

MATLAB 使用

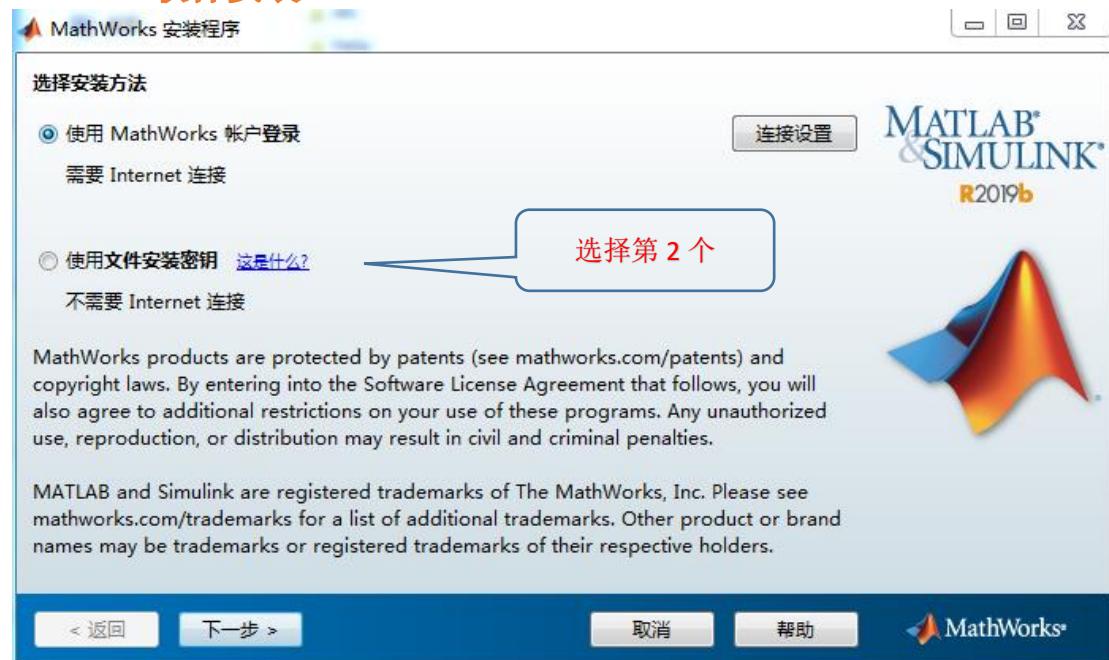
作者:向仔州

MATLAB 软件安装.....	4
MATLAB 代码编辑文本和命令行文本字体大小设置.....	7
MATLAB GUI 图形界面设计.....	8
实现文本输入, 按钮功能.....	8
文本框创建.....	8
按钮创建.....	8
get 函数使用 get(handles, 属性).....	10
set 函数使用 set(handles, 属性, 变量).....	10
滚动条数据显示功能.....	11
让 Matlab GUI 窗口大小可以调整.....	12
单选按钮实现.....	13
勾选按钮实现, 和单选按钮一样.....	13
绘制多种曲线.....	14
按钮组使用.....	14
axes 函数使用.....	16
Switch 语法使用.....	16
下拉菜单实现.....	16
参数列表控件 Listbox (列表框).....	18
Matlab GUI 如何重复打开已经布局的界面, 修改界面布局.....	19
按钮点击, 导入桌面文件操作.....	19
[file,path,index] = uigetfile(指定文件扩展名,导入对话框名称).....	19
寻找 CSV 和 excel 注意事项.....	21
指定路径保存生成的 csv 文件.....	21
[file,path,indx] = uiputfile('test.csv'); %执行之后会生成个对话框, 指定文件保存路径.....	21
Matlab 读取 excel 数据.....	22
plot 曲线绘图函数使用, plot(横座标变量, 纵座标变量).....	23
MATLAB 语法详解.....	24
for 循环使用.....	24
for 循环也可以用矩阵来循环.....	24
while 循环使用.....	25
if , elseif, else 的使用, 类似 C 语言, if , else if , else.....	26
switch case end 使用.....	27
try catch end 使用, 就是用来判断函数程序是否执行成功	28
det 函数使用 det(参数).....	28
break 和 continue 在 for 和 while 使用.....	28
直接构造数组.....	29
增量法构造数组.....	29
linspace 函数用法.....	29
创建矩阵.....	30
grid on 和 hold on 函数用法.....	30
Matlab 一维插值算法函数使用.....	31
Interp1 函数使用.....	32
nearest 邻近插值.....	34
三次样条插值(用得最多, 接近实际情况的算法).....	35
spline 三次样条插值.....	35
拉格朗日插值法算法实现.....	36
function 函数定义方法.....	38
二维插值(曲面插值), 两个输入值 x, y, 一个输出值 z.....	40
surf 三维图函数使用.....	42
interp2 二维插值函数使用.....	42
二维插值做离散点很大的插值算法.....	43
plot3 函数使用 绘制 x, y, z 的图.....	43
griddata 函数使用.....	44
contour 函数使用, contour(x 座标, y 座标, z 座标).....	45

figure 关键字使用.....	46
线性拟合实现.....	47
ones 函数用法.....	49
多项式拟合.....	51
polyfit Matlab 自带多项式拟合函数使用.....	52
polyval 函数使用 输出变量 $y = \text{polyval}(\text{系数}, \text{自变量 } x)$	53
非线性拟合.....	56
残差平方和的定义.....	56
人口预测案例.....	56
lsqcurvefit 非线性拟合函数使用.....	60
土壤含水率案例.....	62
MATLAB 串口通信.....	63
fopen(对象变量).....	63
fprintf(对象变量,要发送的字符串).....	63
fclose(对象变量) 关闭串口.....	63
注意有时候打开串口会失败.....	63
delete(instrfindall) %删除所有串口设备, 执行该函数之后才能做串口相关操作.....	63
串口接收数据实现.....	64
记住如果不再使用串口对象要顺序执行以下三条命令.....	65
串口中断接收数据.....	66
串口中断接收一帧数据, 然后对数据里面的数据进行多字节拼接.....	66
[变量 1,变量 2]=fread(.....) //串口 fread 返回两个变量.....	66
变量 = fread(obj,接收字节数,'uint8') //串口返回 1 个变量.....	67
串口字节拼接.....	68
C = Bitor(A,B) %或运算,将 A 变量与 B 变量进行位或, 将位或之后的值返回给 C 变量.....	68
返回值 = Bitshift(变量,左移多少位) %左移运算.....	68
看看右移运算.....	68
全局变量, 两个不同的 m 文件相互调用.....	69
全局数组两个不同的 m 文件相互调用.....	69
MATLAB 动态绘制曲线图.....	70
plot(.....) %曲线工具多参数设置.....	70
对以上动态绘制曲线的方法进行详细介绍.....	72
axis 函数将 x 轴的间距缩小会发生什么?.....	75
在主循环中无限执行 plot 可能会让绘图不断重复造成卡程序.....	76
line 在 GUI 绘制曲线中的使用, 替代 plot 绘图函数.....	78
多个 GUI 曲线窗口, 如何实现每条数据线对应一个窗口.....	79
定时器使用.....	83
MATLAB 向 Excel 写数据.....	84
xswrite(....) %向 excel 写数据.....	84
如果在 xswrite 写 excel 过程中出现服务器出现意外情况, 这不是网络问题.....	84
自定义数据格式写入 excel 文件.....	85
MATLAB 连续写数据到 excel, 做实时采集数据存储.....	86
返回值(字符数组) = num2str(变量) %将数值数组转换为表示数字的字符数组.....	86
Int2str(变量) %将十进制转换成字符串 就算填入 ff 转换出来也是 255 显示.....	86
xswrite(文件路径, 写入 excel 的变量, 写入 excel 哪一页, 写入页中某一格).....	86
一维数组使用.....	86
MATLAB 全局变量使用注意事项.....	87
串口实时采集数据存入 Excel 案例(全局变量的使用).....	87
将串口数据放入缓存, 然后再存入 excel 数据(未完成).....	88
MATLAB 读取 excel 数据进行绘图.....	88
MATLAB 获取本地时间.....	89
返回值 = datetime %基本获取时间函数.....	89
返回值 = datestr(变量) %将日期和时间按照指定的格式转成成字符形式.....	89
字符串写入 excel 注意事项.....	90
字符串分隔.....	90

MATLAB 读写 txt 文件(解决 <code>xlswrite</code> 写 excel 速度慢的问题).....	91
换行写入字符串.....	91
让 txt 文本每一列数据排列整齐.....	91
读取 txt 文本多列多行数据.....	91
返回几 x 几矩阵 = <code>textscan</code> (句柄, 列格式变量) %读取 txt 文本多行多列数据.....	92
CSV 文件创建与操作.....	93
文件 IO 方式写 CSV 文件.....	94
时间戳写入 CSV 文件, 需要字符转整数.....	95
返回浮点数 = <code>str2num</code> (字符串变量) %将字符串转换为浮点数.....	95
读取 CSV 文件中的数据进行解析.....	95
矩阵 = <code>csvread</code> (文件名/文件路径) %获取 CSV 文件内容, 返回给矩阵.....	95
返回值 = <code>size</code> (变量) %返回变量的行列数.....	96
<code>M(:,1)</code> %这种冒号是取出第 1 列所有数据, 返回给 x.....	96
字符串拼接.....	97
MATLAB 在 GUI 模式下设置 Plot 曲线显示卡死, 在 2019b 版本中,GUI 模式下绘图是用 Plot, 但是曲线放大缩小必须用 <code>axtoolbar</code> 函数实现(重点).....	98
返回值 = <code>axtoolbar</code> (GUI 显示得句柄,{参数设置}) %给 plot 显示得曲线增加放大缩小, 数据显示等功能.....	98
如果坐标系长宽设置过大, 可能显示会超出边框(注意).....	99
<code>axtoolbar</code> 多条曲线显示.....	99
数值过大, 坐标轴显示问题.....	100
<code>set(gca,'XTickLabel',X 坐标开始值 :X 坐标刻度步进:X 坐标最大值)</code> %设置 x 轴刻度线.....	101
MATLAB 在关闭 GUI 界面的时候需要增加一些回调函数, 让正在死循环的主程序也能关闭, 如果 GUI 界面关闭了, 不清除串口, 主程序死循环, 就会导致 MATLAB 多次启动后很卡.....	102
MATLAB 图表使用.....	103
直方图绘制.....	103
<code>bar3</code> (多少列纵轴数据, 每个数据高度) %三维直方图.....	104
饼图绘制.....	105
<code>log</code> 表示曲线(数据量大适合使用).....	106
面积图.....	106
极坐标使用.....	107
散点图绘制.....	108
MATLAB 生成 C 语言代码.....	117
Simulink 基本使用.....	117
Simulink 未完待续.....	

MATLAB 软件安装

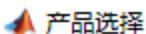


安装密匙 09806-07443-53955-64350-21751-41297



一定选择英文目录的安装路径

工具箱选择



选择要安装的产品

<input type="checkbox"/>	产品
<input checked="" type="checkbox"/>	MATLAB Parallel Server 7.1
<input checked="" type="checkbox"/>	MATLAB 9.7
<input checked="" type="checkbox"/>	Simulink 10.0

<input checked="" type="checkbox"/>	Control System Toolbox 10.7
<input checked="" type="checkbox"/>	Curve Fitting Toolbox 3.5.10
<input checked="" type="checkbox"/>	Data Acquisition Toolbox 4.0.1

<input checked="" type="checkbox"/>	MATLAB Coder 4.3
<input checked="" type="checkbox"/>	MATLAB Compiler 7.1
<input checked="" type="checkbox"/>	MATLAB Compiler SDK 6.7

<input checked="" type="checkbox"/>	Partial Differential Equation Toolbox 3.3
-------------------------------------	---

<input checked="" type="checkbox"/>	Signal Processing Toolbox 8.3
-------------------------------------	-------------------------------

<input checked="" type="checkbox"/>	Statistics and Machine Learning Toolbox 11.6
<input checked="" type="checkbox"/>	Symbolic Math Toolbox 8.4

<input checked="" type="checkbox"/>	Global Optimization Toolbox 4.2
-------------------------------------	---------------------------------

<input checked="" type="checkbox"/>	Audio Toolbox 2.1
-------------------------------------	-------------------

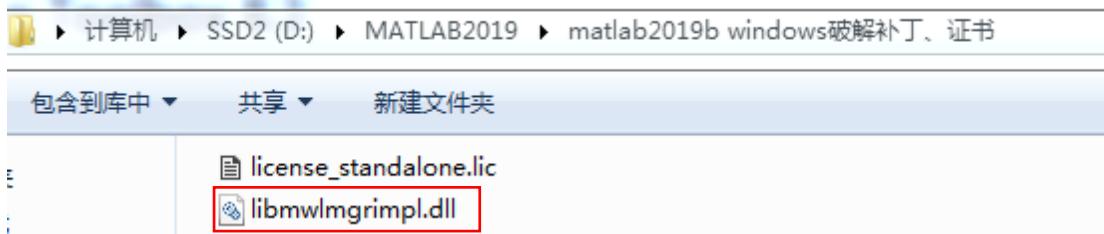
最好是全部选择上当然你可以自己测试下，因为选择性勾选发现有些工具箱没有安装上。
比如我的统计学与机器学习工具箱就没有安装上，导致 regress 函数无法使用。

然后选择否，不需要附加产品，开始安装

开始破解

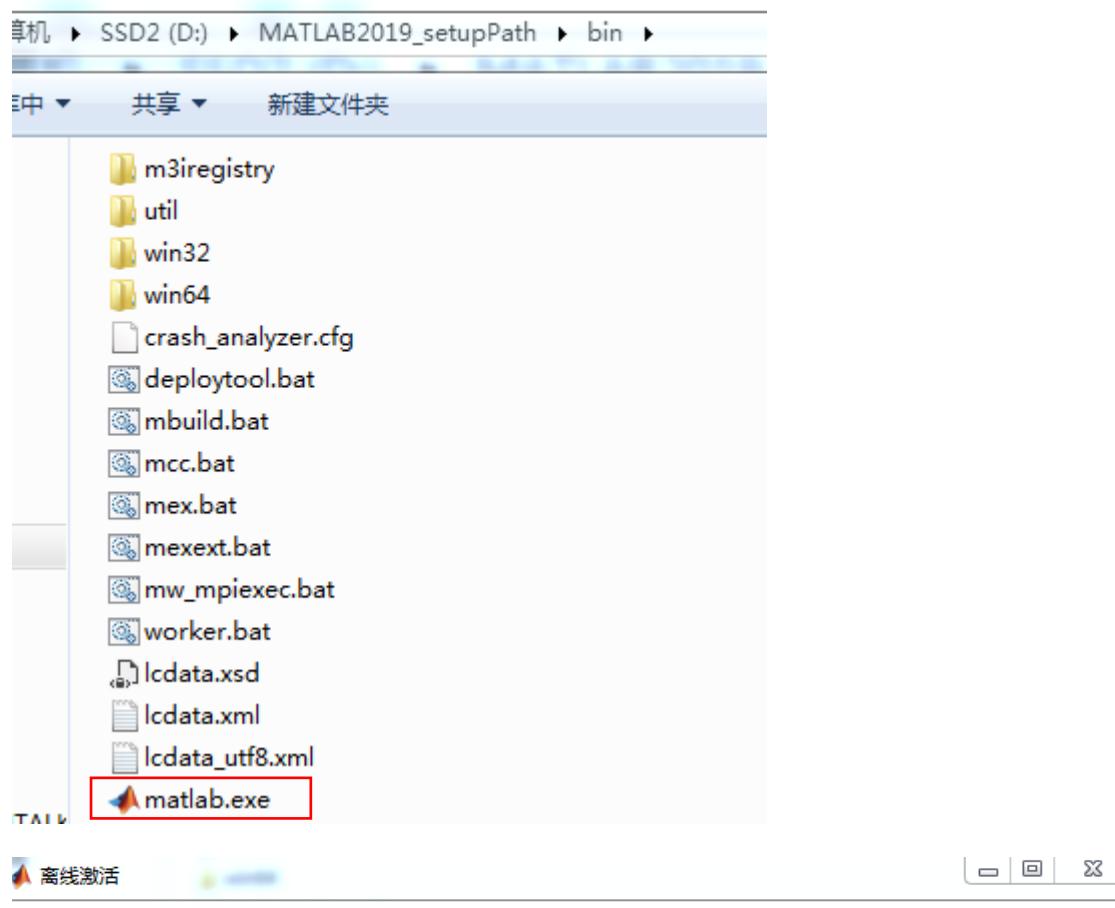
记住破解时一定要断网，或者断开网卡。断网，断网，断网。

进入你安装路径下的 D:\MATLAB2019_setupPath\bin\win64\matlab_startup_plugins\lmgrimpl 目录

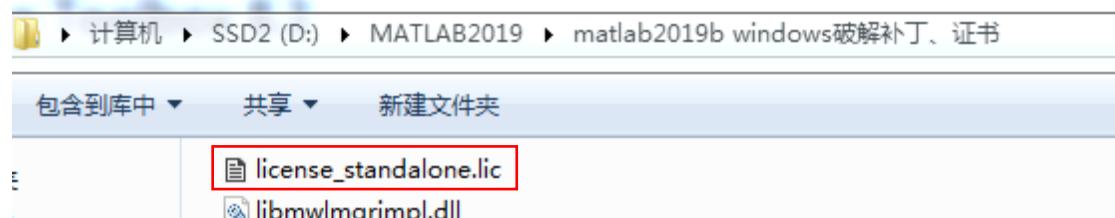


将破解的 dll 文件覆盖进去

回到 bin 目录运行 MATLAB



将浏览选择，你破解目录下的 lic 文件



这样你的 MATLAB 就激活了，



给这个 exe 创建快捷方式，运行就是

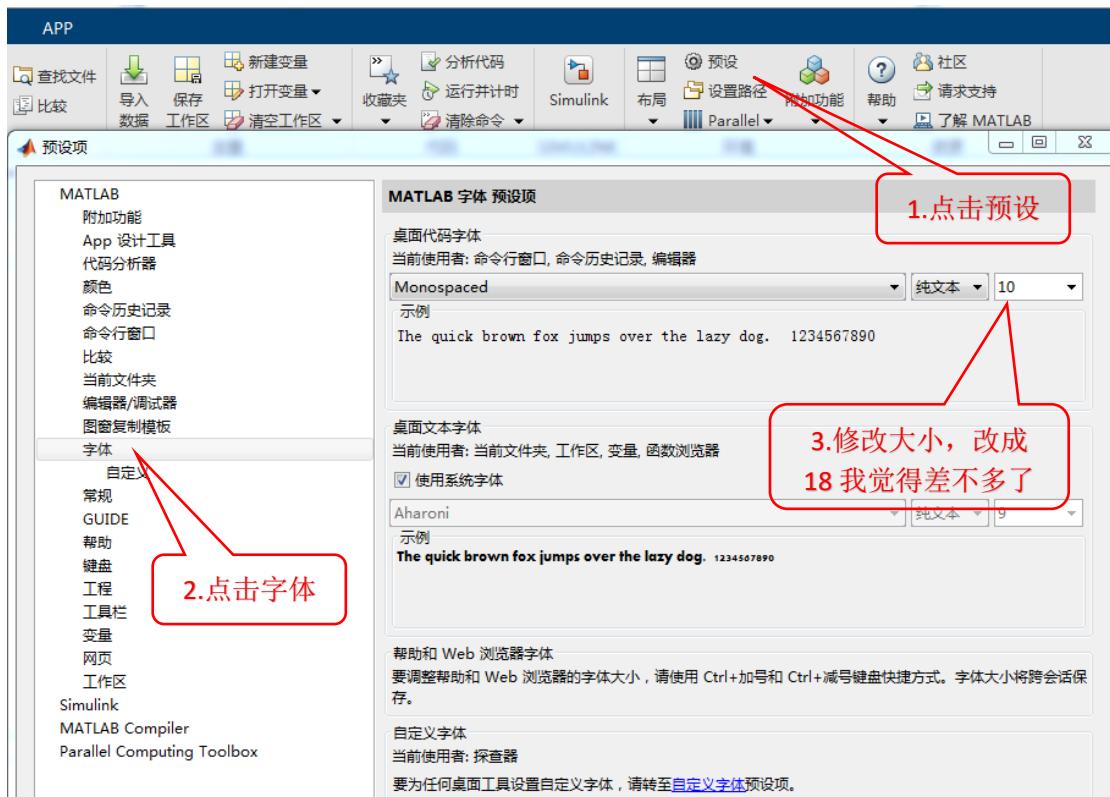
MATLAB 代码编辑文本和命令行文本字体大小设置

命令行窗口

```
列 25 至 36
1.2000 1.2500 1.3000 1.3500 1.4000 1.4500 1.5000 1.5500 1.6000 1.6500 1.7000 1.7500
列 37 至 41
1.8000 1.8500 1.9000 1.9500 2.0000
>> test(x)
ans =
```

代码和命令行字体太小

```
function y = test(x)
y=x.*exp(1-x);
plot(x,y);
xlabel('x');
ylabel('y');
end
```



test.m

```
function y = test(x)
y=x.*exp(1-x);
plot(x,y);
xlabel('x');
ylabel('y');
end
```

列 37 至 41

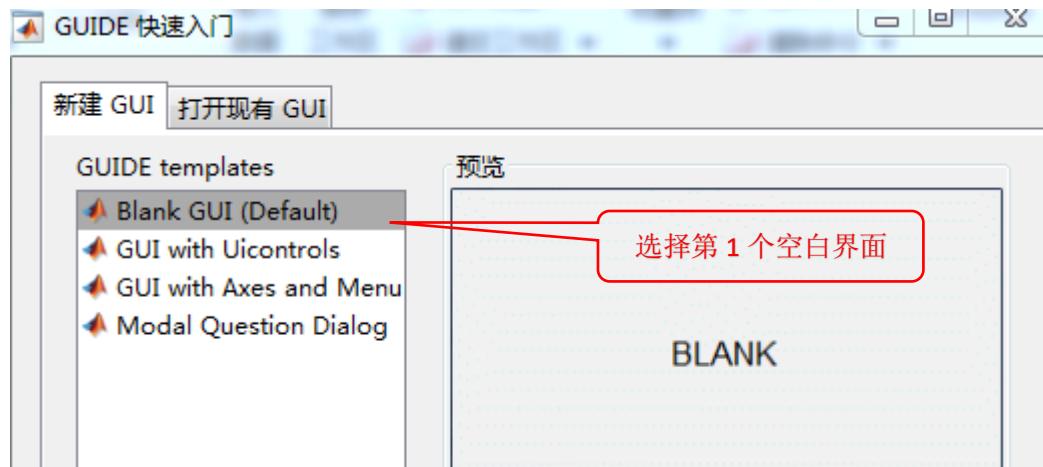
0.8088 0.7907 0.7725 0.7541 0.7358

fx >>

MATLAB GUI 图形界面设计

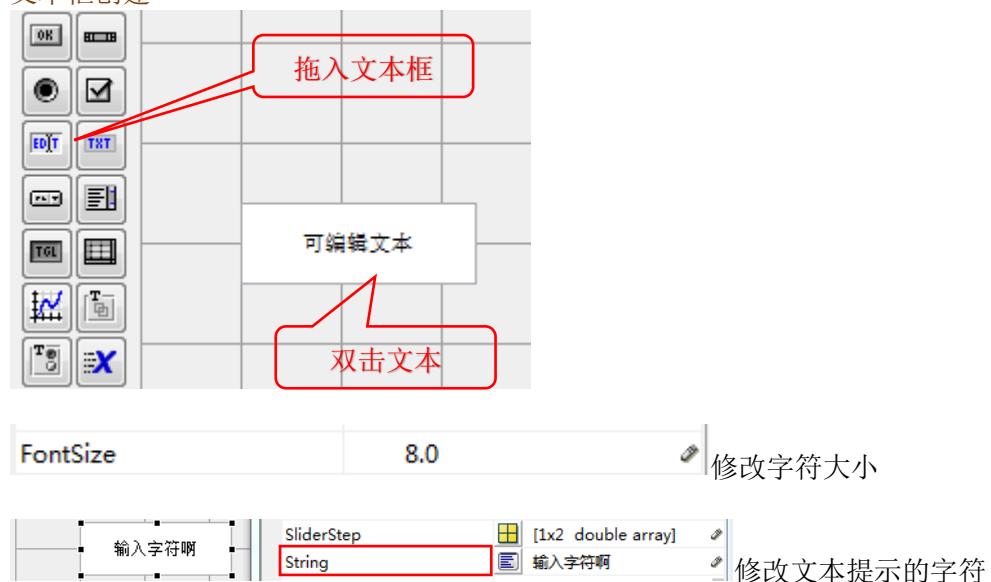
命令行窗口

>> guide
输入 guide 回车

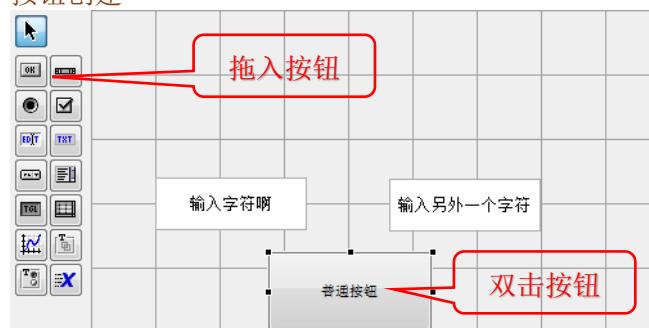


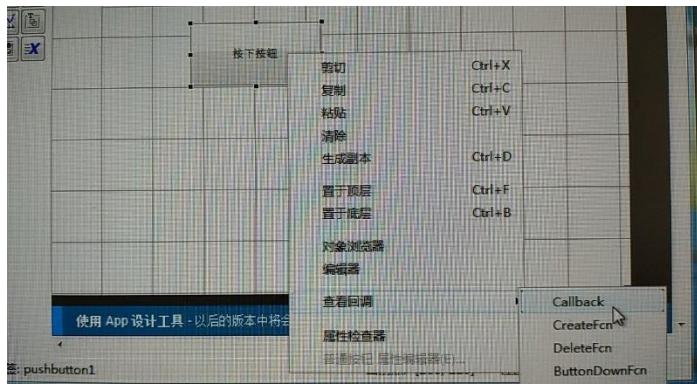
实现文本输入，按钮功能

文本框创建



按钮创建



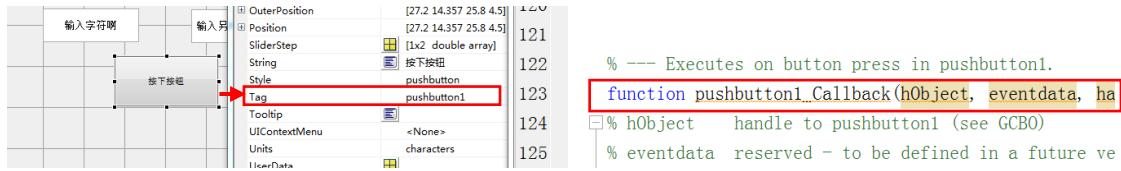


给按钮创建个回调函数

然后就会生成你自定义名称的两个文件



m 文件就是按钮按下后要执行哪些代码, fig 就是 GUI 布局图可以重复打开修改布局



按钮的 Tag 属性就是定义 m 文件的回调函数名, 按钮按下, 程序就是根据这个 Tag 定义的回调函数名去执行回调函数



你看文本框也是这样, 用 Tag 属性来确定回调函数名

下面来做数据传递



我们知道

“输入字符啊” 文本框是 edit1

“输入另外一个字符” 文本框 edit2

按下按钮 是 pushbutton1

这样 tag 参数确定了, 下面就在 m 文件里面操作 tag 参数

get 函数使用 get(handles, 属性)

handles: 就是要获取哪个控件

属性: 就是获取这个控件里面的什么东西

点击按钮之后执行 Callback

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
str = get(handles.edit1, 'String');
set(handles.edit2, 'String', str);
```

handles. 点后面就是要获取哪个控件

String 就是填入的属性, 我要获取这个控件输入的字符

然后将获取到的字符返回给变量 str

set 函数使用 set(handles, 属性, 变量)

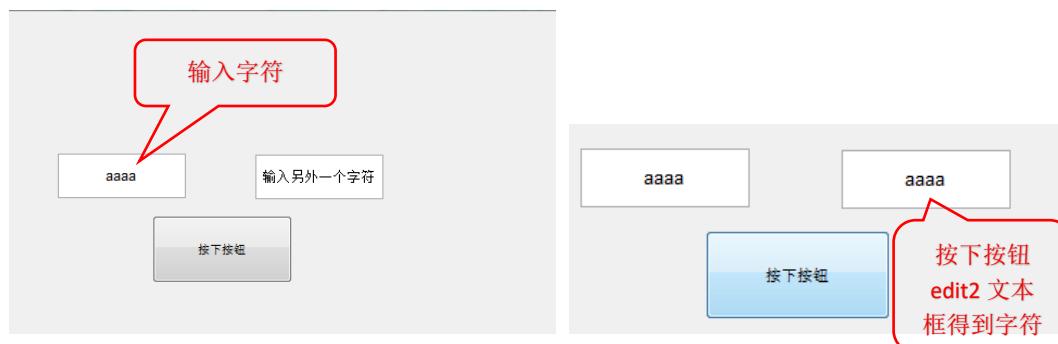
handles: 就是要获取哪个控件

属性: 就是向这个控件里面写哪一类型的数据

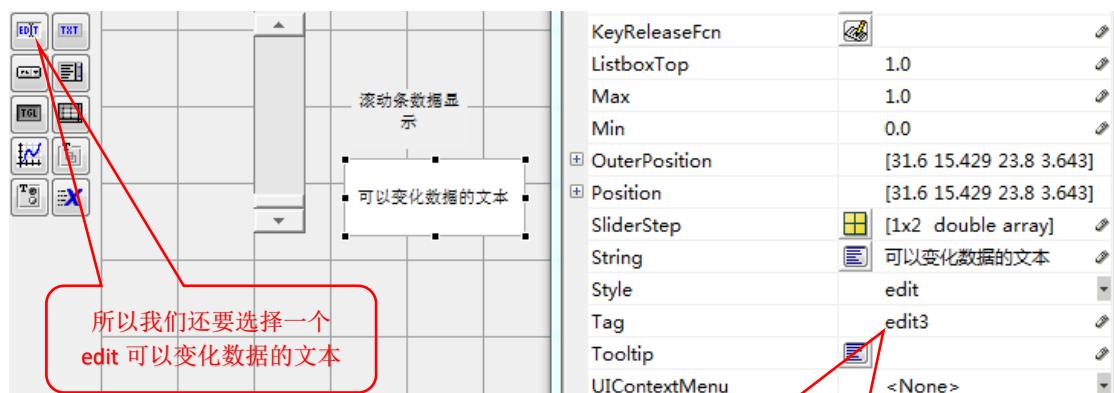
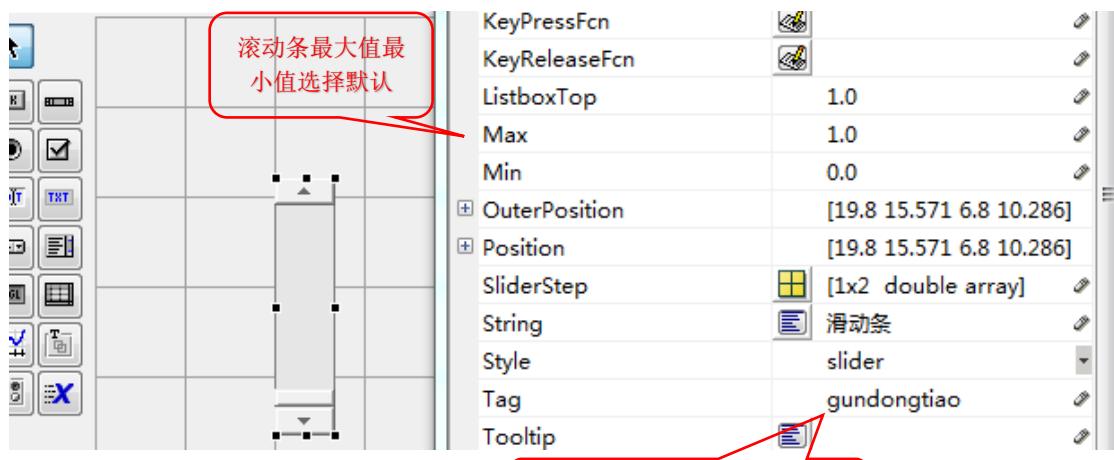
变量: 写入值

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
str = get(handles.edit1, 'String');
set(handles.edit2, 'String', str);
```

将 str 获取的字符写入 edit2 文本



滚动条数据显示功能



用运行的方式可以更新 m 文件里面的回调函数
不然你找不到现在 m 文件的 edit3 函数

这次文本框的 tag 是 edit3 了，注意回调函数名

num2str 函数使用 num2str(参数) 将数值转换成字符串

参数：就是变量或者一个值

% --- Executes on slider movement.

```
|function gundongtiao_Callback(hObject, eventdata, handles)
```

```
|    var = get(handles.gundongtiao, 'value'); % 获取滚动条数据
```

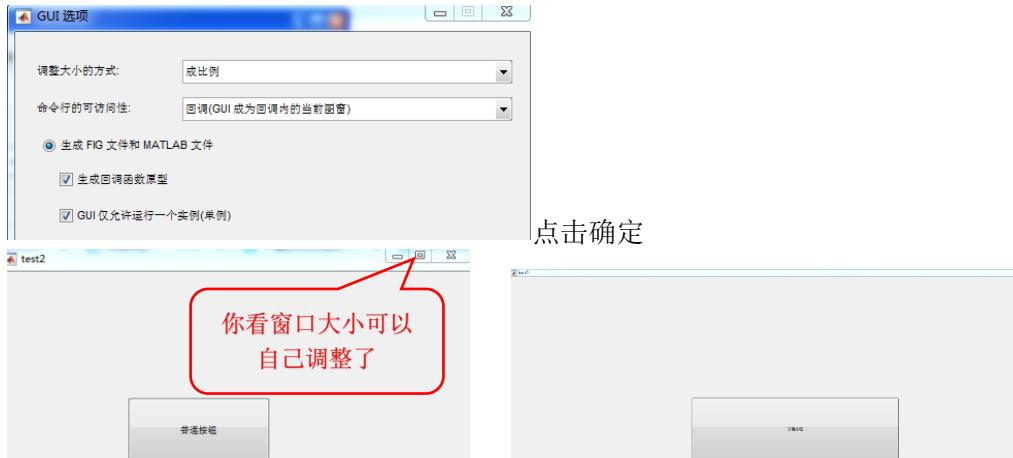
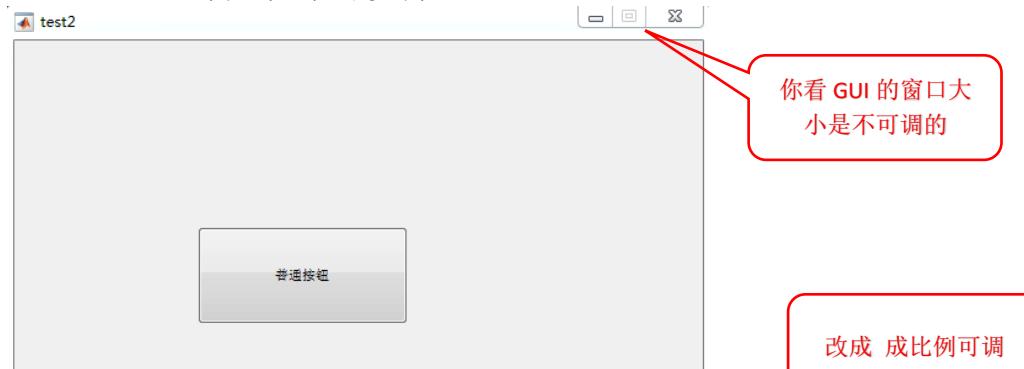
```
|    set(handles.edit3, 'string', num2str(var));
```

滚动条数据获取了，用 set 拿去给 edit3 显示

```
function edit3_Callback(hObject, eventdata, handles) edit3 回调函数
```



让 Matlab GUI 窗口大小可以调整

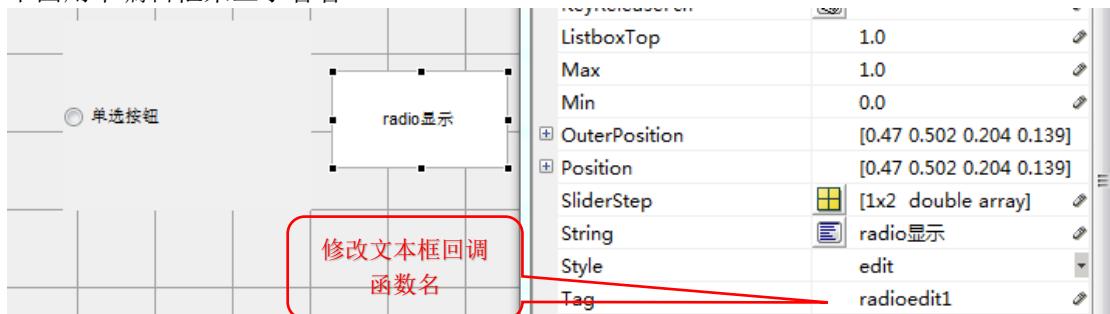


单选按钮实现



这里的 Max 和 Min 不能是其它值，只能是 1 和 0，如果是其它值，单选按钮不显示

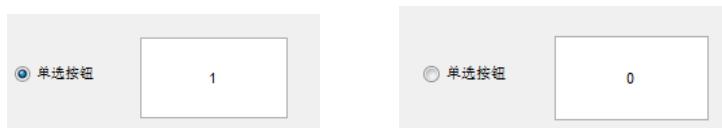
下面用个编辑框来显示看看



```
function radiobutton1_Callback(hObject, eventdata, handles)
var = get(handles.radiobutton1, 'value');
set(handles.radioedit1, 'string', num2str(var));
```

单选按钮点击后会执行该回调函数，
获取单选按钮值

传递给文本框



勾选按钮实现，和单选按钮一样



```

] function checkbox1_Callback(hObject, eventdata, handles)
var = get(handles.checkbox1, 'value');
set(handles.radioedit1, 'string', num2str(var));

```

把获取状态
改为勾选按
钮 tag

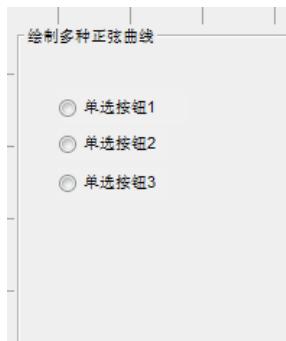


绘制多种曲线

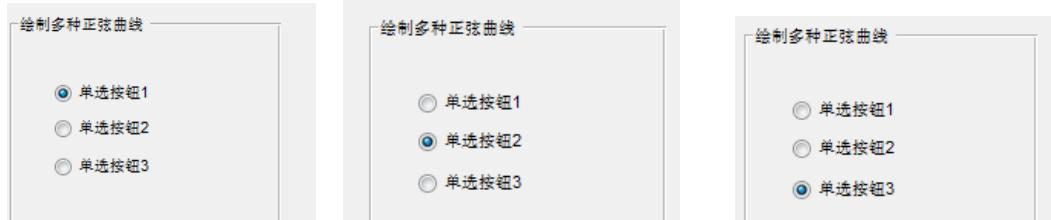
按钮组使用



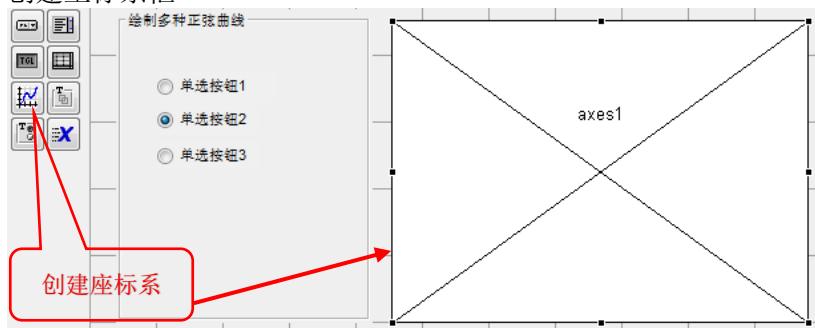
按钮组里面的按钮，同一时间只能选择 1 个

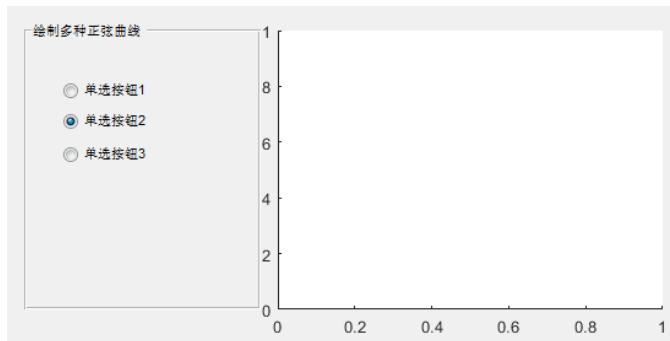


我拖了三个按钮进来，我运行给你看。



创建坐标系框





你看坐标系出来了



```
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
```

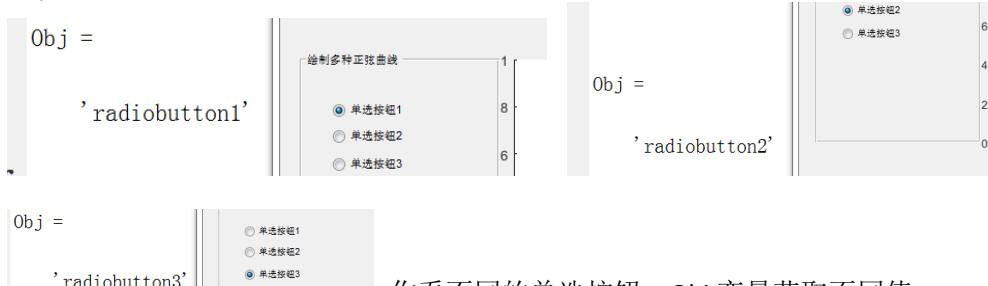
SelectionChangedFcn 回调函数的意思是，如果按钮组的单选按钮发生点击变化，就会执行这个回调函数，下面看看这个函数的好处

```
|function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uibuttongroup1
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Obj = get(eventdata.NewValue, 'Tag'); %获取现在到底选择的哪个单选按钮
Obj
```

