

## 第二章 **MATLAB**基础

### ● 第四节 MATLAB程序设计

- 一、**MATLAB**的脚本文件
- 二、程序结构
- 三、**M**文件的程序流控制
- 四、**MATLAB**编程调试

## 一.脚本文件.m的形式和特点

### ❖ 两种常用的工作方式:

- 直接交互的指令行操作方式

- M文件的编程工作方式

- .M文件是ASCII码文件(标准的文本文件); 任何字处理软件都可以编写。

- .M文件有两种形式:

- 命令文件 (脚本文件)

- 函数文件

## ❖ 命令文件

- 只是一串指令的集合，不需要预定义，只是按照在指令窗中的指令输入顺序将指令编辑在命令文件中即可。
- 命令文件中的语句可以访问工作区中的所有数据。
- 运行过程中产生的所有变量均是全局变量。一直保存在内存中。
- 符号“%”引导的是注释行，不予执行。

## ❖ 命令文件

- 包含MATLAB语言代码的文件称为 M文件，其扩展名为.m。
- 命令文件就是由一系列的MATLAB指令和命令组成的纯文本格式的M文件。
- 命令文件没有输入参数，也没有输出参数。
- 执行命令文件时，文件中的指令或者命令按照出现在命令文件中的顺序依次执行。

# ❖ 命令文件

命令文件示例

% 注释行

% M文件示例

% “flower petal”

% 以下为代码行

% 计算

```
theta=-pi:0.01:pi;
```

```
rho(1,:)=2*sin(5*theta).^2;
```

```
rho(2,:)=cos(10*theta).^3;
```

```
rho(3,:)=sin(theta).^2;
```

```
rho(4,:)=5*cos(3.5*theta).^3;
```

```
for k=1:4
```

```
    % 图形输出
```

```
    subplot(2,2,k)
```

```
    polar(theta,rho(k,:))
```

```
end
```

```
disp('程序运行结束!')
```

