第 3 章 MATLAB的流程结构

CHAPTER 3

前面介绍的程序语句是顺序执行的交互式编程方式，MATLAB运行机制按照给定的语句依次执行。在实际应用中，有时需要依照某些条件更改程序语句的运行顺序，或反复执行某一段代码，所以，需要引入基于流程结构的编程方式。

流程结构又称控制流程（control flow），是指计算机对语句运行顺序的控制方法。作为一种程序设计语言，MATLAB提供了循环语句结构、条件语句结构、开关语句结构以及与众不同的试探语句。本章将介绍各种各样的流程语句结构。

3.1节介绍两种不同的循环结构，并介绍迭代方法的循环结构实现与向量化编程的方法，后者的最终目标是以更高效的编程结构取代循环结构。3.2节介绍有条件的转移结构，还介绍分段函数向量化的处理方法，以便用高效的向量化编程取代转移结构。3.3节与3.4节分别介绍开关结构与试探结构。

3.1 循环结构

如果想反复执行一段代码，则需要使用循环结构。一般情况下，循环结构又分为两种形式：一种循环是事先指定次数n，然后反复n次执行一段代码，这里称其为`for`循环；另一种循环为条件循环，在给定的条件满足时反复地执行一段代码，称为`while`循环。本节将分别介绍这两种循环结构及其使用方法。本节还探讨向量化的编程方法，该方法可以作为循环结构的一种替代方法，可以提升程序运行效率。

3.1.1 for循环结构

`for`循环是最常用的一类循环结构，其一般结构为

`for` i`=`$\boldsymbol{v}$`,` 循环结构体`, end`

在标准的`for`循环结构中，$\boldsymbol{v}$为一个向量，循环变量i每次从$\boldsymbol{v}$向量中依次取一个数值，执行一次循环体的内容，再返回`for`语句，将$\boldsymbol{v}$向量中的下一个分量提取出来赋给i，再次执行循环体的内容。这样的过程一次次反复执行下去，直至执行完$\boldsymbol{v}$向量中所有的分量，将自动结束循环体的执行。

如果$\boldsymbol{v}$是矩阵，则每次i从中取一个列向量，直至提取完矩阵的所有列向量。

例3-1　先考虑一个简单的例子，用循环结构求解$S=\sum_{i=1}^{100} i$。

解 可以考虑用for循环结构求解这样的问题。将$\boldsymbol{v}$向量设定为行向量$[1,2,\cdots,100]$,并设定一个初值为0的累加变量s。让循环变量每次从$\boldsymbol{v}$向量中取一个数值,加到累加变量s上,这样可以写出如下的程序段,得出累加结果为$s=5050$。

```
>> s=0; for i=1:100, s=s+i; end, s  %简单的循环结构
```

事实上,前面的求和用sum(1:100)这样的简单命令就能够直接求解,得出完全一致的结果。这样做借助了MATLAB的sum()函数对整个向量进行直接操作,故程序更简单了。

如果用C这样的底层语言表示循环,则需要给出for (i=1; i<=100; i++)语句表示循环条件,故此要求循环条件满足有固定规律的演化过程。MATLAB循环语句中,向量$\boldsymbol{v}$为任意排列的给定向量。由此可见,这样的格式比C语言的相应格式灵活得多。下面将再给出一个例子演示for循环在迭代过程中的应用。

例3-2 假设序列第一项为$a_1=3$,第二项可以由第一项计算出来,$a_2=\sqrt{1+a_1}$,后续各项通过递推公式$a_{k+1}=\sqrt{1+a_k}$,$k=1,2,\cdots,m$计算出来,如何计算a_{32}呢?

解 这样逐项递推计算的过程可以通过一种称为“迭代”的方法实现。迭代过程的计算机语言伪代码表示为$a=\sqrt{1+a}$,其含义是$\sqrt{1+a}\Rightarrow a$,亦即在当前的a值下计算$\sqrt{1+a}$的值,再更新a的值。令$a=3$,让循环变量k执行31次,则可以计算出最新的a。可以用循环结构求解:

```
>> a=3; format long                          %设置初值并设定显示格式
   for k=1:31, a=sqrt(1+a); disp(a), end %迭代计算,每步显示结果
```

从得出的结果看,在当前的显示状况下(显示从略,读者可以自己观察显示),最后两项都是1.618033988749895,在双精度意义下是完全一致的,再继续迭代下去结果也不会有变化,可以认为这样的迭代过程是收敛的,该收敛的最终结果又称为黄金分割数[3]。

如果从数学表达式角度理解这样的序列,则可以将其前几项写成

$$3,\sqrt{1+3},\sqrt{1+\sqrt{1+3}},\sqrt{1+\sqrt{1+\sqrt{1+3}}},\cdots$$

如果这样的序列收敛,并假设其收敛到x,则由迭代公式可以得出$x=\sqrt{1+x}$。求解该方程则可以得出$x=\left(1+\sqrt{5}\right)/2\approx 1.618033988749895$。

