MM9 Euler Method and Runge-Kutta Methods

- kl.12.30-14.00 MM9 lecture
- kl.14.10 – 15.40 exercise (see notes)

1

Linear Spline Interpolation (MM8)

At the nodes $a = x_0 < x_1 < x_2 \cdots < x_n = b$, there is

$$s(x_k) = f(x_k) \quad for \quad k = 0,1,2\cdots,n$$

In the node interval $[x_k, x_{k+1}]$, s is a polynomial of degree 1 passing through the points $(x_k, f(x_k))$ and $(x_{k+1}, f(x_{k+1}))$, thereby

$$s(x) = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k), \quad x \in [x_k, x_{k+1}]$$

simply

$$s_k(x) = a_k + b_k(x - x_k), \quad k = 0,1,2\cdots,n-1$$

where

$$a_k = f(x_k)$$

$$b_k = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

```
clear, clc

x=[0.0,0.2,0.3,0.5];
y=[0.0,0.18,0.26,0.41];

X=0.0:0.01:0.5;
Y=interp1(x,y,X);
Y1=interp1(x,y,0.1);
Y2=interp1(x,y,0.4);
%Y3=interp1(x,y,0.8);

plot(x,y,'r*',X,Y)
hold on
plot(0.1,Y1,'bo')
plot(0.4,Y2,'bo')
%plot(0.8,Y3,'bo')

hold off
```

2

Cubic Spline Interpolation (MM8)

At the nodes $a = x_0 < x_1 < x_2 \cdots < x_n = b$, the piecewise polynomial has the form

$$s_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 \quad k = 0,1,2,\cdots n-1$$

The following conditions should be satisfied

(i) $s_k(x_k) = f(x_k)$ and $s_k(x_{k+1}) = f(x_{k+1})$, for $k = 0,1,2\cdots, n-1$

(ii) $s_k'(x_{k+1}) = s_{k+1}'(x_{k+1})$ for $k = 0,1,2\cdots, n-2$

(ii) $s_k''(x_{k+1}) = s_{k+1}''(x_{k+1})$ for $k = 0,1,2\cdots, n-2$

There are 4n coefficients, but 4n-2 equations ---- 2 degrees of freedom!

The natural cubic spline is defined by imposing the additional conditions:
s''(a)=s''(b)=0

3

Cubic Spline Formula (MM8)

$$s_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 \quad k = 0,1,2,\cdots n-1$$

$$a_k = f(x_k) \quad \text{for} \quad k = 0,1,2\cdots, n-1$$

Denote $h_k = x_{k+1} - x_k$, there is $b_k + c_k h_k + d_k h_k^2 = \frac{f(x_{k+1}) - f(x_k)}{h_k} \triangleq \delta_k$ $\Leftrightarrow$ $b_k = \delta_k - c_k h_k - d_k h_k^2$

$2c_k + 6d_k h_k = 2c_{k+1}$ for $k = 0,1,2\cdots, n-2$ $\Leftrightarrow$ $d_k = \frac{c_{k+1} - c_k}{3h_k}$

$b_k + 2c_k h_k + 3d_k h_k^2 = b_{k+1}$ $\Leftrightarrow$ for $k = 0,1,2\cdots, n-2$ $\Leftrightarrow$ $c_k h_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1}c_{k+2} = 3(\delta_{k+1} - \delta_k)$

$$H\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = 3\begin{bmatrix} \delta_1 - \delta_0 \\ \delta_2 - \delta_1 \\ \vdots \\ \delta_{n-1} - \delta_{n-2} \end{bmatrix},$$

H is the tridiagonal matrix: $H = \begin{bmatrix} 2(h_0 + h_1) & h_1 & & \\ h_1 & 2(h_1 + h_2) & h_2 & \\ & h_2 & \ddots & h_{n-2} \\ & & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{bmatrix}$

4

Cubic Spline – Matlab Codes (MM8)

```
function s=cspline(nodes,data,x)

% computes coefficients for natural cubic spline
% at nodes with values data
% evaluate the spline at all points of the array x

N=length(nodes)-1;
P=length(x);

% compute the steplength
h=diff(nodes);
%compute first divided differences of data
D=diff(data)./h;
% compute of D for rhs of linear system
dD3=3*diff(D);

a=data(1:N)
```

```
clear clc
X=[25,36,49,64,81]; Y=[5,6,7,8,9];
x=[25:0.4:81];
y=cspline(X,Y,x);
plot(x,y)
```

```
% generate tridiagonal system
H=diag(2*(h(2:N)+h(1:N-1)));
for k=1:N-2
   H(k,k+1)=h(k+1);
   H(k+1,k)=h(k+1);
end
c=zeros(1,N+1);
c(2:N)=H\ dD3';

b=D-h.*(c(2:N+1)+2*c(1:N))/3;
d=(c(2:N+1)-c(1:N))./(3*h);

% evaluattion

for i=1:P
   k=1;
   while (x(i)>nodes(k+1)) & (k<N)
      k=k+1;
   end
   z=x(i)-nodes(k);
   s(i)=a(k)+z*(b(k)+ z*(c(k)+z*d(k)));
end
```

