Morteza Kazemi | 97243054
Amir Masoud Shaker | 97243081
11/14/2021

Homework #3

Q1

Syntax	Meaning	Description
1) semilogx(X,Y) 2) semilogx(X,Y,LineSpec) 3) semilogx(X1,Y1,…,Xn,Yn) 4) semilogx(X1,Y1,LineSpec1,…,Xn,Yn,LineSpecn) 5) semilogx(Y) 6) semilogx(Y,LineSpec) 7) semilogx(__ ,Name,Value) 8) semilogx(ax, __) 9) lineobj = semilogx(__)	Semilog plot (x-axis has log scale)	1) This syntax plots x- and y-coordinates using a base-10 logarithmic scale on the x-axis and a linear scale on the y-axis. 2) This syntax creates the plot using the specified color, marker, and line style. 3) This syntax plots multiple pairs of x- and y-coordinates on the same set of axes.

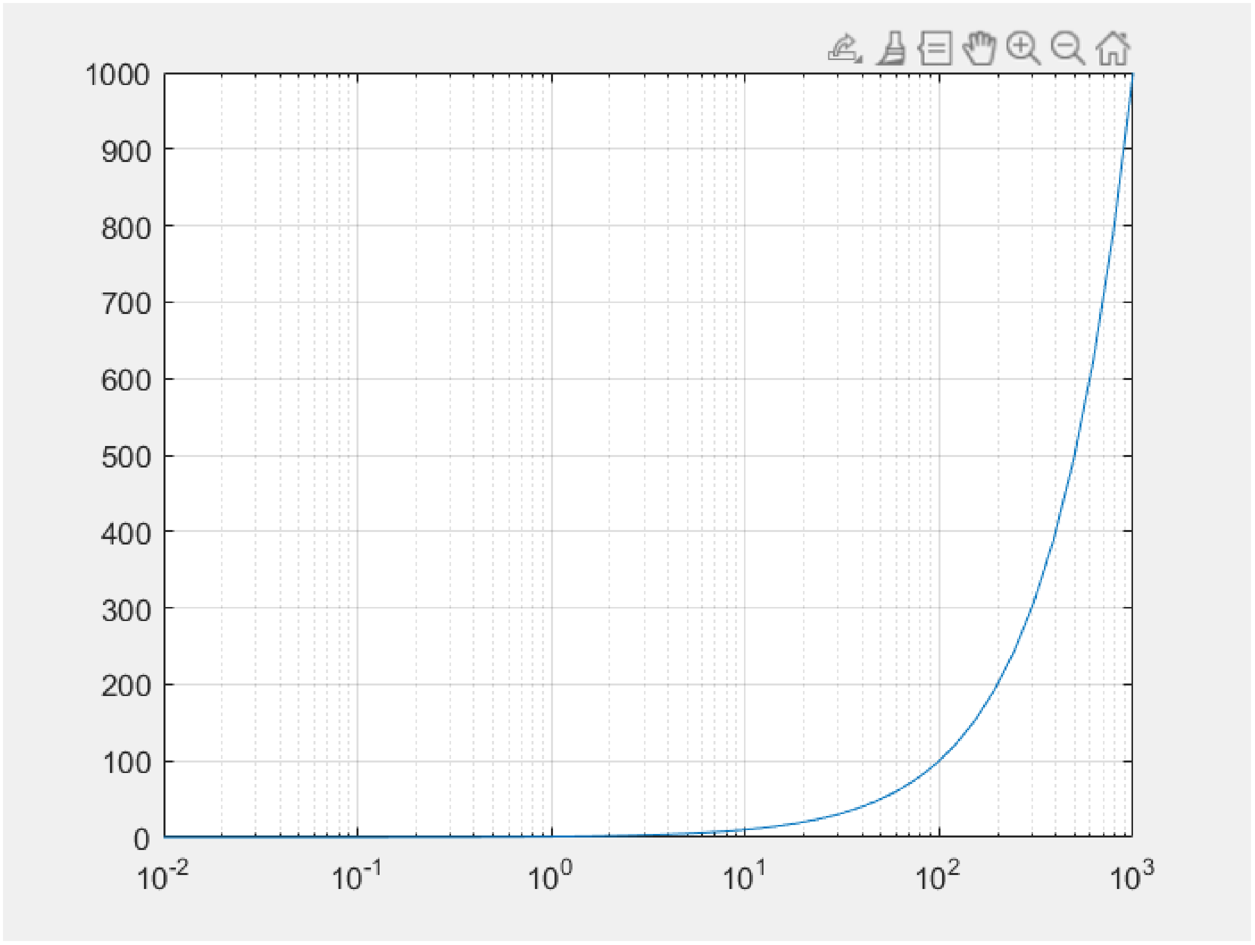
		4) This syntax assigns specific colors, markers, and line styles to each x-y pair. 5) This syntax plots Y against an implicit set of x-coordinates. Note: If Y is a vector, the x-coordinates range from 1 to length(Y). Note: If Y is a matrix, the plot contains one line for each column in Y. The x-coordinates range from 1 to the number of rows in Y. 6) This syntax specifies color, marker, and line style.

		7) This syntax specifies Line properties using one or more Name,Value pair arguments. 8) This syntax displays the plot in the target axes. 9) This syntax returns a Line object or an array of Line objects.

Examples:

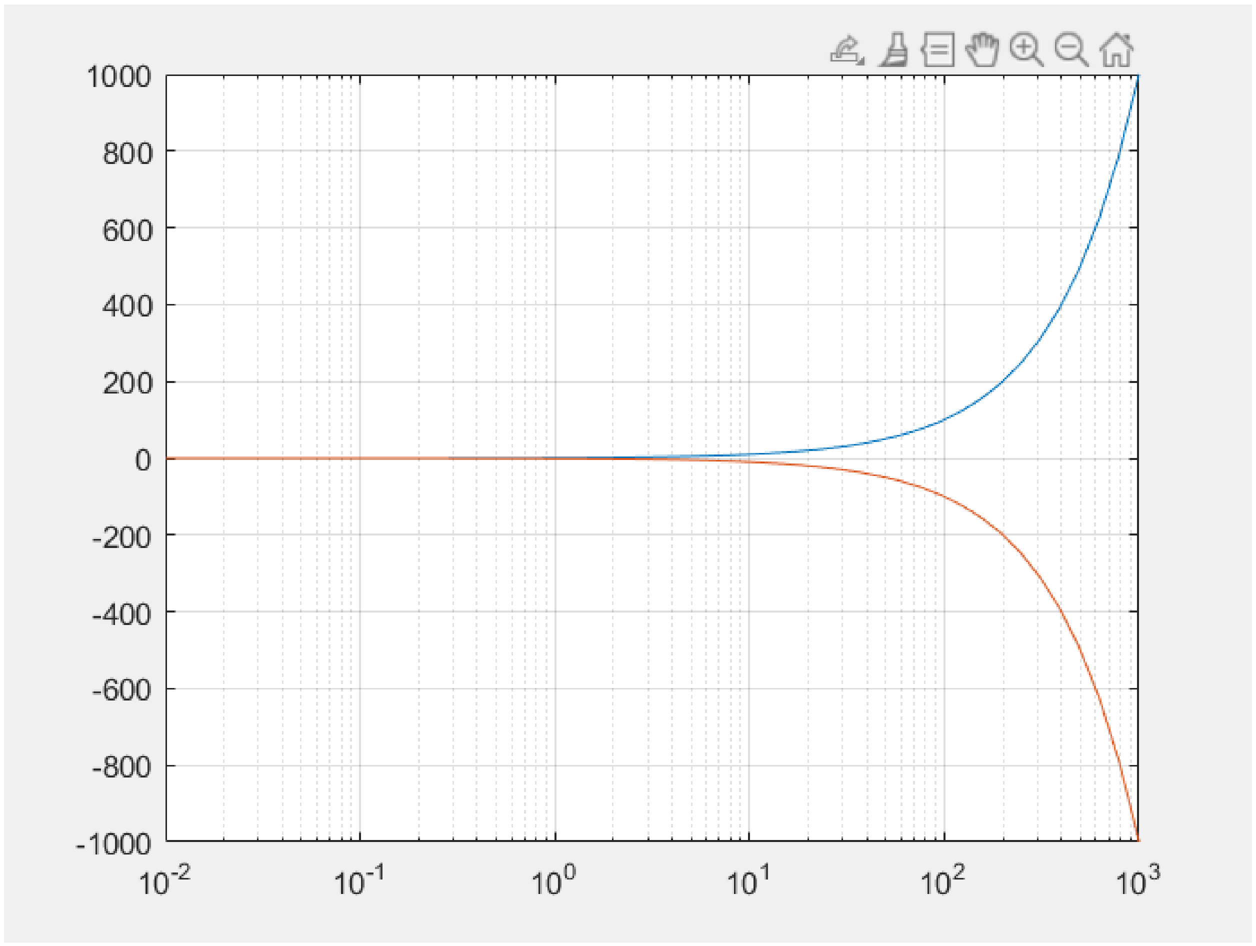
Semilogx1:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y = x;
% assign x to y
semilogx(x,y)
% create a linear-log plot of x and y
grid on
% show the grid lines
```



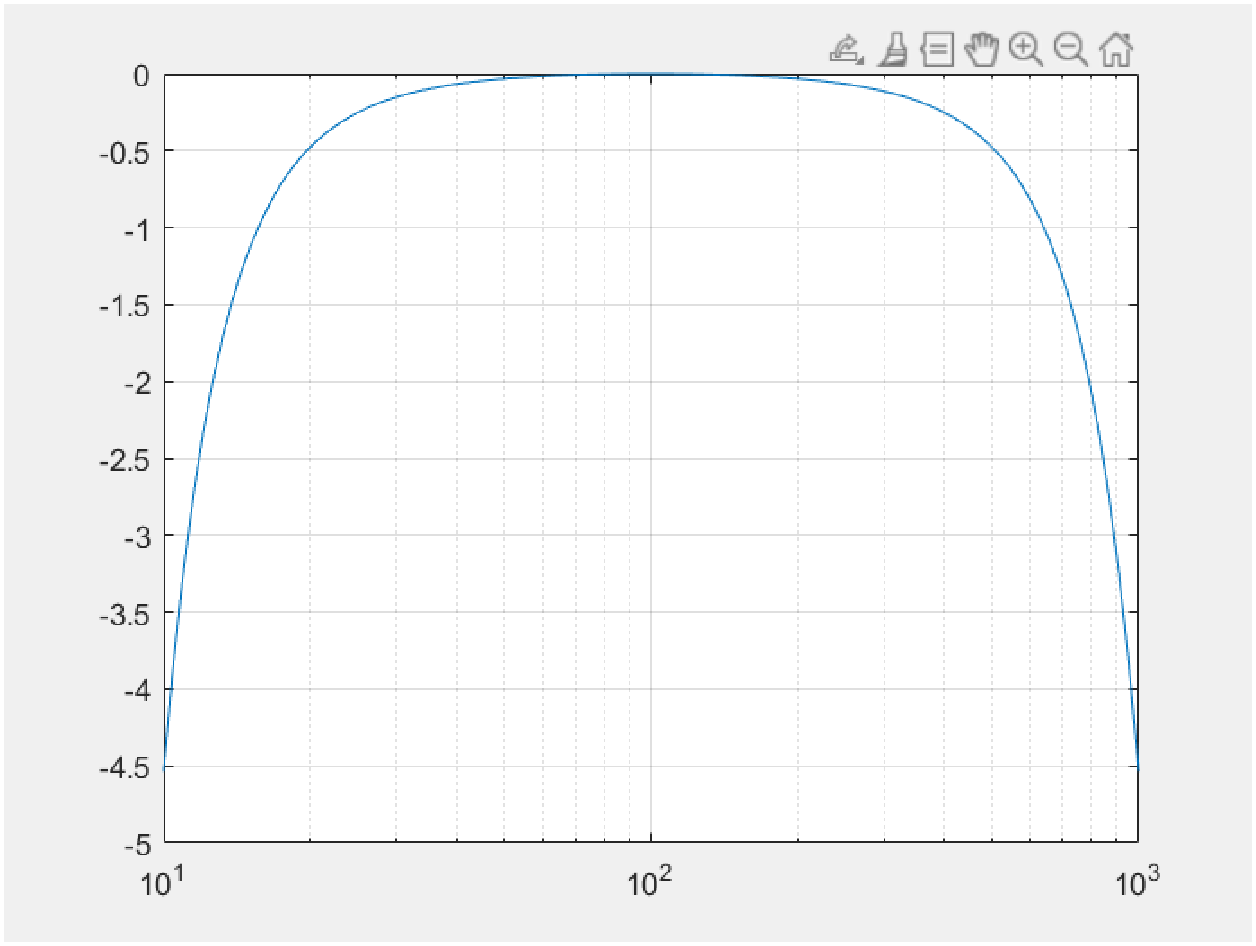
Semilogx2:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y1 = x;
% assign x to y1
y2 = -x;
% assign -x to y2
semilogx(x,y1,x,y2)
% plot two lines by passing comma-separated x-y pairs to
semilogx
grid on
% show the grid lines
```



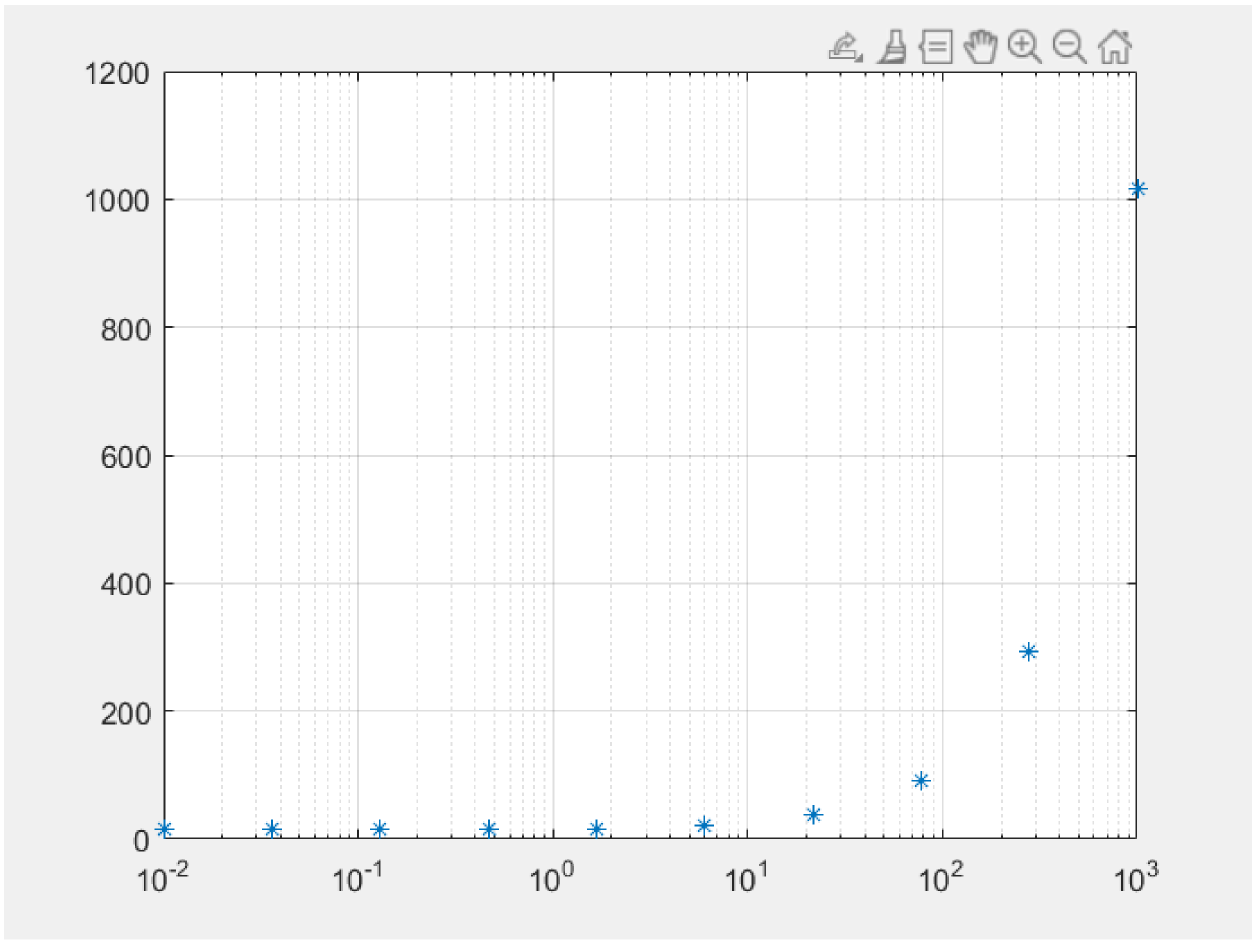
Semilogx3:

```
f = logspace(1,3,100);
% generate a vector containing the frequencies from 10 Hz
to 1000 Hz
v = linspace(-50,50,100);
gain = (1-exp(5*(2.5*v.^2)./7500))/14;
% define a vector of power gain values
semilogx(f,gain)
% plot the gain values against frequency
grid on
% show the grid lines
```



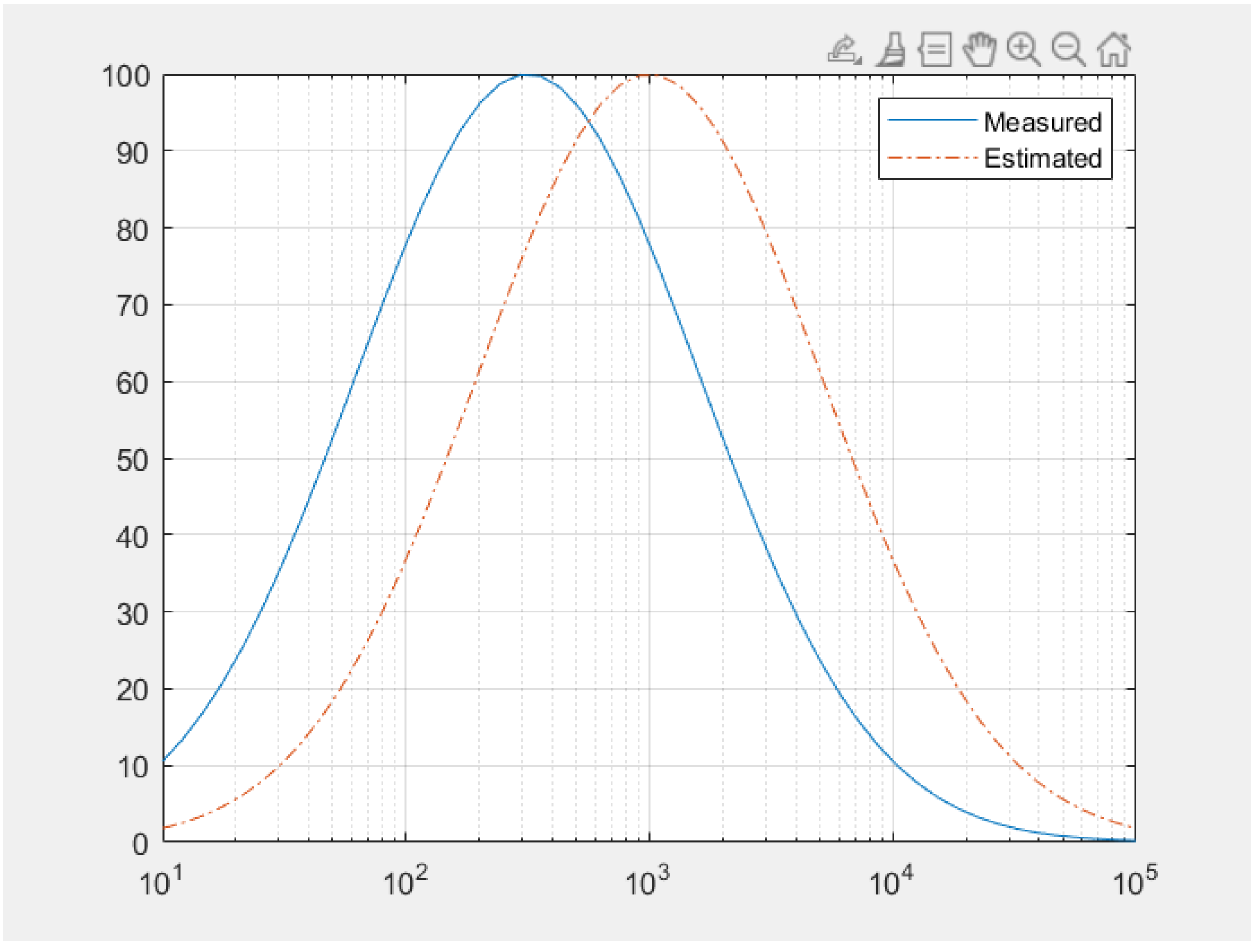
Semilogx4:

```
x = logspace(-2,3,10);
% create a set of x- coordinates
y = 15 + x;
% create a set of y- coordinates
semilogx(x,y,'*','MarkerFaceColor',[0 0.447 0.741])
% specify the line style as '*'. Specify the marker fill
color as the RGB triplet [0 0.447 0.741]
grid on
% show the grid lines
```



