MATLAB快速入门学习教程

科学计算可分为两类 ：一类是纯数值的计算 ，例如求函数的值 ，方程的数值解 ；另一类计算是符号计算，又称代数运算 ，这是一种智能化的计算 ，处理的是符号 . 符号可以代表整数 ，有理数, 实数和复数，也可以代表多项式 ，函数, 还可以代表数学结构如集合 ，群的表示等等 . 我们在数学的教学和研究中用笔和纸进行的数学运算多为符号运算 .

我们知道，从计算机发明到现在的 50 多年时间里，用计算机进行的科学计算主要是数值计算, 如天气预报，油藏模拟，航天等领域的大规模数值计算 . 长期以来，人们一直盼望有一个可以进行符号计算的计算机系统 . 早在 50 年代末，人们就开始了研究 . 进入 80 年代后，随着计算机的普及和人工智能的发展 ，用计算机进行代数运算的研究在国外发展非常迅速 ，涉及的数学领域也在不断地扩大 ，相继出现了多种功能齐全的计算机代数系统 ，这些系统可以分为专用系统和通用系统，专用系统主要是为解决物理 ，数学和其他科学分支的某些计算问题而设计的 ，专用系统在符号和数据结构上都适用于相应的领域 ，而且多数是用低级语言写成的 ，使用方便，计算速度快，在专业问题的研究中起着重要的作用 . 通用系统具有多种数据结构和丰富的数学函数, 应用领域广泛 . 其中, REDUCE,RERIVE,Mathematica 和 Maple 是用户教为广泛的通用数学软件 . 最近的计算机代数系统都是用 C 语言写成的，这种语言为软件开发者提供了编写有效的可移植的计算机程序的平台 ，所以这种计算机代数系统可以在绝大多数计算机上使用. Mathematica 和 Maple 就是这样的系统 . Mathematica 是第一个将符号运算 ，数值计算和图形显示很好地结合在一起的数学软件 ，用户能够方便地用它进行多种形式的数学处理. Maple 是 80 年代初就开始研制的计算机代数系统 ，起初并不为人们所注意 ，但这个软件发展很快 . 自从 1992 年 MapleVR2 出版后，更多的用户就发现它是一个功能强大而且界面友好的计算机代数系统 .

尽管不同的数学软件之间有较大的差别 ，但也有一些共同的特点 ：

1. 可以进行符号运算 ，数值计算和图形显示 ，这是通用数学软件包的三大基本功能 . 具有高效的可编程功能 .

2. 多数计算机代数系统都是交互式的 ，人们通过键盘输入命令 ，计算机计算后显示结果 . 好的系统都有 Windows 操作系统下的版本 ，人机界面友好，命令输入方便灵活 ，很容易寻求帮助. 结果的输出有多种形式 ，好的数学软件都提供了人们习惯的数学符号表达形式 .

3. 各个系统都在不断地发展完善 ，不断地更新换代 ，更新的速冻也在逐渐加快 . 数学软件在向着智能化，自动化方向发展 . 数学软件的实质是数学方法及其算法在计算机上的实现 ，这些方法是千百年来无数数学家的工作与智慧的结晶 .

4. 参与软件开发和应用的人员的数量在不断增加 ，而且日趋国际化 . 随着 Internet 的普及，软件用户可以很方便地与软件开发者进行沟通 ，反映软件中存在的问题 ，也把新的应用情况和好的程序提供给软件的开发者 . 软件的开发不再只是软件开发者的事情 ，也是广大用户的事情.

计算机代数系统的优越性主要在于它能够进行大规模的代数运算 . 通常我们用笔和纸进行代数运算只能处理符号较少的算式 ，当算式的符号上升到百位数后 ，手工计算便成为可能而不可行的事，主要原因是在做大量符号运算时 ，我们很容易出错 ，并且缺乏足够的耐心 . 当算式

的符号个数上升到四位数后 ，手工计算便成为不可能的事 ，这时用计算机代数系统进行运算就可以做到准确 ，快捷，有效.

尽管计算机代数系统在代替人进行繁琐的符号运算上有着无比的优越性 ，但是，计算机毕竟是机器，它只能执行人们给它的指令 .数学软件都有一定的局限性 .首先，多数计算机代数系统对计算机硬件有较高的要求 ，在进行符号运算时 ，通常需要很大的内存和较长的计算时间 ，而精确的代数运算以时间和空间为代价的 .一些人工计算的简单问题 ，计算机代数系统却做不出来.用数学软件的第二个问题是计算结果往往很长 ，人们很难从结果中看到问题的要害 .用计算机代数系统进行数值计算 ，虽然计算精度可以到任意位 ，但由于计算机代数系统是用软件本身浮点运算代替硬件算术运算 ，所以在速度要比用 Fortran 语言算同样的问题慢百倍甚至千倍.另外，虽然计算机代数系统包含大量的数学知识 ，但这仅仅是数学的一小部分 ，目前有许多数学领域计算机代数系统还未能涉及 .

MATLAB简介

http://www.mathworks.com/

1. MATLAB的概况

MATLAB是矩阵实验室 （Matrix Laboratory ）之意。除具备卓越的数值计算能力外，它还提供了专业水平的符号计算，文字处理，可视化建模仿真和实时控制等功能。

MATLAB的基本数据单位是矩阵， 它的指令表达式与数学 ，工程中常用的形式十分相似 ，故用 MATLAB来解算问题要比用 C,FORTRAN等语言完相同的事情简捷得多 .

当前流行的 MATLAB 5.3/Simulink 3.0 包括拥有数百个内部函数的主包和三十几种工具包(Toolbox). 工具包又可以分为功能性工具包和学科工具包 .功能工具包用来扩充 MATLAB的符号计算，可视化建模仿真 ，文字处理及实时控制等功能 .学科工具包是专业性比较强的工具包，控制工具包，信号处理工具包 ，通信工具包等都属于此类 .

开放性使 MATLAB广受用户欢迎 .除内部函数外，所有 MATLAB主包文件和各种工具包都是可读可修改的文件 ，用户通过对源程序的修改或加入自己编写程序构造新的专用工具包 .

2. MATLAB产生的历史背景

在 70 年代中期 ,Cleve Moler 博士和其同事在美国国家科学基金的资助下开发了调用 EISPACK和 LINPACK的 FORTRAN子程序库 .EISPACK是特征值求解的 FOETRAN程序库 ,LINPACK是解线性方程的程序库 .在当时，这两个程序库代表矩阵运算的最高水平 .

到 70 年代后期，身为美国 NewMexico 大学计算机系系主任的 Cleve Moler, 在给学生讲授线性代数课程时 ，想教学生使用 EISPACK和 LINPACK程序库，但他发现学生用 FORTRAN编写接口程序很费时间 ，于是他开始自己动手 ，利用业余时间为学生编写 EISPACK和 LINPACK的接口程序 .Cleve Moler 给这个接口程序取名为 MATLAB,该名为矩阵 (matrix) 和实验室 (labotatory) 两个英文单词的前三个字母的组合 .在以后的数年里 ,MATLAB在多所大学里作为教学辅助软件使用 ，并作为面向大众的免费软件广为流传 .

1983 年春天 ,Cleve Moler 到 Standford 大学讲学 ,MATLAB 深深地吸引了工程师 John Little.John Little 敏锐地觉察到 MATLAB 在工程领域的广阔前景 . 同年 , 他和 Cleve Moler,Steve Bangert 一起 , 用 C 语言开发了第二代专业版 . 这一代的 MATLAB 语言同时具备了数值计算和数据图示化的功能 .

1984 年 ,Cleve Moler 和 John Little 成立了 Math Works 公司 , 正式把 MATLAB 推向市场 , 并继续进行 MATLAB 的研究和开发 .

在当今 30 多个数学类科技应用软件中 , 就软件数学处理的原始内核而言 , 可分为两大类 . 一类是数值计算型软件 , 如 MATLAB,Xmath,Gauss 等 , 这类软件长于数值计算 , 对处理大批数据效率高 ; 另一类是数学分析型软件 ,Mathematica,Maple 等 , 这类软件以符号计算见长 , 能给出解析解和任意精确解 , 其缺点是处理大量数据时效率较低 .MathWorks 公司顺应多功能需求之潮流 , 在其卓越数值计算和图示能力的基础上 , 又率先在专业水平上开拓了其符号计算 , 文字处理 , 可视化建模和实时控制能力 , 开发了适合多学科 , 多部门要求的新一代科技应用软件 MATLAB 经过多年的国际竞争 ,MATLAB 以经占据了数值软件市场的主导地位 .

在 MATLAB 进入市场前，国际上的许多软件包都是直接以 FORTRAN 语言等编程语言开发的。这种软件的缺点是使用面窄，接口简陋，程序结构不开放以及没有标准的基库，很难适应各学科的最新发展，因而很难推广。 MATLAB 的出现，为各国科学家开发学科软件提供了新的基础。在 MATLAB 问世不久的 80 年代中期，原先控制领域里的一些软件包纷纷被淘汰或在 MATLAB 上重建。

MathWorks公司 1993年推出了 MATLAB 4。0 版， 1995 年推出 4。2C版（for win3 。X） 1997 年推出 5。0 版。 1999年推出 5。3 版。 MATLAB 5。X较 MATLAB 4。X无论是界面还是内容都有长足的进展，其帮助信息采用超文本格式和 PDF格式，在 Netscape 3 。0 或 IE 4 。0 及以上版本， Acrobat Reader 中可以方便地浏览。

时至今日，经过 MathWorks公司的不断完善， MATLAB 已经发展成为适合多学科，多种工作平台的功能强大大大型软件。 在国外，MATLAB 已经经受了多年考验。 在欧美等高校，MATLAB 已经成为线性代数，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 被广泛用于科学研究和解决各种具体问题。 在国内，特别是工程界， MATLAB 一定会盛行起来。可以说，无论你从事工程方面的哪个学科，都能在 MATLAB 里找到合适的功能。

2. MATLAB 的语言特点

一种语言之所以能如此迅速地普及，显示出如此旺盛的生命力，是由于它有着不同于其他语言的特点，正如同 FORTRAN 和 C等高级语言使人们摆脱了需要直接对计算机硬件资源进行操作一样，被称作为第四代计算机语言的 MATLAB 利用其丰富的函数资源，使编程人员从繁琐的程序代码中解放出来。 MATLAB 最突出的特点就是简洁。 MATLAB 用更直观的， 符合人们思维习惯的代码，代替了 C和 FORTRAN 语言的冗长代码。 MATLAB 给用户带来的是最直观，最简洁的程序开发环境。以下简单介绍一下 MATLAB 的主要特点。

1）。语言简洁紧凑，使用方便灵活，库函数极其丰富。 MATLAB 程序书写形式自由，利用起丰富的库函数避开繁杂的子程序编程任务，压缩了一切不必要的编程工作。由于库函数都

由本领域的专家编写，　用户不必担心函数的可靠性。　可以说，用 MATLAB 进行科技开发是站在专家的肩膀上。

具有 FORTRAN 和 C 等高级语言知识的读者可能已经注意到，如果用 FORTRAN 或 C 语言去编写程序，尤其当涉及矩阵运算和画图时，编程会很麻烦。例如，如果用户想求解一个线性代数方程，就得编写一个程序块读入数据，然后再使用一种求解线性方程的算法（例如追赶法）编写一个程序块来求解方程，最后再输出计算结果。在求解过程中，最麻烦的要算第二部分。解线性方程的麻烦在于要对矩阵的元素作循环，选择稳定的算法以及代码的调试动不容易。即使有部分源代码，用户也会感到麻烦，且不能保证运算的稳定性。解线性方程的程序用 FORTRAN 和 C 这样的高级语言编写，至少需要四百多行，调试这种几百行的计算程序可以说很困难。以下用　MATLAB 编写以上两个小程序的具体过程。

MATLAB 求解下列方程，并求解矩阵　A 的特征值。

Ax=b, 其中：

A= 32　13　45　67

　23　79　85　12

　43　23　54　65

　98　34　71　35

b=　1

　2

　3

　4

解为：x=A\b; 设 A 的特征值组成的向量　e，e=eig （A）。

可见，MATLAB 的程序极其简短。更为难能可贵的是，MATLAB 甚至具有一定的智能水平，比如上面的解方程，　MATLAB 会根据矩阵的特性选择方程的求解方法，所以用户根本不用怀疑 MATLAB 的准确性。

2）运算符丰富。由于　MATLAB 是用 C 语言编写的，　MATLAB 提供了和 C 语言几乎一样多的运算符，灵活使用　MATLAB 的运算符将使程序变得极为简短。

3）MATLAB 既具有结构化的控制语句　（如 for　循环，while　循环，break 语句和 if　语句），又有面向对象编程的特性。

4）程序限制不严格，程序设计自由度大。例如，在　MATLAB 里，用户无需对矩阵预定义就可使用。

5）程序的可移植性很好，　基本上不做修改就可以在各种型号的计算机和操作系统上运行。

6）MATLAB的图形功能强大。在 FORTRAN和 C语言里，绘图都很不容易，但在 MATLAB里，数据的可视化非常简单。 MATLAB还具有较强的编辑图形界面的能力。

7）MATLAB的缺点是，它和其他高级程序相比，程序的执行速度较慢。由于 MATLAB的程序不用编译等预处理，也不生成可执行文件，程序为解释执行，所以速度较慢。

8）功能强大的工具箱是 MATLAB的另一特色。 MATLAB包含两个部分：核心部分和各种可选的工具箱。核心部分中有数百个核心内部函数。其工具箱又分为两类：功能性工具箱和学科性工具箱。功能性工具箱主要用来扩充其符号计算功能，图示建模仿真功能，文字处理功能以及与硬件实时交互功能。功能性工具箱用于多种学科。而学科性工具箱是专业性比较强的，如 control,toolbox,signl proceessing toolbox,commumnication toolbox 等。这些工具箱都是由该领域内学术水平很高的专家编写的，所以用户无需编写自己学科范围内的基础程序，而直接进行高，精，尖的研究。

