第 3 章 程 序 设 计

CHAPTER 3

类似于其他高级语言编程，MATLAB 提供了非常方便易懂的程序设计方法，利用 MATLAB 编写的程序简洁、可读性强，而且调试十分容易。本章重点讲解 MATLAB 中最基础的程序设计，包括编程原则、分支结构、循环结构、其他控制程序命令及程序调试等内容。

学习目标：

（1）了解 MATLAB 编程基础知识；

（2）掌握 MATLAB 编程原则；

（3）掌握 MATLAB 各种控制指令；

（4）熟悉 MATLAB 程序的调试。

3.1 MATLAB 编程概述

MATLAB 拥有强大的数据处理能力，能够很好地解决几乎所有的工程问题。作为一款科学计算软件，MATLAB 能让用户任意编写函数，调用和修改脚本文件，以及根据需要修改 MATLAB 工具箱函数等。

3.1.1 编辑器窗口

在 MATLAB 中，单击 MATLAB 主界面“文件”选项卡下的“新建”按钮或者单击“新建”按钮下的“脚本”选项，此时的界面会出现“编辑器”窗口，如图 3-1 所示。

在编辑器窗口中，可以进行注释的书写，字体默认为绿色，新建文件系统默认为 Untitled 文件，依次为 Untitled 1、Untitled 2、Untitled 3、…，单击“保存”按钮可以另存为需要的文件名称。在编写代码时，要及时保存阶段性成果，单击“保存”按钮保存当前的 M 文件。

进行程序书写时或进行注释文字或字符时，光标是随字符而动的，可以更加轻松地定位书写程序所在位置。完成代码书写之后，要试运行代码，看看有没有运行错误，然后根据针对性的错误提示对程序进行修改。

MATLAB 运行程序代码，如果程序有误，MATLAB 就像 C 语言编译器一样，能够报错，并给出相应的错误信息；用户针对错误的信息，鼠标单击错误信息，MATLAB 工具能够自动定位到脚本文件（M 文件），供用户修改；此外用户还可以进行断点设置，进行逐行或者逐段运行，查找相应的错误和查看相应的运行结果，整体上使得编程简易。

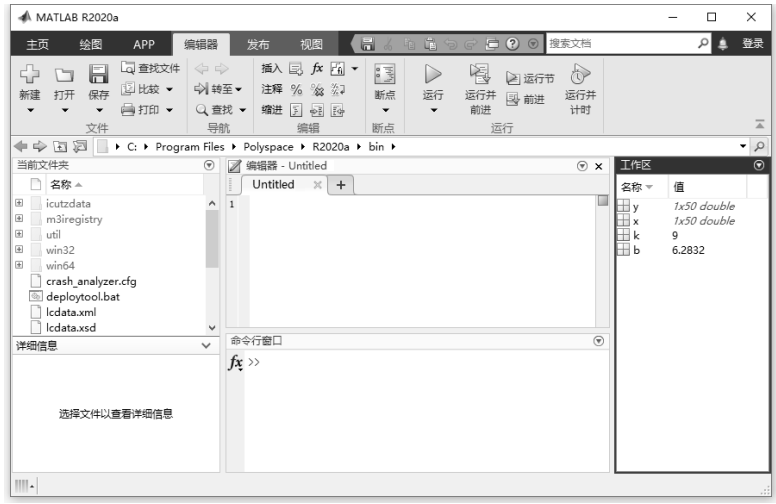

图 3–1　M–File 编辑

MATLAB 程序编辑在编辑器中进行，程序运行结果或错误信息显示在命令行窗口，程序运行过程产生的参数信息显示在工作区，如图 3–1 所示，因为程序有问题，命令行窗口中出现了程序错误提示。

【例 3-1】请修改图 3–2 中的程序代码，使得命令行窗口无错误提示，并给出正确结果。

解：由图 3–2 中代码可知，第 6、7 行中乘号运用有问题，修改如下：

```
clear,clc
n=3;
N=10000;
theta=2*pi*(0:N)/N;
r=cos(n*theta);
x=r.*cos(theta);
y=r.*sin(theta);
comet(x,y)
```

运行程序后，得到优化后的程序运行界面如图 3–3 所示，正确结果如图 3–4 所示。

图 3–2　MATLAB 编辑器错误提示

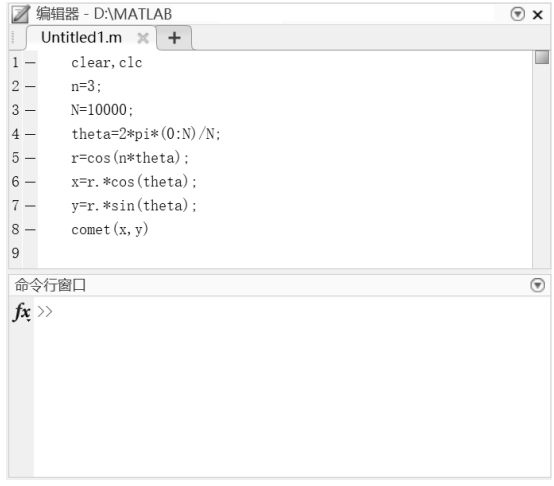

图 3–3　优化后的程序运行界面

3.1.2　编程原则

MATLAB 软件提供了一个供用户自己书写代码的文本文件，用户可以通过文本文件轻松地注释程序代码以及封装程序框架，真正地给用户提供一个人机交互的平台。MATLAB 具体的一个编程程序脚本文件如图 3–5 所示。

