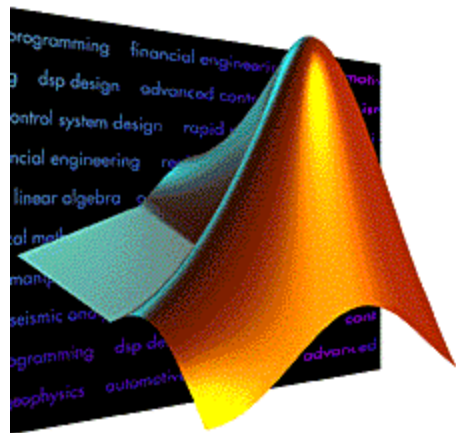

Matlab 程序设计与应用

第4章: Matlab程序设计

伍振海



September 24, 2017
Autumn, @swpu

第4章：Matlab程序设计



本章要点

- **4.1 M文件**
- **4.2 程序控制结构**
- **4.3 函数文件**
- **4.4 程序调试**

4.1 M文件

- 命令窗口只能输入简单命令，回车后就会马上执行。如果命令很多，很不方便。

怎么办？

- 把所有命令写在一个文件里，再执行整个文件！

→M文件：Matlab命令的集合，文件后缀为“.m”。

➤ 命令文件(Script File):

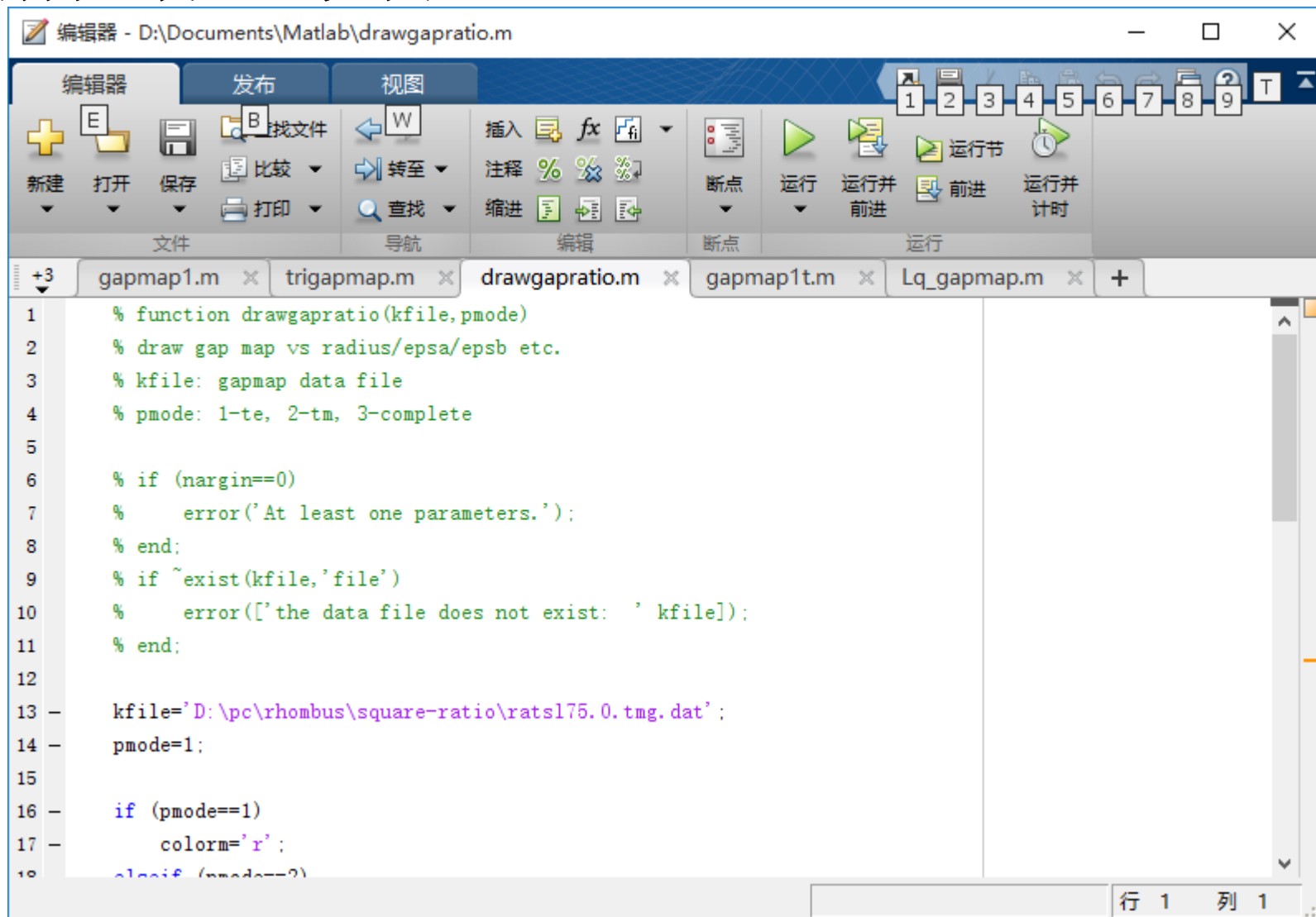
- 命令的集合，无输入参数，无返回值；
- 运行后，变量保存在变量空间中，与命令窗口输入命令运行的结果一样；
- 可在命令窗口中直接输入文件名执行。

➤ 函数文件(Function File):