获取当前回调函数新值

我要得到这个新值里面的 Tag，这个 Tag 返回的就是我现在点击的是哪一个单选按钮

Obj 变量得到这次点击的单选按钮对象，打印出来



你看不同的单选按钮，Obj 变量获取不同值

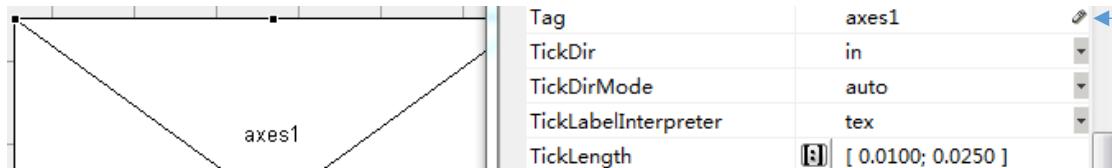
我们根据 obj 这个变量的值来做判断，选择执行哪条正弦曲线

axes 函数使用

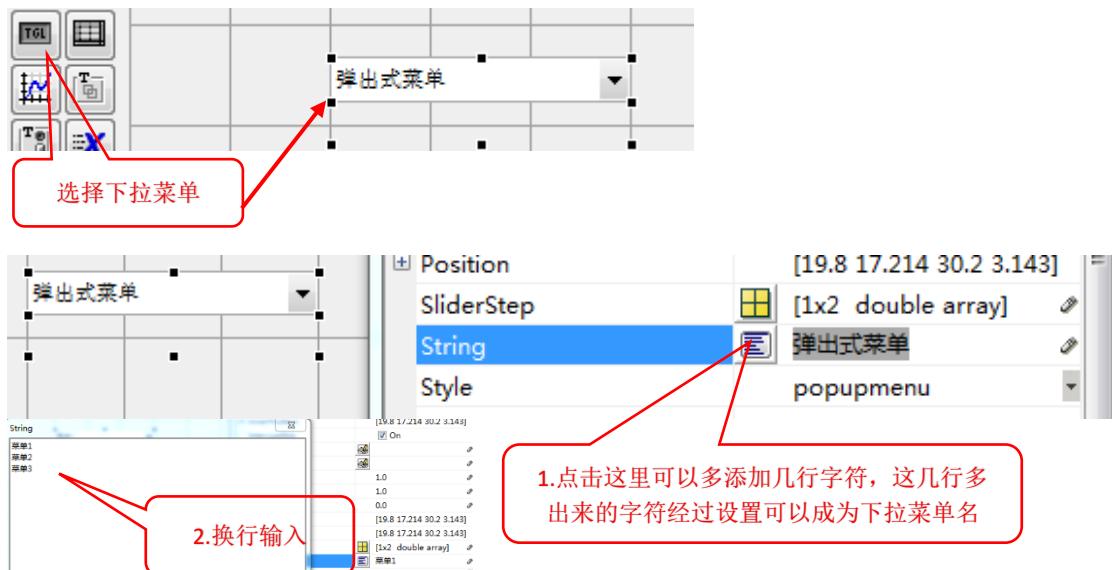
Switch 语法使用

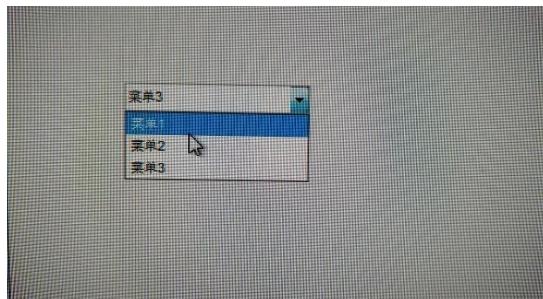
```
% --- Executes when selected object is changed in uibuttongroup1.  
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)  
% hObject    handle to the selected object in uibuttongroup1  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
x=0:0.01:2*pi; %绘制正弦函数，每1个点增加0.01，一直增加到6.48，x有一堆数据了  
Obj = get(eventdata.NewValue,'Tag'); %获取现在到底选择的哪个单选按钮  
%axes(handles.axes1) %将界面的坐标系设置为当前，axes1就是我这个座标的Tag  
switch Obj  
    case 'radiobutton1' %单选按钮1 Tag  
        y = sin(x);  
        plot(x, y); %显示x和y值的曲线  
    case 'radiobutton2' %单选按钮2 Tag  
        y = cos(x);  
        plot(x, y);  
    case 'radiobutton3' %单选按钮3 Tag  
        y = sin(x)+cos(x);  
        plot(x, y);  
end
```

Switch 和 C 语言一样 Obj 得到
哪个对象，就执行哪个 case



下拉菜单实现



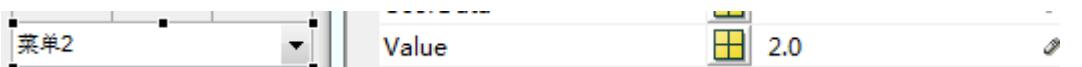


你看，菜单名出来了

但是这些菜单名没有编号，到时候进入回调函数不知道点击的是哪一个菜单
所以下面要用 `value` 属性来设置



Value 值和下拉菜单字符串的每一行一一对应的。



我把 value 值改成 2.0，那么下拉菜单首先显示第 2 行菜单

```
function popupmenu1_Callback(hObject, eventdata, handles)
```

这就是下拉菜单回调函数，名称有 `tag` 设置，这里不多解释。这个回调函数，你选择一次下拉菜单就会重复执行一次。

```
function popupmenu1_Callback(hObject, eventdata, handles)
var = get(handles.popupmenu1, 'value')
```

获取下拉菜单的 value
值，我这里不打分号，
所以 var 的值会打印在
命令行

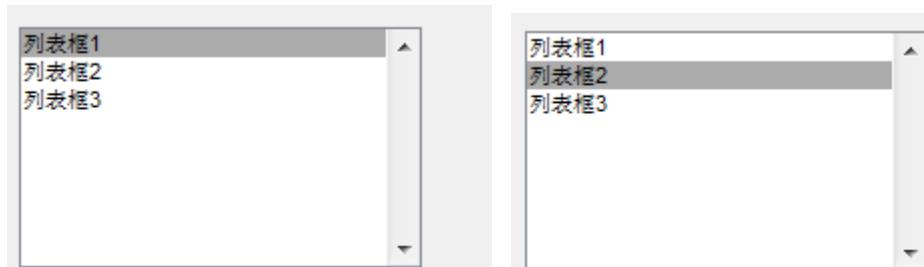
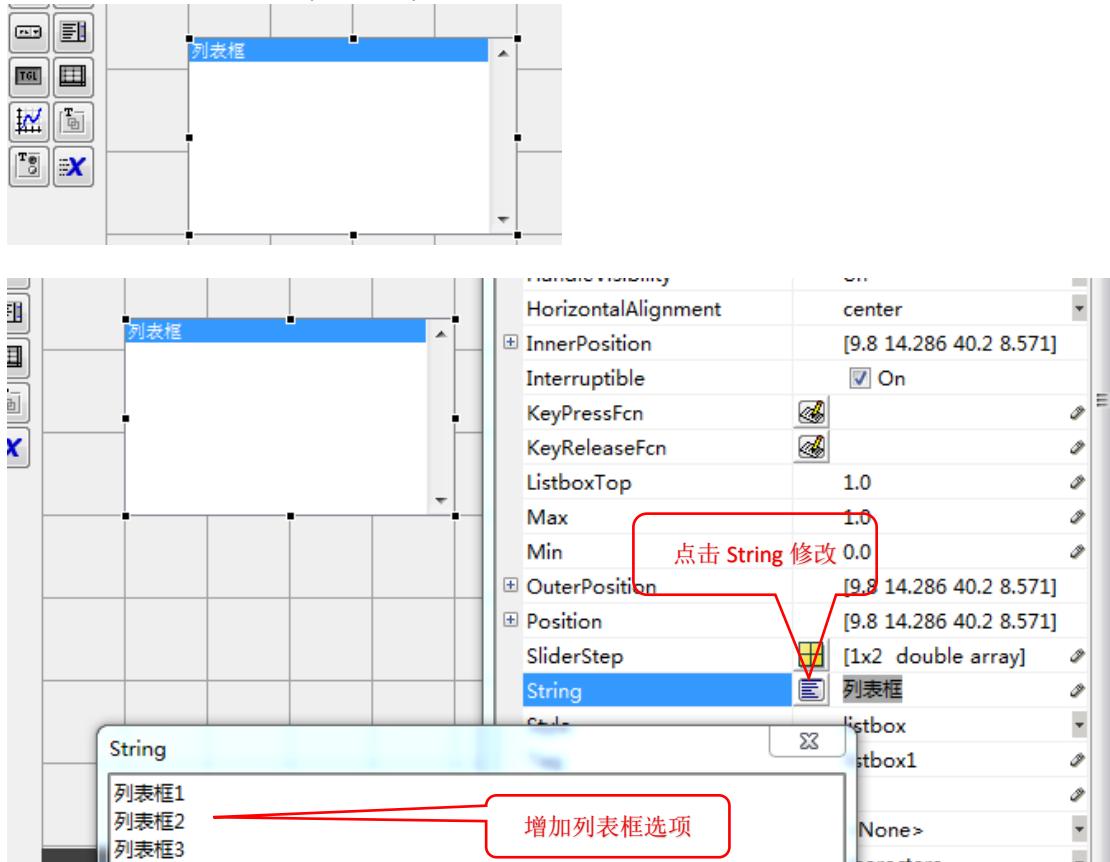
```
var = 1
```

```
var = 2
```

```
var = 3
```

我们发现每行菜单 MATLAB 都给你分配好编号了的。所以你每在菜单下加一行字符串，
MATLAB 就会按着顺序给你字符串添加上对应的编号。
然后你就用 `switch` 去判断执行你要的菜单程序就是了。

参数列表控件 Listbox (列表框)



列表框选项已经做好，怎么获取每个列表框选项的值？

```
function listbox1_Callback(hObject, eventdata, handles)
%这是 listbox1 的回调函数，其实获取值和上一节下拉菜单实现的方式一样。
function listbox1_Callback(hObject, eventdata, handles)
val = get(handles.listbox1, 'value')
```



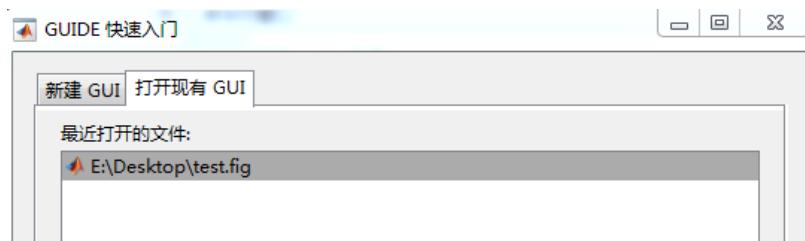
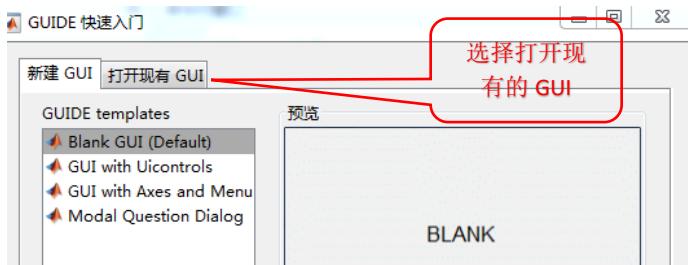
列表每行选项和 value 值也是一一对应了，所以和下拉菜单的方式没有区别，编号 Matlab 已经帮你设置好了。

Matlab GUI 如何重复打开已经布局的界面，修改界面布局

命令行窗口

>> guide

fx >>



你看我前面说的 fig 文件就是布局文件，点击打开它



布局回来了

按钮点击，导入桌面文件操作

[file, path, index] = uigetfile(指定文件扩展名, 导入对话框名称)

指定文件扩展名： 比如 {'.xlsx', 'csv-files (*.xlsx)'}, 那么就是现在所有的 **xlsx** 后缀(**excel**)的文件

导入对话框名称：名称随便取，比如我填入 ‘**xlsxe**’

file: 返回打开的文件名称就放在该变量，这个很重要，后面操作文件就需要用到这个变量
path: 返回文件的路径，如果我们要打开的文件在当前路径下，可以不关心 path，如果我们打开的文件在其它路径，那么就要记住 path 变量

index: 文件打开成功返回 1，如果在对话框中没有找到文件，点击 x 叉叉取消了对话框，那么就返回 0。可以根据返回值决定后面程序怎么操作。

[FileName, PathName, FilterIndex] = uigetfile({'*.xlsx', 'csv-files (*.xlsx)'}, 'xlsxe') % 打开对话框寻找文件
FileName: 自己定义的变量，存放打开的文件名

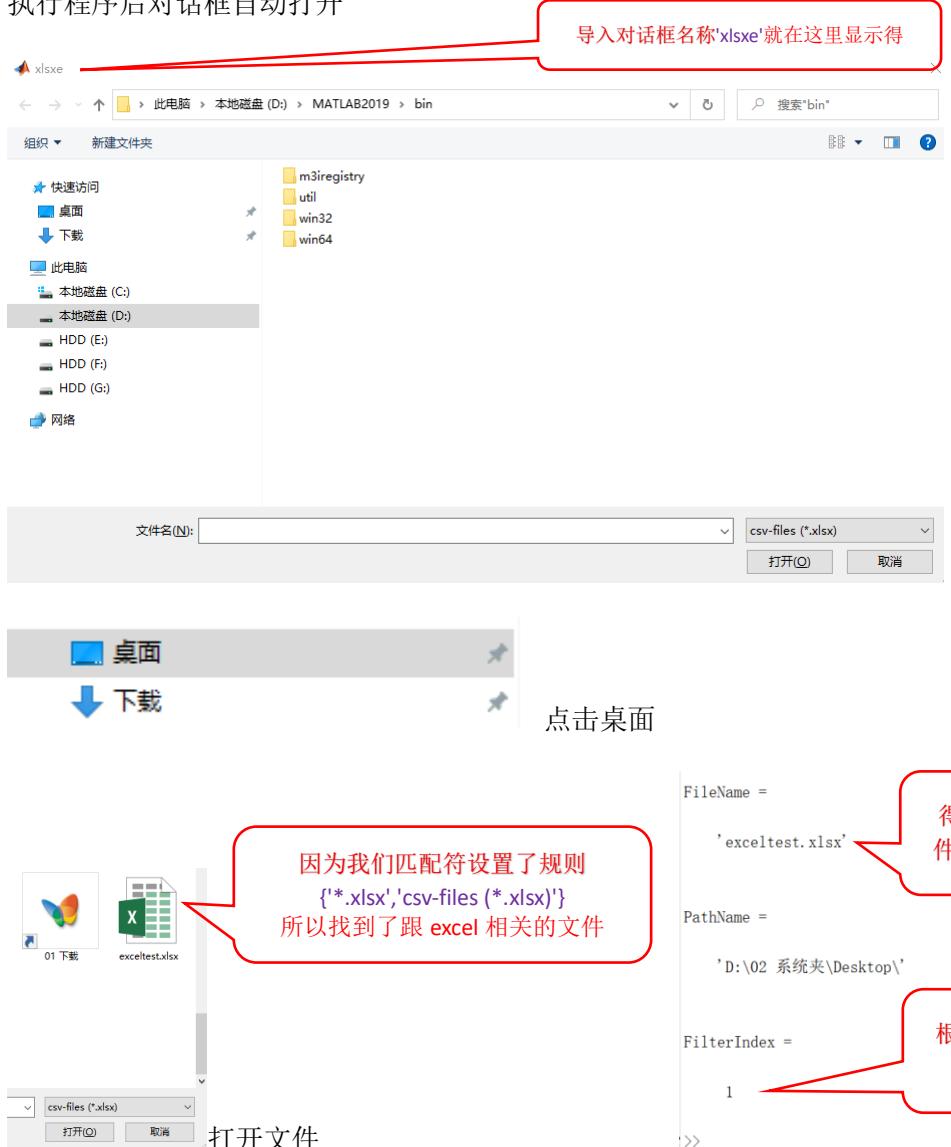
PathName: 自己定义的变量存放文件路径

FilterIndex: 自己定义的变量存放下标， {'*.xlsx', 'csv-files (*.xlsx)'}, 这就是我要寻找文件的后缀匹配参数

{'*.xlsx','csv-files (*.xlsx)'}如果没有这个匹配参数，我的对话框就显示不出 excel 的文件

>> [FileName, PathName, FilterIndex]=uigetfile({'*.xlsx', 'csv-files (*.xlsx)'}, 'xlsxe')

执行程序后对话框自动打开



>> data = xlsread(FileName, 'Sheet1') 可以直接将 FileName 变量指向的 excel 文件读出来，`xlsread` 函数后面章节有介绍。

```
>> data=xlsread(FileName,'Sheet1')  
错误使用 xlsread (line 136)  
XLSREAD 无法打开文件 'exceltest.xlsx'。  
未找到文件 'D:\MATLAB2019\bin\exceltest.xlsx'。出现报错，这是因为我们在命令行模式下操作，而命令行使用的是 bin 目录下的程序。所以找不到 excel
```

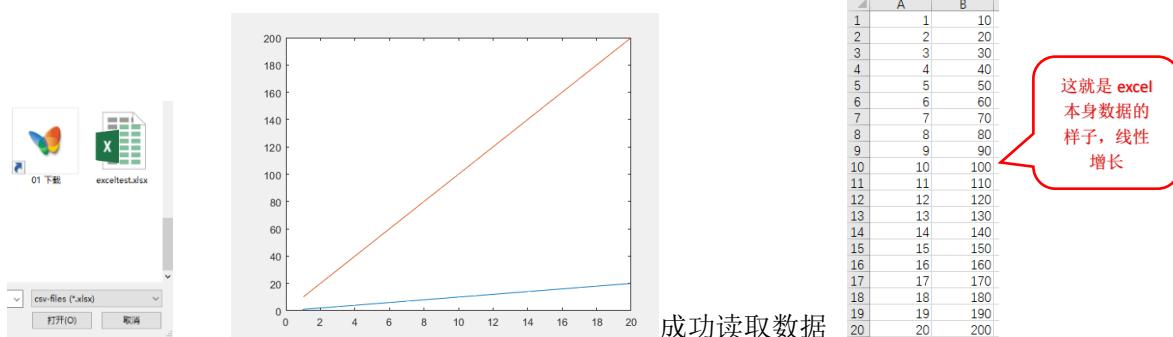
```

IOCTL.m x +
function IOCTL()
[FileName, PathName, FilterIndex]=uigetfile( {'*.xlsx', 'csv-files (*.xlsx)'}, 'xlsx');
data = xlsread(FileName, 'Sheet1');
plot(data);
pause(10);
end

```

我直接写个 m 文件，在桌面下运行 MATLAB，那么 `xlsread` 就在桌面读取，excel 文件在桌面，所以没有问题。

运行程序



寻找 CSV 和 excel 注意事项

```
[FileName, PathName, FilterIndex]=uigetfile( {'*.csv'}, 'csvload')
```

符号之间不能有空格，不然也找不到指定文件

```
[FileName, PathName, FilterIndex]=uigetfile( {'*.xlsx'}, 'csvload')
```

这是寻找 excel 文件

如果不知道文件路径，就用字符串拼接来处理，

`M = csvread([PathName, FileName]);` %路径+文件名，这种字符串拼接打开csv文件比较保险

指定路径保存生成的 csv 文件

`[file, path, indx] = uiputfile('test.csv');` %执行之后会生成个对话框，选择保存csv文件的路径。路径将以字符串形式返回给path变量

`file:` 返回保存的文件名称，我这里是 `test.csv`

`path:` 返回保存文件的指定路径字符串。

```
[file, path, indx] = uiputfile('test.csv');
```

创建一个 csv 文件，指定保存的路径，这句执行后不会保存，只会给出文件名和路径

`A = [10, 20, 30, 40, 50];`

`STR = [path, file];`

`csvwrite(STR, A);`

`n = 0;`

`while n<=5`

`pause(1);`

`end`

将路径和文件名拼接起来
D:\02 系统夹\Desktop\test.csv

还是要用写 CSV 函数才能生成文件，只是这个文件的路径在 STR 变量里面已经确定好了

Matlab 读取 excel 数据

A	B
10	11
20	12
30	30
40	45
50	60

准备一份 excel 数据，我把 A 当作 x， B 当作 y



这里选择导入数据，点击你的 excel 文件

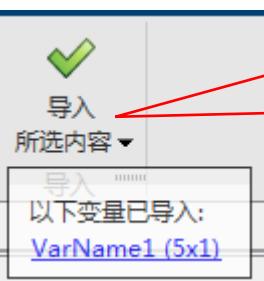
VarName1	VarName2
10	11
20	12
30	30
40	45
50	60

进来之后发现你只能一个一个的数据点去选

VarName1	VarName2
10	11
20	12
30	30
40	45
50	60

你看，你可以一列一列的选择了

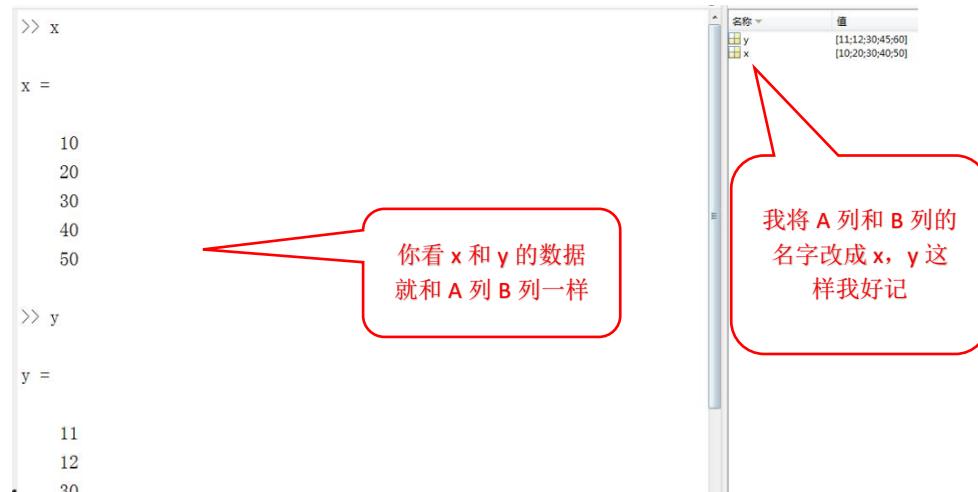
选择列向量



选择 A 列，点击导入，用(导入数据)选项

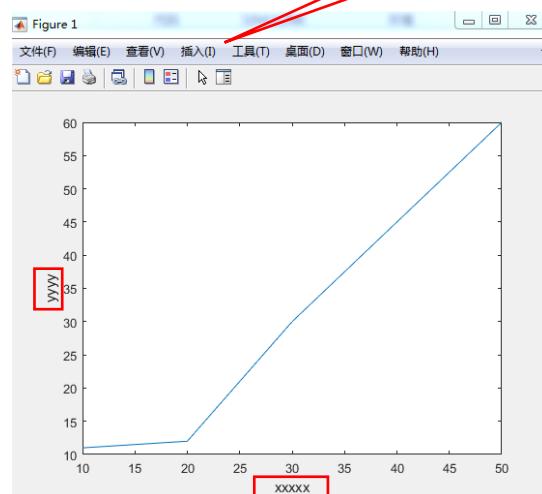
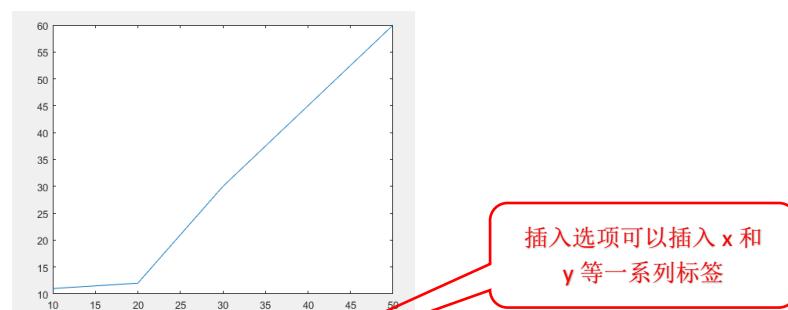


B 列导入也是一样的方法



plot 曲线绘图函数使用, plot(横坐标变量, 纵坐标变量)

`>> plot(x, y)`



MATLAB 语法规详解

for 循环使用

```
array = [-2 10 6];
for k=array
    disp(k)
end
```

定义数组变量

for 循环是根据数组里面的数据个数来决定循环几次，而不是像 C 语言那样根据数据值大小来循环，然后把当前循环的数据赋值给 k 变量

```
>> test
-2
10
6
```

运行正确，取出每一次循环的值给 k。

for 循环也可以用矩阵来循环

```
array = [1 2 3 4
         5 6 7 8
         9 10 11 12];
```

MATLAB 数组里面换行就是矩阵了

```
]for k=array
    disp(k)
end
```

每次循环会提取数组的一列下所有数据赋值给 K，有多少列循环多少次

```
>> test
1
5
9
2
6
10
3
7
11
```

所以循环 4 列，每循环一次，k 得到一列所有数据

矩阵向量循环相加案例

一列代表 1 个向量

```
array=[3 4 2 8
      7 0 5 1
      9 7 6 5];
s=0;
for m=array
    s=s+m;
end
```

循环次数	1	2	3	4
循环变量 m	3	4	2	8
变量 s 的值	3	7	9	17
	7	7	12	13
	9	16	22	27

第 1 次循环, m 得到第 1 列 [3, 7, 9] 向量

第 1 次循环 $s=0 + [3 7 9]$ 所以 $s = [3 7 9]$

第 2 次循环 $s=[3 7 9] + [4 0 7]$ 这就是两个矩阵列相加, 所以 s 得到 [7 7 16]

第 3 次循环以此类推, 看表就知道了

```
array = [3 4 2 8
         7 0 5 1
         9 7 6 5];
s = 0;
for k=array
    s = s + k;
end
disp(s) % 显示 s 最后值
```

```
>> test
17
13
27
```

这就是结果,
和上面表格循
环第 4 次一样

while 循环使用

```
n = 0;
while n <= 5
    n= n+1;
    disp(n)
end
```

while 循环和 C 语言差不多

如果 $n < 5$ 返回真,
while 继续循环,
如果 $n > 5$ 返回假,
跳出 while 循环

```
>> test
1
2
3
4
5
6
```

你看循环第 6 次
就跳出循环

If , elseif, else 的使用, 类似 C 语言, if , else if , else

```
a = 0;          a 是 0
if a > 0
    disp('a 数据 > 0')
elseif a == 0
    disp('a 数据 == 0')
elseif a < 0
    disp('a 数据 < 0')
else
    disp('a 数据 无法判断')
end
```

```
>> test
a 数据 == 0
```

```
a = 2;          a 是 2
if a > 0
    disp('a 数据 > 0') ←
elseif a == 0
    disp('a 数据 == 0')
elseif a < 0
    disp('a 数据 < 0')
else
    disp('a 数据 无法判断')
end
```

```
>> test
a 数据 > 0
```

```
a = -1;         a 是 -1
if a > 0
    disp('a 数据 > 0')
elseif a == 0
    disp('a 数据 == 0')
elseif a < 0 ←
    disp('a 数据 < 0')
else
    disp('a 数据 无法判断')
end
```

```
>> test
a 数据 < 0
```

```
a = []:
if a > 0
    disp('a 数据 > 0')
elseif a == 0
    disp('a 数据 == 0')
elseif a < 0
    disp('a 数据 < 0')
else
    disp('a 数据 无法判断')
end
```

```
>> test
a 数据 无法判断
```

所以和 c 语言一样, 但是 Matlab 的 else if 是用 elseif 连写表示

switch case end 使用

```
n=10; switch n
```

n 为 10, 就执行 case 10 的分支

```
    case 5  
        disp('n = 5')  
    case 10  
        disp('n = 10')  
    otherwise
```

```
        disp('other value')
```

```
end
```

>> test

n = 10

```
n=5; switch n
```

n 为 5, 就执行 case 5 的分支

```
    case 5  
        disp('n = 5')  
    case 10  
        disp('n = 10')  
    otherwise
```

```
        disp('other value')
```

```
end
```

>> test

n = 5

```
n=1; switch n
```

n 为 1, 不符合 case 分支就执行 otherwise

```
    case 5  
        disp('n = 5')  
    case 10  
        disp('n = 10')  
    otherwise
```

```
        disp('other value')
```

```
end
```

>> test

other value

以上和 C 语言 switch 一模一样

MATLAB 的 switch 比 C 语言 switch 多一些功能的地方是, case 可以用单元数组
month=2;

```
switch month
```

```
    case {1, 2, 3}  
        disp('数组 = 1 2 3')
```

单元数组就是 month 变量里面的值,
只要符合 case 数组里面的任何一个
值, 该 case 分支就执行

```
    case {7, 8, 9}  
        disp('数组 = 7 8 9')
```

```
    otherwise
```

```
        disp('other value')
```

>> test

```
end
```

数组 = 1 2 3

```

month=9;
switch month
    case {1, 2, 3}
        disp('数组 = 1 2 3')
    case {7, 8, 9}
        disp('数组 = 7 8 9')
    otherwise
        disp('other value')
end

```

>> test
数组 = 7 8 9

我修改 month 为 9，那么就执行 case {7,8,9} 这分支单元数组

try catch end 使用，就是用来判断函数程序是否执行成功

det 函数使用 **det(参数)**
参数：填入矩阵变量

```

A=[5 5;6 6]; %这是一个2x2的矩阵
try
    ret = det(A);
    disp('矩阵A是2x2的矩阵')
catch
    disp('矩阵A不是一个2x2的矩阵')
end

```

det()函数执行成功，就执行 try 分支，最后跳出

>> test
矩阵A是2x2的矩阵

A=[5 5 6]; %这是数组，不是矩阵

```

try
    ret = det(A);
    disp('矩阵A是2x2的矩阵')
catch
    disp('矩阵A不是一个2x2的矩阵')
end

```

try 先执行，det()函数执行失败，然后再去执行 catch 分支

>> test
矩阵A不是一个2x2的矩阵

break 和 **continue** 在 **for** 和 **while** 使用

```

n = 0;
while n <5
    n= n + 1;
    disp(n);
    break;
end

```

和 C 语言一样直接
跳出循环

>> test

1

因为直接跳出循环，所以 n 只加了 1 次

continue 功能和 C 语言一样，自行百度

直接构造数组

```
>> x = [1 2 3 4 5]
```

```
x =
```

```
1 2 3 4 5
```

这就是创建数组
数组类型是 double 双精度

```
>> x = [1, 2, 3, 4, 5]
```

```
x =
```

```
1 2 3 4 5 加逗号也是一样的，看自己习惯
```

增量法构造数组

变量 = 开始数:结束数 数组每一个元素增加 1

```
>> x=10:15
```

创建 10 到 15 的数组

```
x =
```

```
10 11 12 13 14 15
```

变量 = 开始数 : 步长 : 结束数

```
>> x=10:5:50
```

这个步长 5，就是数组
每一个元素增加 5

```
x =
```

```
10 15 20 25 30 35 40 45 50
```

```
>> x=3:.2:3.8
```

这种步长为 .2 的写法
就是每个元素增加 0.2

```
x =
```

```
3.0000 3.2000 3.4000 3.6000 3.8000
```

```
>> x=9:-1:1
```

递减方式增量

```
x =
```

```
9 8 7 6 5 4 3 2 1
```

linspace 函数用法 变量=linspace(开始数, 结束数, 开始结束之间取多少个数)

```
>> x=linspace(0, 10, 5)
```

0 到 10 之间取
5 个数

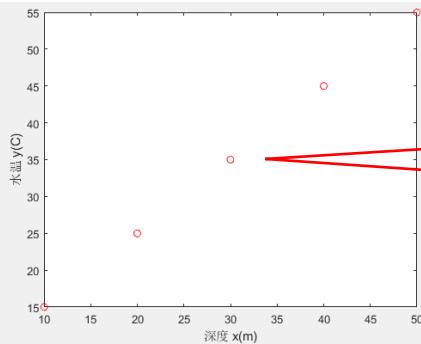
```
x =
```

```
0 2.5000 5.0000 7.5000 10.0000
```

创建矩阵

grid on 和 hold on 函数用法

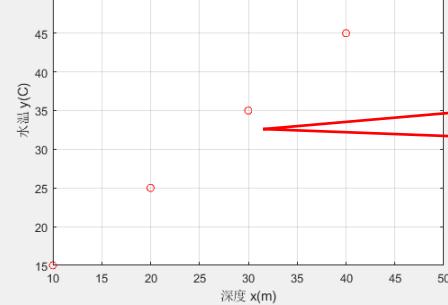
```
x = [10 20 30 40 50];  
y = [15 25 35 45 55];  
plot(x,y, 'ro');  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');
```



这是没有网格
显示的坐标图

加入 grid on

```
x = [10 20 30 40 50];  
y = [15 25 35 45 55];  
plot(x,y, 'ro');  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on
```

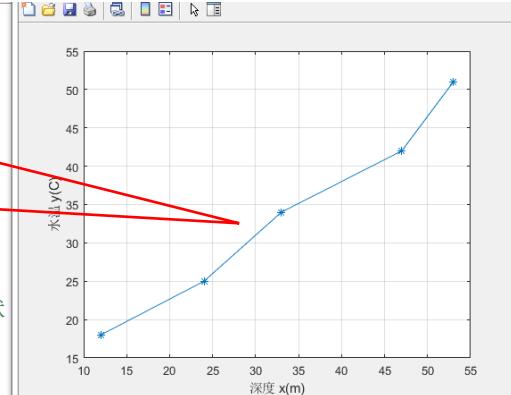


这下坐标就有
网格了

hold on 是执行两次 plot 显示两种曲线的情况下使用的

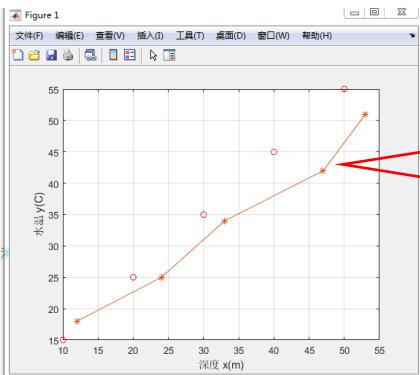
```
x = [10 20 30 40 50];  
y = [15 25 35 45 55];  
plot(x,y, 'ro');  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on  
  
x2 = [12 24 33 47 53];  
y2 = [18 25 34 42 51];  
plot(x2,y2, '-*');%用另外一种符号-*区分开上面座标ro形状  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on
```

x2, y2 的坐标曲线有
了，但是上一次 x 和 y
的坐标怎么不见了？



这是因为第二次 x2,y2 的 plot 覆盖了第一次 x, y 的 plot 图
所以需要加入 hold on 将第一次 x, y 保持住

```
x = [10 20 30 40 50];  
y = [15 25 35 45 55];  
plot(x,y, 'ro');  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on  
hold on  
x2 = [12 24 33 47 53];  
y2 = [18 25 34 42 51];  
plot(x2,y2, '-*');%用另外一种符号-*区分开上面座标ro形状  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on
```



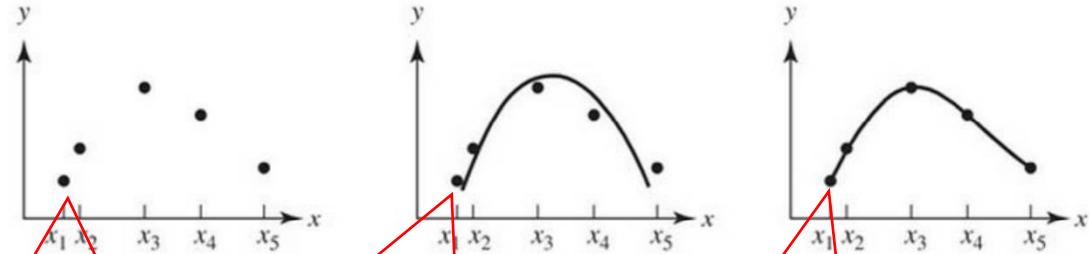
你看加入 hold on 两
条曲线保存住了

Matlab 一维插值算法函数使用

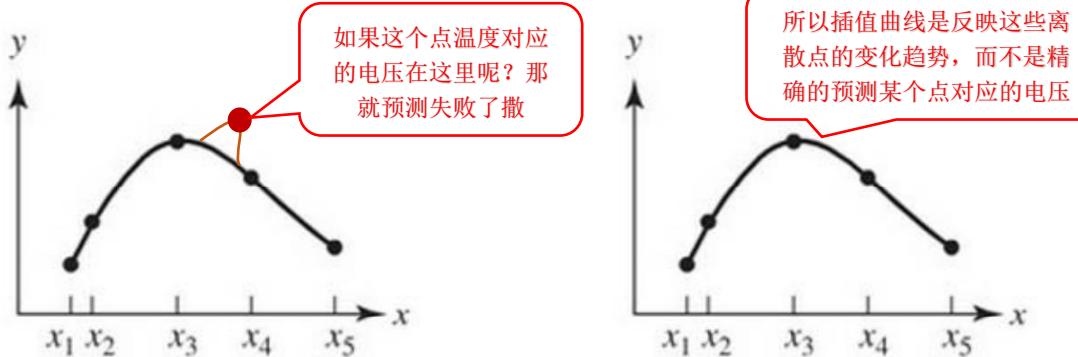
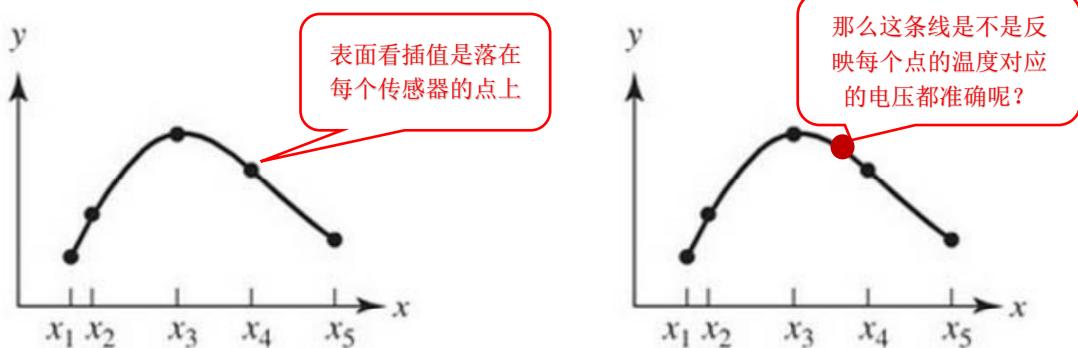
比如 x_1 是温度传感器的温度值, y_1 是温度传感器输出的电压值, x_2, y_2 是第 2 次传感器采集的温度值和输出的电压值, 那么传感器输出的电压如下面曲线这样。

输入值	x_1	x_2	x_3	x_n
输出值	y_1	y_2	y_3	y_n

■ 数据插值 (interpolation)



是不是插值就很完美呢? 就可以完全预测传感器任何温度下的电压值呢? 其实不是的



Interp1 函数使用

ret = Interp1(原始数据 x 变量, 原始数据 y 变量, 新采集的 x 数据, method)

ret: 返回新采集 x 数据对应的新 y 值

method: linear 线性插值(默认插值方法)

nearest 最邻近插值

spline 三次样条插值

在某海域，测得海洋不同深度处的海水温度如下表

深度 x (m)	466	714	950	1422	1634
水温 y (C°)	7.04	4.28	3.40	2.54	2.13



600 米深水温是多少？

800 米深水温是多少？

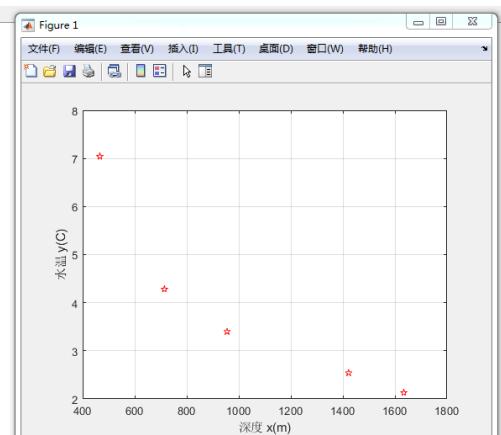
1200 米深水温是多少？

我就只有经过现在采集的已知 5 个数据点 x, y 来推算这三个深度的水温

设置海水从 400 米深开始->每增加深度间隔 20 米->一直加到 1700 米，这一段连续数据的温度

```
x = [466 714 955 1422 1634]; %这就是已知表中的x数据点  
y = [7.04 4.28 3.40 2.54 2.13];%这就是已知表中的y数据点
```

```
plot(x, y, 'rp') %rp表示数据点用圆圈表示  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on %打开坐标网格
```



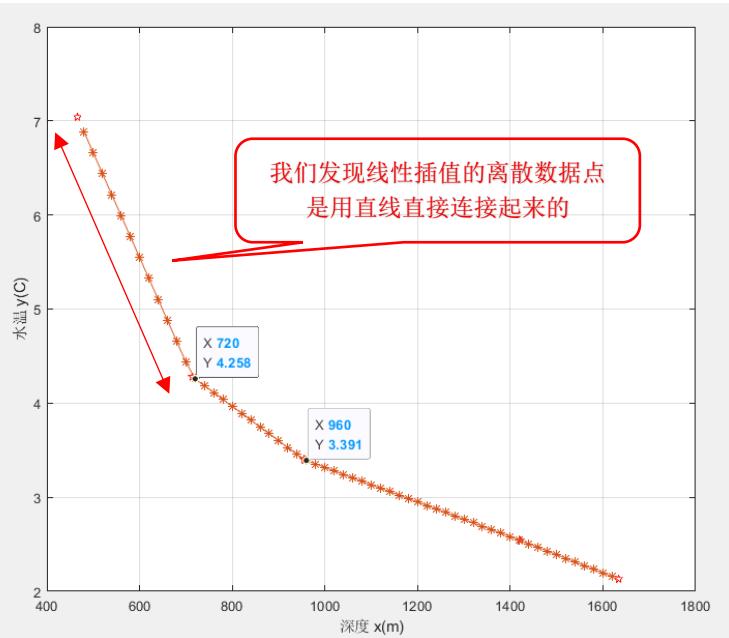
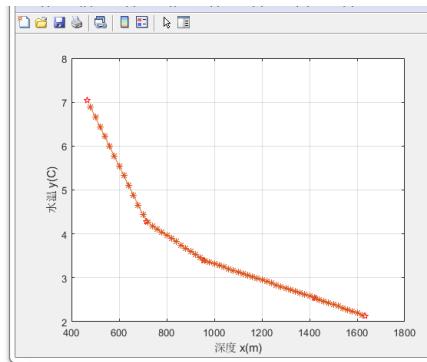
有了已知 xy 离散数据点的数据，下面来用线性插值算法，看能不能算出趋势。

```
x = [466 714 955 1422 1634]; %这就是已知表中的x数据点
y = [7.04 4.28 3.40 2.54 2.13];%这就是已知表中的y数据点
```

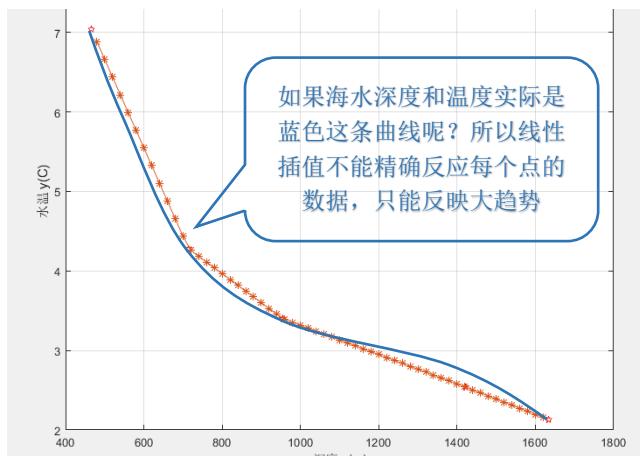
```
plot(x, y, 'rp') %rp表示数据点用圆圈表示
xlabel('深度 x(m)');
ylabel('水温 y(℃)');
grid on %打开坐标网格
```

```
xq=400:20:1700; %从400米没增加间隔20米，一直到1700米，xq就是个数组了
```

```
yq = interp1(x, y, xq, 'linear');
%填入已知xy，用线性插值linear计算xy关系，再填入你想要的深度xq得到温度yq
hold on %当前轴及图像保持而不被刷新
plot(xq, yq, '-*');% 符合-*意思是 xq yq用*号绘制
```



