Chapter 7

Graphics in MATLAB

1. Introduction

Visualization is one of the important tools to judge and analyze the numerical data and provide meaningful information about the data under study. MATLAB provides such feature to visualize data in different manners including, 2D plot, 3D plot, overlay plot and surface plot. Furthermore, MATLAB provides additional information with plots such as title, labels of axes, labels for data and grids and other useful information.

2. 2D plot

`plot (X, Y)`: Creates a 2-D line plot of the data in *Y* versus the corresponding values in *X*. The following example shows a script file introduced to plot the root square function in the interval [1 10].

plotXY.m

```
% Plotting the root square function in the interval [1 10].
X = 1:10;                                % x values.
Y = sqrt(X);                             % The corresponding y values.
figure                                   % opens a new figure window
plot(X, Y);                              % plot the data
```

In the above script, we define a set of x values and calculate their corresponding y values defined by $y = \sqrt{x}$. The function **`plot()`** is used to plot the corresponding data in two dimension 2D. The result of the introduced script is showcased in Figure.1.

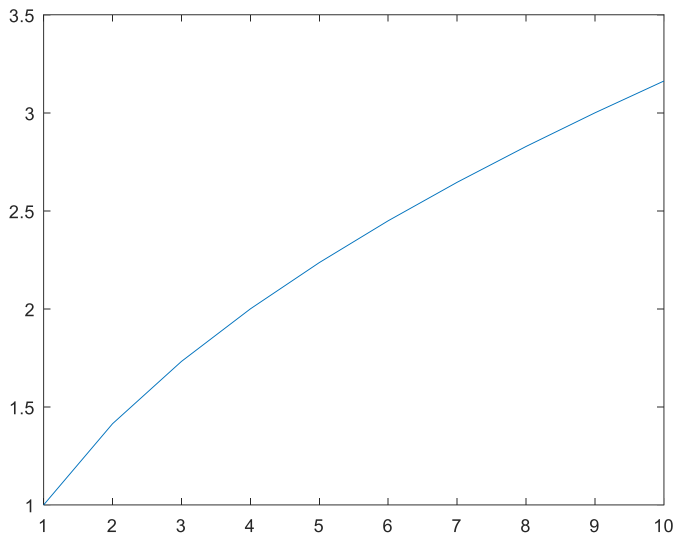


Figure.1 An example of a plot function

Let us modify our script by adding title and labels to axes:

plotXY.m

```
% Plotting the root square function in the interval [1 10].
X = 1:10;                              % x values.
Y = sqrt(X);                           % The corresponding y values.
figure                                 % opens a new figure window
plot(X, Y);                            % plot the data
xlabel('x value');                     % Label of the x axis.
ylabel('y = sqrt (x)');                % Label of the y axis.
title('Plot of root square function'); % Title of the plot.
```

The functions **`xlabel()`** and **`ylabel()`** are used to place the x label and y label in the graph respectively. The title of the graph is showed in the plot using the **`title()`** command. The result of the ameliorated script is showcased in Figure.2.

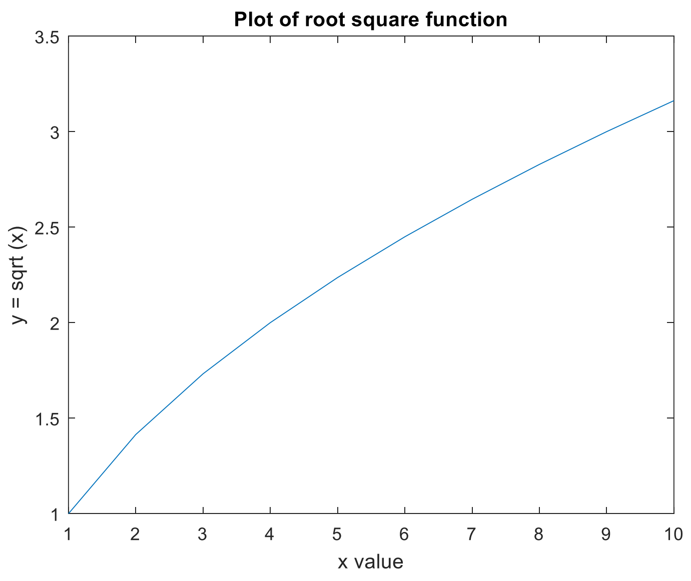


Figure.2 A plot with title and labeled axes

More specifications can be added to the plot command as indicated in what follows.

