ECE 4960: Computational and Software Engineering

Spring 2017

Note 6: Ordinary Differential Equation

Reading Assignments:

1. Chapters 22 and 23, Charpa.

1. Class logistics

- Programming assisgnment 4
- Reading reviews 5 and 6 (will be multiple choice questions on Blackboard)

2. Comparison of Local Analysis and Ordinary Differential Equation (ODE)

Ordinary differential equations (ODE) render the fundamental descriptions for many behavioral (such as financial analysis, item testing, etc.) and topological (such as circuits and pseudo-rigid-body motion) models in science and engineering. The general form in the multi-variable case can be written as:

$$\frac{d\vec{x}}{dt} = \vec{f}\left(\vec{x}, t\right) \tag{1}$$

Notice here we have $\vec{x}\,(t)$ as the **dependent** variable as a vector of rank *n* and $\vec{f}$ as the functions that describes the relation of $\vec{x}$ and *t* to the time derivative of $\vec{x}$. $\vec{f}$ is also often a vector of rank *n*, but we do not depend on the same rank to give a well-posed problem. The time *t* is the only "**independent**" variable here, which makes the differential equation "ordinary". If the spatial coordinates (*x, y, z*) are also independent variables, i.e., we can have partial derivatives in space and time, the differential equation will be called "partial", which has very different nature from ODE. The most obvious difference between the partial differential equation (PDE) and ODE is that only the first order ODE is important, as for any higher order derivatives, we can add more variables to transfer the higher order derivatives to the first order, such as:

$$x_2 = \frac{dx_1}{dt}; \qquad \frac{d^2 x_1}{dt^2} = \frac{dx_2}{dt} \tag{2}$$

This operation is not possible for PDE, where we have to study the different orders in the PDE system (such as parabolic, elliptical, KdV, etc.)

An example of ODE can be an RC low pass circuits in Fig. 1 that you are familiar with:

$$C\frac{dV}{dt} = I_{in}(t) - \frac{V}{R}; \qquad \frac{dV}{dt} = \frac{1}{C}I_{in}(t) - \frac{V}{RC} \tag{3}$$

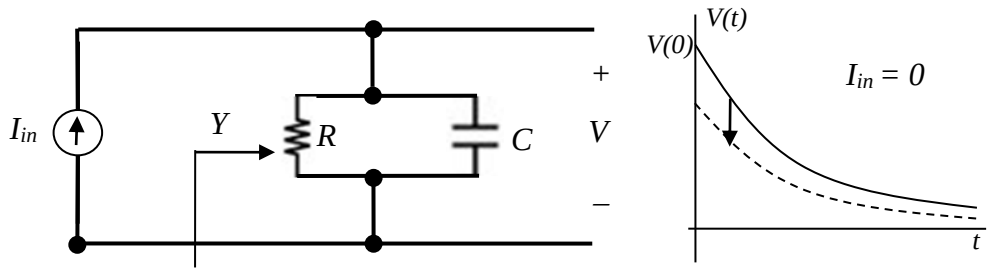


Fig. 1. Simple RC circuits to illustrate ODE and SPICE solution.

Notice how the derivative of *V(t)* depends on a function of *f(V, t)*. The solution of interest will be *V(t)*, as a function of *V(t = 0)* and $I_{in}(t)$. This is in general called the "**initial-value problems**". There are two parts in *f(V, t)*, one depends on the dependent variable *V* explicitly, which is called the **homogeneous** part as in any differential equation, and the other on the independent variable *t* through *i(t)*, which is called the inhomogeneous part.

Another possible example is to describe the free fall of an object. Notice that we treat this object as a pseudo-rigid body, so the description is refered to its center of mass by:

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2 \tag{4}$$

where *g* is the gravitational acceleration, *m* is the mass and c_d the drag coefficient of air. The solution will be *v(t)*, given an initial condition of *v(t = 0)*. The last example is the harmonic oscillator with damping from a friction coefficient of *c*:

$$m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = 0 \tag{5}$$

After transforming *dx/dt* to *v*, we have a pair of ODE:

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -\frac{c}{m}v - \frac{k}{m}x \end{aligned} \tag{6}$$

We can see Eqs. (3), (4), and (6) all fit the general ODE form in Eq. (1)[1]. We need to know $\vec{x}(t=0)$ to obtain the transient solution of ODE of $\vec{x}(t)$ for each dependent variable. For Eq. (6), it is equivalent to say that we need to know *x(t=0)* and *v(t=0)*, or *x(t=0)* and *x'(t=0)*. Higher-order time derivatives and the associated boundary conditions can be viewed accordingly.

We notice that the discretization of the first-order derivatives will be similar to what we have learned in the local analysis. For the first-order precision, we can opt to use two points in *t*:

$$x'(t) = \frac{x(t+h) - x(x)}{h} + O(h) \tag{7}$$

[1] Also notice now the ODE system is entirely homogeneous, i.e., the right hand side has only dependent variables.

$$x'(t)=\frac{x(t+h)-x(t-h)}{2h}+O(h^2) \tag{8}$$

This is similar to the Euler method introduced below. Recall that for the knowledge of three points in *t*, we can approximate the first derivative to the second order precision according to the Taylor expansion by two (or multiple) time steps, as in Eqs. (3 – 6) in Note 3, which is repeated below as Eqs. (9 – 11):

$$x(t+h_1)=x(t)+h_1\cdot x'(t)+\frac{1}{2}h_1^2x''(t)+O(h^3) \tag{9}$$

$$x(t+h_2)=x(t)+h_2\cdot x'(t)+\frac{1}{2}h_2^2x''(t)+O(h^3) \tag{10}$$

$$x'(t)=\frac{h_1}{h_2(h_1-h_2)}x(t+h_2)-\frac{h_1+h_2}{h_1h_2}x(t)-\frac{h_2}{h_1(h_1-h_2)}x(t+h_1)+O(h^2) \tag{11}$$

This is similar to the **Runge-Kutta** method in the second order, which will be introduced later.

