微分方程数值解笔记

MATLAB 数值实现

田刘涛

2021 年 12 月

目 录

1 常微分方程数值解

1.1 一阶常微分方程初值问题

设 $f(t,u)$ 在区域 $\mathrm{G}: 0 \leqslant t \leqslant T, |u| < \infty$ 上连续, 求 $u=u(t)$ 满足

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

其中 u_0 是给定的初值, 这就是一阶常微分方程的初值问题. 为使问题 (1.1) 的解存在、唯一且连续依赖初值 u_0, 即初值问题 (1.1) 适定，还必须对右端 $f(t,u)$ 加适当限制，通常要求 f 关于 u 满足 Lipschitz 条件：存在常数 L, 使

$$|f(t,u_1) - f(t,u_2)| \leqslant L|u_1 - u_2|. \tag{1.2}$$

对所有 $t \in [0,T]$ 和 $u_1, u_2 \in (-\infty, +\infty)$ 成立.

1.2 Euler 法

最简单的数值解法是 Euler 法。将区间 $[0,T]$ 作 N 等分, 小区间的长度 $h = T/N$ 称为步长, 点列 $t_n = nh(n = 0,1,\cdots,N)$ 称为节点, $t_0 = 0$. 由已知初值 $u(t_0) = u_0$，可算出 $u(t)$ 在 $t = t_0$ 的导数值 $u'(t_0) = f(t_0, u(t_0)) = f(t_0, u_0)$. 利用 Taylor 展式

$$\begin{aligned} u\left(t_1\right) &= u\left(t_0+h\right) = u\left(t_0\right) + h u'\left(t_0\right) + \frac{h^2}{2} u''\left(t_0\right) + \frac{h^3}{6} u'''(\zeta) \\ &= u_0 + h f\left(t_0, u_0\right) + R_0, \end{aligned} \tag{1.3}$$

其中 $\zeta \in (t_0, t_1)$，并略去二阶小量 R_0, 得

$$u_1 = u_0 + hf(t_0, u_0).$$

u_1 就是 $u(t_1)$ 的近似值, 利用 u_1 又可算出 u_2, 如此下去可算出在所有节点的近似值.

Euler 法一般递推公式:

$$u_{n+1} = u_n + hf(t_n, u_n), \quad n = 0,1,\cdots,N-1. \tag{1.4}$$

现在用数值积分法推导 Euler 法. 将问题 (1.1) 写成等价的积分形式:

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))\mathrm{d}\tau, \quad (t_0 = 0), \tag{1.5}$$

特别地,

$$u\left(t_1\right) = u_0 + \int_{t_0}^{t_1} f(\tau, u(\tau))\mathrm{d}\tau, \quad (t_0 = 0). \tag{1.6}$$

用左矩形公式近似右端积分, 并用 u_1 代替 $u(t_1)$ 即得 $u_1 = u_0 + hf\left(t_0, u_0\right)$, 这就是 Euler 方法.

例 1.1

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}x} = t^2 + t - u, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases}$$

方程的真解：$u(x) = -e^{-t} + t^2 - t + 1.$

```
% Euler1.m
% Euler method for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
h=0.1;
x=0:h:1;                        % interval partition
N=length(x)-1;
u(1)=0;                         % initial value
fun=@(t,u) t.^2+t-u;            % RHS
for n=1:N
    u(n+1)=u(n)+h.*fun(x(n),u(n));
end
ue=-exp(-x)+x.^2-x+1;           % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1)
legend('Exact','Numerical','location','northwest')
% title('Euler method','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize',16), ylabel('u','fontsize',16)

% print -dpng -r600 Euler1.png
% print -depsc2 Euler1.eps
```

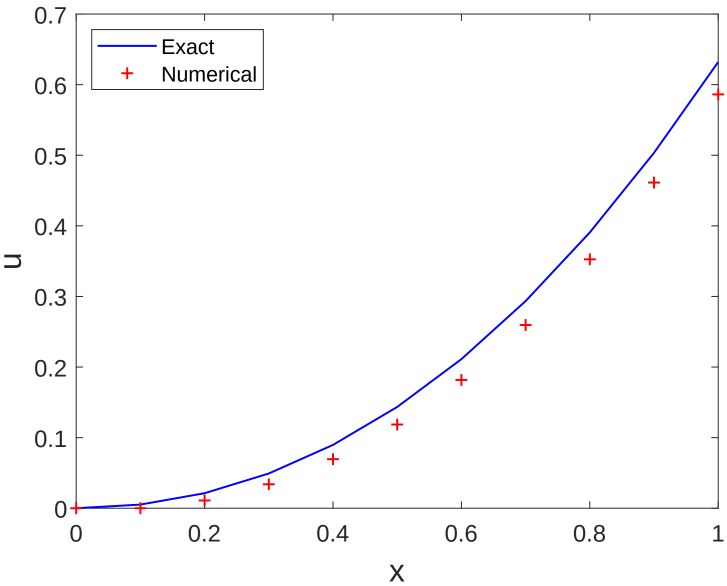


图 1: Euler 法

1.3 梯形方法

对于积分公式 (1.5), 用梯形公式近似右端积分, 用 u_1 替代 $u(t_1)$, 得

$$u_1 = u_0 + \frac{h}{2}\left[f\left(t_0, u_0\right) + f\left(t_1, u_1\right)\right], \tag{1.7}$$

一般的通式:

$$u_{n+1} = u_n + \frac{h}{2}\left[f\left(t_n, u_n\right) + f\left(t_{n+1}, u_{n+1}\right)\right], n = 0, 1, \cdots, N-1. \tag{1.8}$$

这就是**梯形方法**. 梯形公式是隐式方法, 每一步计算都要解一个非线性方程.

例 1.2

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = t^2 + t - u, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases}$$

方程的真解：$u(x) = -e^{-t} + t^2 - t + 1$. 这是一个线性常微分方程, 数值格式是显式的.

方程的离散格式：

$$u_{n+1} = u_n + \frac{h}{2}\left[t_n^2 + t_n - u_n + t_{n+1}^2 + t_{n+1} - u_{n+1}\right],$$

可得

$$u_{n+1} = \frac{2-h}{2+h}u_n + \frac{h}{2+h}\left[t_n^2 + t_n + t_{n+1}^2 + t_{n+1}\right].$$

```matlab
% Trapezoidal.m
% Trapezoidal rule for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
h=0.1;
x=0:h:1;                    % interval partition
N=length(x)-1;
u(1)=0;                     % initial value
for n=1:N
    u(n+1)=(2-h)/(2+h).*u(n)+h/(2+h).*(x(n)^2+x(n)+x(n+1)^2+x(n+1));
end
ue=-exp(-x)+x.^2-x+1;       % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1)
legend('Exact','Numerical','location','northwest')
%title('Trapezoidal rule','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize',16), ylabel('u','fontsize',16)

% print -dpng -r600 Trapezoidal.png
% print -depsc2 Trapezoidal.eps
```

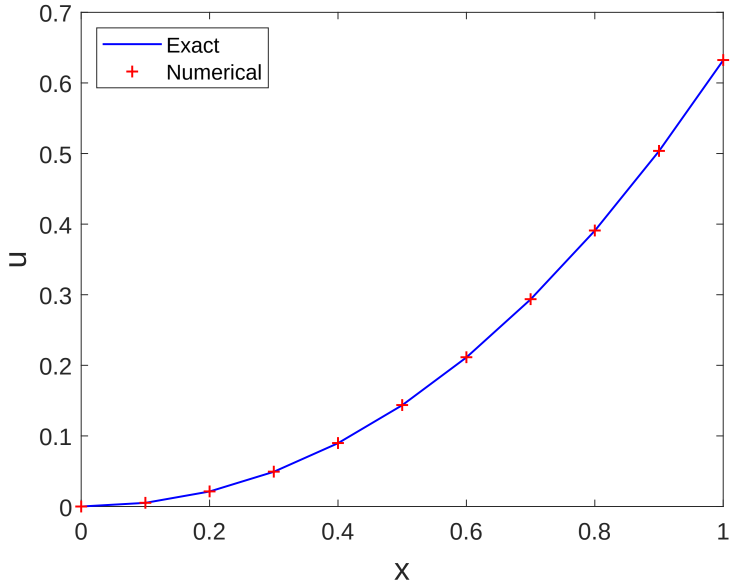


图 2: 梯形公式

1.4 后退 Euler 法

对于积分公式 (1.5), 用右矩形公式近似右端积分, 用 u_1 替代 $u(t_1)$,

$$u_1 = u_0 + hf\left(t_1, u_1\right), \tag{1.9}$$

一般的通式:

$$u_{n+1} = u_n + hf\left(t_{n+1}, u_{n+1}\right), \quad n = 0, 1, \cdots, N-1. \tag{1.10}$$

这就是 **后退 Euler 法**. 后退 Euler 法是隐式方法, 每一步计算都要解一个非线性方程.

例 1.3

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = t - \dfrac{t}{u}, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases}$$

方程的真解: $u(t) = \sqrt{2t+1}$. 这是一个非线性常微分方程, 数值格式是隐式的.

用后退 Euler (隐式 Euler) 格式离散

$$u_{n+1} = u_n + h\left(u_{n+1} - \frac{2t_{n+1}}{u_{n+1}}\right), \tag{1.11}$$

可转化为

$$u_{n+1} - u_n - hu_{n+1} + h\frac{2t_{n+1}}{u_{n+1}} = 0. \tag{1.12}$$

每一步要解一个关于 u_{n+1} 的非线性方程.

牛顿迭代: 假设 $f(x) = 0$ 有近似根 x_k ($f'\left(x_k\right) \neq 0$), 求 x_{k+1} 的方法

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots. \tag{1.13}$$

对于以上非线性微分方程的离散，记 $u_{n+1}=X$ (为了记号方便)

$$\begin{aligned} F(X) &= X - u_n - hX + \frac{2ht_{n+1}}{X}, \\ F'(X) &= 1 - h - \frac{2ht_{n+1}}{X^2}. \end{aligned} \tag{1.14}$$

数值格式

$$X_{k+1} = X_k - \frac{F(X_k)}{F'(X_k)}, \quad k = 0, 1, \cdots, N-1, \tag{1.15}$$

即

$$X_{k+1} = X_k - \frac{X_k - u_n - hX_k + \frac{2ht_{n+1}}{X_k}}{1 - h - \frac{2ht_{n+1}}{X_k^2}}, \quad k = 0, 1, \cdots, N-1. \tag{1.16}$$

设置终止迭代条件 $\|X_{k+1}-X_k\|<\delta$ (δ 为允许误差), 每一步计算都要解一个非线性方程.

```matlab
% BackEuler.m
% Backward Euler Method for Nonlinear ODE:
% u'(x)=u-2x/u in [0,1]
% Initial condition: u(0)=1;
% Exact solution: u=sqrt(2*x+1)
clear all; close all;
h=0.01;
x=0:h:1;
N=length(x)-1;
u(1)=1;
NI(1,N)=0;  % Record the number of iterations
for n=1:N
    % Newton iteration
    Xn=u(n);
    Xp=Xn;
    Xprev=0;
    while abs(Xp-Xprev) > eps*abs(Xp)
        Xprev=Xp;
        Xp=Xp-(Xp-h*Xp+2*h*x(n+1)/Xp-u(n))./(1-h-2*h*x(n+1)/(Xp^2));
        NI(n)=NI(n)+1;
    end
    u(n+1)=Xp;
end
ue=sqrt(2*x+1);    % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1)
legend('Exact ','Numerical','location','northwest')
% title('Backward Euler method','fontsize',12)
set(gca,'fontsize',12)
xlabel('t','fontsize',16), ylabel('u','fontsize',16)

% computing error
error=max(abs(u-ue))

% print -dpng -r600 BackEuler.png
% print -depsc2 BackEuler.eps
```

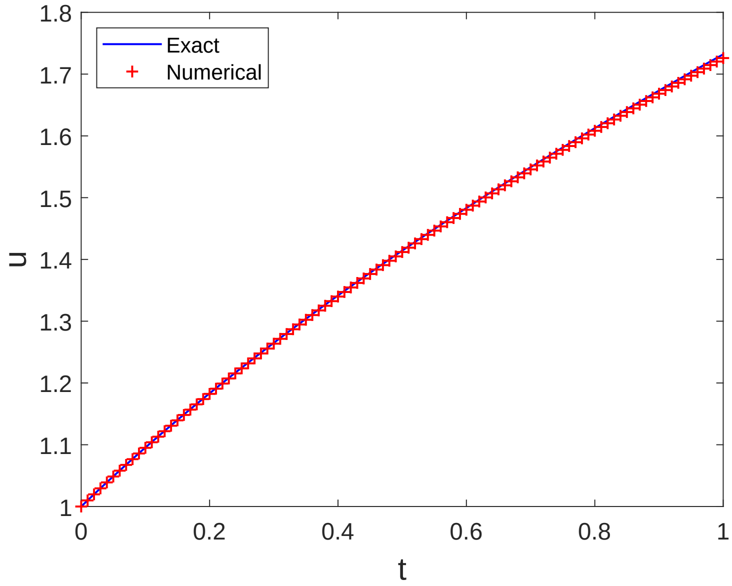


图 3: 后退 Euler 法

1.5 改进 Euler 方法

改进 Euler 方法 (Heun's method) 是一种预估矫正方法,

$$\begin{cases} \bar{u}_{n+1} = u_n + hf\left(x_n, u_n\right), \\ u_{n+1} = u_n + \dfrac{h}{2}\left[f\left(x_n, u_n\right) + f\left(x_{n+1}, \bar{u}_{n+1}\right]\right. \end{cases} \tag{1.17}$$

例 1.4

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = t^2 + t - u, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases} \tag{1.18}$$

方程的真解: $u(x) = -e^{-t} + t^2 - t + 1$.

```
% ModiEuler.m
% Modified Euler method for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
h=0.1;
x=0:h:1;                    % interval partition
N=length(x)-1;
u(1)=0;                     % initial value
fun=@(x,u) x.^2+x-u;        % RHS
for n=1:N
    k1=fun(x(n),u(n));
    k2=fun(x(n+1),u(n)+h*k1);
    u(n+1)=u(n)+(h/2)*(k1+k2);
end
```

```
ue=-exp(-x)+x.^2-x+1;        % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1)
legend('Exact','Numerical','location','northwest')
%title('Modified Euler Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize',16), ylabel('u','fontsize',16)

% print -dpng -r600 ModiEuler.png
% print -depsc2 ModiEuler.eps
```

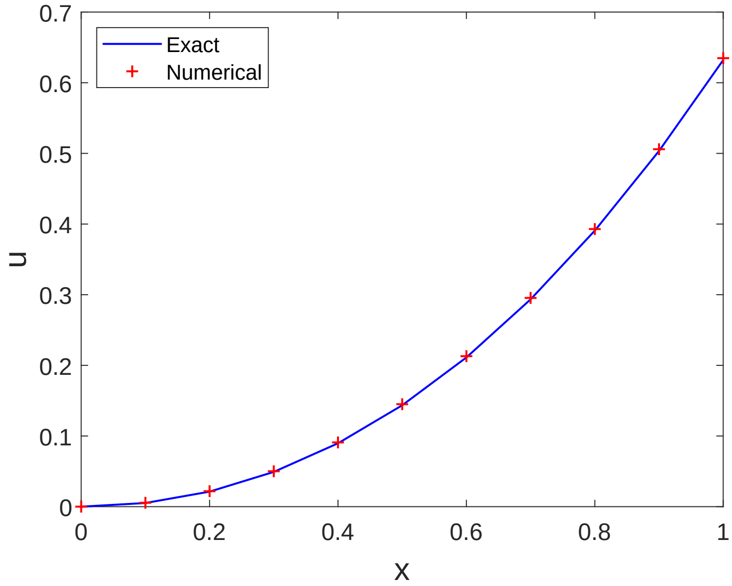


图 4: 改进 Euler 法

1.6 经典四阶 Runge-Kutta 方法

经典四阶 Runge-Kutta 方法

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

例 1.5

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = t^2 + t - u, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases} \tag{1.19}$$

方程的真解：$u(x) = -e^{-t} + t^2 - t + 1$.

```matlab
% RungeKutta.m
% Runge-Kutta method for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
h=0.1;
x=0:h:1;                   % interval partition
N=length(x)-1;
u(1)=0;                    % initial value
fun=@(x,u) x.^2+x-u;       % RHS
for n=1:N
    k1=fun(x(n),u(n));
    k2=fun(x(n)+h./2,u(n)+h.*k1/2);
    k3=fun(x(n)+h./2,u(n)+h.*k2/2);
    k4=fun(x(n)+h,u(n)+h.*k3);
    u(n+1)=u(n)+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)
legend('Exact','Numerical','location','northwest')
% title('Runge-Kutta Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize',16), ylabel('u','fontsize',16)

% print -dpng -r600 RungeKutta.png
% print -depsc2 RungeKutta.eps
```

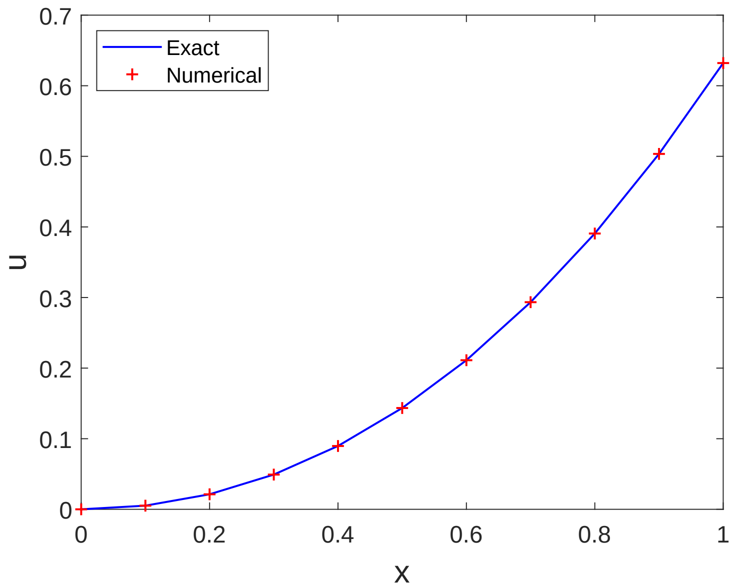


图 5: 经典四阶 Runge-Kutta 方法

1.7 计算数值方法收敛阶

数值解法的基本思想是，通过某种离散化手段将微分方程转化为差分方程，如单步法

$$u_{n+1} = u_n + h\varphi(x_n, u_n, h). \tag{1.20}$$

它在 x_n 处的数值解为 u_n, 而初值问题在 x_n 处的精确解为 $u(x_n)$, 记 $e_n = u(x_n) - u_n$ 称为整体截断误差. 收敛性就是讨论当 $x = x_n$ 固定且 $h = \frac{x_n - x_0}{n} \to 0$ 时 $e_n \to 0$ 的问题.

