目录

第五章 自定义函数

在第三章中，我们强调了好的编程习惯的重要性。我们进行开发的基本手段是自上而下的编程方法。在自上而下的编程方法中，它开始于对所要解决问题的精确陈述和定义输入量和输出量。下一步，我们在大面上进行算法的描述，然后把算法分解成一个一个的子问题。再然后，程序员把这一个个子问题进行再一次的分解，直到分解成简单而且能够清晰理解的伪代码。最后把伪代码转化为 **MATLAB** 代码。

尽管我们在前面的例子中，按照上面的步骤进行了编程。但是产生的结果在某种程度上还是受限制的。因为我们必须把每一个子问题产生的 **MATLAB** 代码嵌入到一个单独的大程序中。在嵌入之前我们无法对每一次子问题的代码进行独立地验证和测试。

幸运的是，**MATLAB** 有一个专门的机制，在建立最终的程序之前用于独立地开发与调试每一个子程序。每一个子程序都可以独立函数的形式进行编程，在这个程序中，每一个函数都能独立地检测与调试，而不受其他子程序的影响。良好的函数可以大大提高编程的效率。它的好处如下：

1.子程序的独立检测

每一个子程序都可以当作一个独立的单元来编写。在把子程序联合成一个的大程序之前，我们必须检测每一个子程序以保证它运转的正确性。这一步就是我们熟知的单元检测。在最后的程序建立之前，它排除了大量的问题。

2.代码的可复用性

在许多的情况下，一个基本的子程序可应用在程序的许多地方。例如，在一个程序的许多地方，要求对一系列按由低到高的顺序进行排序。你可以编一个函数进行排序，然后当再次需要排序时可以调用这个函数。可重用性代码有两大好处：它大大提高了整体编程效率，它更易于调试，因为上面的排序函数只需要调试一次。

3.远离意外副作用

函数通过输入参数列表（input argument list）从程序中读取输入值，通过输出参数列表（output argument list）给程序返回结果。程序中，只有在输入参数列表中的变量才能被函数利用。函数中，只有输出参数列表中的变量才能被程序利用。这是非常重要的，因为在一个函数中的突发性编程错误只会发生错误的函数的变量。一旦一个大程序编写并发行，它还要面临的问题就是维护。程序的维护包括修补错误，修改程序以适应新或未知的环境。作维护工作的程序员在一般情况下不会是程序的原作者。如果程序编写的不好，改动一处代码就可能对程序全局产生负面影响。这种情况的发生，可能是因为变量在其他部分被重新定义或利用。如果程序员改变这个变量，可能会导致后面的程序无法使用。

好的函数的应用可以通过数据隐藏使问题最小化。在主函数中的变量在函数中是不可见的（除了在输入变量列表中的变量），在主程序中的变量不能被函数任意修改。所以在函数中改变变量或发生错误不会在程序的其他部分发生意外的副作用。

好的编程习惯

把大的程序分解成函数，有很多的好处，例如，程序部分的独立检测，代码的可复用性，避免意想不到的错误。

5.1 MATLAB 函数简介

到目前为止，我们看到的所有的 M 文件都是脚本文件。脚本文件只是用于存储 **MATLAB** 语句。当一个脚本文件被执行时，和直接在命令窗口中直接键入 **MATLAB** 语句

所产生的结果是一样的。脚本文件分享命令窗口中的工作区，所以所有的在脚本文件运行之前定义的变量都可以在脚本文件中运行，所有在脚本文件中创建的变量在脚本文件运行之后仍然存在工作区。一个脚本文件没有输入参数，也不返回结果。但是所有脚本文件可以通过存于工作区中的数据进行交互。

相对地，**MATLAB** 函数是一种特殊形式的 M 文件，它运行在独立的工作区。它通过输入参数列表接受输入数据，它通过输出参数列表返回结果给输出参数列表。**MATLAB** 函数的基本形式如下：

```
function [outarg1, outarg2, ...] = fname(inarg1, inarg2, ...)
%H1 comment line
%Other comment lines
...
(Executable code)
...
(return)
```

function 语句标志着这个函数的开始。它指定了函数的名称和输入输出列表。输入函数列表显示在函数名后面的括号中。输出函数列表显示在等号左边的中括号中。（如果只有一个输出参数，中括号可以省略。）

输入参数列表是名字的列表，这些名字代表从调用者到函数的值。这些名字被称作形参。当函数被调用时，它们只是从调用者得来实际变量的占位符而已。相似地，输出参数列表也形参组成，当函数结束运行时，这些形参是返回到调用者的值的占位符。

在一个表达式中，调用一个函数需要用到实参列表。在命令窗口直接（或在脚本文件中，另一个函数中）键入函数的名字就可以调用这个函数了。当调用一个函数时，第一个实参的值用在第一个形参的位置，而且其余的形参和实参都一一对应。

函数的执行从函数的顶部开始，结束于 return 语句或函数的终点。因为在函数执行到结尾就会结束，所以 return 语句在大部分的程序中没有必要使用。在输出参数列表中每一个项目都必须出现在 function 语句中的左边。当函数返回时，存储于输出函数列表的值就会返回给调用者，用于下一步的运算。

在一个函数中的初始注释行有特定的目的。在 function 语句的第一个行注释被称为 **H1 注释行**。它应当是对本函数功能的总结。这一行的重要性在于，通过 lookfor 命令它能被搜索到并显示出来。从 **H1 注释行**到第一个空行或第一个可执行性语句可以通过 help 命令或帮助窗口搜索到。它们则应包含如何使用这个函数的简单总结。

下面是一个自定义函数的简单例子。函数 dist2 用于计算笛卡尔坐标系中点（x_1，y_1）与点（x_2，y_2）之间的距离。(把以下代码保存成 dist2.m 文件)

```
function distance = dist2 (x1, y1, x2, y2)
%DIST2 Calculate the distance between two point
% Function DIST2 calculates the distance between
% two points (x1, y1) and (x2,y2) in a cartesian
% coordinate system.
%
% Calling sequence:
%     res = dist2(x1, y1, x2, y2)
%
% Define variables:
% x1                --x-position of point 1
% y1                --y-position of point 1
% x2                --x-position of point 2
% y2                --y-position of point 2
% distance          --Distance between points
%
% Record of revisions:
%          Date            Pragrammer          Description of change
%       ========          ==========          =================
%         12/15/98        S.J.Chapman         Original code
%
```

```
% Calculate distance.
distance = sqrt((x2-x1).^2 + (y2-y1).^2);
```

这个函数有 4 个输入参数各和 1 个输出参数。一个简单的利用这个函数的例子显示如下：

```
%  Script file: test_dist2.m
%
%  Purpose:
%           This program test2 function dist2.
%
%  Record of revisions:
%           Date            Pragrammer              Description of change
%        ========           ==========              =================
%           12/15/98        S.J.Chapman             Original code
%
%  Define variables:
%  ax       --x-position of point a
%  ay       --y-position of point a
%  bx       --x-position of point b
%  by       --x-position of point b
%
%  Get input data.
disp('Calculate the distance between two points:');
ax = input ('Enter x value of point a:');
ay = input ('Enter y value of point a:');
bx = input ('Enter x value of point b:');
by = input ('Enter y value of point b:');

%  Evaluate function
result = dist2 (ax, ay, bx, by);
%  Write out result.
fprintf('The distance between points a and b is %f \n', result);
```

当脚本文件被执行时，它的结果显示如下：

```
>> test_dist2
Calculate the distance between two points:
Enter x value of point a:1
Enter y value of point a:1
Enter x value of point b:4
Enter y value of point b:5
The distance between points a and b is 5.000000
```

通过手动运算我们可知程序运算的结果是正确的。

函数 dist2 也支持 **MATLAB** 帮助子系统。如果你键入“help dist2”，将会得到的结果是：

```
>> help dist2
  DIST2 Calculate the distance between two point
    Function DIST2 calculates the distance between
    two points (x1, y1) and (x2,y2) in a cartesian
    coordinate system.

    Calling sequence:
       res = dist2(x1, y1, x2, y2)

    Define variables:
    x1                  --x-position of point 1
    y1                  --y-position of point 1
```

```
x2                --x-position of point 2
y2                --y-position of point 2
distance  --Distance between points

Record of revisions:
      Date            Pragrammer          Description of change
   ========        ==========          ================
     12/15/98        S.J.Chapman         Original code

Calculate distance.
```

相似地，键入“lookfor dist2”后将会产生如下的结果：

```
>> lookfor dist2
DIST2 Calculate the distance between two point
test_dist2.m: %       Script file: test_dist2.m
>> lookfor distance
DIST2 Calculate the distance between two point
```

为了仔细观察工作区在函数执行前后的变化，我们将在 **MATLAB** 调试器中加载函数 dist2 和脚本文件 test_dist2。在函数加载前，加载中，加载后设置断点（如图 5.1 所示）。

当程序中止在函数调用之前的断点，它的工作区如图 5.2(a)所示。注意工作区中只有变量 ax，ay，bx 和 by。当程序中止在函数调用过程中的断点，它的工作区如图 5.2(b)所示。注意工作区中只有变量 x1，x2，y1，y2 和 distance。当程序中止在函数调用后的断点，它的工作区如图 5.2(c)所示。注意工作区中原来的变量又重复出现，再加上函数返回的变量 result。这些图显示了 **MATLAB** 调用 M 文件的过程中工作区的变化。

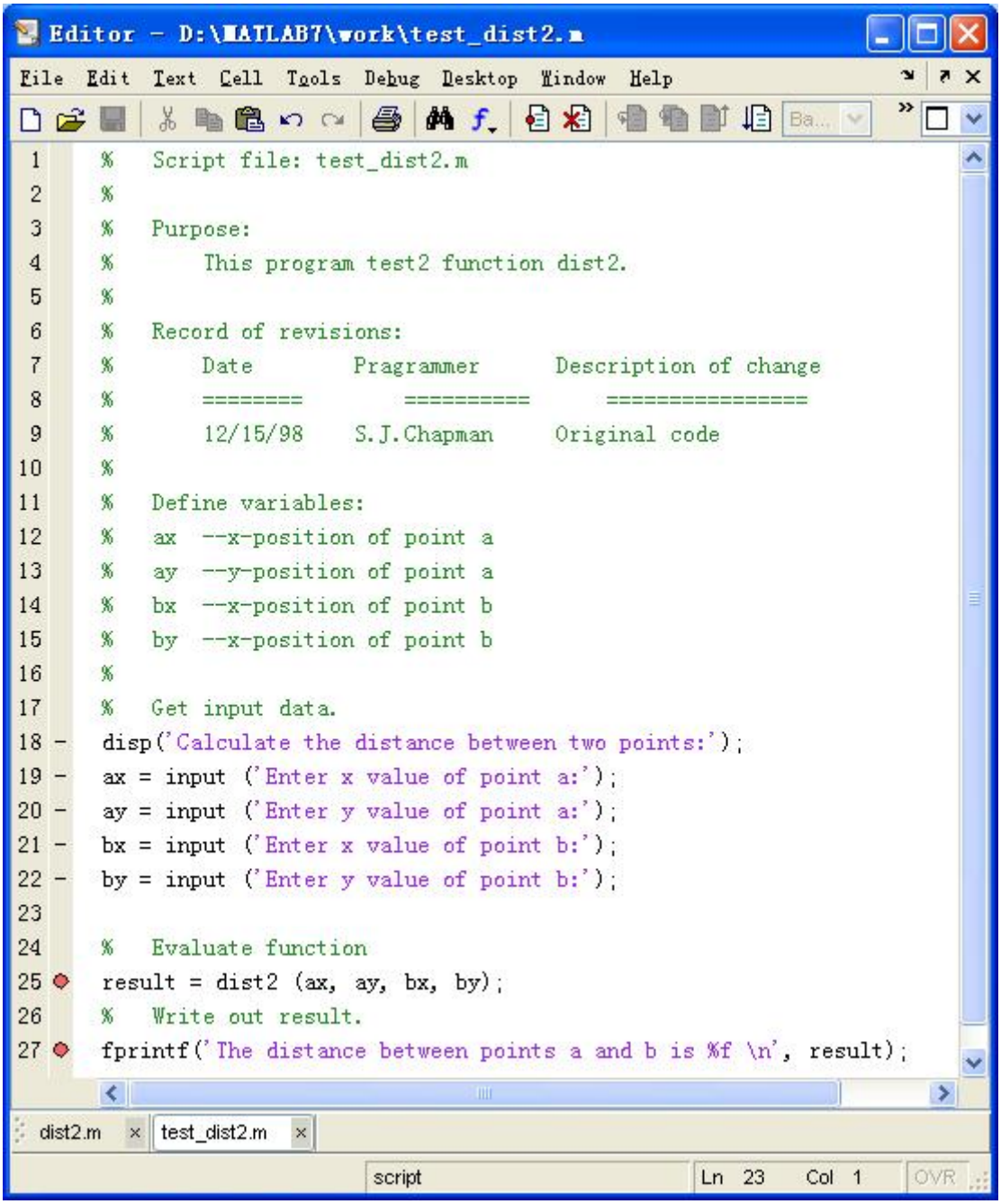


图 5.1 M 文件和函数 dist2 将会被加载到调试器，在函数调用前，调用过程中，调用后设置合适断点

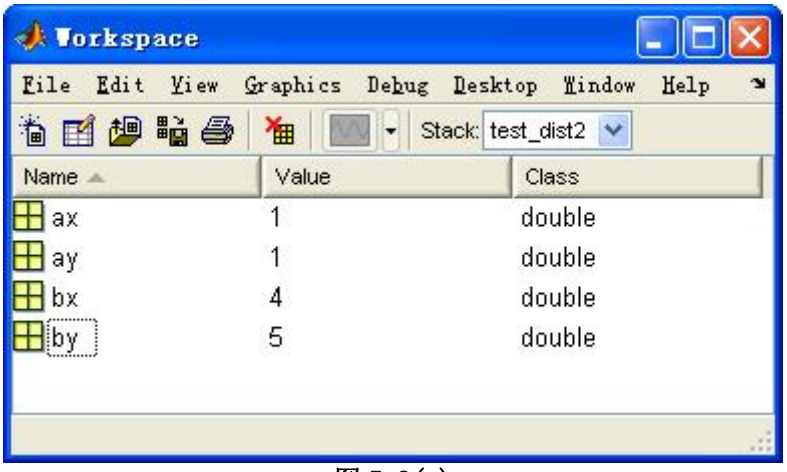


图 5.2(a)

