2　MATLAB 编程

课前学习任务

（1）复习微积分知识，包括微分、积分、微分方程的基本概念等。

（2）通过查阅资料或利用 MATLAB help 功能等通过编程求微分方程 $xy' + y\text{-}e^x = 0$ 在初始条件 $y(1) = 2e$ 下的特解，并绘制所求特解函数的图形（图形中 x 坐标范围-10 到 10）。

2.1数值微分和积分

积分和微分的问题大家并不陌生，在微积分里面我们已经学习过了。微积分课程里面主要给大家讲述的是采用解析的方法求解这类问题，今天我们主要讲述如何利用 MATLAB 采用数值的方法进行求解。

2.1.1　数值微分

微分和导数密切相关，MATLAB 与微分相关的函数主要有三个，如表 1 所示。其中 diff 用于差分和近似导数求解，gradient 用于数值梯度（导数）的求解，del2 用于离散拉普拉斯算子的求解。关于导数和拉普拉斯算子的概念，同学们应该都学习过，我们就不在赘述。下面我们对 3 个函数的用法做简要介绍。

表 1 有限差分导数

diff	差分和近似导数
gradient	数值梯度
del2	离散拉普拉斯算子

diff 主要用于求差分和近似导数，简要说明如下：

（1）当已知函数表达式时，可用 diff(y, n)求解函数 y 的 n 阶导数。要使用 diff 求解函数导数，需要定义符号变量，具体示例如下：

```
syms x                %定义符号变量
y = x^2+cos(x);       %函数表达式
diff1_y = diff(y,1);  %一阶导数
diff2_y = diff(y,2);  %二阶导数
```

上面例子是求解函数一阶导数和二阶导数的简单示例，大家可自行在 MATLAB 中运行体会一下。

（2）求数组差分，引用格式如下：

Y = diff(X)

Y = diff(X,n)

Y = diff(X,n,dim)

具体说明如下：

Y = diff(X) 计算沿大小不等于 1 的第一个数组维度的 X 相邻元素之间的差分：

如果 X 是长度为 m 的向量，则 Y = diff(X) 返回长度为 m-1 的向量。Y 的元素是 X 相邻元素之间的差分。

Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

如果 X 是不为空的非向量 p×m 矩阵，则 Y = diff(X) 返回大小为 (p-1)×m 的矩阵，其元素是 X 的行之间的差分。

Y = [X(2,:)-X(1,:); X(3,:)-X(2,:); ... X(p,:)-X(p-1,:)]

如果 X 是 0×0 的空矩阵，则 Y = diff(X) 返回 0×0 的空矩阵。

Y = diff(X,n) 通过递归应用 diff(X) 运算符 n 次来计算第 n 个差分。在实际操作中，这表示 diff(X,2) 与 diff(diff(X)) 相同。

Y = diff(X,n,dim) 是沿 dim 指定的维计算的第 n 个差分。dim 输入是一个正整数标量。

以一个二维 p x m 输入数组 A 为例：

- diff(A,1,1) 会对 A 的列中的连续元素进行处理，然后返回 (p-1)xm 的差分矩阵。
- diff(A,1,2) 会对 A 的行中的连续元素进行处理，然后返回 px(m-1) 的差分矩阵。

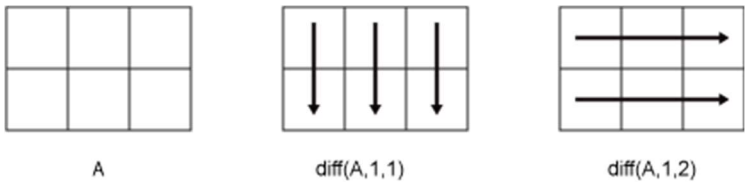


图 1 关于指定维 dim 的说明

示例：

计算差分函数 f=sin(X)导数的近似值：

```
h = 0.001;       % step size
```

```
X = -pi:h:pi;    % domain
f = sin(X);      % range
Y = diff(f)/h;   % first derivative
Z = diff(Y)/h;   % second derivative
plot(X(:,1:length(Y)),Y,'r',X,f,'b', X(:,1:length(Z)),Z,'k')
```

请各位同学自行运行上述程序以熟悉 diff 的用法。

关于 diff 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/diff.html#btwmxq8-10

gradient 主要用于求梯度，简要说明如下：

算法：

gradient 计算内部数据点的中心差分。例如，考虑一个包含单位间距数据的矩阵 A，它具有水平梯度 G = gradient(A)。内部梯度值 G(:,j) 为：

G(:,j) = 0.5*(A(:,j+1) - A(:,j-1));

下标 j 在 2 和 N-1 之间变化，其中 N = size(A,2)。

gradient 使用单侧差分计算沿矩阵边的值：

G(:,1) = A(:,2) - A(:,1);

G(:,N) = A(:,N) - A(:,N-1);

当指定点间距时，gradient 会对差分进行相应的缩放。如果指定了两个或更多个输出，该函数还可以按类似方式计算沿其他维度的差分。**与 diff 函数不同，gradient 返回与输入具有相同数量元素的数组。**

语法：

FX = gradient(F)

[FX,FY] = gradient(F)

[FX,FY,FZ,...,FN] = gradient(F)

[___] = gradient(F,h)

[___] = gradient(F,hx,hy,...,hN)

