

# MATLAB Notes and Codes

Liutao Tian

*Version: 1.2*

*October 10, 2019*

## Contents

<b>1</b>	<b>Numerical Methods for ODEs</b>	<b>2</b>
1.1	Euler Method . . . . .	2
1.2	Modified Euler Method . . . . .	3
1.3	Runge-Kutta Method . . . . .	4
<b>2</b>	<b>Finite Difference Method</b>	<b>8</b>
2.1	Finite Difference Methods for 1-D Problem . . . . .	8
2.2	Finite Difference Methods for 2-D Problem . . . . .	10
<b>3</b>	<b>Finite Element Methods</b>	<b>16</b>
3.1	Galerkin Method for 1-D Problem . . . . .	16
<b>4</b>	<b>Spectral Methods</b>	<b>22</b>
4.1	Legendre-Galerkin Spectral Methods . . . . .	22
4.2	Collocation Methods . . . . .	28

# 1 Numerical Methods for ODEs

Consider the initial value problem of ordinary differential equation,  $f(t, u)$  is continuous on area  $G : 0 \leq t \leq T, |u| < \infty$ ,  $u = u(t)$  satisfy the equation

$$\begin{cases} \frac{du}{dt} = f(t, u), & 0 < t \leq T, \\ u(0) = 0. \end{cases} \quad (1.1)$$

generally,  $f$  satisfy Lipschitz condition:  $|f(t, u_1) - f(t, u_2)| \leq L|u_1 - u_2|$ .

## 1.1 Euler Method

Euler method scheme:

$$u_{n+1} = u_n + hf(t_n, u_n)$$

### Example 1.1

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & t \in [0, 1] \\ u(0) = 0. \end{cases} \quad (1.2)$$

```
1 % Euler1.m
2 % Euler method for the ODE model
3 % u'(t)=t^2+t-u, t \in [0,1]
4 % Initial condition : u(0)=0 ;
5 % Exact solution : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1; % function interval
9 n=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     u(i+1)=u(i)+h.*fun(x(i),u(i));
14 end
15 ue=-exp(-x)+x.^2-x+1; % exact solution
16 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
17 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
18 legend('Exact','Numerical','location','North')
19 title('Euler Method','fontsize',14)
20 set(gca,'fontsize',14)
```

## 1.2 Modified Euler Method

Modified Euler method scheme:

$$u_{n+1} = u_n + \frac{h}{2}[f(t_n, u_n) + f(t_{n+1}, u_{n+1})]$$

### Example 1.2

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & t \in [0, 1] \\ u(0) = 0. \end{cases} \quad (1.3)$$

```
1 % Euler2.m
2 % Modified Euler method for the ODE model
3 % u'=t^2+t-u,  t \in [0,1]
4 % Initial condition : u(0)=0
5 % Exact solution : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1;           % function interval
9 n=length(x)-1;
10 u(1)=0;           % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     k1=fun(x(i),u(i));
14     k2=fun(x(i+1),u(i)+h*k1);
15     u(i+1)=u(i)+(h/2)*(k1+k2);
16 end
17 ue=-exp(-x)+x.^2-x+1; % exact solution
18 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
19 xlabel('x','fontsize', 16), ylabel('y','fontsize',16,'Rotation',0)
20 legend('Exact','Numerical','location','North')
21 title('Modified Euler Method','fontsize',14)
22 set(gca,'fontsize',14)
```

## 1.3 Runge-Kutta Method

Runge-Kutta method scheme:

$$\begin{cases} u_{n+1} = u_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(t_n, u_n) \\ k_2 = f\left(t_n + \frac{h}{2}, u_n + \frac{1}{2}k_1\right) \\ k_3 = f\left(t_n + \frac{h}{2}, u_n + \frac{1}{2}k_2\right) \\ k_4 = f(t_n + h, u_n + k_3) \end{cases}$$

### Example 1.3

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & t \in [0, 1] \\ u(0) = 0. \end{cases} \quad (1.4)$$

```
1 % RungeKutta.m
2 % Runge-Kutta method for the ODE model
3 % u'=t^2+t-u, t \in [0,1]
4 % Initial condition : u(0)=0
5 % Exact : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1; % function interval
9 n=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     k1=fun(x(i),u(i));
14     k2=fun(x(i)+h./2,u(i)+h.*k1/2);
15     k3=fun(x(i)+h./2,u(i)+h.*k2/2);
16     k4=fun(x(i)+h,u(i)+h.*k3);
17     u(i+1)=u(i)+h.*(k1+2.*k2+2.*k3+k4)./6;
18 end
19 ue=-exp(-x)+x.^2-x+1; % exact solution
20 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
21 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
22 legend('Exact','Numerical','location','North')
23 title('Runge-Kutta Method','fontsize',14)
24 set(gca,'fontsize',14)
```

The general s-stage Runge-Kutta method for the problem

$$y' = f(x, y), \quad y(a) = \eta, \quad f : \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}^m$$

is defined by

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i \\ k_i = f(x_n + c_i h, y_n + h \sum_{j=1}^s a_{ij} k_j), \quad i = 1, 2, \dots, s \end{cases} \quad (1.5)$$

Assume that the following (the row-sum condition) holds

$$c_i = \sum_{j=1}^s a_{ij}, \quad i = 1, 2, \dots, s \quad (1.6)$$

It is convenient to display the coefficients as a Butcher array:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

$$c = [c_1, c_2, \dots, c_s]^T, \quad b = [b_1, b_2, \dots, b_s]^T, \quad A = (a_{ij})_s$$

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=1}^s b_i f(x_n + c_i h, Y_i) \\ Y_i = y_n + h \sum_{j=1}^s a_{ij} f(x_n + c_j h, Y_j), \quad i = 1, 2, \dots, s \end{cases} \quad (1.7)$$

The forms (1.5) and (1.7) are seen to be equivalent if we make the interpretation

$$k_i = f(x_n + c_i h, Y_i), \quad i = 1, 2, \dots, s$$

Implicit Runge-Kutta method (Gauss method) 2 stage order 4:

$$\begin{cases} y_{n+1} = y_n + \frac{1}{2} (K_1 + K_2), & n = 0, 1, \dots, N-1, \\ K_1 = hf \left( t_n + \left( \frac{1}{2} - \frac{\sqrt{3}}{6} \right) h, y_n + \frac{1}{4} K_1 + \left( \frac{1}{4} - \frac{\sqrt{3}}{6} \right) K_2 \right), \\ K_2 = hf \left( t_n + \left( \frac{1}{2} + \frac{\sqrt{3}}{6} \right) h, y_n + \left( \frac{1}{4} + \frac{\sqrt{3}}{6} \right) K_1 + \frac{1}{4} K_2 \right). \end{cases} \quad (1.8)$$

Butcher array

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=1}^2 b_i f(x_n + c_i h, Y_i), \\ Y_1 = y_n + h \sum_{j=1}^2 a_{1j} f(x_n + c_j h, Y_j), \\ Y_2 = y_n + h \sum_{j=1}^2 a_{2j} f(x_n + c_j h, Y_j), \end{cases} \quad (1.9)$$