5

Matlab Built-in Functions (MM8)

- YI = INTERP1(X,Y,XI,METHOD)
  - 'nearest' - nearest neighbor interpolation
  - 'linear' - linear interpolation
  - 'spline' - piecewise cubic spline interpolation (SPLINE)
  - 'pchip' - shape-preserving piecewise cubic interpolation
  - 'cubic' - same as 'pchip'
  - 'v5cubic' - the cubic interpolation from MATLAB 5, which does not extrapolate and uses 'spline' if X is not equally spaced.
- PP = SPLINE(X,Y) provides the piecewise polynomial form of the cubic spline interpolant to the data values Y at the data sites X, for use with the evaluator PPVAL and the spline utility UNMKPP.

6

Differential Equations

- Ordinary Differential Equations (ODE)
- Dynamic systems
- Initial value problem
- Numerical solution to ODE

7

Euler's Method

- **Motivation:** Taylor expansion

$$y' = f(x, y), \quad y(x_0) = y_0$$
$$y(x_0 + h) \approx y(x_0) + y'(x_0)h = y(x_0) + f(x_0, y_0)h$$
$$\Downarrow$$
$$y_{k+1} \hat{=} y(x_0 + (k+1)h), \quad x_k \hat{=} x_0 + kh$$

$$y_{k+1} = y_k + hf(x_k, y_k), \quad k = 0,1,2,\cdots$$

- Error analysis

8

Euler's Method –Matlab Codes

```
function sol=euler1(f,a,b,y0,N)
% find the Euler's solution of y'=f(x,y)
%on an interval
% [a,b] with N steps, with initial value
% y(a)=y0

h=(b-a)/N;
x=a:h:b;
y=zeros(1,N+1);
y(1)=y0;
for k=1:N
    y(k+1)=y(k)+h*feval(f,x(k),y(k));
end
sol=[x',y']
```

```
function dy=example61(x,y)

dy=3*x.^2*y;

% true solution exp(x^3)
```

```
>> s=euler1('example61',0,1,1,4)
s =
         0    1.0000
    0.2500    1.0000
    0.5000    1.0469
    0.7500    1.2432
    1.0000    1.7676
>> exp(1)
ans =    2.7183
>> s=euler1('example61',0,1,1,128);
s(129,2)-exp(1)
ans =
   -0.0498
```

Runge-Kutta Methods

- **Motivation:**

  Use more terms of Taylor series (by adding extra points between x_k and x_k+1

- Second-order RK method - **RK2**
- Fourth-order RK method – **RK4**

10

Illustration of RK2 Method

$$y(x_0+h) \approx y(x_0) + hy'(x_0) + \frac{h^2}{2} y''(x_0)$$

$$\Downarrow$$

$$y(x_0+h) \approx y(x_0) + hf(x_0, y_0) + \frac{h^2}{2} y''(x_0)$$

denote $k_1 = f(x_0, y_0)$, and define a coefficient $\alpha \in [0,1]$. Consider

$$y(x_0+\alpha h) \approx y(x_0) + \alpha h k_1,$$

$$y''(x_0) = \frac{y'(x_0+\alpha h) - y'(x_0)}{\alpha h} = \frac{f(x_0+\alpha h, y(x_0+\alpha h)) - f(x_0, y_0)}{\alpha h}$$

$$\approx \frac{f(x_0+\alpha h, y_0+\alpha h k_1) - f(x_0, y_0)}{\alpha h} \triangleq \frac{k_2 - k_1}{\alpha h}$$

$$\Downarrow$$

$$y_1 = y_0 + h[k_1(1 - \frac{1}{2\alpha}) + \frac{k_2}{2\alpha}]$$

$$\Downarrow$$

$$y_{n+1} = y_n + h[k_1(1 - \frac{1}{2\alpha}) + \frac{k_2}{2\alpha}], \text{ where } k_1 = f(x_n, y_n), \quad k_2 = f(x_n + \alpha h, y_n + \alpha h k_1)$$