说明：

FX = gradient(F) 返回向量 F 的一维数值梯度。输出 FX 对应于 ∂F/∂x，即 x（水平）方向上的差分。点之间的间距假定为 1。

[FX,FY] = gradient(F) 返回矩阵 F 的二维数值梯度的 x 和 y 分量。附加输出 FY 对应于 ∂F/∂y，即 y（垂直）方向上的差分。每个方向上的点之间的间距假定为 1。

[FX,FY,FZ,...,FN] = gradient(F) 返回 F 的数值梯度的 N 个分量，其中 F 是一个 N 维数组。

[___] = gradient(F,h) 使用 h 作为每个方向上的点之间的均匀间距。可以指定上述语法中的任何输出参数。

[___] = gradient(F,hx,hy,...,hN) 为 F 的每个维度上的间距指定 N 个间距参数。

示例：

计算 $xe^{-x^2-y^2}$ 在网格上的二维梯度，并绘制向量场的等高线图：

```
x = -2:0.2:2;
y = x';
z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z);

figure
contour(x,y,z)      %绘制等高线
hold on
quiver(x,y,px,py)  %绘制向量图
hold off
```

请各位同学自行运行上述程序以熟悉 gradient 的用法。

关于 gradient 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/gradient.html

delt2 主要用于求离散拉普拉斯算子，简要说明如下：

算法：

如果输入 U 是一个矩阵，则 L 的内部点通过取 U 中的点与其四个相邻点的平均值之间的差值找到：

$$L_{ij} = \left[\frac{\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right)}{4} - u_{i,j} \right].$$

然后，del2 会通过线性外插第二个差分来从内部延伸计算 L 边界上的值。此公式会针对多维 U 进行扩展。

拉普拉斯微分运算子与 delt2 的关系：

如果矩阵 U 是在方形网格的点位置计算的函数 U(x,y)，则 4*del2(U) 是应用到 U 的拉普拉斯微分运算子的有限微分近似值，关系如下：

$$L = \frac{\Delta U}{4} = \frac{1}{4}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right).$$

对于带有更多变量的函数 U(x,y,z,...)，离散拉普拉斯算子 del2(U) 会计算每个维度中的第二个导数，

$$L = \frac{\Delta U}{2N} = \frac{1}{2N}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + \ldots\right),$$

其中 N 是 U 中的维度数，且 N≥2。

语法：

L = del2(U)

L = del2(U,h)

L = del2(U,hx,hy,...,hN)

说明：

L = del2(U) 在所有点之间使用默认间距 h = 1，返回应用于 U 的拉普拉斯微分运算子的离散近似值。

L = del2(U,h) 用于在 U 的所有维度中的点之间指定均匀的标量间距 h。

L = del2(U,hx,hy,...,hN) 用于在 U 的每个维度中的点之间指定间距 hx,hy,...,hN。将每个间距输入指定为坐标的标量或向量。间距输入数必须等于 U 中的维度数。

第一个间距值 hx 指定点的 x 间距（标量）或 x 坐标（向量）。如果是向量，则其长度必须等于 size(U,2)。

第二个间距值 hy 指定点的 y 间距（标量）或 y 坐标（向量）。如果是向量，则其长度必须等于 size(U,1)。

所有其他间距值指定 U 中对应维度的各点的间距（标量）或坐标（向量）。对于 n > 2 的情况，如果第 n 个间距输入是向量，则其长度必须等于 size(U,n)。

示例：

计算并绘制多变量函数的离散拉普拉斯算子：

```
[x,y] = meshgrid(-5:0.25:5,-5:0.25:5);
U = 1/3.*(x.^4+y.^4);
h = 0.25;
L = 4*del2(U,h);
figure
surf(x,y,L)
grid on
h = title('Plot of $\Delta U(x,y) =4x^2+4y^2$');
set(h,'Interpreter','latex')
xlabel('x')
ylabel('y')
zlabel('z')
view(35,14)
```

备注：MATLAB 2019b 版本写标题（title）的两行代码可以直接用一行代码

```
title('Plot of $\Delta U(x,y) = 4x^2+4y^2$','Interpreter','latex')
```

替代

请各位同学自行运行上述程序以熟悉 delt2 的用法。

关于 delt2 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/del2.html

2.1.2 数值积分

通过微积分的学习我们知道积分分为不定积分和定积分。有些积分问题可以直接通过解析的方法进行求解，但有很多积分问题是没有解析解的或者解析解的求解过程非常复杂，这个时候我们就需要采用数值积分了。对于不定积分，可以用 matlab 中的 int 函数求解，由于本小节主要介绍数值积分，int 的具体用法不做详细介绍，有兴趣的同学可以通过 MATLAB 的 help 自行学习。MATLAB 与数值积分相关的函数主要分为两类，包括求函数表达式的积分和求数值数据的积分，具体如表 2 所示，其中前 5 个已知函数表达式的情况下求积分，最后一个为已知离散数据的情况下求积分。需要说明的是 MATALAB 早期版本（比如 2012 之前的版本）和 quad 相关的函数较多，目前新版本中已基本被 integral 相关函数替代，大家除了数值积分问题的时候使用表 2 中的函数即可。下面我们将对表 2 中的函数做简要介绍。

表 2 MATLAB 相关数值积分函数

integral	数值积分
integral2	对二重积分进行数值计算
integral3	对三重积分进行数值计算
quadgk	以自适应高斯-勒让德积分法计算数值积分
quad2d	计算二重数值积分 - tiled 方法
trapz	梯形数值积分

integral 和 quadgk 主要用于数值积分，两者调用方法类似，下面以 integral 为例进行简要说明：

语法：

q = integral(fun,xmin,xmax)

q = integral(fun,xmin,xmax,Name,Value)

