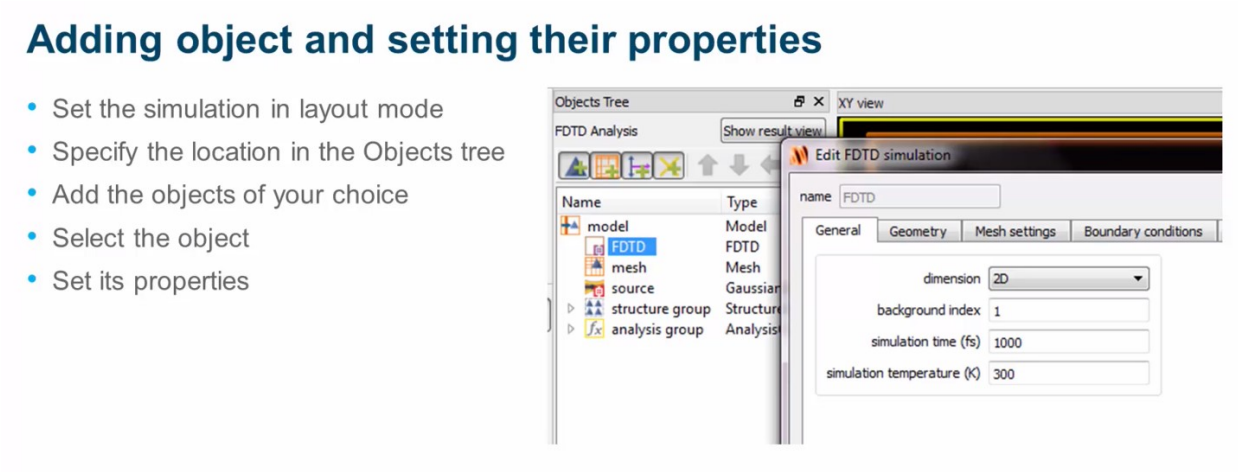
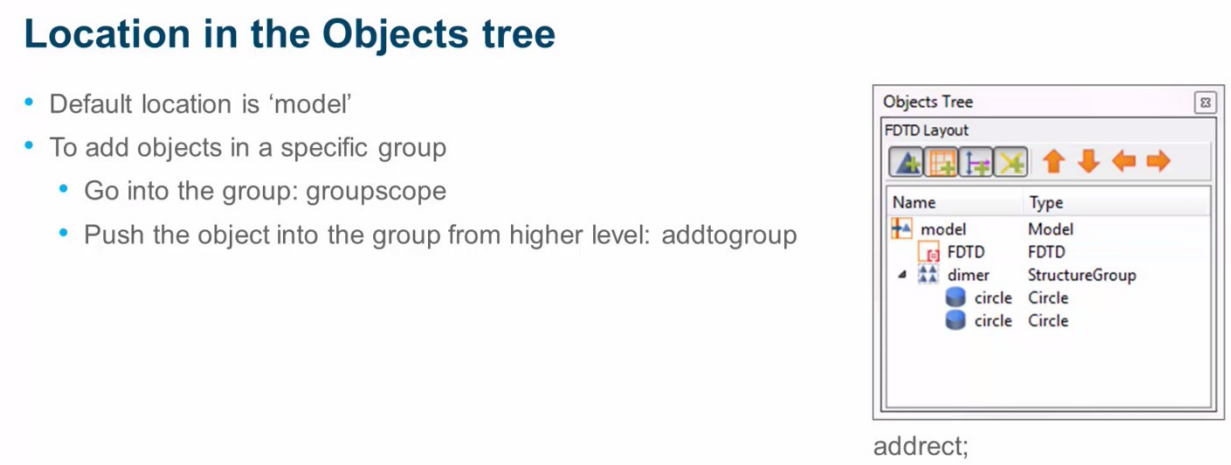
1. 使用模拟类

1.1 添加并设置类属性的流程



主要涉及：指定类的所属关系，选定类并设置其属性

1.2 定位 objects tree



addrect;