定义 1.1 (收敛性) 若一种数值方法(如单步法对于固定的 $x = x_n + nh$, 当 $h \to 0$ 时有 $u_n \to u(x_n)$, 其中 $u(x)$ 是初值问题的准确解，则称该方法是**收敛**的。

定理 1.1 (收敛性) 假设单步法具有 p 阶精度, 且增量函数 $\varphi(x_n, u_n, h)$ 关于 u 满足 Lipschitz 条件

$$|\varphi(x, u, h) - \varphi(x, \bar{u}, h)| \leqslant L_\varphi |u - \bar{u}|,$$

又设初值 u_0 是准确的, 即 $u_0 = u(x_0)$, 则其**整体截断误差**

$$u(x_n) - u_n = O(h^p).$$

数值方法误差阶的计算

设数值方法的误差阶

$$\|u(x_n) - u_n\|_* = Ch^p + O\left(h^{p+1}\right).$$

省略 h^p 的高阶无穷小，记误差

$$E = Ch^p.$$

两边都取对数

$$\ln(E) = p\ln(h) + C.$$

p 为 $\ln(E)$ 关于 $\ln(h)$ 的斜率

$$p = \frac{\Delta E}{\Delta h} = \frac{\ln(E_1) - \ln(E_2)}{\ln(h_1) - \ln(h_2)} = \frac{\ln(E_1/E_2)}{\ln(h_1/h_2)}.$$

如果函数区间为 $[a, b]$, 等距剖分为 N 等份, $h = \frac{b-a}{N}$, 则

$$p = -\frac{\ln(E_1/E_2)}{\ln(N_1/N_2)}.$$

线性 ODE 例子

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = t^2 + t - u, \quad 0 < t \leqslant 1, \\ u(0) = 0. \end{cases}$$

方程的真解：$u(x) = e^t + t^2 - t + 1$.

```matlab
% Euler1_error.m
% Euler method for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
Nvec=[10 50 100 500 1000];         % Number of partitions
Error=[];
fun=@(x,u) x.^2+x-u;               % RHS
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
    x=0:h:1;                       % interval division
    u(1)=0;                        % initial value
    for n=1:N
        u(n+1)=u(n)+h.*fun(x(n),u(n));
    end
    ue=-exp(-x)+x.^2-x+1;          % exact solution
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
%loglog(Nvec,Error,'ro-','LineWidth',1.5)
hold on
%loglog(Nvec, Nvec.^(-1), '--')
plot(log10(Nvec), log10(Nvec.^(-1)), '--')
grid on
%title('Convergence of Euler method','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)

% add annotation of slope
ax = [0.57 0.53];
ay = [0.68 0.63];
annotation('textarrow',ax,ay,'String','slope = -1 ','fontsize',14)

% computating convergence order
for n=1:length(Nvec)-1
    order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
end
Error
order

% print -dpng -r600 Euler1_error.png
% print -depsc2 Euler1_error.eps
```

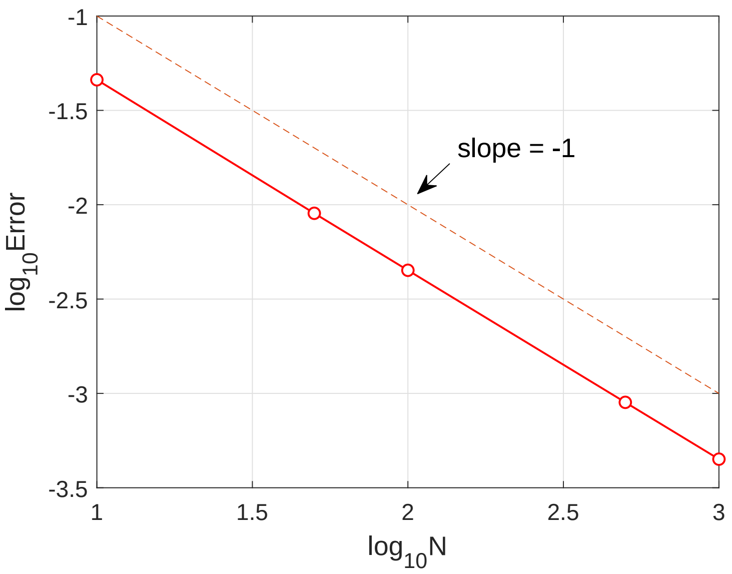


图 6: Euler 法收敛速度

```
% Trapezoidal_error.m
% Trapezoidal rule for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all;  clf
Nvec=[10 50 100 500 1000];     % Number of partitions
Error=[];
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
x=0:h:1;                       % interval division
N=length(x)-1;
u(1)=0;                        % initial value
for n=1:N
    u(n+1)=(2-h)/(2+h).*u(n)+h/(2+h).*(x(n)^2+x(n)+x(n+1)^2+x(n+1));
end
    ue=-exp(-x)+x.^2-x+1;      % exact solution
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
%loglog(Nvec,Error,'ro-','LineWidth',1.5)
hold on
%loglog(Nvec, Nvec.^(-2), '--')
plot(log10(Nvec), log10(Nvec.^(-2)), '--')
grid on
%title('Convergence of Trapezoidal rule','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)

% add annotation of slope
```

```matlab
ax = [0.63 0.58];
ay = [0.70 0.65];
annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)

% computating convergence order
for n=1:length(Nvec)-1
    order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
end
Error
order

% print -dpng -r600 Trapezoidal_error.png
% print -depsc2 Trapezoidal_error.eps
```

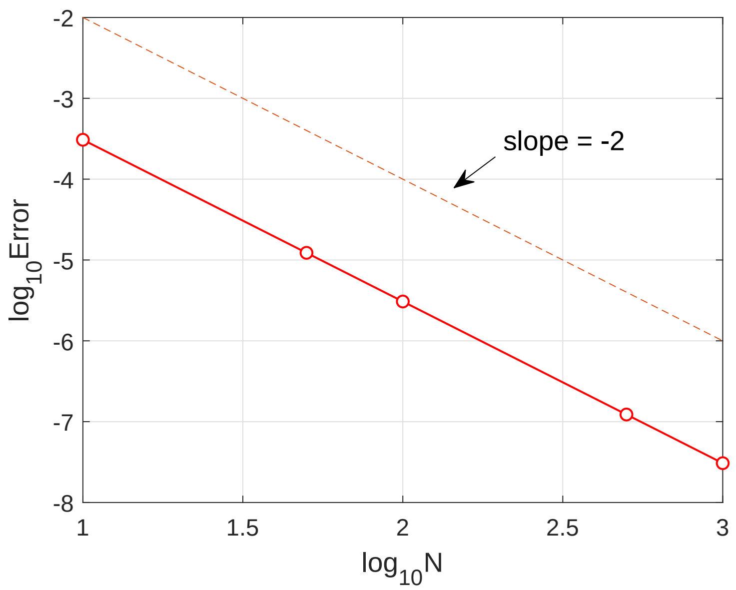


图 7: 梯形公式收敛速度

```matlab
% ModiEuler_error.m
% Modified Euler method for the ODE model
% u'(x)=x^2+x-u, x in [0,1]
% Initial condition: u(0)=0 ;
% Exact solution: u(x)=-exp(-x)+x^2-x+1.
clear all; close all;
Nvec=[10 50 100 500 1000];         % Number of partitions
Error=[];
fun=@(x,u) x.^2+x-u;               % RHS
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
    x=0:h:1;                       % interval division
    u(1)=0;                        % initial value
    for n=1:N
        k1=fun(x(n),u(n));
        k2=fun(x(n+1),u(n)+h*k1);
```

```matlab
        u(n+1)=u(n)+(h/2)*(k1+k2);
    end
    ue=-exp(-x)+x.^2-x+1;          % exact solution
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
%loglog(Nvec,Error,'ro-','LineWidth',1.5)
hold on
%loglog(Nvec, Nvec.^(-2), '--')
plot(log10(Nvec),log10(Nvec.^(-2)),'--')
grid on
%title('Convergence of Trapezoidal rule','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)

% add annotation of slope
ax = [0.58 0.53];
ay = [0.68 0.63];
annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)

% computating convergence order
for n=1:length(Nvec)-1
    order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
end
Error
order

% print -dpng -r600 ModiEuler_error.png
% print -depsc2 ModiEuler_error.eps
```

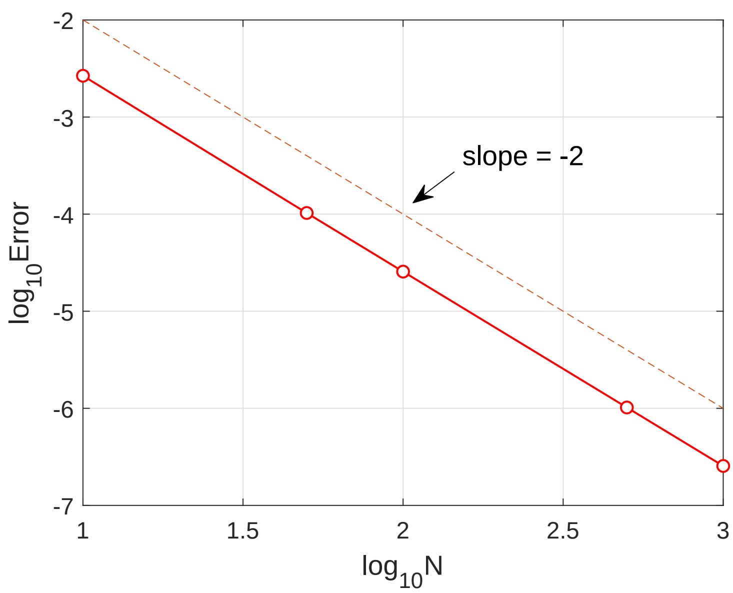


图 8: 改进 Euler 法收敛速度

```
% RungeKutta_error.m
% Runge-Kutta method for the ODE model
% u'(t)=t^2+t-u, t \in [0,2]
% Initial value : u(0)=0 ;
% Exact solution : u(t)=-exp(-t)+t^2-t+1.
clear all; close all;
Nvec=[10 50 100 500 1000];          % Number of partitions
Error=[];
fun=@(t,u) t.^2+t-u;                % RHS
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
    x=0:h:1;                        % interval division
    u(1)=0;                         % initial value
    for n=1:N
        k1=fun(x(n),u(n));
        k2=fun(x(n)+h./2,u(n)+h.*k1/2);
        k3=fun(x(n)+h./2,u(n)+h.*k2/2);
        k4=fun(x(n)+h,u(n)+h.*k3);
        u(n+1)=u(n)+h.*(k1+2.*k2+2.*k3+k4)./6;
    end
    ue=-exp(-x)+x.^2-x+1;           % exact solution
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(log10(Nvec),log10(Nvec.^(-4)), '--')
grid on
%title('Convergence of Runge-Kutta Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',16), ylabel('log_{10}Error','fontsize',16)

% add annotation of slope
ax = [0.57 0.53];
ay = [0.68 0.63];
annotation('textarrow',ax,ay,'String','slope = -4 ','fontsize',14)

% computating convergence order
for n=1:length(Nvec)-1
    order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
end
Error
order

% print -dpng -r600 RungeKutta_error.png
% print -depsc2 RungeKutta_error.eps
```

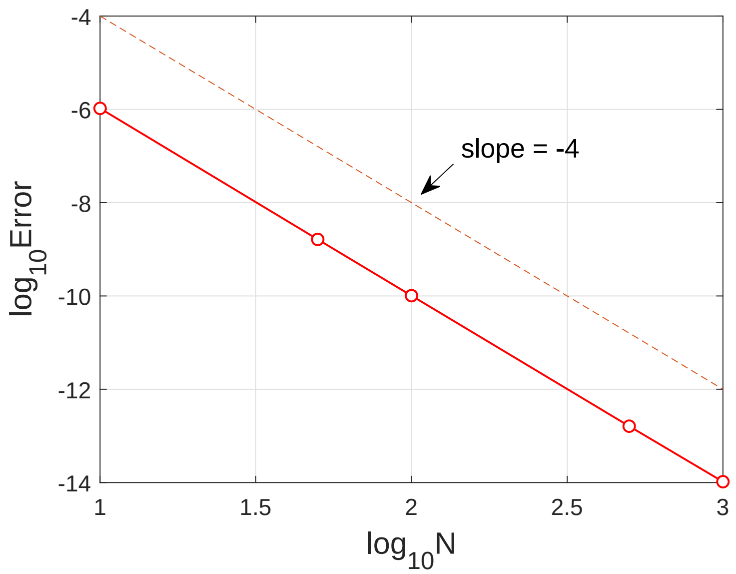


图 9: Runge-Kutta 法收敛速度

1.8 隐式 Runge-Kutta 方法

对于常微分方程

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

一般 s 级 Runge-Kutta 方法：

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

对应的 Butcher 阵：

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

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

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

形式 (I) 与形式 (II) 等价, 可以通过插值 $k_i = f\left(x_n + c_i h, Y_i\right), i = 1, 2, \ldots, s$ 证明.

Gauss Method 2 级 4 阶

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

对应的 Butcher 阵：

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

方程组的牛顿迭代

$$\begin{cases} f_1\left(x_1, x_2, \cdots, x_n\right) = 0, \\ f_2\left(x_1, x_2, \cdots, x_n\right) = 0, \\ \quad \vdots \\ f_n\left(x_1, x_2, \cdots, x_n\right) = 0, \end{cases}$$

记 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$, $\boldsymbol{F} = (f_1, f_2, \cdots, f_n)^{\mathrm{T}}$, 方程组可写为

$$\boldsymbol{F}(\boldsymbol{x}) = \mathbf{0}.$$

方程组的牛顿迭代法

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{F}'(\boldsymbol{x}^{(k)})^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right), \quad k = 0, 1, \cdots,$$

这里 $\boldsymbol{F}'(\boldsymbol{x})$ 是 Jacobi 矩阵.

例 1.6

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = f(t, u), \quad 0 < t \leqslant 1, \\ u(0) = 1. \end{cases} \tag{1.21}$$

其中：$f(t,u) = u$. 方程的真解：$u(x) = e^x$.

```
% 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; close all;
Nvec=[10 50 100 200 500 1000];
Error=[];
for n=1:length(Nvec)
    N=Nvec(n);
    h=1/N;
    x=[0:h:1];
```

```
    u(1)=1;
    Y=[1;1];
    % Newton iteration
    for i=1:N
        k=u(i);  tol=1;
        while tol>1.0e-10
            X=Y;
            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
            Y=X-D\F;
            tol=norm(Y-X);
        end
        u(i+1)=k+(h/2)*(Y(1)+Y(2));
    end
    ue=exp(x);               % exact solution
    error=max(abs(u-ue));   % maximum error
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on,
plot(log10(Nvec), log10(Nvec.^(-4)), '--')
grid on,
% title('Convergence order of Gauss method ','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)

% add annotation of slope
ax = [0.62 0.58];
ay = [0.72 0.66];
annotation('textarrow',ax,ay,'String','slope = -4 ','fontsize',14)

% computating convergence order
for i=1:length(Nvec)-1     % computating convergence order
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 IRK2s_error.png
% print -depsc2 IRK2s_error.eps
```

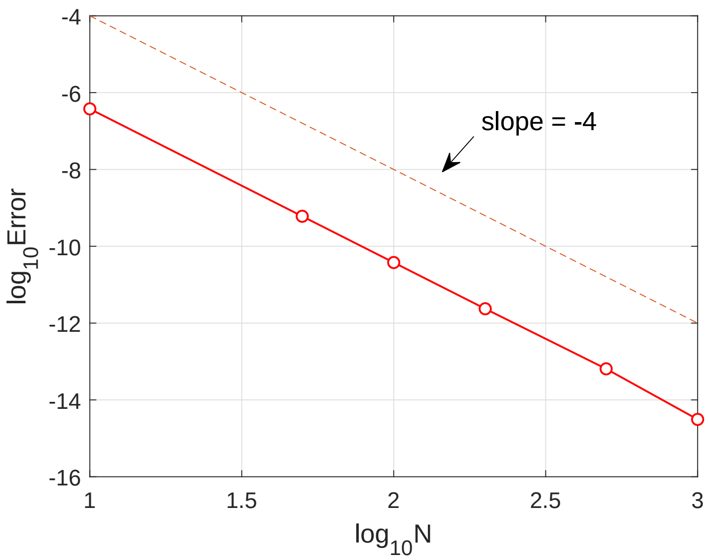


图 10: Gauss Method 关于 (1.21) 的收敛速度

例 1.7

$$\begin{cases} \dfrac{\mathrm{d}u}{\mathrm{d}t} = f(t,u), \quad 0 < t \leqslant 1,, \\ u(0) = 0. \end{cases} \tag{1.22}$$

其中：$f(t,u) = \dfrac{u^2}{2} + \dfrac{1}{2}$. 方程的真解：$u(x) = \dfrac{1-e^x}{1+e^x}$.

```
% IRK2s2_error.m
% Implicit Runge-Kutta(Gauss method) 2 stage and order 4
% u'=u^2/2-1/2 in [0,1] with initial condition u(0)=0
% exact solution: ue=(1-exp(x))/(1+exp(x))
clear all; close all;
Nvec=[10 50 100 200 500];
Error=[];
for n=1:length(Nvec)
    N=Nvec(n);
    h=1/N;
    x=[0:h:1];
    u(1)=0;
    Y=[1;1];
    % Newton iteration
    for i=1:N
        k=u(i);  tol=1;
        while tol>1.0e-10
            X=Y;
            D=[1-h*0.25*X(1), -h*(0.25-(sqrt(3))/6)*X(2);...
                -h*(0.25+(sqrt(3))/6)*X(1), 1-h*0.25*X(2)];
            F=[X(1)-k-h*(0.25*(0.5*(X(1))^2-0.5)+(0.25-(sqrt(3))/6)*(0.5*(X
               (2))^2-0.5));...
                X(2)-k-h*((0.25+(sqrt(3))/6)*(0.5*(X(1))^2-0.5)+0.25*(0.5*(X
                   (2))^2-0.5))];
            Y=X-D\F;
```

```
            tol=norm(Y-X);
        end
        u(i+1)=k+(h/2)*(0.5*Y(1)^2-0.5+0.5*Y(2)^2-0.5);
    end
    ue=(1-exp(x))./(1+exp(x));  % exact solution
    error=max(abs(u-ue));       % maximum error
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on,
plot(log10(Nvec),log10(Nvec.^(-4)),'--')
grid on,
% title('Convergence order of Gauss method ','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)

% add annotation of slope
ax = [0.62 0.58];
ay = [0.72 0.66];
annotation('textarrow',ax,ay,'String','slope = -4 ','fontsize',14)

% computing convergence order
for i=1:length(Nvec)-1     % computing convergence order
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 IRK2s2_error.png
% print -depsc2 IRK2s2_error.eps
```

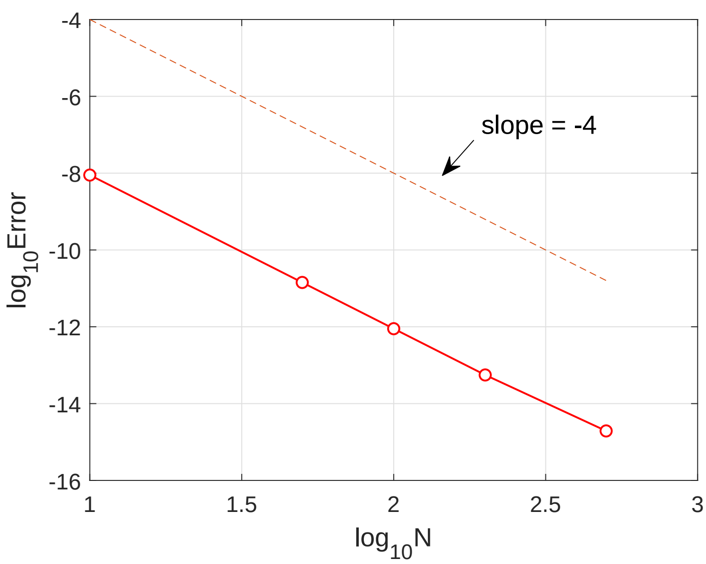


图 **11:** Gauss Method 关于 (1.22) 的收敛速度

2 有限差分方法

在偏微分方程的数值解法中, 有限差分法数学概念直观, 推导自然, 是发展较早且比较成熟的数值方法. 由于计算机只能存储有限个数据和做有限次运算, 所以任何一种用计算机解题的方法, 都必须把连续问题 (微分方程的边值问题、初值问题等) 离散化, 最终化成有限形式的线性代数方程组.

