MATLAB 入门教程

1．MATLAB 的基本知识

1-1、基本运算与函数

在 MATLAB 下进行基本数学运算，只需将运算式直接打入提示号（>>）之後，并按入 **Enter** 键即可。例如：

```
>> (5*2+1.3-0.8)*10/25

ans =4.2000
```

MATLAB 会将运算结果直接存入一变数 ans，代表 MATLAB 运算後的答案（Answer）并显示其数值於萤幕上。

小提示： ">>"是 MATLAB 的提示符号（Prompt），但在 PC 中文视窗系统下，由於编码方式不同，此提示符号常会消失不见，但这并不会影响到 MATLAB 的运算结果。

我们也可将上述运算式的结果设定给另一个变数 x：

```
x = (5*2+1.3-0.8)*10^2/25

x = 42
```

此时 MATLAB 会直接显示 x 的值。由上例可知，MATLAB 认识所有一般常用到的加（+）、减（-）、乘（*）、除（/）的数学运算符号，以及幂次运算（^）。

小提示： MATLAB 将所有变数均存成 double 的形式，所以不需经过变数宣告（Variable declaration）。MATLAB 同时也会自动进行记忆体的使用和回收，而不必像 C 语言,必须由使用者一一指定.这些功能使的 MATLAB 易学易用，使用者可专心致力於撰写程式，而不必被软体枝节问题所干扰。

若不想让 MATLAB 每次都显示运算结果，只需在运算式最後加上分号（;）即可，如下例：

```
y = sin(10)*exp(-0.3*4^2);
```

若要显示变数 y 的值，直接键入 y 即可：

```
>>y

y =-0.0045
```

在上例中，sin 是正弦函数，exp 是指数函数，这些都是 MATLAB 常用到的数学函数。

下表即为 MATLAB 常用的基本数学函数及三角函数：

小整理：MATLAB 常用的基本数学函数

abs(x)：纯量的绝对值或向量的长度

angle(z)：复 数 z 的相角(Phase angle)

sqrt(x)：开平方

real(z)：复数 z 的实部

imag(z)：复数 z 的虚 部

conj(z)：复数 z 的共轭复数

round(x)：四舍五入至最近整数

fix(x)：无论正负，舍去小数至最近整数

floor(x)：地板函数，即舍去正小数至最近整数

ceil(x)：天花板函数，即加入正小数至最近整数

rat(x)：将实数 x 化为分数表示

rats(x)：将实数 x 化为多项分数展开

sign(x)：符号函数 (Signum function)。

当 x<0 时，sign(x)=-1；

当 x=0 时，sign(x)=0;

当 x>0 时，sign(x)=1。

> 小整理：MATLAB 常用的三角函数

sin(x)：正弦函数

cos(x)：馀弦函数

tan(x)：正切函数

asin(x)：反正弦函数

acos(x)：反馀弦函数

atan(x)：反正切函数

atan2(x,y)：四象限的反正切函数

sinh(x)：超越正弦函数

cosh(x)：超越馀弦函数

tanh(x)：超越正切函数

asinh(x)：反超越正弦函数

acosh(x)：反超越馀弦函数

atanh(x)：反超越正切函数

变数也可用来存放向量或矩阵，并进行各种运算，如下例的列向量（Row vector）运算：

x = [1 3 5 2];

y = 2*x+1

y = 3 7 11 5

小提示：变数命名的规则

1.第一个字母必须是英文字母 2.字母间不可留空格 3.最多只能有 19 个字母，MATLAB 会忽略多馀字母

我们可以随意更改、增加或删除向量的元素：

y(3) = 2 % 更改第三个元素

y =3 7 2 5

y(6) = 10 % 加入第六个元素

y = 3 7 2 5 0 10

y(4) = [] % 删除第四个元素，

y = 3 7 2 0 10

在上例中，MATLAB 会忽略所有在百分比符号（%）之後的文字，因此百分比之後的文字均可视为程式的注解（Comments）。MATLAB 亦可取出向量的一个元素或一部份来做运算：

x(2)*3+y(4) % 取出 x 的第二个元素和 y 的第四个元素来做运算

ans = 9

y(2:4)-1 % 取出 y 的第二至第四个元素来做运算

ans = 6 1 -1

在上例中，2:4 代表一个由 2、3、4 组成的向量

若对 MATLAB 函数用法有疑问，可随时使用 help 来寻求线上支援（on-line help）：help linspace

小整理：MATLAB 的查询命令

help：用来查询已知命令的用法。例如已知 inv 是用来计算反矩阵，键入 help inv 即可得知有关 inv 命令的用法。（键入 help help 则显示 help 的用法，请试看看！） lookfor：用来寻找未知的命令。例如要寻找计算反矩阵的命令，可键入 lookfor inverse，MATLAB 即会列出所有和关键字 inverse 相关的指令。找到所需的命令後 ，即可用 help 进一步找出其用法。（lookfor 事实上是对所有在搜寻路径下的 M 档案进行关键字对第一注解行的比对，详见後叙。）

将列向量转置（Transpose）後，即可得到行向量（Column vector）：

z = x'

z = 4.0000

5.2000

6.4000

7.6000

8.8000

10.0000

不论是行向量或列向量，我们均可用相同的函数找出其元素个数、最大值、最小值等：

length(z) % z 的元素个数

ans = 6

max(z) % z 的最大值

ans = 10

min(z) % z 的最小值

ans = 4

小整理：适用於向量的常用函数有：

min(x): 向量 x 的元素的最小值

max(x): 向量 x 的元素的最大值

mean(x): 向量 x 的元素的平均值

median(x): 向量 x 的元素的中位数

std(x): 向量 x 的元素的标准差

diff(x): 向量 x 的相邻元素的差

sort(x): 对向量 x 的元素进行排序（Sorting）

length(x): 向量 x 的元素个数

norm(x): 向量 x 的欧氏（Euclidean）长度

sum(x): 向量 x 的元素总和

prod(x): 向量 x 的元素总乘积

cumsum(x): 向量 x 的累计元素总和

cumprod(x): 向量 x 的累计元素总乘积

dot(x, y): 向量 x 和 y 的内 积

cross(x, y): 向量 x 和 y 的外积 （大部份的向量函数也可适用於矩阵，详见下述。）

若要输入矩阵，则必须在每一列结尾加上分号（;），如下例：

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

同样地，我们可以对矩阵进行各种处理：

A(2,3) = 5 % 改变位於第二列，第三行的元素值

A =

1 2 3 4

5 6 5 8

9 10 11 12

B = A(2,1:3) % 取出部份矩阵 B

B = 5 6 5

A = [A B'] % 将 B 转置後以行向量并入 A

A =

1 2 3 4 5

5 6 5 8 6

9 10 11 12 5

A(:, 2) = [] % 删除第二行（：代表所有列）

A =

1 3 4 5

5 5 8 6

9 11 12 5

A = [A; 4 3 2 1] % 加入第四列

A =

1 3 4 5

5 5 8 6

9 11 12 5

4 3 2 1

A([1 4], :) = [] % 删除第一和第四列（：代表所有行）

A =

5 5 8 6

9 11 12 5

这几种矩阵处理的方式可以相互叠代运用，产生各种意想不到的效果，就看各位的巧思和创意。

小提示：在 MATLAB 的内部资料结构中,每一个矩阵都是一个以行为主（Column-oriented ）的阵列（Array）因此对於矩阵元素的存取，我们可用一维或二维的索引（Index）来定址。举例来说，在上述矩阵 A 中，位於第二列、第三行的元素可写为 A(2,3) （二维索引）或 A(6)（一维索引，即将所有直行进行堆叠後的第六个元素）。

