精通 MATLAB

—综合辅导与指南—

前 言

这是一本有关 MATLAB 的参考书，适合于使用 MATLAB 或正在打算使用 MATLAB 的读者。本书另辟蹊径可以借助或不借助 MATLAB 随带的文档资料让读者自学 MATLAB。书中口语化的风格,使读者易于阅读。如书名所示,本书提供了读者精通 MATLAB 所需的工具。作为编程语言和可视化工具,MATLAB 具有丰富的一系列功能,可解决工程、科学计算和数学学科中许多问题。本书的基本目的是通过向读者展示如何有效地使用这些功能来帮助读者增强工作能力。由于 MATLAB 交互式的性质,书中内容以举例方式来描述。在读者阅读本书的同时,这些例子可以通过运行 MATLAB 而再现。

本书只涉及一般读者所用到的一些专题,所提供的资料可用于包括 UNIX 工作站、Macintosh 和 PC 在内的所有计算机平台。除了标准的 MATLAB 本身这部分的功能之外,书中只讨论了字符工具箱。其它更为专用的工具箱没有进行讨论。而且,没有讨论与机器有关的 MATLAB 诸方面,例如 MEX 文件的编写。

本书开发了许多 M 文件函数,它扩展了 MATLAB 的功能。在书中,作者演示了各种 MATLAB 的功能和编程技术,它们总称为精通 MATLAB 的工具箱。这些 M 文件存在软盘中,只要寄送书内的明信片,可由 MathWorks 公司免费提供。另一种办法可用 MathWork 的 FTP 获得。有关这个办法的信息,参阅 23 章。读者可写信到 **MathWorks Inc.,24 Prime Parkway ,Natick,MAO1760;** 电话:**(508)647-7000;** 传真:**(508)647-7001;email: info@mathworks.com; WWW:http://www.mathwork.com** 与 **Mathworks** 公司直接联系。

因为,作为一个软件工具,MATLAB 在不断的演变, 本书重点是 MATLAB 的版本 4.2c,其绝大部分内容同样适用于所有 MATLAB 4.x 版本。必须时, 我们指出了版本之间的区别, 而且标注了在 MATLAB 版本 5.0 中所能见到的变化。

作者鼓励大家对本书提出反馈意见:本书的最佳特点是什么?哪些地方需要作更多的工作?哪些专题应该删去?应该加上什么专题?用 email 可与我们联系 地址:mm@eece.maine.edu。

致谢 (略)

达恩.亨塞尔曼
勃鲁司.利特

目　录

第 5 章　关系和逻辑运算

除了传统的数学运算，MATLAB 支持关系和逻辑运算。如果你已经有了一些编程经验，就会对这些运算熟悉。这些操作符和函数的目的是提供求解真/假命题的答案。一个重要的应用是控制基于真/假命题的一系列 MATLAB 命令（通常在 M 文件中）的流程，或执行次序。

作为所有关系和逻辑表达式的输入，MATLAB 把任何非零数值当作真，把零当作假。所有关系和逻辑表达式的输出，对于真，输出为 1；对于假，输出为零。

5.1　关系操作符

MATLAB 关系操作符包括所有常用的比较。

表 5.1

关系操作符	说明
<	小于
< =	小于或等于
>	大于
> =	大于或等于
= =	等于
~ =	不等于

MATLAB 关系操作符能用来比较两个同样大小的数组，或用来比较一个数组和一个标量。在后一种情况，标量和数组中的每一个元素相比较，结果与数组大小一样。下面给出几个示例：

```
» A=1:9, B=9-A
A =
     1     2     3     4     5     6     7     8     9
B =
     8     7     6     5     4     3     2     1     0
» tf=A>4
tf =
     0     0     0     0     1     1     1     1     1
```

找出 **A** 中大于 4 的元素。0 出现在 **A<=4** 的地方，1 出现在 **A>4** 的地方。

```
» tf=(A= =B)
tf =
     0     0     0     0     0     0     0     0     0
```

找出 **A** 中的元素等于 **B** 中的元素。**注意，=和= =意味着两种不同的事：= = 比较两个变量，当它们相等时返回 1，当它们不相等时返回 0；在另一方面， = 被用来将运算的结果赋给一个变量。**

```
» tf=B-(A>2)
tf =
     8     7     5     4     3     2     1     0    -1
```

找出 **A>2**，并从 **B** 中减去所求得的结果向量。这个例子说明，由于逻辑运算的输出是 1 和 0 的数组，它们也能用在数学运算中。

```
» B=B+(B==0)*eps
B =
  Columns 1 through 7
    8.0000    7.0000    6.0000    5.0000    4.0000    3.0000    2.0000
  Columns 8 through 9
    1.0000    0.0000
```

这是一个演示，表明如何用特殊的 MATLAB 数 **eps** 来代替在一个数组中的零元素，**eps** 近似为 2.2e-16。这种特殊的表达式在避免被 0 除时是很有用的。

```
» x=(-3:3)/3
x =
   -1.0000   -0.6667   -0.3333         0    0.3333    0.6667    1.0000
» sin(x)./x
Warning: Divide by zero
ans =
    0.8415    0.9276    0.9816       NaN    0.9816    0.9276    0.8415
```

由于第四个数据是 0 ，计算函数 sin(x)/x 时给出了一个警告。由于 sin(0)/0 是没定义的，在该处 MATLAB 结果返回 **NaN**。用 **eps** 替代 0 以后，再试一次，

```
» x=x+(x==0)*eps;
» sin(x)./x
ans =
    0.8415    0.9276    0.9816    1.0000    0.9816    0.9276    0.8415
```

现在 sin(x)/x 在 x=0 处给出了正确的极限。

5.2 逻辑操作符

逻辑操作符提供了一种组合或否定关系表达式。MATLAB 逻辑操作符包括：

表 5.2

逻辑操作符	说明
&	与
|	或
~	非

逻辑操作符用法的一些例子有：

```
» A=1:9;B=9-A;
» tf=A>4
tf =
 0     0     0     0     1     1     1     1     1
```

找出 **A** 大于 4。

```
» tf=~(A>4)
tf =
    1     1     1     1     0     0     0     0     0
```

对上面的结果取非，也就是 1 替换 0，0 替换 1。

```
» tf=(A>2)&(A<6)
tf =
    0     0     1     1     1     0     0     0     0
```

在 **A** 大于 2‘与’**A** 小于 6 处返回 1。

最后，上面的功能易于产生数组来表示不连续信号，或由多段其他信号所组成的信号。基本想法是，把数组中要保持的那些值与 1 相乘，所有其他值与 0 相乘。例如，

```
» x=linspace(0, 10, 100);        %   create data
» y=sin(x) ;                     %   compute    sine
» z=(y>=0).*y ;                  %   set negative values of sin(x) to zero
» z=z+0.5*(y<0) ;                %   where sin(x) is negative add 1/2
» z=(x<=8).*z ;                  %   set values past x=8 to zero
» plot(x, z)
» xlabel(' x '),    ylabel(' z=f(x) '),    title(' A Discontinuous Signal ')
```

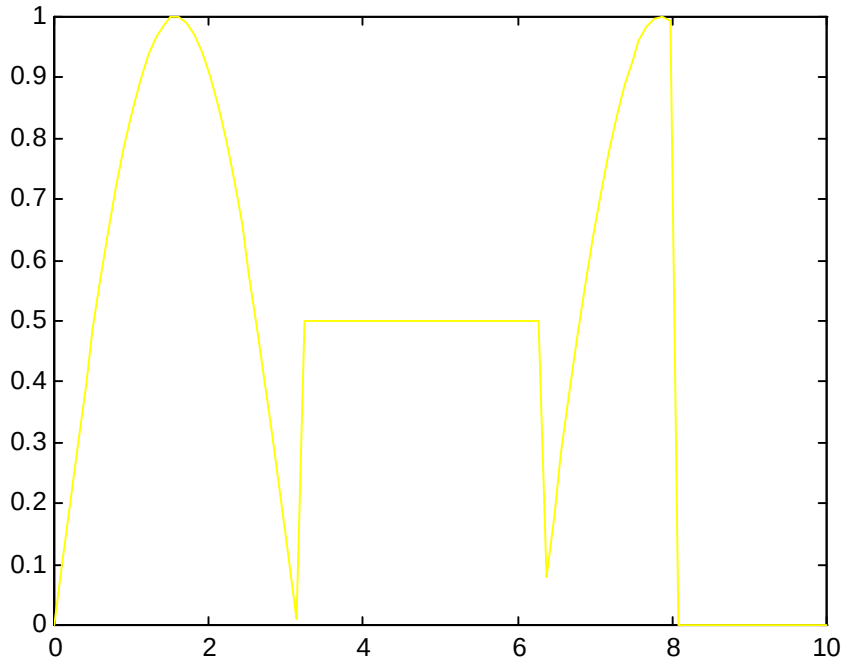


图 5.1　不连续信号

5. 3　关系与逻辑函数

除了上面的关系与逻辑操作符，MATLAB 提供了大量的其他关系与逻辑函数，包括：

表 5.3

其　他　关　系　与　逻　辑　函　数	
xor(x,y)	异或运算。x 或 y 非零(真)返回 1，x 和 y 都是零(假)或都是非零(真)返回 0。
any(x)	如果在一个向量 x 中，任何元素是非零，返回 1；矩阵 x 中的每一列有非零元素，返回 1。
all(x)	如果在一个向量 x 中，所有元素非零，返回 1；矩阵 x 中的每一列所有元素非零，返回 1。

除了这些函数，MATLAB 还提供了大量的函数，测试特殊值或条件的存在，返回逻辑值。

表 5.4

测　试　函　数	
finite	元素有限，返回真值。
isempty	参量为空，返回真值。
isglobal	参量是一个全局变量，返回真值。
ishold	当前绘图保持状态是‘ON’，返回真值。
isieee	计算机执行 IEEE 算术运算，返回真值。
isinf	元素无穷大，返回真值。
isletter	元素为字母，返回真值。
isnan	元素为不定值，返回真值。
isreal	参量无虚部，返回真值。
isspace	元素为空格字符，返回真值。
isstr	参量为一个字符串，返回真值。
isstudent	MATLAB 为学生版，返回真值。
isunix	计算机为 UNIX 系统，返回真值。
isvms	计算机为 VMS 系统，返回真值。

5.4　NaNs 和空矩阵

NaNs 和空矩阵([　])要求在 MATLAB 中作特殊处理，特别是用在逻辑或关系表达式里。根据 IEEE 数学标准，对 **NaNs** 的几乎所有运算都得出 **NaNs**。例如，

```
» a=[1    2    nan    inf    nan]     %    note,in use, NaN can be lowercase
a =
       1        2      NaN      Inf      NaN

» b=2*a
b =
       2        4      NaN      Inf      NaN

» c=sqrt(a)
c =
      1.0000       1.4142            NaN            Inf            NaN

» d = (a= =nan)
d =
       0        0        0        0        0

» f = (a~ =nan)
f =
       1        1        1        1        1
```

上面的第一和第二计算式对 **NaN** 输入给出 **NaN** 结果。然而，最后两个计算式产生有点令人惊讶的结果。当 **NaN** 与 **NaN** 相比较时，(**a= =nan**)产生全部为 0 或假的结果，同时(**a~ =nan**)产生全部 1 或真值。于是，单个 **NaNs** 相互不相等。由于 **NaNs** 的这种特性，MATLAB 有一个内置逻辑函数寻找 **NaNs**。

```
» g=isnan(a)
g =
     0     0     1     0     1
```

这个函数用 **find** 命令能找出 **NaNs** 的下标值。例如，

```
» i=find(isnan(a))          %    find indices of NaNs
i =
     3     5
» a(i)=zeros(size(i))       %    changes NaNs in a to zeros
a =
     1     2     0   Inf     0
```

当 **NaNs** 数学上由 IEEE 标准充分定义时，空矩阵由 MATLAB 的生成器确定，并有它自己的特性。空矩阵是简单的，它们是 MATLAB 大小为零的变量。

```
» size([   ])
ans =
     0     0
```

当没有其它合适的结果时，在 MATLAB 中的许多函数返回空矩阵。或许最普通的例子是函数 **find**：

```
» x=(1:5)-3         %    new data
x =
    -2    -1     0     1     2
» y=find(x>2)
y =
     [   ]
```

在这个例子里，**x** 没有包含大于 2 的值，所以没有返回下标。为了测试空结果，MATLAB 提供了逻辑函数 **isempty**。

```
» isempty(y)
ans =
     1
```

在 MATLAB 里，空矩阵不等于任何非零矩阵(或标量)。这个事实由下面例子给出：

```
» y=[    ];
» a=(y==0)
a =
        0
```

这说明一个空矩阵不等于一个标量，因此，

```
»find(y= =0)
ans =
        [    ]
```

也就是说没有返回下标。同样地，

```
b=(y~=0)
b =
        1
```

一个空矩阵不等于一个标量。但

```
» j=find(y~=0)
j =
        1
```

尽管 y 大小为零，现在有一个下标值！上面最后的一个例子是 MATLAB 中一个非文本变化，由于版本 3.5 先于版本 4.0，一个空矩阵与一个非空矩阵比较返回一个空矩阵。‘**这个新的解释通常导致产生问题，因为 y(find(y~=0))不存在。**’例如，

```
» y(find(y~=0))
???    Index exceeds matrix dimensions.
```

这里 MATLAB 报告一个错误，因为下标超出了空矩阵 y 的维数。

NaNs 和空矩阵的特性概括在表 5.5 中。

表 5.5

NaNs 和 空 矩 阵

数据	a=[1 2 nan inf nan]
表达式	结果
2*a	[2 4 NaN ∞ NaN]
(a= =nan)	[0 0 0 0 0]
(a~ =nan)	[1 1 1 1 1]
isnan(a)	[0 0 1 0 1]
y=find(a= =0)	y=[ ]
isempty(y)	1

(y= =0)	0
find(y= =0)	[ ]
(y~ =0)	1
j=find(y~ =0)	j=1
y(j)	Error! y(j) does not exist.

第 6 章 文 本

MATLAB 真正强有力的地方在于它的数值处理能力。然而，经常希望操作文本，例如把标号和标题放在图上。在 MATLAB 里，文本当作特征字符串或简单地当作字符串。

6.1 字符串

在 MATLAB 中的字符串一般是 ASCII 值的数值数组，它作为字符串表达式进行显示。例如，

```
» t=' How about this character string? '
t =
How about this character string?
» size(t)
ans =
     1    32
» whos
  Name          Size              Elements    Bytes  Density  Complex
  ans           1 by 2                   2       16  Full     No
  t             1 by 32                 32      256  Full     No
```

一个字符串是由单引号括起来的简单文本。在字符串里的每个字符是数组里的一个元素，字符串的存储要求每个字符 8 个字节，如同 MATLAB 的其它变量。因为 ASCII 字符只要求一个字节，故这种存储要求是浪费的，7/8 所分配的存储空间无用。然而，对字符串保持同样的数据结构简化 MATLAB 的内部数据结构。所给出的字符串操作并不是 MATLAB 的基本特点，但这种表达是方便和可接受的。

为了了解下面字符串的 ASCII 表达，只需对字符串执行一些算术运算。最简单和计算上最有效的方法是取数组的绝对值。例如，

```
» u=abs(t)
u =
  Columns 1 through 12
```

```
    72   111   119    32    97    98   111   117   116    32   116   104
  Columns 13 through 24
   105   115    32    99   104    97   114    97    99   116   101   114
  Columns 25 through 32
    32   115   116   114   105   110   103    63

» u=t+0
u =
  Columns 1 through 12
    72   111   119    32    97    98   111   117   116    32   116   104
  Columns 13 through 24
   105   115    32    99   104    97   114    97    99   116   101   114
  Columns 25 through 32
    32   115   116   114   105   110   103    63
```

在上面后一个例子里，加零到字符串也改变了它的 ASCII 的表示。函数 **setstr** 提供了逆转换。

```
» v=setstr(u)
v =
How about this character string?
```

因为字符串是数值数组，它们可以用 MATLAB 中所有可利用的数组操作工具进行操作。例如，

```
» u=t(16:24)
u =
character
```

字符串象数组一样进行编址。这里元素 16 到 24 包含单词 *character*

```
» u=t(24:-1:16)
u =
retcarahc
```

这是单词 *character* 的反向拼写。

```
» u=t(16:24)'
u =
c
h
a
r
a
c
```

```
t
e
r
```

用转置算子将单词 *character* 变换成一个列。

```
» v=' I can''t find the manual! '
v =
I can't find the manual!
```

字符串内的单引号是由两个连续的单引号来表示。
字符串连接可以直接从数组连接中得到。

```
» u=' If a woodchuck could chuck wood, ' ;
» v=' how much wood would a woodchuck chuck? ' ;
» w=[u    v]
w =
If a woodchuck could chuck wood, how much wood would a woodchuck chuck?
```

函数 **disp** 允许不打印它的变量名而显示一个字符串。例如，

```
»disp(u)
If a woodchuck could chuck wood,
```

注意 **u =** 语句被去掉了。这对脚本文件内显示帮助的文本有用。

如同矩阵，字符串可以有多个行，但每行必须有相同数目的列数。因此，显然要用空格以使所有行有相同长度，例如，

```
» v=[' Character strings having more    than '
      ' one row must have the same number '
      ' of column just like matrices!              ']
v =
Character strings having more than
one row must have the same number
of column just like matrices!
```

考虑下面例子，它把一个字符串转换成大写。首先，函数 **find** 用来找出小写字符的下标值，然后，从小写元素中只减去小写与大写之差，最后，用 **setstr** 把求得的数组转换成它的字符串表示。

```
» disp(u)
If a woodchuck could chuck wood,

» i=find(u>=' a ' & u<= ' z ');          %    find is a very powerful function!
```

```
» u(i)=setstr(u(i)-(' a '-' A '))
u =
IF A WOODCHUCK COULD CHUCK WOOD,
```

事实上，如在 3.7 节所描述的，矩阵能由单个下标标识。而不是由行和列下标标识，所以上面例子对字符串矩阵 **v** 也同样适用：

```
» disp(v)
Character strings having more than
one row must have the same number
of column just like matrices!

» i=find(v>=' a ' & v<=' z ');          %    here i is a single index vector into v,

» v(i)=setstr(v(i)-(' a '-' A '))          %    and matrix keeps the same orientation.
v =
CHARACTER STRINGS HAVING MORE THAN
ONE ROW MUST HAVE THE SAME NUMBER
OF COLUMN JUST LIKE MATRICES!
```

最后，当使用前面脚本文件这一章节中的函数 **input** 时，字符串是很有用的。

```
» t=' Enter number of rolls of tape > ' ;
» tape=input(t)
Enter number of rolls of tape > 5
tape =
        5
```

另外，函数 **input** 能输入一个字符串：

```
» x=input(' Enter anything > ',' s ')
Enter anything > anything can be entered
x =
anything can be entered
```

这里，在函数 **input** 里的附加参量**'s'**告诉 MATLAB，作为一个字符串，只要把用户输入传送到输出变量，就不需要引号。事实上，如果将引号包括进去，它们就变成返回字符串的一部分。

6.2 字符串转换

除了上面讨论的，字符串和它的 ASCII 表示之间转换外，MATLAB 还提供了大量的其它

的有用的字符串转换函数。它们包括：

表 6.1

字 符 串 转 换	
abs	字符串到 ASCII 转换
dec2hex	十进制数到十六进制字符串转换
fprintf	把格式化的文本写到文件中或显示屏上
hex2dec	十六进制字符串转换成十进制数
hex2num	十六进制字符串转换成 IEEE 浮点数
int2str	整数转换成字符串
lower	字符串转换成小写
num2str	数字转换成字符串
setstr	ASCII 转换成字符串
sprintf	用格式控制，数字转换成字符串
sscanf	用格式控制，字符串转换成数字
str2mat	字符串转换成一个文本矩阵
str2num	字符串转换成数字
upper	字符串转换成大写

在许多情况下，希望把一个数值嵌入到字符串中。几个字符串转换可完成这个任务。

```
» rad=2.5;   area=pi*rad^2;
» t=[' A circle of radius '   num2str(rad)   ' has an area of ' num2str(area)   ' . ' ] ;
» disp(t)
A circle of radius 2.5   has an area of   19.63.
```

这里函数 **num2str** 用来把数值转换成字符串，字符串连接用来把所转换的数嵌人到一个字符串句子中。按类似方式，**int2str** 把整数转换成字符串。无论是 **num2str** 还是 **int2str** 都调用函数 **sprintf**，它用类似 C 语言语法把数值转换成字符串。

函数 **fprintf** 经常是函数 **disp** 的一个有用替换，由于它提供了对结果更多的控制。当准备把格式化的数据写到一个文件中去时，按缺省它在**命令**窗口显示结果。例如，

```
» fprintf(' See what this does ')
See what this does»

» fprintf(' See what this does\n ')
See what this does
```

在上面第一个例子里，**fprintf** 显示字符串，然后立即给出 MATLAB 提示符。相反，在第二个例子里，**\n** 插入一个新行字符，在 MATLAB 提示符出现之前创建一个新行。

无论 **fprintf** 还是 **sprintf** 以同样方式处理输入参量，但 **fprintf** 把输出送到显示屏或文件中，而 **sprintf** 把输出返回到一个字符串中。例如，上面的例子用 **num2str** 可重写为

```
» t=sprintf(' A circle of radius %.4g has an area of %.4g. ', rad, area);
```

```
» disp(t)
A circle of radius 2.5 has an area of 19.63.

» fprintf(' A circle of radius %.4g has an area of %.4g.\n ' , rad, area)
A circle of radius 2.5 has an area of 19.63.
```

这里**%.4g** 是用在函数 **num2str** 中的数据格式。**%.4g** 就是用指数或定点标记，不管哪一种更短些，只显示至 4 位数字。除了 **g** 格式，还可用 **e** (指数)和 **f** (定点)转换。表 6.2 表明在各种不同转换下，如何显示 **pi** 结果。

表 6.2

数 值 格 式 转 换 例 子

命令	结果
fprintf(' %.0e\n ',pi)	3e+00
fprintf(' %.1e\n ',pi)	3.1e+00
fprintf(' %.3e\n ',pi)	3.142e+00
fprintf(' %.5e\n ',pi)	3.14159e+00
fprintf(' %.10e\n ',pi)	3.1415926536e+00
fprintf(' %.0f\n ',pi)	3
fprintf(' %.1f\n ',pi)	3.1
fprintf(' %.3f\n ',pi)	3.142
fprintf(' %.5f\n ',pi)	3.14159
fprintf(' %.10f\n ',pi)	3.1415926536
fprintf(' %.0g\n ',pi)	3
fprintf(' %.1g\n ',pi)	3
fprintf(' %.3g\n ',pi)	3.14
fprintf(' %.5g\n ',pi)	3.1416
fprintf(' %.10g\n ',pi)	3.141592654
fprintf(' %.8.0g\n ',pi)	3
fprintf(' %.8.1g\n ',pi)	3
fprintf(' %.8.3g\n ',pi)	3.14
fprintf(' %.8.5g\n ',pi)	3.1416
fprintf(' %.8.10g\n ',pi)	3.141592654

注意，对 **e** 和 **f** 格式，小数点右边的十进制数就是小数点右边要显示的多少位数字。相反，在 **g** 的格式里，小数点右边的十进制数指定了显示数字的总位数。另外，注意最后的五行，其结果指定为 8 个字符长度，且是右对齐。在最后一行，8 被忽略，因为指定超过了 8 位。

概括起来，当需要比缺省函数 **disp**，**num2str** 和 **int2str** 所提供的更多的控制时，**fprintf** 和 **sprintf** 是有用的。

函数 **str2mat** 把一列的几个字符串转换成一个字符串矩阵。例如，

```
» a=' one ' ; b= ' two ' ; c= ' three ' ;
» disp(str2mat(a, b, c, ' four '))
```

```
one
two
three
four
```

尽管从上面看不明显，上面的每行有同样数目的元素。较短行用空格补齐，使结果形成一个有效的矩阵

在逆方向转换中，有时是很方便的。

```
» s= ' [1    2; pi    4] '        %    a string of a MATLAB matrix
s =
[1    2; pi    4]
» str2num(s)

ans =
      1.0000       2.0000
      3.1416       4.0000

» s=' 123e+5 '          %    a string containing a simple number
s =
123e+5

» str2num(s)
ans =
      12300000
```

函数 **str2num** 不能接受用户定义的变量，也不能执行转换过程的算术运算。更多的信息，请看在线帮助。

6.3 字符串函数

MATLAB 提供了大量的字符串函数，包括列在表 6.3 当中的。

表 6.3

字　符　串　函　数	
eval(string)	作为一个 MATLAB 命令求字符串的值
eval(try,catch)	
blanks(n)	返回一个 n 个零或空格的字符串
deblank	去掉字符串中后拖的空格
feval	求由字符串给定的函数值
findstr	从一个字符串内找出字符串
isletter	字母存在时返回真值
isspace	空格字符存在时返回真值
isstr	输入是一个字符串，返回真值

lasterr	返回上一个所产生 MATLAB 错误的字符串
strcmp	字符串相同，返回真值
strrep	用一个字符串替换另一个字符串
strtok	在一个字符串里找出第一个标记

列在上面的第一个函数 **eval** 给 MATLAB 提供宏的能力。其中，该函数提供了将用户创建的函数名传给其它函数能力，以便求值。它的应用例子包括：

```
» a=eval(' sqrt(2) ')
a =
      1.4142

» eval(' a=sqrt(2) ')
a =
      1.4142
```

上面的例子演示了函数 **eval**。显然，它们不是计算 2 的平方根的最简单方法。当被求值的字符串是由子字符串连接而成，或将字符串传给一个函数以求值时，**eval** 非常有用。说明这种用途例子本书的以后会提及。

如果字符串传递到 **eval** 不能被辨认，MATLAB 提供下列语法：

```
» eval(' a=sqrtt(2) ',' a=[    ] ')
a =
      [    ]
```

这里第二个参量被执行。由于第一个参量有误，即 **sqrtt** 不是一个有效的 MATLAB 函数。这种形式经常被描述为 **eval(try,catch)**。

函数 **feval** 与 **eval** 类似，但在用法上有更多的限制。**feval(' fun ',x)**求由字符串**' fun '**给定的函数值，其输入参量是变量 x。即 **feval(' fun ',x)**等价于求 **fun(x)**值。例如，

```
» a=feval(' sqrt ' ,2)
a =
      1.4142
```

函数 **eval**，**feval** 的基本用途限在用户创建的函数内。一般地，**feval** 可求出有大量输入参量的函数值，例如，**feval(' fun ', x, y, z)** 等价于求 **fun(x, y, z)**值。

列在上面表中的许多字符串函数提供了基本的字符串语法分析能力。例如，**findstr** 返回一个在另一个字符串内字符串的起始下标值。

```
» b=' Peter Piper picked a peck of pickled peppers ' ;

» findstr(b,'    ')     %    find space
ans =
      6      12      19      21      26      29      37
```

```
» findstr(b,' p ')     %    find the letter p
ans =
       9      13      22      30      38      40      41

» find(b= = ' p ')     % for single character searches the find command works too
ans =
       9      13      22      30      38      40      41

» findstr(b, ' cow ')     %    find the word cow
ans =
       [    ]

» findstr(b,' pick ')        %    find the string pick
ans =
     13      30
```

注意这个函数对大小写是敏感的，当不匹配时，返回空矩阵。**findstr** 对字符串矩阵不起作用。

```
» strrep(b, ' p ', ' P ')          %    capitalize all p ' s
ans =
Peter PiPer Picked a Peck of Pickled PePPers

» strrep(b, ' Peter ', ' Pamela ')       %    change Peter to Pamela
ans =
Pamela Piper picked a peck of pickled peppers
```

正如上面所看到的，**strrep** 执行简单的字符串替代。**strrep** 对字符串矩阵不起作用。

函数 **strtok** 找出由特定字符指定的字符串内的标记，空格是缺省限定字符。例如，

```
» disp(b)
Peter Piper picked a peck of pickled peppers

» strtok(b)       %    find first token in above string separated by whitespace
ans =
Peter

» [c, r]=strtok(b)       %    return the remainder of the string array in r
c =
Peter
r =
Piper picked a peck of pickled peppers

» [d,s]=strtok(r) %find the next token by using the previous remainder
```

```
d =
Piper
s =
picked a peck of pickled peppers
```

用空格作为限定符，**strtok** 找出在数组中的单词。**strtok** 对字符串矩阵不起作用。

```
» [d, s]=strtok(b, ' pP ') %let delimiter be lower or upper case P
d =
eter
s =
Piper picked a peck of pickled peppers
```

如果提供一个可选的字符串，它的字符是限定符。注意在标记里，不返回限定符，但返回所有限定符之前的字符。也就是，在上面的字符串 **d = eter** 末端有一个空格。

第 7 章　决策：控 制 流

计算机编程语言和可编程计算器提供许多功能，它允许你根据决策结构控制命令执行流程。如果你以前已经使用过这些功能，对此就会很熟悉。相反，如果不熟悉控制流，本章材料初看起来或许复杂些。如果这样，就慢慢来。

控制流极其重要，因为它使过去的计算影响将来的运算。MATLAB 提供三种决策或控制流结构。它们是：For 循环，While 循环和 If-Else-End 结构。由于这些结构经常包含大量的 MATLAB 命令，故经常出现在 M 文件中，而不是直接加在 MATLAB 提示符下。

7.1　**For** 循环

For 循环允许一组命令以固定的和预定的次数重复。For 循环的一般形式是：

```
for   x = array
      {commands}
end
```

在 for 和 end 语句之间的{commands}按数组中的每一列执行一次。在每一次迭代中，**x** 被指定为数组的下一列，即在第 n 次循环中，**x=array(:, n)**。例如，

```
» for n=1:10
    x(n)=sin(n*pi/10);
```

```
 end

» x
x =
   Columns 1 through 7
     0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
   Columns 8 through 10
     0.5878    0.3090    0.0000
```

换句话，第一语句是说：对 **n** 等于 1 到 10，求所有语句的值，直至下一个 **end** 语句。第一次通过 For 循环 n=1，第二次，n=2，如此继续，直至 n=10。在 n=10 以后，For 循环结束，然后求 end 语句后面的任何命令值，在这种情况下显示所计算的 **x** 的元素。

For 循环的其它重要方面是：

1. For 循环不能用 For 循环内重新赋值循环变量 **n** 来终止。

```
» for n=1:10
     x(n)=sin(n*pi/10);
     n=10;
 end

» x
x =
   Columns 1 through 7
     0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
   Columns 8 through 10
     0.5878    0.3090    0.0000
```

2. 语句 **1 :10** 是一个标准的 MATLAB 数组创建语句。在 For 循环内接受任何有效的 MATLAB 数组。

```
» data=[3   9   45   6;   7   16   -1   5]
data =
      3      9     45      6
      7     16     -1      5

for n=data
                x=n(1)-n(2)
          end
x =
     -4
x =
     -7
x =
     46
```

```
x =
     1
```

3. For 循环可按需要嵌套。

```
for n=1:5
   for m=5:-1:1
      A(n,m)=n^2+m^2;
   end
   disp(n)
 end
     1
     2
     3
     4
     5

» A
A =
     2     5    10    17    26
     5     8    13    20    29
    10    13    18    25    34
    17    20    25    32    41
    26    29    34    41    50
```

4. 当有一个等效的数组方法来解给定的问题时，应避免用 For 循环。例如，上面的第一个例子可被重写为

```
» n=1:10;
» x=sin(n*pi/10)
x =
  Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

两种方法得出同样的结果，而后者执行更快，更直观，要求较少的输入。

5. 为了得到最大的速度，在 For 循环(While 循环)被执行之前，应预先分配数组。例如，前面所考虑的第一种情况，在 For 循环内每执行一次命令，变量 **x** 的大小增加 1。迫使 MATLAB 每通过一次循环要花费时间对 **x** 分配更多的内存。为了消去这个步骤，For 循环的例子应重写为

```
»x=zeros(1,10);       %    preallocated memory for x
» for n=1:10
```

```
        x(n)=sin(n*pi/10);
    end
```

现在，只有 **x(n)**的值需要改变。

7.2 **While** 循环

与 For 循环以固定次数求一组命令的值相反，While 循环以不定的次数求一组语句的值。While 循环的一般形式是：

```
while expression
    {commands}
end
```

只要在表达式里的所有元素为真，就执行 while 和 end 语句之间的{commands}。通常，**表达式**的求值给出一个标量值，但数组值也同样有效。在数组情况下，所得到数组的所有元素必须都为真。考虑下列例子：

```
» num=0;EPS=1;
» while (1+EPS)>1
      EPS=EPS/2;
      num=num+1;
    end

» num
num =
     53

» EPS=2*EPS
EPS =
   2.2204e-016
```

这个例子表明了计算特殊 MATLAB 值 **eps** 的一种方法，它是一个可加到 1，而使结果以有限精度大于 1 的最小数值。这里我们用大写 **EPS**，因此 MATLAB 的 **eps** 的值不会被覆盖掉。在这个例子里，**EPS** 以 1 开始。只要(1+**EPS**)>1 为真(非零)，就一直求 While 循环内的命令值。由于 **EPS** 不断地被 2 除，**EPS** 逐渐变小以致于 **EPS**+1 不大于 1。(记住，发生这种情况是因为计算机使用固定数的数值来表示数。MATLAB 用 16 位，因此，我们只能期望 **EPS** 接近 10^{-16}。)在这一点上，(1+**EPS**)>1 是假(零)，于是 While 循环结束。最后，**EPS** 与 2 相乘，因为最后除 2 使 **EPS** 太小。

7.3 **IF-ELSE-END** 结构

很多情况下，命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种 If-Else-End 结构来提供。最简单的 If-Else-End 结构是：

```
if    expression
        {commands}
end
```

如果在表达式中的所有元素为真(非零)，那么就执行 if 和 end 语言之间的{commands}。在表达式包含有几个逻辑子表达式时，即使前一个子表达式决定了表达式的最后逻辑状态，仍要计算所有的子表达式。例如，

```
» apples=10;              %    number of apples
» cost=apples*25         %    cost of apples
cost =
    250

» if apples>5             %    give 20% discount for larger purchases
     cost=(1-20/100)*cost;
 end

» cost
cost =
    200
```

假如有两个选择，If-Else-End 结构是：

```
if    expression
      commands evaluated if True
else
      commands evaluated if False
end
```

在这里，如果表达式为真，则执行第一组命令；如果表达式是假，则执行第二组命令。

当有三个或更多的选择时，If-Else-End 结构采用形式

```
if expression1
    commands evaluated if expression1 is True
elseif expression2
    commands evaluated if expression2 is True
elseif expression3
    commands evaluated if expression3 is True
elseif expression4
    commands evaluated if expression4 is True
elseif ……
```

```
        .
        .
        .
    else
        commands evaluated if no other expression    is True
    end
```

最后的这种形式，只和所碰到的、与第一个真值表达式相关的命令被执行；接下来的关系表达式不检验，跳过其余的 If-Else-End 结构。而且，最后的 else 命令可有可无。

现在我们知道了如何用 If-Else-End 结构来决策，就有可能提出一种合理的方法来跳出或中断 For 循环和 While 循环。

```
» EPS=1;
» for num=1:1000
     EPS=EPS/2;
     if (1+EPS)<=1
       EPS=EPS*2
         break
       end
   end
EPS =
   2.2204e-016

» num
num =
       53
```

这个例子演示了估算 **EPS** 的另一种方法。在这种情况下，For 循环构造成要执行足够多的次数。If-Else-End 结构检验要看 **EPS** 是否变得足够小。如果是，**EPS** 乘 2，**break** 命令强迫 For 循环提早结束，**num=53**。

在这个例子里，当执行 **break** 语句时，MATLAB 跳到循环外下一个语句。在现在情况下，它返回到 MATLAB 的提示符并显示 **EPS**。如果一个 **break** 语句出现在一个嵌套的 For 循环或 While 循环结构里，那么 MATLAB 只跳出 **break** 所在的那个循环，不跳出整个嵌套结构。

7.4 小结

MATLAB 控制流功能可以概括如下：

表 7.1

控　制　流　结　构	
for x = array 　　commands end	For 循环，每一次迭代将 x 赋以数组的第 i 列，执行命令

while expression 　　commands end	While 循环，只要表达式的所有元素为真或非零，执行命令。
if expression 　　commands end	简单的 If-Else-End 结构，若在表达式中的所有元素是真值或非零，执行命令。
if expression 　　commands evaluated if True else expression 　　commands evaluated if False end	具有两条路径的 If-Else-End 结构，若表达式为真或非零，则执行一组命令。若表达式为假或零，则执行另一组命令。
if expression1 　commands evaluated if expression1 is True elseif expression2 　commands evaluated if expression2 is True if expression3 　commands evaluated if expression3 is True elseif …… • • • else 　　commands evaluated if no other 　　expression is True end	最一般的 If-Else-End 结构。只执行与第一个真值表达式相关的命令。
break	结束 For 循环和 While 循环的执行

7.5 M 文件举例

为了演示 If-Else-End 结构，考虑**精通 MATLAB 工具箱**中函数 **mmono**，它检查一个向量的单调性。

```
» mmono(1:12)        %    strictly increasing input
ans =
       2

» mmono([1:12    12    13:24])        %    non decreasing input
ans =
       1

» mmono([1    3    2    -1])        %    not monotonic in any sense
ans =
       0
```

```
» mmono([12:-1:0    0    -1])        %    non increasing
ans =
      -1

» mmono(12:-1:0)         %    strictly decrasing
ans =
      -2
```

这个**精通 MATLAB 工具箱**的函数主体给出如下：

```
function f=mmono(x)
%    MMONO Test for monotonic vector.
%    MMONO(x) where x is a vector return:
%       2 if x is strictly increasing,
%       1 if x is non decreasing,
%      -1 if x is non increasing,
%      -2 if x is strictly decreasing,
%       0 otherwise.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

x=x(:);              %    make x a column vector
y=diff(x);           %    find differences between consecutive elements

if all(y>0)          %    test for strict first
                  f=2;
elseif all(y>=0)
                  f=1;
elseif all(y<0)      %    test for strict first
                  f=-2;
elseif all(y<=0)
                  f=-1;
else
                  f=0;      %    otherwise response
end
```

函数 **mmono** 直接利用了 If-Else-End 结构。由于严格单调是一般单调的子集，首先检验严格的单调是必要的，因为在所碰见的第一个真值分支里，其语句执行之后，结构就结束。

第 8 章　M 文 件 函 数

使用 MATLAB 函数时，例如 **inv**, **abs**, **angle** 和 **sqrt**，MATLAB 获取传递给它的变量，利用所给的输入，计算所要求的结果。然后，把这些结果返回。由函数执行的命令，以及由这些命令所创建的中间变量，都是隐含的。所有可见的东西是输入和输出，也就是说函数是一个黑箱。

这些属性使得函数成为强有力的工具，用以计算命令。这些命令包括在求解一些大的问题时，经常出现的有用的数学函数或命令序列。由于这个强大的功能，MATLAB 提供了一个创建用户函数的结构，并以 M 文件的文本形式存储在计算机上。MATLAB 函数 fliplr 是一个 M 文件函数良好的例子。

```
function y = fliplr(x)
%   FLIPLR      Flip matrix in the left/right direction.
%    FLIPLR(X) returns X with row preserved and columns flipped
%    in the left/right direction.
%
%    X = 1    2    3        becomes    3    2    1
%          4    5    6                    6    5    4
%
%    See also FLIPUD, ROT90.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

[m, n] = size(x);
y = x(: , n : -1 : 1);
```

一个函数 M 文件与脚本文件类似之处在于它们都是一个有**.m** 扩展名的文本文件。如同脚本 M 文件一样，函数 M 文件不进入**命令**窗口，而是由文本编辑器所创建的外部文本文件。一个函数的 M 文件与脚本文件在通信方面是不同的。函数与 MATLAB 工作空间之间的通信，只通过传递给它的变量和通过它所创建的输出变量。在函数内中间变量不出现在 MATLAB 工作空间，或与 MATLAB 工作空间不交互。正如上面的例子所看到的，一个函数的 M 文件的第一行把 M 文件定义为一个函数，并指定它的名字。它与文件名相同，但没有**.m** 扩展名。它也定义了它的输入和输出变量。接下来的注释行是所展示的文本，它与帮助命令： **» help fliplr** 相对应。第一行帮助行称为 H1 行，是由 **lookfor** 命令所搜索的行。最后，M 文件的其余部分包含了 MATLAB 创建输出变量的命令。

8.1 规则和属性

M 文件函数必须遵循以下特定的规则。除此之外，它们有许多的重要属性。包括：

1. 函数名和文件名必须相同。例如，函数 **fliplr** 存储在名为 **fliplr.m** 文件中。

2. MATLAB 头一次执行一个 M 文件函数时，它打开相应的文本文件并将命令编辑成存储器的内部表示，以加速执行以后所有的调用。如果函数包含了对其它 M 文件函数的引用，它们也同样被编译到存储器。普通的脚本 M 文件不被编译，即使它们是从函数 M 文件内调

用；打开脚本 M 文件，调用一次就逐行进行注释。

3. 在函数 M 文件中，到第一个非注释行为止的注释行是帮助文本。当需要帮助时，返回该文本。例如， **» help fliplr** 返回上述前八行注释。

4. 第一行帮助行，名为 H1 行，是由 **lookfor** 命令搜索的行。

5. 函数可以有零个或更多个输入参量。函数可以有零个或更多个输出参量。

6. 函数可以按少于函数 M 文件中所规定的输入和输出变量进行调用，但不能用多于函数 M 文件中所规定的输入和输出变量数目。如果输入和输出变量数目多于函数 M 文件中 **function** 语句一开始所规定的数目，则调用时自动返回一个错误。

7. 当函数有一个以上输出变量时，输出变量包含在括号内。例如， **[V,D] = eig(A)**。不要把这个句法与等号右边的**[V,D]** 相混淆。右边的**[V,D]** 是由数组 **V** 和 **D** 所组成。

8. 当调用一个函数时，所用的输入和输出的参量的数目，在函数内是规定好的。函数工作空间变量 **nargin** 包含输入参量个数；函数工作空间变量 **nargout** 包含输出参量个数。事实上，这些变量常用来设置缺省输入变量，并决定用户所希望的输出变量。例如，考虑 MATLAB 函数 **linspace** ：

```
function y = linspace(d1, d2, n)
%   LINSPACE Linearly spaced vector.
%    LINSPACE(x1, x2) generates a row vector of 100 linearly
%    equally spaced points between x1 and x2.
%    LINSPACE(x1, x2, N) generates N points between x1 and x2.
%
%    See also LOGSPACE, :.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

if nargin == 2
     n = 100;
end
y = [d1+(0:n-2)*(d2-d1)/(n-1) d2] ;
```

这里，如果用户只用两个输入参量调用 **linspace** ，例如 **linspace(0,10)** ，**linspace** 产生 100 个数据点。相反，如果输入参量的个数是 3，例如，**linspace(0,10,50)**，第三个参量决定数据点的个数。

可用一个或两个输出参量调用的函数的一个例子是 MATLAB 函数 **size**。尽管这个函数不是一个 M 文件函数(它是一个内置函数)，**size** 函数的帮助文本说明了它的输出参量的选择。

```
SIZE  Matrix dimensions.
     D = SIZE(X), for M-by-N matrix X, returns the two-element
     row vector D = [M, N] containing the number of rows and columns
     in the matrix.

     [M, N] = SIZE(X) returns the number of rows and columns
     in separate output variables.
```

如果函数仅用一个输出参量调用，就返回一个二元素的行，它包含行数和列数。相反，如果出现两个输出参量，size 分别返回行和列。在 M 文件函数里，变量 **nargout** 可用来检验输出参量的个数，并按要求修正输出变量的创建。

9. 当一个函数说明一个或多个输出变量，但没有要求输出时，就简单地不给输出变量赋任何值。MATLAB 函数 **toc** 阐明了这个属性。

```
function t = toc
%   TOC   Read the stopwatch timer.
%    TOC, by itself, prints the elapsed time since TIC was used.
%    t = TOC; saves the elapsed time in t, instead of printing it out.
%
%    See also TIC, ETIME, CLOCK, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TOC uses ETIME and the value of CLOCK saved by TIC.
global TICTOC
if nargout < 1
    elapsed_time = etime(clock, TICTOC)
else
    t = etime(clock, TICTOC);
end
```

如果用户用不以输出参量调用 **toc** ，例如， » toc，就不指定输出变量 **t** 的值，函数在**命令**窗口显示函数工作空间变量 **elapsed_time** ，但在 MATLAB 工作空间里不创建变量。相反，如果 toc 是以 **» out=toc** 调用，则按变量 **out** 将消逝的时间返回到**命令**窗口。

10. 函数有它们自己的专用工作空间，它与 MATLAB 的工作空间分开。函数内变量与 MATLAB 工作空间之间唯一的联系是函数的输入和输出变量。如果函数任一输入变量值发生变化，其变化仅在函数内出现，不影响 MATLAB 工作空间的变量。函数内所创建的变量只驻留在函数的工作空间，而且只在函数执行期间临时存在，以后就消失。因此，从一个调用到下一个调用，在函数工作空间变量存储信息是不可能的。(然而，如下所述，使用全局变量就提供这个特征。)

11. 如果一个预定的变量，例如，**pi，** 在 MATLAB 工作空间重新定义，它不会延伸到函数的工作空间。逆向有同样的属性，即函数内的重新定义变量不会延伸到 MATLAB 的工作空间中。

12. 当调用一个函数时，输入变量不会拷贝到函数的工作空间，但使它们的值在函数内可读。然而，改变输入变量内的任何值，那么数组就拷贝到函数工作空间。进而，按缺省，如果输出变量与输入变量相同，例如，函数 **x=fun(x, y, z)** 中的 **x** ，那么就将它拷贝到函数的工作空间。因此，为了节约存储和增加速度，最好是从大数组中抽取元素，然后对它们作修正，而不是使整个数组拷贝到函数的工作空间。

13. 如果变量说明是全局的,函数可以与其它函数、MATLAB 工作空间和递归调用本身共享变量。为了在函数内或 MATLAB 工作空间中访问全局变量，在每一个所希望的工作空间，变量必须说明是全局的。全局变量使用的例子可以在 MATLAB 函数 **tic** 和 **toc** 中看到，它们合在一起工作如一个跑表。

```
function tic
%   TIC    Start a stopwatch timer.
%    The sequence of commands
%           TIC
%           any stuff
%           TOC
%    prints the time required for the stuff.
%
%    See also TOC, CLOCK, ETIME, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TIC simply stores CLOCK in a global variable.
global TICTOC
TICTOC = clock;



function t = toc
%   TOC   Read the stopwatch timer.
%    TOC, by itself, prints the elapsed time since TIC was used.
%    t = TOC; saves the elapsed time in t, instead of printing it out.
%
%    See also TIC, ETIME, CLOCK, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TOC uses ETIME and the value of CLOCK saved by TIC.
global TICTOC
if nargout < 1
    elapsed_time = etime(clock,TICTOC)
else
    t = etime(clock,TICTOC);
end
```

在函数 **tic** 中，变量 **TICTOC** 说明为全局的，因此它的值由调用函数 **clock** 来设定。以后在函数 toc 中，变量 **TICTOC** 也说明为全局的，让 **toc** 访问存储在 **TICTOC** 中的值。利用这个值，**toc** 计算自执行函数 **tic** 以来消逝的时间。值得注意的是，变量 **TICTOC** 存在于 **tic** 和 **toc** 的工作空间，而不在 MATLAB 工作空间。

14. 实际编程中，无论什么时候应尽量避免使用全局变量。要是用了全局变量，建议全局变量名要长，它包含所有的大写字母，并有选择地以首次出现的 M 文件的名字开头。如果遵循建议，则在全局变量之间不必要的互作用减至最小。例如，如果另一函数或 MATLAB 工作空间说明 **TICTOC** 为全局的，那么它的值在该函数或 MATLAB 工作空间内可被改变，而函数 **toc** 会得到不同的、可能是无意义的结果。

15. MATLAB 以搜寻脚本文件的同样方式搜寻函数 M 文件。例如，输入**» cow** ，MATLAB 首先认为 **cow** 是一个变量。如果它不是，那么 MATLAB 认为它是一个内置函数。如果还不是，MATLAB 检查当前 **cow.m** 的目录或文件夹。如果它不存在， MATLAB 就检查 **cow.m** 在 MATLAB 搜寻路径上的所有目录或文件夹。如需要更多的信息，请参阅本书的 2.10 节或 MATLAB 用户指南中“MATLAB 搜寻路径”。

16. 从函数 M 文件内可以调用脚本文件。在这种情况下，脚本文件查看函数工作空间，不查看 MATLAB 工作空间。从函数 M 文件内调用的脚本文件不必用调用函数编译到内存。函数每调用一次，它们就被打开和解释。因此，从函数 M 文件内调用脚本文件减慢了函数的执行。

17. 函数可以递归调用。即 M 文件函数能调用它们本身。例如，考虑一个傻函数 **iforgot**：

```
function iforgot(n)
%   IFORGOT Recursive Function Call Example

%   Copyright (c) 1996 by Prentice-Hall,Inc

if nargin==0,n=20;end
if n>1
    disp(' I will remember to do my homework. ')
    iforgot(n-1)
else
    disp(' Maybe NOT! ')
end
```

调用这个函数产生

```
» iforgot(10)
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
Maybe NOT!
```

递归调用函数功能在许多应用场合是有用的。在编制要递归调用的函数时，必须确保会终止，否则 MATLAB 会陷入死循环。最后，在一个递归函数内，如果变量说明是全局的，则该全局变量对以后所有函数调用是可用的。在这个意义下，全局变量变成静态的，并在函数调用之间不会消失。

18. 当函数 M 文件到达 M 文件终点，或者碰到返回命令 **return**，就结束执行和返回。**return** 命令提供了一种结束一个函数的简单方法，而不必到达文件的终点。

19. MATLAB 函数 **error** 在**命令**窗口显示一个字符串，放弃函数执行，把控制权返回给键盘。这个函数对提示函数使用不当很有用，如在以下文件片段中：

```
if length(val)>1
    error('   VAL must be a scalar.   ')
end
```

这里，如果变量 **val** 不是一个标量，**error** 显示消息字符串，把控制权返回给**命令**窗口和键盘。

20. 当一个函数的输入参量的个数超出了规定的范围，MATLAB 函数 **nargchk** 提供了统一的响应。函数 **nargchk** 给定为：

```
function   msg = nargchk(low, high, number)
%   NARGCHK   Check number of input arguments.
%    Return error message if not between low and high.
%    If it is, return empty matrix.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

msg = [   ] ;
if (number < low)
    msg = ' Not enough input arguments. ' ;
elseif (number   >   high)
    msg = ' Too many input arguments. ' ;
end
```

下列的文件片段表明了在一个 M 文件函数内的典型用法：

```
  error(nargchk(nargin, 2, 5))
```

如上所示，如果 **nargin** 的值小于 2，函数 **error** 象前面描述的那样进行处理，**nargchk** 返回字符串‘没有足够的输入参量。’。如果 **nargin** 的值大于 5，函数 **error** 执行处理，**nargchk** 返回字符串‘太多输入参量。’。如果 **nargin** 是在 2 和 5 之间，函数 **error** 简单地将控制传递给下一个语句，**nargchk** 返回一个空字符串。也就是说，当它的输入参量为空，**error** 函数什么也不做。

21. 当 MATLAB 运行时，它缓存了(**caches**)存储在 **Toolbox** 子目录和 **Toolbox** 目录内的所有子目录中所有的 M 文件的名字和位置。这使 MATLAB 很快地找到和执行函数 M 文件。也使得命令 **lookfor** 工作更快。被缓存的 M 文件函数当作是只读的。如果执行这些函数，以后又发生变化，MATLAB 将只执行以前编译到内存的函数，不管已改变的 M 文件。而且，在 MATLAB 执行后，如果 M 文件被加到 Toolbox 目录中，那么它们将不出现在缓存里，因此不可利用。所以，在 M 文件函数的使用中，最好把它们存储在 **Toolbox** 目录外，或许最好存储在 MATLAB 目录下，直至它们被认为是完备的(complete)。当它们是完备时，就将它们移到一个只读的 Toolbox 目录或文件夹的子目录内。最后，要确保 MATLAB 搜索路径改变，以确认它们的存在。

22. 在 **Toolbox** 目录外，MATLAB 跟踪 M 文件的修改日期。所以，当遇到一个以前编译到内存的 M 文件函数时，MATLAB 把已编译的 M 文件的修改日期与在磁盘上的 M 文件比较。如果日期是相同的，MATLAB 执行已编译的 M 文件。相反，如果在磁盘上的 M 文件是新的，MATLAB 清除以前已编译的 M 文件，且编译这个新的和修改过的 M 文件。

23. M 文件的缓存过程按 MATLAB 版本而稍有不同。例如，MATLAB 4.2c 在 Macintosh 机上同样可以缓存当前的目录，因为这是第一个所搜索的磁盘位置。这个 MATLAB 版本也允许有选择地将整个 MATLAB 搜索路径缓存，并把高速缓存信息存储在一个文件中。这样，使 MATLAB 引导更快，寻找和编译所有函数 M 文件更快。退出缓存，不检测已修改的或已增加的 M 文件。当新的 M 文件加到一个缓存区时，只有当高速缓存由命令**» path** 刷新时，MATLAB 才能找到它们；另一方面，当修改缓存的 M 文件时，只有当以前编译过的版本由 **clear** 命令从内存中清除，MATLAB 才识别这个变化。例如，**» clear myfun**，从内存中清除 M 文件函数 **myfun**，或**» clear functions**，从内存中清除所有已编译的函数。

24. 在变量 **mfilename** 函数内，有要执行的 M 文件的名字。例如，正在执行 M 文件 **function.m** 时，函数的工作空间包含变量 **mfilename**，它包含函数字符串。这个变量也存在于脚本文件里，在这种情况下，它包含了要执行的脚本文件的名字。

25. M 文件函数可象 MATLAB 命令一样工作，典型的 MATLAB 命令包括 **clear**, **disp**, **echo**, **diary**, **save**, **hold**, **load**, **more**,和 **format**。通常，调用一个函数把参量放在括号内。例如 **size(A)**。然而，如果函数有字符串参量，那么，函数可按通常函数进行调用，如，**disp(' To be or not to be ')**，或象一个 MATLAB 命令来使用，如 **clear functions**。换句话说，当要求 MATLAB 解释一个表达式**» command argument** 时，MATLAB 认为它如同**» command(' argument ')**一样。事实上，MATLAB 命令本身能象函数那样调用！例如**» format long** 和**» format(' long ')**二者都把数据变成长格式。类似地，**» format short e** 等价于**» format(' short ' , ' e ')**。正如最后的例子所示，空格(逗号，分号)把各个命令参量分开。因此，**» disp How about this?** 产生一个错误，因为命令 **disp** 只允许一个输入参量，不是三个。如果参量包含在引号里，那么 MATLAB 就忽略空格；例如，**» disp ' How about this? '** 与**» disp(' How about this? ')**等价，并产生所希望的结果。

总之，函数 M 文件提供了一个简单的扩展 MATLAB 功能的方法。事实上，MATLAB 本身的许多标准函数就是 M 文件函数。

第 9 章　数 据 分 析

由于 MATLAB 面向矩阵，所以它很容易对数据集合进行统计分析。按规定，**数据集存储在面向列的矩阵里**。也就是，一个矩阵的每一列代表不同的被测变量，每一行代表各个样本或观察值。例如，让我们假定，一个月 31 天的三城市每日高温(单位为 ^{0}C)被记录，并赋给脚本 M 文件中的变量 **temps**，在精通 MATLAB 工具箱里取名为 **mmtemp.m**。运行 M 文件，把变量 **temps** 放在 MATLAB 工作空间里。这样，变量 **temps** 包含：

```
» temps
temps =
```

12	8	18
15	9	22
12	5	19
14	8	23
12	6	22
11	9	19
15	9	15
8	10	20
19	7	18
12	7	18
14	10	19
11	8	17
9	7	23
8	8	19
15	8	18
8	9	20
10	7	17
12	7	22
9	8	19
12	8	21
12	8	20
10	9	17
13	12	18
9	10	20
10	6	22
14	7	21
12	5	22
13	7	18
15	10	23
13	11	24
12	12	22

每一行包含了给定一天的高温；每一列包含不同城市的高温。为了使数据可视，把它绘图：

```
» d=1:31;          %    number the days of the month

» plot(d, temps)

» xlabel(' Day of Month '),ylabel(' Celsius ')

» title(' Daily High Temperatures in Three Cities ')
```

(见图 9.1)

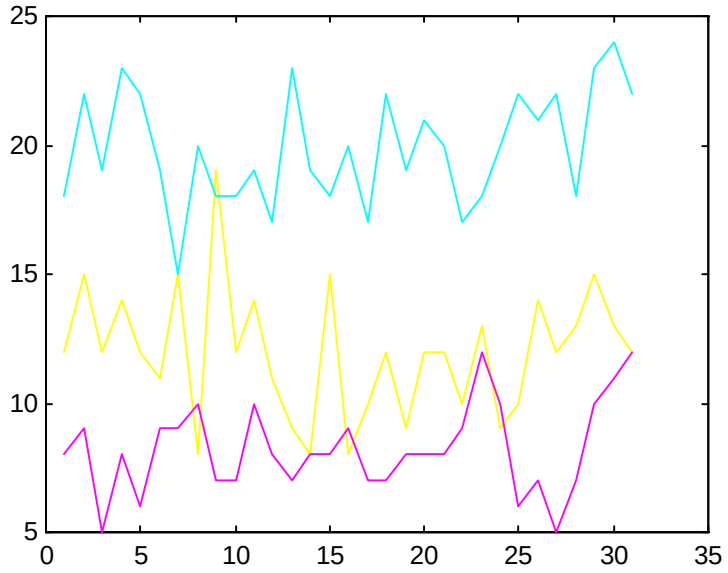


图 9.1　三个城市的每日高温

上面的 **plot** 命令也说明了 **plot** 命令用法的另一种形式。变量 **d** 是一个长度为 31 的向量，而 **temps** 是一个 31×3 矩阵。给定这些数据，**plot** 命令绘出了 **temps** 对每一列 **d** 的曲线。绘图在第 7 和 8 章进一步讨论。

为了说明 MATLAB 数据分析的一些功能，根据上面温度数据考虑以下命令。

```
» avg_temp=mean(temps)
avg_temp =
    11.9677      8.2258     19.8710
```

表明第三个城市有最高平均温度。这里 MATLAB 分别地找出了各列的平均值。

```
» avg_avg=mean(avg_temp)
avg_avg =
    13.3548
```

找出了三个城市的总平均温度。当输入到数据分析函数是行或列向量时，MATLAB 仅对向量执行运算，返回一个标量。

考虑从各城市的均值求每日偏差的问题。即必须从 **temps** 的 **i** 列中减去 **avg_temp(i)**。我们不能仅仅用以下的语句

```
» temps-avg_temp
??? Error using ==> -
Matrix dimensions must agree.
```

因为这个操作不是一个已定义的数组操作(**temps** 是 31×3 和 **avg_temp** 是 1×3)。或许最直接的方法是使用 For 循环。

```
for i=1:3
        tdev( : , i)=temps( : , i)-avg_temp(i) ;
end
```

```
» tdev
tdev =
    0.0323   -0.2258   -1.8710
    3.0323    0.7742    2.1290
    0.0323   -3.2258   -0.8710
    2.0323   -0.2258    3.1290
    0.0323   -2.2258    2.1290
   -0.9677    0.7742   -0.8710
    3.0323    0.7742   -4.8710
   -3.9677    1.7742    0.1290
    7.0323   -1.2258   -1.8710
    0.0323   -1.2258   -1.8710
    2.0323    1.7742   -0.8710
   -0.9677   -0.2258   -2.8710
   -2.9677   -1.2258    3.1290
   -3.9677   -0.2258   -0.8710
    3.0323   -0.2258   -1.8710
   -3.9677    0.7742    0.1290
   -1.9677   -1.2258   -2.8710
    0.0323   -1.2258    2.1290
   -2.9677   -0.2258   -0.8710
    0.0323   -0.2258    1.1290
    0.0323   -0.2258    0.1290
   -1.9677    0.7742   -2.8710
    1.0323    3.7742   -1.8710
   -2.9677    1.7742    0.1290
   -1.9677   -2.2258    2.1290
    2.0323   -1.2258    1.1290
    0.0323   -3.2258    2.1290
    1.0323   -1.2258   -1.8710
    3.0323    1.7742    3.1290
    1.0323    2.7742    4.1290
    0.0323    3.7742    2.1290
```

虽然使用上面的方法有效，但比使用 MATLAB 的数组操作功能要慢。复制 **avg_temp**，使得它与 **temps** 有同样的大小，然后再做减法，这样就快得多。

```
» tdev=temps-avg_temp(ones(31,1),:)
tdev =
```

```
    0.0323   -0.2258   -1.8710
    3.0323    0.7742    2.1290
    0.0323   -3.2258   -0.8710
    2.0323   -0.2258    3.1290
    0.0323   -2.2258    2.1290
   -0.9677    0.7742   -0.8710
    3.0323    0.7742   -4.8710
   -3.9677    1.7742    0.1290
    7.0323   -1.2258   -1.8710
    0.0323   -1.2258   -1.8710
    2.0323    1.7742   -0.8710
   -0.9677   -0.2258   -2.8710
   -2.9677   -1.2258    3.1290
   -3.9677   -0.2258   -0.8710
    3.0323   -0.2258   -1.8710
   -3.9677    0.7742    0.1290
   -1.9677   -1.2258   -2.8710
    0.0323   -1.2258    2.1290
   -2.9677   -0.2258   -0.8710
    0.0323   -0.2258    1.1290
    0.0323   -0.2258    0.1290
   -1.9677    0.7742   -2.8710
    1.0323    3.7742   -1.8710
   -2.9677    1.7742    0.1290
   -1.9677   -2.2258    2.1290
    2.0323   -1.2258    1.1290
    0.0323   -3.2258    2.1290
    1.0323   -1.2258   -1.8710
    3.0323    1.7742    3.1290
    1.0323    2.7742    4.1290
    0.0323    3.7742    2.1290
```

这里 **avg_temp(ones(31, 1),:)**复制 **avg_temp** 的第一行(且仅)31 次，创建了一个 31×3 的矩阵，其第 **i** 列是 **avg_temp(i)**。

```
» max_temp=max(temps)
max_temp =
    19    12    24
```

找出了每个城市一个月的最高温度。

```
» [max_temp, x]=max(temps)
max_temp =
    19    12    24
```

```
x =
     9    23    30
```

找出了每个城市的最高温度和出现最高温度的行下标 **x**。对于这个例子，当发生最高温度时，**x** 辨认了月中的日期。

```
» min_temp=min(temps)
min_temp =
     8     5    15
```

找出了各城市一个月的最低温度。

```
» [min_temp, n]=min(temps)
min_temp =
     8     5    15
n =
     8     3     7
```

找出了每个城市的最低温度和出现最低温度时行下标 **n**。对于这个例子，当发生最低温度时，**n** 辨认月中的日期。

```
» s_dev=std(temps)
s_dev =
   2.5098    1.7646    2.2322
```

找出 **temps** 的标准偏差。

```
» daily_change=diff(temps)
daily_change =
     3     1     4
    -3    -4    -3
     2     3     4
    -2    -2    -1
    -1     3    -3
     4     0    -4
    -7     1     5
    11    -3    -2
    -7     0     0
     2     3     1
    -3    -2    -2
    -2    -1     6
    -1     1    -4
     7     0    -1
    -7     1     2
```

```
     2     -2     -3
     2      0      5
    -3      1     -3
     3      0      2
     0      0     -1
    -2      1     -3
     3      3      1
    -4     -2      2
     1     -4      2
     4      1     -1
    -2     -2      1
     1      2     -4
     2      3      5
    -2      1      1
    -1      1     -2
```

计算每日高温之间的偏差，它描述了逐天日高温的变化有多大。例如，**daily_change** 的第一行是每月的第一天和第二天之间的日温度变化量。

9.1 数据分析函数

在 MATLAB 里的数据分析是按面向列矩阵而进行的。不同的变量存储在各列中，而每行表示每个变量的不同观察。MATLAB 统计函数包括

表 9.1

数 据 分 析 函 数	
corrcoef(x)	求相关系数
cov(x)	协方差矩阵
cplxpair(x)	把向量分类为复共轭对
cross(x, y)	向量的向量积
cumprod(x)	列累计积
cumsum(x)	列累计和
del2(A)	五点离散拉氏算子
diff(x)	计算元素之间差
dot(x, y)	向量的点积
gradient(Z, dx, dy)	近似梯度
histogram(x)	直方图和棒图
max(x), max(x, y)	最大分量
mean(x)	均值或列的平均值
median(x)	列的中值
min(x), min(x, y)	最小分量
prod(x)	列元素的积
rand(x)	均匀分布随机数

randn(x)	正态分布随机数
sort(x)	按升序排列
std(x)	列的标准偏差
subspace(A, B)	两个子空间之间的夹角
sum(x)	各列的元素和

9.2 M 文件举例

在这一章里，说明在**精通 MATLAB 工具箱**里的两个函数。这些函数说明了本章所示的 **min** 和 **max** 函数的变种和如何编写一个 M 文件。关于 M 文件的更多信息，参阅第 8 章。

在讨论 M 文件函数 **mmin** 和 **mmax** 的内部结构之前，考虑他们有什么功能。

```
» amn_temp=mmin(temps)
amn_temp =
     5

» [m , i]=mmin(temps)
m =
     5
i =
     3     2

» amx_temp=mmax(temps)
amx_temp =
    24

» [m , j]=mmax(temps)
m =
    24
j =
    30     3
```

具有一个输出参量的函数 **mmin** 找出矩阵中的单个最小值。用第二个输出参量，返回单个最小值的行和列的下标。除了 **mmax** 返回矩阵中的单个最大值外，函数 **mmax** 的工作方式与 **mmin** 相同。这些 M 文件的函数是：

```
function [m , i]=mmin(a)
%   MMIN Matrix minimum value.
%   MMIN(A) returns the minimum value in the matrix A
%   [M,I] = MMIN(A) in addition returns the indices of
%   the minimum value in I = [row col].

%   Copyright (c) 1996 by Prentice Hall,Inc.
```

```
if nargout==2, % return indices
      [m , i]=min(a) ;
            [m , ic]=min(m) ;
      i=[i(ic)    ic] ;
      else,
      m=min(min(a));
end


function [m , i]=mmax(a)
%    MMAX Matrix maximum value.
%    MMAX(A) returns the maximum value in the matrix A
%    [M,I] = MMAX(A) in addition returns the indices of
%    the maximum value in I = [row col].

%    Copyright (c) 1996 by Prentice Hall,Inc.

if nargout==2,              %    return indices
      [m , i]=max(a) ;
            [m , ic]=max(m) ;
      i=[i(ic)    ic] ;
      else,
      m=max(max(a)) ;
end
```

第 10 章　多 项 式

10.1　根

找出多项式的根，即多项式为零的值，可能是许多学科共同的问题，。MATLAB 求解这个问题，并提供其它的多项式操作工具。**在 MATLAB 里，多项式由一个行向量表示，它的系数是按降序排列**。例如，输入多项式 $x^4-12x^3+0x^2+25x+116$

```
» p=[1    -12    0    25    116]
p =
      1     -12       0      25     116
```

注意，必须包括具有零系数的项。除非特别地辨认，MATLAB 无法知道哪一项为零。给出这种形式，用函数 **roots** 找出一个多项式的根。

```
» r=roots(p)
r =
   11.7473
    2.7028
   -1.2251 + 1.4672i
   -1.2251 - 1.4672i
```

因为在 MATLAB 中，无论是一个多项式，还是它的根，都是向量，**MATLAB 按惯例规定，多项式是行向量，根是列向量**。给出一个多项式的根，也可以构造相应的多项式。在 MATLAB 中，命令 **poly** 执行这个任务。

```
» pp=poly(r)
pp =
   1.0e+002 *
   Columns 1 through 4
    0.0100                  -0.1200                  0.0000                  0.2500
   Column 5
    1.1600 + 0.0000i

» pp=real(pp) %throw away spurious imaginary part
pp =
      1.0000   -12.0000        0.0000       25.0000   116.0000
```

因为 MATLAB 无隙地处理复数，当用根重组多项式时，如果一些根有虚部，由于截断误差，则 poly 的结果有一些小的虚部，这是很普通的。消除虚假的虚部，如上所示，只要使用函数 real 抽取实部。

10.2 乘法

函数 **conv** 支持多项式乘法(执行两个数组的卷积)。考虑两个多项式 $a(x)=x^3+2x^2+3x+4$ 和 $b(x)= x^3+4x^2+9x+16$ 的乘积：

```
» a=[1   2   3   4] ;   b=[1   4   9   16];
» c=conv(a , b)
c =
       1       6      20      50      75      84      64
```

结果是 $c(x)=x^6+6x^5+20x^4+50x^3+75x^2+84x+64$。两个以上的多项式的乘法需要重复使用 **conv**。

10.3 加法

对多项式加法，MATLAB 不提供一个直接的函数。如果两个多项式向量大小相同，标准的数组加法有效。把多项式 a(x)与上面给出的 b(x)相加。

```
» d=a+b
d =
     2     6    12    20
```

结果是 $d(x)= 2x^3+6x^2+12x+20$。当两个多项式阶次不同，低阶的多项式必须用首零填补，使其与高阶多项式有同样的阶次。考虑上面多项式 c 和 d 相加：

```
» e=c+[0  0  0  d]
e =
     1     6    20    52    81    96    84
```

结果是 $e(x)= x^6+6x^5+20x^4+52x^3+81x^2+96x+84$。要求首零而不是尾零，是因为相关的系数象 x 幂次一样，必须整齐。

如果需要，可用一个文件编辑器创建一个函数 M 文件来执行一般的多项式加法。**精通 MATLAB 工具箱**包含下列实现：

```
function p=mmpadd(a,b)
%   MMPADD Polynomial addition.
%   MMPADD(A,B) adds the polynomial A and B

%   Copyright (c) 1996 by Prentice Hall,Inc.

if nargin<2
      error(' Not enough input arguments ')
end
a=a(:).' ;              %   make sure inputs are polynomial row vectors
b=b(:).' ;
na=length(a) ;          %   find lengths of a and b
nb=length(b) ;
p=[zeros(1,nb-na) a]+[zeros(1,na-nb) b] ;      %   add zeros as necessary
```

现在，为了阐述 **mmpadd** 的使用，再考虑前一页的例子。

```
» f=mmpadd(c,d)
f =
     1     6    20    52    81    96    84
```

它与上面的 **e** 相同。当然，**mmpadd** 也用于减法。

```
»g=mmpadd(c , -d)
g =
      1       6      20      48      69      72      44
```

结果是 $g(x)= x^6+6x^5+20x^4+48x^3+69x^2+72x+44$。

10.4 除法

在一些特殊情况，一个多项式需要除以另一个多项式。在 MATLAB 中，这由函数 **deconv** 完成。用上面的多项式 **b** 和 **c**

```
» [q , r]=deconv(c , b)
q =
      1       2       3       4
r =
      0       0       0       0       0       0       0
```

这个结果是 **b** 被 **c** 除，给出商多项式 **q** 和余数 **r**，在现在情况下 r 是零，因为 **b** 和 **q** 的乘积恰好是 **c**。

10.5 导数

由于一个多项式的导数表示简单，MATLAB 为多项式求导提供了函数 **polyder**。

```
» g
g =
      1       6      20      48      69      72      44
» h=polyder(g)
h =
      6      30      80     144     138      72
```

10.6 估值

根据多项式系数的行向量，可对多项式进行加，减，乘，除和求导，也应该能对它们进行估值。在 MATLAB 中，这由函数 **polyval** 来完成。

```
» x=linspace(-1, 3) ;          %   choose 100 data points between -1and 3.
» p=[1   4   -7   -10] ;       %   uses polynomial p(x) = x^3+4x^2—7x—10
» v=polyval(p , x) ;
```

计算 **x** 值上的 **p(x)**，把结果存在 **v** 里。然后用函数 plot 绘出结果。

```
» plot(x , v),title(' x^3+4x^2-7x-10 '),    xlabel(' x ')
```

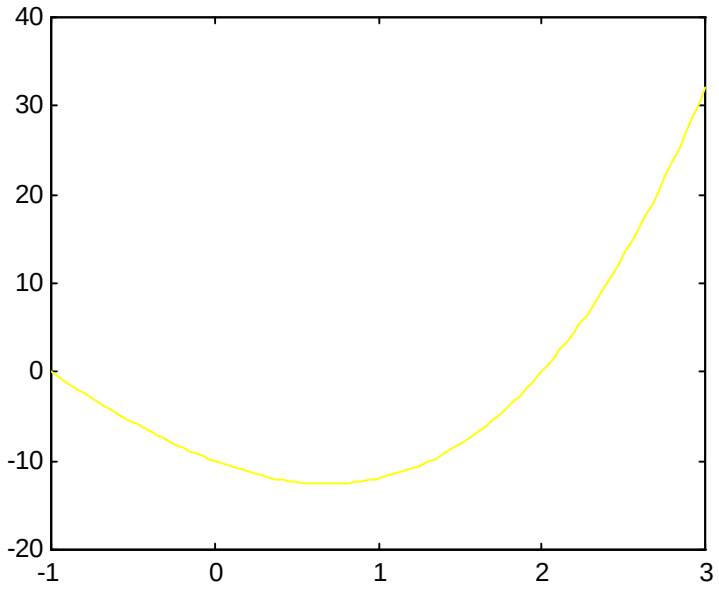


图 10.1　多项式估值

10. 7　有理多项式

在许多应用中，例如富里哀(Fourier)，拉普拉斯(Laplace)和 Z 变换，出现有理多项式或两个多项式之比。在 MATLAB 中，有理多项式由它们的分子多项式和分母多项式表示。对有理多项式进行运算的两个函数是 **residue** 和 **polyder**。函数 **residue** 执行部分分式展开。

```
» num=10*[1    2] ;                    %    numerator polynomial
» den=poly([-1;    -3;    -4]) ;      %    denominator polynomial
» [res, poles, k]=residue(num, den)
res =
    -6.6667
     5.0000
     1.6667
poles =
    -4.0000
    -3.0000
    -1.0000
k =
      [    ]
```

结果是余数、极点和部分分式展开的常数项。上面的结果说明了该问题：

$$\frac{10(s+2)}{(s+1)(s+3)(s+4)} = \frac{-6.6667}{s+4} + \frac{5}{s+3} + \frac{1.6667}{s+1} + 0$$

这个函数也执行逆运算。

```
» [n, d]=residue(res, poles, k)
n =
     0.0000     10.0000     20.0000
d =
     1.0000      8.0000     19.0000     12.0000

» roots(d)
ans =
    -4.0000
    -3.0000
    -1.0000
```

在截断误差内，这与我们开始时的分子和分母多项式一致。**residue** 也能处理重极点的情况，尽管这里没有考虑。

正如前面所述，函数 **polyder**，对多项式求导。除此之外，如果给出两个输入，则它对有理多项式求导。

```
» [b , a]=polyder(num , den)
b =
    -20   -140   -320   -260
a =
      1     16    102    328    553    456    144
```

该结果证实：

$$\frac{d}{ds}\left\{\frac{10(s+2)}{(s+1)(s+3)(s+4)}\right\} = \frac{-20s^3 - 140s^2 - 32s - 260}{s^6 + 16s^5 + 102s^4 + 328s^3 + 553s^2 + 456s + 144}$$

10.8 M 文件举例

本章说明了在**精通 MATLAB 工具箱** 中两个函数。这些函数说明了本章所论述的多项式概念和如何写 M 文件函数。关于 M 文件的更多信息，参阅第 8 章。

在讨论 M 文件函数的内部结构之前，我们考虑这些函数做些什么。

```
» n          %    earlier data
```

```
n =
       0.0000      10.0000      20.0000

» b           %    earlier data
b =
     -20    -140    -320    -260

» mmpsim(n)          %    strip away negligible leading term
ans =
     10.0000      20.0000

» mmp2str(b)         %    convert polynomial to string
ans =
-20s^3 - 140s^2 - 320s^1 - 260

» mmp2str(b , ' x ')
ans =
-20x^3 - 140x^2 - 320x^1 - 260

» mmp2str(b , [    ] , 1)
ans =
-20*(s^3 + 7s^2 + 16s^1 + 13)

» mmp2str(b , ' x ' , 1)
ans =
-20*(x^3 + 7x^2 + 16x^1 + 13)
```

这里函数 **mmpsim** 删除在多项式 **n** 中近似为零的第一个系数。函数 mmp2str 把数值多项式变换成等价形式的字符串表达式。该两个函数的主体是：

```
function y=mmpsim(x,tol)
%    MMPSIM Polynomial Simplification,Strip Leading Zero Terms.
%    MMPSIM(A) Delets leading zeros and small coefficients in the
%    polynomial A(s). Coefficients are considered small if their
%    magnitude is less than both one and norm(A)*1000*eps.
%    MMPSIM(A,TOL) uses TOL for its smallness tolerance.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

if nargin<2, tol=norm(x)*1000*eps; end
x=x(:).' ;                              %    make sure input is a row
i=find(abs(x)<.99&abs(x)<tol) ;       %    find insignificant indices
x(i)=zeros(1, length(i)) ;              %    set them to zero
i=find(x~=0) ;                          %    find significant indices
```

```
if isempty(i)
        y=0 ;                                %    extreme case: nothing left!
else
       y=x(i(1) : length(x)) ;               %    start with first term
end                                          %    and go to end of polynomial


function s=mmp2str(p,v,ff)
%    MMP2STR Polynomial Vector to String Conversion.
%    MMP2STR(P) converts the polynomial vector P into a string.
%    For example: P = [2    3    4] becomes the string ' 2s^2 + 3s + 4 '
%
%    MMP2STR(P,V) generates the string using the variable V
%    instead of s. MMP2STR([2    3    4],' z ') becomes ' 2z^2 + 3z + 4 '
%
%    MMP2STR(P,V,1) factors the polynomial into the product of a
%    constant and a monic polynomial.
%    MMP2STR([2    3    4],[    ],1) becomes ' 2(s^2 + 1.5s + 2) '

%    Copyright (c) 1996 by Prentice-Hall,Inc.

if nargin<3, ff=0; end                       %    factored form is False
if nargin <2, v=' s ' ; end                  %    default variable is ' s '
if isempty(v), v=' s ' ; end                 %    default variable is ' s '
v=v(1) ;                                     %    variable must be scalar
p=mmpsim(p) ;                                %    strip insignificant terms
n=length(p) ;
if ff                                        %    put in factored form
       K=p(1) ; Ka=abs(K) ; p=p/K;
       if abs(K-1)<1e-4
            pp=[    ]; pe=[    ] ;
       elseif abs(K+1)<1e-4
            pp=' -(' ; pe= ') ' ;
       elseif abs(Ka-round(Ka))<=1e-5*Ka
            pp=[sprintf(' %.0f ', K) '*( ' ] ; pe= ') ' ;
       else
            pp=[sprintf(' %.4g ' , K) '*(' ] ; pe= ') ' ;
       end
else                                         %    not factored form
       K=p(1);
       pp=sprintf(' %.4g ' , K) ;
       pe=[    ];
end
if n==1                                      %    polynomial is just a constant
```

```
        s=sprintf(' %.4g ',K);
        return
end
s=[pp v ' ^ ' sprintf(' %.0f ',n-1)];        %    begin string construction

for i=2:n-1                                  %    catenate center terms in polynomial
    if p(i)<0, pm= ' -    ' ;    else,    if p(i)<0,pm= ' ;    end
    if p(i)= =1,pp=[    ] ; else,    pp=sprintf(' %.4g ', abs(p(i))) ;    end
    if p(i)~ =0,s=[s    pm    pp    v    ' ^ ' sprintf(' %.0f ',n-i)] ;    end
end

if p(n)~ =0,pp=sprintf(' %.4g ',abs(p(n))); else,    pp=[    ];    end
if p(n)<0 , pm= ' -    ' ; elseif    p(n)>0 , pm= ' +    ' ; else,    pm=[    ];    end
s=[s    pm    pp    pe];                     %    add final terms
```

10.9 小结

下列表格概括了在本章所讨论的多项式操作特性。

表 10.1

多　项　式　函　数	
conv(a, b)	乘法
[q, r]=deconv(a, b)	除法
poly(r)	用根构造多项式
polyder(a)	对多项式或有理多项式求导
polyfit(x, y, n)	多项式数据拟合
polyval(p, x)	计算 x 点中多项式值
[r, p, k]=residue(a, b)	部分分式展开式
[a, b]=residue(r, p, k)	部分分式组合
roots(a)	求多项式的根

表 10.2

精 通 **MATLAB** 多 项 式 操 作	
mmp2str(a)	多项式向量到字符串变换，**a(s)**
mmp2str(a, ' x ')	多项式向量到字符串变换，**a(x)**
mmp2str(a, ' x ', 1)	常数和符号多项式变换
mmpadd(a, b)	多项式加法
mmpsim(a)	多项式简化

第 11 章　曲线拟合与插值

在大量的应用领域中，人们经常面临用一个解析函数描述数据(通常是测量值)的任务。对这个问题有两种方法。在插值法里，数据假定是正确的，要求以某种方法描述数据点之间所发生的情况。这种方法在下一节讨论。这里讨论的方法是曲线拟合或回归。人们设法找出某条光滑曲线，它最佳地拟合数据，但不必要经过任何数据点。图 11.1 说明了这两种方法。标有**'o'**的是数据点；连接数据点的实线描绘了线性内插，虚线是数据的最佳拟合。

11.1　曲线拟合

曲线拟合涉及回答两个基本问题：**最佳拟合**意味着什么？应该用什么样的曲线？可用许多不同的方法定义**最佳拟合**，并存在无穷数目的曲线。所以，从这里开始，我们走向何方？正如它证实的那样，当**最佳拟合**被解释为在数据点的最小误差平方和，且所用的曲线限定为多项式时，那么曲线拟合是相当简捷的。数学上，称为多项式的最小二乘曲线拟合。如果这种描述使你混淆，再研究图 11.1。虚线和标志的数据点之间的垂直距离是在该点的误差。对各数据点距离求平方，并把平方距离全加起来，就是误差平方和。这条虚线是使误差平方和尽可能小的曲线，即是**最佳拟合**。最小二乘这个术语仅仅是使误差平方和最小的省略说法。

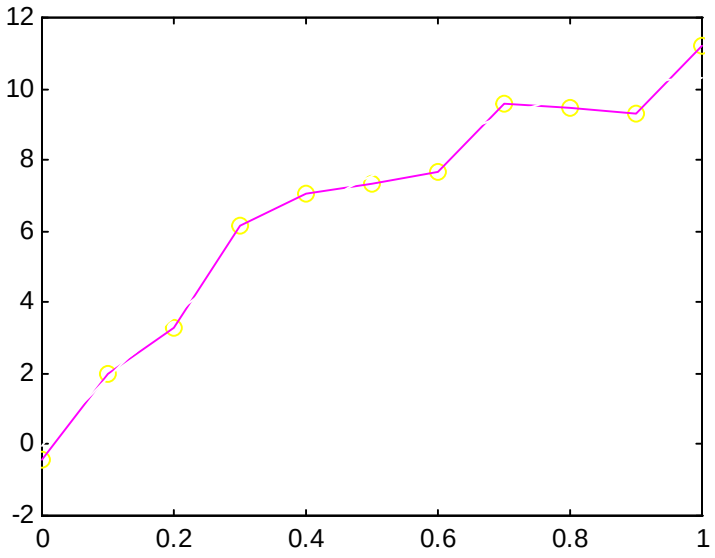


图 11.1　2 阶曲线拟合

在 MATLAB 中，函数 **polyfit** 求解最小二乘曲线拟合问题。为了阐述这个函数的用法，让我们以上面图 11.1 中的数据开始。

```
» x=[0   .1   .2   .3   .4   .5   .6   .7    .8   .9   1];
```

```
» y=[-.447    1.978    3.28    6.16    7.08    7.34    7.66    9.56    9.48    9.30    11.2];
```

为了用 **polyfit**，我们必须给函数赋予上面的数据和我们希望最佳拟合数据的多项式的阶次或度。如果我们选择 **n=1** 作为阶次，得到最简单的线性近似。通常称为线性回归。相反，如果我们选择 **n=2** 作为阶次，得到一个 2 阶多项式。现在，我们选择一个 2 阶多项式。

```
» n=2;                    %    polynomial order

» p=polyfit(x, y, n)
p =
    -9.8108      20.1293      -0.0317
```

polyfit 的输出是一个多项式系数的行向量。其解是 y = －9.8108x^2 ＋20.1293x－0.0317。为了将曲线拟合解与数据点比较，让我们把二者都绘成图。

```
» xi=linspace(0, 1, 100);        %    x-axis data for plotting
» z=polyval(p, xi);
```

为了计算在 **xi** 数据点的多项式值，调用 MATLAB 的函数 **polyval**。

```
» plot(x, y, ' o ' , x, y, xi, z, ' : ' )
```

画出了原始数据 **x** 和 **y**，用**'o'**标出该数据点，在数据点之间，再用直线重画原始数据，并用点**' : '**线，画出多项式数据 **xi** 和 **z**。

```
» xlabel(' x '), ylabel(' y=f(x) '), title(' Second Order Curve Fitting ')
```

将图作标志。这些步骤的结果表示于前面的图 11.1 中。

多项式阶次的选择是有点任意的。两点决定一直线或一阶多项式。三点决定一个平方或 2 阶多项式。按此进行，**n+1** 数据点唯一地确定 **n** 阶多项式。于是，在上面的情况下，有 11 个数据点，我们可选一个高达 10 阶的多项式。然而，高阶多项式给出很差的数值特性，人们不应选择比所需的阶次高的多项式。此外，随着多项式阶次的提高，近似变得不够光滑，因为较高阶次多项式在变零前，可多次求导。例如，选一个 10 阶多项式

```
» pp=polyfit(x, y, 10) ;
» format short e              %    change display format

» pp.'                        %    display polynomial coefficients as a column
ans =
  -4.6436e+005
    2.2965e+006
  -4.8773e+006
    5.8233e+006
  -4.2948e+006
```

```
  2.0211e+006
 -6.0322e+005
  1.0896e+005
 -1.0626e+004
  4.3599e+002
 -4.4700e-001
```

要注意在现在情况下，多项式系数的规模与前面的 2 阶拟合的比较。还要注意在最小(-4.4700e-001)和最大(5.8233e+006)系数之间有 7 个数量级的幅度差。将这个解作图，并把此图与原始数据及 2 阶曲线拟合相比较，结果如何呢？

```
» zz=polyval(pp, xi);                    %   evaluate 10th order polynomial
» plot(x, y, ' o ' , xi, z, ' : ' , xi, zz)      %   plot data
» xlabel(' x '),   ylabel(' y=f(x) '),   title(' 2nd and 10th Order curve Fitting ')
```

在下面的图 11.2 中，原始数据标以**'o'**，2 阶曲线拟合是虚线，10 阶拟合是实线。注意，在 10 阶拟合中，在左边和右边的极值处，数据点之间出现大的纹波。当企图进行高阶曲线拟合时，这种纹波现象经常发生。根据图 11.2，显然，‘ 越多就越好 ’ 的观念在这里不适用。

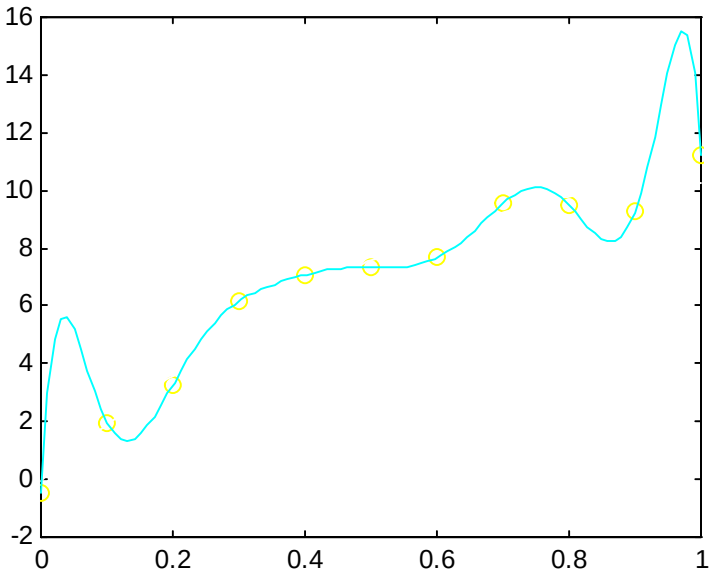


图 11.2　2 阶和 10 阶曲线拟合

11.2　一维插值

正如在前一节对曲线拟合所描述的那样，插值定义为对数据点之间函数的估值方法，这些数据点是由某些集合给定。当人们不能很快地求出所需中间点的函数值时，插值是一个有

价值的工具。例如，当数据点是某些实验测量的结果或是过长的计算过程时，就有这种情况。

或许最简单插值的例子是 MATLAB 的作图。按缺省，MATLAB 用直线连接所用的数据点以作图。**这个线性插值猜测中间值落在数据点之间的直线上**。当然，当数据点个数的增加和它们之间距离的减小时，线性插值就更精确。例如，

```
» x1=linspace(0, 2*pi, 60);

» x2=linspace(0, 2*pi, 6);

» plot(x1, sin(x1), x2, sin(x2), ' - ')

» xlabel(' x '),    ylabel(' sin(x) '),    title(' Linear Interpolation ')
```

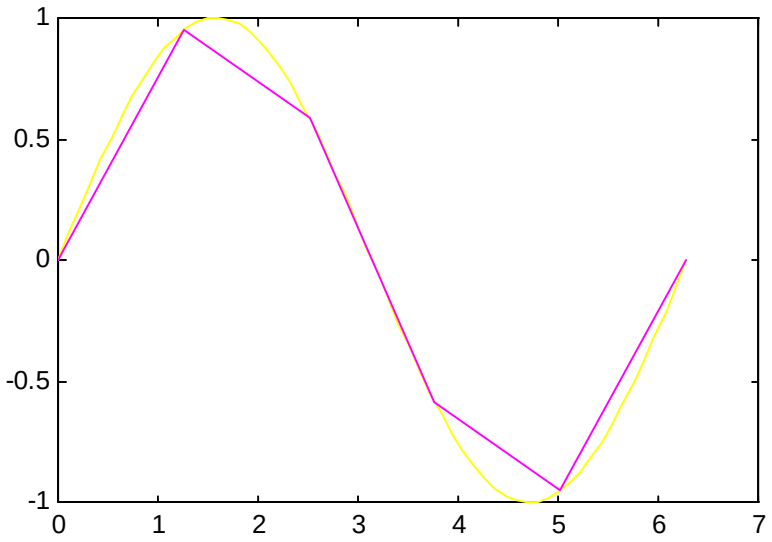


图 11.3　线性插值

图 11.3 是 sine 函数的两个图，一个在数据点之间用 60 个点，它比另一个只用 6 个点更光滑和更精确。

如曲线拟合一样，插值要作决策。根据所作的假设，有多种插值。而且，可以在一维以上空间中进行插值。即如果有反映两个变量函数的插值，**z=f(x, y)**，那么就可在 x 之间和在 y 之间，找出 z 的中间值进行插值。MATLAB 在一维函数 **interp1** 和在二维函数 **interp2** 中，提供了许多的插值选择。其中的每个函数将在下面阐述。

为了说明一维插值，考虑下列问题，12 小时内，一小时测量一次室外温度。数据存储在两个 MATLAB 变量中。

```
» hours=1:12;          %    index for hour data was recorded

» temps=[5    8    9    15    25    29    31    30    22    25    27    24]; %    recorded temperatures
```

```
» plot(hours, temps, hours, temps,' + ')        %    view temperatures

» title(' Temperature ')

» xlabel(' Hour '),    ylabel(' Degrees Celsius ')
```

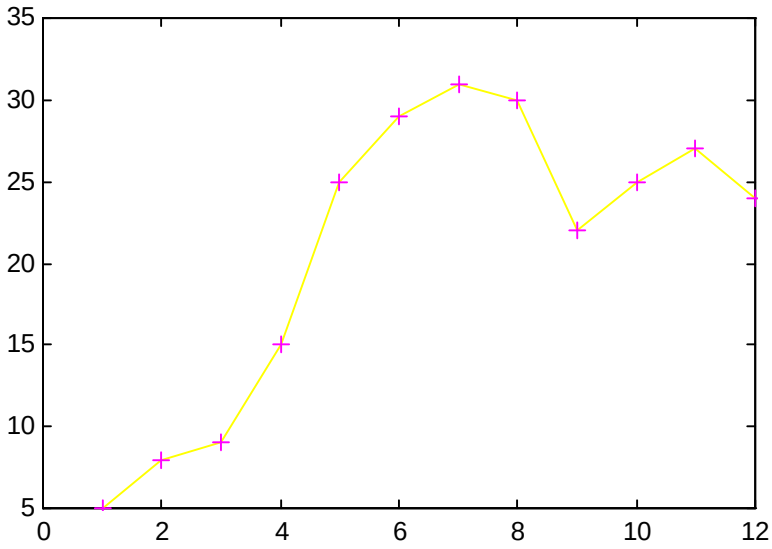


图 11.4　在线性插值下室外温度曲线

正如图 11.4 看到的，MATLAB 画出了数据点线性插值的直线。为了计算在任意给定时间的温度，人们可试着对可视的图作解释。另外一种方法，可用函数 **interp1**。

```
» t=interp1(hours, temps, 9.3)              %    estimate temperature at hour=9.3
t =
    22.9000

» t=interp1(hours, temps, 4.7)              %    estimate temperature at hour=4.7
t =
     22

» t=interp1(hours, temps, [3.2    6.5    7.1    11.7])        %    find temp at many points!
t =
    10.2000
    30.0000
    30.9000
    24.9000
```

interp1 的缺省用法是由 **interp1(x, y, xo)**来描述，这里 **x** 是独立变量(横坐标)，**y** 是应变

量(纵坐标)，**xo** 是进行插值的一个数值数组。另外，该缺省的使用假定为线性插值。

若不采用直线连接数据点，我们可采用某些更光滑的曲线来拟合数据点。最常用的方法是用一个 3 阶多项式，即 3 次多项式，来对相继数据点之间的各段建模，每个 3 次多项式的头两个导数与该数据点相一致。这种类型的插值被称为 **3 次样条**或简称为**样条**。函数 **interp1** 也能执行 3 次样条插值。

```
» t=interp1(hours, temps, 9.3, ' spline ')        %    estimate temperature at hour=9.3
t =
    21.8577

» t=interp1(hours, temps, 4.7, ' spline ')        %    estimate temperature at hour=4.7
t =
    22.3143

» t=interp1(hours, temps, [3.2    6.5    7.1    11.7], ' spline ')
t =
     9.6734
    30.0427
    31.1755
    25.3820
```

注意，样条插值得到的结果，与上面所示的线性插值的结果不同。因为插值是一个估计或猜测的过程，其意义在于，应用不同的估计规则导致不同的结果。

一个最常用的样条插值是对数据平滑。也就是，给定一组数据，使用样条插值在更细的间隔求值。例如，

```
» h=1:0.1:12;              %    estimate temperature every 1/10 hour

» t=interp1(hours, temps, h, ' spline ') ;

» plot(hours, temps, ' - ' , hours, temps, ' + ' , h, t)        %    plot comparative results

» title(' Springfield Temperature ')

» xlabel(' Hour '),    ylabel(' Degrees Celsius ')
```

在图 11.5 中，虚线是线性插值，实线是平滑的样条插值，标有**' + '**的是原始数据。如要求在时间轴上有更细的分辨率，并使用样条插值，我们有一个更平滑、但不一定更精确地对温度的估计。尤其应注意，在数据点，样条解的斜率不突然改变。作为这个平滑插值的回报，3 次样条插值要求更大量的计算，因为必须找到 3 次多项式以描述给定数据之间的特征。关于样条的更详细信息可见下一章。

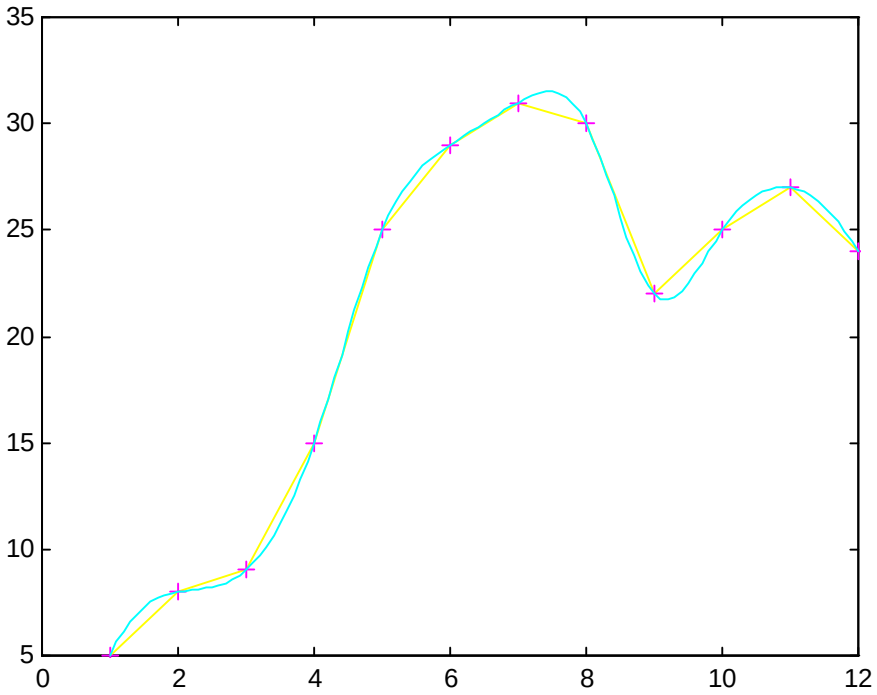


图 11.5　在不同插值下室外温度曲线

在讨论二维插值之前，了解 **interp1** 所强制的二个强约束是很重要的。首先，人们不能要求有独立变量范围以外的结果，例如，**interp1(hours, temps, 13.5)**导致一个错误，因为 **hours** 在 1 到 12 之间变化。其次，独立变量必须是单调的。即独立变量在值上必须总是增加的或总是减小的。在我们的例子里，**hours** 是单调的。然而，如果我们已经定义独立变量为一天的实际时间，

```
» time_of_day=[7:12     1:6]          %    start at 7AM,end at 6PM
time_of_day =
     7     8     9     10     11     12     1     2     3     4     5     6
```

则独立变量将不是单调的，因为 **time_of_day** 增加到 12，然后跌到 1，再然后增加。如果用 **time_of_day** 代替 **interp1** 中的 **hours**，将会返回一个错误。同样的理由，人们不能对 **temps** 插值来找出产生某温度的时间(小时)，因为 **temps** 不是单调的。

11.3　二维插值

二维插值是基于与一维插值同样的基本思想。然而，正如名字所隐含的，二维插值是对两变量的函数 **z=f(x, y)** 进行插值。为了说明这个附加的维数，考虑一个问题。设人们对平板上的温度分布估计感兴趣，给定的温度值取自平板表面均匀分布的格栅。

采集了下列的数据：

```
» width=1:5;                    %    index for width of plate (i.e.,the x-dimension)
```

```
» depth=1:3;                    %    index for depth of plate (i,e,,the y-dimension)

» temps=[82    81    80    82    84; 79    63    61    65    81; 84    84    82    85    86]    %
temperature data
temps =
     82     81     80     82     84
     79     63     61     65     81
     84     84     82     85     86
```

如同在标引点上测量一样，矩阵 **temps** 表示整个平板的温度分布。**temps** 的列与下标 **depth** 或 y-维相联系，行与下标 **width** 或 x-维相联系(见图 11.6)。为了估计在中间点的温度，我们必须对它们进行辨识。

```
» wi=1:0.2:5;          %    estimate across width of plate

» d=2;                 %    at a depth of 2

» zlinear=interp2(width, depth, temps, wi, d) ;                %    linear interpolation

» zcubic=interp2(width, depth, temps, wi,d, ' cubic ') ;       %    cubic interpolation

» plot(wi, zlinear, ' - ' , wi, zcubic)         %    plot results

» xlabel(' Width of Plate '),    ylabel(' Degrees Celsius ')

» title( [' Temperature at Depth =    '    num2str(d) ] )
```

另一种方法，我们可以在两个方向插值。先在三维坐标画出原始数据，看一下该数据的粗糙程度(见图 11.7)。

```
» mesh(width, depth, temps)                %    use mesh plot

» xlabel(' Width of Plate '),    ylabel(' Depth of Plate ')

» zlabel(' Degrees Celsius '),    axis(' ij '),    grid
```

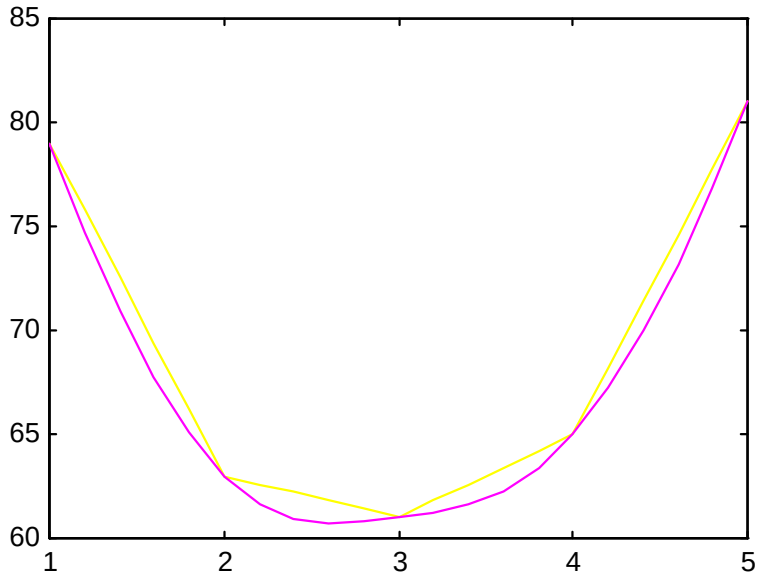


图 11.6　在深度 d=2 处的平板温度

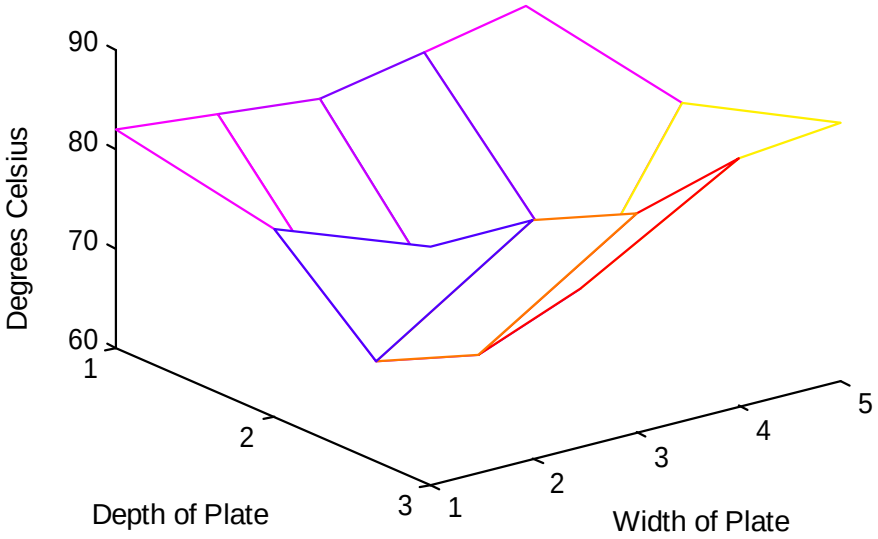


图 11.7　平板温度

然后在两个方向上插值，以平滑数据。

» di=1:0.2:3;　　%　choose higher resolution for depth

» wi=1:0.2:5;　　%　choose higher resolution for width

» zcubic=interp2(width, depth, temps, wi, di, ' cubic ') ;　　%　cubic

» mesh(wi, di, zcubic)

» xlabel(' Width of Plate '), ylabel(' Depth of Plate ')

» zlabel(' Degrees Celsius '), axis(' ij '), grid

上面的例子清楚地证明了，二维插值更为复杂，只是因为有更多的量要保持跟踪。**interp2** 的基本形式是 **interp2(x, y, z, xi, yi, method)**。这里 **x** 和 **y** 是两个独立变量，**z** 是一个应变量矩阵。**x** 和 **y** 对 **z** 的关系是

z(i, :) = f(x, y(i)) 和 z(:, j) = f(x(j), y).

也就是，当 **x** 变化时，**z** 的第 **i** 行与 **y** 的第 **i** 个元素 **y(i)**相关，当 **y** 变化时，**z** 的第 **j** 列与 **x** 的第 **j** 个元素 **x(j)**相关，。**xi** 是沿 x-轴插值的一个数值数组；**yi** 是沿 y-轴插值的一个数值数组。

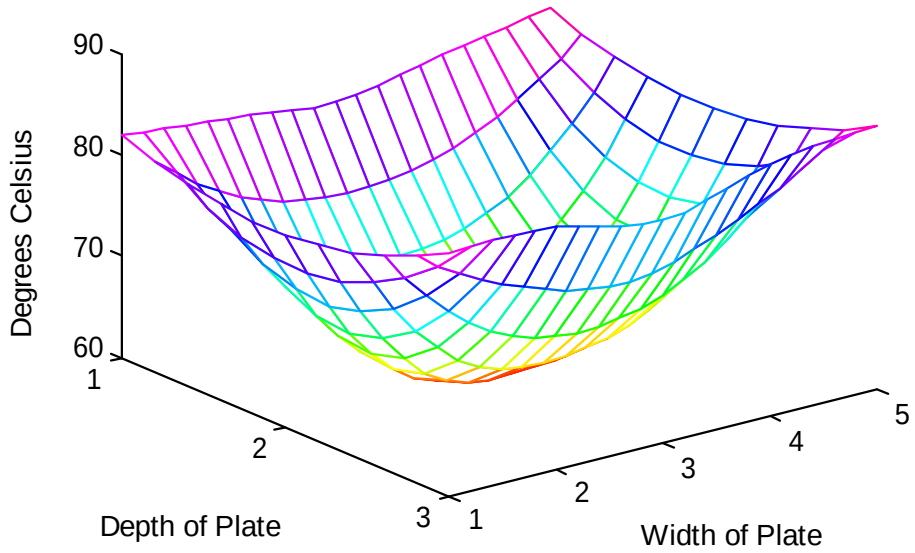


图 11.8　二维插值后的平板温度

可选的参数 **method** 可以是 **'linear'**，**'cubic'**或**'nearest'**。在这种情况下，**cubic** 不意味着 3 次样条，而是使用 3 次多项式的另一种算法。**linear** 方法是线性插值，仅用作连接图上数据点。**nearest** 方法只选择最接近各估计点的粗略数据点。在所有的情况下，假定独立变量 **x** 和 **y** 是线性间隔和单调的。关于这些方法的更多的信息，可请求在线帮助，例如，**» help interp2**，或参阅 MATLAB 参考手册。

11.4　M 文件举例

虽然对于许多应用，函数 **interp1** 和 **interp2** 是很有用的，但它们限制为对单调向量进行插值。在某些情况，这个限制太严格。例如，考虑下面的插值：

```
» x=linspace(0, 5);

» y=1-exp(-x).*sin(2*pi*x);
» plot(x, y)
```

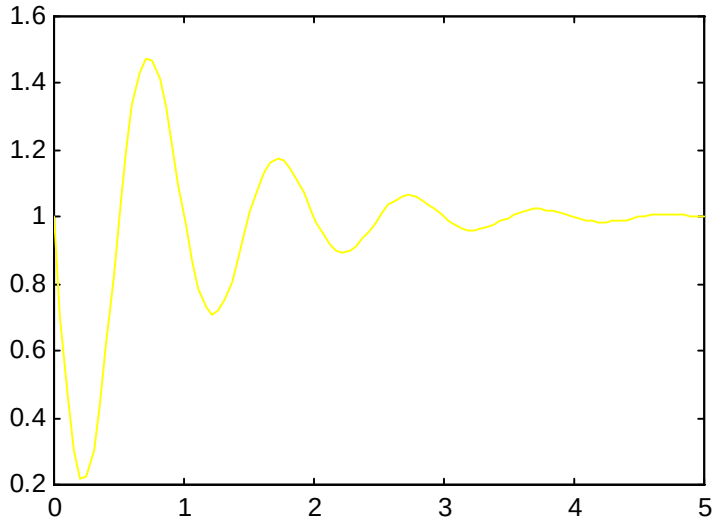


图 11.9　函数 1-exp(-x).*sin(2*pi*x)的曲线

函数 **interp1** 可用来在任何值或 **x** 的值上估计 **y** 值。

```
» yi=interp1(x, y, 1.8)
yi =
    1.1556
```

然而，**interp1** 不能找出对应于某些 **y** 值的 **x** 值。例如，如在图 11.9 上所示，考虑寻找 **y=1.1** 处的 **x** 值：

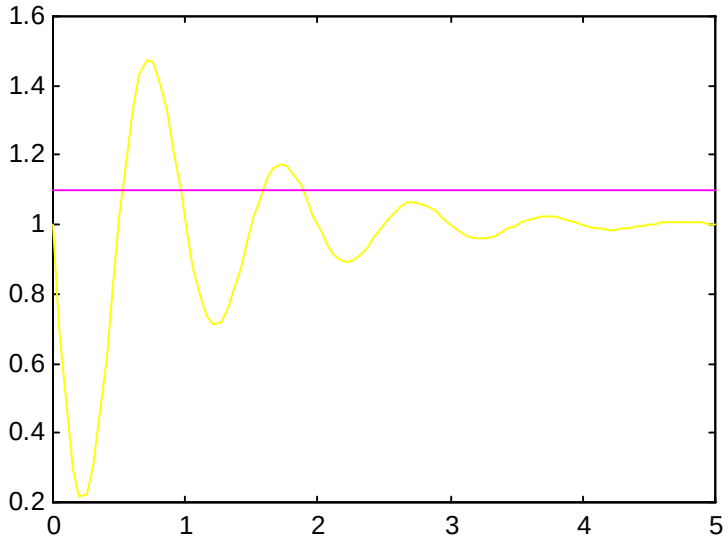


图 11.10　给 y 值在函数曲线上求 x 的值

```
» plot(x, y, [0, 5], [1.1       1.1] )
```

从图 11.10 上，我们看到有四个交点。使用 **interp1**，我们得到：

```
» xi=interp1(y, x, 1.1)
??? Error using ==> table1
First column of the table must be monotonic.
```

这个函数 **interp1** 失败，由于 **y** 不是单调的。
在本章**精通 MATLAB 工具箱**所说明的 M 文件例子，消除了单调性的要求。

```
» table=[x; y].' ;          %    create column oriented table from data

» xi=mminterp(table, 2, 1.1)
xi =
      0.5281      1.1000
      0.9580      1.1000
      1.5825      1.1000
      1.8847      1.1000
```

这里使用了线性插值，函数 **mminterp** 估计了 **y=1.1** 处的四个点。由于函数 **mminterp** 的一般性质，要插值的数据是由面向列矩阵给出，在上面的例子中称作为表(**table**)。第二个输入参量是被搜索矩阵 **table** 的列，第三个参量是要找的值。

这个**精通 MATLAB 工具箱**函数的主体由下面给出：

```
function y=mminterp(tab, col, val)
%    MMINTERP 1-D Table Search by Linear Interpolation.
```

```
%    Y=MMINTERP(TAB,COL,VAL) linearly interpolates the table
%    TAB searching for the scalar value VAL in the column COL.
%    All crossings are found and TAB(:,COL) need not be monotonic.
%    Each crossing is returned as a separate row in Y and Y has as
%    many columns as TAB.Naturally,the column COL of Y contains
%    the value VAL. If VAL is not found in the table,Y=[].

%    Copyright (c) 1996 by Prentice-Hall,Inc.

[rt, ct]=size(tab);
if length(val) > 1,    error(' VAL must be a scalar. '),    end
if col>ct|col < 1,    error(' Chosen column outside table width. '),    end
if rt < 2,    error(' Table too small or not oriented in columns. '),    end

above=tab(: , col) > val;              %    True where > VAL
below=tab(: , col) < val;              %    True where < VAL
equal=tab(: , col) = = val;            %    True where = VAL

if all(above = = 0) | all(below = = 0),          %    handle simplest case
      y=tab(find(equal), : );    return
end
pslope=find(below(1:rt-1)&above(2:rt));     %    indices where slope is +
nslope=find(below(2:rt)&above(1:rt-1));     %    indices where slope is -

ib=sort([pslope; nslope+1]);         %    put indices below in order
ia=sort([nslope; pslope+1]);         %    put indices above in order
ie=find(equal);                       %    indices where equal to val

[tmp,ix]=sort( [ib, ie] );             %    find where equals fit in result
ieq=ix > length(ib);                  %    True where equals values fit
ry=length(tmp);                       %    # of rows in result y

y=zeros(ry, ct);          %    poke data into a zero matrix

alpha=(val-tab(ib,col))./(tab(ia,col)-tab(ib,col));
alpha=alpha(: , ones(1, ct));         %    duplicate for all columns
y(~ieq, : )=alpha.*tab(ia, : )+(1-alpha).*tab(ib, : );         %    interpolated values

y(ieq, : )=tab(ie, : );       %    equal values
y( : , col)=val*ones(ry, 1);       %    remove roundoff error
```

正如所见的，**mminterp** 利用了 **find** 和 **sort** 函数、逻辑数组和数组操作技术。**没有 For 循环和 While 循环**。不论用其中哪一种技术来实现将使运行变慢，尤其对大的表。注意

mminterp 与含有大于或等于 2 的任意数列的表一起工作，如同函数 **interp1** 一样。而且，在这种情况下，插值变量可以是任意的列。例如，

```
» z=sin(pi*x);            %    add more data to table

» table=[x; y; z].' ;

» t=mminterp(table, 2, 1.1)        %    same interpolation as earlier
t =
      0.5281      1.1000      0.9930
      0.9580      1.1000      0.1314
      1.5825      1.1000     -0.9639
      1.8847      1.1000     -0.3533

» t=mminterp(table, 3, -.5)         %    second third column now
t =
      1.1669      0.7316     -0.5000
      1.8329      1.1377     -0.5000
      3.1671      0.9639     -0.5000
      3.8331      1.0187     -0.5000
```

这些最后的结果估计了 **x** 和 **y** 在 **z= -.5** 处的值。

尽管逐条地对函数 **mminterp** 解释如何工作是很有帮助的，但这样做要求有更多的篇幅和时间。解释 **mminterp** 如何工作最容易的方法是创建一个小表格，然后，在重要的语句末尾删除分号以后，调用函数。这样，中间值将帮助用户理解函数是如何找到与所需值相符的数据值以及如何执行插值。

前面已阐述了 **interp1** 的用法。当用于线性插值时，只要所要求的插值点的个数少，**interp1** 工作很好。在要求许多插值点情况下，由于所用的算法，**interp1** 工作较慢。为了克服这个问题，**精通 MATLAB 工具箱**包括了函数 **mmtable**，它的帮助文本是：

```
»help table
MMTABLE 1-D Monotonic Table Search by Linear Interpolation.
   YI=MMTABLE(TAB,COL,VALS) linearly interpolates the table TAB
   searching for values VALS in the column COL.

   TAB(:,COL) must be monotonic, but need NOT be equally spaced.
   YI has as many rows as VALS and as many columns TAB
   NaNs are returned where VALS are outside the range of TAB(:,COL).

   YI=MMTABLE(TAB,VALS) interpolates using COL=1 and does not return
   TAB(:,1) in Y.    This matches the usage of TABLE1(TAB,X0).

   YI=MMTABLE(X,Y,XI) interpolates the vector X to find YI associated
with XI. This match the usage of INTERP1(X,Y,XI)
```

This routine is 10X faster than TABLE1which is called by INTERP1.

MMTABLE 由线性插值实现一维单调表搜索
YI=MMTABLE(TAB,COL,VALS) 线性地对表 TAB 进行插值，在列 COL 中搜索值为 VALS

TAB(:,COL)必须是单调的，但不必等价地生成空间。
YI 与 VALS 有同样的行和与 TAB 有同样的列。
当 VALS 超出 TAB(:,COL)的范围，返回 NaNs.

YI=MMTABLE(TAB,VALS) 使用 COL=1 进行插值，不返回在 Y 中的 TAB(:,1)
这和 TABLE1(TAB,X0)的用法匹配。

YI=MMTABLE(X,Y,XI) 为了找出 YI 和 XI 的关系，对向量 X 进行插值。
这和 INTERP1(X,Y,XI)的用法匹配。

这个例程比由 INTERP1 调用 TABLE1 快 10 倍。

正如前面描述的，可以用几种方式调用 **mmtable**。此外，要插值的列或向量不需要线性间隔。由于这个原因，**mmtable** 比 **ilinear** 函数更普遍。在 MATLAB 版本 5 中，**interp1** 将用 **ilinear** 来实现线性插值。

11.5 小结

下面的表 11.1 总结了在 MATLAB 中所具有的曲线拟合和插值函数。

表 11.1

曲 线 拟 合 和 插 值 函 数	
polyfit(x, y, n)	对描述 n 阶多项式 y=f(x)的数据 进行最小二乘曲线拟合
interp1(x, y, xo)	1 维线性插值
interp1(x, y, xo, ' spline ')	1 维 3 次样条插值
interp1(x, y, xo, ' cubic ')	1 维 3 次插值
interp2(x, y, Z, xi, yi)	2 维线性插值
interp2(x, y, Z, xi, yi, ' cubic ')	2 维 3 次插值
interp2(x, y, Z, xi, yi, ' nearest ')	2 维最近邻插值

第 12 章 三次样条

众所周知，使用高阶多项式的插值常常产生病态的结果。目前，有多种消除病态的方法。在这些方法中，三次样条是最常用的一种。在 MATLAB 中，实现基本的三次样条插值的函数有 **spline**，**ppval**，**mkpp** 和 **unmkpp**。在这些函数中，仅 **spline** 在《MATLAB 参考指南》中有说明。下面几节，将展示在 M 文件函数中实现三次样条的基本特征。

12.1 基本特征

在三次样条中，要寻找三次多项式，以逼近每对数据点间的曲线。在样条术语中，这些数据点称之为**断点**。因为，两点只能决定一条直线，而在两点间的曲线可用无限多的三次多项式近似。因此，为使结果具有唯一性。在三次样条中，增加了三次多项式的约束条件。通过限定每个三次多项式的一阶和二阶导数，使其在断点处相等，就可以较好地确定所有内部三次多项式。此外，近似多项式通过这些断点的斜率和曲率是连续的。然而，第一个和最后一个三次多项式在第一个和最后一个断点以外，没有伴随多项式。因此必须通过其它方法确定其余的约束。最常用的方法，也是函数 **spline** 所采用的方法，就是采用非扭结**(not-a-knot)**条件。这个条件强迫第一个和第二个三次多项式的三阶导数相等。对最后一个和倒数第二个三次多项式也做同样地处理。

基于上述描述，人们可能猜想到，寻找三次样条多项式需要求解大量的线性方程。实际上，给定 N 个断点，就要寻找 N-1 个三次多项式，每个多项式有 4 个未知系数。这样，所求解的方程组包含有 4*(N-1)个未知数。把每个三次多项式列成特殊形式，并且运用各种约束，通过求解 N 个具有 N 个未知系数的方程组，就能确定三次多项式。这样，如果有 50 个断点，就有 50 个具有 50 个未知系数的方程组。幸好，用稀疏矩阵，这些方程式能够简明地列出并求解，这就是函数 **spline** 所使用的计算未知系数的方法。

12.2 分段多项式

在最简单的用法中，**spline** 获取数据 **x** 和 **y** 以及期望值 **xi**，寻找拟合 **x** 和 **y** 的三次样条内插多项式，然后，计算这些多项式，对每个 **xi** 的值，寻找相应的 **yi**。例如：

```
>>x=0 : 12;
>>y=tan(pi*x/25);
>>xi=linspace(0, 12);
>>yi=spline(x, y, xi)
>>plot(x, y, ‘ o ‘, xi, yi), title(‘ Spline fit ‘)
```

（见图 12.1 样条拟合）

这种方法适合于只需要一组内插值的情况。不过，如果需要从相同数据集里获取另一

组内插值，再次计算三次样条系数是没有意义的。在这种情况下，可以调用仅带前两个参量的 **spline**：

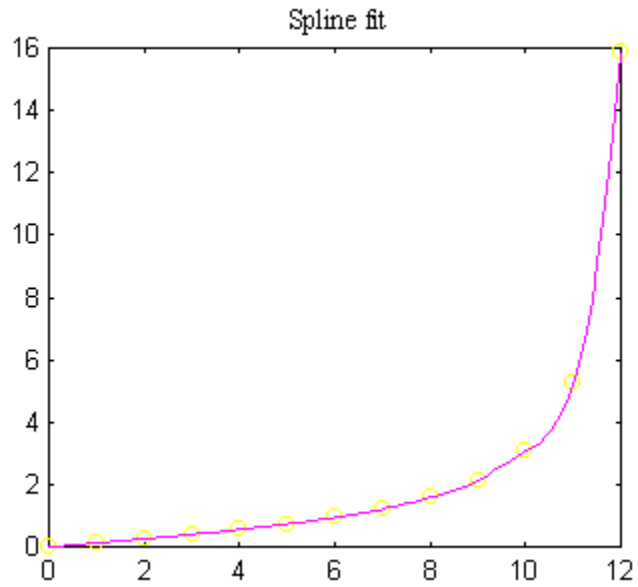


图 12.1 样条拟合

```
>>pp=spline(x, y)
pp =
    Columns 1 through 7
     10.0000    1.0000   12.0000         0    1.0000    2.0000    3.0000
    Columns 8 through 14
      4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000
    Columns 15 through 21
     11.0000   12.0000    4.0000    0.0007    0.0007    0.0010    0.0012
    Columns 22 through 28
      0.0024    0.0019    0.0116   -0.0083    0.1068   -0.1982    1.4948
    Columns 29 through 35
      1.4948   -0.0001    0.0020    0.0042    0.0072    0.0109    0.0181
    Columns 36 through 42
      0.0237    0.0586    0.0336    0.3542   -0.2406    4.2439    0.1257
    Columns 43 through 49
      0.1276    0.1339    0.1454    0.1635    0.1925    0.2344    0.3167
    Columns 50 through 56
      0.4089    0.7967    0.9102    4.9136         0    0.1263    0.2568
    Columns 57 through 63
      0.3959    0.5498    0.7265    0.9391    1.2088    1.5757    2.1251
    Columns 64 through 65
      3.0777    5.2422
```

当采用这种方式调用时，**spline** 返回一个称之为三次样条的 **pp 形式**或**分段多项式形式**

的数组。这个数组包含了对于任意一组所期望的内插值和计算三次样条所必须的全部信息。给定 pp 形式，函数 **ppval** 计算该三次样条。例如，

```
>>yi=ppval(pp, xi);
```

计算先前计算过的同样的 yi。
类似地，

```
>>xi2=linspace(10, 12);
>>yi2=ppval(pp, xi2);
```

运用 pp 形式，在限定的更细区间[10，12]内，再次计算该三次样条。

```
>>xi3=10 : 15
>>yi3=ppval(pp, xi3)
yi3 =
    3.0777    5.2422   15.8945   44.0038   98.5389  188.4689
```

它表明，可在计算三次多项式所覆盖的区间外，计算三次样条。当数据出现在最后一个断点之后或第一个断点之前时，则分别运用最后一个或第一个三次多项式来寻找内插值。

上述给定的三次样条 pp 形式，存储了断点和多项式系数，以及关于三次样条表示的其它信息。因为，所有信息都被存储在单个向量里，所以这种形式在 MATLAB 中是一种方便的数据结构。当要计算三次样条表示时，必须把 pp 形式分解成它的各个表示段。在 MATLAB 中，通过函数 **unmkpp** 完成这一过程。运用上述 pp 形式，该函数给出如下结果：

```
>>[break, coefs, npolys, ncoefs]=unmkpp(pp)
breaks =
  Columns 1 through 12
     0     1     2     3     4     5     6     7     8     9    10    11
  Column 13
    12
coefs =
    0.0007   -0.0001    0.1257         0
    0.0007    0.0020    0.1276    0.1263
    0.0010    0.0042    0.1339    0.2568
    0.0012    0.0072    0.1454    0.3959
    0.0024    0.0109    0.1635    0.5498
    0.0019    0.0181    0.1925    0.7265
    0.0116    0.0237    0.2344    0.9391
   -0.0083    0.0586    0.3167    1.2088
    0.1068    0.0336    0.4089    1.5757
   -0.1982    0.3542    0.7967    2.1251
    1.4948   -0.2406    0.9102    3.0777
    1.4948    4.2439    4.9136    5.2422
```

```
npolys =
     12
ncoefs =
      4
```

这里 **break** 是断点，**coefs** 是矩阵，它的第 i 行是第 i 个三次多项式，**npolys** 是多项式的数目，**ncoefs** 是每个多项式系数的数目。注意，这种形式非常一般，样条多项式不必是三次。这对于样条的积分和微分是很有益的。

给定上述分散形式，函数 **mkpp** 恢复了 pp 形式。

```
>>pp=mkpp(break, coefs)
pp =
  Columns 1 through 7
   10.0000    1.0000   12.0000         0    1.0000    2.0000    3.0000
  Columns 8 through 14
    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000
  Columns 15 through 21
   11.0000   12.0000    4.0000    0.0007    0.0007    0.0010    0.0012
  Columns 22 through 28
    0.0024    0.0019    0.0116   -0.0083    0.1068   -0.1982    1.4948
  Columns 29 through 35
    1.4948   -0.0001    0.0020    0.0042    0.0072    0.0109    0.0181
  Columns 36 through 42
    0.0237    0.0586    0.0336    0.3542   -0.2406    4.2439    0.1257
  Columns 43 through 49
    0.1276    0.1339    0.1454    0.1635    0.1925    0.2344    0.3167
  Columns 50 through 56
    0.4089    0.7967    0.9102    4.9136         0    0.1263    0.2568
  Columns 57 through 63
    0.3959    0.5498    0.7265    0.9391    1.2088    1.5757    2.1251
  Columns 64 through 65
    3.0777    5.2422
```

因为矩阵 **coefs** 的大小确定了 **npolys** 和 **neofs**，所以 **mkpp** 不需要 **npolys** 和 **ncoefs** 去重构 pp 形式。pp 形式的数据结构仅在 **mkpp** 中给定为 **pp=[10 1 npolys break(:)' ncoefs coefs(:)']**。前两个元素出现在所有的 pp 形式中，它们作为确认 pp 形式向量的一种方法。

12.3 积分

在大多数情况下，需要知道由三次样条所描述的、自变量为 x 的函数所包含的面积。也就是，如果这个函数记为 y=f(x)，我们感兴趣的是计算：

$S(x)=\int_{x1}^{x} s(x)dx$　其中，是 s(x1)=0

式中的 x1 是第一个样条的断点。因为 s(x)由被连接的三次多项式组成，其中第 k 个三次多项式为：

$s_k(x)=a_k(x-x_k)^3+ b_k(x-x_k)^2+ c_k(x-x_k)+d_k$,　$x_k<=x<= x_{k+1}$

并且该函数在区间 $x_k<=x<= x_{k+1}$ 所含的面积为：

$$S_k(x)=\int_{x1}^{x} s_k(x)dx = a_k/4(x-x_k)^4+b_k/3(x-x_k)^3+c_k/2(x-x_k)^2+d_k(x-x_k)$$

三次样条下的面积容易求得为：

$$S(x)=\sum_{i=1}^{k-1}\int_{x_i}^{x_{i=1}} s_i(x)dx+\int_{x_k}^{x} s_k(x)dx$$　式中 $x_k<=x<= x_{k+1}$

或者

$$S(x)=\sum_{i=1}^{k-1} S_i(x_i+1)+S_k(x)$$　式中 $x_k<=x<= x_{k+1}$

上式相加项是所有处理的三次多项式面积的累加和。照此，因为 $S_k(x)$是一个多项式，所以该面积很容易计算，且在描述 S(x)的多项式中可形成常数项。有了上述理解，积分本身可写成样条形式。在这种情况下，因为单个多项式具有 4 阶，所以积分是四次样条。

MATLAB 中使用的 pp 形式支持任意阶的样条，所以在《精通 MATLAB 工具箱》中的函数 **spintgrl** 体现了上述样条积分。该函数的主体如下：

```
function z=spintgrl(x, y, xi)
%   SPINTGRL Cubic Spline Integral Interpolation
%   YI=SPINTRGL(X, Y, XI) uses cubic spline interpolation to fit the date in X and Y, integrates
%   the spline and returns values of the integral evaluated at the points in XI.
%
%   PPI=SPINTGRL(PP) returns the piecewise polynomial vector PPI
%   describing the integral of the cubic spline described by
%   the piecewise polynomial in PP. PP is returned by the function
%   SPLINE and is a data vector containing all information to
%   evaluate and manipulate a spline.
```

```
%
%    YI=SPINTGRL(PP, XI) integrates the cubic splines given by
%    the piecewise polynomial PP, and returns the values of the
%    integral evaluated at the points in XI.
%
%    See also SPLINE, PPVAL, MKPP, UNMKPP, SPDERIV

%    Copyright    (c) 1996 by Prentice-Hall, Inc.
if nargin==3
pp=spline(x, y)
else
     pp=x;
end
[br, co, npy, nco]=unmkpp(pp);    %    take apart pp
if pp(1)==10
     error(' Spline data does not have the correct form. ')
end
sf=nco : -1 : 1;    %    scale factors for integration
ico=[co ./ sf(ones(npy, 1), : ) zeros(npy, 1)];    %    integral coefficients
nco=nco+1;    %    spline order increases
for k=2 : npy    %    find constant terms
     ico(k, nco)=polyval(ico(k-1, : ),br(k)-br(k-1));
end
ppi=mkpp(br, ico);    %    build pp form for integral
if nargin==1
     z=ppi;
elseif nargin==2
     z=ppval(ppi, y);
else
     z=ppval(ppi, xi);
```

```
end
```

考虑如下运用 spintgrl 的例子：

```
>>x=(0:  .1:  1)*2*pi;
>>y=sin(x);  %  create rough data
>>pp=spline(x, y);  %  pp-form fitting rough data
>>ppi=spintgrl(pp);  %  pp-form of integral
>>xi=linspace(0, 2*pi);  %  finer points for interpolation
>>yi=ppval(pp, xi);  %  evaluate curve
>>yyi=ppval(ppi, xi);  %  evaluate integral
>>plot(x, y, ' o ', xi, yi, xi, yyi, ' - ')  %  plot results
```

注意，如图 12.2 所示，它定性地证明了恒等式：

$$\int_{x_1}^{x} \sin(x)dx = 1 - \cos(x)$$

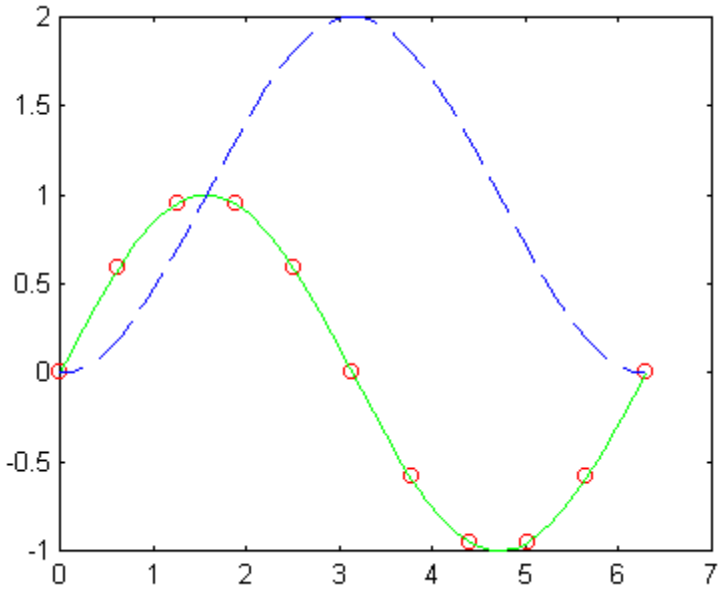


图 12.2　函数的积分图形

12.4　微分

如同人们对样条积分感兴趣一样，一个由样条所描述的函数的微分或斜率也是很有用的。给定第 k 个三次多项式为：

$s_k(x)=a_k(x-x_k)^3+ b_k(x-x_k)^2+ c_k(x-x_k)+d_k, \quad x_k<=x<= x_{k+1}$

容易写出其导数为：

$$\frac{ds_k(x)}{dx} = 3a_k(x-x_k)^2 + 2b_k(x-x_k) + c_k, x_k \le x \le x_{k+1}$$

如同样条积分，样条微分也是一种样条。不过，在这种情况下，这个多项式为二阶，所以它是二次样条。

基于上述描述，《精通 MATLAB 工具箱》中的函数 **spderiv** 实施样条微分。**spderiv** 的主体为：

```
function z=spderiv(x, y, xi)
%   SPDERIV Cubic Spline Derivative Interpolation
%   YI=SPDERIV(X, Y, XI) uses cubic spline interpolation to fit the
%   data in X and Y, differentiates the spline and returns values
%   of the spline derivatives evaluated at the points in XI.
%
%   PPD=SPDERIV(PP) returns the piecewise polynomial vector PPD
%   describing the cubic spline derivative of the curve described by
%   the piecewise polynomial in PP. PP is returned by the function
%   SPLINE and is a data vector containing all information to
%   evaluate and manipulate a spline.
%
%   YI=SPDERIV(PP, XI) differentiates the cubic spline given by
%   the piecewise polynomial PP, and returns the value of the
%   spline derivatives evaluated at the points in XI.
%
%   See also SPLINE, PPVAL, MKPP, UNMKPP, SPINTGRL

%   Copyright (c) 1996 by Prentice-Hall, Inc.

if nargin==3
    pp=spline(x, y);
else
    pp=x;
end
[br, co, npy, nco]=unmkpp(pp);   %   take apart pp
if nco==1 | pp(1)~=10
    error(' Spline data does not have the correct PP form. ')
end
sf=nco-1 : -1 : 1;   %   scale factors for differentiation
dco=sf(ones(npy, 1),:).*co( : , 1 : nco-1);   %   derivative coefficients
ppd=mkpp(br, dco);   %   build pp form for derivative
```

```
if nargin==1
    z=ppd;
elseif nargin==2
    z=ppval(ppd, y);
else
    z=ppval(ppd, xi);
end
```

为演示 **spderiv** 的使用，考虑如下例子：

```
>>x=(0 :  .1 :  1)*2*pi;  %  same data as earlier
>>y=sin(x);
>>pp=spline(x, y);  %  pp-form fitting rough data
>>ppd=spderiv(pp);  %  pp-form of derivative
>>xi=linspace(0, 2*pi);  %  finer points for interpolation
>>yi=ppval(pp, xi);  %  evaluate curve
>>yyd=ppval(ppd, xi);  %  evaluate derivative
>>plot(x, y, ' o ', xi, yi, xi, yyd, ' - ')  %  plot results
```

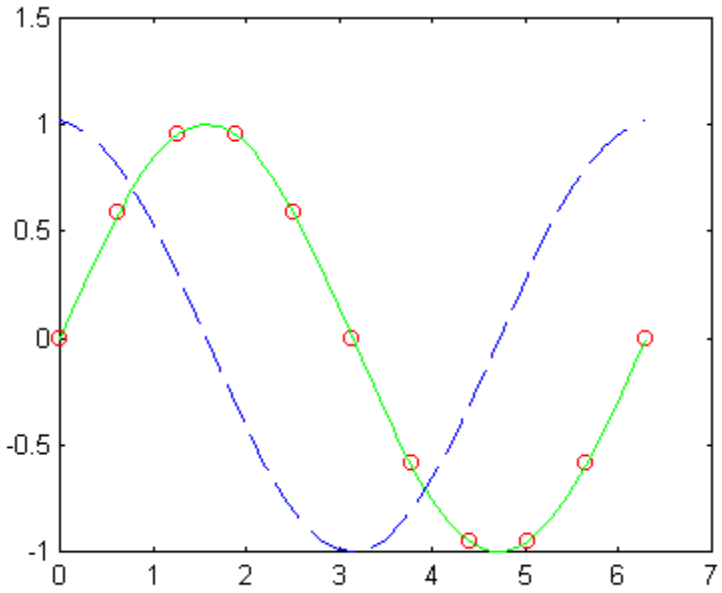


图 12.3　函数的微分图形

注意，图 12.3 定性地证明了恒等式：

$$\frac{d}{dx}\sin(x) = \cos(x)$$

12.5　小结

下面的两个表总结了本章所讨论的样条函数。

表 12.1

三次样条函数	
yi=spline(x,y,xi)	y=f(x)在 **xi** 中各点的三次样条插值
pp=spline(x,y)	返回 y=f(x)的分段多项式的表示
yi=ppval(pp,xi)	计算 **xi** 中各点的分段多项式
[break,coefs,npolys,ncoefs]=unmkpp(pp)	分解分段多项式的表示
pp=mkpp(break,coefs)	形成分段多项式的表示

表 12.2

精通 **MATLAB** 的样条函数	
yi=spintgrl(x,y,xi)	在 xi 各点对 y=f(x)积分的三次样条的插值
ppi=spintgrl(pp)	给定 y=f(x)的分段多项式的表示，返回 y=f(x)积分的分段多项式的表示。
yi=spintgrl(pp,xi)	给定 y=f(x)的分段多项式的表示，计算点 **xi** 的值，寻找 y=f(x)积分的分段多项式的描述。
yi=spderiv(x,y,xi)	在 **xi** 各点对 y=f(x)微分的三次样条的插值
ppi=spderiv(pp)	给定 y=f(x)的分段多项式的表示，返回 y=f(x)微分的分段多项式的表示。
yi=spderiv(pp,xi)	给定 y=f(x)的分段多项式的表示，计算点 **xi** 的值，寻找 y=f(x)微分的分段多项式的表示。

第 13 章　数值分析

每当难以对一个函数进行积分、微分或者解析上确定一些特殊的值时，就可以借助计算机在数值上近似所需的结果。这在计算机科学和数学领域，称之为数值分析。至此，可以猜到，MATLAB 提供了解决这些问题的工具。本章将介绍这些工具的使用。

13.1　绘图

说到绘图，只要计算函数在某一区间的值，并且画出结果向量，这样就得到了函数的图形。在大多数情况下，这就足够了。然而，有时一个函数在某一区间是平坦的并且无激励，而在其它区间却失控。在这种情况下，运用传统的绘图方法会导致图形与函数真正的特性相去甚远。MATLAB 提供了一个称为 **fplot** 的巧妙的绘图函数。该函数细致地计算要绘图的函

数，并且确保在输出的图形中表示出所有的奇异点。该函数的输入需要知道以字符串表示的被画函数的名称以及 2 元素数组表示的绘图区间。例如：

```
>>fplot(‘ humps ‘ , [0    2])
>>title(‘ FPLOT OF HUMPS ‘)
```

在 0 和 2 之间计算函数 **humps**，并显示该函数的图形。（见图 13.1)。

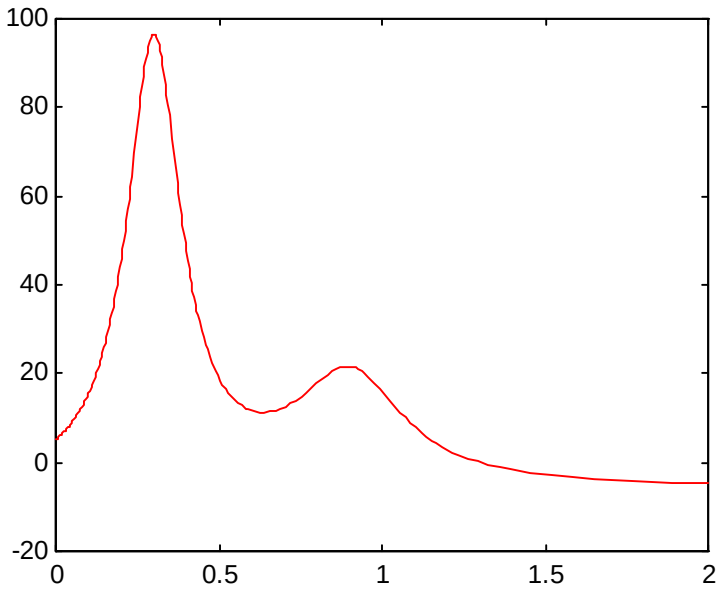


图 13.1　函数 humps 的图形

在这个例子中，‘ **humps** ‘是 MATLAB 的 M 文件函数。

```
function y=humps(x)
%    HUMPS A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
%    HUMPS(X) is a function with strong maxima near x= .3 and x= .9.
%    See QUADDEMO,    ZERODEMO and FPLOTDEMO.
%    Copyright (c) 1984-93 by The MathWorks, Inc.
y=1 ./ ((x - .3) .^ 2+ .01)+1 ./ ((x - .9) .^ 2+ .04) - 6;
```

fplot 适用于任何具有单输入和单输出向量的函数 M 文件。即如同 **humps**，输出变量 y 返回一个与输入 **x** 同样大小的数组，在数组到数组意义上 **y** 和 **x** 有联系。在使用 **fplot**（以及其它数值分析函数）的过程中，最普遍犯的错误是忘记把函数名加上引号。即 **fplot** 需要知道字符串形式的函数名。如果输入 **fplot(humps , [0 , 2])**，MATLAB 认为 **humps** 是工作空间中的一个变量，而不是函数的名称。注意把变量 **humps** 定义为所需要的字符串，就可避免这个问题。

```
>>humps=‘ humps ‘;
>>fplot(hump , [0    2])
```

这时，MATLAB 从变量 **humps** 中获得字符串‘ **humps**‘。

对于可表示成一个字符串的简单的函数，如 $y = 2e^{-x}\sin(x)$，**fplot** 绘制这类函数的曲线时，不用建立 M 文件，只需把 x 当作自变量，把被绘图的函数写成一个完整的字符串。

```
>>f=‘ 2*exp(-x) .* sin(x) ‘;
```

式中，运用数组乘法定义了函数 $f(x) = 2e^{-x}\sin(x)$

```
>>fplot(f , [0    8]);
>>title(f) , xlabel(‘x‘)
```

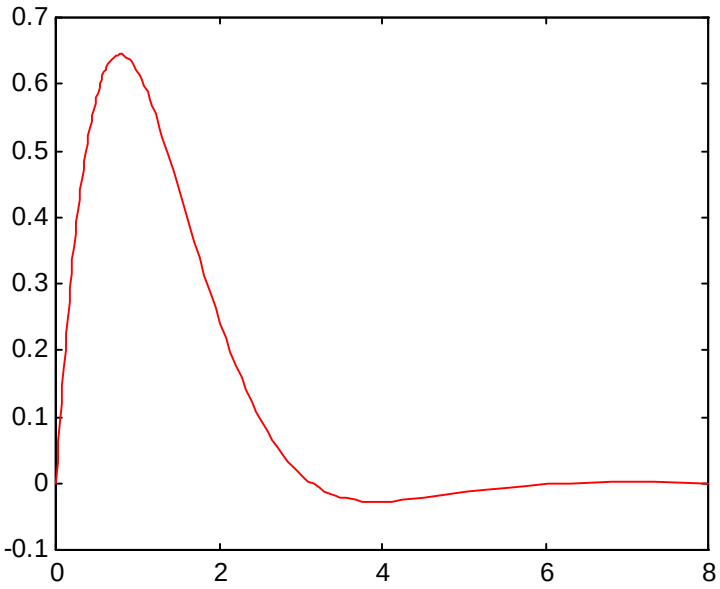


图 13.2　$f(x) = 2e^{-x}\sin(x)$ 的曲线

在区间 $0 \le x \le 8$ 绘出上述函数，产生如图 13.2 所示的图形。

除了这些基本特性，函数 **fplot** 还有很多强大的功能，有关详细的信息，参阅《MATLAB 参考指南》或在线帮助。

13.2　极小化

作图除了提供视觉信息外，还常常需要确定一个函数的其它更多的特殊属性。在许多应用中，特别感兴趣的是确定函数的极值，即**最大值**（峰值）和**最小值**（谷值）。数学上，可通过确定函数导数（斜率）为零的点，解析上求出这些极值点。检验 **humps** 的图形在峰值和谷值点上的斜率就很容易理解这个事实。显然，如果定义的函数简单，则这种方法常常

奏效。然而，即使很多容易求导的函数，也常常很难找到导数为零的点。在这种情况下，以及很难或不可能解析上求得导数的情况下，必须数值上寻找函数的极值点。MATLAB 提供了两个完成此功能的函数 **fmin** 和 **fmins**。这两个函数分别寻找一维或 n 维函数的最小值。这里仅讨论 **fmin**。有关 **fmins** 的详细信息，参阅《MATLAB 参考指南》。因为 f(x)的最大值等于-f(x)的最小值，所以，上述 **fmin** 和 **fmins** 可用来求最大值和最小值。如果还不清楚，把上述图形倒过来看，在这个状态下，峰值变成了谷值，而谷值则变成了峰值。

为了解释求解一维函数的最小值和最大值，再考虑上述例子。从图 13.2 可知，在 x_{max}=0.7 附近有一个最大值，并且在 x_{min}=4 附近有一个最小值。而这些点的解析值为：$x_{\max} = \pi / 4 \approx 0.785$ 和 $x_{\min} = 5\pi / 4 \approx 3.93$。为了方便，用文本编辑器编写一个脚本 M 文件，并用 fmin 寻出数值上极值点，给出函数主体如下：

```
%    ex_fmin.m
fn=‘ 2*exp(-x)*sin(x) ‘;          %    define function for min
xmin=fmin(fn , 2 , 5)             %    search over range 2<x<5
emin=5*pi / 4-xmin                %    find error
x=xmin;                           %    need x since fn has x as its variable
ymin=eval(fn)                     %    evaluate at xmin
fx=‘ -2*exp(-x)*sin(x) ‘;         %    define for max:note minus sign
xmax=fmin(fx , 0 , 3)             %    search over range 0<x<3
emax=pi / 4-xmax                  %    find error
x=xmax;                           %    need x since fn has x as its variable
ymax=eval(fn)                     %    evaluate at xmax
```

下面是 M 文件的运行结果：

```
>>ex-fmin
xmin =
      3.9270
emin =
   1.4523e-006
ymin =
    -0.0279
xmax =
      0.7854
emax =
 -1.3781e-005
ymax =
      0.6448
```

这些结果与上述图形非常吻合。注意，**fmin** 的工作方式很像 **fplot**。要计算的函数可用一个函数 M 文件表达，或者只给出一个 x 为自变量的字符串。上述例子就是使用后一种方法。这个例子也使用了函数 **eval**，它获取一个字符串，并解释它，如同在 MATLAB 提示符下输入该字符串。由于要计算的函数以 **x** 为自变量的字符串形式给出，那么设置 **x** 等于 **xmin**

和 **xmax**，允许 **eval** 计算该函数，找到 **ymin** 和 **ymax**。

最后，特别注意，求数值上的最小值包含一个搜索过程，**fmin** 不断计算函数值，寻求其最小值。如果计算的函数需要很大的计算量，或者该函数在搜索区间不止一个最小值，则该搜索过程所花的时间比较长。在有些情况下，搜索过程根本找不到结果。当 **fmin** 找不到最小值时，它会停止运行并提供解释。

与函数 **fmin** 一样，函数 **fmins** 搜索最小值。不过，**fmins** 搜索向量的标量函数的最小值。即 **fmins** 寻找

$$\min_{x} f(x)$$

这里 **x** 是函数 f(.)的向量参数，函数 f(.)返回标量值。函数 fmins 利用单纯形法求最小值，它不需要精确的梯度计算。任何一种优化工具箱中具有更多扩展的优化算法

13.3 求零点

正如人们对寻找函数的极点感兴趣一样，有时寻找函数过零或等于其它常数的点也非常重要。一般试图用解析的方法寻找这类点非常困难，而且很多时候是不可能的。在上述函数 **humps** 的图中(如图 13.3 所示)，该函数在 x=1.2 附近过零。

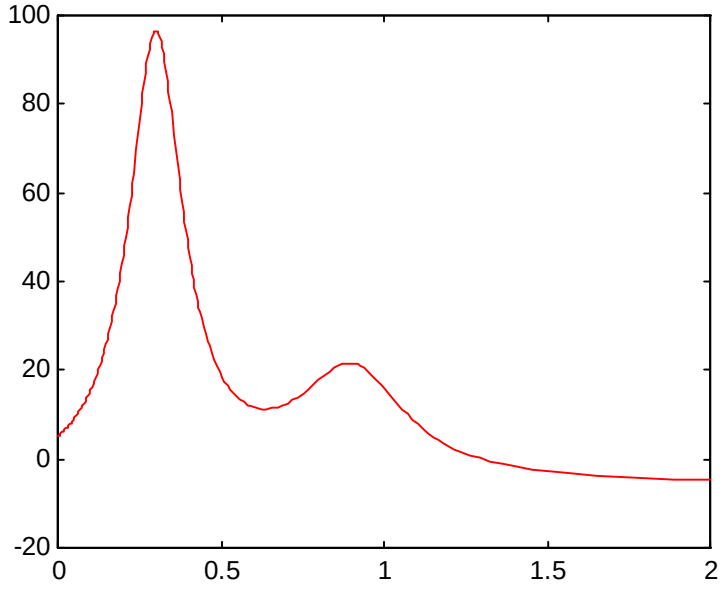


图 13.3 humps 函数的图形

MATLAB 再一次提供了该问题的数值解法。函数 **fzero** 寻找一维函数的零点。为了说明该函数的使用，让我们再运用 **humps** 例子。

```
>>xzero=fzero(‘ humps ‘ , 1.2)        %    look for a zero near 1.2
```

```
xzero=
      1.2995
>>yzero=humps(xzero , 1.2)          %    evaluate at xzero
yzero=
    3.5527e-15
```

所以，humps 的零点接近于 1.3。如前所述，寻找零点的过程可能失败。如果 **fzero** 没有找到零点，它将停止运行并提供解释。

当调用函数 **fzero** 时，必须给出该函数的名称。但由于某种原因，它不能接受以 x 为自变量的字符串来描述的函数。这样，即使在 **fplot** 和 **fmin** 中都具有的这个特性，**fzero** 将不工作。

fzero 不仅能寻找零点，它还可以寻找函数等于任何常数值的点。仅仅要求一个简单的再定义。例如，为了寻找 f(x)=c 的点，定义函数 g(x)=f(x)-c，然后，在 **fzero** 中使用 g(x)，就会找出 g(x)为零的 x 值，它发生在 f(x)=c 时。

13.4 积分

一个函数的积分或面积也是它的另一个有用的属性。MATLAT 提供了在有限区间内，数值计算某函数下的面积的三种函数：**trap2 , quad** 和 **quad8**。函数 **trapz** 通过计算若干梯形面积的和来近似某函数的积分，这些梯形如图 13.4 所示，是通过使用函数 **humps** 的数据点形成。

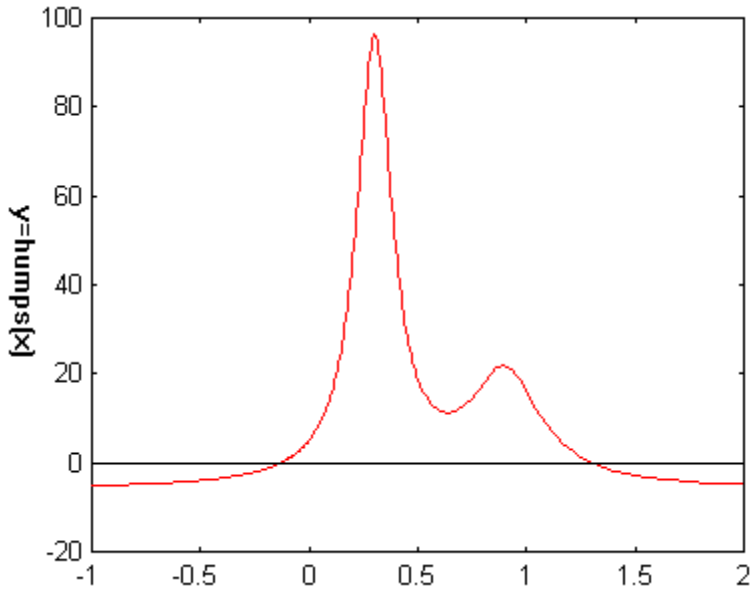


图 13.4 粗略的梯形逼近曲线下的面积示意图

从图中可明显地看出，单个梯形的面积在某一段欠估计了函数真正的面积，而在其它段又过估计了函数的真正面积。如同线性插值，当梯形数目越多时，函数的近似面积越准确。例如，在图 13.4 中，如果我们大致增加一倍数目的梯形，我们得到如下页（如图 13.5）所

示的更好的近似结果。

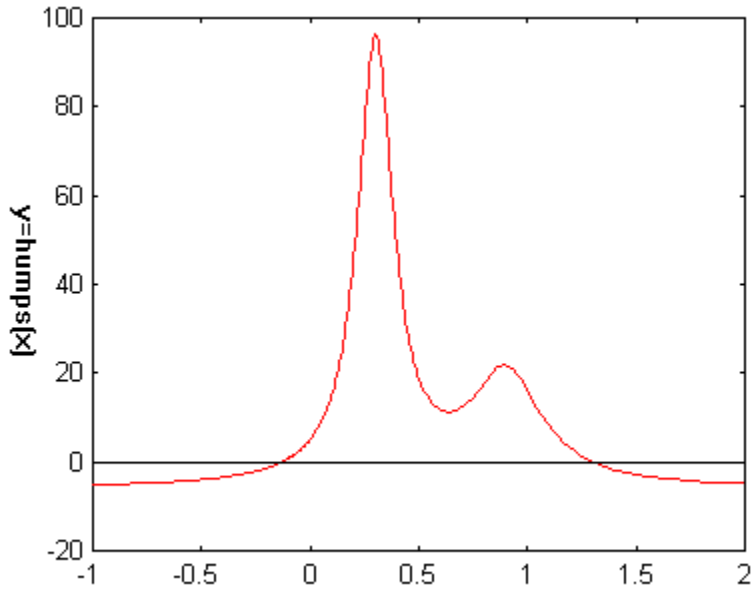


图 13.5 较好的梯形逼近曲线下的面积示意图

对如上所示的两个曲线，用 **trapz** 在区间-1<x<2 上计算 y=humps(x)下面的面积：

