

# Numerical Solutions of Ordinary Differential Equations

Jing Li, Kent State University

Most ordinary differential equations cannot be solved analytically. Very often, we need to find numerical solution for a ODE.

## 1 Euler's method

Let us look at a method to solve a general ODE of the form

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)). \quad (1)$$

At any given instant  $t$ , the equation (1) tells us the rate of change of the solution  $\mathbf{y}(t)$  at any instant  $t$ .

Rather than looking for a continuous solution on the interval  $t \in [t_0, T]$ , where  $t_0$  is the initial time,  $T$  is the final time, we try to find an approximation discrete solution of the ODE. We discretize the time interval  $t \in [t_0, T]$  by a sequence of discrete points  $t_0, t_1, t_2, \dots, t_n = T$ . As for the use of notations, we always represent the exact solution at the time  $t_k$  by  $\mathbf{y}(t_k)$ , and the approximate value of the solution at the time  $t_k$  by  $\mathbf{y}_k$ .

Euler's method can be derived as follows. Given initial value  $\mathbf{y}(t_0)$  at the time  $t_0$ , we need to find an approximate value of the solution  $\mathbf{y}(t)$  at time  $t_1$ . From the equation (1) we have

$$\int_{t_0}^{t_1} \mathbf{y}'(t) dt = \int_{t_0}^{t_1} \mathbf{f}(t, \mathbf{y}(t)) dt,$$

i.e.,

$$\mathbf{y}(t_1) - \mathbf{y}(t_0) = \int_{t_0}^{t_1} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

We use rectangle quadrature rule on the integral and we have

$$\mathbf{y}(t_1) - \mathbf{y}(t_0) = (t_1 - t_0)\mathbf{f}(t_0, \mathbf{y}(t_0)) + O((t_1 - t_0)^2),$$

where  $O((t_1 - t_0)^2)$  is the error term. Without considering the error term, we have an approximation of  $\mathbf{y}(t_1)$  by

$$\mathbf{y}_1 = \mathbf{y}_0 + h_0\mathbf{f}(t_0, \mathbf{y}_0),$$

where  $h_0 = t_1 - t_0$  is the step size.

In general, to advance from time  $t_k$  to  $t_{k+1}$ , we have

$$\int_{t_k}^{t_{k+1}} \mathbf{y}'(t) dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt,$$

i.e.,

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

Using rectangle quadrature rule on the right hand side, we have

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = (t_{k+1} - t_k)\mathbf{f}(t_k, \mathbf{y}(t_k)) + O((t_{k+1} - t_k)^2).$$

Denote  $h_k = t_{k+1} - t_k$ , and neglect the error term  $O(h_k^2)$ , and we have the Euler's method

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k), \quad k = 0, 1, 2, \dots$$

## 2 Convergence of Euler's method

We call  $O(h_k^2)$  the local error of the Euler's method at each step. This error comes from the approximation of the integral

$$\int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t))$$

by rectangle rule. In general, given an arbitrary time-stepping method

$$\mathbf{y}_{k+1} = \mathcal{Y}_k(\mathbf{f}, h, \mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k), \quad k = 0, 1, \dots$$

the local error is determined by

$$\mathcal{Y}_k(\mathbf{f}, h, \mathbf{y}(t_0), \mathbf{y}(t_1), \dots, \mathbf{y}(t_k)) - \mathbf{y}(t_{k+1}).$$

If the local error of a numerical method for ODE is the order of  $O(h^{p+1})$ , then we say this method is of order  $p$ , where we use  $h$  to denote the average of step size. We can simply assume that the equal step size is used.

Our main interest is not the local error but the global error,  $\mathbf{y}_k - \mathbf{y}(t_k)$ . Let a numerical method of ODE has a local error is of the order  $O(h^{p+1})$ . Since the number of advance steps from  $t_0$  to  $t_k$  is of the order  $O(h^{-1})$ . The naive expectation is that the global error is of the order  $O(h^p)$ . This is the reason that we say an algorithm is of the order  $p$ , when its local error is of the order  $O(h^{p+1})$ .

The local error of the Euler's method is of the order  $O(h^2)$ . We will show in the following a detailed derivation of the global error  $\mathbf{y}_k - \mathbf{y}(t_k)$ , which is the order of  $O(h)$ .

