北京航空航天大學

《数值分析 A》

计算实习

题目三

刘春辉

SY1405119

目录

一．算法设计方案

算法共分为四步，首先是求解u、t，然后进行插值运算，再进行曲面拟合并求解系数矩阵C，最后计算k值及系数矩阵。

1.1 根据 x、y 求解 u、t

将 $x_i^* = 0.08i(i = 0,1,\cdots,10)$ 和 $y_j^* = 0.5 + 0.05j(j = 0,1,\cdots,20)$ 代入非线性方程组

$$\begin{cases} 0.5\cos t + u + v + w - x = 2.67 \\ t + 0.5\sin u + v + w - y = 1.07 \\ 0.5t + u + \cos v + w - x = 3.74 \\ t + 0.5u + v + \sin w - y = 0.79 \end{cases}$$

中，用 Newton 法解出 t_i 和 u_j；

对应函数为 call solvaleq(x,y,u,t,pe,qe)　　!解 11*21 个非线性方程并回代验证

1.2 通过已知的二维数表，进行分片二次插值

对应函数为 fenpianchazhi(u,t,z,pe,qe,n)　!分片二次插值求出 ut 对应的 z

1)找到 1.1 中求解的点 A（t_i，u_j）所在已知数表的单元格 tb(m)< t_i <tb(m+1),ub(n)< u_j <ub(n+1),找到距离 A 最近的单元节点 B 作为九点插值的中心节点，如果 B 位于边界上，则将 B 设置为就近的内部单元节点

2)u 和 t 的九个点确定一个分片区域，即选 $(t_k,u_r)(k = m-1,m,m+1;r = n-1,n,n+1)$ 为插值节点，相应的 Lagrange 形式的

插值多项式为

$$p_{22}(t,u)=\sum_{k=m-1}^{m+1}\sum_{r=n-1}^{n+1}l_k(t)\tilde{l}_r(u)f(t_k,u_r)$$

其中

$$l_k(t)=\prod_{\substack{w=m-1\\ w\neq k}}^{m+1}\frac{t-t_w}{t_k-t_w}\quad (k=m-1,\ m,\ m+1)$$

$$\tilde{l}_r(u)=\prod_{\substack{w=n-1\\ w\neq r}}^{n+1}\frac{y-y_w}{y_r-y_w}\quad (r=n-1,\ n,\ n+1)$$

并将t_i和u_j代入$p_{22}(t,u)$，便得到了数表$x_i,y_j,f(x_i,y_j)$。

1.3 进行曲面拟合和求系数矩阵$\mathbf{C}=[c_{rs}^{*}]$

对应函数为

nihe(c,x,y,z,pe,qe,k)

将拟合节点x_i,y_j代入非线性方程组，解出中间变量u,t。将x_i,y_j对应的中间变量u,t值代入分片插值多项式中，找出x_i,y_j与z_{ij}的对应关系。即$z_{ij}=f\left(x_i,y_j\right)$。选取拟合基函数为$1,x,x^2,x^3\cdots,1,y,y^2,y^3\cdots$，先设定 k=1, 将拟合节点代入拟合基函数，计算出矩阵B,G，其中 B,G的列数根据拟合精度要求来自动确定。

$$\mathbf{C}=(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{U}\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}$$

其中

$$\mathbf{B}=[j_r(x_i)]=\begin{bmatrix}x_0 & \cdots & x_0^k\\ x_1 & \cdots & x_1^k\\ \vdots & \vdots & \vdots\\ x_{10} & \cdots & x_{10}^k\end{bmatrix},G=[y_s(y_j)]=\begin{bmatrix}y_0 & \cdots & y_0^k\\ y_1 & \cdots & y_1^k\\ \vdots & \vdots & \vdots\\ y_{10} & \cdots & y_{10}^k\end{bmatrix}$$

$$\mathbf{U}=[f(x_i,y_j)]$$

因为B^TB为正定矩阵,所以矩阵求逆使用列主元素高斯消去法（初等行变换）将其变换为单位阵来求得。计算BCG^T，BCG^T与U相应元素平方和即为拟和精

度。

1.4 计算 k 值及相应的系数矩阵

totalerror(error,c,x,y,z,k,pe,qe)

solvek(c,x,y,z,pe,qe,k)，

根据另外给出的节点值来观察逼近效果。k 从 1 开始逐渐增大，当$\sigma \le 10^{-7}$，得到满足精度要求的系数c_{rs}和相应的 k。

1.5 计算并输出第二小题的答案（x,y,f(x,y),p(x,y)）

对应函数为 lastquestion(c,k)

