第十一章：有限差分方法

1. 差商公式

8.2.1 差商公式

将微分方程中的微商用差商代替(离散化)，是差分法求解偏微分方程的基础。设 u 是坐标 x 的函数，取 $\Delta x=h$ 等分坐标轴，结点坐标为 x_i，相应的有 $u_i(i=1,2,\cdots)$，则有泰勒展开

$$u_{i+1}=u_i+hu_i'+\frac{h^2}{2!}u_i''+\frac{h^3}{3!}u_i'''+\cdots \tag{8.19}$$

$$u_{i-1}=u_i-hu_i'+\frac{h^2}{2!}u_i''-\frac{h^3}{3!}u_i'''+\cdots \tag{8.20}$$

$$u_{i+2}=u_i+2hu_i'+\frac{(2h)^2}{2!}u_i''+\frac{(2h)^3}{3!}u_i'''+\cdots \tag{8.21}$$

$$u_{i-2}=u_i-2hu_i'+\frac{(2h)^2}{2!}u_i''-\frac{(2h)^3}{3!}u_i'''+\cdots \tag{8.22}$$

1. 误差为 O(h)的差商公式

由式(8.19)可得一阶向前差商公式

$$\frac{\mathrm{d}u_i}{\mathrm{d}x}=\frac{u_{i+1}-u_i}{h}+\mathrm{O}(h) \tag{8.23}$$

由式(8.20)可得一阶向后差商公式

$$\frac{\mathrm{d}u_i}{\mathrm{d}x}=\frac{u_i-u_{i-1}}{h}+\mathrm{O}(h) \tag{8.24}$$

式(8.21)−2×式(8.19)，可得二阶向前差商公式

$$\frac{\mathrm{d}^2u_i}{\mathrm{d}x^2}=\frac{u_{i+2}-2u_{i+1}+u_i}{h^2}+\mathrm{O}(h) \tag{8.25}$$

式(8.22)−2×式(8.20)，可得二阶向后差商公式

$$\frac{\mathrm{d}^2u_i}{\mathrm{d}x^2}=\frac{u_i-2u_{i-1}+u_{i-2}}{h^2}+\mathrm{O}(h) \tag{8.26}$$

2. 误差为 $\mathrm{O}(h^2)$ 的差商公式

将式(8.25)代入式(8.19)，可得一阶向前差商公式

$$\frac{\mathrm{d}u_i}{\mathrm{d}x}=\frac{-u_{i+2}+4u_{i+1}-3u_i}{2h}+\mathrm{O}(h^2) \tag{8.27}$$

将式(8.26)代入式(8.20)，可得一阶向后差商公式

$$\frac{\mathrm{d}u_i}{\mathrm{d}x}=\frac{u_{i-2}-4u_{i-1}+3u_i}{2h}+\mathrm{O}(h^2) \tag{8.28}$$

[式(8.19)−式(8.20)]/(2h)，可得一阶中心差商公式

$$\frac{\mathrm{d}u_i}{\mathrm{d}x}=\frac{u_{i+1}-u_{i-1}}{2h}+\mathrm{O}(h^2) \tag{8.29}$$

[式(8.19)+式(8.20)]/h^2，可得二阶中心差商公式

$$\frac{\mathrm{d}^2u_i}{\mathrm{d}x^2}=\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\mathrm{O}(h^2) \tag{8.30}$$

泊松方程的差分方程组为

$$\nabla^2 u_{ij} = \frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}) = f_{ij} \tag{8.34}$$

该方程组中方程个数等于网格结点数，并且每个方程的左边最多有五项。故该方程组的系数矩阵具有大型、稀疏和带状的特点。为节省内存和机时，一般不采用消元法，而是采用迭代法，包括如下三种方法。

(1) 同步法。

将式(8.34)化为

$$u_{ij}^{k+1} = \frac{1}{4}(u_{i,j-1} + u_{i-1,j} + u_{i,j+1} + u_{i+1,j} - h^2 f_{ij})^k \tag{8.35}$$