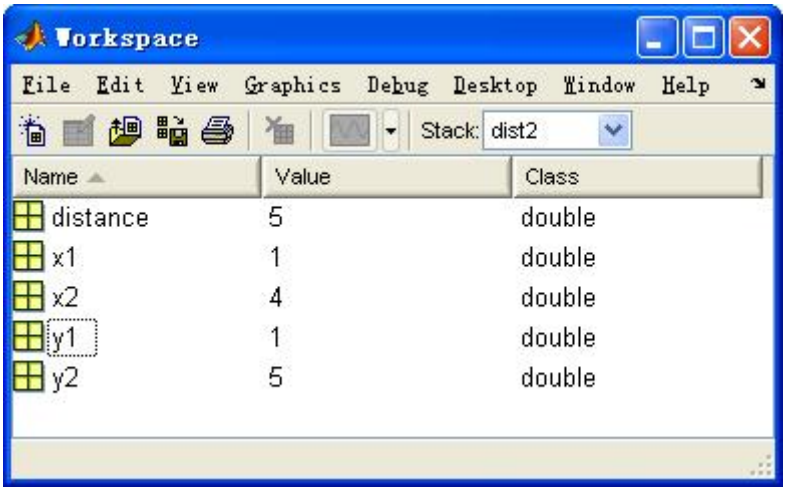


图 5.2(b)

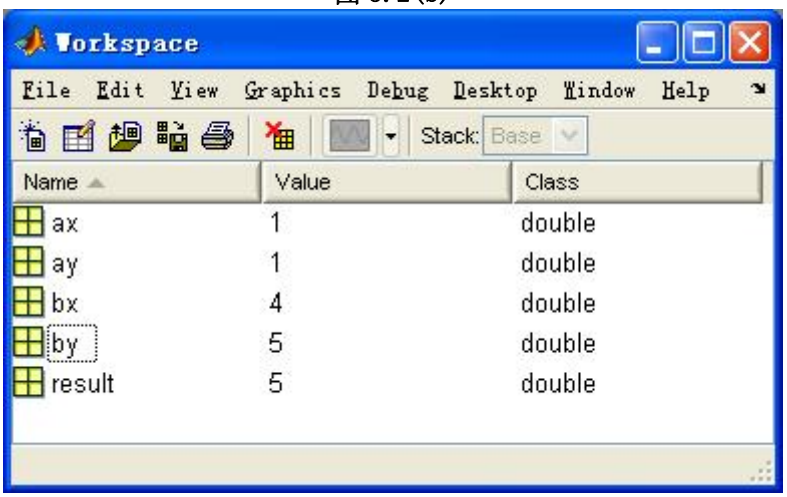


图 5.2(c)

图 5.2(a)在函数调用之前的工作区（b）函数调用过程中的工作区（c）函数调用之后的工作区

5.2 在 MATLAB 中传递变量：按值传递机制

maltab 程序与它们函数之间的交互用是按值传递机制。当一个函数调用发生时，**MATLAB** 将会复制实参生成一个副本，然后把它们传递给函数。这次复制是非常重要的，因为它意味着虽然函数修改了输入参数，但它并没有影响到调用者的原值。这个特性防止

了因函数修改变量而导致的意想不到的严重错误。

这一特性将在下面的函数中得到说明。这个函数中有两个输入参数：a 和 b。在它的计算中，它修改了变量的值：

```
function out = sample(a, b)
fprintf('In Sample: a = %f, b = %f %f\n',a,b);
a = b(1) + 2*a;
b = a .* b;
out = a + b(1);
fprintf('In Sample: a = %f, b = %f %f \n',a,b);
```

下面是调用这个函数的检测程序：

```
a = 2; b = [6 4];
fprintf('Before sample: a = %f, b = %f %f\n', a, b);
out = sample(a, b);
fprintf('After sample: a = %f, b = %f %f\n',a,b);
fprintf('After sample: out = %f \n', out);
```

当这个程序被执行将产生如下的结果：

```
>> test_sample
Before sample: a = 2.000000, b = 6.000000 4.000000
In Sample: a = 2.000000, b = 6.000000 4.000000
In Sample: a = 10.000000, b = 60.000000 40.000000
After sample: a = 2.000000, b = 6.000000 4.000000
After sample: out = 70.000000
```

注意，a 和 b 在函数 sample 内都改变了，但这些改变对调用函数中的值并没有任何的影响。

C 语言的使用者对按值传递机制比较熟悉，因为 C 应用它把标量值传递给函数。尽管 C 语言不能用按值传递机制传递数组，所以对在 C 语言函数中的形参数组进行意想不到的修改将会导致在调用程序时产生错误。**MATLAB** 改进了按值传递机制，既适于标量，又适应于数组（在 **MATLAB** 中参数传递过程中的执行要远比上面讨论中指出的要复杂的多。正如上面指出的，与按值传递相联系的复制将花去大量的时间，但是保护程序以至于不产生错误。实际上，**MATLAB** 用了两种方法，它先对函数的每一个参数进行分析，确定函数的那些参数进行了修改。如果函数没有修改这个参数，它将不会对此参数进行复制，而是简单地指向程序外面的外面的变量，如果函数修改了这个参数，那么这个复制就会被执行）。

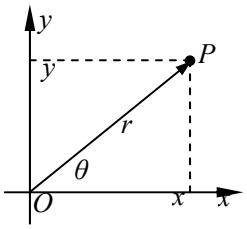


图 5.3 在笛卡尔平面内的一点 P，既可以用直角坐标系来描述，又可以有极坐标来描述

例 5.3

直角坐标与极坐标的转换

在笛卡尔平面上的一点的坐标既可以通过直角坐标（x，y）来描述，也可以通过极坐标（r，θ）来描述，如图 5.3 所示。两套坐标体系的关系如下式所示：

$$x = r\cos\theta \qquad (5.1)$$

$$y = r\sin\theta \qquad (5.2)$$

$$r = \sqrt{x^2 + y^2} \qquad (5.3)$$

$$\theta = \tan^{-1}\frac{y}{x} \qquad (5.4)$$

编写两个函数 rect2polar 和 polar2rect，用来实现两坐标体系的转换。其中 θ 单位于为度。

答案：

我们现在应用标准的问题解决方法来创建函数。注意 matab 的 trigonometric(三角)函数的参数的单位是弧度。所以在解决这个问题时，我们必须把度转化为弧度，反之亦然。基本的度与弧度的转换关系如下：

$$180^{o} = \pi \text{ radians} \qquad (5.5)$$

1.陈述问题对这个问题的简单陈述为：

编写一个函数，把直角坐标系描的笛卡尔平面内的一个坐标转化对应的极坐标，角 θ 的单位为度。反之，把极坐标系内的一个坐标转化对直角坐标。角 θ 的单位也为度。

2.定义输入输出量

函数 rect2polar 的输入量是直角坐标系（x，y）中的一个点。这个函数的输出量是极坐标（r，θ）中的一个点。函数 polar2rect 的输入量是极坐标（r，θ）中的一个点。此函数的输出量是直角坐标系（x，y）中的一个点。

3.定义算法

这些函数是非常简单的，所以我们能直接的写出它们的伪代码。函数 polar2rect 的伪代码如下：

```
x ← r * cos(theta * pi/180)
y ← r * sin(theta * pi/180)
```

函数 rect2polar 的伪代码将会用到函数 atan2，因为函数的取值范围覆盖了笛卡尔平面的所有象限。（通过 **MATLAB** 的帮助系统查找这个函数。）

4.把算法转化为 **MATLAB** 语句

函数 polar2rect 的 **MATLAB** 代码如下所示

```
function [x, y] = polar2rect(r, theta)
%POLAR2RECT Convert rectangular to polar coordinates
%  Function POLAR2RECT accepts the polar coordinates
%  (r, theta), where theta is expressed in degrees,
% and converts them into the rectangular coordinates (x, y)
%
%  Calling sequence:
%            [x, y] = polar2rect(r, theta)

%  Define variables:
%  r                    --Length of polar vector
%  theta                --Angle of vector in degrees
%  x                    --x-position of point
%  y                    --y-position of point

%  Record of revisions:
%            Date            Programmer          Description of change
%            ======          ==========          =====================
%            09/19/00        S.J.Chapman         Original code
x = r * cos(theta * pi/180);
y = r * sin(theta * pi/180);
```

函数 rect2polar 的 **MATLAB** 代码如下所示

```
function [r, theta] = rect2polar(x, y)
%RECT2POLAR Convert rectangular to polar coordinates
%  Function RECT2POLAR accept the rectangular coordinates
%  (x, y) and converts them into the polar coordinates
%  (r, theta), where theta is expressed in degrees.
%
```

```
%  Calling sequence:
%              [r, theta] = rect2polar(x, y)
%
%  Define variables:
%  r                          --Length of polar vector
%  theta                      --Angle of vector in degrees
%  x                          --x-position of point
%  y                          --y-position of point
%
%  Record of revisions:
%              Date           Programmer          Descriptoin of change
%              ====           ==========          =====================
%              09/19/00       S.J.Chapman         Original code

r = sqrt( x .^2 + y .^2);
theta = 180/pi * atan2(y, x);
```

注意这两个函数中都包含了帮助信息，所以在应用 **MATLAB** 的帮助子系统中，或使用 lookfor 命令中它们将正常的运作。

5.检测程序

为了检测这些程序，我们将在 **MATLAB** 命令窗口中直接运行它们。我们将用边长分别为 3，4，5 的三角进行检测，这个三角在初中我们就非常的熟悉了。在这个三角形中最小的角约为 36。87 度。我们将在四个象限对函数进行检测，以保证在任何情况下转换都是正确的。

```
>> [r, theta]=rect2polar(4,3)
r =
     5
theta =
    36.8699
>> [r, theta]=rect2polar(-4,3)
r =
     5
theta =
   143.1301
>> [r, theta]=rect2polar(-4,-3)
r =
     5
theta =
  -143.1301
>> [r, theta]=rect2polar(4,-3)
r =
     5
theta =
   -36.8699

>> [x, y]= polar2rect(5,36.8699)
x =
    4.0000
y =
    3.0000
>> [x, y]= polar2rect(5,-143.1301)
x =
   -4.0000
y =
   -3.0000
>> [x, y]= polar2rect(5,143.1301)
```

```
x =
    -4.0000
y =
     3.0000
>> [x, y]= polar2rect(5,-36.8699)
x =
     4.0000
y =
    -3.0000
```

在笛卡尔坐标的四个象限内得到的结果均是正确的。

例 5.2

数据排序在许多的科研和工程应用中，随机输入一组数据并对它进行由低到高排序或由高到低进行排序是十分必要的。假设你是一个动物学家，你正在研究大量的动物，并想要鉴定这些动物最大的 5%。

解决这个问题的最直接的方法是对这些动物的大小按照降序进行排列，取其前 5%。对数据进行升序或降序排列似乎是一件非常容易的工作。毕竟，我们经常作这样的事。有一个简单的例子，把数据（10，3，6，4，9）按升序排列成（3，4，6，9，10）。我们应当怎样做呢。我们首先应当浏览整个输入数据列表（10，3，6，4，9）找出其中的最小值（3），然后浏览剩下的输入数据（10，6，4，9）并找到下一个最小值（4），然后继续重复上面的步骤，直到所有的列表中的所有数都能排完。

