MATLAB 快速入门

周吕文

2014 年 11 月 20 日

引言

MATLAB 是 Matrix Laboratory “矩阵实验室” 的缩写. MATLAB 语言是由美国的 Clever Moler 博士于 1980 年开发的, 初衷是为解决 “线性代数” 课程的矩阵运算问题. 1984 年由美国 MathWorks 公司推向市场, 历经十多年的发展与竞争, 现已成为国际公认的最优秀的工程应用开发环境. MATLAB 功能强大, 简单易学, 编程效率高, 深受广大科技工作者的欢迎.

在数学建模竞赛中, 由于只有短短的三到四天, 而论文的评判不仅注重计算的结果更注重模型的创造性等很多方面, 因此比赛中把大量的时间花费在编写和调试程序上只会喧宾夺主, 是很不值得的. 使用 MATLAB 可以很大程度上的方便计算, 节省时间, 使我们将精力更多的放在模型的完善上, 所以是较为理想的.

这里快速的介绍一下 MATLAB 与数学建模相关的基础知识, 并列举一些简单的例子, 很多例子都是源于国内外的数学建模赛题. 希望能帮助同学们在短时间内方便, 快捷的使用 MATLAB 解决数学建模中的问题. 当然要想学好 MATLAB 更多的依赖自主学习, 一个很好的学习 MATLAB 的方法是查看 MATLAB 的帮助文档:

- 如果你知道一个函数名，想了解它的用法，你可以用 ‘help’ 命令得到它的帮助文档:

  `>> help` 函数名

- 如果你了解含某个关键词的函数，你可以用 ‘lookfor’ 命令得到相关的函数:

  `>> lookfor` 函数名

例如 `help sum` 命令将输出 `sum` 函数的帮助信息. 其它一此可能有用的帮助命令 ‘`info`’, ‘`what`’ 和 ‘`which`’ 等. 这些命令的详细用法和作用都可以用`help`获得. MatLab 中还提供了很多程序演示实例, 这些例子可以通过 ‘`demo`’ 获得.

变量

MATLAB 程序的基本数据单元是数组, 一个数组是以行和列组织起来的数据集合, 并且拥有一个数组名. 标量在 MATLAB 中也被当作数组来处理, 它被看作只有一行一列的数组. 数组可以定义为向量或矩阵. 向量一般来描述一维数组, 而矩阵往往来描述二维或多维数组. 数组中的任何元素都可以是实数或者复数; 在 MATLAB 中, $\sqrt{-1}$ 是由 ‘i’ 或 ‘j’ 来表示, 当然前提是用户没有预先重新定义 ‘`i`’ 或 ‘`j`’. MATLAB 中, 数组的定义要用 ‘`[ ]`’ 来括起来, 数组中同一行元素间以空格或逗号 ‘`,`’ 格开, 行与行之间由分号 ‘`;`’ 格开. 下面给出实数, 复数, 行向量, 列向量和矩阵的定义及赋值方式.

实数	`>> x = 5`		
复数	`>> x = 5 + 10i`	或者	`>> 5 + 10j`
行向量	`>> x = [1 2 3]`	或者	`>> x = [1, 2, 3]`
列向量	`>> x = [1; 2; 3]`		
3×3 矩阵	`>> x = [1 2 3; 4 5 6; 7 8 9]`		

需要注意的是, 一个数组每一行元素的个数必须完全相同, 每一列元素的个数也必须完全相同. 对于复数的输入, 虚部前的系数和 'i' 或'j' 之间不能有空格, 如`-1 + 2 i`是不对的, 而`-1 + 2i`或`-1 + i*2` 才是有效的输入方式.

固定变量

前面我们说如果用户没有预先重新定义 'i' 或 'j', 则 'i' 或 'j' 表示 $\sqrt{-1}$. 在 MATLAB 中还有几个常见的固定变量, 如果用户没有预先重新定义, 这些固定的变量有着自身的意义.

`pi`	π
`i, j`	$\sqrt{-1}$
`inf`	∞
`NaN`	非数
`ans`	默认变量

其中`NaN`是 not a number 的缩写, 像`0/0`, `inf*inf`等情况的计算会产生非数; `ans`是 answer 的缩写, 如果不定义变量, MATLAB 会将运算结果放在默认变量 ans 中.

