Matlab复习笔记(个人向)

学Matlab无非就一个讲究，老老实实把系统的 `help` 帮助中心部分给看了，非常的管用

1. 变量

- 申明

  ```
  变量=表达式;
  ```

  - 写了；则不会显示在命令行窗口中，不写则会显示
- 查询
  - `who` 语句查看工作空间中的所有变量
  - `whos` 查看工作空间中变量的详细信息
- 预置变量名
  - `pi` :圆周率数Π
  - `inf,Inf` :无穷大
  - `nan,NaN` :一个不定值，代表置空
  - `ans` :数值若无指定变量接着，系统会把值赋给特殊变量 `ans` .

> **应避免给系统预定义的变量重新赋值，可以用 `iskeyword` 命令查看**

- 可以取连续的范围赋值,其格式为

  变量名=范围边界1:步长:范围边界2

  其中步长可以省略，**默认为1**

```
>> x=1:3:15

x =

     1     4     7    10    13
```

- 变量的存储
  `save 文件名 变量名列表`
  或者
  `save(文件名，变量名1，变量名2，……)`
- 从文件中提取变量
  `load 文件名 变量名`
- 清除当前工作空间中的变量
  `clear 文件名 变量名` %清除指定的变量
  `clear all` %清除所有的变量
  `clc` 在命令行中输出，清屏
- 变量的使用格式
  在文件/脚本一开始可以用 `format ElemType` 框定好文件要使用的格式
- **Matlab中的文件格式是 `.mat` ,脚本(代码)格式是 `.m`**

2.运算

- `+,-,*` :加减乘
- `/` :左除(左边除右边)
  `\` :右除(右边除左边)
- `^` :幂运算
- **矩阵四则运算**
  - `.*` :点乘，即矩阵各元素相乘
  - `./` :点除，即矩阵各元素相除
  - `.^` :点幂，即矩阵各元素相幂

```
x=[1,2,3];
y=[4,5,6];
x.*y=[1*4,2*5,3*6];
y./x=[4/1,5/2,6/3];
x.\^y=[1\^4,2\^5,3\^6];
```

- **Matlab以双精度double执行所有运算，运算结果可以在屏幕上输出并赋给变量**
- **Matlab对矩阵的计算操作都是以 列 为单位的**
- 数的输出格式可以通过format命令指定，它只会改变变量的输出格式，但不会影响变量的值。
- 在输入命令行直接运算时，可以按 tab 补全信息，按 Esc 删除正在编辑的这行命令行

3.矩阵

- **在Matlab中，矩阵的运算速度非常快，可以短时间内处理大量的数据。**
- 用 [] 框住矩阵元素，一行输入为标准，；来分隔每一行的信息，同行的元素可以用空格或者逗号，分隔
  如以下例子

```
mat1=[1,2,3;5,6,7;9,6,1]
```

```
mat1 =

     1     2     3
     5     6     7
     9     6     1
```

- 矩阵赋值有两种形式，除了直接赋值还有对单个元素修改的赋值方法，**也可以把小矩阵当成其元素进行赋值**

```
A=[1,2;5,6];  %直接赋值
A(1,1)=0;    %对单个元素的修改
A=[A;7,8];    %把矩阵当成元素进行赋值
```

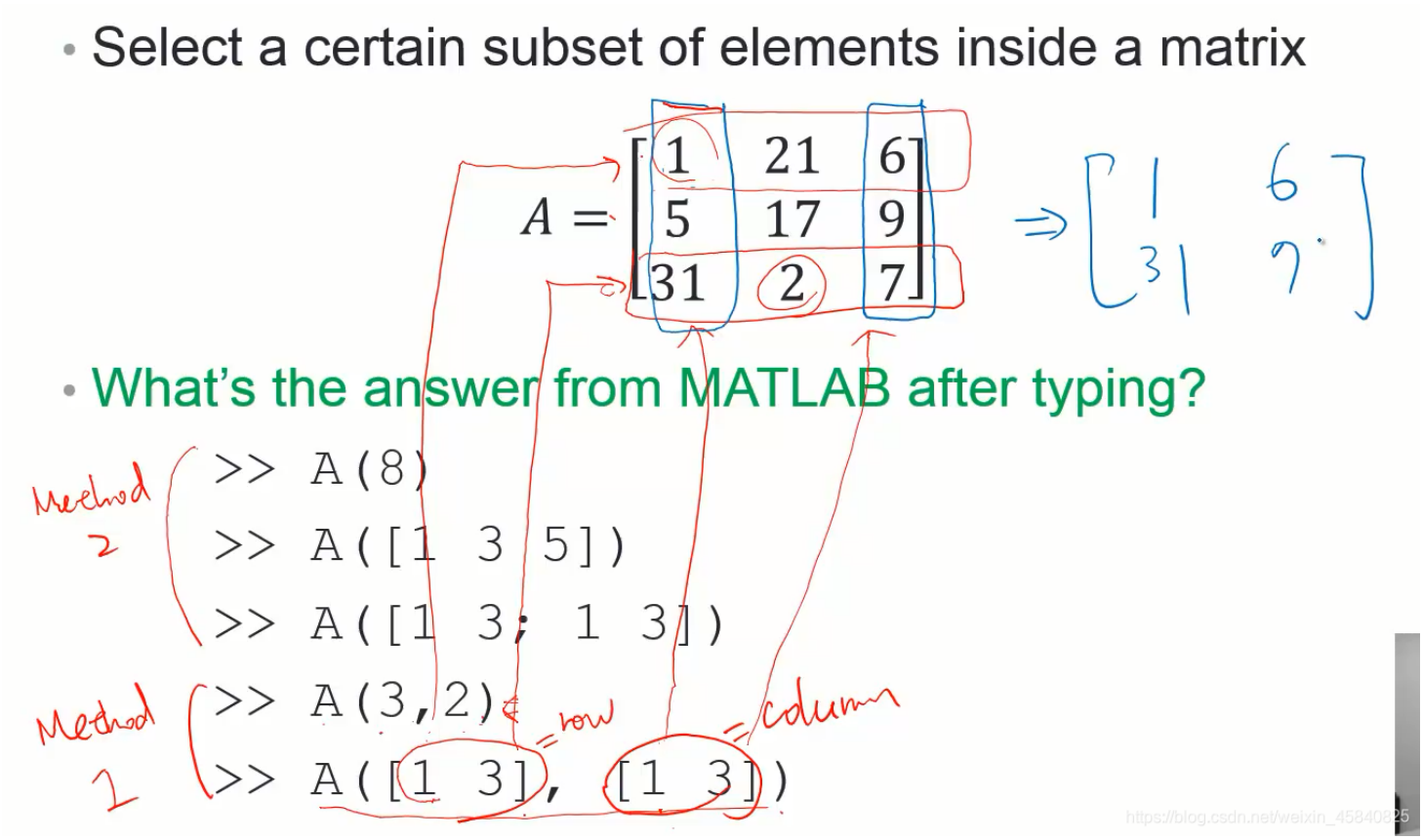
- 对矩阵内元素的使用是一种“引用”的方式，即 利用下标调用矩阵元素 。

- 可以用：在矩阵引用中表示整个被引用(创建向量)，前后有数字表示引用的范围

```
mat1=[1,2,3;5,6,7;9,6,1]
mat1(1,3)    %调用单个元素
mat1(:,2)    %调用某一列元素
mat1(1,:)    %调用某一行元素
mat1(:,:)    %全部调用
```

```
mat1 =

     1     2     3
     5     6     7
     9     6     1


ans =

     3


ans =

     2
     6
     6


ans =

     1     2     3


ans =
```

