ODEs: Initial Value Problems

1 More topics on ODE IVPs methods

1.1 Review of the Runge-Kutta methods

The Runge-Kutta methods are a class of numerical methods for solving ordinary differential equations. The general form of the Runge-Kutta method is

$$\phi_{n+1} = \phi_n + h \sum_{i=1}^{s} b_i f_i,$$

where h is the step size, y_n is the numerical solution at time t_n, y_{n+1} is the numerical solution at time $t_{n+1} = t_n + h$, b_i are the weights, and f_i are the slopes. The slopes f_i are defined as

$$f_1 = f(t_n, \phi_n),\ f_2 = f(t_n+c_2h, \phi_n+h(a_{21}f_1)),\ \ldots,\ f_s = f(t_n+c_sh, \phi_n+h\sum_{j=1}^{s-1} a_{sj}f_j),$$

where $f(t, y)$ is the right-hand side of the ODE, a_{ij} are the coefficients of the method, and c_i are the nodes of the method. We have the constraint $\sum_{i=1}^{s} b_i = 1$ for the weights, and $\sum_{j=1}^{s} a_{ij} = c_i$ for the coefficients. If $a_{ij} = 0, i \leq j$, the RK method will be explicit. And if $a_{ii} \neq 0, a_{ij} = 0, i < j$, we have diagonally implicit RK method (DIRK). We can use the Butcher tableau to describe the coefficients of the Runge-Kutta method:

$$\begin{array}{c|ccccc}
c_1 & a_{11} & a_{12} & \dots & a_{1s-1} & a_{1s} \\
c_2 & a_{21} & a_{22} & \dots & a_{2s-1} & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} & a_{ss} \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s
\end{array}$$

We also note, the RK methods can be written as

$$\frac{\phi_{n+1} - \phi_n}{h} = \sum_{i=1}^{s} b_i f_i = \sum_{i=1}^{s} b_i f(t_n + c_i h, \phi_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_j h, \phi_n)).$$

So this is the approximation of the equation:

$$\frac{y_{n+1} - y_n}{h} = b_1 f_1 + b_2 f_2 + \ldots + b_s f_s + \mathcal{O}(h^p),$$

For the order of a s stage Runge-Kutta method, we have 1) for $s \leq 4$, the best possible order is s, and 2) for $s \geq 5$, the best possible order is $s - 1$. The order of the method is determined by the number of stages s and the coefficients of the method.

1.2 Review of stability

A stiff problem usually means there are multiple scales in the solutions to an ODE IVP.

Region of absolute stability: The region of absolute stability of a method is the set of points $z = h\lambda$ in the complex plane for which the numerical solution of the test equation $y' = \lambda y$ (h is the step size of the numerical method and λ is a complex number) $\phi_n, n = 0, 1, \ldots$ satisfies $\phi_n \to 0$ as $n \to \infty$.

A-Stability: A method is A-stable if its region of absolute stability contains the entire left half-plane, i.e., the region of absolute stability includes $\{z \in \mathbb{C} : \Re(z) \leq 0\}$.

We note that backward Euler method is A-stable, and none of the explicit Runge-Kutta methods are A-stable.

../images/Compare_RAS.png

1.3 Adaptive time step

One-step methods: The one-step methods are methods that use only the information at the current time point t_n to advance the solution to the next time point t_{n+1}. So we **can choose the step size** h to advance the solution to t_{n+1}. If we do not choose the step size every step, we can use a fixed step size h throughout the simulation. This is called the **fixed step** method. The fixed step method is not efficient, as it may require a very small step size to meet the accuracy requirement, especially for those stiff problems.

To be efficient, the algorithm should be able to choose an h just small enough to **meet the desired needs for error**. The simplest approach is to request a tolerance ϵ and attempt local error control by requiring that

$$|e_n| = |\phi_n - y(t_n)| \le \epsilon. \text{ for all } n.$$

We note, both absolute error and relative error can be used to control the error of the solution. The absolute error is $|e_n|$, and the relative error is $|e_n|/|\phi_n|$. The relative error need to be used when the solution is close to zero. An absolute error is used in the following discussion, and you can always use both in your code.

Adaptive step control: With adaptive step size control we compute two solutions of the ODE over a single step using a step length h using an order p method and an order $p+1$ method and calculate the difference between the two solutions. If this difference is less than a desired accuracy we say that the step has succeeded else we say that the step has failed. If the step was successful we increase h and move to the next step. If the step had failed we decrease and repeat the same step.

Two methods of different order of accuracy: We can use two methods of different order of accuracy to *estimate* the error of the solution. Suppose we have two methods of order p and $p+1$, and the solution of the two methods are: Method A approximation $\phi^{(p)}$ with local error $e_n(h) = O(h^p)$; and Method B approximation $\phi^{(p+1)}$, with error $O(h^{p+2})$.

