Lecture 25

Last time we gave 2 algorithms to solve initial value differential equations

$$\begin{cases} \dfrac{dy}{dx} = f(x,y) \\ y(x_0) = y_0 \end{cases} \qquad a = x_0 \leq x \leq b$$

1) <u>Euler's method</u>

$$y_{n+1} = y_n + h f(x_n, y_n)$$

$$n = 0, 1, 2, \ldots, N$$

$x_n = a + nh.$

h = step-size $= (b-a)/N$

y_n = approximation of y at x_n.

Euler's method has a lot of errors.
However it is stable

2) Taylor's method

$$\frac{dy}{dx} = f(x,y)$$

$$y(x_0) = y_0$$

One can expand y at a nbhd of x_0

$$y(x) = y_0 + (x-x_0)\, y'(x_0) + \frac{(x-x_0)^2}{2} y''(x_0)$$

$$+ \; \cdot \; \cdots + \frac{(x-x_0)^k}{k!} y^{(k)}(x_0)$$

$$+ \frac{(x-x_0)^{k+1}}{(k+1)!} y^{(k+1)}(\xi)$$

$$y' = f(x,y)$$

$$y'' = f' = f_x + f_y y' = f_x + f_y f$$

$$y''' = f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_x f_y + f_y^2 f$$

and so on

$$T_k(x,y) = f(x,y) + \frac{h}{2!} f'(x,y) + \frac{h^2}{3!} f''(x,y) + \cdots$$

$$+ \frac{h^{k-1}}{k!} f^{(k-1)}(x,y)$$

$$k = 1, 2, \ldots$$

Taylor's algorithm (of order k)

$$\frac{dy}{dx} = f(x, y)$$

$$y(a) = y_0$$

$$a \le x \le b$$

1) choose a step-size $h = \frac{b-a}{N}$

$$x_n = a + nh$$

2) Generate approximation y_n to $y(x_n)$ from the recursion

$$y_{n+1} = y_n + h\, T_k(x_n, y_n)$$

Disadvantages of Taylor's method

Need to compute derivative of $f(x, y(x))$

This requires symbolic differentiation.

This is difficult to implement in FORTRAN

Error estimate and convergence of Euler Method

$$\frac{dy}{dx} = f(x,y) \qquad y(x_0) = y_0$$

Assume $|f_y(x,y)| \le L$ and $|y''(x)| \le Y$ on $[x_0, b]$

$e_n = y(x_n) - y_n$ satisfies

$$|e_n| \le \frac{hY}{2L}\left(e^{(x_n - x_0)L} - 1\right)$$

Runge - Kutta Methods

short form R-K methods.

Euler's method is not very useful in practical problems because it requires a very small step size for reasonable accuracy.

Taylor's algorithm of higher order is unacceptable as a general purpose procedure because of the need to obtain higher total derivatives of $y(x)$

RK methods attempt to obtain greater accuracy and at the same time avoid the need for higher derivatives by evaluating $f(x,y)$ at selected points at each subinterval

RK-method of order 2

We choose a formula of the following form

$$(*) \quad y_{n+1} = y_n + ak_1 + bk_2$$

$$k_1 = hf(x_n, y_n)$$

$$k_2 = h\, f(x_n + \alpha h,\ y_n + \beta k_1)$$

a, b, α, β are constants to be determined so that $(*)$ will agree with Taylor algorithm of as high an order as possible.

On expanding $y(x_{n+1})$ in a Taylor series through term of order h^3 we obtain

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2} y''(x_n) + \frac{h^3}{6} y'''(x_n) + O(h^4)$$

$$(**) \quad = y(x_n) + hf(x_n, y_n) + \frac{h^2}{2}(f_x + ff_y)_n + \frac{h^3}{6}(f_{xx} + 2ff_{xy} + f_{yy}f^2 + f_xf_y + f_y^2 f)_n + O(h^4)$$

By Taylor's expansion of function of two variables we get that

$$\frac{k_2}{h} = f(x_n + \alpha h, y_n + \beta k_1)$$

$$= f(x_n, y_n) + \alpha h f_x + \beta k_1 f_y + \frac{\alpha^2 h^2}{2} f_{xx} + \alpha h \beta k_1 f_{xy} + \frac{\beta^2 k_1^2}{2} f_{yy} + O(h^3)$$

we plugin value of k_2 into $(*)$

$$y_{n+1} = y_n + a k_1 + b k_2$$

$$= y_n + a h f(x_n, y_n) + b k_2$$

$$\boxed{y_{n+1} = y_n + (a+b) h f + b h^2 (\alpha f_x + \beta f f_y) + b h^3 \left(\frac{\alpha^2}{2} f_{xx} + \alpha \beta f f_{xy} + \frac{\beta^2}{2} f^2 f_{yy} \right) + O(h^4)}$$