1	2	3
5	6	7
9	6	1



- 行数相同的矩阵可以使用 `[A1,A2,…]` 或者 `[A1 A2 …]` 的格式串联合并
- 对于比较大且复杂的矩阵，可以选择先数据导入后创建一个特殊的 `.m` 文件的方法存储。需要用的时候再 `load` 导入变量即可

4.矩阵常用函数

1. 创建矩阵
    i. `zeros` :创建一个全零矩阵
    ii. `ones` :创建一个全一矩阵
    iii. `eye` :创建一个单位矩阵
    iv. `rand` :创建一个随机矩阵
    v. `diag` ： 创建一个对角矩阵

> 后面接的格式为：出现一个变量则生成方阵，出现两个变量则生成对应长宽的矩阵

2. 访问元素
    i. `numel` :返回矩阵中的元素的个数
    ii. `size` :返回矩阵的大小
    iii. `length` :返回矩阵的大小

```
>> mat2=rand(3,4)
numel(mat2)
size(mat2)
length(mat2)

mat2 =

    0.7513    0.6991    0.5472    0.2575
    0.2551    0.8909    0.1386    0.8407
    0.5060    0.9593    0.1493    0.2543


ans =

    12


ans =

     3     4


ans =

     4
```

3. 基础运算

i. 对数

log() 表示对数，默认以 e 为底，要说明其他底数的对数，则可在 log 后加数字,如 log2

ii. 指数

用 `a^b` 的格式表示即可；**若要表示自然对数 `e`，用 `exp(1)` 即可**；以 `e` 为底的指数函数用 `exp(n)`

iii. 三角函数

`sin()`， `cos()` 和 `tan()` 都是支持的

iv. 转置

`transpose()` 或者后边加 `'` 可以实现

v. `reshape` :改变矩阵的形状，变成其他长宽的矩阵

vi. `repmat` :复制并拼接矩阵

> 例如A为2X2，repmat(A,2)则是拼成[A,A;A,A]的造型

vii. `kron` :计算克罗内克积*

viii. `dot` :计算点积(内积)

ix. `cross` :计算叉积(外积)

x. `det` :计算矩阵的行列式

xi. `inv` :计算矩阵的逆

xii. `eig` :计算矩阵的特征值和特征向量

xiii. `svd` :计算矩阵的奇异值分解*

xiv. `sum` :计算矩阵元素的积

xv. `prod` :计算矩阵元素的乘积

xvi. `max` :返回矩阵中的最大值

xvii. `min` :返回矩阵中的最小值

xviii. `sort()` :对每一列按从小到大的顺序排列，若只有一行，则对行从小到大排

xix. `sortrows()` :以行为单位，按照第一列的数从小到大排列

xx. `[m,M]=bounds(数组名)` :计算数组中的最小最大值

xxi. `mean` :计算数组元素的平均值

xxii. `median` :计算数组元素的中位数

xxiii. `std` :计算数组元素的标准差

xxiv. `var` :计算数组元素的方差

xxv. `x=A\b` :计算**Ax=b**的解

xxvi. `[A,B]=equationToMatrix([eqn1,eqn2,……],[x,y,……]);` 和 `X=linsolve(A,B)` %求解线性方程组

xxvii. `y=y(x);subs(y,x,val)` %可以来求出 `y=y(x)` 函数在 `x=val` 时的y值

4. 图像可视化

i. 二维平面图象与坐标系

使用线性坐标曲线命令 `plot` ,可以生成线段、曲线和参数方程的函数图像，其格式为 `plot(X,Y)` 或者 `plot(x1,y1,x2,y2,…)` ,其中前者代表创建一个关于(X,Y)的函数图像，后者代表创建多个关于(xi,yi)的函数图像，其中xi和yi是一对相应的点集，yi也可以是关于xi的函数，

`plot` 还支持在点击说明后添加 `LineSpec` 信息，使用指定的线型、标记和颜色创建绘图，且可以同时对多条曲线设置。

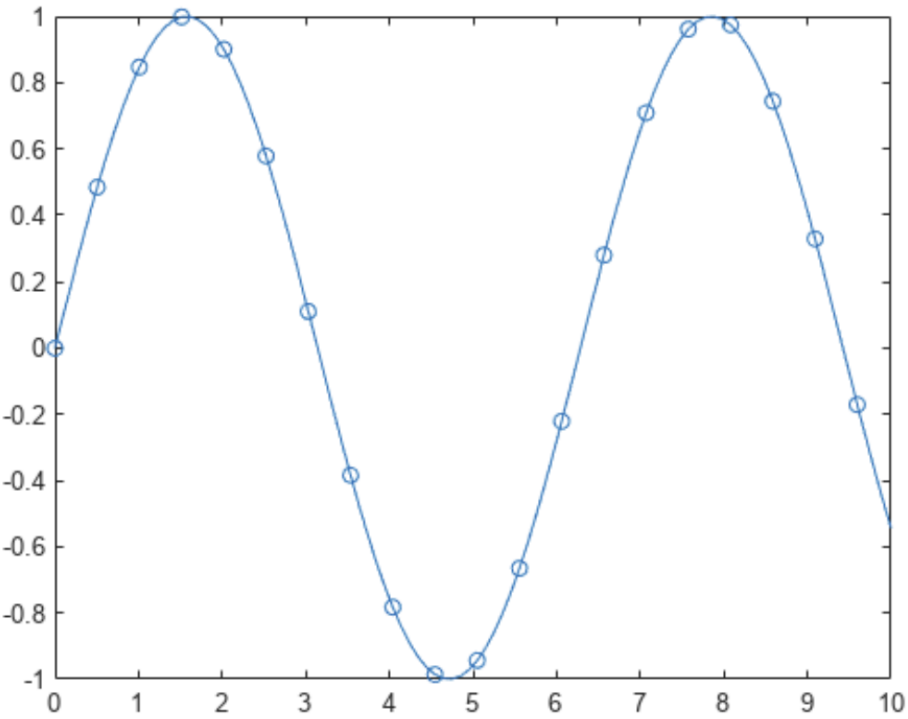
`plot(Y)` 代表一种隐式X坐标的图，如果Y是向量，则X坐标范围从1到 `length(Y)`；如果Y是矩阵，则对于Y中的每一列，图中包含一个对应的行。X的范围是从1到Y的行数; 如果Y包含复数，Matlab会绘制出Y的虚部对应实部的图(复平面),如果同时指定了X和Y，虚部则会被忽略。

