

[原创]偏微分方程数值解法的 MATLAB 源码【更新完毕】

说明：由于偏微分的程序都比较长，比其他的算法稍复杂一些，所以另开一贴，专门上传偏微分的程序
谢谢大家的支持！

其他的数值算法见：

[../Announce/Announce.asp?BoardID=209&id=8245004](http://Announce/Announce.asp?BoardID=209&id=8245004)

1、古典显式格式求解抛物型偏微分方程（一维热传导方程）

```
function [U x t]=PDEParabolicClassicalExplicit(uX,uT,phi,psi1,psi2,M,N,C)
%古典显式格式求解抛物型偏微分方程
%[U x t]=PDEParabolicClassicalExplicit(uX,uT,phi,psi1,psi2,M,N,C)
%
%方程：  $u_t = C * u_{xx}$   $0 \leq x \leq uX, 0 \leq t \leq uT$ 
%初值条件：  $u(x,0) = \phi(x)$ 
%边值条件：  $u(0,t) = \psi_1(t), u(uX,t) = \psi_2(t)$ 
%
%输出参数：U -解矩阵，第一行表示初值，第一列和最后一列表示边值，第二行表示第2层.....
%      x -空间变量
%      t -时间变量
%输入参数：uX -空间变量 x 的取值上限
%      uT -时间变量 t 的取值上限
%      phi -初值条件，定义为内联函数
%      psi1 -边值条件，定义为内联函数
%      psi2 -边值条件，定义为内联函数
%      M -沿 x 轴的等分区间数
%      N -沿 t 轴的等分区间数
%      C -系数，默认情况下 C=1
%
%应用举例：
%uX=1;uT=0.2;M=15;N=100;C=1;
%phi=inline('sin(pi*x)');psi1=inline('0');psi2=inline('0');
%[U x t]=PDEParabolicClassicalExplicit(uX,uT,phi,psi1,psi2,M,N,C);

%设置参数 C 的默认值
if nargin==7
    C=1;
end

%计算步长
dx=uX/M;%x 的步长
dt=uT/N;%t 的步长
```

```

x=(0:M)*dx;
t=(0:N)*dt;

r=C*dt/dx/dx;%步长比
r1=1-2*r;

if r > 0.5
    disp('r > 0.5,不稳定')
end

%计算初值和边值
U=zeros(M+1,N+1);
for i=1:M+1
    U(i,1)=phi(x(i));
end
for j=1:N+1
    U(1,j)=psi1(t(j));
    U(M+1,j)=psi2(t(j));
end

%逐层求解
for j=1:N
    for i=2:M
        U(i,j+1)=r*U(i-1,j)+r1*U(i,j)+r*U(i+1,j);
    end
end

U=U';

%作出图形
mesh(x,t,U);
title('古典显式格式，一维热传导方程的解的图像')
xlabel('空间变量 x')
ylabel('时间变量 t')
zlabel('一维热传导方程的解 U')

return;