此外，若要重新安排矩阵的形状，可用 reshape 命令：

B = reshape(A, 4, 2) % 4 是新矩阵的列数，2 是新矩阵的行数

B =

5 8

9 12

5 6

11 5

小提示： A(:)就是将矩阵 A 每一列堆叠起来，成为一个行向量，而这也是 MATLAB 变数的内部储存方式。以前例而言，reshape(A, 8, 1)和 A(:)同样都会产生一个 8x1 的矩阵。

MATLAB 可在同时执行数个命令，只要以逗号或分号将命令隔开：

x = sin(pi/3); y = x^2; z = y*10,

z =

7.5000

若一个数学运算是太长，可用三个句点将其延伸到下一行：

z = 10*sin(pi/3)* ...

sin(pi/3);

若要检视现存於工作空间（Workspace）的变数，可键入 who：

who

Your variables are:

testfile x

这些是由使用者定义的变数。若要知道这些变数的详细资料，可键入：

whos

Name Size Bytes Class

A 2x4 64 double array

B 4x2 64 double array

ans 1x1 8 double array

x 1x1 8 double array

y 1x1 8 double array

z 1x1 8 double array

Grand total is 20 elements using 160 bytes

使用 clear 可以删除工作空间的变数：

clear A

A

??? Undefined function or variable 'A'.

另外 MATLAB 有些永久常数（Permanent constants），虽然在工作空间中看不 到，但使用者可直接取用，例如：

pi

ans = 3.1416

下表即为 MATLAB 常用到的永久常数。

小整理：MATLAB 的永久常数 i 或 j：基本虚数单位

eps：系统的浮点（Floating-point）精确度

inf：无限大， 例如 1/0 nan 或 NaN：非数值（Not a number） ，例如 0/0

pi：圆周率 p（= 3.1415926...）

realmax：系统所能表示的最大数值

realmin：系统所能表示的最小数值

nargin: 函数的输入引数个数

nargin: 函数的输出引数个数

1-2、重复命令

最简单的重复命令是 for 圈（for-loop），其基本形式为：

for 变数 = 矩阵；

运算式；

end

其中变数的值会被依次设定为矩阵的每一行，来执行介於 for 和 end 之间的运算式。因此，若无意外情况，运算式执行的次数会等於矩阵的行数。

举例来说，下列命令会产生一个长度为 6 的调和数列（Harmonic sequence）：

x = zeros(1,6); % x 是一个 16 的零矩阵

```
for i = 1:6,

x(i) = 1/i;

end
```

在上例中，矩阵 x 最初是一个 16 的零矩阵，在 for　圈中，变数 i 的值依次是 1 到 6，因此矩阵 x 的第 i 个元素的值依次被设为 1/i。我们可用分数来显示此数列：

```
format rat % 使用分数来表示数值

disp(x)

1 1/2 1/3 1/4 1/5 1/6
```

for 圈可以是多层的，下例产生一个 16 的 Hilbert 矩阵 h，其中为於第 i 列、第 j 行的元素为

```
h = zeros(6);

for i = 1:6,

for j = 1:6,

h(i,j) = 1/(i+j-1);

end

end

disp(h)

1 1/2 1/3 1/4 1/5 1/6

1/2 1/3 1/4 1/5 1/6 1/7

1/3 1/4 1/5 1/6 1/7 1/8

1/4 1/5 1/6 1/7 1/8 1/9

1/5 1/6 1/7 1/8 1/9 1/10

1/6 1/7 1/8 1/9 1/10 1/11
```

小提示：预先配置矩阵 在上面的例子，我们使用 zeros 来预先配置（Allocate）了一个适当大小的矩阵。若不预先配置矩阵，程式仍可执行，但此时 MATLAB 需要动态地增加（或减小）矩阵的大小，因而降低程式的执行效率。所以在使用一个矩阵时，若能在事前知道其大小，则最好先使用 zeros 或 ones 等命令来预先配置所需的记忆体（即矩阵）大小。

在下例中，for　圈列出先前产生的 Hilbert 矩阵的每一行的平方和：

```
for i = h,
```

```
disp(norm(i)^2); % 印出每一行的平方和

end
```

1299/871

282/551

650/2343

524/2933

559/4431

831/8801

在上例中，每一次 i 的值就是矩阵 h 的一行，所以写出来的命令特别简洁。

令一个常用到的重复命令是 while　圈，其基本形式为：

```
while 条件式;

运算式;

end
```

也就是说，只要条件示成立，运算式就会一再被执行。例如先前产生调和数列的例子，我们可用 while　圈改写如下：

```
x = zeros(1,6); % x 是一个 16 的零矩阵

i = 1;

while i <= 6,

x(i) = 1/i;

i = i+1;

end

format short
```

1-3、逻辑命令

最简单的逻辑命令是 if, ..., end，其基本形式为：

```
if 条件式;

运算式;
```

```
end

if rand(1,1) > 0.5,

disp('Given random number is greater than 0.5.');

end
```

Given random number is greater than 0.5.

1-4、集合多个命令於一个 M 档案

若要一次执行大量的 MATLAB 命令，可将这些命令存放於一个副档名为 m 的档案，并在 MATLAB 提示号下键入此档案的主档名即可。此种包含 MATLAB 命令的档案都以 m 为副档名，因此通称 M 档案（M-files）。例如一个名为 test.m 的 M 档案，包含一连串的 MATLAB 命令，那麼只要直接键入 test，即可执行其所包含的命令：

```
pwd %  显示现在的目录

ans =

D:\MATLAB5\bin

cd c:\data\mlbook %  进入 test.m 所在的目录

type test.m %  显示 test.m 的内容

% This is my first test M-file.

% Roger Jang, March 3, 1997

fprintf('Start of test.m!\n');

for i = 1:3,

fprintf('i = %d ---> i^3 = %d\n', i, i^3);

end

fprintf('End of test.m!\n');

test %  执行 test.m

Start of test.m!

i = 1 ---> i^3 = 1

i = 2 ---> i^3 = 8

i = 3 ---> i^3 = 27

End of test.m!
```

小提示：第一注解行（H1 help line） test.m 的前两行是注解，可以使程式易於了解与管理。特别要说明的是，第一注解行通常用来简短说明此 M 档案的功能，以便 lookfor 能以关键字比对的方式来找出此 M 档案。举例来说，test.m 的第一注解行包含 test 这个字，因此如果键入 lookfor test，MATLAB 即可列出所有在第一注解行包含 test 的 M 档案，因而 test.m 也会被列名在内。

严格来说，M 档案可再细分为命令集（Scripts）及函数（Functions）。前述的 test.m 即为命令集，其效用和将命令逐一输入完全一样，因此若在命令集可以直接使用工作空间的变数，而且在命令集中设定的变数，也都在工作空间中看得到。函数则需要用到输入引数（Input arguments）和输出引数（Output arguments）来传递资讯，这就像是 C 语言的函数,或是 FORTRAN 语言的副程序（Subroutines）。举例来说，若要计算一个正整数的阶乘（Factorial），我们可以写一个如下的 MATLAB 函数并将之存档於 fact.m：

```
function output = fact(n)

% FACT Calculate factorial of a given positive integer.

output = 1;

for i = 1:n,

output = output*i;

end
```