二．源程序及注释

```
!!!!!!!!!!!!!!!!!!!!!!!!!
!hw 2.12.9
!以 xy 为输入值，求解非线性方程组 v,w,t,u
! 牛顿法求解非线性方程组
!拉格朗日分片二次插值
!最小二乘法拟合曲面
!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
      program step
    implicit none
!       integer,parameter::k=5
      integer,parameter::n=6
      integer,parameter::pe=11
      integer,parameter::qe=21
    integer,parameter::it=1000   !控制最大迭代步数
    real,parameter::er=1.e-12   !控制最大迭代步数
    real x(pe),y(qe),u(pe,qe),t(pe,qe),z(pe,qe)   !解非线性方程组用到的
变量
    real ub(n),tb(n),zb(n,n)
    real c(20,20)   !拟合系数,拟合多项式的值
    real v,w
    real error                    !残差
    integer i,j,m,l,k    !自由变量
    do i=1,pe
    x(i)=0.08*(i-1)
    end do

    do j=1,qe
    y(j)=0.5+0.05*(j-1)
```

```
      end do

      call solvaleq(x,y,u,t,pe,qe)       !解 11*21 个非线性方程并回代验证
      call output(u,t,z,pe,qe)            !输出 utz 的值

      call fenpianchazhi(u,t,z,pe,qe,n)   !分片二次插值求出 ut 对应的 z
      call outputxyz(x,y,z,pe,qe)

      call solvek(c,x,y,z,pe,qe,k)
!     call lastquestion(c,k)
!     call nihe(c,x,y,z,pe,qe,k)
!     call totalerror(error,c,x,y,z,k,pe,qe)
!     write(*,*)error
      stop
      end

      subroutine lastquestion (c,k)
      integer,parameter::n=6
          integer,parameter::pe=8 !规定 x geshu
          integer,parameter::qe=5 !规定 y geshu
      integer i,j
```

```
      real x(pe),y(qe),z(pe,qe),u(pe,qe),t(pe,qe),p(pe,qe)
      real ub(n),tb(n),zb(n,n)
      real c(20,20)

      open(100,FILE='last xyzp.txt',STATUS='UNKNOWN')
      do i=1,pe
      x(i)=0.1*i
      end do
      do j=1,qe
      y(j)=0.5+0.2*j
      end do
!     write(*,*)(x(i),i=1,pe)
!解非线性方程组
      call solvaleq(x,y,u,t,pe,qe)       !解 8*5 个非线性方程并回代验证
      call output(u,t,z,pe,qe)            !输出 utz 的值

      call fenpianchazhi(u,t,z,pe,qe,n)   !分片二次插值求出 ut 对应的 z
      call outputxyz(x,y,z,pe,qe)
      do i=1,pe                                  !利用之前求出来的 Cij,求 p
的值同 z 对比
      do j=1,qe
```

```
call fpvalue(c,x(i),y(j),k,fp)
p(i,j)=fp
write(100,10)x(i),y(j),z(i,j),p(i,j)
end do
end do
10      FORMAT (F4.1,F4.1,E20.12,E20.12)
close(100)



end

subroutine solvek(c,x,y,z,p,q,k)
integer i,j
integer p,q,k
real x(p),y(q),z(p,q)
    integer,parameter::m=20 !规定最大迭代次数
real c(m,m)
do k=1,19
call nihe(c,x,y,z,p,q,k)
```

```
    call totalerror(error,c,x,y,z,k,p,q)
!测试不同的 k 对拟合精度的影响
!   if (error.lt.(1.e-7)) then
!   write(*,*)k,error
!   exit
!   else
    write(*,*) k,error
!   end if

    end do
    open(6,FILE='k error c.txt',STATUS='UNKNOWN')
    write(6,*)'k=',k
    write(6,*)'error=',error
    write(6,*)'r=          s=          Crs='
    do i=1,k+1
    do j=1,k+1
    write(6,*) i,j,c(i,j)
    end do
    end do

    end
```

```
subroutine totalerror(err,c,x,y,z,k,p,q)
integer p,q
real c(20,20),x(p),y(q),z(p,q),fp
integer i,j
real err
err=0
do i=1,p
do j=1,q
call fpvalue(c,x(i),y(j),k,fp)
err=err+(z(i,j)-fp)**2
end do
end do

end



subroutine fpvalue(c,x,y,k,fp)
real c(20,20),x,y
integer i,j
real s
```

```
s=0
do i=1,k+1
do j=1,k+1
s=s+c(i,j)*x**(i-1)*y**(j-1)
end do
end do
fp=s
end

subroutine nihe(c,x,y,z,p,q,k)
integer p,q,k
integer i,j,l
real c(20,20),x(p),y(q),z(p,q)
real b(p,k+1),b1(k+1,k+1),b2(k+1,q),a(k+1,q)
real g(q,k+1),g1(k+1,k+1),d(q,k+1),e(k+1)
real s

do i=1,p
do j=1,k+1
b(i,j)=x(i)**(j-1)
end do
```

```
    end do
!求 b1
        do i=1,k+1
    do j=1,k+1
    s=0
    do l=1,p
    s=s+b(l,i)*b(l,j)
    end do
    b1(i,j)=s
    end do
    end do
!求 b1 的逆，返回给 b1
        do i=1,k+1
    e(i)=1
    end do
    call gaussj(b1,k+1,e)
!求 b2   B2=Bt*z
        do i=1,k+1
    do j=1,q
    s=0
    do l=1,p
```

```
s=s+b(l,i)*z(l,j)
end do
b2(i,j)=s
end do
end do
!求 A    A=B1*B2
    do i=1,k+1
do j=1,q
s=0
do l=1,k+1
s=s+b1(i,l)*b2(l,j)
end do
a(i,j)=s
end do
end do
!X 方向拟合结束，开始 Y 方向拟合
!求 g
    do i=1,q
do j=1,k+1
g(i,j)=y(i)**(j-1)
end do
```

```
    end do
!求 g1 G1=Gt*G
        do i=1,k+1
    do j=1,k+1
    s=0
    do l=1,q
    s=s+g(l,i)*g(l,j)
    end do
    g1(i,j)=s
    end do
    end do
!求 G1 的逆，返回给 G1
        CALL gaussj(g1,k+1,e)
!求 D   D=G*G1
        do i=1,q
    do j=1,k+1
    s=0
    do l=1,k+1
    s=s+g(i,l)*g1(l,j)
    end do
    d(i,j)=s
```