```

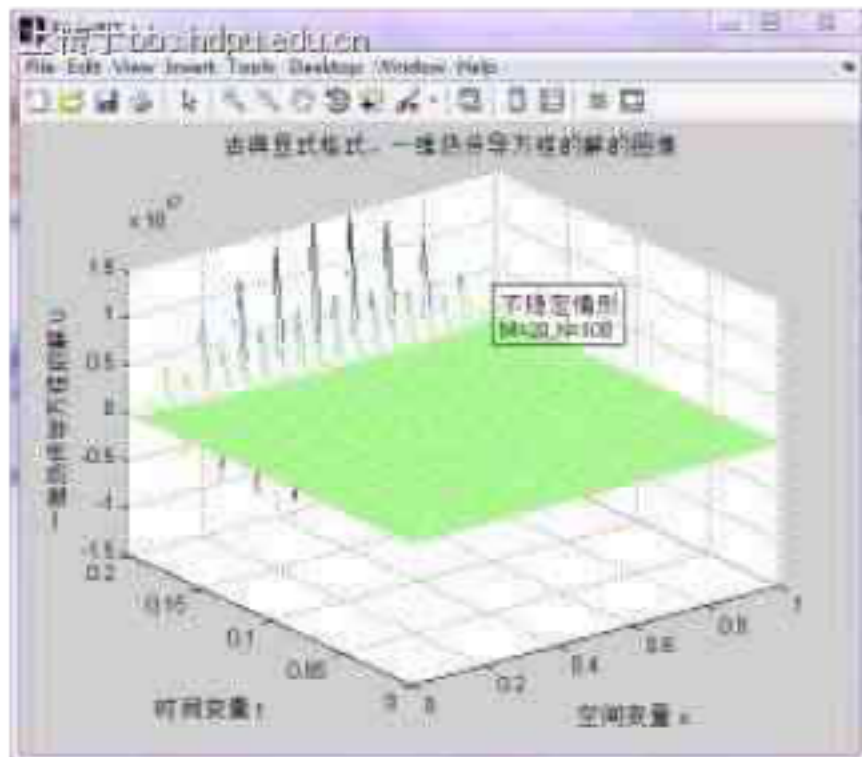
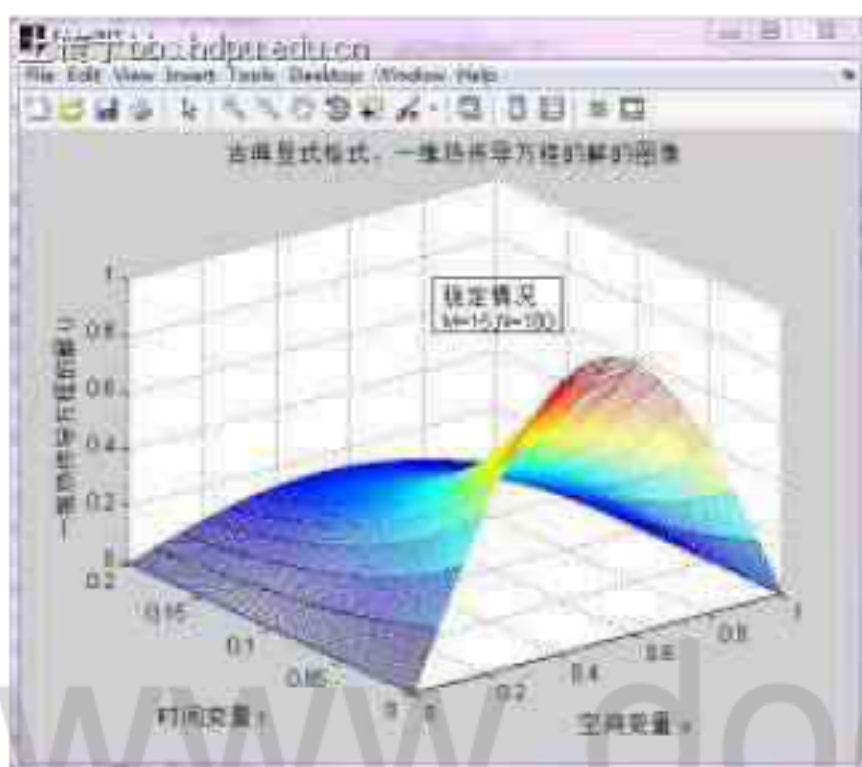