2.1 一维差分方法

二阶 BVP (常系数)

考虑二阶常微分方程边值问题 (常系数):

$$\begin{cases} Lu = -\dfrac{\mathrm{d}^2 u}{\mathrm{d}x^2} + \dfrac{\mathrm{d}u}{\mathrm{d}x} + qu = f, \quad a < x < b, \\ u(a) = \alpha, \quad u(b) = \beta. \end{cases} \tag{2.1}$$

其中 q, f 为 $[a, b]$ 上的连续函数, $q \geqslant 0$; α, β 为给定常数. 这是最简单的椭圆方程第一边值问题.

将区间 $[a, b]$ 分成 N 等分, 节点为

$$x_i = a + ih \quad i = 0, 1, \cdots, N,$$

其中 $h = (b-a)/N$. 于是得到区间 $I = [a, b]$ 的一个网格剖分. x_i 称为网格的**节点**, h 称为**步长**.

差分方程:

$$L_h u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \frac{u_{i+1} - u_{i-1}}{h} + q_i u_i = f_i, \quad 1 \leqslant j \leqslant N-1.$$

其中 $q_i = q(x_i)$, $f_i = f(x_i)$.

以上差分方程对于 $i = 1, 2, \cdots, N-1$ 都成立, 加上边值条件 $u_0 = \alpha$, $u_N = \beta$, 就得到关于 u_i 的差分格式:

$$\begin{aligned} &L_h u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \frac{u_{i+1} - u_{i-1}}{2h} + q_i u_i = f_i, \quad i = 1, 2, \cdots, N-1, \\ &u_0 = \alpha, \quad u_N = \beta. \end{aligned}$$

它的解 u_i 是 $u(x)$ 在 $x = x_i$ 处的差分解.

先定义向量 $\boldsymbol{u}$:

$$\boldsymbol{u} = (u_1, u_2, \cdots, u_{N-1})^{\mathrm{T}}.$$

差分格式可以写为矩阵形式:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}.$$

其中矩阵 $\boldsymbol{A}$、向量 $\boldsymbol{f}$ 的定义如下, 注意向量 $\boldsymbol{f}$ 的首尾元素已包含 $x=a$ 和 $x=b$ 处的边界条件.

$$\boldsymbol{A}=\begin{bmatrix} \frac{2}{h^2}+q_1 & \frac{1}{2h}-\frac{1}{h^2} & & & \\ -\frac{1}{2h}-\frac{1}{h^2} & \frac{2}{h^2}+q_2 & \frac{1}{2h}-\frac{1}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2h}-\frac{1}{h^2} & \frac{2}{h^2}+q_{N-2} & \frac{1}{2h}-\frac{1}{h^2} \\ & & & -\frac{1}{2h}-\frac{1}{h^2} & \frac{2}{h^2}+q_{N-1} \end{bmatrix},$$

$$\boldsymbol{f}=(f_1+\frac{\alpha}{h^2}+\frac{\alpha}{2h}, f_2, \cdots, f_{N-1}+\frac{\beta}{h^2}-\frac{\beta}{2h})^{\mathrm{T}}.$$

例 2.1

$$\begin{cases} -\frac{\mathrm{d}^2u}{\mathrm{d}x^2}+\frac{\mathrm{d}u}{\mathrm{d}x}=\pi^2\sin(\pi x)+\pi\cos(\pi x), \quad 0<x<1, \\ u(0)=0, \;\; u(1)=0. \end{cases} \tag{2.2}$$

方程的真解: $u(x)=\sin(\pi x)$.

```
% 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; close 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)
Error=max(abs(u-ue))
legend('Exact ','Numerical','location','NorthEast')
%title('Finite Difference Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize', 16), ylabel('u','fontsize',16)

% print -dpng -r600 fdm1d1.png
% print -depsc2 fdm1d1.eps
```

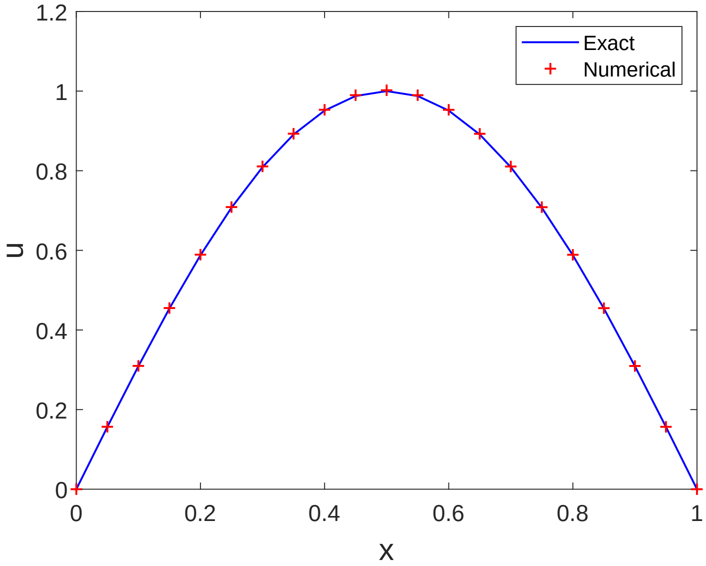


图 12: 一维差分格式

```
% fdm1d1_error.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; close all;
Nvec=[10 50 100 500 1000];    % Number of partitions
Error=[];
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
    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');
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(log10(Nvec), log10(Nvec.^(-2)), '--')
grid on
%title('Convergence of Finite Difference Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),

% add annotation of slope
```

```
ax = [0.55 0.52];
ay = [0.67 0.62];
annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)

% computating convergence order
for i=1:length(Nvec)-1
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 fdm1d1_error.png
% print -depsc2 fdm1d1_error.eps
```

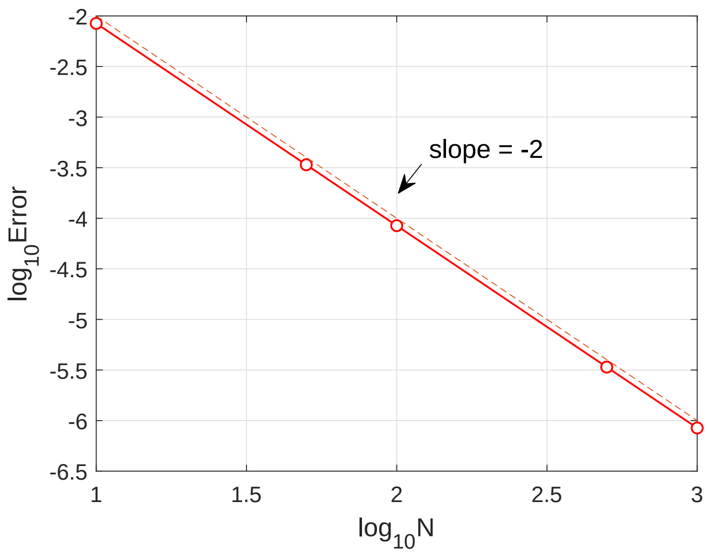


图 13: 一维差分格式收敛速度

二阶 BVP (变系数)

考虑二阶常微分方程边值问题 (变系数):

$$\begin{cases} Lu = -\dfrac{\mathrm{d}}{\mathrm{d}x}\left(p\dfrac{\mathrm{d}u}{\mathrm{d}x}\right) + r\dfrac{\mathrm{d}u}{\mathrm{d}x} + qu = f, \quad a < x < b, \\ u(a) = \alpha, \quad u(b) = \beta. \end{cases} \tag{2.3}$$

假定 $p \in C^1[a,b], p(x) \geqslant p_{\min} > 0, r, q, f \in C[a,b], \alpha, \beta$ 是给定的常数.

首先取 $N+1$ 个节点：$a = x_0 < x_1 < \cdots < x_i < \cdots < x_N = b$. 将区间 $I = [a,b]$ 分成 N 个小区间：

$$I_i : x_{i-1} \leqslant x \leqslant x_i, \quad i = 1, 2, \cdots, N.$$

记 $h_i = x_i - x_{i-1}$, 称 $h = \max\limits_i h_i$ 为最大网格步长.

取相邻节点 x_{i-1}, x_i 的中点 $x_{i-\frac{1}{2}} = \frac{1}{2}(x_{i-1} + x_i)$ $(i = 1, 2, \cdots, N)$ 称为半整数点, 则由节点

$$a = x_0 < x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{i-\frac{1}{2}} < \cdots < x_{N-\frac{1}{2}} < x_N = b.$$

又构成 $[a,b]$ 的一个剖分, 称为**对偶剖分**.

差分方程:

$$\begin{aligned}L_h u_i \equiv & -\frac{2}{h_i+h_{i+1}}\left[p_{i+\frac{1}{2}}\frac{u_{i+1}-u_i}{h_{i+1}}-p_{i-\frac{1}{2}}\frac{u_i-u_{i-1}}{h_i}\right]+\\ & \frac{r_i}{h_i+h_{i+1}}(u_{i+1}-u_{i-1})+q_i u_i=f_i, \quad i=1,\cdots,N-1,\\ u_0=\alpha,\ \ & u_N=\beta.\end{aligned}$$

当网格均匀, 即 $h=h_i(i=1,2,\cdots,N)$ 时, 差分格式:

$$\begin{aligned}L_h u_i = & -\frac{1}{h^2}\left[p_{i+\frac{1}{2}}u_{i+1}-\left(p_{i+\frac{1}{2}}+p_{i-\frac{1}{2}}\right)u_i+p_{i-\frac{1}{2}}u_{i-1}\right]+\\ & r_i\frac{u_{i+1}-u_{i-1}}{2h}+q_i u_i=f_i, \quad i=1,\cdots,N-1,\\ u_0=\alpha,\ \ & u_N=\beta.\end{aligned}$$

例 2.2

$$\begin{cases}-\dfrac{\mathrm{d}}{\mathrm{d}x}\left(x\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)+x\dfrac{\mathrm{d}u}{\mathrm{d}x}=\pi^2 x\sin(\pi x)+\pi(x-1)\cos(\pi x), \ \ 0<x1,\\ u(0)=0,\ \ u(1)=0.\end{cases} \tag{2.4}$$

真解: $u(x)=\sin(\pi x)$.

```
% 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; close 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)
Error=max(abs(u-ue))
legend('Exact ','Numerical','location','NorthEast')
%title('Finite Difference Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('x','fontsize', 16), ylabel('u','fontsize',16)

% print -dpng -r600 fdm1d2.png
% print -depsc2 fdm1d2.eps
```

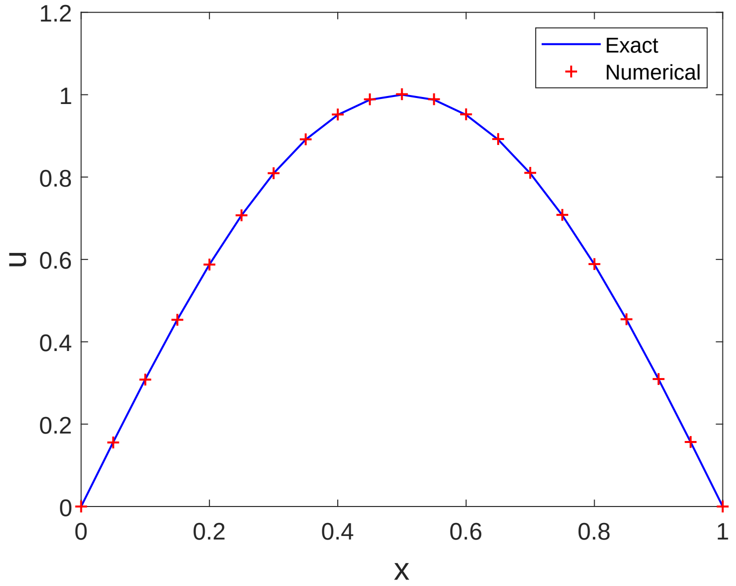


图 14: 一维差分格式 (例 2.2)

```
% fdm1d2_error.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; close all;
Nvec=[10 20 50 100 200 500 1000];     % Number of partitions
Error=[];
for k=1:length(Nvec)
    N=Nvec(k);
    h=1/N;
    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');
    error=max(abs(u-ue));
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(log10(Nvec), log10(Nvec.^(-2)), '--')
grid on
%title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)
xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),

% add annotation of slope
```

```
ax = [0.57 0.53];
ay = [0.68 0.63];
annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)

% computating convergence order
for i=1:length(Nvec)-1
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 fdm1d2_error.png
% print -depsc2 fdm1d2_error.eps
```

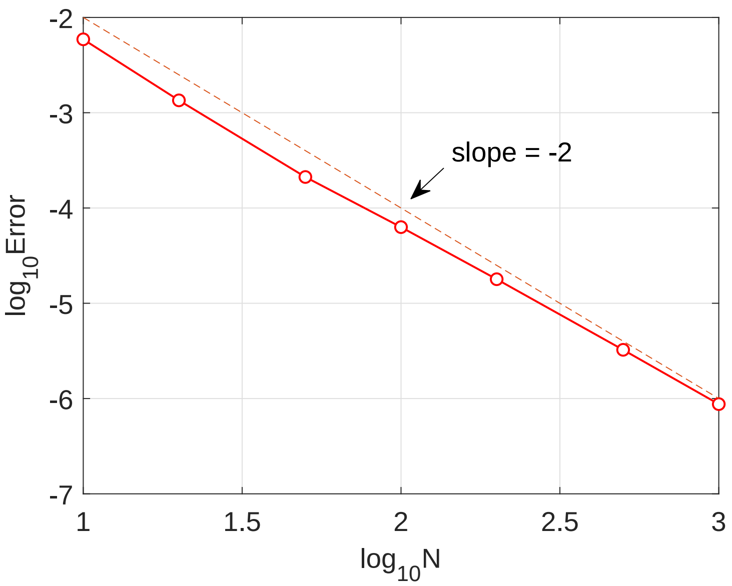


图 15: 一维差分格式收敛速度 (例 2.2)

2.2 二维矩形网的差分格式

二维 Possion 方程五点差分格式的 MATLAB 编程实现.

考虑二维 Possion 方程:

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

其中 $\partial\Omega$ 为区域 Ω 的边界, $f(x,y)$ 和 $\phi(x)$ 为已知函数, $u|_{\partial\Omega}=\phi(x,y)$ 为边界条件.

五点差分格式

考虑 Ω 为矩形的情况, 即 $a<x<b$, $c<x<d$. 取定沿 x 轴和 y 轴的步长 h_1 和 h_2, $h_1=(b-a)/N, h_2=(d-c)/M$. 则 $x_i=a+ih_1,\quad 0\leqslant i\leqslant N$, $y_j=c+jh_2,\quad 0\leqslant j\leqslant M$. (x_i,y_j) 称为**网格节点**.

二维 Possion 方程的五点差分格式:

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

先定义向量 $\boldsymbol{u}_j=(u_{1j},u_{2j},\ldots,u_{N-1,j})^{\mathrm{T}},\quad 0\leqslant j\leqslant M.$

差分格式可以写为矩阵形式:

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

其中矩阵 $\boldsymbol{C}$、$\boldsymbol{D}$、向量 $\boldsymbol{f}_j$ 的定义如下, 注意向量 $\boldsymbol{f}_j$ 的首尾元素已包含了 $x=a$ 和 $x=b$ 处的边界条件.

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

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

以上矩阵形式的差分格式还可以进一步写为如下的矩阵形式, 注意等号右边向量的首尾元素加入了 $y=c$ 和 $y=d$ 处的边界条件.

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

例 2.3

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

其中: $f(x,y)=-2\pi^2 e^{\pi(x+y)}\left(\sin(\pi x)\cos(\pi y)+\cos(\pi x)\sin(\pi y)\right)$.
真解: $u(x,y)=e^{\pi(x+y)}\sin(\pi x)\sin(\pi y), \quad (x,y)\in\Omega=(0,1)\times(0,1)$.

```
% 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; close all;
% generate coordinates on the grid
h=0.02;
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)
%title('Finite Difference Method','fontsize',14)
set(gca,'fontsize',12)
xlabel('x','fontsize', 16)
ylabel('y','fontsize',16)
zlabel('u','fontsize',16)

% print -dpng -r600 fdm2d1.png
% print -depsc2 fdm2d1.eps
```

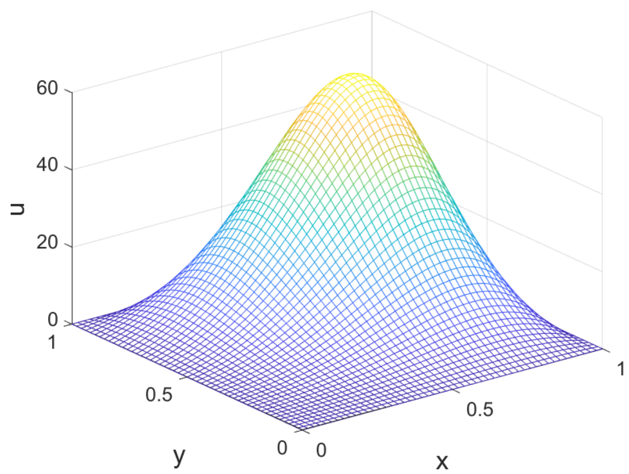


图 16: 矩形网差分格式

```
% 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; close all;
Nvec=2.^[4:10];
Error=[];
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+pi*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);
    error=max(max(abs(u-ue)));     % maximum error
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(log10(Nvec), log10(Nvec.^(-2)), '--')
grid on
%title('Convergence of Finite Difference Method','fontsize',12)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),

% add annotation of slope
ax = [0.46 0.50];
ay = [0.41 0.46];
annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)

% computating convergence order
for i=1:length(Nvec)-1
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 fdm2d1_error.png
% print -depsc2 fdm2d1_error.eps
```

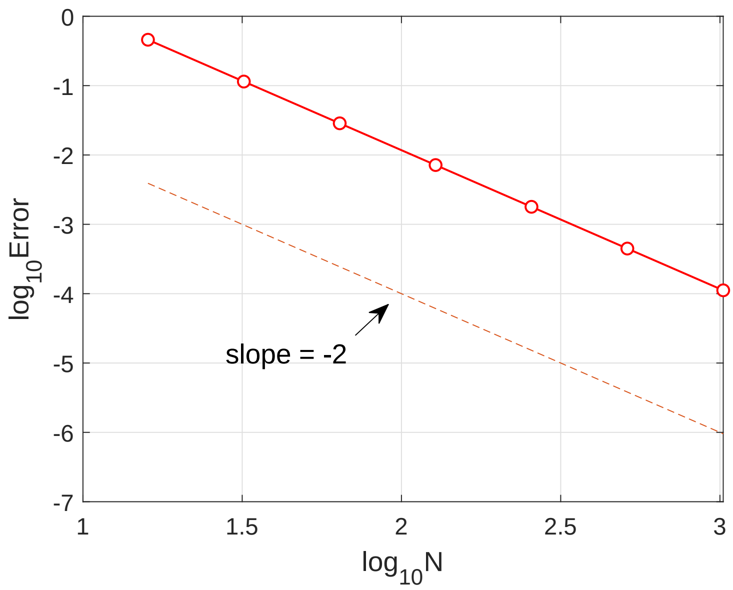


图 17: 矩形网差分格式收敛速度

例 2.4

$$\begin{cases} -\Delta u=\cos 3x\sin\pi y, & (x,y)\in G=(0,\pi)\times(0,1)\\ u(x,0)=u(x,1)=0, & 0\leqslant x\leqslant\pi,\\ u_x(0,y)=u_x(\pi,y)=0, & 0\leqslant y\leqslant 1\end{cases}\tag{2.5}$$

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

对应《微分方程数值解法》(李荣华) 104-105 页数值例子.