说明：

q = integral(fun,xmin,xmax) 使用全局自适应积分和默认误差容限在 xmin 至 xmax 间以数值形式为函数 fun 求积分。

q = integral(fun,xmin,xmax,Name,Value) 指定具有一个或多个 Name,Value 对组参数的其他选项，可以指定绝对误差容限、相对误差容限、数组值函数标志、积分路点等。

示例：

创建带有一个参数 c 的函数 $f(x)=1/(x^3-2x-c)$。在 c=5 时，计算从 x=0 至 x=2 的积分。

```
fun = @(x,c) 1./(x.^3-2*x-c);
q = integral(@(x)fun(x,5),0,2)
```

请各位同学自行运行上述程序以熟悉 integral 的用法。

关于 integral 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/integral.html#btdd9y9

quadgk 和 integral 函数调用方法类似，实现算法上略有区别，quadgk 基于高斯-勒让德对组（第 15 和第 7 阶次公式）实现自适应积分。关于 quadgk 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/quadgk.html

integral2 和 quad2d 主要用于二重积分，两者调用方法类似，下面以 integral2 为例进行简要说明：

语法：

```
q = integral2(fun,xmin,xmax,ymin,ymax)
q = integral2(fun,xmin,xmax,ymin,ymax,Name,Value)
```

说明：

q = integral2(fun,xmin,xmax,ymin,ymax) 在平面区域 xmin ≤ x ≤ xmax 和 ymin(x) ≤ y ≤ ymax(x) 上逼近函数 z = fun(x,y) 的积分。

q = integral2(fun,xmin,xmax,ymin,ymax,Name,Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。，可以指定绝对误差容限、相对误差容限、积分法等。

示例：

使用参数 a=3 和 b=5 创建匿名的参数化函数 $f(x,y)=ax^2+by^2$。

```
a = 3;
b = 5;
fun = @(x,y) a*x.^2 + b*y.^2;          %定义函数表达式
format long
q = integral2(fun,0,5,-5,0,'Method','iterated',...      %...表示换行
'AbsTol',0,'RelTol',1e-10)
```

请各位同学自行运行上述程序以熟悉 integral2 的用法。

关于 integral2 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/integral2.html

quad2d 和 integral2 函数调用方法类似，quad2d 基于 tiled 方法实现二重积分。关于 quad2d 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/quad2d.html

integral3 主要用于三重积分简要说明如下：

语法

```
q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax)
q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name,Value)
```

说明

q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax) 在区域 xmin ≤ x ≤ xmax、ymin(x) ≤ y ≤ ymax(x) 和 zmin(x,y) ≤ z ≤ zmax(x,y) 逼近函数 z=fun(x,y,z) 的积分。

q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name,Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。

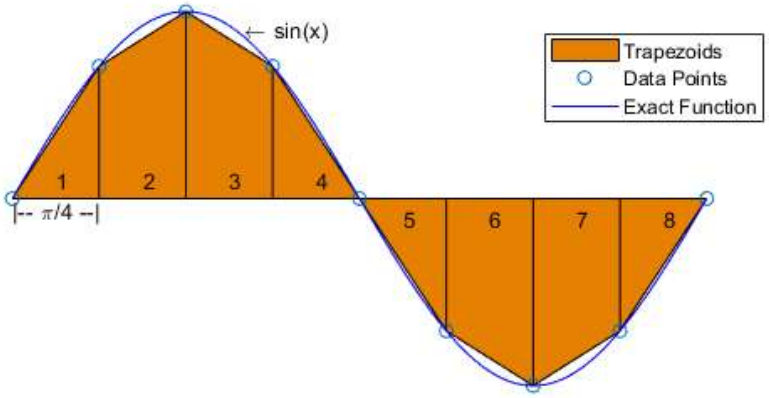
关于 integral3 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/integral3.html

trapz 在已知离散数据下采用梯形法进行数值积分求解，简要说明如下：

梯形法：

通过将一个区域分为包含多个更容易计算的区域的梯形，该方法对区间内的积分计算近似值。例如，下面是使用八个均匀间隔的梯形对正弦函数求梯形积分：



对于具有 N+1 个均匀分布的点的积分，近似值为：

$$\int_a^b f(x)dx \approx \frac{b-a}{2N}\sum_{n=1}^{N}\left(f(x_n)+f(x_{n+1})\right)$$

$$=\frac{b-a}{2N}\left[f(x_1)+2f(x_2)+\ldots+2f(x_N)+f(x_{N+1})\right],$$

其中，各点之间的间距等于标量值(*b*-*a*)/*N*。默认情况下，MATLAB®使用的间距为 1。如果各 N+1 点之间的间距不是常量，则公式可以推及到

$$\int_a^b f(x)dx \approx \frac{1}{2}\sum_{n=1}^{N}(x_{n+1}-x_n)\left[f(x_n)+f(x_{n+1})\right],$$

其中，$a=x_1<x_2<\ldots<x_N<x_{N+1}=b$ 和$(x_{n+1}-x_n)$是每对连续点之间的间距。

语法：

Q = trapz(Y)
Q = trapz(X,Y)
Q = trapz(___,dim)