这种其实并不能说明海水各点深度与温度的关系



下面用邻近插值算法试试

nearest 邻近插值

y = [7.04 4.28 3.40 2.54 2.13]; %这就是已知表中的y数据点

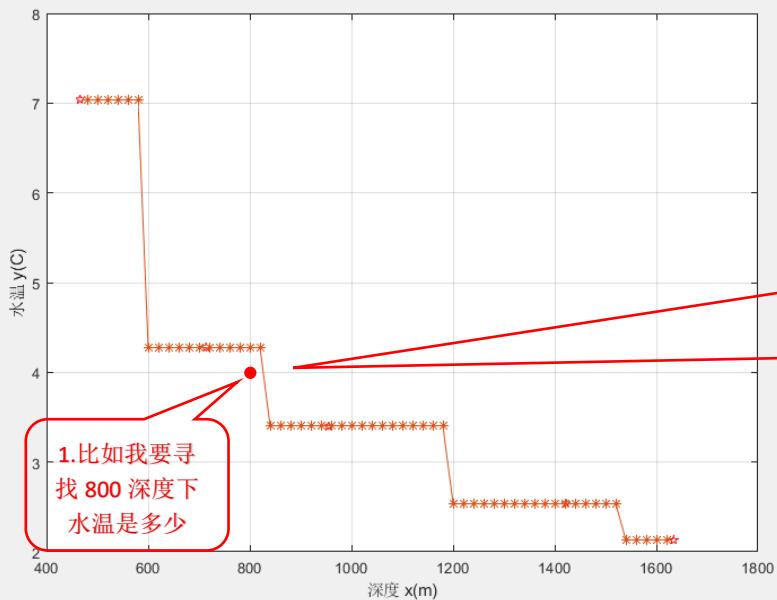
```
plot(x, y, 'rp') %rp表示数据点用圆圈表示  
xlabel('深度 x(m)');  
ylabel('水温 y(C)');  
grid on %打开坐标网格
```

```
xq=400:20:1700; %从400米没增加间隔20米, 一直到1700米 xq就是个数组了  
yq = interp1(x, y, xq, 'nearest'); %填入已知xy, 用线性插值linear计算xy关系, 再填入你想要的深度xq得到温度yq  
hold on %当前轴及图像保持而不被刷新
```

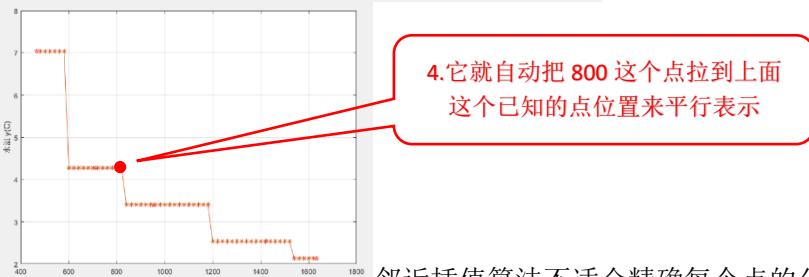
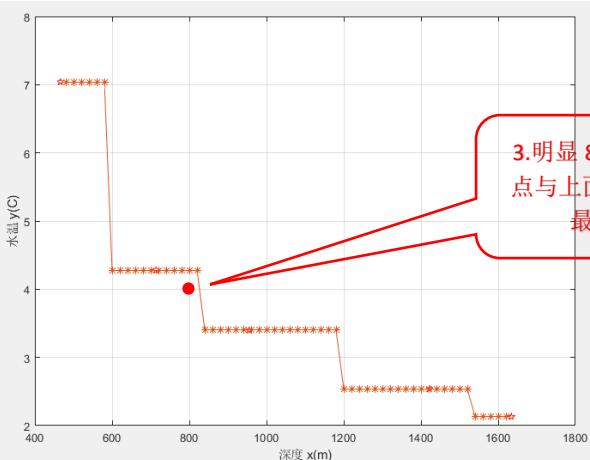
选择邻插值

你会发现更
不靠谱

其实邻近插值想说明的意思如下:



2.它就会在 800 这个深度的水温位置, 上下判断 800 这个点与已知的 xy 五个点哪个最近

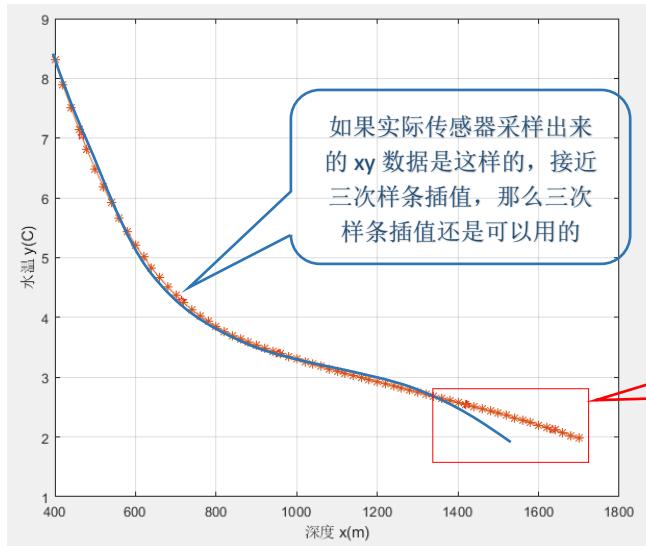
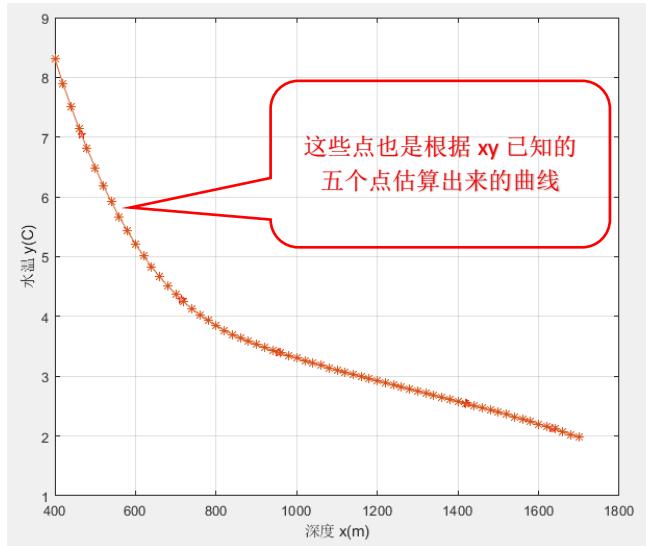
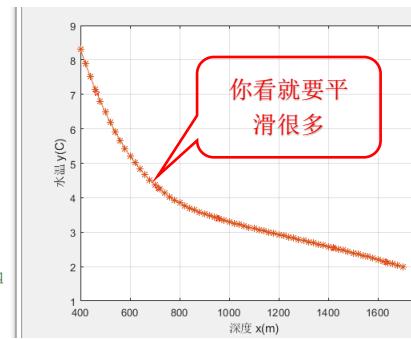


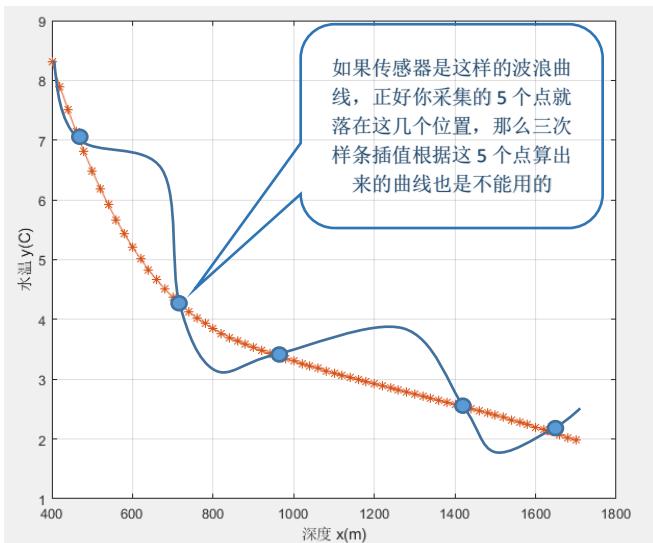
邻近插值算法不适合精确每个点的分析, 但是可以用到一些区间很大, 做区域性判断的程序上。

三次样条插值(用得最多，接近实际情况的算法)

spline 三次样条插值

```
plot(x, y, 'rp') %rp表示数据点用圆圈表示  
 xlabel('深度 x(m)');  
 ylabel('水温 y(C)');  
 grid on %打开坐标网格  
  
xq=400:20:1700; %从400米没增加间隔20米，一直到1700米, xq就是个数组了  
yq = interp1(x, y, xq, 'spline'); //选择 spline 三次  
%填入已知xy，用线性插值linear计算xy关系，再填入你想要的深度xq得到温度yq  
hold on %当前轴及图像保持而不被刷新  
plot(xa, ya, '-*'); %符合-*意思是 xa ya用*号绘制
```





所以工程开发，都是先采集传感器的数据做成曲线，然后再去找各种曲线算法，看哪种曲线符合传感器采集出来的曲线，那么就选定这种曲线算法来做非线性补偿。

拉格朗日插值法算法实现

$$y_q = \sum_{k=1}^n l_k(x_q) \cdot y_k$$

$$l_k(x_q) = \prod_{\substack{j=1 \\ j \neq k}}^n \frac{(x_q - x_j)}{(x_k - x_j)} = \frac{(x_q - x_1)(x_q - x_2) \cdots (x_q - x_{k-1})(x_q - x_{k+1}) \cdots (x_q - x_n)}{(x_k - x_1)(x_k - x_2) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}, \quad k = 1, 2, \dots, n$$

$[x_1, y_1], [x_2, y_2], [x_3, y_3] \dots [x_n, y_n]$ 这些都是我传感器采集好的离散数据点
 x_q 就是我要随时写入的传感器数据

y_q 就是我写入 x_q 之后得到传感器输出的结果 y_q

$$y_q = \sum_{k=1}^n l_k(x_q) \cdot y_k$$

y_k 就是已知离散数据的y座标 比如 $[x_1, y_1], [x_2, y_2], [x_3, y_3] \dots [x_n, y_n]$

$y_1, y_2, y_3 \dots y_n$ 现在我们要求 l_k

如何得到 $l_k(x_q)$ ？就是求我随便输入 x_q 情况下 l_k 的值 \prod 表示连乘 类似 \sum 连续求和

$$l_k(x_q) = \prod_{\substack{j=1 \\ j \neq k}}^n \frac{(x_q - x_j)}{(x_k - x_j)}$$

表示j从1到n
但是 $j \neq k$ 就是j不能等于n，那么就只能计算到-1

l_k 就是这样一个式子做连乘

x_k x 就是已知座标点 $[x_1, y_1]..[x_n, y_n]$ 的其中1个值 k 就是选择 $[x_1, y_1]..[x_n, y_n]$ 里的第几个x

x_j 就是连乘里面每一次循环在已知 $[x_1, y_1]..[x_n, y_n]$ 里面取的x值

下面代码来实现

深度 x (m)	466	714	950	1422	1634
水温 y (C°)	7.04	4.28	3.40	2.54	2.13

还是求深度与水温的关系

```
x = [466 714 955 1422 1634]; %这就是已知表中的x数据点
y = [7.04 4.28 3.40 2.54 2.13];%这就是已知表中的y数据点
```

```
xq = 500;%我现在想求500米下的水温
```

```
n = 5; %采集到的离散数据点个数
```

le就是 l_k $\boxed{1 \ 1 \ 1 \ 1 \ 1}$

```
le=ones(1, n); %1x5的一维矩阵, 这下le就要5个元素了初始为1
```

```
]for k = 1:n %循环1到5次
]    for j = 1:n %这是连乘的循环
        if j ~= k %当j不等于k的时候才执行连乘
            le(k) = le(k)*(xq-x(j))/(x(k)-x(j));
        end
    end
end
sum = le(1)*y(1)+le(2)*y(2)+le(3)*y(3)+le(4)*y(4)+le(5)*y(5);
%将连乘之后的每一个结果  $l_k(x_q)$  与y值相乘然后求和
yq = sum;
```

$$\sum_{k=1}^n l_k(x_q) \cdot y_k$$

$$l_k(x_q) = \prod_{\substack{j=1 \\ j \neq k}}^n \frac{(x_q - x_j)}{(x_k - x_j)}$$

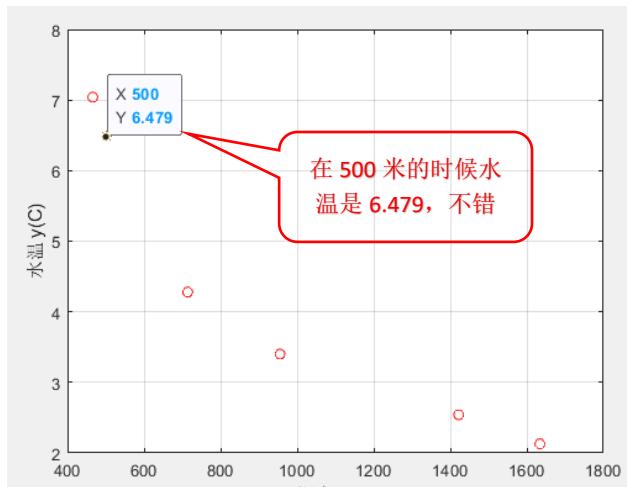
这里不管你怎么循环
 x_q 都是我传入进来的
500 深度, 值不变

$$\sum_{k=1}^n l_k(x_q) \cdot y_k$$

```
plot(x, y, 'ro');
xlabel('深度 x(m)');
ylabel('水温 y(C)');
grid on
hold on

plot(xq, yq, '-*');
xlabel('深度 x(m)');
ylabel('水温 y(C)');
grid on
```

看看计算结果



如果你对你的算法有疑问，可以找个标准拉格朗日插值法的软件进行对比

下面把每个深度点的温度用拉格朗日插值法做出来，做成曲线

```
function 函数定义方法
function ret = test(x1,x2,xn....)
    /*写入函数内容*/
end
```

关键字function和end组合包括完函数体

test是函数名，必须和m文件名字一样，这就说明了1个m文件只能有一个函数

x1 , x2 , xn....是函数传入的形参，可以传入多个，可以自定义形参名称

ret 是函数返回值，可以自定义返回值名称

```
|function yq = test(x, y, xq)
n = length(x); %采集到的离散数据点个数
1e=ones(1, n); %1x5的一维矩阵，这下1e就要5个元素了初始为1
|for k = 1:n %循环1到5次
|    for j = 1:n %这是连乘的循环
|        if j ~= k %当j不等于k的时候才执行连乘
|            1e(k) = 1e(k)*(xq-x(j))/(x(k)-x(j));
|        end
|    end
end
sum = 1e(1)*y(1)+1e(2)*y(2)+1e(3)*y(3)+1e(4)*y(4)+1e(5)*y(5);
%将连乘之后的每一个结果1k (xq) 与y值相乘然后求和
yq = sum;
end
```

我们把每一个 **xq** 计算出对应 **yq** 点的算法封装成函数

函数返回值变量不能与外部变量重名

```
x = [466 714 955 1422 1634]; %这就是已知表中的x数据点
```

```
y = [7.04 4.28 3.40 2.54 2.13];%这就是已知表中的y数据点
```

```
plot(x, y, 'ro');
```

```
xlabel('深度 x(m)');
```

```
ylabel('水温 y(℃)');
```

```
grid on
```

```
hold on
```

```
xq = 400:20:1700; %查询400到1700深度的整个温度曲线
```

我要查询 400~1700 深度
的所有温度点

1700 个深度点，循环 1700-
400 次，自己算

```
for m=1:length(xq)
```

```
yyq(m) = test(x, y, xq(m));
```

%因为函数返回值变量是yyq

传入原始数据和要自己要
查询的深度给自定义的拉
格朗日插值函数

%所以外部变量不能和函数返回值变量重名

%这里用yyq替代yq

```
end
```

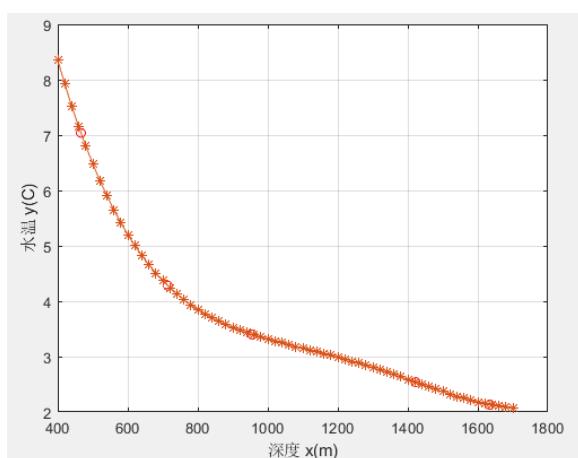
```
plot(xq, yyq, '-*');
```

得到每个点深度对应的温
度输出给 yyq 数组保存

```
xlabel('深度 x(m)');
```

```
ylabel('水温 y(℃)');
```

```
grid on
```



这就是拉格朗日插值算法的曲线，完美。

二维插值(曲面插值), 两个输入值 x, y, 一个输出值 z

已知数据	x_1	x_2	x_n
y_1	z_{11}	z_{12}	z_{1n}
y_2	z_{21}	z_{22}	z_{2n}
.....
y_m	z_{m1}	z_{m2}	z_{mn}

$x_1, x_2, x_3 \dots x_n$ 是已知坐标数据

对应的 y_1, y_2, y_3 也是已知的坐标数据

$z_{11} \dots z_{mn}$ 就是每个 x_n, y_n 对应的 z 数据

下面是一个加热板的数据，这些 x, y 都是已知的，对应的 z 也就是温度 T 也是已知的，所以我们尝试画出这个加热板的温度分布点

x/y	1	2	3	4	5
1	82	81	80	82	84
2	79	63	61	65	81
3	84	84	82	85	86

Meshgrid函数使用

$x=[1,2,3,4,5]$  x 一维坐标线

$y=[1,2,3,4,5]$  y 一维坐标先

$[xx, yy] = \text{meshgrid}(x, y)$ 用meshgrid(填入xy) 就能返回xy的二维坐标数据

就会出现x如果有5行, 那么y的座标就会变成5列

但是每一列的数据都是y的数据, 只不过是纵向排列了

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

如果y有5列, 那么x数据就要向下排5行

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

如果看不出来我们下面代码演示

```
x = [1 2 3 4 5];%一维数据5行  
y = [1 2 3];%一位数据3列
```

```
[xx, yy]=meshgrid(x, y);
```

```
xx =  
  
1 2 3 4 5  
1 2 3 4 5  
1 2 3 4 5  
  
>> yy  
  
yy =  
  
1 1 1 1 1  
2 2 2 2 2  
3 3 3 3 3
```

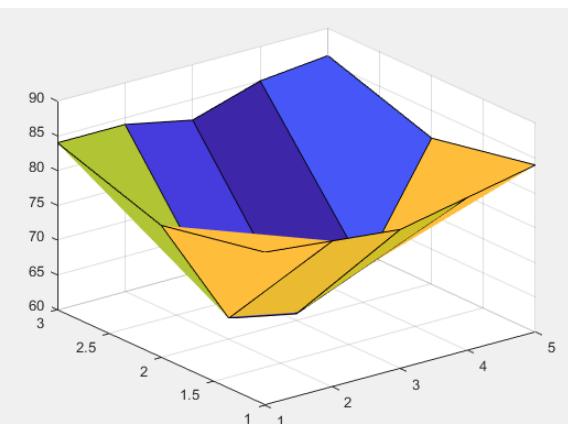
你看数据就是这样的，我们就用变成二维的 xx 和 yy 去画平面坐标。

```
x = [1 2 3 4 5];%一维数据5行  
y = [1 2 3];%一位数据3列
```

```
[xx, yy]=meshgrid(x, y);
```

```
zT = [82 81 80 82 84;  
      79 63 61 65 81;  
      84 84 82 85 86]; %这是每个xy对应的温度
```

```
surf(xx, yy, zT);
```



这就是任意 x, y 座标点对应的温度高度 zT 值

surf 三维图函数使用

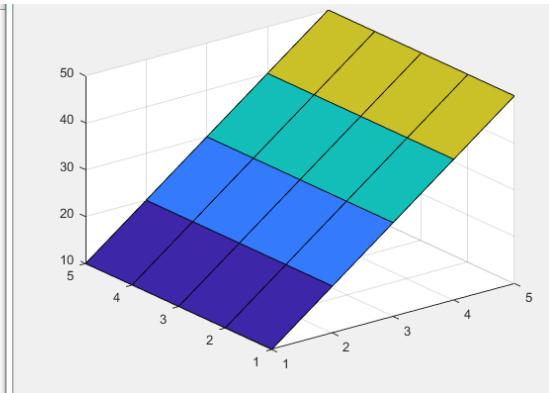
surf(网格的x座标，网格的y座标，z座标)

但是z座标必须是矩阵，就是z座标的数值数

量必须满足 $x \times y$ ，也就是 $5 \times 5 = 25$ 个数

```
x=[1 2 3 4 5];
y=[1 2 3 4 5];
z=[10 20 30 40 50
   10 20 30 40 50
   10 20 30 40 50
   10 20 30 40 50
   10 20 30 40 50];%你看z必须满足x*y矩阵个数

surf(x, y, z);
```



你看，这就是 surf 函数的使用

interp2 二维插值函数使用

z1 = interp2(已知x0 , 已知y0 , 已知z0 , 插值间隔x1 , y1 , 插值类型选择)

根据已知的x0 , y0 , z0 , 用method选择一种插值算法，对x0 , y0 , z0进

行插值，x1 , y1是要求插值算法按照现在x1 , y1的坐标来，最后得到插值后的z1

```
x=1:5;
y=1:3;

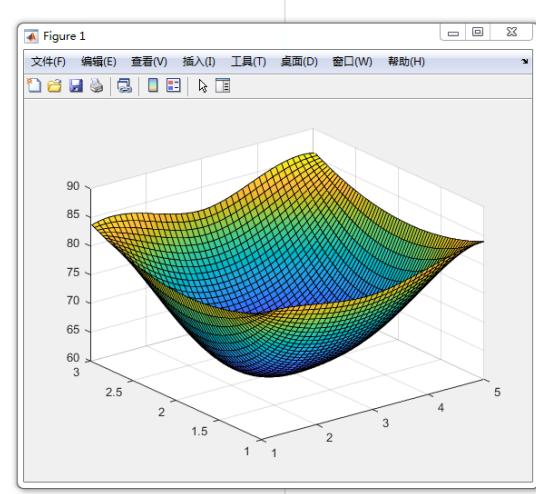
xq=1:0.1:5;
yq=1:0.03:3;

[xx, yy]=meshgrid(x, y);
[xxq, yyq]=meshgrid(xq, yq); %插值要求的座标系数

zT = [82 81 80 82 84;
       79 63 61 65 81;
       84 84 82 85 86]; %这是每个xy对应的温度

new_tmp = interp2(xx, yy, zT, xxq, yyq, 'spline');

%将已知的网格点xx, yy, 根据xx
%new_tmp也是zT插值之后的结果
surf(xxq, yyq, new_tmp);
```



其实我觉得就是三维图看起更细腻一点，不知道实际应用如何。

二维插值做离散点很大的插值算法

在中国南海的某海域，测得一些坐标点 $[x, y]$ 处的水深如下表，已知船的吃水深度为 **5m**，求在矩形区域 **[75, 200] x [-50, 150]** 的哪些地方，船要避免进入？

x	129.0	140.0	108.5	88.0	185.5	195.0	105.5
y	7.5	141.5	28.0	147.0	22.5	137.5	85.5
z	-4	-8	-6	-8	-6	-8	-8
x	157.5	107.5	77.0	81.0	162.0	162.0	117.5
y	-6.5	-81.0	3.0	56.5	-66.5	84.0	-38.5
z	-9	-9	-8	-8	-9	-4	-9

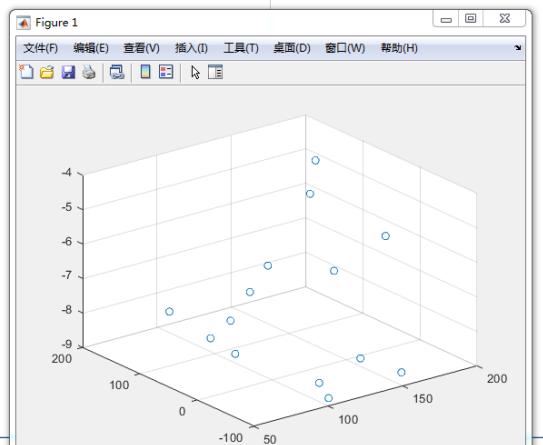
z 值是负值，是因为海平面我认为是 0，那么负数越大，水越深。

plot3 函数使用 绘制 x, y, z 的图

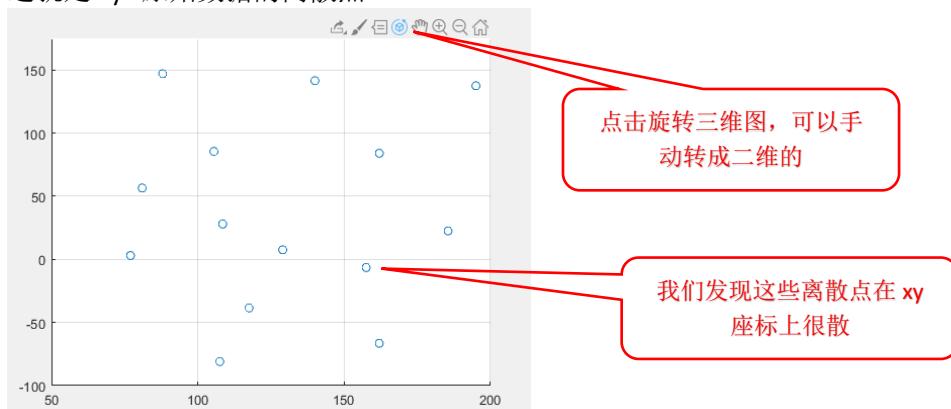
其实 **plot3** 就是比 **plot** 多了一个维度的形参，一看便知

```
x = [129.0 140.0 108.5 88.0 185.5 195.0 105.5 157.5 107.5 77.0 81.0 162.0 162.0 117.5];  
y = [7.5 141.5 28.0 147.0 22.5 137.5 85.5 -6.5 -81.0 3.0 56.5 -66.5 84.0 -38.5];  
z = [-4 -8 -6 -8 -6 -8 -9 -9 -8 -8 -9 -4 -9];
```

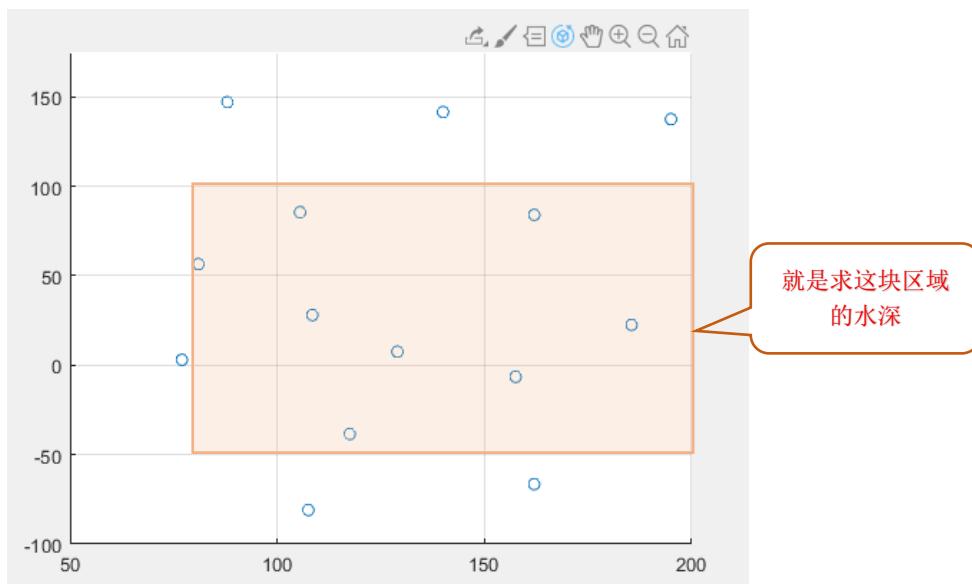
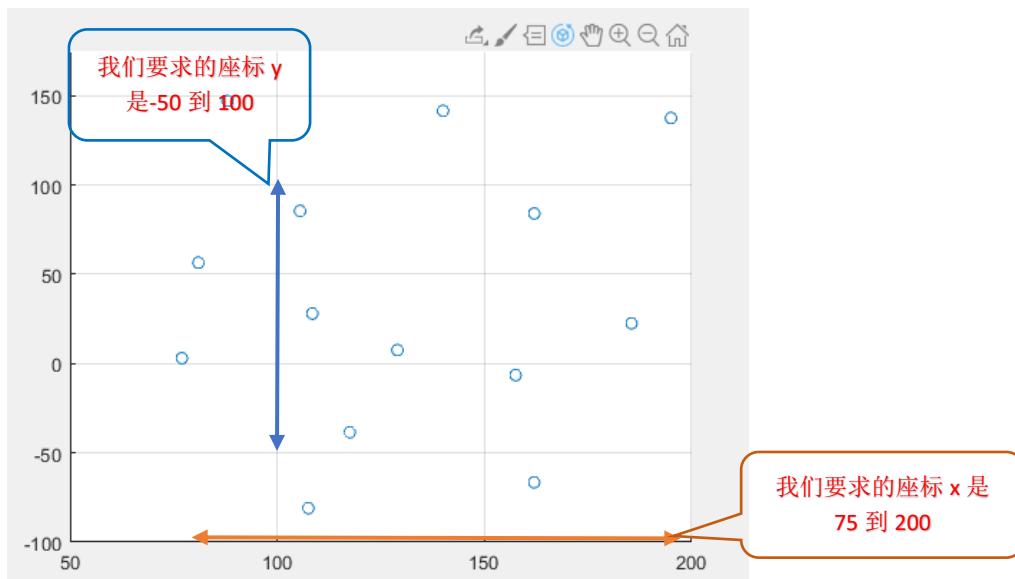
```
plot3(x,y,z, 'o'); %散点用圆圈表示  
grid on
```



这就是 xyz 原始数据的离散点



用二维插值方式，将已知这些散点处的水深，将其扩展到网格点上。
这样可以将网格画得很密



griddata 函数使用

```

x = [129.0 140.0 108.5 88.0 185.5 195.0 105.5 157.5 107.5 77.0 81.0 162.0 162.0 117.5];
y = [7.5 141.5 28.0 147.0 22.5 137.5 85.5 -6.5 -81.0 3.0 56.5 -66.5 84.0 -38.5];
z = [-4 -8 -6 -8 -6 -8 -9 -9 -8 -8 -9 -4 -9];

plot3(x, y, z, 'o'); % 散点用圆圈表示
grid on

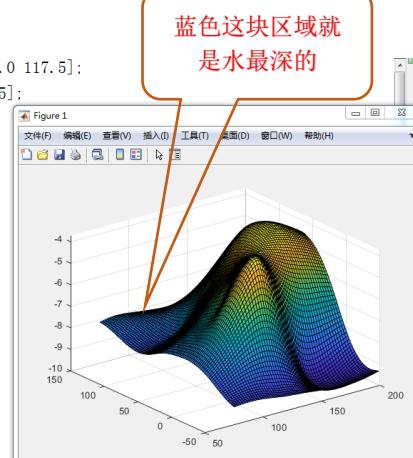
xq = 75:2:200;%这就是我们要的x区域
yq = -50:2:150;%这就是我们要的y区域

[xxq, yyq] = meshgrid(xq, yq);

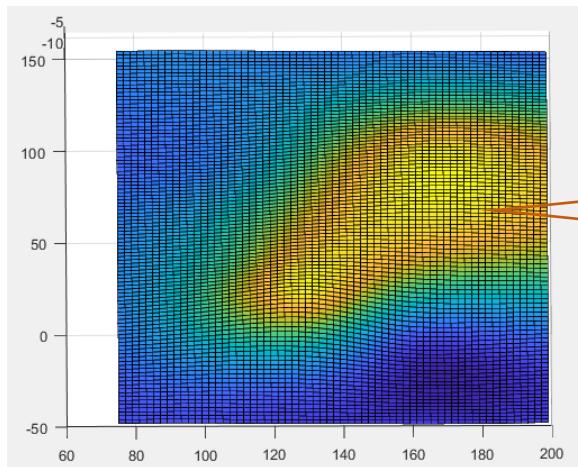
new_z = griddata(x, y, z, xxq, yyq, 'v4');
%把x, y, z这些已知的离散数据点, 用v4插值法扩展到xxq, yyq这种比较密的网格上面来
surf(xxq, yyq, new_z);

```

蓝色这块区域就是水最深的



如果你觉得水最深的座标不好看, 可以旋转成二维的



我把图像拉成二维就知道黄色区域是水最浅的，船不能过去

当然我们也可以用等高线来表示

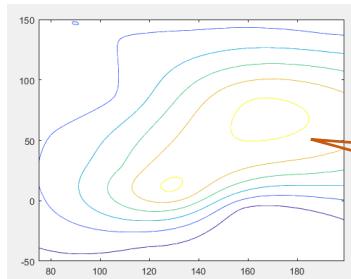
contour函数使用, contour(x座标, y座标, z座标)

```
[xxq, yyq] = meshgrid(xq, yq);
```

```
new_z = griddata(x, y, z, xxq, yyq, 'v4');
```

%把x, y, z这些已知的离散数据点, 用v4插值法扩展到xxq, yyq这种比较密的网格上面来
surf(xxq, yyq, new_z);

contour(xxq, yyq, new_z); %插值之后的 xxq, yyq, new_z用等高线表示



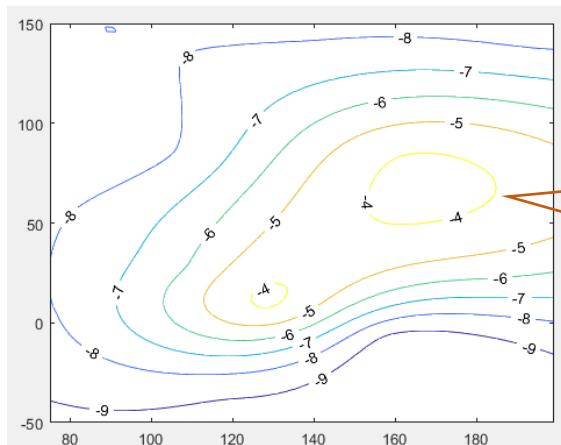
但是我看不到每条曲线的水深值, 必须鼠标点击

这就是等高线

%把x, y, z这些已知的离散数据点, 用v4插值法扩展到xxq, yyq这种比较密的网格上面来
surf(xxq, yyq, new_z);

加入 showtext on 打开

contour(xxq, yyq, new_z, 'ShowText', 'on'); %插值之后的 xxq, yyq, new_z用等高线表示



你看水深值出来了, 黄色的区域水最浅, 船不能靠近

figure 关键字使用

```
x = [129.0 140.0 108.5 88.0 185.5 195.0 105.5 157.5 107.5 77.0 81.0 162.0 162.0 117.5];  
y = [7.5 141.5 28.0 147.0 22.5 137.5 85.5 -6.5 -81.0 3.0 56.5 -66.5 84.0 -38.5];  
z = [-4 -8 -6 -8 -6 -8 -9 -9 -8 -8 -9 -4 -9];
```

```
plot3(x, y, z, 'o'); %散点用圆圈表示  
grid on
```

显示散点座标图

```
xq = 75:2:200;%这就是我们要的x区域  
yq = -50:2:150;%这就是我们要的y区域
```

```
[xxq, yyq] = meshgrid(xq, yq);
```

显示三维座标图

```
new_z = griddata(x, y, z, xxq, yyq, 'v4');
```

```
%把x, y, z这些已知的离散数据点, 用v4插值法扩展到xxq, yyq这种比较密的网格上面来  
surf(xxq, yyq, new_z);
```

```
contour(xxq, yyq, new_z, 'ShowText', 'on'); %插值之后的 xxq, yyq, new_z用等高线表示
```

显示等高线图

我们发现运行程序之后只能显示最后执行的等高线图, 前面两个图没有窗口出来显示? 貌似被最后的等高线图覆盖了。

这是因为你没有用 `figure` 去控制住前面的图, `figure` 类似 `hold on` 关键字, `hold on` 关键字是控制多条曲线显示, `figure` 就是控制多座标图显示。

```
plot3(x, y, z, 'o'); %散点用圆圈表示
```

```
figure
```

```
grid on
```

每个显示图函数后面加 figure

```
xq = 75:2:200;%这就是我们要的x区域  
yq = -50:2:150;%这就是我们要的y区域
```

```
[xxq, yyq] = meshgrid(xq, yq);
```

```
new_z = griddata(x, y, z, xxq, yyq, 'v4');
```

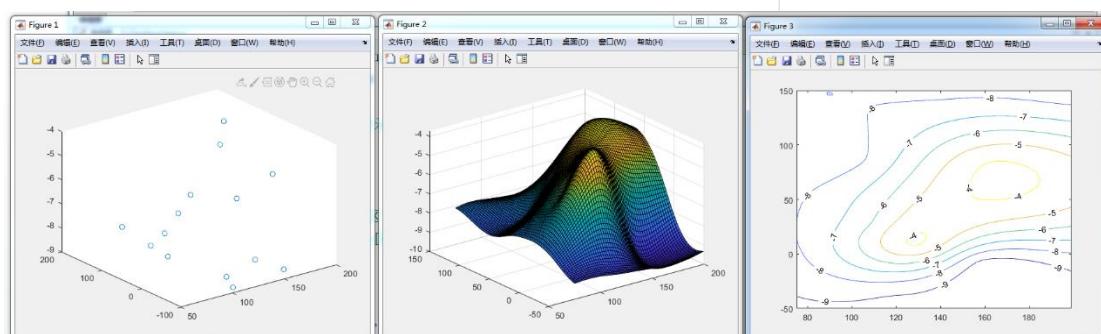
```
%把x, y, z这些已知的离散数据点, 用v4插值法扩展到xxq, yyq这种比较密的网格上面来
```

```
surf(xxq, yyq, new_z);
```

```
figure
```

每个显示图函数后面加 figure

```
contour(xxq, yyq, new_z, 'ShowText', 'on'); %插值之后的 xxq, yyq, new_z用等高线表示
```



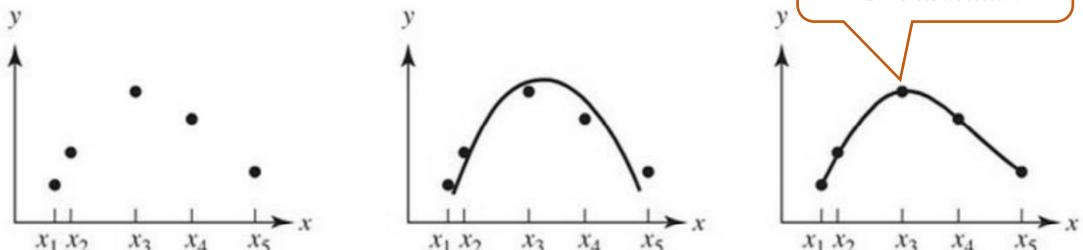
你看, 三个图都显示出来了。

线性拟合实现

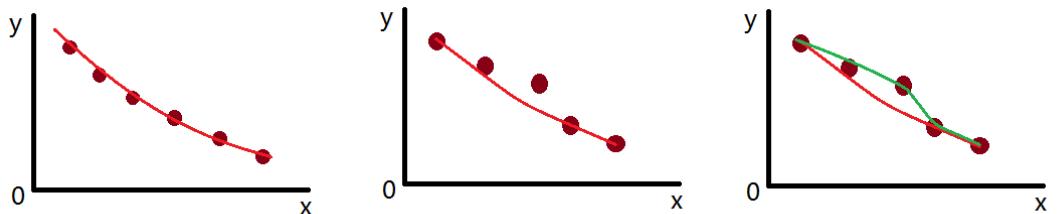
还是老问题， x 和 y 的关系问题

输入值	x_1	x_2	x_3	x_n
输出值	y_1	y_2	y_3	y_n

■ 数据插值 (interpolation)



这样插值就存在一个问题，你看似插值曲线通过了每个离散数据点，但是又有多少插值算法符合这种要求呢？



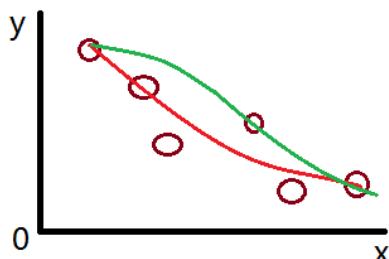
这是某一种类型的插值曲线
算法，正好符合这些离散数
据点的要求，所以能做到很
好的匹配这些离散数据点。

假设这是A类插值算法

用A类插值算法，来计算另一种传感器
的离散点，发现A类插值算法不灵了

这种情况就要找另外一种插值算法看
符不符合要求，绿色线就是我找到符
合要求的插值算法，我假定为B类

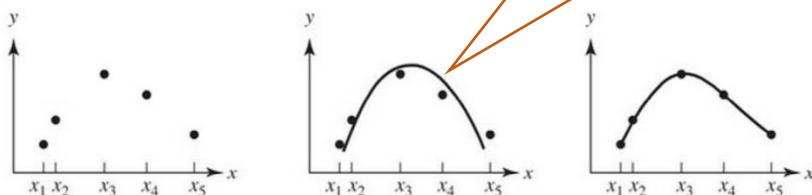
但是这些都是属于能找到对应离散数据点的插值算法的理想情况下。

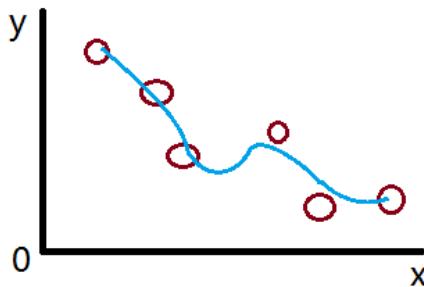


有些传感器的离散数据点，用A和B
类算法都不符合离散数据点，找的
其他插值算法也做不到，那么就不
能再用插值算法来做该离散点的线
性化方法

■ 数据插值 (interpolation)

