第 1 章　MATLAB 基础知识

MATLAB 是矩阵实验室（Matrix Laboratory）的简称，是美国 MathWorks 公司出品的商业数学软件。MATLAB 是用于科学计算的一种高性能语言，其应用范围非常广泛，包括数学运算、算法开发、数据采集与挖掘、数据建模与模拟分析、数据研究与可视化，以及应用程序开发等。

1.1　启动与退出

MATLAB 的启动与其他软件的启动方法相同，此处不再赘述。

其退出有不同之处，常用方法是直接在命令窗口提示符（>>）后输入 exit 或 quit 命令，相当于从菜单 File 中选择 Exit MATLAB 选项。

例如（其中↙表示回车）：

```
>> exit↙
```

如果在 Preferences 对话框（File→Preferences）中选中“Confirm before exiting MATLAB”（见图 1-1），则在退出之前会出现如图 1-2 所示的确认对话框。

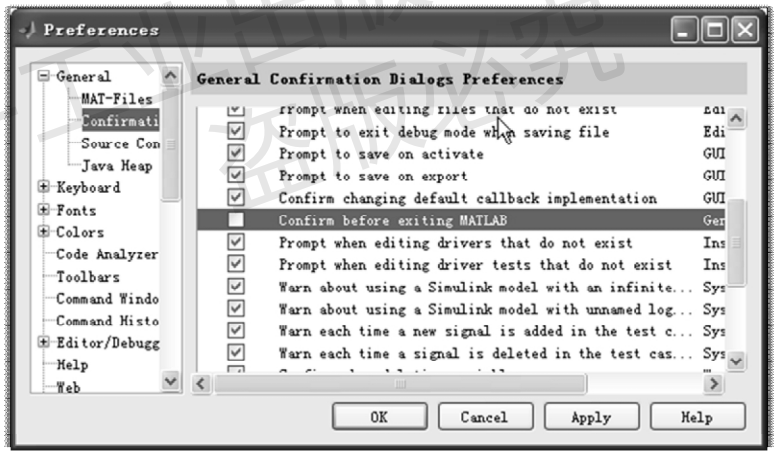

图 1-1　偏好设置对话框

注意：可以在偏好设置窗口中改变字体或设置其他选项。

图 1-2　退出确认对话框

1.2 MATLAB 界面组成

MATLAB 界面组成如图 1-3 所示，主要包含四类窗口：命令窗口（command window）、工作区窗口（workspace）、命令历史窗口（command history）和当前目录窗口（current folder）。

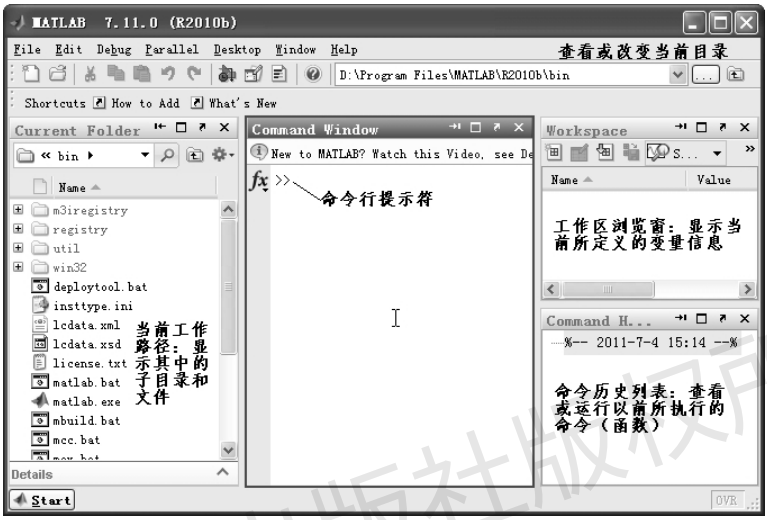

图 1-3 MATLAB 界面组成

每个窗口的右上角有四个图标 ⇥ □ ↗ ✕，其中 ⇥ 表示可以将窗口停靠在左侧或右侧边框，当鼠标移动至窗口标签上时，窗口会自动显示；点击 ⧉ 则可以恢复其先前状态；点击 □ 则可以将某个窗口在 MATLAB 主窗口内最大化，点击 ↗ 则可以将其移出 MATLAB 主窗口，作为一般窗口来显示，点击 ↘ 则将移出的窗口移回 MATLAB 主窗口，如果关闭了某个窗口，则可以通过菜单 Window→Command Window 将其恢复，或者选择 Desktop→Desktop Layout→Default 恢复所有的默认排列。

1.3 命令的执行

1．直接在命令窗口中输入语句

我们可以在命令窗口中直接对变量进行赋值，运行命令或函数，所生成的结果或辅助信息也会在命令窗口中显示。如果命令语句有返回值，在语句末尾添加分号（;）可以屏蔽掉结果的输出，对于没有返回值的命令或函数，则没有分别。

例如，生成一变量 a，并将其赋值为 10

```
>> a=10↙

a =
10
```

生成魔方阵（矩阵的每行之和以及每列之和相等），

```
>> magic(5)↙

ans =
   17    24     1     8    15
```

```
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

可以在一行中输入多条语句，例如，生成两个变量 a 和 b，并赋值为 10 和 20，然后将 a 和 b 相加，并显示结果，

```
>> a=10;b=20;a+b↵
ans =
      30
```

对于有返回值的语句，如果没有明确将返回值赋予某个变量，则默认将其赋予 ans。

如果输入的是程序控制语句（如 if … end 或 for … end 等），则回车后并不显示提示符，而要等到完成对应的语句组并回车后，才会出现命令提示符。

例如，计算 1+2+ … +10 的和

```
>> a=0;for i=1:10 a=a+i;↵
end↵
>> a↵   %显示变量 a 的内容
a =
    55
```

●注意：控制语句末尾不需要加分号

下面的命令清除命令窗口中所有的内容

```
>> clc↵
```

●注意：如果需要重新执行最近所使用的命令或语句，可以通过上下按键来显示；也可以先输入前几个字符，再通过上下键快速定位，如果要取消当前在命令窗口中输入的内容，可以按 Esc 键。

另外 Tab 键在命令窗口中有特殊作用，当输入某部分内容后，再按 Tab 键，MATLAB 会以列表的形式显示可供选择的内容，这样可以节省输入的字符，如输入：

```
>> edit  anal <Tab>
```

则显示如图 1-4 所示的列表，通过鼠标或上下键进行选择即可。

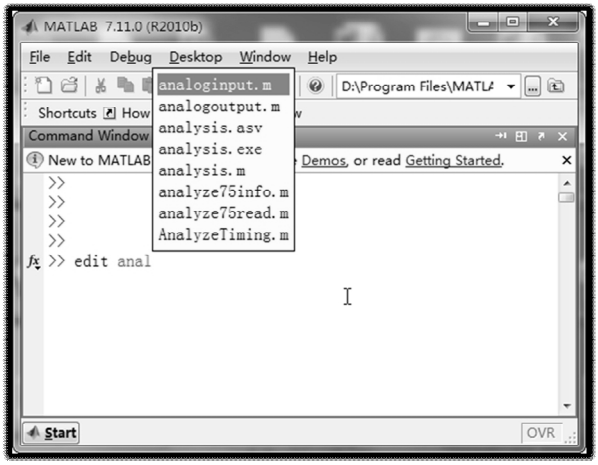

图 1-4　Tab 列表选择

2．粘贴多行代码语句

你可以把从其他地方（网站、程序文件等）复制的 MATLAB 语句代码粘贴在命令窗口中一次性执行其中的内容。

例如，在记事本中输入以下内容

```
for i=1:10
    for j=1:i
        fprintf('%d\t',i);
    end
    fprintf('\n');
end
```

然后将其粘贴至 MATLAB 命令窗口中，会输出以下数字三角形，

```
>> for i=1:10
    for j=1:i
        fprintf('%d\t',i);
    end
    fprintf('\n');
end
1
2    2
3    3    3
4    4    4    4
5    5    5    5    5
6    6    6    6    6    6
7    7    7    7    7    7    7
8    8    8    8    8    8    8    8
9    9    9    9    9    9    9    9    9
10   10   10   10   10   10   10   10   10   10
```

3．F9 执行

如果要执行 MATLAB 环境下的示例代码（此处 MATLAB 环境是指命令历史列表、帮助文件、命令窗口中的内容、MATLAB 程序文件等），可以将代码片段选中，然后从右键弹出的快捷菜单中选择 Evaluate Selection 选项或直接按功能键 F9 执行。

4．建立快捷键

选择一条或多条语句，然后将其拖至快捷键条上，如果选中的是命令历史窗口的内容，还可以从快捷菜单中选择 Create Shortcut 命令，通过窗口（见图 1-5）来设置创建的快捷键的细节，其中 Label 指定标签名，Icon 可以设置标签图标，设置完毕后点击 Save 按钮后，在快捷键条上出现清除命令窗口（见图 1-6）。

5．建立程序文件（M 文件）

将 MATLAB 函数或命令语句有机地组合在一起，保存在 M 文件中（其扩展名为.m），然后运行 M 文件就可以执行其中的命令代码（参见 4.4 节）。

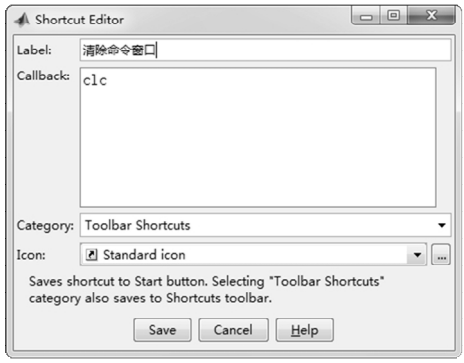

图 1-5　快捷键编辑窗口

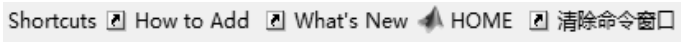

图 1-6　快捷键工具条

1.4　获取帮助信息

MATLAB 中的帮助信息非常重要，而且也提供了强大的帮助资料来方便大家查询各类函数的功能。

最常用的方法是在命令窗口中，输入 help command，其中 command 为所要查询的有关命令或函数，有关某主题的帮助信息会显示在命令窗口中。例如，查询有关命令 clc 的帮助信息，则输入

```
>> help clc↵
 CLC    Clear command window.
    CLC clears the command window and homes the cursor.
    See also home.
    Reference page in Help browser
       doc clc
```

帮助信息给出命令 CLC 的功能和用法，通过 See also 后面的函数链接可以查询相关函数的帮助信息，在 doc 之后的链接可以打开帮助浏览器窗口（见图 1-7），也可以直接通过 doc 命令打开某主题的帮助窗口，例如