想要对某一个类属性进行更改，或者想要为某一个类添加类属性，需要先定位到该类，思路跟MATLAB是一样的。一般生成的新类，其从属于 model 分析组。如果想将某类添加到制定的 group 中，需要使用 addtogroup 之类的函数。

改变 groupscope 的方法如下：

- groupscope("::model"); # change the group scope back to the model
- groupscope("dimer"); # same as groupscope("::model::dimer");
  # assuming you are in the default location

1.3 添加类

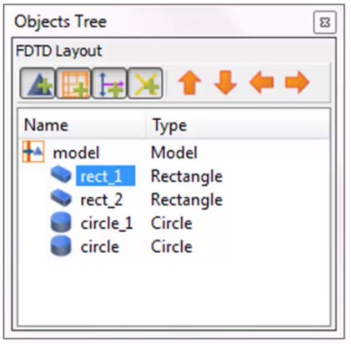
Adding objects

- General format: 'add' + 'object name'

1.4 选择类

Selecting object

- Adding an object automatically selects the object
- select; # select existing object
- shiftselect; # select new object keeping previously selected objects
- selectall; # select all object
- selectpartial; # select objects with a given partial name



select('rect_1');

1.5 设置类属性

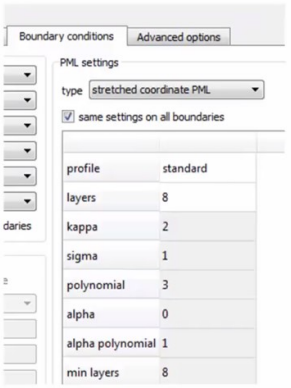
Setting properties

- set('property_name',value);
- setnamed('object_name','property_name',value); # same as select + set
- Property names are mostly the same as they appear in the Edit object window, but there are exceptions.

```
set('pml profile',1); # 1: standard, 2: stabilized
                      # 3: steep angle, 4: custom

set('same settings on all boundaries',true);
```

- ?set; # returns a list of the properties of the selected object(s)
  ?get; # same as ?set;
- ?get('parameter_name'); # returns the value of the parameter
- getnamed('object_name','property_name'); # same as select + get



