Matlab关于微分方程的解法

MATLAB使用**龙格-库塔-芬尔格（Runge-Kutta-Fehlberg）**方法来解ODE问题。在有限点内计算求解。而这些点的间距有解的本身来决定。当解比较平滑时，区间内使用的点数少一些，在解变化很快时，区间内应使用较多的点。

为了得到更多的有关何时使用哪种解法和算法的信息，推荐使用helpdesk。所有求解方程通用的语法或句法在命令集中头两行给出。时间间隔将以向量t=[t0,tt]给出。

命令ode23可以求解(2,3)阶的常微分方程组，函数ode45使用(4,5)阶的龙格-库塔-芬尔格方法。注意，在这种情况下x'是x的微分不是x的转置。

在命令集中solver将被诸如ode45函数所取代。

命令集 龙格-库塔-芬尔格方法

[time,x]=solver(str,t,x0) 计算ODE或由字符串str给定的ODE的值，部分解已在向量time中给出。在向量time中给出部分解，包含的是时间值。还有部分解在矩阵x中给出，x的列向量是每个方程在这些值下的解。对于标量问题，方程的解将在向量x中给出。这些解在时间区间t(1)到t(2)上计算得到。其初始值是x0即x(t(1)).此方程组有str指定的M文件中函数表示出。这个函数需要两个参数：标量t和向量x,应该返回向量x'(即x的导数)。因为对标量ODE来说，x和x'都是标量。在M文件中输入odefile可得到更多信息。同时可以用命令numjac来计算Jacobi函数。

[t,x]=solver(str,t,x0,val) 此方程的求解过程同上，结构val包含用户给solver的命令。参见odeset和表1，可得到更多信息。

Ode45 此方法被推荐为首选方法。

Ode23 这是一个比ode45低阶的方法。

Ode113 用于更高阶或大的标量计算。

Ode23t 用于解决难度适中的问题。

Ode23s 用于解决难度较大的微分方程组。对于系统中存在常量矩阵的情况也有用。

Ode15s 与ode23相同，但要求的精度更高。

Ode23tb 用于解决难度较大的问题，对于系统中存在常量矩阵的情况也有用。

Set=odeset(set1,vak1,set2,val2,…) 返回结构set,其中包含用于ODE求解方程的设置参数，有关可用设置的信息参见表1。

Odeget(set,'set1') 返回结构set中设置set1的值。

有许多设置对odeset控制的ODE解是有用的，参见表1。例如，如果要在求解过程中画出解的图形，可以输入：inst=odeset('outputfcn','odeplot');.也可使用命令odedemo。

表1 ODE求解方程的设置参数

RelTol 给出求解方程允许的相对误差

AbsTol 给出求解方程允许的绝对误差

Refine 给出与输入点数相乘的因子

OutputFcn 这是一个带有输入函数名的字符串，该字符串将在求解函数执行的每步被调用：odephas2(画出2D的平面相位图)。Odephas3(画出3D的平面相位图)，odeplot(画出解的图形)，odeprint(显示中间结果)

OutputSel 是一个整型向量。指出哪些元素应该被传递给函数，特别是传递给OutputFcn

Stats 如果参数Stats为on,则将统计并显示出计算过程中资源消耗情况

Jacobian… 如果编写ODE文件代码以便F（t,y,jocobian）返回dF/dy,则将jacobian设置为on

Jconstant…如果雅可比数df/dy是常量，则将此参数设置为on

Jpattern… 如果编写ODE文件的编码以便函数F（[],[],jpattern）返回带有零的稀疏矩阵并输出非零元素dF/fy,则需将Jpattern设置为on