上式中右边的 u 用第 k 步的值，则左边得到第 $k+1$ 步的值。这种迭代方法称为同步法，它需要两套内存，一套存第 k 步的值，一套存第 $k+1$ 步的值，收敛较慢。

(2) 异步法。

由于 u 的计算是按照 i、j 由小到大进行的，迭代到某一步在计算新的 u_{ij} 时，新的 $u_{i,j-1}$ 和 $u_{i-1,j}$ 已被算出，所以式(8.35)可以改写为

$$u_{ij}^{k+1} = \frac{1}{4}(u_{i,j-1}^{k+1} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k} + u_{i+1,j}^{k} - h^2 f_{ij}) \tag{8.36}$$

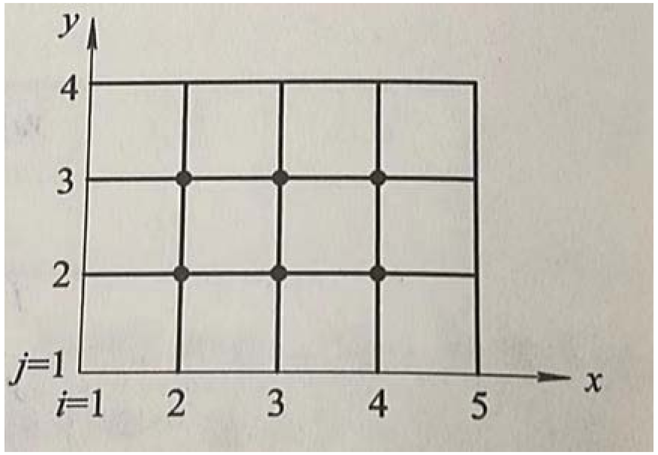
这种迭代方法称为异步法，它只需一套内存，收敛较快。

2. 试着采用有限差分方法求解拉普拉斯方程在网格结点处的值

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0 \quad (0<x<4, 0<y<3)$$

$$u\big|_{x=0}=y(y-3),\ u\big|_{x=4}=0$$

$$u\big|_{y=0}=\sin\left(\frac{\pi}{4}\right)x,\ u\big|_{y=3}=0$$



取 h=1，ω=1.25，迭代终止误差为ε=10e-4

程序如下：

```
program main
    implicit none
    integer i,j
    real x(5),y(4),u(5,4),t,p,q
```

```
real,parameter:: w=1.25,eps=1e-4,h=1,pi=3.1415926

open(1,file='examples.dat')

do i=1,5

     x(i)=(i-1)*h

     u(i,1)=sin(pi*x(i)/4)

     u(i,4)=0.0


end do


do j=1,4

    y(j)=(j-1)*h

    u(1,j)=y(j)*(y(j)-3)

    u(5,j)=0.0


end do
```

```
      do i=2,4
          do j=2,3
              u(i,j)=0.0
          end do
      end do
10    p=0.0
       do i=2,4
          do j=2,3
              t=u(i,j)
              u(i,j)=w*(u(i,j-1)+u(i-1,j)+u(i,j+1)+u(i+1,j))/4+(1-w)*u(i,j)
              q=abs(u(i,j)-t)
              if(q.gt.p) p=q
          end do
       end do
```

```
      if(p.gt.eps) goto 10
      do i=1,5
         do j=1,4
            write(*,*) i,j,u(i,j)
            write(1,20) i,j,u(i,j)
         end do
      end do
20    format(1x,2I5,f15.6
end program
```

计算结果如下：

I	j	u(I,j)
1	1	-0.000000
1	2	-2.000000
1	3	-2.000000
1	4	0.000000
2	1	0.707107
2	2	-0.440337
2	3	-0.637549
2	4	0.000000
3	1	1.000000
3	2	0.169035
3	3	-0.109850
3	4	0.000000
4	1	0.707107
4	2	0.226313
4	3	0.029117
4	4	0.000000
5	1	0.000000
5	2	0.000000
5	3	0.000000
5	4	0.000000

