LECTURE 6

Numerical Methods for ODEs, II

1. Runge-Kutta Methods

In Lecture 5, we discussed the Euler method; a fairly simple iterative algorithm for determining the solution of an intial value problem

$$\frac{dx}{dt} = F(t,x) \quad , \quad x(t_0) = x_0 \; . \tag{0}$$

The key idea was to interprete the $F(x,t)$ as the slope m of the best straight line fit to the graph of a solution at the point (t,x). Knowing the slope of the solution curve at (t_0,x_0) we could get to another (approximate) point on the solution curve by following the best straight-line-fit to a point $(t_1,x_1) = (t_0+\Delta t, x_0+m_0\Delta t)$, where $m_0 = F(t_0,x_0)$. And then we could repeat this process to find a third point $(t_2,x_2) = (t_1+\Delta t, x_1+m_1\Delta t)$, and so on. Iterating this process n times gives us a set of $n+1$ values $x_i = x\left(t_i\right)$ for an approximate solution on the interval $[t_0, t_0+n\Delta t]$.

Now recall from our discussion of the numerical procedures for calculating derivatives (Lecture 6) that the formal definition

$$\frac{dx}{dt} = \lim_{h\to 0} \frac{x(t+h)-x(t)}{h}$$

does not actually provide the most accurate numerical procedure for computing derivatives. For while

$$\frac{dx}{dt} = \frac{x(t+h)-x(t)}{h} + \mathcal{O}(h) \qquad ,$$

a more accurate formula would be

$$\frac{dx}{dt} = \frac{4}{3h}\left(x(t+h/2)-x(t-h/2)\right) - \frac{1}{6h}\left(x(t+h)-x(t-h)\right) + \mathcal{O}\left(h^4\right)$$

and even more accurate formulas were possible using Richardson Extrapolations of higher order.

In a similar vein, we shall now seek to improve on the Euler method. Let us begin with the Taylor series for $x(t+h)$:

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \frac{h^3}{6}x'''(t) + \mathcal{O}(h^4)$$

From the differential equation we have (by differentiating the differential equation and applying the two-variable chain rule)

$$\begin{aligned}
\frac{d^2x}{dt^2} &= \frac{d}{dt}\frac{dx}{dt} \\
&= \frac{d}{dt}F\left(t,x\right) \\
&= \frac{\partial F}{\partial t}\left(t,x\right)\frac{dt}{dt} + \frac{\partial F}{\partial x}\frac{dx}{dt} \\
&= F_t\left(t,x\right) + F_x\left(t,x\right)F\left(t,x\right)
\end{aligned}$$

And so the Taylor series for $x(t+h)$

$$x(t+h) = x(t) + hx'\left(t\right) + \frac{h^2}{2}x''\left(t\right) + \mathcal{O}(h^3)$$

can be expressed interms of $F(t,x)$ and its derivatives

$$x(t+h) = x(t) + hF(x,t) + \frac{h^2}{2}\left(F_t(t,x) + F_x(t,x)F(t,x)\right) + \mathcal{O}\left(h^3\right)$$

which after regrouping terms is the same as

$$x(t+h) = x(t) + \frac{1}{2}hF(t,x) + \frac{1}{2}h\left(F(t,x) + hF_t(t,x) + hF(t,x)F_x(t,h)\right) + \mathcal{O}(h^3) \tag{2}$$

Now, regarding $F(t+h, x+hF(t,x))$ as a function of h, we have the following Taylor expansion

$$F(t+h, x+hF(t,x)) = F(t,x) + hF_t(t,x) + F_x(t,h)\left(hF(t,h)\right) + \mathcal{O}(h^2) \tag{3}$$

Now you see the purpose of the regrouping in (2); the tail end of (2) coincides with $\frac{1}{2}h$ times the right hand side of (3). This allows us to write

$$x(t+h) = x(t) + \frac{h}{2}F(t,x) + \frac{h}{2}F\left(t+h, x+hF(t,x)\right) + \mathcal{O}\left(h^3\right)$$

or

$$x(t+h) \approx x(t) + \frac{1}{2}\left(f_1 + f_2\right) \tag{4}$$

where

$$f_1 \equiv hF(t,x) \tag{5}$$

$$f_2 \equiv hF\left(t+h, x+f_1\right) \tag{6}$$

Notice that formulas (4), (5), and (5) allow us to use initial data $x(t_0) = x_0$, to compute an approximate value for $x(t_0+h)$ in three steps:

(i) compute $f_1 = hF(t_0, x_0)$
(ii) compute $f_2 = hF(t_0+h, x+f_1)$
(iii) compute $x(t+h) = x_0 + \frac{1}{2}(f_1+f_2)$

We thus arrive at the following algorithm for computing a solution to the intial value problem (1):

(1) Partition the solution interval $[a,b]$ into n subintervals:

$$\begin{aligned} \Delta t &= \frac{b-a}{n} \\ t_k &= a + k\Delta t \end{aligned}$$

(2) Set x_0 equal to $x(a)$ and then for k from 0 to $n-1$ calculate

$$\begin{aligned} f_{1,k} &= \Delta t F(t_k, x_k) \\ f_{2,k} &= \Delta t F\left(t_k + \Delta t, x_k + \Delta t f_{1,k}\right) \\ x_{k+1} &= x_k + \frac{1}{2}\left(f_{1,k} + f_{2,k}\right) \end{aligned}$$

(3) This method is known as **Heun's method** or the **second order Runge-Kutta method** .