以步长 $h_1=\frac{\pi}{N}, h_2=\frac{1}{N}$ 作矩形剖分, 网格节点为 $x_i=ih_1, y_j=jh_2, i,j=0,1,\cdots,N$.

差分方程：

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

边界条件：

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

离散格式：

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

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

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

矩阵形式:

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

最后再利用边界条件处理边界处的值.

```
% fdm2d2.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;
N=4    % N=4 8 16 32
% coordinates on the grid
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));

format long
% value of u and ue in the selected points (i*pi/4,j/4), i,j=1,2,3.
u_select=u(N/4+1:N/4:3*N/4+1,N/4+1:N/4:3*N/4+1)
ue_select=u(N/4+1:N/4:3*N/4+1,N/4+1:N/4:3*N/4+1)
```

对应书上结果:

```
N =
     4
u_select =
  -0.045480502664596  -0.000000000000000   0.045480502664596
  -0.064319143691817  -0.000000000000000   0.064319143691817
  -0.045480502664596  -0.000000000000000   0.045480502664596
ue_select =
  -0.045480502664596  -0.000000000000000   0.045480502664596
  -0.064319143691817  -0.000000000000000   0.064319143691817
  -0.045480502664596  -0.000000000000000   0.045480502664596
```

```
% 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];
Error=[];
for N=Nvec
    % coordinates on the grid
    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));
    error=max(max(abs(u-ue)));   % maximum error
    Error=[Error,error];
end
plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(log10(Nvec), log10(Nvec.^(-1)), '--')
grid on
%title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)
xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),

% add annotation of slope
ax = [0.64 0.60];
ay = [0.69 0.64];
annotation('textarrow',ax,ay,'String','slope = -1 ','fontsize',14)

% computating convergence order
for i=1:length(Nvec)-1
    order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
Error
order

% print -dpng -r600 fdm2d2_error.png
% print -depsc2 fdm2d2_error.eps
```

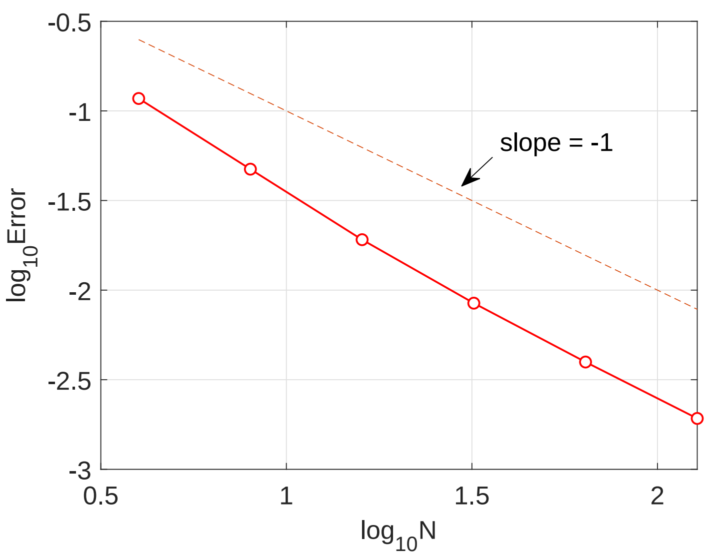


图 18: 矩形网差分格式收敛速度 (例 (2.4))

2.3 一维热传导方程的差分格式

一维热传导方程差分方法的 MATLAB 编程实现.

考虑一维热传导方程:

$$
\begin{cases}
\dfrac{\partial u}{\partial t} = a\dfrac{\partial^2 u}{\partial x^2} + f(x,t), & 0 < x < 1,\ 0 < t \leqslant T, \\
u(x,0) = \phi(x), & 0 \leqslant x \leqslant 1, \\
u(0,t) = \alpha(t), u(1,t) = \beta(t), & 0 < t \leqslant T.
\end{cases}
$$

其中 a 是正常数, $f(x,t), \phi(x), \alpha(x)$ 和 $\beta(x)$ 为已知函数. $u(x,0) = \phi(x)$ 为初始条件, $u(0,t) = \alpha(t)$ 和 $u(1,t) = \beta(t)$ 为边界条件.

向前差分格式

以空间步长 $h = 1/M$, 时间步长 $\tau = T/N$ 分别将 x 轴上区间 $[0,1]$, t 轴上区间 $[0,T]$ 分成 M、N 等分, 可得

$$
\begin{aligned}
x_i &= jh, \quad 0 \leqslant j \leqslant M, \\
t_n &= n\tau, \quad 0 \leqslant n \leqslant N.
\end{aligned}
$$

一维热传导方程的向前差分格式

$$
\frac{u_j^{n+1} - u_j^n}{\tau} = a\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} + f(x_j, t_n),
$$

$$
u_j^0 = \phi_j = \phi\left(x_j\right), u_0^n = u_M^n = 0,
$$

其中 $j = 1, 2, \cdots, M-1, n = 0, 1, \cdots, N-1$. 以 $r = a\tau/h^2$ 表示 ** 网比 **.

以上格式可改写差分格式

$$
u_j^{n+1} = ru_{j+1}^n + (1-2r)u_j^n + ru_{j-1}^n + \tau f_j,
$$

$$
1 \leqslant j \leqslant M-1,\ 0 \leqslant n \leqslant N-1.
$$

先取 $n = 0$, 利用 u_j^0 和边值 $u_0^n = u_M^n = 0$ 算出第一层值 u_j^1, 再取 $n = 2$, 利用 u_j^1 和边值便可算出 u_j^2. 如此下去, 便可求出所有 u_j^n.

设 $\boldsymbol{u}^n = (u_1^n, u_2^n, \cdots, u_{M-2}^n, u_{M-1}^n)^{\mathrm{T}}, \quad 0 \leqslant n \leqslant N.$

差分格式写成矩阵的形式：

$$
\boldsymbol{u}^{n+1} = \boldsymbol{A}\boldsymbol{u}^n + \boldsymbol{f}^n, \quad 0 \leqslant n \leqslant N-1.
$$

其中矩阵 $\boldsymbol{A}$、向量 $\boldsymbol{f}^n$ 的定义如下, 注意向量 $\boldsymbol{f}^n$ 的首尾元素已包含了边界条件.

$$\boldsymbol{A}=\begin{pmatrix} 1-2r & r & & & \\ r & 1-2r & r & & \\ & \ddots & \ddots & \ddots & \\ & & r & 1-2r & r \\ & & & r & 1-2r \end{pmatrix},$$

$$\boldsymbol{f}^n=\begin{pmatrix} \tau f\left(x_1,t_n\right)+ru_0^n \\ \tau f\left(x_2,t_n\right) \\ \vdots \\ \tau f\left(x_{M-2},t_n\right) \\ \tau f\left(x_{M-1},t_n\right)+ru_M^n \end{pmatrix}.$$

例 2.5

$$\begin{cases} \dfrac{\partial u}{\partial t}=\dfrac{\partial^2 u}{\partial x^2}, \quad 0<x<1, \quad 0<t\leqslant 1, \\ u(x,0)=\mathrm{e}^x, \quad 0\leqslant x\leqslant 1, \\ u(0,t)=\mathrm{e}^t, \quad u(1,t)=\mathrm{e}^{1+t}, \quad 0<t\leqslant 1. \end{cases}$$

方程的真解：$u(x,t)=\mathrm{e}^{x+t}$.

```matlab
% fdm_heat.m
% forward difference scheme for heat equation
% u_t=u_{xx}, (x,t) in (0,1)x(0,1],
% u(x,0)=exp(x), x in [0,1],
% u(0,t)=exp(t), u(1,t)=exp(1+t), t in (0,1]
% exact solution: u(x,t)=exp(x+t)
clear all; close all;
a=1;
h=0.05; x=[0:h:1];
tau=0.00125; t=[0:tau:1];
r=a*tau/h^2;
M=length(x)-1; N=length(t)-1;
% constructing the coefficient matrix
e=r*ones(M-1,1);
A=spdiags([e 1-2*e e],[-1 0 1],M-1,M-1);
% setting initial and boundary conditions
u=zeros(M+1,N+1);
u(:,1)=exp(x);
u(1,:)=exp(t);
u(end,:)=exp(1+t);
for n=1:N
    u(2:M,n+1)=A*u(2:M,n);
    u(2,n+1)=u(2,n+1)+r*u(1,n);
    u(M,n+1)=u(M,n+1)+r*u(end,n);
end
```

```
% plot the figure
mesh(t(1:20:end),x,u(:,1:20:end))
set(gca,'fontsize',12)
xlabel('t','fontsize', 14)
ylabel('x','fontsize',14)
zlabel('u','fontsize',14)

% calculating maximum error
[T X]=meshgrid(t,x);
ue=exp(X+T);
Error=max(max(abs(ue-u)))

% print -dpng -r600 fdm_heat.png
% print -depsc2 fdm_heat.eps
```

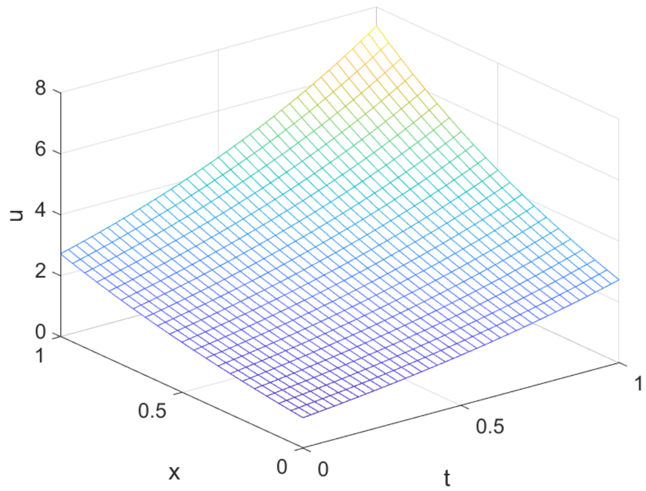


图 19: 一维热传导方程的差分格式

2.4 一维热传导方程 CN 格式

一维热传导方程 Crank-Nicolson 格式的 MATLAB 编程实现.

考虑一维热传导方程:

$$\begin{cases} \dfrac{\partial u}{\partial t} = a\dfrac{\partial^2 u}{\partial x^2} + f(x,t), & 0 < x < 1,\ 0 < t \leqslant T, \\ u(x,0) = \phi(x), & 0 \leqslant x \leqslant 1, \\ u(0,t) = \alpha(t), u(1,t) = \beta(t), & 0 < t \leqslant T. \end{cases}$$

其中 a 是正常数, $f(x,t), \phi(x), \alpha(x)$ 和 $\beta(x)$ 为已知函数. $u(x,0) = \phi(x)$ 为初始条件, $u(0,t) = \alpha(t)$ 和 $u(1,t) = \beta(t)$ 为边界条件.

Crank-Nicolson 格式

以空间步长 $h = 1/M$, 时间步长 $\tau = T/N$ 分别将 x 轴上区间 $[0,1]$, t 轴上区间 $[0,T]$ 分成 M, N 等分, 可得

$$x_i = jh, \quad 0 \leqslant j \leqslant M,$$
$$t_n = n\tau, \quad 0 \leqslant n \leqslant N.$$

一维热传导方程的六点对称格式 (Crank-Nicolson 格式)

$$\frac{u_j^{n+1} - u_j^n}{\tau} = \frac{a}{2}\left[\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}\right] + \frac{f(x_j,t_{n+1}) + f(x_j,t_n)}{2},$$

$$u_j^0 = \phi_j = \phi(x_j), \quad u_0^n = u_M^n = 0.$$

以上格式可改写为

$$-\frac{r}{2}u_{j+1}^{n+1} + (1+r)u_j^{n+1} - \frac{r}{2}u_{j-1}^{n+1} = \frac{r}{2}u_{j+1}^n + (1-r)u_j^n + \frac{r}{2}u_{j-1}^n + \frac{\tau}{2}\left[f(x_j,t_{n+1}) + f(x_j,t_n)\right].$$

$$1 \leqslant j \leqslant M-1,\ 0 \leqslant n \leqslant N-1.$$

即

$$-ru_{j+1}^{n+1} + (2+2r)u_j^{n+1} - ru_{j-1}^{n+1} = ru_{j+1}^n + (2-2r)u_j^n + ru_{j-1}^n + \tau f(x_j,t_{n+1}) + \tau f(x_j,t_n).$$

$$1 \leqslant j \leqslant M-1,\ 0 \leqslant n \leqslant N-1.$$

利用 u_j^0 和边值便可逐层求到 u. 六点对称格式是隐格式, 由第 n 层计算第 $n+1$ 层时, 需解线性代数方程组（因系数矩阵严格对角占优, 方程组可唯一求解）.

设 $\boldsymbol{u}^n = (u_1^n, u_2^n, \cdots, u_{M-2}^n, u_{M-1}^n)^{\mathrm{T}}, \quad 0 \leqslant n \leqslant N.$

差分格式可以写为矩阵形式：

$$\boldsymbol{A}\boldsymbol{u}^{n+1} = \boldsymbol{B}\boldsymbol{u}^n + \boldsymbol{f}^n, \quad 0 \leqslant n \leqslant N-1.$$

其中矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$、向量 $\boldsymbol{f}^n$ 的定义如下, 注意向量 $\boldsymbol{f}^n$ 的首尾元素已包含了边界条件.

$$\boldsymbol{A} = \begin{pmatrix} 2+2r & -r & & & \\ -r & 2+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 2+2r & -r \\ & & & -r & 2+2r \end{pmatrix},$$

$$\boldsymbol{B} = \begin{pmatrix} 2-2r & r & & & \\ r & 2-2r & r & & \\ & \ddots & \ddots & \ddots & \\ & & r & 2-2r & r \\ & & & r & 2-2r \end{pmatrix},$$

$$\boldsymbol{f}^n = \begin{pmatrix} \tau f(x_1,t_n) + \tau f(x_1,t_{n+1}) + ru_0^n + ru_0^{n+1} \\ \tau f(x_2,t_n) + \tau f(x_2,t_{n+1}) \\ \vdots \\ \tau f(x_{M-2},t_n) + \tau f(x_{M-2},t_{n+1}) \\ \tau f(x_{M-1},t_n) + \tau f(x_{M-1},t_{n+1}) + ru_M^n + ru_M^{n+1} \end{pmatrix}.$$

例 2.6

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} + f(x,t), \quad 0 < x < 1, \quad 0 < t \leqslant 1, \\ u(x,0) = \sin(x), \quad 0 \leqslant x \leqslant 1, \\ u(0,t) = \sin(t), \quad u(1,t) = \sin(1+t), \quad 0 < t \leqslant 1. \end{cases}$$

其中 $f(x,t) = \cos(x+t) + \sin(x+t)$.

方程的真解 $u(x,t) = \sin(x+t)$.

```
% fdm_cn.m
% Crank-Nicolson scheme for heat equation
% u_t=u_{xx}+f(x,t), (x,t) in (0,1)x(0,1],
% u(x,0)=sin(x), x in [0,1],
% u(0,t)=sin(t), u(1,t)=sin(1+t), t in (0,1].
% f(x,t)=cos(x+t)+sin(x+t),
% exact solution: u(x,t)=sin(x+t)
clear all; close all;
a=1;
h=0.05; x=[0:h:1];
tau=0.001; t=[0:tau:1];
r=a*tau/h^2;
M=length(x)-1; N=length(t)-1;
% constructing the coefficient matrix
e=r*ones(M-1,1);
A=spdiags([-e 2+2*e -e],[-1 0 1],M-1,M-1);
B=spdiags([e 2-2*e e],[-1 0 1],M-1,M-1);
```

```
% setting initial and boundary conditions
u=zeros(M+1,N+1);
u(:,1)=sin(x);
u(1,:)=sin(t);
u(end,:)=sin(1+t);
for n=1:N
    F=tau*cos(x(2:M)'+t(n))+tau*sin(x(2:M)'+t(n))...
        +tau*cos(x(2:M)'+t(n+1))+tau*sin(x(2:M)'+t(n+1));
    F(1)=F(1)+r*u(1,n)+r*u(1,n+1);
    F(M-1)=F(M-1)+r*u(end,n)+r*u(end,n+1);
    % solving the system
    u(2:M,n+1)=A\B*u(2:M,n)+A\F;
end
% plot the figure
mesh(t(1:20:end),x,u(:,1:20:end))
set(gca,'fontsize',12)
xlabel('t','fontsize', 14)
ylabel('x','fontsize',14)
zlabel('u','fontsize',14)

% calculating maximum error
[T X]=meshgrid(t,x);
ue=sin(X+T);

Error=max(max(abs(ue-u)))

% print -dpng -r600 fdm_cn.png
% print -depsc2 fdm_cn.eps
```

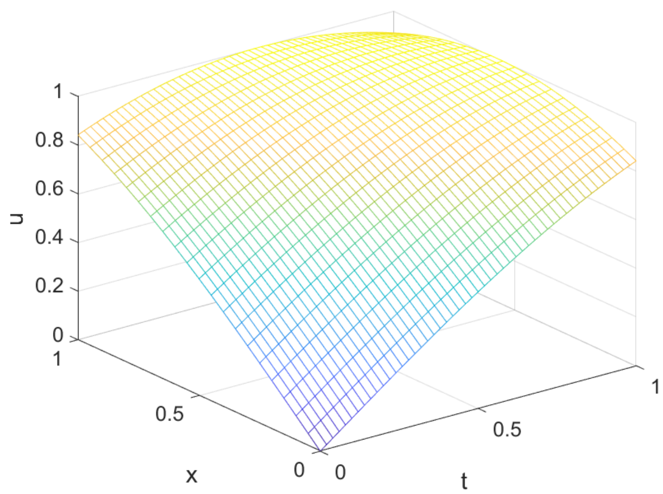


图 20: 一维热传导方程的 C-N 格式

2.5 一维波动方程的差分格式

一维波动方程差分方法的 MATLAB 编程实现.

