Matlab

在基础物理学中的应用

(内部讲义)

贾砚宾　管　靖　编

北京师范大学物理系

2004. 10

Matlab 在基础物理学中的应用

这是选读内容，由教师组织学习当然更好，同学自学也是可以的. 老师不一定是专家，韩愈说过: “师不必贤于弟子，弟子未必不如师. ”“玩”计算机，正是年轻人的特长. 在这里，你有在大学一年级就超越老师的机会.

计算机是“玩”会的，你拿出玩电子游戏的热情与专注，很快就会成功. Matlab 在科研和工程设计中被广泛使用，它几乎可以做你想做的所有事，学会了你可以终生受益，这比玩电子游戏有价值得多. 现在，我们就可以借助它学习物理学.

你不要学了再做，要在做中去学. 装上软件，边看书边实践. 先按书输入程序，让它能运行. 再模仿它，做一点点修改……发挥你的创造性，做你想做的事……

这里只是入门，入门后有很多书可看. 遇到问题，最快捷的方法是向别人请教.

Matlab 有很好的帮助系统（§7 之 3），但它是英文界面，开始看不懂是有情可原的.

Matlab 是 “Matrix Laboratory”的缩写，是一种功能强、效率高、便于进行科学和工程计算的交互式软件包.

由于使用 Matlab 编程运算与人进行科学计算的思路和表达方式完全一致，所以不象学习其它高级语言那样难于掌握. 实践证明，学习者可在短短几个小时的学习和使用中就能初步掌握 Matlab 的基础知识，从而使学习者能够进行高效率和富有创造性的计算.

Matlab 对于学习者的编程语言基础要求不高，库函数和编程语句丰富多样且简单易学，在数据可视化上也有独特的优势. 学习者不需要投入太多的时间在学习编程语言知识上，可以直接利用软件提供的丰富的函数，编写较简单的程序即可解决许多普通物理学的问题.

下面将结合实例，简要介绍 Matlab 的基础知识和在普

通物理学中的应用.

注意：带*的内容开始不必看，使用时再查阅.

§1 Matlab 使用介绍

在 Windows 窗口中用鼠标双击 Matlab 图标即可进入 Matlab 的工作窗口（Command Window），如图 1 所示. 没有图标可利用 Matlab\bin 目录下的 Matlab.exe 文件在桌面上建立一个快捷方式.

退出 Matlab 的方法有三种：单击工作窗口右上角的关闭按钮；用菜单 File→Exit Matlab 命令；或者直接在工作窗口中输入 quit 后回车.

工作窗口是标准的 Windows 窗口形式，用户在命令窗口中输入各种指令，进行运算；在左侧的变量窗口中监控当前所创立的所有变量.

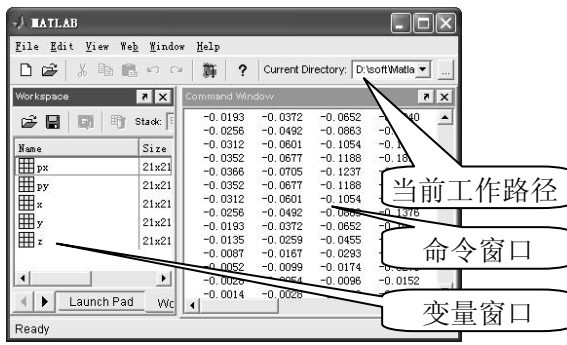


图 1　Matlab 工作窗口

Current Directory 是系统的当前工作路径，Matlab 对函数或文件等进行搜索，用户每次文件的创建、保存都在这个路径下进行. 初次启动 Matlab 时系统的默认工作路径是 Matlab 目录下的 Work 子目录，如果要改变当前的工作路径，

可以单击如图 2 所示的路径栏右侧的…，在弹出的路径选择对话框内选择想要设置的路径.



图 2　工作路径栏和选择工作路径对话框

§2 Matlab 能做什么

本节先给出使用 Matlab 辅助普通物理学学习的几个简单的例子，意在引导读者对 Matlab 的功能特点及语句编程有一个概括的了解.

（1）绘制简谐振动的振动曲线

简谐振动的运动学方程是 $x = A\sin(\omega t + \varphi)$，可以根据这个方程利用 Matlab 的绘图语句画出质点简谐振动的位移曲线.

在命令窗口中输入：（lt21.m）

```
a=1;     omiga=1;                    %为变量赋值
t=0:0.05*pi:4*pi;                    %为变量赋值
phi=0.1*pi;                          %为变量赋值
x=a*sin(omiga*t+phi);                %建立表达式并运算
plot(t,x)                  %以 t 为横坐标，x 为纵坐标绘图
```

运行结果如图 3 左图所示.

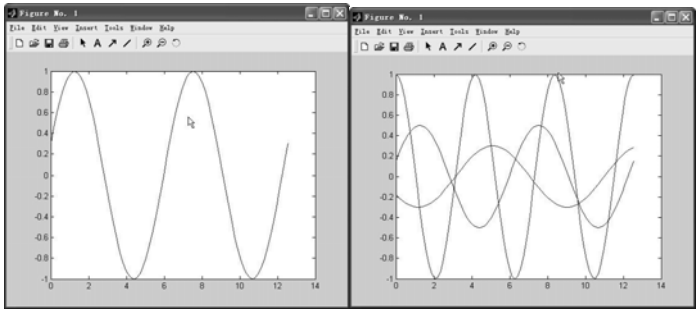


图 3　简谐运动的位移曲线

在这个例子中，a 是振幅 A，omiga 是圆频率 ω，phi 是初相位 φ，t 是自变量，开始为这些参量以及自变量赋值. 可以看出，Matlab 不像其他编程语言那样必须进行变量的预定义，创建变量和为变量赋值是同时完成的. 另外，对时间 t 的赋值是一组数据，即从 0 到 4π 每隔 0.05π 取一个值，形成一个数组：t=[0 0.1571 0.3142 … 12.2522 12.4093 12.5664]（数组中的数用空格或逗号分开），总共有 81 个元素. 语句中的 pi 是 Matlab 默认的常数 π.

x=a*sin(omiga*t+phi)是建立表达式并运算，sin（）函数是 Matlab 的内置函数，直接调用即可，由此得出对应每一个 t 值的质点的位移 x，x 是一个与 t 所含元素个数相同的数组.

最后的 plot(t,x)是二维绘图语句，表示以 t 为横坐标，x 为纵坐标绘图，得出的就是正弦曲线. 如果对各个参量取不同的值，得出的曲线形状也就不一样. 图 3 右图是选择不同参量时的曲线对比情况.

对于数组 t 的赋值，一般格式为 t=t0:tstep:tend. t0 为数组中第一个元素的值，tend 为最后一个元素的值，tstep 为取值的间隔.

Matlab 会根据计算的结果自动调整坐标轴范围，当然用户也可以通过语句对坐标轴进行控制.

（2）等量异号点电荷的电势分布

这个例子将介绍二维网格和三维曲面绘图的语句. 物理情景是 *Oxy* 平面上在 *x*=2, *y*=0 处有一正电荷，*x*=-2, *y*=0 处有一负电荷，根据 $U=\frac{q}{4\pi\varepsilon_0 r}$ 计算两点电荷电场中电势的分布，$r=\sqrt{(x-x_0)^2+(y-y_0)^2}$.

在命令窗口中输入：（lt22.m）

```
[x,y]=meshgrid(-5:0.2:5,-4:0.2:4);          %建立数据网格
z=1./sqrt((x-2).^2+y.^2+0.01)-1./sqrt((x+2).^2+y.^2+0.01);
                                            %电势的表达式
mesh(x,y,z)                                 %三维曲面绘图
```

运行结果如图 4 所示.

选定一系列的 x 和 y 后，就组成了平面上的网格点，再计算对应每一点上的 z 值. -5:0.2:5,-4:0.2:4 分别是选取横坐标与纵坐标的一系列数值，meshgrid 是生成数据网格的命令， [x,y]是 *xy* 平面上的坐标网格点.

z=1./sqrt((x-2).^2+y.^2+0.01)-1./sqrt((x+2).^2+y.^2+0.01) 是场点（x, y）的电势，其中 sqrt( )是 Matlab 默认的函数：求变量的平方根. 当场点即在电荷处时，会出现分母为零的情况，因此在 *r* 里加了一个小量 0.01，这样既可以完成计算，又不会对结果的正确性造成太大影响.

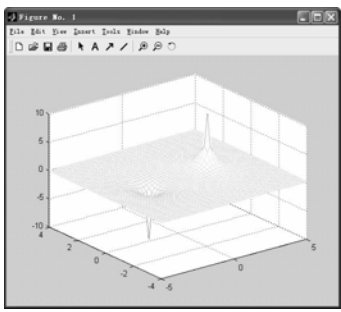


图 4 等量异号点电荷的电势分布

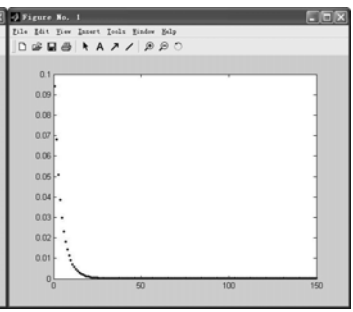


图 5 λ=0.8 时的叠代情况

另外需要注意的是表达式中的“./”、“.^”是对数组运算的算符，含义与数值运算中的“/”、“^”相同，不同之处是后者只对单个数值变量进行运算，而前者对整个数组变量中的所有元素同时进行运算.

mesh 是三维网格作图命令，mesh(x,y,z)画出了每一个格点（x, y）上对应的 z 值（电势）.

（3）Logistic 叠代图

$x_{n+1} = \lambda x_n (1 - x_n)$ 是一个著名的非线性叠代方程，称为Logistic叠代，给定一个初值x_0（$0<x_0<1$）和系数λ（$0<\lambda<4$）后，进行n次叠代，研究在给定系数情况下x_n的变化趋势.

在命令窗口中输入：（lt23.m）

```
x0=abs(sin(randn));          %取 x0 为[0，1]间的随机数
lamda=0.8;                   %赋 λ 值
xn=x0;                       %将 x0 赋给叠代变量 xn
for n=1:150
        %以 n 为循环变量开始循环叠代，进行叠代 150 次
    xn=lamda*xn*(1-xn);   %叠代公式
    plot(n,xn, '.');
      %画出每次叠代出的 xn 值，横坐标为叠代次数
    hold on                  %保持已画出的图
end
```

运行结果如图 5 所示：

第一条语句是取初值x_0，为了保证随机数x_0在[0,1]内，使用了x0= abs(sin(randn)). 其中abs( )是求绝对值，randn是生成一个随机数.