`plot(tb,xvar,yvar)` 表示要绘制tb中的变量xvar和yvar。

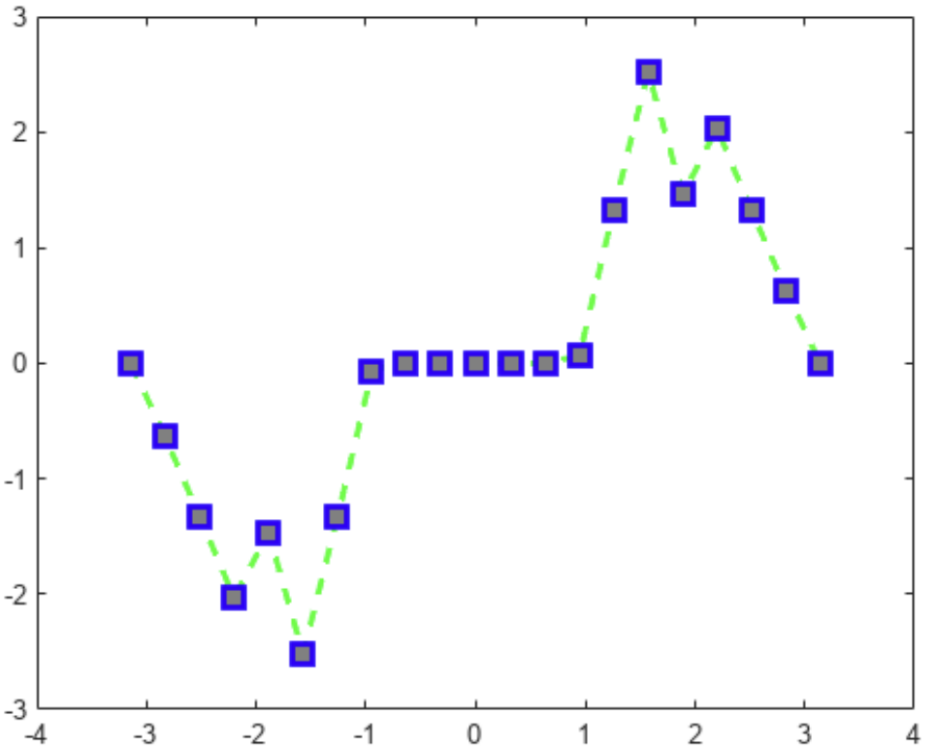
没有变量就要指定一个变量; 要绘制多个数据集，要为xvar和与yvar指定多个变量; 若两个参数都包含多个变量，它们包含的变量数目必须相同。

如以下例子:

```
x = linspace(0,10);
y = sin(x);
plot(x,y,'-o','MarkerIndices',1:5:length(y))
```



```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));

figure
plot(x,y,'--gs',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor',[0.5,0.5,0.5])
```



ii. 三维图像与坐标系

a. `linespace(边界1，步长，边界2)`

设置线性坐标的变量值集合

b. `title()`

设置图像的标题

c. `xlabel()` 或 `ylabel()` 、 `zlabel()`

设置坐标系基向量的名称

d. `[X,Y]=meshgrid(…,…)`

为三维绘图的图像构建在 XOY 平面的取值点集合(或者说图像在 XOY 平面的投影点集)

e. `surf(X,Y,Z)`

形成三维图像

f. `disp()`

等同于 `print()` 函数，展示任意变量或值

iii. 进阶绘图

可以查看这个链接的文件

5.数据导出导入

- 可以用 `load(文件名)` 来包含外部数据文件并将其变成矩阵赋值保存，常见的数据格式有excel和txt
- 可以用 `save(文件名，变量名)` 将变量从工作区保存到文件中，文件格式是 `.mat` (特别用来存数组); 如果文件已经存在, `save` 会覆盖该文件。若在变量名后加 `"-append"` 的说明，便可以使变量多加进文件中，当然若有重复仍覆盖。
  当然也可以写作 `save 文件名` 直接保存，但没说明具体变量，则会把工作区的全部变量都保存了。
  要指定变量则为 `save 文件名 变量名1 变量名2 ……`
- 从工作区删除变量则用 `clear` 命令
- 如果要保留部分精度，则不能使数据在保存时被压缩，这需要使用 `-nocompression` 作为括号内的最后词条表示

- **从Excel中读取表格**
  以前是用 `xlsread()` 和 `xlswrite()` ，不过从R2019a后更新了新的函数，兼容更多的格式，强烈推荐使用：

```
1.
变量名=readmatrix(文件名,'Sheet',选用的sheet名,'Range',读取范围/方形范围)      %固定格式，将指定的
*readmatrix()只读数据，遇到其他数据类别直接跳过
*也可以直接readmatrix(文件名)读取全部数据
2.
变量名=readcell(文件名,'Sheet',选用的sheet名,'Range',读取范围/方形范围)      %固定格式，将指定的范
*其他用法基本与readmatrix()一致
```

- **将数据写出为相应格式**
  i. 可以用 `writematrix()` 将矩阵写入 `.txt` 文档，默认分隔符是 `,` ，若要使用不同的分隔符，则使用 `Delimiter` 名称-值对组

例如以下例子

1.

```
writematrix(变量名)    %默认存成txt文件
type '变量名.txt'     %输出txt文件结果
```

2.

```
writematrix(变量名,自定义文件名,'Delimiter','分隔符类别')    %固定格式，可以自定义存储情况和分隔符
```

3.

```
writematrix(变量名,'文件名.xls')     %将矩阵存储数据写入excel表格
writematrix(变量名,'文件名.xls','Sheet',sheet号码, 'Range',字段范围/开始:结束)     %将矩阵存储数
```

```
M = magic(5)
```

```
M = 5×5

    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
```

将该矩阵写入 M.xls 文件中的第二个工作表，从第三行开始写入。

读取并显示该矩阵。

```
readmatrix('M.xls','Sheet',2,'Range','A3:E8')
```

```
ans = 5×5

    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
```