Our goal is to find the new time step size h^* such that the error estimate is less than the tolerance ϵ, i.e., $e_n(h^*) < \epsilon$.

We should note, assuming the value at t_n is exact, we have

$$y(t_n) = \phi^{(p)} + O(h^p) = \phi^{(p+1)} + O(h^{p+2}).$$

Now we can use the difference between the two methods to estimate the error of the solution (subtracting the two equations above): $0 = \phi^{(p+1)} - \phi^{(p)} + O(h^{p+2}) - O(h^{p+1})$, so we introduce the error estimate

$$e_{\text{est}} = |\phi^{(p+1)} - \phi^{(p)}| = \mathcal{O}(h^{p+1}) + O(h^{p+2}) \sim O(h^{p+1}).$$

We need the new step h^* such that $C(h^*)^{p+1} \leq \epsilon$ (we check the equal sign). And we can define the ratio $r = h^*/h$, so we have

$$\frac{C(h^*)^{p+1}}{Ch^{p+1}} = \frac{\epsilon}{e_{\text{est}}} \to (\frac{h^*}{h})^{p+1} = r^{p+1} = \frac{\epsilon}{e_{\text{est}}} \to r = \left(\frac{\epsilon}{e_{\text{est}}}\right)^{1/(p+1)}.$$

If $e_{\text{est}} > \epsilon$, we need to reduce the step size h^* to get the error estimate. Usually we use $0.8r$ to make sure the error estimate is less than the tolerance ϵ.

1.4 Embedded RK methods

As shown in previous discussion, we can use two methods of different order of accuracy to estimate the error of the solution, and adaptively control the step. But this requires two methods to be used for one step, and the **cost** is twice of the fixed step method. German mathematician Erwin Fehlberg proposed a method that uses two Runge-Kutta methods of order 4 and 5 to solve the ODEs, and the method is 4th order accurate in terms of global error. The method is explicit in time, which makes the implementation much easier. The method uses adaptive time stepping to control the error of the solution. The method is called the **embedded Runge-Kutta method**. It is implemented in Matlab as the function `ode45`. The Butcher tableau of the embedded RK method is in the form:

$$\begin{array}{c|ccccc}
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} & 0 \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s \\
 & b_1^* & b_2^* & \dots & b_{s-1}^* & b_s^*
\end{array}$$

0	0					
1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	−9/10	6/5			
1	−11/54	5/2	−70/27	35/27		
7/8	1631/55296	175/512	578/13824	44275/110592	253/4096	
	37/378	0	250/621	125/594	0	512/1771
	2825/27648	0	18575/48384	13525/55296	277/14336	1/4

For example, the Cash-Karp order 4(5) Runge-Kutta method (RKCK45) has the table above.

1.5 Estimating errors

Relative error and absolute error: for a given solution $\phi(t)$, if we know the exact value of the equation at t is $y(t)$, then the absolute error is $|y(t) - \phi(t)|$, and the relative error is $|y(t)-\phi(t)|/|y(t)|$. We note relative error shows the error in terms of the size of the solution.

Example: If we know the exact value of the solution at t_n is $y(t_n) = 2.20345$, and a numerical method produced the value $\phi_n = 2.20$, then the absolute error is $|2.20345 - 2.20| = 0.00345 = 3.45 \times 10^{-3}$. Also, the relative error is $|2.20345 - 2.20|/|2.20345| = 3.45 \times 10^{-3}/2.20345 \approx 1.57 \times 10^{-3}$.

Estimate errors from numerical solutions: If we do not know the exact solution of the ODE, we can two methods

- **Estimates from a fine-grid solution**: if we can afford to run a very fine grid simulation, we can use the solution from the fine grid as the reference in computing the errors on some sequence of much coarser steps. Note, in this case, typically we choose the grids in such a way that all grid points on the coarser grid are also fine-grid points. Then we can use the slope of the **log-log plot** to estimate the order of the method.

- **Estimates from coarser solutions**: If we run simulations on step size h $h/2$ and $h/4$, we can use the difference between the solutions to estimate the error of the solution. Suppose the numerical results are denoted by $\phi_h, \phi_{h/2}, \phi_{h/4}$, then we can use the the following to estimate errors:

$$\tilde{E}(h) = \phi_h - \phi_{h/2}, \quad \tilde{E}(h/2) = \phi_{h/2} - \phi_{h/4},$$

Note

$$\tilde{E}(h) = E(h) - E(h/2) \approx C(1 - \frac{1}{2^p})h^p$$

$$\tilde{E}(h/2) = E(h/2) - E(h/4) \approx C(1 - \frac{1}{2^p})\frac{h^p}{2^p}$$

So we see

$$\tilde{E}(h)/\tilde{E}(h/2) = 2^p \Longrightarrow p = \log_2(\tilde{E}(h)/\tilde{E}(h/2)).$$