- 在命令文件中，主要由注释行和代码行组成

- M文件的注释行需要使用%定义符

- 注释定义符仅能影响一行代码

- M文件的代码行是一些简单的MATLAB指令或命令

- 命令可以完成相应的计算处理数据、绘制图形结果的操作

- 可以在脚本文件中调用其他的函数完成复杂的数学运算

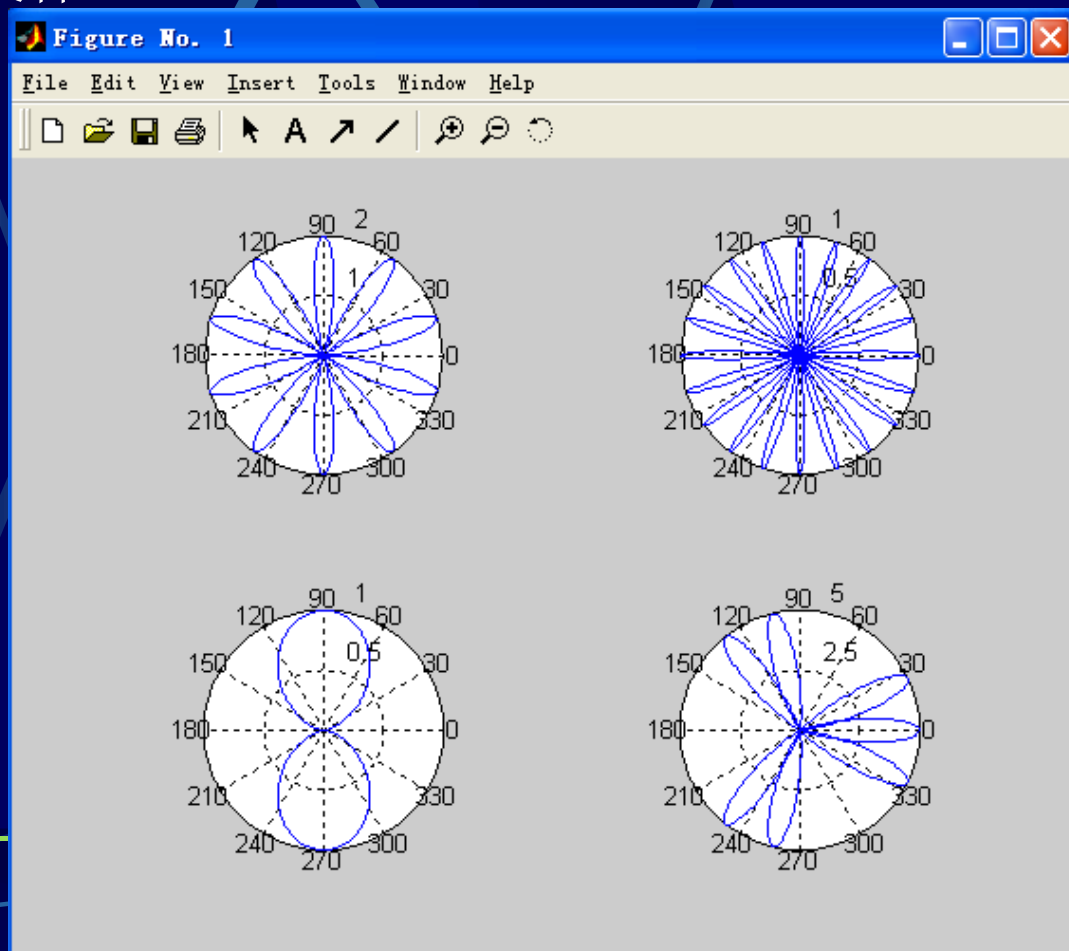
# ❖ 命令文件

在MATLAB命令行中运行该命令文件：

```
>> script_examp
```

程序运行结束！

MATLAB会出现相应的图形窗体



## ❖ 函数文件

- 函数文件第一行包含function。定义了函数名、输入参数和输出参数。
- 函数文件的变量仅在函数文件内部起作用，当函数文件执行完后这些变量将被清除。
- 在m文件全面连续几行带“%”的注释行有两个作用：
  - ✓ 随m文件全部显示或打印时起解释提示作用
  - ✓ 供help指令联机查询用。Help指令运行后所显示的是注释语句中的第一个连续块。

# 1 基本结构

- M语言函数文件具有下面的不同部分
  - 函数定义行
  - 在线帮助
  - 注释行
  - M语言代码



## 基本结构（续）

函数m文件的格式：

**function** 返回变量 = 函数名（输入变量）

注释说明语句段

程序语句段

特定规则：

- 1) 函数m文件第一行必须以单词**function**作为引导词，必须遵循如下形式：**function** <因变量>=<函数名>(<自变量>)
- 2) 程序中的变量均为局部变量，不保存在工作空间中，其变量只在函数运行期间有效。

## 基本结构（续）

```
001 function y=average(x)
002 % AVERAGE 求向量元素的均值
003 % 语法：
004 % Y=average(X)
005 % 其中，X是向量，Y为计算得到向量元素的均值
006 % 若输入参数为非向量则出错
007
008 % 代码行
009 [m,n]=size(x)
010 % 判断输入参数是否为向量
011 if(~((m==1)|(n==1))|(m==1&n==1))
012     % 若输入参数不是向量，则出错
013     error('Input must be a vector')
014 end
015 % 计算向量元素的均值
016 y=sum(x)/length(x)
```

# 基本结构（续）

- 函数定义行

001 function y=average(x)

- 包括

- 关键字function
- 函数输出参数y
- 函数名称average
- 函数输入参数x

- 函数名称定义要求

- 必须以字符开头，后面可以用字符、数字和下划线的组合构成函数名称
- MATLAB对函数名称的长度有限定
- 函数的M文件名称最好和函数名称保持一致，若不一致，则调用函数时需要使用文件名称而非函数名称

