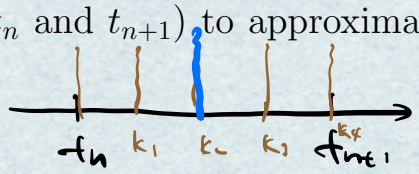
3 Runge-Kutta Methods

3.1 Introduction

The Runge-Kutta methods are a family of methods that are based on the idea of using several stages to approximate the integral in (2). Indeed both Euler's method and Heun's method are special cases of the Runge-Kutta methods.

The Runge-Kutta method for advancing from t_n to t_{n+1} is to use multiple stages (f at intermiediate time points between t_n and t_{n+1}) to approximate the integral in (2), and it has the general form



$$y(t_{n+1}) \approx y(t_n) + h \sum_{i=1}^{s} b_i f_i, \tag{7}$$

where f_i are the slopes (f) at the stages $t_n + c_i h$, and b_i are the weights. The slopes f_i are computed as

$$f_i = f(t_n + c_i h, y(t_n) + h \sum_{j=1}^{i-1} a_{ij} f_j),$$

where a_{ij} are the coefficients (note to make sure k_i are at the right estimates, we need to ensure $c_i = \sum_{j=1}^{i-1} a_{ij}$). The Runge-Kutta method is explicit if the coefficients a_{ij} are zero for $i \geq j$, and it is implicit if the coefficients a_{ij} are non-zero for $i \geq j$. It is important to note that the sumation in (7) is to approximate the integral in (2): $y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))$.

To approximate the integral, we have the goal (the Runge-Kutta method) to use function evaluations

$$f(t_h a, y + hb)$$

to replace the derivatives $y_n'', y'''n, \cdots$ and get a method with the desired order of accuracy. We note the number of stages as explained above is the number of function evaluations in the formula. Why? Let's see.

Recall that Euler's method was derived by expanding y_{n+1} in a Taylor series:

$$y_{n+1} = y_n + hy'(t_n) + O(h^2) \Longrightarrow y_{n+1} = y_n + hf(t_n, y_n) + O(h^2).$$

Since $y'(t) = f(t, y(t))$, we can derive a one step method (involving function evaluations only at t_N) by using

$$y_{n+1} = y_n + hy'(t_n) + \cdots \frac{h^p}{p!} y^{(p)}(t_n) + O(h^{p+1}).$$

We want to point out

$$y'' = \frac{d}{dt}(f(t, y)) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} y' = f_t + f_y f.$$

We can see if we want to compute higher derivatives for y, there will be some

ugly formula to compute. And if you do, then you can derive the **Taylor series method** for the ODE.

3.2 Simple Runge-Kutta Methods

Now we see that $y' = f(t_n, y_n)$ are reasonable to compute, but $y'' = f_t + f_y f$ are not. The Runge-Kutta methods are designed to approximate the integral in (2) by using interpolant of $f(t, y(t))$ at several stages between t_n and t_{n+1}, and the interpolant is constructed by using the slopes f_i at these stages. The simplest Runge-Kutta method is the second order Runge-Kutta method, which is also known as the midpoint method. The method can be illustrated as the following

$$\begin{cases} f_1 = f(t_n, y_n), \quad f_2 = f(t_n + ch, y_n + haf_1), \\ y_{n+1} = y_n + h(b_1 f_1 + b_2 f_2) \end{cases}$$

with constants b_1, b_2, a, c to be determined. We note the method is second order accurate and the term $h(b_1 f_1 + b_2 f_2)$ is the approximation of the integral in $y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt$.

The goal is to choose the constants so that

$$LHS = \frac{1}{h}(y_{n+1} - y_n) = b_1 f_1 + b_2 f_2 + \tau_{n+1} = RHS, \ \tau_{n+1} \sim O(h^2). \quad (8)$$

The strategy is to expand in a Taylor series around t_n and y_n to get the terms of f and its derivatives and **match terms**!

