly diagonally implicit (SDIRK) methods.

All restrictions on IRK methods for computational convenience reduce the number of free parameters and therefore lower the attainable order in general, but the amount lost varies greatly with the number of stages involved.

Later we will develop *extrapolation methods* and *deferred corrections methods* which amount to automatic generation of s -stage ERKs with order- p accuracy, where $s = p^2/4 + 1$. These are suboptimally efficient for high s , but require no explicit formula derivation. Similar methods are possible for IRKs and provide excellent high-order stiff solvers.

4 Consistency conditions

Our first task is to derive algebraic conditions guaranteeing that the local truncation error is small.

Let's consider a specific example: derive a 2-stage ERK method of maximum accuracy. Such a method looks like

$$\begin{aligned} k_1 &= f(t_n, u_n) \\ k_2 &= f(t_n + ah, u_n + hak_1) \\ u_{n+1} &= u_n + h(b_1k_1 + b_2k_2) \end{aligned}$$

where $a = c_2 = a_{21}$ for simplicity. The local truncation error is

$$\begin{aligned} \tau_{n+1} &= y_{n+1} - y_n - h(b_1f(t_n, y_n) + b_2f(t_n + ah, y_n + haf(t_n, y_n))) \\ &= y + hy' + \frac{1}{2}h^2y'' + \frac{1}{6}h^3y''' + O(h^4) - y - h(b_1f + b_2f(t_n + ah, y + haf)) \end{aligned}$$

where we have omitted the argument whenever it is the center point $(t_n, y(t_n))$ of our Taylor expansion.

We need to evaluate τ_{n+1} in multidimensional Taylor series, so let's review how to do it and introduce some simpler (or at least shorter) notation. When we differentiate a vector function $f(t, y(t))$ with respect to t , the chain rule says we have to keep track of variation of f with t plus the variation of each component $y_j(t)$ with t :

$$\frac{d}{dt}f_i(t, y(t)) = \frac{\partial f_i}{\partial t}(t, y(t)) + \sum_{j=1}^d \frac{\partial f_i}{\partial y_j}(t, y(t)) \cdot y'_j(t).$$

A simpler notation uses subscripts with commas for partial derivatives of f , omits the argument whenever it is $(t, y(t))$, and introduces the summation convention that repeated subscripts are automatically summed over:

$$\frac{d}{dt}f_i(t, y(t)) = f_{i,t} + f_{i,j}y'_j.$$

With this notation, we can express the derivatives of y in terms of f whenever y is a solution of the ODE:

$$\begin{aligned} y'_i &= f_i, \\ y''_i &= f_{i,t} + f_{i,j}f_j \\ y'''_i &= f_{i,tt} + f_{i,tj}f_j + (f_{i,jt} + f_{i,jk}f_k)f_j + f_{i,j}(f_{j,t} + f_{j,k}f_k). \end{aligned} \quad (1)$$

Next we observe that life will simplify greatly if we consider only *autonomous* ODEs, where f does not depend explicitly on t . This will not cost us very much generality, because any ODE can be transformed to autonomous form by regarding t as another solution component satisfying an additional ODE $t' = 1$ and initial condition $t(0) = 0$. After “autonomization”, all the partial derivatives with respect to t vanish into the crowd of spatial partial derivatives. For our Runge–Kutta method this requires us to treat t the same as all the other variables, which imposes the row sum condition

$$c_i = \sum_{j=1}^s a_{ij}.$$

In other words, the arguments of f approximate $u(t_n + c_i h)$ at time $t_n + c_i h$; there is no mismatch. Now the derivatives of y simplify to

$$\begin{aligned} y'_i &= f_i, \\ y''_i &= f_{i,j}f_j \\ y'''_i &= f_{i,jk}f_kf_j + f_{i,j}f_{j,k}f_k, \end{aligned} \quad (2)$$

a much more manageable collection.

Returning to the local truncation error and dropping the time step index $n + 1$ in favor of the vector component index i , we get

$$\begin{aligned}\tau_i &= y_i + hf_i + \frac{1}{2}h^2 f_{i,j}f_j + \frac{1}{6}h^3(f_{i,jk}f_kf_j + f_{i,j}f_{j,k}f_k) + O(h^4) - y_i \\ &\quad - h(b_1f_i + b_2f_i(y + haf))\end{aligned}$$

where it remains only to expand the final f value. To do this, we use the multidimensional Taylor formula

$$f_i(y + haf) = f_i + f_{i,j}haf_j + \frac{1}{2}f_{i,jk}(ha)^2f_jf_k + O(h^3)$$

to get

$$\begin{aligned}\tau_i &= h(f_i - b_1f_i - b_2f_i) \\ &\quad + h^2\left(\frac{1}{2}f_{i,j}f_j - b_2af_{i,j}f_j\right) \\ &\quad + h^3\left(\frac{1}{6}(f_{i,jk}f_kf_j + f_{i,j}f_{j,k}f_k) - \frac{1}{2}b_2a^2f_{i,jk}f_jf_k\right) + O(h^4).\end{aligned}$$