复数运算

MATLAB 中提供了一此复数运算的函数, 这里列出一些重要的复数运算函数.

```
复数输入   >> x = 3 + 4j
实部       >> real(x)     ⟹ 3
虚部       >> image(x)    ⟹ 4
模长       >> abs(x)      ⟹ 5
共轭复数   >> conj(x)     ⟹ 3 - 4i
幅角       >> angle(x)    ⟹ 0.9273
```

向量, 矩阵的快捷生成

创建一个小数组用一一列举出元素的方法是比较容易的, 但是当创建包括成千上万个元素的数组时则不太现实. 在 MatLab 中, 向量可以通过冒号 ':' 方便快捷地生成, 用两个冒号安顺序隔开 '第一个值','步增' 和 '最后一个值' 就可生成指定的向量. 如果步增为 1, 则可以省略掉步增和一个冒号, 比如

```
>> x = 1:0.5:3   ⟹ x = [1.0 1.5 2.0 2.5 3.0]
>> y = 1:3       ⟹ x = [1 2 3]
```

向量的快捷生成还可以调用函数`linspace`, `linspace`函数只需给出向量的第一个值, 最后一个值和等分的个数就可以生成指定的向量. 转置运算符 (单引号)`'` 可以用来将行向量转置为列向量, 或更加复杂的矩阵的转置.

```
>> x = [1:2:5]'   ⟹ x = [1; 3; 5]
```

对于特殊的向量和矩阵, MATLAB 提供了一些内置函数来创建它们. 例如, 函数 `zeros` 可以初始化任何大小的全为零的数组. 如果这个函数的参数只是一个标量, 将会创建一个方阵, 行数和列数均为这个参数. 如果这个函数有两个标量参数, 那么第一个参数代表行数, 第二个参数代表列数.

```
>> x = zeros(2)     ⟹ x = [0 0; 0 0]
>> x = zeros(2,3)   ⟹ x = [0 0 0; 0 0 0]
```

类似的内置函数还有`ones`, `eye`, 用法与`zeros`一致. 函数`ones`产生的数组包含的元素全为 1; 函

数ones用来产生单位矩阵, 只有主对角线的元素为 1.

获取向量/矩阵中的元素

通过定元素所在的指行和列, 可以获得矩阵中指定的一个或多个元素. 比如可以用以下语句获得矩阵 `A = [1 2 3; 4 5 6; 7 8 9]` 的第 1 行的第 3 列的元素

```
>> x = A(1,3)   ⟹ x = 3
```

可以用以下语句获得矩阵A的第 2 行所有元素

```
>> y = A(2,:)   ⟹ x = [4 5 6]
```

其中的冒号 ':' 表示 "所有列" 的意思. A 矩阵的前两行前两列组成的矩阵可通过以下语句获得

```
>> z = A(1:2,1:2)   ⟹ z = [1 2; 4 5]
```

矩阵和数组运算

针对矩阵和数组, MATLAB 有一系列的算术运算, 关系运算和逻辑运算.

算术运算

矩阵的基本数学算数主要有加法, 减法, 乘法, 右除, 左除, 指数和转置运算, 运算符号如下:

`+`	加法运算
`-`	减法运算
`*`	乘法运算
`/`	右除运算
`\`	左除运算
`^`	指数运算
`'`	转置运算

这些运算的法则都与线性代数中的相同. 如果矩阵 A 与矩阵 B 相乘, 则必需满足 A 的列数必需等于 B 的行数, 否则 MATLAB 会报错. 对于左除和右除运算, `x = A\b` 是 $Ax = b$ 的解, 而`x=b/A`是 $xA = b$ 的解.

上面介绍的矩阵运算中的乘除运算都不是针对同阶数组对应分量的运算, 而 MATLAB 中还有针对同阶数组对应分量的运算, 这种称为数组的运算, 也称点运算. 点运算包括点乘, 点除和点乘方.

