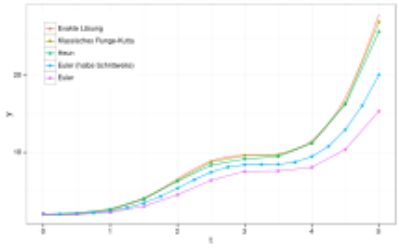
WIKIPEDIA

Runge–Kutta methods

In numerical analysis, the **Runge–Kutta methods** (English: /ˈrʊŋəˈkʊtɑː/ (listen) *RUUNG-ə-KUUT-tah*[1]) are a family of implicit and explicit iterative methods, which include the well-known routine called the Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations.[2] These methods were developed around 1900 by the German mathematicians Carl Runge and Wilhelm Kutta.

Contents



Comparison of the Runge-Kutta methods for the differential equation $y' = sin(t)^2 * y$ (red is the exact solution)

The Runge–Kutta method

The most widely known member of the Runge–Kutta family is generally referred to as "RK4", the "classic Runge–Kutta method" or simply as "the Runge–Kutta method".

Let an initial value problem be specified as follows:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

Here y is an unknown function (scalar or vector) of time t, which we would like to approximate; we are told that $\frac{dy}{dt}$, the rate at which y changes, is a function of t and of y itself. At the initial time t_0 the corresponding y value is y_0. The function f and the initial conditions t_0, y_0 are given.

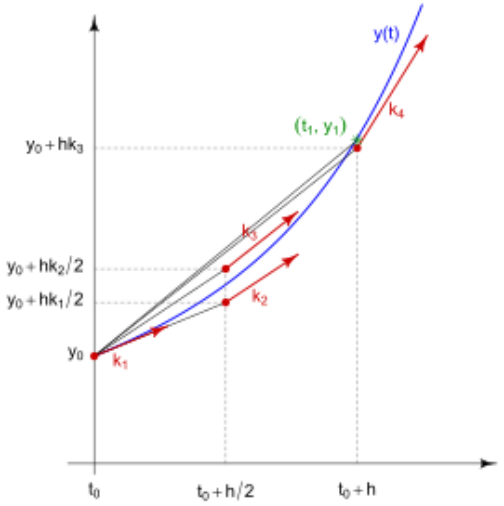
Now pick a step-size $h > 0$ and define

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

for *n* = 0, 1, 2, 3, ..., using[3]

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$

(Note: the above equations have different but equivalent definitions in different texts).[4]



Slopes used by the classical Runge-Kutta method

Here y_{n+1} is the RK4 approximation of $y(t_{n+1})$, and the next value (y_{n+1}) is determined by the present value (y_n) plus the weighted average of four increments, where each increment is the product of the size of the interval, *h*, and an estimated slope specified by function *f* on the right-hand side of the differential equation.

- k_1 is the slope at the beginning of the interval, using y (Euler's method);
- k_2 is the slope at the midpoint of the interval, using y and k_1;
- k_3 is again the slope at the midpoint, but now using y and k_2;
- k_4 is the slope at the end of the interval, using y and k_3.

In averaging the four slopes, greater weight is given to the slopes at the midpoint. If f is independent of y, so that the differential equation is equivalent to a simple integral, then RK4 is Simpson's rule.[5]

The RK4 method is a fourth-order method, meaning that the local truncation error is on the order of $O(h^5)$, while the total accumulated error is on the order of $O(h^4)$.

In many practical applications the function f is independent of t (so called autonomous system, or time-invariant system, especially in physics), and their increments are not computed at all and not passed to function f, with only the final formula for t_{n+1} used.

Explicit Runge–Kutta methods

The family of explicit Runge–Kutta methods is a generalization of the RK4 method mentioned above. It is given by

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i,$$

where[6]

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + c_2 h, y_n + h(a_{21}k_1)),$$
$$k_3 = f(t_n + c_3 h, y_n + h(a_{31}k_1 + a_{32}k_2)),$$
$$\vdots$$
$$k_s = f(t_n + c_s h, y_n + h(a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1})).$$

(Note: the above equations may have different but equivalent definitions in some texts).[4]