说明：

Q = trapz(Y) 通过梯形法计算 Y 的近似积分（采用单位间距）。Y 的大小确定求积分所沿用的维度：

如果 Y 为向量，则 trapz(Y) 是 Y 的近似积分。

如果 Y 为矩阵，则 trapz(Y) 对每列求积分并返回积分值的行向量。

如果 Y 为多维数组，则 trapz(Y) 对其大小不等于 1 的第一个维度求积分。该维度的大小变为 1，而其他维度的大小保持不变。

Q = trapz(X,Y) 根据 X 指定的坐标或标量间距对 Y 进行积分。

如果 X 是坐标向量，则 length(X) 必须等于 Y 的大小不等于 1 的第一个维度的大小。

如果 X 是标量间距，则 trapz(X,Y) 等于 X*trapz(Y)。

Q = trapz(___,dim) 使用以前的任何语法沿维度 dim 求积分。必须指定 Y，也可以指定 X。如果指定 X，则它可以是长度等于 size(Y,dim) 的标量或向量。例如，如果 Y 为矩阵，则 trapz(X,Y,2) 对 Y 的每行求积分。

示例：

对具有非均匀数据间距的矩阵的行求积分。

```
X = [1 2.5 7 10];
Y = [5.2   7.7   9.6   13.2;
     4.8   7.0  10.5   14.5;
     4.9   6.5  10.2   13.8];
Q1 = trapz(X,Y,2)
```

请各位同学自行运行上述程序以熟悉 trapz 的用法。

关于 trapz 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/matlab/ref/trapz.html#buaijhw-1

2.2 微分方程

微分方程，是指含有未知函数及其导数的关系式。这里我们着重介绍 3 类偏微分方程的求解方法：（1）有解析解的微分方程；（2）常微分方程（ODE）的数值求解方法；（3）二元偏微分方程（PDE）的数值求解方法

2.2.1 微分方程解析解求解

对于通常的微分方程，可尝试采用 dsolve 函数求解其解析解。MATALB 中可以用 dsolve 进行求解。dsolve 函数的简要说明如下：

语法：

S = dsolve(eqn)
S = dsolve(eqn,cond)
S = dsolve(___,Name,Value)
[y1,...,yN] = dsolve(___)

说明：

S = dsolve(eqn)用于求解微分方程 eqn，这里 eqn 是一个符号方程。使用 diff 和==表示微分方程。例如 diff(y,x) == y 表示方程 dy/dx = y.对于微分方程组，eqn 可以看成是一个向量，向量的每一个值为一个方程。

S = dsolve(eqn,cond) 用于求解带有初始或边界条件 cond 的微分方程（组）eqn

S = dsolve(___,Name,Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。

[y1,...,yN] = dsolve(___) 用于把求解结果赋给变量 y1,...,yN。

示例：

求解微分方程组：

$$\frac{dy}{dt} = z$$
$$\frac{dz}{dt} = -y.$$

程序如下：

```
syms y(t) z(t)
eqns = [diff(y,t) == z, diff(z,t) == -y];
S = dsolve(eqns)
```

上述程序运行结果如下：

```
S = struct with fields:
    z: [1x1 sym]
    y: [1x1 sym]
```

S 为一个结构体数组，结构体元素调用命令如下：

```
ySol(t) = S.y
zSol(t) = S.z
```

使用上述命令即可看到微分方程组求解结果。

请各位同学自行运行上述程序以熟悉 dsolve 的用法。

关于 **dsolve** 的具体用法可参见下列链接：

https://ww2.mathworks.cn/help/symbolic/dsolve.html

2.2.2 常微分方程的数值求解

常微分方程 (ordinary differential equation，ODE) 包含与一个自变量 t（通常称为时间）相关的因变量 y 的一个或多个导数。此处用于表示 y 相对于 t 的导数的表示法对于一阶导数为 y′，对于二阶导数为 y″，依此类推。ODE 的阶数等于 y 在方程中出现的最高阶导数。

对于常微分方程（组），当解析解无法求解的时候，此时需要利用 ODE 家族求解器进行求。MATLAB® 中的常微分方程 (ODE) 求解器可对具有各种属性的初始值问题进行求解。求解器可以处理刚性或非刚性问题、具有质量矩阵的问题、微分代数方程 (DAE) 或完全隐式问题。

MATLAB ODE 求解器可解一阶方程（组）：

$$\begin{pmatrix} y'1 \\ y'2 \\ \vdots \\ y'n \end{pmatrix} = \begin{pmatrix} f1\left(t, y1, y2, \ldots, yn\right) \\ f2\left(t, y1, y2, \ldots, yn\right) \\ \vdots \\ fn\left(t, y1, y2, \ldots, yn\right) \end{pmatrix},$$

原则上方程的数量仅受内存的限制。当方程数量较多时，可通过编写该方程组代码的函数将返回一个向量，其中包含 n 个元素，对应于 $y'1$, $y'2$, … , $y'n$ 值。例如，考虑以下包含两个方程的方程组

$$\begin{cases} y'1 = y2 \\ y'2 = y1\,y2 - 2. \end{cases}$$

用于编写该方程组代码的函数为

```
function dy = myODE(t,y)
dy(1) = y(2);
dy(2) = y(1)*y(2)-2;
```

