MATLAB入门

2020.02

目录

安装与激活

- 进入http://itc.zju.edu.cn/14742/list.htm，按照给出的步骤完成MathWorks账户的注册。
- 按照给出的步骤进行下载和安装。
- *建议下载R2019b版本，并且建议进行在线安装（官网上R2019b似乎默认是在线安装的）.*
- *安装之前关闭所有的安全软件和防火墙，否则可能会出现连接错误.*
- *有时候关闭防火墙和安全软件之后还会出现连接错误，一般来说多试几次就可以了.*
- *在“选择要安装的产品”这一步时，除了选择预先选择的几个推荐产品之外，再额外选择Signal Processing Toolbox, Communications Toolbox 和 DSP System Toolbox.*
- *安装的最后一步是激活，如果提示连接错误的话，可以多激活几次试试。*
- *如果不能在线安装，进入http://itc.zju.edu.cn/matlab/list.htm，下载浙大云盘中R2019b的安装包。*
- *直接下载得到的是ISO文件，Win10系统自带虚拟光驱，直接双击这个文件，将其装载至虚拟光驱；再打开其中的setup.exe，剩余安装过程依然可以直接参照信息中心的给出的步骤进行。*

入门-界面介绍

- 最下方的窗口为Command Windows（如果直接进入Matlab并且不打开任何脚本文件，那么Command Windows会占据中间整个区域），该窗口用于执行脚本。在“>>”后面输入语句，并敲击回车，语句就会即时执行（和C、Pascal等编译执行的语言不同）。
- 右边的窗口是Work Space，在此区域可以查看程序运行中用到的变量。Matlab的变量采用矩阵或元胞矩阵的形式存储，双击对应的变量名就可以查看这个矩阵各元素的值。

入门-基本设定

- 在工具栏中依次单击“主页”-“预设”，弹出预设项窗口。
- 单击“颜色”菜单，可以修改IDE中的语法高亮的颜色风格等;
- 单击“字体”菜单，可以修改各类显示字体的类别和大小等;
- 完成设定后，单击“应用”或“确定”保存修改。
- 在工具栏中依次单击“主页”-“布局”，可以对界面布局进行修改。

基础-脚本文件

- 单击工具栏的“主页”-“新建脚本”，可以建立一个脚本文件;
- 在脚本文件中编写相应的代码，编写完成后利用快捷键Ctrl+S保存该文件。（注意：一般不建议将文件保存到MATLAB的安装目录下）
- 保存脚本文件后，单击工具栏的“编辑器”-“运行”即可运行这个脚本文件。

基础-脚本文件

例子：

新建一个脚本文件，输入如下代码：

clear

clc

A=magic(3)

完成输入后保存脚本文件并运行（可能会弹出“更改文件夹”的对话框，一般情况下单击“更改文件夹”按钮），Command Space将会自动输入脚本文件名并执行，执行结果显示为一个三阶幻方矩阵（三阶幻方是横竖斜各方向数字之和均相等的矩阵）。

下面尝试不使用“运行”按钮执行程序（这在执行用户自定义函数的时候非常有用）。将保存的文件名直接输入到Command Windows（注意文件名大小写，Matlab是大小写敏感的），并敲击回车，程序运行得到相同的结果。

基础-基本运算符

- MATLAB的大多数运算是被视为矩阵运算进行的，常用的运算符见下表：

符号	功能
+	加法
-	减法
.*	按元素乘法
*	矩阵乘法
./	按元素右除
/	矩阵右除
.\	按元素左除
\	矩阵左除 （也称为*反斜杠*）
.^	按元素求幂
^	矩阵幂
.'	转置
'	复共轭转置

符号	功能
==	等于
~=	不等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于

符号	功能
&	逻辑 AND
|	逻辑 OR
&&	逻辑 AND（具有短路功能）
||	逻辑 OR（具有短路功能）
~	逻辑非

基础-基本运算符

- 矩阵除了像线性代数中那样直接用矩阵运算进行处理，也可以像在其它高级语言中那样被当做数组操作，这时就需要用到矩阵索引。

例子:

```
>> A=magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
>> A(2:4,:)
ans =
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
>> A(3,:)
ans =
     4     6    13    20    22
>> A(3,2)
ans =
     6
```

```
>> A(end,:)
ans =
    11    18    25     2     9
>> A(end-1,3:5)
ans =
    19    21     3
>> A(end-3:end,3:5)
ans =
     7    14    16
    13    20    22
    19    21     3
    25     2     9
```

