MATLAB 数学工具软件
实例简明教程

王 正 盛　　编写

南 京 航 空 航 天 大 学

第一章　MATLAB 简介

MALAB 译于矩阵实验室（MATrix LABoratory），是用来提供通往 LINPACK 和 EISPACK 矩阵软件包接口的。 后来，它渐渐发展成了通用科技计算、图视交互系统和程序语言。

MATLAB 的基本数据单位是矩阵。它的指令表达与数学、工程中常用的习惯形式十分相似。比如，矩阵方程 Ax=b，在 MATLAB 中被写成 A*x=b。而若要通过 A,b 求 x，那么只要写 x=A\b 即可，完全不需要对矩阵的乘法和求逆进行编程。因此，用 MATLAB 解算问题要比用 C、Fortran 等语言简捷得多。

MATLAB 发展到现在，已经成为一个系列产品： MATLAB“主包”和各种可选的 toolbox“工具包”。主包中有数百个核心内部函数。迄今所有的三十几个工具包又可分为两类：功能性工具包和学科性工具包。功能性工具包主要用来扩充 MATLAB 的符号计算功能、图视建模仿真功能、文字处理功能以及硬件实时交互功能。这种功能性工具包用于多种学科。而学科性工具包是专业性比较强的，如控制工具包（Control Toolbox）、信号处理工具包(Signal Processing Toolbox) 、通信工具包(Communication Toolbox)等都属此类。开放性也许是 MATLAB 最重要、最受人欢迎的特点。除内部函数外，所有 MATLAB 主包文件和各工具包文件都是可读可改的源文件，用户可通过对源文件的修改或加入自己编写文件去构成新的专用工具包。

MATLAB 已经受了用户的多年考验。在欧美发达国家，MATLAB 已经成为应用线性代数、自动控制理论、数理统计、数字信号处理、时间序列分析、动态系统仿真等高级课程的基本教学工具；成为攻读学位的大学生、硕士生、博士生必须掌握的基本技能。在设计研究单位和工业部门，MATLAB 被广泛地用于研究和解决各种具体工程问题。

第二章　MATLAB 入门

2.1 工作窗和指令行的操作

除了通过菜单选项对工作窗进行控制外， MATLAB 还提供了许多通过键盘输入的控制指令。如下表

MATLAB 工作窗中的部分通用指令

quit	关闭和退出 MATLAB
clc	擦除 MATLAB　工作窗中的所有显示内容
clf	擦除 MATLAB　的当前图形窗中的图形
clear	清除内存中的变量和函数
pack	收集内存碎片以扩大内存空间
dir	列出指定目录下的文件和子目录清单
cd	改变当前工作子目录
disp	（在运行中）显示变量和文字内容
type	显示所有指定文件的全部内容
echo	控制运行文件指令是否显示的开关

hold	控制当前图形窗对象是否被刷新

启动 MATLAB 后，就可以利用它工作了。由于 MATLAB 是一种交互式语言，随时输入指令、即时给出运算结果是它的主要工作方式（当然更可以编制程序，详见第七章）。

比如要计算 $\frac{2\sin(0.3\pi)}{1+\sqrt{5}}$ 的值，只要在光标位置处键入：

```
2*sin(0.3*pi)/(1+sqrt(5))
```

然后按[Enter]键,该指令便被执行并给出结果：

```
ans =    0.5000
```

下面介绍控制光标、对指令进行编辑的一些常用操作键。

常用操作键

键　名	作　用	键　名	作　用
↑	前寻式调回已输入过的指令行	Home	使光标移到当前行的首端
↓	后寻式调回已输入过的指令行	End	使光标移到当前行的尾端
←	在 当前行中左移光标	Delete	删除光标右表边的字符
→	在 当前行中右移光标	Backspace	删除光标左表边的字符
PageUp	前寻式翻阅当前窗中的内容	Esc	清楚当前行的全部内容
PageDown	后寻式翻阅当前窗中的内容		

2.2 简单矩阵的输入

在 MATLAB 中，矩阵输入的方法有多种，此处只简单介绍矩阵的直接输入法，详细介绍见第三章。在 MATLAB 中不必对矩阵维数做任何说明，存储将自动配置。在直接输入矩阵时，矩阵元素用空格或逗号分隔，矩阵行用“；”隔离。整个矩阵放在方括号“[ ]”里。

[例 1]

```
A=[1,2,3;4,5,6;7,8,9;10,11,12]
A =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

说明：指令执行后，矩阵 A 被保存在 MATLAB 的工作间（Workspace）中，以备后用。如果用户不用 clear 指令清除它，或对它重新定义，该矩阵会一直保存在工作间中，直到本 MATLAB 指令窗被关闭为止。

[例 2]矩阵分行输入

```
A=[1 2 3 4
     5 6 7 8
     0 1 2 3]
A =  1     2     3     4
     5     6     7     8
```

```
     0     1     2     3
```

[例 3]矩阵元素输入

```
B(1,2)=3;B(4,4)=6;B(4,2)=11
B =  0     3     0     0
     0     0     0     0
     0     0     0     0
     0    11     0     6
```

2.3 语句与变量

MATLAB 采用表达式语句。用户输入语句由 MATLAB 系统结实运行。MATLAB 语句有两种常见的形式：（1）表达式；（2）变量=表达式

说明：（1）表达式由算符、函数、变量名和数字构成。

（2）在第一种形式中，表达式被执行后产生的矩阵，将被自动赋给名为“ans”的变量，并 显示在屏幕上，“ans”是一个缺省变量名，它会被以后类似的操作刷新。

（3）在第二种形式中，等号右边的表达式是被演绎后产生的矩阵，将被赋给等号左边的变量存入内存，并显示在屏幕上。

（4） 书写表达式时，运算符号“=”、“+”、“—”以及“*”等两侧允许有空格，以增加可读性。但在复数或符号表达式中，要尽量避免“装饰性”空格，以防出错。

（5）变量名、函数名以一个字母打头，后面最多可接 19 个字母或数字。**注意**：MATLAB 是区分字母的大小写的。

[例1] 表达式的计算结果

```
2001/81
ans =   24.7037
```

[例2]运算结果的赋值。

```
s=1-1/2+1/3-1/4+1/5-1/6+1/7-1/8;
```

说明：结尾的分号“；”作用是：指令执行结果将不会显示在屏幕上，但变量 s 仍将驻留在内存中。如想看 s 的值，只要键入：

```
s
s =    0.6345
```

2.4 Who、Whos 和永久变量

Who 和 Whos 这两个指令的作用都是列出在 MATLAB 工作间中已经驻留的变量名清单。不过，Whos 在给出变量名的同时，还给出它们的维数及性质。

[例1]用 who 检查内存变量

```
who
Your variables are:
s
```

[例2]用 whos 检查驻留变量的详细情况。

```
 whos
  Name      Size            Bytes  Class
  s         1x1                 8  double array
Grand total is 1 elements using 8 bytes
```

在 MATLAB 工作内存中，还驻留几个由系统本身在启动时定义的变量（如下表），称为“永久变量”（Permanent variables），或称为“预定义变量”（Predefined variables）。

系统预定义的变量

eps	计算机的最小正数，在 pc 机上，它等于 2^{-52}
pi	圆周率π 的近似值 3.14159265358979
inf 或 Inf	无穷大
NaN	不定量
i,j	虚数单位，定义 $i = j = \sqrt{-1}$
flops	浮点运算次数，用于统计计算量

说明：（1）它们是在 MATLAB 启动时自定义的。

（2）它们不会被“清除内存变量”指令 clear 所清除。

（3）他们可以重新定义为其他值，但用 clear 可清除重定义值，恢复预定义值。

[例1] 无穷大

```
s=1/0
Warning: Divide by zero.
s =   Inf
```

无穷大

```
a=Inf/inf
a =
   NaN
```

2.5 数与表达式

MATLAB 的数值采用习惯的十进制表示，可以带小数点或负号。如下是合法的：

3　　-99　　0.0013　　9.2445154　　1.2434e-6　　4.673e33

在采用 IEEE 浮点算法的计算机上，数值的相对精度是 eps，即大约保持 16 位有效数字，数值范围大致为 1×10^{-308} ~ 1×10^{308} 。

表达式由下列算符构成，并按习惯的优先次序进行运算。

\+ 加法　— 减法　* 乘法　/ 右除　\ 左除　^ 乘方

注意：设置两种除法是为了方便矩阵的运算，对标量而言两者作用相同。

[例 1]

```
x=2*pi/3+2^3/5-0.3e-3

x =
    3.6941
```

2.6 复数和复矩阵

MATLAB 认识复数，并定义 i 和 j 作为虚数单位。矩阵元素允许是复数、复变量和由它们组成的表达式。

[例 1]

```
z1=3+4*i,z2=2*exp(i*pi/6)
z=z1*z2
```

```
z1 =
   3.0000 + 4.0000i
z2 =
   1.7321 + 1.0000i
z =
   1.1962 + 9.9282i
```

[例 2]

```
A=[1,3;2,4]-i*[5,8;6,9]
B=[1+5*i,2+6*i;3+8*i,4+9*i]
C=A*B
```

```
A =
   1.0000 - 5.0000i   3.0000 - 8.0000i
   2.0000 - 6.0000i   4.0000 - 9.0000i
B =
   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 8.0000i   4.0000 + 9.0000i
C =
  1.0e+002 *
   0.9900           1.1600 - 0.0900i
   1.1600 + 0.0900i   1.3700
```

2.7 函数

MATLAB 的强大功能可函数中略见一斑，本质上讲分为三类：

（1） 内部函数

（2） 系统附带各种工具包中的 M 文件所提供的大量函数

（3） 用户自己增加的函数，这一特点是其他许多软件平台无法比拟的。

MATLAB 提供的通用数理类函数包括：

- 基本数学函数
- 特殊函数
- 基本矩阵函数
- 特殊矩阵函数
- 矩阵分解和分析函数
- 数据分析函数
- 微分方程求解
- 多项式函数
- 非线性方程及其优化函数
- 数值积分函数
- 信号处理函数

[例]

```
z=1233.344
x=sqrt(log(z))
```

```
z =
```

```
    1.2333440000000000e+003
x =
   2.66786140168028
```

2.8 显示格式

在缺省的状态下，MATLAB 以短格式（short 格式）显示计算结果。可以用 MATLAB 命令窗口中 format 指令来改变数字的显示格式。由于 MATLAB 以双精度执行所有运算，显示格式的设置仅影响矩阵的显示，不影响矩阵的计算与存储。

如果矩阵的所有元素都是整数，则矩阵以不带小数点的格式显示。如果有一个元素不是整数，则有几种输出格式。默认格式为 short 格式，只显示 5 位有效数字，其他的显示格式可显示更多的有效数字，还可用科学表示法。

[例]

```
x=[4/3  1.2345e-6]
默认（short）格式
x =
    1.3333    0.0000
format short e  （短格式科学表示）
x

x =
  1.3333e+000  1.2345e-006
format long  （长格式）
x

x =
   1.33333333333333   0.00000123450000
format long e  （长格式科学表示）
x

x =
    1.333333333333333e+000    1.234500000000000e-006
format bank   （银行格式）
x
x =         1.33         0.00
format hex   （十六进制格式）
x
x =   3ff5555555555555   3eb4b6231abfd271
format +  （+格式—用于显示大矩阵的紧凑格式，+、—、空格分别表示正数、负数和零。
x
x =++
```

另外，还有一种命令为 format compact(紧凑格式)，它消去了矩阵之间的间隔行，这样可在一屏中显示更多的信息。

2.9 变量的存储与调用

quit 和 exit 指令都可退出 MATLAB 。结束 MATLAB 任务会删除工作间中的变

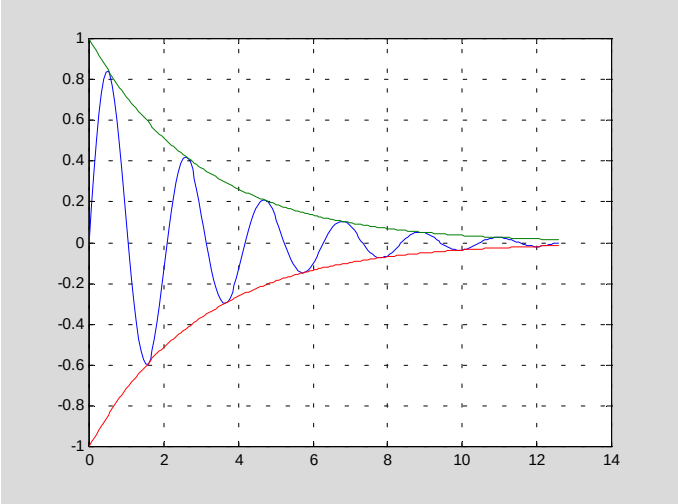
量。在退出前，可以保存工作空间，以备再次调出使用这些变量。
保存的指令格式：
（1） save 工作间中的所有变量保存在磁盘上名为 matlab.mat 的文件中
（2） save [文件名] [变量名] 将指定的变量保存在指定文件中。如：
save temp x y z 把 x,y,z 这三个变量保存在文件 temp.mat 中。
在下次加载 MATLAB 时，可以利用 load 指令将保存在文件中的变量恢复到工作间中，其格式有：
（1）load 将保存在 matlab.mat 中的变量装入到 MATLAB 工作间中。
（2）load [文件名] [变量名] 从指定的文件中将指定的变量装入 MATLAB 工作间。如：
load temp x 从文件 temp.mat 中只将变量 x 装入到 MATLAB 工作间中。

2.10 图形

图形是 MATLAB 的主要特色之一。MATLAB 图形指令具有自然、简洁、灵活及易扩充的特点。MATLAB 的指令很多，这里仅介绍几个简单的绘图指令，详见第六章。

[例1] 作多条曲线

```
t=0:pi/50:4*pi;
y0=exp(-t/3);
y=exp(-t/3).*sin(3*t);
plot(t,y,t,y0,t,-y0)
grid
```



[例2] 三维曲面

```
x=-8:0.5:8;
y=x';
X=ones(size(y))*x;
Y=y*ones(size(x));
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(Z);
colormap([1,0,0])
```

2.11 lp 指令、lookfor 指令及其他帮助指令

MATLAB 的在线帮助系统相当完备，就查询系统的调用方式而言，可分为两种：

（1） 从 MATLAB 指令窗的 help 菜单选项中寻求帮助。此与一般 windows 的求助方法一样。

（2） 在 MATLAB 指令窗中，直接键入求助指令。

(i)help 不带任何参数，显示出 MATLAB 的目录项，产生清单信息：

```
help

HELP topics:
matlab\general       -  General purpose commands.
matlab\ops          -  Operators and special characters.
matlab\lang         -  Programming language constructs.
matlab\elmat            -   Elementary matrices and matrix
manipulation.
matlab\elfun        -  Elementary math functions.
matlab\specfun      -  Specialized math functions.
matlab\matfun       -  Matrix functions - numerical linear
algebra.
matlab\datafun            -    Data  analysis  and  Fourier
transforms.
matlab\polyfun      -  Interpolation and polynomials.
matlab\funfun       -  Function functions and ODE solvers.
matlab\sparfun      -  Sparse matrices.
matlab\graph2d      -  Two dimensional graphs.
matlab\graph3d      -  Three dimensional graphs.
matlab\specgraph    -  Specialized graphs.
matlab\graphics     -  Handle Graphics.
matlab\uitools      -  Graphical user interface tools.
matlab\strfun       -  Character strings.
matlab\iofun        -  File input/output.
matlab\timefun      -  Time and dates.
matlab\datatypes    -  Data types and structures.
matlab\winfun        -  Windows Operating System Interface
Files (DDE/ActiveX)
matlab\demos        -  Examples and demonstrations.
toolbox\runtime      -   MATLAB Runtime Server Development
Kit
rtw\windows         -  Real Time Windows Target.
```

```
daq\daq            -  Data Acquisition Toolbox
daq\daqdemos          -  Data Acquisition Toolbox - Data
Acquisition Demos.
toolbox\dials       -  Dials & Gauges Blockset
toolbox\rptgenext    -  Simulink Report Generator
toolbox\rptgen      -  MATLAB Report Generator
database\database    -  Database Toolbox.
database\dbdemos      -    Database Toolbox Demonstration
Functions.
powersys\powerdemo   -  Power System Blockset Demos.
powersys\powersys    -  Power System Blockset
toolbox\compiler       -   MATLAB Compiler (and Compiler
1.2.1)
comm\comm          -  Communications Toolbox.
comm\commmasks       -  Communications Toolbox mask helper
functions.
comm\commsfun       -  Communications Toolbox S-functions.
comm\commsim          -  Communications Toolbox Simulink
files.
toolbox\symbolic     -  Symbolic Math Toolbox.
nag\nag            -  NAG Foundation Toolbox - Numerical &
Statistical Library
nag\examples         -  NAG Foundation Toolbox - Numerical
& Statistical Library
map\map            -  Mapping Toolbox
map\mapdisp          -  Mapping Toolbox Map Definition and
Display.
map\mapproj         -  Mapping Toolbox Projections.
wavelet\wavelet      -  Wavelet Toolbox.
wavelet\wavedemo     -  Wavelet Toolbox Demos.
toolbox\pde             -   Partial Differential Equation
Toolbox.
finance\finance      -  Financial Toolbox.
finance\calendar        -    Financial  Toolbox  calendar
functions.
finance\findemos      -   Financial Toolbox demonstration
functions.
lmi\lmictrl             -   LMI  Control  Toolbox:  Control
Applications
lmi\lmilab          -  LMI Control Toolbox
qft\qft            -  QFT Control Design Toolbox.
qft\qftdemos         -  QFT Control Design Toolbox Demos
toolbox\fixpoint     -  Fixed-Point Blockset
fixpoint\fxpdemos    -  Fixed-Point Blockset Demos
fixpoint\obsolete    -  Obsolete Fixed-Point Blockset
dspblks\dspblks      -  DSP Blockset.
dspblks\dspmex       -  (No table of contents file)
dspblks\dspdemos       -   DSP  Blockset  demonstrations  and
examples.
dspblks\dspmasks     -  DSP Blockset mask helper functions.
fuzzy\fuzzy         -  Fuzzy Logic Toolbox.
fuzzy\fuzdemos       -  Fuzzy Logic Toolbox Demos.
mpc\mpccmds         -  Model Predictive Control Toolbox.
mpc\mpcdemos        -  Model Predictive Control Toolbox
```

```
fdident\fdident       -  Frequency Domain Identification
Toolbox.
fdident\fddemos       -  Demonstrations for the FDIDENT
Toolbox
hosa\hosa             -  Higher-Order Spectral Analysis
Toolbox.
hosa\hosademo         -  Higher-Order Spectral Analysis
Toolbox - Demo suite
toolbox\stats       -  Statistics Toolbox.
toolbox\ncd         -  Nonlinear Control Design Blockset
images\images       -  Image Processing Toolbox.
images\imdemos      -  Image Processing Toolbox --- demos
and sample images
nnet\nnet           -  Neural Network Toolbox.
nnet\nndemos        -  Neural Network Demonstrations.
nnet\nnutils        -  (No table of contents file)
nnet\nnobsolete     -  (No table of contents file)
mutools\commands    -  Mu-Analysis and Synthesis Toolbox.
mutools\subs        -  Mu-Analysis and Synthesis Toolbox.
signal\signal       -  Signal Processing Toolbox.
signal\siggui       -  Signal Processing Toolbox GUI
signal\sigdemos          -   Signal  Processing  Toolbox
Demonstrations
toolbox\splines     -  Spline Toolbox.
toolbox\optim       -  Optimization Toolbox.
toolbox\robust      -  Robust Control Toolbox.
toolbox\ident       -  System Identification Toolbox.
toolbox\control     -  Control System Toolbox.
control\ctrlguis      -   Control  System  Toolbox  --  GUI
support functions.
control\obsolete    -  Control System Toolbox -- obsolete
commands.
toolbox\rtw         -  Real-Time Workshop
rtw\rtwdemos        -  (No table of contents file)
stateflow\sfdemos     -   Stateflow  demonstrations  and
samples.
toolbox\sb2sl       -  SystemBuild to Simulink Translator
stateflow\stateflow  -  Stateflow
simulink\simulink   -  Simulink
simulink\blocks     -  Simulink block library.
simulink\simdemos     -   Simulink  3  demonstrations  and
samples.
simulink\dee        -  Differential Equation Editor
MATLAB53\work       -  (No table of contents file)
toolbox\local       -  Preferences.