```
>>x=-1 : 0.17 : 2;                    %   rough approximation
>>y=humps(x);
>>area=trapz(x , y)                   %   call trapz just like the plot command
area =
    25.9174
>>x=-1 : 0.07 : 2;                    %   better approximation
>>y=humps(x);
>>area=trapz(x , y)
area =
    26.6243
```

自然地，上述两个结果不同。基于对图形的观察，粗略近似可能低估了实际面积。除非特别精确，没有准则说明哪种近似效果更好。很明显，如果人们能够以某种方式改变单个梯形的宽度，以适应函数的特性，即当函数变化快时，使得梯形的宽度变窄，这样就能够得到更精确的结果。

MATLAB 的函数 **quad** 和 **quad8** 是基于数学上的正方形概念来计算函数的面积，这些积分函数的操作方式一样。为获得更准确的结果，两个函数在所需的区间都要计算被积函数。此外，与简单的梯形比较，这两个函数进行更高阶的近似，而且 **quad8** 比 **quad** 更精确。这两个函数的调用方法与 **fzero** 相同，即

```
>>area=quad(‘ humps ‘ , -1 , 2)    %    find area between -1 and 2
area =
```

```
    26.3450
>>area=quad8(‘ humps ‘ , -1 , 2)
area =
    26.3450
```

注意，这两个函数返回完全相同的估计面积，而且这个估计值在两个 trapz 面积的估计值之间。有关 MATLAB 的积分函数的其它信息，参阅《MATLAB 参考指南》或在线帮助。

13.5 微分

与积分相反，数值微分非常困难。积分描述了一个函数的整体或宏观性质，而微分则描述一个函数在一点处的斜率，这是函数的微观性质。因此积分对函数的形状在小范围内的改变不敏感。而微分却很敏感。一个函数小的变化，容易产生相邻点的斜率的大的改变。

由于微分这个固有的困难，所以尽可能避免数值微分，特别是对实验获得的数据进行微分。在这种情况下，最好用最小二乘曲线拟合这种数据，然后对所得到的多项式进行微分。或用另一种方法，对该数据进行三次样条拟合，然后寻找如第 11 章所讨论的样条微分。例如，再次考虑第 11 章曲线拟合的例子。

```
>>x=[0   .1   .2   .3   .4   .5   .6   .7   .8   .9   1]
>>y=[-.447   1.978   3.28   6.16   7.08   7.34   7.66   9.56   9.48   9.30   11.2];   % data
>>n=2;   %   order of fit
>>p=polyfit(x , y , n)     %   find polynomial coefficients
p =
    -9.8108     20.1293     -0.0317
>>xi=linspace(0 , 1 , 100);
>>z=polyval(p , xi);       %   evaluate polynomial
>>plot(x , y , ‘ o ' , x , y , xi , z , ' : ')
>>xlabel(‘ x ‘) , ylabel(‘ y=f(x) ‘) , title(‘ Second Order Curve Fitting ‘)
```

在这种情况下，运用多项式微分函数 **polyder** 求得微分。

```
>>pd=polyder(p)
pd =
   -19.6217     20.1293
```

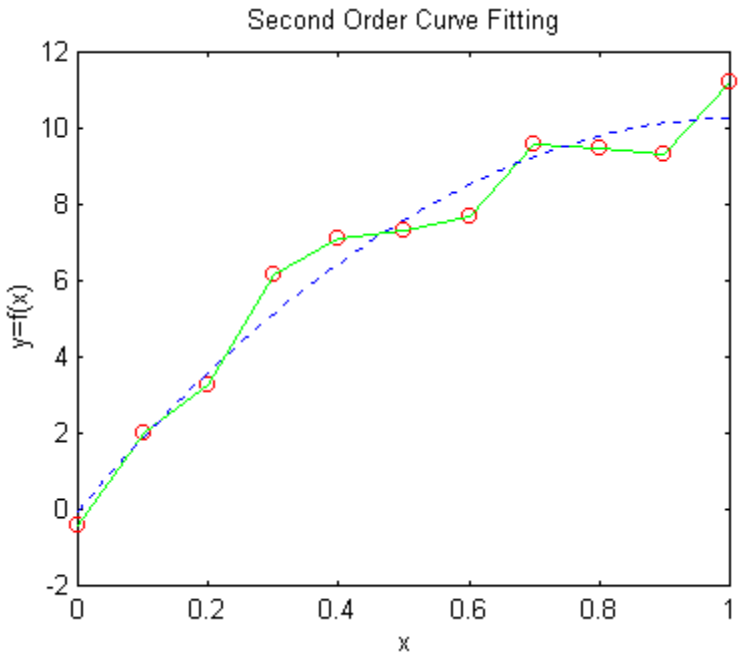


图 13.6 二次曲线拟合

$y = -9.8108x^2 + 20.1293x - 0.0317$ 的微分是 dy/dx=-19.6217x+20.1293。由于一个多项式的微分是另一个低一阶的多项式，所以还可以计算并画出该函数的微分。

```
>>z=polyval(pd , xi);    %    evaluate derivative
>>plot(xi , z)
>>xlabel(‘ x ‘) , ylabel(‘ dy/dx ‘) , title(‘ Derivative of a curve Fit Polynimial ‘)
```

(微分曲线如图 13.7 所示）

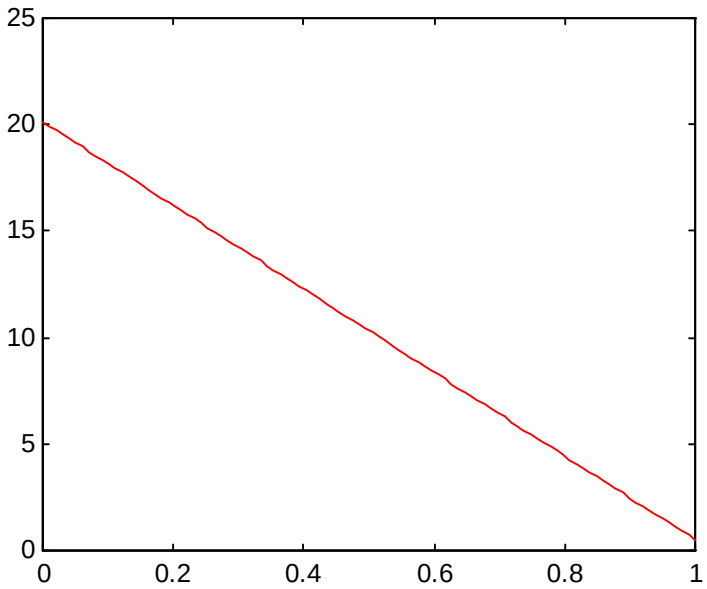


图 13.7 曲线拟合多项式微分

在这种情况下，拟合的多项式为二阶，使其微分为一阶多项式。这样，微分为一条直线，它意味该微分与 x 成线性变化。

给定一些描述某函数的数据，MATLAB 提供了一个计算其非常粗略的微分的函数。这个函数命名为 **diff**，它计算数组中元素间的差分。因为微分定义为：

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{(x+h) - (x)}$$

则 y=f(x)的微分可近似为：

$$\frac{dy}{dx} \approx \frac{f(x+h) - f(x)}{(x+h) - (x)}$$ 这里 h>0

它是 y 的有限差分除以 x 的有限差分。因为 **diff** 计算数组元素间的差分，所以在 MATLAB 中，可近似求得函数的微分。继续前一个例子：

```
>>dy=diff(y) ./ diff(x);          %   compute differences and use array division
>>xd=x(1 : length(x)-1);          %   create new x axis since dy is shorter than y
>>plot(xd , dy);
>>title(‘ Approximate Derivative Using DIFF ‘)
>>ylabel(‘ dy/dx ‘) , xlabel(‘ x ‘)
```

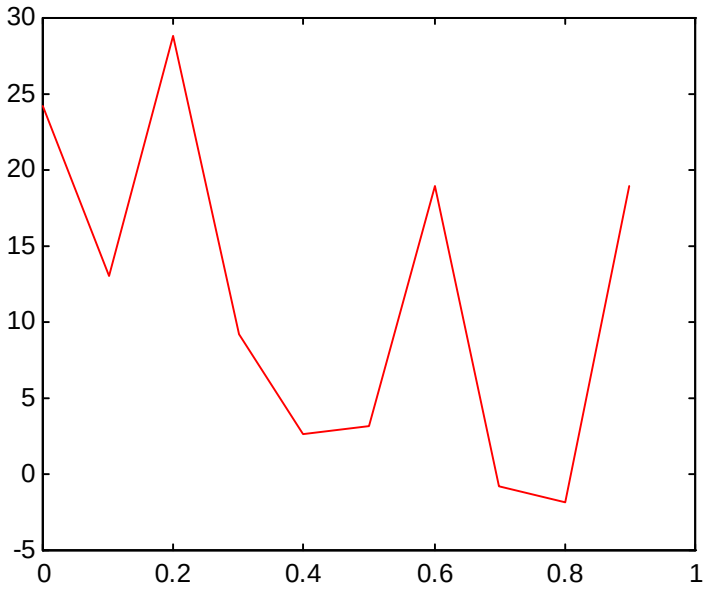


图 13.8 用 diff 得到的近似微分

由于 **diff** 计算数组元素间的差分，所以，其所得输出比原数组少了一个元素。这样，画微分曲线时，必须舍弃 x 数组中的一个元素。当舍弃 x 的第一个元素时，上述过程给出向后差分近似，而舍弃 x 的最后一个元素，则给出向前差分近似。比较上述两条曲线，显而易见，用有限差分近似微分会导致很差的结果，特别是被噪声污染了的数据。

13.6 微分方程

一般微分方程式描述系统内部变量的变化率如何受系统内部变量和外部激励，如输入，的影响。当常微分方程式能够解析求解时，可用 MATLAB 的符号工具箱中的功能找到精确解。在本书的后面将介绍该工具箱的一些特点。

在微分方程难以获得解析解的情况下，可以方便地在数值上求解。为了说明起见，考虑描述振荡器的经典的范得波（Var der Pol）微分方程。

$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$$

与所有的数值求解微分方程组的方法一样，高阶微分方程式必须等价地变换成一阶微分方程组。对于上述微分方程，通过重新定义两个新的变量，来实现这种变换。

令 $y_1=x$ 且 $y_2=dy/dx$
则 $dy1/dt=y_2$

$$dy_2 / dt = \mu(1-y_2) - y_1$$

根据这个微分方程组，可用 MATLAB 的函数 **ode23** 和 **ode45** 求出系统随时间变化的运动情况。调用这些函数时，需要编写一个函数 M 文件，给定当前时间及 y1 和 y2 的当前值，该函数返回上述导数值。MATLAB 中，这些导数由一个列向量给出。在本例中，这个列向量为 yprime。同样，y1 和 y2 合并写成列向量 y。所得函数 M 文件是：

```
function yprime=vdpol(t , y);
%   VDPOL(t , y) returns derivatives of the Van der Pol equation:
%
%   x ‘‘-mu *(1-x ^2)*x ‘+x=0 (‘ = d/dx , ‘‘ = d^2/dx^2)
%
%   let y(1)=x and y(2)=x‘
%
%   then y(1) ‘ = y(2)
%        y(2) ‘ = MU*(1-y(1)^2)*y(2)-y(1)

global MU %     choose 0<MU<10 in Command workspace
```

```
yprime=[y(2)    MU*(1-y(1)^2)*y(2)-y(1)];    %    output must be a column
```

给定这个完整地描述微分方程的函数，计算结果如下：

```
>>global MU %    define MU as a global variable in the Command Workspace
>>MU=2;    %    set global parameter to desired value
>>[t , y]=ode23(‘ vdpol ‘ , 0 , 30 , [1 ; 0]);    %    to=0 , tf=30 , yo=[1 ; 0]
>>y1=y( : , 1);    %    first column is y(1) versus time points in t
>>y2=y( : , 2);    %    second column is y(2)
>>plot(t , y1 , t , y2 , ‘ -- ‘)
>>xlabel(‘ Time , Second ‘) , ylabel(‘ Y(1) and Y(2) ‘)
>>title(‘ Van der Pol Solution for mu=2 ‘)
```

所得的图见图 13.9。

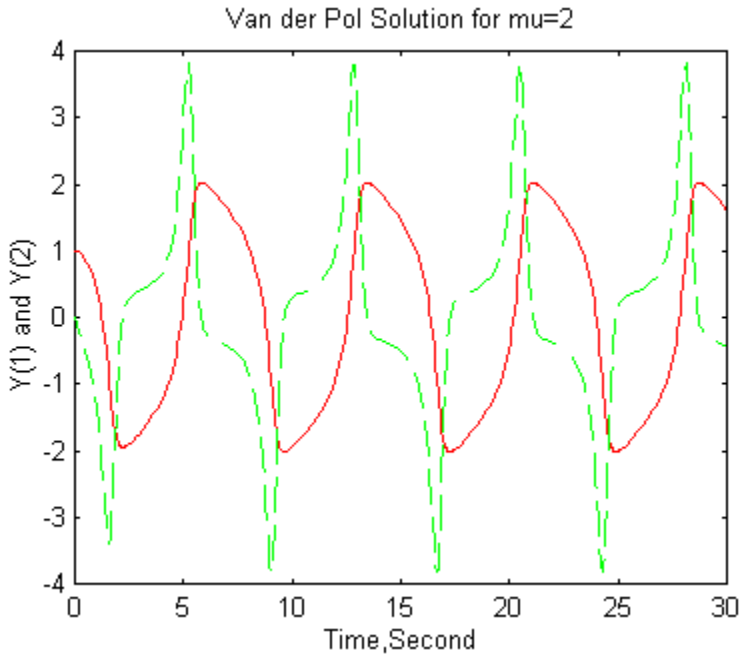


图 13.9 当 mu=2 时的范得波方程的运动曲线

在图 13.9 中，y2（虚线）是 y1（实线）的导数。传递给 **ode23** 的参数由 **ode23(f_name , to , tf , yo , to1)**描述。这里 **f_name** 是计算导数的 M 文件函数的字符串名，**to** 是初始时间，**tf** 是终止时间，**yo** 是初始条件向量。可选择的参数 **to1**（缺省值 **to1=1e-3**）是所需的相对精度。在上例中，起始时间是第 0 秒，终止时间是第 30 秒，初始条件为 **y=[1;0]**。两个输出参数是列向量 **t** 和矩阵 y，其中向量 t 包含了估计响应的时间点，而矩阵 y 的列数等于微分方程组的个数（本例为 2），且其行数与 t 相同。t 中的时间点不是等间隔的，因为为了保持所需的相对精度，积分算法改变了步长。

函数 **ode45** 的使用与 **ode23** 完全一样。两个函数的差别在于必须与所用的内部算法相关。两个函数都运用了基本的龙格-库塔(Runge-Kutta)数值积分法的变形。**ode23** 运用一个组合得 2/3 阶龙格-库塔-芬尔格(Runge-Kutta-Fehlerg)算法，而 **ode45** 运用组合的 4/5 阶龙格-库塔-芬尔格算法。一般地，**ode45** 可取较多的时间步。所以，要保持与 **ode23** 相同误差时，

在 to 和 tf 之间可取较少的时间步。然而，在同一时间，**ode23** 每时间步至少调用 **f_name 3** 次，而 **ode45** 每时间步至少调用 **f_name 6** 次。

正如使用高阶多项式内插常常得不到最好的结果一样，**ode45** 也不总是比 **ode23** 好。如果 **ode45** 产生的结果，对作图间隔太大，则必须在更细的时间区间，对数据进行内插，比如用函数 **interp1**。这个附加时间点会使 **ode23** 更有效。作为一条普遍规则，在所计算的导数中，如有重复的不连续点，为保持精度致使高阶算法减少时间步长，这时低阶算法更有效。正是由于这个原因，电子电路分析按缺省，就用一阶算法编程，并且最多提供二阶算法来解决暂态时间响应问题。此外，通过对 **tol** 设置更小的值，要达到更高的精度，没有必要使绝对误差更小。**tol** 设置每时间步的相对精度，不一定引起绝对误差减少。

总之，**不要盲目使用数值方法**。对于给定的问题，在决定最好的方法之前，要试一试各种可能的方法。有关微分方程数值解法的更进一步信息，请参考数值分析方面的书籍。有些参考书还提供了一些关于算法选择和如何处理那些时间常数变化范围大的病态方程的非常实用的信息。

13.7 M 文件举例

这里所介绍的《精通 MATLAB 工具箱》中的 M 文件可近似求解由采样值给出的函数的积分和微分。这里假定这些函数本身不存在，且独立变量也许不是线性间隔。例如，已装载到 MATLAB 中要分析的数据来源于实验测试。

对于所包含的数据缺乏函数描述，有许多种积分和微分的方法。如前所述，人们可以用最小二乘多项式拟合数据，然后在多项式的描述上进行操作。另一种方法是寻找数据的三次样条表示，然后运用《精通 MATLAB 工具箱》中的函数 **spintgrl** 和 **spderiv** 来分别寻找积分和微分的样条表示。这里所介绍的方法提供了另一种更简单的方法。积分用梯形规则计算。用加权中心差分计算微分。此外，将函数设计成在矩阵形式下工作，矩阵的列代表各与自变量有关的因变量。

正如这章前面所述，MATLAB 函数 **trapz** 计算在某有限区间的梯形积分。这里我们寻找的积分是自变量为 x 的函数。即如果 y=f(x)，我们寻找：

$$S(x) = \int_{x1}^{x} f(x)dx$$

式中的 x1 是向量 x 的第一个元素。用梯形规则，这个积分近似为：

$$S(x_k) = \sum_{i=1}^{k} 0.5(y_i + y_{i-1})(x_i - x_{i-1}) \quad \text{且 } S(x_1)=0$$

这样，第 k 个数据点的积分是上述梯形面积的累加和。函数 **mmintgrl** 实现的这个算法如下：

```
function z=mmintgrl(x , y)
%    MMINTgrl Compute Integral using Trapezoidal Rule.
```

```
%     MMINTGRL(X , Y) computes the integral of the function y=f(x) given the
%     data in X and Y. X must be a vector ,    but Y may be a column oriented
%     data matrix. The length of X must equal the length of Y if Y is a
%     vector ,    or it must equal the number of rows in Y if Y is a matrix.
%
%     X need not be equally spaced. The trapezoidal algorithm is used.
%
%     See also mmderiv

%     Copyrigth (c) 1996 by Prentice-Hall , Inc.

flag=0;                    %     falg is True if y is a row
x=x( : ); nx=length(x);    %     make x a column
[ry , cy]=size(y);
if ry==1&cy==nx , y=y .' ; ry=cy ; cy=1 ; flag=1 ; end
if nx~=ry ,    error(' X and Y not the right size ') , end
dx=x(2 : nx)-x(1 : nx-1);        %     width of each trapezoid
dx=dx( : , ones(1 , cy));        %     duplicate for each column in y
yave=(y(2 : ry , : )+y(1 : ry-1 , :))/2;                %     average of heights
z=[zeros(1 , cy); cumsum(dx .* yave)];        %     Use cumsum to find area
if flag , z=z'; end                                     %     if y was a row , return a row
```

在介绍上述函数的使用之前，考虑微分。在这种情况下，人们感兴趣的就是刚给定数据点的近似斜率。这里介绍一种下述的中心差分，而不是简单的向前或向后差分：

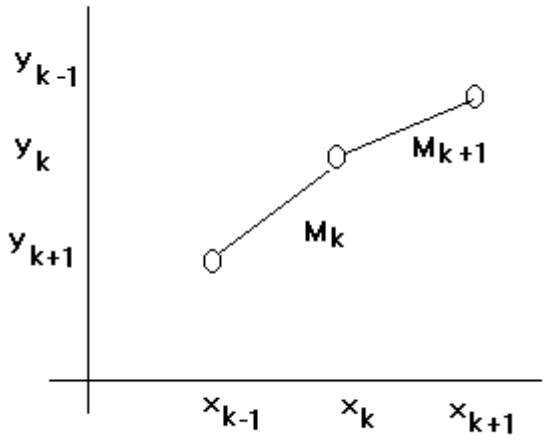


图 13.10 加权中心差分方法

从图 13.11 可知，在第 k 个点的近似微分是：

$$D(x_k) = (1-\alpha_k)M_k + (\alpha_k)M_{k+1} \qquad \text{式中} \alpha_k = \frac{x_k - x_{k-1}}{x_{k+1} - x_{k-1}},$$

并且 M_k 是连接 y_{k-1} 到 y_k 的直线的斜率。这样，第 k 点的微分是相邻两点间斜率的加权平均，离该点越近的点权越重。在第一个和最后一个数据点上，不能简单按照上述方法进行处理，因为这两个点都没有伴随的直线段。对于这些数据点，需要用另外的方法。这里所采取的方法是用二次多项式拟合前 3 个点（或最后 3 个点），并且计算这个多项式第一个（或最后一个）点的微分。函数 **mmderiv** 实现的这个算法如下：

```
function z=mmderiv(x , y)
%    MMDERIV Compute Derivative Using Weighted Central Differences.
%      MMDERIV(X , Y) computes the derivative of the function y=f(x) given the
%      data in X and Y. X must be a vector , but Y may be a column oriented
%      data matrix. The length of X must equal the length of Y if Y is a
%      vector ,    or it must equal the number of rows in Y if Y is a matrix.
%
%      X need not be equally spaced.Weighted central difference are used.
%      Quadratic approximation is used at the endpoints.
%
%      See also mmintgrl

%      Copyrigth (c) 1996 by Prentice-Hall , Inc.

flag=0;                      %      flag is True if y is a row
x=x( : ); nx=length(x);      %      make x a column
[ry , cy]=size(y);
if ry==1&cy==nx , y=y .'; ry=cy; cy=1; flag=1; end
if nx~=ry ,    error(' X and Y not the right size ') , end
if nx<3 ,    error(' X and Y must have st least three elements ') , end

dx=x(2 : nx)-x(1 : nx-1);              %      first difference in x
dx=dx+(dx==0)*eps;                     %      make infinite slopes finite
dxx=x(3 : nx)-x(1 : nx-2);             %      second difference in x
dxx=dxx+(dxx==0)*eps;          %      make infinite slopes finite
alpha=dx(1 : nx-2) ./ dxx                    %      central difference weight
alpha=alpha( : , ones(1 , cy));        %      duplicate for each column in y

dy=y(2 : ry , :)-y(1 : ry-1 , : );             %      first difference in y
dx=dx( : , ones(1 , cy));              %      duplicate dx for each column in y

%      now apply weighting to dy
z=alpha .* dy(2:ry-1 , :) ./ dx(2 : nx-1 , : )+(1-alpha) .* dy(1 : ry-2 , : ) ./ dx(1 : nx-2 , : );
```

```
z1=zeros(1 , cy)>=z1;
for i=1 : cy        %      fit quadratic at endpoints of each column
    p1=polyfit(x(1 : 3) , y(1 : 3 , i) , 2);                %      quadratic at first point
    z1(i)=2*p1(1)*x(1)+p1(2);                    %      evalute poly derivative
    pn=polyfit(x(nx-2 : nx) , y(ry-2 : ry , i) , 2);       %      quadratic at last point
    zn(i)=2*pn(1)*x(nx)+pn(2);                   %      evaluate poly derivative
end
z=[z1;    z;    zn];
if flag , z=z'; end                    %      if y was a row , return a row
```

最后，给出一个例子：

```
>>x=linspace(0 , 2*pi , 30)
>>y=sin(x);        %    create data
>>yi=mmintgrl(x , y);    %    find integral
>>yd=mmderiv(x , y);    %    find derivative
>>plot(x , y , x , yi , ' - ' , x , yd , ' : ')      %    plot results
```

注意这个积分定性地证明了等式：

$$\int_0^x \sin(x)dx = 1 - \cos(x)$$

而微分定性地证明了等式：

$$\frac{d}{dx}\sin(x) = \cos(x)$$

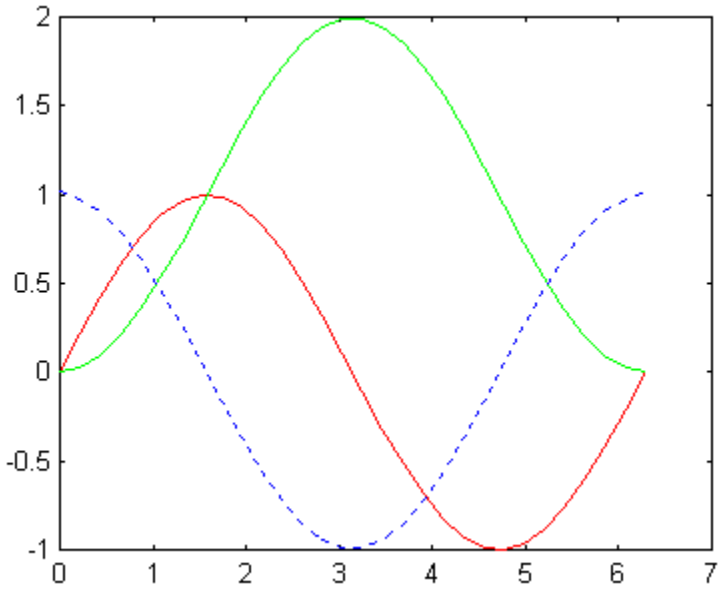


图 13.11 y=sin(x)极其积分、微分曲线

13.8 小结

表 13.1 总结了本章所讨论的函数。

表 13.1

数值分析函数	
fplot(‘ fname ‘ , [lb ub])	绘出上下限之间的函数
fmin(‘ fname ‘ , [lb ub])	寻找上下限内的标量最小值
fimis(‘ fname ‘ , xo)	寻找 **xo** 附近的向量最小值
fzero(‘ fname ‘ , xo)	寻找 **xo** 附近的标量函数的零点
trapz(x , y)	给定数据点 **x** 和 **y**，计算 y=f(x)下的梯形面积积分。
diff(x)	数组元素间的差分
[t , y]=ode23(‘ fname ‘ , to , tf , yo)	用 2 阶/3 阶龙格-库塔算法解微分方程组
[t , y]=ode45(‘ fname ‘ , to , tf , yo)	用 4 阶/5 阶龙格-库塔算法解微分方程组

第 14 章　富里哀分析

象富里哀级数，富里哀变换以及它们离散时间相应部分构成了信号处理的基础。为了便于这类问题的分析，MATLAB 提供了函数 **fft,ifft,fft2,ifft2** 和 **fftshift**。这类函数集执行一维和二维离散富里哀变换及其逆变换。这些函数允许人们完成很多信号处理任务。除此之外，还可在可选的信号处理工具箱中得到其他扩展的信号处理工具。

因为信号处理包含如此广泛的领域，甚至要说明用 MATLAB 中离散富里哀变换函数可解决的这类小问题，就超出了本书的范围。因此，这里将只介绍用函数 **fft** 近似连续时间信号的富里哀变换的一个例子。此外，还将讨论《精通 MATLAB 工具箱》中处理富里哀级数的函数集。

14.1 快速富里哀变换

在 MATLAB 中，函数 **fft** 计算一个信号的离散富里哀变换。在数据的长度是 2 的幂次或质因数的乘积的情况下，就用快速富里哀变换（FFT)来计算离散富里哀变换。当数据长度是 2 的幂次时，计算速度显著增加，因此，只要可能，选择数据长度为 2 的幂次或者用零来填补数据，使得数据长度等于 2 的幂次显得非常重要。在《MATLAB 参考指南》中可找到有关

该问题的讨论。

MATLAB 中实现的快速富里哀变换，是按照工科教材中常使用的方法。

F(k)=FFT{f(n)}

$$F(k)=\sum_{n=0}^{N-1} f(n)e^{-j2\pi nk/N} \qquad k=0,1,\cdots\cdots,N-1$$

因为 MATLAB 不允许零下标，所以移动了一个下标值。

$$F(k)=\sum_{n=1}^{N} f(n)e^{-j2\pi(n-1)(k-1)/N} \qquad k=1,2，\cdots\cdots,N$$

相应的逆变换为：

$$f(n)=FFT^{-1}\{F(K)\}$$

$$f(n)=\frac{1}{N}\sum_{k=1}^{N} F(k)e^{-j2\pi(n-1)(k-1)/N} \qquad n=1,2,\cdots\cdots,N$$

为了说明 FFT 的使用，考虑估计连续信号的富里哀变换的问题。

$$f(t)=\begin{cases}12e^{-3t} & t\geq 0\\ 0 & t<0\end{cases}$$

解析上，该富里哀变换为：

$$F(w)=\frac{12}{3+jw}$$

虽然在这种情况下，由于知道了富里哀变换的解析结果，再运用 FFT 没有多大的实用价值，但这个例子说明了对不常见的信号，特别是那些解析上难以找到富里哀变换的信号，一个估计富里哀变换的方法。下面的 MATLAB 语句用 FFT 估计 F(w)，并且用图形把所得到结果与上面的解析表达式的结果进行比较：

```
>>N=128;     %    choose a power of 2 for speed
>>t=linspace(0, 3, N);    %    time points for function evaluation
>>f=2*exp(-3*t);    %    evaluate the function and minimize aliasing:f(3)~0
>>Ts=t(2)-t(1);    %    the sampling period
>>Ws=2*pi/Ts;    %    the sampling frequency in rad/sec
>>F=fft(f);    %    compute the fft
>>Fp=F(1 : N/2+1)*Ts;
```

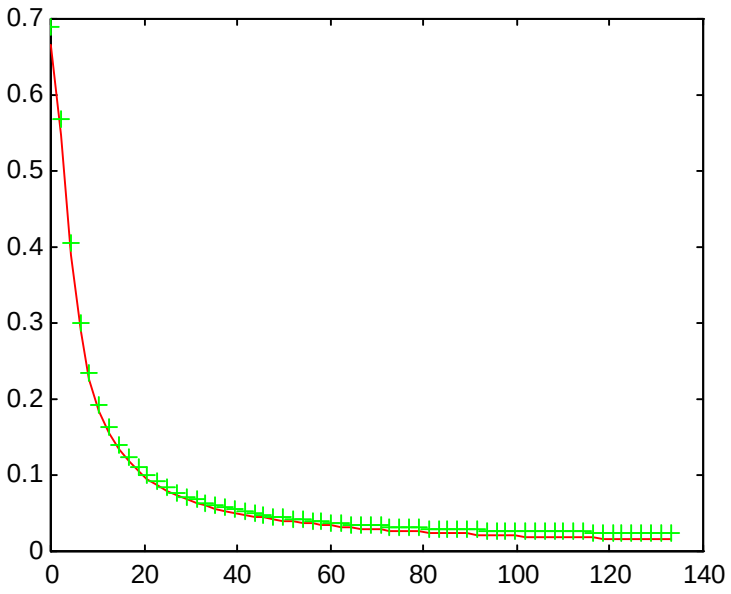


图 14.1 富里哀变换两种结果的比较

仅从 F 中取正频率分量，并且乘以采样间隔计算 F(w)。

```
>>W=Ws*(0 : N/2)/N
```

它建立了连续频率轴，该轴起始于 0，终止于奈魁斯特（Nyquist）频率 Ws/2，

```
>>Fa=2./(3+j*w);    %    evaluate analytical Fourier transform
>>plot(W, abs(Fa), W, abs(Fp), ' + ' )    %    generate plot, ' + ' mark fft results
>>xlabel(' Frequency, Rad/s '),ylabel(' |F(w)| ')
```

MATLAB 提供了大量的完成一般信号处理任务的函数。它们列于表 14.1：

表 14.1

信号处理函数	
conv	卷积
conv2	2 维卷积
fft	快速富里哀变换
fft2	2 维快速富里哀变换
ifft	逆快速富里哀变换
ifft2	2 维逆快速富里哀变换
filter	离散时间滤波器
filter2	2 维离散时间滤波器
abs	幅值
angle	四个象限的相角
unwrap	在 360° 边界清除相角突变
fftshift	把 FFT 结果平移到负频率上

14.2 富里哀级数

MATLAB 本身没有特别关于富里哀级数分析和处理的函数。不过，通过创建 M 文件函数，可容易加上这些函数。在这一节，将介绍《精通 MATLAB 工具箱》中富里哀级数函数。在介绍之前，首先定义实周期信号 f(t)的富里哀级数表示形式。

给出富里哀级数的**复指数形式**为：

$$f(t)=\sum_{n=\infty}^{\infty}F_n e^{jnw_0 t}$$

式中的富里哀级数的系数是：

$$F_n=\frac{1}{T_0}\int_t^{t+T_0}f(t)e^{-jnw_0 t}dt$$

且基频为$\omega_0=2\pi/T_0$。式中 T_0 满足 $f(t+T_0)=f(t)$。

给出富里哀级数的**三角数形式**为：

$$f(t)=A_0+\sum_{n=1}^{\infty}\{A_n\cos(n\omega_0 t)+B_n\sin(n\omega_0 t)\}$$

式中的富里哀级数的系数是：

$$A_0=\frac{1}{T_0}\int_t^{t+T_0}f(t)dt$$

$$A_n=\frac{2}{T_0}\int_t^{t+T_0}f(t)\cos(n\omega_0 t)dt$$

$$B_n=\frac{2}{T_0}\int_t^{t+T_0}f(t)\sin(n\omega_0 t)dt$$

且基频为$\omega_0=2\pi/T_0$。式中 T_0 满足 $f(t+T_0)=f(t)$。

表 14.2

精通 MATLAB 的富里哀级数函数	
fsderiv(Kn,Wo)	富里哀级数的微分
fseval(Kn,t,Wo)	计算富里哀级数
fsfind(‘ fname ‘,T,N)	寻找时间函数的富里哀级数的系数
[An,Bn,Ao]=fsform(Kn)	富里哀级数不同形式之间的转换
Kn=fsform(An,Bn,Ao)	
fsharm(Kn,i)	提取特殊的富里哀级数的谐波
fsmsv(Kn)	计算信号的均方根值
fsresize(Kn,N)	重新调整富里哀级数的系数向量的大小
fsresp(Num,Den,Un,Wo)	线性系统对输入富里哀级数 Un 的富里哀级数响应
fsround(Kn)	设置次要的富里哀级数的系数为 0
fswindow(N, ‘ type ‘)	产生一个窗口的函数，使吉布（Gibb）现象极小
fswindow(Kn, ‘ type ‘)	

上述两种形式中，一般复指数的富里哀级数更容易进行解析上处理。而三角富里哀级数则提供了更多的直观理解，更容易将正弦和余弦波形可视化。由于这个原因，《精通 MATLAB 工具箱》中的富里哀级数函数一般假定为复指数形式。不过，该工具箱提供了这两种富里哀级数形式之间转换的函数。表 14.2 总结了《精通 MATLAB 工具箱》中的富里哀级数函数。

因为有无穷多个谐波或富里哀级数的系数，有必要对富里哀级数截尾，只考虑有限个谐波。如果考虑 N 个谐波，MATLAB 中富里哀级数表示成一个长度为 2N+1 的行向量，向量中的元素为复指数的富里哀级数的系数。该向量包含以升序排列的富里哀级数系数，即：

$$F = [F_{-N}\ F_{-N+1} \cdots F_{-1}\ F_0\ F_1 \cdots F_{N-1}\ F_N]$$

为了说明这些函数的使用，考虑寻找锯齿信号的富里哀级数表示（见图 14.2）。

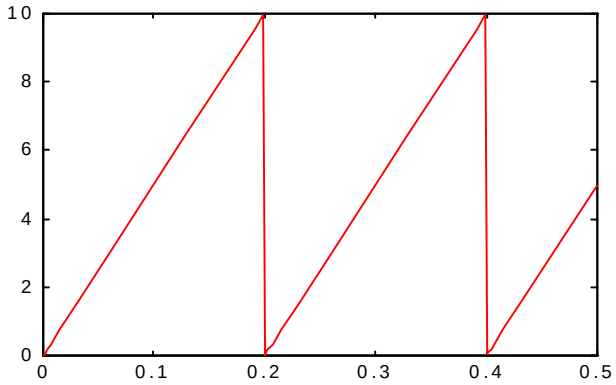


图 14.2 锯齿信号

虽然很容易找到这个信号的富里哀级数的系数，但可用函数 **fsfind** 近似求解这些系数。

```
function [Fn, nwo, f, t]=fsfind(fun, T, N, P)
%    FSFIND Find Fourier Series Approximation.
%     Fn=FSFIND(FUN, T, P) computes the Complex Exponential
```

```
%     Fourier Series of a signal described by the function ' FUN '.
%     FUN is the character string name of a user created M-file function.
%     the unction is called as f=FUN(t) where t is a vector over
%     the range 0<=t<=T.
%
%     The FFT is used. Choose sufficient harmonics to minimize aliasing.
%
%     T is the period of the function. N is the number of harmonics.
%     Fn is the vector of FS coefficients.
%
%     [Fn, nWo, t, p]=FSFIND(FUN, T, P) returns the frequencies associated
%     with Fn in nWo and return values of the function FUN
%     in f evaluated at the points int t over the range 0<=t<=T.
%
%     FSFIND(FUN, T, P) passes the data in P to function FUN as
%     f=FUN(t, p). This allows parameters to be passed to FUN.
%     Copyrigth (c) 1996 by Prentice-Hall,Inc.

n=2*N;
t=linspace(0, T, n+1);
if nargin==3
     f=feval(fun, t);
else
     f=feval(fun, t, P);
end
Fn=fft(f(1 : n));
Fn=[0    conj(Fn(N : -1 : 2))    Fn(1 : N)    0]/n;
nwo=2*pi/T*(-N : N);
```

根据上述描述，必须先建立一个函数以计算周期内锯齿信号值。这样，我们有：

```
function f=sawtooth(t, T)
%    SAWTOOTH Sawtooth Waveform Generation.
%     SAWTOOTH(t, T) computes values of a sawtooth having a
%    period T at the values in t.
%
%    Used in Fourier series examples.
f=10*rem(t, T)/T;
```

现在求近似富里哀级数的系数。

```
>>T= .2;        %    desired period
>>N=25;      %    number of harmonics
>>Fn=fsfind(‘ sawtooth ‘, T, N, T);      %    compute Fourier series coefficients
```

因为用 FFT 求近似系数，所以得到的系数并不精确。不过，如果有足够多的系数，特别是如果时间函数本身，或者其一阶，或者二阶导数连续（锯齿信号不连续），误差就比较小。如果需要更高的精度，可调用积分程序来计算每一个富里哀级数系数的积分关系式。后一种方法需要很长的时间，因为 FFT 一次要近似所有的系数。不过，尽管有误差，对于许多应用，FTT 的近似结果已足够准确了。

用函数 **fseval** 可实现富里哀级数的计算。

```
function y=fseval(kn, t, wo)
%    FSEVAL Fourier Series Function Evaluation.
%      FSEVAL(Kn, t, Wo) computes values of a real valued function given
%      its complex exponential Fourier series coefficients Kn, at the
%      points given in t where the fundamental frequency is Wo rad/s.
%      K contains the Fourier coefficentsin ascending order:
%      Kn = [k        k        ...        k        ...        k        k]
%                    -N       -N+1          0             N-1    N
%      if Wo is not given, Wo=1 ia assumed.
%      Note: this function creates a matrix of size:
%      rows = length(t) and columns = (length(K)-1)/2

%      Copyrigth (c) 1996 by Prentice-Hall,Inc.

if    nargin==2, wo=1; end
nk=length(kn);
if    rem(nk-1, 2) | (nk==1)
      erro(‘ Number of element in K must be odd and greater than 1 ‘)
end
n=0.5*(nk-1); %      highest harmonic
nwo=wo*(1 : n);    %      harmonic frequencies
ko=kn(n+1);   %      average value
kn=kn(n+2 : nk) ‘; %      positive frequency coefs
y=ko+2*(real(exp(j*t( : )*nwo)*kn))' ;
```

遵照所需的语法，我们得到：

```
>>n=-N : N;    %    harmonic index
>>Fna=j*5 ./ (pi*n);    %    actual Fourier series coefficients
>>Fna(N+1)=5;      %    poke in average value
>>t=linspace(0 ,    .4);    %    time points to evaluate functions
>>wo=2*pi/T;      %    fundamental frequency
>>f=fseval(Fn, t, wo);    %    evaluate approximated Fourier series
>>fa=fseval(Fna, t, wo);    %    evaluate actual Fourier series
>>plot(t, f, t, fa)    %    plot results for comparison
```

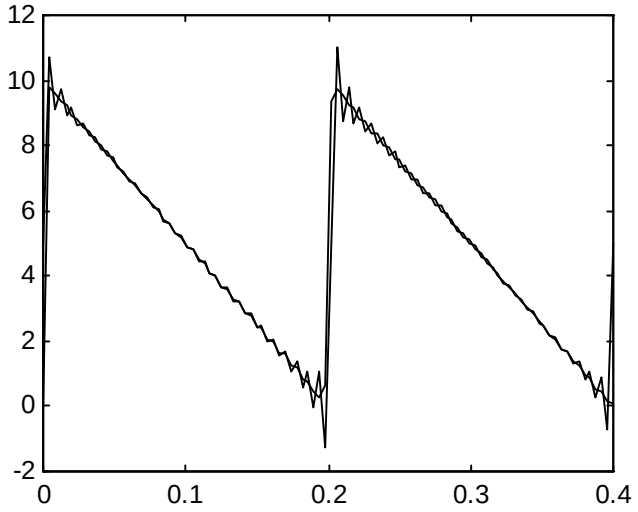


图 14.3 富里哀级数的结果比较

注意上述两个富里哀级数给出了相似的结果。从图 14.3 中可知，在锯齿信号的不连续点的周围出现了明显的吉布现象波纹。用函数 **fswindow**，将一个**窗口**加于富里哀级数系数，就能减少这种波纹。

```
>>help fswindow

FSWINDOW Generate Window Functions.
  FSWINDOW(N, TYPE) creates a window vector of type TYPE having
  a length equal to the scale N.
  FSWINDOW(X, TYPE) creates a window vector of type TYPE having
  a length and orientation the same as the vector X.
  FSWINDOW(X, TYPE, alpha) porvides a parameter alpha as required
  for some window types.
  FSWINDOW with no input arguments returns a string matrix whose
  i-th row is the i-th TYPE given below.
  TYPE is a string designating the window type desired:
  'rec' = Rectangular or Boxcar
  'tri' = Triangular or Bartlet
  'han' = Hann or hanning
  'ham' = Hamming
  'bla' = blackman common coefs.
  'blx' = Blackman exact coefs.
  'rie' = Rieman {sin(x)/x}
  'tuk' = Tukey, 0<alpha<1; alpha = 0.5 is default
  'poi' = Poisson, 0<alpha<inf; alpha = 1 is default
  'cau' = Cauchy, 1<alpha<inf; alpha = 1 is default
  'gau' = Gaussian, 1<alpha<inf; alpha = 1 is default
Reference: F. J. Harris,"On the Use of Windows for Harmonic Analysis with the Discrete
```

Fourier Transform," IEEE Proc. Vol. 66, no. 1, Jan. 1978, pp.51-83

```
>>Fnh=Fn.*fswindow(Fn, ' han ');     %    apply Hanning window
>>f=fseval(Fnh, t, wo);     %    evaluate windowed FS
>>plot(t, f)     %    plot results
```

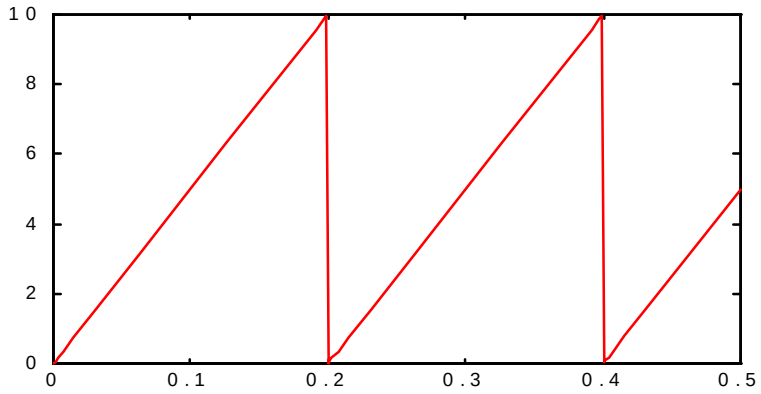


图 14.4　用汉宁窗口滤波后的富里哀级数

因为乘以窗口函数，改变了富里哀级数的系数，所以，现在图形（见图 14.4）看起来好多了。

接下来，让我们把锯齿波加到传递函数为 $H(s)=\frac{60}{s+60}$ 的线性系统中。

函数 **fsresp** 专门解决此类问题。

```
function y=fsresp(num, den, un, wo)
%    FSRESP Fourier Series Linear System Response
%    FSRESP(N, D, Un, Wo) returns the comples exponential FS of the
%    output of a linear system when the input is given by a FS
%    N and D are the numerator and denominator coefficients
%    respectively of the system transfer function.
%    Un is the complex exponential Fourier Series of the system input.
%    Wo is the fundamental frequency associated with the input.
%    Copyright (c) 1996 by Prentice-Hall, Inc.
if nargin<4, wo=1; end
N=(length(un)-1)/2;        %      highest harmonic
jnWo=sqrt(-1)*(-N : N)*wo; %      frequencies of harmonics
y=(polyval(num, jnWo) ./ polyval(den, jnWo)).*un;    %      output is Yn=H(jnWo)*Un
```

运用 **fsresp**，我们得到：

```
>>Yn=fsresp(60, [1    60], Fn, wo);    %    find response coefficients
>>y=fseval(Yn, t, wo);    %    evaluate output
>>plot(t, f, f, y)    %    plot system input and output
```

结果见图 14.5。

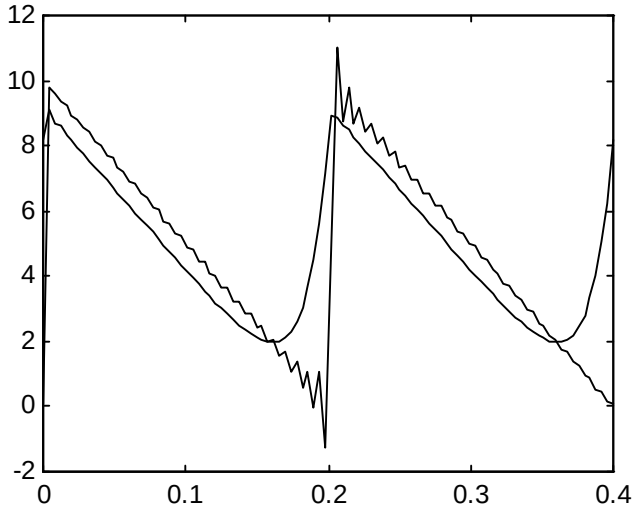


图 14.5　线性系统的输入输出曲线

作为最后一个例子，运用 **fsform** 把锯齿信号的富里哀级数转换为三角形式。

```
function [a, b, ao]=fsform(c, d, co)
%    FSFORM Fourier Series Format Conversion
%      KN=FSFORM(An, Bn, Ao) converts the trigonometric FS with
%      An being the CSOINE and Bn being the SINE coefficients to
%      the complex exponential FS with coefficients Kn.
%      Ao is the DC component and An,Bn and Ao are assumed to be real.
%
%      [Kni]=FSFORM(An, Bn, Ao) returns the index vector i that
%      identifies the harmonic number of each element of Kn.
%
%      [An, Bn, Ao]=FSFORM(Kn) does the reverse format conversion.

%      Copyrigth (c) 1996 by Prentice-Hall,Inc.

nc=length(c);
if nargin==1 %      complex exp -> trig form
     if rem(nc-1, 2)|(nc==1)
      error(' Number of elements in K must bo odd and greater than 1 ')
     end
     nn=(nc+3)/2;
     a=2*real(c(nn : nc));
     b=-2*imag(c(nn : nc));
     ao=real(c(nn-1));
elseif    nargin==3      %    trig from -> complex exp form
```

```
    nd=length(d);
    if nc~=nd;
        error(' A and B must be the same length ')
    end
    a=0.5*(c-sqrt(-1)*d);
    a=[conj(a(nc : -1 : 1)) co(1) a];
    b=-nc : nc;
else
    error(' Improper number of input arguments. ')
end

>>[An, Bn, Ao]=fsform(Fn)
An =
  Columns 1 through 7
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 8 through 14
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 15 through 21
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 22 through 25
   -0.2000   -0.2000   -0.2000         0
Bn =
  Columns 1 through 7
   -3.1789   -1.5832   -1.0484   -0.7789   -0.6155   -0.5051   -0.4250
  Columns 8 through 14
   -0.3638   -0.3151   -0.2753   -0.2418   -0.2130   -0.1878   -0.1655
  Columns 15 through 21
   -0.1453   -0.1269   -0.1100   -0.0941   -0.0792   -0.0650   -0.0514
  Columns 22 through 25
   -0.0382   -0.0253   -0.0126         0
Ao =
    4.9000
```

忽略平均值，这个锯齿应展示了奇对称特性，所有 An 系数应当为 0。显然，它们不为 0。发生这种情况的原因是因为我们使用了 FFT 来近似系数。此外，Ao 平均值有一点儿误差，它的值应当是 5.0。

14.3 小结

下面的两个表总结了本章所讨论的函数。

表 14.3

信号处理函数	
conv	卷积
conv2	2 维卷积
fft	快速富里哀变换
fft2	2 维快速富里哀变换
ifft	逆快速富里哀变换
ifft2	2 维逆快速富里哀变换
filter	离散时间滤波器
filter2	2 维离散时间滤波器
abs	幅值
angle	四个象限的相角
unwrap	在 360°边界清除相角突变
fftshift	把 FFT 结果平移到负频率上
nextpow2	2 的下一个较高幂次

表 14.4

精通 MATLAB 的富里哀级数函数	
fsderiv(Kn,Wo)	富里哀级数的微分
fseval(Kn,t,Wo)	计算富里哀级数
fsfind(‘ fname ‘,T,N)	寻找时间函数的富里哀级数的系数
[An,Bn,Ao]=fsform(Kn) Kn=fsform(An,Bn,Ao)	富里哀级数不同形式之间的转换
fsharm(Kn,i)	提取特殊的富里哀级数的谐波
fsmsv(Kn)	计算信号的均方根值
fsresize(Kn,N)	重新调整富里哀级数的系数向量的大小
fsresp(Num,Den,Un,Wo)	线性系统对输入富里哀级数 Un 的富里哀级数响应
fsround(Kn)	设置次要的富里哀级数的系数为 0
fswindow(N, ‘ type ‘) fswindow(Kn, ‘ type ‘)	产生一个窗口的函数，使吉布（Gibb）现象极小

第 15 章　低级文件 I/O

对于大多数用户，MATLAB 函数 **load** 和 **save** 为装载和存储数据提供了足够的工具。利用以扩展名为**.mat** 结尾的文件名，**load** 和 **save** 假定数据是以与平台无关的二进制格式保存，或者用称之为 flat 的简单的 ASCII 文件格式保存。当 flat ASCII 或**.mat** 这两种格式还不够时，MATLAB 提供了基于 C 语言实现的低级文件 I/O 函数。用这些低级文件 I/O 函数，MATLAB 可以读写你所知道的任意文件格式。例如，知道电子文件或数据库程序所使用的格式，就可以把这些数据读进 MATLAB 的矩阵中去。类似地，也可以创建电子文件或数据库文件。

MATLAT 中这种基本的低级文件 I/O 命令如下：

表 15.1

MATLAB 低级文件 I/O 函数	
fclose：	关闭文件
feof：	测试文件结束
ferror：	查询文件 I/O 的错误状态
fgetl：	读文件的行，忽略回行符
fgets：	读文件的行，包括回行符
fpoen：	打开文件
fprintf：	把格式化数据写到文件或屏幕上
fread：	从文件中读二进制数据
frewind：	返回到文件开始
fscanf：	从文件中读格式化数据
fseek：	设置文件位置指示符
ftell：	获取文件位置指示符
fwrite：	把二进制数据写到文件里

除了这些函数外，所具有的 MATLAB 版本可能为一个或多个公用软件包提供读写文件的特定函数 M 文件。有关这些函数的进一步的信息，请使用在线帮助：**>>help iofun**

第 16 章　调试工具

在开发函数 M 文件过程中，不可避免地出现错误，即故障。MATLAB 提供了很多函数和方法，帮助调试函数。

在 MATLAB 表达式中，有两类错误：语法错误和运行错误。当 MATLAB 计算一个表达式的值或一个函数被编译到内存时会发现语法错误。一旦发现语法错误，MATLAB 立即标志这些错误，并提供有关所遇到的错误类型，以及发生错误处 M 文件的行数。给定这些反馈信息，就很容易纠正这些错误。

而另一方面，即使 MATLAB 标志了运行错误，但找出错误一般比较困难。当发现运行错误时，MATLAB 把控制权返回给命令窗口和 MATLAB 的工作空间。失去了对发生错误的函数空间的访问权，因此，用户不能询问函数工作空间中的内容排除问题。

根据作者的经验，当一些操作结果导致空矩阵或 **NaNs** 时，最容易发生运行错误。所有有关 **NaNs** 的操作都返回 **NaNs** 值。因此，如果有可能出现 **NaNs** 结果，则当出现 **NaNs** 时，最好运用逻辑函数 **isnan** 来执行一些缺省操作。因为空矩阵为零维，所以对空矩阵寻址常常导致错误。函数 **find** 表示了可产生空矩阵结果的一般情况。如果函数 **find** 的空矩阵输出用于索引其它数组，所返回的值也将是空的。这样，空矩阵具有传播性质。例如：

```
>>x=pi*(1 : 4)       %    example data

>>i=find(x>20)    %    use find function

>>y=2*x(i)          %    propagate the empty matrix
```

很清楚，当希望 y 具有有限维数和值时，可能发生运行错误。当执行一个操作或使用可返回空结果的函数时，逻辑函数 **isempty** 有利于为空矩阵定义一个缺省值，这样避免运行错误。

有几种调试函数 M 文件的方法。对于简单的问题，可直接使用下列的方法组合：

1、去掉文件中所选择的行的分号，以便中间结果显示在命令窗口中。

2、在文件中加入显示感兴趣的变量的语句

3、把 **keyboard** 命令放在文件中所选择的地方，给键盘暂时控制权。这样，可以查询函数空间并按需要改变其值。

4、在 M 文件开始，在 function 语句前加上%，将函数 M 文件改变为脚本 M 文件。当 MATLAB 执行该脚本 M 文件时，该空间就是 MATLAB 工作空间。这样，当发生错误时可以询问。

当 M 文件大，递归调用或者多次嵌套（即调用其它 M 文件函数，被调用 M 文件函数又调用其它 M 文件函数，等等）时，用 MATLAB 的调试函数会更方便。与上述所列方法相反，这些调试函数不要求将有问题的 M 文件进行编辑。表 16.1 所给出的这些函数类似于其它高级编程语言中所提供的函数。有关进一步的信息，以及它们的使用实例，参阅《MATLAB 用户指南》。

表 16.1

MATLAB 调试函数	
dbclear：	取消断点
dbcont：	在断点后恢复运行
dbdown：	工作空间下移
dbquit：	退出调试模式
dbstack：	列出谁调用谁
dbstatus：	列出所用的断点
dbstep：	执行一行或多行
dbstop：	设置断点
dbtype：	列出带行号的 M 文件
dbup：	工作空间上移

第十八章　三维图形

为了显示三维图形，MATLAB 提供了各种各样的函数。有一些函数可在三维空间中画线，而另一些可以画曲面与线格框架。另外，颜色可以用来代表第四维。当颜色以这种方式使用时，由于它不再象照片中那样显示信息的自然属性----色彩，而且也不是基本数据的内在属性，所以它称作**伪彩色**。为了简化对三维图形的讨论，对颜色的介绍推迟到下一章。在这一章，主要讨论绘制三维图形的基本概念。

18.1 函数 plot3

plot3 命令将绘制二维图形的函数 **plot** 的特性扩展到三维空间。函数格式除了包括第三维的信息（比如 z 方向）之外，与二维函数 **plot** 相同。**plot3** 一般语法调用格式是 **plot3($x_1,y_1,z_1,S_1,x_2,y_2,z_2,S_2,\cdots$)**，这里 x_n,y_n 和 z_n 是向量或矩阵，S_n 是可选的字符串，用来指定颜色、标记符号和/或线形。

总的来说，**plot3** 可用来画一个单变量的三维函数。如下为一个三维螺旋线例子：

```
» t=0:pi/50:10*pi;
» plot3(sin(t),cos(t),t)
» title( ‘ Helix ‘ ),xlabel( ‘ sint(t) ‘ ),ylabel( ‘ cos(t) ‘ ),zlabel( ‘ t ‘ )
» text(0,0,0, ‘ Origin ‘ )
» grid
» v = axis
v =
     -1      1     -1      1      0     40
```

输出见图 18.1.

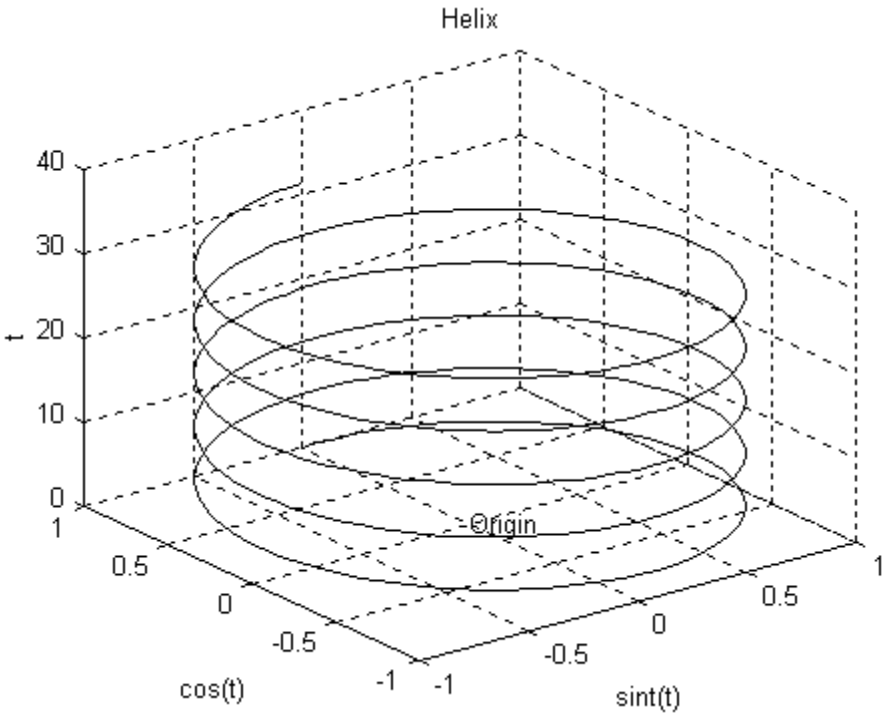


图 18.1　螺旋线图

从上例可明显看出，二维图形的所有基本特性在三维中仍都存在。**axis** 命令扩展到三维只是返回 Z 轴界限（0 和 40），在数轴向量中增加两个元素。函数 **zlabel** 用来指定 z 轴的数据名称，函数 **grid** 在图底绘制三维网格。函数 **test(x,y,z,** ‘ **string** ‘ **)**在由三维坐标 **x,y,z** 所指定的位置放一个字符串。另外，子图和多图形窗口可以直接应用到三维图形中。

在最后一章可以看到，通过指定 **plot** 命令的多个参量或使用 **hold** 命令，可以把多条直线或曲线重叠画出。**plot3** 以及其它的三维图形函数都可以提供相同的能力。例如，增加维数的 **plot3** 命令可以使多个二维图形沿一个轴排列起来，而不是直接将二维图形叠到另一个的上面。

```
» x=linspace(0,3*pi);    %    x-axis data
» z1=sin(x);    %    plot in x-z plane
» z2=sin(2*x);
» z3=sin(3*x);
» y1=zeros(size(x));    %    spread out along y-axis
» y3=zeros(size(x));    %    by giving each diffent y-axis values
» y2=y3/2;
» plot3(x,y1,z1,x,y2,z2,x,y3,z3);
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ,abel( ‘ z-axis ‘ )
» title( ‘ sin(x),sin(2x),sin(3x) ‘ )
```

输出见图 18.2.

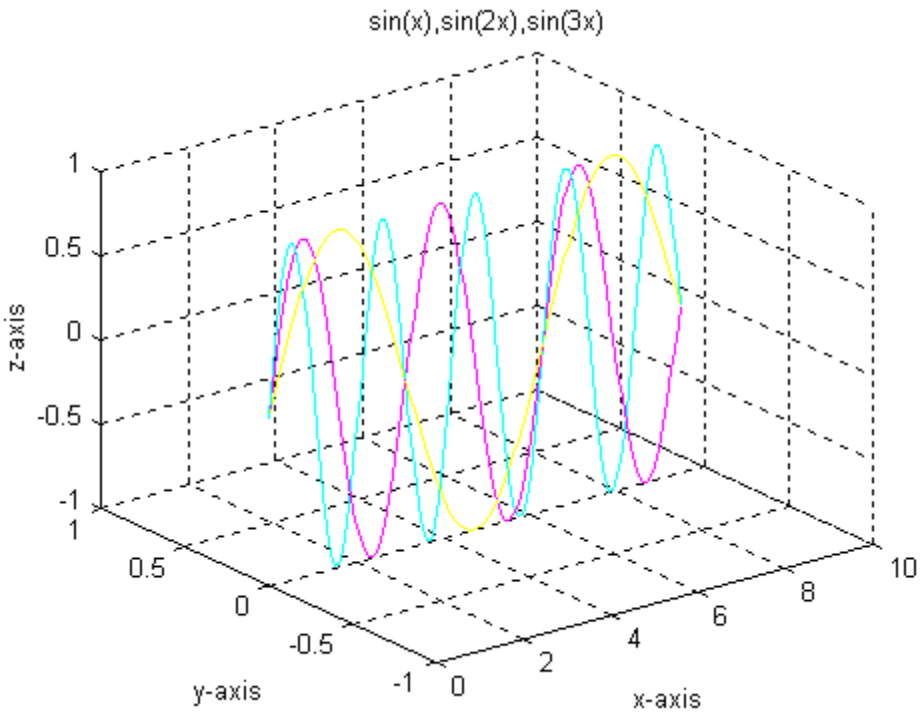


图 18.2　正弦曲线图

上述图形也可以沿另外方向堆列。

```
» plot3(x,z1,y1,x,z2,y2,x,z3,y3)
» grid,xlabel( ‘ x-axis ‘ ,ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ sin(x),sin(2x),sin(3x) ‘ )
```

输出见图 18.3.

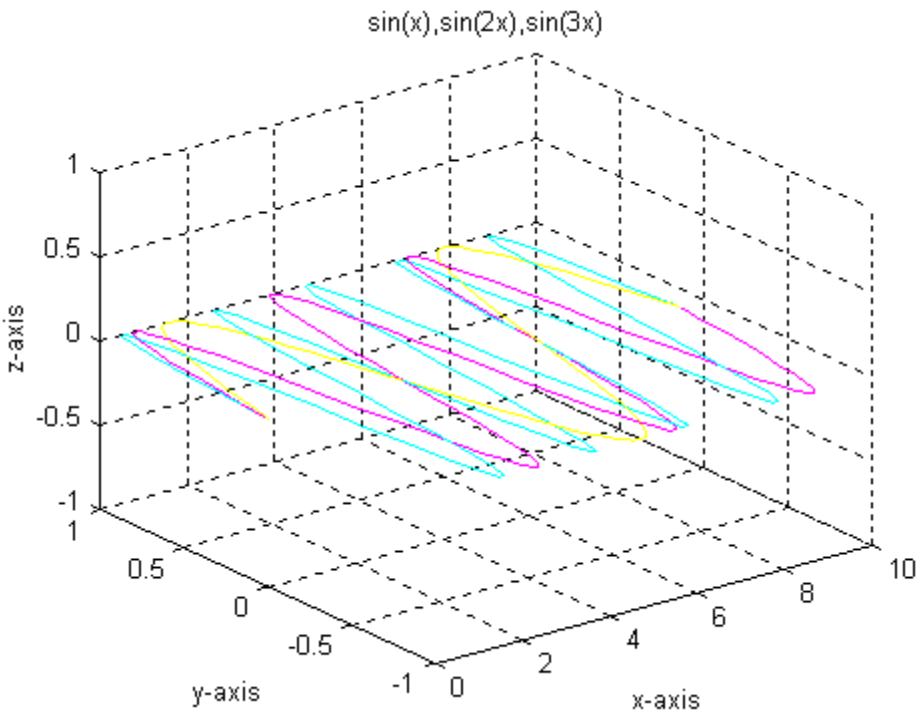


图 18.3　正弦曲线图

18. 2 改变视角

注意两个图形，一个是以 30 度视角向下看 z=0 平面,一个是以 37.5 度视角向上看 x=0 平面。这是对所有三维图形的缺省视角。与 z=0 平面所成的方向角叫**仰角**，与 x=0 平面的夹角叫做**方位角**。这样，缺省的三维视角方向仰角为 30 度，方位角为-37.5 度。而缺省的二维视角仰角为 90 度，方位角为 0 度。仰角和方位角的概念在图 18.4 中形象地画出。

图 18.4　仰角和方位角示意图

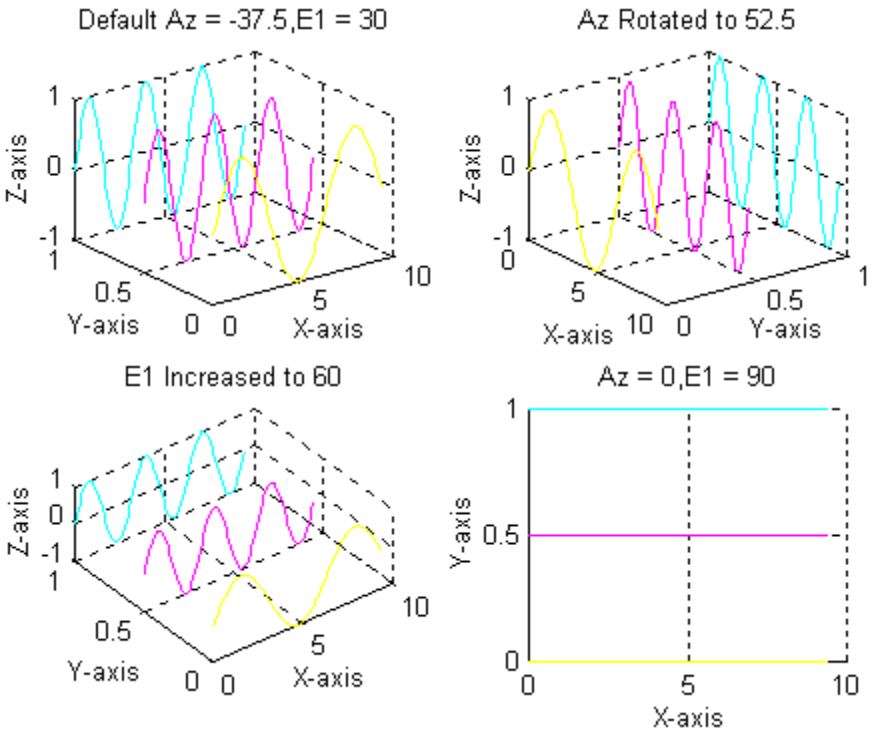


图 18.5　视角举例图

在 MATLAB 中，函数 **view** 改变所有类型的二维和三维图形的图形视角。**view(az,el)**和**view([az,el)]**将视角改变到所指定的方位角 **az** 和仰角 **el**。考虑下面脚本 M 文件形式的例子。

```
%    viewpoint example using subplots

x=linspace(0,3*pi).';

Z=[sin(x)    sin(2*x)    sin(2*x)];    %    create Y and Z axes as matrices

Y=[zeros(size(x))    ones(size(x))/2    ones(size(x))];

subplot(2,2,1)
```

```
plot3(x,Y,Z)    %    plot3 works with column-oriented matrices too
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ Default Az = -37.5,E1 = 30 ‘ )
view(-37.5,30)

subplot(2,2,2)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ Az Rotated to 52.5 ‘ )
view(-37.5+90,30)

subplot(2,2,3)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ E1 Increased to 60 ‘ )
view(-37.5,60)

subplot(2,2,4)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
title( ‘ Az = 0,E1 = 90 ‘ )
view(0,90)
```

输出见图 18.5。

除了上面的形式，**view** 还提供了综合在表 18.1 的其它特性：

表 18.1

函数 view	
view(az,el)	将视图设定为方位角 **az** 和仰角 **el**
view([az,el])	
view([x,y,z])	在笛卡儿坐标系中将视图设为沿向量**[x,y,z]**指向原点，例如 **view([0 0 1])=view(0,90)**
view(2)	设置缺省的二维视角，**az=0**，**el=90**
view(3)	设置缺省的三维视角，**az=-37.5**，**el=30**
[az,el]=view	返回当前的方位角 **az** 和仰角 **el**
view(T)	用一个 4×4 的转矩阵 **T** 来设置视图角
T=view	返回当前的 4×4 转矩阵

最后，为了演示 MATLAB 句柄图形能力，精通 *MATLAB* 工具箱包含了函数 **mmview3d**。在产生二维或三维图形后调用此函数，**» mmview3d**，在当前图形中放置水平角和方位角滑标（滚动条）以设置视角。使用函数 **mmview3d** 的更详细的信息见在线帮助。

18.3 两个变量的标量函数

相对于 **plot3** 产生的线条图形，经常希望画出两个变量的标量函数，比如：

z=f(x,y)

这里每一对 x 与 y 的值产生一个 z 的值。它作为 x 与 y 的函数，是三维空间中的一个曲面。为了在 MATLAB 里画出这个曲面，z 的值存放在一个矩阵中。象在**二维插值这一节所描述的那样，给出 **x** 与 **y** 的值作为独立的变量，**z** 是因变量矩阵，**x**、**y** 与 **z** 的联系就是：

z(i,:)=f(x,y(i)) and z(:,j)=f(x(j),y)

即 **z** 的第 i 行与的 **y** 第 i 个元素相关，而 **z** 的第 j 列与 **x** 的第个 j 元素相关。或者说，**y** 沿着 z 的列变化，而 **x** 沿着 **z** 的行变化。

当 z=f(x,y)能简化表示时，可以方便地用数组运算在单个语句中算出 z 的所有的值。这样做，要求我们以合适的方向创建所有 x 与 y 值的矩阵。（这种方向有时被 *Mathwork* 公司称作为**方格**)。MATLAB 提供了函数 **meshgrid** 来执行这个步骤。

```
» x=-3:3;   %   choose x-axisd values
» y=1:5;   %   y-axis values
» [X,Y]=meshgrid(x,y)
X =
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
Y =
      1      1      1      1      1      1      1
      2      2      2      2      2      2      2
      3      3      3      3      3      3      3
      4      4      4      4      4      4      4
      5      5      5      5      5      5      5
```

如上所见，函数 **meshgrid** 对 **y** 中行的每一行复制 **x**，同样也对 **x** 中列的每一列复制 **y**。这种方向与前面语句相一致，即 **y** 向下改变其列，而 **x** 横跨改变其行。给定 **X** 和 **Y**，如果 $z=f(x,y)=(x+y)^2$，那么 z 便定义一个三维曲面的数据矩阵：

```
» Z=(X+Y).^2
Z =
      4      1      0      1      4      9     16
      1      0      1      4      9     16     25
      0      1      4      9     16     25     36
```

```
 1     4     9    16    25    36    49
 4     9    16    25    36    49    64
```

当函数不能象上面那样简单表示出来时，便只能用 For 循环或 While 循环来计算 **Z** 的元素。在很多情况下，有可能按行或按列计算 **Z**。例如，如果能按行计算 **Z**，下面的脚本文件段会很有帮助：

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for r=1:nx
    {preliminary commands}
    Z(r,:)={a function of y and x(r) defining r-th row of Z}
end
```

相反，如果能按列计算 **Z**，下面的脚本文件段会很有帮助：

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for c=1:ny
    {preliminary commands}
    Z(:,c)={a function of y(c) and x defining c-th column of Z}
end
```

只有当 **Z** 中的元素必须一个一个地计算时，就常常要求象下面的脚本文件一样,用嵌套循环进行计算。

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for r=1:nx
```

```
    for c=1:ny
        {preliminary commands}
        Z(r,c)={a function of y(c) and x(r) defining Z(r,c)}
    end
end
```

18.4 杂乱或散射数据的插值

在有些情况下，两个变量的标量函数的值，如 z=f(x,y)，不能简单地算出。这是因为要么 x 和 y 的值是非均匀间隔的（最坏时是随机分布），要么是用了不同的坐标系，比如非长方形的网格。出现这些情况时，MATLAB 中的函数 **griddata** 就用来产生经插值后的均匀间隔数据以作图。首先考虑前面的例子。假设要求较高的分辨率，但我们不想重新计算函数来得到新值。

```
» x=-3:3;   %   original x-axis values
» y=1:5;   %   original y-axis values
» [X,Y]=meshgrid(x,y);   %   create plaid data matrices
» Z=(X+Y).^2;   %   original z values
» size(Z)   %   original array size
ans =
     5     7
» xi=-3:.5:4;   %   interpolated x-axis values
» length(xi)   %   get new x-axis length
ans =
    15
» yi=0:.2:5;   %   interpolated y-axis values
» length(yi)   %   get new y-axis length
ans =
    26
» [Xi,Yi]=meshgrid(xi,yi);   %   make new data plaid
» Zi=griddata(X,Y,Z,Xi,Yi);   %   interpolated Z data using original data
» size(Zi)   %   interpolated size is correct
ans =
    26    15
```

这里函数 **griddata** 用三个原始矩阵 **X，Y，Z** 和需要插值的方格矩阵，创建一个新的因变量矩阵 **Zi**。要注意插值不必在原始数据的范围内，比如 **x** 在-3 与 3 间变化，而 **xi** 在-3 与 4 间变化。

与第 11 章里所述的二维插值相对应，**griddata** 对于数据非单调或不规则分布的情况也同样有效。例如，考虑随机数据：

```
» x=2*rand(1,20);   %   nonmonotonic x-axis
» y=4*rand(1,20)-2;   %   nonmonotonic y-axis
```

```
» [X,Y]=meshgrid(x,y);  %  make data plaid
» Z=(X+Y).^2;  %  compute function
» xi=linspace(0,2,50);  %  interpolated monotonic x-axis values
» yi=linspace(-2,2,30);  %  interpolated monotonic y-axis values
» [Xi,Yi]=meshgrid(xi,yi);  %  make data plaid
» Zi=griddata(X,Y,Z,Xi,Yi);  %  interpolated on monotonic plaid data
```

这里，对随机数据插值给出更适合于作图的单调数据。这对于本章后面讨论的等值线图特别重要，因为它要求数据定义在均匀间隔的网格中。

在上面的两个例子中，较容易对所需的插值重新计算函数。作为一般规则，如果容易计算出感兴趣的函数，就要避免使用函数 **griddata**。而当感兴趣的点的函数不可得到或需要很大的计算量时，函数 **griddata** 提供了一个工具，以均匀分布的内插数据点，对基本函数进行估计。

18.5 网格图

利用在 **x—y** 平面的矩形网格点上的 **z** 轴坐标值，MATLAB 定义了一个**网格**曲面。MATLAB 通过将邻接的点用直线连接起来形成网状曲面，其结果好象在数据点有结点的鱼网。例如，用 MATLAB 的函数 **Peaks** 可以画一个简单的曲面。

```
» [X,Y,Z]=peaks(30);
» mesh(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ MESH of PEAKS  ‘ )
```

输出见图 18.6.

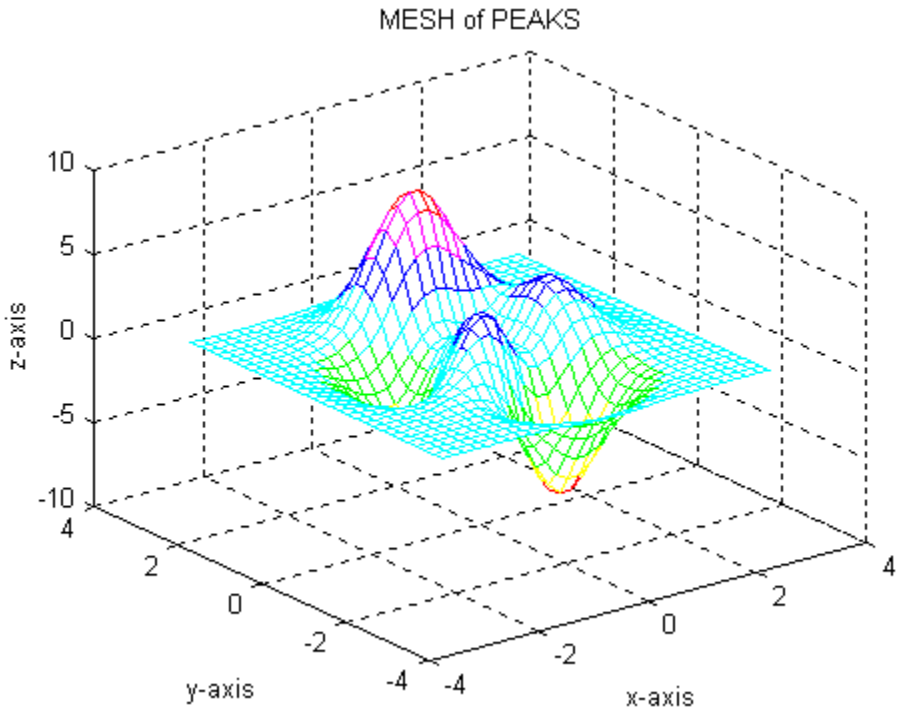


图 18.6　函数 PEAKS 的网格图

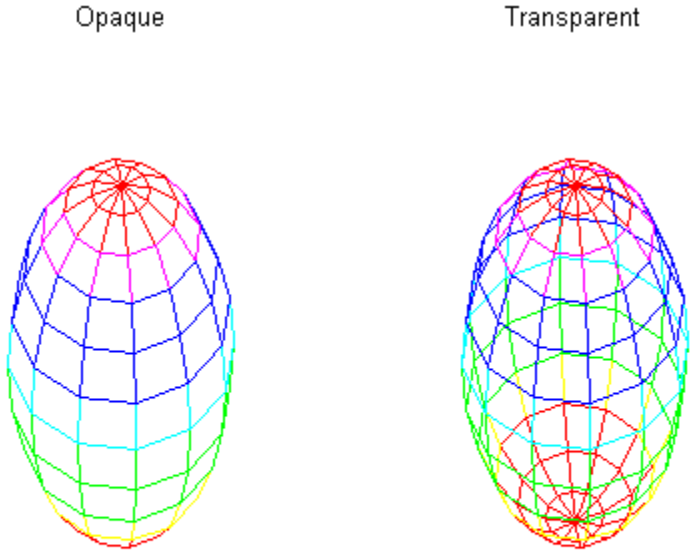
在显示器上要注意到线的颜色与网格的高度有关。一般情况下，函数 **mesh** 有可选的参量来控制绘图中所用的颜色。关于 MATLAB 如何使用、改变颜色在下一章讨论。在任何情况下，由于颜色用于增加图形有效的第四维，这样使用的颜色被称做伪彩色。

除了上例中的输入参量，函数 **mesh** 和大多数三维绘图函数都可按多种输入参量调用。这里所用的句法是最详细的，它给出了所有三个坐标轴的信息。最通常的变更方法是使用向量，将它传递给 **meshgrid**，生成 x 与 y 坐标轴，比如 **mesh(x,y,z)**。有关其它调用语法形式的信息，参阅 *MATLAB 参考指南*或在线帮助。

如上图所示，网格线条之间的区域是不透明的。MATLAB 命令 **hidden** 控制网格图的这个特性。比如，用 MATLAB 的函数 **sphere** 产生两个球面如下：

```
» [X,Y,Z]=sphere(12);
» subplot(1,2,1)
» mesh(X,Y,Z),title( ‘ Opaque ‘ )
» hidden on
» axis off
» subplot(1,2,2),title( ‘ Transparent ‘ )
» mesh(X,Y,Z)
» hidden off
» axis off
```

输出见图 18.7.



不透明　　透明

图 18.7

左边的球是不透明的（线被隐蔽），而右边的球是透明的（线未被隐蔽）。

MATLAB 的 **mesh** 有两个同种函数：**meshc**，它画网格图和基本的等值线图；**meshz**，它画包含零平面的网格图。

```
» [X,Y,Z]=peaks(30);
» meshc(X,Y,Z)    %    mesh plot with underlying contour plot
» title( ‘ MESHC of PEAKS ‘ )
» meshz(X,Y,Z)    %    mesh plot with zero plane
» tiTle( ‘ MESH of PEAKS ‘ )
» title( ‘ MESH of PEAKS ‘ )
» hidden off    %    make mesh transparent so minimums can be seen
```

输出分别见图 18.8 和图 18.9.

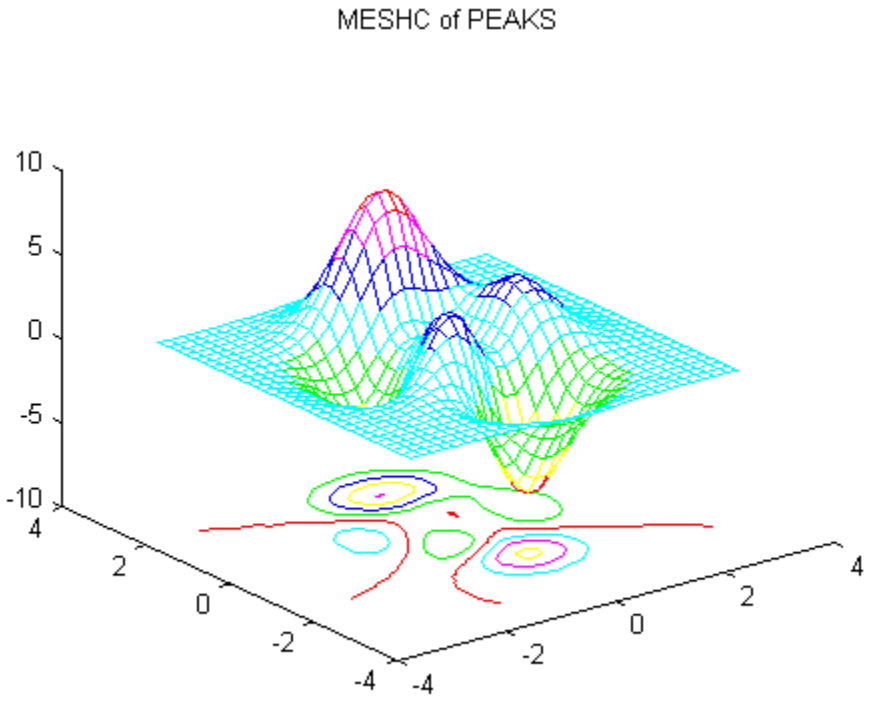


图 18.8　函数 PEAKS 的网格图和基本等值线图

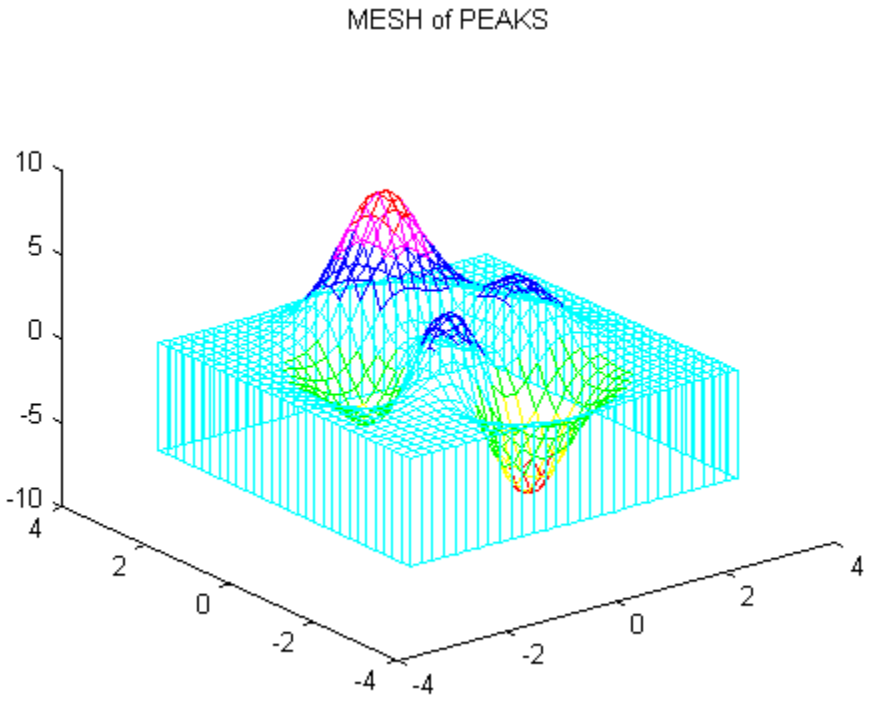


图 18.9　函数 PEAKS 的带零平面的网格图

关于以上函数更详细的信息参阅 *MATLAB* 参考指南或使用在线帮助。

18.6 曲面图

曲面图，除了各线条之间的空档（称作**补片**）用颜色填充以外，和网格图看起来是一样的。这种图一般使用函数 **surf** 来绘制。自然，函数 **surf** 使用和函数 **mesh** 相同的调用语法。比如：

```
» [X,Y,Z]=peaks(30);
» surf(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURF of PEAKS ‘ )
```

输出见图 18.10.

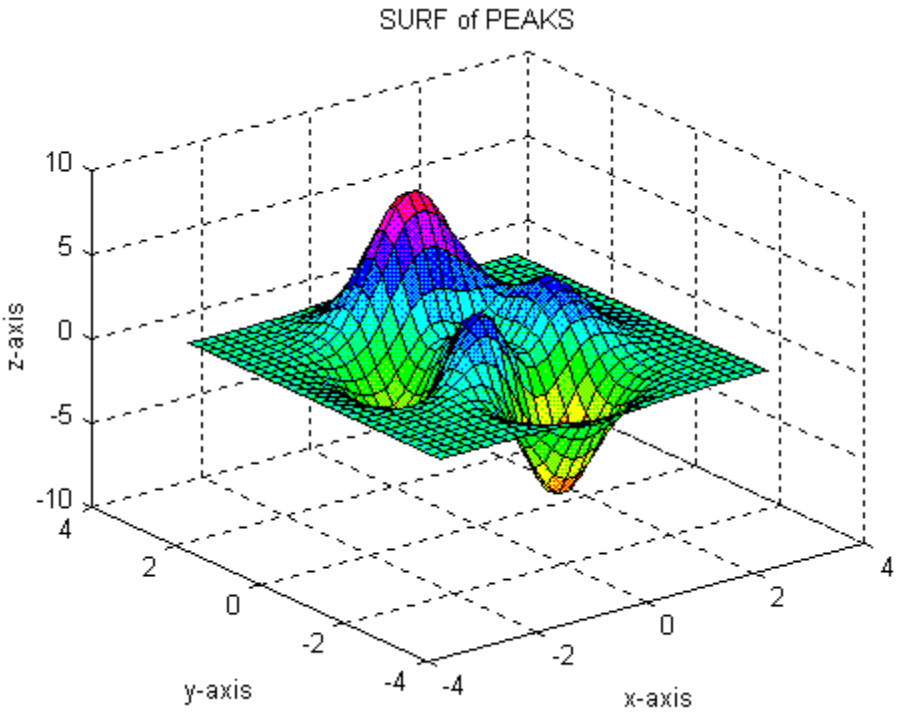


图 18.10　函数 PEAKS 的曲面图

曲面图的一些特性正好和网格图相反：它的线条是黑色的，线条之间的补片有颜色；而在网格图里，补片是黑色的而线条有颜色。对函数 **mesh**，颜色沿着 z 轴按每一补片变化，而线条颜色不变。

在曲面图里，人们不必考虑象网格图一样隐蔽线条，但要考虑用不同的方法对表面加色彩。在前面的曲面图的例子中，就是分割成**块**，每块就象一块染色玻璃窗口或物体，黑线便是各单色染色玻璃块之间的连接。除此以外，MATLAB 还提供了**平滑**加颜色和**插值**加颜色功能。这可以通过调用函数 **shading** 来实现。

```
» [X,Y,Z]=peaks(30);
```

» surf(X,Y,Z)　%　same plot as above
» grid,xlabel(‘ x-axis ‘),ylabel(‘ y-axis ‘),zlabel(‘ z-axis ‘)
» title(‘ SURF of PEAKS ‘)
» shading flat

输出见图 18.11.

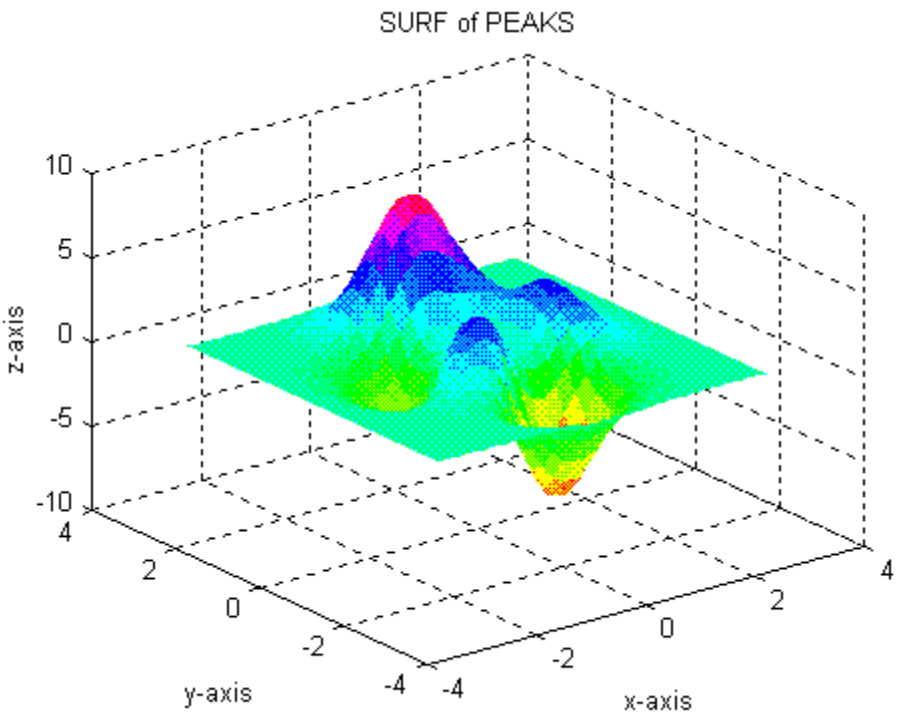


图 18.11　函数 PEAKS 的平滑加彩色曲面图

如上所示平滑加色彩的例子中，每一补片仍保存着单一的颜色，但各块连接处的黑线已去掉。

» shading interp

输出见图 18.12.

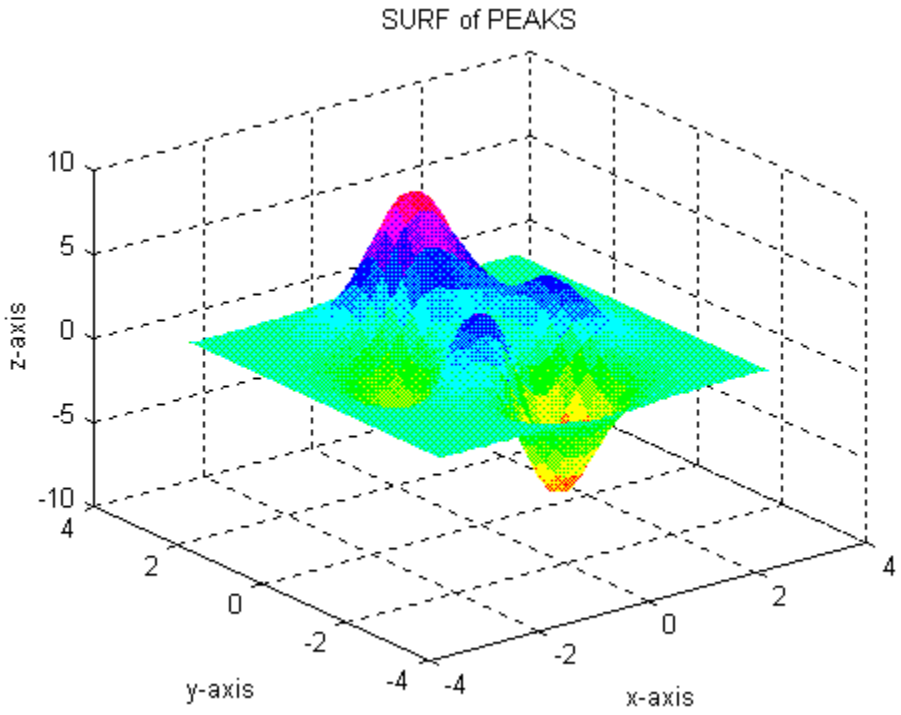


图 18.12　函数 PEAKS 的插值加彩色曲面图

如上所示内插加色彩的例子中，同样去掉了线条，但各补片以插值加颜色，即各补片的颜色根据赋予顶点的色值，对其区间进行了插值计算。很明显，插值色彩需要比分块和平滑更多的计算量。**在一些计算机系统中，插值色彩会产生非常长的打印延时或打印错误。这问题不在于 PostScript 文件太大，而是由于在打印机上产生沿图形曲面连续变化的阴影所需的巨大计算量。通常对这个问题最简单的解决方法是使用平滑加色彩法来打印**。

色彩对 **surf** 作图的视觉效果有着巨大的影响。对网格图也是如此，尽管由于只有线条有颜色，对视觉效果的影响相对要小一些。

因为曲面图不能作成透明，但在一些情况下可以很方便地移走一部分表面以便看到表面以下部分，在 MATLAB 中，这是通过在所期望的洞孔的所在位置，将数据置为特定的 **NaN** 来实现。**由于 NaN 没有任何值，所有的 MATLAB 作图函数都忽略 NaN 的数据点，在该点出现的地方留下一个洞孔**。例子如下：

```
» [X,Y,Z]=peaks(30);
» x=X(1,:);    %    vector of x axis
» y=Y(:,1);    %    vector of y axis
» i=find(y>.8 & y<1.2);    %    find x-axis indices of hole
» j=find(x>-.6 & x<.5);    %    find x-axis indices of hole
» Z(i,j)=nan*Z(i,j);    %    set values at hole indices to NaNs
» surf(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURF of PEAKS with a Hole ‘ )
```

输出见图 18.13.

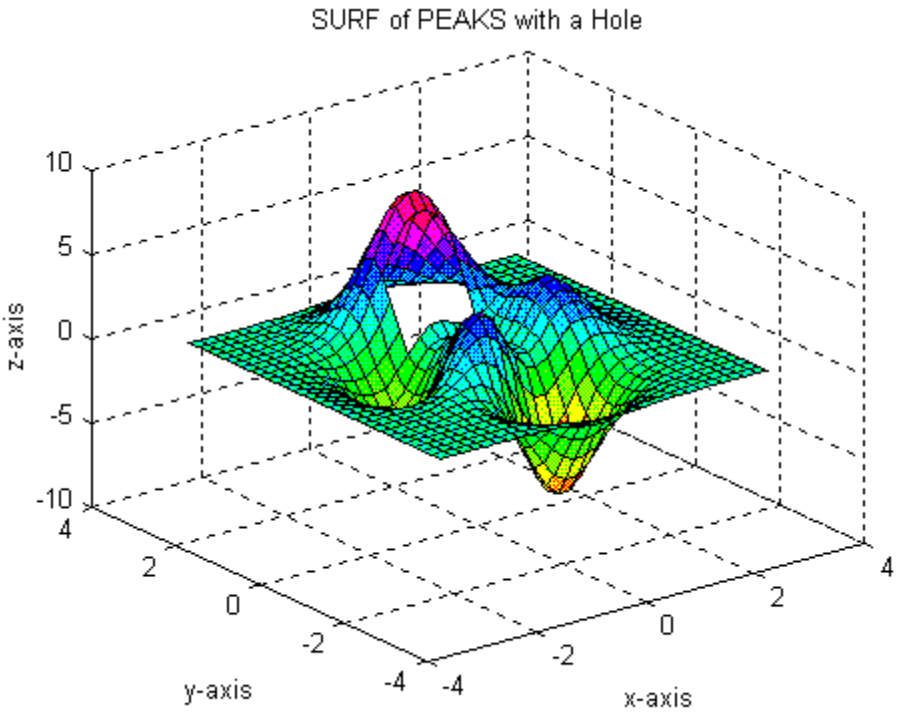


图 18.13　函数 PEAKS 的带洞孔曲面图

MATLAB 的 **surf** 也有两个同种函数：**surfc**，它画出具有基本等值线的曲面图；**surfl**，它画出一个有亮度的曲面图。例如：

```
» [X,Y,Z]=peaks(30);
» surfc(X,Y,Z)    %    surf plot with    contour plot
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURFC of PEAKS ‘ )
```

输出见图 18.14.

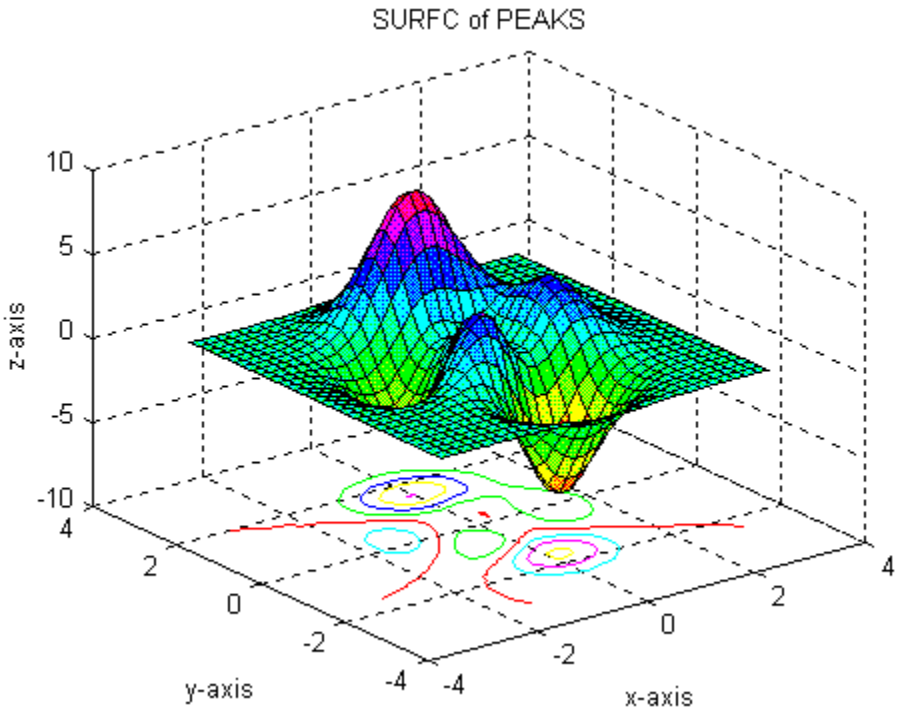


图 18.14　函数 PEAKS 的曲面图和基本等值线图

```
» [X,Y,Z]=peaks(30);
» surfl (X,Y,Z)    %    surf plot with lighting
» shading interp    %    surfl plots look best with interp shading
» colormap pink    %    they also look better with shades of a single color
» grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
»title( ‘ SURFL OF PEAKS ‘ )
```

输出见图 18.14.

图 18.16　函数 PEAKS 的带光线照明曲面图

关于加到曲面的亮度，函数 **surfl** 作了许多假设。有关设置亮度属性的详细信息参阅 *MATLAB* 参考指南的函数 **surfl** 或使用在线帮助。同样，在上面执行的命令中，**colormap** 是 MATLAB 函数，它对图形施加一套不同的颜色。这个函数在下一章讨论。

18.7 等值线图

MATLAB 提供了另一种基本的三维图形，即三维等值线图。这种图形通过函数 **contour3** 来绘制。

```
» [x,y,z]=peaks(30);
» contour3(X,Y,Z,16)    %    draw sixteen contour lines
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
```

输出见图 18.16.

图 18.16　函数 PEAKS 的三维等值线图

可以看到，图形中每一条线的颜色遵循了与二维函数 **plot** 一样的次序。这种颜色次序可以表现出明显的对比，但经常模糊了所代表的数据的一些重要特性。如果能使每一条线遵循在**网格图**和**曲面图**里所用的加色方法，那么效果会好得多。也许在 MATLAB 的下一个版本中，这种颜色设置会成为缺省设置，但使用在下一章要讨论的 MATLAB 图形处理能力，也能解决这个问题。

```
» [X,Y,Z]=peaks(30);
» N=16;    %    number of contour lines and their colors
» clf    %    clear the current figure
» view(3)    %    set view to 3-D
» hold on    %    hold blank screen
» set(gca, ‘ ColorOrder ‘ ,hsv(N))    %    use colors from default hsv colormap
» contour3(X,Y,Z,N)    %    draw N contour lines
» grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
» hold off
```

输出见图 18.17.

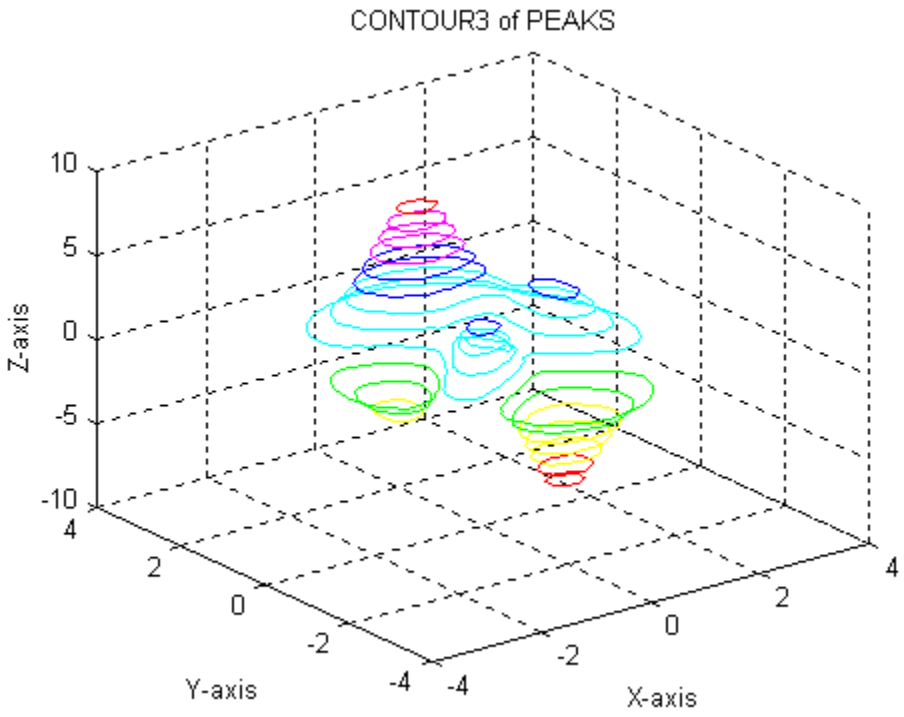


图 18.17　函数 PEAKS 的三维等值线图

现在，各条等值线的颜色沿着 z 轴的变化和**网格图**和**曲面图**一样。为方便起见，这种策略已体现在精通 *MATLAB* 工具箱的函数 **mmcont3** 中。**mmcont3** 具有和函数 **contour3** 相同的调用语法的变化，并允许选择可用的颜色映象。例如，» **mmcont3（X，Y，Z，N，‘hsv’）** 复制上面的图形。**mmcont3** 的在线帮助如下：

» help mmcont3

MMCONT3 3-Dcontour plot using a colormap

MMCONT3(X,Y,Z,N,C) plots N contours of Z in 3-D using the color specified in C.C can be a linestyle and color as used in plot,e.g., ‘ r- ‘ ,or C can be the string name of a colormap.X and Y.

define the axis limits.

If not given default argument values are :N=10 ,C= ‘ hot ‘ ,X and Y=row and column indices of

Z.Examples:

MMCONT3(Z)	10 lines with hot colormap
MMCONT3(Z,20)	20 lines with hot colormap
MMCONT3(Z, ‘ copper ‘)	10 lines with copper colormap
MMCONT3(Z,20, ‘ gray ‘)	20 lines with gray colormap
MMCONT3(X,Y,Z, ‘ jet ‘)	10 lines with jet colormap
MMCONT3(Z, ‘ c-- ‘)	10 dashed lines in cyan

MMCONT3(X,Y,Z,25, ‘ pink ‘) 25 lines in pink colormap

CS=MMCONT3(. . .) returns the contour matrix CS as described in CONTOURC.
[CS,H]=MMCONT3(. . .) returns a column vector H of handles to line objects

---

帮助信息：

MMCONT3(X,Y,Z,N,C)用由 C 指定的颜色在三维空间内画 N 条 Z 方向的等值线图。C 可以是在 plot 中使用的线形 和颜色，例如 ‘ r- ‘ ；或者 C 可以是一个颜色映象的字符串名。X 和 Y 指定了坐标轴的范围。如果 未指定参数，缺省的参数值是：N=10，C= ‘ hot ‘ ，X 和 Y 分别是 Z 的行和列的下标。举例：

MMCONT3(Z)	用暖色映象画 10 条等值线
MMCONT3(Z,20)	用暖色映象画 20 条等值线
MMCONT3(Z, ‘ copper ‘)	用铜黄色映象画 10 条等值线
MMCONT3(Z，20， ‘ gray ‘)	用灰色映象画 20 条等值线
MMCONT3(X,Y,Z, ‘ jet ‘)	用**‘jet’暖色映象画 10 条等值线
MMCONT3(Z, ‘ c-- ‘)	画 10 条青蓝色的虚划线等值线
MMCONT3(X,Y,Z,25, ‘ pink ‘)	用粉红色映象画 25 条等值线
CS=MMCONT3(…)	如在 CONTOURC 中描述，返回等值线矩阵 CS。
[CS,H]=MMCONT3(…)	把句柄的列向量 H 返回到线条对象。

等值线也可由一种颜色给出：

```
» [x,y,z]=peaks(30);
» contour3(X,Y,Z,16, ‘ y ‘ )  %  draw sixteen contour lines in yellow
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
```

输出见图 18.18

图 18.18 函数 PEAKS 的黄色三维等值线图

关于颜色使用的详细信息参阅下一章；以上函数使用的详细信息参阅 *MATLAB* 参考指南或使用在线帮助。

18.8 三维数据的二维图

有些情况下，希望得到三维数据的二维表示。在 MATLAB 里这一点是通过用函数 **view** 设置视角使其中一维不出现来实现的。另外，MATLAB 还提供了两个函数，将 **contour3** 和 **surf** 向下正视到 x－y 平面。例如，函数 **contour3** 的二维图就等价于 **contour**。

```
» [X,Y,Z]=peaks(30);
» contour(X,Y,Z,16)    %    draw sixteen contour lines
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ CONTOUR of PEAKS ‘ )
```

输出见图 18.19

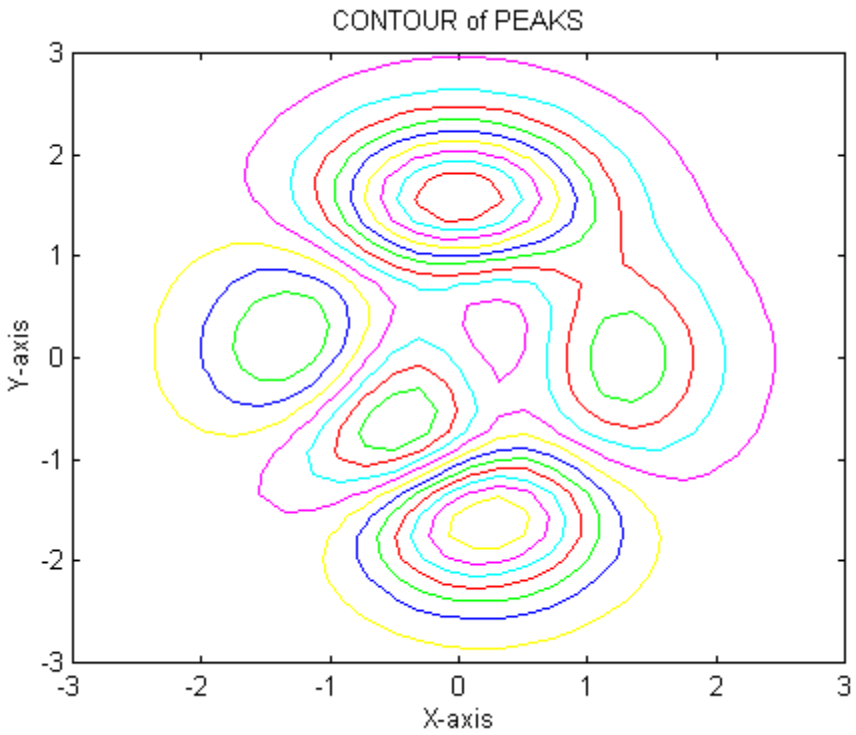


图 18.19　函数 PEAKS 的等值线图

要注意，它如何等效于使用 **contour3** 以及如何改变视点俯视到 x－y 平面。如同 **contour3**，图中的等值线利用 **plot** 命令的六种基本颜色。如前所述，因为颜色不提供视觉效果，这类图形不好使用并且引起混淆。这种缺省行为也许在 MATLAB 的下一版本中会改变，但也可用句柄图形来改变。

```
» [X,Y,Z]=peaks(30);
» N=16;    %    number of contour lines and their colors
» clf    %    clear the current figure
» hold on    %    hold blank screen
» set(gca, ‘ ColorOrder ‘ ,hsv(N))    %    use colors from default hsv colormap
» contour(X,Y,Z,N)    %    draw N contour lines
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
» title( ‘ CONTOUR of PEAKS ‘ )
» hold off
```

输出见图 18.20

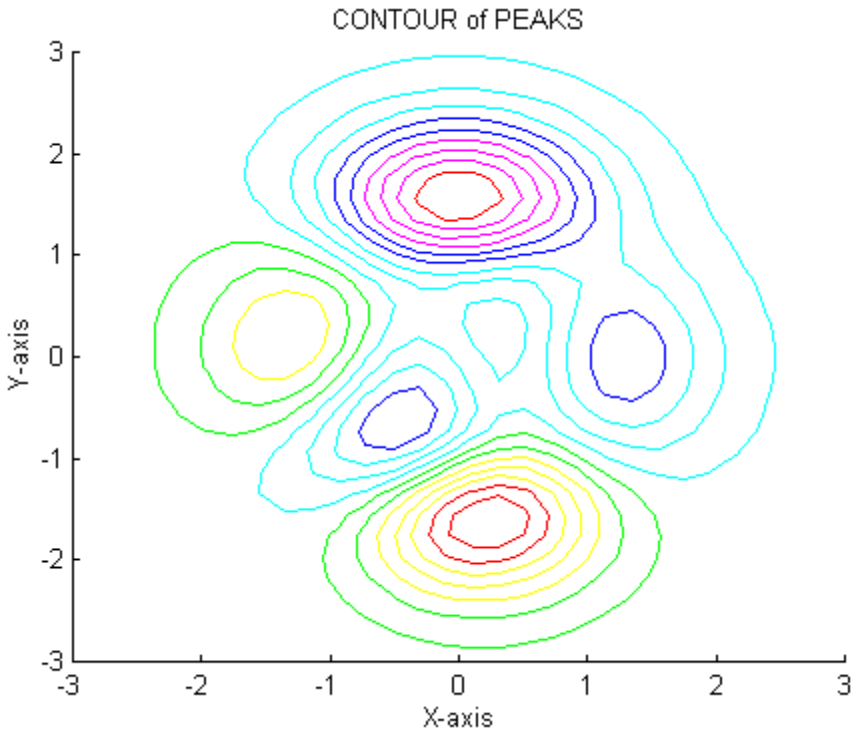


图 18.20　　函数 PEAKS 的等值线图

现在，等值线遵循 **hsv** 颜色映象里的级差，颜色提供了一种有用的效果！为方便起见，上述策略已体现在精通 *MATLAB* 工具箱的函数 **mmcont2** 中。**mmcont2** 接受和函数 **contour** 相同的调用语法变更，并允许选择一个可用的颜色映象。例如，» **mmcont2（X，Y，Z，N，‘hsv’）**复制上面的图形。**mmcont2** 的在线帮助如下：

```
» help mmcont2
MMCONT2 2-D contour plot using a colormap.
    MMCONT2(X,Y,Z,C) plots N contours of Z in 2-D using the color specified in C.C can be
    a linestyle and color as used in plot,e.g., ‘ r- ‘ ,or C can be the string name of a
colormap.X and          Y.
    define the axis limits.
```

If not given default argument values are :N=10,C= ‘ hot ‘ ,X and Y =row and column indices of Z.Examples:

MMCONT2(Z) 10 lines with hot colormap
MMCONT2(Z,20) 20 lines with hot colormap
MMCONT2(Z, ‘ copper ‘) 10 lines with copper colormap
MMCONT2(Z,20, ‘ gray ‘) 20 lines with gray colormap
MMCONT2(X,Y,Z, ‘ jet ‘) 10 lines with jet colormap
MMCONT2(Z, ‘ c-- ‘) 10 dashed lines in cyan
MMCONT2(X,Y,Z,25, ‘ pink ‘) 25 lines in pink colormap

CS=MMCONT2(. . .) returns the contour matrix CS as described in CONTOURC.
[CS,H]=MMCONT2(. . .) returns a column vector H of handles to line objects

---

帮助信息：

MMCONT2(X,Y,Z,N,C)用由 C 指定的颜色在二维平面内画出关于 Z 方向的 N 条等值线。C 可以是在函数 **plot** 中使用的线形和颜色，例如‘r-‘；或者是一个字符串指明颜色映象名。 X 和 Y 指定了坐标轴的范围。

如果未指定参数，缺省的参数值是：N=10，C= ‘ hot ‘ ，X 和 Y 分别是 Z 的行列下标。 例子如下：

MMCONT2(Z) 使用暖色颜色映象画 10 条等值线
MMCONT2(Z,20) 使用暖色颜色映象画 20 条等值线
MMCONT2(Z, ‘ copper ‘) 使用铜黄色颜色映象画 10 条等值线
MMCONT2(Z，20， ‘ gray ‘) 使用灰色颜色映象画 20 条等值线
MMCONT2(X,Y,Z, ‘ jet ‘) 使用‘jet’暖色颜色映象画 10 条等值线
MMCONT2(Z, ‘ c-- ‘) 使用青蓝色颜色映象用虚划线画 10 条等值线
MMCONT2(X,Y,Z,25, ‘ pink ‘) 使用粉红色颜色映象画 25 条等值线
CS=MMCONT2(…)返回等值线矩阵 CS，它在 CONTOURC 中描述。
[CS,H]=MMCONT2(…)返回一个包含有线条对象的句柄的列向量。

上面讨论的三维或二维等值线图都假定数据定义在矩形网格或论域内。当不是这种情形时，数据必须被转换到矩形网格上，使等值线图看起来逼真。如本章前面 18.4 节所述，MATLAB 函数 **griddata** 能实现所需的转换，更详细的信息参阅该节。

函数 **surf** 的二维等效函数是 **pcolor**，它代表伪彩色。

```
» [X,Y,Z]=peaks(30);
» pcolor(X,Y,Z);   %   surf plot view from above
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ PCOLOR   of   PEAKS ‘ )
```

输出见图 18.21.

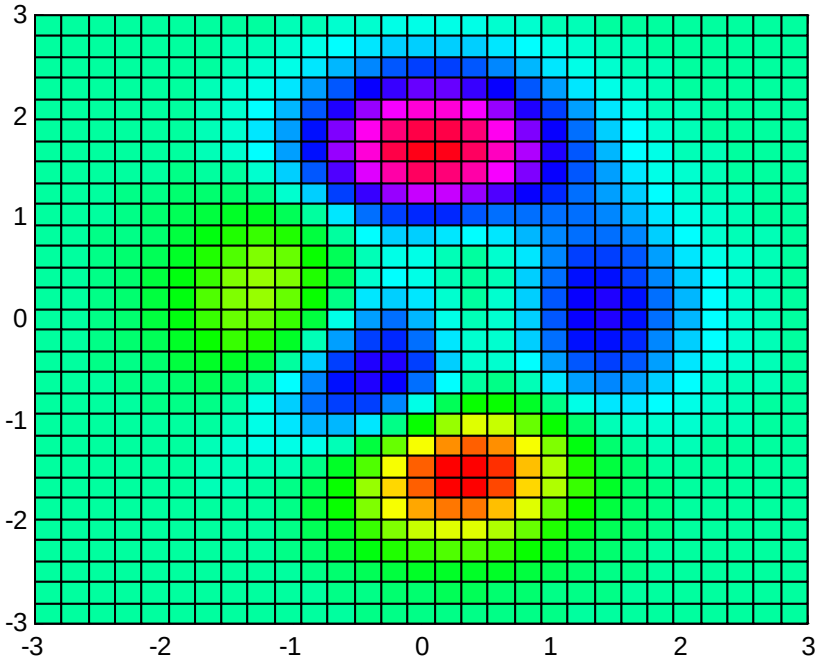


图 18.21　函数 PEAKS 的伪彩色图

由于这是一个 **surf** 图，可以使用函数 **shading**。另外，有时在 **pcolor** 图的上面放一个单色**等值线图**是很有用的。

```
» [X,Y,Z]=peaks(30);
» pcolor(X,Y,Z);
» shading interp
» hold on
» contour(X,Y,Z,19, ‘ k ‘ )    %    add 19 contour lines in black
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ PCOLOR    and    CONTOUR    of    PEAKS ‘ )
» hold off
```

输出见图 18.22.

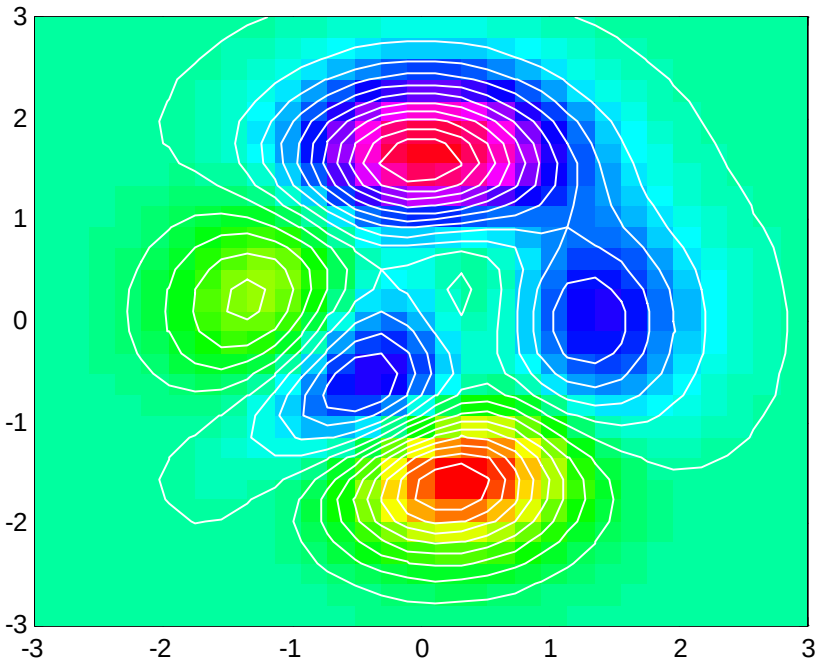


图 18.21　函数 PEAKS 的伪彩色图和等值线图

18. 9 其它函数

除了上面讨论的三维函数，MATLAB 还提供了函数 **waterfall**，**quiver**，**fill3**，和 **clabel**。函数 **waterfall** 与函数 **mesh** 一样，只是它的网格线是在 x 轴方向出现。例如：

```
» [X,Y,Z]=peaks(30);
» waterfall(X,Y,Z)
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
```

输出见图 18.23.

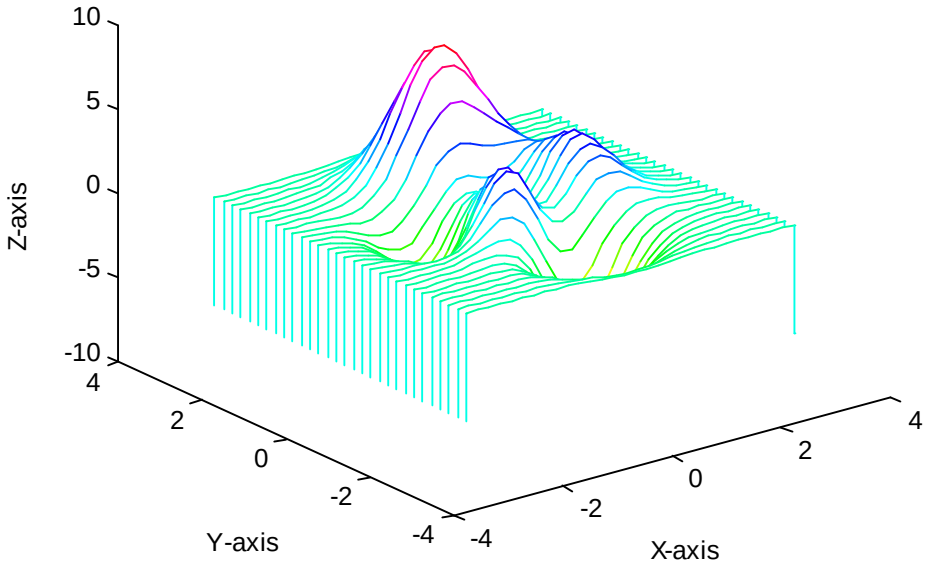


图 18.23　函数 PEAKS 的 waterfall 图

函数 **quiver** 在等值线图上画出方向或速度箭头。例如：

```
» [X,Y,Z]=peaks(16);
» [DX,DY]=gradient(Z, .5, .5);
» contour(X,Y,Z,10)
» hold on
» quiver(X,Y,DX,DY)
» hld off
```

输出见图 18.24.

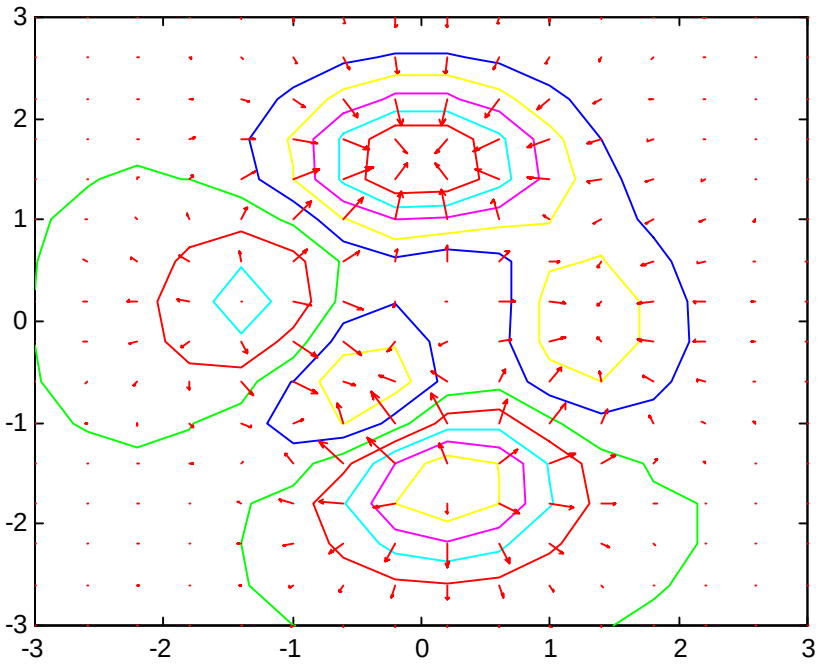


图 18.24　函数 PEAKS 的 quiver 图

函数 **fill3** 等效于三维函数 **fill**，可在三维空间内画出填充过的多边形。函数 **fill3(x,y,z,c)** 使用数组 **x**，**y** 和 **z** 作为多边形的顶点而 **c** 指定了填充的颜色。例如，下例用随机的顶点坐标值画出五个黄色三角形。

» fill3(rand(3,5),rand(3,5),rand(3,5), ' y ')

输出见图 18.25.

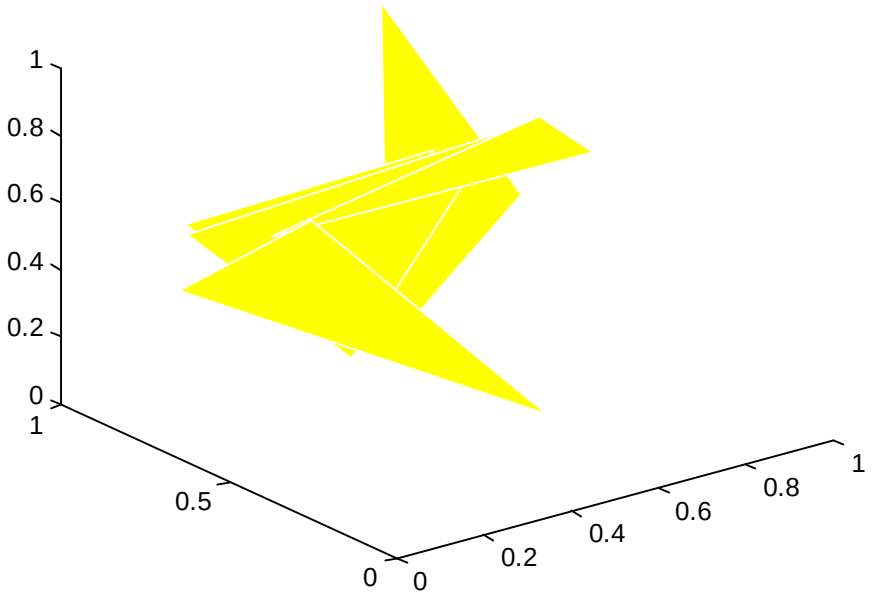


图 18.25　函数 fill3 图示

函数 **clabel** 给等值线图标上高度值。这样做时，函数 **clabel** 需要函数 **contour** 等值线矩阵的输出。

```
» [X,Y,Z]=peaks(30);
» cs=contour(X,Y,Z,8);    %    request output from contour
» clabel(cs)    %    add labels identifying heights
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ CONTOUR    of    PEAKS    with    LABELS ‘ )
```

输出见图 18.26.

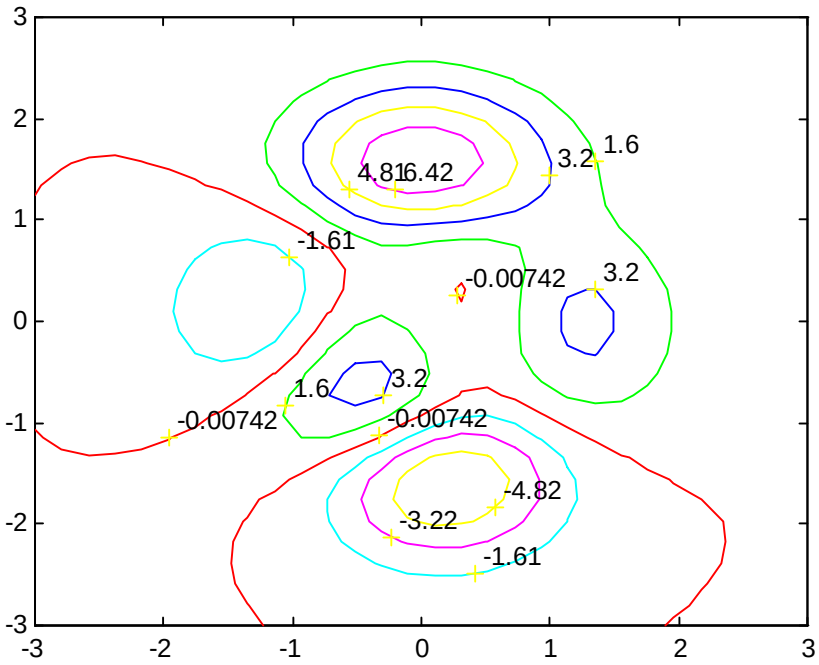


图 18.26　函数 PEAKS 的带标志等值线图

18. 10 动画

MATLAB 提供了一种能力，它可以存储一系列各种类型的二维或三维图，然后象放电影一样把它们按次序重放出来。在某种意义上，动画提供的运动为图形增加另一个维数。通常图形的次序不必以任意的方式关联起来。一种明显的动画类型是取出三维图形然后缓慢地将它旋转，这样我们就可以从不同角度来观察它。另一种类型是当一个参数变化时，依次显示某些问题解的图形。

MATLAB 中的函数 **moviein**，**getframe** 和 **movie** 提供了捕捉和播放动画的所需工具。函数 **moviein** 可以产生一个帧矩阵来存放动画中的帧；函数 **getframe** 对当前的图象进行快照；而函数 **movie** 按顺序回放各帧。照这样，捕捉和回放动画的方法是：（1）创建帧矩阵；（2）对动画中的每一帧生成图形，并把它捕捉到到帧矩阵里；（3）从帧矩阵里回放动画。

考虑下面的一段脚本 M 文件例子，例中绘制了函数 **peaks** 并且将它绕 z 轴旋转。

```
%    movie making example: rotate a 3-D surface plot

[X,Y,Z]=peaks(30);    %    create data
surfl(X,Y,Z)    %    plot surface with lighting
axis([-3    3    -3    3    -10    10])    %    fix axes so that scaling does not change
axis off    %    erase axes because they jump around
shading interp    %    make it pretty with interpolated shading
colormap(hot)    %    choose a good colormap for lighting
```

```
m=moviein(15);    %    choose 15 movie frames for frame matrix m

for I=1:15    %    rotate and capture each frame
      view(-37.5+24*(i-1),30)    %    change viewpoint for this frame
      m(:,i)=getframe;    %    add figure to frame matrix
end

movie(m)    %    play the movie!
```

注意到动画中的每一帧在帧矩阵中占据一个不同的列。帧矩阵的大小随着动画中的帧数和图形窗口的大小而增加，而与所绘图形的复杂性无关，这是因为函数 **getframe** 仅仅是捕捉位图。按缺省，函数 **movie** 只放一遍动画。通过加入其它输入参量，它可以向前放、向后倒放、放指定次数或按特定的桢速率播放。关于这些特征的详细信息，参阅 *MATLAB* 参考指南或使用在线帮助。

由于上述的动画制作策略很有用，它已体现在精通 *MATLAB* 工具箱的函数 **mmspin3d** 中。

```
function M=mmspin3d(n)
%    MMSPIN3D Make Movie by 3D Azimuth Rotation of Current Figure.
%    MMSPIN3D(N) captures and plays N frames of the current figure
%    through one rotation about the Z-axis at the current elevation.
%    M=MMSPIN3D(N) returns the movie in M for later playing with movie.
%    If not given,N=18 is used.
%    MMSPIN3D fixes the axis limits and issues axis off.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

if nargin<1,n=18;end
n=max(abs(round(n)),2);

axis(axis);
axis off
incaz=round(360/n);
[az,el]=view;

m=movie(n);
for i=1:n
      view(az+incaz*(i-1),el)
      m(:,i)=getframe;
end
if nargout,    M=m;
else,          movie(m);
end
```

使用 **mmspin3d**，上述脚本简化如下：

```
%    movie-making example:rotate a 3-D surface plot

[X,Y,Z]=peaks(30);   %    create data
surfl(X,Y,Z)              %    plot surface with lighting
shading interp            %    make it pretty with interpolated shading
colormap(hot)             %    choose a good colormap for lighting
mmspin3d(15)
```

18.11 小结

本章所讨论的函数和它们的特征总结在表 18.2、表 18.3、表 18.4 和表 18.5 中：

表 18.2

三维绘图函数	
contour	二维等值线图，即从上向下看 contour3 等值线图
contour3	等值线图
fill3	填充的多边形
mesh	网格图
meshc	具有基本等值线图的网格图
meshz	有零平面的网格图
pcolor	二维伪彩色绘图，即从上向下看 **surf** 图
plot3	直线图
quiver	二维带方向箭头的速度图
surf	曲面图
surfc	具有基本等值线图的曲面图
surfl	带亮度的曲面图
waterfall	无交叉线的网格图

表示 18-3

三维绘图工具	
axis	修正坐标轴属性
clf	清除图形窗口
clabel	放置等值线标签
close	关闭图形窗口
figure	创建或选择图形窗口
getframe	捕捉动画桢
grid	放置网格
griddata	对画图用的数据进行内插
hidden	隐蔽**网格图**线条
hold	保留当前图形
meshgrid	产生三维绘图数据
movie	放动画

moviein	创建桢矩阵，存储动画
shading	在**曲面图**和**伪彩色图**中用**分块**、**平滑**和**插值**加阴影
subplot	在图形窗口内画子图
text	在指定的位置放文本
title	放置标题
view	改变图形的视角
xlabel	放置 x 轴标记
ylabel	放置 y 轴标记
zlabel	放置 z 轴标记

表 18.4

函数 view	
view(az,el)	设置视图的方位角 **az** 和仰角 **el**
view([az,el])	
view([x,y,z])	在笛卡儿坐标系中沿向量[x,y,z]正视原点设置视图，例如 **view([0 0 1])=view(0,90)**
view(2)	设置缺省的二维视图，**az=0，el=90**
view(3)	设置缺省的三维视图，**az=-37.5，el=30**
[az,el]=view	返回当前的方位角 **az** 和仰角 **el**
view(T)	用一个 4×4 的转置矩阵 **T** 来设置视图
T=view	返回当前的 4×4 转置矩阵

表 18.5

掌握 MATLAB 高级图形功能	
mmcont2(X,Y,Z,C)	具有颜色映象的二维等值线图
mmcont3(X,Y,Z,C)	具有颜色映象的三维等值线图
mmspin3d(N)	旋转当前图形的三维方位角来制作动画
mmview3d	用滑标来调整视角

关键词索引

chap 18

pcolor	伪彩色
3-D helix	三维螺旋线
3-D grid	三维网格
subplot	子图
Figure	图形窗口
viewpoint	视角
elevation	仰角

azimuth 方位角
Handle Graphics 句柄图形
slider 滑标
plaid 方格
mesh plots 网格图
mesh surface 网状曲面
surface 曲面图
contour plot 等值线图
zero plane 零平面
patch 补片
flat shading 平滑加色彩
interpolated shading 插值加色彩
progression 级差
color map 颜色映象
directional or velocity arrows 方向或速度箭头
movie 动画
frame 帧
underlying contour plot 基本等值线图

第十九章 颜色的使用

MATLAB 提供了许多在二维和三维空间内显示可视信息的工具。例如，看一条 sin 函数的曲线图就会比一堆数据提供更多的信息。这种用图表和图形来表示数据的技术叫做**数据可视化**。MATLAB 不仅是一个强大的计算工具，并且在以引人入胜和直观方式可视地表示数据方面也很有特色。

但是很多时候，一个简单的二维或三维图形不能一次显示出想要提供的全部信息。这时，颜色可以对图形提供一个附加的维数。前面章节讨论的许多绘图函数都可以接受一个可用的颜色参量，来增加这附加的维数。

本章的讨论以研究颜色映象开始：如何使用、显示、修改和如何创建用户自己的颜色映象。然后，阐述在一个图形窗口中仿真多个颜色映象的技术或只使用颜色映象的一部分的技术。最后，讨论照明模型并提供例子。

19.1 颜色映象理解

MATLAB 有一个叫**颜色映象**的数据结构来代表颜色值。颜色映象定义为一个有三列和若干行的矩阵。利用 0 到 1 之间的数，矩阵的每一行都代表了一种色彩。任一行的数字都指定了一个 **RGB** 值，即红、黄、蓝三种颜色的强度，形成一种特定的颜色。一些有代表性的 **RGB** 值在表 19.1 中给出。

表 19.1

简单颜色

Red（红）	Green（绿）	Blue（蓝）	颜色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
.5	0	0	深红
.5	.5	.5	灰色

有十个 MATLAB 函数产生预定的颜色映象。见表 19.2

表 19.2

标准颜色映象	
hsv	色彩饱和值（以红色开始和结束）
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	**hsv** 的一种变形（以蓝色开始和结束）
copper	线性铜色度
prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
flag	交替为红色、白色、蓝色和黑色

按缺省，上面所列的各个颜色映象产生一个 64×3 的矩阵，指定了 64 种颜色 **RGB** 的描述。这些函数都接受一个参量来指定所产生矩阵的行数。比如 **hot(m)**产生一个 m×3 的矩阵，它包含的 **RGB** 颜色值的范围从黑经过红、橘红和黄，到白。

大多数计算机在一个 8 位的硬件查色表中一次可以显示 256 种颜色，当然有些计算机的显示卡可以同时显示更多的颜色。这就意味着在不同的图中，一般一次可以用三或四个 64×3 的颜色映象。如果使用了更多的颜色映象输入项，计算机必须经常在它的硬件查色表中调出输入项。比如，当在画 MATLAB 图形时背景图案发生了变化，就是发生了这种情况。所以，除非计算机有一次显示更多种颜色的显示卡，最好任何一次所用的颜色映象输入项数都小于 256。

19.2 颜色映象使用

语句 **colormap(M)**将矩阵 **M** 作为当前图形窗口所用的颜色映象。例如，**colormap**（**cool**）装入了一个有 64 个输入项的 **cool** 颜色映象。**colormap default** 装入了缺省的颜色映象（**hsv**）。

函数 **plot**、**plot3**、**contour** 和 **contour3** 不使用颜色映象，它们使用列在 plot 颜色和线形表中的颜色。而大多数其它绘图函数，比如 **mesh**、**surf**、**fill**、**pcolor** 和它们的各种变形函数，使用当前的颜色映象。

接受颜色参量的绘图函数中的颜色参量通常采用以下三种形式之一：（1）字符串。代表 **plot** 颜色或线型表中的一种颜色，比如，' **r** '代表红色；（2）三个输入的行向量。它代表一个单独的 **RGB** 值，比如**[.25 .50 .75]**；（3）矩阵。如果颜色参量是一个矩阵，其元素作了调整，并把它们用作当前颜色映象的下标。最后一种形式会在以后作更多讨论。

19.3 颜色映象显示

可以用多种途径来显示一个颜色映象。其中一个方法是观察颜色映象矩阵的元素。

```
» hot(8)
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

上面的数据显示出第一行是 1/3 红色，而最后一行是白色。另外，函数 **pcolor** 可以用来显示一个颜色映象。例如：

```
» n=16;
» colormap(jet(n))
» pcolor([1:n+1;1:n+1]' )
» title( 'Using Pcolor to Display a Color Map' )
```

输出见图 19.1.

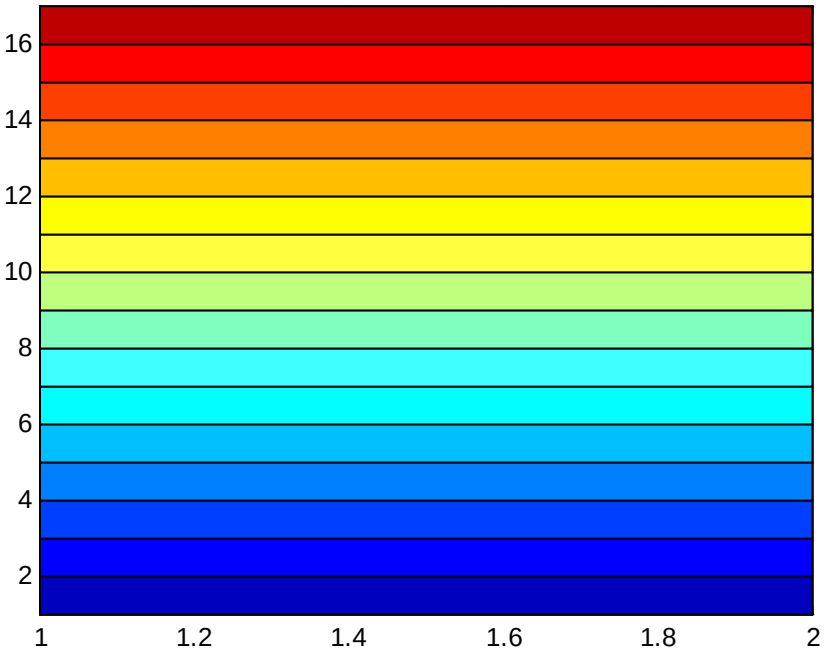


图 19.1　用伪彩色来显示颜色映象

因为上面这一段程序很有用处，它已经装入精通 MATLAB 工具箱中的函数 **mmshow** 中。

```
» help mmshow
MMSHOW PCOLOR Colormap Display
    MMSHOW uses pccolor to display the current colormap.
    MMSHOW(MAP) displays the colormap MAP.
    MMSHOW(MAP(N)) displays the colormap MAP having N elements.
    Examples:MMSHOW(hot)
             MMSHOW(pink(30))
```

帮助信息：

MMSHOW　显示 PCOLOR 颜色映象
MMSHOW　使用 pcolor 来显示当前颜色映象
MMSHOW(MAP)　显示 MAP 颜色映象
MMSHOW（MAP(N)）　显示一个有 N 个元素的 MAP 颜色映象
例子：MMSHOW（hot）
　　MMSHOW（pink（30））

函数 **mmshow** 取和 **colormap** 同样的输入参量，但在这种情况下它用自己的伪彩色显示而不是把颜色映象施加到当前图形。

另一种途径是使用 MATLAB 的函数 **rgbplot**，它可以把颜色映象的各列分别画成红、绿

和蓝色。例如：

» rgbplot(hot)

输出见图 19.2.

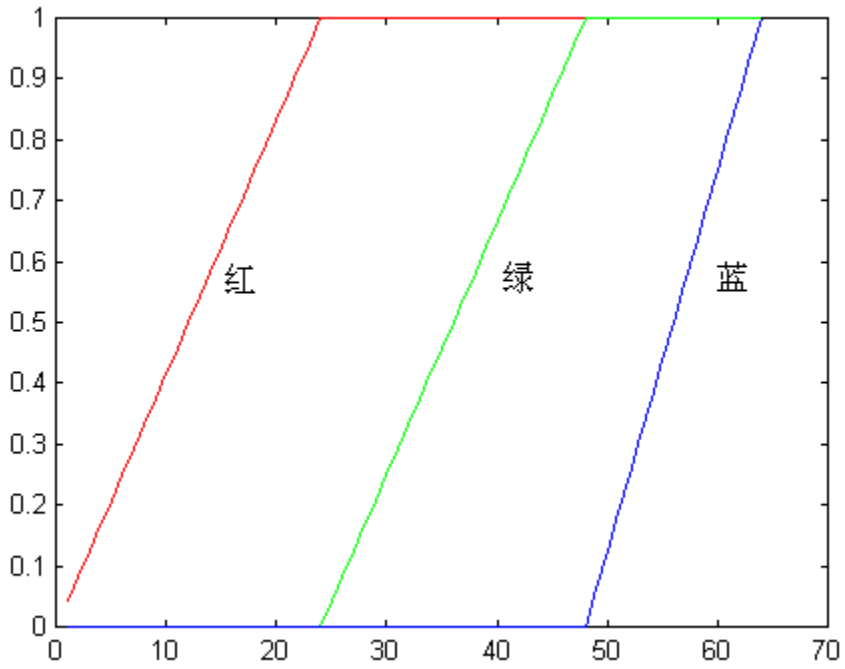


图 19.2　用红、绿和蓝色画颜色映象

图中显示红色分量首先增加，然后是绿色，最后是蓝色。**rgbplot（gray）**表示所有三列数据均匀线性地增加（三条线重叠）。

最后，函数 **colorbar** 在当前的图形窗口中增加水平或垂直的颜色标尺以显示当前坐标轴的颜色映象。» **colorbar('horiz')** 在当前的图形下面放一个水平的颜色条。» **colorbar('vert')** 在当前的图形右边放一个垂直的颜色条。对无参量的 **colorbar**，如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条。下面的例子就演示了 **colorbar** 的用法。

```
» [x,y,z]=peaks;
» mesh(x,y,z);
» colormap(hsv)
» axis([-3   3   -3   3   -6   8])
» colorbar
```

输出见图 19.3.

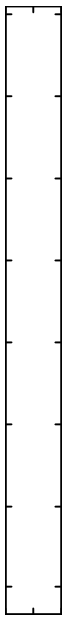


图 19.3　使用颜色条

19.4 颜色映象的建立和修改

颜色映象就是矩阵，意味着你可以象其它数组那样对它们进行操作。函数 **brighten** 就利用这一点通过调整一个给定的颜色映象来增加或减少暗色的强度。**brighten(n)**（**0<1<=1)**使当前颜色映象变亮；而 **brighten(n)**（**-1<=n<0**）使它变暗。**brighten(n)**后加一个 **brighten(-n)**使颜色映象恢复原来状态。**newmap=brighten(n)** 命令创建一个比当前颜色映象更暗或者更亮的新的颜色映象，而并不改变当前的颜色映象。命令**» newmap=brighten(cmap,n)**对指定的颜色映象创建一个已调整过的式样，而不影响当前的颜色映象或指定的颜色映象 **cmap**。

可以通过生成 m×3 的矩阵 **mamap** 来建立用户自己的颜色映象，并用 **colormap(mymap)** 来安装它。颜色映象矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小比 1 大的任意值，函数 **colormap** 会提示一个错误然后退出。

也可以在算术上来组合颜色映象，虽然结果有时是不可预料的。比如，一个叫 **pink** 的颜色映象仅仅是：

```
» pinkmap=squr(2/3*gray+1/3*hot);
```

只当所有元素都在 0 与 1 之间时，才能保证结果是一个有效的颜色映象。精通 MATLAB 工具箱中包含了一个名叫 **rainbow** 的颜色映象，它把可视范围扩展到整个颜色映象。函数 **rainbow** 的在线帮助为：

```
» help    rainbow
RAINBOW Colormap varient to HSV.
```

RAINBOW(M) Rainbow Colormap with M entries.
Red-Orange-Yellow-Green-Blue-Violet
RAINBOW by itself is the same length as the current colormap.
Apply using :colormap(rainbow)

帮助信息：

RAINBOW　HSV 颜色映象的变形
RAINBOW(M)　有 M 个入口项的 RAINBOW 颜色映象
　　红—橘黄—黄—绿—蓝—天蓝
RAINBOW 本身和当前颜色映象的长度相同
应用：colormap(rainbow)

精通 MATLAB 工具箱中还包含了一个名叫 **mmap** 的函数，它可以根据你所提供的颜色建立一个单色（比如**粉红**、**灰色**或**铜黄色**）的颜色映象。函数 **mmap** 的在线帮助是：

» help　mmap
MMAP Single Color Colormap.
MMAP(C,M) makes a colormap of length M starting with the basic colorspec C.The map changes from dark to light.
MMAP(C) is the same length as current colormap.
Examples:mmap('y') is a yellow colormap.
　　mmap([.49　1　.83]) is an aquamarine colormap.
　　mmap('c' ,20) is a cyan colormap having length 20.

帮助信息：

MMAP　单色颜色映象
MMAP(C,M)　制作一个以颜色 C 为基色的长度为 M 的颜色映象。该表的颜色从暗到明变化。
MMAP(C)　颜色映象的长度和当前颜色映象相同
例子：mmap('y')是一个黄色颜色映象
　　mmap([.49　1　.83])是一个水色的颜色映象
　　mmap('c' ,20)是一个长度为 20 的青蓝色的颜色映象
应用：colormap(mmap(c,m))

一个颜色映象定义了用于绘制图形的调色板。一个缺省的颜色映象允许对数据使用 64 种不同的 **RGB** 值。MATLAB 使用函数 **cxis** 来决定哪一个数据值映射到颜色映象中输入项。

通常，颜色映象进行过调节，把数据从最小扩展到最大，也就是说整个颜色映象都用于绘图。有时也许想改变颜色使用的方法。函数 **caxis** 代表颜色轴，因为颜色增加了另一个维数，它允许对数据范围的一个子集使用整个颜色映象或者对数据的整个集合只使用当前颜色映象的一部分。

[cmin,cmax]=caxis 返回映射到颜色映象中第一和最后输入项的最小和最大的数据。它们通常被设成数据的最小值和最大值。比如，函数 **mesh(peaks)** 会画出函数 **peaks** 的网格图，并把颜色轴 **caxis** 设为**[-6.5466，8.0752]**，即 **Z** 的最小值和最大值。这些值之间的数据点，使用从颜色映象中经插值得到的颜色。

caxis([cmin，cmax]）对 **cmin** 和 **cmax** 范围区内的数据使用整个颜色映象。比 **cmax** 大的数据点用与 **cmax** 值相关的颜色绘图，比 **cmin** 小的数据点的颜色用与 **cmin** 值相关的颜色绘图。如果 **cmin** 小于 **min(data)**和/或 **cmax** 大于 **max(data)**，那么与 **cmin** 和/或 **cmax** 点相关的颜色将永远用不到。也就是说，只用到和数据相关的那一部分颜色映象。» **caxis(** ‘**auto**’ **)**设置 **cmin** 和 **cmax** 的缺省值。

由于下面的例子很难在书中清晰区分灰度，运行脚本 M 文件 **mmcaxisd.m** 可显示所包含的一系列更多的例子。缺省的颜色范围由下例说明：

```
» pcolor([1:17;1:17]' ),colormap(hsv(8))
» title( 'Default Color Range' )
» caxis( 'auto' )
» colorbar
» caxis
ans =
      1      17
```

输出见图 19.4.

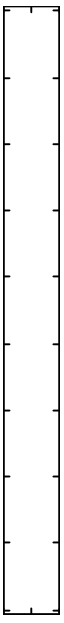


图 19.4　缺省的颜色范围

可见对整个数据集合，当前颜色映象使用了所有 8 种颜色。每种颜色有两条。如果颜色被映射到从-3 到 23 的数据，那么，图中只用到五种颜色。这可以通过下面的命令实现：

```
» title( ‘Extended Color Range’ )
» caxis([-3,23])    %    extended the color range
» colorbar    %    redraw the color scale
```

输出见图 19.5.

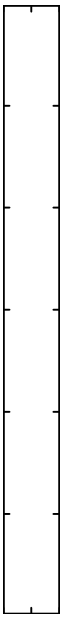


图 19.5　扩展的颜色范围

如果颜色映射到从 5 到 12 的值，会用到所有的颜色。但是，比 5 小的数据和比 12 大的数据分别映射到与数据值 5 和 12 相关的颜色。这可以通过下面的命令产生：

