MATLAB 中文论坛常见问题归纳

前言

论坛经常会碰到一些重复出现的问题，这里总结归纳了一些比较常见的基础问题。一方面可以避免作重复性回答，另一方面为 MATLAB 初学者提供一些常见问题的解决办法，希望对大家有所帮助。感谢@winner245 版主对内容提供的一些建议，如果你觉得还有什么值得归纳的问题，也希望能够提出来。

一、求解方程

求解方程通常有两种方法，符号求解和数值求解。

1、solve

通常在不确定方程是否有符号解的时候，推荐先使用 solve 进行尝试，因为 solve 相比于数值求解来说，它不需要提供初值，并且一般情况下能够得到方程的所有解。对于一些简单的超越方程，solve 还能够自动调用数值计算系统给出一个数值解。

solve 的调用形式：

sol=solve(eq)

sol=solve(eq,var)

sol=solve(eq1,eq2,…,eqn)

sol=solve(eq1,eq2,…,eqn,var1,var2,…,varn)

eq 为符号表达式，var 为指定的要求解的变量。如果不声明要求解的变量(第一和第三种形式)，则 matlab 自动按默认变量进行求解，默认变量可以由 symvar(eq)确定。

例：求解方程组: x+y=1, x-11y=5

```
syms x y  %声明符号变量
eq1=x+y-1
eq2=x-11*y-5
sol=solve(eq1,eq2,x,y)
x=sol.x
y=sol.y
```

这时候solve求得的解通过结构体的形式赋值给sol，然后再通过x=sol.x和y=sol.y分别赋值给x和y。

也可以直接使用:

```
[x,y]=solve(eq1,eq2,x,y)
```

进行求解，但这时需要注意，等式左边接收参数时应当按字母表进行排序，否则 MATLAB 不会自动识别你的参数顺序，比如：

```
[x,y]=solve(eq1,eq2,x,y)
[y,x]=solve(eq1,eq2,x,y)
```

solve 会把答案按字母表进行排序后进行赋值，x 解赋值给第一个参数，y 解赋值给第二个参数，那么对于第二种形式，实际上最终结果是变量 y 存储了 x 的解而变量 x 存储了 y 的解。

由于是符号求解，有时候得到的解是一大串式子(符号求解无精度损失，所以 MATLAB 不会自动将答案转化为浮点数)，这时候可以用 vpa 或者 double 函数将结果转换为单一的数。

另外很多人习惯对于 solve 的参数采用字符型输入，这种方式有几个弊端，首先就是程序的调试，一旦式子输入有误(最常见的就是括号的配平)，将会对程序调试带来很大的困难，一个典型的例子就是：

```
solve('10^(-4.74)*0.965*y/60000x/(10^(-4.74)+x)+0.1/36500+10^(-14)/x-
x=0','10^(-3.2)*x+0.333/3000+8*10^((-3.2)*0.1+0.1/333*y','x','y')
```

这时候要去寻找式子的输入错误时将会是一件很头痛的事，MATLAB也不会报告具体出错的地方。

但是如果采用符号变量输入：

```
syms x y
eq1=10^(-4.74)*0.965*y/60000x/(10^(-4.74)+x)+0.1/36500+10^(-14)/x-x
eq2=10^(-3.2)*x+0.333/3000+8*10^((-3.2)*0.1+0.1/333*y
sol=solve(eq1,eq2,x,y)
```

这时候对于程序的调试会带来许多便利，对于某些错误MATLAB会给出错误代码颜色的高亮， 命令行还能返回具体的错误信息；

其次是采用字符型输入时，对变量的赋值并不能传入方程，以x+y*sin(x)=1这个方程为例：

```
y=1;
sol=solve('x+y*sin(x)=1','x')
```

MATLAB会返回一个空解，但是对于sym型输入：

```
syms x
y=1
eq=x+y*sin(x)-1
sol=solve(eq,x)
```

能够得到sol=0.510973429388569109520013971145 08，其中的区别就在于char型输入尽管在solve前对y有一个赋值，但solve求解时依然会将y当作一个未赋值的常数。