例3-3 已知Fibonacci序列可以由式$a_k=a_{k-1}+a_{k-2}$,$k=3,4,\cdots$生成,其中,初值为$a_1=a_2=1$,试生成Fibonacci序列的前100项。

解 由Fibonacci序列的生成公式可见,可以用向量描述该序列,并令初始值$a(1)=a(2)=1$,从第三项开始就可以用递推公式计算了,生成Fibonacci序列的命令如下:

```
>> a=[1 1]; for k=3:100, a(k)=a(k-1)+a(k-2); end, a(end)
```

不过,从得出的结果看,由于双精度数据结构只能保留15位有效数字,因此这样得出的序列数值可能不完全,说明双精度数据结构在这个问题上不适用,应该采用符号型数据结构。若想使用符号型数据结构,只须修改初始值即可。将其改成a=sym([1,1]),则可以得出精确的$a_{100}=354224848179261915075$。

例3-4 例2-6中曾经说过,由命令行显示的方式最多可以显示32766个字符,其余的不能显示出来,如何用MATLAB显示π的前1000000位数字呢?

解 由于不可能在一行显示全部内容,不妨考虑将全部数字用分行的方式显示,比如每行显示10000个字符。显然,需要使用循环方式分行显示,不过分行之前应该将得出的π值由数值型变量

转换成字符串型变量,然后给出下面命令分段显示。

```
>> P=vpa(pi,1000000); str=char(P); n=10000; %转换成字符串
   for i=1:n:length(str)                      %逐行显示
      disp(str(i:min(i+n-1,length(str))));  %用disp()函数显示结果
   end
```

上面语句可以将π的值分101行显示出来,最后一行显示一位数字,因为实际转换出来的字符一共有1000001位(小数点占一位)。

for循环事先预定一个循环的执行条件,选择循环变量应该取得所有的值,然后开始循环过程,所以可以认为这样的循环是一种纯粹的循环或无条件的循环。下面将通过例子演示无条件循环的局限性。

例3-5　试求满足$S=\sum_{i=1}^{m} i>10000$的最小m值。

解　前面介绍的求和计算是已知m的值,而这里的m值是未知的,无法事先建立$\boldsymbol{v}$向量,所以,for循环这种无条件循环是不可行的,需要引入有条件循环,如后面将介绍的while循环。

3.1.2 while循环结构

while循环是另一种常用的循环结构。与for循环这种无条件的循环相比,在while循环中允许条件表达式的使用,一旦条件表达式不成立,则自动终止循环过程。while循环的典型结构为

```
while (条件式), 循环结构体, end
```

while循环中,while语句的“条件式”是一个逻辑表达式,若其值为“真”则将自动执行一次“循环结构体”,执行后返回while语句,再判定其条件式的真伪,如果为真则仍然执行循环结构体,否则将退出循环结构。如果使用的不是逻辑变量,则若表达式非零可以认为表达式为真,否则为伪,可以终止循环。

循环结构可以由for或while语句引导,用end语句结束,在这两个语句之间的部分称为循环体。这两种语句结构的用途与使用方法不尽相同,下面将通过例子演示它们的区别及适用场合。

例3-6　用while循环结构重新计算$s=\sum_{i=1}^{100} i$。

解　与for循环一样,仍然可以设定一个初值为0的累加变量s,同时,for循环中的i变量在这里也设置成一个单独的变量,并令其初值为0。while循环将判定i的值,如果$i\leqslant 100$,则将i的值累加到s上,其本身也自增1,然后返回while语句再判定条件式$i\leqslant 100$是否成立,如果成立就一直累加下去,如果不成立,就说明已经累加完100项了,程序就可以结束了。这样的思路可以由下面的MATLAB语句直接实现,得出累加的结果为$s=5050$,与前面得出的完全一致。

```
>> s=0; i=1;
   while (i<=100), s=s+i; i=i+1; end, s %不满足条件则结束循环
```

对这个具体问题而言,while循环要比for循环结构稍显麻烦。

例3-7　求出满足$s=\sum_{i=1}^{m} i>10000$的最小m值。

解 此问题用for循环结构就不便求解了，因为在做加法之前事先并不知道加到哪一项，用for循环这样的无条件循环是不可行的。求解这样的问题应该用while结构求出所需的m值，其思路是：一项项累加，在每次累加之前判定一下和s是否超过了10000，如果未超过则继续累加；如果超过则停止累加，终止循环过程。这里介绍的想法可以由下面的语句具体实现，得出的结果为$s=10011, m=141$，该结果也可以通过sum(1:m)命令检验。

```
>> s=0; m=0;     %设置初值
   while (s<=10000), m=m+1; s=s+m; end, s, m %和大于10000时终止循环
```

例3-8 例3-2中的问题采用32步迭代得出收敛的结果。如果未知迭代的步数，试用循环方式找到收敛的结果。

解 可以设置循环终止条件，如果新计算的a与前一步的a之差的绝对值小于eps，则可以终止循环。这样，由while结构很容易得出与例3-2一致的结果，且$k=32$。

```
>> a0=4; a=3; k=0; format long %设置初值并设定显示格式
   while abs(a-a0)>=eps          %设置循环条件:如果误差大则继续循环
      a0=a; a=sqrt(1+a0); k=k+1; disp(a)  %迭代计算,每步显示结果
   end, k                                 %结束循环,显示迭代步数
```