程序中用了 for 循环结构，循环变量为 n. for 循环结构的形式为

```
for n=N0:Nstep:Nend
  循环体
end
```

for n=1:150，就是从 n=1 开始，把 x0 代入公式，把得到的结果 x1 描点，然后令 x0=x1，再作为初值代入公式进行计算，直到 n=150 结束. 这里的 Nstep 是取值间隔，默

认为 1，用户也可以取其它整数.

hold on 是作图控制，即在坐标图上绘出新的点的同时，也要保留上一次循环已经绘出来的点. 如果没有这一句，最终图上只会留下最后一次叠代结果对应的点.

§3 数据表示　运算符和表达式

数据、算符和表达式是学习任何一门编程语言都要首先掌握的基础知识. 本节就介绍数据、算符和表达式的写法.

（一）数据格式及表示

Matlab 使用常规的十进制表示法，小数位数不限，可以用加号和减号表示正负数. 10 的幂用 e 加上数字表示. 虚数单位是 i 和 j，数字后直接加上 i 或 j 表示虚数，中间不要有空格或者乘号；但表达式中要在变量名和虚数单位间加上*号. 以下都是合法的数值表示法.

$$\begin{array}{ccc} 2 & -88 & 0.3201 \\ 9.63972 & 1.02e-20 & 6.02e23 \\ 5+3i & -3.242i & 3e5i \end{array}$$

（二）运算符和操作符

（1）一般运算符

单个数据的运算有加、减、乘、除、幂和括号，所用的算符分别是：

\+　-　*　/　^　（）

（2）操作符

①冒号“：”此符号在矩阵（数组）的构造和运算中非常有用，可以用来产生矢量（数组），其基本用法有：

j:k　等价于[j,j+1,……,k]　　若 j>k 则返回空值

j:i:k　等价于[j,j+i,j+2*i,……,k]　i>0 则要求 j<k，i<0 则要求

j>k，否则返回空值

此外，冒号还可以用作矩阵的下标，以及部分的选择矩阵的元素，执行循环操作等，后面几节中会陆续介绍.

②百分号“%”此符号在命令行中表示注释，即在一行中百分号后面的语句都被忽略而不被执行.

③连续点“…”如果一条命令很长，一行容不下，可以用 3 个点加在一行的末尾，表示此行未完，而在下一行继续. 例如在建立等量异号点电荷电场中电势的表达式时，可以这样写：

```
z=1./sqrt((x-2).^2+y.^2+0.01)…
-1./sqrt((x+2).^2+y.^2+0.01);
```

这样虽然占了两行，但 Matlab 仍然认为这是一条语句，不会出现错误.

④分号“；” 分号用在每行命令的结尾，要求执行命令但不显示计算结果.

＊（3）关系运算符

关系运算符主要用于在数与数、矩阵与矩阵之间进行比较，基本的有：

运算符	<	>	<=	>=	==	~=
功能	小于	大于	小于等于	大于等于	等于	不等于

＊（4）逻辑运算符

在 Matlab 中包含与“&”、或“|”、非“~”、异或“xor”4 种逻辑运算符.

（三）变量与表达式

用运算符把数字、变量和函数组合在一起，就建立了一个表达式. 例如前面例子中的 x0=abs(sin(randn))，x=a*sin(omiga*t+phi)等. 在 Matlab 中，一个变量可以通过给它分配一个数值或表达式来定义，如下所示：

variable =*expression*

一个变量的值可以通过输入它的名字=值（或

表达式），并按回车键获得，Matlab 以显示这个变量的名字和值作为回答. 如果这个变量并不存在，就显示一个错误信息，如图 6 所示.

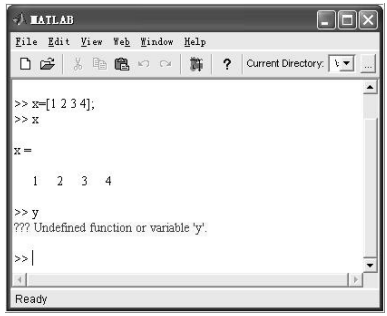


图 6　显示变量

在 *expreesion* 之后可以加分号后按回车键，也可以直接按回车键. 没有结尾分号的每个命令在屏幕上显示出其结果；若结尾带分号，就执行计算，但计算结果并不显示.

如果不指定变量而直接输入 *expression* 项，则 Matlab 用 ans（ answer 的缩写）显示这个值.

***（4）管理变量**

当创建了很多不同类型的变量后，就需要获得当前工作窗口中的变量的信息，并清除不再需要的变量. Matlab 提供了一些命令专门用来管理变量.

名称	含义
who,whos	列出当前工作窗口的变量
clear	从内存中清除变量和函数
size	获得矩阵的尺寸（矩阵的维数、元素个数等）
length	获得矢量的长度
disp	在显示变量名称的同时显示变量的内容

***（5）基本数学函数**

Matlab 中有许多已定义好的数学函数，用户可以直接调用，例如前面例子中的 sin，randn，abs 等，常见的数学函数可参考下表.

①三角函数与双曲函数

名称	含义	名称	含义	名称	含义
sin	正弦	sec	正割	asinh	反双曲正弦
cos	余弦	csc	余割	acosh	反双曲余弦
tan	正切	asec	反正割	atanh	反双曲正切
cot	余切	acsc	反余割	acoth	反双曲余切
asin	反正弦	sinh	双曲正弦	sech	双曲正割
acos	反余弦	cosh	双曲余弦	csch	双曲余割
atan	反正切	tanh	双曲正切	asech	反双曲正割
acot	反余切	coth	双曲余切	acsch	反双曲余割

②对数函数

名称	含义	名称	含义	名称	含义
Exp	*e* 为底的指数	log10	10 为底的对数	pow2	2 的幂
Log	自然对数	log2	2 为底的对数	sqrt	平方根

③复数函数

名称	含义	名称	含义	名称	含义
Abs	绝对值	conj	复数共轭	real	复数实部
Angle	相角	imag	复数虚部		

④统计函数

名称	含义	名称	含义
min	最小值	max	最大值
mean	平均值	median	中位数
sum	总和	diff	相邻元素的差
sort	排序	length	个数

§4 矩阵与数组

（一）定义矩阵

Matlab 中数据的基本格式是矩阵. 2 维矩阵是一个以行和列排列的元素矩形表. 如果有 m 行、n 列，这个矩阵的大小就是 $m \times n$.

如一个 2×3 的矩阵：$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.

第 1 行是[1 2 3]，第 2 列是$\begin{bmatrix} 2 \\ 5 \end{bmatrix}$.

这里之所以说到矩阵，是因为 Matlab 就是矩阵试验室. 实际上读者没学过矩阵也没关系，等你学过以后，Matlab 会显示出更强大的功能. 现在不必去学矩阵，我们先把矩阵理解为或一行、或一列、或 m 行 n 列的数组，我们也主要只进行数组的运算. 矢量在三维空间里有三个分量，现在我们把矢量推广为有多个分量；如果数组只有一行，它就是一个行矢量；如果数组只有一列，它就是一个列矢量. 下面我们就使用矩阵、矢量进行讲解，你把它想成数组就行了.

矩阵的元素，即数a_{ij}，通常是实数，但也可以是复数. a_{ij}是指第i行、第j列的数. 在上例中，有a_{21}= 4.

当矩阵仅由一行组成时，它是一个特例，就是一个行矢量. 如果矩阵仅有一列,就是一个列矢量. 矢量中元素的数量就是矢量的长度. 如果矩阵的维数是 1，它是一个标量，就是一个数.

二维矩阵的建立可以有多种方法实现，最简单的方法是由方括号[ ]包围的逐行给定元素. 相同行中的元素用空格‘’或逗号‘,’分隔，不同行由分号‘;’或回车键分隔，如图 7 所示. 如果定义一个标量，则方括号就不需要了.

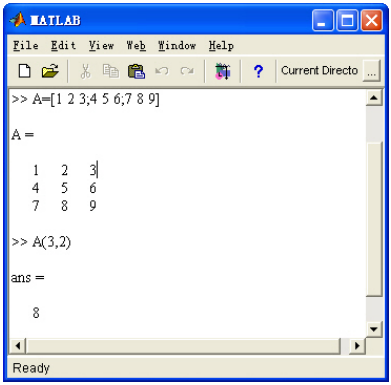


图 7 矩阵建立和元素的引用

（二）矩阵元素的标识与修改

一个矩阵 A（或一个矢量 A）的指定元素用 A(i,j)来表示，例如 A(3,2)就表示矩阵 A 的第三行第二列的元素. 可以单独引用某个矩阵元素，并且对其直接赋值或建立表达式进行修改.

*（三）一些特殊矩阵的生成指令

建立 1 矩阵使用 ones 命令，这种矩阵的元素全部都是 1. 建立 0 矩阵使用 zeros 命令，这种矩阵的元素全部都是 0. 单位矩阵的对角线元素全部是 1，而其他元素全部是 0，建立它使用 eye 命令.

ones(n)	建立一个 $n\times n$ 的 1 矩阵.
ones(size(A))	建立一个和矩阵 A 同样大小的 1 矩阵.
zeros(n)	建立一个 $n\times n$ 的 0 矩阵.
zeros(size(A))	建立一个和矩阵 A 同样大小的 0 矩阵.
eye(n)	建立一个 $n\times n$ 的单位矩阵.
eye(size(A))	建立一个和矩阵 A 同样大小的单位矩阵.

（四）数组与数组运算

对于一个矩阵 A，Matlab 既可以进行矩阵运算，也可以进行数组运算. 所谓矩阵运算，就是线性代数中规定的矩阵所特有的运算，例如矩阵的乘法，求矩阵的逆，特征值等等；而数组运算，则是把参与运算的二维矩阵看作二维的数组，对矩阵中的对应元素进行相同的运算，运算结果到一个新的矩阵，新矩阵和原矩阵中的元素一一对应. 这种数组运算使得对大量的数据进行相同的运算变得十分方便快捷，**我们首先掌握数组运算**.

