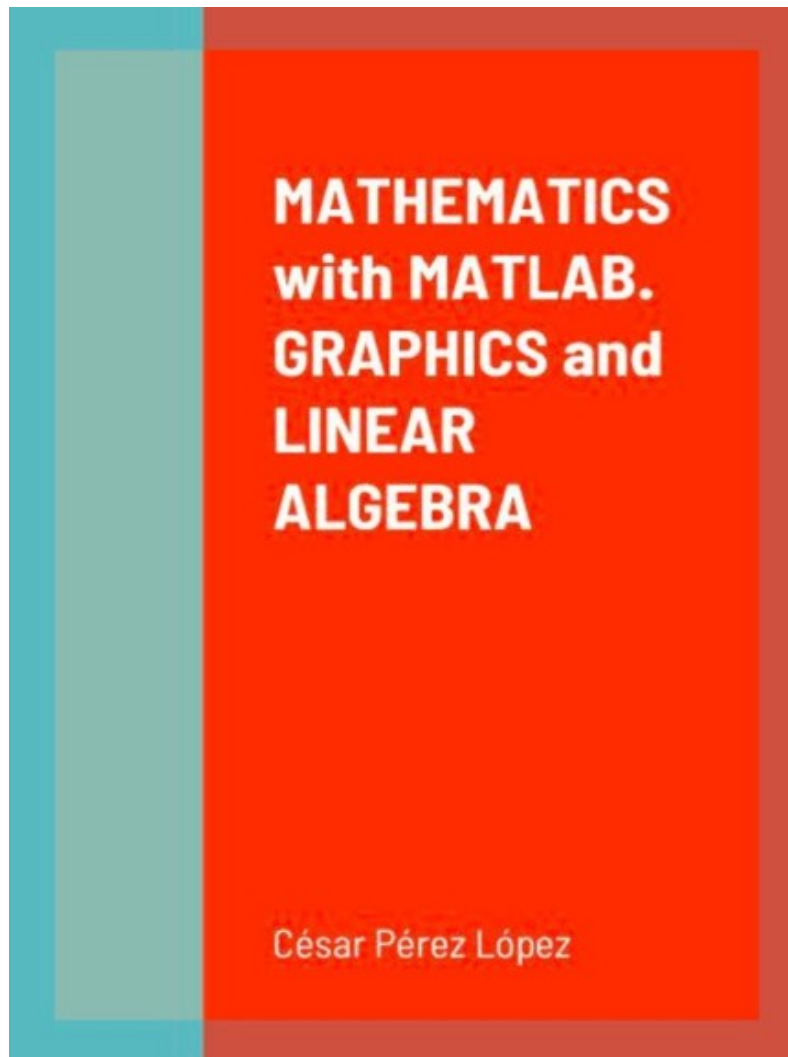


MATHEMATICS
with MATLAB.
GRAPHICS and
LINEAR
ALGEBRA

César Pérez López



**MATHEMATICS with MATLAB.
GRAPHICS and
LINEAR ALGEBRA**

César Pérez López

Table of Contents

[INTRODUCTION](#)

[INTRODUCTION TO MATLAB](#)

[1.1 MATLAB LANGUAGE](#)

[1.2 STARTING MATLAB ON WINDOWS. THE MATLAB](#)

[WORKING ENVIRONMENT](#)

[1.3 NUMERICAL COMPUTATION WITH MATLAB](#)

[1.4 SYMBOLIC CALCULATIONS WITH MATLAB](#)

[1.5 GRAPHICS WITH MATLAB](#)

[1.6 HELP WITH COMMANDS](#)

[MATLAB LANGUAGE ELEMENTS. VARIABLES,](#)

[NUMBERS, OPERATORS AND FUNCTIONS](#)

[2.1 VARIABLES](#)

[2.1.1 VECTOR VARIABLES](#)

[2.1.2 MATRIX VARIABLES](#)

[2.1.3 CHARACTER VARIABLES](#)

[2.2 NUMBERS](#)

[2.2.1 INTEGERS](#)

[2.2.2 FUNCTIONS OF INTEGERS AND DIVISIBILITY](#)

[2.2.3 ALTERNATIVE BASES](#)

2.2.4 REAL NUMBERS

2.2.5 FUNCTIONS WITH REAL ARGUMENTS

2.2.6 COMPLEX NUMBERS

2.2.7 FUNCTIONS WITH COMPLEX ARGUMENTS

2.2.8 ELEMENTARY FUNCTIONS THAT SUPPORT
COMPLEX VECTOR ARGUMENTS

2.2.9 ELEMENTARY FUNCTIONS THAT SUPPORT
COMPLEX MATRIX ARGUMENTS

2.2.10 RANDOM NUMBERS

2.3 OPERATORS

2.3.1 ARITHMETIC OPERATORS

2.3.2 RELATIONAL OPERATORS

2.3.3 LOGICAL OPERATORS

2.3.4 LOGICAL FUNCTIONS

2-D GRAPHICS IN MATLAB. EXPLICIT, PARAMETRIC
AND POLAR CURVES. EXPLORATORY GRAPHS

3.1 BASIC COMMANDS FOR 3-D GRAPHICS

3.2 ADDITIONAL ELEMENTS IN A GRAPHS (TITLES,
TAGS, MESHES AND TEXTS)

3.3 WORKING WITH AXES AND MULTIPLE GRAPHS

3.4 LOGARITHMIC AND SEMI-LOGARITHMIC GRAPHS

3.5 POLYGONS IN TWO DIMENSIONS

3.6 2-D GRAPHICS IN PARAMETRIC COORDINATES

3.7 2-D GRAPHICS IN POLAR COORDINATES

3.8 EXPLORATORY GRAPHS. BARS AND SECTORS
GRAPS. HISTOGRAMS

3.9 STATISTICAL ERRORS AND ARROWS GRAPHICS

3-D GRAPHICS IN MATLAB. 3-D CURVES ON SPACE,
SURFACES, MESHES AND CONTOURS. SURFACES OF
REVOLUTION

4.1 3-D CURVES ON SPACE

4.2 3-D POLYGONS

4.3 3-D CURVES ON SPACE IN PARAMETRIC
COORDINATES

4.4 SURFACES IN EXPLICIT COORDINATES

4.5 SURFACE GRAPHICS: MESH GRAPHICS

4.6 SURFACE GRAPHICS: CONTOUR GRAPHICS

4.7 AXIS, VIEWS, SHADES, COLORS AND BRIGHT IN
THREE-DIMENSIONAL GRAPHICS

4.8 PARAMETRIC SURFACES AND REVOLUTION
SURFACES

4.9 SPHERES AND CILYNDERS

4.10 HANDLING GRAPHICS COMMANDS

MATLAB LANGUAGE ELEMENTS. ALGEBRAIC EXPRESSIONS, POLYNOMIALS, EQUATIONS AND SYSTEMS

5.1 EXPANDING, SIMPLIFYING AND FACTORING ALGEBRAIC EXPRESSIONS

5.2 POLYNOMIALS

5.3 POLYNOMIAL INTERPOLATION

5.4 SOLVING EQUATIONS AND SYSTEMS OF EQUATIONS

5.5 GENERAL METHODS

5.6 THE BICONJUGATE GRADIENT METHOD

5.7 THE CONJUGATE GRADIENTS METHOD

5.8 THE RESIDUAL METHOD

5.9 THE SYMMETRIC AND NON-NEGATIVE LEAST SQUARES METHODS

5.10 SOLVING LINEAR SYSTEMS OF EQUATIONS MATRICES, VECTOR SPACES, LINEAR APPLICATIONS AND QUADRATIC FORMS

6.1 MATRIX CALCULUS

6.2 MATRIX OPERATIONS

6.3 EIGENVECTORS AND EIGENVALUES. MATRIX

DECOMPOSITIONS

6.4 VECTOR SPACES, LINEAR APPLICATIONS AND

QUADRATIC FORMS

introduction

MATLAB can be used as a high level programming language including data structures, variables, functions, vectors, matrix, arrays, instructions for flow control, management of inputs/outputs and even object-oriented programming. MATLAB programs are often written into files called M-files. An M-file is nothing more than a MATLAB code (script) that executes a series of commands or functions that accept arguments and produce an output.

MATLAB is able to implement a number of algorithms which provide numerical solutions to certain problems which play a central role in the solution of non-linear equations. Such algorithms are easy to construct in MATLAB and are stored as M-files. Also MATLAB allows you to easily manipulate and operate on formulae and expressions symbolically. It is possible to expand, factor and simplify polynomials and rational and trigonometric expressions; find algebraic solutions of polynomial equations and systems of equations; evaluate derivatives and integrals symbolically; find symbolic solutions of differential equations; manipulate powers, limits and many other facets of algebraic series.

MATLAB is a platform for scientific computing that can work in almost all areas of the experimental sciences and engineering. Logically, this software allows you to work in the field of graphics and linear algebra, featuring some pretty extensive capabilities. The commands that implements Matlab, about working with graphics and matrix algebra are quite high and very efficient.

Matlab functions for working with two-dimensional and three-dimensional graphics, statistical graphs, curves and surfaces in explicit, implicit, parametric and polar coordinates. Additional work perfectly implements the twisted curves, surfaces, meshes, contours, volumes and graphical interpolation.