For LHS

$$\frac{1}{h}(y_{n+1} - y_n) = y'_n + \frac{h}{2} y''_n + O(h^2)$$
$$= f + \frac{h}{2}(f_t + f_y f) + O(h^2)$$

For the RHS, first expand f_2 using a Taylor series around (t_n, y_n):

$$f_2 = f_n + (ch)\frac{\partial f(t_n, y_n)}{\partial t} + (haf_1)\frac{\partial f(t_n, y_n)}{\partial y} + O(h)$$
$$= f + ch f_t + haf f_y + O(h^2)$$

Then we can get the RHS:

$$RHS = b_1 f + b_2(f + ch f_t + haf f_y) + \tau_{n+1} + O(h^2)$$

Both LHS and RHS of equation (8) are now available and we can match the terms to get the coefficients b_1, b_2:

$$b_1 + b_2 = 1,$$
$$cb_2 = ab_2 = \frac{1}{2}$$

If we choose $b_2 = \theta$, then $b_1 = 1 - \theta$, $a = c = 1/(2\theta)$, so for any $\theta \neq 0$, we

have

$$\begin{aligned} f_1 =& f(t_n, y_n), \\ f_2 =& f(t_n + h, y_n + h\theta f_1), \\ y_{n+1} =& y_n + h((1-\theta)f_1 + \theta f_2). \end{aligned}$$

The most popular choice is $\theta = 1/2$, and we can get the coefficients $b_1 = \frac{1}{2}$, $b_2 = \frac{1}{2}$, $a = 1$, $c = 1$.

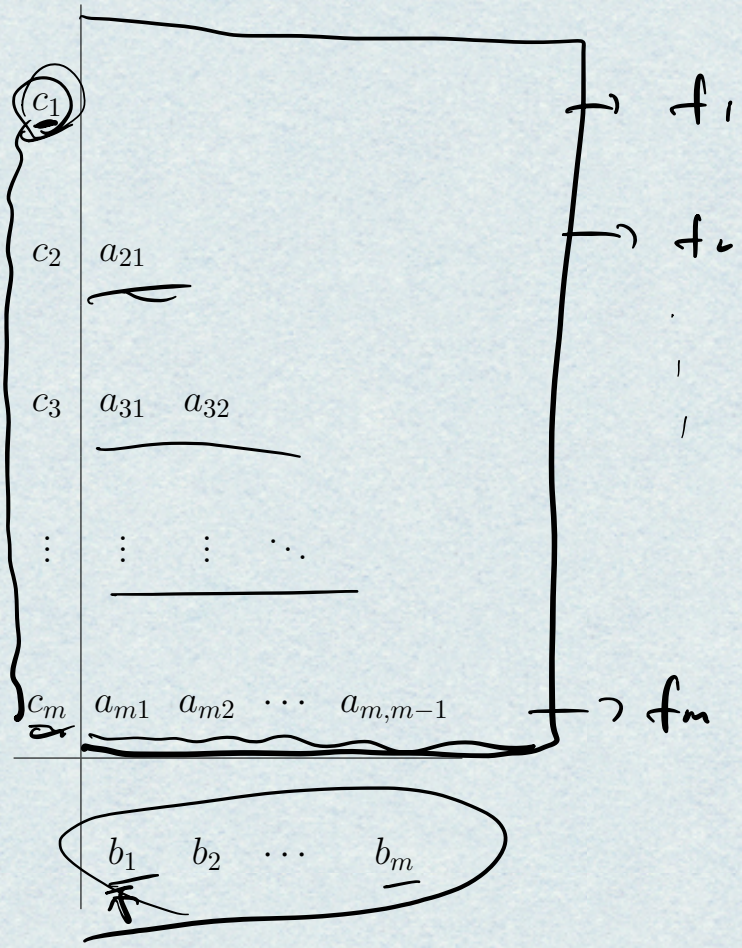
3.3 Typical Runge-Kutta Methods

We note the general form of the Runge-Kutta method in (7), and we can see that the method is determined by the coefficients a_{ij}, b_i, c_i. The most popular Runge-Kutta methods are the second order Runge-Kutta method (midpoint method), the fourth order Runge-Kutta method (RK4), and the fifth order Runge-Kutta method (RK5).