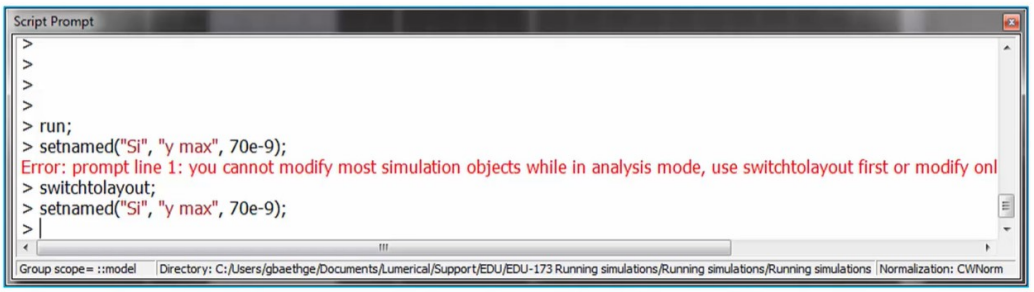
2. 参数化扫描

2.1 通过 for 循环加 run 进行参数化扫描

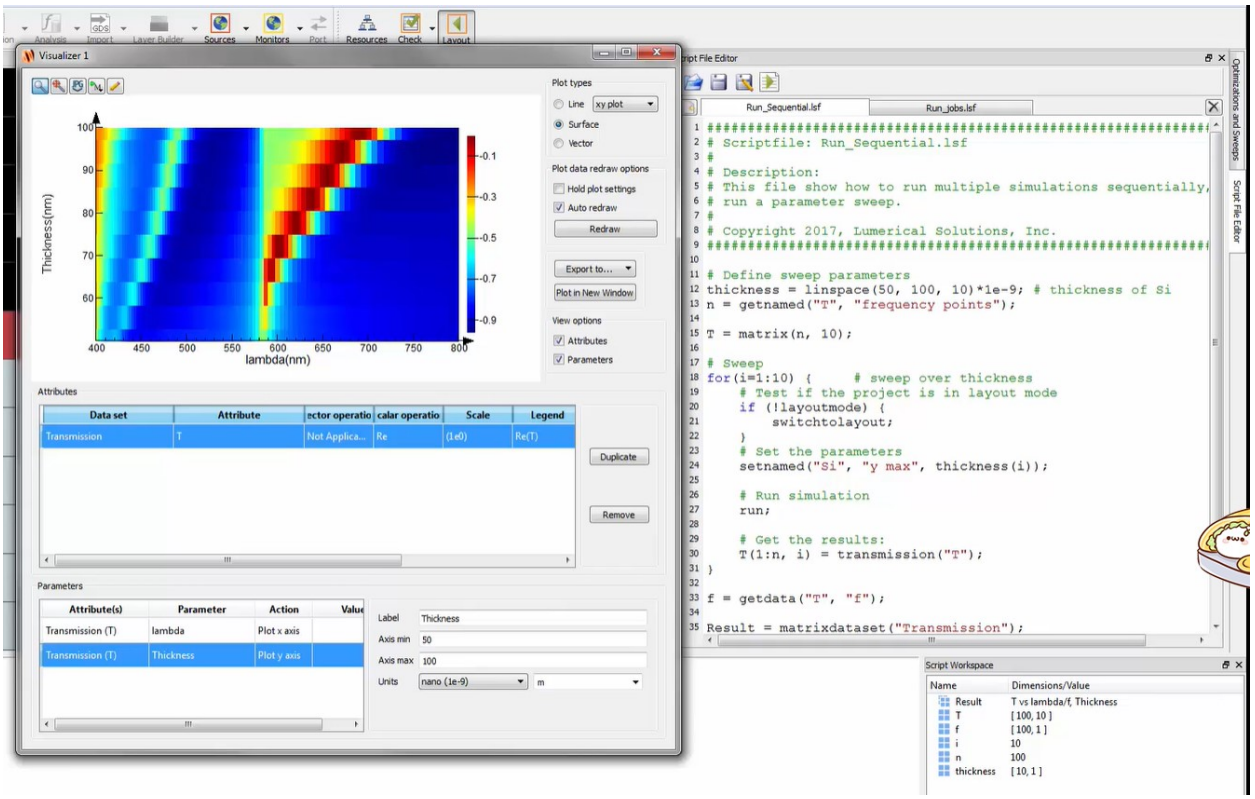
设置好模型后（无论是通过手动设置，还是通过code设置），点击工具栏的`run`按钮，或者在命令行窗口输入`run`，都可以启动运行。运行之后，系统处于分析状态。此时多数类的属性无法设置，如需设置属性，则需要将状态切回layout状态：