`.*`	乘法运算
`./`	右除运算
`.\`	左除运算
`.^`	指数运算

以下给出一个例子来说明矩阵运算和数组运算的区别:

```
>> A = [1 2; 3 4]
A =
     1     2
     3     4
>> B = A*A
B =
     7    10
    15    22
>> C = A.*A
C =
     1     4
     9    16
```

关系运算

关系运算是用来判断两同阶数组 (或者一个是矩阵, 另一个是标量) 对应分量的间的大小关系. 关系运算包括以下操作:

```
<    小于
<=   小于等于
>    大于
>=   大于等于
==   等于
~=   不等于
```

若参与运算的是两个矩阵, 关系运算是将两个矩阵对应元素逐一进行关系运算, 关系运算的结果是一个同维数逻辑矩阵, 其元素值为只含 0(假) 和 1(真). 若参与运算的一个是矩阵, 另一个是标量, 则是矩阵中每个元素与该标量进行关系运算, 最终产生一个同维数逻辑矩阵, 其元素值为只含 0(假) 和 1(真). 例如

```
>> A = [1 3 4 2 5]
>> B = [2 1 3 5 4]
>> C = A>B   % C = [1> 2, 3 >1, 4 >3, 2 >5, 5 >4]
C =
     0     1     1     0     1
>> D = A<=3  % D = [1<=3, 3<=3, 4<=3, 2<=3, 5<=3]
D =
     1     1     0     1     0
```

需要注意的是=和==差别: 前者是赋值运算; 后者是关系运算符 “等于”, 判断是否相等.

逻辑运算

MATLAB 的基本逻辑运算符为: &(与), |(或), ~(非). 参与逻辑运算的是两个同维数矩阵; 或者一个是矩阵, 另一个是标量. 若参与运算的是两个矩阵, 逻辑运算是将两个矩阵对应元素逐一进行逻辑运算, 逻辑运算的结果是一个同维数逻辑矩阵, 其元素值为只含 0(假) 和 1(真). 若参与运算的一个是矩阵, 另一个是标量, 则是矩阵中每个元素与该标量进行逻辑运算, 最终产生一个同维数逻辑矩阵, 其元素值为只含 0(假) 和 1(真). 例如

```
>> A = [1 3 4 2 5];
>> B = [2 1 3 5 4];
>> C = (A>B) & (A<=3)
C =
     0     1     0     0     0
```

逻辑运算常与 MATLAB 中的程序控制结构语句 (如if, while等) 相结合.

常用数学函数

MATLAB 中包括了大量的内置函数, 这些函数的组合可以很方便的完成很多复杂的功能. 下面给出常用的数学函数:

sin	正弦	asin	反正弦
cos	余弦	acos	反余弦
tan	正切	atan	反正切
cot	余切	acot	反余切
exp	指数函数	sqrt	平方根
log	自然对数	log10	以 10 为底的对数
abs	绝对数	sign	符号函数
min	取小	max	取大
sum	求和		

以上这些函数大多 (除min, max和sum) 也是针对矩阵对应元素逐一进行函数的运算, 比如

```
>> theta = 0:pi/3:pi
theta =
         0    1.0472    2.0944    3.1416
>> sin(theta)
ans =
         0    0.8660    0.8660    0.0000
```

控制结构语句

在 MATLAB 中, 常用的控制结构语句主要是if结构和for结构. if语句有 3 种格式:

- 单分支if语句, 其格式如下. 当条件成立时, 则执行语句组, 执行完之后继续执行 if 语句的后继语句, 若条件不成立, 则直接执行 if 语句的后继语句.

```
if 条件
    语句组
end
```

- 双分支if语句, 其格式如下. 当条件成立时, 执行语句组 1, 否则执行语句组 2, 语句组 1 或语句组 2 执行后, 再执行 if 语句的后继语句.

```
if 条件
    语句组1
else
    语句组2
end
```

- 多分支if语句. 其格式如下. 多分支if语句用于实现多分支选择结构, 可以代替switch语句.

```
if 条件1
    语句组1
elseif 条件2
    语句组2
      .....
elseif 条件n
    语句组n
else
    语句组n+1
end
```

循环结构for语句的格式如下:

```
for 循环变量 = 表达式1:表达式2:表达式3
    循环体语句
end
```

其中表达式 1 的值为循环变量的初值, 表达式 2 的值为步长, 表达式 3 的值为循环变量的终值. 步长为 1 时, 表达式 2 可以省略. “表达式 1: 表达式 2: 表达式 3” 还可用一向量或矩阵替代, 执行过程是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. 下面给出一个例子来说明if和for语句的用法:

```
% 计算10以内的奇数和
tot = 0;
for i = 1:10
   if mod(i,2); tot = tot + i; end
end
```