- 一般有输入参数，有返回值（但可以没有）；
- 运行后，函数内部的局部变量会被清除；
- 需使用函数调用的方式执行；

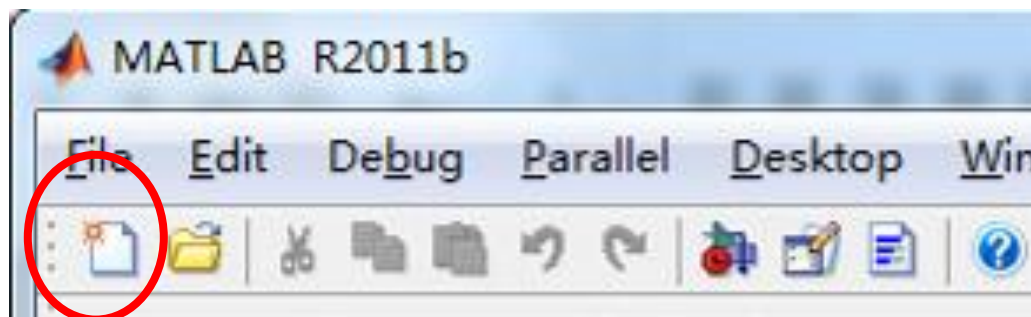
4.1 M文件

- 编辑器窗口，如图：

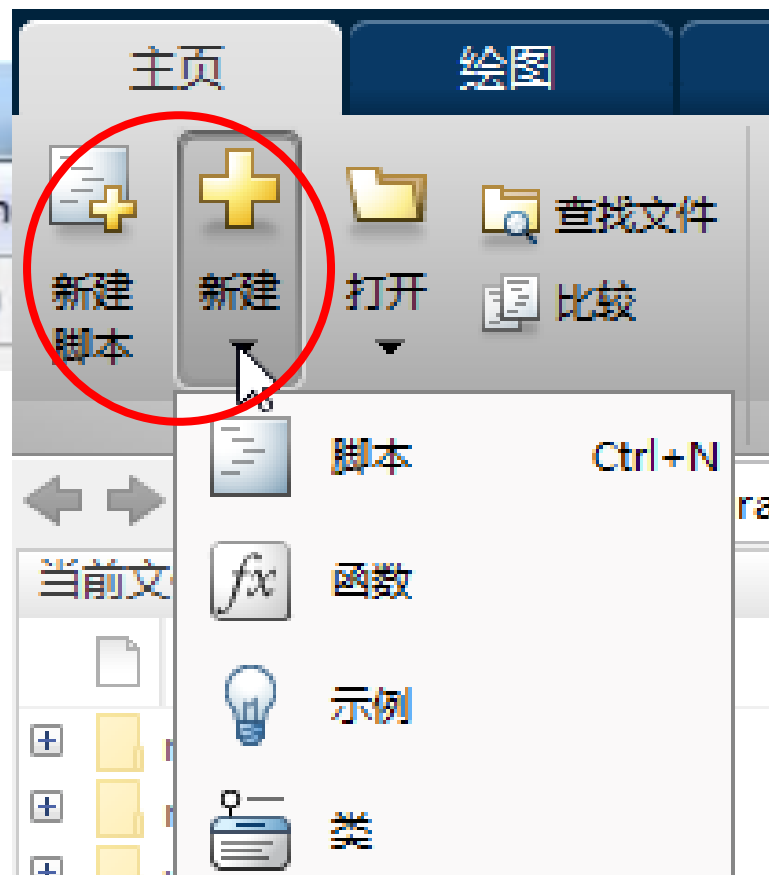


1. 建立新的M文件

- 菜单：File → New → M-file (快捷：Ctrl+N)
- 命令：在命令窗口输入：edit
- 命令按钮：工具栏New M-File按钮
- 新版本：点击主页中的新建

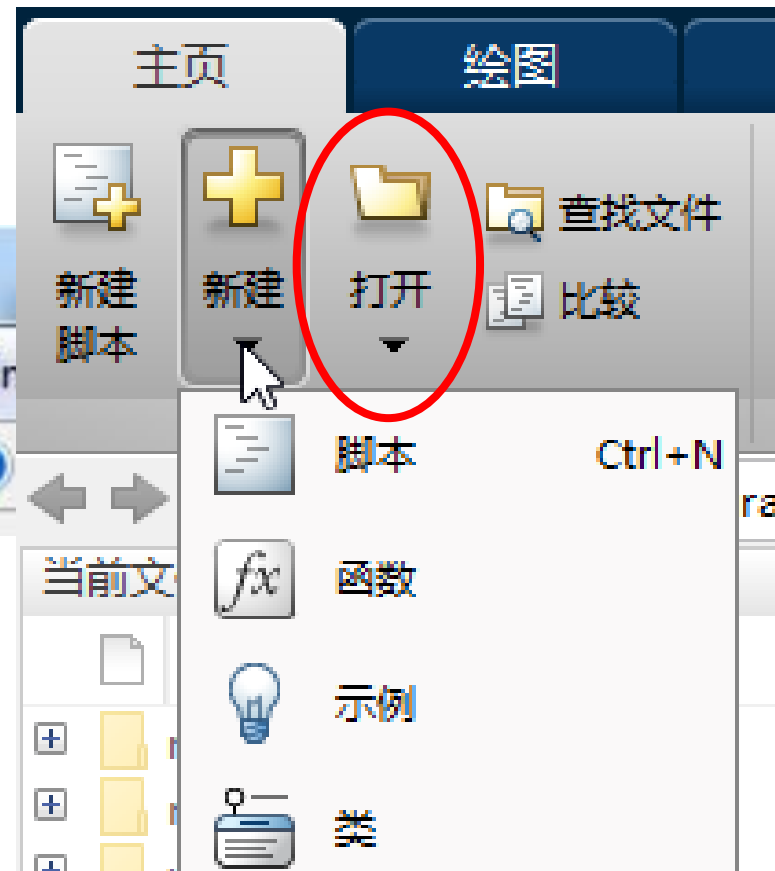
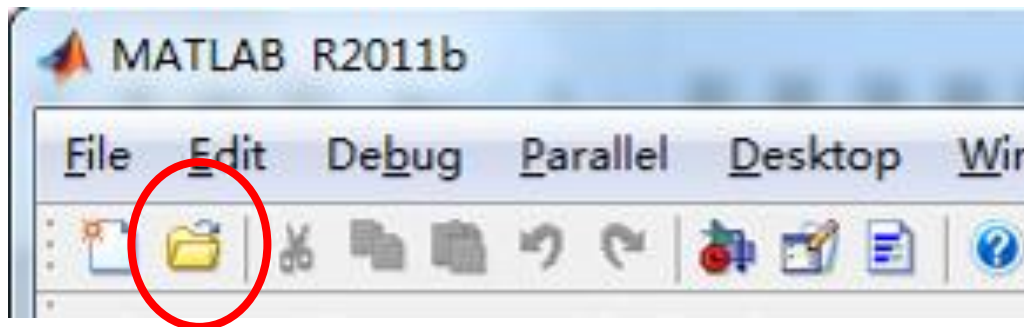