9）源程序的开放性。开放性也许是 MATLAB最受人们欢迎的特点。除内部函数以外，所有 MATLAB的核心文件和工具箱文件都是可读可改的源文件， 用户可通过对源文件的修改以及加入自己的文件构成新的工具箱。

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 （ 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 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 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

```
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

```
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 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）

MATLAB入门教程

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

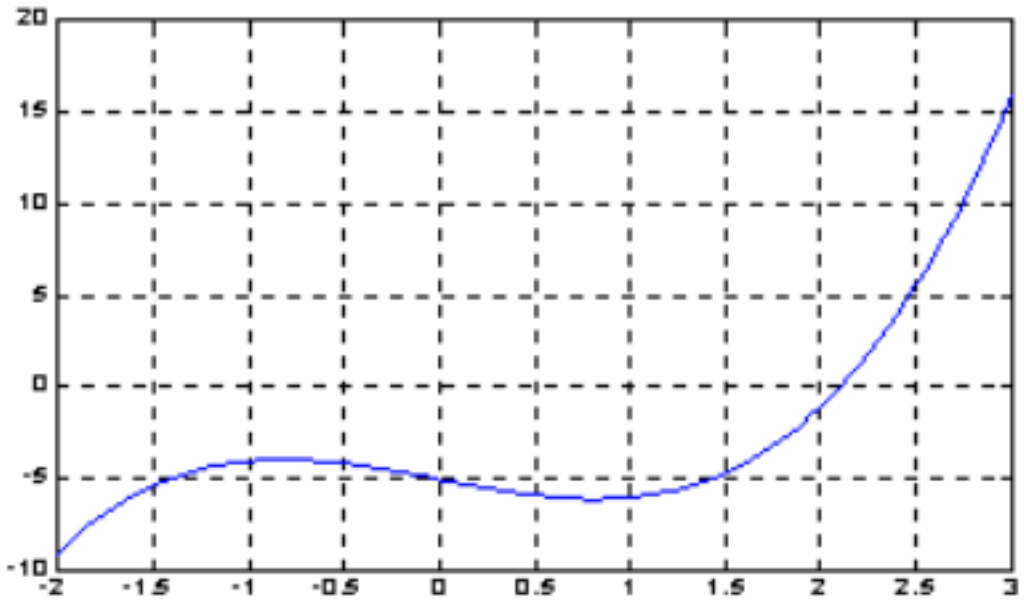
我们知道上式的根有 ， 求根方式如下：

```
>> 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。

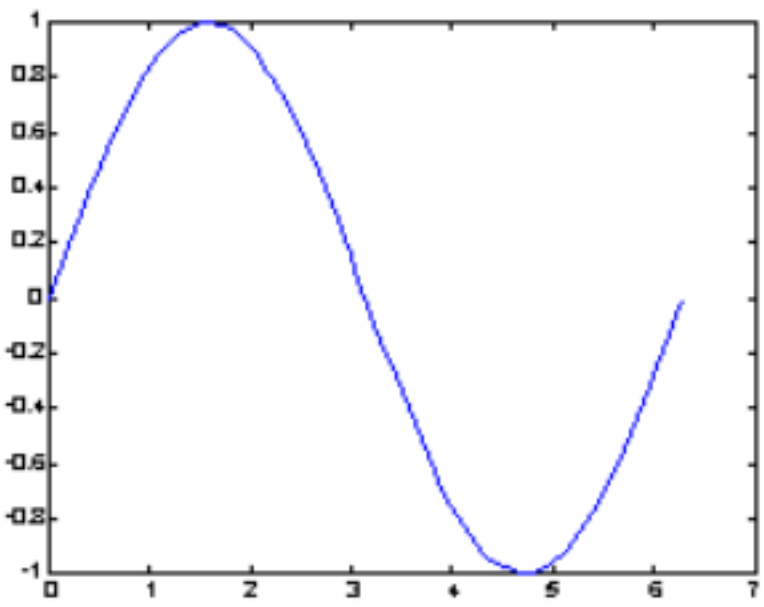
下例可画出一条正弦曲线：

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 ）

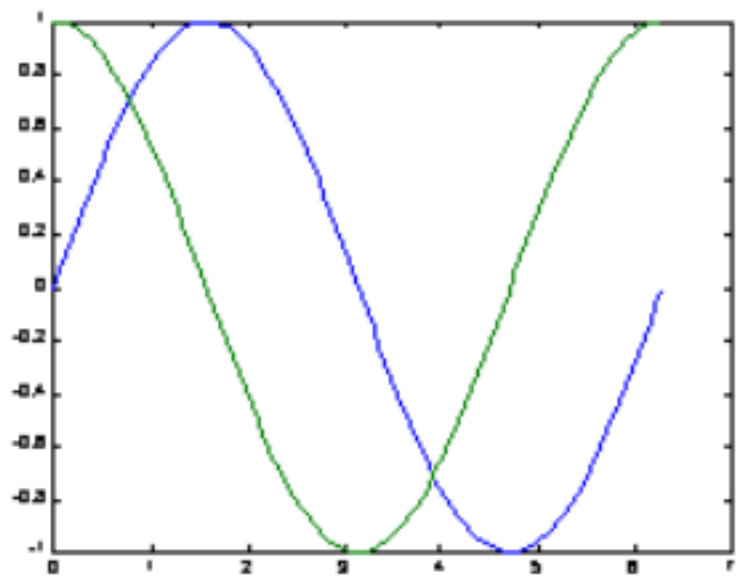
loglog: x　轴和 y 轴均为对数刻度（ Logarithmic scale ）

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

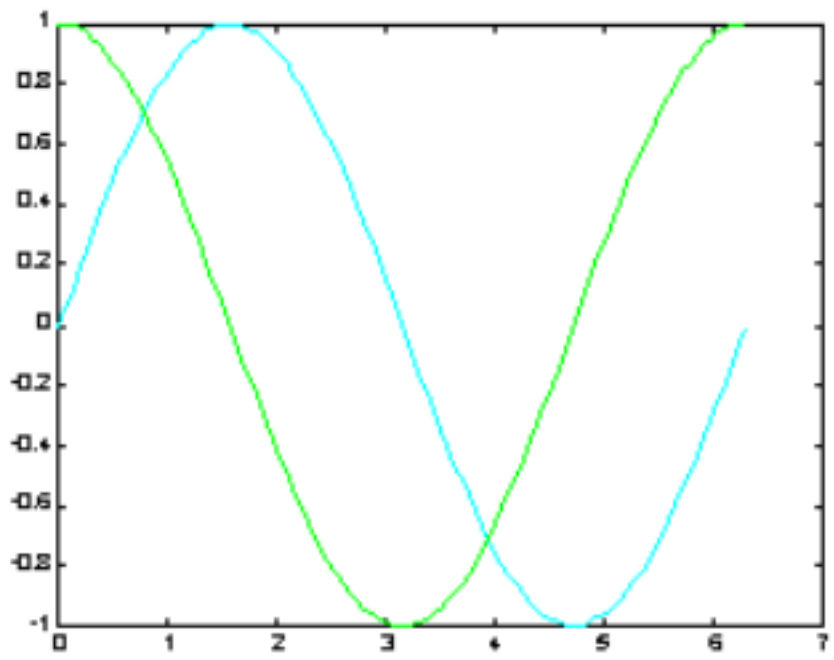
若要画出多条曲线，只需将座标对依次放入　plot　函数即可：

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：

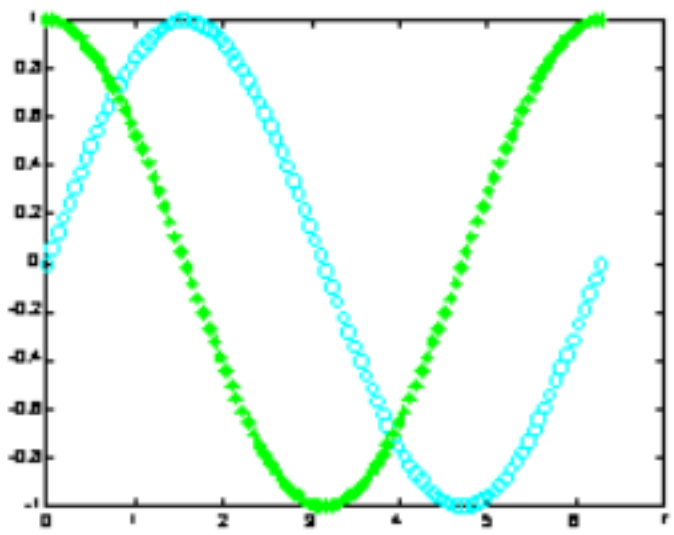
plot(x, sin(x), 'c', x, cos(x), 'g');



<![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：

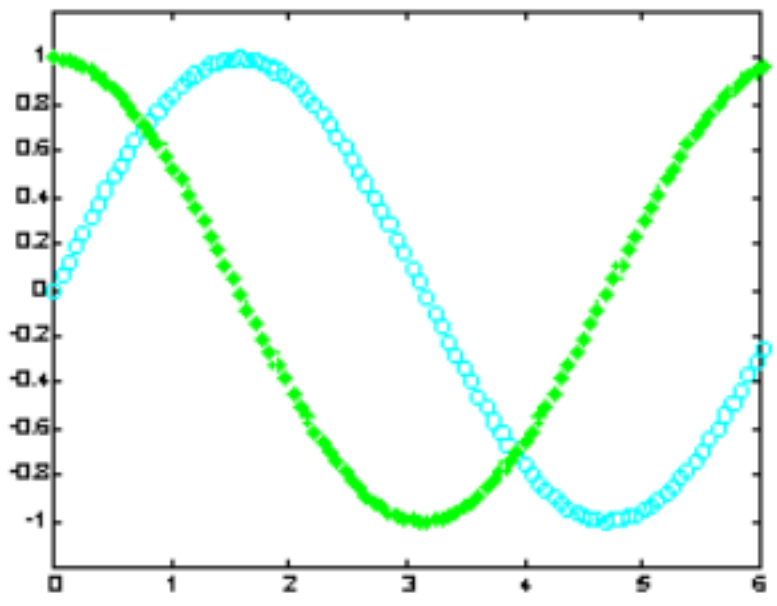
plot(x, sin(x), 'co', x, cos(x), 'g*');



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

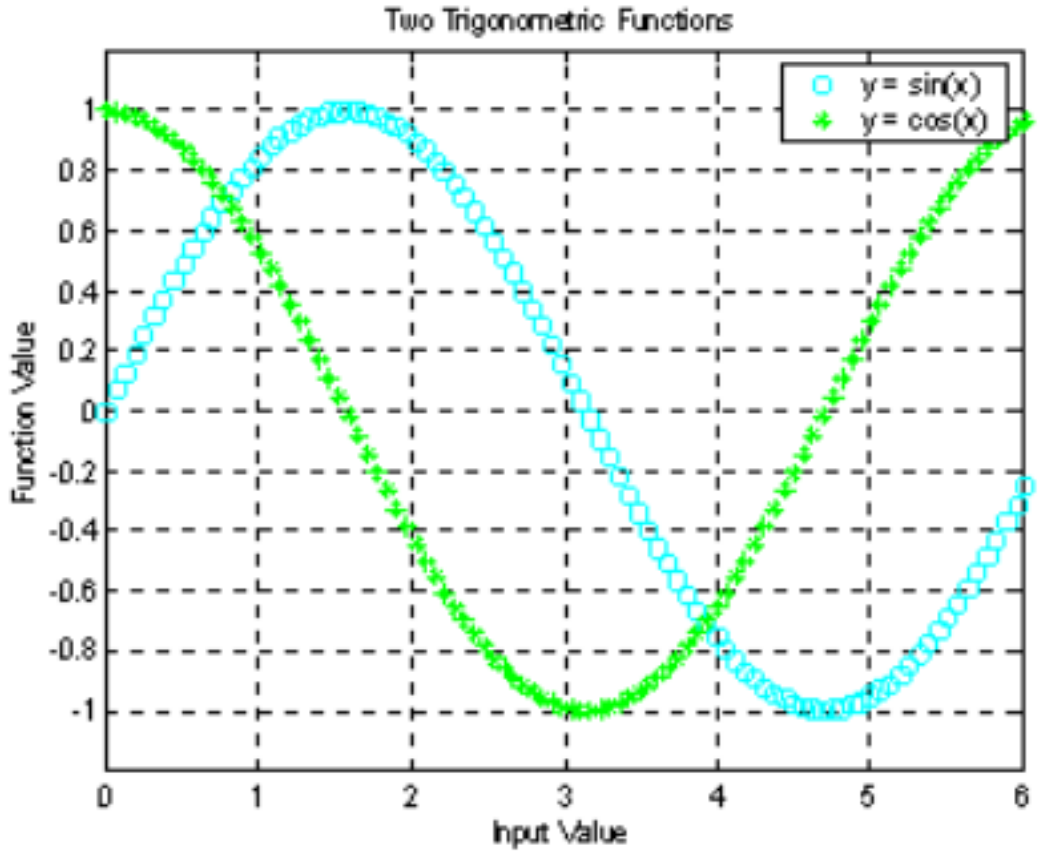
图形完成後，我们可用 axis([xmin,xmax,ymin,ymax]) 函数来调整图轴的范围：

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



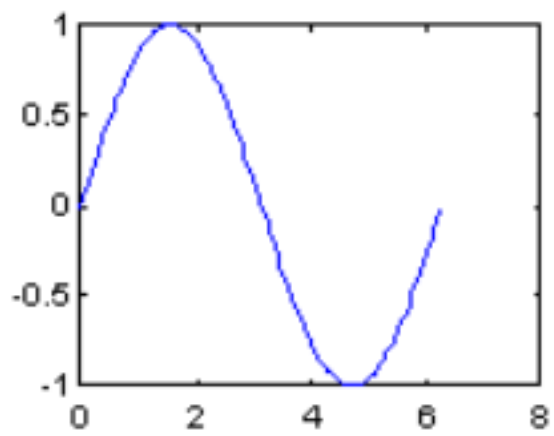
此外， MATLAB也可对图形加上各种注解与处理：

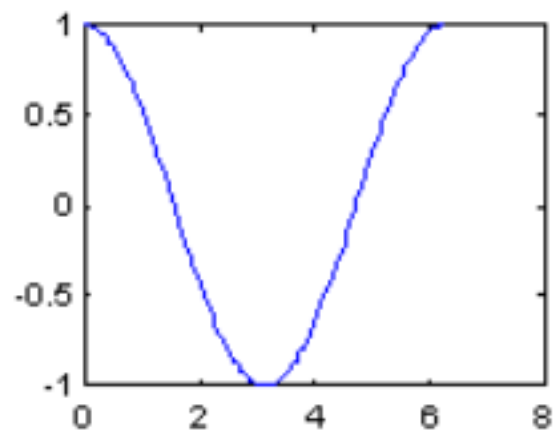
```
xlabel('Input Value'); % x          轴注解