```
    end do
    end do
!求 c    C=A*D
        DO i=1,k+1
    do j=1,k+1
    s=0
    do l=1,q
    s=s+a(i,l)*d(l,j)
    end do
    c(i,j)=s
    end do
    end do

!check
        do i=1,k+1
    do j=1,k+1
!   write(*,*)i,j,c(i,j)
    end do
    end do

    end
```

```
subroutine outputxyz(x,y,z,p,q)
integer i,j,p,q
real x(p),y(q),z(p,q)
open(4,FILE='xyz.txt',STATUS='UNKNOWN')
write(4,*) 'x                y                z'
do i=1,p
do j=1,q
write(4,*) x(i),y(j),z(i,j)
end do
end do

end

subroutine fenpianchazhi(u,t,z,p,q,n)   !分片二次插值求出 ut 对应的
z
integer p,q
real r(4)
real u(p,q),t(p,q),z(p,q)
real ub(n),tb(n),zb(n,n)
```

```
      integer i,j,k,l,m,s
!     write(*,*) p,q,n

      call readdata(ub,tb,zb,n)          !读入表格中的数据

      do i=1,p                    !p
      do j=1,q                 !q

      do k=1,n-1                          !找到 u 所在区间，记录 k
      if (u(i,j).GE.ub(k).AND.u(i,j).LT.ub(k+1)) then
      exit
      end if
      end do

      do l=1,n-1                          !找到 t 所在区间，记录 l
      if (t(i,j).GE.tb(l).AND.t(i,j).LT.tb(l+1)) then
      exit
      end if
      end do

!求出距离最近的节点作为插值中心节点
```

```
call distance(u(i,j),t(i,j),ub(k),tb(l),r(1))
call distance(u(i,j),t(i,j),ub(k),tb(l+1),r(2))
call distance(u(i,j),t(i,j),ub(k+1),tb(l),r(3))
call distance(u(i,j),t(i,j),ub(k+1),tb(l+1),r(4))
call nearestp(r,s)
if (s.eq.1) then
k=k
l=l
end if
if(s.eq.2) then
k=k
l=l+1
end if
if(s.eq.3) then
k=k+1
l=l
end if
if(s.eq.4) then
k=k+1
l=l+1
end if
```

```
!处理中心节点位于边界的情况，向内部平移一层
        if (k.eq.1) then
         k=k+1
    end if
        if(k.eq.n) then
         k=k-1
    end if

    if (l.eq.1) then
         l=l+1
    end if
        if(l.eq.n) then
         l=l-1
    end if
!求出中心节点左下角节点标号 k，l
    k=k-1
    l=l-1
!   write(*,*),k,l
    call interpolate(ub,tb,zb,u(i,j),t(i,j),z(i,j),k,l,n)
!   write(*,*)'z(i,j)',i,j,z(i,j)          !错误，我求出的 z 都是 0
```

```
      end do
      end do
      end
!分片二次插值
      subroutine interpolate(ub,tb,zb,u,t,z,k,l,n)
      real ub(n),tb(n),zb(n,n)
      real u,t,z
      integer k,l
      dimension x1a(3),x2a(3),ya(3,3)
      real x1,x2,y,dy
      integer i,j
      integer m
      m=3
      x1=u
      x2=t

        do i=1,3
       x1a(i)=ub(k+i-1)
      do j=1,3
       x2a(j)=tb(l+j-1)
       ya(i,j)=zb(k+i-1,l+j-1)
```

```
      end do
      end do


!     write(*,*) x1,x2
!     write(*,*) (x1a(i),i=1,3)
!     write(*,*) (x2a(i),i=1,3)
      do i=1,3
      write(*,*)(ya(i,j),j=1,3)
      end do
      call POLIN2(X1A,X2A,YA,m,m,X1,X2,Y,DY)
      z=y
!     write(*,*)'DY=',dy
!     write(*,*)'Y=',y,'z=',z
      end
!分片二次插值核心算法 拉格朗日插值


      SUBROUTINE polin2(x1a,x2a,ya,m,n,x1,x2,y,dy)
      INTEGER m,n,NMAX,MMAX
      REAL dy,x1,x2,y,x1a(m),x2a(n),ya(m,n)
       PARAMETER (NMAX=20,MMAX=20)
       !USES polint
```

```
 INTEGER j,k
 REAL ymtmp(MMAX),yntmp(NMAX)
 do j=1,m
 do k=1,n
 yntmp(k)=ya(j,k)
  end do
  call polint(x2a,yntmp,n,x2,ymtmp(j),dy)
  end do
  call polint(x1a,ymtmp,m,x1,y,dy)
 END SUBROUTINE polin2

 SUBROUTINE polint(xa,ya,n,x,y,dy)
     INTEGER n,NMAX
  REAL dy,x,y,xa(n),ya(n)
 PARAMETER (NMAX=10)
 INTEGER i,m,ns
REAL den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)
  ns=1
dif=abs(x-xa(1))
do i=1,n
dift=abs(x-xa(i))
```

```
 if (dift<dif) then
 ns=i
 dif=dift
endif
c(i)=ya(i)
d(i)=ya(i)
 end do
y=ya(ns)
 ns=ns-1
 do m=1,n-1
do i=1,n-m
ho=xa(i)-x
hp=xa(i+m)-x
w=c(i+1)-d(i)
den=ho-hp
if(den==0.) pause 'failure in polint'
den=w/den
d(i)=hp*den
c(i)=ho*den
end do
if (2*ns<n-m) then
```

```
        dy=c(ns+1)
        else
        dy=d(ns)
        ns=ns-1
        endif
        y=y+dy
        end do
        END SUBROUTINE polint


!找到距离插值节点最近的节点，返回 p
        subroutine nearestp(r,s)
    integer i,s
    real r(4),t
    s=1
        t=r(1)

    do i=2,4
    if(t.GT.r(i)) then
    t=r(i)
    s=i
```

```
    end if
    end do


    end

!求两点之间的距离,并检查
    subroutine distance(x1,y1,x2,y2,d)
    real x1,y1,x2,y2,d
    d=sqrt((x1-x2)**2+(y1-y2)**2)
    end

    subroutine checkdistance()
    real x1,y1,x2,y2,r
    x1=1
    x2=2
    y1=1
    y2=2
    call distance(x1,y1,x2,y2,r)
!   write(*,*)'-------',r
