Saghar Gorjiduz | 95243096
Tara Barghian | 97243009
Mohammad Hashemi | 97243073
Instructor: Dr. Yaser Shekofte
May 17, 2020

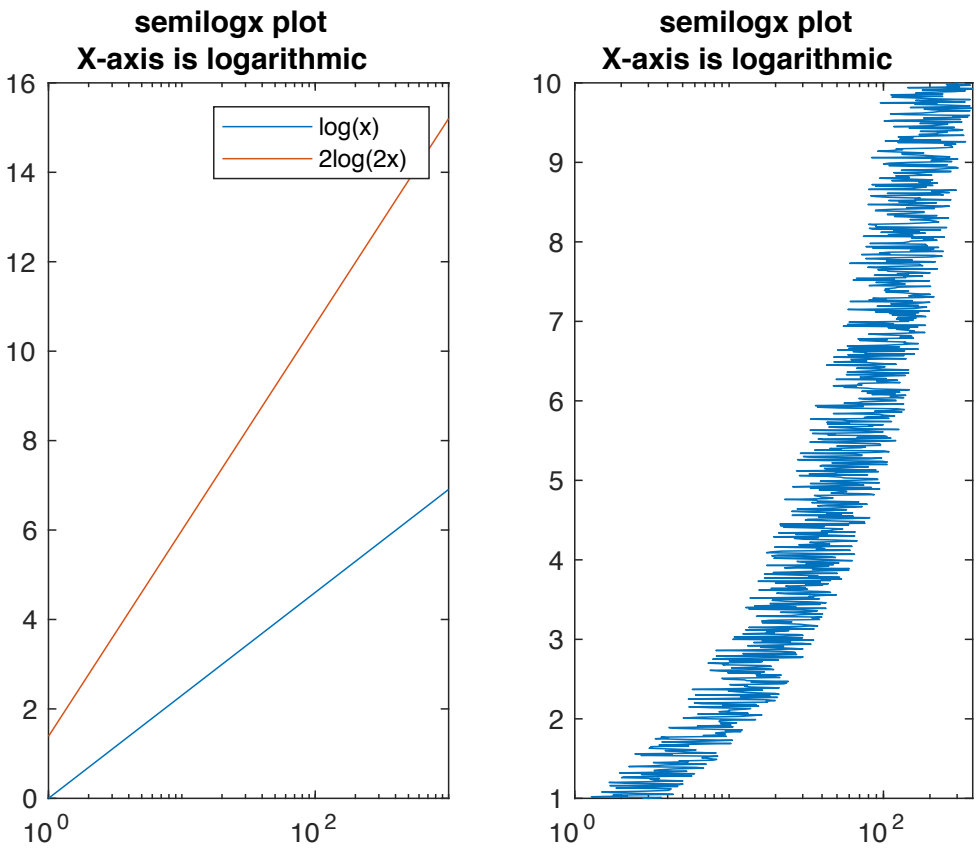
Matlab #3 Report

Matlab Programming Workshop | Spring 99

3-1 : Semilogx & Semilogy

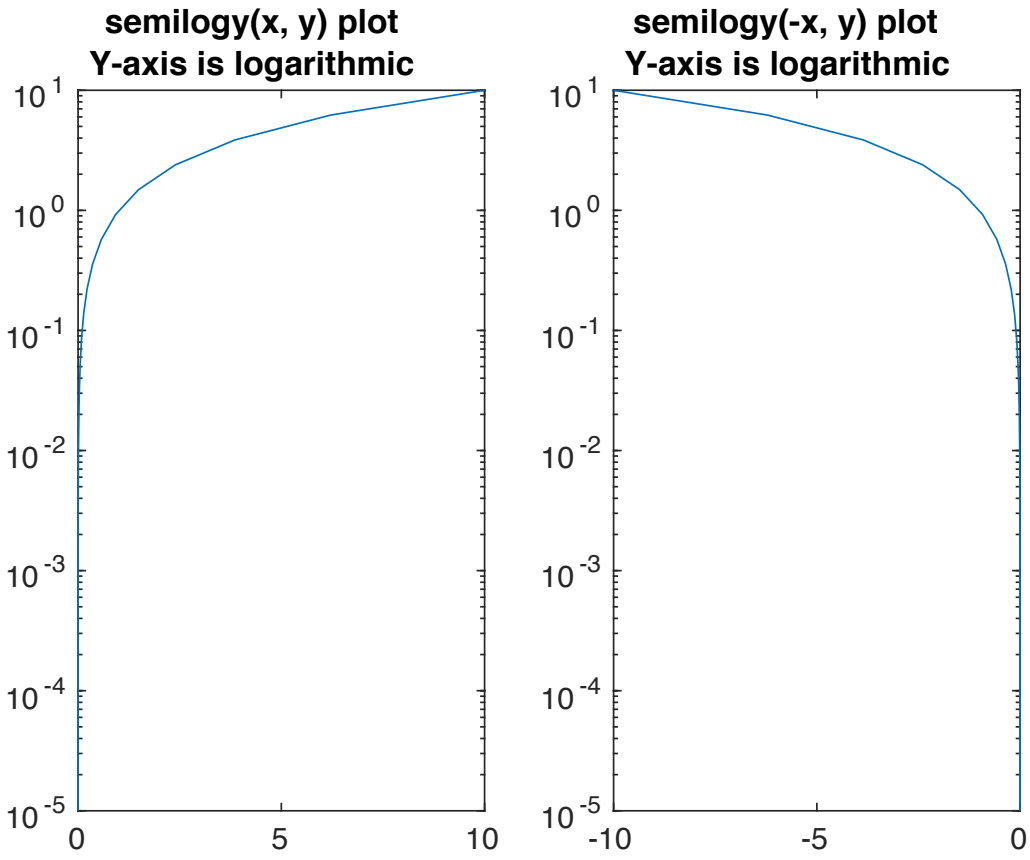
Name	Syntax	Description
Semilogx	`semilogx(Y)`	`semilogx` plot data as logarithmic scales for the *x*-axis. `semilogx(Y)` creates a plot using a base 10 logarithmic scale for the *x*-axis and a linear scale for the *y*-axis.
	`semilogx(X1,Y1,...)`	Plots all `Yn` versus `Xn` pairs.
	`semilogx(X1,Y1,LineSpec)`	Plots all lines defined by the `Xn,Yn,LineSpec` triples. `LineSpec` determines line style, marker symbol, and color of the plotted lines.
	`semilogx(...,'PropertyName', PropertyValue,...)`	Sets property values for all charting lines created by `semilogx`.
	`semilogx(ax,...)`	reates the line in the axes specified by ax instead of in the current axes (`gca`). The option ax can precede any of the input argument combinations in the previous syntaxes.
	`h = semilogx(...)`	Return a vector of chart line objects.

```
clc; clear;
x1 = 0:1000;
y1 = log(x1);

x2 = 0:1000;
y2 = 2 * log(2 * x2);

figure(1)
 subplot (1,2,1);
semilogx(x1, y1,  x2, y2);
title(["semilogx plot", "X-axis is logarithmic"]);
legend('log(x)', '2log(2x)');

 subplot (1,2,2);
 x = 1:0.01:10;
 y = (x .* (1 + rand (size (x)))) .^ 2;
 semilogx (y, x);
 title (["semilogx plot", "X-axis is logarithmic"]);
```