对于高阶 ODE，需要转换成一阶 ODE 后进行求解，转换方法如下：

$$\begin{aligned} y1 &= y \\ y2 &= y' \\ y3 &= y'' \\ &\vdots \\ yn &= y^{(n-1)}. \end{aligned}$$

对于 ODE 类问题，MATLAB 提供了一系列的求解器，如表 3 所示。从表 3 可以看出，ODE 问题主要是非刚性、刚性和完全隐性三种。大部分情况下，OED 问题是非刚性问题，**首选 ode45 进行求解。**但对于精度要求更宽松或更严格的问题而言，ode23 和 ode113 可能比 ode45 更加高效。

一些 ODE 问题具有较高的计算*刚度*或难度。对于一些 ODE 问题，求解器采用的步长被强制缩小为与积分区间相比过小的级别，甚至在解曲线平滑的区域亦如此。这些步长可能过小，以至于遍历很短的时间区间都可能需要数百万次计算。这可能导致求解器积分失败，即使积分成功也需要花费很长时间。导致 ODE 求解器出现此行为的方程称为*刚性*方程。刚性 ODE 造成的问题是，显式求解器（例如 ode45）获取解的速度慢得令人无法忍受。此时需要采用刚性求解器。

表 3 MATLAB ODE 求解器

求解器	问题类型	精度	何时使用
ode45	非刚性	中	大多数情况下，应当首先尝试求解器 ode45。
ode23		低	对于容差较宽松的问题或在刚度适中的情况下，ode23 可能比 ode45 更加高效。
ode113		低到高	对于具有严格误差容限的问题或在 ODE 函数需要大量计算开销的情况下，ode113 可能比 ode45 更加高效。
ode15s	刚性	低到中	若 ode45 失败或效率低下并且怀疑面临刚性问题，请尝试 ode15s。此外，当解算微分代数方程 (DAE) 时，请使用 ode15s。
ode23s		低	对于误差容限较宽松的问题，ode23s 可能比 ode15s 更加高效。它可以解算一些刚性问题，而使用 ode15s 解算这些问题的效率不高。 ode23s 会在每一步计算 Jacobian，因此通过 odeset 提供 Jacobian 有利于最大限度地提高效率和精度。 如果存在质量矩阵，则它必须为常量矩阵。
ode23t		低	对于仅仅是刚度适中的问题，并且需要没有数值阻尼的解，请使用 ode23t。 ode23t 可解算微分代数方程 (DAE)。
ode23tb		低	与 ode23s 一样，对于误差容限较宽松的问题，ode23tb 求解器可能比 ode15s 更加高效。
ode15i	完全隐式	低	对于完全隐式问题 f(t,y,y') = 0 和微分指数为 1 的微分代数方程 (DAE)，请使用 ode15i。

虽然如此，可能有些同学还是有疑问，如何区分“刚性”和“非刚性”，其实不用太纠结。对于一个 ODE 问题，我们可以首先尝试采用 ode45 等非刚性求解器进行求解。当观察到非刚性求解器的速度很慢时，可尝试改用 ode15s 等刚性求解器。

关于 ODE 求解器的选择，可参见下列链接：

https://ww2.mathworks.cn/help/matlab/math/choose-an-ode-solver.html

关于非刚性 ODE 求解，可参见下列链接：

https://ww2.mathworks.cn/help/matlab/math/solve-nonstiff-odes.html

关于刚性 ODE 求解，可参见下列链接：

https://ww2.mathworks.cn/help/matlab/math/solve-stiff-odes.html

对于 ODE 系列求解器中每个求解器的具体用法，由于时间关系，我们就不一一介绍了，大家可使用 matlab 的 help 功能进行学习。下面我们结合一个简单的例子对 ODE 求解器的使用进行介绍。

导弹追踪问题。设位于坐标原点的甲舰向位于 x 轴上点 A（1,0）处的乙舰发射导弹，导弹头始终对准乙舰。如果乙舰以最大速度 v_0（常数）沿平行于 y 轴的直线行驶，导弹的速度为 $5v_0$，求导弹的运行的曲线方程，以及乙舰行驶多远时，导弹将击中它？

上述问题中，导弹头始终对准乙舰，因此任意时刻导弹速度的方向始终和导弹位置指向乙舰位置的方向相同。根据这个条件即可列写导弹运动的曲线方程，具体过程如下：

记导弹的速度为 w，乙舰的速率恒为 v_0，设时刻 t 乙舰的坐标为 $(X(t),Y(t))$，导弹的坐标为 $(x(t),y(t))$。当零时刻时，$(X(0),Y(0))=(1,0),(x(0),y(0))=(0,0)$ 建立微分方程模型：

$$\begin{cases}\dfrac{dx}{dt}=\dfrac{w}{\sqrt{(X-x)^2+(Y-y)^2}}(X-x)\\[2ex]\dfrac{dy}{dt}=\dfrac{w}{\sqrt{(X-x)^2+(Y-y)^2}}(Y-y)\end{cases}$$

因为乙舰以速度 v_0 沿直线 $x=1$ 运动，设 $v_0=1$，$w=5$，$X=1$，$Y=t$，因此导弹运动轨迹的参数方程为

$$\begin{cases}\dfrac{dx}{dt}=\dfrac{5}{\sqrt{(1-x)^2+(t-y)^2}}(1-x)\\[2ex]\dfrac{dy}{dt}=\dfrac{5}{\sqrt{(1-x)^2+(t-y)^2}}(t-y)\\x(0)=0,y(0)=0\end{cases}$$

