基于Matlab的偏微分方程数值解

- Ordinary Differential Eqution(ODE):常微分方程
- Partial Differential Eqution(PDE):偏微分方程
- Ordinary Differential Equtions(ODEs):常微分方程组
- Partial Differential Equtions(PDEs):偏微分方程组

求数值解方法

- 差分方法
- 有限元方法
- MATLAB的pdepe()函数

(求解一般的PDEs,具有较大的通用性，但只支持命令行形式调用)

- MATLAB的PDEtool工具箱

(求解特殊PDE问题,PDEtool有较大的局限性，比如只能求解二阶PDE问题，且不能解决偏微分方程组，但是它提供了GUI界面，从繁杂的编程中解脱出来了，同时还可以通过File->Save As直接生成M代码)

偏微分方程分类

- 椭圆型偏微分方程 (Elliptic)
- 双曲线型偏微分方程 (Hyperbolic)
- 抛物线型偏微分方程 (Parabolic)
- 特征值型偏微分方程 (Elgenmodes)

MATLAB的pdepe函数——pdepe函数的说明

MATLAB软件提供了pdepe函数，该函数不但可以用来求解偏微分方程，也可以用来求解偏微分方程组，函数的调用格式为：

$$sol = pdepe(\mathrm{m}, @\mathrm{pdefun}, @\mathrm{pdeic}, @\mathrm{pdebc}, \mathrm{x}, \mathrm{t})$$

输入的参数中

@pdefun是偏微分方程的描述函数，方程必须具有如下形式

$$c(\mathrm{x},\mathrm{t},\mathrm{u},\frac{\partial u}{\partial x})\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left[x^m f(\mathrm{x},\mathrm{t},\mathrm{u},\frac{\partial u}{\partial x})\right] + s(\mathrm{x},\mathrm{t},\mathrm{u},\frac{\partial u}{\partial x}) \qquad (1)$$

函数pdefun由用户自己编写，函数形式为：

$$[\mathrm{c},\mathrm{f},\mathrm{s}] = \mathrm{pdefun}(\mathrm{x},\mathrm{t},\mathrm{u},\mathrm{du})$$

其输出的c,f,s即为式(1)中的三个已知函数c,f,s,它们也可以是向量值函数，x,t,u与方程(1)中的参数意义相同，du表示的是u对x的一阶导数。

MATLAB的pdepe函数——pdepe函数的说明

@pdebc是偏微分方程的边界条件描述函数，函数必须具有如下形式：

$$p(\mathrm{x},\mathrm{t},\mathrm{u})+\mathrm{q}(\mathrm{x},\mathrm{t},\mathrm{u})\cdot f(\mathrm{x},\mathrm{t},\mathrm{u},\frac{\partial u}{\partial x})=0$$

函数pdebc由用户自己编写，函数形式为：

$$[\mathrm{pa},\mathrm{qa},\mathrm{pb},\mathrm{qb}]=\mathrm{pdebc}(\mathrm{xa},\mathrm{ua},\mathrm{xb},\mathrm{ub},\mathrm{t})$$

其中是xa,xb,ua,ub分别表示变量x,u的下边界和上边界。

@pdeic是偏微分方程的初值条件描述函数，函数必须具有如下形式：

$$u(\mathrm{x},\mathrm{t}_0)=\mathrm{u}_0$$

函数pdeic由用户自己编写，函数形式如下：

$$u0=pdeic(\mathrm{x})$$

函数pdepe中的m即为方程(1)中的m。x,t是偏微分方程的自变量，它们一般是多维向量。

输出的sol是一个三维数组，sol(i,j,k)表示的是自变量分别取x(i),t(j)时u(k)的值。由sol可以直接通过pdeval()某个点的函数值。

MATLAB的pdepe函数——pdepe函数的实例

求解实例：求解偏微分方程组

$$\begin{cases} \dfrac{\partial u_1}{\partial t} = 0.024\dfrac{\partial^2 u_1}{\partial x^2} - F(u_1 - \mathrm{u}_2) \\ \dfrac{\partial u_2}{\partial t} = 0.17\dfrac{\partial^2 u_2}{\partial x^2} + F(u_1 - \mathrm{u}_2) \\ u_1(\mathrm{x},0) = 1, \mathrm{u}_2(\mathrm{x},0) = 0 \\ \left.\dfrac{\partial u_1}{\partial x}\right|_{x=0} = 0, u_1(1,\mathrm{t}) = 1, \mathrm{u}_2(0,\mathrm{t}) = 0, \left.\dfrac{\partial u_2}{\partial x}\right|_{x=1} = 0 \end{cases} \quad (2)$$

其中$\mathrm{F}(\mathrm{x}) = \mathrm{e}^{5.73x} - e^{-11.46x}$.