**Example 1.4**

$$\begin{cases} \frac{du}{dt} = u, & t \in [0, 1] \\ u(0) = 1. \end{cases} \quad (1.10)$$

```

1 % IRK2s_error.m
2 % Implicit Runge-Kutta(Gauss method) 2 stage and order 4
3 % u'=u in [0,1] with initial condition u(0)=1
4 % exact solution: ue=exp(x)
5 clear all
6 Nvec=[10 50 100 200 500 1000];
7 Err=[];
8 for n=1:length(Nvec)
9     N=Nvec(n); h=1/N;
10    x=[0:h:1];
11    u(1)=1;
12    X0=[1;1];
13    % Newton iteration
14    for i=1:N
15        k=u(i);
16        r=X0; tol=1;
17        while tol>1.0e-6
18            X=r;
19            D=[1-0.25*h, -h*(0.25-(sqrt(3))/6);...
20              -h*(0.25+(sqrt(3))/6), 1-h*0.25]; % Jacobian matrix
21            F=[X(1)-k-h*(0.25*X(1)+(0.25-(sqrt(3))/6)*X(2));...
22              X(2)-k-h*((0.25+(sqrt(3))/6)*X(1)+0.25*X(2))]; % RHS
23            r=X-D\F;
24            tol=norm(r-X);
25        end
26        k1=r(1); k2=r(2);
27        u(i+1)=k+(h/2)*(k1+k2);
28    end
29    ue=exp(x); % exact solution
30    err=max(abs(u-ue)); % maximum error
31    Err=[Err,err];
32 end
33 plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
34 hold on,
35 plot(log10(Nvec), log10(Nvec.^(-4)), '--')
36 grid on,
37 xlabel('log_{10}N','fontsize',16), ylabel('log_{10}Error','fontsize',16)
38 title('Convergence order of Gauss method ','fontsize',14)
39 set(gca,'fontsize',14)
40 for i=1:length(Nvec)-1 % computing convergence order
41     order(i)=-log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
42 end
43 Err
44 order

```

## 2 Finite Difference Method

### 2.1 Finite Difference Methods for 1-D Problem

Consider the two-point boundary value problem (constant coefficient):

$$-\frac{d^2u}{dx^2} + \frac{du}{dx} + u = f(x), \quad x \in [a, b] \quad (2.1)$$

Discrete difference scheme:

$$-\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1}))}{h^2} + \frac{u(x_{i+1}) - u(x_{i-1}))}{h} + u(x_i) = f(x_i), i = 1, 2, \dots, N-1. \quad (2.2)$$

#### Example 2.1

$$\begin{cases} -\frac{d^2u}{dx^2} + \frac{du}{dx} = \pi^2 \sin(\pi x) + \pi \cos(\pi x), & x \in [0, 1] \\ u(0) = 0, u(1) = 0. \end{cases} \quad (2.3)$$

Exact solution:  $u(x) = \sin(\pi x)$ .

```
1 % fdm1d1.m
2 % finite difference method for 1D problem
3 % -u''+u'=pi^2*sin(pi*x)+pi*cos(pi*x) in [0,1]
4 % u(0)=0, u(1)=0 ;
5 % exact solution : u=sin(pi*x)
6 clear all
7 h=0.05;
8 x=0:h:1;
9 N=length(x)-1;
10 A=diag((2/h^2)*ones(N-1,1))...
11     +diag((1/(2*h)-1/h^2)*ones(N-2,1),1)...
12     +diag((-1/(2*h)-1/h^2)*ones(N-2,1),-1);
13 b=pi^2*sin(pi*x(2:N))+pi*cos(pi*x(2:N));
14 u=A\b';
15 u=[0;u;0];
16 ue=sin(pi*x)';
17 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
18 Error=max(abs(u-ue))
19 xlabel('x','fontsize', 16), ylabel('y','fontsize',16,'Rotation',0)
20 legend('Exact ', 'Numerical', 'location', 'North')
21 title('Finite Difference Method','fontsize',14)
22 set(gca,'fontsize',14)
```



Consider the two-point boundary value problem (variable coefficient):

$$-\frac{d}{dx}\left(p\frac{du}{dx}\right) + r\frac{du}{dx} + qu = f(x), \quad x \in (a, b) \quad (2.4)$$

Discrete difference scheme:

$$-\frac{2}{h_i + h_{i+1}} \left[ p_{i+\frac{1}{2}} \frac{u(x_{i+1}) - u(x_i)}{h_{i+1}} + p_{i-\frac{1}{2}} \frac{u(x_i) - u(x_{i-1}))}{h_i} \right] + \frac{r_i}{h_i + h_{i+1}} (u(x_{i+1}) - u(x_{i-1})) + q_i u(x_i) = f(x_i), i = 1, \dots, N-1. \quad (2.5)$$

### Example 2.2

$$\begin{cases} -\frac{d}{dx} \left( x \frac{du}{dx} \right) + x \frac{du}{dx} = \pi^2 x \sin(\pi x) + \pi(x-1) \cos(\pi x), x \in (0, 1) \\ u(0) = 0, u(1) = 0. \end{cases} \quad (2.6)$$

Exact solution:  $u(x) = \sin(\pi x)$ .

```

1 % fdm1d2.m
2 % finite difference method for 1D problem
3 % -(xu')'+x*u'=pi^2*x*sin(pi*x)-pi*cos(pi*x)+pi*x*cos(pi*x) in [0,1]
4 % u(0)=0, u(1)=0 ;
5 % exact solution : u=sin(pi*x)
6 clear all
7 h=0.05;
8 x=0:h:1;
9 N=length(x)-1;
10 A=diag(2*x(2:N)./h^2)+diag(x(2:N-1)./(2*h)-(x(2:N-1)+0.5*h)./h^2,1)...
11     +diag(-x(3:N)./(2*h)-(x(3:N)-0.5*h)./h^2,-1);
12 b=pi^2*x(2:N).*sin(pi*x(2:N))+pi*(x(2:N)-1).*cos(pi*x(2:N));
13 u=A\b';
14 u=[0;u;0];
15 ue=sin(pi*x');
16 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
17 Error=max(abs(u-ue))
18 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
19 legend('Exact','Numerical','location','North')
20 title('Finite Difference Method','fontsize',14)
21 set(gca,'fontsize',14)

```

## 2.2 Finite Difference Methods for 2-D Problem

Consider the two-dimensional Poisson problem:

$$\begin{cases} -\Delta u = f(x, y), & (x, y) \in \Omega, \\ u|_{\partial\Omega} = \phi(x, y), & (x, y) \in \partial\Omega. \end{cases} \quad (2.7)$$

Discrete difference scheme:

$$-\frac{1}{h_2^2}u_{i,j-1} - \frac{1}{h_1^2}u_{i-1,j} + 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right)u_{i,j} - \frac{1}{h_1^2}u_{i+1,j} - \frac{1}{h_2^2}u_{i,j+1} = f(x_i, y_j), \quad (2.8)$$

$$1 \leq i \leq N-1, \quad 1 \leq j \leq M-1.$$

Define the vector:  $\mathbf{u}_j = (u_{1j}, u_{2j}, \dots, u_{N-1,j})^T$ ,  $0 \leq j \leq M$ .