```

```
      end


!读入数据课本中的表格 A1
      subroutine readdata(u,t,z,n)
      real u(n),t(n),z(n,n)
      integer i,j,k
      character*65 DUMMY
          open(1,FILE='shuju.txt',STATUS='UNKNOWN')

      read(1,300) DUMMY
      write(*,300)DUMMY

      read(1,*) (u(i),i=1,6)
      WRITE(*,*) (u(i),i=1,6)

      read(1,*) DUMMY
      write(*,*)DUMMY

      read(1,*)    (t(i),i=1,6)
      write(*,*) (t(i),i=1,6)
      read(1,300) DUMMY
```

```
      write(*,300)DUMMY


      do j=1,n


      read(1,*) (z(i,j),i=1,6)


      end do
      close(1)



!          read(1,*) z(1,:),z(2,:),z(3,:),z(4,:),z(5,:),z(6,:)


          write(*,*) z(1,:),z(2,:),z(3,:),z(4,:),z(5,:),z(6,:)



300     FORMAT(A65)


      end
      subroutine output(u,t,z,p,q)
      integer p,q
      integer i,j
```

```
real u(p,q),t(p,q),z(p,q)
open(2,FILE='utz.txt',STATUS='UNKNOWN')
write(2,*) 'u        t          z'
do i=1,p
do j=1,q
write(2,*) u(i,j),t(i,j),z(i,j)
end do
end do


end


subroutine      solvaleq(x,y,u,t,p,q)
integer p,q
real x(p),y(q),u(p,q),t(p,q)
real v,w
integer i,j


open(3,FILE='solerror.txt',STATUS='UNKNOWN')



do i=1,p
```

```
do j=1,q
call solnleq(v,w,x(i),y(j),u(i,j),t(i,j))

write(3,*) u(i,j),t(i,j)
write(3,*) 0.5*cos(t(i,j))+u(i,j)+v+w-x(i)-2.67
write(3,*) t(i,j)+0.5*sin(u(i,j))+v+w-y(j)-1.07
write(3,*) 0.5*t(i,j)+u(i,j)+cos(v)+w-x(i)-3.74
write(3,*) t(i,j)+0.5*u(i,j)+v+sin(w)-y(j)-0.79
end do
end do

end

    subroutine solnleq(v,w,x,y,u,t)
real v,w,x,y,u,t
real fanshu(2)
real d(4,100000),deltd(4)
integer i,k,j
```

```
    real er
    real error
!12.9 添加
    integer,parameter::m=4
    real A(m,m),s(m),b(m)   !解方程 As=b
    er=1.E-12
    v=1
    w=1
    u=1
    t=0


    do i=2,100000

    do k=1,m
    do j=1,m
    A(k,j)=1
    end do
    end do
    A(3,4)=0.5
    A(4,3)=0.5
```

```
      A(1,4)=-0.5*sin(t)

      A(2,3)=0.5*cos(u)

      A(3,1)=-sin(v)

      A(4,2)=cos(w)


      b(1)=-(v+w+u+0.5*cos(t)-(2.67+x))

      b(2)=-(v+w+0.5*sin(u)+t-(y+1.07))

      b(3)=-(cos(v)+w+u+0.5*t-(x+3.74))

      b(4)=-(v+sin(w)+0.5*u+t-(y+0.79))


      call soleq(m,A,b,s)
!     write(*,*)('--------')
!     write(*,*)(s(j),j=1,m)
!     write(*,*)('--------')



      v=v+s(1)

      w=w+s(2)

      u=u+s(3)

      t=t+s(4)
```

```
error=max(s(1),s(2),s(3),s(4),fanshu(1))/
     >                    max(v,w,u,t,fanshu(2))


if(error.LT.er) then
exit
end if

if (i.eq.100000.and.error.GT.er) then
write(*,*)'netown failed i=',i,error
end if

end do
END

subroutine max(a,b,c,d,e)
real a,b,c,d,e
real t,u(4)
integer i

u(1)=abs(a)
```

```
u(2)=abs(b)
u(3)=abs(c)
u(4)=abs(d)
t=u(1)

do i=2,4
if(t.LT.u(i)) then
t=u(i)
end if
end do
e=t

end

subroutine soleq(m,c,b,x)
real c(m,m),b(m),x(m)
integer i
CALL gaussj(c,m,b)
do i=1,m
x(i)=b(i)
end do
```

```
end



    SUBROUTINE gaussj(a,n,b)
     INTEGER n,NMAX
    REAL a(n,n),b(n)
     PARAMETER (NMAX=50)
    INTEGER i,icol,irow,j,k,l,ll,indxc(NMAX)
    INTEGER    indxr(NMAX),ipiv(NMAX)
    REAL big,dum,pivinv
    do j=1,n
        ipiv(j)=0
    end do
     do i=1,n
        big=0.
    do j=1,n
     if(ipiv(j)/=1) then
    do k=1,n
       if (ipiv(k)==0) then
          if (abs(a(j,k))>=big) then
```

```
            big=abs(a(j,k))
            irow=j
            icol=k
          endif
        else if (ipiv(k)>1) then
          pause 'singular matrix in gaussj'
        endif
      end do
      endif
      end do
      ipiv(icol)=ipiv(icol)+1
      if (irow/=icol) then
      do l=1,n
      dum=a(irow,l)
      a(irow,l)=a(icol,l)
      a(icol,l)=dum
      end do
      dum=b(irow)
b(irow)=b(icol)
b(icol)=dum
      endif
```

```
indxr(i)=irow
indxc(i)=icol
if (a(icol,icol)==0.) pause 'singular matrix in gaussj'
pivinv=1./a(icol,icol)
a(icol,icol)=1.
do l=1,n
a(icol,l)=a(icol,l)*pivinv
end do
b(icol)=b(icol)*pivinv
do ll=1,n
if(ll/=icol) then
dum=a(ll,icol)
a(ll,icol)=0.
do l=1,n
   a(ll,l)=a(ll,l)-a(icol,l)*dum
end do
b(ll)=b(ll)-b(icol)*dum
 endif
end do
end do
do l=n,1,-1
```

```
if(indxr(l)/=indxc(l)) then
do k=1,n
dum=a(k,indxr(l))
a(k,indxr(l))=a(k,indxc(l))
a(k,indxc(l))=dum
end do
endif
end do
 END SUBROUTINE gaussj
```