A method is said to be convergent if, for every ODE (1) with a Lipschitz function  $\mathbf{f}$  and every  $T > 0$  it is true that

$$\lim_{h \rightarrow 0} \max_{k=0,1,\dots,T/h} \|\mathbf{y}_{k,h} - \mathbf{y}(t_k)\| = 0,$$

where  $\mathbf{y}(t_k)$  is the exact solution of the ODE at the time  $t_k$ , and  $\mathbf{y}_{k,h}$  is the approximate solution at the time  $t_k$  with step size  $h$ . Therefore convergence means that for every Lipschitz function  $\mathbf{f}(t, \mathbf{y})$ , the numerical solution tends to the true solution as the grid becomes increasingly finer.

**Theorem** *Euler's method is convergent.*

*Proof:* Given  $h > 0$ , denote the error at time  $t_k$  as  $\mathbf{e}_{k,h} = \mathbf{y}_{k,h} - \mathbf{y}(t_k)$ , where  $t_k = kh$ , with  $k = 0, 1, \dots, T/h$ . It never hurts to say  $T/h$  as an integer.

From Taylor expansion and the ODE (1), we have

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + h\mathbf{y}'(t_k) + O(h^2) = \mathbf{y}(t_k) + h\mathbf{f}(t_k, \mathbf{y}(t_k)) + ch^2.$$

where  $c$  is a constant independent of  $h$ . Subtract this equation with the Euler's iteration

$$\mathbf{y}_{k+1,h} = \mathbf{y}_{k,h} + h\mathbf{f}(t_k, \mathbf{y}_{k,h}).$$

we have

$$\mathbf{e}_{k+1,h} = \mathbf{e}_{k,h} + h(\mathbf{f}(t_k, \mathbf{y}_{k,h}) - \mathbf{f}(t_k, \mathbf{y}(t_k))) + ch^2.$$

Use Lipschitz continuity of the function  $\mathbf{f}$ ,

$$\|\mathbf{f}(t_k, \mathbf{y}_{k,h}) - \mathbf{f}(t_k, \mathbf{y}(t_k))\| \leq L\|\mathbf{y}_{k,h} - \mathbf{y}(t_k)\|,$$

we have

$$\|\mathbf{e}_{k+1,h}\| \leq (1 + hL)\|\mathbf{e}_{k,h}\| + ch^2, \quad k = 0, 1, 2, \dots, T/h.$$

From the following recurrence

$$\begin{aligned} \|\mathbf{e}_{0,h}\| &= 0, \\ \|\mathbf{e}_{1,h}\| &\leq (1 + hL)\|\mathbf{e}_{0,h}\| + ch^2 = ch^2, \\ \|\mathbf{e}_{2,h}\| &\leq (1 + hL)\|\mathbf{e}_{1,h}\| + ch^2 = (1 + (1 + hL))ch^2, \\ \|\mathbf{e}_{3,h}\| &\leq (1 + hL)\|\mathbf{e}_{2,h}\| + ch^2 = (1 + (1 + hL) + (1 + hL)^2)ch^2, \\ &\vdots \end{aligned}$$

we have in general, for  $k = 1, 2, 3, \dots, T/h$ ,

$$\begin{aligned} \|\mathbf{e}_{k,h}\| &\leq (1 + (1 + hL) + (1 + hL)^2 + \dots + (1 + hL)^{k-1})ch^2, \\ &= \frac{(1 + hL)^k - 1}{(1 + hL) - 1}ch^2 = \frac{c}{L}((1 + hL)^k - 1)h \end{aligned}$$

Notice that

$$(1 + hL)^k \leq (1 + hL)^{T/h}, \quad k = 1, 2, 3, \dots, T/h$$

and

$$\lim_{h \rightarrow 0} (1 + hL)^{T/h} = e^{TL},$$

we have

$$\|\mathbf{e}_{k,h}\| \leq \frac{c}{L}(e^{TL} - 1)h, \quad k = 1, 2, 3, \dots, T/h$$

Therefore

$$\lim_{h \rightarrow 0} \max_{k=0,1,\dots,T/h} \|\mathbf{e}_{k,h}\| = 0,$$

and it converges to zero in the order of  $O(h)$ .