## 基本结构（续）

- 在线帮助
    - M函数文件的在线帮助为紧随在函数定义行的注释行
- ```
002 % AVERAGE 求向量元素的均值
003 % 语法:
004 % Y=average(X)
005 % 其中，X是向量，Y为计算得到向量元素的均值
006 % 若输入参数为非向量则出错
```

## 基本结构（续）

- 若在MATLAB命令行窗口中键入指令help average  
则

```
>> help average
```

AVERAGE 求向量元素的均值

语法：

`Y=average(X)`

其中，X是向量，Y为计算得到向量元素的均值  
若输入参数为非向量则出错

## 基本结构（续）

- 若在MATLAB命令行窗口中键入指令lookfor average  
则

```
>> lookfor average
```

AVERAGE 求向量元素的均值

MEAN Average or mean value.

MOVAVG Leading and lagging moving averages chart.

MBSWAL Weighted Average Life of mortgage pool.

...

- 由于H1帮助行的特殊作用，所以在用户自己定义M函数文件时，一定要编写相应的H1帮助行，对函数进行简明、扼要的说明或解释

## 基本结构（续）

- 注释行

008 % 代码行

010 % 判断输入参数是否为向量

012 % 若输入参数不是向量，则出错

015 % 计算向量元素的均值

- 注释行不会显示在在线帮助中，主要原因是这些注释行没有紧随在H1帮助行的后面

## 基本结构（续）

- M语言代码

```
008 % 代码行
009 [m,n]=size(x)
010 % 判断输入参数是否为向量
011 if (~((m==1)|(n==1))|(m==1&n==1))
012     % 若输入参数不是向量，则出错
013     error('Input must be a vector')
014 end
015 % 计算向量元素的均值
016 y=sum(x)/length(x)
```

- 代码行需要完成具体的算法，实现用户的具体功能



## 2 输入输出参数

- MATLAB在定义输入输出参数时不需要指出变量的类型，而是将参数默认为双精度型
- MATLAB在定义参数时，没有确定输入参数的维数或者尺寸
- M语言的函数文件不仅可以有一个输入参数和一个返回值，还可以为M语言函数文件定义多个输入参数和多个输出参数

## 输入输出参数（续）

例4-15 多个输入输出参数的M函数

```
function[avg,stdev,r]=ourstats(x,tol)
% OURSTATS多输入输出参数示例
% 该函数计算处理矩阵，得到相应的均值、标准差和矩阵的秩
[m,n]=size(x);
if m==1
    m=n;
end
% Average
avg=sum(x)/m;
% Sandad deviation
stdev=sqrt(sum(x.^2)/m-avg.^2);
% Rank
s=svd(x);
r=sum(s>tol);
```

## 输入输出参数（续）

运行例4-15

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

```
A =
```

```
    1    2    3
    4    5    6
```

```
>> [a,s,r]=ourstats(A,0.1)
```

```
a =
```

```
    2.5000    3.5000    4.5000
```

```
s =
```

```
    1.5000    1.5000    1.5000
```

```
r =
```

```
    2
```

```
>> [a,s]=ourstats(A,0.1)
```

```
a =
```

```
    2.5000    3.5000    4.5000
```

```
s =
```

```
    1.5000    1.5000    1.5000
```

```
>> a=ourstats(A,0.1)
```

```
a =
```

```
    2.5000    3.5000    4.5000
```

调用该函数时，将输出参数依次写在一个向量中，

若输出参数的个数与函数定义的输出参数个数不一致时，将计算得到的前几个输出参数作为返回值，

若输出参数的个数等于指定的输出参数个数时，计算结果依次赋值给不同的变量。

## 输入输出参数（续）

例4-16 nargout和nargin示例

```
function c=testarg(a,b)
% TESTARG检测输入输出参数个数
% 该函数根据不同的输入输出参数个数进行相应的操作
if (nargout~=1)
    disp('使用该函数必须指定一个输出参数!');
    return
end
switch nargin
    case 0
        disp('使用该函数必须指定一个输入参数!');
        c=[];
        return
    case 1
        c=a.^2;
    case 2
        c=a+b;
end
```