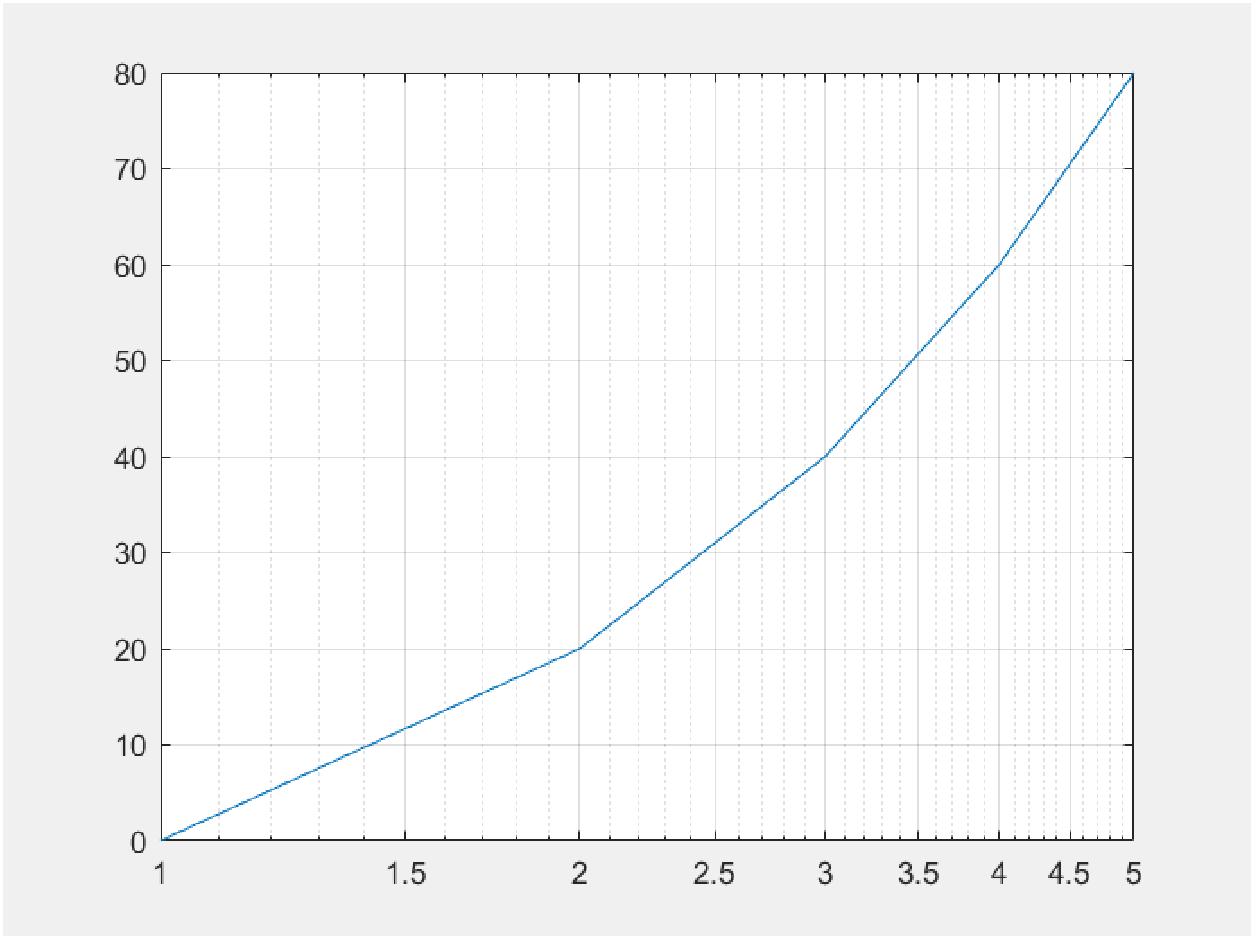
Semilogx5:

```
x = logspace(1,5,50);
% create a vector of logarithmically spaced x-coordinates
v = linspace(-20,20,50);
y1 = 100*exp(-1*((v+5).^2)./100);
% create a vector of logarithmically spaced y-coordinates
y2 = 100*exp(-1*(v.^2)./100);
% create a vector of logarithmically spaced y-coordinates
semilogx(x,y1,x,y2,'-.')
% plot two lines by passing comma-separated x-y pairs
legend('Measured','Estimated')
% display a legend
grid on
% show the grid lines
```



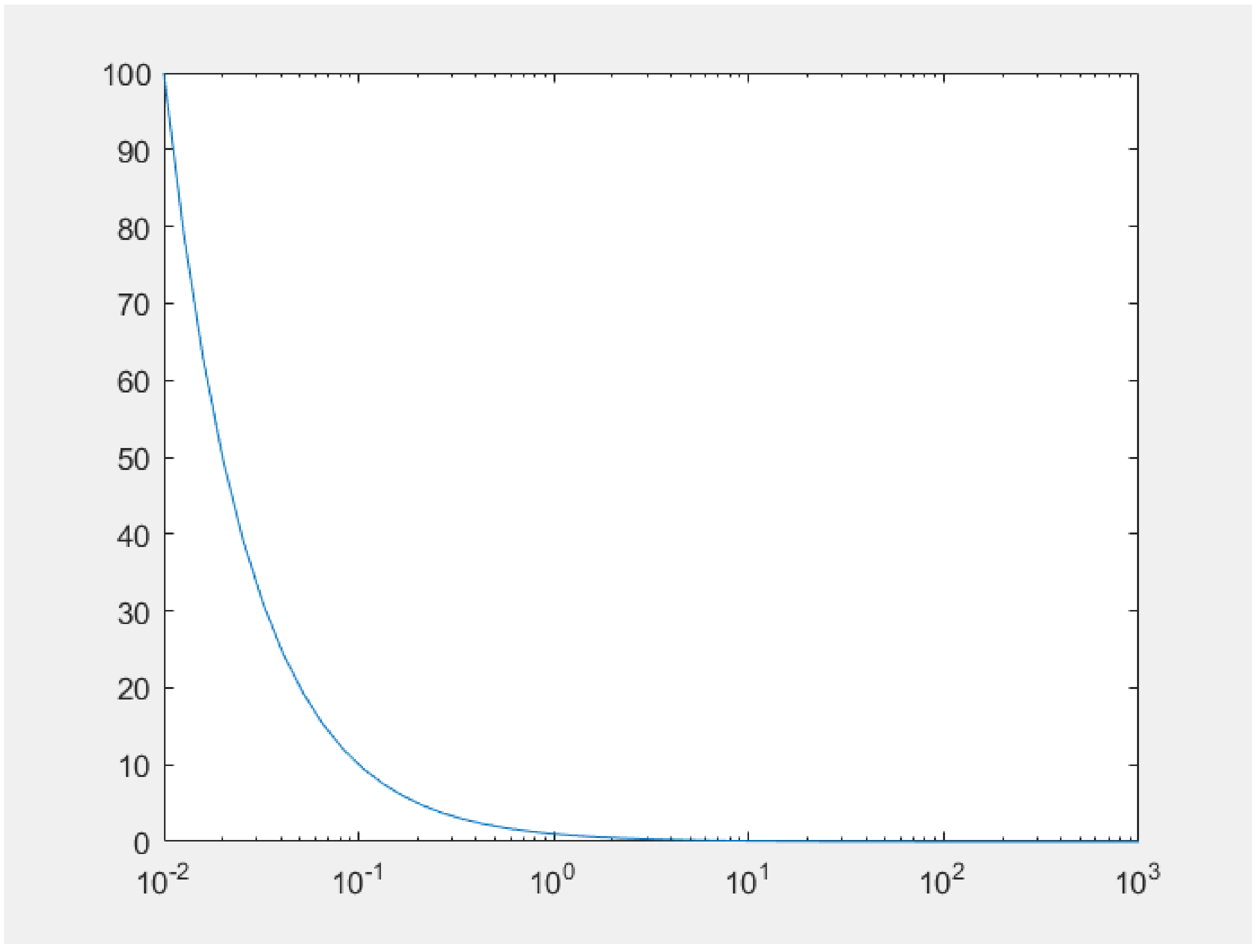
Semilogx6:

```
y = [0 20 40 60 80];
% specify only one coordinate vector
semilogx(y)
% plot the coordinates against the values 1:length(y)
grid on
% show the grid lines
```



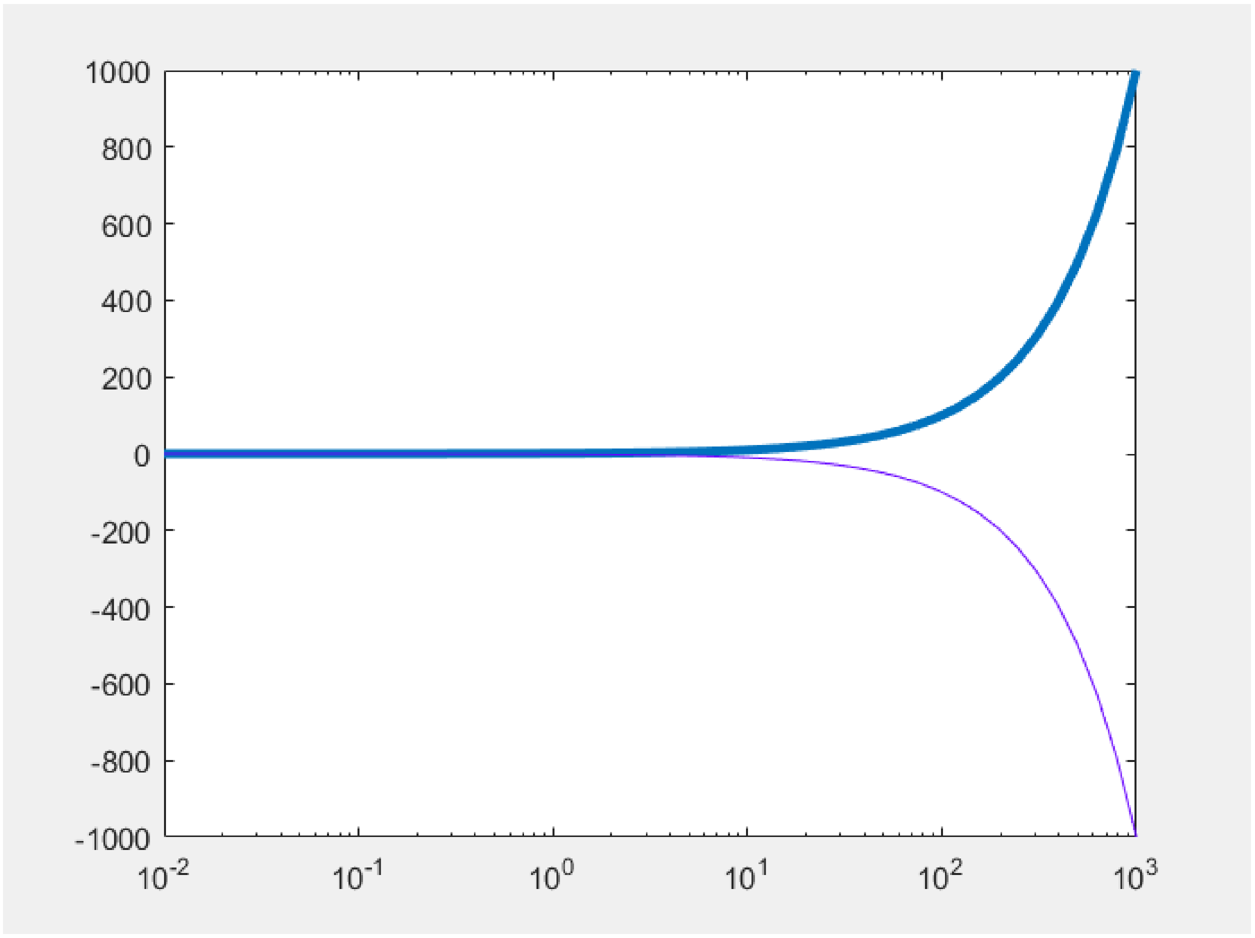
Semilogx7:

```
tiledlayout('flow')
% create a tiled chart layout in the 'flow' tile
arrangement
ax1 = nexttile;
% create an axes object
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y1 = 1./x;
% elementwise divide 1 by x
semilogx(ax1,x,y1)
% display a linear-log plot
```



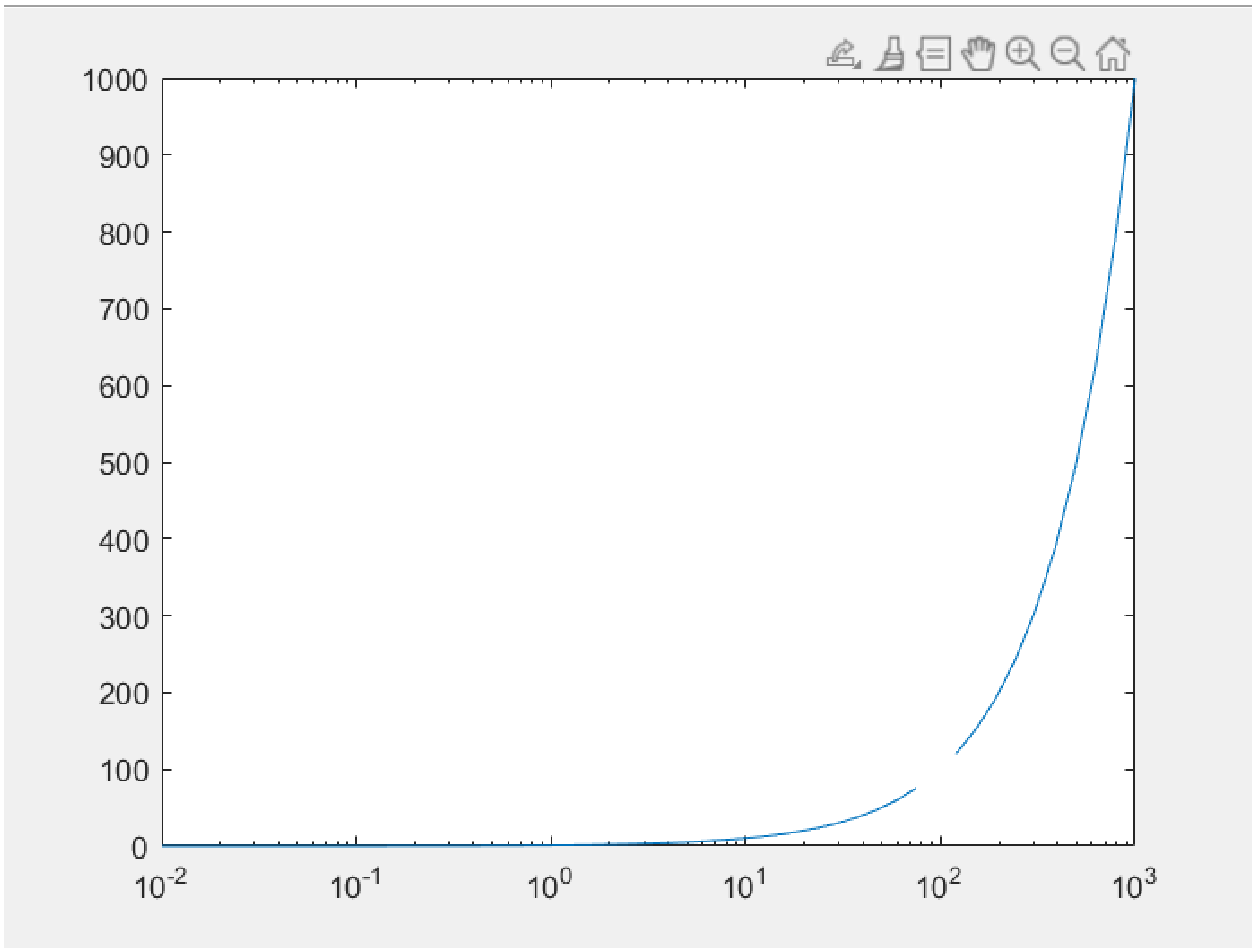
Semilogx8:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y1 = x;
% assign x to y1
y2 = -x;
% assign -x to y2
slg = semilogx(x,y1,x,y2);
% plot two lines by passing comma-separated x-y pairs to
semilogx
slg(1).LineWidth = 3;
% Change the width of the first line to 4
slg(2).Color = [0.4 0 1];
% and change the color of the second line to purple
```



Semilogx9:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y = x;
% assign x to y
y(40) = NaN;
% replace the fortieth y-coordinate with a NaN value
semilogx(x,y)
% create a linear-log plot of x and y
```



Syntax	Meaning	Description
1) semilogy(X,Y) 2) semilogy(X,Y,LineSpec) 3) semilogy(X1,Y1,…,Xn,Yn) 4) semilogy(X1,Y1,LineSpec1,…,Xn,Yn,LineSpecn) 5) semilogy(Y) 6) semilogy(Y,LineSpec) 7) semilogy(__ ,Name,Value) 8) semilogy(ax, __) 9) lineobj = semilogy(__)	Semilog plot (y-axis has log scale)	1) This syntax plots x- and y-coordinates using a base-10 logarithmic scale on the y-axis and a linear scale on the x-axis. 2) This syntax creates the plot using the specified color, marker, and line style. 3) This syntax plots multiple pairs of x- and y-coordinates on the same set of axes. 4) This syntax assigns specific colors, markers, and line styles to each x-y pair.

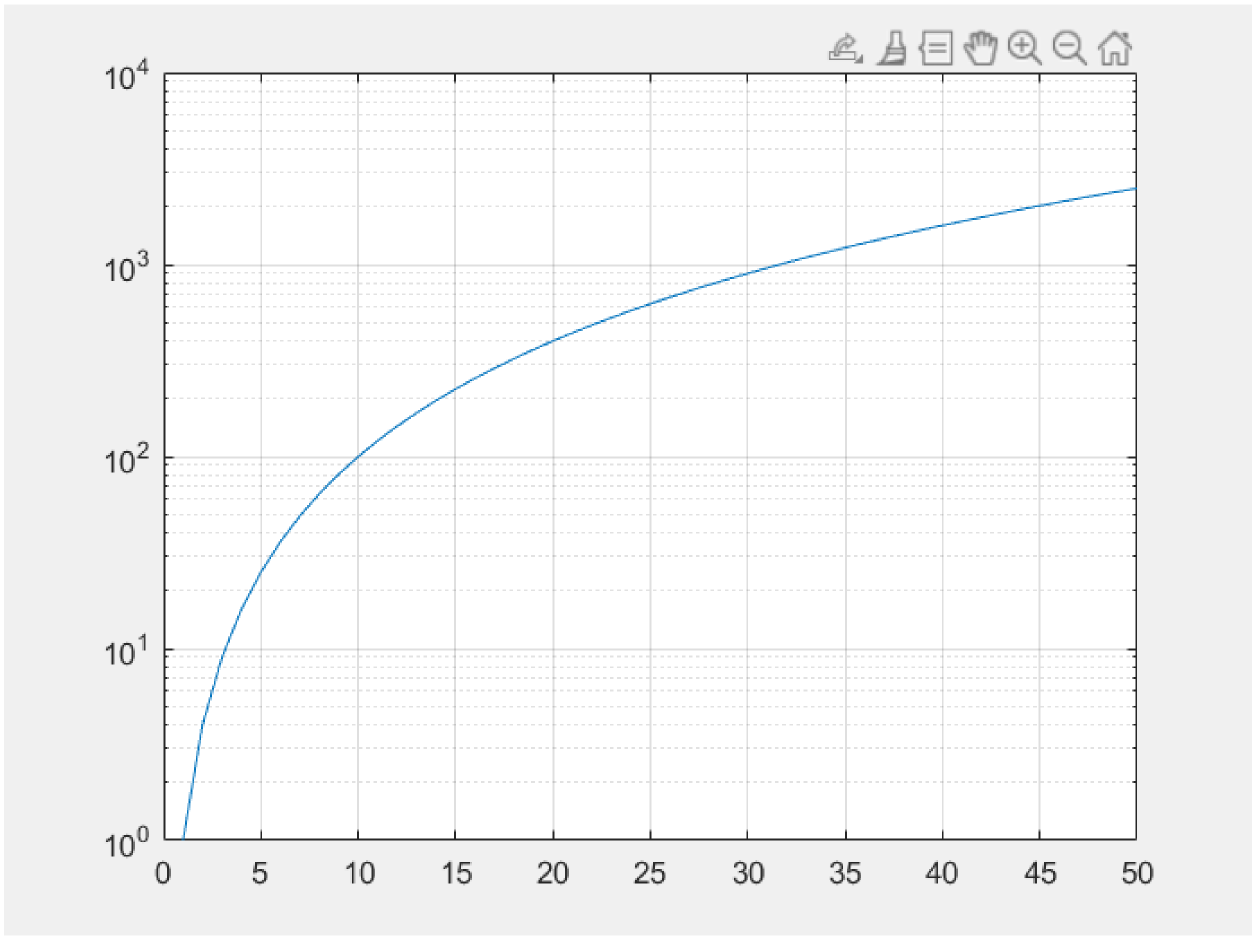
		5) This syntax plots Y against an implicit set of x-coordinates. Note: If Y is a vector, the x-coordinates range from 1 to length(Y). Note: If Y is a matrix, the plot contains one line for each column in Y. The x-coordinates range from 1 to the number of rows in Y. 6) This syntax specifies color, marker, and line style. 7) This syntax specifies Line properties using one or more Name,Value pair arguments.