Vectorized…如果编写ODE文件的编码以便函数F(t,[y1,y2……])返回[F(t,y1) F(t,y2)…],则将此参数设置成on

Events… 如果ODE文件中带有参数‘events’，则将此参数设置成on

Mass… 如果编写ODE文件编码以实现函数F(t,[],‘mass’)返回M和M(t),应将此参数设置成on

MassConstant…如果矩阵M(t)是常量，则将此参数设置成on

MaxStep… 此参数是限定算法能使用的区间长度上限的标量

InitialStep… 给出初始步长的标量。如果给定的区间太大，算法就使用一个较小的步长

MaxOrder… 此参数只能被ode15s使用，它主要是指定ode15s的最高阶数,并且此参数应是从1到5的整数

BDF… 此参数只能被ode15s使用，如果倒推微分公式而不是使用通常所使用的微分公式，则要将它设置为on

NormControl…如果算法根据norm(e)<=max(Reltol*norm(y),Abstol)来步积分过程中的错误，则要将它设置成on

下面举几个例子

例1

（a）求解下面的ODE:

$$\begin{cases} x' = -x^2 \\ x(0) = 1 \end{cases}$$

创建函数xprim1,将此函数保存在M文件xprim1.m中：

```
function xprim=xprim1(t,x)
xprim=-x.^2;
```

然后调用MATLAB的ODE算法求解方程。然后画出解的图形：

```
[t,x]=ode45('xprim1',[0,1],1);
plot(t,x,'-',t,x,'o');
xlabel('time t0=0,tt=1');
ylabel('x values x(0)=1');
```

得到图1，MATLAB计算出的点用圆圈标记。

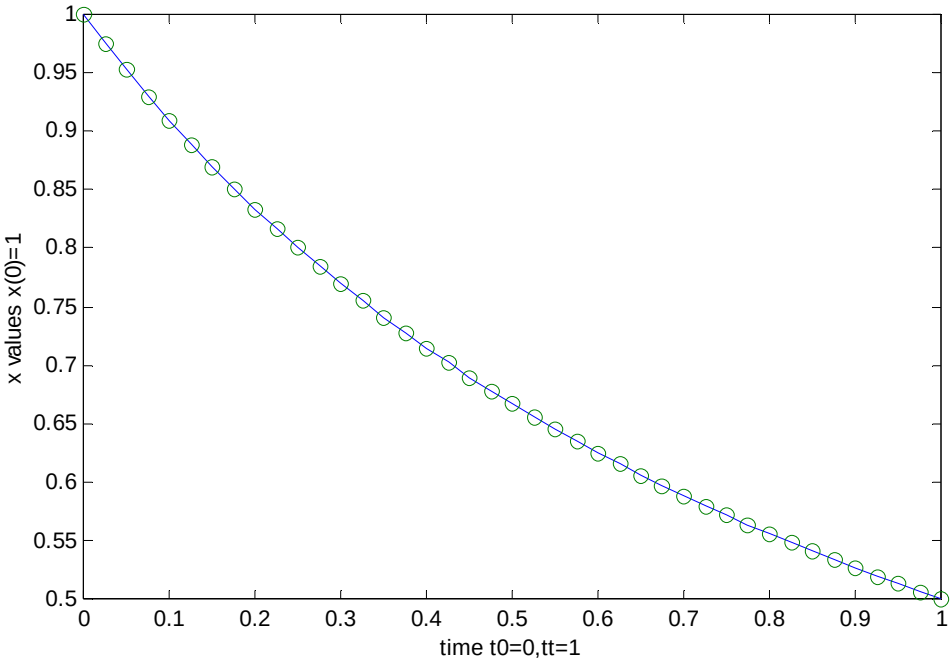

图1 由函数xprim1定义的ODE解的图形

（b） 解下面的ODE过程是等价的：

$$\begin{cases} x' = x^2 \\ x(0) = 1 \end{cases}$$

首先创建xprim2,将此函数保存在M文件xprim2.m中：

```
function xprim=xprim2(t,x)
xprim=x.^2;
```

然后调用MATLAB的ODE算法求解方程。然后画出解的图形：

```
[t,x]=ode45('xprim2',[0,0.95],1);
plot(t,x,'o',t,x,'-');
xlabel('time t0=0,tt=0.95');
ylabel('x values x(0)=1');
```

