
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
1.1	一阶常微分方程初值问题	1
1.2	Euler 法	1
1.3	梯形方法	3
1.4	后退 Euler 法	4
1.5	改进 Euler 方法	6
1.6	经典四阶 Runge-Kutta 方法	7
1.7	计算数值方法收敛阶	9
1.8	隐式 Runge-Kutta 方法	15
2	有限差分方法	20
2.1	一维差分方法	20
2.2	二维矩形网的差分格式	27
2.3	一维热传导方程的差分格式	35
2.4	一维热传导方程 CN 格式	38
2.5	一维波动方程的差分格式	41
2.6	极坐标形式的差分格式	45
3	有限元方法	49
3.1	一维有限元方法	49
3.2	有限元一维算例	55
4	谱方法	58
4.1	谱方法与配置法	59
4.2	Legendre 谱方法及其算例	61
4.3	Legendre 谱方法及其算例二	65
4.4	Chebyshev 谱方法及其算例	68
4.5	Legendre 配置法及其算例	71

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 $G: 0 \leq t \leq T, |u| < \infty$ 上连续, 求 $u = u(t)$ 满足

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

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

$$|f(t, u_1) - f(t, u_2)| \leq L|u_1 - u_2|. \quad (1.2)$$

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 $t_n = nh (n = 0, 1, \dots, N)$ 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

$$\begin{aligned} u(t_1) &= u(t_0 + h) = u(t_0) + hu'(t_0) + \frac{h^2}{2}u''(t_0) + \frac{h^3}{6}u'''(\zeta) \\ &= u_0 + hf(t_0, u_0) + R_0, \end{aligned} \quad (1.3)$$

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:

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

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:

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

特别地,

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

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

例 1.1

$$\begin{cases} \frac{du}{dx} = t^2 + t - u, & 0 < t \leq 1, \\ u(0) = 0. \end{cases}$$

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

```
1 % Euler1.m
2 % Euler method for the ODE model
3 % u'(x)=x^2+x-u, x in [0,1]
4 % Initial condition: u(0)=0 ;
5 % Exact solution: u(x)=-exp(-x)+x^2-x+1.
6 clear all; close all;
7 h=0.1;
8 x=0:h:1; % interval partition
9 N=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for n=1:N
13     u(n+1)=u(n)+h.*fun(x(n),u(n));
14 end
15 ue=-exp(-x)+x.^2-x+1; % exact solution
16 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
17 legend('Exact','Numerical','location','northwest')
18 % title('Euler method','fontsize',12)
19 set(gca,'fontsize',12)
20 xlabel('x','fontsize',16), ylabel('u','fontsize',16)
21
22 % print -dpng -r600 Euler1.png
23 % print -depsc2 Euler1.eps
```

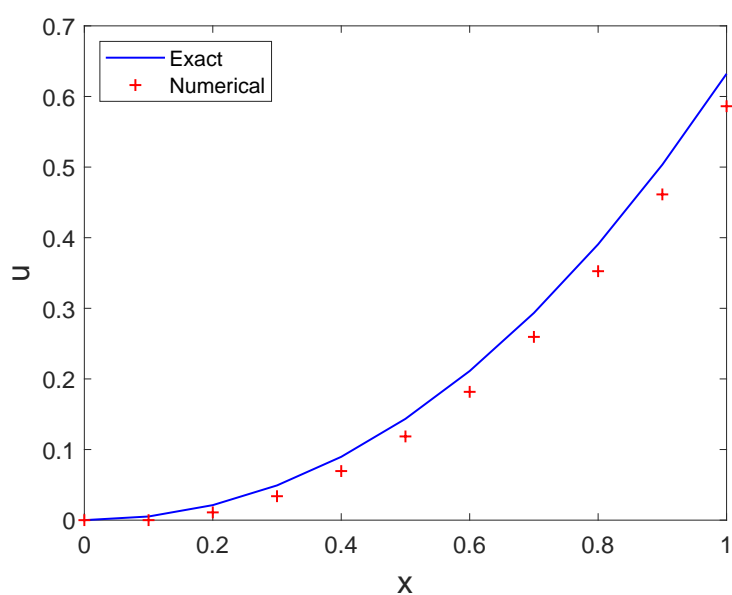


图 1: Euler 法

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

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

一般的通式:

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

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

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & 0 < t \leq 1, \\ u(0) = 0. \end{cases}$$

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:

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

可得

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

```
1 % Trapezoidal.m
2 % Trapezoidal rule for the ODE model
3 % u'(x)=x^2+x-u, x in [0,1]
4 % Initial condition: u(0)=0 ;
5 % Exact solution: u(x)=-exp(-x)+x^2-x+1.
6 clear all; close all;
7 h=0.1;
8 x=0:h:1; % interval partition
9 N=length(x)-1;
10 u(1)=0; % initial value
11 for n=1:N
12     u(n+1)=(2-h)/(2+h).*u(n)+h/(2+h).*(x(n)^2+x(n)+x(n+1)^2+x(n+1));
13 end
14 ue=-exp(-x)+x.^2-x+1; % exact solution
15 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
16 legend('Exact','Numerical','location','northwest')
17 %title('Trapezoidal rule','fontsize',12)
18 set(gca,'fontsize',12)
19 xlabel('x','fontsize',16), ylabel('u','fontsize',16)
20
21 % print -dpng -r600 Trapezoidal.png
22 % print -depsc2 Trapezoidal.eps
```

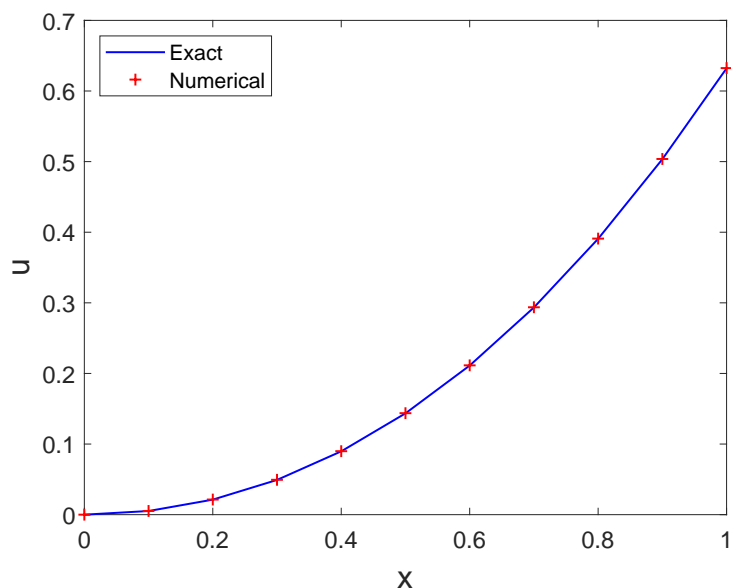


图 2: 梯形公式

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 $u(t_1)$,

$$u_1 = u_0 + hf(t_1, u_1), \quad (1.9)$$

一般的通式:

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

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

$$\begin{cases} \frac{du}{dt} = t - \frac{t}{u}, & 0 < t \leq 1, \\ u(0) = 0. \end{cases}$$

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

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

可转化为

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

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 ($f'(x_k) \neq 0$), 求 x_{k+1} 的方法

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

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)

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

数值格式

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

即

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

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.

```

1 % BackEuler.m
2 % Backward Euler Method for Nonlinear ODE:
3 % u'(x)=u-2x/u in [0,1]
4 % Initial condition: u(0)=1;
5 % Exact solution: u=sqrt(2*x+1)
6 clear all; close all;
7 h=0.01;
8 x=0:h:1;
9 N=length(x)-1;
10 u(1)=1;
11 NI(1,N)=0; % Record the number of iterations
12 for n=1:N
13     % Newton iteration
14     Xn=u(n);
15     Xp=Xn;
16     Xprev=0;
17     while abs(Xp-Xprev) > eps*abs(Xp)
18         Xprev=Xp;
19         Xp=Xp-(Xp-h*Xp+2*h*x(n+1)/Xp-u(n))./(1-h-2*h*x(n+1)/(Xp^2));
20         NI(n)=NI(n)+1;
21     end
22     u(n+1)=Xp;
23 end
24 ue=sqrt(2*x+1); % exact solution
25 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
26 legend('Exact ', 'Numerical', 'location', 'northwest')
27 % title('Backward Euler method', 'fontsize', 12)
28 set(gca, 'fontsize', 12)
29 xlabel('t', 'fontsize', 16), ylabel('u', 'fontsize', 16)
30
31 % computing error
32 error=max(abs(u-ue))
33
34 % print -dpng -r600 BackEuler.png
35 % 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,

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

例 1.4

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & 0 < t \leq 1, \\ u(0) = 0. \end{cases} \quad (1.18)$$

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

```

1 % ModiEuler.m
2 % Modified Euler method for the ODE model
3 % u'(x)=x^2+x-u, x in [0,1]
4 % Initial condition: u(0)=0 ;
5 % Exact solution: u(x)=-exp(-x)+x^2-x+1.
6 clear all; close all;
7 h=0.1;
8 x=0:h:1; % interval partition
9 N=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(x,u) x.^2+x-u; % RHS
12 for n=1:N
13     k1=fun(x(n),u(n));
14     k2=fun(x(n+1),u(n)+h*k1);
15     u(n+1)=u(n)+(h/2)*(k1+k2);
16 end

```



```

17 ue=-exp(-x)+x.^2-x+1;           % exact solution
18 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
19 legend('Exact','Numerical','location','northwest')
20 %title('Modified Euler Method','fontsize',12)
21 set(gca,'fontsize',12)
22 xlabel('x','fontsize',16), ylabel('u','fontsize',16)
23
24 % print -dpng -r600 ModiEuler.png
25 % print -depsc2 ModiEuler.eps

```

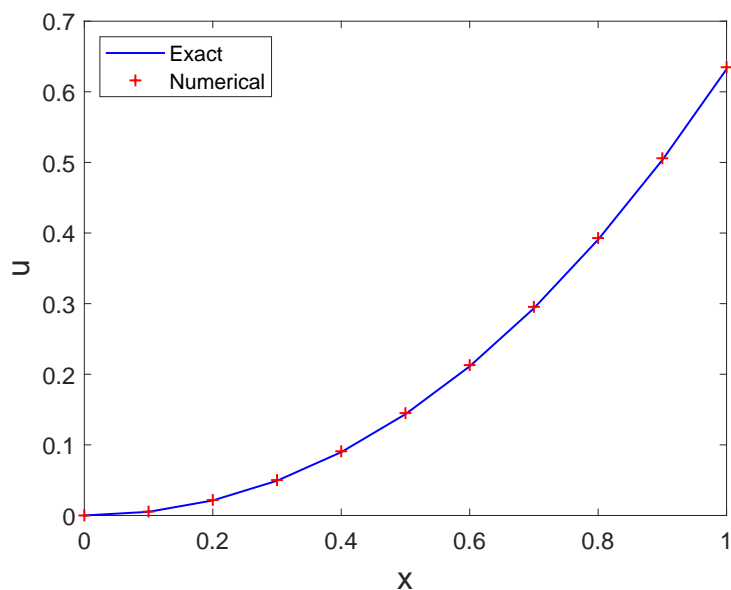


图 4: 改进 Euler 法

1.6 经典四阶 Runge-Kutta 方法

经典四阶 Runge-Kutta 方法

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

例 1.5

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & 0 < t \leq 1, \\ u(0) = 0. \end{cases} \quad (1.19)$$

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

```

1 % RungeKutta.m
2 % Runge-Kutta method for the ODE model
3 %  $u'(x)=x^2+x-u$ ,  $x$  in  $[0,1]$ 
4 % Initial condition:  $u(0)=0$  ;
5 % Exact solution:  $u(x)=-\exp(-x)+x^2-x+1$ .
6 clear all; close all;
7 h=0.1;
8 x=0:h:1; % interval partition
9 N=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(x,u) x.^2+x-u; % RHS
12 for n=1:N
13     k1=fun(x(n),u(n));
14     k2=fun(x(n)+h./2,u(n)+h.*k1/2);
15     k3=fun(x(n)+h./2,u(n)+h.*k2/2);
16     k4=fun(x(n)+h,u(n)+h.*k3);
17     u(n+1)=u(n)+h.*(k1+2.*k2+2.*k3+k4)./6;
18 end
19 ue=-exp(-x)+x.^2-x+1; % exact solution
20 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
21 legend('Exact','Numerical','location','northwest')
22 % title('Runge-Kutta Method','fontsize',12)
23 set(gca,'fontsize',12)
24 xlabel('x','fontsize',16), ylabel('u','fontsize',16)
25
26 % print -dpng -r600 RungeKutta.png
27 % print -depsc2 RungeKutta.eps

```

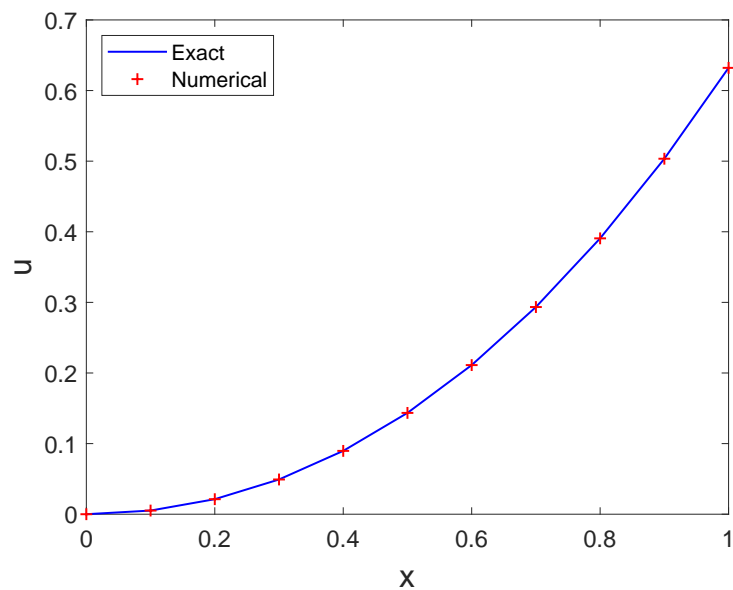


图 5: 经典四阶 Runge-Kutta 方法

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

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

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 $h = \frac{x_n - x_0}{n} \rightarrow 0$ 时 $e_n \rightarrow 0$ 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 $h \rightarrow 0$ 时有 $u_n \rightarrow u(x_n)$, 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

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

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

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

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

$$\begin{cases} \frac{du}{dt} = t^2 + t - u, & 0 < t \leq 1, \\ u(0) = 0. \end{cases}$$

方程的真解: $u(x) = e^t + t^2 - t + 1$.