Get back to layout mode

- switchtolayout



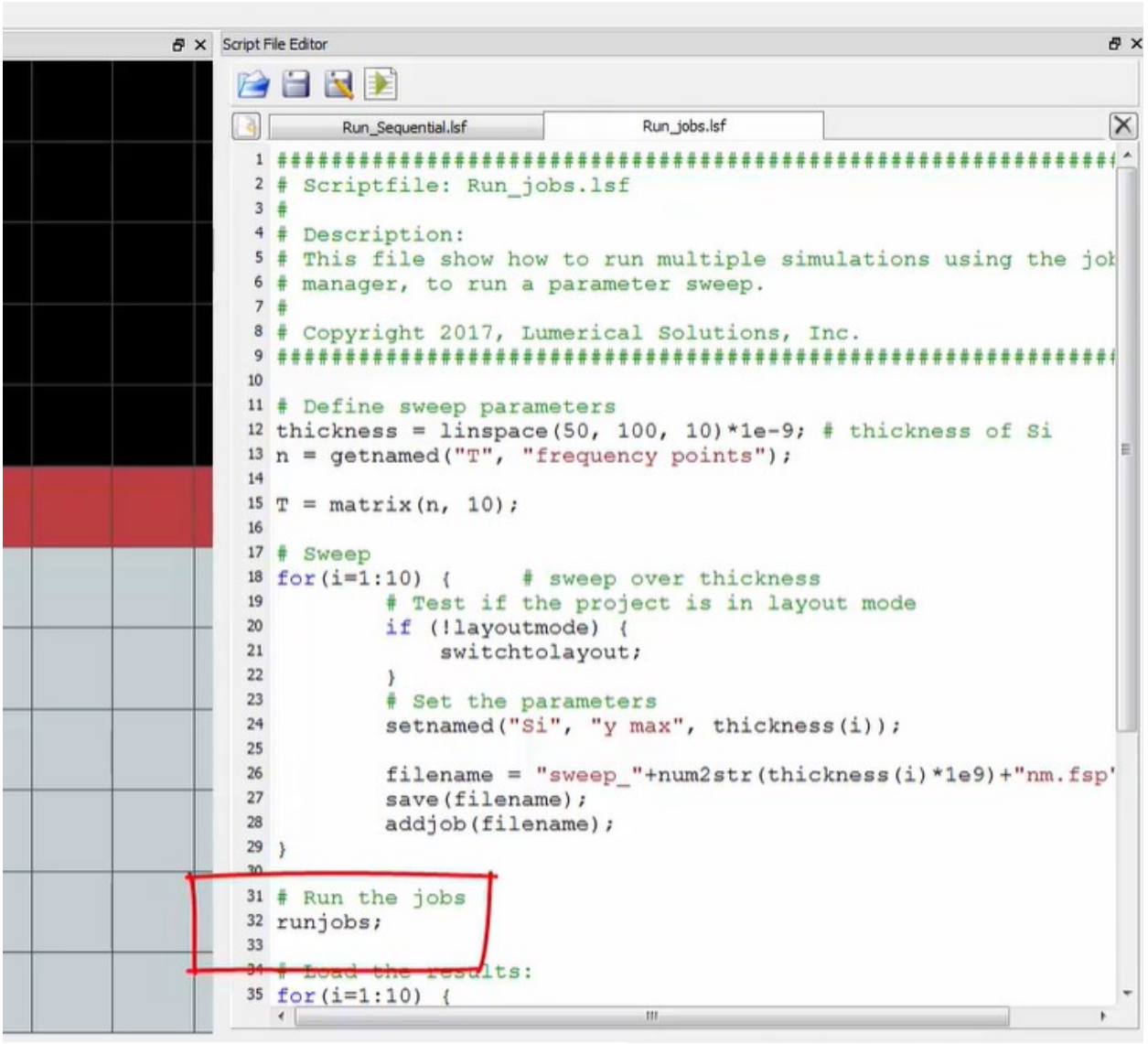
下图是一个通过script进行参数化扫描的案例：



可见，为了更改参数设置，需要切回layout状态。

2.3 通过 job manager（进行参数化扫描作业管理器）进行参数化扫描

job manager 可以同时运行多个添加的文件。添加文件的方法是 addjob 函数。通过 clearjobs 可以将序列中所有的 jobs 都清除掉，而 runjobs 则可以运行序列中的所有 jobs。下面演示通过 job manager 的方法进行参数化扫描：