数组运算的一般运算符与单个数值运算的一般运算符相似，相应的有

\+　　-　　.*　　./　　.^　.\　( )

含义与数值运算的算符相同.

例：比较“*”和“.*”的不同.

```
>> A=[1,2,3;4,5,6;7,8,9];
>> B=[1,1,1;2,2,2;3,3,3];
>> D=A.*B
D =
     1     2     3
     8    10    12
    21    24    27
>> d=A*B
d =
    14    14    14
    32    32    32
    50    50    50
```

可见，两种算符得出的结果是不一样的，写表达式时一定要分清要做哪种运算. **现在我们所做的通常是数组运算**. 参与运算的矩阵的维数必须相同，否则则会报错. 例如：

```
>> A=[1,2,3;4,5,6]
A =
     1     2     3
     4     5     6
>> B=[1,2;3,4;5,6]
B =
     1     2
```

```
     3     4
     5     6
>> A.*B
??? Error using ==> .*
Matrix dimensions must agree.
```

在Matlab中预定义的数学函数是基于矩阵的数组运算. 也就是说函数f对于矩阵A，有$f(A)_{ij}=f(A_{ij})$，矩阵的维数没有改变，例如：

```
A =
     1     2     3
     4     5     6
>> sin(A).*cos(A).^2
ans =
    0.2456    0.1575    0.1383
   -0.3233   -0.0772   -0.2576
```

§5 编程

Matlab 输入命令的方式有两种，一种就是像前文所例举的在命令窗口中直接输入简单的语句，这种方式适应于命令比较简单、且处理的问题没有普遍应用性、差错处理比较简单的场合. 但是在进行大量重复性的计算时，或者语句结构比较复杂需要进行流程控制时，这种方式就显得不太灵活，如果输入错误则很不好改正. 因此需要借助另一种工作方式：M 文件的编程工作方式. M 文件是一个简单的文本文件，语法比一般的高级语言都要简单，程序容易调试，交互性强；而且可以像一般文本文件那样在任何文本编辑器中进行编辑、存储、修改和读取（输入时用英文,不可用中文标点,注意空格）. 前面所举的所有示例都可以编辑为 M 文件.

（一）创建并编辑 M 文件

尽管 M 文件可以用任何文本编辑器（例如记事本，写字板等程序）创建和编辑，Matlab 还是提供了一个方便实用的 M 文件编辑器，利用这个编辑器，用户可以完

成程序的创建、编辑、调试、存储和运行等工作.

在 Matlab 命令窗口中输入“edit”并回车，就调出了如图 8 所示的 M 文件编辑器（编辑窗口）.

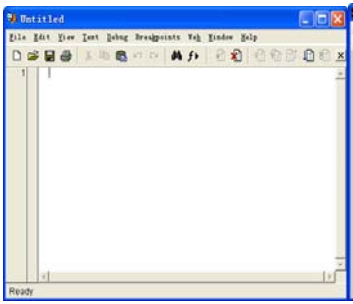


图 8 M 文件编辑器

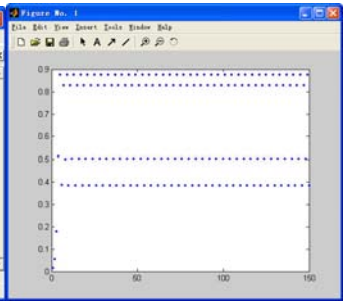


图 9 修改参数 λ 后的运行结果

在 M 文件编辑器的菜单栏里选择“File→New→M-file”命令，或单击工具栏上的按钮，可以创建一个新的 M 文件；选择“File→Open”命令，或者单击工具栏上的按钮，可以打开工作路径上的已有的 M 文件；当编辑完 M 文件后，在菜单栏里选择“File→Save”命令或者单击工具栏上的按钮，可以保存文件. 此外，在 Matlab 主窗口中的菜单栏和工具栏上，也有相应的命令和按钮来创建和编辑 M 文件.

例 1：将§2 中的 Logistic 叠代图的语句编辑成 M 文件并运行.

启动 M 文件编辑器，创建一个新的 M 文件，在编辑窗口中键入下列语句:

```
x0=abs(sin(randn));
lamda=0.8;
xn=x0;
for n=1:150
    xn=lamda*xn*(1-xn);
    plot(n,xn,'.');
    hold on
end
```

输入完毕后保存文件为 Logistic.m，放在当前的工作目录下.

在 Matlab 的命令窗口中输入“Logistic”并回车，则运行

此程序，显示的结果与§2 节中相同. 用户随时可以更改 M 文件内容并重新运行，例如将“lamda=0.8”改为“lamda=3.5”， **保存文件**并再次运行此程序，则所得结果如图 9 所示.

读者可以将前面所举的例子改为以 M 文件的形式来完成.

（二）脚本文件与函数文件

M 文件有两种形式，即脚本文件（也称命令文件）和函数文件. 脚本文件就是前面所举例中的 M 文件形式，是命令和语句的简单叠加，Matlab 会自动按顺序执行文件中的命令. 函数文件主要用以解决参量传递和函数调用的问题，它的第一句以 function 语句为引导.

需要注意的是命令文件在运行过程中可以调用 Matlab 工作域内所有的数据，而且所产生的所有变量均为**全局变量**. 也就是说，这些变量一旦生成，就一直保存在内存空间中，直到用户执行“clear”命令或退出 Matlab 时才会被清除. 而函数文件中所有变量除特殊声明外，均为**局部变量**，即只在调用函数时存在，一旦函数调用结束，这些变量立即被清除.

按下“F5”键或单击工具栏上的按钮，可以运行该程序.

例 2：将 Logistic 方程以函数文件形式完成.

在 M 文件编辑器中输入以下语句：

```
function f=logistic(lamda,N)
x0=abs(sin(randn));
xn=x0;
for n=1:N
    xn=lamda*xn*(1-xn);
    plot(n,xn, '.');
    hold on
end
```

保存文件为 logistic.m，在命令窗口中输入 logistic(3.5,150)，所得结果与图 9 相同.

在本例中，第一行语句必须以 function 开头，f 是函数的返回值（可以是空值），logistic 是函数名，lamda 和 N 是参量，在调用此函数文件时由用户给出. 之后的语句与前面例子相同，只是不再在程序中对 lamda 和循环次数 N 赋值了.

调用函数文件的命令是 logsitic(3.5,150)，即向函数传递两个参量（λ 和 N），表示当 λ=3.5 时叠代 150 次，调用函数得出结果. 写成函数文件的好处是不需要对 M 文件本身进行修改，只需在调用函数时就可以随时改变参量的值，得到不同情况下的结果，十分方便. 例如在命令窗口中输入 logistic(3.2,200)，就是求 λ=3.2 时叠代 200 次的结果.

需要注意的是要保存函数文件的文件名与首行语句中的函数名要一致，才能保证成功调用函数.

（三）流程控制语句

对于比较复杂的任务，往往需要设计一定的算法，运用流程控制的语句才能完成，因此有必要熟悉 Matlab 中的流程控制语句和一些算法知识. 程序一般分为顺序结构、循环结构、分支结构 3 种，在理论上，只要有以上三种结构就可以构造功能强大的程序. 下面只介绍顺序结构 for 循环结构.

（1）顺序结构

顺序结构就是依次顺序执行程序的各条语句. 语句在程序文件中的物理位置就反映了程序的执行顺序. 例如绘制等量异号点电荷电场的电势分布的语句就是典型的顺序结构：

```
[x,y]=meshgrid(-5:0.2:5,-4:0.2:4);          %建立数据网格
z=1./sqrt((x-2).^2+y.^2+0.01)-1./sqrt((x+2).^2+y.^2+0.01);
                                            %电势的表达式
mesh(x,y,z)                                 %三维曲面绘图
```

（2）for 循环结构

循环是计算机解决问题的主要手段，许许多多实际问题大都包含有规律性的重复计算和对某些语句的重复执行.

for 循环将循环体中的语句重复执行给定的次数，循

环的次数一般情况下是已知的，除非用其它的语句将循环提前结束. 在前面介绍 Logistic 叠代的例子时就已经接触到 for 循环了，其结构为：

```
for i=表达式
   可执行语句 1
   ……
   可执行语句 n
end
```

表达式是一个数组，可以为 m:s:n，m,s,n 一般情况下为整数，也可以取小数. 此数组中的元素被逐一赋值给 i，对于每个 i 的不同取值都要执行一次循环体内的语句. 表达式也可为 m:n，此时默认的步长为 1.

for 循环的循环体中，可以多次嵌套 for 和其他的结构体，这在一些情况下是非常有用的，大大扩展了 for 循环的用途.

例 3：利用 for 循环求 1!+2!+3!+…+20!的值. （lt53.m）

```
sum=0;                          %先将所求的值预定义为零
 for i=1:20                     %以 i 为循环变量，循环 20 次
   prd=1;   %每次循环中都设一个临时变量 prd，初始值为 1
   for k=1:i    %以 k 为循环变量，从 1 至 i 完成子循环
            prd=prd*k;      %计算 k 的阶乘
         end                %结束内层循环
   sum=sum+prd;       %把计算出的 k 的阶乘加入变量 sum 中
end                         %结束外层循环
sum                         %求值
sum =
   2.5613e+018
```

循环的嵌套类似于时钟的运行，外层循环变量 i 是时针，内层循环变量 k 是分针，当分针 k 完成一轮循环时，时针 i 只完成了一步循环.

***（3）其他指令**

还有其它一些用于流程控制的命令，它们的名称和功能如下表所示.

pause	使程序运行暂停，按任意键恢复运行，而 pause(n)则是暂停 n 秒.

disp(‘…’)	在屏幕上显示引号中的内容
input	将用户从键盘输入的数值、字符串或表达式赋予指定的变量.

§6 普通物理学中常用的几种计算

本节介绍一些在普通物理中经常用到的计算指令和函数.

