

## 第11章 积分和微分方程组

在有效的MATLAB命令帮助下，可以求解出定积分和普通微分方程的数字解并绘制出其图形。

### 11.1 积分

在MATLAB中能求解如下形式的定积分并给出数字解：

$$q = \int_a^b g(x) dx$$

有许多方法都可以能够解决积分问题（又叫做求面积）。如果要用MATLAB监控整个计算过程，可以使用quad命令。同样能计算出被积函数g的值，并且让MATLAB使用梯形规则和trapz命令计算出积分。当只有离散的数据点和被积函数的数学表达式为未知时，这种方法是非常有效的。

#### 命令集107 定积分计算

trapz(x,y)	计算出函数x的积分并将结果返回到y。向量x和y有相同的长度，(xi, yi)代表曲线上的一点。曲线上点的距离不一定相等，x值也不一定有序。然而，负值间距和子区间被认为是负值积分。
trapz(y)	计算方法同上，但x值间隔为1。
trapz(x,A)	将A中每列的值带入x的函数算出其积分，并返回一组包含积分结果的向量。A的列向量必须和向量x的长度相同。
Z=trapz(x,A,dim)	在矩阵A中dim指定的维内进行数据积分。如果给定向量x，则x的长度必须与size(A, dim)相同。
cumtrapz(A,dim)	返回大小和A相同的数组，包含的是将矩阵A进行梯形积分的累积值。如果dim已给定，则在dim维内进行计算。
quad(fcn,a,b)	返回在区间[a, b]上g的积分近似值。字符串fcn包含一个与g相对应的MATLAB函数名，也就是预定义函数或者是M文件。这个函数接收一个向量参数，并返回一个向量结果。MATLAB利用辛普森规则执行递归的积分，计算误差为 $10^{-3}$ 。
quad(fcn,a,b,tol)	求g的积分近似值，其相对误差由参数tol定义。否则，计算过程同上。
quad(fcn,a b,tol pic)	求g的积分近似值，其相对误差由参数tol所定义。如果参数pic是非零值，则在图形中显示求值的点。
quad(...,trace)	如果trace是非零值，则画出积分图形。

`quad8(...)`

可以与 `quad` 一样用于相同的参数组合并返回相同的结果，但使用更高精度的方法。因此，如果被积函数的导数在某一区间内是不定的，例如： $q = \int_0^1 \sqrt{\sin x} dx$ ，使用此命令将会更好一些。`quad` 和 `quad8` 都要求被积函数在整个区间里是有限的。

`dblquad(f, min1, max1, min2, max2, tol, trace, order)`

计算双变量函数  $f$  的二重积分。函数中的第一个自变量用于内层积分。内层积分在  $min1$  和  $max1$  之间进行，外层积分在  $min2$  和  $max2$  之间进行。变量  $tol$  指定相对误差。 $trace$  的使用方法与 `quad` 相同。根据字符串 **order**，对于相同的访问，`dblquad` 能选择使用 `quad`、`quad8` 和许多用户定义的积分方法，并返回与 `quad` 相同的变量。

输入 `quaddemo` 可以看到一个演示实例。

#### 例11.1

下面用不同的方法来计算下列积分：

$$\int_0^1 e^{-x^2} dx$$

(a) 使用 `trapz` 命令。首先创建一个有  $x$  值的向量。用 5 和 10 两个值进行计算：

```
x5 = linspace(0,1,5); x10 = linspace(0,1,10);
```

然后创建  $x$  的函数  $y$ ：

```
y5 = exp(-x5.^2); y10 = exp(-x10.^2);
```

现在计算出积分值：

```
format long;
integral5 = trapz(x5,y5), ...
integral10 = trapz(x10,y10)
```

返回

```
integral5 =
    0.74298409780038
```

```
integral10 =
    0.74606686791267
```

(b) 使用 `quad` 命令。首先在 M 文件中创建函数。此文件 `integrand.m` 包含函数，如下：

```
function y = integrand(x)
```

```
y = exp(-x.^2);
```