For time derivative in local approximation, the truncation (at what orders of *h* the derivative is accurate) and roundoff (floating-point precision limitation) errors are very similar to general local analysis. However, derivatives in time have several unique properties from spatial or parametrical derivatives:

1. Errors in previous time steps are **accumulated** unless the history is re-visited: In Fig. 1, we can see that if the present solution has an error to shift to the trajectory of another solution of different *V(t=0)*, the later time steps will follow the new trajectory, unless the previous history points are revisited. This feature can be observed in the standard solution of Fig. 1 for $I_{in}(t) = 0$ with *V(t)* goes to an exponential decay from *V(t=0)*. The analytical solution is:

$$V(t)=C\cdot\exp\left(-\frac{t}{RC}\right)=V(0)\cdot\exp\left(-\frac{t}{RC}\right) \tag{12}$$

   We can observe that the integration constant *C* (appears in every integration of the first derivative) will match to *V(0)*, but will cause accumulation of errors in time progression.

2. Any physical laws that contain the correct thermodynamics should have **time irreversibility**, and therefore, forward and backward discretization will make a difference in its trajectory.

3. The One-Step Methods

If we only use one point in history, this is called the one-step method, in contrast to the multi-step method in the next chapter. From Eq. (1), the one-step method can be in general expressed in a time progression stamp of *i* as:

$$\vec{x}_{i+1}=\vec{x}_i+\vec{\phi}h \tag{13}$$

where h is the time step (or Δt in some text) and ϕ is called the incremental function. The time progression in i will approximate the solution $\vec{x}(t)$ with given initial condition. We have different ways to evaluate ϕ in the discrete time steps according to Eq. (1).

3.1 Euler's method

The Euler method evaluates ϕ by the first-order approximation of the time derivative. Observe the Taylor expansion of Eq. (13):

$$\vec{x}_{i+1} = x(t+h) \cong \vec{x}_i + \frac{dx}{dt}h + \frac{1}{2!}\frac{d^2x}{dt^2}h^2 + R(h^3) \tag{13}$$

There are three common choices to evaluate *dx/dt*:

1. **Forward Euler**: $\phi = \left.\frac{dx}{dt}\right|_{t=t_i} = f(x_i, t_i)$

2. **Backward Euler**: $\phi = \left.\frac{dx}{dt}\right|_{t=t_{i+1}} = f(x_{i+1}, t_{i+1})$

3. **Trapezoidal Euler**: $\phi = \frac{f(x_i, t_i) + f(x_{i+1}, t_{i+1})}{2}$

For sure other different weighting of $f(x_i, t_i)$ and $f(x_{i+1}, t_{i+1})$ to evaluate *dx/dt* is possible, which we will introduce in Sec. 3.2. Forward and backward Euler methods are $O(h)$ accurate, while the trapedzoidal Euler is $O(h^2)$ accurate, due to the cancellation of the d^2x/dt^2 term as well, similar to the central difference in the local analysis. We will use a most popular example to illustrate the difference in the three Euler methods, which is also used for **stability** testing. Then we will generalize the approximation to the Huen's and Runge-Kutta methods.

Example 1:

For the ODE $\frac{dx}{dt} = -ax$ with $a > 0$, we know the analytical solution is: $x = x_0 e^{-at}$, which decays in time.
We will test this system in the three Euler's methods:

1. Forward Euler: $x_{i+1} = x_i - ahx_i = (1-ah)x_i$. The solution is a geometric series: $x_n = (1-ah)^n x_0$. We can see that x will decay in time only if $|1 - ah| < 1$. We call this "**conditionally stable**", as the system stablility depends on the choice of h. Remember that this is only valid for the simple ODE here. In multi-variable and nonlinear cases, system stability can be more complicated.

2. Backward Euler: $x_{i+1} = x_i - ahx_{i+1}$; $x_{i+1} = \frac{1}{1+ah} x_i$. The solution is a geometric series: $x_n = \left(\frac{1}{1+ah}\right)^n x_0$. We can see that *x* will decay in time regardless of the choice of *h* (remember *a > 0* and *h> 0*). We call this "**unconditionally stable**" or A-stable (Absolute stable)[2], as the system stablility does not depend on the choice of *h*.

3. Trapedzoidal Euler: $x_{i+1} = x_i - ah\left(\frac{x_i + x_{i+1}}{2}\right)$; $x_{i+1} = \frac{2-ah}{2+ah} x_i$. The solution is a geometric series: $x_n = \left(\frac{2-ah}{2+ah}\right)^n x_0$. This is also unconditionally stable for the simple stability testing ODE here.

□

Exercise:

For the period of time of *l* where *N* time steps will be applied, i.e., *h = l/N*, prove that the forward, back and trapezoidal Euler methods all converge to the solution of $x = x_0 e^{-al}$ when $N \to \infty$.

$$y = \lim_{N\to\infty} (1-ah)^N; \qquad \ln y = \lim_{N\to\infty} N \ln(1-ah) = \lim_{N\to\infty} \frac{\ln\left(1-\frac{al}{N}\right)}{\frac{1}{N}} = \lim_{N\to\infty} \frac{\left(1-\frac{al}{N}\right)\cdot\frac{al}{N^2}}{-\left(\frac{1}{N}\right)^2} = -al .$$

Therefore, $\lim_{N\to\infty} x_0(1-ah)^N = x_0 e^{-al}$. The above step has used the L'Hospital rule in the infinite series.

□

Example 1 is often referred as the stability test of any ODE solution method. More texts use the sample problem of $\frac{dx}{dt} = ax$, where the convergent series will require Re(a) < 0 (the left plane has no pole in the control theory). System stability can be a generalized form for this test problem if the system is described by multi-variable ODE.