基础-基本运算符

例子（续）：

- 对这个例子的各个语句解释如下：
- Magic(5) 产生了一个5阶幻方矩阵；
- A(2:4,:) 将矩阵第2到第4行的所有元素取出，其中冒号前后均缺省数字表示与矩阵原有维度相同，在这里即等价于A(2:4,1:5);
- A(3,:) 将矩阵第3行所有元素取出；
- A(3,2) 将矩阵第3行第2列的元素取出；
- A(end,:) 将矩阵最后一行的所有元素取出，此处等价于A(5,:);
- A(end-1,3:5) 将矩阵倒数第2行的第3到5列的元素取出；
- A(end-3:end,3:5) 将矩阵倒数第4行到倒数第一行第3到5列的元素取出。

基础-基本函数

- MATLAB 提供了大量标准初等数学函数，包括 abs、sqrt、exp 和 sin。生成负数的平方根或对数不会导致错误；系统会自动生成相应的复数结果。MATLAB 还提供了许多其他高等数学函数，包括贝塞尔函数和 gamma 函数。其中的大多数函数都接受复数参数。有关初等数学函数的列表，请键入
- *help elfun*
- 有关更多高等数学函数和矩阵函数的列表，请键入
- *help specfun*
- *help elmat*

(基本函数、基本常数和表达式的例子的内容均来源于MATLAB官网，详细内容参阅 https://ww2.mathworks.cn/help/matlab/learn_matlab/expressions.html)

基础-基本常量

- 科学计算中通常需要用到很多基本常数，当然也可以使用手工定义的方法来使用这些常数，但是使用MATLAB内置的常数通常更加方便和精确，以下是MATLAB的常用数学常数表。

pi	3.14159265...
i	虚数单位
j	与 i 相同
eps	浮点相对精度，实际应用中基本可以视为无穷小量（真正的无穷小量应该是realmin）
Inf	无穷大

基础-表达式

例子：

```
rho = (1+sqrt(5))/2
rho =
    1.6180

a = abs(3+4i)
a =
     5

z = sqrt(besselk(4/3,rho-i))
z =
   0.3730+ 0.3214i

huge = exp(log(realmax))
huge =
  1.7977e+308

toobig = pi*huge
toobig =
   Inf
```

基础-顺序结构、选择结构、循环结构

- 和其它高级语言类似，MATLAB也是由三种基本的结构组成的。
- 顺序结构：在没有选择结构和循环结构的情况下，语句执行默认是顺序的；
- 选择结构：使用if…elseif…else…实现选择结构，使用switch…case…otherwise…实现多分支结构；
- 循环结构：使用for实现规定次数的循环，使用while实现不确定次数的循环。
- 以上这些结构的详细用法在MATLAB自带的帮助文件中有非常详细的说明。在Command Windows中输入”help xxx”即可获得这些信息（xxx是需要查询的函数、关键字以及运算符）。

基础-例子：产生质数

例子：

```
MaxNum=100;
MinNum=2;
res=[];
for i=MinNum:1:MaxNum
    f=0;
    % f为0表示不能找到因数
    for j=2:1:fix(sqrt(i))
        if mod(i,j)==0
            f=1;
            break;
        end
    end
    if f==0
        res(end+1)=i;
    end
end
res
```

进阶-提升程序运行效率

- MATLAB计算循环的效率非常低下，然而其进行矩阵运算的效率很高，因此为了提高程序的运行效率，应当尽量使用矩阵运算代替循环。
- 利用线性代数和离散数学的知识，一些需要使用循环的操作可以被转化为矩阵运算，下面给出这种等效的一些例子。

例子：

```
>> a=[1,2,3,4,5,6,7];
>> b=ones(1,7);
>> c=a*b'

c =

    28
```

进阶-提升程序运行效率

例子（续）：

```
>> a=[1,2,3,4,5,6,7];
>> b=0;
>> c=0;
>> for b=1:7
   c=c+a(b);
end
>> c
c =

   28
```