Name	Syntax	Description
Semilogy	`semilogy(Y)`	`Semilogy` plot data as logarithmic scales for the *y*-axis. `semilogy(Y)` creates a plot using a base 10 logarithmic scale for the *y*-axis and a linear scale for the *x*-axis.
	`semilogy(X1,Y1,...)`	Plots all `Yn` versus `Xn` pairs.
	`semilogy(X1,Y1,LineSpec)`	Plots all lines defined by the `Xn,Yn,LineSpec` triples. `LineSpec` determines line style, marker symbol, and color of the plotted lines.
	`semilogy(...,'PropertyName', PropertyValue,...)`	Sets property values for all charting lines created by `semilogy`.
	`semilogy(ax,...)`	reates the line in the axes specified by ax instead of in the current axes (`gca`). The option ax can precede any of the input argument combinations in the previous syntaxes.
	`h = semilogy(...)`	Return a vector of chart line objects.

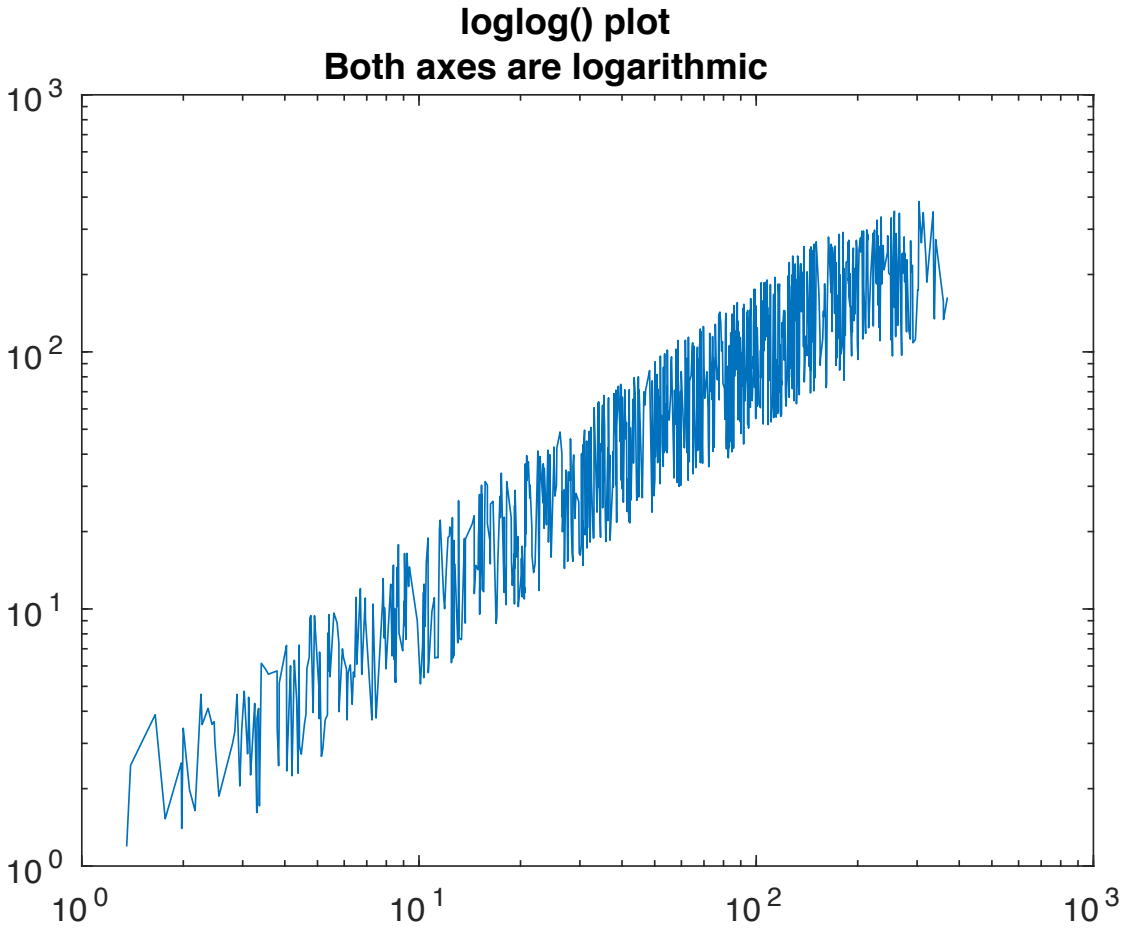
```
clc; clear;

figure(1)
x = logspace (-5, 1, 30);
y = logspace (-5, 1, 30);

subplot (1,2,1);
semilogy (x, y);
title (["semilogy(x, y) plot", "Y-axis is logarithmic"]);

subplot (1,2,2);
semilogy (-x, y);
title (["semilogy(-x, y) plot", "Y-axis is logarithmic"]);
```


semilogy(x, y) plot
Y-axis is logarithmic
10 1
10 0
10 -1
10 -2
10 -3
10 -4
10 -5
0
5
10
semilogy(-x, y) plot
Y-axis is logarithmic
10 1
10 0
10 -1
10 -2
10 -3
10 -4
10 -5
-10
-5
0