实际上，排序是一个非常困难的工作。当值的个数增加时，用上面简单的排序方法进行运算所消耗的时间将会迅速增加，因为每排一个数就要浏览一个遍输入值。对于大的数据集合，这个方法因太耗时，而不用。更糟糕的是，如果有大量的数据占有计算机的大部分内存我们将如何排序。开发大数据集合的高效排序技术是一个相当活跃的研究领域，它已经成为了一个新的科目。

在这个例子中，我们将尽可能简单的算法来说明排序的内容。这个最简单的算法叫做选择性排序（selection sort）。它只是对应上面描述方法的计算机执行。选择性排序的基本算法如下：

1.浏览被排序数的列表，并找出其中的最小值。把最小值与排在最前面的数进行交换。如要排在最前面的数就是这个数表最小值，什么也不用做。

2.从这个数据列表的第二个数开始浏览找到第二个最小的数。把这个数与当前排在第二个数进行交换。如果当前排在第二位的数就是下一个最小值，那么什么也不用做。

3.从数据列表的第三个数开始找到第三个最小的数。把这个数与当前排在第三个数进行交换。如果当前排在第三位的数就是第三个最小值，那么什么也不用做。

4.重复以上步骤直至最后一位置排完。当最后一个位置排完后，排序结束。

注意:如果我们要对 N 个数进行排序，这个排序算法要进行 N-1 次浏览才能完成排序。

这个步骤的说明如图 5.4 所示。因为有 5 个数进行排序，所以要对数据进行 4 次浏览。首先对整个数据列表进行浏览，得到最小值 3，把 3 置于第一位，故与 10 进行交换。从第二位开始浏览，得到第二个最小值 4，与 10 交换。从第三位进行浏览，得到最小值 6，6 恰在第三位上，不用交换。从第四位开始浏览，得到最小值 9，与排在第 4 位的 10 交换。排序结束。

性能提示

选择性编程算法是一种极易理解的编程算法，但它的效率是极低的。我们绝不能用它进行大数据集合的排序（例如含有 1000 个元素的数组）。这个几年里，计算机专家已经发展了许多高效的排序算法。内置于 MATLAB 的 sort 和 sortrows 函数是非常高效的，在实际工作中我们应当应用这些函数。

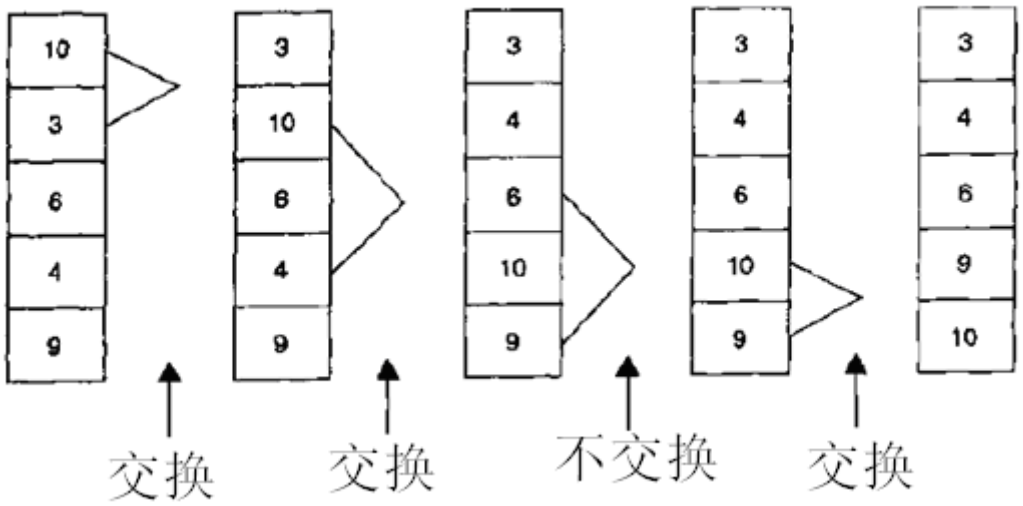


图 5.4 选择性排序的一个简单例子。

我们将开发一个程序，读取从命令窗口读取一个数据集，对它进行升序排列，并出排序后的结果。这个排序将会由独立的自定义函数来完成。

答案：

这个程序必须提示使用者提供输入数据，对其进行排序，并输出排序结要。这个程序的设计过程如下：

1.陈述问题我们刚才没有指定要排序的数据类型。如果数据是数字，那么问题的陈述如下。开发一个程序，它能够读取在命令窗口中输入的任意类型的数字，用独立的自定义函数对读取的值进行排序，并在命令窗口写出排序结果。

2.定义输入输出量这个程序的输入值是在命令窗口键入的数字值。这个程序的输出量是写在命令窗口中的排序结果。

3.设计算法这个问题可以分解为三大步：

```
Read the input data into an array
Sort the data in ascending order
Write the sorted data
```

第一大步是读取数据。我们必须提示使用者输入输入数据的个数，然后读取数据。因为我们知道所要读取的数的确切个数，所以可以用 for 循环主读取合适的数据。它的伪代码如下：

```
Prompt user for the number of data values
Read the number of data values
Preallocate an input array
for ii = 1:number of values
    Prompt for next value
    Read value
end
```

下一步，我们必须要用独立的函数对数据进行排序。我们需要对数据进行 naval-1 次浏览，每一次找出一个最小值。我们将用一个指针来寻找每一次浏览的最小值。一量最小值被找到，如果它不在列表的顶端，它就与列表顶端的元素进行交换。伪代码如下：

```
for ii = 1:nvals -1
% Find the minimum value in a(ii) through a(nvals)
      iptr ← ii
      for jj = ii +1 to nvals
            if a(jj) < a(iptr)
                  iptr ← a(iptr)
            end
      end
```

```
% iptr now points to the min value, so swap a(iptr)
% with a(ii) if iptr ~= ii.
      if ii ~= iptr
            temp ← a(ii)
            a(ii) ← a(iptr)
            a(iptr) ← temp
      end
end
```

最后一步是输出排序结果。这个步骤的伪代码不需要重复。最终的伪代码是这三大步伪代码的联合。

4.把伪代码转化为 **MATLAB** 语言选择性排序的 **MATLAB** 代码如下所示：

```
function out = ssort(a)
%SSORT Selection sort data in ascending order
% Function SSORT sorts a numeric data set into
% ascending order. Note that the selection sort
% is relatively inefficient. DO NOT USE THIS
% FUNCTION FOR LARGE DATA SETS. Use MATLAB's
% "sort" function instead.
% Define variables:
% a              --Input array to sort
% ii             --Index variable
% iptr           --Pointer to min value
% jj             --Index variable
% nvals          --Number of values in "a"
% out            --Sorted output array
% temp           --Temp variable for swapping
% Record of revisions:
% Date           Programmer          Description of change
% ====           ==========          =====================
% 12/19/98    S. J. Chapman      Original code
% Get the length of the array to sort
nvals = size(a,2);
% Sort the input array
for ii = 1:nvals-1
% Find the minimum value in a(ii) through a(n)
      iptr = ii;
      for jj = ii+1:nvals
            if a(jj) < a(iptr)
                  iptr = jj;
            end
      end
% iptr now points to the minimum value, so swap a(iptr)
% with a(ii) if ii ~= iptr.
      if ii ~= iptr
            temp = a(ii);
            a(ii) = a(iptr);
            a(iptr) = temp;
      end
end
% Pass data back to caller
out = a;
```

调用选择性排序函数的程序如下：

```
% Script file: test_ssort.m
%
% Purpose:
```

```
% To read in an input data set, sort it into ascending
% order using the selection sort algorithm, and to
% write the sorted data to the Command window. This
% program calls function "ssort" to do the actual
% sorting.
%
% Record of revisions:
% Date          Programmer          Description of change
% ====          ==========          =====================
% 12/19/98    S. J. Chapman      Original code
%
% Define variables:
% array              --Input data array
% ii                 --Index variable
% nvals              --Numberof input values
% sorted             --Sorted data array
% Prompt for the number of values in the data set
nvals = input('Enter number of values to sort: ');
% Preallocate array
array = zeros(1,nvals);
% Get input values
for ii = 1:nvals
    %Prompt for next value
    string = ['Enter value ' int2str(ii) ': '];
    array(ii)=input(string);
end
% Now sort the data
sorted = ssort(array);
% Display the sorted result.
fprintf('\nSorted data:\n');
for ii = 1:nvals
    fprintf(' %8.4f \n',sorted(ii));
end
```

5.检测程序

为了检测这个程序，我们应当创建一个输入数据集，并运行这个程序。这个数据集应当包含由正负数混合组成，还有至少包括一个完全一样的值，以观察在这些情况下程序是否正常工作。

```
>> test_ssort
Enter number of values to sort: 6
Enter value 1: -5
Enter value 2: 4
Enter value 3: -2
Enter value 4: 3
Enter value 5: -2
Enter value 6: 0

Sorted data:
   -5.0000
   -2.0000
   -2.0000
    0.0000
    3.0000
    4.0000
```

对于我们检测的数据，程序给出了正确的结果。注意从正数到负数，还有重复值，这个程序工作正常。

5.3 选择性参数

许多的 **MATLAB** 函数都支持选择性输入参数和输出参数。例如，我们调用 plot 函数，输入参数既可以少到 2 个，也可以多到 7 个参数。从另一方面说，函数 max 既支持一个输出参数，也支持两个输出参数。如果只有一个输出参数，max 将会返回函数的最大值。如果有两个输出参数将会返回数组的最大值和最大值所在的位置。如何知道一个 **MATLAB** 函数有几个输入输出参数呢，以及函数相应的功能呢？

在 **MATLAB** 中有八种专门的函数用于获取关于选择性参数的信息和用于报告这些参数的错误。其中的六个函数我们在这里介绍，其余的两个我们将会在第七章讲单元数据类型时介绍。

nargin	这个函数返回调用这个函数时所需要的实际输入参数的个数
nargout	这个函数返回调用这个函数时所需要的实际输出参数的个数
nargchk	如要一个函数调用被调用时参数过多或过少，那么 nargchk 函数将返回一个标准错误信息
error	显示错误信息，并中止函数以免它产生这个错误。如果参数错误是致命的，这个函数将会被调用。
warning	显示警告信息并继续执行函数，如果参数错误不是致命的，执行还能继续，则这个将会被调用。
inputname	这个函数将会返回对于特定参数个数的实际变量名。

函数 nargin 和 nargout 只用在用户自定义函数中。当他们被调用时，这些函数将会分别返回实际输入、输出参数的个数。如果一个函数在被调用时含有过多或过少的参数，函数 nargchk 将会产生一个包含标准错误的字符串。此函数的语法如下：

```
message = nargchk(min_args, max_args, num_args);
```

其中 min_args 是指参数的最小个数，max_args 是指数的最大个数，num_args 是指参数的实际个数。如果参数的个数不在允许的范围，将会产生一个标准的错误信息。如果参数的个数在允许的范围之内，那么这个函数将返回一个空字符。

函数 error 是用于显示标准的错误信息和用于中止导致错误信息的自定义函数的一种标准方式。这个函数的语法是 error('msg')，其中 msg 是一个包含错误信息的字符串。当 error 函数执行，它将会中止当前函数，并返回到键盘输入状态，在命令窗中显示出错误信息。如果这个信息字符中为空，error 函数将什么也不做，当前函数继续执行。如果当前函数与线程数 nargchk 工作良好，当有错误发生时，error 将产生一个信息字符串，当没有错误时，error 将产生一个空字符。

函数 warning 是用于显示函数或线程数中的警告信息的一种标准方法。此函数的语法为 warning('msg')，其中 msg 是指含有警告信息的字符串。当执行 waring 函数时，它将在命令窗口显示警告信息，和列出警告出现的函数和线程数。如果信息子符串为空，warning 将什么也不做。在其他情况下，函数将继续执行。

当一个函数被调用时，inputname 函数将会返回实参的名字。inputname 函数的语法为

```
name = inputname(argno);
```

其中 argno 是参安息的个数。如果这个参数是一个变量，那么返回将只是变量名。如果参数是一个表达式，那么这个函数将会返回空字符。例如考虑下面的函数

```
function myfun(x, y, z)
name = inputname(2);
disp(['The second argument is named ' name]);
```

当这个函数被调用时，结果如下

```
>> myfun(dog,cat)
The second argument is named cat
>>myfun(1,2+cat)
The second argument is named
```