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
```

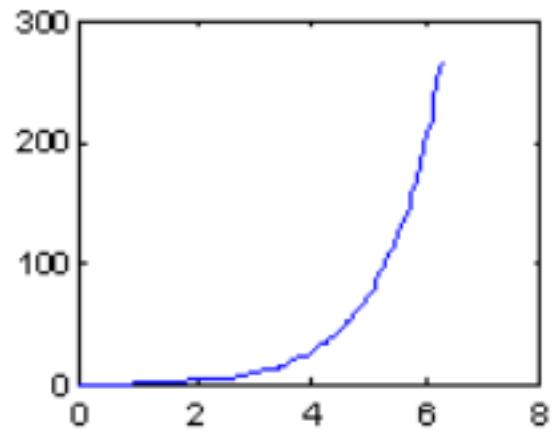


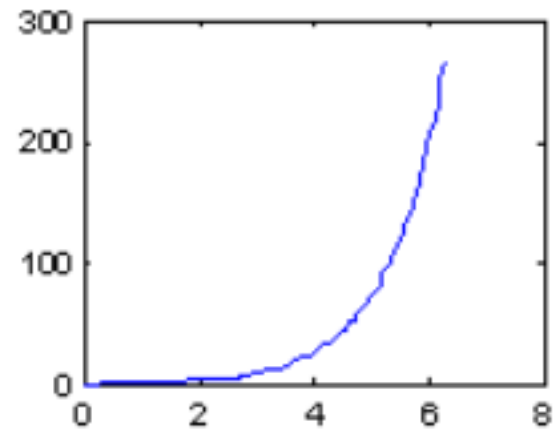
我们可用 subplot 来同时画出数个小图形於同一个视窗之中：

```
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

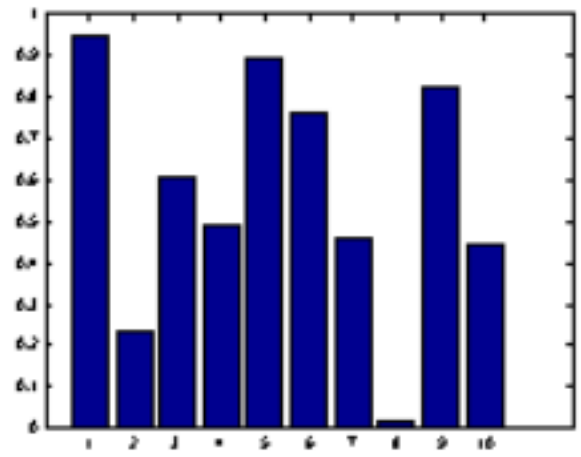
compass 罗盘图

quiver　向量场图

以下我们针对每个函数举例。

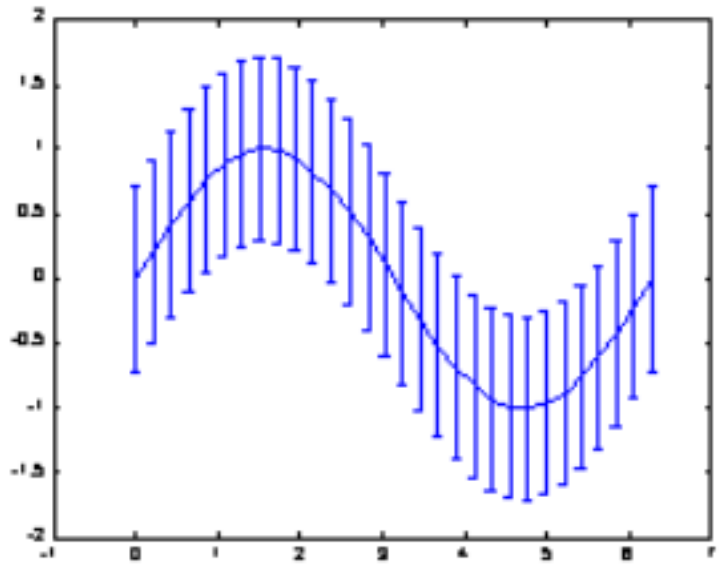
当资料点数量不多时，长条图是很适合的表示方式：

```
close all; %　　关闭所有的图形视窗

x=1:10;

y=rand(size(x));

bar(x,y);
```



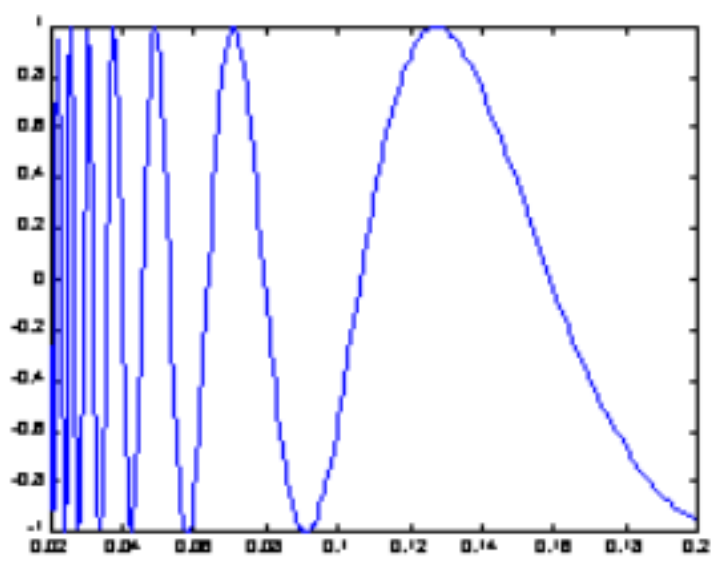
如果已知资料的误差量，就可用　errorbar　来表示。下例以单位标准差来做资的误差量：

```
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



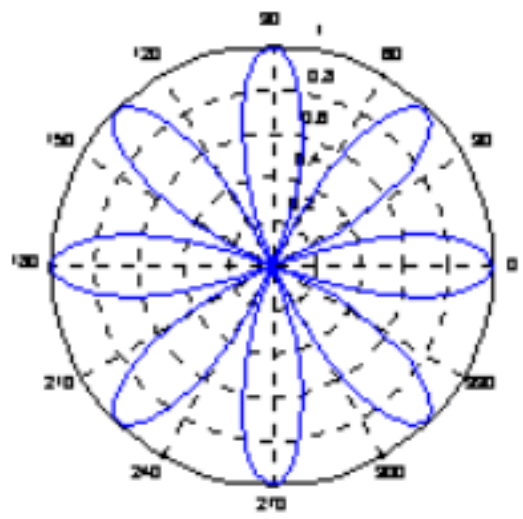
<![endif]>

若要产生极座标图形，可用　polar ：

theta=linspace(0, 2*pi);

r=cos(4*theta);

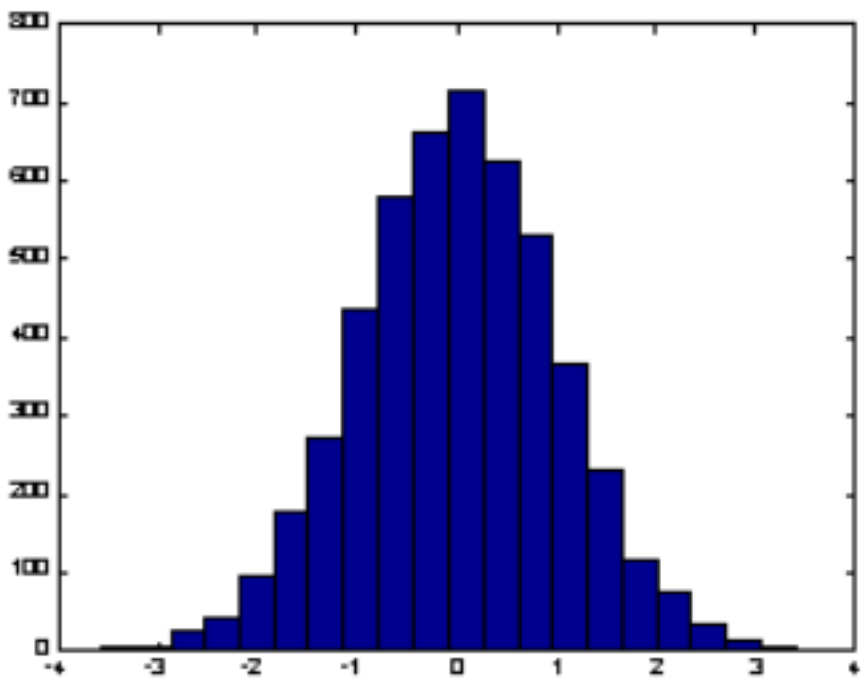
polar(theta, r);



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 ：

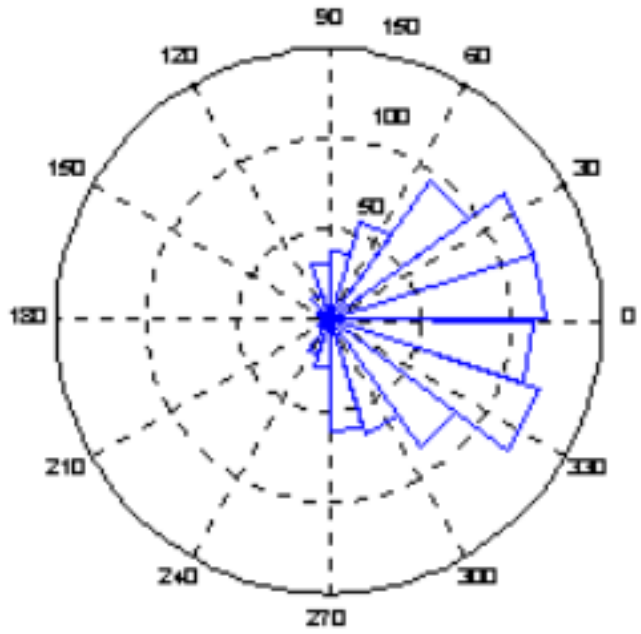
x=randn(5000, 1); % 产生 5000 个 m=0, s=1 的高斯乱数

hist(x,20); % 20 代表长条的个数



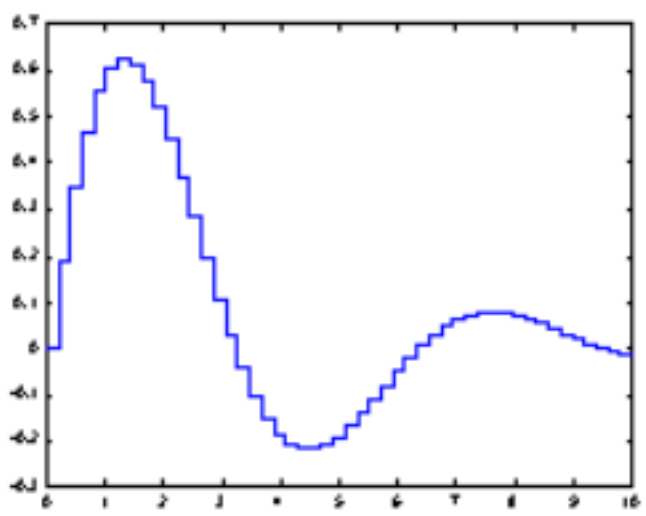
rose 和 hist 很接近，只不过是将资料大小视为角度，资料个数视为距离，并用极座标绘制表示：

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

rose(x);
```



stairs 可画出阶梯图：

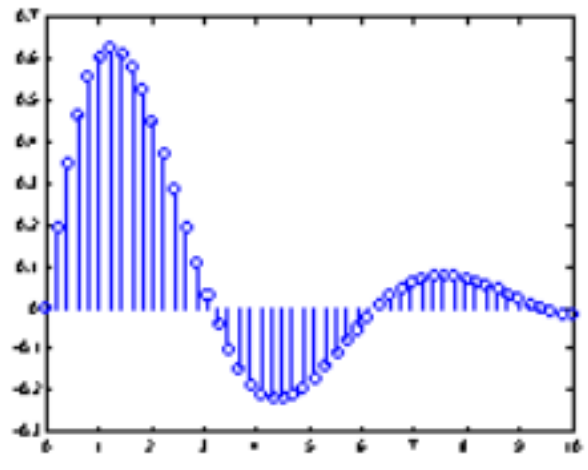
```
x=linspace(0,10,50);

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

stairs(x,y);
```