### 3 Stability of Euler's method

The convergence of Euler's method tells us that the approximation solution will converge to the exact solution, when  $h \rightarrow 0$ . However, for the implementation, we cannot take  $h \rightarrow 0$ . We need to work on some fixed  $h$ . For a fixed  $h$ , an upper bound of the error is provided by  $(e^{TL} - 1)h$ . Therefore even though you take very small  $h$ , the error could also be very large due to the term  $(e^{TL} - 1)$ , if we have large  $T$  and/or large  $L$ . This comes to the issue of stability.

Lets us recall that we say a solution of a ODE is stable if perturbations of the solution will not diverge away from it over time. Here we say a numerical method is stable, if small perturbations will not resulting numerical solutions of a stable solution of ODE to diverge away without bound.

We study the stability of a numerical method by applying it to solve a scalar ODE

$$y' = \lambda y, \quad y(0) = y_0 \quad (2)$$

We know that when  $\lambda < 0$ , the solutions of this ODE will converge to zero as  $t \rightarrow \infty$  and therefore the solutions are stable. We will find the conditions under which the numerical solutions will also converge to zero as  $t \rightarrow \infty$ . We use that condition to determine whether this numerical method is stable or not.

From the Euler's method

$$y_{k+1} = y_k + hf(t_k, y_k),$$

we have

$$\begin{aligned} y_1 &= y_0 + h\lambda y_0 = (1 + h\lambda)y_0 \\ y_2 &= y_1 + h\lambda y_1 = (1 + h\lambda)y_1 = (1 + h\lambda)^2 y_0 \\ y_3 &= y_2 + h\lambda y_2 = (1 + h\lambda)y_2 = (1 + h\lambda)^3 y_0 \\ &\vdots \end{aligned}$$

and in general

$$y_k = (1 + h\lambda)^k y_0, \quad k = 1, 2, 3, \dots$$

When  $\lambda > 0$ , we know the true solution of the ODE (2),  $e^{\lambda t}y_0$ , goes to infinity as  $t \rightarrow \infty$ , and the same for the approximation sequence  $y_k$ .

We only consider stability for the case when  $\lambda < 0$ . We know the exact solution of the ODE,  $e^{\lambda t}y_0$ , goes to zero as  $t_k \rightarrow \infty$ . But the approximate solution  $y_k$  converges to zero as  $k \rightarrow \infty$ , only when  $|1 + h\lambda| < 1$ , i.e., only when  $-2 < h\lambda < 0$ . Otherwise  $y_k$  will oscillate away from zero as  $k$  increases. In order that  $y_k$  converges to zero as  $k \rightarrow \infty$  when  $\lambda < 0$ , the step size  $h$  must

satisfy

$$h < \frac{2}{|\lambda|},$$

such that Euler's method be stable for solving  $y' = \lambda y$ . The larger is  $|\lambda|$ , the smaller  $h$  is required.

Generalization to the stability for solving

$$\mathbf{y}' = \mathbf{A}\mathbf{y}$$

is straightforward.

The Euler's method gives us

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{y}_0 + h\mathbf{A}\mathbf{y}_0 = (\mathbf{I} + h\mathbf{A})\mathbf{y}_0 \\ \mathbf{y}_2 &= \mathbf{y}_1 + h\mathbf{A}\mathbf{y}_1 = (\mathbf{I} + h\mathbf{A})\mathbf{y}_1 = (\mathbf{I} + h\mathbf{A})^2\mathbf{y}_0 \\ \mathbf{y}_3 &= \mathbf{y}_2 + h\mathbf{A}\mathbf{y}_2 = (\mathbf{I} + h\mathbf{A})\mathbf{y}_2 = (\mathbf{I} + h\mathbf{A})^3\mathbf{y}_0 \\ &\vdots \end{aligned}$$

and in general

$$\mathbf{y}_k = (\mathbf{I} + h\mathbf{A})^k \mathbf{y}_0.$$

Assuming that  $\mathbf{A}$  is a diagonalizable, i.e., it has  $n$  real distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$ , and  $\mathbf{A}$  can be written as