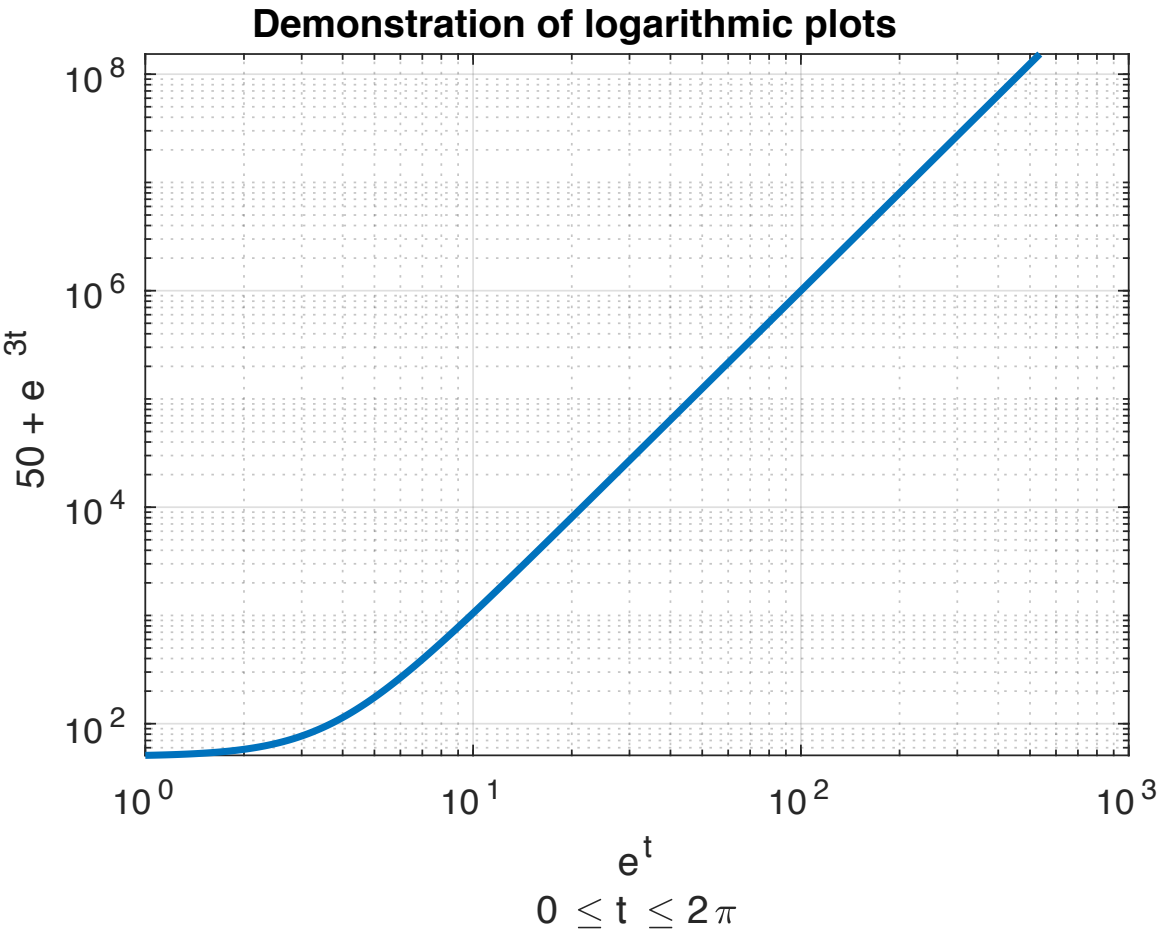
3-2 : loglog

Name	Syntax	Description
loglog	`loglog(X,Y)`	Plots *x*- and *y*-coordinates using logarithmic scales on the *x*-axis and the *y*-axis.
	`loglog(X,Y,LineSpec)`	Creates the plot using the specified line style, marker, and color.
	`loglog(X1,Y1,...,Xn,Yn)`	Plots multiple pairs of *x*- and *y*-coordinates on the same set of axes. Use this syntax as an alternative to specifying coordinates as matrices.
	`loglog(X1,Y1,LineSpec1,...,Xn, Yn,LineSpecn)`	Assigns specific line styles, markers, and colors to each *x*-*y* pair.
	`loglog(__,Name,Value)`	Specifies `Line` properties using one or more `Name,Value` pair arguments. The properties apply to all the plotted lines. Specify the `Name,Value` pairs after all the arguments in any of the previous syntaxes.
	`loglog(ax,__)`	Displays the plot in the target axes. Specify the axes as the first argument in any of the previous syntaxes.
	`lineobj = loglog(__)`	Returns a Line object or an array of Line objects. Use lineobj to modify properties of the plot after creating it.

```
clc; clear;

figure(1)
t = 1:0.01:10;
 x = sort ((t .* (1 + rand (size (t)))) .^ 2);
 y = (t .* (1 + rand (size (t)))) .^ 2;
 loglog (x, y);
 title ({"loglog plot", "Both axes are logarithmic"});
```



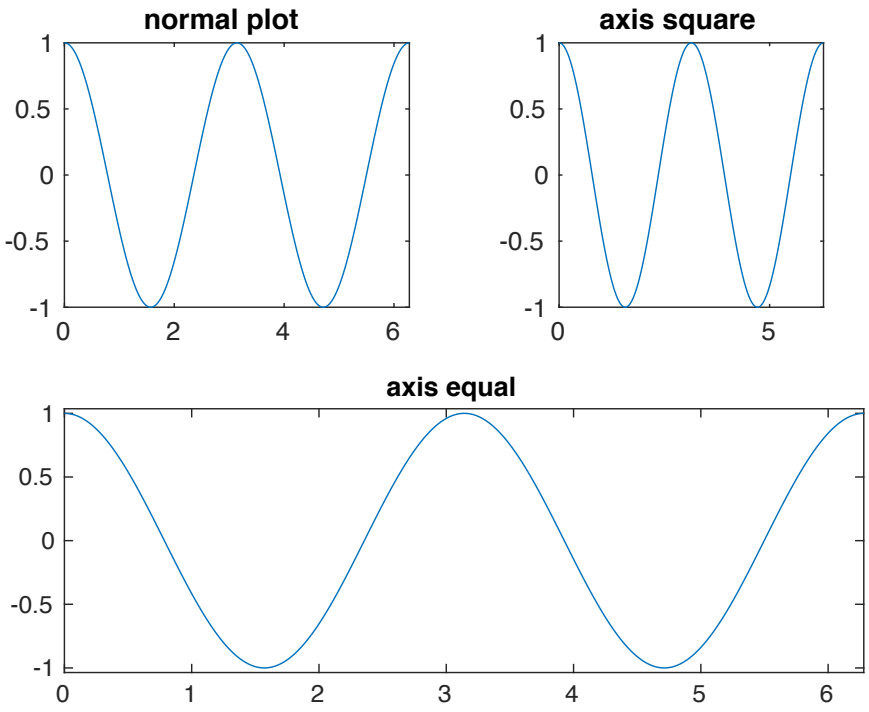
```
clear; clc; close all

t = 0 : 2*pi/360 : 2*pi;

% Define values along your x-axis
x = exp(t);
% Define values along your y-axis
y = 50 + exp(3*t);

loglog(x, y, 'LineWidth', 2)
grid

title('Demonstration of logarithmic plots')


xlabel([{'e^{t}'}; {'0 \leq t \leq 2\pi'}])

% Label your y-axis
ylabel('50 + e^{3t}')
```