可以用拟合的方式来
做插值无法做到的离
散点线性化





虽然拟合曲线无法像插值那样精确的落在每个点上面，但是拟合的曲线都是接近每个点的，总比插值算法在有些离散点的位置完全无法接近要好

$y = kx + b$ 这是线性拟合公式的一种

$y = ax^3 + bx^2 + cx$ 这也是线性拟合公式的一种

这些拟合公式可以通过修改 k, b 待定系数，或者修改 a, b, c 这些待定系数让拟合曲线接近于每个离散点

拟合公式就比较灵活，可以修改参数来做到离散点的线性化，不想插值算法，必须选择某一个固定的插值算法，如果所有固定的插值算法都无法满足还做不成

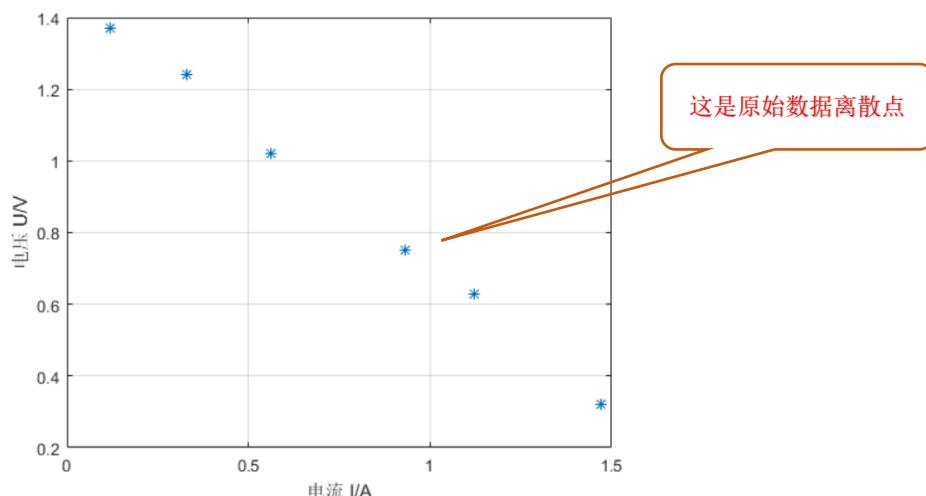
下面来讲案例

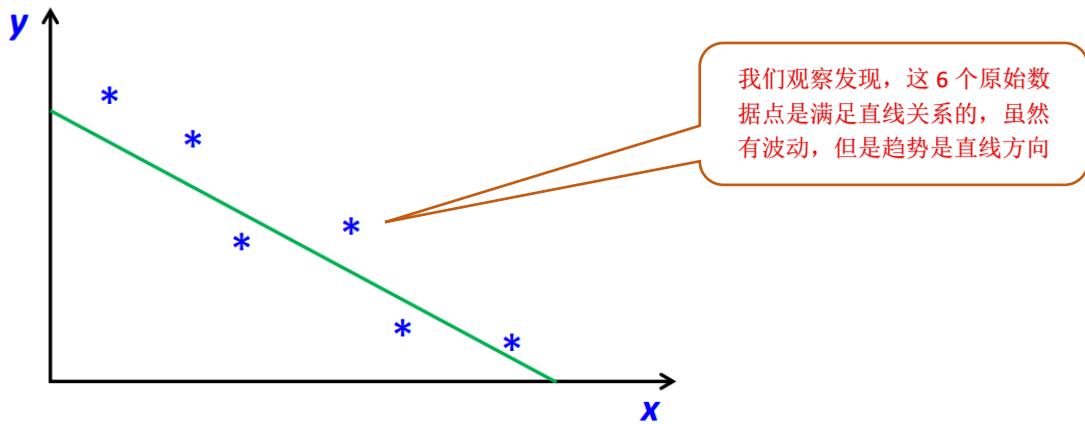
$$U = E - I \times r$$

U 和 I 是已知量，现在要用 U 和 I 已知量求出 E 和 r 未知量。所以这是一个 2 自变量和 2 因变量的方程，这还不像 $y = kx + b$ 只有 y 一个输出量，这是由 E 和 r 两个输出量(因变量)的方程。

电流 I/A	0.12	0.33	0.56	0.93	1.12	1.47
电压 U/V	1.37	1.24	1.02	0.75	0.63	0.32

U 和 I 的原始数据





$$\begin{cases} U_1 = E - I_1 r \\ U_2 = E - I_2 r \\ U_3 = E - I_3 r \\ U_4 = E - I_4 r \\ U_5 = E - I_5 r \\ U_6 = E - I_6 r \end{cases} \Leftrightarrow \begin{bmatrix} 1 & -I_1 \\ 1 & -I_2 \\ 1 & -I_3 \\ 1 & -I_4 \\ 1 & -I_5 \\ 1 & -I_6 \end{bmatrix} \begin{bmatrix} E \\ r \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{bmatrix} \Leftrightarrow AX = B$$

用矩阵乘法，提取未知量 E, r

$$\begin{cases} U_1 = E - I_1 r \\ U_2 = E - I_2 r \\ U_3 = E - I_3 r \\ U_4 = E - I_4 r \\ U_5 = E - I_5 r \\ U_6 = E - I_6 r \end{cases} \Leftrightarrow \begin{bmatrix} 1 & -I_1 \\ 1 & -I_2 \\ 1 & -I_3 \\ 1 & -I_4 \\ 1 & -I_5 \\ 1 & -I_6 \end{bmatrix} \begin{bmatrix} E \\ r \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{bmatrix} \Leftrightarrow AX = B$$

方程个数为 6 个

未知个数为 2 个

所以方程个数大于未知个数 $6 > 2$ ，这叫做线性超定方程，不存在精确解。

MATLAB 求解超定方程组 $AX = B$ 的最小二乘解： $X = A \setminus B$

ones 函数用法

ones(参数 1) 生成默认为 1 的几行几列矩阵

`>> ones(3)`

`ans =`

```
1     1     1
1     1     1
1     1     1
```

`ones(行, 列)` 指定几行, 几列矩阵, 默认为 1

```
>> ones(2, 4)
```

```
ans =
```

```
1 1 1 1  
1 1 1 1
```

生成 6 行, 1 列矩阵 默认为 1

```
>> ones(6, 1)
```

```
ans =
```

```
1  
1  
1  
1  
1  
1
```

下面来写线性超定方程代码

```
I = [0.12 0.33 0.56 0.93 1.12 1.47]'; % 符号 ' 把数组行变成数组列
```

```
U = [1.37 1.21 1.02 0.75 0.63 0.32]'; % 符号 ' 把数组行变成数组列
```

```
plot(I, U, '*')
```

```
xlabel('电流 I/A')
```

```
ylabel('电压 U/V')
```

```
grid on
```

```
hold on
```

```
A=[ones(6, 1), -I];%二维矩阵, 第1列全为1, 第2列为I变量, 给I变量加上负数
```

```
B=U;%把电压给B变量
```

```
X=A\B;% 矩阵除法, 6行2列矩阵A \ 1列矩阵B
```

```
disp(X)
```

```
>> test2
```

1.4615

得到 E 系数

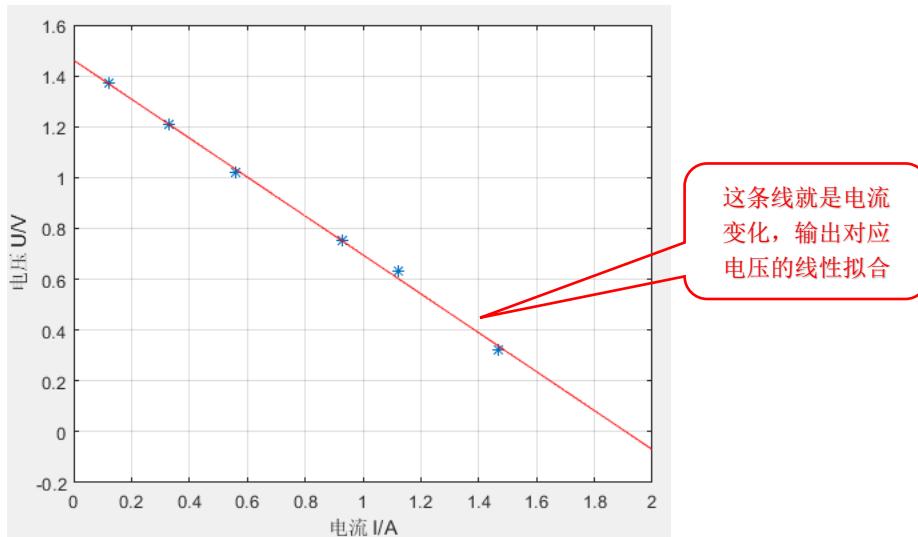
0.7657

得到 r 系数

```
A=[ones(6,1), -I]; %二维矩阵，第1列全为1，第2列为I变量，给I变量加上负数
B=U; %把电压给B变量
```

```
X=A\B; % 矩阵除法，6行2列矩阵A \ 1列矩阵B
disp(X) %得到E和r系数
```

```
E=X(1); %取出E系数
r=X(2); %取出r系数
Igrid = 0:0.1:2;
U1=E - Igrid * r;
plot(Igrid,U1, 'r');
```



你会发现并不是所有电流的变化，电压都能精确的反映出来，只是可以近似的反映，看自己精度要求。

多项式拟合

已知数据 x 和 y 之间满足三次多项式的关系： $y = p_1x^3 + p_2x^2 + p_3x + p_4$

这个多项式公式是自己找的，并不是自己发明的，我们就看这个多项式公式适不适合我们下面对原始数据进行拟合。

x	-2.8	-2	-0.8	0.7	1.8	2.9	3.5
y	-48.1	-7.4	14.2	-8.3	-16.7	12.6	33.5

将各组 x y 数据，代入到三次多项式里，并改写成矩阵的形式

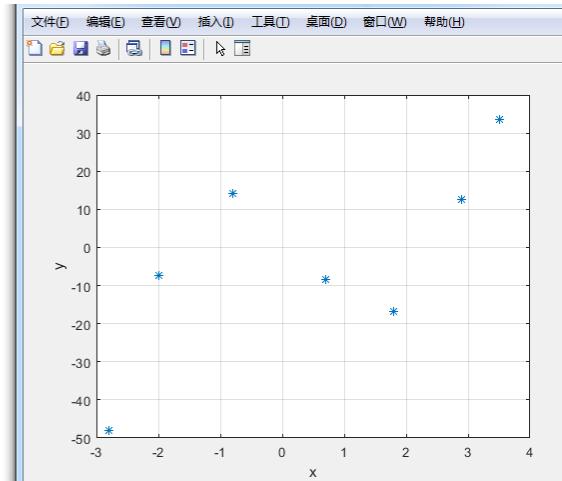
$$\begin{cases} y_1 = p_1x_1^3 + p_2x_1^2 + p_3x_1 + p_4 \\ y_2 = p_1x_2^3 + p_2x_2^2 + p_3x_2 + p_4 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \Leftrightarrow \\ y_6 = p_1x_6^3 + p_2x_6^2 + p_3x_6 + p_4 \\ y_7 = p_1x_7^3 + p_2x_7^2 + p_3x_7 + p_4 \end{cases} \begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_6^3 & x_6^2 & x_6 & 1 \\ x_7^3 & x_7^2 & x_7 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_6 \\ y_7 \end{bmatrix} \Leftrightarrow Ap = B$$

也就是矩阵乘法。这也是方程个数>未知个数，所以叫线性超定方程。

现在我们要求系数 P1, P2, P3, P4

```
x = [-2.8 -2 -0.8 0.7 1.8 2.9 3.5]';
y = [-48.1 -7.4 14.2 -8.3 -16.7 12.6 33.5]';
```

```
plot(x,y,'*');
xlabel('x');
ylabel('y');
grid on
hold on
```



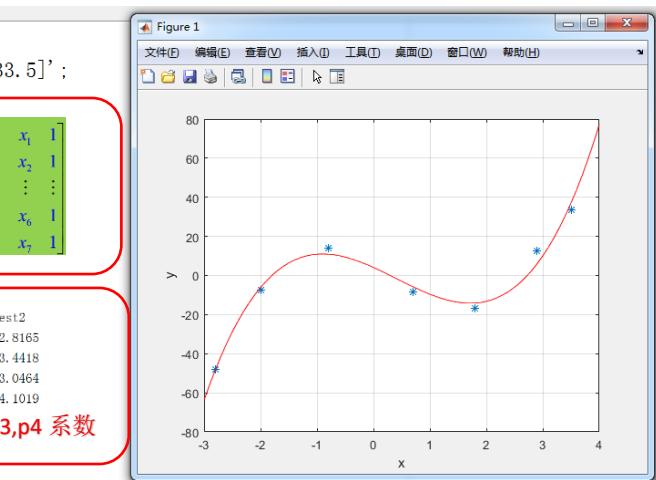
这是原始数据

```
x = [-2.8 -2 -0.8 0.7 1.8 2.9 3.5]';
y = [-48.1 -7.4 14.2 -8.3 -16.7 12.6 33.5]';

plot(x,y,'*');
xlabel('x');
ylabel('y');
grid on
hold on

A=[x.^3, x.^2, x, ones(7, 1)];
B=y;
P=A\B;

p1,p2,p3,p4 系数
```



$y = p_1x^3 + p_2x^2 + p_3x + p_4$
将 P1P2P3P4 系数带入，求出每个 x 变量对应输出的 y 值

我们看到曲线接近每个离散点。

这就是多项式拟合，但是多项式拟合的系数和次方是可以修改的，让曲线更接近每个离散点。

polyfit Matlab 自带多项式拟合函数使用

p = polyfit(x, y, n)

xy: 已知数据点

n: 为多项式的阶次

p: 返回的系数数组[P1,P2,P3.....Pn]

$$y = p_1x^n + p_2x^{n-1} + \dots + p_nx + p_{n+1}$$

这就是 **polyfit** 的多项式函数对应的式子，其实 n 就是 x 的次数，p 就是得到的系数，可以写任意次数的多项式。

我们还是以 3 次多项式为例，求解上两页的 xy 数据

x	-2.8	-2	-0.8	0.7	1.8	2.9	3.5
y	-48.1	-7.4	14.2	-8.3	-16.7	12.6	33.5

$$y = P_1 x^3 + P_2 x^2 + P_3 x + P_4$$

$x = [-2.8 \ -2 \ -0.8 \ 0.7 \ 1.8 \ 2.9 \ 3.5]'$;
 $y = [-48.1 \ -7.4 \ 14.2 \ -8.3 \ -16.7 \ 12.6 \ 33.5]'$;

```
plot(x, y, '*');
xlabel('x');
ylabel('y');
grid on
hold on
```

```
P = polyfit(x, y, 3); %你看多方便就把系数p求出来了
disp(P);
```

```
>> test2
      P1      P2      P3      P4
    2.8165  -3.4418  -13.0464  4.1019
```

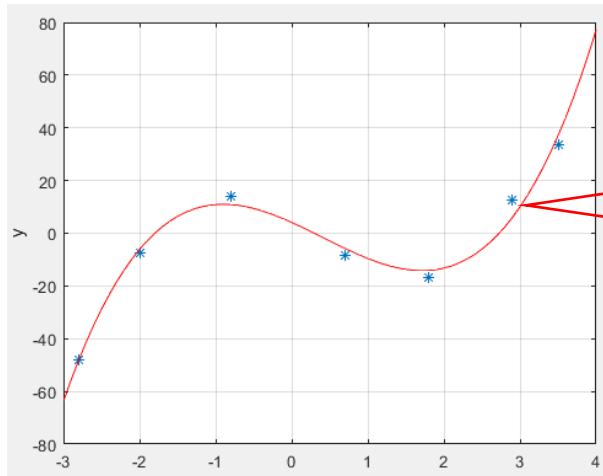
polyval 函数使用 输出变量 $y = \text{polyval}(\text{系数}, \text{自变量 } x)$

```
x = [-2.8 -2 -0.8 0.7 1.8 2.9 3.5]';
y = [-48.1 -7.4 14.2 -8.3 -16.7 12.6 33.5]';
```

```
plot(x, y, '*');
xlabel('x');
ylabel('y');
grid on
hold on
```

```
P = polyfit(x, y, 3); %你看多方便就把系数p求出来了
disp(P);
```

```
x1 = -3:0.1:4;
y1 = polyval(P, x1); %根据系数和变量输出y1
plot(x1, y1, 'r');
```

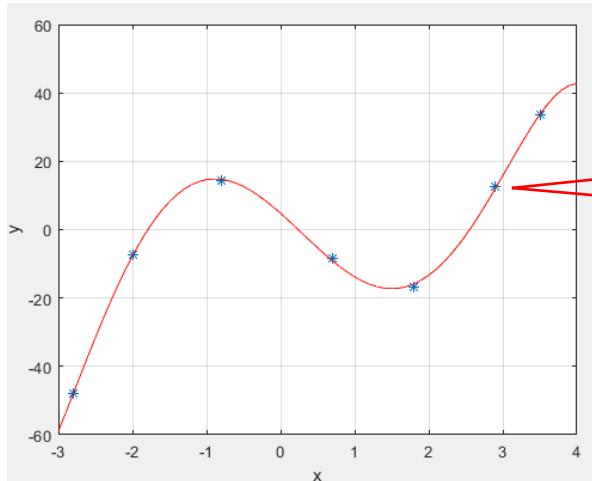


你看输出结果和上一节一样，其实可以增加次数来让曲线更靠近离散点

```
P = polyfit(x, y, 5); %你看多方便就把系数p求出来了
```

```
disp(P);
```

我做成 5 次试试



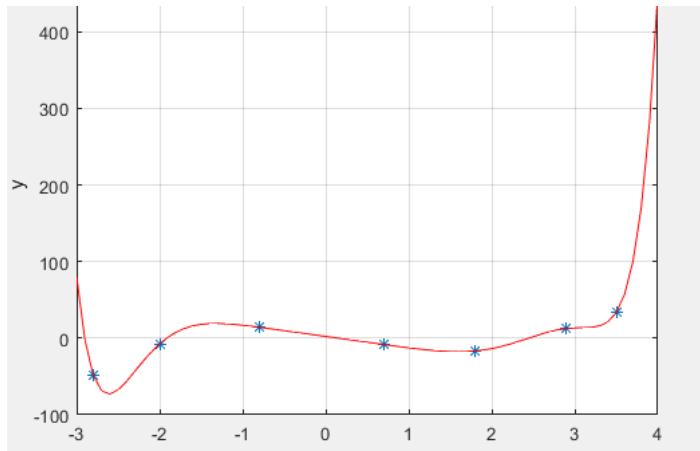
你看 5 次就基本和离散点一模一样了

那是不是次数越多越好呢?

```
P = polyfit(x, y, 10); %你看多方便就把系数p求出来了
```

```
disp(P);
```

我改成 10 次



你看，我改成 10 次就过拟合了。

所以不是次数越多越好，而是选择合适的次数。

x	-2.8	-2	-0.8	0.7	1.8	2.9	3.5
y	-48.1	-7.4	14.2	-8.3	-16.7	12.6	33.5

在这种离散点拟合案例里面 5 次最好。

指数线性拟合

药物浓度案例

时刻 x	0.25	0.5	1	1.5	2	3	4	6	8
浓度 y	19.21	18.15	15.36	14.1	12.89	9.32	7.45	5.24	3.01

```

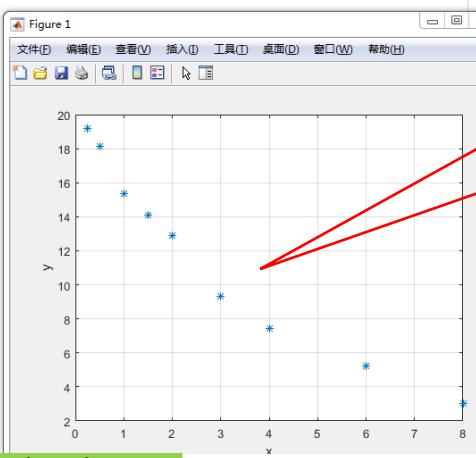
x = [0.25 0.5 1 1.5 2 3 4 6 8]';
y = [19.21 18.15 15.36 14.1 12.89 9.32 7.45 5.24 3.01]';

```

```

plot(x,y,'*');
xlabel('x');
ylabel('y');
grid on
hold on

```



根据原始数据表示，我们发现这个离散点弯曲得很厉害，有点像指数的曲线关系

所以这时候不能用上面的 $y = p_1x^3 + p_2x^2 + p_3x + p_4$ 线性方程来拟合了，要用指数方程。

设 x 和 y 之间满足指数函数的关系： $y = ae^{bx}$

因为没有办法用前面的线性方程组，所以只有用指数方程组，指数方程组需要下面这样变换对 xy 的函数关系式的两边分别取对数，可得

$$y = ae^{bx} \Leftrightarrow \ln y = \ln a + \ln e^{bx} = \ln a + bx$$

我们主要是求待定系数 a, b

$$\begin{cases} \ln y_1 = \ln a + bx_1 \\ \ln y_2 = \ln a + bx_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \ln y_8 = \ln a + bx_8 \\ \ln y_9 = \ln a + bx_9 \end{cases} \Leftrightarrow \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_8 \\ 1 & x_9 \end{bmatrix} \begin{bmatrix} \ln a \\ b \end{bmatrix} = \begin{bmatrix} \ln y_1 \\ \ln y_2 \\ \vdots \\ \ln y_8 \\ \ln y_9 \end{bmatrix} \Leftrightarrow AX = B$$

```

x = [0.25 0.5 1 1.5 2 3 4 6 8]';
y = [19.21 18.15 15.36 14.1 12.89 9.32 7.45 5.24 3.01]';

```

```

plot(x,y,'*');
xlabel('x');
ylabel('y');
grid on
hold on
A = [ones(9,1),x];
B = log(y);
X = A \ B;
disp(X)

```

所以 $\ln a = 2.9943$ 还必须把 a 求出来

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_8 \\ 1 & x_9 \end{bmatrix}$$

$$\begin{bmatrix} \ln y_1 \\ \ln y_2 \\ \vdots \\ \ln y_8 \\ \ln y_9 \end{bmatrix}$$

2.9943 是 $\ln a$ 的系数结果

```

>> test2
2.9943
-0.2347

```

-0.2347 是 b 的系数

```

A = [ones(9, 1), x];
B = log(y);
X = A \ B;
disp(X)

```

$$\begin{cases} \ln a = X(1) \\ b = X(2) \end{cases} \Rightarrow \begin{cases} a = e^{X(1)} \\ b = X(2) \end{cases}$$

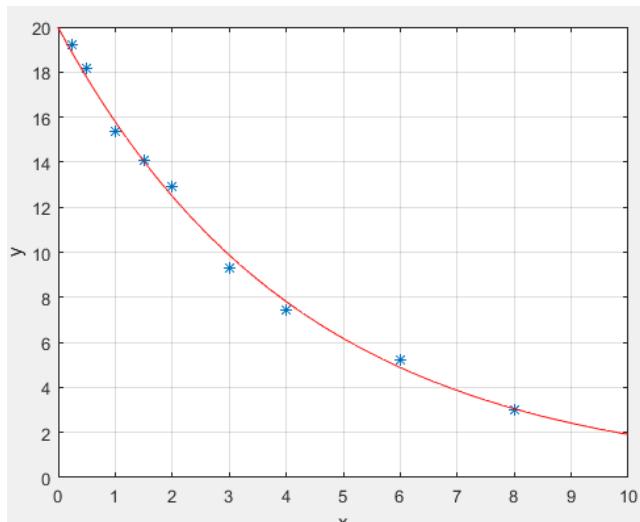
$a = \exp(X(1))$; % a 取系数 $X(1)$ 1na 的变化得到 a
 $b = X(2)$; % b 就是第 2 行系数

```

x1 = 0:0.1:10;
y1 = a*exp(b*x1);
plot(x1, y1, 'r');

```

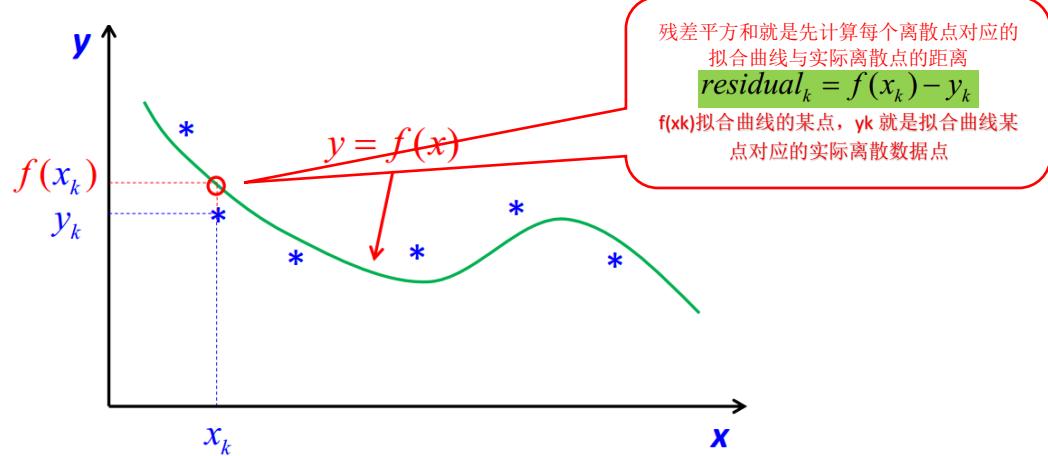
$$y = ae^{bx}$$



这就是指数线性拟合。

非线性拟合

残差平方和的定义



$$resnorm = \sum_{k=1}^n [f(x_k) - y_k]^2 = \sum_{k=1}^n residual_k^2$$

将每个点的 residual k 求平方，然后再一起加起来得到 resnorm。

所以 resnorm 残差平方和，就是衡量你这条拟合曲线是否理想的指标。

人口预测案例

■ 已知某地区在 1970 - 2015 年的人口数据 (万人) 如下，预测该地区在 2040 年前的人口数量

年份	1970	1975	1980	1985	1990	1995	2000	2005	2010	2015
第 x 年	0	5	10	15	20	25	30	35	40	45
人口 y	9.7	20.3	34.2	55.7	95.3	168.5	217.8	257.6	296.1	330.7

■ **Malthus** 人口模型：假设单位时间内的人口增长率 r 为常数

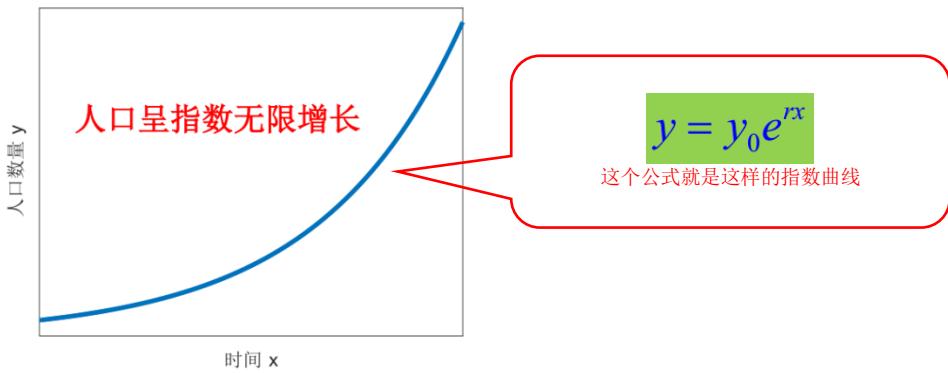
$$r = \frac{y(x + \Delta x) - y(x)}{y(x)\Delta x} \Rightarrow \frac{y(x + \Delta x) - y(x)}{\Delta x} = ry(x)$$

如果 Δx 无限接近于 0，那么就变成下面这个公式

■ 当 $\Delta x \rightarrow 0$ 时，可得微分方程 \rightarrow 起始年份的人口数量为 $y_0 \rightarrow$ 微分方程的解析解

$$\begin{cases} \frac{dy}{dx} = ry \\ y(0) = y_0 \end{cases} \Rightarrow y = y_0 e^{rx}$$

给定一个起始年份的人口数量，那么这个人口数量就是 y_0



由于资源和环境容量等的限制



人口不可能无限制地增长

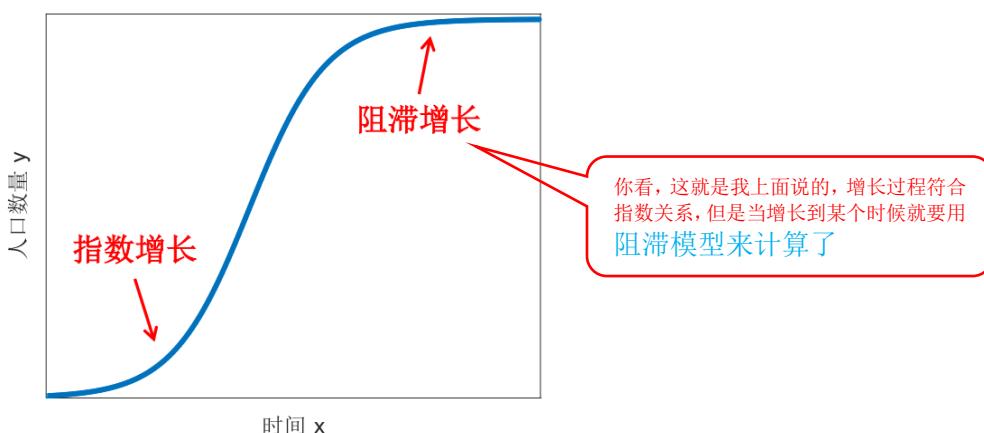
所以上面这个指数公式只能在某一段有效

那么就要修改上面的公式，使用人口阻滞增长模型 —— **Logistics** 模型，这样的模型才比较合理

人口增长率不再是常数，而是人口数量 y 的函数 $r(y) \rightarrow r(y)$ 是 y 的减函数

$$\begin{cases} \frac{dy}{dx} = r(y)y = r\left(1 - \frac{y}{y_m}\right)y \\ y(0) = y_0 \end{cases} \quad \rightarrow \quad y = \frac{y_m}{1 + \left(\frac{y_m}{y_0} - 1\right)e^{-rx}}$$

所以 $r(y)$ 就是随着人口 y 的增大， $r(y)$ 是降低的



设:有限的资源和环境，能够供养的最大人口数量为 ym

$$y = \frac{y_m}{1 + \left(\frac{y_m}{y_0} - 1\right)e^{-rx}}$$

如果 x 越大，因为是乘以 $-r$ ，那么 e 的结果就越小，这就会导致分母越小

所以 x 越大， y 就越大。

$$y = \frac{y_m}{1 + \left(\frac{y_m}{y_0} - 1 \right) e^{-rx}}$$

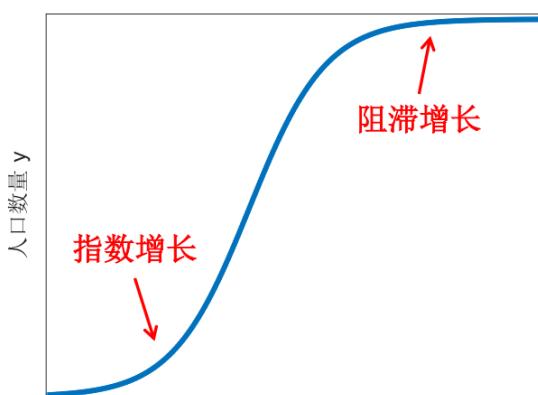
但是当 x 趋向于无穷大的时候， e 就会趋向于 0，那么分母就会趋向于 1，那么 y 就会变小， y 就是趋向于 y_m

$$r(y)y = r \left(1 - \frac{y}{y_m} \right) y$$

当 y 很小的时候，括号里面趋向于 1，那么 $r(y)$ 就趋向于 r 这么一个常数，所以当 y 很小的时候人口呈指数增长（也就是 malthus 模型）

$$r(y)y = r \left(1 - \frac{y}{y_m} \right) y$$

当 y 接近于 y_m 的时候，括号里面接近于 0，人口增长率接近于 0，人口不再增长



这就是上面公式的曲线表现。

- 已知某地区在 1970 - 2015 年的人口数据 (万人) 如下，预测该地区在 2040 年前的人口数量

年份	1970	1975	1980	1985	1990	1995	2000	2005	2010	2015
第 x 年	0	5	10	15	20	25	30	35	40	45
人口 y	9.7	20.3	34.2	55.7	95.3	168.5	217.8	257.6	296.1	330.7

我们用 *Logistics* 来做这组数据的人口预测

$$y = \frac{y_m}{1 + \left(\frac{y_m}{y_0} - 1 \right) e^{-rx}}$$

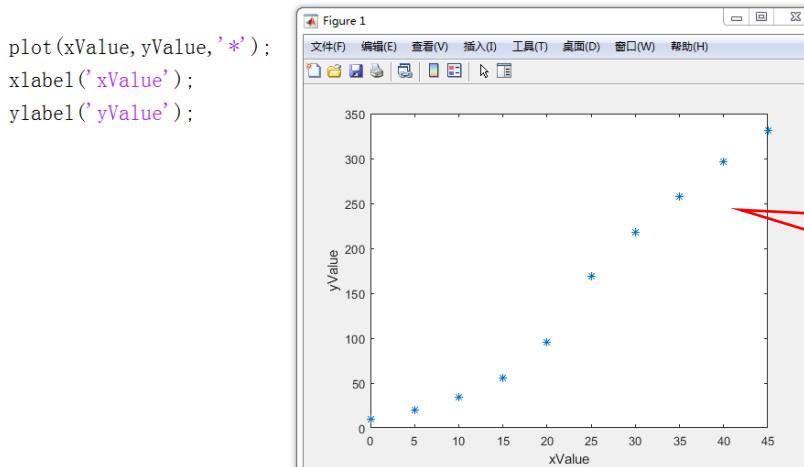
先要求出待定参数： $r & y_m$

y_0 是已知人口数量的起始数量值 9.7

有了 r 和 y_m 就可以填入 x 求出 y

已知的 x 和 y 根据上面表格来看是 10 组，那么就是 10 个方程。但是未知数只有 2 个 y_m 和 r ，应该像前面章节那样做线性超定方程。但是根据表格发现这是非线性的数据曲线，所以没办法做线性拟合。

```
xValue = [0 5 10 15 20 25 30 35 40 45];
yValue = [9.7 20.3 34.2 55.7 95.3 168.5 217.8 257.6 296.1 330.7];
```



这是原始数据离散点，
我们觉得开始像指数，
后面趋近于平缓

像这种数据形式我们就可以用 **malthus** 模型和 **Logistics** 模型

lsqcurvefit 非线性拟合函数使用

```
param = lsqcurvefit(fun, param0, xdata, ydata)
xdata: x 点原始数据
ydata: y 点原始数据
param0: 待定参数初始值(r, ym)
param: lsqcurvefit 执行后求解出来的待定参数新初值
fun: 定义拟合函数 y=f(x), 其实就是你要先写一个拟合函数传递给 lsqcurvefit, 这样 lsqcurvefit 才能执行, 所以你要写两个 m 文件。
Fun 调用拟合函数格式要求必须是 function y= (param, x)
```

待定参数初值的确定 **param0 = [0.15, 400]**

✓ 当人口总数较少时, 近似成指数增长, 令 **x=5**

$$\frac{y}{y_0} = e^{rx} \Rightarrow \frac{y_{1975}}{y_{1970}} = \frac{20.3}{9.7} \approx e^{5r} \Rightarrow r \approx 0.15$$

待定参数初始是确定人口比较少的时候的 r, 和最大人口数量的 ym

```
xValue = [0 5 10 15 20 25 30 35 40 45];
yValue = [9.7 20.3 34.2 55.7 95.3 168.5 217.8 257.6 296.1 330.7];
```

如果 x=5, 就取 x=1975 年到 1970 年人口 y 值, 相除, 经过 e 运算得到 r

```
plot(xValue, yValue, '*');
xlabel('xValue');
ylabel('yValue');
```

待定初值 400, 是 ym 人口最大的值。我取已知数据 x=45 时 y=330.7, 为了稳妥取的 400

```
global y0 % 定义全局变量给其它m文件用
y0 = yValue(1); % 把起始年份人口数量 y 的初始值提取出来
param0 = [0.15, 400]; % 待定参数初值
param = lsqcurvefit('test', param0, xValue, yValue);
% 把函数和待定参数和原始数据传入非线性拟合函数
disp(param);
<stopping_criteria_details>
```

0.1337 352.1669

求解出来的新待定参数 r 和 ym

这就是 param 新待定参数 r 和 ym

下面就是传入 lsqcurvefit 函数的子函数 test

```
test.m × +
function y = test(param, x)
global y0 %其它m文件定义的全局变量，这里直接使用
r = param(1); %参数r
ym = param(2); %参数ym
y = ym ./ (1 + (ym/y0 - 1) * exp(-r*x));
end
```

‘./’，点除和 ‘.*’一样

这就是新待定参数 r 和 ym

$$y = \frac{y_m}{1 + \left(\frac{y_m}{y_0} - 1 \right) e^{-rx}}$$

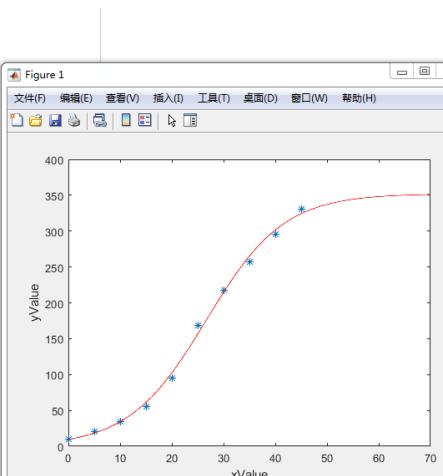
将求得的 r 和 ym 带入，可以算出 x 对应的 y

画出拟合后的曲线

```
xValue = [0 5 10 15 20 25 30 35 40 45];
yValue = [9.7 20.3 34.2 55.7 95.3 168.5 217.8 257.6 296.1 330.7];
plot(xValue, yValue, '*');
xlabel('xValue');
ylabel('yValue');
hold on

global y0 %定义全局变量给其它m文件用
y0 = yValue(1); %把起始年份人口数量y的初始值提取出来
param0 = [0.15, 400]; %待定参数初值
param = lsqcurvefit('test', param0, xValue, yValue);
%把函数和待定参数和原始数据传入非线性拟合函数
disp(param);

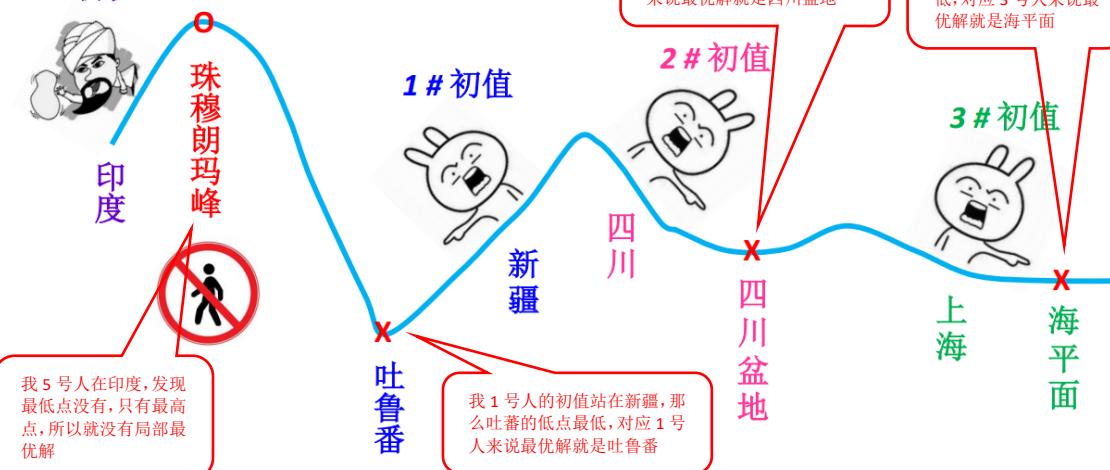
x=0:0.1:70;
y=test(param, x); %调用test，传入新的待定系数param，求每个x对应的y
plot(x, y, 'r');
```



待定参数初值的意义

首先得讲一个概念局部最优解

5# 初值

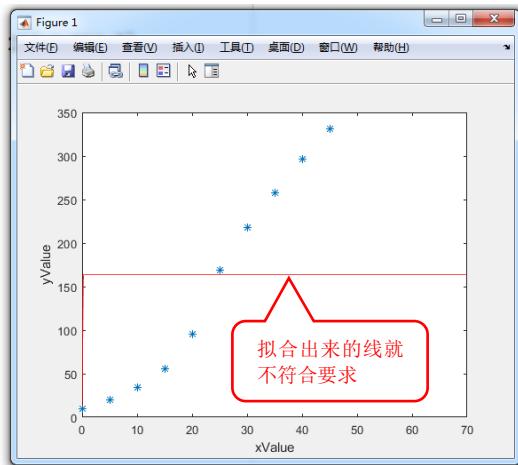


所以局部最优解是根据你人的初值来的，初值设置的好坏决定了你曲线和离散点的逼近程度

```
xValue = [0 5 10 15 20 25 30 35 40 45];
yValue = [9.7 20.3 34.2 55.7 95.3 168.5 217.8 257.6

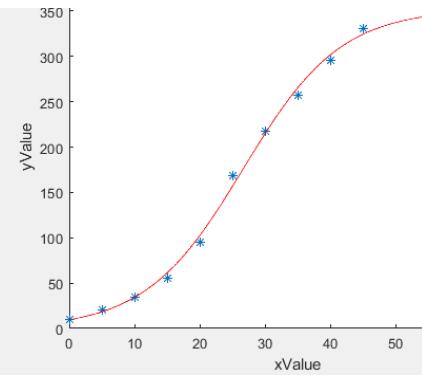
plot(xValue,yValue,'*');
xlabel('xValue');
ylabel('yValue');
hold on

global y0 %定义全局变量给其它m文件用
y0 = yValue(1);%把起始年份人口数量y的初始值提取出来
param0 = [30,600];%待定参数初值
param = lsqcurvefit('test',param0,xValue,yValue);
%把函数和待定参数和原始数据传入非线性拟合函数
disp(param);
```



```
ylabel('yValue');
hold on

global y0 %定义全局变量给其它m文件用
y0 = yValue(1);%把起始年份人口数量y的初始值提取出来
param0 = [0.15,400];%待定参数初值
param = lsqcurvefit('test',param0,xValue,yValue);
%把函数和待定参数和原始数据传入非线性拟合函数
disp(param);
```



记住 0.15 和 400 只是这段曲线的最优解，如果曲线拉长，后面出现更多的离散点，那么可能还需要在后面曲线找新的初值，得到新的最优解。

土壤含水率案例

水头 h	0.00	27.2	53.04	62.56	69.36	81.60	95.20	108.80	126.48
含水率 θ	0.476	0.434	0.406	0.401	0.392	0.382	0.365	0.351	0.335
水头 h	159.12	197.20	251.60	262.48	286.96	359.04	452.88	503.20	
含水率 θ	0.312	0.287	0.271	0.261	0.253	0.252	0.236	0.234	

$$\theta = \theta_r + \frac{\theta_s - \theta_r}{\left[1 + |\alpha h|^n\right]^{1-\frac{1}{n}}}$$

求待定参数： $\alpha, n, \theta_r, \theta_s$

MATLAB 串口通信

1. 创建串口对象

对象变量 = serial(串口号, 'BaudRate', 波特率参数, 'TimeOut', 超时时间, 'Terminator', 数据终止符)
如: sobject = serial('COM13', 'BaudRate', 9600, 'TimeOut', 10, 'Terminator', 'LF') %默认是 8 位数据位,
1 位停止位

串口号: 端口号

波特率参数: 波特率参数

超时时间: 接受超时时间, 这里会出现一个现象, 就是串口助手给 MATLAB 程序发数据, MATLAB 也不会显示, 一定要超时时间到了才显示接受的数据

数据终止符: 如果是字符传输, 用于读取和写入以 ASCII 字符结尾的数据的终止符字符, "LF" (默认)
| "CR" | "CR/LF" | 0 至 255

2. 打开串口对象

fopen(对象变量)

如: fopen(sobject)

3. 串口发送数据

fprintf(对象变量, 要发送的字符串)

如: fprintf(sobject, 'xxxxzzzz')

4. 关闭串口

fclose(对象变量); 关闭串口

如: fclose(sobject)

完整代码

```
>> sobject = serial('COM13', 'BaudRate', 9600, 'TimeOut', 1, 'Terminator', 'LF')
```

```
Serial Port Object : Serial-COM13
```

```
Communication Settings
```

```
Port: COM13
```

```
BaudRate: 9600
```

```
Terminator: 'LF'
```

```
>> fopen(sobject);
>> fprintf(sobject, 'xxxxzzzz')
>> fclose(sobject);
```



字符数据发送成功

注意有时候打开串口会失败

```
>> fopen(sobject);
```

错误使用 serial/fopen (line 72)

打开失败: Port: COM4 is not available. Available ports: COM13.