To specify a particular method, one needs to provide the integer s (the number of stages), and the coefficients a_{ij} (for $1 \le j < i \le s$), b_i (for $i = 1, 2, ..., s$) and c_i (for $i = 2, 3, ..., s$). The matrix $[a_{ij}]$ is called the *Runge–Kutta matrix*, while the b_i and c_i are known as the *weights* and the *nodes*.[7] These data are usually arranged in a mnemonic device, known as a *Butcher tableau* (after John C. Butcher):

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \end{array}$$

A Taylor series expansion shows that the Runge–Kutta method is consistent if and only if

$$\sum_{i=1}^{s} b_i = 1.$$

There are also accompanying requirements if one requires the method to have a certain order p, meaning that the local truncation error is $O(h^{p+1})$. These can be derived from the definition of the truncation error itself. For example, a two-stage method has order 2 if $b_1 + b_2 = 1$, $b_2 c_2 = 1/2$, and $b_2 a_{21} = 1/2$.[8] Note that a popular condition for determining coefficients is [9]

$$\sum_{j=1}^{i-1} a_{ij} = c_i \text{ for } i = 2, \ldots, s.$$

This condition alone, however, is neither sufficient, nor necessary for consistency. [10]

In general, if an explicit s-stage Runge–Kutta method has order p, then it can be proven that the number of stages must satisfy $s \ge p$, and if $p \ge 5$, then $s \ge p + 1$.[11] However, it is not known whether these bounds are *sharp* in all cases; for example, all known methods of order 8 have at least 11 stages, though it is possible that there are methods with fewer stages. (The bound above suggests that there could be a method with 9 stages; but it could also be that the bound is simply not sharp.) Indeed, it is an open problem what the precise minimum number of stages s is for an explicit Runge–Kutta method to have order p in those cases where no methods have yet been discovered that satisfy the bounds above with equality. Some values which are known are:[12]

p	1	2	3	4	5	6	7	8
$\min s$	1	2	3	4	6	7	9	11

The provable bounds above then imply that we can not find methods of orders $p = 1, 2, \ldots, 6$ that require fewer stages than the methods we already know for these orders. However, it is conceivable that we might find a method of order $p = 7$ that has only 8 stages, whereas the only ones known today have at least 9 stages as shown in the table.

Examples

The RK4 method falls in this framework. Its tableau is[13]

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ \end{array}$$

1	0	0	1	
	1/6	1/3	1/3	1/6

A slight variation of "the" Runge–Kutta method is also due to Kutta in 1901 and is called the 3/8-rule.[14] The primary advantage this method has is that almost all of the error coefficients are smaller than in the popular method, but it requires slightly more FLOPs (floating-point operations) per time step. Its Butcher tableau is

0				
1/3	1/3			
2/3	-1/3	1		
1	1	−1	1	
	1/8	3/8	3/8	1/8

However, the simplest Runge–Kutta method is the (forward) Euler method, given by the formula $y_{n+1} = y_n + hf(t_n, y_n)$. This is the only consistent explicit Runge–Kutta method with one stage. The corresponding tableau is

0	
	1

Second-order methods with two stages

An example of a second-order method with two stages is provided by the midpoint method:

$$y_{n+1} = y_n + hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n,\ y_n)\right).$$

The corresponding tableau is

0		
1/2	1/2	
	0	1

The midpoint method is not the only second-order Runge–Kutta method with two stages; there is a family of such methods, parameterized by α and given by the formula[15]

$$y_{n+1} = y_n + h\left((1 - \tfrac{1}{2\alpha})f(t_n, y_n) + \tfrac{1}{2\alpha}f(t_n + \alpha h, y_n + \alpha h f(t_n, y_n))\right).$$

Its Butcher tableau is

0		
α	α	
	$(1 - \frac{1}{2\alpha})$	$\frac{1}{2\alpha}$

In this family, $\alpha = \frac{1}{2}$ gives the midpoint method, and $\alpha = 1$ is Heun's method.[5]

Use

As an example, consider the two-stage second-order Runge–Kutta method with α = 2/3, also known as Ralston method. It is given by the tableau

0	
2/3	2/3
	1/4 3/4

with the corresponding equations

$$\begin{aligned} k_1 &= f(t_n,\ y_n), \\ k_2 &= f(t_n + \tfrac{2}{3}h,\ y_n + \tfrac{2}{3}hk_1), \\ y_{n+1} &= y_n + h\left(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2\right). \end{aligned}$$

This method is used to solve the initial-value problem

$$\frac{dy}{dt} = \tan(y) + 1, \quad y_0 = 1,\ t \in [1, 1.1]$$

with step size $h = 0.025$, so the method needs to take four steps.