The discrete scheme to matrix form:

$$\mathbf{D}\mathbf{u}_{j-1} + \mathbf{C}\mathbf{u}_j + \mathbf{D}\mathbf{u}_{j+1} = \mathbf{f}_j, \quad 1 \leq j \leq M-1.$$

$$\mathbf{C} = \begin{pmatrix} 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} & & & \\ -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} \\ & & & -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} -\frac{1}{h_2^2} & & & & \\ & -\frac{1}{h_2^2} & & & \\ & & \ddots & & \\ & & & -\frac{1}{h_2^2} & \\ & & & & -\frac{1}{h_2^2} \end{pmatrix} \quad \mathbf{f}_j = \begin{pmatrix} f(x_1, y_j) + \frac{1}{h_1^2}\phi(x_0, y_j) \\ f(x_2, y_j) \\ \vdots \\ f(x_{N-2}, y_j) \\ f(x_{N-1}, y_j) + \frac{1}{h_1^2}\phi(x_N, y_j) \end{pmatrix}$$

Next, above can be written in the following matrix form

$$\begin{pmatrix} \mathbf{C} & \mathbf{D} & & & \\ \mathbf{D} & \mathbf{C} & \mathbf{D} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{D} & \mathbf{C} & \mathbf{D} \\ & & & \mathbf{D} & \mathbf{C} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{M-2} \\ \mathbf{u}_{M-1} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 - \mathbf{D}\mathbf{u}_0 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{M-2} \\ \mathbf{f}_{M-1} - \mathbf{D}\mathbf{u}_N \end{pmatrix}$$

### Example 2.3

$$\begin{cases} -\Delta u = f(x, y), & (x, y) \in \Omega = (0, 1) \times (0, 1) \\ u = 0, & (x, y) \in \partial\Omega. \end{cases}$$

where  $f(x, y) = -2\pi^2 e^{\pi(x+y)} (\sin \pi x \cos \pi y + \cos \pi x \sin \pi y)$ .

Exact solution:  $u(x, y) = e^{\pi(x+y)} \sin \pi x \sin \pi y, \quad (x, y) \in \Omega = (0, 1) \times (0, 1)$ .

```
1 % fdm2d1.m
2 % finite difference method for 2D problem
3 % -d^2u/dx^2-d^2u/dy^2=f(x,y)
4 % f(x,y)=-2*pi^2*exp(pi*(x+y))*(sin(pi*x)*cos(pi*y)+cos(pi*x)*sin(pi*y))
5 % exact solution: ue=exp(pi*x+pi*y)*sin(pi*x)*sin(pi*y)
6 clear all
7 h=0.01;
8 x=[0:h:1]';
9 y=[0:h:1]';
10 N=length(x)-1;
11 M=length(y)-1;
12 [X,Y]=meshgrid(x,y);
13 X=X(2:M,2:N);
14 Y=Y(2:M,2:N);
15 % generate the matrix of RHS
16 f=-2*pi^2*exp(pi*X+pi*Y).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
17 % constructing the coefficient matrix
18 C=4/h^2*eye(N-1)-1/h^2*diag(ones(N-2,1),1)-1/h^2*diag(ones(N-2,1),-1);
19 D=-1/h^2*eye(M-1);
20 A=kron(eye(M-1),C)+kron(diag(ones(M-2,1),1)+diag(ones(M-2,1),-1),D);
21 % solving the linear system
22 f=f';
23 u=zeros(M+1,N+1);
24 u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
25 u(:,1)=0;
26 u(:,end)=0;
27 ue=zeros(M+1,N+1);
28 ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
29 % compute maximum error
30 Error=max(max(abs(u-ue)))
31 mesh(x,y,u)
32 xlabel('x','fontsize',16), ylabel('y','fontsize',16), ...
    zlabel('u','fontsize',16,'Rotation',0)
33 title('Finite Difference Method','fontsize',14)
34 set(gca,'fontsize',14)
```

```

1 % fdm2d1_error.m
2 % finite difference method for 2D problem
3 %  $-d^2u/dx^2-d^2u/dy^2=f(x,y)$ 
4 %  $f(x,y)=-2\pi^2\exp(\pi(x+y))\cdot(\sin(\pi x)\cos(\pi y)+\cos(\pi x)\sin(\pi y))$ 
5 % exact solution:  $ue=\exp(\pi x+\pi y)\sin(\pi x)\sin(\pi y)$ 
6 clear all
7 Nvec=2.^[3:10]; Err=[];
8 for n=Nvec
9     h=1/n;
10    x=[0:h:1]'; y=[0:h:1]';
11    N=length(x)-1; M=length(y)-1;
12    [X,Y]=meshgrid(x,y);
13    X=X(2:M,2:N);
14    Y=Y(2:M,2:N);
15    % generate the matrix of RHS
16    f=-2*pi^2*exp(pi*(X+Y)).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
17    % constructing the coefficient matrix
18    e=ones(N-1,1);
19    C=1/h^2*spdiags([-e 4*e -e],[-1 0 1],N-1,N-1);
20    D=-1/h^2*eye(N-1);
21    e=ones(M-1,1);
22    A=kron(eye(M-1),C)+kron(spdiags([e e],[-1 1],M-1,M-1),D);
23    % solving the linear system
24    f=f';
25    u=zeros(M+1,N+1);
26    u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
27    u(:,1)=0;
28    u(:,end)=0;
29    ue=zeros(M+1,N+1); % numerical solution
30    ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
31    err=max(max(abs(u-ue))); % maximum error
32    Err=[Err,err];
33 end
34 plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
35 grid on,hold on, plot(log10(Nvec), log10(Nvec.^(-2)), '--')
36 xlabel('log_{10}N','fontsize',16), ylabel('log_{10}Error','fontsize',16),
37 title('Convergence of Finite Difference Method','fontsize',14)
38 set(gca,'fontsize',14)
39 for i=1:length(Nvec)-1 % computing convergence order
40     order(i)=-log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
41 end
42 Err
43 order

```

### Example 2.4

$$\begin{cases} -\Delta u = \cos 3x \sin \pi y, & (x, y) \in G = (0, \pi) \times (0, 1), \\ u(x, 0) = u(x, 1) = 0, & 0 \leq x \leq \pi, \\ u_x(0, y) = u_x(\pi, y) = 0, & 0 \leq y \leq 1. \end{cases} \quad (2.9)$$

Exact solution:  $u = (9 + \pi^2)^{-1} \cos 3x \sin \pi y$ .

Rectangular division:  $h_1 = \frac{\pi}{N}$ ,  $h_2 = \frac{1}{N}$ , grid node:  $x_i = ih_1$ ,  $y_j = jh_2$ ,  $i, j = 0, 1, \dots, N$ .