```
» title( ‘Restricted Color Range’ )
» caxis([5,12])    %    resstrict the color range
» colorbar    %    redraw the color scale
```

输出见图 19.6.

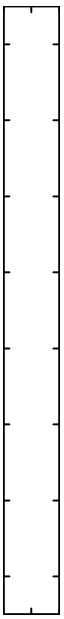


图 19.6　受限的颜色范围

19.5 图形中使用一个以上的颜色映象

有时，在一幅图的不同部分使用不同的颜色是很有作用的。由于颜色映象是图形窗口本身的一个属性，在任意一个图形窗口中，只能用一个颜色映象。但是，可以创建自己的颜色映象来达到想要的效果。例如，精通 MATLAB 工具箱中含有脚本 M 文件 **mmcmapd.m**，它执行下述操作。

```
» figure    %    create a figure window
» mymap=[rainbow(32);copper(32)];    %    stack two color maps into one
» colormap(mymap)    %    install it
» mesh(peaks+8);view(0,0);    %    create two sample polots
» hold on ;mesh(peaks-8);
» colorbar    %    and add a color scale
» title( 'Merging two colormaps' )
» hold off
```

输出见图 19.7.

图 19.7　合并两个颜色映象

19.6 用颜色描述第四维

一些函数，比如 **mesh** 和 **serf**，除非给出颜色参量，颜色将沿 z 轴数据变化。比如，**surf(X，Y，Z**）等效于 **surf(X，Y，Z，Z**）。将颜色施加于 z 轴能够产生色彩漂亮的图画，但由于 z 轴已经存在，它并不提供新的信息。为更好的利用颜色，建议用颜色来描述不受三个轴影响的数据的某些属性。为此需要赋给三维作图函数的颜色参量不同的数据。

如果作图函数的颜色参量是一个向量或矩阵，那么就用作颜色映象的下标。这个参量可以是任何实向量或与其参量维数相同的矩阵。考虑下面这些例子：

```
» x=-7.5: .5 : 7.5; y=x;    %    create a data set - the frame scmbrero
» [X Y]=meshgrid(x,y);    %    create placid data
» R=sqrt(X.^2+Y.^2)+eps;
» Z=sin(R)./R
» surf(X,Y,Z,Z)    %    default color order
» surf(X,Y,Z,-Z)    %    reverse the default color order
» surf(X,Y,Z,X)    %    color varies along the X-axis
» surf(X,Y,Z,X+Y)    %    color varies along the XY diagonal
» surf(X,Y,Z,R)    %    color varies radially from the center
» surf(X,Y,Z,abs(del2(Z)))    %    color varies with absolute value of Laplacian
» [dZdx,dZdy]=gradient(Z);    %    compute gradient or slope of surface
» surf(X,Y,Z,abs(dZdx))    %    color varies with absolute slope in x-direction
» surf(X,Y,Z,abs(dZdy))    % color varies with absolute slope in y-direction
» dz=sqrt(dZdx.^2+dZdy.^2);
» surf(X,Y,Z,dZ)    %    color varies with magnitude of slope
```

输出分别见图 19.8、图 19.9、图 19.10、图 19.11、图 19.12、图 19.13 和图 19.14.

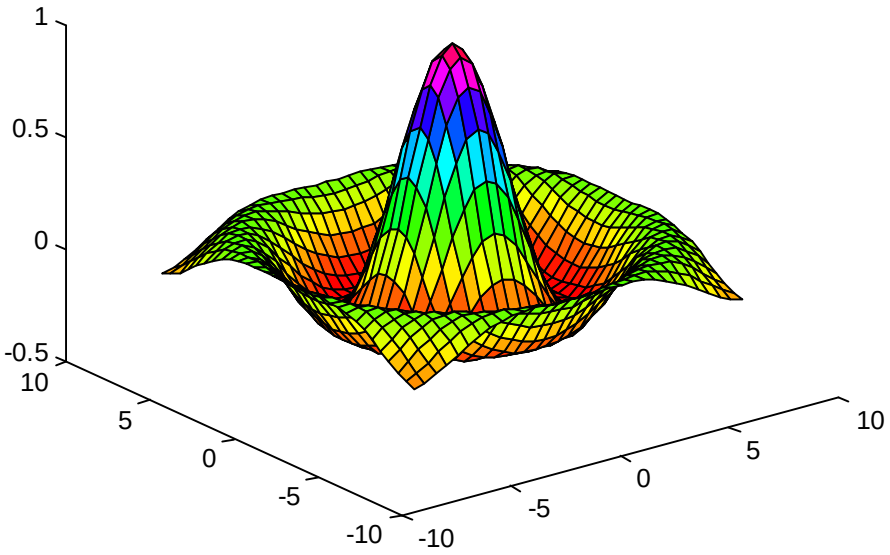


图 19.8　surf(X,Y,Z,Z)

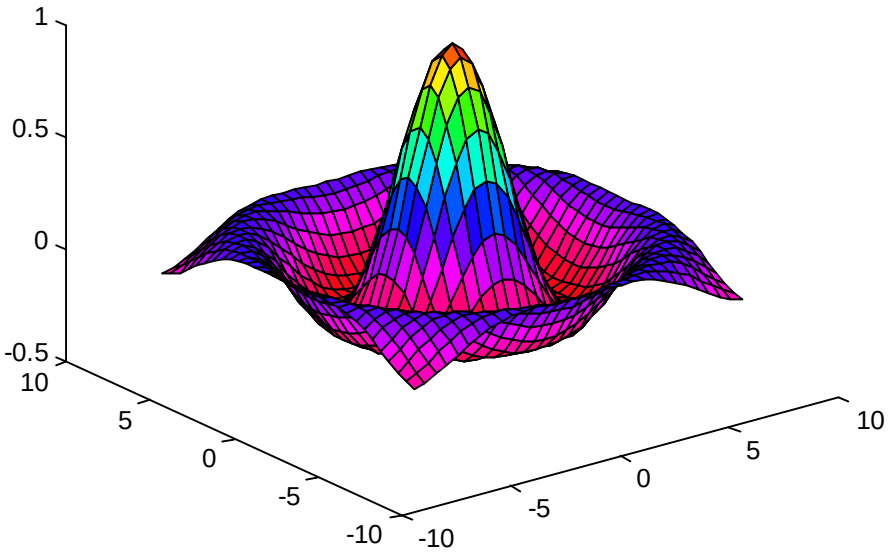


图 19.9　surf(X,Y,Z,-Z)

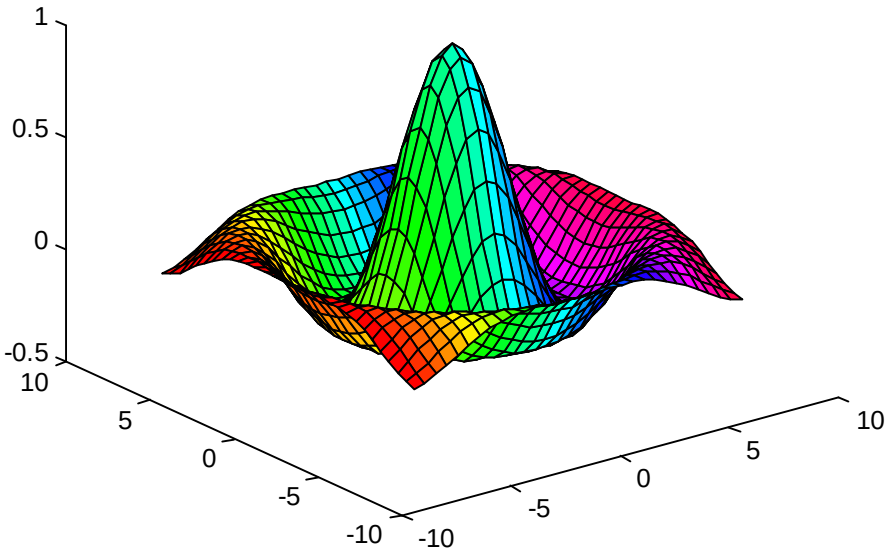


图 19.10　surf(X,Y,Z,X)

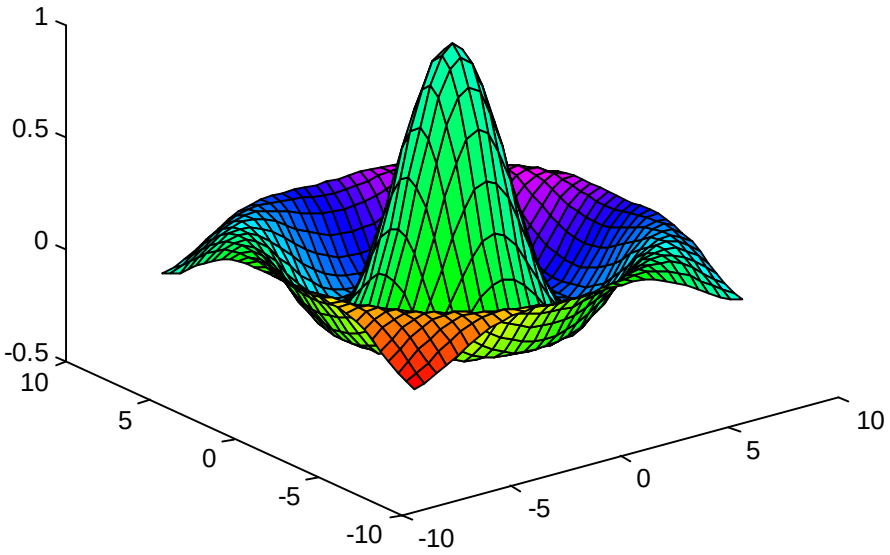


图 19.11　surf(X,Y,Z,X+Y)

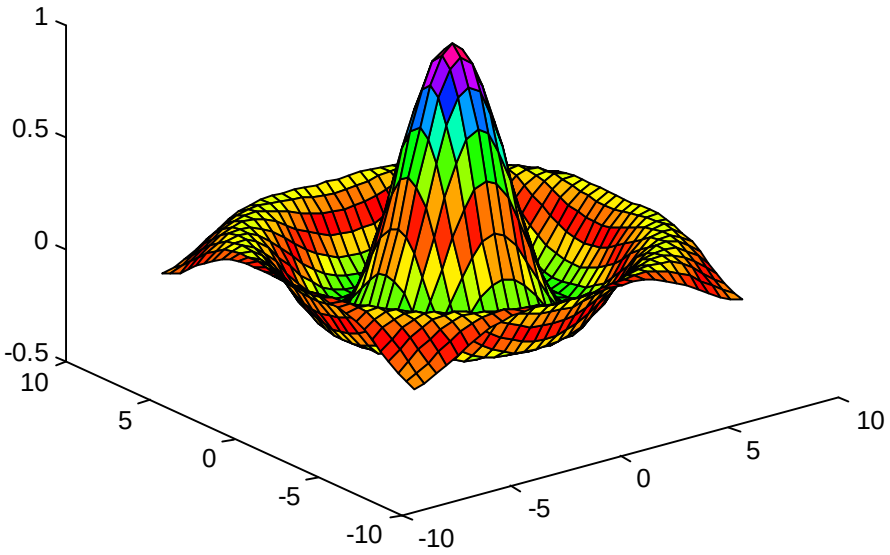


图 19.12　surf(X,Y,Z,R)

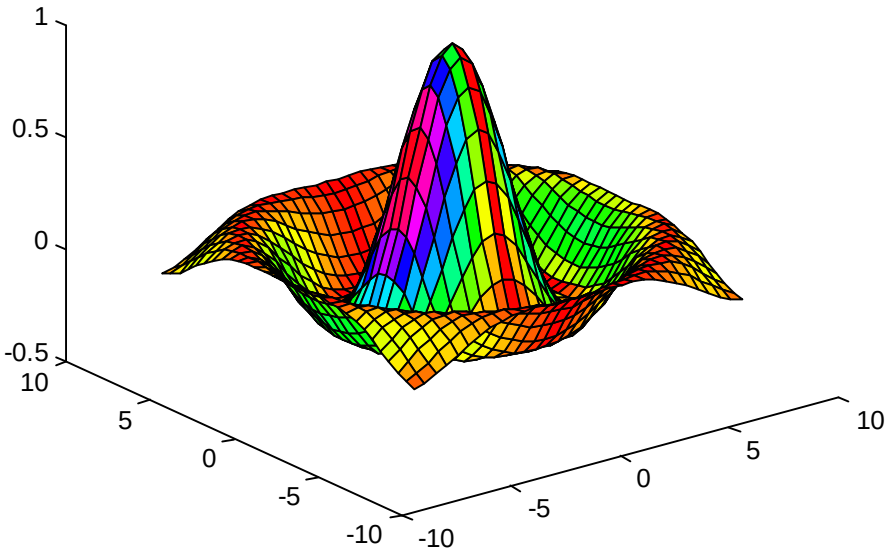


图 19.13　surf(X,Y,Z,abs(del2(Z)))

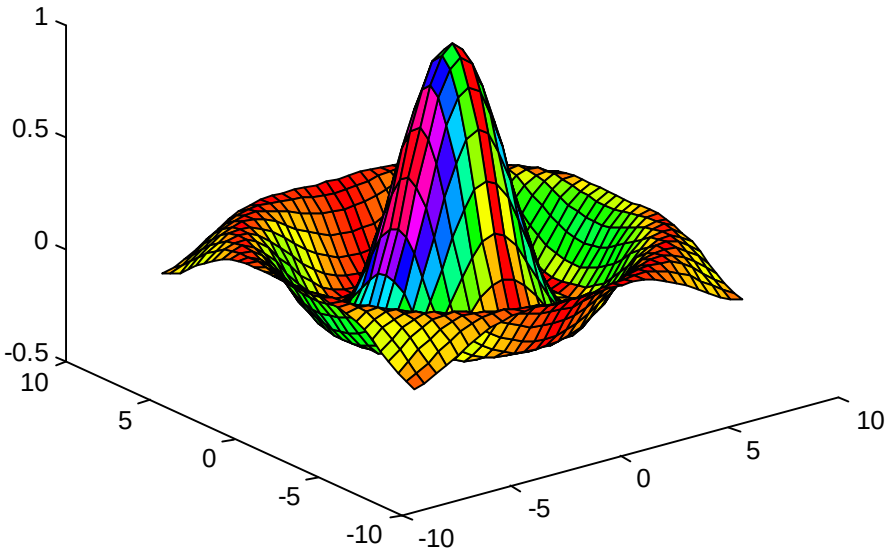


图 19.14　surf(X,Y,Z,dZ)

图 19.15　surf(X,Y,Z,abs(dZdx))

图 19.16　surf(X,Y,Z,abs(dZdy))

注意到上面后五个例子中，颜色如何为所画的曲面提供了一个附加的维数。函数 **del2** 是离散拉氏函数，它根据表面弯曲程度来使用颜色。函数 **del2** 描述如下：

```
» help del2
DEL2 Five-point discrete Laplacian.
    V=del2(U) is a matrix the same size as U with each element equal to the difference
    between an element of U and the average of its four neighbours.For the “corners” and
    “edges”,only two or three neighbours are used.
    See also GRADIENT,DIFF
```

帮助信息：

DEL2　五点的离散 Laplacian

V=del2(U)是一个和 U 同样大小的矩阵。它的每个元素是 U 中的对应位置的元素和它 的四个相邻点元素的平均值的差值。对于角上和边上的元素，只使用两个或三个相邻点。

参阅函数：GRADIENT ,DIFF

如上述，函数 **gradient** 逼近表面的梯度或坡度。为了方便，通过运行精通 MATLAB 工具箱中的脚本 M 文件 **mm4d**，便可执行上述命令。

19.7 照明模型

基于运用漫射、镜面反光和环境照明模型，函数 **surfl** 画出了一个类似于函数 **surf** 产生的带彩色的曲面。使用一个单色颜色映象（如**灰色**，**纯白**，**铜黄**或**粉红色**）和插值色彩，会画出效果最好的曲面。

正常的参量表为 **surfl（X，Y，Z，S）**，这里 **X**，**Y** 和 **Z** 与 **surf(X，Y，Z）**相同。而 **S** 以**[Sx，Sy，Sz]**或**[az , el]**的形式定义了光源的方向。如果没有指明，其缺省光源是逆时针 45 度，即从现在的视角向右转 45 度。

环境照明，漫射反射，镜面反光对视觉效果的相对贡献以及镜面扩展因子可以通 过**K=[ka,kd,ks,spread]**的五个元素来设定，这里 **K** 是函数 **surfl** 的第五个参量，即 **surfl（X，Y，Z，S，K）**。**K** 的缺省值是**[.55　.6　.4　10]**。为了了解这些参量如何影响图形照明，可以参阅下面这些例子。

```
» [X,Y,Z]=peaks(32);　%　data to plot
» surfl(X,Y,Z) , colormap(copper) , title( ‘Default Lighting’ ) , shading interp
» surfl(X,Y,Z,[7.5　30],[.55　.6　.4　10]) , shading interp
» surfl(X,Y,Z,[-90　30],[.55　.6　2　10]) , shading interp
```

如前所述，插值上色会极大地降低打印速度。这是因为每一象素都有一个不同的颜色值，打印机对每点都要分别地上色。

19.8 小结

本章所用的函数总结在表 19.3、表 19.4、表 19.5 和表 19.6 中。

表 19.3

简单颜色			
Red（红）	Green（绿）	Blue（蓝）	颜色
0	0	0	黑
1	1	1	白
1	0	1	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
. 5	0	0	深红
. 5	. 5	. 5	灰色

表 19.4

标准颜色映象	
hsv	色彩饱和值（以红色开始和结束）
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	**hsv** 的一种变形（以蓝色开始和结束）
copper	线性铜色度
prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
fag	交替为红色、白色、蓝色和黑色

表 19.5

在 surf，mesh 和 pcolor 图中作第四维的颜色	
surf(X,Y,Z,fun(X,Y,Z))	根据函数 **fun(X,Y,Z)**来施加颜色
surf(X,Y,Z)=surf(X,Y,Z,Z)	缺省动作，加颜色于 Z 轴
surf(X,Y,Z,X)	加颜色于 X 轴
surf(X,Y,Z,Y)	加颜色于 Y 轴
surf(X,Y,Z,X.^2+Y.^2)	根据 **z=0** 平面距原点（**x=0，y=0**）的距离施加颜色
surf(X,Y,Z,del2(Z))	根据曲面的拉氏函数值施加颜色

[dZdx,dZdy]=gradient(Z); **surf(X,Y,Z,abs(dZdx))**	根据 X 轴方向的曲面斜率施加颜色
dz=sqrt(dZdx.^2+dZdy.^2); **surf(X,Y,Z,dz)**	根据曲面斜率大小施加颜色

表 19.6

颜色和照明函数	
colormap(map)	在当前的图形窗口中安装一个颜色映象
coorbar	在当前的图形上显示一个水平的或垂直的颜色标尺
rgbplot(map)	颜色映象中红、绿、蓝分量的直线图
brighten(a)	**0<a<1**，当前颜色映象加亮；**-1<a<0**，当前颜色映象加暗
m=brighten(map,a)	返回加亮的颜色映象 **m**
[cmin,cmax]=caxis	返回颜色轴的界限
caxis([cmin,cmax]）	设置颜色轴的界限

关键词索引

chap 19

data visualization 数据可视化

lighting model 照明模型

hardware color lookup table 硬件查色表

entry 输入项

color scale 颜色标尺

color bar 颜色条

dark color 暗色

color axis 颜色轴

discrete Laplacian function 离散拉氏函数
diffuse 漫射
specular 镜面反光
ambient lighting model 环境照明模型
specular-spread coefficient 镜面扩展因子

第二十章　句柄图形

`　什么是句柄图形？句柄图形是对底层图形例程集合的总称，它实际上进行生成图形的工作。这些细节通常隐藏在图形 M 文件的内部，但如果想使用它们也是可得到的。

MATLAB 用户指南给人的一种印象是，句柄图形非常复杂，只对熟练的高级用户才有用。而实际上不是这样的。句柄图形可以被任何人用来改变 MATLAB 生成图形的方式，不论是只想在一幅图里做一点小变动，还是想做影响所有图形输出的全局变动。

句柄图形允许你定制图形的许多特性，而这用高级命令和前几章里描述的函数是无法实现的。例如，如果想用橘黄色来画一条线，而不是 **plot** 命令中可用的任何一种颜色，该怎么做呢？句柄图形就可以提供一种方法。

本章不对句柄图形作详细讨论，因为那样涉及问题太细。这里的目的只是对句柄图形概念作基本了解，并提供足够多的信息，使得即使是偶尔使用一下 MATLAB 的用户也可以利用句柄图形。在这个背景下，在本章最后给出了关于句柄图形对象属性和它们的值，它不仅很有用也很有意义。

20.1 谁需要句柄图形？

开始，我们要强调本章主要是针对那些不满足于 MATLAB 普通图形特性的读者。如果对所画的图形已经很满意，那么就跳过当前的讨论。如果以后要定制图形，只要记住这里有可用的信息。

现在，对于那些还在犹豫的用户，我们要强调学习使用句柄图形并不困难。如果只想改变图形的标题字体，或者改变一个图形窗口的背景颜色，那么，你不必成为一个句柄图形的专家也可做到。

另一方面，如果想定制图形，并且要打算对图形的每个可能方面进行控制，那么句柄图形会为此提供强有力的工具。

前面那些章提供的图形功能被认为是高级的命令和函数，包括 **plot**，**mesh**，**axis** 及其它。这些函数是建立在底层函数和属性的基础上，总称为句柄图形。

20.2 什么是句柄图形对象

句柄图形是基于这样的概念，即一幅图的每一组成部分是一个对象，每一个对象有一系列句柄和它相关，每一个对象有按需要可以改变的属性。

当今计算机行业最流行的术语之一便是对象这个词。面向对象的编程语言，数据库对象，

操作系统和应用程序接口都使用了对象的概念。一个对象可以被粗略地定义为由一组紧密相关、形成唯一整体的数据结构或函数集合。**在 MATLAB 中，图形对象是一幅图中很独特的成分，它可以被单独地操作**。

由图形命令产生的每一件东西都是图形对象。它们包括图形窗口或仅仅说是图形，还有坐标轴、线条、曲面、文本和其它。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其它对象的父亲。图形窗口是根对象的子对象；坐标轴和用户界面对象（在下一章讨论）是图形窗口的子对象；线条、文本、曲面、补片和图象对象是坐标轴对象的子对象。这种层次关系在图 20-1 中给出。

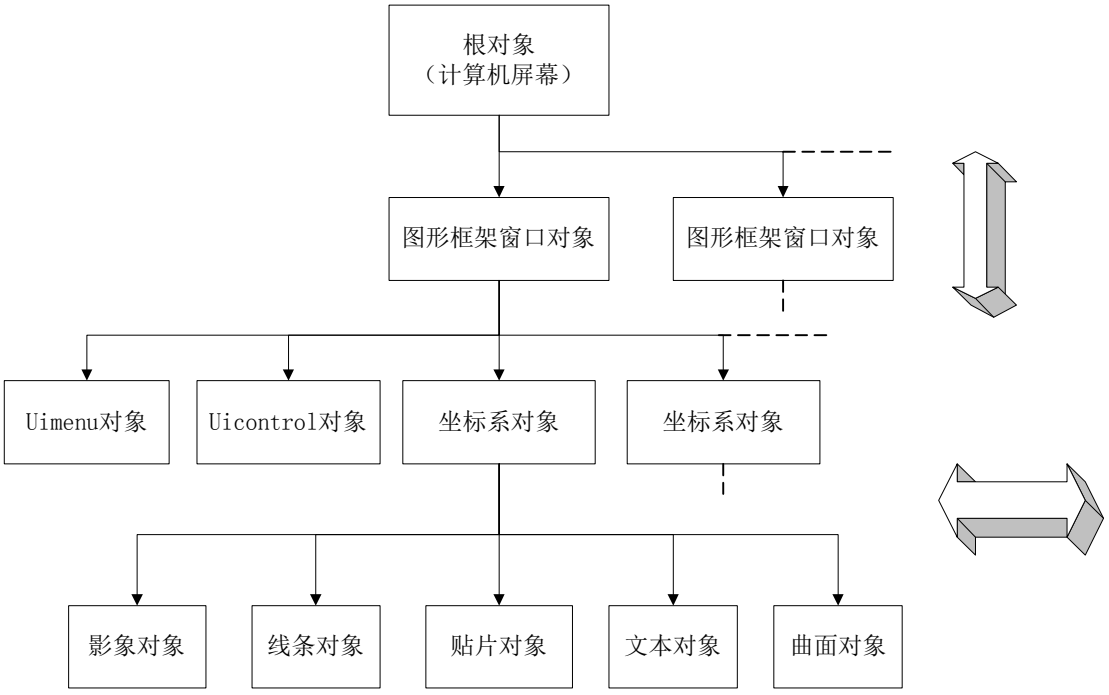


图 20-1　对象层次结构

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其它的对象（除了在下一章讨论的 *uicontrol* 和 *uimenu* 外）都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，如果没有图形窗口，**plot(rand(size([1：10])))**函数会用缺省属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

20.3 句柄对象

假设已打开了三个图形窗口，其中两个有两幅子图。并要改变其中一幅子图坐标轴内一条线的颜色，如何认定想要改变的那条线？在 MATLAB 中，每一个对象都有一个数字来标识，叫做**句柄**。

每次创建一个对象时，就为它建立一个唯一的句柄。计算机屏幕作为根对象常常是 0。**» Hf_fig=figure** 命令建立一个新的图形窗口，变量 **Hf_fig** 中返回它的句柄值。图形窗口的句柄为整数，通常显示在图形窗口标题条中。其它对象句柄是 MATLAB 满精度的浮点值。

MATLAB 可以用来获得图形、坐标轴和其它对象的句柄。例如，» **Hf_fig=gcf** 返回当前图形窗口的句柄值，而» **Ha_ax=gca** 返回当前图形窗口内当前坐标轴的句柄值。这些函数和其它对象操作的工具在本章以后讨论。

为了提高可读性，在本书中包含句柄对象的变量取名以大写的 **H** 开头，跟之以一个辨识对象类型的字母，然后是一个下划线，最后是一个或几个描述符。因此，**Hf_fig** 是一个图形窗口的句柄，**Ha_ax1** 是坐标轴对象的句柄，而 **Ht_title** 是一个文本对象的句柄。当对象类型不知道时，用字母 **x**，比如 **Hx_obj**。虽然句柄变量可以取任意名字，遵循这种规则使得能在 M 文件中很容易找到句柄变量。

所有产生对象的 MATLAB 函数都为所建立的每个对象返回一个句柄（或句柄的列向量）。这些函数包括 **plot**，**mesh**，**surf** 及其它。有一些图形由一个以上对象组成。比如，一个**网格图**由一个曲面组成，它只有一个句柄；而 **waterfall** 图形由许多线条对象组成，每个线条对象都有各自的句柄。例如，» **Hl_wfall=waterfall(peaks(20))** 对线条返回一个包含着 20 个句柄的列向量。

20.4 通用函数 get 和 set

所有对象都有**属性**来定义它们的特征，正是通过设定这些属性来修正图形显示的方式。尽管许多属性所有的对象都有，但与每一种对象类型（比如坐标轴，线，曲面）相关的属性列表都是独一无二的。对象属性可包括诸如对象的位置、颜色、类型、父对象、子对象及其它内容。每一个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。和每一种对象类型(图形，坐标轴，线，文本，曲面，补片和图象)相关的完整的属性列表在本章的后面给出。

对象属性包括属性名和与它们相联关的值。属性名是字符串，它们通常按混合格式显示，每个词的开头字母大写，比如：' **LineStyle** ' 。但是，MATLAB 识别一个属性时是不分大小写的。另外，只要用足够多的字符来唯一地辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用 ' **Position** ' ， ' **position** ' ，甚至是 ' **pos** ' 来调用。

当建立一个对象时，它用一组缺省属性值，该值可以用两种方法来改变。可以用{**属性名，属性值**}对来建立对象生成函数；或者在对象建立后改变属性。前一种方法的例子是：

```
» Hf_1=figure( 'color', 'white' )
```

它用缺省的属性值建立一个新的图形窗口，只是背景颜色被设为白色而不是缺省的黑色。

为了获得和改变句柄图形对象的属性只需要两个函数。函数 **get** 返回某些对象属性的当前值。使用函数 **get** 的最简单语法是 **get(handle, ' PropertyName')**。例如：

```
» p=get(Hf_1, 'position' )
```

返回具有句柄 **Hf_1** 图形窗口的位置向量。

```
» c=get(Hl_a, 'color' )
```

返回具有句柄 **Hl_a** 对象的颜色。

函数 **set** 改变句柄图形对象属性，使用语法 **set(handle,** '**PropertyName**' **,value)**。例如：

» set(Hf_1, 'Position' ,p_vect)

将具有句柄 Hf_1 的图形位置设为向量 **p_vect** 所指定的值。同样

» set(Hl_a, 'color' , 'r')

将具有句柄 **Hl_a** 的对象的颜色设置成红色。

一般情况下，函数 **set** 可以有任意数目的**(** '**PropertyName**' **,PropertyValue)**对。比如：

» set(Hl_a, 'Color' , 'r' , 'Linewidth' ,2, 'LinStyle' , '--' ）

将具有句柄 **Hl_a** 的线条变成红色，线宽为 2 点，线型为破折号。

除了这些主要功能，函数 **set** 和函数 **get** 还能提供帮助。例如» **set(handle,** '**PropertyName**' **)**返回一个可赋给由 **handle** 所描述对象的属性值列表。例如：

» set(Hf_1, 'Units')
[inches|centimeters|normalized|points|{pixels}]

表明由 **Hf_1** 所引用的图形的 '**Unites**' 属性是五个可允许的字符串，而其中 '**pixels**' 是缺省值。

如果指定一个没有固定值的属性，那么，MATLAB 就会通知如下：

» set(Hf_1, 'Position')
A figure's 'Position' property does not have a fixed set of property values。

除了 **set** 命令，句柄图形对象创建函数（例如 **figure**，**axis**，**line** 等等）接受多个属性名和属性值对。例如：

» figure('Color' , 'blue' , 'NumberTitle' , 'off' , 'Name' , 'My Figure' ）

创建一个图形窗口，背景为兰色，标有 **'My Figure'** 而不是缺省标题 **'Figure No. 1'**。

为了形象说明上述概念，考虑下面的例子：

```
» Hf_fig =figure    %    create a figure having an interger handle
Hf_fig=
        1
» Hl_line=line    %    create a line having a floating-pointer handle
Hl_line =
    59.0002
» set(Hl_line);    %    list settable properties and potential values
      Color
```

```
    EraseMode: [ {normal} | background | xor | none ]
    LineStyle: [  {-}  |  --  |  :  |  -.  |  +  |  o  |  *  |  .  |  x  ]
    LineWidth
    MarkerSize
    Xdata
    Ydata
    Zdata

    ButtonDownFcn
    Clipping: [ {on} | off ]
    Interruptible: [ {no} | yes ]
    Parent
    UserData
    Visible: [ {on} | off ]

» get(Hl_line);   %   list properties and current property values
Color = [1 1 1]
    EraseMode = normal
    LineStyle = -
    LineWidth = [0.5]
    MarkerSize = [6]
    Xdata = [0 1]
    Ydata = [0 1]
    Zdata = [ ]

    ButtonDownFcn =
    Children = [ ]
    Clipping = on
    Interruptible = no
    Parent = [58.0002]
    Type = line
    UserData = [ ]
    Visible = on
```

在上例中，所创建的线条中的 '**Parent**' 属性就是包含线条的坐标轴的句柄。而且所显示的图形列表被分为两组。在空行上的第一组，列出了该对象的独有属性，而空行下的第二组列出所有的对象共有的属性。注意到函数 **set** 和函数 **get** 返回不同的属性列表。函数 **set** 只列出可以用 set 命令改变的属性，而 **get** 命令列出所有对象的属性。在上面的例子中，函数 **get** 列出了 '**Children**' 和 '**Type**' 属性，而 **set** 命令却没有。这一类属性只可读，但不能被改变，它们叫做只读属性。

与每一个对象有关的属性数目是固定的，但不同的对象类型有不同数目的属性。象上面所显示的，一个线条对象列出了 16 个属性，而一个坐标轴对象列出了 64 个属性。显然，透彻地说明和描述所有对象类型的全部属性超出本书的范围。但是，其中的很多属性本书以后要详细讨论，并且列出全部属性。

作为一个使用图象句柄的例子，可以考虑前面提出的问题。它要用非标准颜色画一条线。在这里，线的颜色用 **RGB** 值**[1 .5 0]**来指定，它是适中的橘黄色。

```
» x=-2*pi:pi/40:2*pi;  %  create data
» y=sin(x);  %  find the sine of x
» Hl_sin=plot(x,y)  %  plot sine and save line handle
Hl_sin=
    59.0002
» set(Hl_sin, 'Color' ,[1 .5 0], 'LineWidth' ,3)  %  Change the color and width
```

现在加一个浅蓝色的 cosine 曲线：

```
» z=cos(x);  %  find the cosine of x
» hold on  %  keep the sine curve
» Hl_cos=plot(x,z);  %  plot the cosine and save the handle
» set(Hl_cos, 'Color' ,[.75  .75  1])  %  color it light blue
» hold off
```

也可以用较少的步骤来实现同样的功能：

```
» Hl_lines=plot(x,y,x,z);  %  plot both curves and save both handles
» set(Hl_line(1), 'Color' ,[1 .5 0], 'LineWidth' ,3)
» set(Hl_line(2), 'Color' ,[.75 .75 1])
```

如何加上一个标题并且使字体比正常大一些呢？

```
» title( 'Handle Graphics Example' )  %  add a title
» Ht_text=get(gca, 'Title' )  %  get a handle to the title
» set(Ht_text, 'FontSize' ,16)  %  customize the font size
```

最后一个例子说明了关于坐标轴对象令人感兴趣的性质。每一个对象都含有 '**Parent**' 属性和 '**Children**' 属性，该属性包含属于派生对象的句柄。画在一组坐标轴上的线，具有当作 '**Parent**' 属性值的坐标轴对象的句柄，而 '**Children**' 属性值是一个空距阵。同时，这个坐标轴对象具有当作 '**Parent**' 属性值的图形句柄，而 '**Children**' 属性值是线条对象的句柄。标题字符串和坐标轴的标志不包含在坐标轴的 '**Children**' 属性值里，而是保存在 '**Title**' 、 '**Xlabel**' 、 '**Ylabel**' 和 '**Zlabel**' 的属性内。创建坐标轴对象时，这些文本对象就建立。**title** 命令设置当前坐标轴内标题文本对象的 '**String**' 属性。最终，标准 MATLAB 的函数 **title,xlabel,ylabel** 和 **zlabel** 不返回句柄，而只接受属性和数值参量。例如下面的命令给当前图加一个 24 点的绿色标题：

```
» title( 'This  is  a  title.' , 'Fontsize' ,24, 'Color' , 'green' )
```

除了函数 **set** 和 **get**，MATLAB 还提供了另外两个函数来操作对象和它们的属性。任意对象和它们的子对象可以用 **» delete(handle)** 来删除。同样 **» reset(handle)** 将与句柄有关的

全部对象属性（除了‘**Position**’属性）重新设置为该对象类型的缺省值。

20.5 查找对象

正如你所见，句柄图形提供了对图形对象的访问途径，并且允许用函数 **get** 和 **set** 定制图形。但是，如果忘记保存句柄或图中对象的句柄将会怎样呢？或者，也许变量被覆盖又如何呢？如果不知道它们的句柄，怎么改变对象的属性呢？为解决这个问题，MATLAB 提供了查找对象句柄的工具。

gcf 和 **gca** 这两种工具前面已经介绍过了。

```
» Hf_fig=gcf
```

返回当前图形的句柄，而

```
» Ha_ax=gca
```

返回当前图形的当前坐标轴的句柄。

除了这些，还有一个获取当前对象的 **gco**。

```
» Hx_obj=gco
```

返回当前图形的当前对象的句柄。或者用另一种方法：

```
» Hx_obj=gco(Hf_fig)
```

返回与句柄 **Hf_fig** 有关的图形中当前对象的句柄。

当前对象的定义为用鼠标刚刚点过的对象。这种对象可以是除根对象（计算机屏幕）之外的任何图形对象。但是，如果鼠标指针处在一个图形中而鼠标按钮未点，**gco** 返回一个空距阵。为了让当前对象存在，我们必须选择一些东西。

一旦获得了一个对象的句柄，它的对象类型可以通过查询对象的‘**Type**’属性来获得。该属性是一个字符串对象名，比如‘**figure**’，’**axes**’或‘**text**’。例如：

```
» x_type=get(Hx_obj, ‘Type’ )
```

MATLAB 中的函数 **gcf**，**gca** 和 **gco** 是很好的例子，它们说明如何利用句柄图形来获得有关对象的信息。函数 **gcf** 获得根对象的‘**CurrentFigure**’的属性值，即是当前图形的句柄。**gcf** M 文件包含：

```
function h=gcf()
%   GCF    Get current figure handle.
%   H=GCF returns the handle to the current figure.The current fugure is the figure(graphics window)that graphics commands like PLOT,TITLE,SURF,etc.draw to if issued.
%
```

```
%    Use the commands FIGURE to change the current figure to a diffent figure,or to create
new      %    ones.
%
%    See also FIGURE,CLOSE,CLF,GCA.
%    Copyright    (c)    1984-94 by The MathWorks,Inc.

h=get(0, 'CurrentFigure' );
```

类似的，函数 **gca** 返回当前图形的 '**CurrentAxes**' 属性值，它的 M 文件描述如下。

```
function h=gca()
%    GCA    Get current axis handle.
%     H=GCA returns the handle to the current axis.The current axis is the axis that graphics
%    command like PLOT,TITLE,SURF,etc.draw to if issued.
%
%    Use the commands AXES or SUBPLOT to change the current axis to a different axis,or to
%    create new ones.
%    see also AXES,SUBPLOT,DELETE,CLA,HOLD,GCF.
%    Copyright    (c)    1984-94 by The MathWorks,Inc.
h=get(get(0, 'CurrentFigure' ), 'CurrentAxes' );
```

函数 **gco** 也相同，只是它在试图获得当前对象之前先检查图形是否存在。**注意函数 gcf 和 gca 能促使建立相关的对象，如果它们不存在的话**。如下所示的函数 **gco**，它先检查子对象（ '**Children**' ）是否存在，如果不存在，就不创建图形对象。

```
function object=gco(figure)
%GCO    Handle of current object.
%    OBJECT=GCO returns the current object in the current figure.
%
%    OBJECT=GCO(FIGURE) returns the current object in figure FIGURE.
%
%    The current object for a given figure is the last object clicked on with mouse.

%Copyright    (c)    1984-94 by The MathWorks,Inc.

if isempty(get(0, 'Children' ))
      object=[ ];
      return;
end;

if (nargin==0)
      figure=get(0, 'CurrentFigure' );
end
```

```
object=get(figure, 'CurrentObject' );
```

当需要一些除了 '**CurrentFigure**' 、 '**CurrentAxes**' 和 '**CurrentObject**' 之外的某些东西时，可以用函数 **get** 来获得一个对象的子对象的句柄向量。例如：

```
» Hx_kids=get(gcf, 'Children' )
```

返回一个向量，它包含当前图形子对象的句柄。

可以用获得子对象 '**Children**' 句柄的技术彻底搜索句柄图形的层次结构中来找到所要的对象。例如，在画出一些数据后，寻找绿色线条句柄的问题。

```
» x=-pi:pi/20:pi;    %    create some data
» y=sin(x);z=cos(x);
» plot(x,y, 'r' ,x,z, 'g' );    %    plot two lines in red and green
» Hl_lines=get(gca, 'Children' );    %    get the line handles
» for k=1:size(Hl_lines)    %    find the green line
»if get(Hl_lines(k), 'Color' )==[0 1 0]
»              Hl_green=Hl_lines(k)
»       end
» end
Hl_green=
      58.0001
```

尽管这种技术有效，但是如果有很多对象存在就变得复杂。该技术也丢失了标题和坐标轴标志中的文本对象，除非能逐个检测这些对象。

当有多个图形，每个图形上又有多个坐标轴时，考虑查找所有绿色线条的句柄的问题。

```
» Hf_all=get(0, 'Children' );    %    get all figure handles
» for k=1:length(Hf_all)
»      Ha_all=[Ha_all;get(Hf_all(k), 'Children' )];    %    get all axes handles
» end
» for k=1:length(Ha_all)
»      Hx_all=[Hx_all;get(Ha_all(k), 'Children' )];    %    get axes child handles
» end
» for k=1:length(Hx_all)
»      if get(Hx_all(k), 'Type' )== 'line'
»          Hl_all=[Hl_all;Hx_all(k)];    %    get line handles only
»      end
» end
» for    k=1:length(Hl_all)
»      if get(Hl_all(k), 'Color' )==[0 1 0]
»          Hl_green=[Hl_green;Hl_all(k)];    %    find green ones
»      end
» end
```

为了简化查找对象句柄的过程，MATLAB 4.2 版本和以后的版本提供了内置函数 **findobj**，该函数返回有指定属性值的所有对象句柄。它的在线帮助如下：

» help findobj

FINDOBJ Find objects with specified property values.

H=FINDOBJ('PlName' ,PlValue,...) returns the handle of the objects at the root level and below whose property values match those as param-value pairs to the FINDOBJ command.

H=FINDOBJ(ObjectHandles, 'PlName' ,PlValue,...) restricts the search to the objects listed in ObjectHandle and their descendents.

H=FINDOBJ(ObjectHandles, 'flat' , 'PlName' ,PlValue,...) restrics the search only to the objects listed in ObjectHandles.Their descendents are not searched.

H=FINDOBJ returns the handles of the root object and all its descendents.

H=FINDOBJ(ObjectHandles) returns the handles listed in Objecthandles ,and the handles of all their descendents.

See also SET,GET,GCF,GCA.

帮助信息：

FINDOBJ 寻找具有指定的属性值的对象

H=FINDOBJ('P1Name' ， P1Value，…）返回根部和根部以下那些属性值与 FINDOBJ 参数对相匹配的对象句柄。

H=FINDOBJ(ObjectHandles， 'P1Name' ， P1Value，…）搜索限定在 ObjectHandles 中列出的对象和它们的子对象。

H=FINDOBJ(ObjectHandles， 'flat' ， 'P1Name' ，P1Value，…）搜索限定在 ObjectHandles 中列出的对象，不搜索它们的子对象。

H=FINDOBJ 返回根对象和它所有的子对象的句柄。

H=FINDOBJ(ObjectHandles）返回 ObjectHandles 中列出的对象和它们的子对象的句柄。

参阅 SET，GET，GCF，GCA。

函数 **findobj** 返回符合所选判据的对象的句柄。它检查所有的 '**Children**' ，包括坐标轴的标题和标志。如果没有对象满足指定的判据，**findobj** 返回空距阵。

用函数 **findobj**，前面的例子变成一行：

» Hl_green=findobj(0, 'Type' , 'line' , 'Color' ,[0 1 0]);

20.6 用鼠标选择对象

gco 命令返回当前对象的句柄，该对象就是用鼠标刚刚点击过的对象。然而，MATLAB 怎么知道哪个对象被选中了呢？例如，当点击一幅图中两条线的交点时，应该返回哪个句柄？或者当鼠标点击时指针离线有多远仍能选中该线？这些答案要根据 MATLAB 选择对象规则和**堆积次序**。

堆积次序决定哪一对象叠加在其它对象上。开始时，堆积次序在对象被创建时就被决定，最后创建的对象在堆栈的顶部。例如，如果发出两条 **figure** 命令，就产生两个图形。第二个图画在第一个的上面。而最终的堆积次序是图 2 在图 1 的上面，当前图形 **gcf** 是图 2。如果发出 **figure(1)**命令，或者点击图 1，堆积次序就改变，图形框架 1 移动到堆栈的顶端，成为当前图形。

在前面的例子中，在计算机屏幕上，堆放次序可以从窗口交叠中显而易见。但是，情况并不总是这样的。如果画了两条线，在它们的交叉点，第二条线在第一条线的上面。如果用鼠标在第一条线其它一些点上点击，那么第一条线条就在栈顶，用鼠标点击交叉点会选中第一条线。**当前的堆积次序是由一个给定的对象 'Children' 句柄出现的次序给出的。也就是说，» Hx_kids=get(handle,'Children')按堆积次序返回 Children 句柄。存储句柄的向量 Hx_kids 的第一个元素在栈顶，而最后一个元素在栈底。**

除了堆积次序，当用鼠标在一个对象附近点击时，该对象也会被选中。每一种对象类型都有与之相关的它自己的选择区域。例如，对于一条线来说，如鼠标的指针在线条的 5 个象素之内，它就会被选中。曲面，补片和文本对象的选择区域是包含对象的最小矩形。坐标轴对象的选择区域是坐标框本身加上标题和标志出现的区域。坐标轴内的对象，例如线条和曲面，在堆积次序中处于较高地位。因此，点击它们时会选择相关的对象而不是坐标轴。选择坐标轴选择区以外的区间就会选中图形本身。

这里讨论的堆积次序和对象选择规则是针对 MATLAB 4.02c 的版本。但要注意，MATLAB 定义、使用堆积次序和对象选择的方法在 5.0 版本中可能会扩充。

20.7 位置和单位

一个图形对象和许多其它句柄图形对象的位置属性 '**Position**' 是一个 4 元素的行向量。在该向量中的值是**[left,bottom,width,height]**。其中**[left,bottom]**是该对象相对于它的父对象的的左下角的位置，而**[width,height]**是该对象的宽度和高度。参阅图 20.2。

图 20.2　Position 属性示意图

这些位置向量中的值的单位是由该对象的单位属性（**Units**）所指定的。例如：

```
» get(gcf, 'Position' )
ans=
        360      544      560      420
» get(gcf, 'Units' )
ans=
    pixels
```

表明了当前图形对象的左下角，相对于屏幕左下角的位置是：向右 360 个象素，向上 544 个象素；且图形对象的宽度为 560 个象素点，高度为 420 个象素点。**要注意，位置向量给出的是图形本身的可画区域，它并不包括该窗口的边界、滚动条或标题条。**

'Units' 属性的缺省值是象素，但它也可以是英寸、厘米、点或归一化坐标。象素代表了屏幕象素，即在屏幕上可表示出来的最小的矩形对象。例如，一个分辨率设置为 800×600 的计算机显示区宽为 800 个象素，高为 600 个象素。点是一种打印设置标准，每一点等于 1/72 英寸。归一化坐标是在 0 到 1 范围内。在归一化坐标中，屏幕的左下角在**[0　0]**，右上角在**[1.0　1.0]**。至于英寸和厘米是不言而喻的。

为了形象说明不同的 **Units** 属性值，重新考虑上面的例子。

```
» set(gcf, 'Units' , 'inches' );
» get(gcf, 'Position' )
ans =
    3.7764    5.7203    5.8907    4.4245

» set(gcf, 'Units' , 'centimeters' )
» get(gcf, 'Position' )
ans =
    9.5847   14.5185   14.9511   11.2297

» set(gcf, 'Units' , 'normalized' )
» get(gcf, 'Position' )
ans =
    271.9001   411.8597   424.1339   318.5655
» set(gcf, 'Units' , 'normalized' )
» get(gcf, 'Position' )
ans =
    0.2805    0.5303    0.4375    0.4102
```

对于特定的显示器，这些值代表了相对于计算机屏幕的相同位置。

坐标轴对象的位置也是四元素向量，具有同样形式**[left,bottom,width,height]**。但它指定的是相对于它的父对象图形左下角的位置。一般来说，一个子对象的‘**Position**‘属性的值是相对于其父对象的位置。

为了更好描述，计算机屏幕或根对象的位置属性不叫做‘**Position**‘，而叫做‘**ScreenSize**‘。这时，**[left,bottom]**总是**[0 0]**，而**[width,height]**是计算机屏幕的尺寸，它的单位由根对象的‘**Units**‘属性指定。

20.8 图形打印

除了将图形放置在计算机屏幕上，MATLAB 还提供了控制打印页上图形位置的属性，以及指定打印纸本身特性的属性。例如，MATLAB 的 **orient** 命令使用函数 **get** 和 **set** 来改变打印纸属性。函数 **orient** 的帮助如下：

» help orient

ORIENT Hardcopy paper orientation.

ORIENT LANDSCAPE causes subsequent PRINT operation from the current figure window to generate output in full-page landscape orientation on the paper.

ORIENT PORTRAIT returns to the default PORTRAIT orientation with the figure window occupying a rectangle with aspect ratio 4/3 in the middle of the page.

ORIENT TALL causes the figure window to map to the whole page in portrait orientation.

ORIENT ,by itself,returns a string containing the current paper orientation,either PORTRAIT,LANDSCAPE or TALL.

ORIENT is an M-file that sets the PaperOrientation and PaperPosition properties of the current figure window.

帮助信息：

ORIENT 硬拷贝纸张反向

ORIENT LANDSCAPE 使得以后从该图形窗口的打印操作时，图形在打印纸上的设置为景象 方向的全幅方式。

ORIENT PORTRAIT 使得以后从该图形窗口的打印操作时，图形在打印纸的中间占据一个肖 像方向 4/3 长宽比的矩形。

ORIENT TALL 使得以后从该图形窗口的打印操作时，图形在在打印纸上的设置为肖像方向 的全幅方式。

ORIENT 不加参数时，返回一个包含当前纸张设置的字符串，值为 LANDSCAPE，PORTRAIT 或 TALL。

ORIENT 是一个 M 文件，它设置当前图形窗口对象中的 PaperOrientation 和 PaperPosition 属性。

参阅 PRINT

影响打印输出的图形属性在表 20.1 中：

表 20.1

图形的打印纸属性	
PaperUnits	**[{inches}|centimeters|normalized|points]**
PaperOrientati-on	**[{portrait}|landscape]**
PaperPosition	位置向量，形式为**[left,bottom,width,height]**。**[left,bottom]**代表了从打印页左下角的偏移，**[width,height]**是图形尺寸
PaperSize	包含纸张尺寸（**[8.5　　11]**）的两元素向量
PaperType	**[{usletter}|uslegal1|a3|a4letter|a5|b4|tabloid]**

‘**PaperPosition**’ 和 ‘**PaperSize**’ 属性的返回值的单位由 ‘**PaperUnits**’ 属性指定。与改变图形对象 ‘**Position**’ 属性以改变屏幕上图形窗口的大小和位置一样，改变 ‘**PaperPosition**’ 属性可以改变图形在打印页上的大小和位置。

考虑下面的例子，说明如何使用纸张属性。

» set(gcf, ‘PaperType’ , ‘a4letter’)

将当前图形打印所用纸张设为 ‘ **a4letter** ‘ 。

» set(gcf, ‘PaperOrientation’ , ‘landscape’)

将当前图形窗口的图的方向设为 ‘ **landscape** ‘ 。

象其它图形属性一样，上面的纸张属性用于单个图形。关于修改所有图形属性在下节“缺省属性”中进行讨论。

20. 9 缺省属性

MATLAB 在建立对象时把缺省属性赋给各对象。如果想不采用这些缺省值，那么，必须使用句柄图形工具对它们进行设置。当每次都要改变同一属性时，MATLAB 允许设置用户自己的缺省属性。MATLAB 让用户改变对象层次结构中任意一点上的单个对象或对象类型的缺省属性。

可以使用一个特殊性质名字符串来设置缺省值，该字符串以 ‘**Default**’ 开头，跟之以对象类型和属性名。使用 **set** 命令中的句柄，确定对象父—子等级图中的点，在该点使用缺省值。例如：

» set(0, ‘DefaultFigureColor’ ,[.5　　.5　　.5])

将所有的**新**图形对象设为适中的灰色，而不是 MATLAB 缺省的黑色。该属性值应用于根对象（它的句柄总是 0），所以所有新图形会有一个灰色的背景。

下面是另外一些可改变缺省值的例子。

```
» set(0, 'DefaultAxesFontSize' ,14)        %    larger axes fonts - all figures
» set(gcf, 'DefaultAxesLineWidth' ,2)    %    thick axes lines - this figure
» set(gcf, 'DefaultAxesColor' , 'y' )        %    yellow X-axis lines and labels
» set(gcf, 'DefaultAxesGrid' , 'on' )        %    Y axis grid lines - this figure
» set(0, 'DefaultAxesBox' , 'on' )            %    enclose axes - all figures
» set(gcf, 'DefaultLineLineStyle' , ':' )    %    dotted linestyle - these axes
```

当应用已存在对象工作时，使用后把它们恢复到初始的状态是一个很好的想法。**如果想在一段例程中改变对象的缺省属性，那么保存原来的设置并在激活例程时将它们恢复**。例如，考虑下面一段函数：

```
oldunits=get(0, 'DefaultFigureUnits' );
set(0, 'DefaultFigureUnits' , 'normalized' );
      <MATLAB statements>
set(0, 'DefaultFigureUnits',oldunits);
return
```

如果在所有的时刻用自己的缺省值来设定 MATLAB，那么只要在 **startup.m** 文件里包括进所需的 **set** 命令就可以了。例如，如果在所有的轴上想要缺省的网格和坐标轴框，并且经常在 A4 纸上打印，就把下面这些行加到 **start.m** 文件中即可。

```
set(0, 'DefaultAxesXGrid' , 'on' )
set(0, 'DefaultAxesYGrid' , 'on' )
set(0, 'DefaultAxesZGrid' , 'on' )
set(0, 'DefaultAxesBox' , 'on' )
set(0, 'DefultFigurePaperType' , 'a4paper' )
```

关于脚本 M 文件 start.m 的更详细的信息，可参阅第二章。

有三个特殊的属性值字符串 '**remove**' 、 '**factory**' 和 '**default**' ，它们逆转、取消或获得用户自定义缺省属性。如果改变了一个缺省属性，可以使用 '**remove**' 逆转这种变化，把它重新设为初始的缺省值。使用 'remove' 说明如下：

```
» set(0, 'DefaultFigureColor' ,[.5 .5 .5])     %    set a new default
» set(0, 'DefaultFigureColor' , 'remove' )    %    return to MATLAB defaults
```

对一特殊对象，为了暂时取消缺省值并用最初的 MATLAB 缺省值，就用特殊的属性值 '**factory**' 。例如：

```
» set(0, 'DefaultFigureColor' ,[.5 .5 .5])    %    set a new user default
» figure( 'Color' , 'factory' )    %    create a new figure using the MATLAB default
```

第三个特殊的属性值字符串是 '**default**' 。这个属性值迫使 MATLAB 搜索对象层次结构，直到查到所需属性的一个缺省值。如果找到，它就使用该缺省值。如果查到根对象，没有找

到用户定义的缺省值，MATLAB 就使用 factory 缺省值。在用不同的属性值创建一个对象后，要把对象缺省设成缺省属性值时，这个概念是很有用的。为了弄清 '**default**' 的使用，考虑下面的例子。

```
» set(0, 'DefaultFigureColor' , 'r' )        %    set the default at the root level
» set(gcf, 'DefaultFigureColor' , 'g' )    %    current figure level default
» set(gca, 'DefaultFigureColor' , 'b' )    %    current axes level default
» Hl_rand=plot(rand(size([1:10])))        %    plot a yellow line
» set(H1_rand, 'Color' , 'default' )        %    the line become blue
» set(gca, 'DefaultFigureColor' , 'remove' )    %    the axes level default is removed
» set(Hl_rand, 'Color' , 'default' )         %    the line become green
» close(gcf)    %    close the window
» Hl_rand=plot(rand(size([1:10])))      %    plot a yellow line in a new window
» set(Hl_rand, 'Color' , 'default' )        %    the line becomes red
```

注意到 **plot** 命令并不对线的颜色设为线条对象的缺省值。如颜色参量未指定，**plot** 命令使用坐标轴 '**ColorOrder**' 属性来指定它所产生的每条线的颜色。

20.10 非文件式属性

用函数 **get** 和 **set** 对每一个对象列出的属性是文件式属性。也有由 MATLAB 开发者所用的非文件式属性。其中一些可以被设置，但另外一些是只读的。

每一对象类型的一个有用的非文件式属性是 '**Tag**' 属性。这个属性用用户自定义的文本字符串来标志一个对象时有用。例如：

```
» set(gca, 'Tag' , 'My    axes' )
```

就把字符串 '**My   axes**' 加到当前图形的当前坐标轴。这个字符串不在图形或坐标轴中上显示出来，但可以查询 '**Tag**' 属性来辨别对象。例如，有许多个坐标轴，可以通过

```
» Ha_myaxes=findobj(0, 'Tag' , 'My    Axes' );
```

来寻找上面的坐标轴对象的句柄。象在下一章讨论的 '**UserData**' 属性一样， '**Tag**' 属性备作专门使用。没有任何 MATLAB 函数和 M 文件改变或对这些属性所含值作出假设。然而，如在下章要讨论的，有一些用户提供的 M 文件和几个精通 *MATLAB* 工具箱函数使用 '**UserData**' 属性来存储临时数据。

由于一些非文件式属性是故意作成非文件式的，所以在使用时必须**非常**小心。它们有时比文件式属性脆弱，并且**常常**引起变化。在以后的 MATLAB 版本中，非文件式属性也许会仍保持或消失，或者改变功能，甚至会成为文件式属性。

在 MATLAB 5.0 中应变成文件式属性的非文件式属性列在表 20.2 中。

表 20.2

属性	对象

'TerminalHideGraphCommand'	根
'TerminalDimensions'	根
'TerminalShowGraphCommand'	根
'Tag'	所有对象
'Layer'	坐标轴
'PaletteModel'	曲面，补片

20.11 M 文件例子

精通 MATLAB 工具箱含有许多实用函数，它们可以验证本章的许多概念。这些函数的基本部分已经在二维和三维图形这些章阐述过了。有了前面对句柄图形的讨论，我们现在可以更彻底地讨论这些函数。

最简单的精通 MATLAB 工具箱的函数之一提出了一个共同的问题。MATLAB 函数 **gcf** 返回当前图形的句柄。但是，它有一个副作用。如果图形不存在，**gcf** 就创建一个，并返回它的句柄。如果想寻找一个图形是否存在于头一个位置，要是没有，又不得不创建，怎么办？函数 **mmgcf** 正好实现由其内容所描述的工作。

```
function HF=mmgcf
%MMGCF Get Current Figure if it Exists.
%    MMGCF returns the handle of the current figure if it exists.
%    If no current figure exists,MMGCF returns an empty handle.
%
%    Note that the function GCF is different.It creates a figure and returns its handle if it does
not  %    exist.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

Hf=get(0, 'Children' );    %    check for figure children
if isempty(Hf)
      return
else
      Hf=get(0, 'CurrentFigure' );
end
```

函数 **mmgcf** 首先检查根对象的子对象的图形是否存在，如至少有一个图形对象时，根对象的 '**CurrentFigure**' 属性就返回当前的图形。

函数 **mmgca** 为坐标轴对象执行同样的功能，如同在它的 M 文件内所描述的那样。

```
function Ha=mmgca
%MMGCA Get Current Axes if it exists.
%    MMGCA returns the handle of the current axes if it exists.
%    If no current axes exists,MMGCA returns an empty handle.
%
```

```
%    Note that the function GCA is diffent.It create a figure and an axes and returns the axes
%    handle if they do not exist.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

Ha=findobj(0, 'Type' , 'axes' );
if isempty(Ha)
      return
else
      Ha=get(get(0, 'CurrentFigfure' ), 'CurrentAxes' );
end
```

由于函数 **gco** 已经表现出当对象不存在时返回空矩阵的行为特性，就不需要函数 **mmgco** 了。

在精通 MATLAB 工具箱中的另一个函数是 **mmzap**，在二维图形那一章里已作过介绍。如下 M 文件中所示，它使用 **mmgcf** 作错误检查，与 **findobj** 和 **get** 一起删除一个指定的图形。

```
function mmzap(arg)
%MMZAP Delete graphics object using mouse.
%    MMZAP waits for a mouse click on an object in a figure window and deletes the object.
%    MMZAP or MMZAP text erases text objects.
%    MMZAP axes erases axes objects.
%    MMZAP line erases line objects.
%    MMZAP surf erases surface objects.
%    MMZAP patch erases patch objects.
%
%    Clicking on an object other than the selected type or striking a key on the keyboard
aborts    %    the command.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

if nargin<1,arg= 'text' ;end

Hf=mmgcf;
if isempty(Hf),error( 'No Figure Available.' ),end
if length(findobj(0, 'Type' , 'figure' ))==1
      figure(Hf)    %    bring only figure forword
end
key=waitforbuttonpress;    %    pause untill user takes some action
if key    %    key on keyboard pressed
      return    %    take no action
else    %    object selected
      object=gco    %    get object selected by buttonpress
      type=get(object, 'Type' );
```

```
        if all(type(1:4)==arg(1:4))    %    delete only if    'Type'    is correct
            delete(object)
        end
    end
```

在编写句柄图形函数的 M 文件时，函数 **mmzap** 描述了一种很有用的技术。它利用函数 **waitforbuttonpress** 和 **gco** 的结合用鼠标来获取所选定对象的句柄。 **waifortbuttonpress** 是一个 MATLAB 内置函数，它的功能是等待鼠标点击或按键。它的帮助文本如下：

```
» help waitforbuttonpress

WAITFOR BUTTONPRESS Wait for key/buttonpress over figure.
    T= WAITFOR BUTTONPRESS stops program execution untill a key or mouse button is
    pressed over a figure window.Returns 0 when terminated by a mouse buttonpress,or 1
    when terminated by a keypress.Additional information about the terminating event is
        available from the current figure.

    See also GINPUT,GCF.
```

帮助信息：

WAITFORBUTTONPRESS 等待一个鼠标/按钮对图形按下。

T=WAITFORBUTTONPRESS 停止程序的执行，直到鼠标按钮或键在一个图形窗口按下。当鼠标按钮按下时返回 0；当键按下时返回 1。其它的结束事件的信息可从当前的图形中窗口　　获取。

参阅 GINPUT 和 GCF。

鼠标按钮在鼠标指针指的图形上按下后，函数 **gco** 返回所点中对象的句柄。然后，该句柄可用来操作选中的对象。在精通 MATLAB 工具箱中，用这种简单的选择技术的函数还有 **mmline** 和 **mmaxes**。其中， **mmline** 的 M 文件描述如下：

```
function mmline(arg1,arg2, arg3, arg4, arg5, arg6)
%MMLINE Set Line Properties Using Mouse
%   MMLINE waites for a mouse click on a line then applies the desired properties to the
%   selected line.
%   Properties are given in parts,e.g.,MMLINE Name value...
%   Properties:
%   NAME            VALUE{default}
%   color           [Y m c r g b w k] or an RGB in quotes: '[r g b]'
%   style                 [-   --   ;   -.]
%   mark            [o   +   .   *   X]
%   width           points for linewidth {0.5}
%   size            points for marker size {6}
```

```
%   zap                 (n.a.)      delete selected line
%   Examples:
%   MMLINE color r width 2      sets color to red and width to 2 points
%   MMLINE mark + size 8        sets marker type to + and size to 8 points
%
%   Clicking on an object other than a line,or striking a key on the keyboard aborts the
%   command.

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

Hf=mmgcf;
if isempty(Hf),error( 'No Figure Avaiable.' ),end
if length(get(0, 'Children' ))==1
    figure(Hf)   %   bring only figure forword
end
key=waitforbuttonpress;
if key   %   key on keyboard pressed
    return
else   %   object selected
    Hl=gco
    if strcmp(get(Hl, 'Type' ), 'line' )   %   line object selected
        for i=1:2:max(nargin-1,1)
            Name=eval(sprintf( 'arg%.0f ' ,i),[]);   get Name argument
            if strcmp(Name, 'zap' )
                delete(Hl),return
            end
            value=eval(sprintf( 'arg%.0f ' ,i+1),[]);   %   get value
            if strcmp(Name, 'color' )
                set(Hl, 'Color' ,value)
            elseif strcmp(Name, 'style' )
                set(Hl, 'Linestyle' ,value)
            elseif strcmp(Name, 'mark' )
                set(Hl, 'Linestyle' ,value)
            elseif strcmp(Name, 'width' )
                value=abs(eval(value))
                set(Hl, 'LineWidth' ,value)
            elseif strcmp(Name, 'size' )
                value=abs(eval(value))
                set(Hl, 'MarkerSize' ,value)
            else
                disp([ 'Unknown Property Name: ' Name'])
            end
        end
    end
```

```
end
```

精通 MATLAB 工具箱中的函数 **mmpaper** 以简单的方式阐述了对纸张属性的使用。如下所示，函数 **mmpaper** 设置当前图形的纸张属性，并将所有以后的图形设成缺省值。函数 **mmpage** 在下一章讨论，它是一个 **mmpaper** 友函数。**mmpage** 建立一个图形用户界面，设定图形在打印页上的位置。

```
function mmpaper(arg1,arg2,arg3,arg4,arg5,arg6)
%MMPAPER Set Default Paper Properties.
%   MMPAPER Name value...
%    sets default paper properties for the current figure and succeeding figures based on Name    %   value pairs.
%   Properties:
%   NAME                VALUE     {default}
%   Units               [{inches},centimeters,points,normal]
%   orient              [{portrait},landscape]
%   type                    [{usletter},uslegal,a3,a4letter,a5,b4,tabloid]
%
%   Examples:
%   MMPAPER Units inch orient landscape
%   MMPAPER type tabloid
%
%   MMPAPER with no arguments returns the current paper defaults.

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

Hf=mmgcf;
flag=0;
if isempty(Hf)
    flag=1;
    Hf=figure( 'Visible' , 'off ' );
end
if nargin
    for i=1:2:max(nargin-1,1)
        Name=eval(sprintf( 'arg%.0f ' ,i)0,[]);   %   get Name argument
        value=eval(sprintf( 'arg%.0f ' ,i+1)0,[]);   %   get Name argument
        if Name(1)== 'o'
            set(0, 'DefaultFigurePaperOrientation' ,value)
            set(Hf, 'PaperOrientation' ,value)
        elseif Name(1)== 't'
            set(0, 'DefaultFigurePaperType' ,value)
            set(Hf, 'PaperType' ,value)
        elseif Name(1)== 'u'
            set(0, 'DefaultFigurePaperUnits' ,value)
```

```
            set(Hf, 'PaperUnits' ,value)
        else
            disp([ 'Unknown Property Name:' Name])
        ens
    end
end
```

当把对象放在一个特定的位置时，有时在象素和归一化坐标之间进行转化是很有用的。在精通 MATLAB 工具箱中有两个函数进行这种转换。第一个是 **mmpx2n**，它将象素转化为归一化坐标；第二个是 **mmn2px**，它进行相反的转换。这些函数演示了如何以所需的一组单位获取 '**Position**' 属性值。首先，把对象的当前 '**Units**' 属性保存起来；然后，将 '**Units**' 属性设成所需的值并获取所需的 '**Position**' 属性值；最后，将 '**Units**' 的值恢复为初始值。**mmpx2n** 的 M 文件描述如下：

```
function Y=mmpx2n(X,Hf)
%MMPX2N Pixel to Normalized Coordinate Transformation.
%   MMPX2N(X) converts the Position vector   X   from pixel coordinates to normalized
%   coordinates w.r.t.the computer screen.
%
%    MMPX2N converts the Position vector X from pixel coordinate to narmalized coordinates
%   w.r.t.the figure windoiw having handle H.
%
%   X=[left   bottom   width   height] or X=[width   height]

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

msg= 'Input is not a pixel Position vector.' ;
lx=length(X);

sz= 'Position' ;
if nargin==1,Hf=0;sz= 'ScreenSize' ;end

if any(X<1)| (lx~=4&lx~=2)
    error(msg)
end
if lx==2,X=[1   1   X(:)' ];end   %   [width   height] input format
u=get(Hf, 'Units' );   %   get Units
set (Hf, 'Units' , 'pixels' );   %   set Units to pixels
s=get(Hf,sz);
Y=(X-1)./([s(3:4)]-1);   %   convert
set(Hf, 'Units' ,u);   %   reset Units
if any(Y>1)
    error(msg)
```

```
end
if lx==2,Y=Y(3:4);end    %    [width height] output format
```

精通 MATLAB 工具箱中的两个函数 **mmcont2** 和 **mmcont3** 都用用户指定的颜色映象画等值线图。每一个函数分析输入参量并建立一个字符串，它包含了颜色的说明。一旦设置了字符串，就设置了当前坐标轴的 '**ColorOrder**' 属性；最后，它们分别调用具有合适的参量的函数 **contour** 和 **contour3** 来画出图形。函数 **mmcont2** 的 M 文件描述如下：

```
function[cs,h]=mmcont2(arg1,arg2,arg3,arg4,arg5)
%MMCONT2 2-D contour plot using a colormap.
%    MMCONT2(X,Y,Z,N,C) plots N contours of Z in 2-D using the color
%    specified in C.C can be a linestyle and color as used in plot,
%    e.g., 'r-' ,orC can be the string Name of a colormap. X and Y
%    define the axis limits.
%    If not given default argument values are: N=10,C= 'hot' ,
%    X and Y =row and column indices of Z. Examples:
%    MMCONT2(Z)                  10 lines with hot colormap
%    MMCONT2(Z,20)                    20 lines with hot colormap
%    MMCONT2(Z, 'copper' )       10 lines with copper colormap
%    MMCONT2(Z,20, 'gray' )           20 lines with gray colormap
%    MMCONT2(X,Y,Z, 'jet' )      10 lines with jet colormap
%    MMCONT2(Z, 'c-' )           10 dashed lines in cyan
%    MMCONT2(X,Y,Z,25, 'pink' )  25 lines in pink colormap
%
%    CS=MMCONT2(…) returns the contour matrix CS as described in
%    CONTOURC.
%    [CS,H]=MMCONT2(…) returns a column vector H of handles to
%    line objects.

%    Copyright (c) 1996 by Prentice-Hall,Inc.
n=10;c= 'hot' ;    %    default values
nargs=nargin;cflag=1;
if nargin<1,error( 'Not enough input arguments.' ), end

for i=2:nargin    %    check input arguments for N and C
      argi=eval(sprintf( 'arg%.0f ' ,i));
      if ~isstr(argi)&length(argi)==1    %    must be N, grab it
            n=argi;
            nargs=i;    %    # args to pass to contour2
      elseif isstr(argi)    %    must be C
            if exist(argi)==2    %    is colormap,so grab it
                  c=argi;
                  nargs=i-1;
            else    %    is single color/linestyle
```

```
                cflag=0;
                nargs=i;
            end
        end
    end
    if cflag    %    a colormap has been chosen
        clf    %    clear figure
        view(2)    %    make it 2-D
        hold on    %    hold it
        mapstr=sprintf([c '(%.0f)' ],n);
        set(gca,'ColorOrder',eval(mapstr));
    end
    evalstr= '[CS,H]=contour(' ;
    for i=1:nargs
        evalstr=[evalstr sprintf( 'arg%.0f ' ,i)    ',' ];
    end
    lstr=length (evalstr);
    evalstr(lstr:lstr+1)= ');' ;
    eval(evalstr)
    hold off
    if nargout==1,          cs=CS;
        elseif nargout==2,      cs=CS;h=H;
    end
```

这里要讨论的最后一个精通 MATLAB 工具箱函数是 **mmtile**。就象在二维函数那一章里所描述的一样，该函数在计算机屏幕上将 4 个已存在的图形按平铺模式排列起来。函数 **mmtile.m** 的内容如下所示：

```
function h=mmtile(n)
%    MMTILE Tile Figure Windows.
%    MMTILE with no arguments, tiles the current figure windows
%    and brings them to the foreground.
%    Figure size is adjusted so that 4 figure windows fit on the screen.
%    Figures are arranged in a clockwise fashion starting in the
%    upper-left corner of the display
%
%    MMTILE(N) makes tile N the current figure if it exists.
%    Otherwise, the next tile is created for subsequent plotting
%
%    Tiled figure windows are titled TILE #1,TILE #2, TILE #3, TILE #4.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

HT=40;          %tile height fudge in pixels
```

```
WD=20;      %  tile width fudge
%  adjust the above as necessary to eliminate tile overlaps
%  bigger fudge numbers increase gaps between tiles

Hf=sort(get(0,'Children'));  %  get handles of current figures
nHf=length(Hf);
set(0,'Units','Pixels')  %  set screen dimensions to pixels
sz=get(0,'Screensize');  %  get screen size in pixels
tsz=0.9*sz(3:4);  %  defaut tile area is almost whole monitor
if sz(4)>sz(3),  %  if portrait monitor
    tsz(2)=.75*tsz(1);  %  take a landscape chunk
end
tsz=min(tsz,[920 690]);  %  hold tile area on large screens to 920 by 690
t1(1,1)=sz(3)-tsz(1)+1;  %  left side of left tiles
t1(2,1)=t1(1,1)+tsz(1)/2;  %  left side of right tiles
tb(1,1)=sz(4)-tsz(2)+1;  %  bottom of bottom tiles
tb(2,1)=tb(1,1)+tsz(2)/2;  %  bottom of top tiles

tpos=zeros(4);  %  matrix holding tile Position vectors
tpos(:,1)=t1([1 2 2 1],1);  %  left sides
tpos(:,2)=t1([2 2 1 1],1);  %  bottoms
tpos(:,3)=(tsz(1)/2-WD)*ones(4,1);  %  widths
tpos(:,4)=(tsz(2)/2-HT)*ones(4,1);  %  heights
tpos=fix(tpos);  %  make sure pixel Positions are integers
if nargin==0  %  tile figures as needed
    for i=1:min(nHf,4)
        set(Hf(i),'Units','pixels')
        if any(get(Hf(i),'Position')~=tpos(i,:))
            set(Hf(i),'Position',tpos(i,:),…
              'NumberTitle','off',…
              'Name',['TILE #' int2str(i)])
        end
        figure(Hf(i))
    end
else  %  go to tile N or create it
    n=rem(abs(n)-1,4)+1;  %  N must be between 1 and 4
    if n<=nHf  %  tile N exists,make it current
        figure(Hf(n))
    else  %  tile N does not exist,create next one
        n=nHf+1;
        figure('Position',tpos(n,:),…
          'NumberTitle','off',…
          'Name',['TILE #' int2str(n)])
    end
```

end

如上面所描述的，函数 **mmtile** 从根对象得到所有的图形对象的句柄和屏幕尺寸，为该图形计算新的位置和尺寸，然后设置每个图形的 '**Units**'，'**Position**'，'**Number**' 和 '**Name**' 属性。它具有安置和缩放图形的效能，并在每个窗口标题中，改变名字字符串。**HT** 和 **WT** 给出的号码与计算机平台有关。它们对图形的 '**Position**' 描述窗口内的可画区域而不是外部尺寸有补偿作用。

20.12 属性名和属性值

下面各表中列出了 MATLAB 4.2 版本中的属性名和属性值。有一个星号*的属性是非文件化的。用大括号{}括起来的属性值是缺省值。

表 20.3

根对象属性	
BlackAndWhite	自动硬件检测标志
on:	认为显示是单色的，不检测；
{off}:	检测显示类型
***VlaxkOutUnusedSlots**	值为**[{no}|yes]**
***CaptureMap**	
CaptureMatrix	由 **CaptureRect** 矩形所包围的区域内图象数据的只读矩阵，使用 **image** 来显示
CaptureRect	捕捉矩形的尺寸和位置，是一个 4 元素的向量 **[left,bottom,width,height]**，单位由 **Units** 属性指定。
***CaseSen**	值为**[{on}|off]**
CurrentFigure	当前图形的句柄。
Diary	会话记录
on:	将所有的键盘输入和大部分输出拷贝到文件中
{off}:	不将输入和输出存入文件
DiaryFile	一个包含 **diary** 文件名的字符串，缺省的文件名为 **diary**
Echo	脚本响应模式
on:	在文件执行时，显示脚本文件的每一行
{off}:	除非指定 **echo on**，否则不响应
Format	数字显示的格式
{short}:	5 位的定点格式
shortE:	5 位的浮点格式
long:	15 位换算过的定点格式
longE:	15 位的浮点格式
hex:	16 进制格式
bank:	美元和分的定点格式
+:	显示+和-符号
rat:	用整数比率逼近

FormatSpacing 输出间隔

{loose}: 显示附加行的输入

compact: 取消附加行的输入

***HideUndocumented** 控制非文件式属性的显示

no: 显示非文件式属性

{yes}: 不显示非文件式属性

PointerLocation 相对于屏幕左下角指针位置的只读向量**[left,bottom]**或**[X,Y]**，单位由 **Units** 属性指定

PointerWindow 含有鼠标指针的图形句柄，如果不在图形窗口内，值为 0。

ScreenDepth 整数，指定以比特为单位的屏幕颜色深度，比如：1 代表单色，8 代表 256 色或灰度

ScreenSize 位置向量**[left,bottom,width,height]**，其中**[left,bottom]**常为**[0 0]**，**[width,height]**是屏幕尺寸，单位由 **Units** 属性指定

***StatusTable** 向量

***TerminalHideGraphCommand** 文本串

TerminalOneWindow 由终端图形驱动器使用

no: 终端有多窗口

yes: 终端只有一个窗口

***TerminalDimensions** 终端尺寸向量**[width,height]**

TerminalProtocal 启动时终端类型设置，然后为只读

none: 非终端模式，不连到 X 服务器

X: 找到 X 显示服务器，X Windows 模式

tek401x: Tektronix 4010/4014 仿真模式

tek410x: Tektronix 4100/4105 仿真模式

***TerminalShowGraphCommand** 文本串

Units Position 属性值的度量单位

inches: 英寸

centimeters: 厘米

normalized: 归一化坐标，屏幕的左下角映射到[0 0]，右上角映射到[1 1]

points: 排字机的点，等于 1/72 英寸

{pixels}: 屏幕象素，计算机屏幕分辨率的最小单位

***UsageTable** 向量

ButtonDowFcn MATLAB 回调字符串，当对象被选择时传给函数 **eval**，初始值是一空矩阵

Children 所有图形对象句柄的只读向量

Clipping 数据限幅模式

{on}: 对根对象无效果

off: 对根对象无效果

Interruptible **ButtonDowFcn** 回调字符串的可中断性

{no}: 不能被其它回调中断

yes: 可以被其它回调中断

Parent	父对象的句柄，常为空矩阵
***Selected**	值为**[on|off]**
***Tag**	文本串
Type	只读的对象辨识字符串，常是 **root**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	对象可视性
{on}:	对根对象无效果
off:	对根对象无效果

表 20.4

图形对象属性

BackingStore	为了快速重画，存储图形窗口的拷贝
{on}:	当一个图原来被覆盖的一部分显露时，拷贝备份，刷新窗口较快，但需要较多的内存
off:	重画图形以前被覆盖的部分，刷新较慢，但节省内存
***CapterMap**	矩阵
***Client**	矩阵
Color	图形背景色，一个 3 元素的 **RGB** 向量或 MATLAB 预定的颜色名，缺省的颜色是**黑色**
Colormap	m×3 的 **RGB** 向量矩阵，参阅函数 **colormap**
***Colortable**	矩阵，也许包含一份系统颜色映象的拷贝
CurrentAxes	图形的当前坐标轴的句柄
CurrentCharacter	当鼠标指针在图形窗口中，键盘上最新按下的字符键
CurrentMenu	最近被选择的菜单项的句柄
CurrentObject	图形内，最近被选择的对象的句柄，即由函数 **gco** 返回的句柄
CurrentPoint	一个位置向量**[left,bottom]**或图形窗口的点的**[X,Y]**，该处是鼠标指针最近一次按下或释放时所在的位置。
FixedColors	n×3 的 **RGB** 向量矩阵，它使用系统查色表中的槽来定义颜色，初始确定的颜色是 **black** 和 **white**
***Flint**	
InvertHardcopy	改变图形元素的颜色以打印
{on}:	将图形的背景色改为白色，而线条、文本和坐标轴改为黑色以打印
off:	打印的输出颜色和显示的颜色完全一致
KeyPressFcn	当鼠标指针处在图形内，按下键，传递给函数 **eval** 的 MATLAB 回调字符串
MenuBar	将 MATLAB 菜单在图形窗口的顶部显示，或在某些系统中在屏幕的顶部显示
{figure}:	显示缺省的 MATLAB 菜单
none:	不显示缺省的 MATLAB 菜单
MinColormap	颜色表输入项使用的最小数目。它影响系统颜色表。如设置太低，会使未选中的图形以伪彩色显示。
Name	图形框架窗口的标题（**不是**坐标轴的标题）。缺省时是

空串，如设为 string（字符串），窗口标题变为：**Figure No.n：string**

NextPlot 决定新图作图行为

new： 画前建立一个新的图形窗口

{add}： 在当前的图形中加上新的对象

replace： 在画图前，将除**位置**属性外的所有图形对象属性重新设置为缺省值，并删除所有子对象

NumberTitle 在图形标题中加上图形编号

{on}： 如果 **Name** 属性值被设为 string，窗口标题是 **Figure No.N：string**

off： 窗口标题仅仅是 **Name** 属性字符串

PaperUnits **纸张**属性的度量单位

{inches}： 英寸

centimeters： 厘米

normalized： 归一化坐标

points： 点，每一点为 1/72 英寸

PaperOrientation 打印时的纸张方向

{portrait}： 肖像方向，最长页面尺寸是垂直方向

landscape： 景象方向，最长页面尺寸是水平方向

PaperPosition 代表打印页面上图形位置的向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于打印页面图形左下角的位置，**[width,height]**是打印图形的尺寸，单位由 **PaperUnits** 属性指定

PaperSize 向量**[width,height]**代表了用于打印的纸张尺寸，单位由 **PaperUnits** 属性指定，缺省的纸张大小为**[8.5 11]**

PaperType 打印图形纸张的类型。当 **PaperUnits** 设定为归一化坐标时，MATLAB 使用 **PaperType** 来按比例调整图形的大小

{usletter}： 标准的美国信纸

uslegal1： 标准的美国法定纸张

a3： 欧洲 A3 纸

a4letter： 欧洲 A4 信纸

a5： 欧洲 A5 纸

b4： 欧洲 B4 纸

tabloid： 标准的美国报纸

Pointer 鼠标指针形状

crosshair： 十字形指针

{arrow}： 箭头

watch： 钟表指针

top1： 指向左上方的箭头

topr： 指向右上方的箭头

bot1： 指向左下方的箭头

botr： 指向右下方的箭头

circle： 圆

cross：双线十字形

fleur：4 头箭形或指南针形

Position　位置向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于计算机屏幕的左下角窗口左下角的位置，**[width,height]**是屏幕尺寸，单位由 **Units** 属性指定

Resize　允许不允许交互图形重新定尺寸

{on}：窗口可以用鼠标来重新定尺寸

off：窗口不能用鼠标来重新定尺寸

ResizeFcn　MATLAB 回调字符串，当窗口用鼠标重新定尺寸时传给函数 **eval**

***Scrolled**　值为**[{on}|off]**

SelectionType　一个只读字符串，提供了有关最近一次鼠标按钮选择所使用方式的信息。但实际是哪个键和/或按钮按下与平台有关

{normal}：点击（按下和释放）鼠标左键，或只是鼠标按钮

extended：按下 **shift** 键并进行多个常规（normal）选择；同时击双按钮鼠标的两个按钮；或点击一个三按钮鼠标的中按钮

alt：按下 **Control** 键并进行一次常规选择；或者点击一个双按钮或三按钮鼠标的右按钮

open：双击任何鼠标按钮

Share Colors　共享颜色表的槽

no：不和其它窗口共享颜色表的槽

{yes}：只要可能，重用颜色表中的槽

***StatusTable**　向量

Units　各种位置属性值的度量单位

inches：英寸

centimeters：厘米

normalized：归一化坐标，屏幕的左下角映射到 [0　0]，右上角映射到 [1　1]

points：排字机的点，等于 1/72 英寸

{pixels}：屏幕象素，计算机屏幕分辨率的最小单位

***UsageTable**　向量

WindowButtonDownFcn　当鼠标指针在图形内时，只要按一个鼠标按钮，MATLAB 回调字符串传递给函数 **eval**

WindowButtonMotionFcn　当鼠标指针在图形内时，只要移动一个鼠标按钮，MATLAB 回调字符串传递给函数 **eval**

***WindowID**　长整数

ButtonDownFcn　当图形被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵

Children　图形中所有子对象句柄的只读向量；坐标轴对象，uicontrol 对象和 uimenu 对象

Clipping　数据限幅模式

{on}:	对图形对象不起作用
off:	对图形对象不起作用
Interruptible	指定图形回调字符串是否可中断
{no}:	不能被其它回调中断
yes:	可以被其它回调中断
Parent	图形父对象的句柄，常是 0
***Selected**	值为**[on|off]**
***Tag**	文本串
Type	只读的对象辨识字符串，常是 **figure**
UserDate	用户指定的数据，可以是矩阵、字符串等等
Visible	图形窗口的可视性
{on}:	窗口在屏幕上可视
off:	窗口不可视

表 20.5

坐标轴对象属性

AspectRatio	纵横比向量**[axis_ratio,data_ratio]**，这里 **axis_ratio** 是坐标轴对象的纵横比（宽度/高度），**data_ratio** 是沿着水平轴和垂直轴的数据单位的长度比。如设置，则 MATLAB 建立一个最大的坐标轴，保留这些比率，该最大轴将在 **Position** 定义的矩形内拟合。该属性的缺省值为**[NaN,NaN]**
Box	坐标轴的边框
on:	将坐标轴包在一个框架或立方体内
{off}:	不包坐标轴
CLim	颜色界限向量**[cmin cmax]**，它确定将数据映射到颜色映象。**cmin** 是映射到颜色映象第一个入口项的数据，**cmax** 是映射到最后一项的数据。参阅函数 **cmais**
CLimMode	颜色限制模式
{auto}:	颜色界限映成轴子对象的数据整个范围
manual:	颜色界限并不自动改变。设置 **CLim** 就把 **CLimMode** 值设为人工
Color	坐标轴背景颜色。一个三元素的 **RGB** 向量或一个预定义的颜色名。缺省值是 **none**，它使用图形的背景色
ColorOrder	一个 m×3 **RGB** 值矩阵。如果线条颜色没有用函数 **plot** 和 **plot3** 指定，就用这些颜色。缺省的 **ColorOrder** 为黄，紫红，洋红，红，绿和蓝
CurrentPoint	包含在坐标轴空间内的一对点的坐标矩阵，它定义了从坐标空间前面延伸到后面的一条三维直线。其形式是**[xb yb zb ; xf yf zf]**。单位在 **Units** 属性中指定。点**[xf yf zf]**是鼠标在坐标轴对象中上一次点击的坐标
DrawMode	对象生成次序

{normal}：将对象排序，然后按照当前**视图**从后向前绘制

fast：按已建立的次序绘制对象，不首先排序

***ExpFontAngle** 值为**[{normal}|italic|oblique]**

***ExpFontName** 缺省值为 **Helvetica**

***ExpFontSize** 缺省值为 8 点

***ExpFontStrikeThrough** 值为**[on|{off}]**

***ExpFontUnderline** 值为**[on|{off}]**

***ExpFontWeight** 值为**[light|{normal}|demi|bold]**

FontAngle 坐标轴文本为斜体

{normal}：正常的字体角度

italic：斜体

oblique：某些系统中为斜体

FontName 坐标轴单位标志的字体名。坐标轴上的标志并不改变字体，除非通过设置 **XLabel**，**YLabel** 和 **ZLable** 属性来重新显示它们。缺省的字体为 **Helvetica**

FontSize 坐标轴标志和标题的大小，以点为单位，缺省值为 12 点

***FontStrikeThrough** 值为**[on|{off}]**

***FontUnderline** 值为**[on|{off}]**

FontWeight 坐标轴文本加黑

light：淡字体

{normal}：正常字体

demi：适中或者黑体

bold：黑体

GridLineStyle 格栅线形

-：实线

--：虚线

{:}：点线

-.：点划线

***Layer** 值为**[top|{bottom}]**

LineStyleOrder 指定线形次序的字符串，用在坐标轴上画多条线。例如：’ .- | : | -- | - ‘ 将通过点划线、点线、虚线和实线循环。LineStyleOrder 缺省值为‘-’，即只有实线

LineWidth **X** ，**Y** 和 **Z** 坐标轴的宽度。缺省值为 **0.5** 点

***MinorGridLineStyle** 值为**[- | -- | {:} | -.]**

NextPlot 画新图时要采取的动作

new：在画前建立新的坐标轴

add：把新的对象加到当前坐标轴，参阅 **hold**

{replace}：在画前，删除当前坐标轴和它的子对象，并用新的坐标轴对象来代替它

Position 位置向量**[left,bottom,width,height]**，这里**[left,bottom]** 代表了相对于图形对象左下角的坐标轴左下角位置，**[width,height]**是坐标轴的尺寸，单位由 **Units** 属性指定

TickLength 向量**[2Dlength 3Dlength]**，代表了在二维和三维**视图**中坐标轴刻度标记的长度。该长度是相对于坐标轴的长度。缺省值为[0.01 0.01]，代表二维**视图**坐标轴长度的 1/100，三维**视图**坐标轴长度的 5/1000

TickDir 值为**[{in}|out]**

in： 刻度标记从坐标轴线向内，二维**视图**为缺省值

out： 刻度标记从坐标轴线向外，三维**视图**为缺省值

Title 坐标轴标题文本对象的句柄

Units 位置属性值的度量单位

inches： 英寸

centimeters： 厘米

{normalized}： 归一化坐标，对象左下角映射到**[0 0]**，右上角映射到**[1 1]**

points： 排字机的点，等于 1/72 英寸

pixels： 屏幕象素，计算机屏幕分辨率的最小单位

View 向量**[az el]**，它代表了观察者的视角，以度为单位。**az** 为方位角或视角相对于负 **Y** 轴向右的转角；**el** 为 **X-Y** 平面向上的仰角。详细细节见三维图形这一章

XColor **RGB** 向量或预定的颜色字符串，它指定 **X** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）

XDir **X** 值增加的方向

{normal}： **X** 值从左向右增加

reverse： **X** 值从右向左增加

XForm 一个 4×4 的**视图**转换矩阵。设置 **view** 属性影响 **XForm**

XGrid **X** 轴上的格栅线

on： **X** 轴上每个刻度标记处画格栅线

{off}： 不画格栅线

XLabel **X** 轴标志文本对象的句柄

XLim 向量**[xmin xmax]**，指定 **X** 轴最小和最大值

XLimMode **X** 轴的界限模式

{auto}： 自动计算 **XLim**，包括所有轴子对象的 **XData**

manual： 从 **XLim** 取 **X** 轴界限

***XMinorGrid** 值为**[on|{off}]**

***XMinorTicks** 值为**[on|{off}]**

Xscale **X** 轴换算

{linear}： 线形换算

log： 对数换算

XTick 数据值向量，按此数据值将刻度标记画在 **X** 轴上，将 **XTick** 设为空矩阵就撤消刻度标记

XTickLabels 文本字符串矩阵，用在 **X** 轴上标出刻度标记。如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值

XTickLabelMode **X** 轴刻度标记的标志模式

{auto}： **X** 轴刻度标记张成 **XData**

manual：	从 **XTickLabels** 中取 **X** 轴刻度标记
XTickMode	**X** 轴刻度标记的间隔模式
{auto}：	**X** 轴刻度标记间隔以张成 **XData**
manual：	从 **XTick** 生成 **X** 轴刻度标记
YColor	**RGB** 向量或预定的颜色字符串，它指定 **Y** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）
YDir	**Y** 值增加的方向
{normal}：	**Y** 值从左向右增加
reverse：	**Y** 值从右向左增加
YGrid	**Y** 轴上的格栅线
on：	**Y** 轴上每个刻度标记处画格栅线
{off}：	不画格栅线
YLabel	**Y** 轴标志文本对象的句柄
YLim	向量**[Ymin Ymax]**，指定 **Y** 轴最小和最大值
YLimMode	**Y** 轴的界限模式
{auto}：	自动计算 **YLim**，包括所有轴子对象的 **YData**
manual：	从 **YLim** 取 **Y** 轴界限
***YMinorGrid**	值为**[on|{off}]**
***YMinorTicks**	值为**[on|{off}]**
Yscale	**Y** 轴换算
{linear}：	线形换算
log：	对数换算
YTick	数据值向量，按此数据值将刻度标记画在 **Y** 轴上。将 **YTick** 设为空矩阵就消去刻度标记
YTickLabels	文本字符串矩阵，用在 **Y** 轴上标出刻度标记，如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
YTickLabelMode	**Y** 轴刻度标记的标志模式
{auto}：	**Y** 轴刻度标记张成 **YData**
manual：	从 **YTick**Lab**el**s 中取 **Y** 轴刻度标记
YTickMode	**Y** 轴刻度标记的间隔模式
{auto}：	**Y** 轴刻度标记间隔以张成 **YData**
manual：	从 **YTick** 生成 **Y** 轴刻度标记
ZColor	**RGB** 向量或预定的颜色字符串，它指定 **Z** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）
ZDir	**Z** 值增加的方向
{normal}：	**Z** 值从左向右增加
reverse：	**Z** 值从右向左增加
ZGrid	**Z** 轴上的格栅线
on：	**Z** 轴上每个刻度标记处画格栅线
{off}：	不画格栅线
ZLabel	**Z** 轴标志文本对象的句柄
ZLim	向量**[Zmin Zmax]**，指定 **Z** 轴最小和最大值
ZLimMode	**Z** 轴的界限模式
{auto}：	自动计算 **ZLim**，包括所有轴子对象的 **ZData**

manual：	从 **ZLim** 取 **Z** 轴界限
***ZMinorGrid**	值为**[on|{off}]**
***ZMinorTicks**	值为**[on|{off}]**
Zscale	**Z** 轴换算
{linear}：	线形换算
log：	对数换算
ZTick	数据值向量，按此数据值将刻度标记画在 **Z** 轴上，将 **ZTick** 设为空矩阵就撤消刻度标记
ZTickLabels	文本字符串矩阵，用在 **Z** 轴上标出刻度标记，如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
ZTickLabelMode	**Z** 轴刻度标记的标志模式
{auto}：	**Z** 轴刻度标记张成 **ZData**
manual：	从 **ZTickLabels** 中取 **Z** 轴刻度标记
ZTickMode	**Z** 轴刻度标记的间隔模式
{auto}：	**Z** 轴刻度标记间隔以张成 **ZData**
manual：	从 **ZTick** 生成 **Z** 轴刻度标记
ButtonDownFcn	MATLAB 回调字符串，当坐标轴被选中时，将它传递给函数 **eval**；初始值是一个空矩阵
Children	除了轴标志和标题对象以外，所有子对象句柄的只读向量；包括线、曲面、图象、补片和文本对象
Clipping	数据限幅模式
{on}：	对坐标轴对象不起作用
off：	对坐标轴对象不起作用
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	该回调字符串不能被其它回调所中断
yes：	该回调字符串可以被其它回调所中断
Parent	包含坐标轴对象的图形句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **axes**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	轴线、刻度标记和标志的可视性
{on}：	坐标轴在屏幕上可视
off：	坐标轴不可视

表 20.6

线条对象属性

Color	线条颜色。一个三个元素 **RGB** 向量或 MATLAB 预定的颜色名之一。缺省值是 **white**（白色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
backgrount：	通过在图形背景色中重画线来消除线条。这会破坏被

	消除的线后的对象
	xor：用线下屏幕的颜色执行异或 **OR（XOR）**运算，画
none：	出和消除线条。当画在其它对象上时，可造成不正确的颜色
	当移动或删除线条时该线不会被消除
LineStyle	线形控制
{-}：	画通过所有数据点的实线
--：	画通过所有数据点的虚线
:：	画通过所有数据点的点线
-.：	画通过所有数据点的点划线
+：	用加号作记号，标出所有的数据点
o：	用圆圈作记号，标出所有的数据点
***：**	用星号作记号，标出所有的数据点
.：	用实点作记号，标出所有的数据点
X：	用 **X** 符号作记号，标出所有的数据点
LineWidth	以点为单位的线宽。缺省值是 **0.5**
MarkerSize	以点为单位的记号大小，缺省值是 **6** 点
Xdate	线的 **X** 轴坐标的向量
Ydate	线的 **Y** 轴坐标的向量
Zdate	线的 **Z** 轴坐标的向量
ButtonDownFcn	当线条对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，线条对象没有子对象
Clipping	数据限幅模式
{on}：	在坐标轴界限外的线的任何部分不显示
off：	线条数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	不能被其它回调中断
yes：	可以被其它回调中断
Parent	包含线条对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **line**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	线的可视性
{on}：	线在屏幕上可视
off：	线在屏幕上不可视

表 20.7

文本对象属性

Color	线条颜色。一个三个元素 **RGB** 向量或 MATLAB 预定的颜色名之一。缺省值是 **white**（白色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地

画出。这是最精确的，也是最慢的一种模式

backgrount： 通过在图形背景色中重画文本来消除文本。这会破坏被消除的文本后的对象

xor： 用文本下屏幕颜色执行异或 **OR（XOR）**运算，画出和消除该文本。当画在其它对象上时，会造成不正确的颜色

none： 当移动或删除文本时该文本不会被消除

Extent 文本位置向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于坐标轴对象左下角的文本对象左下角的位置，**[width,height]**是包围文本串的矩形区域的大小，单位由 **Units** 属性指定

FontAngle 文本为斜体

{normal}： 正常的字体角度

italics： 斜体

oblique： 某些系统中为斜体

FontName 文本对象的字体名。缺省的字体名为 **Helvetica**

FontSize 文本对象的大小，以点为单位，缺省值为 **12** 点

***FontStrikeThrough** 值为**[on|{off}]**

***FontUnderline** 值为**[on|{off}]**

FontWeight 文本对象加黑

light： 淡字体

{normal}： 正常字体

demi： 适中或者黑体

bold： 黑体

HorizontalAlignment 文本水平对齐

{left}： 文本相对于它的 **Position** 左对齐

center： 文本相对于它的 **Position** 中央对齐

right： 文本相对于它的 **Position** 右对齐

Position 两元素或三元素向量[**X Y (Z)**]，指出文本对象在三维空间中的位置。单位由 **Units** 属性指定

Rotation 以旋转度数表示的文本方向，

{0}： 水平方向

±90： 文本旋转±90 度

±180： 文本旋转±180 度

±270： 文本旋转±270 度

String 要显示的文本串

Units 位置属性值的度量单位

inches： 英寸

centermeters： 厘米

normalized： 归一化坐标，对象左下角映射到**[0 0]**，右上角映射到**[1 1]**

points： 排字机的点，等于 1/72 英寸

pixels： 排字机的点，等于 1/72 英寸

{data}： 屏幕象素，计算机屏幕分辨率的最小单位

父坐标轴的数据单位

VerticalAlignment		文本垂直对齐
	top:	文本串放在指定的 **Y** 位置顶部
	cap:	字体的大写字母的高度在指定的 **Y** 位置
	{middle}:	文本串放在指定的 **Y** 位置中央
	baseline:	字体的基线在指定的 **Y** 位置
	bottom:	文本串放在指定的 **Y** 位置底部
ButtonDownFcn		当文本对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children		空矩阵，文本对象没有子对象
Clipping		数据限幅模式
	{on}:	在坐标轴界限外的文本的任何部分不显示
	off:	文本数据不限幅
Interruptible		指定 **ButtonDownFcn** 回调字符串是否可中断
	{no}:	不能被其它回调中断
	yes:	可以被其它回调中断
Parent		包含文本对象的坐标轴句柄
***Selected**		值为**[on|{off}]**
***Tag**		文本串
Type		只读的对象辨识字符串，常为 **text**
UserData		用户指定的数据，可以是矩阵、字符串等等
Visible		文本的可视性
	{on}:	文本在屏幕上可视
	off:	文本在屏幕上不可视

表 20.8

曲面对象属性

CData		指定 **ZData** 中每一点颜色的数值矩阵。如果 **CData** 的大小与 **ZData** 不同，**CData** 中包含的图象被映射到 **ZData** 所定义的曲面
EdgeColor		曲面边缘颜色控制
	none:	不画边缘线
	{flat}:	边缘线为单一颜色，由该面 **CData** 的第一个入口项决定。缺省值是 **black**（黑色）
	interp:	各边缘的颜色由顶点的值通过线性插值得到
	A ColorSpec:	3 元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定边缘的单一颜色。缺省值是 **black**（黑色）
EraseMode		消除和重画模式
	{normal}:	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
	backgrount:	通过在图形背景色中重画曲面来消除曲面。这会破坏
	xor:	被消除的曲面后的对象
	none:	用曲面下屏幕颜色执行异或 **OR（XOR）**运算，画出和消除曲面。当画在其它对象上时会造成不正确的颜色当移动或删除曲面时该曲面不会被消除

FaceColor		曲面表面颜色控制
	none:	不画表面，但画出边缘
	{flat}:	第一个 **CData** 入口项决定曲面颜色
	interp:	各面颜色由曲面网格点通过线性插值得到
	A ColorSpec:	3 元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定表面为单一颜色
LineStyle		边缘线形控制
	{-}:	画通过所有网格点的实线
	--:	画通过所有网格点的虚线
	:	画通过所有网格点的点线
	-.:	画通过所有网格点的点划线
	+:	用加号作记号，标出所有的网格点
	o:	用圆圈作记号，标出所有的网格点
	***:**	用星号作记号，标出所有的网格点
	.:	用实点作记号，标出所有的网格点
	X:	用 **X** 符号作记号，标出所有的网格点
LineWidth		边缘线的宽度，缺省值是 **0.5** 点
MarkerSize		边缘线的记号大小，缺省值是 **6** 点
MeshStyle		画行和/或列线
	{both}:	画所有的边缘线
	row:	只画行边缘线
	column:	只画列边缘线
***PaletteMode**		值为**[{scaled}|direct|bypass]**
XData		曲面中点的 **X** 坐标
YData		曲面中点的 **Y** 坐标
ZData		曲面中点的 **Z** 坐标
ButtonDownFcn		当曲面对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children		空矩阵，曲面对象没有子对象
Clipping		数据限幅模式
	{on}:	在坐标轴界限外的曲面的任何部分不显示
	off:	曲面数据不限幅
Interruptible		指定 **ButtonDownFcn** 回调字符串是否可中断
	{no}:	不能被其它回调中断
	yes:	可以被其它回调中断
Parent		包含曲面对象的坐标轴句柄
***Selected**		值为**[on|{off}]**
***Tag**		文本串
Type		只读的对象辨识字符串，常为 **surface**
UserData		用户指定的数据，可以是矩阵、字符串等等
Viible		曲面的可视性
	{on}:	曲面在屏幕上可视
	off:	曲面在屏幕上不可视

表 20.9

补片对象属性

CData	指定沿补片边缘每一点颜色的数值矩阵。只有 **EdgeColor** 或 **FaceColor** 被设为 **interp** 或 **flat** 时才使用
EdgeColor	补片边缘颜色控制
none：	不画边缘线
{flat}：	边缘线为单一颜色，由补片颜色数据的均值指定。缺省值是 **black**（黑色）
interp：	边缘颜色由补片顶点的值通过线性插值得到
A ColorSpec：	三元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定边缘为单一颜色。缺省值是 **black**（黑色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
backgrount：	通过在图形背景色中重画补片来消除该补片。这会破坏被消除的补片后的对象
xor：	用补片下屏幕颜色执行异或 **OR**（**XOR**）运算，画出和消除补片。当画在其它对象上时会造成不正确的颜色
none：	当移动或删除补片时该补片不会被消除
FaceColor	补片表面颜色控制
none：	不画表面，但画出边缘
{flat}：	颜色参量 **c** 中的值决定各补片的表面颜色
interp：	各表面颜色由 **CData** 属性指定的值通过线性插值决定
A ColorSpec：	三元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定表面为单一颜色
LineWidth	轮廓线的宽度，以点为单位。缺省值为 **0.5** 点
***PaletteModel**	值为**[{scaled}|direct|bypass]**
XData	沿补片边缘点的 **X** 坐标
YData	沿补片边缘点的 **Y** 坐标
ZData	沿补片边缘点的 **Z** 坐标
ButtonDownFcn	当补片对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，补片对象没有子对象
Clipping	数据限幅模式
{on}：	在坐标轴界限外的补片的任何部分不显示
off：	补片数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	不能被其它回调中断
yes：	可以被其它回调中断
Parent	包含补片对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **patch**
UserData	用户指定的数据，可以是矩阵、字符串等等

Visible	补片的可视性
{on}:	补片在屏幕上可视
off:	补片在屏幕上不可视

表 20.10

图象对象属性	
CData	指定图象中各元素颜色的值矩阵。**image（c）**将 **c** 赋给 **CData**。**CData** 中的元素是当前颜色映象的下标
XData	图象 **X** 数据;指定图象中行的位置。如忽略,使用 **CData** 中的行下标
YData	图象 **X** 数据;指定图象中列的位置。如忽略,使用 **CData** 中的列下标
ButtonDownFcn	当图象对象被选中时,MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，图象对象没有子对象
Clipping	数据限幅模式
{on}:	在坐标轴界限外的图象的任何部分不显示
off:	图象数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}:	不能被其它回调中断
yes:	可以被其它回调中断
Parent	包含图象对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **image**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	图象的可视性
{on}:	图象在屏幕上可视
off:	图象在屏幕上不可视

20.13 小结

句柄图形函数让用户对图形进行细调，并且显示所建立的图形。每一个图形对象都有一个和它相关的句柄，并可用句柄来操作该对象。对象属性可以用函数 **get** 和 **set** 来修改，以便来定制用户的图形。

本章讨论的函数总结在表 20.11 和表 20.12 中：

表 20.11

句柄图形函数	
set(handle, 'PropertyName' ,Value)	设置对象属性
get(handle, 'PropertyName')	获取对象属性
reset(handle)	将对象属性重设为缺省值
delete(handle)	删除一个对象和它所有的子对象

gcf	获取当前图形的句柄
gca	获取当前坐标轴的句柄
gco	获取当前对象的句柄
findobj('PropertyName' ,Value)	获取具有指定的属性值的对象的句柄
waitforbuttonpress	等待键或鼠标按钮在图形中按下
figure('PropertyName' ,Value)	创建图形对象
axes('PropertyName' ,Value)	创建坐标轴对象
line(X,Y, 'PropertyName' ,Value)	创建线条对象
text(X,Y,S, 'PropertyName' ,Value)	创建文本对象
patch(X,Y,C, 'PropertyName' ,Value)	创建补片对象
surface(X,Y,Z, ' PropertyName' ,Value)	创建曲面对象
image(C, 'PropertyName' ,Value)	创建图象对象

表 20.**12** 中是本章所提到的精通 *MATLAB* 工具箱中的函数：

表 20.**12**

精通 MATLAB 工具箱句柄图形函数	
mmgcf	如当前的图形存在，获取它的句柄
mmgca	如当前的坐标轴存在，获取它的句柄
mmzap(T)	用鼠标删除类型 **T** 的图形对象
mmpx2n(X)	象素到归一化坐标的转换
mm2px(X)	归一化坐标到象素的转换
mmline Name Value…	用鼠标设置线属性
mmaxes Name Value…	用鼠标设置坐标轴属性
mmcont2(X,Y,Z,N,C)	用用户自定义颜色作二维等值线图
mmcont3(X,Y,Z,N,C)	用用户自定义颜色作三维等值线图
mmtile	以平铺形式安排图形窗口
mmpager Name Value…	设置打印的缺省纸张属性

关键词索引

chap20

graphics routine 图形例程
handle 句柄
object 对象
property 属性
stacking order 堆积次序
pixel 象素
Normalized coordinates 归一化坐标
landscape 景象（横向）
portrait 肖像（纵向）
full page 全幅
documented property 文件式属性
undocumented property 非文件式属性
root 根（对象）
figure 图形窗口（对象）
axis 坐标轴（对象）
line 线条（对象）
surface 曲面（对象）
text 文本（对象）
patch 补片（对象）
image 图象（对象）
clipping 限幅

第21章　创建图形用户界面

用户界面是人，即用户与计算机或计算机程序的接触点或交互方式，是用户与计算机进行信息交流的方式。计算机在屏幕显示图形和文本，若有扬声器还可产生声音。用户通过输入设备，如：键盘、鼠标、跟踪球、绘制板或麦克风，与计算机通讯。用户界面设定了如何观看和如何感知计算机、操作系统或应用程序。通常，多是根据悦目的结构和用户界面功能

的有效性来选择计算机或程序。

图形用户界面或GUI是包含图形对象，如：窗口、图标、菜单和文本的用户界面。以某种方式选择或激活这些对象，通常引起动作或发生变化。最常见的激活方法是用鼠标或其它点击设备去控制屏幕上的鼠标指针的运动。按下鼠标按钮，标志着对象的选择或其它动作。

与上一章讨论MATLAB句柄图形功能的相同方式，它让用户按规定设计MATLAB显示信息的方法，本章所描述的图形用户界面的功能，它让用户定制用户与MATLAB的交互方式。命令窗口不是唯一与MATLAB的交互方式。

本章将说明图形句柄**uicontrol** 和**uimenu**对象的使用，把图形界面加到MATLAB的函数和M文件。**uimenu**对象能在图形窗口中产生下拉式菜单和子菜单。**uicontrol**对象能建立如按钮，滚动条，弹出式菜单以及文本框等对象。

MATLAB在**demo**命令中包含了GUI功能的极好例子。

```
>>  demo
```

研究该命令，以了解**uimenu**和**uicontrol**如何给MATLAB函数提供交互输入。

21.1 谁创建图形界面GUI?为什么?

在运行了demo例子后，很可能会问“为什么要在MATLAB中建立一个*GUI*？”这是一个很好的问题，简单的回答是可能并不需要．使用MATLAB来分析数据，求解问题，绘制结果的绝大多数的人，并不会发现GUI工具很有用。

但另一方面，GUI可以在MATLAB中生成非常有效的工具和应用程序，或是建立演示工作的交互式界面。

生成用户图形界面的最常见的理由：

编写一个需多次反复使用的实用函数，菜单、按钮、文本框作为输入方法具有意义；或

编写函数或开发应用程序供别人使用；或

创建一个过程、技术或分析方法的交互式示例；或

认为GUI的简洁，性能良好，并且想实践一下。

许多基于GUI的工具函数包含在精通MATLAB工具箱中，将在后续章节进行讨论。其它由MATLAB用户编制的工具和实用程序装入MATLAB的GUI函数。工具的大多数可在**Mathworks** 匿名**FTP**节点和其它资源中获得。

在我们开始讨论之前，记住对“句柄图形”的理解是设计和实现GUI的先决条件，如果你跳过了前一章，现在应重新回去阅读。

21.2 GUI对象层次结构

正如我们在上一章所展示的那样，由图形命令生成的每一事物是一个图形对象。图形对象不仅包括**uimenu**和**uicontrol**对象，而且还包括图形、坐标轴和他们的子对象。让我们从另一个角度来看这一层次结构。计算机的屏幕本身是根结点，图形是根对象的子对象，坐标

轴，**uimenu** ，**uicontrol**是图形的子对象。

根可以包括多个图形，每个图形含有一组或多组坐标轴以及其子对象，每个图形也可以有一个或多个与坐标轴无关的**uimenu**和**uicontrol**。虽然**uicontrol**对象无子对象结点，但他们确实具有多种类型。**uimenu**对象常将其它的**uimenu**对象作为其子对象。

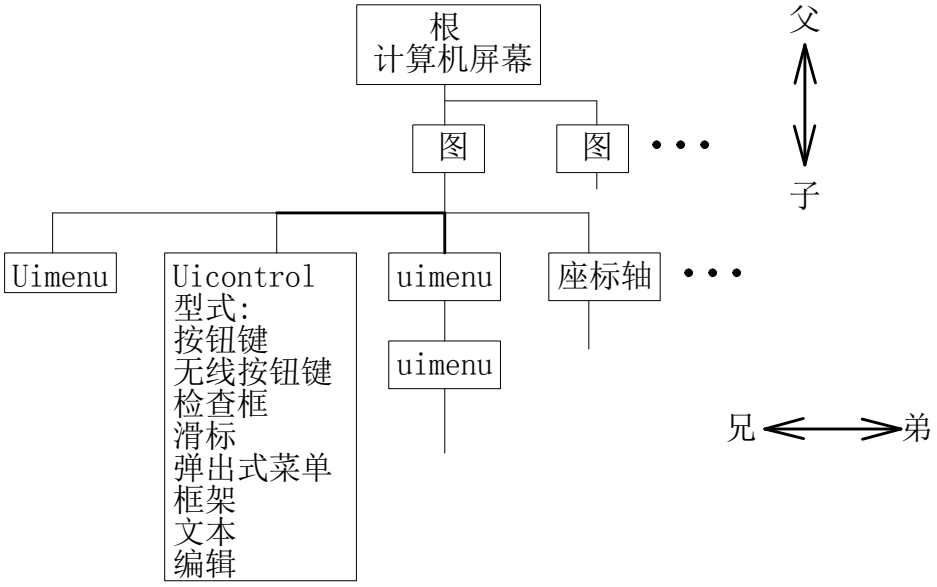


图21.1　GUI对象层次结构图

运行MATLAB的不同型号的计算机或平台上，产生不同的图形显示。Unix工作站使用不同的X Window系统，具有几个窗口程序，如**mwn**或**twm**以控制显示的布局。PC机靠Microsoft Windows或Windows NT进行窗口管理，Macintosh计算机用Macintosh工具箱程序作窗口。虽然在各种平台上，显示看起来有很大的不同，但在很多的情况下，句柄图形的编码是一致的。MATLAB在内部处理平台和窗口系统的差别。体现句柄图形例程的函数，包括应用**uimenu**和**uicontrol**对象的函数，通常运行在所有平台。存在已知差异的地方将在本章后面给出。

21.3　菜单

在每一个窗口系统中使用菜单让用户选择命令和选项。通常在显示屏或窗口的顶部有一菜单条。移动鼠标指针到菜单标志上按下鼠标按键，顶层菜单就被选中，一列菜单项就从菜单标志拉下来。这种款式菜单就叫下拉式菜单。按下鼠标将指针移动至菜单项并松开鼠标，则完成菜单项的选择。MS-Windows 和一些X Window系统平台还提供另一种选择菜单的方法。在顶层菜单上按下并松开鼠标，或称单击鼠标，则打开下拉菜单。然后，移动鼠标指针至下拉菜单项再次单击鼠标，就选择菜单项。在下拉菜单中选择一项就引起动作的发生。

一个菜单项还可用自己的菜单项列表而作为子菜单。子菜单项在子菜单的标志右边显示小三角或箭头以表示菜单还有更多子菜单项可供选择。如果子菜单的菜单项被选择，另一个具有更多菜单项的菜单显示在此菜单的右边的下拉菜单中。有时这种菜单称之行走菜单。选

中其中一个菜单项也引起某些动作的产生。

子菜单可以嵌套，但层次的数目受到窗口系统及有用资源的限制。

菜单的布置

Macintosh在显示屏的顶部使用包含下拉菜单标题的菜单条。选择图形窗口时，菜单条变化以反映激活的图形窗中可利用的选项。标准的Macintosh菜单标题包括**Apple**、 **File**、**Edit**、 **Window** 和**Help** 或**Balloon Help**。由**uimenu**所加的菜单标题放在**Window**和**Help**之间。如果想从菜单条中删去**File**、 **Edit**、和 **Window**菜单标题，可以使用**Set**命令。

```
>> set(Hf_fig1，‘ Menubar ‘，‘ none ‘)
```

Apple和**Help**不可从菜单条中删除。同样，标准的菜单是用以下的命令恢复：

```
>> set(Hf_fig1，‘Menubar‘，‘figure‘)
```

在Microsoft窗口系统下，菜单条位于图形窗口的顶部。每个图形窗口有自己的菜单条，它包含**File**，**Edit**，**Window**和**Help**标题。由**uimenu**所加的菜单标题放在**Help**之后。可以使用与上面相同的**Set**命令从菜单条中删去或恢复所有的标准菜单。

在X Window系统工作站上没有MATLAB标准图形窗口菜单的标题。窗口管理器可将菜单条放置在屏幕上每一个窗口上端，但这些菜单条与MATLAB菜单无关。当建立第一个**uimenu**对象时，MATLAB 在图形窗口的顶部边缘生成自己的菜单条。

建立菜单和子菜单

我们采用函数**uimenu**建立菜单项。**uimenu**的句法与其他对象创建函数相似。如，

```
>> Hm_1=uimenu(Hx_parent，‘PropertyName‘，PropertyValue，...)
```

其中**Hm_1**是由**uimenu**生成的菜单项的句柄，通过设定**uimenu**对象的属性值‘**PropertyName**’，**PropertyValue**这对命令定义了菜单特性；**Hx_parent**是缺省的父辈对象的句柄，必须是图形和**uimenu**对象。

uimenu对象中最重要的属性是'Label'和'Callback'。'Label'属性值是菜单条和下拉菜单项上的文本字符串，以确认菜单项。'Callback'属性值是MATLAB字符串，当选中菜单项时，它传给eval，用以执行。

菜单举例

下面的例子用函数**uimenu**将简单菜单加到当前的图形窗口中。这里提出的例子说明如何只用几个MATLAB命令来建立工作菜单。后面的例子将详细地讨论**uimenu**的命令和属性。

下例用两个下拉菜单将菜单条加到当前窗口中。首先，建立名为Example的顶部菜单输入。

```
>> Hm_ex=uimenu(gcf，‘Label‘，‘Example‘);
```

在此菜单下有两个菜单项。第一项标志为Grid，切换坐标轴格栅的状态。

```
>>  Hm_exgrid=uimenu(Hm_ex，‘Label‘，‘Grid‘，‘Callback‘，‘Grid‘)；
```

注意，句柄Hm_ex是用于与上层菜单相关联的。这项**uimenu**按**eval**的要求出现在上层菜单之下。还需注意的是属性‘Callback‘的值，它是一个带引号的字符串。

Example下的第二项标志为**View**，并带有子菜单。

```
>>  Hm_exview=uimenu(Hm_ex，‘Label‘，‘View‘)；
```

View菜单有两项选择2-D和3-D视图。

```
>>  Hm_ex2d=uimenu(Hm_exview，‘Label‘，‘2-D‘，‘Callback‘，‘view(2)‘)；

>>  Hm_ex3d=uimenu(Hm_exview，‘Label‘，‘3-D‘，‘Callback‘，‘View(3)‘)；
```

注意以上这些是**View**的子菜单，因为它们指定**Hm_exview**作为其父对象。
现在，将第二个顶层菜单加到标题为**Close**的菜单条中。

```
>>  Hm_ex=uimenu(gcf，‘Label‘，‘Close‘)；
```

由该顶层菜单Close加入了两个菜单项。第一项关闭图形窗口，第二项使图形窗口打开，但去掉用户菜单。

```
>>  Hm_clfig=uimenu(Hm_close，‘Label‘，‘Close Figure‘，‘Callback‘，‘Close‘)；

>>  Hm_clmenu=uimenu(Hm_close，‘Label‘，‘Remove Menu‘，...
                ‘Callback‘，‘delete(Hm_ex)；  delete(Hm_ex)；  drawnow‘)；
```

在精通MATLAB工具箱的脚本文件**mmenu1.m**中含有上面的例子。所以你可以运行**mmenu1.m**来验证上例。

菜单属性

如上所示，同句柄图形函数一样，在建立图形对象时可定义**uimenu**属性，或用set改变属性。所有可设定的属性，包括标题、菜单颜色、甚至回调字符串都可以用set来改变。这种功能十分便于迅速地定制菜单和属性。

表21.1列出了MATLAB 4.2版本中的**uimenu**对象的属性及其属性值。带有*的属性是非文件式的，使用时需加小心。在括号{}内的属性值是缺省值。

表21.1

Uimenu 对象的属性

Accelerator		指定菜单项等价的按键或快捷键。对于X-windows，按键顺序是**Control**- 字符；Macintosh系统，按键顺序是**Command** - 字符或# - 字符
BackgroundColor		**uimenu**背景色,是一个3元素的RGB向量或MATLAB预先定义的颜色名称。缺省的背景色是**亮灰**色
Callback		MATLAB回调字符串，选择菜单项时，回调串传给函数**eval**；初始值为空矩阵
Checked		被选项的校验标记
	on:	校验标记出现在所选项的旁边
	{**off**}:	校验标记不显示
Enable		菜单使能状态
	{**on**}:	菜单项使能。选择菜单项能将**Callback**字符串传给
	off:	**eval**
		菜单项不使能，菜单标志变灰。选择菜单项不起任何作用。
ForegroundColor		**uimenu**前景（文本）色,是一个三元素的RGB向量或MATLAB预
		先定义的颜色名称。缺省的前景色是**黑色**
Label		含有菜单项标志的文本串。在PC系统中，标记中前面有‘&‘，定
		义了快捷键，它由**Alt - 字符**激活
Position		**uimenu**对象的相对位置。顶层菜单从左到右编号，子菜单从上至
		下编号
Separator		分割符 - 线模式
	on:	分割线在菜单项之上
	{**off**}:	不画分割线
*Visible		**uimenu**对象的可视性
	{**on**}:	**uimenu**对象在屏幕上可见
	off:	**uimenu**对象不可见
ButtonDownFcn		当对象被选择时，MATLAB的回调串传给函数**eval**。初始值为空矩
		阵。
Children		其它**uimenu**对象的句柄。
Clipping		限幅模式
	{**on**}:	对uimenu对象无效果
	off:	对uimenu对象无效果
DestroyFcn		仅用于Macintosh 4.2 版本。没有文本说明。
Interrruptible		指明**ButtonDownFcn**和**CallBack**串可否中断
	{**no**}:	回调不可中断
	yes:	回调串可中断
Parent		父对象的句柄；如果**uimenu**对象是顶层菜单，则为图形对象；若
		uimenu是子菜单，则为父的**uimenu**对象句柄

*Select	值为**[on|off]**
*Tag	文本串
Type	只读对象辩识串，通常为**uimenu**
UserData	用户指定的数据。可以是矩阵，字符串等等
Visible	**uimenu**对象的可视性
{**on**}：	**uimenu**对象在屏幕上可见
off：	**uimenu**对象不可见

属性值仅仅定义了**uimenu**对象的性质并且控制菜单如何显示；它们也决定了选择菜单项所引起的动作。其中某些性质将在下面更详细地讨论。

菜单快捷键

' Label ' 属性义定了出现在菜单或菜单项中的标志。它也可以用来定义Microsoft Windows系统的快捷键：标志字符串中，在所需字符前加上&，例如：

```
>>  Hm_top=uimenu( ' Label ' ,  ' Example ' );

>>  uimenu(Hm_top,  ' Label ' ,  ' &Grid ' ,  ' CallBack ' ,  ' grid ' );
```

它定义了键盘上**G**为快捷键。菜单项标志将以Grid形式出现在菜单上。为激活快捷键，在选择图形窗口时按Alt键并按下**G**键。快捷键不一定是字符串的第一字符。下例中**R**为快捷键：

```
>>  uimenu(Hm_top,  ' Label ' ,  ' G & rid ' ,  ' CallBack ' ,  ' grid ' );
```

则标志以**Grid**形式出现在菜单上。

Macintosh平台用 ' **Accelerator** ' 属性而不是 ' **Label** ' 来定义快捷键。在Macintosh

```
>>  uimenu(Hm_top,  ' Label ' ,  ' Grid ' ,  ' Accelerator ' ,  ' G ' ,  ' CallBack ' ,  ' grid ' );
```

定义**G**为快捷键，菜单标志以**Grid#G**的形式出现。为激活快捷键，选择图形窗口时，按**Command**键或#键并按下**G**键。

在Macintosh上不可为顶层菜单定义快捷键。另外，已经定义在标准Macintosh上的快捷键，如**Command C**或# **C**不撤除标准Macintosh菜单，就不能复制。

在X-Window 系统中定义和使用快捷键与Macintosh相似，但仍存在某些差异。同Macintosh一样，用 **' Accelerator '** 属性而不是 **' Label '** 属性来定义快捷键。但快捷键在菜单上显示不同。例如，

```
>>  uimenu(Hm_top,  ' Label ' ,  ' Grid ' ,  ' Accelerator ' ,  ' G ' ,  ' CallBack ' ,  ' grid ' );
```

定义字母**G**为快捷键。菜单标志常以Gr**id<Ctrl>-G**出现在菜单上。未使用快捷键，需按**Control**键并按下**G**键。同Macintosh一样，不可为顶层菜单定义快捷键。

虽然我们没有对此验证，但MATLAB用户手册指出当给定相同命令时，用X的某些工作站动作不一样。如果顶层菜单标志中有带下划线的字符，就可按**Meta**键并按下下划字符键来选择菜单。如果子菜单项标志中含有带下划线的字符，就按下该字符键来选择菜单项。请参阅键盘使用说明，以为系统确定合适的Meta键。

菜单的外观

影响菜单的布置和外观的三个属性为 **'Position'**， **'Checked'**，和 **'Separator'**.**uimenu**对象的 **'Position'** 属性值是一个整数，它定义了相对于其它菜单和菜单项的位置。在生成菜单时，设定 **'Position'** 属性。菜单条的最左端的菜单条和下拉菜单中的上端菜单项处在位置1。

设置‘**Position'** 属性可以重新排列菜单位置。考虑下面的例子：

```
>>  Hm_1=uimenu( ‘ Label ‘ ， ‘ first ‘ )；      %  Create two menus

>>   Hm_2=uimenu( ‘ Label ‘ ， ‘ Second ‘ )；

>>  get(Hm_1， ‘ Position ‘ )              %  Check the locations
ans=
     1

>>  get(Hm_2， ‘ Position ‘ )
ans=
     2

>>  set(Hm_2， ‘ Position ‘ ，1)              %  Change menu order

>>  get(Hm_1， ‘ Position ‘ )              %  check the locations

ans=
     2

>>  get(Hm_2， ‘ Position ‘ )

ans=
     1
```

注意，当一个**uimenu**的 **'Position'** 属性改变，就将移动其它**uimenu**以适应此变化，它们的 **'Position'** 属性均更新。子菜单中菜单选项的编号以同样的方式重新排列。

属性 **'Checked'** 的值使校验标记出现在菜单项标志的左边。缺省值为 **'off'** 。命令

```
>>  set(Hm_item， ‘ Checked ‘ ， ‘ on ‘ )
```

使校验标记出现在**Hm_item uimenu**标志的旁边。对于创建代表属性的菜单项，该命令十分

有用。例如，

```
>>Hm_top=uimenu(‘Label‘，‘Example‘);
>>Hm_box=uimenu(Hm_top，‘Label‘，‘Axis Box‘，...‘CallBack‘，[...
                ‘if strcmp(get(fca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
                ‘seet(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
               set(Hm_box，‘‘Checked‘‘，...off‘‘)，‘，...
                ‘else，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
                ‘set(Hm_box，‘‘Checked‘‘，‘‘on‘‘)，‘，...‘end‘]);
```

建立了以**Axis Box**为标志的下拉菜单项。当选中该项时，就运行回调字符串所表示的命令。回调字符串确定了当前坐标轴的**'Box'**属性值，并适当地设定坐标轴的‘**Box'**属性及**uimenu**的**'Checked'**属性。该例子可从精通MATLAB工具箱的M脚本文件**mmenu2.m**中得到。

可以通过改变**uimenu**的**'Label'**属性以反映菜单项当前的状态。下面的例子（在**mmenu3.m**中)改变了菜单项标志本身，而不是加一个校验标记：

```
>>  Hm_top=uimenu(‘Label‘，‘Example‘);

>>  Hm_box+uimenu(Hm_top，‘Label‘，‘Axis Box‘，...
                ‘CallBack‘，[...
                ‘if strcmp(get(gca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
               set(Hm_box，‘‘Label‘‘，‘‘Set Box On‘‘)，‘，...
                ‘else，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
                ‘set(Hm_box，‘‘Label‘‘，‘‘Set box Off‘‘)，‘，...
                ‘end‘]);
```

使用**' Separator'**属性可将下拉菜单分成局部组。如果一个**uimenu**项的**' Separator'**属性是**'on'**，则在下拉菜单中显示此项时，此项的上端有一条水平线将其与前面的菜单项隔开。缺省值是**'off'**，但在菜单生成时可以改变，

```
>>  Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘，‘on‘);
```

或是在以后使用**set**命令

```
>>  set(Hm_box，‘Separator‘，‘on‘);
```

顶层菜单忽略**'Seprator'**的属性值。

使用**'Seperator'**属性可将下拉菜单项分成若干逻辑组。如果对**uimenu**项**'Seperator'**属性设置为**'on'**，用在一条其上面的水平线来表现该项，将其与前面的菜单项形象地区分开。缺省值为**'off'**，但在生成菜单时可以改变，

```
>> Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘.‘on‘)
```

或是在以后使用**set**命令：

```
>> set(Hm_box.‘Seperator‘，‘on‘)
```

顶层菜单忽略 **'Seperator'** 属性值。

颜色控制

uimenu对象可设置两个颜色属性。 **'BackGroundColor'** 属性控制填充菜单背景的颜色。缺省值是浅灰。另一颜色属性为 **'ForeGroundColor'**，它确定菜单项文本的颜色，缺省值是黑色。

颜色属性同样能很好地用于顶层菜单条和下拉菜单。颜色可以用来表示状态信息或简单加上菜单的特 色。例如，挑选线段颜色。在子菜单中，各菜单项的背景可以填充合适的色彩。

```
>> Hm_green=uimenu(Hm_color，'Label'，'Green'，'BackgroundColor'，'g'，...
                    'Callback'，'set(Hl_line，'Color'，'g')');
```

菜单项去能

改变对象**uimenu**的 **'Enable'** 值或 **'Visible'** 属性可使菜单项暂时去能。 **'Enable'** 属性通常设
为 **'on'** 。当 **'Enable'** 属性 ' 设为 **'off'** 时，标志字符串变灰，菜单项去能。在这种状态下，菜单项保持可见但不能被选择。此属性可用来将不恰当的菜单选择去能。

下面的例子(**mmenu4.m**)说明了用两个菜单项和 **'Enable'** 属性来设定坐标轴的 **'Box'** 属性的另一种方法。

```
>> Hm_top = uimenu('Label'，Example');

>> Hm_boxon = uimenu(Hm_top，'Label'，'Set Box On'...'CallBack'，[...
            'set(gca，  'Box'  '，  'on'  ')，  '，...
            'set(Hm_boxon，  'Enable'  '，  'off'  ')，  '，...
            'set(Hm_boxoff，  'Enable'  '，  'Enable'  '，  'on'  ')
'']);

>> Hm_boxoff = uimenu(Hm_top，'Label'，Set Box Off'，...'Enable'，'
off'，...'CallBack'，[...
            'set(gca，  'Box'  '，  'off'  ')，  '，...
            'set(Hm_boxon，  'Enable'  '，  'on'  ')，  '，...
            'set(Hm_boxoff，  'Enable'  '，  'off'  ')']);
```

设定**'Visible'** 属性为 **'off'**， 可将菜单项完全隐藏。菜单项象是从屏幕中消失，而其它菜

单项改变了在显示器上的位置以填补由当前不可见菜单造成的空隙。然而，不可见的菜单仍然存在，而且**uimenu** 对象
的 **'Position'** 属性值也不改变。当属性 **'Visible'** 又重新设为 **'on'** 时，菜单项重新出现在正常的位置。

这个性质可以用来暂时地撤消一个菜单。下面的例子(**mmenu5.m**)建立了两个顶层菜单和两个菜单项。

```
>> Hm_control = uimenu('Label', 'Control');

>> Hm_extra = uimenu('Label', 'Extra');

>> Hm_limit = uimenu(Hm_control, 'Label', 'Limited Menus', ...
                'CallBack', 'set(Hm_extra, ''Viseble'', ''off'')
');

>> Hm_full = uimenu(Hm_control, 'Label', 'Full Menus', ...
            'CallBack', 'set(Hm_extra, ''Visible'', ''on'')');
```

当选择了**Limited Menus**项时，**Extra**菜单就从菜单条中消失。当选择了**Full Menus**项时，**Extra**菜单又重新显示在原来的位置的菜单条上。

回调属性

'Callback' 属性值是一个MATLAB字符串，MATLAB将它传给函数**eval**并在**命令窗口工作空间**执
行。它对于函数M文件有重要的隐含意义，我们将在本章后面继续讨论这一属性。

因为 **'Callback'** 属性必须是字符串，所以在字符号内多重MATLAB命令、后续行以及字符串都会使必需的句法变得十分复杂。如果有不止一个命令要执行，命令间必须适当地分隔开来。例如，

```
>>uimenu('Label', 'Test', 'CallBack', 'grid on; set(gca, ''Box'', ''on'
')');
```

把一个字符串传给**eval**，使命令

```
>> grid on; set(gca, 'Box', 'on')
```

在命令窗口工作空间中执行。这是合法的句法，因为命令用逗号或分号隔开，多重命令可输入到同一命令行中。在定义回调函数时，也遵循MATLAB规定，即在已引用的字符串内，用两个单引号来表示单引号。

字符串可以串接起来生成一个合法MATLAB字符串，只要把它们括在方括号中。

```
>>uimenu('Label', 'Test', 'CallBack', ['grid on, ', ' set(gca, ''Box'', '
''on '' )']);
```

注意字符串 **'grid on'** 含有所需的逗号以分隔两个命令。

如果使用了续行号，上述命令可写为

```
>>uimenu('Label', 'Test', ...
        'CallBack', [...
        'grid on, ', ...
        '  set(gca, ' 'Box' ', ' 'on' ')'...
       ]);
```

上例中命令行被分隔，每行的末尾加上了三个句号表示命令的继续。注意到上列单行的所有元素都被保留，包括字符串分隔命令的逗号。在 'grid on, …' 行中最后引号后的逗号是可选的；下一行开始的空格起相同的作用。欲了解详情，请参阅前面关于建立行向量的章节。

如果引号、逗号和括号不正确输入，MATLAB将给出警告；但在复杂回调字符串中很难寻找错误

的。为了使错误最少，对包含MATLAB语句的回调字符串请记住以下的一些规则：

把整个回调字符串括在方括号中，不要忘记最后的右括号'）'。

把各语句括上单引号。

已引用的字符串内，要用双引号。如：**'quoted': 'a ' 'quoted' ' string';'Quote ' 'a ' ' 'quoted' ' ' string' 'now'**。在引号后要用逗号或空格结尾。

除了最后一句，各语句在**引号内**要以逗号或分号结尾；在**引号后**要用逗号或空格结尾。

有后续行的各行要以三个句号(...)结尾。

前面的例子之一**mmenu4.m**是所涉及的回调字符串句法的很好说明。

```
>> Hm_top = unimenu('Label', 'Example');

>> HM_boxon = uimenu(Hm_top,...
                     'Label', 'Set' Box on', ...
                     'CallBack', [...
                          'set(gca,"Box","on"), ', ...
                          set(Hm_boxon,"Enable","off"), ', ...
                          set(Hm_boxoff,"Enable","on")'];

>> Hm_boxoff = uimenu(Hm_top,...
                      'Label', 'Set Box off', ...
                      'Enable', 'off' ', ', ...
                      'CallBack', [...
                           'set(gca,"Box","off", ', ...
                           'set(Hm_boxon,"Enable","on"), ', ...
                           'set(Hm_boxoff'Enable","off")'];
```

上例中还引出了关于回调函数另一个重点，在变量**Hm_boxoff**定义之前，在回调串中用**Hm_boxoff**替代**Hm_boxon**。因为回调串只是一个字符串，MATLAB 不会给出警告，而且仅在**uimenu**被激活并将字符串传给**eval**时才由MATLAB执行。它隐含有函数M文件的设计和测试，这将在本章后面讨论。

M文件的举例

下例将演示一组简单菜单的生成。该例子包含在精通MATLAB工具箱的函数M文件**mmenus**中。正如下面所示的那样，这个函数文件被分隔成了若干块，以便于讨论函数的各个方面。

首先，定义一个函数并在当前的图形中用顶层**Line**菜单建立菜单条，该菜单分别含有三个子菜单：**Line Style**，**Line Width**，**Line Color**。

```
function mmenus()
%   MMENUS Simple menu example.
%   MMENUS uses waitforbuttonpress and gco in callback strings
%   to let the user make a menu selection and then select an object
%   by clicking on it with the mouse. The callback strings then use
%   the set function to apply the property value to the selected
%   object.
%   Copyright (c) 1996 by Prentice-Hall，Inc.

Hm_line   =   uimenu(gcf，  'label'.'Line');
Hm_lstyle   =   uimenu(Hm_line，  'label'，  'Line Style');
Hm_lwidth   =   uimenu(Hm_line，  'label'，  'Line width');
Hm_lcolor   =   uimenu(Hm_'label，  'Line Color');
```

其次，使用**waitforbuttonpress**和**gco**得到当前对象的句柄，确认它为一个线对象，并采用适当的**'Linestyle'**值。注意这些菜单项句柄以后不再使用，所以它们不必保存。

```
uimenu(Hm_lstyle，  'Label'，Solid'，...
            'CallBack'，('waitforbuttonpress；  '，...
                    'if get(gco，"type")=="line"，  '...
                        'set(gco，"LineStyle"，"-")，  '...
                    'end']);

uimenu(Hm_lstyle，  'Label'，Dotted'，...
            'Callback'，['waitforbuttonpress；  '，...
                    'if get(gco，"Type")=="line"，  '，...
                        'set(gco ，"LineStyle"，"，"：")，  '...
                    'end']);

uimenu(Hm_lstyle，  'Label'，Dashed'，...
              'Callback'，['waitforbuttonpress；  '，...
```

```
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineStyle"，"-"：")， '...
                    'end']);

    uimenu(Hm_lstyle， 'Label'，DashDot'，...
                'Callback'，['waitforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineStyle"，"-"：")， '...
                    'end']);
```

现在，对**Line width**子菜单项也做同样的操作：

```
    uimenu(Hm_lwitdth， 'Label'， 'Default'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，0.5)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thick'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，2.0)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thicker'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，3.0)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thickest'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，4.0)， '，...
                    'end']);
```

对**Line Color**子菜单项也做同样的操作。将菜单项背景加色并在合适时改变文本颜色。

```
    uimenu(Hm_lcolor， 'Label'， 'yellow'，...
            'BackgroundColor'， 'y'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"color"， '"y")， '，...
                    'end']);
```

```
uimenu(Hm_lcolor， 'Label'， 'Magenta'， ...
          'BackgroundColor'， 'm'， 'ForegroundColor'， 'w'...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"m")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'yan'，...
          'BackgroundColor'， 'c'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"c")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Red'，...
          'BackgroundColor'， 'r'， 'ForegroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"r")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Green'，...
          'BackgroundColor'， 'g'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"g")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Blue'，...
          'BackgroundColor'， 'b'， 'ForegroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"b")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'White'，...
          'BackgroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"w")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Black'，...
```

'BackgroundColor'， 'k'， 'ForegroundColor'， 'w'， ...
'CallBack'，['witforbuttonpress； '， ...