For more help on directory/topic, type "help topic".
```

(ii) help　目录名——显示指定目录中的所有命令及其函数。

help lang ——将列出与 MATLAB 编程语言的所有命令及其函数。

help matfun——将列出与数值线性代数有关的所有矩阵函数。

help elfun——列出所有基本函数。

（iii） help 命令名/函数名/符号——显示指定的命令名/函数名/符号的详细信息。

[例]

help eig——显示计算矩阵特征值和特征向量的函数 eig 的说明。

```
help eig
 EIG   Eigenvalues and eigenvectors.
    E = EIG(X) is a vector containing the eigenvalues of a
square
    matrix X.

    [V,D] = EIG(X) produces a diagonal matrix D of
eigenvalues and a
    full matrix V whose columns are the corresponding
eigenvectors so
    that X*V = V*D.

    [V,D] = EIG(X,'nobalance') performs the computation
with balancing
    disabled, which sometimes gives more accurate results
for certain
    problems with unusual scaling.

    E = EIG(A,B) is a vector containing the generalized
eigenvalues
    of square matrices A and B.

    [V,D] = EIG(A,B) produces a diagonal matrix D of
generalized
    eigenvalues and a full matrix V whose columns are the
    corresponding eigenvectors so that A*V = B*V*D.

    See also CONDEIG, EIGS.

 Overloaded methods
    help sym/eig.m
    help lti/eig.m
```

注意：help 的工作机理是是把指定名字的那个 M 文件的第一段注释内容显示出来，以构成自己文件的再线求助。

lookfor 指令可以根据用户提供的完整或不完整的关键词，去搜索出一组与之有关的指令。

[例1]查找有关积分的指令

```
lookfor integral

[例2] ELLIPKE Complete elliptic integral.
[例3] EXPINT Exponential integral function.
[例4] DBLQUAD Numerically evaluate double integral.
[例5] INNERLP Used with DBLQUAD to evaluate inner loop of
    integral.
[例6] QUAD  Numerically evaluate integral, low order
    method.
[例7] QUAD8  Numerically evaluate integral, higher order
    method.
```

```
[例8] COSINT Cosine integral function.
[例9] SININT Sine integral function.
[例10]ASSEMA Assembles area integral contributions in a
    PDE problem.
[例11]COSINT Cosine integral function.
[例12]FOURIER Fourier integral transform.
[例13]IFOURIER Inverse Fourier integral transform.
[例14]SININT Sine integral function.
BLKPIDCON The output of the block is the sum of
proportional, integral and
```

[例 2]查找有关傅里叶变换的指令

```
lookfor fourier

FFT Discrete Fourier transform.
FFT2 Two-dimensional discrete Fourier Transform.
FFTN N-dimensional discrete Fourier Transform.
IFFT Inverse discrete Fourier transform.
IFFT2 Two-dimensional inverse discrete Fourier transform.
IFFTN N-dimensional inverse discrete Fourier transform.
XFOURIER Graphics demo of Fourier series expansion.
INSTDFFT Inverse non-standard 1-D fast Fourier transform.
NSTDFFT Non-standard 1-D fast Fourier transform.
EXPFOU Write data to a Fourier vector or a (maybe
existing) file (for ELIS).
IMPFOU Read complex amplitudes from a Fourier vector or
file (used by ELIS).
MODIFYFV Modify Fourier (maybe also variance) data by
given transfer function.
SIMFOU Generate simulated Fourier amplitudes.
PLOTFOU Plot contents of Fourier files
DFTMTX Discrete Fourier transform matrix.
FOURIER Fourier integral transform.
IFOURIER Inverse Fourier integral transform.
```

注意： lookfor 指令的机制：对 MATLAB 中的每个 M 文件注释区的第一行进行扫描，一旦发现包含要查询的字符串就显示出来。提示：用户也可利用此机理建立自己文件的在线帮助。

其他帮助指令

exist	检查指定名字的变量或函数文件的存在性
what	按扩展名分类列出（在搜索路径中）指定目录上的文件名
which	列出指定名字文件所在的目录

2.12 用户目录的建立和搜索路径

为了保护 MATLAB 目录结构的严整，为了用户自己用 MATLAB 所创建、修改的 M 文件和其他文件的方便，用户应建立自己的工作目录。MATLAB 启动后的默认目录是 C:\MATLAB\BIN，若不建子目录则 MATLAB 环境产生的数据文件就登陆在这个缺省目录上。建立工作目录两种方法：（1）在 DOS 环境中建立 ;(2)在 windows 环境下建立。

MATLAB 只能在启动时（由 mathabrc.m）设定的路径上搜索，不能与原定路径以外的其他目录交换信息。可用以下三种方法扩充：

(1）在 MATLAB 指令窗口中键入：　CD　C:\MYDIR
（2）利用 path 指令扩展搜索路径：　path(path, 'c:\mydir')
（3）在 MATLAB 环境下，键入：pathtool　或者在 MATLAB 指令窗口菜单上 File 中的 Set path 项设置。

第三章　MATLAB 的数值计算功能

3.1 数值矩阵的创建、保存和数据格式

3.1.1创建矩阵的直接输入法

前面已述，此出不赘述。

[例]

```
x=14 ;y=4.32;
A=[x,2*x-y,0;
sin(pi/4),3*y+x,sqrt(y)]
```

```
A =
   14.0000   23.6800         0
    0.7071   26.9600    2.0785
```

3.1.2利用 MATLAB 函数和语句创建数值矩阵

[例 1] 利用指令 reshape 创建数值矩阵

```
av=1:12     %产生 12 个元素的行向量 av（以%开头的是注释行）
bm=reshape(av,3,4)      %利用向量 av 创建3×4矩阵 bm
```

```
av =
     1     2     3     4     5     6     7     8     9    10    11
12
bm =
     1     4     7    10
     2     5     8    11
     3     6     9    12
```

[例 2] 利用指令 diag 产生对角阵

```
ar=rand(4,4) %产生4×4的“0-1均匀分布随即矩阵”ar
d=diag(ar)   %用矩阵的主对角线元素形成向量 d
D=diag(d)    %用向量 d 构成对角矩阵 D
```

```
ar =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
d =
    0.9501
    0.7621
    0.6154
    0.4057
D =
    0.9501         0         0         0
         0    0.7621         0         0
```

```
     0         0    0.6154         0
     0         0         0    0.4057
```

1.0.0利用 M 文件创建和保存矩阵

本节方法既适用于数值矩阵，又适用于符号矩阵。

[例1]创建和保存矩阵 AM 的 matrix.m 文件生成过程。

步骤 1：使用 DOS 的编辑器（edit）,Windows 的书写器(write)、记事本（notepad）或其他字处理软件（如 Word 等）编辑如下：AM=[1 2 3;3 4 5]

步骤 2：把此内容以纯文本方式（ASCII）保存在用户自己的目录下名为 matrix.m 的文件中；

步骤 3：在 MATLAB 指令窗中，只要键入 matrix，矩阵 AM 就会自动生成于 MATLAB 工作内存中（即产生一个名为 AM 的变量），供显示和调用。

3.1.4通过 MAT 文件保存和获取矩阵

MAT 文件是 MATLAB 保存数据的一种标准格式二进制文件。MAT 文件的生成和调用由指令 save 和 load 进行。

[例1] 把矩阵 AR 保存到文件大他 data.mat

步骤 1：在矩阵 AR 存在于 MATLAB 内存空间的前提下，键入：

save data AR

步骤 2：在下次进入 MATLAB 后，需要矩阵 AR 时，键入如下边可将 data.mat 中的内容读入 MATLAB 内存空间：

load data

说明：MATLAB 默认扩展名为.mat，默认路径为 matlab\bin 子目录。用户如把 data.mat 登陆在指定目录可用如下命令保存或调入：

save c:\mydir\data AR

load c:\mydir\data AR

3.1.5利用外部数据文件装入到指定矩阵

假如磁盘中已有名为 c:\mydir\data.dat 的 ASCII 数据文件，利用指令 load c:\mydir\data.dat 可在 MATLAB 工作空间产生一个名为 data 的矩阵（即变量）。当然，也可以用指令“fopen"、"fread"及其他 MATLAB 底层数据输入输出（I/O）指令实现（可查看帮助，如 help fopen） 。

3.2 矩阵的标识

矩阵的元素、子矩阵可以通过标识、向量、冒号的标识来援引和赋值。

[例 1]

```
b=[1 2 3 4 5; 6 7 8 9 10 ;11 12 13 14 15]
b23=b(2,3)
b1=b(1:2,[1 3 5])
b2=b([3 1],:)
 b([1 3],[2 4])=zeros(2)
```

```
b =
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
b23 =
     8
```

```
b1 =
     1     3     5
     6     8    10
b2 =
    11    12    13    14    15
     1     2     3     4     5
b =
     1     0     3     0     5
     6     7     8     9    10
    11     0    13     0    15
```

[例 2]

```
x=[1 2 3 4 5]     %产生1×5向量
x =
      1       2       3       4       5
l=x<=3     %标出小于等于 3 的元素的位置
l =
      1       1       1       0       0
x=x(l)     %获得元素不超过 3 的子向量
x =
      1       2       3
```

3.3 矩阵运算和数组运算

矩阵运算和数组运算是 Matlab 的数值运算中的两大类运算。矩阵运算是按矩阵运算法则进行的运算；数组运算无论是何种运算操作都是对元素逐个进行。

矩阵运算和数组运算指令对照汇总

矩阵运算指令	指令含义	数组运算指令	指令含义
A'	矩阵转置	A.+B	对应元素相加
A+B	矩阵相加	A.-B	对应元素相减
A-B	矩阵相减	A.*B	同维数组对应元素相乘
s+B	标量加矩阵	s.*A	A 的每个元素乘 s
s-B,B-s	标量矩阵相减	A./B	A 的元素被 B 的对应元素除
A*B	矩阵相乘	B.\A	同上
A/B	A 右除 B	s./B, B.\s	s 分别被 B 的元素除
B\A	A 左除 B	A.^n	A 的每个元素自乘 n 次
inv(A)	矩阵求逆	log(A)	对 A 的每个元素求对数
A^n	矩阵的 n 次幂	sqrt(A)	对 A 的每个元素求平方根
		f(A)	求 A 的各个元素的函数值

例：

```
a=[1 2 3; 4 5 6; 7 8 9];b=[1 2 3; 3 2 1;1 4 5];
c=[1 1 1;2 3 1;1 0 2];
d=a*c^2+b
d =
```

```
 32    31    36
 82    79    82
128   129   134
```

3.4 矩阵函数和数组函数

3.4.1基本数组函数

数组函数是对各个元素的函数设计的。

f(.)基本函数表

函数名称	功 能	函数名称	功 能
sin	正弦	acosh	反双曲余弦
cos	余弦	atanh	反双曲正切
tan	正切	acoth	反双曲余切
cot	余切	asech	反双曲正割
sec	正割	acsch	反双曲余割
csc	余割	fix	朝零方向取整
asin	反正弦	ceil	朝正无穷大方向取整
acos	反余弦	floor	朝负无穷大方向取整
atan	反正切	round	四舍五入到整数
atan2	四象反正切	rem	除后取余数
acot	反余切	sign	符号函数
asec	反正割	abs	绝对值
acsc	反余割	angle	复数相角
sinh	双曲正弦	imag	复数虚部
cosh	双曲余弦	real	复数实部
tanh	双曲正切	conj	复数共轭
coth	双曲余切	log10	常用对数
sech	双曲正割	log	自然对数
csch	双曲余割	exp	指数
asinh	反双曲正弦	aqrt	平方根

f(.)特殊函数表

函数名称	功 能	函数名称	功 能
bessel	第一、第二类 Bessel 函数	erf	误差函数
beta	Beta 函数	eerfinv	逆误差函数
gamma	Gamma 函数	ellipk	第一、第二类全椭圆积分
rat	有理近似	ellipj	Jacobi 椭圆函数

3.1.2基本矩阵函数

基本矩阵函数指令

函数指令	指令含义	函数指令	指令含义
cond(A)	矩阵的条件数（最大奇异值除以最小奇异值）	svd(A)	矩阵的奇异值分解
det(A)	方阵的行列式	trace(A)	矩阵的迹
dot(A,B)	矩阵的点积	expm(A)	矩阵指数e^A
eig(A)	矩阵的特征值	expm1(A)	用 Pade 近似求e^A
norm(A,1)	矩阵 1-范数	expm2(A)	用 Taylor 级数近似求e^A，精度稍差，但对任何方阵适用
norm(A)	矩阵的 2-范数	expm3(A)	用矩阵分解求e^A，仅当独立调整向量数目等于秩时适用
norm(A,inf)	矩阵的无穷范数	logm(A)	矩阵对数 ln(A)
norm(A,'fro')	矩阵的 f-范数（全部奇异值平方和的正平方根）	sqrtm(A)	平方根矩阵$A^{\frac{1}{2}}$
rank(A)	矩阵的秩（非零奇异值的个数）		
rcond(A)	矩阵的倒条件数	funm(A,'fn')	A 阵的一般矩阵函数

例：注意观察奇异值与矩阵各性质的关系

```
a=magic(5);
s=svd(a)'
d=det(a),t=trace(a),rk=rank(a),c=cond(a)
n1=norm(a,1),n2=norm(a),ninf=norm(a,inf),nf=norm(a,'fro')
```

```
s =
   65.0000   22.5471   21.6874   13.4036   11.9008
d =
    5070000
t =
    65
rk =
     5
c =
    5.4618
n1 =
    65
```

```
n2 =
   65.0000
ninf =
    65
nf =
   74.3303
```

3.5 线性方程组的直接解法

线性方程组 Ax=b，A 是 $n \times m$ 的系数矩阵

1) 当 n=m 且非奇异时，此方程称为“恰定”方程（Properly Determined Equation）

2) 当 n>m 时，此方程称为“超定”方程（Overdetermined Equation）

3) 当 n<m 时，此方程称为“欠定”方程（Underdetermined Equation）

3.5.1 矩阵逆和除法解恰定方程组

（1）采用求逆运算：x=inv(A)*b

（2）采用左除运算：x=A\b

说明：

1、由于 MATLAB 遵循 IEEE 算法，所以即使 A 阵奇异，该运算也照样进行。但在运算结束时，一方面给出警告：“Warning:Matrix is singular to working precision”；另一方面，所得逆阵的元素都是“Inf”（无穷大）。

1、当 A 为“病态”时，也给出警告信息。

2、在 MATLAB 中，inv 指令不很有用。MATLAB 推荐：尽量使用除运算，少用逆运算。

例 1：“求逆”法和“左除”法解恰定方程组的性能对比。

为对比两种方法的性能，先用以下指令构造一个条件数很大的高阶恰定方程组。

```
rand('seed',12);%选定随机种子，目的是可重复产生随机矩阵 A
A=rand(500)+1.e8;%rand(500)生成500×500均匀分布的随机矩阵
                 %每个随机矩阵元素加一个数的目的是使 A 的条件数变大
x=ones(500,1);  %令解向量 x 为全 1 的 500 元列向量
b=A*x;     %为使 Ax=b 方程一致，用 A 和 x 生成 b 向量
cond(A)    %计算矩阵 A 的条件数
ans =
  1.7608e+013
过程 1：“求逆”法解恰定方程组的误差、残差和所用计算时间
tic   %启动记时器（Stopwatch Timer）
xi=inv(A)*b; %xi 是用“求逆”法解恰定方程组所得的解
toc    %关闭计时器，并显示解方程所用的时间
eri=norm(x-xi) %解向量 xi 与真解向量的 2-范误差
rei=norm(A*xi-b)/norm(b) %方程的 2-范相对残差
elapsed_time =
    7.2500
eri =
    0.0066
rei =
  1.5488e-006
```

过程 2：“左除”法解恰定方程组的误差、残差和所用计算时间