（一）解方程

（1）解一元函数方程

解一元函数方程 $f(x)=0$，即利用指令 fzero( )来求一元函数 $y=f(x)$的零点. 语句格式及各项符号说明如下：

x=fzero(fun,x0)

x=fzero(fun,x,options)

x=fzero(fun,x,options,p1,p2,…)

[x,fval]=fzero(…)

x	找到的零点，函数值在此会改变符号，没有零点则返回 NaN.
fval	零点对应的函数值（由于数值解法有一定的精确度，所以往往零点对应的函数值不是恰好为零，而是一个小量）.
fun	用 M 文件或指令建立的单变量函数，其函数值必须是实数.
x0	猜测的初始值或搜寻零点的区间[x0(1),x0(2)]，必须是实数. 如果是区间，要求函数值在 x0(1)与 x0(2)符号相反. 如果是猜测值，则 fzero 将寻找一个包含 x0，同时函数值符号发生改变的区间，如果寻找中遇到了 Inf，NaN 或复数，则寻找终止.

options　　控制算法的优化选项，可选 display 和 TolX 两项，[]是默认值.

p1,p2　　传递给函数的参数.

例 1：求方程 $x\sin x+\cos x^2-3\sin^2 x\cos x=0$ 的根.

首先需要在 M 文件编辑器中建立函数文件：

```
function y=qx(x)
y=x.*sin(x)+cos(x.^2)-3*sin(x).^2.*cos(x);
```

在当前工作路径下保存文件为 qx.m. 在 Matlab 命令窗口中输入“[x,y]=fzero('qx',-3)”并回车，所得结果如下：

```
>> [x,y]=fzero('qx',-3)
x =
    -2.9396
y =
   1.3878e-017
```

值得注意的是，用这种方法寻找函数的零点只能在初始猜测值附近找到最接近的一个零点，如果函数有多个零点，则有遗漏的可能. 所以想要找到函数的所有零点，最好使用函数绘图语句 fplot( )，结合图像判断搜寻范围. 在命令窗口中输入 fplot('qx',[-9 9])，可画出此函数在区间[-9 9]上的图像，如图 10 所示. 可以看到在此区间上的-6、-3、3、6 处均存在零点，可以分别以这几个点为猜测值代入指令，求出较为精确的解.

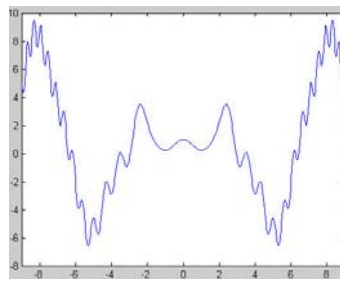


图 10 $y=x\sin x+\cos x^2-3\sin^2 x\cos x$在[-9 9]上的图像

***（2）解线性方程组**

这里作的是矩阵运算，还是不必去学矩阵，你只要模仿例题的做法，就能解线性方程组.

对形式为 ***Ax=b*** 的线性方程组，这里 ***A*** 是 m×m 的矩阵，***b*** 是已知矢量，***x*** 是未知矢量. 用 Matlab 求解时使用矩阵左除的方法，即使用程序式 ***x=A\b***.

例 2：利用 Matlab 求下方程组的数值解. （lt62.m）

$$\begin{cases} 2x+6y-7z=28 \\ 4x-5y+3z=14 \\ 3x+y-2z=9 \end{cases}$$

```
>> A=[2 6 -7;4 -5 3; 3 1 -2];          %建立系数矩阵
>>b=[28;14;9];                         %建立已知矢量
>>x=A\b                                %求解
x =
    -6.6471
   -24.0000
   -26.4706
```

本例程序中的 x 是未知数 *x*、*y*、*z* 所组成的解的矢量，它的三个元素 x(1)、x(2)、x(3)分别代表未知数 *x*、*y*，*z*.

（二）差分、梯度、积分

本小节介绍数值计算差分、梯度和积分的相关指令.

（1）*差分*

给出矢量 x（数组），Matlab 提供了一个近似计算微分的函数 diff，它计算数组中元素间的差分. diff 的用法为：

diff(x)　其值为[x(2)-x(1);x(3)-x(2);…;x(n)-x(n-1)]

对函数 *y*=*f*(*x*)，计算导数 d*y*/d*x* 则用 diff(y)./diff(x)来进行.

例 3：求 $y=\sin x$ 在[-π,π]上的导数.（lt63.m）

```
x=-pi:0.01*pi:pi;          %在[-π,π]上每隔 0.01π 取一个值
y=sin(x);                  %建立函数关系
dy=diff(y)./diff(x);
                %利用对 y 和 x 的差分计算各点上的导数
plot(x,y);                 %画出原函数曲线
hold on        %图形控制，使绘制新图时不抹掉旧图
x(length(x)-1)=[];         %将 x 的最后一个元素去掉
plot(x,dy, 'r.')
              %绘制 x 和的 y 的函数关系，即导函数曲线
```

最终结果如图 11 所示，蓝线表示 $y=\sin x$，红点为导函数曲线.

当 x 在区间上取值越密集时，用差分的办法求出的导数越精确的. 如果自变量取值间隔太大，用有限差分近似计算微分会导致很差的结果.

由于 diff 计算数组元素间的差分，输出的数组比原数组少了一个元素. 所以，画导函数曲线时，必须舍弃 x 数组中的一个元素.

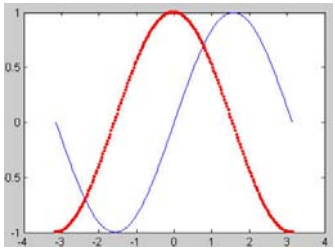


图 11 利用 diff 求导数

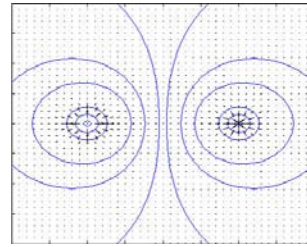


图 12 绘制等势面和电场强度

（2）梯度

Matlab 用 gradient( )函数来求矩阵梯度，其语句格式如下：

[fx,fy]=gradient(f)

[fx,fy]=gradient(f,h)

[fx,fy]=gradient(f,hx,hy)

[fx,fy,fz]=gradient(f)

[fx,fy,fz]=gradient(f,hx,hy,hz)

各项符号含义为：

fx	代表 df/dx，是 x（列）方向的偏导数
fy	代表 df/dy，是 y（行）方向的偏导数
fz	代表 df/dz，是 z（层）方向的偏导数
f	求梯度的矩阵
h	标量，是各个方向的步长. 不指定 h 的时候则取默认值 1
hx,hy,hz	分别表示在 x,y,z 方向上的步长

当 f 是矢量时，gradient 可以代替 diff 函数来求 f 的导数. 例 3 也可以改为：（lt631.m）

```
x=-pi:0.01*pi:pi;
```

```
y=sin(x);
dy=gradient(y,0.01*pi); %用 gradient 函数求导，0.01π 为步长
plot(x,y);
hold on
plot(x,dy,'r.')
```

所得结果与例 3 没有明显区别.

例 4：对§2 中的“等量异号点电荷电场中电势分布”的示例，求出沿 *x,y* 方向电势的梯度，并画出等势面和各点上电场的大小和方向.（lt64.m）

```
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
                        %以 0.1 为步长建立平面数据网格
z=1./sqrt((x-1).^2+y.^2+0.01)... %写出电势表达式
-1./sqrt((x+1).^2+y.^2+0.01);
[px,py]=gradient(z);
                    %求电势在 x,y 方向的梯度即电场强度
contour(x,y,z,[-12,-8,-5,-3,-1,...      %画出等势线
        -0.5,-0.1,0.1,0.5,1,3,5,8,12])
hold on                                  %作图控制
quiver(x,y,px,py,'k')      %画出各点上电场的大小和方向
```

计算结果如图 12 所示.

本例中为了减少计算量，增加精确度，与先前的示例相比，计算范围由原先的-5<x<5,-4<y<4 改为-2<x<2,-2<y<2；步长由 0.5 改为 0.1，电荷位置也改在[-1,0]和[1,0]处.

contour 是绘制等高线的函数，以 z 为对象，则绘出的就是等势线.

quiver 是绘制点[x,y]处的矢量[px,py]，即画出各点电场的大小和方向.

（3）quad, quad8 函数积分

当已知函数的解析形式时，用此命令可以求得满意的积分结果. 其中 quad 基于自适应辛普生算法，属低阶方法；quad8 基于自适应柯特法，属高阶方法，精度较高. 其语句格式为：（quad 或 quad8）

```
q=quad('fun',a,b)
q=quad('fun',a,b,tol)
q=quad('fun',a,b,tol,trace)
    q=quad('fun',a,b,tol,trace,p1,p2,…)
```

各项符号的含义为：

fun	积分函数，需要写在一个函数文件中
a,b	积分区间
tol	积分的精度，默认值为 0.0001
trace	取 0 表示不用图形显示积分过程，非 0 表示用图形显示积分过程
p1,p2,…	是向函数传递的参数

可以看出，这两个命令的用法类似于前面介绍过的 fzero 函数.

例 5：质点匀变速直线运动，满足$v=v_0+at$， $v_0=0.1$，$a=1$，求在 10s内质点走过的路程.

首先在工作路径下建立函数文件 example.m：

```
function y=example(x)                    %函数名为 example
y=0.1+x;                                 %v 与 t 的关系表达式
```

然后在命令窗口中输入：

```
>> sum=quad('example',0,10)
                          %调用 quad 函数求[0,10]上的积分
sum =
    51.0000
```

（三）解常微分方程

Matlab 解常微分方程组的能力很强而且很方便，对于我们在普通物理学中遇到的大多数动力学方程都可以用命令 ode45 求解.