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_n \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_n \end{pmatrix}^{-1},$$

where  $\mathbf{u}_1 \dots \mathbf{u}_n$  are the eigenvectors of  $A$ , then we can write

$$\mathbf{y}_k = (\mathbf{I} + h\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1})^k = \mathbf{U}(\mathbf{I} + h\mathbf{\Lambda})^k \mathbf{U}^{-1} \mathbf{y}_0 = \mathbf{U} \begin{pmatrix} (1 + h\lambda_1)^k & & \\ & \ddots & \\ & & (1 + h\lambda_n)^k \end{pmatrix} \mathbf{U}^{-1} \mathbf{y}_0.$$

We know the exact solution of the ODE,  $\mathbf{y}' = \mathbf{A}\mathbf{y}$ , is

$$\mathbf{y} = \sum_{k=1}^n x_k(0) e^{t\lambda_k} \mathbf{u}_k.$$

where the vector  $\mathbf{x}_0 = (x_1(0), x_2(0), \dots, x_n(0))$  is determined by  $\mathbf{x}_0 = \mathbf{U}^{-1} \mathbf{y}_0$ .

If all  $\lambda_i < 0$ , then we know the solutions of this ODE is stable, and in fact they will converges to zero as  $t$  goes to infinity. In order that the numerical solution  $\mathbf{y}_k$  also converges to zero, we need to have

$$|1 + h\lambda_i| < 1, \quad i = 1, 2, \dots, n.$$

therefore the step size  $h$  has to satisfy

$$h < \min_i \frac{2}{|\lambda_i|},$$

for the stability concern. In another word, the step size  $h$  has to be small enough such that no solution component can diverge with the increase of time.

## 4 Backward Euler method

We already observed that, for solving an ODE with Euler method (forward), there is a stability issue. For example, if we are solving

$$y' = \lambda y, \quad y(0) = y_0,$$

by using Euler method

$$y_{k+1} = y_k + h\lambda y_k, \quad k = 0, 1, 2, \dots$$

we know that

$$h < \frac{2}{|\lambda|},$$

such that  $y_k \rightarrow 0$  as  $k \rightarrow \infty$ , which is consistent with the exact solution  $y(t) = e^{\lambda t} y_0$  when  $\lambda < 0$ .

This constraint on the step size is not desirable, since  $h$  need to be extremely small due to the stability concern when  $\lambda \ll 0$ .

The backward Euler method is derived as follows. Similar to the forward Euler method, to advance from the time  $t_k$  to  $t_{k+1}$ , we have

$$\int_{t_k}^{t_{k+1}} \mathbf{y}'(t) dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt,$$

i.e.,

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

Use (backward) rectangle quadrature rule on the right hand side, we have

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = (t_{k+1} - t_k) \mathbf{f}(t_{k+1}, \mathbf{y}(t_{k+1})) + O((t_{k+1} - t_k)^2).$$

Denote  $h_k = t_{k+1} - t_k$ , and neglect the error term  $O(h_k^2)$ , we have the backward Euler's method as

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}).$$

As for the accuracy of Euler method, we see that the local error is in the order of  $O(h^2)$ , which tells us that the backward Euler method is also a method of accuracy order 1, like the Euler's method.

Let us notice one difference between forward and backward Euler methods. The forward Euler method is an explicit method in the sense that, given the information at the time  $t_k$ , the forward Euler method

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k),$$

simply gives us the explicit formulation of the solution value at the time  $t_{k+1}$ , which is easy to obtain. We call this type of methods as explicit methods. From the backward Euler method,

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}),$$

it is not able to write out the function value  $y_{k+1}$  explicitly from the formulation. In order to obtain  $y_{k+1}$ , we have to solve the above equation. This type of methods are called implicit methods.

We can see that explicit methods are more simpler than implicit methods. Since in each step of implicit method, we have to solve an equation of  $y_{k+1}$ . But the reason that we often consider implicit methods is due to its good stability.

We know that the step size of an explicit algorithm is restricted due to the stability of the method. Now let us look at what may happen for backward Euler method.