```

1 % Euler1_error.m
2 % Euler method for the ODE model
3 %  $u'(x)=x^2+x-u$ ,  $x$  in  $[0,1]$ 
4 % Initial condition:  $u(0)=0$  ;
5 % Exact solution:  $u(x)=-\exp(-x)+x^2-x+1$ .
6 clear all; close all;
7 Nvec=[10 50 100 500 1000]; % Number of partitions
8 Error=[];
9 fun=@(x,u) x.^2+x-u; % RHS
10 for k=1:length(Nvec)
11     N=Nvec(k);
12     h=1/N;
13     x=0:h:1; % interval partition
14     u(1)=0; % initial value
15     for n=1:N
16         u(n+1)=u(n)+h.*fun(x(n),u(n));
17     end
18     ue=-exp(-x)+x.^2-x+1; % exact solution
19     error=max(abs(u-ue));
20     Error=[Error,error];
21 end
22 plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
23 %loglog(Nvec,Error,'ro-','LineWidth',1.5)
24 hold on
25 %loglog(Nvec, Nvec.^(-1), '--')
26 plot(log10(Nvec), log10(Nvec.^(-1)), '--')
27 grid on
28 %title('Convergence of Euler method','fontsize',12)
29 set(gca,'fontsize',12)
30 xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
31
32 % add annotation of slope
33 ax = [0.57 0.53];
34 ay = [0.68 0.63];
35 annotation('textarrow',ax,ay,'String','slope = -1 ','fontsize',14)
36
37 % computing convergence order
38 for n=1:length(Nvec)-1
39     order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
40 end
41 Error
42 order
43
44 % print -dpng -r600 Euler1_error.png
45 % print -depsc2 Euler1_error.eps

```



图 6: Euler 法收敛速度

```

1 % Trapezoidal_error.m
2 % Trapezoidal rule for the ODE model
3 % u'(x)=x^2+x-u, x in [0,1]
4 % Initial condition: u(0)=0 ;
5 % Exact solution: u(x)=-exp(-x)+x^2-x+1.
6 clear all; clf
7 Nvec=[10 50 100 500 1000]; % Number of partitions
8 Error=[];
9 for k=1:length(Nvec)
10     N=Nvec(k);
11     h=1/N;
12     x=0:h:1; % interval partition
13     N=length(x)-1;
14     u(1)=0; % initial value
15     for n=1:N
16         u(n+1)=(2-h)/(2+h).*u(n)+h/(2+h).*(x(n)^2+x(n)+x(n+1)^2+x(n+1));
17     end
18     ue=-exp(-x)+x.^2-x+1; % exact solution
19     error=max(abs(u-ue));
20     Error=[Error,error];
21 end
22 plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
23 %loglog(Nvec,Error,'ro-','LineWidth',1.5)
24 hold on
25 %loglog(Nvec, Nvec.^(-2), '--')
26 plot(log10(Nvec), log10(Nvec.^(-2)), '--')
27 grid on
28 %title('Convergence of Trapezoidal rule','fontsize',12)
29 set(gca,'fontsize',12)
30 xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
31
32 % add annotation of slope

```

```

33 ax = [0.63 0.58];
34 ay = [0.70 0.65];
35 annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)
36
37 % computing convergence order
38 for n=1:length(Nvec)-1
39     order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
40 end
41 Error
42 order
43
44 % print -dpng -r600 Trapezoidal_error.png
45 % print -depsc2 Trapezoidal_error.eps

```



图 7: 梯形公式收敛速度

```

1 % ModiEuler_error.m
2 % Modified Euler method for the ODE model
3 % u'(x)=x^2+x-u, x in [0,1]
4 % Initial condition: u(0)=0 ;
5 % Exact solution: u(x)=-exp(-x)+x^2-x+1.
6 clear all; close all;
7 Nvec=[10 50 100 500 1000]; % Number of partitions
8 Error=[];
9 fun=@(x,u) x.^2+x-u; % RHS
10 for k=1:length(Nvec)
11     N=Nvec(k);
12     h=1/N;
13     x=0:h:1; % interval partition
14     u(1)=0; % initial value
15     for n=1:N
16         k1=fun(x(n),u(n));
17         k2=fun(x(n+1),u(n)+h*k1);

```

```

18     u(n+1)=u(n)+(h/2)*(k1+k2);
19     end
20     ue=-exp(-x)+x.^2-x+1;           % exact solution
21     error=max(abs(u-ue));
22     Error=[Error,error];
23     end
24     plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
25     %loglog(Nvec,Error,'ro-','LineWidth',1.5)
26     hold on
27     %loglog(Nvec, Nvec.^(-2), '--')
28     plot(log10(Nvec),log10(Nvec.^(-2)),'--')
29     grid on
30     %title('Convergence of Trapezoidal rule','fontsize',12)
31     set(gca,'fontsize',12)
32     xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
33
34     % add annotation of slope
35     ax = [0.58 0.53];
36     ay = [0.68 0.63];
37     annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)
38
39     % computing convergence order
40     for n=1:length(Nvec)-1
41         order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
42     end
43     Error
44     order
45
46     % print -dpng -r600 ModiEuler_error.png
47     % print -depsc2 ModiEuler_error.eps

```

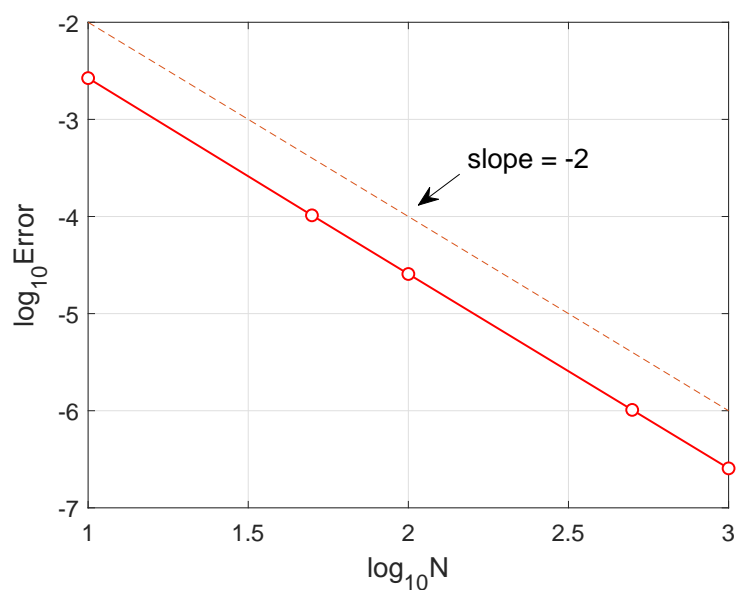


图 8: 改进 Euler 法收敛速度

```

1 % RungeKutta_error.m
2 % Runge-Kutta method for the ODE model
3 %  $u'(t)=t^2+t-u$ ,  $t \in [0,2]$ 
4 % Initial value :  $u(0)=0$  ;
5 % Exact solution :  $u(t)=-\exp(-t)+t^2-t+1$ .
6 clear all; close all;
7 Nvec=[10 50 100 500 1000]; % Number of partitions
8 Error=[];
9 fun=@(t,u) t.^2+t-u; % RHS
10 for k=1:length(Nvec)
11     N=Nvec(k);
12     h=1/N;
13     x=0:h:1; % interval partition
14     u(1)=0; % initial value
15     for n=1:N
16         k1=fun(x(n),u(n));
17         k2=fun(x(n)+h./2,u(n)+h.*k1/2);
18         k3=fun(x(n)+h./2,u(n)+h.*k2/2);
19         k4=fun(x(n)+h,u(n)+h.*k3);
20         u(n+1)=u(n)+h.*(k1+2.*k2+2.*k3+k4)./6;
21     end
22     ue=-exp(-x)+x.^2-x+1; % exact solution
23     error=max(abs(u-ue));
24     Error=[Error,error];
25 end
26 plot(log10(Nvec),log10(Error),'ro-', 'MarkerFaceColor','w', 'LineWidth',1)
27 hold on
28 plot(log10(Nvec),log10(Nvec.^(-4)), '--')
29 grid on
30 %title('Convergence of Runge-Kutta Method','fontsize',12)
31 set(gca, 'fontsize',12)
32 xlabel('log_{10}N','fontsize',16), ylabel('log_{10}Error','fontsize',16)
33
34 % add annotation of slope
35 ax = [0.57 0.53];
36 ay = [0.68 0.63];
37 annotation('textarrow',ax,ay,'String','slope = -4 ', 'fontsize',14)
38
39 % computing convergence order
40 for n=1:length(Nvec)-1
41     order(n)=-log(Error(n)/Error(n+1))/(log(Nvec(n)/Nvec(n+1)));
42 end
43 Error
44 order
45
46 % print -dpng -r600 RungeKutta_error.png
47 % 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

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

一般 s 级 Runge-Kutta 方法:

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

对应的 Butcher 阵:

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

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

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

形式 (I) 与形式 (II) 等价, 可以通过变换 $k_i = f(x_n + c_i h, Y_i), i = 1, 2, \dots, s$ 得到.

Gauss Method 2 级 4 阶

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

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

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0, \\ f_2(x_1, x_2, \dots, x_n) = 0, \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0, \end{cases}$$

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

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

方程组的牛顿迭代法

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{F}'(\mathbf{x}^{(k)})^{-1} \mathbf{F}(\mathbf{x}^{(k)}), \quad k = 0, 1, \dots,$$

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

例 1.6

$$\begin{cases} \frac{du}{dt} = f(t, u), & 0 < t \leq 1, \\ u(0) = 1. \end{cases} \quad (1.21)$$

其中: $f(t, u) = u$. 方程的真解: $u(x) = e^x$.

```
1 % IRK2s_error.m
2 % Implicit Runge-Kutta(Gauss method) 2 stage and order 4
3 % u'=u in [0,1] with initial condition u(0)=1
4 % exact solution: ue=exp(x)
5 clear all; close all;
6 Nvec=[10 50 100 200 500 1000];
7 Error=[];
8 for n=1:length(Nvec)
9     N=Nvec(n);
10    h=1/N;
```

```

11     x=[0:h:1];
12     u(1)=1;
13     Y=[1;1];
14     % Newton iteration
15     for i=1:N
16         k=u(i); tol=1;
17         while tol>1.0e-10
18             X=Y;
19             D=[1-0.25*h,-h*(0.25-(sqrt(3))/6);...
20               -h*( 0.25+(sqrt(3))/6),1-h*0.25]; % Jacobian matrix
21             F=[X(1)-k-h*(0.25*X(1)+(0.25-(sqrt(3))/6)*X(2));...
22               X(2)-k-h*((0.25+(sqrt(3))/6)*X(1)+0.25*X(2))]; % RHS
23             Y=X-D\F;
24             tol=norm(Y-X);
25         end
26         u(i+1)=k+(h/2)*(Y(1)+Y(2));
27     end
28     ue=exp(x); % exact solution
29     error=max(abs(u-ue)); % maximum error
30     Error=[Error,error];
31 end
32 plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
33 hold on,
34 plot(log10(Nvec), log10(Nvec.^(-4)), '--')
35 grid on,
36 % title('Convergence order of Gauss method ','fontsize',12)
37 set(gca,'fontsize',12)
38 xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
39
40 % add annotation of slope
41 ax = [0.62 0.58];
42 ay = [0.72 0.66];
43 annotation('textarrow',ax,ay,'String','slope = -4 ','fontsize',14)
44
45 % computing convergence order
46 for i=1:length(Nvec)-1 % computing convergence order
47     order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
48 end
49 Error
50 order
51
52 % print -dpng -r600 IRK2s_error.png
53 % print -depsc2 IRK2s_error.eps

```



图 10: Gauss Method 关于 (1.21) 的收敛速度

例 1.7

$$\begin{cases} \frac{du}{dt} = f(t, u), & 0 < t \leq 1, \\ u(0) = 0. \end{cases} \quad (1.22)$$

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

```

1 % IRK2s2_error.m
2 % Implicit Runge-Kutta(Gauss method) 2 stage and order 4
3 % u'=u^2/2-1/2 in [0,1] with initial condition u(0)=0
4 % exact solution: ue=(1-exp(x))/(1+exp(x))
5 clear all; close all;
6 Nvec=[10 50 100 200 500];
7 Error=[];
8 for n=1:length(Nvec)
9     N=Nvec(n);
10    h=1/N;
11    x=[0:h:1];
12    u(1)=0;
13    Y=[1;1];
14    % Newton iteration
15    for i=1:N
16        k=u(i); tol=1;
17        while tol>1.0e-10
18            X=Y;
19            D=[1-h*0.25*X(1), -h*(0.25-(sqrt(3))/6)*X(2);...
20              -h*(0.25+(sqrt(3))/6)*X(1), 1-h*0.25*X(2)];
21            F=[X(1)-k-h*(0.25*(0.5*(X(1))^2-0.5)+(0.25-(sqrt(3))/6)*(0.5*(X
22              (2))^2-0.5));...
23              X(2)-k-h*((0.25+(sqrt(3))/6)*(0.5*(X(1))^2-0.5)+0.25*(0.5*(X
24              (2))^2-0.5))];
25            Y=X-D\F;
26        end
27    end
28    Error=[Error; log10(Error)];
29 end