```
        ' if get(gco，"Type")=="line"， ' ，...
            ' set(gco ，"color"， '"k")， ' ，...
        ' end ' ])；
```

为使这一函数更完全，可用同样方法增加另外的菜单以改变坐标轴、曲面、补片和图形的属性。例如，下面加上了一个**Color**映象以改变图形颜色的映象。

```
Hm_cmap=uimenu(gcf， ' Label ' ， ' Color Map ' )；

uimenu(Hm_cmap， ' Label ' ， ' Lighter ' ， ' CallBack ' ， ' brighten(.3) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Darker ' ， ' CallBack ' ， ' brighten(-.3) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Default ' ， ' CallBack ' ， ' colormap("default") ' )；
uimenu(Hm_cmap， ' Label ' ， ' Gray ' ， ' CallBack ' ， ' colormap(gray) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Hot ' ， ' CallBack ' ， ' colormap(hot) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Cool ' ， ' CallBack ' ， ' colormap(cool) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Bone ' ， ' CallBack ' ， ' colormap(bone) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Copper ' ， ' CallBack ' ， ' colormap(copper) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Pink ' ， ' CallBack ' ， ' colormap(pink) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Prism ' ， ' CallBack ' ， ' colormap(prism) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Jet ' ， ' CallBack ' ， ' colormap(jet) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Flag ' ， ' CallBack ' ， ' colormap(flag) ' )；
uimenu(Hm_cmap， ' Label ' ， ' HSV ' ， ' CallBack ' ， ' colormap(hsvflag) ' )；
```

最后，加上含有两个菜单项的**Quit**菜单。其中，**Close Figure**关闭图形；**Remove Menus**让图形打开但删除用户的顶层菜单及所有它的子菜单。**drawnow**命令立即删除菜单。

```
Hm_quit=uimenu(GCF， ' lABEL ' ， ' quit ' )；

uimenu(Hm_quit， ' Label ' ， ' Close Figure ' ， ' CallBack ' ， ' close；return ' )；

uimenu(Hm_quit， ' Label ' ， ' Remove Menu ' ，...
        ' Callback ' ，[...
            ' delete(findobj(gcf，"Type，"uimenu"，"parent"，gcf))， ' ，...
            ' drawnow ' ])；
```

精通MATLAB工具箱中有许多采用这种技术的函数和其它建立有用的基于对象工具的函数。其中许多函数将在本章后面详细讨论。

21.4 控制框

在各计算机平台上，窗口系统都采用控制框和菜单，让用户进行某些操作，或设置选项或属性。控制框是图形对象，如图标、文本框和滚动条，它和菜单一起使用以建立用户图形界面，称之为窗口系统和计算机窗口管理器。

MATLAB控制框，又称**uicontrol**，与窗口管理器所用的函数十分相似。它们是图形对象，可以放置在MATLAB的图形窗中的任何位置并用鼠标激活。MATLAB的 **uicontrol**包括按钮、滑标、文本框及 弹出式菜单。

由MATLAB生成的**uicontrol**对象在Macintosh、MS-Windows 和X Window系统平台上，有稍微不同的外观，因为窗口系统表达图形对象的方法是不同的。但是，功能本质是相同的，所以相同的MATLAB编码将生成同样的对象，它在不同平台完成同样的功能。

Uicontrol由函数**uicontrol**生成。常用句法与前面所讨论的**uimenu**相同。

```
>>Hc_1=uicontrol(Hf_fig， 'PropertyName'，PropertyValue，...）
```

其中，**Hc_1**是由函数**uicontrol**生成**uicontrol**对象的句柄。通过设定**uicontrol**对象的属性值**'PropertyName'**， **'PropertyValue'** 定义了**uicontrol**的属性；**Hf_fig**是父对象的句柄，它必须是图形。如果图形对象句柄省略，就用当前的图形。

建立不同类型的控制框

MATLAB共有八种不同类型或型式的控制框。它们均用函数**uicontrol**建立。属性 **'Style'** 决定了所建控制框的类型。 **'Callback'** 属性值是当控制框激活时，传给**eval**在命令窗口空间执行的MATLAB字符串。

下面将分别对八种**uicontrol**对象进行讨论，并用示例说明。**uicontrol**对象的属性更为透彻的讨论和应用中更为完整的例子将在以后给出。

按钮键

按钮键，又称命令按钮或只叫按钮，是小的长方形屏幕对象，常常在对象本身标有文本。将鼠标指针移动至对象，来选择按钮键**uicontrol**，单击鼠标按钮，执行由回调字符串所定义的动作。按钮键的 **'Style'** 属性值是 **'pushbutton'** 。

按钮键典型地用于执行一个动作而不是改变状态或设定属性。下面的例子（**mmctl1.m**）建立标志为**Close**的按钮键**uicontrol**。当激活该按钮时，**close**关闭当前的图形。以象素为单位的**'Position'** 属性定义按钮键的大小和位置，这是缺省的 **'Units'** 属性值。属性 **'String'** 定义了按钮的标志。

```
>>Hc_close=uicontrol(gcff， 'Style'， 'push'，...
                    'Position'，[10 10 100 25]，...
                    'String'，Close'，...
                    'CallBack'， 'close)；
```

无线按钮

无线按钮，又称选择按钮或切换按钮，它由一个标志并和标志文本的左端一个小圆圈或小菱形所形
成。当选择时，圆圈或菱形被填充，且 **'Value'** 属性值设为1；若未被选择，指示符被清除，**'Value'** 属性值设为0。无线按钮键 **'style'** 的属性值是 **'radiobutton'** 。

无线按钮典型地用在一组互斥的选项中选择一项。为了确保互斥性，各无线按钮

uicontrol的回调字符串必须不选组中其它项，将它们各项的 **'Value'** 设为0。然而，这只是一个约定，如果需要，无线按钮可与检查框交换使用。

下面的例子（**mmctl2.m**）建立了两个互斥选项的无线按钮，它将坐标轴 **'Box'** 属性开或关闭。

```
>> Hc_boxon = uicontrol(gcf, 'Style', 'radio', ...
                'Position', [20 45 100 20], ...
                'String', 'Set box on
                'Value', 0, ...', ...
                'CallBack', [...
                    'set(Hc_boxon', "Value", 1'...
                    'set(Hc_boxoff', "Value", 0'...
                    'set(gca, "Box", "on")']);

>> Hc_boxoff = uicontrol(gcf, 'Style', 'radio', ...
                'Position', [20 20 100 20], ...
                'String', 'Set box off
                'Value', 1, ...', ...
                'CallBack
                'CallBack', [...
                    'set(Hc_boxon', "Value", 0'...
                    'set(Hc_boxoff', "Value", 1'...
                    'set(gca, "Box", "off")']);
```

检查框

检查框，又称*切换按钮*，它由具有标志并在标志的左边的一个小方框所组成。激活时，**uicontrol**在检查和清除状态之间切换。在检查状态时，根据平台的不同，方框被填充，或在框内含**x**， **'Value'** 属性值设为1。若为清除状态，则方框变空， **'Value'** 属性值设为0。

检查框典型地用于表明选项的状态或属性。通常检查框是独立的对象，如果需要，检查框可与无线按钮交换使用。

下面的例子（**mmctl3.m**）建立了一个检查框**uicontrol**， 设置坐标轴 **'Box'** 属性，当此检查框被激活时，测试 **'Value'** 属性以确定检查框是否以往被检查或清除过，并适当设置 **'Box'** 属性。

```
>> Hc_box = uicontrol(gcf, 'Style', 'check', ...
                'Position', [100 50 100 20], ...
                'String', 'Axis Box', ...
                'CallBack', [...
                    'if get(Hc_box, "Value")==1, '...
                        'set(gca, "Box", "on"), '...
                    'else, '...
                        '(gca, "Box", "off", '...
                    'end']);
```

静态文本框

静态文本框是仅仅显示一个文本字符串的**uicontrol**，该字符串是由 **'string'** 属性所确定的。静态文本框的 **'Style'** 属性值是 **'text'** 。静态文本框典型地用于显示标志、用户信息及当前值。

静态文本框之所以称之为 ' 静态 ' ，是因为用户不能动态地修改所显示的文本。文本只能通过改变
'String' 属性来更改。

在X Window系统中，静态文本框可只含有一行文字；如文本框太短，不能容纳文本串，则只显示部分文字。然而在Macintosh和MS-Window平台，长于文本框的文本串将字串起来，即在可能的地方，用词间分割的虚线显示多行。

下面的例子（**mmct14.m**）建立了含有MATLAB版本号的文本框。

```
>>  Hc_ver  =  uimenu(gcf，  'Style'，  'text'，...
                    'Position'，[10   10   150   20]，...
                    'String'，['MATLAB Version'，version])；
```

与其它的文本对象，如：坐标标题和坐标标志不同，函数的编著者对用在uicontrol文本串的字体不明显地控制。用在uicontrol文本串的字体和命令窗口的字体一致，可由用户设定。

X window系统的用户在激活MATLAB时，可以给出命令行的参量，如

```
matlab   -fn 9x14bold
```

该命令在命令窗和**uicontrol**文本串中使用**9x14bold**字体。Mocintosh对命令窗和**uicontrol**文本串使用相同的字体，但字体可以由用户在命令窗的**Options**菜单中用**Text Style**项来设定。PC用户具有选项，可从命令窗口的**Options**菜单的**Command Window Font**项来设置命令窗口字体，并从同一菜单中的**Uicontrol Font**项，为**uincontrol**文本串设置不同的字体。我们希望未来的MATLAB版本会在**uimenu**和**uicontrol**文本串中增加控制字体的属性。

可编辑文本框

编辑文本框，象静态文本框一样，在屏幕上显示字符。但与静态文本框不同，可编辑文本框允许用户动态地编辑或重新安排文本串，就象使用文本编辑器或文字处理器一样。在 **'String'** 属性中有该信息。可编辑文本框**uicontrol**的 **'Style'** 属性值是 **'edit'** 。可编辑文本框典型地用在让用户输入文本串或特定值。

可编辑文本框可包含一行或多行文本。单行可编辑文本框只接受一行输入，而多行可编辑文本框可接受行以上的输入。单行可编辑文本框的输入以**Return**键结尾。在X window和MS-Window系统中，多行文本输入以**Control-Return**键结尾，而在Macintosh中用**Command-Return**键。

下面的例子（**mmct15.m**)建立了静态文本标志和一个单行可编辑文本框。用户可以在文本框中输入颜色映象名，而后回调字符串把它放到在图中。

```
>>Hc_label=uicontrol(gcf, 'Style', 'text', ...
                    'Position'[10  10  70  20], ...
                    'String', 'Colormap: ');

>>Hc_map =uicontrol(gcf, 'Style', 'edit ', ...
                    'Position'[80  10  60  20], ...
                    'String', 'hsv', ...
                    'callback', 'coloormap(eval(get(Hc_map, "String")))');
```

通过把 ' Max ' 属性及 ' Min ' 属性设置成数值，诸如Max-Min>1，建立多行可编辑文本框。Max属性不指定最大的行数。多行可编辑文本框可具有无限多行。

一个多行可编辑文本框表示如下：

```
>>Hc_multi=uicontrol(gcf, 'Style', 'edit', ...
                      'Position', [20 50 75 75], ...
                      'String', 'Line 1 |Line 2|Line 3' ...
                      'Max', 2);
```

多行字符串被指定为单个引号的字符串，用垂直条字符 '|' 指明在何处分行。

滑标

滑标，或称滚动条，包括三个独立的部分，分别是滚动槽、或长方条区域，代表有效对象值范围；滚动槽内的指示器，代表滑标当前值；以及在槽的两端的箭头。滑标**uicontrol**的**'Style'** 属性值是**'slider'**。

滑标典型地用于从几个值域范围中选定一个。滑标值有三种方式设定。方法一：鼠标指针指向指示器，移动指示器。拖动鼠标时，要按住鼠标按钮，当指示器位于期望位置后松开鼠标。方法二：当指针处于槽中但在指示器的一侧时，单击鼠标按钮，指示器按该侧方向移动距离约等于整个值域范围的10% ；方法三：在滑标不论哪端单击鼠标箭头；指示器沿着箭头的方向移动大约为滑标范围的1% 。滑标通常与所用文本**uicontrol**对象一起显示标志、当前滑标值及值域范围。

下面的例子（**mmct16.m**）实现了一个滑标，可以用于设置视点方位角。用了三个文本框分别指示滑标的最大值，最小值和当前值。

```
>> vw = get(gca, 'View');

>> Hc_az = uicontrol(gcf, 'Style', 'slider ', ...
                    'Position'[10  5  140  20], ...
                    'Min', -90, 'Max', 90, 'Value', vw(1), ...
                    'CallBack', [...
                          'set(Hc_cur,"String",num22str(get(Hc_az,"Value"))),
'...
                          'set(gca, "View", [get(Hc_az, "Value") vw(2)])']);
```

```
>> Hc_min = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[10  25  40  20]，...
                    'String'，num2str(get(Hc_az， 'Min')))；

num2str(get(Hc_az， 'Min')))；

>> Hc_max = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[110  25  40  20]，...
                    'String'，num2str(get(Hc_az， 'Max))；

>> Hc_cur = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[60  25  40  20]，...
                    'String' num2str(get(Hc_az， 'Value')))；
```

滑标的 **'Position'** 属性包含熟悉的向量[**left bottom width height**]，其单位由 **'Units'** 属性设定。滑标的方向取决于宽与高之比。如果**width > height**就画水平方向的滑标，如果**width < height**就画垂直方向的滑标。仅在X-Window系统平台中，如果滑标的一个方向的大小比另一个方向小4倍，就不显示。其它操作平台上的滑标均有箭头。

弹出式菜单

弹出式菜单典型地用于向用户提出互斥的一系列选项清单，让用户可以选择。弹出式菜单，不同于前面论述过的下拉式菜单，不受菜单条的限制。弹出式菜单可位于图形窗口内的任何位置。弹出式菜单的 **'Style'** 属性值是 **'popupmenu'** 。

当关闭时，弹出式菜单以矩形或按钮的形式出现，按钮上含有当前选择的标志，在标志右侧有一个向下的箭头或凸起的小方块来表明**uicontrol**对象是一个弹出式菜单。当指针处在弹出式**uicontrol**之上并按下鼠标时，出现其它选项。移动指针到不同的选项，松开鼠标就关闭弹出式菜单，显示新的选项。MS-Windows 和某些X Window系统平台允许用户单击弹出式菜单，打开它，而后单击另一个选项来进行选择。

当选择一个弹出项时， **'Value'** 属性值设置成选择向量所选元素的下标。选项的标志指定为一个字符串，用垂直条 '|' 分隔，与指定多行文本的方法一样。下面的例子（**mmct17.m**）建立了图形颜色的弹出式菜单。回调函数把图形的 **'Color'** 属性值设定为所选值。每种与颜色相关的RGB值存储在弹出控制框的
'UserDate' 属性中。所有句柄图形对象的 **'UserData'** 属性仅仅为单独矩阵提供孤立的存储。

```
>> Hc_fcolor = uicontrol(gcf， 'Style'， 'popumenu'，...
        'Position'，[20 20 80 20]，...
        'String'， 'Black|Red|Yellow|Green|Cyan|Blue|Magenta|White'，...
        'Value'，1，...
       UserData'，[[0 0 0]；...
                   [1 0 0]；...
                   [1 1 0]；...
                   [0 1 0]；...
```

```
                                    [0 1 1]；...
                                    [0 0 1]；...
                                    [1 0 1]；...
                                    [1 1 1]；...
            CallBack'，[...
                'UD=get(Hc_fcolor，  '  'UserData'  ')；  '，...
                'set(gcf，  '  'Color'  '，UD(get(Hc_fcolor，  '  'Value'  '),：))
    ']);
```

弹出式菜单的 **'Position'** 属性含有熟悉的向量[**left bottom width height**]，其中宽度与高度决定了弹出对象的大小。在X Window和Macintosh系统中，就是被关闭的弹出式菜单的大小。打开时，菜单展开适合显示屏幕大小所有的选项。在MS-Windows系统中，高度值基本上被忽略，这些平台建立高度足够的弹出式菜单，显示一行文本而不管**height**的值。

框架

框架**uicontrol**对象仅是带色彩的矩形区域。框架提供了视觉的分隔性。在这点上，框架与**uimenu**的 **'Sepatator'** 属性相似。框架典型地用于组成无线按钮或其它**uicontrol**对象。在其它对象放入框架之前，框架应事先定义。否则，框架可能覆盖控制框使它们不可见。下面的例子（**mmct18.m**）建立了一个框架，把两个按钮和一个标志放入其中。

```
>>  Hc_frame  =  uicontrol(gcf,'Style'，'frame'，'Position'，[250 200 95 65])；

>>  Hc_pb1  =  uicontrol(gcf，  'Style'，  'pudhbutton'，...
                'Position'，[255 205 40 40]，  'String'，  'OK')；

>>  Hc_pb2  =  uicontrol(gcf，  'Style'，  'pushbutton'，...
                'Position'，[300 205 40 40]，  'String'，  'NOT')；


>>  Hc_1b1  =  uicontrol(gcf，  'Style'，  'text'，...
                'Position'，[255 250 85 10]，  'Str'，  'Push Me')；
```

控制框属性

如句柄图形对象建立函数一样，**uicontrol**属性可在对象建立时定义，或如上所示，用**set**命令来改
变。所有可设定的属性，包括字符串文本、回调串、甚至控制框函数类型都可以用**set**来改变。本章后面有若干例子。

表21.2列出了MATLAB 4.2版本中**uicontrol**对象的属性及其值。带有*的属性为非文件式的，使用时需加小心。由{}括起来的属性值是缺省值。

表21.2

Uicontrol 对象属性

BackgroundColor	**uicontrol**背景色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的背景色是浅灰色。
Callback	MATLAB回调串，当**uicontrol**激活时，回调串传给函数**eval**；初始值为空矩阵。
ForegroundColor	**uicontrol**前景（文本）色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的文本色是黑色。
HorizontalAlignment	标志串的水平排列
left：	相对于**uicontrol**文本左对齐
{**center**}：	相对于**uicontrol**文本居中
right：	相对于**uicontrol**文本右对齐
Max	属性 **'Value'** 的最大许可值。最大值取决于**uicontrol**的 'Type' 当**uicontrol**处于**on**状态时，无线按钮及检查框将**Value**设定为**Max**；该值定义了弹出式菜单最小下标值或滑标的最大值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为1
Min	属性 **'Value'** 的最小许可值。最小值取决于**uicontrol**的 **'Type' uicontrol**处于**off**状态时。无线按钮及检查框将**Value**设定为**Min**；该值定义了弹出式菜单最小下标值或滑标的最小值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为0
Position	位置向量**[left bottom width height]**。其中，**[left height]**表示相对于图形对象左下角的**uicontrol**的左下角位置。**[widht height]**表示**uicontrol**的尺寸大小，其单位由属性**Units**确定。
Enable*	控制框使能状态
{**on**}：	**uicontrol**使能。激活**uicontrol**，将**Callback**字符串传给
off：	**eval**
	unicntrol不使能，标志串模糊不清。激活**unicontrol**不起作用
String	文本字符串，在按钮键，无线按钮，检查框和弹出式菜单上指定**uicontrol**的标志。对于可编辑文本框，该属性设置成由用户输入的字符串。对弹出式菜单或可编辑文本框中多个选项或，每一项用垂直条(|)分隔，整个字符串用引号括起来。框架和滑标，不用引号

Style		定义**uicontrol**对象的类型
	{**pushbutton**}:	按钮键：选择时执行一个动作。
	radiobutton:	无线按钮键：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	checkbox:	检查框：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	edit:	可编辑框：显示一个字符串并可让用户改变
	text:	静态文本框：显示一个字符串
	slider:	滑标：让用户在值域范围内选择一个值。
	frame:	框架：显示包围一个或几个**uicontrol**的框架，使其形
	popumenu:	成一个逻辑群。
		弹出式菜单 ：含有许多互斥的选择的弹出式菜单
Units		位置属性值的单位
	inches:	英寸
	centimeters:	厘米
	normalized:	归一化的坐标值，图形的左下角映射为[0　0]而右上
	points:	角的映射为为[1　1]
	{**pixels**}:	打印设置点，等于1/72 英寸
		屏幕的象素。计算机屏幕分辨率的最小单位。
Value		**uicontrol**的当前值。无线按钮和检查框在 **'on'** 状态时，**value**设为**Max**，当是 **'off'** 状态时，**value**设为**Min**。由滑标将滑标的**value**设置为数值（**Min≤Value≤Max**），弹出式菜单把**value**值设置所选择选项的下标（**1≤Value≤Max**）。文本对象和按钮不设置该属性。
ButtonDownFcn		当**uicontrol**被选择时，MATLAB回调串传给函数**eval**。
		初始值为空矩阵
Children		**Uicontrol**对象一般无子对象，通常返回空矩阵
Clipping		限幅模式
	{**on**}:	对**uicontrol**对象无作用效果
	off:	对**uicontrol**对象无作用效果
DestroyFcn		只对Macintosh 4.2 版本。没有文件说明
Interrruptible		指定**ButtonDownFcn**和**CallBack**串是否可中断
	{**on**}:	回调不能由其它回调中断
	off:	回调串可被中断
Parent		包含**uicontrol**对象的图形句柄
*Select		值为**[on|off]**
*Tag		文本串
Type		只读对象辩识串，通常为**uicontrol**
UserData		用户指定的数据。可以是矩阵，字符串等等
Visible		**uicontrol**对象的可视性
	{**on**}:	**uicontrol**对象在屏幕上可见
	off:	**uicontrol**对象不可见，但仍然存在

控制框布置的考虑

uicontrol的属性 **'Position'** 和 **'Units'** 用于分配图形窗口中对象的位置。**uicontrol**的缺省 'Position' 向量为**[20 20 60 20]**，以象素表示。该值是缺省 **'Units'** 值。这是一个**60×20** 象素**uicontrol**的象素矩形，**uicontrol**左下角位于父图形左下角的靠右边20个象素点，靠上边20 个象素。缺省的图形尺寸大约为**560×420**个象素，位于显示屏的中上部。利用这些信息，**uicontrol**的布置就变成了一个2维几何布局问题。

要加若干限制。比如，MS-Window忽略位置向量高度值，仅有足够高度以显示一行文本。在所有其它情况下，必须保证控制框足够大以容纳控制框标志字符串。因为显示控制框 **'String'** 属性值所用的字体与用在命令窗口的字体一致，所以，用户对于标志每个控制框的字体属性无法控制，不同平台用不同字体，整体上可由用户改变。

通常确定控制框的大小和位置是一个尝试的过程。即使结果很满意，图形在另一个平台上的外观也许会很不一样，还需调整。通常希望把控制框制作得比所需要的更大一些，以便在所有平台上其标志都可读。

正因为图形有缺省尺寸，不能保证图形都具有缺省大小。若在现有的图形窗口内加上**uicontrol**和**uimenu**，则图形尺寸会比缺省值大或小。另外，用户可以在任何时候，对任何图形重调尺寸，除非把图形的 **'Resize'** 属性值置 **'off'** ，才可避免这样做。

当把**uicontrol**加到尺寸可能重新调整的图形时，有两点需要考虑：属性 **'Units'** 和由固定字体标志字符串所加的约束。若**uicontrol**的位置是以绝对单位指定如：**象素、英寸、厘米或点**，则窗口尺寸的重调不会影响**uicontrol**的大小和位置。**uicontrol**相对于图形左下角的位置不变；若图形变小，则某些**uicontrol**可能不可见。

若**uicontrol**的位置以归一化单位指定，则当图形尺寸调整时，**uicontrol**相互之间及与图形本身之间的关系保持不变。但是，使用归一化的单位有一个缺点，即如果图形变小，**uicontrol**对象大小也要调整，则**uicontrol**的标志可能看不见，因为字体大小是固定的。任何超出调整后的**uicontrol**的部分就被删去。希望以后MATLAB版本会给程序员对**uicontrol**和**uimenu** 的标记字符串的字体属性有更多的控制框。

M 文件举例

下面例子说明了本章所讨论的某些控制框的用法。精通MATLAB工具箱中的函数**mmclock**在屏幕上生成一个数字钟，它用任选参数设定钟的位置。它使用了框架、文本、无线按钮、检查框和按钮键**uicontrol**。

为了在PC上运行这个例子，在命令窗的**Option**菜单中选择**Enable Background Process**菜单项。这样将引起MATLAB进入无限循环和死锁。为了在Macintosh上得到较好的结果，关掉**Option**菜单的**Background Operation**检查标记。X Windows系统版本不需任何调整。

首先，建立函数的语句和帮助文本。

```
function T=mmclock(X，Y)
%MMCLOCK Place a digital clock on the screen.
% MMCLOCK places a digital clock at the upper-right corner
% of the display screen.
% MMCLOCK(X，Y) places a digital clock at position X pixels
% to the right and Y pixels above the bottom of the screen.
% T=MMCLOCK returns the current date and time as a string
% when  'Close'  is pressed.
```

设定初始值，分析输入参量。

```
fsize=[200 150]；  sec=1；  mil=0；
mstr=[ ' Jan ' ；  ' Feb ' ；  ' Mar ' ；  ' Apr ' ；  ' May ' ；  ' Jun '
        ' Jul ' ；  ' Aug ' ；  ' Sep ' ；  ' Oct ' ；  ' Nov ' ；  ' Dec ' ]；
scr=get(0，  ' ScreenSize ' )；

if nargin= =0
    figpos=[scr(3)-fsize(1)-20 scr(4)-fsize(2)-5 fsize(1：2)]；
elseif nargin= =2
    figpos={X Y fsize(1：2)]；
else
     error( ' Invalid Arguments ' )；
end
```

建立图形，在图中设置**uicontrol**为某些缺省值。

```
Hf_clock=figure( ' Position ' ，  figpos ' ，  ...
                ' Color ' ，[.7   .7   .7]，...
                ' NumberTitle ' ，  ' off ' ，  ...
                ' Name ' ，  ' Digital Clock ' )；
set(Hf_clock，  ' DefaultUicontrolUnits ' ，  ' normalized ' ，  ...
               ' DefaultUicontrolBackgroundColor ' ，  get(Hf_clock，  ' Color ' ))；
```

对秒建立**Close** 按钮键、无线按钮；对24小时建立检查框。

```
Hc_close=uicontrol( ' Style ' ，  ' push ' ，  ...
                ' Position ' ，[.65   .05   .30   .30]，...
                ' BackgroundColor ' ，[.8   .8   .9]，...
                ' String ' ，  ' Close ' ，  ...
                ' CallBack ' ，  ' close(gcf) ' )；

Hc_sec=uicontrol( ' Style ' ，  ' radiobutton ' ，  ...
                ' Position ' ，[.05   .05   .50   .13]，...
                ' Value ' ，  sec，...
                ' String ' ，  ' Seconds ' )；

Hc_mil=uicontrol( ' Style ' ，  ' checkbox ' ，  ...
                ' Position ' ，[.05   .22   .50   .13]，...
                ' Value ' ，  mil，...
                ' String ' ，  ' 24-Hour ' )；
```

对日期和时间显示创建框架和文本串。

```
Hc_dframe=uicontrol('Style', 'frame', 'Position', [.04 .71 .92 .24]);
Hc_date      =uicontrol('Style', 'text', 'Position', [.05 .72 .90 .22]);
Hc_tframe=uicontrol('Style', 'frame', 'Position', [.04 .41 .92 .24]);
Hc_time=uicontrol('Style', 'text', 'Position', [.05 .42 .90 .22]);
```

循环，直到按钮键回调函数关闭图形，每秒更新显示一次。

```
while find(get(0, 'Children') = = Hf_clock % Looop while clock exists
        sec = get(Hc_sec, 'Value');
        mil = get(Hc_mil, 'Value');
        now=fix(clock);
        datestr=sprintf('%s %2d %4d', mstr(now(2),:, now(3), now(1));
        if mil
                suffx=' ';
        else
                if now(4)>12
                        suffix='pm';
                        now(4)=rem(now(4), 12);
                else
                        suffix='am';
                end
        end
        timestr=[num2str9now(4)) ':' sprintf('%02d', now(5))];
        if sec
             timestr=[timestr ':  sprintf('%02d, now(6))];
        end
timestr=[timestr suffix];
        set(Hc_date.'String', datestr);
        set(Hc_time, 'String', timestr);
        pause(1)
end
```

最后，如果有需，返回日期和时间字符串。

```
if nargout
        T=[datestr '-' timestr];
end
```

虽然这个例子说明了如何用**uicontrol**建立一个完整的个GUI函数，但并不实用。因为该函数保持循环直至**Close**按钮按下，非等到时钟图形关闭，命令窗口才能使用。

注意，上例只含有一个回调函数字符串，即在按钮键回调串中的 **'close'** 语句。其它按钮值是在While循环中得到，而不是按钮本身用回调函数激发一个动作。更复杂的**M文件**需要复杂的回调。

21.5 编程和回调考虑

用户句柄图形界面的基础部分已经阐述过了，现在是应用它的时候了。正如所见，在命令行通过输入**uimenu**和**uicontrol**来建立效率不高。脚本或函数M文件使用更为简便。假定想实现一个M文件，首先确定是否要编写脚本文件或函数文件。

脚本与函数

脚本文件似乎是当然的选择。在脚本中，所有的命令都在命令工作窗口执行，因此随时可以使用所有的MATLAB函数和对象句柄。将信息传给回调函数无任何困难。然而这里有几种权衡。首先，当所有的变量都是可利用时，工作空间内充斥了变量名和变量值，即使它们已经不再有用。其次，如果用户使用**clear**命令，重要的对象句柄就可能丢失。另一个缺点是：用脚本文件定义回调字符串可能变得十分复杂。例如，以下有一个滑标的文件片段，节选自精通MATLAB工具箱中脚本文件**mmsetclr**

```
Hc_rsli=uicontrol(Hf_fig， 'Style'， 'slider'， ...
    'Positon'， [.10 .55 .35 .05]， ...
    'min'， 0， 'max'， 1， 'Value'， initrgb(1)， ...
    'CallBack'， [...
        'set(Hc_nfr，"BackgroundColor"， ' ， ...
          '[get(Hc_rsli，"Val")，get(Hc_gsli，"Val")，get(Hc_bsli，"Val")])， ' ， ...
        'set(Hc_ncur，"String"， ' ， ...
        'sprintf("[%.2f %.2f %.2f]"，get(Hc_nfr，"BakgroundColor")))， ' ， ...
        'hv=rgb2hsv(get(Hc_nfr，"BakgroundColor"))； ' ， ...
        'set(Hc_hsli，"Val"，hv(1))， ' ， ...
        'set(Hc_hcur，"String"，sprintf("%.2f"，hv(1)))， ' ， ...
        'set(Hc_ssli，"Val"，hv(2))， ' ， ...
        'set(Hc_scur，"String"，sprintf("%.2f"，hv(2)))， ' ， ...
        'set(Hc_vsli，"Val"，hv(3))， ' ， ...
        'set(Hc_vcur，"String"，sprintf("%.2f"，hv(3)))， ' ， ...
        'set(Hc_rcur，"String"，sprintf("%.2f"，get(Hc_rsli，"Val")))' ]);
```

另一个问题是，脚本文件比函数文件运行得慢，脚本在第一次运行时要编译。最后一点，脚本文件没有函数灵活。函数可接受输入参量并返回值。因此，函数可用作其它函数的参变量。

函数不会使命令窗工作空间拥挤；当反复调用时运行快速；接受输入参量并返回值；回调字符串书写不复杂。因此，在许多场合，函数M文件是最佳的选择。

考虑前面脚本文件中滑标定义的例子。以下是等价的文件片段，取自精通MATLAB工具箱中函数文件**mmsetc**。

```
Hc_rsli=uicontrol(Hf_fig， 'Style'， 'slider'， ...
        'Position'， [.10 .55 .35 .05]， ...
        'Min'， 0， 'Max'， 1， 'Value'， initrgb(1)， ...
```

```
'CallBack'， 'mmsetc(0，"rgb2new")')；
```

注意回调字符串以不同的参量调用**mmsetc**。这是一个在回调中使用递归函数调用的例子。然而函数本身有它自己的问题。主要的困难源于要将回调字符串传给**eval**并在**命令窗口工作空间中运行**，而其余的程序码在函数工作空间内执行。这里用前面章节所讨论的变量和函数的大量规则。在函数内定义的变量在命令窗口工作空间不存在，因此不在回调串中使用。同时在命令窗工作空间的变量在函数本身内部也不存在。

有若干既能解决该问题又能利用函数优点的方法。全局变量、**'UserData'** 属性、仅用于回调的特殊函数M文件和递归函数调用均是创建GUI函数十分有用的工具。

独立的回调函数

建立GUI函数的一个有效方法是编写独立的回调函数，专门执行一个或多个回调。函数使用的对象句柄和其它变量可以作为参量传递，必要时回调函数可返回值。

考虑先前的一个例子，建立一个方位角的滑标，以脚本文件来实现。

```
% setview.m script file

vw=get(gca， 'View')；

Hc_az=uicontrol(gcf， 'Style'， 'slider'，...
                'Position'，[10  5  140  20]，...
                'Min'，-90， 'Max'，90， 'Value'，vw(1)，...
                'Callback'，[...
                    'set(Hc_cur，'String'，num2str(get(Hc_az，'Value')))，'...
                    'set(gca， 'View'，[get(Hc_az， 'Value') vw(2)])'])；
Hc_min=uicontrol(gcf， 'style'， 'text'，...
                 'Position'，[10  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Min')))；

Hc_max=uicontrol(gcf， 'Style'， 'text'，...
                 'Position'，[110  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Max')))；

Hc_cur=uicontrol(gcf， 'Style'， 'text'，...
                 'Position'，[60  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Value')))；
```

下面是同样的例子。作为一个函数，采用 **'Tag'** 属性来辨别控制框，并使用独立的M文件来执行回调。

```
funtion setview( )

vw=get(gca， 'View')；
```

```
Hc_az=uicontrol(gcf，'Style'，'Slider'，...
                'Position'，[10  5  140  20]，...
                'Min'，-90，'Max'，90，'Value'，vw(1)，...
                'Tag'，'Azslider'，...
                'Callback'，'svcback')；

Hc_min=uicontrol(gcf，'style'，'text，...
                'Position'，[10  25  40  20]，...
                'String'，num2str(get(Hc_az，'Min')))；

Hc_max=uicontrol(gcf，'Style'，'text'，...
                'Position'，[110  25  40  20]，...
                'String'，num2str(get(Hc_az，'Max')))；

Hc_cur=uicontrol(gcf，'Style'，'text'，...
                'Position'，[60  25  40  20]，...
                'Tag'，'Azcur'，...
                'String'，num2str(get(Hc_az，'Value')))；
```

回调函数本身如下：

```
function svcback( )

vw = get(gca，'View')；

Hc_az = findobj(gcf，'Tag'，'AZslider')；
Hc_cur = findobj(gcf，'Tag'，'AZcur')；

str = num2str(get(Hc_az，'Value'))；
newview =[get(Hc_az，'Value') vw(2)]；
set(Hc_cur，'String'，str)
set(gca，'View'，newview)
```

上面的例子并不节省很多代码，但却得到了用函数而不用脚本文件的优点：回调函数可以利用临时变量，而不使命令窗口工作空间拥挤；不需要**eval**所需的引号和字符串；在回调函数中命令的句法变得十分简单。使用独立回调函数技术，越复杂的回调(函数)越简单。

独立回调函数的缺点是：需要很大数目的M文件以实现一个含有若干控制框和菜单项的GUI函数，所有这些M文件必须在MATLAB路径中可得，且每一个文件又必须要有一个不同的文件名。在对文件名大小有限制且对大小写不敏感的平台上，如MS-windows，文件冲突的机会就增加了。而且回调函数只能被GUI函数调用而不能被用户调用。

递归函数调用

利用单独的M文件并递归地调用该文件，既可以避免多个M文件的复杂性，又可以利用函数的优点。使用开关 **switches**或**if elseif**语句，可将回调函数装入调用函数内。通常这样一种函数调用的结构为

```
function guifunc(switch)。
```

其中**switch**确定执行哪一个函数开关的参量，它可以是字符串 **'startup'， 'close'， 'sectolor'** 等等，也可以是代码或数字。如**switch**是字符串，则可如下面所示的M文件片段那样将开关编程。

```
if nargin < 1， switch = 'startup'； end；
if ~isstr(switch)， error('Invalid argument')， end；
if strcmp(switch， 'startup')，
    <statement to create controls or menus>
    <statements to implement the GUI function>
elseif strcmp(switch， 'setcolor')，
    <statements to perform the Callback associated with setcolor>
elseif strcmp(switch， 'close')，
    <statements to perform the Callback associated with close>
end
```

如果是代码或字符串，开关也可以相同方式编程。

```
if nargin < 1， switch = 0； end；
if isstr(switch)， error('Invalid argument')， end；
if switch = = 0，
    <statements to create controls or menus>
    <statements to implement the GUI function>
elseif switch = =1，
    <statements to perform the Callback associated with setcolor>
elseif switch ==2，
    <statements to perform the Callback associated with close>
end
```

下面的例子说明了方位角滑标如何可作为单独的函数M文件来实现：

```
function setview(switch)

if nargin < 1， switch = 'startup'； end；
if ~isstr(switch)， error('Invalid argument.')； end；

vw = get(gca， 'view')； % This information is needed in both sections

if strcmp(switch， 'startup') % Define the controls and tag them
```

```
        Hc_az = uicontrol(gcf， 'Style'， 'slider'， ...
                    'Position'，[10 5 140 20]，...
                    'Min'，-90， 'Max'，90， 'Value' vw(1)，...
                    'Tag'， 'AZslider'，...
                    'Callback'， 'setview('set')');

        Hc_min=uicontrol(gcf， 'Style'， 'text'，...
                     'Position'，[10 25 40 20]，...
                     'String'，num2str(get(Hc_az， 'Min')));
        Hc_max = uicontrol(gcf， 'Style'， 'text'，...
                     'Position'，[110 25 40 20]，...
                     'String'，num2str(get(Hc_az， 'Max')));

        Hc_cur =uicontrol(gcf， 'Style'， 'text'，...
                    'Position'，[60 25 40 20]，...
                    'Tag'， 'AZcur'，...
                    'string'，num2str(get(Hc_az， 'Value')));

    elseif strcmp(switch， 'set') % Execute the Callback

        Hc_az=findobj(gcf， 'Tag'， 'AZslider');
        Hc_cur-findobj(gcf， 'Tag'， 'AZcur');

        str = num2str(get(Hc_az， 'Value'));
        newview - [get(Hc_az， 'Value') vw(2)];

        set(Hc_cur， 'String'， str)
        set(gca， 'View'，newview)
    end
```

上述的两个例子均设置了**'tag'**属性，利用该属性和函数**findobj**寻找回调函数所需对象的句柄。另外两种方法将在下章描述。

全局变量

全局变量可用在函数中，使某些变量对GUI函数的所有部分都可用，全局变量是在函数的公共区说明，因此整个函数以及所有对函数的递归调用都可以利用全局变量，下面的例子说明如何利用全局变量将方位角滑标编程。

```
    function setview(switch)

    global HC_AZ HC_CUR % Create global variables
```

```
if nargin < 1， switch = ' startup ' ； end；
if ~isstr(switch， error( ' Invalid argument. ' )； end；

vw = get (gca， ' View ' )； % This information is needed in both sections
if strcmp(switch， ' startup ' ) % Define the controls

    Hc_AZ=uicontrol(gcf， ' style ' ， ' slider ' ，...
                ' Position ' ，[10 5 140 20]，...
                ' Min ' ，-90， ' Max ' ，90， ' Value ' ，vw(1)，...
                ' Callback ' ， ' setview( ' set ' ) ' )；

    Hc_min=uicontrol(gcf， ' Style ' ， ' text ' ，...
                ' Position ' ，[10 25 40 20]，...
                ' String ' ，num2str(get(Hc-AZ， ' Min ' ，)))；

    HcMax=uicontrol(gcf， ' Style ' ， ' text ' ，...
                ' Position ' ，[110 25 40 20]，...
                ' String ' ，num2str(get(Hc_AZ， ' Max ' )))；

    Hc_cur=uicontrol(gcf， ' style ' ， ' text ' ，...
                ' Position ' ，[60 25 40 20]，...
                ' String ' ，num2str(get(HC_AZ， ' Value ' )))；

elseif strcmp(switch， ' set ' ) % Execute the Callback

      str=num2str(get(HC_AZ， ' Value ' ))；
      newview= [get(HC_AZ， ' Value ' ) vw(2)]；

      set(HC_CUR， ' String ' ，str)
      set(gca， ' View ' ，newview)

end
```

全局变量遵循MATLAB的规定，变量名要大写。不需要**'tag'**属性，且不使用它，另外因为对象句柄存在的，不需要用函数**findobj**去获取，故回调代码比较简单，全局变量通常使一个函数更有效。

不过有一点要注意，尽管一个变量在函数内说明为**全局**的，变量并不能自动地在命令窗口工作空间中利用，也不能在回调字符串内使用。但是，如果用户发命令：**>> clear global**，则所有全局变量则都被破坏，包括在函数内定义的那些变量。

当单独的一个图形或有限个变量要被所有的回调(函数)利用时，全局变量使用和递归性函数调用都是有效的技术。对于包含多个图形的更复杂的函数，或用独立对象回调函数实现的情况，**'UserData'**属性更合适。另外，只要可获得对象句柄，对象**'UserData'**的属性值在命令窗口工作空间中是存在的。

用户数据属性

同属性 **'Tag'** 一样， **'UserData'** 属性可在函数之间或递归函数的不同部分之间传递信息。如果需要多个变量，这些变量可以在一个容易辨识的对象的属性 **'UsetData'** 中传递。**如前面所述， 对与句柄图形对象在一起的单个数据矩阵'UserData' 提供了存储**。下面的程序利用了当前图形的 **'UserData'** 属性来实现方位角滑标。

```
function setview(switch)

if nargin < 1， switch = 'startup'； end；

vw = get(gca， 'View')； % This information is needed in both sections

if strcmp(switch， 'startup') % Define the controls

    Hc_az = uicontrol(gcf， 'Style'， 'slider'， ...
                'Position'，[10 5 140 20]，...
                'Min'，-90， 'Max'，90， 'Value'，vw(1)，...
                'Callback'， 'setview("set")')；

    Hc_min = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[10 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Min')))；

   Hc_max = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[110 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Max')))；

   Hc_cur = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[60 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Value')))；

   set(gcf， 'UserData'，[Hc_az Hc_cur])； % Store the object handles

elseif strcmp(switch， 'set') % Execute the Callback

   Hc_all = get(gcf， 'UserData')； % retrieve the object handles
   Hc_az = Hc_all(1)；
   Hc_cur = Hc_all(2)；

   str = num2str(get(Hc_az， 'Value'))；
   newview = [get(Hc_az， 'Value') vw(2)]；
   set(Hc_cur， 'String'， str)
   set(gca， 'View'， newview)
```

```
end
```

句柄存储于 **'startup'** 末端，图形 属性**'UserData'** 中，在回调被执行前对此进行检索。如果有许多回调，如下面的程序片断所示， **'UserData'** 只需检索一次。

```
if strcmp(switch， 'startup') % Define the controls and tag them

    % <The 'startup' code is here>

    set(gcf， 'UserData' ，[Hc_az Hc_cur])； % Store the object handles

else      % This must be a Callback

     Hc_all=get(gcf， 'UserData')； % Retrieve the object handles
     Hc_az=Hc_all(1)；
     Hc_cur=Hc_all(2)；

     if strcmp(switch， 'set')

          % <The 'set' Callback code is here>

     elseif strcmp (switch， 'close')

          %<The 'close' Callback code is here>
     % <Any other Callback code uses additional elseif clauses>
     end
end
```

调试GUI M文件

回调字符串在命令窗口工作空间中计算并执行的，这个情况对编写和调试GUI函数和脚本文件有某种隐含意义。回调字符串可以很复杂，尤其是在脚本文件中，这为句法错提供了许多机会，记录单引号、逗号、括号是令人头痛的事。如果出现了句法错误，MATLAB给出提示；只要对象的 **'Callback'** 属性值是一个真正的文本串，MATLAB就认可了。只有当对象被激活并将回调字符串传给**eval**时，才检查回调字符串内部的句法错误。

这样让用户定义回调字符串，它涉及还未曾定义过的对象句柄和变量，这使编写相互参照的程序变得更容易，但是每个回调函数必须分别测试，保证回调字符串是合法的MATLAB命令，并且回调字符中涉及的所有变量可在命令窗口工作空间中是可利用的。

将回调象函数M文件一样编程或象GUI函数本身内的开关一样编程，就可以不运行整个GUI函数而对各个回调进行改变或测试。

因为回调字符串是在命令窗工作空间中而不在函数本身内计算，在函数与各回调之间传递数据就变得十分复杂。例如，函数test包含如下程序：

```
function test()
```

```
tpos1=[20   20   50   20];

tpos2=[20   80   50   20];

Hc_text=uicontrol('Style','text','String','Hello','Position',tposl);

Hc-push=uicontrol('Style', 'push', 'String', 'Move Text', ...
  'Position',[15   50   100   25], ...
  'Callback', 'set(Hc-text,"Position",tpos2)');
```

所有语句都是有效的MATLAB命令，且回调字符串也对有效的MATLAB语句估值。文本对象和按钮出现在图形上，但当激活按钮键时，MATLAB就出示错误。

```
>>  test
>>
??? Undefined functiion or variable Hc_text.

??? Error while evaluating Callback string.
```

如果**test**是个脚本文件，就不会出现这样问题，因为所有变量可在命令窗口工作空间中使用，因为**test**是个函数，**Hc_text**和**tpos2**在命令窗口工作空间中均未定义，回调字符串执行失败。

一种解决方法是使用各个字符串元素来建立回调，该字符串元素由数值而非变量建立，例如，改变回调字符串如下：

```
'Callback', ['set9', ...
        sprintf('&.15g', Hc_text), ...
        ',  ' 'Position' ',  ', ...
        sprintf('[%.15g   %.15g   %.15g   %.15g]', tpos2), ...
        ')']);
```

建立了包括**Hc_text**对象句柄值的一个字符串，该值变换成具有15位精度的字符串，而**tpos2**变量转换成矩阵表示的字符串。在函数内计算**sprinf**语句，然后将所得的字符串用在回调中。在命令窗工作空间执行的实际命令如下所示

```
eval('set(87.000244140625,  ' 'Position' ', [20   80   50   100])')
```

将一个对象句柄转换为字符串，必须保持全精度。上例中的变换，使用了小数点后15位的数字的精度。在MATLAB中句柄对象转换应使用这样的精度。

要记住，变量随后的变换不会改变回调字符串。在前面的例子中，在控制框定义之后改变**tpos2**的值，就无效果。例如，在函数结尾处加命令

tpos2=[20　200　50　20]

就无效果，因为在**tpos2**重新定义之前，通过计算**tpos2**，回调字符串已经建立。

21.6　指针和鼠标按钮事件

GUI函数利用鼠标箭头的位置和鼠标按钮的状态来控制MATLAB行动。本节讨论指针、对象位置和鼠标按钮动作之间的交互，以及MATLAB如何响应变化或事件，诸如：揿下按钮、松开按钮或箭头移动等。

回调属性，选择区域和堆积顺序

所有句柄图形对象具有一个至今还未阐述过的 **'ButtonDownFcn'** 属性，本节就进行讨论。**uimenu**和**uicontrol**均有 **'CallBack'** 属性，这个属性是菜单和控制框的应用核心。另外，图形有 **'KeyPressFcn'** 和 **'ResizeFcn'** 属性以及 **'WindowButtonUpFcn'** 和 **'WindowButtonMotionFcn'** 属性。与这些属性相关的值是回调字符串，即MATLAB字符串，当属性激活时，它传给**eval**。指针的位置确定事件涉及到哪些回调以及当事件发生时它们被激励的顺序。

前一章讨论过堆积顺序和与讨论相关的对象选择区域。根据图形中3个区域，MATLAB决定哪一个回调将被激励。如果指针是在句柄图形对象内，如同对象的 **'Position'** 属性所确定的那样，则指针就是在对象**上**。如果指针不在一个对象上，而在对象的选择区域内，则指针**靠近**对象。如果指针在图形内但既不靠近也不在另一对象之上，则可以认为指针关掉其它对象。如果若干对象及其选择区域相重叠，重叠顺序就决定了选择顺序。

句柄图形的线、曲面、补片、文本和坐标轴对象的选择区域已在上一章讨论过了。**uimenu**对象没有外部的选择区域，指针要么在**uimenu上**，要么不在其上。**uicontrol**对象越过图基位置，在各方向延伸大约5个象素有一个选择区域，指针可以**靠近**或在控制框**上**。记住这里讨论的堆积顺序和选择区域是针对4.2c版本的，在以后的MATLAB有某种程度的变化。

按钮点击

按钮点击可以定义为当鼠标指针在同一对象上时，揿下并随后松开鼠标按钮。如果鼠标指针在**uicontrol**或**uimenu**项上，按钮点击触发对象 **'Callback'** 属性字符串的执行，按钮揿下使控制框作好准备，并常常在视觉上改变**uicontrol**或**uimenu**，松开按钮触发回调。当鼠标指针不在**uicontrol**或**uimeun**上，**揿下按钮**和**松开按钮**事件，将在后面讨论。

揿下按钮

当鼠标指针位于一个图形窗口内，揿下鼠标按钮，根据指针位置和对句柄图形对象靠近程度将会发生不同的动作。如果一个对象被选择，它就变成了当前的对象，并上升到堆积顺序的最高端。如果没有对象被选择，图形本身就是当前对象，图形的 **'CurrentPoint'** 和 **'SelectionType'** 属性都被更新，然后激发适当的回调。表格21.3列出了指针位置与按钮揿下所激发的回调：

表21.3

指针位置	所激发的属性
在**uimenu** 或**uicontro**项上面	准备释放事件
在句柄图形**上**，或**接近**控制框	图形的**WindowButtonDownFcn**属性，然后是对象的**ButtonDownFcn**属性
在图形内，但不**在**或**接近**任何对象	图形的**WindowButtonDownFcn**属性，然后是图形的**ButtonDownFcn**属性

注意：按钮揿下事件总是在所选对象的 **'ButtonDownFcn'** 回调之前，引起图形的 **'WindowButtonDownFcn'** 回调，除非指针是在**uicontrol**或**uimenu**对象上。如果指针靠近控制框，则在图形的 **'WindouButtonDownFcn'** 回调完毕后，引起控制框的 **'ButtonDownFcn'** 回调而不是 **'Callback'** 属性回调。

按钮松开

当松开鼠标按钮时，图形的 **'CurrentPoint'** 属性就更新。如果没有定义 **'WindowButtonUpFcn'** 回调，则鼠标按钮松开时，属性 **'CurrentPoint'** 不更新。

指针的移动

当指针在图形内移动时，图形的 **'CurrentPoint'** 属性被更新，引起图形的 **'WindowButtonMotionFcn'** 回调；如果没有定义图形的 **'WindowButtonMotionFcn'** 回调，则指针移动时，属性 **'CurrentPoint'** 不更新。

回调的复合能产生许多有趣的效应。试一下包含在MATLAB于**demo**目录中的函数**sigdemo1**，**sigdemo2**，就可解其中的一些效果。另外将要讨论的一个例子是 精通MATLAB工具箱的函数**mmdraw**。

21.7 中断回调的规则

一旦回调开始执行，通常都在下一个回调事件处理之前运行完毕。将 **'Interruptible'** 属性设置为 **'yes'** 可改变这种缺省行为，从而当正在执行的回调遇到**drawnow**，**figure**，**getframe**或**pause**命令时，允许处理的回调事件悬挂起来。事件队列执行计算操作或设置对象属性的命令一经发出，MATLAB便进行处理；而涉及图形窗口输入或输出的命令则生存事件。事件包括产生回调的指针移动和鼠标按钮动作，以及重新绘制图形的命令。

回调处理

回调在达到**drawnow**，**getframe**，**pause**或**figure**命令之前一直执行。注意**gcf**和**gca**引起**figure**命令，而精通MATLAB工具箱中的函数**mmgcf**和**mmgca**不引发**figure**命令。不含有这些特殊命令的回调不会被中断，一旦达到这些特殊命令之一，停止执行回调，将其悬挂起来；并检查事件队列中每一个悬挂的事件。如果产生悬挂回调的对象的**Interruptible**属性设为 **'yes'**，则在被悬挂的回调恢复之前按序处理所有悬挂。如果**Interruptible**属性设为 **'no'**，即缺省值，则只处理悬挂的重画事件，放弃回调事件。

防止中断

即使正在执行回调是不能被中断的，当回调达到**drawnow**，**figure**，**gefframe**或**pause**命令时，仍然处理悬挂的重画事件。通过避免在回调中使用所有这些特殊命令，消除此类事情。如果回调中需要这些特殊命令，但又不要任何悬挂事件，甚至是刷新事件，来中断回调，则可以使用如下所讨论的特殊形式的**drawnow**。

Drawnow

Drawnow命令迫使MATLAB更新屏幕，只要MATLAB回到命令提示，或执行**drawnow**，**figure**，**getframe**或**pause**命令，屏幕就更新。**drawnow**的特殊形式**draunow(' discard ')**使事件队列中所有事件的放弃。在回调中将**drawnow(' discard ')**包含在一个特殊命令之前，就含有清除事件队列的效果，防止刷新事件，以及回调事件中断回调。

21.8 M文件举例

精通MATLAB工具箱中一些函数阐明了上面所讨论的若干技术。第一个例子**mmview3d**应用全局变量和递归函数调用，把方位角和仰角滑标加到图形中。函数中有大量的对象，但函数很直观。因为**mmview3d**文件相当得长，故分段表示。第一段定义了函数标号，帮助文本和全局变量。

```
function mmview3d(cmd)
%    MMVIE3D GUI controllled Azimuth and Elevation adustment.
%    for adjusting azimuth and elevation using the mouse.
%
%    The    ' Revert '    pushbutton reverts to the original view.

%    The    ' cmd '    argument executes the callbacks.

%    Copyright (c) 1996 by Prentice-Hall，    Inc.

global Hc_acur Hc_esli Hc_ecur CVIEW
```

第二段处理初始用户的调用，建立必要的**uicontrol**对象并把回调定义为递归函数调用。

```
if    nargin==0

%--------------------------------------------------------------------
%    Assign a handle to the current figure window.
%    Get the current view for slider initial values.
%    Use normalized uicontrol units rather than the default    ' pixels ' .
%--------------------------------------------------------------------
```

```
Hf_fig=gcf;
CVIEW=get(gca， 'View');
if abs(CVIEW(1))>180， CVIEW(1)=CVIEW(1)-(360*sign(CVIEW(1)))；end
set(Hf_fig， 'DefaultUicontrolUnits' ， 'normalized');

%--------------------------------------------------------------------
%  Define azimuth and elevation sliders.
%  The position is in normalized units  (0-1).
%  Maximum， minimum， and initial values are set.
%--------------------------------------------------------------------

Hc_asli=uicontrol(Hf_fig， 'style' ， 'slider' ， ...
 'position' ，[.09  .02  .3  .05]，...
 'min' ，-180， 'max' ，180， 'value' ， CVIEW(1)，...
 'callback' ， 'mmview3d(991)');

Hc_esli=uicontrol(Hf_fig， 'style' ， 'slider' ，...
 'position;，[.92  .5  .04  .42]，...
 'min' ，-90， 'max' ，90， 'val' ，CVIEW(2)，...
 'callback' ， 'mmview3d(992)');

%--------------------------------------------------------------------
%  Place the text boxes showing the minimum and maximum values at the
%  ends of each slider， These are text displays， and cannot be edited.
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.02  .02  .07  .05]，...
 'string' ，num2str(get(Hc_asli， 'min')));

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.39  .02  .07  .05]，...
 'string' ，num2str(get(Hc_esli， 'min')));

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.915  .92  .05  .05]，...
 'string' ，num2str(get(Hc_esli， 'max')));

%--------------------------------------------------------------------
% Place labels for each slider
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style' ， 'text' ，...
```

```
 'pos'，[9.095  .08  .15  .05]，...
 'string'， 'Azimuth')；

uicontrol(Hf_fig， 'style'， 'text'，...
 'pos'，[.885  .39  .11  .05]，...
 'string'， 'Elevation')；

%--------------------------------------------------------------------
%  Define the current value text displays for each slider，
%--------------------------------------------------------------------
%  These are editable text display the current value
%  of the slider and at the same time allow the user to enter
%  a value using the keyboard.
%
%  Note that the text is centered on XWindow Systems，but is
%  left-justified on MS-Windows and Macintosh machines.
%
%  The initial value is found from the value of the sliders.
%  When text is entered into the text area and the return key is
%  pressed， the callback string is evalueated.
%--------------------------------------------------------------------

Hc_acur=uicontrol(Hf_fig， 'style'， 'edit'，...
 'pos'，[.25  .08  .13  .053]，...
 'string'，num2str(get(Hc_asli， 'val'))，...
 'callback'， 'mmview3d(993)')；

Hc_ecur=uicontrol(Hf_fig， 'style'， 'edit'，...
 'pos'，[.885  .333  .11  .053]，...
 'string'，num2str(get(Hc_esli， 'val'))，...
 'callback'， 'mmview3d(994)')；

%--------------------------------------------------------------------
%  Place a  'Done'  button is the lower right corner.
%  When clicked， all of the uicontrols eill be erased.
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style'， 'push'，...
 'pos'，[.88  .02  .10  .08]，...
 'backgrouncolor'，[.7  .7  .8]，...
 'string'， 'Done'，...
 'callback'， 'delete(finobj(gcf， '  'Type'  '， '  'uicontrol'  '))')；

%--------------------------------------------------------------------
```

```
%   Place a  'Revert'  button next to the  'Done'  button.
%   When clicked，  the view reverts to the original view.
%---------------------------------------------------------------------

Hc_ecur=uicontrol(Hf_fig，  'style'，  'edit'，...
 'pos'，[.77   .02   .10   .08]，...
 'backgroundcolor'，[.7   .7   .8]，...
 'string'，  'Revert'，...
 'callback'，  'mmview3d(995)')；
```

现在已建立了控制框并定义了回调。

```
else

%---------------------------------------------------------------------
%   The callbacks for the azimuth and elevation sliders；
%---------------------------------------------------------------------
%   1)   get the value of the slider and display it in the text windows
%   2)   set the  'View'  property to the current values of the azimuth
%         and elevation sliders.
%---------------------------------------------------------------------

if   cmd==991
     set(Hc_acur，  'string'，  num2str(get(Hc_asli，  'val')))；
     set(gca，  'View'，  [get(Hc_asli，  'val')，get(Hc_esli，  'val')])；

elseif   cmd==992
     set(Hc_ecur，  'string'，  num2str(get(Hc_esli，  'val')))；
     set(gca，  'View'，  [get(Hc_asli，  'val')，get(Hc_esli，  'val')])；

%---------------------------------------------------------------------
%   The  'slider current value'  text display callbacks；
%---------------------------------------------------------------------
%   When text is entered into the text area and the return key is
%   pressed，  the entered value is compared to the limits.
%
%   If the limits have been exceeded，  the display is reset to the
%   value of the slider and an error message is displayed.
%
%   If the value is within the limits，  the slider is set to the
%   new value，  and the view is updated.
%---------------------------------------------------------------------

elseif cmd==993
```

```
        if str2num(get(Hc_acur， 'string')) < get(Hc_asli， 'min)...
          | str2num(get(Hc_acur， 'string')) >get(Hc_asli， 'max')
              set(Hc_acur， 'string'， num2str(get(Hc_asli， 'val')))；
              disp('ERROR - Value out of range')；
        else
            set(Hc_asli， 'val'， str2num(get(Hc_acur， 'string')))；
            set(gca， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；
        end
    elseif cmd==994
        if str2num(get(Hc_ecur， 'string')) < get(Hc_esli， 'min)...
          | str2num(get(Hc_ecur， 'string')) >get(Hc_esli， 'max')
              set(Hc_ecur， 'string'， num2str(get(Hc_esli， 'val')))；
              disp('ERROR - Value out of range')；
        else
            set(Hc_esli， 'val'， str2num(get(Hc_ecur， 'string')))；
            set(gca， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；
        end

    %-----------------------------------------------------------------
    %   Revert to the original view.
    %-----------------------------------------------------------------

    elseif cmd==995
        set(Hc_asli， 'val'， CVIEW(1))；
        set(Hc_esli， 'val'， CVIEW(2))；
        set(Hc_acur， 'sting'， num2str(get(Hc_asli， 'val')))；
        set(Hc_ecur， 'sting'， num2str(get(Hc_esli， 'val')))；
        set(Hc_acur， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；

    %-----------------------------------------------------------------
    %   Must be bad arguments.
    %-----------------------------------------------------------------

      else
        disp('mmview3d： Illegal argument.')
      end
    end
```

第二个例子**mmcxy**，在图形左下角建立一个小文本框，当指针处在图形中时，显示图形内鼠标指针的坐标**[x，y]**。点击鼠标清除坐标的显示。

虽然**mmcxy**是一个短小的函数，它仍然利用许多有效的GUI函数的元素，包括递归函数调用、全局变量和 **'UserData'** 属性。此例还说明图形的 **'WindowButtonDownFcn'** 和 **'WindowButtonMotionFcn'** 属性起动回调的用法。

```
function out=mmcxy(arg)
%   MMCXY Show x-y Coordinates of the mouse in the
%   lower left hand corner of the current 2-D figure window.
%   When the mouse is clicked，  the coordinates are erased.
%   XY=MMCXY returns XY=[x，y] coordinates where mouse was clicked.
%   XY=MMCXY returns XY=[] if a key press was used.

%   Copyright (c) by Prentice-Hall，  Inc.

global MMCXY_OUT

if  -nargin
      Hf=mmgcf;
      if  isempty(Hf)，  error('No Figure Available.')，  end
      Ha=findobj(Hf，  'Type'，  'axes');
      if  isempty(Ha)，  error('No Axes in Current Figure，  ')，end.

      Hu=uicontrol(Hf，  'Style'，  'text'，  ...
                          'units'，  'pixels'，  ...
                          'Position'，[1   1   140   15]，...
                          'HorizontalAlignment'，  'left');
      set(Hf，  'Pointer'，  'crossh'，  ...
            'WindowButtonMotionFcn'，  'mmcxy('  'move'  ')，...
            'WindowButtonDownFc'，  'mmcxy('  'end'  ')'，  ...
            'Userdata'，Hu)
      figure(Hf)   %bring figure forward
      if nargout   %must return x-y data
            key=waitforbuttonpress；    %   pause until mouse is pressed
            if  key
                  out=[]；              %return empty if aborted
                  mmcxy('end')     %clean thins up
            else
                  out=MMCXY_OUT；   %now that move is complete return point
            end
            return
      end

elseif strcmp(arg，  'move')          %   mouse is moving in figure window
      cp=get(gca，  'CurrentPoint')；     %   get current mouse position
      MMCXY_OUT=cp(1，1：2)；
      xystr=sprintf('[%.3g]'，MMCXY_OUT)；
      Hu=get(gcf，  'Userdata')；
      set(Hu，  'String'xystr)   %   put x-y coordinaates in text box
```

```
elseif strcmp(arg，  'end')  %  mouse click occurred，  clean things up
    Hu=get(gcf，  'Userdata')；
    set(Hu，  'visible'，  'off')  %  make sure text box disappears
    delete(Hu)
    set(gcf，  'Pointer'，  'arrow'，...
            'WindowButtonMotionFcn'，  ''，...
            'WindowButtonDownFcn'，  ''，...
            'Userdata'，[])
end
```

第一次被调用时，**mmcxy**建立文本**uicontrol**，改变指针形状，设定 **'WindowButtonDownFcn'** 和 **'WindowButtonMotionFcn'** 的回调，然后等待按键或先撤按钮。若有键撤下，就调用清除（cleanup）程序，清除文本框，恢复鼠标指针，清除图形回调及 **'UserData'** 属性。若点击鼠标按钮，**'WindowButtonDownFcn'** 回调就处理清除任务。在等待时，图形中鼠标指针的移动会触发 **'WindowButtonMotionFcn'** 回调，更新uicontrol中文本串。

精通MATLAB工具箱中的另一个函数是**mmtext**，它利用 **'WindowButtonDownFcn'** ，**'WindowButtonUpFcn'** 和 **'WindowButtonMotionFcn'** 回调的另一个短小函数以安置和拖曳文本。

```
function mmtext(arg)
%  MMTEXT place and drag text with mouse
%  MMTEXT waits for a mouse click on a text object
%  in the current figure then allows it to be dragged
%  while the mouse button remains down.
%  MMTEXT('whatever') places the string  'whatever'  on
%  the current axes and allows it to be dragged.
%
%  MMTEXT becomes inactive after the move is complete or
%  no text object is selected.

%  Copyright (c) 1996 by Prentice-Hall，  Inc.

if -nargin，  arg=0；end

if isstr(arg) % user entered text to be placed
    Ht=text('units'，  'normalized'，...
        'Position'，[.05  .05]，...
        'String'，arg，...
        'HorizontalAlignment'，  'left'，...
        'VerticalAlignment'，  'baseline')；
    mmtext(0)  %  call mmtext again to drag it

elseif arg==0 %  initial call，  select text for dragging
    Hf=mmgcf；
```

```
if isempty(Hf),    error('No Figure Available.'),  end
set(Hf, 'BackingStore', 'off', ...
    'WindowButtonDownFcn', 'mmtext(1)')
figure(Hf)  %  bring figure forward

elseif arg==1  &  strcmp(get(gco, 'Type'), 'text')  % text object selected
set(gco, 'Units', 'data', ...
            'HorizontalAlignment', 'left', ...
```

mmdraw是精通MATLAB工具箱中另一个有用GUI函数，它与**mmtext**十分相似，但更为复杂，此函数允许用户用鼠标在当前的坐标轴上画线并设置线的属性。

```
function mmdraw(arg1，arg2，arg3，arg4，arg5，arg6，arg7)
%MMDRAW Draw a Line and Set Properties Using Mouse.
%MMDRAW Draw a Line in the current axes using to the mose.
%Click at the starting point and drag to the end point
%In addition，properties can be given to the line.
%Properties are given in pairs，e.g.，MMDRW name vale...
%Properties：
%NAME              VALUE          (defult)
%color               [y m c r g b {w} k] or an rgb in quotes：   '[r g b]
%style               [-----{：}--.]
%mark                [0 + . * x]
%width               points for linewidth {0.5}
%size                 points for marker size {6}
%Examples：
%MMDRAW color r width 2 sets color to red and width to 2 points%MMDRAW mark
+ size 8   set marker type to +and size to 8 points
%MMDRAW color  '[1 5 0]'  set color to orange

%Copyright (c) 1996 by Prentice-Hall，Inc.

global MMDRAW_Hl MMDRAW_EVAL

if nargin==0
        arg1='color'；arg2='w'；arg3='style'；arg4='：  '；nargin=4；
end

if isstr(arg1) % initial call，set thing up
        if isempty(Hf)，error('No Figure Available.')，end
        Ha=findobj(Hf,  'Type,  'axes')；
        if isempty(Ha)，error('No Axes in Current Figure.')，end
        set(Hf,  'Pointer',  'Crossh',  ...%set up callback for line star
                'BackingStore',  'off',  ...
```

```
                    'WindowButtonDownFcn', 'mmdraw(1)')
        figure(Hf)
        MMDRAW_EVAL='mmdrw(99);%set up string to set attributes
        for i=1:nargin
            argi=eval(sprintf('arg%.of', i));
            MMDARW_EVAL=[MMDARW_EVAL ',' 'arg' ' ' ' '];
        end
        MMDARW_EVAL=[MMDARW_EVAL ,')'];

elseif arg1==1 % callback is line start point
        fp=get(gca, 'CurrentPoint');        %start point
        set(gca, 'Userdata', fp(1, 1:2))    %store in axes userdata
        set(gcf, 'WindowButtonMotionFcn', 'mmdraw(2)', ...
                    '          'WindowButtonUpFcn', 'mmdraw(3)')

elseif arg1==2 % callback is mouse motion
        cp=get(gca, 'CurrentPoint'); cp=cp(1, 1:2);
        fp=get(gca, 'Userdata');
        H1=line('Xdata', [fp(1); cp(1)], 'Ydata', [fp(2); cp(2)], ...
                    'EraseMode', 'Xor', ...
                    'Color', 'w', 'LineStyle', ':', ...
                   "Clipping', 'off';
        if ~isempty(MMDRAW_HL) % delete prior line if it exsts
            delete(MMDRAW_HL)
        end
        MMDRAW_HL=H1;  % store current line handle

elseif arg1==3 % callback is line end point, finish up
        set(gcf, 'Point', 'arrow', ...
                    'BackingStore', 'on', ...
                    'WindowButtonDownFcn', '', ...
                     'WindowButtonMotionFcn', 'mmtext(2)', ...
                     'WindowButtonUpFcn', '')
        set(gca, 'Userdata', [])
        set(MMDRAW_HL, 'EraseMode', 'normal') % render line better
        eval(MMDRAW_EVAL)
        MMDRAW_EVAL=[];

elseif arg1==99 % process line properties
        for i=2:2:nargin-1
            name=eval(sprintf('arg%.of', i), []);  % get name argument
            vale=eval(sprintf('arg%.of', i+1), []);  % get value argument
            if strcmp(name, 'color')
                if value(1)=='[', value=eval(value); end
```

```
                set(MMDRAW_HL， 'color'， value)
            elseif strcmp(name， 'style')
                set(MMDRAW_HL， 'Lineatyle'， value)
            elseif strcmp(name， 'mark')
                set(MMDRAW_HL， 'Linestyle'， value)
            elseif strcmp(name， 'width')
                 value=abs(eval(value))；
                set(MMDRAW_HL， 'LineWidth'， value)
            elseif strcmp(name， 'size')
                 value=abs(eval(value))；
                set(MMDRAW_HL， 'MarkerSize' ''， value)
            end
        end
        MMDRAW_HL=[]；
    end
```

虽然这里说明太长，但精通MATLAB工具箱中的函数**mmsetc**和**mmsetf**都是使用递归、全局变量和 '**UserData**' 属性的GUI函数的直观例子。也许愿意看一下**mmsetclr M**文件，它是函数**mmsetc**文件的脚本M文件。比较这两个文件，可以了解到，为实现各种GUI M文件要做出一些折衷。

21.9 对话框和请求程序

MATLAB具有建立对话框和 ' 请求 ' 的几个有用工具。对话框是弹出显示的单独窗口，它显示信息字符串。对话框含有一个或多个按钮键以供用户输入。请求框是在弹出显示的单独窗口，利用鼠标或键盘获得输入，并返回信息给调用函数。

对话框

所有MATLAB的对话框都是基于函数**dialog**，它的帮助文本如下

```
>>help diolog
DIALOG displays a dialog box.

    FIG = DIALOG (pl ， vl....)displays a dialog box.
    valid param/value pairs include
    Style               error  |  warning  |  help  |  question
    Name                string
    Replace        on  |  off
    Resize         on  |  off
    BackgroundColor  ColorSpec
    ButtonString        'Button1String  |  Button2String  |  ...  '
    ButtonCalls         'ButtonCallback  |  Button2Callback  | ...  '
```

ForegroundColor ColorSpec
Position [x y width height]
[x y] - centers around screen point
TextString string
Units pixels | normal | cent | inches | points
UserData matrix

Note： Until dialog becomes built-in， set and get
are not valid for dialong objects.
At most three buttons are allowed.
The callbacks are ignored for"question"dialogs.
If ButtonStrings /ButtonCalls are unspecified then it
defaults to a single "ok" button which removes the figure.
There ' s still problems with making the question dialog modal.
The entire parameter name must be passed in.
(i.e. no automatic completion).
Nothing beeps yet.
See also ERRORDLG，HELPDLG，QUESTDLG

注意：对话窗本身不是句柄图形对象，而是由一系列句柄图形对象构成的M文件。对话窗口是图形，包括与框架、编辑和按钮**uicontrol**对象共存的坐标轴。在将来的版本中，**dialog**可能成为具有更多功能的内置函数。缺省的对话框是一个帮助对话框，它是由 '**Default help string**' 字符串和标有 'OK' 的按钮键组成的编辑文本框。作为例子，键入>> **dialog**。

预先定义的对话框是由函数**helpdlg**，**enordlg**，**warndlg**和**gucstdlg**建立。**helpdlg**和**warndlg**接受文本字符串和窗口标题字符串作为输入参量。**errordlg**接收 '**Replace**' 变量作为输入。除了**questdlg**，所有上述函数都产生类似的对话框，它有各自缺省的标题和文本字符串。标有 '**OK**' 的单个按钮，则关闭对话框。**helpdlg**帮助文本是：

>> help helpdlg

HELPDLG： Displays a help dialog box.
HANDLE=HELPDLG(HELPSTRING，DLGNAME) displays the
message HelpString in a dialog box with title DLGNAME.
if a help dialog with that name is aldeady on the screen，
it is brought to the front.Otherwise it is created.

See also：DIALOG

帮助信息

Helpdlg：显示一个帮助文本框

HANDLE = HELPDLG(HELPSTRING，DLGNAME)在对话框中显示标题为dlgname的帮助信息helpstring。如果名为dlgname的帮助对话框已在屏幕上显示，则引到屏幕正面，否则就建立该帮助对话框。

参阅：dialog

---

warndlg的帮助文本是：

>>help warndlg
WARNDLG Creates a warning dialog box.
HANDLE=WARNDLG(WARNSTR， DLGNAME) creates a warning
diolog box which displays WARNSTR in a window named DLGNAME.A pushbutton labeled OK must be pressed to make the warning box disappear.

See also：DIALOG

errordlg的帮助文本是

>>help errordlg

ERRORDLG Creates an error dialog box
HANDLE = ERRORDLG(ERRORSTR，DLGNAME，Replace) creates an error dialog box which diplays ERRORSTR in a window nnnamed DLGNAME.A pushbutton labeled OK must be pressed to make the error box disappear.If REPLACE='on' and an error dialog with Name DLGNAME already exits， it is simply brought to the front (no new dialog is created).

See also ：DIALOG

帮助信息
ERRORDLG：建立出错对话框
HANDLE=ERRORDLG(ERRORSTR，DLGNAME，REPLACE)建立显示出错信息errorstr、名为dlgname的出错对话框，要消除出错信息对话框，必须揿下标记为OK的按钮，如果replace='on'并且名为digname的出错对话框已经存在，则就引到屏幕正面，不再建立新的对话框。

参阅：dialog

提问的对话框稍有不同，前三种函数只显示一个按钮键并返回对话框图形对象的句柄。函数**questdlg**显示两个或三个按钮键，返回由用户所选的按钮的标志字符串。 **questdlg**的帮助文本如下：

>>help qeustdlg

QEUSTDLG Creates a question dialog box.
CLICK=QUESTDLG(Q，YES，O，CANCEL，DEFAULT) creates a question dialog box which displays Q.Up to three pushbuttons，with strings given by YES，NO，and CANCEL，will appear along with Q in the dialog.The dialog wil be destroyed returning the string CLICK depending on which button is clicked.DEFAULT is the default button number.

帮助信息
QUESTDLG：建立问题对话框
CLICK=QVESTDLG(Q，YES，NO，CANCEL，DEFAULT)建立显示信息Q的问题对话框。至多有三个按钮具有由YES，NO和CANCEL给定的字符串，按钮与Q一起，显示在对话框中。根据所击的按钮返回字符串CLICK对话框消失。DEFAULT是缺省的按钮数。

下面是函数的片断，说明了函数中提问对话框的应用：

```
question1=' Change color map to copper? ' ；
response1=questdlg(question1， ' Sure ' ， ' Nope ' ， ' Maybe ' ，2)；
if stromp(response1， ' Sure ' )
    colormap(copper)；
elseif stromp(response1， ' Maybe ' )
    wandlg( ' That response does not compute! ' )；
    response2=questdly([ ' Please make up your mind.｜ ' question1]， ' Yes ' ， ' No ' )；
    if stromp (response2， ' Yes ' )
        colormap(copper)；
    end
end
```

请求程序

请求程序通过对话框获取用户的输入。请求程序是内置式GUI函数，它使用平台原有的窗口系统建立外观熟悉的请求程序。

函数**uigecfile**和**uiputfile**是所有平台上都有的内置式函数，用于交互地获得文件名，从而调用函数用它读取文件中数据或将数据存于文件中。**uigetfile**的帮助文本是：

>>help uigetfile

UIGETFILE Interactively retrieve a filename by displaying a dialog box.

[FILENAME，PATHNAME]=UIGETFILE(' filterSpec ' ， ' dialogTitle ' ，X，Y)
displays a dialog box for the user to fill in，and returns the
filename and path strings.A successful return occurs only if
the file exists.If the user selects a file that does not exist，
an error message is displayed，and control returns to the dialog box.
The user may then enter another filename，or press the Cancel button.

All parameters are optional，but if one is used，all previous
parameters must also be used.

The felterSpec parameter determines the initial display of files in
the dialog box.For example ' *.m ' lists all the MATLAB M-files.

Parameter ' dialogTitle ' is a string containing the title of the dialog
box.

The X and Y parameters define the initial position of the dialog
box in units of pixels. Some systems may not support this option.

The output variable FILENAME is a string containing the mane of the
file selected in the dialog box.If the user presses the Cancel button
or if any error occurs，it is set to 0.

The output parameter PATHNAME is a string containing the path of
the file selected in the dialog box. If the user presses the Cancel
button or if any error occurs，it is set to 0.

See also UIPUTFILE.

帮助信息

UIGETFILE：通过显示对话框交互式地检索文件名

[FILENAME，ATHNAME]=UIGETFILE(' filterspec ' ， ' dialogtitle ' ，x，y)显示一个对话框，让用户输入，并返回路径和文件名字符串。仅当文件存在时，才成功地返回。如果用户选择了一个并不存在的文件，就显示出错信息，控制框返回到对话框。用户可以输入另一个文件名或掀下Cancel按钮。

所有输入参数都是可任选的，如果用其中之一个，也必须使用所有先前参数。
参数filteospec决定对话框中文件的初始显示。例如 ' *.m ' 列出的所有M文件。
参数 ' dialogtitle ' 是对话框标题字符串。
以象素为单位参数x，y定义对话框的初始位置，有些系统可能不支持这个选项。
输出变量filename是对话框内所选文件的名称字符串。如果用户掀了取消按钮或有错误发生，filename的值设置为0。
输出参数pathname是对话框内所选文件的路径名字符串。如果用户掀了取消按钮或

有错误发生，pathname的值设置为0。
参阅 uiputfile

下面的例子说明了在函数中如何利用**uigetfile**，交互式地检索ASCII码数据文件，并绘制正弦数据

```
% Ask the user for a file name.

[datafile datapath]=uigetfile( ' .dat ' ， ' Choose a data file ' )；

% If the user selected an existing file，read the data.
% (The extra quotes avoid problems with spaces in file or path names
% on the Macintosh platform.)
% Then determine the variable name from the file name，
% copy the data to a variable，and plot the data.
if datafile
        eval([ ' load( '  ' datapath datafile '  ' ) ' ] ' )；
        x=eval(strtok(datafile. ' . ' )；
        plot (x，sin(x))；
end
```

请求程序不接受并不存在的文件名。这种情况可能发生的原因是用户在请求程序中把文件名输入到文件框中。Macintosh版本无文本框的，用户必须选择一个存在的文件或揿下**Cancel**按钮以退出请求程序。

函数**uiputfile**与函数**uigetfie**十分相似：输入参量相似，而且两者均返回文件和路径字符串，**uiputfile**的帮助文件是

```
>> help uiputfile

UIPUTFILE Interactively retrieve a filename by displaying a dialog box.
[FILENAME，PATHNAME]=UIPUTFILE( ' inifFile ' ， ' dialogTitle ' )
displays a dialog box and returns the filename and path strings.

The initFile paramete determines the initial display of files
in the dialog box.Full file name specifications as well as wildcards
are allowed.For example， ' newfile.m '  initializes the display to
that particular file and lists all othe existing .m files.This may
be used to provide a default file mane. A wildcard specification such
as  ' *.m '  lists all the existing MATLAB M-files.

Parameter  ' dialogTitle '  is a string containing the title of the dialog
box.
```

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Cancel butto
or if any error occurs，it is set to 0.

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Concel button
or if any error occurs，it is set to 0.

[FILENAME，PATHNAME]=UIPUTFILE (' initFile ' ， ' dialogTitle ' ，X，Y)
places the dialog box at screen position [X，Y] in pixel units.
Not all systems support this option.

Example：

[newmatfile，newpath]=uiputfile (' *.mat ' . ' Save As ')；

See also UIGETFILE.

如果用户选择了已经存在的文件，则出现一个对话窗，询问用户是否要删除存在的文件。如果回答**no**，则返回原来的请求程序等待另一次尝试；如果回答**yes**，则关闭请求程序和对话框并把文件名返回，文件并未被请求程序删除。如果需要，调用函数必须删除或覆盖文件。

值得牢记的是，这些函数中不论哪一个都未真正地读或写任何文件，调用函数必须做这些工作。这些函数仅仅是将文件名和路径返回给调用函数。

在MS-Windows和Macintosh平台上，**uisetcolor**让用户交互式地选择颜色并选择性地将此颜色施加到一个对象上。如同MATLAB4.2C版本，X Window系统平台不支持**uisetcolor**。

uisetcolor的帮助文本是：

>>help uisetclor

UISETCOLOR Interactively set a ColorSpec by displaying a dialog box.
C=UISETCOLOR(ARG， ' dialogTitle ') displays dialog box for the
user to fill in，and applies the selected color to the input
graphics object.

The parmeters are optional and may be specified in any order.

ARG may be either a handle to a graphics object or an RGB triple.
If a handle is used，it must specify a graphics object that supports
color.If RGB is used，it must be a valid RGB triple (e.g.，[1 0 0]
for red). In both cases，the color specified is used to initialize
initializes the color to black.

If parameter ' dialogTitle ' is used， it is a string containing the
title of the dialog box.

The output value C is the selected GB triple. If the input parmeter is a handle，the graphics object ' s color is set to the RGB color selected.

If the user presses Cancel from the dialog box，or if any error occurs，the output value is set to input RGB triple，if provided；otherwise，it is set to 0.

Example：

C=uisetcolor(hText，　' Set Text Color ')

NOTE：This function is only available in MS-Windows and Macintosh versions of MATLAB.

帮助信息

UISETCOLOR ：显示对话框，交互式地设置ColorSpace。

C=UISETCOLOR(ARG，　' dialogtitle ')显示一个对话框，让用户输入，并将所选颜色用于输入的图形对象。

参数是可任选，并可以以任何次序指定。

ARG可以是一个图形对象的句柄或是RGB3元组。如果使用句柄，必须指定支持颜色的图形对象；如果使用RGB，必须是有效的RGB3元组(如：[1　0　0]是红色)。在这两种情况下，所指定的颜色用于对话框的初始化。如果没有指定初始的RGB，将对话框初始化为黑色。

如果使用参数 ' dialogTitle ' ，该参数是对话框名称的字符串。

输出值C是所选的RGB3元组。如果输入参数是句柄，则图形对象的颜色就设定为所选的颜色。

如果用户揿下对话框中的Cancel按钮或有错误发生，输出值就设定为输入的RGB3元组；如果没有输入RGB3元组，则输出值设置为0。

例如：c=uisetcoior(htext，　' settextcoior ')

注意：该函数仅在的MS-Windous和Macintosh版本中可用。

精通MATLAB工具箱具有前面所提及的函数**mmsetc**。该函数可在所有的平台上工作，功能与**uisetcolor**十分相似。**mmsetc**甚至可允许用户用鼠标来选择颜色并将所选的颜色加到所选择的对象上。以下就是**mmsetc**的帮助文本：

>> help mmsetc

MMSETC Obtain an RGB triple interactively form a color sample.

MMSETC displays a box for the user to select
a color nteractively and displays the result.

X=MMSETC returns the selected color in X.

MMSETC ([r g b]) uses the RGB triple as the initial
RGB value for modification.

MMSETC C -or-
MMSETC (' C ') where C is a color spec (y，m，c，r，g，b，w，k)，uses the specified color as the initial value.

MMSETC(H) where the input argument H is the handle of a valid graphics object that supports color，uses the color progerty of the object as the initial RGB value.

MSETC select -or-
MMSTEC(' select ') waits for the user to click on a valid graphics object that supports color，and uses the color property of the object as the initial RGB value.

If the initial RGB value was obtained from an object or object handle，the ' Done ' pushbutton will apply the resulting color property to the selected object.

If no initial color is specified，black will be used.

Examples：

```
mmsetc
mycolor=mmsetc
mmsetc([.25 .62 .54])
mmsetc(H)
mmsetc g
mmsetc red
mmsetc select
mycolor=mmsetc( ' select ' )
```

下面的例子是利用精通MATLAB工具箱中函数**mmap**和**mmsetc**，交互式地使用单色映象：

```
>>mesh(peaks)
```

>>coiormap(mmap(mmsetc))

在MS-Windows和Machitosh平台上，有最后的请求程序**uisetfont**，它让用户可交互式选择字型属性并将其加于对象上。如同MATLAB4.2c版本，X Window系统平台不支持**uisetfont**。**uisetfont**帮助文件是：

```
>>  help uisetfont

UISETFONT   Interactively set a font by displaying a dialog box.
    H=UISETFONT(HIN，  ' dialogTitle ' ) displays a dialog box for the
    user to fill in，and applies the selected font to the input
    graphics object.

    The parmeters are optional and may be specified in any order.

    If parameter HIN is used，it must specity a handle tp a text or
    axis graphics object. The font properties currently assigned to
    this object are used to initialize the font dialog box.

    If parameter  ' dialogTitle '  is used，it is a string containing the
    title of the dialog box.

    The output H is a handle to graphics object. If HIN is specified，
    H is identical to HIN.If HIN is not specified，a nwe text object
    is created with the selected font properties，  and its handle is
    returned.

    If the user presses Cancel from the dialog box，or if any error
    occurs，the output value is set to the input handle，if provided；
    otherwise ，it is set to 0.

    Example：
        uisetfont(hText，  ' Update Font ' )

    NOTE：This function is only available in MS-Windows and Macintosh
        versions of MATLAB.
```

精通MATLAB工具箱有一个称为函数**mmsetf，**功能类似于**uisetfont**。但可工作在所有平台上。**mmsetf**甚至可允许用户用鼠标来选择文本对象，然后将所选的字体属性加在所选的对象上。以下就是**mmsetf**的帮助文本：

```
>>help mmsetf

MMSETF Choose font characteristics interactively.
```

MMSETF displays a dialog box for the user to select
fornt characteristics.

X=MMSETF returns the handle of the text object or 0
if an error occurs of ' Cancel ' is pressed.

MMSETF(H) where the input argument H is the handle of
a valid text or axes object，uses the font characteristics
of the object as the initial values.

MMSETF select -or-
MMSETF (' select ')waits for the user to click on a valid
graphics object，and uses the font characteristics
of the object as the initial values.

If the initial values were obtained from an object or
object handle，the ' Done ' pushbutton will apply the
resulting text properties to the selected object.

If no initial object handle is specified，a zero is returned in X.

Examples：
mmsetf
mmsetf(H)
mmsetf select
Hx_obj=mmsetf(' select ')

然而**mmsetf**中用于字体的选择是受限制的。不同的平台类型，甚至同一类型的不同计算机都可能安装了不同的字体。MATLAB的函数**uisetfont**是置于内部的，并利用计算机平台的操作系统列出可用的字体。因为**mmsetf**是的GUI 函数，不能确定哪种字体是可用的，所以用了通常可获得的有限选择。字体的大小也因同样原因受到限制。

mmsetf请求程序中的样本文本字符串指明了每一个属性改变的作用。选择后，如果出现的文本字符串并不如所希望的那样，则说明所选择的字体属性(如字体名称、大小等等)在所用计算机上是不存在的。见到的即所得到的。

21.10 用户自制的GUI M文件

许多MATLAB用户充分利用的MATLAB所提供的GUI工具，编写了一些有趣GUI函数。这中间大部分可在MATLAB的 匿名**FTP** 地址**/pub/contrib/graphics**路径中得到。Internet资源一章将详细地讨论如何从这一资源库中获得文件的详细指令。

一些M文件和M文件集含有大量的句柄图形GUI对象，并说明了**uimenu**和**uicontrol**的使用。其中让人印象最深刻的是科斯·荣格(Keith Roger)编写的**matdraw**集合。该文件集合可在**/pub/contrib/graphics/matdraw2.0**目录下获得。该文件集将工具条和画图工具调色板加到

图形窗口，效果同集成到MATLAB的绘图程序一样。

另一个值得研究的是派瑞克·马查德(Patrick Marchard)编写的**guimaker**集合。这是交互式工具的集合，让用户应用GUI来建立GUI函数。可以用鼠标建立GUI对象，放置GUI对象和指定GUI对象的大小。版本1.0以freeware形式发布，即它可以无偿使用。2.0版本以及以后的版本以shareware形式发布的，即你可以无偿地使用30天，然后你需注册并付不多的费用。两个版本均可在的 匿名ftp地址**/pub/contrib/graphics**路径中得到。

这里只是现有GUI函数很多例子中的一小部分。核实一下。也许会对别人要用但已消失的GUI应用程序有想法。第23章有详细指导信息帮助用户访问现有的MATLAB资源，甚至帮助用户成为M文件的贡献者。

21.11 小结

图形用户界面设计不是为每一个人的。但如果需要，MATLAB使它有可能仅由M文件来建立令人印象深刻的GUI函数。需用**mex**文件、高级语言或数据库调用对有用的MATLAB函数来建立一个赏心悦目的界面。

对GUI对象的字体控制局限性，在下一版本MATLAB的发行中应有所强调。对话框也有可能在未来成为内置函数。所有这些改进工作将使编写GUI函数更加容易，使它们既与平台无关并且更加悦目。

本章所讨论的函数总结如下

表21.4

句柄图形GUI函数	
uimenu(handle， 'PropertyName'，value)	创建或改变图形的菜单
uicontrol(handle， 'PropertyName'，value)	创建或改变对象的性质
dialog('PropertyName'，value)	显示对话框
helpdlg('HelpString'， 'DlgName')	显示 ' 帮助 ' 对话框
warndlg('WarnString'， 'DlgName')	显示 ' 警告 ' 对话框
errordlg('ErrString'， 'DlgName'，Replace)	显示 ' 出错对话框
questdlg('Qstring'，S1，s2，s3，Default)	显示 ' 提问 ' 对话框
uigetfile(Filter，Dlgname，X，Y)	交互式地检索文件名
uiputfile(InitFile，Dlgname，X，Y)	交互式地检索文件名
uisetcolor(Handle，Dlgname)	交互式地选择颜色
uisetcolor([r g b]，Dlgname)	交互式地选择颜色
uisetfont(Handle，Dlgname)	交互式地选择字体属性

以下是本章所提到的精通MATLAB工具箱中的函数：

表21.5

精通MATLAB工具箱中的句柄图形GUI函数	
mmenu	**uimenu**函数示例
mmclock(X，Y)	**uicontrol**函数示例
mmsetclr	函数**mmsetc**的有限脚本型式
mmview3d	把方位角和仰角的滑标加到图形上

mmcxy	用鼠标显示的x-y坐标
mmtext(' String ')	用鼠标放置和拖曳文本
mmdraw(' Name ' ， Value)	用鼠标画线并设置属性
mmsetc(Handle)	用鼠标设置颜色属性
mmsetc(ColorSpec)	
mmsetc(' select ')	
mmsetf(Handle)	用鼠标设置字体属性
mmsetf(' select ')	

第22章　符号数学工具

MATLAB所具有的符号数学工具箱与其它所有工具不同，它适用于广泛的用途，而不是针对一些特殊专业或专业分支。另外，MATLAB符号数学工具箱与其它的工具箱区别还因为它使用字符串来进行符号分析，而不是基于数组的数值分析。为此，本章包含了该工具箱的教学辅导材料。

22.1　引言

符号数学工具箱是操作和解决符号表达式的符号数学工具箱(函数)集合，有复合、简化、微分、积分以及求解代数方程和微分方程的工具。另外还有一些用于线性代数的工具，求解逆、行列式、正则型式的精确结果，找出符号矩阵的特征值而无由数值计算引入的误差。工具箱还支持可变精度运算，即支持符号计算并能以指定的精度返回结果。

符号数学工具箱中的工具是建立在功能强大的称作Maple软件的基础上。它最初是由加拿大的滑铁卢（Waterloo）大学开发的。当要求MATLAB进行符号运算时，它就请求Maple去计算并将结果返回到MATLAB命令窗口。因此，在MATLAB中的符号运算是MATLAB处理数字的自然扩展。

22.2　符号表达式

符号表达式是代表数字、函数、算子和变量的MATLAB字符串，或字符串数组。不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则，这些符号表达式几乎完全象基本的MATLAB命令。表22.1列有几则符号表达式例子以及MATLAB等效表达式。

表22.1

符号表达式	MATLAB表达式
$\frac{1}{2x^n}$	'1/(2*x^n)'

$\frac{1}{y=\sqrt{2x}}$	y= ' 1/sqrt(2*x) '
$\cos(x^2)-\sin(2x)$	' cos(x^2)-sin(2*x) '
$M=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym(' [a，b；c，d] ')
$\int_a^b \frac{x^3}{\sqrt{1-x}}dx$	f=int(' x^3/sqrt(1-x) ' ， ' a ' ， ' b ')

MATLAB符号函数可让用户用多种方法来操作这些表达式，比如，

```
>> diff( ' cos(x) ' )   %   differentiate cos(x) with respect to x
ans=
  -sin(x)

>> M=sym( ' [a，b；c，d] ' )   %   create a symbolic matrix M
M=
  [a，b]
  [c，d]

>> determ(M)   %   find the determinant of the symbolic matrix M
ans=
  a*d-b*c
```

请注意，上面的第一个例子的符号表达式是用单引号以隐含方式定义的。它告诉MATLAB **' cos(x) '** 是一个字符串并说明**diff(' cosx ')**是一个符号表达式而不是数字表达式；然而在第二个例子中，用函数**sym**显式地告诉MATLAB **M=sym(' [a，b；c，d] ')**是一符号表达式。在MATLAB可以自己确定变量类型的场合下，通常不要求显式函数**sym**。

正如在第八章所阐述，MATLAB中函数**function argument**形式是与**function(' argument ')**等价的。其中，**function**是一个函数，**argument**是一字符串。例如，MATLAB可以构造**diff cos(x)**和**diff(' cos(x) ')**两者都意味**diff (sym ' cos(x) ')**。但第一种形式显然更便于输入。然而，很多时候**sym**是必要的。在上述的第二个例子中，

```
>> M=[a，b；c，d]   %   M is a numeric matrix using value of a through d
???Uundefine function or variable a.

>> M= ' [a，b；c，d] '    %   M is a character string，  but not a symbolic matrix
M=
  [a，b；c，d]

>> M=sym( ' [a，b；c，d] ' )   %   M is a symbolic matrix
M=
```

[a，b]
[c，d]

M以三种方式定义： 数字型(如果**a**、**b**、**c**、**d**已预先确定)、字符串型或符号矩阵型。许多符号函数非常巧妙能够自动将字符转变为符号表达式。但在某些情况下，尤其是建立符号数组时，必须用函数**sym**，特别地将字符串变为符号表达式。隐含形式，例如**diff cos(x)**，对于那些不需要参考先前结果的简单任务，最有用。但是最简单形式(无引号)要求一个参量，它是一个单字符的字符串、不包含插入的空格。

```
>> diff x^2+3*x+5    %    the argument is equivalent to    ' x^2+3*x+5 '
ans=
  2*x+3

>> diff x^2 + 3*x + 5    %    spaces break the argument into separate strings
???Error using==>diff
Too manyinput arguments
```

无变量的符号表达式称作符号常量。符号常量常常与整数很难区别，例如

```
>> f=symop( ' (3*4-2)/5+1 ' )    %    reduce a symbolic constant to its simplest form
f =
  3

>> isstr(f) %    is f a string? (1=yes，  0=no)
ans=
  1
```

在这个例子中，f代表符号常数 '3' ；而不是数字3。正如第六章所阐述的，MATLAB是以字符ASCII码形式来存储字符串的。所以，如果对字符串进行数字运算，则在运算中，采用各字符串的ASCII码值。因为数字51是字符 '3' 的ASCII表示，所以f加1在数值上不能得到期望的结果

```
>> f+1
ans=
  52
```

符号变量

当字符表达式中含有多于一个的变量时，只有一个变量是独立变量。如果不告诉MATLAB哪一个变量是独立变量，MATLAB将基于以下规则选择一个：

在符号表达式中缺省的独立变量是唯一的，除去i和j的小写字母，不是单词的一部分。如果没有这种字母，就选择x作为独立变量。如字符不是唯一的，就选择在字母顺序中最接近x的字母。如果有相连的字母，就选择在字母表中较后的那一个。

缺省的独立变量，有时称作自由变量，在表达式 **'1/(5+cos(x))'** 中是 **'x'** ；在 **'3*y+z'** 中是 **'y'** ；在 **'a+sin(t)'** 是 **'t'** 。在表式 **'sin(pi/4)-cos(3/5)'** 中自由符号变量是 **'x'** ，因为此式是一个符号常数无符号变量。可利用函数**symvar**询问MATLAB在符号表达式中哪一个变量它认为是独立变量。

```
>> symvar( ' a*x+y*)   %   find the default symbolic variable
ans=
  x

>> symvar( ' a*t+s/(u+3) ' )   %   u is the closest to   ' x '
ans=
  u

>> symvar( ' sin(omega) ' )   %    ' omega '   is not a singlee character。
ans=
  x

>> symvar( ' 3*i+4*j ' )   %   i and j are equel to sqrt(-1)
ans=
  x

>> symvar( ' y+3*s ' ,  ' t ' )   %   find the variable closest to t rather than x
ans=
  s
```

如果利用规则**symvar**不能找到一个缺省独立变量，它便假定无独立变量并返回x。这一结论对含有由多个字母组成的变量，如：alpha或s2的表达式，或不含变量的符号常数均成立。如果需要，绝大多数命令都使用用户选项以指定独立变量。

```
>> diff( ' x^n ' )   %    differentiate with respect to the default variable   ' x '
ans=
  x^n*n/x

>>   diff( ' x^n ' ,  ' n ' )   %    differentiate x^n with respect to   ' n '
ans=
  x^n*log(x)

>> diff( ' sin(omega) ' )   %    differentiate using the default variables (x)
ans=
  0

>> diff( ' sin(omega) ' ,  ' omega ' )   %    specify the independent variable
ans=
```

cos(omega)

22.3 符号表达式运算

一旦创建了一个符号表达式，或许想以某些方式改变它；也许希望提取表达式的一部分，合并两个表达式或求得表达的数值。有许多符号工具可以帮助完成这些任务。

所有符号函数(很少特殊例外的情况，讨论于后)作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可能看起来象一个数字，但事实上它是一个内部用字符串表示的一个符号表达式。正如我们前面所讨论的，可以运用MATLAB函数**isstr**来找出像似数字的表达式是否真是一个整数或是一个字符串。

提取分子和分母

如果表达式是一个有理分式(两个多项式之比)，或是可以展开为有理分式(包括哪些分母为1的分式)，可利用**numden**来提取分子或分母。例如，给定如下的表达式：

$$m = x^2 \qquad f = \frac{ax^2}{b-x} \qquad g = \frac{3}{2}x^2 + \frac{2}{3}x - \frac{3}{5} \qquad h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1} \qquad k = \begin{bmatrix} \frac{3}{2} & \frac{2x+1}{3} \\ \frac{4}{x^2} & 3x+4 \end{bmatrix}$$

在必要时，**numden**将表达式合并、有理化并返回所得的分子和分母。进行这项运算的MATLAB语句是：

```
>> m= ' x^2 '     %    create a simple expression
m=
     x^2

>> [n，d]=numden(m)    %    extract the numerator and denominator
n=
     x^2
d=
     1

>> f= ' a*x^2/(b-x) '     %    create a rational expression
f=
     a*x^2/(b-x)

>> [n，d]=numden(f)    %    extract the numerator and denominator
n=
     a*x^2
d=
     b-x
```

前二个表达式得到期望结果。

```
>> g= ' 3/2*x^2+2/3*x-3/5 '     %    rationalize and extract the parts
g=
     3/2*x^2+2/3*x-3/5

>> [n，d]=numden(g)
n=
     45*x^2+20*x-18
d=
     30

>> h= ' (x^2+3)/(2*x-1)+3*x/(x-1) '     %    the sum of rational polynomials
h=
     (x^2+3)/(2*x-1)+3*x/(x-1)

>> [n，d]=numden(h)    %    rationalize and extract
n=
     x^3+5*x^2-3
d=
     (2*x-1)*(x-1)
```

在提取各部分之前，这二个表达式g和h被有理化，并变换成具有分子和分母的一个简单表达式。

```
>> k=sym( ' [3/2，(2*x+1)/3；4/x^2，3*x+4] ' )    %     try a symbolic array
k=
     [     3/2，(2*x+1)/3]
     [4/x^2，          3*x+4]

>> [n，d]=numden(k)
n=
     [3，  2*x+1]
     [4，  3*x+4]
d=
     [     2，3]
     [x^2，1]
```

这个表达式k是符号数组，**numden**返回两个新数组**n**和**d**，其中**n**是分子数组，**d**是分母数组。如果采用**s=numden（f)**形式，**numden**仅把分子返回到变量**s**中。

标准代数运算

很多标准的代数运算可以在符号表达式上执行，函数**symadd**、**symsub**、**symlnul**和**symdiv**为加、减、乘、除两个表达式，**sympow**将一个表达式上升为另一个表达式的幂次。例如： 给定两个函数

$$f = 2x^2 + 3x - 5 \qquad g = x^2 - x + 7$$

```
>> f= ' 2*x^2+3*x-5 '     %    define the symbolic expression
f=
     2*x^2+3*x-5

>> g= ' x^2-x+7 '
g=
     x^2-x+7

>> symadd(f，g)    %    find an expression for f+g
 ans=
     3*x^2+2*x+2

>> symsub(f，g)    %    find an expression for f-g
ans=
     x^2+4*x-12

>> symmul(f，g)    %    find an expression for f*g
ans=
     (2*x^2+3*x-5)*(x^2-x+7)

>> symdiv(f，g)    %    find an expression for f/g
ans=
     (2*x^2+3*x-5)/(x^2-x+7)

>> sympow(f，  ' 3*x ' )    %    find an expression for  f^3*
ans=
     (2*x^2+3*x-5)^3**
```

另一个通用函数可让用户用其它的符号变量、表达式和算子创建新的表达式。**symop**取由逗号隔开的、多至16个参量。各个参量可为符号表达式、数值或算子(' + '、' - '、'*'、' / '、' ^ '、' ('或') ')，然后**symop**可将参量联接起来，返回最后所得的表达式.

```
>> f= ' cos(x) '     %    create an expression
f=
     cos(x)

>> g= ' sin(2*x) '     %    create another expression
g=
```

```
    sin(2*x)

>> symop(f，'/　′，g，'+'，3)　%　combine them
ans=
    cos(x)/sin(2*x)+3
```

所有这些运算也同样用数组参量进行。

高级运算

MATLAB具有对符号表达式执行更高级运算的功能。函数**compose**把*f(x)*和*g(x)*复合成*f(g(x))*。函数**finverse**求表达式的函数逆，而函数**symsum**求表达式的符号和。

给定表达式

$$f=\frac{1}{1+x^2} \qquad g=\sin(x) \qquad h=\frac{1}{1+u^2} \qquad \mathrm{k}=\sin(\mathrm{v})$$

```
>> f= ' 1/(1+x^2) ' ；　%　create the four expression

>> g= ' sin(x) ' ；

>> h= ' 1/(1+u^2) ' ；

>> k=' sin(v) ' ；

>> compose(f，g)　%　find an expression for f(g(x))
ans=
   1/(1+sin(x)^2)

>> compose(g，f)　%　find an expression for g(f(x))
ans=
   sin(1/(1+x^2))
```

compose也可用于含有不同独立变量的函数表达式。

```
>> compose(h，k，'u'，'v')　%　given h(u)，k(v)，find(k(v))
ans=
   1/(1+sin(v)^2)
```

表达式譬如*f(x)*的函数逆*g(x)*，满足*g(f(x))=x*。例如，e^x的函数逆是*ln(x)*，因为*ln*(e^x)=*x*。*sin(x)*的函数逆是*arcsin(x)*，函数$\frac{1}{\tan(x)}$的函数逆是*arcsin*$(\frac{1}{x})$。函数**fincerse**返回表达式的函数逆。如果解不是唯一就给出警告。

```
>> finverse( ' 1/x)    %    the inverse of 1/x is 1/x since    ' 1/（1/x)=x '
ans=
    1/x

>> finverse( ' x^2 ' )    %    g(x^2)=x has more than one solution
Warning：      finverse(x^2) is not unique
ans=
    x^(1/2)

>> finverse( ' a*x+b ' )    %    find the solution to    ' g(f(x))=x '
ans=
    -(b-x)/a

>> finverse( ' a*b+c*d-a*z ' )，    ' a ' )    %    find the solution to    ' g(f(a))=a '
ans=
    -(c*d-a)/(b-z)
```

symsun函数求表达式的符号和有四种形式：**symsun(f)**返回$\sum_{0}^{x-1} f(x)$；**symsum(f，'s')**返回$\sum_{0}^{s-1} f(s)$，**symsun(f，a，b)**返回$\sum_{a}^{b} f(x)$；最普通的形式**symsun(f，'s'，a，b)**返回$\sum_{a}^{b} f(s)$。

让我们试一试$\sum_{0}^{x-1} x^2$，它应返回：$\frac{x^3}{3}-\frac{x^2}{2}+\frac{x}{6}$。

```
>> symsum('x^2')
ans=
    1/3*x^3-1/2* x^2+1/6*x
```

$\sum_{1}^{n}(2n-1)^2$ 又怎么样呢?它应返回$\frac{n(2n-1)(2n+1)}{3}$。

```
>> sym('(2*n-1)^2'，1，'n')
ans=
    11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3

>> factor(ans)    %    change the form ( we will revisit 'factor' later on)
ans=
    1/3*n*(2*n-1)*(2*n+1)
```

最后让我们试一试$\sum_{1}^{\infty}\frac{1}{(2n-1)^2}$，其返回应是$\frac{\pi^2}{8}$。

```
>> symsum( ' 1/(2*n-1)^2 ' ， 1， inf)
ans=
    1/8*pi^2
```

变换函数

本节提出许多工具，将符号表达式变换成数值或反之。有极少数的符号函数可返回数值。然而请注意，某些符号函数能自动地将一个数字变换成它的符号表达式，如果该数字是函数许多参量中的一个。

函数**sym**可获取一个数字参量并将其转换为符号表达式。函数**numneric**的功能正好相反，它把一个符号常数（无变量符号表达式）变换为一个数值。

```
>> phi=' (1+sqrt(5))/2 '   %   the   ' golden '   ratio
phi=
    (1+sqrt(5))/2   %   convert to a numeric value

>> numeric(phi)
ans=
    1.6180
```

正如第六章所介绍，函数**eval**将字符串传给MATLAB以便计算。所以**eval**是另一个可用于把符号常数变换为数字或计算表达式的函数。

```
>> eval(phi)   %   execute the string   ' (1+sqrt(5))/2
ans=
    1.6180
```

正如所期望那样，**numeric**和**eval**返回相同数值。

符号函数**sym2poly**将符号多项式变换成它的MATLAB等价系数向量。函数**poly2syrn**功能正好相反，并让用户指定用于所得结果表达式中的变量。

```
>> f=' 2*x^2+x^3-3*x+5 '   %     f is the symbolic polynomials
f=
    2*x^2+x^3-3*x+5

>> n=sym2poly(f)   %   extract eht numeric coefficient vector
n=
    1    2    -3    5
```

```
>> poly2sym(n)   %   recreate the polynomials in x (the default)
ans=
     2*x^2+x^3-3*x+5

>> poly2sym(n，'s')   %   recreate the polynomials in s
ans=
     s^3+2*s^2-3*s+5
```

变量替换

假设有一个以x为变量的符号表达式，并希望将变量转换为y。MATLAB提供一个工具称作**subs**，以便在符号表达式中进行变量替换。其格式为**subs（f，new，old)**，其中f是符号表达式，**new**和**old**是字符、字符串或其它符号表达式。‘新’字符串将代替表达式f中各个‘旧'字符串。以下有几个例子：

```
>> f='a*x^2+b*x+c'    %   create a function f(x)
f=
     a*x^2+b*x+c

>> subs(f，'s'，'x')   %   substitute   's'   for   'x'   in the expression f
ans=
     a*s^2+b*s+c

>> subs(f，'alpha'，'a')   %   substitute   'alpha'   for   'a'   in   f
ans=
     alpha*x^2+b*x+c

>> g=' 3*x^2+5*x-4'   %   create another function
g=
     3*x^2+5*x-4

>> h=subs(g，'2'，'x')   %   substitute   '2' for   'x'   in   g
h=
     18

>> isstr(h)   %   show that the result is a symbolic expression
ans=
     1
```

最后一个例子表明**subs**如何进行替换，并力图简化表达式。因为替换结果是一个符号常数，MATLLAB可以将其简化为一个符号值。注意，因为**subs**是一个符号函数，所以它返回一个符号表达式。尽管看似数字，实质上是一个符号常数。为了得到数字，我们需要使用函数**numeric**或**eval**来转换字符串。

```
>> numeric(h)    %    convert a symbolic expression to a number
ans=
      18

>> isstr(ans)    %    show that the result is a numeric value
ans=
      0
```

22.4 微分和积分

微分和积分是微积分学研究和应用的核心，并广泛地用在许多工程学科。MATLAB符号工具能帮助解决许多这类问题。

微分

符号表达式的微分以四种形式利用函数**diff**：

```
>> f= ' a*x^3+x^2-b*x-c '      %    define a symbolic expression
f=
      a*x^3+x^2-b*x-c

>> diff(f)    %    differentiate with respect to the default variable x
ans=
      3*a*x^2+2*x-b

>> diff(f，'a ')    %      differentiate with respect to a
ans=
      x^3

>> diff(f，2)    %      differentiate twice with respect to x
ans=
      6*a*x+2

>> diff(f，' a '，2)    %      differentiate twice with respect to a
ans=
      0
```

函数**diff**也可对数组进行运算。如果F是符号向量或数组，**diff(F)**对数组内的各个元素进行微分。

```
>> F=sym(' [a*x，     b*x^2；     c*x^3，     d*s] ')    %    create a symbolic array
F=
  [     a*x，     b*x^2]
  [c*x^3，          d*s]
```

```
>> diff(F)      %     differentiate the element with respect to x
ans=
  [              a，2*b*x]
  [3*c*x^2，          0]
```

注意函数**diff**也用在MATLAB，计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵M，diff(M)计算M(2： m，：)-M(1： m-1，：)的数值差分，如下所示：

```
>> m=[(1： 8).^2)]   %   create a vector
        M=
            1    4    9    16    25    36    49    64

        >> diff(M)   %   find the differences between elements
        ans=
            3    5    7    9    11    13    15
```

如果diff的表达式或可变参量是数值，MATLAB就非常巧妙地计算其数值差分；如果参量是符号字符串或变量，MATLAB就对其表达式进行微分。

积分

积分函数**int(f)**，其中f是一符号表达式，它力图求出另一符号表达式F使**diff(F)=f**。正如从研究微分学所了解的，积分比微分复杂得多。积分或逆求导不一定是以封闭形式存在，或许存在但软件也许找不到，或者软件可明显地求解，但超过内存或时间限制。当MATLAB不能找到逆导数时，它将返回未经计算的命令。

```
>> int( ' log(x)/exp(x^2) ' )   %   attempt to integrate
ans=
    log(x)/exp(x^2)
```

同微分一样，积分函数有多种形式。形式**int(f)**相对于缺省的独立变量求逆导数；形式**(f, 's ')**相对于符号变量s积分；形式**int(f，a，b)**和**int(f，'s'，a，b)，a，b**是数值，求解符号表达式从**a**到**b**的定积分；形式**int(f，'m' ，'n ')**和形式**int(f，'s'，'m'，'n ')**，其中**m，n**是符号变量，求解符号表达式从**m**到**n**的定积分。

```
>> f=' sin(s+2*x) ')   %   crate a symbolic function
f=
    sin(s+2*x)

>> int(f)   %   integrate with respect to x
ans=
    -1/2*cos(s+2*x)
```

```
>> int(f，' s ')     %    integrate with respect to s
ans=
      -cos(s+2*x)

>> int(f，pi/2，pi)    %    integrate with respect to x from π/2 to π
ans=
      -cos(x)

>> int(f，' s '，pi/2，pi)    %    integrate with respect to s from π/2 to π
ans=
      cos(2*x)-sin(2*x)

>> int(f，' m '，' n ')     %    integrate with respect to x from m to n
ans=
      -1/2*cos(s+2*n)+1/2*cos(s+2*m)
```

正如函数**diff**一样，积分函数**int**对符号数组的每一个元素进行运算。

```
>> F=sym( ' [a*x，b*x^2；c*x^3，d*s] ' )    %    create a symbolic array
F=
  [ a*x，b*x^2]
  [c*x^3，  d*s]

>> diff(F)    %    ubtegrate the array elements with respect to x
ans=
  [1/2*a*x^2，1/3*b*x^3]
  [1/4*c*x^4，        d*s*x]
```

22.5　符号表达式画图

在许多的场合，将表达式可视化是有利的。MATLAB提供了函数**ezplot**来完成该任务。

```
>> y=' 16*x^2+64*x+96 '    %    expression to plot
y=
  16*x^2+64*x+96

>> ezplot(y)
```

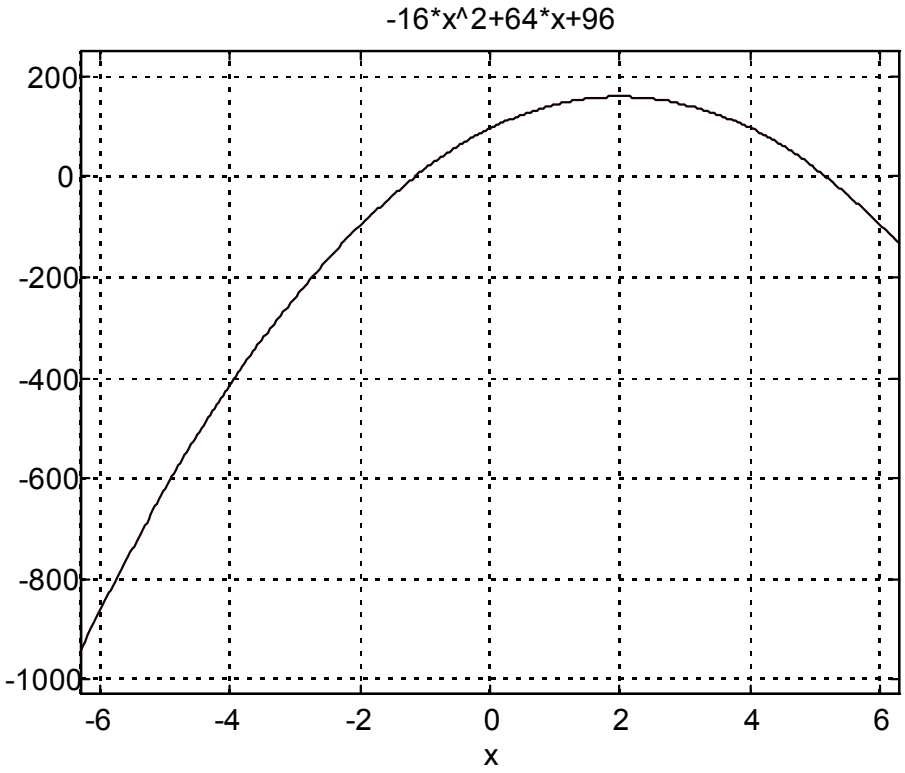


图22.1　符号函数16*x^2+64*x+96　(-2π≤x≤2π)

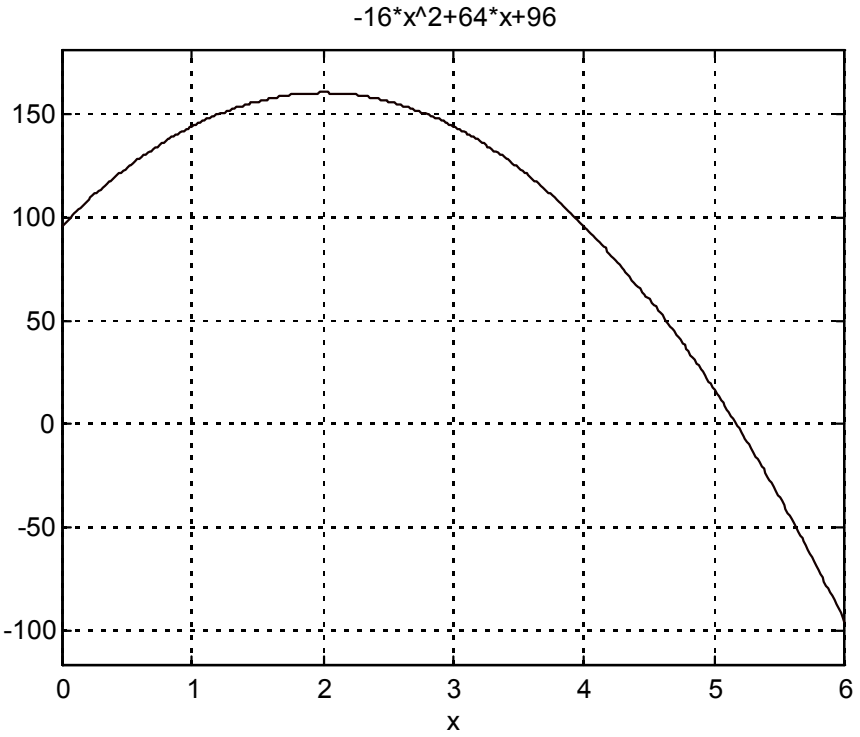


图22.2　符号函数16*x^2+64*x+96　　0≤x≤6

正如图22.1所示，**ezplot**绘制了定义域为-2π≤x≤2π的给定符号函数，并相应地调整了y轴比例，还加了网格栅和标志。在这个例子中，我们感兴趣的时间是从0到6。让我们再试一下，并指定时间范围，见图22.2

```
>> ezplot(y，[0 6])    %    plot y for 0≤x≤6
```

现在，在所感兴趣的范围内显示得略好些。一旦该图处在图形窗口内，它可以象其它图象一样作修改。

22.6　符号表达式简化和格式化

有时MATLAB返回的符号表达式难以理解，有许多工具可以使表达式变得更易读懂。第一个就是函数**pretty**，该命令以类似于数学课本上的形式来显示符号表达式。

```
>> f=taylor( ' log(x+1)/(x-5) ' )    %      6 terms is the default
f =
  -1/5*x+3/50*x^2-41/750*x^3+293/7500*x^4-1207/37500*x^5+O(x^6)

>> pretty(f)
                                          2        41     3        293     4        1207
5              6
                - 1/5 x + 3/50 x    -    -----x      + ------x      -    -------x    + O(x )
                                                 750              7500             37500
```

符号表达式可用许多等价形式来提供。在不同的场合，某种形式可能胜于另一种。MATLAB用许多命令来简化或改变符号表达式。

```
>> f=sym( ' (x^2-1)*(x-2)*(x-3) ' )    %    create a function
f=
      (x^2-1)*(x-2)*(x-3)

>> collect(f)    %    collect all like terms
ans=
      x^4-5*x^3+5*x^2+5*x-6

>> hornor(ans)      %    change to Horner or nested representation
ans=
      -6+(5+(5+(-5+x)*x)*x)*x

>> factor(ans)    %    express as a product of polynomials
ans=
      (x-1)*(x-2)*(x-3)*(x+1)

>> expand(f)    %    distribute products over sums
```

```
ans=
    x^4-5*x^3+5*x^2+5*x-6
```

Simplify是功能强大、通用的工具。它利用各种类型代数恒等式，包括求和、积分和分数幂、三角、指数和log函数、Bessel函数、超几何函数和γ函数，来简化表达式。

下面例子说明函数的乘幂。

```
>> simplify( ' log(2*x/y) ' )
ans=
    log(2)+log(x)-log(y)

>> simplify( ' sin(x)^2+3*x+cos(x)^2-5 ' )
ans=
    3*x-4

>> simplify( ' (-a^2+1)/(1-a) ' )
ans=
    a+1
```

其中要讨论的最后一个函数是最有用的，但也是最不正统的。函数**simple**试用了几种不同的简化工具，然后选择在结果表达式中含有最少字符的那种形式。让我们看一下立方根：

$$f=\sqrt[3]{\frac{1}{x^3}+\frac{6}{x^2}+\frac{12}{x}+8}$$

```
>> f=' (1/x^3+6/x^2+12/x+8)^(1/3) '    %    create the expression
f=
    (1/x^3+6/x^2+12/x+8)^(1/3)

>> simple(f)    %    simplify it
simplify ：
    (2*x+1)/x
ans=
    (2*x+1)/x

>> simple(ans)    %    try it once again - another method may help
combine(trig)
    2+1/x
ans=
    2+1/x
```

正如所见，**simple**试用了几种可简化表达式的简化方式，并让看到每一个尝试的结果。有时，它帮助多次使用函数**simple**并对第一次的结果作不同的简化操作，如上所作。**simple**对于含有三角函数的表达式尤为有用。让我们试一下$\cos(x)+\sqrt{-\sin(x)^2}$

```
>> simple( ' cos(x)+sqrt(-sin(x)^2) ' )    %    simplify a trig expression
simplify：
    cos(x)+(cos(x)^2-1)^(1/2)
radsimp：
    cos(x)+i*sin(x)
combine(trig)：
    cos(x)+(-1/2+1/2*cos(2*x))^(1/2)
factor：
    cos(x)+(-sin(x)^2)^(1/2)
expand：
    cos(x)+(-sin(x)^2)^(1/2)
convert(exp)：
    1/2*exp(i*x)+1/2/exp(i*x)+1/4*4^(1/2)*(exp(i*x)-1/exp(i*x))
convert(tan)：
    (1-tan(1/2*x)^2)/(1+tan(1/2*x)^2)+(-4*tan(1/2*x)^2/(1+tan(1/2*)^2)^2)^(1/2)
ans =
    cos(x)+i*sin(x)

>> simple(ans)   %    one   more   time
convert(exp)：
                exp(i*x)
convert(tan)：
                (1-tan(1/2*x)^2)/(1+tan(1/2*x)^2)+2*i*tan(1/2*x)/(1+tan(1/2*x)^2)
ans =
                exp(i*x)
```

22.7　可变精度算术运算

因为数值的精度受每次操作所保留的数位的限制，所以数值的任何运算都会引入舍入误差，重复的多次数值运算会造成累积误差。而对符号表达式的运算是非常准确的，因为它们不需要进行数值运算，所以无舍入误差。对符号运算结果用函数**eval**或**numeric**，仅在结果转换时会引入舍入误差。

MATLAB对数的处理完全依靠计算机的浮点算术运算，显然在内存中进行运算，又快又好，只是浮点运算受到所支持字长的限制，每次操作会引入舍入误差，所以不能产生精确的结果。MATLAB中各个算术运算的相对精度大约是16位。相反，Maple的符号处理能力可以实现任何数位的运算。当缺省的数位增加时，每次计算就需要附加时间和计算机内存。

Maple缺省为16位的精度。函数**digits**返回全局**Digits**参数的当前值。Maple缺省准确度可以由**digits(n)**来改变，其中n是所期望的准确度数位。用这种方法增加准确度的副作用是，每个Maple函数随后进行的计算都以新的准确度为准，增加了计算时间。结果的显示不会改变，只有所用的Maple函数的缺省准确度受到影响。

另外有一个函数，它可以任何精度实行单个计算，而使全局的**Digits**参数不变。即：可变精度的算术或函数**vpa**，它以缺省的精度或任何指定的精度对单个符号表达式进行计算，并以同样的精度来显示结果。

```
>> format long    %    let ' s see all the usual digits

>> pi    %    how about  π  to numeric accuracy
ans=
  3.14159265358979

>> digits    %    display the default    ' Digits '    value
Digits=16

>> vpa(' pi ')    %    how about  π  to    ' Digits '    value
ans=
  3.141592653589793

>> digits(18)    %    change the default to 18 digits

>> vpa(' pi ')    %    how about  π  to    ' Digits '    accuracy
ans=
  3.14159265358979324

>> vpa( ' pi ' ，20)    %    how about    π  to 20 digits
ans=
  3.1415926535897932385

>> vpa( ' pi ' ，50)    %    how about  π  to 50 digits
ans=
  3.14159265358979323846264338327950288419716939937751

>> vpa( ' 2^(1/3) ' ，200)     %    the cube root of 2 to 200 digits
ans=
1.25992104989487316476721060727822835057025146470150798008197511215529967651395948372939656243625509415431025603561566525939900240406137372284591103042
693552469606426166250009774745265654803068671854055
```

将函数**vpa**作用于符号矩阵，对它的每一个元素进行计算也同样达到所指定的位数。

```
>> A=sym( ' [1/4，log(sqrt(2))；exp(1)，3/7] ' )
A=
     [     1/4  ，   log(sqrt(2))]
     [exp(1)  ，              3/7)]

>> vpa(A，20)    %      evaluate to 20 digits
ans=
```

```
[ .2500000000000000000,    .34657359027997265471]
[2.7182818284590452354,    .42857142857142857143]
```

22.8 方程求解

用MATLAB所具有的符号工具可以求解符号方程。有一些工具已经在前面介绍过，更多将在本节予以检验。

求解单个代数方程

我们在前面已经看到，MATLAB具有求解符号表达式的工具。如果表达式不是一个方程式(不含等号)，则在求解之前函数**solve**将表达式置成等于0。

```
>> solve( ' a*x^2+b*x+c ' )   %   solve for the roots of the quadratic eqution
ans=
    [1/2/a*(-b+(b^2-4*a*c)^1/2)]
    [1/2/a*(-b-(b^2-4*a*c)^1/2)]
```

结果是符号向量，其元素是方程的2个解。如果想对非缺省x变量求解，**solve**必须指定变量。

```
>> solve( ' a*x^2+b*x+c '  ,  ' b ' )   %   solve for b
ans=
    -(a*x^2+c)/x
```

带有等号的符号方程也可以求解。

```
>> f=solve( ' cos(x)=sin(x) ' )   %   solve for x
f=
    1/4*pi

>> t=solve( ' tan(2*x)=sin(x) ' )
t=
    [                                    0]
    [acos(1/2+1/2*3^(1/2))]
    [acos(1/2=1/2*3^(1/2))]
```

并得到数值解。

```
>> numeric(f)
ans=
    0.7854

>> numeric(t)
```

```
ans=
      0
      0 + 0.8314i
     1.9455
```

注意在求解周期函数方程时，有无穷多的解。在这种情况下，**solve**对解的搜索范围限制在接近于零的有限范围，并返回非唯一的解的子集。

如果不能求得符号解，就计算可变精度解。

```
>> x=solve( ' exp(x)=tan(x) ' )
x=
     1.306326940423079
```

代数方程组求解

可以同时求解若干代数方程，语句**solve(s1，s2，.....，sn)**对缺省变量求解**n**个方程，语句**solve(s1，s2，...，sn，'v1，v2，...，vn ')**对**n**个**' v1，v2，...vn** '的未知数求解**n**个方程。

如何处理中小学典型的代数问题？

黛安娜(Diane)想去看电影，她从小猪存钱罐倒出硬币并清点，她发现：

10美分的硬币数加上5美分的硬币总数的一半等于25美分的硬币数。

1美分的硬币数比5美分、10美分以及25美分的硬币总数多10。

25美分和10美分的硬币总数等于1美分的硬币数加上1/4的5美分的硬币数

25美分的硬币数和1美分的硬币数比5美分的硬币数加上8倍的10美分的硬币数多1。

如果电影票价为3. 00美元，爆米花为1. 00美元，糖棒为50美分，她有足够的钱去买这三样东西？

首先，根据以上给出的信息列出一组线性方程，假如*p*，*n*，*d*和*q*分别表示1美分，5美分，10美分，和25美分的硬币数

$$d+\frac{n+p}{2}=q \qquad p=n+d+q-10 \qquad q+d=p+\frac{n}{4}$$

$$q+p=n+8d-1$$

然后，建立MATLAB符号方程并对变量求解。

```
>> eq1= ' d+(n+p)/2=q ' ；

>> eq2= ' p=n+d+q-10 ' ；

>> eq3= ' q+d=p+n/4 ' ；
```

```
>> eq4= ' q+p=n+8*d-1 ' ；

>>[pennies，nickles，dimes，quarters]=solve(equ1，equ2，equ3，equ4，'p，n，d，q ')
pennies=
     16
nickles=
     8
dimes=
     3
quarters=
     15
```

所以，黛安娜有16枚1美分的硬币，8枚5美分的硬币，3枚10美分的硬币，15枚25美分的硬币，这就意味着

```
>> money=.01*16+.05*8+.10*3+.25*15
money=
     4.6100
```

她就有足够的钱去买电影票，爆米花和糖棒并剩余11美分。

单个微分方程

常微分方程有时很难求解，MATLAB提供了功能强大的工具，可以帮助求解微分方程。函数**dsovle**计算常微分方程的符号解。因为我们要求解微分方程，就需要用一种方法将微分包含在表达式中。所以，**dsovle**句法与大多数其它函数有一些不同，用字母**D**来表示求微分，**D2**，**D3**等等表示重复求微分，并以此来设定方程。任何**D**后所跟的字母为因变量。方程d^2y/dx^2=0用符号表达式D2y=0来表示。独立变量可以指定或由**symvar**规则选定为缺省。例如，一阶方程*dy/dx*=1+y^{2}的通解为：

```
>> dsolve( ' Dy=1+y^2 ' )   %   find the general solution
ans=
     -tan(-x+C1)
```

其中，**C1**是积分常数。求解初值*y(0)=1*的同一个方程就可产生：

```
>>   dsolve(' Dy=1+y^2 '，' y(0)=1 ')   %   add an initial condition
y=
     tan(x+1/4*pi)
```

独立变量可用如下形式指定：

```
>>   dsolve(' Dy=1+y^2 '，' y(0)=1 '，' v ')   %   find solution to dy/dv
```

```
ans=
      tan(v+1/4*pi)
```

让我们举一个二阶微分方程的例子，该方程有两个初始条件：

$$\frac{d^2y}{dx^2}=\cos(2x)-y \qquad \frac{dy}{dx}(0)=0 \quad y(0)=1$$

```
>> y=dsolve(' D2y=cos(2*x)-y ', ' Dy(0)=0 ', ' y(0)=1 ')
y=
      -2/3*cos(x)^2+1/3+4/3*cos(x)

>> y=simple(y)   %   y looks like it can be simplified
y=
      -1/3*cos(2*x)+4/3*cos(x)
```

通常，要求解的微分方程含有一阶以上的项，并以下述的形式表示：

$$\frac{d^2y}{dx^2}-2\frac{dy}{dx}-3y=0$$

通解为：

```
>> y=solve( 'D2y-2Dy-3*y=0 ')
y=
      C1*exp(-x)+C2*exp(3*x)
```

加上初始条件：*y(0)=0*和*y(1)=1*可得到：

```
>> y=solve( ' D2y-2Dy-3*y=0 ' ,  ' y(0)=0，y(1)=1 ' )
y=
      1/(exp(-1)-exp(3))*exp(-x)-1/(exp(-1)-exp(3))*exp(3*x)

>> y=simple(y)   %   this looks like a candidate for simplification
y=
      -(exp(-x)-exp(3*x))/(exp(3)-exp(-1))

>> pretty(y)   %      pretty it up
            exp(-x)-exp(3 x)
      - ----------------------
              exp(3) -exp(-1)
```

现在来绘制感兴趣的区域内的结果。

```
>> ezplot(y，[-6 2])
```

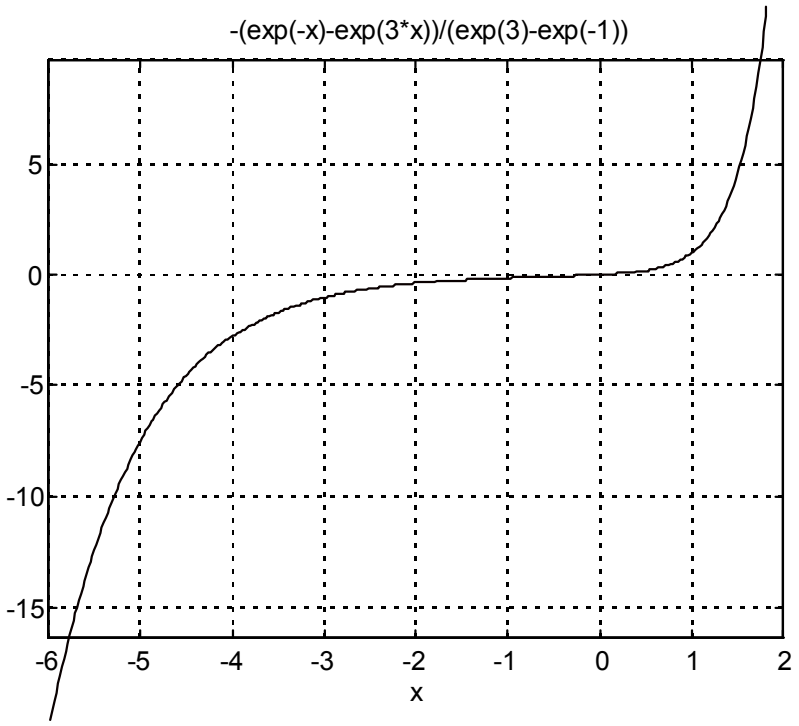


图22.3　符号函数y=-(exp(-x)-exp(3*x))/(exp(3)-exp(-1))　(-6≤x≤2)

微分方程组

函数**dsolve**也可同时处理若干个微分方程式，下面有两个线性一阶方程。

$$\frac{dy}{dx}=3f+4g \qquad \frac{dg}{dx}=-4f+3g$$

通解为：

```
>> [f，g]=dsolve('Df=3*f+4*g'，'Dg=-4*f+3*g')
f=
    C1*exp(3*x)*sin(4*x)+C2*exp(3*x)*cos(4*x)
g=
    -C2*exp(3*x)*sin(4*x)+C1*exp(3*x)*cos(4*x)
```

加上初始条件：*f(0)=0*和*g(0)=1*，我们可以得到：

```
>> [f，g]=dsolve('Df=3*f+4*g'，'Dg=-4*f+3*g'，'f(0)=0，g(0)=1')
f=
  exp(3*x)*sin(4*x)
g=
  exp(3*x)*cos(4*x)
```

22.9 线性代数和矩阵

在本节中，我们将介绍符号矩阵和MATLAB提供的工具，它用线性代数求解问题。

符号矩阵

符号矩阵和向量是数组，其元素为符号表达式，可用函数**sym**来产生。

```
>> A=sym( ' [a，  b，  c；  b，  c，  a；  c，  a，  b] ' )
A=
        [ a，  b，  c]
        [ b，  c，  a]
        [ c，  a，  b]

>> G=sym( ' [cos(t)，  sin(t)；-sin(t)，cos(t)] ' )
G=
        [ cos(t)，  sin(t)]
        [-sin(t)，  cos(t)]
```

函数**sym**也可以扩展成定义各元素的公式。**注意，只有在这种情况下，i，j分别表示行列的位置；且不影响i，j的缺省值**（它代表 $\sqrt{-1}$ ）。下面的例子建立了3×3的矩阵，其元素依赖于行和列的位置。

```
>> S=sym(3，3，' (i+j)+(i-j+s) ')   %  create a matrix using a formula
S=
  [       2/s，      3/(-1+s)，      4/(-2+s)]
  [1/(1+s)，        4/s，        s/(-1+s)]
  [4/(2+s)，      5/(1+s)，        6/s]

>> S=sym(3，  3，  ' m ',  ' n ',  ' (m-n)/(m-n-t) ')  %  use m and n in another formula
S=
        [          0，      -1/(-1-t)，      -2/(-2-t)]
        [1/(1-t)，          0，      -1/(-1-t)]
        [2/(2-t)，      1/(1-t)，          0]
```

函数**sym**也可以把数值矩阵转换成符号形式

```
>> M=[1.1，  1.2，  1.3；  2.1，  2.2，  2.3；  3.1，  3.2，  3.3]  %  a numeric matrix
M=
        1.1000    1.2000    1.3000
        2.1000    2.2000    2.3000
```

```
                3.1000      3.2000      3.3000

>> S=sym(M)   %   convert to symbolic form
S=
                [11/10，      6/5，      13/10]
                [21/10，     11/5，      23/10]
                [31/10，     16/5，      33/10]
```

如果数值矩阵的元素可以指定为小的整数之比，则函数**sym**将采用有理分式表示。如果元素是无理数，则在符号形式中**sym**将用符号浮点数表示元素。

```
>> E=[exp(1) sqrt(2)]
E=
    2.7183    1.4142

>> sym(E)
ans=
    [3060513257434036*2^(-50)，          3184525836262886*2^(-51)]
```

用函数**symsize**可以得到符号矩阵的大小(行，列数)。函数返回数值或向量，而不是符号表达式。**symsize**的四种形式说明如下：

```
>> S=sym(' [a，b，c；d，e，f] ')   %   create a symbolic matrix
S=
   [a，b，c]
   [d，e，f]

>> d=symsize(S)   %   return the size of S as the 2-element vector d
d=
   2      3

>> [m，n]=symsize(S)   %   return the number of rows in m，  and column in n
m=
   2
n=
   3

>> m=symsize(S，1)   %   return the number of rows
m=
   2

>> n=symsize(S，2)   %   return the number of columns
n=
   3
```

数值数组用**N(m，n)**形式来访问单个元素，但符号数组元素必须用函数如**sym(S，m，n)**来获取。若能用相同的句法当然好，但MATLAB符号表达式的表示不方便。在内部，符号数组表示成一个字符串数组；而**S(m，n)**返回单个字母。所以，符号数组的各个元素，必须由符号函数，如**sym**，来指定,而不是直接指定。

```
>> G=sym(' [ab，cd；ef，gh] ')  %  create a 2-by-2 symbolic matrix
G=
  [ab，  cd]
  [ef ，  gh]

>> G(1，2)  %  this is the second character of the first row of G
ans=
  a
>> r=sym(G，1，2)  %  this is the second expressinon in the first row of G
r=
  cd
```

记住，在上例中的符号矩阵**G**，实际是以2×7字符数组存储在计算机中。第一行为' [ab，cd] '，所以第二个元素是' a '。

最后，**sym**可以用于改变符号数组的一个元素。

```
>> sym(G，2，2，' pq ')  %  change the (2，2) element in G from  ' gh '  to  ' pq
'
ans=
  [ab，   cd]
  [ef，    pq]
```

代数运算

用函数**symadd**，**symsub**，**symmul**和**symdiv**，对符号矩阵可以执行许多通用的代数运算，用**sympow**可计算乘幂，用**transpose**计算符号矩阵的转置。

```
>> G=sym(' [cos(t)，sin(t0；-sin(t0，cos(t)] ')  %  create a symbolic matrix
G=
  [ cos(t)，   sin(t)]
  [ -sin(t)，  cos(t)]

>> symadd(G，  ' t ' )  %  add  ' t '  to each element
ans=
  [ cos(t)+t，        sin(t)+t]
  [ -sin(t)+t，       cos(t)+t]

>> symmul(G，G)  %  multiply G by G；  Synpow(G，2) does the same thing
```

```
ans=
  [cos(t)^2-sint(t)^2，              2*cos(t)*sin(t)]
  [ -2*cos(t)* sin(t)，          cos(t)^2-sin(t)^2]

>> simple(G)    %    try to simplify
ans=
  [cos(2*t)，     sin(2*t)]
  [-sin(2*t)，   cos(2*t)]
```

下面通过证明**G**的转置是它的逆来证明G是正交阵。

```
>> I=symmul(G，transpose(G))    %    multiply G by its transpose
I=
  [cos(t)^2+sin(t)^2，                                    0]
  [                                  0，   cos(t)^2+sin(t)^2]
>> simplify(I)    %    there appears to be a trig identity here
ans=
  [1，   0]
  [0，   1]
```

正如所期望的那样，这是单位阵。

线性代数运算

用函数**inverse**和**determ**，可计算符号矩阵的逆阵以及行列式。

```
>> H=sym(hilb(3))    %    the symbolic form of the numeric 3-by-3 Hilbert matrix
H=
  [     1，   1/2，   1/3]
  [1/2，   1/3，   1/4]
  [1/3，   1/4，   1/5]

>> determ(H)    %    dind the determinant of H
ans=
  1/2160

>> J=inverse(H)    %    find the inverse of H
J=
  [     9，        -36，             30]
  [-36，       192，        -180]
  [ 30，     -180，          180]

>> determ(J)    %    find the determinant of the inverse
ans=
```

2160

用函数**linsolve**求解齐次线性方程；这是等价于基本的MATLAB的逆斜杠\算子的符号，**linsolve(A，B)**对X方阵求解矩阵方程*A*X=B*。回到以前的硬币问题：

黛安娜想去看电影，她从小猪存钱罐倒出硬币并清点，她发现：

10美分的硬币数加上5美分的硬币总数的一半等于25美分的硬币数。
1美分的硬币数比5美分、10美分以及25美分的硬币总数多10。
25美分和10美分的硬币总数等于1美分的硬币数加上1/4的5美分的硬币数
25美分的硬币数和1美分的硬币数比5美分的硬币数加上8倍的10美分的硬币数多1。。

象上次所做的那样，列出线性方程组，令*p*，*n*，*d*和*q*分别为1美分，5美分，10美分，和25美分的硬币数

$$d+\frac{n+p}{2}=q \qquad p=n+d+q-10 \qquad q+d=p+\frac{n}{4} \qquad q+p=n+8d-1$$

重新以p，n，d，q的顺序排列表达式

p/2+n/2+d-q=0　　p-n-d-q=-10　　-p-n/4+d+q=0
p-n-8d+q=1

接下来，列出方程系数的符号数组。

```
>> A=sym(' [1/2，1/2，1，-1；1，-1，-1，-1；-1，-1/4，1，1；1，-1，-8，1] ')
A=
                    [1/2，      1/2，      1，  -1]
                    [   1，       -1，     -1，  -1]
                    [  -1，   -1/4，      1，     1]
                    [   1，       -1，     -8，     1]

>> B=sym(' [0；-10；0；-1] ')   %   Create the symbolic vector B
B=
                    [   0]
                    [-10]
                    [   0]
                    [  -1]

>> X=linsolve(A，B)   %   solve the symbolic system A ' X=B for x
x=
                    [16]
                    [   8]
```

```
[   3]
[15]
```

结果是相同的，黛安娜有16枚1美分的硬币，8枚5美分的硬币，3枚10美分的硬币，15枚25美分的硬币。

其他特性

symop将其参量串接起来，并计算所得到的表达式。

```
>> f=' cos(x) '   %   create an expression
f=
     cos(x)

>> symop( ' atan( ' ， f， ' + ' ， a， ' ) ' ， ' ^2 ' )
ans=
     atan(cos(x)+a)^2
```

在用函数**symop**时，若将数组和标量混合应慎重。例如：

```
>> M=sym( ' [a b；  c d] ' )
M=
     [a，    b]
     [c，    d]

>> symop(M， ' + ' ， ' t ' )
ans=
     [a+t，       b]
     [    c，  d+t]
```

是把**t**加到**M**的对角线上。
函数**charpoly**求解矩阵的特征多项式。

```
>> G=sym(' [1，1/2；1/3，1/4] ')   %    create a symbolic matrix
G=
  [    1，  1/2]
  [1/3，  1/4]

>> charpoly(G)   %    find the characteristic polynomial of G
ans=
  x ' 2-5/4*x+1/12
```

用函数**eigensys**可以求得符号矩阵的特征根和特征向量

```
>> F=sym(' [1/2，1/4；1/4，1/2] ')  %    create a symbolic matrix
F=
  [1/2，1/4]
  [1/4，1/2]

>> eigensys(F)  %    find the eigenvalues of F
ans=
  [3/4]
  [1/4]

>> [V，E]=eigensys(F)  %    find eigenvalues E and eigenvectors v
V=
  [-1，  1]
  [ 1，  1]
E=
  [1/4]
  [3/4]
```

矩阵的约当(Jordan)标准型是特征值的对角矩阵；转换矩阵的列是特征向量。对于给定的矩阵A， **jordan(A)**求出非奇异的矩阵**V**，使得**inv(V)*A*V**成为约当标准型，函数**jordan**有两种形式

```
>> jordan(F)  %    find the Jordan form of F ，above
ans=
  [1/4，        0]
  [    0，  3/4]

>> [V，J]=jordan(F)  %    find the Jordan form and eigenvectors
V=
  [ 1/2，  1/2]
  [-1/2，  1/2]
J=
  [1/4，        0]
  [    0，    3/4]
```

标准形和特征向**V**的列是某些F可能的特征向量。

因为**F**是非奇异的，**F**的零空间的基是空矩阵，而列空间的基是单位阵。

```
>> F=sym( ' [1/2，1/4；1/4，1/  2] ' )  %    recreate F
F=
     [1/2，  1/4]
     [1/4，  1/2]

>> nullspace(F)  %  the nullspace of F is the empty matrix
```

```
ans=
     []

>> colspace(F)    %    find the column space of F
ans=
     [1,     0]
     [0,     1]
```

用函数**singvals**可求解矩阵奇异值。

```
>> A=sym(magic(3))    %    Generate a 3-by-3 matrix
A=
     [8,    1,    6]
     [3,    5,    7]
     [4,    9,    2]

>> singvals(A)    %    find the singular expressions
ans=
     [15.00000000000000]
     [6.928203230275511]
     [3.464101615137752]
```

函数**jacobian(w，v)**可相对于**v**求解**w**的雅可比(Jacobia)值。其结果的第**(i，j)**项是**df(i)/dv(j)**。注意，当**f**是标量时，**f**的雅可比值是f的梯度。

```
>> jacobian( ' u*exp(v) ' ，sym( ' u，v ' ))
ans=
     [exp(v，u*exp(v)]
```

22.10　小结

下列各表(表22.2-表22.8)综合了符号数学工具箱的特性：

表22.2

符号表达式的运算	
numeric	符号到数值的转换
pretty	显示悦目的符号输出
subs	替代子表达式
sym	建立符号矩阵或表达式
symadd	符号加法
symdiv	符号除法
symmul	符号乘法

symop	符号运算
sympow	符号表达式的幂运算
symrat	有理近似
symsub	符号减法
symvar	求符号变量

表22.3

符号表达式的简化	
collect	合并同类项
expand	展开
factor	因式
simple	求解最简形式
simplify	简化
symsum	和级数

表22.4

符号多项式	
charpoly	特征多项式
horner	嵌套多项式表示
numden	分子或分母的提取
poly2sym	多项式向量到符号的转换
sym2poly	符号到多项式向量的转换

表22.5

符号微积分	
diff	微分
int	积分
jordan	约当标准形
taylor	泰勒级数展开

表22.6

符号可变精度算术	
digits	设置可变精度
vpa	可变精度计算

表22.7

求解符号方程	
compose	函数的复合
dsolve	微分方程的求解
finverse	函数逆
linsolve	齐次线性方程组的求解
solve	代数方程的求解

表22.8

符号线性代数	
charploy	特征多项式
determ	矩阵行列式的值
eigensys	特征值和特征向量
inverse	矩阵逆
jordan	约当标准形
linsolve	齐次线性方程组的解
transpose	矩阵的转置

第23章　Internet资源

访问Internet，不论是全面存取或发送简单的电子邮件，MATLAB有可以利用的丰富资源。通过从教育机构、政府部门及商业公司直接与Internet相连,以及通过商业的Internet所提供的咨询和在线服务如：America　Online、Delphi、CompuServe等等，都可获得这些资源。

23.1　USENET新闻组

在Internet上一个最重要的信息资源和论坛就是一个称之为NetNews FeeD、USENET或简称为News的公告栏系统。它是一个巨大的消息或公告的集合，在全球范围内从一个计算机到另一个计算机流动。News是由成千上万的单个的新闻公告组成，它收集在数以千计的新闻组或标题中。许多站点建立了可供利用的所有新闻组，而其它一些地方由于磁盘空间的限制或公司策略关系则更具选择性。

如果访问NetNews，就可以阅读MATLAB的其他用户发出的新闻公告或者发布你自己的评论或问题。MATLAB新闻组称为**comp.soft-sys.matlab**。订阅这个新闻组,可以通过提问、评论和从其他用户以及从**Mathworks**职员来的解答、浏览新闻。实际上，MATLAB的开发者经常向论坛发布消息。如果在用户站点上得到新闻组有限，不能找到comp.soft-sys.matlab，就请与当地的新闻管理人员联系，要求将**comp.soft-sys.matlab**包括进来。

要知道，这个新闻组是未作修饰的，而这意味任何人都可以发布自己的提问或评论,无法将不适当的公告过滤,全世界有数以千计的人阅读这个新闻组,所以发布时要慎重和适当。

如果不能访问NetNews，但具有FTP功能，则如下面讨论，可以从匿名FTP站点上得到这个新闻的公告摘要，即用目录**/pub/doc/cssm-digest/**中的**ftp.mathworks.com**。

如果没有News，且FTP不可选，但有连接到Internet的电子邮件(E-mail)，在**Mathworks**有一个邮件－新闻的网关，允许通过E-mail来向新闻组发布消息。由E-mail发往**comp.soft-sys.matlab@mathworks.com**的消息将发布到新闻组上。如果你不能阅读新闻组，则一定要在发布的消息中加入请求，使回答直接用E-mail送回给你，同时要提供E-mail地址。

23.2　匿名FTP

MATLAB用户的一个最有用的资源是由**Mathworks**维护的一个匿名FTP站点。FTP是文件传输协议，用来连接主机，从而将文件送到计算机和从计算机传出。匿名FTP站点允许用户和它们相连并传送文件，不要求用户在主机上有帐户和口令。

有几个计算机程序用用户图形界面进行FTP连接和传送文件。包括PC上的**WinFTP**，Macintosh上的**Fetch**，UNIX上的**ftptool**。如下面要讨论的，Web浏览器，如**Mosaic**，**Netscape**也用于访问这个站点。如果站点没有这些程序，就必须使用原始的面向命令行的**ftp**程序，因为原始**ftp**程序通常在UNIX工作站上可获得，PC机和Macintosh上也有。它的用法将在本节进行描述。

MATLAB FTP站点在**ftp.mathworks.com**，它包括用户提供的M文件、产品信息、含有常问问题(FAQs)、补片和诊断的文档。为了连接到站点，即**ftp**到站点，用 '**anonymous**' 名字或 '**ftp**' 登录，然后当要求口令时，输入用户的E-mail地址。

下面是一个简短的**ftp**会话样例。输入项以黑体表示。它连接到站点，列出可用的文件及目录，变换成ASCII模式以传送文本文件；检索文件**README**；关闭连接。

ftp ftp.mathworks.com
Connected to ftp.mathworks.com
220 ftp FTP server (Version wu-2.r(1) Thu April 14:25:23) ready
Name (ftp.mathworks.com:user):**anonymous**
331 Guest login ok， send your complete E-mail address as password.
Password: **user@host.maine.edu**
230
230Welcome to the Mathworks Library!
...
230 Guest lgoin ok， access restrications apply.
Remote system type is UNIX
Using bianry mode too transfer files
ftp> **dir**
200 PROT command suceessful
150 Opeing ASCII mdoe data connection for /bin/ls
226 Transfer complete.
total 27
-rx-r--r-- 1 admin daemon 488 Jun 24 1994 .welcome.msg
-rw-r--r-- 1 admin ftpusers 2172 Sep 28 1994 README
-rw-r--r-- 1 admin ftpusers 2626 Sep 28 1994 README.incoming
drwxr-xr-x 2 root daemon 512 Jun 16 1994 bin
drwxr-xr-x 2 toot daemon 512 Jun 28 1993 dev
drwxr-xr-x 2 toot daemon 512 Jun 7 1993 etc
drxwrwx-wx 38 admin 102 14336 Apr 21 17:23 incoming
lrxwrwxrwx 1 root daemon 3 Nov 6 1993 matlab->pub
drwxr-xr-x 12 admin ftpusers 512 Mar 27 15:17 pub
drwxr-xr-x 3 root daemon 512 Apr 23 1993 usr
ftp> **ascii**
200 Type set to A
ftp> **get README**

200 port command successful
150 Opening ASCII mdoe data connection for README (2172 bytes).
226 bytes received
ftp> **bye**
221 Goodbye

第1列用 ' d ' 列出的是目录，第1列有 ' - ' 是文件，第1列为 ' l ' 表示连接。它可以指向另外一个目录或其它某站点上的文件。目录列表提供的信息还包括拥有者、组、文件或目录大小、修改日期、文件或目录名称。

如果在连接时出现问题，也许是还没有访问到一个Internet名字服务器,该服务器将主机的名称翻译成IP地址。在这种情况下，试用 IP地址**144.212.100.10**来代替**ftp.mathwoeks.com**

ftp 144.21.100.10
connected to ftp.mathworks.com

fpt 程序支持大量的命令和选项。最有用的列在表23.1：

表23.1

	基本FTP命令
help	列出所有可用的**ftp**命令
help **command**	返回一个简单的命令描述
cd **/pub/contrib**	改变到远程站点上的**/pub/contrib**目录
lcd **localdir**	改变到用户的本地机上的目录
ascii	以ASCII文本模式传输文件。对自己机器上的文本文件，将行结束回车/换行变换为缺省值
binary	以二进制方式传输，不进行文本转换
get **remotefile**	从远程站点检索名为**remotefile**的文件
put **localfile**	将名为**locatefilede** 的文件传送到远程站点
dir	在远程站点列出详细的当前目录
ls	在远程站点列出当前目录，详细有限
bye，quit	退出ftp

MATLAB具有其它许多匿名FTP上都不具备的一个非常好的特性，它有能力即时地完成文件的归档和压缩。所支持的文件压缩格式有MIT的GNU中的**gzip** (**file.gz**)，标准Unix上的**compress** (**file.z**)以及在PC和Unix上的**zip** (**file.zip**)。文件的归档用标准的Unix **tar**格式(**dir.tar**)、Unix **shar**格式(**dir.sh**)和以及PC和Unix工作站的zip格式(**dir.zip**)，其中各格式对PC和Unix工作站都可用，且绝大部分也可以用于Macintosh计算机。

归档和压缩可以同时进行。如用以下命令，目录 **/pub/contrib/math**的全部内容可以在Unix**压缩的tar**格式文件中进行检索，

cd /pub/contrib
get math.tar.z

其他的组合也可以。命令

```
get  math.zip
get  math.sh
get  math.tar.gz
```

分别以**zip**文档、**shar**文档、**gzipped tar**文档来检索目录。不幸的是，对于Macintosh的用户，MATLAB站点不支持**stuffit** (**dir.sit**)文档或**BinHex** (**file.hqx**)编码。

表23.2是MATLAB站点的特性：

表23.2

MATLAB匿名FTP站点上重要的文件和目录

名称	类型	内容
/README	文本	新用户的文本信息
/README.incoming	文本	用户提交的文本信息
/incoming	目录	M文件用户意见投入箱的目录
/pub	目录	文件的主目录
/matlab	连接	连接到 **/pub**
/pub/INDEX	文本	有关目录内容的信息
/pub/NEWFILES	文本	站点上的新文件
/pub/ls-lr	文本	站点上所有文件的递归列表
/pub/ftphelp	文本	新**ftp**用户的入门指导
/pub/books	目录	与MATLAB基础有关的M文件
/pub/conference	目录	即将要召开的MATLAB会议的信息
/pub/contrib	目录	用户提供的M文件和MEX文件
/pub/doc	目录	文档，指导，帮助文件，FAQs，等等
/pub/mathworks	目录	MATLAB开发者提供的诊断和M文件
/pub/mosaic	目录	用于Unix的Mosaic的WWW浏览程序
/pub/pentium	目录	奔腾迷论文集
/pub/proceedings	目录	过去的MATLAB会议的论文集
/pub/product-info	目录	MATLAB，SIMULINK和工具箱的信息
/pub/tech-support	目录	技术支持和其他信息

如果有连接的麻烦或需要帮助，可以向**fptadmin@mathworks.com**站点的维护人员发E-mail.

23.3 全球广域网WWW

向Internet获取信息、图象、文件的最新、最容易、也是最灵活的方法是**World Wide Web**或简称为**WWW**或**Web**。它是根据请求向本地计算机提供超文本文档的站点服务器的集合。超文本文档可以与内置的图形结合在一起并连接到其它文件，只需用鼠标点击闪亮的文字或图形，就可以激活这些连接。然后，检索所连接的文档进行观察。如果愿意，也可以将文件存到自己的计算机上。有些连接可以有接到**Gopher**站点上（基于文本信息的服务器）以及全球的FTP站点。

如果有访问全面服务的Internet节点，就有这类MATLAB信息的图形界面供选择。**Mathworks**有一个WWW服务器，可以用Web的客户程序如**Mosaic**、**Netscape**或基于字符的*Lynx*来访问。用菜单项的**Open Location...**或**Open URL...**然后输入**http://www.mathworks.com**则可以连接到**Mathworks**的主页。主页是最高层的超文本文档，它包含了到其他的文档的连接。一旦连通，就可以将访问到的信息作为书签保存下来。以方便下一次的连接。

从**Mathworks**的主页，超文本连接可查询**Mathworks**公司和它的产品，在线拷贝通讯季刊，列出MATLAB的书籍（在写这本书时已超过100本）及相关M文件，职员要参加的有关商贸展览会的信息和最新的MATLAB会议论文集。还有一个常问问题及其关于MATLAB和SIMULINK解答的文档库。也可以阅读由MathWork技术支持人员提供的技术要点，包括内存管理、图形打印、MEX文件、工具箱以及如何将MATLAB与其他的软件集成的技术和技巧。

Mathworks主页也有到匿名FTP站点上的直接连接，可以浏览目录、读索引文件、并用浏览器检索文件。如果没有Web浏览器程序，在FTP站点的**/pub/mosaic**目录下可以获得许多工作站的**Mosaic**的版本；用于PC及Macintosh的**Mosaic**也可以通过匿名FTP的**ftp.ncsa.uiuc.edu**得到，而用于工作站和微机的**Netscape**可以通过**ftp.netscape.com**得到。必须明白，虽然**Mosaic**是免费 软件，但对免费使用**Netscape**，商业公司有一些限制。

23.4 MATLAB的自动电子邮件自动应答系统

对于那些不具有FTP功能，而有E-mail连接到Internet上的用户并没有排斥在外。**Mathworks** 拥有一个自动的用mail的文件服务，称为MATLIB，它从FTP站点用E-mail传输文件。向**matlab@mathworks.com**发一个E-mail，在消息主体中带有单向**help**就可以得到有关访问该服务器的更详尽的信息。MATLIB的mail文件服务是一个可以从消息主体中取得其指令信息的计算机程序，标题行被忽略掉。所以，消息句法正确十分重要。

在消息中可以用的许多指令是标准FTP命令的子集，包括：**cd**，**ls**，**dir**，**get**，**quit**。其他的命令用于指定编码、存挡、所用的压缩方式。其它还有指导**matlib**的程序，以回复不同的E-mail地址或限制回复文件的大小。

一个例子是**reply-to**命令，这是可选的。但如果用了，就可以使MATLIB服务器向E-mail地址发送由命令指定的文件。如果没有用，则向发送者的E-mail地址回复。例如：

```
reply-to user@host.maine.edu
```

则向**user@host.maine.edu**回答，而不是原来的E-mail地址。

二进制的文件不能用E-mail传送，所以在发送前要编码（转换成ASCII字符表示）。二进制文件按缺省是用**uuencode**发送。如果需要，也可以用**mime**编码来发送，也有用**compress**、

gzip、**zip**、**shar**、**tar**进行压缩或归档。例如：考虑发送到**matlib@mathworks.com**的一则消息，使用如下命令：

```
dir
cd /pub/contrib
get INDEX
get intergration.tar.z
get math.zip
cd games
get fifteens.m.gz
cd ..
get diffeq.sh
quit
```

这则脚本指示MATLIB首先检索目录，列出顶层目录。然后改到**/pub/contrib**目录，检索**INDEX**文件。然后作为受压缩的**tar**文件检索整个**/pub/contrib/integration/**目录，这是二进制文件，所以在传送之前先要编码**uuencoded**。接下来对整个**/pub/contrib/math**目录的内容，检索**uuencoded zip**文档，现在改到**games**目录，用**gzip**压缩检索M文件**fifteens.m**，最后改到下一个较高层目录**/pub/contrib**，作为**shar**文档，检索整个**diffeq**目录，然后退出。

回复限制在100K字节以内，所以较长的文件或文档要以多个E-mail的方式传送。如果E-mail大小有限制，也可以用**size**命令进一步限制消息的大小。**help**消息中有更详尽的说明。

23.5 MathWorks MATLAB文摘

MATLAB文摘是电子通报月刊，通过E-mail向订户发送。这个通报包括MATLAB的新闻，MATLAB的开发者和用户提供的文章，提示以及对用户问题的回答。想要订阅文摘，则要向**subscribe@mathworks.com**发送E-mail，申请加入邮寄名单，或在MATHLAB提示行中键入**>> subscribe**。**subscribe**命令提出问题，利用回答产生打印的申请表，该表邮寄或传真到***Mathworks***。Unix提供了自动的对申请发送E-mail的功能，而不是打印出来。在MATLAB FTP的**/pub/doc/tmw-digest**目录中，有过期的通报。

23.6 MATLAB通报

使用MATLAB任何用户，都可得到季度出版物MATLAB **News & Notes**。如果你是订户，可以收到文摘及季度通报以及免费的技术支持。不需注册就可以成为一个订户，这是免费服务的。可利用WWW站点上的表格，键入**>>subscribe**，或向**subscribe@mathworks.com**发送E-mail提出要求。通报通常有新闻、提示、克利夫·莫勒(Cleve Moler)和其他人的文章以及大事表。

23.7 MathWorks 电子邮件及网络地址

表23.3

Mathworks公司的E-mail地址	
suport@mathworks.com	技术支持
bugs@mathworks.com	错误报导
doc@mathworks.com	文档错误报导
suggest@mathworks.com	产品升级建议
service@amthworks.com	订购情况，延长许可，许可码
subscribe@mathworks.com	订户信息
info@mathworks.com	销售，价格和一般信息
micro-updates@mathworks.com	PC及Macintosh的升级
matlib@mathworks.com	mail文件服务器
digest@mathworks.com	MATLAB文摘
ftpadmin@mathworks.com	**Mathworks** FTP站点
webmaster@mathworks.com	**Mathworks**的WWW维护人员

表23.4

MATLAB的网络资源	
www.mathworks.com	WWW站点
ftp.mathworks.com	匿名FTP站点
144.212.100.10	WWW及FTP的Internet地址
novell.felk.cvut.cz	**ftp.mathworks.com**的镜象
192.108.154.33	Internet地址的镜象

表23.5

其他的网络资源	
mm@eeece.maine.edu	向作者提出有关精通MATLAB及精通MATLAB工具箱的问题和评论的E-mail地址
ftp.ncsa.uiuc.edu	**Telnet**和FTP的PC及Macintosh版本，PC、Macintosh和Unix的*Mosaic*
ftp2.netscape.com	PC Mac. Unix的**Netscape**的浏览器
sumex-aim.stanford.edu	**Mosaic**、**Telnet(FTP)**、**Fetch**、**gzip**、**zip**、**shar**、**compress**的Mac版本
sics.se	在**sumex-aim.stanford.edu**的文件镜象
gatekeeper.dec.com	大的匿名FTP站点
sunsite.unc.edu	大的匿名FTP站点
ftp.wustl.edu	大的匿名FTP站点
nic.funet.fi	大的匿名FTP站点
ftp.luth.se	大的匿名FTP站点
ftp.cdrom.com	大的匿名FTP站点
ftp.nws.edu.au	大的匿名FTP站点

精通 MATLAB

—综合辅导与指南—

西安交通大学

李人厚　张平安 等译校

一九九七年三月

前 言

这是一本有关 MATLAB 的参考书，适合于使用 MATLAB 或正在打算使用 MATLAB 的读者。本书另辟蹊径可以借助或不借助 MATLAB 随带的文档资料让读者自学 MATLAB。书中口语化的风格,使读者易于阅读。如书名所示,本书提供了读者精通 MATLAB 所需的工具。作为编程语言和可视化工具,MATLAB 具有丰富的一系列功能,可解决工程、科学计算和数学学科中许多问题。本书的基本目的是通过向读者展示如何有效地使用这些功能来帮助读者增强工作能力。由于 MATLAB 交互式的性质,书中内容以举例方式来描述。在读者阅读本书的同时,这些例子可以通过运行 MATLAB 而再现。

本书只涉及一般读者所用到的一些专题,所提供的资料可用于包括 UNIX 工作站、Macintosh 和 PC 在内的所有计算机平台。除了标准的 MATLAB 本身这部分的功能之外,书中只讨论了字符工具箱。其它更为专用的工具箱没有进行讨论。而且,没有讨论与机器有关的 MATLAB 诸方面,例如 MEX 文件的编写。

本书开发了许多 M 文件函数,它扩展了 MATLAB 的功能。在书中,作者演示了各种 MATLAB 的功能和编程技术,它们总称为精通 MATLAB 的工具箱。这些 M 文件存在软盘中,只要寄送书内的明信片,可由 MathWorks 公司免费提供。另一种办法可用 MathWork 的 FTP 获得。有关这个办法的信息,参阅 23 章。读者可写信到 **MathWorks Inc.,24 Prime Parkway ,Natick,MAO1760;** 电话:**(508)647-7000;** 传真:**(508)647-7001;email: info@mathworks.com; WWW:http://www.mathwork.com** 与 **Mathworks** 公司直接联系。

因为,作为一个软件工具,MATLAB 在不断的演变, 本书重点是 MATLAB 的版本 4.2c,其绝大部分内容同样适用于所有 MATLAB 4.x 版本。必须时, 我们指出了版本之间的区别, 而且标注了在 MATLAB 版本 5.0 中所能见到的变化。

作者鼓励大家对本书提出反馈意见:本书的最佳特点是什么?哪些地方需要作更多的工作?哪些专题应该删去?应该加上什么专题?用 email 可与我们联系 地址:mm@eece.maine.edu。

致谢 (略)

达恩.亨塞尔曼
勃鲁司.利特尔费尔特

序

商标信息(见原页)

目　录

第 5 章　关系和逻辑运算

除了传统的数学运算，MATLAB 支持关系和逻辑运算。如果你已经有了一些编程经验，就会对这些运算熟悉。这些操作符和函数的目的是提供求解真/假命题的答案。一个重要的应用是控制基于真/假命题的一系列 MATLAB 命令（通常在 M 文件中）的流程，或执行次序。

作为所有关系和逻辑表达式的输入，MATLAB 把任何非零数值当作真，把零当作假。所有关系和逻辑表达式的输出，对于真，输出为 1；对于假，输出为零。

5.1　关系操作符

MATLAB 关系操作符包括所有常用的比较。

表 5.1

关系操作符	说明
<	小于
< =	小于或等于
>	大于
> =	大于或等于
= =	等于
~ =	不等于

MATLAB 关系操作符能用来比较两个同样大小的数组，或用来比较一个数组和一个标量。在后一种情况，标量和数组中的每一个元素相比较，结果与数组大小一样。下面给出几个示例：

```
» A=1:9, B=9-A
A =
     1     2     3     4     5     6     7     8     9
B =
     8     7     6     5     4     3     2     1     0
» tf=A>4
tf =
     0     0     0     0     1     1     1     1     1
```

找出 **A** 中大于 4 的元素。0 出现在 **A<=4** 的地方，1 出现在 **A>4** 的地方。

```
» tf=(A= =B)
tf =
     0     0     0     0     0     0     0     0     0
```

找出 **A** 中的元素等于 **B** 中的元素。**注意，=和= =意味着两种不同的事：= = 比较两个变量，当它们相等时返回 1，当它们不相等时返回 0；在另一方面， = 被用来将运算的结果赋给一个变量。**

```
» tf=B-(A>2)
tf =
     8     7     5     4     3     2     1     0    -1
```

找出 **A>2**，并从 **B** 中减去所求得的结果向量。这个例子说明，由于逻辑运算的输出是 1 和 0 的数组，它们也能用在数学运算中。

```
» B=B+(B==0)*eps
B =
  Columns 1 through 7
    8.0000    7.0000    6.0000    5.0000    4.0000    3.0000    2.0000
  Columns 8 through 9
    1.0000    0.0000
```

这是一个演示，表明如何用特殊的 MATLAB 数 **eps** 来代替在一个数组中的零元素，**eps** 近似为 2.2e-16。这种特殊的表达式在避免被 0 除时是很有用的。

```
» x=(-3:3)/3
x =
   -1.0000   -0.6667   -0.3333         0    0.3333    0.6667    1.0000
» sin(x)./x
Warning: Divide by zero
ans =
    0.8415    0.9276    0.9816       NaN    0.9816    0.9276    0.8415
```

由于第四个数据是 0 ，计算函数 sin(x)/x 时给出了一个警告。由于 sin(0)/0 是没定义的，在该处 MATLAB 结果返回 **NaN**。用 **eps** 替代 0 以后，再试一次，

```
» x=x+(x==0)*eps;
» sin(x)./x
ans =
    0.8415    0.9276    0.9816    1.0000    0.9816    0.9276    0.8415
```

现在 sin(x)/x 在 x=0 处给出了正确的极限。

5.2 逻辑操作符

逻辑操作符提供了一种组合或否定关系表达式。MATLAB 逻辑操作符包括：

表 5.2

逻辑操作符	说明
&	与
|	或
~	非

逻辑操作符用法的一些例子有：

```
» A=1:9;B=9-A;
» tf=A>4
tf =
 0     0     0     0     1     1     1     1     1
```

找出 **A** 大于 4。

```
» tf=~(A>4)
tf =
    1     1     1     1     0     0     0     0     0
```

对上面的结果取非，也就是 1 替换 0，0 替换 1。

```
» tf=(A>2)&(A<6)
tf =
    0     0     1     1     1     0     0     0     0
```

在 **A** 大于 2‘与’**A** 小于 6 处返回 1。

最后，上面的功能易于产生数组来表示不连续信号，或由多段其他信号所组成的信号。基本想法是，把数组中要保持的那些值与 1 相乘，所有其他值与 0 相乘。例如，

```
» x=linspace(0, 10, 100);        %   create data
» y=sin(x) ;                     %   compute    sine
» z=(y>=0).*y ;                  %   set negative values of sin(x) to zero
» z=z+0.5*(y<0) ;                %   where sin(x) is negative add 1/2
» z=(x<=8).*z ;                  %   set values past x=8 to zero
» plot(x, z)
» xlabel(' x '),    ylabel(' z=f(x) '),    title(' A Discontinuous Signal ')
```

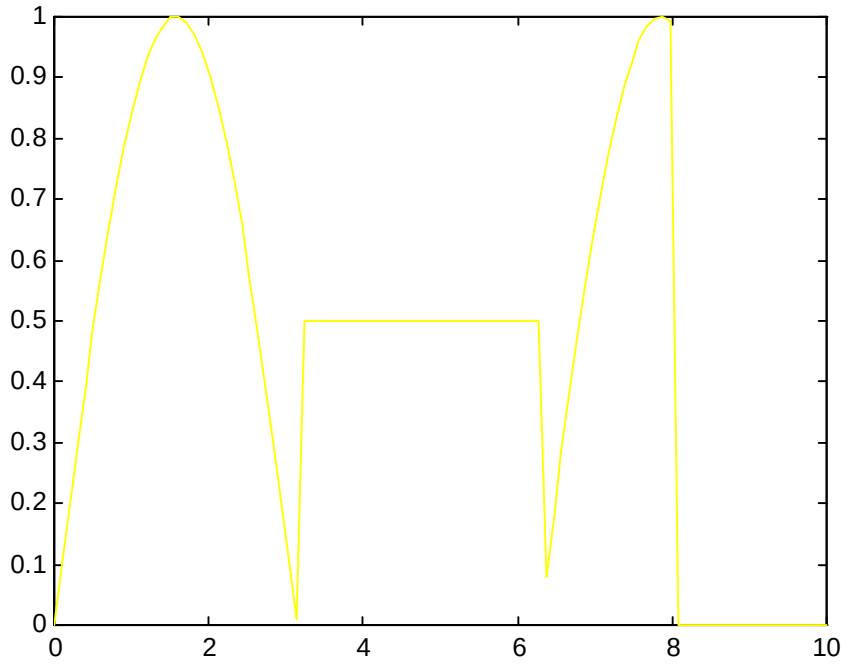


图 5.1　不连续信号

5. 3　关系与逻辑函数

除了上面的关系与逻辑操作符，MATLAB 提供了大量的其他关系与逻辑函数，包括：

表 5.3

其　他　关　系　与　逻　辑　函　数	
xor(x,y)	异或运算。x 或 y 非零(真)返回 1，x 和 y 都是零(假)或都是非零(真)返回 0。
any(x)	如果在一个向量 x 中，任何元素是非零，返回 1；矩阵 x 中的每一列有非零元素，返回 1。
all(x)	如果在一个向量 x 中，所有元素非零，返回 1；矩阵 x 中的每一列所有元素非零，返回 1。

除了这些函数，MATLAB 还提供了大量的函数，测试特殊值或条件的存在，返回逻辑值。

表 5.4

测　试　函　数	
finite	元素有限，返回真值。
isempty	参量为空，返回真值。
isglobal	参量是一个全局变量，返回真值。
ishold	当前绘图保持状态是‘ON’，返回真值。
isieee	计算机执行 IEEE 算术运算，返回真值。
isinf	元素无穷大，返回真值。
isletter	元素为字母，返回真值。
isnan	元素为不定值，返回真值。
isreal	参量无虚部，返回真值。
isspace	元素为空格字符，返回真值。
isstr	参量为一个字符串，返回真值。
isstudent	MATLAB 为学生版，返回真值。
isunix	计算机为 UNIX 系统，返回真值。
isvms	计算机为 VMS 系统，返回真值。

5.4　NaNs 和空矩阵

NaNs 和空矩阵([　])要求在 MATLAB 中作特殊处理，特别是用在逻辑或关系表达式里。根据 IEEE 数学标准，对 **NaNs** 的几乎所有运算都得出 **NaNs**。例如，

```
» a=[1    2    nan    inf    nan]     %    note,in use, NaN can be lowercase
a =
       1        2      NaN      Inf      NaN

» b=2*a
b =
       2        4      NaN      Inf      NaN

» c=sqrt(a)
c =
      1.0000       1.4142            NaN            Inf            NaN

» d = (a= =nan)
d =
       0        0        0        0        0

» f = (a~ =nan)
f =
       1        1        1        1        1
```

上面的第一和第二计算式对 **NaN** 输入给出 **NaN** 结果。然而，最后两个计算式产生有点令人惊讶的结果。当 **NaN** 与 **NaN** 相比较时，(**a= =nan**)产生全部为 0 或假的结果，同时(**a~ =nan**)产生全部 1 或真值。于是，单个 **NaNs** 相互不相等。由于 **NaNs** 的这种特性，MATLAB 有一个内置逻辑函数寻找 **NaNs**。

```
» g=isnan(a)
g =
     0     0     1     0     1
```

这个函数用 **find** 命令能找出 **NaNs** 的下标值。例如，

```
» i=find(isnan(a))          %    find indices of NaNs
i =
     3     5
» a(i)=zeros(size(i))       %    changes NaNs in a to zeros
a =
     1     2     0   Inf     0
```

当 **NaNs** 数学上由 IEEE 标准充分定义时，空矩阵由 MATLAB 的生成器确定，并有它自己的特性。空矩阵是简单的，它们是 MATLAB 大小为零的变量。

```
» size([   ])
ans =
     0     0
```

当没有其它合适的结果时，在 MATLAB 中的许多函数返回空矩阵。或许最普通的例子是函数 **find**：

```
» x=(1:5)-3         %    new data
x =
    -2    -1     0     1     2
» y=find(x>2)
y =
     [   ]
```

在这个例子里，**x** 没有包含大于 2 的值，所以没有返回下标。为了测试空结果，MATLAB 提供了逻辑函数 **isempty**。

```
» isempty(y)
ans =
     1
```

在 MATLAB 里，空矩阵不等于任何非零矩阵(或标量)。这个事实由下面例子给出：

```
» y=[    ];
» a=(y==0)
a =
        0
```

这说明一个空矩阵不等于一个标量，因此，

```
»find(y= =0)
ans =
        [    ]
```

也就是说没有返回下标。同样地，

```
b=(y~=0)
b =
        1
```

一个空矩阵不等于一个标量。但

```
» j=find(y~=0)
j =
        1
```

尽管 y 大小为零，现在有一个下标值！上面最后的一个例子是 MATLAB 中一个非文本变化，由于版本 3.5 先于版本 4.0，一个空矩阵与一个非空矩阵比较返回一个空矩阵。‘**这个新的解释通常导致产生问题，因为 y(find(y~=0))不存在。**’例如，

```
» y(find(y~=0))
???    Index exceeds matrix dimensions.
```

这里 MATLAB 报告一个错误，因为下标超出了空矩阵 y 的维数。

NaNs 和空矩阵的特性概括在表 5.5 中。

表 5.5

NaNs 和 空 矩 阵

数据	a=[1 2 nan inf nan]
表达式	结果
2*a	[2 4 NaN ∞ NaN]
(a= =nan)	[0 0 0 0 0]
(a~ =nan)	[1 1 1 1 1]
isnan(a)	[0 0 1 0 1]
y=find(a= =0)	y=[ ]
isempty(y)	1

(y= =0)	0
find(y= =0)	[ ]
(y~ =0)	1
j=find(y~ =0)	j=1
y(j)	Error! y(j) does not exist.

第 6 章 文 本

MATLAB 真正强有力的地方在于它的数值处理能力。然而，经常希望操作文本，例如把标号和标题放在图上。在 MATLAB 里，文本当作特征字符串或简单地当作字符串。

6.1 字符串

在 MATLAB 中的字符串一般是 ASCII 值的数值数组，它作为字符串表达式进行显示。例如，

```
» t=' How about this character string? '
t =
How about this character string?
» size(t)
ans =
     1    32
» whos
  Name          Size              Elements    Bytes  Density  Complex
  ans           1 by 2                   2       16  Full     No
  t             1 by 32                 32      256  Full     No
```

一个字符串是由单引号括起来的简单文本。在字符串里的每个字符是数组里的一个元素，字符串的存储要求每个字符 8 个字节，如同 MATLAB 的其它变量。因为 ASCII 字符只要求一个字节，故这种存储要求是浪费的，7/8 所分配的存储空间无用。然而，对字符串保持同样的数据结构简化 MATLAB 的内部数据结构。所给出的字符串操作并不是 MATLAB 的基本特点，但这种表达是方便和可接受的。

为了了解下面字符串的 ASCII 表达，只需对字符串执行一些算术运算。最简单和计算上最有效的方法是取数组的绝对值。例如，

```
» u=abs(t)
u =
  Columns 1 through 12
```

```
    72   111   119    32    97    98   111   117   116    32   116   104
  Columns 13 through 24
   105   115    32    99   104    97   114    97    99   116   101   114
  Columns 25 through 32
    32   115   116   114   105   110   103    63

» u=t+0
u =
  Columns 1 through 12
    72   111   119    32    97    98   111   117   116    32   116   104
  Columns 13 through 24
   105   115    32    99   104    97   114    97    99   116   101   114
  Columns 25 through 32
    32   115   116   114   105   110   103    63
```

在上面后一个例子里，加零到字符串也改变了它的 ASCII 的表示。函数 **setstr** 提供了逆转换。

```
» v=setstr(u)
v =
How about this character string?
```

因为字符串是数值数组，它们可以用 MATLAB 中所有可利用的数组操作工具进行操作。例如，

```
» u=t(16:24)
u =
character
```

字符串象数组一样进行编址。这里元素 16 到 24 包含单词 *character*

```
» u=t(24:-1:16)
u =
retcarahc
```

这是单词 *character* 的反向拼写。

```
» u=t(16:24)'
u =
c
h
a
r
a
c
```

```
t
e
r
```

用转置算子将单词 *character* 变换成一个列。

```
» v=' I can''t find the manual! '
v =
I can't find the manual!
```

字符串内的单引号是由两个连续的单引号来表示。
字符串连接可以直接从数组连接中得到。

```
» u=' If a woodchuck could chuck wood, ' ;
» v=' how much wood would a woodchuck chuck? ' ;
» w=[u    v]
w =
If a woodchuck could chuck wood, how much wood would a woodchuck chuck?
```

函数 **disp** 允许不打印它的变量名而显示一个字符串。例如，

```
»disp(u)
If a woodchuck could chuck wood,
```

注意 **u =** 语句被去掉了。这对脚本文件内显示帮助的文本有用。

如同矩阵，字符串可以有多个行，但每行必须有相同数目的列数。因此，显然要用空格以使所有行有相同长度，例如，

```
» v=[' Character strings having more    than '
      ' one row must have the same number '
      ' of column just like matrices!              ']
v =
Character strings having more than
one row must have the same number
of column just like matrices!
```

考虑下面例子，它把一个字符串转换成大写。首先，函数 **find** 用来找出小写字符的下标值，然后，从小写元素中只减去小写与大写之差，最后，用 **setstr** 把求得的数组转换成它的字符串表示。

```
» disp(u)
If a woodchuck could chuck wood,

» i=find(u>=' a ' & u<= ' z ');          %    find is a very powerful function!
```

```
» u(i)=setstr(u(i)-(' a '-' A '))
u =
IF A WOODCHUCK COULD CHUCK WOOD,
```

事实上，如在 3.7 节所描述的，矩阵能由单个下标标识。而不是由行和列下标标识，所以上面例子对字符串矩阵 **v** 也同样适用：

```
» disp(v)
Character strings having more than
one row must have the same number
of column just like matrices!