三、运行结果

3.1 数表(x,y,z)

x y z

0.000000000000000E+000 0.500000000000000 0.446504018479934

0.000000000000000E+000 0.550000000000000 0.324683262927430

0.000000000000000E+000 0.600000000000000 0.210159686682549

0.000000000000000E+000 0.650000000000000 0.103043608315948

0.000000000000000E+000 0.700000000000000 3.401895562658904E-003

0.000000000000000E+000 0.750000000000000 -8.873581363800537E-002

0.000000000000000E+000 0.800000000000000 -0.173371632749730

0.000000000000000E+000 0.850000000000000 -0.250534611466608

0.000000000000000E+000 0.900000000000000 -0.320276506387631

0.000000000000000E+000 0.950000000000000 -0.382668066109709

0.000000000000000E+000 1.00000000000000 -0.437795766738356

0.000000000000000E+000 1.05000000000000

-0.485758941443773

0.000000000000000E+000 1.10000000000000

-0.526667254883527

0.000000000000000E+000 1.15000000000000

-0.560638479796510

0.000000000000000E+000 1.20000000000000

-0.587796538767668

0.000000000000000E+000 1.25000000000000

-0.608269779089934

0.000000000000000E+000 1.30000000000000

-0.622189452876368

0.000000000000000E+000 1.35000000000000

-0.629688378185600

0.000000000000000E+000 1.40000000000000

-0.630899760002776

0.000000000000000E+000 1.45000000000000

-0.625956152545434

0.000000000000000E+000 1.50000000000000

-0.614988546609357

8.000000000000000E-002 0.500000000000000

0.638015226510213

8.000000000000000E-002 0.550000000000000
0.506611755146211
8.000000000000000E-002 0.600000000000000
0.382176369277215
8.000000000000000E-002 0.650000000000000
0.264863491153617
8.000000000000000E-002 0.700000000000000
0.154780200284818
8.000000000000000E-002 0.750000000000000
5.199268349092778E-002
8.000000000000000E-002 0.800000000000000
-4.346804020490879E-002
8.000000000000000E-002 0.850000000000000
-0.131601056788529
8.000000000000000E-002 0.900000000000000
-0.212431088308821
8.000000000000000E-002 0.950000000000000
-0.286004551058031
8.000000000000000E-002 1.00000000000000
-0.352386078979358
8.000000000000000E-002 1.05000000000000