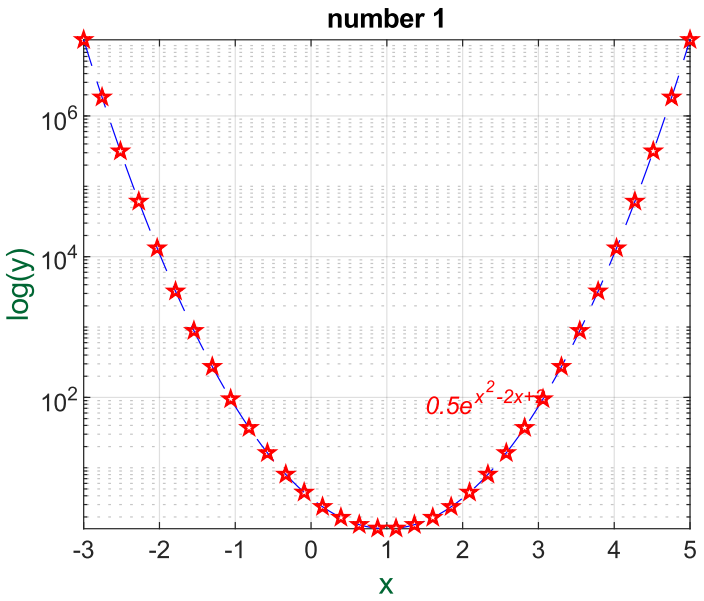
3-3 : axis

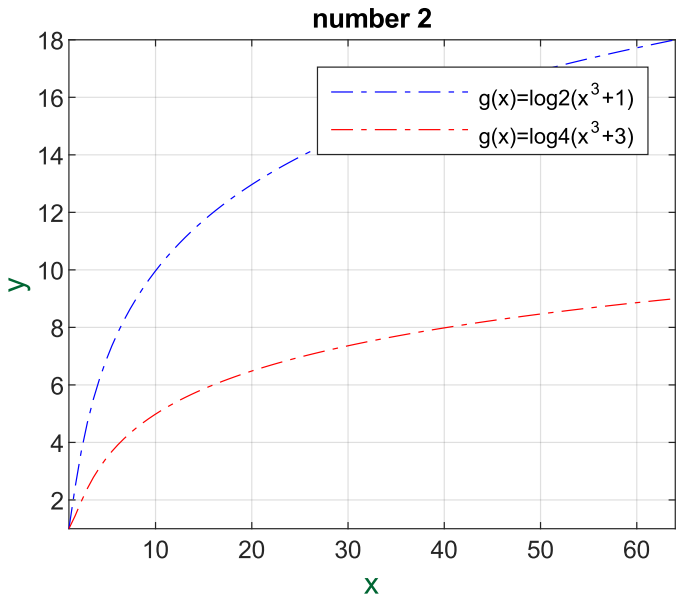
Name	Syntax	Description
axis	`axis(limits)`	Specifies the limits for the current axes. Specify the limits as vector of four, six, or eight elements.
	`axis style`	Uses a predefined style to set the limits and scaling. For example, specify the style as `equal` to use equal data unit lengths along each `axis`
	`axis mode`	Sets whether MATLAB automatically chooses the limits or not. Specify the mode as `manual`, `auto`, or one of the semiautomatic options, such as `'auto x'`
	`axis direction`	Assigns specific line styles, markers, and colors to each *x*-*y* pair.
	`Axis visibility`	Where `visibility` is `off`, turns off the display of the axes background. Plots in the axes still display. The default for `visibility` is `on`, which displays the axes background.
	`Lim = axis`	`axis` returns the *x*-axis and *y*-axis limits for the current axes. For 3-D axes, it also returns the *z*-*axis* limits. For polar axes, it returns the *theta*-axis and *r-axis* limits.

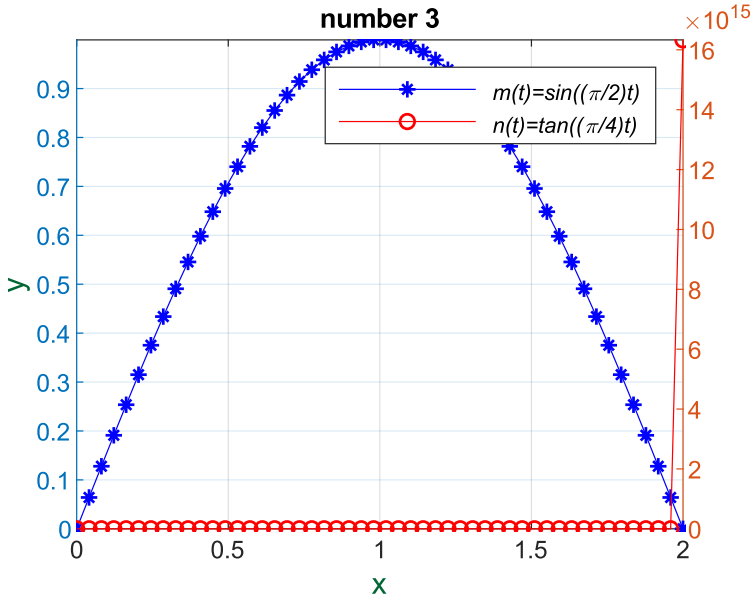
```
clear; clc; close all

t = 0:0.01:2*pi;
 x = cos (2 * t);

 subplot (2, 2, 1);
  plot (t, x);
  title ("normal plot");

 subplot (2, 2, 2);
  plot (t, x);
  title ("axis square");
  axis ("square");

 subplot (2, 2, [3, 4]);
  plot (t, x);
  title ("axis equal");
  axis ("equal");
```