MATLAB allows you to work with ease in the field of Linear Algebra. Its matrix structure facilitates the treatment of matrix algebra, vector and matrix variables and functions, vector spaces, equations and systems, algebraic expressions, polynomials, linear applications, quadratic forms, diagonalization and other typical tasks of linear algebra.

Chapter 1.

INTRODUCTION TO MATLAB

The MATLAB language, based on matrices, is the most natural way to express computational mathematics. The integrated graphics facilitate the visualization of the data and the obtaining of information from them. The desktop environment invites you to experience, explore and discover. All of these MATLAB tools and functions are rigorously tested and designed to work together.

Mathematical functions offer a wide variety of calculation methods to analyze data, develop algorithms and create models. The main functions use libraries optimized for processors that allow quick calculations of vectors and matrices. You can use MATLAB as a powerful numerical computer. While most calculators handle numbers only to a preset degree of precision, MATLAB performs exact calculations to any desired degree of precision. In addition, unlike calculators, we can perform operations not only with individual numbers, but also with objects such as arrays.

Most of the topics of classical numerical analysis are treated by this software. It supports matrix calculus, statistics, interpolation, least squares fitting, numerical integration, minimization of functions, linear programming, numerical and algebraic solutions of differential equations and a long list of further methods that we'll meet as this book progresses.

To start MATLAB, simply double-click on the shortcut icon to the program on the Windows desktop. Alternatively, if there is no desktop shortcut, the easiest and most common way to run the program is to choose programs from the Windows Start menu and select MATLAB . Having launched MATLAB by either of these methods, the welcome screen briefly appears, followed by the screen depicted in Figure 1-1, which provides the general environment in which the program works.

The most important elements of the MATLAB screen are the following:

The Command Window: This runs MATLAB functions .

The Current Folder: This shows MATLAB files and execute files (such as opening and search for content operations).

The Workspace: This shows the present contents of the workspace and allows you to make changes to it.

The Menú Options: This shows the most important work options with the program (Figure 1-2).

Options palettes (HOME, PLOTS and APPS): This allows you to access to the main MATLAB options, graphic options and predefined applications.

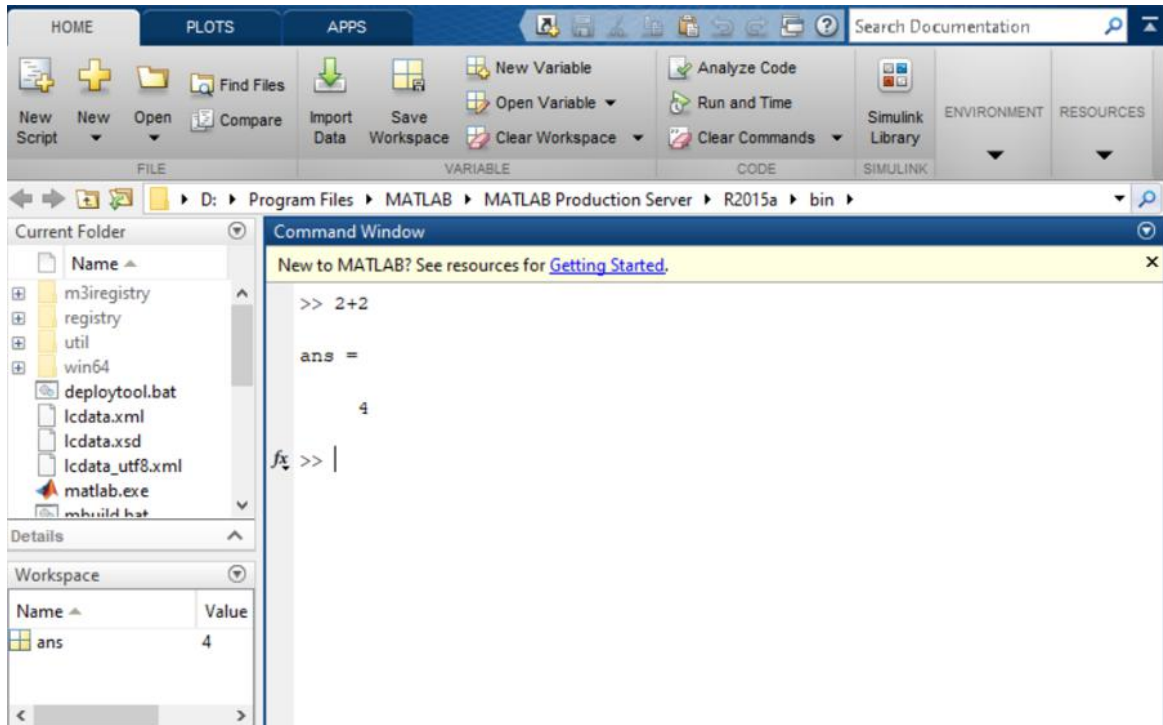


Figure 1-1

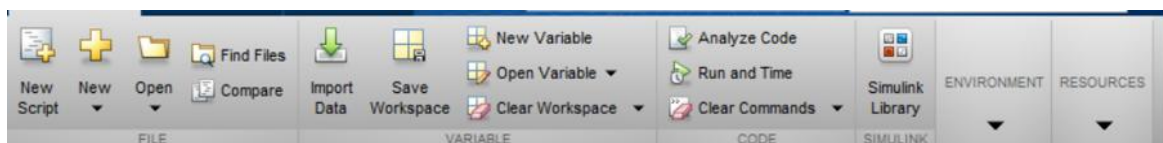


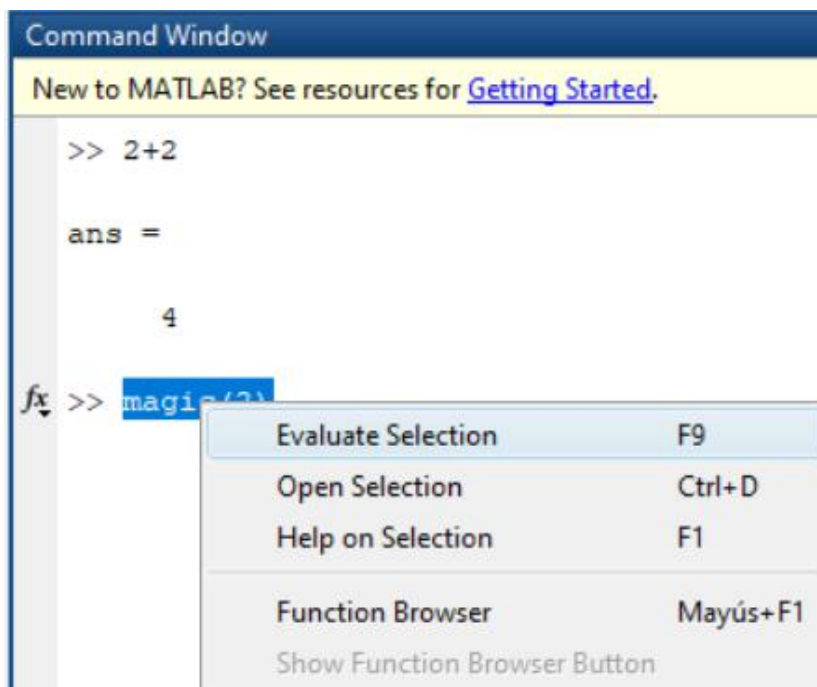
Figure 1-2

The Command Window(Figure 1-3) is the main way to communicate with MATLAB. It appears on the desktop when MATLAB starts and is used to execute all operations and functions. The entries are written below the prompt >> and, once completed, they run afterpressing Enter . The first

line of Figure 1-3 defines a matrix and, after pressing Enter, the results itself is displayed as output.

Figure 1-3

In the Command Window is possible to work with operations, functions, graphics and MATLAB code in general. Simply enter an expression with valid MATLAB syntax and pressing Enter is executed. Also it is possible to evaluate previously executed operations. To do this, simply select the syntax you wish to evaluate, right-click, and choose the option Evaluate Selection from the resulting pop-up menu (Figures 1-4 and 1-5). Choosing Open Selection from the same menu opens in the Editor/Debugger an M-file previously selected in the Command Window (Figures 1-6 and 1-7).



```

Command Window
New to MATLAB? See resources for Getting Started.

>> magic(2)

ans =

     1     3
     4     2

fx >>
<

```

Figure 1-4

Figure 1-5

```

Command Window
New to MATLAB? See resources for Getting Started.

>> magic(2)

ans =

     1
     4

fx >>
<