其中 fact 是函数名，n 是输入引数，output 是输出引数，而 i 则是此函数用到的暂时变数。要使用此函数，直接键入函数名及适当输入引数值即可：

```
y = fact(5)

y = 120
```

（当然，在执行 fact 之前，你必须先进入 fact.m 所在的目录。）在执行 fact(5)时，

MATLAB 会跳入一个下层的暂时工作空间（Temperary workspace），将变数 n 的值设定为 5，然後进行各项函数的内部运算，所有内部运算所产生的变数（包含输入引数 n、暂时变数 i，以及输出引数 output）都存在此暂时工作空间中。运算完毕後，MATLAB 会将最後输出引数 output 的值设定给上层的变数 y，并将清除此暂时工作空间及其所含的所有变数。换句话说，在呼叫函数时，你只能经由输入引数来控制函数的输入，经由输出引数来得到函数的输出，但所有的暂时变数都会随着函数的结束而消失，你并无法得到它们的值。

小提示：有关阶乘函数 前面（及後面）用到的阶乘函数只是纯粹用来说明 MATLAB 的函数观念。若实际要计算一个正整数 n 的阶乘（即 n!）时，可直接写成 prod(1:n)，或是直接呼叫 gamma 函数：gamma(n-1)。

MATLAB 的函数也可以是递 式的（Recursive），也就是说，一个函数可以呼叫它本身。

举例来说，n! = n*(n-1)!，因此前面的阶乘函数可以改成递式的写法：

```
function output = fact(n)
```

```
% FACT Calculate factorial of a given positive integer recursively.

if n == 1, % Terminating condition

output = 1;

return;

end

output = n*fact(n-1);
```

在写一个递函数时，一定要包含结束条件（Terminating condition），否则此函数将会一再呼叫自己，永远不会停止，直到电脑的记忆体被耗尽为止。以上例而言，n==1 即满足结束条件，此时我们直接将 output 设为 1，而不再呼叫此函数本身。

1-5、搜寻路径

在前一节中，test.m 所在的目录是 d:\mlbook。如果不先进入这个目录，MATLAB 就找不到你要执行的 M 档案。如果希望 MATLAB 不论在何处都能执行 test.m，那麼就必须将 d:\mlbook 加入 MATLAB 的搜寻路径（Search path）上。要检视 MATLAB 的搜寻路径，键入 path 即可：

```
path

MATLABPATH

d:\matlab5\toolbox\matlab\general

d:\matlab5\toolbox\matlab\ops

d:\matlab5\toolbox\matlab\lang

d:\matlab5\toolbox\matlab\elmat

d:\matlab5\toolbox\matlab\elfun

d:\matlab5\toolbox\matlab\specfun

d:\matlab5\toolbox\matlab\matfun

d:\matlab5\toolbox\matlab\datafun

d:\matlab5\toolbox\matlab\polyfun

d:\matlab5\toolbox\matlab\funfun

d:\matlab5\toolbox\matlab\sparfun

d:\matlab5\toolbox\matlab\graph2d

d:\matlab5\toolbox\matlab\graph3d
```

d:\matlab5\toolbox\matlab\specgraph

d:\matlab5\toolbox\matlab\graphics

d:\matlab5\toolbox\matlab\uitools

d:\matlab5\toolbox\matlab\strfun

d:\matlab5\toolbox\matlab\iofun

d:\matlab5\toolbox\matlab\timefun

d:\matlab5\toolbox\matlab\datatypes

d:\matlab5\toolbox\matlab\dde

d:\matlab5\toolbox\matlab\demos

d:\matlab5\toolbox\tour

d:\matlab5\toolbox\simulink\simulink

d:\matlab5\toolbox\simulink\blocks

d:\matlab5\toolbox\simulink\simdemos

d:\matlab5\toolbox\simulink\dee

d:\matlab5\toolbox\local

此搜寻路径会依已安装的工具箱（Toolboxes）不同而有所不同。要查询某一命令是在搜寻路径的何处，可用 which 命令：

which expo

d:\matlab5\toolbox\matlab\demos\expo.m

很显然 c:\data\mlbook 并不在 MATLAB 的搜寻路径中，因此 MATLAB 找不到 test.m 这个 M 档案：

which test

c:\data\mlbook\test.m

要将 d:\mlbook 加入 MATLAB 的搜寻路径，还是使用 path 命令：

path(path, 'c:\data\mlbook');

此时 d:\mlbook 已加入 MATLAB 搜寻路径（键入 path 试看看），因此 MATLAB 已经"看"得到

test.m:

which test

c:\data\mlbook\test.m

现在我们就可以直接键入 test，而不必先进入 test.m 所在的目录。

小提示：如何在其启动 MATLAB 时，自动设定所需的搜寻路径？ 如果在每一次启动 MATLAB 後都要设定所需的搜寻路径，将是一件很麻烦的事。有两种方法，可以使 MATLAB 启动後 ，即可载入使用者定义的搜寻路径：

1.MATLAB 的预设搜寻路径是定义在 matlabrc.m（在 c:\matlab 之下，或是其他安装 MATLAB 的主目录下），MATLAB 每次启动後，即自动执行此档案。因此你可以直接修改 matlabrc.m ，以加入新的目录於搜寻路径之中。

2.MATLAB 在执行 matlabrc.m 时，同时也会在预设搜寻路径中寻找 startup.m，若此档案存在，则执行其所含的命令。因此我们可将所有在 MATLAB 启动时必须执行的命令（包含更改搜寻路径的命令），放在此档案中。

每次 MATLAB 遇到一个命令（例如 test）时，其处置程序为：

1.将 test 视为使用者定义的变数。

2.若 test 不是使用者定义的变数，将其视为永久常数 。

3.若 test 不是永久常数，检查其是否为目前工作目录下的 M 档案。

4.若不是，则由搜寻路径寻找是否有 test.m 的档案。

5.若在搜寻路径中找不到，则 MATLAB 会发出哔哔声并印出错误讯息。

以下介绍与 MATLAB 搜寻路径相关的各项命令。

1-6、资料的储存与载入

有些计算旷日废时，那麼我们通常希望能将计算所得的储存在档案中，以便将来可进行其他处理。MATLAB 储存变数的基本命令是 save，在不加任何选项（Options）时，save 会将变数以二进制（Binary）的方式储存至副档名为 mat 的档案，如下述：

save：将工作空间的所有变数储存到名为 matlab.mat 的二进制档案。

save filename：将工作空间的所有变数储存到名为 filename.mat 的二进制档案。 save filename x y z ：将变数 x、y、z 储存到名为 filename.mat 的二进制档案。

以下为使用 save 命令的一个简例：

who % 列出工作空间的变数

Your variables are:

B h j y

ans i x z

save test B y % 将变数 B 与 y 储存至 test.mat

dir % 列出现在目录中的档案

. 2plotxy.doc fact.m simulink.doc test.m ~$1basic.doc

.. 3plotxyz.doc first.doc temp.doc test.mat

1basic.doc book.dot go.m template.doc testfile.dat

delete test.mat % 删除 test.mat

以二进制的方式储存变数，通常档案会比较小，而且在载入时速度较快，但是就无法用普通的文书软体（例如 pe2 或记事本）看到档案内容。若想看到档案内容，则必须加上-ascii 选项，详见下述：

save filename x -ascii：将变数 x 以八位数存到名为 filename 的 ASCII 档案。

Save filename x -ascii -double：将变数 x 以十六位数存到名为 filename 的 ASCII 档案。

另一个选项是-tab，可将同一列相邻的数目以定位键（Tab）隔开。

小提示：二进制和 ASCII 档案的比较 在 save 命令使用-ascii 选项後，会有下列现象:save 命令就不会在档案名称後加上 mat 的副档名。

因此以副档名 mat 结尾的档案通常是 MATLAB 的二进位资料档。

若非有特殊需要，我们应该尽量以二进制方式储存资料。

load 命令可将档案载入以取得储存之变数：

load filename：load 会寻找名称为 filename.mat 的档案，并以二进制格式载入。若找不到 filename.mat，则寻找名称为 filename 的档案，并以 ASCII 格式载入。load filename -ascii：load 会寻找名称为 filename 的档案，并以 ASCII 格式载入。