<![endif]>

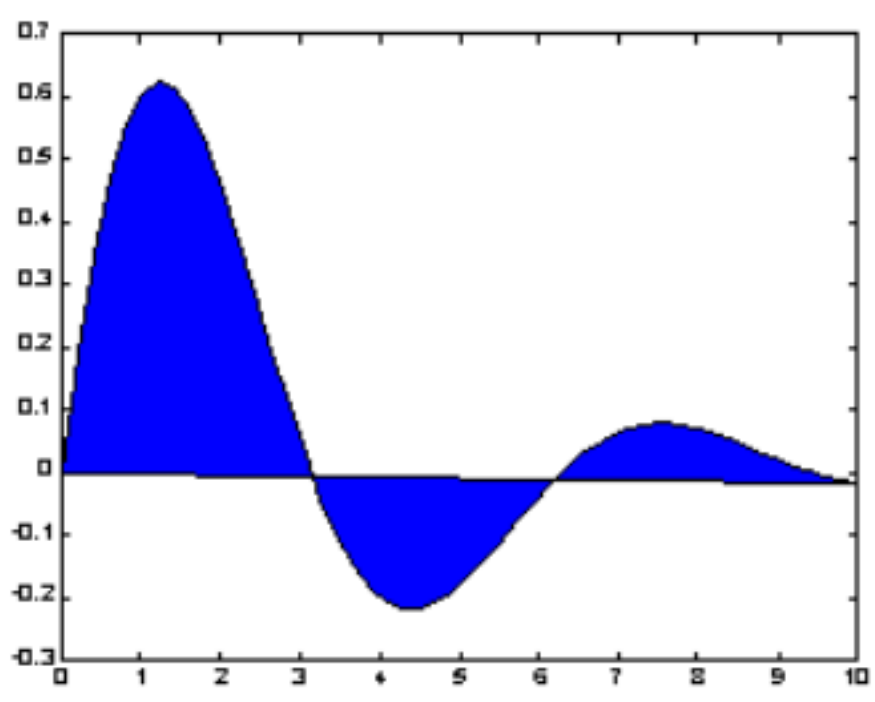
stems 可产生针状图，常被用来绘制数位讯号：

```
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
```

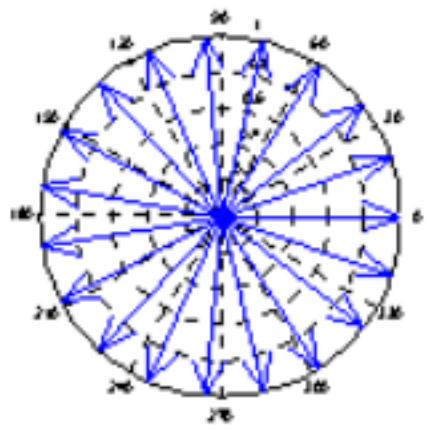


feather　将每一个资料点视复数，并以箭号画出：

```
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);
```