解：分别编写pdefun函数、pdebc函数、pdeic函数：

MATLAB的pdepe函数——pdepe函数的实例

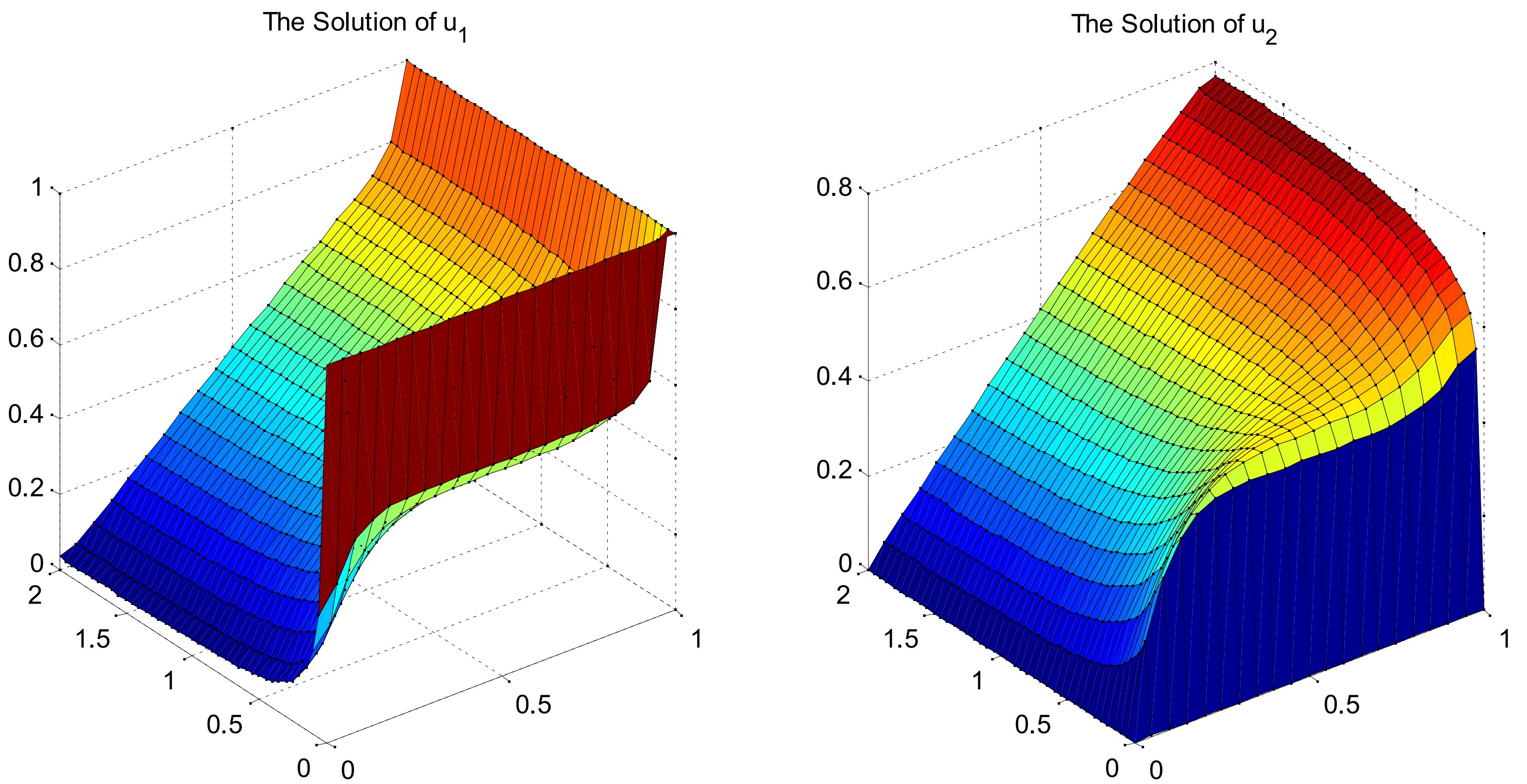
```
%% 目标PDE函数
function [c,f,s]=pdefun (x,t,u,du)
c=[1;1];
f=[0.024*du(1);0.17*du(2)];
temp=u(1)-u(2);
s=[-1;1].*(exp(5.73*temp)-exp(-11.46*temp));

%% 边界条件函数
function [pa,qa,pb,qb]=pdebc(xa,ua,xb,ub,t)
%a表示下边界，b表示上边界
pa=[0;ua(2)];
qa=[1;0];
pb=[ub(1)-1;0];
qb=[0;1];

%% 初值条件函数
function u0=pdeic(x)
u0=[1;0];
```

编写好以上函数之后执行：

MATLAB的pdepe函数——pdepe函数的实例

```
x=0:0.05:1;
t=0:0.05:2;
m=0;
sol=pdepe(m,@pdefun,@pdeic,@pdebc,x,t);
subplot(121)
surf(x,t,sol(:,:,1))
title('The Solution of u_1')
subplot(122)
surf(x,t,sol(:,:,2))
title('The Solution of u_2')
```