Use INSTRFIND to determine if other instrument objects are connected

这是 MATLAB 串口的一个 BUG, 所以在创建任何一个串口之前要执行

```
delete(instrfindall) %删除所有串口设备, 执行该函数之后才能做串口相关操作,
```

串口接收数据实现（记住接收字符串形式和接收 16 进制(二进制)形式操作是不一样的）

上一节串口设置的方法 `sobject = serial('COM13','BaudRate',9600,'TimeOut',10,'Terminator','LF')`

实际上不这么设置，这样设置就将串口设置死了，后面修改参数就会报错

```
set(sobject,'inputBufferSize',1024000) %设置输入缓冲区域为 1M
```

```
set(sobject,'BytesAvailableFcnMode','bytes');%设置中断响应函数对象  
set(sobject,'BytesAvailableFcnCount',10);%设置中断触发方式
```

`sobject.BytesAvailableFcn=@ReceiveCallback;` %`ReceiveCallback` 是中断的触发函数，这里我是自定义的。系统的回调函数为 `instrcallback`；

```
delete(instrfindall)  
scom = serial('串口号');  
scom.InputBufferSize =512; %接收数据缓冲区  
scom.OutputBufferSize =512; %发送数据缓冲区  
scom.ReadAsyncMode='continuous';  
scom.BaudRate =115200; %波特率  
scom.Parity ='none'; %无奇偶校验位  
scom.StopBits =1; %停止位 1  
scom.DataBits =8; %数据位 8  
scom.Terminator ='CR'; %如果字符形式传输， CR 表示一帧字符结束  
scom.FlowControl ='none'; %无硬件流控  
scom.timeout =1; %接收超时时间 1 秒，这就可能出现你还没有发数据，接收程序就执行完了。所以无需要死循环执行。  
scom.BytesAvailableFcnMode = 'byte';  
scom.BytesAvailableFcnCount = 1024;  
scom.BytesAvailableFcn = @callback;
```

如果 `BytesAvailableFcnMode` 设置的为 `byte`，则使用 `fwrite`。

如果 `BytesAvailableFcnMode` 设置的为 `byte`，则使用 `fread`

代码案例

```
delete(instrfindall) %清除所有端口，必须先执行  
  
scom = serial('COM4');  
scom.InputBufferSize =512; %接收数据缓冲区  
scom.OutputBufferSize =512; %发送数据缓冲区  
scom.ReadAsyncMode='continuous';  
scom.BaudRate =9600; %波特率  
scom.Parity ='none'; %无奇偶校验位  
scom.StopBits =1; %停止位1  
scom.DataBits =8; %数据位8  
scom.Terminator ='CR'; %如果字符形式传输， CR 表示一帧字符结束  
scom.FlowControl ='none'; %无硬件流控  
scom.timeout = 1; %接收超时时间1秒，如果超过1秒还没有数据就会出现timeout警告  
  
fopen(scom);  
n = 0;  
while n <= 5  
  
    out = fscanf(scom, '%c', 1); %字符形式的数据必须用fscanf来接受  
    fprintf('%3c', out);  
end  
  
fclose(scom);
```

Warnings.
警告：读取未成功：A timeout occurred before the Terminal
'serial' unable to read any data. For more information c

使用 `fscanf` 接收串口字符正常

如果我换成 `fread(...)` 接收呢？

```
scom.timeout = 10; %接收超时时间1秒，如果超过1秒还没有数据就会

fopen(scom);

n = 0;
while n <= 5
    out = fread(scom, 512, 'uint8'); %从串口对象中读取size字节长短的二进
    fprintf('%3x', out);
end

fclose(scom);
```

我串口助手发送数据，没有数据输出

Warnings.

警告：The specified amount of data was not returned within the Timeout period.
'serial' unable to read all requested data. For more information on possible reasons, see [Serial Read Warnings](#).

1 2 3 5 8 4

等待 `TIMEOUT` 之后，接收数据才打印出来，同时会出现警告

这其实是缓存没设置正确造成的

`fread(scom, 512, 'uint8')`

这是因为 `fread` 设置了缓存，必须接收完 512 字节才会跳出阻塞。

```
fopen(scom);

n = 0;
while n <= 5
    out = fread(scom, 1, 'uint8'); %从串口对象中读取size字节长短的二进制数据，以数组形式存于out
    fprintf('%3x', out);
end

fclose(scom);
```



1 2 3 4 5 1 2 3 4 5

发送数据

接收数据正常

这就证明了 `fread` 的缓存区大小是会阻塞的。

记住如果不再使用串口对象要顺序执行以下三条命令

`fclose(scom);` %除了执行 `fclose` 关闭串口外

`delete(scom);` %删除内存中串口设备对象

`clear scom;` %清除工作空间中串口设备对象

串口中断接收数据

```
function serialfb()

delete(instrfindall)    %清除所有端口，必须先执行，怕有其它串口占用

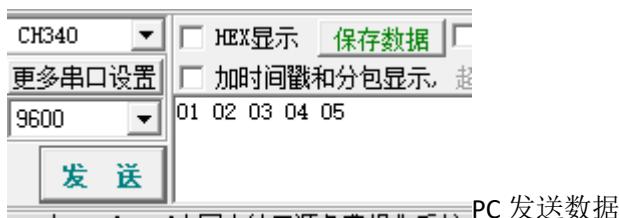
scom = serial('COM4');
scom.InputBufferSize =512; %接收数据缓冲区
scom.OutputBufferSize =512; %发送数据缓冲区
scom.ReadAsyncMode='continuous';
scom.BaudRate =9600;      %波特率
scom.Parity ='none';     %无奇偶校验位
scom.StopBits =1;         %停止位1
scom.DataBits =8;         %数据位 8
scom.Terminator ='CR';    %如果字符形式传输，CR表示一帧字符结束
scom.FlowControl ='none'; %无硬件流控
scom.timeout = 10;        %接收超时时间1秒，如果超过1秒还没有数据就会出现timeout警告
scom.BytesAvailableFcnMode = 'byte'; %设置中断触发方式，那么ByteAvailableFcnMode就要设置为byte模式
scom.BytesAvailableFcnCount=1;   %接收缓冲区每收到1个字节时，触发BytesAvailableFcn定义的回调函数
scom.BytesAvailableFcn=@fback; %fback回调函数自定义
```

/*记住，回调函数一定要和主函数定义在同一个 m 文件下，如果另外新建一个 M 文件定义回调函数，是无法执行的*/

```
function fback(obj,event)

str = fread(obj,1,'uint8'); //obj参数就是传入的串口句柄
fprintf('%3x',str);

end
```



PC 发送数据

```
>> serialfb
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

MATLAB 中断 fback 函数接收到数据

串口中断接收一帧数据，然后对数据里面的数据进行多字节拼接

对这帧数据进行拼接 aa ff ff ff ff ff 01 01 01 01 fe fe fe fe fe 02 02 02 02 fc fc fc fc fc 03 03 03 03 fd fd fd fd fd 04 04 04 04 04 10 10 10 BB

aa ff ff ff ff 01 01 01 01 fe fe fe fe fe 02 02 02 02 fc fc fc fc fc 03 03 03 03 fd fd fd fd fd 04 04 04 04 10 10 10 BB

把aa提取出来判断是不是数据头 把6个ff提取出来拼接成64位的数 把01取出来拼接成32位的数

这就需要把串口的数据，接收成数组，按照下标去取想要的字节

```
[变量1, 变量2]=fread(obj, 接收字节数, 'uint8') //串口fread返回两个变量
```

变量 1：返回接收一帧数据的数组变量

变量 2：返回接收一帧数据的长度

变量 = fread(obj, 接收字节数, 'uint8') //串口返回 1 个变量
变量: 返回的是一帧串口的数据。

变量(0), 这样是无法获取串口数组里面的某个元素的, 运行报错

变量(1), 这样是可以获取串口数组第 1 个字节的数据, 所以 MATLAB 数组是从下标 1 开始, 而不是 C 语言数组的下标 0。

The diagram illustrates the MATLAB code for initializing and reading data from a serial port, comparing two approaches: a sub-file and a main file.

Sub-File (uartinit.m):

```
function uartInit(port)
    global scom;
    delete(instrfindall);
    scom = serial(port);
    scom.InputBufferSize = 512; %接收数据缓冲区
    scom.OutputBufferSize = 512; %发送数据缓冲区
    scom.ReadAsyncMode='continuous';
    scom.BaudRate = 115200; %波特率
    scom.Parity ='none'; %无奇偶校验位
    scom.StopBits =1; %停止位1
    scom.DataBits =8; %数据位8
    scom.Terminator ='CR'; %如果字符形式传输, CR表示一帧字符结束
    scom.FlowControl ='none'; %无硬件流控
    scom.timeout =10; %接收超时时间1秒, 这就可能出现你还没有发数据, 接收
    scom.BytesAvailableFcnMode = 'byte';
    scom.BytesAvailableFcnCount=1;
    scom.BytesAvailableFcn=@fbback;
    fopen(scom);
end
```

Main File (readtimedata.m):

```
function readtimedata()
    uartInit('COM3');
    n = 0;
    while n<=5
        end
    end
```

Command Window Output:

```
>> readtimedata
aa ff ff ff ff ff 1 1 1 fe fe fe fe fe 2 2 2 fc fc fc fc fc 3 3 3 fd fd fd fd fd fd 4 4 4 4 1C
:1
```

Annotations:

- 指向 `recvBuff(8)` 的注释: "这就是获取一帧数据中的第 8 个元素"
- 指向 `recvBuff` 的注释: "这就是接收的一帧数据"
- 指向 `aa` 的注释: "获取第 8 个元素"
- 指向 `recvBuff(1)` 的注释: "获取数组中第 1 个元素"
- 指向 `aa` 的注释: "这就是获取数组中第 1 个元素。记住不能获取第 0 个元素, 因为 MATLAB 里面没有 0 元素"

串口字节拼接

C = Bitor(A, B) %或运算, 将A变量与B变量进行位或, 将位或之后的值返回给C变量

返回值 = Bitsleft(变量, 左移多少位) %左移运算

变量: 需要左移的数

左移多少位: 可以填 1,2,3,4,5,6,7,8. 从左移 1 位~左移 8 位可选

返回值: 左移之后的数据返回给变量

```
recvBuff = fread(obj, 46, 'uint8'); %obj参数就是传入的串口句柄 1次接收46字节数据才跳出阻塞  
fprintf(' %3x', recvBuff);  
fprintf('\n');
```

```
packedNum = recvBuff(1); % aa赋值给变量最低位  
packedNum = bitor(packedNum, bitshift(recvBuff(2), 8)); %ff左移8位或上aa  
packedNum = bitor(packedNum, bitshift(recvBuff(3), 8*2));%ff左移16位或上aa  
packedNum = bitor(packedNum, bitshift(recvBuff(4), 8*3));%ff左移24位或上aa  
  
fprintf('%x', packedNum);
```

```
>> readtimedata
```

```
aa ff ff ff ff ff ff 1 1 1 1 fe fe fe fe fe fe 2 2
```

```
:fffffaa
```

这就是左移之后拼接的数据

再测试一个比较好识别的帧数据

```
>> readtimedata  
aa f1 f2 f3 f4 ff ff 1 1 1 1 fe fe fe fe fe 2 2 2 2 fc fc fc fc fc 3 3 3 3 fd fd fd fd :  
f3f2f1aa
```

确实是左移运算

看看右移运算

其实不叫右移, 就是把后面的数放在最低位

```
packedNum = recvBuff(4); % f3 放在最低位  
packedNum = bitor(packedNum, bitshift(recvBuff(3), 8)); %f2左移8位  
packedNum = bitor(packedNum, bitshift(recvBuff(2), (8*2))); %f1左移16位  
packedNum = bitor(packedNum, bitshift(recvBuff(1), (8*3))); %aa 左移24位
```

```
fprintf('%x', packedNum);
```

```
aaf1f2f3 aa f1 f2 f3 f4 ff ff 1 1 1 1
```

```
aaf1f2f3
```

其实右移很简单, 就是加上负号即可

```
packedNum = bitor(packedNum, bitshift(recvBuff(3), -8)); %f2右移8位
```

```
packedNum = bitor(packedNum, bitshift(recvBuff(2), -(8*2))); %f1右移16位
```

全局变量，两个不同的 m 文件相互调用

```
readtimedata.m  uartinit.m +  
1  
2  
3  
4 function fbback(obj, event)  
5     global gdata;  
6     recvBuff = fread(obj, 46, 'uint8'); %obj参数就是传入的串口句柄 1  
7     fprintf('%3x', recvBuff);  
8     fprintf('\n');  
9  
10    packedNum = recvBuff(4); % f3 放在最低位  
11    packedNum = bitor(packedNum, bitshift(recvBuff(3), 8));  
12    packedNum = bitor(packedNum, bitshift(recvBuff(2), (8*2)));  
13    packedNum = bitor(packedNum, bitshift(recvBuff(1), (8*3)));  
14  
15    fprintf('%x', packedNum);  
16    gdata = packedNum;  
17 end
```

在串口 m 文件中定义的全局变量

```
readtimedata.m  uartinit.m +  
1 function readtimedata()  
2     global gdata;  
3     uartInit('COM4');  
4     A = 25;  
5     n = 0;  
6     while n<=5  
7         disp(A);  
8         fprintf('main = %3x\n', gdata);  
9         pause(1);  
10    end  
11  
12 end  
aa f1 f6 f3 f4 ff ff 1 1 1 1 fe f  
aa f1 f6 f3 aa f1 f6 f3 f4 ff ff 1 1 1  
aa f1 f6 f3 25
```

在主程序 m 文件可以调用其它 m 文件定义的全局变量，获取 uartInit 文件的 gdata 数据

全局变量输出串口 m 文件得到的数据

串口文件收到数据进行拼接，赋值给 gdata 全局变量

主程序获取全局变量 gdata，得到串口数据

全局数组两个不同的 m 文件相互调用

比如我在界面端 m 文件定义了全局数组

```
function rtcurve.CreateFcn(hObject, eventdata, handles)  
global timecount;  
global RTdata;  
timecount=[0 1];  
RTdata=[0 1];  
global p;  
p = plot(timecount, RTdata, 'EraseMode', 'none', 'MarkerSize', 5);
```

先定义全局变量

本来全局变量只是单个变量，但是初始化方式让全局变量变成了全局数组

主程序使用全局数组

```
global timecount;  
global RTdata;  
disp(RTdata); 主程序调用全局数组
```

0 1 输出正常，全局数组就是界面端 m 文件定义的值

MATLAB 动态绘制曲线图

plot(x, y, 填入 EraseMode, 填入 EraseMode 下的参数, 填入 MarkerSize, 填入标记点大小值) %曲线工具多参数设置

设置(EraseMode)刷新模式: EraseMode 有 4 种模式 (但是注意EraseMode 不支持高版本在 2019 版本中将就使用)

normal 模式: 重绘整个显示区, 这种方式产生的图形最准确, 但较慢

none 方式: 不做任何擦除, 直接在原来图形上绘制

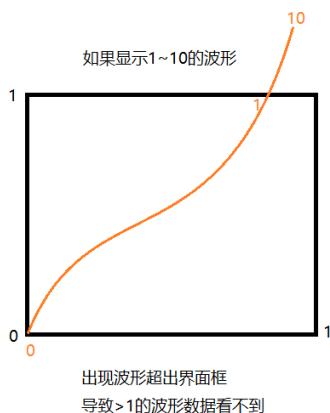
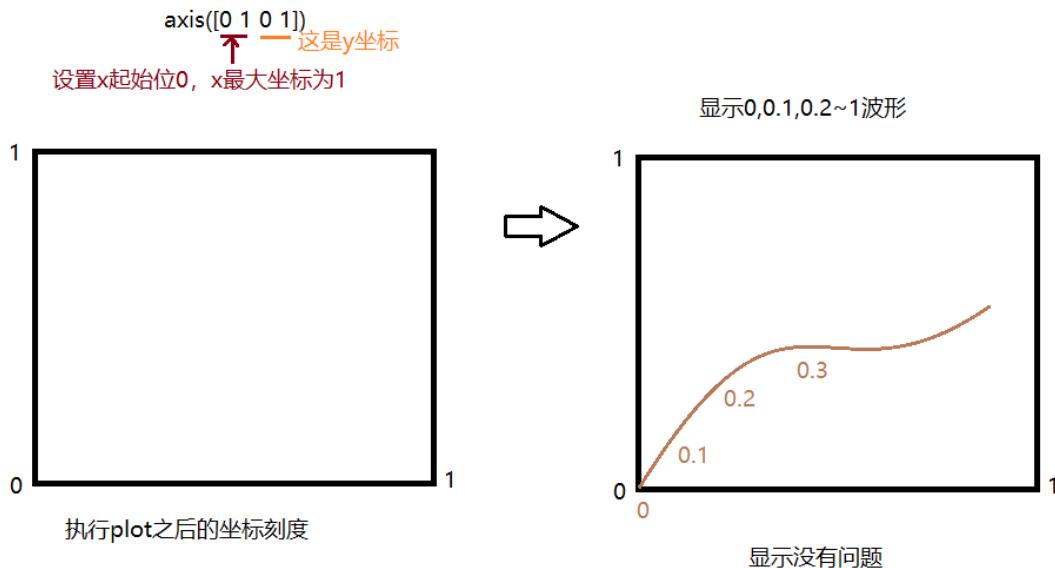
background 方式: 把旧对象的颜色变为背景色, 这种方式影响被擦除对象下面的对象

EraseMode 选择后会出现 EraseMode 属性不再受支持, 而且在以后的版本中会出错, 但是在测试程序将就使用

MarkerSize: 标记点大小用数字表示

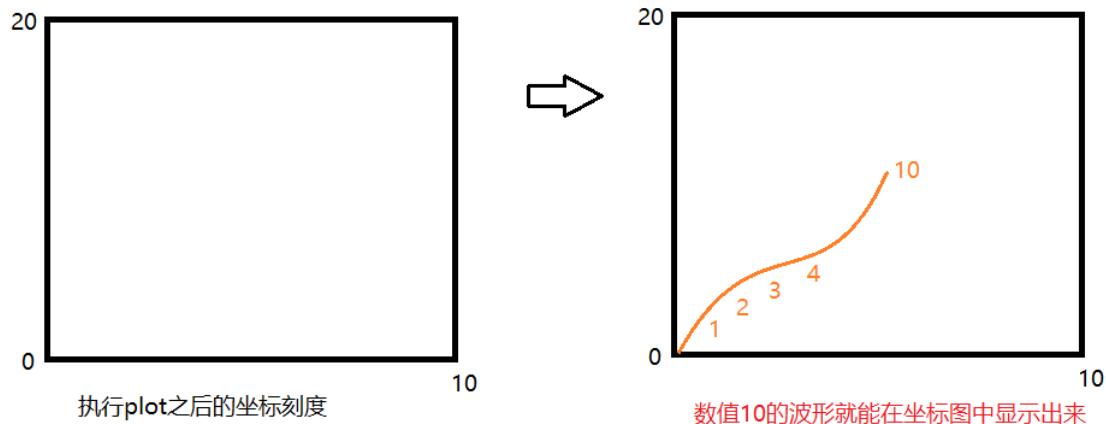
例子: p = plot(x, y, 'EraseMode', 'none', 'MarkerSize', 5);

axis([x轴起始坐标, x轴最大坐标, y起始坐标, y最大坐标]) % 设置网格坐标范围



`axis([0 10 0 20])` 将坐标范围设置更大

如果显示1~10的波形

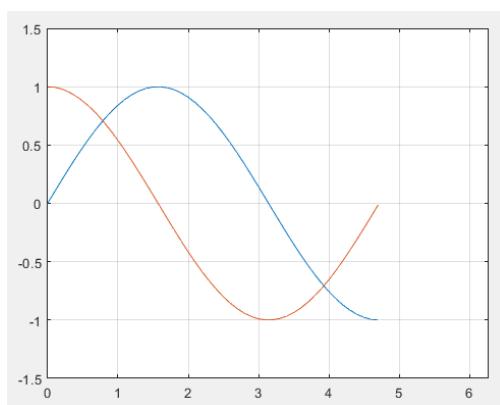


```

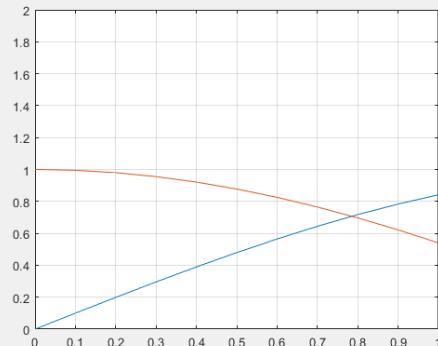
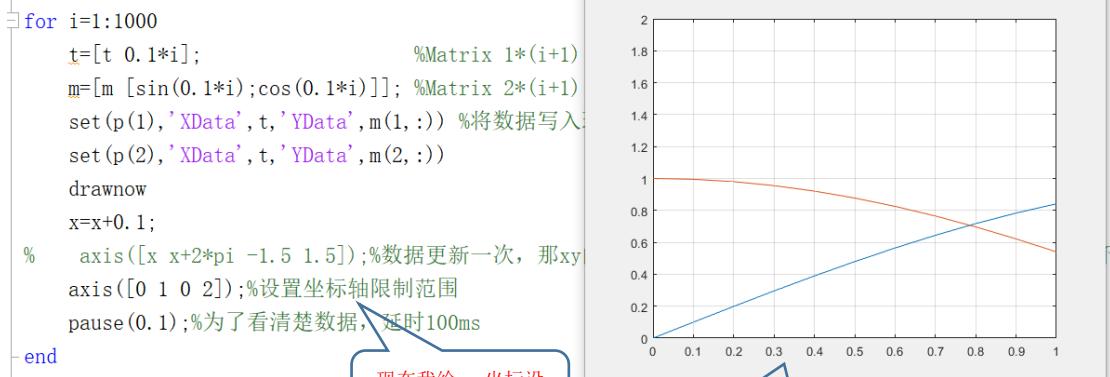
function RTtest() %创建中断响应函数
t=[0]
m=[sin(t);cos(t)]
p = plot(t,m,'EraseMode','none','MarkerSize',5);
x=-1.5*pi;
axis([x x+2*pi -1.5 1.5]);%设置坐标轴限制范围
grid on;

for i=1:1000
    t=[t 0.1*i]; %Matrix 1*(i+1)
    m=[m [sin(0.1*i);cos(0.1*i)]]; %Matrix 2*(i+1)
    set(p(1),'XData',t,'YData',m(1,:)) %将数据写入现有的plot界面, p就是现有plot指针
    set(p(2),'XData',t,'YData',m(2,:))
    drawnow
    x=x+0.1;
    axis([x x+2*pi -1.5 1.5]);%数据更新一次, 那xy的坐标轴也更新, 这样数据就不会超出坐标范围显示不出来
    pause(0.1);%为了看清楚数据, 延时100ms
end
end

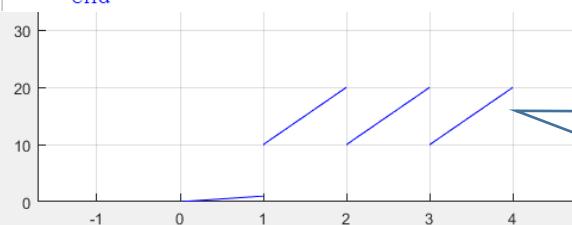
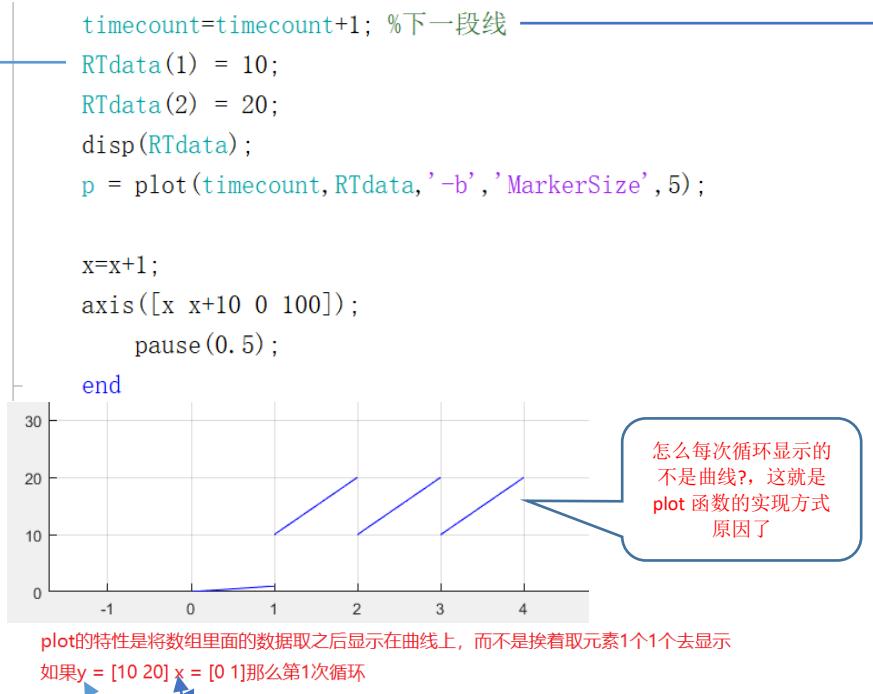
```



数据正常的滑动, xy 坐标也跟着变化

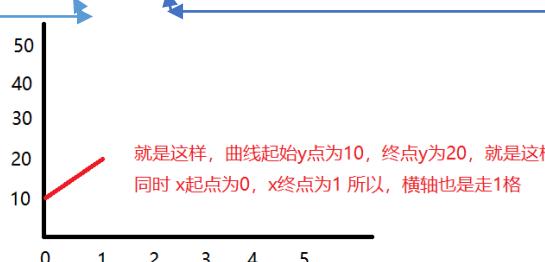


对以上动态绘制曲线的方法进行详细介绍
先说说 x 轴实时平移的问题



怎么每次循环显示的
不是曲线？这就是
plot 函数的实现方式
原因了

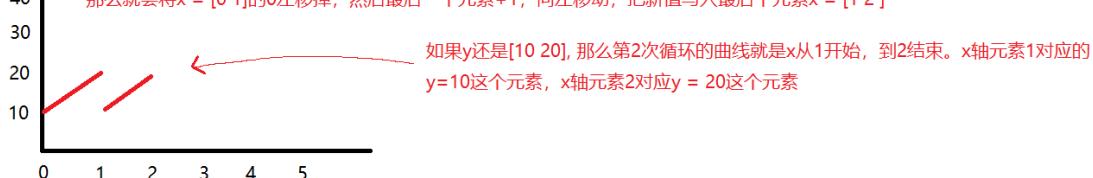
plot 的特性是将数组里面的数据取之后显示在曲线上，而不是挨着取元素1个1个去显示
如果 $y = [10 20]$ $x = [0 1]$ 那么第1次循环



第2次循环 $x=x+1$

如果 x 初始化的时候定义的就是两个元素 $x = [0 1]$

那么就会将 $x = [0 1]$ 的 0 左移掉，然后最后一个元素 +1，向左移动，把新值写入最后那个元素 $x = [1 2]$



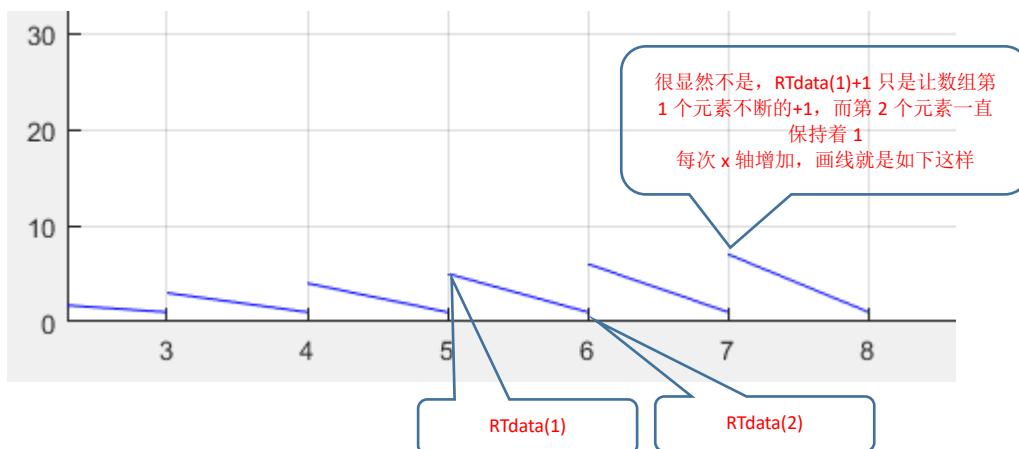
所以 `plot` 是一次性将数组里面的元素显示完，而不是 1 个 1 个挨着数组去显示。

为了更直观，我们让 `RTdata` 变量，也就是 y 轴自动+1

```
hold on  
timecount=timecount+1; %下一段线  
RTdata(1) = RTdata(1)+1;  
  
disp(RTdata);  
p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
```

y 轴数组循环每次+1，会是曲线效果吗？

```
x=x+1;  
axis([x x+10 0 100]);  
pause(0.5);  
end
```



```
hold on  
timecount=timecount+1; %下一段线  
RTdata = RTdata + 1;  
  
disp(RTdata);  
p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
```

这样 `RTdata` 每次循环，最后个元素就会+1，然后最后个元素向前移动

```
x=x+1;  
axis([x x+10 0 100]);  
pause(0.5);  
end
```



但是这又会出现一个新的问题，如果得到的值是变量呢？变量的数字是不会随时变化的

```
DATAV = 0; %定义个变量
```

```
n = 0;
```

```
while n < 5
```

```
    hold on
```

```
    DATAV = DATAV + 1;
```

变量每次循环+1

```
    timecount=timecount+1; %下一段线
```

```
    RTdata = DATAV;%变量传入数组
```

变量传入数组

```
    disp(RTdata);
```

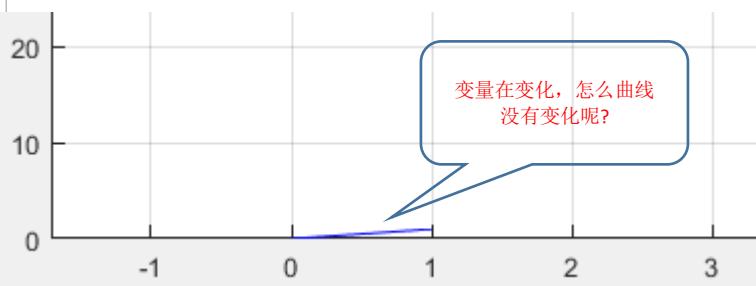
```
    p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
```

```
x=x+1;
```

```
axis([x x+10 0 100]);
```

```
    pause(0.5);
```

```
end
```



因为变量是单个数值，而 RTdata 是数组，只有取数组下标某个元素才能得到变量的值。

```
hold on
```

```
DATAV = DATAV + 1;
```

```
timecount=timecount+1; %下一段线
```

将单个变量赋值
给数组中最后一个
元素

```
RTdata(2) = DATAV;%变量传入数组最后一个元素
```

```
disp(RTdata);
```

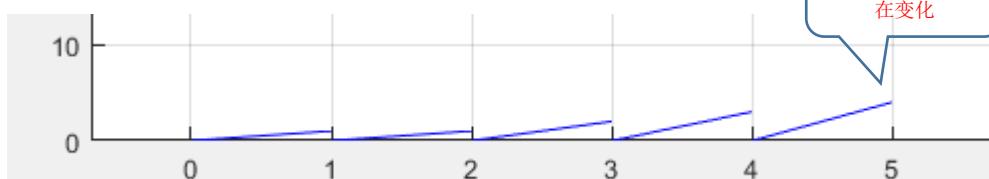
```
p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
```

```
x=x+1;
```

```
axis([x x+10 0 100]);
```

```
    pause(0.5);
```

```
end
```



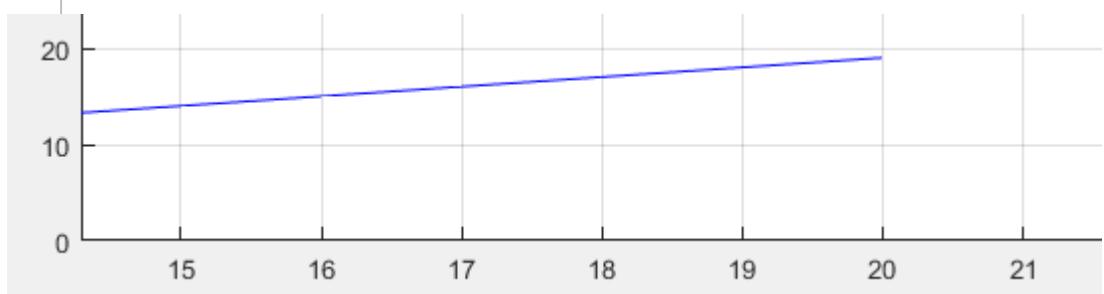
```
RTdata(2) = DATAV;%变量传入数组最后一个元素
```

就是因为赋值的是最后一个元素，
所以曲线只能显示最后元素的变化

```

DATAV = DATAV + 1;
RTdata(2) = DATAV;%将本次变量加的数值放在第2个元素
timecount=timecount+1; %下一段线
p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
RTdata(1) = DATAV; %将上次加的数值放在第1个元素
x=x+1;
axis([x x+10 0 100]);
pause(0.5);
end

```



你看曲线形成了

axis 函数将 x 轴的间距缩小会发生什么?

```
hold on
```

```

DATAV = DATAV + 1;
RTdata(2) = DATAV;%将本次变量加的数值放在第2个元素
timecount=timecount+1; %下一段线

```

```

p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
RTdata(1) = DATAV; %将上次加的数值放在第1个元素

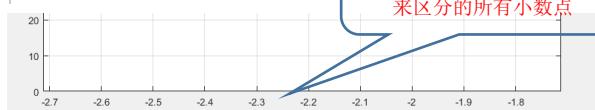
```

```
x=x+1;
```

```
axis([x x+1 0 100]);
```

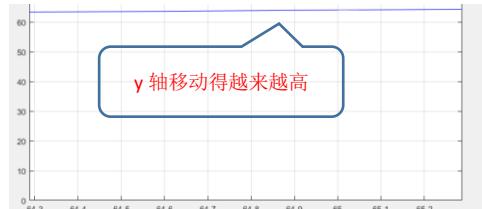
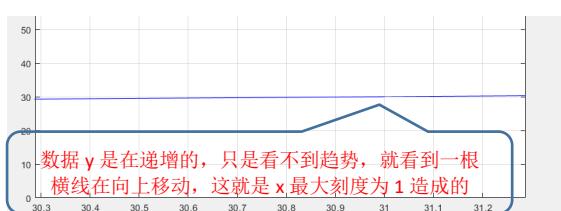
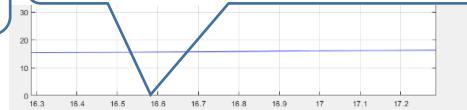
```
pause(0.5);
```

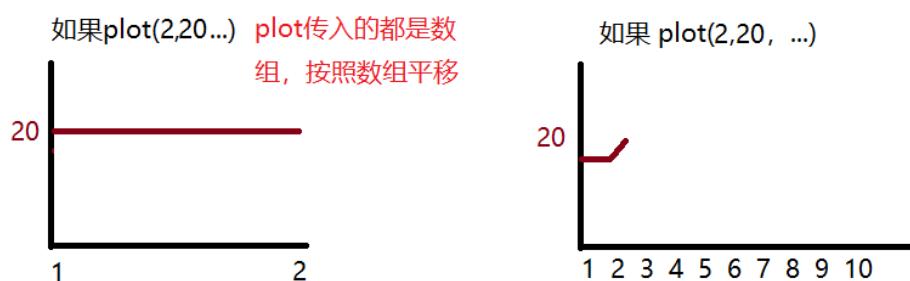
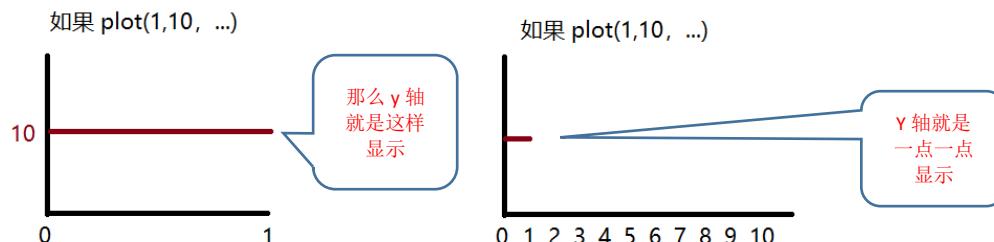
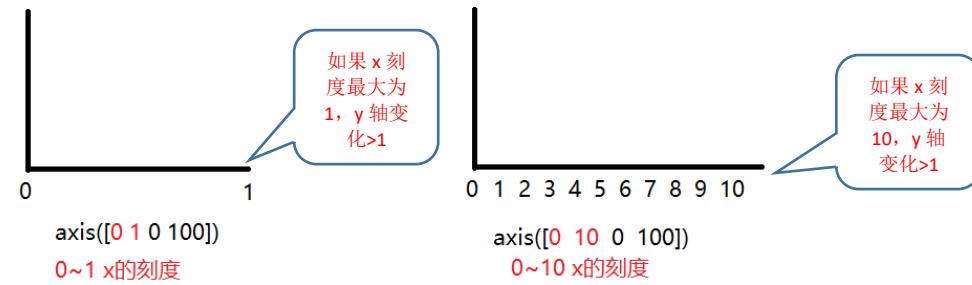
```
end
```



我将 x 轴改成 0 到 1 的刻度

运行发现曲线并没有出现，而且 x 轴是按照 0~1 来区分的所有小数点
因为 y 轴是按照+1 递增，所以每次递增后都会占用 x 整个区间段，因为 x 最大刻度才 1



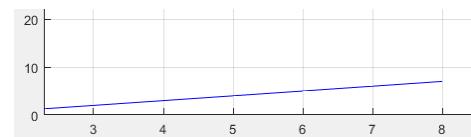


这就是坐标刻度的问题

```
DATAV = DATAV + 1;
RTdata(2) = DATAV;%将本次变量加的数值放在第2个元素
timecount=timecount+1; %下一段线
```

```
p = plot(timecount, RTdata, '-b', 'MarkerSize', 5);
RTdata(1) = DATAV; %将上次加的数值放在第1个元素
x=x+1;
axis([x x+10 0 100]);
pause(0.5);
end
```

X 轴最大
范围为
0~10



问题得到解决

在主循环中无限执行 plot 可能会让绘图不断重复造成卡程序

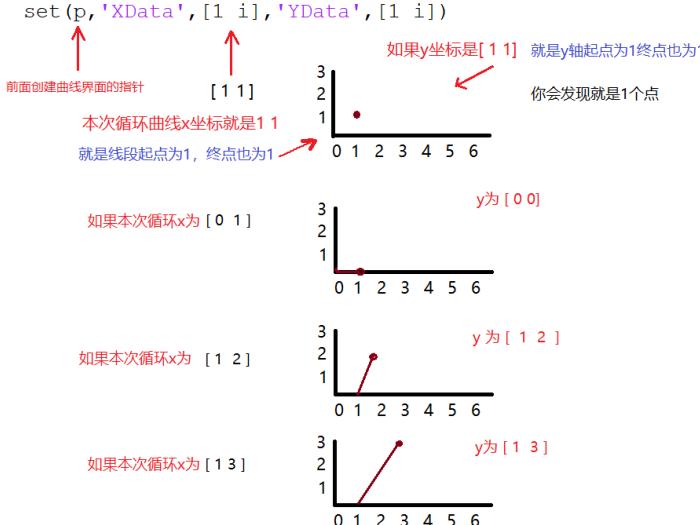
```
x = 0;
y = 0;
p = plot(x,y, 'EraseMode', 'none', 'MarkerSize', 5);%none每次执行set不查除以前的数据点, 直接在现有的图上继续绘制新数据点
axis([x x+1 -1.5 1.5]);
grid on;
```

如果是 GUI 编程, 那么在 GUI 界面中加入了曲线显示区, 就要在 GUI 曲线显示回调函数中创建 plot, 不能在主循环创建, 不然会死机

```

for i=1:1000000
    set(p, 'XData',[1 i], 'YData',[1 i]) %每次循环设置数据点
    axis([x-5 x+5 0 x+5]);
    x=x+1;
    drawnow

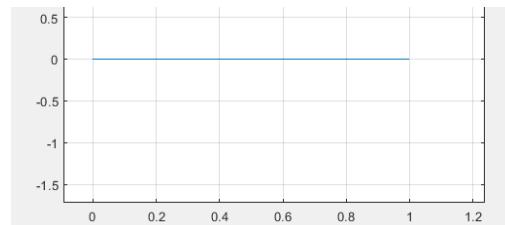
```



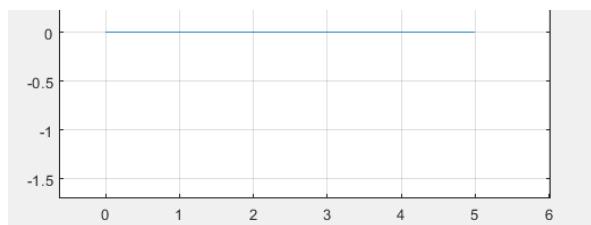
这就是 `set` 设置 `plot` 曲线的方法

实际测试如下：

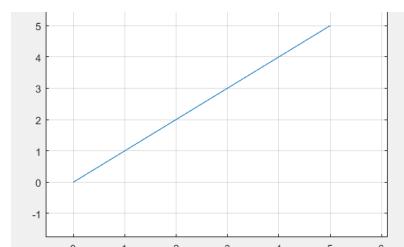
```
set(p, 'XData', [0 1], 'YData', [0 0])
```



```
set(p, 'XData', [0 5], 'YData', [0 0])
```



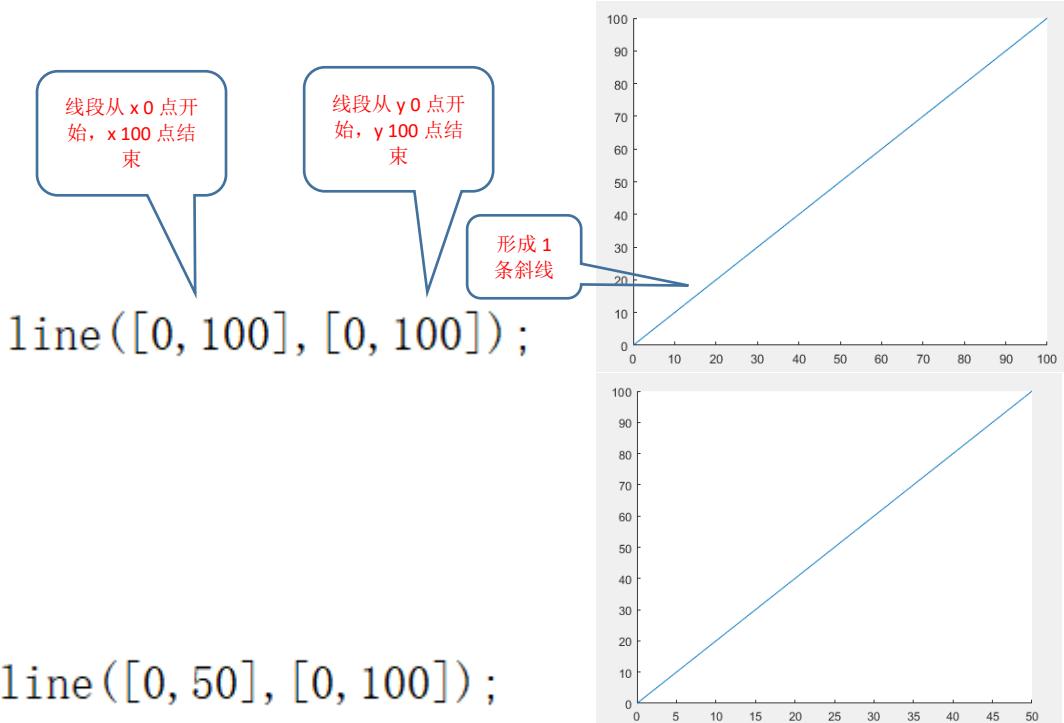
```
set(p, 'XData', [0 5], 'YData', [0 5])
```



这种操作 `plot` 指针变量 `p` 的方式也不能扩大数据点数量，最多 2000 个点。下面用 `line` 绘图，来扩充数据点。

line 在 GUI 绘制曲线中的使用，替代 plot 绘图函数

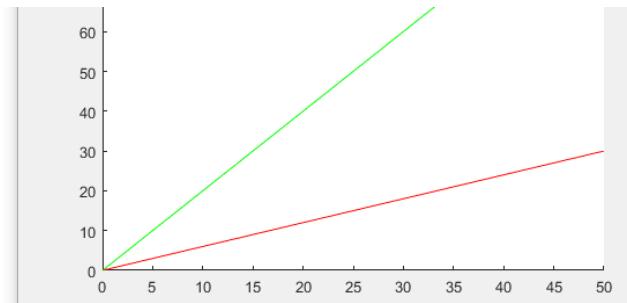
plot 和 line 本质上是有区别的，plot 为高层绘图函数，line 为底层操作函数，line 为绘制曲线对象，plot 是在 line 的基础上编写的，所以 line 绘制效果更好。可以在 GUI 界面中绘制达到 4000 个数据点。但是！如果你每次请求 line 函数，MATLAB 在现有的坐标中画一条线，并不会擦除我们刚才画的那条。这与高级函数不同，如 plot 一样的高级函数，如果我们第二次请求，PLOT 会删除刚才的图像，重置坐标属性（除了 Position 和 Units）



其实和 plot 的 set 很像，set 底层就是 line。

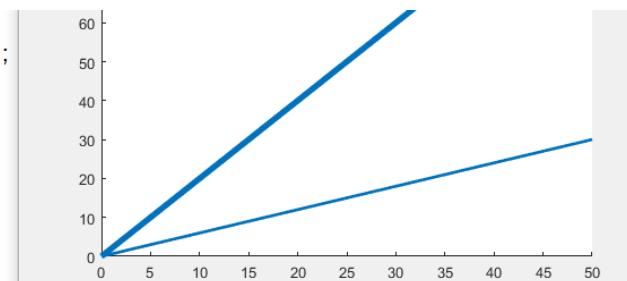
改变指定线条颜色