```
>>doc sin↵
```

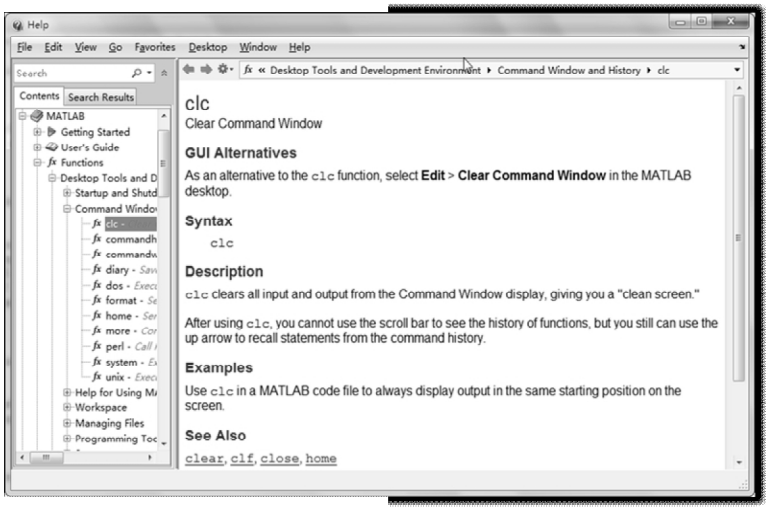

图 1-7　帮助窗口

当在命令窗口某函数名处按功能键 F1 时，会弹出如图 1-8 所示的快捷帮助窗口。

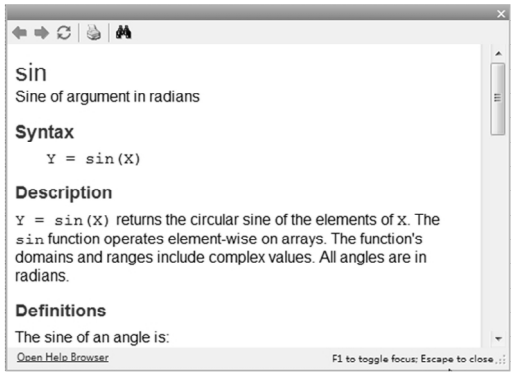

图 1-8　快捷帮助窗口

按 Shift+F1 组合键或从菜单选择 Help→Function Browser，则会弹出函数查找快捷窗口，输入要查询的函数，则显示所有与之匹配的选项（见图 1-9），将鼠标移至某个条目上会显示相关的帮助信息（见图 1-10）。

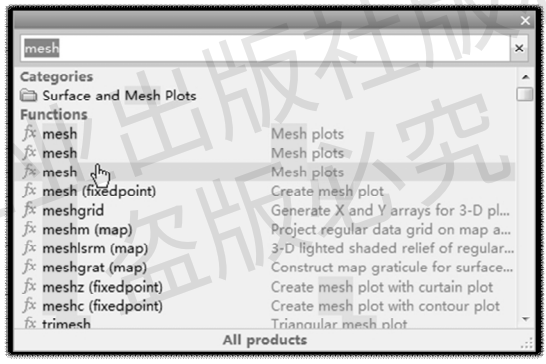

图 1-9　函数查找窗口

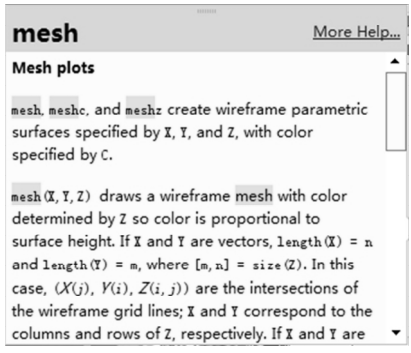

图 1-10　帮助信息窗口

第2章 数 据 类 型

MATLAB 支持多种数据类型，并且对不同数据类型的处理提供了丰富的函数，而且还可以自定义数据类型，合理地使用不同类型的数据，一方面可以节省内存的占用空间，另一方面也可以使程序编制事半功倍。

2.1 数 值 型

数值型包括整数、浮点数和复数。

1．整数

整数又因能够容纳的数值大小的不同分为不同位数的有无符号的整数。由于计算机采用二进制编码，因此 1 个字节可以容纳 2^8 个数，2 个字节可以容纳 2^{16} 个数，4 个字节可以容纳 2^{32} 个数等。不同类型的整数表示的范围见表 2-1。如果表示分值介于 0～150 之间的功课成绩，则可以使用无符号 8 位整数；如果要表示一个国家的人口数，则可以使用无符号 32 位整数（大约可以表示 42.9 亿），整数表示范围如表 2-1 所示。

表 2-1 整数的表示范围

数 据 类 型	值 范 围	转 换 函 数
有符号 8 位整数	-2^7～2^7-1	int8
有符号 16 位整数	-2^{15}～$2^{15}-1$	int16
有符号 32 位整数	-2^{31}～$2^{31}-1$	int32
有符号 64 位整数	-2^{63}～$2^{63}-1$	int64
无符号 8 位整数	0～2^8-1	uint8
无符号 16 位整数	0～$2^{16}-1$	uint16
无符号 32 位整数	0～$2^{32}-1$	uint32
无符号 64 位整数	0～$2^{64}-1$	uint64

由于 MATLAB 默认的数据类型为双精度浮点数，使用表 2-1 中的转换函数可以创建相应的整数类型，例如

```
>> a=uint8(93);↵
>> b=int16(327);↵
>> c=int32([76 89]);↵
>> d=1;↵
>> whos↵
```

```
  Name      Size            Bytes  Class    Attributes
    a       1x1                 1  uint8
    b       1x1                 2  int16
    c       1x2                 8  int32
    d       1x1                 8  double
```

通过 whos 命令可列出当前工作区中所定义的变量信息，其中 Name 栏为变量名，Size 栏为行列数，如变量 c 有 1 行 2 列数据；Bytes 栏为占用的总字节数；Class 为数据类型，Attributes 为属性。

如果数值超出了某种数值类型所能表示的范围，则取该类型能够表示的最大值或最小值，例如，

```
>> int8(300)↵
ans =
  127
```

转换函数还可以将其他数据类型转换成整型，例如

```
>> a=970;uint16(a);↵
>> whos↵
  Name        Size          Bytes  Class     Attributes
    a         1x1               8  double
    ans       1x1               2  uint16
```

甚至还可以将字符型数据转换为整型，例如，

```
>> str='Hello Matlab';int8(str)↵
ans =
   72  101  108  108  111   32   77   97  116  108   97   98
```

转换后的整数为相应字母的 ASCII 码。如果是中文字符，则需要能够包含 65535 在内的数值类型（如 uint16 或 int32 等），例如

```
>> uint16('你好，我的实验')↵
ans =
  20320  22909  65292  25105  30340  23454  39564
```

函数 isinteger 可以验证某变量是否为整数，例如

```
>> a=15;isinteger(a)↵
ans =
   0
```

使用函数 class 可以获知某数据的类型

```
>> a=uint8(15);class(a)↵
ans =
uint8
```

2．浮点数

浮点数又可以分为单精度（single）和双精度（double），区别在于表示的数值范围不同，单精度浮点数的存储需要 32 位，可以表示的数值范围为（–3.4028234663852886e+038～–1.1754943508222875e-038）和（1.1754943508222875e-038 ～3.4028234663852886e+038），双精度数的存储需要 64 位，可以表示的数值范围为（–1.7976931348623157e+308～–2.2250738585072014e-308）和（2.2250738585072014e-308～1.7976931348623157e+308），由此可见，尽管浮点数所能表示的数值范围不包括零在内，但 0.0 是浮点数的特例。由于 MATLAB 默认的数据类型为双精度浮点数，所以直接赋值操作就将某变量定义为了双精度浮点数，如果要定义单精度浮点数，则需要使用 single 转换函数，与双精度浮点数对应的转换函数为 double。例如

```
>> x=48.95;↵
>> y=single(x);↵
>> z=double(y);↵
>> whos↵
  Name      Size          Bytes  Class     Attributes
  x         1x1               8  double
```

```
  y         1x1            4       single
  z         1x1            8       double
```

利用 realmin 和 realmax 函数可以获取双精度浮点数的最小值和最大值，单精度浮点数的最小值和最大值使用'single'作为上述函数的参数即可，例如

```
>> realmin↙

ans =
  2.2251e-308

>> realmax('single')↙

ans =
  3.4028e+038
```

函数 isfloat 检验某个变量或数是否为浮点数。

3．复数

复数包括：实数部分和虚数部分，直接以 x+yi 的形式输入或使用 complex 函数定义复数，例如

```
>> x=5+3i↙

x =
   5.0000 + 3.0000i

>> y=complex(15,30)↙

y =
  15.0000 +30.0000i
```

利用函数 real 和 imag 可以获取复数的实数部分和虚数部分，例如

```
>> x=complex(rand(3),rand(3))↙

x =
   0.7952 + 0.6551i   0.4456 + 0.4984i   0.7547 + 0.5853i
   0.1869 + 0.1626i   0.6463 + 0.9597i   0.2760 + 0.2238i
   0.4898 + 0.1190i   0.7094 + 0.3404i   0.6797 + 0.7513i

>> real(x)↙

ans =
    0.7952    0.4456    0.7547
    0.1869    0.6463    0.2760
    0.4898    0.7094    0.6797

>> imag(x)↙

ans =
    0.6551    0.4984    0.5853
    0.1626    0.9597    0.2238
    0.1190    0.3404    0.7513
```

除此之外，MATLAB 中使用 inf 表示无穷大数，NaN 表示非数值，即不是一个数。如以 0 作被除数时会产生无穷大，而无穷大除以无穷大或 0 除以 0 会产生非数值，

```
>> 5/0↙

ans =
   Inf

>> 0/0↙

ans =
NaN
```

```
>> inf/inf↵
ans =
NaN
```

可以使用 isinf 和 isnan 函数来检验某数是否是无穷大或非数值。

2.2 字 符 型

MATLAB 中的字符串为 Unicode 字符数组，即每个字符都有对应的一个数值，定义字符串只需将字符内容置于单引号之内即可，例如

```
>> school='Soochow University'↵
school =
Soochow University
>> whos school↵
  Name       Size            Bytes  Class    Attributes
  school     1x18               36  char
```

变量 school 中包含 18 个字符（空格计在内），类型为字符型（char）

函数 ischar 可以判断某变量是否为字符型，可以使用函数 strcat 和 strvcat 进行字符串的横向和纵向拼接，char 函数也可以进行纵向拼接，例如

```
>> s1=strcat('Hello', 'everyone')↵
s1 =
Helloeveryone
>> s2=strvcat('Hello', 'everyone')↵
s2 =
Hello
everyone
>> s3=char('Hello', 'everyone')↵
s3 =
Hello
everyone
>> whos s1 s2 s3↵
  Name      Size            Bytes  Class    Attributes
  s1        1x11               22  char
  s2        2x8                24  char
  s3        2x8                24  char
```

可以看出函数 strvcat 和 char 会在较短字符串后添加空格，以使每个字符串等长，strcat 函数在横向拼接时会忽略字符串末尾的空格，例如