```

Evaluate Selection	F9
Open Selection	Ctrl+D
Help on Selection	F1
Function Browser	Mayús+F1
Show Function Browser Button	
Function Hints	Ctrl+F1

```

Editor - D:\Program Files\MATLAB\MATLAB Production Serv
magic.m x +
1 function M = magic(n)
2 %MAGIC Magic square.
3 % MAGIC(N) is an N-by-N matrix co
4 % 1 through N^2 with equal row, c
5 % Produces valid magic squares fc
6
7 % Copyright 1984-2011 The MathWor
<

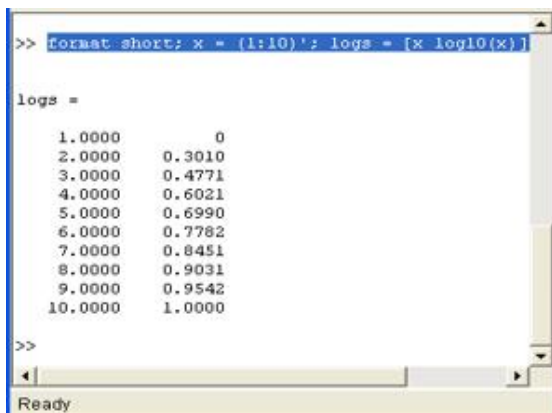
```

Figure 1-6

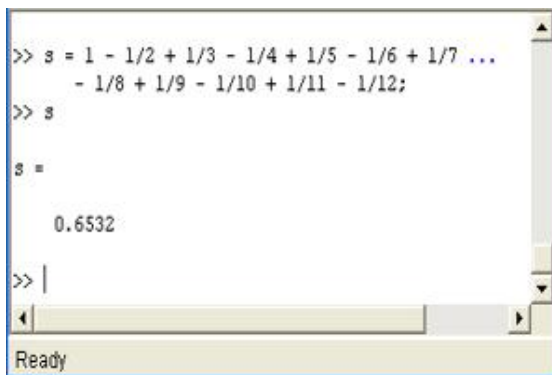
Figure 1-7

MATLAB is sensitive to the use of uppercase and lowercase characters, and blank spaces can be used before and after minus signs, colons and parentheses. MATLAB also allows you to write several commands on the same line, provided they are separated by semicolons (Figure 1-8). Entries are executed sequentially in the order they appear on the line. Every command which ends with a semicolon will run, but will not display its output.

Long entries that will not fit on one line can be continued onto a second line by placing dots at the end of the first line (Figure 1-9).



```
>> format short; x = (1:10)'; logs = [x log10(x)]  
  
logs =  
  
  1.0000    0  
  2.0000    0.3010  
  3.0000    0.4771  
  4.0000    0.6021  
  5.0000    0.6990  
  6.0000    0.7782  
  7.0000    0.8451  
  8.0000    0.9031  
  9.0000    0.9542  
 10.0000    1.0000  
  
>>
```



```
>> s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...  
      - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;  
>> s  
  
s =  
  
  0.6532  
  
>> |
```

Figure 1-8

Figure 1-9

The option Clear Command Window from the Edit menu (Figure 1-10) allows you to clear the Command Window. The command `clc` also performs this function (Figure 1-11).

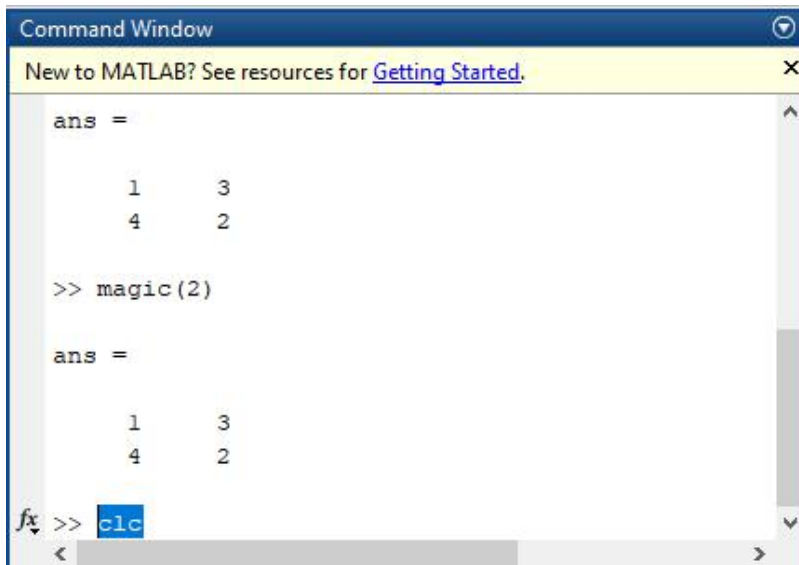
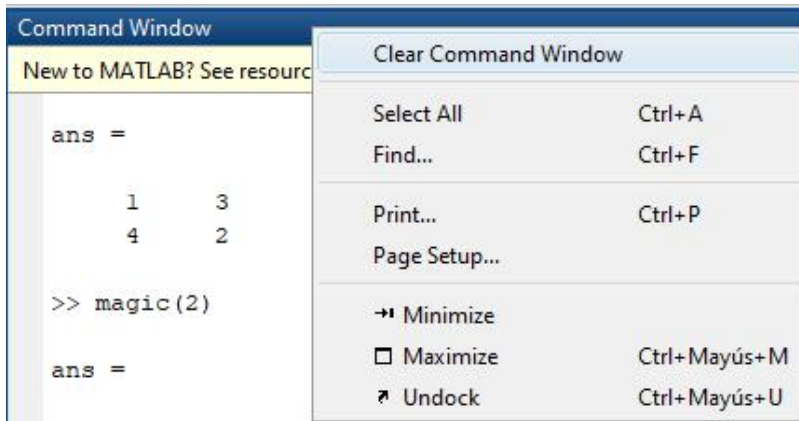


Figure 1-10

Figure 1-11

Below is a list of keys, arrows and combinations that can be used in the Command Window.

You can use MATLAB as a powerful numerical computer. While most calculators handle numbers only to a preset degree of precision, MATLAB performs exact calculations to any desired degree of precision. In addition, unlike calculators, we can perform operations not only with individual numbers, but also with objects such as arrays.

Most of the topics of classical numerical analysis are treated by this software. It supports matrix calculus, statistics, interpolation, least squares fitting, numerical integration, minimization of functions, linear

programming, numerical and algebraic solutions of differential equations and a long list of further methods that we'll meet as this book progresses.

Here are some examples of numerical calculations with MATLAB. (As we know, to obtain the results it is necessary to press Enter once the desired command has been entered after the prompt " » ".)

1. We simply calculate $4 + 3$ to obtain the result 7. To do this, just type $4 + 3$, and then Enter .

```
» 4 + 3
```

```
ans =
```

```
7
```

2. We find the value of 3 to the power of 100, without having previously set the precision. To do this we simply enter $3 ^ 100$.

```
» 3 ^ 100
```

```
ans =
```

```
5. 1538e + 047
```

3. We can use the command "format long e" to obtain results to 15 digits (floating-point).

```
» format long e
```

```
» 3^100
```

```
ans =
```

```
5.153775207320115e+047
```

4. We can also work with complex numbers. We find the result of the operation raising $(2 + 3i)$ to the power 10 by typing the expression $(2 + 3i) ^ 10$.

```
» (2 + 3i) ^ 10
```


ans =

-1.4152499999999998e + 005 - 1.4566800000000000e + 005i

5. The previous result is also available in short format, using the “format short” command.

» format short

» (2 + 3i)^10

ans =

-1.4152e+005- 1.4567e+005i

6. We can calculate the value of the Bessel function J_0 at 11.5. To do this we type `besselj(0,11.5)`.

`besselj(0,11.5)`

ans =

-0.0677

MATLAB perfectly handles symbolic mathematical computations, manipulating and performing operations on formulae and algebraic expressions with ease. You can expand, factor and simplify polynomials and rational and trigonometric expressions, find algebraic solutions of polynomial equations and systems of equations, evaluate derivatives and integrals symbolically, find solutions of differential equations, manipulate powers, and investigate limits and many other features of algebraic series.

To perform these tasks, MATLAB first requires all the variables (or algebraic expressions) to be written between single quotes. When MATLAB receives a variable or expression in quotes, it is interpreted as symbolic.

Here are some examples of symbolic computations with MATLAB.

1. We can expand the following algebraic expression: $((x + 1)(2.4x + 2)(2.4x + 2)^2)^3$. This is done by typing: `expand('((x + 1)(2.4 . x+2)(2.4.x+2) ^ 2) ^ 3')`. The result will be another algebraic expression:

```
» syms x; expand(((x + 1) *(x + 2)-(x + 2) ^ 2) ^ 3)
```

ans =

```
-x ^ 3-6 * x ^ 2-12 * x-8
```

2. We can factor the result of the calculation in the above example by typing: `factor('((x + 1) *(x + 2)-(x + 2) ^ 2) ^ 3')`

```
» syms x; factor(((x + 1)*(x + 2)-(x + 2)^2)^3)
```

ans =

```
-(x+2)^3
```

3. We can find the indefinite integral of the function $(x^2) \sin(x)^2$ by typing: `int(x ^ 2 * sin(x) ^ 2, x)`

```
syms x
```

```
int(x^2*sin(x)^2,x)
```

ans =

```
sin(2x)/8 - (xcos(2x))/4 - (x^2sin(2*x))/4 + x^3/6
```

4. We can simplify the previous result:

```
syms x; simplify(int(x^2*sin(x)^2, x))
```

ans =

```
sin(2x)/8 -(xcos(2x))/4 -(x^2sin(2*x))/4 + x^3/6
```

5. We can present the previous result using a more elegant mathematical notation:

```
syms x; pretty(simplify(int(x^2*sin(x)^2, x)))
```

ans =

2 3

$$\sin(2x) \cdot x \cos(2x) \cdot x \sin(2x) \cdot x$$

$$\frac{\sin(2x) \cdot x \cos(2x) \cdot x \sin(2x) \cdot x}{8 \cdot 4 \cdot 4 \cdot 6} + \dots$$

7. We can solve the equation $3ax - 7x^2 + x^3 = 0$ (where a is a parameter):

```
syms a x
```

```
solve(3ax-7*x^2 + x^3 == 0, x)
```

ans =

0

$7/2 - (49 - 12a)^{1/2}/2$

$(49 - 12a)^{1/2}/2 + 7/2$

On the other hand, MATLAB can use the Maple program libraries to work with symbolic math, and can thus extend its field of action. In this way, MATLAB can be used to work on such topics as differential forms, Euclidean geometry, projective geometry, statistics, etc.