得到图2. 注意：在MATLAB中计算出的点在微分绝对值大的区域内更密集些。

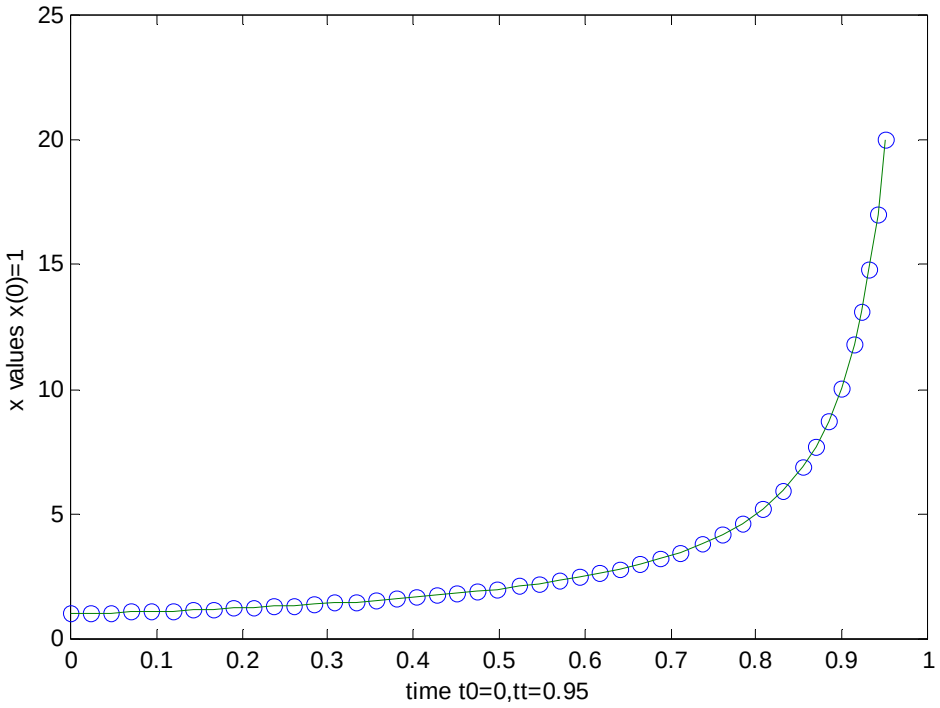

图2 由函数xprim2定义的ODE解的图形

（c） 求解

$$\begin{cases} x' = x^2 \\ x(0) = -1 \end{cases}$$

可使用与（b）中相同的函数，只要改一下初始数据即可：

```
[t,x]=ode45('xprim2',[0,1],-1);
plot(t,x);
xlabel('time t0=0,tt=1');
ylabel('x values x(0)=-1');
```