```
>> strcat('Hello  ', 'everyone')↵
ans =
Helloeveryone
```

避免这种情况的一种简便方法就是使用拼接运算符[]进行横向拼接，但在使用其进行纵向拼接时要确保每个字符的长度相同，例如

```
>> s1=['Hello  ' 'everyone']↵
s1 =
```

```
    Hello  everyone
    >> s2=['Hello ';'everyone']↵
    s2 =
    Hello
    everyone
```

char 函数可以将 ASCII 码转换为对应的字符，例如

```
    >> char(32:127)↵
    ans =
     !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
ijklmnopqrstuvwxyz{|}~
```

2.3　日期时间型

MATLAB 用于表示日期的格式有多种，可以以字符串形式表示，也可以以向量形式或者序列日期表示（公元 0000 年 1 月 1 日定义为序列 1，然后距离该日期的天数即为某日期的序列日期数）。

```
    >> date↵
    ans =
    06-Jul-2011
    >> whos ans↵
      Name      Size            Bytes  Class   Attributes
      ans       1x11               22  char
    >> clock↵
    ans =
      1.0e+003 *
       2.0110    0.0070    0.0060         0    0.0270    0.0011
    >> whos clock↵
      Name      Size            Bytes  Class     Attributes
      ans       1x6                48  double
    >> now↵
    ans =
      7.3469e+005
```

2.4　单　　元

单元是 MATLAB 中的一种特殊类型，它可以存储不同类型的数据，一般通过以下形式来建立单元（数组），将单元存储的数据放在{}内，例如

```
    >> c1={[1 2;3 4;5 6]}↵
    c1 =
        [3x2 double]
```

单元 c1 为包含 3×2 的二维浮点数组，如果向 c1 中增加其他内容，则变为单元数组，

```
    >> c1(2)={['Hello everyone']};↵
    >> c1(2,1)={date};↵
    >> c1(2,2)={now};↵
    >> whos c1↵
      Name   Size          Bytes    Class    Attributes
```

```
  c1     2x2          346     cell
```

可以通过函数 celldisp 来显示单元（数组）中的内容

```
>> celldisp(c1)
c1{1,1} =
     1     2
     3     4
     5     6
c1{2,1} =
06-Jul-2011

c1{1,2} =
Hello everyone

c1{2,2} =
  7.3469e+005
```

如果要显示其中某个元素的内容，则需要使用{}，例如

```
>> c1{1,1}
ans =
     1     2
     3     4
     5     6
```

2.5 结 构

结构数据类型适合于根据自己的需要定义具有不同含义的字段，并且其中可以包含任何类型的数据，如在被试信息结构中可以包含姓名、性别、年龄、专业和优势等，可以这样定义上述结构信息，

```
>> subject.name='张三';
>> subject.gender='Male';
>> subject.age=18;
>> subject.major='Psychology';
>> subject.eye='Left';
>> whos subject
  Name        Size            Bytes  Class     Attributes
  subject     1x1               668  struct
>> subject
subject =
            name:   '张三'
            gender: 'Male'
            age:    18
            major:  'Psychology'
            eye:    'Left'
```

如果要定义多个被试，则可以采用结构数组的形式，下面利用 struct 函数来建立结构数组，

```
>> subject(2:3)=struct('name',{'李四','王五'},'gender',{'Female','Male'},
'age',{20 19},'major',{'English','Maths'},'eye',{'Left','Right'});
```

如果要查看某被试的信息，则可以通过索引来查看，例如

```
>> subject(2)
ans =
            name:   '李四'
            gender: 'Female'
```

```
        age:    20
        major:  'English'
        eye:    'Left'
```

如果要显示或引用某个字段的内容，使用.运算符即可，例如

```
>> subject(:).name↙
ans =
张三

ans =
李四

ans =
王五
```

或者将结果放入某数组中，

```
>> [subject(:).age]↙
ans =
   18    20    19
```

第 3 章　MATLAB 编程基础

3.1　常数与常量

常数或常量是其值不会发生变化的量，有的是常数，如圆周率；有的是通过函数返回的与系统有关的固定值，如 MATLAB 的安装路径、计算机操作系统等。既可以在命令窗口中获取这些值，也可以在程序中使用这些值，例如

```
>> pi↙
ans =
3.1416
>> version↙
ans =
7.11.0.584 (R2010b)
>> intmax('int32')↙
ans =
  2147483647
>> realmin↙
ans =
  2.2251e-308
```


3.2　变　量

变量相对于常数而言，是可以变化的一个值，它可以是数值、字符、日期、单元、结构等类型的数据。MATLAB 不同于 C 语言，不需要事先声明变量，可以在任何需要的时候直接对变量进行赋值，MATLAB 会自动建立变量并分配内存空间。变量就如同某个值的别名一样，对变量进行操作就是对变量所表示的值的操作。

在 MATLAB 中定义的变量名必须以字母开头，其后可以组合数字、下画线，但中间不可以有空格，另外不能使用 MATLAB 中已经定义的函数名和关键字，变量名的长度没有限制（足够用），区分大小写（即变量 abc 和 Abc 不同），变量名最好具有可读性，还要注意减少重复定义变量的机会，避免由于使用相同变量名而更改具有不同意义的值。

下面的变量名是合法的，Myfile、picturenames、m1、file_type、tempstr、fhandle；下面的变量是非法的，3file、file ab、data-m1、file$9。

3.3　变 量 类 型

3.3.1　局部变量

在函数体内定义的变量为局部变量，只能被定义的函数所访问，函数运行结束，局部变量所占用的内存空间自动释放（局部变量也就不存在了），不同的函数可以使用相同的局部变量名，因为不同的函数有独立的存储空间。

3.3.2　全局变量

全局变量需要使用 global 关键字来声明，不同函数可以共享全局变量的内容，当一个函数对全局变量进行修改后，其他函数所读取的全局变量值也发生相应变化。

3.3.3　永久变量

永久变量（persistent）只能在 M 文件的函数中定义和使用，需要在使用该变量的前面定义，其初始化值为空（[ ]）。与局部变量不同的是，即使函数退出，永久变量也不会从内存中清除，当下次调用该函数时，仍保留退出前的变量值。

下面的例子中将变量 avalue 定义为永久变量，每调用一次函数 plusone，永久变量 avalue 的值增加 1，并将其赋值给返回参数 ret，程序代码如下：

```
%plusone.m      （文件位于 Psyfeng\Little Examples 目录下）
function ret=plusone
persistent avalue; %定义变量 avalue 为永久变量
if isempty(avalue)
   avalue=1; %如果变量值为空，则对其赋值 1
else
   avalue=avalue+1; %如果不为空，则变量值加 1
end
ret=avalue; %将变量值赋予返回参数 ret
end
```

然后在命令窗口中调用函数 plusone，结果如下：

```
>> plusone↙
ans =
     1
>> plusone↙
ans =
     2
>> plusone↙
ans =
     3
```

可以看出，每执行函数 pluseone 一次，返回值增加 1。

3.4　关　键　字

通过函数 iskeyword 列出 MATLAB 中系统使用的关键字

```
>> iskeyword↙
ans =
    'break'
    'case'
    'catch'
    'classdef'
    'continue'
    'else'
    'elseif'
```

```
'end'
'for'
'function'
'global'
'if'
'otherwise'
'parfor'
'persistent'
'return'
'spmd'
'switch'
'try'
'while'
```

3.5 运 算 符

运算符包括：算术运算符、关系运算符和逻辑运算符。

3.5.1 算术运算符

1. 矩阵运算

矩阵的运算要符合相应的运算规则（此处不再赘述）。

（1）加减（+、–）

```
>> a=[4 5;7 3]↙
a =
     4     5
     7     3
>> b=[2 9;10 6]↙
b =
     2     9
    10     6
>> a+b↙
ans =
     6    14
    17     9
>> a-b↙
ans =
     2    -4
    -3    -3
>> a+10↙
ans =
    14    15
    17    13
```

（2）乘除（*、/）

```
>> a=[4 5;7 3;10 6]↙
a =
     4     5
     7     3
    10     6
```

```
>> b=[2 9;10 6;-1 -3]↙
b =
     2     9
    10     6
    -1    -3
>> a*b'↙
ans =
     53    70   -19
     41    88   -16
     74   136   -28
>> a/b↙
ans =
         0.3333  0.3333  0
        -0.1538  0.7308  0
        -0.0000  1.0000  0
```

（3）右除（\）

```
>> a=rand(3,2)↙
a =
    0.9572    0.1419
    0.4854    0.4218
    0.8003    0.9157
>> b=rand(3,2)↙
b =
    0.7922    0.0357
    0.9595    0.8491
    0.6557    0.9340
>> a\b↙
ans =
     0.8690   -0.0998
     0.1783    1.2788
```

（4）乘方（^）

```
>> a=rand(3)↙
a =
    0.5060    0.9593    0.1493
    0.6991    0.5472    0.2575
    0.8909    0.1386    0.8407
>> a^3↙
ans =
     1.6561    1.6428    0.8004
     1.6031    1.5407    0.7914
     2.1680    1.9382    1.1993
```

（5）转置（'），行列互换

```
>> a=rand(3)↙
a =
    0.2543    0.9293    0.2511
    0.8143    0.3500    0.6160
    0.2435    0.1966    0.4733
```

```
>> a'↙
ans =
    0.2543    0.8143    0.2435
    0.9293    0.3500    0.1966
    0.2511    0.6160    0.4733
```

(6) 索引 (:)

```
>> a=rand(5)↙
a =
   0.3517    0.2858    0.0759    0.1299    0.1622
   0.8308    0.7572    0.0540    0.5688    0.7943
   0.5853    0.7537    0.5308    0.4694    0.3112
   0.5497    0.3804    0.7792    0.0119    0.5285
   0.9172    0.5678    0.9340    0.3371    0.1656
>> a(:,2)↙
ans =
   0.2858
   0.7572
   0.7537
   0.3804
   0.5678
>> a(:,2:end)↙
ans =
    0.2858    0.0759    0.1299    0.1622
    0.7572    0.0540    0.5688    0.7943
    0.7537    0.5308    0.4694    0.3112
    0.3804    0.7792    0.0119    0.5285
    0.5678    0.9340    0.3371    0.1656
```

2. 数组运算

数组运算符是数组间元素与元素间的运算。

(1) 点乘 (.*)

```
>> a=randi([1 9],[3,2])↙
a =
    7    1
    8    4
    8    3
>> b=randi([1 9],[3,2])↙
b =
   8    2
   4    3
   9    2
>> a.*b↙
ans =
    56     2
    32    12
    72     6
```

(2) 点左除 (./)

```
>> a./b↙
```

```
ans =
    0.8750    0.5000
    2.0000    1.3333
    0.8889    1.5000
```

（3）点右除（.\）

```
>> a.\b
```

```
ans =
    1.1429    2.0000
    0.5000    0.7500
    1.1250    0.6667
```

（4）点乘方（.^）