Let us consider to solve

$$y' = \lambda y, \quad y(0) = y_0$$

From the backward Euler method

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}),$$

we have

$$\begin{aligned} y_1 = y_0 + h\lambda y_1 &\implies y_1 = (1 - h\lambda)^{-1}y_0, \\ y_2 = y_1 + h\lambda y_2 &\implies y_2 = (1 - h\lambda)^{-1}y_1 = (1 - h\lambda)^{-2}y_0, \\ y_3 = y_2 + h\lambda y_3 &\implies y_3 = (1 - h\lambda)^{-1}y_2 = (1 - h\lambda)^{-3}y_0, \\ &\vdots \end{aligned}$$

and in general

$$y_k = (1 - h\lambda)^{-k}y_0, \quad k = 1, 2, 3, \dots$$

When  $\lambda < 0$ , we know the exact solution of the ODE,  $e^{\lambda t}y_0$ , goes to zero as  $t_k \rightarrow \infty$ . Also from the backward Euler method, we know that  $y_k$  goes to 0, as long as  $\lambda < 0$ , which means no condition need to be put on the step size  $h$ , for the stability concern.

Generalization to solving

$$\mathbf{y}' = \mathbf{A}\mathbf{y}$$

is straightforward.

The backward Euler method gives us

$$\begin{aligned} \mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{A}\mathbf{y}_1 &\implies \mathbf{y}_1 = (\mathbf{I} - h\mathbf{A})^{-1}\mathbf{y}_0 \\ \mathbf{y}_2 = \mathbf{y}_1 + h\mathbf{A}\mathbf{y}_2 &\implies \mathbf{y}_2 = (\mathbf{I} - h\mathbf{A})^{-2}\mathbf{y}_0 \\ \mathbf{y}_3 = \mathbf{y}_2 + h\mathbf{A}\mathbf{y}_3 &\implies \mathbf{y}_3 = (\mathbf{I} - h\mathbf{A})^{-3}\mathbf{y}_0 \\ &\vdots \end{aligned}$$

and in general

$$\mathbf{y}_k = (\mathbf{I} - h\mathbf{A})^{-k}\mathbf{y}_0.$$

Assuming that  $\mathbf{A}$  is a diagonalizable, i.e., it has  $n$  real distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$ , and  $\mathbf{A}$  can be written as

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_n \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_n \end{pmatrix}^{-1},$$

where  $\mathbf{u}_1 \dots \mathbf{u}_n$  are the eigenvectors of  $A$ , then we can write

$$\mathbf{y}_k = (\mathbf{I} - h\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1})^{-k} = \mathbf{U}(\mathbf{I} - h\mathbf{\Lambda})^{-k}\mathbf{U}^{-1}\mathbf{y}_0 = \mathbf{U} \begin{pmatrix} (1 - h\lambda_1)^{-k} & & \\ & \ddots & \\ & & (1 - h\lambda_n)^{-k} \end{pmatrix} \mathbf{U}^{-1}\mathbf{y}_0.$$

We know the exact solution of the ODE,  $\mathbf{y}' = \mathbf{A}\mathbf{y}$ , is

$$\mathbf{y} = \sum_{k=1}^n x_k(0)e^{t\lambda_k} \mathbf{u}_k.$$

where the vector  $\mathbf{x}_0 = (x_1(0), x_2(0), \dots, x_n(0))$  is determined by  $\mathbf{x}_0 = \mathbf{U}^{-1}\mathbf{y}_0$ . If all  $\lambda_i < 0$ , then we know the solutions of this ODE is stable, and in fact they will converges to zero as  $t$  goes to infinity. The numerical solution  $\mathbf{y}_k$ , given by the backward Euler method also converges to zero, no matter what the step size  $h$  is used. In another word, the step size  $h$  has no restriction due to the stability of the backward Euler method.

## 5 Using trapezoid rule

We notice that the accuracy of backward Euler's method is order of 1. By using the trapezoid rule, instead of rectangle rule, we can derive a method of accuracy order 2.

From

$$\int_{t_k}^{t_{k+1}} \mathbf{y}'(t) dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt,$$

we have

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

Use trapezoid rule on the right hand side, we have

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \frac{t_{k+1} - t_k}{2} (\mathbf{f}(t_{k+1}, \mathbf{y}(t_{k+1})) + \mathbf{f}(t_k, \mathbf{y}(t_k))) + O((t_{k+1} - t_k)^3).$$

Denote  $h_k = t_{k+1} - t_k$ , and neglect the error term  $O(h_k^3)$ , we have trapezoid method as

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h_k}{2} (\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})),$$

which is a method of second order accurate.