-0.411655456522184

8.000000000000000E-002 1.100000000000000

-0.463904911518766

8.000000000000000E-002 1.150000000000000

-0.509236724700547

8.000000000000000E-002 1.200000000000000

-0.547761117962330

8.000000000000000E-002 1.250000000000000

-0.579594388339079

8.000000000000000E-002 1.300000000000000

-0.604857258889532

8.000000000000000E-002 1.350000000000000

-0.623673421331758

8.000000000000000E-002 1.400000000000000

-0.636168248413258

8.000000000000000E-002 1.450000000000000

-0.642467656690050

8.000000000000000E-002 1.500000000000000

-0.642697102699610

0.160000000000000 0.500000000000000

0.840081395765142

0.160000000000000 0.550000000000000 0.699764165672570

0.160000000000000 0.600000000000000 0.566061442351442

0.160000000000000 0.650000000000000 0.439171608117521

0.160000000000000 0.700000000000000 0.319242138040752

0.160000000000000 0.750000000000000 0.206376192387373

0.160000000000000 0.800000000000000 0.100638523891430

0.160000000000000 0.850000000000000 2.060740067835784E-003

0.160000000000000 0.900000000000000 -8.935402476698084E-002

0.160000000000000 0.950000000000000 -0.173626968864850

0.160000000000000 1.00000000000000 -0.250799956159861

0.160000000000000 1.05000000000000

-0.320932269444613

0.160000000000000 1.10000000000000

-0.384097735004564

0.160000000000000 1.15000000000000

-0.440382175417512

0.160000000000000 1.20000000000000

-0.489881152312580

0.160000000000000 1.25000000000000

-0.532697965533796

0.160000000000000 1.30000000000000

-0.568941879292126

0.160000000000000 1.35000000000000

-0.598726549515135

0.160000000000000 1.40000000000000

-0.622168629750329

0.160000000000000 1.45000000000000

-0.639386535697154

0.160000000000000 1.50000000000000

-0.650499350787842

0.240000000000000 0.500000000000000

1.05151509180126

0.240000000000000 0.550000000000000 0.902927430830220

0.240000000000000 0.600000000000000 0.760580266859262

0.240000000000000 0.650000000000000 0.624715198145480

0.240000000000000 0.700000000000000 0.495519756000861

0.240000000000000 0.750000000000000 0.373134042774563

0.240000000000000 0.800000000000000 0.257656748872341

0.240000000000000 0.850000000000000 0.149150559410230

0.240000000000000 0.900000000000000 4.764698677336622E-002

0.240000000000000 0.950000000000000 -4.684932320146153E-002

0.240000000000000 1.00000000000000 -0.134356760384909

0.240000000000000 1.05000000000000

-0.214913344927402

0.240000000000000 1.10000000000000

-0.288573700634810

0.240000000000000 1.15000000000000

-0.355406364785712

0.240000000000000 1.20000000000000

-0.415491396488576

0.240000000000000 1.25000000000000

-0.468918249969471

0.240000000000000 1.30000000000000

-0.515783883124747

0.240000000000000 1.35000000000000

-0.556191075200143

0.240000000000000 1.40000000000000

-0.590246930562859

0.240000000000000 1.45000000000000

-0.618061548241234

0.240000000000000 1.50000000000000

-0.639746839257865

0.320000000000000 0.500000000000000

1.27124675148115

0.320000000000000 0.550000000000000 1.11500201814598

0.320000000000000 0.600000000000000 0.964607727215351

0.320000000000000 0.650000000000000 0.820347369474939

0.320000000000000 0.700000000000000 0.682447678179402

0.320000000000000 0.750000000000000 0.551085208597475

0.320000000000000 0.800000000000000 0.426392385901768

0.320000000000000 0.850000000000000 0.308462995633190

0.320000000000000 0.900000000000000 0.197357129691923

0.320000000000000 0.950000000000000 9.310562085940516E-002

0.320000000000000 1.00000000000000 -4.285992234033330E-003

0.320000000000000 1.05000000000000

-9.483392529688754E-002

0.320000000000000 1.10000000000000

-0.178572990363992

0.320000000000000 1.15000000000000

-0.255553779054632

0.320000000000000 1.20000000000000

-0.325840150157500

0.320000000000000 1.25000000000000

-0.389506988363433

0.320000000000000 1.30000000000000

-0.446638204599494

0.320000000000000 1.35000000000000

-0.497324951767697

0.320000000000000 1.40000000000000

-0.541664032699418

0.320000000000000 1.45000000000000

-0.579756479795119

0.320000000000000 1.50000000000000

-0.611706288147578

0.400000000000000 0.500000000000000

1.49832105248068

0.400000000000000 0.550000000000000 1.33499863206573

0.400000000000000 0.600000000000000 1.17712512373886

0.400000000000000 0.650000000000000 1.02502405501997

0.400000000000000 0.700000000000000 0.878960023174302

0.400000000000000 0.750000000000000 0.739145108703474

0.400000000000000 0.800000000000000 0.605744871487092

0.400000000000000 0.850000000000000 0.478883861066560

0.400000000000000 0.900000000000000 0.358650625881831

0.400000000000000 0.950000000000000 0.245102236196428

0.400000000000000 1.00000000000000 0.138268350928467

0.400000000000000 1.05000000000000

3.815486540699203E-002

0.400000000000000 1.100000000000000

-5.525282116814331E-002

0.400000000000000 1.150000000000000

-0.141986880813721

0.400000000000000 1.200000000000000

-0.222094439095899

0.400000000000000 1.250000000000000

-0.295635232459798

0.400000000000000 1.300000000000000

-0.362679511502828

0.400000000000000 1.350000000000000

-0.423306164223961

0.400000000000000 1.400000000000000

-0.477601036132450

0.400000000000000 1.450000000000000

-0.525655426667203

0.400000000000000 1.500000000000000

-0.567564743655107

0.480000000000000 0.500000000000000

1.73189274038194

0.480000000000000 0.550000000000000 1.56203457720811

0.480000000000000 0.600000000000000 1.39721691820796

0.480000000000000 0.650000000000000 1.23780100673931

0.480000000000000 0.700000000000000 1.08408753267769

0.480000000000000 0.750000000000000 0.936322772314868

0.480000000000000 0.800000000000000 0.794704449053701

0.480000000000000 0.850000000000000 0.659387198028234

0.480000000000000 0.900000000000000 0.530487586839948

0.480000000000000 0.950000000000000 0.408088685454233

0.480000000000000 1.00000000000000 0.292244201229539

0.480000000000000 1.05000000000000

0.182982206853550

0.480000000000000 1.10000000000000

8.030849403542600E-002

0.480000000000000 1.15000000000000

-1.579041305164065E-002

0.480000000000000 1.20000000000000

-0.105344551620979

0.480000000000000 1.25000000000000

-0.188398090609595

0.480000000000000 1.30000000000000

-0.265007149318921

0.480000000000000 1.35000000000000

-0.335237838904025

0.480000000000000 1.40000000000000

-0.399164503886759

0.480000000000000 1.45000000000000

-0.456868143301566

0.480000000000000 1.50000000000000

-0.508434993278188

0.560000000000000 0.500000000000000

1.97122178639989

0.560000000000000 0.550000000000000
1.79532959950066
0.560000000000000 0.600000000000000
1.62406711322791
0.560000000000000 0.650000000000000
1.45783058270776
0.560000000000000 0.700000000000000
1.29695464975211
0.560000000000000 0.750000000000000
1.14171810544722
0.560000000000000 0.800000000000000
0.992349533324319
0.560000000000000 0.850000000000000
0.849032663329384
0.560000000000000 0.900000000000000
0.711911352264108
0.560000000000000 0.950000000000000
0.581094158921853
0.560000000000000 1.00000000000000
0.456658513233449
0.560000000000000 1.05000000000000