Higher order Runge-Kutta methods are also possible; however, they are very tedius to derive. Here is the formula for the **classical fourth-order Runge-Kutta method**:

$$x(t+h) = x(t) + \frac{1}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right)$$

where

$$\begin{aligned} f_1 &= hF(t,x) \\ f_2 &= hF\left(t+\frac{1}{2}h, x+\frac{1}{2}f_1\right) \\ f_3 &= hF\left(t+\frac{1}{2}h, x+\frac{1}{2}f_2\right) \\ f_4 &= hF\left(t+h, x+f_3\right) \end{aligned}$$

Below is a Maple program that implements the fourth order Runge-Kutta method to solve

$$\frac{dx}{dt} = -\frac{x^2+t^2}{2xt} \quad , \quad x(1)=1 \tag{7}$$

on the interval $[1,2]$.

```
F := (x,t) -> - (x^2 +t^2)/(2*x*t) ;
n := 100;
t[0] := 1.0;
x[0] := 1.0;
h := 1.0/n
for i from 0 to n-1 do
   f1 := evalf(h*F(t[i],x[i]));
   f2 := evalf(h*F(t[i]+h/2,x[i]+f1/2));
   f3 := evalf(h*F(t[i] +h/2,x[i]+f2/2));
   f4 := evalf(h*F(t[i]+h,x[i]+f3));
   t[i+1] := t[i] +h;
   x[i+1] := x[i] +(f1 +2*f2+2*f3+f4)/6;
od:
```

The exact solution to (7) is

$$x(t) = \sqrt{\frac{1}{3}\left(\frac{4}{t}-t^3\right)}$$

2. Error Analysis for the Runge-Kutta Method

Recall from the preceding lecture the formula underlying the fourth order Runge-Kutta Method: if $x(t)$ is a solution to

$$\frac{dx}{dt} = F(t,x)$$

then

$$x(t_0+h) = x(t_0) + \frac{1}{6}\left(f_1+2f_2+2f_3+f_4\right) + \mathcal{O}(h^5)$$

where

$$\begin{aligned} f_1 &= hF(t_0, x_0) \\ f_2 &= hF\left(t_0 + \frac{1}{2}h, x_0 + \frac{1}{2}f_1\right) \\ f_3 &= hF\left(t_0 + \frac{1}{2}h, x_0 + \frac{1}{2}f_2\right) \\ f_4 &= hF(t_0 + h, x_0 + f_3) \end{aligned}$$

Thus, the term $\mathcal{O}\left(h^5\right)$ is the "local truncation error"; it corresponds to the error induced for each successive stage of the iterated algorithm; at each stage the difference between the computed value and the actual value will be of the form

$$err = Ch^5$$

for some constant C. Here C is a number independent of h, but dependent on t_0 and the fourth derivative of the exact solution $\tilde{x}(t)$ at t_0 (the constant factor in the error term corresponding to truncating the Taylor series for $x(t_0 + h)$ about t_0 at order h^4. To estimate Ch^5 we shall assume that the constant C does not change much as t varies from t_0 to $t_0 + h$.

Let u be the approximate solution to $\tilde{x}(t)$ at $t_0 + h$ obtained by carrying out a one-step fourth order Runge-Kutta approximation:

$$\tilde{x}(t) = u + Ch^5$$

Let v be the approximate solution to $\tilde{x}(t)$ at $t_0 + h$ obtained by carrying out a two-step fourth order Runge-Kutta approximation with step sizes of $\frac{1}{2}h$

$$\tilde{x}(t) = v + 2C\left(\frac{h}{2}\right)^5$$

Substracting these two equations we obtain

$$0 = u - v + C\left(1 - 2^{-4}\right)h^5$$

or

$$\text{local truncation error } = Ch^5 = \frac{u - v}{1 - 2^{-4}} \approx u - v$$

In a computer program that uses a Runge-Kutta method, this local truncation error can be easily monitored, by occasionally computing $|u - v|$ as the program runs through its iterative loop. Indeed, if this error rises above a given threshold, one can readjust the step size h on the fly to restore a tolerable degree of accuracy. Programs that uses algorithms of this type are known as **adaptive Runge-Kutta methods**.