Note: M文件是一个**文本文件**，它可以用任何编辑程序来建立和编辑，一般用MATLAB自带文本编辑器，也可以用记事本、Notepad++等来建立和编辑。但是运行必须在Matlab中运行！



2. 打开M文件

- 菜单: **File**→**Open** （快捷**Ctrl+O**）
- 命令: 命令窗口输入命令: **edit filename**
- 命令按钮: 工具栏**Open File**按钮
- 新版本: 点击主页中的打开



4.1 M文件

例： 分别建立命令文件和函数文件，将华氏温度 f 转换为摄氏温度 c 。

解： 1、命令文件 f2c.m

```
clear;           %清除工作空间中的变量  
f=input('Input Fahrenheit temperature: ');  
c=5*(f-32)/9
```

然后在MATLAB的命令窗口中输入f2c，将会执行该命令文件，执行情况为：

```
Input Fahrenheit temperature: 73  
c =  
    22.7778
```

4.1 M文件

2、函数文件f2cm.m

```
function c=f2cm(f)  
c=5*(f-32)/9;
```

然后在MATLAB的命令窗口调用该函数文件。

```
x=f2cm(96)
```

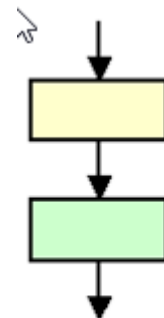
输出情况为：

```
x =  
21.1111
```


4.2 程序控制结构

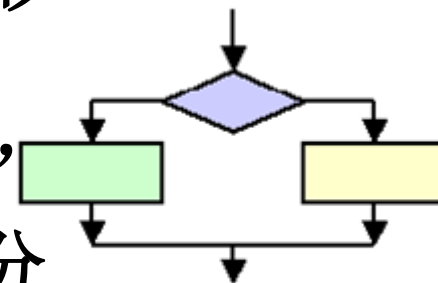
- 三种基本控制结构：

- **顺序结构：**按照语句的书写顺序从上到下、逐条语句执行。排在前面的代码先执行，排在后面的代码后执行，执行过程中没有任何分支。顺序结构是最普遍的结构形式，也是后面两种结构的基础。



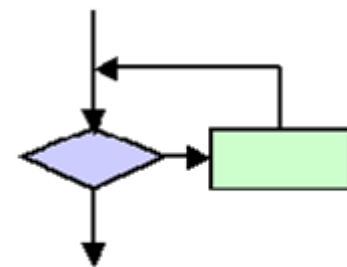
顺序结构

- **选择结构：**又叫分支结构。根据“条件”来选择执行哪一支中的语句。包括二分支和多分支，以及分支的嵌套。



选择结构

- **循环结构：**循环结构的思想是利用计算机高速处理运算的特性，重复执行某一部分代码，以完成大量有规则的重复运算。



循环结构

4.2.1 顺序结构

1. 提示输入及获取数据

A=input(提示信息, 选项);

提示信息：‘字符串’，提示用户输入什么样的数据。

输入数据：数据、矩阵、字符串等

```
>> A=input(' input data:')  
input data:10  
A =  
    10
```

```
>> A=input('input data:')  
input your data:'zhang san'  
A =  
zhang san
```

必须有引号

```
>> A=input('input data:')  
input your data:[1,2;3,4]  
A =  
     1     2  
     3     4
```

```
>> A=input('input data:', 's')  
input data:zhang san  
A =  
zhang san
```

加's'参数，表示输入字符串，故输入时可不加引号

4.2.1 顺序结构

2. 数据的输出: **disp**(输出项)

输出项: 字符串、矩阵等。

例4-2 输入x,y的值, 并将它们的值互换后输出。

```
x=input('Input x: ');  
y=input('Input y: ');  
z=x;  
x=y;  
y=z;  
disp(x);  
disp(y);
```

```
>> x=1;  
>> x  
x =  
    1  
>> disp(x)  
    1
```

4.2.1 顺序结构

- **disp**函数常用于输出一些复杂的结果

例4-3 求一元二次方程 $ax^2 + bx + c = 0$ 的根。

```
a=input('a=?');
```

```
b=input('b=?');
```

```
c=input('c=?');
```

```
d=b*b-4*a*c;
```

```
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)];
```

```
disp(['x1=',num2str(x(1)),',x2=',num2str(x(2))]);
```

4.2.1 顺序结构

3. 程序的暂停: `pause(n)`

暂停n秒, n——可以是整数, 也可以是小数

exa:

`pause(1/1000)` % 暂停1毫秒

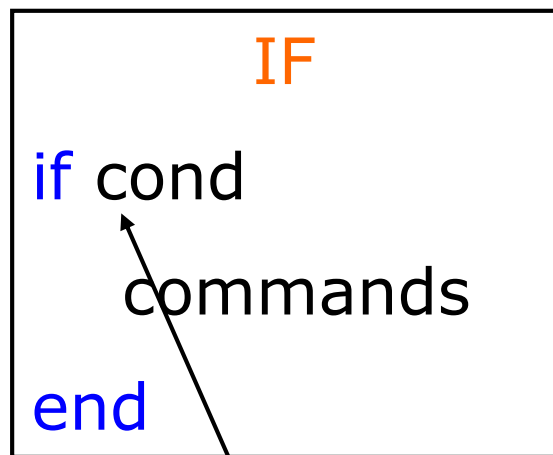
强行中止程序的运行:**Ctrl+C**。

4.2.2 选择结构—`if...else...elseif...end`

- 基本的流程控制语句，基本上所有语言都有
- **Matlab** 语法类似，如下：

IF

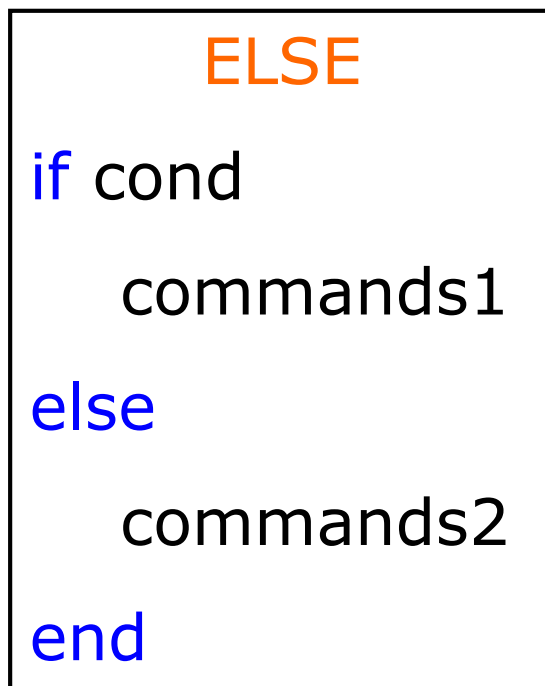
```
if cond
    commands
end
```

A box containing the code for an IF statement. The word 'if' is blue, 'cond' is black, 'commands' is black, and 'end' is blue. An arrow points from the word 'if' to the word 'end'.

条件声明：
真（1）或假（0）

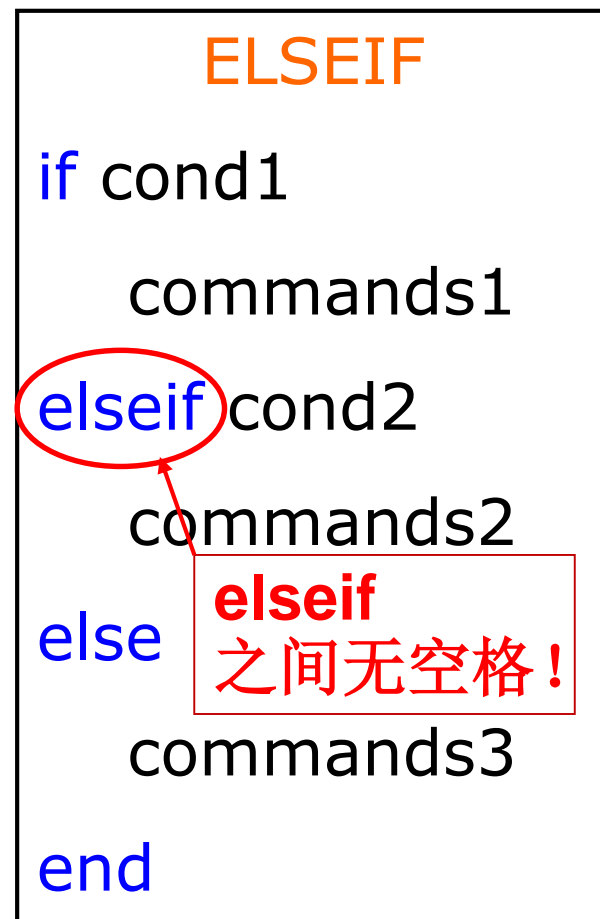
ELSE

```
if cond
    commands1
else
    commands2
end
```

A box containing the code for an ELSE statement. The word 'if' is blue, 'cond' is black, 'commands1' is black, 'else' is blue, 'commands2' is black, and 'end' is blue.

ELSEIF

```
if cond1
    commands1
elseif cond2
    commands2
else
    commands3
end
```

A box containing the code for an ELSEIF statement. The word 'if' is blue, 'cond1' is black, 'commands1' is black, 'elseif' is blue and circled in red, 'cond2' is black, 'commands2' is black, 'else' is blue, 'commands3' is black, and 'end' is blue. A red box highlights the text 'elseif 之间无空格!' with an arrow pointing to the 'elseif' word.

- **Note:** 条件和命令都不需要括号，
它们写在保留字符`if...else...end`之间。

4.2.2 选择结构—`if...else...elseif...end`

例 已知分段函数 $y = \begin{cases} \frac{x + \sqrt{\pi}}{e^2} & (x \leq 0) \\ \ln(x + \sqrt{1 + x^2})/2 & (x > 2) \end{cases}$

输入一个数，并计算其值。

```
x=input('请输入x的值:');  
if x<=0  
    y=(x+sqrt(pi))/exp(2);  
else  
    y=log(x+sqrt(1+x*x))/2;  
end  
y
```