MATLAB的pdepe函数——pdepe函数的实例



MATLAB的PDEtool工具箱

MATLAB专门提供了用于求解偏微分方程的工具箱PDEtool，它可用来解各种常见的二阶偏微分方程，但是只能求解特殊二阶的PDE问题，并且不能求解偏微分方程组！

PDE toolbox支持命令行形式求解PDE问题，但是工具箱对四种常见的二阶偏微分要记住那些命令以及调用形式真的很繁冗，还好MATLAB提供了GUI可视交互界面pdetool，在pdetool中可以很方便的求解一个PDE问题，并且可以帮我们直接生成M代码（File->Save As）。

下面先了解下四个典型的二阶PDE，然后介绍pdetool，至于命令行由于太繁冗，故若需要的话就让MATLAB直接生成即可。

1. 典型偏微分方程的描述

（1）双曲型方程

$$d\frac{\partial^2 u}{\partial t^2}-c\Delta u+au=f$$

（2）抛物型方程

$$d\frac{\partial u}{\partial t}-c\Delta u+au=f$$

上两式中，d,c,a,f 必须是常数。

注：Δu表示u的Laplace算子。

（3）椭圆型方程

$$-c\Delta u+au=f$$

式中，c,a,f 为给定的函数或常数。

（4）特征值方程

$$d\frac{\partial^2 u}{\partial t^2}-c\Delta u+au=\lambda du$$

式中，d,c,α,λ必须是常数。

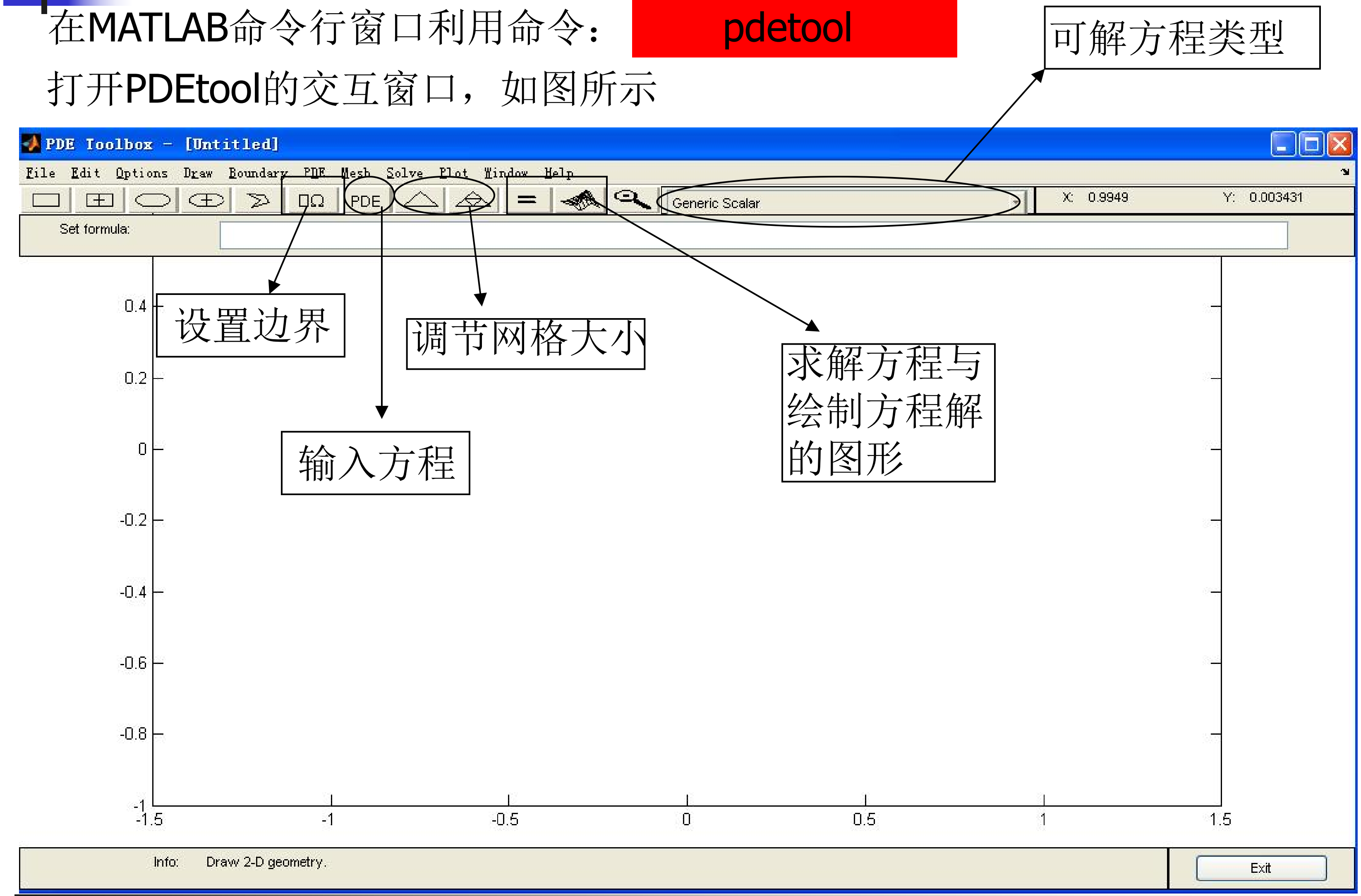
从上面可以看出，四类典型二阶偏微分方程的区别在于u对t 的导数阶次。椭圆形PDE中，c、a、d、和f可以是给定的函数或者常数，但是其他三中必须都是常数。