```
tic
xd=A\b;% xd 是用“左除”法解恰定方程组所得的解
toc
erd=norm(x-xd)
red=norm(A*xd-b)/norm(b)

elapsed_time =
    3.3500
erd =
    0.0021
red =
  1.2695e-015
```

说明：

1）计算结果表明：除法求解比求逆求解速度明显快，精度相当；但“除法”的相对残差几乎是“机器零”，而“逆阵”法的相对残差高得多。

2）MATLAB 在设计求逆函数 inv 时，采用的是 Gauss 消去法。

3）MATLAB 在设计“左除”运算解恰定方程时，并不求逆，而是直接采用 Gauss 消去法求解，有效地减少了残差，所以即便在高条件数下也能得到较好的结果。

3.5.2 矩阵除法解超定方程组

（1）求正则方程（Normal equations）$(A^TA)x = A^Tb$ 的解。

（2）用 Householder 变换（Householder transformation）直接求原超定方程的最小二乘解。

由于第二种方程法采用的是正交变换，据最小二乘理论可知，第二种方法所得的解的准确性、可靠性都比第一种方法好得多。MATLAB 解超定方程组用的就是第二种方法。

例：除运算解超定方程的简单算例。

```
a=[1 2 3;4 5 -6;7 8 9;10 11 12];
b=[1:4]';
x=a\b
x = -0.3333
     0.6667
     0.0000
```

3.5.3 矩阵除法解欠定方程组

欠定方程的解是不唯一的。用除法运算所得的解有两个重要特征：（1）在解中至多有 Rank(A)个非零元素；（2）它是这类型解中范数最小的一个。

例：除运算解欠定方程的简单算例。

```
a=[1 2 3;4 5 -6;7 8 9;10 11 12];
b=a';
c=[1 3 3]';
x=b\c
 x =
    2.0000
    0.1667
         0
   -0.1667
```

3.6 矩阵分解函数

3.6.1 LU 分解

[L,U]=lu(X)：产生一个上三角矩阵 U 和一个“心理上下三角矩阵“L（即由下三角 矩阵和置换矩阵构成），使得 X=L*U 。

[L,U,P]=lu(X)：产生一个上三角矩阵 U 和一个下三角矩阵 L 以及一个置换矩阵 P，使得 P*X=L*U

注意：X 必须是方阵。

例 1：用 lu 分解的两种指令格式对矩阵 A 进行 LU 分解。

解：
```
A=[1 -1  1; 5 -4 3; 2 1 1]
[L,U]=lu(A)
[L,U,P]=lu(A)
```

```
A =
     1    -1     1
     5    -4     3
     2     1     1

[L,U]=lu(a)
L =
    0.2000   -0.0769    1.0000
    1.0000         0         0
    0.4000    1.0000         0
U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846
[L,U,P]=lu(a)
L =
    1.0000         0         0
    0.4000    1.0000         0
    0.2000   -0.0769    1.0000
U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846
P =
     0     1     0
     0     0     1
     1     0     0
```

例 2：A 为严格对角占优矩阵

```
A=[4 1 2 ;2 5 -1; 5 3 11]
[L,U]=lu(a)
```

```
A =
     4     1     2
     2     5    -1
     5     3    11
[L,U]=lu(A)
```

```
L=
    0.8000   -0.3684    1.0000
    0.4000    1.0000         0
  1.0000          0          0

U =
    5.0000    3.0000   11.0000
         0    3.8000   -5.4000
         0         0   -8.7895
```

例 3：利用 LU 分解求解线性方程组 AX=b

```
A=[1,-1,1;5,-4,3;2,1,1]
A =  1   -1    1
     5   -4    3
     2    1    1
b=[2;-3;1]
b =    2
      -3
       1
[L,U,P]=lu(A)
L =
    1.0000         0         0
    0.4000    1.0000         0
    0.2000   -0.0769    1.0000
U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846
P =
     0     1     0
     0     0     1
     1     0     0
y=L\(P*b)
y =
   -3.0000
    2.2000
    2.7692
x=U\y
x =
   -3.8000
    1.4000
  7.2000
```

1.0.0 QR 分解

[Q,R]=qr(A)：产生一个与矩阵 A 相同维数的上三角矩阵 R 和一个酉阵 Q，使得 A=Q*R

[Q,R,E]=qr(A)：产生一个置换矩阵 E，上三角矩阵 R 和一个酉阵 Q，使得 A*E=Q*R

例 1：

```
a=[1 -1 1;5 -4 3;2 1 1]
```

```
[Q,R]=qr(a)
[Q,R,E]=qr(a)
```

```
a =
     1    -1     1
     5    -4     3
     2     1     1
[Q,R]=qr(a)
Q =   -0.1826   -0.1501   -0.9717
      -0.9129   -0.3412    0.2242
      -0.3651    0.9279   -0.0747
R =
   -5.4772    3.4689   -3.2863
         0    2.4427   -0.2456
         0         0   -0.3737
[Q,R,E]=qr(a)
Q =
   -0.1826   -0.1501   -0.9717
   -0.9129   -0.3412    0.2242
   -0.3651    0.9279   -0.0747
R =
   -5.4772    3.4689   -3.2863
         0    2.4427   -0.2456
         0         0   -0.3737
E =
     1     0     0
     0     1     0
     0     0     1
```

例 2： a 为严格对角占优矩阵

```
a=[10 -1 1;5 -14 3;2 1 21]
[Q,R]=qr(a)
Q =
   -0.1826   -0.1501   -0.9717
   -0.9129   -0.3412    0.2242
   -0.3651    0.9279   -0.0747
R =
   -5.4772    3.4689   -3.2863
         0    2.4427   -0.2456
         0         0   -0.3737
[Q,R,E]=qr(a)
```

```
Q =
   -0.0471   -0.0678   -0.9966
   -0.1413   -0.9872    0.0738
   -0.9889    0.1443    0.0369
R =
  -21.2368    1.0359   -3.1549
         0   14.0331   -5.3254
         0         0   -9.5230
E =
     0     0     1
```

```
     0     1     0
     1     0     0
```

3.6.3 Choleshy 分解

Cholesky 分解要求被分解矩阵 A 为对称正定矩阵。

R=chol(A)：产生一个上三角矩阵 R，使得 $A = R^T * R$

例：

```
a=[2,1,1;1,2,-1;1,-1,3]
```

```
R=chol(a)
a =
     2     1     1
     1     2    -1
     1    -1     3
R=chol(a)
R =
    1.4142    0.7071    0.7071
         0    1.2247   -1.2247
         0         0    1.0000
```

3.7 多项式

3.7.1 多项式的表达和创建

多项式表达方式的约定：

多项式 $p(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$ 用以下系数行向量表示：

$p = [a_0, a_1, \cdots, a_{n-1}, a_n]$

多项式行向量的创建方法：

（1） 多项式系数向量的直接输入法；

（2） 利用指令：p=poly(AR)，产生多项式系数向量。

若 AR 是方阵，则多项式为特征多项式；若 AR 是向量，即 $AR = [ar_1, ar_2, \cdots, ar_n]$，则所得的多项式满足：

$(x - ar_1)(x - ar_2)\cdots(x - ar_n) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$

例：求 3 阶方阵 A 的特征多项式。

```
a=[11 12 13;14 15 16;17 18 19];
pa=poly(a)
ppa=poly2str(pa,'s')
```

```
pa =
    1.0000  -45.0000  -18.0000    0.0000
ppa =
  s^3 - 45 s^2 - 18 s + 5.3518e-015
```

注意：（1）n 阶方阵的特征多项式系数向量一定是(n+1)维的。

（2）特征多项式向量的第一的元素必是 1。

例：由给定根向量求多项式系数向量。

```
r=[-0.5,-0.3+0.4*i,-0.3-0.4*i];
p=poly(r)
pr=real(p)
```

```
ppr=poly2str(pr,'x')

p =    1.0000    1.1000    0.5500    0.1250
pr =    1.0000    1.1000    0.5500    0.1250
ppr =   x^3 + 1.1 x^2 + 0.55 x + 0.125
```

说明：

（1） 要形成实数多项式，根向量中的复数根必须共轭成对。

（2） 含复数根的根向量所生成的多项式系数向量（如 p）的系数有可能带在截断误差数量级的虚部。此时可采用取实部的指令'real'把虚部去掉。

（3） poly2str 是一个函数文件，它存在于 MATLAB 的控制工具包(Control Toolbox)中。

3.6.2 常用多项式运算指令

R=roots(p) 求多项式向量 p 的根

PA=polyval(p,S) 按数组运算规则计算多项式值。p 为多项式，S 为矩阵。

PM=polyvalm(p;S) 按矩阵运算规则计算多项式值。p 为多项式，S 为矩阵

[r,p,k]=residue(b,a) 部分分式展开。

b,a 分别是分子、分母多项式系数向量；

r,p,k 分别是留数、极点、直项向量。

P=polyfit(x,y,n) 用 n 阶多项式拟合 x,y 向量给定的数据。

例 1：求多项式 $x^3-6x^2-72x-27$ 的根。

解：

```
    r=roots([1,-6,-72,-27])
r =
  12.1229
  -5.7345
  -0.3884
```

注意：MATLAB 约定：多项式系数用行向量表示，一组根用列向量表示。

例 2：两种多项式求值指令的差别。

```
s=pascal(4)
p=poly(s);
pp=poly2str(p,'x')
pa=polyval(p,s)
pm=polyvalm(p,s)

s =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
pp =
   x^4 - 29 x^3 + 72 x^2 - 29 x + 1
```

```
pa =
  1.0e+004 *
    0.0016    0.0016    0.0016    0.0016
    0.0016    0.0015   -0.0140   -0.0563
    0.0016   -0.0140   -0.2549   -1.2089
    0.0016   -0.0563   -1.2089   -4.3779
pm =
  1.0e-011 *
   -0.0077    0.0053   -0.0096    0.0430
   -0.0068    0.0481   -0.0110    0.1222
    0.0075    0.1400   -0.0095    0.2608
    0.0430    0.2920   -0.0007    0.4737
```

说明: pm 中的元素都很小，它是运算误差造成的。理论上它应该是零。这就是著名的“Caylay-Hamilton”定理：任何一个矩阵满足它自己的特征多项式。

例 3：用 6 阶多项式对区间[0，2.5]上的误差函数 $y(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ 进行最小二乘拟合。

解：

```
x=0:0.1:2.5;
y=erf(x);     %计算“误差函数”在[0,2.5]内的数据点
p=polyfit(x,y,6)
px=poly2str(p,'x')
```

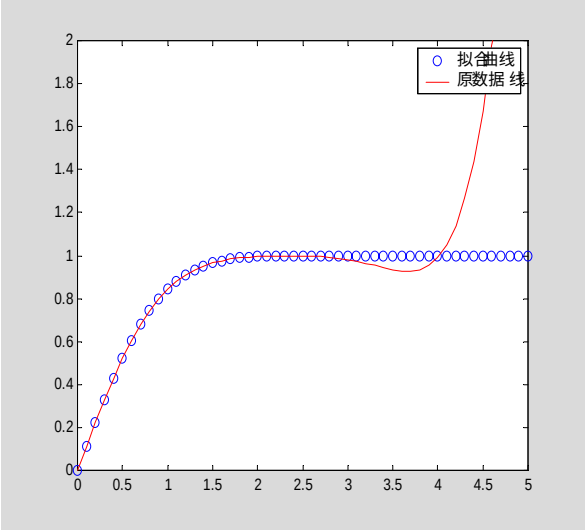
```
p =
    0.0084   -0.0983    0.4217   -0.7435    0.1471    1.1064
0.0004
px =
   0.0084194 x^6 - 0.0983 x^5 + 0.42174 x^4 - 0.74346 x^3
+ 0.1471 x^2
   + 1.1064 x + 0.00044117
```

例 4：有效拟合的**区间性**图示（用[0,2.5]区间数据拟合曲线拟合[0,5]区间数据）

```
x=0:0.1:5;
x1=0:0.1:2.5
y=erf(x);
y1=erf(x1);
p=polyfit(x1,y1,6)
f=polyval(p,x);
plot(x,y,'bo',x,f,'r-')
axis([0,5,0,2])
legend('拟合曲线','原数据线')
```

```
x1 =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000
0.6000
  Columns 8 through 14
    0.7000    0.8000    0.9000    1.0000    1.1000    1.2000
1.3000
  Columns 15 through 21
```

```
    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000
2.0000
  Columns 22 through 26
    2.1000    2.2000    2.3000    2.4000    2.5000
p =
    0.0084   -0.0983    0.4217   -0.7435    0.1471    1.1064
0.0004
```



说明：在[0,2.5]区间之外，两条曲线的表现完全不同。

3.8 数值积分

求函数数值积分指令

S=quad('fname',a,b,tol,trace)　——自适用递推 Simpson 数值积分法

S=quad8('fname',a,b,tol,trace)——自适用递推 Newton -Cotes 数值积分法

说明：

（1）输入参数'fname'是被积分函数表达式字符串或函数文件名。

（2）输入参数 a,b 分别是积分上下限。

（3）输入参数 tol 用来控制积分精度。缺省时取 tol=0.001 。

（4）输入参数 trace，若取 1 则用图形展先积分过程，取 0 无图形。缺省时，不画图。

（5）quad8 比 quad 有更高的积分精度。但无论是 quad8 还是 quad,都不能处理可积的“软奇异点”。

例 1：设 $f(x)=e^{-0.5x}\,sin(x+\frac{\pi}{6})$，求 $S=\int_0^{3\pi} f(x)dx$

解：

(1)建立被积函数文件 fesin.m

```
function f=fesin(x)
f=exp(-0.5*x).*sin(x+pi/6);
```

(2)把函数文件 fesin.m 存入自己的工作目录（如：d:\wzs），并在 matlab 命令窗口中用下指令使该目录成为当前目录。

```
cd  d:\wzs
```

(3)调用数值积分函数 quad 求定积分

```
S=quad('fesin',0,3*pi)
S=
```

0. 9008

建立被积函数文件时，注意所写程序应允许向量作为输入参数，所以在该函数文件中采用了.*运算符号。

例 2：分别用 quad 函数和 quad8 函数求 $\int_1^{2.5} e^{-x}dx$ 的近似值，并在同一积分精度下，比较被积函数被调用的次数。

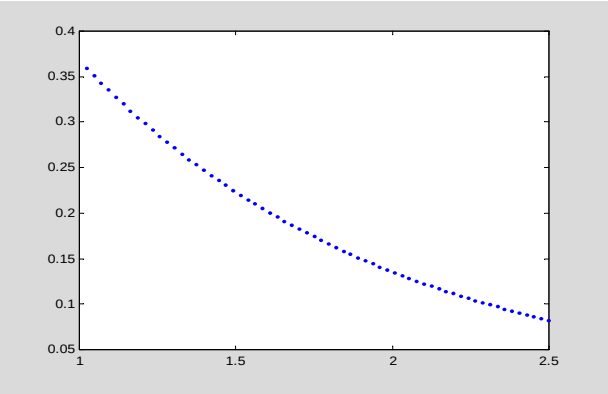
解：为了统计被积函数被调用的次数，可在被积函数文件中定义一个全局变量 num，每调用一次，num 加 1。被积分函数为 ftest.m：

```
function fx=ftest(x)
global num;
num=num+1;
fx=exp(-x);
```

调用函数 quad 求定积分：

```
global num;
num=0;
format long;
I=quad('ftest',1,2.5,1e-6,1),num
```

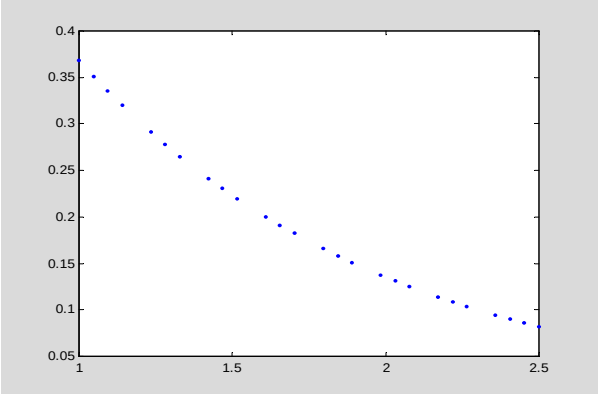
```
I =   0.28579444302661
num =    32
```



调用函数 quad8 求定积分：

```
global num;
num=0;
format long;
I=quad8('ftest',1,2.5,1e-6,1),num
```

```
I =   0.28579444254754
num =     4
```



可见，quad8 函数调用被积函数的次数是 4 次，而 quad 函数是 32 次，而且 quad8 的计算精度和效率也优于 quad。

3.9 优化和解非线性方程

非线性方程的求根方法很多，常用的有牛顿迭代法、弦截法等。MATLAB 提供了有关指令（函数）用于非线性方程求根。

3.9.1 多项式非线性函数求根： r=roots(p)

例：
```
p=[1 2 3 4];
r=roots(p)
r =
  -1.6506
  -0.1747 + 1.5469i
  -0.1747 - 1.5469i
```

3.9.2 单变量非线性方程求解（单变量函数求零点）

z=fzero('fname',x0,tol,trace)

其中：fname 是待求根的函数名；x0 为搜索的起点，一个函数可能有多个根，但 fzero 函数只给出离 x0 最近的那个根；tol 控制结果的相对精度，缺省时取 tol=eps;trace 指定迭代信息是否在运算中显示，为 1 时显示，为 0 时不显示，缺省时 trace=0。

例：求函数 $y(t)=0.2t-e^{-0.5t}\,sin(t+\frac{\pi}{6})$ 在 t=2 附近的零点。

步骤一：建立函数文件 f2.m

```
function y=f2(t)
y=0.2*t-exp(-0.5*t).*sin(t+pi/6);
```

步骤二：求方程的根（函数的零点）

```
z=fzero('f2',2)
z= 1.6993
```

3.9.3 一般非线性方程（组）求解

此部分内容已超出 MATLAB 的“主包”范围，它需要用到 MATLAB 的优化（Optimization）工具包。但由于非线性方程组是常用遇到的一类数学问题，因此在此一并简单介绍。对于一般非线性方程组 F(X)=0，其数值解 X 的求解指令：

X=fsolve('fname',X0)

其中：fname 是待求根的函数名；x0 是对解的初始猜测值。

例：求 $\begin{cases} sin(x)+y\text{^}2+ln(z)=7 \\ 3x+2\text{^}y-z\text{^}3+1=0 \\ x+y+z=5 \end{cases}$ 的数值解。

步骤一：建立函数文件 myxyz.m