3.2 The predictor-corrector method (Huen's method)

After we observe the second-order trapezoidal Euler method, we hope to do the best we can in the estimation of the incremental function ϕ within the given time steps. Notice the information at t_i, i.e., $f(x_i, t_i)$ is already available from the previous time step (in most applications, the time progression, evaluation of $f(x,t)$ is the most expensive step in the multi-variable case), so we will definitely use it as

[2] A-stability is defined for the given test problem where *a* can be a complex number with a positive real part. If a method is A-stable, it will converge to 0 when $t \to \infty$.

part of the incremental function. Instead of just using the discretized point t_{i+1} for f(x_{i+1}, t_{i+1}) to evaluate *dx/dt*, can we use the estimated slope to obtain an intermediate point, and iteratively use that intermediate point to make an even better estimation of *dx/dt*? This is the idea behind the predictor (use the current information at hand to predict an intermediate point) and corrector (use the new estimated *dx/dt* to make correction to x_i). This is called the Huen's[3] method.

The Huen's predictor-corrector method without iteration can be expressed as:

$$\text{Predictor: } x_{i+1}^{j=0} = x_i + f(t_i, x_i)h$$

$$\text{Corrector: } x_{i+1}^{j=1} = x_i + \phi h = x_i + \frac{f(x_i, t_i) + f\left(x_{i+1}^{j=0}, t_{i+1}\right)}{2} h \tag{14}$$

This is illustrated in Fig. 2. We have purposedly used a new symbol $x_{i+1}^{j=0}$. If $x_{i+1}^{j=0} = x_{i+1}$ directly, we can see this is just the trapezoidal Euler formation.

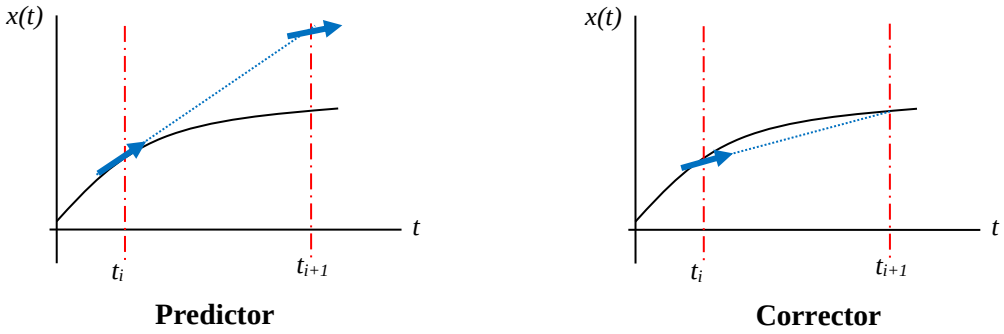


Fig. 2. The predictor-corrector method without iteration.

However, the predictor-corrector method can be iterative for further improvement, by generalizing Eq. (14) to an iteration with respect to *j*, and the general form of the predictor-corrector method can be expressed as:

$$x_{i+1}^{j} = x_i + \frac{f(x_i, t_i) + f\left(x_{i+1}^{j-1}, t_{i+1}\right)}{2} h \tag{15}$$

The iteration will stop at the criteria:

$$\left|\varepsilon_x\right| = \left|\frac{x_{i+1}^{j} - x_{i+1}^{j-1}}{x_{i+1}^{j}}\right| < tol \tag{16}$$

Example 2:

[3] The predictor-corrector method is developed by Karl Huen around 1900's, when the ODE solution for cannon projectile is a matter of life and death. Karl Huen, Carl Runge and Martin Kutta and are all mathematicians in Germany at the time. I guess you can see why their artillery and tank units were the best for quite a long time.

For the ODE of $\frac{dx}{dt} = 4e^{0.8t} - 0.5x$, the exact solution for *x(0)* = 2 is: $x(t) = \frac{4}{1.3}\left(e^{0.8t} - e^{-0.5t}\right) + 2e^{-0.5t}$.
The computation by the forward Euler method, the Huen's method without iteration (trapezoidal) and the Huen's method with iteration (using $tol = 10^{-7}$) is summarized in Table 1.

Table 1. Example of the one-step methods for ODE solution:

t	x_{true}	$x_{forwardEuler}$	$\lvert\varepsilon_x\rvert$ (%)	x_{Huen} (no iter.)	$\lvert\varepsilon_x\rvert$ (%)	x_{Huen} (iter.)	$\lvert\varepsilon_x\rvert$ (%)
0	2.0000	2.0000		2.0000		2.0000	
1	6.1946	5.0000	19.28	6.7011	8.18	6.3609	2.68
2	14.843	11.402	23.19	16.320	9.94	15.302	3.09
3	33.677	25.513	24.24	37.199	10.46	34.743	3.17
4	75.339	56.849	24.54	83.34	10.62	77.735	3.18

We can make some observation on Table 1. First, $|\varepsilon_x|$ usually deteriorates when time progresses due to error accumulation, but this may be affected by the size of time steps. Second, the Huen's method without iteration (trapezoidal Euler) improves the solution, but not by much. The Huen's method with iteration makes significant accuracy improvement by the most information available at t_i and t_{i+1}. However, there are also a few nonideal features in the predictor-corrector method with iteration:

1. In each *j* iteration, $f\left(x_{i+1}^j, t_{i+1}\right)$ has to be re-evaluated due to the new values of x_{i+1}^j. In many real cases with multiple variables, this is computationally expensive.

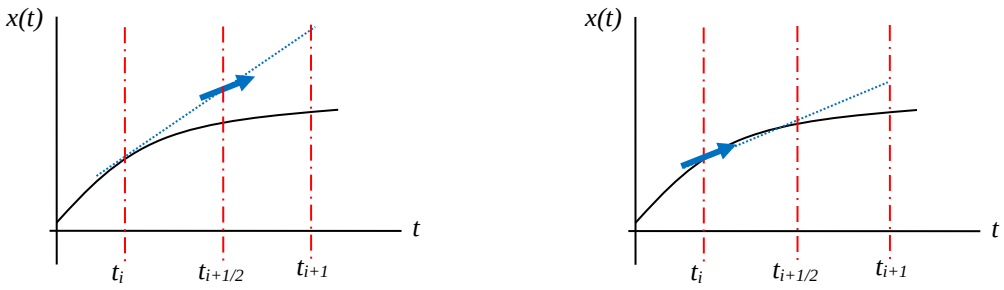
2. We do not know the number of iterations before convergence, and we have no obvious bound or strict proof of the order of accuracy. This is not ideal for debugging and validation. If we know the precise accuracy, by testing the time steps with *4h*, *2h* and *h*, we can know whether our software implementation is correct from external observation, similar to the Richardson extrapolation in the local analysis.