Trapezoid method for solving ODE is also a implicit method, since there is not a explicit formulation for the value  $\mathbf{y}_{k+1}$  by the given information at the time  $t_k$ .

The stability of trapezoid rule can also be analyzed by solving the ODE

$$y' = \lambda y, \quad y(0) = y_0.$$

The trapezoid method gives us

$$\begin{aligned} y_1 &= y_0 + \frac{h}{2}\lambda(y_0 + y_1) \implies y_1 = \frac{(1 + h\lambda)}{(1 - h\lambda)}y_0, \\ y_2 &= y_1 + \frac{h}{2}\lambda(y_1 + y_2) \implies y_2 = \frac{(1 + h\lambda)}{(1 - h\lambda)}y_1 = \frac{(1 + h\lambda)^2}{(1 - h\lambda)^2}y_0, \\ y_3 &= y_2 + \frac{h}{2}\lambda(y_2 + y_3) \implies y_3 = \frac{(1 + h\lambda)}{(1 - h\lambda)}y_2 = \frac{(1 + h\lambda)^3}{(1 - h\lambda)^3}y_0, \\ &\vdots \end{aligned}$$

and in general

$$y_k = \frac{(1 + h\lambda)^k}{(1 - h\lambda)^k}y_0, \quad k = 1, 2, 3, \dots$$

Therefore, when  $\lambda < 0$ , for any  $h > 0$ ,  $y_k$  converges to 0 as  $k \rightarrow \infty$ , which tells us that no condition on the step size  $h$  need to be put due to the stability reason of the trapezoid method.

## 6 Stiff Ordinary Differential Equations

A stiff problem (a stiff ODE) is one in which the underlying physical process contains components operating on highly disparate time scales, or a process whose time scale is very short compared to the interval of time over which the problem is studied.

Let us generate a simple stiff ODE as follows.

Take the matrices  $\mathbf{U}$ ,  $\mathbf{\Lambda}$ , and  $\mathbf{A}$  as

$$\mathbf{U} = (\mathbf{u}_1 \quad \mathbf{u}_2) = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{\Lambda} = \begin{pmatrix} -1 & 0 \\ 0 & -1000 \end{pmatrix}$$

and

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & -1000 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -500.5 & 499.5 \\ 499.5 & -500.5 \end{pmatrix}.$$

then consider the ODE

$$\mathbf{y}' = \mathbf{A}\mathbf{y}, \quad \mathbf{y}(0) = \mathbf{y}_0.$$

We know  $A$  has two negative eigenvalues  $-1$  and  $-1000$ , with its eigenvectors as  $\mathbf{u}_1$  and  $\mathbf{u}_2$ . We know this ODE is stable, and its solution converge to zero as  $t \rightarrow \infty$ . Its solution is given by

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \mathbf{u}_1 e^{-t} x_1(0) + \mathbf{u}_2 e^{-1000t} x_2(0)$$

where the vector  $\mathbf{x}_0 = [x_1(0) \ x_2(0)]'$  is determined by  $\mathbf{x}_0 = \mathbf{U}^{-1}\mathbf{y}_0$ . For example if we take  $\mathbf{y}_0 = [1 \ 3]'$ , then we know  $x_1(0) = 2$  and  $x_2(0) = 1$ , and therefore the solution is

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} 2e^{-t} + \begin{pmatrix} -1 \\ 1 \end{pmatrix} e^{-1000t} = \begin{pmatrix} 2e^{-t} - e^{-1000t} \\ 2e^{-t} + e^{-1000t} \end{pmatrix}.$$

We can see that the solution has two component:  $e^{-t}$  changes slowly with time, while  $e^{-1000t}$  decaying to zero rapidly. Such a ODE is often called stiff.

## 7 Runge-Kutta Methods

Runge-Kutta methods are high order single step methods.