```
function q=myxyz(p)
x=p(1);y=p(2);z=p(3);
q=zeros(3,1)   % 此指令可省略
q(1)=sin(x)+y^2+log(z)-7;
q(2)=3*x+2^y-z^3+1;
q(3)=x+y+z-5;
```

步骤二：在给定的初值下，调用 fsolve 函数求方程的根

x=fsolve('myxyz',[1 1 1]) 或 xyz0=[1 1 1];x=fsolve('myxyz',xyz0)

x= 0.5990
2.3959
2.0050

说明：从原理上讲，非线性方程组也可用符号工具包中的 solve 和 vpa 指令求数值解（详细可用 help 指查询），但计算时间难以接受。

3.10 微分方程的数值解

```
[t,x]=ode23('xprime',to,tf,x0,tol,trace)
[t,x]=ode45('xprime',to,tf,x0,tol,trace)
```

或

```
[t,x]=ode23('xprime',[t0,tf],x0,tol,trace)
[t,x]=ode45('xprime',[t0,tf],x0,tol,trace)
```

说明：

（1） 两个指令的调用格式完全相同，均为用 Runge-Kutta 法。

（2） 该指令是针对一阶常微分设计的。因此，假如待解的是高阶微分方程，那么它必须先演化为形如 $\dot{x} = f(x,t)$ 的一阶微分方程组，即“状态方程”。

（3） 'xprime'是定义 f(x,t)的函数文件名。该函数文件必须以 $\dot{x}$ 为一个列向量输出，以 t,x 为输入参量（注意变量的次序不可颠倒，一定先“时间变量”，后“状态变量”）。

（4） 输入参量 t0 和 tf 分别是积分的起始值和终止值。

（5） 输入参量 x0 为初始状态列向量。

（6） 输出参量 t 和 x 分别给出“时间”向量和相应的状态向量。

（7） tol 控制解的精度，可缺省。缺省时，ode23 默认 tol=1.e-3；ode45 默认 tol=1.e-6。

（8） 输入参量 trace 控制求解的中间结果是否显示，可缺省。缺省时，默认 tol=0，不显示中间结果。

（9） 一般地，两者分别采用自适应变步长（即当解的变化较慢时采用较大的步长，从而使得计算速度快；当解的变化速度较快时步长会自动地变小，从而使得计算精度高）的二、三阶 Runge-Kutta 算法和四、五阶 Runge_Kutta 算法，ode45 比 ode23 的积分分段少，而运算速度快

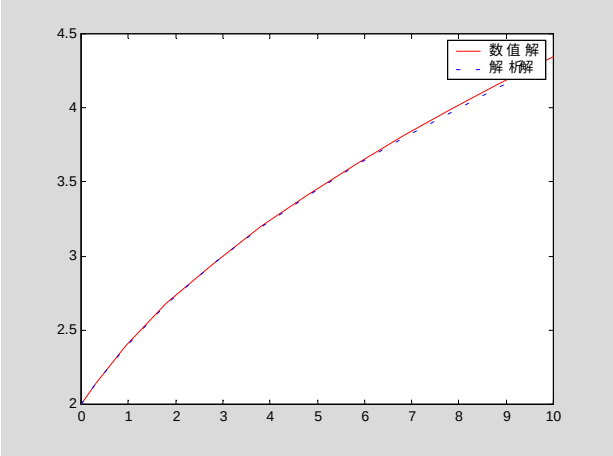
例 1：求初值问题 $\begin{cases} y' = \dfrac{y^2 - t - 2}{4(t+1)}, 0 \le t \le 10 \\ y(0) = 2 \end{cases}$ 的数值解，并与解析解 $y(t) = \sqrt{t+1} + 1$ 相比较。

解：(1)建立函数文件 funt.m

```
function   yp=funt(t,y)
yp=(y^2-t-2)/4/(t+1);
```

(2)求解微分方程

```
    t0=0;tf=10;y0=2;
[t,y]=ode23('funt',[t0,tf],y0);
y1=sqrt(t+1)+1;
t',y',y1'
plot(t,y,'-r',t,y1,':b')
      legend('数值解 ','解析解 ')
```




```
ans =
  Columns 1 through 7
         0    0.3200    0.9380    1.8105    2.8105    3.8105
4.8105
  Columns 8 through 13
    5.8105    6.8105    7.8105    8.8105    9.8105   10.0000
ans =
  Columns 1 through 7
    2.0000    2.1490    2.3929    2.6786    2.9558    3.1988
3.4181
  Columns 8 through 13
    3.6198    3.8079    3.9849    4.1529    4.3133    4.3430
ans =
  Columns 1 through 7
    2.0000    2.1489    2.3921    2.6765    2.9521    3.1933
3.4105
  Columns 8 through 13
    3.6097    3.7947    3.9683    4.1322    4.2879    4.3166
```

例 2：求著名的 Van Der Pol 方程 $\ddot{x}+(x^2-1)\dot{x}+x=0$ 的数值解并绘制其时间响应曲线和状态轨迹图。

（1） 演化为状态方程

令 $x_1=\dot{x},x_2=x$，把 $\ddot{x}+(x^2-1)\dot{x}+x=0$ 写成状态方程 $\begin{cases}\dot{x}_1=(1-x_2^2)x_1-x_2\\ \dot{x}_2=x_1\end{cases}$

（2） 建立函数文件 vdp.m

```
function    xdot=vdp(t,x)
xdot=zeros(2,1);%使 xdot 成为二元零向量（采用列向量，以便被 matlab 其他指令调用）
xdot(1)=(1-x(2)^2)*x(1)-x(2);
xdot(2)=x(1);
```

或者

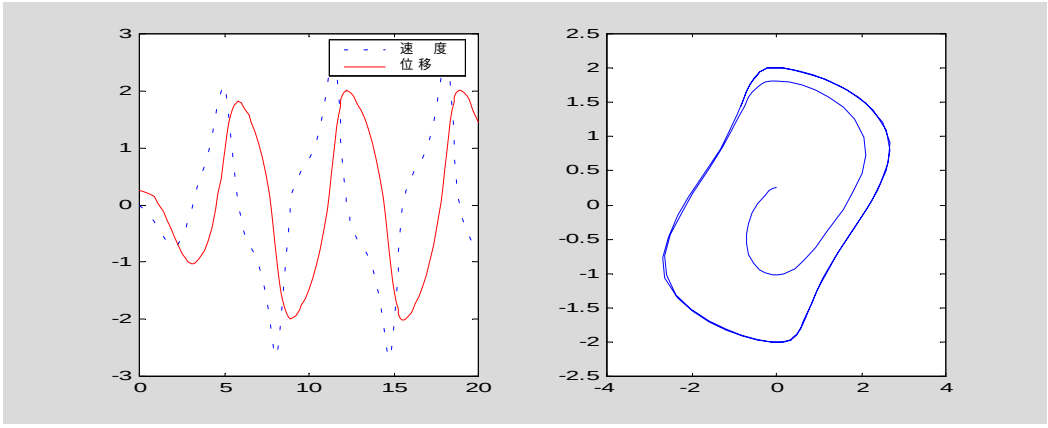
```
function    xdot=vdp(t,x)
xdot(1)=(1-x(2)^2)*x(1)-x(2);
xdot(2)=x(1);
xdot=xdot'     %采用列向量，以便被 matlab 其他指令调用
```

或者

function xdot=vdp(t,x)
xdot=[(1-x(2)^2)*x(1)-x(2);x(1)];
或者
function xdot=vdp(t,x)
xdot=[(1-x(2)^2),-1;1,0]*x;

(3)求解微分方程

```
t0=0;tf=20;x0=[0,0.25]';
[t,x]=ode23('vdp',t0,tf,x0);
subplot(1,2,1)
plot(t,x(:,1),':b',t,x(:,2),'-r')
legend('速度','位移')
subplot(1,2,2)
plot(x(:,1),x(:,2))
```



Van Der Pol 方程的时间响应曲线和状态轨迹图

另外：较高版本 MATLAB 中可使用指令 ode23p('fname',t0,tf,x0,tol)可以直接得到状态方程状态轨迹（可用“help ode23t”查询）。如下

```
clf;
axis([-3,3,-3,3]);
hold on
ode23t ('vdp',0,20,[0;0.25])
```

但由于现有的 MATLAB 的 Notebook 不具备表现动态图形的能力，此处无法显示。读者可在 MATLAB 的指令窗（Command windows）运行上述指令，并在图形窗口（Figure）中观察 ode23p 给出的动态轨迹。

3.11盘文件管理与工作平台间交换数据

3.11.1 磁盘文件管理

MATLA B 的 save 和 load 提供了对基本工作空间的保存和重新调入的功能，这对需长时间操作或分时工作的设计带来了方便。另外，MATLAB 还提供了管理磁盘文件、输入、输出数据等功能。

1、 文件管理

MATLAB 也提供了 dir, type, delete 及 cd 等用于文件管理的操作系统命令。具体用法请参见帮助系统（如 help dir）。

```
help dir
DIR List directory.
```

```
   DIR directory_name lists the files in a directory.
Pathnames and
   wildcards may be used.  For example, DIR *.m lists all
the M-files
   in the current directory.

   D = DIR('directory_name') returns the results in an M-
by-1
   structure with the fields:
      name  -- filename
      date  -- modification date
      bytes -- number of bytes allocated to the file
      isdir -- 1 if name is a directory and 0 if not

   See also WHAT, CD, TYPE, DELETE.
```

另：diary 指令可将 MATLAB 任务保存到磁盘的 ASCII 码文件（但不能保存图形），这样就可应用到由其它字处理器产生的文件或报表中。有关 diary 的详细具体用法，参阅 help diary 。

2、 执行外部程序

感叹号（！）是 MATLAB 的一个特殊命令，它指示其后的命令为操作系统命令。利用“！”可以在不退出 MATLAB 的情况下执行其它程序或操作系统命令，这是很有用的。例如：！edit wzs.m 它在不退出 MATLAB 情况下调用名为 edit 的编辑器，并对文件 wzs.m 文件进行编辑。当完成编辑任务后，仍返回 MATLAB 工作空间。

又如： ！ dir 执行 DOS 的 dir 命令，即列出当前目录下的所有文件。

3、 输入和输出数据

1）输入数据

使用 fopen 函数打开二进制文件，然后使用 fread 函数将数据装入一个变量中。此方法可装入由其它文件格式的应用程序产生的数据。

2）输出数据

采用 save 命令及-ascii 选项可按 ASCII 形式保存指定的数据。例如：

```
a=rand(4,3)
save wzs.dat a –ascii
```

则建立一个 ASCII 码文件 wzs.dat，其内容为矩阵 a 的内容。

3.9.2 工作平台间交换数据

MATLAB 提供了文件输入及输出（I/O）函数。利用这些函数可将其它格式的数据读入到 MATLAB 中，也可将 MATLAB 产生的数据按指定格式写入到文件，MATLAB 的 I/O 函数只能读写格式化文本或二进制数据文件。

1、 打开和关闭文件

在读写文件之前，必须先用 fopen 指令打开一个文件，并指定允许进行的操作，对文件读写操作后，应及时关闭这个文件，因为操作系统对可同时打开的文件数目有一定的限制。

例如：打开一个名为 wzs.dat 的文件，并进行后续的读操作：

fid=fopen('wzs.dat','r')

fopen 指令有两个参数：第一个参数为操作对象即文件名（可包括路径）；第二个参数指示允许操作的类型。具体如下表。

文件操作类型

字 符	允许操作类型
r	读操作
w	写操作
a	添加操作
r+	读/写操作

fopen 函数返回一个文件表识符，具体用法请参阅帮助系统。如下：

```
help fopen
 FOPEN  Open file.
    FID = FOPEN(FILENAME,PERMISSION) opens the specified
file with
    the specified PERMISSION. If the file is opened for
reading and it
    is not found in the current working directory, FOPEN
searches down
    MATLAB's search path.  Permission is one of the
strings:

        'r'     read
        'w'     write (create if necessary)
        'a'     append (create if necessary)
        'r+'    read and write (do not create)
        'w+'    truncate or create for read and write
        'a+'    read and append (create if necessary)
        'W'     write without automatic flushing
        'A'     append without automatic flushing

    Files can be opened in binary mode (the default) or in
text mode.
    In binary mode no characters get singled out for
special treatment.
    In text mode line separators can get special treatment
whereby
    characters are deleted on input before they reach
MATLAB and added
    on output.  To open in text mode, add 't' to the
permission string,
    for example 'rt' and 'wt+'.  (On Unix and Macintosh
systems, text
    and binary mode are the same so this has no effect.
But on PC and
    VMS systems this is critical.)
```

```
   If the open is successful, FID gets a scalar MATLAB
integer, the
   file identifier, to be used as the first argument to
other FileIO
   routines. If the open was not successful, -1 is
returned for FID.

   Two file identifiers are automatically available and
need not be
   opened.  They are FID=1 (standard output) and FID=2
(standard error).

   [FID, MESSAGE] = FOPEN(FILENAME,PERMISSION) returns a
system
   dependent error message if the open is not successful.

   [FID, MESSAGE] =
FOPEN(FILENAME,PERMISSION,MACHINEFORMAT) opens the
   specified file with the specified PERMISSION and
treats data read
   using FREAD or data written using FWRITE as having a
format given
   by MACHINEFORMAT. MACHINEFORMAT is one of the
following strings:

   'native'     or 'n' - local machine format - the
default
   'ieee-le'    or 'l' - IEEE floating point with little-
endian
                       byte ordering
   'ieee-be'    or 'b' - IEEE floating point with big-
endian
                       byte ordering
   'vaxd'       or 'd' - VAX D floating point and VAX
ordering
   'vaxg'       or 'g' - VAX G floating point and VAX
ordering
   'cray'       or 'c' - Cray floating point with big-
endian
                       byte ordering
   'ieee-le.l64' or 'a' - IEEE floating point with
little-endian
                       byte ordering and 64 bit long data
type
   'ieee-be.l64' or 's' - IEEE floating point with big-
endian byte
                       ordering and 64 bit long data type.

   [FILENAME,PERMISSION,MACHINEFORMAT] = FOPEN(FID)
returns the filename,
   permission, and machineformat associated with the
given file
   identifier. If FID does not exist then an empty string
is returned for
```

```
   each variable.

   FIDS = FOPEN('all') returns a row vector, the file
identifiers for
   all the files currently opened by the user (but not 1
or 2).

   The 'W' and 'A' permissions are designed for use with
tape drives and
   do not automatically perform a flush of the current
output buffer
   after output operations. For example, open a 1/4"
cartridge tape on a
   SPARCstation for writing with no auto-flush:

          fid = fopen('/dev/rst0','W')

   See also FCLOSE, FREWIND, FREAD, FWRITE, FPRINTF.
```

1、 读二进制数据文件

fread 函数可读取二进制的数据文件。

例如：将 wzs.dat 中的所有数据（无符号字符）读入并写入到矩阵 a 中：

```
fid=fopen('penny.dat','r');
a=fread(fid);
ststus=fclose(fid);
```

fread 函数还可以有两个参数，用于控制读出数据数目和每个数据的精度。

例如：将 wzs.dat 的前 100 个数据（无符号字符）读入并写入矩阵 a 中。

```
fid=fopen('wzs.dar','r');
a=fread(fid,100);
ststus=fclose(fid);
```

或 读入 wzs.dat 的前 100 个数据，并组合成 10 阶方阵：

```
fid=fopen('wzs.dar','r');
a=fread(fid,[10，10]);
ststus=fclose(fid);
```

3、 写二进制数据文件

MATLAB 的 write 函数与 fread 函数相对应，其功能是按照指定的数据精度，将矩阵中的元素写入到文件中，并返回写入数据的个数。

例如：创建一个含 100 个字节的二进制文件，即包含了 5 阶魔方矩阵的 25 个元素，每个元素以 4 字节的整数方式存储。在变量 count 中得到 fwrite 函数的返回值 25。

```
fwriteid=fopen('magic.bin','w');
count=fwrite(fwriteid,magic(5),'integer*4');
status=fclose(fwriteid);
```

第四章　　MATLAB 的符号计算功能

除了数值计算以外，在数学、物理、应用科学和工程中经常遇到符号计算问题。 MATLAB 开发和销售商 MathWorks 以 Maple 系统的“内核”为符号计算“引擎（Engine）”，依赖 Maple 已有的库（Library），开发了 MATLAB 环境下实现符号计算的工具包 Symbolic Math Toolbox。

此工具箱在 MATLAB 中的符号是 Toolbox/Symbolic，下面对它的常用功能作简单介绍，详细用法请参阅帮助系统。

一、　符号变量与符号表达式

符号运算工具箱处理的主要对象是符号和符号表达式，为此要定义新的数据类型——符号变量，MATLAB 中用 **sym** 来定义一个符号或符号表达式。

例:

```
sym('x')
ans =
x
r=sym('(1+sqrt(x))/2')
r =
(1+sqrt(x))/2
```

syms 可以定义多个符号：

```
syms a b c x k t y
f=a*(2*x-t)^3+b*sin(4*y)
f =
a*(2*x-t)^3+b*sin(4*y)
g=f+cos(k*x)
g =
a*(2*x-t)^3+b*sin(4*y)+cos(k*x)
```

用 **findsym** 来确认符号表达式中的符号：

```
findsym(g)
ans =
a, b, k, t, x, y
```

二、　微积分运算

1、 导数

diff(f)　——函数 f 对符号变量 x 或字母表上最接近字母 x 的符号变量求导数；

diff(f,t)——函数 f 对符号变量 t 求导数。

例：

```
 syms a b t x y
f=sin(a*x)+cos(b*t);
g=diff(f)
gg=diff(f,t)    %可以看作二元函数求偏导数）
g =
cos(a*x)*a
gg =
-sin(b*t)*b
```

用 diff(f,2)求二阶导数

例：

```
 syms a b t x y
f=sin(a*x)+cos(b*t);
f=sin(a*x*t)+cos(b*t*x^2)-2*x*t^3;
diff(f,2)
diff(f,t,2)
ans =
-sin(a*x*t)*a^2*t^2-4*cos(b*t*x^2)*b^2*t^2*x^2-
2*sin(b*t*x^2)*b*t
ans =
-sin(a*x*t)*a^2*x^2-cos(b*t*x^2)*b^2*x^4-12*x*t
```

当微分运算作用于符号矩阵时，是作用于矩阵的每个元素，如：