根据微分方程组合初值条件可以采用 ODE 求解器对上述问题进行求解，具体代码如下：

编写方程组代码函数：

```
function dy = eq2(t,y)
dy = zeros(2,1);
dy(1) = 5*(1-y(1))/sqrt((1-y(1))^2+(t-y(2))^2);
dy(2) = 5*(t-y(2))/sqrt((1-y(1))^2+(t-y(2))^2);
```

调用函数采用 ode45 求解器进行求解并绘制运动曲线。

```
t0=0;
tf=0.3;
[t, y] = ode45('eq2', [t0 tf], [0 0]);
Y=t;
X(1:length(Y))=1;
figure;
plot(X,Y,'--','linewidth',2);      %乙舰的运动曲线
hold on
plot(y(:,1),y(:,2),'*');           %导弹的运动曲线
```

2.2.3 偏微分方程的数值求解

当微分方程方程中不止一个变量时，称为偏微分方程。MATLAB 提供两种方法用于偏微分方程的求解：pdede 函数和 PDE 工具箱，其中 **pdede 函数具有较大的通用性，但只支持命令行调用方式；PDE 工具箱能够求解一些常见的二阶 PDE 问题（特定的 PDE 问题），支持命令行调用（solvepde 函数）和 GUI 界面操作两种方式**。由于 PDE 工具箱使用非常方便，因此当问题属于特定 PDE 范畴时，推荐使用 PDE 工具箱进行求解。

pdepe 使用一个空间变量 x 和时间 t 对抛物型和椭圆型 PDE 方程组求解，格式为：

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

当需要是用 pdede 求解偏微分方程（组）的时候，需要首先将方程写成上式所示的形式。上式中，PDE 适用于 $t_0 \le t \le t_f$ 且 $a \le x \le b$。[a, b] 必须是有限区间。m 可以是 0、1 或 2，分别对应平板、柱状或球面对称性。如果 $m > 0$，则 a 必须大于或等于 0。

关于 pdede 的详细用法请参见下列链接：

https://ww2.mathworks.cn/help/matlab/math/partial-differential-equations.html#bu8tnr1

链接中有很多案例可以帮助同学们快速掌握 pdede 的用法。

PDE 工具箱可以用于求解一些特定的**二阶 PDE 问题**，支持求解的方程类型如下：

$$m\frac{\partial^2 u}{\partial t^2}+d\frac{\partial u}{\partial t}-\nabla\cdot(c\nabla u)+au=f$$

（标量方程）

$$-\nabla\cdot(c\nabla u)+au=\lambda du$$

（本征方程）

$$-\nabla\cdot(c\nabla u)+au=\lambda^2 mu$$

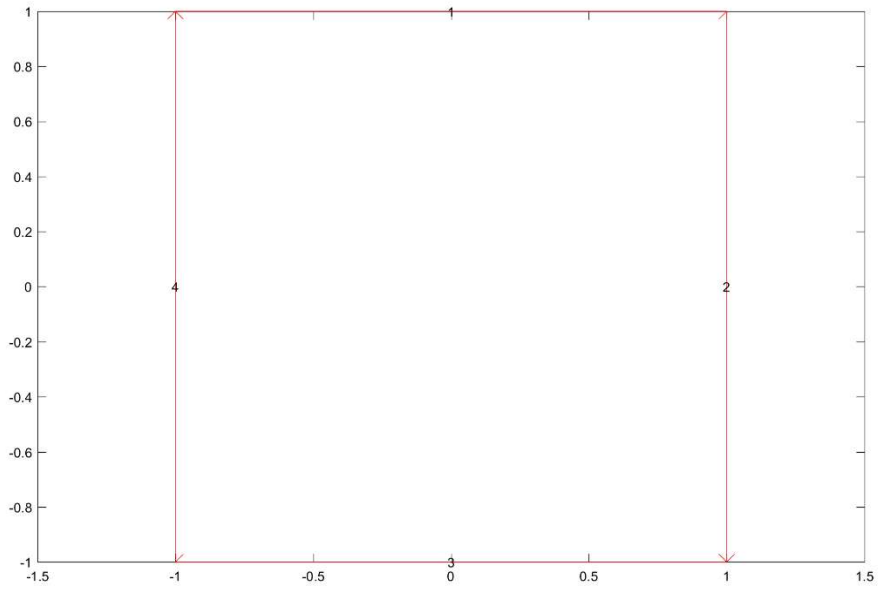
（本征方程，2019 版本支持）

在 MATLAB 界面 APP 功能区里面选择（工具箱界面方式。APP 功能区如果找不到，可能是没有安装，安装下 PDE 即可），也可以在命令行窗口输 pdetool（2017a 版本）或者 pdeModeler（2019b）可以调用工具箱。