图 1 古典显式格式稳定情况



2、古典隐式格式求解抛物型偏微分方程（一维热传导方程）

```
function [U x t]=PDEParabolicClassicalImplicit(uX,uT,phi,psi1,psi2,M,N,C)
%古典隐式格式求解抛物型偏微分方程
%[U x t]=PDEParabolicClassicalImplicit(uX,uT,phi,psi1,psi2,M,N,C)
%
%方程:  $u_t = C * u_{xx}$   $0 \leq x \leq uX, 0 \leq t \leq uT$ 
%初值条件:  $u(x,0)=\phi(x)$ 
%边值条件:  $u(0,t)=\psi_1(t), u(uX,t)=\psi_2(t)$ 
%
%输出参数: U -解矩阵, 第一行表示初值, 第一列和最后一列表示边值, 第二行表示第 2 层.....
%      x -空间变量
%      t -时间变量
%输入参数: uX -空间变量 x 的取值上限
```

```

%      uT -时间变量 t 的取值上限
%      phi -初值条件，定义为内联函数
%      psi1 -边值条件，定义为内联函数
%      psi2 -边值条件，定义为内联函数
%      M -沿 x 轴的等分区间数
%      N -沿 t 轴的等分区间数
%      C -系数，默认情况下 C=1
%
%应用举例：
%uX=1;uT=0.2;M=50;N=50;C=1;
%phi=inline('sin(pi*x)');psi1=inline('0');psi2=inline('0');
%[U x t]=PDEParabolicClassicalImplicit(uX,uT,phi,psi1,psi2,M,N,C);

%设置参数 C 的默认值
if nargin==7
    C=1;
end

%计算步长
dx=uX/M;%x 的步长
dt=uT/N;%t 的步长

x=(0:M)*dx;
t=(0:N)*dt;

r=C*dt/dx/dx;%步长比

Diag=zeros(1,M-1);%矩阵的对角线元素
Low=zeros(1,M-2);%矩阵的下对角线元素
Up=zeros(1,M-2);%矩阵的上对角线元素
for i=1:M-2
    Diag(i)=1+2*r;
    Low(i)=-r;
    Up(i)=-r;
end
Diag(M-1)=1+2*r;

%计算初值和边值
U=zeros(M+1,N+1);
for i=1:M+1
    U(i,1)=phi(x(i));
end
for j=1:N+1

```

```

    U(1,j)=psi1(t(j));
    U(M+1,j)=psi2(t(j));
end

%逐层求解，需要使用追赶法（调用函数 EqtsForwardAndBackward）
for j=1:N
    b1=zeros(M-1,1);
    b1(1)=r*U(1,j+1);
    b1(M-1)=r*U(M+1,j+1);
    b=U(2:M,j)+b1;
    U(2:M,j+1)=EqtsForwardAndBackward(Low,Diag,Up,b);
end

U=U';

%作出图形
mesh(x,t,U);
title('古典隐式格式，一维热传导方程的解的图像')
xlabel('空间变量 x')
ylabel('时间变量 t')
zlabel('一维热传导方程的解 U')

return;

```

此算法需要使用追赶法求解三对角线性方程组，这个算法在上一篇帖子中已经给出，为了方便，再给出来

追赶法解三对角线性方程组

```

function x=EqtsForwardAndBackward(L,D,U,b)
%追赶法求解三对角线性方程组 Ax=b
%x=EqtsForwardAndBackward(L,D,U,b)
%x:三对角线性方程组的解
%L:三对角矩阵的下对角线，行向量
%D:三对角矩阵的对角线，行向量
%U:三对角矩阵的上对角线，行向量
%b:线性方程组 Ax=b 中的 b，列向量
%
%应用举例:
%L=[-1 -2 -3];D=[2 3 4 5];U=[-1 -2 -3];b=[6 1 -2 1]';
%x=EqtsForwardAndBackward(L,D,U,b)