```
>> a.^b
```

```
ans =
    5764801          1
       4096         64
  134217728          9
```

（5）点转置（.'）

```
>> (a+b*i)'
```

```
ans =
   7.0000 - 8.0000i   8.0000 - 4.0000i   8.0000 - 9.0000i
   1.0000 - 2.0000i   4.0000 - 3.0000i   3.0000 - 2.0000i
```

```
>> (a+b*i).'
```

```
ans =
   7.0000 + 8.0000i   8.0000 + 4.0000i   8.0000 + 9.0000i
   1.0000 + 2.0000i   4.0000 + 3.0000i   3.0000 + 2.0000i
```

3.5.2　关系运算符

关系运算符用于大小相等比较，可以是两个单独的元素之间，也可以是两个数组之间进行比较。

1. 大于小于（>、<）

```
>> a=randi([1 9],[3,2])
```

```
a =
    9    8
    4    4
    2    3
```

```
>> b=randi([1 9],[3,2])
```

```
b =
    4    9
    1    9
    2    6
```

```
>> a>b
```

```
ans =
     1     0
     1     0
     0     0
```

```
>> a>5
```

```
ans =
```

```
     1     1
     0     0
     0     0
>> b<3↵
ans =
     0     0
     1     0
     1     0
```

2. 大于等于和小于等于（>=、<=）

```
>> a=randi(10,[1,5])↵
a =
     6     3     4     5     3
>> a>=5↵
ans =
     1     0     0     1     0
>> a<=3
ans =
     0     1     0     0     1
```

3. 等于和不等（==、~=）

```
>> a=randi(10,[1,5])↵
a =
     9     2     3     2     3
>> a==2↵
ans =
     0     1     0     1     0
>> a~=2
ans =
     1     0     1     0     1
>> a='Hello';↵
>> b='hello';↵
>> a==b↵
ans =
     0     1     1     1     1
```

3.5.3 逻辑运算符

1. 逻辑与（&、&&），两个对象均为真，结果才为真

```
>> a=5;b=3;c=0;↵
>> disp(a & b)↵
     1
>> disp(a && c)↵
     0
```

2. 逻辑或（|、||），两个对象中只要有一个为真，结果就为真

```
>> a=5;b=3;c=0;↵
>> disp(a | b)↵
     1
>> disp(b || c)↵
```

```
     1
```

3．逻辑非（~），原结果为真，则变为假，原结果为假，则改变为真

```
>> a=5;b=3;c=0;↵
>> ~a↵
ans =
     0
>> ~c↵
ans =
     1
>> a=[];↵
>> ~isempty(a)↵
ans =
     0
```

3.5.4　逻辑运算函数

1．异或运算（xor），两个对象一个为真，一个为假时，结果为真；否则结果为假

```
>> a=5;b=3;c=2;↵
>> xor(a>b,b>c)↵
ans =
     0
>> xor(a<b,c<b)↵
ans =
     1
```

2．位与运算（bitand），非负整数二进制位之间的与运算

```
>> bitand(3,5)↵
ans =
     1
```

3．位或运算（bitor）

```
>> bitor(3,5)↵
ans =
     7
```

4．位反运算（bitcmp）

```
>> bitcmp(uint8(3))↵
ans =
  252
```

5．位异或运算（bitxor）

```
>> bitxor(3,5)↵
ans =
     6
```

6．任意为真（any）

```
>> a=randi([0 1],4)↵
a =
     0     1     0     1
```

```
     0     0     1     0
     0     0     1     0
     0     1     0     1
>> any(a)↙
ans =
     0     1     1     1
>> any(a,2)↙
ans =
     1
     1
     1
     1
```

函数 any 的第 2 个参数指定在行或列上进行判断，默认值为 1，在列上判断，即某列的各个值只要有任意一个为真，即为真；如果参数值为 2，则在行上判断，即某行的各个值只要有任意一个为真，即为真。

7．所有为真（all）

```
>> a=randi([0 1],4)↙
a =
     1     1     1     1
     1     1     0     0
     0     0     1     0
     0     0     1     0
>> all(a)↙
ans =
     0     0     0     0
>> all(a,2)↙
ans =
     1
     0
     0
     0
```

函数 all 的第 2 个参数的含义与函数 any 相同。

3.6 过 程 控 制

过程控制用于控制程序的流程，如不同条件下执行的代码片段不同，常用的过程控制结构有如下几种。

3.6.1 if 条件语句

对于简单的 if 条件语句，可采用如下形式（如果条件表达式 expression 为值，则执行 statements 语句）

```
if expression, statements, end
```

对于复杂的 if 条件语句，可采用下面的形式（更具有可读性）。如果表达式 expression1 为真，则执行 statements1 语句并结束判断。如果表达式 expression1 为假，则判断表达式 expression2 是否为真，

如果为真，则执行 statements2 语句并结束判断；如果表达式 expression2 为假，以此类推，如果所有带 if 字样的分支都不满足，则执行 else 后面的语句 statements3（如果有 else 的话）。

```
if expression1
   statements1
elseif expression2
   statements2
else
   statements3
end
```

程序示例：根据学生编号的奇偶平衡反应按键。

```
if mod(subid,2) %被试编号存储在变量 subid 中
   correctKey=KbName('f'); %如果是奇数，正确按键为 F 键
else
   correctKey=KbName('j'); %否则，如是偶数，正确按键为 J 键
end
```

或者写成

```
if mod(subid,2),correctKey=KbName('f');else,correctKey=KbName('j');end
```

程序示例：根据 IQ 进行等级划分

```
if iq>=130
   desc='超常';
elseif iq>=120
   desc='优秀';
elseif iq>=110
   desc='良好';
elseif iq>=100
   desc='中上';
elseif iq>=90
   desc='中等';
elseif iq>=70
   desc='中下';
else
   desc='较差';
end
```

在上面的例子中，需要注意分界值应按一定顺序给出，否则可能会得不出正确结果。例如，下面代码中交换了分界值 100 和 110 的位置，当 iq=115 时，变量 desc 的值为“中上”。

```
if iq>=130
   desc='超常';
elseif iq>=120
   desc='优秀';
elseif iq>=100
   desc='中上';
elseif iq>=110
   desc='良好';
elseif iq>=90
   desc='中等';
elseif iq>=70
   desc='中下';
else
   desc='较差';
end
```

if 条件语句还可以嵌套。假如，一个 2×2 实验设计，两个自变量的水平值分别用 1 和 2 来表示，我们通过 if 语句来设置需要呈现的图片名称（当然有更简单的方法实现），程序示例如下（其中 v1 和 v2 为两个自变量）：

```
if v1==1
    if v2==1
        filename='f1.bmp';
    else
        filename='f2.bmp';
    end
else
    if v2==1
        filename='f3.bmp';
    else
        filename='f4.bmp';
    end
end
```

上面语句也可以利用关系运算符改写为如下形式：

```
if v1==1 && v2==1
    filename='f1.bmp';
elseif v1==1 && v2==2
    filename='f2.bmp';
elseif v2==1 && v2==1
    filename='f3.bmp';
else
    filename='f4.bmp';
end
```

3.6.2 switch 条件语句

switch 条件语句的结构如下：

```
switch switch_expr
 case case_expr
    statement1, ..., statement2
  case {case_expr1, case_expr2, case_expr3, ...}
    statement3, ..., statement4
  otherwise
    statement5, ..., statement6
end
```

根据表达式 switch_expr 的值转入对应的条件分支，即如果表达式 switch_expr 的值符合 case_expr 条件，则执行语句 statement1, … statement2；如果符合 case_expr1，case_expr2，case_expr3 中任何一个条件，则执行语句 statement3, …, statement4；当上述条件均不满足，并且有 otherwise 关键词时，则执行其后的语句 statement5, …, statement6。在判断的过程中，按照 case 语句顺序，只要符合当前的 case 语句，则后面的 case 语句及 otherwise 语句均略过。

程序示例：根据月份字符判断季度

```
switch month_expr
    case {'Jan','Feb','Mar'}
        disp('First Season');
    case {'Apr','May','Jun'};
        disp('Second Season');
    case {'Jul','Aug','Sep'};
```

```
        disp('Third Season');
    case {'Oct','Nov','Dec'};
        disp('Fourth Season');
end
```

如果变量 month_expr 的值为“Jul”，则输出显示为：Third Season，可以在命令窗口中先对变量赋值，然后将上述代码粘贴至命令窗口中查看执行结果。

程序示例：假设一 5 点量表，根据其等级显示对应的文本描述

```
switch select
    case 1
        option='完全不同意';
    case 2
        option='基本不同意';
    case 3
        option='中立';
    case 4
        option='基本同意';
    case 5
        option='完全同意';
otherwise
        option='未做选择';
end
```

3.6.3　for 循环语句

for 循环结构如下：

```
for index = values
program statements
        :
end
```

其中 values 可以取以下三种形式：

（1）initval:endval，initval 代表初始值，endval 代表结束值，中间用冒号（:）隔开；

（2）initval:step:endval，step 代表步长，前一种形式每次递增值为 1，此处可以指定每次递增值，如果为负值，则递减；

（3）valArray，循环按列遍历数组中的值。

注意：使用 break 可以提前中止 for 循环。

程序示例：计算 1～100 的和

```
sums=0;
for i=1:100
    sums=sums+i;
end
disp(sums);
```

程序示例：计算 1～100 间所有奇数之和

```
sums=0;
for i=1:2:100
    sums=sums+i;
end
disp(sums);
```

程序示例：计算 1～100 间 5 的倍数之和

```
sums=0;
for i=100:-5:1
   sums=sums+i;
end
disp(sums);
```

程序示例：数组值累加

```
sums=0;
for i=[27 34 18 19 52 25]
   sums=sums+i;
end
disp(sums);
```

程序示例：数组值累加

```
sums=0;
a=rand(3,5)
for i=a
   sums=sums+i;
end
disp(sums);
```

结果如下：

```
a =
   0.7513   0.6991   0.5472   0.2575   0.8143
   0.2551   0.8909   0.1386   0.8407   0.2435
   0.5060   0.9593   0.1493   0.2543   0.9293

   3.0694
   2.3689
   2.7981
```

程序示例：显示 10 以内的偶数

```
for i=1:10
   if mod(i,2)
      continue; %如果是奇数，则跳过，如果 for 循环存在嵌套，使用 continue 语句作用于
                  包含该语句的内层循环
   end
   disp(i);
end
```

3.6.4 while 循环语句

while 循环结构如下，只要条件表达式成立，就执行中间的语句，直至条件表达式不成立，循环中间需要改变表达式的值或使用循环中断指令（break），否则将有可能限于死循环。

```
while expression
program statements
        :
end
```

程序示例：循环显示文件信息，文件命名规则为，主文件名后加 1, 2, 3, …, n 后缀，通过函数 exist 判断文件是否存在，如果存在，则利用函数 imfinfo 显示文件信息；如果不存在，则中止循环。