		8) This syntax displays the plot in the target axes. 9) This syntax returns a Line object or an array of Line objects.

Examples:

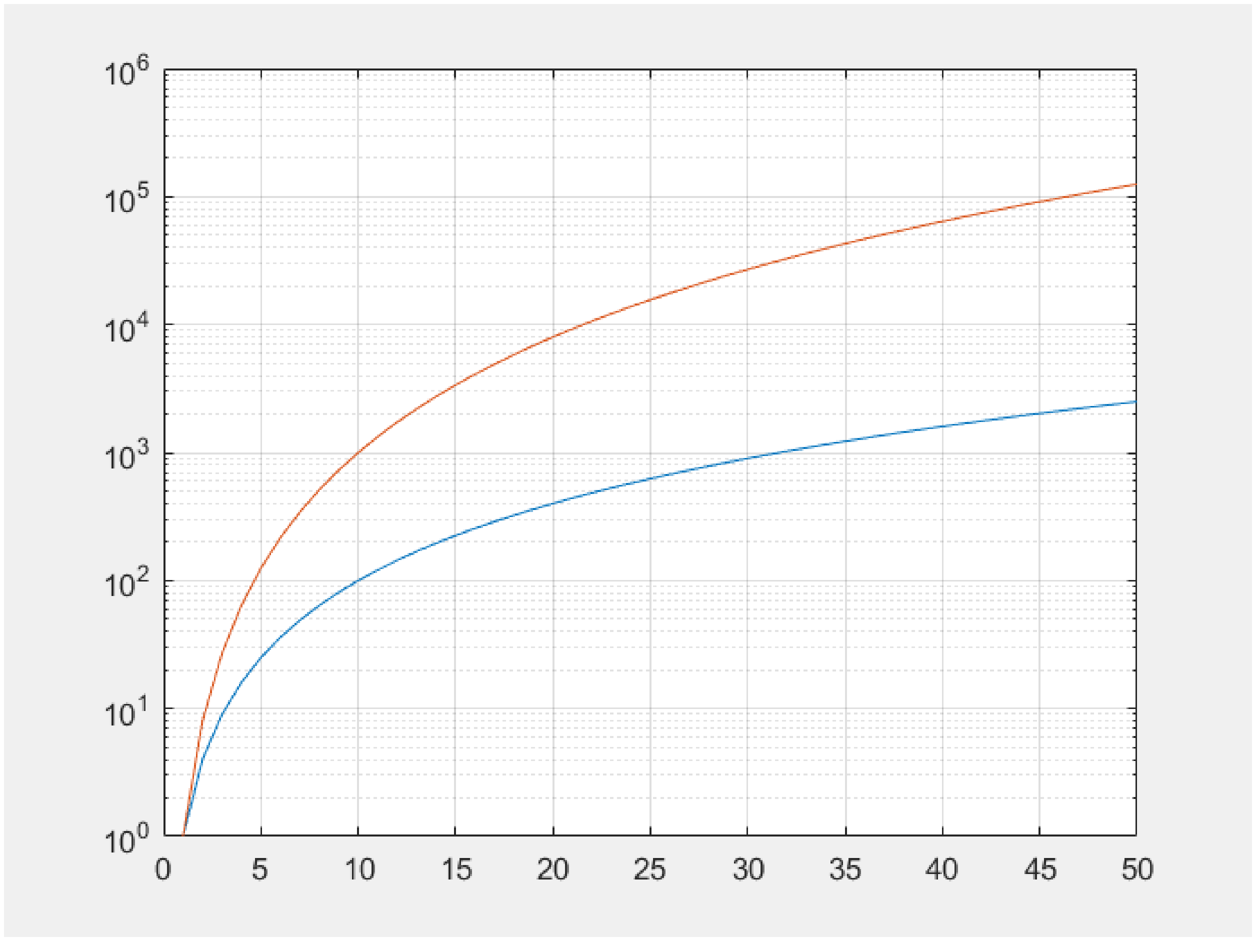
Semilogy1:

```
x = 1:50;
% create a vector of x- coordinates
y = x.^2;
% create a vector of y- coordinates
semilogy(x,y)
% create a log-linear plot of x and y
grid on
% % show the grid lines
```



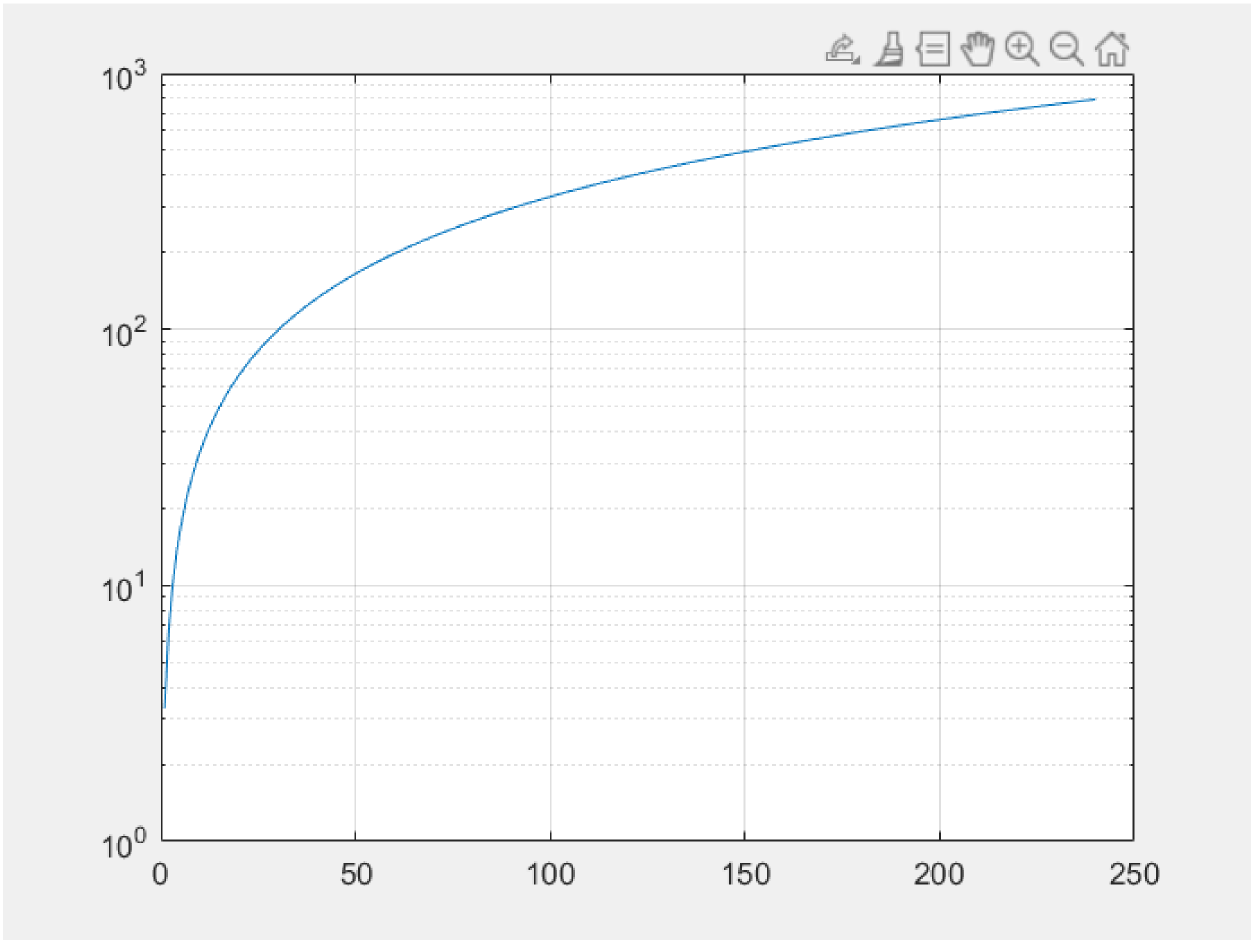
Semilogy2:

```
x = 1:50;
% create a vector of x- coordinates
y1 = x.^2;
% create the fist vector of y- coordinates
y2 = x.^3;
% create the second vector of y- coordinates
semilogy(x,y1,x,y2)
% plot two lines by passing comma-separated x-y pairs to
semilogy
grid on
% show the grid lines
```



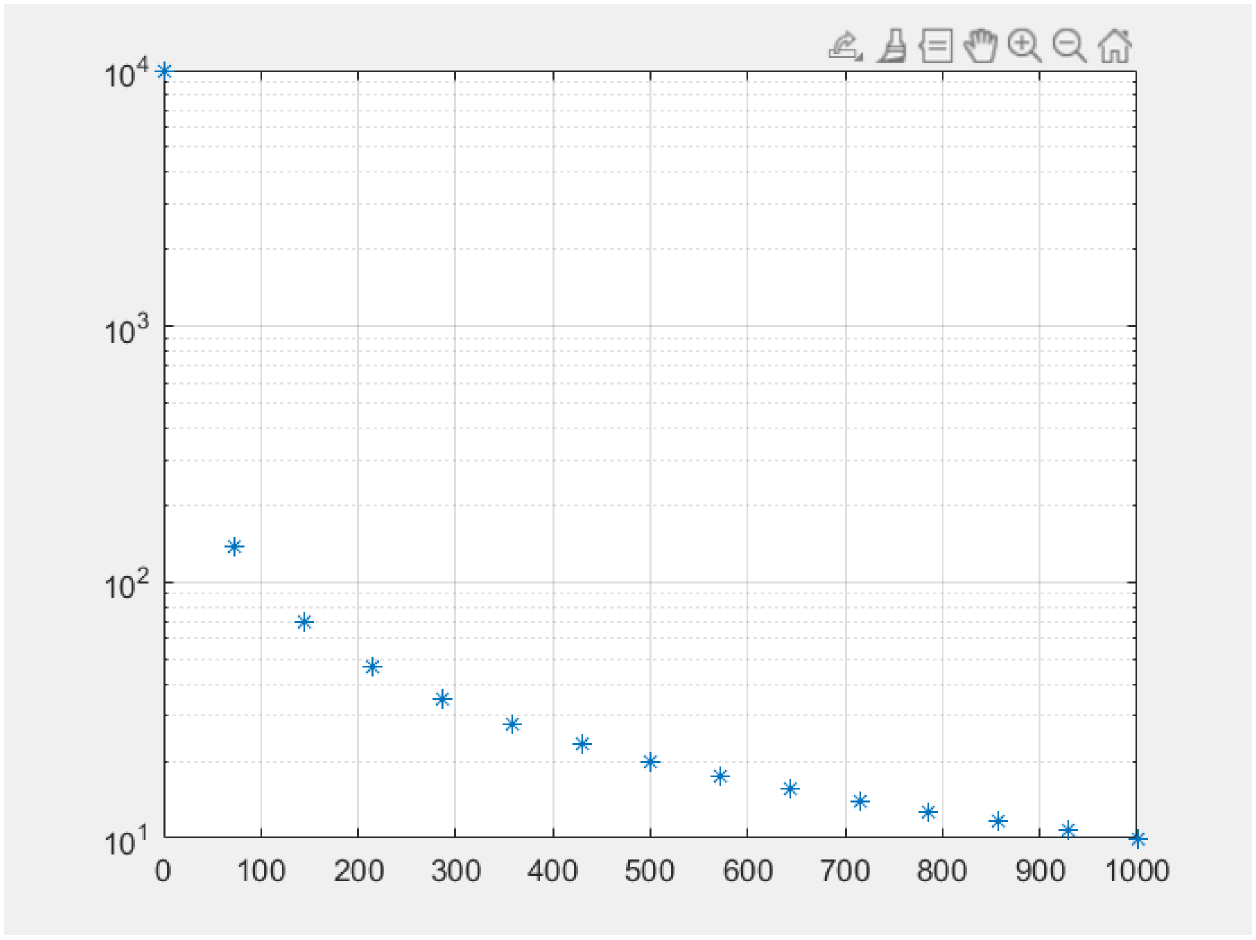
Semilogy3:

```
P = 500;
npayments = 240;
rate = 0.05/12;
mpayment =
P*(rate*(1+rate)^npayments)/(((1+rate)^npayments) - 1);
x = 1:240;
% create installments on a 20 year loan
y = x * mpayment;
% create cumulative cost of a $1000 loan with an interest
rate of 5%
semilogy(x,y);
% plot the cumulative cost at each installment
grid on
% show the grid lines
```



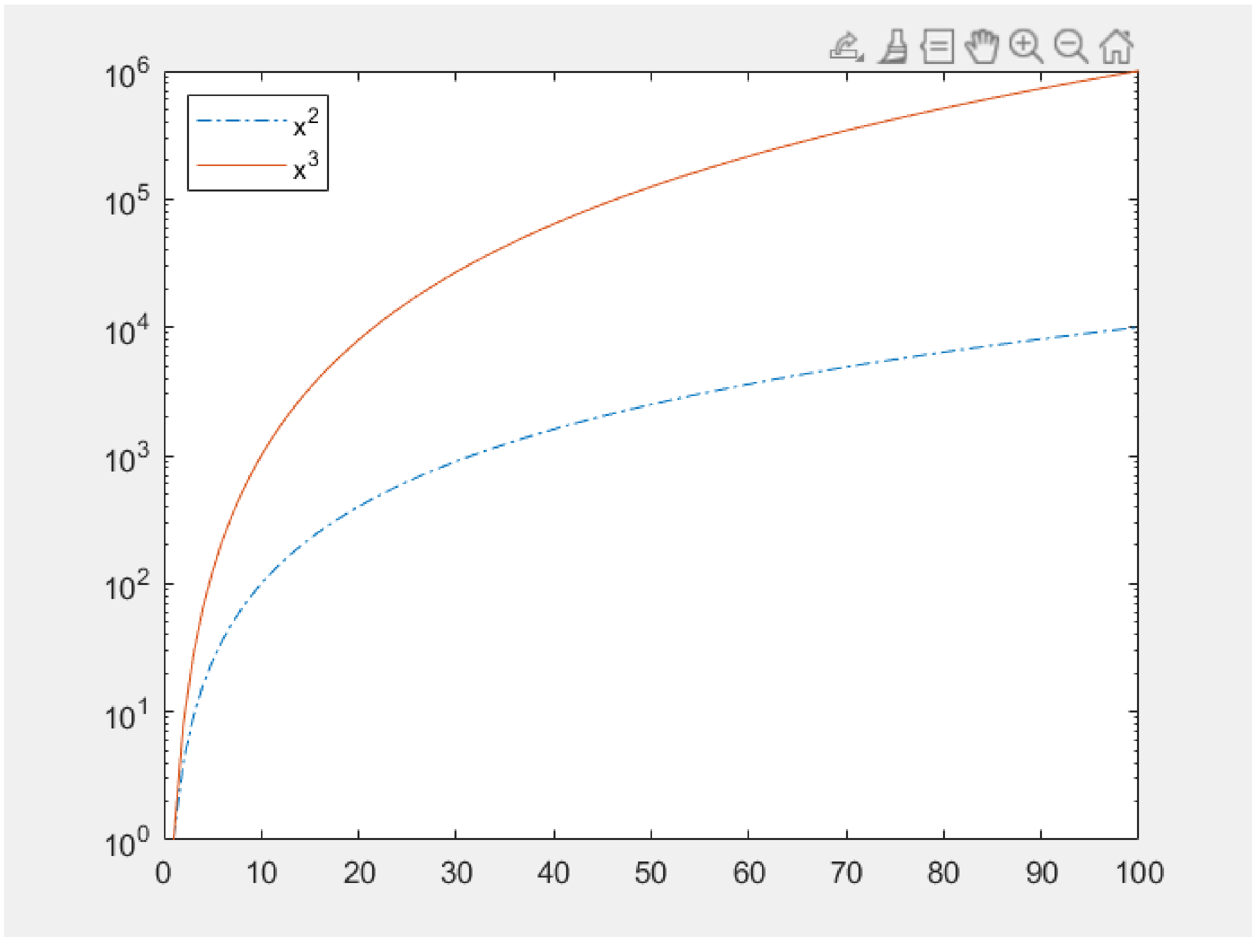
Semilogy4:

```
x = linspace(1,1000,15);
% create a set of x- coordinates
y = (1./x) * 10000;
% create a set of y- coordinates
semilogy(x,y,'*','MarkerFaceColor',[0 0.447 0.741])
% plot x and y in a log-linear plot
% specify the line style as '*'
% Specify the marker fill color as the RGB triplet [0 0.447
0.741]
grid on
% show the grid lines
```



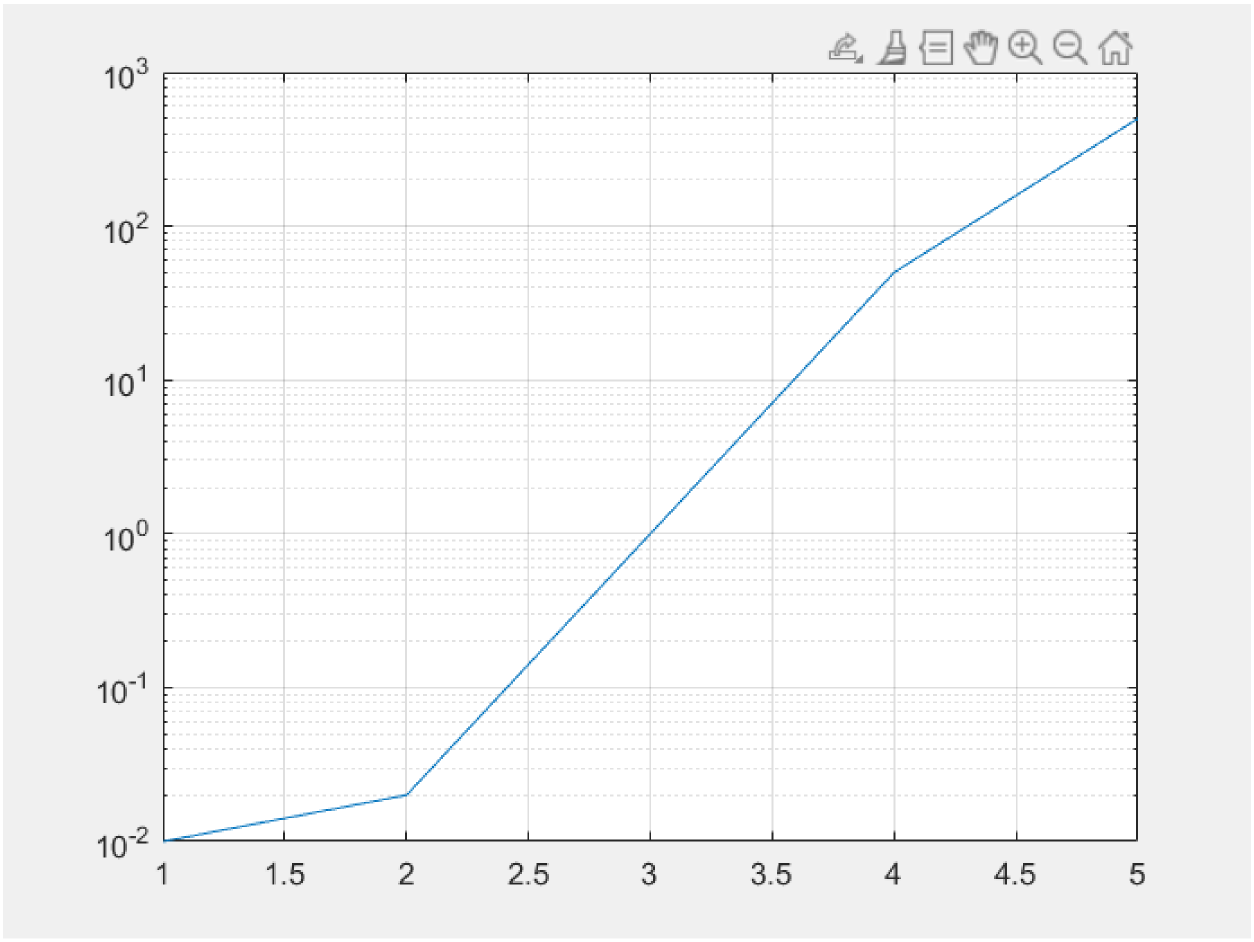
Semilogy5:

```
x = 1:100;
% create a vector of x- coordinates
y1 = x.^2;
% create the fist vector of y- coordinates
y2 = x.^3;
% create the second vector of y- coordinates
semilogy(x,y1,'-.',x,y2)
% display x,y1,y2 in a log-linear plot
legend('x^2','x^3','Location','northwest')
% display a legend in the upper left corner of the plot
```