Discrete difference scheme:

$$-\left( \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h_1^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h_2^2} \right) = \cos 3x_i \sin \pi y_j, \quad (2.10)$$

$$i, j = 1, 2, \dots, N-1.$$

Boundary conditions:

$$\begin{aligned} u_{i0} &= u_{iN} = 0, i = 0, \dots, N \\ u_{0j} &= u_{1j}, j = 1, \dots, N-1 \\ u_{Nj} &= u_{N-1,j}, j = 1, \dots, N-1 \end{aligned}$$

Discrete scheme:

$$\mathbf{D}u_{j-1} + \mathbf{C}u_j + \mathbf{D}u_{j+1} = \mathbf{f}_j, \quad 1 \leq j \leq M-1.$$

$$\mathbf{C} = \begin{pmatrix} \left(\frac{1}{h_1^2} + \frac{2}{h_2^2}\right) & -\frac{1}{h_1^2} & & & \\ -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h_1^2} & 2\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) & -\frac{1}{h_1^2} \\ & & & -\frac{1}{h_1^2} & \left(\frac{1}{h_1^2} + \frac{2}{h_2^2}\right) \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} -\frac{1}{h_2^2} & & & & \\ & -\frac{1}{h_2^2} & & & \\ & & \ddots & & \\ & & & -\frac{1}{h_2^2} & \\ & & & & -\frac{1}{h_2^2} \end{pmatrix} \quad \mathbf{f}_j = \begin{pmatrix} f(x_1, y_j) \\ f(x_2, y_j) \\ \vdots \\ f(x_{N-2}, y_j) \\ f(x_{N-1}, y_j) \end{pmatrix}$$

Matrix form:

$$\begin{pmatrix} \boldsymbol{C} & \boldsymbol{D} & & \\ \boldsymbol{D} & \boldsymbol{C} & \boldsymbol{D} & \\ & \ddots & \ddots & \ddots \\ & & \boldsymbol{D} & \boldsymbol{C} & \boldsymbol{D} \\ & & & \boldsymbol{D} & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{M-2} \\ \boldsymbol{u}_{M-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_{M-2} \\ \boldsymbol{f}_{M-1} \end{pmatrix}$$

```

1 % fdm2d2_error.m
2 % finite difference method for 2D problem
3 %  $-\Delta u = \cos(3x) \sin(\pi y)$  in  $(0, \pi) \times (0, 1)$ 
4 %  $u(x, 0) = u(x, 1) = 0$  in  $[0, \pi]$ 
5 %  $u_x(0, y) = u_x(\pi, y) = 0$  in  $[0, 1]$ 
6 % exact solution:  $u_e = (9 + \pi^2)^{-1} \cos(3x) \sin(\pi y)$ 
7 clear all; close all;
8 Nvec=2.^[2:7]; Err=[];
9 for N=Nvec
10     h1=pi/N; h2=1/N;
11     x=[0:h1:pi]'; y=[0:h2:1]';
12     [X,Y]=meshgrid(x,y);
13     X1=X(2:N,2:N); Y1=Y(2:N,2:N);
14     % generate the matrix of RHS
15     f=cos(3*X1).*sin(pi*Y1);
16     % constructing the coefficient matrix
17     e=ones(N-1,1);
18     C=diag([1/h1^2+2/h2^2, (2/h1^2+2/h2^2)*ones(1,N-3), 1/h1^2+2/h2^2])...
19         -1/h1^2*diag(ones(N-2,1),1)-1/h1^2*diag(ones(N-2,1),-1);
20     D=-1/h2^2*eye(N-1);
21     A=kron(eye(N-1),C)+kron(diag(ones(N-2,1),1)+diag(ones(N-2,1),-1),D);
22     %A=kron(eye(N-1),C)+kron(spdia([e e],[-1 1],N-1,N-1),D);
23     % solving the linear system
24     f=f';
25     u=zeros(N+1,N+1);
26     u(2:N,2:N)=reshape(A\f(:),N-1,N-1)';
27     % Neumann boundary condition
28     u(:,1)=u(:,2);
29     u(:,end)=u(:,end-1);
30     ue=1/(9+pi^2)*(cos(3*X)).*(sin(pi*Y));
31     err=max(max(abs(u-ue))); % maximum error
32     Err=[Err,err];
33 end
34 plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
35 grid on,hold on, plot(log10(Nvec),log10(Nvec.^(-1)),'--')
36 xlabel('log_{10}N','fontsize',16),ylabel('log_{10}Error','fontsize',16),
37 title('Convergence of Finite Difference Method','fontsize',14)
38 set(gca,'fontsize',14)
39 for i=1:length(Nvec)-1 % computing convergence order
40     order(i)=log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
41 end
42 order

```

## 3 Finite Element Methods

### 3.1 Galerkin Method for 1-D Problem

Consider the two-point boundary value problem:

$$\begin{cases} -u''(x) + \mu u(x) = f(x), & x \in I = (a, b) \\ u(a) = 0, u'(b) = 0. \end{cases} \quad (3.1)$$

Set

$$\begin{aligned} V &\triangleq \left\{ v | v, v \in L^2(a, b), \int_a^b (v^2 + v'^2) dx < +\infty, v(0) = 0 \right\}, \\ a(u, v) &= \int_a^b u'v' dx + \mu \int_a^b uv dx, \\ \langle f, v \rangle &= \int_a^b f v dx. \end{aligned}$$

The variational problem to find  $u \in V$  such that

$$a(u, v) = \langle f, v \rangle \quad \forall v \in V, \quad (3.2)$$

Let  $V_h$  be a subspace of  $V$  which is finite dimensional,  $h$  stands for a discretization parameter. The Galerkin method of the variation problem is then to find  $u_h \in V_h$  such that

$$a(u_h, v_h) = \langle f, v_h \rangle \quad \forall v \in V_h. \quad (3.3)$$

Suppose that  $\{\phi_1, \dots, \phi_N\}$  is a basis for  $V_h$ , Then (3.3) is equivalent to

$$a(u_h, \phi_i) = \langle f, \phi_i \rangle, \quad i = 1, \dots, N. \quad (3.4)$$

Writing  $u_h$  in the form

$$u_h = \sum_{j=1}^N u_j \phi_j, \quad (3.5)$$

we are led to the system of equations

$$\sum_{j=1}^N a(\phi_j, \phi_i) u_j = \langle f, \phi_i \rangle, \quad i = 1, \dots, N, \quad (3.6)$$

which we can write in the matrix-vector form as

$$A\mathbf{u} = \mathbf{b} \quad (3.7)$$

where  $A_{ij} = a(\phi_j, \phi_i)$ , and  $b_i = \langle f, \phi_i \rangle$ .



$$A\mathbf{u} \triangleq \begin{pmatrix} a(\phi_1, \phi_1) & a(\phi_2, \phi_1) & \cdots & a(\phi_n, \phi_1) \\ a(\phi_1, \phi_2) & a(\phi_2, \phi_2) & \cdots & a(\phi_n, \phi_2) \\ \vdots & \vdots & \vdots & \vdots \\ a(\phi_1, \phi_n) & a(\phi_2, \phi_n) & \cdots & a(\phi_n, \phi_n) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

$$\mathbf{b} \triangleq \begin{pmatrix} (f, \phi_1) \\ (f, \phi_2) \\ \vdots \\ (f, \phi_n) \end{pmatrix}$$

Mesh splitting, the nodes:  $a = x_0 < x_1 < \cdots < x_n = b$

Element:  $I_i = [x_{i-1}, x_i]$ ,  $h_i = x_i - x_{i-1}$ ,  $h = \max_i h_i$

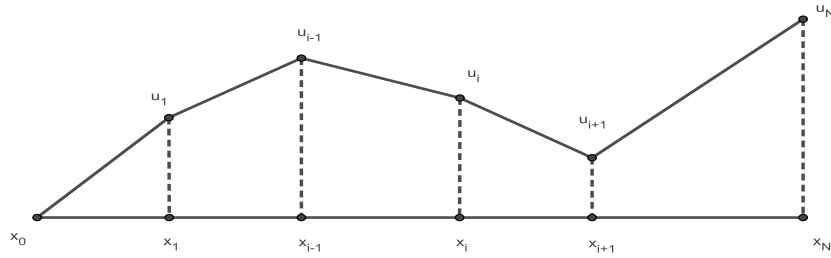
The test function space  $U_h$  is composed of piecewise linear functions. Its set of values on the node

$$u_0, u_1, u_2, \dots, u_n,$$

Linear interpolation formula

$$u_h(x) = \frac{x_i - x}{h_i} u_{i-1} + \frac{x - x_{i-1}}{h_i} u_i, x \in I_i, i = 1, 2, \dots, n. \quad (3.8)$$