函数 inputname 用来显示警告或错误信息中的参数名非常有用。

例 5.3

选择性参数的应用

通过创建函数把直角坐标值（x，y）转化相应的极坐标值，我们向大家说选择性参数的应用。这个函数支持两个输入参数，x 和 y。但是，如果支持只有一个参数的情况，那么函数就假设 y 值为 0，并使用它进行运算。函数在一般情况下输出量为模与相角（单位为度）。但只有一个输出参数只有一个时，它只返回模。函数如下所示。

```
function [mag, angle] = polar_value(x, y)
%  POLAR_VALUE Converts(x, y) to (r, theta)
%  Punction POLAR_VALUE converts an input(x,y)
%  va1ue into (r, theta), with theta in degrees.
%  It illustrates the use of optional arguments.
%  Define variables:
%     angle                --Angle in degrees
%     msg                   --Error message
%     mag                   --Magnitude
%     x                     --Input x value
%     y                     --Input y value(optional)
%     Record Of revisions:
%     Date       Programmer          Description of change
% ======== ============== ========================
% 12/16/98 S.J.Chapman          Original code
% Check for a legal number of input arguments
msg = nargchk(1,2,nargin);
error(msg);
% If the y argument is missing, set it to 0.
if nargin < 2
      y = 0;
end
% Check for (0,0) input argument, and print out
% a warning message.
if x == 0 & y == 0
      msg = 'Both x and y are zero: angle is meaningless!';
      warning(msg);
end
% Now calculate the magnitude
mag = sqrt(x .^2 + y .^2);
% If the second output argument is present,calculate
% angle in degrees
if nargout == 2
      angle = atan2(y,x) * 180/pi;
end
```

我们通过在命令窗口反复调用这个函数来检测它。首先，我们用过多或过少的参数来调用这个函数。

```
>> [mag angle]=polar_value
??? Error using ==> polar_value
Not enough input arguments.

>> [mag angle]=polar_value(1,-1,1)
??? Error using ==> polar_value
Too many input arguments.
```

在两种情况下均产生了相应的错误信息。我们将用一个参数或两个参数调用这个函数。

```
>> [mag angle]=polar_value(1)
mag =
      1
angle =
      0
>> [mag angle]=polar_value(1,-1)
```

```
mag =
      1.4142
angle =
    -45
```

在这两种情况下均产生了正确的结果。我们调用这个函数使之输出有一个或两个参数。

```
>> mag = polar_value(1,-1)
mag =
      1.4142
>> [mag angle]=polar_value(1,-1)
mag =
      1.4142
angle =
    -45
```

这个函数提供了正确的结果。最后当 x=0，y=0 时，调用这个函数。

```
>> [mag angle] = polar_value(0,0)
Warning: Both x and y are zero: angle is meaningless!
> In polar_value at 27
mag =
       0
angle =
       0
```

在这种情况下，函数显示了警告信息，但执行继续。

注意一个 **MATLAB** 函数将会被声明有多个输出函数，超出了实际所需要的，这是一种错误。事实上，函数没有必要调用函数 nargout 来决定是否有一个输出参数存在。例如，考虑下面的函数。

```
function [z1, z2] = junk(x, y)
z1 = x +   y;
z2 = x - y;
```

这个函数输出可以有一个或两个输出参数。

```
>> a = junk(2,1)
a =
       3
>> [a b] = junk(2,1)
a =
       3
b =
       1
```

在一个函数中检查 nargout 的原因是为了防止无用的工作。如果我们找不到输出结果，为什么不在第一位置计算出来？程序员可以不必为无用的运算耐恼，也能加速程序的运算。

测试 5.1

本测试提供了一个快速的检查方式，看你是否掌握了 5.1 到 5.3 的基本内容。如果你对本测试有疑问，你可以重读 5.1 到 5.3，问你的老师，或和同学们一起讨论。在附录 B 中可以找到本测试的答案。

1. 脚本文件与函数的区别是什么？
2. 自定义函数的 help 命令是如何工作的？
3. 函数中的 H1 注释行有什么重要性？
4. 什么是按值传递机制？它对结构化编程有什么好处。

5. 如何使 **MATLAB** 函数带有选择性参数。

第 6，7 题中，请你确定函数的调用是否正确。如果它是错误的，指出错误所在。

6.

```
out = test1(6);

function res = test1(x, y)
res = sqrt(x .^2 + y .^2);
```

7.

```
out = test2(12);

function res = test2(x, y)
error (nargchk(1,2,nargin));
if nargin == 2
    res = sqrt(x .^2 + y .^2);
else
    res = x;
end
```

5.4 用全局内存分享数据

我们已经看到了，函数与程序之间交换数据是通过参数列表来完成的。当一个函数被调用时，每一个实参都会被复制，而这个复制量将会在函数中用到。

对于参数列表还有一些补充，**MATLAB** 函数与每一个参数或基本工作区通过全局内存交换数据。全局内存是指内存的一种特殊类型，它能够被所有的工作区访问。如果一个变量在函数中被声明全局变量，那

么它将占用的是全局内存，而不是本地工作区。如果相同的变量在另一个函数中被声明为全局变量，那么这个变量所占有内存区域就是第一个函数中的相同变量。声明有全局变量的脚本文件或函数将有办法访问相同的值，所以全局变量为函数之间分享数据提供了一个方法。

全局变量的声明要用到 global 主语句，基本形式如下

```
global var1 var2 var3 ...
```

其中 var1，var2，var3 等等是用全局内存的变量。为了方便，全局变量将在函数开头被声明，但是实际上没有这个必要。

好的编程习惯

最是把全局变量声明在函数的开头，这样可以区别于本地变量。

每一个全局变量在函数第一次使用之前必须声明如果在本地工作区中已经被创建，那么声明为再次声明全局变量将会产生错误。为了避免这种错误，在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量。

好的编程习惯

在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量

全局变量尤其适用于在许多函数分享大容量数据，这样全部的数据在每一次被函数调用时就不必再复制了，用全局变量在函数之间交换数据的不利一面为函数只能为特定的数据工作。通过输入数据参数交换数据的函数能用不同的参数调用它，而用全局变量进行数据交换的函数必须进行修改，以许它和不同的数据进行工作。

好的编程习惯

在一个程序，你能利用全局内存，在函数之间对大规模数据进行交换。

例 5.4

随机数发生器对真实世界进行精确度量是十分重要的。对于每一个测量值来说，都有一定的测量噪声（误差）。对于系统的设计来说，这个情况必须要重要考虑，例如，真实世界中机械装置（飞机等）的运转。好的工程设计必须把他的测量误差控制在一定的范围之内，不能因误差导致系统的不稳定。

在系统建立之前，许多的工程设计的检测是通过系统操作的模拟（simulation）来完成的。模拟包括系统动作的数学模型和符合这个模型的输入数据。如果这个模型对**模拟输入数据**反应正确，那么我们就能合理的证明，真实世界中的系统对真实世界中输入值有正确的反应。

提供给数学模型的**模拟输入数据**必须带有**模拟测量噪声**。模拟测量噪声是指加入理想输入值中的一系列随机数。模拟噪声一般由**随机数发生器**产生。

随机数发生器是一个函数，当它每一次被调用时，将会返回一个不同的随机出现的数。事实上，这些数是由一个**确定性算法**产生的，它们只是表现为随机。但是，如果产生它们的算法足够复杂，那么应用于模拟中的这些数就足够地随机。

下面是一个简单随机数发生器的算法。它是利用大数求余的不可预知性。考虑下面的等式。

$$n_{i+1} = mod(8121n_i + 28411, 134456) \qquad (5.6)$$

假设 n_i 为非负整数，那么由于求余函数的关系，n_{i+1} 只能在 0 到 13445 之间的整数中进行取值。重复以上过程，得到的结果永远是在区间[0, 134455]中。如果我们事先不知道 8121，28411 和 134456 这三个数你很可能猜测这个顺序是由 n 值产生的。进一步说，它说明，所有在 0 到 13445 之间的整数出现的次序是等可能性。由于这些属性，等式（5.6）可以当一个简单的随机数发生器的基础。

现在我们用公式（5.6）设计一个随机数发生器，它的输出是一个实数，其取值范围这[0.0, 1.0]。

答案

我们要编写一个函数，在每一次调用时，它能产生 0≤ran<1.0 的随机数。随机数的产生将依赖于下面的公式。

$$ran_i = \frac{n_i}{1334456} \qquad (5.7)$$

通过公式 5.6n_i，在 0 到 134455 之间进行取值。公式 5.6，5.7 中产生的随机数的顺序取决于 n_0 的初始值(种子，seed)。我们要为用户提供一种途径，让它用于指定 n_0，这样每次运行这个函数得到的随机数顺序都是不一样的。

1.陈述问题

编写一个函数 random0，使之产生一个数组，数组中包括一个或多个随机数，它的取值范围是 0≤ran<1.0，它的顺序由公式 5.6 和 5.7 指定。函数应当有一个或多个输入参数(n 和 m)，用来指定返回数组的大小。如果它有一个参数，函数将产生一个 n 阶方阵;如果有两个参数，函数将会产生一个 $n \times m$ 的数组。种子 n_0 的初始值将会由函数 seed 指定。

2.定义输入量和输出量

在这个问题中共有两个函数:seed 和 random0。函数 seed 的输入是一个整数。这个函数没有输出。random0 的输入量是一个或两个整数。如果它有一个参数，函数将产生一个 n 阶方阵；如果有两个参数，函数将会产生一个 $n \times m$ 的数组。这个函数的输出是由在 0.0 和 1.0 之间的随机数组成的数组

3.定义算法

函数 random0 的伪代码如下:

```
function ran = random0 (n, m)
Check for valid arguments
Set m ← n if not supplied
Create output array with "zeros" function
```

```
for ii = 1:number of rows
    for jj = 1: number of columns
            ISEED ← mod (8121 * ISEED + 28441, 134456)
            ran(ii,jj) ← ISEED /134456
    end
end
```

其中，ISEED 是全局变量，所以它能被所有的函数调用。函数 seed 的伪代码如下:

```
function seed ( new_seed)
new_seed ← round( new_seed)
ISEED ← abs( new_seed)
```

当用户输入不是整数，round 函数会对其进行四舍五入。如果输入的是一个负数，abs 将会把他取正。用户事先不需知道只有正整数才是合法的种子。

ISEED 是全局变量，所以它能被所有的函数调用。

4.把算法转化为 **MATLAB** 语句

函数 random0 的代码如下:

```
function ran = random0(n,m)
%RANDOM0 Generate uniform random numbers in [0,1)
% Function RANDOM0 generates an array of uniform
% random numbers in the range [0,1). The usage
% is:
%
% random0(n)            --Generate an n x n array
% random0(n,m)          --Generate an n x m array
% Define variables:
% ii                    --Index variable
% ISEED                   --Random number seed (global)
% jj                    --Index variable
% m                       --Number of columns
% msg                     --Error message
% n                       --Number of rows
% ran                     --Output array
% Record of revisions:
% Date Programmer Description of change
% ==== ========== ====================
% 12/16/98 S. J. Chapman Original code
% Declare globl values
global ISEED            % Seed for random number generator
% Check for a legal number of input arguments.
msg = nargchk(1,2,nargin);
error(msg);
% If the m argument is missing, set it to n.
if nargin < 2
    m = n;
end
% Initialize the output array
ran = zeros(n,m);
% Now calculate random values
for ii = 1:n
    for jj = 1:m
        ISEED = mod(8121*ISEED + 28411, 134456 );
        ran(ii,jj) = ISEED / 134456;
    end
end
```

函数 seed 的代码如下:

```
function seed(new_seed)
```

```
%SEED Set new seed for function RANDOM0
% Function SEED sets a new seed for function
% RANDOM0. The new seed should be a positive
% integer.
% Define variables:
% ISEED                 --Random number seed (global)
% new_seed              --New seed
% Record of revisions:
% Date Programmer Description of change
% ==== ========== ====================
% 12/16/98 S. J. Chapman Original code
% Declare globl values
global ISEED % Seed for random number generator
% Check for a legal number of input arguments.
msg = nargchk(1,1,nargin);
error(msg);
% Save seed
new_seed = round(new_seed);
ISEED = abs(new_seed);
```

5.检测产生的 **MATLAB** 程序

如是程序产生的这些数是真正的取值范围在 0≤ran<1.0 的等可能性随机数，那么它们的平均数应接近 0.5，它们的标准差应接近 $\frac{1}{\sqrt{12}}$。

进一步说，如果一个如果把区间[0, 1)分许多相同长度的子区间。那么落在每一个子区间的随机数的数目应当是相同的。我们可以利用柱状统计图来统计落于每一个子区间的随机数的数目。**MATLAB** 函数 hist 能够读取输入数据并能创建出相应的柱状图，所以我们将利用它随机数的等可能性。