例3-9 回顾例2-38中给出的MATLAB文件源代码显示器代码。为方便起见，这里重新列出该例的代码。学习了while语句，再理解该代码就容易了。程序的主体是由while循环构成的。循环的条件表达式为~feof(h)，表示文件未结束。如未结束，则由fgetl()读文件的下一行，直至读文件结束。

```
>> [f,p]=uigetfile('*.m');     %打开文件名对话框,选择文件
   ff=[p,f]; h=fopen(ff,'r'); %生成文件名并以只读形式打开文件
   while ~feof(h), str=fgetl(h); disp(str); end %逐行读入并显示
   key=fclose(h);                               %完成后关闭文件
```

3.1.3 循环语句的嵌套

与其他编程语言一样，循环语句是可以嵌套的，即循环语句内部可以带有其他的循环结构。注意，每个循环引导词for或while都应该带有与其匹配的end语句，否则，MATLAB的执行机制将给出错误信息。当然，用这样的方式可以生成多重循环结构，也可以在循环结构中嵌入其他的流程结构，反之亦然。下面将通过例子演示循环语句的嵌套结构。

例3-10 Hilbert矩阵是一个常用的测试矩阵，其通项为$h_{ij}=1/(i+j-1), i=1,2,\cdots,n$，$j=1,2,\cdots,m$。试用循环语句生成$4\times 6$ Hilbert矩阵。

解 可以建立双重循环，即在外循环内部嵌入内循环结构。所以，Hilbert矩阵可以由下面的嵌套循环结构实现。

```
>> n=4; m=6; %指定生成矩阵的行数与列数,用双重循环生成矩阵
   for i=1:n, for j=1:m, H(i,j)=1/(i+j-1); end, end, H
```

这样生成的Hilbert矩阵为

$$\boldsymbol{H}=\begin{bmatrix} 1 & 0.5 & 0.3333 & 0.25 & 0.2 & 0.1667 \\ 0.5 & 0.3333 & 0.25 & 0.2 & 0.1667 & 0.1429 \\ 0.3333 & 0.25 & 0.2 & 0.1667 & 0.1429 & 0.125 \\ 0.25 & 0.2 & 0.1667 & 0.1429 & 0.125 & 0.1111 \end{bmatrix}$$

3.1.4 向量化编程与循环结构

在MATLAB程序中，循环结构的执行速度较慢。所以在实际编程过程中，如果能对整个矩阵或向量进行运算时，尽量不要采用循环结构，应该采用向量化方法完成任务，这样可以提高代码的效率。

向量化编程是MATLAB程序设计中引人注意的问题，向量化编程的使用会使得MATLAB程序具有美感，而过多使用循环的程序会被业内人士认为代码质量不高。下面将通过例子演示循环与向量化编程的区别。

例3-11　假设有一组圆，其半径分别为$r=1.0,1.2,0.9,0.7,0.85,0.9,1.12,0.56,0.98$，试求这些圆的面积。

解　圆面积公式为$S=\pi r^2$，有C语言基础的MATLAB初学者可能给出下面命令：

```
>> r=[1.0,1.2,0.9,0.7,0.85,0.9,1.12,0.56,0.98];
   for i=1:length(r), S(i)=pi*r(i)^2; end, S %通过循环逐一计算
```

这些命令可以正确地计算出这组圆的面积，不过这不是地道的MATLAB编程。如果使用MATLAB的向量化编程结构，则上面一整行循环语句应该替换成如下的一条语句，得出的结果与前面是完全一致的，但程序漂亮得多。

```
>> S=pi*r.^2 %向量化编程，可以避免循环，结构更简洁
```

例3-12　求解级数求和问题$S=\sum_{i=1}^{10000000}\left(\frac{1}{2^i}+\frac{1}{3^i}\right)$。

解　对这个例子而言，可以仿照例3-1用循环语句直接实现，得出的和为1.5，总耗时为2.079 s。

```
>> N=10000000;
   tic, s=0; for i=1:N, s=s+1/2^i+1/3^i; end; toc %普通循环运算
```

如果构造一个i行向量，则$1/2^i$的数学表达式可以由点运算`1./2.^i`命令实现，结果仍然是一个行向量，同理可以将数学表达式$1/3^i$用向量`1./3.^i`表示。将得出的向量`1./2.^i+1./3.^i`逐项加起来，最简洁的方法是调用`sum()`函数。这样做就可以避开循环，由向量化的方式得出问题的解。这段代码得出的和仍然为1.5，耗时减为0.566 s。

```
>> tic, i=1:N; s=sum(1./2.^i+1./3.^i); toc %向量化编程
```