MATLAB是采用有限元的方法求解各种PDE，MATLAB为我们提供一个pdetool的交互界面，可以求解二元偏微分u(x1,x2)（注意只能求解二元）。方程的参数有a、c、d和f确定，求解域由图形确定，求解域确定好后，需要对求解域进行栅格化（这个是自动）。

MATLAB的PDEtool工具箱

在MATLAB命令行窗口利用命令：pdetool

打开PDEtool的交互窗口，如图所示



MATLAB的PDEtool工具箱——实例

抛物型方程定解问题：

$$\begin{cases} \dfrac{\partial u}{\partial t} - 2\dfrac{\partial^2 u}{\partial x^2} - 2\dfrac{\partial^2 u}{\partial y^2} + 3u = 0 \\ u(0, \mathrm{x}, \mathrm{y}) = \mathrm{xy}(\mathrm{x}-1)(\mathrm{y}-1) \\ u\big|_{\partial\Omega} = 0 \end{cases}$$

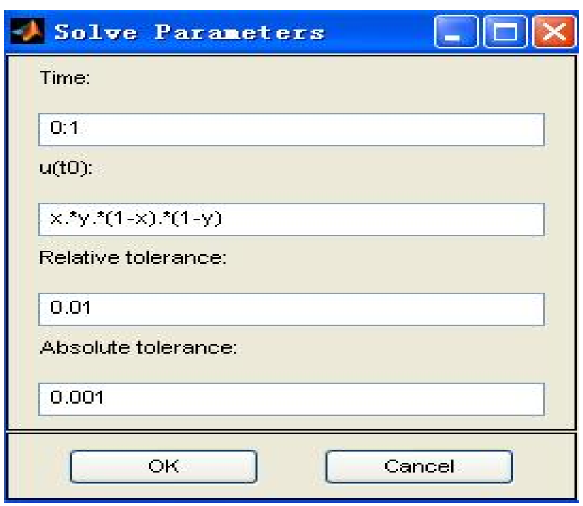
第一步：单击工具栏的“PDE”按钮，在弹出的左窗口左侧选择Parabolic(抛物型)，在右侧输入相应的值，输入完成后单击“OK”按钮。c=2,a=3,f=0,d=1.

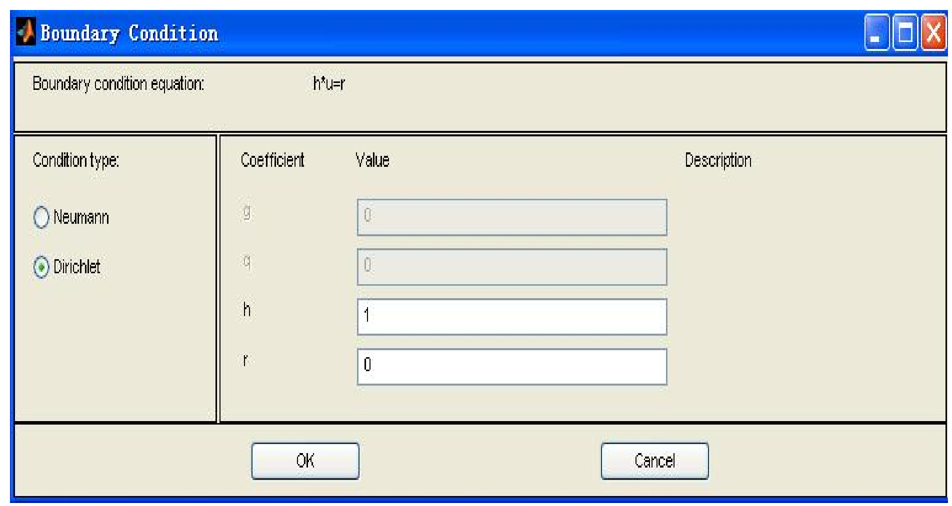
第二步：设置绘制区域，在“Option”菜单中选择“Axis Limits”,打开绘制区域窗口，设置x和y的范围均为[0,1]。选中“Option”里面的“grid”使得绘制区域内画出网格。