Matlab 只能解一阶的常微分方程组，高阶的常微分方程需要转化成一阶方程组才能求解. 对于二阶常微分方程 $F(\ddot{x},\dot{x},x,t)=0$,首先需要化成显式形式 $\ddot{x}=f(\dot{x},x,t)$，然后令 $y(1)=x$， $y(2)=\dot{x}$，则二阶常微分方程化为两个一阶常微分方程组成的方程组，从而使问题得到解决.

$$\frac{\mathrm{d}y(1)}{\mathrm{d}t} = y(2)$$
$$\frac{\mathrm{d}y(2)}{\mathrm{d}t} = f(y(1), y(2), t)$$

例 6：质点在万有引力作用下的运动. 以万有引力的固定不动的施力质点m_0所在位置为坐标原点O, 建立直角坐标系Oxy，质点的运动微分方程为 $m\ddot{\vec{r}} = -\frac{Gmm_0}{r^3}\vec{r}$，分量方程为：

$$\ddot{x} = -\frac{Gm_0}{x^2+y^2}\frac{x}{(x^2+y^2)^{1/2}}, \quad \ddot{y} = -\frac{Gm_0}{x^2+y^2}\frac{y}{(x^2+y^2)^{1/2}}$$

这两个方程都是二阶常微分方程，定义解矢量为 $\boldsymbol{y}$，令

$$y(1) = x, \quad y(2) = \dot{x}, \quad y(3) = y, \quad y(4) = \dot{y}$$

可将方程组化为：

$$\frac{\mathrm{d}y(1)}{\mathrm{d}t} = y(2), \quad \frac{\mathrm{d}y(2)}{\mathrm{d}t} = \frac{-Gm_0 \cdot y(1)}{\left(y(1)^2 + y(3)^2\right)^{3/2}}$$

$$\frac{\mathrm{d}y(3)}{\mathrm{d}t} = y(4), \quad \frac{\mathrm{d}y(4)}{\mathrm{d}t} = \frac{-Gm_0 \cdot y(3)}{\left(y(1)^2 + y(3)^2\right)^{3/2}}$$

编写微分方程组函数文件 yxlcfun.m：

```
function ydot=yxlcfun(t,y,flag,p)%函数首行，p为参量Gm0
ydot=[y(2);                          %建立微分方程组
    p*y(1)/sqrt(y(1).^2+y(3).^2).^3;
    y(4);
    p*y(3)/sqrt(y(1).^2+y(3).^2).^3];
```

解微分方程的主程序 yxlc.m：

```
p=-1;                                %取Gm0=1
y0=[-10 0.2 6 0.2;-25 0.5 5 0;-25 0.8 6 0];
                                     %三组不同的初始条件
plot(0,0,'*r')                       %画出 O 点
for i=1:3                    %分别以不同初始条件解 3 次方程
    [t,y]=ode45('yxlcfun',[0:0.1:300],y0(i,:),[ ],p);
```

```
    hold on
    axis([-25 25 -20 20]);          %指定坐标范围
    comet(y(:,1),y(:,3))            %绘出质点运动轨迹(x,y)
end                                 %结束循环
```

解出的结果如图 13 所示.

由上面例子，我们初步了解了 Matlab 解常微分方程的一般过程，首先是建立微分方程函数文件，文件的格式如下：

```
fuction ydot=filename(t,y, p1,p2) %t,y 是积分区间和解矩阵
                                  %p1,p2 是参数
ydot=[关于 t,y 的表达式];          %ydot 表示 dy/dt
```

下面介绍 ode45 命令的用法，ode45 的一般调用格式为：

[T,Y]=ode45(‘fun’,tspan,y0,options,p1,p2,…)

其中含义为：

fun	求解的微分方程函数名
tspan	单调递增（减）的积分区间[t0:tstep:tfinal]
y0	初始条件矢量
options	用 odeset 建立的优化选项，一般都用默认值，为空矢量“[ ]”
p1,p2	传递给 fun 函数的参数
T,Y	T 是输出的时间列矢量，矩阵 Y 的每一个列矢量是解的一个分量

各个项在命令中的位置和顺序不能颠倒，否则程序就会出错.

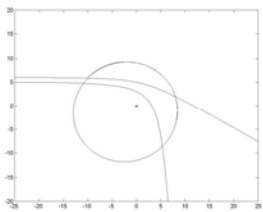


图 13 万有引力场中质点运动轨迹

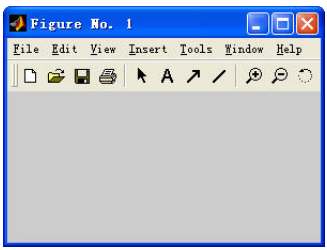


图 14 图形窗口

§7 作图和动画

无论在科研还是数学上，计算的结果都希望能做到可视化，而 Matlab 完善与强大的作图功能，正好满足了这种需求.

本节介绍最基本的一些绘图指令和动画指令.

（一）Matlab 的图形窗口

当使用绘图语句时，Matlab 就自动打开一个图形窗口；如果已经有图形窗口存在，作图命令便会使用已存在的图形窗口. 如果使用命令 figure，就会打开一个新的图形窗口. 每个图形窗口的标题栏都会有一个编号 n，打开第 n 个图形窗口的指令是 figure(n)；在已有图形上继续作图的指令是 hold on；取消这种功能的指令是 hold off. 图形窗口如图 14 所示.

在图中，上端是菜单栏，选择 tools 菜单中 Show Toolbar 命令可以显示工具栏的图标. 各图标的含义分别是新建、打开、保存和打印文件，编辑图形、加注文字、画箭头和画直线，放大、缩小和旋转图形.

在按下工具栏上的“鼠标箭头”后，可以对当前的图形对象的各种属性（颜色、线宽、坐标格式、标志等）加以编辑. File 菜单下的“选项”（Preferences）可以改变图形窗口的各种功能，如数据格式、字体，图形存储的格式等.

（二）二维图形

（1）plot 语句

二维图形绘图命令中最基本的指令就是 plot，它有不同的形式，与输入的量有关. 对于矢量 y，plot(y)产生一个折线图，纵轴为 y 的元素，横轴为 y 的元素指标. 如果输入两个矢量 x,y，则 plot(x,y)产生的是 y 相对于 x 的图形，如前面列举的绘制振动曲线的例子.

一个 plot 命令可以同时画多个图形. Matlab 会自动用不同颜色区分每组数据的图形，颜色是预先设置好的，但可以改动.

指定颜色、线型和数据点标志的 plot 命令格式如下：

plot(x1,y1,'color style marker',x2,y2,'color style marker',…)

例如 plot(x,y,'y:+')画出的是黄色点状线，在每个数据点有加号标志. 如果指定了数据点而不指定线型，则只在各个数据点画出标志符号. 颜色、线型和数据点标志的可选项有：

颜色	青'c'，洋红'm'，黄'y'，黑'k'，红'r'，绿'g'，蓝'b'，白'w'
线型	实线'-'，短划线'--'，点线':'，点划线'-.'
标志	点'.'，加号'+'，圆圈'o'，星号'*'，叉号'×'，四方形's'，钻石形'd'，五角形'p'，六角形'h'，三角形（向上'v'，向下'^'，向左'<'，向右'>'）

例 1：在同一窗口中画多个函数图像. （lt71.m）

```
x=0.01*pi:0.01*pi:pi;                    %建立自变量数组
y1=(2*x-pi).^2-2;                        %各个函数表达式
y2=sqrt(abs(3*x));
y3=2*sin(2*x);
y4=2*cos(2*x);
y5=1./(10*x);
figure;                                  %开启图形窗口
plot(x,y1,x,y2,'g--.',x,y3,'m:*',x,y4,'k.',x,y5,'b-.')
                                         %绘制函数图像
```

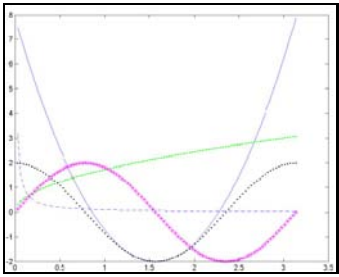
运行结果如图 15 所示.

（2）fplot 语句

当已知函数关系的时候，采用 fplot 语句可以更加快速精确地绘制出指定区间上的函数图像. fplot 绘图的数据点是自适应产生的，即在函数变化小的地方取较少的点，在函数变化剧烈的地方取较多的点. 如

```
fplot('sin(1./x)',[0.01 0.1],1e-3);
```

直接画出了在[0.01 0.1]区间上函数sin(1/x)的曲线，其误差为 10^{-3}，画出图形如图 16 所示.



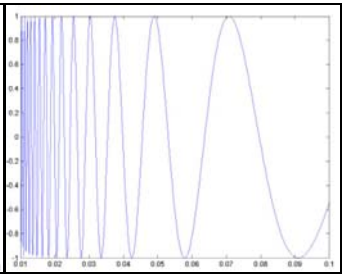


图 15 用 plot 画出的函数图形　　图 16 fplot 绘制的函数图像

读者可以尝试将例 1 的各函数用 fplot 语句绘制出来，fplot 的格式和用法类似于 fzero,fsolve 语句.

***（3）二维特殊图形　Matlab 的帮助系统**

Matlab 有许多画特殊图形的专门指令，调出 Demo 演示窗口（在命令窗口键入 Demo，回车）后选择 Visualization，再在右边的对话框中选择 2D-plots 就可以演示各种二维作图指令，如果选择其它选项，则可以演示其它作图指令.

Matlab 的帮助系统除上述 Demo 演示窗口外，还有以下两种方式：

①Help 窗口：选择菜单栏中“Help→Matlab Help”命令，或者在命令窗口中键入“Helpdesk”，就可调出 Matlab 的帮助窗口，它可以访问本地硬盘和光驱上的帮助信息，也可以通过互联网访问 Internet 上的大量资源.

②Help 指令：对某一指令，如果想详细了解其用法和功能，可以采用在命令窗口中直接键入 Help *command* 的帮助形式. 例如希望了解 sin 函数的用法，就可以在命令窗口中输入“Help sin”，即可得到帮助信息.