```
line([0,50], [0,100], 'color', 'g');  
line([0,50], [0,30], 'color', 'r');
```



改变线条粗细

```
line([0,50], [0,100], 'linewidth', 4);  
line([0,50], [0,30], 'linewidth', 2);
```



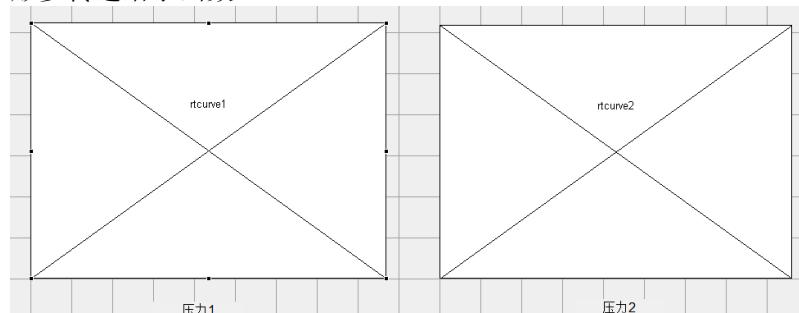
line 实时曲线显示实现

```
timecount = 0;
YdataOld = [0];
xdataOld = [0];
simpleData1 = 0; %采集的数据放入该变量

n = 0;
while n < 5 %死循环
    timecount = timecount+1;
    xdataOld = [xdataOld timecount];
    YdataOld = [YdataOld simpleData1]; %将旧数据和新数据放入数组
    line([xdataOld,timecount],[YdataOld,simpleData1]); %曲线显示
    axis([x-5 x+5 0 10]); %坐标格子移动
    xdataOld = timecount;
    YdataOld = simpleData1;
    x=x+1;
    pause(0.1);
end
```

多个 GUI 曲线窗口，如何实现每条数据线对应一个窗口

在 GUI m 文件中，有一个全局的变量，叫 **handles**。该变量只在 GUI 文件中全局。可以经过形参传递给子函数。



我建立了两个曲线窗口，分别显示不同的两个曲线。

此类型的变量不支持使用点进行索引。

```
出错 untitled>rtcurve1_CreateFcn (line 239)
plot(handles.rtcurve1, 0, 0, 'b-', 'LineWidth', 1.5); %压力1 窗口调用自身handles.rtcurve1句柄来绘制曲线
```

运行报错。这是因为压力 1 窗口回调函数不能自身调用自己的 **handles** 句柄。必须交给其它函数去调用自己的 **handles** 句柄。

```
function simpleButton_Callback(hObject, eventdata, handles)
global gsavefilepath;
global openflag; %打开串口全局标志
openflag = 1;

plot(handles.rtcurve1, 0, 0, 'b-', 'LineWidth', 1.5);
hold on;

plot(handles.rtcurve2, 0, 0, 'b-', 'LineWidth', 1.5);
hold on;
```

我在其它回调函数中调用压力 1 窗口的句柄就没有问题

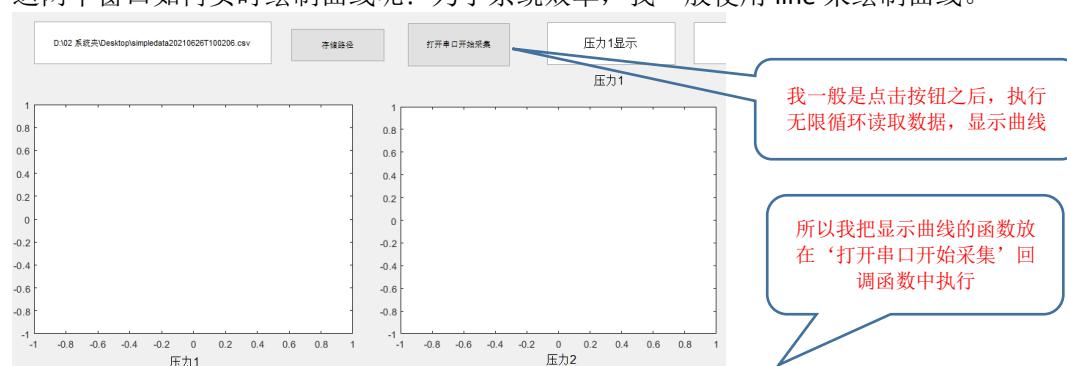
```

plot(handles.rtcurve1, 0, 0, 'b-', 'LineWidth', 1.5); %该函数就是将数据绘制在压力窗口1.
plot(handles.rtcurve2, 0, 0, 'b-', 'LineWidth', 1.5); %该函数就是将数据绘制在压力窗口2.

```

这就是指定绘制曲线到某个窗口的方法。

这两个窗口如何实时绘制曲线呢？为了系统效率，我一般使用 `line` 来绘制曲线。



```

]function simpleButton_Callback(hObject, eventdata, handles)
global gsavefilepath;
global openflag; %打开串口全局标志
openflag = 1;

fprintf("OPEN UART SIMPLE\n");
- readtimedata(gsavefilepath, handles);

```

现在来看看 `readtimedata` 函数具体实现

```

function readtimedata(filepath, handle)
uartInit('COM47');

YdataOld = [0];
YdataOld2 = [0];
xdataOld = [0];
n = 0;
while n < 5
    simpleData1 = Pressure1Data;
    simpleData2 = Pressure2Data;
    timecount = timecount+1; %下一段线

```

这个函数就是显示读取串口数据，将数据显示在线上的函数，因为 `handles` 无法外部文件调用，只有用传参的方式让外部文件调用

```

xdataOld = [xdataOld timeout];
YdataOld = [YdataOld simpleData1];
YdataOld2 = [YdataOld2 simpleData2];

line(handle.rtcurve1, [xdataOld, timeout], [YdataOld, simpleData1]);
line(handle.rtcurve2, [xdataOld, timeout], [YdataOld2, simpleData2]);

handle.rtcurve1.XLim = [10 10]; %设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
handle.rtcurve1.YLim = [10 10];
hold(handle.rtcurve1, 'on'); % rtcurve1 曲线窗口 hold on
handle.rtcurve2.XLim = [10 10];%设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
handle.rtcurve2.YLim = [10 10];
hold(handle.rtcurve2, 'on'); % rtcurve2 曲线窗口 hold on

```

这儿会出现‘值必须是数值类型的 1x2 向量，其中第二个元素大于第一个元素，并且可能为 Inf’ 错误，因为坐标格的 x 起始为 10，终止为 10，根本就不合理

修改后如下：

```

handle.rtcurve1.XLim = [0 10]; %设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
handle.rtcurve1.YLim = [0 10];
hold(handle.rtcurve1, 'on'); % rtcurve1 曲线窗口 hold on
handle.rtcurve2.XLim = [0 10];%设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
handle.rtcurve2.YLim = [0 10];
hold(handle.rtcurve2, 'on'); % rtcurve2 曲线窗口 hold on

```

错误使用 `line`

也有可能出现 向量长度必须相同。

这是因为 `line` 的 y 轴用的是数组

```

YdataOld = [0];
YdataOld2 = [0];
xdataOld = [0];
而你的 timeout 用的是变量值 global timeout;

```

```

timeout = [0];
YdataOld = [0];
YdataOld2 = [0];
xdataOld = [0]; 将 line 使用的 timeout 改成数组就 ok 了

```

代码整体演示

```

x = 0; %x必须定义初值
timeout = [0];
YdataOld = [0];
YdataOld2 = [0];
xdataOld = [0];

```

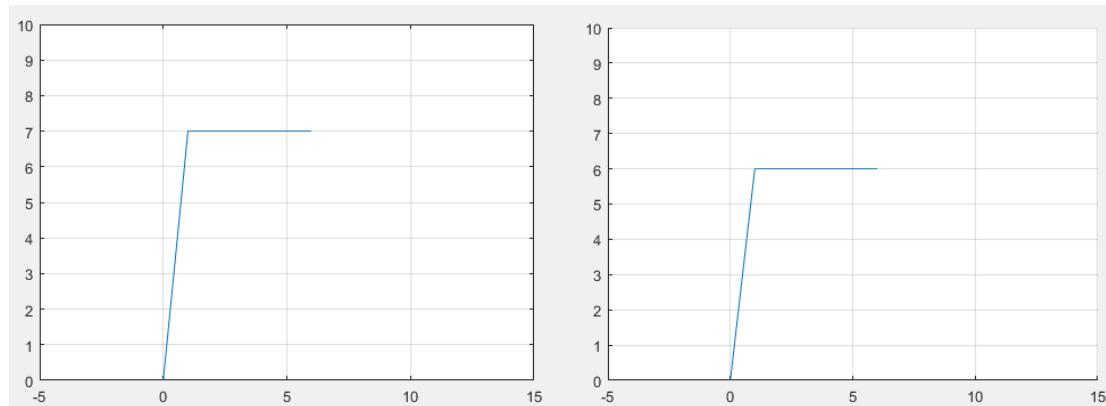
```

n = 0;
while n < 5
    simpleData1 = Pressure1Data;
    simpleData2 = Pressure2Data;
    timecount = timecount+1;
    xdataOld = [xdataOld timecount];
    YdataOld = [YdataOld simpleData1];
    YdataOld2 = [YdataOld2 simpleData2];

    line(handle.rtcurve1,[xdataOld,timecount],[YdataOld,simpleData1]); %在窗口1显示曲线1
    line(handle.rtcurve2,[xdataOld,timecount],[YdataOld2,simpleData2]); %在窗口2显示曲线2
    handle.rtcurve1.XLim = [x-10 x+10]; %设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
    handle.rtcurve1.YLim = [0 10];
    hold(handle.rtcurve1,'on'); % rtcurve1 曲线窗口 hold on
    handle.rtcurve2.XLim = [x-10 x+10]; 设置x轴坐标,每次都要进行移动,这样才能显示实时曲线
    handle.rtcurve2.YLim = [0 10];
    hold(handle.rtcurve2,'on'); % rtcurve2 窗口 hold on

    xdataOld = timecount;
    YdataOld = simpleData1;
    YdataOld2 = simpleData2;
    x=x+1; %移动x轴表格
    pause(0.1);
end

```



MATLAB 定时器使用

定时器主要用于指定时间到了之后，去执行一些自己需要执行的 GUI 回调函数。

```
global num;
num = 0;
stop(timerfind);
delete(timerfind); % 删除现有的定时器，使用定时器之前一定要先执行
t=timer('StartDelay',1,'TimerFcn',@t_TimerFcn,'Period',0.1,'ExecutionMode','fixedRate'); %设置定时器
start(t); %启动定时器
n = 0;
while n < 5
end

function t_TimerFcn(hObject,eventdata) %定时时间到回调函数
    global num;
    num = num + 1;
    fprintf('num = %d\n',num);
end
```

不熟悉 MATLAB? 参阅有关 [快速入门](#) 的资源

```
num = 791
num = 792
num = 793
num = 794
num = 795
num = 796
num = 797
num = 798
```

参数详解

'StartDelay' : 定时器函数开始和第一次调度之前的延迟。 这里是 1 秒
'TimerFcn' : 定时时间到执行的回调函数，这里用@指定自定义函数 t_TimerFcn
'Period' : 定时时间，这是真正意义上的定时时间间隔设置，所以设置定时时间，在这里设置。我设置的 0.1 秒执行一次回调
'ExecutionMode' : 时间事件的调度方式

我发现这个定时器程序结束了，后台还在定时器计数，关不了，怎么办？

```
stop(t)
delete(t)
```

我现在主函数加入 stop 之后，再次运行程序，可以把第一次运行的定时器关闭了。

定时器在 GUI 界面的 m 文件中使用

```
handles.timer = timer; %创建定时器，handles.timer 是GUI界面固定变量
set(handles.timer,'ExecutionMode','FixedRate'); %定时模式
set(handles.timer,'Period',1); %定时1秒
set(handles.timer,'TimerFcn',{@timedisp,handles});%定时1秒之后触发回调，回调函数就是@指定的函数
start(handles.timer);%启动定时器

function timedisp(hObject,eventdata,handles) %定时时间到执行该回调
fprintf("timer ++ \n");
```

在 GUI 界面
中，定义回调
函数不需要加
end

MATLAB 向 Excel 写数据

1. 指定 excel 文件创建的路径 变量 = 'D:\MATLABExcel\myexcel.xlsx' % 存放在 E 盘指定目录下，excel 文件名为 myexcel.xlsx

```
filepath =  
  
'D:\MATLABExcel\myexcel.xlsx'
```

2. 自定义个数组 A = [10,20,30,40,50,60,70,80,90,100]

```
>> A = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]  
  
A =  
  
10 20 30 40 50 60 70 80 90 100
```

3. xlswrite(上面定义的路径变量，传入写入 excel 的数组变量，指定写入 excel 的位置) % 向 excel 写数据

```
>> xlswrite(filepath, A, 'E2:K2')
```

代码演示

```
>> filepath = 'D:\MATLABExcel\myexcel.xlsx'  
  
filepath =  
  
'D:\MATLABExcel\myexcel.xlsx'  
  
>> A = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]  
  
A =  
  
10 20 30 40 50 60 70 80 90 100
```

```
>> xlswrite(filepath, A, 'E2:K2')  
在指定目录下产生了 excel
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2					10	20	30	40	50	60	70		
3													

发现数据没有写完，这是因为只指定了数据写入位置是 excel 的 E2 到 K2
xlswrite(filepath, A, 'E2:K2')
那么数组就按照从 E2 开始到 K2 结束这样排列

如果我把结束位置改到 Q2 呢，excel 写入数据更长呢？

```
>> xlswrite(filepath, A, 'E2:Q2')  
错误使用 xlswrite (line 224)  
文件 D:\MATLABExcel\myexcel.xlsx 不可写。它可能被其他进程锁定。
```

该错误是因为你打开了 myexcel.xlsx 文件，在 MATLAB 写 excel 的过程中，不能在 windows 上手动打开 excel 文件。现在关闭该 excel 文件。

```
>> xlswrite(filepath, A, 'E2:Q2') 执行成功
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																	
2				10	20	30	40	50	60	70	80	90	100	#N/A	#N/A	#N/A	
3																	
4																	

你看数据写入完整。

如果在 `xlswrite` 写 excel 过程中出现服务器出现意外情况，这不是网络问题

```
>> xlswrite(filepath, A, 'E2:E10');
```

错误使用 `xlswrite` (line 224)

错误：服务器出现意外情况。

MATLAB 操作 excel 是单机行为。这种意外情况可能是你已经打开过 excel 了。唯一的办法就是将你创建的 excel 打开，然后再关闭。

自定义数据格式写入 excel 文件

```
>> filepath = 'D:\MATLABExcel\myexcel.xlsx'
```

filepath =

'D:\MATLABExcel\myexcel.xlsx'

创建 excel 文件路径

```
>> B = {'Time', 'Temperature'; 12, 10; 13, 20; 14, 3}
```

B =

excel 第 1 列是 time

excel 第 2 列是 time

指定格式写入 excel

12 对应的是 Time 段数据，‘，’
逗号之后的 10 对应的是
Temperature 数据，然后用；分号
隔离，表示第 1 行数据写完。
13,20 是第 2 行数据，以此类推

4×2 cell 数组

```
{'Time'}    {'Temperature'}  
{[ 12]}     {[      10]}  
{[ 13]}     {[      20]}  
{[ 14]}     {[        3]}
```

```
>> xRange = 'E1'
```

从 excel, E1 这一列开始写

xRange =

'E1',

写入数据到 sheet2

```
>> xlswrite(filepath, B, 'sheet2', xRange)
```

警告：已添加指定的工作表。

```
> In xlswrite>activate_sheet (line 298)
```

```
In xlswrite/ExecuteWrite (line 264)
```

```
In xlswrite (line 218)
```

D	E	F	G
Time	Temperature		
12	10		
13	20		
14	3		

数据按照
指定格式
进行排列

MATLAB 连续写数据到 excel，做实时采集数据存储

返回值(字符数组) = num2str(变量) %将数组转换为表示数字的字符数组

返回字符串 = int2str(变量) %将十进制转换成字符串 就算填入 ff 转换出来也是 255 显示

MATLAB 没有将 16 进制 0xff 转换成字符串'0xff'显示的方法，只有十进制显示将就看看。

用 for 连续写入数据来模拟实际情况

xlswrite(文件路径, 写入 excel 的变量, 写入 excel 哪一页, 写入页中某一格)

```
filepath = 'D:\02 系统夹\Desktop\myexcel2.xlsx';  
for n=1:10  
    xlswrite(filepath, n, 'sheet1', ['A', num2str(n)]);  
end
```

	A	B
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

实时写入 excel 成功，但是我发现
写这 10 个数据到 excel 很慢。不
是瞬间执行完程序

写入 'sheet1' 页, 第 A
列, 第 n 行, 每次循环
得到一个 n 行

xlswrite 写 excel 很慢是正常的，因为该函数是每调用一次就要执行，打开 excel 和关闭 excel 这两个操作，所以就很慢，而且上面例程是循环 10 次，那么就要打开关闭 10 次 excel。

可以先建立个数组，然后数组里面的元素存放到某个下标之后，一次写入 excel

一维数组使用

变量 = [] %这样就建立了一维数组，和 C 语言不同的时候，建立一维数组不需要指定大小，向里面写数据就是

变量(1) = 10 %这就是一维数组写数据的方法，不是像 C 语言那样用“变量[1] = 10”，这种写法。而是用括号来代替下标，“变量(1) = 10”这样写就对了

变量(2) = 20

>> xiang = []	>> xiang(1) = 10	>> xiang(2) = 20
xiang =	xiang =	xiang =
[]	10	10 20

这就是一维数组写数据的方法。

```
>> ret = xiang(2)
```

```
ret =
```

20

获取一维数组数据也简单，直接()在等号右边就可以了

MATLAB 全局变量使用注意事项

```
function exceltest()
    global x0;
    global filepath;
    x0 = 0;
    filepath = 'D:\02 系统夹\Desktop\myexcel2.xlsx';
end
```

在一个文件中定义全局变量，不能马上赋值，global 只能定义，定义之后，单独使用全局变量赋值，这点和 C 语言不一样

```
|function serialfb(obj, event)
|    global x0;
|    global filepath;
|    x0 = x0+1;
|    ReadData = fread(obj, 1, 'uint8');
|    fprintf('%3x', ReadData);
|    xlswrite(filepath, ReadData, 'sheet1', ['A', num2str(x0)]);
end
```

另一个文件使用全局变量，也不能马上赋值，必须重复定义一遍

然后才可以赋值，记住，虽然定义了两遍全局变量，但是他们还是操作的同一个变量

串口实时采集数据存入 Excel 案例(全局变量的使用)

```
function exceltest()
    global x0;
    global filepath;
    x0 = 0;
    filepath = 'D:\02 系统夹\Desktop\myexcel2.xlsx';
    delete(instrfindall) %清除所有端口，必须先执行，怕有其它串口占用
    scom = serial('COM47');
    scom.InputBufferSize = 512; %接收数据缓冲区
    scom.OutputBufferSize = 512; %发送数据缓冲区
    scom.ReadAsyncMode='continuous';
    scom.BaudRate = 9600; %波特率
    scom.Parity ='none'; %无奇偶校验位
    scom.StopBits =1; %停止位1
    scom.DataBits =8; %数据位 8
    scom.Terminator ='CR'; %如果字符形式传输，CR表示一帧字符结束
    scom.FlowControl ='none'; %无硬件流控
    scom.timeout = 10; %接收超时时间1秒，如果超过1秒还没有数据
    scom.BytesAvailableFcnMode = 'byte'; %设置中断触发方式，那么ByteAvailableFcnCount=1; %接收缓冲区每收到1个字节时，触发BytesAvailableFcn=@serialfb;
    scom.BytesAvailableFcn=@serialfb;
    fopen(scom); %打开串口

    n = 0;
    while n <= 5
        end

        fclose(scom);
        delete(scom);
        clear scom;
end
```

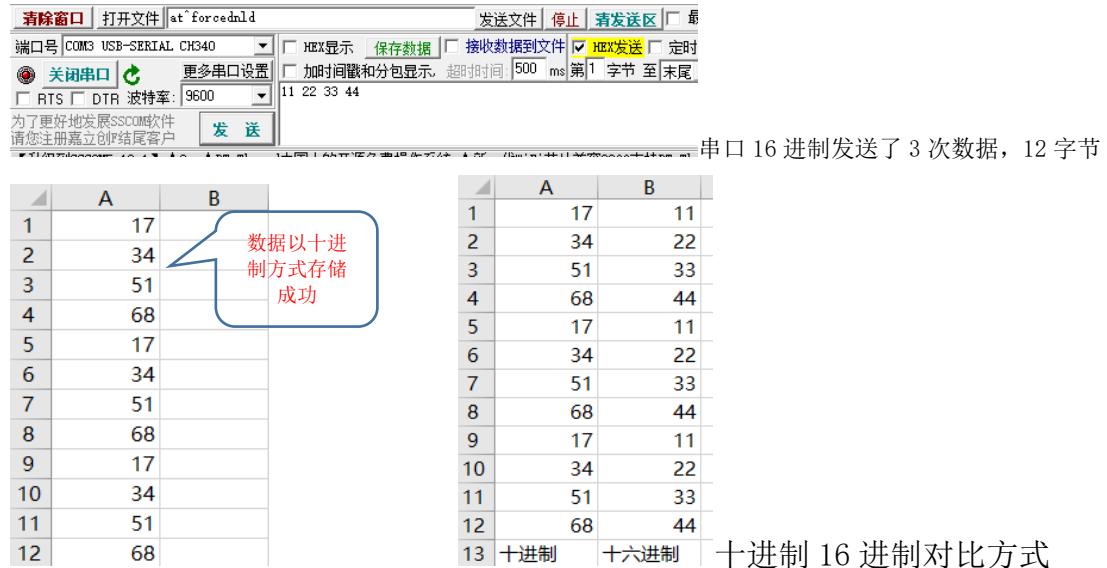
定义全局变量

```

function serialfb(obj, event)
    global x0;
    global filepath;
    x0 = x0+1;           操作全局变量
    ReadData = fread(obj, 1, 'uint8');
    fprintf('%3x', ReadData);
    xlswrite(filepath, ReadData, 'sheet1', ['A', num2str(x0)]);
end

```

看起来写得很慢，但是还是成功写入 excel 了的



将串口数据放入缓存，然后再存入 excel 数据(未完成)

MATLAB 读取 excel 数据进行绘图

返回数据 = xlsread('excel 文件', excel 文件页) %读取 excel 数据

excel 文件: 传入需要读取的.xlsx 后缀 excel 文件

excel 文件页: 就是指定 excel 文件里面的 Sheet

返回数据: 如果 excel 文件数据是 2 列，就返回 2 列数据，如果 excel 文件是 3 列就返回 3 列数据以此类推



MATLAB 获取本地时间

返回值 = `datetime` % 基本获取时间函数

```
>> datetime
n = 0;
ans =
while n <= 5
    xtime = datetime;
    fprintf("%s\n", xtime);
    pause(1);
end
2021-05-05 13:10:38
2021-05-05 13:10:39
2021-05-05 13:10:40
2021-05-05 13:10:41
```

但是这种字符形间隔的时间 2021-05-05 13:10:38 是无法写入 excel 或者 csv 的，而且也不好解析

返回值 = `datestr(变量)` % 将日期和时间按照指定的格式转成成字符形式

变量：传入得到的时间变量，比如用 `now` 生成时间传递给 `datestr`，如：`datestr(now)`

返回值：返回时间字符形式

```
>> datestr(now)
ans =
'06-May-2021 20:06:31' 这就是字符形式的时间, 我用 datestr 主要用来做时间戳看下面
```

`datestr(now,'yyyyymmddTHHMMSS')` % 将时间变成字符串排列的时间戳

```
30 (ISO 8601) 'yyyyymmddTHHMMSS' 20000301T154517
```

`Datestr` 指定格式为 '`yyyyymmddTHHMMSS`' 就是要求把时间做出时间戳的样子，只是不是真正意义上的时间戳，而是把时间和日期连接起来

```
>> datestr(now, 'yyyyymmddTHHMMSS')
ans =
'20210506T200206' 我们可以把这个字符串形式的实际保存在 excel
```

字符串写入 excel 注意事项

```
for x = 1:3 % 循环3秒, 写入3秒时间
    datatime = datestr(now, 'yyyyymmddTHHMMSS'); % 获取当前时间转成类似时间戳
    disp(datatime);
    xlswrite('datetimexls.xlsx', datatime, 'sheet1', ['A', num2str(x)]);
end
```

```
20210506T204233
20210506T204234
20210506T204235
```

:>> 程序运行

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	2	0	2	1	0	5	0	6 T	2	0	4	2	3	3	
2	2	0	2	1	0	5	0	6 T	2	0	4	2	3	4	
3	2	0	2	1	0	5	0	6 T	2	0	4	2	3	5	

我发现写入 excel 的时间是正确的，但是就是每个时间字符放一个单元格，不好用。

这是因为 excel 需要元胞数组才认为是一个单元格，那么我们必须将字符串转换成元胞

```
function IOCTL()

for x = 1:3
    datatime = datestr(now, 'yyyymmddTHHMMSS');
    disp(datatime);
    xlswrite('datetimexls.xlsx', {datatime}, 'sheet1', ['A', num2str(x)]);
    pause(1);
end
```

```
end
>> IOCTL
```

```
20210506T204922
```

```
20210506T204923
```

```
20210506T204924
```

```
>> 程序运行
```

A	
1	20210506T204922
2	20210506T204923
3	20210506T204924

你看，字符串就写入 1 个单元格了

这个写入的时间戳 20210506T204922 里面有个 T，很难用来做运算。

我们用字符串分隔方式，分隔出我们的时间。

字符串分隔

```
>> a='123456789' 定义个字符串
```

```
b =
```

```
>> b=a(1:4) 截取字符串 1234 , 1234'
```

```
c =
```

```
>> c=a(5:9) 截取字符串 56789 , 56789'
```

可以将字符串分隔用在时间写入 excel 应用中，消除 T 字符

```
|for x = 1:3
    datatime = datestr(now, 'yyyymmddTHHMMSS');
    disp(datatime);
    xstr = datatime(10:15);
```

将 T 后面的字符获取出来，T 是第 9 个，从第 10 个开始截取

```
    xlswrite('datetimexls.xlsx', {xstr}, 'sheet1', ['A', num2str(x)]);
    pause(1);
end
```

再用元胞数组写入 excel

```
>> IOCTL
```

```
20210506T205851
```

```
20210506T205853
```

```
20210506T205854
```

	A	B
1	205851	
2	205853	
3	205854	
4		

你看，取出的时间数据，可以做运算。

时间戳如何写入 CSV 文件？请阅读下面 CSV 章节

MATLAB 读写 txt 文件(解决 xlswrite 写 excel 速度慢的问题)

句柄 = fopen('输入文件名.txt', 权限) %打开文件

输入文件名: 假如不指定文件路径, 如果当前 MATLAB 运行软件的桌面没有要打开的文件, 那就会自动创建一个打开的文件。该文件路径就和 MATLAB 运行软件在同一级目录下。比如运行软件在桌面, 那么打开的文件也自动创建在桌面

权限: 对打开的文件是读, 还是写。'r' 为读, 'w' 为写。

```
>> fid = fopen('abc.txt', 'w')
```

fid =

3 返回句柄 3

返回写入多少个数据 = fprintf(句柄, 格式, 数据) %向文件写数据是用 fprintf 来操作

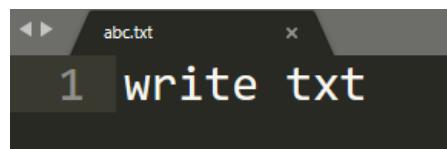
句柄: 就是 fopen 返回的句柄变量

格式: 有 %s, %d, %l, %f 就是指定写数据的类型, 有字符串, 整形, 长整形, 浮点数。

数据: 写入 txt 文本的内容

```
>> fprintf(fid, '%s', 'write txt')
```

ans =



写入数据成功

fclose(句柄) %文件操作完之后一定要执行 fclose

```
>> fclose(fid)
```

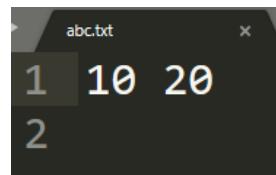
换行写入字符串

```
fid = fopen('abc.txt', 'w')  
fprintf(fid, '%c\r\n', 'write txt')  
fclose(fid)
```



让 txt 文本每一列数据排列整齐

```
function IOCTL()  
a = 10;  
b = 20;  
fid = fopen('ABC.txt', 'w');  
fprintf(fid, '%d %d\r\n', a, b);  
fclose(fid);  
end
```



看起好像没什么问题

```

function IOCTL()
a = 1;
b = 2;
fid = fopen('ABC.txt', 'w');
for x = 1:20
    a=(a+1)*x;
    b=(b+1)*x;
    fprintf(fid, '%d %d\r\n', a, b);
end
fclose(fid);
end

```

我循环增加每个数据的位数

88	112
445	565
2676	3396
18739	23779
149920	190240
1349289	1712169
13492900	17121700

发现数据不是整齐排列了，这是因为数据位数在变化

```

function IOCTL()
a = 1;
b = 2;
fid = fopen('ABC.txt', 'w');
for x = 1:20
    a=(a+1)*x;
    b=(b+1)*x;
    fprintf(fid, '%5d %5d\r\n', a, b);
end
fclose(fid);
end

```

这种%5d就是向右空出5格，给多余位数的数据填充，简称右对齐

2	3
6	8
21	27
88	112
445	565
2676	3396
18739	23779
149920	190240
1349289	1712169

在5位数数据范围内的都能右对齐

超出5位数范围就无法对齐，当然你可以填%10d来做10位数右对齐

```

for x = 1:20
    a=(a+1)*x;
    b=(b+1)*x;
    fprintf(fid, '%-5d %-5d\r\n', a, b);
end

```

%-5d左对齐
5位，一般都用左对齐

2	3
6	8
21	27
88	112
445	565
2676	3396
18739	23779
149920	190240
1349289	1712169

而且操作txt文本的方式就是用 fopen 系列的文件 IO，程序运行速度很快，没有任何延时的感觉，不像 xlswrite 操作 excel 那样延时严重。

读取 txt 文本多列多行数据

返回几 x 几矩阵 = textscan(句柄, 列格式变量) %读取 txt 文本多行多列数据

1	2	3
2	6	8
3	21	27
4	88	112
5	445	565
6	2676	3396
7	18739	23779
8	149920	190240
9	1349289	1712169
10	13492900	17121700
11	148421911	188338711
12	1781062944	2260064544
13	23153818285	29380839085
14	324153456004	411331747204
15	4862301840075	6169976208075
16	77796829441216	98719619329216

```

function IOCTL()
fid = fopen('ABC.txt', 'r'); %打开当前路径文本文件
A = textscan(fid, '%d %d'); %读取多行文本
disp(A); %返回矩阵行列数
disp(A{1}); %取出第1列数据
disp(A{2}); %取出第2列数据
fclose(fid);
end

```

```
>> IOCTL  
[20×1 int32] [20×1 int32] 返回两列数据的矩阵
```

2	3
6	8
21	27
88	112
445	565
2676	3396
18739	23779
149920	190240
1349289	1712169
13492900	17121700
148421911	188338711
1781062944	2147483647
2147483647	第 2 列数据

这个值超出 int(32 位) 范围

第 1 列数据

CSV 文件创建与操作

`csvwrite`(文件名, 写入文件的变量)

文件名: 向指定 csv 文件写入数据, 如果指定文件没有, 就自动在当前路径创建文件。一般是 ‘xxx.csv’ 这种写法

写入文件的变量: 比如数组 `A = [10, 20, 30, 40, 50]`, 就将 A 变量填入, csv 文件就按照 A 数组格式排列

```
function IOCTL()
```

```
A = [10, 20, 30, 40, 50];  
csvwrite('myFile.csv', A);  
end
```



可以用 excel 打开,
也可以用 txt 打开

	A	B	C	D	E	F
1	10	20	30	40	50	
2						
3						

你看 excel 就能把数据读出来, 怎么数据是横向排列的呢?
因为我们建立的一维数组默认是横向的, 所以就只有横向排列,
如果要纵向, 需要数组转置



用 txt 文件打开, 我们发现 CSV 数据是用逗号分隔的。这就说明了 CSV 数据分隔使用逗号, 那么列的数据也是逗号分隔, excel 也是识别逗号分隔的。

这样看来 CSV 文件格式的数据是比较方便的, 兼容 txt 和 excel。

实时写数据到 csv 文件

```
|function IOCTL()
a = 1;
b = 2;
for x = 1:20
    a=(a+1)*x;
    b=(b+1)*x;
    csvwrite('myFile.csv', a, x, 0);
end
end
```

18	
19	
20	
21	90462000
22	00000000
23	000

发现 csvwrite 不能连续向 csv 文件写数据，只能一次性写入，如果是这样，就只有用缓存来做

```
|function IOCTL()
a = 1;
b = 2;
buff(20)=0; %定义个缓存数组
for x = 1:20
    buff(x)=(a+1)*x;%向缓存数组写数据
    b=(b+1)*x;

end
disp(buff)
csvwrite('myFile.csv', buff);%缓存数组一次性写入CSV
end
```

这种缓存写法就可以

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40						
2																										
3																										

```
|function IOCTL()
a = 1;
b = 2;
buff(20)=0; %定义个缓存数组
for x = 1:20
    buff(x)=(a+1)*x;%向缓存数组写数据
    b=(b+1)*x;

end
buff = buff';%将缓存行排列，用转置改成列排列
disp(buff)
csvwrite('myFile.csv', buff);%缓存数组一次性写入CSV
end
```

转置之后，数据就是列排列了

A
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

其实 csvwrite 并不怎么好用，我们下面还是用文件 IO 来实时写 csv 文件。

文件 IO 方式写 CSV 文件

```
|function IOCTL()
a = 1;
b = 2;
filename='myFile.csv';%.csv可以更改为.txt等
fid=fopen(filename, 'w');

for x = 1:20
    a=(a+1)*x;%向缓存数组写数据
    b=(b+1)*x;
    fprintf(fid, '%d,%d\r\n', a, b);
end
fclose(fid);
end
```

文件 IO 方式操作 CSV，数据写入又快又整齐

A	B	C
1	2	3
2	6	8
3	21	27
4	88	112
5	445	565
6	2676	3396
7	18739	23779
8	149920	190240
9	1349289	1712169
10	13492900	17121700
11	1.48E+08	188338711
12	1.78E+09	2260064544
13	2.32E+10	29380839085
14	3.24E+11	4.11332E+11
15	4.86E+12	6.16998E+12
16	7.78E+13	9.87196E+13
17	1.22E+15	1.67222E+15

时间戳写入 CSV 文件，需要字符转整数

返回浮点数 = str2num(字符串变量) %将字符串转换为浮点数

```
function IOCTL()
```

```
fid = fopen('mytimeFile.csv', 'w');
for x = 1:3
    datatime = datestr(now, 'yyyymmddTHHMMSS')
    disp(datatime);
    xstr = datatime(10:15);
    num = str2num(xstr);
    fprintf(fid, '%d, %d\r\n', x, num);
    pause(1);
end
fclose(fid);
```

```
end
```

```
>> IOCTL
```

```
20210506T212253
20210506T212254
20210506T212255
```

	A	B
1	1	212253
2	2	212254
3	3	212255

```
文件(F) 编辑(E)
1,212253
2,212254
3,212255
```

读取 CSV 文件中的数据进行解析

矩阵 = csvread(文件名/文件路径) %获取 CSV 文件内容，返回给矩阵。

	A	B
1	2	3
2	6	8
3	21	27
4	88	112
5	445	565
6	2676	3396
7	18739	23779
8	149920	190240

XZZMY.csv - 记事本

文件(F) 编辑(E) 格式(C)

2,3

6,8

21,27

88,112

445,565

2676,3396

18739,23779

149920,190240

必须确保 CSV 文件是逗号分隔，而不是 excel 创建的 CSV 文件，用空白分隔。空白分隔的 CSV 文件，csvread 函数读取出来是乱的。

```
M = csvread('XZZMY.csv');
disp(M);
```

```
>> test
      2      3
      6      8
     21     27
     88     112
    445     565
   2676    3396
  18739   23779
149920  190240
```

数据读出来是矩阵。

如何获取 CSV 矩阵的每行每列数据呢?