The method proceeds as follows:

$t_0 = 1$:

$$y_0 = 1$$

$t_1 = 1.025$:

$$y_0 = 1 \qquad k_1 = 2.557407725 \quad k_2 = f(t_0 + \tfrac{2}{3}h,\ y_0 + \tfrac{2}{3}hk_1) = 2.7138981400$$

$$y_1 = y_0 + h(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2) = \underline{1.066869388}$$

$t_2 = 1.05$:

$$y_1 = 1.066869388 \quad k_1 = 2.813524695 \quad k_2 = f(t_1 + \tfrac{2}{3}h,\ y_1 + \tfrac{2}{3}hk_1)$$

$$y_2 = y_1 + h(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2) = \underline{1.141332181}$$

$t_3 = 1.075$:

$$y_2 = 1.141332181 \quad k_1 = 3.183536647 \quad k_2 = f(t_2 + \tfrac{2}{3}h,\ y_2 + \tfrac{2}{3}hk_1)$$

$$y_3 = y_2 + h(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2) = \underline{1.227417567}$$

$t_4 = 1.1$:

$$y_3 = 1.227417567 \quad k_1 = 3.796866512 \quad k_2 = f(t_3 + \tfrac{2}{3}h,\ y_3 + \tfrac{2}{3}hk_1)$$

$$y_4 = y_3 + h(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2) = \underline{1.335079087}.$$

The numerical solutions correspond to the underlined values.

Adaptive Runge–Kutta methods

Adaptive methods are designed to produce an estimate of the local truncation error of a single Runge–Kutta step. This is done by having two methods, one with order p and one with order $p-1$. These methods are interwoven, i.e., they have common intermediate steps. Thanks to this, estimating the error has little or negligible computational cost compared to a step with the higher-order method.

During the integration, the step size is adapted such that the estimated error stays below a user-defined threshold: If the error is too high, a step is repeated with a lower step size; if the error is much smaller, the step size is increased to save time. This results in an (almost) optimal step size, which saves computation time. Moreover, the user does not have to spend time on finding an appropriate step size.

The lower-order step is given by

$$y^*_{n+1} = y_n + h\sum_{i=1}^{s} b^*_i k_i,$$

where k_i are the same as for the higher-order method. Then the error is

$$e_{n+1} = y_{n+1} - y^*_{n+1} = h\sum_{i=1}^{s}(b_i - b^*_i)k_i,$$

which is $O(h^p)$. The Butcher tableau for this kind of method is extended to give the values of b^*_i:

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & b^*_1 & b^*_2 & \cdots & b^*_{s-1} & b^*_s \end{array}$$

The Runge–Kutta–Fehlberg method has two methods of orders 5 and 4. Its extended Butcher tableau is:

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	−7200/2197	7296/2197			
1	439/216	−8	3680/513	-845/4104		
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	
	16/135	0	6656/12825	28561/56430	−9/50	2/55
	25/216	0	1408/2565	2197/4104	−1/5	0

However, the simplest adaptive Runge–Kutta method involves combining Heun's method, which is order 2, with the Euler method, which is order 1. Its extended Butcher tableau is:

0		
1	1	
	1/2	1/2
	1	0

Other adaptive Runge–Kutta methods are the Bogacki–Shampine method (orders 3 and 2), the Cash–Karp method and the Dormand–Prince method (both with orders 5 and 4).

Nonconfluent Runge–Kutta methods

A Runge–Kutta method is said to be *nonconfluent* [16] if all the $c_i,\ i = 1, 2, \ldots, s$ are distinct.

Runge–Kutta–Nyström methods

Runge–Kutta–Nyström methods are specialized Runge-Kutta methods that are optimized for second-order differential equations of the following form:[17][18]

$$\frac{d^2y}{dt^2} = f(y, \dot{y}, t).$$

Implicit Runge–Kutta methods

All Runge–Kutta methods mentioned up to now are explicit methods. Explicit Runge–Kutta methods are generally unsuitable for the solution of stiff equations because their region of absolute stability is small; in particular, it is bounded.[19] This issue is especially important in the solution of partial differential equations.