```

```

24     tol=norm(Y-X);
25     end
26     u(i+1)=k+(h/2)*(0.5*Y(1)^2-0.5+0.5*Y(2)^2-0.5);
27     end
28     ue=(1-exp(x))./(1+exp(x)); % exact solution
29     error=max(abs(u-ue)); % maximum error
30     Error=[Error,error];
31     end
32     plot(log10(Nvec),log10(Error),'ro-', 'MarkerFaceColor','w','LineWidth',1)
33     hold on,
34     plot(log10(Nvec),log10(Nvec.^(-4)),'--')
35     grid on,
36     % title('Convergence order of Gauss method ','fontsize',12)
37     set(gca,'fontsize',12)
38     xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
39
40     % add annotation of slope
41     ax = [0.62 0.58];
42     ay = [0.72 0.66];
43     annotation('textarrow',ax,ay,'String','slope = -4 ','fontsize',14)
44
45     % computing convergence order
46     for i=1:length(Nvec)-1 % computing convergence order
47         order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
48     end
49     Error
50     order
51
52     % print -dpng -r600 IRK2s2_error.png
53     % print -depsc2 IRK2s2_error.eps

```

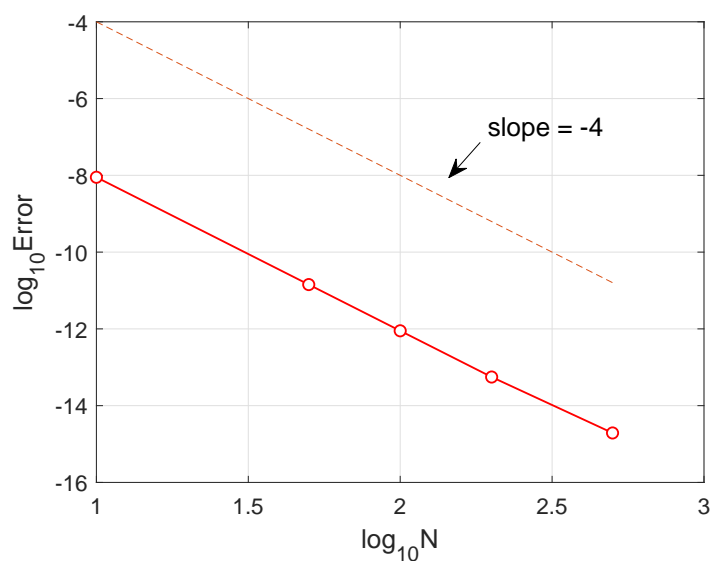


图 11: Gauss Method 关于 (1.22) 的收敛速度

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):

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

其中 q, f 为 $[a, b]$ 上的连续函数, $q \geq 0$; 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

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

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:

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

其中 $q_i = q(x_i)$, $f_i = f(x_i)$.

以上差分方程对于 $i = 1, 2, \dots, N-1$ 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:

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

它的解 u_i 是 $u(x)$ 在 $x = x_i$ 处的差分解.

先定义向量 \mathbf{u} :

$$\mathbf{u} = (u_1, u_2, \dots, u_{N-1})^T.$$

差分格式可以写为矩阵形式:

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

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

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

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

例 2.1

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

方程的真解: $u(x) = \sin(\pi x)$.

```

1 % fdm1d1.m
2 % finite difference method for 1D problem
3 % -u''+u'=pi^2*sin(pi*x)+pi*cos(pi*x) in [0,1]
4 % u(0)=0, u(1)=0 ;
5 % exact solution: u=sin(pi*x)
6 clear all; close all;
7 h=0.05;
8 x=0:h:1;
9 N=length(x)-1;
10 A=diag((2/h^2)*ones(N-1,1))...
11     +diag((1/(2*h)-1/h^2)*ones(N-2,1),1)...
12     +diag((-1/(2*h)-1/h^2)*ones(N-2,1),-1);
13 b=pi^2*sin(pi*x(2:N))+pi*cos(pi*x(2:N));
14 u=A\b';
15 u=[0;u;0];
16 ue=sin(pi*x)';
17 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
18 Error=max(abs(u-ue))
19 legend('Exact ', 'Numerical', 'location', 'NorthEast')
20 %title('Finite Difference Method','fontsize',12)
21 set(gca, 'fontsize',12)
22 xlabel('x','fontsize', 16), ylabel('u','fontsize',16)
23
24 % print -dpng -r600 fdm1d1.png
25 % print -depsc2 fdm1d1.eps

```

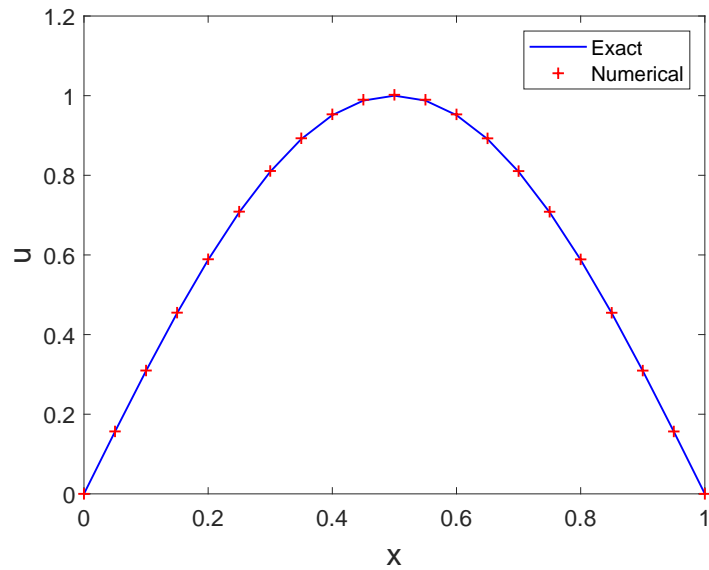


图 12: 一维差分格式

```

1 % fdm1d1_error.m
2 % finite difference method for 1D problem
3 % -u''+u'=pi^2*sin(pi*x)+pi*cos(pi*x) in [0,1]
4 % u(0)=0, u(1)=0 ;
5 % exact solution : u=sin(pi*x)
6 clear all; close all;
7 Nvec=[10 50 100 500 1000]; % Number of partitions
8 Error=[];
9 for k=1:length(Nvec)
10     N=Nvec(k);
11     h=1/N;
12     x=0:h:1;
13     N=length(x)-1;
14     A=diag((2/h^2)*ones(N-1,1))...
15         +diag((1/(2*h)-1/h^2)*ones(N-2,1),1)...
16         +diag((-1/(2*h)-1/h^2)*ones(N-2,1),-1);
17     b=pi^2*sin(pi*x(2:N))+pi*cos(pi*x(2:N));
18     u=A\b';
19     u=[0;u;0];
20     ue=sin(pi*x');
21     error=max(abs(u-ue));
22     Error=[Error,error];
23 end
24 plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
25 hold on
26 plot(log10(Nvec), log10(Nvec.^(-2)), '--')
27 grid on
28 %title('Convergence of Finite Difference Method','fontsize',12)
29 set(gca,'fontsize',12)
30 xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14),
31
32 % add annotation of slope

```



```

33 ax = [0.55 0.52];
34 ay = [0.67 0.62];
35 annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)
36
37 % computing convergence order
38 for i=1:length(Nvec)-1
39     order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
40 end
41 Error
42 order
43
44 % print -dpng -r600 fdm1d1_error.png
45 % print -depsc2 fdm1d1_error.eps

```

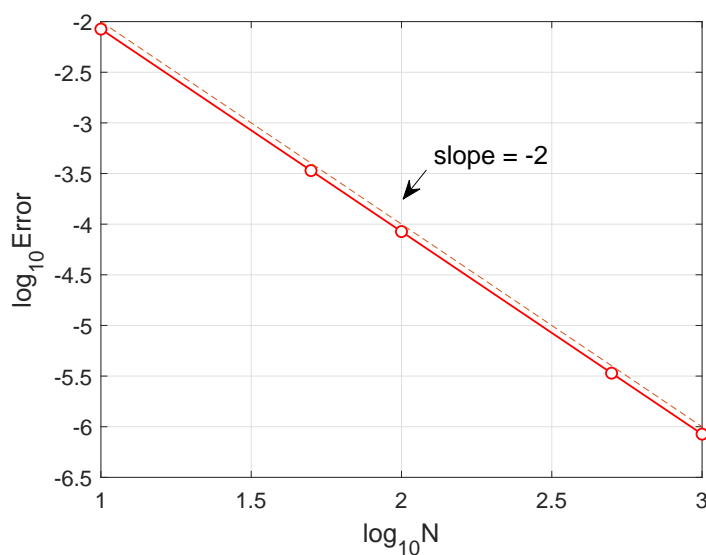


图 13: 一维差分格式收敛速度

二阶 BVP (变系数)

考虑二阶常微分方程边值问题 (变系数):

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

假定 $p \in C^1[a, b]$, $p(x) \geq p_{\min} > 0$, $r, q, f \in C[a, b]$, α, β 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:

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

记 $h_i = x_i - x_{i-1}$, 称 $h = \max_i h_i$ 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:

$$L_h u_i \equiv -\frac{2}{h_i + h_{i+1}} \left[p_{i+\frac{1}{2}} \frac{u_{i+1} - u_i}{h_{i+1}} - p_{i-\frac{1}{2}} \frac{u_i - u_{i-1}}{h_i} \right] + \frac{r_i}{h_i + h_{i+1}} (u_{i+1} - u_{i-1}) + q_i u_i = f_i, \quad i = 1, \dots, N-1,$$

$$u_0 = \alpha, \quad u_N = \beta.$$

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

$$L_h u_i = -\frac{1}{h^2} \left[p_{i+\frac{1}{2}} u_{i+1} - \left(p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}} \right) u_i + p_{i-\frac{1}{2}} u_{i-1} \right] + r_i \frac{u_{i+1} - u_{i-1}}{2h} + q_i u_i = f_i, \quad i = 1, \dots, N-1,$$

$$u_0 = \alpha, \quad u_N = \beta.$$

例 2.2

$$\begin{cases} -\frac{d}{dx} \left(x \frac{du}{dx} \right) + x \frac{du}{dx} = \pi^2 x \sin(\pi x) + \pi(x-1) \cos(\pi x), & 0 < x < 1, \\ u(0) = 0, \quad u(1) = 0. \end{cases} \quad (2.4)$$

真解: $u(x) = \sin(\pi x)$.

```

1 % fdm1d2.m
2 % finite difference method for 1D problem
3 % -(xu')'+x*u'=pi^2*x*sin(pi*x)-pi*cos(pi*x)+pi*x*cos(pi*x) in [0,1]
4 % u(0)=0, u(1)=0 ;
5 % exact solution : u=sin(pi*x)
6 clear all; close all;
7 h=0.05;
8 x=0:h:1;
9 N=length(x)-1;
10 A=diag(2*x(2:N)./h^2)+diag(x(2:N-1)./(2*h)-(x(2:N-1)+0.5*h)./h^2,1)...
11     +diag(-x(3:N)./(2*h)-(x(3:N)-0.5*h)./h^2,-1);
12 b=pi^2*x(2:N).*sin(pi*x(2:N))+pi*(x(2:N)-1).*cos(pi*x(2:N));
13 u=A\b';
14 u=[0;u;0];
15 ue=sin(pi*x');
16 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1)
17 Error=max(abs(u-ue))
18 legend('Exact ', 'Numerical', 'location', 'NorthEast')
19 %title('Finite Difference Method', 'fontsize',12)
20 set(gca, 'fontsize',12)
21 xlabel('x', 'fontsize', 16), ylabel('u', 'fontsize',16)
22
23 % print -dpng -r600 fdm1d2.png
24 % print -depsc2 fdm1d2.eps

```



图 14: 一维差分格式 (例 2.2)

```

1 % fdm1d2_error.m
2 % finite difference method for 1D problem
3 %  $-(xu')' + x*u = \pi^2 * x * \sin(\pi * x) - \pi * \cos(\pi * x) + \pi * x * \cos(\pi * x)$  in  $[0,1]$ 
4 %  $u(0)=0, u(1)=0$  ;
5 % exact solution :  $u = \sin(\pi * x)$ 
6 clear all; close all;
7 Nvec=[10 20 50 100 200 500 1000]; % Number of partitions
8 Error=[];
9 for k=1:length(Nvec)
10     N=Nvec(k);
11     h=1/N;
12     x=0:h:1;
13     N=length(x)-1;
14     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)...
15         +diag(-x(3:N)./(2*h)-(x(3:N)-0.5*h)./h^2,-1);
16     b=pi^2*x(2:N).*sin(pi*x(2:N))+pi*(x(2:N)-1).*cos(pi*x(2:N));
17     u=A\b';
18     u=[0;u;0];
19     ue=sin(pi*x');
20     error=max(abs(u-ue));
21     Error=[Error,error];
22 end
23 plot(log10(Nvec),log10(Error),'ro-', 'MarkerFaceColor','w','LineWidth',1)
24 hold on
25 plot(log10(Nvec), log10(Nvec.^(-2)), '--')
26 grid on
27 %title('Convergence of Finite Difference Method','fontsize',14)
28 set(gca,'fontsize',14)
29 xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),
30
31 % add annotation of slope

```

```

32 ax = [0.57 0.53];
33 ay = [0.68 0.63];
34 annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)
35
36 % computing convergence order
37 for i=1:length(Nvec)-1
38     order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
39 end
40 Error
41 order
42
43 % print -dpng -r600 fdm1d2_error.png
44 % 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 Poisson 方程五点差分格式的 MATLAB 编程实现.

考虑二维 Poisson 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 $\phi(x, y)$ 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 $a < x < b, c < y < d$. 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 $x_i = a + ih_1, 0 \leq i \leq N, y_j = c + jh_2, 0 \leq j \leq M$. (x_i, y_j) 称为网格节点.

二维 Poisson 方程的五点差分格式:

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

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

先定义向量 $\mathbf{u}_j = (u_{1j}, u_{2j}, \dots, u_{N-1,j})^T, 0 \leq j \leq M$.

差分格式可以写为矩阵形式:

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

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

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

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

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.

$$\begin{pmatrix} C & D & & & \\ D & C & D & & \\ & \ddots & \ddots & \ddots & \\ & & D & C & D \\ & & & D & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{M-2} \\ u_{M-1} \end{pmatrix} = \begin{pmatrix} f_1 - Du_0 \\ f_2 \\ \vdots \\ f_{M-2} \\ f_{M-1} - Du_M \end{pmatrix}.$$

例 2.3

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

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

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