若以 ASCII 格式载入，则变数名称即为档案名称（但不包含副档名）。若以二进制载入，则可保留原有的变数名称，如下例：

clear all; % 清除工作空间中的变数

x = 1:10;

save testfile.dat x -ascii % 将 x 以 ASCII 格式存至名为 testfile.dat 的档案

load testfile.dat % 载入 testfile.dat

who % 列出工作空间中的变数

Your variables are:

testfile x

注意在上述过程中，由於是以 ASCII 格式储存与载入，所以产生了一个与档案名称相同的变数 testfile，此变数的值和原变数 x 完全相同。

1-7、结束 MATLAB

有三种方法可以结束 MATLAB：

1.键入 exit

2.键入 quit

3.直接关闭 MATLAB 的命令视窗（Command window）

2．数值分析

2．1 微分

diff 函数用以演算一函数的微分项，相关的函数语法有下列 4 个：

diff(f) 传回 f 对预设独立变数的一次微分值

diff(f,'t') 传回 f 对独立变数 t 的一次微分值

diff(f,n) 传回 f 对预设独立变数的 n 次微分值

diff(f,'t',n) 传回 f 对独立变数 t 的 n 次微分值

数值微分函数也是用 diff，因此这个函数是靠输入的引数决定是以数值或是符号微分，如果引数为向量则执行数值微分，如果引数为符号表示式则执行符号微分。

先定义下列三个方程式，接著再演算其微分项：

```
>>S1 = '6*x^3-4*x^2+b*x-5';

>>S2 = 'sin(a)';

>>S3 = '(1 - t^3)/(1 + t^4)';

>>diff(S1)

ans=18*x^2-8*x+b

>>diff(S1,2)

ans= 36*x-8

>>diff(S1,'b')
```

ans= x

>>diff(S2)

ans=

cos(a)

>>diff(S3)

ans=-3*t^2/(1+t^4)-4*(1-t^3)/(1+t^4)^2*t^3

>>simplify(diff(S3))

ans= t^2*(-3+t^4-4*t)/(1+t^4)^2

2．2 积分

int 函数用以演算一函数的积分项， 这个函数要找出一符号式 F 使得 diff(F)=f。如果积

分式的解析式 (analytical form, closed form) 不存在的话或是 MATLAB 无法找到，则 int 传回原输入的符号式。相关的函数语法有下列 4 个：

int(f) 传回 f 对预设独立变数的积分值

int(f,'t') 传回 f 对独立变数 t 的积分值

int(f,a,b) 传回 f 对预设独立变数的积分值，积分区间为[a,b]，a 和 b 为数值式

int(f,'t',a,b) 传回 f 对独立变数 t 的积分值，积分区间为[a,b]，a 和 b 为数值式

int(f,'m','n') 传回 f 对预设变数的积分值，积分区间为[m,n]，m 和 n 为符号式

我们示范几个例子：

>>S1 = '6*x^3-4*x^2+b*x-5';

>>S2 = 'sin(a)';

>>S3 = 'sqrt(x)';

>>int(S1)

ans= 3/2*x^4-4/3*x^3+1/2*b*x^2-5*x

>>int(S2)

ans= -cos(a)

>>int(S3)

ans= 2/3*x^(3/2)

>>int(S3,'a','b')

ans= 2/3*b^(3/2)- 2/3*a^(3/2)

>>int(S3,0.5,0.6)

ans= 2/25*15^(1/2)-1/6*2^(1/2)

>>numeric(int(S3,0.5,0.6)) % 使用 numeric 函数可以计算积分的数值

ans= 0.0741

2．3 求解常微分方程式

MATLAB 解常微分方程式的语法是 dsolve('equation','condition')，其中 equation 代表常微分方程式即 y'=g(x,y)，且须以 Dy 代表一阶微分项 y' D2y 代表二阶微分项 y" ，

condition 则为初始条件。

假设有以下三个一阶常微分方程式和其初始条件

y'=3x2, y(2)=0.5

y'=2.x.cos(y)2, y(0)=0.25

y'=3y+exp(2x), y(0)=3

对应上述常微分方程式的符号运算式为：

>>soln_1 = dsolve('Dy = 3*x^2','y(2)=0.5')

ans= x^3-7.500000000000000

>>ezplot(soln_1,[2,4]) % 看看这个函数的长相

>>soln_2 = dsolve('Dy = 2*x*cos(y)^2','y(0) = pi/4')

ans= atan(x^2+1)

>>soln_3 = dsolve('Dy = 3*y + exp(2*x)',' y(0) = 3')

ans= -exp(2*x)+4*exp(3*x)

2．4 非线性方程式的实根

要求任一方程式的根有三步骤：

先定义方程式。要注意必须将方程式安排成 f(x)=0 的形态，例如一方程式为 sin(x)=3，则该方程式应表示为 f(x)=sin(x)-3。可以 m-file 定义方程式。

代入适当范围的 x, y(x) 值，将该函数的分布图画出，藉以了解该方程式的「长相」。

由图中决定 y(x)在何处附近(x0)与 x 轴相交，以 fzero 的语法 fzero('function',x0) 即可求出在 x0 附近的根，其中 function 是先前已定义的函数名称。如果从函数分布图看出根不只一个，则须再代入另一个在根附近的 x0，再求出下一个根。

以下分别介绍几数个方程式，来说明如何求解它们的根。

例一、方程式为

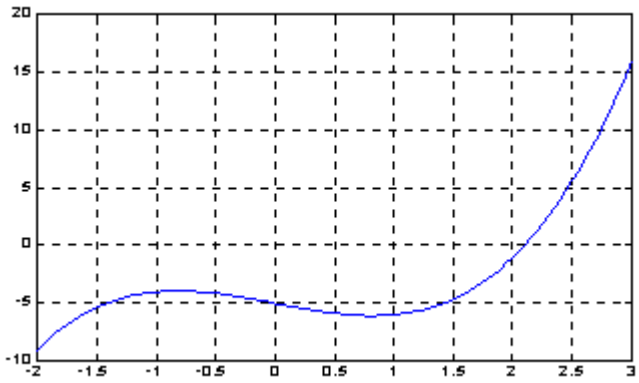
sin(x)=0

我们知道上式的根有 ，求根方式如下：

```
>> r=fzero('sin',3) % 因为 sin(x)是内建函数，其名称为 sin，因此无须定义它,选择 x=3 附近求根

   r=3.1416

>> r=fzero('sin',6) % 选择 x=6 附近求根

r = 6.2832
```

例二、方程式为 MATLAB 内建函数 humps，我们不须要知道这个方程式的形态为何，不过我们可以将它划出来，再找出根的位置。求根方式如下：

```
>> x=linspace(-2,3);

>> y=humps(x);

>> plot(x,y), grid % 由图中可看出在 0 和 1 附近有二个根
```




```
>> r=fzero('humps',1.2)

r = 1.2995
```

