第四章 M 文件和程序设计

4.1 程序控制语句

4.1.1 分支控制

分支控制语句有 if 和 switch 两种语句

1. if 结构

```
if expression
    commands
end
```

【说明】：当表达式 expression 的值为真，则执行 commands 语句组，否则跳过 commands 语句组，执行 end 之后的语句。

2. if-else 结构

```
if expression
    commands1
else
    commands2
end
```

【说明】：如果表达式 expression 的值为真，则执行语句组 commands1，然后跳过语句组 commands2 向下执行；若表达式 expression 的值为假，则跳过语句组 commands1 而执行语句组 commands2。

3. if-elseif-else 结构

```
if expression_1
    commands_1
elseif  expression_2
    commands_2
    ……
elseif  expression_n
    commands_n
else
    commands_el
end
```

【说明】：如果表达式 expression_1 的值为真，则执行语句组 commands_1，若表达式 expressions_1 的值为假，则判断 expression_2 的值，若为真，则执行语句组 commands_2，否则向下继续判断表达式。如果所有表达式都为假，则执行语句组 commands_el。

例 4.1.1 有分段函数 $f(x)=\begin{cases}3x+4 & x<-1\\ e^{-x} & -1\le x\le 1\\ \sin x+\cos x & x>1\end{cases}$，编程输入 x 的值，计算并显示函数值。

解：运行下面语句，输入 x 的值，观察运行结果

```
x=input('请输入自变量值：  ');
if x<-1
```

```
        str='3x+4';
        y=3*x+4;
    elseif x<=1
        str='exp(-x)';
        y=exp(-x);
    else
        str='sin(x)+cos(x)';
        y=sin(x)+cos(x);
    end
    disp([str,blanks(4),num2str(y)])
```

4. switch-case 结构

```
switch exp_const
   case   value_1
            commands_1
   case   value_2
            commands_2
    ---
   case   value_n
            commands_n
   otherwise
            commands_ow
end
```

【说明】

1. 表达式 exp_const 的值和哪一个 case 语句后的值相等，就执行那个 case 语句下的语句组。如果 exp_const 和所有值都不相等，就执行 otherwise 后的语句组 commands_ow。

2. switch 后的 exp_const 表达式可以为标量或者字符串。对于标量形式，用关系运算符“==”比较，对于字符串形式，用函数 strcmp 比较。

3. case 语句后的 value_n 值可以是标量、字符串，也可以是细胞数组。如果是细胞数组，MATLAB 会把表达式 exp_const 的值和细胞数组中的每个元素比较，只要有一个元素和 exp_const 值相同，比较结果就为相等。

4.1.2 循环控制

1. for 循环

```
for   x=Array
   commands
end
```

【说明】: x 称为循环变量，commands 称为循环体。循环的执行步骤是：循环变量从左到右依次取 Array 数组的一列，每取一列就执行一次循环体，循环体被执行的次数为 Array 的列数。for 循环一般用于循环次数已知的程序流程控制。

2. while 循环

```
while expression
   commands
end
```

【说明】: 先判断表达式 expression 的值，如果其值为真，则执行 commands 循环体；执行完循环体后，继续判断表达式 expression 的值，直到表达式 expression 的值为假，结束循环。while 循环通常用在循环次

数未知的情况下，而且要在循环体修改循环表达式的值，否则容易造成死循环。

3. break

从循环体中跳出，并使循环结束。

例 4.1.2 输入物品的标签号码，显示物品的种类。其中标签号码为 1、5、7 的物品为食品，号码为 10 到 19 的物品为办公用品，号码为 20、24、28 的为音像制品。要求不断从键盘输入标签号码，输入一个号码显示一次物品种类，输入错误号码要给出提示，输入号码 0 则结束输入。

解：运行下列语句，输入标签号码，观察运行结果。

```
code1= [{1},{5},{7}];   code3=[{20},{24},{28}];   %构造细胞数组，分类编号
for i=10:19
    code2{i}=i;
end
while 1
    a=input('输入标签号码: ');
    switch a
        case   code1
            disp('食品');
        case   code2
            disp('办公用品');
        case   code3
            disp('音像制品');
        case   0                                %输入为 0，退出循环
            break;
        otherwise
            disp('错误代码');
    end
end
```