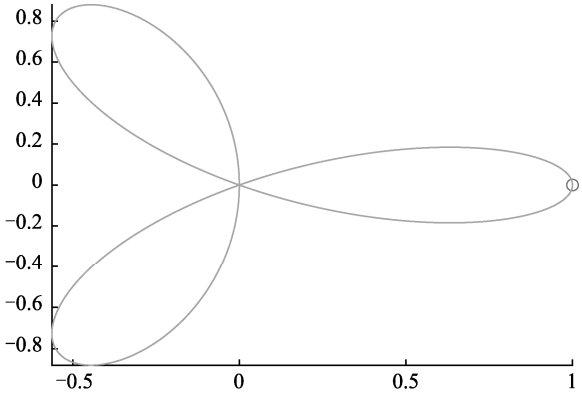

图 3–4　程序输出的正确结果

图 3–5　编程流程

前文中已经用到了%、clc、clear、close 等符号及命令，下面进行详细说明。

（1）%：表示注释符号，在注释符号后面可以写相应的文字或者字母，表示该程序语句的作用，使程序具有更强的可读性。

（2）clc：表示清屏操作。程序运行界面常常暂存运行过的程序代码，使屏幕不适合用户编写程序，采用 clc 命令能把前面的程序全部从命令行窗口中清除，以方便后续程序书写。

（3）clear：表示清除工作区中的所有数据，使后续程序运行的变量之间不相互冲突。编程时应该注意清除某些变量的值，以免造成程序运行错误，因为此类错误在复杂的程序中较难查找。

（4）close all：表示关闭所有的图形窗口，便于下一个程序运行时更加直观地观察图形的显示，为用户提供较好的图形显示界面。特别在图像和视频处理中，能够较好地实现图形参数化设计，以提高执行速度。

程序应该尽量显得清晰可见，多设计可调用的执行程序，能提升编程的效率，程序设计好以后可进行运行调试。

MATLAB 代码的编写通常强调高效性，譬如“尽量不要用循环”，除此之外，还要考虑代码（格式）的正确性、清晰性与通用性。

（1）高效性：循环向量化，少用或不用循环，尽量调用 MATLAB 自带函数；

（2）正确性：程序能准确地实现原有仿真目的；

（3）清晰性：养成良好的编程习惯，使程序具有良好的可读性；

（4）通用性：程序具有高度的可移植性和可扩展性，便于后续开发调用。

在 MATLAB 编程中，还需要遵循以下规则：

（1）定义变量，以英文单词小写缩写开头表示类别名，再接具体变量的英文名称。例如，定义变量存储临时数组 TempArray 的最大值为 maxTempArray。根据工程大小确定变量名长短，小范围应用的变量应该用短的变量名。定义要清晰，避免混淆。

（2）循环变量使用常用变量 i、j、k。当程序中使用复数时，采用 i、j 以外的循环变量以避免和虚数单位冲突，同时要在注释部分说明变量的意义。

（3）编写的程序应该高内聚、低耦合、模块函数化，便于移植、重复使用。

（4）使用 if 语句判断变量是否等于某一常数时，将常变量数写在等号之前，常数写在等号之后。例如判断变量 a 是否等于 10，写为 if a==10。

（5）用常数代替数字，少用或不用数字。例如例 3-1 中，如果要定义期望常量，写为 if a==10 则不标准。应该先定义 b=10，同时在注释中说明，然后在程序部分写为：if a==b。如果后续要修改期望常量，则在程序定义部分修改。

【例 3-2】在编辑器中编写程序代码示例。

解：在编辑器窗口中输入以下代码。

```
clc,clear,close                 %clc 清屏，clear 删除工作区中的变量，close 关掉显示图形窗口
format short
%Initial                        初始化操作
F = 0.3;                        %等效载荷（单位：kN）
l = 100/1000;                   %杆长（单位：mm）
d = 0.7/1000;                   %直径（单位：mm）
k = 20/1000;                    %两杆间距（单位：mm）
E = 70*10^9;                    %杨氏模量（单位：GPa）
A = pi*d^2/4;                   %杆的横截面积（单位：m²）
S1 = 10/1000                    %水平方向位移(单位：m)
Ia = pi*d^4/64                  %转动惯量（单位：kg·m²）
```

单击主界面“编辑器”选项卡“运行”面板中的“运行”按钮运行程序，输出结果如下

```
S1 =
    0.0100
Ia =
   1.1786e-14
```

如上述程序可知，程序采用清晰化编程，可以很清晰地知道每句程序代码是什么意思，通过一系列的求解，最终得到相应的结果输出，然后将所有子程序合并在一起来执行全部的操作。

调试过程中应特别注意错误提示，通过断点设置、单步执行等操作对程序进行修改，以便程序运行。当然，更复杂的程序还需要调用子程序，或与其他应用程序相结合，后文会进行相应的讲解。