例三、方程式为 y=x.^3-2*x-5

这个方程式其实是个多项式，我们说明除了用 roots 函数找出它的根外，也可以用这节介绍的方法求根，注意二者的解法及结果有所不同。求根方式如下：

```
% m-function, f_1.m
```

```
function y=f_1(x) % 定义 f_1.m 函数
y=x.^3-2*x-5;
>> x=linspace(-2,3);
>> y=f_1(x);
>> plot(x,y), grid % 由图中可看出在 2 和-1 附近有二个根
>> r=fzero('f_1',2); % 决定在 2 附近的根
r = 2.0946
>> p=[1 0 -2 -5]
>> r=roots(p) % 以求解多项式根方式验证
r =
2.0946
-1.0473 + 1.1359i
-1.0473 - 1.1359i
```

2．5 线性代数方程（组）求解

我们习惯将上组方程式以矩阵方式表示如下

AX=B

其中 A 为等式左边各方程式的系数项，X 为欲求解的未知项，B 代表等式右边之已知项

要解上述的联立方程式，我们可以利用矩阵左除 \ 做运算，即是 X=A\B。

如果将原方程式改写成 XA=B

其中 A 为等式左边各方程式的系数项，X 为欲求解的未知项，B 代表等式右边之已知项

注意上式的 X, B 已改写成列向量，A 其实是前一个方程式中 A 的转置矩阵。上式的 X 可以矩阵右除 / 求解，即是 X=B/A。

若以反矩阵运算求解 AX=B, X=B，即是 X=inv(A)*B，或是改写成 XA=B, X=B，即是 X=B*inv(A)。

我们直接以下面的例子来说明这三个运算的用法：

```
>> A=[3 2 -1; -1 3 2; 1 -1 -1]; % 将等式的左边系数键入
>> B=[10 5 -1]'; % 将等式右边之已知项键入，B 要做转置
```

>> X=A\B % 先以左除运算求解

X = % 注意 X 为行向量

-2

5

6

>> C=A*X % 验算解是否正确

C = % C=B

10

5

-1

>> A=A'; % 将 A 先做转置

>> B=[10 5 -1];

>> X=B/A % 以右除运算求解的结果亦同

X = % 注意 X 为列向量

10 5 -1

>> X=B*inv(A); % 也可以反矩阵运算求解

3.基本 xy 平面绘图命令

MATLAB 不但擅长於矩阵相关的数值运算，也适合用在各种科学目视表示（Scientific visualization）。

本节将介绍 MATLAB 基本 xy 平面及 xyz 空间的各项绘图命令，包含一维曲线及二维曲面的绘制、列印及存档。

plot 是绘制一维曲线的基本函数，但在使用此函数之前，我们需先定义曲线上每一点的 x 及 y 座标。

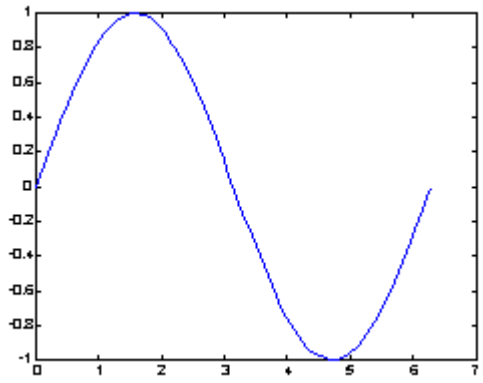
下例可画出一条正弦曲线：

close all;

x=linspace(0, 2*pi, 100); % 100 个点的 x 座标

y=sin(x); % 对应的 y 座标

plot(x,y);



小整理：MATLAB 基本绘图函数

plot: x 轴和 y 轴均为线性刻度（Linear scale）

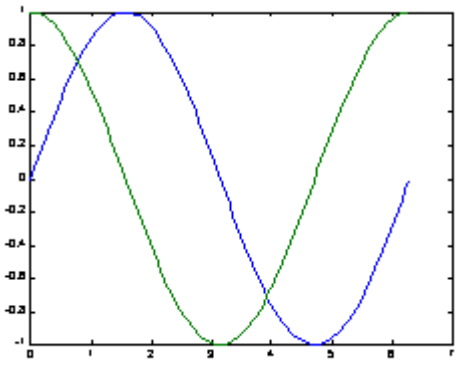
loglog: x 轴和 y 轴均为对数刻度（Logarithmic scale）

semilogx: x 轴为对数刻度，y 轴为线性刻度

semilogy: x 轴为线性刻度，y 轴为对数刻度

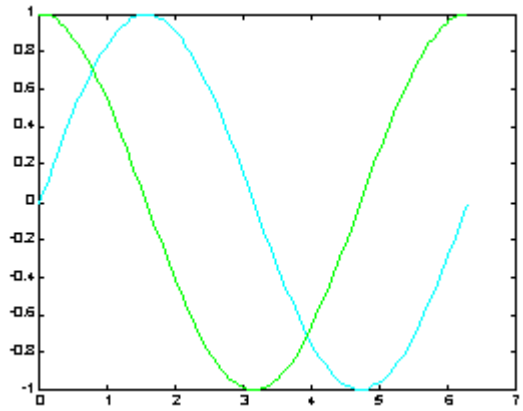
若要画出多条曲线，只需将座标对依次放入 plot 函数即可：

plot(x, sin(x), x, cos(x));



若要改变颜色，在座标对後面加上相关字串即可：

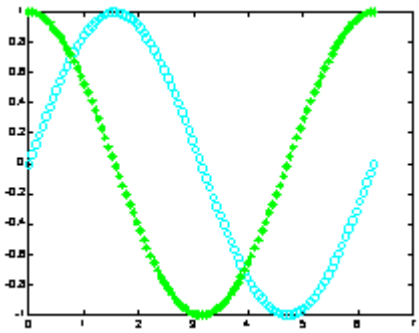
plot(x, sin(x), 'c', x, cos(x), 'g');



<![endif]>

若要同时改变颜色及图线型态（Line style），也是在座标对後面加上相关字串即可：

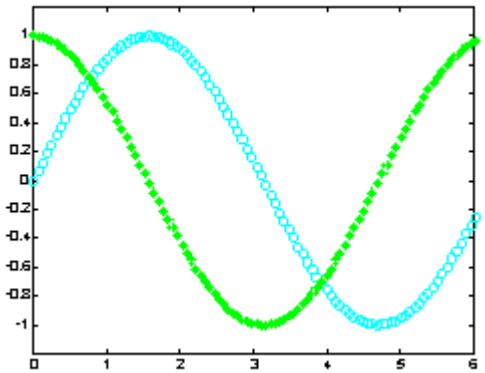
plot(x, sin(x), 'co', x, cos(x), 'g*');



小整理：plot 绘图函数的叁数 字元 颜色字元 图线型态 y 黄色. 点 k 黑色 o 圆 w 白色 x xb 蓝色+ +g 绿色* *r 红色- 实线 c 亮青色: 点线 m 锰紫色-. 点虚线-- 虚线

图形完成後，我们可用 axis([xmin,xmax,ymin,ymax])函数来调整图轴的范围：

axis([0, 6, -1.2, 1.2]);