考虑一维波动方程:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = a^2 \dfrac{\partial^2 u}{\partial x^2} + f(x,t), \quad 0 < x < 1,\ 0 < t \leqslant T, \\ u(0,t) = \alpha(t), \quad u(1,t) = \beta(t), \quad 0 < t \leqslant T, \\ u(x,0) = \phi(x), \quad \dfrac{\partial u(x,0)}{\partial t} = \psi(x), \quad 0 \leqslant x \leqslant 1. \end{cases}$$

其中 a 是正常数, $f(x,t)$、$\phi(x)$、$\psi(x)$、$\alpha(x)$ 和 $\beta(x)$ 为已知函数. $u(x,0) = \phi(x)$、$\frac{\partial u(x,0)}{\partial t} = \psi(x)$ 为初始条件, $u(0,t) = \alpha(t)$ 和 $u(1,t) = \beta(t)$ 为边界条件.

一维波动方程的差分格式

以空间步长 $h = 1/M$、时间步长 $\tau = T/N$ 分别将 x 轴上区间 $[0,1]$、t 轴上区间 $[0,T]$ 分成 M、N 等分, 可得

$$x_i = jh, \quad 0 \leqslant j \leqslant M,$$
$$t_n = n\tau, \quad 0 \leqslant n \leqslant N.$$

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\tau^2} = a^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} + f(x_j, t_n),$$

其中 $j = 1, 2, \cdots, M-1$, $n = 0, 1, \cdots, N-1$. 以 $r = a\tau/h$ 表示**网比**.

以上格式可改写差分格式

$$u_j^{n+1} = r^2(u_{j-1}^n + u_{j+1}^n) + 2(1-r^2)u_j^n - u_j^{n-1} + \tau^2 f(x_j, t_n),$$

$$1 \leqslant j \leqslant M-1,\ 1 \leqslant n \leqslant N-1.$$

根据 CFL 条件, 仅当步长比 $s \leqslant 1$ 时, 上式才是稳定的, 误差也会被控制在较小的程度.

设 $\boldsymbol{u}^n = (u_1^n, u_2^n, \cdots, u_{M-2}^n, u_{M-1}^n)^{\mathrm{T}}, \quad 0 \leqslant n \leqslant N.$

差分格式写成矩阵的形式:

$$\boldsymbol{u}^{n+1} = \boldsymbol{A}\boldsymbol{u}^n - \boldsymbol{u}^{n-1} + \boldsymbol{f}^n, \quad 1 \leqslant n \leqslant N-1.$$

其中矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$、向量 $\boldsymbol{f}^n$ 的定义如下, 注意向量 $\boldsymbol{f}^n$ 的首尾元素已包含了边界条件.

$$\boldsymbol{A} = \begin{pmatrix} 2(1-r^2) & r^2 & & & \\ r^2 & 2(1-r^2) & r^2 & & \\ & \ddots & \ddots & \ddots & \\ & & r^2 & 2(1-r^2) & r^2 \\ & & & r^2 & 2(1-r^2) \end{pmatrix},$$

$$
\boldsymbol{f}^n = \begin{pmatrix} \tau^2 f(x_1, t_n) + r^2 u_0^n \\ \tau^2 f(x_2, t_n) \\ \vdots \\ \tau^2 f(x_{M-2}, t_n) \\ \tau^2 f(x_{M-1}, t_n) + r^2 u_M^n \end{pmatrix}.
$$

注意：计算最开始时需要两个已知向量 $\boldsymbol{u}^0$ 和 $\boldsymbol{u}^1$, 第一个向量可直接通过初始条件得到, 即 $u_j^0 = \phi(x_j)$, 第二个向量通常选用 Euler 法来近似估算, 即 $u_j^1 = \phi(x_j) + \tau\psi(x_j)$.

例 2.7

$$
\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2} + (t^2 - x^2)\sin(xt), & 0 < x < 1,\ 0 < t \leqslant 1, \\ u(0,t) = 0, \quad u(1,t) = \sin t, & 0 < t \leqslant 1, \\ u(x,0) = 0, \quad \dfrac{\partial u(x,0)}{\partial t} = x, & 0 \leqslant x \leqslant 1. \end{cases}
$$

方程的真解：$u(x,t) = \sin(xt)$.

```
% fdm_wave.m
% finite difference method for wave equation
% u_{tt}=u_{xx}+f(x,t), (x,t) in (0,1)x(0,1],
% u(x,0)=0, u_t(x,0)=x, x in [0,1],
% u(0,t)=0, u(1,t)=sin(t), t in (0,1].
% f(x,t)=cos(x+t)+sin(x+t),
% exact solution: u(x,t)=sin(x+t)
clear all; close all;
a=1;
h=0.05; x=[0:h:1];
tau=0.05; t=[0:tau:1];
r=a*tau/h;
M=length(x)-1; N=length(t)-1;
[T X]=meshgrid(t,x);
% constructing the coefficient matrix
e=r^2*ones(M-1,1);
A=spdiags([e 2*(1-e) e],[-1 0 1],M-1,M-1);
% setting initial and boundary conditions
u=zeros(M+1,N+1);
u(:,1)=0; u(:,2)=tau*x;
u(1,:)=0; u(end,:)=sin(t);
for n=2:N
    u(2:M,n+1)=A*u(2:M,n)-u(2:M,n-1)+ ...
        tau^2*(T(2:M,n).^2-X(2:M,n).^2).*sin(X(2:M,n).*T(2:M,n));
    u(2,n+1)=u(2,n+1)+r^2*u(1,n);
    u(M,n+1)=u(M,n+1)+r^2*u(end,n);
end
% plot the figure
mesh(t,x,u), view(20,40)
set(gca,'fontsize',12)
```

```
xlabel('t','fontsize', 14)
ylabel('x','fontsize',14)
zlabel('u','fontsize',14)

% calculating maximum error
ue=sin(X.*T);
Error=max(max(abs(ue-u)))

% print -dpng -r600 fdm_wave.png
% print -depsc2 fdm_wave.eps
```

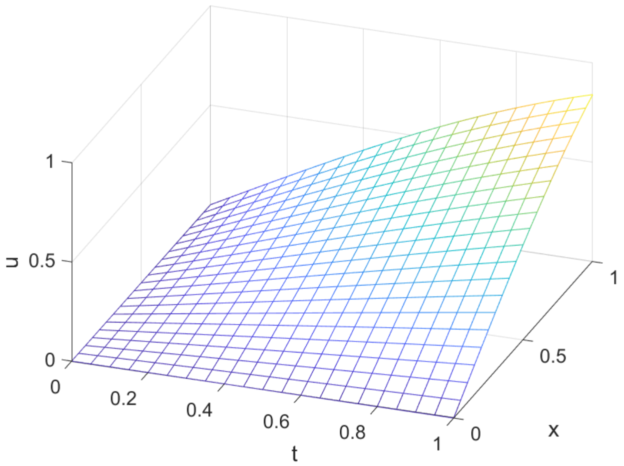


图 21: 一维波动方程的差分格式

例 2.8

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1, \quad 0 < t \leqslant T, \\ u(0,t) = u(1,t) = 0, \quad 0 < t \leqslant T, \\ u(x,0) = \sin(4\pi x), \ \dfrac{\partial u(x,0)}{\partial t} = \sin(8\pi x), \quad 0 < x < 1. \end{cases}$$

方程的真解：$u = \sin(4\pi x)\cos(4\pi t) + \sin(8\pi x)\sin(8\pi t)/(8\pi)$.

对应《微分方程数值解法》157-158 页数值例子.

```
% fdm_wave2.m
% finite difference method for wave equation
% u_{tt}=u_{xx}, (x,t) in (0,1)x(0,T],
% u(x,0)=0, u_t(x,0)=x, x in [0,1],
% u(0,t)=0, u(1,t)=sin(t), t in (0,1].
% exact solution: u(x,t)=sin(4*pi*x)*cos(4*pi*t)
%        +sin(8*pi*x)*sin(8*pi*t)/(8*pi);
clear all; close all;
a=1;
T=5;   % time
h=0.0025; x=[0:h:1];
tau=0.002; t=[0:tau:T];
```

```
r=a*tau/h;
M=length(x)-1; N=length(t)-1;
[T X]=meshgrid(t,x);
% constructing the coefficient matrix
e=r^2*ones(M-1,1);
A=spdiags([e 2*(1-e) e],[-1 0 1],M-1,M-1);
% setting initial and boundary conditions
u=zeros(M+1,N+1);
u(:,1)=sin(4*pi*x');
u(:,2)=sin(4*pi*x')+tau*sin(8*pi*x');
u(1,:)=0; u(end,:)=0;
for n=2:N
    u(2:M,n+1)=A*u(2:M,n)-u(2:M,n-1);
    u(2,n+1)=u(2,n+1)+r^2*u(1,n);
    u(M,n+1)=u(M,n+1)+r^2*u(end,n);
end
% plot the figure
mesh(t,x,u), view(20,40)
set(gca,'fontsize',12)
xlabel('t','fontsize', 14)
ylabel('x','fontsize',14)
zlabel('u','fontsize',14)

% calculating maximum error
ue=sin(4*pi*X).*cos(4*pi*T)+sin(8*pi*X).*sin(8*pi*T)/(8*pi);

disp('t = 1, 2, 3, 4, 5')
Error=max(abs(ue(:,1/tau+1:1/tau:end)-u(:,1/tau+1:1/tau:end)))

% print -dpng -r600 fdm_wave2.png
% print -depsc2 fdm_wave2.eps
```

时间 $T=1$；步长 $h=0.0025$、$\tau=0.0025$, 得到下图

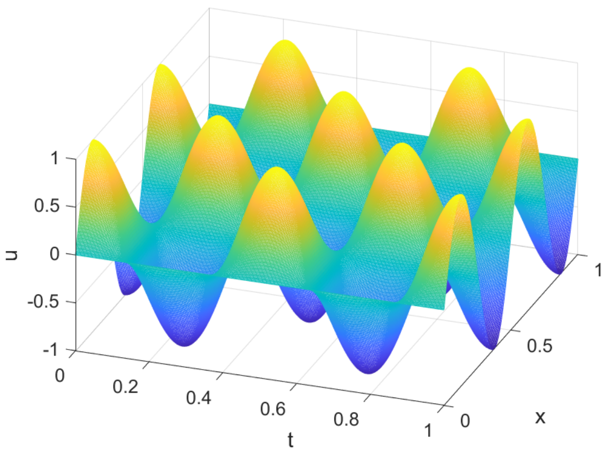


图 22: 一维波动方程的差分格式

2.6 极坐标形式的差分格式

极坐标形式的 Poisson 方程差分方法的 MATLAB 编程实现.

考虑极坐标形式的 Poisson 方程:

$$-\Delta_{r,\theta}u=-\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 u}{\partial\theta^2}\right]=f(r,\theta),$$

其中 $r=\sqrt{x^2+y^2},\ \tan\theta=y/x, r\theta$ 平面半带形域 $\{0\leqslant r<\infty,0\leqslant\theta\leqslant 2\pi\}$.

方程的系数于 $r=0$ 处奇异, 因此只当 $r>0$ 时有意义. 在 $r=0$ 需补充 u 为光滑的条件 (原点是可去奇点). 由 $u(0,\theta)=u(r,\theta)-ru_r(r,\theta)+O(r^2), r\to 0$, 则知

$$\lim_{r\to 0^+} r\frac{\partial u}{\partial r}=0.$$

差分格式

对变量 r,θ 分别取等步长 h_r 和 h_θ. 令

$$r_i=(i+0.5)h_r,\ i=0,1,2,\cdots,N-1,$$
$$\theta_j=(j+1)h_\theta,\ j=0,1,\cdots,M-1,\ h_\theta=2\pi/M.$$

则半带形域的网格节点为 (r_i,θ_j), 它们在 $r\theta$ 平面上的分布如图

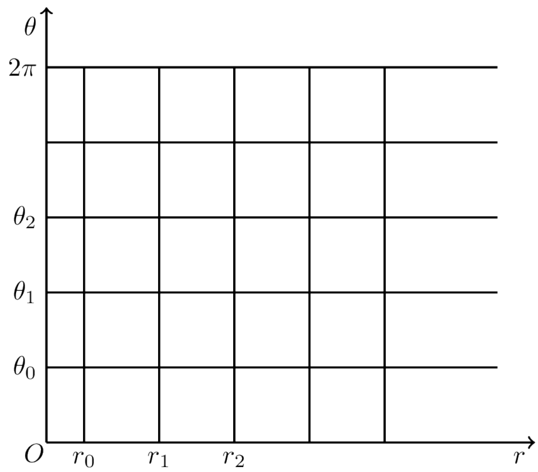


图 23: 半带形域的网格剖分

差分方程:

$$-\frac{1}{r_i}\frac{r_{i+\frac{1}{2}}u_{i+1,j}-\left(r_{i+\frac{1}{2}}+r_{i-\frac{1}{2}}\right)u_{ij}+r_{i-\frac{1}{2}}u_{i-1,j}}{h_r^2}$$
$$-\frac{1}{r_i^2}\frac{u_{i,j+1}-2u_{ij}+u_{i,j-1}}{h_\theta^2}=f\left(r_i,\theta_j\right),\quad 1\leqslant i\leqslant N-1.$$

下面是关于点 (r_0,θ_j) 的差分方程

$$-\frac{2}{h_r}\frac{u_{1j}-u_{0j}}{h_r}-\frac{4}{h_r^2}\frac{u_{0,j+1}-2u_{0j}+u_{0,j-1}}{h_\theta^2}=f_{0j}.$$

这样就得到 N $(i=0,1,\cdots,N-1)$ 个方程的方程组.

注意：实际计算时, 取 $(N+h_r/2)$ 为边界比较合适.

差分格式推导：

$$-\underbrace{\frac{1}{r_i^2}\frac{1}{h_\theta^2}}_{e_i}u_{i,j-1}-\underbrace{\frac{1}{r_i}\frac{r_{i-\frac12}}{h_r^2}}_{\beta_i}u_{i-1,j}+\underbrace{\frac{1}{r_i}\frac{r_{i+\frac12}+r_{i-\frac12}}{h_r^2}u_{ij}+\frac{1}{r_i^2}\frac{2}{h_\theta^2}u_{ij}}_{c_i}$$

$$-\underbrace{\frac{r_{i+\frac12}}{r_i h_r^2}}_{\alpha_i}u_{i+1,j}-\underbrace{\frac{1}{r_i^2}\frac{1}{h_\theta^2}}_{d_i}u_{i,j+1}=f_{ij}\qquad(i\geq 1).$$

$$-\underbrace{\frac{4}{h_r^2h_\theta^2}}_{e_0}u_{0,j-1}+\underbrace{\frac{2}{h_r^2}u_{0,j}+\frac{8}{h_r^2h_\theta^2}u_{0j}}_{c_0}-\underbrace{\frac{2}{h_r^2}}_{\alpha_0}u_{1j}-\underbrace{\frac{4}{h_r^2h_\theta^2}}_{d_0}u_{0,j+1}=f_{0j}.$$

$$C=\begin{pmatrix} c_0 & \alpha_0 & & & & \\ \beta_1 & c_1 & \alpha_1 & & & \\ & \beta_2 & c_2 & \alpha_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{N-2} & c_{N-2} & \alpha_{N-2} \\ & & & & \beta_{N-1} & c_{N-1} \end{pmatrix}\qquad D=\begin{pmatrix} d_0 & & & & & \\ & d_1 & & & & \\ & & d_2 & & & \\ & & & \ddots & & \\ & & & & d_{N-2} & \\ & & & & & d_{N-1} \end{pmatrix}$$

$$E=\begin{pmatrix} e_0 & & & & & \\ & e_1 & & & & \\ & & e_2 & & & \\ & & & \ddots & & \\ & & & & e_{N-2} & \\ & & & & & e_{N-1} \end{pmatrix}.\qquad f_j=\begin{pmatrix} f_{0j} \\ f_{1j} \\ \vdots \\ f_{N-2,j} \\ f_{N-1,j} \end{pmatrix}.$$

定义向量: $U_j=(u_{0j},u_{1j},\cdots,u_{N-1,j})^T$, $\quad 0\leq j\leq M-1$.