首先以标准误差计算积分，然后再以指定误差计算积分。

```
format long;
integralStd = quad('integrand',0,1)
integralTol = quad('integrand',0,1,0.00001)
```

给出

```
integralStd =
    0.74682612052747
```

```
integralTol =
    0.74682414517798
```

(c) 使用quad8命令：使用在(b)中创建的M文件,然后输入：

```
integral8Std = quad8('integrand',0,1)
```

```
integral8Std =
    0.74682413281243
```

这是MATLAB所能给出的最精确的结果。

(d) 使用cumtrapz命令能很容易地计算出不同区间的积分。

```
x = 0:5;
cumtrapz(x)
```

```
ans =
    0    0.5000    2.0000    4.5000    8.0000   12.5000
```

(e) 计算二重积分：

$$\int_0^1 \int_0^1 e^{-x^2-y^2} dy dx$$

如图11-1。首先创建一个包含函数 M文件:integrand2.m:

```
function y = integrand2(x,y)
```

```
y = exp(-x.^2-y.^2);
```

然后用quad命令计算对于固定的x值在y方向的一些积分值：

```
x = linspace(0,1,15);
```

```
for i = 1:15
```

```
    integral(i) = quad('integrand2',0,1,[],[],x(i));
```

```
end
```

现在已计算出在y方向的15个积分值。trapz命令能使用这些值来计算二重积分：

```
format short;
```

```
dIntegral = trapz(x,integral)
```

```
dIntegral =
    0.5575
```

输入下列语句可以得到一个积分区域的图形：

```
[X,Y] = meshgrid(0:.1:1,0:.1:1);
```

```
Z = integrand2(X,Y);
```

```
mesh(X,Y,Z); view(30,30);
```

结果如图11-1所示。命令mesh和view定义在13.5节中。

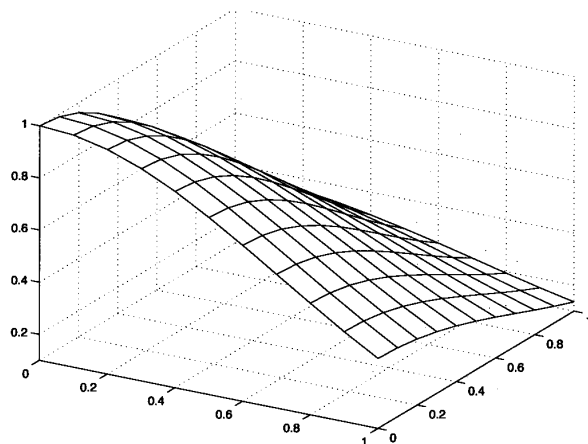


图11-1 函数 $e^{-x^2-y^2}$ 在区间 $[0, 1] \times [0, 1]$ 的上的图形

不定积分  $\int_a^x f(t)dt$  不能使用上面的命令来计算。MATLAB中的数学符号工具箱和MATLAB的编辑器能提供处理这些积分的命令。

## 11.2 常微分方程组

下面来研究常微分方程系统 ODE, 该系统处理的是初始值已知的一阶微分方程。在本节中主要讨论这种类型的微分方程, 同时也会举出两个有关边界值问题的例子。可以利用 ODE 系统创建稀疏线性系统方程来求解这些例子。

在数学符号工具箱中, 有一些命令能给出常微分方程的符号解, 即解以数学表达式的形式给出。在下面的初始值问题中, 有两个未知函数:  $x_1(t)$  和  $x_2(t)$ , 并用以下式子表达其微分形式:

$$\frac{dx_i}{dt} = x_i$$

在许多应用中, 独立变量参数  $t$  表示时间。

$$\begin{cases} x_1' = f_1(x_1, x_2, t) \\ x_2' = f_2(x_1, x_2, t) \\ x_1(t_0) = x_{1,0} \\ x_2(t_0) = x_{2,0} \end{cases}$$

高阶的 ODE 能表达成第 1 阶的 ODE 系统。例如, 有以下微分方程:

$$\begin{cases} x'' = f(x, x', t) \\ x(t_0) = x_0 \\ x'(t_0) = xp_0 \end{cases}$$