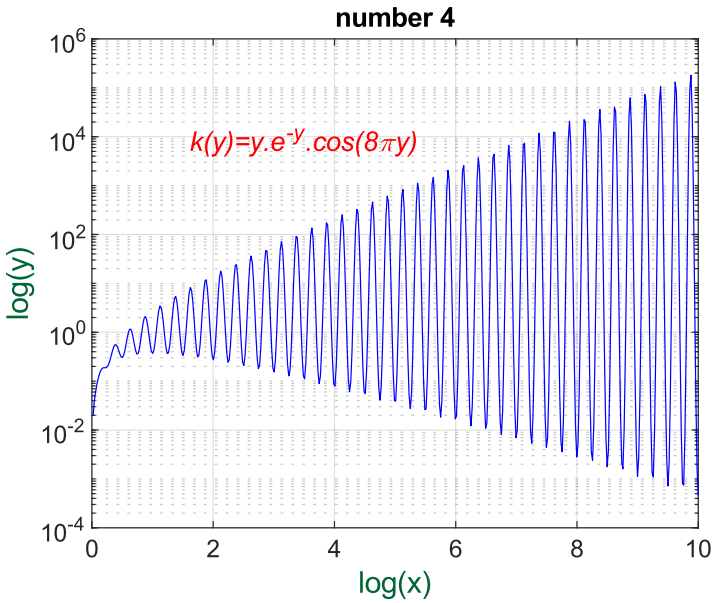


3-4 : plot & subplot



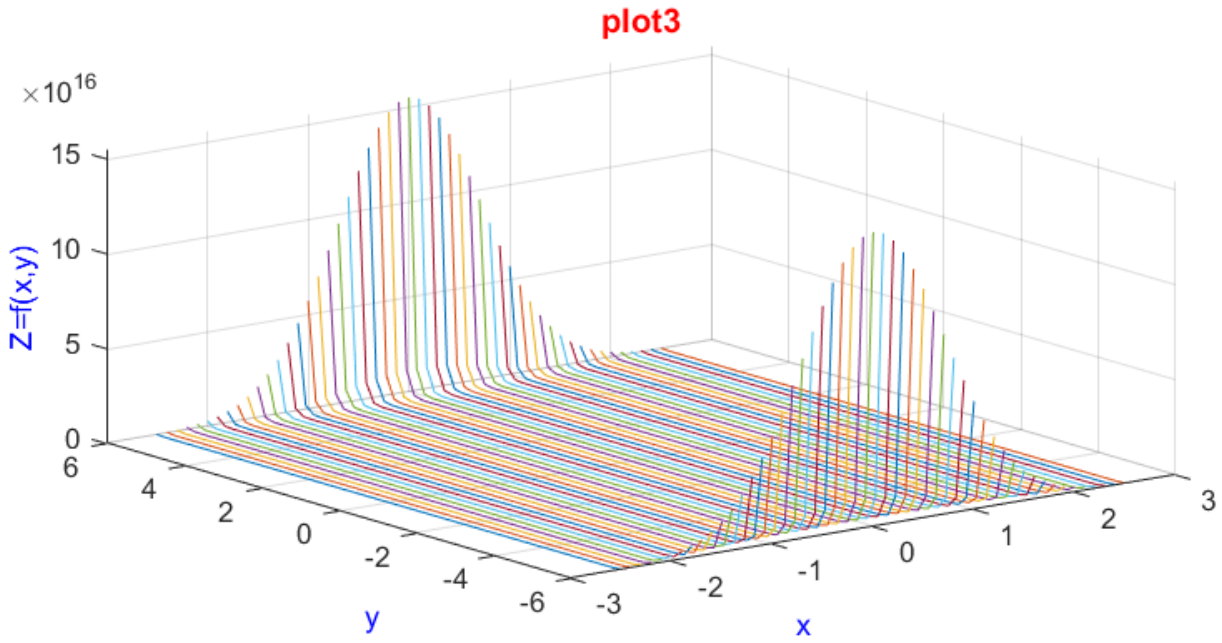


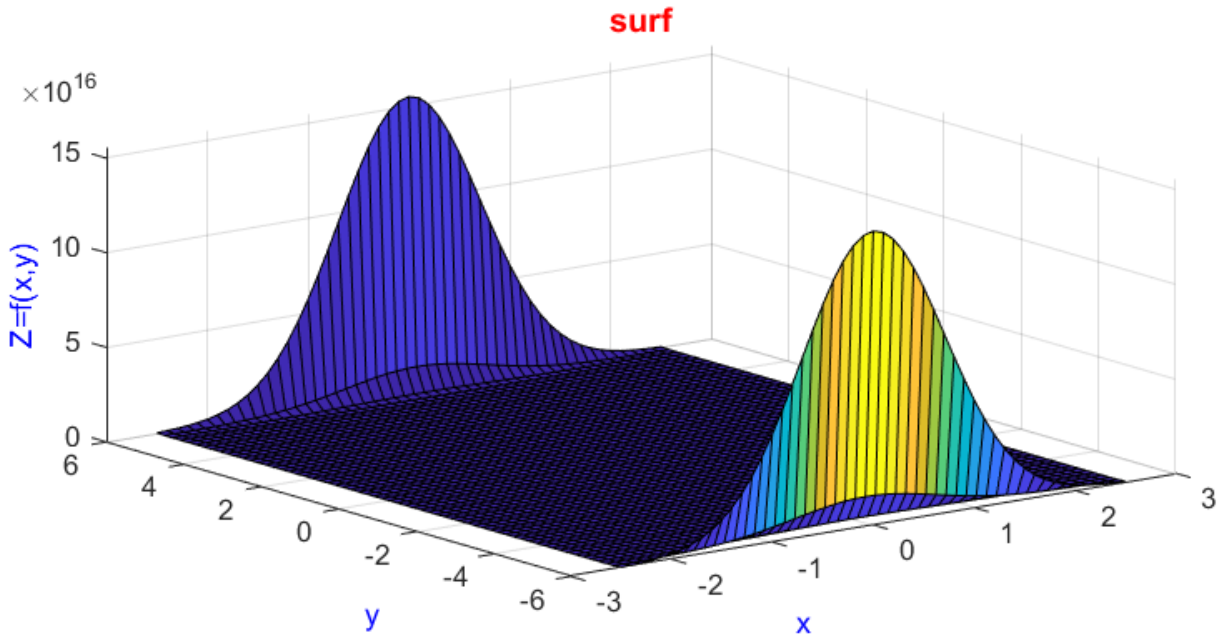




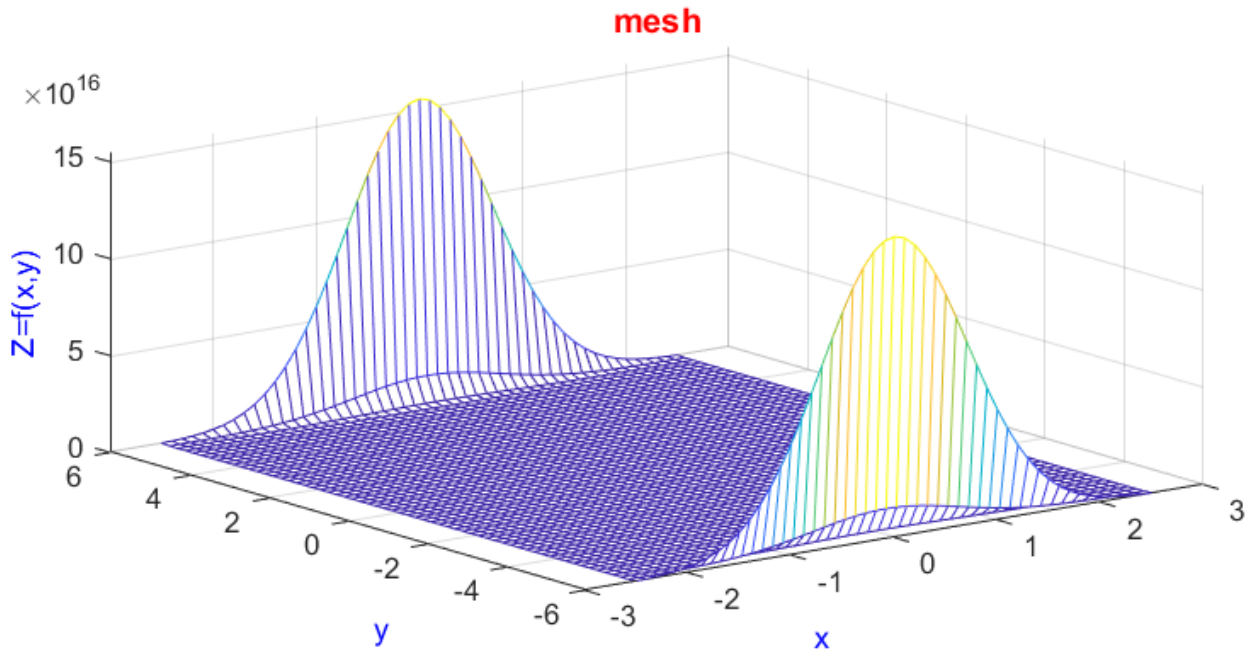
3-5 : 3D-plot

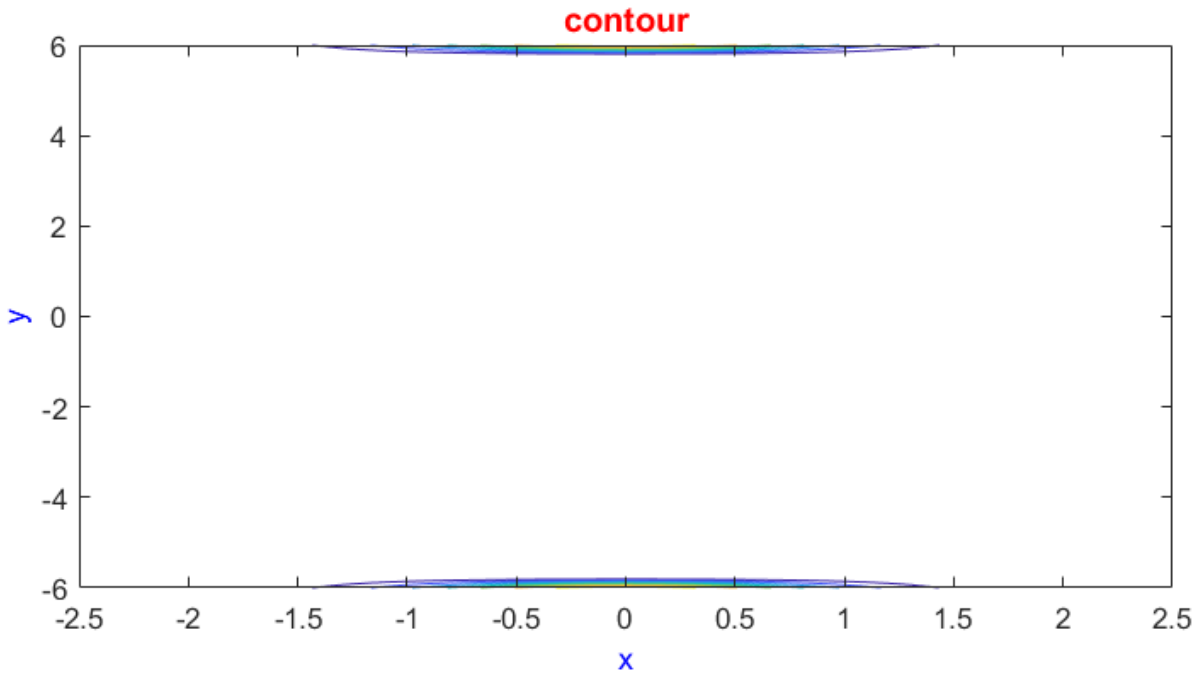