对这个例子而言，向量化编程的效率明显高于循环结构。其实，MATLAB新版本对循环结构的效率已经有了大幅提升，如果在早期版本下比较两种方法，差距将更为悬殊。

例3-13　MATLAB函数提供的`meshgrid()`函数可以生成二维甚至三维网格数据。试观察该函数生成网格数据的形式。

解　假设横坐标设定为$\boldsymbol{v}_1=[1,2,4,3,5,7,9]$，纵坐标选作$\boldsymbol{v}_2=[-1,0,2]$，则由下面的语句可以调用`meshgrid()`函数，生成两个网格矩阵。

```
>> v1=[1 2 4 3 5 7 9]; v2=[-1 0 2]; %横坐标纵坐标划分
   [x,y]=meshgrid(v1,v2)              %直接生成网格，构造两个网格矩阵
```

得出网格矩阵如下。可以看出，这两个矩阵可以描述二维网格上每个网格点的坐标。

$$\boldsymbol{x}=\begin{bmatrix}1&2&4&3&5&7&9\\1&2&4&3&5&7&9\\1&2&4&3&5&7&9\end{bmatrix},\ \boldsymbol{y}=\begin{bmatrix}-1&-1&-1&-1&-1&-1&-1\\0&0&0&0&0&0&0\\2&2&2&2&2&2&2\end{bmatrix}$$

例3-14 例3-10介绍了Hilbert矩阵的生成。如果想生成50000×50的大型Hilbert矩阵，应该如何输入？

解 很显然，利用下面的双重循环就可以直接构造所需的矩阵，耗时35.52 s。

```
>> tic, for i=1:50000, for j=1:50, H(i,j)=1/(i+j-1); end, end, toc
```

现在考虑50000×500的矩阵，使用上面的方法过于耗时，是不能生成这样矩阵的。如果调换循环的次序，将大循环移入内层，则耗时降至0.334 s。可见，对同样的问题而言，可以将大循环移至内层，小循环移至外层，会使得效率大大地提高。

```
>> tic    %大循环移至内层重新运行
   for j=1:500, for i=1:50000, H1(i,j)=1/(i+j-1); end, end, toc
```

如果内层循环由向量化语句取代，则耗时会进一步减少至0.26 s。

```
>> tic, for j=1:500, i=1:50000; H2(i,j)=1./(i+j-1); end, toc
```

还可以由meshgrid()函数生成网格，再由向量化的方式取代双重循环，则耗时将减少至0.16 s。不过与循环相比，向量化更耗费存储资源(本例多生成两个大矩阵)。

```
>> tic, [i,j]=meshgrid(1:50000,1:500); H3=1./(i+j-1); toc
```

例3-15 如果x、y都是正整数，试求解代数方程$x^2+y^3=80893009$，并验证结果。另外，该方程的解唯一吗？(习题2.25)

解 在求解某些问题时，如果变换一个角度思考，则可能用简单方式得出问题的解。由于需要正整数解，即使y为1，x的最大值也不能超过$\sqrt{80893009}$，同理，y不能超过$\sqrt[3]{80893009}$。所以可以由meshgrid()函数构造所有可能的x、y组合。由下面的语句可以得出结论：$x=521$，$y=432$是方程的唯一正整数解。

```
>> N=80893009; [x y]=meshgrid(1:sqrt(N),1:N^(1/3)); %生成所有可能
   K=x.^2+y.^3; ij=find(K==N);   %找出满足方程的全部组合
   x=x(ij), y=y(ij), x.^2+y.^3-N %方程解的检验
```

3.2 转移结构

条件转移结构是一般程序设计语言都支持的结构。通常条件转移语句通过判定条件决定到底执行哪个程序分支，在不同的条件下执行不同的任务。MATLAB下的最基本的转移结构是if … end型的，也可以配合else语句与elseif语句扩展转移语句。本节将介绍各种条件转移结构，并通过例子介绍其应用方法。

3.2.1 简单的条件转移结构

最简单的条件转移结构为

```
if (条件表达式), 语句段落, end
```

其中“条件表达式”是一个逻辑表达式。这种条件转移语句的物理意义为：如果条件表达式为真，则执行“语句段落”中的程序段，执行完成后，转移到end关键词的后面继续执行。如果条件表达式不成立，则直接跳过语句段落后继续执行。

另一种简单的条件转移语句结构为

```
if (条件表达式), 段落1, else, 段落2, end
```

类似于前面的最简单结构，如果“条件表达式”为真，则执行“段落1”，之后完成此结构；如果“条件表达式”不成立，则执行“段落2”，执行完成后，转移到end关键词后继续执行。

3.2.2 条件转移结构的一般形式

除了前面给出的简单条件转移结构之外，MATLAB还支持一般的条件转移结构，其相应的语句格式为

```
if（条件1）      %如果条件1满足，则执行下面的语句组1
    语句组1      %这里也可以嵌套下级的if结构
elseif（条件2）%否则，如果满足条件2，则执行下面的语句组2
    语句组2
    ⋮          %可以按照这样的结构设置多种转移条件
else           %上面的条件均不满足时，执行下面的语句组
    语句组n+1
end
```