```
filename='a1.bmp';
index=1;
```

```
while exist(filename,'file')
   imfinfo(filename)
   index=index+1;
   filename=['a' num2str(index) '.bmp'];
end
```

程序示例：计算 1～100 的累加和

```
i=1;
sums=0;
while i<=100
   sums=sums+i;
   i=i+1;
end
disp(sums);
```

程序示例：随机数累加，直至产生的随机数大于 0.7

```
sums=0;
while 1
   a=rand
   if a>0.7
      break; %如果产生的随机数大于 0.7，则跳出 while 循环，与 continue 类似，如果存在
             循环嵌套，则跳出包含该语句的内层的循环
   end
   sums=sums+a;
end
disp(sums);
```

3.6.5　try 错误控制语句

结构如下，在处理程序代码 program statements 部分时如果发生异常，则由 error-handling statements 处的程序代码来处理。

```
try
  program statements
        :
catch exception
  error-handling statements
        :
end
```

程序示例：当读取图片文件出错时，处理是否因文件扩展名使用有误所造成的。

```
function d_in = read_image(filename) （文件位于 Psyfeng\Little_Examples 目录下）
[path name ext] = fileparts(filename); %将文件拆分为路径、前缀和后缀
try %尝试打开文件
  fid = fopen(filename, 'r');
  d in = fread(fid);
catch exception %如果文件打开失败

  if ~exist(filename, 'file') %判断是否是因为未找到文件所致

    % 尝试改变文件的扩展名
    switch ext
    case '.jpg'    % Change jpg to jpeg
       altFilename = strrep(filename, '.jpg', '.jpeg')
    case '.jpeg'   % Change jpeg to jpg
       altFilename = strrep(filename, '.jpeg', '.jpg')
```

```
        case '.tif'    % Change tif to tiff
           altFilename = strrep(filename, '.tif', '.tiff')
        case '.tiff'   % Change tiff to tif
           altFilename = strrep(filename, '.tiff', '.tif')
        otherwise
          rethrow(exception);
        end
        %更改文件扩展名后再进一步尝试打开文件
        try
          fid = fopen(altFilename, 'r');
          d_in = fread(fid);
        catch
         %如果仍然无法打开文件，则抛出异常，由系统来处理
          rethrow(exception)
        end
      else %如果文件存在，但无法打开
        d_in=-1;
        return; %return 语句可以提前返回（中止）某脚本或函数，其后的语句不管有多少均不再执行
      end
    end
```

第4章　程 序 设 计

MATLAB 中所编写的程序文件以.m 为扩展名，统称为 M 文件，其形式有两种：脚本和函数，前者是 MATLAB 语句序列，不能接受参数，也不返回结果，但可以将其中所定义的变量存储在工作区中；后者可以接受参数和返回结果。由于函数比脚本更灵活，更具有普适性，所以应用较广。MATLAB 提供了程序开发的编辑工具 Editor，通过菜单 File→New→Script 或 File→New→Function 可以打开程序编辑器，也可以在命令窗口中直接输入 edit 指令打开如图 4-1 所示的编辑器，其编辑功能与 Windows 的记事本相似，但具有语法着色功能（关键字用不同颜色表示），我们可以先忽略不太熟悉的界面和菜单。

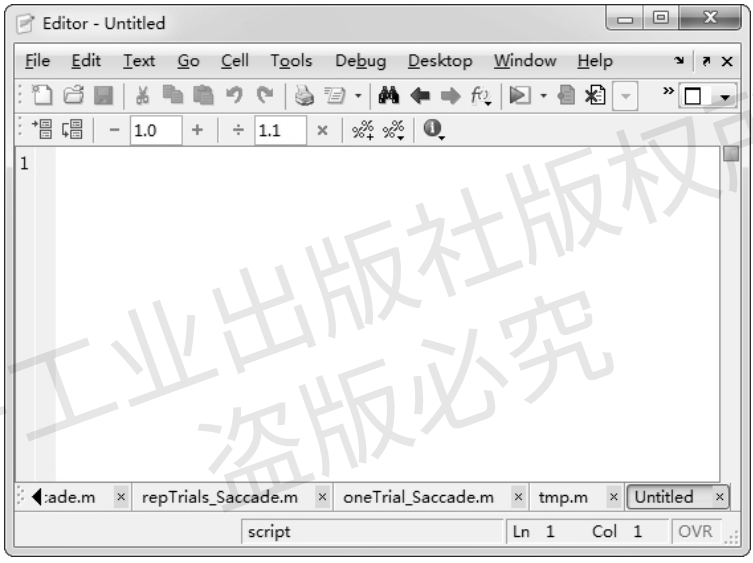

图 4-1　程序编辑器

脚本的名称即为 M 文件的文件名，而函数名称则分为，当函数是主函数时，函数名称与 M 文件名相同（可以不同，但建议取相同的名称），而子函数则与 M 文件名不同，甚至可以没有名称（匿名函数）。而实际上，脚本也可以使用函数的形式，只不过不需要输入参数和返回参数之类的内容罢了。

4.1　M 文件的建立

M 文件可以使用任意的文本编辑器来创建，在保存文件时必须用.m 作为扩展名，建议使用 MATLAB 自带的程序编辑器（Editor）。

在程序编写时，为了便于自己或他人知晓脚本或函数的功能，可以使用注释来对程序代码和参数的含义进行说明。M 文件具有建议的规范格式，对于函数而言，示例如下。

```
function f = fact(n)                          函数定义行，脚本无此行，否则为函数
% Compute a factorial value.                 H1 行
% FACT(N) returns the factorial of N,        帮助文本
% usually denoted by N!
```

```
% Put simply, FACT(N) is PROD(1:N).        注释
f = prod(1:n);                             函数体
disp(f); %Display the result               行内注释
%{
The above function f defines the factorial value of an integer, for convenient,
   just use the function prod to compute the product of elements.
%}
end %函数结束
```

（1）函数定义行负责定义函数的名称以及输入和输出参数的个数和顺序，函数名称最好一目了然，即从名字上就可以大略知晓其功能；输入参数需放在（）内，如果有多个参数，则用逗号分隔开；输出参数如果有多个，则需将它们放置在[ ]内，多个输出参数间用空格或逗号分隔，函数定义语句的格式如下所示：

function [out1, out2, ...] = myfun(in1, in2, ...)

其中 function 是关键字；myfun 是函数名；in1, in2 是输入参数；out1,out2 是输出参数。

（2）H1 行为紧跟函数定义行之后以%打头的一行，在命令窗口输入 help 函数名，可显示该行信息，一般用作函数功能的说明。

（3）帮助文本是 H1 行接下来以%打头的一行或多行，直到第一个非百分号（%）打头的行结束；注意 H1 行和帮助文本之间不能有空行，否则使用 help 命令无法获取帮助文本信息。

（4）注释行以%开始，可以单独作为一行，也可以放在代码的末尾，如果注释的信息较多需要用多行时，则使用%{ 和 %}符号对。当然你可以省略掉所有注释和帮助文本以及 H1 行，这不会妨碍程序的正常运行。

4.2 脚 本 示 例

下面是一段简单的脚本程序示例，程序的功能是播放随机生成的纯音，每个纯音的频率介于 500～1000 赫兹之间，其长度为 0.1 秒，采样频率为 44 100 赫兹，代码如下。

文件名：playnoise.m（文件位于 Psyfeng\Little_Examples 目录下）

```
%Play some noise :)
%First generate the beep with random freq,
%then play it with function sound

%We use the MakeBeep to generate the noise
sf=44100; %声音采样频率
for i=1:50 %循环 50 次（播放 50 次）
    r=randi([500,1000]); %生成随机频率
    s1=MakeBeep(r,0.1,sf); %生成噪音 0.1 秒
    sound(s1,sf); %播放
end
```

可以在命令窗口中输入以下命令来运行脚本。

```
>> playnoise↵
```

4.3 函 数 示 例

在前面的脚本示例中，程序编写好后，假如要使用其他的采样频率、声音频率或时间长度，均需要通过修改程序源代码后，才能得到想要的效果，接下来看函数在这方面的灵活性。

文件名：playnoisef.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef(sf,duration,lowfreqs,highfreqs,loops)
%Play some noise :)
%First generate the beep with random freq,
%then play it with function sound
%The meaning of the parameter as follows:
%playnoisef(sf,duration,freqs,loops)
%sf:samplerate,such as 44100
%duration:the duration of each noise in secs. such as 0.5
%freqs:the range of the freqs, such as [100 1000]
%loops:the number of loops

%We with the MakeBeep to generate the noise
for i=1:loops %循环播放
    r=randi([lowfreqs highfreqs]); %从范围中随机频率
    s1=MakeBeep(r,duration,sf); %生成噪音
    sound(s1,sf); %播放
end
end
```

4.4　程序（M 文件）的运行

程序的运行方式和 MATLAB 中内置函数的运行方式一致，可分为两种形式：一种是在 MATLAB 开发集成环境下，即在命令窗口中输入脚本文件名或主函数名即可，例如，对于 4.2 节和 4.3 节中的示例可直接在命令窗口中输入，也可以直接在程序编辑器中通过菜单 Debug→Run 或快捷键 F5 或工具栏 快捷按钮方式运行。

```
>> playnoise
```

如果是函数，一种方式是把参数放置括号内，参数之间用逗号分隔；一种方式是不用括号，中间用空格把参数值隔开，但这种方式的弊端，一是无法获得函数的返回值，二是会把输入的参数当成字符而非你想要的数值型（所以建议你用括号形式），例如

```
>> playnoisef(8890,0.05,[500 5000],20)
```

如果要获取自编程序的使用帮助信息，输入 help 名称

```
>> help playnoise
```

```
 Play some noise :)
 First generate the beep with random freq,
 then play it with function sound
```

如果要确定程序所在目录，可以使用 which 命令，例如

```
>> which playnoise
```

另外一种程序的运行方式，就是使用 MATLAB 的 mcc 编译器，将 m 程序编译为可执行文件（exe 文件），这样系统不需要安装 MATLAB 就可以运行可执行文件，其编译格式为 mcc　–m　m 文件，例如

```
>> mcc -m playnoise
```

如果要运行在其他未安装 MATLAB 的计算机上，还需要安装 MCRInstaller.exe，该文件位于 MATLAB 安装目录下的 toolbox\compiler\deploy\win32\子目录中。但如果是函数，就需要特殊处理输入参数的非数值现象（参见 4.5 节）。

4.5 函数参数的处理方式

对比 4.2 节和 4.3 节中的示例，你会发现函数的灵活性，因为在运行程序时，只要给出不同参数就可以得到想要的结果，这要比更改程序的代码方便得多。如果某函数的使用频率较高且参数较多，每次输入参数值就比较麻烦，而且当参数输入有误时，程序就会无法正确运行，如 4.3 节中的示例而言，如果输入以下内容

```
>> playnoisef(8890,0.05,500, 5000)↙
??? Input argument "loops" is undefined.

Error in ==> playnoisef at 13
for i=1:loops %循环 50 次（播放 50 次）
```