```

1 % fdm2d1.m
2 % finite difference method for 2D problem
3 % -d^2u/dx^2-d^2u/dy^2=f(x,y)
4 % f(x,y)=-2*pi^2*exp(pi*(x+y))*(sin(pi*x)*cos(pi*y)+cos(pi*x)*sin(pi*y))
5 % exact solution: ue=exp(pi*x+pi*y)*sin(pi*x)*sin(pi*y)
6 clear all; close all;
7 % generate coordinates on the grid
8 h=0.02;
9 x=[0:h:1]';
10 y=[0:h:1]';
11 N=length(x)-1;
12 M=length(y)-1;
13 [X,Y]=meshgrid(x,y);
14 X=X(2:M,2:N);
15 Y=Y(2:M,2:N);
16 % generate the matrix of RHS
17 f=-2*pi^2*exp(pi*X+pi*Y).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
18 % constructing the coefficient matrix
19 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);

```

```

20 D=-1/h^2*eye(N-1);
21 A=kron(eye(M-1),C)+kron(diag(ones(M-2,1),1)+diag(ones(M-2,1),-1),D);
22 % solving the linear system
23 f=f';
24 u=zeros(M+1,N+1);
25 u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
26 u(:,1)=0;
27 u(:,end)=0;
28 ue=zeros(M+1,N+1);
29 ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
30 % compute maximum error
31 Error=max(max(abs(u-ue)))
32 mesh(x,y,u)
33 %title('Finite Difference Method','fontsize',14)
34 set(gca,'fontsize',12)
35 xlabel('x','fontsize',16)
36 ylabel('y','fontsize',16)
37 zlabel('u','fontsize',16)
38
39 % print -dpng -r600 fdm2d1.png
40 % print -depsc2 fdm2d1.eps

```

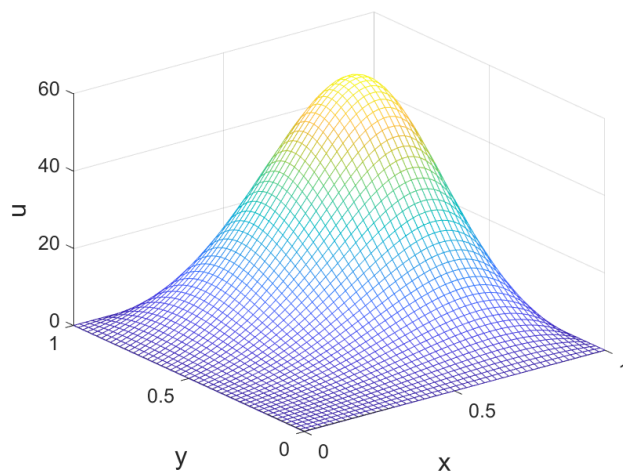


图 16: 矩形网差分格式

```

1 % fdm2d1_error.m
2 % finite difference method for 2D problem
3 % -d^2u/dx^2-d^2u/dy^2=f(x,y)
4 % f(x,y)=-2*pi^2*exp(pi*(x+y))*(sin(pi*x)*cos(pi*y)+cos(pi*x)*sin(pi*y))
5 % exact solution: ue=exp(pi*x+pi*y)*sin(pi*x)*sin(pi*y)
6 clear all; close all;
7 Nvec=2.^[4:10];
8 Error=[];
9 for n=Nvec
10     h=1/n;
11     x=[0:h:1]';

```

```

12     y=[0:h:1]';
13     N=length(x)-1;
14     M=length(y)-1;
15     [X,Y]=meshgrid(x,y);
16     X=X(2:M,2:N);
17     Y=Y(2:M,2:N);
18     % generate the matrix of RHS
19     f=-2*pi^2*exp(pi*X+pi*Y).*(sin(pi*X).*cos(pi*Y)+cos(pi*X).*sin(pi*Y));
20     % constructing the coefficient matrix
21     e=ones(N-1,1);
22     C=1/h^2*spdiags([-e 4*e -e],[-1 0 1],N-1,N-1);
23     D=-1/h^2*eye(N-1);
24     e=ones(M-1,1);
25     A=kron(eye(M-1),C)+kron(spdiags([e e],[-1 1],M-1,M-1),D);
26     % solving the linear system
27     f=f';
28     u=zeros(M+1,N+1);
29     u(2:M,2:N)=reshape(A\f(:),N-1,M-1)';
30     u(:,1)=0;
31     u(:,end)=0;
32     ue=zeros(M+1,N+1);
33     % numerical solution
34     ue(2:M,2:N)=exp(pi*X+pi*Y).*sin(pi*X).*sin(pi*Y);
35     error=max(max(abs(u-ue))); % maximum error
36     Error=[Error,error];
37     end
38     plot(log10(Nvec),log10(Error),'ro-','MarkerFaceColor','w','LineWidth',1)
39     hold on
40     plot(log10(Nvec), log10(Nvec.^(-2)), '--')
41     grid on
42     %title('Convergence of Finite Difference Method','fontsize',12)
43     set(gca,'fontsize',12)
44     xlabel('log_{10}N','fontsize',14), ylabel('log_{10}Error','fontsize',14),
45
46     % add annotation of slope
47     ax = [0.46 0.50];
48     ay = [0.41 0.46];
49     annotation('textarrow',ax,ay,'String','slope = -2 ','fontsize',14)
50
51     % computing convergence order
52     for i=1:length(Nvec)-1
53         order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
54     end
55     Error
56     order
57
58     % print -dpng -r600 fdm2d1_error.png
59     % print -depsc2 fdm2d1_error.eps

```

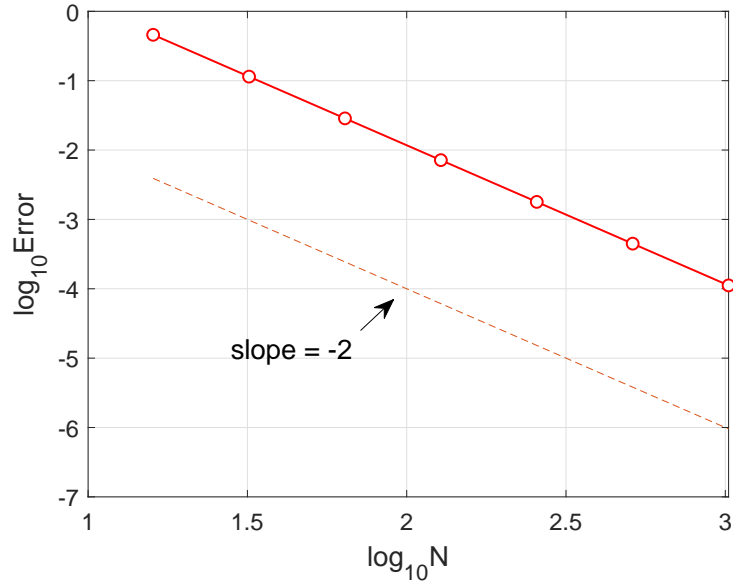



图 17: 矩形网差分格式收敛速度

例 2.4

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

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

差分方程:

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

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

边界条件:

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

离散格式:

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

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

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

矩阵形式:

$$\begin{pmatrix} C & D & & \\ D & C & D & \\ & \ddots & \ddots & \ddots \\ & & D & C & D \\ & & & D & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{M-2} \\ u_{M-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{M-2} \\ f_{M-1} \end{pmatrix}.$$

最后再利用边界条件处理边界处的值.

```

1 % fdm2d2.m
2 % finite difference method for 2D problem
3 % -\Delta u = \cos(3*x)*\sin(\pi*y) in (0,\pi) \times (0,1)
4 % u(x,0)=u(x,1)=0 in [0,\pi]
5 % u_x(0,y)=u_x(\pi,y)=0 in [0,1]
6 % exact solution: ue=(9+\pi^2)^{-1}*\cos(3*x)*\sin(\pi*y)
7 clear all; close all;
8 N=4 % N=4 8 16 32
9 % coordinates on the grid
10 h1=\pi/N; h2=1/N;
11 x=[0:h1:\pi]';
12 y=[0:h2:1]';
13 [X,Y]=meshgrid(x,y);
14 X1=X(2:N,2:N);
15 Y1=Y(2:N,2:N);
16 % generate the matrix of RHS
17 f=\cos(3*X1).*\sin(\pi*Y1);
18 % constructing the coefficient matrix
19 e=ones(N-1,1);
20 C=diag([1/h1^2+2/h2^2, (2/h1^2+2/h2^2)*ones(1,N-3), 1/h1^2+2/h2^2])...
21     -1/h1^2*diag(ones(N-2,1),1)-1/h1^2*diag(ones(N-2,1),-1);
22 D=-1/h2^2*eye(N-1);
23 % A=kron(eye(N-1),C)+kron(diag(ones(N-2,1),1)+diag(ones(N-2,1),-1),D);
24 A=kron(eye(N-1),C)+kron(spdiags([e e],[-1 1],N-1,N-1),D);
25 % solving the linear system
26 f=f';
27 u=zeros(N+1,N+1);
28 u(2:N,2:N)=reshape(A\f(:),N-1,N-1)';
29 % Neumann boundary condition
30 u(:,1)=u(:,2);
31 u(:,end)=u(:,end-1);

```

```

32 ue=1/(9+pi^2)*(cos(3*X)).*(sin(pi*Y));
33
34 format long
35 % value of u and ue in the selected points (i*pi/4,j/4), i,j=1,2,3.
36 u_select=u(N/4+1:N/4:3*N/4+1,N/4+1:N/4:3*N/4+1)
37 ue_select=ue(N/4+1:N/4:3*N/4+1,N/4+1:N/4:3*N/4+1)

```

对应书上结果:

```

1 N =
2     4
3 u_select =
4     -0.045480502664596     -0.000000000000000     0.045480502664596
5     -0.064319143691817     -0.000000000000000     0.064319143691817
6     -0.045480502664596     -0.000000000000000     0.045480502664596
7 ue_select =
8     -0.045480502664596     -0.000000000000000     0.045480502664596
9     -0.064319143691817     -0.000000000000000     0.064319143691817
10    -0.045480502664596     -0.000000000000000     0.045480502664596

```

```

1 % fdm2d2_error.m
2 % finite difference method for 2D problem
3 % -\Delta u=cos(3*x)*sin(pi*y) in (0,pi)x(0,1)
4 % u(x,0)=u(x,1)=0 in [0,pi]
5 % u_x(0,y)=u_x(pi,y)=0 in [0,1]
6 % exact solution: ue=(9+pi^2)^(-1)*cos(3*x)*sin(pi*y)
7 clear all; close all;
8 Nvec=2.^[2:7];
9 Error=[];
10 for N=Nvec
11     % coordinates on the grid
12     h1=pi/N; h2=1/N;
13     x=[0:h1:pi]';
14     y=[0:h2:1]';
15     [X,Y]=meshgrid(x,y);
16     X1=X(2:N,2:N);
17     Y1=Y(2:N,2:N);
18     % generate the matrix of RHS
19     f=cos(3*X1).*sin(pi*Y1);
20     % constructing the coefficient matrix
21     e=ones(N-1,1);
22     C=diag([1/h1^2+2/h2^2, (2/h1^2+2/h2^2)*ones(1,N-3), 1/h1^2+2/h2^2])...
23         -1/h1^2*diag(ones(N-2,1),1)-1/h1^2*diag(ones(N-2,1),-1);
24     D=-1/h2^2*eye(N-1);
25     %A=kron(eye(N-1),C)+kron(diag(ones(N-2,1),1)+diag(ones(N-2,1),-1),D);
26     A=kron(eye(N-1),C)+kron(spdiags([e e],[-1 1],N-1,N-1),D);
27     % solving the linear system
28     f=f';

```

```

29     u=zeros(N+1,N+1);
30     u(2:N,2:N)=reshape(A\f(:),N-1,N-1)';
31     % Neumann boundary condition
32     u(:,1)=u(:,2);
33     u(:,end)=u(:,end-1);
34     ue=1/(9+pi^2)*(cos(3*X)).*(sin(pi*Y));
35     error=max(max(abs(u-ue))); % maximum error
36     Error=[Error,error];
37 end
38 plot(log10(Nvec),log10(Error),'ro-', 'MarkerFaceColor','w','LineWidth',1)
39 hold on
40 plot(log10(Nvec), log10(Nvec.^(-1)), '--')
41 grid on
42 %title('Convergence of Finite Difference Method','fontsize',14)
43 set(gca,'fontsize',14)
44 xlabel('log_{10}N','fontsize', 14), ylabel('log_{10}Error','fontsize',14),
45
46 % add annotation of slope
47 ax = [0.64 0.60];
48 ay = [0.69 0.64];
49 annotation('textarrow',ax,ay,'String','slope = -1 ','fontsize',14)
50
51 % computing convergence order
52 for i=1:length(Nvec)-1
53     order(i)=-log(Error(i)/Error(i+1))/(log(Nvec(i)/Nvec(i+1)));
54 end
55 Error
56 order
57
58 % print -dpng -r600 fdm2d2_error.png
59 % print -depsc2 fdm2d2_error.eps

```

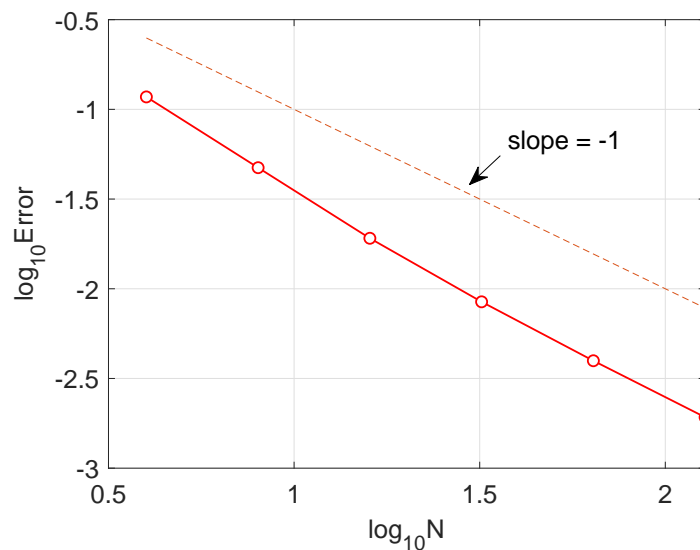


图 18: 矩形网差分格式收敛速度 (例 (2.4))

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:

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

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

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

一维热传导方程的向前差分格式

$$\begin{aligned} \frac{u_j^{n+1} - u_j^n}{\tau} &= a \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} + f(x_j, t_n), \\ u_j^0 &= \phi_j = \phi(x_j), u_0^n = u_M^n = 0, \end{aligned}$$

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

以上格式可改写差分格式

$$\begin{aligned} u_j^{n+1} &= ru_{j+1}^n + (1-2r)u_j^n + ru_{j-1}^n + \tau f_j, \\ 1 &\leq j \leq M-1, 0 \leq n \leq N-1. \end{aligned}$$

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 $\mathbf{u}^n = (u_1^n, u_2^n, \dots, u_{M-2}^n, u_{M-1}^n)^T$, $0 \leq n \leq N$.

差分格式写成矩阵的形式:

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

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

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

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

例 2.5

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

方程的真解: $u(x, t) = e^{x+t}$.