Element shape function



Affine transform

$$\xi = \frac{x - x_{i-1}}{h_i},$$

Change  $I_i$  to the reference unit  $[-1, 1]$ ,

$$N_{-1}(\xi) = \frac{1 - \xi}{2}, \quad N_1(\xi) = \frac{1 + \xi}{2}$$

$$\Rightarrow u_h(x) = N_{-1}(\xi)u_{i-1} + N_1(\xi)u_i, \quad x \in I_i$$

Every local element have two element shape function:

$$\Phi_1^{I_i}(x) = \begin{cases} \frac{x_i - x}{h_i}, & x \in [x_{i-1}, x_i]; \\ 0, & \text{otherwise.} \end{cases}$$

$$\Phi_2^{I_i}(x) = \begin{cases} \frac{x - x_{i-1}}{h_i}, & x \in [x_{i-1}, x_i]; \\ 0, & \text{otherwise.} \end{cases}$$

Basis function

$$\varphi_1 = \frac{1}{2}(\Phi_2^{I_1} + \Phi_1^{I_2}), \quad \varphi_2 = \frac{1}{2}(\Phi_2^{I_2} + \Phi_1^{I_3}), \quad \dots$$

$$\varphi_i = \frac{1}{2}(\Phi_2^{I_i} + \Phi_1^{I_{i+1}}), \quad \dots \quad \varphi_n = \Phi_2^{I_n}.$$

In local unit  $I_i$ , element stiffness matrix  $K_{2 \times 2}^{I_i}$ .

$$K_{11}^{I_i} = a(\Phi_1^{I_i}, \Phi_1^{I_i}) = \int_{x_{i-1}}^{x_i} (p\Phi_1^{I_i'} \cdot \Phi_1^{I_i'} + q\Phi_1^{I_i} \cdot \Phi_1^{I_i}) dx$$

$$K_{22}^{I_i} = a(\Phi_2^{I_i}, \Phi_2^{I_i})$$

$$K_{12}^{I_i} = a(\Phi_2^{I_i}, \Phi_1^{I_i})$$

$$K_{21}^{I_i} = a(\Phi_1^{I_i}, \Phi_2^{I_i})$$

Global element of stiffness matrix  $A$  consist of

$$K_{ij} = \sum_{k=1}^n K_{ij}^{I_k}$$

**Example 3.1** Consider the two-point boundary value problem:

$$\begin{cases} -u''(x) + \alpha u(x) = f(x), & x \in I = (-1, 1) \\ u(-1) = 0, u(1) = 0. \end{cases} \quad (3.9)$$

Exact solution:  $u = x(1 - x) \sin(x)$ ,  $f = (4x - 2) \cos(x) + (2 + 2x - 2x^2) \sin(x)$ .

```

1 % FEM1D.m
2 % Finite Element Method
3 % -u_xx+u=f in (0,1) with boundary condition u(0)=u(1)=0;
4 % exact : u=x*(1-x)*sin(x)
5 % RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x);
6 % Thanks to the code from Shuangshuang Li & Qian Tong
7 clear all
8 Num=[16 32 64 128 256 512]; % Number of splits
9 Err=[]; DOF=[];
10 for j=1:length(Num)
11     N=Num(j); h=1/N; x=0:h:1;
12     % The global node number corresponds to element local node number
13     M=[1:N;2:N+1];
14     [xv,wv]=jags(2,0,0); % nodes and weights of gauss quadrature
15
16     K=zeros(N+1); % global stiffness matrix
17     F=zeros(N+1,1); % RHS load vector
18     for i=1:N % loop for each element
19         K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
20             +((h/2)*(((1/4)*(2/h)^2+((1-xv)/2).^2))'*wv;
21         K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
22             +((1-xv)/2).*((1+xv)/2))'*wv;
23         K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
24             +((1-xv)/2).*((1+xv)/2))'*wv;
25         K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2
26             +((1+xv)/2).^2))'*wv;
27
28         t=h*xv/2+(x(i+1)+x(i))/2;
29         F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
30             +(2+2*t-2*t.^2).*sin(t))'*wv;
31         F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
32             +(2+2*t-2*t.^2).*sin(t))'*wv;
33     end
34     % Dirichlet boundary condition
35     K(1,:)=zeros(1,N+1);
36     K(:,1)=zeros(1,N+1);
37     K(N+1,:)=zeros(1,N+1);
38     K(:,N+1)=zeros(1,N+1);
39     K(1,1)=1; K(N+1,N+1)=1;
40     F(1)=0; F(N+1)=0;
41
42     U=K\F; % numerical solution at the value of the node
43     error=max(abs(U'-x.*(1-x).*sin(x))); % node error
44     doff=N+1; % degrees of freedom, number of unknowns
45     Err=[Err, error];

```

```

46     DOF=[DOF, doff];
47 end
48 plot(log10(DOF),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5),
49 hold on,
50 plot(log10(DOF),log10(DOF.^(-2)),'--')
51 grid on,
52 xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
53 title('Convergence of Finite Element Method','fontsize',14)
54 set(gca,'fontsize',14)

```

```

1 % FEM1DP.m
2 % FEM for 1D elliptic problem
3 % -u_{xx}+u=f in [0,1] with boundary condition u(0)=u(1)=0;
4 % exact solution: u=x*(1-x)*sin(x);
5 % RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x)
6 % Thanks to the code from Shuangshuang Li & Qian Tong
7 clear all
8 Num=[16 32 64 128 256 512]
9 node_Err=[]; L2_Err=[]; H1_Err=[]; DOF=[];
10 for j=1:length(Num)
11     N=Num(j); h=1/N; x=0:h:1;
12     % The global node number corresponds to element local node number
13     M=[1:N;2:N+1];
14     [xv,wv]=jags(3,0,0); % nodes and weights of gauss quadrature
15     K=zeros(N+1); % global stiffness matrix
16     F=zeros(N+1,1); % RHS load vector
17
18     for i=1:N % loop for each element
19         K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
20             +((h/2)*((1/4)*(2/h)^2+((1-xv)/2).^2))*wv;
21         K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
22             +((1-xv)/2).*((1+xv)/2)))*wv;
23         K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
24             +((1-xv)/2).*((1+xv)/2)))*wv;
25         K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*((1/4)*(2/h)^2
26             +((1+xv)/2).^2))*wv;
27
28         t=h*xv/2+(x(i+1)+x(i))/2;
29         F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
30             +(2+2*t-2*t.^2).*sin(t)))*wv;
31         F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
32             +(2+2*t-2*t.^2).*sin(t)))*wv;
33     end
34 % Handling Dirichlet boundary condition