3.2.3 其他流程控制命令

对循环结构而言，除了for和while语句之外，还可以使用break和continue等语句控制循环结构，其中，break语句强制中断上一层的循环；continue语句略去本次循环语句段剩余的部分，返回for或while引导语句。这些语句可以配合if或其他流程控制语句执行。下面将通过例子演示break语句的应用。

例3-16　用for循环和if语句的形式重新求解例3-7的问题。

解　例3-7中提及只用for循环结构不便于实现求出和式大于10000的最小m值，利用该结构必须配合if语句结构才能实现。其具体的思路是：让m在一个大范围内取值做for循环，将结果累加到s上，累加后判定和式是不是大于10000，若不是则继续循环；若是则给出break命令，强行退出循环结构，得出所需结果。求解问题具体的MATLAB命令如下：

```
>> s=0; for m=1:10000, s=s+m; if s>10000, s, m, break; end, end
```

可见，对本例而言这样的结构较烦琐，不如直接使用while结构直观、方便。

3.2.4 分段函数的向量化计算

考虑下面给出的典型一元分段函数：

$$y=f(x)=\begin{cases} f_1(x), & x\geqslant 2 \\ f_2(x), & \text{其他} \end{cases}$$

如果自变量x是一个由数据点构成的向量$\boldsymbol{x}$，如何求出相应的函数值向量$\boldsymbol{y}$呢？对有C语言基础的MATLAB初学者而言，很自然会想到用循环结构处理每一个x_i值，再用if语句对其分类，得出函数的值。

其实，如果读者能理解$\boldsymbol{x}$>=2这个语句是什么含义，则可以采用更简洁的方法处理这个分段函数。语句$\boldsymbol{x}$>=2将生成一个与$\boldsymbol{x}$等长的向量，其元素为0和1，满足$x_i\geqslant 2$的一些点对

应的值标为1，其余的为0。除此之外，还需要确定性地表示出“其他”。从逻辑上理解，“其他”应该是不满足$x_i \geqslant 2$的条件，亦即$x_i < 2$。这两个条件可以看成两个事件，如果一个发生则另一个肯定不发生，这样的关系称为互斥。

有了互斥的逻辑条件，则可以用向量化方法计算出分段函数的值：

$\boldsymbol{y}$=$f_1(\boldsymbol{x})$.*($\boldsymbol{x}$>=2) + $f_2(\boldsymbol{x})$.*($\boldsymbol{x}$<2)

这样做的好处是可以避免循环与条件转移语句，用简单的点乘运算就可以直接计算分段函数的值。这样的方法还可以直接用于多元分段函数的数值计算，不过计算之前一定要确认逻辑条件是互斥的，否则可能得出错误的结果。

此外，MATLAB提供的piecewise()函数可以在符号型数据结构框架下直接定义分段函数，该函数的调用格式为

f=piecewise(条件1，函数1，条件2，函数2，$\cdots$，条件m，函数m)

其中“条件i”为第i段分段函数的条件表达式，“函数i”为其函数表达式，使用时需要确保条件与函数表达式成对出现，否则将给出错误信息。

例3-17 试用MATLAB表示饱和非线性函数 $y=\begin{cases} 1.1\,\mathrm{sign}(x), & |x|>1.1 \\ x, & |x|\leqslant 1.1 \end{cases}$

解 如果$|x| \leqslant 1.1$在数学上表示成$-1.1 \leqslant x \leqslant 1.1$，也可以将其理解成$x \geqslant -1.1$且$x \leqslant 1.1$，这时相应的符号表达式表示应该为$x$>=$-1.1$&$x$<=1.1。

若生成一个样本点向量$\boldsymbol{x}$，则可以由点运算的方式描述$\boldsymbol{y}$向量，由下面的语句计算出函数值：

```
>> x=-2:0.01:2; y=1.1*sign(x).*(abs(x)>1.1)+x.*(abs(x)<=1.1);
```

在符号运算的框架下也可以描述分段函数：

```
>> syms x    %在符号运算框架下输入分段函数
   f(x)=piecewise(abs(x)>1.1,1.1*sign(x),abs(x)<=1.1,x);
```

这里介绍的分段函数向量化表示方法同样适用于二维甚至多维$\boldsymbol{x}$变量的情形，下面将通过例子演示二维分段函数的计算方法。

例3-18 假设某联合概率密度函数由下面分段函数表示[17]：

$$p(x_1,x_2)=\begin{cases} 0.5457\mathrm{e}^{-0.75x_2^2-3.75x_1^2-1.5x_1}, & x_1+x_2>1 \\ 0.7575\mathrm{e}^{-x_2^2-6x_1^2}, & -1<x_1+x_2\leqslant 1 \\ 0.5457\mathrm{e}^{-0.75x_2^2-3.75x_1^2+1.5x_1}, & x_1+x_2\leqslant -1 \end{cases}$$