At the same time, Maple can also benefit from MATLAB's powers of numerical calculation, which might be used, for example, in combination with the Maple libraries (combinatorics, optimization, number theory, etc.)

MATLAB can generate two- and three-dimensional graphs, as well as contour and density plots. You can graphically represent data lists, controlling colors, shading and other graphics features. Animated graphics are also supported. Graphics produced by MATLAB are portable to other programs.

Some examples of MATLAB graphics are given below.

1. We can represent the function $x \sin(1/x)$ for x ranging between $-1/4$ and $1/4$, taking 300 equidistant points in the interval. See Figure 1-13.

```
» x = linspace(-pi/4,pi/4,300);
```

```
» y=x.*sin(1./x);
```

```
» plot(x,y)
```

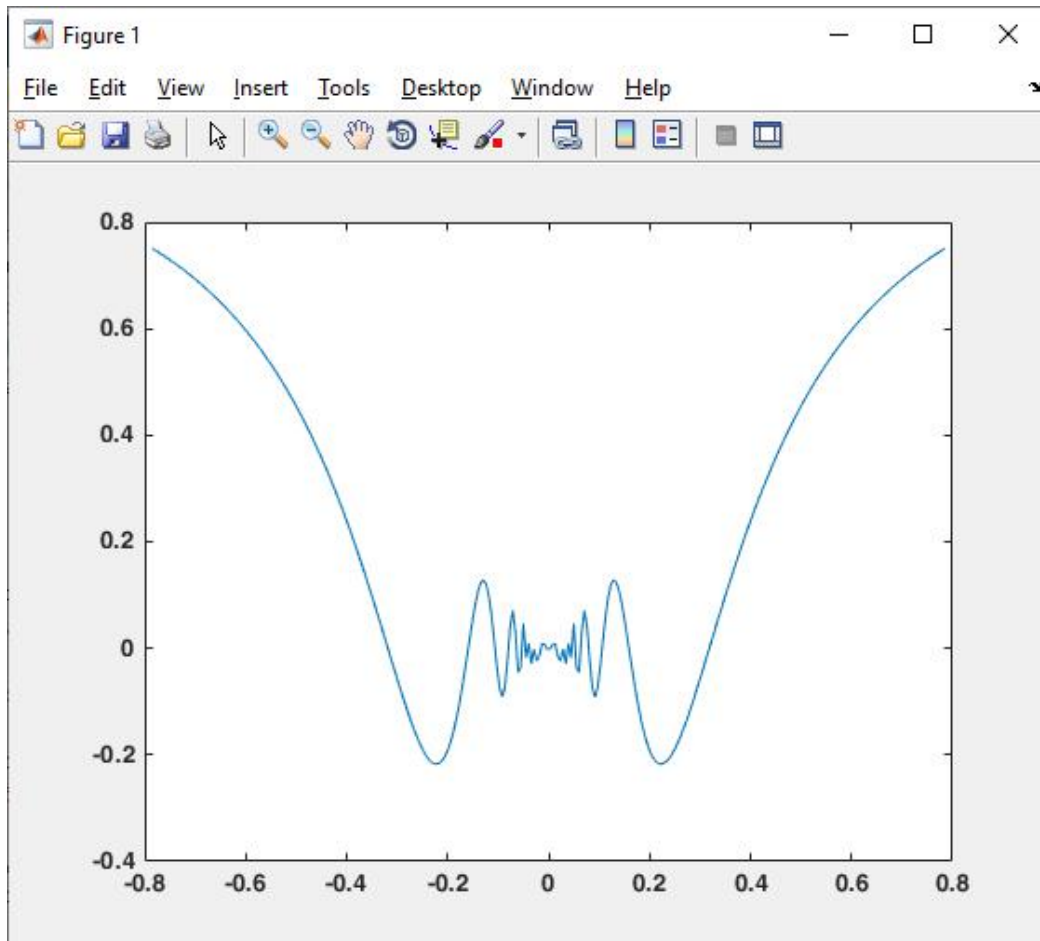


Figure 1-13

2. We can give the above graph a title and label the axes, and we can add a grid. See Figure 1-14.

```
» x = linspace(-pi/4,pi/4,300);
```

```
» y=x.*sin(1./x);
```

```
» plot(x,y);
```

```
» grid;
```

```
» xlabel('Independent variable X');
```

- » ylabel('Dependent variable Y');
- » title('The function y=xsin(1/x)')

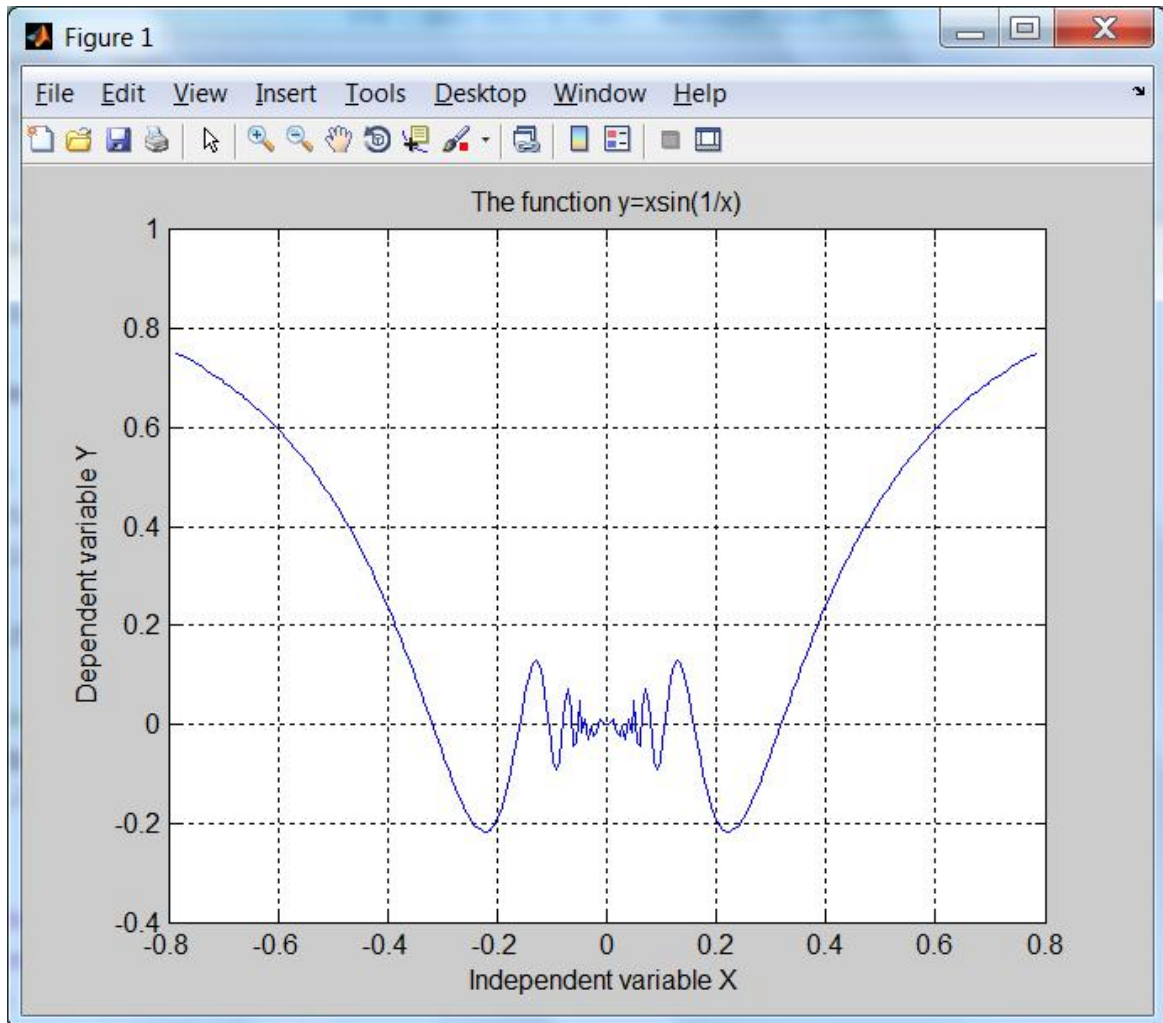


Figure 1-14

3. We can generate a graph of the surface defined by the function $z = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$, where x and y vary over the interval $(-7.5, 7.5)$, taking equally spaced points 0.5 apart. See Figure 1-15.