Clearly the choice $b_1 + b_2 = 1$ is required for first-order or better accuracy, while the choice $b_2a = 1/2$ gives a one-parameter family of second-order accurate ERK methods (the explicit trapezoidal rule we saw above has $b_2 = 1/2$ and $a = 1$).

The $O(h^3)$ term consists of two different “elementary differentials” $f_{i,jk}f_kf_j$ and $f_{i,j}f_{j,k}f_k$ which cannot be both eliminated by choice of b_1 , b_2 and a , so this method will be second order at best.

Exercise 1 *Derive the equations for a 3-stage third-order ERK method and show that they are satisfied by the Heun method*

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

and the Kutta method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

5 Order conditions

Next we will prove convergence and show easier ways to derive order conditions for Runge–Kutta methods. These topics are covered in Sect. II.2 of [HNW93], Sect. 1.2 of [Sha94], Chap. 2 of [Gea71], and Chap. 5 of [Lam91].

Previously we derived a second–order two–stage ERK method by Taylor expansion of the stages and cancelling terms. If we try this approach on a more complex method, or on any implicit method, it rapidly becomes unmanageable because the stages appear on both sides and we are trying to keep track of too many coefficients at once. Instead, we can take the viewpoint that the local truncation error

$$\tau = y(t+h) - y(t) - h \sum_{p=1}^s b_p k_p$$

is a series

$$\tau(h) = C_0 + C_1 h + \frac{1}{2} C_2 h^2 + \cdots + O(h^{p+1})$$

in h , and compute the coefficients C_j by the standard Taylor series formula

$$C_j = \tau^{(j)}(h=0).$$

This treats the errors term by term and simplifies the calculations into manageable pieces. A method is accurate of order p , then, iff $C_0 = C_1 = \cdots = C_p = 0$. These requirements constitute a collection of nonlinear equations called the “order conditions” for Runge–Kutta methods, and are arduous to solve for large s and p .

Consider an s -stage Runge–Kutta method, not necessarily explicit, with Butcher array

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

The i th vector component of the local truncation error is

$$\begin{aligned} \tau_i &= y_i(t+h) - y_i(t) - h \left(\sum_{p=1}^s b_p k_p^i \right) \\ &= y_i + h y_i' + \frac{1}{2} h^2 y_i'' + \frac{1}{6} h^3 y_i''' + \cdots - y_i \\ &\quad - h \left(b_1 k_1^i + \cdots + b_s k_s^i \right). \end{aligned}$$

Differentiating with respect to h gives

$$\begin{aligned}\tau_i'(h) &= y_i' + h y_i'' + \frac{1}{2} h^2 y_i''' + \dots \\ &\quad - (b_1 k_1^i + \dots + b_s k_s^i) \\ &\quad - h (b_1 k_1^{i'} + \dots + b_s k_s^{i'})\end{aligned}$$

and

$$\begin{aligned}\tau_i''(h) &= y_i'' + h y_i''' + \dots \\ &\quad - 2 (b_1 k_1^{i'} + \dots + b_s k_s^{i'}) \\ &\quad - h (b_1 k_1^{i''} + \dots + b_s k_s^{i''}).\end{aligned}$$

Thus we need to evaluate derivatives of the stages with respect to h : we find

$$k_p^{i'}(h) = f_{i,j}(y + h \sum_q a_{pq} k_q) \left(\sum_q a_{pq} k_q^j + h \sum_q a_{pq} k_q^{j'} \right).$$

As $h \rightarrow 0$, we get $k_p^i \rightarrow f_i$ and

$$k_p^{i'}(h) = f_{i,j} \left(\sum_q a_{pq} k_q^j \right) \rightarrow \left(\sum_q a_{pq} \right) f_{i,j} f_j.$$

Thus as $h \rightarrow 0$, the first derivative of the local truncation error converges to

$$C_1 = \tau_i'(h=0) = y_i' - \left(\sum_p b_p \right) f_i = (1 - \sum_p b_p) f_i$$

and the second derivative to

$$C_2 = \tau_i''(h=0) = y_i'' - \left(2 \sum_p b_p \sum_q a_{pq} \right) f_{i,j} f_j = (1 - 2 \sum_p b_p \sum_q a_{pq}) f_{i,j} f_j$$

since $y_i'' = f_{i,j} f_j$.