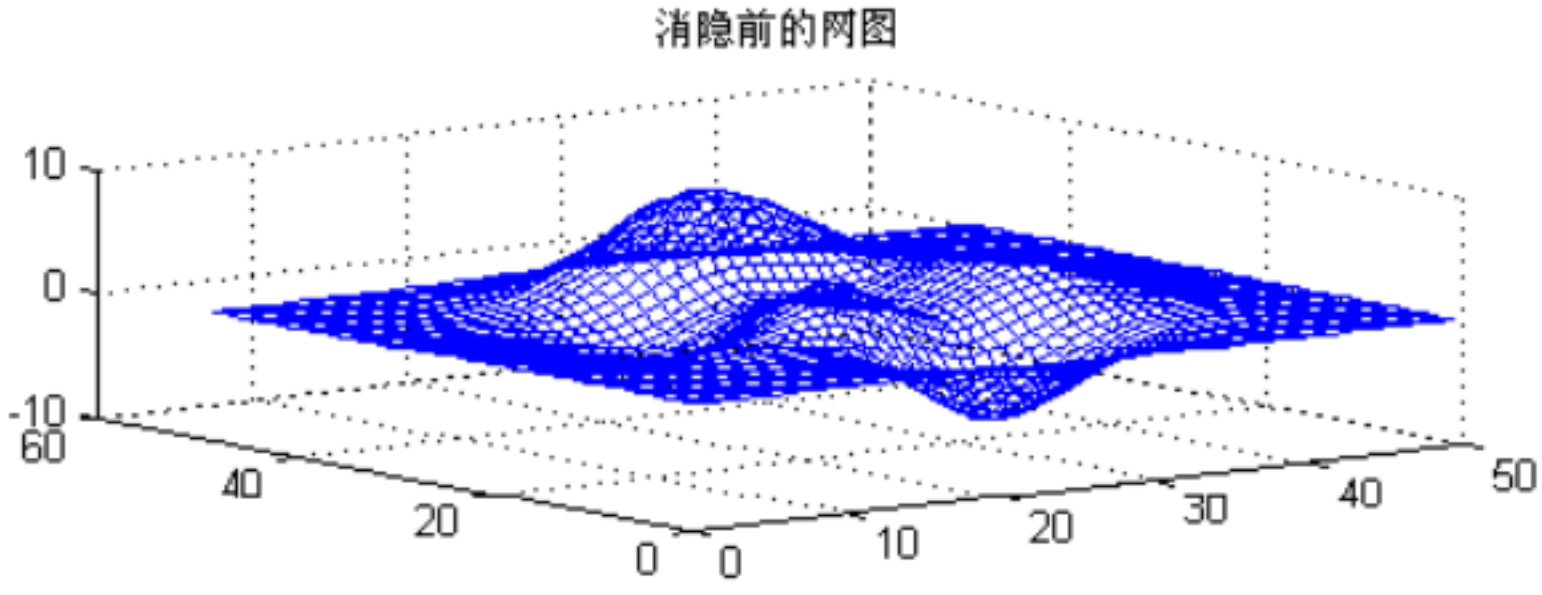
<![endif]>

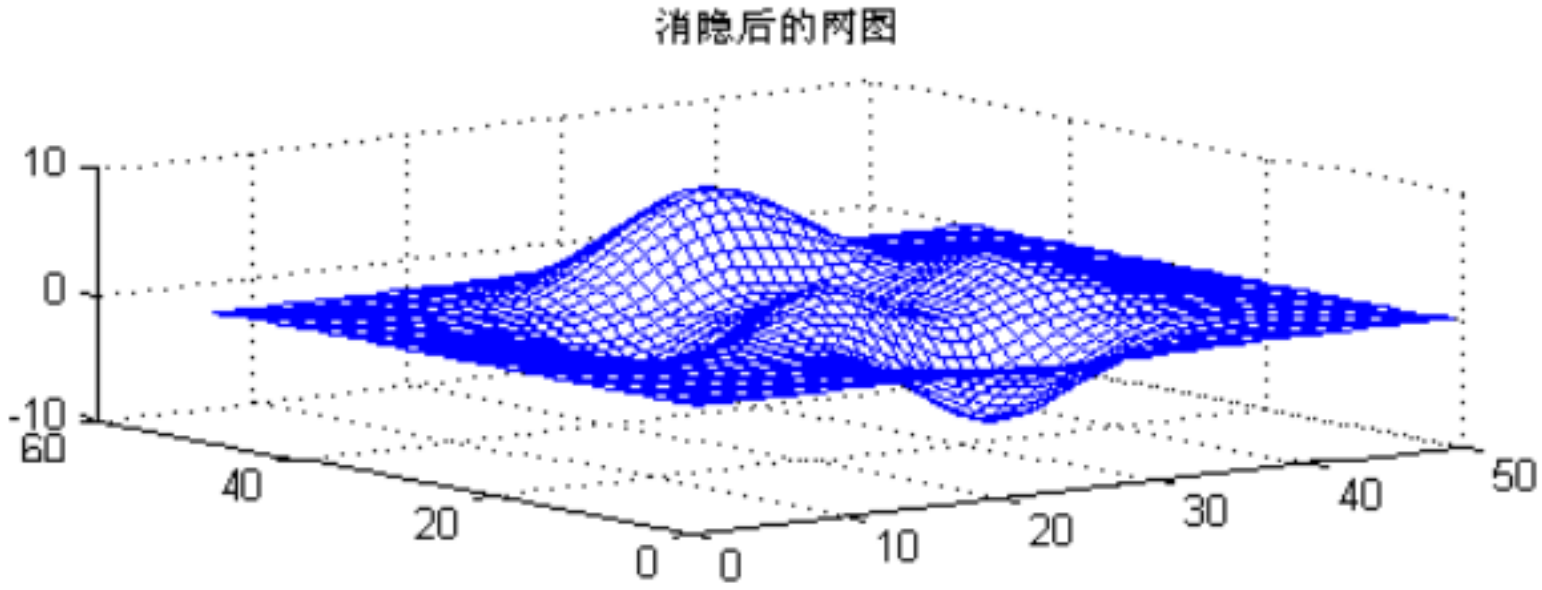
4. 三维网图的高级处理

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])
```



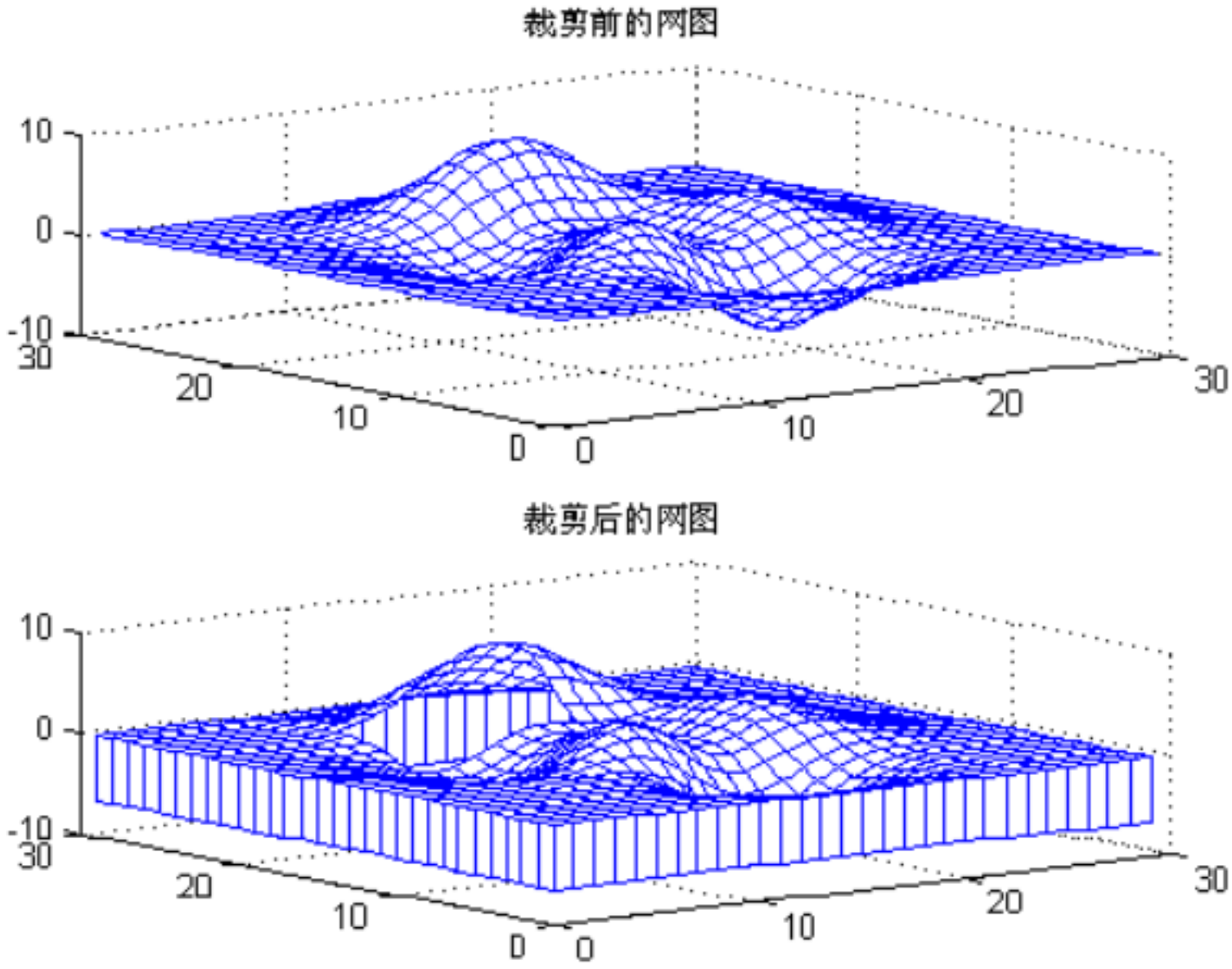


2. 裁剪处理

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

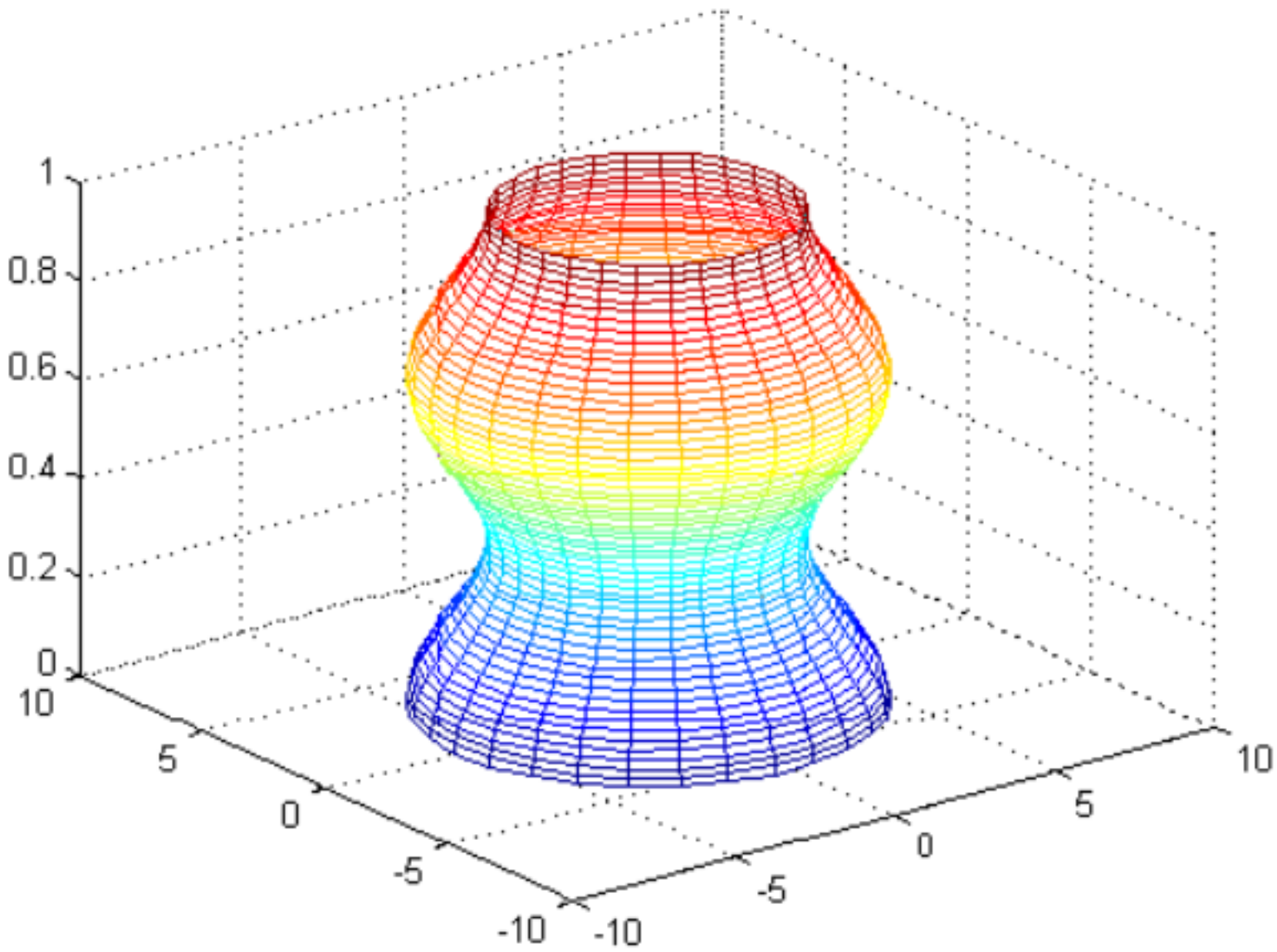
柱面图绘制由函数 cylinder 实现 .

[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]

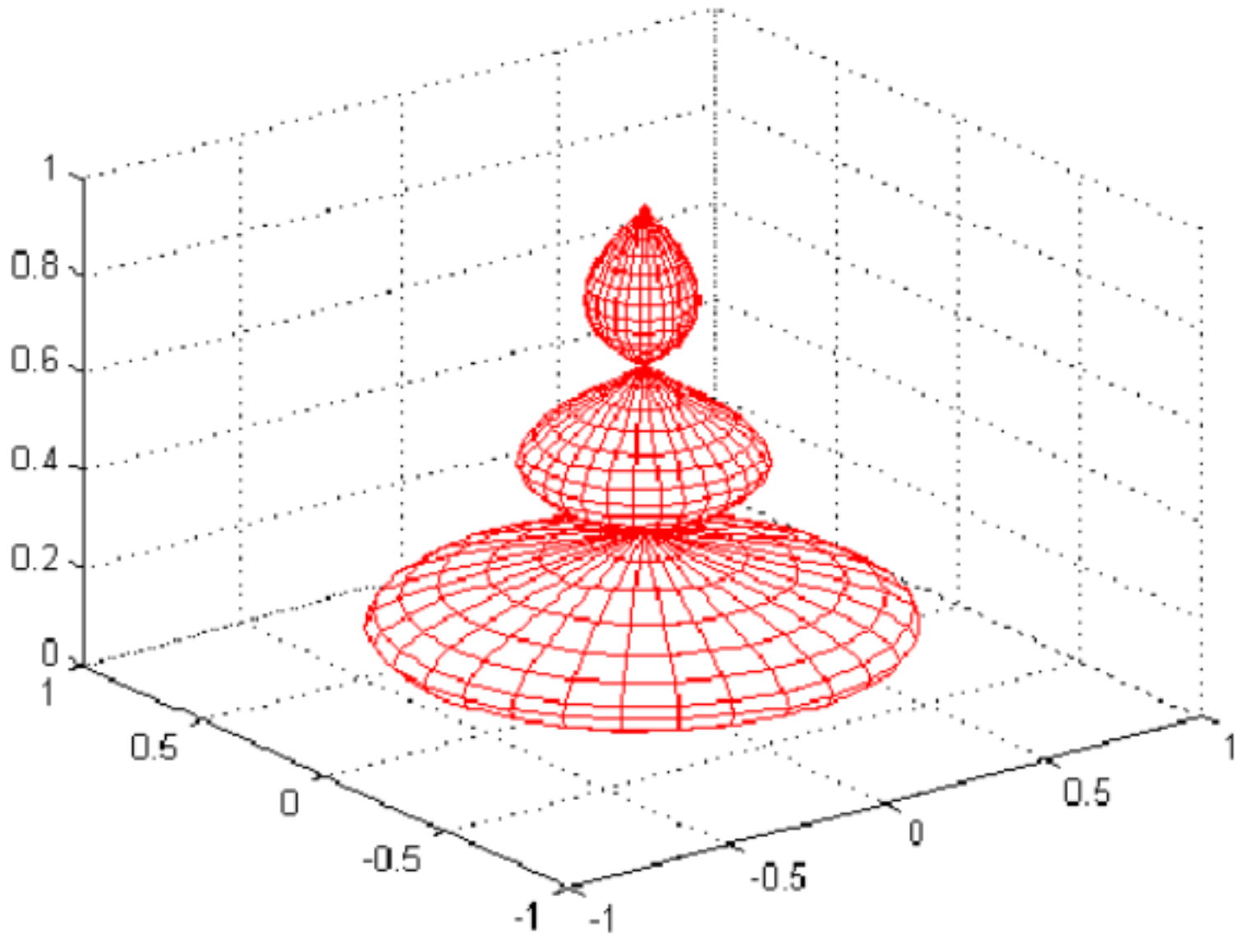
例. 柱面函数演示举例