MATLAB的PDEtool工具箱——实例

第三步：设置初始条件，选择“Solve”菜单中的“parameters”,在弹出的求解参数窗口中设置初始值。

第四步：设置边界条件，首先选择“Boundary”菜单中的“Boundary Mode”进入边界条件设置模式。利用工具条左边的椭圆，矩形等按钮选择要设置的区域，选择“Boundary”菜单中的“Specify Boundary Conditions...”设置该区域的边界条件，重复这个步骤直到设置好全部的边界条件。本题是矩形上的Dirichlet边界条件。





2. 偏微分方程边界条件的描述

一般在PDE中边界条件包括Dirichlet条件和Neumann条件:

(1) Dirichlet条件

一般描述为 $h(x,t,u,\frac{\partial u}{\partial x})*u\Big|_{\partial\Omega}=r(x,t,u,\frac{\partial u}{\partial x})$，其中$\partial\Omega$表示求解域的边界。

假设在边界上满足该方程，则只需要给出r和h即可，它们可以是常数也可以是给定的函数。

(1) Neumann条件

一般描述为$\left[\frac{\partial}{\partial n}(c\nabla u)+q*u\right]\Big|_{\partial\Omega}=\mathrm{g}$，其中$\frac{\partial u}{\partial n}$表示$u$的法向偏导数。

通过其它的操作调出边界条件设置，注意在这之前一定要使用【区域边界】按钮制定边界。

求解实例：

试求解双曲线型偏微分方程：$\frac{\partial^2 u}{\partial t^2}-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}+2u=10,$

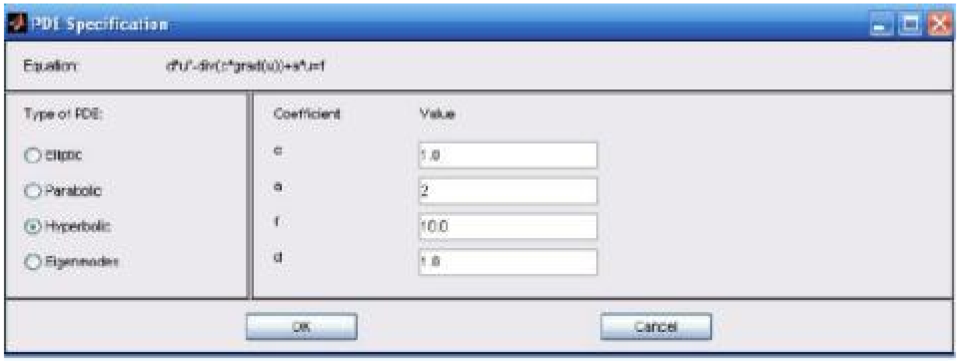
求解域s为： $s1: x^2+y^2\le 9,$

$$s2: \frac{x^2}{4}+\frac{y^2}{16}\le 1,$$

$$s=(s1-s2)\cup(s2-s1).$$

边界条件为：构成求解域的边界值都为5.

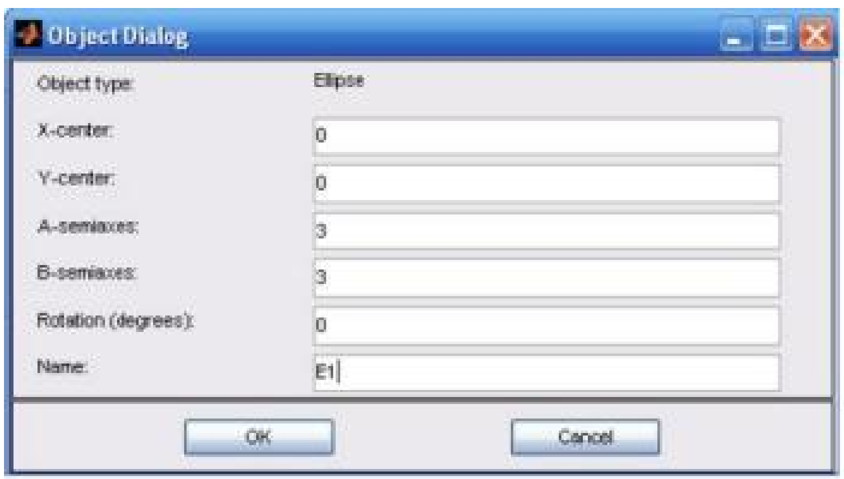
第一步（输入PDE的参数和类型）：单机工具栏的“PDE”按钮，在弹出的左窗口左侧选择Hyperbolic(双曲型)，在右侧输入相应的值，输入完成后单机“OK”按钮。d=1，c=1, a=2, f=10.