4.1.3 异常检测

MATLAB 有异常处理机制，即可以检测某些语句的执行是否正确，如果发生运行错误，MATLAB 的异常处理机制会捕捉到这个错误，并跳转到用户指定的错误处理程序。异常处理是通过 try-catch 语句实现的，其语法如下：

```
try
  commands1
catch
  handles
end
```

【说明】：执行语句组 commands 1，当语句组 commands 1 执行发生错误时，跳转到错误处理语句组 handles。

例 4.1.3 运行下列语句，理解异常处理机制。

```
A=[1,1];B=[1,1];
try
    C=A*B;
catch
    disp('矩阵乘法错误');
    C=[];
```

end

lasterr

矩阵乘法错误

ans =

Error using ==> mtimes

Inner matrix dimensions must agree.

4.1.4 人机交互控制

1. **input**

【调用格式】

v=input('message') 将用户键入的内容赋给变量 v，messages 是显示信息

v=input('message', 's') 将用户键入的内容作为字符串赋给变量 v

2. **keyboard**

从键盘读入多个 MATLAB 指令，直到用户输入 return 指令才返回。

3. **yesinput**

【调用格式】

v=yesinput('Prompt', default, possible)

【说明】Prompt 是显示在屏幕上的提示信息；default 是缺省值，即如果没有键盘输入时变量 v 赋值为 default；possible 是变量 v 可能接受的值。

4. **pause**

【调用格式】

pause 暂停执行文件，等用户输入任意按键后才继续执行

pause(n) 暂停 n 秒后，程序继续执行

5. **disp**

【调用格式】

disp(X) 显示数组 X 的内容，但是不显示数组 X 的名字

4.1.5 其他程序流控制语句

1. **返回指令**

return 强制结束函数或者命令的调用，将控制权交给主调函数或者命令窗口。

2. **出错处理**

error('message') 显示出错信息 message，中止程序运行

errortrap 发生错误后，程序继续执行或退出的状态切换

lasterr 显示 MATLAB 给出的最新的出错信息，并中止程序运行

3. **警告处理**

warning('message') 显示警告信息 message，程序继续运行

lastwarn 显示 MATLAB 最新给出的警告信息，程序自动运行

4.2 命令文件和函数文件

用户可以在MATLAB命令窗口输入并运行MATLAB的指令，这种方法适合于指令较少、不复杂且不经常被修改、运行的情况下。如果需要编写复杂的程序，需要大量、复杂的指令才能实现，直接在命令窗口

输入指令的方法十分不方便。用户可以把需要运行的MATLAB指令保存在一个以“.m”为扩展名的文件中，通过在命令窗口键入这个文件的名字来运行文件中的MATLAB指令，我们称这些文件为m文件。按照m文件的组成和特点，可以分为命令文件和函数文件。

4.2.1 命令文件

有关命令文件的描述如下：

(1) 命令文件没有输入参数也没有输出参数，只是一些 MATLAB 命令和函数的组合。

(2) 命令文件可以操作基本工作空间（Base Workspace）的变量，也可以生成新的变量。命令文件执行结束后新变量将保存在基本工作空间中，不会被自动清除。

(3) 命令文件是用“.m”为扩展名的文件，只要命令文件在搜索路径上，在命令窗口键入文件名就可以运行命令文件。

例 4.2.1 编写命令文件 test_com.m，用于求解小于 1000 且为 2 的整数次幂的正整数。

解：按照如下步骤编写并运行命令文件

（1） 选择命令窗口的【File|New|M-File】菜单项，会调出文件编辑窗口

（2） 在文件编辑窗口中键入如下的 MATLAB 语句

```
%test_com.m
F(1)=2;
k=1;
while F(k)<1000
    F(k+1)= 2*F(k);
    k=k+1;
end
F(end)=[];
k=k-1;
```