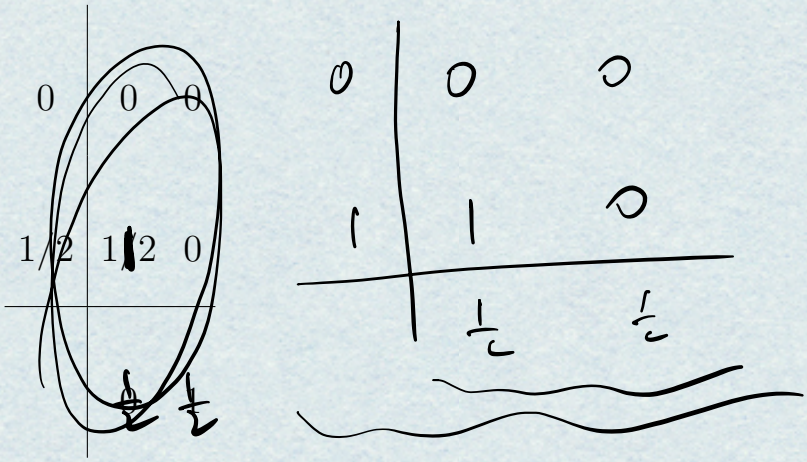
$$\begin{aligned} f_1 =& f(t_n, \phi_n), \\ f_2 =& f(t_n + c_2 h, \phi_n + h a_{21} f_1), \\ f_3 =& f(t_n + c_3 h, \phi_n + h a_{31} f_1 + h a_{32} f_2), \\ \vdots& \\ f_m =& f(t_n + c_m h, \phi_n + h a_{m1} f_1 + h a_{m2} f_2 + \cdots + h a_{m,m-1} f_{m-1}), \\ \phi_{n+1} =& \phi_n + h(b_1 f_1 + b_2 f_2 + \cdots + b_m f_m). \end{aligned}$$

$$\phi_{n+1} = \phi_n + h(b_1 f_1 + b_2 f_2 + \cdots b_m f_m)$$

And this can be summarized in a tableau as



The second order Runge-Kutta method is the midpoint method, and it has the following coefficients:



The second order Runge-Kutta method is

$$f_1 = f(t_n, y_n),$$

$$f_2 = f(t_n + h/2, y_n + hf_1/2),$$

$$y_{n+1} = y_n + hf_2.$$

The second order Runge-Kutta method is 2nd order accurate in terms of global error.

The most popular Runge-Kutta method is the fourth order Runge-Kutta method (RK4), which is explicit one step method and has the following coefficients:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

explicit

$$\sum_{i=1}^{m} b_i = 1$$

$$\phi_{n+1} = \phi_n + h \sum_{i=1}^{m} b_i f_i \longrightarrow \frac{\phi_{n+1} - \phi_n}{h} = \sum_{i=1}^{m} b_i f_i$$