4.2.2 选择结构—**if...else...elseif...end**

例： 输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符则输出其对应的数值，若为其他字符则原样输出。

ASCII: A~Z: 65~106 , a~z:97~122 , 0~9:64~73

```
c=input('请输入一个字符:', 's');  
if c>='A' & c<='Z'  
    disp(char(abs(c)+abs('a')-abs('A')));  
elseif c>='a' & c<='z'  
    disp(char(abs(c)-abs('a')+abs('A')));  
elseif c>='0' & c<='9'  
    disp(abs(c)-abs('0'));  
else  
    disp(c);  
end
```



abs:取得字符的ASCII码
char:将ASCII码转换为字符

4.2.2 选择结构—switch...case...end

```
switch expression
  case expr_1
    commands_1
  case expr_2
    commands_2
  otherwise
    commands_n
end
```

当**expression=expr_n**时：执行**commands_n**
当任意一个分支的语句执行完后，直接执行switch语句的下一句。

4.2.2 选择结构—switch...case...end

Note 1: Matlab中的switch case, **只寻找第一个满足条件的case并执行, 然后跳出switch语句, 如果该case后有其它满足条件的case, 它不会执行。**

```
x=1;  
switch x  
    case 1  
        y=1;  
    case 1  
        y=2;  
end  
结果: y=1
```

4.2.2 选择结构—switch...case...end

Note 2: switch case**不能使用类似if else语句中的逻辑表达式来判断**，若使用了逻辑表达式，它不会报错，但也得不到正确结果，容易引发程序功能问题。

当出现逻辑表达式时，会先算出其值，再来和x做比较

如：当 $x=10$ 时，

case $x < 0$ 相当于 case 0

case $x > 0$ 相当于 case 1

```
x=input('input x: ');
switch x
    case x<0
        y=-1;
    case x==0
        y=0;
    case x>0
        y=1;
    otherwise
        y=100;
end
y
```

4.2.2 选择结构—switch...case...end

例：某商场对顾客所购买的商品实行打折销售，标准如下(商品价格用price来表示)：

price<200 没有折扣

200≤price<500 3%折扣

500≤price<1000 5%折扣

1000≤price<2500 8%折扣

2500≤price<5000 10%折扣

5000≤price 14%折扣

输入所售商品的价格，求其实际销售价格。

分析：
这种写法对不对？

错误，只能使用**if...else..end**
语句来匹配逻辑表达式

```
clear all;
price=input('输入价格: ');
switch price
    case price<200
        rate=0;
    case price>=200 & price<500
        rate=3/100;
    case price>=500 & price<1000
        rate=5/100;
    case price>=1000 & price<2500
        rate=8/100;
    case price>=2500 & price<5000
        rate=10/100;
```

```
    otherwise
        rate=14/100;
    end
    price=price*(1-rate)
```

用if...else...end来写:

```
clear all;  
price=input('输入价格');  
if price<200  
    rate=0;  
elseif price>=200 & price<500  
    rate=3/100;  
elseif price>=500 & price<1000  
    rate=5/100;  
elseif price>=1000 & price<2500  
    rate=8/100;  
elseif price>=2500 & price<5000  
    rate=10/100;  
else
```

```
    rate=14/100;  
end  
price=price*(1-rate)
```

用switch...case来写:

```
price=input('输入价格');  
switch fix(price/100)  
    case {0,1}  
        %价格小于200  
        rate=0;  
    case {2,3,4}  
        %价格>=200但<500  
        rate=3/100;  
    case num2cell(5:9)  
        %价格>=500但<1000  
        rate=5/100;  
    case num2cell(10:24)  
        %价格>=1000但<2500
```

```
        rate=8/100;  
    case num2cell(25:49)  
        %价格>=2500但<5000  
        rate=10/100;  
    otherwise  
        %价格>=5000  
        rate=14/100;  
end  
price=price*(1-rate)  
%输出商品实际销售价格
```

使用find语句来实现该功能

```
price=input('请输入商品价格');  
pricerange = [0, 200, 500, 1000, 2500, 5000];  
           %价格范围临界点  
Raterange = [0, 3/100, 5/100, 8/100, 10/100,  
            14/100];  
           %对应的折扣率  
rateind=find(price >= pricerange,1,'last');  
           %找出最后一个满足条件的  
rate=raterange(rateind)  
price=price*(1-rate)
```


4.2.2 选择结构—try...catch...end

```
try
    commands1
catch
    commands2
end
```

先试探性执行commands1，如果在执行过程中出现错误，则将错误信息赋给保留的**lasterr**变量，并转去执行commands2。

例 先求两矩阵的乘积，若出错，则转去求点乘。

```
A=[1,2,3;4,5,6]; B=[7,8,9;10,11,12];
```

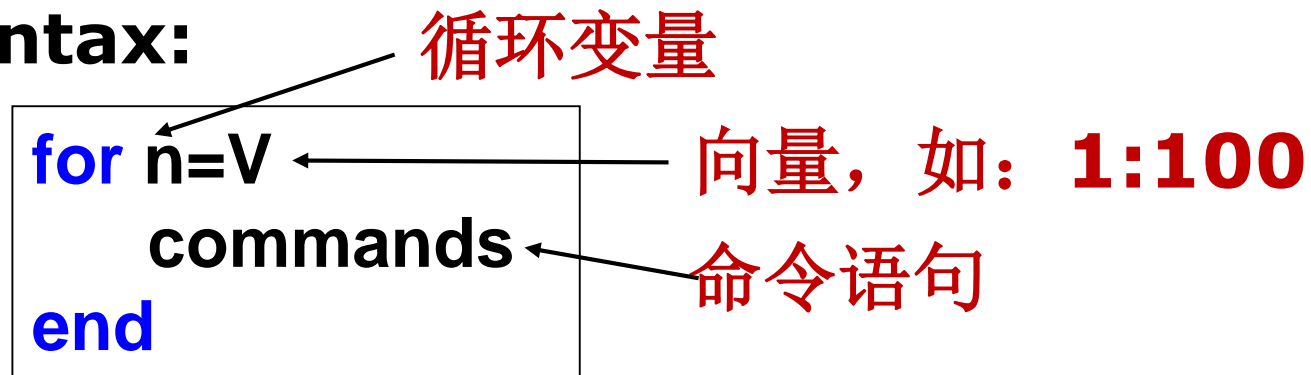
```
try
    C=A*B;
catch
    C=A.*B;
end
C
lasterr
```