（3） 建立“C:\work”子目录，将新编辑的文件保存为“C:\work\ test_com.m”。

（4） 选择【File】|Set Path】菜单，弹出路径设置对话框，加入刚刚建立的“C:\work”路径到所有搜索路径的最前端。

（5） 在命令窗口里面键入如下指令，就可以得到以下的运行结果。

```
>> clear
>> test_com
>> F,k
F =
     2     4     8    16    32    64   128   256   512
k =
     9
```

4.2.2 函数文件

函数文件比命令文件更加灵活，它能够更好地实现复杂问题的功能模块划分。函数文件如同一个“黑箱子”，调用者仅仅需要把输入变量传递给函数，就会得到函数的输出变量，即函数的运行结果。对函数文件有如下的描述：

1. 函数文件要在第一行用 function 关键字来显式定义；
2. 函数文件有输入变量和输出变量；
3. 函数文件的名字和第一行所定义的函数名相同；

4. 可以用比函数定义中数目少的输入变量和输出变量来调用函数；

5. 函数文件内部定义的变量属于临时变量，只有函数运行期间才被生成，函数运行结束后，这些临时变量会被 MATLAB 系统自动删除。

4.2.3 函数文件的组成

MATLAB 的函数 M 文件通常由以下几个部分组成：

1. 函数定义行

函数 M 文件的第一行用关键字“function”把 M 文件定义为一个函数，指定函数的名字，同时定义了函数的输入变量和输出变量。输入变量的定义用圆括号（），如果有多个输入变量则用逗号分隔；输出变量的定义用中括号[]，如果有多个输出变量则用逗号分隔。

2. H1 行

所谓 H1 行指帮助文本的第一行，它紧跟在定义行之后并以“%”符号开头，用于概括说明函数的功能。在命令窗口用 lookfor 命令时将显示函数的 H1 行。

3. 函数帮助文本

帮助文本指位于 H1 行之后函数体之前的说明文本，它同样以“%”符号开头，一般用来比较详细地介绍函数的功能、用法以及函数的修改记录。在命令窗口用 help 命令时将显示函数的 H1 行和所有帮助文本。

4. 函数体

是函数的主体部分，函数的功能是通过函数体实现的。函数体可以包括所有的 MATLAB 合法命令、函数和流程控制语句。

5. 注释

除了函数开始处独立的帮助文本外，还可以在函数体中添加对语句的注释。注释必须以“%”符号开头，MATLAB 在解释执行 M 文件时把每一行中“%”后面的全部内容作为注释不进行解释执行。编程时要对关键的语句或者变量写注释，便于程序的阅读和维护。

例 4.2.2 编写函数文件求 $f(n)=1+2+...+n$。

解：按如下步骤操作编写函数文件，并测试函数文件。

（1）新建 m 文件，在文件编辑器中键入如下内容：

```
function    [f]=Addn(n)
%计算 1+2+…+n
%输入变量：n    累加次数
%输出变量：f    计算结果
x=1:n;
f=sum(x);                    %sum 为求和函数
```

（2）将文件保存为“C:\work\Addn.m”

（3）在命令窗口中输入下列指令，测试函数文件。

```
>> s=Addn(3)
s =
  6
>> help Addn
计算 1+2+…+n
输入变量：n    最后一个累加数值
输出变量：f    计算结果
```

4.2.4 函数的输入和输出变量

1. 输入变量和输出变量的检测

MATLAB 提供了检测输入变量和输出变量的函数，其格式和功能如下：

n = nargin	用于函数内，返回实际输入变量的个数
n = nargin('fun')	获取 fun 函数的声明的输入变量个数
n = nargout	用于函数体，返回实际输出变量的个数
n = nargout('fun')	获取 fun 函数的声明的输出变量个数
vname=inputname(n)	用于函数内，返回第 n 个输入变量的实际调用变量名字