```

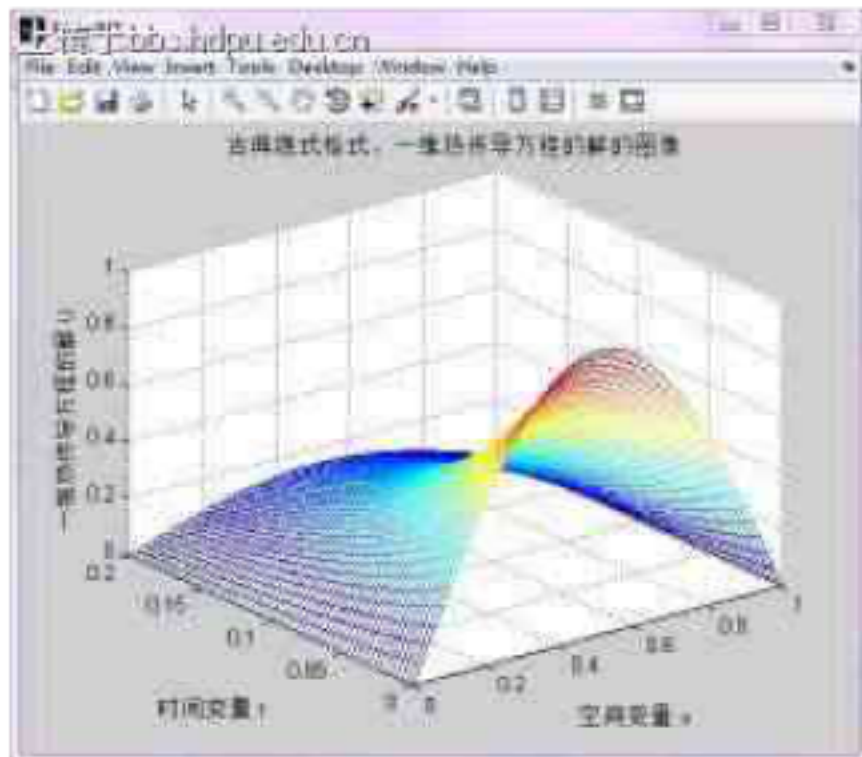
```

%检查参数的输入是否正确
n=length(D);m=length(b);
n1=length(L);n2=length(U);
if n-n1 ~= 1 || n-n2 ~= 1 || n ~= m
    disp('输入参数有误! ')
    x=' ';
    return;
end

%追的过程
for i=2:n
    L(i-1)=L(i-1)/D(i-1);
    D(i)=D(i)-L(i-1)*U(i-1);
end
x=zeros(n,1);
x(1)=b(1);
for i=2:n
    x(i)=b(i)-L(i-1)*x(i-1);
end

%赶的过程
x(n)=x(n)/D(n);
for i=n-1:-1:1
    x(i)=(x(i)-U(i)*x(i+1))/D(i);
end
return;

```

在以后的程序中，我们都取 $C=1$ ，不再作为一个输入参数处理

3、Crank-Nicolson 隐式格式求解抛物型偏微分方程 需要调用追赶法的程序

```
function [U x t]=PDEParabolicCN(uX,uT,phi,psi1,psi2,M,N)
%Crank-Nicolson 隐式格式求解抛物型偏微分方程
%[U x t]=PDEParabolicCN(uX,uT,phi,psi1,psi2,M,N)
%
%方程:  $u_t = u_{xx}$   $0 \leq x \leq uX, 0 \leq t \leq uT$ 
%初值条件:  $u(x,0)=\phi(x)$ 
%边值条件:  $u(0,t)=\psi_1(t), u(uX,t)=\psi_2(t)$ 
%
%输出参数: U -解矩阵, 第一行表示初值, 第一列和最后一列表示边值, 第二行表示第 2 层.....
%      x -空间变量
%      t -时间变量
%输入参数: uX -空间变量 x 的取值上限
%      uT -时间变量 t 的取值上限
%      phi -初值条件, 定义为内联函数
%      psi1 -边值条件, 定义为内联函数
%      psi2 -边值条件, 定义为内联函数
%      M -沿 x 轴的等分区间数
%      N -沿 t 轴的等分区间数
%
%应用举例:
%uX=1;uT=0.2;M=50;N=50;
%phi=inline('sin(pi*x)');psi1=inline('0');psi2=inline('0');
%[U x t]=PDEParabolicCN(uX,uT,phi,psi1,psi2,M,N);
```

```

%计算步长
dx=uX/M;%x 的步长
dt=uT/N;%t 的步长

x=(0:M)*dx;
t=(0:N)*dt;

r=dt/dx/dx;%步长比

Diag=zeros(1,M-1);%矩阵的对角线元素
Low=zeros(1,M-2);%矩阵的下对角线元素
Up=zeros(1,M-2);%矩阵的上对角线元素
for i=1:M-2
    Diag(i)=1+r;
    Low(i)=-r/2;
    Up(i)=-r/2;
end
Diag(M-1)=1+r;

%计算初值和边值
U=zeros(M+1,N+1);
for i=1:M+1
    U(i,1)=phi(x(i));
end
for j=1:N+1
    U(1,j)=psi1(t(j));
    U(M+1,j)=psi2(t(j));
end

B=zeros(M-1,M-1);
for i=1:M-2
    B(i,i)=1-r;
    B(i,i+1)=r/2;
    B(i+1,i)=r/2;
end
B(M-1,M-1)=1-r;

%逐层求解，需要使用追赶法（调用函数 EqtsForwardAndBackward）
for j=1:N
    b1=zeros(M-1,1);
    b1(1)=r*(U(1,j+1)+U(1,j))/2;
    b1(M-1)=r*(U(M+1,j+1)+U(M+1,j))/2;
    b=B*U(2:M,j)+b1;