```
 syms a x
 a=[sin(a*x),cos(a*x);-cos(a*x),-sin(a*x)]
dy=diff(a)
a =
[  sin(a*x),  cos(a*x)]
[ -cos(a*x), -sin(a*x)]
dy =
[  cos(a*x)*a, -sin(a*x)*a]
[  sin(a*x)*a, -cos(a*x)*a]
```

2、 积分

int(f)——函数 f 对符号变量 x 或接近字母 x 的符号变量求不定积分；

int(f,t)——函数 f 对符号变量 t 求不定积分；

int(f,a,b)——函数 f 对符号变量 x 或接近字母 x 的符号变量求从 a 到 b 的定积分；

int(f,t,a,b)——函数 f 对符号变量 t 求从 a 到 b 的定积分。

例：

```
syms a x
f=sin(a*x)
g=int(f)
gg=int(f,a)

f =sin(a*x)
ff =sin(x^3)
g =-1/a*cos(a*x)
gg =-1/x*cos(a*x)
```

例：

```
syms a x
f=sin(a*x)
g=int(f,0,pi)
f =
sin(a*x)
g =-(cos(pi*a)-1)/a
```

注：当不定积分无解析表达式时，可用 double 计算其定积分的数值解。

例：

```
sym x
```

```
f=exp(-x^2)
g=int(f)
gg=int(f,0,1)
a=double(gg)

ans =x
f =exp(-x^2)
g =1/2*pi^(1/2)*erf(x)
gg =1/2*erf(1)*pi^(1/2)
a =   0.7468
```

3、 极限

limit(f)——当符号变量 x （或最接近字母 x 的符号变量）—>0 时函数 f 的极限；

limit(f,t,a)——当符号变量 t —>a 时，函数 f 的极限。

例：

```
syms x t a
f=sin(x)/x
g=limit(f)
limit((cos(x+a)-cos(x))/a,a,0)
limit((1+x/t)^t,t,inf)

f =sin(x)/x
g =1
ans =-sin(x)
ans =exp(x)
```

例：左、右极限的求法

```
sym x
limit(1/x)
limit(1/x,x,0,'left')
limit(1/x,x,0,'right')

ans =x
ans =NaN
ans =-inf
ans =inf
```

4、 级数和

symsum(s,t,a,b)——表示 s 中的符号变量 t 从 a 到 b 的级数和（t 缺省时设定为 x 或 最接近 x 的字母）

例：

```
syms x k
symsum(1/x,1,3)
ans =11/6
s1=symsum(1/x^2,1,inf)
s2=symsum(x^k,k,0,inf)
s1 =1/6*pi^2
s2 =-1/(x-1)
```

5、 泰勒（Taylor）多项式

taylor(f,n,a)——函数 f 对符号变量 x （或最接近字母 x 的符号变量）在 a 点的 n-1 阶泰勒多项式（n 缺省时值为 6，a 缺省值为 0）

例：

```
taylor(sin(x))
ans =x-1/6*x^3+1/120*x^5

f=log(x)
s=taylor(f,4,2)
f =log(x)
s =log(2)+1/2*x-1-1/8*(x-2)^2+1/24*(x-2)^3
```

三、 解方程

1、 代数方程

solve(f,t)——对 f 中的符号变量 t 解方程 f=0（t 缺省值为 x 或最接近 x 的字母）

例：

```
syms a b x c
f=a*x^2+b*x+c
s=solve(f)
ss=solve(f,b)
f =a*x^2+b*x+c
s =
  [ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
  [ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
ss =-(a*x^2+c)/x
```

注：求解形如 f(x)=q(x)形式的方程，则需要用单引号把方程括起来。

例：

```
s=solve('cos(2*x)+sin(x)=1')
s =
[     0]
[    pi]
[ 1/6*pi]
[ 5/6*pi]
```

例：求解方程组

```
[x,y]=solve('x^2+x*y+y=3','x^2-4*x+3=0')
x =
[ 1]
[ 3]
y =
[   1]
[ -3/2]
```

即解为（1，1）和（3，-3/2）

2、 微分方程

dsolve('S','s1','s2',…,'x')

其中 S 为方程，s1,s1,s3,…为初始条件，x 为自变量。方程 S 中用 D 表示求导数，D2,D3,…表示二阶、三阶等高阶导数；初始条件缺省时，给出带任意常数 C1,C2,..的通解；自变量缺省值为 t 。也可求解微分方程组。

例：

```
dsolve('Dy=1+y^2')
  ans =tan(t+C1)
y=dsolve('Dy=1+y^2','y(0)=1','x')
  y =tan(x+1/4*pi)
x=dsolve('D2x+2*D1x+2*x=exp(t)','x(0)=1','Dx(0)=0')
  x =1/5*exp(t)+3/5*exp(-t)*sin(t)+4/5*exp(-t)*cos(t)

S=dsolve('Df=3*f+4*g','Dg=-4*f+3*g')     %解微分方程组
S =
    f: [1x1 sym]
    g: [1x1 sym]
```

计算结果返回在一个结构 S 中，为了看到其中 f,g 的值，有如下指令

```
f=S.f
g=S.g

f =exp(3*t)*(cos(4*t)*C1+sin(4*t)*C2)
g =-exp(3*t)*(sin(4*t)*C1-cos(4*t)*C2)
```

四、 线性代数

MATLAB 中大多数用于数值线性代数计算的命令，都可以用于符号变量线性代数的运算。

例：

```
A=[a b c;b c a;c a b]
B=[1 1 1]'
x=A\B
A1=triu(A)
L=eig(A)


A =
[ a, b, c]
[ b, c, a]
[ c, a, b]
B =
     1
     1
     1
x =
   [ 1/(a+c+b)]
   [ 1/(a+c+b)]
   [ 1/(a+c+b)]
A1 =
     [ a, b, c]
     [ 0, c, a]
     [ 0, 0, b]
L =
[                          a+c+b]
[  (b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)]
[ -(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)]
```

五、　　化简和代换

工具包中提供了许多化简符号表达式的函数，具有专门用途的函数及其功能如下：

collect ——合并同类项

expand ——将乘积展开为和式

horner ——把多项式转换为嵌套表示形式

simplify ——利用各种恒等式化简代数式

例：

```
clear
syms x t y

collect(x^3+2*x^2-5*x^2+4*x-3*x+12-3)
ans =
x^3-3*x^2+x+9

g=collect((1+x)*t+t*x)
g =
2*t*x+t

expand((x-1)*(x-2)*(x-3))
ans =
x^3-6*x^2+11*x-6
gg=expand(cos(x+y))
gg =
cos(x)*cos(y)-sin(x)*sin(y)

expand(cos(3*acos(x)))
ans =
4*x^3-3*x

horner(x^3-6*x^2+11*x-6)
ans =
-6+(11+(-6+x)*x)*x

ggg=horner(1.1+2.2*x+3.3*x^2)
ggg =
11/10+(11/5+33/10*x)*x

factor(x^3-6*x^2+11*x-6)
ans =
(x-1)*(x-2)*(x-3)

n=1:5;
x=x(ones(size(n)))
p=x.^n+1
f=factor(p)
x =
[ x, x, x, x, x]
p =
[   1+x, x^2+1, x^3+1, x^4+1, x^5+1]
```

```
f =
[                  1+x,                  x^2+1,
(1+x)*(x^2-x+1),              x^4+1, (1+x)*(x^4-
x^3+x^2-x+1)]

simplify((1-x^2)/(1-x))
ans =x+1

s=simplify(sin(x)^2+cos(x)^2)
s =1

q=simplify((1/x^3+6/x^2+12/x+8)^(1/3))
q =((2*x+1)^3/x^3)^(1/3)
```

工具包还提供了一个强有力的函数 simple，它综合运用上述函数进行化简，并找出长度最短的表达式，指令如下：

f=simple(S)——对表达式 S 进行化简，输出长度最短的表达式；

simple(S)——对表达式 S 进行化简，输出用各种函数化简的结果，及长度最短的表达式；

[f,how]=simple(S)——对表达式 S 进行化简，输出长度最短的表达式 f 及 f 是哪一个函数作用的结果 how 。

例：

```
f=simple(sin(x)^2+cos(x)^2)
f =1

simple(1/x^3+6/x^2+12/x+8)


simplify:
 (1+6*x+12*x^2+8*x^3)/x^3
 radsimp:
 (1+6*x+12*x^2+8*x^3)/x^3
 combine(trig):
 (1+6*x+12*x^2+8*x^3)/x^3
 factor:
 (2*x+1)^3/x^3
 expand:
 1/x^3+6/x^2+12/x+8
 combine:
 1/x^3+6/x^2+12/x+8
 convert(exp):
 1/x^3+6/x^2+12/x+8
 convert(sincos):
 1/x^3+6/x^2+12/x+8
 convert(tan):
 1/x^3+6/x^2+12/x+8
 collect(x):
 1/x^3+6/x^2+12/x+8
ans =
(2*x+1)^3/x^3
```

```
[f,how]=simple(1/x^3+6/x^2+12/x+8)
f =(2*x+1)^3/x^3
how =factor
```

工具包还提供了两种代换指令：
subs(S,old,new)——用符号 new 代替表达式 S 中的符号 old；
subexpr(S)——将表达式 S 中的公共部分用 sigma 表示。
例：

```
subs(x+y,x,4)
ans =4+y

syms a b
f=subs(cos(a)+sin(b),[a,b],[sym('alpha'),2])
f =cos(alpha)+sin(2)
```

subexpr 的用法请参阅帮助系统。

六、 其它

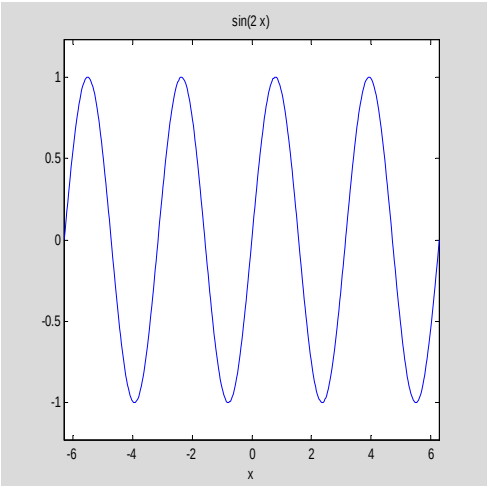
工具包中提供了 50 多个特殊函数，如 Bessel 函数、椭圆函数、误差函数及 Chebshev 正交多项式、Lagrange 正交多项式等，用指令 mfunlist 可以看到这些函数的列表；用指令 mhelp <函数名>可以了解每个函数的细节。
工具包对于数值计算提供了有理数计算方式和可变位数的浮点计算方式，用法如下**例：**

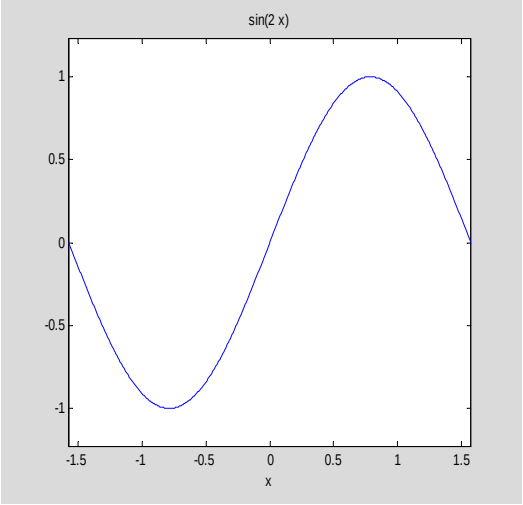
```
a=1/2+1/3
a1=sym(a)
a2=vpa(a,10)

a =   0.8333
a1 =5/6
a2 =.8333333333
```

工具包还提供了一个非常简便的画图指令：设表达式 f 中只有一个符号变量，比如 x ，则 ezplot(f,xmin,xmax)画出以 x 为横坐标的曲线 f ， x 在[xmin，xmax]内，缺省值为 xmin=-2Pi,xmax=2Pi.
例：

```
ezplot(sin(2*x))
ezplot(sin(2*x),-pi/2,pi/2)
```





第五章　MATLAB 的图形和可视化功能

作为一个功能强大的科技应用软件，MATLAB 具有很强的图形处理能力。

5.1 二维图形

MATLAB 中最常用的绘图函数为 plot，根据不同的坐标参数，它可以在二维平面上绘制出不同的曲线。

5.1.1　plot 函数

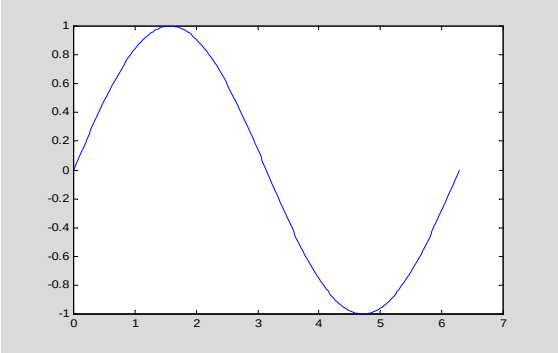
1. 格式与功能

plot 函数调用格式：plot(x,y)　其中 x 和 y 为坐标向量。

函数功能：以向量 x 作为 X 轴，以向量 y 作为 Y 轴，绘制 X—Y 二维曲线。

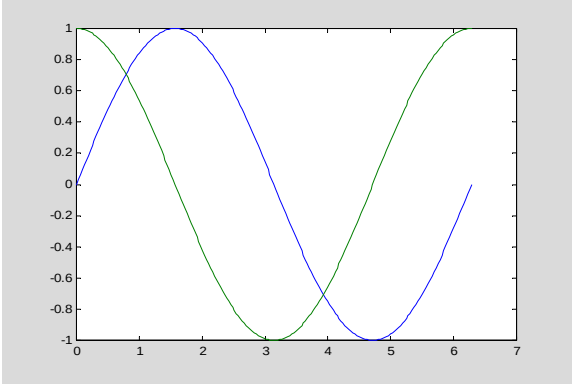
例5.1　在区间 *[* 0,2π *]* 内，绘制正弦曲线 $y = sin(x)$。

```
 x=0:pi/100:2*pi;
y=sin(x);
plot(x,y)
```



例5.2 在区间 $[0,2\pi]$ 内，同时绘制正弦曲线 $y=sin(x)$ 和余弦函数 $y=cos(x)$。

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,x,y2)
```



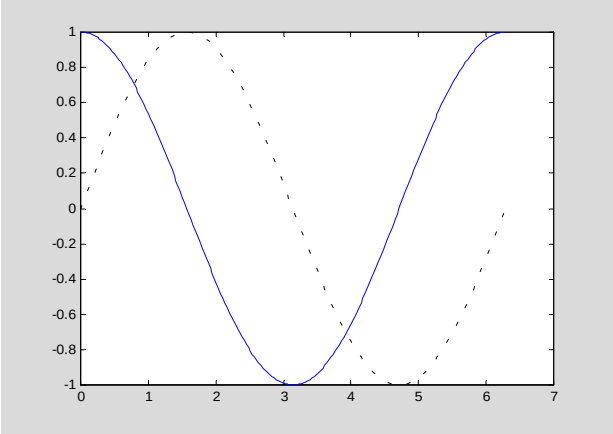
注意：plot(x,y1,x,y2,x,y3,…)以公共向量 x 为 X 轴，分别以 y1,y2,y3,…为 Y 轴，在同一副图内绘制出多条曲线。

2.线型与颜色

在 plot 绘图指令中增加一些参数，可以绘制出不同颜色与不同线型的图形。

例5.3 在区间 $[0,2\pi]$ 内，同时绘制不同线型不同颜色正弦曲线 $y=sin(x)$ 和余弦函数 $y=cos(x)$。

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'k:',x,y2,'b-')
```



每条曲线的线型和颜色由字符串'cs'指定，其中 c 表示颜色，s 表示线型。

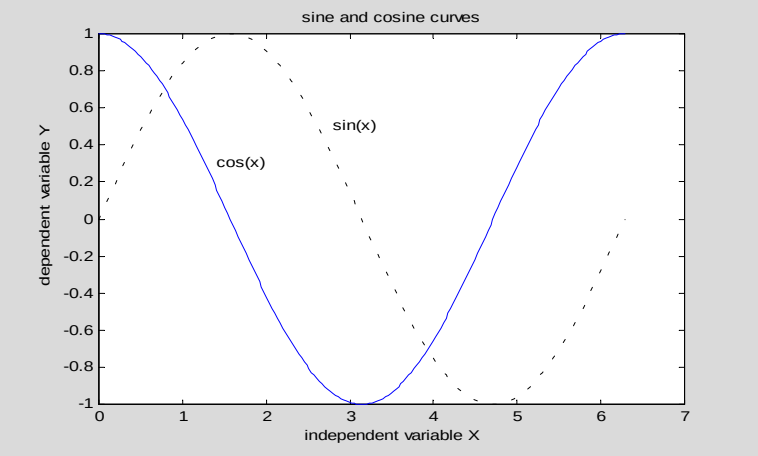
颜色与线型

颜色符号	颜色	线型符号 s	线型
y	黄色	.	点
m	紫色	。	圆圈
c	青色	x	叉号
r	红色	+	加号
g	绿色	*	星号
b	蓝色	-	实线
w	白色	:	点线
k	黑色	— .	点划线
		—	虚线

3.图形标记

可以对图形加上一些说明，如图形名称、图形某一部分的含义、坐标说明等。如下例

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'k:',x,y2,'b-')
 title('sine and cosine curves');
 xlabel('independent variable X');
 ylabel('dependent variable Y');
 text(2.8,0.5,'sin(x)');
text(1.4,0.3,'cos(x)');
```



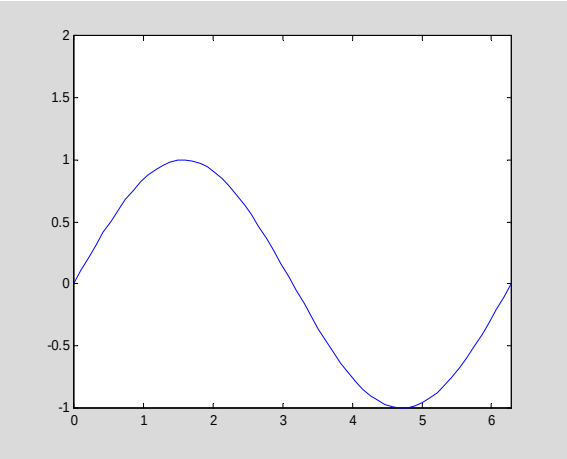
添加文本说明也可用 gtext('Z.S.Wang')指令，格式与功能可用 help gtext 查询。