%显示出错原因

4.2.3 循环结构—**for...end**

- **for loops:** 对一个已知量进行迭代

- **MATLAB syntax:**



循环变量:

- **V**定义为一个向量（如: **V=1:100**）
 - **n**在命令语句中是一个标量（数量）（如**n=99**）
 - 不一定是连续变化的值
- 命令语句**commands**:
 - 在**for...end**之间的任意命令集，不需要括号

4.2.3 循环结构—**for...end**

● **for loop**常用的语法形式:

```
for x=1:5  
    disp(x)  
end
```

直接使用冒号表达式是最常见的形式

```
for x=1:2:9  
    disp(x)  
end
```

可以是不连续的任意向量表达式

可以先定义循环变量

```
for x=[1, 5, 9, 2, 3]  
    disp(x)  
End
```

```
V=linspace(-pi,pi,5);  
for x=V  
    disp(x)  
end
```

4.2.3 循环结构—**for...end**

例 一个三位整数各位数字的立方和等于该数本身则称该数为水仙花数。输出全部水仙花数。

```
for m=100:999
```

```
    m1=fix(m/100);
```

```
%求m的百位数字
```

```
    m2=rem(fix(m/10),10);
```

```
%求m的十位数字
```

```
    m3=rem(m,10);
```

```
%求m的个位数字
```

```
    if m==m1^3+m2^3+m3^3
```

```
        disp(m)
```

```
    end
```

```
end
```

4.2.3 循环结构—**for...end**

思考：如何用**find**函数求解？

```
m=100:999;  
m1=fix(m/100);  
m2=rem(fix(m/10),10);  
m3=rem(m,10);  
A=find(m1.^3+m2.^3+m3.^3==m);  
B=m(A)
```

结果：

```
A =  
    54    271    272    308  
B =  
    153    370    371    407
```

Note：向量化运算是**Matlab**的巨大优势，能不使用循环的时候尽量不要使用循环。

4.2.3 循环结构—**for...end**

例4-9 已知, $y = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$,

当 $n = 100$ 时, 求 y 的值

使用**for**循环:

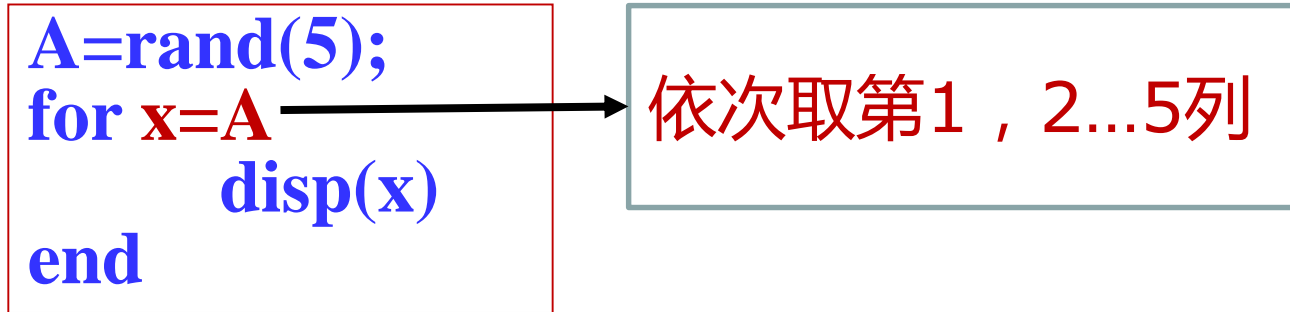
```
y=0;  
n=100;  
for m=1:n  
    y=y+1/(m^2);  
end  
y
```

使用向量运算:

```
n=100;  
m=1:n;  
y=sum(1./m.^2);  
y
```

4.2.3 循环结构—for...end

- 当循环变量为矩阵时：



依次将矩阵的

各列元素

赋给循环变量，然后执行循环体语句，直至各列元素处理完毕。

4.2.3 循环结构—**for...end**

例4-10 写出下列程序的执行结果。

```
s=0;  
a=[12,13,14;15,16,17;18,19,20;21,22,23];  
for k=a  
    s=s+k;  
end  
disp(s');
```

a =

12	13	14
15	16	17
18	19	20
21	22	23

s =

39
48
57
66

4.2.3 循环结构—while...end

while语句的一般格式为：

```
while cond  
    commands  
end
```

若条件成立，则执行循环体语句，执行后再判断条件是否成立，如果不成立则跳出循环。

4.2.3 循环结构—while...end

例4-11 从键盘输入若干个数，当输入0时结束输入，求这些数的平均值和它们之和。

```
sumv=0;
cnt=0;
val=input('Enter a number (end in 0):');
while val~=0
    sumv=sumv+val;
    cnt=cnt+1;
    val=input('Enter a number (end in 0):');
end
if cnt > 0
    sumv
    mean=sumv/cnt
end
```

4.2.4 循环结构—break/continue

Break:终止循环的执行。

——当在循环体内执行到该语句时，程序将跳出循环，继续执行循环语句的下一语句。

Continue: 语句控制跳过循环体中的某些语句。


——当在循环体内执行到该语句时，程序将跳过循环体中所有剩下的语句，**继续下一次循环**。

break和**continue**常与**if**语句配合使用。

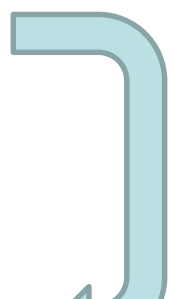
4.2.4 循环结构—break/continue

例4-13 求[100, 200]之间第一个能被21整除的整数。

```
for n=100:200
    if rem(n,21)~=0
        continue
    end
    break
end
n
```



```
for n=100:200
    if rem(n,21)==0
        break
    end
end
n
```