2. 也可以用 `writecell()` 将cell写入excel文档，其调用格式与上边的 `writecell()` 一致

例如以下操作

```
>> M=magic(6)
M =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
>> title1={'day1','day2','day3','day4','day5','day6'}
title1 =
  1×6 cell 数组
    {'day1'}    {'day2'}    {'day3'}    {'day4'}    {'day5'}    {'day6'}
>> title2={'mut1';'mut2';'mut3';'mut4';'mut5';'mut6'}
title2 =
  6×1 cell 数组
    {'mut1'}
    {'mut2'}
    {'mut3'}
    {'mut4'}
    {'mut5'}
    {'mut6'}
>> writematrix(M,'A.xls','Sheet',2,'Range','B2:G7')
>> writecell(title1,'A.xls','Sheet','sheet2','Range','B1:G1')
>> writecell(title2,'A.xls','Sheet','sheet2','Range','A2:A7')
```

该代码完整写入了表头和数据，效果如下：

	A	B	C	D	E	F	G	H
1		day1	day2	day3	day4	day5	day6	
2	mut1	35	1	6	26	19	24	
3	mut2	3	32	7	21	23	25	
4	mut3	31	9	2	22	27	20	
5	mut4	8	28	33	17	10	15	
6	mut5	30	5	34	12	14	16	
7	mut6	4	36	29	13	18	11	
8								
9								

- **低阶文件处理I/O函数**
  i. fopen(文件名，准许符)

  %无论是写入还是读取都要先打开

  *常见的准许符(permission)有：'r','w','r+','w+','a+','a'

fopen(path, mode);

r 打开只读文件，该文件必须存在。

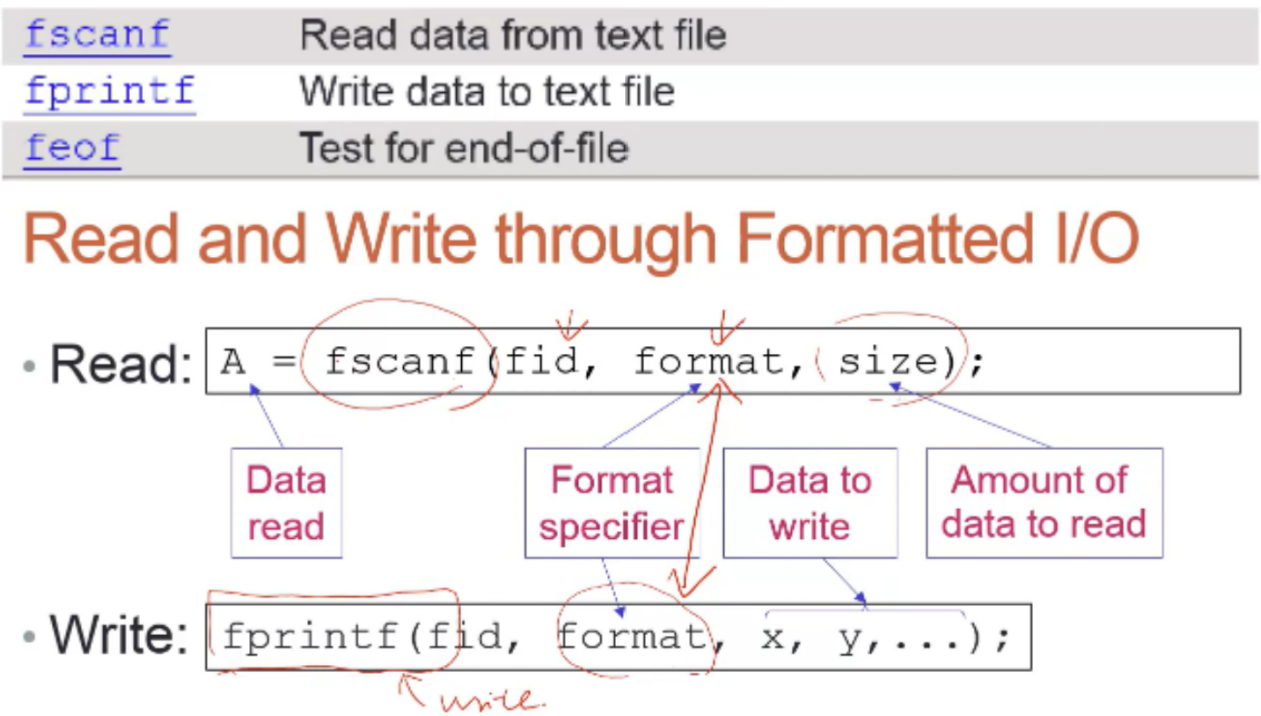
r+ 打开可读写的文件，该文件必须存在。
w 打开只写文件，若文件存在则文件长度清为0，即该文件内容会消失。若文件不存在则建立该文件。
w+ 打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。
a 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。
a+ 以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。
上述的形态字符串都可以再加一个b字符，如rb、w+b或ab+等组合，加入b 字符用来告诉函数库打开的文件为二进制文件，而非纯文字文件。不过在POSIX系统，包含Linux都会忽略该字符。

ii. `fclose(文件名)`

%关闭文件，否则matlab会一直占用文件

iii.

fscanf	Read data from text file
fprintf	Write data to text file
feof	Test for end-of-file



iv. `feof(文件名)`

%检查是否到达文件结尾，到达结尾时返回 `true`

6.数据清理和处理

- 可以使用 `unique()` 函数去除重复数据项
- 可以使用 `isnan()` 函数找出数据中的缺失项，并用 `NaN` 来代替它，让数据表示完整。
- 可以使用

`str2num()`

%将字符数组或者字符串转化成数值数组

`cell2mat()`

%将元胞数组转换成基础数据类型的普通数组
或者说打碎结构体、数组的界限，将数据全部统一排成一个数组内的元素情况。

- 可以使用 数组名(比较条件) 的方法实施数据的简单筛选
- **详细内容可以查看网址Matlab数据预处理内的文章**

7.数据流的输出输入

- input(输入说明文字)
  %请求并接收一个数值输入，若在说明文字后加上"s"，则表示接受的内容不作为数字输入而作为字符串输入
  > 例如 txt=input("Do you want more?","s") 表示接收一个 问题的文本回答