11

Classfications of RK2 Method

$$y(x_0+h) \approx y(x_0) + hy'(x_0) + \frac{h^2}{2} y''(x_0)$$

$$\Downarrow$$

Define a coefficient $\alpha \in [0,1]$.

$$y_{n+1} = y_n + h[k_1(1 - \frac{1}{2\alpha}) + \frac{k_2}{2\alpha}], \text{ where } k_1 = f(x_n, y_n), \quad k_2 = f(x_n + \alpha h, y_n + \alpha h k_1)$$

Classifica tions :

$\alpha = 1/2$, corrected Euler (middlepoi nt) method :

$$y_{n+1} = y_n + hk_2, \text{ where } k_2 = f(x_n + h/2, y_n + hk_1/2), k_1 = f(x_n, y_n)$$

$\alpha = 1$, modified Euler method :

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2), \text{ where } k_2 = f(x_n + h, y_n + hk_1), k_1 = f(x_n, y_n)$$

$\alpha = 2/3$, Heun' s method :

$$y_{n+1} = y_n + \frac{h}{4}(k_1 + 3k_2), \text{ where } k_2 = f(x_n + 2h/3, y_n + 2hk_1/3), k_1 = f(x_n, y_n)$$

12

Corrected Euler Method – Matlab codes

```
function [x,y]=coreuler(f,a,b,y0,N)

h=(b-a)/N;
x=a:h:b;
y=zeros(1,N+1);
y(1)=y0;
for n=1:N
    k=feval(f,x(n),y(n));
    y(n+1)=y(n)+h*feval(f,x(n)+h/2,y(n)+k*h/2);
end
x,y
```

```
>> s1=euler1('example61',0,1,1,128);
>> s1(129,2)-exp(1)

ans =

   -0.0498

>> s2=coreuler('example61',0,1,1,128);
>> s2(129,2)-exp(1)

ans =

 -3.2993e-004
```

13

RK4 Method

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

where

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

$$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{hk_1}{2})$$

$$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{hk_2}{2})$$

$$k_4 = f(x_n + h, y_n + hk_3)$$

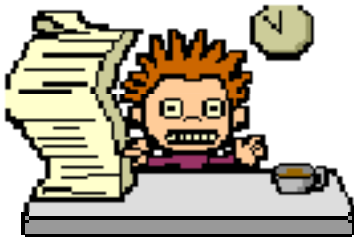
```
>> s3=RK4('example61',0,1,1,128);
>> s3(129,2)-exp(1)

ans =

 -1.8669e-009
```

```
function sol=RK4(f,a,b,y0,N)
% Generates RK4 solution to
%   y'=f(x,y)
% on[a,b], using N steps.
% Initial condition is
%   y(a)=y0.
h=(b-a)/N;
x=a:h:b;
y=zeros(1,N+1);
y(1)=y0;
for n=1:N
    k1=feval(f,x(n),y(n));
    k2=feval(f,x(n)+h/2,y(n)+h*k1/2);
    k3=feval(f,x(n)+h/2,y(n)+h*k2/2);
    k4=feval(f,x(n)+h,y(n)+h*k3);
    y(n+1)=y(n)+h*(k1+2*(k2+k3)+k4)/6;
end
sol=[x',y']
```

Exercises (MM8)



15

Question One (Exercise 6.1.1-4, pp.178):

Concern the differential equation

$$y' = x/y. \tag{1}$$

- Find the general solution of the differential equation;
- For the initial condition $y(0) = 3$, use the Euler's method with steps $h = 1,\ 1/2$ and $1/4$ to approximate $y(1)$;
- Use Euler's method to solve the initial-value problem $y0) = 3$ over $[0, 4]$ with $N = 10,\ 100$ and 200 steps;
- Tabulate the errors in the approximate values of $y(4)$.

Question Two (Exercise 6.2.1, pp.185):

Continue to concern the differential equation (1). For the initial condition $y(0) = 3$, use the corrected Euler's method with steps $h = 1,\ 1/2$ and $1/4$ to approximate $y(1)$.

Question Three (Exercise 6.2.7, pp.185-186):

Use the classical Runge-Kutta RK4 method with steplengths $h = 10^{-k}$ for $k = 1, 2, 3$ to solve the initial-value problem

$$y' = x + y^2$$

with $y(0) = 0$ on $[0, 1]$. Tabulate the results for $x = 0, 0.1, 0.2, \cdots, 1$ and graph the solutions.

16