最后，在今后的高版本solve将不支持char型参数输入，因此应该尽量放弃使用这种方法。

论坛关于 solve 的问题：

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=434328

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=436167

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=281433

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=268347

2、fzero

然而在很多情况下solve并不能求得方程的解析解，这时就可以采用数值法求解。

数值求解法包括fzero和fsolve，其区别在于fzero只适用求解一元函数零点，而fsolve适用于求解多元函数零点(包括一元函数)。

当求解一元函数零点时，推荐优先使用fzero，原因是fzero求解一元方程往往更容易，因为它不仅支持提供初值的搜索，还支持在一个区间上进行搜索。

fzero的常用形式：

```
x = fzero(fun,x0)
[x,fval] = fzero(fun,x0)
```

其中fun为函数句柄，x0为搜索初值，fval为求解误差。

以一元方程sin(x)+cos(x)^2=0为例：

```
y=@(x)sin(x)+cos(x).^2      %这里采用匿名函数，也可以使用函数文件形式
```

```
[x,fval]=fzero(y,1)  %1为搜索初值
```

如果方程有多个零点时，fzero只能根据你提供的初值求得最靠近初值的一个零点，如果希望求得多个零点的话，那么只能够通过改变初值来得到不同的零点。

对于零点的选取，目前来说没有什么比较好的办法，只能够通过分析方程的性质，或者通过作图的方法去寻找一个比较靠近零点的初值。另外，fzero能够提供区间搜索，注意区间两端的端点函数值符号需要反向：

```
y=@(x)sin(x)+cos(x).^2
[x,fval]=fzero(y,[-1 1])      %fzero在[-1,1]这个区间搜索初值
```

除此之外，fzero还能够求解积分方程：

http://www.ilovematlab.cn/thread-333673-1-1.html

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=439192

论坛关于fzero的问题：

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=434434

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=319218

3、fsolve

fsolve可以求解多元方程，用法和fzero类似。

fsolve的常用形式：

```
x = fsolve(fun,x0)
[x,fval] = fsolve(fun,x0)
```

其中fun为函数句柄，x0为搜索初值，fval为求解误差

例：求解方程组 x+y=1, x-11y=5

```
eq=@(x)[x(1)+x(2)-1;x(1)-11*x(2)-5]
[sol,fval]=fsolve(eq,[1,1])
```

这里对于方程的的输入需要采用矩阵的形式，其中 x(1)代表 x，x(2)代表 y。有时候变量较多时可能会容易混淆，这里提供另一种方法，采用符号变量形式再利用 matlabFunction 转化为函数句柄：

```
syms x y
eq1=x+y-1
eq2=x-11*y-5
eq1=matlabFunction(eq1);   %将符号函数转化为函数句柄
eq2=matlabFunction(eq2);
eq=@(x)[eq1(x(1),x(2));
eq2(x(1),x(2))]
[sol,fval]=fsolve(eq,[1,1])
```

效果与之前相同，但不容易出错。求得的解以矩阵形式返回给 sol，即 sol 的第一个值是匿名函数的第一个输入参数值 x，sol 的第二个值是匿名函数的第二个输入参数值 y。

论坛关于fsolve的问题：

http://www.ilovematlab.cn/thread-438375-2-1.html

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=274668

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=273545

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=320750

4、vpasolve

最后再补充一个数值解法 vpasolve，vpasolve 是 R2012b 引进的函数，可以求解一元或多元函数零点。相比于 fzero 和 fsolve 来说，vpasolve 最大的一个优点就是不需要提供初值，并且能够自动搜索指定范围内的多个解。

vpasolve 调用形式：

```
S = vpasolve(eqn)
S = vpasolve(eqn,var)
S = vpasolve(eqn,var,init_guess)
___ = vpasolve(___,Name,Value)
```

其中eqn是符号方程，var为需要求解的变量，也可以不提供(第一种形式，这是默认求解变量由symvar(eqn)求得)，init_guess为搜索初值，Name,Value为一些选项控制。