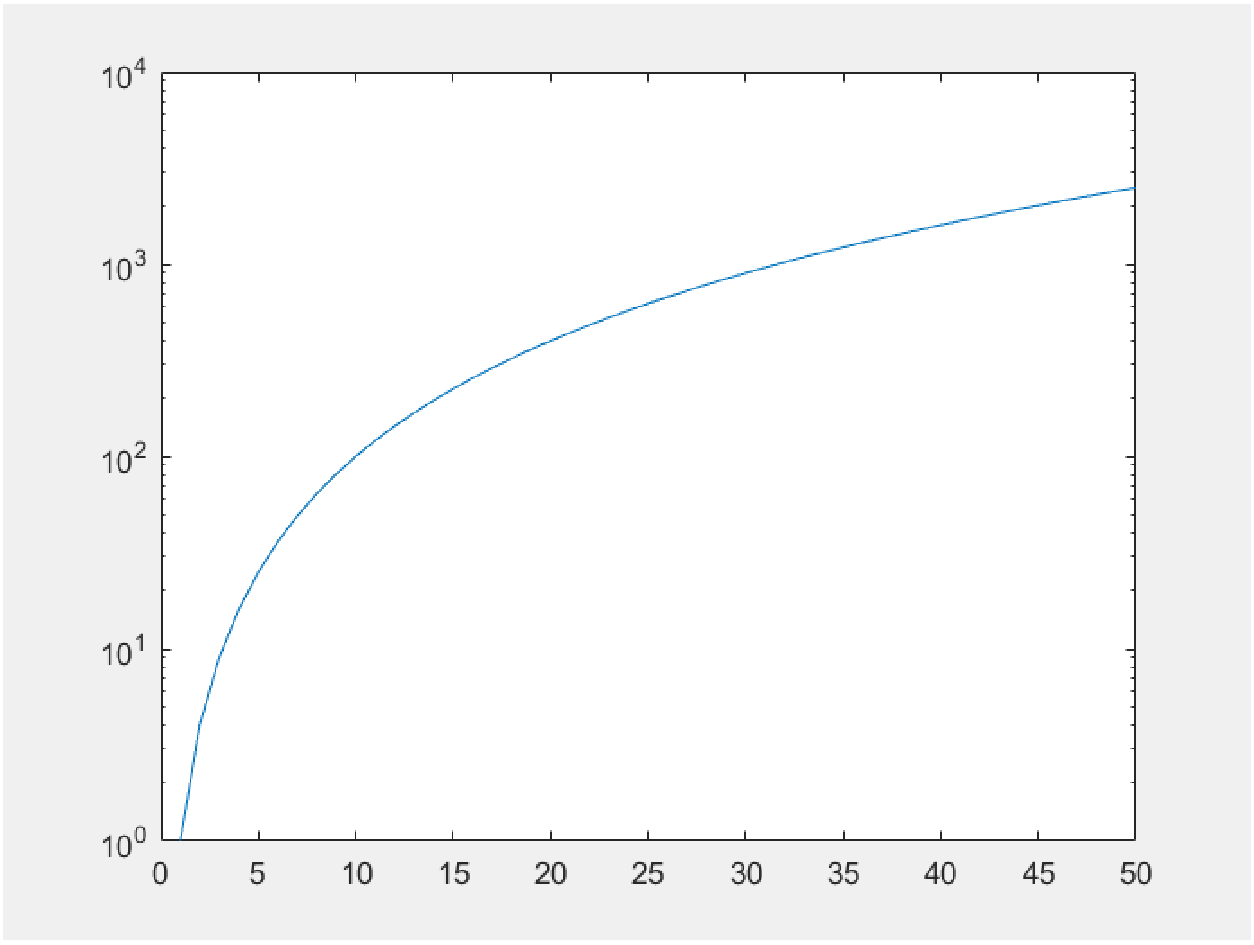
Semilogy6:

```
y = [0.01 0.02 1 50 500];
% create only one coordinate vector
semilogy(y)
% plot the coordinates against the values 1:length(y)
grid on
% show the grid lines
```



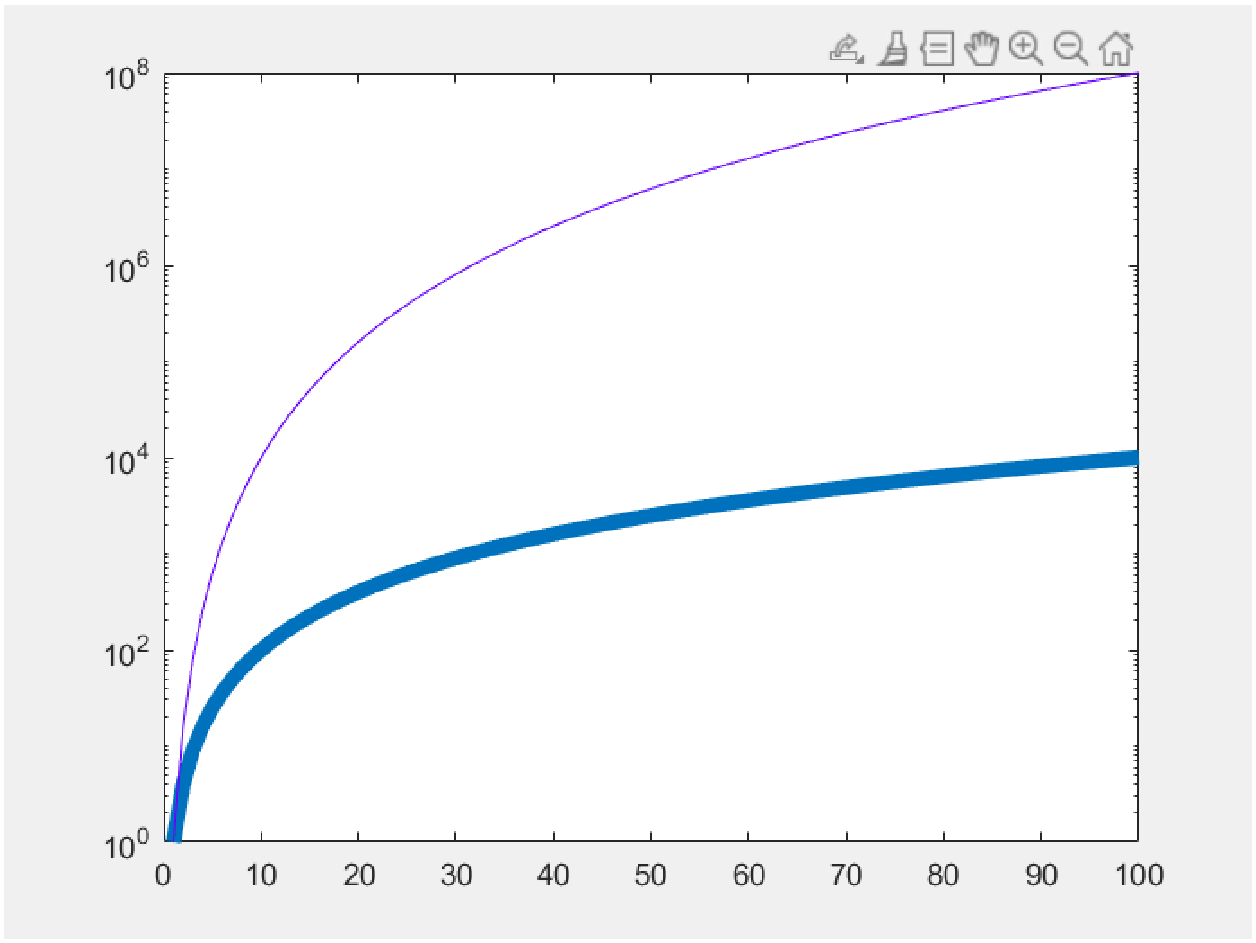
Semilogy7:

```
tiledlayout('flow')
% create a tiled chart layout in the 'flow' tile
arrangement
ax = nexttile;
% create an axes object
x = 1:50;
% create a vector of x- coordinates
y = x.^2;
% assign elementwise power of x to y
semilogy(ax,x,y)
% display a log-linear plot
```



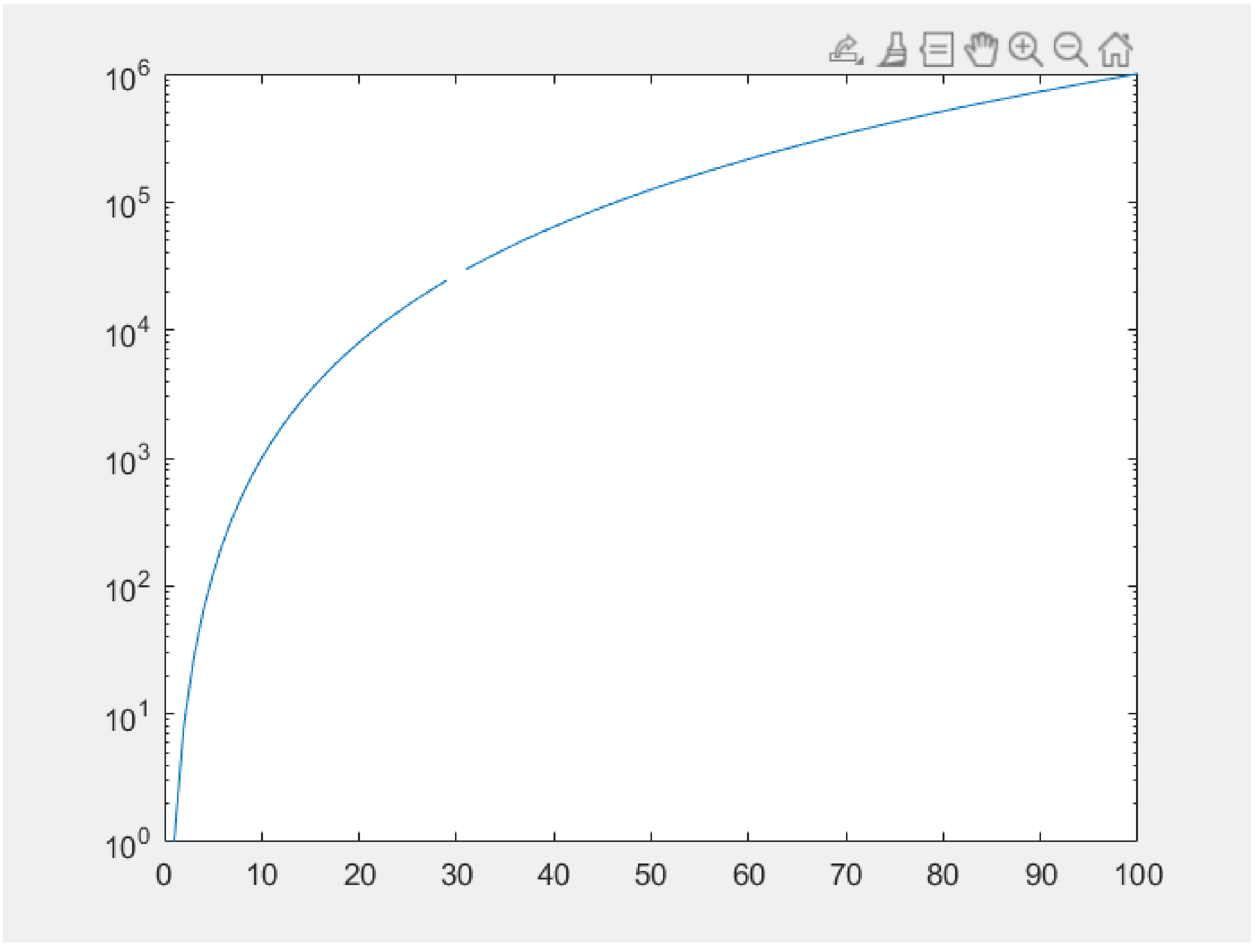
Semilogy8:

```
x = 1:100;
% create a vector of x- coordinates
y1 = x.^2;
% create the first vector of y- coordinates
y2 = x.^4;
% create the second vector of y- coordinates
slg = semilogy(x,y1,x,y2);
% create the line objects
slg(1).LineWidth = 5;
% change the width of the first line to 5
slg(2).Color = [0.4 0 1];
% change the color of the second line to purple
```



Semilogy9:

```
x = 1:100;
% create a vector of x- coordinates
y = x.^3;
% create a vector of y- coordinates
y(30) = NaN;
% replace the thirtieth y-coordinate with a NaN value
semilogy(x,y)
% create a linear-log plot of x and y
```



Q2

Syntax	Meaning	Description
1) loglog(X,Y) 2) loglog(X,Y,LineSpec) 3) loglog(X1,Y1,…,Xn,Yn) 4) loglog(X1,Y1,LineSpec1,…,Xn,Yn,LineSpecn) 5) loglog(Y) 6) loglog(Y,LineSpec) 7) loglog(__ ,Name,Value) 8) loglog(ax, __) 9) lineobj = loglog(__)	Log-log scale plot	1) This syntax plots x- and y-coordinates using a base-10 logarithmic scale on the x-axis and the y-axis. 2) This syntax creates the plot using the specified color, marker, and line style. 3) This syntax plots multiple pairs of x- and y-coordinates on the same set of axes. 4) This syntax assigns specific colors, markers, and line styles to each x-y pair.

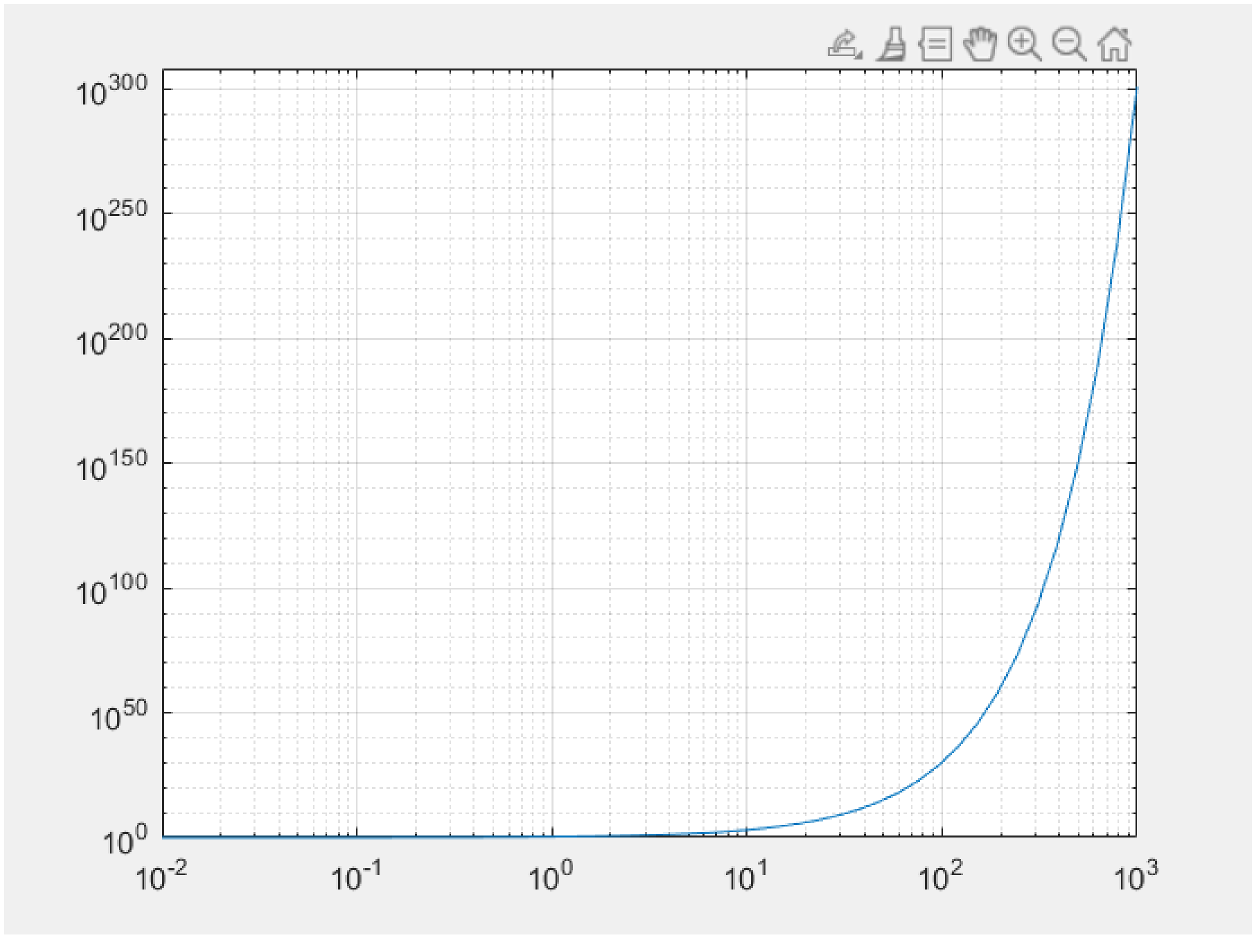
		5) This syntax plots Y against an implicit set of x-coordinates. Note: If Y is a vector, the x-coordinates range from 1 to length(Y). Note: If Y is a matrix, the plot contains one line for each column in Y. The x-coordinates range from 1 to the number of rows in Y. 6) This syntax specifies color, marker, and line style. 7) This syntax specifies Line properties using one or more Name,Value pair arguments.

		8) This syntax displays the plot in the target axes. 9) This syntax returns a Line object or an array of Line objects.

Examples:

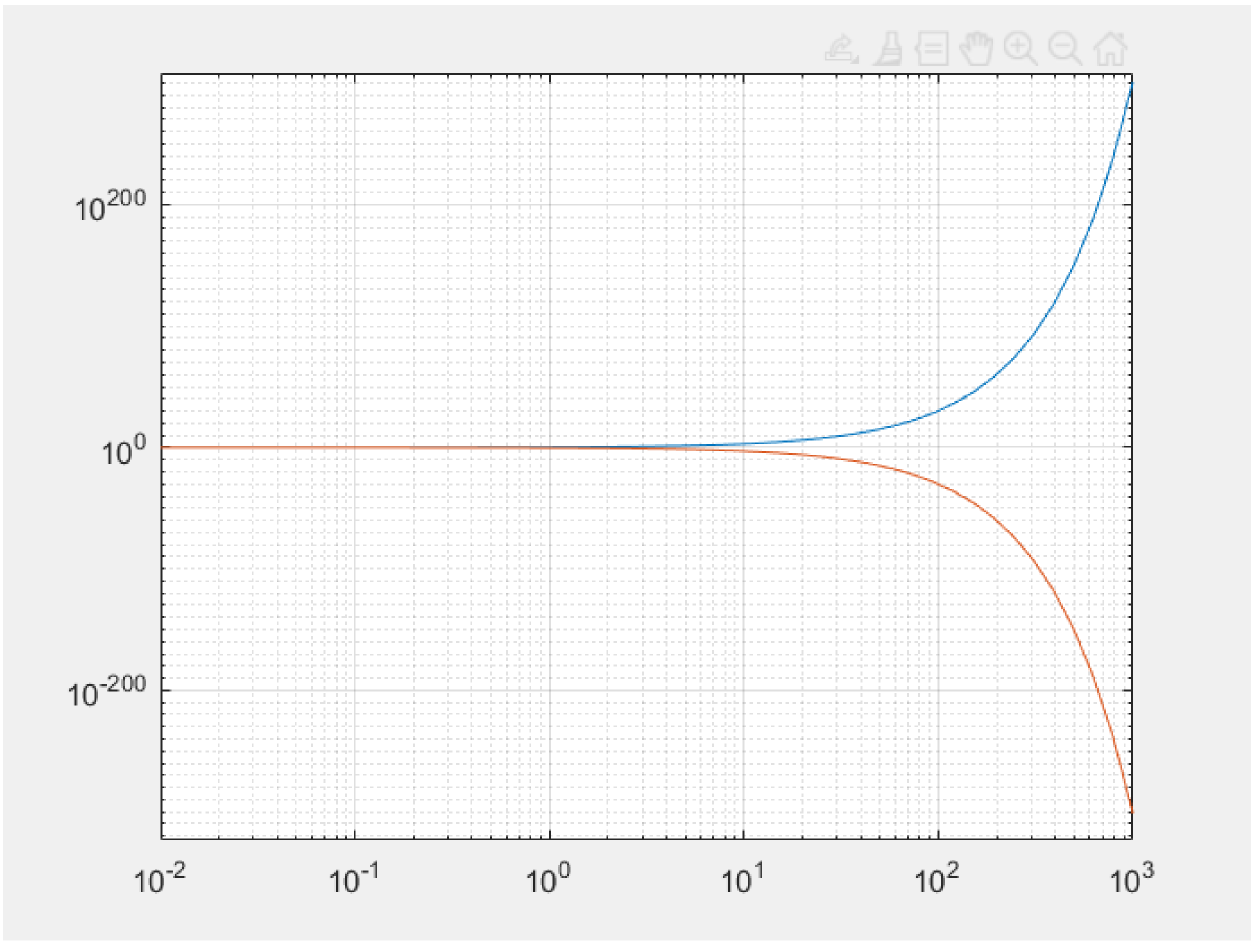
Loglog1:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y = 2.^x;
% assign 2 (elementwise) pwoer by x to y
loglog(x,y)
% plot x and y
grid on
% show the grid lines
```



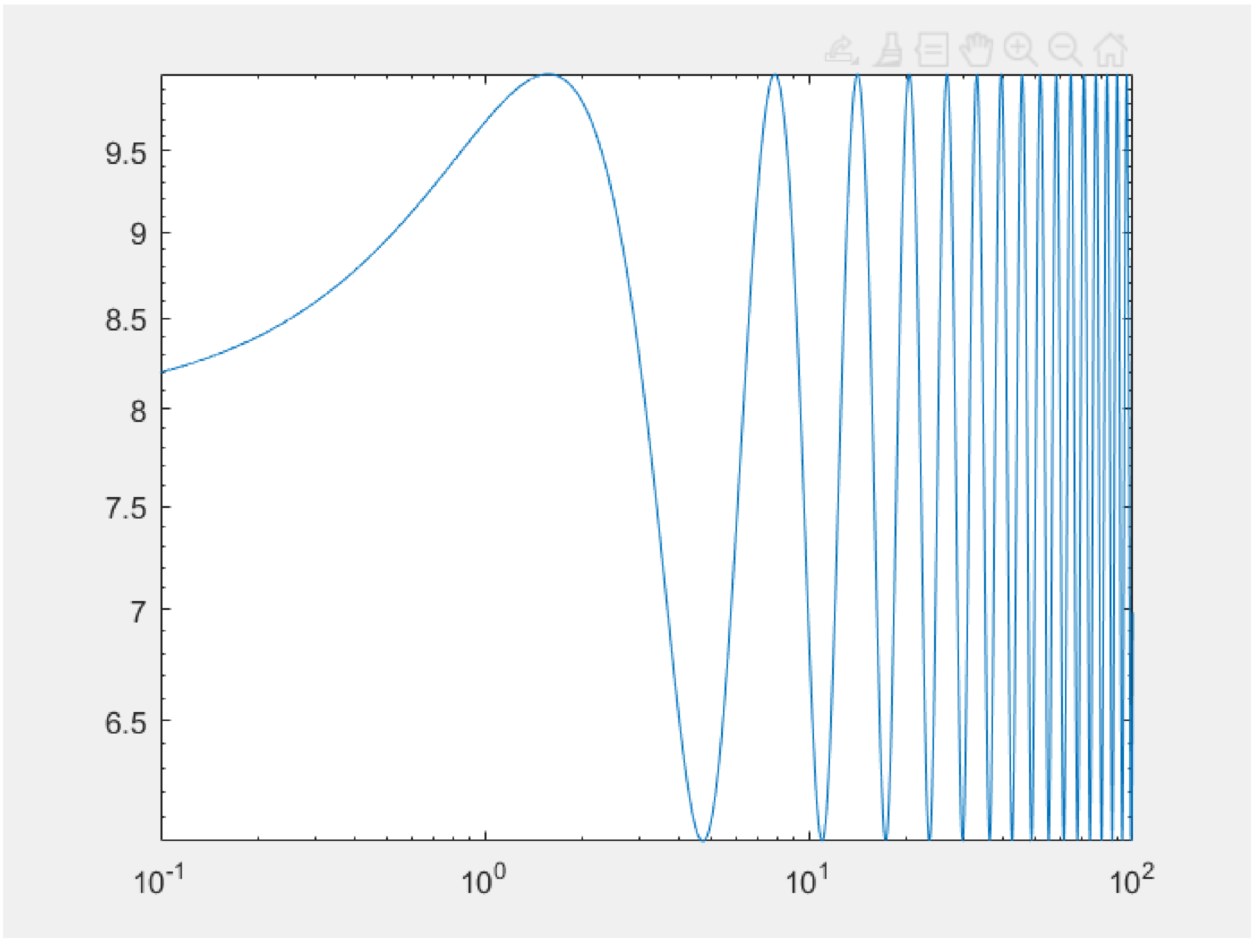
Loglog2:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y1 = 2.^x;
% create the first y- coordinates vector
y2 = 1./2.^x;
% create the second y- coordinates vector
loglog(x,y1,x,y2)
% plot two lines by passing comma-separated x-y pairs to
loglog
grid on
% show the grid lines
```



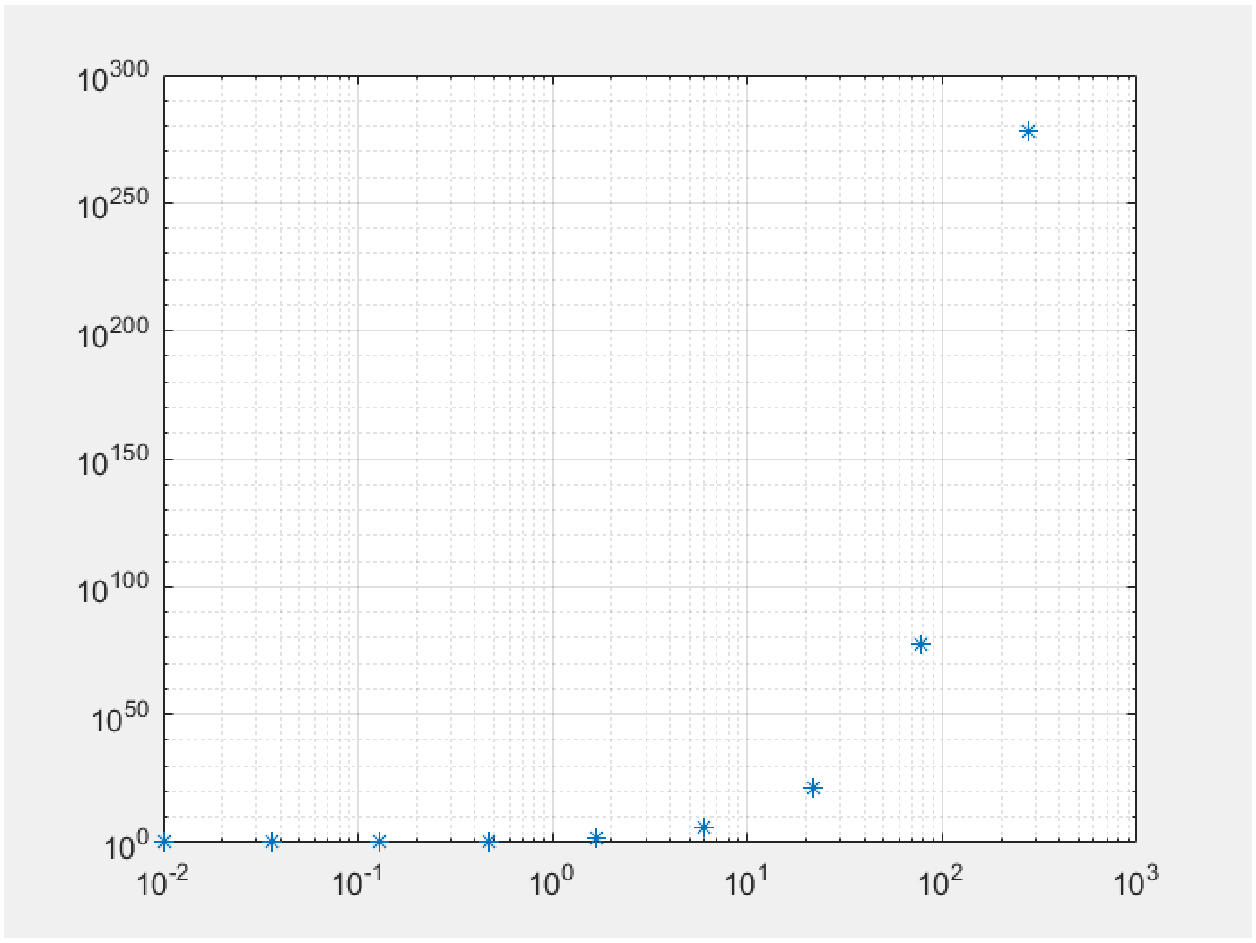
Loglog3:

```
x = logspace(-1,2,10000);
% create a set of x- coordinates
y = 8 + 2*sin(x);
% create a set of y- coordinates
loglog(x,y)
% display x and y in a loglog plot
```



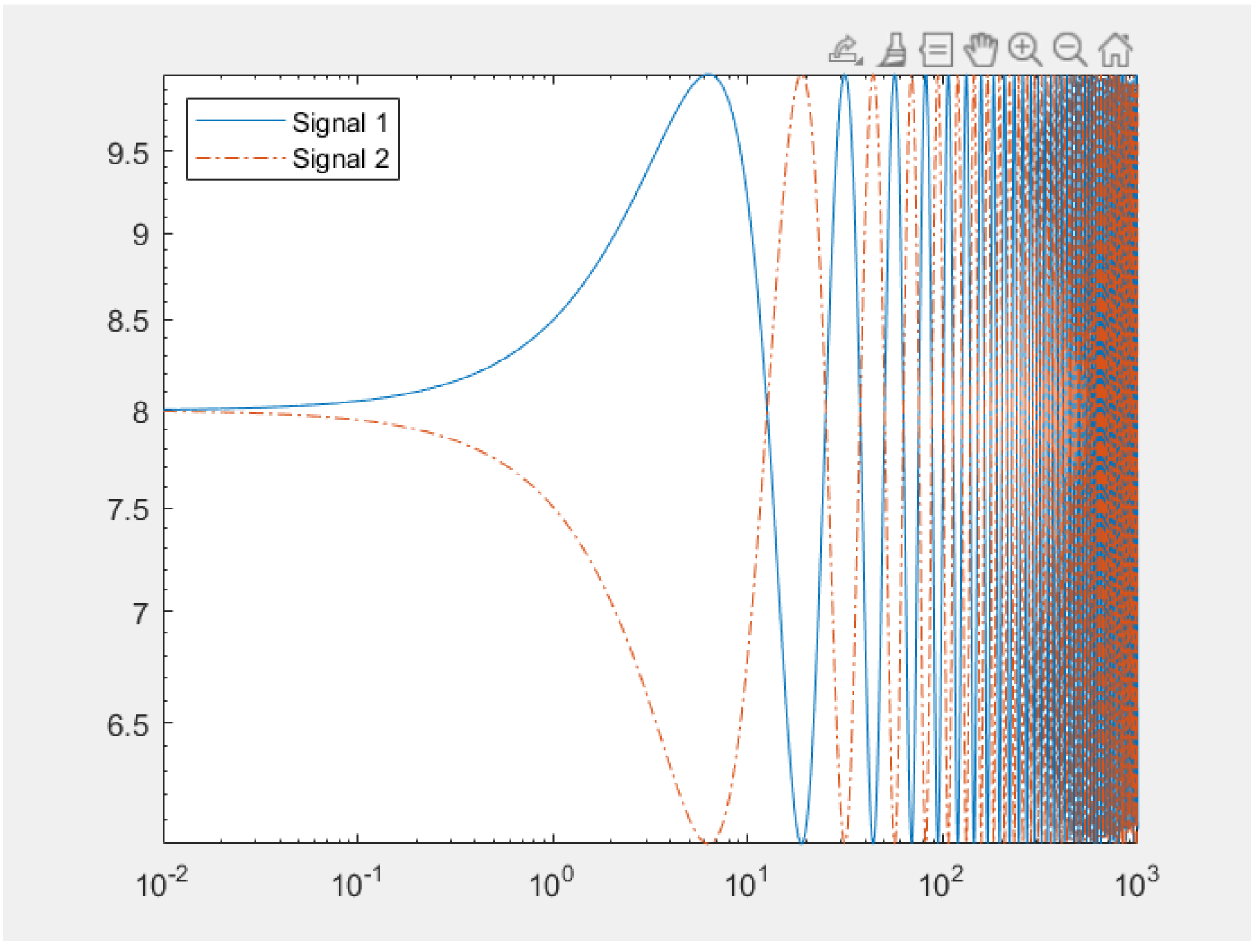
Loglog4:

```
x = logspace(-2,3,10);
% create a set of x- coordinates
y = 10.^x;
% create a set of y- coordinates
loglog(x,y,'*','MarkerFaceColor',[0 0.447 0.741])
% display x and y in a loglog plot. specify the line style
as '*'. Specify the marker fill color as the RGB triplet [0
0.447 0.741]
grid on
% show the grid lines
```



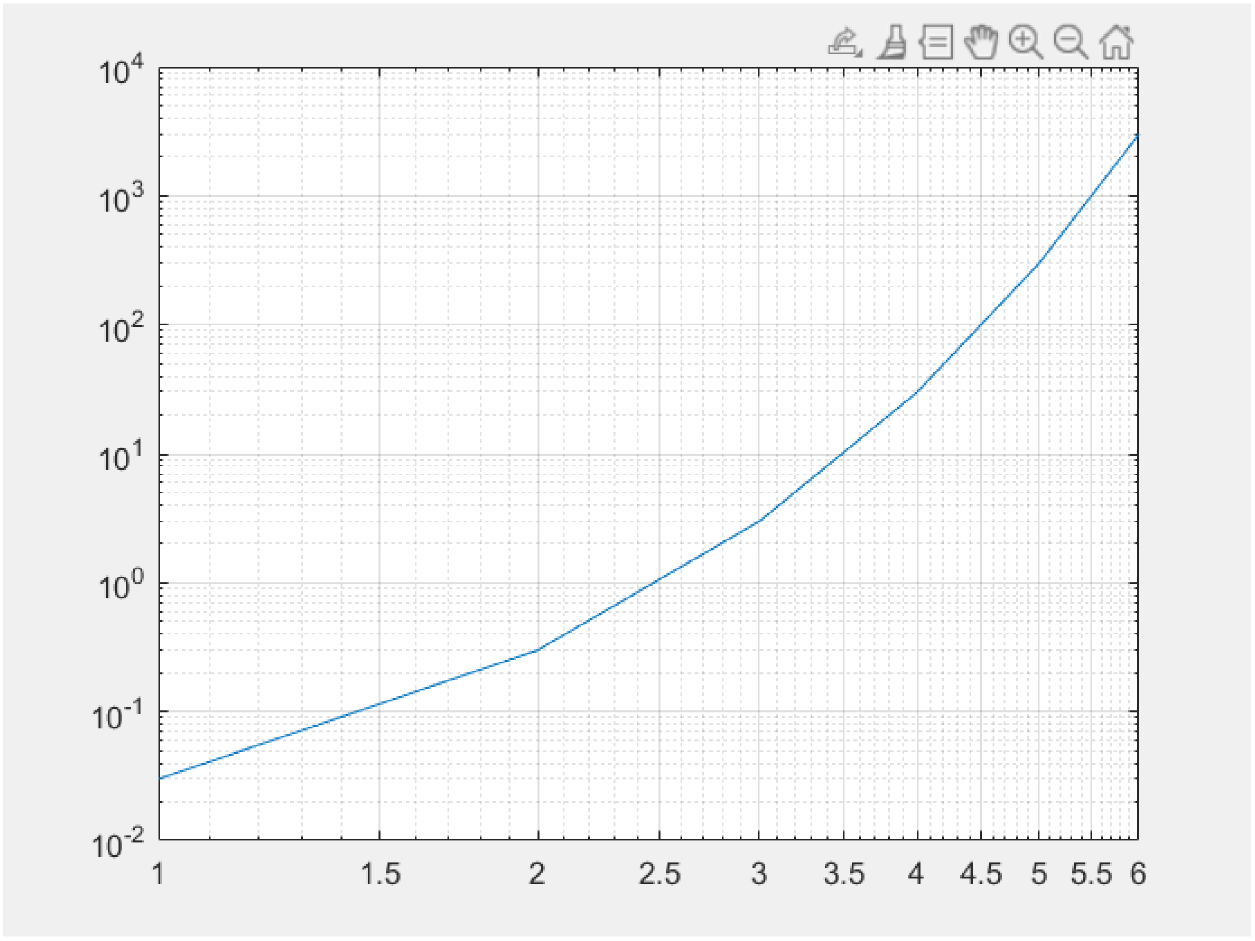
Loglog5:

```
x = logspace(-2,3,5000);
% create a set of logarithmically spaced x-coordinates
y1 = 8 + 2*sin(x/4);
% create the first set of y- coordinates
y2 = 8 - 2*sin(x/4);
% % create the second set of y- coordinates
loglog(x,y1,x,y2,'-.')
% display x,y1,y2 in a loglog plot
legend('Signal 1','Signal 2','Location','northwest')
% display a legend in the upper left corner of the plot
```



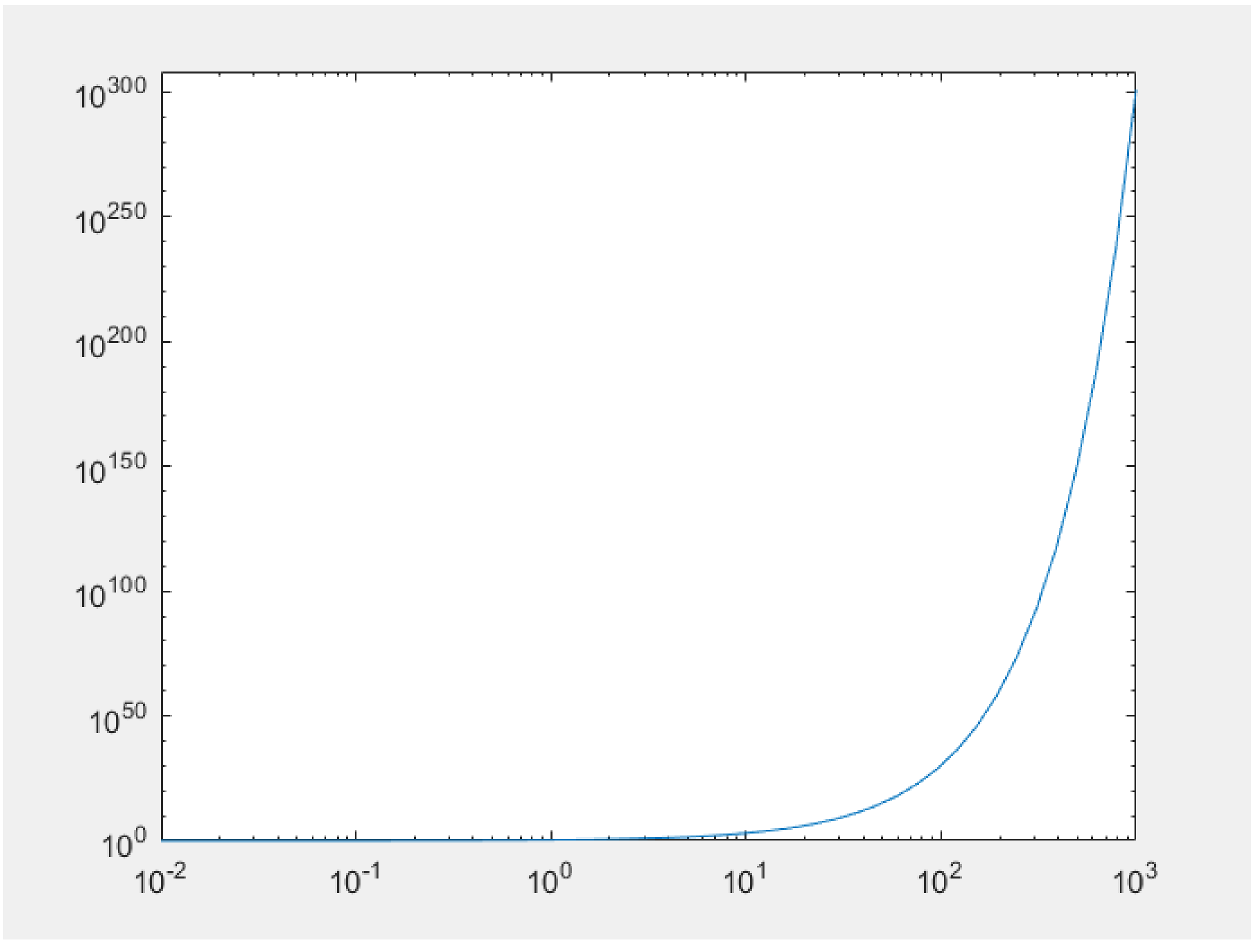
Loglog6:

```
y = [0.03 0.3 3 30 300 3000];
% specify only one coordinate vector
loglog(y)
% plot the coordinates against the values 1:length(y)
grid on
% show the grid lines
```