`plot(X,Y,LineSpecifiction)` where **`Linespecification`** sets the line style, marker symbol, and color which are specified as a character vector of symbols. The symbols can be placed in any order and we do not need to specify all three characteristics. An example of such specification is showcased in the next script.

plotXY.m

```
% Plotting the root square function in the interval[1 10].
X = 1:10;                                % x values.
Y = sqrt(X);                             % The corresponding y values.
figure                                   % opens a new figure window
plot(X, Y, 'r—o');                       % plot the data with specifications
xlabel('x value');                       % Label of the x axis.
ylabel('y = sqrt (x)');                  % Label of the y axis.
title('Plot of root square function');   % Title of the plot.
```

In the above example, the line is specified by `'r--o'` where `'r'` indicates the color of the line (red), `'--'` indicates the type of the line and `'o'` represents the marker of the points. Figure.3 shows the introduced specifications to the line of the curve.

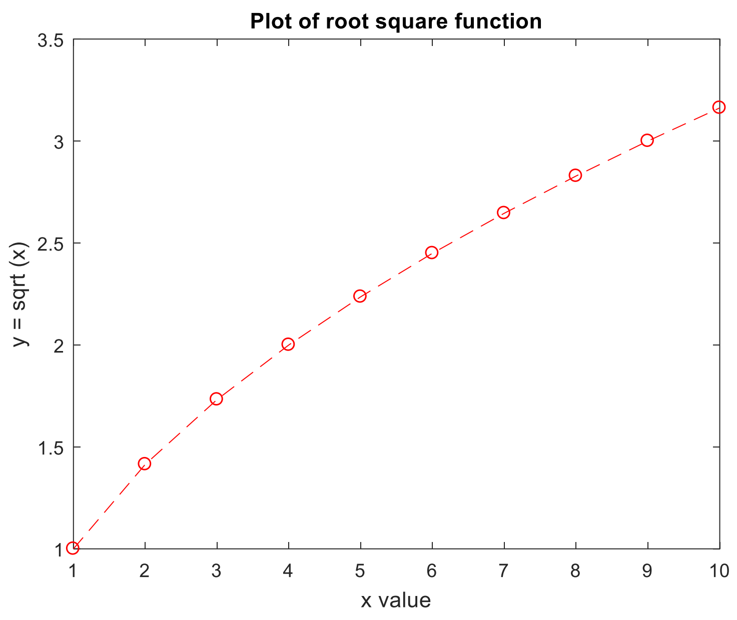


Figure.3 An example of a plot function with more specifications

3. Plotting Multiple Plots on the Same Graph

MATLAB provides the possibility of plotting multiple plots in the same graph. To plot multiple lines in the same figure using the same axes we use the notation indicated below.

`plot(X1,Y1,...,Xn,Yn)` : Creates n 2-D line plots of the data in Y_i versus the corresponding values in X_i. The next example shows a script file presented to plot three curves of three functions: $\sqrt{x}$, $2\sqrt{x}$ and $3\sqrt{x}$ in the interval [1 10].

plotXiYi.m

```
% Plotting three curves in the same plot.
X = 1:10;                            % x values.
% The corresponding yi values.
Y1 = sqrt(X);
Y2 = 2 * sqrt(X);
Y3 = 3 * sqrt(X);
figure                               % opens a new figure window
plot(X, Y1, X, Y2, X, Y3);           % plot the data
xlabel('x values');                  % Label of the x axis.
ylabel('y = sqrt (x)');              % Label of the y axis.
title('Plot of square root function'); % Title of the plot.
```

We can add more specifications to the corresponding lines as we adopted in the previous graph. In addition, we can add a legend to differentiate between the plotted curves. Therefore, the above script is modified as given below.

plotXiYi.m

```
% Plotting three curves in the same plot.
X = 1:10;                                     % x values.
% The corresponding yi values:
Y1 = sqrt(X);
Y2 = 2 * sqrt(X);
Y3 = 3 * sqrt(X);
figure                                        % opens a new figure window
plot(X, Y1,'r', X, Y2,'go', X, Y3, 'b--*'); % plot the data
xlabel('x values');                           % Label of the x axis.
ylabel('y = a*sqrt (x)');                     % Label of the y axis.
title('Plot of root square functions');       % Title of the plot.
legend(' y1', ' y2',' y3');                   % Legend of the plot.
```

In the plot function of this script, we have added the following specifications to the lines of the corresponding curves:

- 'r': Use a red line with no markers for the first curve;
- 'go' : Use only green circles markers for the second curve;
- 'b--*' : Use a blue dashed line with star markers for the third curve ;

The command legend is introduced to add a legend to the curve

Figure.4 shows the corresponding plot of the three curves.