函数nargin用来获取函数的  
输入参数个数  
函数nargout用来获取输出  
函数个数

## 输入输出参数（续）

运行例4-16

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

```
>> testarg(A,B)
```

使用该函数必须指定一个输出参数！

```
>> C=testarg
```

使用该函数必须指定一个输入参数！

```
C =
```

```
[]
```

```
>> C=testarg(A)
```

```
C =
```

```
1      4      9
```

```
>> D=testarg(A,B)
```

```
D =
```

```
3      5      8
```

```
>> E=testarg(A,B,C)
```

```
??? Error using ==> testarg
```

```
Too many input arguments.
```

## 二.程序结构

顺序结构  
循环结构  
分支结构

### ❖ 顺序结构:

说明:

- 由复合表达式构成，以“，”或“；”隔离
- “；”表示计算结果不显示，但中间变量结构仍保存在内存中。
- 若是命令文件，则程序运行后，中间变量予以保留；若是函数文件，则程序运行后中间保留将被全部删除。

## ❖ 循环结构→重复运算

{ For — — end  
While — — end

### ✓ (1).For-end循环

➤ 格式 

```
for x=array
    {commands}
end
```

Commands按数组中的每一列执行一次。在每一次迭代中，x被指定为数组的一列。所以在第n次循环中

$x = \text{array}(:,n)$

➤ 例: 

```
for n=1: 10
    x(n)=sin(n*pi/10)
end
```



```
n=1: 10
x=sin(n*pi/10);
x
```

## ➤说明:

- ✓ For循环不能用for循环内部重新赋值循环变量n的方法来终止。
- ✓ 语句1:10是一个标准的Matlab数组创建语句，在for内接受任何有效的Matlab数组。
- ✓ 当用一个有效的数组来解给定的问题时，应避免用for。→上例后者更快。
- ✓ For循环可以嵌套。
- ✓ 为了得到最大的速度，在for(while)被执行之前，应预先分配数组。

→加上x的初始化语句`x=zeros(1,10)`



## ✓ (2).while-end循环

➤ 格式     while expression  
                  { commands }  
                  end

➤ 例:    EPS=1; num=0;  
         while (1+EPS>1)  
              EPS = EPS/2;  
              Num=num+1;  
         End  
         num =  
         EPS =2\*EPS;

只要expression的所有元素为真，就执行commands。

## ❖ 分支结构

### ➤ 格式1

```
if expression
{commands}
end
```

只要expression的所有  
元素为真，就执行  
commands。

### ➤ 格式2

```
if expression
{commands1}
else
{commands2}
end
```

expression为真  
时的指令序列。

Expression为假  
时的指令序列。

### ➤ 格式3

```
if expression1
{commands1}
elseif expression2
{commands2}
elseif expression3
{commands3}
elseif .....
.....
end
```

expression1为真  
时的指令序列。

expression2为真  
时的指令序列。

## ❖ 例1: 求EPS的另一种方法

```
EPS=1;  
for num=1:100  
    EPS = EPS/2;  
    if (1+EPS)<=1  
        EPS = 2*EPS;  
        break;  
    end  
end
```

break出现在嵌套的for或while循环结构里，那么matlab只跳出break所在的那个循环里，而不跳出整个嵌套结构。

## ❖例2：折扣问题。

购买2.5元/斤的苹果，如果购买量超过50斤，给20%的折扣；超过100斤，给30%的折扣。

```
Apples = 100;           % apple数目
Cost = apples*2.5;       % 购买apple的费用
Cost = 250
If(apples>50)           % 购买量超过50斤，
    cost = (1 - 20/100)*cost; % 给20%折扣
elseif (apple>100)      % 超过100斤，
    cost = (1-30%)*cost % 给30%折扣
end
```