例：对于多项式方程，vpasolve能够给出所有解：

```
syms x
vpasolve(4*x^4 + 3*x^3 + 2*x^2 + x + 5 == 0, x)

ans =

 - 0.88011 - 0.76332i
   0.50511 + 0.81599i
   0.50511 - 0.81599i
 - 0.88011 + 0.76332i
```

对于非多项式方程，vpasolve给出它找到的第一个解：

```
syms x
vpasolve(sin(x^2) == 1/2, x)

ans =

-226.94
```

这时可以提供搜索初值，来改变它找到的解：

```
syms x
vpasolve(sin(x^2) == 1/2, x,100)

ans =

99.996
```

可以指定搜索范围，但不同于solve，solve指定求解范围是用assume函数，vpasolve则是直接在输入参数中指定：

```
syms x
vpasolve(x^8 - x^2 == 3, x, [-Inf Inf]) %实数范围内求解
```

最后，vpasolve一个很强大的用法，将'random'选项设置为true可以直接搜索指定范围内不同解：

```
syms x
```

```
f = x-tan(x);
for n = 1:3
  vpasolve(f,x,'random',true)
end
```

二、求解积分

求解积分与求解方程相同，也有两种方法，符号求解和数值求解。

1、int

int 是符号积分求解器，调用形式简单，但是功能非常强大。
int 常用形式：

int(expr,var)　%不指定积分上下限，即求解不定积分
int(expr,var,a,b)　%指定积分上下限，即求解定积分

例：求解不定积分 $\int \frac{5}{(x-1)(x-2)(x-3)}dx$

```
syms x
f=5/((x-1)*(x-2)*(x-3))
F=int(f,x)
```

例：求解定积分 $\int_0^1 \frac{x}{1+y\wedge 2}dy$

```
syms x z;
f=x/(1 + z^2)
F=int(f, z,0,1)
```

有时需要指定变量范围再进行求解：

例：求解不定积分 $\int x^a dx$

```
syms x a
assume(a ~= -1)
f=x^a
F=int(f,x)
```

但是大多情况下 int 都得不到解析解，这时候就可以采用数值积分。

2、quad/quadl/quadgk/quadv

MATLAB 在 R2012a 版本引入了 integral，完全可以替代 quad/quadl/quadv，并且在以后的高版本中，MATLAB 将移除这 3 个函数，所以如果你的 MATLAB 版本高于 R2012a 的话，建议直接使用 integral。

这 4 个函数都是数值积分函数，调用形式完全相同，只是分别适用不同积分函数对象。其中：
quad 采用自适应 simpson 公式数值积分，适用于精度要求低，被积函数平滑性较差的数值积分；
quadl 采用自适应 Lobatto 数值积分，适用于精度要求高，被积函数曲线比较平滑的数值积分；
quadgk采用自适应Gauss-Kronrod数值积分，适用于高精度和震荡数值积分，支持无穷区间，并且能够处理端点包含奇点的情况，同时还支持沿着不连续函数积分，复数域线性路径的围道积分法；
quadv与quad算法相同，是quad的向量化版本，能够一次性计算多个积分。
应当注意，如果要采用数值积分计算一重积分的话，积分函数除了积分变量外，其它的参数都应当具有确定的数值。
调用形式以quad为例：

```
q=quad(fun,a,b)
q=quad(fun,a,b,tol)
```

其中fun为函数句柄，a为积分下限，b为积分上限，tol为积分精度，默认为1e-6

例：计算 $\int_0^2 \frac{1}{x^3-2x-5}dx$

```
y = @(x)1./(x.^3-2*x-5);
q=quad(y,0,2)
```

例：计算 $\int_0^\infty e^{-x^2}dx$

```
y=@(x)exp(-x.^2)
q=quadgk(y,0,inf)
```

对于quadv向量化积分，可以参考winner245版主的帖子：
http://www.ilovematlab.cn/thread-265346-1-1.html

3、integral

integral是2012a引进的一个函数，一元函数积分中功能最为强大，调用形式和quad基本一致：

```
q = integral(fun,xmin,xmax)
q = integral(fun,xmin,xmax,Name,Value)
```