3. As we are calculating x_{i+1}^j anyway, there is NO reason to insist that *t* has to be t_{i+1} every time. From Eq. (11), we know if we relax this constraint to evaluate *f* at different time points between t_i and t_{i+1}, we can improve the order of accuracy in the deterministic manner.

Just from the limitation 3, we can already devise a new method called the midpoint method by estimate the incremental function ϕ at $t_{i+1/2}$ as illustrated in Fig. 3, i.e.,

$$x_{i+1} = x_i + f\left(x_{i+1/2}, t_{i+1/2}\right)h \tag{17}$$

This is very similar to the midpoint evaluation in the integration approximation in the local analysis, and also has a second-order accuracy. We can see that relaxation of the time point for evaluation can improve the accuracy. The midpoint analysis however requires evaluating *f(t, x)* at a new time instance in comparison with Huen's method without iteration or trapezoidal Euler. All these one-step methods do not generate a new spatial solution *x(t)* at $t_{i+1/2}$.



Evaluation of ϕ at the midpoint **Corrector**

Fig. 3. The midpoint one-step method.

The general method to resolve all three limitations of the predictor-corrector method but keeping its main advantages is included in the adaptive-order Runge-Kutta method, as presented in the next section.

3.3 The Runge-Kutta method within a time interval of fixed *h*

Following the spirit of Eq. (11), we can write down the estimation of the incremental function ϕ as a linear combination as evaluation of $f(x, t)$. This is the Runge-Kutta method of arbitrary order. We do not know now what the parameters should be that can make the best approximation, but we will later match them with the Taylor expansion, just like the derivation of Eq. (11) in the local analysis.

In the Runge Kutta method of order n, the incremental function is expressed as:

$$\phi = a_1 k_1 + a_2 k_2 + ... + a_n k_n \tag{18}$$

where k_j with $j = 1...n$ is the estimation of $f(t, x)$ between the interval of (t_i, t_{i+1}), and iteratively defined (i.e., k_j will depend on k_1, … k_{j-1}, in the iterative spirit of the Huen's method). The parameter a_j is unknown now, but will later be determined by matching with the Taylor series. Notice that we exchange the order of x and t in the expression of f for convenience and clarity, as the choice of x will depend on previous k_j and t. The slope function k_j for $j = 1...n$ between t_i and t_{i+1} is defined as:

$$\begin{aligned} k_1 &= f(t_i, x_i) \\ k_2 &= f(t_i + p_1 h, x_i + q_{11} k_1 h) \\ k_3 &= f(t_i + p_2 h, x_i + q_{21} k_1 h + q_{22} k_2 h) \\ &... \\ k_n &= f(t_i + p_{n-1} h, x_i + q_{n-1,1} k_1 h + ... + q_{n-1,n-1} k_{n-1} h) \end{aligned} \tag{19}$$

where p and q are all constants between [0, 1] to be chosen to match with the Taylor series expansion in the n-th order. The slope function k_j will be the evaluation of f at some t in $[t_i, t_{i+1}]$ and x in $[x_i, x_{i+1}]$, and is iteratively defined but does not need internal iternations as in the predictor-corrector method.

Before we go on, we will look at two of the simplest cases of $n = 1$ and $n = 2$ in the Runge-Kutta series. For $n = 1$, only k_1 exists, and no p or q is needed, we retrieve the forward Euler method by choosing $a_1 = 1$! For $n = 2$, we have:

$$
\begin{aligned}
x_{i+1} &= x_i + a_1 k_1 h + a_2 k_2 h \\
k_1 &= f\left(t_i, x_i\right) \\
k_2 &= f\left(t_i + p_1 h, x_i + q_{11} k_1 h\right)
\end{aligned}
\tag{20}
$$

We have FOUR parameters of a_1, a_2, p_1 and q_{11} to determine from the Talyor expansion series, which can be expressed as:

$$
\begin{aligned}
x_{i+1} &= x_i + \frac{dx}{dt} h + \frac{1}{2} \frac{d^2 x}{dt^2} h^2 + O\left(h^3\right) = x_i + fh + \frac{1}{2} \frac{df}{dt} h^2 + O\left(h^3\right) \\
&= x_i + fh + \frac{1}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial t} \right) h^2 + O\left(h^3\right)
\end{aligned}
\tag{21}
$$

We know the Taylor expansion of k_2 will be:

$$
k_2 = f\left(t_i + p_1 h, x_i + q_{11} k_1 h\right) = f\left(t_i, x_i\right) + p_1 h \frac{\partial f}{\partial t} + q_{11} k_1 h \frac{\partial f}{\partial x} \frac{\partial x}{\partial t}
\tag{22}
$$

Therefore, Eq. (20) can be written as:

$$
x_{i+1} = x_i + a_1 f\left(t_i, x_i\right) h + a_2 \left[f\left(t_i, x_i\right) + p_1 h \frac{\partial f}{\partial t} + q_{11} k_1 h \frac{\partial f}{\partial x} \frac{\partial x}{\partial t} \right] h
\tag{23}
$$

By matching Eqs. (21) and (23), we obtain the following condition for the 2nd-order Runge-Kutta expansion as:

$$
\begin{aligned}
a_1 + a_2 &= 1 \\
a_2 p_1 &= \frac{1}{2} \\
a_2 q_{11} &= \frac{1}{2}
\end{aligned}
\tag{24}
$$

There are three conditions for the four parameters of a_1, a_2, p_1 and q_{11} , which means we will have an additional degree of freedom to choose the parameters and still maintain 2nd-order accuracy for all choices as long as Eq. (24) is satisfied. The choice is either for proof of concepts, or more often, for computational efficiency in error estimation or in the future multi-step methods (which will come in the next section).