差分格式写成矩阵形式：

$$DU_{j-1}+CU_j+DU_{j+1}=f_j,\quad 0\leq j\leq M$$

其中 $U_i=U_{M+i}$. 定义 $U_{-1}=U_{M-1}$, $U_0=U_M$.

$$\begin{pmatrix} C & D & & & & E \\ E & C & D & & & \\ & E & C & D & & \\ & & \ddots & \ddots & \ddots & \\ & & & E & C & D \\ D & & & & E & C \end{pmatrix}\begin{pmatrix} U_0 \\ U_1 \\ U_2 \\ \vdots \\ U_{M-2} \\ U_{M-1} \end{pmatrix}=\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{M-2} \\ f_{M-1} \end{pmatrix}.$$

例 2.9 单位圆上的 Poisson 方程边值问题：

$$\begin{cases} -\Delta u = 1, \quad \Omega = \{(x,y) \mid x^2+y^2<1\}, \\ u|_{\partial\Omega} = 0. \end{cases}$$

方程的真解：

$$u(x,y) = \frac{(1-x^2-y^2)(x+y)}{4}.$$

```
% fdm_polar.m
% finite difference method for polar coordinate problem
% -{Delta}_{r,theta}u(r,theta)=f(r,theta), [0,1]x[0,2*pi]
% u(1,theta)=0,  theta in [0,2*pi]
% exact solution: ue=(1-r^2)*r*(sin(theta)+cos(theta))/4.
clear all; close all;
N=50;
M=100;
dr=1/(N+1/2);
% dr=0.02;
dthe=2*pi/M;
r=(dr/2:dr:1)';
the=(0:dthe:2*pi)';
% N=length(r)-1;
% M=length(the);
% generate coordinates on the grid
[The,R]=meshgrid(the,r);
% generate the matrix of RHS
The1=The(1:N,2:end); R1=R(1:N,2:end);
f=2*R1.*(sin(The1)+cos(The1));
f=f';
% constructing the coefficient matrix
e=[2/dr^2+8/(dr^2*dthe^2);(2./(dthe^2*r(2:N).^2)+2/dr^2)];
e1=[-2/dr^2;-(r(2:N-1)+dr/2)./(dr^2*r(2:N-1))];
e2=-(r(2:N)-dr/2)./(dr^2*r(2:N));
C=diag(e)+diag(e1,1)+diag(e2,-1);
D=diag([-4/(dr^2*dthe^2);-1./(dthe^2*r(2:N).^2)]);
E=diag([-4/(dr^2*dthe^2);-1./(dthe^2*r(2:N).^2)]);
A=kron(eye(M),C)+kron(diag(ones(M-1,1),1)+diag(1,1-M),D)...
    +kron(diag(ones(M-1,1),-1)+diag(1,M-1),E);
% solving the linear system
f=f';
% u=zeros(M+1,N+1);
uh=reshape(A\f(:),N,M);
un=[uh;zeros(1,M)];
un=[un(:,end),un];
ue=(1-R.^2).*R.*(sin(The)+cos(The))/4;
% error on the node of mesh
Err=abs(un(1:N,2:end)-ue(1:N,2:end));
% compute maximum error
MaxErr=max(max(abs(un-ue)))
```

```
% plot the figure
[X,Y] = pol2cart(The,R);
mesh(X,Y,ue)
set(gca,'fontsize',12)
xlabel('x','fontsize', 16)
ylabel('y','fontsize',16)
zlabel('u','fontsize',16)
view(36,24)

% print -dpng -r600 fdm_polar.png
% print -depsc2 fdm_polar.eps
```

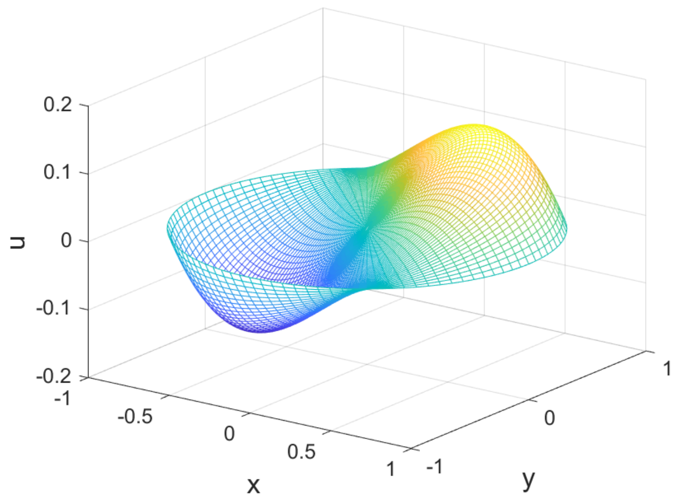


图 24: 极坐标方程的差分格式

3 有限元方法

有限元法的基本问题归纳为：

1. 把原问题 (偏微分方程初边值问题、边值问题) 转化为变分问题 (形式)；
2. 选定单元的形状，对求解区域进行剖分. 一维情形的单元是小区间，二维情形的单元主要有两种：三角形和四边形 (矩形、任意凸四边形). 至于三维情形，重要的单元有四面体、正六面体和三棱柱等；
3. 构造基函数或单元形状函数, 形成有限元空间；
4. 形成有限元方程组. 它是一个线性代数方程组，其系数矩阵还是稀疏的；
5. 选用适当的方法求解有限元方程.
6. 收敛性及误差估计.

3.1 一维有限元方法

模型问题与变分问题

考虑两点边值问题

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

其中 $p \in C^1[a,b], q, f \in C[a,b]$, 且 $p(x) \geqslant p_0 > 0, q(x) \geqslant 0, p_0$ 是一个常数.

定义函数空间

$$V = \left\{v \in H^1(I) \mid v(a) = 0\right\}.$$

原问题 (3.1) 的弱形式是找一个 $u \in H_0^1(I)$, 使得

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

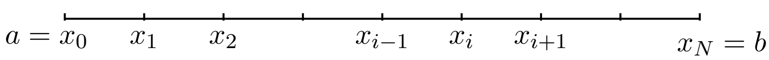
其中

$$a(u,v) = \int_a^b (p\frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x} + quv)\,\mathrm{d}x, \quad F(v) = \int_a^b fv\,\mathrm{d}x.$$

用有限元方法求解上述变分问题的 (3.2) 的具体计算步骤如下

1. 单元剖分

对求解区域 $[a,b]$ 进行网格剖分, 网格节点为 $x_i,\ 0 \leq i \leq N$,



有限元单元 $I_i = [x_{i-1}, x_i]$, 其长度 $h_i = x_i - x_{i-1}, (i = 1, 2, \cdots, N)$, 并记 $h = \max\limits_{1 \leqslant i \leqslant N}\{h_i\}$.

2. 试探函数空间的构造

考虑 $[a, b$ 上的分段多项式函数 v_h, 它满足如下条件:

(1) v_h 在 $[a, b]$ 上连续;
(2) 在每个单元 I_i 上是 k 次多项式 $(i = 1, 2, \cdots, N)$.

满足以上条件的函数构成如下集合:

$$V_h = \{\varphi \mid \varphi \in H^1(I) \text{ 且 } \varphi(I_i) \in \mathcal{P}_k(I_i)\}.$$

其中 $\mathcal{P}_k$ 是不超过 k 次的多项式集合.

这里假设有限元解 $u_h = V_h$ 是分片线性函数. 若设 u_h 在网格节点 $a = x_0, x_1, \cdots, x_{i-1}, x_i, \cdots, x_{n-1}, x_n = b$ 上分别取值为

$$u_0 = 0, u_1, \cdots, u_{i-1}, u_i, \cdots, u_{i-1}, \cdots, u_N = 0.$$

则 u_h 可表示成一次 Lagrange 插值函数的形式

$$u_h(x) = u_{i-1}\frac{x_i - x}{h_i} + u_i\frac{x - x_{i-1}}{h_i}, \quad x \in I_i,\ i = 1, \cdots, N. \tag{3.3}$$

上式中 $(x_i - x)/h_i$ 和 $(x - x_{i-1})/h_i$ 分别为分别称为相应于节点 x_{i-1} 和 x_i 的节点单元基函数.

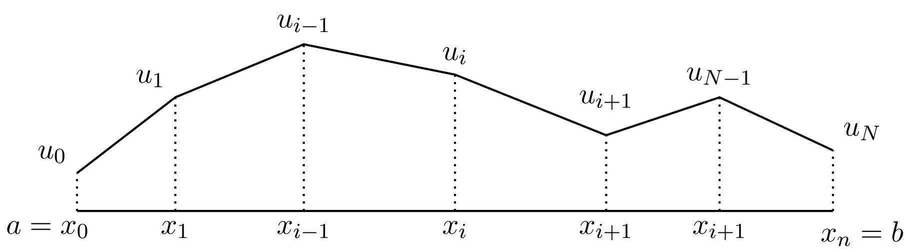


图 25: V_h 中函数的几何形状

在单元 I_i, I_{i+1} 考察线性插值公式 (3.3) 及 u_i 的系数, 对每一节点 x_i 构造出山形函数:

(i) 当 $i = 0$ 时,

$$\varphi_0(x) = \begin{cases} \dfrac{x_1 - x}{h_1}, & x \in I_1 = [x_0, x_1], \\ 0, & \text{其他}. \end{cases}$$

(ii) 当 $0 < i < N$ 时,

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i}, & x \in I_i = [x_{i-1}, x_i], \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & x \in I_{i+1} = [x_i, x_{i+1}], \\ 0, & \text{其他}. \end{cases}$$

(iii) 当 $i = N$ 时,

$$\varphi_N(x) = \begin{cases} \dfrac{x - x_{N-1}}{h_N}, & x \in I_N = [x_{N-1}, x_N], \\ 0, & \text{其他}. \end{cases}$$

不难验证

$$
\begin{aligned}
&\varphi_i\left(x_j\right)=\delta_{ij}=\begin{cases}1, & i=j,\\ 0, & i\neq j,\end{cases} \quad 1\leq i,j\leq n,\\
&\varphi_i(x)\cdot\varphi_j(x)=0, \quad |i-j|\geq 2,\\
&\varphi_i'(x)\cdot\varphi_j'(x)=0, \quad |i-j|\geq 2.
\end{aligned}
$$

显然, $\varphi_0(x),\varphi_1(x),\cdots,\varphi_n(x)$ 线性无关, 且对任意一个 $u_h\in V_h$ 可表示为

$$
u_h(x)=\sum_{i=0}^{N}u_i\varphi_i(x), \quad u_i=u_h\left(x_i\right)
$$

3. 有限元方程的建立

为了找到变分问题 (3.2) 中 V 的一个子空间, 先对线性函数空间 V_h 中的任意函数加上 $v_h(a)=0$ 的条件, 即定义为

$$
V_h^0=V_h\cap V=\{v_h\in V_h \mid v_h(a)=0\}.
$$

变分问题 (3.2) 的有限元逼近问题是找一个 $u_h\in V_h^0$, 使得

$$
a\left(u_h,v_h\right)=F(v_h), \quad \forall v\in V_h^0. \tag{3.4}
$$

假设 $\{\varphi_0,\varphi_1,\cdots,\varphi_N\}$ 是 V_h 的一组基, 将 u_h 展开为

$$
u_h=\sum_{i=0}^{N}u_i\varphi_i, \tag{3.5}
$$

问题 (3.4) 归结为求 $u_1,u_2,\cdots,u_n$, 成立以下线性方程组

$$
\sum_{i=0}^{N}a\left(\varphi_i,\varphi_j\right)u_i=F(\varphi_j), \quad j=1,\cdots,N, \tag{3.6}
$$

即

$$
\sum_{i=1}^{N}a\left(\varphi_i,\varphi_j\right)u_i=F(\varphi_j), \quad j=1,\cdots,N, \tag{3.7}
$$

线性方程组可以写成如下矩阵形式

$$
K\boldsymbol{u}=\boldsymbol{b}. \tag{3.8}
$$

其中

$$
K=\begin{bmatrix}
a(\varphi_1,\varphi_1) & a(\varphi_2,\varphi_1) & \cdots & a(\varphi_N,\varphi_1)\\
a(\varphi_1,\varphi_2) & a(\varphi_2,\varphi_2) & \cdots & a(\varphi_N,\varphi_2)\\
\vdots & \vdots & \vdots & \vdots\\
a(\varphi_1,\varphi_N) & a(\varphi_2,\varphi_N) & \cdots & a(\varphi_N,\varphi_N)
\end{bmatrix},
$$

$$
\boldsymbol{u}=\left(u_1,u_2,\cdots,u_N\right)^{\mathrm{T}}, \quad \boldsymbol{F}=\left(F(\varphi_1),F(\varphi_2),\cdots,F(\varphi_N)\right)^{\mathrm{T}}.
$$

4. 有限元方程的求解

为了计算矩阵每个元素

$$K_{ij} = a(\varphi_j, \varphi_i) = \int_a^b [p\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + q\varphi_i\varphi_j]\,\mathrm{d}x.$$

对单元形状函数作仿射变换, 把区间 I_i 变换到参考单元 $[-1,1]$,

$$\xi = \frac{2x - x_i - x_{i-1}}{h_i}, \quad x = \frac{h_i\xi}{2} + \frac{x_{i-1}+x_i}{2}, \quad x \in I_i,\ \xi \in [-1,1]. \tag{3.9}$$

引入两个标准山形函数

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

它们满足

$$\begin{cases} N_{-1}(-1) = 1, \\ N_{-1}(1) = 0, \end{cases} \qquad \begin{cases} N_1(-1) = 0, \\ N_1(1) = 1. \end{cases}$$

于是基函数 $\varphi_i(x)$ 可写为

$$\varphi_0(x) = \begin{cases} \dfrac{x_1 - x}{h_1} = N_{-1}(\xi(x)), & x \in I_1 = [x_0, x_1], \\ 0, & \text{其他}, \end{cases}$$

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i} = N_1(\xi(x)), & x \in I_i = [x_{i-1}, x_i], \\ \dfrac{x_{i+1} - x}{h_{i+1}} = N_{-1}(\xi(x)), & x \in I_{i+1} = [x_i, x_{i+1}], \\ 0, & \text{其他}, \end{cases} \quad 0 < i < N,$$

$$\varphi_N(x) = \begin{cases} \dfrac{x - x_{N-1}}{h_N} = N_1(\xi(x)), & x \in I_N = [x_{N-1}, x_N], \\ 0, & \text{其他}. \end{cases}$$

线性插值公式 (3.3) 可写为

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

因为 u_h 的自由度是 N, 故 V_h^0 是 N 维线性空间.

显然, 当 $|j-i| \geq 2$ 时 $\varphi_i \cdot \varphi_j = 0$, 系数矩阵 K 是一个三对角矩阵,

$$\begin{aligned} a(\varphi_{j-1}, \varphi_j) &= \int_{x_{j-1}}^{x_j} [p\varphi_j'\varphi_{j-1}' + q\varphi_j\varphi_{j-1}]\,\mathrm{d}x \\ &= \int_{x_{j-1}}^{x_j} [-p\frac{1}{h_j^2} + q\varphi_j(x)\varphi_{j-1}(x)]\,\mathrm{d}x \\ &= \int_{-1}^{1} [p(x_{j-1} + h_j\frac{1+\xi}{2})\frac{1}{h_j^2} + q(x_{j-1} + h_j\frac{1+\xi}{2})\frac{(1-\xi)(1+\xi)}{4}]\frac{h_j}{2}\,\mathrm{d}\xi \end{aligned} \tag{3.10}$$

$$
\begin{aligned}
a(\varphi_j,\varphi_j) &= \int_{x_{j-1}}^{x_j} [p\varphi_j'^2 + q\varphi_j^2]\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} [p\varphi_j'^2 + q\varphi_j^2]\,\mathrm{d}x \\
&= \int_{x_{j-1}}^{x_j} [p\frac{1}{h_j^2} + q\varphi_j^2(x)]\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} [p\frac{1}{h_{j+1}^2} + q\varphi_j^2(x)]\,\mathrm{d}x \\
&= \int_{-1}^{1} [p(x_{j-1}+h_j\frac{1+\xi}{2})\frac{1}{h_j^2} + q(x_{j-1}+h_j\frac{1+\xi}{2})\frac{(1+\xi)^2}{4}]\frac{h_j}{2}\,\mathrm{d}\xi \\
&\quad + \int_{-1}^{1} [p(x_j+h_{j+1}\frac{1+\xi}{2})\frac{1}{h_{j+1}^2} + q(x_j+h_{j+1}\frac{1+\xi}{2})\frac{(1-\xi)^2}{4}]\frac{h_{j+1}}{2}\,\mathrm{d}\xi
\end{aligned} \tag{3.11}
$$

$$
\begin{aligned}
a(\varphi_j,\varphi_{j+1}) &= \int_{x_j}^{x_{j+1}} [p\varphi_j'\varphi_{j+1}' + q\varphi_j\varphi_{j+1}]\,\mathrm{d}x \\
&= \int_{x_j}^{x_{j+1}} [-p\frac{1}{h_{j+1}^2} + q\varphi_j(x)\varphi_{j+1}(x)]\,\mathrm{d}x \\
&= \int_{-1}^{1} [p(x_j+h_{j+1}\frac{1+\xi}{2})\frac{1}{h_{j+1}^2} + q(x_j+h_{j+1}\frac{1+\xi}{2})\frac{(1-\xi)(1+\xi)}{4}]\frac{h_{j+1}}{2}\,\mathrm{d}\xi
\end{aligned} \tag{3.12}
$$

这里 $j=2,3,\cdots,N-1$ 第一行只有两个非零元素 $a(\varphi_1,\varphi_1),a(\varphi_2,\varphi_1)$ 第 N 行也只有两个非零元素 $a(\varphi_n,\varphi_{n-1})$ 和

$$
a(\varphi_N,\varphi_N) = \int_{-1}^{1} [p(x_{N-1}+h_N\frac{1+\xi}{2})\frac{1}{h_N^2} + q(x_{N-1}+h_N\frac{1+\xi}{2})\frac{(1+\xi)^2}{4}]\frac{h_N}{2}\,\mathrm{d}\xi, \tag{3.13}
$$

另一方面

$$
\begin{aligned}
F(\varphi_j) &= \int_{x_{j-1}}^{x_j} f(x)\varphi_j(x)\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} f(x)\varphi_j(x)\,\mathrm{d}x \\
&= h_j\int_0^1 f(x_{j-1}+h_j\frac{1+\xi}{2})\frac{1+\xi}{2}\,\mathrm{d}\xi + h_{j+1}\int_{-1}^1 f(x_j+h_{j+1}\frac{1+\xi}{2})\frac{1-\xi}{2}\,\mathrm{d}\xi
\end{aligned} \tag{3.14}
$$

$$
F(\varphi_N) = \int_{x_{N-1}}^{x_N} f(x)\varphi_j(x)\,\mathrm{d}x = h_N\int_0^1 f(x_{j-1}+h_j\frac{1+\xi}{2})\frac{1+\xi}{2}\,\mathrm{d}\xi \tag{3.15}
$$

注 (3.10)–(3.15)可以用数值积分计算.

系数矩阵 $K_{N\times N}, K_{ij}=a(\varphi_j,\varphi_i)$ 是由 (3.10)–(3.13) 形成的有限元代数方程组中的系数矩阵. 在工程计算中, K_{ij} 往往不是由 (3.10)–(3.13) 形成的, 而是先分析每一个单元的局部双线性形及单元矩阵, 力学上称为单元刚度矩阵 (Element stiffness martix); 再由单元刚度矩阵形成总刚度矩阵，称为总刚度矩阵 (Global stiffness martix). 这种分析比较灵活, 程序也易实现.

每个局部单元 I_i 上, 两个基函数 (单元形状函数).

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

$$
\Phi_2^{I_i}(x) = \begin{cases} \dfrac{x-x_{i-1}}{h_i} = N_1(\xi), & x\in I_i=[x_{i-1},x_i],\ \xi\in[-1,1], \\ 0, & \text{其他}. \end{cases}
$$

则 V_h 的基函数 $\varphi_1, \varphi_2, \cdots, \varphi_N$ 为

$$\begin{aligned} &\varphi_1 = (\Phi_2^{I_1} + \Phi_1^{I_2}), \quad \varphi_2 = (\Phi_2^{I_2} + \Phi_1^{I_3}), \quad \cdots \\ &\varphi_i = (\Phi_2^{I_i} + \Phi_1^{I_{i+1}}), \quad \cdots \quad \varphi_N = \Phi_2^{I_N}. \end{aligned} \tag{3.16}$$

每一个单元 I_i 上, 对应一个单元刚度矩阵 $K_{2\times 2}^{I_i}$, 其中

$$K_{11}^{I_i} = a(\Phi_1^{I_i}, \Phi_1^{I_i}) = \int_{x_{i-1}}^{x_i} [p\Phi_1^{I_i\prime} \cdot \Phi_1^{I_i\prime} + q\Phi_1^{I_i} \cdot \Phi_1^{I_i}]\,\mathrm{d}x,$$

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

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

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

称 $K_{2\times 2}^{I_i}$ 为单元刚度矩阵.