$$z = f(x,y) = y^2.e^{(-x^2+y^2)} \quad \begin{cases} -2.5 \le x \le 2.5 \\ -6 \le y \le 6 \end{cases}$$



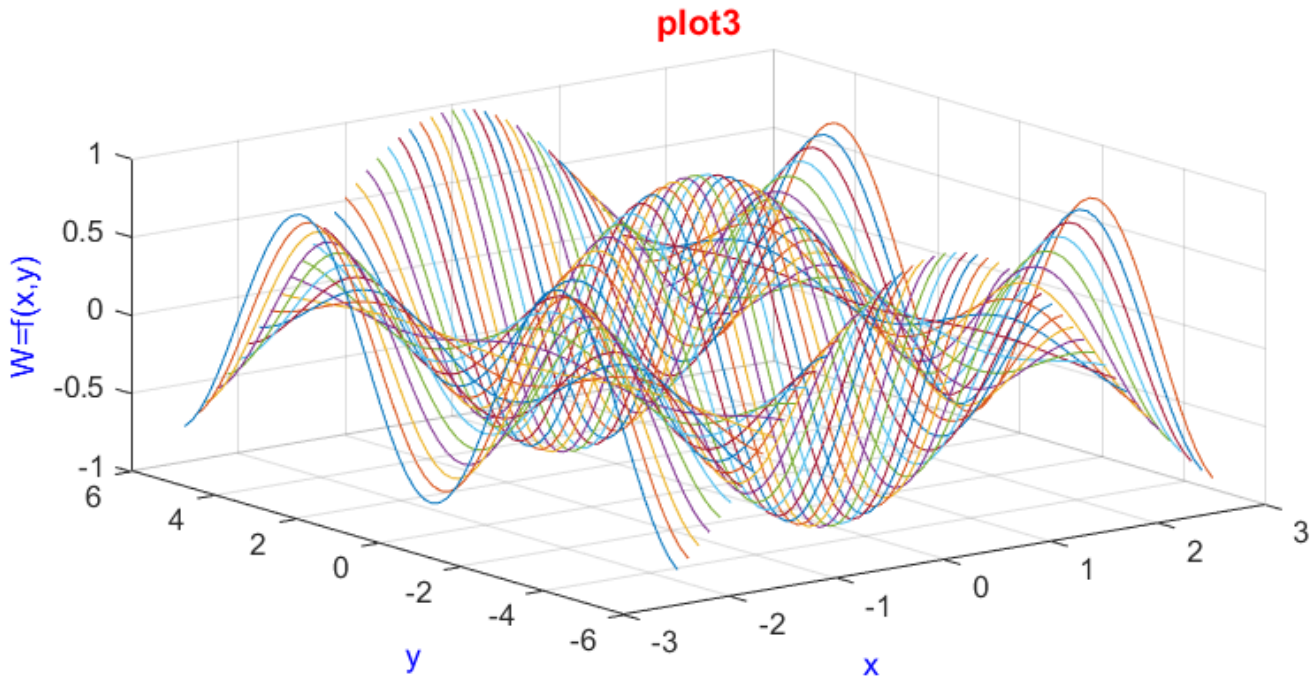


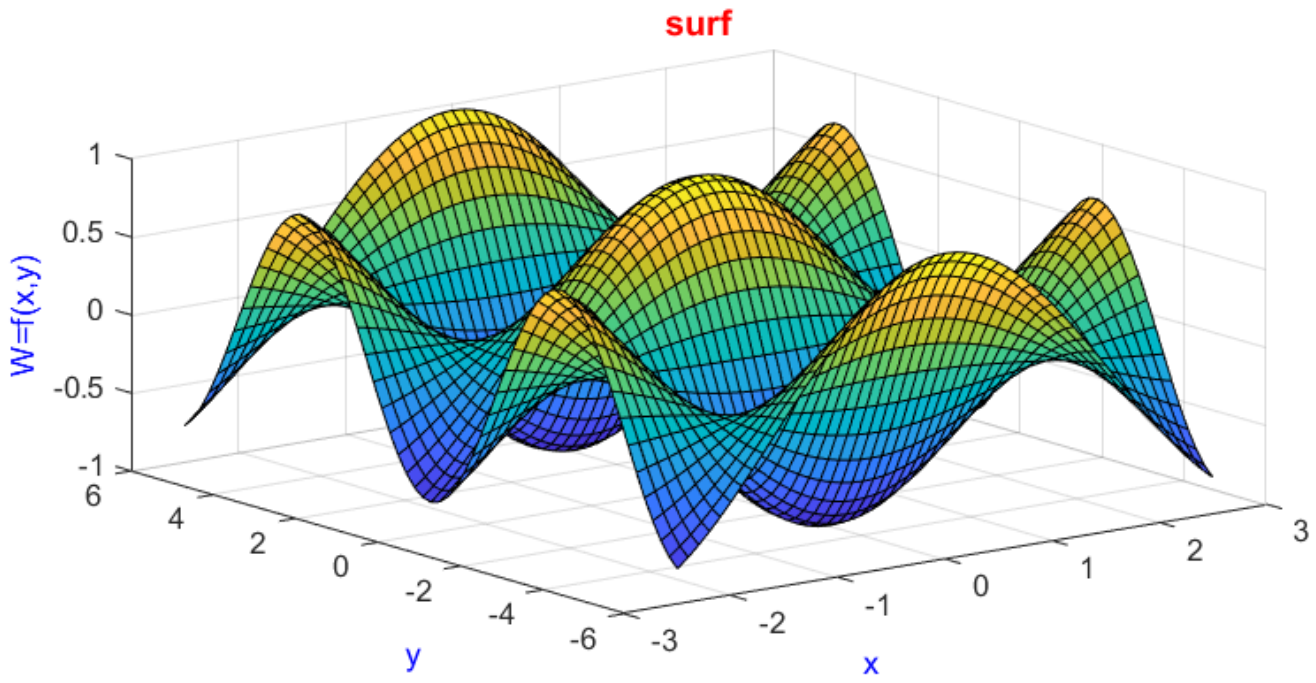
$$z = f(x,y) = y^2.e^{(-x^2+y^2)} \quad \begin{cases} -2.5 \le x \le 2.5 \\ -6 \le y \le 6 \end{cases}$$