下面结合一个简单的例子对 PDE 工具箱的使用做简要介绍。考虑一个简单的二阶波动方程：

$$\frac{\partial^2 u}{\partial t^2}-\nabla\cdot\nabla\mathrm{u}=0$$

对于上述波动方程求解下图边界 1~4 围成区域内 t=30 时 u 的分布。



1 和 3 处的边界条件为$\nabla u=0$；2 和 4 处的边界条件为$u=0$。利用 PDE 工具箱求解过程如下：

步骤 1：打开 Matlab 软件，输入 pdetool（2019b 版本的输入 pdeModeler）并按回车键，弹出界面如图 1 所示；

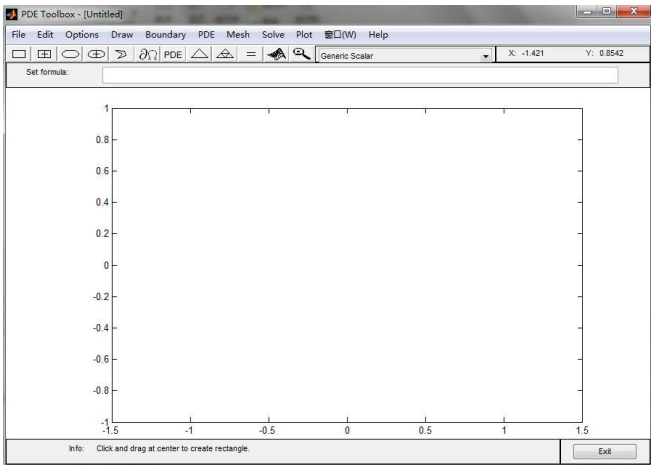


图 1 pdetool 图形界面

步骤 2：工具栏中，点击红框中的按钮，在图中坐标区域按住鼠标左键，同时拖动鼠标画出矩形框，如图 2 所示；或是单击工具栏中的“Draw”，在弹出的选项中选择“Rectangle/square”进行；

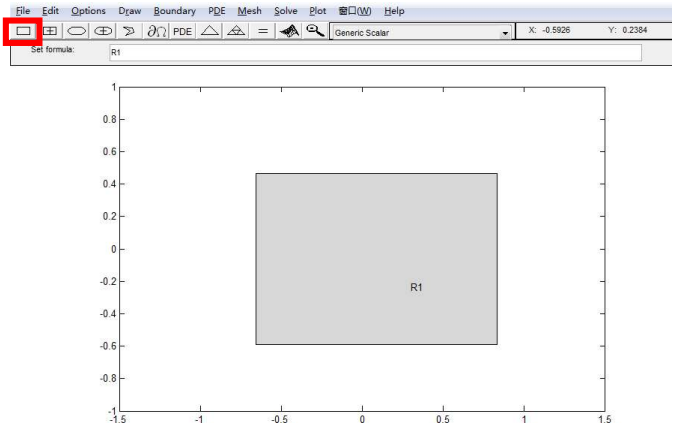


图 2 画矩形框

步骤 3：双击所画矩形框，在弹出窗口中设置参数，设定矩形框尺寸，并点击“ok”，如图 3 所示；

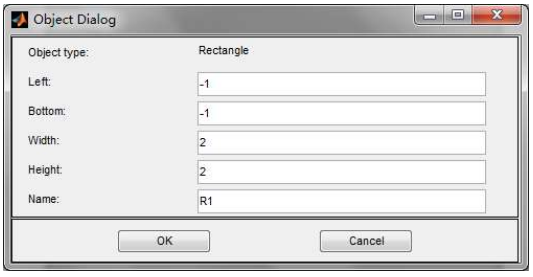


图 3 矩形框尺寸设置

步骤 4：点击工具栏中红色框中的按钮，再点击“Boundary”，在其选项中 “Show Edge Lebels”；或是直接点击工具栏中的“Boundary”，在其选项中点击“Boundary Mode”以及“Show Edge Lebels” 后矩形框变成如下形式,如图 4 所示；

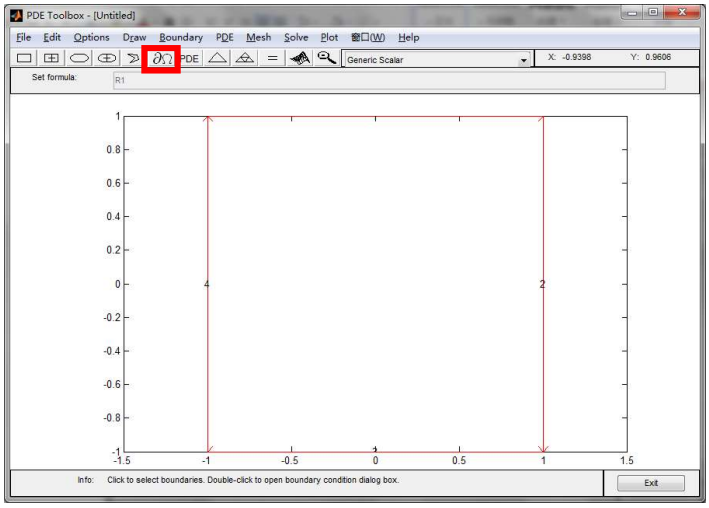


图 4 矩形框模式设置

步骤 5：设置边界条件；分别双击图 4 中的‘1’，‘2’，‘3’，‘4’四条边界线，弹出界面如图 5 所示；将 ‘2’、‘4’设置为“Dirichlet”，其中 “h=1，r=0”，将‘1’、‘3’设置为“Neumann”，其中“g=0，q=0”；设置后如图 6 所示；