3.2 M 文件和函数

M 文件和函数是 MATLAB 中非常重要的内容，下面分别进行讲解。

3.2.1 M 文件

M 文件通常为使用的脚本文件，即供用户编写程序代码的文件，通过代码调试可以得到优化的 MATLAB 可执行代码。

1. M文件的类型

MATLAB 程序文件分为函数调用文件和主函数文件，主函数文件通常可单独写成简单的 M 文件，执行

（run），得到相应的结果。

1）脚本文件

脚本文件即后缀为.m 的文件，脚本文件也是主函数文件，用户可以将脚本文件写为主函数文件。在脚本文件中可以进行主要程序的编写，遇到需要调用函数来求解某个问题时，则需要调用该函数文件，输入该函数文件相应的参数值，即可得到相应的结果。

2）函数文件

函数文件即可供用户调用的程序文件，能够避免变量之间的冲突。函数文件一方面可以节约代码行数，另一方面也可以通过调用函数文件使整体程序显得清晰明了。

函数文件和脚本文件有差别，函数文件也是一个能通过输入变量得到相应的输出变量，实现返回后的变量显示在命令行窗口或者供主函数继续使用。

函数文件中的变量将作为函数文件的独立变量，不和主函数文件冲突，因此极大地扩展了函数文件的通用性。主函数可以多次调用封装后的子函数，以达到精简、优化程序的目的。

2. M文件的结构

脚本文件和函数文件均属于 M 文件，函数名称一般包括文件头、躯干、结尾（end）。文件头首先是清屏及清除工作区变量。代码如下：

```
clc                                          %清屏
clear all;                                   %删除工作区变量
clf;                                         %清空图形窗口
close all;                                   %关掉显示图形窗口
```

躯干部分编写脚本文件中各变量的赋值，以及公式的运算。躯干部分一般为程序的主要部分，进行部分注释是必要的，读者可以清晰地看出程序要解决的问题以及解决问题的思路。代码示例如下：

```
l = 100/1000;                                %杆长（单位：mm）
d = 0.7/1000;                                %直径（单位：mm）
x=linspace(0,l,200);
y=linspace(-d/2,d/2,200);
```

结尾（end）常用于主函数文件中，一般的脚本文件不需要加，end 常和 function 搭配，代码如下：

```
function djb
…
end
```

end 语句表示该函数已经结束。在一个函数文件中可以同时嵌入多个函数文件，具体如下：

```
function djb1
…
End
function djb2
…
End
…
function djb3
…
end
```

函数文件实现了代码的精简操作，用户可以多次调用。MATLAB 编程中，函数名称也不用刻意去声明，因此整个程序的可操作性极强。

3. M文件的创建

脚本文件的创建较容易，可以直接在编辑器窗口中进行代码编写。MATLAB 能快速实现矩阵的基本运算。通过编写函数文件，方便用户直接进行调用。

【例 3-3】编写函数文件。

解：在编辑器窗口中输入以下代码。

```
function main
clc,clear,close
x=[1:4]
mean(x)
end

function y = mean(x,dim)
if nargin==1
    dim = find(size(x)~=1, 1 );
    if isempty(dim), dim = 1; end
    y = sum(x)/size(x,dim);
else
    y = sum(x,dim)/size(x,dim);
end
end
```

单击主界面“编辑器”选项卡“运行”面板中的“运行”按钮运行程序，输出结果如下。

```
x =
     1     2     3     4
ans =
    2.5000
```

从主函数可看出，该函数包括主函数 main 和被调用函数 y = mean(x,dim)，该函数主要用于求解数组的平均值，可以调用多次，达到精简程序的目的。

3.2.2 匿名函数

匿名函数没有函数名，也不是函数 M 文件，只包含一个表达式和输入/输出参数。用户可以在命令行窗口中输入代码，创建匿名函数。匿名函数的创建方法为

```
f = @(input1,input2,…) expression
```

其中，f 为创建的函数句柄。函数句柄是一种间接访问函数的途径，可以使用户调用函数过程变得简单，减少了程序设计中的繁杂，而且可以在执行函数调用过程中保存相关信息。

【例 3-4】当给定实数 x、y 的具体数值后，计算表达式 x^y+3xy 的结果，请用创建匿名函数的方式求解。

解：在命令行窗口中输入以下代码。

```
>> clear,clc
>> Fxy = @(x,y) x.^y + 3*x*y                    %创建一个名为 Fxy 的函数句柄
Fxy =
  包含以下值的 function_handle:
    @(x,y)x.^y+3*x*y
>> whos Fxy                                     %调用 whos 函数查看到变量 Fxy 的信息
  Name      Size           Bytes  Class            Attributes
  Fxy       1x1               32  function_handle
>> Fxy(2,5)                                     %求当 x=2、y=5 时表达式的值
ans =
    62
>> Fxy(1,9)                                     %求当 x=1、y=9 时表达式的值
ans =
    28
```

3.2.3　主函数与子函数

1. 主函数

主函数可以写为脚本文件也可以写为主函数文件，主要是格式上的差异。当写为脚本文件时，直接将程序代码保存为 M 文件即可。如果写为主函数文件，代码主体需要采用函数格式，如下所示。

```
function main
…
end
```

主函数是 MATLAB 编程中的关键环节，几乎所有的程序都在主函数文件中操作完成。

2. 子函数

在 MATLAB 中，多个函数的代码可以同时写到一个 M 函数文件中。其中，出现的第一个函数称为主函数（Primary Function），该文件中的其他函数称为子函数（Sub Function）。保存时所用的函数文件名应当与主函数定义名相同，外部程序只能对主函数进行调用。