在 MATLAB 编程中, 采用循环语句会降低其执行速度, 因此如果可以用内置函数或数组或矩阵的运算实现的功能, 尽量不用循环语句, 例如以上程序可以由以下语句实现

```
tot = sum(1:2:10);
```

其它控制结构语句还有while, switch, continue, break, return等, 请读者自己了解这些控制结构语句.

MATLAB 文件

为了保存程序, 数据, 图像等. MATLAB 有一系列的文件来存储程序, 数据, 图像等. 本节主要介绍用于保存 MATLAB 程序脚本的 m 文件和 m 函数文件以及 fig 图形文件.

M 文件和 M 函数文件

当用户要运行的指令较多时, 直接从键盘上逐行输入指令比较麻烦, 而命令文件可以较好地解决这一问题. 用户可以将一组相关指令编辑在同一个文件中, 即从指令窗口工具栏的新建按钮或选择菜单 File:New:M-File 进入 MATLAB 的程序编辑器窗口, 以编写自己的 M 文件, 运行时 M 文件时, 只需点击工具栏中的运行按键或在命令窗口中输入文件名字, MATLAB 就会自动按顺序执行文件中的命令. 如图1所示, 一个文件名为 main.m 的 M 文件. 需要注意的是:

- M 文件名只能是字母, 下划线和数字的组合, 且只能以字母开头. M 文件名也不要与 MATLAB 内置函数名相同.
- 在 MATLAB 中, “%” 为注释符, %后的语句用于注释, 不会被执行.
- 以 “;” 结尾的 MATLAB 命令语句, 将不会在命令窗口中输出该语句的运行结果.

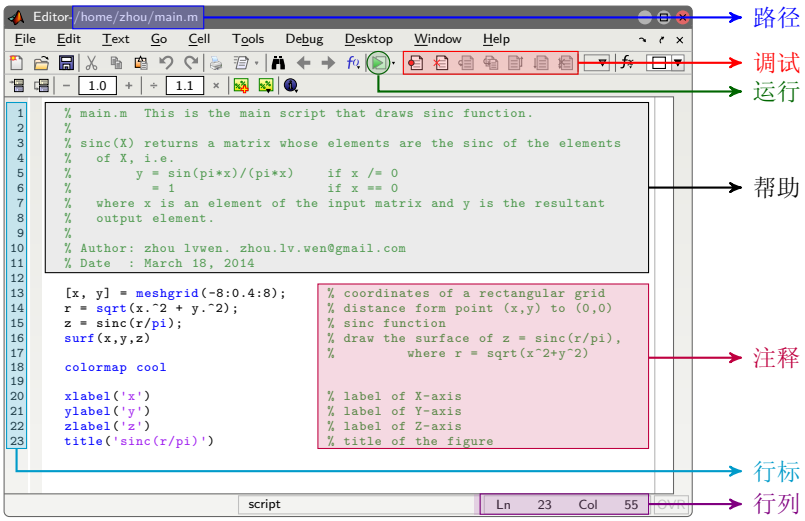


图 1: MATLAB 程序编辑器窗口

M 文件中有一种特殊的文件, 这种文件除注释外以function开头, 我们称这样的文件为 M 函数文件. M函数文件用得更为广泛, 它具有参数传递功能, 可以很方便地在命令窗口或其它脚本文件中进行调用. M 函数文件的文件格式如下

```
function [输出1, 输出2, ...] = 函数名(输入1, 输入2,...)
% 函数的一些说明

MATLAB 语句1;
......
MATLAB 语句n;
```

需要注意的是: 函数文件的变量是局部变量, 运行期间有效, 运行完毕就自动被清除; M 函数文件名必需与函数名一致, 即 M 函数文件必需保存成 “函数名.m”. 下面给出一个 M 函数的例子, 该 M 函数用来把角度转化为孤度的函数, M 函数名为 “ang2rad.m”.

```
function rad = ang2rad(ang)
% ANG2RAD: 这是一个把角度转化为孤度的函数
rad = ang/180*pi;
```