1.调用函数 seed，把 new_seed 设置为 1024

2.调用 random0(4)，观察得到的结果

3.再次调用 rnadom0(4)，证明每次产生的数不相同

4.重新调用函数 seed，把 new_seed 设置为 1024

5.再次调用 rnadom0(4)，观察与 2 得到的结果是否相同

6.调用 random0(2, 3)证明函数可以输入两个参数

7.调用 random0(1, 20000)并计算产生的数组的平均数和标准差。看得到结果是否分别与 0.5，$\frac{1}{\sqrt{12}}$ 接近。

```
>> seed(1024)
>> random0(4)
ans =
    0.0598    1.0000    0.0905    0.2060
    0.2620    0.6432    0.6325    0.8392
    0.6278    0.5463    0.7551    0.4554
    0.3177    0.9105    0.1289    0.6230
>> random0(4)
ans =
    0.2266    0.3858    0.5876    0.7880
    0.8415    0.9287    0.9855    0.1314
    0.0982    0.6585    0.0543    0.4256
    0.2387    0.7153    0.2606    0.8922
>> seed(1024)
>> random0(4)
ans =
    0.0598    1.0000    0.0905    0.2060
    0.2620    0.6432    0.6325    0.8392
```

```
        0.6278    0.5463    0.7551    0.4554
        0.3177    0.9105    0.1289    0.6230
>> random0(2,3)
ans =
        0.2266    0.3858    0.5876
        0.7880    0.8415    0.9287
>> arr = random0(1,20000);
>> mean(arr)
ans =
        0.5020
>> std(arr)
ans =
        0.2881
>> hist(arr,10);
>> title('\bf Historygram of the Output of random0');
>> xlabel('Bin')
>> ylabel('Count')
```

检测的结果是合理的，这些数据的平均值为 0.5020，理论值为 0.5，实际标准差为 0.2881，理论值为 0.2887 接近。柱状图如图 5.5 所示。

在 **MATLAB** 中，有两个产生随机数的内建函数。它们是

rand	用于产生等可能的随机数
randn	用于产生普通的随机数

这两个函数要远比我们创建这个随机数发生器要快得多，产生的随机数也多得多。如果你需要在你的程序中创建一些随机数，可调用它们。

调用函数 rand 和 randn 的形式如下

rand	产生一个随机数
rand(n)	产生一个 n×n 的随机数数组
rand(n, m)	产生一个 n×m 的随机数数组

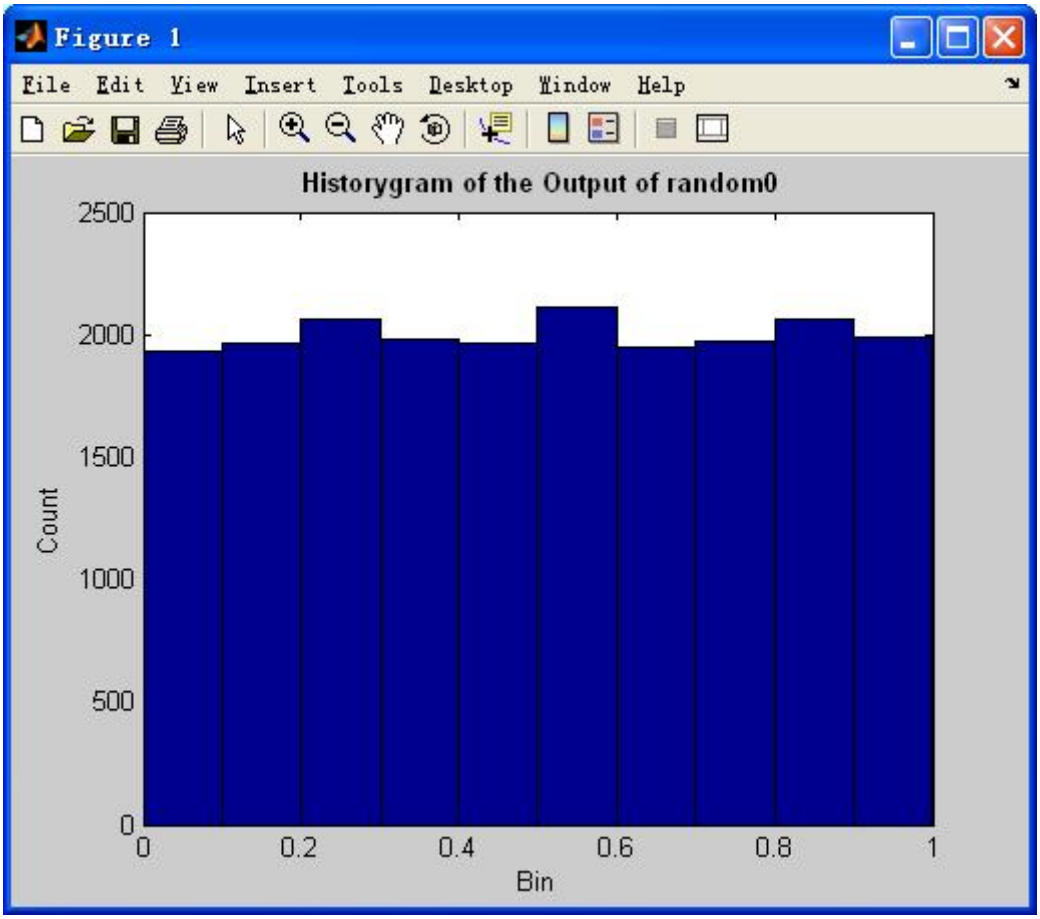


图 5.5 函数 random0 得到的柱状图

5.5 在函数调用两次之间本地数据的存储

当一个函数执行结束，由这个函数创建的特定的工作区将会被破坏，所以在这个函数中的所有本地变量将会消失。当这个函数下一次被调用的时侯，一个新的工作区将会被创建，而且所有本地变量的值都返回为默认。这种特性是我们所期望的，因为只有这样 **MATLAB** 函数才能被重复调用而不受上一次影响。

但在有些情况下，多次调用一个函数，存储一些本地变量的信息还是有用的。例如，我们想创建一个计数器，对函数调用的次数进行计数。如果每一次函数结束执行，计算器就会被破坏，那么计数不超过 1。

从 **MATLAB**5.1 开始，**MATLAB** 中就有了一个特殊的机制。这种机制允许多次调用一个函数时，保存本地变量。**持久内存**(persistent memory)是内存的一种类型，在函数上一次调用之后，这一步调用之前，本地变量被保存在持久内存，值不变。

持久变量应用语句声明。它的形式如下:

```
persistent var1 var2 var3 ...
```

var1，var2，var3...是存储于持久内存中的变量。

好的编程习惯

在两次函数调用之间有持久内存保存本地数据。

例 5.5 运行平均数

当我们键入一些变量总想得到他的统计量。**MATLAB** 内建函数 mean 和 std 就是进行统计数据运算的。我们对一系列的数利用这两个函数进行运算后，再键入一个新数，重新

计算。这时我们就可以利用持久内存提高运算的效率。

算术平均数的定义如下:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (5.8)$$

其中 x_i 是 N 样品中的第 i 个样品。标准差的定义如下:

$$s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1\_}} \qquad (5.9)$$

标准差则体现随机变量取值与其期望值的偏差。标准差的值较大，则表明该随机变量的取值与其期望值的偏差较大，反之，则表明此偏差较小。如果我们能够记录下样本的个数 N，样本的和 Σx_i 以及样本的平方和 Σx_i^2，我们在任何的时侯就可以通过公式（5.8）和（5.9）计算出它的平均数和标准差。编写一个程序，计算当前输入数据的当前输入数据的平均数和标准差。

答案

函数必须能够每接受一次输入值并记录下对应的 N，Σx_i 和 Σx_i^2，用于计算当前的平均数和标准差。N，Σx_i 和 Σx_i^2必须存储在持久内存中，这样在两次调用之间，它不会消失。最后函数必须有一种机制，把运行的和清零。

1.陈述问题

编写一个程序，计算当前输入数据的当前输入数据的平均数和标准差。函数必须有一种机制，把运行的和清零。

2.定义输入输出值

这个函数需要两种类型的输入量

(1)字符型变量“reset”用于 N，Σx_i 和 Σx_i^2的清零

(2)每一次调用函数用于参与运算的数字类型数据这个函数的输出量为到目前为止函数所接受的所有数字数据的数学期望和平方和。

(3)设计算法这个函数被分为四大步。

```
Check for a legal number of arguments
Check for a 'reset', and reset sums if present
Otherwise, add current value to running sums
Calculate and return running average and std dev
            if enough data is available. Return zeros if not enough data is available.
```

这些步骤的伪代码为

```
Check for a legal number of arguments
if x == 'reset'
   n  ←  0
   sum_x  ←  0
   sum_x2  ←  0
else
   n  ←  n+1
   sum_x  ←  sum_x + x
   sum_x2  ←  sum_x2 + x^2
end

% Calculate ave and sd
if n == 0
   ave  ←  0
   std  ←  0
elseif n == 1
   ave  ←  sum_x
   std  ←  0
```

```
else
   ave ← sum_x / n
   std ← sqrt((n*sum_x2 – sum_x^2) / (n * (n-1)))
end
```

4.把算法转化为 **MATLAB** 代码

```
function [ave, std] = runstats(x)
%RUNSTATS Generate running ave / std deviation
% Function RUNSTATS generates a running average
% and standard deviation of a data set. The
% values x must be passed to this function one
% at a time. A call to RUNSTATS with the argument
% 'reset' will reset tue running sums.
% Define variables:

% ave                    --Running average
% msg                    --Error message
% n                      --Number of data values
% std                    --Running standard deviation
% sum_x                  --Running sum of data values
% sum_x2                 --Running sum of data values squared
% x                      --Input value
% Record of revisions:
% Date          Programmer          Description of change
% ====          ==========          =====================
% 12/16/98    S. J. Chapman      Original code
% Declare persistent values
persistent n                % Number of input values
persistent sum_x            % Running sum of values
persistent sum_x2           % Running sum of values squared
% Check for a legal number of input arguments.
msg = nargchk(1,1,nargin);
error(msg);

% If the argument is 'reset', reset the running sums.
if x == 'reset'
      n = 0;
      sum_x = 0;
      sum_x2 = 0;
else
      n = n + 1;
      sum_x = sum_x + x;
      sum_x2 = sum_x2 + x^2;
end

% Calculate ave and sd
if n == 0
      ave = 0;
      std = 0;
elseif n == 1
      ave = sum_x;
      std = 0;
else
      ave = sum_x / n;
      std = sqrt((n*sum_x2 - sum_x^2) / (n*(n - 1)));
end
```

5.检测程序

为了检测这个函数，我们必须创建一个用于复位 runstats 的脚本文件：读取输入数据，调用 rnnstats 函数，并显示出相应的统计量。

一个合适的脚本文件显示如下：

```
% Script file: test_runstats.m
%
% Purpose:
% To read in an input data set andn calculate the
% running statistics on the data set as the values
% are read in. The running stats will be written
% to the Command window.
%
% Record of revisions:
% Date          Programmer          Description of change
% ====          ==========          =====================
% 12/16/98    S. J. Chapman      Original code
%
% Define variables:
% array                         --Input data array
% ave                            --Running average
% std                            --Running standard deviation
% ii                             --Index variable
% nvals                          --Number of input values
% std                            --Running standard deviation

% First reset running sums
[ave std] = runstats('reset');
% Prompt for the number of values in the data set
nvals = input('Enter number of values in data set: ');
% Get input values
for ii = 1:nvals
    % Prompt for next value
    string = ['Enter value ' int2str(ii) ': '];
    x = input(string);
    % Get running statistics
    [ave std] = runstats(x);
    % Display running statistics
    fprintf('Average = %8.4f; Std dev = %8.4f\n',ave, std);
end
```

为了检测函数，通过手动计算 5 个数得到相应的统计量，并与程序得到的结果进行比较。如果这个 5 个数分别为：

```
3.0,    2.0,    3.0,    4.0,    2.8
```

那么手动运算的结果为

Value	n	Σx	Σx^2	Average	Std_dev
3.0	1	3.0	9.0	3.00	0.000
2.0	2	5.0	13.0	2.50	0.707
3.0	3	8.0	22.0	2.67	0.577
4.0	4	12.0	38.0	3.00	0.816
2.8	5	14.8	45.84	2.96	0.713

程序得到的结果为

```
>> test_runstats
Enter number of values in data set: 5
Enter value 1: 3
Average =     3.0000; Std dev =     0.0000
Enter value 2: 2
```

```
Average =     2.5000; Std dev =     0.7071
Enter value 3: 3
Average =     2.6667; Std dev =     0.5774
Enter value 4: 4
Average =     3.0000; Std dev =     0.8165
Enter value 5: 2.8
Average =     2.9600; Std dev =     0.7127
```

这个运算结果与上面的手动计算相符。

5.6 函数的函数(function functions)，

函数的函数(function functions)是指函数的输入参数中含有其他的函数，传递给函数的函数的变量名一般情况应用于这个函数执行的过程中。

例如，**MATLAB** 中有一个函数的函数叫做 fzero。这个函数用于找到传递给它的函数值为 0 时的自变量。例如，语句 fzero('cos', (0, pi))，它能确定 cos 函数在区间[0, π]中何时为 0。语句 fzero('exp(x)-2', [0 1])在区间[0, 1]中何时为 0。当这些语句被执行时，将产生如下的结果：

```
>> fzero('cos',[0 pi])
ans =
      1.5708
>> fzero('exp(x)-2',[0 1])
ans =
      0.6931
```

函数的函数操作的关键字有两个专门的 maltab 函数，eval 和 feval。。函数 eval 对一个字符串进行求值，就如它在命令窗口中已经键入了一样。函数 feval 用一个特定的输入值对命名的函数进行求值。函数 eval 的形式如下：

```
eval(string)
```

例如，语句 x = eval('sin(pi/4)')产生的结果如下：

```
>> x = eval('sin(pi/4)')
x =
      0.7071
```

下面是一个例子，构建一个字符串，并用 eval 函数对其进行求值

```
x = 1;
str = ['exp(' num2str(x) ')-1'];
res = eval(str);
```

在这种情况下，变量 str 的内容为 exp(1)-1，所以 eval 产生的结果为 1.7183。

函数 feval 对在 M 文件进行定义的命名函数进行求值，要求有指定的输入值。函数 feaval 的基本形式如下

feval(fun, value).