由 (3.16) 方式产生的整体基函数 $\varphi_1, \varphi_2, \cdots, \varphi_N$. 按(3.10)–(3.13) 的过程生成的总矩阵 K 称为总刚度矩阵.

$$K = \begin{bmatrix} K_{11}^{I_1} & K_{12}^{I_1} & & & & & \\ K_{21}^{I_1} & K_{22}^{I_1} + K_{11}^{I_2} & K_{12}^{I_2} & & & & \\ & K_{21}^{I_2} & K_{22}^{I_2} + K_{11}^{I_3} & K_{12}^{I_3} & & & \\ & & K_{21}^{I_3} & K_{22}^{I_3} + K_{11}^{I_4} & K_{12}^{I_4} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & K_{21}^{I_{N-1}} & K_{22}^{I_{N-1}} + K_{11}^{I_N} & K_{12}^{I_N} \\ & & & & & K_{21}^{I_N} & K_{22}^{I_N} \end{bmatrix}.$$

3.2 有限元一维算例

例 3.1 考虑两点边值问题：

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

真解：$u=x(1-x)\sin(x)$, 右端项：$f=(4x-2)\cos(x)+(2+2x-2x^2)\sin(x)$.

```
% FEM1DP.m
% finite element method for 1D elliptic problem
% -u_xx+u=f in (0,1)
% boundary condition: u(0)=u(1)=0;
% exact solution: u=x*(1-x)*sin(x);
% right hand function: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x)
clear all; close all;
Num=[16 32 64 128 256 512]
node_Err=[];  L2_Err=[];  H1_Err=[];  DOF=[];
for k=1:length(Num)
    N=Num(k);  h=1/N;  x=0:h:1;

    M=[1:N;2:N+1];         % information matrix
    % the global node indices of the mesh nodes of all the mesh elements
    [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 nodes
        uh=U(i)*(1-xv)/2+U(i+1)*(1+xv)/2;
        % derivative value of finite element solution at Gauss nodes
        duh=-U(i)/2+U(i+1)/2;
        L2_err(i)=h/2*((tt.*(1-tt).*sin(tt)-uh).^2)'*wv;
        % the square of the L2 error of the i-th interval
        H1_err(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_err))];
    H1_Err=[H1_Err, sqrt(sum(L2_err)+sum(H1_err))];
    doff=N+1;    % degrees of freedom, number of unknowns
    DOF=[DOF, doff];
end
loglog(DOF,node_Err,'r+-','LineWidth',1)
hold on
loglog(DOF,L2_Err,'bo-','MarkerFaceColor','w','LineWidth',1)
hold on
loglog(DOF,H1_Err,'m*-','LineWidth',1)
hold on
grid on
legend('L^2 error','L^{\infty} error','H^1 error','location','SouthWest')
% title('Convergence of finite element method','fontsize',14)
set(gca,'fontsize',12)
xlabel('log_{10}N','fontsize', 14),
ylabel('log_{10}Error','fontsize',14)

% sets axis tick and axis limits
xticks(10.^(1:3))
yticks(10.^(-8:-1))
xlim([10 10^3])
ylim([10^(-8) 10^(-1)])

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

```
node_order
L2_order
H1_order

% print -dpng -r600 FEM1DP.png
% print -depsc2 FEM1DP.eps
```

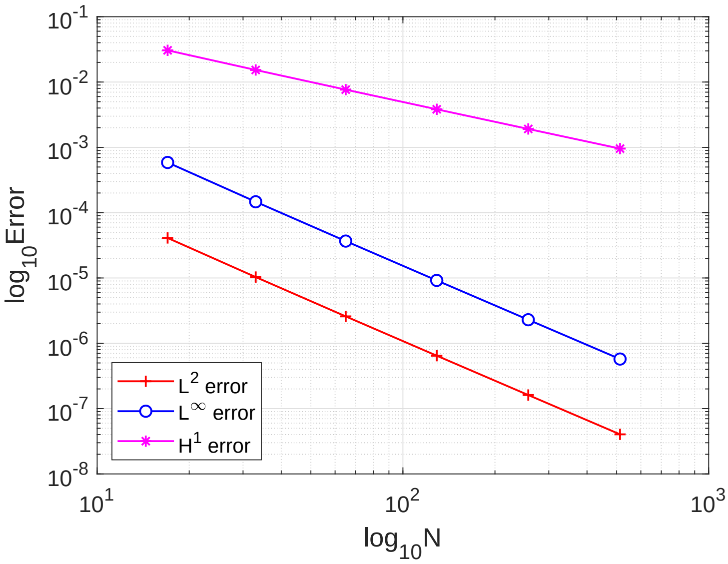


图 26: 一维线性元的收敛速度

4 谱方法

谱方法是一种求解偏微分方程的高精度数值方法. 早在 1820 年, Navier 就运用双重三角级数求解弹性薄板问题. 长期以来, 由于它计算量大而一直没有被广泛使用, 随着计算机运算速度的快速提升与快速 Fourier 变换的出现, 给谱方法带来了生机. 近几十年来, 谱方法得到了蓬勃的发展, 被广泛地应用于科学和工程的各个领域, 如：数值天气预报、流体力学、热传导学、量子力学及电磁学等, 而且它的数值分析理论也不断地完善. 至今, 谱方法已和有限差分法、有限元法一起成为偏微分方程数值求解的第三种基本方法.

谱方法起源于 Ritz-Galerkin 方法, 用正交多项式作为基函数逼近微分方程的解. 根据选取正交多项式的不同, 可分为 Fourier 谱方法、Legendre 谱方法、Chebyshev 谱方法、Jacobi 谱方法、Laguerre 谱方法及 Hermite 谱方法等. 根据基函数与测试函数的情况谱方法又可分为 Galerkin 谱方法 (基函数与测试函数相同)、Petrov-Galerkin 谱方法 (基函数与测试函数不同) 和谱配置法 (测试函数是 Lagrange 插值多项式).

对于有限差分法或有限元法, 误差收敛速率为 $O(N^{-k})$, 常数 k 取决于方法的逼近阶数. 对于谱方法, 如果原方程的解是无穷可微的, 误差收敛速度可以达到 $O(N^{-m})$ (N 是基函数的个数, m 是正则性指数), 如果解是解析的, 收敛速度甚至比 $O(c^N)(0 < c < 1)$ 更快 (即达到指数收敛), 这种谱方法具有的误差收敛行为称为“谱精度”. 这一优点是有限差分法和有限元法无法比拟的, 众多的实际应用和数值实例也证实了该方法的有效性. 因此, 谱方法日益受到人们的重视.

传统的谱方法虽然具有高精度, 却以牺牲区域灵活性为代价, 它与差分方法一样, 只能求解规则区域上的微分方程的解. 而有限元方法适合于解决复杂几何区域的问题. 许多数值方法, 如 h-p 有限元方法和谱元方法, 它们结合了谱方法和有限元方法的优点. 有限差分方法、有限元方法与谱方法的比较如表 1 所示.

表 1: 几种数值方法的比较

	有限差分 和/或 有限元	谱方法
基函数 (有限元, 谱方法)	局部	全局
收敛阶	低阶	高阶
提高精度	细化网格	增加基函数的个数
计算效率	稀疏矩阵	稀疏矩阵 (特定 PDEs)

参考书籍

1. Spectral Methods: Algorithms, Analysis and Applications
2. 谱方法的数值分析

4.1 谱方法与配置法

两点边值问题

考虑二阶微分方程边值问题：

$$\begin{aligned}
&\mathbf{L}u = -u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \quad x \in (-1,1), \\
&B_{\pm}u(\pm 1) = g_{\pm}.
\end{aligned}$$

考虑齐次 Dirichlet 边界条件, 非齐次边界条件可以通过考虑 $v = u - \tilde{u}$ 来处理, 其中 $\tilde{u}$ 是满足非齐次边界条件的“简单”函数.

Spectral-Galerkin 方法

定义有限维逼近空间：

$$X_N = \{\phi \in P_N : B_{\pm}\phi(\pm 1) = 0\} \ \Rightarrow\ \dim(X_N) = N - 1.$$

设 $\{\phi_k\}_{k=0}^{N-2}$ 是 X_N 的一组基函数, 我们将逼近解展开为

$$u_N(x) = \sum_{k=0}^{N-2} \hat{u}_k \phi_k(x) \in X_N.$$

展开式系数 $\{\hat{u}_k\}_{k=0}^{N-2}$ 由以下方程唯一确定.

$$\int_{-1}^{1} \left(\mathbf{L}u_N(x) - f(x)\right)\phi_j(x)\omega(x)dx = 0, \quad 0 \le j \le N-2,$$

等价于 (谱格式)：

$$\begin{cases} \text{Find } u \in X_N \text{ such that} \\ (\mathbf{L}u_N, v_N)_\omega = (f, v_N)_\omega, \quad \forall v_N \in X_N. \end{cases}$$

其中 $(\cdot,\cdot)_\omega$ 是 $L^2_\omega(-1,1)$ 的内积. 设

$$\begin{aligned}
f_j &= (f, \phi_j)_\omega, & \boldsymbol{f} &= (f_0, f_1, \ldots, f_{N-2})^{\mathrm{T}}, \\
s_{jk} &= (\mathbf{L}\phi_k, \phi_j)_\omega, & S &= (s_{jk})_{j,k=0,\cdots,N-2},
\end{aligned}$$

令 $\boldsymbol{u} = (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-2})^{\mathrm{T}}$, 可得

$$S\boldsymbol{u} = \boldsymbol{f}.$$

因此, 选择合适的基函数 $\{\phi_j\}$ 使得：

- 可以有效地计算右端项 $(f, \phi_j)_\omega$.
- 有效求解线性系统 $S\boldsymbol{u} = \boldsymbol{f}$.

其核心思想是利用正交多项式或正交函数的组合构造基函数. 由于 $x \in (-1,1)$ 可以选择 Legendre, Chebyshev 等多项式作为基函数.

Spectral-Collocation 方法

配置方法是使残差在一组预先设定的点上趋近于零. 设 $\{x_j\}_{j=0}^N$ $(x_0=-1, x_N=1)$ 是一组 Gauss-Lobatto 点, 设 P_N 是不低于 N 次的实代数多项式的集合. 谱配置格式是找 $u_N \in P_N$ 使得

$$\mathbf{R}_N\left(x_k\right)=\mathbf{L} u_N\left(x_k\right)-f\left(x_k\right)=0, \quad 1 \leq k \leq N-1, \tag{4.1}$$

而且 u_N 要精确满足边界条件:

$$B_- u_N(x_0)=g_-, \quad B_+ u_N(x_N)=g_+. \tag{4.2}$$

逼近解展开为以下形式

$$u_N(x)=\sum_{j=0}^N u_N\left(x_j\right) h_j(x).$$

其中 h_j 是 Lagrange 基多项式 (也称为节点基函数), 即 $h_j \in P_N, h_j(x_k)=\delta_{kj}$, 将 $u_N(x)$ 的展开式代入 (4.1) 和 (4.2) 可得

$$\sum_{j=0}^N\left[\mathbf{L} h_j\left(x_k\right)\right] u_N\left(x_j\right)=f\left(x_k\right), \quad 1 \leq k \leq N-1,$$

$$\sum_{j=0}^N\left[B_- h_j\left(x_0\right)\right] u_N\left(x_j\right)=g_-, \quad \sum_{j=0}^N\left[B_+ h_j\left(x_N\right)\right] u_N\left(x_j\right)=g_+.$$

上述系统包含 $n+1$ 方程和 $n+1$ 个未知数, 因此我们可以用矩阵形式重写它. 下面考虑 Dirichlet 边界条件 $u(\pm 1)=g_\pm$, 这里设置 $u_N\left(x_0\right)=g_-$, $u_N\left(x_N\right)=g_+$, 可以推出

$$\sum_{j=1}^{N-1}\left[\mathbf{L} h_j\left(x_k\right)\right] u_N\left(x_j\right)=f\left(x_k\right)-\left\{\left[\mathbf{L} h_0\left(x_k\right)\right] g_-+\left[\mathbf{L} h_N\left(x_k\right)\right] g_+\right\}, 1 \leq k \leq N-1. \tag{4.3}$$

对 $u_N(x)$ 进行 m 次微分

$$u_N^{(m)}\left(x_k\right)=\sum_{j=0}^N d_{kj}^{(m)} u_N\left(x_j\right), \quad d_{kj}^{(m)}=h_j^{(m)}\left(x_k\right),$$

矩阵 $D^{(m)}=\left(d_{kj}^{(m)}\right)_{k,j=0,\ldots,N}$ 称为关于节点 $\{x_j\}_{j=0}^N$ 的 m 次微分矩阵. 用 $\boldsymbol{u}^{(m)}$ 表示一个向量, 其分量是配置点处的 $u_N^{(m)}$ 的值, 上式可写为

$$\boldsymbol{u}^{(m)}=D^{(m)} \boldsymbol{u}^{(0)}, \quad m \geq 1.$$

因此

$$\mathbf{L} h_j\left(x_k\right)=-d_{kj}^{(2)}+p\left(x_k\right) d_{kj}^{(1)}+q\left(x_k\right) \delta_{kj}.$$

令 $\boldsymbol{f}$ 表示方程 (4.3) 右端 $N-1$ 个分量的向量. 设

$$\widetilde{D}_m=\left(d_{kj}^{(m)}\right)_{k,j=1,\ldots,N-1}, \quad \text{其中 } m=1,2,$$
$$P=\operatorname{diag}\left(p\left(x_1\right), \ldots, p\left(x_{N-1}\right)\right), \quad Q=\operatorname{diag}\left(q\left(x_1\right), \ldots, q\left(x_{N-1}\right)\right).$$

方程 (4.3) 可以简化为

$$(-\widetilde{D}_2+P\widetilde{D}_1+Q)\boldsymbol{u}^{(0)}=\boldsymbol{f}.$$

4.2 Legendre 谱方法及其算例

本小节介绍一维问题的 Legendre 谱方法, 然后 MATLAB 编程算出数值解以及误差.

考虑两点边值问题:

$$\begin{cases} -u''(x) + \alpha u(x) = f(x), \quad x \in I = (-1,1), \\ u(-1) = 0, \; u(1) = 0. \end{cases} \tag{4.4}$$

Legendre 谱方法

弱形式:

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

令 $\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x)$ 满足边界条件 [Book[1], P146], 可得 $a_k = 0$, $b_k = -1$, 于是

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

设 $\{\phi_k\}_{k=0}^{N-2}$ 是 X_N 的一组基函数, 我们将逼近解展开为

$$u_N(x) = \sum_{k=0}^{N-2} \hat{u}_k \phi_k(x) \in X_N.$$

谱格式:

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

将 $u_N(x)$ 的展开式代入谱格式方程, 取 $v_N = \phi_k$. 令

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

可得以下方程组

$$(S + \alpha M)\boldsymbol{u} = \boldsymbol{f}.$$

刚度矩阵 $S = (s_{jk})$ 是一个对角矩阵 [Book, P146-4.22], 它的非零元素为

$$s_{kk} = -(4k+6)b_k = 4k+6, \quad k = 0, 1, \cdots, N-2.$$

[1]J. Shen, T. Tang, and L.L. Wang. Spectral Methods: Algorithms, Analysis and Applications. Spinger, 2011.

质量矩阵 $M=(m_{jk})$ 是一个对称的五对角矩阵 [Book, P146-4.23], 它的非零元素为

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

如果想让生成的刚度矩阵 S 为单位对角阵, 即 $s_{kk}=1$ (相应的质量矩阵也会改变), 可选择基函数

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

例 4.1 两点边值问题 (4.4), 取 $\alpha=1$, 真解: $u=\sin(k\pi x)$, 右端项: $f=k^2\pi^2\sin(k\pi x)+\sin(k\pi x)$. 程序 LegenSM.m 选取 ϕ_k 为基函数, 程序 LegenSM1.m 选择 $\tilde{\phi}_k$ 为基函数.

```
% LegenSM.m
% Legendre spectral-Galerkin method 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; close all;
kw=1;
Nvec=[4:2:28];
L2_Err=[]; Max_Err=[];  % initialization error
% Loop for various modes N to calculate numerical errors
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);        % exact function
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv);  % Right-hand-side(RHS)
    % Calculting coefficient matrix
    S=diag(4*(0:N-2)+6);       % stiffness matrix
    M=diag(2./(2*(0:N-2)+1)+2./(2*(0:N-2)+5))...
        -diag(2./(2*(0:N-4)+5),2)...
        -diag(2./(2*(2:N-2)+1),-2);  % mass matrix
    A=S+M;
    % Solving the linear system
    Pm=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(x)
    un=Pm'*uh;                     % compositing the numerical solution

    L2_err=sqrt(((un-u).^2)'*wv);  % L^2 error
    Max_err=max(abs(un-u));        % maximum pointwise error
    L2_Err=[L2_Err;L2_err];
    Max_Err=[Max_Err;Max_err];
end
```

```
% Plot the L^2 and maximum pointwise error
plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
grid on,
legend('L^2 error','L^{\infty} error')
% title('Convergence of Legendre-Galerkin method','fontsize',12)
set(gca,'fontsize',12)
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)

% print -dpng -r600  LegenSM.png
% print -depsc2 LegenSM.eps
```

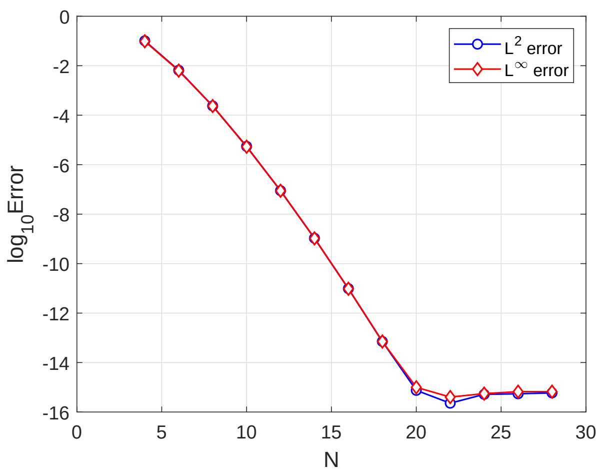


图 27: Legendre 谱方法

```
% LegenSM1.m
% Legendre spectral-Galerkin method 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; close all;
kw=10;
Nvec=[32:2:76];
% Initialization for error
L2_Err=[]; Max_Err=[];
% Loop for various modes N to calculate numerical errors
for N=Nvec
    [xv,wv]=legs(N+1);          % Legendre-Gauss points and weights
    Lm=lepolym(N,xv);           % matrix of Legendre polynomials
    u=sin(kw*pi*xv);            % exact solution
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv);  % 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(x)
    un=Pm'*uh;                  % compositing the numerical solution

    L2_error=sqrt(((un-u).^2)'*wv);  % L^2 error
    Max_error=norm(abs(un-u),inf);   % maximum pointwise error
    L2_Err=[L2_Err;L2_error];
    Max_Err=[Max_Err;Max_error];
end
% Plot L^2 and maximum pointwise error
plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
grid on
legend('L^2 error','L^{\infty} error','location','NorthEast')
set(gca,'fontsize',12)
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)

% sets axis tick and axis limits
xticks(30:10:80)
yticks(-15:3:0)
xlim([30 80])
ylim([-15 0])