有了这个函数, 我们就可以在需要把角度转化为孤度的地方直接调用这个 M 函数, 比如

```
>> theta = 0:60:180
theta =
     0    60   120   180
>> ang2rad(theta)
ans =
         0    1.0472    2.0944    3.1416
```

FIG 文件

MATLAB 最强大的功能之一是作图, 用 MATLAB 作图后, 可以把图存成 FIG 文件. 图2是 MATLAB 作图后显示出来的图形窗口. 通过 File:Save 可以把图存成 “图名.fig” 的文件. FIG 文件保存了图形的所有信息, 可以方便以后的编辑, 以及转存为其它格式的图片. 如果有需要, 我们也可以从 FIG 文件中获得图形的相关数据.

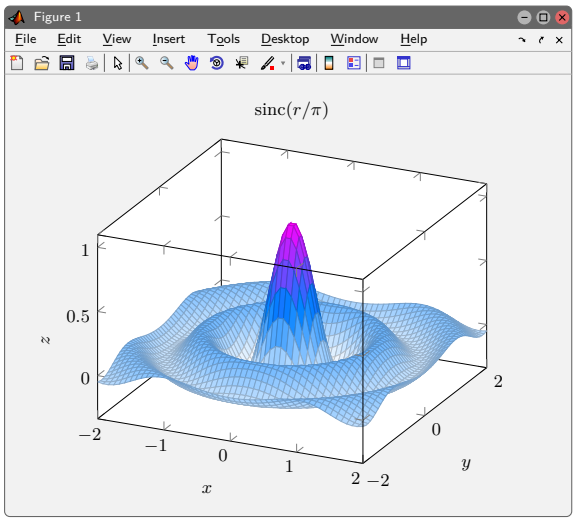


图 2: MATLAB 程序图形窗口

作图

MATLAB 最强大的功能之一是作图, MATLAB 中包括了大量的二维和三维的作图函数. 本节我们简单的介绍一下 MATLAB 简单的二维作图命令及三维作图命令.

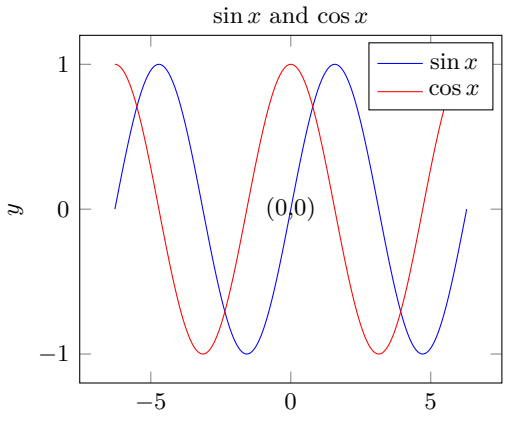
二维作图命令

MATLAB 中最常用也是最基本二维作图命令为`plot`, `plot`函数针对向量或矩阵的列来绘制曲线的. 调用`plot`函数的常用格式有以下几种

- `plot(x)`: 当`x`为一向量时, 以`x`元素的值为纵坐标, `x`的序号为横坐标值绘制曲线. 当`x`为一实矩阵时, 则以其序号为横坐标, 按列绘制每列元素值相对于其序号的曲线, 当`x`为 $m \times n$ 矩阵时, 就有 n 条曲线.
- `plot(x,y)`: 以`x`元素为横坐标值, `y`元素为纵坐标值绘制曲线.
- `plot(x,y1,x,y2,...)`: 以公共的`x`元素为横坐标值, 以`y1, y2...` 元素为纵坐标值绘制多条曲线.

值得注意的是, 在上面的每种用法中, 还可额外指定颜色, 点型或线型来区分不同的数据组. 下面给出一个简单的实例:

```
% 画sin和cos曲线
x = -2*pi:0.1:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x, y1, '-b');
hold on
plot(x, y2, '-r');
xlabel('x')
ylabel('y')
text(0,0, '(0,0)')
legend('sin(x)', 'cos(x)')
```



以上程序中`plot(x,y1,'-b')`画出了一箱 $y = \sin x$ 的曲线, 其中参数`'-b'`指定了曲线为蓝色实

线. 更多参数选项见表1. 在上面的程序中还用到了一些常要命令, 下面更多常用的命令.

xlabel	x 坐标轴标签
ylabel	y 坐标轴标签
title	在图上方加标题
grid on/grid off	开启/关闭网格
text	在指定位置标文本
axis	控制 x 坐标和 y 坐标的范围
hold on/hold off	开启 / 关闭保持当前图形, 允许多个图形叠加于同一坐标系.