```
M = csvread('XZZMY.csv');  
disp(M);
```

2	3
6	8
21	27
88	112
445	565
2676	3396
18739	23779
149920	190240

我们知道 M 变量是一个矩阵

M(5,1) %就是获取 M 矩阵中第 5 行第 1 列的数据

```
disp(M(5,1));  
a=M(5,1); %还可以将第5行第1列的数据复制给变量
```

```
disp(M(1,2)); 第 1 行第 2 列就是 3
```

计算 M 矩阵的大小，自动获取 M 的数据

返回值 = size(变量) %返回变量的行列数

变量: 传入矩阵

返回值: 一般返回值都是行和列两个变量

```
[i, k]=size(M);%返回M矩阵行列数  
disp(i);  
disp(k);  
i= 8 行, k=2 列
```

M(:,1) %这种冒号是取出第 1 列所有数据，返回给 x

```
x=M(:, 1);  
disp(x);
```

2
6
21
88
445
2676
18739
149920

我们可以用这种方式取出某列数据做时间轴。

```

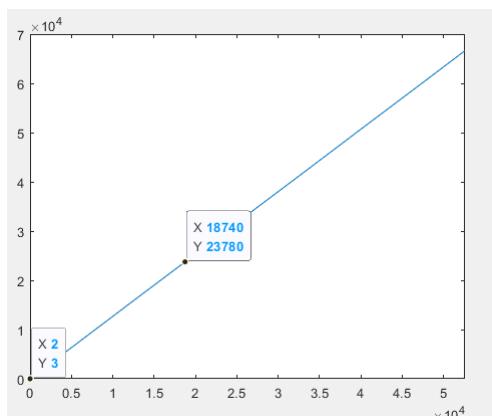
function test()

M = csvread('XZZMY.csv');
disp(M);
disp(M(1, 2));
a=M(5, 1); %还可以将第5行第1列的数据复制给变量
disp(a);
[i, k]=size(M);%返回M矩阵行列数
disp(i); %得到数据行数
disp(k); %得到数据列数

x=M(:, 1); %取出第1列所有数据给x,
disp(x); %显示第1列数据
datax = []; %定义数组
for count = 1:i %数据行数循环
    datax(count) = M(count, 2); %取出每一行的第2列数据
end
disp(datax); %得到每1行第2列数据
plot(x, datax);%将第1列数据放入x做时间, 第2列数据放入Y做大小

end

```



2	3
6	8
21	27
88	112
445	565
2676	3396
18739	23779
149920	190240

因为鼠标精度的原因，我选取的这两个数据和曲线点基本对应上。

字符串拼接

```

>> a = '12345'
>> b = '56789'
>> sc = [a, b]

```

[字符串变量 1,字符串变量 2]拼接之后返回给新变量

sc =

```

'1234556789'

```

MATLAB 在 GUI 模式下设置 Plot 曲线显示卡死, 在 2019b 版本中, GUI 模式下绘图是用 Plot, 但是曲线放大缩小必须用 axtoolbar 函数实现(重点)

返回值 = axtoolbar(GUI 显示得句柄,{参数设置}) %给 plot 显示得曲线增加放大缩小, 数据显示等功能

GUI 显示得句柄: 创建  坐标系, 在 Tar 设置的名称

参数设置: 'zoomin' 放大

'zoomout' 缩小

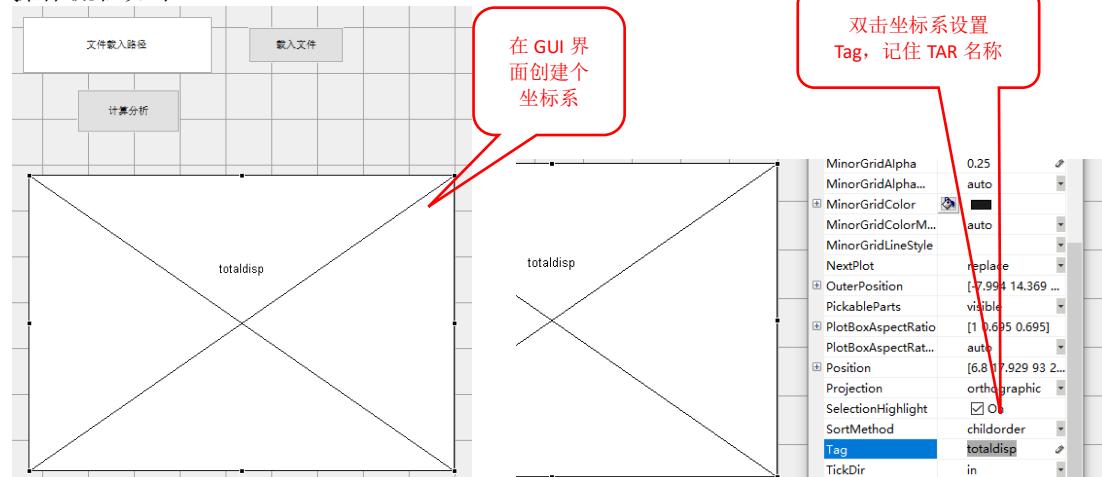
'restoreview' 还原坐标, 比如你放大到不知道什么情况了, 又不想滚动鼠标还原, 就用这个参数

'pan' 移动坐标轴

'datacursor' 显示曲线中某个点数据

```
plot(handles 要显示得界面句柄,x,y)
[tb,btms]=axtoolbar(handles.totaldisp,['zoomin','zoomout','restor
eview','pan','datacursor']);%显示放大缩小, 显示坐标数据
```

操作流程如下



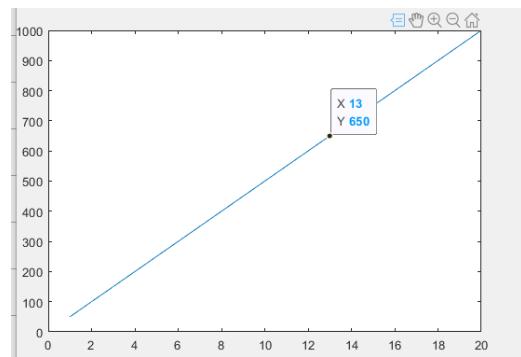
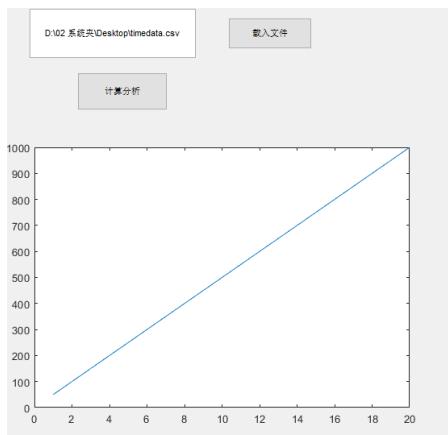
创建回调函数, 一般都用不到

```
% --- Executes during object creation, after setting all properties.
function totaldisp_CreateFcn(hObject, eventdata, handles)
```

在显示曲线的回调函数中, 先用 plot 指定曲线显示在哪个坐标系变量上, 这里是 handles.totaldisp, 这是前面 Tag 设置的

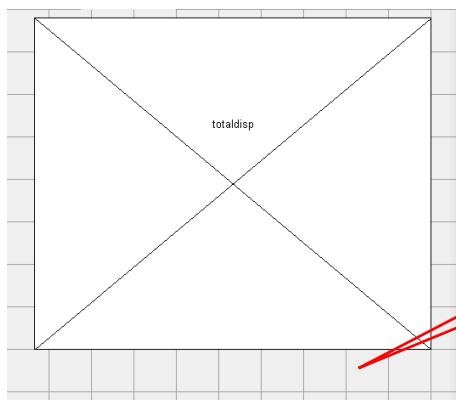
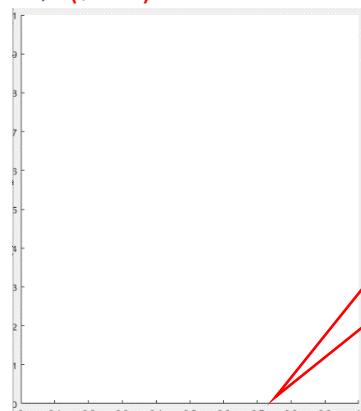
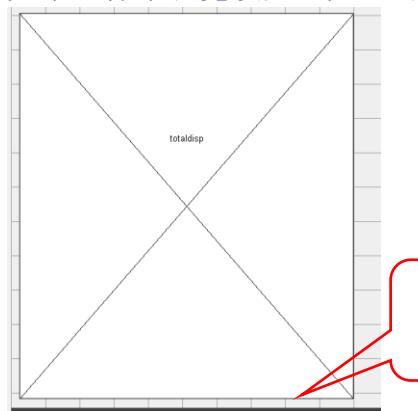
```
plot(handles.totaldisp,x,total);
[tb, btms] = axtoolbar(handles.totaldisp,['zoomin','zoomout','restor
eview','pan','datacursor']);
```

然后设置 axtoolbar 坐标工具也显示在 handles.totaldisp 坐标系上



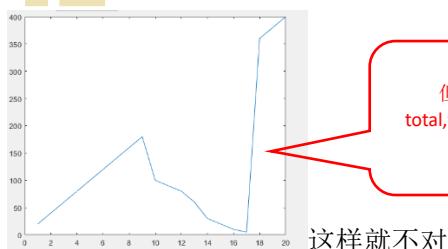
运行成功

如果坐标系长宽设置过大，可能显示会超出边框(注意)



axtoolbar 多条曲线显示

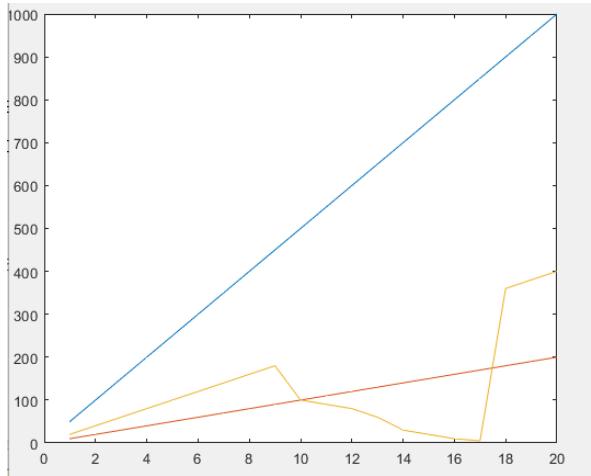
```
plot(handles.totaldisp, x, total);
plot(handles.totaldisp, x, weight1);
plot(handles.totaldisp, x, weight2);
[tb, btns] = axtoolbar(handles.totaldisp, {'zoomin', 'zoomout', 'restoreview', 'pan', 'datacursor'});
```



这样就不对

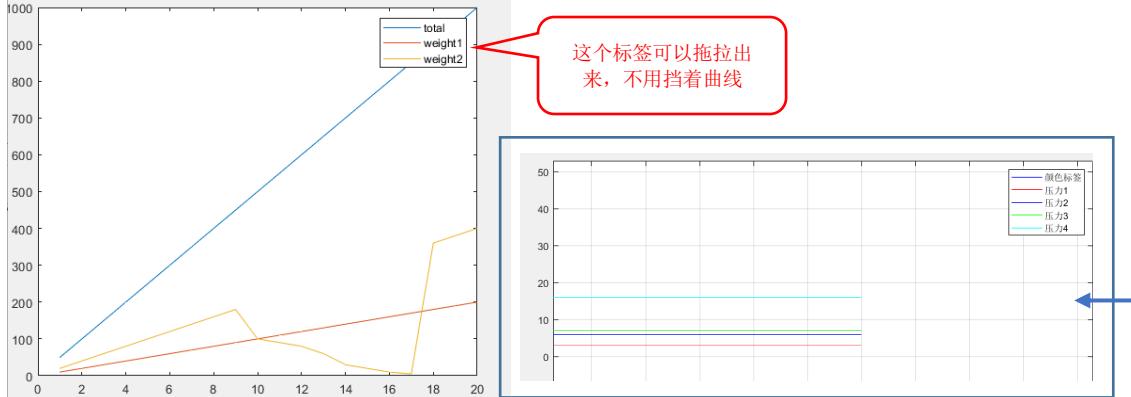
其实就是 plot 填入三组坐标变量即可

```
plot(x, total, x, weight1, x, weight2); %3条曲线用同一个x时间轴  
[tb, btns] = axtoolbar(handles.totaldisp, {'zoomin', 'zoomout', 'restorereview', 'pan', 'datacursor'});
```



这就是三条曲线，还有个问题，这三条颜色的曲线分别代表什么呢？

```
plot(x, total, x, weight1, x, weight2); %3条曲线用同一个x时间轴  
legend('total', 'weight1', 'weight2') %给曲线添加名字  
[tb, btns] = axtoolbar(handles.totaldisp, {'zoomin', 'zoomout', 'restorereview', 'pan', 'datacursor'});
```



如果有四条曲线，给每条曲线加颜色怎么做？

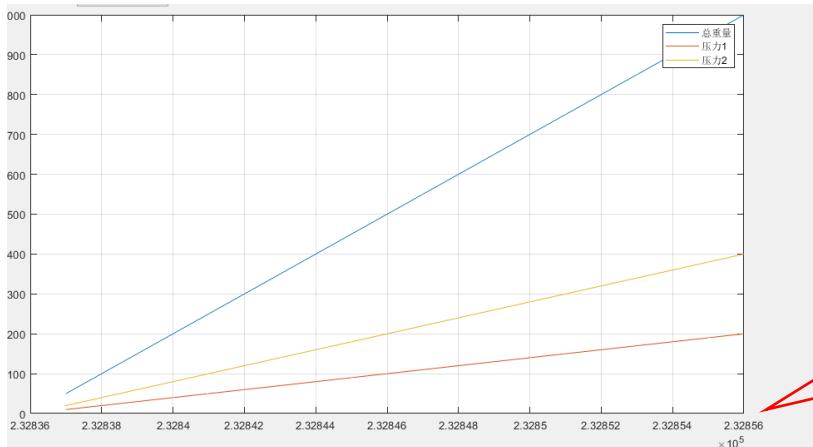
```
plot(timecount, simpleData1, 'r', timecount, simpleData2, 'b', timecount, simpleData3, 'g', timecount,  
simpleData4, 'c', 'MarkerSize', 5); %这里面第1条曲线是红色，因为后面跟的'r'（也就是red），后面3条以此类推  
legend('颜色标签', '压力1', '压力2', '压力3', '压力4'); 给4条曲线加上标签，为什么不把压力1写在第一个  
'颜色标签位置'? 这是因为压力1的颜色提示是从第2个标签开始算得，我也不知道为什么照做就是了  
数值过大，坐标轴显示问题
```

```
x = M(:, 7); %如果直接获取时间  
for count = 1:row  
    weight1(count) = M(count, 2); %压力数据1 第2列  
    weight2(count) = M(count, 3); %压力数据2 第3列  
    weight3(count) = M(count, 4); %压力数据3 第4列  
    weight4(count) = M(count, 5); %压力数据4  
    total(count) = M(count, 6); %总压力数据4  
end
```

	A	B	C	D	E	F	G
1	1	10	20	30	40	50	232837
2	2	20	40	60	80	100	232838
3	3	30	60	90	120	150	232839
4	4	40	80	120	160	200	232840
5	5	50	100	150	200	250	232841
6	6	60	120	180	240	300	232842

```
plot(x, total, x, weight1, x, weight2); %3条曲线用同一个x时间轴  
legend('总重量', '压力1', '压力2') %给曲线添加名字  
[tb, btns] = axtoolbar(handles.totaldisp, {'zoomin', 'zoomout',  
grid on});
```

如果直接把时间赋值给 plot 的 x 坐标显示，那么数值就很大，还有指数显示。



`set(gca,'XTickLabel',X 坐标开始值 : X 坐标刻度步进: X 坐标最大值) %设置 x 轴刻度线`

X 坐标开始值: `x=0` 的位置显示为多少，可以是自定义的值

X 坐标刻度步进: 每个刻度线增加多少值

X 坐标最大值: X 坐标最多显示多少

```

x = M(:, 1); %如果直接获取时间
for count = 1:row
    weight1(count) = M(count, 2);%压力数据1 第2列
    weight2(count) = M(count, 3);%压力数据2 第3列
    weight3(count) = M(count, 4);%压力数据3 第4列
    weight4(count) = M(count, 5);%压力数据4
    total(count) = M(count, 6);%总压力数据4
end
plot(x, total, x, weight1, x, weight2);%3条曲线用同一个x时间轴
set(gca, 'XTickLabel', 232837:1:300000);
legend('总重量', '压力1', '压力2') %给曲线添加名字
[tb, btns] = axtoolbar(handles.totaldisp, {'zoomin', 'zoomout', 'restoreview', 'pan', 'datacursor'});
grid on

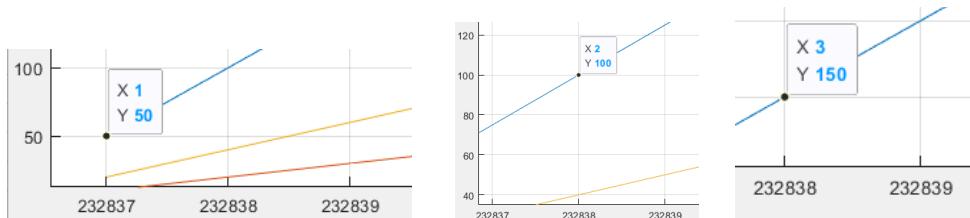
```

X 还是用第 1 列来画图

	A	B
1	1	10
2	2	20
3	3	30
4	4	40
5	5	50
6	6	60

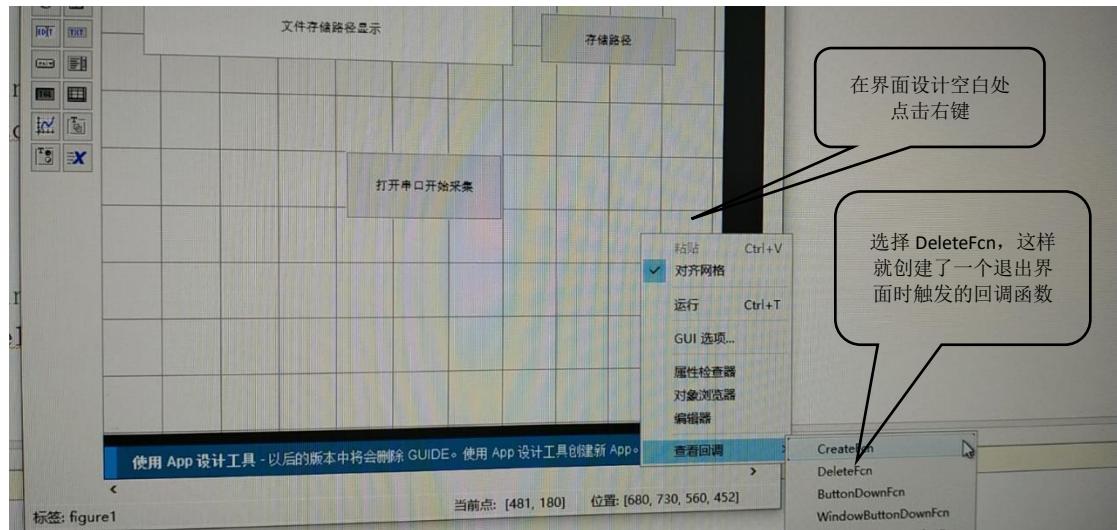
第 1 列数字比较小适合做 x 轴的时间

但是我们眼睛看得 x 坐标轴改成
232837 的真实时间。
这表示 232837 开始，刻度步进
为 1，最大刻度 300000



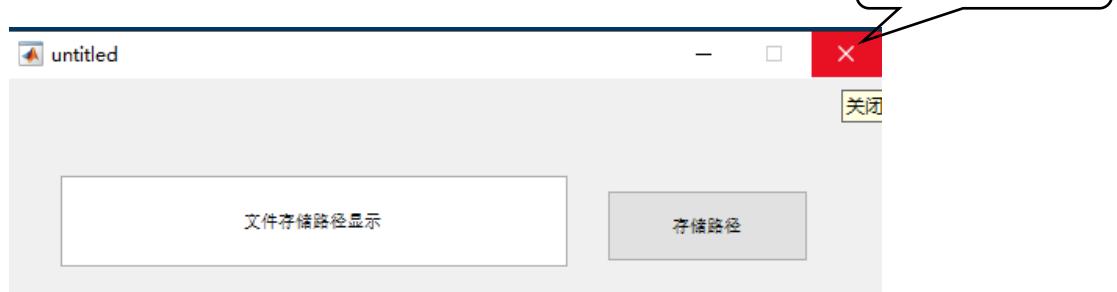
我发现 `x=2` 是 232838，但是 `x=3` 也是 232838，有点视觉误差，只有将就用了。

MATLAB 在关闭 GUI 界面的时候需要增加一些回调函数，让正在死循环的主程序也能关闭，如果 GUI 界面关闭了，不清除串口，主程序死循环，就会导致 MATLAB 多次启动后很卡，操作方式如下：



```
% --- Executes during object deletion, before destroying properties.  
function figure1_DeleteFcn(hObject, eventdata, handles)  
    fprintf("close\n");
```

我现在打开界面，再关闭界面试试



```
>> untitled  
close  
>> DeleteFcn 回调函数中的 close 打印被执行。  
我们就是利用 DeleteFcn 回调函数，向主程序的死循环发命令。也可以在 DeleteFcn 回调函数中做些清理数据的事情，比如关闭串口，或者关闭文件什么的操作。
```

MATLAB 图表使用

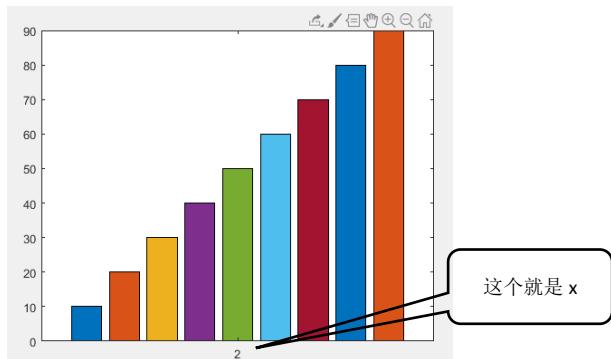
直方图绘制

`bar(x,y)`

x: 必须为单调递增或者递减,x 就是底部显示得数值, x 等于多少, x 位置就显示多少

`y = [10 20 30 40 50 60 70 80 90];`

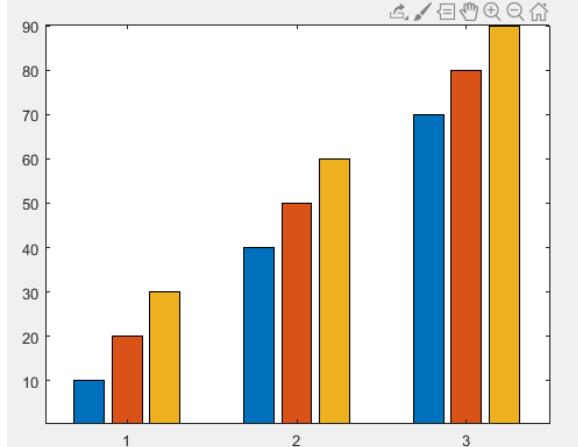
`bar(2,y);`



x 就是用来分组的, y 数组里面每 1 个元素就是一个柱状图

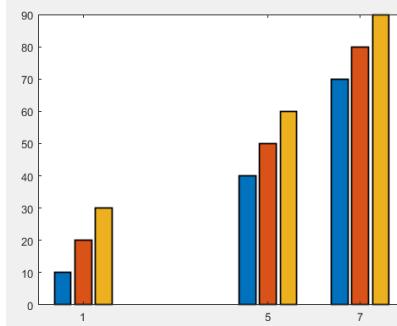
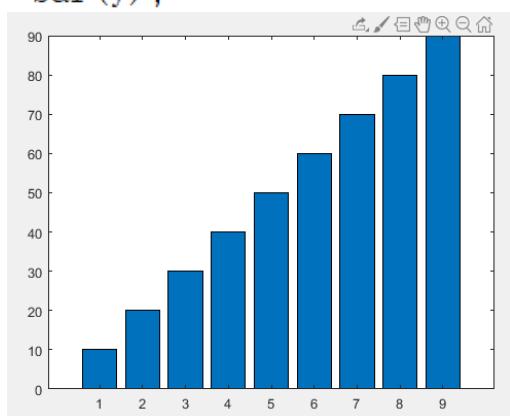
`y = [10 20 30; 40 50 60; 70 80 90];`

`bar([1 2 3],y);`



`y = [10 20 30 40 50 60 70 80 90];`

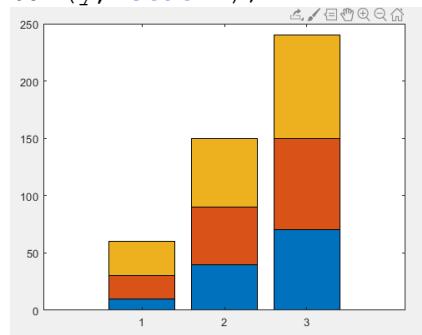
`bar(y);`



`y = [10 20 30; 40 50 60; 70 80 90];`

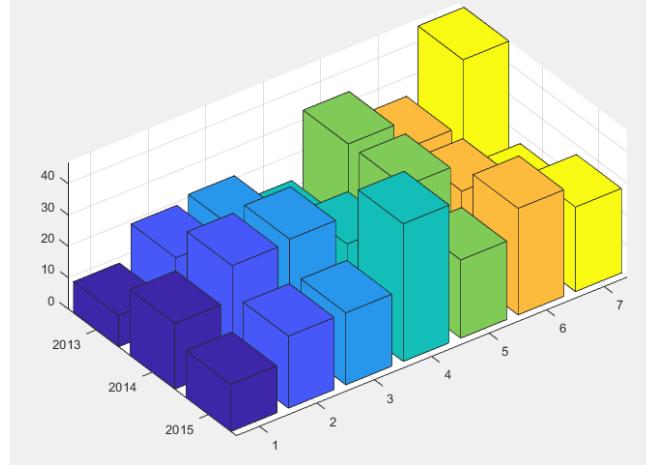
`bar([1 5 7],y,'LineWidth',1.4); %条形图边框线宽度`

```
y = [10 20 30; 40 50 60; 70 80 90]; %分号隔离的数据组成为一个柱状上
bar(y, 'stack');
```



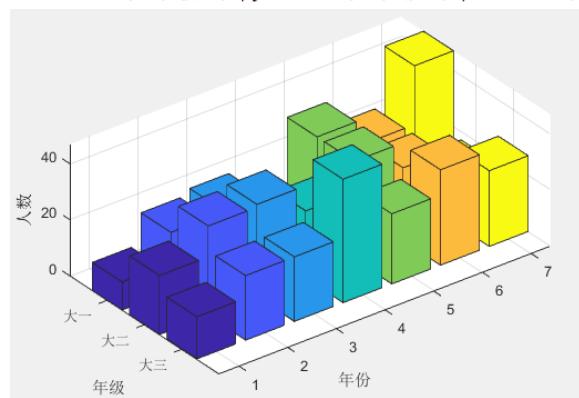
bar3(多少列纵轴数据, 每个数据高度) %三维直方图

```
x=[2013 2014 2015];
y=[10 21 23 14 35 26 47;21 32 33 24 35 26 17;15 23 23 44 25 34
27];
bar3(x,y);
```



用 set 和 xlabel 给数字增加有意义的文字表示

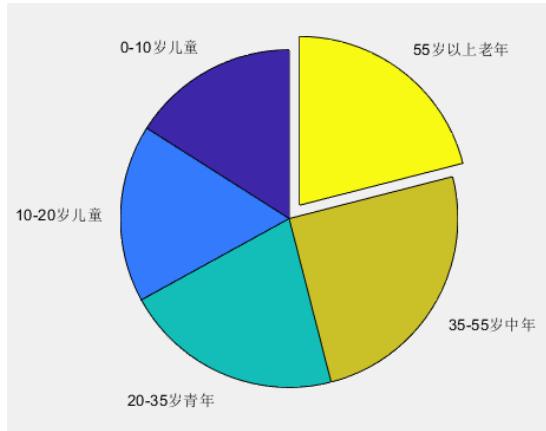
```
x=[2013 2014 2015];
y=[10 21 23 14 35 26 47;21 32 33 24 35 26 17;15 23 23 44 25 34 27];
bar3(x,y);
set(gca,'YTickLabel', {'大一', '大二', '大三', '大四', '硕一', '硕二', '硕三'});
xlabel('年份'); ylabel('年级'); zlabel('人数');
```



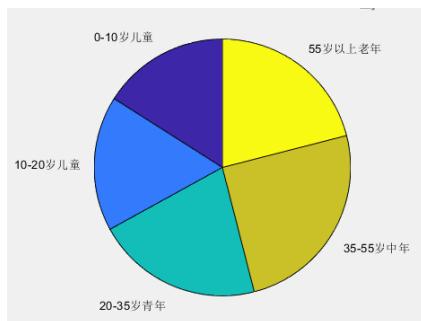
你看有中文文字表示了

饼图绘制

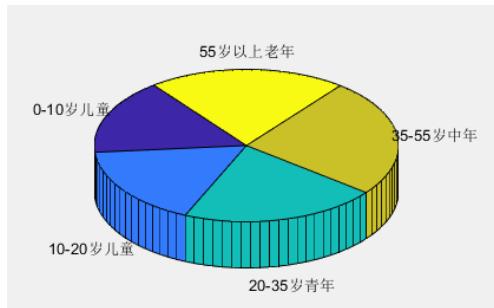
```
x=[16 17 21 25 21];  
pie(x, [0 0 0 0 1], {'0-10岁儿童', '10-20岁儿童', '20-35岁青年', '35-55岁中年', '55岁以上老年'});
```



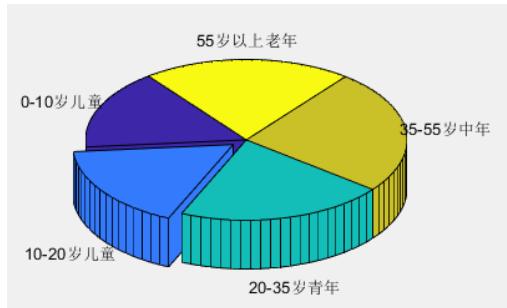
```
x=[16 17 21 25 21];  
pie(x, [0 0 0 0 0], {'0-10岁儿童', '10-20岁儿童', '20-35岁青年', '35-55岁中年', '55岁以上老年'});
```



```
x=[16 17 21 25 21];  
pie3(x, [0 0 0 0 0], {'0-10岁儿童', '10-20岁儿童', '20-35岁青年', '35-55岁中年', '55岁以上老年'});
```



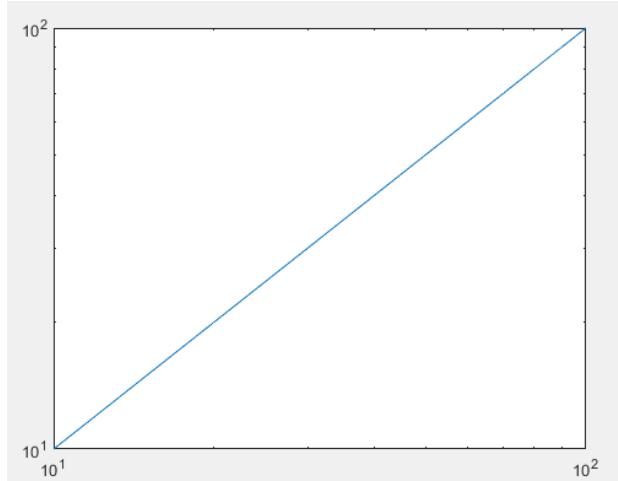
```
x=[16 17 21 25 21];  
pie3(x, [0 2 0 0 0], {'0-10岁儿童', '10-20岁儿童', '20-35岁青年', '35-55岁中年', '55岁以上老年'});
```



log 表示曲线(数据量大适合使用)

loglog(x 轴数组,y 轴数组)

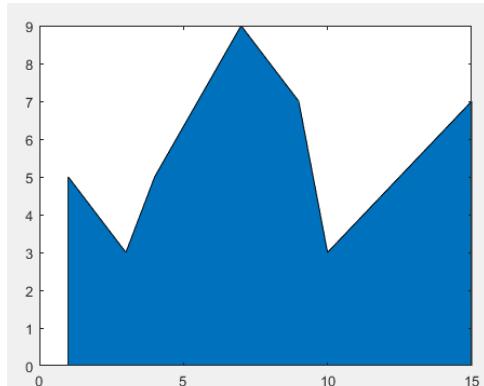
```
x=[10 20 30 40 50 60 70 80 90 100];  
y=[10 20 30 40 50 60 70 80 90 100];  
loglog(x, y);
```



面积图

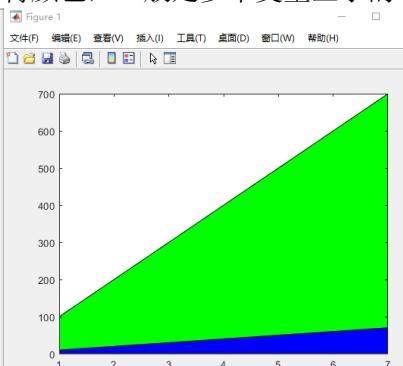
area(x 轴数组, y 轴数组)

```
x=[1 3 4 7 9 10 15]; %注意这里有转置  
y=[5 3 5 9 7 3 7];  
area(x, y);
```



facecolor 选择区域图显示得颜色，一般是多个变量显示的时候用与区分区域

```
x=[10 20 30 40 50 60 70];  
y=[100 200 300 400 500 600 700];  
area(y, 'facecolor', 'g');  
hold on  
area(x, 'facecolor', 'b');
```

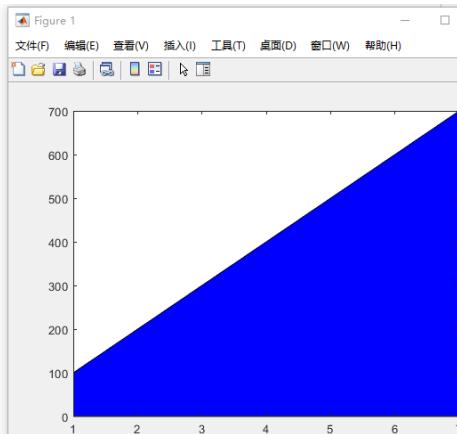


一般数值小的先显示，数值大的后显示

```

x=[10 20 30 40 50 60 70];
y=[100 200 300 400 500 600 700];
area(x,'facecolor','g');
hold on
area(y,'facecolor','b');

```

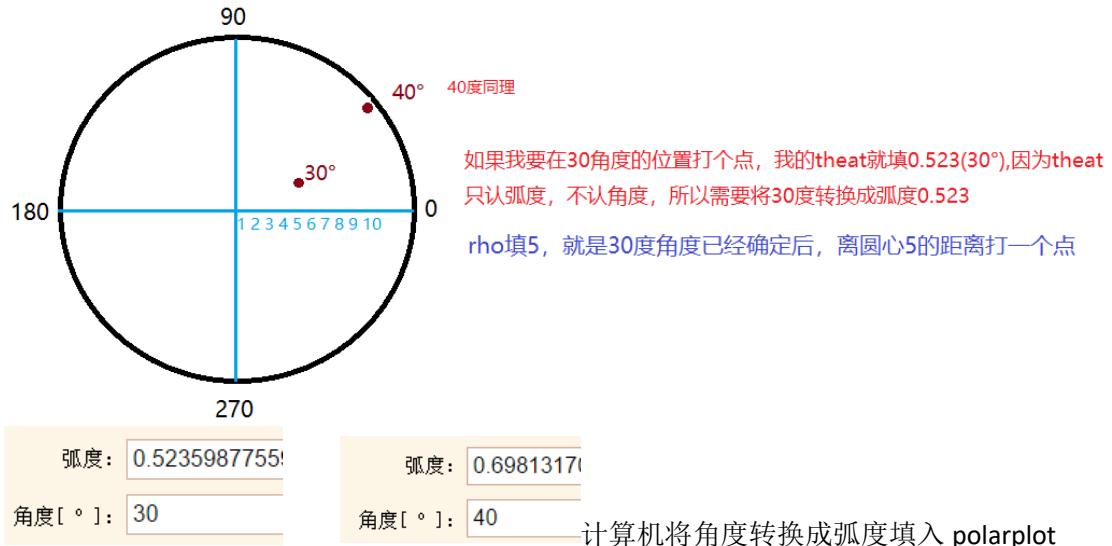


如果数值大的 y 会覆盖数值小的 x

极坐标使用

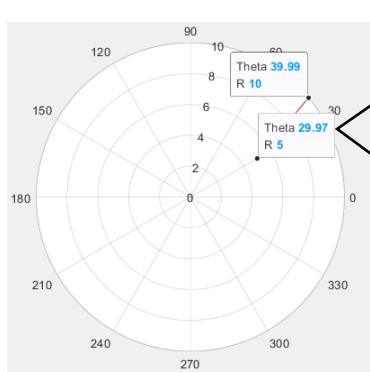
`polarplot(theta, rho)`

`theta`: 弧度设定, 每个数据点在图中的角度位置, 一般这个参数个数要和 `rho` 数组个数一致
`rho`: 设置每个数据点离圆心距离为多少



`a=[0.523 0.698]; %theta是填入弧度, 所以要把sin(角度)转成弧度
b=[5 10]; %设定对应角度的半径`

`polarplot(a, b, 'r')` %绘图

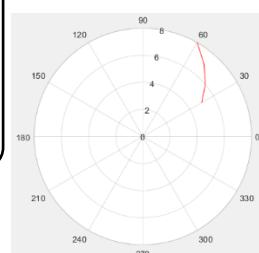


设置的时候是弧度, 但是显示还是用角度显示

`a=[0.523 0.698 0.8726 1.0471]; %theta是填入弧度, 所以要把sin(角度)转成弧度
b=[5 6 7 8]; %设定对应角度的半径`

`polarplot(a, b, 'r')` %绘图

显示多个数据点, 形成一条曲线

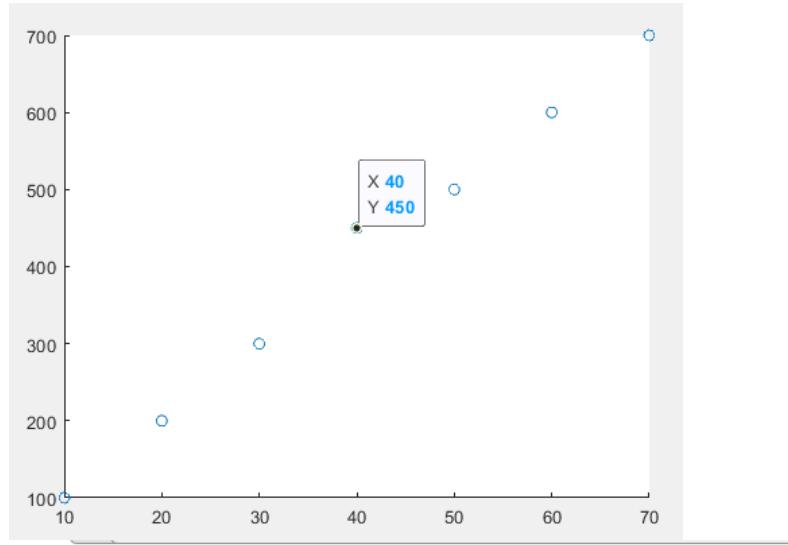


这就是极坐标使用

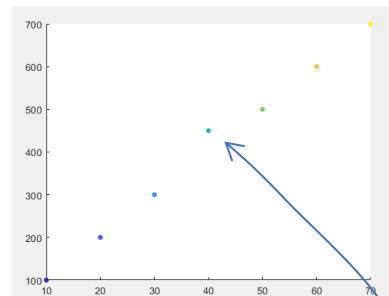
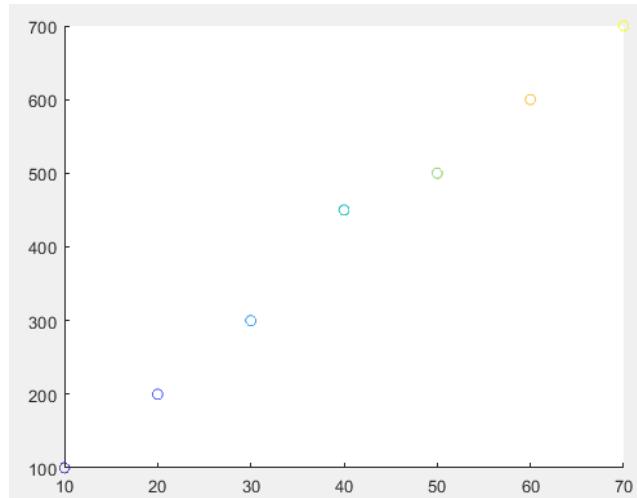
散点图绘制

scatter(x 轴数组, y 轴数组)

```
t = [10 20 30 40 50 60 70];  
y = [100 200 300 450 500 600 700];  
scatter(x, y)
```

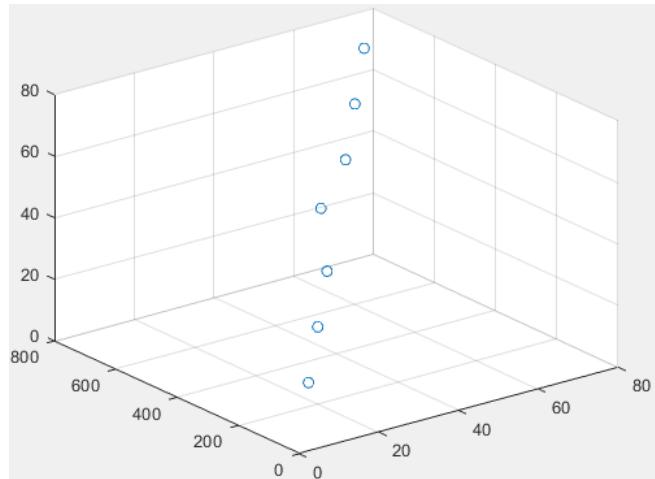


```
x = [10 20 30 40 50 60 70];  
y = [100 200 300 450 500 600 700];  
c = linspace(1, 10, length(x)); % 绘制每个点的颜色  
scatter(x, y, [], c);
```

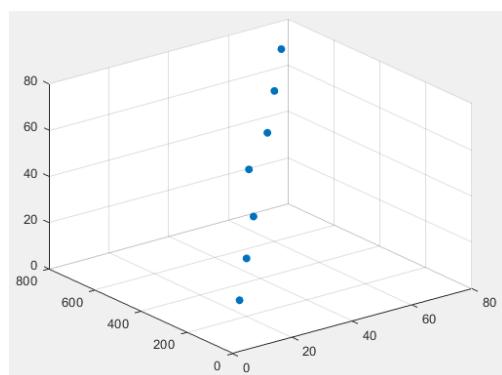


```
x = [10 20 30 40 50 60 70];  
y = [100 200 300 450 500 600 700];  
c = linspace(1, 10, length(x)); % 绘制每个点的颜色  
scatter(x, y, 25, c, 'filled'); % 将[]填入颜色深度值, 末端加入'filled'  
数据点颜色变深了
```

```
scatter3(x 轴数组,y 轴数组,z 轴数组) %三维散点图  
x = [10 20 30 40 50 60 70];  
y = [100 200 300 450 500 600 700];  
z = [15 25 35 45 55 65 75];  
scatter3(x, y, z);
```

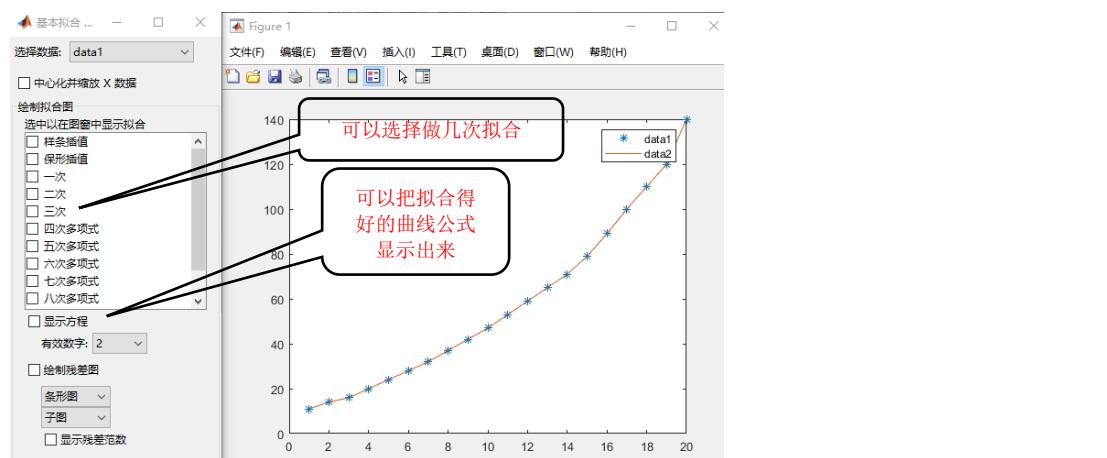
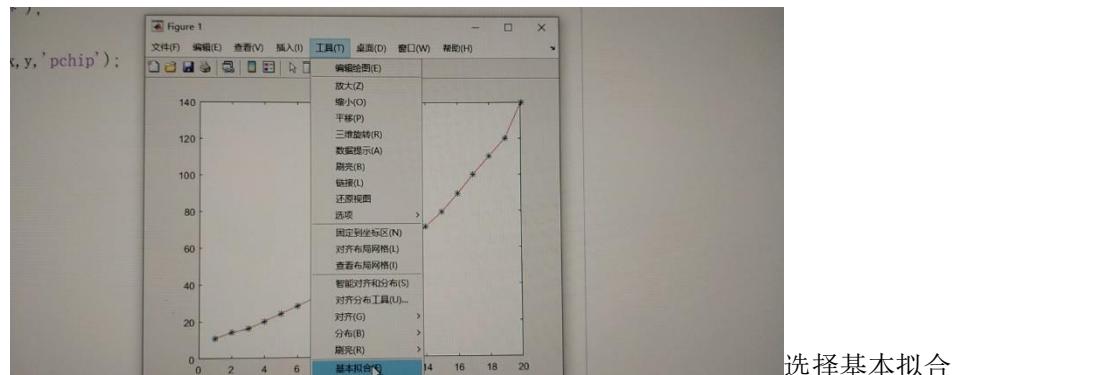
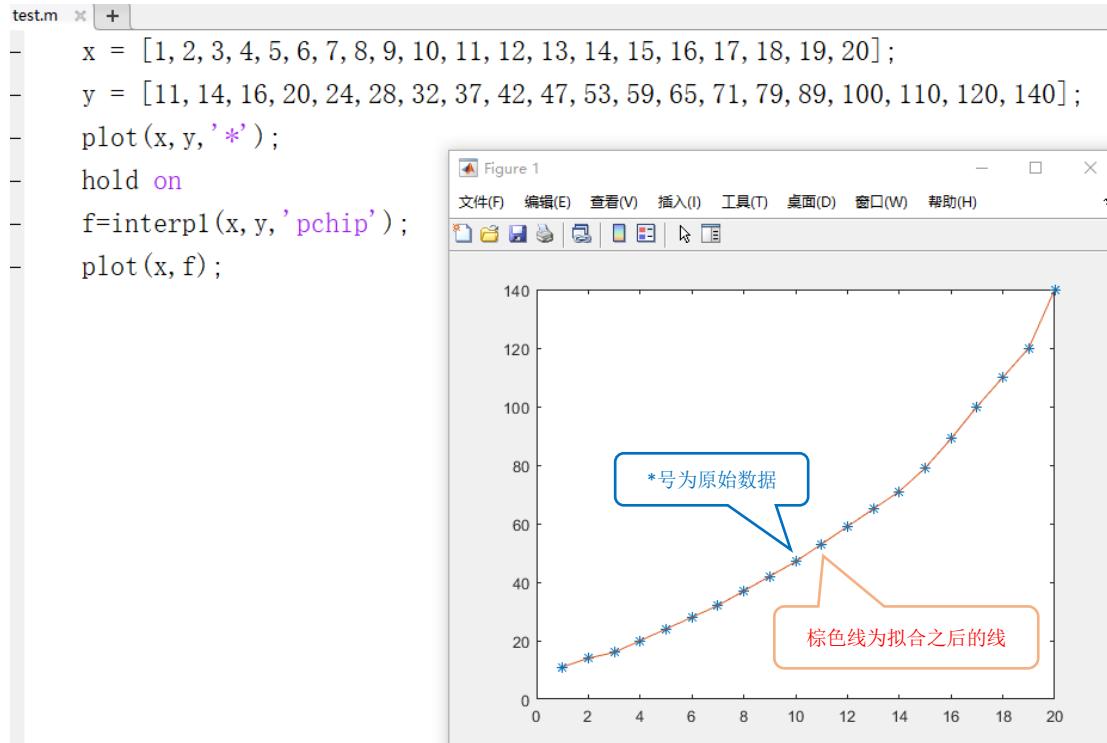


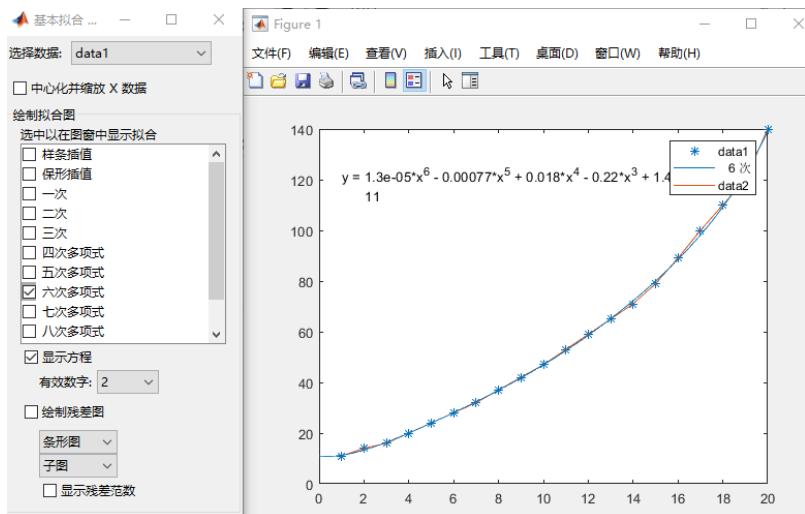
```
x = [10 20 30 40 50 60 70];  
y = [100 200 300 450 500 600 700];  
z = [15 25 35 45 55 65 75];  
scatter3(x, y, z, 'filled');%加入'filled' 让数据点颜色更明显
```



MATLAB 生成 C 代码

拟合后的值 = `interp1(x横轴标值, y纵轴值, 拟合方式)` %多次项曲线拟合函数





我选择了 6 次拟合。

`f=interp1(x,y,'pchip')` 现在我想将 `interp1` 函数生成 C 代码，便于单片机使用。用 MATLAB 生成 C 代码的前提条件，就是你的 MATLAB 代码必须有输入输出。那么能实现输入输出要求的就是定义函数。