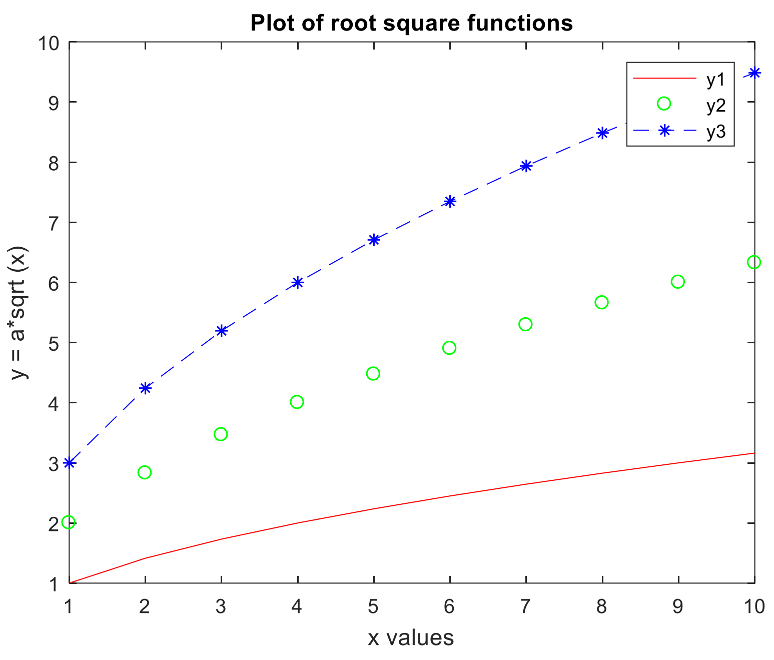


Figure.4 Plotting three curves in same view

More specifications can be added to the curves such as line width, marker size, and marker color. The reader can refer to the MATLAB help for more details.

To add new plots to existing one, we use the command **`hold on`**. To turn off the addition of the plots in same graph we use the notation **`hold off`**. Next command window shows the use of hold command.

```
>> x = linspace(-pi,pi);

>> y1 = sin(x);

>> plot(x,y1)

>> hold on

>> y2 = sin(2*x);

>> plot(x,y2)

>> hold off
```

4. Plotting Multiple Plots in Separate views

To show in a separate view multiple graphs, we use the **`subplot()`** command defined by the notation below.

`subplot(row,coloumn,position)` : This command creates a plot with *row-by-column* grid. The term **`position`** represents the position of the subplot in the grid, where the first subplot is the first column of the first row, the second subplot is the second column of the first row, and so on (subplot positions are numbered by rows). The below script represents an example of plotting two graphs in two subplots, where Figure 5 elucidates the corresponding execution of this script.

`subplot(2,1,1)` means that the plot will be on the first position of the *2-by-1* grid.

`subplot(2,1,2)` means that the plot will be on the second position of the *2-by-1* grid.

SubPlots.m

```
% Plotting multiple plots in separate views.
X = linspace(0,10);         % x values.
% The corresponding yi values.
Y1 = sqrt(X);
Y2 = sin(X);
figure
subplot(2,1,1);            % View 1
plot(X, Y1);               % plot  data 1
title('SubPlot 1 : y1');   % Title of plot 1.

subplot(2,1,2);            % View 2
plot(X, Y2);               % plot  data 2
title('SubPlot 2 : y2');   % Title of plot 2.
```