Thus the first two order conditions, which produce methods of order 2, read

$$\sum_p b_p = 1$$

and

$$2 \sum_p b_p \sum_q a_{pq} = 1.$$

Proceeding similarly, one derives the two third-order conditions

$$3 \sum_p b_p \sum_q a_{pq} \sum_r a_{pr} = 1$$

and

$$6 \sum_p b_p \sum_q a_{pq} \sum_r a_{qr} = 1.$$

The number N of order conditions required for higher-order methods grows rapidly with p ; see Table 1 (copied from [HNW93]).

The eight fourth-order conditions can be satisfied by the “Hammer–Hollingsworth” 2-stage IRK method, even though it has only six free parameters a_{pq} and b_p :

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Table 1: Number N of order conditions as function of order p .

$p = 1$	2	3	4	5	6	7	8	9	10
$N = 1$	2	4	8	17	37	85	200	486	1205

Further simplifications of the order conditions are discussed in detail in [HNW93], but lie beyond the scope of this course. Later we will see how to build Runge–Kutta methods with arbitrarily high order by extrapolation or deferred correction, without ever knowing the Butcher array.

6 Convergence

Let’s check when Runge–Kutta methods satisfy the conditions for order- p convergence. Recall that a one-step method

$$u_{n+1} = u_n + hF(u_{n+1}, u_n; t_{n+1}, h, f)$$

converges with order- p accuracy whenever (a) F is Lipschitz in u_{n+1} and u_n whenever f is Lipschitz, (b) F vanishes identically whenever f does, and (c) the method is consistent of order p (the local truncation error τ is $O(h^{p+1})$). The stability condition (a) is easy to check for Runge–Kutta methods: we need only check that each k_i is a Lipschitz function of u_n . Since

$$k_i = f(t_n + c_i h, u_n + h(a_{i1}k_1 + \cdots + a_{ii}k_i + \cdots + a_{is}k_s)),$$

variation of u_n from u to v varies k_i by

$$\|k_i(u) - k_i(v)\| \leq L(\|u - v\| + h \sum_{j=1}^s |a_{ij}| \|k_j(u) - k_j(v)\|).$$

For *one-stage* methods, we now assume $hL|a_{11}| < 1$ (always true for small h), collect the k_1 terms on the left-hand side of the inequality, and get

$$\|k_1(u) - k_1(v)\| \leq \frac{L}{1 - hL|a_{11}|} \|u - v\|$$

proving stability. For multistage methods, we can do exactly the same calculation except that we have to invert a $s \times s$ matrix $I - hL|A|$ and take the norm of its inverse. Here $|A|$ is the matrix with elements $|a_{ij}|$. The matrix $I - h|A|$ is always invertible for sufficiently small h , so the result is the same:

$$\|k(u) - k(v)\| \leq \frac{L}{1 - hL\| |A| \|} \|u - v\|$$

where we have used the standard result

$$\|(I - F)^{-1}\| \leq \frac{1}{1 - \|F\|}$$

for any matrix F with norm less than 1. Thus Runge–Kutta methods are always stable, and their order of accuracy is completely determined by the local truncation error.

Exercise 2 Express the following method as a Runge–Kutta method, write down the Butcher array, and determine the order of accuracy:

$$y_{n+2/3} = y_n + \frac{h}{3} (f(y_{n+2/3}) + f(y_n)),$$

$$y_{n+1} = y_n + \frac{h}{4} (3f(y_{n+2/3}) + f(y_n)).$$

Exercise 3 Use N steps of the trapezoidal rule with constant step size $h = H/N$ to advance the numerical solution u_n one big step to time $t_{n+1} = t_n + H$. Consider the big step $u_n \rightarrow u_{n+1}$ as a Runge-Kutta method. Write down its Butcher array for the cases $N = 1$, $N = 2$ and general N . Verify the order of accuracy by checking that the appropriate order conditions are satisfied.

Next week, we will discuss the linear stability theory of Runge-Kutta methods (Chap. IV of [HNW96]).

References

- [But87] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. John Wiley and Sons, New York, 1987.
- [Gea71] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.
- [HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential equations I : Nonstiff problems*. Springer-Verlag, second revised edition, 1993.
- [HNW96] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential equations II : Stiff problems*. Springer-Verlag, second revised edition, 1996.
- [Lam91] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley and Sons, 1991.
- [Sha94] L. F. Shampine. *Numerical solution of ordinary differential equations*. Chapman and Hall, New York, 1994.