子函数的书写规范有如下几条：

（1）每个子函数的第一行是其函数声明行。

（2）在 M 函数文件中，主函数的位置不能改变，但是多个子函数的排列顺序可以任意改变。

（3）子函数只能被处于同一 M 文件中的主函数或其他子函数调用。

（4）在 M 函数文件中，任何指令通过“名称”对函数进行调用时，子函数的优先级仅次于内置函数。

（5）同一 M 文件的主函数、子函数的工作区都是彼此独立的。各个函数间的信息传递可以通过输入/输出变量、全局变量或跨空间指令来实现。

（6）help、lookfor 等帮助指令都不能显示一个 M 文件中的子函数的任何相关信息。

【例 3-5】 M 文件中的子函数示例。

解： 在编辑器窗口中输入以下代码。

```
function F = mainfun (n)
A = 1; w = 2; phi = pi/2;
signal = createsig(A,w,phi);
F = signal.^n;
end
```

```
% ---------子函数---------
function signal = createsig(A,w,phi)
x = 0: pi/3 : pi*2;
signal = A * sin(w*x+phi);
end
```

在命令行窗口中输入

```
>> mainfun (1)
ans =
    1.0000   -0.5000   -0.5000    1.0000   -0.5000   -0.5000    1.0000
```

3. 私有函数与私有目录

所谓私有函数，是指位于私有目录 private 下的 M 函数文件，它的主要性质有如下几条。

（1）私有函数的构造与普通 M 函数完全相同。

（2）关于私有函数的调用：私有函数只能被 private 直接父目录下的 M 文件所调用，而不能被其他目录下的任何 M 文件或 MATLAB 指令窗中的命令所调用。

（3）在 M 文件中，任何指令通过“名称”对函数进行调用时，私有函数的优先级仅次于 MATLAB 内置函数和子函数。

（4）help、lookfor 等帮助指令都不能显示一个私有函数文件的任何相关信息。

3.2.4 重载函数

重载是计算机编程中非常重要的概念，经常用于处理功能类似但变量属性不同的函数。例如实现两个相同的计算功能，输入的变量数量相同，不同的是其中一个输入变量类型为双精度浮点类型，另一个输入变量类型为整型，这时就可以编写两个同名函数，分别处理这两种不同情况。当实际调用函数时，MATLAB 就会根据实际传递的变量类型选择执行哪一个函数。

MATLAB 的内置函数中就有许多重载函数，放置在不同的文件路径下，文件夹通常命名为“@+代表 MATLAB 数据类型的字符”。例如@int16 路径下的重载函数的输入变量应为 16 位整型变量，而@double 路径下的重载函数的输入变量应为双精度浮点类型。

3.2.5 eval、feval 函数

1. eval函数

eval 函数可以与文本变量一起使用，实现有力的文本宏工具。函数调用格式为

```
eval(s)                    %使用 MATLAB 的注释器求表达式的值或执行包含文本字符串 s 的语句
```

【例 3-6】 eval 函数的简单运用示例。

解： 在编辑器窗口中输入以下代码。

```
clear,clc
Array = 1:5;
String = '[Array*2; Array/2; 2.^Array]';
Output1 = eval(String)                          % “表达式”字符串

theta = pi;
eval('Output2 = exp(sin(theta))');              % “指令语句”字符串
```

```
who

Matrix = magic(3)
Array = eval('Matrix(5,:)','Matrix(3,:)')          %“备选指令语句”字符串
% errmessage=lasterr

Expression = {'zeros','ones','rand','magic'};
Num = 2;
Output3 = [];
for i=1:length(Expression)
   Output3 = [Output3 eval([Expression{i},'(',num2str(Num),')'])];  %“组合”字符串
end
Output3
```

运行 M 文件，输出结果如下：

```
Output1 =
   2.0000    4.0000    6.0000    8.0000   10.0000
   0.5000    1.0000    1.5000    2.0000    2.5000
   2.0000    4.0000    8.0000   16.0000   32.0000
Output2 =
   1.0000
您的变量为:
Array    Output1  Output2  String   theta
Matrix =
    8     1     6
    3     5     7
    4     9     2
Array =
    4     9     2
Output3 =
       0        0    1.0000    1.0000    0.8491    0.6787    1.0000    3.0000
       0        0    1.0000    1.0000    0.9340    0.7577    4.0000    2.0000
```

2. feval函数

feval 函数的具体句法形式如下：

```
[y1, y2, …] = feval('FN', arg1, arg2, …)   %用变量 arg1,arg2,…来执行函数 FN 指定的计算
```

说明：①在 eval 函数与 feval 函数通用的情况下（使用这两个函数均可以解决问题），feval 函数的运行效率比 eval 函数高。②feval 函数主要用来构造“泛函”型 M 函数文件。

【例 3-7】feval 函数的简单运用示例。

解：（1）在编辑器窗口中输入以下代码。

```
Array = 1:5;
String = '[Array*2; Array/2; 2.^Array]';
Outpute = eval(String)                          %使用 eval 函数运行表达式
Outputf = feval(String)                         %使用 feval 函数运行表达式，FN 不可以是表达式
```

运行 M 文件，结果如下：

```
Outpute =
    2.0000    4.0000    6.0000    8.0000   10.0000
    0.5000    1.0000    1.5000    2.0000    2.5000
    2.0000    4.0000    8.0000   16.0000   32.0000
错误使用 feval
函数名称 '[Array*2; Array/2; 2.^Array]' 无效。
```