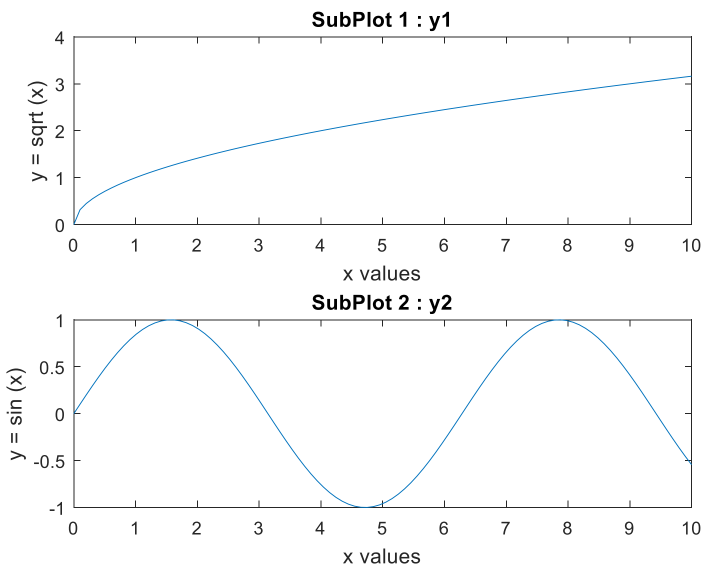


Figure.5 Plotting Multiple Plots in Separate views

5. Other specialized 2D plots

MATLAB offers several specialized 2D plots including: Bar charts, histograms, area, stem...

5.1. Functions `bar()` and `barh()`

These functions display data in bar charts in which **`bar()`** function draws vertical bars and **`barh()`** draws data in horizontal bars. An example of using these functions is indicated below in subplots.

bar_barh.m

```
% bar() and barh() functions.
Y = 2021:2023;
Pop = [4000, 5000, 6000];
figure
subplot(1,2,1);            % View 1
bar(Y, Pop);               % plot vertical bars
xlabel('Year');
ylabel('Population');
title('bar()');            % Title of plot 1.
subplot(1,2,2);            % View 2
barh(Y, Pop);              % plot horizontal bars
xlabel('Year');
ylabel('Population');
title('barh()');           % Title of plot 2.
```

The corresponding figure of the above MATLAB code is illustrated in the next figure.

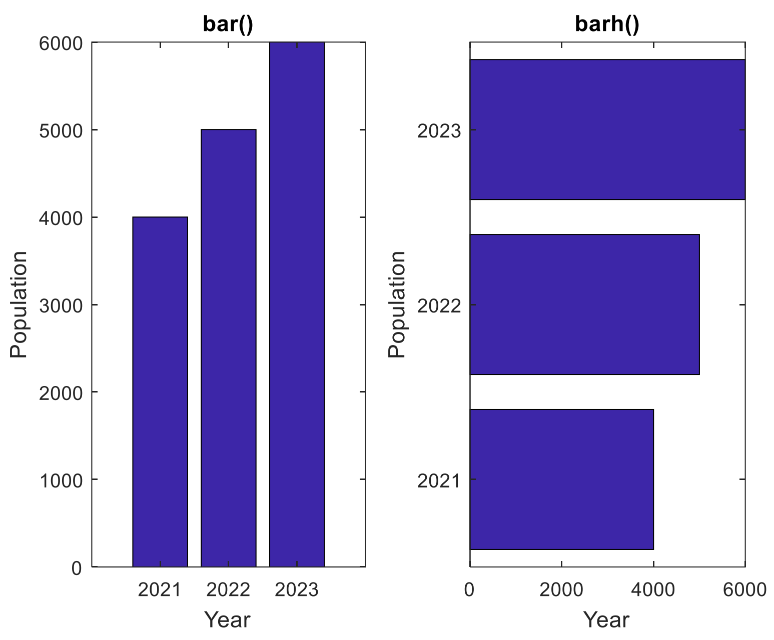


Figure.6 An example of bar and barh plots

5.2. Functions `histogram()` and `pie()`

MATLAB has a function **`histogram()`**, which is a particular type of bar chart illustrating the frequency of occurrence of values in a vector. The **`pie()`** function creates a pie chart and draws the percentage of each element in a vector starting from the top of the circle and going around counterclockwise.

An example of using the **`histogram()`** function is presented in the following command line which analyses the distribution of exam scores for a group of students.

```
>> data = [15, 12, 13, 15, 13, 16, 17, 15, 12, 14, 12, 15, 16];

>> histogram(data);
```

Figure. 7 illustrates the corresponding histogram plot of the above code.