其中fun为函数句柄，xmin为积分下限，xmax为积分上限，Name和Value是一些选项控制，包括误差、向量化积分等等。
integral配合fzero可以求解无法显式表达的函数的定积分。

例：已知 $k=\frac{w^2}{10}\coth(30k)$，求解 $\int_0^{10} kdw$

```
q=@(k,w)w.^2/10.*coth(30*k)-k
v=@(w)fzero(@(k)q(k,w),1e3)    %利用fzero求解k,相当于显式表达k
integral(v,0,10,'ArrayValued',1)
```

论坛关于数值积分的问题：
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=438090
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=275347

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=319403

4、trapz

trapz是基于梯形法则的离散点积分函数。

调用形式：

I=trapz(x,y)

其中x和y分别是自变量和对应函数值，以sin(x)在[0,pi]积分为例：

```
x=linspace(0,pi,1e3);    %生成[0,pi]内的一系列离散点
y=sin(x);
I=trapz(x,y)
```

论坛关于trapz的问题：

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=440457

http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=439162

三、浮点数误差

由于计算机当中都是以二进制形式存储数据，当用十进制数进行计算时，在计算机当中就会存在十进制数和二进制数的相互转换。但是对于某些十进制数转化为二进制时会是一个无限位的小数，这时就必须对该无限位小数进行截断，那么此时就会产生误差，即浮点数误差。

例如十进制的0.9，在转化为二进制时是无限循环小数0.1110011001100110011...。此时必须对该无限位小数进行截断才能保存在内存当中，那么截断后再转换回十进制时，0.9就变成了0.90000000000000002，这就是浮点数误差的产生过程。

由于浮点数误差的存在，当进行数值计算时就会出现一些不可避免的问题，最常见的就是判断两数相等。

例：令a=0.1+0.2，b=0.3，判断a==b时，MATLAB会返回0，当执行a-b时，会发现结果不是精确等于0，而是一个非常小的数5.5511e-17。

或者是在矩阵中寻找数的位置（也相当于是判断两数相等）。

例：

```
>> a=0.1:0.1:0.5

a =

    0.1000    0.2000    0.3000    0.4000    0.5000

>> find(a==0.3)

ans =

   Empty matrix: 1-by-0
```

由于a向量中的0.3是由0.1+0.1+0.1计算得到，那么计算过程中就产生了浮点数误差，这也导致在判断a==0.3时会返回0，所以find(a==0.3)返回一个空矩阵。

那么在进行数值计算判断两数相等时，最好不要直接判断，而是设立一个容差值，当两个浮点数

的差的绝对值小于给定的容差值时，我们就认为这两个浮点数相等。
比如对于上面的例子：

```
>> a=0.1:0.1:0.5

a =

    0.1000    0.2000    0.3000    0.4000    0.5000

>> tol=eps(0.3)*10  %设立容差值，一般比这个点的浮点数误差高一到两个数量级即可。eps函数能够求得该点的浮点数误差值。

tol =

   5.5511e-15

>> find(abs(a-0.3)<tol)

ans =

     3
```

论坛关于浮点数误差的问题：
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=436235
http://www.ilovematlab.cn/thread-333109-1-1.html
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=262078

四、生成一系列有规律名变量，例如 a1、a2、a3……

当循环迭代需要把每次迭代结果进行保存时，如果每次迭代的结果是尺寸不同的矩阵，那么利用eval和num2str这两个函数可以生成一系列例如a1、a2、a3…变量对结果进行保存(不推荐这种方法，原因是eval这个函数有很多缺点)。
eval：将括号内的字符串视为语句并运行。
num2str:将数值转换为字符串。
例：

```
for ii=1:10
    str=['a',num2str(ii),'=1:',num2str(ii)];
    eval(str)
end
```

这样可以生成一系列变量a1、a2…a10将循环结果保存。
不推荐使用eval函数的原因，帮助文档有详细的解释：
http://cn.mathworks.com/help/matlab/matlab_prog/string-evaluation.html

- MATLAB® compiles code the first time you run it to enhance performance for future runs. However, because code in an `eval` statement can change at run time, it is not compiled.
- Code within an `eval` statement can unexpectedly create or assign to a variable already in the current workspace, overwriting existing data.
- Concatenating strings within an `eval` statement is often difficult to read. Other language constructs can simplify the syntax in your code.