### 三.程序流控制

#### ❖ echo 指令

✓ echo on

✓ echo off

✓ echo

✓ echo FileName on

✓ echo FileName off

仅用于命令文件

显示其后的所以被执行命令文件的指令

不显示其后的所以被执行命令文件的指令

在以上两种状态中间切换。

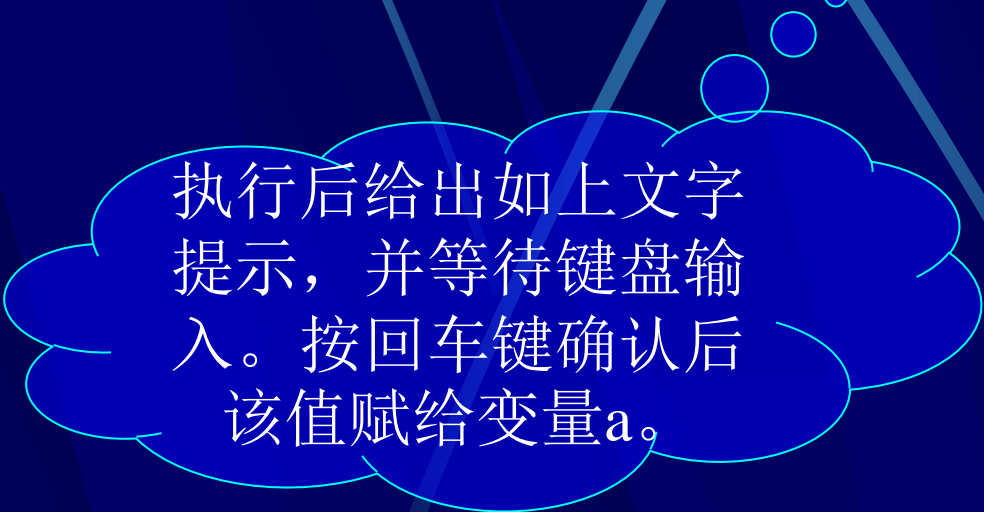
显示其后的所以被执行命令文件的指令。

中止显示FileName文件的执行过程

用于命令文件和函数文件

## ❖ input指令

- 功能：提示用户从键盘输入数值、字符串或表达式，并接受该输入。
- ✓ 格式1： `a=input('please input a number:')`
- ✓ 格式2： `a=input('please input a string:', 's')`



执行后给出如上文字提示，并等待键盘输入。按回车键确认后该值赋给变量a。

## ❖ pause指令

➤ 功能：使程序运行暂停，等待用户按任意键继续该命令在程序调试以及需要看中间结果时特别有用。

✓ 格式1: `pause` % 按任意键继续。

✓ 格式2: `pause(n)` % 在继续执行前暂停n秒钟

n可以不是整数，  
例如8.5。

## ❖ keyboard指令

- **功能：**使程序运行暂停，并且调用机器的键盘命令进行处理。一旦处理完自己的工作之后，输入return，然后按下enter回车键，程序将继续运行。

## ❖ break指令

- **功能：**导致包含break指令的最内层while、for、if循环语句中止。
- **说明：**用break语句可以不必等循环的自然结束，而是根据循环内部令设的条件决定是否退出循环，是否结束if语句。



## ❖ 外部系统命令

### ➤ 同步实时处理命令！

✓ 使用方法：在外部系统命令之前加上“！”，并且在其之后没有其他附加的参数。

✓ 功能：Matlab向Windows或dos发出系统命令，等到该命令完成之后才开始接受新的命令。

### ➤ 后台处理命令 &

✓ 使用方法：在外部系统命令之前加上“！”，并且在其之后以字符“&”结尾。

✓ 功能：同上，但Matlab将此命令作为后台命令处理，不必等到该命令完成之后就可以接受和执行新的命令。

# 三.M文件的调试

举例

### 三.M文件的调试

结束