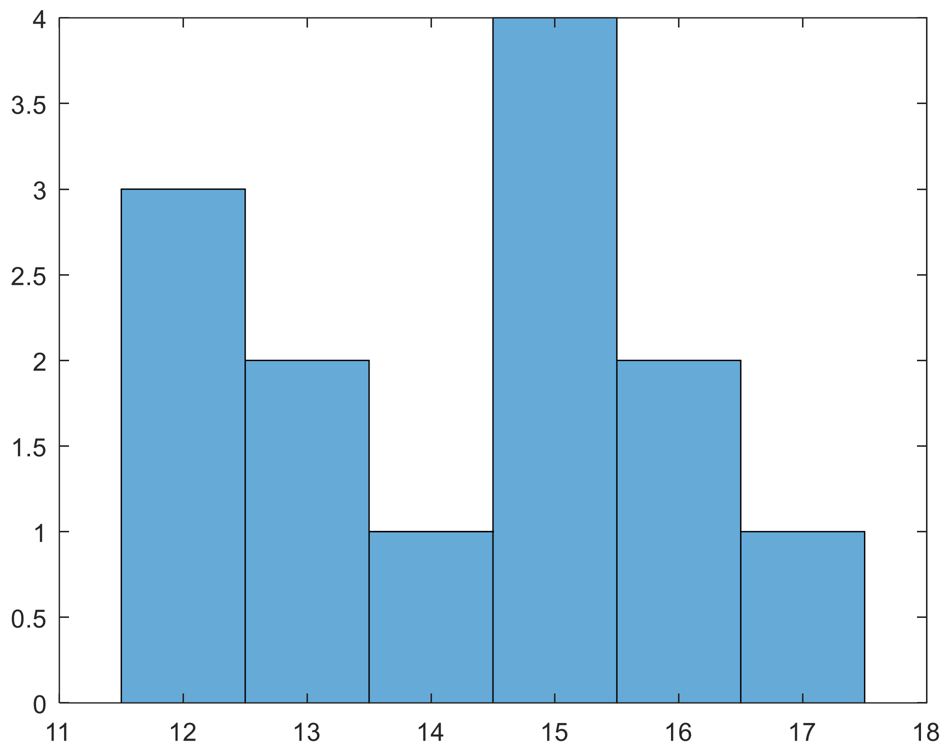


Figure.7 An example of a histogram plot

In the following script we add more specifications to our histogram, including title, labels, bin width, face color and show the grid.

Histogram_exmpl.m

```
% histogram of scores of 12 students.
data = [15, 12, 13, 13, 16, 17, 15, 12, 14, 12, 15, 16];


figure
% plot  the histogram
histogram(data, 'BinWidth',2, 'FaceColor', 'red');
% Add title and labels.
title('Distribution of Exam scores');
xlabel('Score Ranges');
ylabel('Number of students');

grid on; % add grid lines to the current axes.
```

In the `histogram` function of this script, we have added the following specifications to the corresponding plot:

BinWidth: Sets the bin size to 2 (12-14, 14-16, 16-18)

FaceColor: Sets the color of the bars red.

grid: Adds grid lines to make the plot easier to read

Figure 8, elucidated the specified histogram.