4.设定坐标轴

在绘制图形时，系统自动给出图形的坐标轴。用户也可以利用 axis 函数对其重新设定。

例：在坐标范围 $0 \le x \le 2\pi, -1 \le y \le 2$ 内绘制正弦曲线。

```
x=linspace(0,2*pi,60);
y=sin(x);   %生成含有 60 个数据元素的向量 x
plot(x,y);
axis([0,2*pi,-1,2]);    %设定坐标范围
```



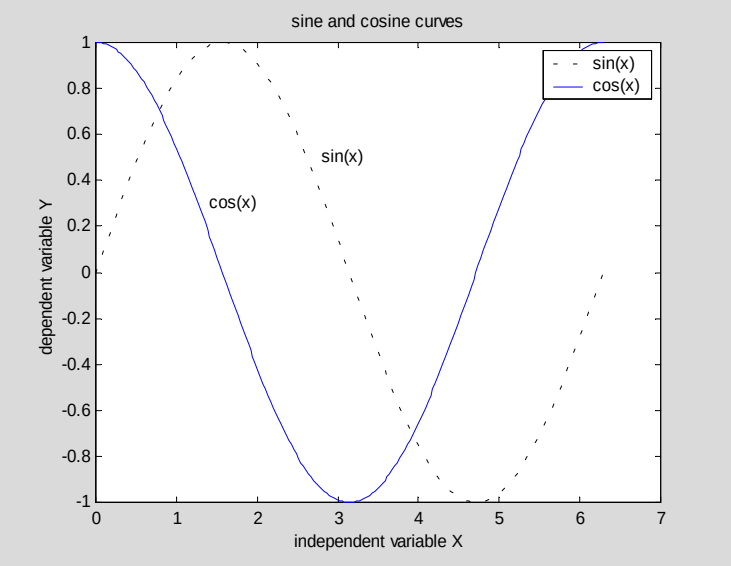
axis 函数功能

axis([xmin xmax ymin max])	设定坐标轴的最大值和最小值
axis('auto')	将坐标系统返回自动缺省状态
axis('square')	将当前图形设置为方形（系统默认为矩）
axis('equal')	两个坐标因子设定成相等
axis('off')	关闭坐标系统
axis('on')	显示坐标系统

5.添加图例

```
x=0:pi/100:2*pi;
```

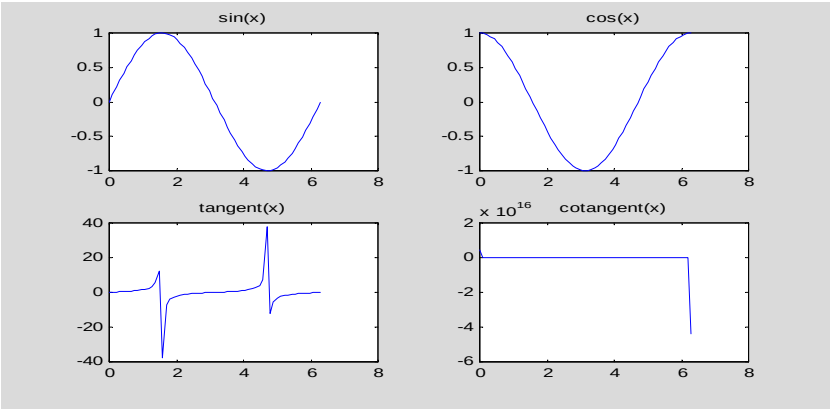
```
 y1=sin(x);
 y2=cos(x);
 plot(x,y1,'k:',x,y2,'b-')
title('sine and cosine curves');
xlabel('independent variable X');
ylabel('dependent variable Y');
text(2.8,0.5,'sin(x)');
text(1.4,0.3,'cos(x)');
 legend('sin(x)','cos(x)');
```



5.1.2 subplot 函数

1.subplot(m,n,p)

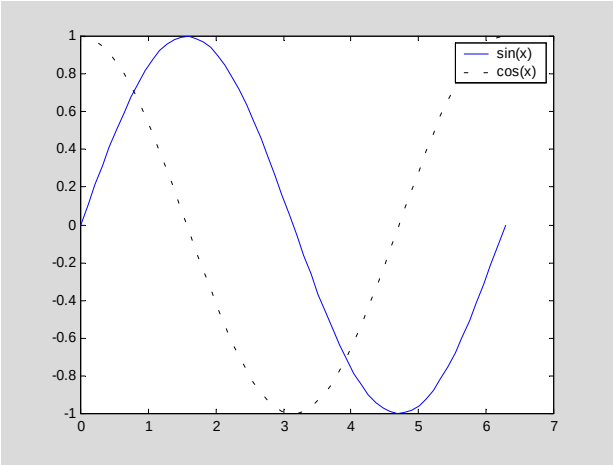
```
 x=linspace(0,2*pi,60);
 y=sin(x);
 z=cos(x);
 t=sin(x)./(cos(x)+eps);
 ct=cos(x)./(sin(x)+eps);
 subplot(2,2,1);
 plot(x,y);
 title('sin(x)');
 %axis([0,2*pi,-1,1]);
 subplot(2,2,2);
 plot(x,z);
 title('cos(x)');
 %axis([0,2*pi,-1,1]);
 subplot(2,2,3);
 plot(x,t);
 title('tangent(x)');
 %axis([0,2*pi,-40,40]);
 subplot(2,2,4);
 plot(x,ct);
 title('cotangent(x)');
%axis([0,2*pi,-40,40]);
```



1.hold 命令

若在已存在的图形窗口中用 plot 函数继续添加新的图形内容，可使用图形保持指令 hold。发出 hold on 后，再执行 plot 函数，在保持原有图形的基础上添加新的 绘制图形。hold off 关闭此功能。

```
x=linspace(0,2*pi,60);
y=sin(x);
z=cos(x);
plot(x,y,'b');
hold on;
plot(x,z,'k:');
%axis([0 2*pi -1 1]);
legend('sin(x)','cos(x)');
hold off
```



此函数与 plot(x,y1,x,y2,….)功能类似。

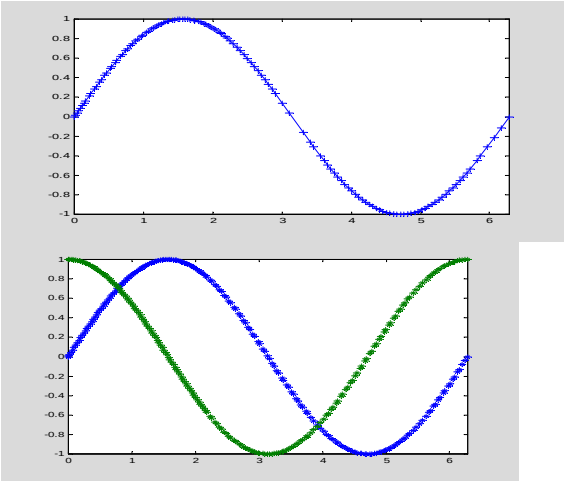
5.1.3 函数 f(x)曲线

绘制函数 f(x)的曲线方法有多种，最常用的方法：对采样点向量 x 计算出 f(x)的值向量 y，再用 plot(x,y)函数绘制。plot 函数一般采用等间隔采样，对绘制高频率变化的函数不够精确。例如函数 $f(x)=cos(tan(\pi x)), x\in(0,1)$ 范围是，有无限个震荡周期，函数变化率大。为提高精度，绘制出比较真实的函数曲线，就不能采用等步长采样，而必须在变化率大的区域密集采用，以充分反映函数的实际变化规律，提高图形的真实度。fplot 函数可自适应的对函数进行采样，能更好反映函数的变化规律。

函数格式： fplot(fname,lims,tol)

其中：fname 为函数名，以字符串形式出现； lims 为变量取值范围；tol 为相对允许误差，其默认值为 2e-3。
如以下都是合法的 fplot 语句：

```
fplot('sin(x)',[0 2*pi],'-+')
fplot('[sin(x),cos(x)]',[0 2*pi],1e-3,'*')
```

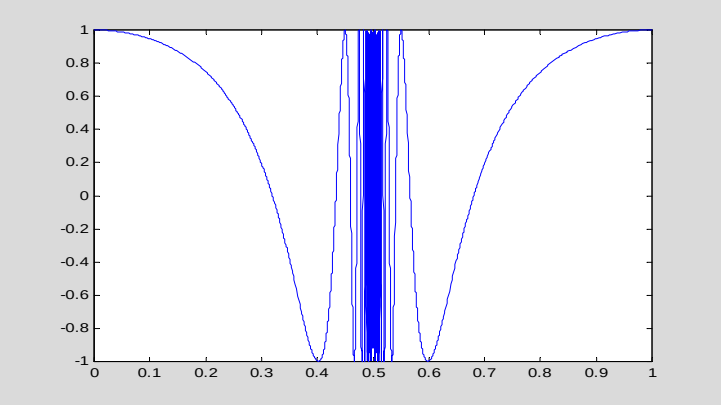


可见变化率大的区段采样点比较集中。
为了绘制 $f(x)=cos(tan(\pi x))$ 曲线，可先建立函数文件 fct.m，如下

```
function y=fct(x)
y=cos(tan(pi*x));
```

用 fplot 函数调用 fct.m 函数为

```
fplot('fct',[0,1],1e-4)
```



可见，在坐标 0.5 附近采样点十分密集，若采用等间隔采样不能真实反映函数变化规律。

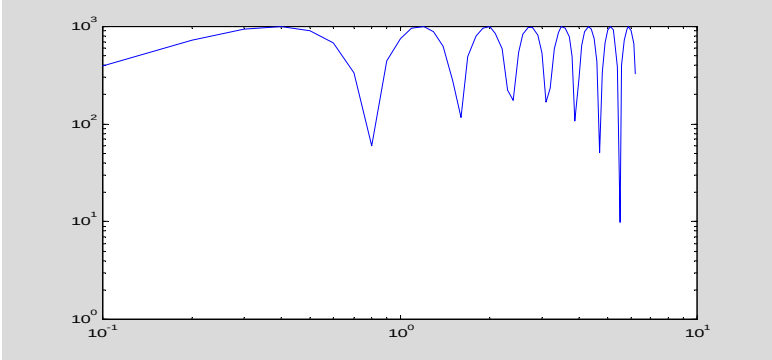
5.2 特殊坐标图形

5.2.1 对数坐标图形

1.双对数坐标

函数 loglog(x,y)用来绘制双对数坐标图。

```
x=0:0.1:2*pi;
y=abs(1000*sin(4*x))+1;
loglog(x,y);
```



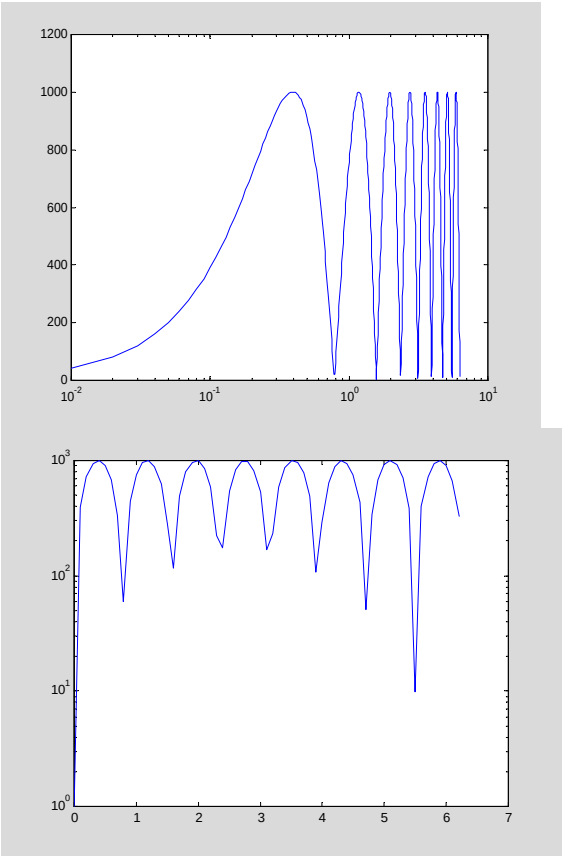
2.单对数坐标

函数 semilogy(x,y)用来绘制单对数坐标图。以 x 轴为对数绘制上图（左图）：

```
x=0:0.01:2*pi;
y=abs(1000*sin(4*x))+1;
semilogx(x,y);
```

同样，以 Y 轴为对数绘制上图（右图）

```
x=0:0.1:2*pi;
y=abs(1000*sin(4*x))+1;
semilogy(x,y);
```

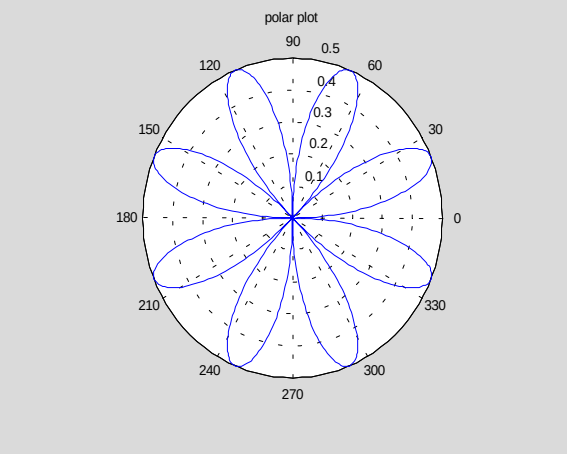


5.2.2 极坐标图形

函数 polar(theta,rho)用来绘制极坐标图，theta 为 极坐标角度，rho 极坐标半径。

例：绘制 $sin(2\theta)cos(2\theta)$ 极坐标图。

```
 theta=0:0.01:2*pi;
 rho=sin(2*theta).*cos(2*theta);
 polar(theta,rho);
title('polar plot');
```



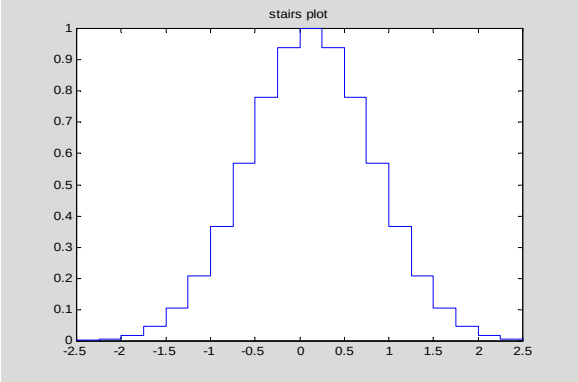
5.3 其他图形函数

除了 plot 等基本绘图函数外，MATLAB 系统还提供许多其他特殊绘图函数。这里举一些代表性例子，更多更详细的信息用户可随时查阅在线帮助，其对应的 M 文件在系统的\MATLAB\TOOLBOX\MATLAB 目录下。

5.3.1 阶梯图形

函数 stairs(x,y)可绘制阶梯图形。例如下：

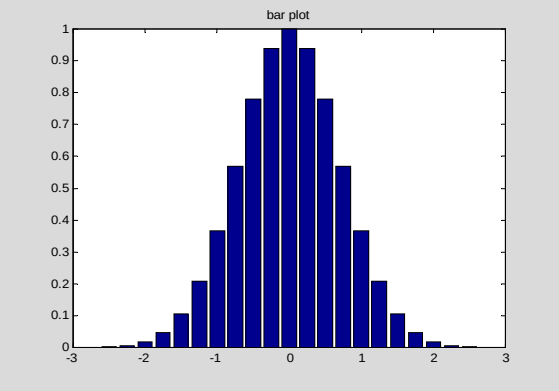
```
x=-2.5:0.25:2.5;
y=exp(-x.*x);
stairs(x,y);
title('stairs plot');
```



5.3.2 条形图形

函数 bar(x,y)用来绘制条形图。例如下：

```
x=-2.5:0.25:2.5;
y=exp(-x.*x);
bar(x,y);
title('bar plot');
```



5.3.3 二维绘图函数小结

plot	二维图形基本函数	close	关闭图形窗口
fplot	f(x)函数曲线绘制	figure	创建 图形窗口
fill	填充二维多边图形	grid	放置坐标网格线
polar	极坐标图形	gtext	用鼠标放置文本
bar	条形图	hold	保持当前图形的内容
loglog	双对数坐标图	subplot	创建子图
semilogx	X 轴为对数的坐标图	text	放置文本
semilogy	Y 轴为对数的坐标图	title	放置图形标题
stairs	阶梯图形	xlabel	放置 X 轴坐标标记
axis	设置坐标轴	ylabel	放置 Y 轴坐标标记
clf	清除图形窗口内容		

5.4 三维图形

为了显示绘制三维图形，系统提供了各种三维图形函数，如三维曲线、三维曲面以及设置图形属性的有关参数。

5.4.1 plot3 函数

最基本的三维图形函数为 plot3，它是将二维函数 plot 的有关功能扩展到三维空间，用来绘制三维图形。函数除了增加了第三维坐标外，其他功能与二维函数 plot 相同。

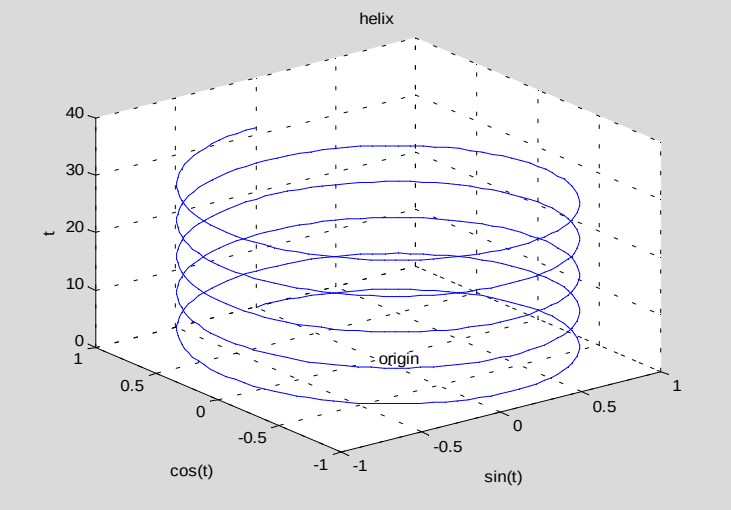
函数调用格式： $plot3(x1,y1,z1,c1,x2,y2,z2,c2,\cdots)$