Consider to solve the ODE

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0.$$

To advance from the time  $t_k$  to  $t_{k+1}$ , we have

$$\int_{t_k}^{t_{k+1}} \mathbf{y}'(t) dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt,$$

i.e.,

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

Use trapezoid quadrature rule on the right hand side, we have

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \frac{t_{k+1} - t_k}{2} (\mathbf{f}(t_k, \mathbf{y}(t_k)) + \mathbf{f}(t_{k+1}, \mathbf{y}(t_{k+1}))) + O((t_{k+1} - t_k)^3).$$

Denote  $h_k = t_{k+1} - t_k$ , and neglect the error term  $O(h_k^3)$ , we have trapezoid method as

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h_k}{2} (\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})),$$

which is an implicit method of second order accuracy and requires a linear or nonlinear solver to obtain  $\mathbf{y}_{k+1}$  in each step.

If we replace the  $\mathbf{y}_{k+1}$  in the right hand side of the trapezoid method, by an approximation of  $\mathbf{y}(t_{k+1})$ , for example by a forward Euler approximation

$$\mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k),$$

then we have the following two-stage, second-order Runge-Kutta method

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h_k}{2} (\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k))),$$

which can be implemented as

*Given  $t_k$ ,  $h_k$  and  $\mathbf{y}_k$ , to compute  $\mathbf{y}_{k+1}$  as follows*

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\ \mathbf{k}_2 &= \mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k \mathbf{k}_1), \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + \frac{h_k}{2} (\mathbf{k}_1 + \mathbf{k}_2). \end{aligned}$$

Runge-Kutta method is an explicit method: we start from  $t_k$ ,  $h_k$  and  $\mathbf{y}_k$ ; first compute  $\mathbf{k}_1$  by evaluating the function value  $\mathbf{f}(t_k, \mathbf{y}_k)$ ; then evaluate  $\mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k \mathbf{k}_1)$  to obtain  $\mathbf{k}_2$ ; at last obtain  $\mathbf{y}_{k+1}$  from  $\mathbf{y}_k$ ,  $\mathbf{k}_1$  and  $\mathbf{k}_2$ . It can also be checked, by using Taylor expansions, that this is a second-order algorithm, i.e.,

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \frac{h_k}{2} (\mathbf{f}(t_k, \mathbf{y}(t_k)) + \mathbf{f}(t_{k+1}, \mathbf{y}(t_k) + h_k \mathbf{f}(t_k, \mathbf{y}(t_k)))) + O(h_k^3).$$

The most popular Runge-Kutta method is the following four-stage fourth-order method

*Given  $t_k$ ,  $h_k$  and  $\mathbf{y}_k$ , to compute  $\mathbf{y}_{k+1}$  as follows:*

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\ \mathbf{k}_2 &= \mathbf{f}(t_k + h_k/2, \mathbf{y}_k + h_k \mathbf{k}_1/2), \\ \mathbf{k}_3 &= \mathbf{f}(t_k + h_k/2, \mathbf{y}_k + h_k \mathbf{k}_2/2), \\ \mathbf{k}_4 &= \mathbf{f}(t_k + h_k, \mathbf{y}_k + h_k \mathbf{k}_3), \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + \frac{h_k}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \end{aligned}$$

which can be thought as deriving from

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

by using Simpson's rule on the right hand side.

These Runge-Kutta methods are high-order explicit method, and therefore is also subject to stability constraints on the step size  $h$ , like for forward Euler method.

## 8 Error control in numerical solution of ODEs

In the practice of solving a ODE numerically, it has to be provided about how to choose a step size  $h_k$  to move forward from  $t_k$  to  $t_{k+1}$ . We know for implicit methods, ( for example, backward Euler, Trapezoid method), the time-stepping is always stable, which means the step size  $h_k$  will be determined only by the required accuracy, not by the stability of the algorithm. For most explicit methods, like forward Euler method, explicit Runge-Kutta methods, the step size has to satisfy some stability conditions, besides accuracy concern. We know the smaller the step size, the smaller the local error and therefore the smaller the global error of a ODE solver. For a given error tolerance, for example require that the relative local error be bounded by  $10^{-4}$  in each time-stepping, we need to find a step size as larger as possible to minimize the number of time steps and therefore minimize the computing cost, while still maintaining the required accuracy.