- » $x = -7.5:0.5:7.5;$
- » $y = x;$
- » $[X, Y] = \text{meshgrid}(x,y);$

```
» Z=sin(sqrt(X.^2+Y.^2))./sqrt(X.^2+Y.^2);
```

```
» surf(X, Y, Z)
```

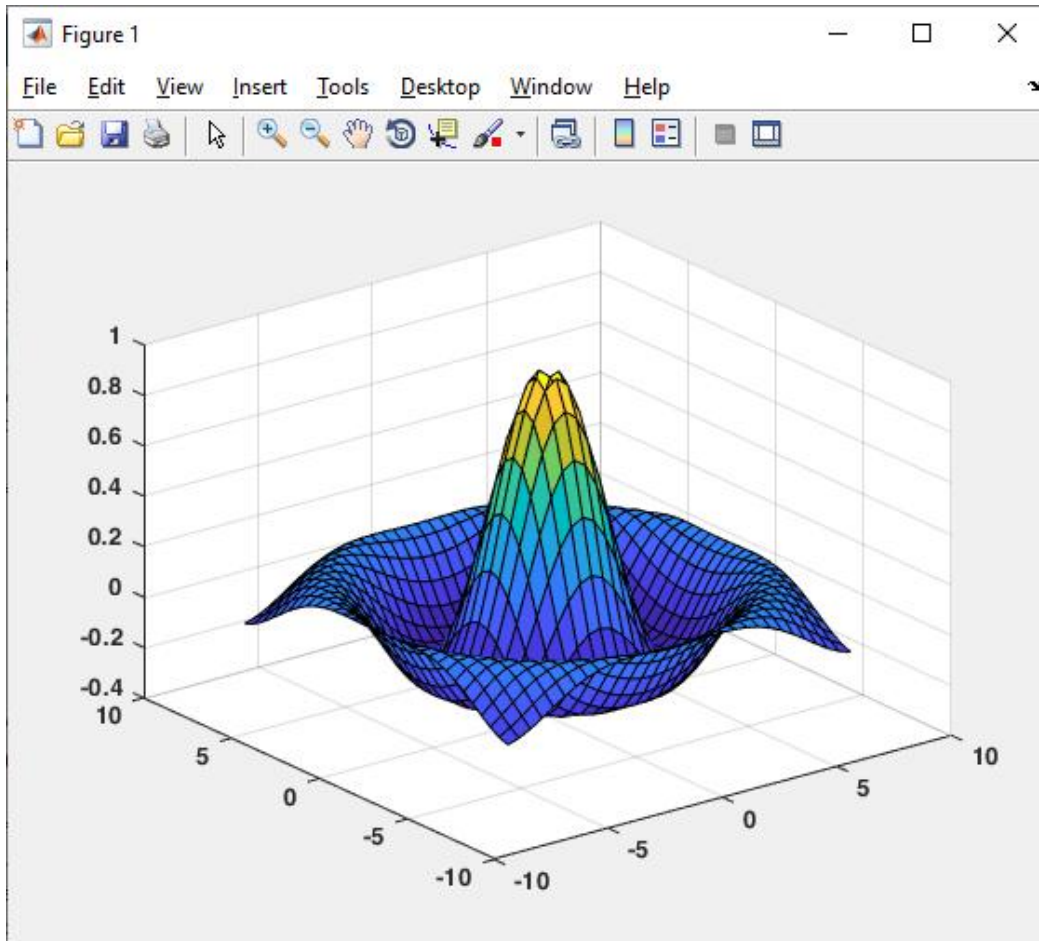


Figure 1-15

These 3D graphics allow you to get a clear picture of figures in space, and are very helpful in visually identifying intersections between different bodies, and in generating all kinds of space curves, surfaces and volumes of revolution.

4. We can generate the three dimensional graph corresponding to the helix with parametric coordinates: $x = \sin(t)$, $y = \cos(t)$, $z = t$. See Figure 1-16.

```
» t=0:pi/50:10*pi;
```

» plot3(sin(t),cos(t),t)

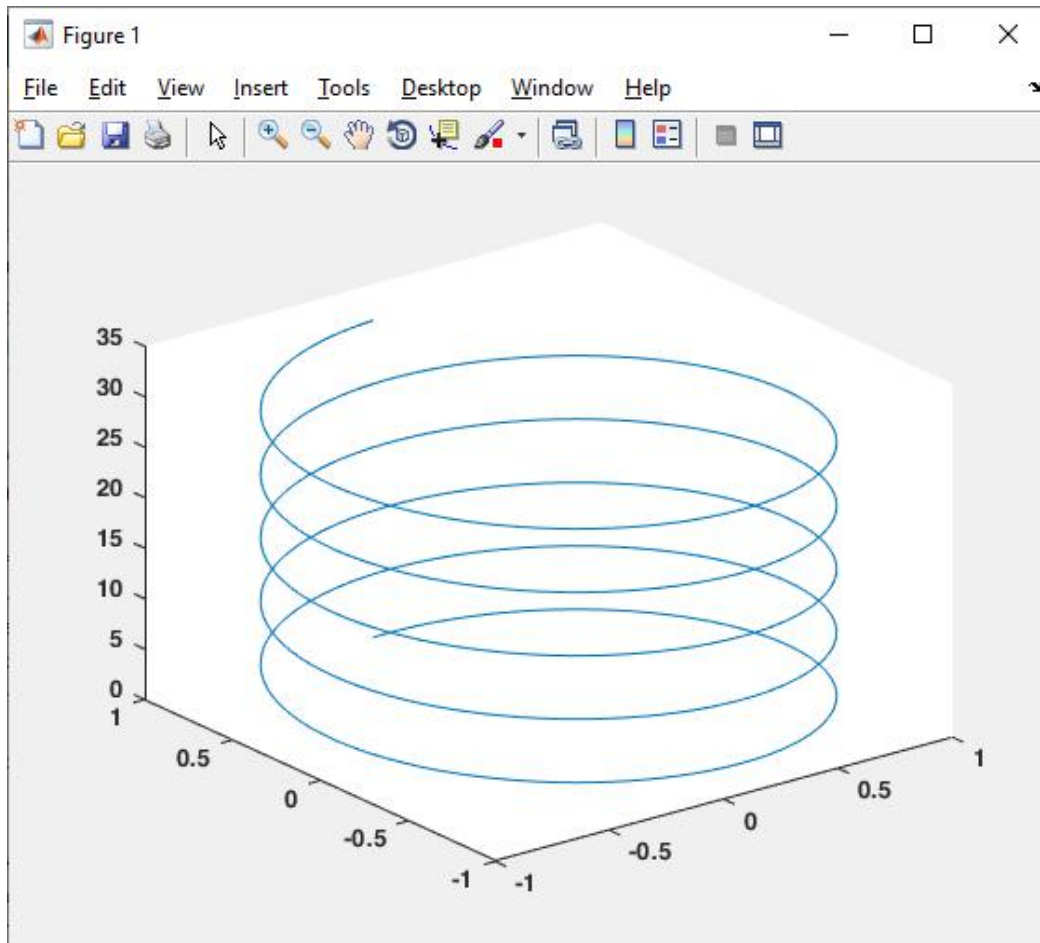


Figure 1-17

We can represent a planar curve given by its polar coordinates $r = \cos(2t) * \sin(2t)$ for t varying in the range between 0 and 2π by equally spaced points 0.01 apart. See Figure 1-17.

