Lecture 31

Higher Order Methods

The order of a method

For numerical solutions of an initial value problem there are two ways to measure the error. The first is the error of each step. This is called the Local Truncation Error or LTE. The other is the total error for the whole interval $[a, b]$. We call this the Global Truncation Error or GTE.

For the Euler method the LTE is of order $O(h^2)$, i.e. the error is comparable to h^2. We can show this directly using Taylor's Theorem:

$$\mathbf{y}(t+h) = \mathbf{y}(t) + h\dot{\mathbf{y}}(t) + \frac{h^2}{2}\ddot{\mathbf{y}}(c)$$

for some c between t and $t+h$. In this equation we can replace $\dot{\mathbf{y}}(t)$ by $f(t, \mathbf{y}(t))$, which makes the first two terms of the right hand side be exactly the Euler method. The error is then $\frac{h^2}{2}\ddot{\mathbf{y}}(c)$ or $O(h^2)$. It would be slightly more difficult to show that the LTE of the modified Euler method is $O(h^3)$, an improvement of one power of h.

We can roughly get the GTE from the LTE by considering the number of steps times the LTE. For any method, if $[a, b]$ is the interval and h is the step size, then $n = (b-a)/h$ is the number of steps. Thus for any method, the GTE is one power lower in h than the LTE. Thus the GTE for Euler is $O(h)$ and for modified Euler it is $O(h^2)$.

By the **order** of a method, we mean the power of h in the GTE. Thus the Euler method is a 1st order method and modified Euler is a 2nd order method.

Fourth Order Runge-Kutta

The most famous of all IVP methods is the classic Runge-Kutta method of order 4:

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(t_i, \mathbf{y}) \\
\mathbf{k}_2 &= h\mathbf{f}(t_i + h/2, \mathbf{y}_i + \mathbf{k}_1/2) \\
\mathbf{k}_3 &= h\mathbf{f}(t_i + h/2, \mathbf{y}_i + \mathbf{k}_2/2) \\
\mathbf{k}_4 &= h\mathbf{f}(t_i + h, \mathbf{y}_i + \mathbf{k}_3) \\
\mathbf{y}_{i+1} &= \mathbf{y}_i + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right).
\end{aligned} \tag{31.1}$$

Notice that this method uses values of $\mathbf{f}(t, \mathbf{y})$ at 4 different points. In general a method needs n values of $\mathbf{f}$ to achieve order n. The constants used in this method and other methods are obtained from Taylor's Theorem. They are precisely the values needed to make all error terms cancel up to $h^{n+1}\mathbf{f}^{(n+1)}(c)/(n+1)!$.

Variable Step Size and RK45

If the order of a method is n, then the GTE is comparable to h^n, which means it is approximately Ch^n, where C is some constant. However, for different differential equations, the values of C may be very different. Thus it is not easy beforehand to tell how small h should be to get the error within a given tolerence. For instance, if the true solution oscillates very rapidly, we will obviously need a smaller step size than for a solution that is nearly constant.

How can a program then choose h small enough to produce the required accuracy? We also do not wish to make h much smaller than necessary, since that would increase the number of steps. To accomplish this a program tries an h and tests to see if that h is small enough. If not it tries again with a smaller h. If it is too small, it accepts that step, but on the next step it tries a larger h. This process is called **variable step size**.

Deciding if a single step is accurate enough could be accomplished in several ways, but the most common are called **embedded methods**. The Runge-Kutta 45 method, which is used in `ode45`, is an embedded method. In the RK45, the function f is evaluated at 5 different points. These are used to make a 5th order estimate $\mathbf{y}_{i+1}$. At the same time, 4 of the 5 values are used to also get a 4th order estimate. If the 4th order and 5th order estimates are close, then we can conclude that they are accurate. If there is a large discrepency, then we can conclude that they are not accurate and a smaller h should be used.

To see variable step size in action, we will define and solve two different ODEs and solve them on the same interval. Create this script and run it:

```
% illustrates variable step size in RK45
dy1 = @(t,y) [-y(2);y(1)];           % create two ODE IVPs
dy2 = @(t,y) [-5*y(2);5*y(1)];
[T1 Y1] = ode45(dy1,[0 20],[1;0]);   % solve with ode45
[T2 Y2] = ode45(dy2,[0 20],[1;0]);
y1 = Y1(:,1);                        % extract position variables
y2 = Y2(:,1);
plot(T1,y1,'bx-')                    % plot both together
hold on
plot(T2,y2,'ro-')
size(T1)                             % print number of steps used
size(T2)
hold off
```

Why order matters

Many people would conclude on first encounter that the advantage of a higher order method would be that you can get a more accurate answer than for a lower order method. In reality, this is not quite how things work. In engineering problems, the accuracy needed is usually a given and it is usually not extremely high. Thus getting more and more accurate solutions is not very useful. So where is the advantage? Consider the following example.

Suppose that you need to solve an IVP with an error of less than 10^{-4}. If you use the Euler method, which has GTE of order $O(h)$, then you would need $h \approx 10^{-4}$. So you would need about $n \approx (b-a) \times 10^4$ steps

to find the solution.

Suppose you use the second order, modified Euler method. In that case the GTE is $O(h^2)$, so you would need to use $h^2 \approx 10^{-4}$, or $h \approx 10^{-2}$. This would require about $n \approx (b-a) \times 10^2$ steps. That is a hundred times fewer steps than you would need to get the same accuracy with the Euler method.

If you use the RK4 method, then h^4 needs to be approximately 10^{-4}, and so $h \approx 10^{-1}$. This means you need only about $n \approx (b-a) \times 10$ steps to solve the problem, i.e. a thousand times fewer steps than for the Euler method.

Thus the real advantage of higher order methods is that they can run a lot faster at the same accuracy. This can be especially important in applications where one is trying to make real-time adjustments based on the calculations. Such is often the case in robots and other applications with dynamic controls.

Exercises

31.1 There is a Runge-Kutta 2 method, which is also known as the midpoint method. It is summarized by the following equations:

$$\begin{aligned} \mathbf{k}_1 &= h\mathbf{f}(t_i, \mathbf{y}_i) \\ \mathbf{k}_2 &= h\mathbf{f}(t_i + h/2, \mathbf{y}_i + \mathbf{k}_1/2) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \mathbf{k}_2. \end{aligned} \tag{31.2}$$

(a) Modify the program `mymodeuler` into a program `myRK2` that does the RK2 method.

(b) Test `myRK2` and `mymodeuler` on the following IVP with time span $[0, 4\pi]$:

$$\ddot{x} + x = 0 \quad \text{with} \quad x(0) = 1 \quad \text{and} \quad \dot{x}(0) = 0.$$

Using `format long`, make a table with $\mathbf{y}_{n+1}$ for each of the two programs for $n = 10$, 100, and 1000. Compute the difference between $\mathbf{y}_{n+1}$ and the true solution $\mathbf{y}(4\pi) = (1, 0)$.

Turn in the modified program and a summary of the results.