```
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')
```

一 数 值 函 数

N[expr] 表达式的机器精度近似值

N[expr, n]　　表达式的 n 位近似值, n 为任意正整数

NSolve[lhs==rhs, var]　　求方程数值解

NSolve[eqn, var, n]　　求方程数值解，结果精度到　n 位

NDSolve[eqns, y, {x, xmin, xmax}]　　微分方程数值解

NDSolve[eqns, {y1,y2,...}, {x, xmin, xmax}]

微分方程组数值解

FindRoot[lhs==rhs, {x,x0}]　　以 x0 为初值，寻找方程数值解

FindRoot[lhs==rhs, {x, xstart, xmin, xmax}]

NSum[f, {i,imin,imax,di}]　　数值求和，　di 为步长

NSum[f, {i,imin,imax,di}, {j,..},..]　　多维函数求和

NProduct[f, {i, imin, imax, di}]　　函数求积

NIntegrate[f, {x, xmin, xmax}]　　函数数值积分

优化函数：

FindMinimum[f, {x,x0}]　　以 x0 为初值，寻找函数最小值

FindMinimum[f, {x, xstart, xmin, xmax}]

ConstrainedMin[f,{inequ},{x,y,..}]

inequ　　为线性不等式组，　f 为 x,y..　之线性函数，得到最小值及此时的　x,y..　取值

ConstrainedMax[f, {inequ}, {x, y,..}]　　同上

LinearProgramming[c,m,b]　　解线性组合 c.x 在 m.x>=b&&x>=0约束下的

最小值，　x,b,c　为向量 ,m 为矩阵

LatticeReduce[{v1,v2...}]　　向量组 vi 的极小无关组

数据处理：

Fit[data,funs,vars]　　用指定函数组对数据进行最小二乘拟和

data　　可以为 {{x1,y1,..f1},{x2,y2,..f2}..}　　多维的情况

emp: Fit[{10.22,12,3.2,9.9}, {1, x, x^2,Sin[x]}, x]

Interpolation[data] 对数据进行差值 ，

data 同上，另外还可以为 {{x1,{f1,df11,df12}},{x2,{f2,.}..} 指定各阶导数

InterpolationOrder 默认为 3 次，可修改

ListInterpolation[array] 对离散数据插值， array 可为 n 维

ListInterpolation[array,{{xmin,xmax},{ymin,ymax},..}]

FunctionInterpolation[expr,{x,xmin,xmax}, {y,ymin,ymax},..]

以对应 expr[xi,yi] 的为数据进行插值

Fourier[list] 对复数数据进行付氏变换

InverseFourier[list] 对复数数据进行付氏逆变换

Min[{x1,x2...},{y1,y2,...}] 得到每个表中的最小值

Max[{x1,x2...},{y1,y2,...}] 得到每个表中的最大值

Select[list, crit] 将表中使得 crit 为 True 的元素选择出来

Count[list, pattern] 将表中匹配模式 pattern 的元素的个数

Sort[list] 将表中元素按升序排列

Sort[list,p] 将表中元素按 p[e1,e2] 为 True 的顺序比较 list

的任两个元素 e1,e2, 实际上 Sort[list] 中默认 p=Greater

集合论：

Union[list1,list2..] 表 listi 的并集并排序

Intersection[list1,list2..] 表 listi 的交集并排序

Complement[listall,list1,list2...] 从全集 listall 中对 listi 的差集

九、虚数函数

Re[expr] 复数表达式的实部

Im[expr]	复数表达式的虚部
Abs[expr]	复数表达式的模
Arg[expr]	复数表达式的辐角
Conjugate[expr]	复数表达式的共轭

十、数的头及模式及其他操作

Integer _Integer	整数
Real _Real	实数
Complex _Complex	复数
Rational_Rational	有理数

(* 注：模式用在函数参数传递中，如 MyFun[Para1_Integer,Para2_Real]
规定传入参数的类型，另外也可用来判断 If[Head[a]==Real,...]*)

IntegerDigits[n,b,len]		数字 n 以 b 近制的前 len 个码元
RealDigits[x,b,len]		类上
FromDigits[list]	IntegerDigits	的反函数
Rationalize[x,dx]		把实数 x 有理化成有理数，误差小于 dx
Chop[expr, delta]		将 expr 中小于 delta 的部分去掉 ,dx 默认为 10^-10
Accuracy[x]		给出 x 小数部分位数 , 对于 Pi,E 等为无限大
Precision[x]		给出 x 有效数字位数 , 对于 Pi,E 等为无限大
SetAccuracy[expr, n]		设置 expr 显示时的小数部分位数
SetPrecision[expr, n]		设置 expr 显示时的有效数字位数

十一、区间函数

Interval[{min, max}] 区间 [min, max](* Solve[3 x+2==Interval[{-2,5}],x]*)

IntervalMemberQ[interval, x]　　x　　在区间内吗？

IntervalMemberQ[interval1,interval2]　　区间 2 在区间 1 内吗？

IntervalUnion[intv1,intv2...]　　区间的并

IntervalIntersection[intv1,intv2...]　　区间的交

十二、矩阵操作

a.b.c　　或 Dot[a, b, c]　　矩阵、向量、张量的点积

Inverse[m]　　矩阵的逆

Transpose[list]　　矩阵的转置

Transpose[list,{n1,n2..}]　　将矩阵 list 第 k 行与第 nk 列交换

Det[m]　　矩阵的行列式

Eigenvalues[m]　　特征值

Eigenvectors[m]　　特征向量

Eigensystem[m]　　特征系统，返回 {eigvalues,eigvectors}

LinearSolve[m, b]　　解线性方程组 m.x==b

NullSpace[m]　　矩阵 m的零空间，即 m.NullSpace[m]== 零向量

RowReduce[m]　　m　　化简为阶梯矩阵

Minors[m, k]　　m　　的所有 k*k 阶子矩阵的行列式的值 (伴随阵，好像是)

MatrixPower[mat, n]　　阵 mat 自乘 n 次

Outer[f,list1,list2..]　listi　　中各个元之间相互组合，并作为 f 的参数的到的矩阵

Outer[Times,list1,list2]　　给出矩阵的外积

SingularValues[m]　　m　　的奇异值，结果为 {u,w,v},

m=Conjugate[Transpose[u]].DiagonalMatrix[w].v

PseudoInverse[m]　　m　　的广义逆

QRDecomposition[m]	QR	分解
SchurDecomposition[m]	Schur	分解
LUDecomposition[m]	LU	分解

. 计算机代数系统

Mathematica 函数大全 -- 运算符及特殊符号

一、运算符及特殊符号

Line1;	执行 Line ， 不显示结果
Line1,line2	顺次执行 Line1 ， 2, 并显示结果
?name	关于系统变量 name的信息
??name	关于系统变量 name的全部信息
!command	执行 Dos 命令
n! N	的阶乘
!!filename	显示文件内容
<<filename	读入文件并执行
Expr>> filename	打开文件写
Expr>>>filename	打开文件从文件末写
()	结合率
[]	函数
{}	一个表
<*Math Fun*>	在 c 语言中使用 math 的函数

(*Note*)	程序的注释
#n	第 n 个参数
##	所有参数
rule&	把 rule 作用于后面的式子
%	前一次的输出
%%	倒数第二次的输出
%n	第 n 个输出
var::note	变量 var 的注释
"Astring "	字符串
Context `	上下文
a+b	加
a-b	减
a*b 或 a b	乘
a/b	除
a^b	乘方
base^^num	以 base 为进位的数
lhs&&rhs	且
lhs||rhs	或
!lha	非
++,--	自加 1，自减 1
+=,-=,*=,/=	同 C语言
>,<,>=,<=,==,!=	逻辑判断（同 c）
lhs=rhs	立即赋值

lhs:=rhs	建立动态赋值
lhs:>rhs	建立替换规则
lhs->rhs	建立替换规则
expr//funname	相当于 filename[expr]
expr/.rule	将规则 rule 应用于 expr
expr//.rule	将规则 rule 不断应用于 expr 知道不变为止
param_	名为 param的一个任意表达式（形式变量）
param__	名为 param的任意多个任意表达式（形式变量）

二、系统常数

Pi	3.1415....	的无限精度数值
E	2.17828...	的无限精度数值
Catalan	0.915966..	卡塔兰常数
EulerGamma	0.5772....	高斯常数
GoldenRatio	1.61803...	黄金分割数
Degree	Pi/180	角度弧度换算

I	复数单位
Infinity	无穷大
-Infinity	负无穷大
ComplexInfinity	复无穷大
Indeterminate	不定式

三、代数计算

Expand[expr]	展开表达式
Factor[expr]	展开表达式

Simplify[expr]	化简表达式
FullSimplify[expr]	将特殊函数等也进行化简
PowerExpand[expr]	展开所有的幂次形式
ComplexExpand[expr,{x1,x2...}]	按复数实部虚部展开
FunctionExpand[expr]	化简 expr 中的特殊函数
Collect[expr, x]	合并同次项
Collect[expr, {x1,x2,...}]	合并 x1,x2,... 的同次项
Together[expr]	通分
Apart[expr]	部分分式展开
Apart[expr, var]	对 var 的部分分式展开
Cancel[expr]	约分
ExpandAll[expr]	展开表达式
ExpandAll[expr, patt]	展开表达式
FactorTerms[poly]	提出共有的数字因子
FactorTerms[poly, x]	提出与 x 无关的数字因子
FactorTerms[poly, {x1,x2...}]	提出与 xi 无关的数字因子
Coefficient[expr, form]	多项式 expr 中 form 的系数
Coefficient[expr, form, n]	多项式 expr 中 form^n 的系数
Exponent[expr, form]	表达式 expr 中 form 的最高指数
Numerator[expr]	表达式 expr 的分子
Denominator[expr]	表达式 expr 的分母
ExpandNumerator[expr]	展开 expr 的分子部分
ExpandDenominator[expr]	展开 expr 的分母部分
TrigExpand[expr]	展开表达式中的三角函数

TrigFactor[expr]	给出表达式中的三角函数因子
TrigFactorList[expr]	给出表达式中的三角函数因子的表
TrigReduce[expr]	对表达式中的三角函数化简
TrigToExp[expr]	三角到指数的转化
ExpToTrig[expr]	指数到三角的转化

RootReduce[expr]

ToRadicals[expr]

Mathematica 入门教程

Mathematica 的基本语法特征

如果你是第一次使用 Mathematica，那么以下几点请你一定牢牢记住：

Mathematica 中大写小写是有区别的，如 Name，name，NAME等是不同的变量名或函数名。

系统所提供的功能大部分以系统函数的形式给出，内部函数一般写全称，而且一定是以大写英文字母开头，如 Sin[x],Conjugate[z] 等。

乘法即可以用 *，又可以用空格表示，如 2 3 ＝2*3＝6 ,x y,2 Sin[x] 等；乘幂可以用“ ^ ”表示，如 x^0.5,Tan[x]^y 。

自定义的变量可以取几乎任意的名称，长度不限，但不可以数字开头。

当你赋予变量任何一个值，除非你明显地改变该值或使用 Clear[变量名] 或“ 变量名 =. ”取消该值为止，它将始终保持原值不变。

一定要注意四种括号的用法： () 圆括号表示项的结合顺序， 如(x+(y^x+1/(2x)));[] 方括号表示函数，如 Log[x],BesselJ[x,1] ；{} 大括号表示一个“ 表 ” (一组数字、任意表达式、函数等的集合)，如{2x,Sin[12 Pi],{1+A,y*x}} ；[[]] 双方括号表示“ 表 ”或“ 表达式 ”的下标，如 a[[2,3]] 、{1,2,3}[[1]]=1 。

Mathematica 的语句书写十分方便，一个语句可以分为多行写，同一行可以写多个语句（但要以分号间隔）。当语句以分号结束时，语句计算后不做输出（输出语句除外），否则将输出计算的结果。

一. 数的表示及计算

1. 在 Mathematica 中你不必考虑数的精确度，因为除非你指定输出精度，Mathematica 总会以绝对精确的形式输出结果。例如：你输入

In[1]:=378/123 ，系统会输出 Out[1]:=126/41 ，如果想得到近似解，则应输入

In[2]:=N[378/123,5], 即求其 5 位有效数字的数值解，系统会输出 Out[2]:=3.073

2，另外 Mathematica 还可以根据你前面使用的数字的精度自动地设定精度。

Mathematica 与众不同之处还在于它可以处理任意大、任意小及任意位精度的数值，如 100^7000,2^(-2000) 等数值可以很快地求出，但在其他语言或系统中这是不可想象的，你不妨试一试 N[Pi,1000] 。

Mathematica 还定义了一些系统常数，如上面提到的 Pi(圆周率的精确值)，还有 E(自然对数的底数)、I(复数单位)，Degree(角度一度， Pi/180) ，Infinity(无穷大) 等，不要小看这些简单的符号， 它们包含的信息远远大于我们所熟知的它们的近似值， 它们的精度也是无限的。

二.“ 表 ”及其用法

“ 表 ”是 Mathematica 中一个相当有用的数据类型，它即可以作为数组，又可以作为矩阵；除此以外，你可以把任意一组表达式用一个或一组 {} 括起来，进行运算、存储。可以说表是任意对象的一个集合。它可以动态地分配内存，可以方便地进行插入、删除、排序、翻转等等几乎所有可以想象到的操作。

如果你建立了一个表，你可以通过下表操作符 [[]](双方括号) 来访问它的每一个元素，如我们定义 table={2,Pi,Sin[x],{aaa,A*I}} 为一个表，那么 table[[1]] 就为 2，table[[2]] 就是 Pi，而 table[[3,1]] 表示嵌套在 table 中的子表 {aaa,A*I} 的第一个元素即 aaa，table[[3,2]] 表示{aaa,A*I} 第二个元素即 A*I 。总之，表每一层次上并列的部分用逗号分割，表可以无穷嵌套。