0.338654496139362

0.560000000000000 1.10000000000000

0.227108255769633

0.560000000000000 1.15000000000000

0.122025089193156

0.560000000000000 1.20000000000000

2.339221963760311E-002

0.560000000000000 1.25000000000000

-6.881870197104234E-002

0.560000000000000 1.30000000000000

-0.154649344212872

0.560000000000000 1.35000000000000

-0.234152666458688

0.560000000000000 1.40000000000000

-0.307391091913287

0.560000000000000 1.45000000000000

-0.374434862348063

0.560000000000000 1.50000000000000

-0.435360556535884

0.640000000000000 0.500000000000000

2.21566786368768

0.640000000000000 0.550000000000000 2.03420113360723

0.640000000000000 0.600000000000000 1.85695514361904

0.640000000000000 0.650000000000000 1.68435816416101

0.640000000000000 0.700000000000000 1.51677635239996

0.640000000000000 0.750000000000000 1.35451904115140

0.640000000000000 0.800000000000000 1.19784408667268

0.640000000000000 0.850000000000000 1.04696304941940

0.640000000000000 0.900000000000000 0.902046083802266

0.640000000000000 0.950000000000000 0.763226477662872

0.640000000000000 1.00000000000000 0.630604821954327

0.640000000000000 1.05000000000000

0.504252814597223

0.640000000000000 1.10000000000000

0.384216715545667

0.640000000000000 1.15000000000000

0.270520476640969

0.640000000000000 1.20000000000000

0.163168572399612

0.640000000000000 1.25000000000000

6.214855811676047E-002

0.640000000000000 1.30000000000000

-3.256661939681810E-002

0.640000000000000 1.35000000000000

-0.121016534844361

0.640000000000000 1.40000000000000

-0.203251399622820

0.640000000000000 1.45000000000000

-0.279330359558427

0.640000000000000 1.50000000000000

-0.349319957540006

0.720000000000000 0.500000000000000

2.46468422265956

0.720000000000000 0.550000000000000 2.27805897939853

0.720000000000000 0.600000000000000 2.09525125084029

0.720000000000000 0.650000000000000 1.91671812799728

0.720000000000000 0.700000000000000 1.74285462877588

0.720000000000000 0.750000000000000 1.57399842733412

0.720000000000000 0.800000000000000 1.41043483523083

0.720000000000000 0.850000000000000 1.25240175060805

0.720000000000000 0.900000000000000 1.10009440962797

0.720000000000000 0.950000000000000 0.953669851261271

0.720000000000000 1.00000000000000 0.813251055248882

0.720000000000000 1.05000000000000

0.678930742965944

0.720000000000000 1.10000000000000

0.550774848504326

0.720000000000000 1.15000000000000

0.428825676973075

0.720000000000000 1.20000000000000

0.313104771739816

0.720000000000000 1.25000000000000

0.203615514032691

0.720000000000000 1.30000000000000

0.100345478240894

0.720000000000000 1.35000000000000

3.268565186572091E-003

0.720000000000000 1.40000000000000

-8.765306591328641E-002

0.720000000000000 1.45000000000000

-0.172467247818849

0.720000000000000 1.50000000000000

-0.251230220752334

0.800000000000000 0.500000000000000

2.71781110946857

0.800000000000000 0.550000000000000
2.52639950125573
0.800000000000000 0.600000000000000
2.33841138685977
0.800000000000000 0.650000000000000
2.15432937728033
0.800000000000000 0.700000000000000
1.97457455665203
0.800000000000000 0.750000000000000
1.79951057909949
0.800000000000000 0.800000000000000
1.62944822055405
0.800000000000000 0.850000000000000
1.46465004375065
0.800000000000000 0.900000000000000
1.30533496765067
0.800000000000000 0.950000000000000
1.15168262130696
0.800000000000000 1.00000000000000
1.00383741990551
0.800000000000000 1.05000000000000

0.861912337227926

0.800000000000000 1.10000000000000

0.725992371111170

0.800000000000000 1.15000000000000

0.596137711520083

0.800000000000000 1.20000000000000

0.472386627913592

0.800000000000000 1.25000000000000

0.354758095897906

0.800000000000000 1.30000000000000

0.243254184181296

0.800000000000000 1.35000000000000

0.137862222524720

0.800000000000000 1.40000000000000

3.855677032640460E-002

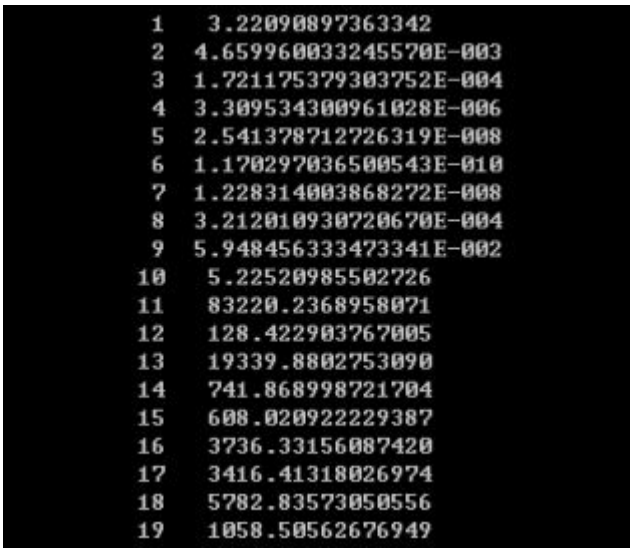
0.800000000000000 1.45000000000000

-5.469859593445851E-002

0.800000000000000 1.50000000000000

-0.141949659708795

3.2 k 的选择过程：



实际运行时 K=5 满足要求，终止选择过程

k= 5

error= 2.541378712726319E-008

3.3 系数

c(r,s)(r=0,1,...,5,s=0,1,...,5)

r=	s=	Crs=
1	1	2.02123051934281
1	2	-3.66842679875870
1	3	0.709248631255292
1	4	0.848605391815966
1	5	-0.415897428314167
1	6	6.743199431922875E-002

2	1	3.19190900840529
2	2	-0.741110375772280
2	3	-2.69712460647187
2	4	1.63118340597396
2	5	-0.484719977251359
2	6	6.061428150603732E-002
3	1	0.256889810834281
3	2	1.57991867478358
3	3	-0.463408167820376
3	4	-8.134436905686471E-002
3	5	0.102094226577037
3	6	-2.101523151640095E-002
4	1	-0.269260334794732
4	2	-0.730247662554447
4	3	1.07614508228124
4	4	-0.807012808769116
4	5	0.302872797206007
4	6	-4.597263283499586E-002
5	1	0.217459754434830
5	2	-0.178372392878291
5	3	-7.240576143929189E-002

5 4 0.243330413273405

5 5 -0.141334703268615

5 6 2.651023281508191E-002

6 1 -5.590326094637071E-002

6 2 0.143199229958642

6 3 -0.136270361272305

6 4 4.071962890214298E-002

6 5 3.775021356565045E-003

6 6 -2.667697763727328E-003

3.4 数表(x,y,z,p)

x*[i],y*[j],f(x*[i],y*[j])，P(x*[i],y*[j])