其中：x1,y1,z1…表示三维坐标向量；c1,c2…表示线型或颜色。

函数功能：以向量 x,y,z 为坐标绘制三维曲线。

例：绘制三维螺旋线。

```
t=0:pi/50:10*pi;
y1=sin(t);,y2=cos(t);
plot3(y1,y2,t)
title('helix'),text(0,0,0,'origin');
xlabel('sin(t)'),ylabel('cos(t)'),zlabel('t');
grid;
```



5.1.2 mesh 函数

mesh 函数用于绘制三维网格图。在不需要绘制特别精细的三维曲面结构时，可通过绘制三维网格图来表示三维曲面图。三维曲面的网格图最突出的优点是：它较好地解决了实验数据在三维空间的可视化问题。

函数调用格式：mesh(x,y,z,c)

其中：x,y 控制 X 和 Y 轴坐标，矩阵 z 是由(x,y)求得的 Z 轴坐标，（x,y,z）组成了三维空间的网格点；c 用于控制网格点的颜色。

例：绘制三维网格曲面图。

```
x=0:0.15:2*pi;
y=0:0.15:2*pi;
z=sin(y')*cos(x);      %矩阵相乘
mesh(x,y,z)
title('三维网格图形');
```

5.1.3 surf 函数

surf 函数用于绘制三维曲面图，各线条之间的补面用颜色填充，其函数调用格式与 mesh 函数一样：surf(x,y,z)

其中：x,y 控制 X 和 Y 轴坐标，矩阵 z 是由(x,y)求得的曲面上 Z 轴坐标。

例：绘制三维曲面图。

```
   x=0:0.15:2*pi;
   y=0:0.15:2*pi;
   z=sin(y')*cos(x);
   surf(x,y,z)
title('3-D surf');
```

由上两图可以发现，曲面图

5.1.4 视点

日常生活中从不同的视点观察物体，所看到的图形是不同的。同样，用户从不同的角度绘制的三维图形的形状也是不一样的。视点位置可由方位角和仰角表示。方位角又称旋转角，它是视点位置在 XY 平面上的投影与 X 轴形成的角度，正值表示逆时针，负值表示顺时针。仰角又称视角，它是 XY 平面的上仰或下仰角，正值表示视点在 XY 平面上方，负值表示视点在 XY 平面下方。MATLAB 系统提供了从不同视点绘制三维图形的函数 view(az,el)，az 为方位角， el 为仰角。下面通过系统提供的多峰函数 peaks 的绘制例子，可进一步说明视点对图形的影响，以及 view(az,el)函数的使用。

例：绘制不同视角图形：

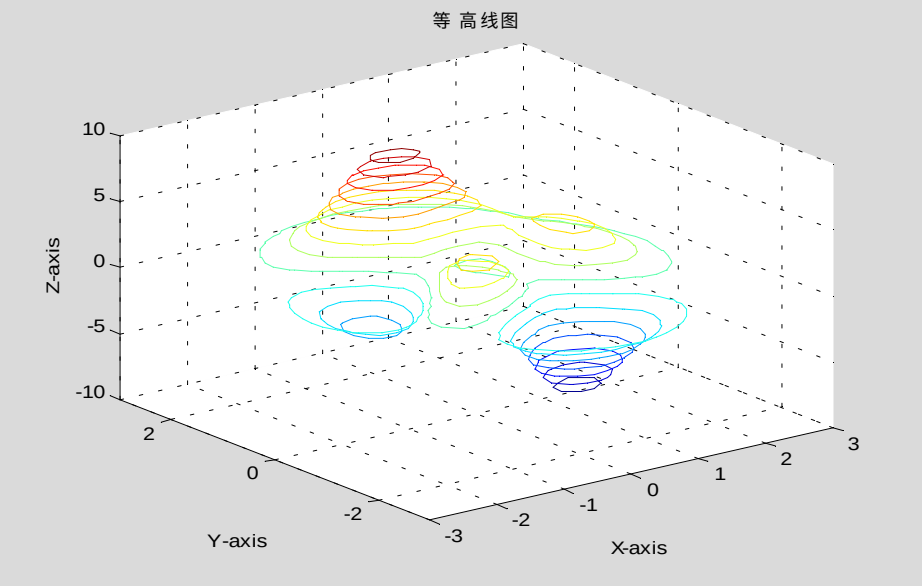
```
p=peaks;    %peaks 为系统提供的多峰函数
subplot(2,2,1);
mesh(peaks,p);
view(-37.5,30);  %指定子图 1 的视点
title('azimuth=-37.5,elevation=30');
subplot(2,2,2);
mesh(peaks,p);
view(-17,60);   %指定子图 2 的视点
title('azimuth=-17,elevation=60');
subplot(2,2,3);
mesh(peaks,p);
view(-90,0);    %指定子图 3 的视点
title('azimuth=-90,elevation=0');
subplot(2,2,4);
mesh(peaks,p);
view(-7,-10);    %指定子图 4 的视点
title('azimuth=-7,elevation=10');
```

5.1.5 等高线图

MATLAB 还提供另一种基本的三维图形，即三维等高线图。等高线图用函数 contour3 绘制。

例：绘制多峰函数 peaks 的等高线图。

```
hold off
[x,y,z]=peaks(30);
contour3(x,y,z,16);
xlabel('X-axis'),ylabel('Y-axis'),zlabel('Z-axis');
title('等高线图');
```



第六章　MATLAB 程序设计

MATLAB 有两种工作方式：一种是交互式的命令行工作方式；另一种是 M 文件的程序工作方式。在前一种工作方式下，MATLAB 被当作一种高级“数学演算纸和图形表现器”来使用的编程语言，为用户提供了二次开发的工具。本章介绍 MATLAB 有关控制语句和程序设计的基本方法。

3.1　M 文件

用 MATLAB 语言编写的程序，称为 M 文件。M 文件有两类：命令文件和函数文件。

两者区别在于：命令文件没有输入参数，也不返回输出参数；而函数文件可以输入参数，也可以返回输出参数。命令文件对 MATLAB 工作空间中的变量进行操作，而函数文件中定义的变量为局部变量，当函数文件执行完毕时，这些变量被清除。本节先介绍命令文件，关于函数文件将在 3.5 节介绍。

3.1.1　M 文件的建立与编辑

M 文件可以用任何编辑程序建立和编辑，而一般常用且最为方便的是使用 MATLAB 提供的 M 文件窗口。

1.建立新的 M 文件

从 MATLAB 命令窗口的 File 菜单中选择 New 菜单项，再选择 M-file 命令，将得到 M 文件窗口。在 M 文件窗口中输入 M 文件的内容，输入完毕后，选择此窗口 File 菜单的 save as 对话框。在对话框的 File 框中输入文件名（注意，其扩展名必须为.M），再选择 OK 按钮即完成新的 M 文件的建立。

2．编辑已有的 M 文件

从 MATLAB 命令窗口的 File 菜单中选择 Open M-file 命令，则屏幕出现 Open 对话框，在 Open 对话框中的 File Name 框中输入文件名（必要时加上路径），或从右边的 Directories 框中打开这个 M 文件所在的目录，再从 File Name 下面的列表框中选中这个文件，然后按 OK 按钮即打开这个 M 文件。在 M 文件窗口可以对打开的 M 文件进行编辑修改。在编辑完成后，选择 File 菜单中的 Save 命令可以把这个编辑过的 M 文件保存下来。

3.1.2 命令文件

当用户要运行的命令较多或需要反复运行多条命令时，直接从键盘逐行输入命令显得比较麻烦，而命令文件则可以较好地解决这一问题。我们可以将需要运行的命令编辑到一个文件中，然后在 MATLAB 命令窗口输入该命令文件的名字，就会顺序执行命令文件中的命令。

[例 3.1]建立一个命令文件将变量 a,b 的值互换，然后运行该命令文件。

解：(1)首先建立命令文件并以文件名 e31.m 存盘：

```
a=1:9;
b=[11,12,13;14,15,16;17,18,19];
c=a;a=b;b=c;
a
b
```

(2)在 MATLAB 的命令窗口中输入 e31,将会执行该命令文件，输出为：

```
a=
   11    12    13
   14    15    16
   17    18    19
b=
    1     2     3     4     5     6     7     8     9
```

调用该命令文件时，不用输入参数，也没有输出参数，文件自身建立需要的变量。当文件执行完毕后，可以用命令 whos 查看工作空间中的变量。这时会发现 a,b,c 等变量仍然保留在工作空间中。

3.2 数据的输入输出

MATLAB 的输入输出方式包括命令窗口的输入输出以及图形界面的输入输出。此外，它还允许对文件进行读写。这里先介绍命令窗口的输入输出。

Input 函数

MATLAB 提供了一些输入输出函数，允许用户和计算机之间进行数据交换。如果用户想给计算机输入一个参数，则可以使用 input 函数来进行，该函数的调用格式为：

```
A=input(提示信息，选项);
```

其中提示信息可以为一个字符串，它用来提示用户输入什么样的数据。例如，用户想输入 A 矩阵。则可以采用下面的命令来完成：

```
A=input('Enter matrix A=>');
```

执行该语句时首先给出 Enter matrix A=>提示，然后等待用户从键盘按 MATLAB 格式输入 A 矩阵。如果在 input 函数调用时采用's'选项，则允许用户输入一个字符串。例如想输入一个人的姓名，可采用命令：

```
xm=input('What"s your name:','s');
```

[例 3.2]求一元二次方程 ax2+bx+c=0 的根。

解 程序如下：

```
a=input('a=?');
b=input('b=?');
c=input('c=?');
d=b*b-4*a*c;
```

```
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)]
```

第一次运行：

```
a=?2
b=?6
c=?1
x=
    -0.1771          -2.8229
```

第二次运行：

```
a=?4
b=?1
c=?4
x=
    -0.1250+0.9922i              -0.1250-0.9922i
```

2.pause 函数

当程序运行时，为了查看程序的中间结果或观看输出的图形，有时需要暂停程序的执行。这时可以使用 pause 函数，其调用格式为：

```
pause(延迟秒数)
```

如果省略延迟时间，直接使用 pause，则将暂停程序，直到用户按任一键后程序继续执行。

3．Disp 函数

MATLAB 提供的命令窗口输出函数主要有 disp 函数，其调用格式为：

```
Disp(输出项)
```

其中输出项既可以为字符串，也可以为矩阵。例如

```
A='Hello,MATLAB';
Disp(A)
```

输出为：

```
Hello,MATLAB
```

又如：

```
A=[1,2,3;4,5,6;7,8,0];
Disp(A)
```

输出为：

```
1     2     3
4     5     6
7     8     9
```

注意，和前面介绍的矩阵显示方式不同，用 disp 函数显示矩阵时将不显示矩阵的名字，而且其格式更紧密，且不留任何没有意义

4、 关系及逻辑运算

在执行关系及逻辑运算时，MATLAB 将输入的不为零的数值都视为真 (True)而为零的数值则视为否 (False)。 运算的输出值将判断为真者以 1 表示而判断为否者以 0 表示。MATLAB 提供以下的关系判断及逻辑的运算元：

符号 关系的意义

< 小于

<= 小于等于
> 大于
>= 大于等于
= = 等于
~ = 不等于
& 逻辑 and
| 逻辑 or
~ 逻辑 not

上述的各个运算元须用在二个大小相同的阵列或是矩阵的比较，以下有几个例子：

```
>> a=1:5, b=5-a,
a =
     1 2 3 4 5
b =
     4 3 2 1 0

>> tf= a>4
tf =
0 0 0 0 1
>> tf= a= =b
tf =
0 0 0 0 0

>> tf= b-(a>2)
tf =
4 3 1 0 -1
>> tf= ~(a>4)
tf =
1 1 1 1 0
>> tf= (a>2)&(a<6)
tf =
0 0 1 1 1
```

以下是算式利用关系及逻辑运算产生一不连续的讯号

```
>> x=linspace(0,10,100); % 产生数据
          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.
>> y=sin(x); % 产生 sine 函数
>> z=(y>=0).*y; % 将 sin(x) 的负值设为零
```

```
>> z=z + 0.5*(y<0); % 再将上式的值加上 0.5
>> z=(x<8).*z; % 将大于 x=8 以后的值设为零
>> hold on
>> plot(x,z)
>> xlabel('x'),ylabel('z=f(x)')
>> title('A discontinuous signal')
>> hold off
```

除了上述的运算元之外，尚有以下的逻辑关系函数：xor(x,y), any(x), all(x), isnan(x), isinf(x), finite(x), find(x)，其 使用方式详见线上说明。

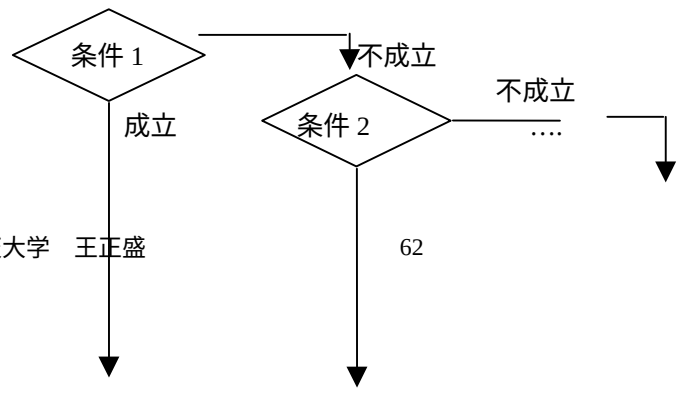
3．3 选择结构

选择结构是根据给定的条件成立或不成立，分别执行不同的语句。MATLAB 提供的用于实现选择结构的语句有 if 语句和 switch 语句。

3．3．1 if 语句

在 MATLAB 中，if 语句有 3 种格式：

```
格式 1：    if 条件
                语句组
            end
格式 2：    if 条件
                语句组 1
            else
                语句组 2
            end
格式 3：    if 条件 1
                语句组 1
            elseif 条件 2
                语句组 2
            …       …
            elseif 条件 m
                语句组 m
            else
                语句组 m+1
            end
```



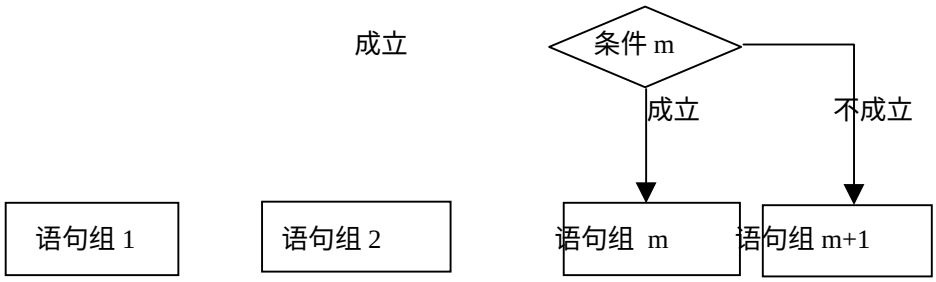


图 3.1　　if 语句格式 3 的执行过程

对于格式 1，当条件成立时，则执行语句组，执行完之后继续执行 if 语句的后继语句，如条件不成立，则直接执行 if 语句的后继语句。对于格式 2，当条件成立时，执行语句组 1，否则执行语句组 2，语句组 1 或语句组 2 执行后，再执行 if 语句的后继语句。格式 3 的执行过程如图 3.1 所示，可用于实现多分支选择结构。下面将通过例子来说明 if 语句的使用方法。

[例 3.3]输入三角形的三条边，求面积。

解　三角形的三条边分别看作 A 矩阵的三个元素，先从键盘输入 A 矩阵，再根据三条边能否构成三角形，决定求面积或给出有关信息。程序如下：

```
A=input('请输入三角形的三条边:');
If A(1)+A(2)>A(3)&A(1)+A(3)>A(2)&A(2)+A(3)>A(1)
   P=(A(1)+A(2)+A(3))/2;
   S=sqrt(p*(p-A(1))*(p-A(2))*(p-A(3)));
   Disp(s);
Else
   Disp('不能构成一个三角形。')
End
```

第一次运行：

```
请输入三角形的三条边：[4    5    6]
9．9216
```

第二次运行：

```
请输入三角形的三条边：[3    1    5]
不能构成一个三角形。
```

[例 3.4]输入一个字符，若为大写字母，则输出其后继字符，若为小写字母，则输出其前导字符，若为其他字符则原样输出。

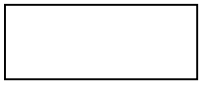
解　程序如下：

```
c=input(",'s');
if c>='A'&c<='Z'
   disp(setstr(abs(c)+1));
elseif c>='a'&c<='z'
   disp(setstr(abs(c)-1))
else
   disp(c);
end
```

3.3.2　switch 语句

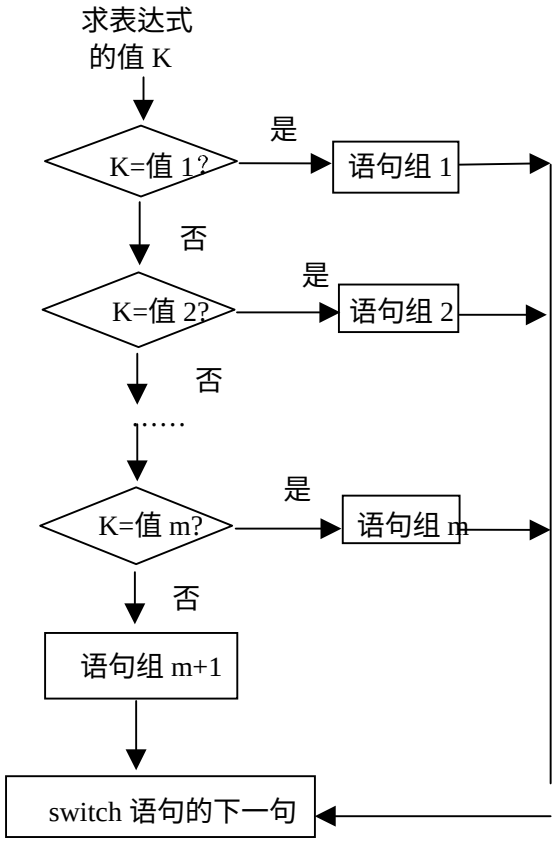
switch 语句根据变量或表达式的取值不同，分别执行不同的语句。其格式为：

switch　　表达式



```
case     值 1
           语句组 1
case     值 2
           语句组 2
           …   …
case     值 m
           语句组 m
otherwise
           语句组 m+1
end
```