» $t = 0:0.01:2*\pi;$

» $r = \sin(2t).\cos(2*t);$

» $\text{polar}(t,r)$

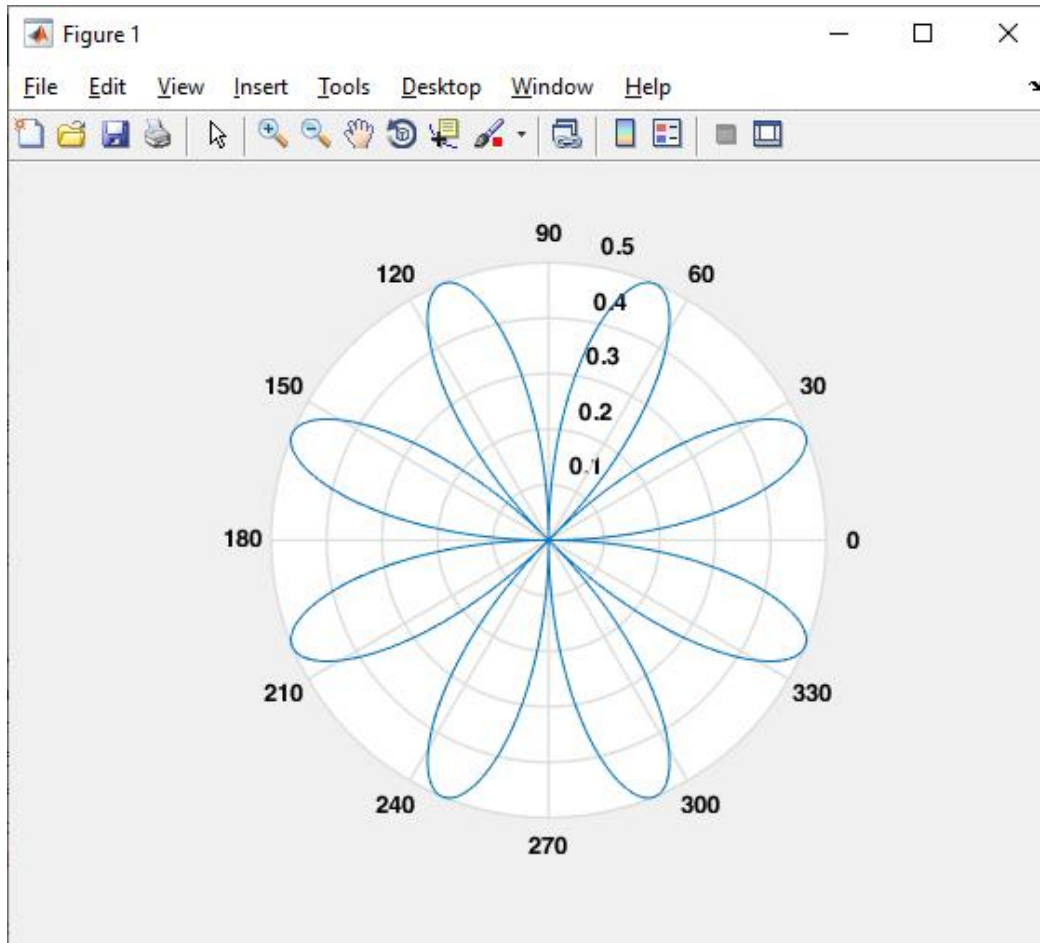


Figure 1-17

6. We can make a graph of a symbolic function using the command “ezplot”. See Figure 1-8.

```
» y = 'x ^ 3 / (x^2-1)';
```

```
» ezplot(y,[-5,5])
```

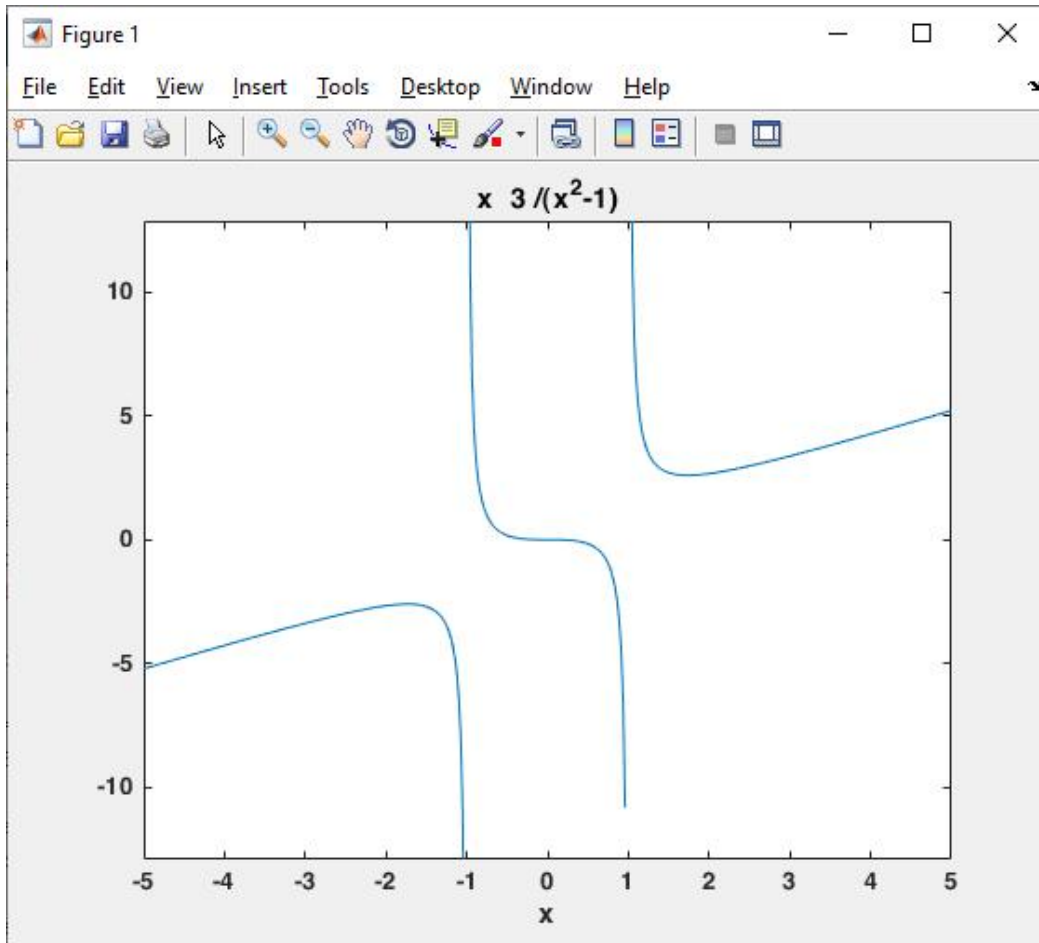



Figure 1-8

We will go into these concepts in more detail in the chapter on graphics.

We have already seen how you can get help using MATLAB's drop down menus.

But, in addition, support can also be obtained via commands (instructions or functions), implemented as MATLAB objects.

You can use the help command to get immediate access to diverse information.

» help

HELP topics:

matlab - General purpose commands.

matlab - Operators and special characters.

matlab - Programming language constructs.

matlab- Elementary matrices and matrix manipulation.

matlab- Elementary math functions.

matlab- Specialized math functions.

matlab- Matrix functions - numerical linear algebra.

matlab- Data analysis and Fourier transforms.

matlab- Interpolation and polynomials.

matlab- Function functions and ODE solvers.

matlab- Sparse matrices.

matlab 2d - Two dimensional graphs.

matlab 3d - Three dimensional graphs.

matlab- Specialized graphs.

matlab - Handle Graphics.

matlab- Graphical user interface tools.

matlab- Character strings.

matlab- File input/output.

matlab- Time and dates.

matlab - Data types and structures.

matlab - Windows Operating System Interface Files(DDE/ActiveX)

matlab - Examples and demonstrations.

toolbox - Symbolic Math Toolbox.

toolbox - MATLAB Tour

toolbox - Preferences .

For more help on directory/topic, type “help topic”.

As we can see, the help command displays a list of program directories and their contents. Help on any given topic can be displayed using the command help topic . For example:

» help inv

INV Matrix inverse.

INV(X) is the inverse of the square matrix X.

A warning message is printed if X is badly scaled or nearly singular.

See also SLASH, PINV, COND, CONDEST, NNLS, LSCOV.

Overloaded methods

help sym/inv.m

» help matlab

Elementary math functions.

Trigonometric.

sin - Sine.

sinh - Hyperbolic sine.

asin - Inverse sine.

asinh - Inverse hyperbolic sine.

cos - Cosine.

cosh - Hyperbolic cosine.

acos - Inverse cosine.
acosh - Inverse hyperbolic cosine.
tan - Tangent.
tanh - Hyperbolic tangent.
atan - Inverse tangent.
atan2 - Four quadrant inverse tangent.
atanh - Inverse hyperbolic tangent.
sec - Secant.
sech - Hyperbolic secant.
asec - Inverse secant.
asech - Inverse hyperbolic secant.
csc - Cosecant.
csch - Hyperbolic cosecant.
acsc - Inverse cosecant.
acsch - Inverse hyperbolic cosecant.
cot - Cotangent.
coth - Hyperbolic cotangent.
acot - Inverse cotangent.
acoth - Inverse hyperbolic cotangent.

Exponential.

exp - Exponential.
log - Natural logarithm.

log10 - Common(base 10) logarithm.
log2 - Base 2 logarithm and dissect floating point number.
pow2 - Base 2 power and scale floating point number.
sqrt - Square root.
nextpow2 - Next higher power of 2.

Complex.

abs - Absolute value.
angle - Phase angle.
conj - Complex conjugate.
imag - Complex imaginary part.
real - Complex real part.
unwrap - Unwrap phase angle.
isreal - True for real array.
cplxpair - Sort numbers into complex conjugate pairs.

Rounding and remainder.

fix - Round towards zero.
floor - Round towards minus infinity.
ceil - Round towards plus infinity.
round - Round towards nearest integer.
mod - Modulus(signed remainder after division).
rem - Remainder after division.
sign - Signum.

There is a command for help on a certain sequence of characters (lookfor string) which allows you to find all those functions or commands that contain or refer to the given string string . This command is very useful when there is no direct support for the specified string, or if you want to view the help for all commands related to the given sequence. For example, if we seek help for all commands that contain the sequence complex , we can use the lookfor complex command to see which commands MATLAB provides.

» lookfor complex

ctranspose.m : %' Complex conjugate transpose.

CONJ Complex conjugate.

CPLXPAIR Sort numbers into complex conjugate pairs.

IMAG Complex imaginary part.

REAL Complex real part.

CDF2RDF Complex diagonal form to real block diagonal form.

RSF2CSF Real block diagonal form to complex diagonal form.

B5 ODE Stiff problem, linear with complex eigenvalues(B5 of EHL).