给出图3

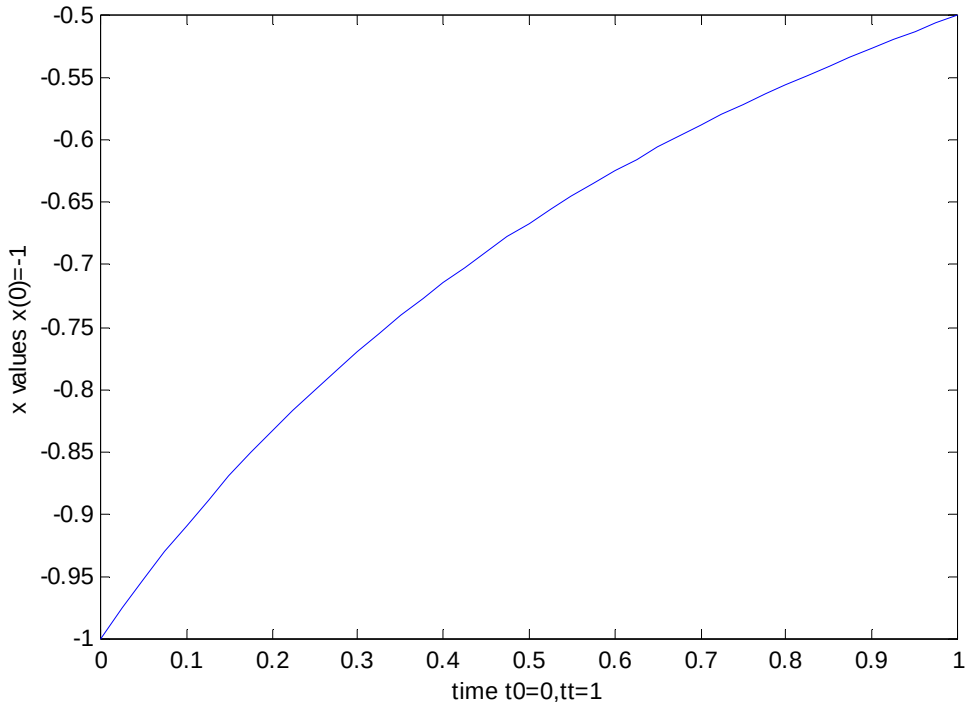

图3 给定新的初始数据，由函数xprim2定义的ODE解的图形

（d） 求解下面方程组并不难：

$$\begin{cases} x_1' = x_1 - 0.1x_1x_2 + 0.01t \\ x_2' = -x_2 + 0.02x_1x_2 + 0.04t \\ x_1(0) = 30 \\ x_2(0) = 20 \end{cases}$$

这个方程组用在人口动力学中。可以认为是单一化的捕食者---被捕食者模式。例如，狐狸和兔子。 x_1 表示被捕食者， x_2 表示捕食者。如果被捕食者有无限的食物，并且不会出现捕食者。于是有 $x_1' = x_1$，这个式子是以指数形式增长的。大量的被捕食者将会使捕食者的数量增长；同样，越来越少的捕食者会使被捕食者的数量增长。而且，人口数量也会增长。

创建xprim3,将此函数保存在M文件xprim3.m中：

```
function xprim=xprim3(t,x)
xprim=[x(1)-0.1*x(1)*x(2)+0.01*t;…
    -x(2)+0.02*x(1)*x(2)+0.04*t];
```

然后调用一个ODE算法和画出解的图形：

```
[t,x]=ode45('xprim3',[0,20],[30;20]);
plot(t,x);
xlabel('time t0=0,tt=20');
ylabel('x values x1(0)=30,x2(0)=20');
```