The instability of explicit Runge–Kutta methods motivates the development of implicit methods. An implicit Runge–Kutta method has the form

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i,$$

where

$$k_i = f\left(t_n + c_i h,\ y_n + h\sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, \ldots, s.$$ [20]

The difference with an explicit method is that in an explicit method, the sum over *j* only goes up to *i* − 1. This also shows up in the Butcher tableau: the coefficient matrix a_{ij} of an explicit method is lower triangular. In an implicit method, the sum over *j* goes up to *s* and the coefficient matrix is not triangular, yielding a Butcher tableau of the form[13]

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \\ & b_1^* & b_2^* & \ldots & b_s^* \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b^T} \end{array}$$

See Adaptive Runge-Kutta methods above for the explanation of the b^* row.

The consequence of this difference is that at every step, a system of algebraic equations has to be solved. This increases the computational cost considerably. If a method with *s* stages is used to solve a differential equation with *m* components, then the system of algebraic equations has *ms* components. This can be contrasted with implicit

linear multistep methods (the other big family of methods for ODEs): an implicit *s*-step linear multistep method needs to solve a system of algebraic equations with only *m* components, so the size of the system does not increase as the number of steps increases.[21]

Examples

The simplest example of an implicit Runge–Kutta method is the backward Euler method:

$$y_{n+1} = y_n + hf(t_n + h,\ y_{n+1}).$$

The Butcher tableau for this is simply:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

This Butcher tableau corresponds to the formulae

$$k_1 = f(t_n + h,\ y_n + hk_1) \quad \text{and} \quad y_{n+1} = y_n + hk_1,$$

which can be re-arranged to get the formula for the backward Euler method listed above.

Another example for an implicit Runge–Kutta method is the trapezoidal rule. Its Butcher tableau is:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$

The trapezoidal rule is a collocation method (as discussed in that article). All collocation methods are implicit Runge–Kutta methods, but not all implicit Runge–Kutta methods are collocation methods.[22]

The Gauss–Legendre methods form a family of collocation methods based on Gauss quadrature. A Gauss–Legendre method with *s* stages has order 2*s* (thus, methods with arbitrarily high order can be constructed).[23] The method with two stages (and thus order four) has Butcher tableau:

$$\begin{array}{c|cc} \frac{1}{2} - \frac{1}{6}\sqrt{3} & \frac{1}{4} & \frac{1}{4} - \frac{1}{6}\sqrt{3} \\ \frac{1}{2} + \frac{1}{6}\sqrt{3} & \frac{1}{4} + \frac{1}{6}\sqrt{3} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} + \frac{1}{2}\sqrt{3} & \frac{1}{2} - \frac{1}{2}\sqrt{3} \end{array}$$ [21]

Stability

The advantage of implicit Runge–Kutta methods over explicit ones is their greater stability, especially when applied to stiff equations. Consider the linear test equation $y' = \lambda y$. A Runge–Kutta method applied to this equation reduces to the iteration $y_{n+1} = r(h\lambda)\, y_n$, with *r* given by

$$r(z) = 1 + zb^T (I - zA)^{-1} e = \frac{\det(I - zA + zeb^T)}{\det(I - zA)},$$ [24]

where *e* stands for the vector of ones. The function *r* is called the *stability function*.[25] It follows from the formula that *r* is the quotient of two polynomials of degree *s* if the method has *s* stages. Explicit methods have a strictly lower triangular matrix *A*, which implies that $\det(I - zA) = 1$ and that the stability function is a polynomial.[26]

The numerical solution to the linear test equation decays to zero if $|r(z)| < 1$ with $z = h\lambda$. The set of such *z* is called the *domain of absolute stability*. In particular, the method is said to be absolute stable if all *z* with $\mathrm{Re}(z) < 0$ are in the domain of absolute stability. The stability function of an explicit Runge–Kutta method is a polynomial, so explicit Runge–Kutta methods can never be A-stable.[26]

If the method has order *p*, then the stability function satisfies $r(z) = \mathrm{e}^z + O(z^{p+1})$ as $z \to 0$. Thus, it is of interest to study quotients of polynomials of given degrees that approximate the exponential function the best. These are known as Padé approximants. A Padé approximant with numerator of degree *m* and denominator of degree *n* is A-stable if and only if $m \le n \le m + 2$.[27]