```



```


    U(2:M,j+1)=EqtsForwardAndBackward(Low,Diag,Up,b);
end

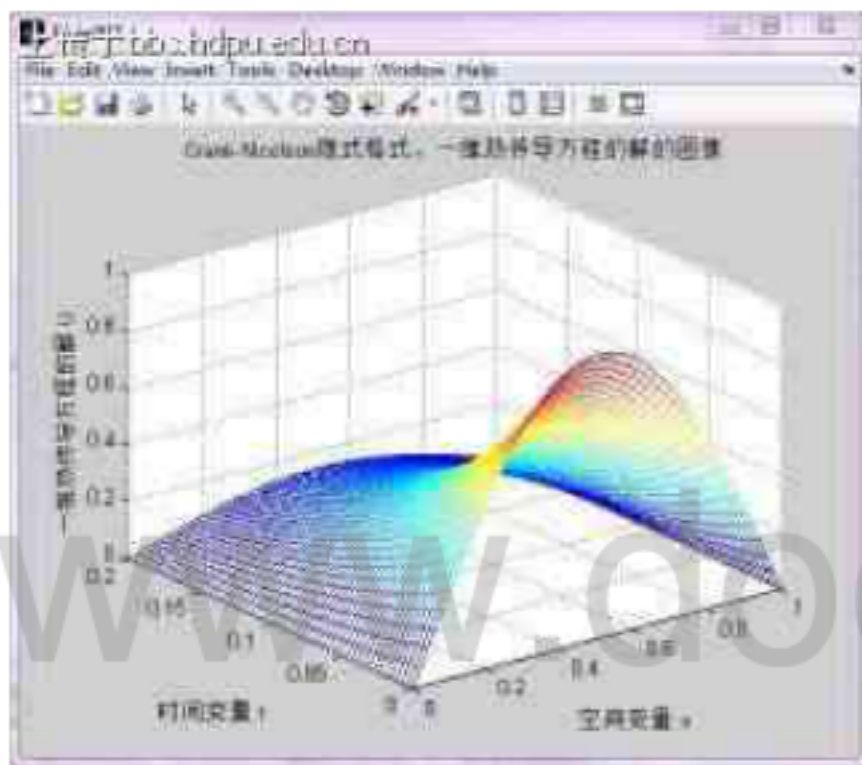
U=U';

%作出图形
mesh(x,t,U);
title('Crank-Nicolson 隐式格式，一维热传导方程的解的图像')
xlabel('空间变量 x')
ylabel('时间变量 t')
zlabel('一维热传导方程的解 U')

return;