给出图4

在MATLAB中，也可以根据 x_2 函数绘制出 x_1 图形，命令plot(x(:2),x(:1))可绘制出平面相位图，如图5。

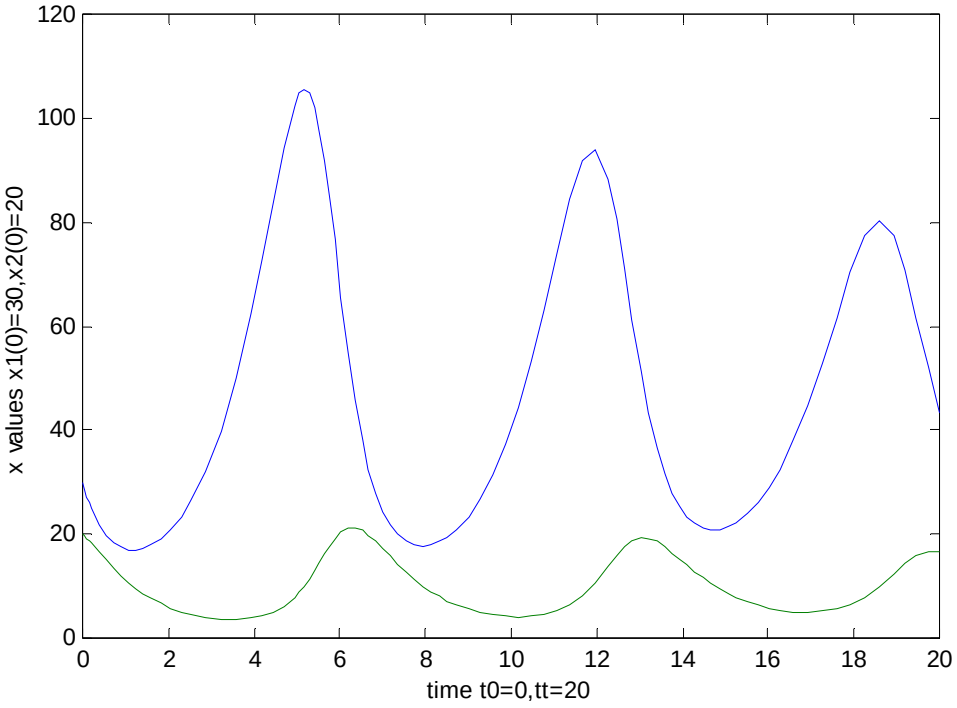

图4 由函数xprim3定义的ODE解的图形

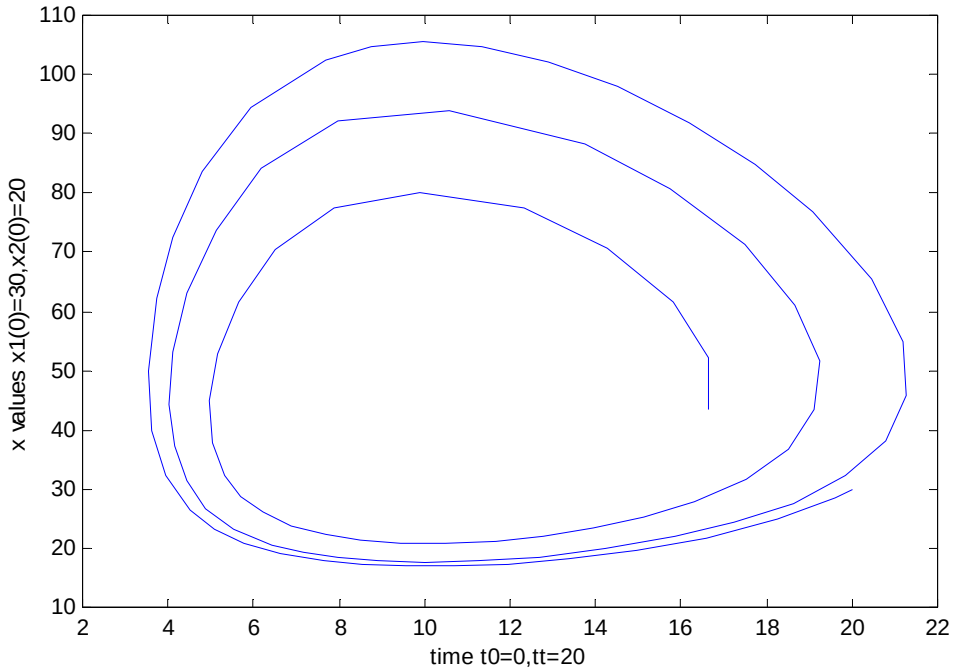

图5 由函数xprim3定义并根据函数x2计算出的x1值的曲线图

例3
对于某些a和b的值，下面的问题比较难解：

$$\begin{cases} x_1' = a-(b+1)x_1 + x_1^2 x_2 \\ x_2' = b x_1 - x_1^2 x_2 \\ x_1^0 = 1 \\ x_2^0 = 3 \end{cases}$$

方程由下面的M文件stiff1.m定义:

```
function stiff=stiff1(t,x)
global a;    %变量不能放入参数表中
glabol b;
stiff=[0,0];    %stiff必须是一个冒号变量
stiff(1)=a-(b+1)*x1+x(1)^2*x(2);
stiff(2)=b*x(1)-x(1)^2*x(2);
```

下面的M文件给出一个比较困难的问题：

```
global a;a=100;
global b;b=1;
tic;
[t,X]=ode23(‘stiff1’，[0 10],[1 3]);
toc
size(t)
```

运行后得到的结果如下：

```
elapsed_time=
  72.1647
ans=
   34009
```

使用专门解决复杂问题的解法ode23s,将得到较好的结果：

```
elapsed_time=
  1.0098
ans=
103
```

对于边界值问题，除了微分方程，还有边界处的值。在一维下这意味着至少有两个条件。现在举量哥如下的例子：

- 假设要研究一根杆的温度分布情况。这根杆一端的温度是T0,另一端的温度是T1;如图6所示。

令y(x)表示这根杆的温度，函数f(x)表示加热源。
从时间t=0开始，在相当长的时间内加热这根杆，直至达到平衡状态。这就是所谓的定常值或稳定状态。这个定常值可由下面的方程模型表示：

$$\begin{cases} -y''(x) = f(x), 0 < x < 1 \\ y(0) = T0 \\ y(1) = T1 \end{cases}$$

假设这根杆两端为：x=0和x=1。

● 假设在其两端又一根固定的柱子（或者可以看成是一个连接两个岛屿的桥），如图7所示。

令y(x)表示加载函数g(x)后弯曲的柱子。此问题需要有两个关于此柱子两端的边界条件。假设这根柱子非常牢固的固定在墙上，即y在墙上的导数是0。可以得到下面的ODE,其中介绍了自然协调系统：

$$\begin{cases} y'''(x) = g(x), 0 < x < 1 \\ y(0) = y'(0) = y(1) = y'(1) = 0 \end{cases}$$