试在$-2 \leqslant x,y \leqslant 2$范围内生成网格数据，并计算网格点上的函数值。

解 这个分段函数当然可以由双重循环描述，不过更简洁的方法是使用上面介绍的向量化方法描述，避免使用循环。虽然对小规模问题而言，这样的方法未必显著增加计算速度，但可以使程序结构显得更美观。

```
>> [x1,x2]=meshgrid(-2:0.1:2); %生成网格，两个坐标轴同样划分
   p=0.5457*exp(-0.75*x2.^2-3.75*x1.^2-1.5*x1).*(x1+x2>1)+...
     0.7575*exp(-x2.^2-6*x1.^2).*(-1<x1+x2 & x1+x2<=1)+...
     0.5457*exp(-0.75*x2.^2-3.75*x1.^2+1.5*x1).*(x1+x2<=-1);
```

3.3　开关结构

开关结构是一种重要的流程结构，其作用就像在电路中安装的多路开关一样。当开关拨至一个挡位时，其对应的回路接通，其他回路切断。

开关结构可以用if、elseif、end结构实现。例如，可以使用下面的语句：

```
if key == 表达式1, 段落1; elseif key == 表达式2, 段落2; ···, end
```

不过这样的语句可读性是比较差的，所以应该引入开关语句结构。典型的开关语句的基本结构为

```
switch 开关表达式
   case 表达式1,语句段1
   case {表达式2,表达式3,···,表达式m}, 语句段2
      ⋮
   otherwise, 语句段n
end
```

其中，开关语句的关键是对“开关表达式”值的判断，当开关表达式的值等于某个case语句后面的条件时，程序将转移到该组语句中执行，执行完成后程序转出开关体继续向下执行。

在使用开关语句结构时应该注意下面几点：

（1）当开关表达式的值等于表达式1时，将执行语句段1，执行完语句段1后将转出开关体，而无须像C语言那样在下一个case语句前加break语句，本结构与C语言是不同的。

（2）当开关表达式满足若干个表达式之一就执行某一程序段时，则应该把这样的一些表达式用花括号括起来，中间用逗号分隔。事实上，这种结构是单元数组表示。

（3）当前面枚举的各个表达式均不满足时，则将执行otherwise语句后面的语句段，此语句等价于C语言中的default语句。

（4）程序的执行结果和各个case语句的次序是无关的。当然，这也不是绝对的，当两个case语句中包含同样的条件时，执行结果则和这两个语句的顺序有关。

（5）在case语句引导的各个表达式中，不要用重复的表达式，否则列在后面的开关通路将永远也不能执行。

开关结构与条件转移结构从本质上看都属于条件转移结构——在满足条件的前提下转移到程序的某个模块执行，它们之间又有什么区别呢？从满足的条件看，条件转移语句经常会以不等式作为条件，如if $x>0$，可以认为是连续的，而开关结构的所有表达式都是可枚举的离散点，所以在条件的表示上可见，二者的适用范围是不同的。

例3-19　试编写一个求圆(球)周长、面积和体积的程序。

解　可以用第2章介绍的input()函数输入圆的半径，然后，由menu()函数建立菜单，允许用户自行选择计算周长、面积还是体积。选择key之后，程序会根据选择作自动计算。由于使用了点运算，所以可以同时计算一组圆(球)的相应信息。

```
>> r=input('输入半径 r=');          %输入圆的半径
   key=menu('选择任务','计算圆周长','计算圆面积','计算球体积');
   switch key                        %根据key的值完成计算任务
      case 1, Result=2*pi*r          %圆的周长计算公式l = 2πr
      case 2, Result=pi*r.^2         %圆的面积计算公式S = πr^2
      case 3, Result=4*pi*r.^3/3 %球的体积计算公式V = 4πr^3/3
   end
```

3.4 试探结构

MATLAB语言提供了一种新的试探式语句结构，其调用格式如下：

try，语句段1，catch，语句段2，end

本语句结构首先试探性地执行“语句段1”，如果在此段语句执行过程中出现错误，则将错误信息赋给保留的lasterr变量，并终止这段语句的执行，转而执行“语句段2”中的语句；如果执行“语句段1”不出错，则执行完之后，整个结构就结束了，不再执行“语句段2”中的语句了。

试探性结构在实际编程中还是很实用的。例如，可以将一段不保险但速度快的算法放到try段落中，而将一个保险的但速度慢的程序放到catch段落中，这样就能保证原始问题的求解更加可靠，且可能使程序高速执行。该结构的另外一种应用是，在编写通用程序时，某算法可能出现失效的现象，这时在catch语句段说明错误的原因。此外，这种试探性结构还经常用于错误陷阱的设置与处理。

例3-20 如果MATLAB工作空间中有一个a变量，试判定该变量是否为数值。

解 MATLAB有两种表示数值a的方法：第一种是用双精度形式表示；第二种是用符号型表达式表示。MATLAB的isnumeric(a)是可以识别出来第一种表示形式的，如果一个数值是由符号型变量给出的，如sqrt(sym(2))，则isnumeric()将返回0，与所期望的相反。所以，用isnumeric()并不能作出正确判定。