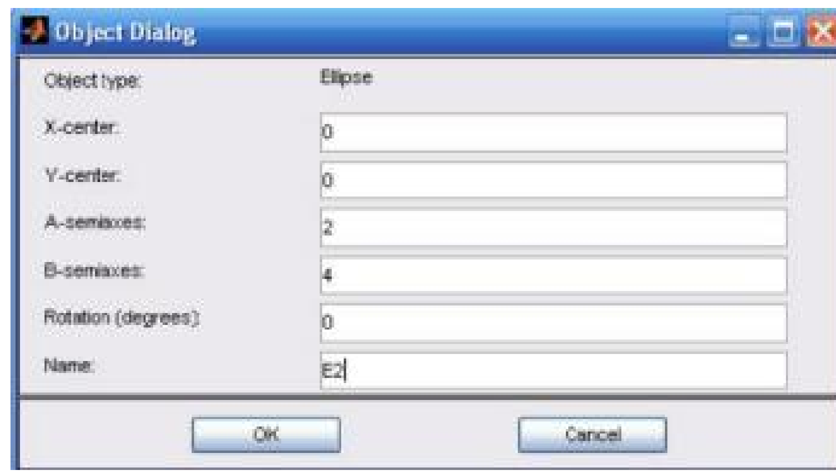


第二步（绘制求解域）：

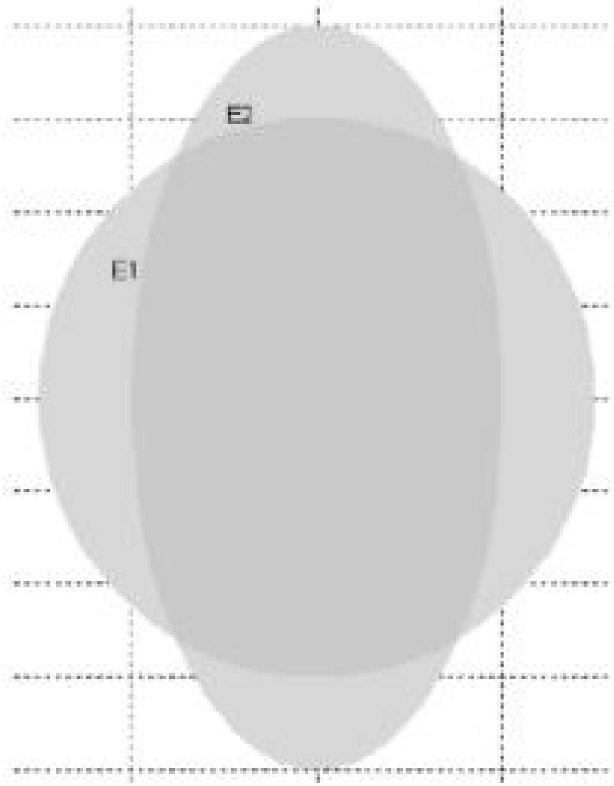
（1）设置绘制区域，在“Option”菜单中选择“Axis Limits”,打开绘制区域窗口，设置x的范围均为[-3,3]，y的范围均为[-5,5]。选中“Option”里面的“grid”使得绘制区域内画出网格。

（2）点击工具栏上的第三个按钮【绘制椭圆】，任意绘制一个椭圆，双击椭圆，设置如下，并重复上面的操作，参数如下





于是得到



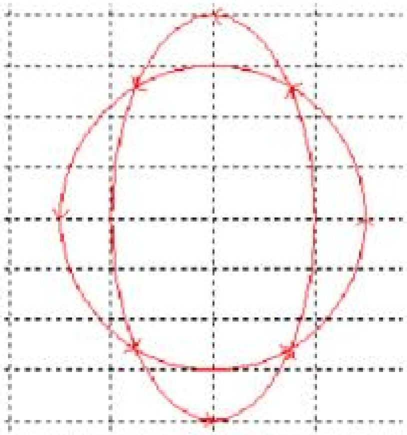
（3）在set formula中如下输入，**“+”表示求并集，“-”表示求差集，注意没有直接求交接的操作符**