上面显示了错误信息，因为少输入一个表示循环次数（播放次数）的参数，就无法找到变量 loops，错误出现在程序 playnoisef.m 中的第 13 行，并且给出了 13 行的内容“for i=1:loops %循环 50 次（播放 50 次）”，这是大家不希望看到的情况。下面来看处理这种情况的方法。

4.5.1 默认处理

采用默认参数值，即如果没有输入某个参数值，则使用默认值。

利用函数 nargin 可以获取某函数输入参数的个数，根据输入参数的个数来设置默认值，示例如下：

文件名：playnoisef1.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef1(sf,duration,lowfreqs,highfreqs,loops)
if nargin<1 %如果没有任何参数
   sf=44100;
   duration=0.1;
   lowfreqs=500;
   highfreqs=1000;
   loops=10;
elseif nargin<2 %如果只有一个参数
   duration=0.1;
   lowfreqs=500;
   highfreqs=1000;
   loops=10;
elseif nargin<3 %如果输入 2 个参数
   lowfreqs=500;
   highfreqs=1000;
   loops=10;
elseif nargin<4 %如果输入了 3 个参数
   highfreqs=1000;
   loops=10;
elseif nargin<5 %如果输入了 4 个参数
   loops=10;
end
for i=1:loops %循环播放
   r=randi([lowfreqs highfreqs]); %从范围中随机频率
   s1=MakeBeep(r,duration,sf); %生成噪音
   sound(s1,sf); %播放
end
end
```

对于上面的示例程序（playnoisef1.m），当利用 mcc 生成可执行文件后，通过开始→运行…（WinXP 系统下）或开始→搜索程序或文件（Win7 系统下），以 Win7 为例，在其中输入 cmd，然后选择 cmd（或者通用方法：开始→（所有）程序→附件→命令提示符），打开命令窗口，假设编译生成的可执行文件在 D 盘根目录下，切换至 D 盘，然后输入 playnoisef1↙，就可以执行程序，但如果使用参数，如 playnoisef1 22500↙，就会出现错误，在 MATLAB 命令窗口中这样运行也同样出错，原因是此时 22500 不再是数值型，而是字符型，显然提供的参数类型与所要求的不符。

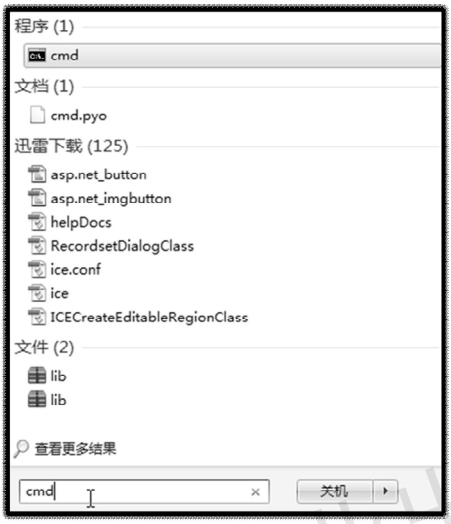

图 4-2　搜索 cmd 命令

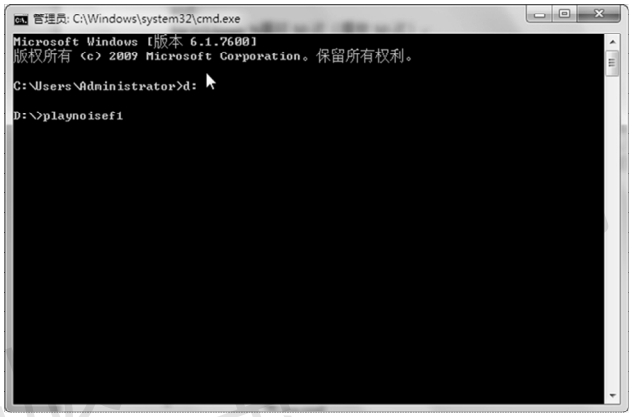

图 4-3　DOS 命令窗口

处理上面错误的一种方法就是使用转换函数 str2double，将字符型转换成数值型。

文件名：playnoisef2.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef2(sf,duration,lowfreqs,highfreqs,loops)
if nargin<1 %如果没有任何参数
    sf=44100;
    duration=0.1;
    lowfreqs=500;
    highfreqs=1000;
    loops=10;
elseif nargin<2 %如果只有一个参数
    duration=0.1;
    lowfreqs=500;
    highfreqs=1000;
    loops=10;
elseif nargin<3 %如果输入 2 个参数
    lowfreqs=500;
    highfreqs=1000;
    loops=10;
elseif nargin<4 %如果输入 3 个参数
    highfreqs=1000;
    loops=10;
elseif nargin<5 %如果输入 4 个参数
    loops=10;
end
if ischar(sf) %如果是字符型，则将其转换为数值型
```

```
    sf=str2double(sf);
end
if ischar(duration)
    duration=str2double(duration);
end
if ischar(lowfreqs)
    lowfreqs=str2double(lowfreqs);
end
if ischar(highfreqs)
    highfreqs=str2double(highfreqs);
end
if ischar(loops)
    loops=str2double(loops);
end
for i=1:loops %循环播放
    r=randi([lowfreqs highfreqs]); %从范围中随机频率
    s1=MakeBeep(r,duration,sf); %生成噪音
    sound(s1,sf); %播放
end
end
```

改动之后，就可以在 MATLAB 命令窗口或 DOS 窗口中，直接使用如下形式来运行程序

```
playnoisef2 3800 0.05 5000 10000 10↙
```

4.5.2 命令行输入

通过提示使用者输入来设置参数，对程序 playnoisef1.m 做如下改动（本例中可以将 function 语句去掉）。

文件名：playnoisef 3.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef3
sf=input('输入采样频率（回车默认值为 44100）：');
if isempty(sf)
    sf=44100;
end
duration=input('输入声音长度（回车默认值为 0.5 秒）：');
if isempty(duration)
    duration=0.5;
end
lowfreqs=input('输入声音频率下限（回车默认值 500Hz）：');
if isempty(lowfreqs)
    lowfreqs=500;
end
highfreqs=input('输入声音频率上限（回车默认值 1000Hz）：');
if isempty(highfreqs)
    highfreqs=1000;
end
loops=input('输入循环次数（回车默认值 10 次）：');
if isempty(loops)
    loops=10;
end
for i=1:loops %循环 50 次（播放 50 次）
    r=randi([lowfreqs highfreqs]); %从范围中随机频率
    s1=MakeBeep(r,duration,sf); %生成噪音
    sound(s1,sf); %播放
```

```
    end
    end
```

4.5.3　对话框设置参数

利用 MATLAB 提供的输入对话框 inputdlg 函数（参见 8.8 节）可以方便快捷地设置参数输入界面，注意下面示例文件中与 function 对应的 end 关键字。

文件名：playnoisef4.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef4
paras=getmyParameter; %获取参数
if isempty(paras)
    return;
else
    sf=str2double(paras{1});
    duration=str2double(paras{2});
    freqs=sscanf(paras{3},'%d');
    loops=str2double(paras{4});
end
for i=1:loops %循环播放
    r=randi(freqs); %从范围中随机频率
    s1=MakeBeep(r,duration,sf); %生成噪音
    sound(s1,sf); %播放
end
end
function paras=getmyParameter  %获取参数的函数
paras=inputdlg({'采样频率','声音长度','声音频率范围','循环次数'},'参数设置',1,
{'44100','0.05','500 1000','10'}); %调用 inputdlg 函数，将返回结果存储于 paras
end
```

运行程序 playnoisef4，首先出现如图 4-4 所示的对话框来设置相应的参数。参数设置完毕后，点击 OK 按钮，就会播放一定频率范围内的纯音。

图 4-4　参数设置窗口

尽管对话框式的参数设置比较方便，也较为直观，但如果实验过程中发现设置的默认值不理想，这样每次运行程序就都需要更改某些内容，而且如果实验开始前忘记更改，对实验结果的影响可想而知，因此需要程序能够记录下上次所设置的参数值，下次执行程序时能够自动将前一次使用的参数填充到图 4-4 所示的对话框。

4.5.4　参数值的记忆与存取

可以利用 MATLAB 提供的文件读写函数 fgetl 和 fprintf 将设置的参数值写入某个文件中，当需要时再从文件中读取来使用。

文件名：playnoisef5.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function playnoisef5
mainfilename=mfilename; %利用 mfilename 函数获取文件名信息
paras=getmyParameter(mainfilename); %将文件名作为参数传给 getmyParameter
if isempty(paras) %如果在参数设置窗口中选择 Cancel，则返回值为空，退出程序
    return;
else
    sf=str2double(paras{1}); %将各个参数转换为数值
    duration=str2double(paras{2});
    freqs=sscanf(paras{3},'%d');
    loops=str2double(paras{4});
```

```
end
for i=1:loops %循环播放
    r=randi(freqs); %从范围中随机频率
    s1=MakeBeep(r,duration,sf); %生成噪音
    sound(s1,sf); %播放
end
end
function paras=getmyParameter(configfilename)
prompt={'采样频率','声音长度','声音频率范围','循环次数'};
dlg_title='参数设置';
default answer=[];
if exist([configfilename '.ini'],'file')      %如果有配置文件，尝试使用之
    fid=fopen([configfilename '.ini'],'r');   %打开配置文件
    if fid~=-1 %如果文件打开成功
        while true
            linestr=fgetl(fid); %逐行读取文件中内容
            if ~ischar(linestr), break, end %如果读至文件尾，则退出 while 循环
            default answer=[default answer {linestr}]; %将读取内容拼接
        end
        fclose(fid); %关闭文件
    else
        error(['无法打开配置文件：' configfilename '.ini']);
    end
else %如果没有配置文件就使用默认值
    default answer={'44100','0.1','500 1000','10'};
end
paras=inputdlg(prompt,dlg_title,1,default answer);
if ~isempty(paras) %如果 paras 为非空
    fid=fopen([configfilename '.ini'],'w');   %打开文件，写入参数值
    if fid~=-1
        for i=1:length(paras)
            fprintf(fid,'%s\r\n',paras{i});     %循环写入字符信息
        end
        fclose(fid);
    else
        error(['无法打开配置文件：' configfilename '.ini']);
    end
end
end
```

4.5.5 函数的返回参数

正如 4.5.4 小节中函数 getmyParameter 返回输入的信息参数 paras 一样，在设计函数时，可以通过函数的返回值获取函数执行结束后的某些重要信息。从函数的定义形式 function [out1, out2, ...] = myfun(in1, in2, ...)知道，out1、out2 代表返回或输出参数，看下面的示例，计算输入矩阵列方向 X 的总和、均值和标准差，并返回这些值。

文件名：myStat.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function [sumx avgx stdx]=myStat(X) %三个输出参数 sumx avgx stdx 放在[]内
X
n=ones(1,size(X,2))*size(X,1); %利用 size 和 ones 函数构建每列元素个数数组
sumx=sum(X); %计算总和，sumx 作为返回参数
avgx=avg(sumx,n); %利用自定义函数 avg 计算均值，avgx 作为返回参数
stdx=stdev(X,avgx,n);%利用自定义函数 stdev 计算标准差，stdx 作为返回参数
end
function avgx=avg(sumx,n) %自定义函数 avgx 计算均值
```

```
avgx=sumx./n;
end
function stdx=stdev(X,avgx,n) %自定义函数 stdev 计算标准差
avgarray=repmat(avgx,size(X,1),1); %构造均值矩阵，使每列具有相同均值
stdx=sqrt(sum((X-avgarray).^2)./n); %计算标准差
end
```