当表达式的值为值 1 时，执行语句组 1，当表达式的值为 2 时，执行语句组 2，……,当表达式值为值 m 时，执行语句组 m，当表达式的值不为 case 所列的值时，执行语句组 m+1。当任一分支的语句执行完成后，直接执行 switch 语句的下一句。执行过程如图 3.2 所示。



[例 3.5]根据变量 num 的值来决定显示的内容。当 num 的值为-1 时，显示：I am a teacher.。当值为 0 时 ，显示：I am a student.。当值为 1 时 ，显示：You are a teacher.。当为其他值时，显示：You are a student.。

解： 程序如下：

```
num=input('请输入一个数');
switch   num
case   -1
   disp('I am a teacher.');
 case    0
   disp('I am a student.');
 case    1
   disp('You are a teacher.');
 otherwise
   disp('You are a student.');
 end
```

3.4 循环结构

循环是指按照给定的条件，重复执行指定的语句，这是十分重要的一种程序结构。MATLAB 提供了两种实现循环结构的语句：for 语句和 while 语句。

3.4.1 for 语句

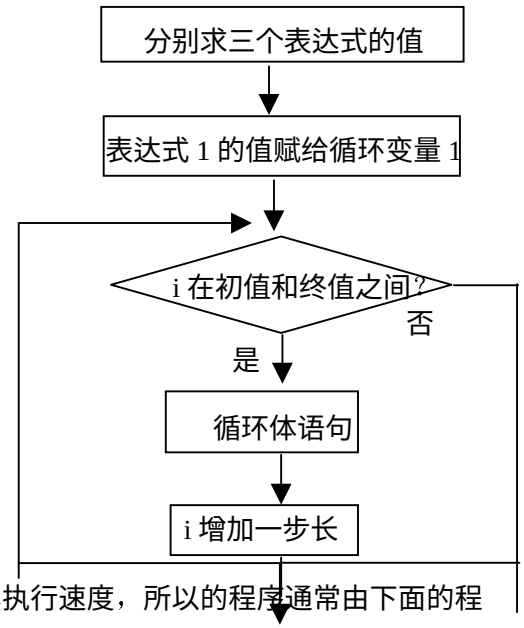
语句格式：

```
for  循环变量=表达式 1：表达式 2：表达式 3
        循环体语句
end
```

其中表达式 1 的值为循环变量的初值，表达式 2 的值为步长，表达式 3 的值为循环变量的终值。步长为 1 时，表达式 2 可以省略。

For 语句的执行过程如图 3.3 所示。首先计算三个表达式的值，再将表达式 1 的值赋给变量，如果此时循环变量的值介于表达式 1 和表达式 3 的值之间，则

执行循环体语句，否则结束循环的执行。执行完一次循环之后，循环变量自增一个表达式 2 的值，然后再判断循环变量的值是否介于表达式 1 和表达式 3 之间，如果满足仍然执行循环体，直至不满足为止。这是将结束 for 语句的执行，而继续执行 for 语句后面的语句。



[例 3.6] 已知 y=1+1/3+1/5+…+1/(2n-1)
当 n=100 时，求时，求 y 的值。

解 程序如下：

```
y=0;n=100;
for i=1:n
y=y+1/(2*I-1);
  end
y
```

在实际 MATLAB 编程中，采用循环语句会降低其执行速度，所以的程序通常由下面的程序来代替：

```
n=100;
    i=1:2:2*n-1;
  y=sum(1./i);
  y
```

在这一程序中，首先生有关向量 i，然后用 sum 函数求 i 向量各个元素的倒数之和。如果前面的 n 值由 100 改成 10000，再分别运行这两个程序，则可以明显的看出，后一种方法编写的程序比前一种快得多。

[例 3.7] 设 f（x）=$e^{-0.5x}$sin（x+pi/6），求 s=$\int_0^{3pi} f(x)dx$

解 以梯形法为例，程序如下：

```
a=0；b=3*pi；n=1000；
h=（b-a）/n；
x=a；s=0；f0=exp（-0.5*x）*sin（x+pi/6）；
for i=1：n
                    x=x+h；
  f1=exp（-0.5*x）*sin（x+pi/6）；
  s=s+（f0+f1）*h/2；
  f0=f1；
end
                s
```

事实上，MATLAB 提供了有关数值积分的标准函数，实际应用中可直接调用这些函数求数值积分（请参阅第 8 章）。

在上述的例子中，for 语句的循环变量都是标量，这与其他高级语言的相关循环语句（如 FORTRAN 语言中的 DO 语句,C 语言中的 for 语句等)等价。按照 MATLAB 的定义，for 语句的循环变量可以是一个列向量。for 语句更一般的形式是：

```
for 循环变量=矩阵表达式
  循环体语句
```

```
end
```

执行过程是依次将矩阵的各列元素赋给循环变量,然后执行循环体语句。

实际上,"表达式 1:表达式 2:表达式 3"是一个仅为一行的矩阵(行向量),因而列向量是单个数据。所以本节一开始给出的 for 语句格式是一种特例。

[例 3.8]写出下列程序的执行结果:

```
s=[0; 0; 0; 0];
a=[12, 13, 14;15, 16, 17;18, 19, 20;21, 22, 23];
for k=a
                    s=s+k;
end
                disp（s’）;
```

解　执行结果是:

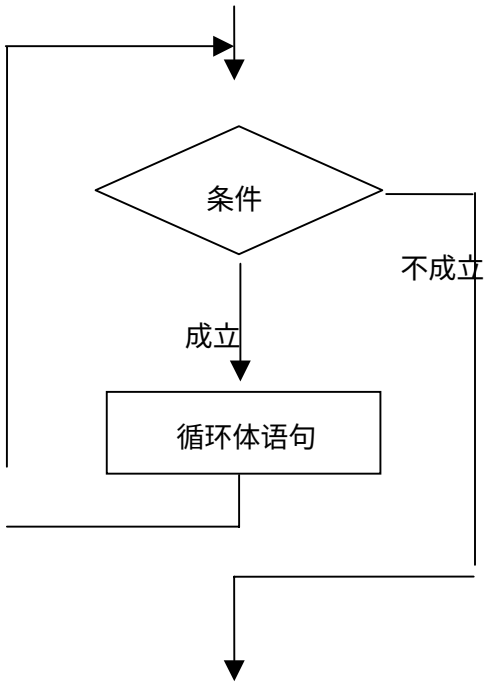
```
 39    48    57    66
```

3.4.2　while 语句

while 语句的一般格式为:

```
while（条件）
      循环体语句
end
```

其执行过程为：若条件成立，则执行循环体语句，执行后再判断条件是否成立，如果不成立则跳出循环（如图 3.4 所示）。



[例 3.9] y=1+1/3+1/5+…+1/(2n-1)

求:

（1）y<3 时的最大 n 的值;

（2）与（1）的 n 值对应的 y 值。

解　程序如下:

```
y=0;i=1;
while 1
        f=1/(2*i-1);
       y=y+f;
       if y>3
                break;
       end
       i=i+1;
end
n=i-1
y=y-f
```

运行结果是:

```
n=
                56
          y=2.9944
```

在程序中，循环的条件为 1，即循环条件总是满足的，这是一个永真循环。为了使循环能正常结束，在循环体中加了一个 if 语句，当 y>3 时，执行 break 命令，从而跳出 for 循环。

```
%  BREAK Terminate execution of WHILE or FOR loop.
     BREAK terminates the execution of FOR and WHILE loops.
     In nested loops, BREAK exits from the innermost loop only
```

[例 3.10]根据矩阵指数函数的幂级数展开式求矩阵指数:

$e^{A}=I+A+1/2!\ A^{2}+1/3!\ A^{3}+\cdots\cdots+1/n!\ A^{n}+\cdots\cdots$

解　设 A 是给定的矩阵，E 是矩阵指数函数值，F 是展开式的项，K 是项数，循环一直进行到 F 很小，以至于 F 值加在 E 上并不改变 E 的值时为止。
程序如下：

```
A=rand(3);
E=zeros(size(A));
F=eye(size(A));
K=1;
while norm(E+F-E,1)>0
        E=E+F;
        F=A*F/K;
        K=K+1;
end
E
expm(A)
```

程序中调用了求矩阵范数的标准函数 norm。程序输出结果是：

```
E=
    1.5913    1.5455    1.1371
    0.5508    3.0230    1.5902
    0.9058    1.0966    1.5804
ans=
    1.5913    1.5455    1.1371
    0.5508    3.0230    1.5902
    0.9058    1.0966    1.5804
```

运行结果表明，程序运行结果与 MATLAB 矩阵函数 expm（A）的结果一致。

3.4.3　循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。实现多重循环结构仍是利用前面介绍的 while 语句和 for 语句。因为任一循环语句的循环体部分都可以包含另一个循环语句，这种循环语句的嵌套为实现多重循环提供了方便。

多重循环的嵌套层数可以是任意的。可以按照嵌套层数，分别叫做二重循环，三重循环等。处于内部的循环叫作内循环，处于外部的循环叫作外循环。

在设计多重循环时，要特别注意内，外循环之间的关系，以及各语句放置的位置，不要搞错。

[例 3.11]求[100，1000]以内的全部素数。

解　素数是大于 1 且除了 1 和它本身以外不能被其他任何整数所整除的整数。为了判断整数 m 是否为素数，一个最简单的办法用 2,3,4,5,……，m-1 这些数逐个去除 m，看能否除尽，如果全都除不尽，则 m 是素数，否则，只要其中一个能除尽，则 m 不是素数。程序如下：

```
n=0;
for m=100:1000
        flag=1;j=m-1;
        i=2;
        while i<=j & flag
                if rem(m,i)==0
                        flag=0;
                end
```

```
                    i=i+1;
                end
                if flag
                    n=n+1;
                    prime(n)=m;
                end
            end
        prime
```

程序中外循环控制 m 的变化，内循环判断每一个 m 是否素数。用变量 n 统计素数的个数，变量 prime 存放素数。

3.5 函数文件

函数文件是另一种形式的 M 文件，每一个函数文件都定义一个函数。事实上，MATLAB 提供的标准函数大部分都是由函数文件定义的。

3.5.1 函数文件格式

函数文件由 function 语句引导，其一般格式为：

function 输出形参表=函数名（输入形参表）

注释说明部分

函数体

其中：函数名的命名规则与变量名相同；输入形参为函数的输入参数；输出形参为函数的输出参数，当输出形参多于 1 个时，则应该用方括号括起来。

函数文件名通常由函数名再加上扩展名.m 组成，不过函数文件名与函数名也可以不相同。当两者不同时，MATLAB 将忽略函数名而确认函数文件名，因此调用时使用函数文件名。不过最好把文件名和函数名统一，以免出错。

[例 3.12]编写函数文件求小于任意自然数 n 的 Fibonacci(斐波纳契函数)数列各项。

解 Fibonacci 数列定义如下：

f1=1

f2=1

f(n)=f(n-1)+f(n-2)　　（n>2）

函数文件 ffib.m：

```
function  f=ffib（n）
%用于      求 Fibonacci 数列的函数文件
%f=ffib（n）
%2002 年 10 月 29 日编
f=[1，1]；
i=1；
while f（i）+f（i+1）<n
                f (i +2)=f (i)+f (i+1);
                i=i+1;
        end
```

将以上函数文件以文件名 ffib.m 存入 C：MATLAB\BIN 下，然后在 MATLAB 命令窗口输入以下命令，可求小于 2000 的 Fibonacci 数列各项：

```
ffib（2000）
```

输出结果是：

```
ans=
```

```
Columns  1  through  6
     1     1     2     3     5     8
Columns  7  through 12
    13    21    34    55    89   144
Columns  13  through  17
   233   377   610   987  1597
```

采用 help 命令或 lookfor 命令可以显示出注释说明部分的内容，其功能和一般 MATLAB 函数的帮助信息是一致的。例如，利用 help 命令可查询 ffib 函数的注释说明：

help ffib

屏幕显示：

用于求 Fibonacci 数列的函数文件
f=ffib（n）
2002 年 10 月 29 日编
先用 path 命令将 C：\MATLAB\BIN 加到 MATLAB 的搜索路径上：
path（'C：\MATLAB\BIN'，path）

再用 lookfor 命令查询 ffib 函数的注释说明：

lookfor fibonacci

屏幕显示：

ffib.m：%用于求 Fibonacci 数列的函数文件

注意：lookfor 命令只在注释说明的第一行查询指定的关键字。

3.5.2 函数调用

函数文件编制好后，就可调用函数进行计算了。如上面定义 ffib 函数后，调用它求小于 2000 的 Fibonacci 数列各项。

函数调用的一般格式为：

[输出实参表]=函数名（输入实参表）

要注意的是，函数调用时个实参出现的顺序，个数应与函数定义时形参的顺序，个数一致，否则会出错。函数调用时，先将实参传递给相应的形参，从而实现参数传递，然后再执行函数的功能。

[例 3.13] 利用函数文件，实现直角坐标（x，y）与极坐标（v，θ）之间的转换。

解 已知转换公式为：

极坐标的矢径： $\rho = \sqrt{x^2+y^2}$

极坐标的幅角： θ =arctan（y/x）

函数文件 tran.m：function[gama,theta]=tran(x,y)

```
gama=sqrt(x*x+y*y);
theta=atan(y/x);
```

调用 tran.m 的命令文件 main1.m:

```
x=input('Please input x=:');
y=input('Please input y=:');
[gam,the]=tran(x,y);
gam
the
```

在 MATLAB 中，函数可以嵌套调用,即一个函数可以调用别的函数,甚至调用它自身(递归调用)。这从又一个侧面,反应了 MATLAB 函数调用功能的强大。

[例 3.14]利用函数的递归调用,求 n!。

解　递归调用函数文件 factor.m:

```
function    f=factor    (n)
if      n<=1
        f=1;
else
        f=factor (n-1)*n;
end
return;                                                    %返回
```

在命令文件 main2.m 中调用函数文件 factor.m:

```
for    i=1:10
        fac ( i )=factor ( i );
end
fac
```

程序运行结果是:

```
fac=
   Columns    1    through    6
      1          2          6          24          120          720
    Columns    7    through 10
      5040       40320      362880     3628800
```

3.5.3　函数所传递参数的可调性

MATLAB 在函数调用上有一个与众不同之处:函数所传递参数数目的可调性.凭借这一点,一个函数可完成多种功能。

在调用函数时,MATLAB 用两个永久变量 nargin 和 nargout 分别记录调用该函数时的输入实参的个数。只要在函数文件中使用这两个变量,就可以准确地知道该函数文件被调用时的输入输出参数个数,从而决定函数如何进行处理。

[例 3.15]nargin 用法示例。

```
NARGIN  Number of function input arguments.
      Inside the body of a user-defined function, NARGIN returns
      the number of input arguments that were used to call the
      function.

    NARGIN('fun') returns the number of declared inputs for the
    M-file function 'fun'.  The number of arguments is negative if
the function has a variable number of input arguments.
```

函数文件 examp.m:

```
function    fout=charray(a,b,c)
if nargin==1
        fout=a;
else if nargin==2
        fout=a+b;
else if nargin==3
        fout=(a*b*c)/2;
end
```

命令文件 mydemo.m:

```
x=[1:3];y=[1;2;3];
examp(x)
examp(x,y')
examp(x,y,3)
```

执行 mydemo.m 后的输出是:

```
ans=
     1     2     3
ans=
     4     5     6
ans=
    21
```

在命令文件中 mydemo.m 中,三次调用函数文件 examp.m,因输入参数的个数分别是 1 个,2 个,3 个,从而执行不同的操作,返回不同的函数值 。

3.6 全局变量和局部变量

在 MATLAB 中，全局变量用命令 global 定义。函数文件内部的变量事实局部变量，它们与其他函数文件及 MATLAB 工作空间相互隔离。但是，如果在若干函数中都把某一变量定义为全局变量，那么这些函数将公用这一个变量。全局变量的作用域是整个 MATLAB 工作空间，即全程有效。所有的函数都可以对它进行存取和修改。因此，定义全局变量是函数见传递信息的一种手段。

需要指出，在程序设计中，全局变量固然可以带来某些方便，但却破坏了函数对变量的封装，降低了程序的可读性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当程序较大，子程序较多时，全局变量将给程序调试和维护带来不便，故不提倡使用全局变量。如果一定要用全局变量，最好给它起一个能反应变量具本含义的名字，并且一般用大写字母表示，以免和其他变量混淆。

[例 3.16]全局变量应用示例。

先建立函数文件 wadd.m，该函数将输入的参数加权相加：

```
function   f   =wadd(x,y)
           %add   two variable with different   weight.
           global   ALPHA BETA
           f   =   ALPHA *x+BETA *y;
```

在命令窗口中输入：

```
global   ALPHA BETA
         ALPHA=1;
         BETA=2;
         s=wadd(1,2)
```

输入为：

```
s=
     5
```

由于在函数 wadd 和基本工作空间中都把 ALPHA 和 BETA 两个变量定义为全局变量，所以只要在命令窗口中改变 ALPHA 和 BETA 的值，就可改变加权值，而无需修改 wadd.m 文件。

上例只在函数 wadd 和命令窗口中把 ALPHA 和 BETA 变量定义为全局变

量，在实际编程时，可在所有需要调用全局变量的函数里定义全局变量，这样就可实现数据共享。为了在基本工作空间中使用全局变量，也要定义全局变量。

在函数文件里，全局变量的定义语句应放在变量使用以前，为了便于了解所有的全局变量，一般把全局变量的定义语句放在文件的前部。

此文完成于 2002 年 7 月 12 日
王正盛　　暑假 南京