下面列举了一些画二维图的指令：

bar	直方图	loglog	双对数坐标曲线
compass	原点出发的复数矢量图	pcolor	假彩色图

contour	在 x-y 面上的等高线图	polar	极坐标图
errobar	误差棒图	quiver	矢量场图
ezplot	符号函数二维曲线	rose	统计频数扇块图
feather	沿 x 轴分布的复数矢量图	semilogx	x 轴对数坐标曲线
fplot	数值函数二维曲线	semilogy	y 轴对数坐标曲线
fill	平面多边形填色	stem	火柴杆图
hist	统计频数直方图	stairs	阶梯图

***（4）坐标轴控制和文字标注**

指令 axis 有很多控制坐标轴的选项（不加控制时 Matlab 会自动地寻找数据的最大值和最小值，选择合适的作图范围和坐标标度）：

axis([xmin xmax ymin ymax])	x 轴和 y 轴的取值范围
axis square	正方形的 x 轴和 y 轴
axis equal	x，y 轴的标度相等
axis auto	把坐标轴伸缩率设为自动模式
axis on	打开坐标轴的标签和刻度线
axis off	取消坐标轴的标签和刻度线
grid off	取消网格线
grid on	恢复网格线
xlabel,ylabel,zlabel	加上 x 轴、y 轴、z 轴的标注
title	在图的上端增加一个标题
text	将文字插入图中

例 2：给例 1 的图形加上坐标轴标注和文本注释.（lt72.m）

```
x=0.01*pi:0.01*pi:pi;
y1=(2*x-pi).^2-2;   y2=sqrt(abs(3*x));      %函数表达式
y3=2*sin(2*x);      y4=2*cos(2*x);          y5=1./(10*x);
figure;                                     %启动图形窗口
```

```
axis([0 pi -3 3]);                          %设定坐标轴范围
hold on                                     %绘图控制
xlabel('-\pi \leq{\it t} \leq \pi','fontsize',15);
                          %x 轴标签，内容，字体大小 15 磅
ylabel('y','fontsize',15);                  %y 轴标签
title('Graph of some function','fontsize',15);
                          %图形标题，内容，字体大小 15 磅
text(1.5,2,'{\it y={x}^{1/2}}','fontsize',10);
                        %文本注释，位置，内容，字体 10 磅
text(1.5,0.5,'{\it y=sin(x)}','fontsize',10);
                       %文本注释，内容可用微软拼音中文
text(1,0,'\it{y=(10x)^{-1}}','fontsize',10);   %文本注释
text(2.2,1.5,'\it y=x^2','fontsize',10);        %文本注释
text(2.7,1.3,'\it y=cos(x)','fontsize',10);     %文本注释
plot(x,y1,x,y2,'g--.',x,y3,'m:*',x,y4,'k.',x,y5,'b-.')  %绘图
```

运行结果如图 17 所示.

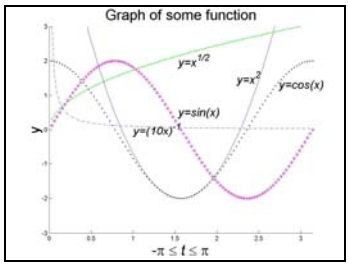


图 17 为图形加标签和注释

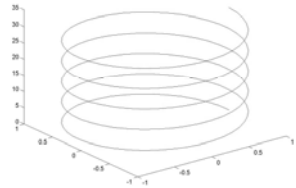


图 18 三维空间中的螺旋线

（三）三维图形

（1）三维空间曲线

三维曲线的作图命令为

plot3(x,y,z,s)

其中 x,y,z 是同维数的矢量或矩阵. 每组 x,y,z 构成一个点的坐标，各点依次相连，形成一条曲线. 如果是矩阵，则它们相应的列构成一条三维曲线的数据点坐标，所以用矩阵可以同时画多条空间曲线. s 是线型、颜色和标志的参数.

例 3: 已知螺旋线的参数方程 $x=\cos t$，$y=\sin t$，$z=t$.

在三维直角坐标系中绘出图形.（lt73.m）

```
t=0:0.05*pi:10*pi;
x=cos(t);
y=sin(t);
z=t;
plot3(x,y,z)
```

运行结果如图 18 所示.

（2）网线和表面图

Matlab 中，曲面是用 *xy* 平面上的各个格点上的 *z* 坐标来定义，相邻点用直线连接. 因此为了描绘二元函数形成的空间曲面，首先要建立平面的数据网格. 即根据 *x,y* 的定义域[x1,x2]，[y1,y2]生成平面上的坐标格点，每个格点对应一组[*x*,*y*]. 生成数据网格的命令是 meshgrid，格式为：

[X,Y]=meshgrid(x,y)

使用 meshgrid 命令，就将两个独立的矢量 x,y 组成了数据网格. 例如：

```
x=-8:0.5:8;y=x;
[X,Y]=meshgrid(x,y);
```

在 *xy* 平面上生成的数据网格如图 19 所示. X、Y 就是数据格点矩阵，需要用它们来构造二元函数，进而绘制空间曲面.



图 19　数据格点

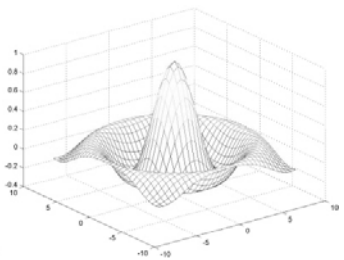


图 20 绘制网格曲面

我们用生成的 X,Y 构造一个二元函数：

$$Z=\frac{\sin R}{R}\quad（其中 R=\sqrt{X^2+Y^2}）$$

建立 M 文件如下：

```
[X,Y]=meshgrid(-8:0.5:8);
            %建立数据网格，x,y 的取值范围都是[-8 8]
R=sqrt(X.^2+Y.^2)+eps;
            %从原点到格点的距离 R，eps 是默认的小量
Z=sin(R)./R;                    %建立函数表达式
mesh(X,Y,Z);                    %绘制三维网格曲面
```

在第二句中加一个小量 eps，是为了避免在原点出现 0/0 的情况. eps 是 Matlab 默认的小量，eps=2.2204e-16. 计算结果如图 20 所示.

（四）动画

Matlab 有多种动画表现方式，包括彗星轨迹图，影片动画等.

（1）彗星轨迹图

彗星轨迹图可以非常容易地绘出质点运动的轨迹，其使用格式为：

comet(y) comet(x,y) comet(x,y,p)	二维平面彗星轨迹，x,y 的含义与 plot 命令中 x,y 含义相同，绘出的轨迹慧长为 p*length(y)，p 的默认值为 0.1
comet3(z) comet3(x,y,z) comet3(x,y,z,p)	三维空间彗星轨迹，x,y,z 的含义与 plot3 命令中的 x,y,z 含义相同，绘出的轨迹慧长为 p*length(y)，p 的默认值为 0.1

例 4：已知质点在平面上同时参与 *x*、*y* 方向的简谐振动

$$x=3\sin(5\pi t+\frac{\pi}{4})，\quad y=2\sin\left(4\pi t+\frac{\pi}{6}\right)$$

绘制出质点在平面上的运动轨迹.（lt74.m）

```
t=0:0.01:20;                    %时间 t 从 0 至 20 秒
```

```
A1=3;A2=2;wx=5*pi;wy=4*pi;        %对振幅、圆频率赋值
phi1=pi/4;phi2=pi/6;              %对初相位赋值
x1=A1*sin(wx*t+phi1);          %x 方向上的简谐振动
y1=A2*sin(wy*t+phi2);          %y 方向上的简谐振动
comet(x1,y1);                     %画彗星轨迹图
```

程序动态地绘制了质点的运动，绘制结束后质点留下的轨迹如图 21 所示.

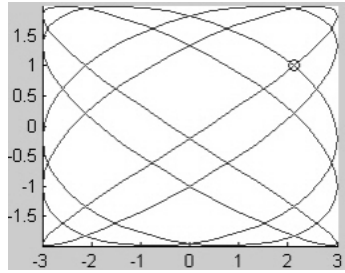


图 21 振动合成的李萨如图形

***（2）影片动画**

制作影片动画和电影制作相类似，即预先将一幅幅图形制作好，放在图形缓存区内，然后连续地播放，播放的速度可快可慢，这种方法需要的内存较大. 制作图形时一般结合循环语句，在循环体内作图，每次循环绘制一幅图形，并用命令 getframe 放到存储矩阵 M 里，当结束循环后使用 Movie 命令将所有图形依次按指定速率播放，形成影片动画.

getframe 的用法为 M(i)=getframe，i 为循环变量

movie 的格式为 movie(M,k,Fps)，表示将 M 所存储的 n 帧画面演示 k 次，每秒 Fps 帧. 默认的 k=1，Fps=12.

例 5：演示行波 z=sin(*kr*+*ωt*)的传播过程.（lt75.m）

```
[x,y]=meshgrid(-8:0.1:8);          %生成数据网格
r=sqrt(x.^2+y.^2);                 %格点距波源的距离 r
for t=1:20                         %时间 t 从 1 至 20 秒
    axis([-8 8 -8 8 -8 8]);        %指定坐标轴范围
    hold on                        %维持当前图形坐标设定
35          z=sin(pi*r-0.2*pi*t);
```

```
                                %距波源 r 处的质点在 t 时刻的振幅
    mesh(x,y,z);                %绘制 t 时刻的波动状态
    f(t)=getframe;              %将画面存储到矩阵 f 中
    cla;        %清除屏幕，便于下一步循环的重新绘图
end                             %结束循环
movie(f,1,12);                  %电影动画播放
```

§8 Matlab 在力学中的应用举例

（一）驻波形成—火柴杆图

本例用 Matlab 来研究机械波的相干叠加与驻波的形成. 考察分别沿 x 轴正向和负向传播的两列相干横波，它们的方程为：

$$y_1 = A_1 \sin k(x - vt)$$
$$y_2 = A_2 \sin k(x + vt)$$

设时间从t=0 开始到t=10 结束，考察区间为[0,4]，令k=π，则λ=2π/k=2，在考察区间上恰好能观察到两个完整波形. 令v=1，A_1=-0.4，A_2=0.4，则方程为：

$$y_1 = -0.4\sin \pi(x - t)$$
$$y_2 = 0.4\sin \pi(x + t)$$

现在就可以编程演示机械波的叠加和驻波的形成了，程序如下：（lt81.m）

```
    %横波驻波的形成
    t=0:0.1:10;x=0:1/15:4;          %时间数组和位置坐标数组
    for i=1:100                         %设置循环
    %由右向左传播的行波
        x1=x(x>=(4-t(i)));
%挑出坐标位置大于 4-t(i)的质元，将其位置坐标赋给新的变量 x1
        y11=0.4*sin(pi*(x1+t(i)));
        %这些位置上的质元在 t(i)时刻的位移，赋给变量 y11
        x2=x(x<(4-t(i)));
%挑出坐标位置小于 4-t(i)的质元，将其位置坐标赋给新的变量 x2
        y12=x2-x2;
%这些位置上的质元在 t(i)时刻的位移，赋给变量 y12
            y1=[y12 y11];
```

```
        %将所有质元的位移组合到同一个列矢量 y1 中
        %由左向右传播的行波
        x3=x(x<=t(i));
        y21=-0.4*sin(pi*(x3-t(i)));
        x4=x(x>t(i));
        y22=x4-x4;
        y2=[y21 y22];
    y3=y1+y2; %质元同时参加两个振动，实现了行波的合成
    y=[y1;y2;y3]; %将 t(i)时刻向左、向右以及合成的波的质
                  %元位移存放在矢量 y 中，以便作图
                  %分区作图
    for j=1:3
        subplot(3,1,j)       %分区作图
        stem(x,y(j,:),'b:.'); %火柴杆图
        axis([0,4,-1,1]);    %指定坐标轴范围
        grid on              %开启网格线
     end
    pause(0);                %暂停的技巧，否则屏幕由于刷新
                             %过快而导致不显示任何图象
    end
```

程序在 10s 内每隔 0.1s 取一个时刻，在考察区间上每隔 1/15 取一个质元.