此外，MATLAB 也可对图形加上各种注解与处理：

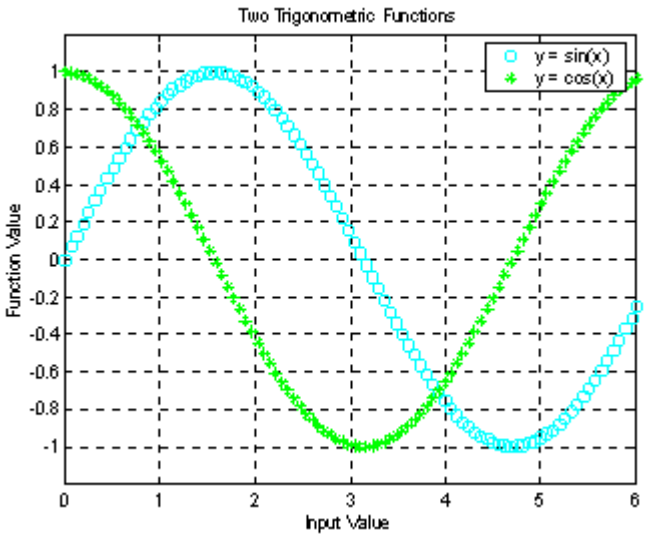
xlabel('Input Value'); % x 轴注解

ylabel('Function Value'); % y 轴注解

title('Two Trigonometric Functions'); % 图形标题

legend('y = sin(x)','y = cos(x)'); % 图形注解

grid on; % 显示格线



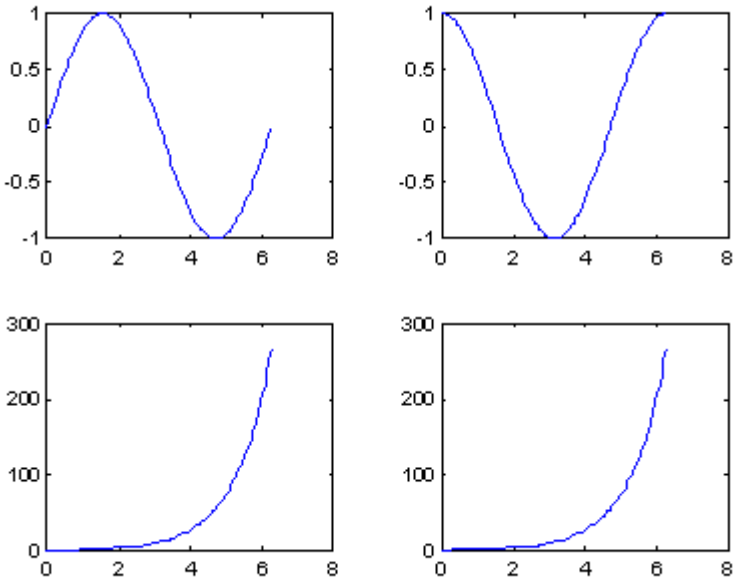
我们可用 subplot 来同时画出数个小图形於同一个视窗之中：

subplot(2,2,1); plot(x, sin(x));

subplot(2,2,2); plot(x, cos(x));

subplot(2,2,3); plot(x, sinh(x));

subplot(2,2,4); plot(x, cosh(x));



MATLAB 还有其他各种二维绘图函数，以适合不同的应用，详见下表。

小整理：其他各种二维绘图函数

bar 长条图

errorbar 图形加上误差范围

fplot 较精确的函数图形

polar 极座标图

hist 累计图

rose 极座标累计图

stairs 阶梯图

stem 针状图

fill 实心图

feather 羽毛图

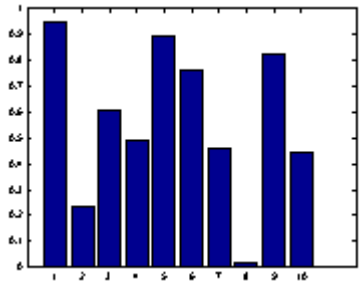
compass 罗盘图

quiver 向量场图

以下我们针对每个函数举例。

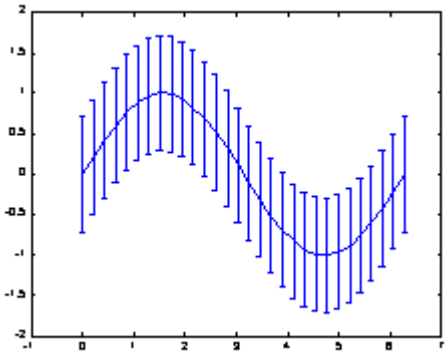
当资料点数量不多时，长条图是很适合的表示方式：

```
close all; %  关闭所有的图形视窗

x=1:10;

y=rand(size(x));

bar(x,y);
```



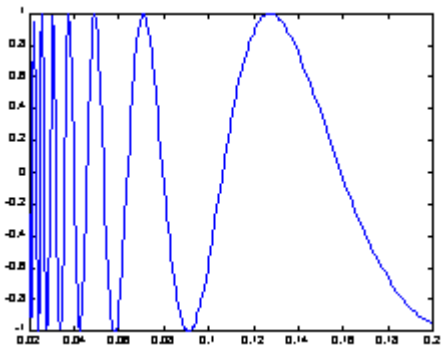
如果已知资料的误差量，就可用 errorbar 来表示。下例以单位标准差来做资的误差量：

```
x = linspace(0,2*pi,30);

y = sin(x);

e = std(y)*ones(size(x));

errorbar(x,y,e)
```



对於变化剧烈的函数，可用 fplot 来进行较精确的绘图，会对剧烈变化处进行较密集的取样，如下例：

```
fplot('sin(1/x)', [0.02 0.2]); % [0.02 0.2]是绘图范围
```



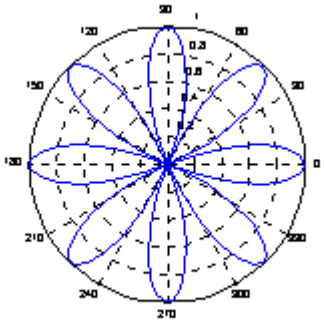
<![endif]>

若要产生极座标图形，可用 polar：

theta=linspace(0, 2*pi);

r=cos(4*theta);

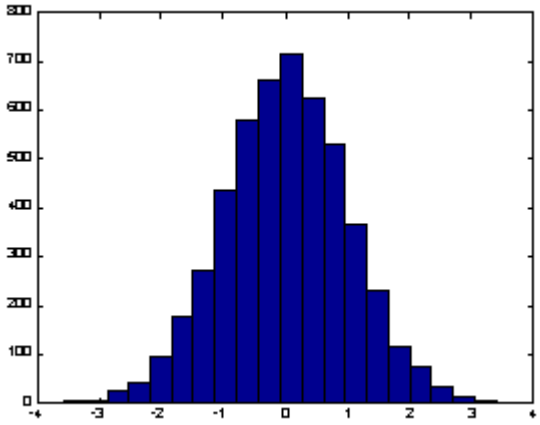
polar(theta, r);