2. 数目可变的输入变量和输出变量

MATLAB 支持输入变量和输出变量数目可变的函数。用户可以使用 MATLAB 提供的两个指令来自己编写参数数目可变的函数。

【调用格式】

varargin	数目可变的输入变量列表
varargout	数目可变的输出变量列表

【说明】

1. 编写参数数目可变的函数时，函数定义行的“数目可变的变量”要放在“普通变量”之后。
2. varargin 的工作过程：

(1) varargin 是一个细胞数组，里面放置的是“数目可变的输入变量”。

(2) 函数被调用时，输入变量的传递规则是：实际输入变量依次逐个传递给函数定义的输入变量列表中的“普通输入变量”；然后把剩余的实际输入变量依次传递给 varargin 细胞数组中的细胞。

(3) varargin 细胞数组中的细胞作为一个“普通输入变量”来使用。

3. varargout 的工作过程和 varargin 类似，只是其对应的是函数的输出变量。

例 4.2.3 用参数可变的方法定义一个绘制圆环的函数。其中第一个输入变量为基圆半径，是必选的参数；第二个输入变量为内圆半径，为可选参数；还可以输入可选的图形属性控制字符串。如果有输出变量，则不画图，只返回绘图数据；如果没有输出变量，则绘制图形。

解：（1）编写 DrawRing.m 文件，文件内容如下：

```
function varargout=DrawRing(r,varargin)
%绘制圆环
%调用格式：
%   [x1,y1,x2,y2]=DrawRing(r,r2,'PropertyName1','PropertyValue1',...)
%输入变量：
%    r              基圆半径(必须的输入变量)
%    r2             内圆半径(可以省略)
%    PropertyName1  指定绘图属性名称(可以省略，也可以设置多组属性)
%    PropertyValue1 指定绘图属性 PropertyValue1 的取值
%输出变量：
%  (1)无输出时，绘制圆或者圆环
%  (2)有输出时，不绘图
%        (x1,y1)和(x2,y2)分别为基圆和内圆的坐标点
vin=length(varargin);
Nin=vin+1;
error(nargchk(1,Nin,nargin));
if nargout>6
   error('输出变量数目超过了函数定义');
```

```
end
t=0:2*pi/50:2*pi;
x=r*exp(i*t);
if nargout==0
   switch Nin
      case 1
         plot(x, 'b');
      case 2
         r2=varargin{1};
         x2=r2*exp(i*t);
         plot(x, 'b'); hold on; plot(x2, 'b'); hold off;
      otherwise
         r2=varargin{1};
         x2=r2*exp(i*t);
         plot(x,varargin{2:end});hold on;
         plot(x2,varargin{2:end});hold off;
   end
   axis image
else
   varargout{1}=real(x); varargout{2}=image(x);
   varargout{3}=[]; varargout{4}=[];
   if Nin>1
      r2=varargin{1};
      x2=r2*exp(i*t);
      varargout{3}=real(x2); varargout{4}=image(x2);
   end
end
```

（2）在命令窗口输入下面语句，测试 DrawRing 函数，绘图结果如图 4.2.1 所示。

```
>> clear
>> r1=4;r2=3;
>> subplot(1,3,1); DrawRing(r1,r2);
>> subplot(1,3,2); DrawRing(r1,r2, 'r-o');
>> subplot(1,3,3); DrawRing(r1,r2, 'LineWidth',5, 'Color',[1,0.3,1]);
```

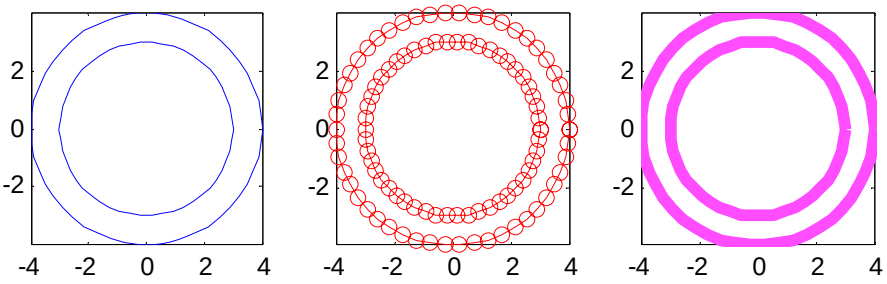

图 4.2.1 DrawRing 绘图结果

4.2.5 有关工作空间

1. 工作空间

有了函数文件后，工组空间会变得复杂起来。MATLAB 将工作空间分成 2 类：

(1) 基本工作空间

基本工作空间是 MATLAB 启动后自动创建的，只有关闭了 MATLAB 后基本工作空间才会被删除。

(2) 函数工作空间

函数工作空间是函数文件运行时自动创建的工作空间，函数工作空间是临时的，当函数运行完毕后，函数工作空间会被系统自动撤销。函数工作空间内保存了函数内部定义或者运算生成的临时变量，这些临时变量在函数执行完成后就不会存在了。