运行函数 myStat，结果如下：

```
>> myStat([10 10 10 10;5 9 10 12])↙
X =
    10    10    10    10
     5     9    10    12
ans =
    15    19    20    22
```

如直接调用，仅返回第一个输出参数的内容，要想获取多个输出参数内容，则

```
>> [sums avg stds]=myStat([10 10 10 10;5 9 10 12])↙
X =
    10    10    10    10
     5     9    10    12
sums =
    15    19    20    22
avg =
    7.5000    9.5000   10.0000   11.0000
stds =
    2.5000    0.5000         0    1.0000
>> [sums avg stds]=myStat([10 10 10 10;5 9 10 12]')↙
X =
    10     5
    10     9
    10    10
    10    12
sums =
    40    36
avg =
    10     9
stds =
         0    2.5495
```

4.5.6　可变数目的输入/输出参数

当输入参数个数不确定时，可以使用函数 varargin，与输出参数对应的函数是 varargout。MATLAB 将可变的输入/输出参数视作单元数组（因其存储的数据类型可变），如当绘制顶点个数不同的多边形时，事先可能不清楚顶点的个数，使用 varargin 作为函数的参数，就可以方便地达到目的，并且具有很强的灵活性。

文件名：varplot.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function varplot(varargin)
x=zeros(1,length(varargin)); %根据输入参数的个数为变量预分配内存
```

```
y=zeros(1,length(varargin));
for i=1:length(varargin) %循环生成顶点坐标值
    x(i)=varargin{i}(1); %由于输入参数为单元数组，使用{}存储元素
    y(i)=varargin{i}(2);
end
minx=min(0,min(x)); %取包含 0 在内的最小值，供坐标轴使用
miny=min(0,min(y));
plot(x,y); %利用 plot 函数绘制多边形
axis([minx max(x)+1 miny max(y)+1]); %设置坐标轴范围
end
```

在命令窗口中输入以下内容，进行测试。

```
>> varplot([5 3],[4 7],[9 6])↙
```

绘图结果如图 4-5 所示。

```
>> varplot([5 3],[4 7],[9 6],[5 3])↙
```

绘图结果如图 4-6 所示。

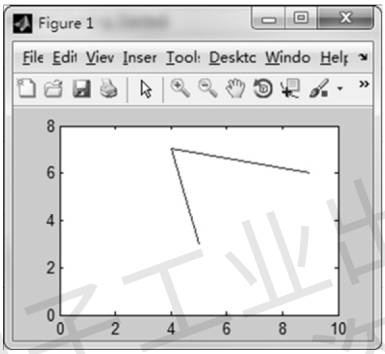

图 4-5 绘图结果 1

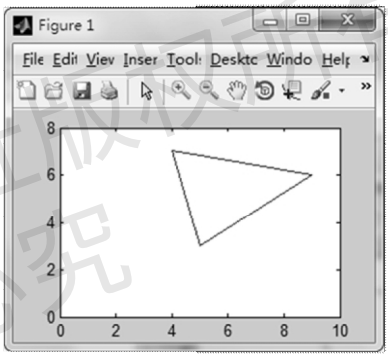

图 4-6 绘图结果 2

下面的示例文件演示了可变输出参数 varargout 函数的应用。

文件名：varout.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function varargout=varout(argin)
if iscell(argin) %如果是单元数组，使用{}索引
    %首先将第 1 个输出参数设置为输入参数的元素个数
    varargout{1}=['输入参数元素个数共' num2str(numel(argin)) '个，分别是：'];
    for i=1:numel(argin) %根据输入参数元素个数多少，变更输出参数数目
        varargout{i+1}=argin{i};
    end
else %否则使用()索引
    varargout{1}=['输入参数元素个数共' num2str(numel(argin)) '个，分别是：'];
    for i=1:numel(argin)
        varargout{i+1}=argin(i);
    end
end
for i=1:numel(varargout)
    disp(varargout{i}); %遍历各个输出参数的内容
end
disp(['输出参数个数为：' num2str(i) '个']); %显示汇总信息，使用类型转换函数
end
```

运行示例如下

```
>> varout([4 5 4 ;9 9 10]);↙
```

输入参数元素个数共 6 个，分别是：

```
     4
     9
     5
     9
     4
    10
```

输出参数个数为 7 个。

4.6　函数的类型

在 4.5.4 小节中的示例文件 playnoisef5.m 中，包含两个函数：一个是 playnoisef5，一个是 getmyParameter。前者与 M 文件的名称相同，称为主函数，后者称为子函数。除此之外，还有私有函数和嵌套函数以及匿名函数，此处主要讲主函数和子函数。

4.6.1　主函数

MATLAB 程序文件（M 文件）中的第一个函数称为主函数，大多数情况下主函数是程序文件中的唯一函数，即如果一个程序文件中只包括一个函数，则该函数就是主函数。在前面的示例文件中，playnoisef.m～playnoisef3.m 文中的函数是唯一的，都是主函数。在 playnoisef4.m 和 playnoisef5.m 中，排在最前面的函数是主函数，即相应的 playnoisef4 和 playnoisef5。一般而言，主函数与 M 文件的名称相同，如果不相同，只能使用文件名来调用函数，而不是关键字 function 后面所起的函数名。

4.6.2　子函数

所谓子函数，是因其无法单独定义，在主函数所在的程序文件中进行定义，且一个程序中可以定义多个子函数，子函数只能供同一程序文件中的主函数或其他子函数调用，而不能被其他程序文件中的主函数或子函数调用，这也是常常将一个函数存储为一个程序文件，即定义为主函数的原因，这样不同的函数之间就可以方便地调用。

4.6.3　函数间的调用关系

尽管有主函数和子函数之分，由于 MATLAB 语言支持递归，故主函数和子函数间的调用关系并没有限制，即主函数可以调用子函数，子函数也可以调用主函数，主函数仅仅是因其排在程序文件中的第一个。看下面的程序示例，函数 afunction 中调用 bfunction，bfunction 函数中调用 cfunction，函数 cfunction 中又调用 afunction，三个函数循环调用（但一般不要将程序设计成循环调用）。

文件名：afunction.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function afunction() %定义函数 afunction
global var
if isempty(var)
   var=1;
else
   var=var+1;
end
```

```
    if var<100
       disp(['In function a:' num2str(var)]); %如果变量 var 值小于 100，则显示信息
       bfunction; %调用函数 bfunction
    end
    end

    function bfunction()
    global var
    if isempty(var)
       var=1;
    else
       var=var+1;
    end
    if var<100
       disp(['In function b:' num2str(var)]);
       cfunction; %调用函数 cfunction
    end
    end

    function cfunction()
    global var
    if isempty(var)
       var=1;
    else
       var=var+1;
    end
    if var<100
       disp(['In function c:' num2str(var)]);
       afunction; %调用函数 afunction
    end
    end
```

运行以上程序，输出结果如下：

```
>> clear all
>> afunction
In function a:1
In function b:2
In function c:3
In function a:4
In function b:5
In function c:6
┆
In function a:97
In function b:98
In function c:99
```

以上三个函数因为 afunction 排在第一个，所以 M 文件名为 afunction.m，即使将程序的文件名改为 bfunction.m，也无法使第二个函数成为主函数，此时在命令窗口中输入 afunction，会出现如下提示信息（未找到函数）。

```
>> afunction
??? Undefined function or variable 'afunction'.
```

改变文件的名称后，如果文件名与主函数名不符，就需要使用文件名来调用函数，但由于函数

afunction 排在第一位，所以当输入 bfunction（文件名）时，会调用 afunction 函数，而非文件中定义的子函数 bfunction，下面验证一下。

```
>> clear all↵
>>bfunction↵
```

```
In function a:1
In function a:2
In function a:3
In function a:4
In function a:5
In function a:6
┆
┆
In function a:97
In function a:98
In function a:99
```

从输出结果可以看出，改名后输入 bfunction 会始终调用 afunction 主函数，尽管在 afunction 中调用了 bfunction，但由于“bfunction”作为了函数 afunction 的文件名，所以相当于自己调用自己，这也是 MATLAB 的强大之处，其他编程语言一般无法实现自己调用自己。例如，下面的程序可以将某个目录下的所有目录（包含子目录）列出。

文件名：listdir.m（文件位于 Psyfeng\Little_Examples 目录下）

```
function listdir(directory)
if nargin<1
   directory='..';
end
s=dir(directory); %获取目录 directory 下内容
for i=1:length(s)
   if s(i).isdir && ~strcmp(s(i).name,'.') && ~strcmp(s(i).name,'..')
      disp(fullfile(directory,s(i).name)); %显示全路径名
      listdir(fullfile(directory,s(i).name)); %如果是目录，则循环调用 listdir 函数
   end
end
end
```

运行示例如下。

```
>> listdir('c:\')↵
```

```
c:\DRIVERS
c:\DRIVERS\WIN
c:\DRIVERS\WIN\4IN1
c:\DRIVERS\WIN\4IN1\MS
c:\DRIVERS\WIN\4IN1\MSx64
c:\DRIVERS\WIN\4IN1\SDMMC
c:\DRIVERS\WIN\4IN1\SDMMCx64
c:\DRIVERS\WIN\4IN1\XD
c:\DRIVERS\WIN\4IN1\xDx64
c:\DRIVERS\WIN\AMTSOL
┆
┆
```

4.6.4 函数的调用顺序

如果不小心定义了与 MATLAB 系统内置函数相同的函数，在调用函数时，到底使用的是哪个函数呢？MATLAB 在查找函数时使用以下规则，首先在当前目录下查找函数，如果没有找到，则按照 MATLAB 路径中设置的先后顺序（通过 path 命令可以列出当前的路径设置情况），所以如果程序没有在当前目录下，并且程序所在的目录也没有加入 MATLAB 路径中，就会出现错误信息（即无法找到指定的函数）。因此，为了确保函数能够顺利运行，一种方式是切换程序所在目录为当前目录，另一种方式是将程序所在目录添加到 MATLAB 路径中（参见 5.6 节）。

可以利用 which 函数明确到底是哪个函数被调用，以及被调用文件所在的目录。

第 5 章　实验设计常用 MATLAB 函数

5.1　矩阵数组操作类函数

5.1.1　数组排序：sort

（1）B = sort(A)

（2）B = sort(A,dim)

（3）B = sort(...,mode)

（4）[B,IX] = sort(A,...)

对向量、矩阵或数组 A 进行排序，并将排序结果存于变量 B 中。如果是矩阵，其中参数 dim 指定是在列方向（1）还是行方向（2）上进行排序（默认为列方向），mode 参数指定排序的升降（ascend=升序，descend=降序），返回值 IX 中储存排序结果 B 中各个元素在排序前 A 中索引位置，并有如下关系：B=A(IX)。

例如：