Let's look at several interesting choices. If we choose $a_2 = 0.5$, we have $a_1 = 0.5$, $p_1 = q_{11} = 1$, and Eq. (20) becomes:

$$
x_{i+1} = x_i + \frac{1}{2} f\left(t_i, x_i\right) h + \frac{1}{2} f\left(t_i + h, x_i + k_1 h\right) h
\tag{25}
$$

which is exactly the Huen's method without iteration. If we choose $a_2 = 1$, we have $a_1 = 0$, $p_1 = q_{11} = 0.5$, and Eq. (20) now becomes:

$$x_{i+1} = x_i + f\left(t_i + \frac{h}{2}, x_i + \frac{k_1 h}{2}\right) \tag{26}$$

which is exactly the midpoint method in Eq. (17)! We know that both cases are 2nd-order accurate. Actually you can choose $a_2 = 2/3$ or $\sqrt{3}/2$ for different purposes and still remain 2nd-order accuracy.

Now you know the spirit of the Runge-Kutta expansion, we will omit the lengthy algebraic derivation here. The Runge Kutta method can be of any order, and it can be seen as an universal expression for the one-step ODE time discretization scheme from any given closed form. The Runge Kutta method has not only inherited the iterative refinement nature in the predictor-corrector method, but also offers a known order of accuracy in its various forms.

The Runge Kutta method can give an internal **error estimation** between two different orders, which can be used to evaluate whether the present time step size is appropriate. This is a very important feature in *hp* adaptivity, which we will revisit very soon. Notice that there is a free parameter to choose, which greatly expands its flexibility and computational efficiency by aligning internal points of $[t_i, t_{i+1}]$ in different orders to minimize the number of evaluation of $f(t, x)$ in each time step.

A popular choice in the fourth-order Runge-Kutta has the following formula:

$$\begin{aligned}
x_{i+1} &= x_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)h \\
k_1 &= f\left(t_i, x_i\right) \\
k_2 &= f\left(t_i + \frac{h}{2}, x_i + \frac{k_1 h}{2}\right) \\
k_3 &= f\left(t_i + \frac{h}{2}, x_i + \frac{k_2 h}{2}\right) \\
k_4 &= f\left(t_i + h, x_i + k_3 h\right)
\end{aligned} \tag{27}$$

Please notice how k_j depends on k_{j-1} in Eq. (27), which is expressed in Fig. 4. We have here $a_1 = 1/6$, $a_2 = 2/6$, $a_3 = 2/6$ and $a_4 = 1/6$. Also, $p_1 = ½$, $p_2 = ½$, $p_3 = 1$, $q_{11} = ½$, $q_{22} = ½$ and $q_{33} = 1$. As k_3 only depends on k_2, not k_1; k_4 only on k_3, not on k_2 and k_1, we have $q_{21} = q_{31} = q_{32} = 0$. The choice of q_{ij} this way is a convenient feature, but not a required necessity.

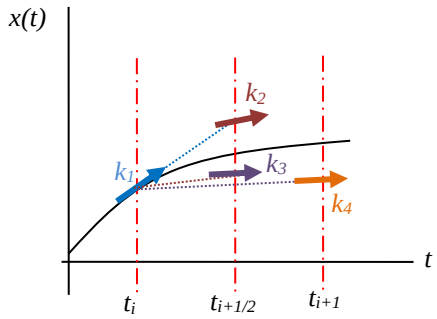


Fig. 4. The 4th-order Runge Kutta method in one time step.

Exercise: Show that the parameter choices in Eq. (27) matches the Taylor series expansion up to the fourth order.

From Eq. (21), we will show the match in the first order (one term before $f(t_i, x_i)$) and the second order (one term before $\frac{\partial f}{\partial t}$, and the other before $\frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$):

First order: $a_1 + a_2 + a_3 + a_4 = \frac{1}{6} + \frac{2}{6} + \frac{2}{6} + \frac{1}{6} = 1$

Second order for $\frac{\partial f}{\partial t}$: $a_2 p_1 + a_3 p_2 + a_4 p_3 = \frac{2}{6}\cdot\frac{1}{2} + \frac{2}{6}\cdot\frac{1}{2} + \frac{1}{6}\cdot 1 = \frac{1}{2}$

Second order for $\frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$: $a_2 q_{11} + a_3 q_{22} + a_4 q_{33} = \frac{2}{6}\cdot\frac{1}{2} + \frac{2}{6}\cdot\frac{1}{2} + \frac{1}{6}\cdot 1 = \frac{1}{2}$

Notice in the present choice of k_2, k_3 and k_4, we have $q_{21} = q_{31} = q_{32} = 0$.

The third order term has three conditions of $\frac{\partial^2 f}{\partial t^2}$, $\frac{\partial f}{\partial x}\cdot\frac{\partial^2 x}{\partial t^2}$ and $\frac{\partial^2 f}{\partial x^2}\cdot\frac{\partial^2 x}{\partial t^2}$ to match to 1/6.

□

Example 3. Following Table 1, if we use the 4th-order Runge Kutta, we will have the following computation for the ODE of $\frac{dx}{dt} = 4e^{0.8t} - 0.5x$, with the exact solution for $x(0) = 2$ as:

$x(t) = \frac{4}{1.3}\left(e^{0.8t} - e^{-0.5t}\right) + 2e^{-0.5t}$.

Table 2. Fourth-order Runge Kutta calculation:

t	x_{true}	k_1	k_2	k_3	k_4	x_{RK4}	$\lvert\varepsilon_x\rvert$ (%)
0	2.0000			2.0000		2.0000	
1	6.1946	3.0000	4.2173	3.9130	5.9457	6.2010	0.103

We can see how the 4th-order Runge Kutta behaves much better than any other methods of forward Euler, trapezoidal Euler and even the predictor-corrector with iterations in this example!