在 t<4s 之前，波前还未走过整个区间，只有部分质元参与了振动，其余质元仍处于静止状态，位移为 0，因此需要根据当前时刻 t(i)找出已经运动和尚未运动的质元，分别对其位移 y11、y12 赋值. 注意 y12=x2-x2 和 y12=0 两种赋值是不等效的，前者是长度与 x2 相同的矢量，后者只是一个数值（标量）.

为了有明显的对比，程序采用了分区作图，分别绘制了当前时刻的两列行波波形以及叠加的波形，这里用到了 subplot 命令，格式为 subplot(m,n,p)，将图形窗口划分 *m* 行 *n* 列共 *m*×*n* 个子绘图区，*p* 是当前使用绘图区的序号.

绘制波形图用了 stem 语句，格式为 stem(x,y,'color style marker')，与 plot 语句的格式用法相类似.

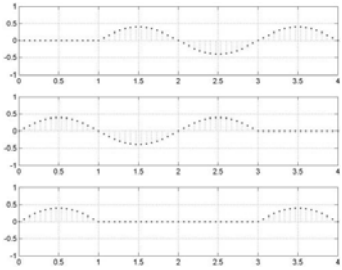


图 22　某时刻行波及驻波的波形

整个程序放在一个大的循环结构中，循环变量是时间 t，每一次循环对应一个时刻，绘出当前时刻的波形；当进入下一次循环时屏幕自动刷新，绘制下一时刻的波形，由此实现了动画效果. 图 22 是绘出的某时刻的波形图.

（二）受迫振动—相图

受迫振动的运动微分方程为：

$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = f\cos\omega t$$

其中β是阻尼系数，ω_0是振动系统的固有频率，ω是策动力的圆频率，f是策动力的幅值. 令$y(1)=x$，$y(2)=\mathrm{d}x/\mathrm{d}t$，则运动微分方程化为

$$\dot{y}(1) = y(2)$$

$$\dot{y}(2) = -2\beta y(2) - \omega_0^2 y(1) + f\cos\omega t$$

改变 ω 值，可研究共振现象. 令 f=0，则为阻尼振动. 用 Matlab 解此微分方程，研究位移与时间的关系，以及位移与速率的关系. 编写微分方程函数文件，保存为 spzdfun.m.

```
function ydot=spzdfun(t,y,flag,b,w0,f0,w)
                        %b,w0,f0 是由主程序传递来的参量
ydot=[y(2);
      -2*b*y(2)-y(1)*w0^2+f0*cos(w*t)];
```

再编写主程序 spzd.m，程序如下所示：

```
b=0.1;w0=2.68;f0=0;w=6;                %为β、ω0、f、ω 赋值
[t,y]=ode45('spzdfun',...               %解微分方程
    [0:0.1:40],[1,0],[ ],b,w0,f0,w);    %初始条件x0=1,v0=0
plot(t,y(:,1),'b')                      %绘出位移－时间曲线
figure(2);                              %新开绘图窗口
plot(y(:,1),y(:,2),'b')                 %绘制位移－速度相图
```

运行结果如图 23 和 24 所示.

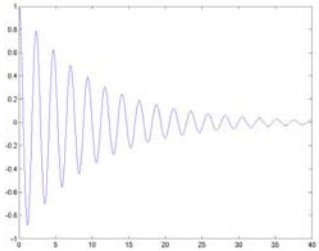


图 23 $x-t$ 图

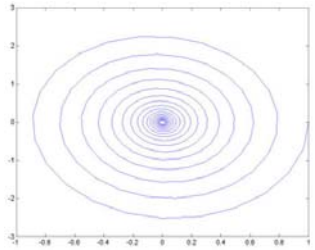


图 24 $x-v$ 相图

相平面法是一种直观的几何方法，它适用于系统的一维运动. 以位置 $x=X$ 、速度 $\dot{x}=Y$ 为坐标建立坐标系 OXY，通常也称此坐标平面为相平面. 相平面中的一个取值点代表系统该时刻的运动状态，称为相点. 相点连续变化形成的轨道，则描述了系统运动状态的发展变化过程，称为相轨道，这种图形也称相图.

利用相图对非线性系统的行为进行研究是一种重要的方法.

§9 Matlab 在电磁学中的应用举例

（一）等量同号点电荷的电场线的绘制

首先建立电场线的微分方程（二维情况）. 因为电场中任一点的电场方向都沿该点电场线的切线方向，所以满足：

$$\frac{\mathrm{d}y}{\mathrm{d}x}=\frac{E_y}{E_x}$$

引入参变量 t 得到：

$$\frac{\mathrm{d}x}{E_x}=\frac{\mathrm{d}y}{E_y}=\mathrm{d}t$$

设二点电荷位于(-2,0)和(2,0)，二点电荷“电量”为q_1和q_2（均等于 10），由库伦定律和电场的叠加原理，得出下列微分方程：

$$\dot{x}=\frac{\mathrm{d}x}{\mathrm{d}t}=E_x=\frac{q_1(x+2)}{[(x+2)^2+y^2]^{3/2}}+\frac{q_2(x-2)}{[(x-2)^2+y^2]^{3/2}}$$

$$\dot{y}=\frac{\mathrm{d}y}{\mathrm{d}t}=E_y=\frac{q_1 y}{[(x+2)^2+y^2]^{3/2}}+\frac{q_2 y}{[(x-2)^2+y^2]^{3/2}}$$

解此方程就可以绘制出电场线.

下面是写微分方程的函数文件：

```
function ydot=dcx1fun(t,y,flag,p1,p2)
                                %p1,p2 是参量，表示电量
ydot=[p1*(y(1)+2)/(sqrt((y(1)+2).^2+y(2).^2).^3)+...
        p2*(y(1)-2)/(sqrt((y(1)-2).^2+y(2).^2).^3);
                                %dx/dt=Ex
      p1*y(2)/(sqrt((y(1)+2).^2+y(2).^2).^3)+...
        p2*y(2)/(sqrt((y(1)-2).^2+y(2).^2).^3)];
                                %dy/dt=Ey
```

这里的 y 是微分方程的解矢量，它包含两个分量，y(1)表示 x，y(2)表示 y，解出 y 后就得到了 x 与 y 的关系，即可依此绘制出电场线.

编写好函数文件后，命名为 dcx1fun.m 存在当前路径下，然后开始编写解微分方程的主程序 dcx1.m：

```
p1=10; p2=10;                   %点电荷所带电量
axis([-5,5,-5,5]);    %设定坐标轴范围 -5≤x≤5,-5≤y≤5
hold on                         %图形控制,不可擦除模式
plot(2,0,'*r');  plot(-2,0,'*r')        %绘制两源电荷
a=(pi/24):pi/12:(2*pi-pi/24);
                        %圆周上电场线起点所对应的角度
b=0.1*cos(a);c=0.1*sin(a);
                        %电场线起点所对应的相对坐标
```

```
b1=-2+b;b2=2+b;   %把起点圆周的圆心放置在源电荷处
b0=[b1 b2];   c0=[c c];   %初始条件，所有电场线的起点
                          %的横、纵坐标构成了矢量 b0 和 c0
for i=1:48                %循环求解 48 次微分方程
  [t,y]=ode45('dcx1fun',[0:0.05:40],[b0(i),c0(i)],[ ],p1,p2);
%调用 ode45 求解，对应一个初条件（起点），求解出一条电场线
  plot(y(:,1),y(:,2),'b')          %绘制出此条电场线
end                       %结束循环，共绘制出 48 条电场线
```

在确定初始条件时，因为源点处是奇点，这点上微分方程的分母为 0，所以电场线不能从源点处绘制，而应当从它附近的邻域圆上绘制. 我们将电场线的起点定在以源点为圆心，0.1 为半径的圆周上.

在程序中就是通过从圆周上取了 24 个不同的角度(从 π/24 到 2π-π/24，每隔 π/12 取一个角度)，然后算出每个角度上的起点的横、纵坐标值；[b1,c]和[b2,c]分别是以两个源点电荷为圆心，0.1 为半径的邻域圆周上的起点位置. b0=[b1 b2]，c0=[c c]是合并矢量，将两个源点处的初始条件组成的矢量放在一起处理.

最后所得结果如图 25 左图所示,将左端源电荷附近放大可以看到这些电场线的起点都在源电荷的邻域圆上，如图 25 右图所示. 从这个圆周上发出的电场线共 24 条，另一端也是相同的情况.