- disp(变量名)
  显示变量的值，不会主动显示变量名称。若变量中包含空数组，则会返回disp，但不显示任何内容
  *如果要在命令行窗口的同一行中显示多个变量值，可以使用字符串构造函数或者外加 [] 变成一个数组。
  > i. 用 [] 将多个字符向量串联起来，对数字使用 num2str 函数将数值转换为字符，从而一起用 disp 输出
  > ii. 用 sprintf 创建文本,然后使用 disp 展示出来，其中字符变量的配位符是 %s ， 数值变量的配位符是 %d 十进制/ %o 八进制/ %x 十六进制/ f 双精度小数/ e 指数
  > iii. 用 fprintf 直接显示文本，这同样要显示配位符，但更重要的是**必须在文本末尾添加换行 \n 字符**

  例如:
  ```
  name = 'Alice';
  age = 12;
  法一 X = [name,' will be ',num2str(age),' this year.'];
  disp(X)
  法二 X = sprintf('%s will be %d this year.',name,age);
  disp(X)
  法三 fprintf('%s will be %d this year.\n',name,age);
  ```

- 可以使用 [s1,s2] 或者 [s1 s2] 的格式将字符串串联连接; 或者使用 [s1;s2] 将字符串并联连接为两行矩阵

7.结构体

Matlab中支持的数据类型有 `logical/boolen` 、 `char` (字符串或者是字符)、 `cell` (元胞)、 `struct` (结构体)、 `numeric` (int8-int64、uint8-uint64、**single(float)**、double)。

1. **numeric**
   可以使用 数据类型(变量名) 的方法强制执行类型转化
2. **struct**
   用

   ```
   变量名=struct(属性1，{属性1的值}，属性2，{属性2的值}，……)
   ```

   的形式表示，其中值可以在创建前赋也可以在创建后赋，属性建议用 `''` 包含词， `""` 包含句，像一个**键值对组**

   上述语句类似于C++的

   ```
   struct xxx{
    ElemType 属性1=xxx;
    ElemType 属性2;
    ……
   };变量名
   ```

3. **cell**
   与矩阵类似，但存储异构数据，每个条目包含不同 `ElemTpye` 的数据,声明使用 `{}` 包含即可，依旧是 , 分列， ; 分行。

   例如以下的例子:

   ```
   c={ , , ;, , };     %创建一个空元胞
   B={1,2,3;'text',datatime('today),hours(1)};     %创建一个2X3的元胞
   B={{1,2,3};'text',datatime('today),hours(1)};     %创建一个2X2的元胞
   ```

   - 使用 `()` 可以索引元胞对应位置的元素名称或者说元素类型;**使用 `{}` 则可以得到具体的矩阵或者 `string`**
   - 以下为一些和元胞相关的函数方法:

   i. `cell2struct()`：将cell数组中包含的信息转变为一个结构体
   ii. `num2cell()`：将数组转化为相同大小的cell
   iii. `mat2cell(数组名,指定的划分方式)`：指定行、列划分数组为cell，例如

A是一个4X6的矩阵

```
C=mat2cell(A,[2,2],[2,4])
%表示把A按行分为2行和2行，按列分为2列和4列，互相交叉配对，变成两个2X2和两个2X4共四个数组元素组成元胞
*若划分只有一组，则默认是关于行的分割
```

iv. 可以用 `celldisp()` 函数来一个一个展示元胞中的元素具体信息

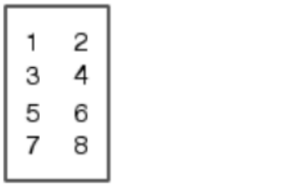
> 元胞常配合 `reshape(生成向量,长宽规格)` 的矩阵生成函数使用

v. 亦可以使用 `cat()` 函数进行数组的串联

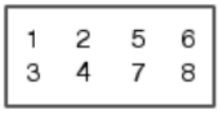
- Array concatenation

```
A=[1 2;3 4]; B=[5 6;7 8];
```

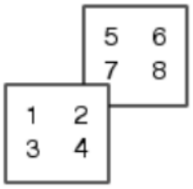
```
C=cat(1,A,B)
```

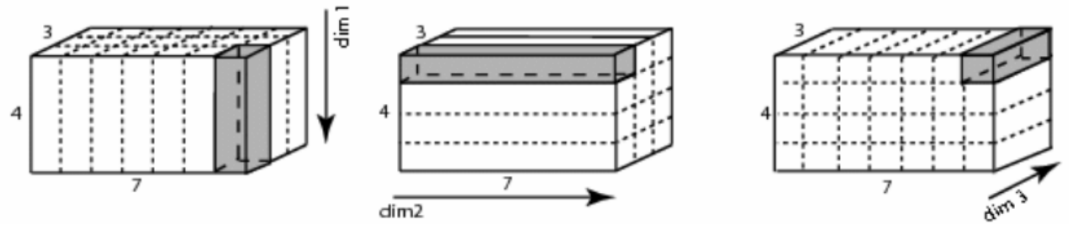



```
C=cat(2,A,B)
```




```
C=cat(3,A,B)
```





- 第一个参数为1：纵向串联，需要A、B两个矩阵行宽（列数）一致；
  第一个参数为2：横向串联，需要A、B两个矩阵列宽一致；
  第一个参数为3：前后三维串联，需要A、B两个矩阵行宽和列宽均一致；
- 直接使用 `[A; B]` 与 `cat(1, A, B)` 效果一致；
  直接使用 `[A B]` 与 `cat(2, A, B)` 效果一致

8.影像处理

详情请看教程1和教程2

9. 数值微积分

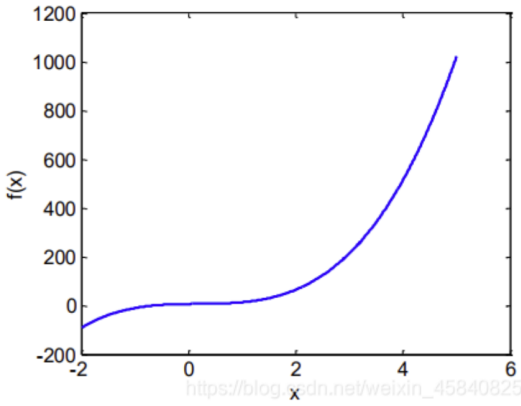
1. 多项式微积分

   i. 求值

   `y=polyval(p,x)`

   %计算多项式 p 在 x 时点集的值，结果生成向量，**参数p是n+1阶向量，是n阶多项式的系数**，排列是**从高阶排到低阶**

```
a = [9,-5,3,7];   x = -2:0.01:5;
f = polyval(a,x);
plot(x,f,'LineWidth', 2);
xlabel('x');   ylabel('f(x)');
set(gca, 'FontSize', 14)
```



- `[y,delta]=polyval(p,x,S`

  %使用 `polyfit` 生成的可选输出结构体 `S` 来生成误差估计值。 `delta` 是使用 `p(x)` 预测 `x` 处的未来观测值时的标准误差估计值。