□

Before we close on the Runge Kutta method, we will make one further observation on stability, as defined in Example 1. We have seen that the back Euler method is "A-stable", as the solution will converge to 0 for the test problem of $\frac{dx}{dt} = -ax$ with $a > 0$ and any choice of h. The Runge Kutta method for the test problem can be generally expressed as:

$$x_{i+1} = x_i + \phi(ha)h = x_0\left(\phi(ha)\cdot x_i\right) = x_0\left(\phi(ha)\right)^{i+1} \tag{28}$$

The Runge Kutta method will be A-stable if $|\phi(ha)| < 1$. On top of A-stability, the Runge Kutta method (as well as the backward Euler) is said to be L-stable for $\lim_{ha\to\infty} \phi(ha) \to 0$, i.e., with the choice of $h = \frac{1}{a^m}; m > 1$, the Runge Kutta method can converge in ONE step! The L-stable adaptive Runge Kutta method is most appropriate for the system with high stiffness.

4. The Adaptive Runge Kutta Method for Stiff Systems

Up to now we have not treated the choice of step size. For a "**stiff system**"[4] with multiple time constants of very different magnitudes. This is especially true for digital and RF circuit systems. In the digital system, when a circuit node is in a "0" or "1" state, it can remain there indefinitely (with the clock signal as an exception). If the time step h is very small in this state, not only it would waste a lot of computation doing nothing (nothing has happened), but the round-off errors will accumulate indefinitely. However, during a transition, it is important to capture the voltage transient waveforms accurately with much smaller time step h so that the short-circuit power can be estimated. If h is large during the transition, we may miss the whole ball game! In the RF system, the base data band often has a much lower frequency than the carrier frequency (or you would have a very large antenna or one person hogging the entire spectrum). The time-domain electromagnetic effects by default form a stiff system (hopefully not a chaotic system).

Surely for a large stiff system, there is NO way we can choose the time step manually. We cannot rely solely on the transient external condition either (although the external time dependence of $4e^{0.8t}$ in Examples 2 and 3 has to be reasonably resolved to be close to the true solution), as the internal node can have a fast transition even when the external time stimulus is slow due to the nonlinearity in the system, such as the turning-on of a diode or transistor where the current changes exponentially. We do not have the luxury here to treat the formal definition of system stiffness through the eigenvalues in Eq. (1), but have to resort to more intuitive ways of thinking.

The first intuitive way for adaptive time stepping is to evaluate *h/2* and *2h*, together with the present time step *h*. However, not only we have to pay three times the execution time, but also the error estimation is

[4] The definition of system stiffness, as put by J. D. Lambert, is "If a numerical method with a finite region of A-stability, applied to a system with any given initial conditions, is forced to use a certain interval of integration step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval. (Sorry that mathematicians use very long sentences to be precise).

rather risky since we generally have no idea about the true solution at the present state. Do not get me wrong, during debugging of your software, this still offers a helpful clue as in the Richardson extrapolation. Can we have a more efficient way of error estimation within the present time step? The answer lies in the difference of two order-of-accuracy Runge Kutta discretization. Different order of accuracy means *p* adaptive in the *hp* adaptivity, and it is a generally good practice that we use both, and particularly good if we "**use *p* to evaluate *h*"**[5]! Not only this will relieve the requirement of some knowledge about the true solution, but also it can be very computationally efficient, as illustrated in the adaptive Runge Kutta method, or so-called "embedded Runge Kutta".

4.1 The adaptive Runge Kutta methods by different orders

We will use an example to illustrate the error estimation in the adaptive Runge Kutta method. By choosing a 3rd-order Runge Kutta discretization as:

$$\begin{aligned} x_{i+1} &= x_i + \frac{1}{9}\left(2k_1 + 3k_2 + 4k_3\right)h \\ k_1 &= f\left(t_i, x_i\right) \\ k_2 &= f\left(t_i + \frac{h}{2}, x_i + \frac{k_1 h}{2}\right) \\ k_3 &= f\left(t_i + \frac{3h}{4}, x_i + \frac{3k_2 h}{4}\right) \end{aligned} \tag{29}$$

This will be the main time stepping scheme to update x_{i+1} from x_i. Notice that in the Runge Kutta method, to match with the first-order term from the Taylor series, we will have $\sum_j a_j = 1$, similar to Eq. (24).

Now we will formulate a 4th-order Runge Kutta discretization as aligned as possible at k_j to the 3rd-order in Eq. (29) by:

$$\begin{aligned} x_{i+1} &= x_i + \frac{1}{24}\left(7k_1 + 6k_2 + 8k_3 + 3k_4\right)h \\ k_1 &= f\left(t_i, x_i\right) \\ k_2 &= f\left(t_i + \frac{h}{2}, x_i + \frac{k_1 h}{2}\right) \\ k_3 &= f\left(t_i + \frac{3h}{4}, x_i + \frac{3k_2 h}{4}\right) \\ k_4 &= f\left(t_i + h, x_{i+1}\right) \end{aligned} \tag{30}$$

Notice how $k_1 - k_3$ are aligned to save the computational cost of evaluating *f* at different instance of the time interval $[t_i, t_{i+1}]$. This is made possible as we have *one parameter of free choice* in each Runge Kutta expression! We can now write the **error estimator** from the difference of Eqs. (29) and (30):

[5] In the circuit simulation and electrostatic problems, actually we more often use *hp* to evaluate/choose *h* in the universal expression of Runge Kutta, but this will require a multi-step method in the next section. The popular TR-BDF2 method in SPICE belongs to this category.

$$E_{i+1} \equiv x_{i+1,3rd} - x_{i+1,4th} = \frac{1}{72}\left(-5k_1 + 6k_2 + 8k_3 - 9k_4\right)h \tag{31}$$

The best thing of this error estimator is: it is almost computationally FREE in terms of evaluation of *f*! We need to calculate for $k_1 - k_3$ for the regular time step in Eq. (29), and k_4 can be re-used for the next time step. The linear combination of k_j in Eq. (31) is often fast and does not involve any nonlinear functions!