The Gauss–Legendre method with *s* stages has order 2*s*, so its stability function is the Padé approximant with $m = n = s$. It follows that the method is A-stable.[28] This shows that A-stable Runge–Kutta can have arbitrarily high order. In contrast, the order of A-stable linear multistep methods cannot exceed two.[29]

B-stability

The *A-stability* concept for the solution of differential equations is related to the linear autonomous equation $y' = \lambda y$. Dahlquist proposed the investigation of stability of numerical schemes when applied to nonlinear systems that satisfy a monotonicity condition. The corresponding concepts were defined as *G-stability* for multistep methods (and the related one-leg methods) and *B-stability* (Butcher, 1975) for Runge–Kutta methods. A Runge–Kutta method applied to the non-linear system $y' = f(y)$, which verifies , is called *B-stable*, if this condition implies $\|y_{n+1} - z_{n+1}\| \le \|y_n - z_n\|$ for two numerical solutions.

Let B, M and Q be three $s \times s$ matrices defined by

A Runge–Kutta method is said to be *algebraically stable* [30] if the matrices B and M are both non-negative definite. A sufficient condition for *B-stability* [31] is: B and Q are non-negative definite.

Derivation of the Runge–Kutta fourth-order method

In general a Runge–Kutta method of order s can be written as:

$$y_{t+h} = y_t + h \cdot \sum_{i=1}^{s} a_i k_i + \mathcal{O}(h^{s+1}),$$

where:

$$k_i = y_t + h \cdot \sum_{j=1}^{s} \beta_{ij} f\left(k_j,\ t_n + \alpha_i h\right)$$

are increments obtained evaluating the derivatives of y_t at the i-th order.

We develop the derivation[32] for the Runge–Kutta fourth-order method using the general formula with $s = 4$ evaluated, as explained above, at the starting point, the midpoint and the end point of any interval $(t,\ t + h)$; thus, we choose:

$$
\begin{array}{ll}
\alpha_i & \beta_{ij} \\
\alpha_1 = 0 & \beta_{21} = \dfrac{1}{2} \\
\alpha_2 = \dfrac{1}{2} & \beta_{32} = \dfrac{1}{2} \\
\alpha_3 = \dfrac{1}{2} & \beta_{43} = 1 \\
\alpha_4 = 1 &
\end{array}
$$

and $\beta_{ij} = 0$ otherwise. We begin by defining the following quantities:

$$
\begin{aligned}
y^1_{t+h} &= y_t + hf\left(y_t,\, t\right) \\
y^2_{t+h} &= y_t + hf\left(y^1_{t+h/2},\, t + \frac{h}{2}\right) \\
y^3_{t+h} &= y_t + hf\left(y^2_{t+h/2},\, t + \frac{h}{2}\right)
\end{aligned}
$$

where $y^1_{t+h/2} = \dfrac{y_t + y^1_{t+h}}{2}$ and $y^2_{t+h/2} = \dfrac{y_t + y^2_{t+h}}{2}$. If we define:

$$
\begin{aligned}
k_1 &= f(y_t,\, t) \\
k_2 &= f\left(y^1_{t+h/2},\, t + \frac{h}{2}\right) = f\left(y_t + \frac{h}{2}k_1,\, t + \frac{h}{2}\right) \\
k_3 &= f\left(y^2_{t+h/2},\, t + \frac{h}{2}\right) = f\left(y_t + \frac{h}{2}k_2,\, t + \frac{h}{2}\right) \\
k_4 &= f\left(y^3_{t+h},\, t + h\right) = f\left(y_t + hk_3,\, t + h\right)
\end{aligned}
$$

and for the previous relations we can show that the following equalities hold up to $\mathcal{O}(h^2)$:

$$
\begin{aligned}
k_2 &= f\left(y^1_{t+h/2},\, t + \frac{h}{2}\right) = f\left(y_t + \frac{h}{2}k_1,\, t + \frac{h}{2}\right) \\
&= f\left(y_t,\, t\right) + \frac{h}{2}\frac{d}{dt}f\left(y_t,\, t\right) \\
k_3 &= f\left(y^2_{t+h/2},\, t + \frac{h}{2}\right) = f\left(y_t + \frac{h}{2}f\left(y_t + \frac{h}{2}k_1,\, t + \frac{h}{2}\right),\, t + \frac{h}{2}\right) \\
&= f\left(y_t,\, t\right) + \frac{h}{2}\frac{d}{dt}\left[f\left(y_t,\, t\right) + \frac{h}{2}\frac{d}{dt}f\left(y_t,\, t\right)\right] \\
k_4 &= f\left(y^3_{t+h},\, t + h\right) = f\left(y_t + hf\left(y_t + \frac{h}{2}k_2,\, t + \frac{h}{2}\right),\, t + h\right) \\
&= f\left(y_t + hf\left(y_t + \frac{h}{2}f\left(y_t + \frac{h}{2}f\left(y_t,\, t\right),\, t + \frac{h}{2}\right),\, t + \frac{h}{2}\right),\, t + h\right) \\
&= f\left(y_t,\, t\right) + h\frac{d}{dt}\left[f\left(y_t,\, t\right) + \frac{h}{2}\frac{d}{dt}\left[f\left(y_t,\, t\right) + \frac{h}{2}\frac{d}{dt}f\left(y_t,\, t\right)\right]\right]
\end{aligned}
$$

where:

$$
\frac{d}{dt}f(y_t,\, t) = \frac{\partial}{\partial y}f(y_t,\, t)\dot{y}_t + \frac{\partial}{\partial t}f(y_t,\, t) = f_y(y_t,\, t)\dot{y} + f_t(y_t,\, t) := \ddot{y}_t
$$

is the total derivative of f with respect to time.

If we now express the general formula using what we just derived we obtain:

$$\begin{aligned} y_{t+h} &= y_t + h\left\{a\cdot f(y_t,\,t) + b\cdot\left[f(y_t,\,t) + \frac{h}{2}\frac{d}{dt}f(y_t,\,t)\right] + \right. \\ &\quad + c\cdot\left[f(y_t,\,t) + \frac{h}{2}\frac{d}{dt}\left[f\left(y_t,\,t\right) + \frac{h}{2}\frac{d}{dt}f(y_t,\,t)\right]\right] + \\ &\quad \left. + d\cdot\left[f(y_t,\,t) + h\frac{d}{dt}\left[f(y_t,\,t) + \frac{h}{2}\frac{d}{dt}\left[f(y_t,\,t) + \frac{h}{2}\frac{d}{dt}f(y_t,\,t)\right]\right]\right]\right\} + \mathcal{O}(h^5) \\ &= y_t + a\cdot hf_t + b\cdot hf_t + b\cdot\frac{h^2}{2}\frac{df_t}{dt} + c\cdot hf_t + c\cdot\frac{h^2}{2}\frac{df_t}{dt} + \\ &\quad + c\cdot\frac{h^3}{4}\frac{d^2f_t}{dt^2} + d\cdot hf_t + d\cdot h^2\frac{df_t}{dt} + d\cdot\frac{h^3}{2}\frac{d^2f_t}{dt^2} + d\cdot\frac{h^4}{4}\frac{d^3f_t}{dt^3} + \mathcal{O}(h^5) \end{aligned}$$

and comparing this with the Taylor series of y_{t+h} around y_t:

$$\begin{aligned} y_{t+h} &= y_t + h\dot{y}_t + \frac{h^2}{2}\ddot{y}_t + \frac{h^3}{6}y_t^{(3)} + \frac{h^4}{24}y_t^{(4)} + \mathcal{O}(h^5) = \\ &= y_t + hf(y_t,\,t) + \frac{h^2}{2}\frac{d}{dt}f(y_t,\,t) + \frac{h^3}{6}\frac{d^2}{dt^2}f(y_t,\,t) + \frac{h^4}{24}\frac{d^3}{dt^3}f(y_t,\,t) \end{aligned}$$

we obtain a system of constraints on the coefficients:

$$\begin{cases} a+b+c+d = 1 \\ \frac{1}{2}b + \frac{1}{2}c + d = \frac{1}{2} \\ \frac{1}{4}c + \frac{1}{2}d = \frac{1}{6} \\ \frac{1}{4}d = \frac{1}{24} \end{cases}$$

which when solved gives $a = \frac{1}{6}, b = \frac{1}{3}, c = \frac{1}{3}, d = \frac{1}{6}$ as stated above.