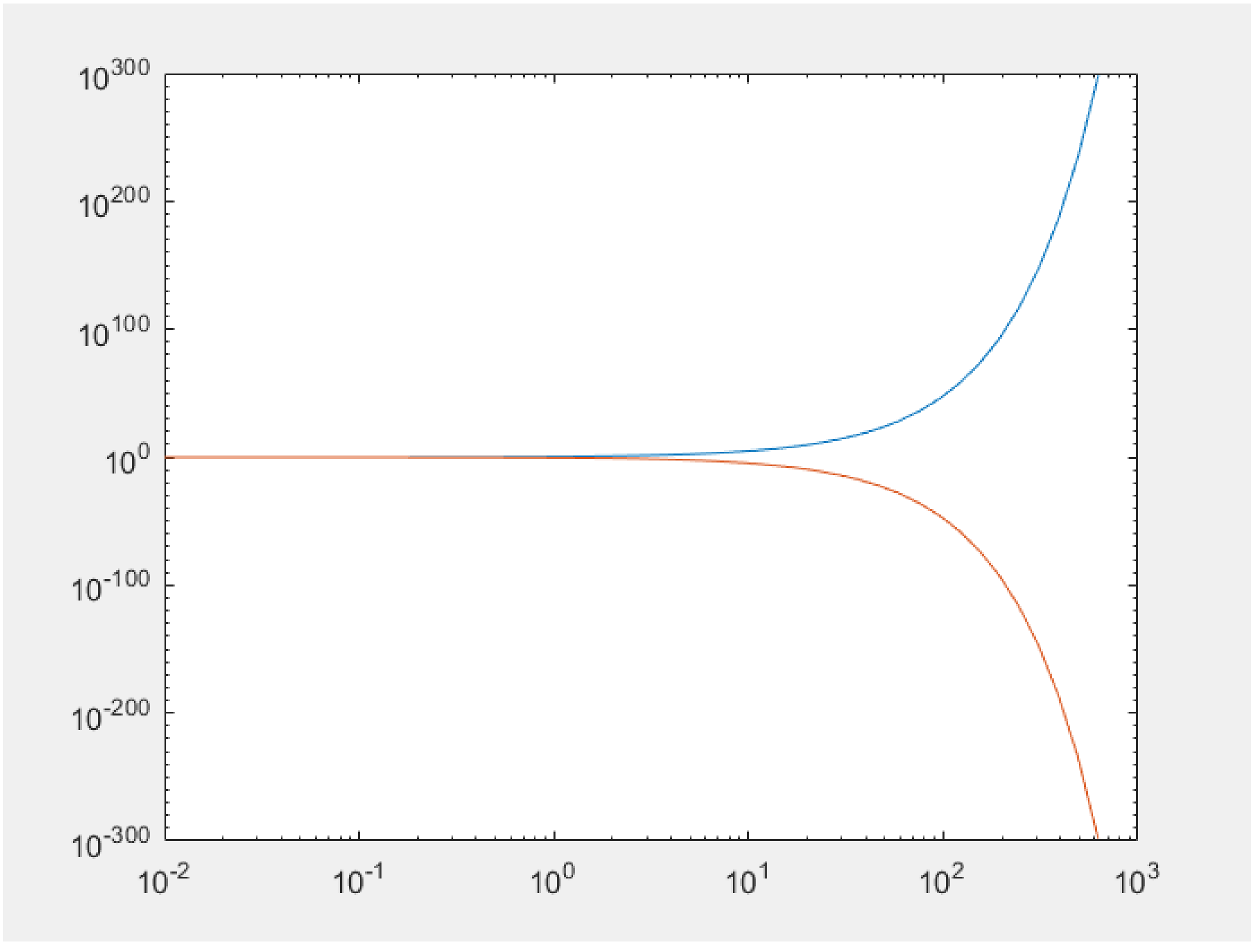
Loglog7:

```
tiledlayout('flow')
% create a tiled chart layout in the 'flow' tile
arrangement
ax = nexttile;
% create an axes object
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y = 2.^x;
% 2 (elementwise) power by x
loglog(ax,x,y)
% display a log-log plot
```



Loglog8:

```
x = logspace(-2,3);
% generate a row vector x of 50 logarithmically spaced
points between decades 10^-2 and 10^3
y1 = 3.^x;
% create the first vector of y- coordinates
y2 = 1./3.^x;
% create the second vector of y- coordinates
lg = loglog(x,y1,x,y2);
% create the line objects
```



Q3

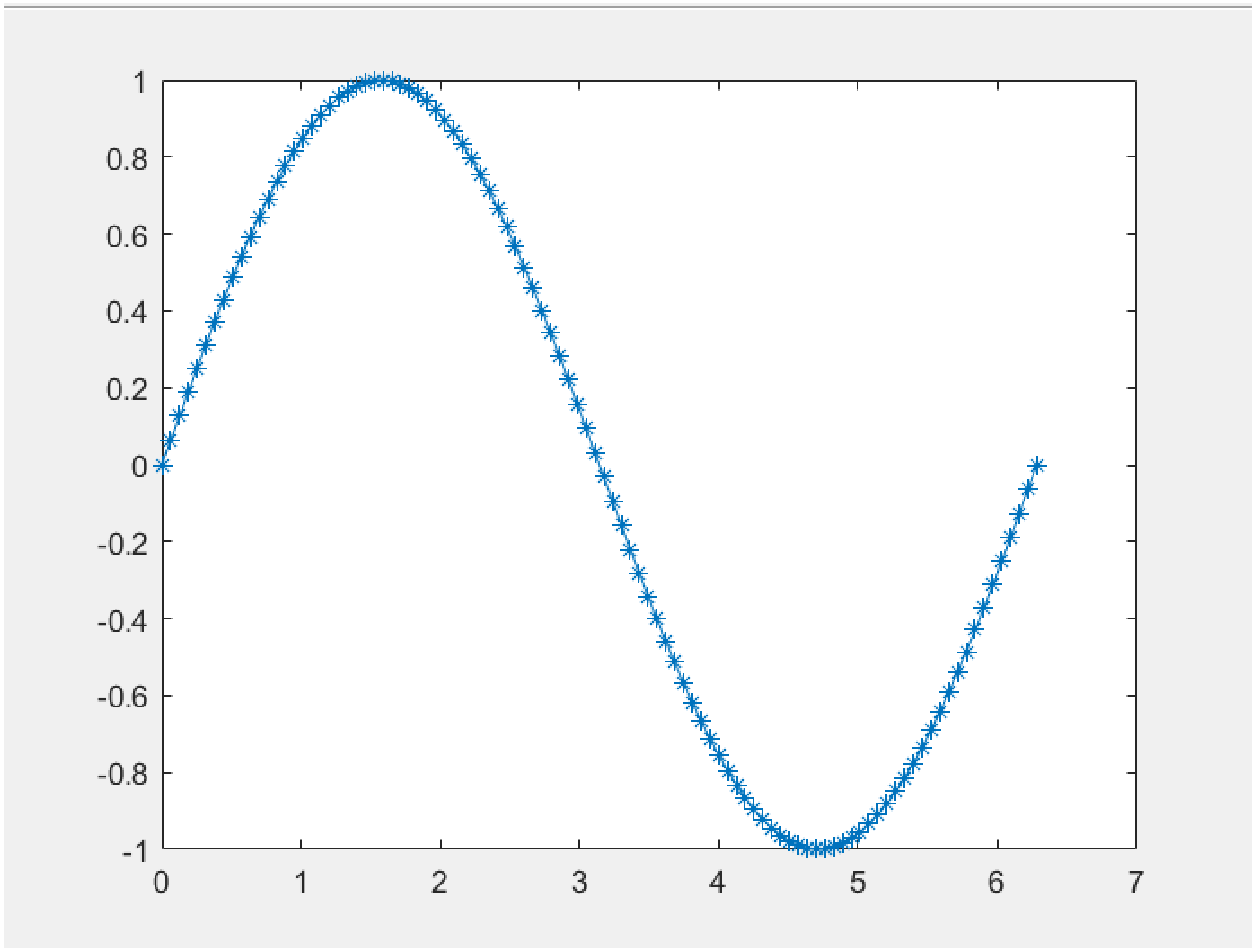
Syntax	Meaning	Description
1) axis(limits) 2) axis style 3) axis mode 4) axis ydirection 5) axis visibility 6) lim = axis 7) [m,v,d] = axis('state') 8) __ = axis(ax, __)	Log-log scale plot	1) This syntax specifies the limits for the current axes (as a vector of four, six, or eight elements). 2) This syntax uses a predefined style to set the limits and scaling. 3) This syntax sets whether MATLAB automatically chooses the limits or not. 4) This syntax places the origin at the upper left corner of the axes. 5) This syntax turns off the display of the axes background. The default for visibility is on.

		6) This syntax returns the x-axis and y-axis limits for the current axes. Note: For polar axes, returns the theta-axis and r-axis limits. 7) This syntax returns the current settings for the axis limit selection, the axes visibility, and the y-axis direction. 8) This syntax uses the axes or polar axes specified by ax instead of the current axes.

Examples:

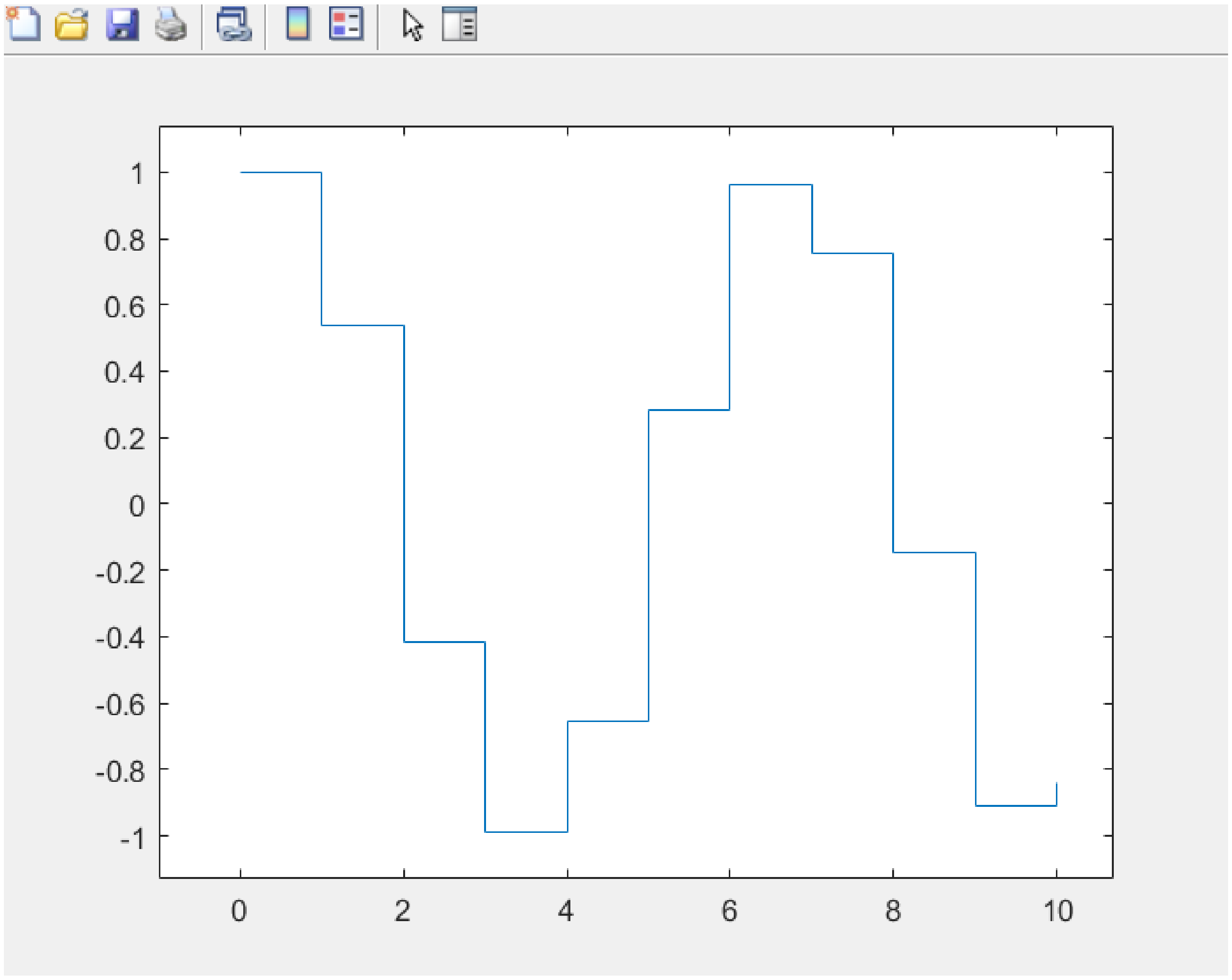
Axis1::

```
x = linspace(0,2*pi);
% create the x- coordinates
y = sin(x);
% create the y- coordinates (sine function of x)
plot(x,y,'-*')
% plot the sine function
```



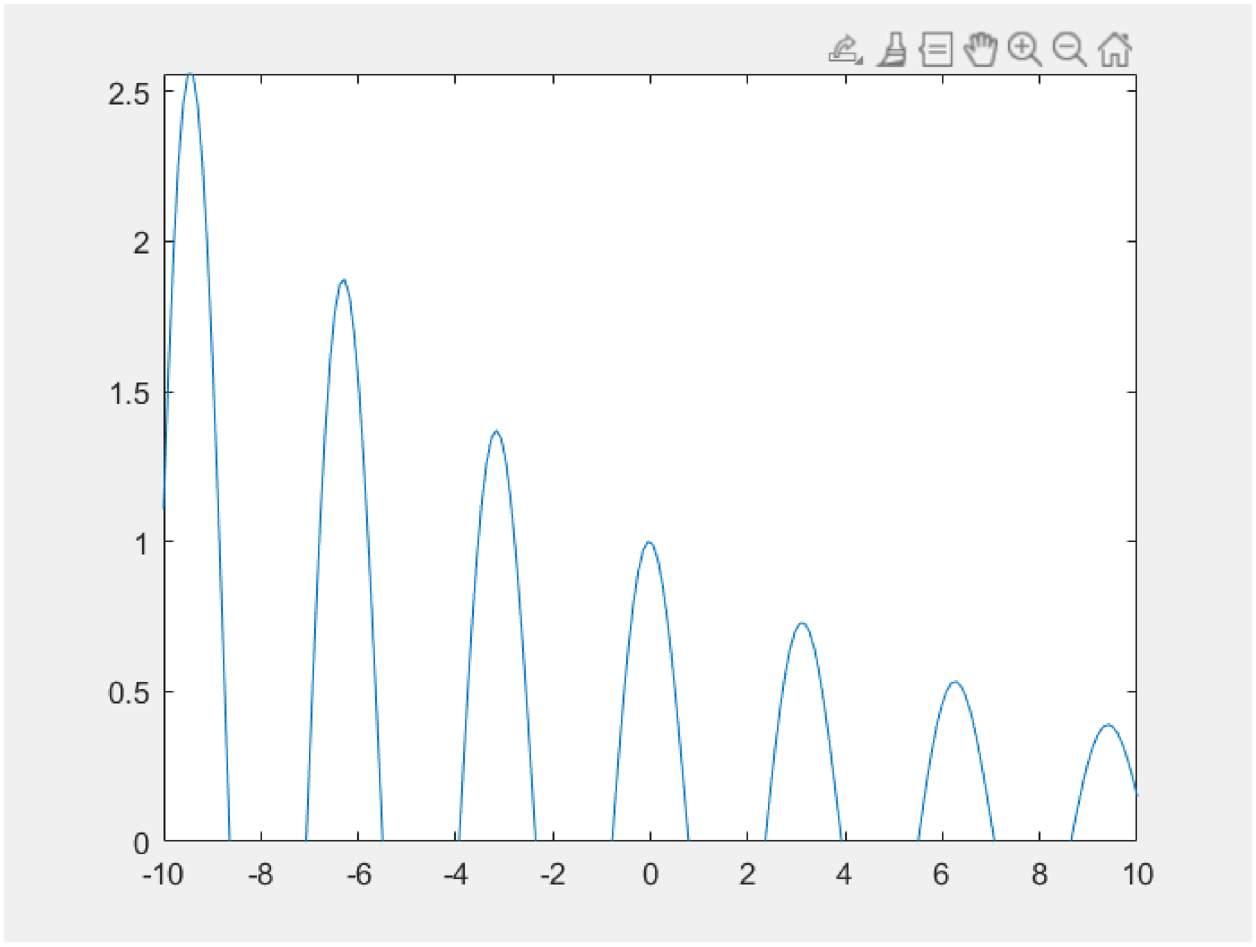
Axis2:

```
x = 0:10;
% create a vector of x- coordinates
y = cos(x);
% create a vector of x- coordinates (cosine function of x)
stairs(x,y)
% create a stairstep plot
axis padded
% add a margin of padding between the plot and the plot box
```



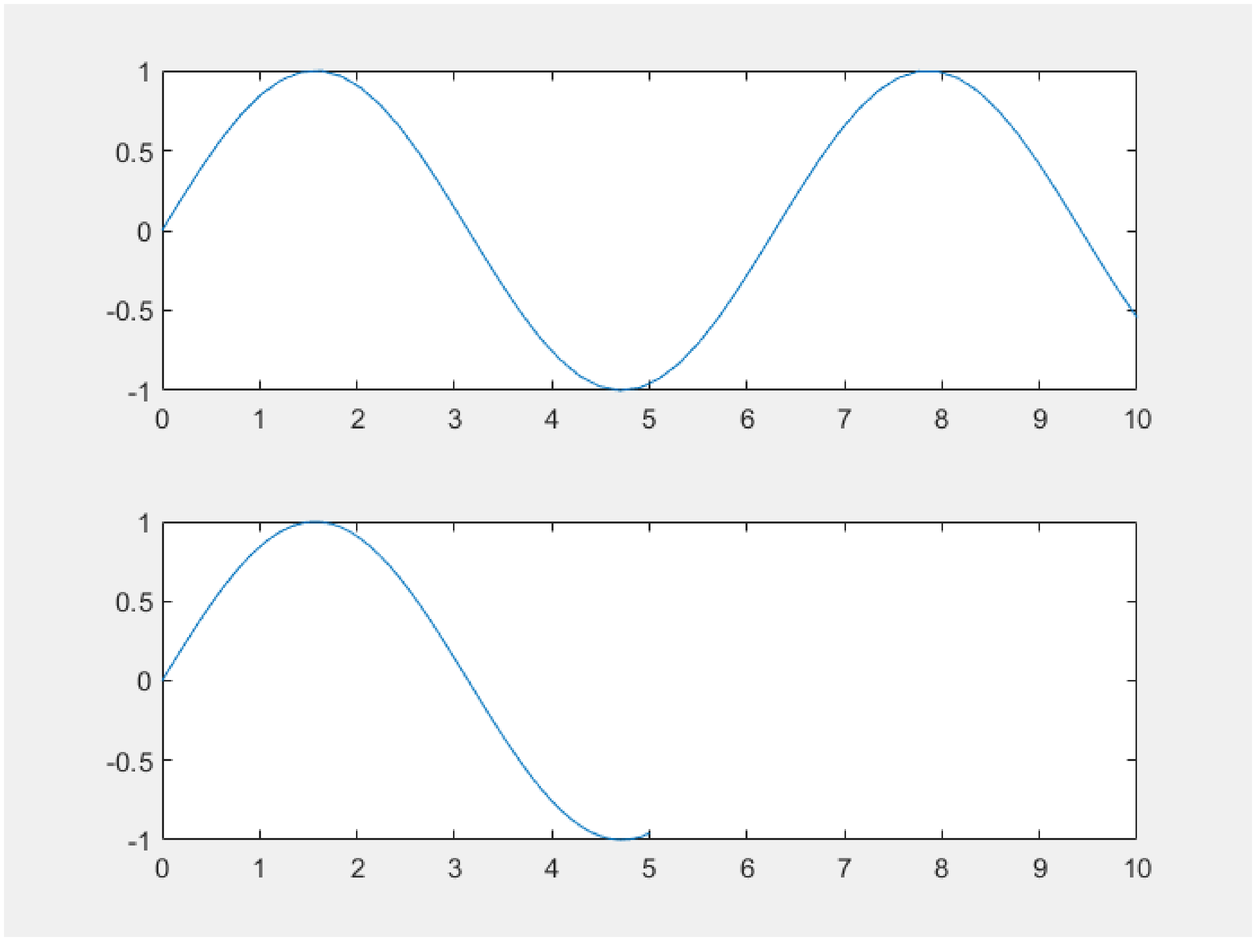
Axis3:

```
x = linspace(-10,10,200);
% create a set of x- coordinates
y = cos(2*x)./exp(.1*x);
% create a set of y- coordinates
plot(x,y)
% plot x and y
axis([-10 10 0 inf])
% use an automatically calculated value for the maximum y-
axis limit
```