MATLAB对于这类问题有更好的解决办法，利用元胞数组对结果进行存储。元胞数组是MATLAB中的特色数据类型，它的元素可以是任意类型的变量，包括不同尺寸或不同维度的矩阵。
对于上面的例子，利用元胞数组：

```
for ii=1:10
    a{ii}=1:ii;
end
```

这样无论是程序的可读性、运行效率还是后面程序对保存结果调用的方便程度，都远胜于eval函数。
除此之外，有时候在处理符号变量时需要生成一系列有规律名符号变量，比如想生成一个多项式：
y=x1+2*x2+3*x3+…+100*x100
eval能够实现，但更简便的方法还是利用矩阵：

```
x=sym('x',[1,100])
w=(1:100).*x
y=sum(w)
```

论坛关于生成系列变量问题：
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=435630
http://www.ilovematlab.cn/thread-272264-1-1.html
http://www.ilovematlab.cn/thread-437236-1-1.html

5、统计向量中连续出现的数字并计数

例如对于一个行向量[0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1],我想要统计其中出现连续1的位置以及连续出现的次数，利用diff函数可以实现这一要求。
diff：求前后两项只差，diff（X）= [X(2)-X(1),X(3)-X(2),...,X(n)-X(n-1)]。
代码如下：

```
A=[0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1];
k=diff([0 A 0]) %对A前后补0之后再作diff，补0是为了保证如果A的第一个和最后一个元素是1的话，也能够通过diff求得1或-1，然后再根据1和-1来寻找连续1的位置和个数
ind=find(k==1)  %1出现的位置即连续1出现的位置
num=find(k==-1)-ind  %-1和1出现的位置差即连续1出现的个数
```

其中ind是出现连续1的首尾的索引，num是该连续1出现的个数。这里ind=[4 8 12 18],num=[3 2 3 1],也就是说A向量中4这个位置开始出现连续1，连续出现3次。同理8位置开始出现连续1，连续出现2次。
很多这种统计向量中数的问题都可以转化为检测连续1出现次数，例如统计向量A中连续数字出现的位置和次数，A=[21,23,25,27,28,29,30,31,33,35,36,38,47,55],那么进行一次转换：k=diff(A)==1，就可以得到k= [0 0 0 1 1 1 1 0 0 1 0 0 0],那么k中连续1出现的位置就是A中连续数字出现的位置，k中连续1出现的次数加1就是A中连续数字出现的次数。

论坛关于统计连续1个数问题：
http://www.ilovematlab.cn/thread-176813-1-1.html
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=440131
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=331012
http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=440520

6、读取文本文件

文本文件的读写函数可以分为两类，一类是高级函数(high-level)，一类是底层函数(low-level)。通常认为高级函数运用起来简单，但是定制性差。而底层函数用法复杂，但是灵活性高。由于MATLAB提供了许多可以读取文本文件的函数，比如load、importdata、textread、dlmread、csvread，要把这些函数各自的适用范围弄清楚也不是一件容易的事。所以我的建议是掌握两个底层函数fscanf和textscan的用法，这样就能够轻松应对一般文本文件的读取了。下面简单介绍几个高级函数的用法，着重介绍两个底层函数fscanf和textscan的用法。

1、csvread

csvread用于读取形式比较简单的文本文件，文件内容只包括数值，并且以逗号或空格为分隔符。

csvread常用用法：

M = csvread(filename)

filename即你需要读取的文件。

例如，创建一个文件，csvlist.dat，包含以下数据：

```
16,2,3,13
5,11,10,8
9,7,6,12
4,14,15,1
```

读取整个文件：

```
filename = 'csvlist.dat';
M = csvread(filename)
M =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

2、dlmread

dlmread的用法比csvread稍微复杂一点，它能够指定分隔符(csvread只能读取逗号分隔符和空格分隔符)。

dlmread常见用法：

M = dlmread(filename)

M = dlmread(filename, delimiter)

其中filename为所读取的文件，delimiter为分隔符。

例：对于包含以下内容的文本文件：