第三步：设置初始条件和边界条件：

初值条件可以通过选择“Solve”菜单中的“parameters”,在弹出的求解参数窗口中设置初始值。

边界条件设置如下

（1）点击工具栏的第六个按钮【区域边界】，显示如下

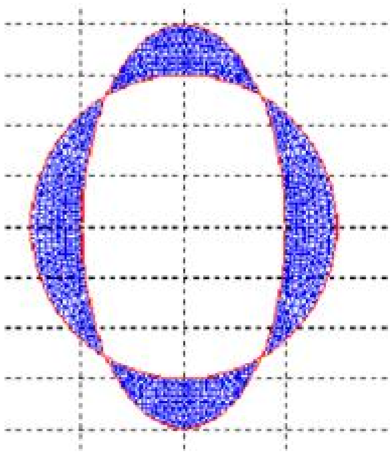


（2）【Boundary】->【Remove All Subdomain Borders】移除所有子域的边界，将得到所有子域合并成一个求解域

（3）【Boundary】->【Secify Boundary Conditions...】设置边界如下，注意我们这里只有Dirichlet条件

第四步（生成使用有限元方法求解方程所需的栅格）：

点击工具栏的第8/9个按钮，对求解域生成栅格，多次点击可以在原来基础上继续细化栅格，直到自己满意为止，当然可以通过【Mesh】主菜单进行精确控制

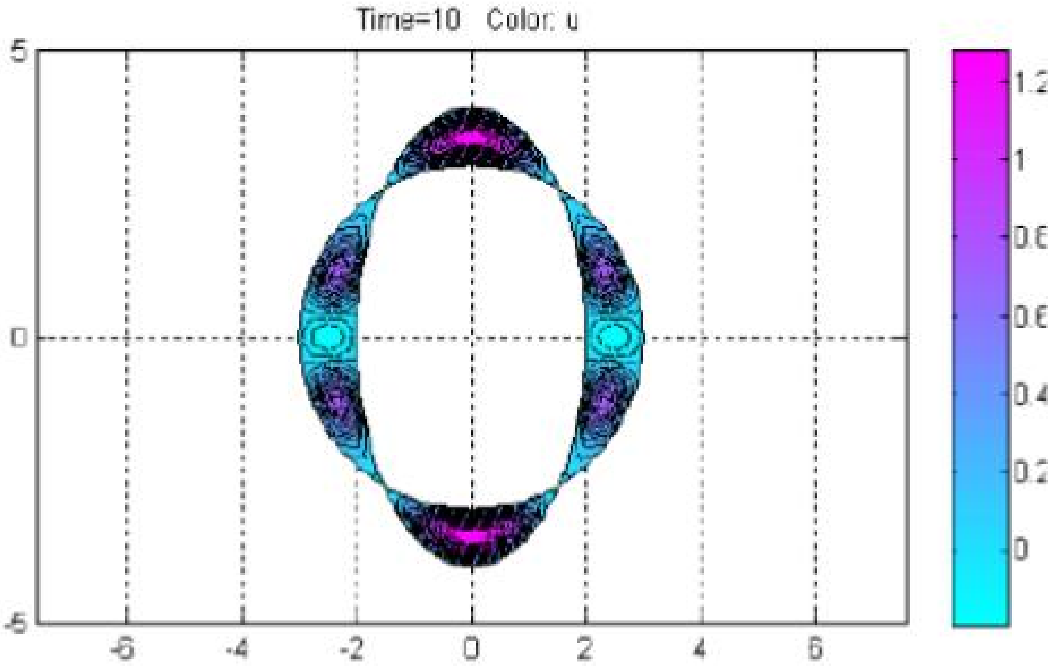


第五步（求解方程）：

点击工具栏的第10个按钮"="【求解方程】

第六步（求解结果绘图）：

点击工具栏的第11个按钮【绘制图形】，里面的选项很丰富，可以绘制等高线等好多，甚至播放动画



动画播放设置：

（1）【Solve】->【Parameters】设置合适的时间向量Time;

（2）【Plot】->【Parameters】选中【Animation】，点击后面的【Options】，设置播放速度和次数，比如6fps表示每秒6帧;

（3）【Plot】->【Export Movie...】，输入动画保存的变量名，比如M;

（4）在Command Windows中直接输入movie（M）即可播放;

（5）在Command Windows中使用movie2avi（M,’demo.avi’）即可播放。

MATLAB的PDEtool工具箱——步骤

- 1.pdetool界面
- 2.选定求解微分方程类型（双曲线、抛物线、椭圆、特殊值型）并设定参数
- 3.绘制求解区域
- 4.边界条件和初值条件(Dirichlet和Neumann)
- 5.生成网格
- 6.求解方程并绘制图形

偏微分的MATLAB数值解法

- 方法总结：
- 1.pdepe调用简单，但计算功能稍弱
- 2.pdetool使用方便，但限于四种方程类型
- 3.程序编写较为繁琐

总结

以上主要介绍了利用MATLAB计算偏微分方程的数值解，可主要分为利用MATLAB自带功能求解和自行编写相关代码求解。

利用MATLAB自带功能求解时，要根据方程的形式，满足要求的方程(组)才能利用函数调用及工具箱求解。这种方法简便但局限性大，许多实际应用的偏微分方程不能满足要求格式。

自行编写相关代码求解包括了多种方法，可查阅上述提到的差分方法、有限元方法，当然还有其他方法。自行编写相关代码求解偏微分方程数值解在工程计算是常用的。因为偏微分方程种类多，自己编写代码可调性强。这里也要根据不同的工程问题以及这些问题对求解精度的要求选择不同的方法。

习题：用pdepe函数实现

应用实例：

$$
\begin{cases}
u_t = 400 u_{xx} \\
u(x,t)\big|_{x=0} = u(x,t)\big|_{x=40} = 0 \\
u(x,t)\big|_{t=0} = \varphi(x)
\end{cases}
$$

$$
\varphi(x) = \begin{cases}
0(x < 10, x > 30) \\
1(10 < x < 30)
\end{cases}
$$

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^{m}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