- 两段代码都完成了对7维矢量a的各维度求和，但是第一种方法使用矩阵运算进行，第二种方法使用循环进行。在矢量a的维数较小时，这两种方法的效率相差不大；随着维数的增加，第一种方法计算消耗的时间更不明显。
- 困难的是，大多数循环结构的等效矩阵运算表达形式并不是显而易见的。

进阶-提升程序运行效率

- 在多数情况下，MATLAB提供的内置函数的执行速度都会比用户自行编写的函数的执行速度更快。在刚才的例子中，我们用较快的矩阵运算替代了较慢的循环累加求和；事实上，MATLAB提供了求和函数sum计算矩阵的和，因此我们也可以使用sum函数替代先前的矩阵运算。
- 总体而言，在MATLAB提供了内置函数的情况下，调用内置函数大概率可以同时降低编程复杂度和时间复杂度；当没有合适的内置函数可供调用时，尽量考虑使用矩阵运算实现需要的功能；当无法找到合适的矩阵运算时，则使用循环、递归等相对低效的方法（但循环和递归也不总是低效的，特别是在分治算法中）。

进阶-分节运行

- 现在考虑这样一个问题：有一段比较长的代码，这段代码从前往后分为A、B、C、D四个部分。为了实现功能1，需要A部分和B部分；为了实现功能2，需要A部分和C部分；为了实现功能3，需要A、C、D这3个部分。为了实现功能4， A、B、C、D四个部分需要被依次运行。怎样可以较为高效的解决这个问题？
- 首先，将A、B、C、D分别保存为4个脚本大概不是个好办法，因为这样做的话，每次运行时都需要切换脚本文件。
- 其次，利用自定义的子程序或者函数可能会使得代码更简洁些，但是需要考虑参数传递等问题，并且需要对程序进行较复杂的改写。
- 是否存在一个办法，可以允许我们每次运行部分或全部脚本？

进阶-分节运行

- 分节运行或许是解决这个问题的一个比较好的办法。在MATLAB中，独占一行的连续的两个百分号%%表示一个分节符。在分节符所在行的上方将出现一条细线，细线将程序分成2节或多节。
- 当光标处于某一节时，单击“编辑器”-“运行节”或者利用快捷键Ctrl+Enter可以运行当前光标所在的节，这就意味着程序将会从本节开始的行运行至下一个分节符出现的位置或是文件末尾。
- 回到刚才的问题，我们只需要将A、B、C、D四个部分用分节符分成4节，再根据需要运行对应的节就可以了。

进阶-分节运行

- 除此以外，分节对提高程序可读性和代码调试都很有帮助。
- 将比较长的程序中相对独立的几个部分分成几节，不仅使程序各段的功能清晰，同时也便于他人阅读。按功能块阅读代码段不仅可以加快阅读代码的速度，同时对理解代码也很有帮助。
- 在代码调试过程中，错误的定位一般来说会消耗大量的时间。如果将代码按照功能分为几节，就可以依次运行每一节并检查结果，当某节的结果与预期结果不符时，就可以将错误定位在这一节。

补充-快速傅里叶变换

- 变换和滤波器是用于处理和分析离散数据的工具，常用在信号处理应用和计算数学中。当数据表示为时间或空间的函数时，傅里叶变换会将数据分解为频率分量。fft 函数使用快速傅里叶变换算法，相对于其他直接实现，这种方式能够减少计算成本。
- 语法：

Y=fft(X)

- 说明：

Y = fft(X) 用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)。

如果 X 是向量，则 fft(X) 返回该向量的傅里叶变换。

如果 X 是矩阵，则 fft(X) 将 X 的各列视为向量，并返回每列的傅里叶变换。

如果 X 是一个多维数组，则 fft(X) 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回每个向量的傅里叶变换。

- MATLAB官网有对傅里叶变换和fft使用方法的详细论述：https://ww2.mathworks.cn/help/matlab/math/fourier-transforms.html

额外的参考资料

- 如前所述，MATLAB内有详尽的帮助文件，调用方法为

help xxx

- MATLAB官网上对内置函数、运算符、语法乃至信号处理、图论和微分方程求解等许多问题均有大量的介绍：

https://ww2.mathworks.cn/help/