MATLAB Notes and Code

Liutao Tian

April 25, 2020

Contents

1 Numerical Methods for ODEs 2

1.1 Euler Method 2

1.2 Modified Euler Method 3

1.3 Runge-Kutta Method 4

2 Finite Difference Method 8

2.1 Finite Difference Methods for 1-D Problem 8

2.2 Finite Difference Methods for 2-D Problem 10

3 Finite Element Methods 15

3.1 Galerkin Method for 1-D Problem 15

4 Spectral Methods 21

4.1 Legendre-Galerkin Spectral Methods 21

4.2 Collocation Methods 27

1 Numerical Methods for ODEs

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

$$\begin{cases} \dfrac{du}{dt} = f(t,u), \quad 0 < t \leqslant T, \\ u(0) = 0. \end{cases} \tag{1.1}$$

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

1.1 Euler Method

Euler method scheme:

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

Example 1.1

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

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

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

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

1.3 Runge-Kutta Method

Runge-Kutta method scheme:

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

Example 1.3

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

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

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

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

is defined by

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

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

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, 2, \ldots, s. \tag{1.6}$$

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

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

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

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

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

$$k_i = f\left(x_n + c_i h, Y_i\right), \quad i = 1, 2, \ldots, s.$$

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

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

Butcher array

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

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

Example 1.4

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

```matlab
% IRK2s_error.m
% Implicit Runge-Kutta(Gauss method) 2 stage and order 4
% u'=u in [0,1] with initial condition u(0)=1
% exact solution: ue=exp(x)
clear all
Nvec=[10 50 100 200 500 1000];
Err=[];
for n=1:length(Nvec)
    N=Nvec(n);  h=1/N;
    x=[0:h:1];
    u(1)=1;
    X0=[1;1];
    % Newton iteration
    for i=1:N
        k=u(i);
        r=X0; tol=1;
        while tol>1.0e-6
            X=r;
            D=[1-0.25*h,-h*(0.25-(sqrt(3))/6);...
            -h*( 0.25+(sqrt(3))/6),1-h*0.25];    % Jacobian matrix
            F=[X(1)-k-h*(0.25*X(1)+(0.25-(sqrt(3))/6)*X(2));...
            X(2)-k-h*((0.25+(sqrt(3))/6)*X(1)+0.25*X(2))];   % RHS
            r=X-D\F;
            tol=norm(r-X);
        end
        k1=r(1);  k2=r(2);
        u(i+1)=k+(h/2)*(k1+k2);
        X0=r;
    end
    ue=exp(x);              % exact solution
    err=max(abs(u-ue));     % maximum error
    Err=[Err,err];
end
plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
hold on,
plot(log10(Nvec), log10(Nvec.^(-4)), '--')
grid on,
xlabel('log_{10}N','fontsize',16), ylabel('log_{10}Error','fontsize',16)
title('Convergence order of Gauss method ','fontsize',14)
set(gca,'fontsize',14)
for i=1:length(Nvec)-1     % computating convergence order
    order(i)=-log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Err
order
```

2 Finite Difference Method

2.1 Finite Difference Methods for 1-D Problem

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

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

Discrete difference scheme:

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

Example 2.1

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

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

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

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

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

Discrete difference scheme:

$$\begin{aligned}&-\frac{2}{h_i+h_{i+1}}\left[p_{i+\frac{1}{2}}\frac{u(x_{i+1})-u(x_i)}{h_{i+1}}+p_{i-\frac{1}{2}}\frac{u(x_i)-u(x_{i-1})}{h_i}\right]+\\&\frac{r_i}{h_i+h_{i+1}}(u(x_{i+1})-u(x_{i-1}))+q_iu(x_i)=f(x_i), i=1,\cdots,N-1.\end{aligned} \tag{2.5}$$

Example 2.2

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

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

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

2.2 Finite Difference Methods for 2-D Problem

Consider the two-dimensional Poisson problem:

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

Discrete difference scheme:

$$\begin{aligned} -\frac{1}{h_2^2}u_{i,j-1} - \frac{1}{h_1^2}u_{i-1,j} + 2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right)u_{i,j} - \frac{1}{h_1^2}u_{i+1,j} - \frac{1}{h_2^2}u_{i,j+1} &= f\left(x_i, y_j\right), \\ 1 \leqslant i \leqslant N-1, \quad 1 \leqslant j \leqslant M-1. \end{aligned} \tag{2.8}$$

Define the vector: $\boldsymbol{u}_j = \left(u_{1j}, u_{2j}, \ldots, u_{N-1,j}\right)^{\mathrm{T}}, \quad 0 \leqslant j \leqslant M.$

The discrete scheme to matrix form:

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

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

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

Next, above can be written in the following matrix form

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

Example 2.3

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

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

```
% fdm2d1.m
% finite difference method for 2D problem
% -d^2u/dx^2-d^2u/dy^2=f(x,y)
% f(x,y)=-2*pi^2*exp(pi*(x+y))*(sin(pi*x)*cos(pi*y)+cos(pi*x)*sin(pi*y))
% exact solution: ue=exp(pi*x+pi*y)*sin(pi*x)*sin(pi*y)
clear all
h=0.01;
x=[0:h:1]';
y=[0:h:1]';
N=length(x)-1;
M=length(y)-1;
[X,Y]=meshgrid(x,y);
X=X(2:M,2:N);
Y=Y(2:M,2:N);
% generate the matrix of RHS
f=-2*pi^2*exp(pi*X+pi*Y).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
% constructing the coefficient matrix
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);
D=-1/h^2*eye(N-1);
A=kron(eye(M-1),C)+kron(diag(ones(M-2,1),1)+diag(ones(M-2,1),-1),D);
% solving the linear system
f=f';
u=zeros(M+1,N+1);
u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
u(:,1)=0;
u(:,end)=0;
ue=zeros(M+1,N+1);
ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
% compute maximum error
Error=max(max(abs(u-ue)))
mesh(x,y,u)
xlabel('x','fontsize', 16), ylabel('y','fontsize',16), zlabel('u','fontsize',16,'Rotation',0)
title('Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)
```

```
% fdm2d1_error.m
% finite difference method for 2D problem
% -d^2u/dx^2-d^2u/dy^2=f(x,y)
%f(x,y)=-2*pi^2*exp(pi*(x+y))*(sin(pi*x)*cos(pi*y)+cos(pi*x)*sin(pi*y))
% exact solution: ue=exp(pi*x+pi*y)*sin(pi*x)*sin(pi*y)
clear all
Nvec=2.^[3:10];  Err=[];
for n=Nvec
   h=1/n;
   x=[0:h:1]';     y=[0:h:1]';
   N=length(x)-1;  M=length(y)-1;
   [X,Y]=meshgrid(x,y);
   X=X(2:M,2:N);
   Y=Y(2:M,2:N);
   % generate the matrix of RHS
   f=-2*pi^2*exp(pi*(X+Y)).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
   % constructing the coefficient matrix
   e=ones(N-1,1);
   C=1/h^2*spdiags([-e 4*e -e],[-1 0 1],N-1,N-1);
   D=-1/h^2*eye(N-1);
   e=ones(M-1,1);
   A=kron(eye(M-1),C)+kron(spdiags([e e],[-1 1],M-1,M-1),D);
   % solving the linear system
   f=f';
   u=zeros(M+1,N+1);
   u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
   u(:,1)=0;
   u(:,end)=0;
   ue=zeros(M+1,N+1);        % numerical solution
   ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
   err=max(max(abs(u-ue)));  % maximum error
   Err=[Err,err];
end
plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
grid on,hold on, plot(log10(Nvec), log10(Nvec.^(-2)), '--')
xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)
for i=1:length(Nvec)-1     % computating convergence order
   order(i)=-log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Err
order
```

Example 2.4

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

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: $x_i = ih_1$, $y_j = jh_2$, $i,j = 0,1,\cdots,N$.