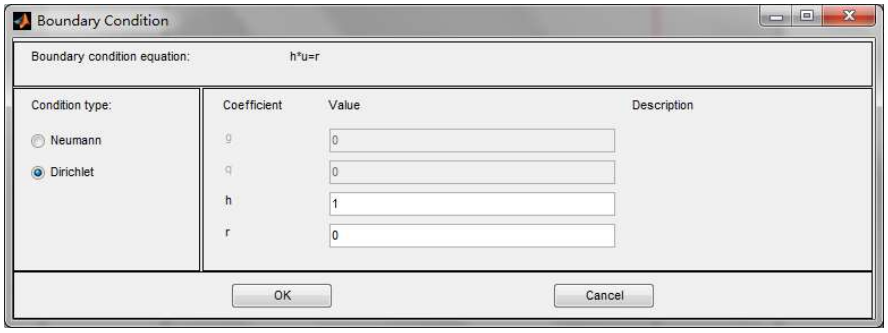


图 5 边界条件设置

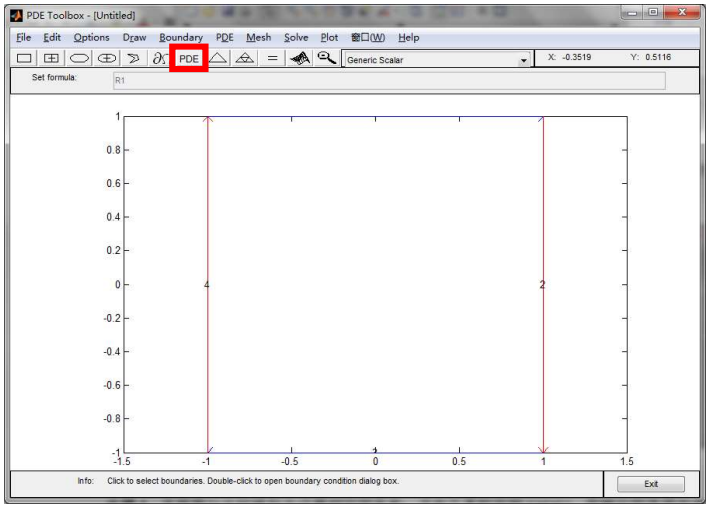


图 6 边界条件设置后结果

步骤 6：选择微分方程模型并设置相应的系数；直接点击图 6 红色框中的按钮，或是点击工具栏中的“PDE”，在弹出的选项中选择“PDE Specification”，弹出界面如图 7 所示；根据所求解的方程形式，这里选择“Hyperbolic”模型，并设置好参数，点击“ok”，如图 7 中“value”所示；

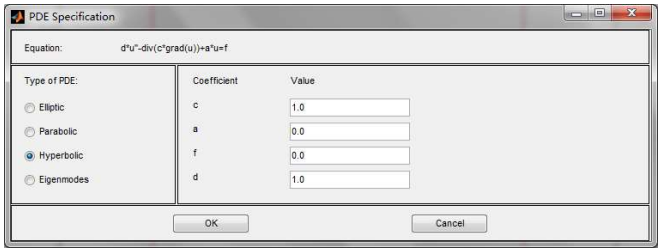


图 7 模型选择及系数设置

步骤 7：初始条件设置；点击工具栏中的“Solve”，选择“Parameters”，弹出界面如图 8 所示，并设置相应的参数；

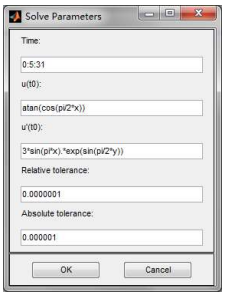


图 8 初始条件

步骤 8：划分区域；点击工具栏中的红框中左边的按钮，结果如图 9 所示；点击右边的按钮，网格变密；

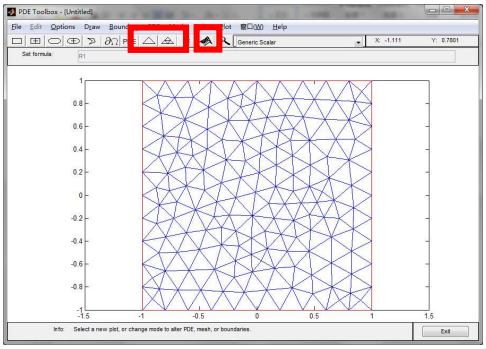


图 9 波空间有限元网格划分

步骤 9：解方程；直接点击图 9 中红色框中的第 3 个按钮，或是点击工具栏中的“Plot”，选择选项“Parameters”，均会弹出界面如图 10 所示，然后选中“Height(3-D plot)”，再点击左下角的“Plot”，开始求解并绘制结果，结果如图 11 所示；

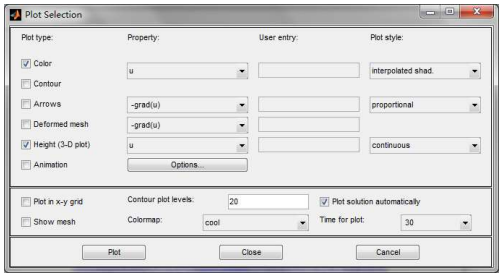


图 10 plot 参数设置

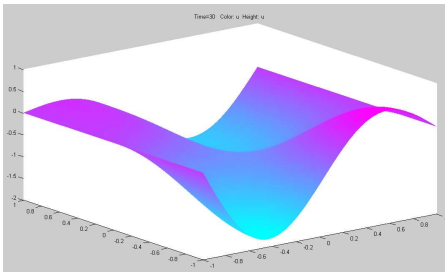


图 11 结果

对于 PDE 工具箱参数设置，大家也可以参见如下链接：

https://ww2.mathworks.cn/help/pde/ug/solve-pde-in-the-pde-app.html