```

```

35     K(1,: )=zeros(1,N+1);
36     K(:,1)=zeros(1,N+1);
37     K(N+1,: )=zeros(1,N+1);
38     K(:, N+1)=zeros(1,N+1);
39     K(1,1)=1;     K(N+1,N+1)=1;
40     F(1)=0;       F(N+1)=0;
41
42     U=K\F;         % numerical solution at the value of the nodes
43     node_error=max(abs(U'-x.*(1-x).*sin(x))); % node error
44     for i=1:N
45         tt=h*xv/2+(x(i+1)+x(i))/2;
46         % value of finite element solution at Gauss point
47         uh=U(i)*(1-xv)/2+U(i+1)*(1+xv)/2;
48         % derivative value of finite element solution at Gauss point
49         duh=-U(i)/2+U(i+1)/2;
50         L2_error(i)=h/2*((tt.*(1-tt).*sin(tt)-uh).^2)*wv;
51         % the square of the L2 error of the i-th interval
52         H1_error(i)=h/2*((sin(tt)-2*tt.*sin(tt)...
53             +tt.*(1-tt).*cos(tt)-duh*2/h).^2)*wv;
54         % the square of the H1 semi-norm error of the i-th interval
55     end
56     node_Err=[node_Err, node_error];
57     L2_Err=[L2_Err, sqrt(sum(L2_error))];
58     H1_Err=[H1_Err, sqrt(sum(L2_error)+sum(H1_error))];
59     doff=N+1; % degrees of freedom, number of unknowns
60     DOF=[DOF, doff];
61 end
62 loglog(DOF,node_Err,'r+-','LineWidth',1.5)
63 hold on
64 loglog(DOF,L2_Err,'bo-','MarkerFaceColor','w','LineWidth',1.5)
65 hold on
66 loglog(DOF,H1_Err,'b*-','LineWidth',1.5)
67 hold on, grid on
68 xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
69 title('Convergence of Finite Difference Method','fontsize',14)
70 set(gca,'fontsize',14)
71
72 for i=1:length(Num)-1 % calculating of convergence order
73     node_order(i)=log(node_Err(i)/node_Err(i+1))/(log(DOF(i)/DOF(i+1)));
74     L2_order(i)=log(L2_Err(i)/L2_Err(i+1))/(log(DOF(i)/DOF(i+1)));
75     H1_order(i)=log(H1_Err(i)/H1_Err(i+1))/(log(DOF(i)/DOF(i+1)));
76 end
77 node_order
78 L2_order
79 H1_order

```

## 4 Spectral Methods

### 4.1 Legendre-Galerkin Spectral Methods

**Example 4.1** Consider the two-point boundary value problem:

$$\begin{cases} -u''(x) + \alpha u(x) = f(x), & x \in I = (-1, 1) \\ u(-1) = 0, u(1) = 0. \end{cases} \quad (4.1)$$

Weak formulation:

$$\begin{cases} \text{Find } u \in H_0^1(I) \text{ such that} \\ (u', v') + \alpha(u, v) = (f, v), & v \in H_0^1(I) \end{cases} \quad (4.2)$$

Let  $\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x)$  satisfies the boundary condition, we have  $a_k = 0, b_k = -1$ . Then,

$$\phi_k(x) = L_k(x) - L_{k+2}(x) \quad (4.3)$$

We denote

$$X_N = \text{span}\{\phi_k : k = 1, 2, \dots, N-2\}.$$

Spectral Scheme:

$$\begin{cases} \text{Find } u_N \in X_N \text{ such that} \\ (u'_N, v'_N) + (u_N, v_N) = (f, v_N), & v_N \in X_N \end{cases} \quad (4.4)$$

Given a set of basis functions  $\{\phi_j\}_{j=0}^{N-2}$  of  $X_N$

$$\begin{aligned} f_k &= \int_I f_N \phi_k dx, & \mathbf{f} &= (f_0, f_1, \dots, f_{N-2})^T \\ u_N &= \sum_{j=0}^{N-2} \hat{u}_j \phi_j, & \mathbf{u} &= (\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{N-2})^T \\ s_{kj} &= - \int_I \phi_j'' \phi_k dx, & m_{kj} &= \int_I \phi_j \phi_k dx \end{aligned}$$

and

$$S = (s_{kj})_{0 \leq k, j \leq N-2}, \quad M = (m_{kj})_{0 \leq k, j \leq N-2}$$

Taking  $v_N = \phi_k$ . The linear system

$$(S + \alpha M)\mathbf{u} = \mathbf{f} \quad (4.5)$$

The stiffness matrix  $S = (s_{jk})$  is a diagonal matrix (P146-4.22):

$$s_{kk} = -(4k + 6)b_k = 4k + 6 \quad (4.6)$$

The mass matrix  $M = (m_{jk})$  is symmetric penta-diagonal (P146-4.23):

$$m_{jk} = m_{kj} = \begin{cases} \frac{2}{2k+1} + \frac{2}{(2k+5)}, & j = k \\ -\frac{2}{(2k+5)}, & j = k+2 \end{cases} \quad (4.7)$$

**Note** An immediate consequence is that  $\{\phi_k\}_{k=0}^{N-2}$  forms an orthogonal basis of  $X_N$  with respect to the inner product  $-(u_N'', v_N)$ . Furthermore, an orthonormal basis of  $X_N$  with respect to this inner product is

$$\tilde{\phi}_k(x) := \frac{1}{\sqrt{-b_k(4k+6)}} \phi_k(x)$$

In the following Matlab codes, we choose  $\tilde{\phi}_k(x)$  as basis function.

```

1 % LegenSM1.m
2 % Legendre-Galerkin Method for for the model equation
3 %  $-u_{xx}+u=f$  in  $(-1,1)$  with boundary condition  $u(-1)=u(1)=0$ ;
4 % exact solution:  $u=\sin(kw\pi x)$ ;
5 % RHS:  $f=kw\pi^2\sin(kw\pi x)+\sin(kw\pi x)$ ;
6 % Rmk: Use routines lepoly(); legs(); lepolym();
7 clear all
8 kw=10;
9 Nvec=[32:2:68]; % kw=10
10 %Nvec=[4:2:22] % kw=1
11 Errv=[]; % Initialization for error
12 for N=Nvec
13     [xv,wv]=legs(N); % Legendre-Gauss points and weights
14     Lm=lepolym(N+1,xv); % Lm is a Legendre polynomial matrix
15     u=sin(kw*pi*xv); % test function
16     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv); % Right-hand-side (RHS)
17     % Calculating coefficient matrix
18     S=eye(N); % stiff matrix
19     M=diag(1./(4*[0:N-1]+6))*diag(2./(2*[0:N-1]+1)+2./(2*[0:N-1]+5))
20         -diag(2./(sqrt(4*[0:N-3]+6).*sqrt(4*[0:N-3]+14)).*(2*[0:N-3]+5)),2)
21         -diag(2./(sqrt(4*[2:N-1]-2).*sqrt(4*[2:N-1]+6)).*(2*[2:N-1]+1)),-2);
22     % mass matrix
23     A=S+M;
24     % Solving the linear system
25     B=diag(1./sqrt(4*[0:N-1]+6))*(Lm(1:end-2,:)-Lm(3:end,:));
26     b=B*diag(wv)*f; % Solving RHS
27     uh=A\b; % expansion coefficients of  $u_N$ 
28     un=B'*uh; % compositing the numerical solution
29
30     error=norm(abs(un-u),inf); % maximum pointwise error
31     Errv=[Errv;error];
32 end
33 % Plot the maximum pointwise error
34 plot(Nvec,log10(Errv),'ro-','MarkerFaceColor','w','LineWidth',1.5)
35 grid on,
36 xlabel('N','fontsize',14), ylabel('log10(Error)','fontsize',14)
37 title('Round-off error of Legendre-Galerkin methods','fontsize',12)
38 set(gca,'fontsize',12)