你可以通过 Append[表, 表达式] 或 Prepend[表, 表达式] 把表达式添加到表的最前面或最后面，如 Append[{1,2,3},a] 表示{1,2,3,a} 。你还可以通过 Union[表 1, 表 2, ...],Jion[表 1, 表 2,.....] 来把几个表合并为一个表，二者不同在于 Union 在合并时删除了各表中重复的元素，而后者仅是简单的合并；你还可以使用 Flatten[表] 把表中所有子表 "抹平" 合并成一个表，而 Patition[表，整数 n] 把表按每 n 个元素分段作为子表，集合成的表。如 Flatten[{1,2,{Sin[x],dog},{{y}}}] 表示 {1,2,Sin[x],y}, 而 Partition[{1,2,Sin[x],y},2] 把表每两个分段，结果为 {{1,2},{Sin[x],y}} ；还可以通过 Delete[表，位置] 、Insert[表，位置] 来向表中按位置插入或删除元素，如要删除上面提到的 table 中的 aaa, 你可以用 Delete[table,{3,1}] 来实现； Sort[表] 给出了表中各元素的大小顺序， Reverse[表] 、RotateLeft[表，整数 n] 、RotateRight[表，整数 n] 可以分别将一个表进行翻转、左转 n 个元素、右转 n 个元素等操作， Length[表] 给出了表第一个层次上的元素个数， Position[表，表达式] 给出了表中出现该表达式的位置， Count[表，表达式] 则给出表达式出现的次数。各种表的操作函数还有很多，这里就不再一一介绍了。

三. 图形函数

Mathematica 的图形函数十分丰富，用寥寥几句就可以画出复杂的图形，而且可以通过变量和文件存储和显示图形，具有极大的灵活性。

图形函数中最有代表性的函数为 Plot[表达式，{变量，下限，上限 }，可选项]，(其中表达式还可以是一个 " 表达式表 "，这样可以在一个图里画多个函数)；变量为自变量； 上限和下限确定了作图的范围；可选项可要可不要，不写系统会按默认值作图，它表示对作图的具体要求。例如 Plot[Sin[x],{x,0,2*Pi},AspectRatio-1] 表示在 0<x<2Pi 的范围内作函数 Sin[x] 的图象， AspectRatio 为可选项，表示图的 x 向 y 向比例， AspectRatio-1 表示纵横比例为 1:1，如果不写这一项，系统默认比例为 1:GodenRatio，即黄金分割的比例 (注意，可选项的写法为可选项名 - 可选项值), Plot 还有很多可选项，如 PlotRange 表示作图的值域， PlotPoint 表画图中取样点的个数，越大则图越精细， PlotStyle 来确定所画图形的线宽、线型、颜色等特性， AxesLabel 表式在坐标轴上作标记等等。

- 二维函数作图

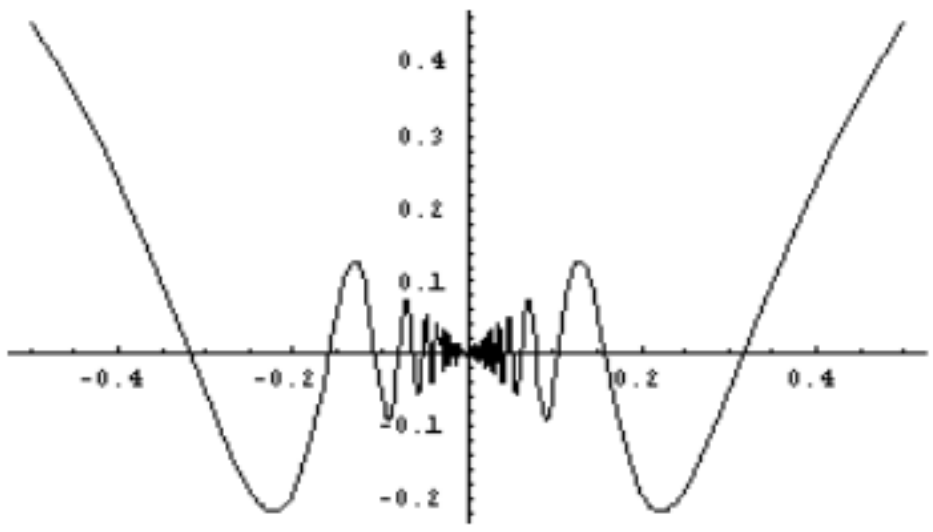
Plot[函数 f，{x， xmin，xmax}，选项]

在区间 {x， xmin，xmax}上，按选项的要求画出函数 f 的图形

Plot[{ 函数 1，函数 2}，{x， xmin，xmax}，选项]

在区间 {x， xmin，xmax}上，按选项的要求画出几个函数的图形

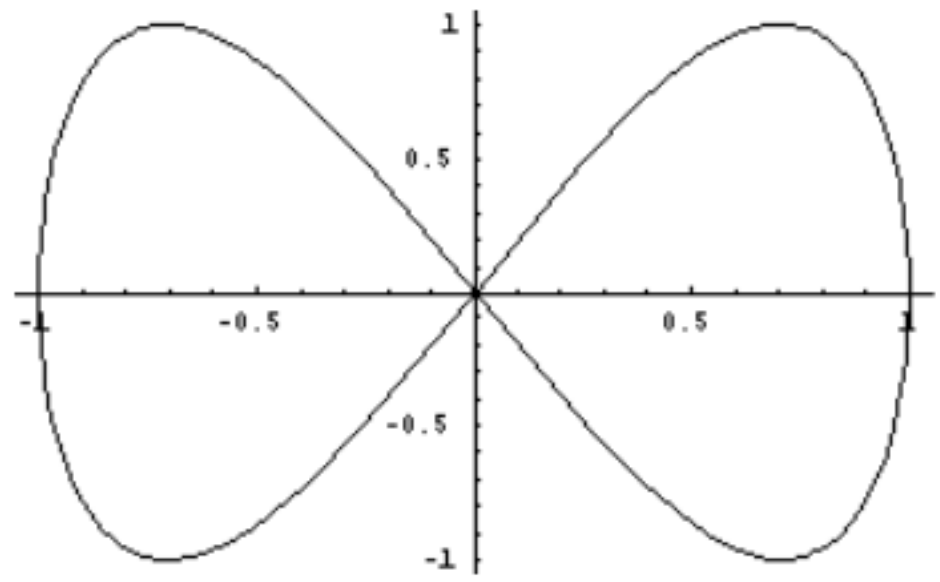
图一. 用 Plot 生成 x*Sin[1/x] 的图形



- 二维参数画图函数

ParametricPlot[{x[t],y[t]},{t,t0,t1}, 选项] 画一个 X轴,Y 轴坐标为 {x[t],y[t]}, 参变量 t 在[t0,t1] 中的参数曲线

图二. 用 ParametricPlot 生成 $\begin{cases} x = \sin[t] \\ y = \sin[2t] \end{cases}$ 的图形

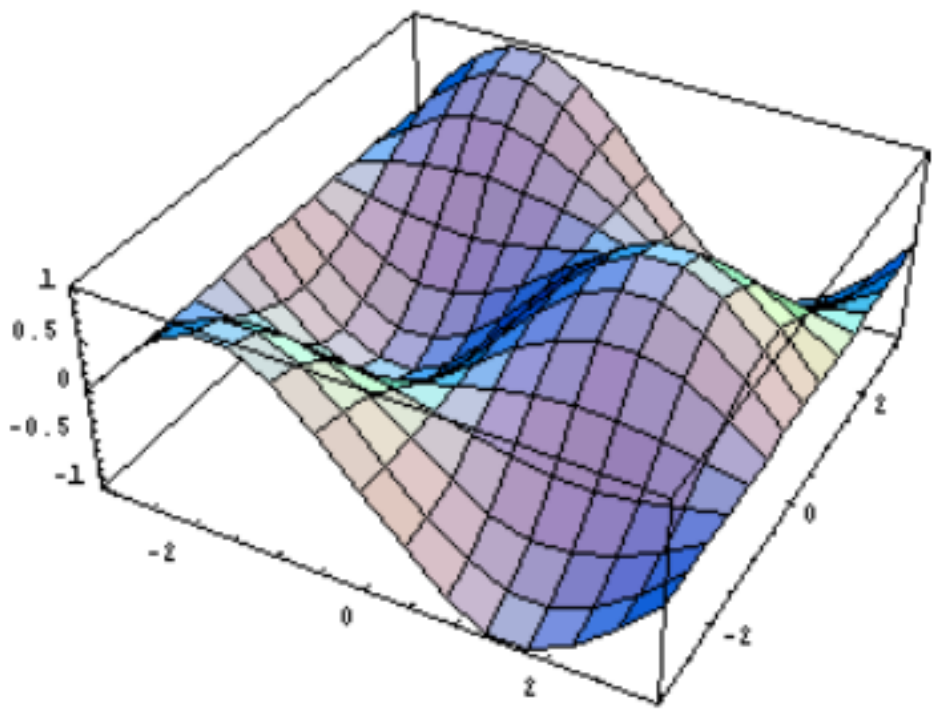


- 三维函数作图

Plot3D[f[x,y],{x,x0,x1},{y,y0,y1}, 选项]

在区域 $x \in [x0, x1]$和$y \in [y0, y1]$ 上, 画出空间曲面 f[x,y].

图 3. 用 Plot3D 生成的 Sin[x]*Cos[y] 的三维图形



除 Plot, 二维参数方程作图的 ParametricPlot[{x(t),y(t)},{t, 下限, 上限}, 可选项]、三维作图的 Plot3D[二维函数表达式, {变量 1, 下限, 上限}, {变量 2, 下限, 上限}, 可选项}] 、三维参数方程作图的 ParametricPlot3D[{x(u,v),y(u,v),z(u,v)},{u, 下限, 上限},{v, 下限, 上限}, 可选项] 外, 还有画二维等高线图 ContourPlot[二元表达式, {变量 1, 下限, 上限}, {变

量 2，下限，上限 }，可选项 }] 、画二维密度图的 DensityPlot[二元表达式，{变量 1，下限，上限},{ 变量 2，下限，上限 }，可选项 }] 等等不一而足。

除使用上述函数作图以外， Mathematica 还可以象其他语言一样使用图形元语言作图，如画点函数 Point[x,y], 画线函数 Line[x1,y1,x2,y2], 画圆的 Circle[x,y,r] ，画矩形和多边形的 Rectangle 和 Polygon, 字符输出的 Text[字符串 , 输出坐标]，还有颜色函数 RGBColor[red,green,blue] 、Hue[],GrayLevel[gray] 来描述颜色的亮度、灰度、饱和度，用 PointSize[相对尺度] 、Thickness[相对尺度] 来表示点和线的宽度。总之 Mathematica 可以精确地调节图形的每一个特征。

四. 数学函数的用法

Mathematica 系统内核提供了丰富的数学计算的函数，包括极限、积分、微分、最值、极值、统计、规划等数学的各个领域，复杂的数学问题简化为对函数的调用，极大地提高了解决问题的效率。

Mathematica 提供了所有的三角、反三角、双曲、反双曲、各种特殊函数 (如贝塞尔函数系、椭圆函数等)，各种复数函数 (如 Im[z],Re[z],Conjugate[z], Abs[z],Arg[z]) ，各种随机函数 (如 Random[n]可以通过不同的参数产生任意范围内整型、实型任意分布的随机数)，矩阵运算函数 (如求特征值特征向量的 EigenVector[],EigenValue[], 求逆的 Inverse[] 等) 。

Mathematica 还提供了大量数学操作的函数，如取极限的 Limit[f[x],{x,a}], 求微分的 D[f[x],x], 全微分的 Dt[f[x],x], 不定积分的 Integrate[f[x],x] 和定积分的 Integrate[f[x],{x,a,b}], 解任意方程的 Solve[lhs=rhs,x] 及微分方程的 DSolve[lhs=rhs,x], 解幂级数和付立叶展开的 Series[f[x]] , Fourier[f[x]] 及其逆变化 InverseSeries,InverseFourier, 求和函数 Sum[], 求积函数 Product[] ，以上函数均可以适用于多维函数或多维方程。

Mathematica 中还有相当数量的数值计算函数，最常用的是 N[表达式 , 整数] 可以求出表达式精确到指定有效数字的数值解，还有如数值求积分的 NIntegrate[], 求方程数值根的 NSolve[] 和 NDSolve[]，最小、最大值的 NFindMinimum[] 和 NFindMaximum[]等等。

Mathematica 还有各种表达式操作的函数，如取分子、分母的 Numerator[expr] , Denormator[expr], 取系数的 Coefficient[expr], 因式分解的 Factor[expr], 以及展开的 Expand[expr] 和 ExpandAll[expr], 表达式化简的 Simplify[expr] 等。 expr 代表一个任意的表达式。

- 求极限

计算函数极限 $\lim_{x \to x0} f(x)$ 的一般形式是 ：

Limit[expr,x->x0] x->x0 时函数的极限

Limit[expr,x->x0,Direction->-1] x-> $x0^-$ 时函数的极限

Limit[expr,x->x0, Direction->1] x-> $x0^+$ 时函数的极限

In[1]:= `Limit[Sin[x]/x, x→0]`

Out[1]:=1

- 微商和微分

在 Mathematica 中能方便地计算任何函数表达式的任意阶微商 (导数). 如果 f 是一元函数,D[f,x] 表示 $\frac{df(x)}{dx}$; 如果 f 是多元函数 ,D[f,x] 表示 $\frac{\partial}{\partial x}f$. 微商函数的常用形式如下 :

D[f,x] 计算偏导数 $\frac{\partial}{\partial x}f$

In[1]:=D[x^x,x]

Out[1]:=

下面列出全微分函数 Dt 的常用形式及其意义 :

Dt[f] 全微分

Dt[f,x] 全导数

Dt[f,x1,x2,,] 多重全导数

In[1]:=Dt[x^2+y^2]

Out[1]:=

- 不定积分和定积分

1. 不定积分

Integreate 函数主要计算只含有 1“简单函数”的被积函数 . “简单函数”包括有理函数、指数函数、对数函数和三角函数与反三角函数。不定积分一般形式如下：

Integrate[f , x] 计算不定积分

Integrate[f , x, y] 计算不定积分

Integrate[f , x, y, z] 计算不定积分 $\int dx \int dy \int f(x,y,z)dz$

In[1] :=
```
Integrate[1/(x^2-1), x]
```

Out[1] := $\frac{1}{2}\text{Log}[-1+x]-\frac{1}{2}\text{Log}[1+x]$

In[2] :=
```
Integrate[3x^2+y, x, y]
```

Out[2] := $x^3 y+\frac{x y^2}{2}$

2. 定积分

计算定积分的命令和计算不定积分是同一个 Integrate 函数，在计算定积分时，除了要给出变量外还要给出积分的上下限。当定积分算不出准确结果时，用 N[%]命令总能得到其数值解.Nintegrate 也是计算定积分的函数 , 其使用方法和形式和 Integrate 函数相同 . 用 Integrate 函数计算定积分得到的是准确解 , Nintegrate 函数计算定积分得到的是近似数值解. 计算多重积分时 , 第一个自变量相应于最外层积分放在最后计算 .

Integrate[f,{x,a,b}] 计算定积分 $\int_a^b f(x)dx$

NIntegrate[f,{x,a,b}] 计算定积分 $\int_a^b f(x)dx$

Integrate[f,{x,a,b},{y,c,d}] 计算定积分 $\int_a^b dx\int_c^d f(x,y)dy$

NIntegrate[f,{x,a,b},{y,c,d}] 计算定积分 $\int_a^b dx\int_c^d f(x,y)dy$

In[1]:=
```
Integrate[Cos[x]^2+Sin[x]^3, {x, 0, 1}]
```

Out[1]:= $\frac{7}{6} - \frac{3\,\text{Cos}[1]}{4} + \frac{\text{Cos}[3]}{12} + \frac{\text{Sin}[2]}{4}$

In[2]:=
```
NIntegrate[Cos[x]^2 + Sin[x]^3, {x, 0, 1}]
```

Out[2]:= 0.906265

In[3]:=
```
Integrate[x + y, {x, b, a}, {y, 0, x}]
```

Out[3]:= $\frac{3}{2}\left(\frac{a^3}{3} - \frac{b^3}{3}\right)$

- 幂级数

幂级数展开函数 Series 的一般形式：

Series[expr,{x,x0,n}] 将 expr 在 x=x0 点展开到 n 阶的级数

Series[expr,{x,x0,n},{y,y0,m}] 先对 y 展开到 m阶再对 x 展开 n 阶幂级数

用 Series 展开后，展开项中含有截断误差 $O[x]^n$

In[1]:=
```
Series[Sin[2 x], {x, 0, 6}]
```

Out[1]:= $2x - \frac{4x^3}{3} + \frac{4x^5}{15} + O[x]^7$

In[2]:=
```
Series[f[x], {x, o, 3}]
```

Out[2]:= $f[o] + f'[o]\,(x-o) + \frac{1}{2}f''[o]\,(x-o)^2 + \frac{1}{6}f^{(3)}[o]\,(x-o)^3 + O[x-o]^4$

In[3]:=
```
Series[Cos[x] Cos[y], {x, 0, 3}, {y, 0, 3}]
```

Out[3]:= $\left(1 - \frac{y^2}{2} + O[y]^4\right) + \left(-\frac{1}{2} + \frac{y^2}{4} + O[y]^4\right)x^2 + O[x]^4$

▪ 常微分方程

求解常微分方程和常微分方程组的函数的一般形式如下 :

Dsolve[eqns,y[x],x] 解 y(x) 的微分方程或方程组 eqns,x 为变量

Dsolve[eqns,y,x] 在纯函数的形式下求解

NDsolve[eqns,y[x],x,{xmin,xmax}] 在区间 {xmin,xmax} 上求解变量 x 的数的形式下求解常微分方程和常微分方程组 eqns 的数值解

In[1]:=
```
DSolve[y'[x] == a y[x], y[x], x
```

Out[1]:=
```
{{y[x] → e^(a x) C[1]}}
```

In[2]:=
```
DSolve[{y'[x] == a y[x], y[0] == 1}, y[x], x]
```

Out[2]:=
```
{{y[x] → e^(a x)}}
```

In[3]:=
```
DSolve[{x'[t] == y[t], y'[t] == x[t]}, {x[t], y[t]},
 t]
```

Out[3]:=
$$\left\{\left\{x[t] \to \frac{1}{2} e^{-t} \left(C[1] + e^{2t} C[1] - C[2] + e^{2t} C[2]\right),\right.\right.$$
$$\left.\left. y[t] \to \frac{1}{2} e^{-t} \left(-C[1] + e^{2t} C[1] + C[2] + e^{2t} C[2]\right)\right\}\right\}$$

▪ 线性代数

1. 定义向量和矩阵函数

定义一个矩阵，可用函数 Table 或 Array. 当矩阵元素能用一个函数表达式时 ，用函数 Table 在定义矩阵大小的同时也给每个矩阵元素定义确定的值 . 用函数 Range只能定义元素为数值的向量 .Array 只能用于定义向量、矩阵和张量 ，并规定矩阵和张量的元素下标从 1 开始 .Array 的一般形式 : Array[向量元素名 ,n,f] 定义下标从 f 开始的有 n 个元素的向量，当 f 是 1 时可省略 . Array[矩阵元素名 ,{m,n}] 定义 m行 n 列的矩阵 . 其中: 矩阵元素名是一个标识符 ，表示矩阵元素的名称 ，当循环范围是 {u,v,w} 时定义一个张量 . Table[表达式 f, 循环范围] 表达式 f 表示向量或矩阵元素的通项公式 ；循环范围定义矩阵的大小 . 循环范围的一般形式 :{ 循环变量名，循环初值，循环终值，循

环步长 }.　在 Array　或 Table 的循环范围表示方法略有区别　. 请在下面的实例中注意观察.