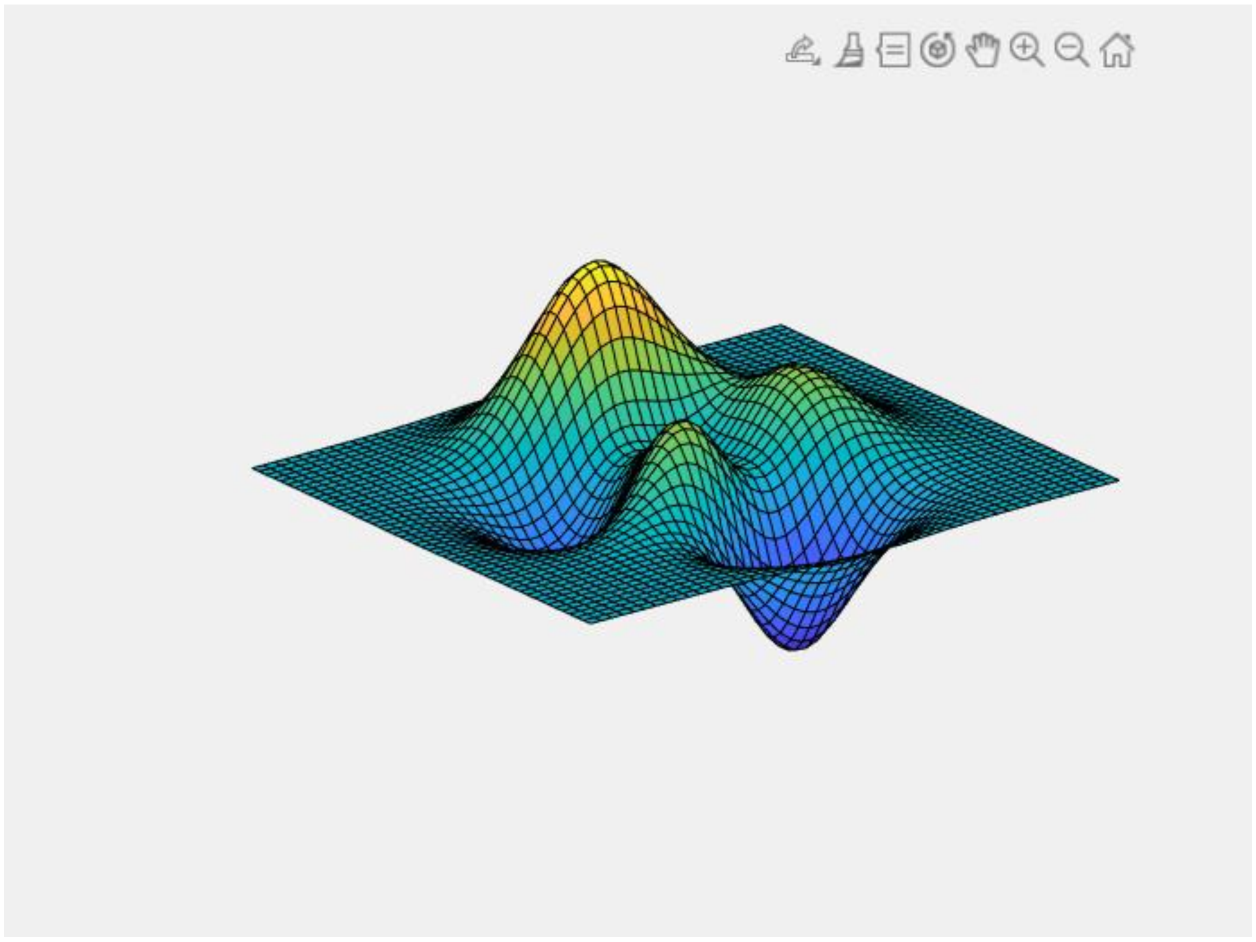
Axis4:

```
tiledlayout(2,1)
% create a 2-by-1 tiled chart layout
x1 = linspace(0,10,50);
% create a set of x- coordinates
y1 = sin(x1);
% create a set of y- coordinates
ax1 = nexttile;
% create the axes objects ax1
plot(ax1,x1,y1)
% plot data in ax1 axes

x2 = linspace(0,5,50);
% create a set of x- coordinates
y2 = sin(x2);
% create a set of y- coordinates
ax2 = nexttile;
% create the axes objects ax1
plot(ax2,x2,y2)
% plot data in ax2 axes

axis([ax1 ax2],[0 10 -1 1])
% set the axis limits for both ax1 and ax2 to the same
values
```


1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10
1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10

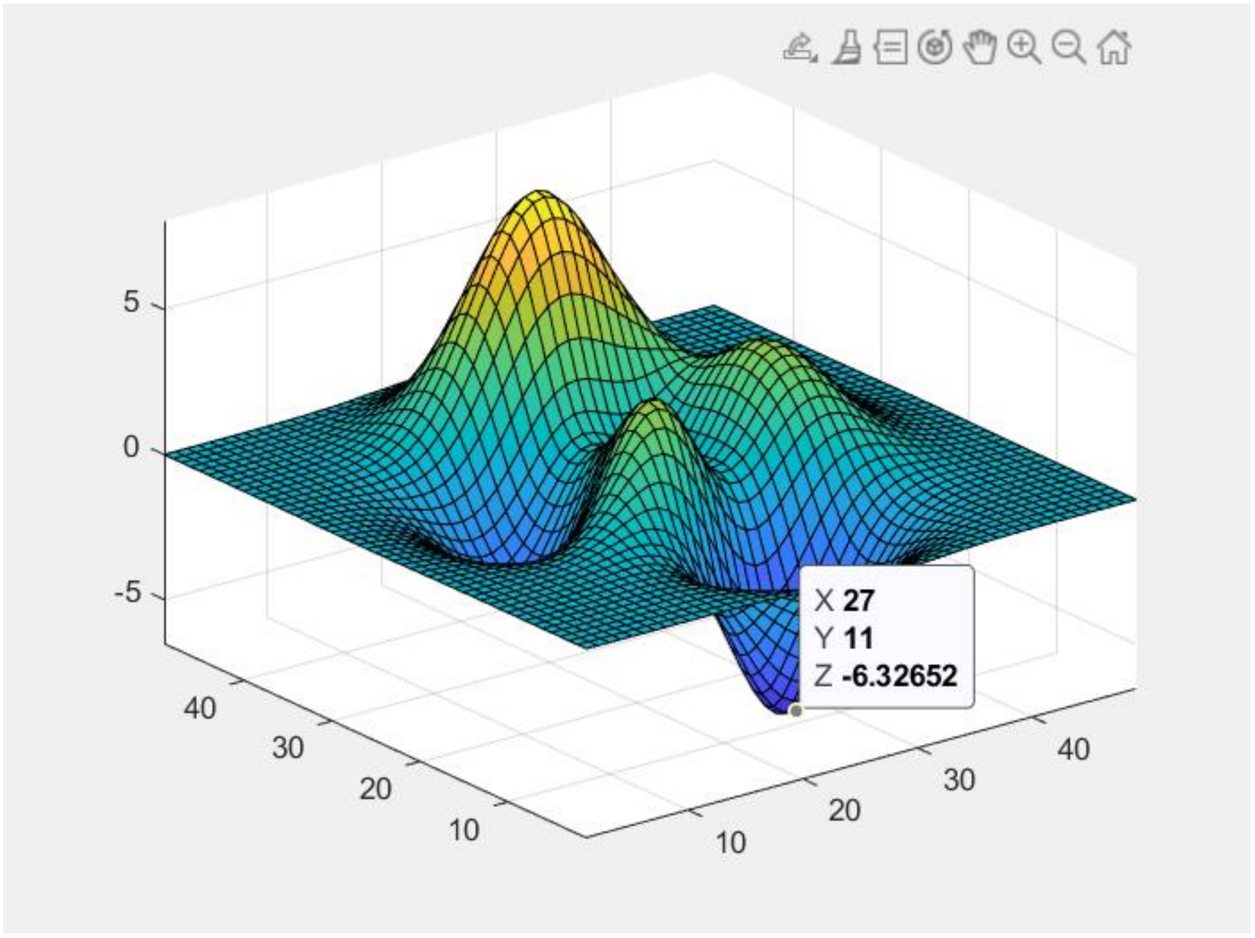
Axis5:

```
surf(peaks)
% plot the surface
axis off
% don't display the axes lines and background
```



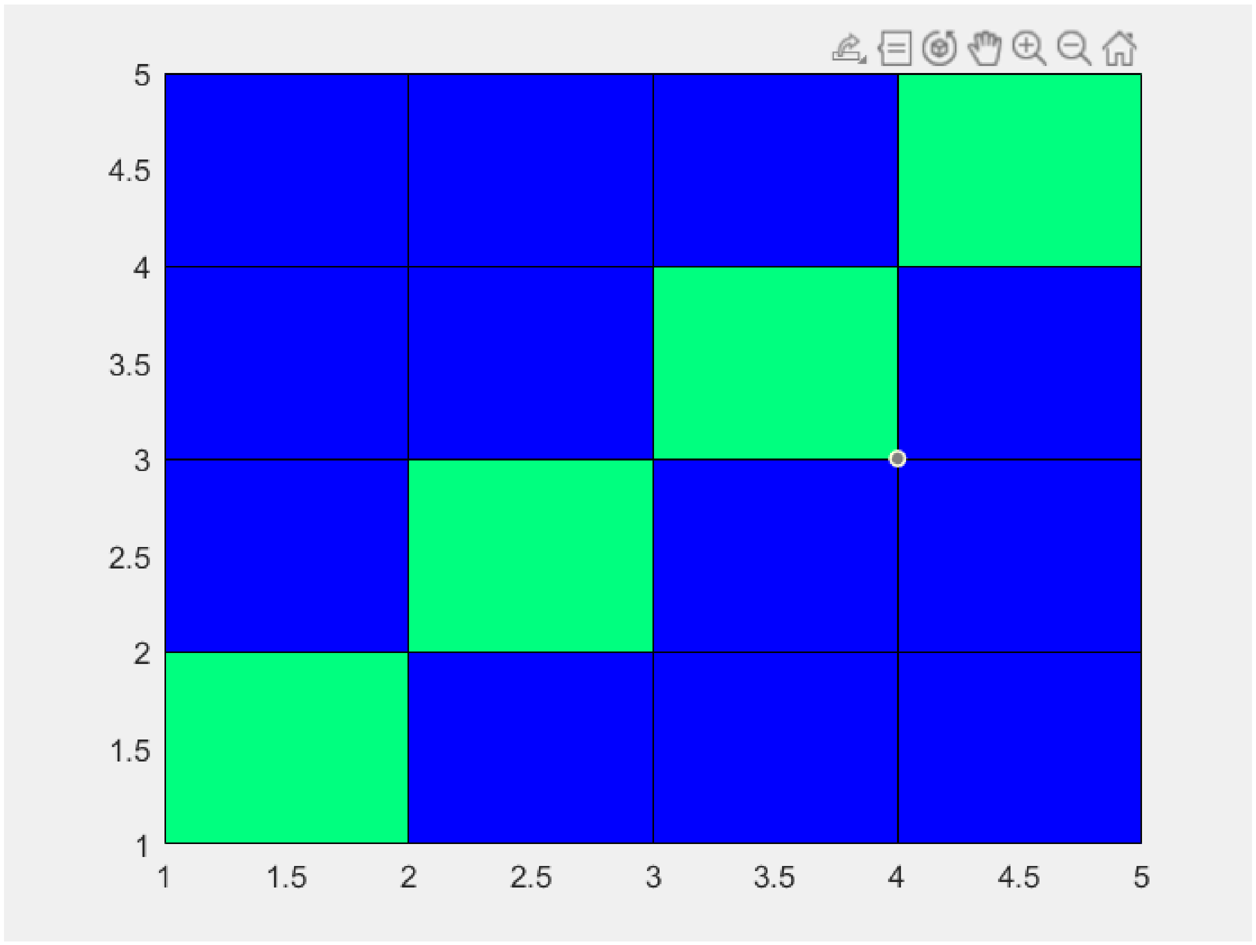
Axis6:

```
surf(peaks)
% plot a surface
axis tight
% set the axis limits to equal the range of the data
```



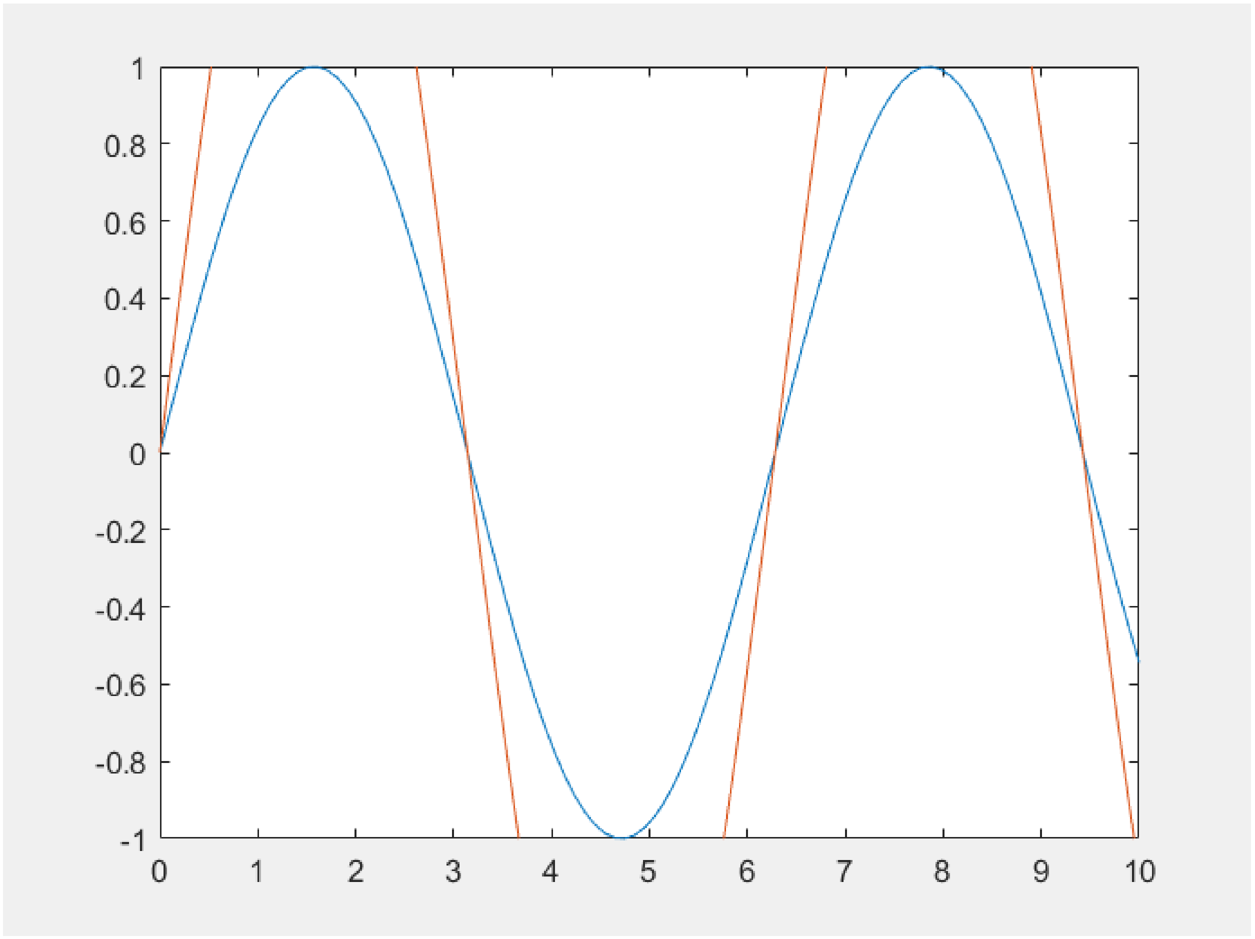
Axis7:

```
C = eye(5);
% create an idintity matrix of size 5
pcolor(C)
% create the plot
colormap winter
% specify the colors
```



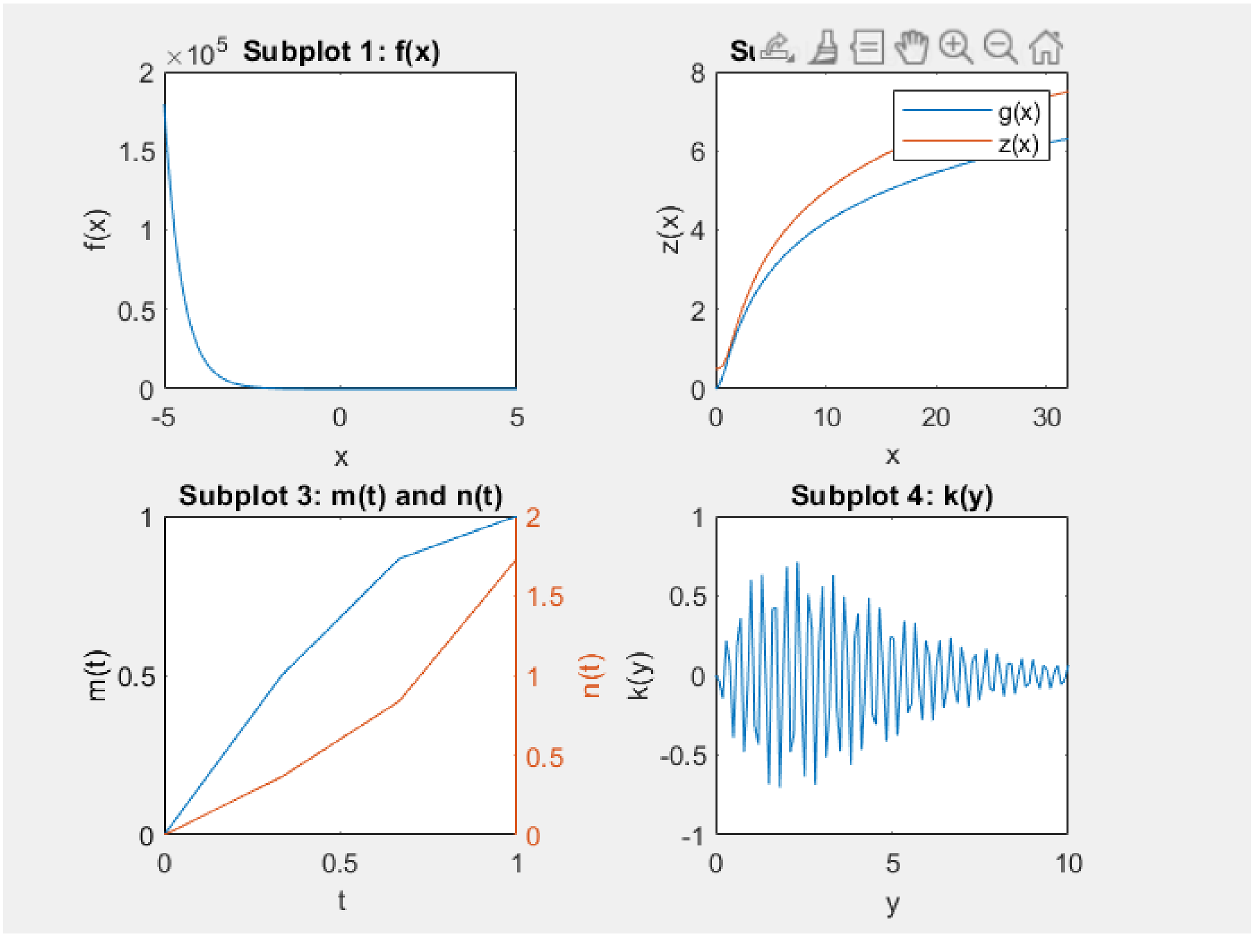
Axis8:

```
x = linspace(0,10);
% create a set of x- coordinates
y = sin(x);
% create a set of y- coordinates (sine function of x)
plot(x,y)
% plot x and y
y2 = 2*sin(x);
% create another set of y- coordinates (2 * sine function
of x)
hold on
% add another sine wave to the axes
axis manual
% keep the current axis limits
plot(x,y2)
% plot x and y2
hold off
```