See also

- Euler's method
- List of Runge–Kutta methods
- Numerical methods for ordinary differential equations
- Runge–Kutta method (SDE)
- General linear methods
- Lie group integrator

Notes

1. "Runge-Kutta method" (https://www.dictionary.com/browse/runge-kutta-method). *Dictionary.com*. Retrieved 4 April 2021.
2. DEVRIES, Paul L. ; HASBUN, Javier E. A first course in computational physics. Second edition. Jones and Bartlett Publishers: 2011. p. 215.
3. Press et al. 2007, p. 908; Süli & Mayers 2003, p. 328
4. Atkinson (1989, p. 423), Hairer, Nørsett & Wanner (1993, p. 134), Kaw & Kalu (2008, §8.4) and Stoer & Bulirsch (2002, p. 476) leave out the factor *h* in the definition of the stages. Ascher & Petzold (1998, p. 81), Butcher (2008, p. 93) and Iserles (1996, p. 38) use the *y* values as stages.
5. Süli & Mayers 2003, p. 328

6. Press et al. 2007, p. 907
7. Iserles 1996, p. 38
8. Iserles 1996, p. 39
9. Iserles 1996, p. 39
10. As a counterexample, consider any explicit 2-stage Runge-Kutta scheme with $b_1 = b_2 = 1/2$ and c_1 and a_{21} randomly chosen. This method is consistent and (in general) first-order convergent. On the other hand, the 1-stage method with $b_1 = 1/2$ is inconsistent and fails to converge, even though it trivially holds that $\sum_{j=1}^{i-1} a_{ij} = c_i \text{ for } i = 2, \dots, s.$.
11. Butcher 2008, p. 187
12. Butcher 2008, pp. 187–196
13. Süli & Mayers 2003, p. 352
14. Hairer, Nørsett & Wanner (1993, p. 138) refer to Kutta (1901).
15. Süli & Mayers 2003, p. 327
16. Lambert 1991, p. 278
17. Dormand, J. R.; Prince, P. J. (October 1978). "New Runge–Kutta Algorithms for Numerical Simulation in Dynamical Astronomy". *Celestial Mechanics*. **18** (3): 223–232. Bibcode:1978CeMec..18..223D (https://ui.adsabs.harvard.edu/abs/1978CeMec..18..223D). doi:10.1007/BF01230162 (https://doi.org/10.1007%2FBF01230162).
18. Fehlberg, E. (October 1974). Classical seventh-, sixth-, and fifth-order Runge–Kutta–Nyström formulas with stepsize control for general second-order differential equations (Report) (NASA TR R-432 ed.). Marshall Space Flight Center, AL: National Aeronautics and Space Administration.
19. Süli & Mayers 2003, pp. 349–351
20. Iserles 1996, p. 41; Süli & Mayers 2003, pp. 351–352
21. Süli & Mayers 2003, p. 353
22. Iserles 1996, pp. 43–44
23. Iserles 1996, p. 47
24. Hairer & Wanner 1996, pp. 40–41
25. Hairer & Wanner 1996, p. 40
26. Iserles 1996, p. 60
27. Iserles 1996, pp. 62–63
28. Iserles 1996, p. 63
29. This result is due to Dahlquist (1963).
30. Lambert 1991, p. 275
31. Lambert 1991, p. 274
32. PDF (http://www.ss.ncu.edu.tw/~lyu/lecture_files_en/lyu_NSSP_Notes/Lyu_NSSP_AppendixC.pdf) reporting this derivation

References

- Runge, Carl David Tolmé (1895), "Über die numerische Auflösung von Differentialgleichungen" (https://zenodo.org/record/2178704), *Mathematische Annalen*, Springer, **46** (2): 167–178, doi:10.1007/BF01446807 (https://doi.org/10.1007%2FBF01446807).
- Kutta, Martin (1901), "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen", *Zeitschrift für Mathematik und Physik*, **46**: 435–453.
- Ascher, Uri M.; Petzold, Linda R. (1998), *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, Philadelphia: Society for Industrial and Applied Mathematics, ISBN 978-0-89871-412-8.
- Atkinson, Kendall A. (1989), *An Introduction to Numerical Analysis* (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-50023-0.