Comparing above eqn with $(**)$ we get that to make corresponding power of h and h^2 to agree we must have

$$a + b = 1$$
$$b\alpha = b\beta = \frac{1}{2} \quad \Bigg\} \ (***)$$

4 unknowns, 3 equations so we have one degree of freedom in $(***)$
simplest

$$a = b = \frac{1}{2}$$
$$\alpha = \beta = 1$$

$\boxed{\text{RK method of order 2}}$

Algorithm

$$\frac{dy}{dx} = f(x, y)$$
$$y(x_0) = y_0$$

generate approximations y_n to $y(x_0 + nh)$ for h fixed and $n = 0, 1, 2, \ldots$ using the formula

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2) \quad \text{with} \quad k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + h, y_n + k_1)$$

Local error of RK-method of order 2

$$y(x_{n+1}) - y_{n+1} = \frac{h^3}{12}\left(f_{xx} + 2ff_{xy} + f^2 f_{yy} + 2f_x f_y - 2ff_y^2 \right)$$

$$+ O(h^4)$$

Thus local error is $O(h^3)$ whereas local error in Euler's method is $O(h^2)$

<u>Example</u>

$$\frac{dy}{dx} = y - x^2 + 1 \qquad 0 \le x \le 2$$

$$y(0) = 0.5$$

Exact ans $\quad y(x) = (x+1)^2 - \frac{1}{2}e^x$

$$h = 0.2$$

x_i	Exact $y(x_i)$	RK order 2 y_i	Error
0	0.5	0.5000	0
0.2	0.8292986	0.826000	0.0032986
0.4	1.2140877	1.20692	0.0071677
0.6	1.6489406	1.6372424	0.0116982
0.8	2.1272295	2.1102357	0.0169938
1.0	2.6408591	2.6176876	0.0231715
1.2	3.1799415	3.1495789	0.0303627
1.4	3.7324000	3.6936862	0.0387138
1.6	4.2834838	4.2350972	0.0483866
1.8	4.8151763	4.7556185	0.0595577
2.0	5.3054720	5.2330546	0.0724173

RK method of order 4

$$\frac{dy}{dx} = f(x, y) \qquad a = x_0 \leq x \leq b$$

$$y(x_0) = y_0$$

Generate appr. y_n to $y(x_0 + nh)$ for

h fixed and $n = 0, 1, 2, \ldots$ using the formula

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + \frac{h}{2},\ y_n + \frac{1}{2}k_1\right)$$

$$k_3 = h f\left(x_n + \frac{h}{2},\ y_n + \frac{1}{2}k_2\right)$$

$$k_4 = h f(x_n + h,\ y_n + k_3)$$

local error is of $O(h^5)$

again the price we pay for the favorable discretization error is that four function evaluations are required per step.

Remark :-

Suppose we are solving

$$\frac{dy}{dx} = f(x,y)$$

$$y(x_0) = y_0 \qquad a = x_0 \le x \le b$$

(*) Euler method with $h = 0.025$
RK-method of order 2 with $h = 0.05$
RK-method of order 4 with $h = 0.1$

then most of the time RK-method of order 4 will give more accurate results.

Note the # function evaluation in all three methods in (*) is the same

Multi-Step formulas

Euler's method, Taylor algorithm of order k and RK-methods are one-step method. They require information about the solution at a single pt $x = x_n$ from which the methods proceed to obtain y at the next pt $x = x_{n+1}$.

Multistep methods make use of information about the solution at more than one point.

Let us assume that we already obtained approximation to y at a number of equally spaced points say $x_0, x_1, \ldots, x_n$.

One class of multistep methods is based on the principle of numerical integration. If we integrate the differential equation $y' = f(x, y)$ from x_n to x_{n+1} we will have

$$\int_{x_n}^{x_{n+1}} y' \, dx = \int_{x_n}^{x_{n+1}} f(x, y(x)) \, dx$$

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x, y(x)) \, dx$$

To carry out the integration we now approximate $f(x, y(x))$ by a polynomial which interpolates $f(x, y(x))$ at the $m+1$ pts $x_n, x_{n-1}, x_{n-2}, \ldots \to x_{n-m}$

This is conveniently done by the socalled Newton's backward diff formula

<u>Digression</u>

Newton's forward diff formula and Newtons backward difference formula

Newton's divided diff formula

$$P_k(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x-x_0)(x-x_1) + \cdots + f[x_0 \ldots x_k](x-x_0)(x-x_1) \cdots (x - x_{k-1})$$

Newton's divided-diff formula can be expressed in a simplified form when $x_0, x_1, \ldots, x_n$ are arranged consecutively with equal spacing.