用  $x_2$  替换  $x'$  用  $x_1$  替换  $x$ , 就能得到:

$$\begin{cases} x_1' = x_2 \\ x_2' = f(x_1, x_2, t) \\ x_1(t_0) = x_0 \\ x_2(t_0) = xp_0 \end{cases}$$

这是一个第1阶的ODE系统。

对于某一时间间隔  $0 \leq t \leq T$ ，初始值问题的解决方法是将时间分成一组有限和离散的时间点，例如用相同的时间间隔  $\Delta t$  进行等分：

$$t_i = i \Delta t, \quad i = 0, \dots, N$$

其中时间步长  $\Delta t = T/N$ ， $N$  为某一整数。这种导数能被微分方程的可微分的商所代替，微分方程表示在不同时间点的解。见例 11.2，给出更多的有关有限微分商的信息。这种方法的稳定性取决于  $\Delta t$  的大小和所采用的数值方法，用这种方法能得到 ODE 的近似值。

在许多应用中有一些微分过程非常复杂的微分方程，在某些区域里这些方程要求有非常小的时间步长  $\Delta t$ 。解决这些问题的困难在于问题中涉及不同的时间尺度，如解的导数可能有较大的变化。

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，而这些点的间距由解本身来决定。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 108 中头两行给出。时间间隔将以向量  $t = [t_0, t_f]$  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  $\mathbf{x}'$  是  $\mathbf{x}$  的微分，不是  $\mathbf{x}$  的转置。

在命令集 108 中 `solver` 将被诸如 `ode45` 函数所代替。

#### 命令集 108 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))`。此方程组由 `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` 来计算雅可比函数。

```
[t,X]=
solver(str,t,x0,val)
ode45
ode23
ode113
ode23t
ode23s
```

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 11-1，可得更多信息。

此方法被推荐为首选方法。

这是一个比 `ode45` 低阶的方法。

用于更高阶或大的标量计算。

用于解决难度适中的问题。

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ode23s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,val1, 返回结构set,其中包含用于ODE求解方程的设置参数。  
set2,val2,...) 有关可用设置的信息参见表 11-1。

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 11-1。例如，如果在求解过程中要画出解的图形，可以输入：`inst=odeset('OutputFcn','odeplot');`。

表11-1 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	给出与输出点数相乘的因子
OutputFcn	这是一个带有输出函数名的字符串，该字符串将在求解函数执行的每步被调用： <code>odephas2</code> (画出2D的平面相位图), <code>odephas3</code> (画出3D的平面相位图), <code>odeplot</code> (画出解的图形), <code>odeprint</code> (显示中间结果)
OutputSel	是一个整型向量，指出哪些元素应被传递给函数，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 $F(t,y,'jacobian')$ 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 $dF/dy$ ，则需将Jpattern设置成 on
Vectorized	如果编写 ODE 文件编码以便函数 $F(t,[y1,y2'\cdots])$ 返回 $[F(t,y1) F(t,y2\cdots)]$ ，则将此参数设置成 on
Events	如果ODE文件中有带有参数 '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，则将此参数设置为 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。如果使用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 $\text{norm}(e) \leq \max(\text{RelTol} * \text{norm}(y), \text{AbsTol})$ 来步积分过程中的错误，则要将它设置为 on

也可试用命令 `odedemo`。

## 例11.2

(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，最后画出解的图形：

```
[t,x] = ode45('xprim1',[0 1],1);
```

```
plot(t,x,'-o',t,x,'o');
```

```
xlabel('time t0 = 0, tt = 1');
```

```
ylabel('x values x(0) = 1');
```