表 1: 绘图参数表

颜色	点型		线点
b 蓝	. 点	^ 向上三角形	- 实线
g 绿	o 圆	< 向左三角形	: 点线
r 红	x 叉号	> 向右三角形	-. 点划线
c 青	+ 加号	p 五角星	-- 虚线
m 紫	* 星号	h 六角星	
y 黄	s 正方形		
k 黑	d 菱形		
w 白	v 向下三角形		

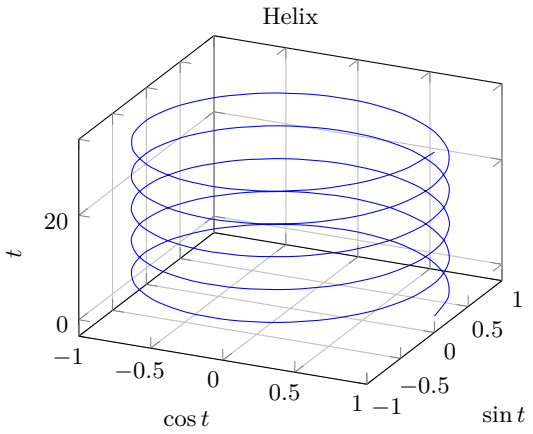
除了plot外, 还有一些常用的二维作图命令, 下面给出这些命令的列表, 请读者自行查找帮助文档来了解它们的用法.

loglog	使用对数坐标系绘图
semilogx	横坐标为对数坐标轴
semilogy	横坐标为线性坐标轴
polar	极坐标图
bar	直方图
errorbar	误差棒图
pie	制饼图
hist	统计直方图

三维作图命令

与二维曲线作图函数plot相对应, MATLAB 提供了plot3函数, 可以在三维空间中绘制三维曲线，它的格式类似于 plot, 不过多了 z 方向的数据. 这里不再祥述其调用格式, 给出一个实例供读者学习.

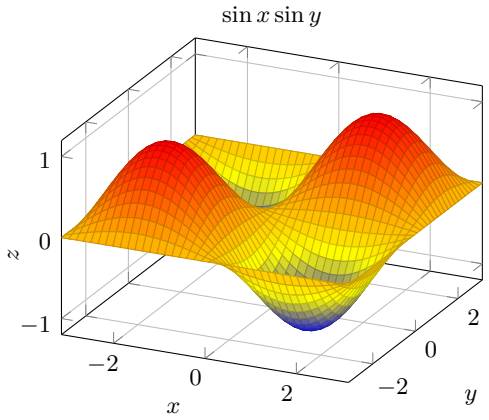
```
% 画螺旋线
t=0:pi/50:10*pi;
x = sin(t);
y = cos(t);
z = t;
plot3(x,y,z)
title('Helix')
xlabel('sint(t)')
ylabel('cos(t)')
zlabel('t')
grid on
```



除了三维空间曲线, 三维作图中还有三维曲面作图. 用于绘制三维曲面的函数主要有mesh和surf. 在用mesh和surf绘制三维曲面时, 常伴随着函数meshgrid的使用, meshgrid是

MATLAB 中用于生成网格采样点的函数, 生成绘制 3-D 图形所需的网格数据. 下面给出surf绘制曲面的实例.

```
% 画曲面 z = sin(x)*cos(y)
[x,y] = meshgrid(-pi:0.1:pi);
z = sin(x).*cos(y);
surf(x,y,z)
xlabel('x')
ylabel('y')
zlabel('z')
title('sin x sin y')
```



以上程序, 先用 meshgrid 函数产生在 x-y 平面上的二维的网格数据，再以一组 z 轴的数据对应到这个二维的网格, 即可用surf(x,y,z)画出三维的曲面.