```
16。2。3。13
5。11。10。8
9。7。6。12
4。14。15。1
```

就可以指定'。'为分隔符进行读取：

```
filename = 'csvlist.dat';
M = dlmread(filename,'。')
M =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

如果行列数不一致的数据，dlmread会自动在空白数据处补0。

例：对于包含以下内容的文本文件：

```
40 5 30 1.6 0.2 1.2
15 25 35 0.6 1 1.4
20 45 10 0.8 1.8 0.4

2.6667 0.33333 2
1 1.6667 2.3333
1.3333 3 0.66667
filename = 'csvlist.dat';
M = dlmread(filename)
M =
    40.0000    5.0000   30.0000    1.6000    0.2000    1.2000
    15.0000   25.0000   35.0000    0.6000    1.0000    1.4000
    20.0000   45.0000   10.0000    0.8000    1.8000    0.4000
     2.6667    0.3333    2.0000         0         0         0
     1.0000    1.6667    2.3333         0         0         0
     1.3333    3.0000    0.6667         0         0         0
```

3、fscanf

按指定格式从文本文件中读取数据。

用法：

A = fscanf(fileID,formatSpec)；通过指定读取格式formatSpec从文本文件中读取数据至列向量A。fscanf会重复应用格式字符串formatSpec直到文件指针到达文件末尾，如果读取到不能匹配formatSpec的数据则读取将自动结束。

A = fscanf(fileID,formatSpec,sizeA)；sizeA能够指定读取数据的大小，当读取到sizeA大小的数据时，文件指针会停止，读取结束。注意fscanf读取的是列主序，通常读取完还需要进行转置操作。

所要读取的文本文件被文件标识符fileID标识，通过fopen函数可以获取文件的fileID。当结束读取时，一定要记得使用fclose函数关闭文件。

光看函数的用法介绍可能会比较难懂，通过下面的例子其实会比较容易理解。

例：文本文件 test.txt 包含以下数据：

```
16。2。3。13
5。11。10。8
9。7。6。12
4。14。15。1
```

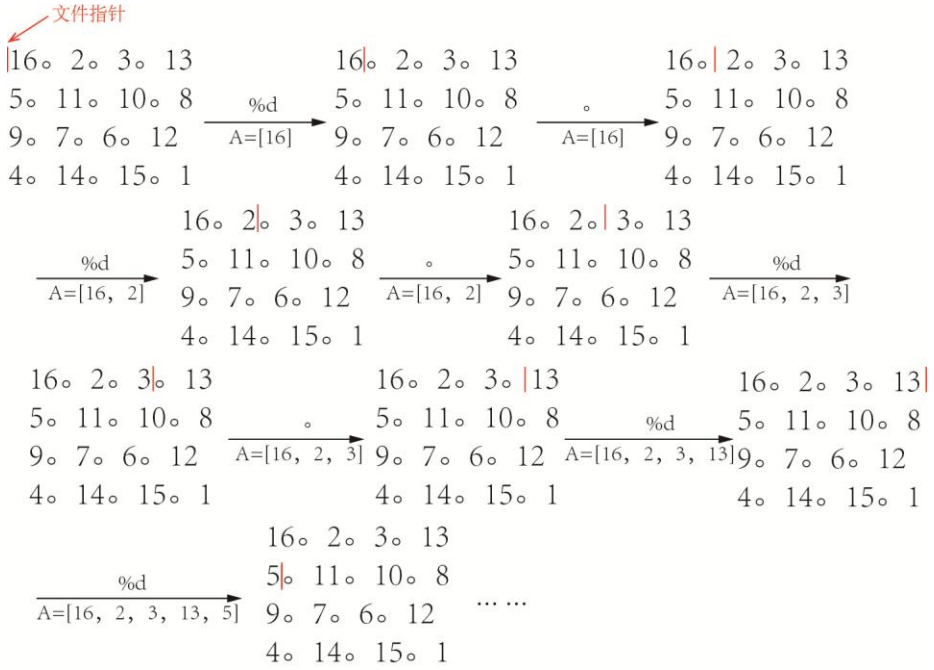
```
fid=fopen('test.txt');  %通过fopen获取文件标识
formatSpec='%d。%d。%d。%d'; %指定读取格式
A=fscanf(fid,formatSpec ,[4,4]); %读取文件数据并存为4*4矩阵
fclose(fid); %调用fclose关闭文件
A=A.' %由于fscanf是列主序，因此读取完还需要进行转置
```

```
A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

下面详细解释一下fscanf的读取原理。

首先当用fopen打开文件时，会有一个文件指针在文件开头。然后fscanf通过你设定的格式字符串formatSpec来读取数据(formatSpec由字符串和转义说明符组成，其中转义说明符由%开头，以转换字母结尾。上面的例子当中%d就是一个转义说明符，代表一个整数，常用的还有%f、%s，分别代表浮点数和字符串)。formatSpec第一个是转义说明符%d，那么fscanf就先将第一个整数16读取给A，之后文件指针跳到16右边，然后formatSpec第2个是字符串'。'，这时候文件指针就会跳过'。'，文件指针到达'。'右边。之后再是转义说明符%d，则将2读入进A，以此类推。用下面图片进行说明：



如果将这个例子的读取代码写成：