将一个线性模型拟合到一组数据点并绘制结果，其中包含预测区间为 95% 的估计值。

打开实时脚本

创建几个由样本数据点 *(x,y)* 组成的向量。使用 `polyfit` 对数据进行一次多项式拟合。指定两个输出以返回线性拟合的系数以及误差估计结构体。

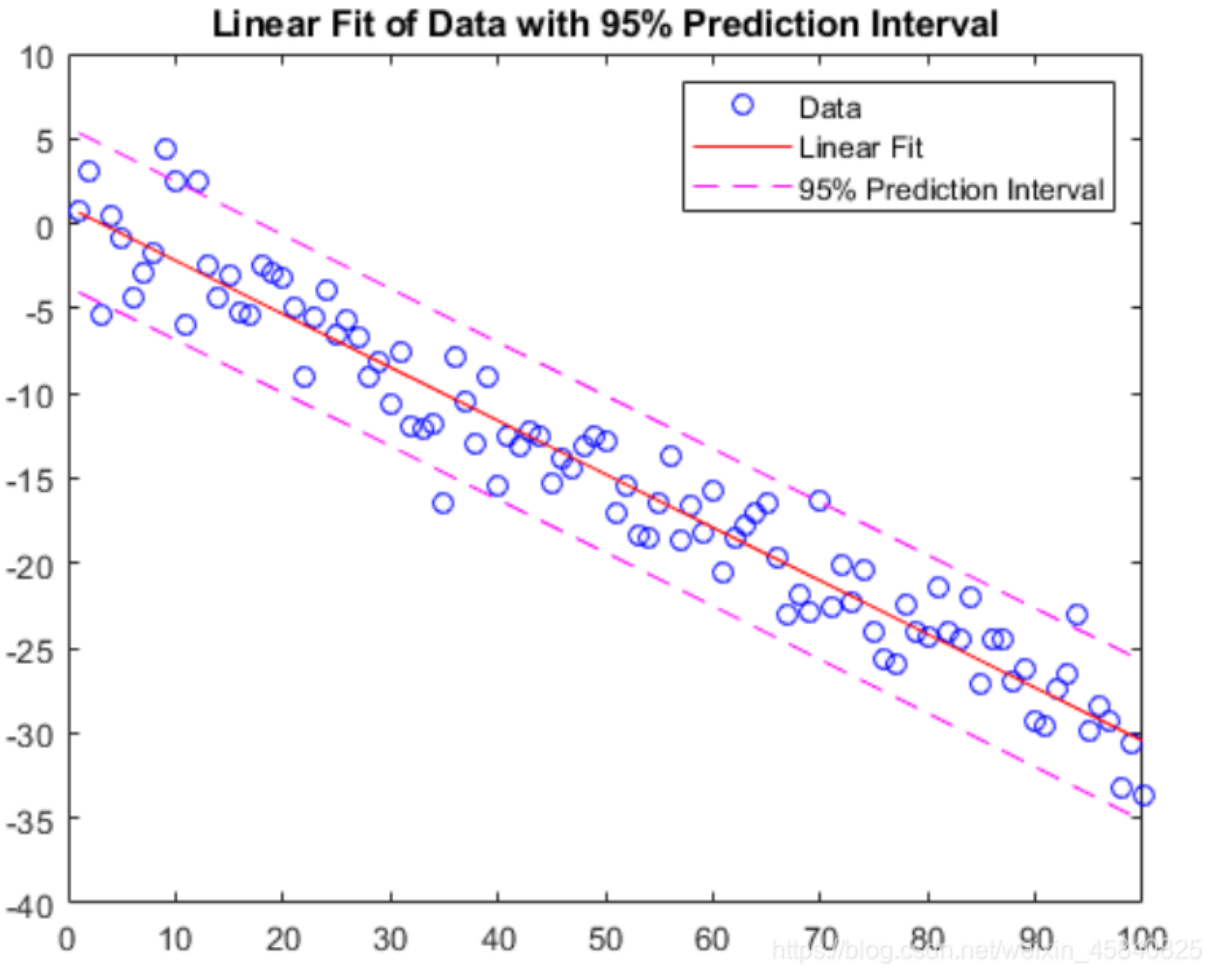
```
x = 1:100;
y = -0.3*x + 2*randn(1,100);
[p,S] = polyfit(x,y,1);
```

计算以 `p` 为系数的一次多项式在 `x` 中各点处的拟合值。将误差估计结构体指定为第三个输入，以便 `polyval` 计算标准误差的估计值。标准误差估计值在 `delta` 中返回。

```
[y_fit,delta] = polyval(p,x,S);
```

绘制原始数据、线性拟合和 95% 预测区间 $y \pm 2\Delta$。

```
plot(x,y,'bo')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
title('Linear Fit of Data with 95% Prediction Interval')
legend('Data','Linear Fit','95% Prediction Interval')
```



ii. 求导

- `y1=polyder(p)`
  %返回多项式 `p` 的导数，是一个向量表示系数，也是从高阶到低阶排列
- `k=polyder(a,b)`
  %返回多项式 `a,b` 的乘积的导数
- `[q,d]=polyder(a,b)`
  %返回多项式 `a`和`b` 的商的导数(`q` 为分子， `d` 为分母)

iii. 积分

- `q=polyint(p,k)`
  %返回多项式积分后的多项式系数，代入的常数项是 `k`
- `q=polyint(p)`
  %常数项默认为0

2. 数值微积分

i. 求导

需要人工构造对象，使用

```
syms x     %不用';',表示声明一个变量形象x
f=f(x);    %声明一个函数
df=diff(f);    %对函数求导数，得到含有x的表达式
disp(df);    %输出导数表达式
ans=subs(df,x,val);    %将x=val带入函数表达式y=df(x)中求解出y的值赋给ans
```

ii. 积分

使用 变量名=integral(函数名，x下限，x上限) 来求二重积分，而三重积分是 integral2() ,其余以此类推

*计算广义积分，正/负无穷可以用 -inf/inf 来表示

*在添加 plot 的图例时，在字符串中显示单引号，**打两个单引号"才相当于一个单引号显示**

10.方程求解

> **符号变量**
> 这是一种特殊的变量，以 syms x 的方法声明，matlab会将它当成符号参与计算，通过计算得到的结果也是符号变量(syms变量)。

1. 求解一般方程

使用 s=solve(eqn,var) 格式的式子求解, eqn 是一般的方程等式，*而不是字符串*; **上述式子的意思是求解关于变量var的方程eqn**，例如:

```
syms x
solve(2*x^2-3*x==16,x);
```