» i=find(v>=' a ' & v<=' z ');          %    here i is a single index vector into v,

» v(i)=setstr(v(i)-(' a '-' A '))          %    and matrix keeps the same orientation.
v =
CHARACTER STRINGS HAVING MORE THAN
ONE ROW MUST HAVE THE SAME NUMBER
OF COLUMN JUST LIKE MATRICES!
```

最后，当使用前面脚本文件这一章节中的函数 **input** 时，字符串是很有用的。

```
» t=' Enter number of rolls of tape > ' ;
» tape=input(t)
Enter number of rolls of tape > 5
tape =
        5
```

另外，函数 **input** 能输入一个字符串：

```
» x=input(' Enter anything > ',' s ')
Enter anything > anything can be entered
x =
anything can be entered
```

这里，在函数 **input** 里的附加参量**'s'**告诉 MATLAB，作为一个字符串，只要把用户输入传送到输出变量，就不需要引号。事实上，如果将引号包括进去，它们就变成返回字符串的一部分。

6.2 字符串转换

除了上面讨论的，字符串和它的 ASCII 表示之间转换外，MATLAB 还提供了大量的其它

的有用的字符串转换函数。它们包括：

表 6.1

字 符 串 转 换	
abs	字符串到 ASCII 转换
dec2hex	十进制数到十六进制字符串转换
fprintf	把格式化的文本写到文件中或显示屏上
hex2dec	十六进制字符串转换成十进制数
hex2num	十六进制字符串转换成 IEEE 浮点数
int2str	整数转换成字符串
lower	字符串转换成小写
num2str	数字转换成字符串
setstr	ASCII 转换成字符串
sprintf	用格式控制，数字转换成字符串
sscanf	用格式控制，字符串转换成数字
str2mat	字符串转换成一个文本矩阵
str2num	字符串转换成数字
upper	字符串转换成大写

在许多情况下，希望把一个数值嵌入到字符串中。几个字符串转换可完成这个任务。

```
» rad=2.5;   area=pi*rad^2;
» t=[' A circle of radius '   num2str(rad)   ' has an area of ' num2str(area)   ' . ' ] ;
» disp(t)
A circle of radius 2.5   has an area of   19.63.
```

这里函数 **num2str** 用来把数值转换成字符串，字符串连接用来把所转换的数嵌人到一个字符串句子中。按类似方式，**int2str** 把整数转换成字符串。无论是 **num2str** 还是 **int2str** 都调用函数 **sprintf**，它用类似 C 语言语法把数值转换成字符串。

函数 **fprintf** 经常是函数 **disp** 的一个有用替换，由于它提供了对结果更多的控制。当准备把格式化的数据写到一个文件中去时，按缺省它在**命令**窗口显示结果。例如，

```
» fprintf(' See what this does ')
See what this does»

» fprintf(' See what this does\n ')
See what this does
```

在上面第一个例子里，**fprintf** 显示字符串，然后立即给出 MATLAB 提示符。相反，在第二个例子里，**\n** 插入一个新行字符，在 MATLAB 提示符出现之前创建一个新行。

无论 **fprintf** 还是 **sprintf** 以同样方式处理输入参量，但 **fprintf** 把输出送到显示屏或文件中，而 **sprintf** 把输出返回到一个字符串中。例如，上面的例子用 **num2str** 可重写为

```
» t=sprintf(' A circle of radius %.4g has an area of %.4g. ', rad, area);
```

```
» disp(t)
A circle of radius 2.5 has an area of 19.63.

» fprintf(' A circle of radius %.4g has an area of %.4g.\n ' , rad, area)
A circle of radius 2.5 has an area of 19.63.
```

这里**%.4g** 是用在函数 **num2str** 中的数据格式。**%.4g** 就是用指数或定点标记，不管哪一种更短些，只显示至 4 位数字。除了 **g** 格式，还可用 **e** (指数)和 **f** (定点)转换。表 6.2 表明在各种不同转换下，如何显示 **pi** 结果。

表 6.2

数 值 格 式 转 换 例 子

命令	结果
fprintf(' %.0e\n ',pi)	3e+00
fprintf(' %.1e\n ',pi)	3.1e+00
fprintf(' %.3e\n ',pi)	3.142e+00
fprintf(' %.5e\n ',pi)	3.14159e+00
fprintf(' %.10e\n ',pi)	3.1415926536e+00
fprintf(' %.0f\n ',pi)	3
fprintf(' %.1f\n ',pi)	3.1
fprintf(' %.3f\n ',pi)	3.142
fprintf(' %.5f\n ',pi)	3.14159
fprintf(' %.10f\n ',pi)	3.1415926536
fprintf(' %.0g\n ',pi)	3
fprintf(' %.1g\n ',pi)	3
fprintf(' %.3g\n ',pi)	3.14
fprintf(' %.5g\n ',pi)	3.1416
fprintf(' %.10g\n ',pi)	3.141592654
fprintf(' %.8.0g\n ',pi)	3
fprintf(' %.8.1g\n ',pi)	3
fprintf(' %.8.3g\n ',pi)	3.14
fprintf(' %.8.5g\n ',pi)	3.1416
fprintf(' %.8.10g\n ',pi)	3.141592654

注意，对 **e** 和 **f** 格式，小数点右边的十进制数就是小数点右边要显示的多少位数字。相反，在 **g** 的格式里，小数点右边的十进制数指定了显示数字的总位数。另外，注意最后的五行，其结果指定为 8 个字符长度，且是右对齐。在最后一行，8 被忽略，因为指定超过了 8 位。

概括起来，当需要比缺省函数 **disp**，**num2str** 和 **int2str** 所提供的更多的控制时，**fprintf** 和 **sprintf** 是有用的。

函数 **str2mat** 把一列的几个字符串转换成一个字符串矩阵。例如，

```
» a=' one ' ; b= ' two ' ; c= ' three ' ;
» disp(str2mat(a, b, c, ' four '))
```

```
one
two
three
four
```

尽管从上面看不明显，上面的每行有同样数目的元素。较短行用空格补齐，使结果形成一个有效的矩阵

在逆方向转换中，有时是很方便的。

```
» s= ' [1    2; pi    4] '        %    a string of a MATLAB matrix
s =
[1    2; pi    4]
» str2num(s)

ans =
      1.0000       2.0000
      3.1416       4.0000

» s=' 123e+5 '          %    a string containing a simple number
s =
123e+5

» str2num(s)
ans =
      12300000
```

函数 **str2num** 不能接受用户定义的变量，也不能执行转换过程的算术运算。更多的信息，请看在线帮助。

6.3 字符串函数

MATLAB 提供了大量的字符串函数，包括列在表 6.3 当中的。

表 6.3

字　符　串　函　数	
eval(string)	作为一个 MATLAB 命令求字符串的值
eval(try,catch)	
blanks(n)	返回一个 n 个零或空格的字符串
deblank	去掉字符串中后拖的空格
feval	求由字符串给定的函数值
findstr	从一个字符串内找出字符串
isletter	字母存在时返回真值
isspace	空格字符存在时返回真值
isstr	输入是一个字符串，返回真值

lasterr	返回上一个所产生 MATLAB 错误的字符串
strcmp	字符串相同，返回真值
strrep	用一个字符串替换另一个字符串
strtok	在一个字符串里找出第一个标记

列在上面的第一个函数 **eval** 给 MATLAB 提供宏的能力。其中，该函数提供了将用户创建的函数名传给其它函数能力，以便求值。它的应用例子包括：

```
» a=eval(' sqrt(2) ')
a =
      1.4142

» eval(' a=sqrt(2) ')
a =
      1.4142
```

上面的例子演示了函数 **eval**。显然，它们不是计算 2 的平方根的最简单方法。当被求值的字符串是由子字符串连接而成，或将字符串传给一个函数以求值时，**eval** 非常有用。说明这种用途例子本书的以后会提及。

如果字符串传递到 **eval** 不能被辨认，MATLAB 提供下列语法：

```
» eval(' a=sqrtt(2) ',' a=[    ] ')
a =
      [    ]
```

这里第二个参量被执行。由于第一个参量有误，即 **sqrtt** 不是一个有效的 MATLAB 函数。这种形式经常被描述为 **eval(try,catch)**。

函数 **feval** 与 **eval** 类似，但在用法上有更多的限制。**feval(' fun ',x)**求由字符串**' fun '**给定的函数值，其输入参量是变量 x。即 **feval(' fun ',x)**等价于求 **fun(x)**值。例如，

```
» a=feval(' sqrt ' ,2)
a =
      1.4142
```

函数 **eval**，**feval** 的基本用途限在用户创建的函数内。一般地，**feval** 可求出有大量输入参量的函数值，例如，**feval(' fun ', x, y, z)** 等价于求 **fun(x, y, z)**值。

列在上面表中的许多字符串函数提供了基本的字符串语法分析能力。例如，**findstr** 返回一个在另一个字符串内字符串的起始下标值。

```
» b=' Peter Piper picked a peck of pickled peppers ' ;

» findstr(b,'    ')     %    find space
ans =
      6      12      19      21      26      29      37
```

```
» findstr(b,' p ')     %    find the letter p
ans =
       9      13      22      30      38      40      41

» find(b= = ' p ')     % for single character searches the find command works too
ans =
       9      13      22      30      38      40      41

» findstr(b, ' cow ')     %    find the word cow
ans =
       [    ]

» findstr(b,' pick ')        %    find the string pick
ans =
     13      30
```

注意这个函数对大小写是敏感的，当不匹配时，返回空矩阵。**findstr** 对字符串矩阵不起作用。

```
» strrep(b, ' p ', ' P ')          %    capitalize all p ' s
ans =
Peter PiPer Picked a Peck of Pickled PePPers

» strrep(b, ' Peter ', ' Pamela ')       %    change Peter to Pamela
ans =
Pamela Piper picked a peck of pickled peppers
```

正如上面所看到的，**strrep** 执行简单的字符串替代。**strrep** 对字符串矩阵不起作用。

函数 **strtok** 找出由特定字符指定的字符串内的标记，空格是缺省限定字符。例如，

```
» disp(b)
Peter Piper picked a peck of pickled peppers

» strtok(b)       %    find first token in above string separated by whitespace
ans =
Peter

» [c, r]=strtok(b)       %    return the remainder of the string array in r
c =
Peter
r =
Piper picked a peck of pickled peppers

» [d,s]=strtok(r) %find the next token by using the previous remainder
```

```
d =
Piper
s =
picked a peck of pickled peppers
```

用空格作为限定符，**strtok** 找出在数组中的单词。**strtok** 对字符串矩阵不起作用。

```
» [d, s]=strtok(b, ' pP ') %let delimiter be lower or upper case P
d =
eter
s =
Piper picked a peck of pickled peppers
```

如果提供一个可选的字符串，它的字符是限定符。注意在标记里，不返回限定符，但返回所有限定符之前的字符。也就是，在上面的字符串 **d = eter** 末端有一个空格。

第 7 章　决策：控 制 流

计算机编程语言和可编程计算器提供许多功能，它允许你根据决策结构控制命令执行流程。如果你以前已经使用过这些功能，对此就会很熟悉。相反，如果不熟悉控制流，本章材料初看起来或许复杂些。如果这样，就慢慢来。

控制流极其重要，因为它使过去的计算影响将来的运算。MATLAB 提供三种决策或控制流结构。它们是：For 循环，While 循环和 If-Else-End 结构。由于这些结构经常包含大量的 MATLAB 命令，故经常出现在 M 文件中，而不是直接加在 MATLAB 提示符下。

7.1　**For** 循环

For 循环允许一组命令以固定的和预定的次数重复。For 循环的一般形式是：

```
for   x = array
      {commands}
end
```

在 for 和 end 语句之间的{commands}按数组中的每一列执行一次。在每一次迭代中，**x** 被指定为数组的下一列，即在第 n 次循环中，**x=array(:, n)**。例如，

```
» for n=1:10
    x(n)=sin(n*pi/10);
```

```
 end

» x
x =
   Columns 1 through 7
     0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
   Columns 8 through 10
     0.5878    0.3090    0.0000
```

换句话，第一语句是说：对 **n** 等于 1 到 10，求所有语句的值，直至下一个 **end** 语句。第一次通过 For 循环 n=1，第二次，n=2，如此继续，直至 n=10。在 n=10 以后，For 循环结束，然后求 end 语句后面的任何命令值，在这种情况下显示所计算的 **x** 的元素。

For 循环的其它重要方面是：

1. For 循环不能用 For 循环内重新赋值循环变量 **n** 来终止。

```
» for n=1:10
     x(n)=sin(n*pi/10);
     n=10;
 end

» x
x =
   Columns 1 through 7
     0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
   Columns 8 through 10
     0.5878    0.3090    0.0000
```

2. 语句 **1 :10** 是一个标准的 MATLAB 数组创建语句。在 For 循环内接受任何有效的 MATLAB 数组。

```
» data=[3   9   45   6;   7   16   -1   5]
data =
      3      9     45      6
      7     16     -1      5

for n=data
                x=n(1)-n(2)
          end
x =
     -4
x =
     -7
x =
     46
```

```
x =
     1
```

3. For 循环可按需要嵌套。

```
for n=1:5
   for m=5:-1:1
      A(n,m)=n^2+m^2;
   end
   disp(n)
 end
     1
     2
     3
     4
     5

» A
A =
     2     5    10    17    26
     5     8    13    20    29
    10    13    18    25    34
    17    20    25    32    41
    26    29    34    41    50
```

4. 当有一个等效的数组方法来解给定的问题时，应避免用 For 循环。例如，上面的第一个例子可被重写为

```
» n=1:10;
» x=sin(n*pi/10)
x =
  Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

两种方法得出同样的结果，而后者执行更快，更直观，要求较少的输入。

5. 为了得到最大的速度，在 For 循环(While 循环)被执行之前，应预先分配数组。例如，前面所考虑的第一种情况，在 For 循环内每执行一次命令，变量 **x** 的大小增加 1。迫使 MATLAB 每通过一次循环要花费时间对 **x** 分配更多的内存。为了消去这个步骤，For 循环的例子应重写为

```
»x=zeros(1,10);       %    preallocated memory for x
» for n=1:10
```

```
        x(n)=sin(n*pi/10);
    end
```

现在，只有 **x(n)**的值需要改变。

7.2 **While** 循环

与 For 循环以固定次数求一组命令的值相反，While 循环以不定的次数求一组语句的值。While 循环的一般形式是：

```
while expression
    {commands}
end
```

只要在表达式里的所有元素为真，就执行 while 和 end 语句之间的{commands}。通常，**表达式**的求值给出一个标量值，但数组值也同样有效。在数组情况下，所得到数组的所有元素必须都为真。考虑下列例子：

```
» num=0;EPS=1;
» while (1+EPS)>1
      EPS=EPS/2;
      num=num+1;
    end

» num
num =
     53

» EPS=2*EPS
EPS =
   2.2204e-016
```

这个例子表明了计算特殊 MATLAB 值 **eps** 的一种方法，它是一个可加到 1，而使结果以有限精度大于 1 的最小数值。这里我们用大写 **EPS**，因此 MATLAB 的 **eps** 的值不会被覆盖掉。在这个例子里，**EPS** 以 1 开始。只要(1+**EPS**)>1 为真(非零)，就一直求 While 循环内的命令值。由于 **EPS** 不断地被 2 除，**EPS** 逐渐变小以致于 **EPS**+1 不大于 1。(记住，发生这种情况是因为计算机使用固定数的数值来表示数。MATLAB 用 16 位，因此，我们只能期望 **EPS** 接近 10^{-16}。)在这一点上，(1+**EPS**)>1 是假(零)，于是 While 循环结束。最后，**EPS** 与 2 相乘，因为最后除 2 使 **EPS** 太小。

7.3 **IF-ELSE-END** 结构

很多情况下，命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种 If-Else-End 结构来提供。最简单的 If-Else-End 结构是：

```
if    expression
        {commands}
end
```

如果在表达式中的所有元素为真(非零)，那么就执行 if 和 end 语言之间的{commands}。在表达式包含有几个逻辑子表达式时，即使前一个子表达式决定了表达式的最后逻辑状态，仍要计算所有的子表达式。例如，

```
» apples=10;              %    number of apples
» cost=apples*25         %    cost of apples
cost =
    250

» if apples>5             %    give 20% discount for larger purchases
     cost=(1-20/100)*cost;
 end

» cost
cost =
    200
```

假如有两个选择，If-Else-End 结构是：

```
if    expression
      commands evaluated if True
else
      commands evaluated if False
end
```

在这里，如果表达式为真，则执行第一组命令；如果表达式是假，则执行第二组命令。

当有三个或更多的选择时，If-Else-End 结构采用形式

```
if expression1
    commands evaluated if expression1 is True
elseif expression2
    commands evaluated if expression2 is True
elseif expression3
    commands evaluated if expression3 is True
elseif expression4
    commands evaluated if expression4 is True
elseif ……
```

```
        .
        .
        .
    else
        commands evaluated if no other expression    is True
    end
```

最后的这种形式，只和所碰到的、与第一个真值表达式相关的命令被执行；接下来的关系表达式不检验，跳过其余的 If-Else-End 结构。而且，最后的 else 命令可有可无。

现在我们知道了如何用 If-Else-End 结构来决策，就有可能提出一种合理的方法来跳出或中断 For 循环和 While 循环。

```
» EPS=1;
» for num=1:1000
     EPS=EPS/2;
     if (1+EPS)<=1
       EPS=EPS*2
         break
       end
   end
EPS =
   2.2204e-016

» num
num =
       53
```

这个例子演示了估算 **EPS** 的另一种方法。在这种情况下，For 循环构造成要执行足够多的次数。If-Else-End 结构检验要看 **EPS** 是否变得足够小。如果是，**EPS** 乘 2，**break** 命令强迫 For 循环提早结束，**num=53**。

在这个例子里，当执行 **break** 语句时，MATLAB 跳到循环外下一个语句。在现在情况下，它返回到 MATLAB 的提示符并显示 **EPS**。如果一个 **break** 语句出现在一个嵌套的 For 循环或 While 循环结构里，那么 MATLAB 只跳出 **break** 所在的那个循环，不跳出整个嵌套结构。

7.4 小结

MATLAB 控制流功能可以概括如下：

表 7.1

控　制　流　结　构	
for x = array 　　commands end	For 循环，每一次迭代将 x 赋以数组的第 i 列，执行命令

while expression 　　commands end	While 循环，只要表达式的所有元素为真或非零，执行命令。
if expression 　　commands end	简单的 If-Else-End 结构，若在表达式中的所有元素是真值或非零，执行命令。
if expression 　　commands evaluated if True else expression 　　commands evaluated if False end	具有两条路径的 If-Else-End 结构，若表达式为真或非零，则执行一组命令。若表达式为假或零，则执行另一组命令。
if expression1 　commands evaluated if expression1 is True elseif expression2 　commands evaluated if expression2 is True if expression3 　commands evaluated if expression3 is True elseif …… • • • else 　　commands evaluated if no other 　　expression is True end	最一般的 If-Else-End 结构。只执行与第一个真值表达式相关的命令。
break	结束 For 循环和 While 循环的执行

7.5 M 文件举例

为了演示 If-Else-End 结构，考虑**精通 MATLAB 工具箱**中函数 **mmono**，它检查一个向量的单调性。

```
» mmono(1:12)        %    strictly increasing input
ans =
       2

» mmono([1:12    12    13:24])        %    non decreasing input
ans =
       1

» mmono([1    3    2    -1])        %    not monotonic in any sense
ans =
       0
```

```
» mmono([12:-1:0    0    -1])        %    non increasing
ans =
      -1

» mmono(12:-1:0)         %    strictly decrasing
ans =
      -2
```

这个**精通 MATLAB 工具箱**的函数主体给出如下：

```
function f=mmono(x)
%    MMONO Test for monotonic vector.
%    MMONO(x) where x is a vector return:
%       2 if x is strictly increasing,
%       1 if x is non decreasing,
%      -1 if x is non increasing,
%      -2 if x is strictly decreasing,
%       0 otherwise.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

x=x(:);              %    make x a column vector
y=diff(x);           %    find differences between consecutive elements

if all(y>0)          %    test for strict first
                  f=2;
elseif all(y>=0)
                  f=1;
elseif all(y<0)      %    test for strict first
                  f=-2;
elseif all(y<=0)
                  f=-1;
else
                  f=0;      %    otherwise response
end
```

函数 **mmono** 直接利用了 If-Else-End 结构。由于严格单调是一般单调的子集，首先检验严格的单调是必要的，因为在所碰见的第一个真值分支里，其语句执行之后，结构就结束。

第 8 章　M 文 件 函 数

使用 MATLAB 函数时，例如 **inv**, **abs**, **angle** 和 **sqrt**，MATLAB 获取传递给它的变量，利用所给的输入，计算所要求的结果。然后，把这些结果返回。由函数执行的命令，以及由这些命令所创建的中间变量，都是隐含的。所有可见的东西是输入和输出，也就是说函数是一个黑箱。

这些属性使得函数成为强有力的工具，用以计算命令。这些命令包括在求解一些大的问题时，经常出现的有用的数学函数或命令序列。由于这个强大的功能，MATLAB 提供了一个创建用户函数的结构，并以 M 文件的文本形式存储在计算机上。MATLAB 函数 fliplr 是一个 M 文件函数良好的例子。

```
function y = fliplr(x)
%   FLIPLR      Flip matrix in the left/right direction.
%    FLIPLR(X) returns X with row preserved and columns flipped
%    in the left/right direction.
%
%    X = 1    2    3        becomes    3    2    1
%          4    5    6                    6    5    4
%
%    See also FLIPUD, ROT90.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

[m, n] = size(x);
y = x(: , n : -1 : 1);
```

一个函数 M 文件与脚本文件类似之处在于它们都是一个有**.m** 扩展名的文本文件。如同脚本 M 文件一样，函数 M 文件不进入**命令**窗口，而是由文本编辑器所创建的外部文本文件。一个函数的 M 文件与脚本文件在通信方面是不同的。函数与 MATLAB 工作空间之间的通信，只通过传递给它的变量和通过它所创建的输出变量。在函数内中间变量不出现在 MATLAB 工作空间，或与 MATLAB 工作空间不交互。正如上面的例子所看到的，一个函数的 M 文件的第一行把 M 文件定义为一个函数，并指定它的名字。它与文件名相同，但没有**.m** 扩展名。它也定义了它的输入和输出变量。接下来的注释行是所展示的文本，它与帮助命令： **» help fliplr** 相对应。第一行帮助行称为 H1 行，是由 **lookfor** 命令所搜索的行。最后，M 文件的其余部分包含了 MATLAB 创建输出变量的命令。

8.1 规则和属性

M 文件函数必须遵循以下特定的规则。除此之外，它们有许多的重要属性。包括：

1. 函数名和文件名必须相同。例如，函数 **fliplr** 存储在名为 **fliplr.m** 文件中。

2. MATLAB 头一次执行一个 M 文件函数时，它打开相应的文本文件并将命令编辑成存储器的内部表示，以加速执行以后所有的调用。如果函数包含了对其它 M 文件函数的引用，它们也同样被编译到存储器。普通的脚本 M 文件不被编译，即使它们是从函数 M 文件内调

用；打开脚本 M 文件，调用一次就逐行进行注释。

3. 在函数 M 文件中，到第一个非注释行为止的注释行是帮助文本。当需要帮助时，返回该文本。例如， **» help fliplr** 返回上述前八行注释。

4. 第一行帮助行，名为 H1 行，是由 **lookfor** 命令搜索的行。

5. 函数可以有零个或更多个输入参量。函数可以有零个或更多个输出参量。

6. 函数可以按少于函数 M 文件中所规定的输入和输出变量进行调用，但不能用多于函数 M 文件中所规定的输入和输出变量数目。如果输入和输出变量数目多于函数 M 文件中 **function** 语句一开始所规定的数目，则调用时自动返回一个错误。

7. 当函数有一个以上输出变量时，输出变量包含在括号内。例如， **[V,D] = eig(A)**。不要把这个句法与等号右边的**[V,D]** 相混淆。右边的**[V,D]** 是由数组 **V** 和 **D** 所组成。

8. 当调用一个函数时，所用的输入和输出的参量的数目，在函数内是规定好的。函数工作空间变量 **nargin** 包含输入参量个数；函数工作空间变量 **nargout** 包含输出参量个数。事实上，这些变量常用来设置缺省输入变量，并决定用户所希望的输出变量。例如，考虑 MATLAB 函数 **linspace** ：

```
function y = linspace(d1, d2, n)
%   LINSPACE Linearly spaced vector.
%    LINSPACE(x1, x2) generates a row vector of 100 linearly
%    equally spaced points between x1 and x2.
%    LINSPACE(x1, x2, N) generates N points between x1 and x2.
%
%    See also LOGSPACE, :.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

if nargin == 2
     n = 100;
end
y = [d1+(0:n-2)*(d2-d1)/(n-1) d2] ;
```

这里，如果用户只用两个输入参量调用 **linspace** ，例如 **linspace(0,10)** ，**linspace** 产生 100 个数据点。相反，如果输入参量的个数是 3，例如，**linspace(0,10,50)**，第三个参量决定数据点的个数。

可用一个或两个输出参量调用的函数的一个例子是 MATLAB 函数 **size**。尽管这个函数不是一个 M 文件函数(它是一个内置函数)，**size** 函数的帮助文本说明了它的输出参量的选择。

```
SIZE  Matrix dimensions.
     D = SIZE(X), for M-by-N matrix X, returns the two-element
     row vector D = [M, N] containing the number of rows and columns
     in the matrix.

     [M, N] = SIZE(X) returns the number of rows and columns
     in separate output variables.
```

如果函数仅用一个输出参量调用，就返回一个二元素的行，它包含行数和列数。相反，如果出现两个输出参量，size 分别返回行和列。在 M 文件函数里，变量 **nargout** 可用来检验输出参量的个数，并按要求修正输出变量的创建。

9. 当一个函数说明一个或多个输出变量，但没有要求输出时，就简单地不给输出变量赋任何值。MATLAB 函数 **toc** 阐明了这个属性。

```
function t = toc
%   TOC   Read the stopwatch timer.
%    TOC, by itself, prints the elapsed time since TIC was used.
%    t = TOC; saves the elapsed time in t, instead of printing it out.
%
%    See also TIC, ETIME, CLOCK, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TOC uses ETIME and the value of CLOCK saved by TIC.
global TICTOC
if nargout < 1
    elapsed_time = etime(clock, TICTOC)
else
    t = etime(clock, TICTOC);
end
```

如果用户用不以输出参量调用 **toc** ，例如， » toc，就不指定输出变量 **t** 的值，函数在**命令**窗口显示函数工作空间变量 **elapsed_time** ，但在 MATLAB 工作空间里不创建变量。相反，如果 toc 是以 **» out=toc** 调用，则按变量 **out** 将消逝的时间返回到**命令**窗口。

10. 函数有它们自己的专用工作空间，它与 MATLAB 的工作空间分开。函数内变量与 MATLAB 工作空间之间唯一的联系是函数的输入和输出变量。如果函数任一输入变量值发生变化，其变化仅在函数内出现，不影响 MATLAB 工作空间的变量。函数内所创建的变量只驻留在函数的工作空间，而且只在函数执行期间临时存在，以后就消失。因此，从一个调用到下一个调用，在函数工作空间变量存储信息是不可能的。(然而，如下所述，使用全局变量就提供这个特征。)

11. 如果一个预定的变量，例如，**pi，** 在 MATLAB 工作空间重新定义，它不会延伸到函数的工作空间。逆向有同样的属性，即函数内的重新定义变量不会延伸到 MATLAB 的工作空间中。

12. 当调用一个函数时，输入变量不会拷贝到函数的工作空间，但使它们的值在函数内可读。然而，改变输入变量内的任何值，那么数组就拷贝到函数工作空间。进而，按缺省，如果输出变量与输入变量相同，例如，函数 **x=fun(x, y, z)** 中的 **x** ，那么就将它拷贝到函数的工作空间。因此，为了节约存储和增加速度，最好是从大数组中抽取元素，然后对它们作修正，而不是使整个数组拷贝到函数的工作空间。

13. 如果变量说明是全局的,函数可以与其它函数、MATLAB 工作空间和递归调用本身共享变量。为了在函数内或 MATLAB 工作空间中访问全局变量，在每一个所希望的工作空间，变量必须说明是全局的。全局变量使用的例子可以在 MATLAB 函数 **tic** 和 **toc** 中看到，它们合在一起工作如一个跑表。

```
function tic
%   TIC    Start a stopwatch timer.
%    The sequence of commands
%           TIC
%           any stuff
%           TOC
%    prints the time required for the stuff.
%
%    See also TOC, CLOCK, ETIME, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TIC simply stores CLOCK in a global variable.
global TICTOC
TICTOC = clock;



function t = toc
%   TOC   Read the stopwatch timer.
%    TOC, by itself, prints the elapsed time since TIC was used.
%    t = TOC; saves the elapsed time in t, instead of printing it out.
%
%    See also TIC, ETIME, CLOCK, CPUTIME.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

%   TOC uses ETIME and the value of CLOCK saved by TIC.
global TICTOC
if nargout < 1
    elapsed_time = etime(clock,TICTOC)
else
    t = etime(clock,TICTOC);
end
```

在函数 **tic** 中，变量 **TICTOC** 说明为全局的，因此它的值由调用函数 **clock** 来设定。以后在函数 toc 中，变量 **TICTOC** 也说明为全局的，让 **toc** 访问存储在 **TICTOC** 中的值。利用这个值，**toc** 计算自执行函数 **tic** 以来消逝的时间。值得注意的是，变量 **TICTOC** 存在于 **tic** 和 **toc** 的工作空间，而不在 MATLAB 工作空间。

14. 实际编程中，无论什么时候应尽量避免使用全局变量。要是用了全局变量，建议全局变量名要长，它包含所有的大写字母，并有选择地以首次出现的 M 文件的名字开头。如果遵循建议，则在全局变量之间不必要的互作用减至最小。例如，如果另一函数或 MATLAB 工作空间说明 **TICTOC** 为全局的，那么它的值在该函数或 MATLAB 工作空间内可被改变，而函数 **toc** 会得到不同的、可能是无意义的结果。

15. MATLAB 以搜寻脚本文件的同样方式搜寻函数 M 文件。例如，输入**» cow** ，MATLAB 首先认为 **cow** 是一个变量。如果它不是，那么 MATLAB 认为它是一个内置函数。如果还不是，MATLAB 检查当前 **cow.m** 的目录或文件夹。如果它不存在， MATLAB 就检查 **cow.m** 在 MATLAB 搜寻路径上的所有目录或文件夹。如需要更多的信息，请参阅本书的 2.10 节或 MATLAB 用户指南中“MATLAB 搜寻路径”。

16. 从函数 M 文件内可以调用脚本文件。在这种情况下，脚本文件查看函数工作空间，不查看 MATLAB 工作空间。从函数 M 文件内调用的脚本文件不必用调用函数编译到内存。函数每调用一次，它们就被打开和解释。因此，从函数 M 文件内调用脚本文件减慢了函数的执行。

17. 函数可以递归调用。即 M 文件函数能调用它们本身。例如，考虑一个傻函数 **iforgot**：

```
function iforgot(n)
%   IFORGOT Recursive Function Call Example

%   Copyright (c) 1996 by Prentice-Hall,Inc

if nargin==0,n=20;end
if n>1
    disp(' I will remember to do my homework. ')
    iforgot(n-1)
else
    disp(' Maybe NOT! ')
end
```

调用这个函数产生

```
» iforgot(10)
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
I will remember to do my homework.
Maybe NOT!
```

递归调用函数功能在许多应用场合是有用的。在编制要递归调用的函数时，必须确保会终止，否则 MATLAB 会陷入死循环。最后，在一个递归函数内，如果变量说明是全局的，则该全局变量对以后所有函数调用是可用的。在这个意义下，全局变量变成静态的，并在函数调用之间不会消失。

18. 当函数 M 文件到达 M 文件终点，或者碰到返回命令 **return**，就结束执行和返回。**return** 命令提供了一种结束一个函数的简单方法，而不必到达文件的终点。

19. MATLAB 函数 **error** 在**命令**窗口显示一个字符串，放弃函数执行，把控制权返回给键盘。这个函数对提示函数使用不当很有用，如在以下文件片段中：

```
if length(val)>1
    error('   VAL must be a scalar.   ')
end
```

这里，如果变量 **val** 不是一个标量，**error** 显示消息字符串，把控制权返回给**命令**窗口和键盘。

20. 当一个函数的输入参量的个数超出了规定的范围，MATLAB 函数 **nargchk** 提供了统一的响应。函数 **nargchk** 给定为：

```
function   msg = nargchk(low, high, number)
%   NARGCHK   Check number of input arguments.
%    Return error message if not between low and high.
%    If it is, return empty matrix.

%    Copyright (c) 1984-94 by The MathWorks, Inc.

msg = [   ] ;
if (number < low)
    msg = ' Not enough input arguments. ' ;
elseif (number   >   high)
    msg = ' Too many input arguments. ' ;
end
```

下列的文件片段表明了在一个 M 文件函数内的典型用法：

```
  error(nargchk(nargin, 2, 5))
```

如上所示，如果 **nargin** 的值小于 2，函数 **error** 象前面描述的那样进行处理，**nargchk** 返回字符串‘没有足够的输入参量。’。如果 **nargin** 的值大于 5，函数 **error** 执行处理，**nargchk** 返回字符串‘太多输入参量。’。如果 **nargin** 是在 2 和 5 之间，函数 **error** 简单地将控制传递给下一个语句，**nargchk** 返回一个空字符串。也就是说，当它的输入参量为空，**error** 函数什么也不做。

21. 当 MATLAB 运行时，它缓存了(**caches**)存储在 **Toolbox** 子目录和 **Toolbox** 目录内的所有子目录中所有的 M 文件的名字和位置。这使 MATLAB 很快地找到和执行函数 M 文件。也使得命令 **lookfor** 工作更快。被缓存的 M 文件函数当作是只读的。如果执行这些函数，以后又发生变化，MATLAB 将只执行以前编译到内存的函数，不管已改变的 M 文件。而且，在 MATLAB 执行后，如果 M 文件被加到 Toolbox 目录中，那么它们将不出现在缓存里，因此不可利用。所以，在 M 文件函数的使用中，最好把它们存储在 **Toolbox** 目录外，或许最好存储在 MATLAB 目录下，直至它们被认为是完备的(complete)。当它们是完备时，就将它们移到一个只读的 Toolbox 目录或文件夹的子目录内。最后，要确保 MATLAB 搜索路径改变，以确认它们的存在。

22. 在 **Toolbox** 目录外，MATLAB 跟踪 M 文件的修改日期。所以，当遇到一个以前编译到内存的 M 文件函数时，MATLAB 把已编译的 M 文件的修改日期与在磁盘上的 M 文件比较。如果日期是相同的，MATLAB 执行已编译的 M 文件。相反，如果在磁盘上的 M 文件是新的，MATLAB 清除以前已编译的 M 文件，且编译这个新的和修改过的 M 文件。

23. M 文件的缓存过程按 MATLAB 版本而稍有不同。例如，MATLAB 4.2c 在 Macintosh 机上同样可以缓存当前的目录，因为这是第一个所搜索的磁盘位置。这个 MATLAB 版本也允许有选择地将整个 MATLAB 搜索路径缓存，并把高速缓存信息存储在一个文件中。这样，使 MATLAB 引导更快，寻找和编译所有函数 M 文件更快。退出缓存，不检测已修改的或已增加的 M 文件。当新的 M 文件加到一个缓存区时，只有当高速缓存由命令**» path** 刷新时，MATLAB 才能找到它们；另一方面，当修改缓存的 M 文件时，只有当以前编译过的版本由 **clear** 命令从内存中清除，MATLAB 才识别这个变化。例如，**» clear myfun**，从内存中清除 M 文件函数 **myfun**，或**» clear functions**，从内存中清除所有已编译的函数。

24. 在变量 **mfilename** 函数内，有要执行的 M 文件的名字。例如，正在执行 M 文件 **function.m** 时，函数的工作空间包含变量 **mfilename**，它包含函数字符串。这个变量也存在于脚本文件里，在这种情况下，它包含了要执行的脚本文件的名字。

25. M 文件函数可象 MATLAB 命令一样工作，典型的 MATLAB 命令包括 **clear**, **disp**, **echo**, **diary**, **save**, **hold**, **load**, **more**,和 **format**。通常，调用一个函数把参量放在括号内。例如 **size(A)**。然而，如果函数有字符串参量，那么，函数可按通常函数进行调用，如，**disp(' To be or not to be ')**，或象一个 MATLAB 命令来使用，如 **clear functions**。换句话说，当要求 MATLAB 解释一个表达式**» command argument** 时，MATLAB 认为它如同**» command(' argument ')**一样。事实上，MATLAB 命令本身能象函数那样调用！例如**» format long** 和**» format(' long ')**二者都把数据变成长格式。类似地，**» format short e** 等价于**» format(' short ' , ' e ')**。正如最后的例子所示，空格(逗号，分号)把各个命令参量分开。因此，**» disp How about this?** 产生一个错误，因为命令 **disp** 只允许一个输入参量，不是三个。如果参量包含在引号里，那么 MATLAB 就忽略空格；例如，**» disp ' How about this? '** 与**» disp(' How about this? ')**等价，并产生所希望的结果。

总之，函数 M 文件提供了一个简单的扩展 MATLAB 功能的方法。事实上，MATLAB 本身的许多标准函数就是 M 文件函数。

第 9 章　数 据 分 析

由于 MATLAB 面向矩阵，所以它很容易对数据集合进行统计分析。按规定，**数据集存储在面向列的矩阵里**。也就是，一个矩阵的每一列代表不同的被测变量，每一行代表各个样本或观察值。例如，让我们假定，一个月 31 天的三城市每日高温(单位为 ^{0}C)被记录，并赋给脚本 M 文件中的变量 **temps**，在精通 MATLAB 工具箱里取名为 **mmtemp.m**。运行 M 文件，把变量 **temps** 放在 MATLAB 工作空间里。这样，变量 **temps** 包含：

```
» temps
temps =
```

12	8	18
15	9	22
12	5	19
14	8	23
12	6	22
11	9	19
15	9	15
8	10	20
19	7	18
12	7	18
14	10	19
11	8	17
9	7	23
8	8	19
15	8	18
8	9	20
10	7	17
12	7	22
9	8	19
12	8	21
12	8	20
10	9	17
13	12	18
9	10	20
10	6	22
14	7	21
12	5	22
13	7	18
15	10	23
13	11	24
12	12	22

每一行包含了给定一天的高温；每一列包含不同城市的高温。为了使数据可视，把它绘图：

```
» d=1:31;          %    number the days of the month

» plot(d, temps)

» xlabel(' Day of Month '),ylabel(' Celsius ')

» title(' Daily High Temperatures in Three Cities ')
```

(见图 9.1)

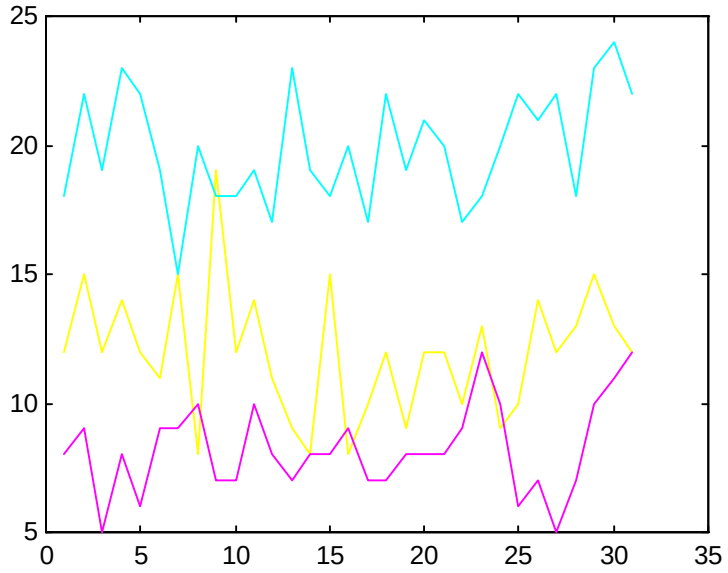


图 9.1　三个城市的每日高温

上面的 **plot** 命令也说明了 **plot** 命令用法的另一种形式。变量 **d** 是一个长度为 31 的向量，而 **temps** 是一个 31×3 矩阵。给定这些数据，**plot** 命令绘出了 **temps** 对每一列 **d** 的曲线。绘图在第 7 和 8 章进一步讨论。

为了说明 MATLAB 数据分析的一些功能，根据上面温度数据考虑以下命令。

```
» avg_temp=mean(temps)
avg_temp =
    11.9677      8.2258     19.8710
```

表明第三个城市有最高平均温度。这里 MATLAB 分别地找出了各列的平均值。

```
» avg_avg=mean(avg_temp)
avg_avg =
    13.3548
```

找出了三个城市的总平均温度。当输入到数据分析函数是行或列向量时，MATLAB 仅对向量执行运算，返回一个标量。

考虑从各城市的均值求每日偏差的问题。即必须从 **temps** 的 **i** 列中减去 **avg_temp(i)**。我们不能仅仅用以下的语句

```
» temps-avg_temp
??? Error using ==> -
Matrix dimensions must agree.
```

因为这个操作不是一个已定义的数组操作(**temps** 是 31×3 和 **avg_temp** 是 1×3)。或许最直接的方法是使用 For 循环。

```
for i=1:3
        tdev( : , i)=temps( : , i)-avg_temp(i) ;
end
```

```
» tdev
tdev =
    0.0323   -0.2258   -1.8710
    3.0323    0.7742    2.1290
    0.0323   -3.2258   -0.8710
    2.0323   -0.2258    3.1290
    0.0323   -2.2258    2.1290
   -0.9677    0.7742   -0.8710
    3.0323    0.7742   -4.8710
   -3.9677    1.7742    0.1290
    7.0323   -1.2258   -1.8710
    0.0323   -1.2258   -1.8710
    2.0323    1.7742   -0.8710
   -0.9677   -0.2258   -2.8710
   -2.9677   -1.2258    3.1290
   -3.9677   -0.2258   -0.8710
    3.0323   -0.2258   -1.8710
   -3.9677    0.7742    0.1290
   -1.9677   -1.2258   -2.8710
    0.0323   -1.2258    2.1290
   -2.9677   -0.2258   -0.8710
    0.0323   -0.2258    1.1290
    0.0323   -0.2258    0.1290
   -1.9677    0.7742   -2.8710
    1.0323    3.7742   -1.8710
   -2.9677    1.7742    0.1290
   -1.9677   -2.2258    2.1290
    2.0323   -1.2258    1.1290
    0.0323   -3.2258    2.1290
    1.0323   -1.2258   -1.8710
    3.0323    1.7742    3.1290
    1.0323    2.7742    4.1290
    0.0323    3.7742    2.1290
```

虽然使用上面的方法有效，但比使用 MATLAB 的数组操作功能要慢。复制 **avg_temp**，使得它与 **temps** 有同样的大小，然后再做减法，这样就快得多。

```
» tdev=temps-avg_temp(ones(31,1),:)
tdev =
```

```
    0.0323   -0.2258   -1.8710
    3.0323    0.7742    2.1290
    0.0323   -3.2258   -0.8710
    2.0323   -0.2258    3.1290
    0.0323   -2.2258    2.1290
   -0.9677    0.7742   -0.8710
    3.0323    0.7742   -4.8710
   -3.9677    1.7742    0.1290
    7.0323   -1.2258   -1.8710
    0.0323   -1.2258   -1.8710
    2.0323    1.7742   -0.8710
   -0.9677   -0.2258   -2.8710
   -2.9677   -1.2258    3.1290
   -3.9677   -0.2258   -0.8710
    3.0323   -0.2258   -1.8710
   -3.9677    0.7742    0.1290
   -1.9677   -1.2258   -2.8710
    0.0323   -1.2258    2.1290
   -2.9677   -0.2258   -0.8710
    0.0323   -0.2258    1.1290
    0.0323   -0.2258    0.1290
   -1.9677    0.7742   -2.8710
    1.0323    3.7742   -1.8710
   -2.9677    1.7742    0.1290
   -1.9677   -2.2258    2.1290
    2.0323   -1.2258    1.1290
    0.0323   -3.2258    2.1290
    1.0323   -1.2258   -1.8710
    3.0323    1.7742    3.1290
    1.0323    2.7742    4.1290
    0.0323    3.7742    2.1290
```

这里 **avg_temp(ones(31, 1),:)**复制 **avg_temp** 的第一行(且仅)31 次，创建了一个 31×3 的矩阵，其第 **i** 列是 **avg_temp(i)**。

```
» max_temp=max(temps)
max_temp =
    19    12    24
```

找出了每个城市一个月的最高温度。

```
» [max_temp, x]=max(temps)
max_temp =
    19    12    24
```

```
x =
     9    23    30
```

找出了每个城市的最高温度和出现最高温度的行下标 **x**。对于这个例子，当发生最高温度时，**x** 辨认了月中的日期。

```
» min_temp=min(temps)
min_temp =
     8     5    15
```

找出了各城市一个月的最低温度。

```
» [min_temp, n]=min(temps)
min_temp =
     8     5    15
n =
     8     3     7
```

找出了每个城市的最低温度和出现最低温度时行下标 **n**。对于这个例子，当发生最低温度时，**n** 辨认月中的日期。

```
» s_dev=std(temps)
s_dev =
   2.5098    1.7646    2.2322
```

找出 **temps** 的标准偏差。

```
» daily_change=diff(temps)
daily_change =
     3     1     4
    -3    -4    -3
     2     3     4
    -2    -2    -1
    -1     3    -3
     4     0    -4
    -7     1     5
    11    -3    -2
    -7     0     0
     2     3     1
    -3    -2    -2
    -2    -1     6
    -1     1    -4
     7     0    -1
    -7     1     2
```

```
     2     -2     -3
     2      0      5
    -3      1     -3
     3      0      2
     0      0     -1
    -2      1     -3
     3      3      1
    -4     -2      2
     1     -4      2
     4      1     -1
    -2     -2      1
     1      2     -4
     2      3      5
    -2      1      1
    -1      1     -2
```

计算每日高温之间的偏差，它描述了逐天日高温的变化有多大。例如，**daily_change** 的第一行是每月的第一天和第二天之间的日温度变化量。

9.1 数据分析函数

在 MATLAB 里的数据分析是按面向列矩阵而进行的。不同的变量存储在各列中，而每行表示每个变量的不同观察。MATLAB 统计函数包括

表 9.1

数 据 分 析 函 数	
corrcoef(x)	求相关系数
cov(x)	协方差矩阵
cplxpair(x)	把向量分类为复共轭对
cross(x, y)	向量的向量积
cumprod(x)	列累计积
cumsum(x)	列累计和
del2(A)	五点离散拉氏算子
diff(x)	计算元素之间差
dot(x, y)	向量的点积
gradient(Z, dx, dy)	近似梯度
histogram(x)	直方图和棒图
max(x), max(x, y)	最大分量
mean(x)	均值或列的平均值
median(x)	列的中值
min(x), min(x, y)	最小分量
prod(x)	列元素的积
rand(x)	均匀分布随机数

randn(x)	正态分布随机数
sort(x)	按升序排列
std(x)	列的标准偏差
subspace(A, B)	两个子空间之间的夹角
sum(x)	各列的元素和

9.2 M 文件举例

在这一章里，说明在**精通 MATLAB 工具箱**里的两个函数。这些函数说明了本章所示的 **min** 和 **max** 函数的变种和如何编写一个 M 文件。关于 M 文件的更多信息，参阅第 8 章。

在讨论 M 文件函数 **mmin** 和 **mmax** 的内部结构之前，考虑他们有什么功能。

```
» amn_temp=mmin(temps)
amn_temp =
     5

» [m , i]=mmin(temps)
m =
     5
i =
     3     2

» amx_temp=mmax(temps)
amx_temp =
    24

» [m , j]=mmax(temps)
m =
    24
j =
    30     3
```

具有一个输出参量的函数 **mmin** 找出矩阵中的单个最小值。用第二个输出参量，返回单个最小值的行和列的下标。除了 **mmax** 返回矩阵中的单个最大值外，函数 **mmax** 的工作方式与 **mmin** 相同。这些 M 文件的函数是：

```
function [m , i]=mmin(a)
%   MMIN Matrix minimum value.
%   MMIN(A) returns the minimum value in the matrix A
%   [M,I] = MMIN(A) in addition returns the indices of
%   the minimum value in I = [row col].

%   Copyright (c) 1996 by Prentice Hall,Inc.
```

```
if nargout==2, % return indices
      [m , i]=min(a) ;
            [m , ic]=min(m) ;
      i=[i(ic)    ic] ;
      else,
      m=min(min(a));
end


function [m , i]=mmax(a)
%    MMAX Matrix maximum value.
%    MMAX(A) returns the maximum value in the matrix A
%    [M,I] = MMAX(A) in addition returns the indices of
%    the maximum value in I = [row col].

%    Copyright (c) 1996 by Prentice Hall,Inc.

if nargout==2,              %    return indices
      [m , i]=max(a) ;
            [m , ic]=max(m) ;
      i=[i(ic)    ic] ;
      else,
      m=max(max(a)) ;
end
```

第 10 章　多 项 式

10.1　根

找出多项式的根，即多项式为零的值，可能是许多学科共同的问题，。MATLAB 求解这个问题，并提供其它的多项式操作工具。**在 MATLAB 里，多项式由一个行向量表示，它的系数是按降序排列**。例如，输入多项式 $x^4-12x^3+0x^2+25x+116$

```
» p=[1    -12    0    25    116]
p =
      1     -12       0      25     116
```

注意，必须包括具有零系数的项。除非特别地辨认，MATLAB 无法知道哪一项为零。给出这种形式，用函数 **roots** 找出一个多项式的根。

```
» r=roots(p)
r =
   11.7473
    2.7028
   -1.2251 + 1.4672i
   -1.2251 - 1.4672i
```

因为在 MATLAB 中，无论是一个多项式，还是它的根，都是向量，**MATLAB 按惯例规定，多项式是行向量，根是列向量**。给出一个多项式的根，也可以构造相应的多项式。在 MATLAB 中，命令 **poly** 执行这个任务。

```
» pp=poly(r)
pp =
   1.0e+002 *
   Columns 1 through 4
    0.0100                  -0.1200                  0.0000                  0.2500
   Column 5
    1.1600 + 0.0000i

» pp=real(pp) %throw away spurious imaginary part
pp =
      1.0000   -12.0000        0.0000       25.0000   116.0000
```

因为 MATLAB 无隙地处理复数，当用根重组多项式时，如果一些根有虚部，由于截断误差，则 poly 的结果有一些小的虚部，这是很普通的。消除虚假的虚部，如上所示，只要使用函数 real 抽取实部。

10.2 乘法

函数 **conv** 支持多项式乘法(执行两个数组的卷积)。考虑两个多项式 $a(x)=x^3+2x^2+3x+4$ 和 $b(x)= x^3+4x^2+9x+16$ 的乘积：

```
» a=[1   2   3   4] ;   b=[1   4   9   16];
» c=conv(a , b)
c =
       1       6      20      50      75      84      64
```

结果是 $c(x)=x^6+6x^5+20x^4+50x^3+75x^2+84x+64$。两个以上的多项式的乘法需要重复使用 **conv**。

10.3 加法

对多项式加法，MATLAB 不提供一个直接的函数。如果两个多项式向量大小相同，标准的数组加法有效。把多项式 a(x)与上面给出的 b(x)相加。

```
» d=a+b
d =
     2     6    12    20
```

结果是 $d(x)= 2x^3+6x^2+12x+20$。当两个多项式阶次不同，低阶的多项式必须用首零填补，使其与高阶多项式有同样的阶次。考虑上面多项式 c 和 d 相加：

```
» e=c+[0  0  0  d]
e =
     1     6    20    52    81    96    84
```

结果是 $e(x)= x^6+6x^5+20x^4+52x^3+81x^2+96x+84$。要求首零而不是尾零，是因为相关的系数象 x 幂次一样，必须整齐。

如果需要，可用一个文件编辑器创建一个函数 M 文件来执行一般的多项式加法。**精通 MATLAB 工具箱**包含下列实现：

```
function p=mmpadd(a,b)
%   MMPADD Polynomial addition.
%   MMPADD(A,B) adds the polynomial A and B

%   Copyright (c) 1996 by Prentice Hall,Inc.

if nargin<2
      error(' Not enough input arguments ')
end
a=a(:).' ;              %   make sure inputs are polynomial row vectors
b=b(:).' ;
na=length(a) ;          %   find lengths of a and b
nb=length(b) ;
p=[zeros(1,nb-na) a]+[zeros(1,na-nb) b] ;      %   add zeros as necessary
```

现在，为了阐述 **mmpadd** 的使用，再考虑前一页的例子。

```
» f=mmpadd(c,d)
f =
     1     6    20    52    81    96    84
```

它与上面的 **e** 相同。当然，**mmpadd** 也用于减法。

```
»g=mmpadd(c , -d)
g =
      1       6      20      48      69      72      44
```

结果是 $g(x)= x^6+6x^5+20x^4+48x^3+69x^2+72x+44$。

10.4 除法

在一些特殊情况，一个多项式需要除以另一个多项式。在 MATLAB 中，这由函数 **deconv** 完成。用上面的多项式 **b** 和 **c**

```
» [q , r]=deconv(c , b)
q =
      1       2       3       4
r =
      0       0       0       0       0       0       0
```

这个结果是 **b** 被 **c** 除，给出商多项式 **q** 和余数 **r**，在现在情况下 r 是零，因为 **b** 和 **q** 的乘积恰好是 **c**。

10.5 导数

由于一个多项式的导数表示简单，MATLAB 为多项式求导提供了函数 **polyder**。

```
» g
g =
      1       6      20      48      69      72      44
» h=polyder(g)
h =
      6      30      80     144     138      72
```

10.6 估值

根据多项式系数的行向量，可对多项式进行加，减，乘，除和求导，也应该能对它们进行估值。在 MATLAB 中，这由函数 **polyval** 来完成。

```
» x=linspace(-1, 3) ;          %   choose 100 data points between -1and 3.
» p=[1   4   -7   -10] ;       %   uses polynomial p(x) = x^3+4x^2—7x—10
» v=polyval(p , x) ;
```

计算 **x** 值上的 **p(x)**，把结果存在 **v** 里。然后用函数 plot 绘出结果。

```
» plot(x , v),title(' x^3+4x^2-7x-10 '),    xlabel(' x ')
```

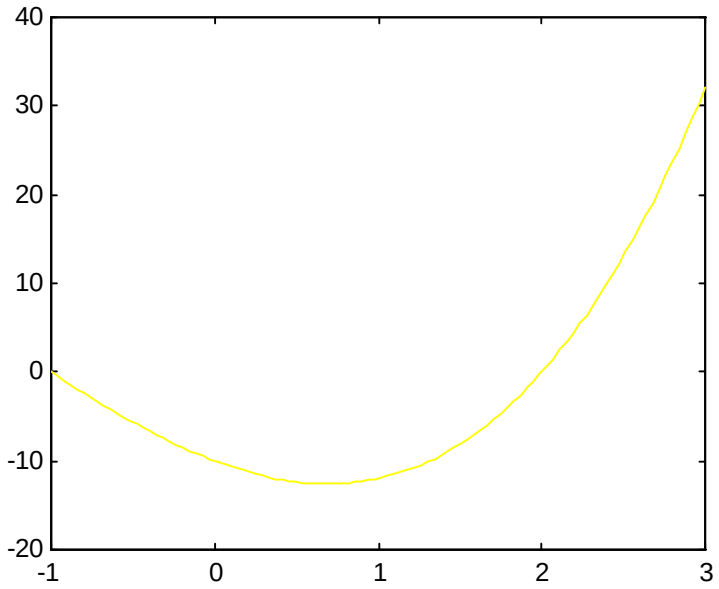


图 10.1　多项式估值

10. 7　有理多项式

在许多应用中，例如富里哀(Fourier)，拉普拉斯(Laplace)和 Z 变换，出现有理多项式或两个多项式之比。在 MATLAB 中，有理多项式由它们的分子多项式和分母多项式表示。对有理多项式进行运算的两个函数是 **residue** 和 **polyder**。函数 **residue** 执行部分分式展开。

```
» num=10*[1    2] ;                    %    numerator polynomial
» den=poly([-1;    -3;    -4]) ;      %    denominator polynomial
» [res, poles, k]=residue(num, den)
res =
    -6.6667
     5.0000
     1.6667
poles =
    -4.0000
    -3.0000
    -1.0000
k =
      [    ]
```

结果是余数、极点和部分分式展开的常数项。上面的结果说明了该问题：

$$\frac{10(s+2)}{(s+1)(s+3)(s+4)} = \frac{-6.6667}{s+4} + \frac{5}{s+3} + \frac{1.6667}{s+1} + 0$$

这个函数也执行逆运算。

```
» [n, d]=residue(res, poles, k)
n =
     0.0000     10.0000     20.0000
d =
     1.0000      8.0000     19.0000     12.0000

» roots(d)
ans =
    -4.0000
    -3.0000
    -1.0000
```

在截断误差内，这与我们开始时的分子和分母多项式一致。**residue** 也能处理重极点的情况，尽管这里没有考虑。

正如前面所述，函数 **polyder**，对多项式求导。除此之外，如果给出两个输入，则它对有理多项式求导。

```
» [b , a]=polyder(num , den)
b =
    -20   -140   -320   -260
a =
      1     16    102    328    553    456    144
```

该结果证实：

$$\frac{d}{ds}\left\{\frac{10(s+2)}{(s+1)(s+3)(s+4)}\right\} = \frac{-20s^3 - 140s^2 - 32s - 260}{s^6 + 16s^5 + 102s^4 + 328s^3 + 553s^2 + 456s + 144}$$

10.8 M 文件举例

本章说明了在**精通 MATLAB 工具箱** 中两个函数。这些函数说明了本章所论述的多项式概念和如何写 M 文件函数。关于 M 文件的更多信息，参阅第 8 章。

在讨论 M 文件函数的内部结构之前，我们考虑这些函数做些什么。

```
» n          %    earlier data
```

```
n =
       0.0000      10.0000      20.0000

» b           %    earlier data
b =
     -20    -140    -320    -260

» mmpsim(n)          %    strip away negligible leading term
ans =
     10.0000      20.0000

» mmp2str(b)         %    convert polynomial to string
ans =
-20s^3 - 140s^2 - 320s^1 - 260

» mmp2str(b , ' x ')
ans =
-20x^3 - 140x^2 - 320x^1 - 260

» mmp2str(b , [    ] , 1)
ans =
-20*(s^3 + 7s^2 + 16s^1 + 13)

» mmp2str(b , ' x ' , 1)
ans =
-20*(x^3 + 7x^2 + 16x^1 + 13)
```

这里函数 **mmpsim** 删除在多项式 **n** 中近似为零的第一个系数。函数 mmp2str 把数值多项式变换成等价形式的字符串表达式。该两个函数的主体是：

```
function y=mmpsim(x,tol)
%    MMPSIM Polynomial Simplification,Strip Leading Zero Terms.
%    MMPSIM(A) Delets leading zeros and small coefficients in the
%    polynomial A(s). Coefficients are considered small if their
%    magnitude is less than both one and norm(A)*1000*eps.
%    MMPSIM(A,TOL) uses TOL for its smallness tolerance.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

if nargin<2, tol=norm(x)*1000*eps; end
x=x(:).' ;                              %    make sure input is a row
i=find(abs(x)<.99&abs(x)<tol) ;       %    find insignificant indices
x(i)=zeros(1, length(i)) ;              %    set them to zero
i=find(x~=0) ;                          %    find significant indices
```

```
if isempty(i)
        y=0 ;                                %    extreme case: nothing left!
else
       y=x(i(1) : length(x)) ;               %    start with first term
end                                          %    and go to end of polynomial


function s=mmp2str(p,v,ff)
%    MMP2STR Polynomial Vector to String Conversion.
%    MMP2STR(P) converts the polynomial vector P into a string.
%    For example: P = [2    3    4] becomes the string ' 2s^2 + 3s + 4 '
%
%    MMP2STR(P,V) generates the string using the variable V
%    instead of s. MMP2STR([2    3    4],' z ') becomes ' 2z^2 + 3z + 4 '
%
%    MMP2STR(P,V,1) factors the polynomial into the product of a
%    constant and a monic polynomial.
%    MMP2STR([2    3    4],[    ],1) becomes ' 2(s^2 + 1.5s + 2) '

%    Copyright (c) 1996 by Prentice-Hall,Inc.

if nargin<3, ff=0; end                       %    factored form is False
if nargin <2, v=' s ' ; end                  %    default variable is ' s '
if isempty(v), v=' s ' ; end                 %    default variable is ' s '
v=v(1) ;                                     %    variable must be scalar
p=mmpsim(p) ;                                %    strip insignificant terms
n=length(p) ;
if ff                                        %    put in factored form
       K=p(1) ; Ka=abs(K) ; p=p/K;
       if abs(K-1)<1e-4
            pp=[    ]; pe=[    ] ;
       elseif abs(K+1)<1e-4
            pp=' -(' ; pe= ') ' ;
       elseif abs(Ka-round(Ka))<=1e-5*Ka
            pp=[sprintf(' %.0f ', K) '*( ' ] ; pe= ') ' ;
       else
            pp=[sprintf(' %.4g ' , K) '*(' ] ; pe= ') ' ;
       end
else                                         %    not factored form
       K=p(1);
       pp=sprintf(' %.4g ' , K) ;
       pe=[    ];
end
if n==1                                      %    polynomial is just a constant
```

```
        s=sprintf(' %.4g ',K);
        return
end
s=[pp v ' ^ ' sprintf(' %.0f ',n-1)];        %    begin string construction

for i=2:n-1                                  %    catenate center terms in polynomial
    if p(i)<0, pm= ' -    ' ;    else,    if p(i)<0,pm= ' ;    end
    if p(i)= =1,pp=[    ] ; else,    pp=sprintf(' %.4g ', abs(p(i))) ;    end
    if p(i)~ =0,s=[s    pm    pp    v    ' ^ ' sprintf(' %.0f ',n-i)] ;    end
end

if p(n)~ =0,pp=sprintf(' %.4g ',abs(p(n))); else,    pp=[    ];    end
if p(n)<0 , pm= ' -    ' ; elseif    p(n)>0 , pm= ' +    ' ; else,    pm=[    ];    end
s=[s    pm    pp    pe];                     %    add final terms
```

10.9 小结

下列表格概括了在本章所讨论的多项式操作特性。

表 10.1

多　项　式　函　数	
conv(a, b)	乘法
[q, r]=deconv(a, b)	除法
poly(r)	用根构造多项式
polyder(a)	对多项式或有理多项式求导
polyfit(x, y, n)	多项式数据拟合
polyval(p, x)	计算 x 点中多项式值
[r, p, k]=residue(a, b)	部分分式展开式
[a, b]=residue(r, p, k)	部分分式组合
roots(a)	求多项式的根

表 10.2

精 通 **MATLAB** 多 项 式 操 作	
mmp2str(a)	多项式向量到字符串变换，**a(s)**
mmp2str(a, ' x ')	多项式向量到字符串变换，**a(x)**
mmp2str(a, ' x ', 1)	常数和符号多项式变换
mmpadd(a, b)	多项式加法
mmpsim(a)	多项式简化

第 11 章　曲线拟合与插值

在大量的应用领域中，人们经常面临用一个解析函数描述数据(通常是测量值)的任务。对这个问题有两种方法。在插值法里，数据假定是正确的，要求以某种方法描述数据点之间所发生的情况。这种方法在下一节讨论。这里讨论的方法是曲线拟合或回归。人们设法找出某条光滑曲线，它最佳地拟合数据，但不必要经过任何数据点。图 11.1 说明了这两种方法。标有**'o'**的是数据点；连接数据点的实线描绘了线性内插，虚线是数据的最佳拟合。

11.1　曲线拟合

曲线拟合涉及回答两个基本问题：**最佳拟合**意味着什么？应该用什么样的曲线？可用许多不同的方法定义**最佳拟合**，并存在无穷数目的曲线。所以，从这里开始，我们走向何方？正如它证实的那样，当**最佳拟合**被解释为在数据点的最小误差平方和，且所用的曲线限定为多项式时，那么曲线拟合是相当简捷的。数学上，称为多项式的最小二乘曲线拟合。如果这种描述使你混淆，再研究图 11.1。虚线和标志的数据点之间的垂直距离是在该点的误差。对各数据点距离求平方，并把平方距离全加起来，就是误差平方和。这条虚线是使误差平方和尽可能小的曲线，即是**最佳拟合**。最小二乘这个术语仅仅是使误差平方和最小的省略说法。

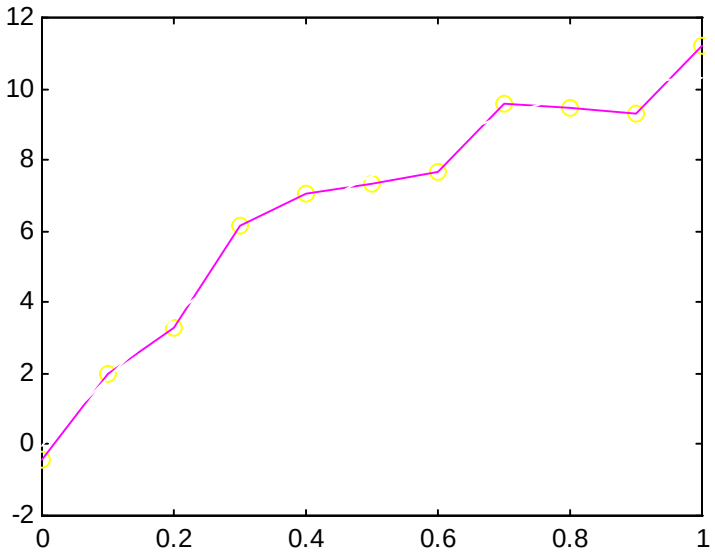


图 11.1　2 阶曲线拟合

在 MATLAB 中，函数 **polyfit** 求解最小二乘曲线拟合问题。为了阐述这个函数的用法，让我们以上面图 11.1 中的数据开始。

```
» x=[0   .1   .2   .3   .4   .5   .6   .7    .8   .9   1];
```

```
» y=[-.447    1.978    3.28    6.16    7.08    7.34    7.66    9.56    9.48    9.30    11.2];
```

为了用 **polyfit**，我们必须给函数赋予上面的数据和我们希望最佳拟合数据的多项式的阶次或度。如果我们选择 **n=1** 作为阶次，得到最简单的线性近似。通常称为线性回归。相反，如果我们选择 **n=2** 作为阶次，得到一个 2 阶多项式。现在，我们选择一个 2 阶多项式。

```
» n=2;                    %    polynomial order

» p=polyfit(x, y, n)
p =
    -9.8108      20.1293      -0.0317
```

polyfit 的输出是一个多项式系数的行向量。其解是 y = －9.8108x^2 ＋20.1293x－0.0317。为了将曲线拟合解与数据点比较，让我们把二者都绘成图。

```
» xi=linspace(0, 1, 100);        %    x-axis data for plotting
» z=polyval(p, xi);
```

为了计算在 **xi** 数据点的多项式值，调用 MATLAB 的函数 **polyval**。

```
» plot(x, y, ' o ' , x, y, xi, z, ' : ' )
```

画出了原始数据 **x** 和 **y**，用**'o'**标出该数据点，在数据点之间，再用直线重画原始数据，并用点**' : '**线，画出多项式数据 **xi** 和 **z**。

```
» xlabel(' x '), ylabel(' y=f(x) '), title(' Second Order Curve Fitting ')
```

将图作标志。这些步骤的结果表示于前面的图 11.1 中。

多项式阶次的选择是有点任意的。两点决定一直线或一阶多项式。三点决定一个平方或 2 阶多项式。按此进行，**n+1** 数据点唯一地确定 **n** 阶多项式。于是，在上面的情况下，有 11 个数据点，我们可选一个高达 10 阶的多项式。然而，高阶多项式给出很差的数值特性，人们不应选择比所需的阶次高的多项式。此外，随着多项式阶次的提高，近似变得不够光滑，因为较高阶次多项式在变零前，可多次求导。例如，选一个 10 阶多项式

```
» pp=polyfit(x, y, 10) ;
» format short e              %    change display format

» pp.'                        %    display polynomial coefficients as a column
ans =
  -4.6436e+005
    2.2965e+006
  -4.8773e+006
    5.8233e+006
  -4.2948e+006
```

```
  2.0211e+006
 -6.0322e+005
  1.0896e+005
 -1.0626e+004
  4.3599e+002
 -4.4700e-001
```

要注意在现在情况下，多项式系数的规模与前面的 2 阶拟合的比较。还要注意在最小(-4.4700e-001)和最大(5.8233e+006)系数之间有 7 个数量级的幅度差。将这个解作图，并把此图与原始数据及 2 阶曲线拟合相比较，结果如何呢？

```
» zz=polyval(pp, xi);                    %   evaluate 10th order polynomial
» plot(x, y, ' o ' , xi, z, ' : ' , xi, zz)      %   plot data
» xlabel(' x '),   ylabel(' y=f(x) '),   title(' 2nd and 10th Order curve Fitting ')
```

在下面的图 11.2 中，原始数据标以**'o'**，2 阶曲线拟合是虚线，10 阶拟合是实线。注意，在 10 阶拟合中，在左边和右边的极值处，数据点之间出现大的纹波。当企图进行高阶曲线拟合时，这种纹波现象经常发生。根据图 11.2，显然，‘ 越多就越好 ’ 的观念在这里不适用。

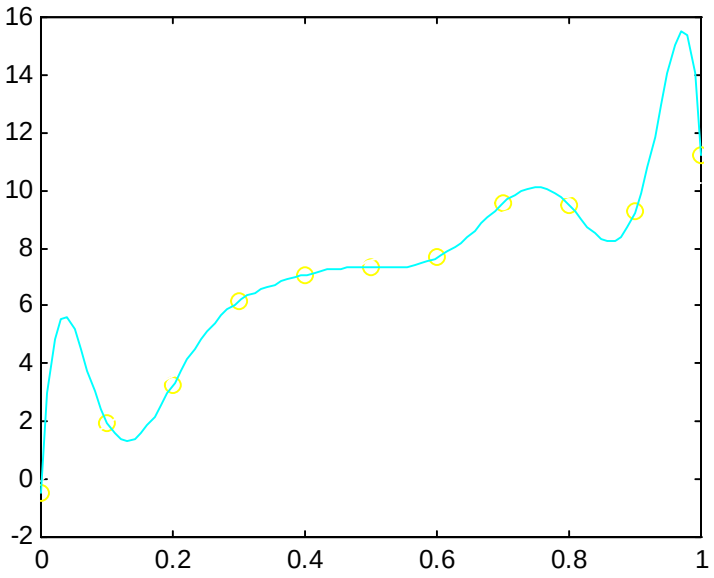


图 11.2　2 阶和 10 阶曲线拟合

11.2　一维插值

正如在前一节对曲线拟合所描述的那样，插值定义为对数据点之间函数的估值方法，这些数据点是由某些集合给定。当人们不能很快地求出所需中间点的函数值时，插值是一个有

价值的工具。例如，当数据点是某些实验测量的结果或是过长的计算过程时，就有这种情况。

或许最简单插值的例子是 MATLAB 的作图。按缺省，MATLAB 用直线连接所用的数据点以作图。**这个线性插值猜测中间值落在数据点之间的直线上**。当然，当数据点个数的增加和它们之间距离的减小时，线性插值就更精确。例如，

```
» x1=linspace(0, 2*pi, 60);

» x2=linspace(0, 2*pi, 6);

» plot(x1, sin(x1), x2, sin(x2), ' - ')

» xlabel(' x '),    ylabel(' sin(x) '),    title(' Linear Interpolation ')
```

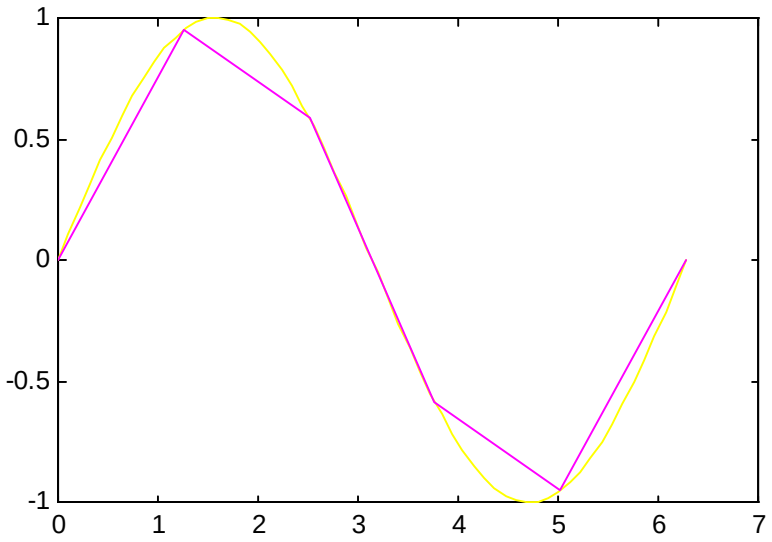


图 11.3　线性插值

图 11.3 是 sine 函数的两个图，一个在数据点之间用 60 个点，它比另一个只用 6 个点更光滑和更精确。

如曲线拟合一样，插值要作决策。根据所作的假设，有多种插值。而且，可以在一维以上空间中进行插值。即如果有反映两个变量函数的插值，**z=f(x, y)**，那么就可在 x 之间和在 y 之间，找出 z 的中间值进行插值。MATLAB 在一维函数 **interp1** 和在二维函数 **interp2** 中，提供了许多的插值选择。其中的每个函数将在下面阐述。

为了说明一维插值，考虑下列问题，12 小时内，一小时测量一次室外温度。数据存储在两个 MATLAB 变量中。

```
» hours=1:12;          %    index for hour data was recorded

» temps=[5    8    9    15    25    29    31    30    22    25    27    24]; %    recorded temperatures
```

```
» plot(hours, temps, hours, temps,' + ')        %    view temperatures

» title(' Temperature ')

» xlabel(' Hour '),    ylabel(' Degrees Celsius ')
```

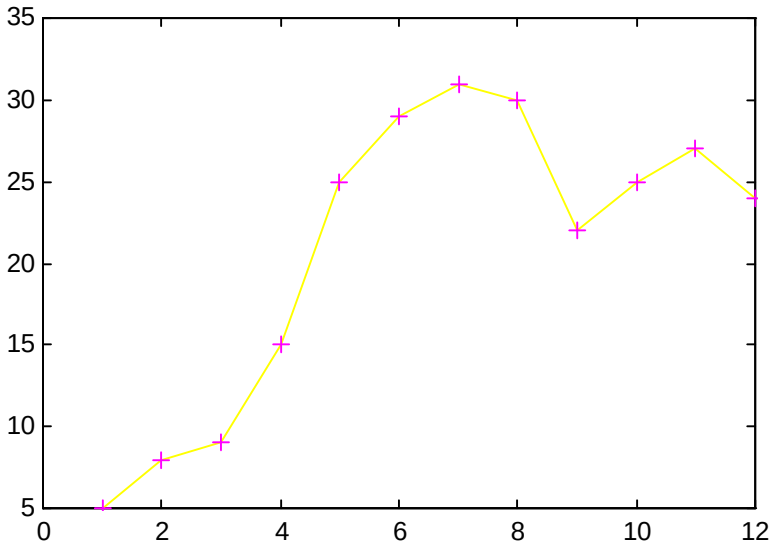


图 11.4　在线性插值下室外温度曲线

正如图 11.4 看到的，MATLAB 画出了数据点线性插值的直线。为了计算在任意给定时间的温度，人们可试着对可视的图作解释。另外一种方法，可用函数 **interp1**。

```
» t=interp1(hours, temps, 9.3)              %    estimate temperature at hour=9.3
t =
    22.9000

» t=interp1(hours, temps, 4.7)              %    estimate temperature at hour=4.7
t =
     22

» t=interp1(hours, temps, [3.2    6.5    7.1    11.7])        %    find temp at many points!
t =
    10.2000
    30.0000
    30.9000
    24.9000
```

interp1 的缺省用法是由 **interp1(x, y, xo)**来描述，这里 **x** 是独立变量(横坐标)，**y** 是应变

量(纵坐标)，**xo** 是进行插值的一个数值数组。另外，该缺省的使用假定为线性插值。

若不采用直线连接数据点，我们可采用某些更光滑的曲线来拟合数据点。最常用的方法是用一个 3 阶多项式，即 3 次多项式，来对相继数据点之间的各段建模，每个 3 次多项式的头两个导数与该数据点相一致。这种类型的插值被称为 **3 次样条**或简称为**样条**。函数 **interp1** 也能执行 3 次样条插值。

```
» t=interp1(hours, temps, 9.3, ' spline ')        %    estimate temperature at hour=9.3
t =
    21.8577

» t=interp1(hours, temps, 4.7, ' spline ')        %    estimate temperature at hour=4.7
t =
    22.3143

» t=interp1(hours, temps, [3.2    6.5    7.1    11.7], ' spline ')
t =
     9.6734
    30.0427
    31.1755
    25.3820
```

注意，样条插值得到的结果，与上面所示的线性插值的结果不同。因为插值是一个估计或猜测的过程，其意义在于，应用不同的估计规则导致不同的结果。

一个最常用的样条插值是对数据平滑。也就是，给定一组数据，使用样条插值在更细的间隔求值。例如，

```
» h=1:0.1:12;              %    estimate temperature every 1/10 hour

» t=interp1(hours, temps, h, ' spline ') ;

» plot(hours, temps, ' - ' , hours, temps, ' + ' , h, t)        %    plot comparative results

» title(' Springfield Temperature ')

» xlabel(' Hour '),    ylabel(' Degrees Celsius ')
```

在图 11.5 中，虚线是线性插值，实线是平滑的样条插值，标有**' + '**的是原始数据。如要求在时间轴上有更细的分辨率，并使用样条插值，我们有一个更平滑、但不一定更精确地对温度的估计。尤其应注意，在数据点，样条解的斜率不突然改变。作为这个平滑插值的回报，3 次样条插值要求更大量的计算，因为必须找到 3 次多项式以描述给定数据之间的特征。关于样条的更详细信息可见下一章。

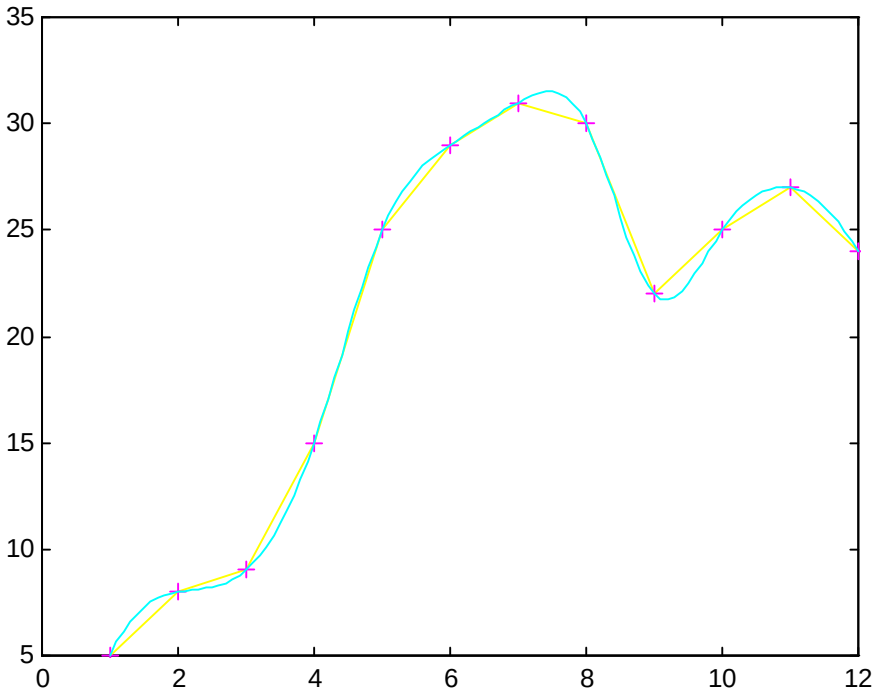


图 11.5　在不同插值下室外温度曲线

在讨论二维插值之前，了解 **interp1** 所强制的二个强约束是很重要的。首先，人们不能要求有独立变量范围以外的结果，例如，**interp1(hours, temps, 13.5)**导致一个错误，因为 **hours** 在 1 到 12 之间变化。其次，独立变量必须是单调的。即独立变量在值上必须总是增加的或总是减小的。在我们的例子里，**hours** 是单调的。然而，如果我们已经定义独立变量为一天的实际时间，

```
» time_of_day=[7:12     1:6]          %    start at 7AM,end at 6PM
time_of_day =
     7     8     9     10     11     12     1     2     3     4     5     6
```

则独立变量将不是单调的，因为 **time_of_day** 增加到 12，然后跌到 1，再然后增加。如果用 **time_of_day** 代替 **interp1** 中的 **hours**，将会返回一个错误。同样的理由，人们不能对 **temps** 插值来找出产生某温度的时间(小时)，因为 **temps** 不是单调的。

11.3　二维插值

二维插值是基于与一维插值同样的基本思想。然而，正如名字所隐含的，二维插值是对两变量的函数 **z=f(x, y)** 进行插值。为了说明这个附加的维数，考虑一个问题。设人们对平板上的温度分布估计感兴趣，给定的温度值取自平板表面均匀分布的格栅。

采集了下列的数据：

```
» width=1:5;                    %    index for width of plate (i.e.,the x-dimension)
```

```
» depth=1:3;                    %    index for depth of plate (i,e,,the y-dimension)

» temps=[82    81    80    82    84; 79    63    61    65    81; 84    84    82    85    86]    %
temperature data
temps =
     82     81     80     82     84
     79     63     61     65     81
     84     84     82     85     86
```

如同在标引点上测量一样，矩阵 **temps** 表示整个平板的温度分布。**temps** 的列与下标 **depth** 或 y-维相联系，行与下标 **width** 或 x-维相联系(见图 11.6)。为了估计在中间点的温度，我们必须对它们进行辨识。

```
» wi=1:0.2:5;          %    estimate across width of plate

» d=2;                 %    at a depth of 2

» zlinear=interp2(width, depth, temps, wi, d) ;                %    linear interpolation

» zcubic=interp2(width, depth, temps, wi,d, ' cubic ') ;       %    cubic interpolation

» plot(wi, zlinear, ' - ' , wi, zcubic)         %    plot results

» xlabel(' Width of Plate '),    ylabel(' Degrees Celsius ')

» title( [' Temperature at Depth =    '    num2str(d) ] )
```

另一种方法，我们可以在两个方向插值。先在三维坐标画出原始数据，看一下该数据的粗糙程度(见图 11.7)。

```
» mesh(width, depth, temps)                %    use mesh plot

» xlabel(' Width of Plate '),    ylabel(' Depth of Plate ')

» zlabel(' Degrees Celsius '),    axis(' ij '),    grid
```

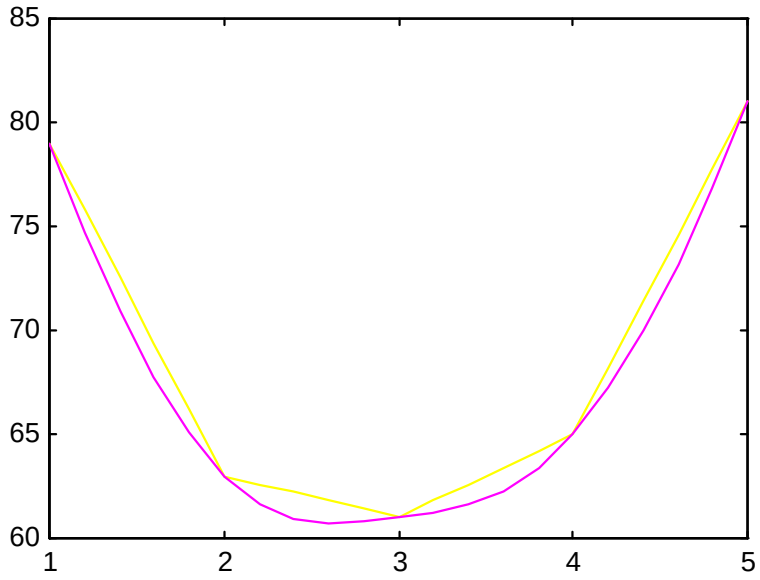


图 11.6　在深度 d=2 处的平板温度

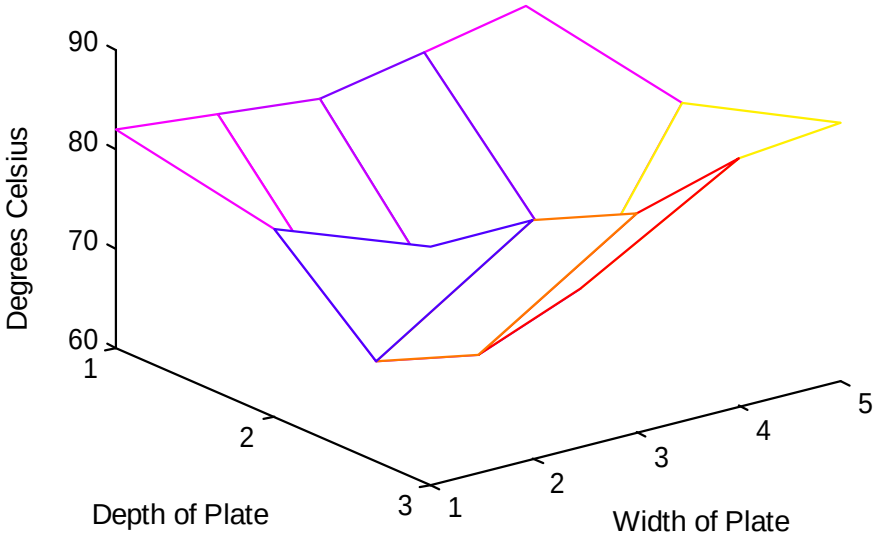


图 11.7　平板温度

然后在两个方向上插值，以平滑数据。

» di=1:0.2:3;　　%　choose higher resolution for depth

» wi=1:0.2:5;　　%　choose higher resolution for width

» zcubic=interp2(width, depth, temps, wi, di, ' cubic ') ;　　%　cubic

» mesh(wi, di, zcubic)

» xlabel(' Width of Plate '), ylabel(' Depth of Plate ')

» zlabel(' Degrees Celsius '), axis(' ij '), grid

上面的例子清楚地证明了，二维插值更为复杂，只是因为有更多的量要保持跟踪。**interp2** 的基本形式是 **interp2(x, y, z, xi, yi, method)**。这里 **x** 和 **y** 是两个独立变量，**z** 是一个应变量矩阵。**x** 和 **y** 对 **z** 的关系是

z(i, :) = f(x, y(i)) 和 z(:, j) = f(x(j), y).

也就是，当 **x** 变化时，**z** 的第 **i** 行与 **y** 的第 **i** 个元素 **y(i)**相关，当 **y** 变化时，**z** 的第 **j** 列与 **x** 的第 **j** 个元素 **x(j)**相关，。**xi** 是沿 x-轴插值的一个数值数组；**yi** 是沿 y-轴插值的一个数值数组。

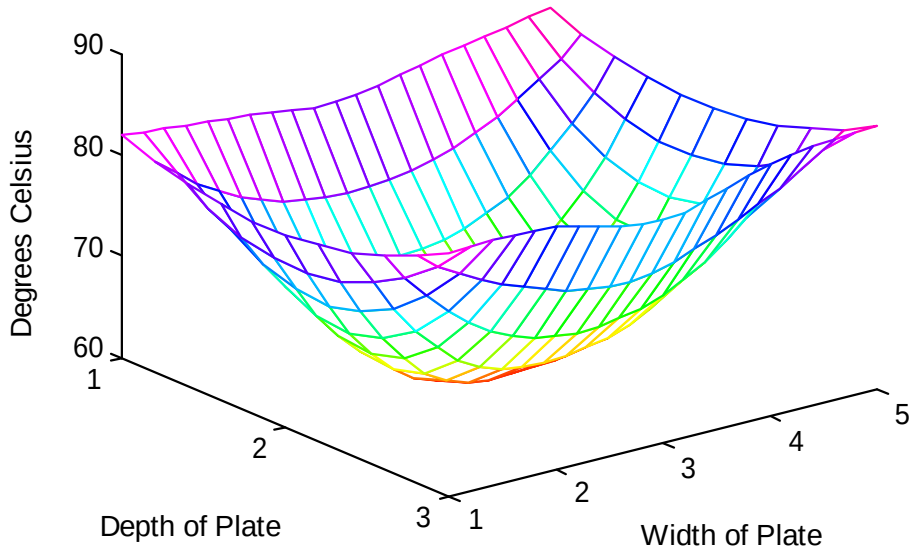


图 11.8　二维插值后的平板温度

可选的参数 **method** 可以是 **'linear'**，**'cubic'**或**'nearest'**。在这种情况下，**cubic** 不意味着 3 次样条，而是使用 3 次多项式的另一种算法。**linear** 方法是线性插值，仅用作连接图上数据点。**nearest** 方法只选择最接近各估计点的粗略数据点。在所有的情况下，假定独立变量 **x** 和 **y** 是线性间隔和单调的。关于这些方法的更多的信息，可请求在线帮助，例如，**» help interp2**，或参阅 MATLAB 参考手册。

11.4　M 文件举例

虽然对于许多应用，函数 **interp1** 和 **interp2** 是很有用的，但它们限制为对单调向量进行插值。在某些情况，这个限制太严格。例如，考虑下面的插值：

```
» x=linspace(0, 5);

» y=1-exp(-x).*sin(2*pi*x);
» plot(x, y)
```

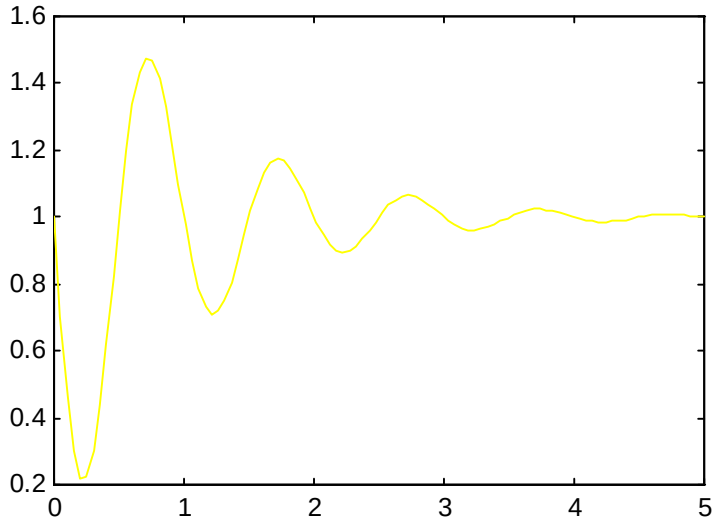


图 11.9　函数 1-exp(-x).*sin(2*pi*x)的曲线

函数 **interp1** 可用来在任何值或 **x** 的值上估计 **y** 值。

```
» yi=interp1(x, y, 1.8)
yi =
    1.1556
```

然而，**interp1** 不能找出对应于某些 **y** 值的 **x** 值。例如，如在图 11.9 上所示，考虑寻找 **y=1.1** 处的 **x** 值：

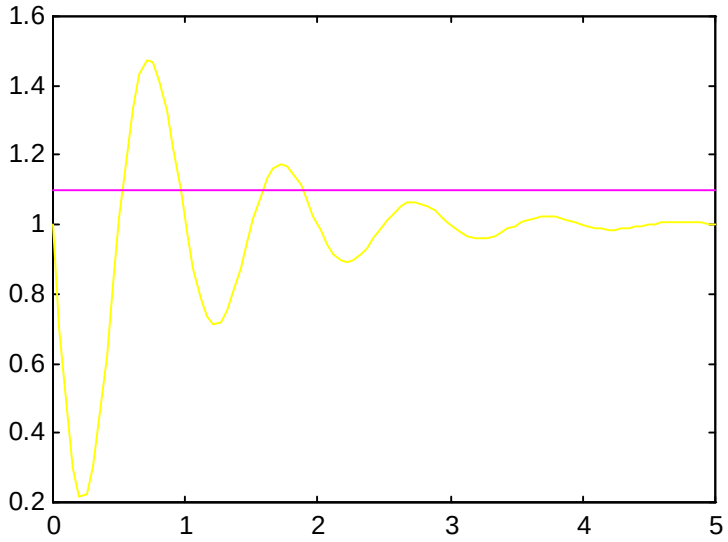


图 11.10　给 y 值在函数曲线上求 x 的值

```
» plot(x, y, [0, 5], [1.1       1.1] )
```

从图 11.10 上，我们看到有四个交点。使用 **interp1**，我们得到：

```
» xi=interp1(y, x, 1.1)
??? Error using ==> table1
First column of the table must be monotonic.
```

这个函数 **interp1** 失败，由于 **y** 不是单调的。
在本章**精通 MATLAB 工具箱**所说明的 M 文件例子，消除了单调性的要求。

```
» table=[x; y].' ;          %    create column oriented table from data

» xi=mminterp(table, 2, 1.1)
xi =
      0.5281      1.1000
      0.9580      1.1000
      1.5825      1.1000
      1.8847      1.1000
```

这里使用了线性插值，函数 **mminterp** 估计了 **y=1.1** 处的四个点。由于函数 **mminterp** 的一般性质，要插值的数据是由面向列矩阵给出，在上面的例子中称作为表(**table**)。第二个输入参量是被搜索矩阵 **table** 的列，第三个参量是要找的值。

这个**精通 MATLAB 工具箱**函数的主体由下面给出：

```
function y=mminterp(tab, col, val)
%    MMINTERP 1-D Table Search by Linear Interpolation.
```

```
%    Y=MMINTERP(TAB,COL,VAL) linearly interpolates the table
%    TAB searching for the scalar value VAL in the column COL.
%    All crossings are found and TAB(:,COL) need not be monotonic.
%    Each crossing is returned as a separate row in Y and Y has as
%    many columns as TAB.Naturally,the column COL of Y contains
%    the value VAL. If VAL is not found in the table,Y=[].

%    Copyright (c) 1996 by Prentice-Hall,Inc.

[rt, ct]=size(tab);
if length(val) > 1,    error(' VAL must be a scalar. '),    end
if col>ct|col < 1,    error(' Chosen column outside table width. '),    end
if rt < 2,    error(' Table too small or not oriented in columns. '),    end

above=tab(: , col) > val;              %    True where > VAL
below=tab(: , col) < val;              %    True where < VAL
equal=tab(: , col) = = val;            %    True where = VAL

if all(above = = 0) | all(below = = 0),          %    handle simplest case
      y=tab(find(equal), : );    return
end
pslope=find(below(1:rt-1)&above(2:rt));     %    indices where slope is +
nslope=find(below(2:rt)&above(1:rt-1));     %    indices where slope is -

ib=sort([pslope; nslope+1]);         %    put indices below in order
ia=sort([nslope; pslope+1]);         %    put indices above in order
ie=find(equal);                       %    indices where equal to val

[tmp,ix]=sort( [ib, ie] );             %    find where equals fit in result
ieq=ix > length(ib);                  %    True where equals values fit
ry=length(tmp);                       %    # of rows in result y

y=zeros(ry, ct);          %    poke data into a zero matrix

alpha=(val-tab(ib,col))./(tab(ia,col)-tab(ib,col));
alpha=alpha(: , ones(1, ct));         %    duplicate for all columns
y(~ieq, : )=alpha.*tab(ia, : )+(1-alpha).*tab(ib, : );         %    interpolated values

y(ieq, : )=tab(ie, : );       %    equal values
y( : , col)=val*ones(ry, 1);       %    remove roundoff error
```

正如所见的，**mminterp** 利用了 **find** 和 **sort** 函数、逻辑数组和数组操作技术。**没有 For 循环和 While 循环**。不论用其中哪一种技术来实现将使运行变慢，尤其对大的表。注意

mminterp 与含有大于或等于 2 的任意数列的表一起工作，如同函数 **interp1** 一样。而且，在这种情况下，插值变量可以是任意的列。例如，

```
» z=sin(pi*x);            %    add more data to table

» table=[x; y; z].' ;

» t=mminterp(table, 2, 1.1)        %    same interpolation as earlier
t =
      0.5281      1.1000      0.9930
      0.9580      1.1000      0.1314
      1.5825      1.1000     -0.9639
      1.8847      1.1000     -0.3533

» t=mminterp(table, 3, -.5)         %    second third column now
t =
      1.1669      0.7316     -0.5000
      1.8329      1.1377     -0.5000
      3.1671      0.9639     -0.5000
      3.8331      1.0187     -0.5000
```

这些最后的结果估计了 **x** 和 **y** 在 **z= -.5** 处的值。

尽管逐条地对函数 **mminterp** 解释如何工作是很有帮助的，但这样做要求有更多的篇幅和时间。解释 **mminterp** 如何工作最容易的方法是创建一个小表格，然后，在重要的语句末尾删除分号以后，调用函数。这样，中间值将帮助用户理解函数是如何找到与所需值相符的数据值以及如何执行插值。

前面已阐述了 **interp1** 的用法。当用于线性插值时，只要所要求的插值点的个数少，**interp1** 工作很好。在要求许多插值点情况下，由于所用的算法，**interp1** 工作较慢。为了克服这个问题，**精通 MATLAB 工具箱**包括了函数 **mmtable**，它的帮助文本是：

```
»help table
MMTABLE 1-D Monotonic Table Search by Linear Interpolation.
   YI=MMTABLE(TAB,COL,VALS) linearly interpolates the table TAB
   searching for values VALS in the column COL.

   TAB(:,COL) must be monotonic, but need NOT be equally spaced.
   YI has as many rows as VALS and as many columns TAB
   NaNs are returned where VALS are outside the range of TAB(:,COL).

   YI=MMTABLE(TAB,VALS) interpolates using COL=1 and does not return
   TAB(:,1) in Y.    This matches the usage of TABLE1(TAB,X0).

   YI=MMTABLE(X,Y,XI) interpolates the vector X to find YI associated
with XI. This match the usage of INTERP1(X,Y,XI)
```

This routine is 10X faster than TABLE1which is called by INTERP1.

MMTABLE 由线性插值实现一维单调表搜索
YI=MMTABLE(TAB,COL,VALS) 线性地对表 TAB 进行插值，在列 COL 中搜索值为 VALS

TAB(:,COL)必须是单调的，但不必等价地生成空间。
YI 与 VALS 有同样的行和与 TAB 有同样的列。
当 VALS 超出 TAB(:,COL)的范围，返回 NaNs.

YI=MMTABLE(TAB,VALS) 使用 COL=1 进行插值，不返回在 Y 中的 TAB(:,1)
这和 TABLE1(TAB,X0)的用法匹配。

YI=MMTABLE(X,Y,XI) 为了找出 YI 和 XI 的关系，对向量 X 进行插值。
这和 INTERP1(X,Y,XI)的用法匹配。

这个例程比由 INTERP1 调用 TABLE1 快 10 倍。

正如前面描述的，可以用几种方式调用 **mmtable**。此外，要插值的列或向量不需要线性间隔。由于这个原因，**mmtable** 比 **ilinear** 函数更普遍。在 MATLAB 版本 5 中，**interp1** 将用 **ilinear** 来实现线性插值。

11.5 小结

下面的表 11.1 总结了在 MATLAB 中所具有的曲线拟合和插值函数。

表 11.1

曲 线 拟 合 和 插 值 函 数	
polyfit(x, y, n)	对描述 n 阶多项式 y=f(x)的数据 进行最小二乘曲线拟合
interp1(x, y, xo)	1 维线性插值
interp1(x, y, xo, ' spline ')	1 维 3 次样条插值
interp1(x, y, xo, ' cubic ')	1 维 3 次插值
interp2(x, y, Z, xi, yi)	2 维线性插值
interp2(x, y, Z, xi, yi, ' cubic ')	2 维 3 次插值
interp2(x, y, Z, xi, yi, ' nearest ')	2 维最近邻插值

第 12 章 三次样条

众所周知，使用高阶多项式的插值常常产生病态的结果。目前，有多种消除病态的方法。在这些方法中，三次样条是最常用的一种。在 MATLAB 中，实现基本的三次样条插值的函数有 **spline**，**ppval**，**mkpp** 和 **unmkpp**。在这些函数中，仅 **spline** 在《MATLAB 参考指南》中有说明。下面几节，将展示在 M 文件函数中实现三次样条的基本特征。

12.1 基本特征

在三次样条中，要寻找三次多项式，以逼近每对数据点间的曲线。在样条术语中，这些数据点称之为**断点**。因为，两点只能决定一条直线，而在两点间的曲线可用无限多的三次多项式近似。因此，为使结果具有唯一性。在三次样条中，增加了三次多项式的约束条件。通过限定每个三次多项式的一阶和二阶导数，使其在断点处相等，就可以较好地确定所有内部三次多项式。此外，近似多项式通过这些断点的斜率和曲率是连续的。然而，第一个和最后一个三次多项式在第一个和最后一个断点以外，没有伴随多项式。因此必须通过其它方法确定其余的约束。最常用的方法，也是函数 **spline** 所采用的方法，就是采用非扭结**(not-a-knot)**条件。这个条件强迫第一个和第二个三次多项式的三阶导数相等。对最后一个和倒数第二个三次多项式也做同样地处理。

基于上述描述，人们可能猜想到，寻找三次样条多项式需要求解大量的线性方程。实际上，给定 N 个断点，就要寻找 N-1 个三次多项式，每个多项式有 4 个未知系数。这样，所求解的方程组包含有 4*(N-1)个未知数。把每个三次多项式列成特殊形式，并且运用各种约束，通过求解 N 个具有 N 个未知系数的方程组，就能确定三次多项式。这样，如果有 50 个断点，就有 50 个具有 50 个未知系数的方程组。幸好，用稀疏矩阵，这些方程式能够简明地列出并求解，这就是函数 **spline** 所使用的计算未知系数的方法。

12.2 分段多项式

在最简单的用法中，**spline** 获取数据 **x** 和 **y** 以及期望值 **xi**，寻找拟合 **x** 和 **y** 的三次样条内插多项式，然后，计算这些多项式，对每个 **xi** 的值，寻找相应的 **yi**。例如：

```
>>x=0 : 12;
>>y=tan(pi*x/25);
>>xi=linspace(0, 12);
>>yi=spline(x, y, xi)
>>plot(x, y, ‘ o ‘, xi, yi), title(‘ Spline fit ‘)
```

（见图 12.1 样条拟合）

这种方法适合于只需要一组内插值的情况。不过，如果需要从相同数据集里获取另一

组内插值，再次计算三次样条系数是没有意义的。在这种情况下，可以调用仅带前两个参量的 **spline**：

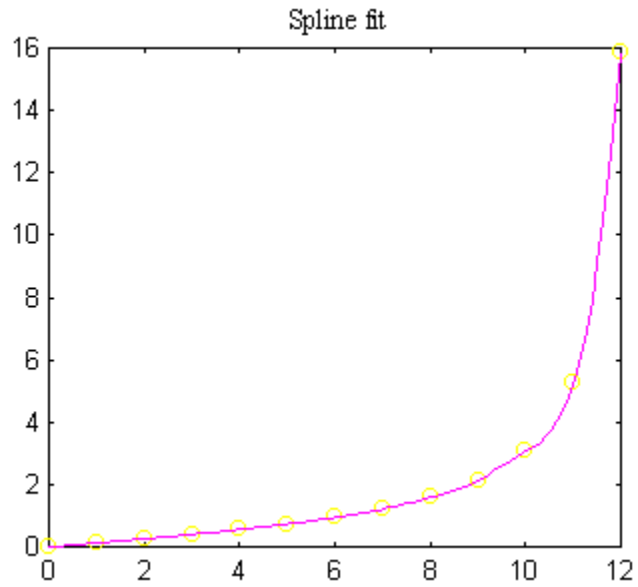


图 12.1 样条拟合

```
>>pp=spline(x, y)
pp =
    Columns 1 through 7
     10.0000    1.0000   12.0000         0    1.0000    2.0000    3.0000
    Columns 8 through 14
      4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000
    Columns 15 through 21
     11.0000   12.0000    4.0000    0.0007    0.0007    0.0010    0.0012
    Columns 22 through 28
      0.0024    0.0019    0.0116   -0.0083    0.1068   -0.1982    1.4948
    Columns 29 through 35
      1.4948   -0.0001    0.0020    0.0042    0.0072    0.0109    0.0181
    Columns 36 through 42
      0.0237    0.0586    0.0336    0.3542   -0.2406    4.2439    0.1257
    Columns 43 through 49
      0.1276    0.1339    0.1454    0.1635    0.1925    0.2344    0.3167
    Columns 50 through 56
      0.4089    0.7967    0.9102    4.9136         0    0.1263    0.2568
    Columns 57 through 63
      0.3959    0.5498    0.7265    0.9391    1.2088    1.5757    2.1251
    Columns 64 through 65
      3.0777    5.2422
```

当采用这种方式调用时，**spline** 返回一个称之为三次样条的 **pp 形式**或**分段多项式形式**

的数组。这个数组包含了对于任意一组所期望的内插值和计算三次样条所必须的全部信息。给定 pp 形式，函数 **ppval** 计算该三次样条。例如，

```
>>yi=ppval(pp, xi);
```

计算先前计算过的同样的 yi。
类似地，

```
>>xi2=linspace(10, 12);
>>yi2=ppval(pp, xi2);
```

运用 pp 形式，在限定的更细区间[10，12]内，再次计算该三次样条。

```
>>xi3=10 : 15
>>yi3=ppval(pp, xi3)
yi3 =
    3.0777    5.2422   15.8945   44.0038   98.5389  188.4689
```

它表明，可在计算三次多项式所覆盖的区间外，计算三次样条。当数据出现在最后一个断点之后或第一个断点之前时，则分别运用最后一个或第一个三次多项式来寻找内插值。

上述给定的三次样条 pp 形式，存储了断点和多项式系数，以及关于三次样条表示的其它信息。因为，所有信息都被存储在单个向量里，所以这种形式在 MATLAB 中是一种方便的数据结构。当要计算三次样条表示时，必须把 pp 形式分解成它的各个表示段。在 MATLAB 中，通过函数 **unmkpp** 完成这一过程。运用上述 pp 形式，该函数给出如下结果：

```
>>[break, coefs, npolys, ncoefs]=unmkpp(pp)
breaks =
  Columns 1 through 12
     0     1     2     3     4     5     6     7     8     9    10    11
  Column 13
    12
coefs =
    0.0007   -0.0001    0.1257         0
    0.0007    0.0020    0.1276    0.1263
    0.0010    0.0042    0.1339    0.2568
    0.0012    0.0072    0.1454    0.3959
    0.0024    0.0109    0.1635    0.5498
    0.0019    0.0181    0.1925    0.7265
    0.0116    0.0237    0.2344    0.9391
   -0.0083    0.0586    0.3167    1.2088
    0.1068    0.0336    0.4089    1.5757
   -0.1982    0.3542    0.7967    2.1251
    1.4948   -0.2406    0.9102    3.0777
    1.4948    4.2439    4.9136    5.2422
```

```
npolys =
     12
ncoefs =
      4
```

这里 **break** 是断点，**coefs** 是矩阵，它的第 i 行是第 i 个三次多项式，**npolys** 是多项式的数目，**ncoefs** 是每个多项式系数的数目。注意，这种形式非常一般，样条多项式不必是三次。这对于样条的积分和微分是很有益的。

给定上述分散形式，函数 **mkpp** 恢复了 pp 形式。

```
>>pp=mkpp(break, coefs)
pp =
  Columns 1 through 7
   10.0000    1.0000   12.0000         0    1.0000    2.0000    3.0000
  Columns 8 through 14
    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000
  Columns 15 through 21
   11.0000   12.0000    4.0000    0.0007    0.0007    0.0010    0.0012
  Columns 22 through 28
    0.0024    0.0019    0.0116   -0.0083    0.1068   -0.1982    1.4948
  Columns 29 through 35
    1.4948   -0.0001    0.0020    0.0042    0.0072    0.0109    0.0181
  Columns 36 through 42
    0.0237    0.0586    0.0336    0.3542   -0.2406    4.2439    0.1257
  Columns 43 through 49
    0.1276    0.1339    0.1454    0.1635    0.1925    0.2344    0.3167
  Columns 50 through 56
    0.4089    0.7967    0.9102    4.9136         0    0.1263    0.2568
  Columns 57 through 63
    0.3959    0.5498    0.7265    0.9391    1.2088    1.5757    2.1251
  Columns 64 through 65
    3.0777    5.2422
```

因为矩阵 **coefs** 的大小确定了 **npolys** 和 **neofs**，所以 **mkpp** 不需要 **npolys** 和 **ncoefs** 去重构 pp 形式。pp 形式的数据结构仅在 **mkpp** 中给定为 **pp=[10 1 npolys break(:)' ncoefs coefs(:)']**。前两个元素出现在所有的 pp 形式中，它们作为确认 pp 形式向量的一种方法。

12.3 积分

在大多数情况下，需要知道由三次样条所描述的、自变量为 x 的函数所包含的面积。也就是，如果这个函数记为 y=f(x)，我们感兴趣的是计算：

$S(x)=\int_{x1}^{x} s(x)dx$　其中，是 s(x1)=0

式中的 x1 是第一个样条的断点。因为 s(x)由被连接的三次多项式组成，其中第 k 个三次多项式为：

$s_k(x)=a_k(x-x_k)^3+ b_k(x-x_k)^2+ c_k(x-x_k)+d_k$,　$x_k<=x<= x_{k+1}$

并且该函数在区间 $x_k<=x<= x_{k+1}$ 所含的面积为：

$$S_k(x)=\int_{x1}^{x} s_k(x)dx = a_k/4(x-x_k)^4+b_k/3(x-x_k)^3+c_k/2(x-x_k)^2+d_k(x-x_k)$$

三次样条下的面积容易求得为：

$$S(x)=\sum_{i=1}^{k-1}\int_{x_i}^{x_{i=1}} s_i(x)dx+\int_{x_k}^{x} s_k(x)dx$$　式中 $x_k<=x<= x_{k+1}$

或者

$$S(x)=\sum_{i=1}^{k-1} S_i(x_i+1)+S_k(x)$$　式中 $x_k<=x<= x_{k+1}$

上式相加项是所有处理的三次多项式面积的累加和。照此，因为 $S_k(x)$是一个多项式，所以该面积很容易计算，且在描述 S(x)的多项式中可形成常数项。有了上述理解，积分本身可写成样条形式。在这种情况下，因为单个多项式具有 4 阶，所以积分是四次样条。

MATLAB 中使用的 pp 形式支持任意阶的样条，所以在《精通 MATLAB 工具箱》中的函数 **spintgrl** 体现了上述样条积分。该函数的主体如下：

```
function z=spintgrl(x, y, xi)
%   SPINTGRL Cubic Spline Integral Interpolation
%   YI=SPINTRGL(X, Y, XI) uses cubic spline interpolation to fit the date in X and Y, integrates
%   the spline and returns values of the integral evaluated at the points in XI.
%
%   PPI=SPINTGRL(PP) returns the piecewise polynomial vector PPI
%   describing the integral of the cubic spline described by
%   the piecewise polynomial in PP. PP is returned by the function
%   SPLINE and is a data vector containing all information to
%   evaluate and manipulate a spline.
```

```
%
%    YI=SPINTGRL(PP, XI) integrates the cubic splines given by
%    the piecewise polynomial PP, and returns the values of the
%    integral evaluated at the points in XI.
%
%    See also SPLINE, PPVAL, MKPP, UNMKPP, SPDERIV

%    Copyright    (c) 1996 by Prentice-Hall, Inc.
if nargin==3
pp=spline(x, y)
else
     pp=x;
end
[br, co, npy, nco]=unmkpp(pp);    %    take apart pp
if pp(1)==10
     error(' Spline data does not have the correct form. ')
end
sf=nco : -1 : 1;    %    scale factors for integration
ico=[co ./ sf(ones(npy, 1), : ) zeros(npy, 1)];    %    integral coefficients
nco=nco+1;    %    spline order increases
for k=2 : npy    %    find constant terms
     ico(k, nco)=polyval(ico(k-1, : ),br(k)-br(k-1));
end
ppi=mkpp(br, ico);    %    build pp form for integral
if nargin==1
     z=ppi;
elseif nargin==2
     z=ppval(ppi, y);
else
     z=ppval(ppi, xi);
```

```
end
```

考虑如下运用 spintgrl 的例子：

```
>>x=(0:  .1:  1)*2*pi;
>>y=sin(x);  %  create rough data
>>pp=spline(x, y);  %  pp-form fitting rough data
>>ppi=spintgrl(pp);  %  pp-form of integral
>>xi=linspace(0, 2*pi);  %  finer points for interpolation
>>yi=ppval(pp, xi);  %  evaluate curve
>>yyi=ppval(ppi, xi);  %  evaluate integral
>>plot(x, y, ' o ', xi, yi, xi, yyi, ' - ')  %  plot results
```

注意，如图 12.2 所示，它定性地证明了恒等式：

$$\int_{x_1}^{x} \sin(x)dx = 1 - \cos(x)$$

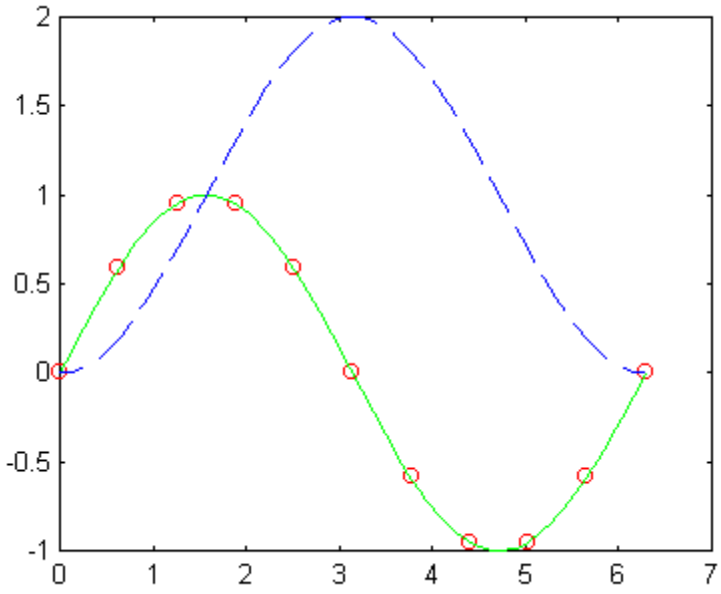


图 12.2　函数的积分图形

12.4　微分

如同人们对样条积分感兴趣一样，一个由样条所描述的函数的微分或斜率也是很有用的。给定第 k 个三次多项式为：

$s_k(x)=a_k(x-x_k)^3+ b_k(x-x_k)^2+ c_k(x-x_k)+d_k, \quad x_k<=x<= x_{k+1}$

容易写出其导数为：

$$\frac{ds_k(x)}{dx} = 3a_k(x-x_k)^2 + 2b_k(x-x_k) + c_k, x_k \le x \le x_{k+1}$$

如同样条积分，样条微分也是一种样条。不过，在这种情况下，这个多项式为二阶，所以它是二次样条。

基于上述描述，《精通 MATLAB 工具箱》中的函数 **spderiv** 实施样条微分。**spderiv** 的主体为：

```
function z=spderiv(x, y, xi)
%   SPDERIV Cubic Spline Derivative Interpolation
%   YI=SPDERIV(X, Y, XI) uses cubic spline interpolation to fit the
%   data in X and Y, differentiates the spline and returns values
%   of the spline derivatives evaluated at the points in XI.
%
%   PPD=SPDERIV(PP) returns the piecewise polynomial vector PPD
%   describing the cubic spline derivative of the curve described by
%   the piecewise polynomial in PP. PP is returned by the function
%   SPLINE and is a data vector containing all information to
%   evaluate and manipulate a spline.
%
%   YI=SPDERIV(PP, XI) differentiates the cubic spline given by
%   the piecewise polynomial PP, and returns the value of the
%   spline derivatives evaluated at the points in XI.
%
%   See also SPLINE, PPVAL, MKPP, UNMKPP, SPINTGRL

%   Copyright (c) 1996 by Prentice-Hall, Inc.

if nargin==3
    pp=spline(x, y);
else
    pp=x;
end
[br, co, npy, nco]=unmkpp(pp);   %   take apart pp
if nco==1 | pp(1)~=10
    error(' Spline data does not have the correct PP form. ')
end
sf=nco-1 : -1 : 1;   %   scale factors for differentiation
dco=sf(ones(npy, 1),:).*co( : , 1 : nco-1);   %   derivative coefficients
ppd=mkpp(br, dco);   %   build pp form for derivative
```

```
if nargin==1
    z=ppd;
elseif nargin==2
    z=ppval(ppd, y);
else
    z=ppval(ppd, xi);
end
```

为演示 **spderiv** 的使用，考虑如下例子：

```
>>x=(0 :  .1 :  1)*2*pi;  %  same data as earlier
>>y=sin(x);
>>pp=spline(x, y);  %  pp-form fitting rough data
>>ppd=spderiv(pp);  %  pp-form of derivative
>>xi=linspace(0, 2*pi);  %  finer points for interpolation
>>yi=ppval(pp, xi);  %  evaluate curve
>>yyd=ppval(ppd, xi);  %  evaluate derivative
>>plot(x, y, ' o ', xi, yi, xi, yyd, ' - ')  %  plot results
```

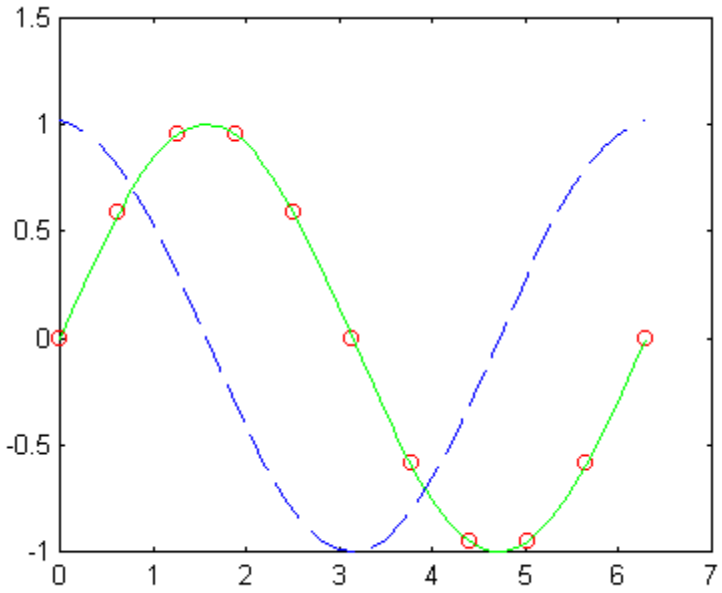


图 12.3　函数的微分图形

注意，图 12.3 定性地证明了恒等式：

$$\frac{d}{dx}\sin(x) = \cos(x)$$

12.5　小结

下面的两个表总结了本章所讨论的样条函数。

表 12.1

三次样条函数	
yi=spline(x,y,xi)	y=f(x)在 **xi** 中各点的三次样条插值
pp=spline(x,y)	返回 y=f(x)的分段多项式的表示
yi=ppval(pp,xi)	计算 **xi** 中各点的分段多项式
[break,coefs,npolys,ncoefs]=unmkpp(pp)	分解分段多项式的表示
pp=mkpp(break,coefs)	形成分段多项式的表示

表 12.2

精通 **MATLAB** 的样条函数	
yi=spintgrl(x,y,xi)	在 xi 各点对 y=f(x)积分的三次样条的插值
ppi=spintgrl(pp)	给定 y=f(x)的分段多项式的表示，返回 y=f(x)积分的分段多项式的表示。
yi=spintgrl(pp,xi)	给定 y=f(x)的分段多项式的表示，计算点 **xi** 的值，寻找 y=f(x)积分的分段多项式的描述。
yi=spderiv(x,y,xi)	在 **xi** 各点对 y=f(x)微分的三次样条的插值
ppi=spderiv(pp)	给定 y=f(x)的分段多项式的表示，返回 y=f(x)微分的分段多项式的表示。
yi=spderiv(pp,xi)	给定 y=f(x)的分段多项式的表示，计算点 **xi** 的值，寻找 y=f(x)微分的分段多项式的表示。

第 13 章　数值分析

每当难以对一个函数进行积分、微分或者解析上确定一些特殊的值时，就可以借助计算机在数值上近似所需的结果。这在计算机科学和数学领域，称之为数值分析。至此，可以猜到，MATLAB 提供了解决这些问题的工具。本章将介绍这些工具的使用。

13.1　绘图

说到绘图，只要计算函数在某一区间的值，并且画出结果向量，这样就得到了函数的图形。在大多数情况下，这就足够了。然而，有时一个函数在某一区间是平坦的并且无激励，而在其它区间却失控。在这种情况下，运用传统的绘图方法会导致图形与函数真正的特性相去甚远。MATLAB 提供了一个称为 **fplot** 的巧妙的绘图函数。该函数细致地计算要绘图的函

数，并且确保在输出的图形中表示出所有的奇异点。该函数的输入需要知道以字符串表示的被画函数的名称以及 2 元素数组表示的绘图区间。例如：

```
>>fplot(‘ humps ‘ , [0    2])
>>title(‘ FPLOT OF HUMPS ‘)
```

在 0 和 2 之间计算函数 **humps**，并显示该函数的图形。（见图 13.1)。

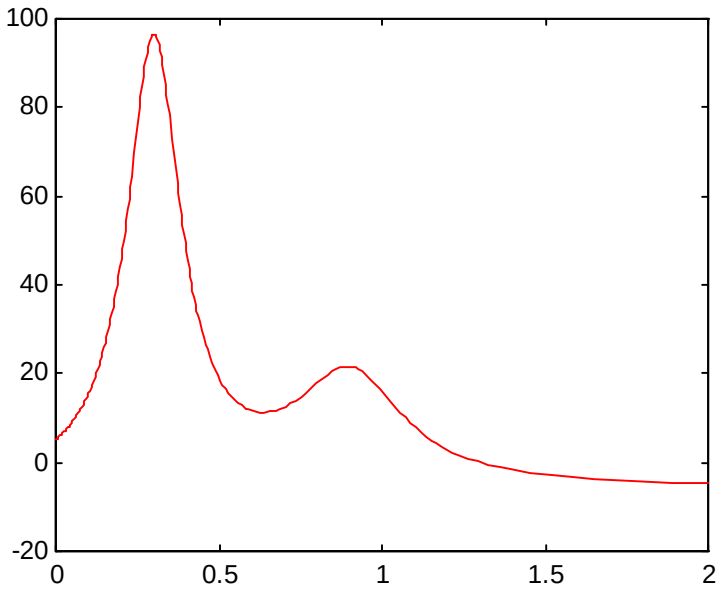


图 13.1　函数 humps 的图形

在这个例子中，‘ **humps** ‘是 MATLAB 的 M 文件函数。

```
function y=humps(x)
%    HUMPS A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
%    HUMPS(X) is a function with strong maxima near x= .3 and x= .9.
%    See QUADDEMO,    ZERODEMO and FPLOTDEMO.
%    Copyright (c) 1984-93 by The MathWorks, Inc.
y=1 ./ ((x - .3) .^ 2+ .01)+1 ./ ((x - .9) .^ 2+ .04) - 6;
```

fplot 适用于任何具有单输入和单输出向量的函数 M 文件。即如同 **humps**，输出变量 y 返回一个与输入 **x** 同样大小的数组，在数组到数组意义上 **y** 和 **x** 有联系。在使用 **fplot**（以及其它数值分析函数）的过程中，最普遍犯的错误是忘记把函数名加上引号。即 **fplot** 需要知道字符串形式的函数名。如果输入 **fplot(humps , [0 , 2])**，MATLAB 认为 **humps** 是工作空间中的一个变量，而不是函数的名称。注意把变量 **humps** 定义为所需要的字符串，就可避免这个问题。

```
>>humps=‘ humps ‘;
>>fplot(hump , [0    2])
```

这时，MATLAB 从变量 **humps** 中获得字符串‘ **humps**‘。

对于可表示成一个字符串的简单的函数，如 $y = 2e^{-x}\sin(x)$，**fplot** 绘制这类函数的曲线时，不用建立 M 文件，只需把 x 当作自变量，把被绘图的函数写成一个完整的字符串。

```
>>f=‘ 2*exp(-x) .* sin(x) ‘;
```

式中，运用数组乘法定义了函数 $f(x) = 2e^{-x}\sin(x)$

```
>>fplot(f , [0    8]);
>>title(f) , xlabel(‘x‘)
```

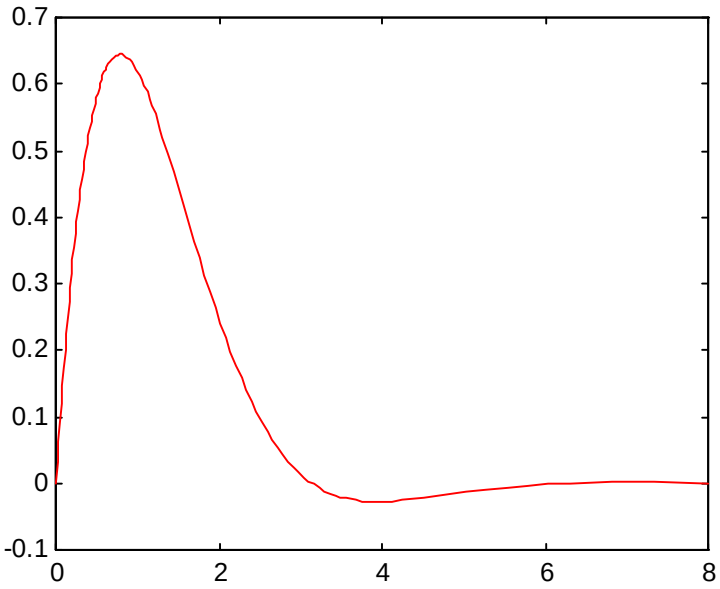


图 13.2　$f(x) = 2e^{-x}\sin(x)$ 的曲线

在区间 $0 \le x \le 8$ 绘出上述函数，产生如图 13.2 所示的图形。

除了这些基本特性，函数 **fplot** 还有很多强大的功能，有关详细的信息，参阅《MATLAB 参考指南》或在线帮助。

13.2　极小化

作图除了提供视觉信息外，还常常需要确定一个函数的其它更多的特殊属性。在许多应用中，特别感兴趣的是确定函数的极值，即**最大值**（峰值）和**最小值**（谷值）。数学上，可通过确定函数导数（斜率）为零的点，解析上求出这些极值点。检验 **humps** 的图形在峰值和谷值点上的斜率就很容易理解这个事实。显然，如果定义的函数简单，则这种方法常常

奏效。然而，即使很多容易求导的函数，也常常很难找到导数为零的点。在这种情况下，以及很难或不可能解析上求得导数的情况下，必须数值上寻找函数的极值点。MATLAB 提供了两个完成此功能的函数 **fmin** 和 **fmins**。这两个函数分别寻找一维或 n 维函数的最小值。这里仅讨论 **fmin**。有关 **fmins** 的详细信息，参阅《MATLAB 参考指南》。因为 f(x)的最大值等于-f(x)的最小值，所以，上述 **fmin** 和 **fmins** 可用来求最大值和最小值。如果还不清楚，把上述图形倒过来看，在这个状态下，峰值变成了谷值，而谷值则变成了峰值。

为了解释求解一维函数的最小值和最大值，再考虑上述例子。从图 13.2 可知，在 x_{max}=0.7 附近有一个最大值，并且在 x_{min}=4 附近有一个最小值。而这些点的解析值为：$x_{\max} = \pi / 4 \approx 0.785$ 和 $x_{\min} = 5\pi / 4 \approx 3.93$。为了方便，用文本编辑器编写一个脚本 M 文件，并用 fmin 寻出数值上极值点，给出函数主体如下：

```
%    ex_fmin.m
fn=‘ 2*exp(-x)*sin(x) ‘;          %    define function for min
xmin=fmin(fn , 2 , 5)             %    search over range 2<x<5
emin=5*pi / 4-xmin                %    find error
x=xmin;                           %    need x since fn has x as its variable
ymin=eval(fn)                     %    evaluate at xmin
fx=‘ -2*exp(-x)*sin(x) ‘;         %    define for max:note minus sign
xmax=fmin(fx , 0 , 3)             %    search over range 0<x<3
emax=pi / 4-xmax                  %    find error
x=xmax;                           %    need x since fn has x as its variable
ymax=eval(fn)                     %    evaluate at xmax
```

下面是 M 文件的运行结果：

```
>>ex-fmin
xmin =
      3.9270
emin =
   1.4523e-006
ymin =
    -0.0279
xmax =
      0.7854
emax =
 -1.3781e-005
ymax =
      0.6448
```

这些结果与上述图形非常吻合。注意，**fmin** 的工作方式很像 **fplot**。要计算的函数可用一个函数 M 文件表达，或者只给出一个 x 为自变量的字符串。上述例子就是使用后一种方法。这个例子也使用了函数 **eval**，它获取一个字符串，并解释它，如同在 MATLAB 提示符下输入该字符串。由于要计算的函数以 **x** 为自变量的字符串形式给出，那么设置 **x** 等于 **xmin**

和 **xmax**，允许 **eval** 计算该函数，找到 **ymin** 和 **ymax**。

最后，特别注意，求数值上的最小值包含一个搜索过程，**fmin** 不断计算函数值，寻求其最小值。如果计算的函数需要很大的计算量，或者该函数在搜索区间不止一个最小值，则该搜索过程所花的时间比较长。在有些情况下，搜索过程根本找不到结果。当 **fmin** 找不到最小值时，它会停止运行并提供解释。

与函数 **fmin** 一样，函数 **fmins** 搜索最小值。不过，**fmins** 搜索向量的标量函数的最小值。即 **fmins** 寻找

$$\min_{x} f(x)$$

这里 **x** 是函数 f(.)的向量参数，函数 f(.)返回标量值。函数 fmins 利用单纯形法求最小值，它不需要精确的梯度计算。任何一种优化工具箱中具有更多扩展的优化算法

13.3 求零点

正如人们对寻找函数的极点感兴趣一样，有时寻找函数过零或等于其它常数的点也非常重要。一般试图用解析的方法寻找这类点非常困难，而且很多时候是不可能的。在上述函数 **humps** 的图中(如图 13.3 所示)，该函数在 x=1.2 附近过零。

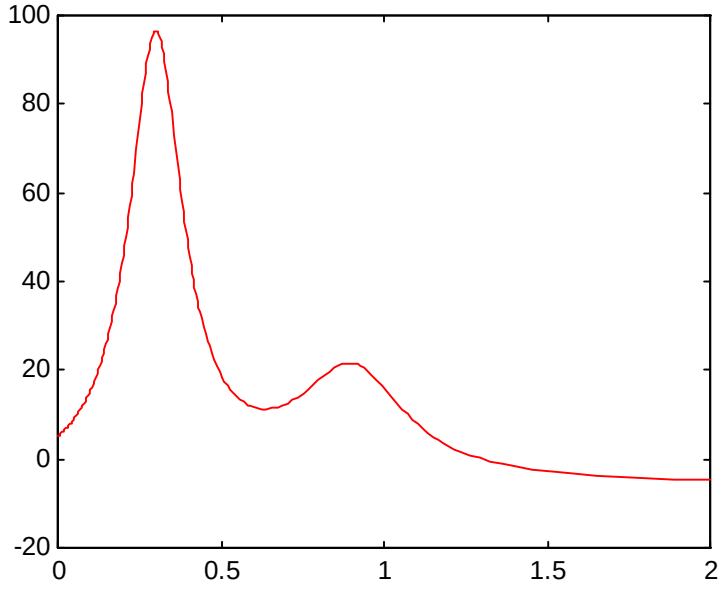


图 13.3 humps 函数的图形

MATLAB 再一次提供了该问题的数值解法。函数 **fzero** 寻找一维函数的零点。为了说明该函数的使用，让我们再运用 **humps** 例子。

```
>>xzero=fzero(‘ humps ‘ , 1.2)        %    look for a zero near 1.2
```

```
xzero=
      1.2995
>>yzero=humps(xzero , 1.2)          %    evaluate at xzero
yzero=
    3.5527e-15
```

所以，humps 的零点接近于 1.3。如前所述，寻找零点的过程可能失败。如果 **fzero** 没有找到零点，它将停止运行并提供解释。

当调用函数 **fzero** 时，必须给出该函数的名称。但由于某种原因，它不能接受以 x 为自变量的字符串来描述的函数。这样，即使在 **fplot** 和 **fmin** 中都具有的这个特性，**fzero** 将不工作。

fzero 不仅能寻找零点，它还可以寻找函数等于任何常数值的点。仅仅要求一个简单的再定义。例如，为了寻找 f(x)=c 的点，定义函数 g(x)=f(x)-c，然后，在 **fzero** 中使用 g(x)，就会找出 g(x)为零的 x 值，它发生在 f(x)=c 时。

13.4 积分

一个函数的积分或面积也是它的另一个有用的属性。MATLAT 提供了在有限区间内，数值计算某函数下的面积的三种函数：**trap2 , quad** 和 **quad8**。函数 **trapz** 通过计算若干梯形面积的和来近似某函数的积分，这些梯形如图 13.4 所示，是通过使用函数 **humps** 的数据点形成。

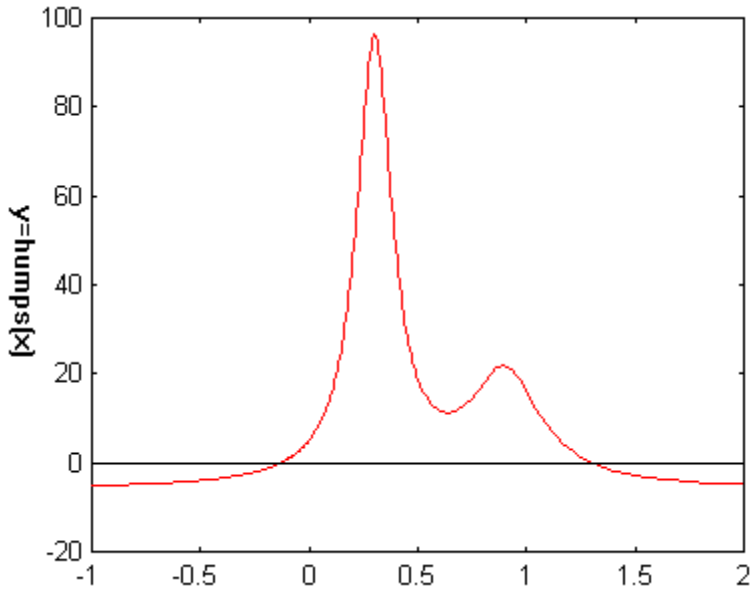


图 13.4 粗略的梯形逼近曲线下的面积示意图

从图中可明显地看出，单个梯形的面积在某一段欠估计了函数真正的面积，而在其它段又过估计了函数的真正面积。如同线性插值，当梯形数目越多时，函数的近似面积越准确。例如，在图 13.4 中，如果我们大致增加一倍数目的梯形，我们得到如下页（如图 13.5）所