其结果与前面通过 for 循环依次运行的参数化扫描操作是一样的。

2.4 通过runsweep进行参数化扫描

runsweep 函数运行所有的任务，而 runsweep('taskname') 则运行指定的任务。没太听懂这个。

3. 访问运行结果并可视化

3.1 访问运行结果

- getdate 和 getresult

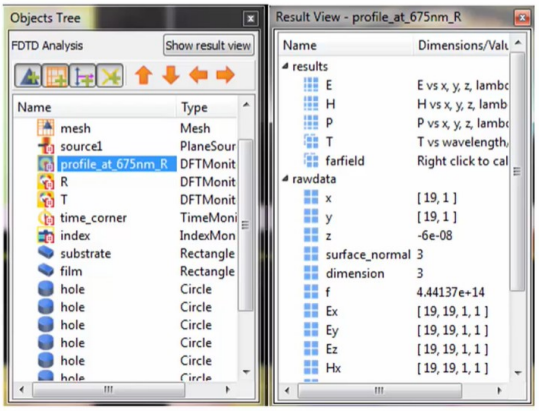
getdata / getresult

- getdata: get individual element in matrix form

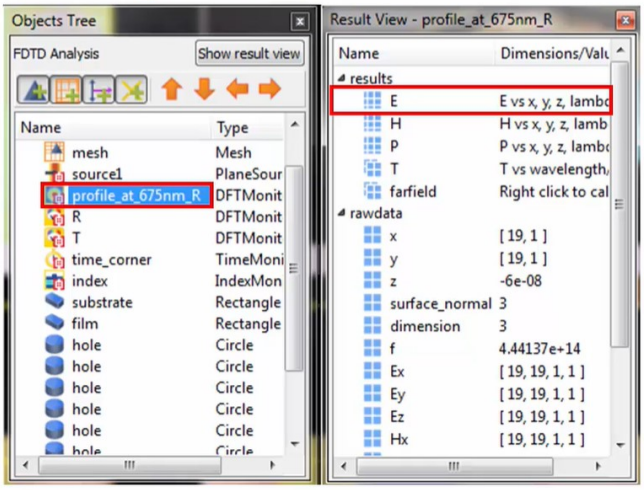
```
out = getdata('obj_name','data_name');
```

- getresult: get dataset

```
out = getresult('obj_name','dataset_name');
```



getdata / getresult




```
mname = 'profile_at_675_R';

x = getdata(mname,'x');
y = getdata(mname,'y');
Ex = getdata(mname,'Ex');
Ey = getdata(mname,'Ey');
Ez = getdata(mname,'Ez');

E = getresult(mname,'E');
x = E.x;
y = E.y;
Ex = E.Ex;
Ey = E.Ey;
Ez = E.Ez;
```

- 特殊命令

Commands for retrieving specific data from monitor

- getelectric: returns $|E_x|^2+|E_y|^2+|E_z|^2$ from the monitor.

```
out = getelectric('monitor name');
```

- getmagnetic: returns $|H_x|^2+|H_y|^2+|H_z|^2$ from the monitor.

```
out = getmagnetic('monitor name');
```

- transmission: returns the fraction of power transmitted through the monitor with respect to the source power.

```
out = transmission('monitor name');
```

需要注意的是，透射方向为x、y、z正方向时，透射率为正，反之为负。

- 获得优化和参数扫描的结果

此时使用 getsweepdata 和 getsweepresult 两个命令，具体操作如上。需要注意的是，这两个命令具体能访问哪些参数，可以通过在命令前面加一个？ 来询问。

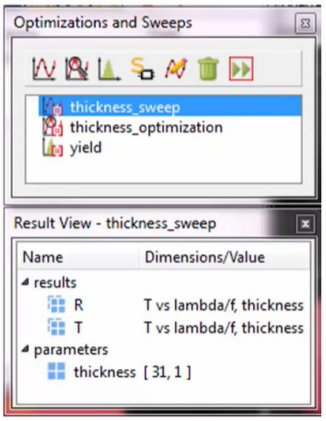
Getting data from Optimizations and Sweeps

- getsweepdata - gets raw data from the Optimization and Sweeps objects

```
?getsweepdata('thickness_sweep');
R
T
thickness
obj = 'thickness_sweep';
R = getsweepdata(obj,'R'); # reflection data
thickness = getsweepdata(obj,'thickness');
```

- getsweepresult - gets dataset from the Optimization and Sweeps objects

```
obj = 'thickness_sweep';
R = getsweepresult(obj,'R'); # reflection dataset
R = R.T;                     # reflection data
thickness = R.thickness;     # thickness data
```