4.2.5 循环结构—嵌套

循环的嵌套：

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为**多重循环结构**。

while语句与**for**语句可互相嵌套。

4.2.5 循环结构—嵌套

例4-14 若一个数等于它的各个真因子之和，则称该数为完数，如 $6=1+2+3$ ，所以6是完数。求 $[1,500]$ 之间的全部完数。

完数 (Perfect number) 是一些特殊的自然数：

它所有的真因子（即除了本身以外的约数）的和，恰好等于它本身。例如：第一个完全数是6，它有约数1、2、3、6，除去它本身6外，其余3个数相加， $1+2+3=6$ 。又如：8的真因子是1，2，4，而 $1+2+4=7$ ，所以8不是完全数

4.2.5 循环结构—嵌套

```
for m=1:500
    s=0;
    for k=1:m/2
        if rem(m,k)==0
            s=s+k;
        end
    end
    if m==s
        disp(m);
    end
end
```

4.3 函数文件

- 函数文件与命令文件类似，但有一个区别：
 - **Functions** 函数必须有一行函数申明

```
C:\MATLAB6p5\work\stats.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons] Stack: Base
1 % stats: computes the average, standard deviation, and range
2 % of a given vector of data
3 %
4 % [avg,sd,range]=stats(x)
5 % avg - the average (arithmetic mean) of x
6 % sd - the standard deviation of x
7 % range - a 2x1 vector containing the min and max values in x
8 % x - a vector of values
9 function [avg,sd,range]=stats(x)
10 avg=mean(x);
11 sd=std(x);
12 range=[min(x); max(x)];
```

帮助文件

函数申明

Outputs Inputs

coinToss.m stats.m

stats Ln 12 Col 24

4.3 函数文件:函数申明

必须以**function**开头

输入参数

function [x, y, z] = **funName**(in1, in2)

%comments 注释说明
commands 函数体语句

函数名, 与
文件名一致

end

若整个文件只有一个
函数, 可以不写**end**

输出参数, 如果多于一个,
必须用中括号括起来

- 不需要**return**: 把运算结果直接赋值给返回参量即可。
- 变量范围: 在函数内部的变量, 执行结束后, 会被清空。
- 函数名的命名规则与变量名相同。
- 当输出形参多于一个时, 则应该用方括号括起来。
- 输入输出参数可以是: 数量、矩阵、字符串等

4.3 函数文件

Note:

- 函数文件名: 函数名+.m, 即文件名与函数名应一致。
——若不一致时, matlab将忽略函数名而使用文件名, 此时调用时应使用文件名
- 第一行注释将作为函数功能描述, 并可供help等命令查询——注释不是必须的。
- **return**: 函数中遇到return将在该处停止执行并返回
- 如果不需要返回值, 函数可以没有返回参数; 同样, 函数也可以没有输入参数, 此时和命令文件功能一样。

4.3 函数文件

函数文件的创建:

菜单: **file**→**new**→**function**, 缺省函数结构如下:

```
function [ output_args ] = Untitled1( input_args )  
%UNTITLED3 Summary of this function goes here  
% Detailed explanation goes here
```

End

回顾华氏温转摄氏温度的函数:

```
function c=f2cm(f)  
% Transorm Fahrenheit temperature to celsius  
c=5*(f-32)/9  
end %end可以不要
```

4.3 函数文件

自定义函数的调用：

[output_args]=funName(input_args)

Note:

- 函数文件必须保存到搜索/当前路径下。
- 各实参出现的顺序、个数，应与定义时形参的顺序、个数一致。
- 函数调用时，先将实参传递给相应的形参，从而实现参数传递，然后再执行函数的功能。

4.3 函数文件

例4-14 编写函数文件求半径为 r 的圆的面积和周长

```
function [s,p]=fcircle(r)
% fcircle calculate the area and
% perimeter of a circle of radii r.
%r    圆半径
%s    圆面积
%p    圆周长
%version: 1.0, 2004.7.30

s=pi*r*r;
p=2*pi*r;
```

4.3 函数文件

输入及输出参数也可以是矩阵，如上题：

```
function sp=fcircle2(r)
% fcircle2 calculate the area and
% perimeter of a circle of radii r
%r        圆半径
%sp        [圆面积, 圆周长]
%version:  1.0,  2004.7.30

sp(1)=pi*r*r;
sp(2)=2*pi*r;
%equals to sp=[pi*r*r, 2*pi*r];
```

4.3 函数文件

函数的递归：一个函数调用它自身

Note: 递归函数中一般应有条件语句来终止这种递归，否则会出现无穷递归现象。

例4-16 利用函数的递归调用，求 $n!$

分析： $n!$ 本身就是以递归形式定义的：求 $n!$ 需要求 $(n-1)!$

```
function f=factor(n)
```

```
if n<=1
```

```
    f=1;
```

```
else
```

```
    f=factor(n-1)*n;    %递归调用求(n-1)!
```

```
end
```

4.3 函数文件：函数的重载

- 回顾我们熟悉的**matlab**函数，如：
`zeros size length sum`
- 查看**size**函数的帮助文件：
» `help size`
- 帮助文件列出了**size**函数的多种语法：
 - `d = size(X)`
 - `[m,n] = size(X)`
 - `m = size(X,dim)`
 - `[d1,d2,d3,...,dn] = size(X)`
- 很多系统函数调用时都可以有不同个数的输入及输出参数→函数的重载。