2. 求解方程组

使用 s=solve(eqn1,eqn2,…,x1,x2,…) 的格式计算，计算出的结果为**结构体**，包含多个解 x1,x2,… 作为成员，可以通过 s.x1, s.x2, …… 参看结果

*除此之外， **eqn 的赋值格式为 eqn=F(x1,x2,……)** 。其实际方程是 F(x1,x2,……)==0 ,这是方程 f(x1,x2,……)==k 的变种

- Solve this equation using symbolic approach:

$$\begin{cases} x - 2y = 5 \\ x + y = 6 \end{cases}$$

```
syms x y
eq1 = x - 2*y - 5;
eq2 = x + y - 6;
A = solve(eq1,eq2,x,y)
```


3. 求解符号方程

`solve()` 还可以用来求解用符号代替常数的方程。

例如:

```
syms x,a,b
solve(a*x^2-b)    %系统会默认是x
solve(a*x^2-b,b)    %将要求出的对象圈定为`b`
```

求符号函数的导数，直接使用 `diff` 即可

例如:

```
syms x
y=4*x^5;
yprime=diff(y)
```

求积分函数的积分，直接使用 `int` 即可

例如:

```
syms x
y=x^2-x*exp(x);
z=int(y);
z=z-subs(z,x,0)     %也就是求解初值为z(0)=0的解
```

4. 数值解

- 使用 `fsolve(函数名，值)` **可以从 `x0` 开始，找出离 `x0` 最近的 `函数=0` 的根**

方程的一种声明格式：
方程名 = @(自变量)（含自变量的函数格式内容）
例如：

```
fn=@(x)(exp(x)+1);
fsolve(fn,x0)
```

- **如果值是矩阵，则返回的结果也会是矩阵**
- 若要解方程式，则需将方程组的方程存入同一个函数处理，其中 `root*d` 为组合函数， `*` 填入方程组的数目。
  它将作为一个结构体存储多个方程和多个变量，也可以用结构体来调用

如以下例子：

```
function F =root2d(x)
F(1)=exp(-exp(-x(1)+x(2)))-x(2)*(1+x(1)^2);
F(2)=x(1)*cos(x(2))+x(2)*sin(x(1))-0.5;    %完成函数设置
fun=@root2d;
x0=[0,0];    %设置基准点
x=fsolve(fun,x0)     %求解
```

- 与 `fsolve()` 类似，但 `fzero()` 求出的解是**真解**，即使得 `f(x+)` 和 `f(x-)` 异号的点，故**fzero解不了如x^2方程的根**
- 使用 `roots()` 函数可以求出多项式方程的解，参数是接收一个作为多项式系数的**向量**，系数是从高阶到低阶排列。

例如：

```
p=[1,-3.5,2.75,2.125,-3.875,1.25];     %设置相应的多项式
roots(p)     %求解相应的多项式
```

> * `inv()` 可以求方程的逆矩阵(前提矩阵是方阵且可逆)，**如果变量是 `syms` 类型元素组成的，也可以直接使用**

11.统计

详细内容可以观看这里的内容

12.线性系统

详细内容可以观看这里的内容

13.回归与内插

详细内容可以观看这里的内容

14.函数与脚本

1. 函数定义

在Matlab中，函数 `function` 的语法为：

```
function [输出的形参] = 函数名(输入的形参)
    函数代码内容
end
```

其中，有效的函数名称要以 字母字符开头，可以包含字母、下划线或者数字 。
【注意】：
1.**只保存函数定义的脚本文件，文件的名称应与文件中其函数的名称一致。**
2.包含命令和函数定义的脚本文件中，**函数必须写在文件的末尾，让命令(main)在前**，脚本文件不能与文件中的函数具有相同的名称(排他性)

> *在仅包含函数定义的函数文件中，局部函数可以任意顺序出现在文件的主函数后边。

3.局部函数/嵌套函数是只在主函数中发挥作用的函数，**尤其需要注意**
局部函数/嵌套函数在文件中的位置。

2. 半个函数

- 声明

1. 匿名函数
   以下语句为了创建匿名函数句柄：

```
句柄名(函数名) = @(形参表) 函数内容     %最好函数内容只有一行
```

2. 内联函数

- 比匿名函数更不好用，不过也是不需要创建M文件(.m)的。
  以下语句为创建的格式:

```
函数名 = inline('函数表达式','变量1','变量2',……)
```

- 调用
  它可以用 变量 = 函数名(实参列表) 来调用函数
- **【注意】**

> 匿名函数具有内联的所有优点，并且效率比更高。匿名函数的主要功能是：
> （1）可以代替“将函数编写为单独的m-文件”；
> （2）可以实现符号函数的赋值运算；
> （3）很方便地对含参变量函数进行操作。

3. 函数调用

- 其格式为:

```
[输出实参表] = 函数名(输入实参表)
```

- **注1**：函数中遇 return 语句时，将退出函数体，此函数调用结束；
  注2：函数体里面也可以定义一个或几个函数，称为子函数；注意：子函数只能存在于主函数体内，不独立存在；子函数在主函数体内的位置可以任意，不影响使用；子函数只能被主函数以及其他位于同一主函数体下的子函数调用，但子函数“句柄”例外；
  注3：在调用函数时，Matlab用两个永久变量 nargin 和 nargout 分别记录调用该函数时的输入实参和输出实参的个数(narg)。只要在函数文件中包含这两个变量，就可以准确地知道该函数文件被调用时的输入输出参数个数，从而决定函数如何进行处理。