为解决这个问题可以编写出一个MATLAB函数，其结构将在第4章中详细介绍。首先提取a的数据结构，如果为双精度变量，则令key为1后结束程序调用；如果a为符号型变量，则可以尝试对其作双精度转换。如果成功，说明a是数值型符号变量，令key为1后返回，否则，说明a不是数值型数据，执行double()函数将出错，转到catch语句后返回，将key置0；如果不是这两种数据结构，则置key为0。

```
function key=isnumber(a)
switch class(a)              %提取输入变元的数据结构
   case 'double', key=1; %若为双精度则置key为1
   case 'sym', try, double(a); key=1; catch, key=0; end
   otherwise, key=0;         %其他数据类型则置key为0
end
```

这里比较关键的一步是对符号变量调用double()函数，该函数调用有时不能成功。什么时候不成功的呢？如果a不是数值型变量，则调用double()函数时会出现错误，这时try结构将终止，转到catch段落执行，将key置为0，说明a不是数值型变量。

3.5 习 题

3.1 试生成一个100×100的魔方矩阵,试分别用循环和向量化的方法找出其中大于1000的所有元素,并强行将它们置0。

3.2 例3-2循环固定执行31步。假设想让迭代循环在得到最精确结果时自动停止,可以设置停止条件,如迭代前后a值的差小于eps。试用语句实现这个想法。

3.3 试用循环结构由底层命令找出1000以下所有的质数。如果不采用底层的循环结构,还有什么解决问题的方法?

3.4 前面叙述中介绍过,for循环中的$\boldsymbol{v}$可以取作矩阵,试分析下面的语句,观察其执行结果,解释$\boldsymbol{v}$为矩阵时的执行过程。

```
>> A=magic(9) %生成并显示一个魔方矩阵
   for i=A, i, end
```

3.5 可以由$\boldsymbol{A}$=rand(3,4,5,6,7,8,9,10,11)命令生成一个多维的伪随机数数组。试判定一共生成了多少个随机数,这些随机数的均值是多少。

3.6 用数值方法可以求出$S=\sum_{i=0}^{63}2^i=1+2+4+8+\cdots+2^{62}+2^{63}$,试不采用循环的形式求出和式的数值解。由于数值方法采用double形式进行计算,难以保证有效位数字,所以结果不一定精确。试采用符号运算的方法求该和式的精确值。

3.7 试构造符号表达式

$$f(x)=\sqrt{x+\sqrt{x+\sqrt{x+\sqrt{x+\sqrt{x+\sqrt{x+\sqrt{x}}}}}}}$$

如果根号重数增至30,试重新表示其表达式。

3.8 若$f(x)=x^2-x-1$,试求$F(x)=f(f(f(f(f(f(f(f(f(f(x)))))))))$。如果结果是多项式,多项式的最高阶次是多少?该结果过于冗长,用普通方法不能显示全部内容,试仿照例3-4,利用循环结构显示其全部内容。提示,应该跳过vpa()函数,直接由char()函数将结果转换成字符串。

3.9 给出阶次n,试将下面矩阵输入计算机。

$$\boldsymbol{A}=\begin{bmatrix}1&-2&4&\cdots&(-2)^{n-1}\\0&1&-2&\cdots&(-2)^{n-2}\\0&0&1&\cdots&(-2)^{n-3}\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&1\end{bmatrix}$$

3.10 已知某迭代序列$x_{n+1}=x_n/2+3/(2x_n)$,$x_0=1$,并已知该序列当n足够大时将趋于某个固定的常数,试选择合适的n,求该序列的稳态值(达到精度要求10^{-14}),并找出精确的数学表示。

3.11 n阶Pascal矩阵的第一行与第一列的值都是1,其余元素可以按照下式计算:

$$p_{i,j}=p_{i,j-1}+p_{i-1,j},\ i=2,3,\cdots,n,\ j=2,3,\cdots,n$$

试编写一段底层的循环程序生成任意阶的Pascal矩阵,并与pascal()函数得出的结果相比较,检验正确性及其执行效率。

3.12 下面对例3-7的语句稍加调整，试判定调整后该程序是否仍能得出正确的结果，为什么？

```
>> s=0; m=1;     %设置初值
   while (s<=10000), s=s+m; m=m+1; end, s, m %和大于10000时终止循环
```

3.13 已知某迭代公式为 $x_{k+1}=(x_k+2/x_k)/2$，任取一个初值 x_0，并设置一个停止条件结束迭代过程，试观察该迭代公式将收敛到什么值。

3.14 已知 $\arctan x=x-x^3/3+x^5/5-x^7/7+\cdots$。取 $x=1$，则立即得出下面的计算公式：

$$\pi\approx 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right)$$

试利用循环累加的迭代方法计算出圆周率π的近似值，要求精度达到 10^{-6}。

3.15 试用下面的方法编写循环语句近似地用连乘的方法计算π值，当乘法因子 $|\delta-1|<\epsilon$ 时停止循环。选择精度 $\epsilon=10^{-15}$，试得出精确的π值并评价结果，并与习题3.14中的代码比较求解效率。

$$\frac{2}{\pi}\approx\frac{\sqrt{2}}{2}\times\frac{\sqrt{2+\sqrt{2}}}{2}\times\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\times\cdots$$