The fourth order Runge-Kutta method is

$$f_1 = f(t_n, y_n),$$

$$f_2 = f(t_n + h/2, y_n + hf_1/2),$$

$$f_3 = f(t_n + h/2, y_n + hf_2/2),$$

$$f_4 = f(t_n + h, y_n + hf_3),$$

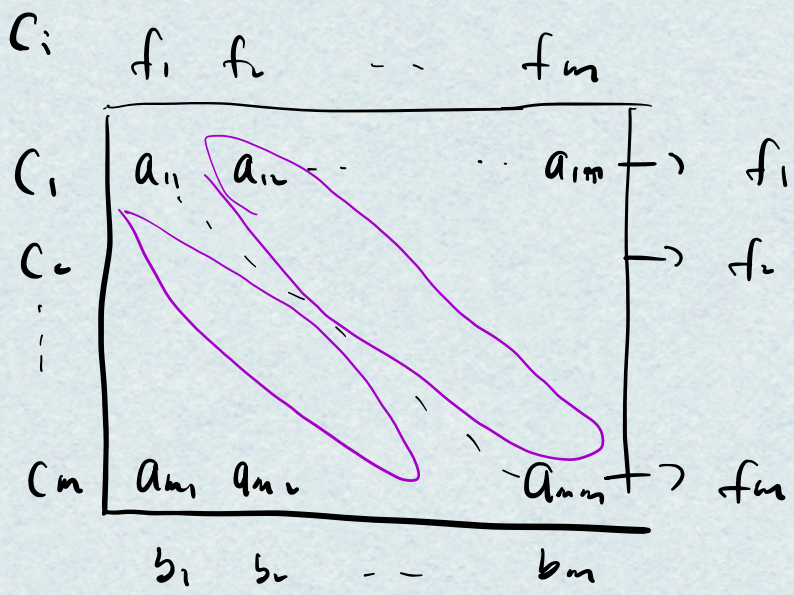
$$y_{n+1} = y_n + h(f_1 + 2f_2 + 2f_3 + f_4)/6.$$

Here the integral $\int_{t_n}^{t_{n+1}} f(t, y(t))dt$ is approximated by the sum $h(f_1 + 2f_2 + 2f_3 + f_4)/6$, which is the approximated Simpson's rule and has local truncation error of order $O(h^5)$. So the RK4 method is 4th order accurate in terms of global error.

3.4 Implicit Runge-Kutta Methods

The implicit Runge-Kutta methods are constructed by using the implicit trapezoidal rule to approximate the integral in (2). The implicit Runge-Kutta methods are more stable than the explicit Runge-Kutta methods, but they are more expensive to compute as they require the solution of nonlinear

equations at each stage. The implicit Runge-Kutta methods are usually used for stiff ODEs.



So implicit RK method require solving non-linear systems of equations f_i

discussion of stability

The eqn. (*) $y' = -20(y - \sin t) + \cos t$, $y(0) = 1$

For this, we need to use $h = \Delta t \le 0.1$

to make the numerical result stable

Stiffness of ODE IVP

Def: An ODE IVP in some interval $[a, b]$ is called stiff if for typical explicit method requires a much smaller time step to be stable than is needed to represent the solution accurately

Q: how to analyze a method to see the stability constraint?

A: stability analysis on the test eqn

$$y'(t) = \lambda y(t), \quad y(0) = y_0$$

where λ is a (complex) number

the solution is just $y(t) = y_0 e^{\lambda t}$ $(\lambda = a + bi)$

note $y(t) = y_0 e^{at}(\cos(bt) + i\sin(bt))$

- $\mathrm{Re}(\lambda) = a < 0$,

  while $t \to \infty$, $e^{at} \to 0$

  so $y(t) \to 0$ w/ $t \to \infty$

$e^{\alpha+\beta i} = e^{\alpha} e^{\beta i}$

$= e^{\alpha}(\cos\beta + i\sin\beta)$

$(\lambda = a + bi)$

- $\mathrm{Re}(\lambda) = a > 0$, then $y(t) \nearrow \infty$

$\mathrm{Re}(\lambda) = a$

$\mathrm{Im}(\lambda) = b$

Note a good numerical method for ϕ_n

the test eqn should have the property,

when $\mathrm{Re}(\lambda) \le 0$, $\phi_n \to 0$ as $n \to \infty$

Now let's look at Euler's method: $\phi_{n+1} = \phi_n + h f(t_n, \phi_n)$

on the test eqn $y' = \lambda y$, $y(0) = y_0$ follows

$\phi_{n+1} = \phi_n + h\lambda\phi_n = (1 + h\lambda)\phi_n$, then it leads to

$\phi_1 = (1 + h\lambda)\phi_0$

$\phi_2 = (1 + h\lambda)\phi_1$

ϕ_{n-1}

$\phi_n = (1 + h\lambda)\phi_{n-1} = (\quad)\phi_0$