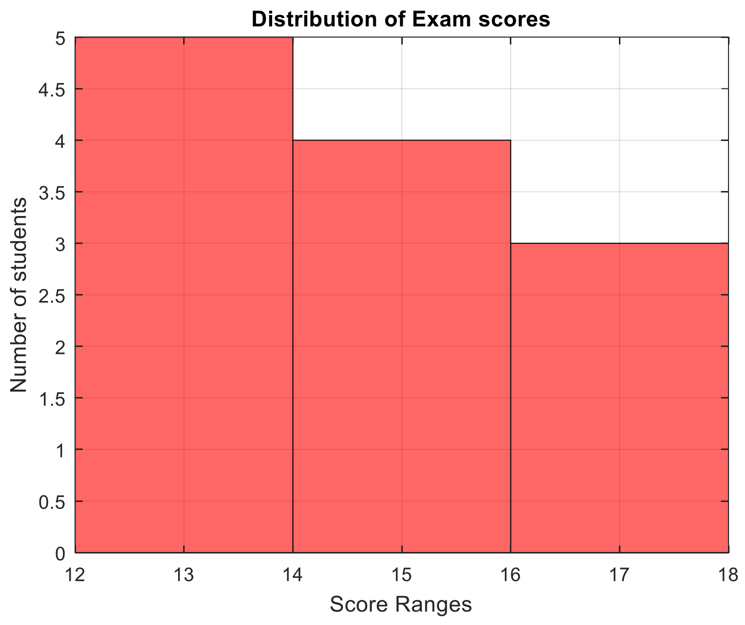


Figure.8 An example of histogram plot with more specifications

The next example introduces the use of the **pie()** function to highlight the distribution of students in four specializations.

Pie_example.m

```
nbstudents = [30, 20, 15, 5];
Spec = {'GP', 'GC', 'ELT', 'EL'};
figure
pie(nbstudents, Spec);                    % plot pie chart
title('Students by specialty');      % Title of the plot 2.
```

Figure.9 illustrates the corresponding histogram and pie chart of the above example.

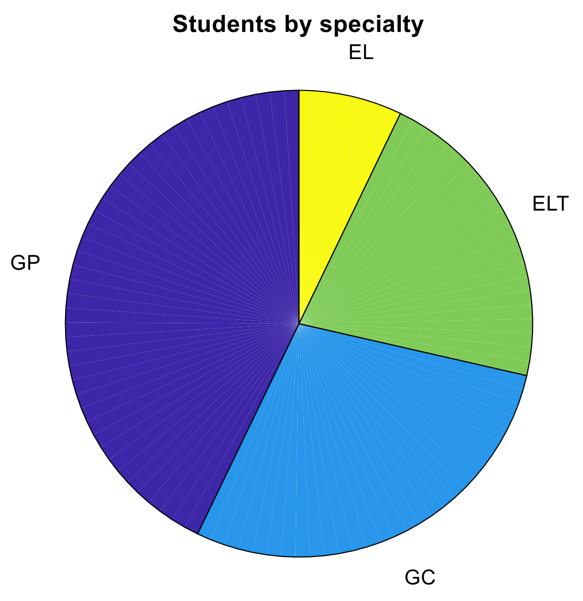


Figure.9 An example of pie plot

6. 3D plots

MATLAB provides many functions to display 3D plots. In general, the 3D functions have the same name of the 2D functions with the addition of 3 in the end of the 2D function: **`plot3()`**, **`bar3()`**, **`barh3()`**, **`pie3()`**, …..

An example of the 3D plots is given in the next script file for the **`plot3()`** function.

plot3_function.m

```
% plot3 function.
a = -pi:pi/50:5*pi;
figure
plot3(sin(a), cos(a),a);              % plot 3D graph
xlabel('sin(a)');                     % Label of the x axis.
ylabel('cos(a)');                     % Label of the y axis.
zlabel('a');                          % Label of the z axis.
title('Example of plot3() function'); % Title of the plot.
```

The corresponding graph of the given example is shown in figure.10.

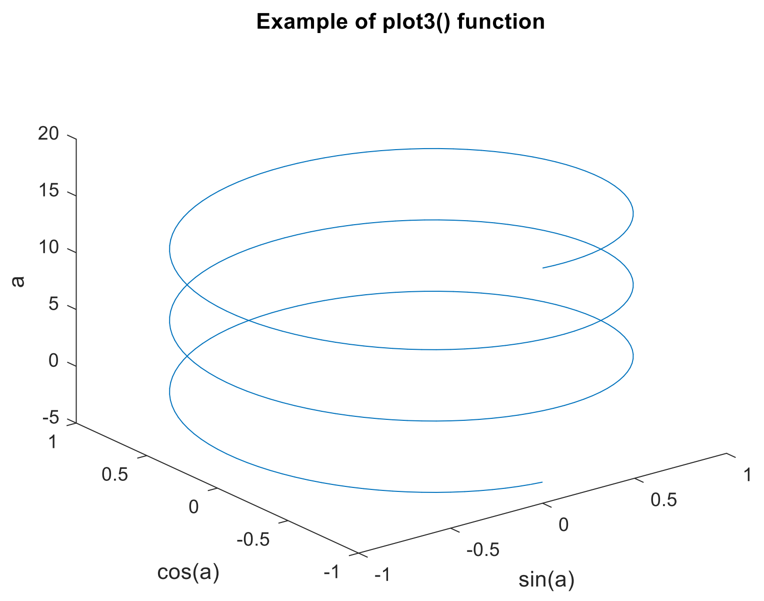


Figure.10 An example of 3D plot

By clicking on the 3D icon, the user can rotate easily the plot to see it from different angles.