3.2 可视化

- plot

plot

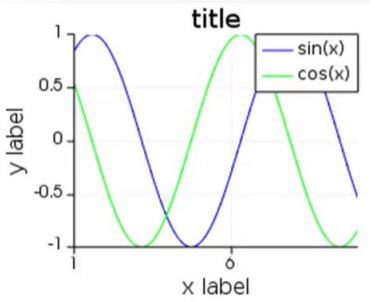
- Creates a line plot. All data sets must be sampled on the same position vector.

```
x=1:0.1:10;
y1=sin(x);
y2=cos(x);

plot(x,y1);

plot(x,y1,'x label', 'y label', 'title');

plot(x,y1,y2,'x label', 'y label', 'title');
legend('sin(x)','cos(x)');
```



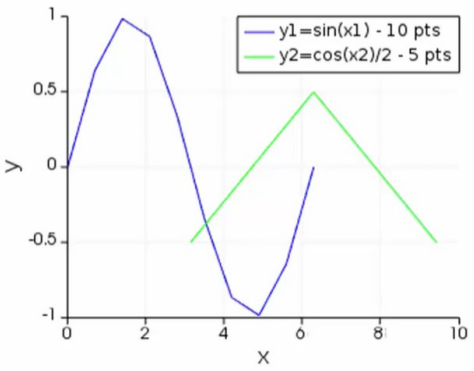
- plotxy

plotxy

- Creates line plots. In particular, the data sets can be sampled on different position vectors.

```
x1=linspace(0,2*pi,10);
y1=sin(x1);
x2=linspace(pi,3*pi,5);
y2=cos(x2)/2;

plotxy(x1,y1,x2,y2,"x","y");
legend("y1=sin(x1)- 10 pts",
       "y2=cos(x2)/2 - 5 pts");
```



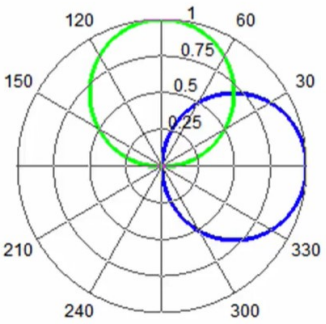
- polar

polar

- Creates polar plots. All data sets must be sampled on the same array of angle values.

```
theta = linspace(0,2*pi,100);
r = cos(theta);
polar(theta,r);

theta = linspace(0,2*pi,100);
r1 = cos(theta);
r2 = sin(theta);
polar(theta,r1,r2);
```



- polar2

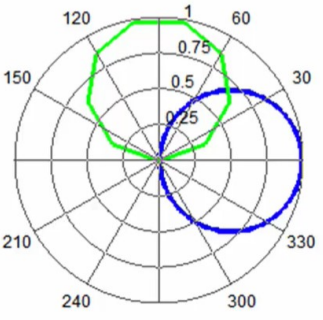
polar2

- Creates polar plots. The data sets can be sampled on different arrays of angle values.

```
theta1 = linspace(0,2*pi,30);
r1 = cos(theta1);

theta2 = linspace(0,pi,10);
r2 = sin(theta2);

polar2(theta1,r1,theta2,r2);
```

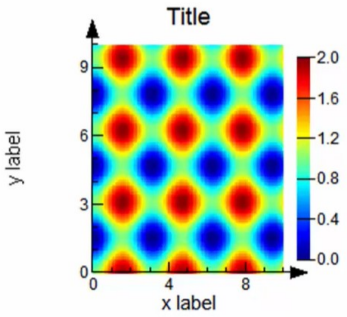


- image

image

- Creates 2D image plots.

```
x=linspace(0,10,100);
y=linspace(0,10,100);
X = meshgridx(x,y); # mesh grid points in x-direction
Y = meshgridy(x,y); # mesh grid points in y-direction
Z = sin(X)^2+cos(Y)^2;
image(x,y,Z,"x label","y label","Title");
```