由于存在边界值问题，不可能象解决初始值问题一样一次只执行一步地来解决问题。因此必须解一个同时给出所有未知参数的方程组。

假设又一个ODE,函数y(x)是它的解。用近似的差分来代替微分方程就能解这个ODE问题。为了能这样做，必须将区间分成有限数量的点：x0,x1,……..xM,其中xj+1=xj+,然后计算出区间内各点的近似值y()=y(),并给出确定的边界值，如y0和yM或更多的值；如图8所示。

解y(x)的导数可由有限的差分代替，如下：

$$\begin{cases} y'(x_j) \approx \dfrac{y(x_{j+1}) - y(x_j)}{\Delta x} \\ y''(x_j) \approx \dfrac{y(x_{j+1}) - 2y(x_j) + y(x_{j-1})}{\Delta x^2} \\ y'''(x_j) \approx \dfrac{y(x_{j+2}) - 4y(x_{j+!}) + 6y(x_j) - 4y(x_{j-1}) + y(x_{j-2})}{\Delta x^4} \end{cases}$$

如果用这些差分方程来代替ODE中的导数，就能得到一个所有未知的yi的方程组。其系数矩阵是一个有序区间，此区间的宽度决定于这个微分方程的导数个数。

例3

根据前面的温度模型的方程研究一下杆的温度分布，将所有的导数换成不同的差分并得到：

$$\begin{cases} \dfrac{-y_{j+1} - 2y_j + y_{j-1}}{\Delta x^2} = f_j, j = 1,...,M \\ y0 = T0 \\ y_M = T1 \end{cases}$$

其中fj=f(xj)。为了简单起见，设M=6,即给定的y0和y6，而y1,y2,….y5 为未知变量。于是就有

$$\begin{cases} -y_0+2y_1-y_2=\Delta x^2 f_1 \\ -y_1+2y_2-y_3=\Delta x^2 f_2 \\ -y_2+2y_3-y_4=\Delta x^2 f_3 \\ -y_3+2y_4-y_5=\Delta x^2 f_4 \\ -y_4+2y_5-y_6=\Delta x^2 f_5 \end{cases}$$

注意，y0=T0和yM=T1必须移到方程组的右边。此时得到的矩阵是一个对角矩阵，其对角线上的元素为2，并且上一对角线和下一对角线上的元素为1。

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -2 \end{pmatrix} * \begin{pmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{pmatrix} = \begin{pmatrix} \Delta x^2 f_1 + T0 \\ \Delta x^2 f_2 \\ \Delta x^2 f_3 \\ \Delta x^2 f_4 \\ \Delta x^2 f_5 + T1 \end{pmatrix}$$

下面解此问题的文件temperature.m。用户必须先给出分段数及f(x)(用点符号)，最后给出T0和T1。有关稀疏矩阵的更多信息参见其他资料。

```
%杆上的温度分布，用T0和T1分别表示两端温度
%这根杆放在x坐标的0和1 区间上，并被分成M个子区间，每个子区间的长度为1/M
%创建稀疏矩阵方程Ax=b并求解
%矩阵A是对角阵，并以稀疏矩阵的形式存储
clear;
M=input( ‘Give the number of subintervals (M):’);
Deltax=1/M;
xx=0:deltax:1;

funcStr= input(‘give f(x),the extra heat source(e.g.,x.^3):’,’s’);
T0=input(‘Give y(0) (left): ’);
T1=input(‘Give y(1) (right): ’);
%构造  对角阵和方程右边b
vectorOnes=ones(M-1,1);
A=spdiags([-vectorOnes,2*vectorOnes,-vectorOnes],[-1 0 1],M-1,M-1);
x=xx(2:end-1); %x为区域内的值。
f=eval(funcStr);  %响应的f(x)的值。
b=deltax^2f;
b(1)=b(1)+T0; %对边界值x=0,x=1 进行特殊处理。
b(end)=b(end)+T1;
b=b’;
%解线性方程
```

```
y=A\b; %y在区间内：j=1:M-1.
y=[T0;y;T1]; %y在整个区间内：0<=x<=1.
clf;
%上面图形表示外部热源。
%下面图形表示杆上的热分布。
subplot(2,1,1);
plot(x,f);
grid on;
title('External heat source f(x).','FontSize',14);
subplot(2,1,2);
plot(xx,y,'r');
grid on;
title('Tempearture distribution in a rod.','FontSize',14);
```