（2）继续在编辑器窗口中输入以下代码。

```
j = sqrt(-1);
Z = exp(j*(-pi:pi/100:pi));
eval('plot(Z)');
set(gcf,'units','normalized','position',[0.2,0.3,
0.2,0.2])
title('Results by eval');axis('square')
figure
set(gcf,'units','normalized','position',[0.2,0.3,
0.2,0.2])
feval('plot',Z);          %feval 函数中的 FN 只接受函数名，
                          %不接受表达式
title('Results by feval');axis('square')
```

运行 M 文件，结果如图 3-6 所示。

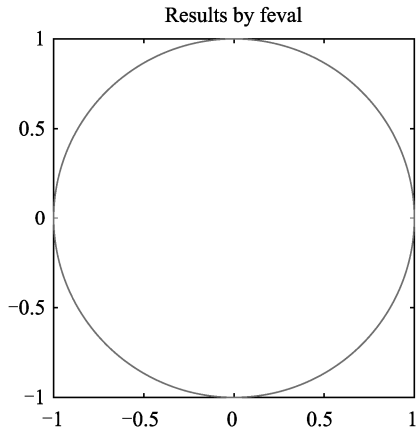

图 3-6　文件的运行结果

3.2.6　内联函数

内联函数（Inline Function）的属性和编写方式与普通函数文件相同，但相对来说，内联函数的创建简单得多。其调用格式为

```
inline('CE')                    %将字符串表达式 CE 转换为输入变量自动生成的内联函数
```

本语句将自动由字母和数字组成的连续字符辨识为变量，预定义变量名（如圆周率 pi）、常用函数名（如 sin、rand）等不会被辨识，连续字符后紧接左括号的，也不会被识别（如 array(1)）。

```
inline('CE', arg1, arg2, …) %把字符串表达式 CE 转换为 arg1、arg2 等指定的输入变量的内联函数
```

本语句创建的内联函数最为可靠，输入变量的字符串可以随意改变，但是由于输入变量已经规定，因此生成的内联函数不会出现辨识失误。

```
inline('CE', n)      % 把字符串表达式 CE 转化为 n 个指定的输入变量的内联函数
```

本语句对输入变量的字符是有限制的，其字符只能是 x,P1,…,Pn 等，其中 P 一定为大写字母。

说明：

（1）字符串 CE 中不能包含赋值符号“=”。

（2）内联函数是沟通 eval 和 feval 两个函数的桥梁，只要是 eval 函数可以操作的表达式，都可以通过 inline 指令转化为内联函数，这样，内联函数总是可以被 feval 函数调用。MATLAB 中的许多内置函数就是通过被转换为内联函数，从而具备了根据被处理的方式不同而变换不同函数形式的能力。

MATLAB 中关于内联函数的属性的相关指令如表 3-1 所示，读者可以根据需要使用。

表 3-1　内联函数属性指令集

指令句法	功　能
class(inline_fun)	提供内联函数的类型
char(inline_fun)	提供内联函数的计算公式
argnames(inline_fun)	提供内联函数的输入变量
vectorize(inline_fun)	使内联函数适用于数组运算的规则

【例 3-8】内联函数的简单运用示例。

解：（1）示例说明：内联函数的第一种创建格式是使内联函数适用于“数组运算”。

在命令行窗口中输入：

```
>> Fun1=inline('mod(12,5)')
Fun1 =
    内联函数:
    Fun1(x) = mod(12,5)
>> Fun2=vectorize(Fun1)
Fun2 =
    内联函数:
    Fun2(x) = mod(12,5)
>> Fun3=char(Fun2)
Fun3 =
    'mod(12,5)'
```

（2）示例说明：第一种内联函数创建格式的缺陷在于不能使用多标量构成的向量进行赋值，此时可以使用第二种内联函数创建格式。

继续在命令行窗口中输入：

```
>> Fun4 = inline('m*exp(n(1))*cos(n(2))'), Fun4(1,[-1,pi/2])
Fun4 =
     内联函数:
     Fun4(m) = m*exp(n(1))*cos(n(2))
错误使用 inline/subsref (line 14)
内联函数的输入数目太多。
>> Fun5 = inline('m*exp(n(1))*cos(n(2))','m','n'), Fun5(1,[-1,pi/2])
Fun5 =
    内联函数:
    Fun5(m,n) = m*exp(n(1))*cos(n(2))
ans =
    2.2526e-017
```

（3）示例说明：产生向量输入、向量输出的内联函数。

继续在命令行窗口中输入：

```
>> y = inline('[3*x(1)*x(2)^3;sin(x(2))]')
y =
     内联函数:
     y(x) = [3*x(1)*x(2)^3;sin(x(2))]
>> Y = inline('[3*x(1)*x(2)^3;sin(x(2))]')
```

```
Y =
     内联函数:
     Y(x) = [3*x(1)*x(2)^3;sin(x(2))]
>> argnames(Y)
ans =
  1×1 cell 数组
    {'x'}
>> x=[10,pi*5/6];y=Y(x)
y =
  538.3034
    0.50000
```

（4）示例说明：用最简练的格式创建内联函数，内联函数可被 feval 函数调用。

继续在命令行窗口中输入：

```
>> Z=inline('floor(x)*sin(P1)*exp(P2^2)',2)
Z =
   内联函数:
    Z(x,P1,P2) = floor(x)*sin(P1)*exp(P2^2)
>> z = Z(2.3,pi/8,1.2), fz = feval(Z,2.3,pi/8,1.2)
z =
    3.2304
fz =
    3.2304
```