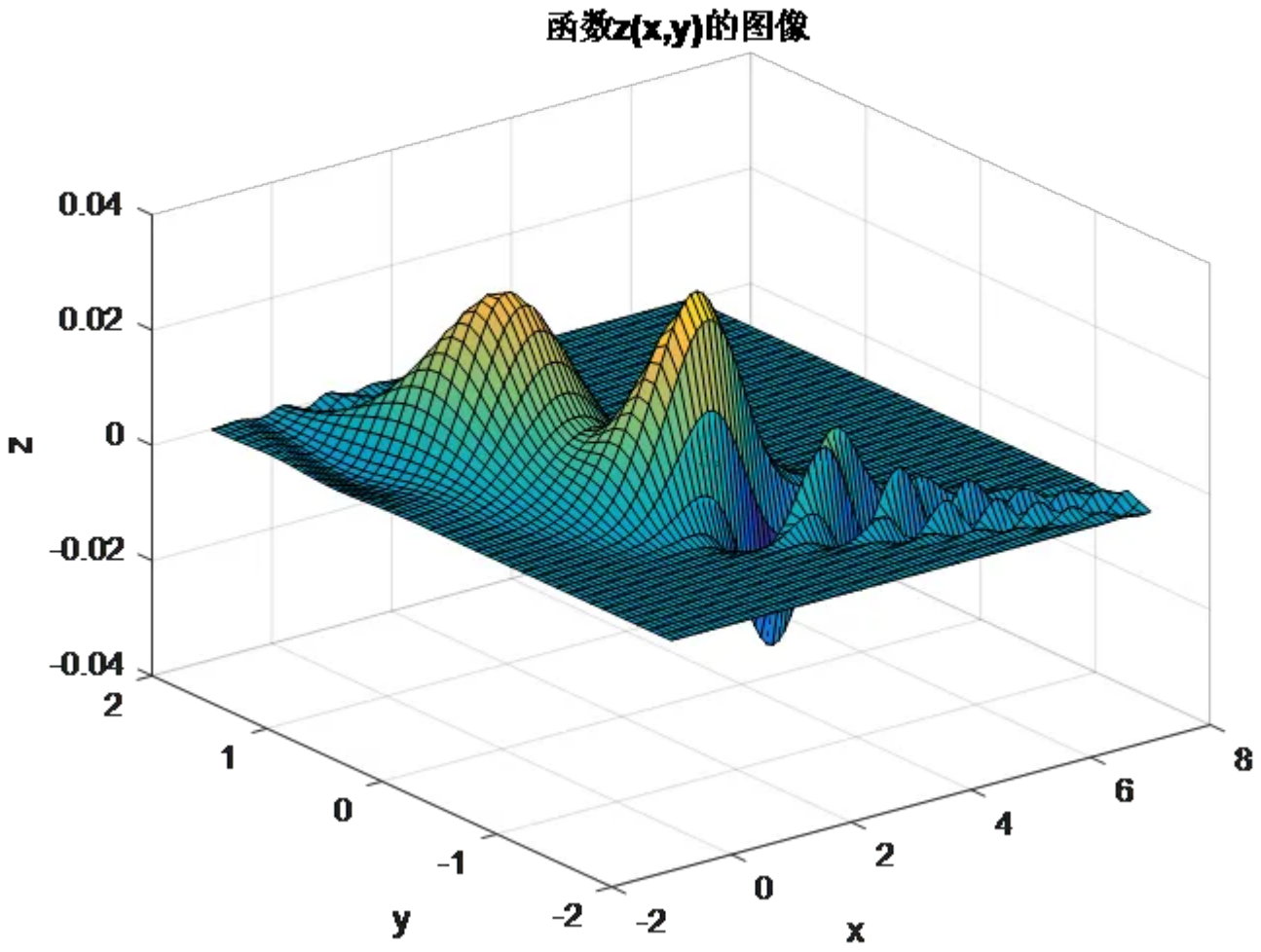
4.隐函数表达

- 隐函数一般无法给出显式表达式，但借助匿名函数和求根函数fzero()可以实现“已知隐函数表达式，对于给定的自变量，通过数值方法求出因变量”。
  如以下的例子：

例4 “显式”表示下列隐函数：

$$z=\sin\left((zx-0.5)^2+2xy^2-\frac{z}{10}\right)\exp\left\{-\left[x-0.5-\exp(-y+z)\right]^2+y^2-\frac{z}{5}+3\right\}$$

```
z = @(x,y) fzero(@(z) z-sin((z*x-0.5)^2+2*x*y^2-z/10) *exp(-((x-0.5-exp(-y+z))^2+y^2-z/5+3)),ra

z(2,0.5)  %求x=2, y=0.5时的z值

%绘制z(x,y)的图像
[X,Y] = meshgrid(-1:0.1:7,-2:0.1:2);
Z = arrayfun(@(x,y) z(x,y),X,Y);

surf(X,Y,Z)
xlabel('x')
ylabel('y')
zlabel('z')
title('函数z(x,y)的图像')
```



5.脚本调用

- 在Matlab中，可以使用 `addpath()` 函数将脚本和函数的文件夹添加到Matalb的搜索路径中
- 接着，可以用 `run()` 函数调用并运行脚本
  例如以下例子：

```
addpath('文件路径');     %如果脚本和函数就在当前目录，则无需使用addpath函数
run(脚本名.m);
变量名 = 函数名(输入形参表)
```

15.控制语句

1.循环控制

- Matlab中包含 `for` 和 `while` 两种循环语句，且要用 `end` 关键字结尾

> 如果意外创建了一个无限循环（永远不会自行结束的循环），请按 Ctrl+C 停止执行循环。

1. **for语句**

```
for 判定函数 = 多个值(可以用向量或者':'生成)
    循环体内容
end
```

2. **while语句**

```
while 判定表达式(不等式判断或者是非判断)
    循环体内容
end
```

> while 1 和break的组合可以保持程序的一致连贯性(**即一直在工作**)

3. **break和continue**
   与其他编程语言一样， `break` 会跳出所有循环， `continue` 会跳出当前循环(进入下次循环)

2.选择控制

- Matlab用于实现选择结构的语句有三种: `if` 、 `switch` 、 `try-catch` ，且结尾都要用 `end` 结尾。

1. **if语句**

(1)单选择

```
if 条件语句
    代码内容
end
```

(2)双选择

```
if 条件语句1
    代码内容1
else
    代码内容2
end
```

(3)多选择

```
if 条件语句1
    代码内容1
elseif 条件语句2
    代码内容2
……
elseif 条件语句n
    代码内容n
else
    代码内容n+1
end
```

2. **switch语句**

同样，也要使用 end 结尾

```
switch  变量值
    case val1
        代码内容1
    case val2
        代码内容2
    ……
    case valn
        代码内容n
    otherwise
        代码内容n+1
end
```

3. **try-catch语句**
   用于执行语句并捕捉程序可能产生的错误

```
try
    代码部分
catch
    对于错误部分的操作内容
end
```

   对于对错误部分的处理，具体信息可以查看以下的相关内容1、相关内容2、相关内容3、相关内容4

3.暂停控制

当程序运行时，为了查看程序的中间结果或者观看输出的图形，有时候需要暂停程序的执行。这时候就需要使用 pause 函数，其标准格式是:

```
pause(延迟秒数)
```

如果省略延迟时间，直接使用 pause ，则会暂停程序，直到用户**按任意键继续执行。**

16.GUI设计与应用

详细可以查看以下内容

1. MATLAB之GUI界面介绍与搭建
2. What Is a MATLAB GUI?
3. MATLAB | 如何写一个简单GUI程序?
4. MATLAB GUI学习手记(一本书够了，内容我存在后边)
5. Learning to Program with MATLAB: Building GUI Tools(同样一本书够了，同上)
6. 如何用MATLAB做图形用户界面(GUI)?| 知乎