% print -dpng -r600 LegenSM1.png
% print -depsc2 LegenSM1.eps
```

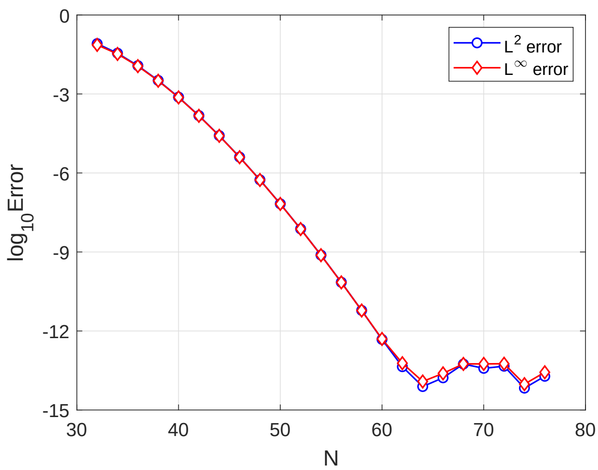


图 28: Legendre 谱方法

4.3 Legendre 谱方法及其算例二

考虑两点边值问题：

$$\begin{cases} -u''(y)+u(y)=f(y), & y\in\Lambda=(0,1), \\ u(0)=1,\ u'(1)=0. \end{cases} \tag{4.5}$$

这里 $y\in\Lambda=[0,1]$, 通过变换方程的方式变到 $[-1,1]$ 的标准形式, 再利用 Legendre 谱方法求解.

令 $x\in I=[-1,1]$, $y=\frac{x}{2}+\frac{1}{2}$, $U(x)=u(y)-1$, 变换后的方程

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

其中 $F(x)=f(2x-1)-1$.

Legendre-Galerkin 方法

弱形式：

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

令 $\phi_k(x)=L_k(x)+a_kL_{k+1}(x)+b_kL_{k+2}(x)$ 满足边界条件, 可得

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

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

逼近解展开为以下形式

$$U_N(x)=\sum_{k=0}^{N-2}\hat{U}_k\phi_k(x)\in X_N.$$

谱格式：

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

将 $u_N(x)$ 的展开式代入谱格式方程. 令

$$f_k=\int_I f_N\phi_k dx, \quad \boldsymbol{f}=(f_0,f_1,\ldots,f_{N-2})^T,$$
$$U_N=\sum_{j=0}^{N-2}\hat{U}_j\phi_j, \quad \boldsymbol{U}=\left(\hat{U}_0,\hat{U}_1,\ldots,\hat{U}_{N-2}\right)^T,$$
$$s_{kj}=\int_I \phi_j'\phi_k' dx, \quad m_{kj}=\int_I \phi_j\phi_k dx,$$
$$S=(s_{kj})_{0\le k,j\le N-2}, \quad M=(m_{kj})_{0\le k,j\le N-2}.$$

取 $\phi=\phi_k$, 可得到以下方程组

$$(4S+M)\boldsymbol{U}=\boldsymbol{f}.$$

刚度矩阵 $S=(s_{jk})$ 是一个对角矩阵 [Book, P146-4.22], 它的非零元素为

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

质量矩阵 $M=(m_{jk})$ 是一个对称的五对角矩阵 [P146-4.23], 它的非零元素为

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

例 4.2 两点边值问题 (4.5).

谱方法可以求解得到 $U_N(x)$, 通过变换 $u(y)=U(x)+1$ 得到原方程的数值解.

```
% LegenSM3.m
% Legendre spectral-Galerkin 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);
% transformed equation: -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; close all;
Nvec=3:18;
% Initialization error and condition number
L2_Err=[];  condnv=[];
% Loop for various modes N to calculate numerical errors
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;     % exact solution
    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,:));
    b=Pm*diag(wv)*F;       % solving RHS
    Uh=A\b;                % expansion coefficients of u_N(x)
    Un=Pm'*Uh;             % compositing the numerical solution

    L2_error=sqrt(((Un-U).^2)'*wv);  % L^2 error
    L2_Err=[L2_Err;L2_error];
    condnv=[condnv,cond(A)];         % condition number
end
% Plot L^2 error
plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
grid on
%title('L^2 error of Legendre-Galerkin method','fontsize',12)
set(gca,'fontsize',12)
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)

% sets axis tick and axis limits
xticks(2:2:18)
yticks(-16:2:0)
xlim([2 18])
ylim([-16 0])

% print -dpng -r600 LegenSM3.png
% print -depsc2 LegenSM3.eps
```

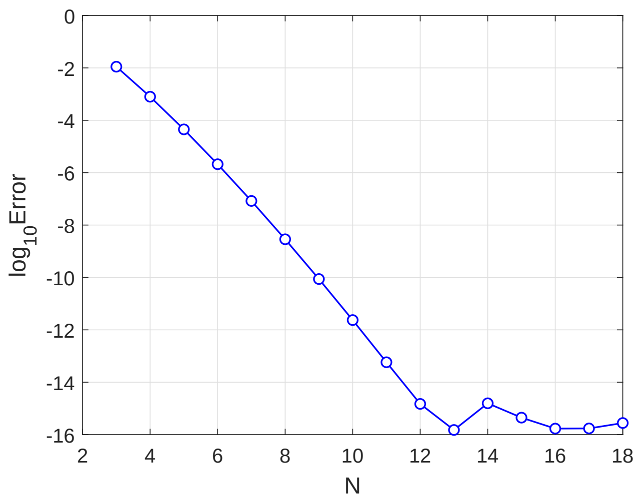


图 29: Legendre 谱方法

4.4 Chebyshev 谱方法及其算例

本小节介绍一维问题的 Chebyshev 谱方法, 然后 MATLAB 编程算出数值解以及误差.

考虑两点边值问题:

$$\begin{cases} -u''(x)+\alpha u(x)=f(x), \quad x\in I=(-1,1), \\ u(-1)=0,\ u(1)=0. \end{cases} \tag{4.6}$$

Chebyshev-Galerkin 方法

Chebyshev 多项式的权函数 $\omega=\left(1-x^2\right)^{-1/2}$.

令 $\phi_k(x)=T_k(x)+a_kT_{k+1}(x)+b_kT_{k+2}(x)$ 满足边界条件, [Book, P146] 可得 $a_k=0, b_k=-1$, 于是

$$\phi_k(x)=T_k(x)-T_{k+2}(x).$$

设 $\{\phi_k\}_{k=0}^{N-2}$ 是 X_N 的一组基函数, 我们将逼近解展开为

$$u_N(x)=\sum_{k=0}^{N-2}\hat{u}_k\phi_k(x)\in X_N.$$

谱格式:

$$\begin{cases} \text{Find } u_N\in X_N \text{ such that} \\ (u_N'',v_N)_\omega+\alpha(u_N,v_N)_\omega=(f,v_N)_\omega, \quad v_N\in X_N. \end{cases}$$

将 $u_N(x)$ 的展开式代入谱格式方程. 令

$$\begin{aligned} f_k &= \int_I f_N\phi_k\omega dx, \quad \boldsymbol{f}=(f_0,f_1,\ldots,f_{N-2})^T, \\ u_N &= \sum_{j=0}^{N-2}\hat{u}_j\phi_j, \quad \boldsymbol{u}=(\hat{u}_0,\hat{u}_1,\ldots,\hat{u}_{N-2})^T, \\ s_{kj} &= -\int_I \phi_j''\phi_k\omega dx, \quad m_{kj}=\int_I\phi_j\phi_k\omega dx, \\ S &= (s_{kj})_{0\le k,j\le N-2}, \quad M=(m_{kj})_{0\le k,j\le N-2}. \end{aligned}$$

取 $v_N=\phi_k$, 可得到以下方程组

$$(S+\alpha M)\boldsymbol{u}=\boldsymbol{f}.$$

质量矩阵 $M=(m_{jk})$ 是一个对称正定的五对角矩阵 [Book, P149-4.29], 它的非零元素为

$$m_{jk}=m_{kj}=\begin{cases} \frac{\pi}{2}\left(c_k+a_k^2+b_k^2\right), & j=k, \\ \frac{\pi}{2}\left(a_k+a_{k+1}b_k\right), & j=k+1, \\ \frac{\pi}{2}b_k, & j=k+2. \end{cases}$$

其中 $c_0=2$, 当 $k\geq 1$ 时, $c_k=1$.

刚度矩阵 $S=(s_{jk})$ 是一个上三角矩阵 [Book, P149-4.30], 其元素如下：

$$s_{kj}=\begin{cases}2\pi(k+1)(k+2), & j=k,\\ 4\pi(k+1), & j=k+2,k+4,k+6,\cdots,\\ 0, & j<k \text{ or } j+k \text{ odd}.\end{cases}$$

数值例子

例 4.3 两点边值问题 (4.6), 取 $\alpha=1$, 真解：$u=\sin(k\pi x)$, 右端项：$f=k^2\pi^2\sin(k\pi x)+\sin(k\pi x)$.

```
% ChebSM1.m
% Chebyshev spectral-Galerkin method for the model equation
% -u_xx+u=f 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)+sin(kw*pi*x);
% Rmk: Use routines japoly(); jags(); japolym();
clear all; close all;
kw=10;
Nvec=32:2:76;
% Initialization for error
L2_Err=[];  Max_Err=[];
% Loop for various modes N to calculate numerical errors
for N=Nvec
    [xv,wv]=jags(N+1,-1/2,-1/2);   % Chebyshev-Gauss points and weights
    Cm=japolym(N,-1/2,-1/2,xv)./japolym(N,-1/2,-1/2,1);  % matrix of
        Chebyshev polynomals
    u=sin(kw*pi*xv);                                 % exact solution
    f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv);  % RHS
    % Calculting coefficient matrix
    S=zeros(N-1);    % stiff matrix
    for k=1:N-1
        for j=1:N-1
            if k==j
                S(k,j)=2*pi*k*(k+1);
            elseif (k<j && mod(j-k,2)==0)
                S(k,j)=4*pi*k;
            else
                S(k,j)=0;
            end
        end
    end

    M=diag([3/2*pi, pi*ones(1,N-2)])+diag(-pi/2*ones(1,N-3),2)...
        +diag(-pi/2*ones(1,N-3),-2);    % mass matrix
    A=S+M;

    % Solving the linear system
```

```
    Pm=Cm(1:end-2,:)-Cm(3:end,:);   % matrix of Phi(x)
    b=Pm*diag(wv)*f;          % Solving RHS
    uh=A\b;                   % expansion coefficients of u_N(x)
    un=Pm'*uh;                % compositing the numerical solution

    L2_err=sqrt(((un-u).^2)'*wv);  % L^2 error
    Max_err=norm(abs(un-u),inf);   % maximum pointwise error
    L2_Err=[L2_Err;L2_err];
    Max_Err=[Max_Err;Max_err];
end
% plot L^2 and maximum pointwise error
plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
grid on
legend('L^2 error','L^{\infty} error','location','NorthEast')
% title('Convergence of Chebyshev-Galerkin method','fontsize',12)
set(gca,'fontsize',12)
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)

% sets axis tick and axis limits
xticks(30:10:80)
yticks(-14:2:0)
xlim([30 80])
ylim([-14 0])

% print -dpng -r600 ChebSM1.png
% print -depsc2 ChebSM1.eps
```

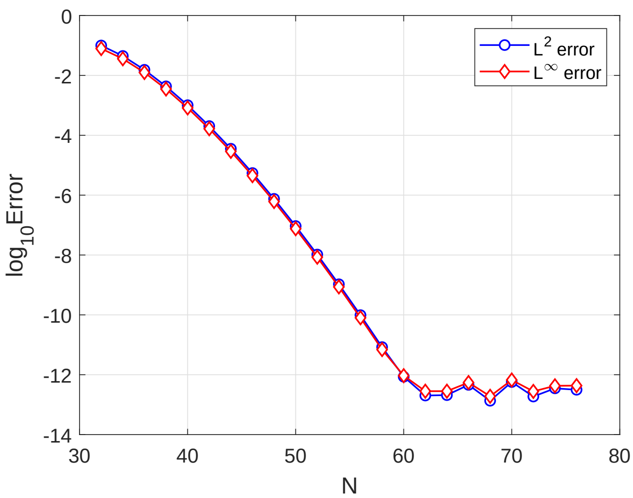


图 30: Chebyshev 谱方法

4.5 Legendre 配置法及其算例

本小节介绍两点边值问题的 Legendre 配置方法, 然后 MATLAB 编程算出数值解以及误差.

考虑两点边值问题:

$$\begin{cases} -\mu u''(x)+\nu u'(x)+\rho u(x)=f(x), \quad x\in I=(-1,1), \\ u(-1)=0,\ u(1)=0. \end{cases} \tag{4.7}$$

Legendre Collocation 方法

在区间 $[-1,1]$ 上选取配置点 $\{x_j\}_{j=0}^N$ 为 Legendre-Gauss-Lobatto 节点, 其中 $x_0=-1, x_N=1$. 配置格式

$$\begin{cases} \text{Find } u\in P_N \text{ such that} \\ -\mu u_N''(x_i)+\nu u_N'(x_i)+\rho u_N(x_i)=f(x_i), \quad 1\le i\le N-1, \\ u_N(-1)=0, \quad u_N(1)=0. \end{cases}$$

逼近解展开为以下形式

$$u_N(x)=\sum_{j=0}^N u_N\left(x_j\right)h_j(x).$$

其中 h_j 是 Lagrange 基多项式（也称为节点基函数）, 即 $h_j\in P_N, h_j(x_k)=\delta_{kj}$, 令 $D=(d_{kj}:=h_j'(x_k))_{k,j=1,2,\cdots,N}$. 记 $w_j=u_N(x_j)$, 将 $u_N(x)$ 展开为 $u_N(x)=\sum_{j=0}^N w_jh_j(x)$, 可知

$$u_N\left(x_k\right)=\sum_{j=0}^N w_jh_j\left(x_k\right)=w_k,$$

$$\begin{aligned} u_N'(x_k) &= \sum_{j=0}^N w_jh_j'(x_k)=\sum_{j=0}^N d_{kj}w_j \\ &= \sum_{j=1}^{N-1} d_{kj}w_j+d_{k0}w_0+d_{kN}w_N, \end{aligned}$$

$$\begin{aligned} u_N''(x_k) &= \sum_{j=0}^N w_jh_j''(x_k)=\sum_{j=0}^N (D^2)_{kj}w_j \\ &= \sum_{j=1}^{N-1}\left(D^2\right)_{kj}w_j+\left(D^2\right)_{k0}w_0+\left(D^2\right)_{kN}w_N. \end{aligned}$$

将以上公式代入方程 (equcm) 可得

$$\begin{cases} \displaystyle\sum_{j=0}^N\left[-\mu\left(D^2\right)_{ij}+\nu d_{ij}+\rho\delta_{ij}\right]w_j=f(x_i), \quad i=1,2,\cdots,N-1, \\ w_0=0, \quad w_N=0. \end{cases}$$

记

$$
\begin{aligned}
&a_{ij} = -\mu(D^2)_{ij} + \nu d_{ij} + \rho\delta_{ij}, \quad 1 \le i \le N-1, 0 \le j \le N,\\
&a_{0j} = \delta_{0j}, \quad a_{Nj} = \delta_{Nj}, \quad 0 \le j \le N,\\
&\mathbf{b} = (0, f(x_1), f(x_2), \cdots, f(x_{N-1}), 0)^T,\\
&\mathbf{w} = (w_0, w_1, \ldots, w_N)^T, \quad A = (a_{ij})_{0\le i,j\le N}\,.
\end{aligned}
$$

线性系统可以简化为

$$
A\mathbf{w} = \mathbf{b}.
$$

例 4.4 两点边值问题 (4.7), 取 $\nu = 1, \mu = 1, \rho = 1$.
真解: $u = \sin(k\pi x)$, 右端项: $f = \mu k^2\pi^2 \sin(k\pi x) + \nu k\pi \cos(k\pi x) + \rho \sin(k\pi x)$. (取 $k = 10$)

```
% LegenCM2.m
% Legendre-collocation method for the model equation:
% -mu*u''(x)+nu*u'(x)+rho*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)=mu*kw^2*pi^2*sin(kw*pi*x)+nu*kw*pi*cos(kw*pi*x)+rho*sin(kw*pi*x)
% Rmk: Use routines lepoly(); legslb(); legslbdm();
clear all; close all;
kw=10;
nu=1;
mu=1;
rho=1;
Nvec=32:2:76;
% Initialization for error
L2_Err=[];  Max_Err=[];
% Loop for various modes N to calculate numerical errors
for N=Nvec
    [xv,wv]=legslb(N);    % Legendre-Gauss-Lobatto points and weights
    u=sin(kw*pi*xv);      % exact solution
    f=mu*kw*kw*pi^2*sin(kw*pi*xv)+nu*kw*pi*cos(kw*pi*xv)+rho*sin(kw*pi*xv);
         % RHS
    % Solve the collocation system
    D1=legslbdiff(N,xv);  % 1st order differentiation matrix
    %D1=legslbdm(N);      % 1st order differentiation matrix
    D2=D1*D1;             % 2nd order differentiation matrix
    % Compositing the coefficient matrix
    D=-mu*D2(2:N-1,2:N-1)+nu*D1(2:N-1,2:N-1)+rho*eye(N-2);
    b=f(2:N-1);           % RHS
    un=D\b;
    un=[0;un;0];

    L2_error=sqrt(((un-u).^2)'*wv);  % L^2 error
    Max_error=norm(abs(un-u),inf);   % maximum pointwise error
```

```
    L2_Err=[L2_Err;L2_error];
    Max_Err=[Max_Err;Max_error];
end
% Plot L^2 and maximum pointwise error
plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
hold on
plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
grid on
legend('L^2 error','L^{\infty} error','location','NorthEast')
set(gca,'fontsize',12)
xlabel('N','fontsize', 14), ylabel('log_{10}Error','fontsize',14)
% title('Convergence of Legendre-collocation method','fontsize',12)

% sets axis tick and axis limits
xticks(30:10:80)
yticks(-15:3:0)
xlim([30 80])
ylim([-15 0])

% print -dpng -r600 LegenCM2.png
% print -depsc2 LegenCM2.eps
```

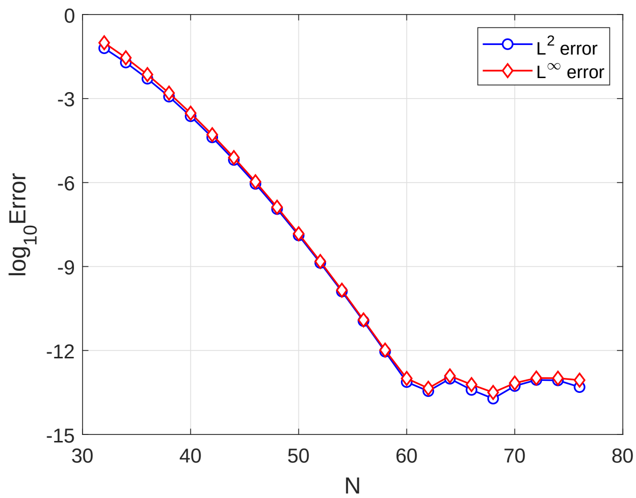


图 31: Legendre 配置方法

注 (1) 程序 LegenCM1.m 对应配置方法求解两点边值问题

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

真解：$u=\sin(k\pi x)$, 右端项：$f=k^2\pi^2\sin(k\pi x)+\alpha\sin(k\pi x)$. (取 $k=1$)

(2) 程序 LegenCM3.m 对应配置方法求解两点边值问题 (4.5).