3.2.7 向量化和预分配

1. 向量化

要想让 MATLAB 高速地工作，需要在 M 文件中对算法进行向量化处理。其他程序语言多采用 for 或 do 循环，而 MATLAB 则采用向量或矩阵进行运算。下面的代码用于创建一个算法表。

```
x = 0.01;
for k = 1:1001
   y(k) = log10(x);
   x = x + 0.01;
end
```

代码的向量化实现如下：

```
x =0 .01:0.01:10;
y = log10(x);
```

对于更复杂的代码，矩阵化处理不总是那么明显。当需要提高运算速度时，应该想办法将算法向量化。

2. 预分配

若一条代码不能向量化，则可以通过预分配输出结果空间来保存其中的向量或数组，以加快 for 循环。下面的代码用 zeros 函数把 for 循环产生的向量预分配，这使得 for 循环的执行速度显著加快。

```
r = zeros(32,1);
for n = 1:32
    r(n) = rank(magic(n));
end
```

若上述代码中没有使用预分配，则 MATLAB 的注释器利用每次循环扩大 ***r*** 向量。向量预分配排除了该步骤以使执行加快。

一种以标量为变量的非线性函数称为“函数的函数”，即以函数名为自变量的函数。这类函数包括求零点、最优化、求积分和常微分方程等。

【例 3-9】简化的 humps 函数的简单运用示例（humps 函数可在路径 MATLAB\demos 下获得）。

解：在编辑器窗口中输入以下代码。

```
clear,clc
a = 0:0.002:1;
b = humps(a);
plot(a,b)                                                    %画出图像
function b = humps(x)
b =1./((x-.3).^2 +.01) + 1./((x-.9).^2 +.04)-6;              %在区间[0,1]求此函数的值
end
```

运行程序后输出如图 3-7 所示的图形。

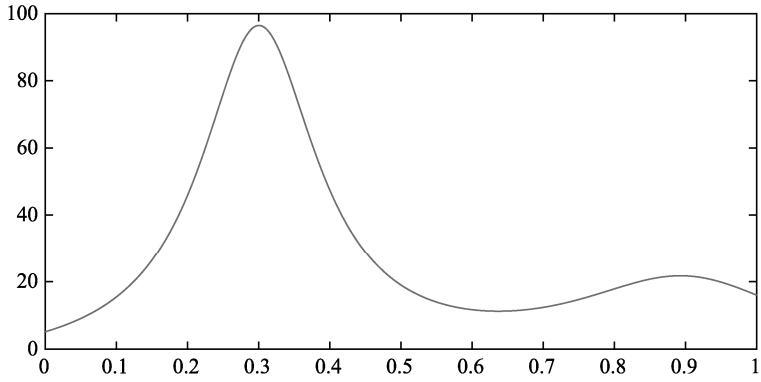

图 3-7　运行结果

图形表明函数约在 x=0.6 附近有局部最小值。接下来用函数 fminsearch 可以求出局部最小值及此时 x 的值。函数 fminsearch 的第一个参数是函数句柄，第二个参数是此时 x 的近似值。

在命令行窗口中输入：

```
>> p = fminsearch(@humps,.5)
p =
    0.6370
>> humps(p)                                          %求出此局部最小值
ans =
    11.2528
```

3.2.8　函数参数传递

MATLAB 编写函数，在函数文件头需要写明输入和输出变量，方能构成一个完整的可调用函数。在主函数中调用时，通过满足输入关系，选择输出的变量，也就是相应的函数参数传递，MATLAB 将这些实际值传回给相应的形式参数变量。每个函数调用时变量之间互不冲突，均有自己独立的函数空间。

1. 函数的直接调用

例如，求解变量的均值，可编写函数如下：

```
function y = mean(x,dim)
```

其中，x 为输入的变量；dim 为数据的维数，默认为 1。直接调用该函数即可得到相应的均值解。MATLAB 在输入变量和输出变量的对应上，优选第一变量值作为输出变量。当然也可以不指定输出的变量，MATLAB 默认用 ans 表示输出变量对应的值。

MATLAB 中可以通过 nargin 和 nargout 函数来确定输入和输出的变量的个数，可以避免输入一些参数，从而提高程序的可执行性。

【例 3-10】函数的直接调用示例。

解：（1）在编辑器窗口中编写 mean 函数。

```
function y = mean(x,dim)
    if nargin==1
      % Determine which dimension SUM will use
      dim = find(size(x)~=1, 1 );
      if isempty(dim)
    dim = 1;
      end
       y = sum(x)/size(x,dim);
    else
      y = sum(x,dim)/size(x,dim);
    end
end
```

若 nargin=1，系统默认 dim=1，则根据 y = sum(x)/size(x,dim)进行求解；若 dim 为指定的一个值，则根据 y = sum(x,dim)/size(x,dim)进行求解。

（2）在命令行窗口中输入以下命令。

```
>> clear,clc
>> format short
>> x=1:4;
>> mean(x)
ans =
    2.5000
>> mean(x,1)
ans =
     1     2     3     4
>> mean(x,2)
ans =
    2.5000
>> mean(x,3)
ans =
     1     2     3     4
>> mean(x,4)
ans =
     1     2     3     4
```

```
>> a=mean(x)
a =
    2.5000
```