```

1 % fdm_heat.m
2 % forward difference scheme for heat equation
3 % u_t=u_{xx}, (x,t) in (0,1)x(0,1],
4 % u(x,0)=exp(x), x in [0,1],
5 % u(0,t)=exp(t), u(1,t)=exp(1+t), t in (0,1]
6 % exact solution: u(x,t)=exp(x+t)
7 clear all; close all;
8 a=1;
9 h=0.05; x=[0:h:1];
10 tau=0.00125; t=[0:tau:1];
11 r=a*tau/h^2;
12 M=length(x)-1; N=length(t)-1;
13 % constructing the coefficient matrix
14 e=r*ones(M-1,1);
15 A=spdiags([e 1-2*e e],[-1 0 1],M-1,M-1);
16 % setting initial and boundary conditions
17 u=zeros(M+1,N+1);
18 u(:,1)=exp(x);
19 u(1,:)=exp(t);
20 u(end,:)=exp(1+t);
21 for n=1:N
22     u(2:M,n+1)=A*u(2:M,n);
23     u(2,n+1)=u(2,n+1)+r*u(1,n);
24     u(M,n+1)=u(M,n+1)+r*u(end,n);
25 end

```

```

26 % plot the figure
27 mesh(t(1:20:end),x,u(:,1:20:end))
28 set(gca,'fontsize',12)
29 xlabel('t','fontsize',14)
30 ylabel('x','fontsize',14)
31 zlabel('u','fontsize',14)
32
33 % calculating maximum error
34 [T X]=meshgrid(t,x);
35 ue=exp(X+T);
36 Error=max(max(abs(ue-u)))
37
38 % print -dpng -r600 fdm_heat.png
39 % print -depsc2 fdm_heat.eps

```

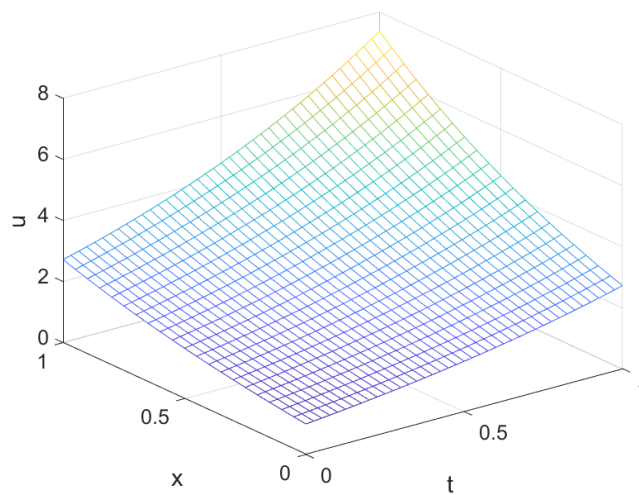


图 19: 一维热传导方程的差分格式

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:

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

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

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

一维热传导方程的六点对称格式 (Crank-Nicolson 格式)

$$\begin{aligned} \frac{u_j^{n+1} - u_j^n}{\tau} &= \frac{a}{2} \left[\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} \right] + \frac{f(x_j, t_{n+1}) + f(x_j, t_n)}{2}, \\ u_j^0 &= \phi_j = \phi(x_j), \quad u_0^n = u_M^n = 0. \end{aligned}$$

以上格式可改写为

$$\begin{aligned} -\frac{r}{2}u_{j+1}^{n+1} + (1+r)u_j^{n+1} - \frac{r}{2}u_{j-1}^{n+1} &= \frac{r}{2}u_{j+1}^n + (1-r)u_j^n + \frac{r}{2}u_{j-1}^n + \frac{\tau}{2}[f(x_j, t_{n+1}) + f(x_j, t_n)], \\ 1 \leq j \leq M-1, \quad 0 \leq n \leq N-1. \end{aligned}$$

即

$$\begin{aligned} -ru_{j+1}^{n+1} + (2+2r)u_j^{n+1} - ru_{j-1}^{n+1} &= ru_{j+1}^n + (2-2r)u_j^n + ru_{j-1}^n + \tau f(x_j, t_{n+1}) + \tau f(x_j, t_n), \\ 1 \leq j \leq M-1, \quad 0 \leq n \leq N-1. \end{aligned}$$

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 $\mathbf{u}^n = (u_1^n, u_2^n, \dots, u_{M-2}^n, u_{M-1}^n)^T$, $0 \leq n \leq N$.

差分格式可以写为矩阵形式:

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

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

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

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

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

例 2.6

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

其中 $f(x, t) = \cos(x+t) + \sin(x+t)$.

方程的真解 $u(x, t) = \sin(x+t)$.

```

1 % fdm_cn.m
2 % Crank-Nicolson scheme for heat equation
3 % u_t=u_{xx}+f(x,t), (x,t) in (0,1)x(0,1],
4 % u(x,0)=sin(x), x in [0,1],
5 % u(0,t)=sin(t), u(1,t)=sin(1+t), t in (0,1].
6 % f(x,t)=cos(x+t)+sin(x+t),
7 % exact solution: u(x,t)=sin(x+t)
8 clear all; close all;
9 a=1;
10 h=0.05; x=[0:h:1];
11 tau=0.001; t=[0:tau:1];
12 r=a*tau/h^2;
13 M=length(x)-1; N=length(t)-1;
14 % constructing the coefficient matrix
15 e=r*ones(M-1,1);
16 A=spdiags([-e 2+2*e -e],[-1 0 1],M-1,M-1);
17 B=spdiags([e 2-2*e e],[-1 0 1],M-1,M-1);

```

```

18 % setting initial and boundary conditions
19 u=zeros(M+1,N+1);
20 u(:,1)=sin(x);
21 u(1,:)=sin(t);
22 u(end,:)=sin(1+t);
23 for n=1:N
24     F=tau*cos(x(2:M)'+t(n))+tau*sin(x(2:M)'+t(n))...
25         +tau*cos(x(2:M)'+t(n+1))+tau*sin(x(2:M)'+t(n+1));
26     F(1)=F(1)+r*u(1,n)+r*u(1,n+1);
27     F(M-1)=F(M-1)+r*u(end,n)+r*u(end,n+1);
28     % solving the system
29     u(2:M,n+1)=A\B*u(2:M,n)+A\F;
30 end
31 % plot the figure
32 mesh(t(1:20:end),x,u(:,1:20:end))
33 set(gca,'fontsize',12)
34 xlabel('t','fontsize',14)
35 ylabel('x','fontsize',14)
36 zlabel('u','fontsize',14)
37
38 % calculating maximum error
39 [T X]=meshgrid(t,x);
40 ue=sin(X+T);
41
42 Error=max(max(abs(ue-u)))
43
44 % print -dpng -r600 fdm_cn.png
45 % print -depsc2 fdm_cn.eps

```

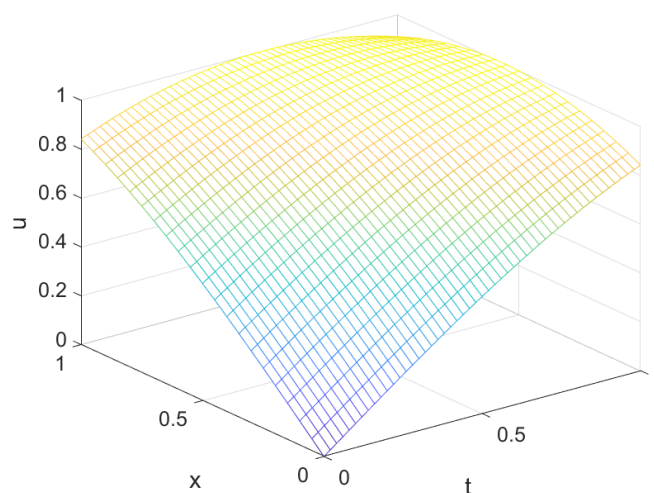


图 20: 一维热传导方程的 C-N 格式

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:

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

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

$$x_i = jh, \quad 0 \leq j \leq M,$$

$$t_n = n\tau, \quad 0 \leq n \leq N.$$

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

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

以上格式可改写差分格式

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

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

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

设 $\mathbf{u}^n = (u_1^n, u_2^n, \dots, u_{M-2}^n, u_{M-1}^n)^T$, $0 \leq n \leq N$.

差分格式写成矩阵的形式:

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

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

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

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

注意：计算最开始时需要两个已知向量 \mathbf{u}^0 和 \mathbf{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

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

方程的真解： $u(x, t) = \sin(xt)$ 。

```

1 % fdm_wave.m
2 % finite difference method for wave equation
3 % u_{tt}=u_{xx}+f(x,t), (x,t) in (0,1)x(0,1],
4 % u(x,0)=0, u_t(x,0)=x, x in [0,1],
5 % u(0,t)=0, u(1,t)=sin(t), t in (0,1].
6 % f(x,t)=cos(x+t)+sin(x+t),
7 % exact solution: u(x,t)=sin(x+t)
8 clear all; close all;
9 a=1;
10 h=0.05; x=[0:h:1];
11 tau=0.05; t=[0:tau:1];
12 r=a*tau/h;
13 M=length(x)-1; N=length(t)-1;
14 [T X]=meshgrid(t,x);
15 % constructing the coefficient matrix
16 e=r^2*ones(M-1,1);
17 A=spdiags([e 2*(1-e) e],[-1 0 1],M-1,M-1);
18 % setting initial and boundary conditions
19 u=zeros(M+1,N+1);
20 u(:,1)=0; u(:,2)=tau*x;
21 u(1,:)=0; u(end,:)=sin(t);
22 for n=2:N
23     u(2:M,n+1)=A*u(2:M,n)-u(2:M,n-1)+ ...
24         tau^2*(T(2:M,n).^2-X(2:M,n).^2).*sin(X(2:M,n).*T(2:M,n));
25     u(2,n+1)=u(2,n+1)+r^2*u(1,n);
26     u(M,n+1)=u(M,n+1)+r^2*u(end,n);
27 end
28 % plot the figure
29 mesh(t,x,u), view(20,40)
30 set(gca,'fontsize',12)

```

```

31 xlabel('t','fontsize',14)
32 ylabel('x','fontsize',14)
33 zlabel('u','fontsize',14)
34
35 % calculating maximum error
36 ue=sin(X.*T);
37 Error=max(max(abs(ue-u)))
38
39 % print -dpng -r600 fdm_wave.png
40 % print -depsc2 fdm_wave.eps

```

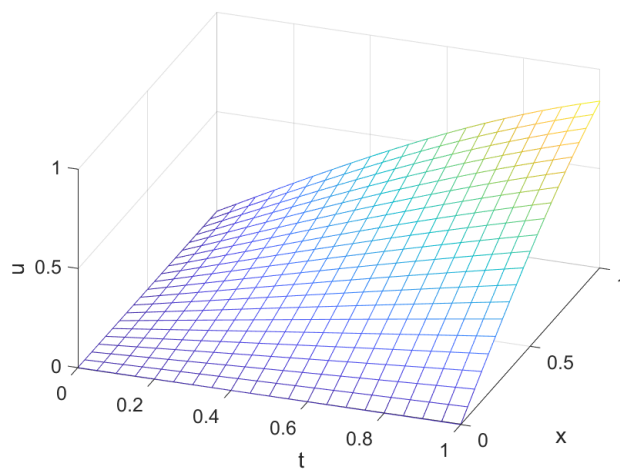


图 21: 一维波动方程的差分格式

例 2.8

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

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.