将区间分成等份，根据方程f(x)=在图9中可以得到解。

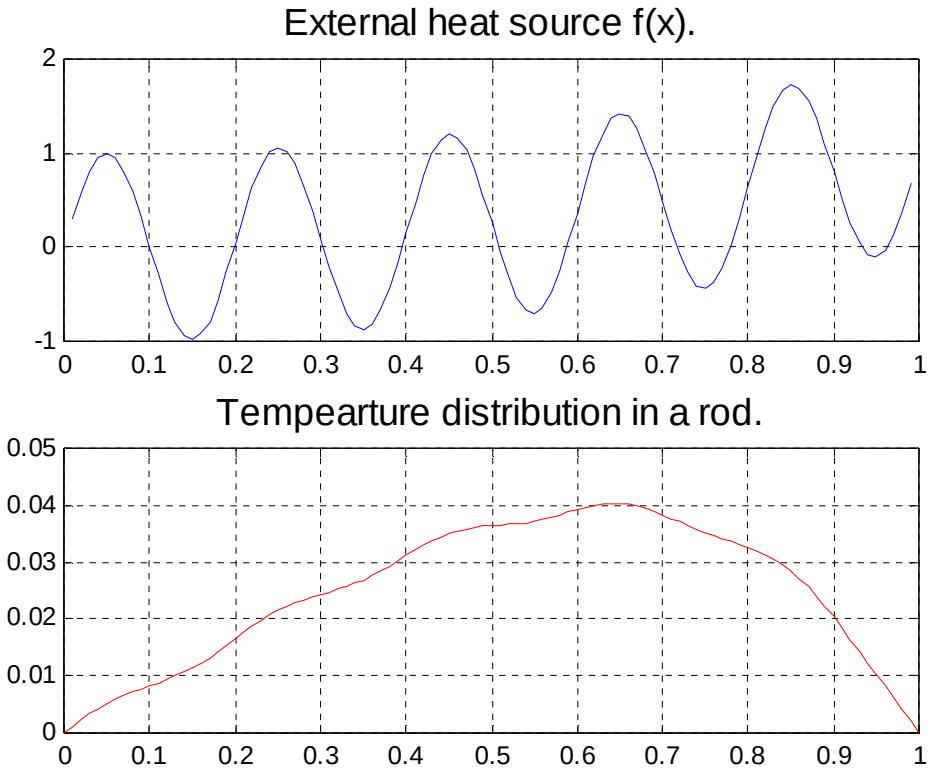

图9

例4

如果把前面柱子问题中的导数替换掉，即用 y_i 近似值表示解，就可以得到：

$$\begin{cases} \dfrac{y_{j+2}-4y_{j+1}+6y_j-4y_{j-1}+y_{j-2}}{\Delta x^4}=g_j, j=2,...,M-2 \\ y0=\dfrac{y1-y0}{\Delta x}=y_M=\dfrac{y_M-y_{M-1}}{\Delta x}=0 \end{cases}$$

将其重写为：

$$\begin{cases} y_{j+2}-4y_{j+1}+6y_j-4y_{j-1}+y_{j-2}=\Delta x^4 g_j, j=2,...,M-2 \\ y0=y1=y_{M-1}=y_M=0 \end{cases}$$

这是一个真正的线性方程组，其中用M-3个方程来解M-3个未知数：。如果M=10,则有：

$$\begin{pmatrix} 6 & -4 & 0 & 0 & 0 & 0 & 0 \\ -4 & 6 & -4 & 0 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 \end{pmatrix} * \begin{pmatrix} y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \\ y8 \end{pmatrix} = \Delta x^4 \begin{pmatrix} g2 \\ g3 \\ g4 \\ g5 \\ g6 \\ g7 \\ g8 \end{pmatrix}$$

解是一个5对角矩阵，运用运算符能很快且有效的解此方程！