2. 局部变量和全局变量

(1) 局部变量

存在于函数工作空间的临时变量（即函数内部定义的变量）称为局部变量。局部变量只能被定义它的函数使用。

在函数的定义中，其输入变量和输出变量就是局部变量，即函数的输入变量和输出变量只能在本函数内部被使用。

(2) 全局变量

通过 global 关键字可以定义全局变量。全局变量可以被所有工作空间共享，即所有工作空间都可以访问全局变量，但是前提是使用之前要先在工作空间内用 global 关键字声明全局变量。建议在函数的开始处定义或声明全局变量，全局变量的名字一般采用大写字母命名。

3. 跨工作空间的变量赋值

有些时候我们需要跨工作空间给变量赋值，MATLAB 提供了相关的函数。

【调用格式】

assignin('ws', 'var', val)

【说明】：把当前工作空间的变量 val 赋值给'ws'工作空间的名为'var'的变量。

其中，'ws'可以取： 'base' 表示基本工作空间

'caller' 表示主调函数工作空间

4.2.6 子函数和私有函数

如果一个项目比较复杂，就需要编写大量的函数文件，这些函数文件通常都是该工程中专用的，其他程序很少会用到。我们应该限制这些自编函数可以被调用的范围，避免在 MATLAB 基本工作空间中出现过多的标识符，尽量避免在一个工作空间内出现同名函数。为了限制函数的使用范围，MATLAB 支持子函数和私有函数。

1.子函数

MATLAB 允许一个函数文件内定义多个函数。其中，第一个定义的函数称为主函数，其他的函数称为子函数。关于主函数和子函数的描述如下：

(1) 每个文件的第一行定义的是主函数；

(2) 只有主函数可以被其他程序调用；

(3) 函数文件的名字必须和主函数相同；

(4) 子函数只能被处在同一个文件中的主函数或者其他子函数调用；

(5) 主函数和各个子函数的工作空间是彼此独立的，即每个函数拥有自己独立的工作空间。

2.私有函数

私有函数是在函数 M 文件所在目录的 private 子目录中的函数 M 文件，其性质如下：

(1) 私有函数只能被 private 的直接父目录中定义的 M 文件调用，其他目录的 M 文件或者命令窗口无

权调用私有函数。

(2) 私有函数的定义和构成与普通函数相同。

例 4.3.2 编写函数文件求 $f(n)=\sum_{k=1}^{n}k!=1!+2!+...+n!$。

解：（1）编辑文件 AddJc.m 的内容如下

```
function  r=AddJc(n)
%AddJc.m   函数 AddJc（n）求 1 到 n 的阶乘的和
% n 为输入参数，Jc 为子函数
r=0; k=1;
while k<=n
    r=r+Jc(k);
    k=k+1;
end
function s=Jc(m)
%Jc(m)求解 m 的阶乘，是文件内部函数（子函数）
s=1; k=1;
while  k<=m
    s=s*k;
    k=k+1;
end
```

（2）在命令窗口输入如下语句测试函数

```
>> clear
>> r=AddJc(5)
r =
   153
```

4.3 字符串的求值

MATLAB 提供了字符串求值的函数，利用这些函数，可以用字符串构造 MATLAB 的函数和命令，并运行这些字符串命令。

4.3.1 字符串表达式计算

【调用格式】

y=eval('expression')	计算字符串表达式 expression
[a1, a2, ...] = eval('function(b1, b2, ...) ')	计算函数调用的字符串表达式