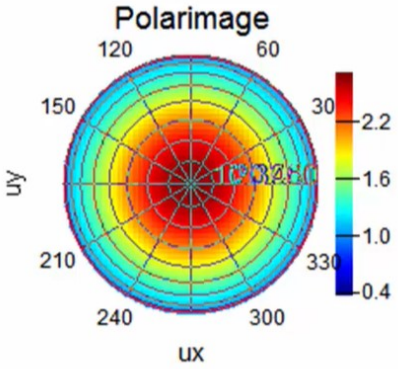


- polarimage

polarimage

- Creates 2D polar image plots. This is typically used to plot far field data.
- Syntax: polarimage(ux,uy,data);

```
ux=linspace(-1,1,51);
uy=linspace(-1,1,51);
Ux=meshgridx(ux,uy);
Uy=meshgridy(ux,uy);
data = exp( 1-Ux^2-Uy^2);
```

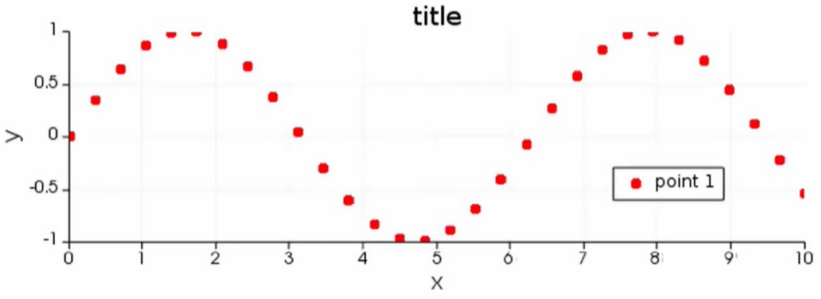



```
polarimage(ux,uy,data,"ux","uy","Polarimage");
```

- plot options

Plotting options

```
x=linspace(0,10,30);
y=sin(x);
plot(x,y,"x","y","title", "plot type = point, color=red, marker style = o");
```



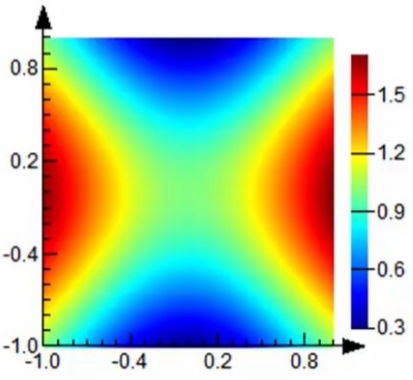
- setplot

setplot

```
x=-1:0.01:1; y=x;
image(x,y,sin(meshgridx(x,y))^2+cos(meshgridy(x,y))^2);

?setplot;
 x min
 x max
 y min
 y max
 title
 grey scale
 colorbar min
 colorbar max

setplot("title","my figure");
```



- 其他命令

Other useful plotting commands

- legend: adds a legend to a line plot.

```
legend("legend1","legend2",..., "legendn");
```

- holdon / holdoff: switches on/off the mode to hold multiple mathematical functions on the same figure.
- selectfigure: selects a figure, which will be shown on screen (gives it focus).

```
selectfigure('3'); # selects figure '3'
selectfigure;      # selects last figure
```

- exportfigure

```
exportfigure('filename',xres,yres);
```

4. 新建和保存文件

4.1 创建和保存文件

- 新建文件：

使用 newproject 或者 newproject(option)，第二种方法是基于某一存在的模板创建一个新文件，而第一个则是创建一个全新文件。

- 保存文件

使用 save 和 save(filename) 保存文件，若不指明保存文件名和文件目录，则在此文件夹下保存此文件，一如 MATLAB。

- 加载文件

使用 load 函数加载文件，如 MATLAB。

4.2 文本文件

使用 write 函数新建一个记事本并向其中写入文本。使用 num2str 函数将数字转换成字符格式。使用 readdate 函数从一个文本中读入文本内容。

4.3 保存工作区的数据

- 保存为 .ldf 格式的数据

使用 savedata(filename) 保存工作区中所有数据，使用 savedata(filename, var1, var2, ...) 保存工作区部分数据。使用 loaddate(filename) 加载保存的数据。

- 保存为 .mat 格式的数据

方法如前，只不过函数使用 matlabsave 和 matlabload。