```
In[1]:=  Table[a[i, j], {i, 2}, {j, 2}]

Out[1]:=  {{a[1, 1], a[1, 2]}, {a[2, 1], a[2, 2]}} (* 矩阵每一行元素用一对　{} 括起来 *)

In[2]:=  U = Array[a, {2, 2}]

Out[2]:=  {{a[1, 1], a[1, 2]}, {a[2, 1], a[2, 2]}}

In[3]:=  IdentityMatrix[3] (*IndentityMatrix[n]　　生成　n 维矩阵 *)

Out[3]:=  {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}

In[4]:=  DiagonalMatrix[{1, 2, 3}] (* 生成对角元素为表元素的对角矩阵　 *)

Out[4]:=  {{1, 0, 0}, {0, 2, 0}, {0, 0, 3}}

In[5]:=  TableForm[%] (*TableForm[m]　或 MatrixForm[m]　按矩阵形式输出　m*)

Out[5]:=
1   0   0
0   2   0
0   0   3
```

一个矩阵可用一个变量表示　, 如 In[2]　所示 U是一个矩阵 , 则 U[[I]]　表示 U的第 I 行的 N个元素 ;Transpose[U][[j]]　　表示 U的第 J 行的 M个元素 ;U[[I,j]]　　或 a[I,j]　表示 U的第 I 行第 J 列元素 ;U[[{i1,i2,,,ip},{j1,j2,,,jq}]]　　表示由行为　{i1,　i2,,,ip}　和列为{ j1,j2,,,jq}　　组成的子矩阵　.

2. 矩阵的运算符号和函数

表达式	意义
A+c	A 为矩阵 ,c　为标量 ,c　与 A 中的每一个元素相加
A+B	A,B 为同阶矩阵或向量　,A 与 B 的对应元素相加

cA	A为矩阵 ,c 为标量 ,c 与 A中的每个元素相乘
U.V	向量 U与 V的内积
A.B	矩阵 A与矩阵 B相乘 , 要求 A的列数等于 B的行数
Det[M]	计算矩阵 M的行列式的值
Transepose[M]	M的转置矩阵 (M^T 或 M')
Inverse[M]	计算矩阵 M的逆矩阵 (M^{-1})
Eigenvalus[A]	计算矩阵 A的全部 (准确解) 特征值
Eigenvalus[N[A]]	计算矩阵 A的全部 (数值解) 特征值
Eigenvectors[A]	计算矩阵 A的全部 (准确解) 特征向量
Eigenvectors[N[A]]	计算矩阵 A的全部 (数值解) 特征向量
Eigensystem[A]	计算矩阵 A的所有 (准确解) 特征值和特征向量
Eigensystem[N[A]]	计算矩阵 A的所有 (数值解) 特征值和特征向量

3. 方程组求解函数

在 Mathematica 中用 LinerSolve[A,B], 求解满足 AX=B的一个解 . 如果 A的行列式不为零 , 那么这个解是方程组的唯一解 ; 如果 A的行列式是零 , 那么这个解是方程组的一个特解 , 方程组的全部解由基础解系向量的线性组合加上这个特解组成 . NullSpace[A] 计算方程组 AX=0的基础解系的向量表 , 用 LinerSolve[A,B] 和 NullSpace[A] 联手解出方程组 AX=B的全部解 . Mathematica 中还有一个美妙的函数 RowReduce[A],它对 A的行向量作化间成梯形的初等线性变换 . 用 RowReduce可计算矩阵的秩 , 判断向量组是线性相关还是线性无关和计算极大线性无关组等工作 .

解方程组函数	意义
RowReduce[A]	作行的线性组合化简 A,A 为 m行 n 列的矩阵
LinerSolve[A,B]	求解满足 AX=B的一个解 ,A 为方阵
NullSpace[A]	求解方程组 AX=0的基础解系的向量表 , A 为方阵

例: 已知 A= $\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 1 \\ 3 & 1 & -1 & 3 \\ 3 & 2 & 1 & 3 \end{matrix}$, 计算 A 的秩, 计算 AX=0的基础解系 .

In[1]:=
```
A = {{1, 1, 1, 1}, {1, 0, -1, 1}, {3, 1, -1, 3}, {3, 2, 1, 3}};
```
In[2]:=
```
RowReduce[A]
```

Out[2]=
```
{{1, 0, -1, 1}, {0, 1, 2, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}
```
(* 显然,A 的秩是 2*)

In[3]:=
```
NullSpace[A]
```

Out[3]=
```
{{-1, 0, 0, 1}, {1, -2, 1, 0}}
```
(*A 的两个线性无关解 *)

五. 程序流程控制

循环语句有 For[赋初值，循环条件，增量语句，语句块] 表示如果满足循环条件，则执行语句块和增量语句，直到不满足条件为止， While[test,block] 表明如果满足条件 test 则反复执行语句块 block，否则跳出循环， Do[block,{i,imin,imax,istep}] 与前者功能是相同的。还有 Goto[lab], Label[lab] 提供了程序中无条件跳转， Continue[] 和 Break[] 提供了继续循环或跳出循环的控制， Catch[语句块 1] 和 Throw[语句块 2] 提供了运算中对异常情况的处理。另外，在程序中书写注释可以用一对 "(* *)" 括起来，注释可以嵌套。

六. 其他

1. 使用帮助， Mathematica 的帮助文件提供了 Mathematica 内核的基本用法的说明，十分详细，可以参照学习。

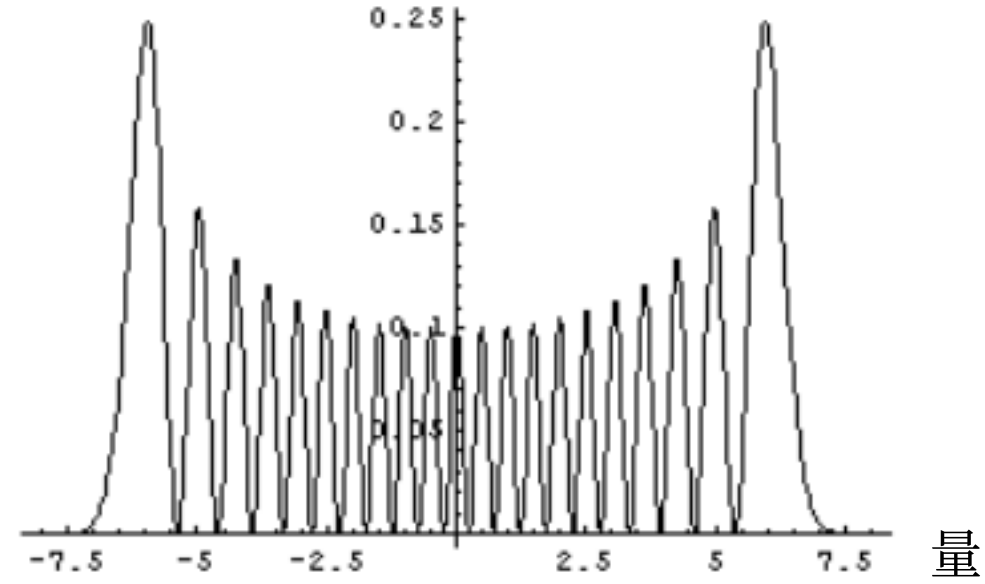
2. 你可以使用 "? 符号名 " 或"?? 符号名 " 来获得关于该符号 (函数名或其他) 的粗略或详细介绍。符号名中还可以使用通配符， 例如 ?M*，则系统将给出所有以 M开头的关键词和函数名，再如 ??For 将会得到关于 For 语句的格式和用法的详细情况。

3. 在 Mathematica 的编辑界面中输入语句和函数，确认光标处于编辑状态 (不断闪烁) ，然后按 Insert 键来对这一段语句进行求值。如果语句有错，系统将用红色字体给出 出错信息，你可以对已输入的语句进行修改，再运行。如果运行时间太长，你可以通过 Alt+.(Alt+ 句号) 来中止求值。

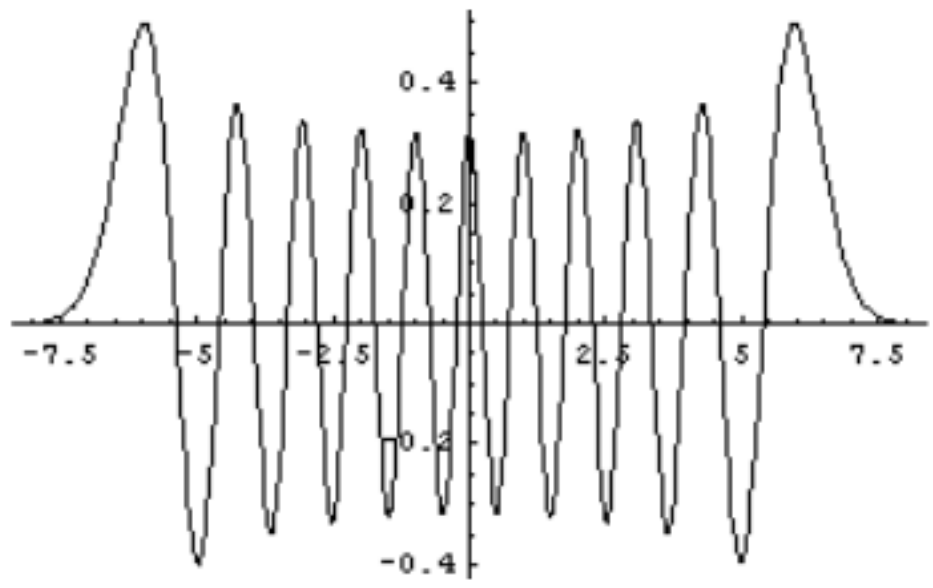
4. 对函数名不确定的，可先输入前面几个字母 (开头一定要大写) ，然后按 Ctrl+K ，系统会自动补全该函数名。

七. 应用例子

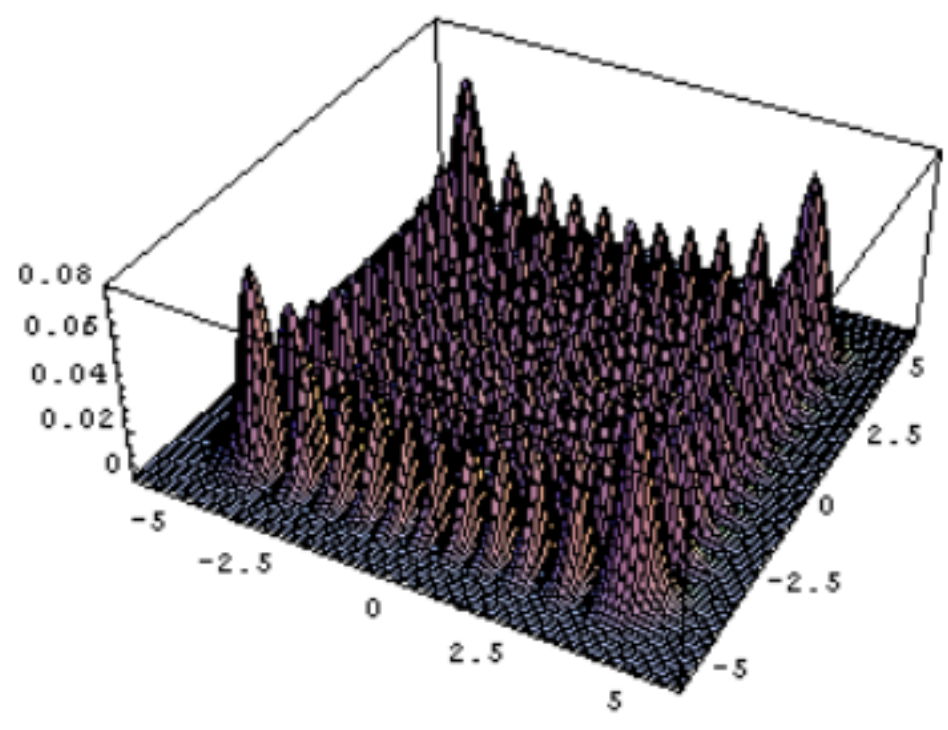
```
In[6]:= H[ζ_, n_] := (-1)^n × (2^n × n! × √π)^(-1/2) × e^(ζ^2) ×
          D[e^(-ζ^2), {ζ, n}];
        ψ[x_, n_] := e^(-1/2 x^2) H[x, n];
        ψ[x, 20];
        Plot[Abs[%]^2, {x, -8, 8}, PlotRange → All,
         PlotPoints → 100];
        Plot[%%, {x, -8, 8}, PlotRange → All,
         PlotPoints → 100];
        ψ[x, 10] × ψ[y, 10];
        Plot3D[Abs[%]^2, {x, -6, 6}, {y, -6, 6},
         PlotRange → All, PlotPoints → 100];
        Show[%, LightSources →
          {{{10, 10, 10}, GrayLevel[0.99]}},
         ViewPoint -> {-5.032, 2.590, 1.338}];
```



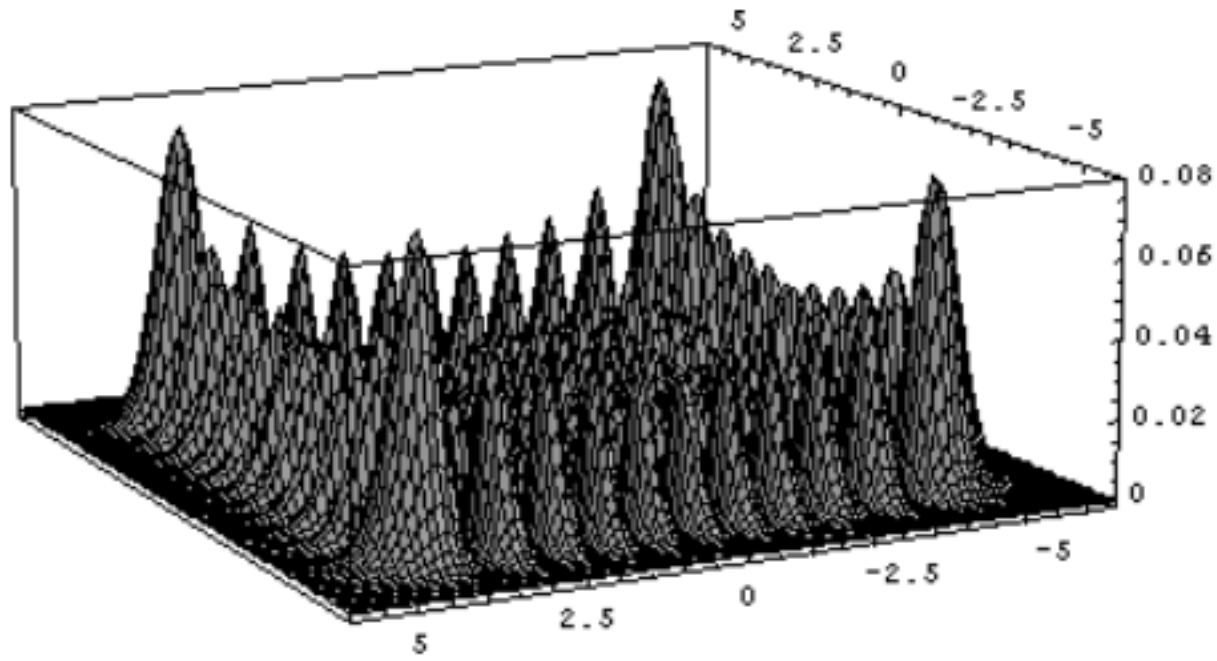
量子一维、二维简谐振子问题



量子一维简谐振子图像



量子二维简谐振子图像



一 . 计算机代数系统

二 .Matlab

三 .Mathematica

四 . 统计软件　SAS

五 . 运筹学软件及教程　(LINDO,LINGO)