4.3 函数文件：函数的重载

- **Matlab** 函数都支持函数的重载：

- 可以有不同的输入参数个数
- 可以有不同的输出参数个数

- 自定义函数也可以实现重载功能：

在调用函数时，**MATLAB**用两个永久变量 记录调用该函数时的输入实参和输出实参的个数。由这两个变量的值，可以在程序中决定函数如何处理。

nargin (number of argument in): 输入参数个数

nargout (number of argument out): 输出参数个数

(see also: **varargin**, **varargout**)

4.3 函数文件：函数的重载

例4-17 输入参数重载。

函数文件：

```
function fout=examp(a,b,c)  
if nargin==1  
    fout=a;  
elseif nargin==2  
    fout=a+b;  
elseif nargin==3  
    fout=(a*b*c)/2;  
end
```

命令文件：

```
x=[1:3];  
y=[1;2;3];  
examp(x)  
examp(x,y')  
examp(x,y,3)
```

4.3 函数文件：全局变量与局部变量

局部变量：在函数内部使用的变量都是局部变量，运行结束后，将被清空。

如果希望变量被保留，或在几个函数中共用，可以申明为全局变量。

全局变量：如果若干个函数(包括基本工作空间)都将某个特定的变量名称定义为全局变量，那么这些函数都可以使用该变量。

global var1

如果这些函数中的任意一个对该变量进行了赋值运算，那么在别的函数中，这个赋值运算的结果同样会起作用。

Note： 尽量避免使用全局变量

4.4 程序的调试

两类错误:

- **语法错误**: 包括词法或文法的错误, 例如函数名的拼写错、表达式书写错等。
- **程序逻辑错误**: 程序的运行结果有错误, 即语法上没有错误, Matlab不报错, 但达不到预计的结果。

● 一般错误的查找:

➤ 通过提示查看错误:

```
>> y=sin(x2)
```

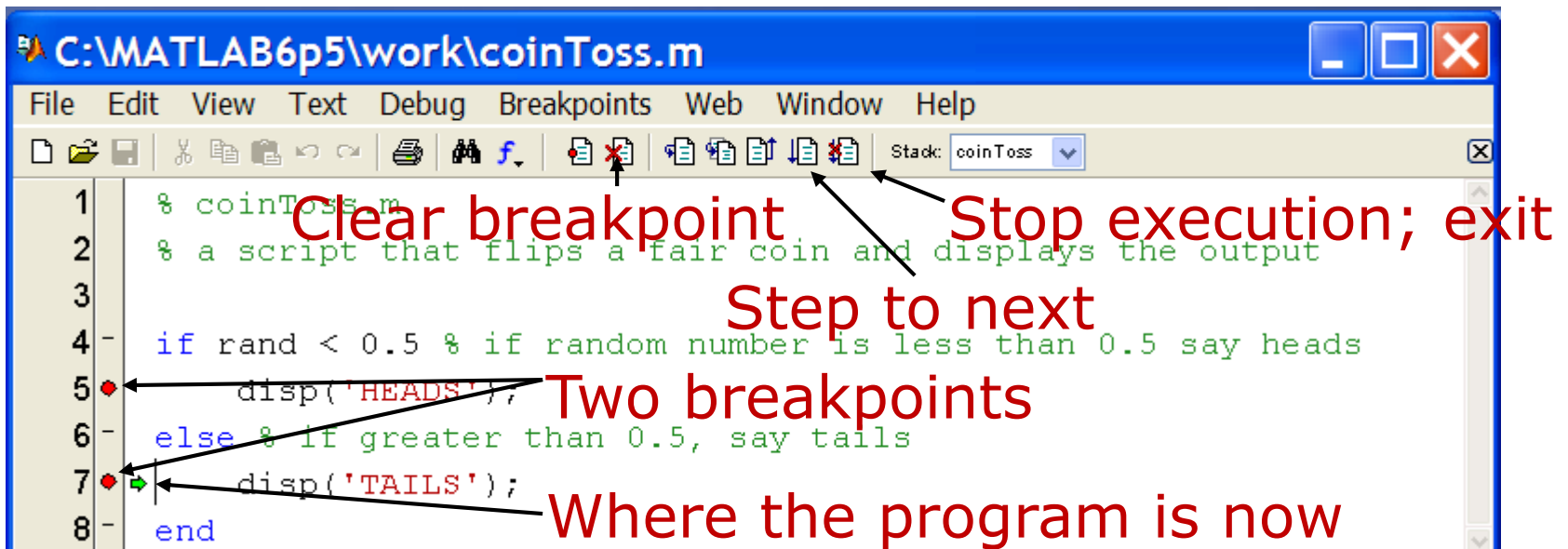
未定义函数或变量 'x2'。

➤ 通过在程序中加入**disp**查看变量值

```
disp(['循环次数: ', num2str(n)]);
```

4.4 程序的调试

- 使用调试器, 先设置断点: 程序运行到断点会暂停
 - 设置断点: 点击编辑器左侧行号右侧
 - 每个红点表示一个断点
 - 程序在断点处暂停时, 可以查看各个变量的值
 - 使用工具栏调试按钮调试



4.4 程序的调试

调试菜单

1. 控制单步运行
2. 断点操作

Debug	
✓ Open M-Files when Debugging	
Step	F10
Step In	F11
Step Out	Shift+F11
Run perfect.m	F5
Run Configuration for perfect.m	
Go Until Cursor	
Set/Clear Breakpoint	F12
Set/Modify Conditional Breakpoint...	
Enable/Disable Breakpoint	
Clear Breakpoints in All Files	
Stop if Errors/Warnings...	
Exit Debug Mode	Shift+F5

4.4 程序的调试

调试命令:

除了采用调试器调试程序外，**Matlab**还提供了一些命令用于程序调试。命令的功能和调试器菜单命令类似。

(see: p80, 表4.1)