CPLXDEMO Maps of functions of a complex variable.

CPLXGRID Polar coordinate complex grid.

CPLXMAP Plot a function of a complex variable.

GRAFCPLX Demonstrates complex function plots in MATLAB.

ctranspose.m: %TRANSPPOSE Symbolic matrix complex conjugate transpose.

SMOKE Complex matrix with a “smoke ring” pseudospectrum.

By properly combining all the objects defined in MATLAB, according to the rules of syntax of the program, you can build useful mathematical

programming code. Programs usually consist of a series of instructions in which values are calculated, are assigned names and are reused in further calculations.

As in programming languages like C or FORTRAN, in MATLAB you can write programs with loops, control flow and conditionals. MATLAB can write procedural programs, i.e., it can define a sequence of standard steps to run. As in C or Pascal, a Do, For, or While loop can be used for repetitive calculations. The language of MATLAB also includes conditional constructs such as If–Then–Else. MATLAB also supports different logical operators, such as AND, OR, NOT and XOR.

MATLAB supports procedural programming (with iterative processes, recursive functions, loops, etc.), functional programming and object-oriented programming. Here are two simple examples of programs. The first generates the Hilbert matrix of order n , and the second calculates all the Fibonacci numbers less than 1000.

```
% Generating the Hilbert matrix of order n
```

```
t = '1/(i+j-1)';
```

```
for i = 1:n
```

```
for j = 1:n
```

```
a(i,j) = eval(t);
```

```
end
```

```
end
```

```
% Calculating the Fibonacci numbers
```

```
f = [1 1]; i = 1;
```

```
while f(i) + f(i-1) < 1000
```

```
f(i+2) = f(i) + f(i+1);
```

```
i = i+1
```

end

Chapter 2.

MATLAB language elements. variables, numbers, operators and functions

MATLAB does not require a command to declare variables. A variable is created simply by directly allocating a value to it. For example:

```
v = 3
```

```
v =
```

```
3
```

The variable v will take the value 3 and using a new mapping will not change its value. Once the variable is declared, we can use it in calculations.

```
v^3
```

```
ans =
```

```
27
```

```
v+5
```

```
ans =
```

```
8
```

The value assigned to a variable remains fixed until it is explicitly changed or if the current MATLAB session is closed.

If we now write:

```
v = 3 + 7
```

```
v =
```


10

then the variable `v` has the value 10 from now on, as shown in the following calculation:

```
v^4
```

```
ans =
```

```
10000
```

A variable name must begin with a letter followed by any number of letters, digits or underscores. However, bear in mind that MATLAB uses only the first 31 characters of the name of the variable. It is also very important to note that MATLAB is case sensitive. Therefore, a variable named with uppercase letters is different to the variable with the same name except in lowercase letters.

A vector variable of n elements can be defined in MATLAB in the following ways:

```
V = [v1, v2, v3,..., vn]
```

```
V = [v1 v2 v3... vn]
```

When most MATLAB commands and functions are applied to a vector variable the result is understood to be that obtained by applying the command or function to each element of the vector:

```
vector1 = [1,4,9,2.25,1/4]
```

```
vector1 =
```

```
1.0000 4.0000 9.0000 2.2500 0.2500
```

```
sqrt(vector1)
```

```
ans =
```

```
1.0000 2.0000 3.0000 1.5000 0.5000
```

The following table presents some alternative ways of defining a vector variable without explicitly bracketing all its elements together, separated by commas or blank spaces.

Below are some examples:

```
vector2 = [5:5:25]
```

```
vector2 =
```

```
5 10 15 20 25
```

This yields the numbers between 5 and 25, inclusive, separated by 5 units.

```
vector3=[10:30]
```

```
vector3 =
```

```
Columns 1 through 13
```

```
10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
Columns 14 through 21
```

```
23 24 25 26 27 28 29 30
```

This yields the numbers between 10 and 30, inclusive, separated by a unit.

```
t:Microsoft.WindowsMobile.DirectX.Vector4 = linspace (10,30,6 )
```

```
t:Microsoft.WindowsMobile.DirectX.Vector4 =
```

```
10 14 18 22 26 30
```

This yields 6 equally spaced numbers between 10 and 30, inclusive.

```
vector5 = logspace(10,30,6)
```

```
vector5 =
```

1. 0e + 030 *

0.0000 0.0000 0.0000 0.0000 0.0001 1.0000

This yields 6 evenly logarithmically spaced numbers between 10^{10} and 10^{30} , inclusive.

One can also consider row vectors and column vectors in MATLAB. A column vector is obtained by separating its elements by semicolons, or by transposing a row vector using a single quotation mark at the end of its definition.

```
a=[10;20;30;40]
```

a =

10

20

30

40

```
a=(10:14);b=a'
```

b =

10

11

12

13

14

```
c=(a)'
```

c =

10 11 12 13 14

You can also select an element of a vector or a subset of elements. The rules are summarized in the following table:

Here are some examples:

```
x=(1:10)
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
x(6)
```

```
ans =
```

```
6
```

This yields the sixth element of the vector x.

```
x(4:7)
```

```
ans =
```

```
4 5 6 7
```

This yields the elements of the vector x located between the fourth and seventh elements, inclusive.

```
x(2:3:9)
```

```
ans =
```

```
2 5 8
```

This yields the three elements of the vector x located between the second and ninth elements, inclusive, but separated in steps of three units.

```
x(9:-3:2)
```

```
ans =
```

9 6 3

This yields the three elements of the vector x located between the ninth and second elements, inclusive, but separated in steps of three units and starting at the ninth.

MATLAB defines arrays by inserting in brackets all its row vectors separated by a semicolon . Vectors can be entered by separating their components by spaces or by commas, as we already know. For example, a 3×3 matrix variable can be entered in the following two ways:

$M = [a_{11} \ a_{12} \ a_{13} ; a_{21} \ a_{22} \ a_{23} ; a_{31} \ a_{32} \ a_{33}]$

$M = [a_{11} , a_{12} , a_{13} ; a_{21} , a_{22} , a_{23} ; a_{31} , a_{32} , a_{33}]$

Similarly we can define an array of variable dimension ($M \times N$) . Once a matrix variable has been defined, MATLAB enables many ways to insert, extract, renumber, and generally manipulate its elements. The following table shows different ways to define matrix variables.

Here are some examples:

We consider first the 2×3 matrix whose rows are the first six consecutive odd numbers:

$A = [1 \ 3 \ 5; 7 \ 9 \ 11]$

$A =$

1 3 5

7 9 11

Now we are going to change the (2,3)-th element, i.e. the last element of A , to zero:

$A(2,3) = 0$

$A =$

1 3 5

7 9 0

We now define the matrix B to be the transpose of A :

$$B = A'$$

B =

1 7

3 9

5 0

We now construct a matrix C , formed by attaching the identity matrix of order 3 to the right of the matrix B :

$$C = [B \text{ eye}(3)]$$

C =

1 7 1 0 0

3 9 0 1 0

5 0 0 0 1

We are going to build a matrix D by extracting the odd columns of the matrix C , a matrix E formed by taking the intersection of the first two rows of C and its third and fifth columns, and a matrix F formed by taking the intersection of the first two rows and the last three columns of the matrix C :

$$D = C(:,1:2:5)$$

D =

1 1 0

3 0 0

5 0 1

$$E = C([1\ 2],[3\ 5])$$

E =

1 0

0 0

$$F = C([1\ 2],3:5)$$

F =

1 0 0

0 1 0

Now we build the diagonal matrix G such that the elements of the main diagonal are the same as those of the main diagonal of D :

$$G = \text{diag}(\text{diag}(D))$$

G =

1 0 0

0 0 0

0 0 1

We then build the matrix H , formed by taking the intersection of the first and third rows of C and its second, third and fifth columns:

$$H = C([1\ 3],[2\ 3\ 5])$$

H =

7 1 0

0 0 1

Now we build an array I formed by the identity matrix of order 5×4 , appending the zero matrix of the same order to its right and to the right of

that the unit matrix, again of the same order. Then we extract the first row of I and, finally, form the matrix J comprising the odd rows and even columns of I and calculate its order (size).

```
I = [eye(5,4) zeros(5,4) ones(5,4)]
```

ans =

```
1 0 0 0 0 0 0 0 0 1 1 1 1
0 1 0 0 0 0 0 0 0 1 1 1 1
0 0 1 0 0 0 0 0 0 1 1 1 1
0 0 0 1 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1
```

```
I(1,:)
```

ans =

```
1 0 0 0 0 0 0 0 0 1 1 1 1
```

```
J=I(1:2:5,2:2:12)
```

J =

```
0 0 0 0 1 1
0 0 0 0 1 1
0 0 0 0 1 1
```

```
size(J)
```

ans =

```
3 6
```

We now construct a random matrix K of order 3 ×4 , reverse the order of the rows of K , reverse the order of the columns of K and then perform

both operations simultaneously. Finally, we find the matrix L of order 4×3 whose columns are obtained by taking the elements of K sequentially by columns.

```
K=rand(3,4)
```

```
K =
```

```
0.5269  0.4160  0.7622  0.7361
```

```
0.0920  0.7012  0.2625  0.3282
```

```
0.6539  0.9103  0.0475  0.6326
```

```
K(3:-1:1,:)
```

```
ans =
```

```
0.6539  0.9103  0.0475  0.6326
```

```
0.0920  0.7012  0.2625  0.3282
```

```
0.5269  0.4160  0.7622  0.7361
```

```
K(:,4:-1:1)
```

```
ans =
```

```
0.7361  0.7622  0.4160  0.5269
```

```
0.3282  0.2625  0.7012  0.0920
```

```
0.6326  0.0475  0.9103  0.6539
```

```
K(3:-1:1,4:-1:1)
```

```
ans =
```

```
0.6326  0.0475  0.9103  0.6539
```

```
0.3282  0.2625  0.7012  0.0920
```

```
0.7361  0.7622  0.4160  0.5269
```

```
L=reshape(K,4,3)
```

```
L =
```

```
0.5269 0.7012 0.0475
```

```
0.0920 0.9103 0.7361
```

```
0.6539 0.7622 0.3282
```

```
0.4160 0.2625 0.6326
```