For more specifications and visualization styles of the plot, the reader can refer to the MATLAB help page.

7. Practical work 7

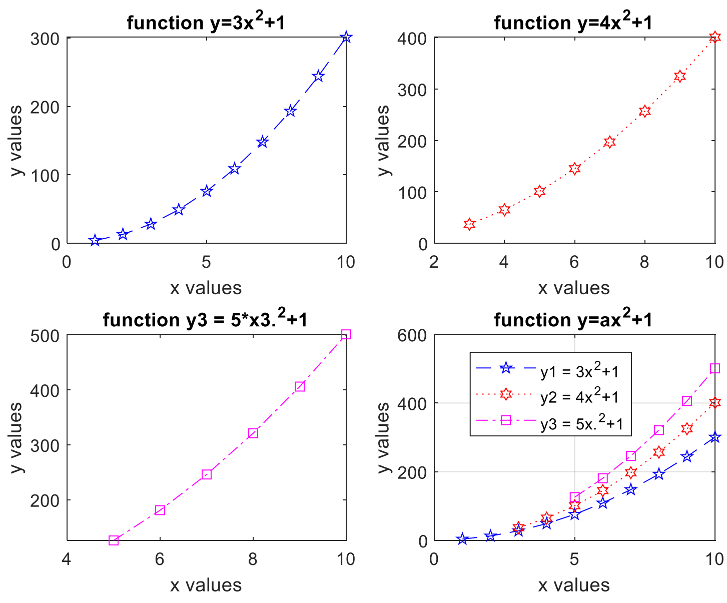
1. Using the plot function, plot the corresponding graph of the *1/square* function in the interval [100 300].
2. In same graph, plot f_1, f_2 and f_3 functions in the interval $[0\ \ 10\pi]$ arranged by the step $\pi/20$. The functions *f*1, *f*2 and *f*3 are given by: $f_1(x) = xcos(x)$, $f_2(x) = 2xcos(x)$ and $f_3(x) = 3xcos(x)$.
   - Use different line styles and colors for each function.
   - Add labels and legend to distinguish the three functions.
3. Write a script that creates vertical and horizontal bar graphs in separate views to represent the number of books sold by a bookstore in six months:
   data = [250, 340, 170, 200, 80], months = {'January', 'February', 'Mach', 'April', 'May', 'June'}.
   - Add appropriate titles, labels, grid and customize the bar colors.
4. Generate 100 random age values ranging from 10 to 80 and plot the corresponding histogram with a bin width of 10. Add the title and label the axes.

Hint: To create `n` random age values we use this notation: `ages = randi([min, max], 1, n).`

5. Plot the corresponding 3D plot of the x, y and z values defined by:
   $x = sin(t) \times cos(30t);$
   $y = sin(t) \times sin(30t);$
   $z = cos(t);$
   where t is defined in the interval $[0 \;\; \pi]$ arranged by the step $\pi/500$.

8. Exercises

1. Plot the function $y = 5x^2 - 2x + 1$
2. On the same figure, plot the following functions Y1 = cos(x), Y2 = sin(x) and Y3 = sin(2x) for $x = 0{:}0.1{:}10$ using different colors, a legend and different line width.
3. Using subplot, draw the following functions using the same range x = 1:0.1:10
   $log(x),\ exp(x),\ x^2\ and\ \sqrt{x}$
4. Generate 200 random integers ranging from 10 to 200 and plot the corresponding **bar**. Add the title and label the axes.
5. Give a script that plot the following figure.


function y=3x^2+1
y values
x values
function y=4x^2+1
y values
x values
function y3 = 5*x3.^2+1
y values
x values
function y=ax^2+1
y1 = 3x^2+1
y2 = 4x^2+1
y3 = 5x.^2+1
y values
x values