In this section, we provide a procedure to determine the time step size  $h_k$  which satisfies a local error bound when moving forward from  $t_k$  to  $t_{k+1}$ .

What we need to do is: given an approximation  $\mathbf{y}_k$  at the time  $t_k$ , determine a time step size  $h_k$ , such that the local (relative) error is bounded by a given tolerance  $\epsilon$ . We only consider the case of Euler method.

*Step 1.* Given  $t_k$  and  $\mathbf{y}_k$ , the forward Euler method is

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k).$$

From the Taylor expansion of  $\mathbf{y}(t_{k+1})$

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + h_k \mathbf{y}'(t_k) + h_k^2 \mathbf{y}''(\xi_k)/2 = \mathbf{y}(t_k) + h_k \mathbf{f}(t_k, \mathbf{y}(t_k)) + h_k^2 \mathbf{y}''(\xi_k)/2$$

where  $t_k \leq \xi_k \leq t_{k+1}$ , we know that the local error of the forward Euler method is  $h_k^2 \mathbf{y}''(\xi_k)/2$ . An estimate of  $\mathbf{y}''(\xi_k)$  is given by the finite difference formula by using the information from the previous step

$$\mathbf{y}'' \approx \frac{\mathbf{y}'_k - \mathbf{y}'_{k-1}}{t_k - t_{k-1}} = \frac{\mathbf{f}(t_k, \mathbf{y}_k) - \mathbf{f}(t_{k-1}, \mathbf{y}_{k-1})}{t_k - t_{k-1}} \equiv \mathbf{y}''_{k-1,k}.$$

Therefore the relative local error bound

$$\left| \frac{h_k^2 \mathbf{y}''(\xi_k)/2}{\mathbf{y}_k} \right| \approx \left| \frac{h_k^2 \mathbf{y}''_{k-1,k}/2}{\mathbf{y}_k} \right| < \epsilon$$

implies

$$h_k < \sqrt{\frac{2\epsilon \mathbf{y}_k}{\mathbf{y}''_{k-1,k}}}.$$

We used to choose  $h_k$  a little less than this bound and also located on the interval  $[h_{min}, h_{max}]$ , where  $h_{min}$  and  $h_{max}$  are the given lower and upper bounds of the step size. In one word, we choose  $h_k$  as

$$h_k = \text{the middle number of } \{h_{min}, \sqrt{\frac{2\epsilon \mathbf{y}_k}{\mathbf{y}''_{k-1,k}}}, h_{max}\}.$$

As for the first step size  $h_0$ , to move from  $t_0$  to  $t_1$ , since the information  $\mathbf{y}''_{k-1,k}$  is not available, we may just take  $h_{max}$  as the initial step size.

*Step 2.* Take the step size  $h_k$  to move from  $t_k$  to  $t_{k+1}$ , i.e., compute  $\mathbf{y}_{k+1}$ . For example, for forward Euler method,  $\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k)$ ; for backward Euler method,  $\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})$ ; or you may also use Runge-Kutta method here.

*Step 3.* Use the updated information  $\mathbf{y}_{k+1}$  at the time  $t_{k+1}$  to confirm this step is not too large. Take

$$\mathbf{y}''_{k,k+1} \equiv \frac{\mathbf{y}'_{k+1} - \mathbf{y}'_k}{t_{k+1} - t_k} = \frac{\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{f}(t_k, \mathbf{y}_k)}{t_{k+1} - t_k}$$

and therefore the local error estimate as

$$\left| \frac{h_k^2 \mathbf{y}''_{k,k+1}/2}{\mathbf{y}_k} \right|$$

to see if it is also bounded by  $\epsilon$ . If it is, then move to the next step, i.e., start *Step 1.* for given  $t_{k+1}$  and  $\mathbf{y}_{k+1}$ . Otherwise, if this error estimate is not bounded by  $\epsilon$ , then it means the  $h_k$  is too large, and therefore we take  $h_k = h_k/2$ , go back to *Step 2.*, and redo the step from  $t_k$  to  $t_{k+1}$ .

This is the most basic idea of the step size control, it is much more sophisticated in practical software. For example, if the step size at one step is too small, it should be doubled, etc. ...