```



```

1 % LegenSM2.m
2 % Legendre-Galerkin Method for the model equation
3 %  $-u''(x)+u'(x)+u(x)=f(x)$ ,  $x$  in  $(-1,1)$ ,
4 % boundary condition:  $u(-1)=u(1)=0$ ;
5 % exact solution:  $u=\sin(kw\pi xv)$ ;
6 % RHS:  $f=kw\pi^2\sin(kw\pi xv)+\sin(kw\pi xv)$ ;
7 % Rmk: Use routines lepoly(); legs(); lepolym();
8 clear all
9 kw=10;
10 Nvec=[32:2:68];
11 Errv=[];
12 for N=Nvec
13     [xv,wv]=legs(N); % Legendre-Gauss points and weights
14     Lm=lepolym(N+1,xv); % Lm is a Legendre polynomial matrix
15     u=sin(kw*pi*xv); % test function
16     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv)+kw*pi*cos(kw*pi*xv); % RHS
17     % Calculating coefficients matrix
18     S=eye(N); % stiffness matrix
19     M=diag(1./(4*[0:N-1]+6))*diag(2./(2*[0:N-1]+1)+2./(2*[0:N-1]+5))
20         -diag(2./(sqrt(4*[0:N-3]+6).*sqrt(4*[0:N-3]+14)).*(2*[0:N-3]+5)),2)
21         -diag(2./(sqrt(4*[2:N-1]-2).*sqrt(4*[2:N-1]+6)).*(2*[2:N-1]+1)),-2);
22     % mass matrix
23     D=diag(1./(sqrt(2.*[0:N-2]+3).*sqrt(2.*[0:N-2]+5)),1)...
24         +diag(-1./(sqrt(2.*[0:N-2]+3).*sqrt(2.*[0:N-2]+5)),-1);
25     % matrix derived from  $u'(x)$ 
26     A=S+M+D; % Coefficient matrix
27     % Solving the linear system
28     B=diag(1./sqrt(4*[0:N-1]+6))*(Lm(1:end-2,:)-Lm(3:end,:));
29     b=B*diag(wv)*f;
30     uh=A\b; % expansion coefficients of  $u_N$ 
31     un=B'*uh; % Coefficients to points
32     error=norm(abs(un-u),inf); % maximum pointwise error
33     Errv=[Errv;error];
34 end
35 % Plot the maximum pointwise error
36 plot(Nvec,log10(Errv),'mo-','MarkerFaceColor','w','LineWidth',1.5)
37 grid on, xlabel('N','fontsize',14), ylabel('log10(Error)','fontsize',14)
38 title('Round-off error of Legendre-Galerkin methods','fontsize',12)
39 set(gca,'fontsize',12)

```

**Example 4.2** Consider the two-point boundary value problem:

$$\begin{cases} -u''(y) + u(y) = f(y), & y \in \Lambda = [0, 1] \\ u(0) = 1, u'(1) = 0. \end{cases} \quad (4.8)$$

Let  $x \in I = [-1, 1]$ ,  $y = \frac{x}{2} + \frac{1}{2}$  and  $U(x) = u(y) - 1$ , the converted problem:

$$\begin{cases} -4U''(x) + U(x) = F(x), & x \in I = [-1, 1] \\ U(-1) = 0, U'(1) = 0. \end{cases} \quad (4.9)$$

where  $F(x) = f(2x - 1) - 1$ .

Weak formulation:

$$\begin{cases} \text{Find } U \in H^1(I) \text{ such that} \\ 4(U', v'_N) + (U, v_N) = (f, v_N), \quad v_N \in H^1(I) \end{cases} \quad (4.10)$$

Let  $\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x)$  satisfies the boundary condition, we have

$$a_k = \frac{2k+3}{(k+2)^2}, \quad b_k = -\frac{(k+1)^2}{(k+2)^2}. \quad (4.11)$$

Let us denote

$$X_N = \text{span}\{\phi_k, k = 0, 1, \dots, N-2\} \quad (4.12)$$

Spectral Scheme:

$$\begin{cases} \text{Find } U_N \in X_N \text{ such that} \\ 4(U'_N, \phi') + (U_N, \phi_N) = (f, \phi), \quad \phi \in X_N \end{cases} \quad (4.13)$$

The stiffness matrix  $S = (s_{jk})$  is a diagonal matrix (P146-4.22):

$$s_{kk} = -(4k+6)b_k = \frac{(4k+6)(k+1)^2}{(k+2)^2} \quad (4.14)$$

The mass matrix  $M = (m_{jk})$  is symmetric penta-diagonal (P146-4.23):

$$m_{jk} = m_{kj} = \begin{cases} \frac{2}{2k+1} + \frac{2(2k+3)}{(k+2)^4} + \frac{2(k+1)^4}{(k+2)^4(2k+5)}, & j = k \\ \frac{2}{(k+2)^2} - \frac{2(k+1)^2}{(k+2)^2(k+3)^2}, & j = k+1 \\ -\frac{2(k+1)^2}{(k+2)^2(2k+5)}, & j = k+2 \end{cases} \quad (4.15)$$

```

1 % LegenSM3.m
2 % Legendre-Spectral Method for 1D elliptic problem
3 %  $-u_{yy}+u=f$  in  $[0,1]$  with boundary condition:  $u(0)=1, u'(1)=0$ ;
4 % exact solution:  $u=(1-y)^2 \exp(y)$ ; RHS:  $f=(2-4*y) \exp(y)$ ;
5 % Converted :  $-4U_{xx}+U=F$  in  $[-1,1]$ 
6 % boundary condition:  $U(-1)=0, U'(1)=0$ ;
7 % exact solution:  $U=(1/2-1/2*x)^2 \exp(1/2*x+1/2)-1$ ;
8 % RHS:  $F=-2*x \exp(1/2*x+1/2)-1$ .
9 clear all
10 Nvec=2:16;
11 Errv=[]; condnv=[]; % Initialization for error and condition number
12 for N=Nvec
13     [xv,wv]=legs(N); % xv and wv are Legendre-Gauss points and weights
14     Lm=lepolym(N+1,xv); % Lm is a Legendre polynomial matrix
15     yv=1/2*(xv+1); % variable substitution
16     U=(1-yv).^2.*exp(yv)-1; % test function
17     F=(2-4*yv).*exp(yv)-1; % RHS in  $[0,1]$ 
18     % Calculating coefficient matrix
19     e1=0:N-1; e2=0:N-2; e3=0:N-3;
20     S=diag( (4*e1+6).*(e1+1).^2./(e1+2).^2 ); % stiff matrix
21     M=diag( 2./(2*e1+1)+2*(2*e1+3)./(e1+2).^4+2*((e1+1)./(e1+2)).^4./(2*e1+5))
22         +diag( 2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2*(e2+3).^2) , 1 )
23         +diag( 2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2*(e2+3).^2) ,-1)
24         +diag( -2*(e3+1).^2./((2*e3+5).*(e3+2).^2) , 2 )
25         +diag(-2*(e3+1).^2./((2*e3+5).*(e3+2).^2),-2); % mass matrix
26     A=4*S+M;
27     % Solving the linear system
28     B=(Lm(1:end-2,:)+diag( (2*e1+3)./(e1+2).^2)*Lm(2:end-1,:))...
29         -diag( (e1+1).^2./(e1+2).^2)*Lm(3:end,:));
30     b=B*diag(wv)*F; % Solving RHS
31     Uh=A\b; % expansion coefficients of  $u_N$ 
32     Un=B'*Uh; % compositing the numerical solution
33     error=norm(abs(Un-U),2); %  $L^2$  error
34     Errv=[Errv;error];
35     condnv=[condnv,cond(A)]; % condition number of A
36 end
37 % Plot the maximum pointwise error
38 plot(Nvec,log10(Errv),'go-','MarkerFaceColor','w','LineWidth',2)
39 grid on,
40 xlabel('N','fontsize',14), ylabel('log_{10}(Error)','fontsize',14)
41 title('L^2 error of Legendre-Galerkin method','fontsize',12)
42 set(gca,'fontsize',12)