得到图11-2。MATLAB计算出的解用圆圈标记。在13.1节中介绍绘图命令`plot`。

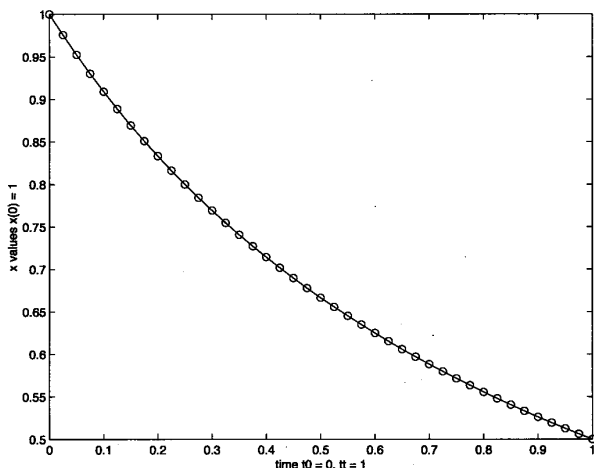


图11-2 由函数`xprim1`定义的ODE解的图形

(b) 解下面的ODE过程是等价的：

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

首先创建函数`xprim2`，并将其保存在M文件`xprim2.m`中：

```
function xprim = xprim2(t,x)
```

```
xprim = x.^2;
```

然后调用ODE的求解方程并画出其解的图形：

```
[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11-3。

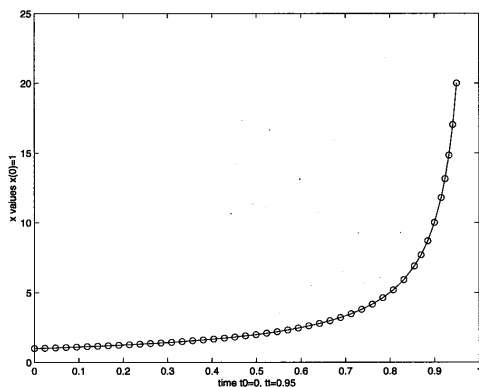


图11-3 由函数xprim2定义的ODE解的图形

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。

(c) 求解：

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

可使用与(b)中相同的函数。只要改变一下初始数据即可：

```
[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11-4。

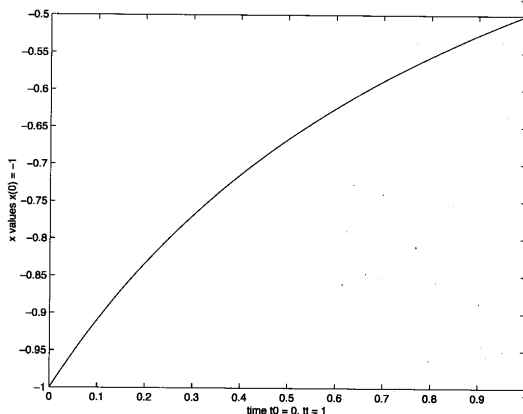


图11-4 给定新的初始数据，由函数xprim2定义的ODE解的图形



(d) 求解下面方程组并不很难：

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

这个方程组应用在人口动力学中，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。洛特卡和伏尔泰拉在20世纪20年代已对这些非线性的微分方程进行了研究。