```
fid=fopen('test.txt');
A=fscanf(fid,'%d。',[4,4])
fclose(fid);
A=A.'
```

将会得到：

```
A =


    16     2     3    13
```

原因就是因为当文件指针读取完13时，formatSpec需要匹配的数据是'。'，但是13的下一个数据5，这时候就会匹配失败，fscanf自动停止。

再以一个比较复杂的文本文件为例：

例：文本文件 test.txt 包含以下数据：

```
lambda:  7.580000e-05
lambdaB:  8.000000e-05
initial pulse width:  7.853636e-13
output pulse width:  6.253030e-13
dispersion length:  6.307732e-02
nonlinear length:  9.572495e-01

lambda:  7.590000e-05
lambdaB:  8.000000e-05
initial pulse width:  7.848788e-13
output pulse width:  5.778485e-13
dispersion length:  5.852858e-02
nonlinear length:  9.195277e-01
…
```

现在想要把所有的数字信息提取出来：

```
fid=fopen('F:\test.txt');
c1='%*s %e';  %第一行的转义说明符，'%'后面接一个'*'代表跳过这个数据，%*s即代表跳过第一个字符串'lambda:',%e表示读取以科学计数法表示的数字。
c2='%*s %e';
c3='%*s %*s %*s %e';
c4='%*s %*s %*s %e';
c5='%*s %*s %e';
c6='%*s %*s %e';
formatSpec=[c1,c2,c3,c4,c5,c6]; %以6行为一组，重复读取，直至读取完整个文件
A=fscanf(fid,formatSpec,[6,inf])
fclose(fid);
```

4、textscan

textscan的用法与fscanf类似，建议先将fscanf的用法弄清楚再来看textscan。

textscan常见用法：

C = textscan(fileID,formatSpec)

C = textscan(fileID,formatSpec,N)

同fscanf一样，fileID为文件标识符，formatSpec为格式字符串。N则是重复匹配formatSpec的次数。

与fscanf不同的是，textscan将每个与formatSpec转义说明符匹配出来的数据都用一个元胞进行存储。并且textscan有很多选项提供，比如'Headerlines'，可以指定跳过文件的前n行；'Delimiter'可以指定分隔符等等。

例：文本文件 test.txt 包含以下数据：

```
16。2。3。13
5。11。10。8
9。7。6。12
4。14。15。1
```

```
fid=fopen('F:\test.txt');
formatSpec='%d'
A=textscan(fid,formatSpec,'delimiter','。'); %指定'。'为分隔符，如果不指定分隔符的话，就需要把formatSpec写成'%d。%d。%d。%d'  。
fclose(fid);
celldisp(A)

A{1} =

    16
     2
     3
    13
```

5
11
10
8
9
7
6
12
4
14
15
1