第 十二章 多项式和一般函数的计算

一维多项式多组求值

六、例

（1）利用系数预处理法计算多项式

$$P(x) = 2x^6 + 5x^5 + 3x^4 + x^3 - 7x^2 + 7x - 20$$

在 $x = \pm 0.9, \pm 1.1, \pm 1.3$ 处的函数值。

主程序(文件名:OPLYS0.FOR)为

一、功能

利用系数预处理法对多项式

$$P(x)=a_n x^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2 x+a_1$$

进行多组求值。其中 $n=2^k(k\geqslant 1)$

二、方法说明

该多项式为

$$P(x)=a_n x^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2 x+a_1$$

其中 $n=2^k(k\geqslant 1)$ 。

首先将多项式变为首一多项式，即令

$$\begin{aligned}T(x)&=P(x)/a_n\\&=x^{n-1}+t_{n-1}x^{n-2}+\cdots+t_2 x+t_1\end{aligned}$$

其中

$$t_k=a_k/a_n,\quad k=n-1,\cdots,2,1$$

然后对首一多项式 $T(x)$ 中的系数进行预处理。其预处理过程如下。

首先，将 $T(x)$ 分解为如下形式：

$$T(x)=(x^j+b)q(x)+r(x)$$

其中 $j=2^{k-1}$，$q(x)$ 与 $r(x)$ 均为 $2^{k-1}-1$ 次的首一多项式。b，$q(x)$ 与 $r(x)$ 按以下规则确定：

多项式中的中项系数减 1 为 b，即

$$b = t_{2^{k-1}} - 1$$

多项式左半部分除以 x^j 为 $q(x)$，即

$$q(x) = x^{s-1} + q_{s-1}x^{s-2} + \cdots + q_2 x + q_1$$

其中

$$s = 2^{k-1}$$

$$q_i = t_{i+s},\ i = s-1, \cdots, 2, 1$$

多项式右半部分减去 b 与 $q(x)$ 中各系数的乘积，即

$$r(x) = x^{s-1} + r_{s-1}x^{s-2} + \cdots + r_2 x + r_1$$

其中

$$r_i = t_i - bq_i,\ i = s-1, \cdots, 2, 1$$

由于 $q(x)$ 与 $r(x)$ 还是首一多项式，也可以按照上述方法分别进行分解。这个过程一直作到 $q(x)$ 与 $r(x)$ 的次数为 1 为止。并且，每次分解后的系数仍放在 $T(x)$ 的各系数的存储单元中。最后就可以得到 $T(x)$ 经分解处理后的系数。

首一多项式 $T(x)$ 的系数经预处理后，就可以用这些预处理后的系数对不同的 x 求函数值。

这种方法特别适用于对多个 x 进行求值，减少求值中的乘法次数。

如果原来的多项式不满足 $n = 2^k$ 这个条件时，可以添加系数为 0 的各项及系数为 1 的最高次项。

三、子程序语句

SUBROUTINE OPLYS(A,N,B,M,X,L,P)