```

## 4.2 Collocation Methods

**Example 4.3** The two-point boundary value problem:

$$\begin{cases} -u''(x) + \alpha u(x) = f(x), & x \in I = (-1, 1) \\ u(-1) = 0, u(1) = 0. \end{cases} \quad (4.16)$$

Exact solution:  $u = \sin(k\pi x), f = k^2\pi^2 \sin(k\pi x) + \alpha \sin(k\pi x)$ .

```

1 % LegenCollo1.m
2 % Legendre-collocation method for the model equation:
3 % -u''(x)+\alpha u(x)=f(x), x in (-1,1);
4 % boundary condition: u(-1)=u(1)=0;
5 % exact solution: u=sin(kw*pi*x);
6 % RHS: f=kw*kw*pi^2*sin(kw*pi*x)+alpha*sin(kw*pi*x);
7 % Rmk: Use routines lepoly(); legslb(); legslbdm();
8 clear all
9 alpha=1;
10 kw=10;
11 N=32;
12 Nvec=[32:2:68];
13 Errv=[];
14 for N=Nvec
15     [x,w]=legslb(N);           % compute LGL nodes and weights
16     u=sin(kw*pi*x);           % test solution
17     udprime=-kw*kw*pi*pi*sin(kw*pi*x);
18     f=-udprime+alpha*u;       % RHS
19     % Setup and solve the collocation system
20     D1=legslbdm(N);           % 1st order differentiation matrices
21     %D1=legslbdiff(N,x);      % 1st order differentiation matrices
22     D2=D1*D1;                 % 2nd order differentiation matrices
23     D=(-D2(2:N-1,2:N-1)+alpha*eye(N-2)); % coefficient matrix
24     b=f(2:N-1);               % RHS
25     un=D\b;
26     un=[0;un;0];              % Solve the system
27
28     error=norm(abs(un-u),inf); % maximum pointwise error
29     Errv=[Errv;error];
30 end;
31 plot(Nvec,log10(Errv),'rd-','MarkerFaceColor','w','LineWidth',1.5)
32 grid on, xlabel('N','fontsize',14), ylabel('log10(Error)','fontsize',14)
33 title('Convergence of Legendre-collocation method','fontsize',12)
34 set(gca,'fontsize',12)

```

```

1 % LegenCollo2.m
2 % Legendre-collocation Method for the model equation:
3 %  $-u''(x)+u'(x)+u(x)=f(x)$ ,  $x$  in  $(-1,1)$ ;
4 % % boundary condition:  $u(-1)=u(1)=0$ ;
5 % exact solution:  $u=\sin(kw\pi x)$ ;
6 % RHS:  $f(x)=kw^2\pi^2\sin(kw\pi x)+\sin(kw\pi x)+kw\pi\cos(kw\pi x)$ ;
7 % Rmk: Use routines lepoly(); legslb(); legslbdm();
8 clear all
9 kw=10;
10 Nv=[32:2:68];
11 Errv=[];
12 for N=Nv
13     [xv,wv]=legslb(N); % compute LGL nodes and weights
14     u=sin(kw*pi*xv); % test function
15     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv)+kw*pi*cos(kw*pi*xv); % RHS
16     % Setup and solve the collocation system
17     D1=legslbdm(N); % 1st order differentiation matrix
18     D2=D1*D1; % 2nd order differentiation matrix
19     D=-D2(2:N-1,2:N-1)+D1(2:N-1,2:N-1)+eye(N-2); % coefficient matrix
20     b=f(2:N-1); % RHS
21     un=D\b;
22     un=[0;un;0]; % Solve the system
23
24     error=norm(abs(un-u),inf);
25     Errv=[Errv;error];
26 end
27 % Plot the maximum pointwise error
28 plot(Nv,log10(Errv),'md-','MarkerFaceColor','w','LineWidth',1.5)
29 grid on,
30 xlabel('N','fontsize',14), ylabel('log10(Error)','fontsize',14)
31 title('Convergence of Legendre-collocation method','fontsize',12)

```

**Example 4.4** The two-point boundary value problem:

$$\begin{cases} -u''(y) + u(y) = f(y), & y \in \Lambda = [0, 1] \\ u(0) = 1, u'(1) = 0. \end{cases} \quad (4.17)$$

Exact solution:  $u(y) = (1 - y)^2 \exp(y)$ ,  $f(y) = (2 - 4y) \exp(y)$ .

```

1 % LegenCollo3.m
2 % Legendre-collocation Method for the model equation:
3 % -u''(y)+u(y)=f(y) in [0,1] with boundary condition: u(0)=1, u'(1)=0;
4 % test function : u(y)=(1-y)^2*exp(y);
5 % RHS : f(y)=(2-4*y)*exp(y);
6 % Rmk: Use routines legslb(); legslbdiff();
7 clear all
8 Nvec=4:18;
9 Errv=[]; condnv=[];
10 for N=Nvec
11     xv=legslb(N);           % compute LGL nodes and weights
12     yv=1/2*(xv+1);         % variable substitution
13     u=(1-yv).^2.*exp(yv);   % test solution in [0,1]
14     f=(2-4*yv).*exp(yv);    % RHS in [0,1]
15
16     % Setup and solve the collocation system
17     D1=legslbdiff(N,xv);    % 1st order differentiation matrices
18     D2=D1*D1;              % 2nd order differentiation matrices
19     D=-4*D2+eye(N);        % coefficient matrix
20     D(1,:)= [1,zeros(1,N-1)]; D(N,:)=D1(N,:);
21     b=[1; f(2:N-1); 0];    % RHS
22     un=D\b;                % Solve the system
23
24     error=norm(abs(un-u),2); % L^2 error
25     Errv=[Errv;error];
26     condnv=[condnv,cond(D)];
27 end
28 % Plot the L^2 error
29 plot(Nvec,log10(Errv),'gd-','MarkerFaceColor','w','LineWidth',1.5)
30 grid on, xlabel('N','fontsize',14), ...
    ylabel('log_{10}(Error)','fontsize',14),
31 title('L^2 error of Legendre-collocation method','fontsize',12)

```