```

 Crank-Nicolson 隐式格式



4、正方形区域 Laplace 方程 Diriclet 问题的求解

需要调用 Jacobi 迭代法和 Guass-Seidel 迭代法求解线性方程组

```

function [U x y]=PDEEllipseSquareLaplaceDirichlet(ub,phi1,phi2,psi1,psi2,M,type)
%正方形区域 Laplace 方程的 Diriclet 边值问题的差分求解
%此程序需要调用 Jacobi 迭代法或者 Guass-Seidel 迭代法求解线性方程组
%[U x y]=PDEEllipseSquareLaplaceDirichlet(ub,phi1,phi2,psi1,psi2,M,type)
%
%方程:  $u_{xx}+u_{yy}=0$   $0 \leq x,y \leq ub$ 
%边值条件:  $u(0,y)=\phi_1(y)$ 

```

```

%      u(ub,y)=phi2(y)
%      u(x,0)=psi1(x)
%      u(x,ub)=psi2(x)
%
%输出参数: U -解矩阵, 第一行表示 y=0 时的值, 第二行表示第 y=h 时的值.....
%      x -横坐标
%      y -纵坐标
%输入参数: ub -变量边界值的上限
%      phi1,phi2,psi1,psi2 -边界函数, 定义为内联函数
%      M -横纵坐标的等分区间数
%      type -求解差分方程的迭代格式, 若 type='Jacobi', 采用 Jacobi 迭代格式
%           若 type='GS', 采用 Guass-Seidel 迭代格式。默认情况下, type='GS'
%
%应用举例:
%ub=4;M=20;
%phi1=inline('y*(4-y)');phi2=inline('0');psi1=inline('sin(pi*x/4)');psi2=inline('0');
%[U x y]=PDEEllipseSquareLaplaceDirichlet(ub,phi1,phi2,psi1,psi2,M,'GS');

if nargin==6
    type='GS';
end

%步长
h=ub/M;
%横纵坐标
x=(0:M)*h;
y=(0:M)*h;

%差分格式的矩阵形式 AU=K
%构造矩阵 A
M2=(M-1)^2;
A=zeros(M2);
for i=1:M2
    A(i,i)=4;
end
for i=1:M2-1
    if mod(i,M-1)~=0
        A(i,i+1)=-1;
        A(i+1,i)=-1;
    end
end
for i=1:M2-M+1
    A(i,i+M-1)=-1;

```

```

        A(i+M-1,i)=-1;
    end

    U=zeros(M+1);
    %边值条件
    for i=1:M+1
        U(i,1)=psi1((i-1)*h);
        U(i,M+1)=psi2((i-1)*h);
        U(1,i)=phi1((i-1)*h);
        U(M+1,i)=phi2((i-1)*h);
    end

    %构造 K
    K=zeros(M2,1);
    for i=1:M-1
        K(i)=U(i+1,1);
        K(M2-i+1)=U(i+1,M+1);
    end
    K(1)=K(1)+U(1,2);
    K(M-1)=K(M-1)+U(M+1,2);
    K(M2-M+2)=K(M2-M+2)+U(1,M);
    K(M2)=K(M2)+U(M+1,M);
    for i=2:M-2
        K((M-1)*(i-1)+1)=U(1,i+1);
        K((M-1)*i)=U(M+1,i+1);
    end

    x0=ones(M2,1);
    switch type
        %调用 Guass-Seidel 迭代法求解线性方程组 AU=K
        case 'Jacobi'
            X=EqtsJacobi(A,K,x0);
        %调用 Guass-Seidel 迭代法求解线性方程组 AU=K
        case 'GS'
            X=EqtsGS(A,K,x0);
        otherwise
            disp('差分格式类型输入错误')
            return;
    end

    %把求解结果化成矩阵型式
    for i=2:M
        for j=2:M