由上述分析可知，对于该均值函数，dim 的赋值需要匹配矩阵的维数，当 dim 为 1 时求解的为列平均，当 dim=2 时求解的为行平均。如果没有指定输出的变量，则 MATLAB 系统默认用 ans 变量替代；如果指定输出变量，则显示输出对应的子母变量的值。

2. 全局变量

通过全局变量可以实现 MATLAB 工作区变量空间和多个函数的函数空间共享，这样，多个使用全局变量的函数和 MATLAB 工作区共同维护这一全局变量，任何一处对全局变量的修改，都会直接改变此全局变量的取值。

全局变量在大型的编程中经常用到，特别是在 App 设计中，对于每个按钮功能模块下的运行程序，则需要调用前面对应的输出和输入变量，这时候需要对应的全局变量，全局变量在 MATLAB 中用 global 表示，指定全局变量后，该变量能够分别在私自函数、子函数、主函数中使用，全局变量在整个程序设计阶段基本保持一致。

在应用全局变量时，通常在各个函数内部通过 global 语句声明；在命令窗口或脚本 M 文件中也要先通过 global 声明，然后进行赋值。

【例 3-11】全局变量应用示例。

解：在编辑器窗口中输入以下代码。

```
clear,clc
global a
a =2;
x=3;
y=djb(x)

function y=djb(x)
    global a
    y= a*(x^2);
end
```

运行程序输出结果如下：

```
y =
    18
```

从程序运行结果可知，全局变量只需要在主函数中进行声明，然后使用 global 在主函数和子函数中分别进行定义即可，最后调用对应的函数，即可完成函数的计算求解。

3.3　程序控制

与 C、C++等语言相似，MATLAB 具有很多函数程序编写句柄，采用这些判别语句可以轻松地进行程序书写。具体的程序控制语句包括：分支控制语句（if 结构和 switch 结构）、循环控制语句（for 循环、while 循环、continue 语句和 break 语句）和程序终止语句（return 语句）。

3.3.1 分支控制语句

MATLAB 程序结构一般可分为顺序结构、循环结构、分支结构 3 种。顺序结构是指按顺序逐条执行，循环结构与分支结构都有其特定的语句，这样可以增强程序的可读性。在 MATLAB 中常用的分支结构包括 if 结构和 switch 结构。

1. if分支结构

如果在程序中需要根据一定条件来执行不同的操作时，可以使用条件语句，在 MATLAB 中提供 if 分支结构，或者称为 if-else-end 语句。

根据不同的条件，if 分支结构有多种形式。其中，最简单的用法是：如果条件表达式为真，则执行语句 1，否则跳过该组命令。

if 结构是一个条件分支语句，若满足表达式的条件，则往下执行；若不满足，则跳出 if 结构。else if 表达式 2 与 else 为可选项，这两条语句可依据具体情况取舍。

if 语法结构如下所示。

```
if  表达式 1
    语句 1
    else if 表达式 2 （可选）
        语句 2
    else （可选）
        语句 3
    end
end
```

注意：

（1）每一个 if 都对应一个 end，即有几个 if，就应有几个 end。

（2）if 分支结构是所有程序结构中最灵活的结构之一，可以使用任意多个 else if 语句，但是只能有一个 if 语句和一个 end 语句。

（3）if 语句可以相互嵌套，可以根据实际需要将各个 if 语句进行嵌套，从而解决比较复杂的实际问题。

【例 3-12】思考下列程序及其运行结果，说明原因。

解：在 MATLAB 命令窗口中输入以下程序。

```
clear,clc
a=100;
b=20;
if a<b
   fprintf ('b>a')
else
   fprintf ('a>b')
end
```

运行后得到：

```
a>b
```

在程序中，用到了 if-else-end 的结构，如果 a<b，则输出 b>a；反之，输出 a>b。由于 a=100，b=20，比较可得结果 a>b。

在分支结构中，多条语句可以放在同一行，但语句间要用“;”分开。

2. switch分支结构

MATLAB 中的 switch 分支结构和 C 语言中类似，适用于条件多而且比较单一的情况，类似于一个数控的多个开关。其一般的语法调用方式如下：

```
switch  表达式
case 常量表达式 1
    语句组 1
    case 常量表达式 2
        语句组 2
        …
    otherwise
        语句组 n
end
```

其中，switch 后面的表达式可以是任何类型，如数字、字符串等。当表达式的值与 case 后面常量表达式的值相等时，就执行这个 case 后面的语句组；如果所有的常量表达式的值都与这个表达式的值不相等，则执行 otherwise 后的语句组。

表达式的值可以重复，在语法上并不错误，但是在执行时，后面符合条件的 case 语句将被忽略。各个 case 和 otherwise 语句的顺序可以互换。

【例 3-13】输入一个数，判断它能否被 5 整除。

解：在 MATLAB 中输入以下程序。

```
clear,clc
n=input('输入 n=');                    %输入 n 值
switch mod(n,5)                        %mod 是求余函数，余数为 0，得 0；余数不为 0，得 1
case 0
   fprintf ('%d 是 5 的倍数',n)
otherwise
   fprintf('%d 不是 5 的倍数',n)
end
```