Q4

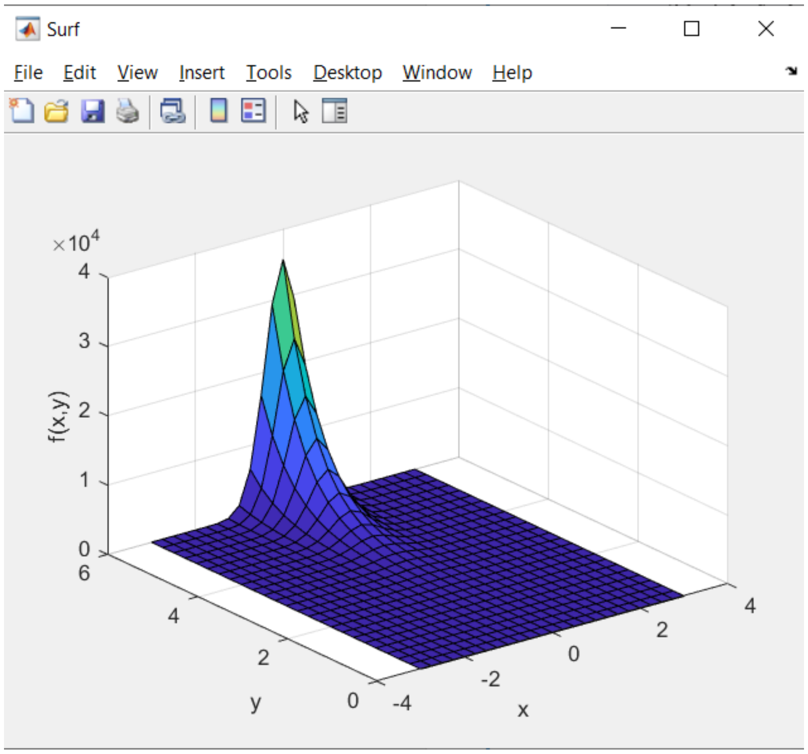
```
subplot(2,2,1)
x1 = linspace(-5,5);
% create a set of x- coordinates
f = 3 * exp(-2*x1+1);
% define the f(x) function
plot(x1,f)
xlabel('x')
ylabel('f(x)')
title('Subplot 1: f(x)')

subplot(2,2,2)
x2 = linspace(0.01,32);
% create a set of x- coordinates
g = log3(x2.^2+1);
% define the g(x) function using log3 fuction written by
myself
z = log4(x2.^3+2);
% define the z(x) function using log4 fuction written by
myself
plot(x2,g)
xlabel('x')
ylabel('g(x)')
hold on
% let us have more than one diagram in a plot
plot(x2,z)
xlabel('x')
ylabel('z(x)')
hold off
legend('g(x)','z(x)')
% label g(x) function and z(x) function in order to
recognize them
title('Subplot 2: g(x) and z(x)')

subplot(2,2,3)
t = linspace(0,1,4);
% create a set of x- coordinates
m = sin(pi*t/2);
% define the m(t) function
n = tan(pi*t/3);
% define the n(t) function
plot(t,m)
xlabel('t')
ylabel('m(t)')
yyaxis right
```

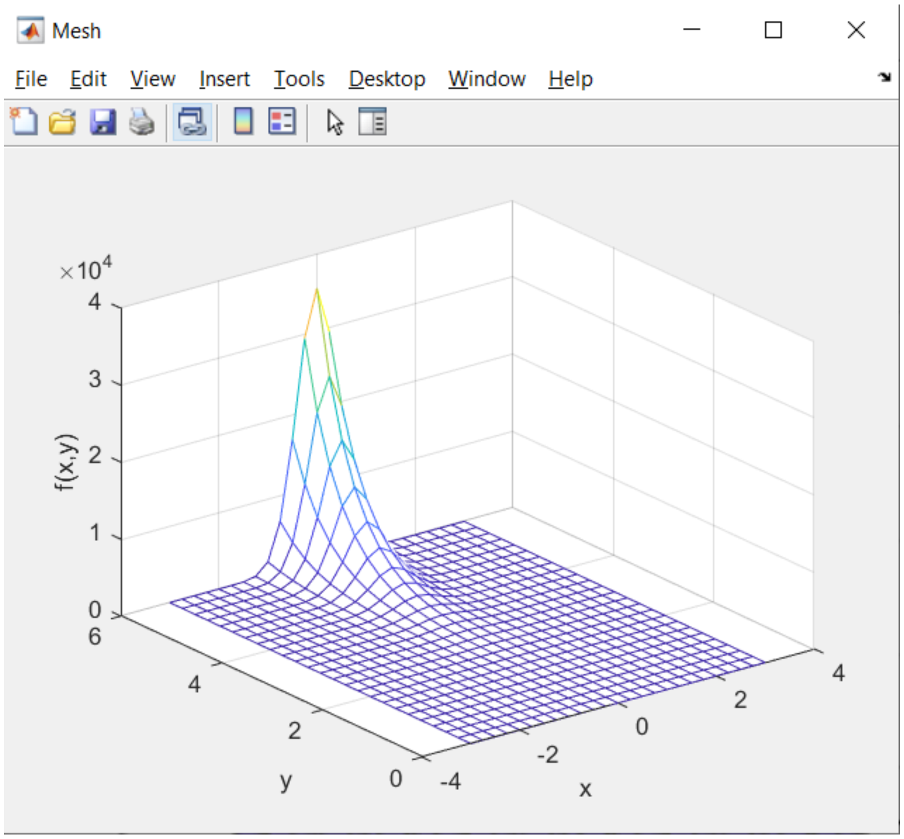

Subplot 1: f(x)
×10^5
f(x)
x
Su
g(x)
z(x)
z(x)
x
Subplot 3: m(t) and n(t)
m(t)
n(t)
t
Subplot 4: k(y)
k(y)
y

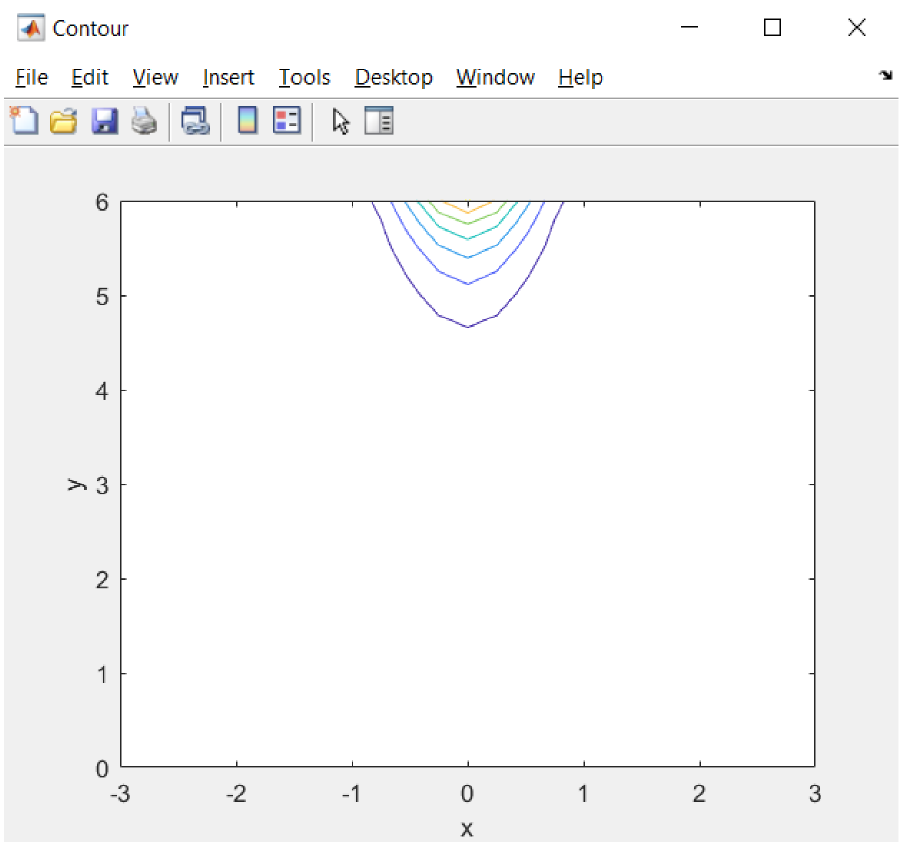
Q5

Syntax	Meaning	Description
1) [X,Y] = meshgrid(x,y) 2) [X,Y] = meshgrid(x) 3) [X,Y,Z] = meshgrid(x,y,z) 4) [X,Y,Z] = meshgrid(x)	2-D and 3-D grids	1) This returns 2-D grid coordinates based on the coordinates contained in vectors x and y. X is a matrix where each row is a copy of x, and Y is a matrix where each column is a copy of y. The grid is represented by the coordinates X and Y has length(y) rows and length(x) columns. 2) This is the same as [X,Y] = meshgrid(x,x). 3) returns 3-D grid coordinates defined by the vectors x, y, and z. The grid represented by X, Y, and Z has size length(y)-by-length(x)-by-length(z). 4) is the same as [X,Y,Z] = meshgrid(x,x,x).

For the first function:

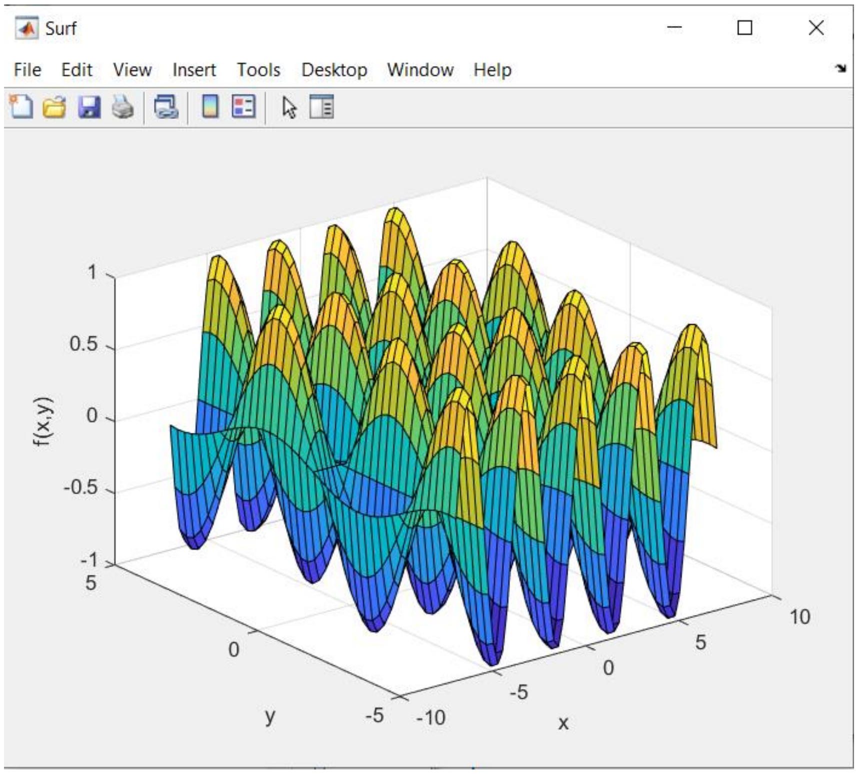


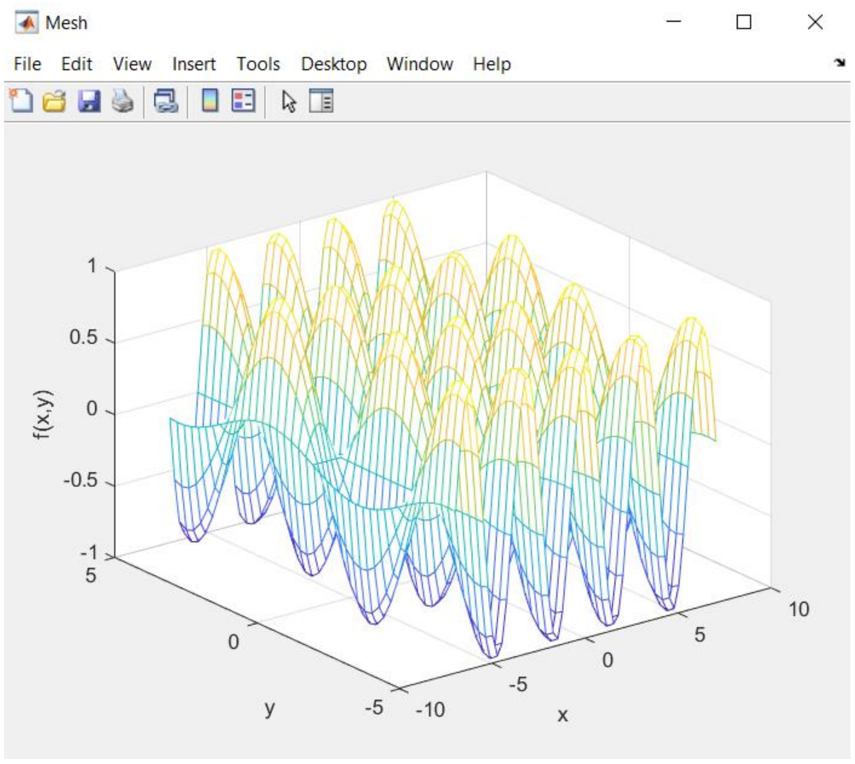

Mesh
File Edit View Insert Tools Desktop Window Help
$\times 10^4$
4
3
2
1
0
f(x,y)
6
4
2
0
y
-4
-2
0
2
4
x

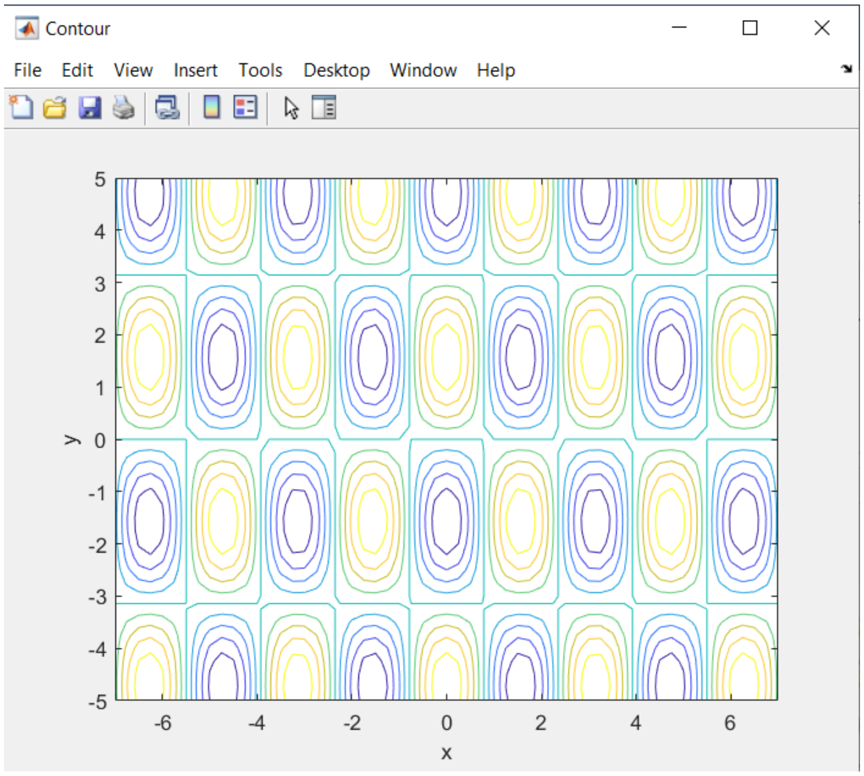

Contour
File Edit View Insert Tools Desktop Window Help
6
5
4
3
2
1
0
y
-3
-2
-1
0
1
2
3
x

<u>The plots are in separate figures as stated in the question.</u>

For the second function:







The code generating these figures is in the next page.

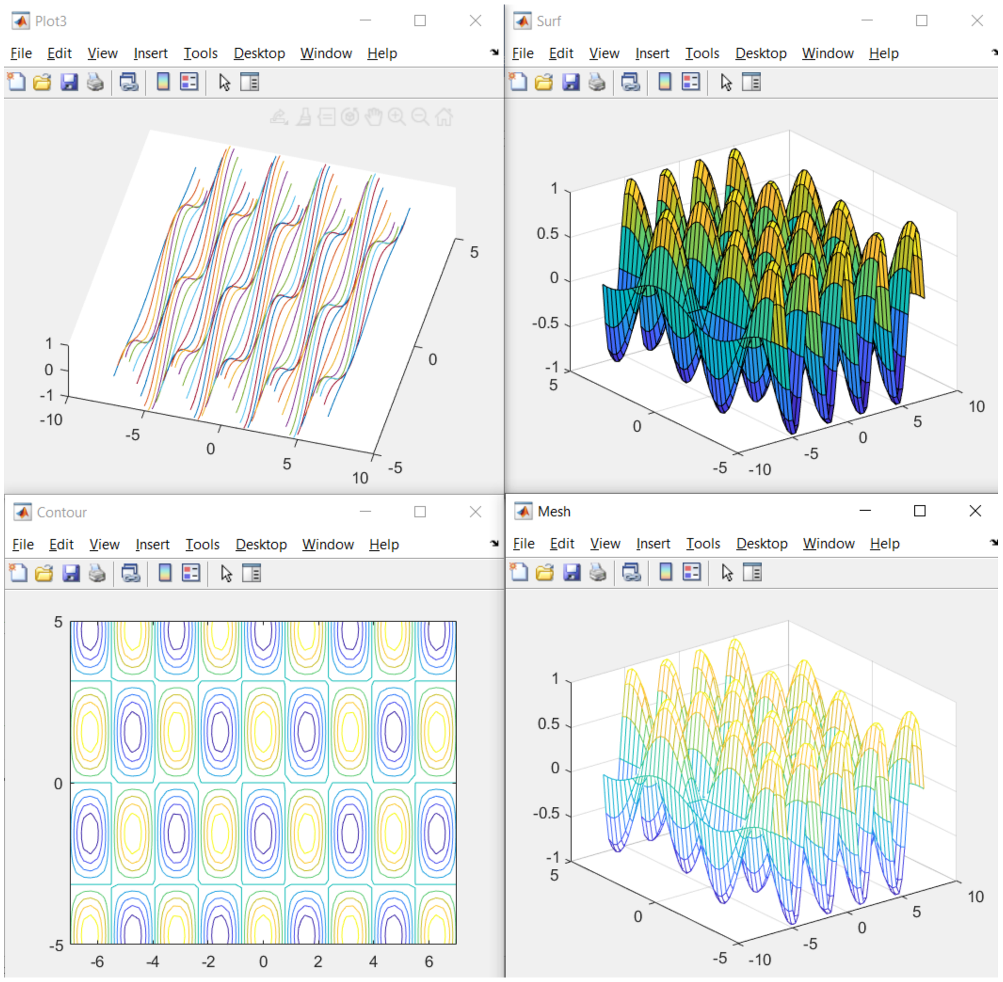
```
clc;
clear;

%first function:
x1 = -3:0.25:3;
y1 = 0:0.25:6;
[X1,Y1] = meshgrid(x1, y1); %meshgrid on x and y domain
F1 = (Y1.^(2.5)).*exp(-3*(X1.^2)+Y1); %f defenition
draw(X1, Y1, F1)

%second function
x2 = -7:0.25:7;
y2 = -5:0.25:5;
[X2,Y2] = meshgrid(x2, y2); %meshgrid on x and y domain
F2 = cos(2*X2).*sin(Y2); %f defenition
draw(X2, Y2, F2)

function draw(X, Y, F)
    %plot the mesh
    figure('Name','Mesh','NumberTitle','off'); %creates a
new figure window named Mesh
    mesh(X,Y ,F)
    xlabel('x') % lable of x axis
    ylabel('y') % lable of y axis
    zlabel('f(x,y)') % lable of z axis
    %plot the contour
    figure('Name','Contour','NumberTitle','off'); %creates
a new figure window named Contour
    contour(X ,Y ,F)
    xlabel('x')
    ylabel('y')
    zlabel('f(x,y)')
    %plot the surf
    figure('Name','Surf','NumberTitle','off'); %creates a
new figure window named Surf
    surf(X ,Y ,F)
    xlabel('x')
    ylabel('y')
    zlabel('f(x,y)')
end
```

Q6

plot3() plots some markers ('.', '*', etc.) at the specified points without making a solid surface between them. However surf() creates a solid surface between the points. The notable difference is that mesh draws a three-dimensional surface but plot3 only draws three-dimensional lines.

Mesh() creates a three-dimensional surface that has solid edge colors and no face colors so still, the difference between plot3() and surf() exists because the difference between mesh and surf is only about the surface color.

A contour plot is used to represent a 3-dimensional surface by plotting constant z slices, called contours, on a 2-dimensional format. This is like looking to the plot from upside however, plot3() is creating a 3d plot that we can look at from any angle.



```
clc;
clear;

x = -7:0.25:7;
y = -5:0.25:5;
[X,Y] = meshgrid(x, y); %meshgrid on x and y domain
F = cos(2*X).*sin(Y); %f defenition
draw(X, Y, F)

function draw(X, Y, F)
    %plot the mesh
    figure('Name','Mesh','NumberTitle','off'); %creates a
new figure window named Mesh
    mesh(X,Y ,F)
    %plot the contour
    figure('Name','Contour','NumberTitle','off'); %creates
a new figure window named Contour
    contour(X ,Y ,F)
    %plot the surf
    figure('Name','Surf','NumberTitle','off'); %creates a
new figure window named Surf
    surf(X ,Y ,F)
    %plot the surf
    figure('Name','Plot3','NumberTitle','off'); %creates a
new figure window named Surf
    plot3(X ,Y ,F)
end
```