四、形参说明

A——双精度实型一维数组，长度为N，输入参数。存放N−1次多项式的系数：

$$P(x) = a_N x^{N-1} + a_{N-1} x^{N-2} + \cdots + a_2 x + a_1$$

返回时该数组将被破坏。

N——整型变量，输入参数。多项式 $P(x)$ 的项数，即该多项式的最高次为N−1。

B——双精度实型一维数组，长度为M=2*N，工作数组。

M——整型变量，输入参数。工作数组B的长度，M=2*N。

X——双精度实型一维数组，长度为L，输入参数。存放L组自变量值。

L——整型变量，输入参数。给定自变量的个数。

P——双精度实型一维数组，长度为L，输出参数。存放给定自变量值的多项式值，即

$$\mathrm{P}(k) = \mathrm{P}(\mathrm{X}(k)),\ k = 1,2,\cdots,\mathrm{L}$$

```
PRAGRAM EXAMPLESS12
DOUBLE PRECISION A(7),X(6),B(14),P(6)
      DATA X/0.9,-0.9,1.1,-1.1,1.3,-1.3/
      DATA A/-20.0,7.0,-7.0,1.0,3.0,-5.0,2.0/
      CALL OPLYS(A,7,B,14,X,6,P)
      DO 20 I=1,6
20    WRITE(*,100) I,X(I),I,P(I)
100      FORMAT(1X,'X(',I2,')=',F5.2,5X,'P(',I2,')=',D13.6)
      END
```

```
PRAGRAM EXAMPLESS13
DOUBLE PRECISION A(16),X(6),B(32),P(6)
    DATA X/0.9,-0.9,1.1,-1.1,1.3,-1.3/
    DATA A/1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,    &
                  11.0,12.0,13.0,14.0,15.0,16.0/
    CALL OPLYS(A,16,B,32,X,6,P)
    DO 20 I=1,6
20  WRITE(*,100) I,X(I),I,P(I)
100     FORMAT(1X,'X(',I2,')=',F5.2,5X,'P(',I2,')=',D13.6)
    END
```

运行结果为

```
X( 1)=    .90      P( 1)= -.185623D+02
X( 2)= -.90       P( 2)= -.267154D+02
X( 3)=   1.10     P( 3)= -.195561D+02
X( 4)=-1.10       P( 4)= -.215130D+02
X( 5)=   1.30     P( 5)= -.208757D+02
X( 6)=-1.30       P( 6)= -.634044D+01
```

(2) 利用系数预处理法计算多项式

$$P(x) = \sum_{k=1}^{16} k x^{k-1}$$

在 $x = \pm 0.9, \pm 1.1, \pm 1.3$ 处的函数值。

运行结果为

```
X( 1)=    .90      P( 1)=   .518215D+02
x( 2)=  -.90       P( 2)= -.133476D+01
X( 3)=   1.10      P( 3)=   .375698D+03
X( 4)=-1.10        P( 4)= -.358245D+02
X( 5)=   1.30      P( 5)=   .282065D+04
X( 6)=-1.30        P( 6)= -.475288D+03
```

```
SUBROUTINE OPLYS(A,N,B,M,X,L,P)
      DOUBLE PRECISION A(N),B(M),X(L),P(L)
      DOUBLE PRECISION Y,Z
      INTEGER T,S
      Y=A(N)
      DO 10 I=1,N
10    B(I)=A(I)/Y
      K=LOG(N-0.5)/LOG(2.0)+1
      NN=1
      DO 20 I=1,K
20    NN=2*NN
      DO 40 I=N+1,NN-1
40    B(I)=0.0
      B(NN)=1.0
      T=NN
```

```
      S=1
      DO 70 I=1,K-1
         T=T/2
         MM=-T
         DO 60 J=1,S
            MM=MM+T+T
            B(MM)=B(MM)-1.0
            DO 50 KK=2,T
50          B(MM-KK+1)=B(MM-KK+1)-B(MM)*B(MM+T-KK+1)
60       CONTINUE
         S=S+S
70    CONTINUE
      DO 200 KK=1,L
         DO 90 I=0,(NN-2)/2
90       A(I+1)=X(KK)+B(2*I+1)
```

```
      MM=1
      Z=X(KK)
      DO 110 I=1,K-1
        MM=MM+MM
        LL=MM+MM
        Z=Z*Z
        DO 100 J=0,NN-1,LL
100         A(J/2+1)=A(J/2+1)+A((J+MM)/2+1)*(Z+B(J+MM))
110     CONTINUE
      Z=Z*Z/X(KK)
      IF (NN.NE.N) A(1)=A(1)-Z
      P(KK)=A(1)*Y
200   CONTINUE
    RETURN
    END
```