Discrete difference scheme:

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

Boundary conditions:

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

Discrete scheme:

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

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

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

Matrix form:

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

```
% fdm2d2_error.m
% finite difference method for 2D problem
% -\Delta u=cos(3*x)*sin(pi*y)  in (0,pi)x(0,1)
% u(x,0)=u(x,1)=0  in [0,pi]
% u_x(0,y)=u_x(pi,y)=0 in [0,1]
% exact solution: ue=(9+pi^2)^(-1)*cos(3*x)*sin(pi*y)
clear all; close all;
Nvec=2.^[2:7];  Err=[];
for N=Nvec
    h1=pi/N;  h2=1/N;
    x=[0:h1:pi]';   y=[0:h2:1]';
    [X,Y]=meshgrid(x,y);
    X1=X(2:N,2:N);  Y1=Y(2:N,2:N);
    % generate the matrix of RHS
    f=cos(3*X1).*sin(pi*Y1);
    % constructing the coefficient matrix
    e=ones(N-1,1);
    C=diag([1/h1^2+2/h2^2, (2/h1^2+2/h2^2)*ones(1,N-3), 1/h1^2+2/h2^2])...
        -1/h1^2*diag(ones(N-2,1),1)-1/h1^2*diag(ones(N-2,1),-1);
    D=-1/h2^2*eye(N-1);
    A=kron(eye(N-1),C)+kron(diag(ones(N-2,1),1)+diag(ones(N-2,1),-1),D);
    %A=kron(eye(N-1),C)+kron(spdiags([e e],[-1 1],N-1,N-1),D);
    % solving the linear system
    f=f';
    u=zeros(N+1,N+1);
    u(2:N,2:N)=reshape(A\f(:),N-1,N-1)';
    % Neumann boundary condition
    u(:,1)=u(:,2);
    u(:,end)=u(:,end-1);
    ue=1/(9+pi^2)*(cos(3*X)).*(sin(pi*Y));
    err=max(max(abs(u-ue)));   % maximum error
    Err=[Err,err];
end
plot(log10(Nvec),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5)
grid on,hold on, plot(log10(Nvec),log10(Nvec.^(-1)),'--')
xlabel('log_{10}N','fontsize', 16),ylabel('log_{10}Error','fontsize',16),
title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)
for i=1:length(Nvec)-1     % computating convergence order
    order(i)=log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
order
```

3 Finite Element Methods

3.1 Galerkin Method for 1-D Problem

Consider the two-point boundary value problem:

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

Set

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

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

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

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

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

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

$$a(u_h, \phi_i) = \langle f, \phi_i \rangle, \quad i = 1, \cdots, N. \tag{3.4}$$

Writing u_h in the form

$$u_h = \sum_{j=1}^N u_j \phi_j, \tag{3.5}$$

we are led to the system of equations

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

which we can write in the matrix-vector form as

$$A\boldsymbol{u} = \boldsymbol{b}. \tag{3.7}$$

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

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

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

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

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

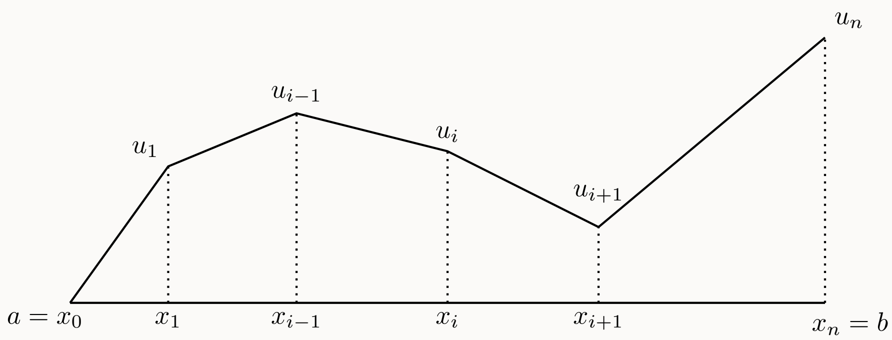
The test function space U_h is composed of piecewise linear functions. Its set of values on the node

$$
u_0, u_1, u_2, \cdots, u_n,
$$

Linear interpolation formula

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

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:

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

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

Basis function

$$\varphi_1 = \frac{1}{2}(\Phi_2^{I_1} + \Phi_1^{I_2}), \quad \varphi_2 = \frac{1}{2}(\Phi_2^{I_2} + \Phi_1^{I_3}), \quad \cdots$$
$$\varphi_i = \frac{1}{2}(\Phi_2^{I_i} + \Phi_1^{I_{i+1}}), \quad \cdots \quad \varphi_n = \Phi_2^{I_n}.$$

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

$$K_{11}^{I_i} = a(\Phi_1^{I_i}, \Phi_1^{I_i}) = \int_{x_{i-1}}^{x_i} (p\Phi_1^{I_i\prime} \cdot \Phi_1^{I_i\prime 2} + q\Phi_1^{I_i} \cdot \Phi_2^{I_i})dx,$$

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

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

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

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:

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

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

```
% FEM1D.m
% Finite Element Method
% -u_xx+u=f in (0,1) with boundary condition u(0)=u(1)=0;
% exact : u=x*(1-x)*sin(x)
% RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x);
% Code from teacher Yi Lijun
clear all
Num=[16 32 64 128 256 512];   % Number of splits
Err=[];  DOF=[];
for j=1:length(Num)
    N=Num(j);    h=1/N;    x=0:h:1;
    % The global node number corresponds to element local node number
    M=[1:N;2:N+1];
```

```
    [xv,wv]=jags(2,0,0); % nodes and weights of gauss quadrature

    K=zeros(N+1);          % global stiffness matrix
    F=zeros(N+1,1);        % RHS load vector
    for i=1:N              % loop for each element
        K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
                 +((h/2)*(((1/4)*(2/h)^2+((1-xv)/2).^2)))'*wv;
        K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2
                 +((1+xv)/2).^2)))'*wv;

        t=h*xv/2+(x(i+1)+x(i))/2;
        F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
        F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
    end
    % Dirichlet boundary condition
    K(1,: )=zeros(1,N+1);
    K(:,1)=zeros(1,N+1);
    K(N+1,: )=zeros(1,N+1);
    K(:, N+1)=zeros(1,N+1);
    K(1,1)=1;   K(N+1,N+1)=1;
    F(1)=0;     F(N+1)=0;

    U=K\F;                 % numerical solution at the value of the node
    error=max(abs(U'-x.*(1-x).*sin(x)));  % node error
    doff=N+1;              % degrees of freedom, number of unknowns
    Err=[Err, error];
    DOF=[DOF, doff];
end
plot(log10(DOF),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5),
hold on,
plot(log10(DOF),log10(DOF.^(-2)),'--')
grid on,
xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
title('Convergence of Finite Element Method','fontsize',14)
set(gca,'fontsize',14)
```

```
% FEM1DP.m
% FEM for 1D elliptic problem