```

1 % fdm_wave2.m
2 % finite difference method for wave equation
3 % u_{tt}=u_{xx}, (x,t) in (0,1)x(0,T],
4 % u(x,0)=0, u_t(x,0)=x, x in [0,1],
5 % u(0,t)=0, u(1,t)=sin(t), t in (0,1].
6 % exact solution: u(x,t)=sin(4*pi*x)*cos(4*pi*t)
7 % +sin(8*pi*x)*sin(8*pi*t)/(8*pi);
8 clear all; close all;
9 a=1;
10 T=1; % time
11 h=0.0025; x=[0:h:1];
12 tau=0.0025; t=[0:tau:T];

```

```

13 r=a*tau/h;
14 M=length(x)-1; N=length(t)-1;
15 [T X]=meshgrid(t,x);
16 % constructing the coefficient matrix
17 e=r^2*ones(M-1,1);
18 A=spdiags([e 2*(1-e) e],[-1 0 1],M-1,M-1);
19 % setting initial and boundary conditions
20 u=zeros(M+1,N+1);
21 u(:,1)=sin(4*pi*x');
22 u(:,2)=sin(4*pi*x')+tau*sin(8*pi*x');
23 u(1,:)=0; u(end,:)=0;
24 for n=2:N
25     u(2:M,n+1)=A*u(2:M,n)-u(2:M,n-1);
26     u(2,n+1)=u(2,n+1)+r^2*u(1,n);
27     u(M,n+1)=u(M,n+1)+r^2*u(end,n);
28 end
29 % plot the figure
30 mesh(t,x,u), view(20,40)
31 set(gca,'fontsize',12)
32 xlabel('t','fontsize',14)
33 ylabel('x','fontsize',14)
34 zlabel('u','fontsize',14)
35
36 % calculating maximum error
37 ue=sin(4*pi*X).*cos(4*pi*T)+sin(8*pi*X).*sin(8*pi*T)/(8*pi);
38
39 disp('t = 1, 2, 3, 4, 5')
40 Error=max(abs(ue(:,1/tau+1:1/tau:end))-u(:,1/tau+1:1/tau:end)))
41
42 % print -dpng -r600 fdm_wave2.png
43 % 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 $\{0 \leq r < \infty, 0 \leq \theta \leq 2\pi\}$.

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 $u(0, \theta) = u(r, \theta) - ru_r(r, \theta) + O(r^2)$, $r \rightarrow 0$, 则知

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

差分格式

对变量 r, θ 分别取等步长 h_r 和 h_θ . 令

$$r_i = (i + 0.5)h_r, \quad i = 0, 1, 2, \dots, N-1,$$

$$\theta_j = (j + 1)h_\theta, \quad j = 0, 1, \dots, M-1, \quad h_\theta = 2\pi/M.$$

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

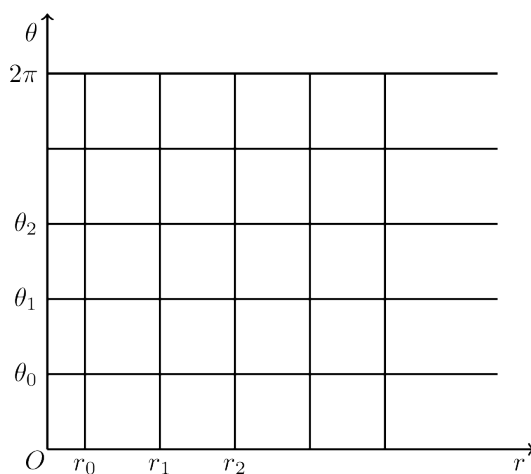


图 23: 半带形域的网格剖分

差分方程:

$$\begin{aligned} & -\frac{1}{r_i} \frac{r_{i+\frac{1}{2}} u_{i+1,j} - (r_{i+\frac{1}{2}} + r_{i-\frac{1}{2}}) u_{ij} + r_{i-\frac{1}{2}} u_{i-1,j}}{h_r^2} \\ & - \frac{1}{r_i^2} \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{h_\theta^2} = f(r_i, \theta_j), \quad 1 \leq i \leq N-1. \end{aligned}$$

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 ($i = 0, 1, \dots, N-1$) 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:

$$\begin{aligned} & \underbrace{-\frac{1}{r_i^2} \frac{1}{h_\theta^2} u_{i,j+1}}_{e_i} - \underbrace{\frac{1}{r_i} \frac{r_{i+\frac{1}{2}}}{h_r^2} u_{i+1,j}}_{\beta_i} + \underbrace{\frac{1}{r_i} \frac{r_{i+\frac{1}{2}} + r_{i-\frac{1}{2}}}{h_r^2} u_{ij}}_{c_i} + \underbrace{\frac{1}{r_i^2} \frac{2}{h_\theta^2} u_{ij}}_{\alpha_i} \\ & - \underbrace{\frac{r_{i+\frac{1}{2}}}{r_i h_r^2} u_{i+1,j}}_{\alpha_i} - \underbrace{\frac{1}{r_i^2} \frac{1}{h_\theta^2} u_{i,j+1}}_{d_i} = f_{ij} \quad (i \geq 1). \\ & \underbrace{-\frac{4}{h_r^2 h_\theta^2} u_{0,j+1}}_{e_0} + \underbrace{\frac{2}{h_r^2} u_{0,j}}_{c_0} + \underbrace{\frac{8}{h_r^2 h_\theta^2} u_{0j}}_{\alpha_0} - \underbrace{\frac{2}{h_r^2} u_{1j}}_{\alpha_0} - \underbrace{\frac{4}{h_r^2 h_\theta^2} u_{0,j+1}}_{d_0} = f_{0j}. \end{aligned}$$

$$\begin{aligned} C &= \begin{pmatrix} c_0 & \alpha_0 & & & \\ \beta_1 & c_1 & \alpha_1 & & \\ & \beta_2 & c_2 & \alpha_2 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{N-2} & c_{N-2} & \alpha_{N-2} \\ & & & & \beta_{N-1} & c_{N-1} \end{pmatrix} & D &= \begin{pmatrix} d_0 & & & & \\ & d_1 & & & \\ & & d_2 & & \\ & & & \ddots & \\ & & & & d_{N-2} \\ & & & & & d_{N-1} \end{pmatrix} \\ E &= \begin{pmatrix} e_0 & & & & \\ & e_1 & & & \\ & & e_2 & & \\ & & & \ddots & \\ & & & & e_{N-2} \\ & & & & & e_{N-1} \end{pmatrix} & f_j &= \begin{pmatrix} f_{0j} \\ f_{1j} \\ \vdots \\ f_{N-2,j} \\ f_{N-1,j} \end{pmatrix}. \end{aligned}$$

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

差分格式写成矩阵形式:

$$D U_{j+1} + C U_j + D U_{j-1} = f_j, \quad 0 \leq j \leq M$$

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:

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

方程的真解:

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

```

1 % fdm_polar.m
2 % finite difference method for polar coordinate problem
3 %  $-\Delta u(r, \theta) = f(r, \theta)$ ,  $[0, 1] \times [0, 2\pi]$ 
4 %  $u(1, \theta) = 0$ ,  $\theta$  in  $[0, 2\pi]$ 
5 % exact solution:  $u = (1 - r^2)r(\sin(\theta) + \cos(\theta))/4$ .
6 clear all; close all;
7 N=50;
8 M=100;
9 dr=1/(N+1/2);
10 % dr=0.02;
11 dthe=2*pi/M;
12 r=(dr/2:dr:1)';
13 the=(0:dthe:2*pi)';
14 % N=length(r)-1;
15 % M=length(the);
16 % generate coordinates on the grid
17 [The, R]=meshgrid(the, r);
18 % generate the matrix of RHS
19 The1=The(1:N, 2:end); R1=R(1:N, 2:end);
20 f=2*R1.*(sin(The1)+cos(The1));
21 f=f';
22 % constructing the coefficient matrix
23 e=[2/dr^2+8/(dr^2*dthe^2); (2./(dthe^2*r(2:N).^2)+2/dr^2)];
24 e1=[-2/dr^2; -(r(2:N-1)+dr/2)./(dr^2*r(2:N-1))];
25 e2=[-(r(2:N)-dr/2)./(dr^2*r(2:N))];
26 C=diag(e)+diag(e1, 1)+diag(e2, -1);
27 D=diag([-4/(dr^2*dthe^2); -1./(dthe^2*r(2:N).^2)]);
28 E=diag([-4/(dr^2*dthe^2); -1./(dthe^2*r(2:N).^2)]);
29 A=kron(eye(M), C)+kron(diag(ones(M-1, 1), 1)+diag(1, 1-M), D)...
30     +kron(diag(ones(M-1, 1), -1)+diag(1, M-1), E);
31 % solving the linear system
32 f=f';
33 % u=zeros(M+1, N+1);
34 uh=reshape(A\f(:), N, M);
35 un=[uh; zeros(1, M)];
36 un=[un(:, end), un];
37 ue=(1-R.^2).*R.*(sin(The)+cos(The))/4;
38 % error on the node of mesh
39 Err=abs(un(1:N, 2:end)-ue(1:N, 2:end));
40 % compute maximum error
41 MaxErr=max(max(abs(un-ue)))

```

```

42 % plot the figure
43 [X,Y] = pol2cart(The,R);
44 mesh(X,Y,ue)
45 set(gca,'fontsize',12)
46 xlabel('x','fontsize',16)
47 ylabel('y','fontsize',16)
48 zlabel('u','fontsize',16)
49 view(36,24)
50
51 % print -dpng -r600 fdm_polar.png
52 % print -depsc2 fdm_polar.eps

```

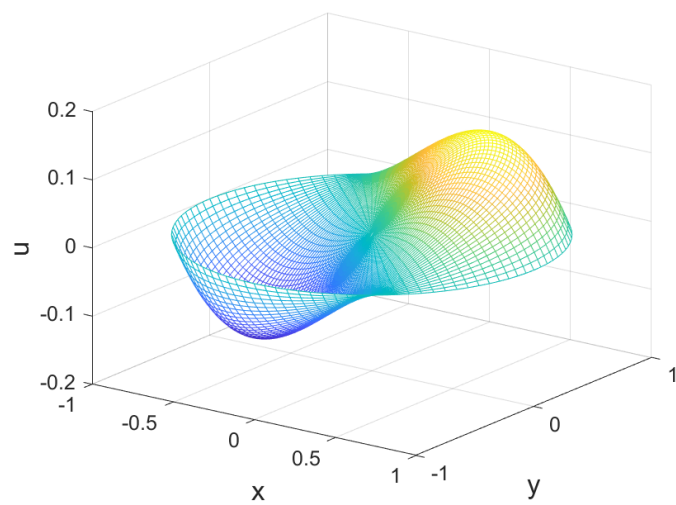


图 24: 极坐标方程的差分格式

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

$$\begin{cases} -\frac{d}{dx}\left(p\frac{du}{dx}\right) + qu(x) = f(x), & x \in I = (a, b), \\ u(a) = 0, \quad u'(b) = 0. \end{cases} \quad (3.1)$$

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

定义函数空间

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

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

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

其中

$$a(u, v) = \int_a^b \left(p \frac{du}{dx} \frac{dv}{dx} + quv\right) dx, \quad F(v) = \int_a^b f v dx.$$

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 , $0 \leq i \leq N$,

$$a = x_0 \quad x_1 \quad x_2 \quad \cdots \quad x_{i-1} \quad x_i \quad x_{i+1} \quad \cdots \quad x_N = b$$

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

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 ($i = 1, 2, \dots, N$).

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 $a = x_0, x_1, \dots, x_{i-1}, x_i, \dots, x_{n-1}, x_n = b$ 上分别取值为

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

则 u_h 可表示成一次 Lagrange 插值函数的形式

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

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.

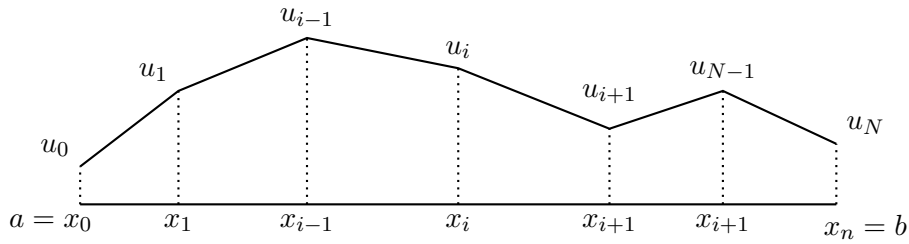


图 25: V_h 中函数的几何形状

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,

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

(ii) 当 $0 < i < N$ 时,

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

(iii) 当 $i = N$ 时,

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

不难验证

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

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

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

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

$$a(u_h, v_h) = F(v_h), \quad \forall v \in V_h^0. \quad (3.4)$$

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

$$u_h = \sum_{i=0}^N u_i \varphi_i, \quad (3.5)$$

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

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

即

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

线性方程组可以写成如下矩阵形式

$$K\mathbf{u} = \mathbf{b}. \quad (3.8)$$

其中

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

$$\mathbf{u} = (u_1, u_2, \dots, u_N)^T, \quad \mathbf{F} = (F(\varphi_1), F(\varphi_2), \dots, F(\varphi_N))^T.$$

4. 有限元方程的求解

为了计算矩阵每个元素

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

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

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

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

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

于是基函数 $\varphi_i(x)$ 可写为

$$\begin{aligned} \varphi_0(x) &= \begin{cases} \frac{x_1 - x}{h_1} = N_{-1}(\xi(x)), & x \in I_1 = [x_0, x_1], \\ 0, & \text{其他}, \end{cases} \\ \varphi_i(x) &= \begin{cases} \frac{x - x_{i-1}}{h_i} = N_1(\xi(x)), & x \in I_i = [x_{i-1}, x_i], \\ \frac{x_{i+1} - x}{h_{i+1}} = N_{-1}(\xi(x)), & x \in I_{i+1} = [x_i, x_{i+1}], \\ 0, & \text{其他}, \end{cases} \quad 0 < i < N, \\ \varphi_N(x) &= \begin{cases} \frac{x - x_{N-1}}{h_N} = N_1(\xi(x)), & x \in I_N = [x_{N-1}, x_N], \\ 0, & \text{其他}. \end{cases} \end{aligned}$$

线性插值公式 (3.3) 可写为

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

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,

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

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

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

这里 $j = 2, 3, \dots, N-1$ 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 $a(\varphi_N, \varphi_N)$ 和

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

另一方面

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

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

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)。

$$\begin{aligned}
\Phi_1^{I_i}(x) &= \begin{cases} \frac{x_i - x}{h_i} = N_{-1}(\xi), & x \in I_i = [x_{i-1}, x_i], \xi \in [-1, 1], \\ 0, & \text{其他.} \end{cases} \\
\Phi_2^{I_i}(x) &= \begin{cases} \frac{x - x_{i-1}}{h_i} = N_1(\xi), & x \in I_i = [x_{i-1}, x_i], \xi \in [-1, 1], \\ 0, & \text{其他.} \end{cases}
\end{aligned}$$