示的更好的近似结果。

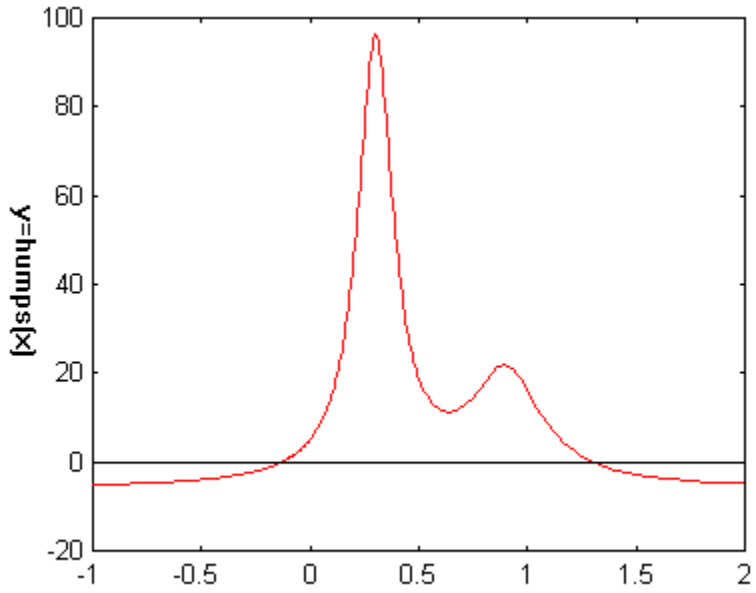


图 13.5 较好的梯形逼近曲线下的面积示意图

对如上所示的两个曲线，用 **trapz** 在区间-1<x<2 上计算 y=humps(x)下面的面积：

```
>>x=-1 : 0.17 : 2;                    %   rough approximation
>>y=humps(x);
>>area=trapz(x , y)                   %   call trapz just like the plot command
area =
    25.9174
>>x=-1 : 0.07 : 2;                    %   better approximation
>>y=humps(x);
>>area=trapz(x , y)
area =
    26.6243
```

自然地，上述两个结果不同。基于对图形的观察，粗略近似可能低估了实际面积。除非特别精确，没有准则说明哪种近似效果更好。很明显，如果人们能够以某种方式改变单个梯形的宽度，以适应函数的特性，即当函数变化快时，使得梯形的宽度变窄，这样就能够得到更精确的结果。

MATLAB 的函数 **quad** 和 **quad8** 是基于数学上的正方形概念来计算函数的面积，这些积分函数的操作方式一样。为获得更准确的结果，两个函数在所需的区间都要计算被积函数。此外，与简单的梯形比较，这两个函数进行更高阶的近似，而且 **quad8** 比 **quad** 更精确。这两个函数的调用方法与 **fzero** 相同，即

```
>>area=quad(‘ humps ‘ , -1 , 2)    %    find area between -1 and 2
area =
```

```
    26.3450
>>area=quad8(‘ humps ‘ , -1 , 2)
area =
    26.3450
```

注意，这两个函数返回完全相同的估计面积，而且这个估计值在两个 trapz 面积的估计值之间。有关 MATLAB 的积分函数的其它信息，参阅《MATLAB 参考指南》或在线帮助。

13.5 微分

与积分相反，数值微分非常困难。积分描述了一个函数的整体或宏观性质，而微分则描述一个函数在一点处的斜率，这是函数的微观性质。因此积分对函数的形状在小范围内的改变不敏感。而微分却很敏感。一个函数小的变化，容易产生相邻点的斜率的大的改变。

由于微分这个固有的困难，所以尽可能避免数值微分，特别是对实验获得的数据进行微分。在这种情况下，最好用最小二乘曲线拟合这种数据，然后对所得到的多项式进行微分。或用另一种方法，对该数据进行三次样条拟合，然后寻找如第 11 章所讨论的样条微分。例如，再次考虑第 11 章曲线拟合的例子。

```
>>x=[0   .1   .2   .3   .4   .5   .6   .7   .8   .9   1]
>>y=[-.447   1.978   3.28   6.16   7.08   7.34   7.66   9.56   9.48   9.30   11.2];   % data
>>n=2;   %   order of fit
>>p=polyfit(x , y , n)     %   find polynomial coefficients
p =
    -9.8108     20.1293     -0.0317
>>xi=linspace(0 , 1 , 100);
>>z=polyval(p , xi);       %   evaluate polynomial
>>plot(x , y , ‘ o ' , x , y , xi , z , ' : ')
>>xlabel(‘ x ‘) , ylabel(‘ y=f(x) ‘) , title(‘ Second Order Curve Fitting ‘)
```

在这种情况下，运用多项式微分函数 **polyder** 求得微分。

```
>>pd=polyder(p)
pd =
   -19.6217     20.1293
```

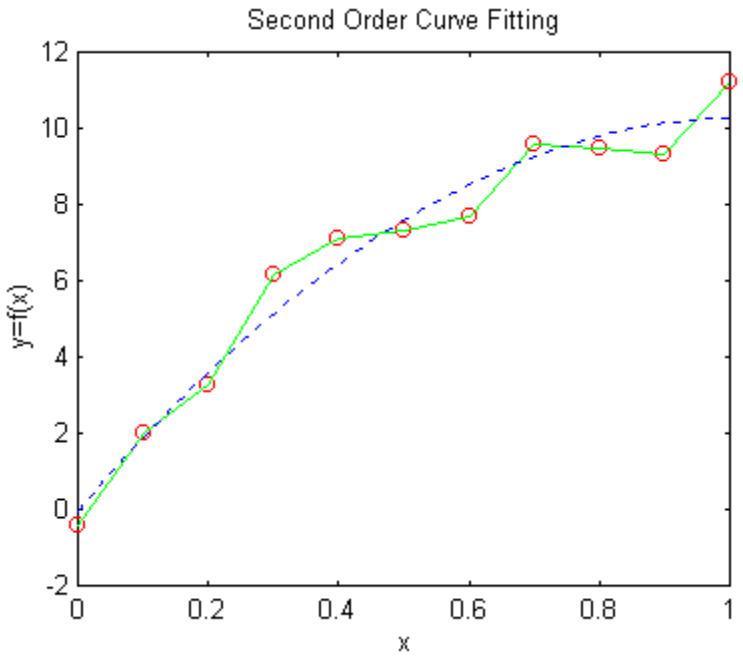


图 13.6 二次曲线拟合

$y = -9.8108x^2 + 20.1293x - 0.0317$ 的微分是 dy/dx=-19.6217x+20.1293。由于一个多项式的微分是另一个低一阶的多项式，所以还可以计算并画出该函数的微分。

```
>>z=polyval(pd , xi);    %    evaluate derivative
>>plot(xi , z)
>>xlabel(‘ x ‘) , ylabel(‘ dy/dx ‘) , title(‘ Derivative of a curve Fit Polynimial ‘)
```

(微分曲线如图 13.7 所示）

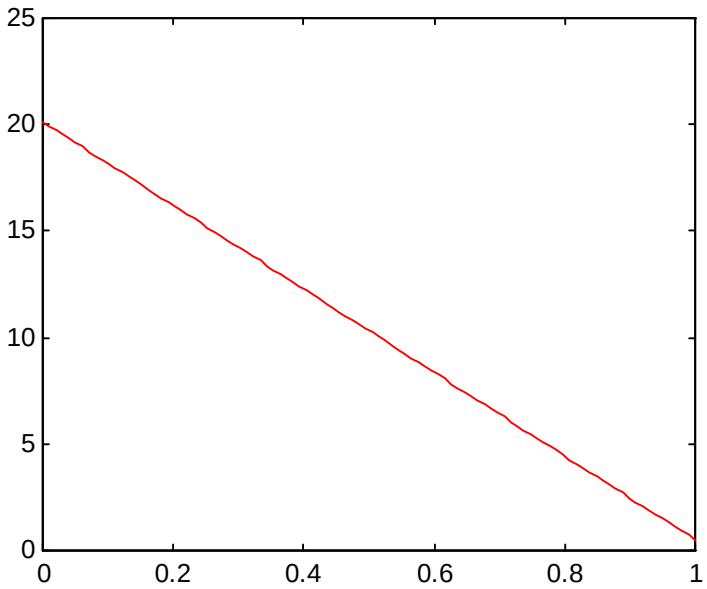


图 13.7 曲线拟合多项式微分

在这种情况下，拟合的多项式为二阶，使其微分为一阶多项式。这样，微分为一条直线，它意味该微分与 x 成线性变化。

给定一些描述某函数的数据，MATLAB 提供了一个计算其非常粗略的微分的函数。这个函数命名为 **diff**，它计算数组中元素间的差分。因为微分定义为：

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{(x+h) - (x)}$$

则 y=f(x)的微分可近似为：

$$\frac{dy}{dx} \approx \frac{f(x+h) - f(x)}{(x+h) - (x)}$$ 这里 h>0

它是 y 的有限差分除以 x 的有限差分。因为 **diff** 计算数组元素间的差分，所以在 MATLAB 中，可近似求得函数的微分。继续前一个例子：

```
>>dy=diff(y) ./ diff(x);          %   compute differences and use array division
>>xd=x(1 : length(x)-1);          %   create new x axis since dy is shorter than y
>>plot(xd , dy);
>>title(‘ Approximate Derivative Using DIFF ‘)
>>ylabel(‘ dy/dx ‘) , xlabel(‘ x ‘)
```

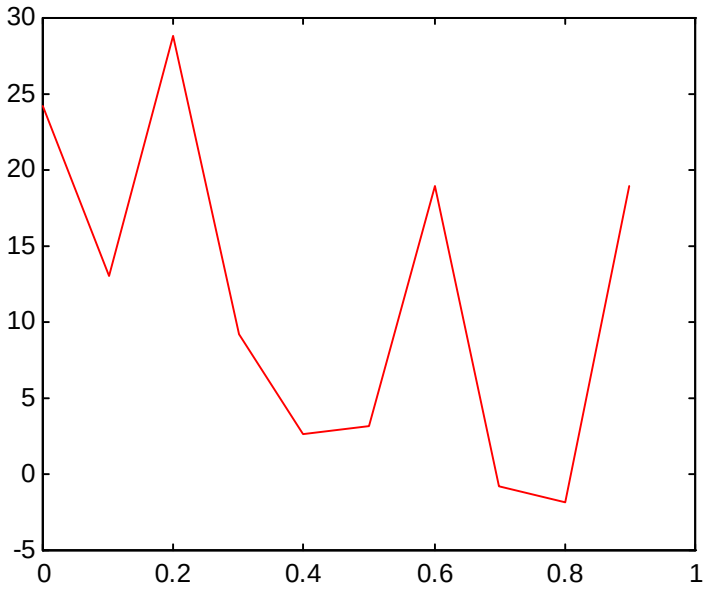


图 13.8 用 diff 得到的近似微分

由于 **diff** 计算数组元素间的差分，所以，其所得输出比原数组少了一个元素。这样，画微分曲线时，必须舍弃 x 数组中的一个元素。当舍弃 x 的第一个元素时，上述过程给出向后差分近似，而舍弃 x 的最后一个元素，则给出向前差分近似。比较上述两条曲线，显而易见，用有限差分近似微分会导致很差的结果，特别是被噪声污染了的数据。

13.6 微分方程

一般微分方程式描述系统内部变量的变化率如何受系统内部变量和外部激励，如输入，的影响。当常微分方程式能够解析求解时，可用 MATLAB 的符号工具箱中的功能找到精确解。在本书的后面将介绍该工具箱的一些特点。

在微分方程难以获得解析解的情况下，可以方便地在数值上求解。为了说明起见，考虑描述振荡器的经典的范得波（Var der Pol）微分方程。

$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$$

与所有的数值求解微分方程组的方法一样，高阶微分方程式必须等价地变换成一阶微分方程组。对于上述微分方程，通过重新定义两个新的变量，来实现这种变换。

令 $y_1=x$ 且 $y_2=dy/dx$
则 $dy1/dt=y_2$

$$dy_2 / dt = \mu(1-y_2) - y_1$$

根据这个微分方程组，可用 MATLAB 的函数 **ode23** 和 **ode45** 求出系统随时间变化的运动情况。调用这些函数时，需要编写一个函数 M 文件，给定当前时间及 y1 和 y2 的当前值，该函数返回上述导数值。MATLAB 中，这些导数由一个列向量给出。在本例中，这个列向量为 yprime。同样，y1 和 y2 合并写成列向量 y。所得函数 M 文件是：

```
function yprime=vdpol(t , y);
%   VDPOL(t , y) returns derivatives of the Van der Pol equation:
%
%   x ‘‘-mu *(1-x ^2)*x ‘+x=0 (‘ = d/dx , ‘‘ = d^2/dx^2)
%
%   let y(1)=x and y(2)=x‘
%
%   then y(1) ‘ = y(2)
%        y(2) ‘ = MU*(1-y(1)^2)*y(2)-y(1)

global MU %     choose 0<MU<10 in Command workspace
```

```
yprime=[y(2)    MU*(1-y(1)^2)*y(2)-y(1)];    %    output must be a column
```

给定这个完整地描述微分方程的函数，计算结果如下：

```
>>global MU %    define MU as a global variable in the Command Workspace
>>MU=2;    %    set global parameter to desired value
>>[t , y]=ode23(‘ vdpol ‘ , 0 , 30 , [1 ; 0]);    %    to=0 , tf=30 , yo=[1 ; 0]
>>y1=y( : , 1);    %    first column is y(1) versus time points in t
>>y2=y( : , 2);    %    second column is y(2)
>>plot(t , y1 , t , y2 , ‘ -- ‘)
>>xlabel(‘ Time , Second ‘) , ylabel(‘ Y(1) and Y(2) ‘)
>>title(‘ Van der Pol Solution for mu=2 ‘)
```

所得的图见图 13.9。

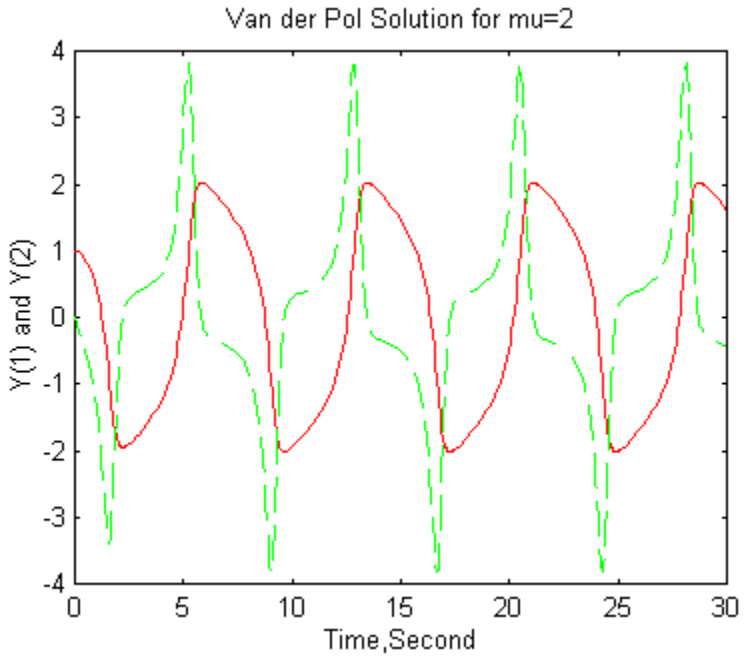


图 13.9 当 mu=2 时的范得波方程的运动曲线

在图 13.9 中，y2（虚线）是 y1（实线）的导数。传递给 **ode23** 的参数由 **ode23(f_name , to , tf , yo , to1)**描述。这里 **f_name** 是计算导数的 M 文件函数的字符串名，**to** 是初始时间，**tf** 是终止时间，**yo** 是初始条件向量。可选择的参数 **to1**（缺省值 **to1=1e-3**）是所需的相对精度。在上例中，起始时间是第 0 秒，终止时间是第 30 秒，初始条件为 **y=[1;0]**。两个输出参数是列向量 **t** 和矩阵 y，其中向量 t 包含了估计响应的时间点，而矩阵 y 的列数等于微分方程组的个数（本例为 2），且其行数与 t 相同。t 中的时间点不是等间隔的，因为为了保持所需的相对精度，积分算法改变了步长。

函数 **ode45** 的使用与 **ode23** 完全一样。两个函数的差别在于必须与所用的内部算法相关。两个函数都运用了基本的龙格-库塔(Runge-Kutta)数值积分法的变形。**ode23** 运用一个组合得 2/3 阶龙格-库塔-芬尔格(Runge-Kutta-Fehlerg)算法，而 **ode45** 运用组合的 4/5 阶龙格-库塔-芬尔格算法。一般地，**ode45** 可取较多的时间步。所以，要保持与 **ode23** 相同误差时，

在 to 和 tf 之间可取较少的时间步。然而，在同一时间，**ode23** 每时间步至少调用 **f_name 3** 次，而 **ode45** 每时间步至少调用 **f_name 6** 次。

正如使用高阶多项式内插常常得不到最好的结果一样，**ode45** 也不总是比 **ode23** 好。如果 **ode45** 产生的结果，对作图间隔太大，则必须在更细的时间区间，对数据进行内插，比如用函数 **interp1**。这个附加时间点会使 **ode23** 更有效。作为一条普遍规则，在所计算的导数中，如有重复的不连续点，为保持精度致使高阶算法减少时间步长，这时低阶算法更有效。正是由于这个原因，电子电路分析按缺省，就用一阶算法编程，并且最多提供二阶算法来解决暂态时间响应问题。此外，通过对 **tol** 设置更小的值，要达到更高的精度，没有必要使绝对误差更小。**tol** 设置每时间步的相对精度，不一定引起绝对误差减少。

总之，**不要盲目使用数值方法**。对于给定的问题，在决定最好的方法之前，要试一试各种可能的方法。有关微分方程数值解法的更进一步信息，请参考数值分析方面的书籍。有些参考书还提供了一些关于算法选择和如何处理那些时间常数变化范围大的病态方程的非常实用的信息。

13.7 M 文件举例

这里所介绍的《精通 MATLAB 工具箱》中的 M 文件可近似求解由采样值给出的函数的积分和微分。这里假定这些函数本身不存在，且独立变量也许不是线性间隔。例如，已装载到 MATLAB 中要分析的数据来源于实验测试。

对于所包含的数据缺乏函数描述，有许多种积分和微分的方法。如前所述，人们可以用最小二乘多项式拟合数据，然后在多项式的描述上进行操作。另一种方法是寻找数据的三次样条表示，然后运用《精通 MATLAB 工具箱》中的函数 **spintgrl** 和 **spderiv** 来分别寻找积分和微分的样条表示。这里所介绍的方法提供了另一种更简单的方法。积分用梯形规则计算。用加权中心差分计算微分。此外，将函数设计成在矩阵形式下工作，矩阵的列代表各与自变量有关的因变量。

正如这章前面所述，MATLAB 函数 **trapz** 计算在某有限区间的梯形积分。这里我们寻找的积分是自变量为 x 的函数。即如果 y=f(x)，我们寻找：

$$S(x) = \int_{x1}^{x} f(x)dx$$

式中的 x1 是向量 x 的第一个元素。用梯形规则，这个积分近似为：

$$S(x_k) = \sum_{i=1}^{k} 0.5(y_i + y_{i-1})(x_i - x_{i-1}) \quad \text{且 } S(x_1)=0$$

这样，第 k 个数据点的积分是上述梯形面积的累加和。函数 **mmintgrl** 实现的这个算法如下：

```
function z=mmintgrl(x , y)
%    MMINTgrl Compute Integral using Trapezoidal Rule.
```

```
%     MMINTGRL(X , Y) computes the integral of the function y=f(x) given the
%     data in X and Y. X must be a vector ,    but Y may be a column oriented
%     data matrix. The length of X must equal the length of Y if Y is a
%     vector ,    or it must equal the number of rows in Y if Y is a matrix.
%
%     X need not be equally spaced. The trapezoidal algorithm is used.
%
%     See also mmderiv

%     Copyrigth (c) 1996 by Prentice-Hall , Inc.

flag=0;                    %     falg is True if y is a row
x=x( : ); nx=length(x);    %     make x a column
[ry , cy]=size(y);
if ry==1&cy==nx , y=y .' ; ry=cy ; cy=1 ; flag=1 ; end
if nx~=ry ,    error(' X and Y not the right size ') , end
dx=x(2 : nx)-x(1 : nx-1);        %     width of each trapezoid
dx=dx( : , ones(1 , cy));        %     duplicate for each column in y
yave=(y(2 : ry , : )+y(1 : ry-1 , :))/2;                %     average of heights
z=[zeros(1 , cy); cumsum(dx .* yave)];        %     Use cumsum to find area
if flag , z=z'; end                                     %     if y was a row , return a row
```

在介绍上述函数的使用之前，考虑微分。在这种情况下，人们感兴趣的就是刚给定数据点的近似斜率。这里介绍一种下述的中心差分，而不是简单的向前或向后差分：

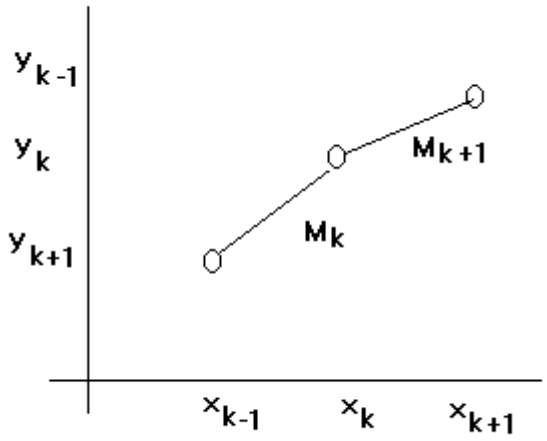


图 13.10 加权中心差分方法

从图 13.11 可知，在第 k 个点的近似微分是：

$$D(x_k) = (1-\alpha_k)M_k + (\alpha_k)M_{k+1} \qquad \text{式中} \alpha_k = \frac{x_k - x_{k-1}}{x_{k+1} - x_{k-1}},$$

并且 M_k 是连接 y_{k-1} 到 y_k 的直线的斜率。这样，第 k 点的微分是相邻两点间斜率的加权平均，离该点越近的点权越重。在第一个和最后一个数据点上，不能简单按照上述方法进行处理，因为这两个点都没有伴随的直线段。对于这些数据点，需要用另外的方法。这里所采取的方法是用二次多项式拟合前 3 个点（或最后 3 个点），并且计算这个多项式第一个（或最后一个）点的微分。函数 **mmderiv** 实现的这个算法如下：

```
function z=mmderiv(x , y)
%    MMDERIV Compute Derivative Using Weighted Central Differences.
%      MMDERIV(X , Y) computes the derivative of the function y=f(x) given the
%      data in X and Y. X must be a vector , but Y may be a column oriented
%      data matrix. The length of X must equal the length of Y if Y is a
%      vector ,    or it must equal the number of rows in Y if Y is a matrix.
%
%      X need not be equally spaced.Weighted central difference are used.
%      Quadratic approximation is used at the endpoints.
%
%      See also mmintgrl

%      Copyrigth (c) 1996 by Prentice-Hall , Inc.

flag=0;                      %      flag is True if y is a row
x=x( : ); nx=length(x);      %      make x a column
[ry , cy]=size(y);
if ry==1&cy==nx , y=y .'; ry=cy; cy=1; flag=1; end
if nx~=ry ,    error(' X and Y not the right size ') , end
if nx<3 ,    error(' X and Y must have st least three elements ') , end

dx=x(2 : nx)-x(1 : nx-1);              %      first difference in x
dx=dx+(dx==0)*eps;                     %      make infinite slopes finite
dxx=x(3 : nx)-x(1 : nx-2);             %      second difference in x
dxx=dxx+(dxx==0)*eps;          %      make infinite slopes finite
alpha=dx(1 : nx-2) ./ dxx                    %      central difference weight
alpha=alpha( : , ones(1 , cy));        %      duplicate for each column in y

dy=y(2 : ry , :)-y(1 : ry-1 , : );             %      first difference in y
dx=dx( : , ones(1 , cy));              %      duplicate dx for each column in y

%      now apply weighting to dy
z=alpha .* dy(2:ry-1 , :) ./ dx(2 : nx-1 , : )+(1-alpha) .* dy(1 : ry-2 , : ) ./ dx(1 : nx-2 , : );
```

```
z1=zeros(1 , cy)>=z1;
for i=1 : cy        %      fit quadratic at endpoints of each column
    p1=polyfit(x(1 : 3) , y(1 : 3 , i) , 2);                %      quadratic at first point
    z1(i)=2*p1(1)*x(1)+p1(2);                    %      evalute poly derivative
    pn=polyfit(x(nx-2 : nx) , y(ry-2 : ry , i) , 2);       %      quadratic at last point
    zn(i)=2*pn(1)*x(nx)+pn(2);                   %      evaluate poly derivative
end
z=[z1;    z;    zn];
if flag , z=z'; end                    %      if y was a row , return a row
```

最后，给出一个例子：

```
>>x=linspace(0 , 2*pi , 30)
>>y=sin(x);        %    create data
>>yi=mmintgrl(x , y);    %    find integral
>>yd=mmderiv(x , y);    %    find derivative
>>plot(x , y , x , yi , ' - ' , x , yd , ' : ')      %    plot results
```

注意这个积分定性地证明了等式：

$$\int_0^x \sin(x)dx = 1 - \cos(x)$$

而微分定性地证明了等式：

$$\frac{d}{dx}\sin(x) = \cos(x)$$

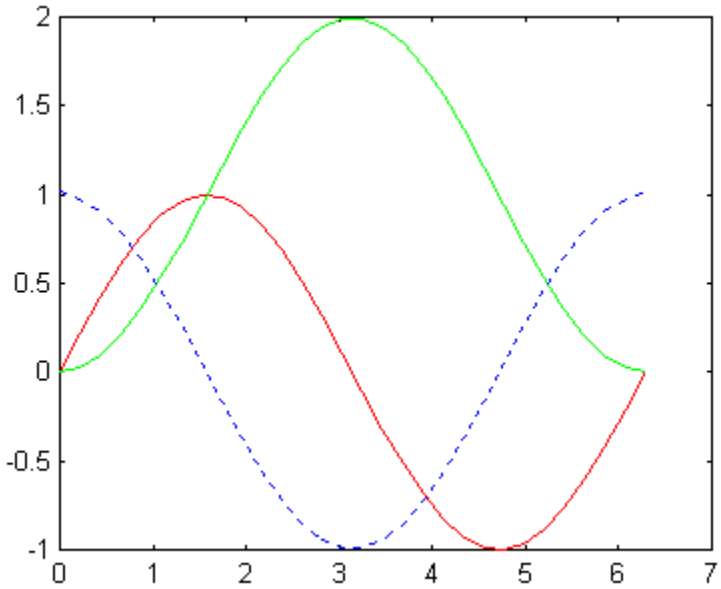


图 13.11 y=sin(x)极其积分、微分曲线

13.8 小结

表 13.1 总结了本章所讨论的函数。

表 13.1

数值分析函数	
fplot(‘ fname ‘ , [lb ub])	绘出上下限之间的函数
fmin(‘ fname ‘ , [lb ub])	寻找上下限内的标量最小值
fimis(‘ fname ‘ , xo)	寻找 **xo** 附近的向量最小值
fzero(‘ fname ‘ , xo)	寻找 **xo** 附近的标量函数的零点
trapz(x , y)	给定数据点 **x** 和 **y**，计算 y=f(x)下的梯形面积积分。
diff(x)	数组元素间的差分
[t , y]=ode23(‘ fname ‘ , to , tf , yo)	用 2 阶/3 阶龙格-库塔算法解微分方程组
[t , y]=ode45(‘ fname ‘ , to , tf , yo)	用 4 阶/5 阶龙格-库塔算法解微分方程组

第 14 章　富里哀分析

象富里哀级数，富里哀变换以及它们离散时间相应部分构成了信号处理的基础。为了便于这类问题的分析，MATLAB 提供了函数 **fft,ifft,fft2,ifft2** 和 **fftshift**。这类函数集执行一维和二维离散富里哀变换及其逆变换。这些函数允许人们完成很多信号处理任务。除此之外，还可在可选的信号处理工具箱中得到其他扩展的信号处理工具。

因为信号处理包含如此广泛的领域，甚至要说明用 MATLAB 中离散富里哀变换函数可解决的这类小问题，就超出了本书的范围。因此，这里将只介绍用函数 **fft** 近似连续时间信号的富里哀变换的一个例子。此外，还将讨论《精通 MATLAB 工具箱》中处理富里哀级数的函数集。

14.1 快速富里哀变换

在 MATLAB 中，函数 **fft** 计算一个信号的离散富里哀变换。在数据的长度是 2 的幂次或质因数的乘积的情况下，就用快速富里哀变换（FFT)来计算离散富里哀变换。当数据长度是 2 的幂次时，计算速度显著增加，因此，只要可能，选择数据长度为 2 的幂次或者用零来填补数据，使得数据长度等于 2 的幂次显得非常重要。在《MATLAB 参考指南》中可找到有关

该问题的讨论。

MATLAB 中实现的快速富里哀变换，是按照工科教材中常使用的方法。

F(k)=FFT{f(n)}

$$F(k)=\sum_{n=0}^{N-1} f(n)e^{-j2\pi nk/N} \qquad k=0,1,\cdots\cdots,N-1$$

因为 MATLAB 不允许零下标，所以移动了一个下标值。

$$F(k)=\sum_{n=1}^{N} f(n)e^{-j2\pi(n-1)(k-1)/N} \qquad k=1,2，\cdots\cdots,N$$

相应的逆变换为：

$$f(n)=FFT^{-1}\{F(K)\}$$

$$f(n)=\frac{1}{N}\sum_{k=1}^{N} F(k)e^{-j2\pi(n-1)(k-1)/N} \qquad n=1,2,\cdots\cdots,N$$

为了说明 FFT 的使用，考虑估计连续信号的富里哀变换的问题。

$$f(t)=\begin{cases}12e^{-3t} & t\geq 0\\ 0 & t<0\end{cases}$$

解析上，该富里哀变换为：

$$F(w)=\frac{12}{3+jw}$$

虽然在这种情况下，由于知道了富里哀变换的解析结果，再运用 FFT 没有多大的实用价值，但这个例子说明了对不常见的信号，特别是那些解析上难以找到富里哀变换的信号，一个估计富里哀变换的方法。下面的 MATLAB 语句用 FFT 估计 F(w)，并且用图形把所得到结果与上面的解析表达式的结果进行比较：

```
>>N=128;     %    choose a power of 2 for speed
>>t=linspace(0, 3, N);    %    time points for function evaluation
>>f=2*exp(-3*t);    %    evaluate the function and minimize aliasing:f(3)~0
>>Ts=t(2)-t(1);    %    the sampling period
>>Ws=2*pi/Ts;    %    the sampling frequency in rad/sec
>>F=fft(f);    %    compute the fft
>>Fp=F(1 : N/2+1)*Ts;
```

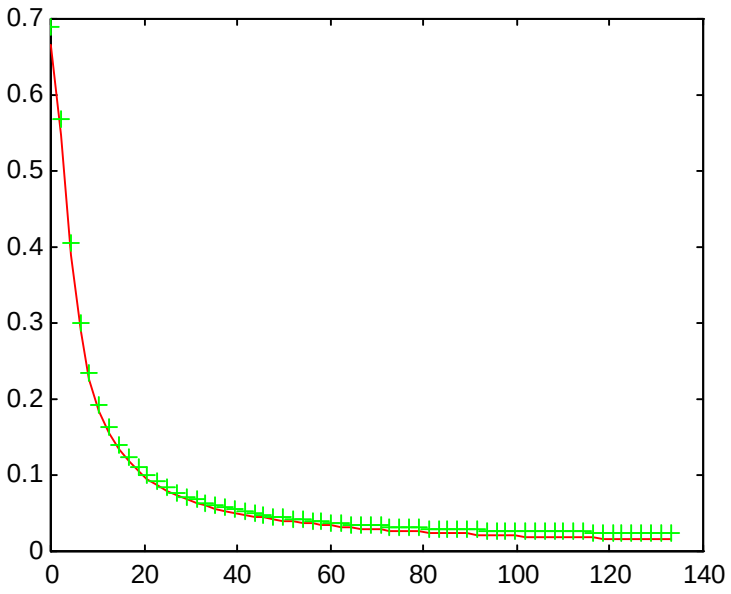


图 14.1 富里哀变换两种结果的比较

仅从 F 中取正频率分量，并且乘以采样间隔计算 F(w)。

```
>>W=Ws*(0 : N/2)/N
```

它建立了连续频率轴，该轴起始于 0，终止于奈魁斯特（Nyquist）频率 Ws/2，

```
>>Fa=2./(3+j*w);    %    evaluate analytical Fourier transform
>>plot(W, abs(Fa), W, abs(Fp), ' + ' )    %    generate plot, ' + ' mark fft results
>>xlabel(' Frequency, Rad/s '),ylabel(' |F(w)| ')
```

MATLAB 提供了大量的完成一般信号处理任务的函数。它们列于表 14.1：

表 14.1

信号处理函数	
conv	卷积
conv2	2 维卷积
fft	快速富里哀变换
fft2	2 维快速富里哀变换
ifft	逆快速富里哀变换
ifft2	2 维逆快速富里哀变换
filter	离散时间滤波器
filter2	2 维离散时间滤波器
abs	幅值
angle	四个象限的相角
unwrap	在 360° 边界清除相角突变
fftshift	把 FFT 结果平移到负频率上

14.2 富里哀级数

MATLAB 本身没有特别关于富里哀级数分析和处理的函数。不过，通过创建 M 文件函数，可容易加上这些函数。在这一节，将介绍《精通 MATLAB 工具箱》中富里哀级数函数。在介绍之前，首先定义实周期信号 f(t)的富里哀级数表示形式。

给出富里哀级数的**复指数形式**为：

$$f(t)=\sum_{n=\infty}^{\infty}F_n e^{jnw_0 t}$$

式中的富里哀级数的系数是：

$$F_n=\frac{1}{T_0}\int_t^{t+T_0}f(t)e^{-jnw_0 t}dt$$

且基频为$\omega_0=2\pi/T_0$。式中 T_0 满足 $f(t+T_0)=f(t)$。

给出富里哀级数的**三角数形式**为：

$$f(t)=A_0+\sum_{n=1}^{\infty}\{A_n\cos(n\omega_0 t)+B_n\sin(n\omega_0 t)\}$$

式中的富里哀级数的系数是：

$$A_0=\frac{1}{T_0}\int_t^{t+T_0}f(t)dt$$

$$A_n=\frac{2}{T_0}\int_t^{t+T_0}f(t)\cos(n\omega_0 t)dt$$

$$B_n=\frac{2}{T_0}\int_t^{t+T_0}f(t)\sin(n\omega_0 t)dt$$

且基频为$\omega_0=2\pi/T_0$。式中 T_0 满足 $f(t+T_0)=f(t)$。

表 14.2

精通 MATLAB 的富里哀级数函数	
fsderiv(Kn,Wo)	富里哀级数的微分
fseval(Kn,t,Wo)	计算富里哀级数
fsfind(‘ fname ‘,T,N)	寻找时间函数的富里哀级数的系数
[An,Bn,Ao]=fsform(Kn)	富里哀级数不同形式之间的转换
Kn=fsform(An,Bn,Ao)	
fsharm(Kn,i)	提取特殊的富里哀级数的谐波
fsmsv(Kn)	计算信号的均方根值
fsresize(Kn,N)	重新调整富里哀级数的系数向量的大小
fsresp(Num,Den,Un,Wo)	线性系统对输入富里哀级数 Un 的富里哀级数响应
fsround(Kn)	设置次要的富里哀级数的系数为 0
fswindow(N, ‘ type ‘)	产生一个窗口的函数，使吉布（Gibb）现象极小
fswindow(Kn, ‘ type ‘)	

上述两种形式中，一般复指数的富里哀级数更容易进行解析上处理。而三角富里哀级数则提供了更多的直观理解，更容易将正弦和余弦波形可视化。由于这个原因，《精通 MATLAB 工具箱》中的富里哀级数函数一般假定为复指数形式。不过，该工具箱提供了这两种富里哀级数形式之间转换的函数。表 14.2 总结了《精通 MATLAB 工具箱》中的富里哀级数函数。

因为有无穷多个谐波或富里哀级数的系数，有必要对富里哀级数截尾，只考虑有限个谐波。如果考虑 N 个谐波，MATLAB 中富里哀级数表示成一个长度为 2N+1 的行向量，向量中的元素为复指数的富里哀级数的系数。该向量包含以升序排列的富里哀级数系数，即：

$$F = [F_{-N}\ F_{-N+1} \cdots F_{-1}\ F_0\ F_1 \cdots F_{N-1}\ F_N]$$

为了说明这些函数的使用，考虑寻找锯齿信号的富里哀级数表示（见图 14.2）。

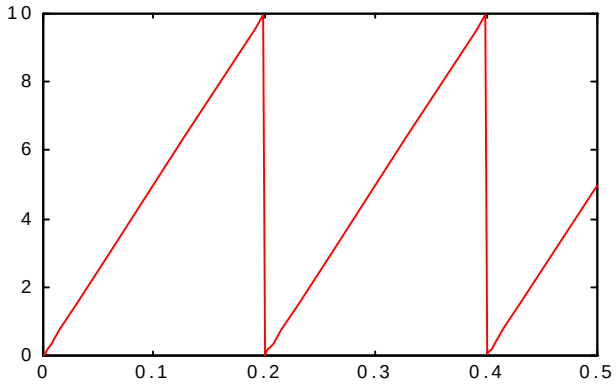


图 14.2 锯齿信号

虽然很容易找到这个信号的富里哀级数的系数，但可用函数 **fsfind** 近似求解这些系数。

```
function [Fn, nwo, f, t]=fsfind(fun, T, N, P)
%    FSFIND Find Fourier Series Approximation.
%     Fn=FSFIND(FUN, T, P) computes the Complex Exponential
```

```
%     Fourier Series of a signal described by the function ' FUN '.
%     FUN is the character string name of a user created M-file function.
%     the unction is called as f=FUN(t) where t is a vector over
%     the range 0<=t<=T.
%
%     The FFT is used. Choose sufficient harmonics to minimize aliasing.
%
%     T is the period of the function. N is the number of harmonics.
%     Fn is the vector of FS coefficients.
%
%     [Fn, nWo, t, p]=FSFIND(FUN, T, P) returns the frequencies associated
%     with Fn in nWo and return values of the function FUN
%     in f evaluated at the points int t over the range 0<=t<=T.
%
%     FSFIND(FUN, T, P) passes the data in P to function FUN as
%     f=FUN(t, p). This allows parameters to be passed to FUN.
%     Copyrigth (c) 1996 by Prentice-Hall,Inc.

n=2*N;
t=linspace(0, T, n+1);
if nargin==3
     f=feval(fun, t);
else
     f=feval(fun, t, P);
end
Fn=fft(f(1 : n));
Fn=[0    conj(Fn(N : -1 : 2))    Fn(1 : N)    0]/n;
nwo=2*pi/T*(-N : N);
```

根据上述描述，必须先建立一个函数以计算周期内锯齿信号值。这样，我们有：

```
function f=sawtooth(t, T)
%    SAWTOOTH Sawtooth Waveform Generation.
%     SAWTOOTH(t, T) computes values of a sawtooth having a
%    period T at the values in t.
%
%    Used in Fourier series examples.
f=10*rem(t, T)/T;
```

现在求近似富里哀级数的系数。

```
>>T= .2;        %    desired period
>>N=25;      %    number of harmonics
>>Fn=fsfind(‘ sawtooth ‘, T, N, T);      %    compute Fourier series coefficients
```

因为用 FFT 求近似系数，所以得到的系数并不精确。不过，如果有足够多的系数，特别是如果时间函数本身，或者其一阶，或者二阶导数连续（锯齿信号不连续），误差就比较小。如果需要更高的精度，可调用积分程序来计算每一个富里哀级数系数的积分关系式。后一种方法需要很长的时间，因为 FFT 一次要近似所有的系数。不过，尽管有误差，对于许多应用，FTT 的近似结果已足够准确了。

用函数 **fseval** 可实现富里哀级数的计算。

```
function y=fseval(kn, t, wo)
%    FSEVAL Fourier Series Function Evaluation.
%      FSEVAL(Kn, t, Wo) computes values of a real valued function given
%      its complex exponential Fourier series coefficients Kn, at the
%      points given in t where the fundamental frequency is Wo rad/s.
%      K contains the Fourier coefficentsin ascending order:
%      Kn = [k        k        ...        k        ...        k        k]
%                    -N       -N+1          0             N-1    N
%      if Wo is not given, Wo=1 ia assumed.
%      Note: this function creates a matrix of size:
%      rows = length(t) and columns = (length(K)-1)/2

%      Copyrigth (c) 1996 by Prentice-Hall,Inc.

if    nargin==2, wo=1; end
nk=length(kn);
if    rem(nk-1, 2) | (nk==1)
      erro(‘ Number of element in K must be odd and greater than 1 ‘)
end
n=0.5*(nk-1); %      highest harmonic
nwo=wo*(1 : n);    %      harmonic frequencies
ko=kn(n+1);   %      average value
kn=kn(n+2 : nk) ‘; %      positive frequency coefs
y=ko+2*(real(exp(j*t( : )*nwo)*kn))' ;
```

遵照所需的语法，我们得到：

```
>>n=-N : N;    %    harmonic index
>>Fna=j*5 ./ (pi*n);    %    actual Fourier series coefficients
>>Fna(N+1)=5;      %    poke in average value
>>t=linspace(0 ,    .4);    %    time points to evaluate functions
>>wo=2*pi/T;      %    fundamental frequency
>>f=fseval(Fn, t, wo);    %    evaluate approximated Fourier series
>>fa=fseval(Fna, t, wo);    %    evaluate actual Fourier series
>>plot(t, f, t, fa)    %    plot results for comparison
```

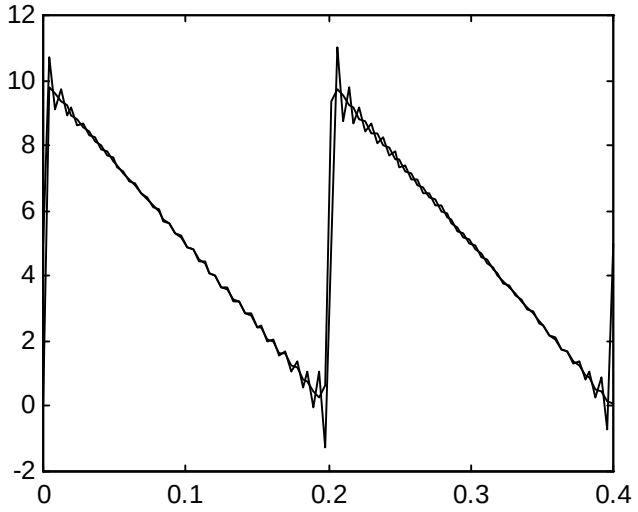


图 14.3 富里哀级数的结果比较

注意上述两个富里哀级数给出了相似的结果。从图 14.3 中可知，在锯齿信号的不连续点的周围出现了明显的吉布现象波纹。用函数 **fswindow**，将一个**窗口**加于富里哀级数系数，就能减少这种波纹。

```
>>help fswindow

FSWINDOW Generate Window Functions.
  FSWINDOW(N, TYPE) creates a window vector of type TYPE having
  a length equal to the scale N.
  FSWINDOW(X, TYPE) creates a window vector of type TYPE having
  a length and orientation the same as the vector X.
  FSWINDOW(X, TYPE, alpha) porvides a parameter alpha as required
  for some window types.
  FSWINDOW with no input arguments returns a string matrix whose
  i-th row is the i-th TYPE given below.
  TYPE is a string designating the window type desired:
  'rec' = Rectangular or Boxcar
  'tri' = Triangular or Bartlet
  'han' = Hann or hanning
  'ham' = Hamming
  'bla' = blackman common coefs.
  'blx' = Blackman exact coefs.
  'rie' = Rieman {sin(x)/x}
  'tuk' = Tukey, 0<alpha<1; alpha = 0.5 is default
  'poi' = Poisson, 0<alpha<inf; alpha = 1 is default
  'cau' = Cauchy, 1<alpha<inf; alpha = 1 is default
  'gau' = Gaussian, 1<alpha<inf; alpha = 1 is default
Reference: F. J. Harris,"On the Use of Windows for Harmonic Analysis with the Discrete
```

Fourier Transform," IEEE Proc. Vol. 66, no. 1, Jan. 1978, pp.51-83

```
>>Fnh=Fn.*fswindow(Fn, ' han ');     %    apply Hanning window
>>f=fseval(Fnh, t, wo);     %    evaluate windowed FS
>>plot(t, f)     %    plot results
```

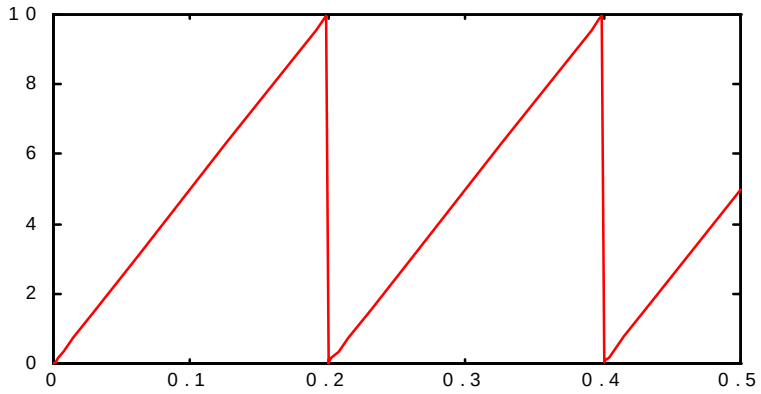


图 14.4　用汉宁窗口滤波后的富里哀级数

因为乘以窗口函数，改变了富里哀级数的系数，所以，现在图形（见图 14.4）看起来好多了。

接下来，让我们把锯齿波加到传递函数为 $H(s)=\frac{60}{s+60}$ 的线性系统中。

函数 **fsresp** 专门解决此类问题。

```
function y=fsresp(num, den, un, wo)
%    FSRESP Fourier Series Linear System Response
%    FSRESP(N, D, Un, Wo) returns the comples exponential FS of the
%    output of a linear system when the input is given by a FS
%    N and D are the numerator and denominator coefficients
%    respectively of the system transfer function.
%    Un is the complex exponential Fourier Series of the system input.
%    Wo is the fundamental frequency associated with the input.
%    Copyright (c) 1996 by Prentice-Hall, Inc.
if nargin<4, wo=1; end
N=(length(un)-1)/2;        %      highest harmonic
jnWo=sqrt(-1)*(-N : N)*wo; %      frequencies of harmonics
y=(polyval(num, jnWo) ./ polyval(den, jnWo)).*un;    %      output is Yn=H(jnWo)*Un
```

运用 **fsresp**，我们得到：

```
>>Yn=fsresp(60, [1    60], Fn, wo);    %    find response coefficients
>>y=fseval(Yn, t, wo);    %    evaluate output
>>plot(t, f, f, y)    %    plot system input and output
```

结果见图 14.5。

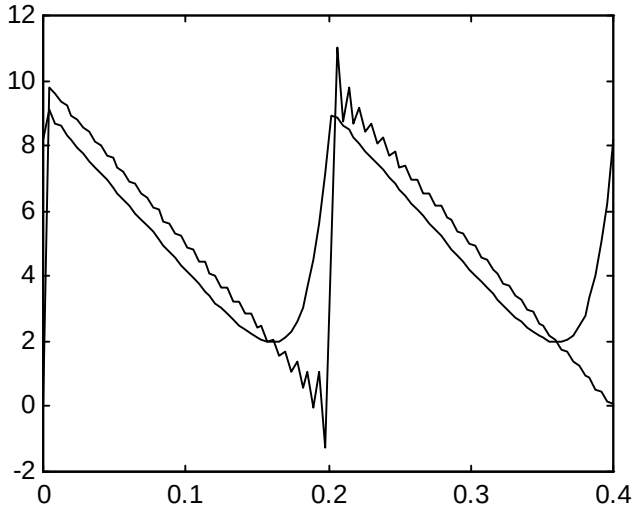


图 14.5　线性系统的输入输出曲线

作为最后一个例子，运用 **fsform** 把锯齿信号的富里哀级数转换为三角形式。

```
function [a, b, ao]=fsform(c, d, co)
%    FSFORM Fourier Series Format Conversion
%      KN=FSFORM(An, Bn, Ao) converts the trigonometric FS with
%      An being the CSOINE and Bn being the SINE coefficients to
%      the complex exponential FS with coefficients Kn.
%      Ao is the DC component and An,Bn and Ao are assumed to be real.
%
%      [Kni]=FSFORM(An, Bn, Ao) returns the index vector i that
%      identifies the harmonic number of each element of Kn.
%
%      [An, Bn, Ao]=FSFORM(Kn) does the reverse format conversion.

%      Copyrigth (c) 1996 by Prentice-Hall,Inc.

nc=length(c);
if nargin==1 %      complex exp -> trig form
     if rem(nc-1, 2)|(nc==1)
      error(' Number of elements in K must bo odd and greater than 1 ')
     end
     nn=(nc+3)/2;
     a=2*real(c(nn : nc));
     b=-2*imag(c(nn : nc));
     ao=real(c(nn-1));
elseif    nargin==3      %    trig from -> complex exp form
```

```
    nd=length(d);
    if nc~=nd;
        error(' A and B must be the same length ')
    end
    a=0.5*(c-sqrt(-1)*d);
    a=[conj(a(nc : -1 : 1)) co(1) a];
    b=-nc : nc;
else
    error(' Improper number of input arguments. ')
end

>>[An, Bn, Ao]=fsform(Fn)
An =
  Columns 1 through 7
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 8 through 14
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 15 through 21
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
  Columns 22 through 25
   -0.2000   -0.2000   -0.2000         0
Bn =
  Columns 1 through 7
   -3.1789   -1.5832   -1.0484   -0.7789   -0.6155   -0.5051   -0.4250
  Columns 8 through 14
   -0.3638   -0.3151   -0.2753   -0.2418   -0.2130   -0.1878   -0.1655
  Columns 15 through 21
   -0.1453   -0.1269   -0.1100   -0.0941   -0.0792   -0.0650   -0.0514
  Columns 22 through 25
   -0.0382   -0.0253   -0.0126         0
Ao =
    4.9000
```

忽略平均值，这个锯齿应展示了奇对称特性，所有 An 系数应当为 0。显然，它们不为 0。发生这种情况的原因是因为我们使用了 FFT 来近似系数。此外，Ao 平均值有一点儿误差，它的值应当是 5.0。

14.3 小结

下面的两个表总结了本章所讨论的函数。

表 14.3

信号处理函数	
conv	卷积
conv2	2 维卷积
fft	快速富里哀变换
fft2	2 维快速富里哀变换
ifft	逆快速富里哀变换
ifft2	2 维逆快速富里哀变换
filter	离散时间滤波器
filter2	2 维离散时间滤波器
abs	幅值
angle	四个象限的相角
unwrap	在 360°边界清除相角突变
fftshift	把 FFT 结果平移到负频率上
nextpow2	2 的下一个较高幂次

表 14.4

精通 MATLAB 的富里哀级数函数	
fsderiv(Kn,Wo)	富里哀级数的微分
fseval(Kn,t,Wo)	计算富里哀级数
fsfind(‘ fname ‘,T,N)	寻找时间函数的富里哀级数的系数
[An,Bn,Ao]=fsform(Kn) Kn=fsform(An,Bn,Ao)	富里哀级数不同形式之间的转换
fsharm(Kn,i)	提取特殊的富里哀级数的谐波
fsmsv(Kn)	计算信号的均方根值
fsresize(Kn,N)	重新调整富里哀级数的系数向量的大小
fsresp(Num,Den,Un,Wo)	线性系统对输入富里哀级数 Un 的富里哀级数响应
fsround(Kn)	设置次要的富里哀级数的系数为 0
fswindow(N, ‘ type ‘) fswindow(Kn, ‘ type ‘)	产生一个窗口的函数，使吉布（Gibb）现象极小

第 15 章　低级文件 I/O

对于大多数用户，MATLAB 函数 **load** 和 **save** 为装载和存储数据提供了足够的工具。利用以扩展名为**.mat** 结尾的文件名，**load** 和 **save** 假定数据是以与平台无关的二进制格式保存，或者用称之为 flat 的简单的 ASCII 文件格式保存。当 flat ASCII 或**.mat** 这两种格式还不够时，MATLAB 提供了基于 C 语言实现的低级文件 I/O 函数。用这些低级文件 I/O 函数，MATLAB 可以读写你所知道的任意文件格式。例如，知道电子文件或数据库程序所使用的格式，就可以把这些数据读进 MATLAB 的矩阵中去。类似地，也可以创建电子文件或数据库文件。

MATLAT 中这种基本的低级文件 I/O 命令如下：

表 15.1

MATLAB 低级文件 I/O 函数	
fclose：	关闭文件
feof：	测试文件结束
ferror：	查询文件 I/O 的错误状态
fgetl：	读文件的行，忽略回行符
fgets：	读文件的行，包括回行符
fpoen：	打开文件
fprintf：	把格式化数据写到文件或屏幕上
fread：	从文件中读二进制数据
frewind：	返回到文件开始
fscanf：	从文件中读格式化数据
fseek：	设置文件位置指示符
ftell：	获取文件位置指示符
fwrite：	把二进制数据写到文件里

除了这些函数外，所具有的 MATLAB 版本可能为一个或多个公用软件包提供读写文件的特定函数 M 文件。有关这些函数的进一步的信息，请使用在线帮助：**>>help iofun**

第 16 章　调试工具

在开发函数 M 文件过程中，不可避免地出现错误，即故障。MATLAB 提供了很多函数和方法，帮助调试函数。

在 MATLAB 表达式中，有两类错误：语法错误和运行错误。当 MATLAB 计算一个表达式的值或一个函数被编译到内存时会发现语法错误。一旦发现语法错误，MATLAB 立即标志这些错误，并提供有关所遇到的错误类型，以及发生错误处 M 文件的行数。给定这些反馈信息，就很容易纠正这些错误。

而另一方面，即使 MATLAB 标志了运行错误，但找出错误一般比较困难。当发现运行错误时，MATLAB 把控制权返回给命令窗口和 MATLAB 的工作空间。失去了对发生错误的函数空间的访问权，因此，用户不能询问函数工作空间中的内容排除问题。

根据作者的经验，当一些操作结果导致空矩阵或 **NaNs** 时，最容易发生运行错误。所有有关 **NaNs** 的操作都返回 **NaNs** 值。因此，如果有可能出现 **NaNs** 结果，则当出现 **NaNs** 时，最好运用逻辑函数 **isnan** 来执行一些缺省操作。因为空矩阵为零维，所以对空矩阵寻址常常导致错误。函数 **find** 表示了可产生空矩阵结果的一般情况。如果函数 **find** 的空矩阵输出用于索引其它数组，所返回的值也将是空的。这样，空矩阵具有传播性质。例如：

```
>>x=pi*(1 : 4)       %    example data

>>i=find(x>20)    %    use find function

>>y=2*x(i)          %    propagate the empty matrix
```

很清楚，当希望 y 具有有限维数和值时，可能发生运行错误。当执行一个操作或使用可返回空结果的函数时，逻辑函数 **isempty** 有利于为空矩阵定义一个缺省值，这样避免运行错误。

有几种调试函数 M 文件的方法。对于简单的问题，可直接使用下列的方法组合：

1、去掉文件中所选择的行的分号，以便中间结果显示在命令窗口中。

2、在文件中加入显示感兴趣的变量的语句

3、把 **keyboard** 命令放在文件中所选择的地方，给键盘暂时控制权。这样，可以查询函数空间并按需要改变其值。

4、在 M 文件开始，在 function 语句前加上%，将函数 M 文件改变为脚本 M 文件。当 MATLAB 执行该脚本 M 文件时，该空间就是 MATLAB 工作空间。这样，当发生错误时可以询问。

当 M 文件大，递归调用或者多次嵌套（即调用其它 M 文件函数，被调用 M 文件函数又调用其它 M 文件函数，等等）时，用 MATLAB 的调试函数会更方便。与上述所列方法相反，这些调试函数不要求将有问题的 M 文件进行编辑。表 16.1 所给出的这些函数类似于其它高级编程语言中所提供的函数。有关进一步的信息，以及它们的使用实例，参阅《MATLAB 用户指南》。

表 16.1

MATLAB 调试函数	
dbclear：	取消断点
dbcont：	在断点后恢复运行
dbdown：	工作空间下移
dbquit：	退出调试模式
dbstack：	列出谁调用谁
dbstatus：	列出所用的断点
dbstep：	执行一行或多行
dbstop：	设置断点
dbtype：	列出带行号的 M 文件
dbup：	工作空间上移

第十八章　三维图形

为了显示三维图形，MATLAB 提供了各种各样的函数。有一些函数可在三维空间中画线，而另一些可以画曲面与线格框架。另外，颜色可以用来代表第四维。当颜色以这种方式使用时，由于它不再象照片中那样显示信息的自然属性----色彩，而且也不是基本数据的内在属性，所以它称作**伪彩色**。为了简化对三维图形的讨论，对颜色的介绍推迟到下一章。在这一章，主要讨论绘制三维图形的基本概念。

18.1 函数 plot3

plot3 命令将绘制二维图形的函数 **plot** 的特性扩展到三维空间。函数格式除了包括第三维的信息（比如 z 方向）之外，与二维函数 **plot** 相同。**plot3** 一般语法调用格式是 **plot3($x_1,y_1,z_1,S_1,x_2,y_2,z_2,S_2,\cdots$)**，这里 x_n,y_n 和 z_n 是向量或矩阵，S_n 是可选的字符串，用来指定颜色、标记符号和/或线形。

总的来说，**plot3** 可用来画一个单变量的三维函数。如下为一个三维螺旋线例子：

```
» t=0:pi/50:10*pi;
» plot3(sin(t),cos(t),t)
» title( ‘ Helix ‘ ),xlabel( ‘ sint(t) ‘ ),ylabel( ‘ cos(t) ‘ ),zlabel( ‘ t ‘ )
» text(0,0,0, ‘ Origin ‘ )
» grid
» v = axis
v =
     -1      1     -1      1      0     40
```

输出见图 18.1.

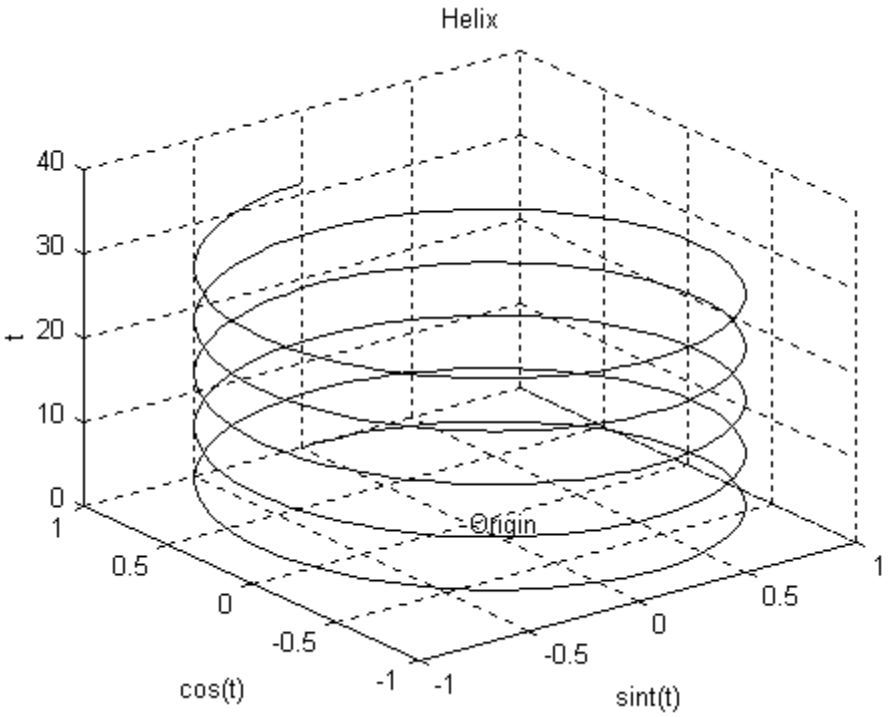


图 18.1　螺旋线图

从上例可明显看出，二维图形的所有基本特性在三维中仍都存在。**axis** 命令扩展到三维只是返回 Z 轴界限（0 和 40），在数轴向量中增加两个元素。函数 **zlabel** 用来指定 z 轴的数据名称，函数 **grid** 在图底绘制三维网格。函数 **test(x,y,z,** ‘ **string** ‘ **)**在由三维坐标 **x,y,z** 所指定的位置放一个字符串。另外，子图和多图形窗口可以直接应用到三维图形中。

在最后一章可以看到，通过指定 **plot** 命令的多个参量或使用 **hold** 命令，可以把多条直线或曲线重叠画出。**plot3** 以及其它的三维图形函数都可以提供相同的能力。例如，增加维数的 **plot3** 命令可以使多个二维图形沿一个轴排列起来，而不是直接将二维图形叠到另一个的上面。

```
» x=linspace(0,3*pi);    %    x-axis data
» z1=sin(x);    %    plot in x-z plane
» z2=sin(2*x);
» z3=sin(3*x);
» y1=zeros(size(x));    %    spread out along y-axis
» y3=zeros(size(x));    %    by giving each diffent y-axis values
» y2=y3/2;
» plot3(x,y1,z1,x,y2,z2,x,y3,z3);
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ,abel( ‘ z-axis ‘ )
» title( ‘ sin(x),sin(2x),sin(3x) ‘ )
```

输出见图 18.2.

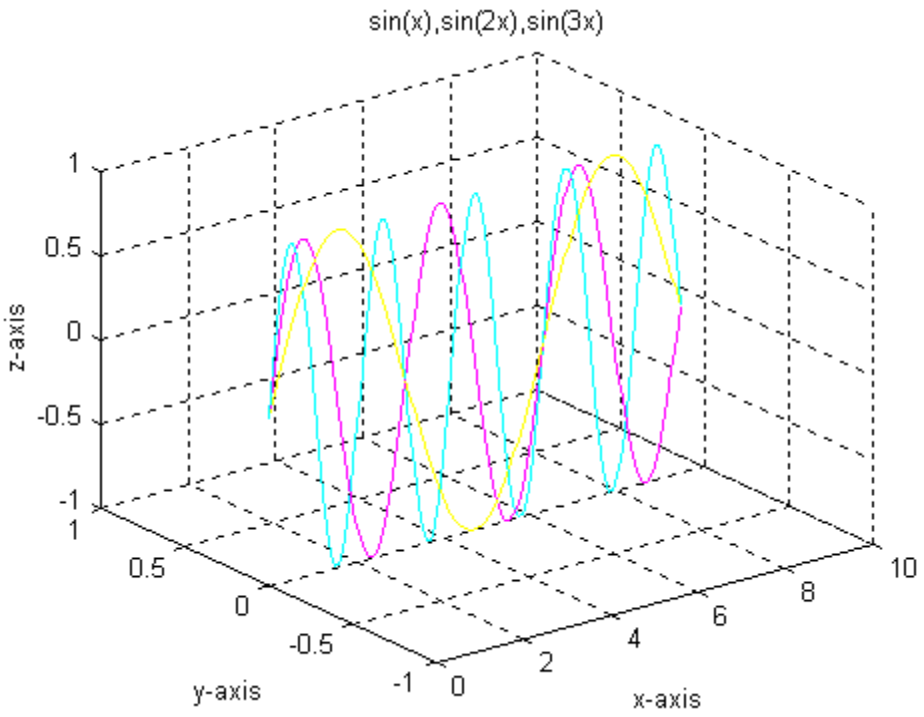


图 18.2　正弦曲线图

上述图形也可以沿另外方向堆列。

```
» plot3(x,z1,y1,x,z2,y2,x,z3,y3)
» grid,xlabel( ‘ x-axis ‘ ,ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ sin(x),sin(2x),sin(3x) ‘ )
```

输出见图 18.3.

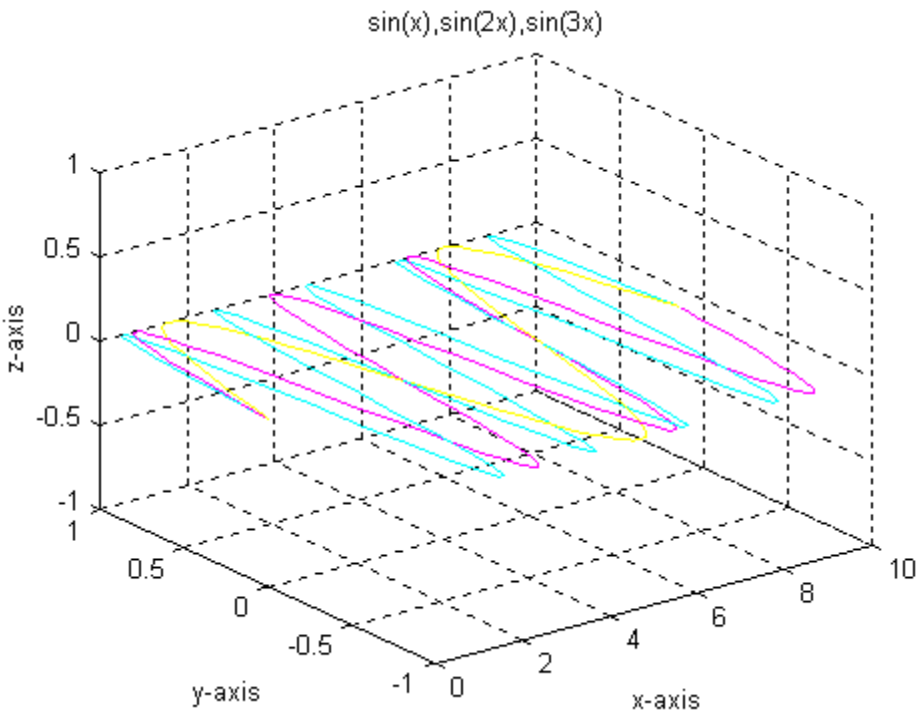


图 18.3　正弦曲线图

18. 2 改变视角

注意两个图形，一个是以 30 度视角向下看 z=0 平面,一个是以 37.5 度视角向上看 x=0 平面。这是对所有三维图形的缺省视角。与 z=0 平面所成的方向角叫**仰角**，与 x=0 平面的夹角叫做**方位角**。这样，缺省的三维视角方向仰角为 30 度，方位角为-37.5 度。而缺省的二维视角仰角为 90 度，方位角为 0 度。仰角和方位角的概念在图 18.4 中形象地画出。

图 18.4　仰角和方位角示意图

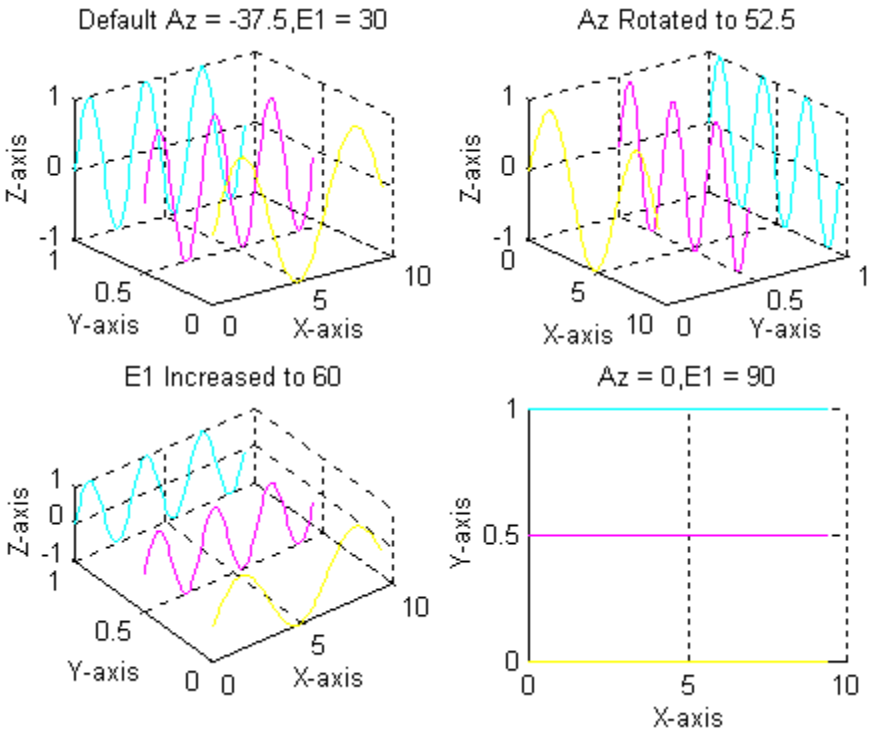


图 18.5　视角举例图

在 MATLAB 中，函数 **view** 改变所有类型的二维和三维图形的图形视角。**view(az,el)**和**view([az,el)]**将视角改变到所指定的方位角 **az** 和仰角 **el**。考虑下面脚本 M 文件形式的例子。

```
%    viewpoint example using subplots

x=linspace(0,3*pi).';

Z=[sin(x)    sin(2*x)    sin(2*x)];    %    create Y and Z axes as matrices

Y=[zeros(size(x))    ones(size(x))/2    ones(size(x))];

subplot(2,2,1)
```

```
plot3(x,Y,Z)    %    plot3 works with column-oriented matrices too
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ Default Az = -37.5,E1 = 30 ‘ )
view(-37.5,30)

subplot(2,2,2)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ Az Rotated to 52.5 ‘ )
view(-37.5+90,30)

subplot(2,2,3)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
title( ‘ E1 Increased to 60 ‘ )
view(-37.5,60)

subplot(2,2,4)
plot3(x,Y,Z)
grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
title( ‘ Az = 0,E1 = 90 ‘ )
view(0,90)
```

输出见图 18.5。

除了上面的形式，**view** 还提供了综合在表 18.1 的其它特性：

表 18.1

函数 view	
view(az,el)	将视图设定为方位角 **az** 和仰角 **el**
view([az,el])	
view([x,y,z])	在笛卡儿坐标系中将视图设为沿向量**[x,y,z]**指向原点，例如 **view([0 0 1])=view(0,90)**
view(2)	设置缺省的二维视角，**az=0**，**el=90**
view(3)	设置缺省的三维视角，**az=-37.5**，**el=30**
[az,el]=view	返回当前的方位角 **az** 和仰角 **el**
view(T)	用一个 4×4 的转矩阵 **T** 来设置视图角
T=view	返回当前的 4×4 转矩阵

最后，为了演示 MATLAB 句柄图形能力，精通 *MATLAB* 工具箱包含了函数 **mmview3d**。在产生二维或三维图形后调用此函数，**» mmview3d**，在当前图形中放置水平角和方位角滑标（滚动条）以设置视角。使用函数 **mmview3d** 的更详细的信息见在线帮助。

18.3 两个变量的标量函数

相对于 **plot3** 产生的线条图形，经常希望画出两个变量的标量函数，比如：

z=f(x,y)

这里每一对 x 与 y 的值产生一个 z 的值。它作为 x 与 y 的函数，是三维空间中的一个曲面。为了在 MATLAB 里画出这个曲面，z 的值存放在一个矩阵中。象在**二维插值这一节所描述的那样，给出 **x** 与 **y** 的值作为独立的变量，**z** 是因变量矩阵，**x**、**y** 与 **z** 的联系就是：

z(i,:)=f(x,y(i)) and z(:,j)=f(x(j),y)

即 **z** 的第 i 行与的 **y** 第 i 个元素相关，而 **z** 的第 j 列与 **x** 的第个 j 元素相关。或者说，**y** 沿着 z 的列变化，而 **x** 沿着 **z** 的行变化。

当 z=f(x,y)能简化表示时，可以方便地用数组运算在单个语句中算出 z 的所有的值。这样做，要求我们以合适的方向创建所有 x 与 y 值的矩阵。（这种方向有时被 *Mathwork* 公司称作为**方格**)。MATLAB 提供了函数 **meshgrid** 来执行这个步骤。

```
» x=-3:3;   %   choose x-axisd values
» y=1:5;   %   y-axis values
» [X,Y]=meshgrid(x,y)
X =
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
     -3     -2     -1      0      1      2      3
Y =
      1      1      1      1      1      1      1
      2      2      2      2      2      2      2
      3      3      3      3      3      3      3
      4      4      4      4      4      4      4
      5      5      5      5      5      5      5
```

如上所见，函数 **meshgrid** 对 **y** 中行的每一行复制 **x**，同样也对 **x** 中列的每一列复制 **y**。这种方向与前面语句相一致，即 **y** 向下改变其列，而 **x** 横跨改变其行。给定 **X** 和 **Y**，如果 $z=f(x,y)=(x+y)^2$，那么 z 便定义一个三维曲面的数据矩阵：

```
» Z=(X+Y).^2
Z =
      4      1      0      1      4      9     16
      1      0      1      4      9     16     25
      0      1      4      9     16     25     36
```

```
 1     4     9    16    25    36    49
 4     9    16    25    36    49    64
```

当函数不能象上面那样简单表示出来时，便只能用 For 循环或 While 循环来计算 **Z** 的元素。在很多情况下，有可能按行或按列计算 **Z**。例如，如果能按行计算 **Z**，下面的脚本文件段会很有帮助：

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for r=1:nx
    {preliminary commands}
    Z(r,:)={a function of y and x(r) defining r-th row of Z}
end
```

相反，如果能按列计算 **Z**，下面的脚本文件段会很有帮助：

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for c=1:ny
    {preliminary commands}
    Z(:,c)={a function of y(c) and x defining c-th column of Z}
end
```

只有当 **Z** 中的元素必须一个一个地计算时，就常常要求象下面的脚本文件一样,用嵌套循环进行计算。

```
x=???   %   statement defining vector of x-axis values
y=???   %   statement defining vector of y-axis values
nx=length(x);   %   length of x is no. of rows in Z
ny=length(y);   %   length of y is no. of columns   in Z

Z=zeros(nx,ny);   %   initialize Z matrix for speed

for r=1:nx
```

```
    for c=1:ny
        {preliminary commands}
        Z(r,c)={a function of y(c) and x(r) defining Z(r,c)}
    end
end
```

18.4 杂乱或散射数据的插值

在有些情况下，两个变量的标量函数的值，如 z=f(x,y)，不能简单地算出。这是因为要么 x 和 y 的值是非均匀间隔的（最坏时是随机分布），要么是用了不同的坐标系，比如非长方形的网格。出现这些情况时，MATLAB 中的函数 **griddata** 就用来产生经插值后的均匀间隔数据以作图。首先考虑前面的例子。假设要求较高的分辨率，但我们不想重新计算函数来得到新值。

```
» x=-3:3;   %   original x-axis values
» y=1:5;   %   original y-axis values
» [X,Y]=meshgrid(x,y);   %   create plaid data matrices
» Z=(X+Y).^2;   %   original z values
» size(Z)   %   original array size
ans =
     5     7
» xi=-3:.5:4;   %   interpolated x-axis values
» length(xi)   %   get new x-axis length
ans =
    15
» yi=0:.2:5;   %   interpolated y-axis values
» length(yi)   %   get new y-axis length
ans =
    26
» [Xi,Yi]=meshgrid(xi,yi);   %   make new data plaid
» Zi=griddata(X,Y,Z,Xi,Yi);   %   interpolated Z data using original data
» size(Zi)   %   interpolated size is correct
ans =
    26    15
```

这里函数 **griddata** 用三个原始矩阵 **X，Y，Z** 和需要插值的方格矩阵，创建一个新的因变量矩阵 **Zi**。要注意插值不必在原始数据的范围内，比如 **x** 在-3 与 3 间变化，而 **xi** 在-3 与 4 间变化。

与第 11 章里所述的二维插值相对应，**griddata** 对于数据非单调或不规则分布的情况也同样有效。例如，考虑随机数据：

```
» x=2*rand(1,20);   %   nonmonotonic x-axis
» y=4*rand(1,20)-2;   %   nonmonotonic y-axis
```

```
» [X,Y]=meshgrid(x,y);  %  make data plaid
» Z=(X+Y).^2;  %  compute function
» xi=linspace(0,2,50);  %  interpolated monotonic x-axis values
» yi=linspace(-2,2,30);  %  interpolated monotonic y-axis values
» [Xi,Yi]=meshgrid(xi,yi);  %  make data plaid
» Zi=griddata(X,Y,Z,Xi,Yi);  %  interpolated on monotonic plaid data
```

这里，对随机数据插值给出更适合于作图的单调数据。这对于本章后面讨论的等值线图特别重要，因为它要求数据定义在均匀间隔的网格中。

在上面的两个例子中，较容易对所需的插值重新计算函数。作为一般规则，如果容易计算出感兴趣的函数，就要避免使用函数 **griddata**。而当感兴趣的点的函数不可得到或需要很大的计算量时，函数 **griddata** 提供了一个工具，以均匀分布的内插数据点，对基本函数进行估计。

18.5 网格图

利用在 **x—y** 平面的矩形网格点上的 **z** 轴坐标值，MATLAB 定义了一个**网格**曲面。MATLAB 通过将邻接的点用直线连接起来形成网状曲面，其结果好象在数据点有结点的鱼网。例如，用 MATLAB 的函数 **Peaks** 可以画一个简单的曲面。

```
» [X,Y,Z]=peaks(30);
» mesh(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ MESH of PEAKS  ‘ )
```

输出见图 18.6.

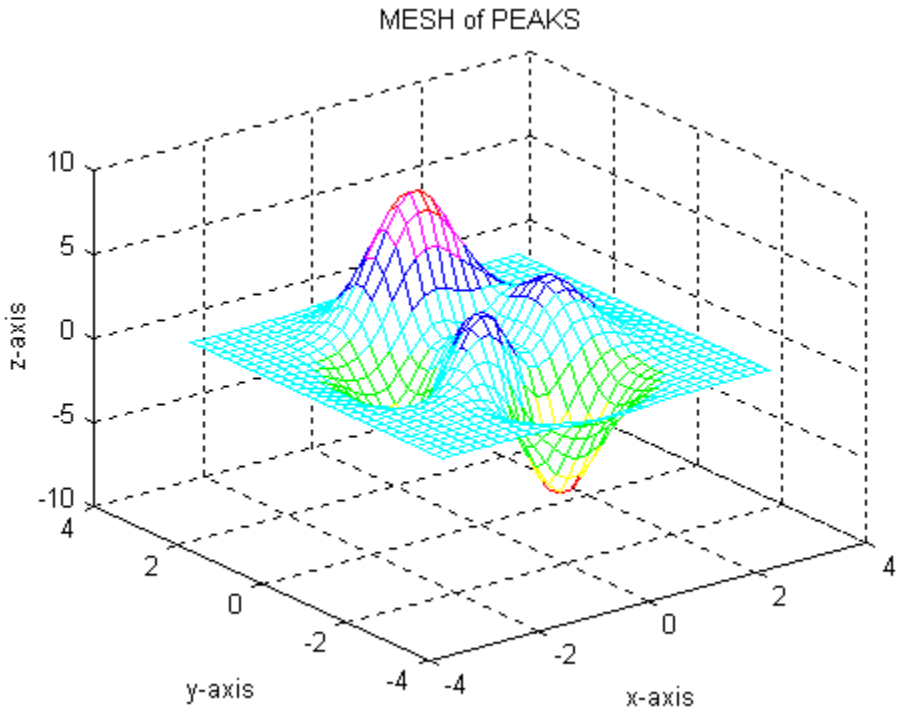


图 18.6　函数 PEAKS 的网格图

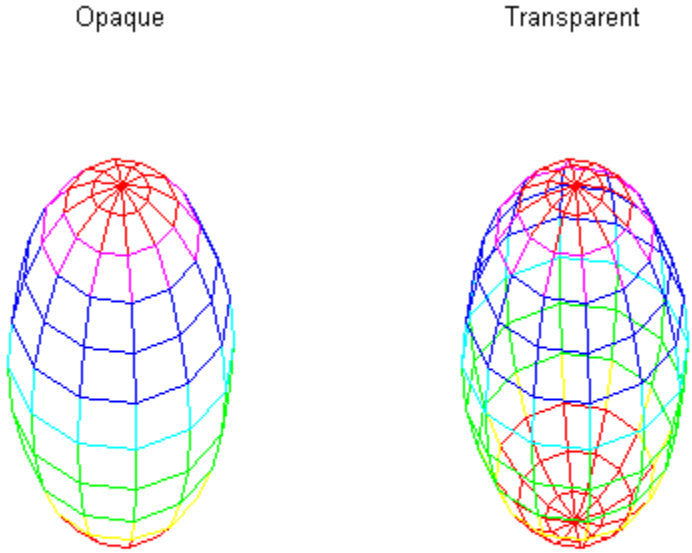
在显示器上要注意到线的颜色与网格的高度有关。一般情况下，函数 **mesh** 有可选的参量来控制绘图中所用的颜色。关于 MATLAB 如何使用、改变颜色在下一章讨论。在任何情况下，由于颜色用于增加图形有效的第四维，这样使用的颜色被称做伪彩色。

除了上例中的输入参量，函数 **mesh** 和大多数三维绘图函数都可按多种输入参量调用。这里所用的句法是最详细的，它给出了所有三个坐标轴的信息。最通常的变更方法是使用向量，将它传递给 **meshgrid**，生成 x 与 y 坐标轴，比如 **mesh(x,y,z)**。有关其它调用语法形式的信息，参阅 *MATLAB 参考指南*或在线帮助。

如上图所示，网格线条之间的区域是不透明的。MATLAB 命令 **hidden** 控制网格图的这个特性。比如，用 MATLAB 的函数 **sphere** 产生两个球面如下：

```
» [X,Y,Z]=sphere(12);
» subplot(1,2,1)
» mesh(X,Y,Z),title( ‘ Opaque ‘ )
» hidden on
» axis off
» subplot(1,2,2),title( ‘ Transparent ‘ )
» mesh(X,Y,Z)
» hidden off
» axis off
```

输出见图 18.7.



不透明　　透明

图 18.7

左边的球是不透明的（线被隐蔽），而右边的球是透明的（线未被隐蔽）。

MATLAB 的 **mesh** 有两个同种函数：**meshc**，它画网格图和基本的等值线图；**meshz**，它画包含零平面的网格图。

```
» [X,Y,Z]=peaks(30);
» meshc(X,Y,Z)    %    mesh plot with underlying contour plot
» title( ‘ MESHC of PEAKS ‘ )
» meshz(X,Y,Z)    %    mesh plot with zero plane
» tiTle( ‘ MESH of PEAKS ‘ )
» title( ‘ MESH of PEAKS ‘ )
» hidden off    %    make mesh transparent so minimums can be seen
```

输出分别见图 18.8 和图 18.9.

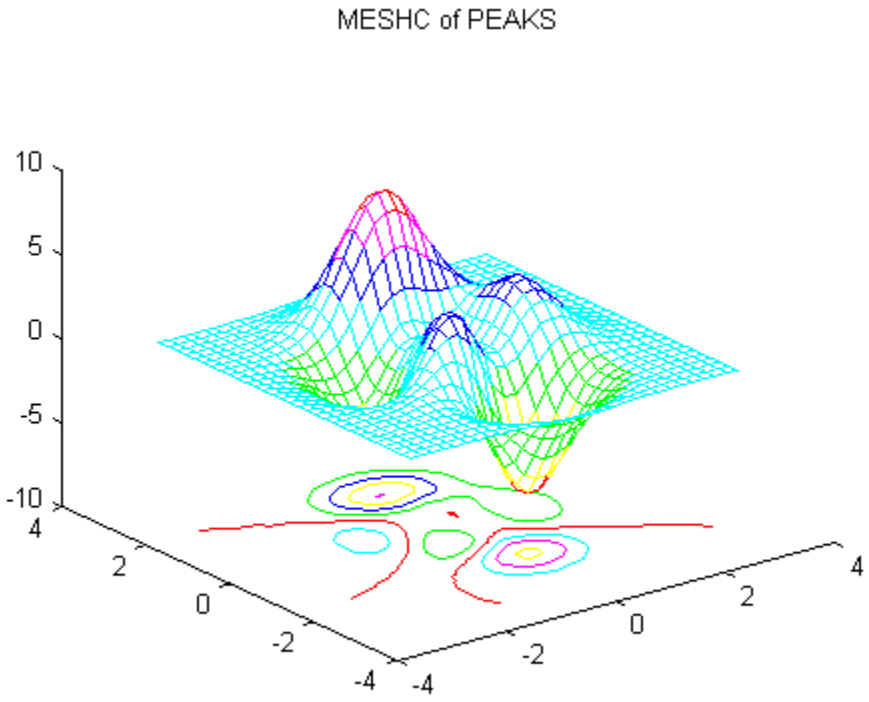


图 18.8　函数 PEAKS 的网格图和基本等值线图

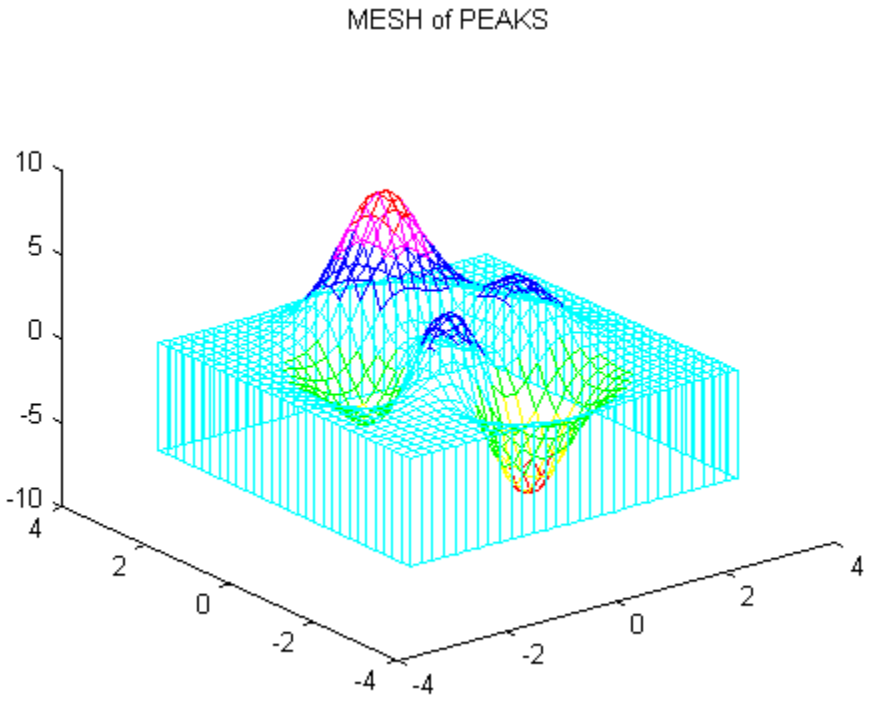


图 18.9　函数 PEAKS 的带零平面的网格图

关于以上函数更详细的信息参阅 *MATLAB* 参考指南或使用在线帮助。

18.6 曲面图

曲面图，除了各线条之间的空档（称作**补片**）用颜色填充以外，和网格图看起来是一样的。这种图一般使用函数 **surf** 来绘制。自然，函数 **surf** 使用和函数 **mesh** 相同的调用语法。比如：

```
» [X,Y,Z]=peaks(30);
» surf(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURF of PEAKS ‘ )
```

输出见图 18.10.

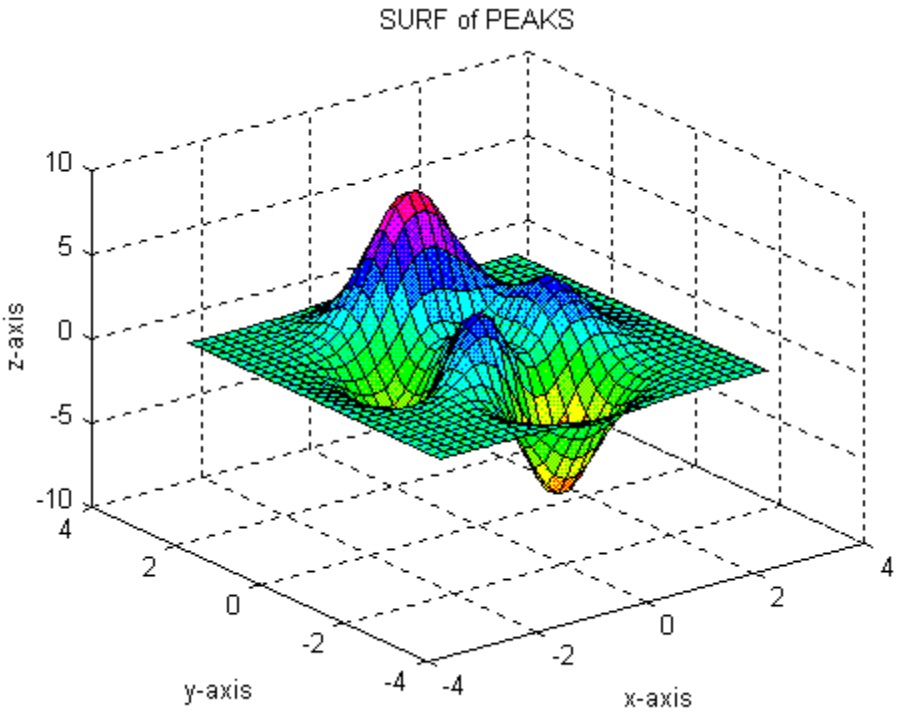


图 18.10　函数 PEAKS 的曲面图

曲面图的一些特性正好和网格图相反：它的线条是黑色的，线条之间的补片有颜色；而在网格图里，补片是黑色的而线条有颜色。对函数 **mesh**，颜色沿着 z 轴按每一补片变化，而线条颜色不变。

在曲面图里，人们不必考虑象网格图一样隐蔽线条，但要考虑用不同的方法对表面加色彩。在前面的曲面图的例子中，就是分割成**块**，每块就象一块染色玻璃窗口或物体，黑线便是各单色染色玻璃块之间的连接。除此以外，MATLAB 还提供了**平滑**加颜色和**插值**加颜色功能。这可以通过调用函数 **shading** 来实现。

```
» [X,Y,Z]=peaks(30);
```

» surf(X,Y,Z)　%　same plot as above
» grid,xlabel(‘ x-axis ‘),ylabel(‘ y-axis ‘),zlabel(‘ z-axis ‘)
» title(‘ SURF of PEAKS ‘)
» shading flat

输出见图 18.11.

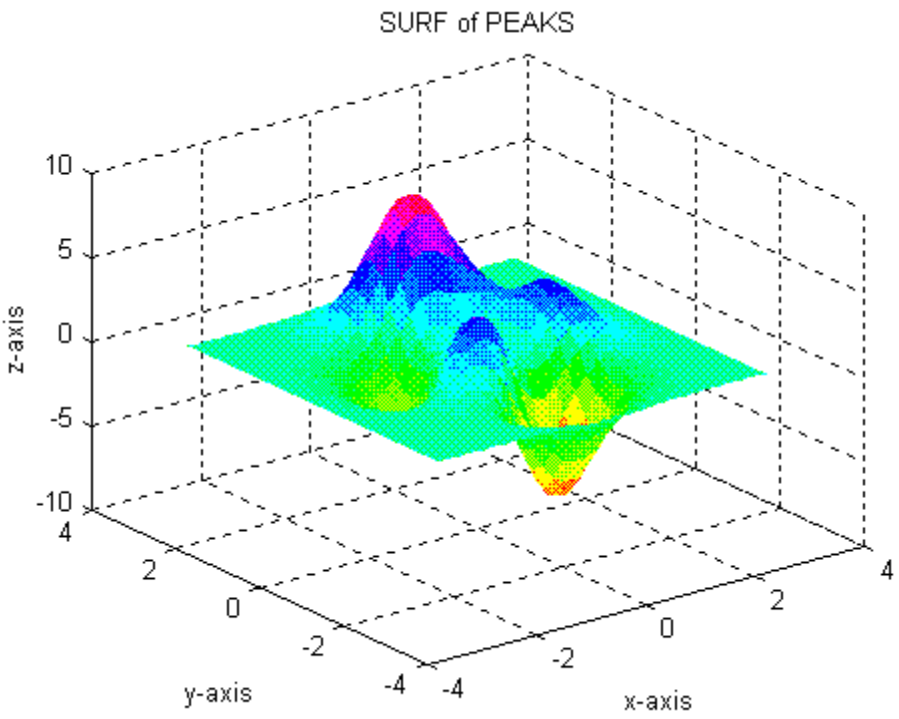


图 18.11　函数 PEAKS 的平滑加彩色曲面图

如上所示平滑加色彩的例子中，每一补片仍保存着单一的颜色，但各块连接处的黑线已去掉。

» shading interp

输出见图 18.12.

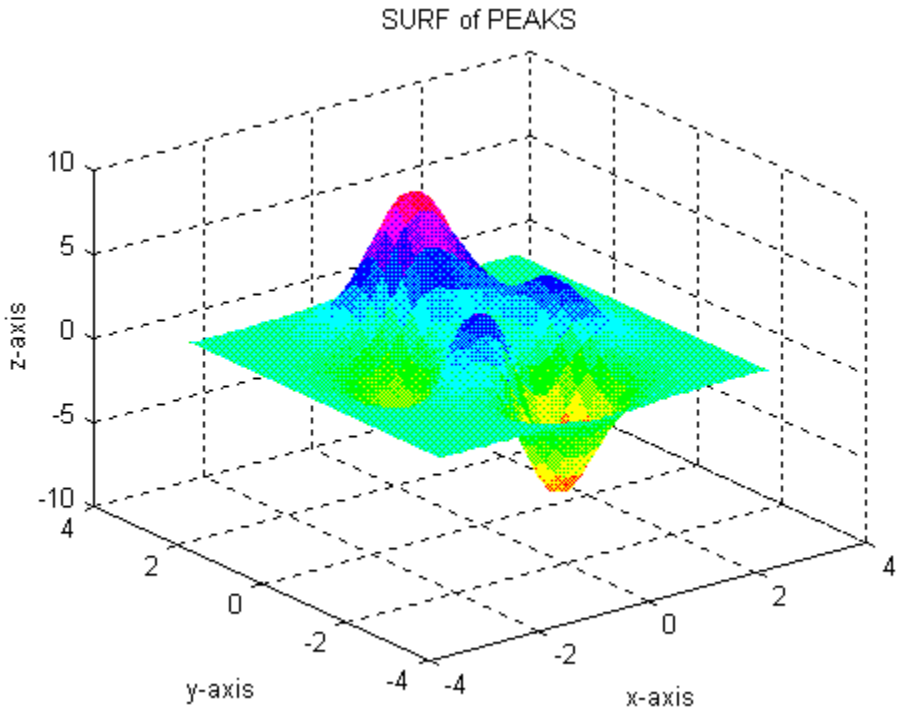


图 18.12　函数 PEAKS 的插值加彩色曲面图

如上所示内插加色彩的例子中，同样去掉了线条，但各补片以插值加颜色，即各补片的颜色根据赋予顶点的色值，对其区间进行了插值计算。很明显，插值色彩需要比分块和平滑更多的计算量。**在一些计算机系统中，插值色彩会产生非常长的打印延时或打印错误。这问题不在于 PostScript 文件太大，而是由于在打印机上产生沿图形曲面连续变化的阴影所需的巨大计算量。通常对这个问题最简单的解决方法是使用平滑加色彩法来打印**。

色彩对 **surf** 作图的视觉效果有着巨大的影响。对网格图也是如此，尽管由于只有线条有颜色，对视觉效果的影响相对要小一些。

因为曲面图不能作成透明，但在一些情况下可以很方便地移走一部分表面以便看到表面以下部分，在 MATLAB 中，这是通过在所期望的洞孔的所在位置，将数据置为特定的 **NaN** 来实现。**由于 NaN 没有任何值，所有的 MATLAB 作图函数都忽略 NaN 的数据点，在该点出现的地方留下一个洞孔**。例子如下：

```
» [X,Y,Z]=peaks(30);
» x=X(1,:);    %    vector of x axis
» y=Y(:,1);    %    vector of y axis
» i=find(y>.8 & y<1.2);    %    find x-axis indices of hole
» j=find(x>-.6 & x<.5);    %    find x-axis indices of hole
» Z(i,j)=nan*Z(i,j);    %    set values at hole indices to NaNs
» surf(X,Y,Z)
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURF of PEAKS with a Hole ‘ )
```

输出见图 18.13.

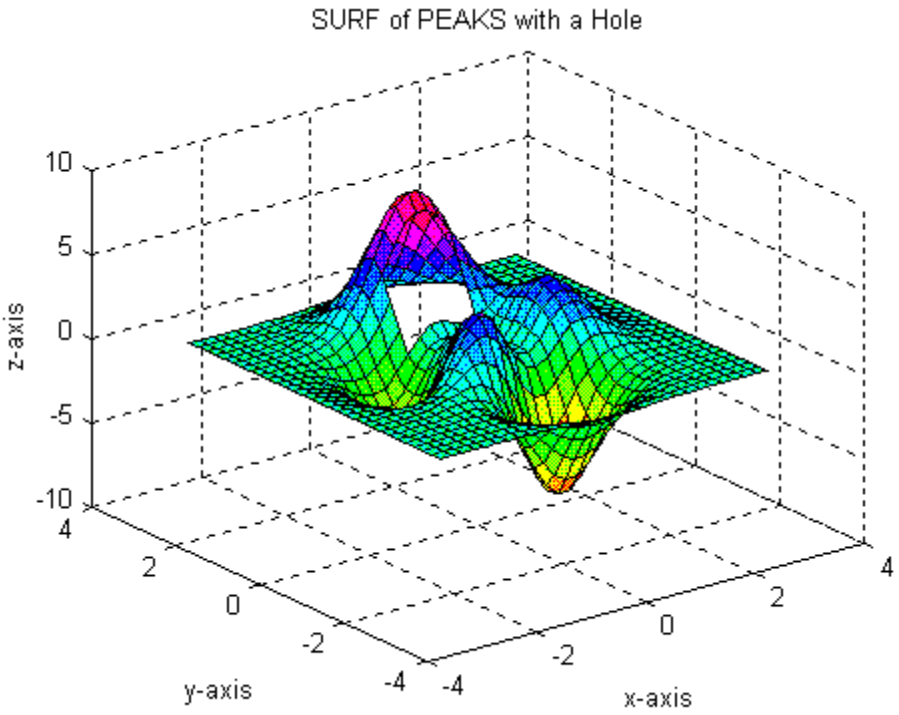


图 18.13　函数 PEAKS 的带洞孔曲面图

MATLAB 的 **surf** 也有两个同种函数：**surfc**，它画出具有基本等值线的曲面图；**surfl**，它画出一个有亮度的曲面图。例如：

```
» [X,Y,Z]=peaks(30);
» surfc(X,Y,Z)    %    surf plot with    contour plot
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ SURFC of PEAKS ‘ )
```

输出见图 18.14.

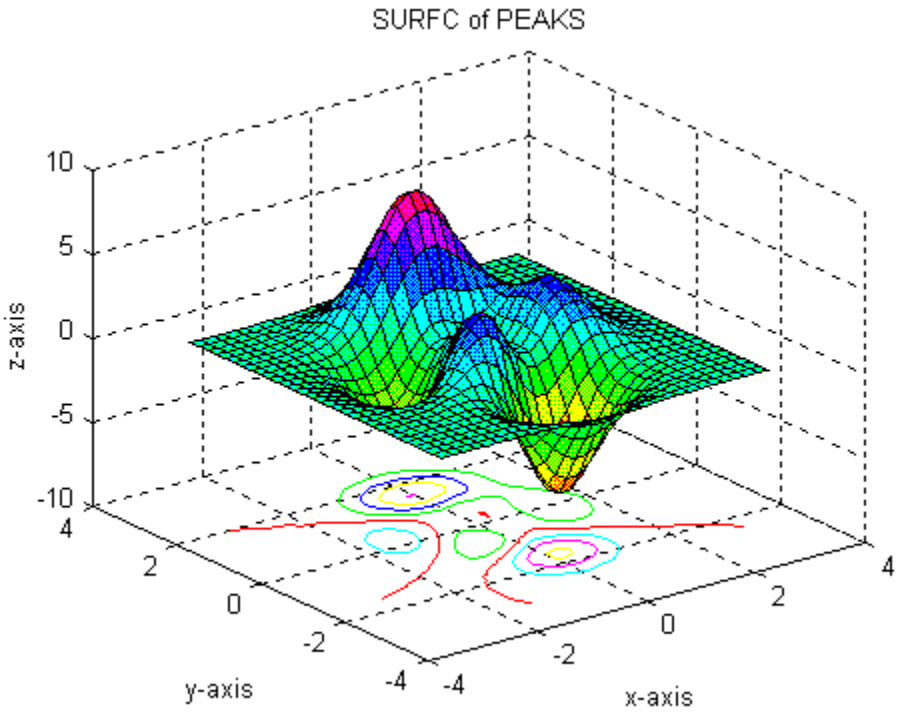


图 18.14　函数 PEAKS 的曲面图和基本等值线图

```
» [X,Y,Z]=peaks(30);
» surfl (X,Y,Z)    %    surf plot with lighting
» shading interp    %    surfl plots look best with interp shading
» colormap pink    %    they also look better with shades of a single color
» grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
»title( ‘ SURFL OF PEAKS ‘ )
```

输出见图 18.14.

图 18.16　函数 PEAKS 的带光线照明曲面图

关于加到曲面的亮度，函数 **surfl** 作了许多假设。有关设置亮度属性的详细信息参阅 *MATLAB* 参考指南的函数 **surfl** 或使用在线帮助。同样，在上面执行的命令中，**colormap** 是 MATLAB 函数，它对图形施加一套不同的颜色。这个函数在下一章讨论。

18.7 等值线图

MATLAB 提供了另一种基本的三维图形，即三维等值线图。这种图形通过函数 **contour3** 来绘制。

```
» [x,y,z]=peaks(30);
» contour3(X,Y,Z,16)    %    draw sixteen contour lines
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
```

输出见图 18.16.

图 18.16　函数 PEAKS 的三维等值线图

可以看到，图形中每一条线的颜色遵循了与二维函数 **plot** 一样的次序。这种颜色次序可以表现出明显的对比，但经常模糊了所代表的数据的一些重要特性。如果能使每一条线遵循在**网格图**和**曲面图**里所用的加色方法，那么效果会好得多。也许在 MATLAB 的下一个版本中，这种颜色设置会成为缺省设置，但使用在下一章要讨论的 MATLAB 图形处理能力，也能解决这个问题。

```
» [X,Y,Z]=peaks(30);
» N=16;    %    number of contour lines and their colors
» clf    %    clear the current figure
» view(3)    %    set view to 3-D
» hold on    %    hold blank screen
» set(gca, ‘ ColorOrder ‘ ,hsv(N))    %    use colors from default hsv colormap
» contour3(X,Y,Z,N)    %    draw N contour lines
» grid,xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
» hold off
```

输出见图 18.17.

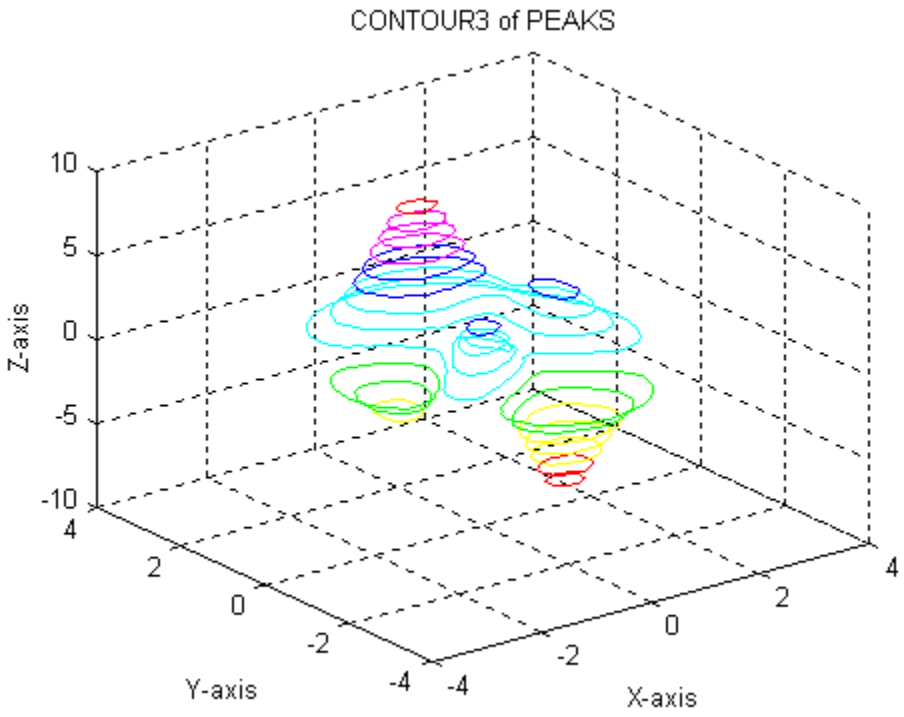


图 18.17　函数 PEAKS 的三维等值线图

现在，各条等值线的颜色沿着 z 轴的变化和**网格图**和**曲面图**一样。为方便起见，这种策略已体现在精通 *MATLAB* 工具箱的函数 **mmcont3** 中。**mmcont3** 具有和函数 **contour3** 相同的调用语法的变化，并允许选择可用的颜色映象。例如，» **mmcont3（X，Y，Z，N，‘hsv’）** 复制上面的图形。**mmcont3** 的在线帮助如下：

» help mmcont3

MMCONT3 3-Dcontour plot using a colormap

MMCONT3(X,Y,Z,N,C) plots N contours of Z in 3-D using the color specified in C.C can be a linestyle and color as used in plot,e.g., ‘ r- ‘ ,or C can be the string name of a colormap.X and Y.

define the axis limits.

If not given default argument values are :N=10 ,C= ‘ hot ‘ ,X and Y=row and column indices of

Z.Examples:

MMCONT3(Z)	10 lines with hot colormap
MMCONT3(Z,20)	20 lines with hot colormap
MMCONT3(Z, ‘ copper ‘)	10 lines with copper colormap
MMCONT3(Z,20, ‘ gray ‘)	20 lines with gray colormap
MMCONT3(X,Y,Z, ‘ jet ‘)	10 lines with jet colormap
MMCONT3(Z, ‘ c-- ‘)	10 dashed lines in cyan

MMCONT3(X,Y,Z,25, ‘ pink ‘) 25 lines in pink colormap

CS=MMCONT3(. . .) returns the contour matrix CS as described in CONTOURC.
[CS,H]=MMCONT3(. . .) returns a column vector H of handles to line objects

---

帮助信息：

MMCONT3(X,Y,Z,N,C)用由 C 指定的颜色在三维空间内画 N 条 Z 方向的等值线图。C 可以是在 plot 中使用的线形 和颜色，例如 ‘ r- ‘ ；或者 C 可以是一个颜色映象的字符串名。X 和 Y 指定了坐标轴的范围。如果 未指定参数，缺省的参数值是：N=10，C= ‘ hot ‘ ，X 和 Y 分别是 Z 的行和列的下标。举例：

MMCONT3(Z)	用暖色映象画 10 条等值线
MMCONT3(Z,20)	用暖色映象画 20 条等值线
MMCONT3(Z, ‘ copper ‘)	用铜黄色映象画 10 条等值线
MMCONT3(Z，20， ‘ gray ‘)	用灰色映象画 20 条等值线
MMCONT3(X,Y,Z, ‘ jet ‘)	用**‘jet’暖色映象画 10 条等值线
MMCONT3(Z, ‘ c-- ‘)	画 10 条青蓝色的虚划线等值线
MMCONT3(X,Y,Z,25, ‘ pink ‘)	用粉红色映象画 25 条等值线
CS=MMCONT3(…)	如在 CONTOURC 中描述，返回等值线矩阵 CS。
[CS,H]=MMCONT3(…)	把句柄的列向量 H 返回到线条对象。

等值线也可由一种颜色给出：

```
» [x,y,z]=peaks(30);
» contour3(X,Y,Z,16, ‘ y ‘ )  %  draw sixteen contour lines in yellow
» grid,xlabel( ‘ x-axis ‘ ),ylabel( ‘ y-axis ‘ ),zlabel( ‘ z-axis ‘ )
» title( ‘ CONTOUR3 of PEAKS ‘ )
```

输出见图 18.18

图 18.18 函数 PEAKS 的黄色三维等值线图

关于颜色使用的详细信息参阅下一章；以上函数使用的详细信息参阅 *MATLAB* 参考指南或使用在线帮助。

18.8 三维数据的二维图

有些情况下，希望得到三维数据的二维表示。在 MATLAB 里这一点是通过用函数 **view** 设置视角使其中一维不出现来实现的。另外，MATLAB 还提供了两个函数，将 **contour3** 和 **surf** 向下正视到 x－y 平面。例如，函数 **contour3** 的二维图就等价于 **contour**。

```
» [X,Y,Z]=peaks(30);
» contour(X,Y,Z,16)    %    draw sixteen contour lines
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ CONTOUR of PEAKS ‘ )
```

输出见图 18.19

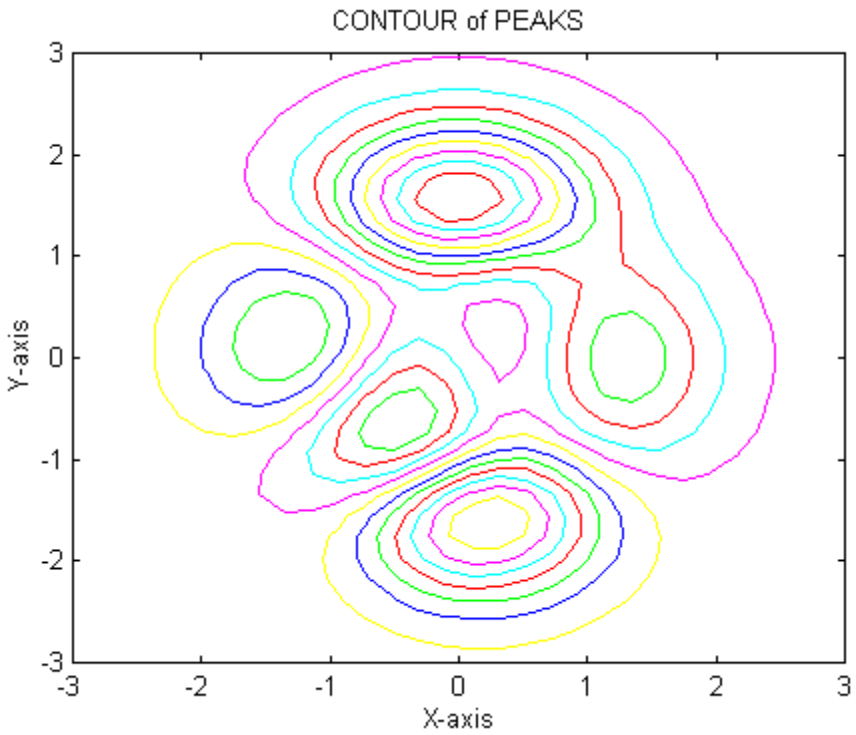


图 18.19　函数 PEAKS 的等值线图

要注意，它如何等效于使用 **contour3** 以及如何改变视点俯视到 x－y 平面。如同 **contour3**，图中的等值线利用 **plot** 命令的六种基本颜色。如前所述，因为颜色不提供视觉效果，这类图形不好使用并且引起混淆。这种缺省行为也许在 MATLAB 的下一版本中会改变，但也可用句柄图形来改变。

```
» [X,Y,Z]=peaks(30);
» N=16;    %    number of contour lines and their colors
» clf    %    clear the current figure
» hold on    %    hold blank screen
» set(gca, ‘ ColorOrder ‘ ,hsv(N))    %    use colors from default hsv colormap
» contour(X,Y,Z,N)    %    draw N contour lines
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
» title( ‘ CONTOUR of PEAKS ‘ )
» hold off
```

输出见图 18.20

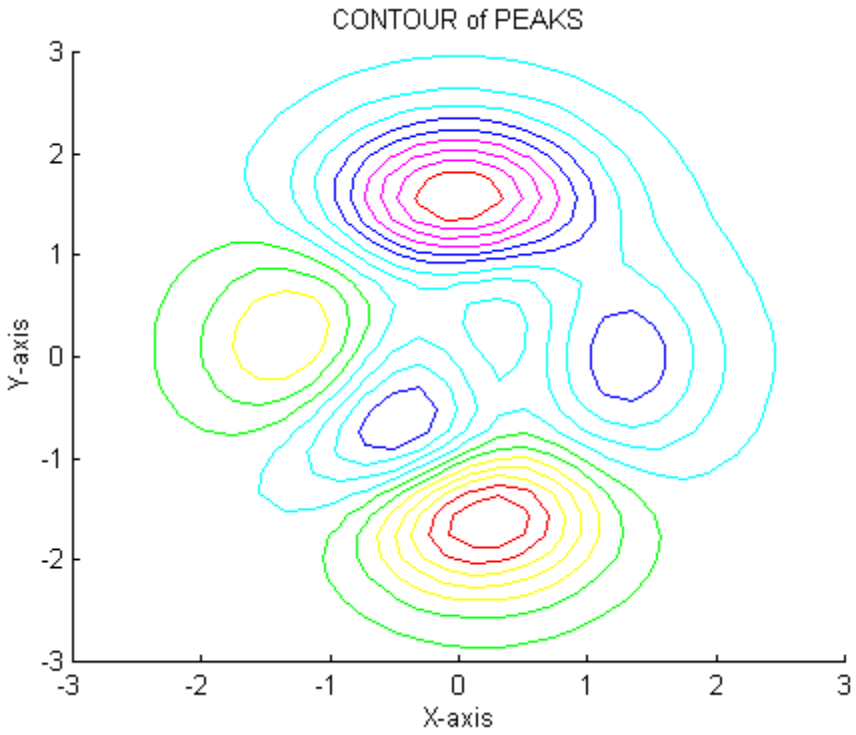


图 18.20　　函数 PEAKS 的等值线图

现在，等值线遵循 **hsv** 颜色映象里的级差，颜色提供了一种有用的效果！为方便起见，上述策略已体现在精通 *MATLAB* 工具箱的函数 **mmcont2** 中。**mmcont2** 接受和函数 **contour** 相同的调用语法变更，并允许选择一个可用的颜色映象。例如，» **mmcont2（X，Y，Z，N，‘hsv’）**复制上面的图形。**mmcont2** 的在线帮助如下：

```
» help mmcont2
MMCONT2 2-D contour plot using a colormap.
    MMCONT2(X,Y,Z,C) plots N contours of Z in 2-D using the color specified in C.C can be
    a linestyle and color as used in plot,e.g., ‘ r- ‘ ,or C can be the string name of a
colormap.X and          Y.
    define the axis limits.
```

If not given default argument values are :N=10,C= ‘ hot ‘ ,X and Y =row and column indices of Z.Examples:

MMCONT2(Z) 10 lines with hot colormap
MMCONT2(Z,20) 20 lines with hot colormap
MMCONT2(Z, ‘ copper ‘) 10 lines with copper colormap
MMCONT2(Z,20, ‘ gray ‘) 20 lines with gray colormap
MMCONT2(X,Y,Z, ‘ jet ‘) 10 lines with jet colormap
MMCONT2(Z, ‘ c-- ‘) 10 dashed lines in cyan
MMCONT2(X,Y,Z,25, ‘ pink ‘) 25 lines in pink colormap

CS=MMCONT2(. . .) returns the contour matrix CS as described in CONTOURC.
[CS,H]=MMCONT2(. . .) returns a column vector H of handles to line objects

---

帮助信息：

MMCONT2(X,Y,Z,N,C)用由 C 指定的颜色在二维平面内画出关于 Z 方向的 N 条等值线。C 可以是在函数 **plot** 中使用的线形和颜色，例如‘r-‘；或者是一个字符串指明颜色映象名。 X 和 Y 指定了坐标轴的范围。

如果未指定参数，缺省的参数值是：N=10，C= ‘ hot ‘ ，X 和 Y 分别是 Z 的行列下标。 例子如下：

MMCONT2(Z) 使用暖色颜色映象画 10 条等值线
MMCONT2(Z,20) 使用暖色颜色映象画 20 条等值线
MMCONT2(Z, ‘ copper ‘) 使用铜黄色颜色映象画 10 条等值线
MMCONT2(Z，20， ‘ gray ‘) 使用灰色颜色映象画 20 条等值线
MMCONT2(X,Y,Z, ‘ jet ‘) 使用‘jet’暖色颜色映象画 10 条等值线
MMCONT2(Z, ‘ c-- ‘) 使用青蓝色颜色映象用虚划线画 10 条等值线
MMCONT2(X,Y,Z,25, ‘ pink ‘) 使用粉红色颜色映象画 25 条等值线
CS=MMCONT2(…)返回等值线矩阵 CS，它在 CONTOURC 中描述。
[CS,H]=MMCONT2(…)返回一个包含有线条对象的句柄的列向量。

上面讨论的三维或二维等值线图都假定数据定义在矩形网格或论域内。当不是这种情形时，数据必须被转换到矩形网格上，使等值线图看起来逼真。如本章前面 18.4 节所述，MATLAB 函数 **griddata** 能实现所需的转换，更详细的信息参阅该节。

函数 **surf** 的二维等效函数是 **pcolor**，它代表伪彩色。

```
» [X,Y,Z]=peaks(30);
» pcolor(X,Y,Z);   %   surf plot view from above
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ PCOLOR   of   PEAKS ‘ )
```

输出见图 18.21.

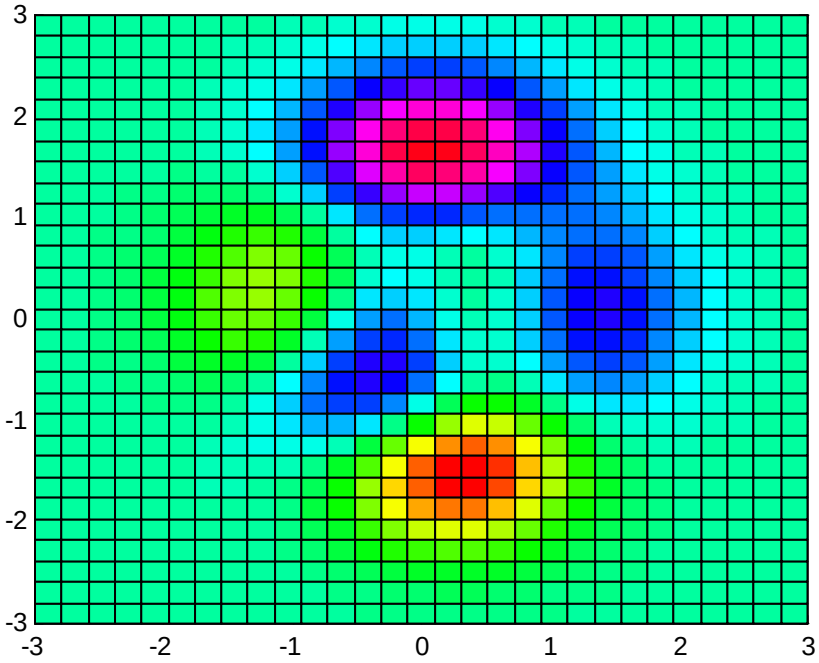


图 18.21　函数 PEAKS 的伪彩色图

由于这是一个 **surf** 图，可以使用函数 **shading**。另外，有时在 **pcolor** 图的上面放一个单色**等值线图**是很有用的。

```
» [X,Y,Z]=peaks(30);
» pcolor(X,Y,Z);
» shading interp
» hold on
» contour(X,Y,Z,19, ‘ k ‘ )    %    add 19 contour lines in black
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ PCOLOR    and    CONTOUR    of    PEAKS ‘ )
» hold off
```

输出见图 18.22.

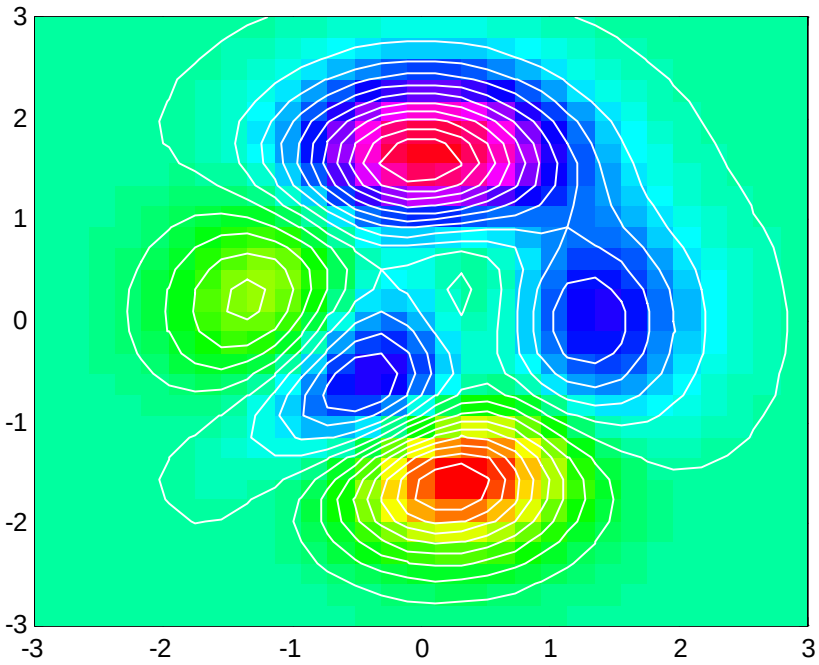


图 18.21　函数 PEAKS 的伪彩色图和等值线图

18. 9 其它函数

除了上面讨论的三维函数，MATLAB 还提供了函数 **waterfall**，**quiver**，**fill3**，和 **clabel**。函数 **waterfall** 与函数 **mesh** 一样，只是它的网格线是在 x 轴方向出现。例如：

```
» [X,Y,Z]=peaks(30);
» waterfall(X,Y,Z)
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ ),zlabel( ‘ Z-axis ‘ )
```

输出见图 18.23.

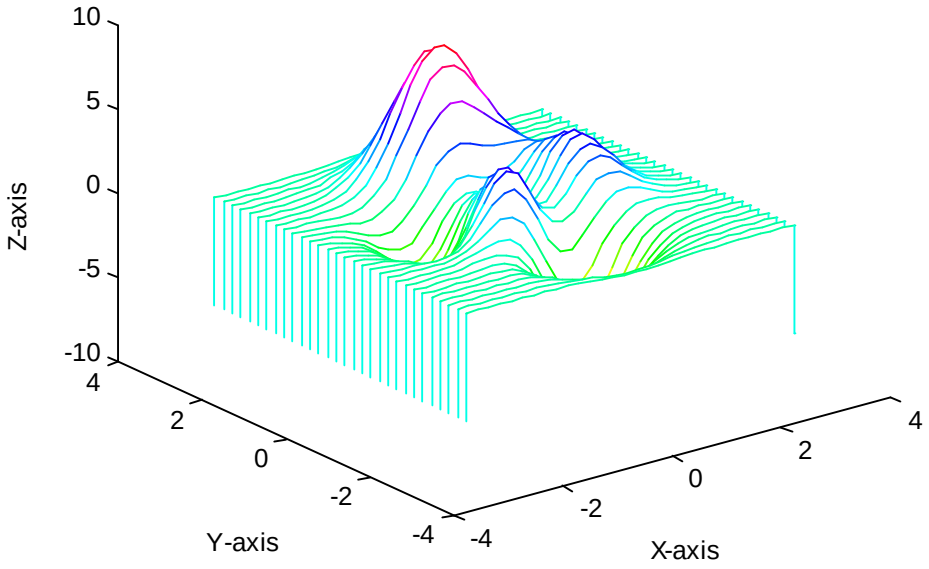


图 18.23　函数 PEAKS 的 waterfall 图

函数 **quiver** 在等值线图上画出方向或速度箭头。例如：

```
» [X,Y,Z]=peaks(16);
» [DX,DY]=gradient(Z, .5, .5);
» contour(X,Y,Z,10)
» hold on
» quiver(X,Y,DX,DY)
» hld off
```

输出见图 18.24.

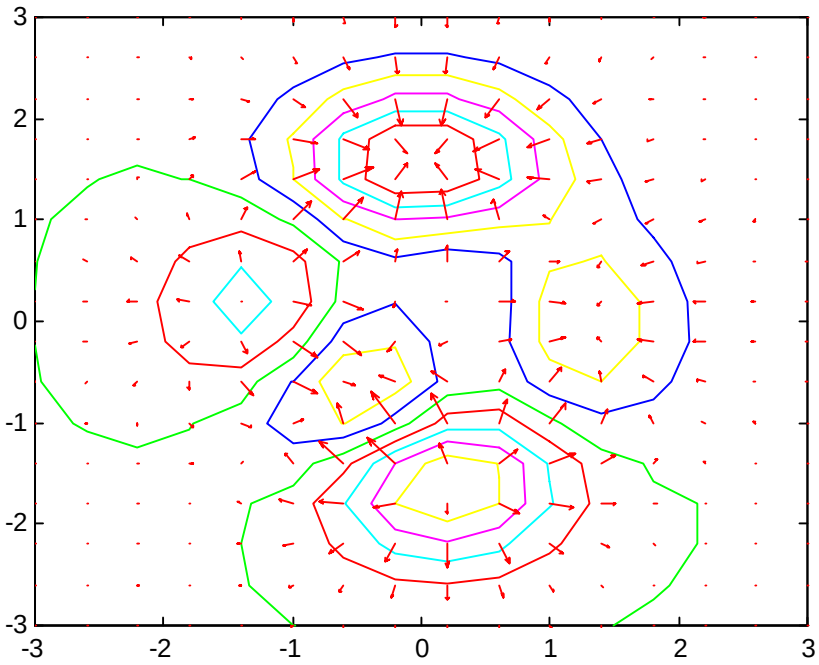


图 18.24　函数 PEAKS 的 quiver 图

函数 **fill3** 等效于三维函数 **fill**，可在三维空间内画出填充过的多边形。函数 **fill3(x,y,z,c)** 使用数组 **x**，**y** 和 **z** 作为多边形的顶点而 **c** 指定了填充的颜色。例如，下例用随机的顶点坐标值画出五个黄色三角形。

» fill3(rand(3,5),rand(3,5),rand(3,5), ' y ')

输出见图 18.25.

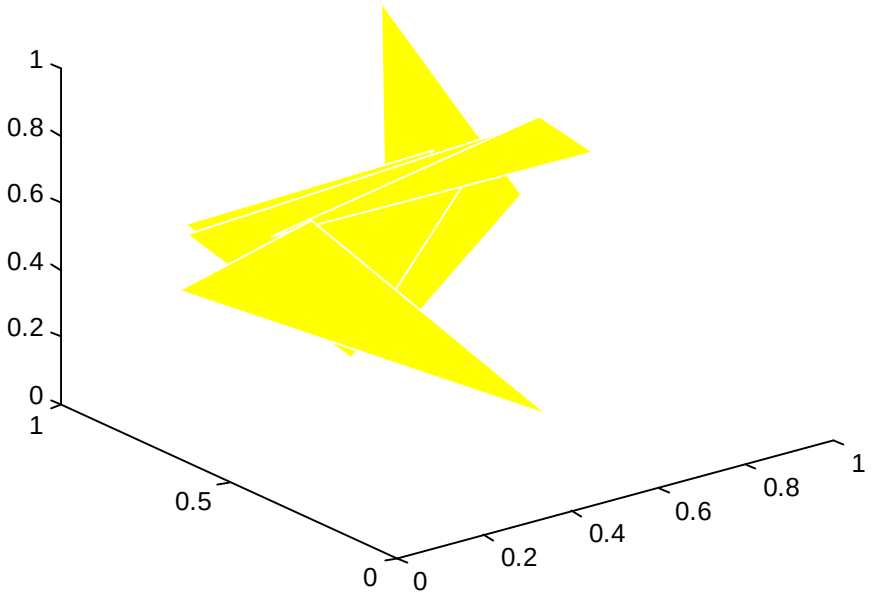


图 18.25　函数 fill3 图示

函数 **clabel** 给等值线图标上高度值。这样做时，函数 **clabel** 需要函数 **contour** 等值线矩阵的输出。

```
» [X,Y,Z]=peaks(30);
» cs=contour(X,Y,Z,8);    %    request output from contour
» clabel(cs)    %    add labels identifying heights
» xlabel( ‘ X-axis ‘ ),ylabel( ‘ Y-axis ‘ )
» title( ‘ CONTOUR    of    PEAKS    with    LABELS ‘ )
```

输出见图 18.26.

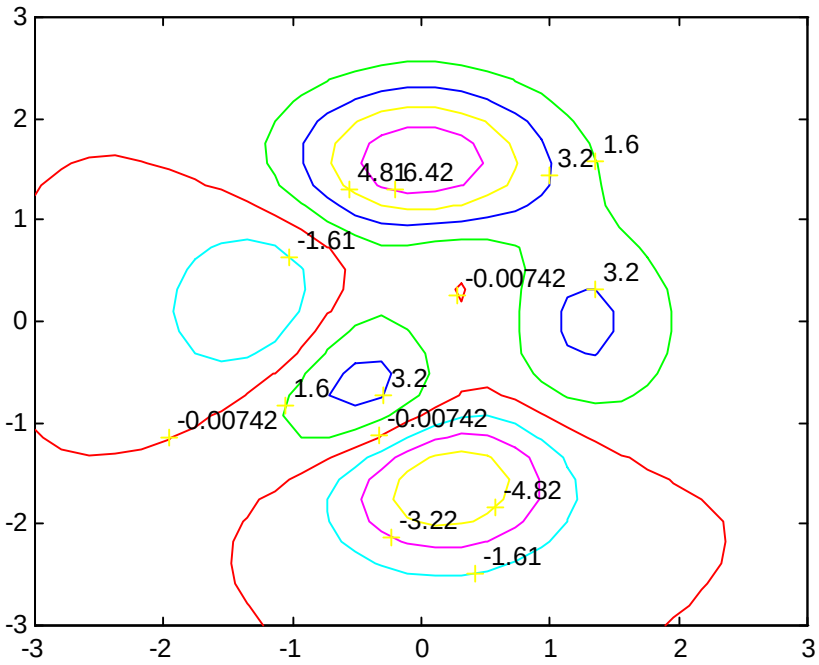


图 18.26　函数 PEAKS 的带标志等值线图

18. 10 动画

MATLAB 提供了一种能力，它可以存储一系列各种类型的二维或三维图，然后象放电影一样把它们按次序重放出来。在某种意义上，动画提供的运动为图形增加另一个维数。通常图形的次序不必以任意的方式关联起来。一种明显的动画类型是取出三维图形然后缓慢地将它旋转，这样我们就可以从不同角度来观察它。另一种类型是当一个参数变化时，依次显示某些问题解的图形。

MATLAB 中的函数 **moviein**，**getframe** 和 **movie** 提供了捕捉和播放动画的所需工具。函数 **moviein** 可以产生一个帧矩阵来存放动画中的帧；函数 **getframe** 对当前的图象进行快照；而函数 **movie** 按顺序回放各帧。照这样，捕捉和回放动画的方法是：（1）创建帧矩阵；（2）对动画中的每一帧生成图形，并把它捕捉到到帧矩阵里；（3）从帧矩阵里回放动画。

考虑下面的一段脚本 M 文件例子，例中绘制了函数 **peaks** 并且将它绕 z 轴旋转。

```
%    movie making example: rotate a 3-D surface plot

[X,Y,Z]=peaks(30);    %    create data
surfl(X,Y,Z)    %    plot surface with lighting
axis([-3    3    -3    3    -10    10])    %    fix axes so that scaling does not change
axis off    %    erase axes because they jump around
shading interp    %    make it pretty with interpolated shading
colormap(hot)    %    choose a good colormap for lighting
```

```
m=moviein(15);    %    choose 15 movie frames for frame matrix m

for I=1:15    %    rotate and capture each frame
      view(-37.5+24*(i-1),30)    %    change viewpoint for this frame
      m(:,i)=getframe;    %    add figure to frame matrix
end

movie(m)    %    play the movie!
```

注意到动画中的每一帧在帧矩阵中占据一个不同的列。帧矩阵的大小随着动画中的帧数和图形窗口的大小而增加，而与所绘图形的复杂性无关，这是因为函数 **getframe** 仅仅是捕捉位图。按缺省，函数 **movie** 只放一遍动画。通过加入其它输入参量，它可以向前放、向后倒放、放指定次数或按特定的桢速率播放。关于这些特征的详细信息，参阅 *MATLAB* 参考指南或使用在线帮助。

由于上述的动画制作策略很有用，它已体现在精通 *MATLAB* 工具箱的函数 **mmspin3d** 中。

```
function M=mmspin3d(n)
%    MMSPIN3D Make Movie by 3D Azimuth Rotation of Current Figure.
%    MMSPIN3D(N) captures and plays N frames of the current figure
%    through one rotation about the Z-axis at the current elevation.
%    M=MMSPIN3D(N) returns the movie in M for later playing with movie.
%    If not given,N=18 is used.
%    MMSPIN3D fixes the axis limits and issues axis off.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

if nargin<1,n=18;end
n=max(abs(round(n)),2);

axis(axis);
axis off
incaz=round(360/n);
[az,el]=view;

m=movie(n);
for i=1:n
      view(az+incaz*(i-1),el)
      m(:,i)=getframe;
end
if nargout,    M=m;
else,          movie(m);
end
```

使用 **mmspin3d**，上述脚本简化如下：

```
%    movie-making example:rotate a 3-D surface plot

[X,Y,Z]=peaks(30);   %    create data
surfl(X,Y,Z)              %    plot surface with lighting
shading interp            %    make it pretty with interpolated shading
colormap(hot)             %    choose a good colormap for lighting
mmspin3d(15)
```

18.11 小结

本章所讨论的函数和它们的特征总结在表 18.2、表 18.3、表 18.4 和表 18.5 中：

表 18.2

三维绘图函数	
contour	二维等值线图，即从上向下看 contour3 等值线图
contour3	等值线图
fill3	填充的多边形
mesh	网格图
meshc	具有基本等值线图的网格图
meshz	有零平面的网格图
pcolor	二维伪彩色绘图，即从上向下看 **surf** 图
plot3	直线图
quiver	二维带方向箭头的速度图
surf	曲面图
surfc	具有基本等值线图的曲面图
surfl	带亮度的曲面图
waterfall	无交叉线的网格图

表示 18-3

三维绘图工具	
axis	修正坐标轴属性
clf	清除图形窗口
clabel	放置等值线标签
close	关闭图形窗口
figure	创建或选择图形窗口
getframe	捕捉动画桢
grid	放置网格
griddata	对画图用的数据进行内插
hidden	隐蔽**网格图**线条
hold	保留当前图形
meshgrid	产生三维绘图数据
movie	放动画

moviein	创建桢矩阵，存储动画
shading	在**曲面图**和**伪彩色图**中用**分块**、**平滑**和**插值**加阴影
subplot	在图形窗口内画子图
text	在指定的位置放文本
title	放置标题
view	改变图形的视角
xlabel	放置 x 轴标记
ylabel	放置 y 轴标记
zlabel	放置 z 轴标记

表 18.4

函数 view	
view(az,el)	设置视图的方位角 **az** 和仰角 **el**
view([az,el])	
view([x,y,z])	在笛卡儿坐标系中沿向量[x,y,z]正视原点设置视图，例如 **view([0 0 1])=view(0,90)**
view(2)	设置缺省的二维视图，**az=0，el=90**
view(3)	设置缺省的三维视图，**az=-37.5，el=30**
[az,el]=view	返回当前的方位角 **az** 和仰角 **el**
view(T)	用一个 4×4 的转置矩阵 **T** 来设置视图
T=view	返回当前的 4×4 转置矩阵

表 18.5

掌握 MATLAB 高级图形功能	
mmcont2(X,Y,Z,C)	具有颜色映象的二维等值线图
mmcont3(X,Y,Z,C)	具有颜色映象的三维等值线图
mmspin3d(N)	旋转当前图形的三维方位角来制作动画
mmview3d	用滑标来调整视角

关键词索引

chap 18

pcolor	伪彩色
3-D helix	三维螺旋线
3-D grid	三维网格
subplot	子图
Figure	图形窗口
viewpoint	视角
elevation	仰角

azimuth 方位角
Handle Graphics 句柄图形
slider 滑标
plaid 方格
mesh plots 网格图
mesh surface 网状曲面
surface 曲面图
contour plot 等值线图
zero plane 零平面
patch 补片
flat shading 平滑加色彩
interpolated shading 插值加色彩
progression 级差
color map 颜色映象
directional or velocity arrows 方向或速度箭头
movie 动画
frame 帧
underlying contour plot 基本等值线图

第十九章 颜色的使用

MATLAB 提供了许多在二维和三维空间内显示可视信息的工具。例如，看一条 sin 函数的曲线图就会比一堆数据提供更多的信息。这种用图表和图形来表示数据的技术叫做**数据可视化**。MATLAB 不仅是一个强大的计算工具，并且在以引人入胜和直观方式可视地表示数据方面也很有特色。

但是很多时候，一个简单的二维或三维图形不能一次显示出想要提供的全部信息。这时，颜色可以对图形提供一个附加的维数。前面章节讨论的许多绘图函数都可以接受一个可用的颜色参量，来增加这附加的维数。

本章的讨论以研究颜色映象开始：如何使用、显示、修改和如何创建用户自己的颜色映象。然后，阐述在一个图形窗口中仿真多个颜色映象的技术或只使用颜色映象的一部分的技术。最后，讨论照明模型并提供例子。

19.1 颜色映象理解

MATLAB 有一个叫**颜色映象**的数据结构来代表颜色值。颜色映象定义为一个有三列和若干行的矩阵。利用 0 到 1 之间的数，矩阵的每一行都代表了一种色彩。任一行的数字都指定了一个 **RGB** 值，即红、黄、蓝三种颜色的强度，形成一种特定的颜色。一些有代表性的 **RGB** 值在表 19.1 中给出。

表 19.1

简单颜色

Red（红）	Green（绿）	Blue（蓝）	颜色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
.5	0	0	深红
.5	.5	.5	灰色

有十个 MATLAB 函数产生预定的颜色映象。见表 19.2

表 19.2

标准颜色映象	
hsv	色彩饱和值（以红色开始和结束）
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	**hsv** 的一种变形（以蓝色开始和结束）
copper	线性铜色度
prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
flag	交替为红色、白色、蓝色和黑色

按缺省，上面所列的各个颜色映象产生一个 64×3 的矩阵，指定了 64 种颜色 **RGB** 的描述。这些函数都接受一个参量来指定所产生矩阵的行数。比如 **hot(m)**产生一个 m×3 的矩阵，它包含的 **RGB** 颜色值的范围从黑经过红、橘红和黄，到白。

大多数计算机在一个 8 位的硬件查色表中一次可以显示 256 种颜色，当然有些计算机的显示卡可以同时显示更多的颜色。这就意味着在不同的图中，一般一次可以用三或四个 64×3 的颜色映象。如果使用了更多的颜色映象输入项，计算机必须经常在它的硬件查色表中调出输入项。比如，当在画 MATLAB 图形时背景图案发生了变化，就是发生了这种情况。所以，除非计算机有一次显示更多种颜色的显示卡，最好任何一次所用的颜色映象输入项数都小于 256。

19.2 颜色映象使用

语句 **colormap(M)**将矩阵 **M** 作为当前图形窗口所用的颜色映象。例如，**colormap**（**cool**）装入了一个有 64 个输入项的 **cool** 颜色映象。**colormap default** 装入了缺省的颜色映象（**hsv**）。

函数 **plot**、**plot3**、**contour** 和 **contour3** 不使用颜色映象，它们使用列在 plot 颜色和线形表中的颜色。而大多数其它绘图函数，比如 **mesh**、**surf**、**fill**、**pcolor** 和它们的各种变形函数，使用当前的颜色映象。

接受颜色参量的绘图函数中的颜色参量通常采用以下三种形式之一：（1）字符串。代表 **plot** 颜色或线型表中的一种颜色，比如，' **r** '代表红色；（2）三个输入的行向量。它代表一个单独的 **RGB** 值，比如**[.25 .50 .75]**；（3）矩阵。如果颜色参量是一个矩阵，其元素作了调整，并把它们用作当前颜色映象的下标。最后一种形式会在以后作更多讨论。

19.3 颜色映象显示

可以用多种途径来显示一个颜色映象。其中一个方法是观察颜色映象矩阵的元素。

```
» hot(8)
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

上面的数据显示出第一行是 1/3 红色，而最后一行是白色。另外，函数 **pcolor** 可以用来显示一个颜色映象。例如：

```
» n=16;
» colormap(jet(n))
» pcolor([1:n+1;1:n+1]' )
» title( 'Using Pcolor to Display a Color Map' )
```

输出见图 19.1.

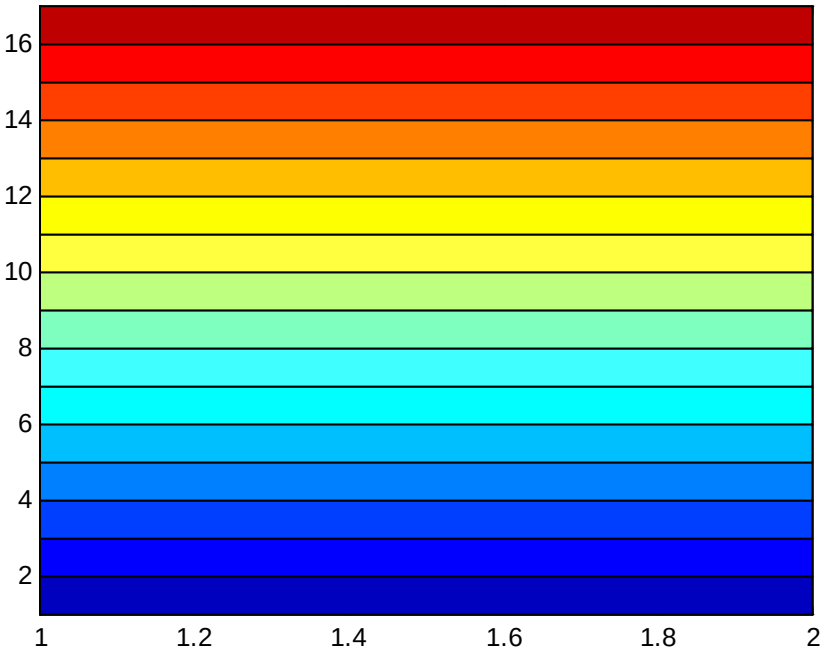


图 19.1　用伪彩色来显示颜色映象

因为上面这一段程序很有用处，它已经装入精通 MATLAB 工具箱中的函数 **mmshow** 中。

```
» help mmshow
MMSHOW PCOLOR Colormap Display
    MMSHOW uses pccolor to display the current colormap.
    MMSHOW(MAP) displays the colormap MAP.
    MMSHOW(MAP(N)) displays the colormap MAP having N elements.
    Examples:MMSHOW(hot)
             MMSHOW(pink(30))
```

帮助信息：

MMSHOW　显示 PCOLOR 颜色映象
MMSHOW　使用 pcolor 来显示当前颜色映象
MMSHOW(MAP)　显示 MAP 颜色映象
MMSHOW（MAP(N)）　显示一个有 N 个元素的 MAP 颜色映象
例子：MMSHOW（hot）
　　MMSHOW（pink（30））

函数 **mmshow** 取和 **colormap** 同样的输入参量，但在这种情况下它用自己的伪彩色显示而不是把颜色映象施加到当前图形。

另一种途径是使用 MATLAB 的函数 **rgbplot**，它可以把颜色映象的各列分别画成红、绿

和蓝色。例如：

» rgbplot(hot)

输出见图 19.2.

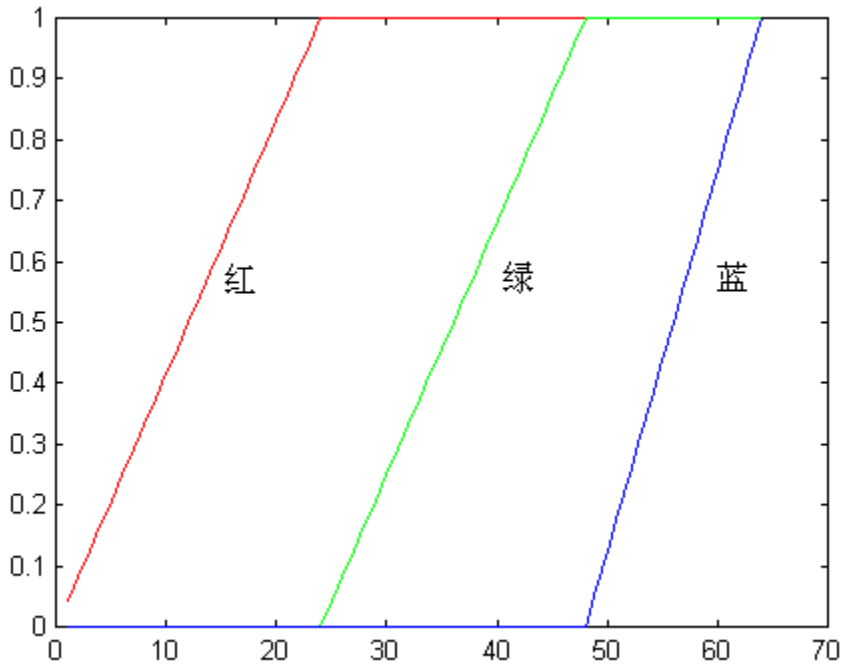


图 19.2　用红、绿和蓝色画颜色映象

图中显示红色分量首先增加，然后是绿色，最后是蓝色。**rgbplot（gray）**表示所有三列数据均匀线性地增加（三条线重叠）。

最后，函数 **colorbar** 在当前的图形窗口中增加水平或垂直的颜色标尺以显示当前坐标轴的颜色映象。» **colorbar('horiz')** 在当前的图形下面放一个水平的颜色条。» **colorbar('vert')** 在当前的图形右边放一个垂直的颜色条。对无参量的 **colorbar**，如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条。下面的例子就演示了 **colorbar** 的用法。

```
» [x,y,z]=peaks;
» mesh(x,y,z);
» colormap(hsv)
» axis([-3   3   -3   3   -6   8])
» colorbar
```

输出见图 19.3.

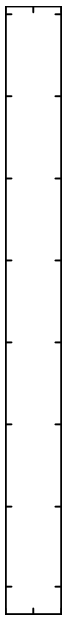


图 19.3　使用颜色条

19.4 颜色映象的建立和修改

颜色映象就是矩阵，意味着你可以象其它数组那样对它们进行操作。函数 **brighten** 就利用这一点通过调整一个给定的颜色映象来增加或减少暗色的强度。**brighten(n)**（**0<1<=1)**使当前颜色映象变亮；而 **brighten(n)**（**-1<=n<0**）使它变暗。**brighten(n)**后加一个 **brighten(-n)**使颜色映象恢复原来状态。**newmap=brighten(n)** 命令创建一个比当前颜色映象更暗或者更亮的新的颜色映象，而并不改变当前的颜色映象。命令**» newmap=brighten(cmap,n)**对指定的颜色映象创建一个已调整过的式样，而不影响当前的颜色映象或指定的颜色映象 **cmap**。

可以通过生成 m×3 的矩阵 **mamap** 来建立用户自己的颜色映象，并用 **colormap(mymap)** 来安装它。颜色映象矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小比 1 大的任意值，函数 **colormap** 会提示一个错误然后退出。

也可以在算术上来组合颜色映象，虽然结果有时是不可预料的。比如，一个叫 **pink** 的颜色映象仅仅是：

```
» pinkmap=squr(2/3*gray+1/3*hot);
```

只当所有元素都在 0 与 1 之间时，才能保证结果是一个有效的颜色映象。精通 MATLAB 工具箱中包含了一个名叫 **rainbow** 的颜色映象，它把可视范围扩展到整个颜色映象。函数 **rainbow** 的在线帮助为：

```
» help    rainbow
RAINBOW Colormap varient to HSV.
```

RAINBOW(M) Rainbow Colormap with M entries.
Red-Orange-Yellow-Green-Blue-Violet
RAINBOW by itself is the same length as the current colormap.
Apply using :colormap(rainbow)

帮助信息：

RAINBOW　HSV 颜色映象的变形
RAINBOW(M)　有 M 个入口项的 RAINBOW 颜色映象
　　红—橘黄—黄—绿—蓝—天蓝
RAINBOW 本身和当前颜色映象的长度相同
应用：colormap(rainbow)

精通 MATLAB 工具箱中还包含了一个名叫 **mmap** 的函数，它可以根据你所提供的颜色建立一个单色（比如**粉红**、**灰色**或**铜黄色**）的颜色映象。函数 **mmap** 的在线帮助是：

» help　mmap
MMAP Single Color Colormap.
MMAP(C,M) makes a colormap of length M starting with the basic colorspec C.The map changes from dark to light.
MMAP(C) is the same length as current colormap.
Examples:mmap('y') is a yellow colormap.
　　mmap([.49　1　.83]) is an aquamarine colormap.
　　mmap('c' ,20) is a cyan colormap having length 20.

帮助信息：

MMAP　单色颜色映象
MMAP(C,M)　制作一个以颜色 C 为基色的长度为 M 的颜色映象。该表的颜色从暗到明变化。
MMAP(C)　颜色映象的长度和当前颜色映象相同
例子：mmap('y')是一个黄色颜色映象
　　mmap([.49　1　.83])是一个水色的颜色映象
　　mmap('c' ,20)是一个长度为 20 的青蓝色的颜色映象
应用：colormap(mmap(c,m))

一个颜色映象定义了用于绘制图形的调色板。一个缺省的颜色映象允许对数据使用 64 种不同的 **RGB** 值。MATLAB 使用函数 **cxis** 来决定哪一个数据值映射到颜色映象中输入项。

通常，颜色映象进行过调节，把数据从最小扩展到最大，也就是说整个颜色映象都用于绘图。有时也许想改变颜色使用的方法。函数 **caxis** 代表颜色轴，因为颜色增加了另一个维数，它允许对数据范围的一个子集使用整个颜色映象或者对数据的整个集合只使用当前颜色映象的一部分。

[cmin,cmax]=caxis 返回映射到颜色映象中第一和最后输入项的最小和最大的数据。它们通常被设成数据的最小值和最大值。比如，函数 **mesh(peaks)** 会画出函数 **peaks** 的网格图，并把颜色轴 **caxis** 设为**[-6.5466，8.0752]**，即 **Z** 的最小值和最大值。这些值之间的数据点，使用从颜色映象中经插值得到的颜色。

caxis([cmin，cmax]）对 **cmin** 和 **cmax** 范围区内的数据使用整个颜色映象。比 **cmax** 大的数据点用与 **cmax** 值相关的颜色绘图，比 **cmin** 小的数据点的颜色用与 **cmin** 值相关的颜色绘图。如果 **cmin** 小于 **min(data)**和/或 **cmax** 大于 **max(data)**，那么与 **cmin** 和/或 **cmax** 点相关的颜色将永远用不到。也就是说，只用到和数据相关的那一部分颜色映象。» **caxis(** ‘**auto**’ **)**设置 **cmin** 和 **cmax** 的缺省值。

由于下面的例子很难在书中清晰区分灰度，运行脚本 M 文件 **mmcaxisd.m** 可显示所包含的一系列更多的例子。缺省的颜色范围由下例说明：

```
» pcolor([1:17;1:17]' ),colormap(hsv(8))
» title( 'Default Color Range' )
» caxis( 'auto' )
» colorbar
» caxis
ans =
      1      17
```

输出见图 19.4.

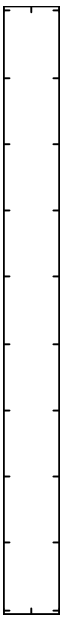


图 19.4　缺省的颜色范围

可见对整个数据集合，当前颜色映象使用了所有 8 种颜色。每种颜色有两条。如果颜色被映射到从-3 到 23 的数据，那么，图中只用到五种颜色。这可以通过下面的命令实现：

```
» title( ‘Extended Color Range’ )
» caxis([-3,23])    %    extended the color range
» colorbar    %    redraw the color scale
```

输出见图 19.5.

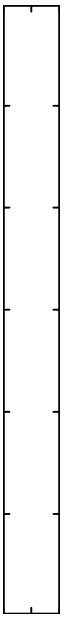


图 19.5　扩展的颜色范围

如果颜色映射到从 5 到 12 的值，会用到所有的颜色。但是，比 5 小的数据和比 12 大的数据分别映射到与数据值 5 和 12 相关的颜色。这可以通过下面的命令产生：

```
» title( ‘Restricted Color Range’ )
» caxis([5,12])    %    resstrict the color range
» colorbar    %    redraw the color scale
```

输出见图 19.6.

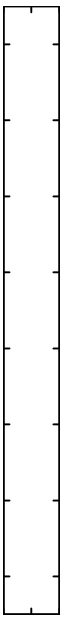


图 19.6　受限的颜色范围

19.5 图形中使用一个以上的颜色映象

有时，在一幅图的不同部分使用不同的颜色是很有作用的。由于颜色映象是图形窗口本身的一个属性，在任意一个图形窗口中，只能用一个颜色映象。但是，可以创建自己的颜色映象来达到想要的效果。例如，精通 MATLAB 工具箱中含有脚本 M 文件 **mmcmapd.m**，它执行下述操作。