% -u_xx+u=f in [0,1] with  boundary condition u(0)=u(1)=0;
% exact solution: u=x*(1-x)*sin(x);
% RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x)
% Code from teacher Yi Lijun
clear all
Num=[16 32 64 128 256 512]
node_Err=[];   L2_Err=[];  H1_Err=[];  DOF=[];
for j=1:length(Num)
    N=Num(j);    h=1/N;    x=0:h:1;
    % The global node number corresponds to element local node number
```

```
    M=[1:N;2:N+1];
    [xv,wv]=jags(3,0,0);   % nodes and weights of gauss quadrature
    K=zeros(N+1);          % global stiffness matrix
    F=zeros(N+1,1);        % RHS load vector

    for i=1:N   % loop for each element
        K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
                 +((h/2)*(((1/4)*(2/h)^2+((1-xv)/2).^2)))'*wv;
        K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2
                 +((1+xv)/2).^2)))'*wv;

        t=h*xv/2+(x(i+1)+x(i))/2;
        F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
        F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
    end
    % Handling Dirichlet boundary condition
    K(1,: )=zeros(1,N+1);
    K(:,1)=zeros(1,N+1);
    K(N+1,: )=zeros(1,N+1);
    K(:, N+1)=zeros(1,N+1);
    K(1,1)=1;   K(N+1,N+1)=1;
    F(1)=0;     F(N+1)=0;

    U=K\F;       % numerical solution at the value of the nodes
    node_error=max(abs(U'-x.*(1-x).*sin(x)));  % node error
    for i=1:N
        tt=h*xv/2+(x(i+1)+x(i))/2;
        % value of finite element solution at Gauss point
        uh=U(i)*(1-xv)/2+U(i+1)*(1+xv)/2;
        % derivative value of finite element solution at Gauss point
        duh=-U(i)/2+U(i+1)/2;
        L2_error(i)=h/2*((tt.*(1-tt).*sin(tt)-uh).^2)'*wv;
        % the square of the L2 error of the i-th interval
        H1_error(i)=h/2*((sin(tt)-2*tt.*sin(tt)...
                    +tt.*(1-tt).*cos(tt)-duh*2/h).^2)'*wv;
        % the square of the H1 semi-norm error of the i-th interval
    end
    node_Err=[node_Err, node_error];
    L2_Err=[L2_Err, sqrt(sum(L2_error))];
    H1_Err=[H1_Err, sqrt(sum(L2_error)+sum(H1_error))];
    doff=N+1;    % degrees of freedom, number of unknowns
    DOF=[DOF, doff];
end
loglog(DOF,node_Err,'r+-','LineWidth',1.5)
hold on
loglog(DOF,L2_Err,'bo-','MarkerFaceColor','w','LineWidth',1.5)
hold on
loglog(DOF,H1_Err,'b*-','LineWidth',1.5)
hold on, grid on
xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
```

```
title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)

for i=1:length(Num)-1    % calculating of convergence order
    node_order(i)=log(node_Err(i)/node_Err(i+1))/(log(DOF(i)/DOF(i+1)));
    L2_order(i)=log(L2_Err(i)/L2_Err(i+1))/(log(DOF(i)/DOF(i+1)));
    H1_order(i)=log(H1_Err(i)/H1_Err(i+1))/(log(DOF(i)/DOF(i+1)));
end
node_order
L2_order
H1_order
```