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

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

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

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

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

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

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

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 $\varphi_1, \varphi_2, \dots, \varphi_N$. 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：

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

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

```
1 % FEM1DP.m
2 % finite element method for 1D elliptic problem
3 % -u_xx+u=f in (0,1)
4 % boundary condition: u(0)=u(1)=0;
5 % exact solution: u=x*(1-x)*sin(x);
6 % right hand function: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x)
7 clear all; close all;
8 Num=[16 32 64 128 256 512]; % Number of partitions
9 node_Err=[]; L2_Err=[]; H1_Err=[]; DOF=[];
10 for k=1:length(Num)
11     N=Num(k); h=1/N; x=0:h:1;
12
13     M=[1:N;2:N+1]; % information matrix
14     % the global node indices of the mesh nodes of all the mesh elements
15     [xv,wv]=jags(3,0,0); % nodes and weights of Gauss quadrature
16
17     K=zeros(N+1); % global stiffness matrix
18     F=zeros(N+1,1); % RHS load vector
19
20     for i=1:N % loop for each element
21         K(M(1,i),M(1,i))=K(M(1,i),M(1,i))+((h/2)*(((1/4)*(2/h)^2+((1-xv)/2)
22             .^2)))'*wv;
23         K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2+((1-xv)/2)
24             .*((1+xv)/2)))'*wv;
25         K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2+((1-xv)/2)
26             .*((1+xv)/2)))'*wv;
27         K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2+((1+xv)/2)
28             .^2)))'*wv;
29
30         t=h*xv/2+(x(i+1)+x(i))/2;
31         F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)+(2+2*t-2*t.^2)
32             .*sin(t)))'*wv;
33         F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)+(2+2*t-2*t.^2)
34             .*sin(t)))'*wv;
35     end
36
37     % Handling Dirichlet boundary condition
38     K(1,:)=zeros(1,N+1);
39     K(:,1)=zeros(1,N+1);
40     K(N+1,:)=zeros(1,N+1);
```

```

35     K(:, N+1)=zeros(1,N+1);
36     K(1,1)=1;   K(N+1,N+1)=1;
37     F(1)=0;     F(N+1)=0;
38
39     U=K\F;      % numerical solution at the value of the nodes
40     node_error=max(abs(U'-x.*(1-x).*sin(x))); % node error
41     for i=1:N
42         tt=h*xv/2+(x(i+1)+x(i))/2;
43         % value of finite element solution at Gauss nodes
44         uh=U(i)*(1-xv)/2+U(i+1)*(1+xv)/2;
45         % derivative value of finite element solution at Gauss nodes
46         duh=-U(i)/2+U(i+1)/2;
47         L2_err(i)=h/2*((tt.*(1-tt).*sin(tt)-uh).^2)*wv;
48         % the square of the L2 error of the i-th interval
49         H1_err(i)=h/2*((sin(tt)-2*tt.*sin(tt)+tt.*(1-tt).*cos(tt)-duh*2/h)
50             .^2)*wv;
51         % the square of the H1 semi-norm error of the i-th interval
52     end
53     node_Err=[node_Err, node_error];
54     L2_Err=[L2_Err, sqrt(sum(L2_err))];
55     H1_Err=[H1_Err, sqrt(sum(L2_err)+sum(H1_err))];
56     doff=N+1; % degrees of freedom, number of unknowns
57     DOF=[DOF, doff];
58 end
59 loglog(DOF,node_Err,'r+-','LineWidth',1)
60 hold on
61 loglog(DOF,L2_Err,'bo-','MarkerFaceColor','w','LineWidth',1)
62 hold on
63 loglog(DOF,H1_Err,'m*-','LineWidth',1)
64 hold on
65 grid on
66 legend('L^2 error','L^{\infty} error','H^1 error','location','SouthWest')
67 % title('Convergence of finite element method','fontsize',14)
68 set(gca,'FontName','Times New Roman','FontSize',12)
69 xlabel('Number of degrees of freedom','fontsize', 14),
70 ylabel('Error','fontsize',14)
71
72 % sets axis tick and axis limits
73 xticks(10.^(1:3))
74 yticks(10.^(-8:-1))
75 xlim([10 10^3])
76 ylim([10^(-8) 10^(-1)])
77
78 % calculating of convergence order
79 for k=1:length(Num)-1
80     node_order(k)=log(node_Err(k)/node_Err(k+1))/(log(DOF(k)/DOF(k+1)));
81     L2_order(k)=log(L2_Err(k)/L2_Err(k+1))/(log(DOF(k)/DOF(k+1)));
82     H1_order(k)=log(H1_Err(k)/H1_Err(k+1))/(log(DOF(k)/DOF(k+1)));
83 end

```

```

83 node_order
84 L2_order
85 H1_order
86
87 % print -dpng -r600 FEM1DP.png
88 % print -depsc2 FEM1DP.eps

```

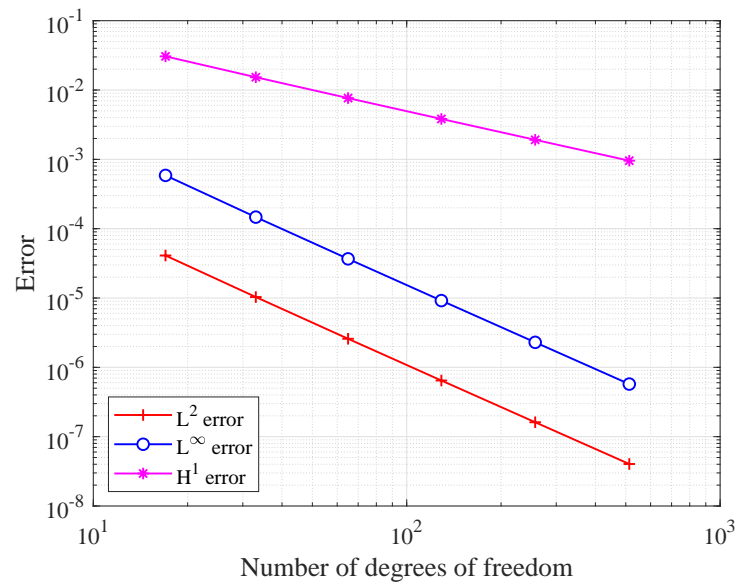


图 26: 一维线性元的收敛速度

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.

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

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 $L_{\omega}^2(-1, 1)$ 的内积. 设

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

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

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

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

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

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

而且 u_N 要精确满足边界条件:

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

逼近解展开为以下形式

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

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

$$\begin{aligned} \sum_{j=0}^N [\mathbf{L}h_j(x_k)] u_N(x_j) &= f(x_k), \quad 1 \leq k \leq N-1, \\ \sum_{j=0}^N [B_- h_j(x_0)] u_N(x_j) &= g_-, \quad \sum_{j=0}^N [B_+ h_j(x_N)] u_N(x_j) = g_+. \end{aligned}$$

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 $u_N(x_0) = g_-, u_N(x_N) = g_+$, 可以推出

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

对 $u_N(x)$ 进行 m 次微分

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

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

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

因此

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

令 \mathbf{f} 表示方程 (4.3) 右端 $N-1$ 个元素的向量. 设

$$\tilde{D}_m = (d_{kj}^{(m)})_{k,j=1,\dots,N-1}, \quad \text{其中 } m = 1, 2,$$

$$P = \text{diag}(p(x_1), \dots, p(x_{N-1})), \quad Q = \text{diag}(q(x_1), \dots, q(x_{N-1})).$$

方程 (4.3) 可以简化为

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

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:

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

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 [Book¹, 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

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

可得以下方程组

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

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

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

¹J. Shen, T. Tang, and L.L. Wang. Spectral Methods: Algorithms, Analysis and Applications. Springer, 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

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

如果能让生成的刚度矩阵 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.

```

1 % LegenSM.m
2 % Legendre spectral-Galerkin method for the model equation
3 % -u_xx+u=f in (-1,1) with boundary condition: u(-1)=u(1)=0;
4 % exact solution: u=sin(kw*pi*x);
5 % RHS: f=kw*kw*pi^2*sin(kw*pi*x)+sin(kw*pi*x);
6 % Rmk: Use routines lepoly(); legs(); lepolym();
7 clear all; close all;
8 kw=1;
9 Nvec=[4:2:28];
10 L2_Err=[]; Max_Err=[]; % initialization error
11 % Loop for various modes N to calculate numerical errors
12 for N=Nvec
13     [xv,wv]=legs(N+1); % Legendre-Gauss points and weights
14     Lm=lepolym(N,xv); % matrix of Legendre polynomials
15     u=sin(kw*pi*xv); % exact function
16     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv); % Right-hand-side(RHS)
17     % Calculating coefficient matrix
18     S=diag(4*(0:N-2)+6); % stiffness matrix
19     M=diag(2./(2*(0:N-2)+1)+2./(2*(0:N-2)+5))...
20         -diag(2./(2*(0:N-4)+5),2)...
21         -diag(2./(2*(2:N-2)+1),-2); % mass matrix
22     A=S+M;
23     % Solving the linear system
24     Pm=Lm(1:end-2,:)-Lm(3:end,:); % matrix of Phi(x)
25     b=Pm*diag(wv)*f; % solving RHS
26     uh=A\b; % expansion coefficients of u_N(x)
27     un=Pm'*uh; % compositing the numerical solution
28
29     L2_err=sqrt(((un-u).^2)*wv); % L^2 error
30     Max_err=max(abs(un-u)); % maximum pointwise error
31     L2_Err=[L2_Err;L2_err];
32     Max_Err=[Max_Err;Max_err];
33 end

```



```

34 % Plot the L^2 and maximum pointwise error
35 plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
36 hold on
37 plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
38 grid on,
39 legend('L^2 error','L^{\infty} error')
40 % title('Convergence of Legendre-Galerkin method','fontsize',12)
41 set(gca,'fontsize',12)
42 xlabel('N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
43
44 % print -dpng -r600 LegenSM.png
45 % print -depsc2 LegenSM.eps

```

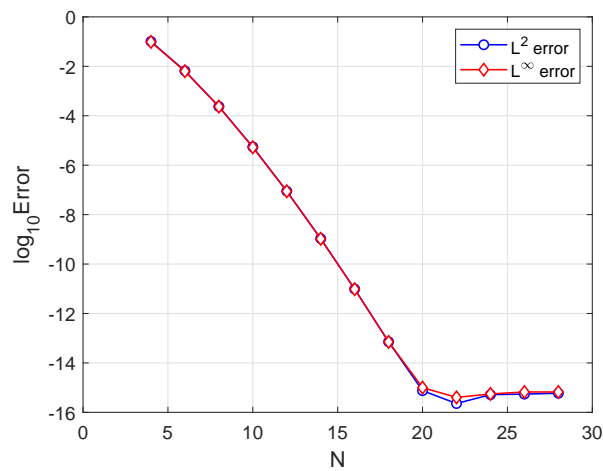


图 27: Legendre 谱方法

```

1 % LegenSM1.m
2 % Legendre spectral-Galerkin method for the model equation
3 % -u_{xx}+u=f in (-1,1) with boundary condition: u(-1)=u(1)=0;
4 % exact solution: u=sin(kw*pi*x);
5 % RHS: f=kw*kw*pi^2*sin(kw*pi*x)+sin(kw*pi*x);
6 % Rmk: Use routines lepoly(); legs(); lepolym();
7 clear all; close all;
8 kw=10;
9 Nvec=[32:2:76];
10 % Initialization for error
11 L2_Err=[]; Max_Err=[];
12 % Loop for various modes N to calculate numerical errors
13 for N=Nvec
14     [xv,wv]=legs(N+1); % Legendre-Gauss points and weights
15     Lm=lepolym(N,xv); % matrix of Legendre polynomials
16     u=sin(kw*pi*xv); % exact solution
17     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv); % RHS
18     % Calculating coefficient matrix
19     S=eye(N-1); % stiff matrix
20     M=diag(1./(4*(0:N-2)+6))*diag(2./(2*(0:N-2)+1)+2./(2*(0:N-2)+5))...
21         -diag(2./(sqrt(4*(0:N-4)+6).*sqrt(4*(0:N-4)+14)).*(2*(0:N-4)+5)),2)

```

```

...
22     -diag(2./(sqrt(4*(2:N-2)-2).*sqrt(4*(2:N-2)+6).*(2*(2:N-2)+1)),-2);
        % mass matrix
23     A=S+M;
24     % Solving the linear system
25     Pm=diag(1./sqrt(4*(0:N-2)+6)).*(Lm(1:end-2,:)-Lm(3:end,:)); % matrix of
        Phi(x)
26     b=Pm*diag(wv)*f;           % solving RHS
27     uh=A\b;                     % expansion coefficients of u_N(x)
28     un=Pm'*uh;                 % compositing the numerical solution
29
30     L2_error=sqrt(((un-u).^2)*wv); % L^2 error
31     Max_error=norm(abs(un-u),inf); % maximum pointwise error
32     L2_Err=[L2_Err;L2_error];
33     Max_Err=[Max_Err;Max_error];
34 end
35 % Plot L^2 and maximum pointwise error
36 plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
37 hold on
38 plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
39 grid on
40 legend('L^2 error','L^{\infty} error','location','NorthEast')
41 set(gca,'fontsize',12)
42 xlabel('N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
43
44 % sets axis tick and axis limits
45 xticks(30:10:80)
46 yticks(-15:3:0)
47 xlim([30 80])
48 ylim([-15 0])
49
50 % print -dpng -r600 LegenSM1.png
51 % print -depsc2 LegenSM1.eps

```

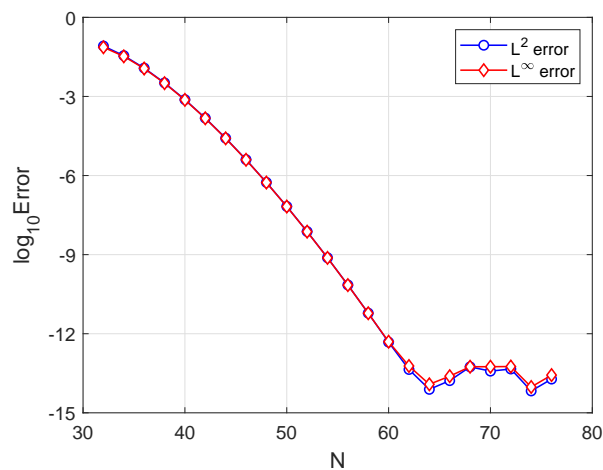


图 28: Legendre 谱方法

4.3 Legendre 谱方法及其算例二

考虑两点边值问题:

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

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 $\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x)$ 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 $X_N = \text{span}\{\phi_k, k = 0, 1, \dots, N-2\}$.