```
» figure    %    create a figure window
» mymap=[rainbow(32);copper(32)];    %    stack two color maps into one
» colormap(mymap)    %    install it
» mesh(peaks+8);view(0,0);    %    create two sample polots
» hold on ;mesh(peaks-8);
» colorbar    %    and add a color scale
» title( 'Merging two colormaps' )
» hold off
```

输出见图 19.7.

图 19.7　合并两个颜色映象

19.6 用颜色描述第四维

一些函数，比如 **mesh** 和 **serf**，除非给出颜色参量，颜色将沿 z 轴数据变化。比如，**surf(X，Y，Z**）等效于 **surf(X，Y，Z，Z**）。将颜色施加于 z 轴能够产生色彩漂亮的图画，但由于 z 轴已经存在，它并不提供新的信息。为更好的利用颜色，建议用颜色来描述不受三个轴影响的数据的某些属性。为此需要赋给三维作图函数的颜色参量不同的数据。

如果作图函数的颜色参量是一个向量或矩阵，那么就用作颜色映象的下标。这个参量可以是任何实向量或与其参量维数相同的矩阵。考虑下面这些例子：

```
» x=-7.5: .5 : 7.5; y=x;    %    create a data set - the frame scmbrero
» [X Y]=meshgrid(x,y);    %    create placid data
» R=sqrt(X.^2+Y.^2)+eps;
» Z=sin(R)./R
» surf(X,Y,Z,Z)    %    default color order
» surf(X,Y,Z,-Z)    %    reverse the default color order
» surf(X,Y,Z,X)    %    color varies along the X-axis
» surf(X,Y,Z,X+Y)    %    color varies along the XY diagonal
» surf(X,Y,Z,R)    %    color varies radially from the center
» surf(X,Y,Z,abs(del2(Z)))    %    color varies with absolute value of Laplacian
» [dZdx,dZdy]=gradient(Z);    %    compute gradient or slope of surface
» surf(X,Y,Z,abs(dZdx))    %    color varies with absolute slope in x-direction
» surf(X,Y,Z,abs(dZdy))    % color varies with absolute slope in y-direction
» dz=sqrt(dZdx.^2+dZdy.^2);
» surf(X,Y,Z,dZ)    %    color varies with magnitude of slope
```

输出分别见图 19.8、图 19.9、图 19.10、图 19.11、图 19.12、图 19.13 和图 19.14.

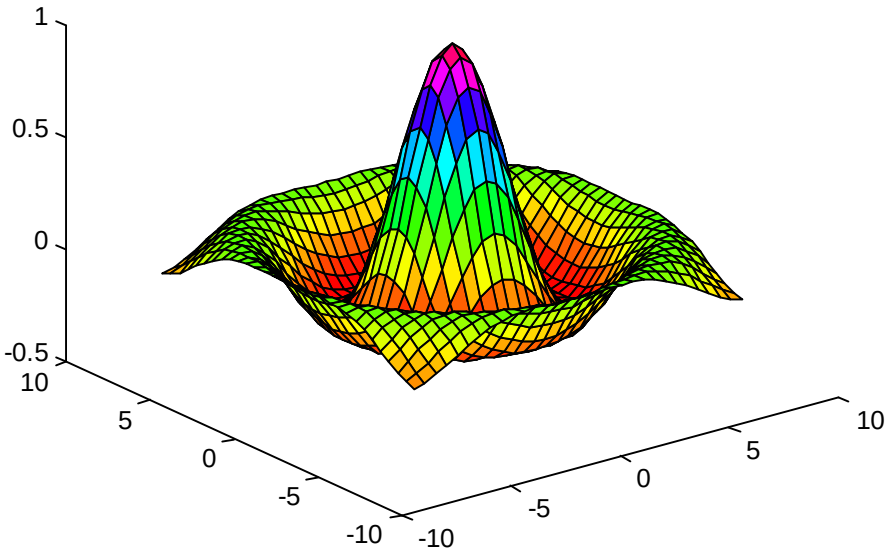


图 19.8　surf(X,Y,Z,Z)

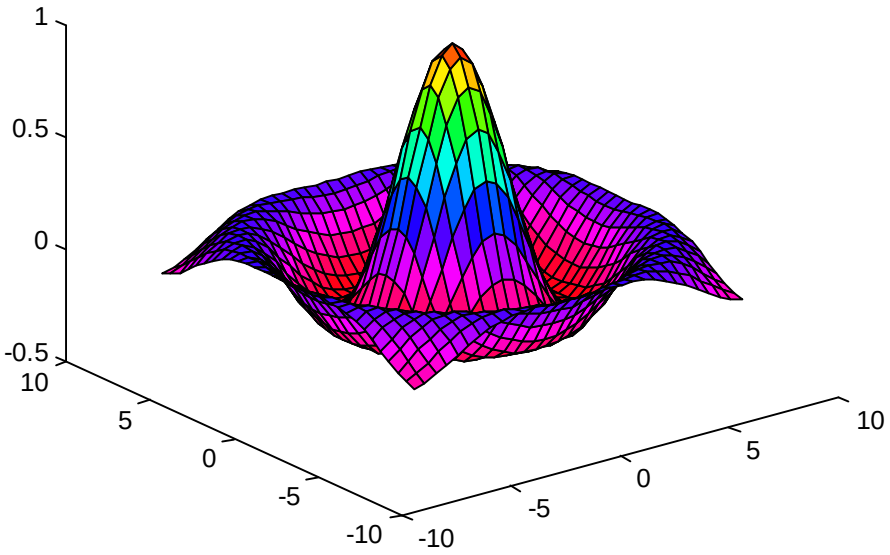


图 19.9　surf(X,Y,Z,-Z)

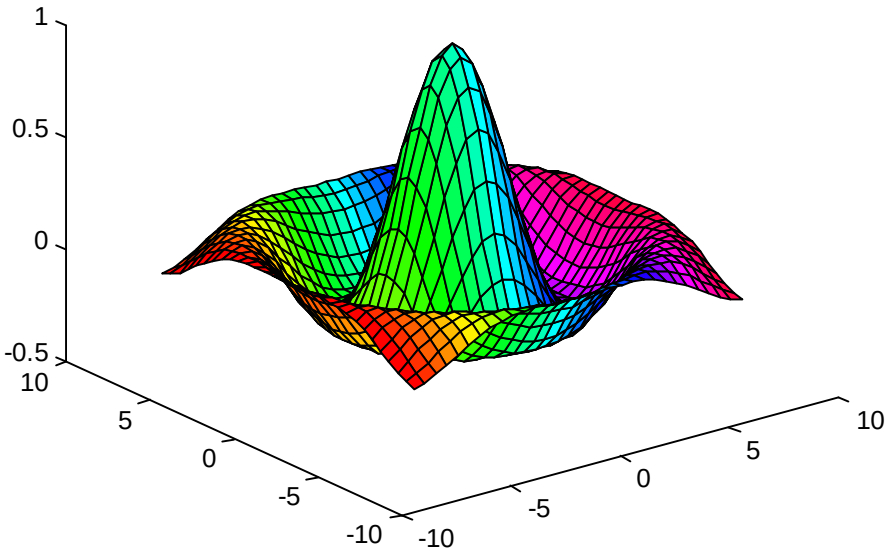


图 19.10　surf(X,Y,Z,X)

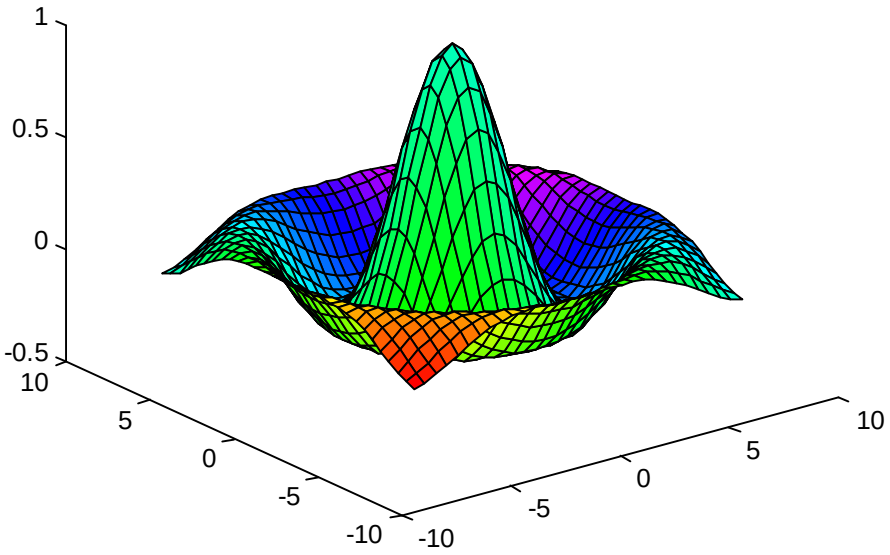


图 19.11　surf(X,Y,Z,X+Y)

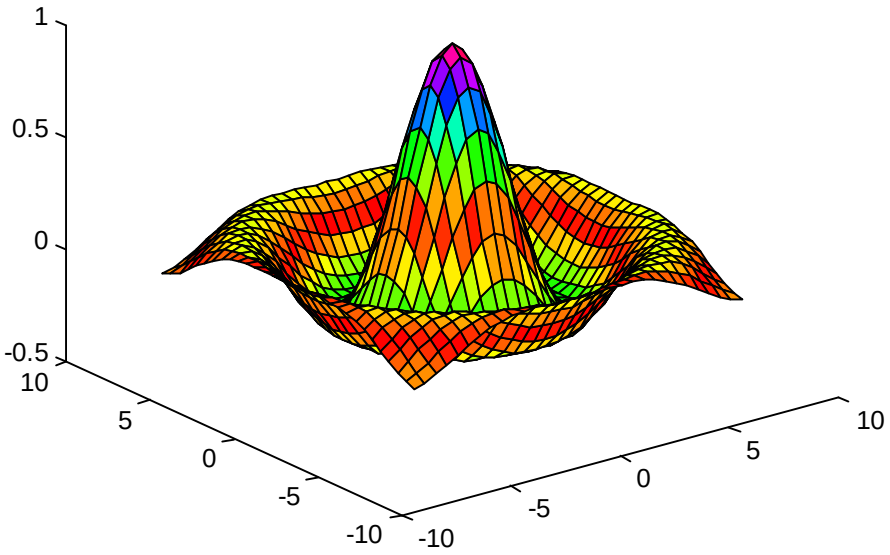


图 19.12　surf(X,Y,Z,R)

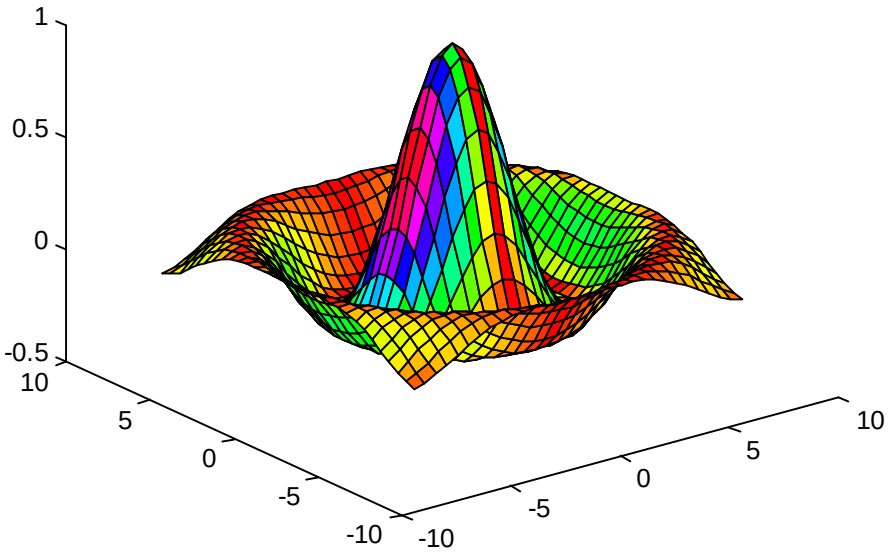


图 19.13　surf(X,Y,Z,abs(del2(Z)))

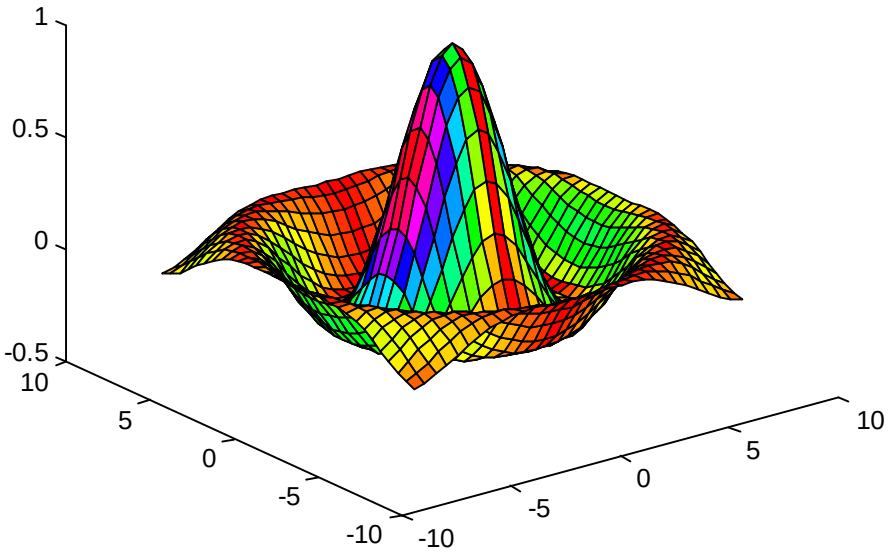


图 19.14　surf(X,Y,Z,dZ)

图 19.15　surf(X,Y,Z,abs(dZdx))

图 19.16　surf(X,Y,Z,abs(dZdy))

注意到上面后五个例子中，颜色如何为所画的曲面提供了一个附加的维数。函数 **del2** 是离散拉氏函数，它根据表面弯曲程度来使用颜色。函数 **del2** 描述如下：

```
» help del2
DEL2 Five-point discrete Laplacian.
    V=del2(U) is a matrix the same size as U with each element equal to the difference
    between an element of U and the average of its four neighbours.For the “corners” and
    “edges”,only two or three neighbours are used.
    See also GRADIENT,DIFF
```

帮助信息：

DEL2　五点的离散 Laplacian

V=del2(U)是一个和 U 同样大小的矩阵。它的每个元素是 U 中的对应位置的元素和它 的四个相邻点元素的平均值的差值。对于角上和边上的元素，只使用两个或三个相邻点。

参阅函数：GRADIENT ,DIFF

如上述，函数 **gradient** 逼近表面的梯度或坡度。为了方便，通过运行精通 MATLAB 工具箱中的脚本 M 文件 **mm4d**，便可执行上述命令。

19.7 照明模型

基于运用漫射、镜面反光和环境照明模型，函数 **surfl** 画出了一个类似于函数 **surf** 产生的带彩色的曲面。使用一个单色颜色映象（如**灰色**，**纯白**，**铜黄**或**粉红色**）和插值色彩，会画出效果最好的曲面。

正常的参量表为 **surfl（X，Y，Z，S）**，这里 **X**，**Y** 和 **Z** 与 **surf(X，Y，Z）**相同。而 **S** 以**[Sx，Sy，Sz]**或**[az , el]**的形式定义了光源的方向。如果没有指明，其缺省光源是逆时针 45 度，即从现在的视角向右转 45 度。

环境照明，漫射反射，镜面反光对视觉效果的相对贡献以及镜面扩展因子可以通 过**K=[ka,kd,ks,spread]**的五个元素来设定，这里 **K** 是函数 **surfl** 的第五个参量，即 **surfl（X，Y，Z，S，K）**。**K** 的缺省值是**[.55　.6　.4　10]**。为了了解这些参量如何影响图形照明，可以参阅下面这些例子。

```
» [X,Y,Z]=peaks(32);　%　data to plot
» surfl(X,Y,Z) , colormap(copper) , title( ‘Default Lighting’ ) , shading interp
» surfl(X,Y,Z,[7.5　30],[.55　.6　.4　10]) , shading interp
» surfl(X,Y,Z,[-90　30],[.55　.6　2　10]) , shading interp
```

如前所述，插值上色会极大地降低打印速度。这是因为每一象素都有一个不同的颜色值，打印机对每点都要分别地上色。

19.8 小结

本章所用的函数总结在表 19.3、表 19.4、表 19.5 和表 19.6 中。

表 19.3

简单颜色			
Red（红）	Green（绿）	Blue（蓝）	颜色
0	0	0	黑
1	1	1	白
1	0	1	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
. 5	0	0	深红
. 5	. 5	. 5	灰色

表 19.4

标准颜色映象	
hsv	色彩饱和值（以红色开始和结束）
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	**hsv** 的一种变形（以蓝色开始和结束）
copper	线性铜色度
prim	三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色
fag	交替为红色、白色、蓝色和黑色

表 19.5

在 surf，mesh 和 pcolor 图中作第四维的颜色	
surf(X,Y,Z,fun(X,Y,Z))	根据函数 **fun(X,Y,Z)**来施加颜色
surf(X,Y,Z)=surf(X,Y,Z,Z)	缺省动作，加颜色于 Z 轴
surf(X,Y,Z,X)	加颜色于 X 轴
surf(X,Y,Z,Y)	加颜色于 Y 轴
surf(X,Y,Z,X.^2+Y.^2)	根据 **z=0** 平面距原点（**x=0，y=0**）的距离施加颜色
surf(X,Y,Z,del2(Z))	根据曲面的拉氏函数值施加颜色

[dZdx,dZdy]=gradient(Z); **surf(X,Y,Z,abs(dZdx))**	根据 X 轴方向的曲面斜率施加颜色
dz=sqrt(dZdx.^2+dZdy.^2); **surf(X,Y,Z,dz)**	根据曲面斜率大小施加颜色

表 19.6

颜色和照明函数	
colormap(map)	在当前的图形窗口中安装一个颜色映象
coorbar	在当前的图形上显示一个水平的或垂直的颜色标尺
rgbplot(map)	颜色映象中红、绿、蓝分量的直线图
brighten(a)	**0<a<1**，当前颜色映象加亮；**-1<a<0**，当前颜色映象加暗
m=brighten(map,a)	返回加亮的颜色映象 **m**
[cmin,cmax]=caxis	返回颜色轴的界限
caxis([cmin,cmax]）	设置颜色轴的界限

关键词索引

chap 19

data visualization 数据可视化

lighting model 照明模型

hardware color lookup table 硬件查色表

entry 输入项

color scale 颜色标尺

color bar 颜色条

dark color 暗色

color axis 颜色轴

discrete Laplacian function 离散拉氏函数
diffuse 漫射
specular 镜面反光
ambient lighting model 环境照明模型
specular-spread coefficient 镜面扩展因子

第二十章　句柄图形

`　什么是句柄图形？句柄图形是对底层图形例程集合的总称，它实际上进行生成图形的工作。这些细节通常隐藏在图形 M 文件的内部，但如果想使用它们也是可得到的。

MATLAB 用户指南给人的一种印象是，句柄图形非常复杂，只对熟练的高级用户才有用。而实际上不是这样的。句柄图形可以被任何人用来改变 MATLAB 生成图形的方式，不论是只想在一幅图里做一点小变动，还是想做影响所有图形输出的全局变动。

句柄图形允许你定制图形的许多特性，而这用高级命令和前几章里描述的函数是无法实现的。例如，如果想用橘黄色来画一条线，而不是 **plot** 命令中可用的任何一种颜色，该怎么做呢？句柄图形就可以提供一种方法。

本章不对句柄图形作详细讨论，因为那样涉及问题太细。这里的目的只是对句柄图形概念作基本了解，并提供足够多的信息，使得即使是偶尔使用一下 MATLAB 的用户也可以利用句柄图形。在这个背景下，在本章最后给出了关于句柄图形对象属性和它们的值，它不仅很有用也很有意义。

20.1 谁需要句柄图形？

开始，我们要强调本章主要是针对那些不满足于 MATLAB 普通图形特性的读者。如果对所画的图形已经很满意，那么就跳过当前的讨论。如果以后要定制图形，只要记住这里有可用的信息。

现在，对于那些还在犹豫的用户，我们要强调学习使用句柄图形并不困难。如果只想改变图形的标题字体，或者改变一个图形窗口的背景颜色，那么，你不必成为一个句柄图形的专家也可做到。

另一方面，如果想定制图形，并且要打算对图形的每个可能方面进行控制，那么句柄图形会为此提供强有力的工具。

前面那些章提供的图形功能被认为是高级的命令和函数，包括 **plot**，**mesh**，**axis** 及其它。这些函数是建立在底层函数和属性的基础上，总称为句柄图形。

20.2 什么是句柄图形对象

句柄图形是基于这样的概念，即一幅图的每一组成部分是一个对象，每一个对象有一系列句柄和它相关，每一个对象有按需要可以改变的属性。

当今计算机行业最流行的术语之一便是对象这个词。面向对象的编程语言，数据库对象，

操作系统和应用程序接口都使用了对象的概念。一个对象可以被粗略地定义为由一组紧密相关、形成唯一整体的数据结构或函数集合。**在 MATLAB 中，图形对象是一幅图中很独特的成分，它可以被单独地操作**。

由图形命令产生的每一件东西都是图形对象。它们包括图形窗口或仅仅说是图形，还有坐标轴、线条、曲面、文本和其它。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其它对象的父亲。图形窗口是根对象的子对象；坐标轴和用户界面对象（在下一章讨论）是图形窗口的子对象；线条、文本、曲面、补片和图象对象是坐标轴对象的子对象。这种层次关系在图 20-1 中给出。

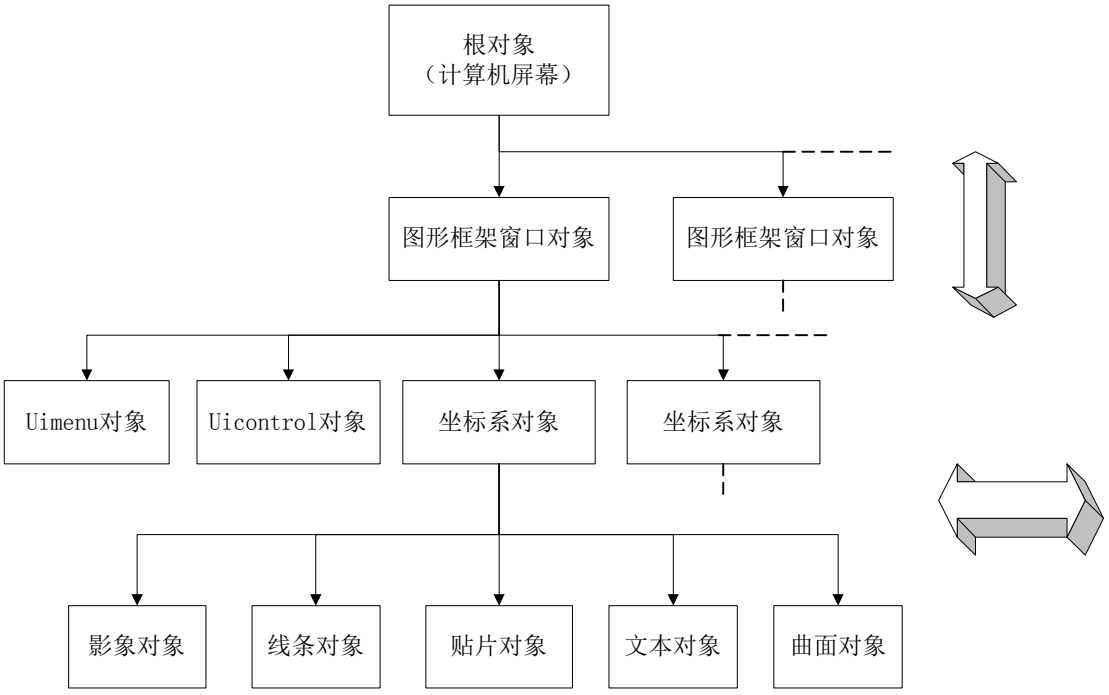


图 20-1　对象层次结构

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其它的对象（除了在下一章讨论的 *uicontrol* 和 *uimenu* 外）都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，如果没有图形窗口，**plot(rand(size([1：10])))**函数会用缺省属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

20.3 句柄对象

假设已打开了三个图形窗口，其中两个有两幅子图。并要改变其中一幅子图坐标轴内一条线的颜色，如何认定想要改变的那条线？在 MATLAB 中，每一个对象都有一个数字来标识，叫做**句柄**。

每次创建一个对象时，就为它建立一个唯一的句柄。计算机屏幕作为根对象常常是 0。**» Hf_fig=figure** 命令建立一个新的图形窗口，变量 **Hf_fig** 中返回它的句柄值。图形窗口的句柄为整数，通常显示在图形窗口标题条中。其它对象句柄是 MATLAB 满精度的浮点值。

MATLAB 可以用来获得图形、坐标轴和其它对象的句柄。例如，» **Hf_fig=gcf** 返回当前图形窗口的句柄值，而» **Ha_ax=gca** 返回当前图形窗口内当前坐标轴的句柄值。这些函数和其它对象操作的工具在本章以后讨论。

为了提高可读性，在本书中包含句柄对象的变量取名以大写的 **H** 开头，跟之以一个辨识对象类型的字母，然后是一个下划线，最后是一个或几个描述符。因此，**Hf_fig** 是一个图形窗口的句柄，**Ha_ax1** 是坐标轴对象的句柄，而 **Ht_title** 是一个文本对象的句柄。当对象类型不知道时，用字母 **x**，比如 **Hx_obj**。虽然句柄变量可以取任意名字，遵循这种规则使得能在 M 文件中很容易找到句柄变量。

所有产生对象的 MATLAB 函数都为所建立的每个对象返回一个句柄（或句柄的列向量）。这些函数包括 **plot**，**mesh**，**surf** 及其它。有一些图形由一个以上对象组成。比如，一个**网格图**由一个曲面组成，它只有一个句柄；而 **waterfall** 图形由许多线条对象组成，每个线条对象都有各自的句柄。例如，» **Hl_wfall=waterfall(peaks(20))** 对线条返回一个包含着 20 个句柄的列向量。

20.4 通用函数 get 和 set

所有对象都有**属性**来定义它们的特征，正是通过设定这些属性来修正图形显示的方式。尽管许多属性所有的对象都有，但与每一种对象类型（比如坐标轴，线，曲面）相关的属性列表都是独一无二的。对象属性可包括诸如对象的位置、颜色、类型、父对象、子对象及其它内容。每一个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。和每一种对象类型(图形，坐标轴，线，文本，曲面，补片和图象)相关的完整的属性列表在本章的后面给出。

对象属性包括属性名和与它们相联关的值。属性名是字符串，它们通常按混合格式显示，每个词的开头字母大写，比如：' **LineStyle** ' 。但是，MATLAB 识别一个属性时是不分大小写的。另外，只要用足够多的字符来唯一地辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用 ' **Position** ' ， ' **position** ' ，甚至是 ' **pos** ' 来调用。

当建立一个对象时，它用一组缺省属性值，该值可以用两种方法来改变。可以用{**属性名，属性值**}对来建立对象生成函数；或者在对象建立后改变属性。前一种方法的例子是：

```
» Hf_1=figure( 'color', 'white' )
```

它用缺省的属性值建立一个新的图形窗口，只是背景颜色被设为白色而不是缺省的黑色。

为了获得和改变句柄图形对象的属性只需要两个函数。函数 **get** 返回某些对象属性的当前值。使用函数 **get** 的最简单语法是 **get(handle, ' PropertyName')**。例如：

```
» p=get(Hf_1, 'position' )
```

返回具有句柄 **Hf_1** 图形窗口的位置向量。

```
» c=get(Hl_a, 'color' )
```

返回具有句柄 **Hl_a** 对象的颜色。

函数 **set** 改变句柄图形对象属性，使用语法 **set(handle,** '**PropertyName**' **,value)**。例如：

» set(Hf_1, 'Position' ,p_vect)

将具有句柄 Hf_1 的图形位置设为向量 **p_vect** 所指定的值。同样

» set(Hl_a, 'color' , 'r')

将具有句柄 **Hl_a** 的对象的颜色设置成红色。

一般情况下，函数 **set** 可以有任意数目的**(** '**PropertyName**' **,PropertyValue)**对。比如：

» set(Hl_a, 'Color' , 'r' , 'Linewidth' ,2, 'LinStyle' , '--' ）

将具有句柄 **Hl_a** 的线条变成红色，线宽为 2 点，线型为破折号。

除了这些主要功能，函数 **set** 和函数 **get** 还能提供帮助。例如» **set(handle,** '**PropertyName**' **)**返回一个可赋给由 **handle** 所描述对象的属性值列表。例如：

» set(Hf_1, 'Units')
[inches|centimeters|normalized|points|{pixels}]

表明由 **Hf_1** 所引用的图形的 '**Unites**' 属性是五个可允许的字符串，而其中 '**pixels**' 是缺省值。

如果指定一个没有固定值的属性，那么，MATLAB 就会通知如下：

» set(Hf_1, 'Position')
A figure's 'Position' property does not have a fixed set of property values。

除了 **set** 命令，句柄图形对象创建函数（例如 **figure**，**axis**，**line** 等等）接受多个属性名和属性值对。例如：

» figure('Color' , 'blue' , 'NumberTitle' , 'off' , 'Name' , 'My Figure' ）

创建一个图形窗口，背景为兰色，标有 **'My Figure'** 而不是缺省标题 **'Figure No. 1'**。

为了形象说明上述概念，考虑下面的例子：

```
» Hf_fig =figure    %    create a figure having an interger handle
Hf_fig=
        1
» Hl_line=line    %    create a line having a floating-pointer handle
Hl_line =
    59.0002
» set(Hl_line);    %    list settable properties and potential values
      Color
```

```
    EraseMode: [ {normal} | background | xor | none ]
    LineStyle: [  {-}  |  --  |  :  |  -.  |  +  |  o  |  *  |  .  |  x  ]
    LineWidth
    MarkerSize
    Xdata
    Ydata
    Zdata

    ButtonDownFcn
    Clipping: [ {on} | off ]
    Interruptible: [ {no} | yes ]
    Parent
    UserData
    Visible: [ {on} | off ]

» get(Hl_line);   %   list properties and current property values
Color = [1 1 1]
    EraseMode = normal
    LineStyle = -
    LineWidth = [0.5]
    MarkerSize = [6]
    Xdata = [0 1]
    Ydata = [0 1]
    Zdata = [ ]

    ButtonDownFcn =
    Children = [ ]
    Clipping = on
    Interruptible = no
    Parent = [58.0002]
    Type = line
    UserData = [ ]
    Visible = on
```

在上例中，所创建的线条中的 '**Parent**' 属性就是包含线条的坐标轴的句柄。而且所显示的图形列表被分为两组。在空行上的第一组，列出了该对象的独有属性，而空行下的第二组列出所有的对象共有的属性。注意到函数 **set** 和函数 **get** 返回不同的属性列表。函数 **set** 只列出可以用 set 命令改变的属性，而 **get** 命令列出所有对象的属性。在上面的例子中，函数 **get** 列出了 '**Children**' 和 '**Type**' 属性，而 **set** 命令却没有。这一类属性只可读，但不能被改变，它们叫做只读属性。

与每一个对象有关的属性数目是固定的，但不同的对象类型有不同数目的属性。象上面所显示的，一个线条对象列出了 16 个属性，而一个坐标轴对象列出了 64 个属性。显然，透彻地说明和描述所有对象类型的全部属性超出本书的范围。但是，其中的很多属性本书以后要详细讨论，并且列出全部属性。

作为一个使用图象句柄的例子，可以考虑前面提出的问题。它要用非标准颜色画一条线。在这里，线的颜色用 **RGB** 值**[1 .5 0]**来指定，它是适中的橘黄色。

```
» x=-2*pi:pi/40:2*pi;  %  create data
» y=sin(x);  %  find the sine of x
» Hl_sin=plot(x,y)  %  plot sine and save line handle
Hl_sin=
    59.0002
» set(Hl_sin, 'Color' ,[1 .5 0], 'LineWidth' ,3)  %  Change the color and width
```

现在加一个浅蓝色的 cosine 曲线：

```
» z=cos(x);  %  find the cosine of x
» hold on  %  keep the sine curve
» Hl_cos=plot(x,z);  %  plot the cosine and save the handle
» set(Hl_cos, 'Color' ,[.75  .75  1])  %  color it light blue
» hold off
```

也可以用较少的步骤来实现同样的功能：

```
» Hl_lines=plot(x,y,x,z);  %  plot both curves and save both handles
» set(Hl_line(1), 'Color' ,[1 .5 0], 'LineWidth' ,3)
» set(Hl_line(2), 'Color' ,[.75 .75 1])
```

如何加上一个标题并且使字体比正常大一些呢？

```
» title( 'Handle Graphics Example' )  %  add a title
» Ht_text=get(gca, 'Title' )  %  get a handle to the title
» set(Ht_text, 'FontSize' ,16)  %  customize the font size
```

最后一个例子说明了关于坐标轴对象令人感兴趣的性质。每一个对象都含有 '**Parent**' 属性和 '**Children**' 属性，该属性包含属于派生对象的句柄。画在一组坐标轴上的线，具有当作 '**Parent**' 属性值的坐标轴对象的句柄，而 '**Children**' 属性值是一个空距阵。同时，这个坐标轴对象具有当作 '**Parent**' 属性值的图形句柄，而 '**Children**' 属性值是线条对象的句柄。标题字符串和坐标轴的标志不包含在坐标轴的 '**Children**' 属性值里，而是保存在 '**Title**' 、 '**Xlabel**' 、 '**Ylabel**' 和 '**Zlabel**' 的属性内。创建坐标轴对象时，这些文本对象就建立。**title** 命令设置当前坐标轴内标题文本对象的 '**String**' 属性。最终，标准 MATLAB 的函数 **title,xlabel,ylabel** 和 **zlabel** 不返回句柄，而只接受属性和数值参量。例如下面的命令给当前图加一个 24 点的绿色标题：

```
» title( 'This  is  a  title.' , 'Fontsize' ,24, 'Color' , 'green' )
```

除了函数 **set** 和 **get**，MATLAB 还提供了另外两个函数来操作对象和它们的属性。任意对象和它们的子对象可以用 **» delete(handle)** 来删除。同样 **» reset(handle)** 将与句柄有关的

全部对象属性（除了‘**Position**’属性）重新设置为该对象类型的缺省值。

20.5 查找对象

正如你所见，句柄图形提供了对图形对象的访问途径，并且允许用函数 **get** 和 **set** 定制图形。但是，如果忘记保存句柄或图中对象的句柄将会怎样呢？或者，也许变量被覆盖又如何呢？如果不知道它们的句柄，怎么改变对象的属性呢？为解决这个问题，MATLAB 提供了查找对象句柄的工具。

gcf 和 **gca** 这两种工具前面已经介绍过了。

```
» Hf_fig=gcf
```

返回当前图形的句柄，而

```
» Ha_ax=gca
```

返回当前图形的当前坐标轴的句柄。

除了这些，还有一个获取当前对象的 **gco**。

```
» Hx_obj=gco
```

返回当前图形的当前对象的句柄。或者用另一种方法：

```
» Hx_obj=gco(Hf_fig)
```

返回与句柄 **Hf_fig** 有关的图形中当前对象的句柄。

当前对象的定义为用鼠标刚刚点过的对象。这种对象可以是除根对象（计算机屏幕）之外的任何图形对象。但是，如果鼠标指针处在一个图形中而鼠标按钮未点，**gco** 返回一个空距阵。为了让当前对象存在，我们必须选择一些东西。

一旦获得了一个对象的句柄，它的对象类型可以通过查询对象的‘**Type**’属性来获得。该属性是一个字符串对象名，比如‘**figure**’，’**axes**’或‘**text**’。例如：

```
» x_type=get(Hx_obj, ‘Type’ )
```

MATLAB 中的函数 **gcf**，**gca** 和 **gco** 是很好的例子，它们说明如何利用句柄图形来获得有关对象的信息。函数 **gcf** 获得根对象的‘**CurrentFigure**’的属性值，即是当前图形的句柄。**gcf** M 文件包含：

```
function h=gcf()
%   GCF    Get current figure handle.
%   H=GCF returns the handle to the current figure.The current fugure is the figure(graphics window)that graphics commands like PLOT,TITLE,SURF,etc.draw to if issued.
%
```

```
%    Use the commands FIGURE to change the current figure to a diffent figure,or to create
new      %    ones.
%
%    See also FIGURE,CLOSE,CLF,GCA.
%    Copyright    (c)    1984-94 by The MathWorks,Inc.

h=get(0, 'CurrentFigure' );
```

类似的，函数 **gca** 返回当前图形的 '**CurrentAxes**' 属性值，它的 M 文件描述如下。

```
function h=gca()
%    GCA    Get current axis handle.
%     H=GCA returns the handle to the current axis.The current axis is the axis that graphics
%    command like PLOT,TITLE,SURF,etc.draw to if issued.
%
%    Use the commands AXES or SUBPLOT to change the current axis to a different axis,or to
%    create new ones.
%    see also AXES,SUBPLOT,DELETE,CLA,HOLD,GCF.
%    Copyright    (c)    1984-94 by The MathWorks,Inc.
h=get(get(0, 'CurrentFigure' ), 'CurrentAxes' );
```

函数 **gco** 也相同，只是它在试图获得当前对象之前先检查图形是否存在。**注意函数 gcf 和 gca 能促使建立相关的对象，如果它们不存在的话**。如下所示的函数 **gco**，它先检查子对象（ '**Children**' ）是否存在，如果不存在，就不创建图形对象。

```
function object=gco(figure)
%GCO    Handle of current object.
%    OBJECT=GCO returns the current object in the current figure.
%
%    OBJECT=GCO(FIGURE) returns the current object in figure FIGURE.
%
%    The current object for a given figure is the last object clicked on with mouse.

%Copyright    (c)    1984-94 by The MathWorks,Inc.

if isempty(get(0, 'Children' ))
      object=[ ];
      return;
end;

if (nargin==0)
      figure=get(0, 'CurrentFigure' );
end
```

```
object=get(figure, 'CurrentObject' );
```

当需要一些除了 '**CurrentFigure**' 、 '**CurrentAxes**' 和 '**CurrentObject**' 之外的某些东西时，可以用函数 **get** 来获得一个对象的子对象的句柄向量。例如：

```
» Hx_kids=get(gcf, 'Children' )
```

返回一个向量，它包含当前图形子对象的句柄。

可以用获得子对象 '**Children**' 句柄的技术彻底搜索句柄图形的层次结构中来找到所要的对象。例如，在画出一些数据后，寻找绿色线条句柄的问题。

```
» x=-pi:pi/20:pi;    %    create some data
» y=sin(x);z=cos(x);
» plot(x,y, 'r' ,x,z, 'g' );    %    plot two lines in red and green
» Hl_lines=get(gca, 'Children' );    %    get the line handles
» for k=1:size(Hl_lines)    %    find the green line
»if get(Hl_lines(k), 'Color' )==[0 1 0]
»              Hl_green=Hl_lines(k)
»       end
» end
Hl_green=
      58.0001
```

尽管这种技术有效，但是如果有很多对象存在就变得复杂。该技术也丢失了标题和坐标轴标志中的文本对象，除非能逐个检测这些对象。

当有多个图形，每个图形上又有多个坐标轴时，考虑查找所有绿色线条的句柄的问题。

```
» Hf_all=get(0, 'Children' );    %    get all figure handles
» for k=1:length(Hf_all)
»      Ha_all=[Ha_all;get(Hf_all(k), 'Children' )];    %    get all axes handles
» end
» for k=1:length(Ha_all)
»      Hx_all=[Hx_all;get(Ha_all(k), 'Children' )];    %    get axes child handles
» end
» for k=1:length(Hx_all)
»      if get(Hx_all(k), 'Type' )== 'line'
»          Hl_all=[Hl_all;Hx_all(k)];    %    get line handles only
»      end
» end
» for    k=1:length(Hl_all)
»      if get(Hl_all(k), 'Color' )==[0 1 0]
»          Hl_green=[Hl_green;Hl_all(k)];    %    find green ones
»      end
» end
```

为了简化查找对象句柄的过程，MATLAB 4.2 版本和以后的版本提供了内置函数 **findobj**，该函数返回有指定属性值的所有对象句柄。它的在线帮助如下：

» help findobj

FINDOBJ Find objects with specified property values.

H=FINDOBJ('PlName' ,PlValue,...) returns the handle of the objects at the root level and below whose property values match those as param-value pairs to the FINDOBJ command.

H=FINDOBJ(ObjectHandles, 'PlName' ,PlValue,...) restricts the search to the objects listed in ObjectHandle and their descendents.

H=FINDOBJ(ObjectHandles, 'flat' , 'PlName' ,PlValue,...) restrics the search only to the objects listed in ObjectHandles.Their descendents are not searched.

H=FINDOBJ returns the handles of the root object and all its descendents.

H=FINDOBJ(ObjectHandles) returns the handles listed in Objecthandles ,and the handles of all their descendents.

See also SET,GET,GCF,GCA.

帮助信息：

FINDOBJ 寻找具有指定的属性值的对象

H=FINDOBJ('P1Name' ， P1Value，…）返回根部和根部以下那些属性值与 FINDOBJ 参数对相匹配的对象句柄。

H=FINDOBJ(ObjectHandles， 'P1Name' ， P1Value，…）搜索限定在 ObjectHandles 中列出的对象和它们的子对象。

H=FINDOBJ(ObjectHandles， 'flat' ， 'P1Name' ，P1Value，…）搜索限定在 ObjectHandles 中列出的对象，不搜索它们的子对象。

H=FINDOBJ 返回根对象和它所有的子对象的句柄。

H=FINDOBJ(ObjectHandles）返回 ObjectHandles 中列出的对象和它们的子对象的句柄。

参阅 SET，GET，GCF，GCA。

函数 **findobj** 返回符合所选判据的对象的句柄。它检查所有的 '**Children**' ，包括坐标轴的标题和标志。如果没有对象满足指定的判据，**findobj** 返回空距阵。

用函数 **findobj**，前面的例子变成一行：

» Hl_green=findobj(0, 'Type' , 'line' , 'Color' ,[0 1 0]);

20.6 用鼠标选择对象

gco 命令返回当前对象的句柄，该对象就是用鼠标刚刚点击过的对象。然而，MATLAB 怎么知道哪个对象被选中了呢？例如，当点击一幅图中两条线的交点时，应该返回哪个句柄？或者当鼠标点击时指针离线有多远仍能选中该线？这些答案要根据 MATLAB 选择对象规则和**堆积次序**。

堆积次序决定哪一对象叠加在其它对象上。开始时，堆积次序在对象被创建时就被决定，最后创建的对象在堆栈的顶部。例如，如果发出两条 **figure** 命令，就产生两个图形。第二个图画在第一个的上面。而最终的堆积次序是图 2 在图 1 的上面，当前图形 **gcf** 是图 2。如果发出 **figure(1)**命令，或者点击图 1，堆积次序就改变，图形框架 1 移动到堆栈的顶端，成为当前图形。

在前面的例子中，在计算机屏幕上，堆放次序可以从窗口交叠中显而易见。但是，情况并不总是这样的。如果画了两条线，在它们的交叉点，第二条线在第一条线的上面。如果用鼠标在第一条线其它一些点上点击，那么第一条线条就在栈顶，用鼠标点击交叉点会选中第一条线。**当前的堆积次序是由一个给定的对象 'Children' 句柄出现的次序给出的。也就是说，» Hx_kids=get(handle,'Children')按堆积次序返回 Children 句柄。存储句柄的向量 Hx_kids 的第一个元素在栈顶，而最后一个元素在栈底。**

除了堆积次序，当用鼠标在一个对象附近点击时，该对象也会被选中。每一种对象类型都有与之相关的它自己的选择区域。例如，对于一条线来说，如鼠标的指针在线条的 5 个象素之内，它就会被选中。曲面，补片和文本对象的选择区域是包含对象的最小矩形。坐标轴对象的选择区域是坐标框本身加上标题和标志出现的区域。坐标轴内的对象，例如线条和曲面，在堆积次序中处于较高地位。因此，点击它们时会选择相关的对象而不是坐标轴。选择坐标轴选择区以外的区间就会选中图形本身。

这里讨论的堆积次序和对象选择规则是针对 MATLAB 4.02c 的版本。但要注意，MATLAB 定义、使用堆积次序和对象选择的方法在 5.0 版本中可能会扩充。

20.7 位置和单位

一个图形对象和许多其它句柄图形对象的位置属性 '**Position**' 是一个 4 元素的行向量。在该向量中的值是**[left,bottom,width,height]**。其中**[left,bottom]**是该对象相对于它的父对象的的左下角的位置，而**[width,height]**是该对象的宽度和高度。参阅图 20.2。

图 20.2　Position 属性示意图

这些位置向量中的值的单位是由该对象的单位属性（**Units**）所指定的。例如：

```
» get(gcf, 'Position' )
ans=
        360      544      560      420
» get(gcf, 'Units' )
ans=
    pixels
```

表明了当前图形对象的左下角，相对于屏幕左下角的位置是：向右 360 个象素，向上 544 个象素；且图形对象的宽度为 560 个象素点，高度为 420 个象素点。**要注意，位置向量给出的是图形本身的可画区域，它并不包括该窗口的边界、滚动条或标题条。**

'Units' 属性的缺省值是象素，但它也可以是英寸、厘米、点或归一化坐标。象素代表了屏幕象素，即在屏幕上可表示出来的最小的矩形对象。例如，一个分辨率设置为 800×600 的计算机显示区宽为 800 个象素，高为 600 个象素。点是一种打印设置标准，每一点等于 1/72 英寸。归一化坐标是在 0 到 1 范围内。在归一化坐标中，屏幕的左下角在**[0　0]**，右上角在**[1.0　1.0]**。至于英寸和厘米是不言而喻的。

为了形象说明不同的 **Units** 属性值，重新考虑上面的例子。

```
» set(gcf, 'Units' , 'inches' );
» get(gcf, 'Position' )
ans =
    3.7764    5.7203    5.8907    4.4245

» set(gcf, 'Units' , 'centimeters' )
» get(gcf, 'Position' )
ans =
    9.5847   14.5185   14.9511   11.2297

» set(gcf, 'Units' , 'normalized' )
» get(gcf, 'Position' )
ans =
    271.9001   411.8597   424.1339   318.5655
» set(gcf, 'Units' , 'normalized' )
» get(gcf, 'Position' )
ans =
    0.2805    0.5303    0.4375    0.4102
```

对于特定的显示器，这些值代表了相对于计算机屏幕的相同位置。

坐标轴对象的位置也是四元素向量，具有同样形式**[left,bottom,width,height]**。但它指定的是相对于它的父对象图形左下角的位置。一般来说，一个子对象的‘**Position**‘属性的值是相对于其父对象的位置。

为了更好描述，计算机屏幕或根对象的位置属性不叫做‘**Position**‘，而叫做‘**ScreenSize**‘。这时，**[left,bottom]**总是**[0 0]**，而**[width,height]**是计算机屏幕的尺寸，它的单位由根对象的‘**Units**‘属性指定。

20.8 图形打印

除了将图形放置在计算机屏幕上，MATLAB 还提供了控制打印页上图形位置的属性，以及指定打印纸本身特性的属性。例如，MATLAB 的 **orient** 命令使用函数 **get** 和 **set** 来改变打印纸属性。函数 **orient** 的帮助如下：

» help orient

ORIENT Hardcopy paper orientation.

ORIENT LANDSCAPE causes subsequent PRINT operation from the current figure window to generate output in full-page landscape orientation on the paper.

ORIENT PORTRAIT returns to the default PORTRAIT orientation with the figure window occupying a rectangle with aspect ratio 4/3 in the middle of the page.

ORIENT TALL causes the figure window to map to the whole page in portrait orientation.

ORIENT ,by itself,returns a string containing the current paper orientation,either PORTRAIT,LANDSCAPE or TALL.

ORIENT is an M-file that sets the PaperOrientation and PaperPosition properties of the current figure window.

帮助信息：

ORIENT 硬拷贝纸张反向

ORIENT LANDSCAPE 使得以后从该图形窗口的打印操作时，图形在打印纸上的设置为景象 方向的全幅方式。

ORIENT PORTRAIT 使得以后从该图形窗口的打印操作时，图形在打印纸的中间占据一个肖 像方向 4/3 长宽比的矩形。

ORIENT TALL 使得以后从该图形窗口的打印操作时，图形在在打印纸上的设置为肖像方向 的全幅方式。

ORIENT 不加参数时，返回一个包含当前纸张设置的字符串，值为 LANDSCAPE，PORTRAIT 或 TALL。

ORIENT 是一个 M 文件，它设置当前图形窗口对象中的 PaperOrientation 和 PaperPosition 属性。

参阅 PRINT

影响打印输出的图形属性在表 20.1 中：

表 20.1

图形的打印纸属性	
PaperUnits	**[{inches}|centimeters|normalized|points]**
PaperOrientati-on	**[{portrait}|landscape]**
PaperPosition	位置向量，形式为**[left,bottom,width,height]**。**[left,bottom]**代表了从打印页左下角的偏移，**[width,height]**是图形尺寸
PaperSize	包含纸张尺寸（**[8.5　　11]**）的两元素向量
PaperType	**[{usletter}|uslegal1|a3|a4letter|a5|b4|tabloid]**

‘**PaperPosition**’ 和 ‘**PaperSize**’ 属性的返回值的单位由 ‘**PaperUnits**’ 属性指定。与改变图形对象 ‘**Position**’ 属性以改变屏幕上图形窗口的大小和位置一样，改变 ‘**PaperPosition**’ 属性可以改变图形在打印页上的大小和位置。

考虑下面的例子，说明如何使用纸张属性。

» set(gcf, ‘PaperType’ , ‘a4letter’)

将当前图形打印所用纸张设为 ‘ **a4letter** ‘ 。

» set(gcf, ‘PaperOrientation’ , ‘landscape’)

将当前图形窗口的图的方向设为 ‘ **landscape** ‘ 。

象其它图形属性一样，上面的纸张属性用于单个图形。关于修改所有图形属性在下节“缺省属性”中进行讨论。

20. 9 缺省属性

MATLAB 在建立对象时把缺省属性赋给各对象。如果想不采用这些缺省值，那么，必须使用句柄图形工具对它们进行设置。当每次都要改变同一属性时，MATLAB 允许设置用户自己的缺省属性。MATLAB 让用户改变对象层次结构中任意一点上的单个对象或对象类型的缺省属性。

可以使用一个特殊性质名字符串来设置缺省值，该字符串以 ‘**Default**’ 开头，跟之以对象类型和属性名。使用 **set** 命令中的句柄，确定对象父—子等级图中的点，在该点使用缺省值。例如：

» set(0, ‘DefaultFigureColor’ ,[.5　　.5　　.5])

将所有的**新**图形对象设为适中的灰色，而不是 MATLAB 缺省的黑色。该属性值应用于根对象（它的句柄总是 0），所以所有新图形会有一个灰色的背景。

下面是另外一些可改变缺省值的例子。

```
» set(0, 'DefaultAxesFontSize' ,14)        %    larger axes fonts - all figures
» set(gcf, 'DefaultAxesLineWidth' ,2)    %    thick axes lines - this figure
» set(gcf, 'DefaultAxesColor' , 'y' )        %    yellow X-axis lines and labels
» set(gcf, 'DefaultAxesGrid' , 'on' )        %    Y axis grid lines - this figure
» set(0, 'DefaultAxesBox' , 'on' )            %    enclose axes - all figures
» set(gcf, 'DefaultLineLineStyle' , ':' )    %    dotted linestyle - these axes
```

当应用已存在对象工作时，使用后把它们恢复到初始的状态是一个很好的想法。**如果想在一段例程中改变对象的缺省属性，那么保存原来的设置并在激活例程时将它们恢复**。例如，考虑下面一段函数：

```
oldunits=get(0, 'DefaultFigureUnits' );
set(0, 'DefaultFigureUnits' , 'normalized' );
      <MATLAB statements>
set(0, 'DefaultFigureUnits',oldunits);
return
```

如果在所有的时刻用自己的缺省值来设定 MATLAB，那么只要在 **startup.m** 文件里包括进所需的 **set** 命令就可以了。例如，如果在所有的轴上想要缺省的网格和坐标轴框，并且经常在 A4 纸上打印，就把下面这些行加到 **start.m** 文件中即可。

```
set(0, 'DefaultAxesXGrid' , 'on' )
set(0, 'DefaultAxesYGrid' , 'on' )
set(0, 'DefaultAxesZGrid' , 'on' )
set(0, 'DefaultAxesBox' , 'on' )
set(0, 'DefultFigurePaperType' , 'a4paper' )
```

关于脚本 M 文件 start.m 的更详细的信息，可参阅第二章。

有三个特殊的属性值字符串 '**remove**' 、 '**factory**' 和 '**default**' ，它们逆转、取消或获得用户自定义缺省属性。如果改变了一个缺省属性，可以使用 '**remove**' 逆转这种变化，把它重新设为初始的缺省值。使用 'remove' 说明如下：

```
» set(0, 'DefaultFigureColor' ,[.5 .5 .5])     %    set a new default
» set(0, 'DefaultFigureColor' , 'remove' )    %    return to MATLAB defaults
```

对一特殊对象，为了暂时取消缺省值并用最初的 MATLAB 缺省值，就用特殊的属性值 '**factory**' 。例如：

```
» set(0, 'DefaultFigureColor' ,[.5 .5 .5])    %    set a new user default
» figure( 'Color' , 'factory' )    %    create a new figure using the MATLAB default
```

第三个特殊的属性值字符串是 '**default**' 。这个属性值迫使 MATLAB 搜索对象层次结构，直到查到所需属性的一个缺省值。如果找到，它就使用该缺省值。如果查到根对象，没有找

到用户定义的缺省值，MATLAB 就使用 factory 缺省值。在用不同的属性值创建一个对象后，要把对象缺省设成缺省属性值时，这个概念是很有用的。为了弄清 '**default**' 的使用，考虑下面的例子。

```
» set(0, 'DefaultFigureColor' , 'r' )        %    set the default at the root level
» set(gcf, 'DefaultFigureColor' , 'g' )    %    current figure level default
» set(gca, 'DefaultFigureColor' , 'b' )    %    current axes level default
» Hl_rand=plot(rand(size([1:10])))        %    plot a yellow line
» set(H1_rand, 'Color' , 'default' )        %    the line become blue
» set(gca, 'DefaultFigureColor' , 'remove' )    %    the axes level default is removed
» set(Hl_rand, 'Color' , 'default' )         %    the line become green
» close(gcf)    %    close the window
» Hl_rand=plot(rand(size([1:10])))      %    plot a yellow line in a new window
» set(Hl_rand, 'Color' , 'default' )        %    the line becomes red
```

注意到 **plot** 命令并不对线的颜色设为线条对象的缺省值。如颜色参量未指定，**plot** 命令使用坐标轴 '**ColorOrder**' 属性来指定它所产生的每条线的颜色。

20.10 非文件式属性

用函数 **get** 和 **set** 对每一个对象列出的属性是文件式属性。也有由 MATLAB 开发者所用的非文件式属性。其中一些可以被设置，但另外一些是只读的。

每一对象类型的一个有用的非文件式属性是 '**Tag**' 属性。这个属性用用户自定义的文本字符串来标志一个对象时有用。例如：

```
» set(gca, 'Tag' , 'My    axes' )
```

就把字符串 '**My   axes**' 加到当前图形的当前坐标轴。这个字符串不在图形或坐标轴中上显示出来，但可以查询 '**Tag**' 属性来辨别对象。例如，有许多个坐标轴，可以通过

```
» Ha_myaxes=findobj(0, 'Tag' , 'My    Axes' );
```

来寻找上面的坐标轴对象的句柄。象在下一章讨论的 '**UserData**' 属性一样， '**Tag**' 属性备作专门使用。没有任何 MATLAB 函数和 M 文件改变或对这些属性所含值作出假设。然而，如在下章要讨论的，有一些用户提供的 M 文件和几个精通 *MATLAB* 工具箱函数使用 '**UserData**' 属性来存储临时数据。

由于一些非文件式属性是故意作成非文件式的，所以在使用时必须**非常**小心。它们有时比文件式属性脆弱，并且**常常**引起变化。在以后的 MATLAB 版本中，非文件式属性也许会仍保持或消失，或者改变功能，甚至会成为文件式属性。

在 MATLAB 5.0 中应变成文件式属性的非文件式属性列在表 20.2 中。

表 20.2

属性	对象

'TerminalHideGraphCommand'	根
'TerminalDimensions'	根
'TerminalShowGraphCommand'	根
'Tag'	所有对象
'Layer'	坐标轴
'PaletteModel'	曲面，补片

20.11 M 文件例子

精通 MATLAB 工具箱含有许多实用函数，它们可以验证本章的许多概念。这些函数的基本部分已经在二维和三维图形这些章阐述过了。有了前面对句柄图形的讨论，我们现在可以更彻底地讨论这些函数。

最简单的精通 MATLAB 工具箱的函数之一提出了一个共同的问题。MATLAB 函数 **gcf** 返回当前图形的句柄。但是，它有一个副作用。如果图形不存在，**gcf** 就创建一个，并返回它的句柄。如果想寻找一个图形是否存在于头一个位置，要是没有，又不得不创建，怎么办？函数 **mmgcf** 正好实现由其内容所描述的工作。

```
function HF=mmgcf
%MMGCF Get Current Figure if it Exists.
%    MMGCF returns the handle of the current figure if it exists.
%    If no current figure exists,MMGCF returns an empty handle.
%
%    Note that the function GCF is different.It creates a figure and returns its handle if it does
not  %    exist.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

Hf=get(0, 'Children' );    %    check for figure children
if isempty(Hf)
      return
else
      Hf=get(0, 'CurrentFigure' );
end
```

函数 **mmgcf** 首先检查根对象的子对象的图形是否存在，如至少有一个图形对象时，根对象的 '**CurrentFigure**' 属性就返回当前的图形。

函数 **mmgca** 为坐标轴对象执行同样的功能，如同在它的 M 文件内所描述的那样。

```
function Ha=mmgca
%MMGCA Get Current Axes if it exists.
%    MMGCA returns the handle of the current axes if it exists.
%    If no current axes exists,MMGCA returns an empty handle.
%
```

```
%    Note that the function GCA is diffent.It create a figure and an axes and returns the axes
%    handle if they do not exist.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

Ha=findobj(0, 'Type' , 'axes' );
if isempty(Ha)
      return
else
      Ha=get(get(0, 'CurrentFigfure' ), 'CurrentAxes' );
end
```

由于函数 **gco** 已经表现出当对象不存在时返回空矩阵的行为特性，就不需要函数 **mmgco** 了。

在精通 MATLAB 工具箱中的另一个函数是 **mmzap**，在二维图形那一章里已作过介绍。如下 M 文件中所示，它使用 **mmgcf** 作错误检查，与 **findobj** 和 **get** 一起删除一个指定的图形。

```
function mmzap(arg)
%MMZAP Delete graphics object using mouse.
%    MMZAP waits for a mouse click on an object in a figure window and deletes the object.
%    MMZAP or MMZAP text erases text objects.
%    MMZAP axes erases axes objects.
%    MMZAP line erases line objects.
%    MMZAP surf erases surface objects.
%    MMZAP patch erases patch objects.
%
%    Clicking on an object other than the selected type or striking a key on the keyboard
aborts    %    the command.

%    Copyright    (c)    1996 by Prentice-Hall,Inc.

if nargin<1,arg= 'text' ;end

Hf=mmgcf;
if isempty(Hf),error( 'No Figure Available.' ),end
if length(findobj(0, 'Type' , 'figure' ))==1
      figure(Hf)    %    bring only figure forword
end
key=waitforbuttonpress;    %    pause untill user takes some action
if key    %    key on keyboard pressed
      return    %    take no action
else    %    object selected
      object=gco    %    get object selected by buttonpress
      type=get(object, 'Type' );
```

```
        if all(type(1:4)==arg(1:4))    %    delete only if    'Type'    is correct
            delete(object)
        end
    end
```

在编写句柄图形函数的 M 文件时，函数 **mmzap** 描述了一种很有用的技术。它利用函数 **waitforbuttonpress** 和 **gco** 的结合用鼠标来获取所选定对象的句柄。 **waifortbuttonpress** 是一个 MATLAB 内置函数，它的功能是等待鼠标点击或按键。它的帮助文本如下：

```
» help waitforbuttonpress

WAITFOR BUTTONPRESS Wait for key/buttonpress over figure.
    T= WAITFOR BUTTONPRESS stops program execution untill a key or mouse button is
    pressed over a figure window.Returns 0 when terminated by a mouse buttonpress,or 1
    when terminated by a keypress.Additional information about the terminating event is
        available from the current figure.

    See also GINPUT,GCF.
```

帮助信息：

WAITFORBUTTONPRESS 等待一个鼠标/按钮对图形按下。

T=WAITFORBUTTONPRESS 停止程序的执行，直到鼠标按钮或键在一个图形窗口按下。当鼠标按钮按下时返回 0；当键按下时返回 1。其它的结束事件的信息可从当前的图形中窗口　　获取。

参阅 GINPUT 和 GCF。

鼠标按钮在鼠标指针指的图形上按下后，函数 **gco** 返回所点中对象的句柄。然后，该句柄可用来操作选中的对象。在精通 MATLAB 工具箱中，用这种简单的选择技术的函数还有 **mmline** 和 **mmaxes**。其中， **mmline** 的 M 文件描述如下：

```
function mmline(arg1,arg2, arg3, arg4, arg5, arg6)
%MMLINE Set Line Properties Using Mouse
%   MMLINE waites for a mouse click on a line then applies the desired properties to the
%   selected line.
%   Properties are given in parts,e.g.,MMLINE Name value...
%   Properties:
%   NAME            VALUE{default}
%   color           [Y m c r g b w k] or an RGB in quotes: '[r g b]'
%   style                 [-   --   ;   -.]
%   mark            [o   +   .   *   X]
%   width           points for linewidth {0.5}
%   size            points for marker size {6}
```

```
%   zap                 (n.a.)      delete selected line
%   Examples:
%   MMLINE color r width 2      sets color to red and width to 2 points
%   MMLINE mark + size 8        sets marker type to + and size to 8 points
%
%   Clicking on an object other than a line,or striking a key on the keyboard aborts the
%   command.

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

Hf=mmgcf;
if isempty(Hf),error( 'No Figure Avaiable.' ),end
if length(get(0, 'Children' ))==1
    figure(Hf)   %   bring only figure forword
end
key=waitforbuttonpress;
if key   %   key on keyboard pressed
    return
else   %   object selected
    Hl=gco
    if strcmp(get(Hl, 'Type' ), 'line' )   %   line object selected
        for i=1:2:max(nargin-1,1)
            Name=eval(sprintf( 'arg%.0f ' ,i),[]);   get Name argument
            if strcmp(Name, 'zap' )
                delete(Hl),return
            end
            value=eval(sprintf( 'arg%.0f ' ,i+1),[]);   %   get value
            if strcmp(Name, 'color' )
                set(Hl, 'Color' ,value)
            elseif strcmp(Name, 'style' )
                set(Hl, 'Linestyle' ,value)
            elseif strcmp(Name, 'mark' )
                set(Hl, 'Linestyle' ,value)
            elseif strcmp(Name, 'width' )
                value=abs(eval(value))
                set(Hl, 'LineWidth' ,value)
            elseif strcmp(Name, 'size' )
                value=abs(eval(value))
                set(Hl, 'MarkerSize' ,value)
            else
                disp([ 'Unknown Property Name: ' Name'])
            end
        end
    end
```

```
end
```

精通 MATLAB 工具箱中的函数 **mmpaper** 以简单的方式阐述了对纸张属性的使用。如下所示，函数 **mmpaper** 设置当前图形的纸张属性，并将所有以后的图形设成缺省值。函数 **mmpage** 在下一章讨论，它是一个 **mmpaper** 友函数。**mmpage** 建立一个图形用户界面，设定图形在打印页上的位置。

```
function mmpaper(arg1,arg2,arg3,arg4,arg5,arg6)
%MMPAPER Set Default Paper Properties.
%   MMPAPER Name value...
%    sets default paper properties for the current figure and succeeding figures based on Name    %   value pairs.
%   Properties:
%   NAME                VALUE     {default}
%   Units               [{inches},centimeters,points,normal]
%   orient              [{portrait},landscape]
%   type                    [{usletter},uslegal,a3,a4letter,a5,b4,tabloid]
%
%   Examples:
%   MMPAPER Units inch orient landscape
%   MMPAPER type tabloid
%
%   MMPAPER with no arguments returns the current paper defaults.

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

Hf=mmgcf;
flag=0;
if isempty(Hf)
    flag=1;
    Hf=figure( 'Visible' , 'off ' );
end
if nargin
    for i=1:2:max(nargin-1,1)
        Name=eval(sprintf( 'arg%.0f ' ,i)0,[]);   %   get Name argument
        value=eval(sprintf( 'arg%.0f ' ,i+1)0,[]);   %   get Name argument
        if Name(1)== 'o'
            set(0, 'DefaultFigurePaperOrientation' ,value)
            set(Hf, 'PaperOrientation' ,value)
        elseif Name(1)== 't'
            set(0, 'DefaultFigurePaperType' ,value)
            set(Hf, 'PaperType' ,value)
        elseif Name(1)== 'u'
            set(0, 'DefaultFigurePaperUnits' ,value)
```

```
            set(Hf, 'PaperUnits' ,value)
        else
            disp([ 'Unknown Property Name:' Name])
        ens
    end
end
```

当把对象放在一个特定的位置时，有时在象素和归一化坐标之间进行转化是很有用的。在精通 MATLAB 工具箱中有两个函数进行这种转换。第一个是 **mmpx2n**，它将象素转化为归一化坐标；第二个是 **mmn2px**，它进行相反的转换。这些函数演示了如何以所需的一组单位获取 '**Position**' 属性值。首先，把对象的当前 '**Units**' 属性保存起来；然后，将 '**Units**' 属性设成所需的值并获取所需的 '**Position**' 属性值；最后，将 '**Units**' 的值恢复为初始值。**mmpx2n** 的 M 文件描述如下：

```
function Y=mmpx2n(X,Hf)
%MMPX2N Pixel to Normalized Coordinate Transformation.
%   MMPX2N(X) converts the Position vector   X   from pixel coordinates to normalized
%   coordinates w.r.t.the computer screen.
%
%    MMPX2N converts the Position vector X from pixel coordinate to narmalized coordinates
%   w.r.t.the figure windoiw having handle H.
%
%   X=[left   bottom   width   height] or X=[width   height]

%   Copyright   (c)   1996 by Prentice-Hall,Inc.

msg= 'Input is not a pixel Position vector.' ;
lx=length(X);

sz= 'Position' ;
if nargin==1,Hf=0;sz= 'ScreenSize' ;end

if any(X<1)| (lx~=4&lx~=2)
    error(msg)
end
if lx==2,X=[1   1   X(:)' ];end   %   [width   height] input format
u=get(Hf, 'Units' );   %   get Units
set (Hf, 'Units' , 'pixels' );   %   set Units to pixels
s=get(Hf,sz);
Y=(X-1)./([s(3:4)]-1);   %   convert
set(Hf, 'Units' ,u);   %   reset Units
if any(Y>1)
    error(msg)
```

```
end
if lx==2,Y=Y(3:4);end    %    [width height] output format
```

精通 MATLAB 工具箱中的两个函数 **mmcont2** 和 **mmcont3** 都用用户指定的颜色映象画等值线图。每一个函数分析输入参量并建立一个字符串，它包含了颜色的说明。一旦设置了字符串，就设置了当前坐标轴的 '**ColorOrder**' 属性；最后，它们分别调用具有合适的参量的函数 **contour** 和 **contour3** 来画出图形。函数 **mmcont2** 的 M 文件描述如下：

```
function[cs,h]=mmcont2(arg1,arg2,arg3,arg4,arg5)
%MMCONT2 2-D contour plot using a colormap.
%    MMCONT2(X,Y,Z,N,C) plots N contours of Z in 2-D using the color
%    specified in C.C can be a linestyle and color as used in plot,
%    e.g., 'r-' ,orC can be the string Name of a colormap. X and Y
%    define the axis limits.
%    If not given default argument values are: N=10,C= 'hot' ,
%    X and Y =row and column indices of Z. Examples:
%    MMCONT2(Z)                  10 lines with hot colormap
%    MMCONT2(Z,20)                    20 lines with hot colormap
%    MMCONT2(Z, 'copper' )       10 lines with copper colormap
%    MMCONT2(Z,20, 'gray' )           20 lines with gray colormap
%    MMCONT2(X,Y,Z, 'jet' )      10 lines with jet colormap
%    MMCONT2(Z, 'c-' )           10 dashed lines in cyan
%    MMCONT2(X,Y,Z,25, 'pink' )  25 lines in pink colormap
%
%    CS=MMCONT2(…) returns the contour matrix CS as described in
%    CONTOURC.
%    [CS,H]=MMCONT2(…) returns a column vector H of handles to
%    line objects.

%    Copyright (c) 1996 by Prentice-Hall,Inc.
n=10;c= 'hot' ;    %    default values
nargs=nargin;cflag=1;
if nargin<1,error( 'Not enough input arguments.' ), end

for i=2:nargin    %    check input arguments for N and C
      argi=eval(sprintf( 'arg%.0f ' ,i));
      if ~isstr(argi)&length(argi)==1    %    must be N, grab it
            n=argi;
            nargs=i;    %    # args to pass to contour2
      elseif isstr(argi)    %    must be C
            if exist(argi)==2    %    is colormap,so grab it
                  c=argi;
                  nargs=i-1;
            else    %    is single color/linestyle
```

```
                cflag=0;
                nargs=i;
            end
        end
    end
    if cflag    %    a colormap has been chosen
        clf    %    clear figure
        view(2)    %    make it 2-D
        hold on    %    hold it
        mapstr=sprintf([c '(%.0f)' ],n);
        set(gca,'ColorOrder',eval(mapstr));
    end
    evalstr= '[CS,H]=contour(' ;
    for i=1:nargs
        evalstr=[evalstr sprintf( 'arg%.0f ' ,i)    ',' ];
    end
    lstr=length (evalstr);
    evalstr(lstr:lstr+1)= ');' ;
    eval(evalstr)
    hold off
    if nargout==1,          cs=CS;
        elseif nargout==2,      cs=CS;h=H;
    end
```

这里要讨论的最后一个精通 MATLAB 工具箱函数是 **mmtile**。就象在二维函数那一章里所描述的一样，该函数在计算机屏幕上将 4 个已存在的图形按平铺模式排列起来。函数 **mmtile.m** 的内容如下所示：

```
function h=mmtile(n)
%    MMTILE Tile Figure Windows.
%    MMTILE with no arguments, tiles the current figure windows
%    and brings them to the foreground.
%    Figure size is adjusted so that 4 figure windows fit on the screen.
%    Figures are arranged in a clockwise fashion starting in the
%    upper-left corner of the display
%
%    MMTILE(N) makes tile N the current figure if it exists.
%    Otherwise, the next tile is created for subsequent plotting
%
%    Tiled figure windows are titled TILE #1,TILE #2, TILE #3, TILE #4.

%    Copyright (c) 1996 by Prentice-Hall,Inc.

HT=40;          %tile height fudge in pixels
```

```
WD=20;      %  tile width fudge
%  adjust the above as necessary to eliminate tile overlaps
%  bigger fudge numbers increase gaps between tiles

Hf=sort(get(0,'Children'));  %  get handles of current figures
nHf=length(Hf);
set(0,'Units','Pixels')  %  set screen dimensions to pixels
sz=get(0,'Screensize');  %  get screen size in pixels
tsz=0.9*sz(3:4);  %  defaut tile area is almost whole monitor
if sz(4)>sz(3),  %  if portrait monitor
    tsz(2)=.75*tsz(1);  %  take a landscape chunk
end
tsz=min(tsz,[920 690]);  %  hold tile area on large screens to 920 by 690
t1(1,1)=sz(3)-tsz(1)+1;  %  left side of left tiles
t1(2,1)=t1(1,1)+tsz(1)/2;  %  left side of right tiles
tb(1,1)=sz(4)-tsz(2)+1;  %  bottom of bottom tiles
tb(2,1)=tb(1,1)+tsz(2)/2;  %  bottom of top tiles

tpos=zeros(4);  %  matrix holding tile Position vectors
tpos(:,1)=t1([1 2 2 1],1);  %  left sides
tpos(:,2)=t1([2 2 1 1],1);  %  bottoms
tpos(:,3)=(tsz(1)/2-WD)*ones(4,1);  %  widths
tpos(:,4)=(tsz(2)/2-HT)*ones(4,1);  %  heights
tpos=fix(tpos);  %  make sure pixel Positions are integers
if nargin==0  %  tile figures as needed
    for i=1:min(nHf,4)
        set(Hf(i),'Units','pixels')
        if any(get(Hf(i),'Position')~=tpos(i,:))
            set(Hf(i),'Position',tpos(i,:),…
              'NumberTitle','off',…
              'Name',['TILE #' int2str(i)])
        end
        figure(Hf(i))
    end
else  %  go to tile N or create it
    n=rem(abs(n)-1,4)+1;  %  N must be between 1 and 4
    if n<=nHf  %  tile N exists,make it current
        figure(Hf(n))
    else  %  tile N does not exist,create next one
        n=nHf+1;
        figure('Position',tpos(n,:),…
          'NumberTitle','off',…
          'Name',['TILE #' int2str(n)])
    end
```

end

如上面所描述的，函数 **mmtile** 从根对象得到所有的图形对象的句柄和屏幕尺寸，为该图形计算新的位置和尺寸，然后设置每个图形的 '**Units**'，'**Position**'，'**Number**' 和 '**Name**' 属性。它具有安置和缩放图形的效能，并在每个窗口标题中，改变名字字符串。**HT** 和 **WT** 给出的号码与计算机平台有关。它们对图形的 '**Position**' 描述窗口内的可画区域而不是外部尺寸有补偿作用。

20.12 属性名和属性值

下面各表中列出了 MATLAB 4.2 版本中的属性名和属性值。有一个星号*的属性是非文件化的。用大括号{}括起来的属性值是缺省值。

表 20.3

根对象属性	
BlackAndWhite	自动硬件检测标志
on:	认为显示是单色的，不检测；
{off}:	检测显示类型
***VlaxkOutUnusedSlots**	值为**[{no}|yes]**
***CaptureMap**	
CaptureMatrix	由 **CaptureRect** 矩形所包围的区域内图象数据的只读矩阵，使用 **image** 来显示
CaptureRect	捕捉矩形的尺寸和位置，是一个 4 元素的向量 **[left,bottom,width,height]**，单位由 **Units** 属性指定。
***CaseSen**	值为**[{on}|off]**
CurrentFigure	当前图形的句柄。
Diary	会话记录
on:	将所有的键盘输入和大部分输出拷贝到文件中
{off}:	不将输入和输出存入文件
DiaryFile	一个包含 **diary** 文件名的字符串，缺省的文件名为 **diary**
Echo	脚本响应模式
on:	在文件执行时，显示脚本文件的每一行
{off}:	除非指定 **echo on**，否则不响应
Format	数字显示的格式
{short}:	5 位的定点格式
shortE:	5 位的浮点格式
long:	15 位换算过的定点格式
longE:	15 位的浮点格式
hex:	16 进制格式
bank:	美元和分的定点格式
+:	显示+和-符号
rat:	用整数比率逼近

FormatSpacing 输出间隔

{loose}: 显示附加行的输入

compact: 取消附加行的输入

***HideUndocumented** 控制非文件式属性的显示

no: 显示非文件式属性

{yes}: 不显示非文件式属性

PointerLocation 相对于屏幕左下角指针位置的只读向量**[left,bottom]**或**[X,Y]**，单位由 **Units** 属性指定

PointerWindow 含有鼠标指针的图形句柄，如果不在图形窗口内，值为 0。

ScreenDepth 整数，指定以比特为单位的屏幕颜色深度，比如：1 代表单色，8 代表 256 色或灰度

ScreenSize 位置向量**[left,bottom,width,height]**，其中**[left,bottom]**常为**[0 0]**，**[width,height]**是屏幕尺寸，单位由 **Units** 属性指定

***StatusTable** 向量

***TerminalHideGraphCommand** 文本串

TerminalOneWindow 由终端图形驱动器使用

no: 终端有多窗口

yes: 终端只有一个窗口

***TerminalDimensions** 终端尺寸向量**[width,height]**

TerminalProtocal 启动时终端类型设置，然后为只读

none: 非终端模式，不连到 X 服务器

X: 找到 X 显示服务器，X Windows 模式

tek401x: Tektronix 4010/4014 仿真模式

tek410x: Tektronix 4100/4105 仿真模式

***TerminalShowGraphCommand** 文本串

Units Position 属性值的度量单位

inches: 英寸

centimeters: 厘米

normalized: 归一化坐标，屏幕的左下角映射到[0 0]，右上角映射到[1 1]

points: 排字机的点，等于 1/72 英寸

{pixels}: 屏幕象素，计算机屏幕分辨率的最小单位

***UsageTable** 向量

ButtonDowFcn MATLAB 回调字符串，当对象被选择时传给函数 **eval**，初始值是一空矩阵

Children 所有图形对象句柄的只读向量

Clipping 数据限幅模式

{on}: 对根对象无效果

off: 对根对象无效果

Interruptible **ButtonDowFcn** 回调字符串的可中断性

{no}: 不能被其它回调中断

yes: 可以被其它回调中断

Parent	父对象的句柄，常为空矩阵
***Selected**	值为**[on|off]**
***Tag**	文本串
Type	只读的对象辨识字符串，常是 **root**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	对象可视性
{on}:	对根对象无效果
off:	对根对象无效果

表 20.4

图形对象属性

BackingStore	为了快速重画，存储图形窗口的拷贝
{on}:	当一个图原来被覆盖的一部分显露时，拷贝备份，刷新窗口较快，但需要较多的内存
off:	重画图形以前被覆盖的部分，刷新较慢，但节省内存
***CapterMap**	矩阵
***Client**	矩阵
Color	图形背景色，一个 3 元素的 **RGB** 向量或 MATLAB 预定的颜色名，缺省的颜色是**黑色**
Colormap	m×3 的 **RGB** 向量矩阵，参阅函数 **colormap**
***Colortable**	矩阵，也许包含一份系统颜色映象的拷贝
CurrentAxes	图形的当前坐标轴的句柄
CurrentCharacter	当鼠标指针在图形窗口中，键盘上最新按下的字符键
CurrentMenu	最近被选择的菜单项的句柄
CurrentObject	图形内，最近被选择的对象的句柄，即由函数 **gco** 返回的句柄
CurrentPoint	一个位置向量**[left,bottom]**或图形窗口的点的**[X,Y]**，该处是鼠标指针最近一次按下或释放时所在的位置。
FixedColors	n×3 的 **RGB** 向量矩阵，它使用系统查色表中的槽来定义颜色，初始确定的颜色是 **black** 和 **white**
***Flint**	
InvertHardcopy	改变图形元素的颜色以打印
{on}:	将图形的背景色改为白色，而线条、文本和坐标轴改为黑色以打印
off:	打印的输出颜色和显示的颜色完全一致
KeyPressFcn	当鼠标指针处在图形内，按下键，传递给函数 **eval** 的 MATLAB 回调字符串
MenuBar	将 MATLAB 菜单在图形窗口的顶部显示，或在某些系统中在屏幕的顶部显示
{figure}:	显示缺省的 MATLAB 菜单
none:	不显示缺省的 MATLAB 菜单
MinColormap	颜色表输入项使用的最小数目。它影响系统颜色表。如设置太低，会使未选中的图形以伪彩色显示。
Name	图形框架窗口的标题（**不是**坐标轴的标题）。缺省时是

空串，如设为 string（字符串），窗口标题变为：**Figure No.n：string**

NextPlot 决定新图作图行为

new： 画前建立一个新的图形窗口

{add}： 在当前的图形中加上新的对象

replace： 在画图前，将除**位置**属性外的所有图形对象属性重新设置为缺省值，并删除所有子对象

NumberTitle 在图形标题中加上图形编号

{on}： 如果 **Name** 属性值被设为 string，窗口标题是 **Figure No.N：string**

off： 窗口标题仅仅是 **Name** 属性字符串

PaperUnits **纸张**属性的度量单位

{inches}： 英寸

centimeters： 厘米

normalized： 归一化坐标

points： 点，每一点为 1/72 英寸

PaperOrientation 打印时的纸张方向

{portrait}： 肖像方向，最长页面尺寸是垂直方向

landscape： 景象方向，最长页面尺寸是水平方向

PaperPosition 代表打印页面上图形位置的向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于打印页面图形左下角的位置，**[width,height]**是打印图形的尺寸，单位由 **PaperUnits** 属性指定

PaperSize 向量**[width,height]**代表了用于打印的纸张尺寸，单位由 **PaperUnits** 属性指定，缺省的纸张大小为**[8.5 11]**

PaperType 打印图形纸张的类型。当 **PaperUnits** 设定为归一化坐标时，MATLAB 使用 **PaperType** 来按比例调整图形的大小

{usletter}： 标准的美国信纸

uslegal1： 标准的美国法定纸张

a3： 欧洲 A3 纸

a4letter： 欧洲 A4 信纸

a5： 欧洲 A5 纸

b4： 欧洲 B4 纸

tabloid： 标准的美国报纸

Pointer 鼠标指针形状

crosshair： 十字形指针

{arrow}： 箭头

watch： 钟表指针

top1： 指向左上方的箭头

topr： 指向右上方的箭头

bot1： 指向左下方的箭头

botr： 指向右下方的箭头

circle： 圆

cross：双线十字形

fleur：4 头箭形或指南针形

Position　位置向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于计算机屏幕的左下角窗口左下角的位置，**[width,height]**是屏幕尺寸，单位由 **Units** 属性指定

Resize　允许不允许交互图形重新定尺寸

{on}：窗口可以用鼠标来重新定尺寸

off：窗口不能用鼠标来重新定尺寸

ResizeFcn　MATLAB 回调字符串，当窗口用鼠标重新定尺寸时传给函数 **eval**

***Scrolled**　值为**[{on}|off]**

SelectionType　一个只读字符串，提供了有关最近一次鼠标按钮选择所使用方式的信息。但实际是哪个键和/或按钮按下与平台有关

{normal}：点击（按下和释放）鼠标左键，或只是鼠标按钮

extended：按下 **shift** 键并进行多个常规（normal）选择；同时击双按钮鼠标的两个按钮；或点击一个三按钮鼠标的中按钮

alt：按下 **Control** 键并进行一次常规选择；或者点击一个双按钮或三按钮鼠标的右按钮

open：双击任何鼠标按钮

Share Colors　共享颜色表的槽

no：不和其它窗口共享颜色表的槽

{yes}：只要可能，重用颜色表中的槽

***StatusTable**　向量

Units　各种位置属性值的度量单位

inches：英寸

centimeters：厘米

normalized：归一化坐标，屏幕的左下角映射到 [0　0]，右上角映射到 [1　1]

points：排字机的点，等于 1/72 英寸

{pixels}：屏幕象素，计算机屏幕分辨率的最小单位

***UsageTable**　向量

WindowButtonDownFcn　当鼠标指针在图形内时，只要按一个鼠标按钮，MATLAB 回调字符串传递给函数 **eval**

WindowButtonMotionFcn　当鼠标指针在图形内时，只要移动一个鼠标按钮，MATLAB 回调字符串传递给函数 **eval**

***WindowID**　长整数

ButtonDownFcn　当图形被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵

Children　图形中所有子对象句柄的只读向量；坐标轴对象，uicontrol 对象和 uimenu 对象

Clipping　数据限幅模式

{on}:	对图形对象不起作用
off:	对图形对象不起作用
Interruptible	指定图形回调字符串是否可中断
{no}:	不能被其它回调中断
yes:	可以被其它回调中断
Parent	图形父对象的句柄，常是 0
***Selected**	值为**[on|off]**
***Tag**	文本串
Type	只读的对象辨识字符串，常是 **figure**
UserDate	用户指定的数据，可以是矩阵、字符串等等
Visible	图形窗口的可视性
{on}:	窗口在屏幕上可视
off:	窗口不可视

表 20.5

坐标轴对象属性

AspectRatio	纵横比向量**[axis_ratio,data_ratio]**，这里 **axis_ratio** 是坐标轴对象的纵横比（宽度/高度），**data_ratio** 是沿着水平轴和垂直轴的数据单位的长度比。如设置，则 MATLAB 建立一个最大的坐标轴，保留这些比率，该最大轴将在 **Position** 定义的矩形内拟合。该属性的缺省值为**[NaN,NaN]**
Box	坐标轴的边框
on:	将坐标轴包在一个框架或立方体内
{off}:	不包坐标轴
CLim	颜色界限向量**[cmin cmax]**，它确定将数据映射到颜色映象。**cmin** 是映射到颜色映象第一个入口项的数据，**cmax** 是映射到最后一项的数据。参阅函数 **cmais**
CLimMode	颜色限制模式
{auto}:	颜色界限映成轴子对象的数据整个范围
manual:	颜色界限并不自动改变。设置 **CLim** 就把 **CLimMode** 值设为人工
Color	坐标轴背景颜色。一个三元素的 **RGB** 向量或一个预定义的颜色名。缺省值是 **none**，它使用图形的背景色
ColorOrder	一个 m×3 **RGB** 值矩阵。如果线条颜色没有用函数 **plot** 和 **plot3** 指定，就用这些颜色。缺省的 **ColorOrder** 为黄，紫红，洋红，红，绿和蓝
CurrentPoint	包含在坐标轴空间内的一对点的坐标矩阵，它定义了从坐标空间前面延伸到后面的一条三维直线。其形式是**[xb yb zb ; xf yf zf]**。单位在 **Units** 属性中指定。点**[xf yf zf]**是鼠标在坐标轴对象中上一次点击的坐标
DrawMode	对象生成次序

{normal}：将对象排序，然后按照当前**视图**从后向前绘制

fast：按已建立的次序绘制对象，不首先排序

***ExpFontAngle** 值为**[{normal}|italic|oblique]**

***ExpFontName** 缺省值为 **Helvetica**

***ExpFontSize** 缺省值为 8 点

***ExpFontStrikeThrough** 值为**[on|{off}]**

***ExpFontUnderline** 值为**[on|{off}]**

***ExpFontWeight** 值为**[light|{normal}|demi|bold]**

FontAngle 坐标轴文本为斜体

{normal}：正常的字体角度

italic：斜体

oblique：某些系统中为斜体

FontName 坐标轴单位标志的字体名。坐标轴上的标志并不改变字体，除非通过设置 **XLabel**，**YLabel** 和 **ZLable** 属性来重新显示它们。缺省的字体为 **Helvetica**

FontSize 坐标轴标志和标题的大小，以点为单位，缺省值为 12 点

***FontStrikeThrough** 值为**[on|{off}]**

***FontUnderline** 值为**[on|{off}]**

FontWeight 坐标轴文本加黑

light：淡字体

{normal}：正常字体

demi：适中或者黑体

bold：黑体

GridLineStyle 格栅线形

-：实线

--：虚线

{:}：点线

-.：点划线

***Layer** 值为**[top|{bottom}]**

LineStyleOrder 指定线形次序的字符串，用在坐标轴上画多条线。例如：’ .- | : | -- | - ‘ 将通过点划线、点线、虚线和实线循环。LineStyleOrder 缺省值为‘-’，即只有实线

LineWidth **X** ，**Y** 和 **Z** 坐标轴的宽度。缺省值为 **0.5** 点

***MinorGridLineStyle** 值为**[- | -- | {:} | -.]**

NextPlot 画新图时要采取的动作

new：在画前建立新的坐标轴

add：把新的对象加到当前坐标轴，参阅 **hold**

{replace}：在画前，删除当前坐标轴和它的子对象，并用新的坐标轴对象来代替它

Position 位置向量**[left,bottom,width,height]**，这里**[left,bottom]** 代表了相对于图形对象左下角的坐标轴左下角位置，**[width,height]**是坐标轴的尺寸，单位由 **Units** 属性指定

TickLength 向量**[2Dlength 3Dlength]**，代表了在二维和三维**视图**中坐标轴刻度标记的长度。该长度是相对于坐标轴的长度。缺省值为[0.01 0.01]，代表二维**视图**坐标轴长度的 1/100，三维**视图**坐标轴长度的 5/1000

TickDir 值为**[{in}|out]**

in： 刻度标记从坐标轴线向内，二维**视图**为缺省值

out： 刻度标记从坐标轴线向外，三维**视图**为缺省值

Title 坐标轴标题文本对象的句柄

Units 位置属性值的度量单位

inches： 英寸

centimeters： 厘米

{normalized}： 归一化坐标，对象左下角映射到**[0 0]**，右上角映射到**[1 1]**

points： 排字机的点，等于 1/72 英寸

pixels： 屏幕象素，计算机屏幕分辨率的最小单位

View 向量**[az el]**，它代表了观察者的视角，以度为单位。**az** 为方位角或视角相对于负 **Y** 轴向右的转角；**el** 为 **X-Y** 平面向上的仰角。详细细节见三维图形这一章

XColor **RGB** 向量或预定的颜色字符串，它指定 **X** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）

XDir **X** 值增加的方向

{normal}： **X** 值从左向右增加

reverse： **X** 值从右向左增加

XForm 一个 4×4 的**视图**转换矩阵。设置 **view** 属性影响 **XForm**

XGrid **X** 轴上的格栅线

on： **X** 轴上每个刻度标记处画格栅线

{off}： 不画格栅线

XLabel **X** 轴标志文本对象的句柄

XLim 向量**[xmin xmax]**，指定 **X** 轴最小和最大值

XLimMode **X** 轴的界限模式

{auto}： 自动计算 **XLim**，包括所有轴子对象的 **XData**

manual： 从 **XLim** 取 **X** 轴界限

***XMinorGrid** 值为**[on|{off}]**

***XMinorTicks** 值为**[on|{off}]**

Xscale **X** 轴换算

{linear}： 线形换算

log： 对数换算

XTick 数据值向量，按此数据值将刻度标记画在 **X** 轴上，将 **XTick** 设为空矩阵就撤消刻度标记

XTickLabels 文本字符串矩阵，用在 **X** 轴上标出刻度标记。如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值

XTickLabelMode **X** 轴刻度标记的标志模式

{auto}： **X** 轴刻度标记张成 **XData**

manual：	从 **XTickLabels** 中取 **X** 轴刻度标记
XTickMode	**X** 轴刻度标记的间隔模式
{auto}：	**X** 轴刻度标记间隔以张成 **XData**
manual：	从 **XTick** 生成 **X** 轴刻度标记
YColor	**RGB** 向量或预定的颜色字符串，它指定 **Y** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）
YDir	**Y** 值增加的方向
{normal}：	**Y** 值从左向右增加
reverse：	**Y** 值从右向左增加
YGrid	**Y** 轴上的格栅线
on：	**Y** 轴上每个刻度标记处画格栅线
{off}：	不画格栅线
YLabel	**Y** 轴标志文本对象的句柄
YLim	向量**[Ymin Ymax]**，指定 **Y** 轴最小和最大值
YLimMode	**Y** 轴的界限模式
{auto}：	自动计算 **YLim**，包括所有轴子对象的 **YData**
manual：	从 **YLim** 取 **Y** 轴界限
***YMinorGrid**	值为**[on|{off}]**
***YMinorTicks**	值为**[on|{off}]**
Yscale	**Y** 轴换算
{linear}：	线形换算
log：	对数换算
YTick	数据值向量，按此数据值将刻度标记画在 **Y** 轴上。将 **YTick** 设为空矩阵就消去刻度标记
YTickLabels	文本字符串矩阵，用在 **Y** 轴上标出刻度标记，如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
YTickLabelMode	**Y** 轴刻度标记的标志模式
{auto}：	**Y** 轴刻度标记张成 **YData**
manual：	从 **YTick**Lab**el**s 中取 **Y** 轴刻度标记
YTickMode	**Y** 轴刻度标记的间隔模式
{auto}：	**Y** 轴刻度标记间隔以张成 **YData**
manual：	从 **YTick** 生成 **Y** 轴刻度标记
ZColor	**RGB** 向量或预定的颜色字符串，它指定 **Z** 轴线、标志、刻度标记和格栅线的颜色。缺省为 **white**（白色）
ZDir	**Z** 值增加的方向
{normal}：	**Z** 值从左向右增加
reverse：	**Z** 值从右向左增加
ZGrid	**Z** 轴上的格栅线
on：	**Z** 轴上每个刻度标记处画格栅线
{off}：	不画格栅线
ZLabel	**Z** 轴标志文本对象的句柄
ZLim	向量**[Zmin Zmax]**，指定 **Z** 轴最小和最大值
ZLimMode	**Z** 轴的界限模式
{auto}：	自动计算 **ZLim**，包括所有轴子对象的 **ZData**

manual：	从 **ZLim** 取 **Z** 轴界限
***ZMinorGrid**	值为**[on|{off}]**
***ZMinorTicks**	值为**[on|{off}]**
Zscale	**Z** 轴换算
{linear}：	线形换算
log：	对数换算
ZTick	数据值向量，按此数据值将刻度标记画在 **Z** 轴上，将 **ZTick** 设为空矩阵就撤消刻度标记
ZTickLabels	文本字符串矩阵，用在 **Z** 轴上标出刻度标记，如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
ZTickLabelMode	**Z** 轴刻度标记的标志模式
{auto}：	**Z** 轴刻度标记张成 **ZData**
manual：	从 **ZTickLabels** 中取 **Z** 轴刻度标记
ZTickMode	**Z** 轴刻度标记的间隔模式
{auto}：	**Z** 轴刻度标记间隔以张成 **ZData**
manual：	从 **ZTick** 生成 **Z** 轴刻度标记
ButtonDownFcn	MATLAB 回调字符串，当坐标轴被选中时，将它传递给函数 **eval**；初始值是一个空矩阵
Children	除了轴标志和标题对象以外，所有子对象句柄的只读向量；包括线、曲面、图象、补片和文本对象
Clipping	数据限幅模式
{on}：	对坐标轴对象不起作用
off：	对坐标轴对象不起作用
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	该回调字符串不能被其它回调所中断
yes：	该回调字符串可以被其它回调所中断
Parent	包含坐标轴对象的图形句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **axes**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	轴线、刻度标记和标志的可视性
{on}：	坐标轴在屏幕上可视
off：	坐标轴不可视

表 20.6

线条对象属性

Color	线条颜色。一个三个元素 **RGB** 向量或 MATLAB 预定的颜色名之一。缺省值是 **white**（白色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
backgrount：	通过在图形背景色中重画线来消除线条。这会破坏被

	消除的线后的对象
	xor：用线下屏幕的颜色执行异或 **OR（XOR）**运算，画
none：	出和消除线条。当画在其它对象上时，可造成不正确的颜色
	当移动或删除线条时该线不会被消除
LineStyle	线形控制
{-}：	画通过所有数据点的实线
--：	画通过所有数据点的虚线
:：	画通过所有数据点的点线
-.：	画通过所有数据点的点划线
+：	用加号作记号，标出所有的数据点
o：	用圆圈作记号，标出所有的数据点
***：**	用星号作记号，标出所有的数据点
.：	用实点作记号，标出所有的数据点
X：	用 **X** 符号作记号，标出所有的数据点
LineWidth	以点为单位的线宽。缺省值是 **0.5**
MarkerSize	以点为单位的记号大小，缺省值是 **6** 点
Xdate	线的 **X** 轴坐标的向量
Ydate	线的 **Y** 轴坐标的向量
Zdate	线的 **Z** 轴坐标的向量
ButtonDownFcn	当线条对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，线条对象没有子对象
Clipping	数据限幅模式
{on}：	在坐标轴界限外的线的任何部分不显示
off：	线条数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	不能被其它回调中断
yes：	可以被其它回调中断
Parent	包含线条对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **line**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	线的可视性
{on}：	线在屏幕上可视
off：	线在屏幕上不可视

表 20.7

文本对象属性

Color	线条颜色。一个三个元素 **RGB** 向量或 MATLAB 预定的颜色名之一。缺省值是 **white**（白色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地

画出。这是最精确的，也是最慢的一种模式

backgrount： 通过在图形背景色中重画文本来消除文本。这会破坏被消除的文本后的对象

xor： 用文本下屏幕颜色执行异或 **OR（XOR）**运算，画出和消除该文本。当画在其它对象上时，会造成不正确的颜色

none： 当移动或删除文本时该文本不会被消除

Extent 文本位置向量**[left,bottom,width,height]**，**[left,bottom]**代表了相对于坐标轴对象左下角的文本对象左下角的位置，**[width,height]**是包围文本串的矩形区域的大小，单位由 **Units** 属性指定

FontAngle 文本为斜体

{normal}： 正常的字体角度

italics： 斜体

oblique： 某些系统中为斜体

FontName 文本对象的字体名。缺省的字体名为 **Helvetica**

FontSize 文本对象的大小，以点为单位，缺省值为 **12** 点

***FontStrikeThrough** 值为**[on|{off}]**

***FontUnderline** 值为**[on|{off}]**

FontWeight 文本对象加黑

light： 淡字体

{normal}： 正常字体

demi： 适中或者黑体

bold： 黑体

HorizontalAlignment 文本水平对齐

{left}： 文本相对于它的 **Position** 左对齐

center： 文本相对于它的 **Position** 中央对齐

right： 文本相对于它的 **Position** 右对齐

Position 两元素或三元素向量[**X Y (Z)**]，指出文本对象在三维空间中的位置。单位由 **Units** 属性指定

Rotation 以旋转度数表示的文本方向，

{0}： 水平方向

±90： 文本旋转±90 度

±180： 文本旋转±180 度

±270： 文本旋转±270 度

String 要显示的文本串

Units 位置属性值的度量单位

inches： 英寸

centermeters： 厘米

normalized： 归一化坐标，对象左下角映射到**[0 0]**，右上角映射到**[1 1]**

points： 排字机的点，等于 1/72 英寸

pixels： 排字机的点，等于 1/72 英寸

{data}： 屏幕象素，计算机屏幕分辨率的最小单位

父坐标轴的数据单位

VerticalAlignment		文本垂直对齐
	top:	文本串放在指定的 **Y** 位置顶部
	cap:	字体的大写字母的高度在指定的 **Y** 位置
	{middle}:	文本串放在指定的 **Y** 位置中央
	baseline:	字体的基线在指定的 **Y** 位置
	bottom:	文本串放在指定的 **Y** 位置底部
ButtonDownFcn		当文本对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children		空矩阵，文本对象没有子对象
Clipping		数据限幅模式
	{on}:	在坐标轴界限外的文本的任何部分不显示
	off:	文本数据不限幅
Interruptible		指定 **ButtonDownFcn** 回调字符串是否可中断
	{no}:	不能被其它回调中断
	yes:	可以被其它回调中断
Parent		包含文本对象的坐标轴句柄
***Selected**		值为**[on|{off}]**
***Tag**		文本串
Type		只读的对象辨识字符串，常为 **text**
UserData		用户指定的数据，可以是矩阵、字符串等等
Visible		文本的可视性
	{on}:	文本在屏幕上可视
	off:	文本在屏幕上不可视

表 20.8

曲面对象属性

CData		指定 **ZData** 中每一点颜色的数值矩阵。如果 **CData** 的大小与 **ZData** 不同，**CData** 中包含的图象被映射到 **ZData** 所定义的曲面
EdgeColor		曲面边缘颜色控制
	none:	不画边缘线
	{flat}:	边缘线为单一颜色，由该面 **CData** 的第一个入口项决定。缺省值是 **black**（黑色）
	interp:	各边缘的颜色由顶点的值通过线性插值得到
	A ColorSpec:	3 元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定边缘的单一颜色。缺省值是 **black**（黑色）
EraseMode		消除和重画模式
	{normal}:	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
	backgrount:	通过在图形背景色中重画曲面来消除曲面。这会破坏
	xor:	被消除的曲面后的对象
	none:	用曲面下屏幕颜色执行异或 **OR（XOR）**运算，画出和消除曲面。当画在其它对象上时会造成不正确的颜色当移动或删除曲面时该曲面不会被消除

FaceColor		曲面表面颜色控制
	none:	不画表面，但画出边缘
	{flat}:	第一个 **CData** 入口项决定曲面颜色
	interp:	各面颜色由曲面网格点通过线性插值得到
	A ColorSpec:	3 元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定表面为单一颜色
LineStyle		边缘线形控制
	{-}:	画通过所有网格点的实线
	--:	画通过所有网格点的虚线
	:	画通过所有网格点的点线
	-.:	画通过所有网格点的点划线
	+:	用加号作记号，标出所有的网格点
	o:	用圆圈作记号，标出所有的网格点
	***:**	用星号作记号，标出所有的网格点
	.:	用实点作记号，标出所有的网格点
	X:	用 **X** 符号作记号，标出所有的网格点
LineWidth		边缘线的宽度，缺省值是 **0.5** 点
MarkerSize		边缘线的记号大小，缺省值是 **6** 点
MeshStyle		画行和/或列线
	{both}:	画所有的边缘线
	row:	只画行边缘线
	column:	只画列边缘线
***PaletteMode**		值为**[{scaled}|direct|bypass]**
XData		曲面中点的 **X** 坐标
YData		曲面中点的 **Y** 坐标
ZData		曲面中点的 **Z** 坐标
ButtonDownFcn		当曲面对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children		空矩阵，曲面对象没有子对象
Clipping		数据限幅模式
	{on}:	在坐标轴界限外的曲面的任何部分不显示
	off:	曲面数据不限幅
Interruptible		指定 **ButtonDownFcn** 回调字符串是否可中断
	{no}:	不能被其它回调中断
	yes:	可以被其它回调中断
Parent		包含曲面对象的坐标轴句柄
***Selected**		值为**[on|{off}]**
***Tag**		文本串
Type		只读的对象辨识字符串，常为 **surface**
UserData		用户指定的数据，可以是矩阵、字符串等等
Viible		曲面的可视性
	{on}:	曲面在屏幕上可视
	off:	曲面在屏幕上不可视

表 20.9

补片对象属性

CData	指定沿补片边缘每一点颜色的数值矩阵。只有 **EdgeColor** 或 **FaceColor** 被设为 **interp** 或 **flat** 时才使用
EdgeColor	补片边缘颜色控制
none：	不画边缘线
{flat}：	边缘线为单一颜色，由补片颜色数据的均值指定。缺省值是 **black**（黑色）
interp：	边缘颜色由补片顶点的值通过线性插值得到
A ColorSpec：	三元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定边缘为单一颜色。缺省值是 **black**（黑色）
EraseMode	消除和重画模式
{normal}：	重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式
backgrount：	通过在图形背景色中重画补片来消除该补片。这会破坏被消除的补片后的对象
xor：	用补片下屏幕颜色执行异或 **OR**（**XOR**）运算，画出和消除补片。当画在其它对象上时会造成不正确的颜色
none：	当移动或删除补片时该补片不会被消除
FaceColor	补片表面颜色控制
none：	不画表面，但画出边缘
{flat}：	颜色参量 **c** 中的值决定各补片的表面颜色
interp：	各表面颜色由 **CData** 属性指定的值通过线性插值决定
A ColorSpec：	三元素 **RGB** 向量或 MATLAB 预定的颜色名之一，指定表面为单一颜色
LineWidth	轮廓线的宽度，以点为单位。缺省值为 **0.5** 点
***PaletteModel**	值为**[{scaled}|direct|bypass]**
XData	沿补片边缘点的 **X** 坐标
YData	沿补片边缘点的 **Y** 坐标
ZData	沿补片边缘点的 **Z** 坐标
ButtonDownFcn	当补片对象被选中时，MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，补片对象没有子对象
Clipping	数据限幅模式
{on}：	在坐标轴界限外的补片的任何部分不显示
off：	补片数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}：	不能被其它回调中断
yes：	可以被其它回调中断
Parent	包含补片对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **patch**
UserData	用户指定的数据，可以是矩阵、字符串等等

Visible	补片的可视性
{on}:	补片在屏幕上可视
off:	补片在屏幕上不可视

表 20.10

图象对象属性	
CData	指定图象中各元素颜色的值矩阵。**image（c）**将 **c** 赋给 **CData**。**CData** 中的元素是当前颜色映象的下标
XData	图象 **X** 数据;指定图象中行的位置。如忽略,使用 **CData** 中的行下标
YData	图象 **X** 数据;指定图象中列的位置。如忽略,使用 **CData** 中的列下标
ButtonDownFcn	当图象对象被选中时,MATLAB 回调字符串传递给函数 **eval**；初始值是一个空矩阵
Children	空矩阵，图象对象没有子对象
Clipping	数据限幅模式
{on}:	在坐标轴界限外的图象的任何部分不显示
off:	图象数据不限幅
Interruptible	指定 **ButtonDownFcn** 回调字符串是否可中断
{no}:	不能被其它回调中断
yes:	可以被其它回调中断
Parent	包含图象对象的坐标轴句柄
***Selected**	值为**[on|{off}]**
***Tag**	文本串
Type	只读的对象辨识字符串，常为 **image**
UserData	用户指定的数据，可以是矩阵、字符串等等
Visible	图象的可视性
{on}:	图象在屏幕上可视
off:	图象在屏幕上不可视

20.13 小结

句柄图形函数让用户对图形进行细调，并且显示所建立的图形。每一个图形对象都有一个和它相关的句柄，并可用句柄来操作该对象。对象属性可以用函数 **get** 和 **set** 来修改，以便来定制用户的图形。

本章讨论的函数总结在表 20.11 和表 20.12 中：

表 20.11

句柄图形函数	
set(handle, 'PropertyName' ,Value)	设置对象属性
get(handle, 'PropertyName')	获取对象属性
reset(handle)	将对象属性重设为缺省值
delete(handle)	删除一个对象和它所有的子对象

gcf	获取当前图形的句柄
gca	获取当前坐标轴的句柄
gco	获取当前对象的句柄
findobj('PropertyName' ,Value)	获取具有指定的属性值的对象的句柄
waitforbuttonpress	等待键或鼠标按钮在图形中按下
figure('PropertyName' ,Value)	创建图形对象
axes('PropertyName' ,Value)	创建坐标轴对象
line(X,Y, 'PropertyName' ,Value)	创建线条对象
text(X,Y,S, 'PropertyName' ,Value)	创建文本对象
patch(X,Y,C, 'PropertyName' ,Value)	创建补片对象
surface(X,Y,Z, ' PropertyName' ,Value)	创建曲面对象
image(C, 'PropertyName' ,Value)	创建图象对象

表 20.**12** 中是本章所提到的精通 *MATLAB* 工具箱中的函数：

表 20.**12**

精通 MATLAB 工具箱句柄图形函数	
mmgcf	如当前的图形存在，获取它的句柄
mmgca	如当前的坐标轴存在，获取它的句柄
mmzap(T)	用鼠标删除类型 **T** 的图形对象
mmpx2n(X)	象素到归一化坐标的转换
mm2px(X)	归一化坐标到象素的转换
mmline Name Value…	用鼠标设置线属性
mmaxes Name Value…	用鼠标设置坐标轴属性
mmcont2(X,Y,Z,N,C)	用用户自定义颜色作二维等值线图
mmcont3(X,Y,Z,N,C)	用用户自定义颜色作三维等值线图
mmtile	以平铺形式安排图形窗口
mmpager Name Value…	设置打印的缺省纸张属性

关键词索引

chap20

graphics routine 图形例程
handle 句柄
object 对象
property 属性
stacking order 堆积次序
pixel 象素
Normalized coordinates 归一化坐标
landscape 景象（横向）
portrait 肖像（纵向）
full page 全幅
documented property 文件式属性
undocumented property 非文件式属性
root 根（对象）
figure 图形窗口（对象）
axis 坐标轴（对象）
line 线条（对象）
surface 曲面（对象）
text 文本（对象）
patch 补片（对象）
image 图象（对象）
clipping 限幅

第21章　创建图形用户界面

用户界面是人，即用户与计算机或计算机程序的接触点或交互方式，是用户与计算机进行信息交流的方式。计算机在屏幕显示图形和文本，若有扬声器还可产生声音。用户通过输入设备，如：键盘、鼠标、跟踪球、绘制板或麦克风，与计算机通讯。用户界面设定了如何观看和如何感知计算机、操作系统或应用程序。通常，多是根据悦目的结构和用户界面功能

的有效性来选择计算机或程序。

图形用户界面或GUI是包含图形对象，如：窗口、图标、菜单和文本的用户界面。以某种方式选择或激活这些对象，通常引起动作或发生变化。最常见的激活方法是用鼠标或其它点击设备去控制屏幕上的鼠标指针的运动。按下鼠标按钮，标志着对象的选择或其它动作。

与上一章讨论MATLAB句柄图形功能的相同方式，它让用户按规定设计MATLAB显示信息的方法，本章所描述的图形用户界面的功能，它让用户定制用户与MATLAB的交互方式。命令窗口不是唯一与MATLAB的交互方式。

本章将说明图形句柄**uicontrol** 和**uimenu**对象的使用，把图形界面加到MATLAB的函数和M文件。**uimenu**对象能在图形窗口中产生下拉式菜单和子菜单。**uicontrol**对象能建立如按钮，滚动条，弹出式菜单以及文本框等对象。

MATLAB在**demo**命令中包含了GUI功能的极好例子。

```
>>  demo
```

研究该命令，以了解**uimenu**和**uicontrol**如何给MATLAB函数提供交互输入。

21.1 谁创建图形界面GUI?为什么?

在运行了demo例子后，很可能会问“为什么要在MATLAB中建立一个*GUI*？”这是一个很好的问题，简单的回答是可能并不需要．使用MATLAB来分析数据，求解问题，绘制结果的绝大多数的人，并不会发现GUI工具很有用。

但另一方面，GUI可以在MATLAB中生成非常有效的工具和应用程序，或是建立演示工作的交互式界面。

生成用户图形界面的最常见的理由：

编写一个需多次反复使用的实用函数，菜单、按钮、文本框作为输入方法具有意义；或

编写函数或开发应用程序供别人使用；或

创建一个过程、技术或分析方法的交互式示例；或

认为GUI的简洁，性能良好，并且想实践一下。

许多基于GUI的工具函数包含在精通MATLAB工具箱中，将在后续章节进行讨论。其它由MATLAB用户编制的工具和实用程序装入MATLAB的GUI函数。工具的大多数可在**Mathworks** 匿名**FTP**节点和其它资源中获得。

在我们开始讨论之前，记住对“句柄图形”的理解是设计和实现GUI的先决条件，如果你跳过了前一章，现在应重新回去阅读。

21.2 GUI对象层次结构

正如我们在上一章所展示的那样，由图形命令生成的每一事物是一个图形对象。图形对象不仅包括**uimenu**和**uicontrol**对象，而且还包括图形、坐标轴和他们的子对象。让我们从另一个角度来看这一层次结构。计算机的屏幕本身是根结点，图形是根对象的子对象，坐标

轴，**uimenu** ，**uicontrol**是图形的子对象。

根可以包括多个图形，每个图形含有一组或多组坐标轴以及其子对象，每个图形也可以有一个或多个与坐标轴无关的**uimenu**和**uicontrol**。虽然**uicontrol**对象无子对象结点，但他们确实具有多种类型。**uimenu**对象常将其它的**uimenu**对象作为其子对象。

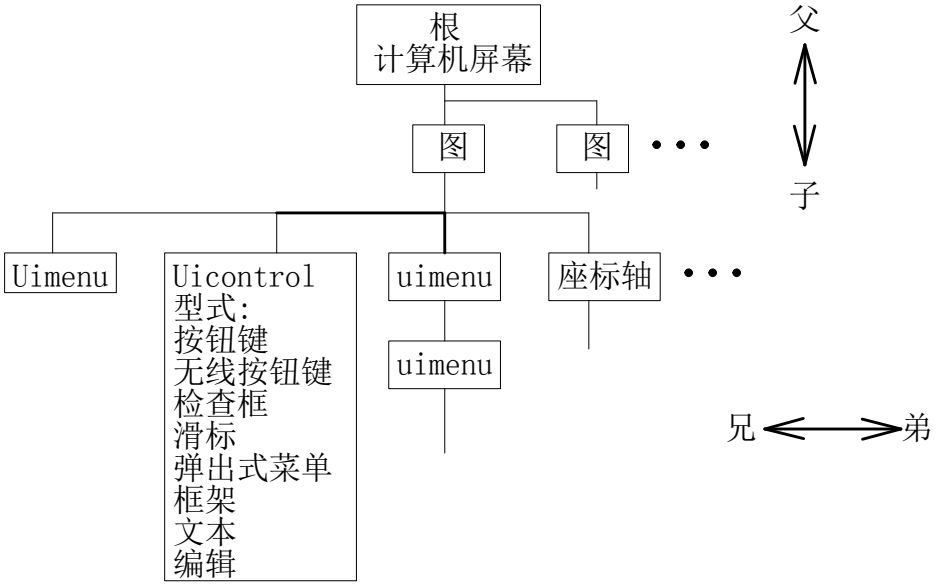


图21.1　GUI对象层次结构图

运行MATLAB的不同型号的计算机或平台上，产生不同的图形显示。Unix工作站使用不同的X Window系统，具有几个窗口程序，如**mwn**或**twm**以控制显示的布局。PC机靠Microsoft Windows或Windows NT进行窗口管理，Macintosh计算机用Macintosh工具箱程序作窗口。虽然在各种平台上，显示看起来有很大的不同，但在很多的情况下，句柄图形的编码是一致的。MATLAB在内部处理平台和窗口系统的差别。体现句柄图形例程的函数，包括应用**uimenu**和**uicontrol**对象的函数，通常运行在所有平台。存在已知差异的地方将在本章后面给出。

21.3　菜单

在每一个窗口系统中使用菜单让用户选择命令和选项。通常在显示屏或窗口的顶部有一菜单条。移动鼠标指针到菜单标志上按下鼠标按键，顶层菜单就被选中，一列菜单项就从菜单标志拉下来。这种款式菜单就叫下拉式菜单。按下鼠标将指针移动至菜单项并松开鼠标，则完成菜单项的选择。MS-Windows 和一些X Window系统平台还提供另一种选择菜单的方法。在顶层菜单上按下并松开鼠标，或称单击鼠标，则打开下拉菜单。然后，移动鼠标指针至下拉菜单项再次单击鼠标，就选择菜单项。在下拉菜单中选择一项就引起动作的发生。

一个菜单项还可用自己的菜单项列表而作为子菜单。子菜单项在子菜单的标志右边显示小三角或箭头以表示菜单还有更多子菜单项可供选择。如果子菜单的菜单项被选择，另一个具有更多菜单项的菜单显示在此菜单的右边的下拉菜单中。有时这种菜单称之行走菜单。选

中其中一个菜单项也引起某些动作的产生。

子菜单可以嵌套，但层次的数目受到窗口系统及有用资源的限制。

菜单的布置

Macintosh在显示屏的顶部使用包含下拉菜单标题的菜单条。选择图形窗口时，菜单条变化以反映激活的图形窗中可利用的选项。标准的Macintosh菜单标题包括**Apple**、 **File**、**Edit**、 **Window** 和**Help** 或**Balloon Help**。由**uimenu**所加的菜单标题放在**Window**和**Help**之间。如果想从菜单条中删去**File**、 **Edit**、和 **Window**菜单标题，可以使用**Set**命令。

```
>> set(Hf_fig1，‘ Menubar ‘，‘ none ‘)
```

Apple和**Help**不可从菜单条中删除。同样，标准的菜单是用以下的命令恢复：

```
>> set(Hf_fig1，‘Menubar‘，‘figure‘)
```

在Microsoft窗口系统下，菜单条位于图形窗口的顶部。每个图形窗口有自己的菜单条，它包含**File**，**Edit**，**Window**和**Help**标题。由**uimenu**所加的菜单标题放在**Help**之后。可以使用与上面相同的**Set**命令从菜单条中删去或恢复所有的标准菜单。

在X Window系统工作站上没有MATLAB标准图形窗口菜单的标题。窗口管理器可将菜单条放置在屏幕上每一个窗口上端，但这些菜单条与MATLAB菜单无关。当建立第一个**uimenu**对象时，MATLAB 在图形窗口的顶部边缘生成自己的菜单条。

建立菜单和子菜单

我们采用函数**uimenu**建立菜单项。**uimenu**的句法与其他对象创建函数相似。如，

```
>> Hm_1=uimenu(Hx_parent，‘PropertyName‘，PropertyValue，...)
```

其中**Hm_1**是由**uimenu**生成的菜单项的句柄，通过设定**uimenu**对象的属性值‘**PropertyName**’，**PropertyValue**这对命令定义了菜单特性；**Hx_parent**是缺省的父辈对象的句柄，必须是图形和**uimenu**对象。

uimenu对象中最重要的属性是'Label'和'Callback'。'Label'属性值是菜单条和下拉菜单项上的文本字符串，以确认菜单项。'Callback'属性值是MATLAB字符串，当选中菜单项时，它传给eval，用以执行。

菜单举例

下面的例子用函数**uimenu**将简单菜单加到当前的图形窗口中。这里提出的例子说明如何只用几个MATLAB命令来建立工作菜单。后面的例子将详细地讨论**uimenu**的命令和属性。

下例用两个下拉菜单将菜单条加到当前窗口中。首先，建立名为Example的顶部菜单输入。

```
>> Hm_ex=uimenu(gcf，‘Label‘，‘Example‘);
```

在此菜单下有两个菜单项。第一项标志为Grid，切换坐标轴格栅的状态。

```
>>  Hm_exgrid=uimenu(Hm_ex，‘Label‘，‘Grid‘，‘Callback‘，‘Grid‘)；
```

注意，句柄Hm_ex是用于与上层菜单相关联的。这项**uimenu**按**eval**的要求出现在上层菜单之下。还需注意的是属性‘Callback‘的值，它是一个带引号的字符串。

Example下的第二项标志为**View**，并带有子菜单。

```
>>  Hm_exview=uimenu(Hm_ex，‘Label‘，‘View‘)；
```

View菜单有两项选择2-D和3-D视图。

```
>>  Hm_ex2d=uimenu(Hm_exview，‘Label‘，‘2-D‘，‘Callback‘，‘view(2)‘)；

>>  Hm_ex3d=uimenu(Hm_exview，‘Label‘，‘3-D‘，‘Callback‘，‘View(3)‘)；
```

注意以上这些是**View**的子菜单，因为它们指定**Hm_exview**作为其父对象。
现在，将第二个顶层菜单加到标题为**Close**的菜单条中。

```
>>  Hm_ex=uimenu(gcf，‘Label‘，‘Close‘)；
```

由该顶层菜单Close加入了两个菜单项。第一项关闭图形窗口，第二项使图形窗口打开，但去掉用户菜单。

```
>>  Hm_clfig=uimenu(Hm_close，‘Label‘，‘Close Figure‘，‘Callback‘，‘Close‘)；

>>  Hm_clmenu=uimenu(Hm_close，‘Label‘，‘Remove Menu‘，...
                ‘Callback‘，‘delete(Hm_ex)；  delete(Hm_ex)；  drawnow‘)；
```

在精通MATLAB工具箱的脚本文件**mmenu1.m**中含有上面的例子。所以你可以运行**mmenu1.m**来验证上例。

菜单属性

如上所示，同句柄图形函数一样，在建立图形对象时可定义**uimenu**属性，或用set改变属性。所有可设定的属性，包括标题、菜单颜色、甚至回调字符串都可以用set来改变。这种功能十分便于迅速地定制菜单和属性。

表21.1列出了MATLAB 4.2版本中的**uimenu**对象的属性及其属性值。带有*的属性是非文件式的，使用时需加小心。在括号{}内的属性值是缺省值。

表21.1

Uimenu 对象的属性

Accelerator		指定菜单项等价的按键或快捷键。对于X-windows，按键顺序是**Control**- 字符；Macintosh系统，按键顺序是**Command** - 字符或# - 字符
BackgroundColor		**uimenu**背景色,是一个3元素的RGB向量或MATLAB预先定义的颜色名称。缺省的背景色是**亮灰**色
Callback		MATLAB回调字符串，选择菜单项时，回调串传给函数**eval**；初始值为空矩阵
Checked		被选项的校验标记
	on:	校验标记出现在所选项的旁边
	{**off**}:	校验标记不显示
Enable		菜单使能状态
	{**on**}:	菜单项使能。选择菜单项能将**Callback**字符串传给
	off:	**eval**
		菜单项不使能，菜单标志变灰。选择菜单项不起任何作用。
ForegroundColor		**uimenu**前景（文本）色,是一个三元素的RGB向量或MATLAB预
		先定义的颜色名称。缺省的前景色是**黑色**
Label		含有菜单项标志的文本串。在PC系统中，标记中前面有‘&‘，定
		义了快捷键，它由**Alt - 字符**激活
Position		**uimenu**对象的相对位置。顶层菜单从左到右编号，子菜单从上至
		下编号
Separator		分割符 - 线模式
	on:	分割线在菜单项之上
	{**off**}:	不画分割线
*Visible		**uimenu**对象的可视性
	{**on**}:	**uimenu**对象在屏幕上可见
	off:	**uimenu**对象不可见
ButtonDownFcn		当对象被选择时，MATLAB的回调串传给函数**eval**。初始值为空矩
		阵。
Children		其它**uimenu**对象的句柄。
Clipping		限幅模式
	{**on**}:	对uimenu对象无效果
	off:	对uimenu对象无效果
DestroyFcn		仅用于Macintosh 4.2 版本。没有文本说明。
Interrruptible		指明**ButtonDownFcn**和**CallBack**串可否中断
	{**no**}:	回调不可中断
	yes:	回调串可中断
Parent		父对象的句柄；如果**uimenu**对象是顶层菜单，则为图形对象；若
		uimenu是子菜单，则为父的**uimenu**对象句柄

*Select	值为**[on|off]**
*Tag	文本串
Type	只读对象辩识串，通常为**uimenu**
UserData	用户指定的数据。可以是矩阵，字符串等等
Visible	**uimenu**对象的可视性
{**on**}：	**uimenu**对象在屏幕上可见
off：	**uimenu**对象不可见

属性值仅仅定义了**uimenu**对象的性质并且控制菜单如何显示；它们也决定了选择菜单项所引起的动作。其中某些性质将在下面更详细地讨论。

菜单快捷键

' Label ' 属性义定了出现在菜单或菜单项中的标志。它也可以用来定义Microsoft Windows系统的快捷键：标志字符串中，在所需字符前加上&，例如：

```
>>  Hm_top=uimenu( ' Label ' ,  ' Example ' );

>>  uimenu(Hm_top,  ' Label ' ,  ' &Grid ' ,  ' CallBack ' ,  ' grid ' );
```

它定义了键盘上**G**为快捷键。菜单项标志将以Grid形式出现在菜单上。为激活快捷键，在选择图形窗口时按Alt键并按下**G**键。快捷键不一定是字符串的第一字符。下例中**R**为快捷键：

```
>>  uimenu(Hm_top,  ' Label ' ,  ' G & rid ' ,  ' CallBack ' ,  ' grid ' );
```

则标志以**Grid**形式出现在菜单上。

Macintosh平台用 ' **Accelerator** ' 属性而不是 ' **Label** ' 来定义快捷键。在Macintosh

```
>>  uimenu(Hm_top,  ' Label ' ,  ' Grid ' ,  ' Accelerator ' ,  ' G ' ,  ' CallBack ' ,  ' grid ' );
```

定义**G**为快捷键，菜单标志以**Grid#G**的形式出现。为激活快捷键，选择图形窗口时，按**Command**键或#键并按下**G**键。

在Macintosh上不可为顶层菜单定义快捷键。另外，已经定义在标准Macintosh上的快捷键，如**Command C**或# **C**不撤除标准Macintosh菜单，就不能复制。

在X-Window 系统中定义和使用快捷键与Macintosh相似，但仍存在某些差异。同Macintosh一样，用 **' Accelerator '** 属性而不是 **' Label '** 属性来定义快捷键。但快捷键在菜单上显示不同。例如，

```
>>  uimenu(Hm_top,  ' Label ' ,  ' Grid ' ,  ' Accelerator ' ,  ' G ' ,  ' CallBack ' ,  ' grid ' );
```

定义字母**G**为快捷键。菜单标志常以Gr**id<Ctrl>-G**出现在菜单上。未使用快捷键，需按**Control**键并按下**G**键。同Macintosh一样，不可为顶层菜单定义快捷键。

虽然我们没有对此验证，但MATLAB用户手册指出当给定相同命令时，用X的某些工作站动作不一样。如果顶层菜单标志中有带下划线的字符，就可按**Meta**键并按下下划字符键来选择菜单。如果子菜单项标志中含有带下划线的字符，就按下该字符键来选择菜单项。请参阅键盘使用说明，以为系统确定合适的Meta键。

菜单的外观

影响菜单的布置和外观的三个属性为 **'Position'**， **'Checked'**，和 **'Separator'**.**uimenu**对象的 **'Position'** 属性值是一个整数，它定义了相对于其它菜单和菜单项的位置。在生成菜单时，设定 **'Position'** 属性。菜单条的最左端的菜单条和下拉菜单中的上端菜单项处在位置1。

设置‘**Position'** 属性可以重新排列菜单位置。考虑下面的例子：

```
>>  Hm_1=uimenu( ‘ Label ‘ ， ‘ first ‘ )；      %  Create two menus

>>   Hm_2=uimenu( ‘ Label ‘ ， ‘ Second ‘ )；

>>  get(Hm_1， ‘ Position ‘ )              %  Check the locations
ans=
     1

>>  get(Hm_2， ‘ Position ‘ )
ans=
     2

>>  set(Hm_2， ‘ Position ‘ ，1)              %  Change menu order

>>  get(Hm_1， ‘ Position ‘ )              %  check the locations

ans=
     2

>>  get(Hm_2， ‘ Position ‘ )

ans=
     1
```

注意，当一个**uimenu**的 **'Position'** 属性改变，就将移动其它**uimenu**以适应此变化，它们的 **'Position'** 属性均更新。子菜单中菜单选项的编号以同样的方式重新排列。

属性 **'Checked'** 的值使校验标记出现在菜单项标志的左边。缺省值为 **'off'** 。命令

```
>>  set(Hm_item， ‘ Checked ‘ ， ‘ on ‘ )
```

使校验标记出现在**Hm_item uimenu**标志的旁边。对于创建代表属性的菜单项，该命令十分

有用。例如，

```
>>Hm_top=uimenu(‘Label‘，‘Example‘);
>>Hm_box=uimenu(Hm_top，‘Label‘，‘Axis Box‘，...‘CallBack‘，[...
                ‘if strcmp(get(fca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
                ‘seet(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
               set(Hm_box，‘‘Checked‘‘，...off‘‘)，‘，...
                ‘else，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
                ‘set(Hm_box，‘‘Checked‘‘，‘‘on‘‘)，‘，...‘end‘]);
```

建立了以**Axis Box**为标志的下拉菜单项。当选中该项时，就运行回调字符串所表示的命令。回调字符串确定了当前坐标轴的**'Box'**属性值，并适当地设定坐标轴的‘**Box'**属性及**uimenu**的**'Checked'**属性。该例子可从精通MATLAB工具箱的M脚本文件**mmenu2.m**中得到。

可以通过改变**uimenu**的**'Label'**属性以反映菜单项当前的状态。下面的例子（在**mmenu3.m**中)改变了菜单项标志本身，而不是加一个校验标记：

```
>>  Hm_top=uimenu(‘Label‘，‘Example‘);

>>  Hm_box+uimenu(Hm_top，‘Label‘，‘Axis Box‘，...
                ‘CallBack‘，[...
                ‘if strcmp(get(gca，‘‘Box‘‘)，‘‘on‘‘)，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘off‘‘)，‘，...
               set(Hm_box，‘‘Label‘‘，‘‘Set Box On‘‘)，‘，...
                ‘else，‘，...
                ‘set(gca，‘‘Box‘‘，‘‘on‘‘)，‘，...
                ‘set(Hm_box，‘‘Label‘‘，‘‘Set box Off‘‘)，‘，...
                ‘end‘]);
```

使用**' Separator'**属性可将下拉菜单分成局部组。如果一个**uimenu**项的**' Separator'**属性是**'on'**，则在下拉菜单中显示此项时，此项的上端有一条水平线将其与前面的菜单项隔开。缺省值是**'off'**，但在菜单生成时可以改变，

```
>>  Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘，‘on‘);
```

或是在以后使用**set**命令

```
>>  set(Hm_box，‘Separator‘，‘on‘);
```

顶层菜单忽略**'Seprator'**的属性值。

使用**'Seperator'**属性可将下拉菜单项分成若干逻辑组。如果对**uimenu**项**'Seperator'**属性设置为**'on'**，用在一条其上面的水平线来表现该项，将其与前面的菜单项形象地区分开。缺省值为**'off'**，但在生成菜单时可以改变，

```
>> Hm_box=uimenu(Hm_top，‘Label‘，‘Box‘，‘Seperator‘.‘on‘)
```

或是在以后使用**set**命令：

```
>> set(Hm_box.‘Seperator‘，‘on‘)
```

顶层菜单忽略 **'Seperator'** 属性值。

颜色控制

uimenu对象可设置两个颜色属性。 **'BackGroundColor'** 属性控制填充菜单背景的颜色。缺省值是浅灰。另一颜色属性为 **'ForeGroundColor'**，它确定菜单项文本的颜色，缺省值是黑色。

颜色属性同样能很好地用于顶层菜单条和下拉菜单。颜色可以用来表示状态信息或简单加上菜单的特 色。例如，挑选线段颜色。在子菜单中，各菜单项的背景可以填充合适的色彩。

```
>> Hm_green=uimenu(Hm_color，'Label'，'Green'，'BackgroundColor'，'g'，...
                    'Callback'，'set(Hl_line，'Color'，'g')');
```

菜单项去能

改变对象**uimenu**的 **'Enable'** 值或 **'Visible'** 属性可使菜单项暂时去能。 **'Enable'** 属性通常设
为 **'on'** 。当 **'Enable'** 属性 ' 设为 **'off'** 时，标志字符串变灰，菜单项去能。在这种状态下，菜单项保持可见但不能被选择。此属性可用来将不恰当的菜单选择去能。

下面的例子(**mmenu4.m**)说明了用两个菜单项和 **'Enable'** 属性来设定坐标轴的 **'Box'** 属性的另一种方法。

```
>> Hm_top = uimenu('Label'，Example');

>> Hm_boxon = uimenu(Hm_top，'Label'，'Set Box On'...'CallBack'，[...
            'set(gca，  'Box'  '，  'on'  ')，  '，...
            'set(Hm_boxon，  'Enable'  '，  'off'  ')，  '，...
            'set(Hm_boxoff，  'Enable'  '，  'Enable'  '，  'on'  ')
'']);

>> Hm_boxoff = uimenu(Hm_top，'Label'，Set Box Off'，...'Enable'，'
off'，...'CallBack'，[...
            'set(gca，  'Box'  '，  'off'  ')，  '，...
            'set(Hm_boxon，  'Enable'  '，  'on'  ')，  '，...
            'set(Hm_boxoff，  'Enable'  '，  'off'  ')']);
```

设定**'Visible'** 属性为 **'off'**， 可将菜单项完全隐藏。菜单项象是从屏幕中消失，而其它菜

单项改变了在显示器上的位置以填补由当前不可见菜单造成的空隙。然而，不可见的菜单仍然存在，而且**uimenu** 对象
的 **'Position'** 属性值也不改变。当属性 **'Visible'** 又重新设为 **'on'** 时，菜单项重新出现在正常的位置。

这个性质可以用来暂时地撤消一个菜单。下面的例子(**mmenu5.m**)建立了两个顶层菜单和两个菜单项。

```
>> Hm_control = uimenu('Label', 'Control');

>> Hm_extra = uimenu('Label', 'Extra');

>> Hm_limit = uimenu(Hm_control, 'Label', 'Limited Menus', ...
                'CallBack', 'set(Hm_extra, ''Viseble'', ''off'')
');

>> Hm_full = uimenu(Hm_control, 'Label', 'Full Menus', ...
            'CallBack', 'set(Hm_extra, ''Visible'', ''on'')');
```

当选择了**Limited Menus**项时，**Extra**菜单就从菜单条中消失。当选择了**Full Menus**项时，**Extra**菜单又重新显示在原来的位置的菜单条上。

回调属性

'Callback' 属性值是一个MATLAB字符串，MATLAB将它传给函数**eval**并在**命令窗口工作空间**执
行。它对于函数M文件有重要的隐含意义，我们将在本章后面继续讨论这一属性。

因为 **'Callback'** 属性必须是字符串，所以在字符号内多重MATLAB命令、后续行以及字符串都会使必需的句法变得十分复杂。如果有不止一个命令要执行，命令间必须适当地分隔开来。例如，

```
>>uimenu('Label', 'Test', 'CallBack', 'grid on; set(gca, ''Box'', ''on'
')');
```

把一个字符串传给**eval**，使命令

```
>> grid on; set(gca, 'Box', 'on')
```

在命令窗口工作空间中执行。这是合法的句法，因为命令用逗号或分号隔开，多重命令可输入到同一命令行中。在定义回调函数时，也遵循MATLAB规定，即在已引用的字符串内，用两个单引号来表示单引号。

字符串可以串接起来生成一个合法MATLAB字符串，只要把它们括在方括号中。

```
>>uimenu('Label', 'Test', 'CallBack', ['grid on, ', ' set(gca, ''Box'', '
''on '' )']);
```

注意字符串 **'grid on'** 含有所需的逗号以分隔两个命令。

如果使用了续行号，上述命令可写为

```
>>uimenu('Label', 'Test', ...
        'CallBack', [...
        'grid on, ', ...
        '  set(gca, ' 'Box' ', ' 'on' ')'...
       ]);
```

上例中命令行被分隔，每行的末尾加上了三个句号表示命令的继续。注意到上列单行的所有元素都被保留，包括字符串分隔命令的逗号。在 'grid on, …' 行中最后引号后的逗号是可选的；下一行开始的空格起相同的作用。欲了解详情，请参阅前面关于建立行向量的章节。

如果引号、逗号和括号不正确输入，MATLAB将给出警告；但在复杂回调字符串中很难寻找错误

的。为了使错误最少，对包含MATLAB语句的回调字符串请记住以下的一些规则：

把整个回调字符串括在方括号中，不要忘记最后的右括号'）'。

把各语句括上单引号。

已引用的字符串内，要用双引号。如：**'quoted': 'a ' 'quoted' ' string';'Quote ' 'a ' ' 'quoted' ' ' string' 'now'**。在引号后要用逗号或空格结尾。

除了最后一句，各语句在**引号内**要以逗号或分号结尾；在**引号后**要用逗号或空格结尾。

有后续行的各行要以三个句号(...)结尾。

前面的例子之一**mmenu4.m**是所涉及的回调字符串句法的很好说明。

```
>> Hm_top = unimenu('Label', 'Example');

>> HM_boxon = uimenu(Hm_top,...
                     'Label', 'Set' Box on', ...
                     'CallBack', [...
                          'set(gca,"Box","on"), ', ...
                          set(Hm_boxon,"Enable","off"), ', ...
                          set(Hm_boxoff,"Enable","on")'];

>> Hm_boxoff = uimenu(Hm_top,...
                      'Label', 'Set Box off', ...
                      'Enable', 'off' ', ', ...
                      'CallBack', [...
                           'set(gca,"Box","off", ', ...
                           'set(Hm_boxon,"Enable","on"), ', ...
                           'set(Hm_boxoff'Enable","off")'];
```

上例中还引出了关于回调函数另一个重点，在变量**Hm_boxoff**定义之前，在回调串中用**Hm_boxoff**替代**Hm_boxon**。因为回调串只是一个字符串，MATLAB 不会给出警告，而且仅在**uimenu**被激活并将字符串传给**eval**时才由MATLAB执行。它隐含有函数M文件的设计和测试，这将在本章后面讨论。

M文件的举例

下例将演示一组简单菜单的生成。该例子包含在精通MATLAB工具箱的函数M文件**mmenus**中。正如下面所示的那样，这个函数文件被分隔成了若干块，以便于讨论函数的各个方面。

首先，定义一个函数并在当前的图形中用顶层**Line**菜单建立菜单条，该菜单分别含有三个子菜单：**Line Style**，**Line Width**，**Line Color**。

```
function mmenus()
%   MMENUS Simple menu example.
%   MMENUS uses waitforbuttonpress and gco in callback strings
%   to let the user make a menu selection and then select an object
%   by clicking on it with the mouse. The callback strings then use
%   the set function to apply the property value to the selected
%   object.
%   Copyright (c) 1996 by Prentice-Hall，Inc.

Hm_line   =   uimenu(gcf，  'label'.'Line');
Hm_lstyle   =   uimenu(Hm_line，  'label'，  'Line Style');
Hm_lwidth   =   uimenu(Hm_line，  'label'，  'Line width');
Hm_lcolor   =   uimenu(Hm_'label，  'Line Color');
```

其次，使用**waitforbuttonpress**和**gco**得到当前对象的句柄，确认它为一个线对象，并采用适当的**'Linestyle'**值。注意这些菜单项句柄以后不再使用，所以它们不必保存。

```
uimenu(Hm_lstyle，  'Label'，Solid'，...
            'CallBack'，('waitforbuttonpress；  '，...
                    'if get(gco，"type")=="line"，  '...
                        'set(gco，"LineStyle"，"-")，  '...
                    'end']);

uimenu(Hm_lstyle，  'Label'，Dotted'，...
            'Callback'，['waitforbuttonpress；  '，...
                    'if get(gco，"Type")=="line"，  '，...
                        'set(gco ，"LineStyle"，"，"：")，  '...
                    'end']);

uimenu(Hm_lstyle，  'Label'，Dashed'，...
              'Callback'，['waitforbuttonpress；  '，...
```

```
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineStyle"，"-"：")， '...
                    'end']);

    uimenu(Hm_lstyle， 'Label'，DashDot'，...
                'Callback'，['waitforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineStyle"，"-"：")， '...
                    'end']);
```

现在，对**Line width**子菜单项也做同样的操作：

```
    uimenu(Hm_lwitdth， 'Label'， 'Default'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，0.5)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thick'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，2.0)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thicker'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，3.0)， '，...
                    'end']);

    uimenu(Hm_lwitdth， 'Label'， 'Thickest'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"LineWidth"，4.0)， '，...
                    'end']);
```

对**Line Color**子菜单项也做同样的操作。将菜单项背景加色并在合适时改变文本颜色。

```
    uimenu(Hm_lcolor， 'Label'， 'yellow'，...
            'BackgroundColor'， 'y'，...
            'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                        'set(gco ，"color"， '"y")， '，...
                    'end']);
```

```
uimenu(Hm_lcolor， 'Label'， 'Magenta'， ...
          'BackgroundColor'， 'm'， 'ForegroundColor'， 'w'...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"m")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'yan'，...
          'BackgroundColor'， 'c'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"c")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Red'，...
          'BackgroundColor'， 'r'， 'ForegroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"r")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Green'，...
          'BackgroundColor'， 'g'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"g")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Blue'，...
          'BackgroundColor'， 'b'， 'ForegroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"b")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'White'，...
          'BackgroundColor'， 'w'，...
          'CallBack'，['witforbuttonpress； '，...
                    'if get(gco，"Type")=="line"， '，...
                         'set(gco ，"color"， '"w")， '，...
                    'end']);

uimenu(Hm_lcolor， 'Label'， 'Black'，...
```

'BackgroundColor'， 'k'， 'ForegroundColor'， 'w'， ...
'CallBack'，['witforbuttonpress； '， ...