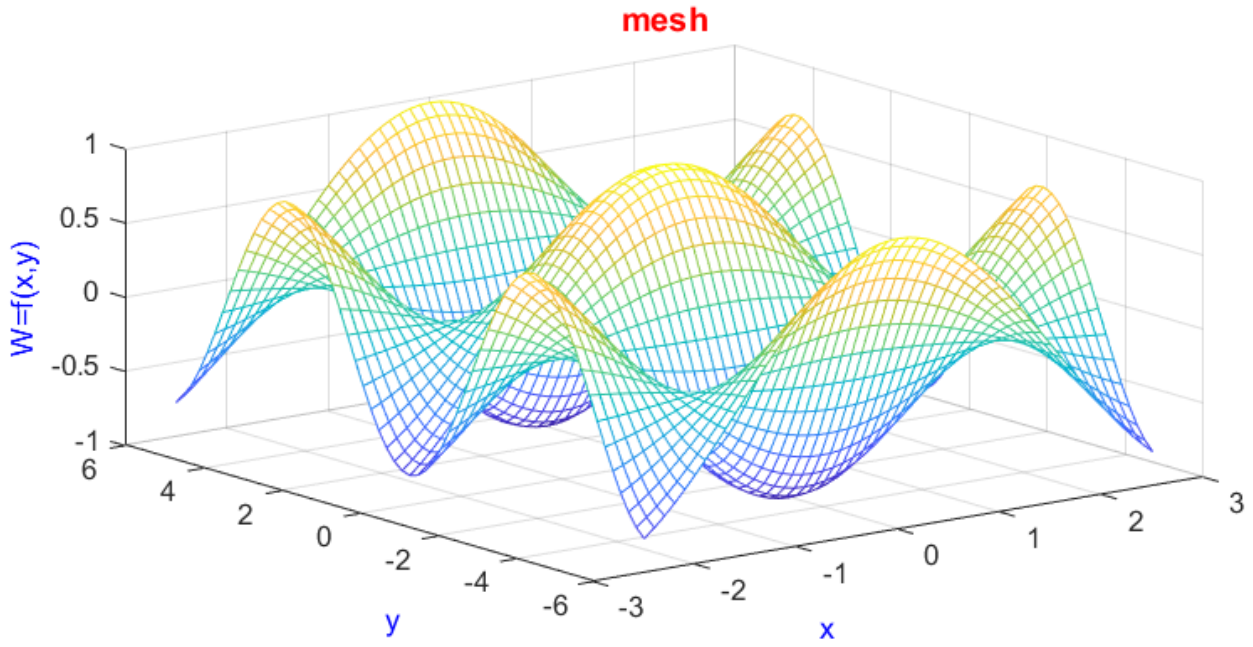


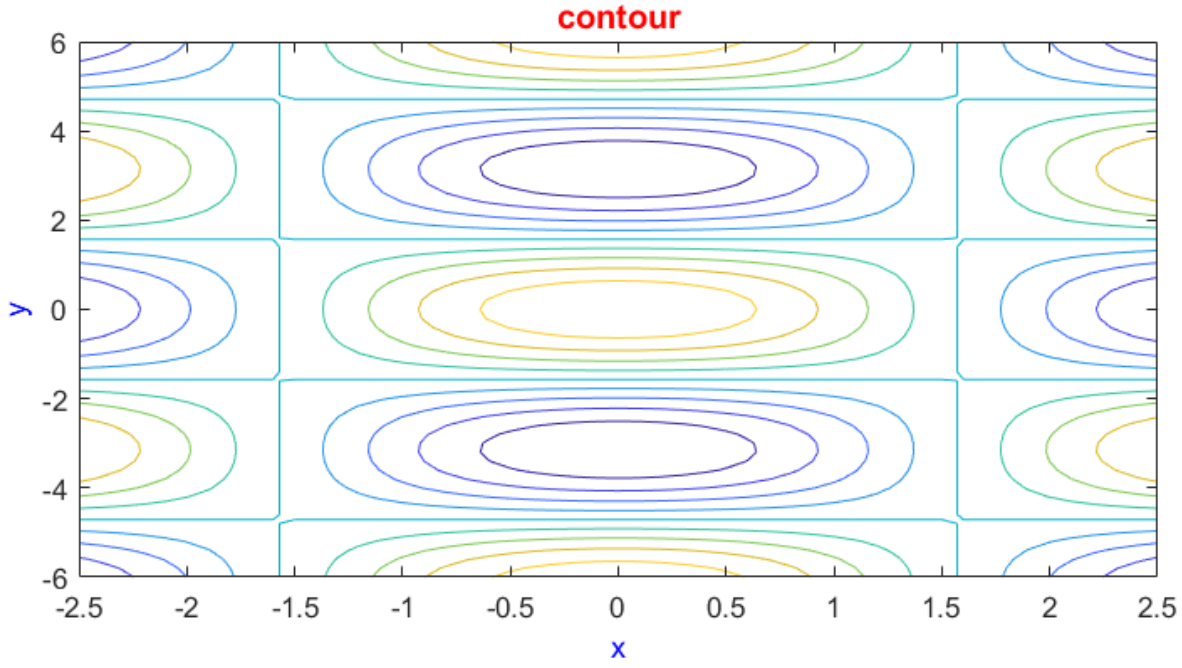
$$w = f(x,y) = \cos(x).\cos(y) \quad \begin{cases} -7 \le x \le 7 \\ -5 \le y \le 5 \end{cases}$$