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

$$\begin{aligned} f_k &= \int_I f_N \phi_k dx, \quad \mathbf{f} = (f_0, f_1, \dots, f_{N-2})^T, \\ U_N &= \sum_{j=0}^{N-2} \hat{U}_j \phi_j, \quad \mathbf{U} = (\hat{U}_0, \hat{U}_1, \dots, \hat{U}_{N-2})^T, \\ s_{kj} &= \int_I \phi'_j \phi'_k dx, \quad m_{kj} = \int_I \phi_j \phi_k dx, \\ S &= (s_{kj})_{0 \leq k, j \leq N-2}, \quad M = (m_{kj})_{0 \leq k, j \leq N-2}. \end{aligned}$$

取 $\phi = \phi_k$, 可得到以下方程组

$$(4S + M)U = f.$$

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 [Book, P146-4.23], 它的非零元素为

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

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.

```

1 % LegenSM3.m
2 % Legendre spectral-Galerkin method for 1D elliptic problem
3 % -u_yy+u=f in [0,1] with boundary condition: u(0)=1, u'(1)=0;
4 % exact solution: u=(1-y)^2*exp(y); RHS: f=(2-4*y)*exp(y);
5 % transformed equation: -4U_xx+U=F in [-1,1]
6 % boundary condition: U(-1)=0, U'(1)=0;
7 % exact solution: U=(1/2-1/2*x)^2*exp(1/2*x+1/2)-1;
8 % RHS: F=-2*x*exp(1/2*x+1/2)-1.
9 clear all; close all;
10 Nvec=3:18;
11 % Initialization error and condition number
12 L2_Err=[]; condnv=[];
13 % Loop for various modes N to calculate numerical errors
14 for N=Nvec
15     [xv, wv]=legs(N+1); % Legendre-Gauss points and weights
16     Lm=lepolym(N, xv); % matrix of Legendre polynomials
17     yv=1/2*(xv+1); % variable substitution
18     U=(1-yv).^2.*exp(yv)-1; % exact solution
19     F=(2-4*yv).*exp(yv)-1; % RHS in [0,1]
20
21 % Calculating coefficient matrix
22 e1=0:N-2; e2=0:N-3; e3=0:N-4;
23 S=diag( (4*e1+6).*(e1+1).^2./(e1+2).^2 ); % stiff matrix
24 M=diag( 2./(2*e1+1)+2*(2*e1+3)./(e1+2).^4+2*((e1+1)./(e1+2)).^4./(2*e1+5)
25     )...
26     +diag( 2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2*(e2+3).^2), 1 )...
27     +diag( 2./(e2+2).^2-2*(e2+1).^2./((e2+2).^2*(e2+3).^2), -1 )...
28     +diag( -2*(e3+1).^2./((2*e3+5).*(e3+2).^2), 2 )...
29     +diag( -2*(e3+1).^2./((2*e3+5).*(e3+2).^2), -2); % mass matrix

```

```

29     A=4*S+M;
30
31     % Solving the linear system
32     Pm=(Lm(1:end-2,:)+diag((2*e1+3)./(e1+2).^2)...
33         *Lm(2:end-1,:)-diag((e1+1).^2./(e1+2).^2)*Lm(3:end,:));
34     b=Pm*diag(wv)*F;           % solving RHS
35     Uh=A\b;                    % expansion coefficients of u_N(x)
36     Un=Pm'*Uh;                % compositing the numerical solution
37
38     L2_error=sqrt(((Un-U).^2)*wv); % L^2 error
39     L2_Err=[L2_Err;L2_error];
40     condnv=[condnv,cond(A)];    % condition number
41 end
42 % Plot L^2 error
43 plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
44 grid on
45 %title('L^2 error of Legendre-Galerkin method','fontsize',12)
46 set(gca,'fontsize',12)
47 xlabel('N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
48
49 % sets axis tick and axis limits
50 xticks(2:2:18)
51 yticks(-16:2:0)
52 xlim([2 18])
53 ylim([-16 0])
54
55 % print -dpng -r600 LegenSM3.png
56 % 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:

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

Chebyshev-Galerkin 方法

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

令 $\phi_k(x) = T_k(x) + a_k T_{k+1}(x) + b_k T_{k+2}(x)$ 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

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

取 $v_N = \phi_k$, 可得到以下方程组

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

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

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

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:

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

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: $f = k^2\pi^2 \sin(k\pi x) + \sin(k\pi x)$.

```

1 % ChebSM1.m
2 % Chebyshev spectral-Galerkin method for the model equation
3 % -u_xx+u=f in (-1,1)
4 % boundary condition: u(-1)=u(1)=0;
5 % exact solution: u=sin(kw*pi*x);
6 % RHS: f=kw*kw*pi^2*sin(kw*pi*x)+sin(kw*pi*x);
7 % Rmk: Use routines japoly(); jags(); japolym();
8 clear all; close all;
9 kw=10;
10 Nvec=32:2:76;
11 % Initialization for error
12 L2_Err=[]; Max_Err=[];
13 % Loop for various modes N to calculate numerical errors
14 for N=Nvec
15     [xv,wv]=jags(N+1,-1/2,-1/2); % Chebyshev-Gauss points and weights
16     Cm=japolym(N,-1/2,-1/2,xv)./japolym(N,-1/2,-1/2,1); % matrix of
        Chebyshev polynomials
17     u=sin(kw*pi*xv); % exact solution
18     f=kw*kw*pi^2*sin(kw*pi*xv)+sin(kw*pi*xv); % RHS
19     % Calculating coefficient matrix
20     S=zeros(N-1); % stiff matrix
21     for k=1:N-1
22         for j=1:N-1
23             if k==j
24                 S(k,j)=2*pi*k*(k+1);
25             elseif (k<j && mod(j-k,2)==0)
26                 S(k,j)=4*pi*k;
27             else
28                 S(k,j)=0;
29             end
30         end
31     end
32
33     M=diag([3/2*pi, pi*ones(1,N-2)])+diag(-pi/2*ones(1,N-3),2)...
34         +diag(-pi/2*ones(1,N-3),-2); % mass matrix
35     A=S+M;
36
37     % Solving the linear system

```

```

38     Pm=Cm(1:end-2,:)-Cm(3:end,:); % matrix of Phi(x)
39     b=Pm*diag(wv)*f; % Solving RHS
40     uh=A\b; % expansion coefficients of u_N(x)
41     un=Pm'*uh; % compositing the numerical solution
42
43     L2_err=sqrt(((un-u).^2)*wv); % L^2 error
44     Max_err=norm(abs(un-u),inf); % maximum pointwise error
45     L2_Err=[L2_Err;L2_err];
46     Max_Err=[Max_Err;Max_err];
47 end
48 % plot L^2 and maximum pointwise error
49 plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
50 hold on
51 plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
52 grid on
53 legend('L^2 error','L^{\infty} error','location','NorthEast')
54 % title('Convergence of Chebyshev-Galerkin method','fontsize',12)
55 set(gca,'fontsize',12)
56 xlabel('N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
57
58 % sets axis tick and axis limits
59 xticks(30:10:80)
60 yticks(-14:2:0)
61 xlim([30 80])
62 ylim([-14 0])
63
64 % print -dpng -r600 ChebSM1.png
65 % print -depsc2 ChebSM1.eps

```

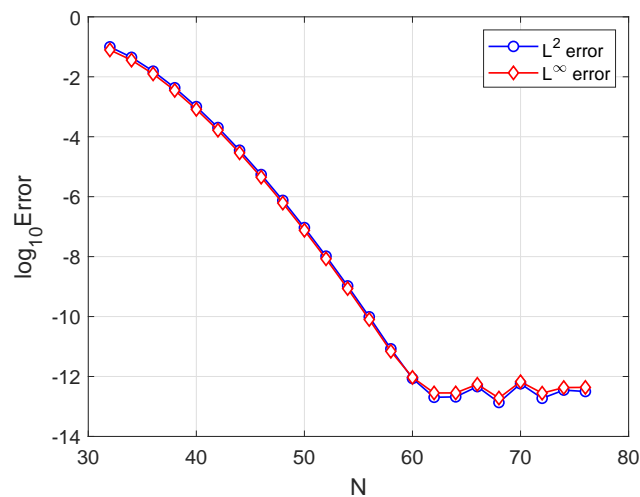


图 30: Chebyshev 谱方法

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:

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

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

$$\begin{cases} \text{Find } u \in P_N \text{ such that} \\ -\mu u_N''(x_i) + \nu u_N'(x_i) + \rho u_N(x_i) = f(x_i), & 1 \leq i \leq N-1, \\ u_N(-1) = 0, \quad u_N(1) = 0. \end{cases} \quad (4.8)$$

逼近解展开为以下形式

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

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 $D = (d_{kj} := h_j'(x_k))_{k,j=1,2,\dots,N}$. 记 $w_j = u_N(x_j)$, 将 $u_N(x)$ 展开为 $u_N(x) = \sum_{j=0}^N w_j h_j(x)$, 可知

$$u_N(x_k) = \sum_{j=0}^N w_j h_j(x_k) = w_k,$$

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

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

将以上公式代入方程 (4.8) 可得

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

记

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

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 $k = 10$)

```

1 % LegenCM2.m
2 % Legendre-collocation method for the model equation:
3 % -mu*u''(x)+nu*u'(x)+rho*u(x)=f(x), x in (-1,1),
4 % boundary condition: u(-1)=u(1)=0;
5 % exact solution: u=sin(kw*pi*x);
6 % RHS: f(x)=mu*kw^2*pi^2*sin(kw*pi*x)+nu*kw*pi*cos(kw*pi*x)+rho*sin(kw*pi*x)
7 % Rmk: Use routines lepoly(); legslb(); legslbmdm();
8 clear all; close all;
9 kw=10;
10 nu=1;
11 mu=1;
12 rho=1;
13 Nvec=32:2:76;
14 % Initialization for error
15 L2_Err=[]; Max_Err=[];
16 % Loop for various modes N to calculate numerical errors
17 for N=Nvec
18     [xv,wv]=legslb(N); % Legendre-Gauss-Lobatto points and weights
19     u=sin(kw*pi*xv); % exact solution
20     f=mu*kw*kw*pi^2*sin(kw*pi*xv)+nu*kw*pi*cos(kw*pi*xv)+rho*sin(kw*pi*xv);
21     % RHS
22     % Solve the collocation system
23     D1=legslbdiff(N,xv); % 1st order differentiation matrix
24     %D1=legslbmdm(N); % 1st order differentiation matrix
25     D2=D1*D1; % 2nd order differentiation matrix
26     % Compositing the coefficient matrix
27     D=-mu*D2(2:N-1,2:N-1)+nu*D1(2:N-1,2:N-1)+rho*eye(N-2);
28     b=f(2:N-1); % RHS
29     un=D\b;
30     un=[0;un;0];
31
32     L2_error=sqrt(((un-u).^2)*wv); % L^2 error
33     Max_error=norm(abs(un-u),inf); % maximum pointwise error
34     L2_Err=[L2_Err;L2_error];
35     Max_Err=[Max_Err;Max_error];

```

```

35 end
36 % Plot L^2 and maximum pointwise error
37 plot(Nvec,log10(L2_Err),'bo-','MarkerFaceColor','w','LineWidth',1)
38 hold on
39 plot(Nvec,log10(Max_Err),'rd-','MarkerFaceColor','w','LineWidth',1)
40 grid on
41 legend('L^2 error','L^{\infty} error','location','NorthEast')
42 set(gca,'fontsize',12)
43 xlabel('N','fontsize',14), ylabel('log_{10}Error','fontsize',14)
44 % title('Convergence of Legendre-collocation method','fontsize',12)
45
46 % sets axis tick and axis limits
47 xticks(30:10:80)
48 yticks(-15:3:0)
49 xlim([30 80])
50 ylim([-15 0])
51
52 % print -dpng -r600 LegenCM2.png
53 % print -depsc2 LegenCM2.eps

```

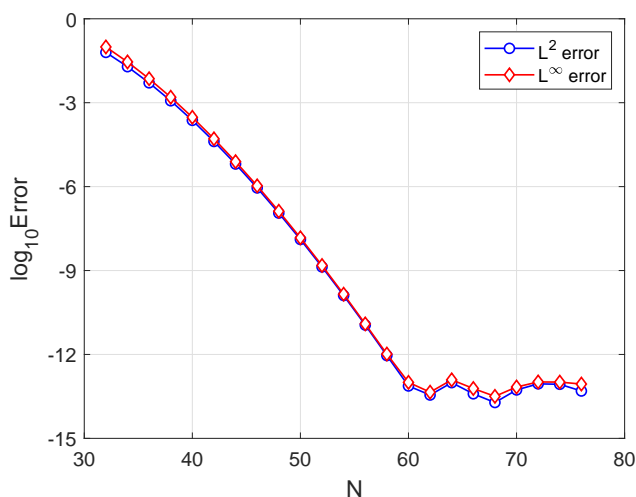


图 31: Legendre 配置方法

注 (1) 程序 LegenCM1.m 对应配置方法求解两点边值问题

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

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).