```

```

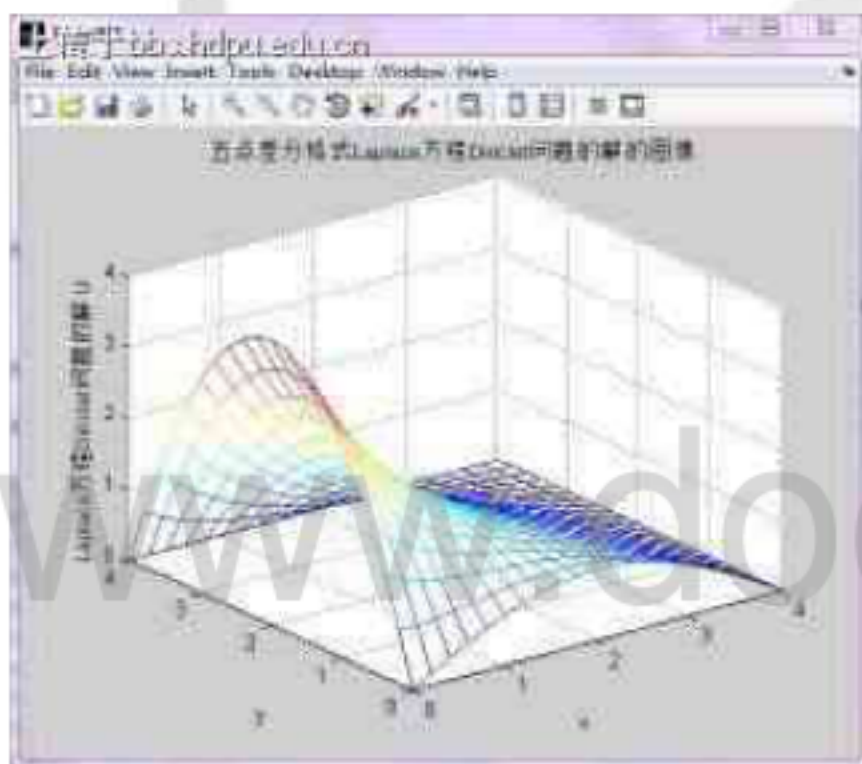
        U(j,i)=X(j-1+(M-1)*(i-2));
    end
end

U=U';
%作出图形
mesh(x,y,U);
title('五点差分格式 Laplace 方程 Diriclet 问题的解的图像')
xlabel('x')
ylabel('y')
zlabel('Laplace 方程 Diriclet 问题的解 U')

return;

```

 正方形区域 Laplace 方程五点差分格式



5、一阶双曲型方程的差分方法

```

function [U x t]=PDEHyperbolic(uX,uT,M,N,C,phi,psi1,psi2,type)
%一阶双曲型方程的差分格式
%[U x t]=PDEHyperbolic(uX,uT,M,N,C,phi,psi1,psi2,type)
%
%方程:u_t+C*u_x=0 0 <= t <= uT, 0 <= x <= uX
%初值条件:u(x,0)=phi(x)
%
%输出参数:U -解矩阵, 第一行表示初值, 第二行表示第 2 个时间层.....

```

```

%      x -横坐标
%      t -纵坐标, 时间
%输入参数:uX -变量 x 的上界
%      uT -变量 t 的上界
%      M -变量 x 的等分区间数
%      N -变量 t 的等分区间数
%      C -系数
%      phi -初值条件函数, 定义为内联函数
%      psi1,psi2 -边值条件函数, 定义为内联函数
%      type -差分格式, 从下列值中选取
%          -type='LaxFriedrichs', 采用 Lax-Friedrichs 差分格式求解
%          -type='CourantIsaacsonRees', 采用 Courant-Isaacson-Rees 差分格式求解
%          -type='LeapFrog', 采用 Leap-Frog (蛙跳) 差分格式求解
%          -type='LaxWendroff', 采用 Lax-Wendroff 差分格式求解
%          -type='CrankNicolson', 采用 Crank-Nicolson 差分格式求解, 此格式需调用追赶法
%          求解三对角线性方程组
%

h=uX/M;%变量 x 的步长
k=uT/N;%变量 t 的步长
r=k/h;%步长比

x=(0:M)*h;
t=(0:N)*k;

U=zeros(M+1,N+1);
%初值条件
for i=1:M+1
    U(i,1)=phi(x(i));
end
%边值条件
for j=1:N+1
    U(1,j)=psi1(t(j));
    U(M+1,j)=psi2(t(j));
    %U(1,j)=NaN;
    %U(M+1,j)=NaN;
end

switch type
    %Lax-Friedrichs 差分格式
    case 'LaxFriedrichs'
        if abs(C*r)>1
            disp('|C*r|>1, Lax-Friedrichs 差分格式不稳定! ')
        end