```

test.m x myfit.m x +
function [f] = myfit(x, y)
    f=interp1(x, y, 'pchip');
end

```



```

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
y = [11, 14, 16, 20, 24, 28, 32, 37, 42, 47, 53, 59, 65, 71, 79, 89, 100, 110, 120, 140];
plot(x, y, '*');
hold on

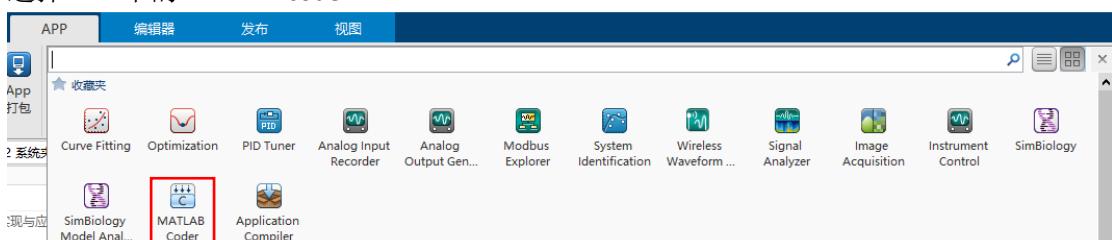
f = myfit(x, y);
plot(x, f);

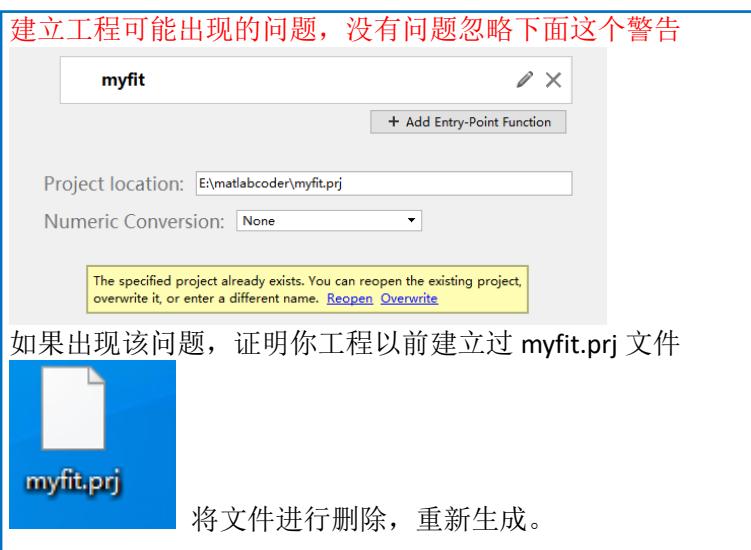
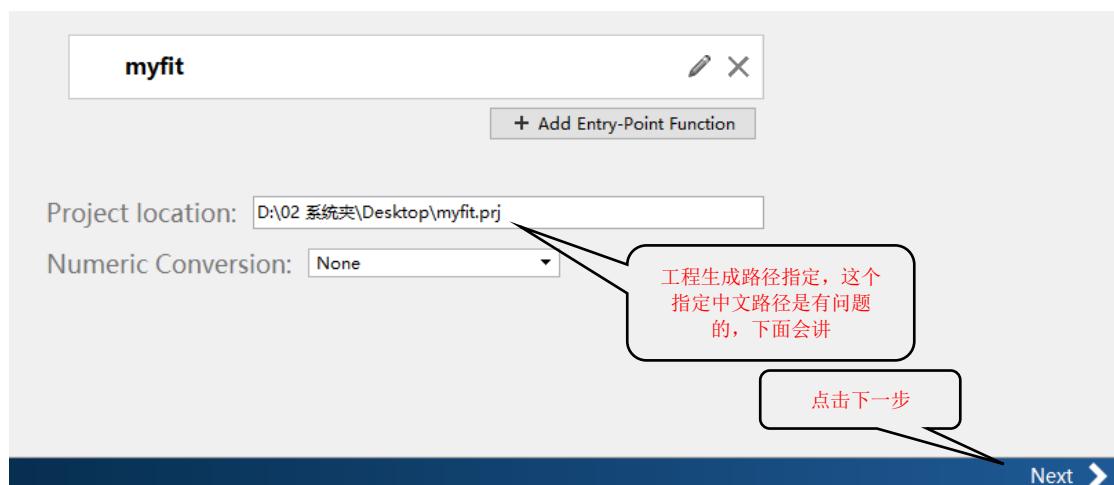
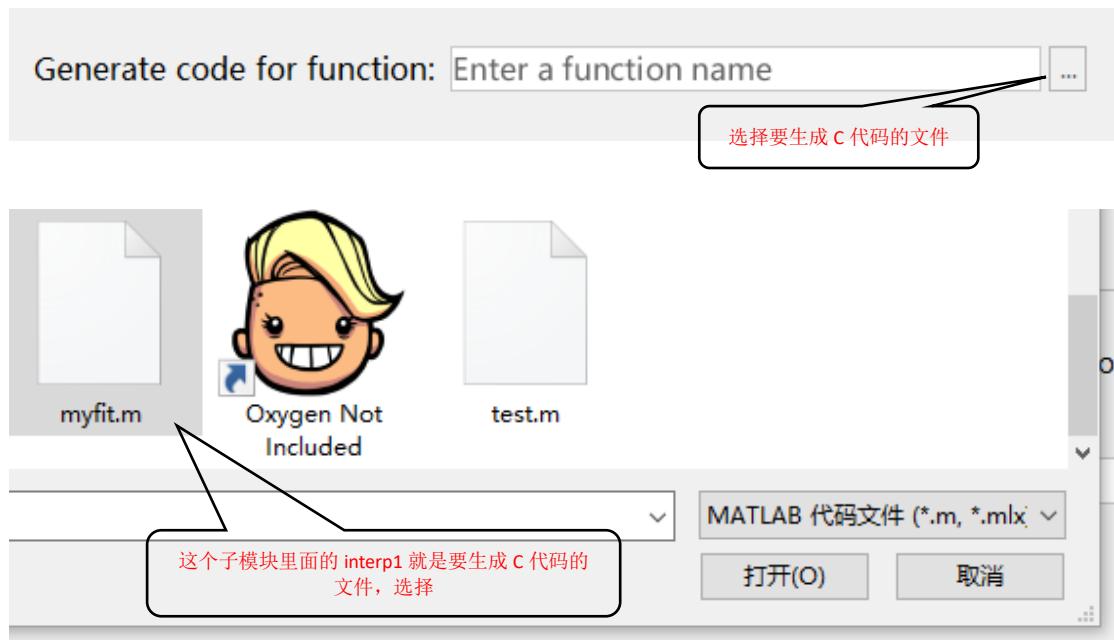
```

运行成功。

下面来生成 C 代码

选择 APP 下的 MATLAB coder

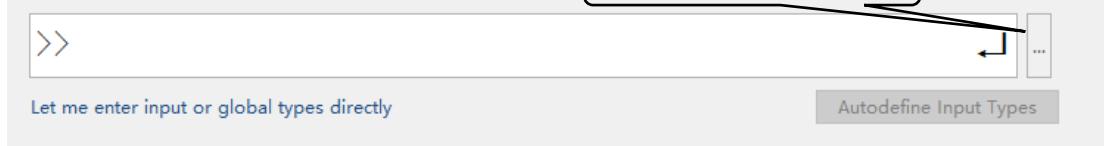




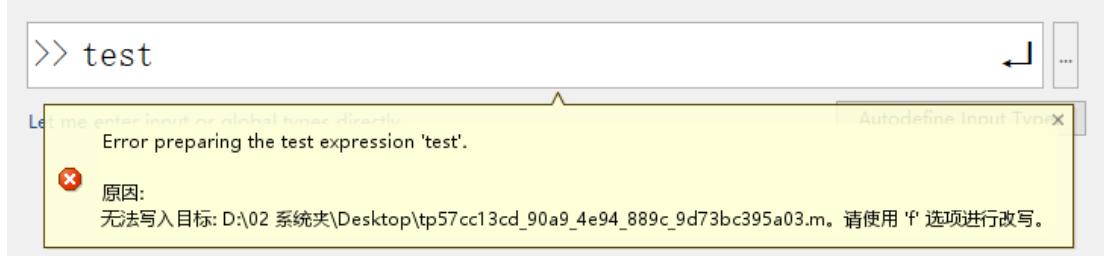
To convert MATLAB to C, you must define the type of each input for every entry point function. [Learn more](#)

To automatically define input types, call myfit or enter a script that calls myfit in the MATLAB prompt below:

选择调用 myfit 函数的主文件



如果出现如下加载错误



这是因为你m文件及你生成的工程是在桌面中文路径下，把工程拷贝到英文路径就能解决。

如果没有出现错误，点击 Autodefine Input Types

To automatically define input types, call myfit or enter a script that calls myfit in the MATLAB prompt below:



还是出现了无法识别问题，你自己写的子模块，返回值或者形参有问题

我下面改成简单的求和运算试试

```

test.m x myfit.m x +
1 - x = 10;
2 - y = 20;
3 -
4 - f = myfit(x, y);
5 - plot(x, f);

```

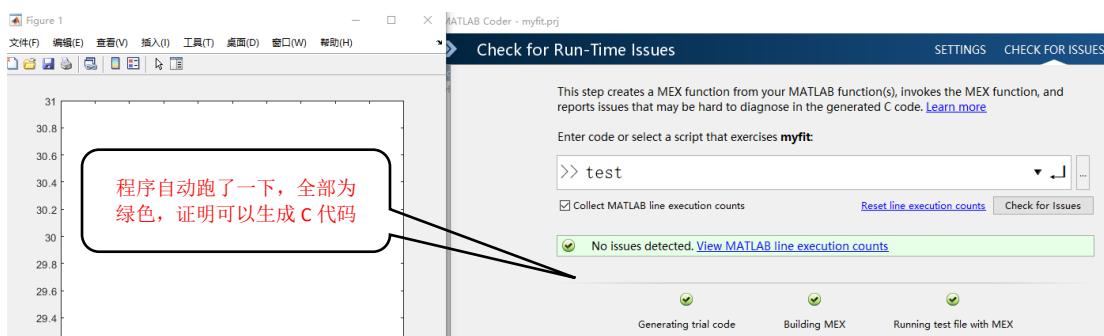
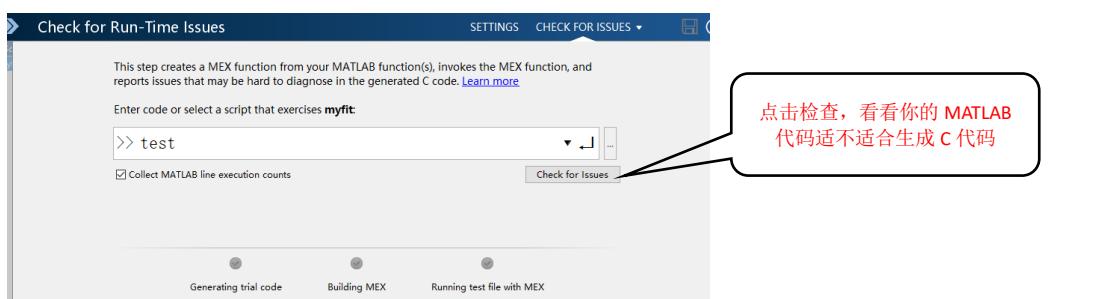
子模块改成求和运算

主程序也改简单了

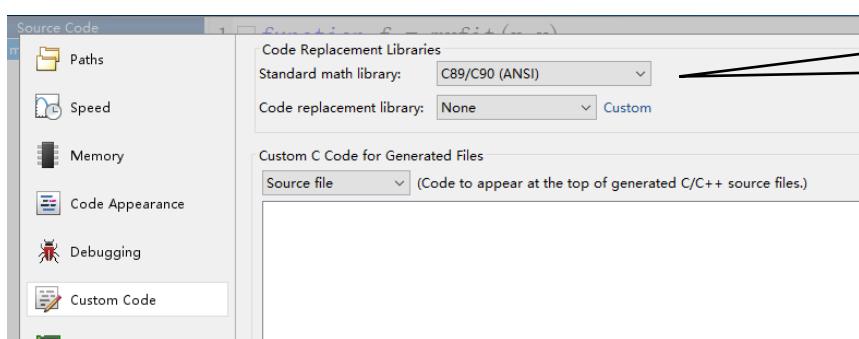
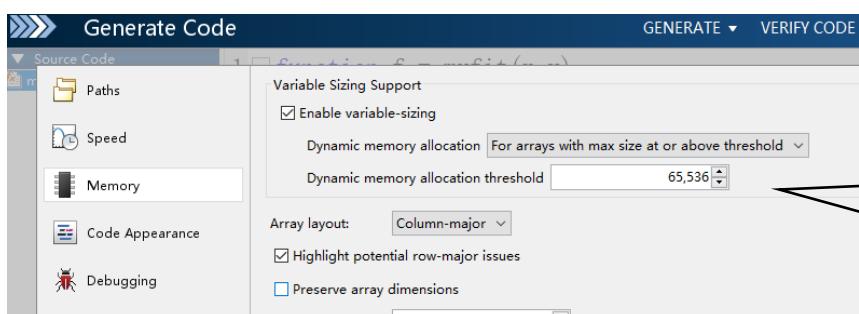
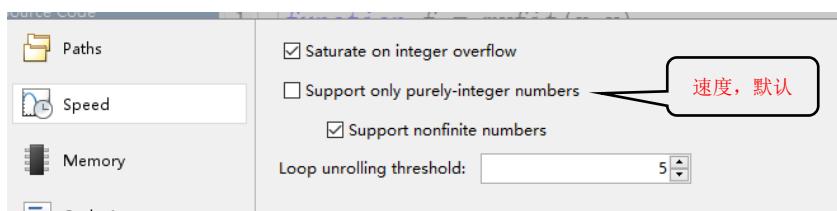
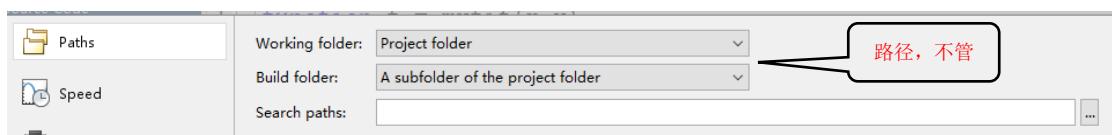
再次删除 myfit.prj 工程，重新点击 MATLAB coder 生成代码流程。



继续下一步



继续点击下一步



The screenshot shows the MATLAB Coder interface. On the left, a tree view displays 'Output Files' containing several C and H files: main.c, myfit_initialize.c, myfit_terminate.c, myfit.c, main.h, myfit_initialize.h, myfit_terminate.h, myfit_types.h, myfit.h, and rtwtypes.h. A red box highlights 'myfit.c'. A callout bubble points to it with the text '这就是生成的代码，使用方式' (This is the generated code, usage method). Another callout bubble points to the tree view with the text '一个加法代码生成了这么多少 c 文件' (A simple addition code generates so many C files). The main pane shows the generated C code:

```

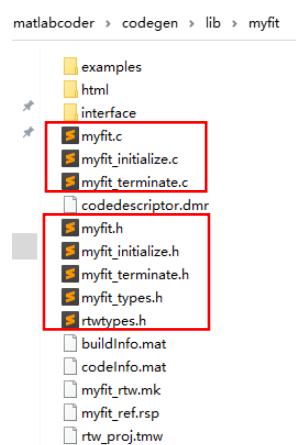
18 double myfit(double x, double y)
19 {
20     double f;
21     f = x + y;
22
23     /* f=interp1(x,y,'pchip'); */
24     return f;
25 }
26
27 /*

```

At the bottom, there are tabs for 'Target Build Log', 'Variables', and 'Size'. The 'Variables' tab is selected.

这些 C 文件和 H 文件都要拷贝进 STM32 工程进行编译。所以 MATLAB 算法复杂点的话，代码量相当大，吃内存。

进入 `codegen\lib\myfit` 生成的工程目录下

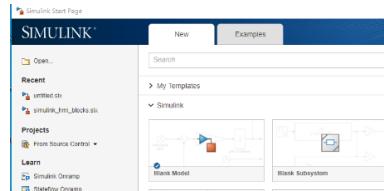


将里面的 C 文件和 H 文件拷贝出来编译。

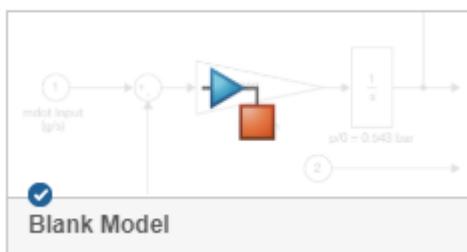
Simulink 基本使用

启动 simulink

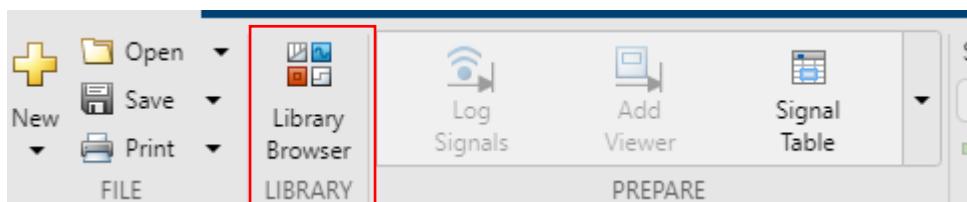
```
>> simulink  
>>
```



▼ Simulink

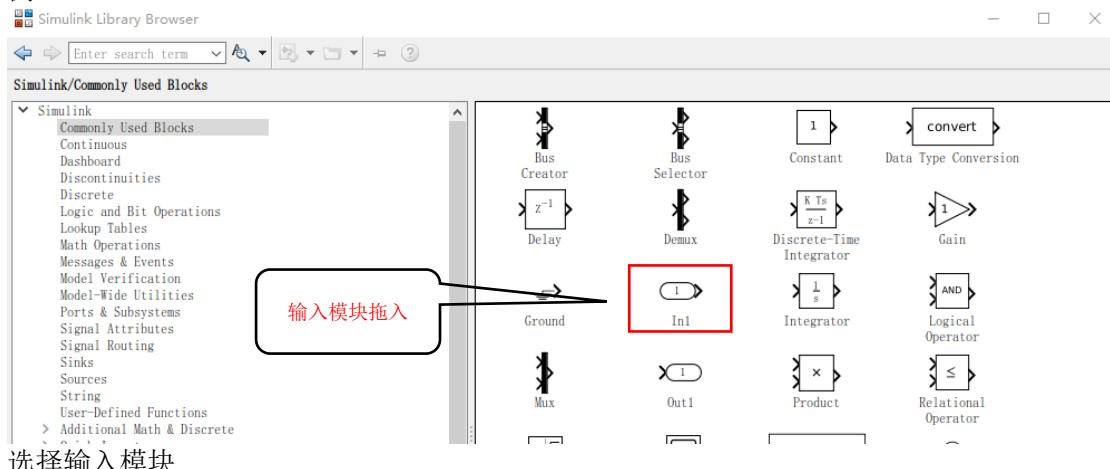


创建空模型

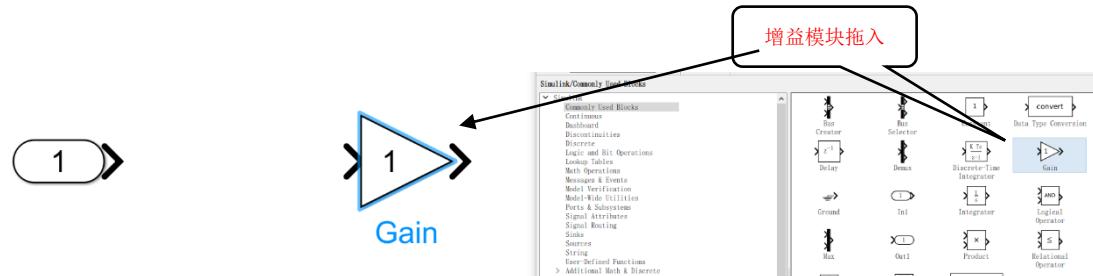


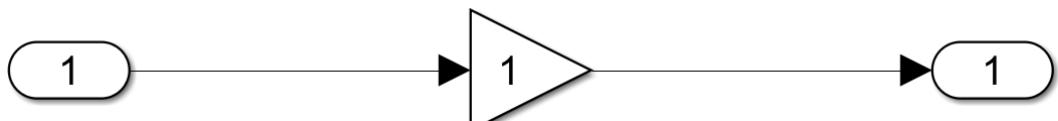
在 libraryBrowser 里面寻找需要的功能模块

例 1:



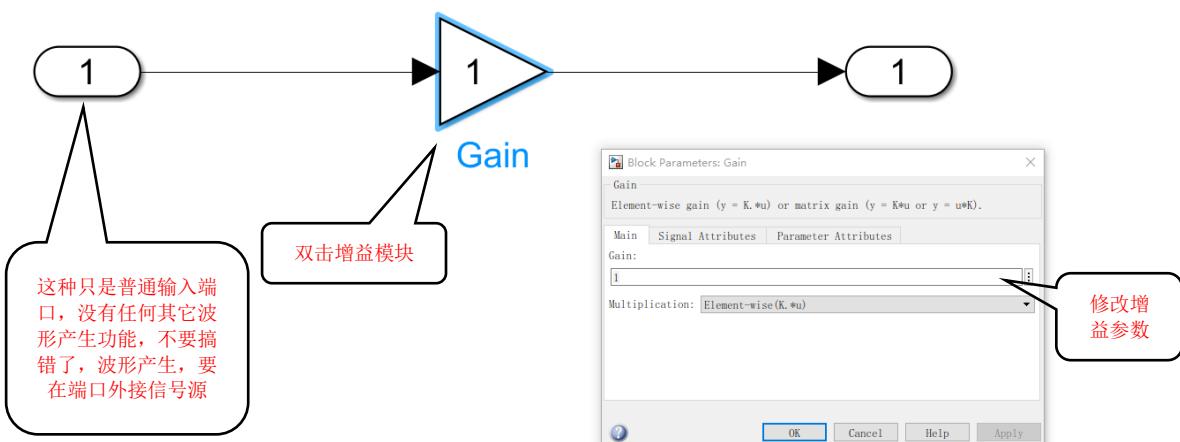
选择输入模块



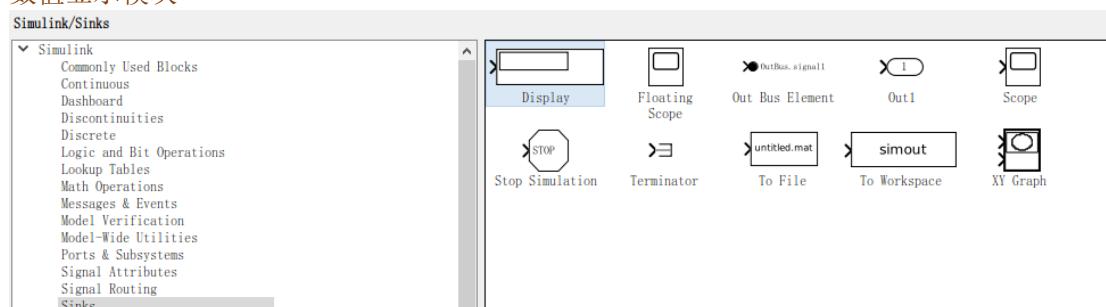


将模块连接起来

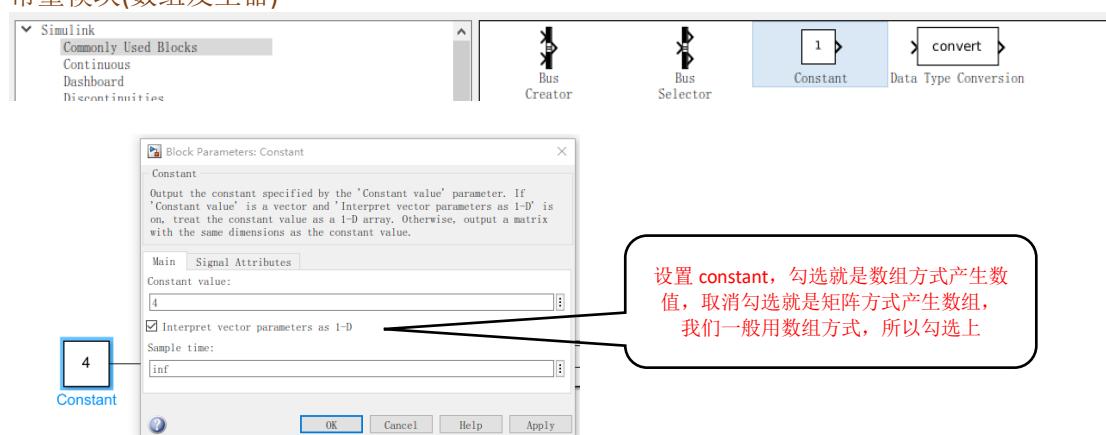
增益模块使用



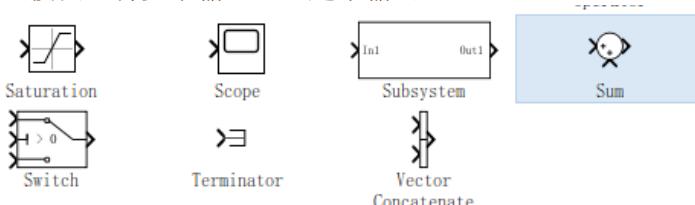
数值显示模块



常量模块(数组发生器)

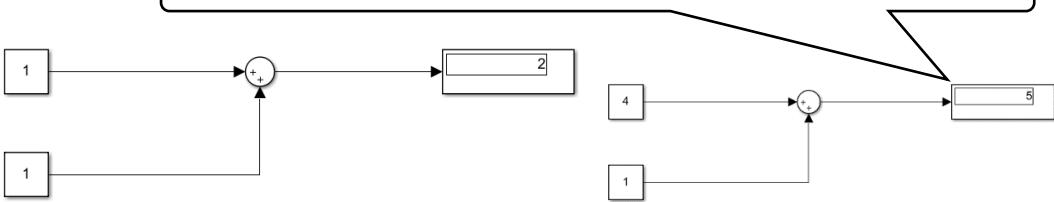


加法模块, 将多个输入叠加起来输出



例 2:

注意 disp 模块默认是 short 显示最大(16位), 如果超过 65535, 必须修改 disp 模块的属性



将两个数组发生模块接入求和模块。
就成了加法输出。输出结果为 2.

修改数组发生模块, 输出 5

常亮模块和加法模块的矩阵运算方式

Block Parameters: Sum

Sum

Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, ≥ 1 , specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Main Signal Attributes

Icon shape: rectangular

List of signs: |++

Constant

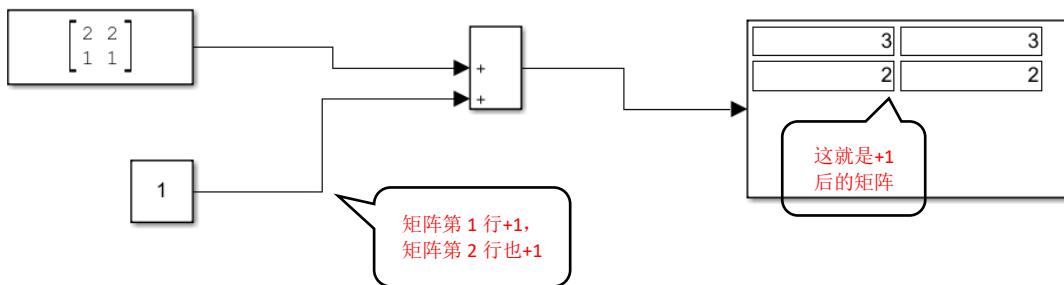
with the same dimensions as the constant value

Main Signal Attributes

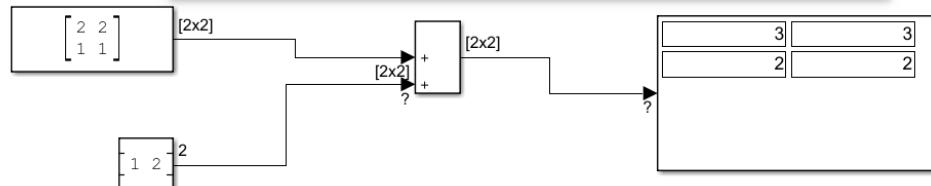
Constant value: [2 2; 1 1]

Interpret vector parameters as 1-D

这是个 2×2 的矩阵

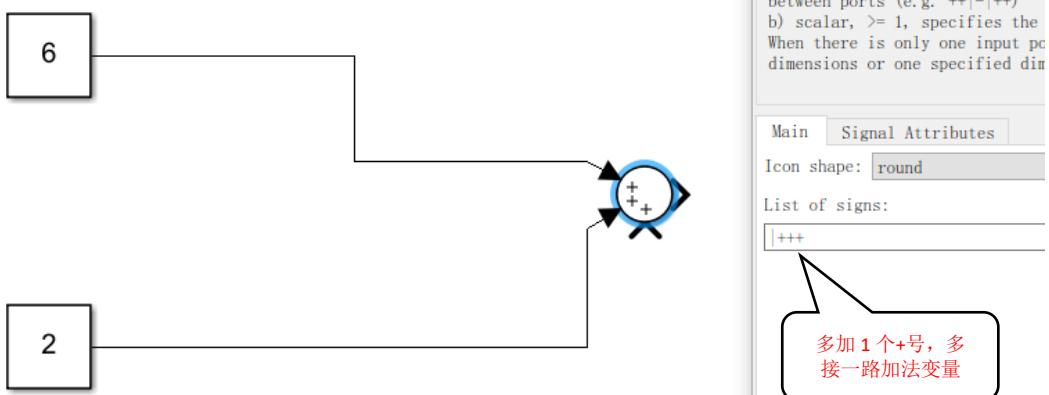


Error in port widths or dimensions. Invalid dimension has been specified for input port 2 of 'untitled/Sum'.
Component: Simulink | Category: Model error
Error in port widths or dimensions. Output port 1 of 'untitled/Constant1' is a one dimensional vector with 2 elements.
Component: Simulink | Category: Model error

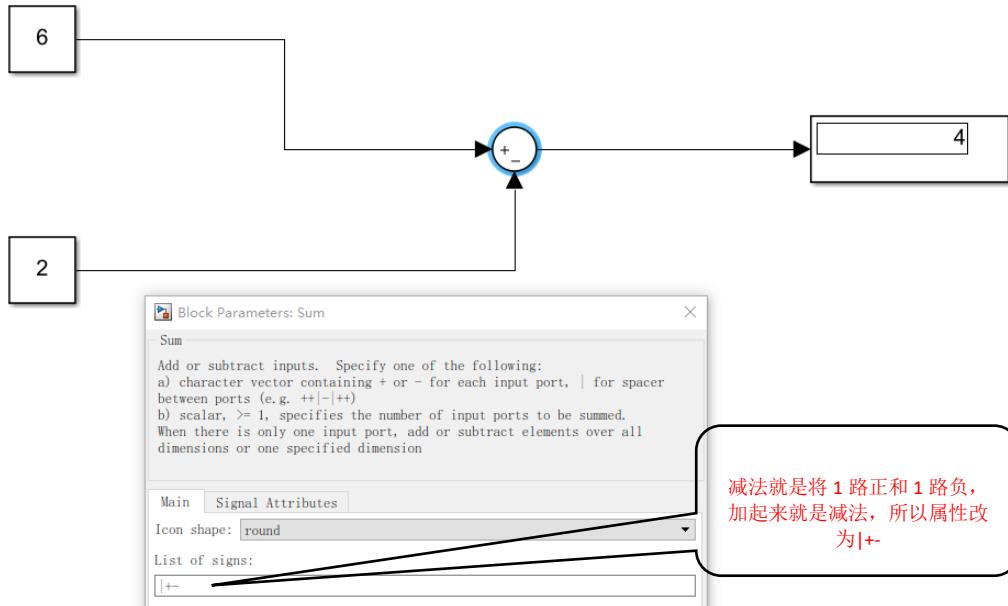
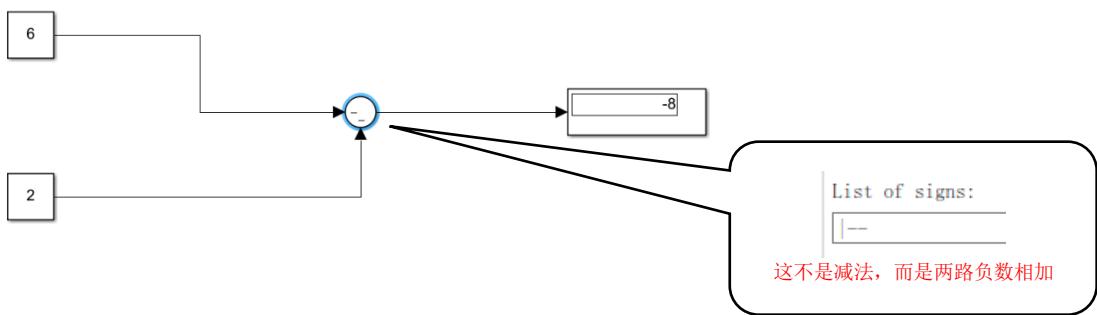


如果两个矩阵相加, 矩阵之间维度不一样, 2×2 矩阵去加 1×1 矩阵。就会报错 Error in port widths or dimensions

如果加法模块要加 3 个变量，修改方法

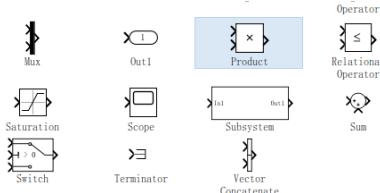


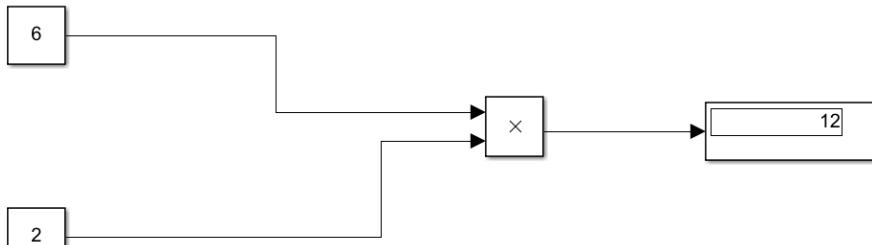
减法模块使用



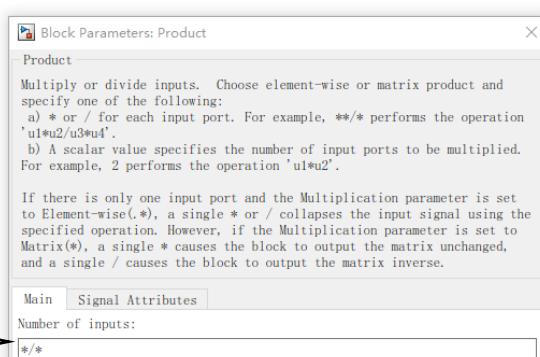
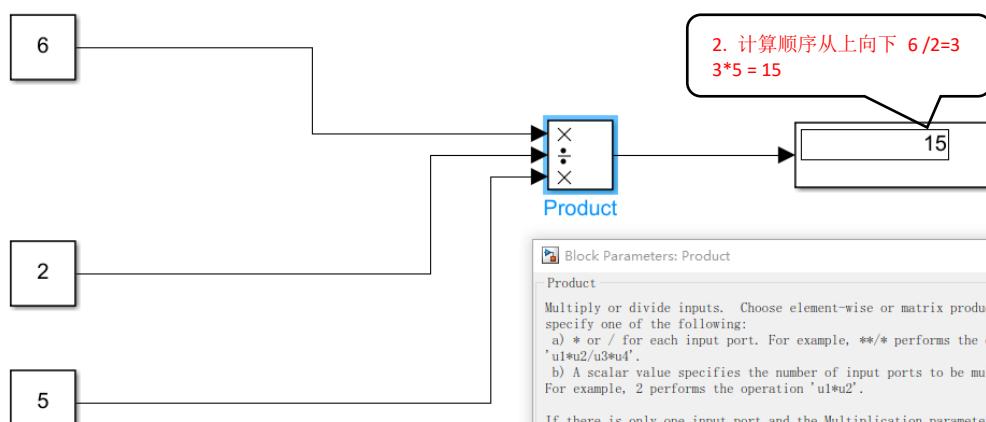
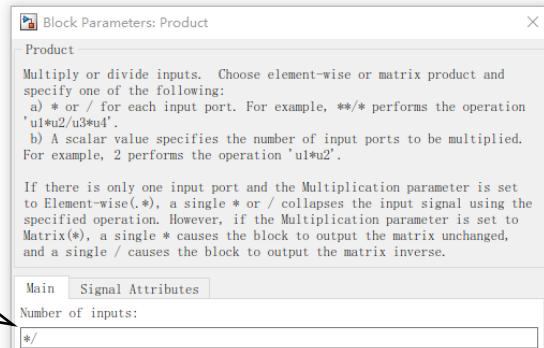
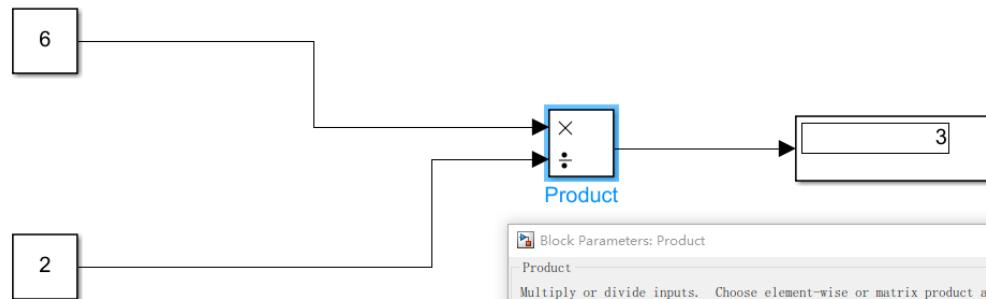
乘法模块使用

在 simulink->Commonly Used Block 中选择乘法模块

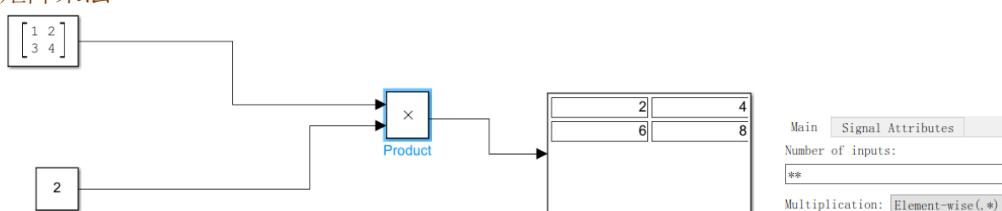




$$6 \times 2 = 12$$



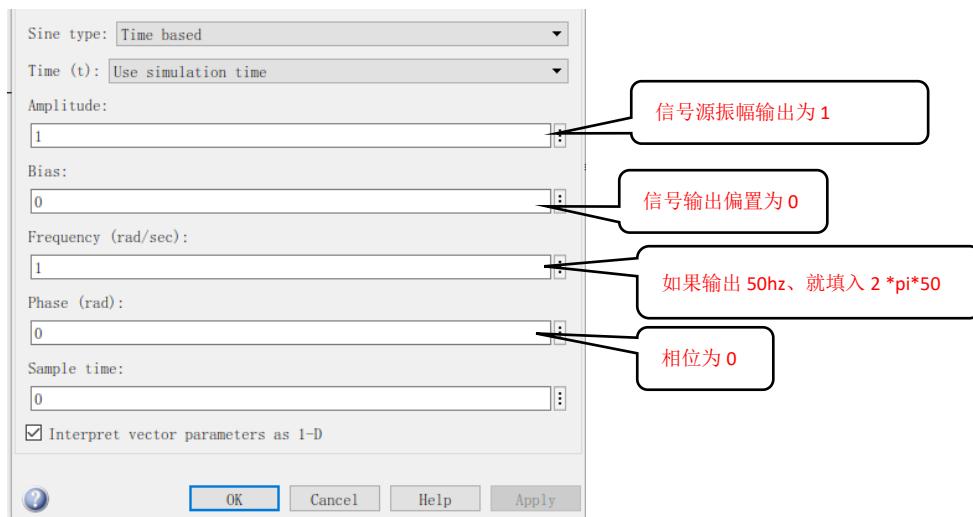
矩阵乘法



矩阵所有元素 乘 一个常量，结果正确。

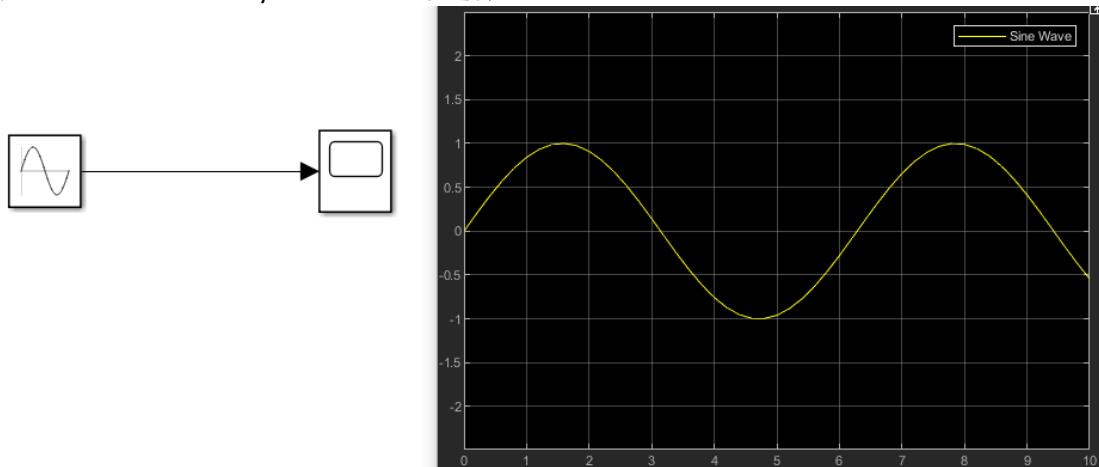
正弦波信号源模块

在 simulink->Sources 中找到信号源



示波器模块

在 simulink->Commonly Used Block 中选择



这是示波器采集信号源振幅 1 的信号

积分模块使用

在 simulink->Commonly Used Block 中选择



微分模块使用

在 simulink->Continuous 中选择