Exercise: Show that the parameter choices in Eq. (29) match the Taylor series expansion up to the third order and those in Eq. (30) match the Taylor series expansion to the fourth order.

$\sum_j a_j = 1$ for matching the term before $f(t_i, x_i)$, which is true for both Eqs. (29) and (30).

Second order for $\frac{\partial f}{\partial t}$ of Eq. (29): $a_2 p_1 + a_3 p_2 = \frac{3}{9}\cdot\frac{1}{2} + \frac{4}{9}\cdot\frac{3}{4} = \frac{1}{2}$.

Second order for $\frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$ of Eq. (29): $a_2 q_{11} + a_3 q_{22} = \frac{3}{9}\cdot\frac{1}{2} + \frac{4}{9}\cdot\frac{3}{4} = \frac{1}{2}$. Notice that this is the same calculation because $q_{21} = 0$.

Second order for $\frac{\partial f}{\partial t}$ of Eq. (30): $a_2 p_1 + a_3 p_2 + a_4 p_3 = \frac{6}{24}\cdot\frac{1}{2} + \frac{8}{24}\cdot\frac{3}{4} + \frac{3}{24}\cdot 1 = \frac{1}{2}$

Second order for $\frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$ of Eq. (30) is the same as $q_{21} = q_{31} = q_{32} = 0$.

You can derive the rest orders for the match.

□

Now we have calculated the error estimator for the present choice of *h* from adaptive *p* in the Runge Kutta expression. We can evaluate a norm of $\|E_{i+1}\|$ against a norm of $\|x_{i+1}\|$ for a possible adaptive time stepping scheme:

1. If $\frac{\|E_{i+1}\|}{\|x_{i+1}\|} > Tol1$, refine *h* to *h/2*.
2. If $\frac{\|E_{i+1}\|}{\|x_{i+1}\|} < Tol2$, extend *h* to 2*h*.
3. If $Tol2 < \frac{\|E_{i+1}\|}{\|x_{i+1}\|} < Tol1$, keep the present choice of *h*.

Typical values of *Tol*1 and *Tol*2 are 10^{-1} and 10^{-3}, respectively, depending on the precision of the floating point numbers and the stiffness in the system. With the proper choice of *h*, the adaptive Runge Kutta

above can achieve 0.001% accuracy for the simple problem in Example 1. With double precision in the floating number calculation, 10^{-7} accuracy can be readily achieved!

If we are very confident about the error estimator here, we can use $\frac{\|E_{i+1}\|}{\|x_{i+1}\|}$ to guide the ratio of adaptive time stepping, so that the time steps can change much faster than 2. We can define:

$$r = \frac{\|E_{i+1}\|}{\varepsilon_R \|x_{i+1}\| + \varepsilon_A} \tag{31}$$

where ε_R is a relative error tolerance and ε_A is an abosolute tolerance in case $\|x_{i+1}\|$ underflows to zero. The adaptive time steps can be:

1. When $r > 2$ or $r < 0.5$, $h_{i+1} = \frac{h_i}{r^m}$, where m is a user defined factor for how aggressive h should change with the present error estimator. A typical value for m is 1. When $m = 2$, the time step will react strongly to the error estimator, when $m = 0.5$, the time step will change slowly with the error estimator.

2. For $0.5 \le r \le 2$, we will keep the time step size.

The popular orders of precision are implemented in most numerical packages such as RK23[6], RK34 and RK45. Some systems have only odd or even-order terms, in which case there would be one more f evaluation to implement RK35, for example.

We can see that the adaptive Runge Kutta method has achieved a lot, especially offering the error estimator that cannot be easily performed in Euler's or Huen's methods. One remain disadvantage is that the intermediate k (ex. k_2 and k_3 in RK34) has evaluated $f(t, x)$, but there is no solution of $x(t+h/2)$ or $x(t+3h/4)$ available. In the multi-variable case when evaluation of $f(t, x)$ is very expensive, it is not computational efficient in terms of the sampling frequency[7]. This is where the multi-step TR-BDF2 comes in to make one last improvement in the adaptive time stepping scheme.

5. The Multi-Step Methods for ODE Solution

From the local analysis, we have noticed that the more instance we use in "history", the higher order of precision can be achieved. For instance, Eq. (7) involves two time instances of t and $t + h$ and is first-order accurate (forward or backward Euler), Eq. (8) involves $t - h$ and $t + h$ and is second order accurate (trapezoidal Euler), and Eq. (11) involves $t + h_1$, $t + h_2$ and t and is second order as well (3-point local analysis). When we are beyond the first few iterations (say $i > 3$ in Eq. (13)), it would seem wasteful that we evaulate x_{i+1} ONLY from x_i, but neglect the information in x_{i-1}, if not further into history. This is not only for accuracy consideration because of error accumulation, but also for stability[8]. A small caveat in

[6] We can see the popular TR-BDF2 as a variation of RK23 with better computational efficiency, which will be clear later.
[7] Sampling frequency in many realistic cases is often limited by the Nyquist frequency.
[8] We need to be more careful here, even in the philosophical sense. For any behavior that depends on history, it can be more stable to avoid making the same mistake, but in danger of never going anywhere by claiming there is nothing new under the sun.

the multi-step method is that the multi-step method may not be **self starting** unless it is used in the hybrid method with an initial one-step method. For example, when we calculate x_1, there is only x_0 available.

We will introduce two multi-step methods, the modified predictor and corrector method for intuition, and the TR-BDF2 for popularity. TR-BDF2 has many similarities in its final form with an adaptive Runge Kutta (RK23), but the logic stems from the multi-step method.