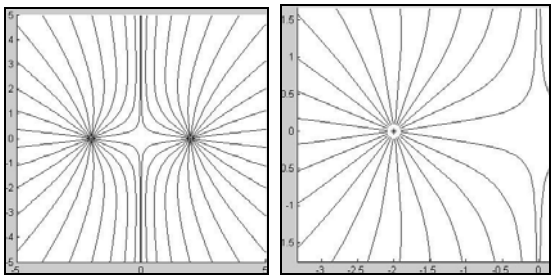


图 25　等量同号点电荷的电场线

（二）带电粒子在均匀电磁场中的运动

设带电粒子质量为 m，带电量为 q，电场强度 $\boldsymbol{E}$ 沿 y 方向，磁感应强度 $\boldsymbol{B}$ 沿 z 方向. 则带电粒子在均匀电磁场中的运动微分方程为

$$\ddot{x} = \frac{qB}{m} v_y = \frac{qB}{m} \dot{y}$$

$$\ddot{y} = \frac{q}{m} E - \frac{qB}{m} v_x = \frac{q}{m} E - \frac{qB}{m} \dot{x}$$

$$\ddot{z} = 0$$

令 $y(1) = x$，$y(2) = \dot{x}$，$y(3) = y$，$y(4) = \dot{y}$，$y(5) = z$，

$y(6) = \dot{z}$ 则上面微分方程可化作：

$$\frac{\mathrm{d}y(1)}{\mathrm{d}t} = y(2)\,, \quad \frac{\mathrm{d}y(2)}{\mathrm{d}t} = \frac{qB}{m} y(4)$$

$$\frac{\mathrm{d}y(3)}{\mathrm{d}t} = y(4)\,, \quad \frac{\mathrm{d}y(4)}{\mathrm{d}t} = \frac{q}{m} E - \frac{qB}{m} y(2)$$

$$\frac{\mathrm{d}y(5)}{\mathrm{d}t} = y(6)\,, \quad \frac{\mathrm{d}y(6)}{\mathrm{d}t} = 0$$

选择 E 和 B 为参量，就可以分别研究 $E \neq 0$，$B = 0$ 和 $E = 0$，$B \neq 0$ 等情况．下面编写微分方程函数文件 ddlzfun.m：

```
function ydot=ddlzfun(t,y,flag,q,m,B,E)    %q,m,B,E 为参量
ydot=[   y(2);
         q*B*y(4)/m;
         y(4);
         q*E/m-q*B*y(2)/m;
         y(6);
         0];
```

再编写解微分方程的主程序 ddlz.m：

```
q=1.6e-2;     m=0.02;      %为粒子的带电量和质量赋值
B=2;    E=1;      %为电磁场的磁感强度和电场强度赋值
[t,y]=ode23('ddlzfun',[0:0.1:20],...
   [0,0.01,0,6,0,0.01],[ ],q,m,B,E);
                %用 ode23 解微分方程组，时间设为 20s
                   %指定初始条件，传递相关参数
plot3(y(:,1),y(:,3),y(:,5),'linewidth',2);
          %绘出三维空间内粒子运动的轨迹，线宽 2 磅
grid on                                    %开启坐标网格线
xlabel('x'); ylabel('y'); zlabel('z');      %为坐标轴命名
```

运行结果如图26所示. 研究时可以采用不同的初始条件和不同的参量观察不同的现象. 例如令 E=0,B=2，所得结果如图 27 所示.

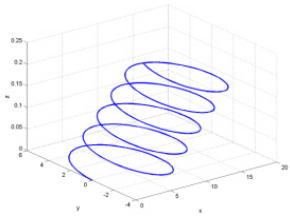


图 26　现有参数运行结果

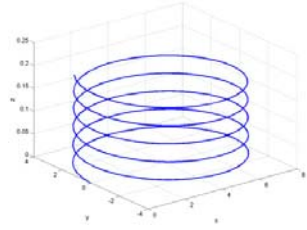


图 27 修改参数运行结果

§10 Matlab 在光学中的应用举例

（一）理想情况下的双缝干涉　色图设置

理想情况下的双缝干涉（不考虑单缝衍射的调制作用）如图 28 所示.

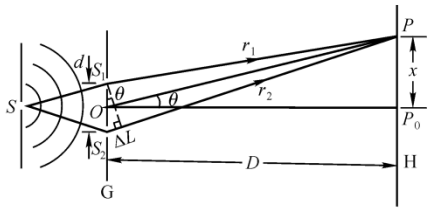


图 28 双缝干涉

满足相干条件的光波叠加后，对于观察屏上某一确定点，即r_1、r_2确定后，叠加后的光强为

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}\cdot\cos\frac{2\pi}{\lambda}(r_2 - r_1),\ I_1 = \frac{E_0^2}{r_1},\ I_2 = \frac{E_0^2}{r_2}$$

式中λ为光波的波长，E_0为常量. 我们用Matlab编写程序，在$z=z_0=D$平面内（即屏上）观察相干光叠加后光强的分布.

用国际制单位，令E_0=1，λ=6e-9，z_0=1，双缝位置为

[-0.0001,0]和[0.0001,0]，编写程序如下：（sf.m）

%双缝干涉——由光程差计算光强

```
x=-0.01:0.00003:0.01;
y=-0.01:0.0003:0.01;      %设置屏上观察范围为 1×1cm^2
z0=1;lamta=600e-9; %设置观察屏与光源的距离及光波长
a=2*pi/lamta;                    %参数 2π/λ
[X,Y]=meshgrid(x,y);             %建立数据网格
R1=sqrt((X+0.0001).^2+z0^2);  %光源 1 到观察点的距离
R2=sqrt((X-0.0001).^2+z0^2);  %光源 2 到观察点的距离
I1=1./R1;                        %光源 1 在观察点的光强
I2=1./R2;                        %光源 2 在观察点的光强
I=I1+I2+2.*sqrt(I1.*I2).*cos(a.*(R2-R1));%叠加后光强
figure;                                %开辟图形窗口
hold on
colormap(gray);                        %选取颜色控制
view(0,90);                            %在 x-y 平面观察
mesh(X,Y,I);                  %画出光强在观察屏上的分布
```

运行结果如图 29 所示.

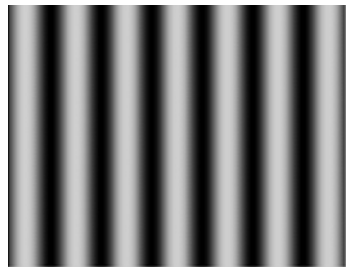


图 29　理想双缝干涉的模拟图样

由于三维网格表面图中的网格色调即代表该处的函数值，数值越大色彩越明亮，越偏暖色. 所以只要能计算出观察点处的光强，就可以用表面图生成对对应的色图.

这里使用了色图设置命令 colormap.

色图是一个 m×3 的矩阵，其中行代表颜色，第 1 列给出红色的数量，第 2 列给出绿色的数量，第 3 列给出蓝色的数量，因此，该色图能给出 m 种颜色.

比如 m=1 时，[1 0 0]为红色，[0 0 1]为蓝色，[1 1 0] 为黄色，[0.5 0.5 0.5] 为灰色，[1 0.62 0.4] 为铜色等. (利用它可以做色彩合成的模拟试验.)

colormap (Cm) 设定当前颜色表为 Cm. 矩阵 Cm 可以是 Matlab 自身的色表，或用户自定义的色表. colormap 则返回 m×3 矩阵的当前图形色表.

为了使用预先定义的色图，可以使用命令 colormap(winter)：

autumn	红色和黄色图
spring	洋红和黄色图
summer	绿色和黄色图
winter	蓝色和绿色图
gray	线性灰色刻度
contrast	提高对比度的灰色图
hsv	从红到蓝再到红的深度颜色
hot	由黑到红到黄和白的混合的暖色
cool	青色和品红色的冷色
bone	带蓝色的灰色刻度
copper	线性铜色刻度
pink	粉红色的变化
flag	红色、白色、蓝色和黑色循环重复
prism	循环重复红色、橘黄色、黄色、绿色、蓝色和紫罗兰色
jet	交替的从红到蓝的色彩模式颜色表
white	全白的色图

本例中使用了线性灰度色图，只保留了灰度成分，舍弃了彩色成分，使模拟图样更接近真实情况. 不过必须清楚这个图样只是根据光强的分布在平面上得出的色彩映射，色图是人为选择的.

另外用了视角控制的 view 命令，格式为 view(v,h). 该命令用来设置观察点角度. 标量 v 是方位角，为 *xy* 平面上逆时针的角度. 平面上方的仰角由标量 h 设定，h 和 v 的单位都是“度”.

view(0,90)可以使观察者正对 *xy* 平面，则看到的就是光强在该平面上的投影，即光强的色彩表达，从而模拟出干涉图样.

（二）光栅

光栅衍射的公式为：

$$I(\sin\theta)=\frac{I_0\sin^2\left(\frac{\pi a}{\lambda}\sin\theta\right)}{\left(\frac{\pi a}{\lambda}\sin\theta\right)^2}\times\frac{\sin^2\left(N\frac{\pi d}{\lambda}\sin\theta\right)}{\sin^2\left(\frac{\pi d}{\lambda}\sin\theta\right)}$$

其中 $\sin\theta=\frac{x}{\sqrt{x^2+D^2}}$，$x$为观察屏上点坐标，D为观察屏到光栅的垂直距离，$a$为光栅各缝的宽度，$d$为光栅常数，$d=a+b$，$b$是相邻狭缝的间距；$N$是光栅上狭缝的数目；$\lambda$是光波波长；$I_0$为常量.

令 $\alpha=\frac{\pi a}{\lambda}\sin\theta,\beta=\frac{\pi d}{\lambda}\sin\theta$，$I_0$=1，光栅衍射公式可化为

$$I=\frac{\sin^2\alpha}{\alpha^2}\cdot\frac{\sin^2 N\beta}{\sin^2\beta}$$

采取国际单位制，并赋给公式中的各个物理量合理的数值，仿照双缝干涉的示例，编写 Matlab 程序如下：（gs.m）

```
a=0.01e-3;b=0.04e-3;d=a+b;D=1; N=8;
                          %为光栅各个参数赋值
lamda=600e-9;                  %光的波长
x=-0.2:0.0003:0.2;y=-1:0.03:1;  %屏上观察平面的大小
[X,Y]=meshgrid(x,y);           %组成数据网格
sinsita=X./(D^2+X.^2).^0.5;    %计算 sin(θ)
arfa=pi*a*sinsita./lamda;      %计算 α
beita=pi*d*sinsita./lamda;     %计算 β
    I=(sin(arfa).^2./(arfa.^2)).*...
```

```
    (sin(N.*beita).^2./(sin(beita).^2));          %相对光强分布
figure;                                          %开辟图形窗口
plot(x,I);                          %画光强与观察点位置关系图
figure;view(0,90);          %新开图形窗口并在 xy 平面内观察
hold on
colormap(gray);                                  %选择灰度色图
mesh(X,Y,I);                                     %绘制衍射图样
```

运行结果如图 30 和图 31 所示.

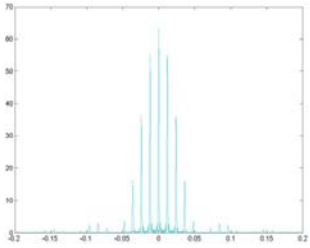


图 30 光强与位置的关系

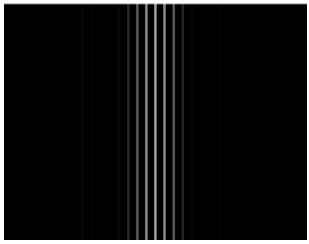


图 31 光栅衍射的模拟图样