【说明】eval 的输入变量只能是字符串

例 4.3.1 表达式字符串的计算。

解：运行下列语句，观察执行结果。

```
clear
x=0.4;
cmd=['y=x*eye(3), z=sin(x/2)+cos(x)'];
eval(cmd);
who
y =
```

```
    0.4000         0         0
         0    0.4000         0
         0         0    0.4000
z =
    1.1197
Your variables are:
  cmd   x     y     z
```

例 4.3.2 eval 和函数调用

解：运行下列语句，观察执行结果。

```
x=0:2*pi/50:2*pi;
eval('plot(x,sin(x), ''r'',x,cos(x), ''b: '')');   %相当于运行 plot(x,sin(x), 'r',x,cos(x), 'b: ')
y=eval('2*sin(x).*cos(x)');
figure
plot(x,y);
```

4.3.2 字符串函数计算

【调用格式】

[y1, y2, ...] = feval('function', x1, ..., xn)

【说明】

(1) 'function'只能是函数名，不能是表达式字符串。

(2) x1、x2 等是调用函数'function'的输入变量，即函数的自变量值。

(3) y1、y2 等是函数的输出变量，即函数的返回值。

例 4.3.3 feval 的使用方法。

解：执行下列语句，观察程序的运行结果

```
x=0:2*pi/50:2*pi;
feval('plot', x, sin(x), 'r', x, cos(x), 'b:');   %相当于运行 plot(x,sin(x), 'r',x,cos(x), 'b: ')
```

4.3.3 内联函数

用户可以用 M 文件来建立函数，函数的功能可以很复杂，函数的输出变量也可以有多个。对于简单的数学表达式，用 M 文件来建立函数就显得不够方便。MATLAB 提供了内联函数的功能，内联函数可以将表达式转换为函数。内联函数是 MATLAB 面向对象的一个类，其类型名为 inline。

1.内联函数的建立

(1)	**g = inline('expr')**	将串表达式 expr 转换为内联函数
(2)	**g = inline('expr', 'arg1', 'arg2',...)**	将串表达式 expr 转化为以 arg1、arg2 等为自变量（输入变量）的内联函数
(3)	**g = inline('expr',n)**	将串表达式 expr 转化为以自变量 x，P1，P2，…，Pn 为自变量的内联函数。其中 P 必须大写。

例 3.4.4 建立内联函数表示数学函数 $f(x)=\dfrac{\sin x}{x}$，并计算 $\lim\limits_{x\to 0} f(x)$ 的值

解：执行下列语句，观察程序的运行结果

```
>>f=inline('sin(x)/x')              %建立内联函数 f(x)=sin(x)/x
f =
     Inline function:
```

```
        f(x) = sin(x)/x
>> f(0)                          %计算 f(0)的值，结果为 NaN，因为用 0 做除数
Warning: Divide by zero.
> In inlineeval at 13
   In inline.subsref at 25
ans =
     NaN
>> f(eps)                        %eps 为浮点数的精度，可以当作是无穷小的数值
ans =
        1
```

例 3.4.5 阅读下列程序代码，了解 inline 的使用方法。

```
>> g = inline('sin(alpha*x)','x','alpha')
g =
        Inline function:
        g(x,alpha) = sin(alpha*x)
>> g1=inline('a*sin(x(1))*cos(x(2))', 'a','x')          %自变量为向量，函数值为标量
g1 =
        Inline function:
        g1(a,x) = a*sin(x(1))*cos(x(2))
>> g1(1,[pi/4,pi/4])
ans =
       0.5000
%自变量为向量，函数值为向量
>> g3=inline('[sin(x(1));cos(x(2));sqrt(x(1).^2+x(2).^2)]')
g3 =
        Inline function:
        g3(x) = [sin(x(1));cos(x(2));sqrt(x(1).^2+x(2).^2)]
>> g3([3,4])
ans =
       0.1411
      -0.6536
       5.0000
>> g4=inline('sqrt(x^2+P1^2+P2^2)',2)          %inline 的第三种格式的使用
g4 =
        Inline function:
        g4(x,P1,P2) = sqrt(x^2+P1^2+P2^2)
>> g4(1,3,5)
ans =
       5.9161
>> feval(g4,1,3,5)                             %内联函数可以通过 feval 计算求值
ans =
       5.9161
```