对於大量的资料，我们可用 hist 来显示资料的分　情况和统计特性。下面几个命令可用来验证 randn 产生的高斯乱数分　：

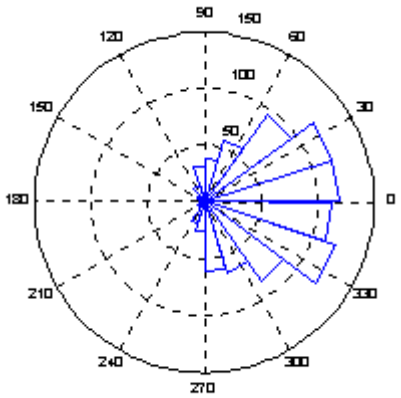
x=randn(5000, 1); %　产生 5000 个　m=0，s=1　的高斯乱数

hist(x,20); % 20 代表长条的个数



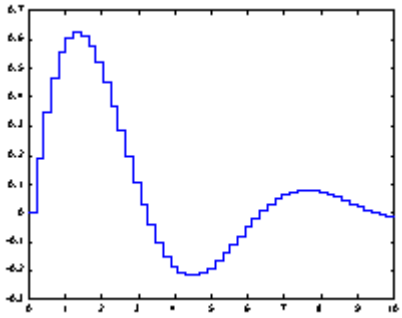
rose 和 hist 很接近，只不过是将资料大小视为角度，资料个数视为距离，并用极座标绘制表示：

```
x=randn(1000, 1);
rose(x);
```



stairs 可画出阶梯图：

```
x=linspace(0,10,50);
y=sin(x).*exp(-x/3);
stairs(x,y);
```



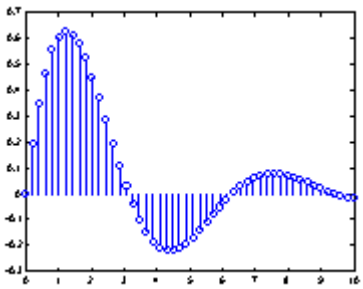
<![endif]>

stems 可产生针状图，常被用来绘制数位讯号：

x=linspace(0,10,50);

y=sin(x).*exp(-x/3);

stem(x,y);

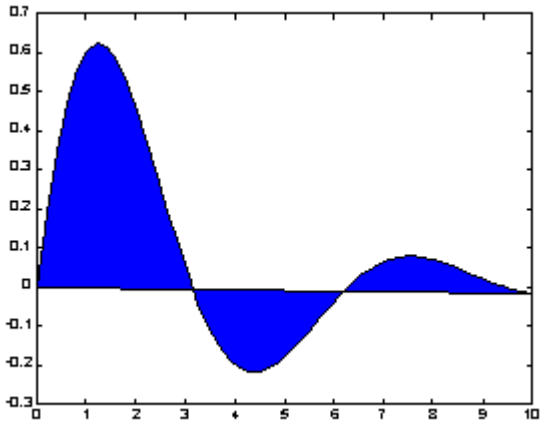


stairs 将资料点视为多边行顶点，并将此多边行涂上颜色：

x=linspace(0,10,50);

y=sin(x).*exp(-x/3);

fill(x,y,'b'); % 'b'为蓝色

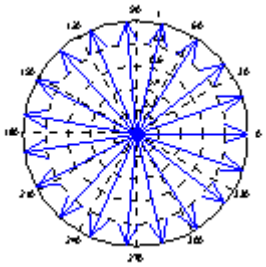


feather 将每一个资料点视复数，并以箭号画出：

```
theta=linspace(0, 2*pi, 20);
z = cos(theta)+i*sin(theta);
feather(z);
```

compass 和 feather 很接近，只是每个箭号的起点都在圆点：

```
theta=linspace(0, 2*pi, 20);
z = cos(theta)+i*sin(theta);
compass(z);
```



<![endif]>

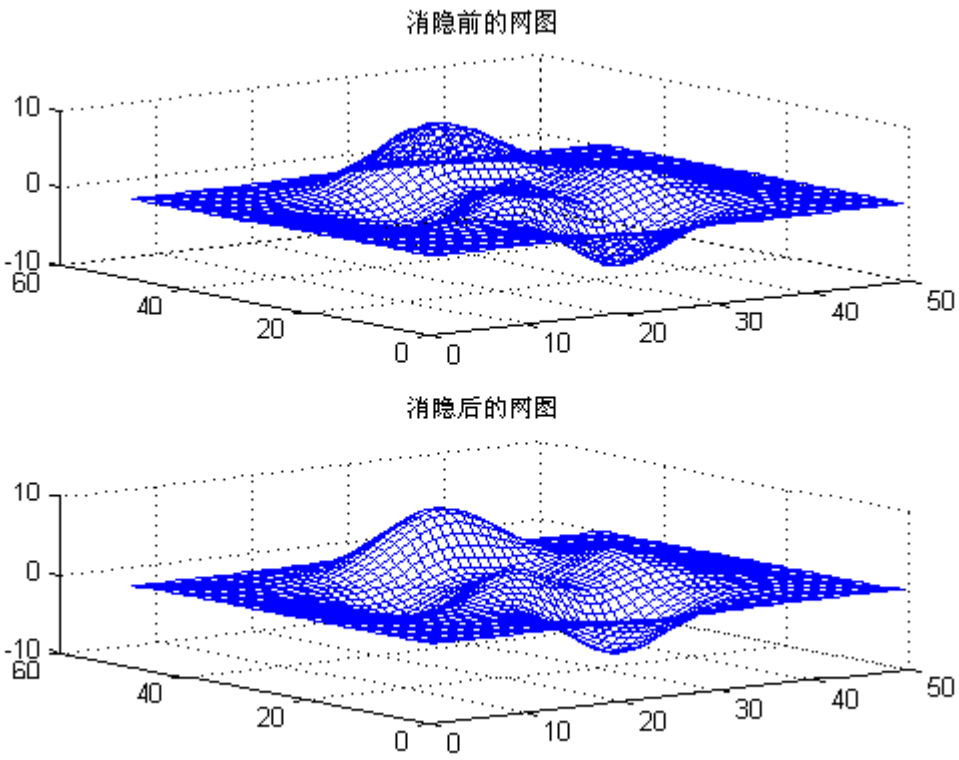
4.三维网图的高级处理

1. 消隐处理

例.比较网图消隐前后的图形

```
z=peaks(50);
subplot(2,1,1);
mesh(z);
title('消隐前的网图')
hidden off
subplot(2,1,2)
mesh(z);
title('消隐后的网图')
hidden on
```

colormap([0 0 1])