例如，语句 x=feval('sin'，pi/4)产生的结果如下

```
>> x = feval('sin',pi/4)
x =
      0.7071
```

更多的函数的函数将会在表 5.1 中列出。在命令窗中键入 help 函数名，了解他们的用途。

表 5.1 常见的函数的函数

fminbnd	求函数的最小值，这函数只有一个自变量
fzero	找出函数为 0 时的自变量的值
quad	在数学上组合一个函数
ezplot	简单易用的函数画图
fplot	通过函数名画出这个函数的图象

例 5.6

创建一个函数的函数，它能够画出所有只有一个自变量的 **MATLAB** 函数的图象，自变量的范围是用户指定的始值和终值。

答案

这个函数有两个输入参数，第一个是要画的函数的函数名，第二个是两元素向量，它指明了画图的取值范围。

1.陈述问题

创建一个函数的函数，它能够画出所有只有一个自变量的 **MATLAB** 函数的图象，自变量的范围由用户指定。

2.定义输入输出函数的输入有两个

(1)包含有函数名的字符串

(2)包含有起始值和终值的 2 元素向量函数的输出是所要画的图象

3.设计算法这个函数可以分为 4 大步：

Check for a legal number of arguments
Check that the second argument has two elements
Calculate the value of the function between the start and stop points
Plot and label the function

第三四大步的伪代码如下

```
n_steps ← 100
step_size ← (xlim(2) – xlim(1)) / nsteps
x ← xlim(1):step_size:xlim(2)
y ← feval(fun, x)
plot(x, y)
title(['bf Plot of function ' fun ' (x)'])
xlabel('\bfx;)
ylabel(['bf ' fun ' (x)'])
```

4.把算法转化为 **MATLAB** 语句

```
function quickplot(fun,xlim)
%QUICKPLOT Generate quick plot of a function
% Function QUICKPLOT generates a quick plot
% of a function contained in a external mfile,
% between user-specified x limits.
% Define variables:
% fun                    --Function to plot
% msg                    --Error message
% n_steps                --Number of steps to plot
% step_size              --Step size
% x                      --X-values to plot
% y                      --Y-values to plot
% xlim                   --Plot x limits
% Record of revisions:
% Date Programmer Description of change
% ==== ========== ====================
% 12/17/98 S. J. Chapman Original code
% Check for a legal number of input arguments.
msg = nargchk(2,2,nargin);
error(msg);
% Check the second argument to see if it has two
% elements. Note that this double test allows the
% argument to be either a row or a column vector.
if ( size(xlim,1) == 1 & size(xlim,2) == 2 ) | ...
( size(xlim,1) == 2 & size(xlim,2) == 1 )
    % Ok            --continue processing.
    n_steps = 100;
```

```
        step_size = (xlim(2) - xlim(1)) / n_steps;
        x = xlim(1):step_size:xlim(2);
        y = feval(fun,x);
        plot(x,y);
        title(['\bfPlot of function ' fun '(x)']);
        xlabel('\bfx');
        ylabel(['\bf' fun '(x)']);
    else
        % Else wrong number of elements in xlim.
        error('Incorrect number of elements in xlim.');
    end
```

5。检测程序为了检测这个程序，我们应当用正确或错误的输入输出参数来调用这个程序。证明它能区分正确和错误的输入参数。结果如下

```
>> quickplot('sin')
??? Error using ==> quickplot
Not enough input arguments.

>> quickplot('sin', [-2*pi 2*pi], 3)
??? Error using ==> quickplot
Too many input arguments.

>> quickplot('sin', -2*pi)
??? Error using ==> quickplot
Incorrect number of elements in xlim.

>> quickplot('sin', [-2*pi 2*pi])
```

最后一次被调用的结果是正确的，它的图象如图 5.6 所示。

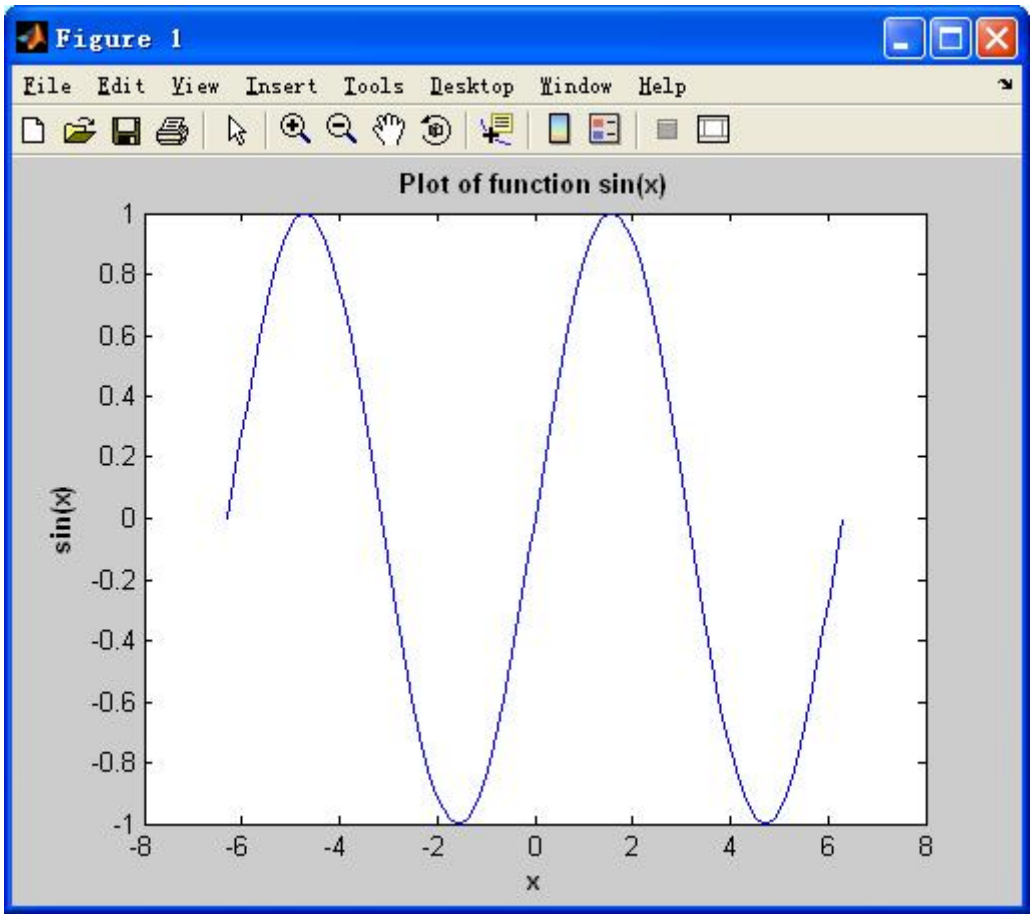


图 5.6 由 quickplot 函数产生的 sinx 图象

5.7 子函数和私有函数

在一个单个的文件中我们可以创建多个函数。如果超过 1 个的函数出现在一个文件中，那么最上面的那个函数为普通函数，下面的函数称为**子函数**或**中间函数**。子函数看起来和普通函数一样，但是只能被同一文件中的函数调用。

下面的例子定义了一个函数 mystats 和两个子函数 mean 和 median。函数 mystats 能被其他的 maltab 函数调用，但是子函数 mean 和 median 只能同一文件中的其他函数调用。

```
function [avg, med] = mystats(u)
%MYSTATS Find mean and median with internal functions.
%     Function MYSTATS calculates the average and median
%     of a data set using subfunctions.

n = length(u);
avg = mean(u, n);
med = median (u, n);

function a = mean(v, n)
%     Subfunction to calculate average.
a = sum(v) / n;
function m = median(v, n)
%     Subfunction to calculate median

w = sort(v);
if rem(n, 2) ==1
      m = w((n+1)/2);
else
      m = (w(n/2) + w(n/2 + 1))/2;
end
```

私有函数是指属于以 private 为名字的子目录中的函数。这些函数只有在父目录中才是可见的。例如，假设在 **MATLAB** 搜索路径中有一 testing 目录，在 testing 目录中，又有一个 private 子目录。private 中的函数只能由 testing 中的函数调用。因为对于父目录以外目标私有函数是不可见的，所以它能用其他目录中的函数重名。有了这种特性，如果你要创建自己的函数，则不必要考虑与其他目录重名。因为 **MATLAB** 先对私有函数查找，然后再对标准的 M 文件函数进行查找，所以它将首先找到私有函数 test.m，再找到非私有 M 文件 test.m。

在包含有你的函数的目录中，我们可以很创建你的私有目录。不要在你的搜索路径中放置你的私有目录。

在一个 M 文件中，调用一个函数，**MATLAB** 先检查看他是否是一个子函数。如果它不是那就检查它是不是一个私有函数。如果也不是私有函数，**MATLAB** 就会检它在不在标搜索路径中。

如果你有特殊的目的，**MATLAB** 函数只能由其他的函数调用，而绝不能由使用者调用并考虑用子函数或私有函数来隐藏它们。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

好的编程习惯

用子函数或私有函数来隐藏特殊目的的函数，这些隐藏的函数只能被其他函数调用。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

5.8 总结

在第五章中，我们向大家介绍了用户自定义函数。函数是 M 文件的一种特殊类型，它通过输入参数接受数据，通过输出参数返回结果。每一个函数都有其独立的工作区。

MATLAB 通过按值传递机制将参数传递给函数，这意味着 **MATLAB** 把每一个参数复

制，并把这个拷贝传递给函数。这个复制是非常重要的，因为函数可以自由的修改输入参数，而不会影响到程序中的实参。

MATLAB 函数支持改变输入输出参数的个数。函数 nargin 可以报告函数在调用过程中所需的实参个数。函数 nargout 则可以报告输出参数的个数。

把数据存于全局内存中，可以实现 **MATLAB** 函数之间数据的共享。全局变量的声明要用到 global 语句。全局变量可由所有声明它的所有函数共享。为了方便，全局变量应在 M 文件的开头声明。

两次调用同一函数之间，中间数据可以存储在持久内存。持久变量可能用 persistent 语句声明。

函数的函数是指函数的输入参数中含有其他的函数，传递给函数的函数的变量名一般情况应用于这个函数执行的过程中。

子函数是在一个单独文件中的附加函数，它只能被同一文件中的其他函数访问。私有函数是在 private 子目录中的函数，它们只能被父目录中的函数访问。子函数和私有函数主要用于限制 **MATLAB** 函数的访问。

5.8.1

好的编程习惯的总结

1.把大的程序分解小的，易于理解的函数

2.在 M 文件的开头声明全局变量。以区分本地变量

3.在函数中的初始注释行之后和第一个可执行性语句之前声明全局变量

4.全局变量适用大规模数据的传输

5.在两次函数调用之间有持久内存保存本地数据。

6.用子函数或私有函数来隐藏特殊目的的函数，这些隐藏的函数只能被其他函数调用。隐藏这些函数防止了它们偶然的使用，也能防止与其他公共函数重名时发生的冲突。

5.8.1 **MATLAB** 总结

下面是对 MATLAB 函数和命令的总结，并带有简短的描述。

nargin	这个函数返回调用这个函数时所需要的实际输入参数的个数
nargout	这个函数返回调用这个函数时所需要的实际输出参数的个数
nargchk	如要一个函数调用被调用时参数过多或过少，那么 nargchk 函数将返回一个标准错误信息
error	显示错误信息，并中止函数以免它产生这个错误。如果参数错误是致命的，这个函数将会被调用。
warning	显示警告信息并继续执行函数，如果参数错误不是致命的，执行还能继续，则这个将会被调用。
inputname	这个函数将会返回对于特定参数个数的实际变量名。
rand	产生一个随机数
rand(n)	产生一个 n×n 的随机数数组
rand(n,m)	产生一个 n×m 的随机数数组
rand	用于产生等可能的随机数
randn	用于产生普通的随机数

5.9 练习

5.1

函数与脚本文件之前的区别是什么？

5.2

当一个函数被调用，数据是是怎样从调用者传递到函数的。函数是怎样把结果返回给调用者?