2. 和内联函数有关的函数

```
    class(fun)          %获取内联函数的数据类型
    char(fun)           %获取内联函数的计算公式字符串
    argnames(fun)       %获取内联函数的输入变量名字
```

vectorize(fun) %使内联函数具有数组运算规则

例 3.4.5 阅读下列程序代码，了解和内联函数相关的函数的使用方法。

```
>>  fun=inline('x^2+y^2')
fun =
     Inline function:
     fun(x,y) = x^2+y^2
>> class(fun)
ans =
inline
>> str=char(fun)
str =
x^2+y^2
>> class(str)
ans =
char
>> argnames(fun)
ans =
    'x'
    'y'
>> fv=vectorize(fun)          %把内联函数 fun 转化为具有数组运算的内联函数 fv
fv =
     Inline function:
     fv(x,y) = x.^2+y.^2
>> x=[1,2;3,4];
>> y=[1,0;0,1];
>> fun(x,y)                   %矩阵运算规则的运算
ans =
     8    10
    15    23
>> fv(x,y)                    %数组运算规则的运算
ans =
     2     4
     9    17
```

4.4 程序性能优化

MATLAB 语言是解释执行的语言，其优点是编程简单、使用方便，但其缺点是程序执行速度缓慢，执行效率低下。对于复杂的程序，程序员要考虑程序性能的优化，让应用程序既能够完成预期的功能，又具有较快的运行速度和较高的执行效率。本节介绍程序性能的优化方法，包括源代码的优化和使用 MATLAB 提供的程序加速功能。

4.4.1 源代码级的优化

有些编程方法有助于提供 M 文件的运行速度，而且效果会比较明显。

1. 循环的向量化

MATLAB 的运算功能是针对向量的，因此尽量少使用 for 循环和 while 循环，用向量化的数组代替单个元素的循环运算，即循环的向量化。循环的向量化不但能缩短源代码的长度，还能加快程序的运行速度，提高程序的执行效率。

例 4.4.1 求 log(n)在 n 从 1 到 100 之间的值，n 为整数。

```
%一般循环编程
for k=1:100
    y(k)=log10(k);
end
%循环的向量化编程
k=1:100;
y=log10(k);
```

2. 数组大小的预定义

MATLAB 在使用变量之前，不需要预先定义变量的名字和大小。如果变量的大小没有被显示的指定，则每当新赋值的元素下标超过变量的维数时，MATLAB 就自动为变量扩充维数，这大大降低了程序的运行效率。如果预先知道变量的维数，就可以预先定义好变量尺寸，通常采用 ones、zeros 或者 cell 等函数预定义变量维数。

例 4.4.2 预定义变量维数编程

```
y=zeros(1,101);          %预定义了结果变量 y 的大小
for i=0:100
    y(i)=sin(i);
end
```

3. 内存管理

MATLAB 系统的运行会占用大量的内存，在编写 MATLAB 程序的时候要注意合理给变量分派内存。以下方法可以提高内存的使用效率，减少内存碎片产生：

(1) 没有用的变量最好用 clear 语句删除
(2) 尽量不产生大的临时变量
(3) 使用 save 和 load 命令保存和读取变量
(4) 尽量采用函数文件代替命令文件

4. 尽可能采用 MATLAB 提供的函数

MATLAB 提供了大量的函数供用户调用，这些函数涵盖了大多数的常用操作。要尽可能使用 MATLAB 提供的函数，不要自己编写具有和 MATLAB 提供的函数具有相同功能的代码，这样无论从效率、可靠性和开发时间上都是得不偿失的。

4.4.2 程序加速器

MATLAB 提供了 JIT(Just In Time)和加速器(Accelerator)，用来加快函数文件和命令文件的运行速度。JIT 和加速器可以通过 MATLAB 命令开启和关闭，默认情况下 JIT 和加速器都是启动的。相关命令如下：

```
feature  accel  on      开启加速器
feature  accel  off     关闭加速器
feature  JIT    on      开启 JIT
feature  JIT    off     关闭 JIT
```