创建函数`xprim3`，并将其保存在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');
```

所得结果如图11-5所示。

在MATLAB中，也可以根据 $x_2$ 函数绘制出 $x_1$ 的图形。命令`plot(x(:,2),x(:,1))`可绘制出平面相位图，如图11-6所示。

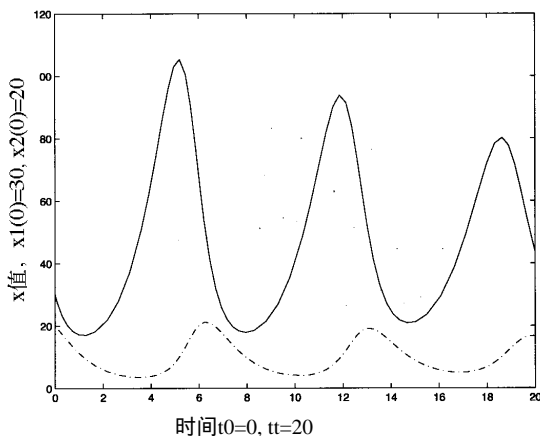


图11-5 函数`xprim3`定义的ODE解的图形

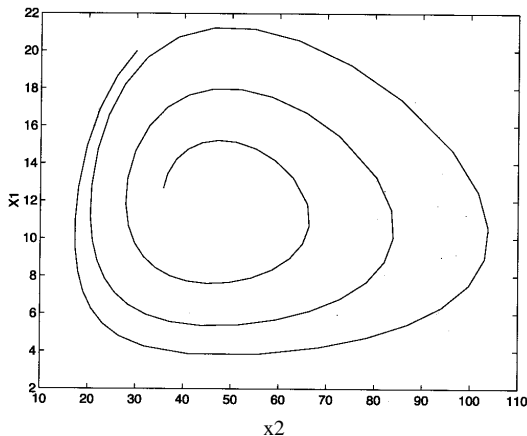


图11-6 由函数`xprim3`定义并根据函数 $x_2$ 计算出的 $x_1$ 值的曲线图

### 例11.3

对于某些 $a$ 和 $b$ 值，下面的问题比较难解：

$$\begin{cases} x_1' = a - (b+1)x_1 + x_1^2 x_2 \\ x_2' = bx_1 - x_1^2 x_2 \\ x_1^0 = 1 \\ x_2^0 = 3 \end{cases}$$

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
global b;
stiff=[0;0];        % Stiff必须是一个冒号向量

stiff(1) = a - (b+1)*x(1) + 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    1
```

使用专门解决复杂问题的解法ode23s，将会得到较好的结果：

```
elapsed_time =
    1.0098
```

```
ans =
    103    1
```

对于边界值问题，除了微分方程，还有在边界处的值。在一维下这意味至少有两个条件。现在举两个如下的例子：

- 假设要研究一根杆的温度分布情况。这根杆一端的温度是  $T_0$ ，另一端的温度是  $T_1$ ；如图11-7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：

$$\begin{cases} -y''(x) = f(x), & 0 < x < 1 \\ y(0) = T_0 \\ y(1) = T_1 \end{cases}$$

假设这根杆两端为： $x=0$  和  $x=1$ 。

- 假设在其两端有一根固定的柱子(或者可以看成是一个连接两个岛屿的桥)，如图11-8所示。

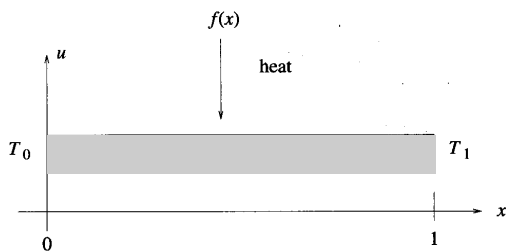


图11-7 在一根杆上的温度分布图

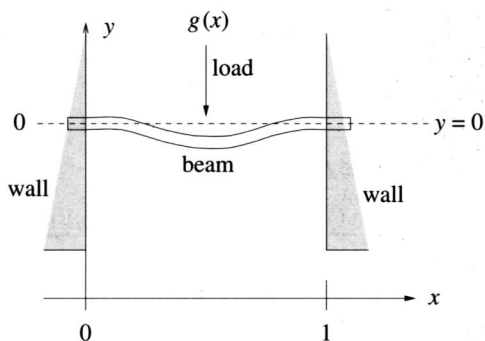


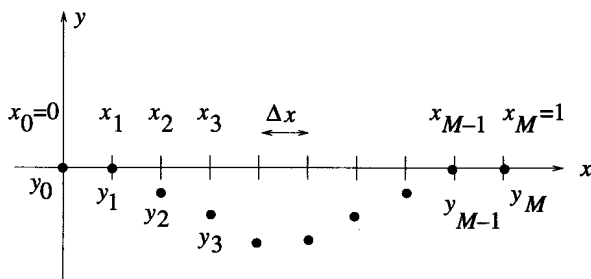
图11-8 在柱子上加载重量(相当于桥上交通堵塞)

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地固定在墙上，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。