5.3

在 **MATLAB** 中应用按图传递机制的优点与缺点?

5.4

修改本章中的选择性排序函数，让他能够接受第二个选择性参数。”up”或”down”。当参数为”up”时，数据按升序排列，当参数为”down”，数据按降序排列。如果输入只有一个，我们默认按降序排列。

5.5

编写一个函数，利用函数 random0 在[-1。0，1。0]产生一个随机数。使 random0 函数成为你新的函数的子函数。

5.6

编写一个函数，利用函数 random0 在[low,high]产生一个随机数，其中 low，high 分别代表输入参数。把 random0 函数成为你新函数的一个私有函数。

5.7

骰子模拟。模拟掷骰子的情况在现实中非常有用。编写一个 maltab 程序模拟掷骰子，每次产生一个 1 到 6 之间随机整数。

5.8

道路交通密度。函数 random0 将在[0.0, 1.0]产生一个等可能性随机数。如果随机事件结果是等可能性，这个函数适合模拟这类随机事件。但是，很多事件的发生都不是等可能性的，那么这个函数不适合模拟这类情况。

例如，一交通工程师研究在一段时间隔 t 内通过某一地点汽车数，发现 k 辆汽车通过一指定地点可能性为

$$P(k,t)=e^{-\lambda t}\frac{(\lambda t)^k}{k!}\quad (t\geq 0,\lambda>0,k=0,1,2,...) \tag{5.10}$$

它的分布符合**指数分布**。指数分布在科研和工程上有很多的应用。例如，在时间 t 内接打电话的次 k，指定容器内的病毒 k，以及复合系统的出错次数 k 都符合指数分布。

编写一个函数，对任意 k，t 和 λ 求指数分布。通过计算在 1 分钟内通过高速路上指定一点 1，2，3，4，5 辆汽车的概率。并画出相应的图象。已知 λ 为 1.5。

5.9

编写下面三个函数，用以计算数 x 的双曲正弦，双曲余弦和双曲正切用你的函数画出它们所对应的图象。

$$\sinh(x)=\frac{e^x-e^{-x}}{2},\cosh(x)=\frac{e^x+e^{-x}}{2},\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

5.10

向量叉乘。编写一个程序计算向量 V_1 和 V_2 的叉乘积。

$V_1\times V_2=(V_{y1}V_{z2}-V_{y2}V_{z1})i+(V_{z1}V_{x2}-V_{z2}V_{x1})j+(V_{x1}V_{y2}-V_{x2}V_{y1})k$

其中 $V_1=V_{x1}i+V_{y1}j+V_{z1}k$，$V_2=V_{x2}i+V_{y2}j+V_{z2}k$。注意这个函数返回的一个实数组。用这个函数计 V_1 = [-2, 4, 0.5]和 V_2 = [0.5, 3, 2]的叉乘积。

5.11

带有搬运的排序。对数组 arr1 进行升序排序，与 arr1 中相对应的 arr2 中的元素也要发

生改变。对这个种排序，每次 arr1 中的一个元素与另一个元素进行交换，arr2 中对应的元素也要进行相应的交换。当排序结束时 arr1 中的元素按升序排列，arr2 中的元素也会有相应的变化。例如下面两个数组

Element	arr1	arr2
1.	6.	1.
2.	1.	0.
3.	2.	10.

当 arr1 的数组排序结束后，arr2 也要进行相应的变化。两数组为

Element	arr1	arr2
1.	1.	0.
2.	2.	10.
3.	6.	1.

编写一个程序，对第一个实数组进行按降序排列，对第二个数组进行相应交化。用下面两个数组检测你的程序

```
a = [-1,   11,   -6,   17,   -23,   0,   5,   1,   -1];
b = [-31,  102,  36,  -17,    0,   10,  -8,  -1,   -1];
```

5.12

用帮助工作台查找 **MATLAB** 标准函数 sortrows 的信息，运行 sortrows 函数，和前面练习中的排序函数进行比较。为了达到此目的，创建含有 1000×2 元素的数组的两个副本，数组中是随机数。应用上面的两个函数分别对第一行进行排序，第二行也对应改变。用 tic 和 toc 函数确定每一个排序执行所需要的时间。你编写的函数的运行速度与标准函数的运行速度相比如何？

5.13

图 5.7 显示是漂浮在海洋上两条船。1 号船所在的位置为（x_1，y_1）按 θ_1 方向运行，2 号船所在的位置为（x_2，y_2）按 θ_2 方向运行。假设一个物体与 1 号船的距离 r_1，并产生 φ_1。编写一个程序计算 2 号船到物体的距离 r_2 和夹角 φ_2。

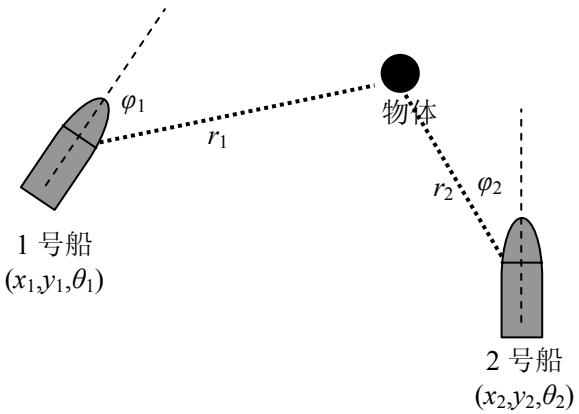


图 5.7 1 号船所在的位置为（x_1，y_1）按 θ_1 方向运行，2 号船所在的位置为（x_2，y_2）按 θ_2 方向运行。

5.14

函数的最大值和最小值。编写一个函数，用于计算任意函数 f(x)在一定区间内的最大值和最小值。所要求最大值和最小值的函数应当用参数的方式传递给你编的函数。这个函数应当有下面的输入参数。

first_value	--x 的第一个值
last_value	--x 的最后一个值
num_steps	--x 取值的步长
func	--所要求的值的函数名

函数的输出参数应为

xmin	--函数 f(x)为最小值时的 x 值
min_value	--函数 f(x)的最小值
xmax	--函数 f(x)为最大值时的 x 值
max_vlaue	--函数 f(x)的最大值

确保输入参数的个数有效，你可以通过 help 和 lookfor 命令得到一定的帮助。

5.15

编写一个程序，用来检测上题中产生的函数。这个检测程序把自定义函数 $f(x)=x^3-5x^2+5x-2$ 传递给函数的函数，并在区间[-1,3]内每隔 1/50 取一次值，找出函数的最大值和最小值，并打印出来。

5.16

函数的微分。函数微分的定义如下

$$\frac{d}{dx}f(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x} \tag{5.11}$$

简化后为

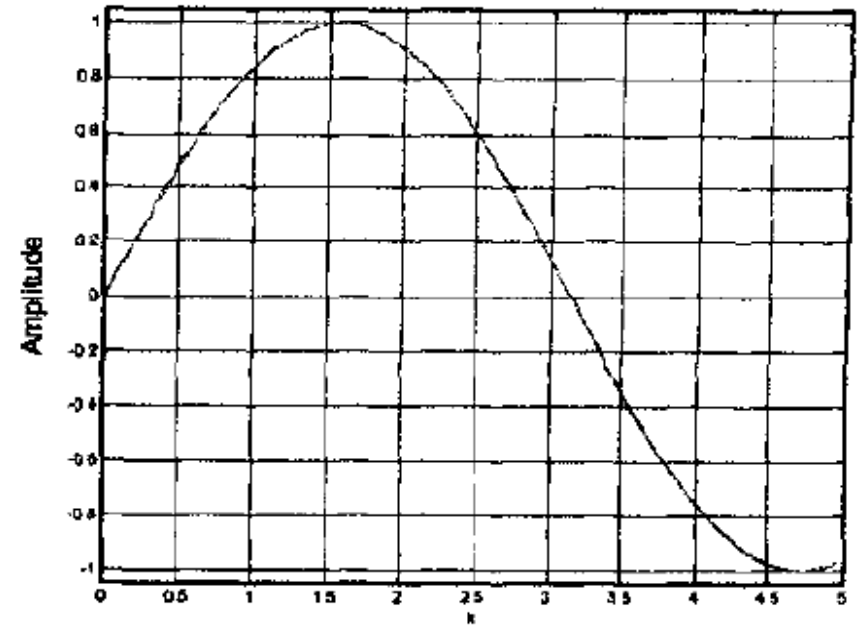
$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{\Delta x} \tag{5.12}$$

其中 $\Delta x=x_{i+1}-x_i$。

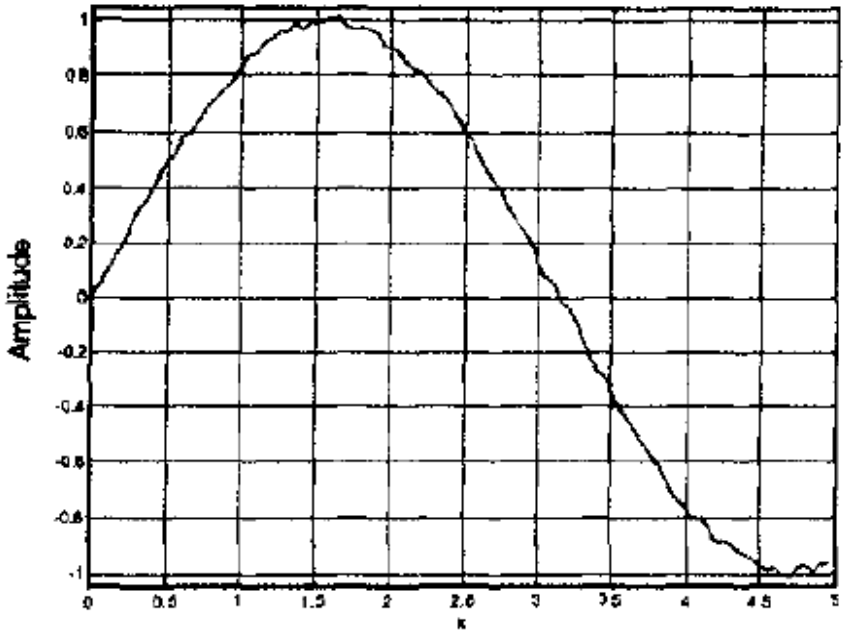
假设一向量 vect 包括一个函数的每隔 dx 的抽样。编写一个程序，通过 5.12 式计算这个向量的微分。这个函数应该能检查 dx 的值，看它是否大于 0，这样可以防止这个函数发生 0 除错误。为了检测你的函数，你应当在脑中产生一个函数，并知道它的微分是什么。与函数产生的结果进行对比。最好的检测函数应是 sin(*x*)。它的微分为 $\frac{d}{dx}(\sin x)=\cos x$。产生一个向量，这个向量包含一百个 sin*x* 函数值，*x* 从 0 开始取值，步长 Δx 为 0.05。用你的函数对这个向量进行微分，然后产生的结果与正确的结果进行对比。你的函数计算得到的结果与正确的微分结果有多接近？

5.17

带有噪声的微分。我们现在编写一个程序，观察输入噪声对数据微分的影响。第一步，产生一个向量，这个向量包含一百个 sinx 函数值，x 从 0 开始取值，步长 Δx 为 0.05。下一步用 random0 函数产生一系列的噪声，并把它加入到上面向量的抽样中(如图图 5.8)。噪声产生的最大偏差不会超过信号幅度的±0.02。现在我们用上题中微分函数对向量进行运算。得到结果和理率值有多相近呢？



(a)



(b)

图 5.8 (a)无噪声的 sinx 的图象（b）带有噪声的 sinx 的图象

5.18

线性最小二乘拟合。开发一个函数，用于计算拟合输入数据的最小二乘直线的斜率 m 和截距 b。输入数据点集（x，y）由两个数据传递给函数，数组 x 和 y。用一个检测程序检测你的函数，表 5.2 中有 20 个点输入数据。

表 5.2

No.	x	y	No.	x	y
1	-4.91	-8.18	11	-0.94	0.21
2	-3.84	-7.49	12	0.59	1.73
3	-2.41	-7.11	13	0.69	3.96

4	-2.62	-6.15	14	3.04	4.26
5	-3.78	-5.62	15	1.01	5.75
6	-0.52	-3.30	16	3.60	6.67
7	-1.83	-2.05	17	4.53	7.70
8	-2.01	-2.83	18	5.13	7.31
9	0.28	-1.16	19	4.43	9.05
10	1.08	0.52	20	4.12	10.95