```
>> a=rand(3)↙
a =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
>> sort(a,2)↙
ans =
    0.2785    0.8147    0.9134
    0.5469    0.6324    0.9058
    0.0975    0.1270    0.9575
>> sort(a,'descend')↙
ans =
    0.9058    0.9134    0.9575
    0.8147    0.6324    0.5469
    0.1270    0.0975    0.2785
>> [b,ix]=sort(a)↙
b =
    0.1270    0.0975    0.2785
    0.8147    0.6324    0.5469
    0.9058    0.9134    0.9575
ix =
     3     3     1
     1     2     2
     2     1     3
```

5.1.2 数组行排序：sortrows

（1）B = sortrows(A)

（2）B = sortrows(A,column)

（3）[B,index] = sortrows(A,...)

按照指定列为关键列进行排序，如指定第 1 列和第 3 列为关键列，则首先以第 1 列的大小排序，当第 1 列的值相同时，则以第 3 列的大小排序，并且当指定的列的序号为正时，按升序排；当指定的列的序号为负时，按降序排。A 为拟排序的数组，column 为关键列向量，排序后的结果存储于变量 B 中，index 为排序结果元素在原数组 A 中的索引，有如下关系：B=A(index)。

例如：

```
>> a=randi([10 100],[4 5])↵
a =
    97    54    93    13    78
    24    82    82    87    77
    98    22    97    94    45
    97    48    69    71    69
>> sortrows(a)↵
ans =
    24    82    82    87    77
    97    48    69    71    69
    97    54    93    13    78
    98    22    97    94    45
>> sortrows(a,[-1 -2])↵
ans =
    98    22    97    94    45
    97    54    93    13    78
    97    48    69    71    69
    24    82    82    87    77
```

5.1.3 矩阵旋转：rot90

（1）B = rot90(A)

（2）B = rot90(A,k)

将矩阵 A 逆时针旋转 90 度，或者逆时针旋转 k×90 度，其中 k 为整数。如果 k 为负数，则顺时针旋转。

例如：

```
>> a=rand(4)↵
a =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>> rot90(a)↵
ans =
    0.9572    0.4854    0.8003    0.1419
    0.9575    0.9649    0.1576    0.9706
    0.6324    0.0975    0.2785    0.5469
```

```
    0.8147    0.9058    0.1270    0.9134
>> rot90(a,2)↙
ans =
    0.1419    0.9706    0.5469    0.9134
    0.8003    0.1576    0.2785    0.1270
    0.4854    0.9649    0.0975    0.9058
    0.9572    0.9575    0.6324    0.8147
>> rot90(a,-1)↙
ans =
    0.9134    0.1270    0.9058    0.8147
    0.5469    0.2785    0.0975    0.6324
    0.9706    0.1576    0.9649    0.9575
    0.1419    0.8003    0.4854    0.9572
```

5.1.4　矩阵左右/上下翻转：fliplr/flipud

（1）B = fliplr(A)

（2）B = flipud(A)

将矩阵左右或上下进行翻转。

例如：

```
>> a=randi([10 20],[3 4])↙
a =
    14    20    19    18
    20    17    20    18
    18    10    17    14
>> fliplr(a)↙
ans =
    18    19    20    14
    18    20    17    20
    14    17    10    18
>> flipud(a)↙
ans =
    18    10    17    14
    20    17    20    18
    14    20    19    18
```

5.1.5　矩阵水平/垂直拼接：horzcat/vertcat

（1）C = horzcat(A1, A2, ...)

（2）C = vertcat(A1, A2, ...)

将矩阵在列方向（水平）或行方向（垂直）上拼接。

例如：

```
>> a=randi([1 5], [2 3])↙
a =
     1     2     2
     1     5     5
>> b=randi([10 50], [2 3])↙
b =
```

```
    19    24    20
    48    18    35
>> c=randi([100 200], [2 3])↵
c =
   147   183   155
   135   159   192
>> horzcat(a,b,c)↵
ans =
     1     2     2    19    24    20   147   183   155
     1     5     5    48    18    35   135   159   192
>> vertcat(a,b,c)↵
ans =
     1     2     2
     1     5     5
    19    24    20
    48    18    35
   147   183   155
   135   159   192
```

5.1.6 数组的重复：repmat

（1）B = repmat(A,m,n)

（2）B = repmat(A,[m n])

（3）B = repmat(A,[m n p...])

将数组 A 的内容在行方向上重复 m 次，在列方向上重复 n 次，还可以构造多维数组，即在第三维上重复 p 次，等等。

例如：

```
>> a=rand(2)↵
a =
    0.7792    0.1299
    0.9340    0.5688
>> repmat(a,2,2)↵
ans =
    0.7792    0.1299    0.7792    0.1299
    0.9340    0.5688    0.9340    0.5688
    0.7792    0.1299    0.7792    0.1299
    0.9340    0.5688    0.9340    0.5688
>> repmat(a,[1,2,2])↵
ans(:,:,1) =
    0.7792    0.1299    0.7792    0.1299
    0.9340    0.5688    0.9340    0.5688
ans(:,:,2) =
    0.7792    0.1299    0.7792    0.1299
    0.9340    0.5688    0.9340    0.5688
```

5.1.7 数组维数变更：reshape

（1）B = reshape(A,m,n)

（2）B = reshape(A,m,n,p,...)

将数组 A 调整为 m 行 n 列，但调整前后元素的个数须相等，即 prod(size(A))=m×n 或 prod(size(A))=m×n×p。

例如：

```
>> a=rand(3,4)↙
a =
    0.4694    0.1622    0.5285    0.2630
    0.0119    0.7943    0.1656    0.6541
    0.3371    0.3112    0.6020    0.6892
>> reshape(a,2,6)↙
ans =
    0.4694    0.3371    0.7943    0.5285    0.6020    0.6541
    0.0119    0.1622    0.3112    0.1656    0.2630    0.6892
```

5.1.8　获取数组维数：size

（1）d = size(X)

（2）[m,n] = size(X)

（3）m = size(X,dim)

获取数组或矩阵 X 的维数，返回值可以为一向量（d）或者分别存储单独的变量（m,n），也可以通过 dim 指定获取的行或列。

例如：

```
>> x=rand(4,6)↙
x =
    0.6551    0.9597    0.7513    0.8909    0.1493    0.8143
    0.1626    0.3404    0.2551    0.9593    0.2575    0.2435
    0.1190    0.5853    0.5060    0.5472    0.8407    0.9293
    0.4984    0.2238    0.6991    0.1386    0.2543    0.3500
>> size(x)↙
ans =
     4     6
>> [m,n]=size(x)↙
m =
     4
n =
     6
>> size(x,1)↙
ans =
     4
```

5.1.9　获取矩阵长度：length

```
numberOfElements = length(array)
```

获取数组维数最大值（行数或列数最大值）。

例如：

```
>> x=rand(3,5,12,8);↙
>> length(x)↙
ans =
    12
```

5.1.10 获取数组元素数：numel

（1）n = numel(A)

（2）n = numel(A, index1, index2, ... indexn)

获取数组（A）包含的元素个数。

例如：

```
>> x=rand(3,5,12,8);
>> numel(x)
ans =
      1440
>> numel(x,1,:)
ans =
     480
```

5.1.11 获取数组的维度数：ndims

```
n = ndims(A)
```

获取数组 A 所包含的维度数，即是二维数组还是三维数组。

例如：

```
>> x=rand(3,5,12,8);
>> ndims(x)
ans =
     4
```

5.1.12 两个常用矩阵：ones/zeros

（1）Y = ones(n)

（2）Y = ones(m,n)

（3）Y = ones([m n])

（4）Y = ones(m,n,p,...)

（5）B = zeros(n)

（6）B = zeros(m,n)

（7）B = zeros([m n])

（8）B = zeros(m,n,p,...)

构建值全为 1 或全为 0 的矩阵，如果指定一个参数，则为方阵，也可以通过参数 m,n,p 等指定多维数组。

例如：

```
>> x=zeros(3)
x =
     0     0     0
     0     0     0
     0     0     0
>> y=ones(3,4)
y =
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

5.1.13　生成等间隔向量：linspace

（1）y = linspace(a,b)

（2）y = linspace(a,b,n)

将[a b]区间进行 n 等分，如果省略参数 n，则进行 100 等分。

例如：

```
>> linspace(3,5,6)↙
ans =
    3.0000    3.4000    3.8000    4.2000    4.6000    5.0000
```

5.1.14　生成网格：meshgrid

（1）[X,Y] = meshgrid(x,y)

（2）[X,Y] = meshgrid(x)

（3）[X,Y,Z] = meshgrid(x,y,z)

生成二维或三维网格。

例如：

```
>> [x y]=meshgrid(-3:3,-3:3)↙
x =
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
    -3  -2  -1   0   1   2   3
y =
    -3  -3  -3  -3  -3  -3  -3
    -2  -2  -2  -2  -2  -2  -2
    -1  -1  -1  -1  -1  -1  -1
     0   0   0   0   0   0   0
     1   1   1   1   1   1   1
     2   2   2   2   2   2   2
     3   3   3   3   3   3   3
```

5.2　判断类函数

5.2.1　是否为空：isempty

```
TF = isempty(A)
```

判断某项内容是否空，所判断的内容可以是字符串型、数值型、单元或结构。如果结果为真，则返回值 TF=1，否则 TF=0。

例如：

```
>> a=[];↙
>> b={};↙
>> c=struct([]);↙
>> d='';↙
```

```
>> isempty(a)↙
ans =
     1
>> isempty(b)↙
ans =
     1
>> isempty(c)↙
ans =
     1
>> isempty(d)↙
ans =
     1
```

参见例 5.1。

5.2.2 是否为列向量：iscolumn

```
TF=iscolumn(V)
```

判断某项内容是否为列向量，如果结果为真，则返回值 TF=1，否则 TF=0。
例如：

```
>> a=rand(3)↙
a =
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
    0.9134    0.2785    0.9649
>> b=rand(1,3)↙
b =
    0.1576    0.9706    0.9572
>> iscolumn(a)↙
ans =
     0
>> iscolumn(b)↙
ans =
     0
>> iscolumn(rot90(b))↙
ans =
     1
```

5.2.3 是否为浮点数：isfloat

```
TF=isfloat(A)
```

判断某项内容是否为浮点数，如果结果为真，则返回值 TF=1，否则 TF=0。
例如：

```
>> a=randi(100)↙
a =
    43
>> b=uint16(randi(50))↙
```