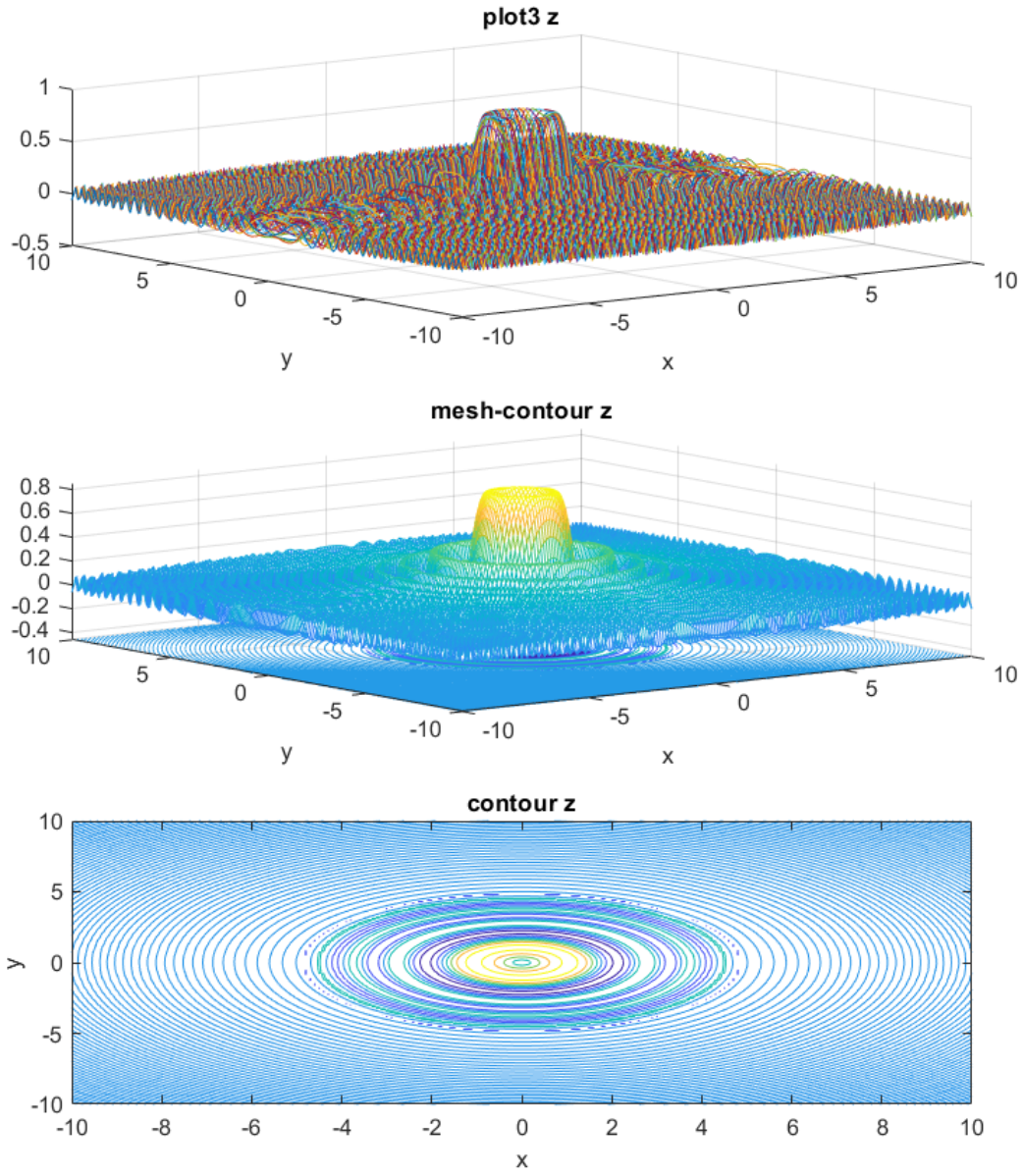
$$w = f(x,y) = \cos(x).\cos(y) \quad \begin{cases} -7 \le x \le 7 \\ -5 \le y \le 5 \end{cases}$$

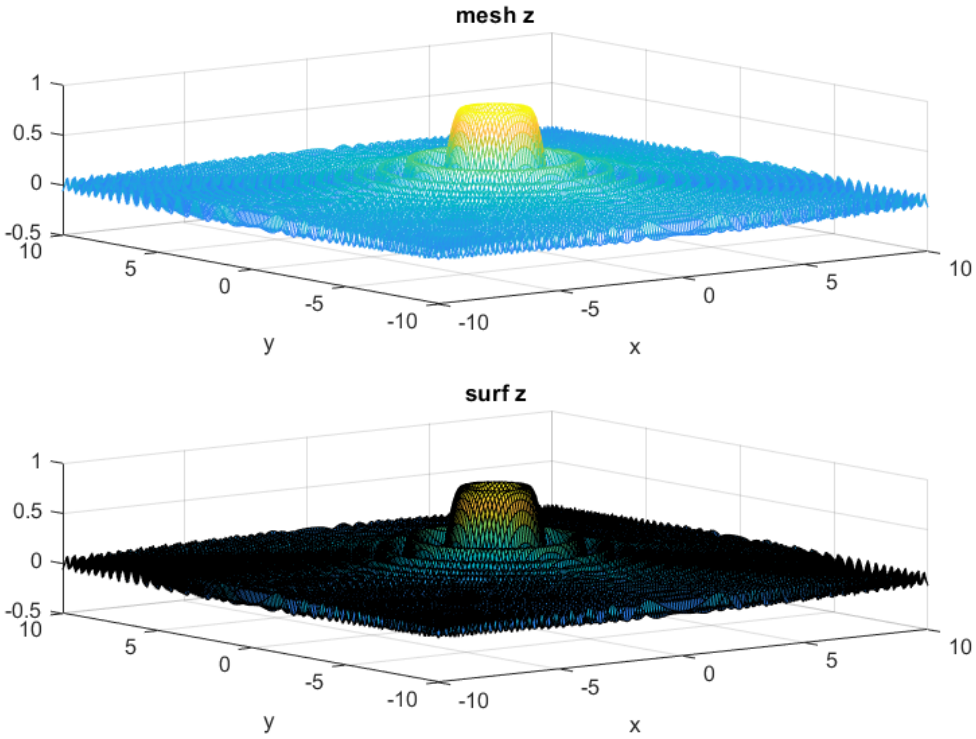




3-6 : mesh vs. contour vs. surf

`plot3` plots some markers at the specified points. It does not make a solid surface between the markers.
`surf` will create a solid surface between the points.
`meshgrid` gives every possible combination of every `x` and every `y`. It is not a plotting function. We use `meshgrid` to see some surfaces.
A `contour` plot provides a 2-dimensional view of the surface where points that have the same response are connected to produce contour lines of constant responses. `Contour` plots are useful for establishing the response values and operating conditions that you want.




mesh z
1
0.5
0
-0.5
10
5
0
-5
-10
y
-10
-5
0
5
10
x
surf z
1
0.5
0
-0.5
10
5
0
-5
-10
y
-10
-5
0
5
10
x