3. Systems of First Order ODEs

It turns out the that the Runge-Kutta method just describes is easily extendable to the situation of a system of first order ODEs. Indeed, consider such a system expressed in matrix notation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(t, \mathbf{x}) = \begin{pmatrix} f_1(t, \mathbf{x}) \\ \vdots \\ f_n(t, \mathbf{x}) \end{pmatrix}$$

Then partitioning the interval in question as $t_0, t_1 = t_0 + h, t_2 = t_0 + 2h, \ldots$ and setting

$$\begin{aligned} \mathbf{x}_k &\approx \mathbf{x}(t_k) \\ \mathbf{F}_k &= \mathbf{F}(t_k, \mathbf{x}_k) \end{aligned}$$

and following the derivation of the Runge-Kutta formula above one arrives at the following recursive formula for $\mathbf{x}_{k+1}$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \left(\frac{h}{6}\right)\left(\mathbf{f}_{1,k} + 2\mathbf{f}_{2,k} + 2\mathbf{f}_{3,k} + \mathbf{f}_{4,k}\right)$$

where

$$\begin{aligned}
\mathbf{f}_{1,k} &= \mathbf{F}\left(t_k, \mathbf{x}_k\right) \\
\mathbf{f}_{2,k} &= \mathbf{F}\left(t_k + \frac{h}{2}, \mathbf{x}_k + \frac{h}{2}\mathbf{f}_{1,k}\right) \\
\mathbf{f}_{3,k} &= \mathbf{F}\left(t_k + \frac{h}{2}, \mathbf{x}_k + \frac{h}{2}\mathbf{f}_{2,k}\right) \\
\mathbf{f}_{4,k} &= \mathbf{F}\left(t_k + h, \mathbf{x}_k + h\mathbf{f}_{3,k}\right)
\end{aligned}$$

4. Multistep Methods

Here we will develop another method for improving the accuracy of numerical solutions of first order ODEs. Like the original Euler method begin with a first order ODE

$$\frac{dx}{dt} = F\left(t, x\right)$$

and a systematic partition of the interval upon which we want to know $x\left(t\right)$:

$$\begin{aligned}
t_0 &= t_0 \\
t_1 &= t_0 + \Delta t \\
&\vdots \\
t_n &= t_0 + n\Delta t
\end{aligned}$$

Recall the Fundamental Theorem of Calculus

$$x\left(t_{n+1}\right) - x\left(t_n\right) = \int_{t_n}^{t_{n+1}} \frac{dx}{dt}\left(t\right) dt$$

The basic idea behind the Adams method is approximate $\frac{dx}{dt}\left(t\right)$ by a polynomial $P\left(t\right)$; which in turn can be easily integrated to yield a formula

$$x\left(t_{n+1}\right) = x\left(t_n\right) + \int_{t_n}^{t_{n+1}} P\left(t\right) dt$$

Suppose, for example, we wish to use a polynomial

$$P_1\left(t\right) = At + B$$

of degree 1 to approximate $\frac{dx}{dt} = F\left(t, x\right)$. We need two conditions on P_1 to fix he coefficients A and B. Assuming that previous $x\left(t_n\right)$ and $x\left(t_{n-1}\right)$ have already know, we can require

$$\begin{aligned}
P_1\left(t_{n-1}\right) &= f_{n-1} \equiv \left(t_{n-1}, x_{n-1}\right) \\
P_1\left(t\right) &= f_n \equiv F\left(t_n, x_n\right)
\end{aligned}$$

More explicity, we require

$$\begin{aligned}
f_{n-1} &= P_1\left(t_{n-1}\right) = At_{n-1} + B \\
f_n &= P_1\left(t_n\right) = At_n + B
\end{aligned}$$

Solving for A and B yields

$$\begin{aligned}
A &= \frac{f_n - f_{n-1}}{t_n - t_{n-1}} = \frac{f_n - f_{n-1}}{h} \\
B &= \frac{f_{n-1}t_n - f_n t_{n-1}}{t_n - t_{n-1}} = \frac{f_{n-1}t_n - f_n t_{n-1}}{h}
\end{aligned}$$

where $h = t_n - t_{n-1}$.

We can now write

$$\begin{aligned} x(t_{n+1}) &= x(t_n) + \int_{t_n}^{t_{n+1}} (At + B)\, dt \\ &= x(t_n) + \frac{A}{2}\left(t_{n+1}^2 - t_n^2\right) + B(t_{n+1} - t_n) \\ &= x(t_n) + \frac{1}{2}\left(\frac{f_n - f_{n-1}}{h}\right)(t_{n+1} - t_n)(t_{n+1} + t_n) + \left(\frac{f_{n-1}t_n - f_n t_{n-1}}{h}\right)(t_{n+1} - t_n) \\ &= x(t_n) + \frac{1}{2}(f_n - f_{n-1})(2t_n + h) + (f_{n-1}t_n - f_n(t_n - h)) \\ &= x(t_n) + \frac{3}{2}hf_n - \frac{1}{2}hf_{n-1} \end{aligned}$$

(note that, in the second to the last line, the terms involving t_n all cancel).