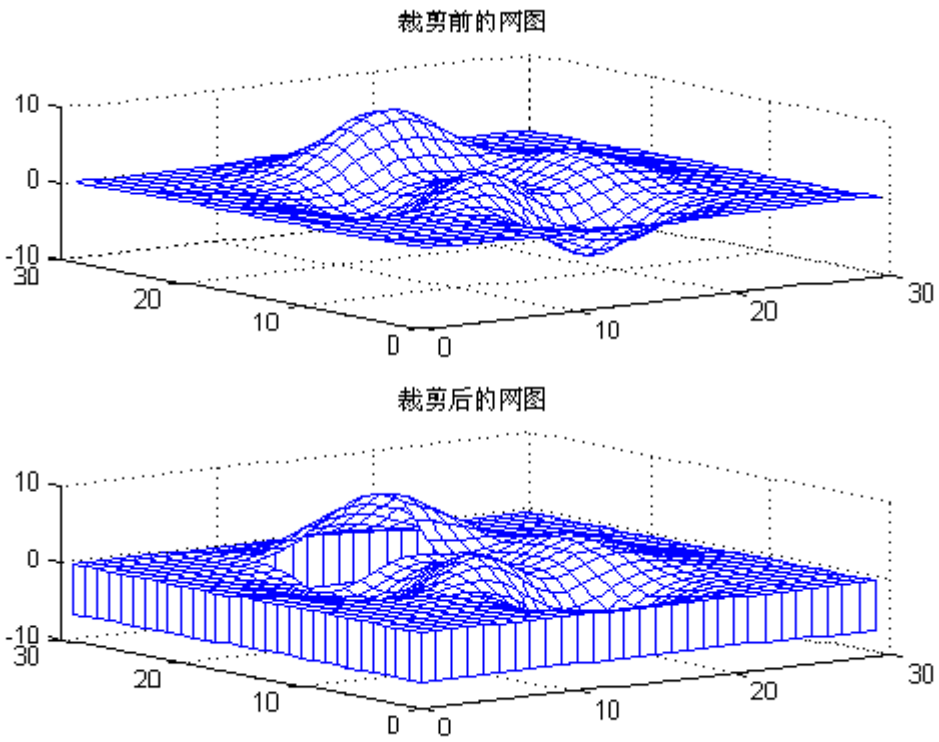


2. 裁剪处理

利用不定数 NaN 的特点,可以对网图进行裁剪处理

例.图形裁剪处理

```
P=peaks(30);
subplot(2,1,1);
mesh(P);
title('裁剪前的网图')
subplot(2,1,2);
P(20:23,9:15)=NaN*ones(4,7);        %剪孔
meshz(P)                            %垂帘网线图
title('裁剪后的网图')
colormap([0 0 1])                   %蓝色网线
```



注意裁剪时矩阵的对应关系,即大小一定要相同.

3. 三维旋转体的绘制

为了一些专业用户可以更方便地绘制出三维旋转体,MATLAB 专门提供了 2 个函数:柱面函数 cylinder 和球面函数 sphere

(1) 柱面图

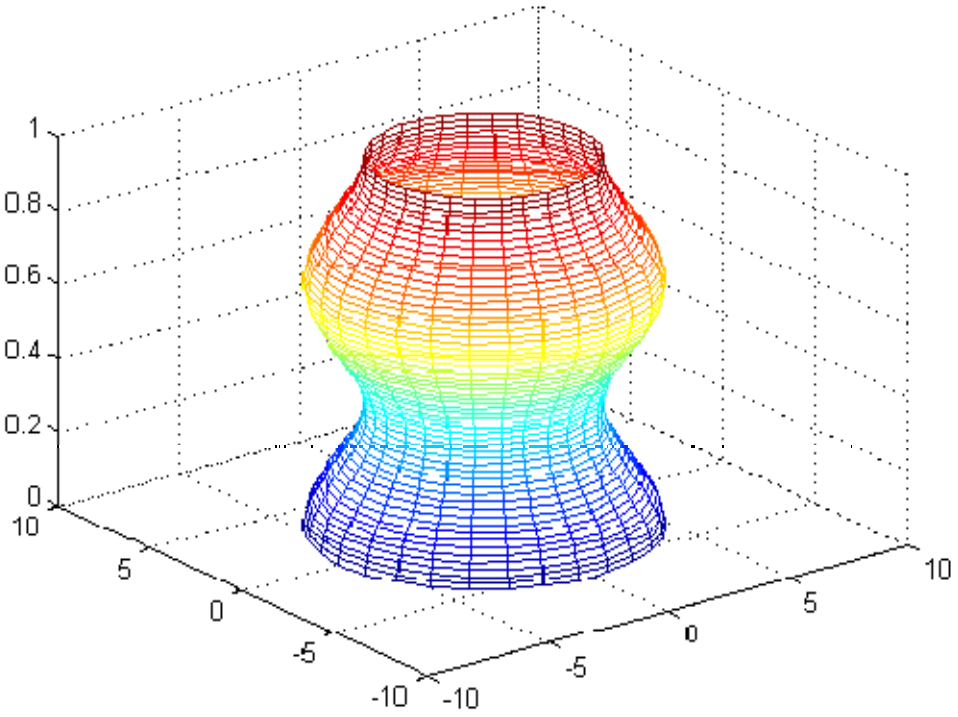
柱面图绘制由函数 cylinder 实现.

[X,Y,Z]=cylinder(R,N) 此函数以母线向量 R 生成单位柱面.母线向量 R 是在单位高度里等分刻度上定义的半径向量.N 为旋转圆周上的分格线的条数.可以用 surf(X,Y,Z)来表示此柱面.

[X,Y,Z]=cylinder(R)或[X,Y,Z]=cylinder 此形式为默认 N=20 且 R=[1 1]

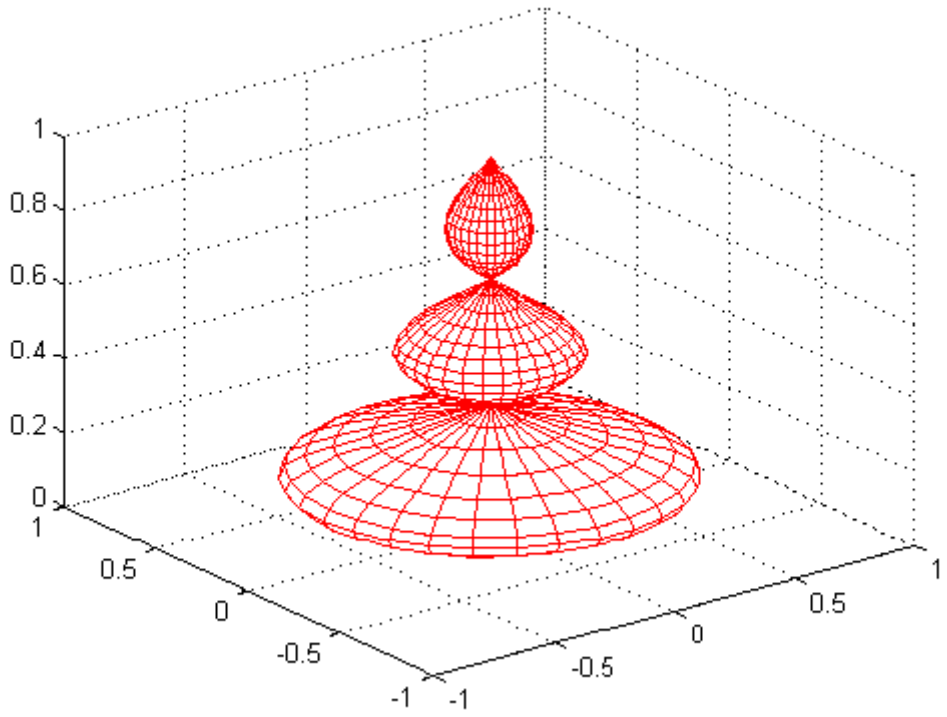
例.柱面函数演示举例

```
x=0:pi/20:pi*3;

r=5+cos(x);

[a,b,c]=cylinder(r,30);

mesh(a,b,c)
```



例.旋转柱面图.

```
r=abs(exp(-0.25*t).*sin(t));

t=0:pi/12:3*pi;

r=abs(exp(-0.25*t).*sin(t));

[X,Y,Z]=cylinder(r,30);

mesh(X,Y,Z)

colormap([1 0 0])
```



(2).球面图

球面图绘制由函数 sphere 来实现

[X,Y,Z]=sphere(N)　　此函数生成 3 个(N+1)*(N+1)的矩阵,利用函数　　surf(X,Y,Z) 可产生单位球面.

[X,Y,Z]=sphere　　此形式使用了默认值 N=20.

Sphere(N)　　只是绘制了球面图而不返回任何值.

例.绘制地球表面的气温分布示意图.

```
[a,b,c]=sphere(40);

t=abs(c);

surf(a,b,c,t);

axis('equal')    %此两句控制坐标轴的大小相同.

axis('square')

colormap('hot')
```