假设有一个 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。为了能做到，必须将区间分成有限数量的点： $x_0, x_1, \dots, x_M$ ，其中  $x_{j+1} = x_j + \Delta x$ ，然后计算出区间内各点的近似值  $y_j \approx y(x_j)$ ，并给出确定的边界值，如  $y_0$  和  $y_M$  或更多的值；如图 11-9 所示。

图11-9 将计算区间  $[0, 1]$  分成  $M$  等份

解  $y(x)$  的导数可由有限的差分代替，如下：

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

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  $y_j$  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11.4

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：

$$\begin{cases} -\frac{y_{j+1} - 2y_j + y_{j-1}}{\Delta x^2} = f_j, & j = 1, \dots, M \\ y_0 = T_0 \\ y_M = T_1 \end{cases}$$

其中  $f_j = f(x_j)$ 。为了简单起见，设  $M=6$ ，即给定  $y_0$  和  $y_6$ ，而  $y_1, y_2, \dots, y_5$  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， $y_0 = T_0$  和  $y_M = T_1$  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。

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} \Delta x^2 f_1 + T_0 \\ \Delta x^2 f_2 \\ \Delta x^2 f_3 \\ \Delta x^2 f_4 \\ \Delta x^2 f_5 + T_1 \end{pmatrix}$$

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  $T_0$  和  $T_1$ 。有关稀疏矩阵更多信息参见第 9 章。

```
% 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，每一个子区间的长度为 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): ');
```

```
% 构造对角矩阵 A 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);         % 相应的f(x)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=1,2,...,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('Temperature distribution in a rod.', 'FontSize', 14);

```

将区间分成100等份，根据方程 $f(x)=x^2+\sin(10\pi x)$ 在图11-10中可以得到解。

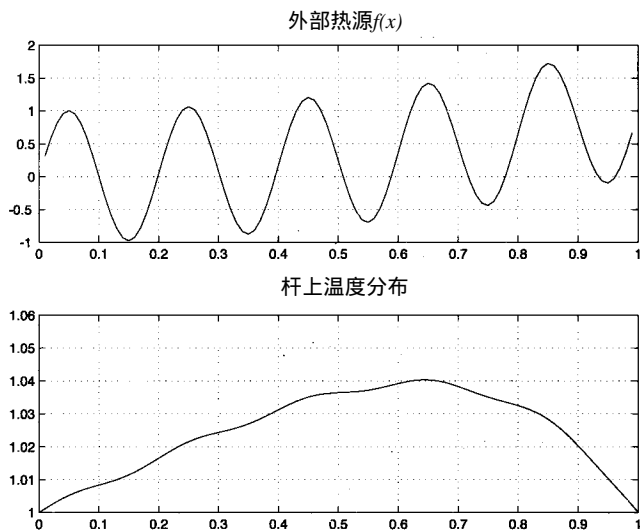


图11-10 边界值问题的解：杆的温度分布

### 例11.5

如果把前面柱子例题中的导数替换掉，即用  $y_j$  近似值表示解，就可以得到：

$$\begin{cases} \frac{y_{j+2} - 4y_{j+1} + 6y_j - 4y_{j-1} + y_{j-2}}{\Delta x^4} = g_j, & j = 2, \dots, M-2 \\ y_0 = \frac{y_1 - y_0}{\Delta x} = y_M = \frac{y_M - y_{M-1}}{\Delta x} = 0, \end{cases}$$

将其重写为：

$$\begin{cases} y_{j+2} - 4y_{j+1} + 6y_j - 4y_{j-1} + y_{j-2} = \Delta x^4 g_j, & j = 2, \dots, M-2 \\ y_0 = y_1 = y_{M-1} = y_M = 0 \end{cases}$$

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： $y_2, y_3, \dots, y_{M-2}$ 。如果  $M=10$ ，就有：

解是一个5对角矩阵，使用 \ 运算符能很快且有效地解出此方程。

$$\begin{pmatrix} 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 \end{pmatrix} \begin{pmatrix} y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} = \Delta x^4 \begin{pmatrix} g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \end{pmatrix}$$