Set $\quad h = x_{i+1} - x_i \qquad i = 0, 1, \ldots, n-1$

$$x = x_0 + sh$$

Note $\quad x_i = x_0 + ih$

So $\quad x - x_i = (s - i)h$

$$P_n(x) = P_n(x_0 + sh)$$

$$= f[x_0] + sh\, f[x_0, x_1] + s(s-1)h^2 f[x_0, x_1, x_2]$$

$$+ \cdots + s(s-1)(s-2)\cdots(s-n+1)h^n f[x_0, x_1, \ldots x_n]$$

We use binomial coeff

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}$$

So

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} k!\, h^k f[x_0, x_1, \ldots x_k]$$

forward diff operator Δ

$$\Delta f(x_i) = f(x_{i+1}) - f(x_i)$$

$$\Delta^2 f(x_i) = \Delta(\Delta f(x_i))$$

$$= \Delta(f(x_{i+1}) - f(x_i))$$

$$= f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)$$

and so-on $\quad \Delta^k f(x_i) = \Delta(\Delta^{k-1} f(x_i)) \quad k \geq 2$

Notice $\quad f[x_0, x_1] = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$

$$= \frac{1}{h} \Delta f(x_0)$$

$$f[x_0, x_1, x_2] = \frac{1}{2h}\left[\frac{\Delta f(x_1) - \Delta f(x_0)}{h}\right]$$

$$= \frac{1}{2h^2} \Delta^2 f(x_0)$$

In general

$$f[x_0, x_1, \ldots, x_k] = \frac{1}{k!\, h^k} \Delta^k f(x_0)$$

Thus we have

$$\boxed{P_n(x) = f(x_0) + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f(x_0)}$$

This is called Newton's Forward diff formula.

If the interpolating nodes are reordered from last to first as $x_n, x_{n-1}, \ldots, x_1, x_0$ we can write the interpolatory formula as

$$P_n(x) = f[x_n] + f[x_n, x_{n-1}](x - x_n)$$
$$+ f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1})(x - x_{n-2})$$
$$+ \cdots$$
$$+ f[x_n, x_{n-1}, \ldots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_0)$$

If in addition nodes are equally spaced then

$x = x_n + sh$

$x = x_i + (s + n - i)h$

$$P_n(x) = P_n(x_n + sh)$$

$$= f[x_n] + sh\, f[x_n, x_{n-1}] + s(s+1)h^2 f[x_n, x_{n-1}, x_{n-2}]$$

$$+ \cdots + s(s+1)\cdots(s+n-1)h^n f[x_n, x_{n-1}, \cdots, x_0]$$

Backward diff operator ∇

Given a seq $\{P_n\}_{n\geq 0}$ define the backward difference ∇P_n (read nabla P_n) by

$$\nabla P_n = P_n - P_{n-1} \quad \text{for } n \geq 1$$

$$\begin{aligned} \nabla^2 P_n &= \nabla(\nabla P_n) \\ &= \nabla(P_n - P_{n-1}) \\ &= \nabla P_n - \nabla P_{n-1} \\ &= P_n - 2P_{n-1} + P_{n-2} \end{aligned}$$

$\vdots$

$$\nabla^k P_n = \nabla(\nabla^{k-1} P_n) \quad \text{for } k \geq 2$$

$$f[x_n, x_{n-1}] = \frac{1}{h} \nabla f(x_n)$$

$$f[x_n, x_{n-1}, x_{n-2}] = \frac{1}{2h^2}\nabla^2 f(x_n)$$

$\vdots$

$$f[x_n, x_{n-1}, \ldots, x_{n-k}] = \frac{1}{k!\,h^k}\nabla^k f(x_n)$$

Consequently

$$P_n(x) = f(x_n) + s\nabla f(x_n) + \frac{s(s+1)}{2}\nabla^2 f(x_n)$$
$$+ \cdots + \frac{s(s+1)\cdots(s+n-1)}{n!}\nabla^n f(x_n)$$

We extend binomial coeff notation

$$\binom{-s}{k} = \frac{-s(-s-1)\cdots(-s-k+1)}{k!}$$
$$= (-1)^k \frac{s(s+1)\cdots(s+k-1)}{k!}$$

So we have

$$P_n(x) = f(x_n) + (-1)\binom{-s}{1}\nabla f(x_n) + (-1)^2\binom{-s}{2}\nabla^2 f(x_n)$$
$$+ \cdots + (-1)^n\binom{-s}{n}\nabla^n f(x_n)$$

Newton's backward diff formula

$$\boxed{P_n(x) = f(x_n) + \sum_{k=1}^{n} (-1)^k \binom{-s}{k} \nabla^k f(x_n)}$$