5.1 The modified predictor-corrector method with 2nd-order accuracy

Let's revisit the predictor-corrector method:

Predictor: $x_{i+1}^{j=0} = x_i + f(t_i, x_i)h$

Corrector: $x_{i+1} = x_i + \phi h = x_i + \dfrac{f(x_i, t_i) + f\left(x_{i+1}^{j=0}, t_{i+1}\right)}{2} h$

The two steps are designed to have a more accurate estimate of the increment function ϕ at the corrector step. However, we also notice that the predictor is first-order accurate and the corrector is second-order accurate. One simple way to achieve better order of accuracy is to modify the predictor to be second-order accurate by accessing $t = t - h$, where the predictor becomes second-order as well (assuming that we have constant time steps now):

Predictor: $x_{i+1}^{j=0} = x_{i-1} + f(t_i, x_i) \cdot 2h$

Now the predictor-corrector method is 2nd order and mutliple steps, as we use t_{i-1} and t_i to evaluate t_{i+1}. The simple modification indeed can improve the accuracy in our sample problem in Example 1, but have the following disadvantages:

1. The trapezoidal rule in both the predictor and corrector steps are only A-stable, not L-stable.
2. A generalization to inhomogeneous time steps will be helpful.
3. It is difficult to write down an error estimator.

However, the multi-step predictor-corrector method does inspires other composite methods, of which the most important one is TR-BDF2 (trapezoid-backward-differentiation-forumula-2nd order), which can be cast into a RK23 method but offer higher computational efficiency in terms of time sampling frequency.

5.2 The TR-BDF2 method

In the spirit of the two-step predictor-corrector method above, we will look at the composite two-step TR-BDF2 method:

1. We will use trapezoidal Euler to make a prediction of $x(t = t_i + \gamma h)$. Notice that TR is second-order accurate, A-stable, but not L-stable.

$$\frac{x_{i+\gamma} - x_i}{\gamma h} = \frac{f_i + f_{i+\gamma}}{2}; \qquad x_{i+\gamma} = x_i + \frac{\gamma h}{2} \cdot \left(f_i + f_{i+\gamma}\right) \tag{32}$$

2. We will use a 2nd-order backward differentiation in Eq. (11) by setting $h_1 = -h$ and $h_2 = -(1-\gamma)h$ to take the march to $t + h$. Notice that all backward differentiation is both A-stable and L-stable.

$$f_{i+1} = \left.\frac{dx_{i+1}}{dt}\right|_{BDF2;h1=-h;h2=-(1-\gamma)h} = \frac{1}{h}\left[\frac{-1}{(1-\gamma)\gamma}x_{i+\gamma} + \frac{(2-\gamma)}{(1-\gamma)}x_{i+1} + \frac{1-\gamma}{\gamma}x_i\right] \tag{33}$$

$$x_{i+1} = -\frac{(1-\gamma)^2}{\gamma(2-\gamma)}x_i + \frac{1}{\gamma(2-\gamma)}x_{i+\gamma} + \frac{(1-\gamma)}{(2-\gamma)}hf_{i+1} \tag{34}$$

Notice that we have use the shorthand on the subscripts i, $i+\gamma$, and $i+1$ to denote the instance of $x(t)$ and $f(t, x)$ evaluated at time t_i, $t_i + \gamma h$ and $t_i + h$, as shown in Fig. 5.

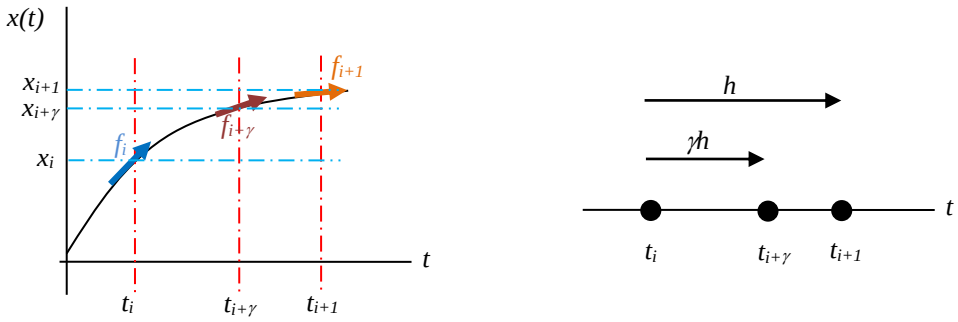


Fig. 5. The composite time steps in TR-BDF2. We have use the subscript for time instance shorthand.

In the spirit of adaptive Runge Kutta of using p to evaluate h, we can write down the error estimator by the difference between BDF and BDF2, where we have all f_i, $f_{i+\gamma}$, and f_{i+1} already evaluated. The algebra is a bit involved (but not difficult), but the result is very useful.

$$E_{i+1} = \frac{3\gamma^2 - 4\gamma + 2}{6(\gamma - 2)} \cdot \left(\frac{1}{\gamma}f_i - \frac{1}{\gamma(1-\gamma)}f_{i+\gamma} + \frac{1}{1-\gamma}f_{i+1}\right) \tag{35}$$

If we take the partial derivative with γ to the prefactor of E_{i+1} to be zero, we find that:

$$\frac{\partial\left(\frac{3\gamma^2 - 4\gamma + 2}{6(\gamma - 2)}\right)}{\partial\gamma} = 0 \quad \Rightarrow \quad \gamma = 2 \pm \sqrt{2} \tag{36}$$

As γ is between 0 and 1, the optimal value of γ to make the smallest prefactor for the error estimator is $\gamma = 2 - \sqrt{2}$. This is not a forced choice, as any γ will still maintain second-order precision in TR-BDF2. We can use Eq. (31) for the adaptive time stepping scheme.

We can make the following observations for TR-BDF2:

1. It is second-order accurate and L-stable (inherited from BDF2).
2. For every evaluation of f, we have a solution of $x(t)$ at that time instance.

3. The error estimator does not only give guidance to time stepping, but also provides an optimal value for γ.

TR-BDF2 has been very popular for ODE where 2^{nd}-order precision is sufficient. This is the default method in SPICE and many other CAD programs.