4.5 面向对象编程

MATLAB 支持面向对象的程序设计方法，本节简单介绍 MATLAB 中面向对象编程的基本概念和应用

方法。

4.5.1 类和对象

(1) 类（Class）是一个抽象的概念，它是具有相同特征和行为的对象的集合。

(2) 对象（Object）是类的具体实例（Instance），相当于类集合中的具体元素。

例如：浮点数 double 是一个类，a=0.3 定义了浮点数变量 a，a 就是浮点数类的一个对象。MATLAB 内建了 6 个类，它们分别是 char、double、cell、struct、sparse 和 unit8。

我们可以通过 class 函数来获取变量的数据类型，

【调用格式】

str=class(v) %返回变量 v 的数据类型名称

4.5.2 属性和方法

类中封装了该类对象共有的特征和行为。

(1) 属性（Property）

对象的特征称为属性，用数据来表示。对象的属性通常用结构体来描述，因此我们访问对象的属性可以象访问结构体的一样用成员运算符“.”来访问，也可以用 get 和 set 函数来访问对象的属性。

get(h,'PropertyName') 返回对象 h 的 PropertyName 属性的值

set(H,'PropertyName', Value,...) 赋值 H 对象的 PropertyName'属性值为 Value。

(2) 方法（Method）

对象的行为称为方法，用函数来表示。某个类的方法只能操作该类的对象。

4.5.3 构造函数

MATLAB 中没有类的声明语句。定义对象要调用类的构造函数（Constructor）。构造函数的名字必须与类同名，比如 cell 函数和 struct 函数就分别是细胞数组类和结构体数组类的构造函数。

c=cell(3,3); %调用构造函数 cell 建立一个 3×3 的细胞数组

4.5.4 重载

一些行为具有相同的定义，但实现方法不同，我们可以用同样的名称来描述这些操作，这种技术称为重载。被重载的函数具有相同的名字，但是对不同对象操作的时候表现出来的行为是不同的，调用的函数代码也不同。

重载包括函数重载和运算符重载。例如：乘法运算符“*”就被 MATLAB 重载过，分别用来实现矩阵乘法和传递系统的串联操作；eig 函数也被重载，用来实现矩阵和线性时不变系统的特征值操作。

4.5.5 继承

动物类和哺乳动物类就是被继承和继承的关系，哺乳动物类继承了动物类的一切特征和行为，也就是说动物有的特征哺乳动物都有，但是哺乳动物还有自己特有的特征和行为。这种关系我们称为继承关系，动物类称为是哺乳动物类的父类，哺乳动物类称为是动物类的子类。

子类会继承父类的属性和方法，同时还会有自己的新属性和新方法。

继承的使用能够提高代码的重用性，减少编程工作量。

4.5.6 创建新类

除了 MATLAB 内建的类之外，用户可以根据实际应用情况创建新类。创建一个新类的基本工作包括如下方面：

1.创建类目录

在 MATLAB 的搜索路径上创建一个子目录，子目录的名字为@加上类名，这个目录称为类目录，该类的代码就放在类目录下。

2.创建类的属性数据

类的属性用结构体来表示，抽象出该类共有的特征定义为结构体成员。

3.编写类的构造函数

用户通过调用和类同名的构造函数来创建类的对象。

4.重载类的显示函数 display

用来实现 MATLAB 调用属于该类的方法时，如何处理屏幕上的显示内容，即用分号和逗号结尾的语句如何显示调用的结果。

5.添加强制数据转换方法

添加必要的强制转换方法，将该类的对象转换为其他类型的数据。

6.重载需要的运算符和函数

7.添加类的其他方法

小结

MATLAB 既是一个开发平台，也是一门程序设计语言。本章主要介绍了 MATLAB 的程序设计，包括程序控制语句、命令文件、函数文件以及工作空间的操作等内容，还对面向对象的设计方法进行了简要介绍，为 MATLAB 的高级开发与应用提供基础。