```

```

%逐层求解
for j=1:N
    for i=2:M
        
$$U(i,j+1)=(U(i+1,j)+U(i-1,j))/2-C*r*(U(i+1,j)-U(i-1,j))/2;$$

    end
end

```

%Courant-Isaacson-Rees 差分格式

```
case 'CourantIsaacsonRees'
```

```

if C<0
    disp('C<0, 采用前差公式')
    if C*r<-1
        disp('Courant-Isaacson-Rees 差分格式不稳定! ')
    end

```

%逐层求解

```

for j=1:N
    for i=2:M
        
$$U(i,j+1)=(1+C*r)*U(i,j)-C*r*U(i+1,j);$$

    end
end

```

```
else
```

```

    disp('C>0, 采用后差公式')
    if C*r>1
        disp('Courant-Isaacson-Rees 差分格式不稳定! ')
    end

```

%逐层求解

```

for j=1:N
    for i=2:M
        
$$U(i,j+1)=C*r*U(i-1,j)+(1-C*r)*U(i,j);$$

    end
end

```

```
end
```

%Leap-Frog（蛙跳）差分格式

```
case 'LeapFrog'
```

```
phi2=input('请输入第二层初值条件函数: psi2=');
```

```

if abs(C*r)>1
    disp('|C*r|>1, Leap-Frog 差分格式不稳定! ')

```

```
end
```

%第二层初值条件

```

for i=1:M+1
    U(i,2)=phi2(x(i));

```

```
end
```

%逐层求解


```

for j=2:N
    for i=2:M
        U(i,j+1)=U(i,j-1)-C*r*(U(i+1,j)-U(i-1,j));
    end
end

%Lax-Wendroff 差分格式
case 'LaxWendroff'
    if abs(C*r)>1
        disp('C*r>1, Lax-Wendroff 差分格式不稳定! ')
    end
    %逐层求解
    for j=1:N
        for i=2:M
            U(i,j+1)=U(i,j)-C*r*(U(i+1,j)-U(i-1,j))/2+C^2*r^2*(U(i+1,j)-2*U(i,j)+U(i-1,j))/2;
        end
    end

%Crank-Nicolson 隐式差分格式，需调用追赶法求解三对角线性方程组的算法
case 'CrankNicolson'
    Diag=zeros(1,M-1);%矩阵的对角线元素
    Low=zeros(1,M-2);%矩阵的下对角线元素
    Up=zeros(1,M-2);%矩阵的上对角线元素
    for i=1:M-2
        Diag(i)=4;
        Low(i)=-r*C;
        Up(i)=r*C;
    end
    Diag(M-1)=4;

    B=zeros(M-1,M-1);
    for i=1:M-2
        B(i,i)=4;
        B(i,i+1)=-r*C;
        B(i+1,i)=r*C;
    end
    B(M-1,M-1)=4;

    %逐层求解，需要使用追赶法（调用函数 EqtsForwardAndBackward）
    for j=1:N
        b1=zeros(M-1,1);
        b1(1)=r*C*(U(1,j+1)+U(1,j))/2;
        b1(M-1)=-r*C*(U(M+1,j+1)+U(M+1,j))/2;
        b=B*U(2:M,j)+b1;

```

```

        U(2:M,j+1)=EqtsForwardAndBackward(Low,Diag,Up,b);
    end

    otherwise
        disp('差分格式类型输入有误！')
        return;
    end

U=U';

%作出图形
mesh(x,t,U);
title([type '格式求解一阶双曲型方程的解的图像']);
xlabel('空间变量 x');
ylabel('时间变量 t');
zlabel('一阶双曲型方程的解 U');

return;

```