0.1 0.7 0.194720407918E+00 0.194730359841E+00

0.1 0.9 -0.183037079189E+00 -0.183041834780E+00

0.1 1.1 -0.445497646915E+00 -0.445500035170E+00

0.1 1.3 -0.597566707641E+00 -0.597558846118E+00

0.1 1.5 -0.646459593901E+00 -0.646446093652E+00

0.2 0.7 0.405979189288E+00 0.405989541713E+00

0.2 0.9 -0.225159583746E-01 -0.225211128653E-01

0.2 1.1 -0.338220816040E+00 -0.338224017159E+00

0.2 1.3 -0.544437831522E+00 -0.544430442159E+00

0.2 1.5 -0.647361338568E+00 -0.647347997029E+00

0.3 0.7 0.634777195151E+00 0.634787454291E+00

0.3 0.9 0.158801168839E+00 0.158796298315E+00

0.3 1.1 -0.207365694171E+00 -0.207368576581E+00

0.3 1.3 -0.465357906898E+00 -0.465349917206E+00

0.3 1.5 -0.620270953075E+00 -0.620257129388E+00

0.4 0.7 0.878960023174E+00 0.878969866056E+00

0.4 0.9 0.358650625882E+00 0.358646044527E+00

0.4 1.1 -0.552528211681E-01 -0.552554350035E-01

0.4 1.3 -0.362679511503E+00 -0.362671060080E+00

0.4 1.5 -0.567564743655E+00 -0.567550578371E+00

0.5 0.7 0.113661091016E+01 0.113662035325E+01

0.5 0.9 0.574980340948E+00 0.574975842600E+00

0.5 1.1 0.115992376792E+00 0.115989321142E+00

0.5 1.3 -0.238568304012E+00 -0.238560420022E+00

0.5 1.5 -0.491434393656E+00 -0.491420901995E+00

0.6 0.7 0.140604179891E+01 0.140605068617E+01

0.6 0.9 0.805941494063E+00 0.805937300268E+00

0.6 1.1 0.304429221045E+00 0.304425829244E+00

0.6 1.3 -0.950161300996E-01 -0.950089506394E-01

0.6 1.5 -0.393902307746E+00 -0.393889845572E+00

0.7 0.7 0.168578351531E+01 0.168579121547E+01

0.7 0.9 0.104988115306E+01 0.104987773602E+01

0.7 1.1 0.508293783940E+00 0.508291039197E+00

0.7 1.3 0.661487967065E-01 0.661563461587E-01

0.7 1.5 -0.276834341778E+00 -0.276822058627E+00

0.8 0.7 0.197457455665E+01 0.197458125845E+01

0.8 0.9 0.130533496765E+01 0.130533199895E+01

0.8 1.1 0.725992371111E+00 0.725989302079E+00

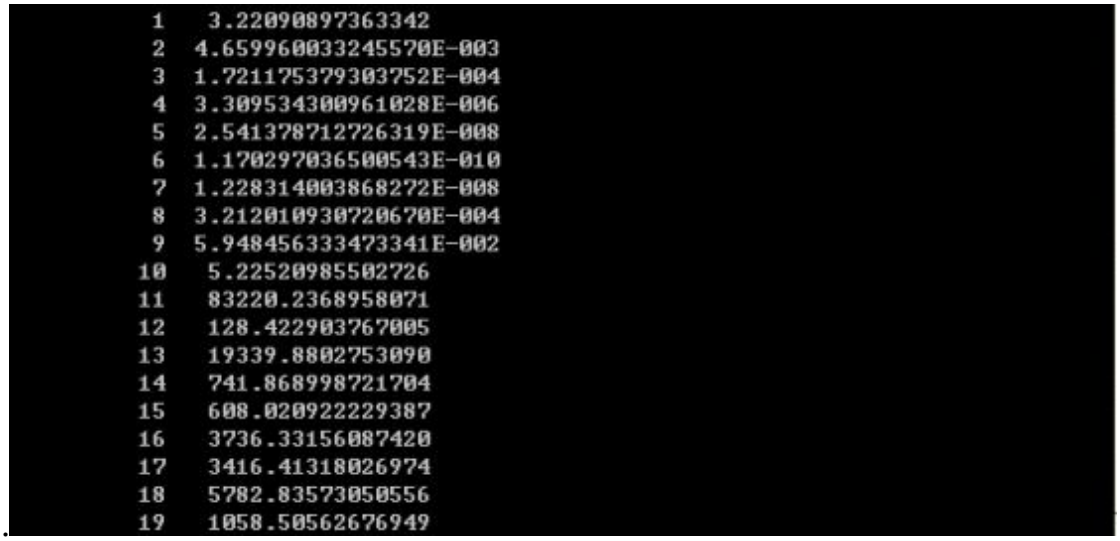
0.8 1.3 0.243254184181E+00 0.243260776217E+00

0.8 1.5 -0.141949659709E+00 -0.141938811820E+00

四．遇到的问题与体会

1.求解非线性方程组不可以用简单迭代法，因为矩阵 G′ (X*)的谱半径不能保证恒小于 1，我最初选用简单迭代法发现求解的部分结果错误。故改用牛顿法

2.求解线性方程组时一定要用选主元的 Gauss 或 Doolittle 法，不进行选主元操作会导致消元过程主对角线值为 0，从而求解发散。

3. 

如上图所示，左边为曲面拟合的次数 K,右边数据为拟合误差 error，可见误差随拟合次数的增加呈现先减小后增大的过程，即高次（>10）拟合会造成节点处的值不收敛到已知值，这一点同拉格朗日插值过程的 Runge 现象类似。

4.我发现数值分析这门课类似于本科的电工电子实验，前者主要包含设计算法与上机实现两个过程，后者则是设计电路图以及通过实验来验证这两个过程，相同之处在于综合要求理论分析与实际动手能力，学以致用。