5.19

最小二乘拟合的相关系数

开发一个函数，既可以用于计算拟合输入数据的最小二乘直线的斜率 m 和截距 b。又可以计算拟合的相关系数。输入数据点集（x，y）由两个数据传递给函数，数组 x 和 y。计算 m 和 b 的公式在例 4.7 中已经给出。相关系数 r 的计算公式如下所示

$$r=\frac{n(\sum xy)-(\sum x)(\sum y)}{\sqrt{[(n\sum x^2)-(\sum x)^2][(n\sum y^2)-(\sum y)^2]}} \qquad (5.13)$$

Σx 代表 x 值的和

Σy 代表 y 值的和

Σx^2 代表 x 值的平方和

Σy^2 代表 y 值的平方和

Σxy 对应的 x，y 相乘的和

n 代表拟和中包括的点数

用一个检测程序检测你的函数，输入参数与上题相同。

5.20

生日问题

生日问题：如果在一个房间有 n 个人，那么有二个或多个人在同一天过生日的概率为多大？我们可以能过数学建模来解决这一问题。编写一个程序，计算在 n 个人中有二个或多个人在同一天过生日的概率，n 为输入参数。编写一个程序来检测这个函数，当 n=2，3，……，40 时，二个或多个人在同一天过生日的概率为多大？

5.21

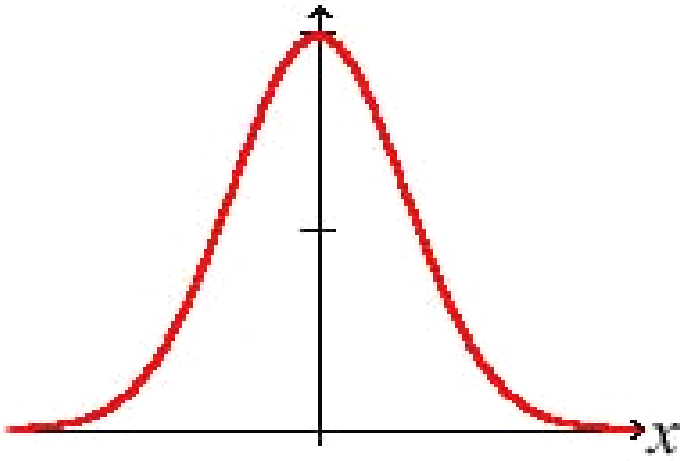
用函数 random0 产生三个由随机数组的数组。三个数组分别包含 100，1000，2000 个元素。下一步，用函数 tic 和 tic 对三个数组用函数 ssort 进行排序计时。随着元素数目的增加，排序消耗的时间如何变化？

5.22

正态分布

由 random0 产生的随机变量符合平均分布。另一种分布类型是正态分布（如图 5.9 所示）。如果一个正态分布的平均数为 0，标准差为 1.0，那么这个正态分布被称为标准正态分布。标准正态分布的公式为

$$p(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2} \qquad (5.14)$$



在(-1, 1)中遵守平均分布的随机变量能够产生遵守正态分布的随机变量。

1.在(-1,1)中任取遵守平均分布的随机变量 x_1 和 x_2，看 $tx_1^2+x_2^2<1$ 是否成立。如果成立用它们，如果不成立，则重试。

2.由下面公式得到的 y_1 和 y_2 将是正态分布随机变量。

$$y_1=\sqrt{\frac{-2\ln r}{r}}x_1 \qquad (5.15)$$

$$y_2=\sqrt{\frac{-2\ln r}{r}}x2 \qquad (5.16)$$

$$r=x_1^2+x_2^2 \qquad (5.17)$$

编写一个函数，每调用一次产生一个正态分布随机变量，通过得到 1000 个随机变量，计算他们的标准差，画出柱状图来检测你的函数。它们的标准差与 1.0 有多接近。

5.23

万有引力公式如下

$$F=G\frac{m_1m_2}{r^2} \qquad (5.18)$$

其中 G 是引力常量，大小为 $6.672\times10^{-11}\mathrm{Nm^2/kg^2}$，$m_1$ 和 m_2 是两物体的质量，单位为 kg，r 为两质点的间距，单位为 m。编写一个程序，已知两物体的质量和距离，计算它们之间的万有引力。在距地表 38000 米的高空重 800kg 的卫星与地球之间的万有引力为多少？用这个数据来检测你的程序

5.24

瑞利分布

瑞利分布是另一种在许多现实问题中出现的随机变量分布类型。符合瑞利分布的随机变量可以由两个符合正态分布的随机变量通过计算得到。计算方法如下所示，n_1 和 n_2 是符合正态分布的随机变量。

$$r=\sqrt{n_1^2+n_2^2} \qquad (5.19)$$

A.创建一函数 rayleigh(n,m)，它将返回一个 n×m 的数组，数组元素符合瑞利分布。如果只有一个输入参数 n，它将会产生一个 n 阶方阵。确保你设计的函数能够检测输入参数，并为 **MATLAB** 帮助系统提供适当文本。

B.通过产生 20000 个符合瑞利分布的随机变量。并画出它们的分布的柱状图。这个分布看起来像什么？

C.计算出这些随机变量的平均数和标准差

5.25

恒虚警率(CFAR)

图 5.10a 显示的是一个简易的雷达接收器。当一个信号被接受器接受，此信号中将包括由目标返回的有用信息，还有一些热噪声。当信号被发现处理后，我们就能够从热噪声背景中挑拣出有用信号。我们可以设定一个阈值，如果信号超过了这个阈值，那么就声明发现了目标。不好的是，接受的噪声也会偶然的超过阈值。如果这种情况发生，噪声被误认为目标，我们称之一个**虚警**。阈值要设得尽可能低，这样可以发现目标的微弱信号，但它又不能设得太低，这样我们就会得到许多的虚警。

在视频发现过后，这个接受机的热噪声符合瑞利分布。图 5.10b 显示的是平均振幅为 10V 的瑞利分布噪声的 100 个抽样，注意当发现阈值为 26 时，只有一次虚警。这些噪声抽样的概率分布如图 5.10c 所示。

发现阈值一般是噪声平均数的倍数，如果噪声水平改变，发现阈值也应随之改变，以控制虚警。这就是我们所说的恒虚警率发现。发现阈值单位一般有分贝。用 dB 表示的阈值和用电压表示的阈值关系如下：

$$\text{Threshold(volts)} = \text{Mean Nosise Level(volts)} \times 10^{\frac{dB}{20}} \tag{5.20}$$

或者是

$$\text{dB} = 20\log_{10}\left(\frac{\text{Threshold(volts)}}{\text{Mean Noise Level(volts)}}\right) \tag{5.21}$$

已知发现阈值，恒虚警率可由下面的公式求得。

$$P_{fa} = \frac{\text{Number of False Alarms}}{\text{Total Number of Samples}} \tag{5.22}$$

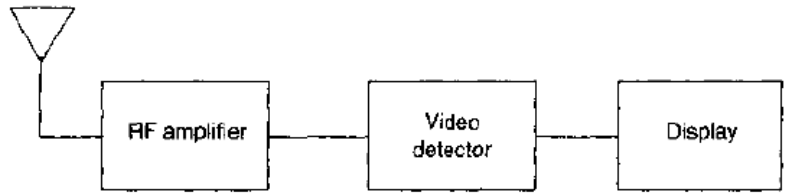
编写一个程序，产生 1000000 个随机噪声抽样，这些抽样的平均幅度为 10V，并遵守瑞利分布。当发现阈值分别超出平均噪声水平时 5,6,7,8,9,10,11,12 和 13dB 时，它的恒虚警率为多少。当发现阈值被设为多少时，恒虚警率为 10^{-4}。

5.26

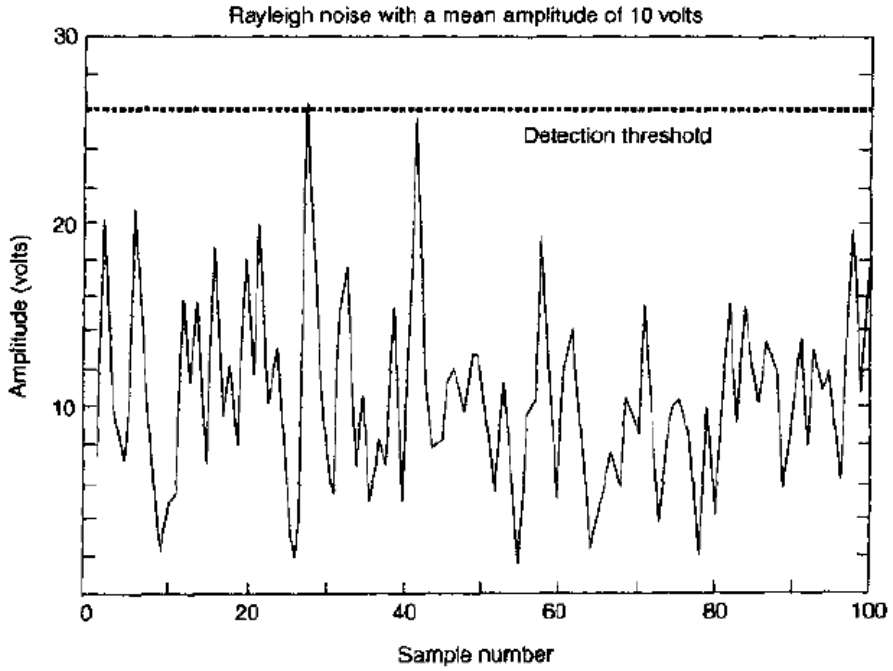
发现概率和虚警概率返回一个雷达目标的信号强度一般会在一定的时间内削弱。如要信号强度超过阈值，目标就会被发现。发现目标的概率可由以下公式计算。

$$P_d = \frac{\text{Number of Target Detections}}{\text{Total Number of Looks}} \tag{5.23}$$

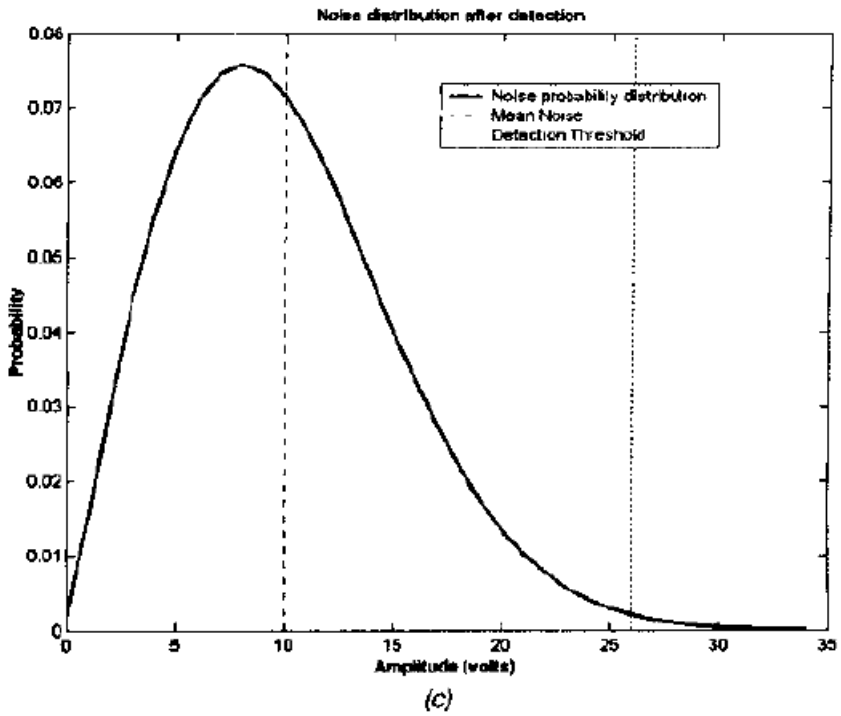
假设在特定位置的雷达不断地发射信号。在每一次发射信号时，在 10km 到 20km 的范围内被分为 100 个独立的范围抽样。这些抽样中的一个是目标，它的幅度符合正态分布，平均幅度为 7V，标准差为 1V。这 100 个抽样中包含有系统噪声，噪声幅度符合瑞利分布。平均幅度为 2V。当发现阈值分别为 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5 和 12.0dB 时，它的发现概率和虚警概率分别为多少？这个雷达的发现阈值应为多少？


RF amplifier
Video detector
Display

(a)


Rayleigh noise with a mean amplitude of 10 volts
Detection threshold
Amplitude (volts)
Sample number
30
20
10
0
20
40
60
80
100

(b)


Noise distribution after detection
Noise probability distribution
Mean Noise
Detection Threshold
Probability
Amplitude (volts)
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
5
10
15
20
25
30
35

(c)

图 5.10