运行后得到结果为

```
输入 n=12
12 不是 5 的倍数>>
```

在 switch 分支结构中，case 命令后的常量表达式不仅可以为一个标量或者字符串，还可以为一个元胞数组。如果常量表达式是一个元胞数组，则 MATLAB 将把 switch 后的表达式的值和该元胞数组中的所有元素进行比较；如果元胞数组中某个元素和表达式的值相等，则 MATLAB 认为比较结构为真。

3.3.2　循环控制语句

在 MATLAB 程序中，循环结构主要包括 while 循环结构和 for 循环结构两种。下面对两种循环结构做详细介绍。

1. while循环结构

除了分支结构之外，MATLAB 还提供多个循环结构。和其他编程语言类似，循环语句一般用于有规律

地重复计算。被重复执行的语句称为循环体，控制循环语句流程的语句称为循环条件。

在 MATLAB 中，while 循环结构的语法形式如下：

```
while 逻辑表达式
    循环语句
end
```

while 结构依据逻辑表达式的值判断是否执行循环体语句。若表达式的值为真，则执行循环体语句一次，在反复执行时，每次都要进行判断。若表达式为假，则程序执行 end 后的语句。

为了避免因逻辑上的失误，而陷入死循环，建议在循环体语句的适当位置加 break 语句，以便程序能正常执行。

while 循环也可以嵌套，其结构如下：

```
while 逻辑表达式 1
    循环体语句 1
while 逻辑表达式 2
    循环体语句 2
end
循环体语句 3
end
```

【例 3-14】请设计一段程序，求 1 ~ 100 的偶数和。

解：在 MATLAB 命令窗口输入以下程序。

```
clear,clc
x=0;                                    %初始化变量 x
sum=0;                                  %初始化 sum 变量
   while x<101                          %当 x<101 执行循环体语句
     sum=sum+x;                         %进行累加
     x=x+2;
   end                                  %while 结构的终点
sum                                     %显示 sum
```

运行后得到的结果为

```
sum =
       2550
```

【例 3-15】请设计一段程序，求 1 ~ 100 的奇数和。

解：在 MATLAB 命令窗口输入以下程序。

```
clear,clc
x=1;                                    %初始化变量 x
sum=0;                                  %初始化 sum 变量
   while x<101                          %当 x<101 执行循环体语句
     sum=sum+x;                         %进行累加
     x=x+2;
   end                                  %while 结构的终点
sum                                     %显示 sum
```

运行后得到的结果为

```
sum =
       2500
```

2. for循环结构

在 MATLAB 中，另外一种常见的循环结构是 for 循环，常用于已知循环次数的情况，其语法规则如下所示。

```
for ii=初值:增量:终值
    语句 1
    ……
    语句 n
end
```

如果 ii=初值:终值，则增量为 1。初值、增量、终值可正可负，可以是整数，也可以是小数，只需符合数学逻辑。

【例 3-16】请设计一段程序，求 1+2+…+100 的和。

解：程序设计如下：

```
clear,clc
sum=0;                                  %设置初值(必须要有)
for ii=1:100;                           %for 循环，增量为 1
    sum=sum+ii;
end
sum
```

运行后得到结果为

```
sum =
       5050
```

【例 3-17】比较以下两个程序的区别。

解：MATLAB 程序 1 设计如下：

```
for ii=1:100;                           %for 循环，增量为 1
    sum=sum+ii;
end
sum
```

运行后得到的结果为

```
sum =
      10100
```

程序 2 设计如下：

```
clear,clc
for ii=1:100;                           %for 循环，增量为 1
    sum=sum+ii;
end
sum
```

运行结果：

```
错误使用 sum
输入参数的数目不足。
```

在一般的高级语言中，若变量没有设置初值，则程序会以 0 作为其初始值，然而这在 MATLAB 中是不允许的。所以，在 MATLAB 中应给出变量的初值。

程序 1 没有 clear，则程序可能会调用内存中已经存在的 sum 值，其结果就成了 sum =10100。程序 2 与例 3-16 的差别是少了 sum=0，由于程序中有 clear 语句，因此会出现错误信息。

注意： while 循环和 for 循环都是比较常见的循环结构，但是两个循环结构还是有区别的。其中最明显的区别在于，while 循环的执行次数是不确定的，而 for 循环的执行次数是确定的。

3.3.3 其他控制语句

在使用 MATLAB 设计程序时，经常遇到提前终止循环、跳出子程序、显示错误等情况，因此需要其他的控制语句来实现上面的功能。在 MATLAB 中，对应的控制语句有 continue、break、return 等。

1. continue命令

continue 语句通常用于 for 或 while 循环体中，其作用就是终止一轮循环的执行，也就是说它可以跳过本轮循环中未被执行的语句，去执行下一轮的循环。下面使用一个简单的实例，说明 continue 命令的使用方法。

【例 3-18】请思考下列程序及其运行结果，说明原因。

解： 在 MATLAB 中输入以下程序。

```
clear,clc
a=3;
b=6;
for ii=1:3
   b=b+1
   if ii<2
     continue
   end                                  %if 语句结束
   a=a+2
end                                     %for 循环结束
```

运行后得到结果为

```
b =
     7
b =
     8
a =
     5
b =
     9
a =
     7
```

当 if 条件满足时，程序将不再执行 continue 后面的语句，而是开始下一轮的循环。continue 语句常用于循环体中，与 if 一同使用。