```
        ' if get(gco，"Type")=="line"， ' ，...
            ' set(gco ，"color"， '"k")， ' ，...
        ' end ' ])；
```

为使这一函数更完全，可用同样方法增加另外的菜单以改变坐标轴、曲面、补片和图形的属性。例如，下面加上了一个**Color**映象以改变图形颜色的映象。

```
Hm_cmap=uimenu(gcf， ' Label ' ， ' Color Map ' )；

uimenu(Hm_cmap， ' Label ' ， ' Lighter ' ， ' CallBack ' ， ' brighten(.3) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Darker ' ， ' CallBack ' ， ' brighten(-.3) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Default ' ， ' CallBack ' ， ' colormap("default") ' )；
uimenu(Hm_cmap， ' Label ' ， ' Gray ' ， ' CallBack ' ， ' colormap(gray) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Hot ' ， ' CallBack ' ， ' colormap(hot) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Cool ' ， ' CallBack ' ， ' colormap(cool) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Bone ' ， ' CallBack ' ， ' colormap(bone) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Copper ' ， ' CallBack ' ， ' colormap(copper) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Pink ' ， ' CallBack ' ， ' colormap(pink) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Prism ' ， ' CallBack ' ， ' colormap(prism) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Jet ' ， ' CallBack ' ， ' colormap(jet) ' )；
uimenu(Hm_cmap， ' Label ' ， ' Flag ' ， ' CallBack ' ， ' colormap(flag) ' )；
uimenu(Hm_cmap， ' Label ' ， ' HSV ' ， ' CallBack ' ， ' colormap(hsvflag) ' )；
```

最后，加上含有两个菜单项的**Quit**菜单。其中，**Close Figure**关闭图形；**Remove Menus**让图形打开但删除用户的顶层菜单及所有它的子菜单。**drawnow**命令立即删除菜单。

```
Hm_quit=uimenu(GCF， ' lABEL ' ， ' quit ' )；

uimenu(Hm_quit， ' Label ' ， ' Close Figure ' ， ' CallBack ' ， ' close；return ' )；

uimenu(Hm_quit， ' Label ' ， ' Remove Menu ' ，...
        ' Callback ' ，[...
            ' delete(findobj(gcf，"Type，"uimenu"，"parent"，gcf))， ' ，...
            ' drawnow ' ])；
```

精通MATLAB工具箱中有许多采用这种技术的函数和其它建立有用的基于对象工具的函数。其中许多函数将在本章后面详细讨论。

21.4 控制框

在各计算机平台上，窗口系统都采用控制框和菜单，让用户进行某些操作，或设置选项或属性。控制框是图形对象，如图标、文本框和滚动条，它和菜单一起使用以建立用户图形界面，称之为窗口系统和计算机窗口管理器。

MATLAB控制框，又称**uicontrol**，与窗口管理器所用的函数十分相似。它们是图形对象，可以放置在MATLAB的图形窗中的任何位置并用鼠标激活。MATLAB的 **uicontrol**包括按钮、滑标、文本框及 弹出式菜单。

由MATLAB生成的**uicontrol**对象在Macintosh、MS-Windows 和X Window系统平台上，有稍微不同的外观，因为窗口系统表达图形对象的方法是不同的。但是，功能本质是相同的，所以相同的MATLAB编码将生成同样的对象，它在不同平台完成同样的功能。

Uicontrol由函数**uicontrol**生成。常用句法与前面所讨论的**uimenu**相同。

```
>>Hc_1=uicontrol(Hf_fig， 'PropertyName'，PropertyValue，...）
```

其中，**Hc_1**是由函数**uicontrol**生成**uicontrol**对象的句柄。通过设定**uicontrol**对象的属性值**'PropertyName'**， **'PropertyValue'** 定义了**uicontrol**的属性；**Hf_fig**是父对象的句柄，它必须是图形。如果图形对象句柄省略，就用当前的图形。

建立不同类型的控制框

MATLAB共有八种不同类型或型式的控制框。它们均用函数**uicontrol**建立。属性 **'Style'** 决定了所建控制框的类型。 **'Callback'** 属性值是当控制框激活时，传给**eval**在命令窗口空间执行的MATLAB字符串。

下面将分别对八种**uicontrol**对象进行讨论，并用示例说明。**uicontrol**对象的属性更为透彻的讨论和应用中更为完整的例子将在以后给出。

按钮键

按钮键，又称命令按钮或只叫按钮，是小的长方形屏幕对象，常常在对象本身标有文本。将鼠标指针移动至对象，来选择按钮键**uicontrol**，单击鼠标按钮，执行由回调字符串所定义的动作。按钮键的 **'Style'** 属性值是 **'pushbutton'** 。

按钮键典型地用于执行一个动作而不是改变状态或设定属性。下面的例子（**mmctl1.m**）建立标志为**Close**的按钮键**uicontrol**。当激活该按钮时，**close**关闭当前的图形。以象素为单位的**'Position'** 属性定义按钮键的大小和位置，这是缺省的 **'Units'** 属性值。属性 **'String'** 定义了按钮的标志。

```
>>Hc_close=uicontrol(gcff， 'Style'， 'push'，...
                    'Position'，[10 10 100 25]，...
                    'String'，Close'，...
                    'CallBack'， 'close)；
```

无线按钮

无线按钮，又称选择按钮或切换按钮，它由一个标志并和标志文本的左端一个小圆圈或小菱形所形
成。当选择时，圆圈或菱形被填充，且 **'Value'** 属性值设为1；若未被选择，指示符被清除，**'Value'** 属性值设为0。无线按钮键 **'style'** 的属性值是 **'radiobutton'** 。

无线按钮典型地用在一组互斥的选项中选择一项。为了确保互斥性，各无线按钮

uicontrol的回调字符串必须不选组中其它项，将它们各项的 **'Value'** 设为0。然而，这只是一个约定，如果需要，无线按钮可与检查框交换使用。

下面的例子（**mmctl2.m**）建立了两个互斥选项的无线按钮，它将坐标轴 **'Box'** 属性开或关闭。

```
>> Hc_boxon = uicontrol(gcf, 'Style', 'radio', ...
                'Position', [20 45 100 20], ...
                'String', 'Set box on
                'Value', 0, ...', ...
                'CallBack', [...
                    'set(Hc_boxon', "Value", 1'...
                    'set(Hc_boxoff', "Value", 0'...
                    'set(gca, "Box", "on")']);

>> Hc_boxoff = uicontrol(gcf, 'Style', 'radio', ...
                'Position', [20 20 100 20], ...
                'String', 'Set box off
                'Value', 1, ...', ...
                'CallBack
                'CallBack', [...
                    'set(Hc_boxon', "Value", 0'...
                    'set(Hc_boxoff', "Value", 1'...
                    'set(gca, "Box", "off")']);
```

检查框

检查框，又称*切换按钮*，它由具有标志并在标志的左边的一个小方框所组成。激活时，**uicontrol**在检查和清除状态之间切换。在检查状态时，根据平台的不同，方框被填充，或在框内含**x**， **'Value'** 属性值设为1。若为清除状态，则方框变空， **'Value'** 属性值设为0。

检查框典型地用于表明选项的状态或属性。通常检查框是独立的对象，如果需要，检查框可与无线按钮交换使用。

下面的例子（**mmctl3.m**）建立了一个检查框**uicontrol**， 设置坐标轴 **'Box'** 属性，当此检查框被激活时，测试 **'Value'** 属性以确定检查框是否以往被检查或清除过，并适当设置 **'Box'** 属性。

```
>> Hc_box = uicontrol(gcf, 'Style', 'check', ...
                'Position', [100 50 100 20], ...
                'String', 'Axis Box', ...
                'CallBack', [...
                    'if get(Hc_box, "Value")==1, '...
                        'set(gca, "Box", "on"), '...
                    'else, '...
                        '(gca, "Box", "off", '...
                    'end']);
```

静态文本框

静态文本框是仅仅显示一个文本字符串的**uicontrol**，该字符串是由 **'string'** 属性所确定的。静态文本框的 **'Style'** 属性值是 **'text'** 。静态文本框典型地用于显示标志、用户信息及当前值。

静态文本框之所以称之为 ' 静态 ' ，是因为用户不能动态地修改所显示的文本。文本只能通过改变
'String' 属性来更改。

在X Window系统中，静态文本框可只含有一行文字；如文本框太短，不能容纳文本串，则只显示部分文字。然而在Macintosh和MS-Window平台，长于文本框的文本串将字串起来，即在可能的地方，用词间分割的虚线显示多行。

下面的例子（**mmct14.m**）建立了含有MATLAB版本号的文本框。

```
>>  Hc_ver  =  uimenu(gcf，  'Style'，  'text'，...
                    'Position'，[10   10   150   20]，...
                    'String'，['MATLAB Version'，version])；
```

与其它的文本对象，如：坐标标题和坐标标志不同，函数的编著者对用在uicontrol文本串的字体不明显地控制。用在uicontrol文本串的字体和命令窗口的字体一致，可由用户设定。

X window系统的用户在激活MATLAB时，可以给出命令行的参量，如

```
matlab   -fn 9x14bold
```

该命令在命令窗和**uicontrol**文本串中使用**9x14bold**字体。Mocintosh对命令窗和**uicontrol**文本串使用相同的字体，但字体可以由用户在命令窗的**Options**菜单中用**Text Style**项来设定。PC用户具有选项，可从命令窗口的**Options**菜单的**Command Window Font**项来设置命令窗口字体，并从同一菜单中的**Uicontrol Font**项，为**uincontrol**文本串设置不同的字体。我们希望未来的MATLAB版本会在**uimenu**和**uicontrol**文本串中增加控制字体的属性。

可编辑文本框

编辑文本框，象静态文本框一样，在屏幕上显示字符。但与静态文本框不同，可编辑文本框允许用户动态地编辑或重新安排文本串，就象使用文本编辑器或文字处理器一样。在 **'String'** 属性中有该信息。可编辑文本框**uicontrol**的 **'Style'** 属性值是 **'edit'** 。可编辑文本框典型地用在让用户输入文本串或特定值。

可编辑文本框可包含一行或多行文本。单行可编辑文本框只接受一行输入，而多行可编辑文本框可接受行以上的输入。单行可编辑文本框的输入以**Return**键结尾。在X window和MS-Window系统中，多行文本输入以**Control-Return**键结尾，而在Macintosh中用**Command-Return**键。

下面的例子（**mmct15.m**)建立了静态文本标志和一个单行可编辑文本框。用户可以在文本框中输入颜色映象名，而后回调字符串把它放到在图中。

```
>>Hc_label=uicontrol(gcf, 'Style', 'text', ...
                    'Position'[10  10  70  20], ...
                    'String', 'Colormap: ');

>>Hc_map =uicontrol(gcf, 'Style', 'edit ', ...
                    'Position'[80  10  60  20], ...
                    'String', 'hsv', ...
                    'callback', 'coloormap(eval(get(Hc_map, "String")))');
```

通过把 ' Max ' 属性及 ' Min ' 属性设置成数值，诸如Max-Min>1，建立多行可编辑文本框。Max属性不指定最大的行数。多行可编辑文本框可具有无限多行。

一个多行可编辑文本框表示如下：

```
>>Hc_multi=uicontrol(gcf, 'Style', 'edit', ...
                      'Position', [20 50 75 75], ...
                      'String', 'Line 1 |Line 2|Line 3' ...
                      'Max', 2);
```

多行字符串被指定为单个引号的字符串，用垂直条字符 '|' 指明在何处分行。

滑标

滑标，或称滚动条，包括三个独立的部分，分别是滚动槽、或长方条区域，代表有效对象值范围；滚动槽内的指示器，代表滑标当前值；以及在槽的两端的箭头。滑标**uicontrol**的**'Style'** 属性值是**'slider'**。

滑标典型地用于从几个值域范围中选定一个。滑标值有三种方式设定。方法一：鼠标指针指向指示器，移动指示器。拖动鼠标时，要按住鼠标按钮，当指示器位于期望位置后松开鼠标。方法二：当指针处于槽中但在指示器的一侧时，单击鼠标按钮，指示器按该侧方向移动距离约等于整个值域范围的10% ；方法三：在滑标不论哪端单击鼠标箭头；指示器沿着箭头的方向移动大约为滑标范围的1% 。滑标通常与所用文本**uicontrol**对象一起显示标志、当前滑标值及值域范围。

下面的例子（**mmct16.m**）实现了一个滑标，可以用于设置视点方位角。用了三个文本框分别指示滑标的最大值，最小值和当前值。

```
>> vw = get(gca, 'View');

>> Hc_az = uicontrol(gcf, 'Style', 'slider ', ...
                    'Position'[10  5  140  20], ...
                    'Min', -90, 'Max', 90, 'Value', vw(1), ...
                    'CallBack', [...
                          'set(Hc_cur,"String",num22str(get(Hc_az,"Value"))),
'...
                          'set(gca, "View", [get(Hc_az, "Value") vw(2)])']);
```

```
>> Hc_min = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[10  25  40  20]，...
                    'String'，num2str(get(Hc_az， 'Min')))；

num2str(get(Hc_az， 'Min')))；

>> Hc_max = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[110  25  40  20]，...
                    'String'，num2str(get(Hc_az， 'Max))；

>> Hc_cur = uicontrol(gcf， 'Style'， 'text'， ...
                    'Position'，[60  25  40  20]，...
                    'String' num2str(get(Hc_az， 'Value')))；
```

滑标的 **'Position'** 属性包含熟悉的向量[**left bottom width height**]，其单位由 **'Units'** 属性设定。滑标的方向取决于宽与高之比。如果**width > height**就画水平方向的滑标，如果**width < height**就画垂直方向的滑标。仅在X-Window系统平台中，如果滑标的一个方向的大小比另一个方向小4倍，就不显示。其它操作平台上的滑标均有箭头。

弹出式菜单

弹出式菜单典型地用于向用户提出互斥的一系列选项清单，让用户可以选择。弹出式菜单，不同于前面论述过的下拉式菜单，不受菜单条的限制。弹出式菜单可位于图形窗口内的任何位置。弹出式菜单的 **'Style'** 属性值是 **'popupmenu'** 。

当关闭时，弹出式菜单以矩形或按钮的形式出现，按钮上含有当前选择的标志，在标志右侧有一个向下的箭头或凸起的小方块来表明**uicontrol**对象是一个弹出式菜单。当指针处在弹出式**uicontrol**之上并按下鼠标时，出现其它选项。移动指针到不同的选项，松开鼠标就关闭弹出式菜单，显示新的选项。MS-Windows 和某些X Window系统平台允许用户单击弹出式菜单，打开它，而后单击另一个选项来进行选择。

当选择一个弹出项时， **'Value'** 属性值设置成选择向量所选元素的下标。选项的标志指定为一个字符串，用垂直条 '|' 分隔，与指定多行文本的方法一样。下面的例子（**mmct17.m**）建立了图形颜色的弹出式菜单。回调函数把图形的 **'Color'** 属性值设定为所选值。每种与颜色相关的RGB值存储在弹出控制框的
'UserDate' 属性中。所有句柄图形对象的 **'UserData'** 属性仅仅为单独矩阵提供孤立的存储。

```
>> Hc_fcolor = uicontrol(gcf， 'Style'， 'popumenu'，...
        'Position'，[20 20 80 20]，...
        'String'， 'Black|Red|Yellow|Green|Cyan|Blue|Magenta|White'，...
        'Value'，1，...
       UserData'，[[0 0 0]；...
                   [1 0 0]；...
                   [1 1 0]；...
                   [0 1 0]；...
```

```
                                    [0 1 1]；...
                                    [0 0 1]；...
                                    [1 0 1]；...
                                    [1 1 1]；...
            CallBack'，[...
                'UD=get(Hc_fcolor，  '  'UserData'  ')；  '，...
                'set(gcf，  '  'Color'  '，UD(get(Hc_fcolor，  '  'Value'  '),：))
    ']);
```

弹出式菜单的 **'Position'** 属性含有熟悉的向量[**left bottom width height**]，其中宽度与高度决定了弹出对象的大小。在X Window和Macintosh系统中，就是被关闭的弹出式菜单的大小。打开时，菜单展开适合显示屏幕大小所有的选项。在MS-Windows系统中，高度值基本上被忽略，这些平台建立高度足够的弹出式菜单，显示一行文本而不管**height**的值。

框架

框架**uicontrol**对象仅是带色彩的矩形区域。框架提供了视觉的分隔性。在这点上，框架与**uimenu**的 **'Sepatator'** 属性相似。框架典型地用于组成无线按钮或其它**uicontrol**对象。在其它对象放入框架之前，框架应事先定义。否则，框架可能覆盖控制框使它们不可见。下面的例子（**mmct18.m**）建立了一个框架，把两个按钮和一个标志放入其中。

```
>>  Hc_frame  =  uicontrol(gcf,'Style'，'frame'，'Position'，[250 200 95 65])；

>>  Hc_pb1  =  uicontrol(gcf，  'Style'，  'pudhbutton'，...
                'Position'，[255 205 40 40]，  'String'，  'OK')；

>>  Hc_pb2  =  uicontrol(gcf，  'Style'，  'pushbutton'，...
                'Position'，[300 205 40 40]，  'String'，  'NOT')；


>>  Hc_1b1  =  uicontrol(gcf，  'Style'，  'text'，...
                'Position'，[255 250 85 10]，  'Str'，  'Push Me')；
```

控制框属性

如句柄图形对象建立函数一样，**uicontrol**属性可在对象建立时定义，或如上所示，用**set**命令来改
变。所有可设定的属性，包括字符串文本、回调串、甚至控制框函数类型都可以用**set**来改变。本章后面有若干例子。

表21.2列出了MATLAB 4.2版本中**uicontrol**对象的属性及其值。带有*的属性为非文件式的，使用时需加小心。由{}括起来的属性值是缺省值。

表21.2

Uicontrol 对象属性

BackgroundColor	**uicontrol**背景色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的背景色是浅灰色。
Callback	MATLAB回调串，当**uicontrol**激活时，回调串传给函数**eval**；初始值为空矩阵。
ForegroundColor	**uicontrol**前景（文本）色。3元素的RGB向量或MATLAB一个预先定义的颜色名称。缺省的文本色是黑色。
HorizontalAlignment	标志串的水平排列
left：	相对于**uicontrol**文本左对齐
{**center**}：	相对于**uicontrol**文本居中
right：	相对于**uicontrol**文本右对齐
Max	属性 **'Value'** 的最大许可值。最大值取决于**uicontrol**的 'Type' 当**uicontrol**处于**on**状态时，无线按钮及检查框将**Value**设定为**Max**；该值定义了弹出式菜单最小下标值或滑标的最大值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为1
Min	属性 **'Value'** 的最小许可值。最小值取决于**uicontrol**的 **'Type' uicontrol**处于**off**状态时。无线按钮及检查框将**Value**设定为**Min**；该值定义了弹出式菜单最小下标值或滑标的最小值。当**Max-Min>1**时，可编辑文本框是多行文本。缺省值为0
Position	位置向量**[left bottom width height]**。其中，**[left height]**表示相对于图形对象左下角的**uicontrol**的左下角位置。**[widht height]**表示**uicontrol**的尺寸大小，其单位由属性**Units**确定。
Enable*	控制框使能状态
{**on**}：	**uicontrol**使能。激活**uicontrol**，将**Callback**字符串传给
off：	**eval**
	unicntrol不使能，标志串模糊不清。激活**unicontrol**不起作用
String	文本字符串，在按钮键，无线按钮，检查框和弹出式菜单上指定**uicontrol**的标志。对于可编辑文本框，该属性设置成由用户输入的字符串。对弹出式菜单或可编辑文本框中多个选项或，每一项用垂直条(|)分隔，整个字符串用引号括起来。框架和滑标，不用引号

Style		定义**uicontrol**对象的类型
	{**pushbutton**}:	按钮键：选择时执行一个动作。
	radiobutton:	无线按钮键：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	checkbox:	检查框：单独使用时，在两个状态之间切换；成组使用时，让用户选择一个选项
	edit:	可编辑框：显示一个字符串并可让用户改变
	text:	静态文本框：显示一个字符串
	slider:	滑标：让用户在值域范围内选择一个值。
	frame:	框架：显示包围一个或几个**uicontrol**的框架，使其形
	popumenu:	成一个逻辑群。
		弹出式菜单 ：含有许多互斥的选择的弹出式菜单
Units		位置属性值的单位
	inches:	英寸
	centimeters:	厘米
	normalized:	归一化的坐标值，图形的左下角映射为[0　0]而右上
	points:	角的映射为为[1　1]
	{**pixels**}:	打印设置点，等于1/72 英寸
		屏幕的象素。计算机屏幕分辨率的最小单位。
Value		**uicontrol**的当前值。无线按钮和检查框在 **'on'** 状态时，**value**设为**Max**，当是 **'off'** 状态时，**value**设为**Min**。由滑标将滑标的**value**设置为数值（**Min≤Value≤Max**），弹出式菜单把**value**值设置所选择选项的下标（**1≤Value≤Max**）。文本对象和按钮不设置该属性。
ButtonDownFcn		当**uicontrol**被选择时，MATLAB回调串传给函数**eval**。
		初始值为空矩阵
Children		**Uicontrol**对象一般无子对象，通常返回空矩阵
Clipping		限幅模式
	{**on**}:	对**uicontrol**对象无作用效果
	off:	对**uicontrol**对象无作用效果
DestroyFcn		只对Macintosh 4.2 版本。没有文件说明
Interrruptible		指定**ButtonDownFcn**和**CallBack**串是否可中断
	{**on**}:	回调不能由其它回调中断
	off:	回调串可被中断
Parent		包含**uicontrol**对象的图形句柄
*Select		值为**[on|off]**
*Tag		文本串
Type		只读对象辩识串，通常为**uicontrol**
UserData		用户指定的数据。可以是矩阵，字符串等等
Visible		**uicontrol**对象的可视性
	{**on**}:	**uicontrol**对象在屏幕上可见
	off:	**uicontrol**对象不可见，但仍然存在

控制框布置的考虑

uicontrol的属性 **'Position'** 和 **'Units'** 用于分配图形窗口中对象的位置。**uicontrol**的缺省 'Position' 向量为**[20 20 60 20]**，以象素表示。该值是缺省 **'Units'** 值。这是一个**60×20** 象素**uicontrol**的象素矩形，**uicontrol**左下角位于父图形左下角的靠右边20个象素点，靠上边20 个象素。缺省的图形尺寸大约为**560×420**个象素，位于显示屏的中上部。利用这些信息，**uicontrol**的布置就变成了一个2维几何布局问题。

要加若干限制。比如，MS-Window忽略位置向量高度值，仅有足够高度以显示一行文本。在所有其它情况下，必须保证控制框足够大以容纳控制框标志字符串。因为显示控制框 **'String'** 属性值所用的字体与用在命令窗口的字体一致，所以，用户对于标志每个控制框的字体属性无法控制，不同平台用不同字体，整体上可由用户改变。

通常确定控制框的大小和位置是一个尝试的过程。即使结果很满意，图形在另一个平台上的外观也许会很不一样，还需调整。通常希望把控制框制作得比所需要的更大一些，以便在所有平台上其标志都可读。

正因为图形有缺省尺寸，不能保证图形都具有缺省大小。若在现有的图形窗口内加上**uicontrol**和**uimenu**，则图形尺寸会比缺省值大或小。另外，用户可以在任何时候，对任何图形重调尺寸，除非把图形的 **'Resize'** 属性值置 **'off'** ，才可避免这样做。

当把**uicontrol**加到尺寸可能重新调整的图形时，有两点需要考虑：属性 **'Units'** 和由固定字体标志字符串所加的约束。若**uicontrol**的位置是以绝对单位指定如：**象素、英寸、厘米或点**，则窗口尺寸的重调不会影响**uicontrol**的大小和位置。**uicontrol**相对于图形左下角的位置不变；若图形变小，则某些**uicontrol**可能不可见。

若**uicontrol**的位置以归一化单位指定，则当图形尺寸调整时，**uicontrol**相互之间及与图形本身之间的关系保持不变。但是，使用归一化的单位有一个缺点，即如果图形变小，**uicontrol**对象大小也要调整，则**uicontrol**的标志可能看不见，因为字体大小是固定的。任何超出调整后的**uicontrol**的部分就被删去。希望以后MATLAB版本会给程序员对**uicontrol**和**uimenu** 的标记字符串的字体属性有更多的控制框。

M 文件举例

下面例子说明了本章所讨论的某些控制框的用法。精通MATLAB工具箱中的函数**mmclock**在屏幕上生成一个数字钟，它用任选参数设定钟的位置。它使用了框架、文本、无线按钮、检查框和按钮键**uicontrol**。

为了在PC上运行这个例子，在命令窗的**Option**菜单中选择**Enable Background Process**菜单项。这样将引起MATLAB进入无限循环和死锁。为了在Macintosh上得到较好的结果，关掉**Option**菜单的**Background Operation**检查标记。X Windows系统版本不需任何调整。

首先，建立函数的语句和帮助文本。

```
function T=mmclock(X，Y)
%MMCLOCK Place a digital clock on the screen.
% MMCLOCK places a digital clock at the upper-right corner
% of the display screen.
% MMCLOCK(X，Y) places a digital clock at position X pixels
% to the right and Y pixels above the bottom of the screen.
% T=MMCLOCK returns the current date and time as a string
% when  'Close'  is pressed.
```

设定初始值，分析输入参量。

```
fsize=[200 150]；  sec=1；  mil=0；
mstr=[ ' Jan ' ；  ' Feb ' ；  ' Mar ' ；  ' Apr ' ；  ' May ' ；  ' Jun '
        ' Jul ' ；  ' Aug ' ；  ' Sep ' ；  ' Oct ' ；  ' Nov ' ；  ' Dec ' ]；
scr=get(0，  ' ScreenSize ' )；

if nargin= =0
    figpos=[scr(3)-fsize(1)-20 scr(4)-fsize(2)-5 fsize(1：2)]；
elseif nargin= =2
    figpos={X Y fsize(1：2)]；
else
     error( ' Invalid Arguments ' )；
end
```

建立图形，在图中设置**uicontrol**为某些缺省值。

```
Hf_clock=figure( ' Position ' ，  figpos ' ，  ...
                ' Color ' ，[.7   .7   .7]，...
                ' NumberTitle ' ，  ' off ' ，  ...
                ' Name ' ，  ' Digital Clock ' )；
set(Hf_clock，  ' DefaultUicontrolUnits ' ，  ' normalized ' ，  ...
               ' DefaultUicontrolBackgroundColor ' ，  get(Hf_clock，  ' Color ' ))；
```

对秒建立**Close** 按钮键、无线按钮；对24小时建立检查框。

```
Hc_close=uicontrol( ' Style ' ，  ' push ' ，  ...
                ' Position ' ，[.65   .05   .30   .30]，...
                ' BackgroundColor ' ，[.8   .8   .9]，...
                ' String ' ，  ' Close ' ，  ...
                ' CallBack ' ，  ' close(gcf) ' )；

Hc_sec=uicontrol( ' Style ' ，  ' radiobutton ' ，  ...
                ' Position ' ，[.05   .05   .50   .13]，...
                ' Value ' ，  sec，...
                ' String ' ，  ' Seconds ' )；

Hc_mil=uicontrol( ' Style ' ，  ' checkbox ' ，  ...
                ' Position ' ，[.05   .22   .50   .13]，...
                ' Value ' ，  mil，...
                ' String ' ，  ' 24-Hour ' )；
```

对日期和时间显示创建框架和文本串。

```
Hc_dframe=uicontrol('Style', 'frame', 'Position', [.04 .71 .92 .24]);
Hc_date      =uicontrol('Style', 'text', 'Position', [.05 .72 .90 .22]);
Hc_tframe=uicontrol('Style', 'frame', 'Position', [.04 .41 .92 .24]);
Hc_time=uicontrol('Style', 'text', 'Position', [.05 .42 .90 .22]);
```

循环，直到按钮键回调函数关闭图形，每秒更新显示一次。

```
while find(get(0, 'Children') = = Hf_clock % Looop while clock exists
        sec = get(Hc_sec, 'Value');
        mil = get(Hc_mil, 'Value');
        now=fix(clock);
        datestr=sprintf('%s %2d %4d', mstr(now(2),:, now(3), now(1));
        if mil
                suffx=' ';
        else
                if now(4)>12
                        suffix='pm';
                        now(4)=rem(now(4), 12);
                else
                        suffix='am';
                end
        end
        timestr=[num2str9now(4)) ':' sprintf('%02d', now(5))];
        if sec
             timestr=[timestr ':  sprintf('%02d, now(6))];
        end
timestr=[timestr suffix];
        set(Hc_date.'String', datestr);
        set(Hc_time, 'String', timestr);
        pause(1)
end
```

最后，如果有需，返回日期和时间字符串。

```
if nargout
        T=[datestr '-' timestr];
end
```

虽然这个例子说明了如何用**uicontrol**建立一个完整的个GUI函数，但并不实用。因为该函数保持循环直至**Close**按钮按下，非等到时钟图形关闭，命令窗口才能使用。

注意，上例只含有一个回调函数字符串，即在按钮键回调串中的 **'close'** 语句。其它按钮值是在While循环中得到，而不是按钮本身用回调函数激发一个动作。更复杂的**M文件**需要复杂的回调。

21.5 编程和回调考虑

用户句柄图形界面的基础部分已经阐述过了，现在是应用它的时候了。正如所见，在命令行通过输入**uimenu**和**uicontrol**来建立效率不高。脚本或函数M文件使用更为简便。假定想实现一个M文件，首先确定是否要编写脚本文件或函数文件。

脚本与函数

脚本文件似乎是当然的选择。在脚本中，所有的命令都在命令工作窗口执行，因此随时可以使用所有的MATLAB函数和对象句柄。将信息传给回调函数无任何困难。然而这里有几种权衡。首先，当所有的变量都是可利用时，工作空间内充斥了变量名和变量值，即使它们已经不再有用。其次，如果用户使用**clear**命令，重要的对象句柄就可能丢失。另一个缺点是：用脚本文件定义回调字符串可能变得十分复杂。例如，以下有一个滑标的文件片段，节选自精通MATLAB工具箱中脚本文件**mmsetclr**

```
Hc_rsli=uicontrol(Hf_fig， 'Style'， 'slider'， ...
    'Positon'， [.10 .55 .35 .05]， ...
    'min'， 0， 'max'， 1， 'Value'， initrgb(1)， ...
    'CallBack'， [...
        'set(Hc_nfr，"BackgroundColor"， ' ， ...
          '[get(Hc_rsli，"Val")，get(Hc_gsli，"Val")，get(Hc_bsli，"Val")])， ' ， ...
        'set(Hc_ncur，"String"， ' ， ...
        'sprintf("[%.2f %.2f %.2f]"，get(Hc_nfr，"BakgroundColor")))， ' ， ...
        'hv=rgb2hsv(get(Hc_nfr，"BakgroundColor"))； ' ， ...
        'set(Hc_hsli，"Val"，hv(1))， ' ， ...
        'set(Hc_hcur，"String"，sprintf("%.2f"，hv(1)))， ' ， ...
        'set(Hc_ssli，"Val"，hv(2))， ' ， ...
        'set(Hc_scur，"String"，sprintf("%.2f"，hv(2)))， ' ， ...
        'set(Hc_vsli，"Val"，hv(3))， ' ， ...
        'set(Hc_vcur，"String"，sprintf("%.2f"，hv(3)))， ' ， ...
        'set(Hc_rcur，"String"，sprintf("%.2f"，get(Hc_rsli，"Val")))' ]);
```

另一个问题是，脚本文件比函数文件运行得慢，脚本在第一次运行时要编译。最后一点，脚本文件没有函数灵活。函数可接受输入参量并返回值。因此，函数可用作其它函数的参变量。

函数不会使命令窗工作空间拥挤；当反复调用时运行快速；接受输入参量并返回值；回调字符串书写不复杂。因此，在许多场合，函数M文件是最佳的选择。

考虑前面脚本文件中滑标定义的例子。以下是等价的文件片段，取自精通MATLAB工具箱中函数文件**mmsetc**。

```
Hc_rsli=uicontrol(Hf_fig， 'Style'， 'slider'， ...
        'Position'， [.10 .55 .35 .05]， ...
        'Min'， 0， 'Max'， 1， 'Value'， initrgb(1)， ...
```

```
'CallBack'， 'mmsetc(0，"rgb2new")')；
```

注意回调字符串以不同的参量调用**mmsetc**。这是一个在回调中使用递归函数调用的例子。然而函数本身有它自己的问题。主要的困难源于要将回调字符串传给**eval**并在**命令窗口工作空间中运行**，而其余的程序码在函数工作空间内执行。这里用前面章节所讨论的变量和函数的大量规则。在函数内定义的变量在命令窗口工作空间不存在，因此不在回调串中使用。同时在命令窗工作空间的变量在函数本身内部也不存在。

有若干既能解决该问题又能利用函数优点的方法。全局变量、**'UserData'** 属性、仅用于回调的特殊函数M文件和递归函数调用均是创建GUI函数十分有用的工具。

独立的回调函数

建立GUI函数的一个有效方法是编写独立的回调函数，专门执行一个或多个回调。函数使用的对象句柄和其它变量可以作为参量传递，必要时回调函数可返回值。

考虑先前的一个例子，建立一个方位角的滑标，以脚本文件来实现。

```
% setview.m script file

vw=get(gca， 'View')；

Hc_az=uicontrol(gcf， 'Style'， 'slider'，...
                'Position'，[10  5  140  20]，...
                'Min'，-90， 'Max'，90， 'Value'，vw(1)，...
                'Callback'，[...
                    'set(Hc_cur，'String'，num2str(get(Hc_az，'Value')))，'...
                    'set(gca， 'View'，[get(Hc_az， 'Value') vw(2)])'])；
Hc_min=uicontrol(gcf， 'style'， 'text'，...
                 'Position'，[10  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Min')))；

Hc_max=uicontrol(gcf， 'Style'， 'text'，...
                 'Position'，[110  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Max')))；

Hc_cur=uicontrol(gcf， 'Style'， 'text'，...
                 'Position'，[60  25  40  20]，...
                 'String'，num2str(get(Hc_az， 'Value')))；
```

下面是同样的例子。作为一个函数，采用 **'Tag'** 属性来辨别控制框，并使用独立的M文件来执行回调。

```
funtion setview( )

vw=get(gca， 'View')；
```

```
Hc_az=uicontrol(gcf，'Style'，'Slider'，...
                'Position'，[10  5  140  20]，...
                'Min'，-90，'Max'，90，'Value'，vw(1)，...
                'Tag'，'Azslider'，...
                'Callback'，'svcback')；

Hc_min=uicontrol(gcf，'style'，'text，...
                'Position'，[10  25  40  20]，...
                'String'，num2str(get(Hc_az，'Min')))；

Hc_max=uicontrol(gcf，'Style'，'text'，...
                'Position'，[110  25  40  20]，...
                'String'，num2str(get(Hc_az，'Max')))；

Hc_cur=uicontrol(gcf，'Style'，'text'，...
                'Position'，[60  25  40  20]，...
                'Tag'，'Azcur'，...
                'String'，num2str(get(Hc_az，'Value')))；
```

回调函数本身如下：

```
function svcback( )

vw = get(gca，'View')；

Hc_az = findobj(gcf，'Tag'，'AZslider')；
Hc_cur = findobj(gcf，'Tag'，'AZcur')；

str = num2str(get(Hc_az，'Value'))；
newview =[get(Hc_az，'Value') vw(2)]；
set(Hc_cur，'String'，str)
set(gca，'View'，newview)
```

上面的例子并不节省很多代码，但却得到了用函数而不用脚本文件的优点：回调函数可以利用临时变量，而不使命令窗口工作空间拥挤；不需要**eval**所需的引号和字符串；在回调函数中命令的句法变得十分简单。使用独立回调函数技术，越复杂的回调(函数)越简单。

独立回调函数的缺点是：需要很大数目的M文件以实现一个含有若干控制框和菜单项的GUI函数，所有这些M文件必须在MATLAB路径中可得，且每一个文件又必须要有一个不同的文件名。在对文件名大小有限制且对大小写不敏感的平台上，如MS-windows，文件冲突的机会就增加了。而且回调函数只能被GUI函数调用而不能被用户调用。

递归函数调用

利用单独的M文件并递归地调用该文件，既可以避免多个M文件的复杂性，又可以利用函数的优点。使用开关 **switches**或**if elseif**语句，可将回调函数装入调用函数内。通常这样一种函数调用的结构为

```
function guifunc(switch)。
```

其中**switch**确定执行哪一个函数开关的参量，它可以是字符串 **'startup'， 'close'， 'sectolor'** 等等，也可以是代码或数字。如**switch**是字符串，则可如下面所示的M文件片段那样将开关编程。

```
if nargin < 1， switch = 'startup'； end；
if ~isstr(switch)， error('Invalid argument')， end；
if strcmp(switch， 'startup')，
    <statement to create controls or menus>
    <statements to implement the GUI function>
elseif strcmp(switch， 'setcolor')，
    <statements to perform the Callback associated with setcolor>
elseif strcmp(switch， 'close')，
    <statements to perform the Callback associated with close>
end
```

如果是代码或字符串，开关也可以相同方式编程。

```
if nargin < 1， switch = 0； end；
if isstr(switch)， error('Invalid argument')， end；
if switch = = 0，
    <statements to create controls or menus>
    <statements to implement the GUI function>
elseif switch = =1，
    <statements to perform the Callback associated with setcolor>
elseif switch ==2，
    <statements to perform the Callback associated with close>
end
```

下面的例子说明了方位角滑标如何可作为单独的函数M文件来实现：

```
function setview(switch)

if nargin < 1， switch = 'startup'； end；
if ~isstr(switch)， error('Invalid argument.')； end；

vw = get(gca， 'view')； % This information is needed in both sections

if strcmp(switch， 'startup') % Define the controls and tag them
```

```
        Hc_az = uicontrol(gcf， 'Style'， 'slider'， ...
                    'Position'，[10 5 140 20]，...
                    'Min'，-90， 'Max'，90， 'Value' vw(1)，...
                    'Tag'， 'AZslider'，...
                    'Callback'， 'setview('set')');

        Hc_min=uicontrol(gcf， 'Style'， 'text'，...
                     'Position'，[10 25 40 20]，...
                     'String'，num2str(get(Hc_az， 'Min')));
        Hc_max = uicontrol(gcf， 'Style'， 'text'，...
                     'Position'，[110 25 40 20]，...
                     'String'，num2str(get(Hc_az， 'Max')));

        Hc_cur =uicontrol(gcf， 'Style'， 'text'，...
                    'Position'，[60 25 40 20]，...
                    'Tag'， 'AZcur'，...
                    'string'，num2str(get(Hc_az， 'Value')));

    elseif strcmp(switch， 'set') % Execute the Callback

        Hc_az=findobj(gcf， 'Tag'， 'AZslider');
        Hc_cur-findobj(gcf， 'Tag'， 'AZcur');

        str = num2str(get(Hc_az， 'Value'));
        newview - [get(Hc_az， 'Value') vw(2)];

        set(Hc_cur， 'String'， str)
        set(gca， 'View'，newview)
    end
```

上述的两个例子均设置了**'tag'**属性，利用该属性和函数**findobj**寻找回调函数所需对象的句柄。另外两种方法将在下章描述。

全局变量

全局变量可用在函数中，使某些变量对GUI函数的所有部分都可用，全局变量是在函数的公共区说明，因此整个函数以及所有对函数的递归调用都可以利用全局变量，下面的例子说明如何利用全局变量将方位角滑标编程。

```
    function setview(switch)

    global HC_AZ HC_CUR % Create global variables
```

```
if nargin < 1， switch = ' startup ' ； end；
if ~isstr(switch， error( ' Invalid argument. ' )； end；

vw = get (gca， ' View ' )； % This information is needed in both sections
if strcmp(switch， ' startup ' ) % Define the controls

    Hc_AZ=uicontrol(gcf， ' style ' ， ' slider ' ，...
                ' Position ' ，[10 5 140 20]，...
                ' Min ' ，-90， ' Max ' ，90， ' Value ' ，vw(1)，...
                ' Callback ' ， ' setview( ' set ' ) ' )；

    Hc_min=uicontrol(gcf， ' Style ' ， ' text ' ，...
                ' Position ' ，[10 25 40 20]，...
                ' String ' ，num2str(get(Hc-AZ， ' Min ' ，)))；

    HcMax=uicontrol(gcf， ' Style ' ， ' text ' ，...
                ' Position ' ，[110 25 40 20]，...
                ' String ' ，num2str(get(Hc_AZ， ' Max ' )))；

    Hc_cur=uicontrol(gcf， ' style ' ， ' text ' ，...
                ' Position ' ，[60 25 40 20]，...
                ' String ' ，num2str(get(HC_AZ， ' Value ' )))；

elseif strcmp(switch， ' set ' ) % Execute the Callback

      str=num2str(get(HC_AZ， ' Value ' ))；
      newview= [get(HC_AZ， ' Value ' ) vw(2)]；

      set(HC_CUR， ' String ' ，str)
      set(gca， ' View ' ，newview)

end
```

全局变量遵循MATLAB的规定，变量名要大写。不需要**'tag'**属性，且不使用它，另外因为对象句柄存在的，不需要用函数**findobj**去获取，故回调代码比较简单，全局变量通常使一个函数更有效。

不过有一点要注意，尽管一个变量在函数内说明为**全局**的，变量并不能自动地在命令窗口工作空间中利用，也不能在回调字符串内使用。但是，如果用户发命令：**>> clear global**，则所有全局变量则都被破坏，包括在函数内定义的那些变量。

当单独的一个图形或有限个变量要被所有的回调(函数)利用时，全局变量使用和递归性函数调用都是有效的技术。对于包含多个图形的更复杂的函数，或用独立对象回调函数实现的情况，**'UserData'**属性更合适。另外，只要可获得对象句柄，对象**'UserData'**的属性值在命令窗口工作空间中是存在的。

用户数据属性

同属性 **'Tag'** 一样， **'UserData'** 属性可在函数之间或递归函数的不同部分之间传递信息。如果需要多个变量，这些变量可以在一个容易辨识的对象的属性 **'UsetData'** 中传递。**如前面所述， 对与句柄图形对象在一起的单个数据矩阵'UserData' 提供了存储**。下面的程序利用了当前图形的 **'UserData'** 属性来实现方位角滑标。

```
function setview(switch)

if nargin < 1， switch = 'startup'； end；

vw = get(gca， 'View')； % This information is needed in both sections

if strcmp(switch， 'startup') % Define the controls

    Hc_az = uicontrol(gcf， 'Style'， 'slider'， ...
                'Position'，[10 5 140 20]，...
                'Min'，-90， 'Max'，90， 'Value'，vw(1)，...
                'Callback'， 'setview("set")')；

    Hc_min = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[10 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Min')))；

   Hc_max = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[110 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Max')))；

   Hc_cur = uicontrol(gcf， 'Style'， 'text'，...
                'Position'，[60 25 40 20]，...
                'String'，num2str(get(Hc_az， 'Value')))；

   set(gcf， 'UserData'，[Hc_az Hc_cur])； % Store the object handles

elseif strcmp(switch， 'set') % Execute the Callback

   Hc_all = get(gcf， 'UserData')； % retrieve the object handles
   Hc_az = Hc_all(1)；
   Hc_cur = Hc_all(2)；

   str = num2str(get(Hc_az， 'Value'))；
   newview = [get(Hc_az， 'Value') vw(2)]；
   set(Hc_cur， 'String'， str)
   set(gca， 'View'， newview)
```

```
end
```

句柄存储于 **'startup'** 末端，图形 属性**'UserData'** 中，在回调被执行前对此进行检索。如果有许多回调，如下面的程序片断所示， **'UserData'** 只需检索一次。

```
if strcmp(switch， 'startup') % Define the controls and tag them

    % <The 'startup' code is here>

    set(gcf， 'UserData' ，[Hc_az Hc_cur])； % Store the object handles

else      % This must be a Callback

     Hc_all=get(gcf， 'UserData')； % Retrieve the object handles
     Hc_az=Hc_all(1)；
     Hc_cur=Hc_all(2)；

     if strcmp(switch， 'set')

          % <The 'set' Callback code is here>

     elseif strcmp (switch， 'close')

          %<The 'close' Callback code is here>
     % <Any other Callback code uses additional elseif clauses>
     end
end
```

调试GUI M文件

回调字符串在命令窗口工作空间中计算并执行的，这个情况对编写和调试GUI函数和脚本文件有某种隐含意义。回调字符串可以很复杂，尤其是在脚本文件中，这为句法错提供了许多机会，记录单引号、逗号、括号是令人头痛的事。如果出现了句法错误，MATLAB给出提示；只要对象的 **'Callback'** 属性值是一个真正的文本串，MATLAB就认可了。只有当对象被激活并将回调字符串传给**eval**时，才检查回调字符串内部的句法错误。

这样让用户定义回调字符串，它涉及还未曾定义过的对象句柄和变量，这使编写相互参照的程序变得更容易，但是每个回调函数必须分别测试，保证回调字符串是合法的MATLAB命令，并且回调字符中涉及的所有变量可在命令窗口工作空间中是可利用的。

将回调象函数M文件一样编程或象GUI函数本身内的开关一样编程，就可以不运行整个GUI函数而对各个回调进行改变或测试。

因为回调字符串是在命令窗工作空间中而不在函数本身内计算，在函数与各回调之间传递数据就变得十分复杂。例如，函数test包含如下程序：

```
function test()
```

```
tpos1=[20   20   50   20];

tpos2=[20   80   50   20];

Hc_text=uicontrol('Style','text','String','Hello','Position',tposl);

Hc-push=uicontrol('Style', 'push', 'String', 'Move Text', ...
  'Position',[15   50   100   25], ...
  'Callback', 'set(Hc-text,"Position",tpos2)');
```

所有语句都是有效的MATLAB命令，且回调字符串也对有效的MATLAB语句估值。文本对象和按钮出现在图形上，但当激活按钮键时，MATLAB就出示错误。

```
>>  test
>>
??? Undefined functiion or variable Hc_text.

??? Error while evaluating Callback string.
```

如果**test**是个脚本文件，就不会出现这样问题，因为所有变量可在命令窗口工作空间中使用，因为**test**是个函数，**Hc_text**和**tpos2**在命令窗口工作空间中均未定义，回调字符串执行失败。

一种解决方法是使用各个字符串元素来建立回调，该字符串元素由数值而非变量建立，例如，改变回调字符串如下：

```
'Callback', ['set9', ...
        sprintf('&.15g', Hc_text), ...
        ',  ' 'Position' ',  ', ...
        sprintf('[%.15g   %.15g   %.15g   %.15g]', tpos2), ...
        ')']);
```

建立了包括**Hc_text**对象句柄值的一个字符串，该值变换成具有15位精度的字符串，而**tpos2**变量转换成矩阵表示的字符串。在函数内计算**sprinf**语句，然后将所得的字符串用在回调中。在命令窗工作空间执行的实际命令如下所示

```
eval('set(87.000244140625,  ' 'Position' ', [20   80   50   100])')
```

将一个对象句柄转换为字符串，必须保持全精度。上例中的变换，使用了小数点后15位的数字的精度。在MATLAB中句柄对象转换应使用这样的精度。

要记住，变量随后的变换不会改变回调字符串。在前面的例子中，在控制框定义之后改变**tpos2**的值，就无效果。例如，在函数结尾处加命令

tpos2=[20　200　50　20]

就无效果，因为在**tpos2**重新定义之前，通过计算**tpos2**，回调字符串已经建立。

21.6　指针和鼠标按钮事件

GUI函数利用鼠标箭头的位置和鼠标按钮的状态来控制MATLAB行动。本节讨论指针、对象位置和鼠标按钮动作之间的交互，以及MATLAB如何响应变化或事件，诸如：揿下按钮、松开按钮或箭头移动等。

回调属性，选择区域和堆积顺序

所有句柄图形对象具有一个至今还未阐述过的 **'ButtonDownFcn'** 属性，本节就进行讨论。**uimenu**和**uicontrol**均有 **'CallBack'** 属性，这个属性是菜单和控制框的应用核心。另外，图形有 **'KeyPressFcn'** 和 **'ResizeFcn'** 属性以及 **'WindowButtonUpFcn'** 和 **'WindowButtonMotionFcn'** 属性。与这些属性相关的值是回调字符串，即MATLAB字符串，当属性激活时，它传给**eval**。指针的位置确定事件涉及到哪些回调以及当事件发生时它们被激励的顺序。

前一章讨论过堆积顺序和与讨论相关的对象选择区域。根据图形中3个区域，MATLAB决定哪一个回调将被激励。如果指针是在句柄图形对象内，如同对象的 **'Position'** 属性所确定的那样，则指针就是在对象**上**。如果指针不在一个对象上，而在对象的选择区域内，则指针**靠近**对象。如果指针在图形内但既不靠近也不在另一对象之上，则可以认为指针关掉其它对象。如果若干对象及其选择区域相重叠，重叠顺序就决定了选择顺序。

句柄图形的线、曲面、补片、文本和坐标轴对象的选择区域已在上一章讨论过了。**uimenu**对象没有外部的选择区域，指针要么在**uimenu上**，要么不在其上。**uicontrol**对象越过图基位置，在各方向延伸大约5个象素有一个选择区域，指针可以**靠近**或在控制框**上**。记住这里讨论的堆积顺序和选择区域是针对4.2c版本的，在以后的MATLAB有某种程度的变化。

按钮点击

按钮点击可以定义为当鼠标指针在同一对象上时，揿下并随后松开鼠标按钮。如果鼠标指针在**uicontrol**或**uimenu**项上，按钮点击触发对象 **'Callback'** 属性字符串的执行，按钮揿下使控制框作好准备，并常常在视觉上改变**uicontrol**或**uimenu**，松开按钮触发回调。当鼠标指针不在**uicontrol**或**uimeun**上，**揿下按钮**和**松开按钮**事件，将在后面讨论。

揿下按钮

当鼠标指针位于一个图形窗口内，揿下鼠标按钮，根据指针位置和对句柄图形对象靠近程度将会发生不同的动作。如果一个对象被选择，它就变成了当前的对象，并上升到堆积顺序的最高端。如果没有对象被选择，图形本身就是当前对象，图形的 **'CurrentPoint'** 和 **'SelectionType'** 属性都被更新，然后激发适当的回调。表格21.3列出了指针位置与按钮揿下所激发的回调：

表21.3

指针位置	所激发的属性
在**uimenu** 或**uicontro**项上面	准备释放事件
在句柄图形**上**，或**接近**控制框	图形的**WindowButtonDownFcn**属性，然后是对象的**ButtonDownFcn**属性
在图形内，但不**在**或**接近**任何对象	图形的**WindowButtonDownFcn**属性，然后是图形的**ButtonDownFcn**属性

注意：按钮揿下事件总是在所选对象的 **'ButtonDownFcn'** 回调之前，引起图形的 **'WindowButtonDownFcn'** 回调，除非指针是在**uicontrol**或**uimenu**对象上。如果指针靠近控制框，则在图形的 **'WindouButtonDownFcn'** 回调完毕后，引起控制框的 **'ButtonDownFcn'** 回调而不是 **'Callback'** 属性回调。

按钮松开

当松开鼠标按钮时，图形的 **'CurrentPoint'** 属性就更新。如果没有定义 **'WindowButtonUpFcn'** 回调，则鼠标按钮松开时，属性 **'CurrentPoint'** 不更新。

指针的移动

当指针在图形内移动时，图形的 **'CurrentPoint'** 属性被更新，引起图形的 **'WindowButtonMotionFcn'** 回调；如果没有定义图形的 **'WindowButtonMotionFcn'** 回调，则指针移动时，属性 **'CurrentPoint'** 不更新。

回调的复合能产生许多有趣的效应。试一下包含在MATLAB于**demo**目录中的函数**sigdemo1**，**sigdemo2**，就可解其中的一些效果。另外将要讨论的一个例子是 精通MATLAB工具箱的函数**mmdraw**。

21.7 中断回调的规则

一旦回调开始执行，通常都在下一个回调事件处理之前运行完毕。将 **'Interruptible'** 属性设置为 **'yes'** 可改变这种缺省行为，从而当正在执行的回调遇到**drawnow**，**figure**，**getframe**或**pause**命令时，允许处理的回调事件悬挂起来。事件队列执行计算操作或设置对象属性的命令一经发出，MATLAB便进行处理；而涉及图形窗口输入或输出的命令则生存事件。事件包括产生回调的指针移动和鼠标按钮动作，以及重新绘制图形的命令。

回调处理

回调在达到**drawnow**，**getframe**，**pause**或**figure**命令之前一直执行。注意**gcf**和**gca**引起**figure**命令，而精通MATLAB工具箱中的函数**mmgcf**和**mmgca**不引发**figure**命令。不含有这些特殊命令的回调不会被中断，一旦达到这些特殊命令之一，停止执行回调，将其悬挂起来；并检查事件队列中每一个悬挂的事件。如果产生悬挂回调的对象的**Interruptible**属性设为 **'yes'**，则在被悬挂的回调恢复之前按序处理所有悬挂。如果**Interruptible**属性设为 **'no'**，即缺省值，则只处理悬挂的重画事件，放弃回调事件。

防止中断

即使正在执行回调是不能被中断的，当回调达到**drawnow**，**figure**，**gefframe**或**pause**命令时，仍然处理悬挂的重画事件。通过避免在回调中使用所有这些特殊命令，消除此类事情。如果回调中需要这些特殊命令，但又不要任何悬挂事件，甚至是刷新事件，来中断回调，则可以使用如下所讨论的特殊形式的**drawnow**。

Drawnow

Drawnow命令迫使MATLAB更新屏幕，只要MATLAB回到命令提示，或执行**drawnow**，**figure**，**getframe**或**pause**命令，屏幕就更新。**drawnow**的特殊形式**draunow(' discard ')**使事件队列中所有事件的放弃。在回调中将**drawnow(' discard ')**包含在一个特殊命令之前，就含有清除事件队列的效果，防止刷新事件，以及回调事件中断回调。

21.8 M文件举例

精通MATLAB工具箱中一些函数阐明了上面所讨论的若干技术。第一个例子**mmview3d**应用全局变量和递归函数调用，把方位角和仰角滑标加到图形中。函数中有大量的对象，但函数很直观。因为**mmview3d**文件相当得长，故分段表示。第一段定义了函数标号，帮助文本和全局变量。

```
function mmview3d(cmd)
%    MMVIE3D GUI controllled Azimuth and Elevation adustment.
%    for adjusting azimuth and elevation using the mouse.
%
%    The    ' Revert '    pushbutton reverts to the original view.

%    The    ' cmd '    argument executes the callbacks.

%    Copyright (c) 1996 by Prentice-Hall，    Inc.

global Hc_acur Hc_esli Hc_ecur CVIEW
```

第二段处理初始用户的调用，建立必要的**uicontrol**对象并把回调定义为递归函数调用。

```
if    nargin==0

%--------------------------------------------------------------------
%    Assign a handle to the current figure window.
%    Get the current view for slider initial values.
%    Use normalized uicontrol units rather than the default    ' pixels ' .
%--------------------------------------------------------------------
```

```
Hf_fig=gcf;
CVIEW=get(gca， 'View');
if abs(CVIEW(1))>180， CVIEW(1)=CVIEW(1)-(360*sign(CVIEW(1)))；end
set(Hf_fig， 'DefaultUicontrolUnits' ， 'normalized');

%--------------------------------------------------------------------
%  Define azimuth and elevation sliders.
%  The position is in normalized units  (0-1).
%  Maximum， minimum， and initial values are set.
%--------------------------------------------------------------------

Hc_asli=uicontrol(Hf_fig， 'style' ， 'slider' ， ...
 'position' ，[.09  .02  .3  .05]，...
 'min' ，-180， 'max' ，180， 'value' ， CVIEW(1)，...
 'callback' ， 'mmview3d(991)');

Hc_esli=uicontrol(Hf_fig， 'style' ， 'slider' ，...
 'position;，[.92  .5  .04  .42]，...
 'min' ，-90， 'max' ，90， 'val' ，CVIEW(2)，...
 'callback' ， 'mmview3d(992)');

%--------------------------------------------------------------------
%  Place the text boxes showing the minimum and maximum values at the
%  ends of each slider， These are text displays， and cannot be edited.
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.02  .02  .07  .05]，...
 'string' ，num2str(get(Hc_asli， 'min')));

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.39  .02  .07  .05]，...
 'string' ，num2str(get(Hc_esli， 'min')));

uicontrol(Hf_fig， 'style' ， 'text' ，...
 'pos' ，[.915  .92  .05  .05]，...
 'string' ，num2str(get(Hc_esli， 'max')));

%--------------------------------------------------------------------
% Place labels for each slider
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style' ， 'text' ，...
```

```
 'pos'，[9.095  .08  .15  .05]，...
 'string'， 'Azimuth')；

uicontrol(Hf_fig， 'style'， 'text'，...
 'pos'，[.885  .39  .11  .05]，...
 'string'， 'Elevation')；

%--------------------------------------------------------------------
%  Define the current value text displays for each slider，
%--------------------------------------------------------------------
%  These are editable text display the current value
%  of the slider and at the same time allow the user to enter
%  a value using the keyboard.
%
%  Note that the text is centered on XWindow Systems，but is
%  left-justified on MS-Windows and Macintosh machines.
%
%  The initial value is found from the value of the sliders.
%  When text is entered into the text area and the return key is
%  pressed， the callback string is evalueated.
%--------------------------------------------------------------------

Hc_acur=uicontrol(Hf_fig， 'style'， 'edit'，...
 'pos'，[.25  .08  .13  .053]，...
 'string'，num2str(get(Hc_asli， 'val'))，...
 'callback'， 'mmview3d(993)')；

Hc_ecur=uicontrol(Hf_fig， 'style'， 'edit'，...
 'pos'，[.885  .333  .11  .053]，...
 'string'，num2str(get(Hc_esli， 'val'))，...
 'callback'， 'mmview3d(994)')；

%--------------------------------------------------------------------
%  Place a  'Done'  button is the lower right corner.
%  When clicked， all of the uicontrols eill be erased.
%--------------------------------------------------------------------

uicontrol(Hf_fig， 'style'， 'push'，...
 'pos'，[.88  .02  .10  .08]，...
 'backgrouncolor'，[.7  .7  .8]，...
 'string'， 'Done'，...
 'callback'， 'delete(finobj(gcf， '  'Type'  '， '  'uicontrol'  '))')；

%--------------------------------------------------------------------
```

```
%   Place a  'Revert'  button next to the  'Done'  button.
%   When clicked，  the view reverts to the original view.
%---------------------------------------------------------------------

Hc_ecur=uicontrol(Hf_fig，  'style'，  'edit'，...
 'pos'，[.77   .02   .10   .08]，...
 'backgroundcolor'，[.7   .7   .8]，...
 'string'，  'Revert'，...
 'callback'，  'mmview3d(995)')；
```

现在已建立了控制框并定义了回调。

```
else

%---------------------------------------------------------------------
%   The callbacks for the azimuth and elevation sliders；
%---------------------------------------------------------------------
%   1)   get the value of the slider and display it in the text windows
%   2)   set the  'View'  property to the current values of the azimuth
%         and elevation sliders.
%---------------------------------------------------------------------

if   cmd==991
     set(Hc_acur，  'string'，  num2str(get(Hc_asli，  'val')))；
     set(gca，  'View'，  [get(Hc_asli，  'val')，get(Hc_esli，  'val')])；

elseif   cmd==992
     set(Hc_ecur，  'string'，  num2str(get(Hc_esli，  'val')))；
     set(gca，  'View'，  [get(Hc_asli，  'val')，get(Hc_esli，  'val')])；

%---------------------------------------------------------------------
%   The  'slider current value'  text display callbacks；
%---------------------------------------------------------------------
%   When text is entered into the text area and the return key is
%   pressed，  the entered value is compared to the limits.
%
%   If the limits have been exceeded，  the display is reset to the
%   value of the slider and an error message is displayed.
%
%   If the value is within the limits，  the slider is set to the
%   new value，  and the view is updated.
%---------------------------------------------------------------------

elseif cmd==993
```

```
        if str2num(get(Hc_acur， 'string')) < get(Hc_asli， 'min)...
          | str2num(get(Hc_acur， 'string')) >get(Hc_asli， 'max')
              set(Hc_acur， 'string'， num2str(get(Hc_asli， 'val')))；
              disp('ERROR - Value out of range')；
        else
            set(Hc_asli， 'val'， str2num(get(Hc_acur， 'string')))；
            set(gca， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；
        end
    elseif cmd==994
        if str2num(get(Hc_ecur， 'string')) < get(Hc_esli， 'min)...
          | str2num(get(Hc_ecur， 'string')) >get(Hc_esli， 'max')
              set(Hc_ecur， 'string'， num2str(get(Hc_esli， 'val')))；
              disp('ERROR - Value out of range')；
        else
            set(Hc_esli， 'val'， str2num(get(Hc_ecur， 'string')))；
            set(gca， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；
        end

    %-----------------------------------------------------------------
    %   Revert to the original view.
    %-----------------------------------------------------------------

    elseif cmd==995
        set(Hc_asli， 'val'， CVIEW(1))；
        set(Hc_esli， 'val'， CVIEW(2))；
        set(Hc_acur， 'sting'， num2str(get(Hc_asli， 'val')))；
        set(Hc_ecur， 'sting'， num2str(get(Hc_esli， 'val')))；
        set(Hc_acur， 'View'，[get(Hc_asli， 'val')，get(Hc_esli， 'val')])；

    %-----------------------------------------------------------------
    %   Must be bad arguments.
    %-----------------------------------------------------------------

      else
        disp('mmview3d： Illegal argument.')
      end
    end
```

第二个例子**mmcxy**，在图形左下角建立一个小文本框，当指针处在图形中时，显示图形内鼠标指针的坐标**[x，y]**。点击鼠标清除坐标的显示。

虽然**mmcxy**是一个短小的函数，它仍然利用许多有效的GUI函数的元素，包括递归函数调用、全局变量和 **'UserData'** 属性。此例还说明图形的 **'WindowButtonDownFcn'** 和 **'WindowButtonMotionFcn'** 属性起动回调的用法。

```
function out=mmcxy(arg)
%   MMCXY Show x-y Coordinates of the mouse in the
%   lower left hand corner of the current 2-D figure window.
%   When the mouse is clicked，  the coordinates are erased.
%   XY=MMCXY returns XY=[x，y] coordinates where mouse was clicked.
%   XY=MMCXY returns XY=[] if a key press was used.

%   Copyright (c) by Prentice-Hall，  Inc.

global MMCXY_OUT

if  -nargin
      Hf=mmgcf;
      if  isempty(Hf)，  error('No Figure Available.')，  end
      Ha=findobj(Hf，  'Type'，  'axes');
      if  isempty(Ha)，  error('No Axes in Current Figure，  ')，end.

      Hu=uicontrol(Hf，  'Style'，  'text'，  ...
                          'units'，  'pixels'，  ...
                          'Position'，[1   1   140   15]，...
                          'HorizontalAlignment'，  'left');
      set(Hf，  'Pointer'，  'crossh'，  ...
            'WindowButtonMotionFcn'，  'mmcxy('  'move'  ')，...
            'WindowButtonDownFc'，  'mmcxy('  'end'  ')'，  ...
            'Userdata'，Hu)
      figure(Hf)   %bring figure forward
      if nargout   %must return x-y data
            key=waitforbuttonpress；    %   pause until mouse is pressed
            if  key
                  out=[]；              %return empty if aborted
                  mmcxy('end')     %clean thins up
            else
                  out=MMCXY_OUT；   %now that move is complete return point
            end
            return
      end

elseif strcmp(arg，  'move')          %   mouse is moving in figure window
      cp=get(gca，  'CurrentPoint')；     %   get current mouse position
      MMCXY_OUT=cp(1，1：2)；
      xystr=sprintf('[%.3g]'，MMCXY_OUT)；
      Hu=get(gcf，  'Userdata')；
      set(Hu，  'String'xystr)   %   put x-y coordinaates in text box
```

```
elseif strcmp(arg，  'end')  %  mouse click occurred，  clean things up
    Hu=get(gcf，  'Userdata')；
    set(Hu，  'visible'，  'off')  %  make sure text box disappears
    delete(Hu)
    set(gcf，  'Pointer'，  'arrow'，...
            'WindowButtonMotionFcn'，  ''，...
            'WindowButtonDownFcn'，  ''，...
            'Userdata'，[])
end
```

第一次被调用时，**mmcxy**建立文本**uicontrol**，改变指针形状，设定 **'WindowButtonDownFcn'** 和 **'WindowButtonMotionFcn'** 的回调，然后等待按键或先撤按钮。若有键撤下，就调用清除（cleanup）程序，清除文本框，恢复鼠标指针，清除图形回调及 **'UserData'** 属性。若点击鼠标按钮，**'WindowButtonDownFcn'** 回调就处理清除任务。在等待时，图形中鼠标指针的移动会触发 **'WindowButtonMotionFcn'** 回调，更新uicontrol中文本串。

精通MATLAB工具箱中的另一个函数是**mmtext**，它利用 **'WindowButtonDownFcn'** ，**'WindowButtonUpFcn'** 和 **'WindowButtonMotionFcn'** 回调的另一个短小函数以安置和拖曳文本。

```
function mmtext(arg)
%  MMTEXT place and drag text with mouse
%  MMTEXT waits for a mouse click on a text object
%  in the current figure then allows it to be dragged
%  while the mouse button remains down.
%  MMTEXT('whatever') places the string  'whatever'  on
%  the current axes and allows it to be dragged.
%
%  MMTEXT becomes inactive after the move is complete or
%  no text object is selected.

%  Copyright (c) 1996 by Prentice-Hall，  Inc.

if -nargin，  arg=0；end

if isstr(arg) % user entered text to be placed
    Ht=text('units'，  'normalized'，...
        'Position'，[.05  .05]，...
        'String'，arg，...
        'HorizontalAlignment'，  'left'，...
        'VerticalAlignment'，  'baseline')；
    mmtext(0)  %  call mmtext again to drag it

elseif arg==0 %  initial call，  select text for dragging
    Hf=mmgcf；
```

```
if isempty(Hf),    error('No Figure Available.'),  end
set(Hf, 'BackingStore', 'off', ...
    'WindowButtonDownFcn', 'mmtext(1)')
figure(Hf)  %  bring figure forward

elseif arg==1  &  strcmp(get(gco, 'Type'), 'text')  % text object selected
set(gco, 'Units', 'data', ...
            'HorizontalAlignment', 'left', ...
```

mmdraw是精通MATLAB工具箱中另一个有用GUI函数，它与**mmtext**十分相似，但更为复杂，此函数允许用户用鼠标在当前的坐标轴上画线并设置线的属性。

```
function mmdraw(arg1，arg2，arg3，arg4，arg5，arg6，arg7)
%MMDRAW Draw a Line and Set Properties Using Mouse.
%MMDRAW Draw a Line in the current axes using to the mose.
%Click at the starting point and drag to the end point
%In addition，properties can be given to the line.
%Properties are given in pairs，e.g.，MMDRW name vale...
%Properties：
%NAME              VALUE          (defult)
%color               [y m c r g b {w} k] or an rgb in quotes：   '[r g b]
%style               [-----{：}--.]
%mark                [0 + . * x]
%width               points for linewidth {0.5}
%size                 points for marker size {6}
%Examples：
%MMDRAW color r width 2 sets color to red and width to 2 points%MMDRAW mark
+ size 8   set marker type to +and size to 8 points
%MMDRAW color  '[1 5 0]'  set color to orange

%Copyright (c) 1996 by Prentice-Hall，Inc.

global MMDRAW_Hl MMDRAW_EVAL

if nargin==0
        arg1='color'；arg2='w'；arg3='style'；arg4='：  '；nargin=4；
end

if isstr(arg1) % initial call，set thing up
        if isempty(Hf)，error('No Figure Available.')，end
        Ha=findobj(Hf,  'Type,  'axes')；
        if isempty(Ha)，error('No Axes in Current Figure.')，end
        set(Hf,  'Pointer',  'Crossh',  ...%set up callback for line star
                'BackingStore',  'off',  ...
```

```
                    'WindowButtonDownFcn', 'mmdraw(1)')
        figure(Hf)
        MMDRAW_EVAL='mmdrw(99);%set up string to set attributes
        for i=1:nargin
            argi=eval(sprintf('arg%.of', i));
            MMDARW_EVAL=[MMDARW_EVAL ',' 'arg' ' ' ' '];
        end
        MMDARW_EVAL=[MMDARW_EVAL ,')'];

elseif arg1==1 % callback is line start point
        fp=get(gca, 'CurrentPoint');        %start point
        set(gca, 'Userdata', fp(1, 1:2))    %store in axes userdata
        set(gcf, 'WindowButtonMotionFcn', 'mmdraw(2)', ...
                    '          'WindowButtonUpFcn', 'mmdraw(3)')

elseif arg1==2 % callback is mouse motion
        cp=get(gca, 'CurrentPoint'); cp=cp(1, 1:2);
        fp=get(gca, 'Userdata');
        H1=line('Xdata', [fp(1); cp(1)], 'Ydata', [fp(2); cp(2)], ...
                    'EraseMode', 'Xor', ...
                    'Color', 'w', 'LineStyle', ':', ...
                   "Clipping', 'off';
        if ~isempty(MMDRAW_HL) % delete prior line if it exsts
            delete(MMDRAW_HL)
        end
        MMDRAW_HL=H1;  % store current line handle

elseif arg1==3 % callback is line end point, finish up
        set(gcf, 'Point', 'arrow', ...
                    'BackingStore', 'on', ...
                    'WindowButtonDownFcn', '', ...
                     'WindowButtonMotionFcn', 'mmtext(2)', ...
                     'WindowButtonUpFcn', '')
        set(gca, 'Userdata', [])
        set(MMDRAW_HL, 'EraseMode', 'normal') % render line better
        eval(MMDRAW_EVAL)
        MMDRAW_EVAL=[];

elseif arg1==99 % process line properties
        for i=2:2:nargin-1
            name=eval(sprintf('arg%.of', i), []);  % get name argument
            vale=eval(sprintf('arg%.of', i+1), []);  % get value argument
            if strcmp(name, 'color')
                if value(1)=='[', value=eval(value); end
```

```
                set(MMDRAW_HL， 'color'， value)
            elseif strcmp(name， 'style')
                set(MMDRAW_HL， 'Lineatyle'， value)
            elseif strcmp(name， 'mark')
                set(MMDRAW_HL， 'Linestyle'， value)
            elseif strcmp(name， 'width')
                 value=abs(eval(value))；
                set(MMDRAW_HL， 'LineWidth'， value)
            elseif strcmp(name， 'size')
                 value=abs(eval(value))；
                set(MMDRAW_HL， 'MarkerSize' ''， value)
            end
        end
        MMDRAW_HL=[]；
    end
```

虽然这里说明太长，但精通MATLAB工具箱中的函数**mmsetc**和**mmsetf**都是使用递归、全局变量和 '**UserData**' 属性的GUI函数的直观例子。也许愿意看一下**mmsetclr M**文件，它是函数**mmsetc**文件的脚本M文件。比较这两个文件，可以了解到，为实现各种GUI M文件要做出一些折衷。

21.9 对话框和请求程序

MATLAB具有建立对话框和 ' 请求 ' 的几个有用工具。对话框是弹出显示的单独窗口，它显示信息字符串。对话框含有一个或多个按钮键以供用户输入。请求框是在弹出显示的单独窗口，利用鼠标或键盘获得输入，并返回信息给调用函数。

对话框

所有MATLAB的对话框都是基于函数**dialog**，它的帮助文本如下

```
>>help diolog
DIALOG displays a dialog box.

    FIG = DIALOG (pl ， vl....)displays a dialog box.
    valid param/value pairs include
    Style               error  |  warning  |  help  |  question
    Name                string
    Replace        on  |  off
    Resize         on  |  off
    BackgroundColor  ColorSpec
    ButtonString        'Button1String  |  Button2String  |  ...  '
    ButtonCalls         'ButtonCallback  |  Button2Callback  | ...  '
```

ForegroundColor ColorSpec
Position [x y width height]
[x y] - centers around screen point
TextString string
Units pixels | normal | cent | inches | points
UserData matrix

Note： Until dialog becomes built-in， set and get
are not valid for dialong objects.
At most three buttons are allowed.
The callbacks are ignored for"question"dialogs.
If ButtonStrings /ButtonCalls are unspecified then it
defaults to a single "ok" button which removes the figure.
There ' s still problems with making the question dialog modal.
The entire parameter name must be passed in.
(i.e. no automatic completion).
Nothing beeps yet.
See also ERRORDLG，HELPDLG，QUESTDLG

注意：对话窗本身不是句柄图形对象，而是由一系列句柄图形对象构成的M文件。对话窗口是图形，包括与框架、编辑和按钮**uicontrol**对象共存的坐标轴。在将来的版本中，**dialog**可能成为具有更多功能的内置函数。缺省的对话框是一个帮助对话框，它是由 '**Default help string**' 字符串和标有 'OK' 的按钮键组成的编辑文本框。作为例子，键入>> **dialog**。

预先定义的对话框是由函数**helpdlg**，**enordlg**，**warndlg**和**gucstdlg**建立。**helpdlg**和**warndlg**接受文本字符串和窗口标题字符串作为输入参量。**errordlg**接收 '**Replace**' 变量作为输入。除了**questdlg**，所有上述函数都产生类似的对话框，它有各自缺省的标题和文本字符串。标有 '**OK**' 的单个按钮，则关闭对话框。**helpdlg**帮助文本是：

>> help helpdlg

HELPDLG： Displays a help dialog box.
HANDLE=HELPDLG(HELPSTRING，DLGNAME) displays the
message HelpString in a dialog box with title DLGNAME.
if a help dialog with that name is aldeady on the screen，
it is brought to the front.Otherwise it is created.

See also：DIALOG

帮助信息

Helpdlg：显示一个帮助文本框

HANDLE = HELPDLG(HELPSTRING，DLGNAME)在对话框中显示标题为dlgname的帮助信息helpstring。如果名为dlgname的帮助对话框已在屏幕上显示，则引到屏幕正面，否则就建立该帮助对话框。

参阅：dialog

---

warndlg的帮助文本是：

>>help warndlg
WARNDLG Creates a warning dialog box.
HANDLE=WARNDLG(WARNSTR， DLGNAME) creates a warning
diolog box which displays WARNSTR in a window named DLGNAME.A pushbutton labeled OK must be pressed to make the warning box disappear.

See also：DIALOG

errordlg的帮助文本是

>>help errordlg

ERRORDLG Creates an error dialog box
HANDLE = ERRORDLG(ERRORSTR，DLGNAME，Replace) creates an error dialog box which diplays ERRORSTR in a window nnnamed DLGNAME.A pushbutton labeled OK must be pressed to make the error box disappear.If REPLACE='on' and an error dialog with Name DLGNAME already exits， it is simply brought to the front (no new dialog is created).

See also ：DIALOG

帮助信息
ERRORDLG：建立出错对话框
HANDLE=ERRORDLG(ERRORSTR，DLGNAME，REPLACE)建立显示出错信息errorstr、名为dlgname的出错对话框，要消除出错信息对话框，必须揿下标记为OK的按钮，如果replace='on'并且名为digname的出错对话框已经存在，则就引到屏幕正面，不再建立新的对话框。

参阅：dialog

提问的对话框稍有不同，前三种函数只显示一个按钮键并返回对话框图形对象的句柄。函数**questdlg**显示两个或三个按钮键，返回由用户所选的按钮的标志字符串。 **questdlg**的帮助文本如下：

>>help qeustdlg

QEUSTDLG Creates a question dialog box.
CLICK=QUESTDLG(Q，YES，O，CANCEL，DEFAULT) creates a question dialog box which displays Q.Up to three pushbuttons，with strings given by YES，NO，and CANCEL，will appear along with Q in the dialog.The dialog wil be destroyed returning the string CLICK depending on which button is clicked.DEFAULT is the default button number.

帮助信息
QUESTDLG：建立问题对话框
CLICK=QVESTDLG(Q，YES，NO，CANCEL，DEFAULT)建立显示信息Q的问题对话框。至多有三个按钮具有由YES，NO和CANCEL给定的字符串，按钮与Q一起，显示在对话框中。根据所击的按钮返回字符串CLICK对话框消失。DEFAULT是缺省的按钮数。

下面是函数的片断，说明了函数中提问对话框的应用：

```
question1=' Change color map to copper? ' ；
response1=questdlg(question1， ' Sure ' ， ' Nope ' ， ' Maybe ' ，2)；
if stromp(response1， ' Sure ' )
    colormap(copper)；
elseif stromp(response1， ' Maybe ' )
    wandlg( ' That response does not compute! ' )；
    response2=questdly([ ' Please make up your mind.｜ ' question1]， ' Yes ' ， ' No ' )；
    if stromp (response2， ' Yes ' )
        colormap(copper)；
    end
end
```

请求程序

请求程序通过对话框获取用户的输入。请求程序是内置式GUI函数，它使用平台原有的窗口系统建立外观熟悉的请求程序。

函数**uigecfile**和**uiputfile**是所有平台上都有的内置式函数，用于交互地获得文件名，从而调用函数用它读取文件中数据或将数据存于文件中。**uigetfile**的帮助文本是：

>>help uigetfile

UIGETFILE Interactively retrieve a filename by displaying a dialog box.

[FILENAME，PATHNAME]=UIGETFILE(' filterSpec ' ， ' dialogTitle ' ，X，Y)
displays a dialog box for the user to fill in，and returns the
filename and path strings.A successful return occurs only if
the file exists.If the user selects a file that does not exist，
an error message is displayed，and control returns to the dialog box.
The user may then enter another filename，or press the Cancel button.

All parameters are optional，but if one is used，all previous
parameters must also be used.

The felterSpec parameter determines the initial display of files in
the dialog box.For example ' *.m ' lists all the MATLAB M-files.

Parameter ' dialogTitle ' is a string containing the title of the dialog
box.

The X and Y parameters define the initial position of the dialog
box in units of pixels. Some systems may not support this option.

The output variable FILENAME is a string containing the mane of the
file selected in the dialog box.If the user presses the Cancel button
or if any error occurs，it is set to 0.

The output parameter PATHNAME is a string containing the path of
the file selected in the dialog box. If the user presses the Cancel
button or if any error occurs，it is set to 0.

See also UIPUTFILE.

帮助信息

UIGETFILE：通过显示对话框交互式地检索文件名

[FILENAME，ATHNAME]=UIGETFILE(' filterspec ' ， ' dialogtitle ' ，x，y)显示一个对话框，让用户输入，并返回路径和文件名字符串。仅当文件存在时，才成功地返回。如果用户选择了一个并不存在的文件，就显示出错信息，控制框返回到对话框。用户可以输入另一个文件名或掀下Cancel按钮。

所有输入参数都是可任选的，如果用其中之一个，也必须使用所有先前参数。
参数filteospec决定对话框中文件的初始显示。例如 ' *.m ' 列出的所有M文件。
参数 ' dialogtitle ' 是对话框标题字符串。
以象素为单位参数x，y定义对话框的初始位置，有些系统可能不支持这个选项。
输出变量filename是对话框内所选文件的名称字符串。如果用户掀了取消按钮或有错误发生，filename的值设置为0。
输出参数pathname是对话框内所选文件的路径名字符串。如果用户掀了取消按钮或

有错误发生，pathname的值设置为0。
参阅 uiputfile

下面的例子说明了在函数中如何利用**uigetfile**，交互式地检索ASCII码数据文件，并绘制正弦数据

```
% Ask the user for a file name.

[datafile datapath]=uigetfile( ' .dat ' ， ' Choose a data file ' )；

% If the user selected an existing file，read the data.
% (The extra quotes avoid problems with spaces in file or path names
% on the Macintosh platform.)
% Then determine the variable name from the file name，
% copy the data to a variable，and plot the data.
if datafile
        eval([ ' load( '  ' datapath datafile '  ' ) ' ] ' )；
        x=eval(strtok(datafile. ' . ' )；
        plot (x，sin(x))；
end
```

请求程序不接受并不存在的文件名。这种情况可能发生的原因是用户在请求程序中把文件名输入到文件框中。Macintosh版本无文本框的，用户必须选择一个存在的文件或揿下**Cancel**按钮以退出请求程序。

函数**uiputfile**与函数**uigetfie**十分相似：输入参量相似，而且两者均返回文件和路径字符串，**uiputfile**的帮助文件是

```
>> help uiputfile

UIPUTFILE Interactively retrieve a filename by displaying a dialog box.
[FILENAME，PATHNAME]=UIPUTFILE( ' inifFile ' ， ' dialogTitle ' )
displays a dialog box and returns the filename and path strings.

The initFile paramete determines the initial display of files
in the dialog box.Full file name specifications as well as wildcards
are allowed.For example， ' newfile.m '  initializes the display to
that particular file and lists all othe existing .m files.This may
be used to provide a default file mane. A wildcard specification such
as  ' *.m '  lists all the existing MATLAB M-files.

Parameter  ' dialogTitle '  is a string containing the title of the dialog
box.
```

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Cancel butto
or if any error occurs，it is set to 0.

The output variable FILENAME is a string containing the name of the
file selected in the dialog box. If the user presses the Concel button
or if any error occurs，it is set to 0.

[FILENAME，PATHNAME]=UIPUTFILE (' initFile ' ， ' dialogTitle ' ，X，Y)
places the dialog box at screen position [X，Y] in pixel units.
Not all systems support this option.

Example：

[newmatfile，newpath]=uiputfile (' *.mat ' . ' Save As ')；

See also UIGETFILE.

如果用户选择了已经存在的文件，则出现一个对话窗，询问用户是否要删除存在的文件。如果回答**no**，则返回原来的请求程序等待另一次尝试；如果回答**yes**，则关闭请求程序和对话框并把文件名返回，文件并未被请求程序删除。如果需要，调用函数必须删除或覆盖文件。

值得牢记的是，这些函数中不论哪一个都未真正地读或写任何文件，调用函数必须做这些工作。这些函数仅仅是将文件名和路径返回给调用函数。

在MS-Windows和Macintosh平台上，**uisetcolor**让用户交互式地选择颜色并选择性地将此颜色施加到一个对象上。如同MATLAB4.2C版本，X Window系统平台不支持**uisetcolor**。

uisetcolor的帮助文本是：

>>help uisetclor

UISETCOLOR Interactively set a ColorSpec by displaying a dialog box.
C=UISETCOLOR(ARG， ' dialogTitle ') displays dialog box for the
user to fill in，and applies the selected color to the input
graphics object.

The parmeters are optional and may be specified in any order.

ARG may be either a handle to a graphics object or an RGB triple.
If a handle is used，it must specify a graphics object that supports
color.If RGB is used，it must be a valid RGB triple (e.g.，[1 0 0]
for red). In both cases，the color specified is used to initialize
initializes the color to black.

If parameter ' dialogTitle ' is used， it is a string containing the
title of the dialog box.

The output value C is the selected GB triple. If the input parmeter is a handle，the graphics object ' s color is set to the RGB color selected.

If the user presses Cancel from the dialog box，or if any error occurs，the output value is set to input RGB triple，if provided；otherwise，it is set to 0.

Example：

C=uisetcolor(hText，　' Set Text Color ')

NOTE：This function is only available in MS-Windows and Macintosh versions of MATLAB.

帮助信息

UISETCOLOR ：显示对话框，交互式地设置ColorSpace。

C=UISETCOLOR(ARG，　' dialogtitle ')显示一个对话框，让用户输入，并将所选颜色用于输入的图形对象。

参数是可任选，并可以以任何次序指定。

ARG可以是一个图形对象的句柄或是RGB3元组。如果使用句柄，必须指定支持颜色的图形对象；如果使用RGB，必须是有效的RGB3元组(如：[1　0　0]是红色)。在这两种情况下，所指定的颜色用于对话框的初始化。如果没有指定初始的RGB，将对话框初始化为黑色。

如果使用参数 ' dialogTitle ' ，该参数是对话框名称的字符串。

输出值C是所选的RGB3元组。如果输入参数是句柄，则图形对象的颜色就设定为所选的颜色。

如果用户揿下对话框中的Cancel按钮或有错误发生，输出值就设定为输入的RGB3元组；如果没有输入RGB3元组，则输出值设置为0。

例如：c=uisetcoior(htext，　' settextcoior ')

注意：该函数仅在的MS-Windous和Macintosh版本中可用。

精通MATLAB工具箱具有前面所提及的函数**mmsetc**。该函数可在所有的平台上工作，功能与**uisetcolor**十分相似。**mmsetc**甚至可允许用户用鼠标来选择颜色并将所选的颜色加到所选择的对象上。以下就是**mmsetc**的帮助文本：

>> help mmsetc

MMSETC Obtain an RGB triple interactively form a color sample.

MMSETC displays a box for the user to select
a color nteractively and displays the result.

X=MMSETC returns the selected color in X.

MMSETC ([r g b]) uses the RGB triple as the initial
RGB value for modification.

MMSETC C -or-
MMSETC (' C ') where C is a color spec (y，m，c，r，g，b，w，k)，uses the specified color as the initial value.

MMSETC(H) where the input argument H is the handle of a valid graphics object that supports color，uses the color progerty of the object as the initial RGB value.

MSETC select -or-
MMSTEC(' select ') waits for the user to click on a valid graphics object that supports color，and uses the color property of the object as the initial RGB value.

If the initial RGB value was obtained from an object or object handle，the ' Done ' pushbutton will apply the resulting color property to the selected object.

If no initial color is specified，black will be used.

Examples：

```
mmsetc
mycolor=mmsetc
mmsetc([.25 .62 .54])
mmsetc(H)
mmsetc g
mmsetc red
mmsetc select
mycolor=mmsetc( ' select ' )
```

下面的例子是利用精通MATLAB工具箱中函数**mmap**和**mmsetc**，交互式地使用单色映象：

```
>>mesh(peaks)
```

>>coiormap(mmap(mmsetc))

在MS-Windows和Machitosh平台上，有最后的请求程序**uisetfont**，它让用户可交互式选择字型属性并将其加于对象上。如同MATLAB4.2c版本，X Window系统平台不支持**uisetfont**。**uisetfont**帮助文件是：

```
>>  help uisetfont

UISETFONT   Interactively set a font by displaying a dialog box.
    H=UISETFONT(HIN，  ' dialogTitle ' ) displays a dialog box for the
    user to fill in，and applies the selected font to the input
    graphics object.

    The parmeters are optional and may be specified in any order.

    If parameter HIN is used，it must specity a handle tp a text or
    axis graphics object. The font properties currently assigned to
    this object are used to initialize the font dialog box.

    If parameter  ' dialogTitle '  is used，it is a string containing the
    title of the dialog box.

    The output H is a handle to graphics object. If HIN is specified，
    H is identical to HIN.If HIN is not specified，a nwe text object
    is created with the selected font properties，  and its handle is
    returned.

    If the user presses Cancel from the dialog box，or if any error
    occurs，the output value is set to the input handle，if provided；
    otherwise ，it is set to 0.

    Example：
        uisetfont(hText，  ' Update Font ' )

    NOTE：This function is only available in MS-Windows and Macintosh
        versions of MATLAB.
```

精通MATLAB工具箱有一个称为函数**mmsetf，**功能类似于**uisetfont**。但可工作在所有平台上。**mmsetf**甚至可允许用户用鼠标来选择文本对象，然后将所选的字体属性加在所选的对象上。以下就是**mmsetf**的帮助文本：

```
>>help mmsetf

MMSETF Choose font characteristics interactively.
```

MMSETF displays a dialog box for the user to select
fornt characteristics.

X=MMSETF returns the handle of the text object or 0
if an error occurs of ' Cancel ' is pressed.

MMSETF(H) where the input argument H is the handle of
a valid text or axes object，uses the font characteristics
of the object as the initial values.

MMSETF select -or-
MMSETF (' select ')waits for the user to click on a valid
graphics object，and uses the font characteristics
of the object as the initial values.

If the initial values were obtained from an object or
object handle，the ' Done ' pushbutton will apply the
resulting text properties to the selected object.

If no initial object handle is specified，a zero is returned in X.

Examples：
mmsetf
mmsetf(H)
mmsetf select
Hx_obj=mmsetf(' select ')

然而**mmsetf**中用于字体的选择是受限制的。不同的平台类型，甚至同一类型的不同计算机都可能安装了不同的字体。MATLAB的函数**uisetfont**是置于内部的，并利用计算机平台的操作系统列出可用的字体。因为**mmsetf**是的GUI 函数，不能确定哪种字体是可用的，所以用了通常可获得的有限选择。字体的大小也因同样原因受到限制。

mmsetf请求程序中的样本文本字符串指明了每一个属性改变的作用。选择后，如果出现的文本字符串并不如所希望的那样，则说明所选择的字体属性(如字体名称、大小等等)在所用计算机上是不存在的。见到的即所得到的。

21.10 用户自制的GUI M文件

许多MATLAB用户充分利用的MATLAB所提供的GUI工具，编写了一些有趣GUI函数。这中间大部分可在MATLAB的 匿名**FTP** 地址**/pub/contrib/graphics**路径中得到。Internet资源一章将详细地讨论如何从这一资源库中获得文件的详细指令。

一些M文件和M文件集含有大量的句柄图形GUI对象，并说明了**uimenu**和**uicontrol**的使用。其中让人印象最深刻的是科斯·荣格(Keith Roger)编写的**matdraw**集合。该文件集合可在**/pub/contrib/graphics/matdraw2.0**目录下获得。该文件集将工具条和画图工具调色板加到

图形窗口，效果同集成到MATLAB的绘图程序一样。

另一个值得研究的是派瑞克·马查德(Patrick Marchard)编写的**guimaker**集合。这是交互式工具的集合，让用户应用GUI来建立GUI函数。可以用鼠标建立GUI对象，放置GUI对象和指定GUI对象的大小。版本1.0以freeware形式发布，即它可以无偿使用。2.0版本以及以后的版本以shareware形式发布的，即你可以无偿地使用30天，然后你需注册并付不多的费用。两个版本均可在的 匿名ftp地址**/pub/contrib/graphics**路径中得到。

这里只是现有GUI函数很多例子中的一小部分。核实一下。也许会对别人要用但已消失的GUI应用程序有想法。第23章有详细指导信息帮助用户访问现有的MATLAB资源，甚至帮助用户成为M文件的贡献者。

21.11 小结

图形用户界面设计不是为每一个人的。但如果需要，MATLAB使它有可能仅由M文件来建立令人印象深刻的GUI函数。需用**mex**文件、高级语言或数据库调用对有用的MATLAB函数来建立一个赏心悦目的界面。

对GUI对象的字体控制局限性，在下一版本MATLAB的发行中应有所强调。对话框也有可能在未来成为内置函数。所有这些改进工作将使编写GUI函数更加容易，使它们既与平台无关并且更加悦目。

本章所讨论的函数总结如下

表21.4

句柄图形GUI函数	
uimenu(handle， 'PropertyName'，value)	创建或改变图形的菜单
uicontrol(handle， 'PropertyName'，value)	创建或改变对象的性质
dialog('PropertyName'，value)	显示对话框
helpdlg('HelpString'， 'DlgName')	显示 ' 帮助 ' 对话框
warndlg('WarnString'， 'DlgName')	显示 ' 警告 ' 对话框
errordlg('ErrString'， 'DlgName'，Replace)	显示 ' 出错对话框
questdlg('Qstring'，S1，s2，s3，Default)	显示 ' 提问 ' 对话框
uigetfile(Filter，Dlgname，X，Y)	交互式地检索文件名
uiputfile(InitFile，Dlgname，X，Y)	交互式地检索文件名
uisetcolor(Handle，Dlgname)	交互式地选择颜色
uisetcolor([r g b]，Dlgname)	交互式地选择颜色
uisetfont(Handle，Dlgname)	交互式地选择字体属性

以下是本章所提到的精通MATLAB工具箱中的函数：

表21.5

精通MATLAB工具箱中的句柄图形GUI函数	
mmenu	**uimenu**函数示例
mmclock(X，Y)	**uicontrol**函数示例
mmsetclr	函数**mmsetc**的有限脚本型式
mmview3d	把方位角和仰角的滑标加到图形上

mmcxy	用鼠标显示的x-y坐标
mmtext(' String ')	用鼠标放置和拖曳文本
mmdraw(' Name ' ， Value)	用鼠标画线并设置属性
mmsetc(Handle)	用鼠标设置颜色属性
mmsetc(ColorSpec)	
mmsetc(' select ')	
mmsetf(Handle)	用鼠标设置字体属性
mmsetf(' select ')	

第22章　符号数学工具

MATLAB所具有的符号数学工具箱与其它所有工具不同，它适用于广泛的用途，而不是针对一些特殊专业或专业分支。另外，MATLAB符号数学工具箱与其它的工具箱区别还因为它使用字符串来进行符号分析，而不是基于数组的数值分析。为此，本章包含了该工具箱的教学辅导材料。

22.1　引言

符号数学工具箱是操作和解决符号表达式的符号数学工具箱(函数)集合，有复合、简化、微分、积分以及求解代数方程和微分方程的工具。另外还有一些用于线性代数的工具，求解逆、行列式、正则型式的精确结果，找出符号矩阵的特征值而无由数值计算引入的误差。工具箱还支持可变精度运算，即支持符号计算并能以指定的精度返回结果。

符号数学工具箱中的工具是建立在功能强大的称作Maple软件的基础上。它最初是由加拿大的滑铁卢（Waterloo）大学开发的。当要求MATLAB进行符号运算时，它就请求Maple去计算并将结果返回到MATLAB命令窗口。因此，在MATLAB中的符号运算是MATLAB处理数字的自然扩展。

22.2　符号表达式

符号表达式是代表数字、函数、算子和变量的MATLAB字符串，或字符串数组。不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则，这些符号表达式几乎完全象基本的MATLAB命令。表22.1列有几则符号表达式例子以及MATLAB等效表达式。

表22.1

符号表达式	MATLAB表达式
$\frac{1}{2x^n}$	'1/(2*x^n)'

$\frac{1}{y=\sqrt{2x}}$	y= ' 1/sqrt(2*x) '
$\cos(x^2)-\sin(2x)$	' cos(x^2)-sin(2*x) '
$M=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym(' [a，b；c，d] ')
$\int_a^b \frac{x^3}{\sqrt{1-x}}dx$	f=int(' x^3/sqrt(1-x) ' ， ' a ' ， ' b ')

MATLAB符号函数可让用户用多种方法来操作这些表达式，比如，

```
>> diff( ' cos(x) ' )   %   differentiate cos(x) with respect to x
ans=
  -sin(x)

>> M=sym( ' [a，b；c，d] ' )   %   create a symbolic matrix M
M=
  [a，b]
  [c，d]

>> determ(M)   %   find the determinant of the symbolic matrix M
ans=
  a*d-b*c
```

请注意，上面的第一个例子的符号表达式是用单引号以隐含方式定义的。它告诉MATLAB **' cos(x) '** 是一个字符串并说明**diff(' cosx ')**是一个符号表达式而不是数字表达式；然而在第二个例子中，用函数**sym**显式地告诉MATLAB **M=sym(' [a，b；c，d] ')**是一符号表达式。在MATLAB可以自己确定变量类型的场合下，通常不要求显式函数**sym**。

正如在第八章所阐述，MATLAB中函数**function argument**形式是与**function(' argument ')**等价的。其中，**function**是一个函数，**argument**是一字符串。例如，MATLAB可以构造**diff cos(x)**和**diff(' cos(x) ')**两者都意味**diff (sym ' cos(x) ')**。但第一种形式显然更便于输入。然而，很多时候**sym**是必要的。在上述的第二个例子中，

```
>> M=[a，b；c，d]   %   M is a numeric matrix using value of a through d
???Uundefine function or variable a.

>> M= ' [a，b；c，d] '    %   M is a character string，  but not a symbolic matrix
M=
  [a，b；c，d]

>> M=sym( ' [a，b；c，d] ' )   %   M is a symbolic matrix
M=
```

[a，b]
[c，d]

M以三种方式定义： 数字型(如果**a**、**b**、**c**、**d**已预先确定)、字符串型或符号矩阵型。许多符号函数非常巧妙能够自动将字符转变为符号表达式。但在某些情况下，尤其是建立符号数组时，必须用函数**sym**，特别地将字符串变为符号表达式。隐含形式，例如**diff cos(x)**，对于那些不需要参考先前结果的简单任务，最有用。但是最简单形式(无引号)要求一个参量，它是一个单字符的字符串、不包含插入的空格。

```
>> diff x^2+3*x+5    %    the argument is equivalent to    ' x^2+3*x+5 '
ans=
  2*x+3

>> diff x^2 + 3*x + 5    %    spaces break the argument into separate strings
???Error using==>diff
Too manyinput arguments
```

无变量的符号表达式称作符号常量。符号常量常常与整数很难区别，例如

```
>> f=symop( ' (3*4-2)/5+1 ' )    %    reduce a symbolic constant to its simplest form
f =
  3

>> isstr(f) %    is f a string? (1=yes，  0=no)
ans=
  1
```

在这个例子中，f代表符号常数 '3' ；而不是数字3。正如第六章所阐述的，MATLAB是以字符ASCII码形式来存储字符串的。所以，如果对字符串进行数字运算，则在运算中，采用各字符串的ASCII码值。因为数字51是字符 '3' 的ASCII表示，所以f加1在数值上不能得到期望的结果

```
>> f+1
ans=
  52
```

符号变量

当字符表达式中含有多于一个的变量时，只有一个变量是独立变量。如果不告诉MATLAB哪一个变量是独立变量，MATLAB将基于以下规则选择一个：

在符号表达式中缺省的独立变量是唯一的，除去i和j的小写字母，不是单词的一部分。如果没有这种字母，就选择x作为独立变量。如字符不是唯一的，就选择在字母顺序中最接近x的字母。如果有相连的字母，就选择在字母表中较后的那一个。

缺省的独立变量，有时称作自由变量，在表达式 **'1/(5+cos(x))'** 中是 **'x'** ；在 **'3*y+z'** 中是 **'y'** ；在 **'a+sin(t)'** 是 **'t'** 。在表式 **'sin(pi/4)-cos(3/5)'** 中自由符号变量是 **'x'** ，因为此式是一个符号常数无符号变量。可利用函数**symvar**询问MATLAB在符号表达式中哪一个变量它认为是独立变量。

```
>> symvar( ' a*x+y*)   %   find the default symbolic variable
ans=
  x

>> symvar( ' a*t+s/(u+3) ' )   %   u is the closest to   ' x '
ans=
  u

>> symvar( ' sin(omega) ' )   %    ' omega '   is not a singlee character。
ans=
  x

>> symvar( ' 3*i+4*j ' )   %   i and j are equel to sqrt(-1)
ans=
  x

>> symvar( ' y+3*s ' ,  ' t ' )   %   find the variable closest to t rather than x
ans=
  s
```

如果利用规则**symvar**不能找到一个缺省独立变量，它便假定无独立变量并返回x。这一结论对含有由多个字母组成的变量，如：alpha或s2的表达式，或不含变量的符号常数均成立。如果需要，绝大多数命令都使用用户选项以指定独立变量。

```
>> diff( ' x^n ' )   %    differentiate with respect to the default variable   ' x '
ans=
  x^n*n/x

>>   diff( ' x^n ' ,  ' n ' )   %    differentiate x^n with respect to   ' n '
ans=
  x^n*log(x)

>> diff( ' sin(omega) ' )   %    differentiate using the default variables (x)
ans=
  0

>> diff( ' sin(omega) ' ,  ' omega ' )   %    specify the independent variable
ans=
```

cos(omega)

22.3 符号表达式运算

一旦创建了一个符号表达式，或许想以某些方式改变它；也许希望提取表达式的一部分，合并两个表达式或求得表达的数值。有许多符号工具可以帮助完成这些任务。

所有符号函数(很少特殊例外的情况，讨论于后)作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可能看起来象一个数字，但事实上它是一个内部用字符串表示的一个符号表达式。正如我们前面所讨论的，可以运用MATLAB函数**isstr**来找出像似数字的表达式是否真是一个整数或是一个字符串。

提取分子和分母

如果表达式是一个有理分式(两个多项式之比)，或是可以展开为有理分式(包括哪些分母为1的分式)，可利用**numden**来提取分子或分母。例如，给定如下的表达式：

$$m = x^2 \qquad f = \frac{ax^2}{b-x} \qquad g = \frac{3}{2}x^2 + \frac{2}{3}x - \frac{3}{5} \qquad h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1} \qquad k = \begin{bmatrix} \frac{3}{2} & \frac{2x+1}{3} \\ \frac{4}{x^2} & 3x+4 \end{bmatrix}$$

在必要时，**numden**将表达式合并、有理化并返回所得的分子和分母。进行这项运算的MATLAB语句是：

```
>> m= ' x^2 '     %    create a simple expression
m=
     x^2

>> [n，d]=numden(m)    %    extract the numerator and denominator
n=
     x^2
d=
     1

>> f= ' a*x^2/(b-x) '     %    create a rational expression
f=
     a*x^2/(b-x)

>> [n，d]=numden(f)    %    extract the numerator and denominator
n=
     a*x^2
d=
     b-x
```

前二个表达式得到期望结果。

```
>> g= ' 3/2*x^2+2/3*x-3/5 '     %    rationalize and extract the parts
g=
     3/2*x^2+2/3*x-3/5

>> [n，d]=numden(g)
n=
     45*x^2+20*x-18
d=
     30

>> h= ' (x^2+3)/(2*x-1)+3*x/(x-1) '     %    the sum of rational polynomials
h=
     (x^2+3)/(2*x-1)+3*x/(x-1)

>> [n，d]=numden(h)    %    rationalize and extract
n=
     x^3+5*x^2-3
d=
     (2*x-1)*(x-1)
```

在提取各部分之前，这二个表达式g和h被有理化，并变换成具有分子和分母的一个简单表达式。

```
>> k=sym( ' [3/2，(2*x+1)/3；4/x^2，3*x+4] ' )    %     try a symbolic array
k=
     [     3/2，(2*x+1)/3]
     [4/x^2，          3*x+4]

>> [n，d]=numden(k)
n=
     [3，  2*x+1]
     [4，  3*x+4]
d=
     [     2，3]
     [x^2，1]
```

这个表达式k是符号数组，**numden**返回两个新数组**n**和**d**，其中**n**是分子数组，**d**是分母数组。如果采用**s=numden（f)**形式，**numden**仅把分子返回到变量**s**中。

标准代数运算

很多标准的代数运算可以在符号表达式上执行，函数**symadd**、**symsub**、**symlnul**和**symdiv**为加、减、乘、除两个表达式，**sympow**将一个表达式上升为另一个表达式的幂次。例如： 给定两个函数

$$f = 2x^2 + 3x - 5 \qquad g = x^2 - x + 7$$

```
>> f= ' 2*x^2+3*x-5 '     %    define the symbolic expression
f=
     2*x^2+3*x-5

>> g= ' x^2-x+7 '
g=
     x^2-x+7

>> symadd(f，g)    %    find an expression for f+g
 ans=
     3*x^2+2*x+2

>> symsub(f，g)    %    find an expression for f-g
ans=
     x^2+4*x-12

>> symmul(f，g)    %    find an expression for f*g
ans=
     (2*x^2+3*x-5)*(x^2-x+7)

>> symdiv(f，g)    %    find an expression for f/g
ans=
     (2*x^2+3*x-5)/(x^2-x+7)

>> sympow(f，  ' 3*x ' )    %    find an expression for  f^3*
ans=
     (2*x^2+3*x-5)^3**
```

另一个通用函数可让用户用其它的符号变量、表达式和算子创建新的表达式。**symop**取由逗号隔开的、多至16个参量。各个参量可为符号表达式、数值或算子(' + '、' - '、'*'、' / '、' ^ '、' ('或') ')，然后**symop**可将参量联接起来，返回最后所得的表达式.

```
>> f= ' cos(x) '     %    create an expression
f=
     cos(x)

>> g= ' sin(2*x) '     %    create another expression
g=
```

```
    sin(2*x)

>> symop(f，'/　′，g，'+'，3)　%　combine them
ans=
    cos(x)/sin(2*x)+3
```

所有这些运算也同样用数组参量进行。

高级运算

MATLAB具有对符号表达式执行更高级运算的功能。函数**compose**把*f(x)*和*g(x)*复合成*f(g(x))*。函数**finverse**求表达式的函数逆，而函数**symsum**求表达式的符号和。

给定表达式

$$f=\frac{1}{1+x^2} \qquad g=\sin(x) \qquad h=\frac{1}{1+u^2} \qquad \mathrm{k}=\sin(\mathrm{v})$$

```
>> f= ' 1/(1+x^2) ' ；　%　create the four expression

>> g= ' sin(x) ' ；

>> h= ' 1/(1+u^2) ' ；

>> k=' sin(v) ' ；

>> compose(f，g)　%　find an expression for f(g(x))
ans=
   1/(1+sin(x)^2)

>> compose(g，f)　%　find an expression for g(f(x))
ans=
   sin(1/(1+x^2))
```

compose也可用于含有不同独立变量的函数表达式。

```
>> compose(h，k，'u'，'v')　%　given h(u)，k(v)，find(k(v))
ans=
   1/(1+sin(v)^2)
```

表达式譬如*f(x)*的函数逆*g(x)*，满足*g(f(x))=x*。例如，e^x的函数逆是*ln(x)*，因为*ln*(e^x)=*x*。*sin(x)*的函数逆是*arcsin(x)*，函数$\frac{1}{\tan(x)}$的函数逆是*arcsin*$(\frac{1}{x})$。函数**fincerse**返回表达式的函数逆。如果解不是唯一就给出警告。

```
>> finverse( ' 1/x)    %    the inverse of 1/x is 1/x since    ' 1/（1/x)=x '
ans=
    1/x

>> finverse( ' x^2 ' )    %    g(x^2)=x has more than one solution
Warning：      finverse(x^2) is not unique
ans=
    x^(1/2)

>> finverse( ' a*x+b ' )    %    find the solution to    ' g(f(x))=x '
ans=
    -(b-x)/a

>> finverse( ' a*b+c*d-a*z ' )，    ' a ' )    %    find the solution to    ' g(f(a))=a '
ans=
    -(c*d-a)/(b-z)
```

symsun函数求表达式的符号和有四种形式：**symsun(f)**返回$\sum_{0}^{x-1} f(x)$；**symsum(f，'s')**返回$\sum_{0}^{s-1} f(s)$，**symsun(f，a，b)**返回$\sum_{a}^{b} f(x)$；最普通的形式**symsun(f，'s'，a，b)**返回$\sum_{a}^{b} f(s)$。

让我们试一试$\sum_{0}^{x-1} x^2$，它应返回：$\frac{x^3}{3}-\frac{x^2}{2}+\frac{x}{6}$。

```
>> symsum('x^2')
ans=
    1/3*x^3-1/2* x^2+1/6*x
```

$\sum_{1}^{n}(2n-1)^2$ 又怎么样呢?它应返回$\frac{n(2n-1)(2n+1)}{3}$。

```
>> sym('(2*n-1)^2'，1，'n')
ans=
    11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3

>> factor(ans)    %    change the form ( we will revisit 'factor' later on)
ans=
    1/3*n*(2*n-1)*(2*n+1)
```

最后让我们试一试$\sum_{1}^{\infty}\frac{1}{(2n-1)^2}$，其返回应是$\frac{\pi^2}{8}$。

```
>> symsum( ' 1/(2*n-1)^2 ' ， 1， inf)
ans=
    1/8*pi^2
```

变换函数

本节提出许多工具，将符号表达式变换成数值或反之。有极少数的符号函数可返回数值。然而请注意，某些符号函数能自动地将一个数字变换成它的符号表达式，如果该数字是函数许多参量中的一个。

函数**sym**可获取一个数字参量并将其转换为符号表达式。函数**numneric**的功能正好相反，它把一个符号常数（无变量符号表达式）变换为一个数值。

```
>> phi=' (1+sqrt(5))/2 '   %   the   ' golden '   ratio
phi=
    (1+sqrt(5))/2   %   convert to a numeric value

>> numeric(phi)
ans=
    1.6180
```

正如第六章所介绍，函数**eval**将字符串传给MATLAB以便计算。所以**eval**是另一个可用于把符号常数变换为数字或计算表达式的函数。

```
>> eval(phi)   %   execute the string   ' (1+sqrt(5))/2
ans=
    1.6180
```

正如所期望那样，**numeric**和**eval**返回相同数值。

符号函数**sym2poly**将符号多项式变换成它的MATLAB等价系数向量。函数**poly2syrn**功能正好相反，并让用户指定用于所得结果表达式中的变量。

```
>> f=' 2*x^2+x^3-3*x+5 '   %     f is the symbolic polynomials
f=
    2*x^2+x^3-3*x+5

>> n=sym2poly(f)   %   extract eht numeric coefficient vector
n=
    1    2    -3    5
```

```
>> poly2sym(n)   %   recreate the polynomials in x (the default)
ans=
     2*x^2+x^3-3*x+5

>> poly2sym(n，'s')   %   recreate the polynomials in s
ans=
     s^3+2*s^2-3*s+5
```

变量替换

假设有一个以x为变量的符号表达式，并希望将变量转换为y。MATLAB提供一个工具称作**subs**，以便在符号表达式中进行变量替换。其格式为**subs（f，new，old)**，其中f是符号表达式，**new**和**old**是字符、字符串或其它符号表达式。‘新’字符串将代替表达式f中各个‘旧'字符串。以下有几个例子：

```
>> f='a*x^2+b*x+c'    %   create a function f(x)
f=
     a*x^2+b*x+c

>> subs(f，'s'，'x')   %   substitute   's'   for   'x'   in the expression f
ans=
     a*s^2+b*s+c

>> subs(f，'alpha'，'a')   %   substitute   'alpha'   for   'a'   in   f
ans=
     alpha*x^2+b*x+c

>> g=' 3*x^2+5*x-4'   %   create another function
g=
     3*x^2+5*x-4

>> h=subs(g，'2'，'x')   %   substitute   '2' for   'x'   in   g
h=
     18

>> isstr(h)   %   show that the result is a symbolic expression
ans=
     1
```

最后一个例子表明**subs**如何进行替换，并力图简化表达式。因为替换结果是一个符号常数，MATLLAB可以将其简化为一个符号值。注意，因为**subs**是一个符号函数，所以它返回一个符号表达式。尽管看似数字，实质上是一个符号常数。为了得到数字，我们需要使用函数**numeric**或**eval**来转换字符串。

```
>> numeric(h)    %    convert a symbolic expression to a number
ans=
      18

>> isstr(ans)    %    show that the result is a numeric value
ans=
      0
```

22.4 微分和积分

微分和积分是微积分学研究和应用的核心，并广泛地用在许多工程学科。MATLAB符号工具能帮助解决许多这类问题。

微分

符号表达式的微分以四种形式利用函数**diff**：

```
>> f= ' a*x^3+x^2-b*x-c '      %    define a symbolic expression
f=
      a*x^3+x^2-b*x-c

>> diff(f)    %    differentiate with respect to the default variable x
ans=
      3*a*x^2+2*x-b

>> diff(f，'a ')    %      differentiate with respect to a
ans=
      x^3

>> diff(f，2)    %      differentiate twice with respect to x
ans=
      6*a*x+2

>> diff(f，' a '，2)    %      differentiate twice with respect to a
ans=
      0
```

函数**diff**也可对数组进行运算。如果F是符号向量或数组，**diff(F)**对数组内的各个元素进行微分。

```
>> F=sym(' [a*x，     b*x^2；     c*x^3，     d*s] ')    %    create a symbolic array
F=
  [     a*x，     b*x^2]
  [c*x^3，          d*s]
```

```
>> diff(F)      %     differentiate the element with respect to x
ans=
  [              a，2*b*x]
  [3*c*x^2，          0]
```

注意函数**diff**也用在MATLAB，计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵M，diff(M)计算M(2： m，：)-M(1： m-1，：)的数值差分，如下所示：

```
>> m=[(1： 8).^2)]   %   create a vector
        M=
            1    4    9    16    25    36    49    64

        >> diff(M)   %   find the differences between elements
        ans=
            3    5    7    9    11    13    15
```

如果diff的表达式或可变参量是数值，MATLAB就非常巧妙地计算其数值差分；如果参量是符号字符串或变量，MATLAB就对其表达式进行微分。

积分

积分函数**int(f)**，其中f是一符号表达式，它力图求出另一符号表达式F使**diff(F)=f**。正如从研究微分学所了解的，积分比微分复杂得多。积分或逆求导不一定是以封闭形式存在，或许存在但软件也许找不到，或者软件可明显地求解，但超过内存或时间限制。当MATLAB不能找到逆导数时，它将返回未经计算的命令。

```
>> int( ' log(x)/exp(x^2) ' )   %   attempt to integrate
ans=
    log(x)/exp(x^2)
```

同微分一样，积分函数有多种形式。形式**int(f)**相对于缺省的独立变量求逆导数；形式**(f, 's ')**相对于符号变量s积分；形式**int(f，a，b)**和**int(f，'s'，a，b)，a，b**是数值，求解符号表达式从**a**到**b**的定积分；形式**int(f，'m' ，'n ')**和形式**int(f，'s'，'m'，'n ')**，其中**m，n**是符号变量，求解符号表达式从**m**到**n**的定积分。

```
>> f=' sin(s+2*x) ')   %   crate a symbolic function
f=
    sin(s+2*x)

>> int(f)   %   integrate with respect to x
ans=
    -1/2*cos(s+2*x)
```

```
>> int(f，' s ')     %    integrate with respect to s
ans=
      -cos(s+2*x)

>> int(f，pi/2，pi)    %    integrate with respect to x from π/2 to π
ans=
      -cos(x)

>> int(f，' s '，pi/2，pi)    %    integrate with respect to s from π/2 to π
ans=
      cos(2*x)-sin(2*x)

>> int(f，' m '，' n ')     %    integrate with respect to x from m to n
ans=
      -1/2*cos(s+2*n)+1/2*cos(s+2*m)
```

正如函数**diff**一样，积分函数**int**对符号数组的每一个元素进行运算。

```
>> F=sym( ' [a*x，b*x^2；c*x^3，d*s] ' )    %    create a symbolic array
F=
  [ a*x，b*x^2]
  [c*x^3，  d*s]

>> diff(F)    %    ubtegrate the array elements with respect to x
ans=
  [1/2*a*x^2，1/3*b*x^3]
  [1/4*c*x^4，        d*s*x]
```

22.5　符号表达式画图

在许多的场合，将表达式可视化是有利的。MATLAB提供了函数**ezplot**来完成该任务。

```
>> y=' 16*x^2+64*x+96 '    %    expression to plot
y=
  16*x^2+64*x+96

>> ezplot(y)
```

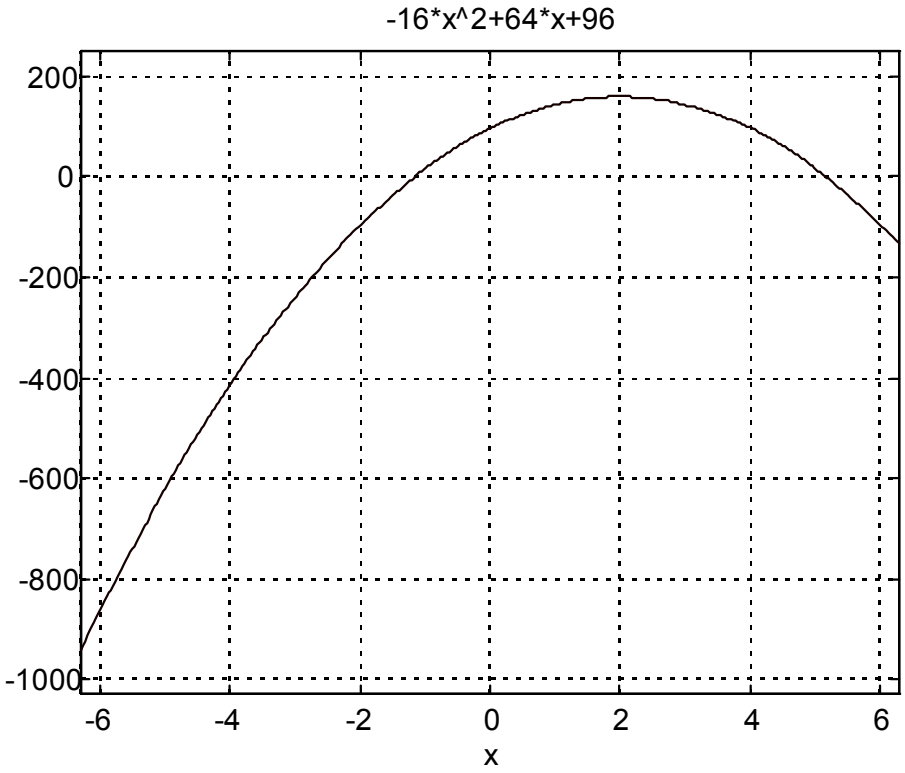


图22.1　符号函数16*x^2+64*x+96　(-2π≤x≤2π)

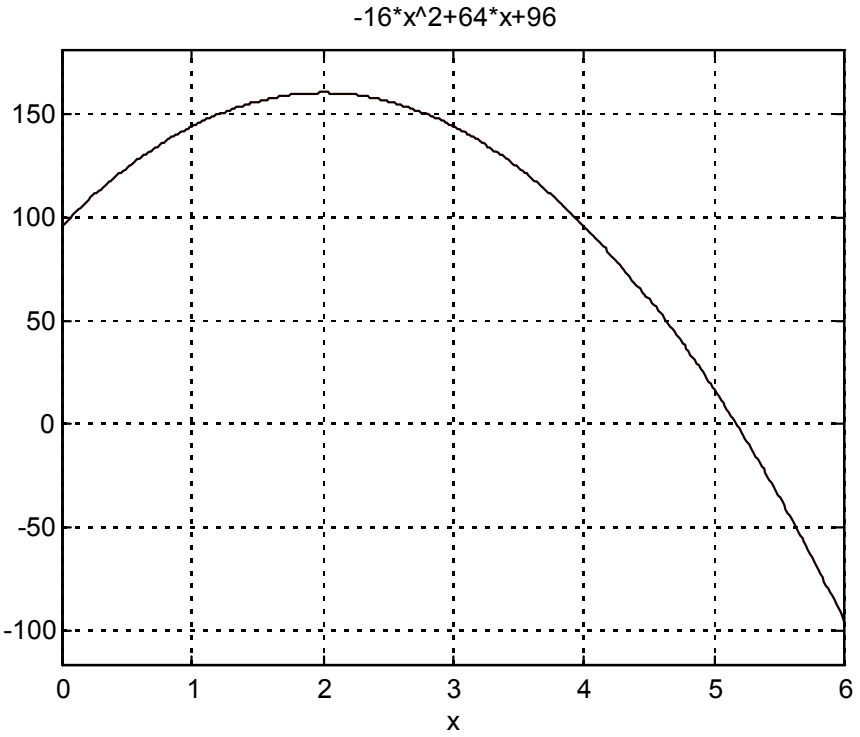


图22.2　符号函数16*x^2+64*x+96　　0≤x≤6

正如图22.1所示，**ezplot**绘制了定义域为-2π≤x≤2π的给定符号函数，并相应地调整了y轴比例，还加了网格栅和标志。在这个例子中，我们感兴趣的时间是从0到6。让我们再试一下，并指定时间范围，见图22.2

```
>> ezplot(y，[0 6])    %    plot y for 0≤x≤6
```

现在，在所感兴趣的范围内显示得略好些。一旦该图处在图形窗口内，它可以象其它图象一样作修改。

22.6　符号表达式简化和格式化

有时MATLAB返回的符号表达式难以理解，有许多工具可以使表达式变得更易读懂。第一个就是函数**pretty**，该命令以类似于数学课本上的形式来显示符号表达式。

```
>> f=taylor( ' log(x+1)/(x-5) ' )    %      6 terms is the default
f =
  -1/5*x+3/50*x^2-41/750*x^3+293/7500*x^4-1207/37500*x^5+O(x^6)

>> pretty(f)
                                          2        41     3        293     4        1207
5              6
                - 1/5 x + 3/50 x    -    -----x      + ------x      -    -------x    + O(x )
                                                 750              7500             37500
```

符号表达式可用许多等价形式来提供。在不同的场合，某种形式可能胜于另一种。MATLAB用许多命令来简化或改变符号表达式。

```
>> f=sym( ' (x^2-1)*(x-2)*(x-3) ' )    %    create a function
f=
      (x^2-1)*(x-2)*(x-3)

>> collect(f)    %    collect all like terms
ans=
      x^4-5*x^3+5*x^2+5*x-6

>> hornor(ans)      %    change to Horner or nested representation
ans=
      -6+(5+(5+(-5+x)*x)*x)*x

>> factor(ans)    %    express as a product of polynomials
ans=
      (x-1)*(x-2)*(x-3)*(x+1)

>> expand(f)    %    distribute products over sums
```

```
ans=
    x^4-5*x^3+5*x^2+5*x-6
```

Simplify是功能强大、通用的工具。它利用各种类型代数恒等式，包括求和、积分和分数幂、三角、指数和log函数、Bessel函数、超几何函数和γ函数，来简化表达式。

下面例子说明函数的乘幂。

```
>> simplify( ' log(2*x/y) ' )
ans=
    log(2)+log(x)-log(y)

>> simplify( ' sin(x)^2+3*x+cos(x)^2-5 ' )
ans=
    3*x-4

>> simplify( ' (-a^2+1)/(1-a) ' )
ans=
    a+1
```

其中要讨论的最后一个函数是最有用的，但也是最不正统的。函数**simple**试用了几种不同的简化工具，然后选择在结果表达式中含有最少字符的那种形式。让我们看一下立方根：

$$f=\sqrt[3]{\frac{1}{x^3}+\frac{6}{x^2}+\frac{12}{x}+8}$$

```
>> f=' (1/x^3+6/x^2+12/x+8)^(1/3) '    %    create the expression
f=
    (1/x^3+6/x^2+12/x+8)^(1/3)

>> simple(f)    %    simplify it
simplify ：
    (2*x+1)/x
ans=
    (2*x+1)/x

>> simple(ans)    %    try it once again - another method may help
combine(trig)
    2+1/x
ans=
    2+1/x
```

正如所见，**simple**试用了几种可简化表达式的简化方式，并让看到每一个尝试的结果。有时，它帮助多次使用函数**simple**并对第一次的结果作不同的简化操作，如上所作。**simple**对于含有三角函数的表达式尤为有用。让我们试一下$\cos(x)+\sqrt{-\sin(x)^2}$

```
>> simple( ' cos(x)+sqrt(-sin(x)^2) ' )    %    simplify a trig expression
simplify：
    cos(x)+(cos(x)^2-1)^(1/2)
radsimp：
    cos(x)+i*sin(x)
combine(trig)：
    cos(x)+(-1/2+1/2*cos(2*x))^(1/2)
factor：
    cos(x)+(-sin(x)^2)^(1/2)
expand：
    cos(x)+(-sin(x)^2)^(1/2)
convert(exp)：
    1/2*exp(i*x)+1/2/exp(i*x)+1/4*4^(1/2)*(exp(i*x)-1/exp(i*x))
convert(tan)：
    (1-tan(1/2*x)^2)/(1+tan(1/2*x)^2)+(-4*tan(1/2*x)^2/(1+tan(1/2*)^2)^2)^(1/2)
ans =
    cos(x)+i*sin(x)

>> simple(ans)   %    one   more   time
convert(exp)：
                exp(i*x)
convert(tan)：
                (1-tan(1/2*x)^2)/(1+tan(1/2*x)^2)+2*i*tan(1/2*x)/(1+tan(1/2*x)^2)
ans =
                exp(i*x)
```

22.7　可变精度算术运算

因为数值的精度受每次操作所保留的数位的限制，所以数值的任何运算都会引入舍入误差，重复的多次数值运算会造成累积误差。而对符号表达式的运算是非常准确的，因为它们不需要进行数值运算，所以无舍入误差。对符号运算结果用函数**eval**或**numeric**，仅在结果转换时会引入舍入误差。

MATLAB对数的处理完全依靠计算机的浮点算术运算，显然在内存中进行运算，又快又好，只是浮点运算受到所支持字长的限制，每次操作会引入舍入误差，所以不能产生精确的结果。MATLAB中各个算术运算的相对精度大约是16位。相反，Maple的符号处理能力可以实现任何数位的运算。当缺省的数位增加时，每次计算就需要附加时间和计算机内存。

Maple缺省为16位的精度。函数**digits**返回全局**Digits**参数的当前值。Maple缺省准确度可以由**digits(n)**来改变，其中n是所期望的准确度数位。用这种方法增加准确度的副作用是，每个Maple函数随后进行的计算都以新的准确度为准，增加了计算时间。结果的显示不会改变，只有所用的Maple函数的缺省准确度受到影响。

另外有一个函数，它可以任何精度实行单个计算，而使全局的**Digits**参数不变。即：可变精度的算术或函数**vpa**，它以缺省的精度或任何指定的精度对单个符号表达式进行计算，并以同样的精度来显示结果。

```
>> format long    %    let ' s see all the usual digits

>> pi    %    how about  π  to numeric accuracy
ans=
  3.14159265358979

>> digits    %    display the default    ' Digits '    value
Digits=16

>> vpa(' pi ')    %    how about  π  to    ' Digits '    value
ans=
  3.141592653589793

>> digits(18)    %    change the default to 18 digits

>> vpa(' pi ')    %    how about  π  to    ' Digits '    accuracy
ans=
  3.14159265358979324

>> vpa( ' pi ' ，20)    %    how about    π  to 20 digits
ans=
  3.1415926535897932385

>> vpa( ' pi ' ，50)    %    how about  π  to 50 digits
ans=
  3.14159265358979323846264338327950288419716939937751

>> vpa( ' 2^(1/3) ' ，200)     %    the cube root of 2 to 200 digits
ans=
1.25992104989487316476721060727822835057025146470150798008197511215529967651395948372939656243625509415431025603561566525939900240406137372284591103042
693552469606426166250009774745265654803068671854055
```

将函数**vpa**作用于符号矩阵，对它的每一个元素进行计算也同样达到所指定的位数。

```
>> A=sym( ' [1/4，log(sqrt(2))；exp(1)，3/7] ' )
A=
     [     1/4  ，   log(sqrt(2))]
     [exp(1)  ，              3/7)]

>> vpa(A，20)    %      evaluate to 20 digits
ans=
```

```
[ .2500000000000000000,    .34657359027997265471]
[2.7182818284590452354,    .42857142857142857143]
```

22.8 方程求解

用MATLAB所具有的符号工具可以求解符号方程。有一些工具已经在前面介绍过，更多将在本节予以检验。

求解单个代数方程

我们在前面已经看到，MATLAB具有求解符号表达式的工具。如果表达式不是一个方程式(不含等号)，则在求解之前函数**solve**将表达式置成等于0。

```
>> solve( ' a*x^2+b*x+c ' )   %   solve for the roots of the quadratic eqution
ans=
    [1/2/a*(-b+(b^2-4*a*c)^1/2)]
    [1/2/a*(-b-(b^2-4*a*c)^1/2)]
```

结果是符号向量，其元素是方程的2个解。如果想对非缺省x变量求解，**solve**必须指定变量。

```
>> solve( ' a*x^2+b*x+c '  ,  ' b ' )   %   solve for b
ans=
    -(a*x^2+c)/x
```

带有等号的符号方程也可以求解。

```
>> f=solve( ' cos(x)=sin(x) ' )   %   solve for x
f=
    1/4*pi

>> t=solve( ' tan(2*x)=sin(x) ' )
t=
    [                                    0]
    [acos(1/2+1/2*3^(1/2))]
    [acos(1/2=1/2*3^(1/2))]
```

并得到数值解。

```
>> numeric(f)
ans=
    0.7854

>> numeric(t)
```

```
ans=
      0
      0 + 0.8314i
     1.9455
```

注意在求解周期函数方程时，有无穷多的解。在这种情况下，**solve**对解的搜索范围限制在接近于零的有限范围，并返回非唯一的解的子集。

如果不能求得符号解，就计算可变精度解。

```
>> x=solve( ' exp(x)=tan(x) ' )
x=
     1.306326940423079
```

代数方程组求解

可以同时求解若干代数方程，语句**solve(s1，s2，.....，sn)**对缺省变量求解**n**个方程，语句**solve(s1，s2，...，sn，'v1，v2，...，vn ')**对**n**个**' v1，v2，...vn** '的未知数求解**n**个方程。

如何处理中小学典型的代数问题？

黛安娜(Diane)想去看电影，她从小猪存钱罐倒出硬币并清点，她发现：

10美分的硬币数加上5美分的硬币总数的一半等于25美分的硬币数。

1美分的硬币数比5美分、10美分以及25美分的硬币总数多10。

25美分和10美分的硬币总数等于1美分的硬币数加上1/4的5美分的硬币数

25美分的硬币数和1美分的硬币数比5美分的硬币数加上8倍的10美分的硬币数多1。

如果电影票价为3. 00美元，爆米花为1. 00美元，糖棒为50美分，她有足够的钱去买这三样东西？

首先，根据以上给出的信息列出一组线性方程，假如*p*，*n*，*d*和*q*分别表示1美分，5美分，10美分，和25美分的硬币数

$$d+\frac{n+p}{2}=q \qquad p=n+d+q-10 \qquad q+d=p+\frac{n}{4}$$

$$q+p=n+8d-1$$

然后，建立MATLAB符号方程并对变量求解。

```
>> eq1= ' d+(n+p)/2=q ' ；

>> eq2= ' p=n+d+q-10 ' ；

>> eq3= ' q+d=p+n/4 ' ；
```

```
>> eq4= ' q+p=n+8*d-1 ' ；

>>[pennies，nickles，dimes，quarters]=solve(equ1，equ2，equ3，equ4，'p，n，d，q ')
pennies=
     16
nickles=
     8
dimes=
     3
quarters=
     15
```

所以，黛安娜有16枚1美分的硬币，8枚5美分的硬币，3枚10美分的硬币，15枚25美分的硬币，这就意味着

```
>> money=.01*16+.05*8+.10*3+.25*15
money=
     4.6100
```

她就有足够的钱去买电影票，爆米花和糖棒并剩余11美分。

单个微分方程

常微分方程有时很难求解，MATLAB提供了功能强大的工具，可以帮助求解微分方程。函数**dsovle**计算常微分方程的符号解。因为我们要求解微分方程，就需要用一种方法将微分包含在表达式中。所以，**dsovle**句法与大多数其它函数有一些不同，用字母**D**来表示求微分，**D2**，**D3**等等表示重复求微分，并以此来设定方程。任何**D**后所跟的字母为因变量。方程d^2y/dx^2=0用符号表达式D2y=0来表示。独立变量可以指定或由**symvar**规则选定为缺省。例如，一阶方程*dy/dx*=1+y^{2}的通解为：

```
>> dsolve( ' Dy=1+y^2 ' )   %   find the general solution
ans=
     -tan(-x+C1)
```

其中，**C1**是积分常数。求解初值*y(0)=1*的同一个方程就可产生：

```
>>   dsolve(' Dy=1+y^2 '，' y(0)=1 ')   %   add an initial condition
y=
     tan(x+1/4*pi)
```

独立变量可用如下形式指定：

```
>>   dsolve(' Dy=1+y^2 '，' y(0)=1 '，' v ')   %   find solution to dy/dv
```

```
ans=
      tan(v+1/4*pi)
```

让我们举一个二阶微分方程的例子，该方程有两个初始条件：

$$\frac{d^2y}{dx^2}=\cos(2x)-y \qquad \frac{dy}{dx}(0)=0 \quad y(0)=1$$

```
>> y=dsolve(' D2y=cos(2*x)-y ', ' Dy(0)=0 ', ' y(0)=1 ')
y=
      -2/3*cos(x)^2+1/3+4/3*cos(x)

>> y=simple(y)   %   y looks like it can be simplified
y=
      -1/3*cos(2*x)+4/3*cos(x)
```

通常，要求解的微分方程含有一阶以上的项，并以下述的形式表示：

$$\frac{d^2y}{dx^2}-2\frac{dy}{dx}-3y=0$$

通解为：

```
>> y=solve( 'D2y-2Dy-3*y=0 ')
y=
      C1*exp(-x)+C2*exp(3*x)
```

加上初始条件：*y(0)=0*和*y(1)=1*可得到：

```
>> y=solve( ' D2y-2Dy-3*y=0 ' ,  ' y(0)=0，y(1)=1 ' )
y=
      1/(exp(-1)-exp(3))*exp(-x)-1/(exp(-1)-exp(3))*exp(3*x)

>> y=simple(y)   %   this looks like a candidate for simplification
y=
      -(exp(-x)-exp(3*x))/(exp(3)-exp(-1))

>> pretty(y)   %      pretty it up
            exp(-x)-exp(3 x)
      - ----------------------
              exp(3) -exp(-1)
```

现在来绘制感兴趣的区域内的结果。

```
>> ezplot(y，[-6 2])
```

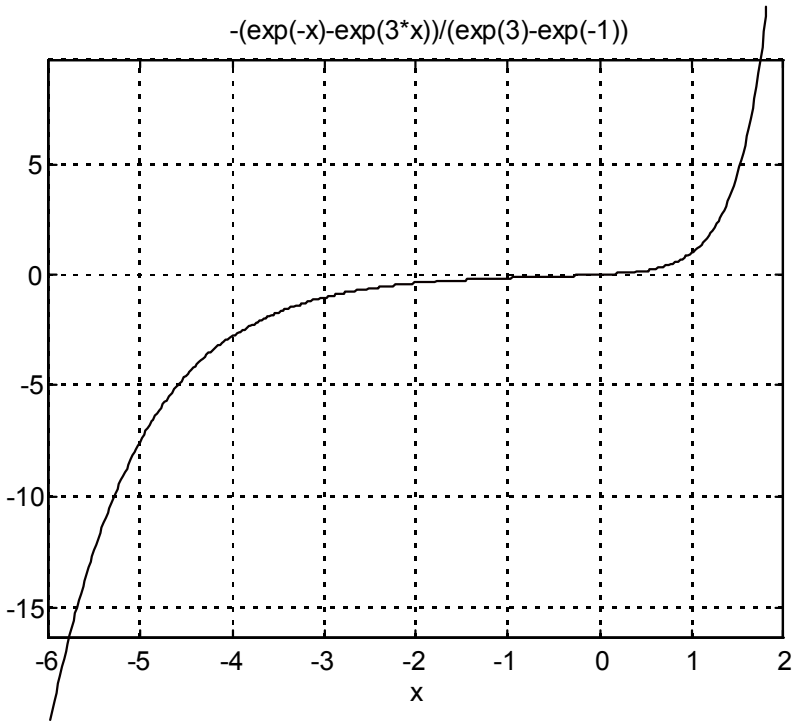


图22.3　符号函数y=-(exp(-x)-exp(3*x))/(exp(3)-exp(-1))　(-6≤x≤2)

微分方程组

函数**dsolve**也可同时处理若干个微分方程式，下面有两个线性一阶方程。

$$\frac{dy}{dx}=3f+4g \qquad \frac{dg}{dx}=-4f+3g$$

通解为：

```
>> [f，g]=dsolve('Df=3*f+4*g'，'Dg=-4*f+3*g')
f=
    C1*exp(3*x)*sin(4*x)+C2*exp(3*x)*cos(4*x)
g=
    -C2*exp(3*x)*sin(4*x)+C1*exp(3*x)*cos(4*x)
```

加上初始条件：*f(0)=0*和*g(0)=1*，我们可以得到：

```
>> [f，g]=dsolve('Df=3*f+4*g'，'Dg=-4*f+3*g'，'f(0)=0，g(0)=1')
f=
  exp(3*x)*sin(4*x)
g=
  exp(3*x)*cos(4*x)
```

22.9 线性代数和矩阵

在本节中，我们将介绍符号矩阵和MATLAB提供的工具，它用线性代数求解问题。

符号矩阵

符号矩阵和向量是数组，其元素为符号表达式，可用函数**sym**来产生。

```
>> A=sym( ' [a，  b，  c；  b，  c，  a；  c，  a，  b] ' )
A=
        [ a，  b，  c]
        [ b，  c，  a]
        [ c，  a，  b]

>> G=sym( ' [cos(t)，  sin(t)；-sin(t)，cos(t)] ' )
G=
        [ cos(t)，  sin(t)]
        [-sin(t)，  cos(t)]
```

函数**sym**也可以扩展成定义各元素的公式。**注意，只有在这种情况下，i，j分别表示行列的位置；且不影响i，j的缺省值**（它代表 $\sqrt{-1}$ ）。下面的例子建立了3×3的矩阵，其元素依赖于行和列的位置。

```
>> S=sym(3，3，' (i+j)+(i-j+s) ')   %  create a matrix using a formula
S=
  [       2/s，      3/(-1+s)，      4/(-2+s)]
  [1/(1+s)，        4/s，        s/(-1+s)]
  [4/(2+s)，      5/(1+s)，        6/s]

>> S=sym(3，  3，  ' m ',  ' n ',  ' (m-n)/(m-n-t) ')  %  use m and n in another formula
S=
        [          0，      -1/(-1-t)，      -2/(-2-t)]
        [1/(1-t)，          0，      -1/(-1-t)]
        [2/(2-t)，      1/(1-t)，          0]
```

函数**sym**也可以把数值矩阵转换成符号形式

```
>> M=[1.1，  1.2，  1.3；  2.1，  2.2，  2.3；  3.1，  3.2，  3.3]  %  a numeric matrix
M=
        1.1000    1.2000    1.3000
        2.1000    2.2000    2.3000
```

```
                3.1000      3.2000      3.3000

>> S=sym(M)   %   convert to symbolic form
S=
                [11/10，      6/5，      13/10]
                [21/10，     11/5，      23/10]
                [31/10，     16/5，      33/10]
```

如果数值矩阵的元素可以指定为小的整数之比，则函数**sym**将采用有理分式表示。如果元素是无理数，则在符号形式中**sym**将用符号浮点数表示元素。

```
>> E=[exp(1) sqrt(2)]
E=
    2.7183    1.4142

>> sym(E)
ans=
    [3060513257434036*2^(-50)，          3184525836262886*2^(-51)]
```

用函数**symsize**可以得到符号矩阵的大小(行，列数)。函数返回数值或向量，而不是符号表达式。**symsize**的四种形式说明如下：

```
>> S=sym(' [a，b，c；d，e，f] ')   %   create a symbolic matrix
S=
   [a，b，c]
   [d，e，f]

>> d=symsize(S)   %   return the size of S as the 2-element vector d
d=
   2      3

>> [m，n]=symsize(S)   %   return the number of rows in m，  and column in n
m=
   2
n=
   3

>> m=symsize(S，1)   %   return the number of rows
m=
   2

>> n=symsize(S，2)   %   return the number of columns
n=
   3
```

数值数组用**N(m，n)**形式来访问单个元素，但符号数组元素必须用函数如**sym(S，m，n)**来获取。若能用相同的句法当然好，但MATLAB符号表达式的表示不方便。在内部，符号数组表示成一个字符串数组；而**S(m，n)**返回单个字母。所以，符号数组的各个元素，必须由符号函数，如**sym**，来指定,而不是直接指定。

```
>> G=sym(' [ab，cd；ef，gh] ')  %  create a 2-by-2 symbolic matrix
G=
  [ab，  cd]
  [ef ，  gh]

>> G(1，2)  %  this is the second character of the first row of G
ans=
  a
>> r=sym(G，1，2)  %  this is the second expressinon in the first row of G
r=
  cd
```

记住，在上例中的符号矩阵**G**，实际是以2×7字符数组存储在计算机中。第一行为' [ab，cd] '，所以第二个元素是' a '。

最后，**sym**可以用于改变符号数组的一个元素。

```
>> sym(G，2，2，' pq ')  %  change the (2，2) element in G from  ' gh '  to  ' pq
'
ans=
  [ab，   cd]
  [ef，    pq]
```

代数运算

用函数**symadd**，**symsub**，**symmul**和**symdiv**，对符号矩阵可以执行许多通用的代数运算，用**sympow**可计算乘幂，用**transpose**计算符号矩阵的转置。

```
>> G=sym(' [cos(t)，sin(t0；-sin(t0，cos(t)] ')  %  create a symbolic matrix
G=
  [ cos(t)，   sin(t)]
  [ -sin(t)，  cos(t)]

>> symadd(G，  ' t ' )  %  add  ' t '  to each element
ans=
  [ cos(t)+t，        sin(t)+t]
  [ -sin(t)+t，       cos(t)+t]

>> symmul(G，G)  %  multiply G by G；  Synpow(G，2) does the same thing
```

```
ans=
  [cos(t)^2-sint(t)^2，              2*cos(t)*sin(t)]
  [ -2*cos(t)* sin(t)，          cos(t)^2-sin(t)^2]

>> simple(G)    %    try to simplify
ans=
  [cos(2*t)，     sin(2*t)]
  [-sin(2*t)，   cos(2*t)]
```

下面通过证明**G**的转置是它的逆来证明G是正交阵。

```
>> I=symmul(G，transpose(G))    %    multiply G by its transpose
I=
  [cos(t)^2+sin(t)^2，                                    0]
  [                                  0，   cos(t)^2+sin(t)^2]
>> simplify(I)    %    there appears to be a trig identity here
ans=
  [1，   0]
  [0，   1]
```

正如所期望的那样，这是单位阵。

线性代数运算

用函数**inverse**和**determ**，可计算符号矩阵的逆阵以及行列式。

```
>> H=sym(hilb(3))    %    the symbolic form of the numeric 3-by-3 Hilbert matrix
H=
  [     1，   1/2，   1/3]
  [1/2，   1/3，   1/4]
  [1/3，   1/4，   1/5]

>> determ(H)    %    dind the determinant of H
ans=
  1/2160

>> J=inverse(H)    %    find the inverse of H
J=
  [     9，        -36，             30]
  [-36，       192，        -180]
  [ 30，     -180，          180]

>> determ(J)    %    find the determinant of the inverse
ans=
```

2160

用函数**linsolve**求解齐次线性方程；这是等价于基本的MATLAB的逆斜杠\算子的符号，**linsolve(A，B)**对X方阵求解矩阵方程*A*X=B*。回到以前的硬币问题：

黛安娜想去看电影，她从小猪存钱罐倒出硬币并清点，她发现：

10美分的硬币数加上5美分的硬币总数的一半等于25美分的硬币数。
1美分的硬币数比5美分、10美分以及25美分的硬币总数多10。
25美分和10美分的硬币总数等于1美分的硬币数加上1/4的5美分的硬币数
25美分的硬币数和1美分的硬币数比5美分的硬币数加上8倍的10美分的硬币数多1。。

象上次所做的那样，列出线性方程组，令*p*，*n*，*d*和*q*分别为1美分，5美分，10美分，和25美分的硬币数

$$d+\frac{n+p}{2}=q \qquad p=n+d+q-10 \qquad q+d=p+\frac{n}{4} \qquad q+p=n+8d-1$$

重新以p，n，d，q的顺序排列表达式

p/2+n/2+d-q=0　　p-n-d-q=-10　　-p-n/4+d+q=0
p-n-8d+q=1

接下来，列出方程系数的符号数组。

```
>> A=sym(' [1/2，1/2，1，-1；1，-1，-1，-1；-1，-1/4，1，1；1，-1，-8，1] ')
A=
                    [1/2，      1/2，      1，  -1]
                    [   1，       -1，     -1，  -1]
                    [  -1，   -1/4，      1，     1]
                    [   1，       -1，     -8，     1]

>> B=sym(' [0；-10；0；-1] ')   %   Create the symbolic vector B
B=
                    [   0]
                    [-10]
                    [   0]
                    [  -1]

>> X=linsolve(A，B)   %   solve the symbolic system A ' X=B for x
x=
                    [16]
                    [   8]
```

```
[   3]
[15]
```

结果是相同的，黛安娜有16枚1美分的硬币，8枚5美分的硬币，3枚10美分的硬币，15枚25美分的硬币。

其他特性

symop将其参量串接起来，并计算所得到的表达式。

```
>> f=' cos(x) '   %   create an expression
f=
     cos(x)

>> symop( ' atan( ' ， f， ' + ' ， a， ' ) ' ， ' ^2 ' )
ans=
     atan(cos(x)+a)^2
```

在用函数**symop**时，若将数组和标量混合应慎重。例如：

```
>> M=sym( ' [a b；  c d] ' )
M=
     [a，    b]
     [c，    d]

>> symop(M， ' + ' ， ' t ' )
ans=
     [a+t，       b]
     [    c，  d+t]
```

是把**t**加到**M**的对角线上。
函数**charpoly**求解矩阵的特征多项式。

```
>> G=sym(' [1，1/2；1/3，1/4] ')   %    create a symbolic matrix
G=
  [    1，  1/2]
  [1/3，  1/4]

>> charpoly(G)   %    find the characteristic polynomial of G
ans=
  x ' 2-5/4*x+1/12
```

用函数**eigensys**可以求得符号矩阵的特征根和特征向量

```
>> F=sym(' [1/2，1/4；1/4，1/2] ')  %    create a symbolic matrix
F=
  [1/2，1/4]
  [1/4，1/2]

>> eigensys(F)  %    find the eigenvalues of F
ans=
  [3/4]
  [1/4]

>> [V，E]=eigensys(F)  %    find eigenvalues E and eigenvectors v
V=
  [-1，  1]
  [ 1，  1]
E=
  [1/4]
  [3/4]
```

矩阵的约当(Jordan)标准型是特征值的对角矩阵；转换矩阵的列是特征向量。对于给定的矩阵A， **jordan(A)**求出非奇异的矩阵**V**，使得**inv(V)*A*V**成为约当标准型，函数**jordan**有两种形式

```
>> jordan(F)  %    find the Jordan form of F ，above
ans=
  [1/4，        0]
  [    0，  3/4]

>> [V，J]=jordan(F)  %    find the Jordan form and eigenvectors
V=
  [ 1/2，  1/2]
  [-1/2，  1/2]
J=
  [1/4，        0]
  [    0，    3/4]
```

标准形和特征向**V**的列是某些F可能的特征向量。

因为**F**是非奇异的，**F**的零空间的基是空矩阵，而列空间的基是单位阵。

```
>> F=sym( ' [1/2，1/4；1/4，1/  2] ' )  %    recreate F
F=
     [1/2，  1/4]
     [1/4，  1/2]

>> nullspace(F)  %  the nullspace of F is the empty matrix
```

```
ans=
     []

>> colspace(F)    %    find the column space of F
ans=
     [1,     0]
     [0,     1]
```

用函数**singvals**可求解矩阵奇异值。

```
>> A=sym(magic(3))    %    Generate a 3-by-3 matrix
A=
     [8,    1,    6]
     [3,    5,    7]
     [4,    9,    2]

>> singvals(A)    %    find the singular expressions
ans=
     [15.00000000000000]
     [6.928203230275511]
     [3.464101615137752]
```

函数**jacobian(w，v)**可相对于**v**求解**w**的雅可比(Jacobia)值。其结果的第**(i，j)**项是**df(i)/dv(j)**。注意，当**f**是标量时，**f**的雅可比值是f的梯度。

```
>> jacobian( ' u*exp(v) ' ，sym( ' u，v ' ))
ans=
     [exp(v，u*exp(v)]
```

22.10　小结

下列各表(表22.2-表22.8)综合了符号数学工具箱的特性：

表22.2

符号表达式的运算	
numeric	符号到数值的转换
pretty	显示悦目的符号输出
subs	替代子表达式
sym	建立符号矩阵或表达式
symadd	符号加法
symdiv	符号除法
symmul	符号乘法

symop	符号运算
sympow	符号表达式的幂运算
symrat	有理近似
symsub	符号减法
symvar	求符号变量

表22.3

符号表达式的简化	
collect	合并同类项
expand	展开
factor	因式
simple	求解最简形式
simplify	简化
symsum	和级数

表22.4

符号多项式	
charpoly	特征多项式
horner	嵌套多项式表示
numden	分子或分母的提取
poly2sym	多项式向量到符号的转换
sym2poly	符号到多项式向量的转换

表22.5

符号微积分	
diff	微分
int	积分
jordan	约当标准形
taylor	泰勒级数展开

表22.6

符号可变精度算术	
digits	设置可变精度
vpa	可变精度计算

表22.7

求解符号方程	
compose	函数的复合
dsolve	微分方程的求解
finverse	函数逆
linsolve	齐次线性方程组的求解
solve	代数方程的求解

表22.8

符号线性代数	
charploy	特征多项式
determ	矩阵行列式的值
eigensys	特征值和特征向量
inverse	矩阵逆
jordan	约当标准形
linsolve	齐次线性方程组的解
transpose	矩阵的转置

第23章　Internet资源

访问Internet，不论是全面存取或发送简单的电子邮件，MATLAB有可以利用的丰富资源。通过从教育机构、政府部门及商业公司直接与Internet相连,以及通过商业的Internet所提供的咨询和在线服务如：America　Online、Delphi、CompuServe等等，都可获得这些资源。

23.1　USENET新闻组

在Internet上一个最重要的信息资源和论坛就是一个称之为NetNews FeeD、USENET或简称为News的公告栏系统。它是一个巨大的消息或公告的集合，在全球范围内从一个计算机到另一个计算机流动。News是由成千上万的单个的新闻公告组成，它收集在数以千计的新闻组或标题中。许多站点建立了可供利用的所有新闻组，而其它一些地方由于磁盘空间的限制或公司策略关系则更具选择性。

如果访问NetNews，就可以阅读MATLAB的其他用户发出的新闻公告或者发布你自己的评论或问题。MATLAB新闻组称为**comp.soft-sys.matlab**。订阅这个新闻组,可以通过提问、评论和从其他用户以及从**Mathworks**职员来的解答、浏览新闻。实际上，MATLAB的开发者经常向论坛发布消息。如果在用户站点上得到新闻组有限，不能找到comp.soft-sys.matlab，就请与当地的新闻管理人员联系，要求将**comp.soft-sys.matlab**包括进来。

要知道，这个新闻组是未作修饰的，而这意味任何人都可以发布自己的提问或评论,无法将不适当的公告过滤,全世界有数以千计的人阅读这个新闻组,所以发布时要慎重和适当。

如果不能访问NetNews，但具有FTP功能，则如下面讨论，可以从匿名FTP站点上得到这个新闻的公告摘要，即用目录**/pub/doc/cssm-digest/**中的**ftp.mathworks.com**。

如果没有News，且FTP不可选，但有连接到Internet的电子邮件(E-mail)，在**Mathworks**有一个邮件－新闻的网关，允许通过E-mail来向新闻组发布消息。由E-mail发往**comp.soft-sys.matlab@mathworks.com**的消息将发布到新闻组上。如果你不能阅读新闻组，则一定要在发布的消息中加入请求，使回答直接用E-mail送回给你，同时要提供E-mail地址。

23.2　匿名FTP

MATLAB用户的一个最有用的资源是由**Mathworks**维护的一个匿名FTP站点。FTP是文件传输协议，用来连接主机，从而将文件送到计算机和从计算机传出。匿名FTP站点允许用户和它们相连并传送文件，不要求用户在主机上有帐户和口令。

有几个计算机程序用用户图形界面进行FTP连接和传送文件。包括PC上的**WinFTP**，Macintosh上的**Fetch**，UNIX上的**ftptool**。如下面要讨论的，Web浏览器，如**Mosaic**，**Netscape**也用于访问这个站点。如果站点没有这些程序，就必须使用原始的面向命令行的**ftp**程序，因为原始**ftp**程序通常在UNIX工作站上可获得，PC机和Macintosh上也有。它的用法将在本节进行描述。

MATLAB FTP站点在**ftp.mathworks.com**，它包括用户提供的M文件、产品信息、含有常问问题(FAQs)、补片和诊断的文档。为了连接到站点，即**ftp**到站点，用 '**anonymous**' 名字或 '**ftp**' 登录，然后当要求口令时，输入用户的E-mail地址。

下面是一个简短的**ftp**会话样例。输入项以黑体表示。它连接到站点，列出可用的文件及目录，变换成ASCII模式以传送文本文件；检索文件**README**；关闭连接。

ftp ftp.mathworks.com
Connected to ftp.mathworks.com
220 ftp FTP server (Version wu-2.r(1) Thu April 14:25:23) ready
Name (ftp.mathworks.com:user):**anonymous**
331 Guest login ok， send your complete E-mail address as password.
Password: **user@host.maine.edu**
230
230Welcome to the Mathworks Library!
...
230 Guest lgoin ok， access restrications apply.
Remote system type is UNIX
Using bianry mode too transfer files
ftp> **dir**
200 PROT command suceessful
150 Opeing ASCII mdoe data connection for /bin/ls
226 Transfer complete.
total 27
-rx-r--r-- 1 admin daemon 488 Jun 24 1994 .welcome.msg
-rw-r--r-- 1 admin ftpusers 2172 Sep 28 1994 README
-rw-r--r-- 1 admin ftpusers 2626 Sep 28 1994 README.incoming
drwxr-xr-x 2 root daemon 512 Jun 16 1994 bin
drwxr-xr-x 2 toot daemon 512 Jun 28 1993 dev
drwxr-xr-x 2 toot daemon 512 Jun 7 1993 etc
drxwrwx-wx 38 admin 102 14336 Apr 21 17:23 incoming
lrxwrwxrwx 1 root daemon 3 Nov 6 1993 matlab->pub
drwxr-xr-x 12 admin ftpusers 512 Mar 27 15:17 pub
drwxr-xr-x 3 root daemon 512 Apr 23 1993 usr
ftp> **ascii**
200 Type set to A
ftp> **get README**

200 port command successful
150 Opening ASCII mdoe data connection for README (2172 bytes).
226 bytes received
ftp> **bye**
221 Goodbye

第1列用 ' d ' 列出的是目录，第1列有 ' - ' 是文件，第1列为 ' l ' 表示连接。它可以指向另外一个目录或其它某站点上的文件。目录列表提供的信息还包括拥有者、组、文件或目录大小、修改日期、文件或目录名称。

如果在连接时出现问题，也许是还没有访问到一个Internet名字服务器,该服务器将主机的名称翻译成IP地址。在这种情况下，试用 IP地址**144.212.100.10**来代替**ftp.mathwoeks.com**

ftp 144.21.100.10
connected to ftp.mathworks.com

fpt 程序支持大量的命令和选项。最有用的列在表23.1：

表23.1

	基本FTP命令
help	列出所有可用的**ftp**命令
help **command**	返回一个简单的命令描述
cd **/pub/contrib**	改变到远程站点上的**/pub/contrib**目录
lcd **localdir**	改变到用户的本地机上的目录
ascii	以ASCII文本模式传输文件。对自己机器上的文本文件，将行结束回车/换行变换为缺省值
binary	以二进制方式传输，不进行文本转换
get **remotefile**	从远程站点检索名为**remotefile**的文件
put **localfile**	将名为**locatefilede** 的文件传送到远程站点
dir	在远程站点列出详细的当前目录
ls	在远程站点列出当前目录，详细有限
bye，quit	退出ftp

MATLAB具有其它许多匿名FTP上都不具备的一个非常好的特性，它有能力即时地完成文件的归档和压缩。所支持的文件压缩格式有MIT的GNU中的**gzip** (**file.gz**)，标准Unix上的**compress** (**file.z**)以及在PC和Unix上的**zip** (**file.zip**)。文件的归档用标准的Unix **tar**格式(**dir.tar**)、Unix **shar**格式(**dir.sh**)和以及PC和Unix工作站的zip格式(**dir.zip**)，其中各格式对PC和Unix工作站都可用，且绝大部分也可以用于Macintosh计算机。

归档和压缩可以同时进行。如用以下命令，目录 **/pub/contrib/math**的全部内容可以在Unix**压缩的tar**格式文件中进行检索，

cd /pub/contrib
get math.tar.z

其他的组合也可以。命令

```
get  math.zip
get  math.sh
get  math.tar.gz
```

分别以**zip**文档、**shar**文档、**gzipped tar**文档来检索目录。不幸的是，对于Macintosh的用户，MATLAB站点不支持**stuffit** (**dir.sit**)文档或**BinHex** (**file.hqx**)编码。

表23.2是MATLAB站点的特性：

表23.2

MATLAB匿名FTP站点上重要的文件和目录

名称	类型	内容
/README	文本	新用户的文本信息
/README.incoming	文本	用户提交的文本信息
/incoming	目录	M文件用户意见投入箱的目录
/pub	目录	文件的主目录
/matlab	连接	连接到 **/pub**
/pub/INDEX	文本	有关目录内容的信息
/pub/NEWFILES	文本	站点上的新文件
/pub/ls-lr	文本	站点上所有文件的递归列表
/pub/ftphelp	文本	新**ftp**用户的入门指导
/pub/books	目录	与MATLAB基础有关的M文件
/pub/conference	目录	即将要召开的MATLAB会议的信息
/pub/contrib	目录	用户提供的M文件和MEX文件
/pub/doc	目录	文档，指导，帮助文件，FAQs，等等
/pub/mathworks	目录	MATLAB开发者提供的诊断和M文件
/pub/mosaic	目录	用于Unix的Mosaic的WWW浏览程序
/pub/pentium	目录	奔腾迷论文集
/pub/proceedings	目录	过去的MATLAB会议的论文集
/pub/product-info	目录	MATLAB，SIMULINK和工具箱的信息
/pub/tech-support	目录	技术支持和其他信息

如果有连接的麻烦或需要帮助，可以向**fptadmin@mathworks.com**站点的维护人员发E-mail.

23.3 全球广域网WWW

向Internet获取信息、图象、文件的最新、最容易、也是最灵活的方法是**World Wide Web**或简称为**WWW**或**Web**。它是根据请求向本地计算机提供超文本文档的站点服务器的集合。超文本文档可以与内置的图形结合在一起并连接到其它文件，只需用鼠标点击闪亮的文字或图形，就可以激活这些连接。然后，检索所连接的文档进行观察。如果愿意，也可以将文件存到自己的计算机上。有些连接可以有接到**Gopher**站点上（基于文本信息的服务器）以及全球的FTP站点。

如果有访问全面服务的Internet节点，就有这类MATLAB信息的图形界面供选择。**Mathworks**有一个WWW服务器，可以用Web的客户程序如**Mosaic**、**Netscape**或基于字符的*Lynx*来访问。用菜单项的**Open Location...**或**Open URL...**然后输入**http://www.mathworks.com**则可以连接到**Mathworks**的主页。主页是最高层的超文本文档，它包含了到其他的文档的连接。一旦连通，就可以将访问到的信息作为书签保存下来。以方便下一次的连接。

从**Mathworks**的主页，超文本连接可查询**Mathworks**公司和它的产品，在线拷贝通讯季刊，列出MATLAB的书籍（在写这本书时已超过100本）及相关M文件，职员要参加的有关商贸展览会的信息和最新的MATLAB会议论文集。还有一个常问问题及其关于MATLAB和SIMULINK解答的文档库。也可以阅读由MathWork技术支持人员提供的技术要点，包括内存管理、图形打印、MEX文件、工具箱以及如何将MATLAB与其他的软件集成的技术和技巧。

Mathworks主页也有到匿名FTP站点上的直接连接，可以浏览目录、读索引文件、并用浏览器检索文件。如果没有Web浏览器程序，在FTP站点的**/pub/mosaic**目录下可以获得许多工作站的**Mosaic**的版本；用于PC及Macintosh的**Mosaic**也可以通过匿名FTP的**ftp.ncsa.uiuc.edu**得到，而用于工作站和微机的**Netscape**可以通过**ftp.netscape.com**得到。必须明白，虽然**Mosaic**是免费 软件，但对免费使用**Netscape**，商业公司有一些限制。

23.4 MATLAB的自动电子邮件自动应答系统

对于那些不具有FTP功能，而有E-mail连接到Internet上的用户并没有排斥在外。**Mathworks** 拥有一个自动的用mail的文件服务，称为MATLIB，它从FTP站点用E-mail传输文件。向**matlab@mathworks.com**发一个E-mail，在消息主体中带有单向**help**就可以得到有关访问该服务器的更详尽的信息。MATLIB的mail文件服务是一个可以从消息主体中取得其指令信息的计算机程序，标题行被忽略掉。所以，消息句法正确十分重要。

在消息中可以用的许多指令是标准FTP命令的子集，包括：**cd**，**ls**，**dir**，**get**，**quit**。其他的命令用于指定编码、存挡、所用的压缩方式。其它还有指导**matlib**的程序，以回复不同的E-mail地址或限制回复文件的大小。

一个例子是**reply-to**命令，这是可选的。但如果用了，就可以使MATLIB服务器向E-mail地址发送由命令指定的文件。如果没有用，则向发送者的E-mail地址回复。例如：

```
reply-to user@host.maine.edu
```

则向**user@host.maine.edu**回答，而不是原来的E-mail地址。

二进制的文件不能用E-mail传送，所以在发送前要编码（转换成ASCII字符表示）。二进制文件按缺省是用**uuencode**发送。如果需要，也可以用**mime**编码来发送，也有用**compress**、

gzip、**zip**、**shar**、**tar**进行压缩或归档。例如：考虑发送到**matlib@mathworks.com**的一则消息，使用如下命令：

```
dir
cd /pub/contrib
get INDEX
get intergration.tar.z
get math.zip
cd games
get fifteens.m.gz
cd ..
get diffeq.sh
quit
```

这则脚本指示MATLIB首先检索目录，列出顶层目录。然后改到**/pub/contrib**目录，检索**INDEX**文件。然后作为受压缩的**tar**文件检索整个**/pub/contrib/integration/**目录，这是二进制文件，所以在传送之前先要编码**uuencoded**。接下来对整个**/pub/contrib/math**目录的内容，检索**uuencoded zip**文档，现在改到**games**目录，用**gzip**压缩检索M文件**fifteens.m**，最后改到下一个较高层目录**/pub/contrib**，作为**shar**文档，检索整个**diffeq**目录，然后退出。

回复限制在100K字节以内，所以较长的文件或文档要以多个E-mail的方式传送。如果E-mail大小有限制，也可以用**size**命令进一步限制消息的大小。**help**消息中有更详尽的说明。

23.5 MathWorks MATLAB文摘

MATLAB文摘是电子通报月刊，通过E-mail向订户发送。这个通报包括MATLAB的新闻，MATLAB的开发者和用户提供的文章，提示以及对用户问题的回答。想要订阅文摘，则要向**subscribe@mathworks.com**发送E-mail，申请加入邮寄名单，或在MATHLAB提示行中键入**>> subscribe**。**subscribe**命令提出问题，利用回答产生打印的申请表，该表邮寄或传真到***Mathworks***。Unix提供了自动的对申请发送E-mail的功能，而不是打印出来。在MATLAB FTP的**/pub/doc/tmw-digest**目录中，有过期的通报。

23.6 MATLAB通报

使用MATLAB任何用户，都可得到季度出版物MATLAB **News & Notes**。如果你是订户，可以收到文摘及季度通报以及免费的技术支持。不需注册就可以成为一个订户，这是免费服务的。可利用WWW站点上的表格，键入**>>subscribe**，或向**subscribe@mathworks.com**发送E-mail提出要求。通报通常有新闻、提示、克利夫·莫勒(Cleve Moler)和其他人的文章以及大事表。

23.7 MathWorks 电子邮件及网络地址

表23.3

Mathworks公司的E-mail地址	
suport@mathworks.com	技术支持
bugs@mathworks.com	错误报导
doc@mathworks.com	文档错误报导
suggest@mathworks.com	产品升级建议
service@amthworks.com	订购情况，延长许可，许可码
subscribe@mathworks.com	订户信息
info@mathworks.com	销售，价格和一般信息
micro-updates@mathworks.com	PC及Macintosh的升级
matlib@mathworks.com	mail文件服务器
digest@mathworks.com	MATLAB文摘
ftpadmin@mathworks.com	**Mathworks** FTP站点
webmaster@mathworks.com	**Mathworks**的WWW维护人员

表23.4

MATLAB的网络资源	
www.mathworks.com	WWW站点
ftp.mathworks.com	匿名FTP站点
144.212.100.10	WWW及FTP的Internet地址
novell.felk.cvut.cz	**ftp.mathworks.com**的镜象
192.108.154.33	Internet地址的镜象

表23.5

其他的网络资源	
mm@eeece.maine.edu	向作者提出有关精通MATLAB及精通MATLAB工具箱的问题和评论的E-mail地址
ftp.ncsa.uiuc.edu	**Telnet**和FTP的PC及Macintosh版本，PC、Macintosh和Unix的*Mosaic*
ftp2.netscape.com	PC Mac. Unix的**Netscape**的浏览器
sumex-aim.stanford.edu	**Mosaic**、**Telnet(FTP)**、**Fetch**、**gzip**、**zip**、**shar**、**compress**的Mac版本
sics.se	在**sumex-aim.stanford.edu**的文件镜象
gatekeeper.dec.com	大的匿名FTP站点
sunsite.unc.edu	大的匿名FTP站点
ftp.wustl.edu	大的匿名FTP站点
nic.funet.fi	大的匿名FTP站点
ftp.luth.se	大的匿名FTP站点
ftp.cdrom.com	大的匿名FTP站点
ftp.nws.edu.au	大的匿名FTP站点