@pdefun(函数格式描述):

[c,f,s]=pdefun(x,t,u,du)

$$p(x,t,u)+q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0$$

@pdebc（边界条件描述）：

[pa,qa,pb,qb]=pdebc(x,t,u,du)

a:左边界，b:右边界

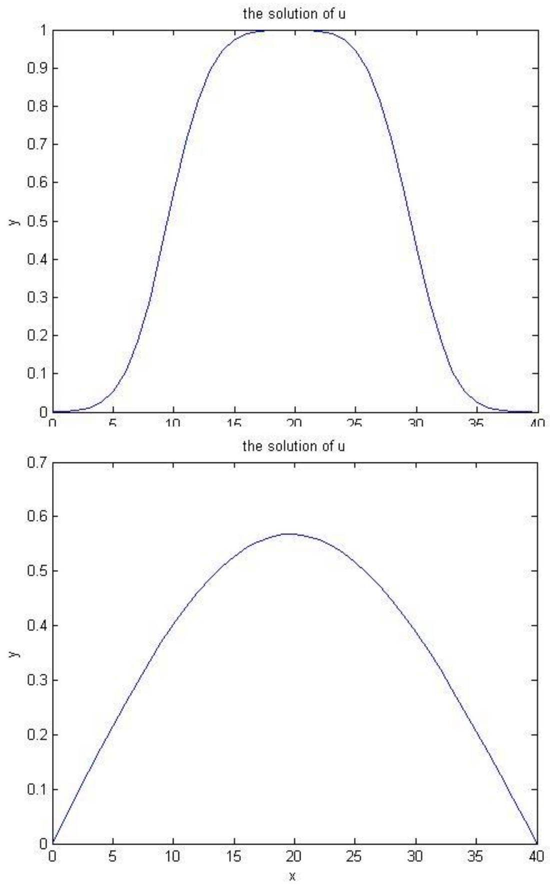
- @pdeic（初值条件描述）：
- 形式：u（x，t0）=u0
- u0=pdeic（x）
- m，x，t对应函数式中参数

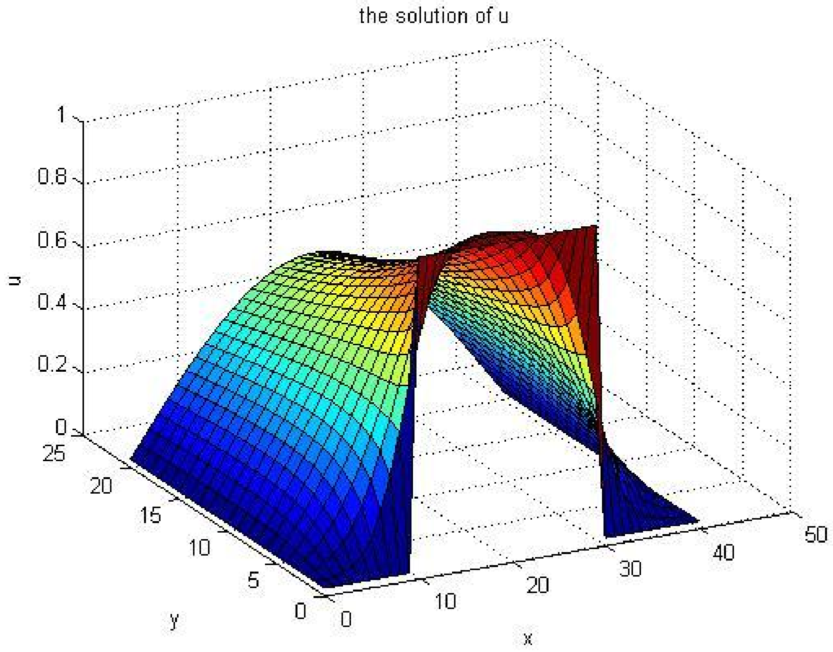
习题解答：

- @pdefun：
- function [c,f,s]=pdefun(x,t,u,du)
- c=1;
- f=400*du;
- s=0;
- @pdebc：
- function [pa,qa,pb,qb]=pdebc(xa,ua,xb,ub,t)
- pa=ua;
- qa=0;
- pb=ub;
- qb=0;

```
@pdeic：
function u0=pdeic(x)
if x<10
    u0=0;
elseif x<30
    u0=1;
else
    u0=0;
end
```

```
x=0:1:40;
t=0:0.01:0.2;
m=0;
sol=pdepe(m,@pdefun,@pdeic,@pdebc,x,t);
b=sol(20,:);
plot(x,b);
title('the solution of u')
xlabel('x')
ylabel('y')
zlabel('u')
```


the solution of u
y
the solution of u
y
x


the solution of u
u
y
x