3.16 矩阵的正弦函数可以由幂级数展开式

$$\sin\boldsymbol{A}=\sum_{k=0}^{\infty}(-1)^k\frac{\boldsymbol{A}^{2k+1}}{(2k+1)!}=\boldsymbol{A}-\frac{1}{3!}\boldsymbol{A}^3+\frac{1}{5!}\boldsymbol{A}^5+\cdots \tag{3-1}$$

试用近似的方法求出给定方阵的正弦函数，如果累加项的范数小于 10^{-10} 即可以停止累加。提示，可以采用测试矩阵 $\boldsymbol{A}$=magic(5)，并与funm($\boldsymbol{A}$,@sin)命令的结果进行比较。

3.17 试求 $S=\prod_{n=1}^{\infty}\left(1+\frac{2}{n^2}\right)$，使计算精度达到 $\epsilon=10^{-12}$ 级。

3.18 试求出多项式 $\prod_{k=1}^{10}(x^k+2k)$，并得出其展开式。

3.19 假设已知乘积序列的通项为 $a_k=(x+k)^{(-1)^k}$，试求 $a_1a_2\cdots a_{40}$。

3.20 试计算扩展Fibonacci序列的前300项，其中 $T(n)=T(n-1)+T(n-2)+T(n-3)$，$n=4,5,\cdots$，且初值为 $T(1)=T(2)=T(3)=1$。

3.21 用MATLAB实现抛硬币实验，并由实验方法求出抛100000次硬币，正面朝上的概率。提示，由于硬币正、反两面朝上的概率均等，所以可以考虑生成100000个[0,1]区间均匀分布的为随机数，看有多少大于0.5的，将其设定为正面朝上。生成随机数等于0.5的概率微乎其微，可以忽略不计。

3.22 分形树的数学模型：任意选定一个二维平面上的初始点坐标 (x_0,y_0)，假设可以生成一个在 $[0,1]$ 区间上均匀分布的随机数 γ_i，那么根据其取值，可以按下面的公式生成一个新的坐标点 (x_1,y_1)[18]。

$$(x_1,y_1)\Leftarrow\begin{cases}x_1=0, & y_1=y_0/2, & \gamma_i<0.05\\ x_1=0.42(x_0-y_0), & y_1=0.2+0.42(x_0+y_0), & 0.05\leqslant\gamma_i<0.45\\ x_1=0.42(x_0+y_0), & y_1=0.2-0.42(x_0-y_0), & 0.45\leqslant\gamma_i<0.85\\ x_1=0.1x_0, & y_1=0.2+0.1y_0, & \text{其他}\end{cases}$$

试判定这里的递推计算可以由向量化方法实现吗？为什么？

3.23 Lagrange插值算法是一般代数插值教材中经常介绍的一类插值算法[10]，对已知的x_i、y_i点，可以求出$\boldsymbol{x}$向量上各点处的插值为

$$\phi(\boldsymbol{x})=\sum_{i=1}^{m}y_i\prod_{j=1,j\neq i}^{m}\frac{\boldsymbol{x}-x_j}{(x_i-x_j)}$$

试编写一段代码实现Lagrange插值(提示:下面给出了一段代码,可供参考)。

```
>> ii=1:length(x0); y=zeros(size(x));     %生成插值的初值向量
   for i=ii, ij=find(ii~=i); y1=1;         %剔除向量中当前值
      for j=1:length(ij), y1=y1.*(x-x0(ij(j))); end %连乘运算
      y=y+y1*y0(i)/prod(x0(i)-x0(ij));    %作外环的累加处理
   end
```

3.24 Monte Carlo方法是一种常用的统计试验方法。考虑在边长为1的正方形内投入N个均匀分布的随机数点，则如果N足够大，π的值可以由$\pi\approx 4N_1/N$近似，其中，N_1是落入半径为1的四分之一圆内的随机数点个数。试选择不同的N值，观察π值的近似效果。提示，下面给出了Monte Carlo算法的语句，以供参考。

```
>> N=100000; x=rand(1,N); y=rand(1,N);
   i=(x.^2+y.^2)<=1; N1=nnz(i); p=N1/N*4
```

3.25 随机整数方阵可以由$\boldsymbol{A}$=randi([a_{m},a_{M}],n)命令直接生成，试找出一个数值在$-8\sim 8$的4×4方阵，使得其行列式的值等于1。有没有可能构造这样的一个复数矩阵呢？提示:行列式可以由det()函数计算，为保证能精确计算小规模矩阵的行列式，建议使用符号运算的方法。

3.26 试用循环结构重新求解例3-15中的方程，并比较求解效率。

3.27 若某个3位数，每位数字的三次方的和等于其本身，则称其为水仙花数，试找出所有水仙花数。如果不使用循环能求出所有的水仙花数吗？

3.28 例3-20给出的判定方法是有漏洞的，因为只考虑双精度与符号变量表示数值，未考虑其他数据结构，如single、uint8等。如果a为这些数据结构，该函数返回的结果是错误的。试修改该函数，使其能正确处理这些数据结构。有没有更简洁的解决方案？