4 Spectral Methods

4.1 Legendre-Galerkin Spectral Methods

Example 4.1 Consider the two-point boundary value problem[1]

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

Weak formulation:

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

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,

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

We denote

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

Spectral Scheme:

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

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

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

and

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

Taking $v_N = \phi_k$. The linear system

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

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

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

1Reference book: Spectral Methods: Algorithms, Analysis and Applications, 2011

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

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

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.

```matlab
% LegenSM1_error.m
% Legendre-Galerkin Method for for the model equation
% -u_xx+u=f in (-1,1) with boundary condition u(-1)=u(1)=0;
% exact solution: u=sin(kw*pi*x);
% RHS: f=kw*kw*pi^2*sin(kw*pi*x)+sin(kw*pi*x);
% Rmk: Use routines lepoly(); legs(); lepolym();
clear all;  clf
kw=10;
Nvec=[32:2:76];    % kw=10
%Nvec=[4:2:22]     % kw=1
Errv=[];           % Initialization for error
for N=Nvec
    [xv,wv]=legs(N+1);   % Legendre-Gauss points and weights
    Lm=lepolym(N,xv);    % matrix of Legendre polynomals
    u=sin(kw*pi*xv);     % test function
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv);    % Right-hand-side(RHS)
    % Calculting coefficient matrix
    S=eye(N-1);            % stiff matrix
    M=diag(1./(4*[0:N-2]+6))*diag(2./(2*[0:N-2]+1)+2./(2*[0:N-2]+5))...
        -diag(2./(sqrt(4*[0:N-4]+6).*sqrt(4*[0:N-4]+14).*(2*[0:N-4]+5)),2)...
        -diag(2./(sqrt(4*[2:N-2]-2).*sqrt(4*[2:N-2]+6).*(2*[2:N-2]+1)),-2);
        % mass matrix
    A=S+M;
    % Solving the linear system
    Pm=diag(1./sqrt(4*[0:N-2]+6))*(Lm(1:end-2,:)-Lm(3:end,:));
    % matrix of Phi(x)
    b=Pm*diag(wv)*f; % Solving RHS
    uh=A\b;          % expansion coefficients of u_N in terms of the basis
    un=Pm'*uh;       % compositing the numerical solution

    %error=norm(abs(un-u),2);  % maximum pointwise error
    error=norm(abs(un-u),2);   % L^2 error
    Errv=[Errv;error];
end
% Plot the maximum pointwise error
plot(Nvec,log10(Errv),'ro-','MarkerFaceColor','w','LineWidth',1.5)
grid on,
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
title('L^2 error of Legendre-Galerkin methods','fontsize',12)
set(gca,'fontsize',12)

print -dpng -r600  LegenSM1_error.png
```

```matlab
% LegenSM2_error.m
% Legendre-Galerkin Method for the model equation
% -u''(x)+u'(x)+u(x)=f(x), x in (-1,1),
% boundary condition: u(-1)=u(1)=0;
% exact solution: u=sin(kw*pi*xv);
% RHS: f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv);
% Rmk: Use routines lepoly(); legs(); lepolym();
clear all;  clf
kw=10;
Nvec=[32:2:76];
Errv=[];
for N=Nvec
    [xv,wv]=legs(N+1);   % Legendre-Gauss points and weights
    Lm=lepolym(N,xv);    % matrix of Legendre polynomals
    u=sin(kw*pi*xv);     % test function
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv)+kw*pi*cos(kw*pi*xv);  % RHS
    % Calculating coefficients matrix
    S=eye(N-1);            % stiffness matrix
    M=diag(1./(4*[0:N-2]+6))*diag(2./(2*[0:N-2]+1)+2./(2*[0:N-2]+5))...
        -diag(2./(sqrt(4*[0:N-4]+6).*sqrt(4*[0:N-4]+14).*(2*[0:N-4]+5)),2)...
        -diag(2./(sqrt(4*[2:N-2]-2).*sqrt(4*[2:N-2]+6).*(2*[2:N-2]+1)),-2);
        % mass matrix
    D=diag(1./(sqrt(2.*[0:N-3]+3).*sqrt(2.*[0:N-3]+5)),1)...
        +diag(-1./(sqrt(2.*[0:N-3]+3).*sqrt(2.*[0:N-3]+5)),-1);
        % matrix of derivative term
    A=S+M+D;     % Coefficient matrix
    % Solving the linear system
    Pm=diag(1./sqrt(4*[0:N-2]+6))*(Lm(1:end-2,:)-Lm(3:end,:));
    % matrix of Phi(x)
    b=Pm*diag(wv)*f;
    uh=A\b;      % expansion coefficients of u_N in terms of the basis
    un=Pm'*uh;   % Coefficiets to points
    error=norm(abs(un-u),inf);   % maximum pointwise error
    Errv=[Errv;error];
end
% Plot the maximum pointwise error
plot(Nvec,log10(Errv),'md-','MarkerFaceColor','w','LineWidth',1.5)
grid on, xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
title('L^{\infty} error of Legendre-Galerkin methods','fontsize',12)
set(gca,'fontsize',12)

print -dpng -r600  LegenSM2_error.png
```

Example 4.2 Consider the two-point boundary value problem:

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

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

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

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

Weak formulation:

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

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

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

Let us denote

$$X_N=span\{\phi_k, k=0,1,\cdots,N-2.\} \tag{4.12}$$

Spectral Scheme:

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

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

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

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

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

```
% LegenSM3_error.m
% Legendre-Spectral Method for 1D elliptic problem
% -u_yy+u=f in [0,1] with boundary condition: u(0)=1, u'(1)=0;
% exact solution: u=(1-y)^2*exp(y);   RHS: f=(2-4*y)*exp(y);
% Converted : -4U_xx+U=F in [-1,1]
% boundary condition: U(-1)=0, U'(1)=0;
% exact solution: U=(1/2-1/2*x)^2*exp(1/2*x+1/2)-1;
% RHS:  F=-2*x*exp(1/2*x+1/2)-1.
clear all;  clf
Nvec=3:16;
Errv=[];  condnv=[];    % Initialization for error and condition number
for N=Nvec
    [xv,wv]=legs(N+1);     % Legendre-Gauss points and weights
    Lm=lepolym(N,xv);      % matrix of Legendre polynomals
    yv=1/2*(xv+1);         % variable substitution
    U=(1-yv).^2.*exp(yv)-1;    % test function
    F=(2-4*yv).*exp(yv)-1;     % RHS in [0,1]
    % Calculting coefficient matrix
    e1=0:N-2;  e2=0:N-3;  e3=0:N-4;
    S=diag( (4*e1+6).*(e1+1).^2./(e1+2).^2 );     % stiff matrix
    M=diag(2./(2*e1+1)+2*(2*e1+3)./(e1+2).^4+2*((e1+1)./(e1+2)).^4./(2*e1+5))...
        +diag( 2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2.*(e2+3).^2) , 1 )...
        +diag(2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2.*(e2+3).^2),-1)...
        +diag( -2*(e3+1).^2./((2*e3+5).*(e3+2).^2) , 2 )...
        +diag(-2*(e3+1).^2./((2*e3+5).*(e3+2).^2),-2);  % mass matrix
    A=4*S+M;

    % Solving the linear system
    Pm=(Lm(1:end-2,:)+diag((2*e1+3)./(e1+2).^2)*Lm(2:end-1,:)...
        -diag((e1+1).^2./(e1+2).^2)*Lm(3:end,:));   % matrix of Phi(x)
    b=Pm*diag(wv)*F;  % Solving RHS
    Uh=A\b;           % expansion coefficients of u_N in terms of the basis
    Un=Pm'*Uh;        % compositing the numerical solution
    error=norm(abs(Un-U),2);   % L^2 error
    Errv=[Errv;error];
    condnv=[condnv,cond(A)];   % condition number of A
end
% Plot the maximum pointwise error
plot(Nvec,log10(Errv),'s-','color',[0 0.5 0],'MarkerFaceColor','w','LineWidth',1.5)
grid on, xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
title('L^2 error of Legendre-Galerkin method','fontsize',12)
set(gca,'fontsize',12)
print -dpng -r600  LegenSM3_error.png
```

4.2 Collocation Methods

Example 4.3 The two-point boundary value problem:

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

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

```
% LegenCM1_error.m
% Legendre-collocation method for the model equation:
% -u''(x)+\alpha u(x)=f(x), x in (-1,1);
% boundary condition: u(-1)=u(1)=0;
% exact solution: u=sin(kw*pi*x);
% RHS: f=kw*kw*pi^2*sin(kw*pi*x)+alpha*sin(kw*pi*x);
% Rmk: Use routines lepoly(); legslb(); legslbdm();
clear all;  clf
alpha=1;
kw=10;
Nvec=[32:2:68];
Errv=[];
for N=Nvec
    [x,w]=legslb(N);        % compute LGL nodes and weights
    u=sin(kw*pi*x);         % test solution
    udprime=-kw*kw*pi*pi*sin(kw*pi*x);
    f=-udprime+alpha*u;     % RHS
    % Setup and solve the collocation system
    D1=legslbdm(N);         % 1st order differentiation matrices
    %D1=legslbdiff(N,x);    % 1st order differentiation matrices
    D2=D1*D1;               % 2nd order differentiation matrices
    D=(-D2(2:N-1,2:N-1)+alpha*eye(N-2)); % coefficient matrix
    b=f(2:N-1);             % RHS
    un=D\b;
    un=[0;un;0];            % Solve the system
    %error=norm(abs(un-u),inf);  % maximum pointwise error
    error=norm(abs(un-u),2);     % L^2 error
    Errv=[Errv;error];
end;
plot(Nvec,log10(Errv),'ro-','MarkerFaceColor','w','LineWidth',1.5)
grid on, xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
title('L^{2} error of Legendre-collocation method','fontsize',12)
set(gca,'fontsize',12)

print -dpng -r600  LegenCM1_error.png
```

```
% LegenCM2_error.m
% Legendre-collocation Method for the model equation:
% -u''(x)+u'(x)+u(x)=f(x),  x in (-1,1);
% % boundary condition: u(-1)=u(1)=0;
% exact solution: u=sin(kw*pi*x);
% RHS: f(x)=kw^2*pi^2*sin(kw*pi*x)+sin(kw*pi*x)+kw*pi*cos(kw*pi*x);
% Rmk: Use routines lepoly(); legslb(); legslbdm();
clear all;  clf
kw=10;
Nv=[32:2:68];
Errv=[];
for N=Nv
    [xv,wv]=legslb(N);     % compute LGL nodes and weights
    u=sin(kw*pi*xv);       % test function
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv)+kw*pi*cos(kw*pi*xv);  % RHS
    % Setup and solve the collocation system
    D1=legslbdm(N);        %1st order differentiation matrix
    D2=D1*D1;              % 2nd order differentiation matrix
    D=-D2(2:N-1,2:N-1)+D1(2:N-1,2:N-1)+eye(N-2);    % coefficient matrix
    b=f(2:N-1);   % RHS
    un=D\b;
    un=[0;un;0];  % Solve the system

    error=norm(abs(un-u),inf);
    Errv=[Errv;error];
end
% Plot the maximum pointwise error
plot(Nv,log10(Errv),'md-','MarkerFaceColor','w','LineWidth',1.5)
grid on,
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
title('L^{\infty} error of Legendre-collocation method','fontsize',12)

print -dpng -r600  LegenCM2_error.png
```

Example 4.4 The two-point boundary value problem:

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

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

```
% LegenCM3_error.m
% Legendre-collocation Method for the model equation:
% -u''(y)+u(y)=f(y) in [0,1] with boundary condition: u(0)=1, u'(1)=0;
% test function : u(y)=(1-y)^2*exp(y);
% RHS : f(y)=(2-4*y)*exp(y);
% Rmk: Use routines legslb(); legslbdiff();
clear all;  clf
Nvec=4:18;
Errv=[]; condnv=[];
for N=Nvec
    xv=legslb(N);            % compute LGL nodes and weights
    yv=1/2*(xv+1);           % variable substitution
    u=(1-yv).^2.*exp(yv);    % test solution in [0,1]
    f=(2-4*yv).*exp(yv);     % RHS in [0,1]

    % Setup and solve the collocation system
    D1=legslbdiff(N,xv);     % 1st order differentiation matrices
    D2=D1*D1;                % 2nd order differentiation matrices
    D=-4*D2+eye(N);          % coefficient matrix
    D(1,:)=[1,zeros(1,N-1)];  D(N,:)=D1(N,:);
    b=[1; f(2:N-1); 0];         % RHS
    un=D\b;                     % Solve the system

    error=norm(abs(un-u),2);    % L^2 error
    Errv=[Errv;error];
    condnv=[condnv,cond(D)];
end
% Plot the L^2 error
plot(Nvec,log10(Errv),'s-','color',[0 0.5 0],'MarkerFaceColor','w','LineWidth',1.5)
grid on, xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),
title('L^2 error of Legendre-collocation method','fontsize',12)

print -dpng -r600  LegenCM3_error.png
```