- Butcher, John C. (May 1963), "Coefficients for the study of Runge-Kutta integration processes", *Journal of the Australian Mathematical Society*, **3** (2): 185–201, doi:10.1017/S1446788700027932 (https://doi.org/10.1017%2FS1446788700027932).
- Butcher, John C. (1975), "A stability property of implicit Runge-Kutta methods", *BIT*, **15** (4): 358–361, doi:10.1007/bf01931672 (https://doi.org/10.1007%2Fbf01931672).
- Butcher, John C. (2008), *Numerical Methods for Ordinary Differential Equations*, New York: John Wiley & Sons, ISBN 978-0-470-72335-7.
- Cellier, F.; Kofman, E. (2006), *Continuous System Simulation*, Springer Verlag, ISBN 0-387-26102-8.
- Dahlquist, Germund (1963), "A special stability problem for linear multistep methods", *BIT*, **3**: 27–43, doi:10.1007/BF01963532 (https://doi.org/10.1007%2FBF01963532), hdl:10338.dmlcz/103497 (https://hdl.handle.net/10338.dmlcz%2F103497), ISSN 0006-3835 (https://www.worldcat.org/issn/0006-3835).
- Forsythe, George E.; Malcolm, Michael A.; Moler, Cleve B. (1977), *Computer Methods for Mathematical Computations*, Prentice-Hall (see Chapter 6).
- Hairer, Ernst; Nørsett, Syvert Paul; Wanner, Gerhard (1993), *Solving ordinary differential equations I: Nonstiff problems*, Berlin, New York: Springer-Verlag, ISBN 978-3-540-56670-0.
- Hairer, Ernst; Wanner, Gerhard (1996), *Solving ordinary differential equations II: Stiff and differential-algebraic problems* (2nd ed.), Berlin, New York: Springer-Verlag, ISBN 978-3-540-60452-5.
- Iserles, Arieh (1996), *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, ISBN 978-0-521-55655-2.
- Lambert, J.D (1991), *Numerical Methods for Ordinary Differential Systems. The Initial Value Problem*, John Wiley & Sons, ISBN 0-471-92990-5
- Kaw, Autar; Kalu, Egwu (2008), *Numerical Methods with Applications* (http://numericalmethods.eng.usf.edu/topics/textbook_index.html) (1st ed.), autarkaw.com.
- Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P. (2007), "Section 17.1 Runge-Kutta Method" (http://apps.nrbook.com/empanel/index.html#pg=907), *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), Cambridge University Press, ISBN 978-0-521-88068-8. Also, Section 17.2. Adaptive Stepsize Control for Runge-Kutta (http://apps.nrbook.com/empanel/index.html#pg=910).
- Stoer, Josef; Bulirsch, Roland (2002), *Introduction to Numerical Analysis* (3rd ed.), Berlin, New York: Springer-Verlag, ISBN 978-0-387-95452-3.
- Süli, Endre; Mayers, David (2003), *An Introduction to Numerical Analysis*, Cambridge University Press, ISBN 0-521-00794-1.
- Tan, Delin; Chen, Zheng (2012), "On A General Formula of Fourth Order Runge-Kutta Method" (http://msme.us/2012-2-1.pdf) (PDF), *Journal of Mathematical Science & Mathematics Education*, **7** (2): 1–10.
- advance discrete maths ignou reference book (code- mcs033)
- John C. Butcher: "B-Series : Algebraic Analysis of Numerical Methods", Springer（SSCM, volume 55), ISBN 978-3030709556 (April, 2021).

External links

- "Runge-Kutta method" (https://www.encyclopediaofmath.org/index.php?title=Runge-Kutta_method), *Encyclopedia of Mathematics*, EMS Press, 2001 [1994]
- Runge–Kutta 4th-Order Method (http://numericalmethods.eng.usf.edu/topics/runge_kutta_4th_method.html)
- Tracker Component Library Implementation in Matlab (https://github.com/USNavalResearchLaboratory/TrackerComponentLibrary/tree/master/Mathematical_Functions/Differential_Equations) — Implements 32 embedded Runge Kutta algorithms in `RungeKStep`, 24 embedded Runge-Kutta Nyström algorithms in `RungeKNystroemSStep` and 4 general Runge-Kutta Nyström algorithms in `RungeKNystroemGStep`.