A character variable (chain) is simply a character string enclosed in single quotes that MATLAB treats as a vector form. The general syntax for character variables is as follows:

```
c = 'string'
```

Among the MATLAB commands that handle character variables we have the following:

Here are some examples:

```
hex2dec('3ffe56e')
```

```
ans =
```

```
67102062
```

Here MATLAB has converted a hexadecimal string into a decimal number.

```
dec2hex(1345679001)
```

```
ans =
```

```
50356E99
```

The program has converted a decimal number into a hexadecimal string.

```
sprintf('%f',[1+sqrt(5)/2,pi])
```

```
ans =
```

2.118034 3.141593

The exact numerical components of a vector have been converted to strings (with default precision).

```
sscanf('121.00012', '%f')
```

ans =

121.0001

Here a numeric string has been passed to an exact numerical format (with default precision).

```
num2str(pi)
```

ans =

3.142

The constant has been converted into a string.

```
str2num('15/14')
```

ans =

1.0714

The string has been converted into a numeric value with default precision.

```
setstr(32:126)
```

ans =

```
! "# $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?  
@ 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 [ \ ] ^  
_ ` 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 { } ~
```

This yields the ASCII characters associated with the whole numbers between 32 and 126, inclusive.

```
abs('{}><#i¿?oa')
```

ans =

123 93 125 62 60 35 161 191 63 186 170

This yields the integers corresponding to the ASCII characters specified in the argument of `abs` .

```
lower('ABCDefgHIJ')
```

ans =

abcdefghijkl

The text has been converted to lowercase.

```
upper('abcd eFGHi jKIMn')
```

ans =

ABCD EFGHI JKLMN

The text has been converted to uppercase.

```
str2mat('The world', 'The country', 'Daily 16', 'ABC')
```

ans =

The world

The country

Daily 16

ABC

The chains comprising the arguments of `str2mat` have been converted to a text array.

```
disp('This text will appear on the screen')
```

ans =

This text will appear on the screen

Here the argument of the command disp has been displayed on the screen.

```
c = 'This is a good example';
```

```
strrep(c, 'good', 'bad')
```

```
ans =
```

```
This is a bad example
```

The string good has been replaced by bad in the chain c. The following instruction locates the initial position of each occurrence of is within the chain c .

```
findstr(c, 'is')
```

```
ans =
```

```
3 6
```

In MATLAB the arguments of a function can take many different forms, including different types of numbers and numerical expressions, such as integers and rational, real and complex numbers.

Arithmetic operations in MATLAB are defined according to the standard mathematical conventions. MATLAB is an interactive program that allows you to perform a simple variety of mathematical operations. MATLAB assumes the usual operations of sum, difference, product, division and power, with the usual hierarchy between them:

To add two numbers simply enter the first number, a plus sign (+) and the second number. Spaces may be included before and after the sign to ensure that the input is easier to read.

```
2 + 3
```

```
ans =
```

```
5
```

We can perform power calculations directly.

100^50

ans =

1. 0000e + 100

Unlike a calculator, when working with integers, MATLAB displays the full result even when there are more digits than would normally fit across the screen. For example, MATLAB returns the following value of 99^{50} when using the vpa function (here to the default accuracy of 32 significant figures).

vpa '99^50'

ans =

. 60500606713753665044791996801256e100

To combine several operations in the same instruction one must take into account the usual priority criteria among them, which determine the order of evaluation of the expression. Consider, for example:

$23^2+(5-2)3$

ans =

27

Taking into account the priority of operators, the first expression to be evaluated is the power 3^2 . The usual evaluation order can be altered by grouping expressions together in parentheses.

In addition to these arithmetic operators, MATLAB is equipped with a set of basic functions and you can also define your own functions. MATLAB functions and operators can be applied to symbolic constants or numbers.

One of the basic applications of MATLAB is its use in realizing arithmetic operations as if it were a conventional calculator, but with one important difference: the precision of the calculation. Operations are performed to whatever degree of precision the user desires. This unlimited precision in calculation is a feature which sets MATLAB apart from other numerical

calculation programs, where the accuracy is determined by a word length inherent to the software, and cannot be modified.

The accuracy of the output of MATLAB operations can be relaxed using special approximation techniques which are exact only up to a certain specified degree of precision. MATLAB represents results with accuracy, but even if internally you are always working with exact calculations to prevent propagation of rounding errors, different approximate representation formats can be enabled, which sometimes facilitate the interpretation of the results. The commands that allow numerical approximation are the following:

Using `format` gives a numerical approximation of $174/13$ in the way specified after the `format` command:

```
174/13
```

```
ans =
```

```
13.3846
```

```
format long; 174/13
```

```
ans =
```

```
13.38461538461539
```

```
format long e; 174/13
```

```
ans =
```

```
1.338461538461539e + 001
```

```
format short e; 174/13
```

```
ans =
```

```
1.3385e + 001
```

```
format long g; 174/13
```

```
ans =
```

```
13.3846153846154
```

```
format short g; 174/13
```

```
ans =
```

```
13.385
```

```
format bank; 174/13
```

```
ans =
```

```
13.38
```

```
format hex; 174/13
```

```
ans =
```

```
402ac4ec4ec4ec4f
```

Now we will see how the value of $\sqrt{17}$ can be calculated to any precision that we desire:

```
vpa '174/13' 10
```

```
ans =
```

```
13.38461538
```

```
vpa '174/13' 15
```

```
ans =
```

```
13.3846153846154
```

```
digits(20); vpa '174/13'
```

```
ans =
```

```
13.384615384615384615
```

In MATLAB all common operations with whole numbers are exact, regardless of the size of the output. If we want the result of an operation

to appear on screen to a certain number of significant figures, we use the symbolic computation command `vpa` (variable precision arithmetic), whose syntax we already know.

For example, the following calculates 6^{400} to 450 significant figures:

```
vpa '6^400' 450
```

```
ans =
```

```
0.
```

The result of the operation is precise, always displaying a point at the end of the result. In this case it turns out that the answer has fewer than 450 digits anyway, so the solution is exact. If you require a smaller number of significant figures, that number can be specified and the result will be rounded accordingly. For example, calculating the above power to only 50 significant figures we have:

```
vpa '6^400' 50
```

```
ans =
```

```
. 18217977168218728251394687124089371267338971528175e312
```

There are several functions in MATLAB with integer arguments, the majority of which are related to divisibility. Among the most typical functions with integer arguments are the following:

Below are some examples.

The remainder of division of 17 by 3:

```
rem (17,3)
```

```
ans =
```

```
2
```

The remainder of division of 4.1 by 1.2:

```
rem(4.1,1.2)
```

ans =

0.5000

The remainder of division of -4.1 by 1.2:

```
rem(-4.1,1.2)
```

ans =

-0.5000

The greatest common divisor of 1000, 500 and 625:

```
gcd(1000, gcd(500,625))
```

ans =

125.00

The least common multiple of 1000, 500 and 625:

```
lcm(1000,lcm(500,625))
```

ans =

5000.00

MATLAB allows you to work with numbers to any base, as long as the extended symbolic math Toolbox is available. It also allows you to express all kinds of numbers in different bases. This is implemented via the following functions:

Below are some examples.

Represent in base 10 the base 2 number 100101.

```
base2dec('100101',2)
```

ans =

37.00

Represent in base 10 the hexadecimal number FFFFAA00.

```
base2dec('FFFAA0', 16)
```

ans =

```
268434080.00
```

Represent the result of the base 16 operation FFFAA2+FF-1 in base 10.

```
base2dec('FFFAA2', 16) + base2dec('FF', 16) - 1
```

ans =

```
16776096.00
```

As is well known, the set of real numbers is the disjoint union of the set of rational numbers and the set of irrational numbers. A rational number is a number of the form p/q , where p and q are integers. In other words, the rational numbers are those numbers that can be represented as a quotient of two integers. The way in which MATLAB treats rational numbers differs from the majority of calculators. If we ask a calculator to calculate the sum $1/2 + 1/3 + 1/4$, most will return something like 1.0833, which is no more than an approximation of the result.

The rational numbers are ratios of integers, and MATLAB can work with them in exact mode, so the result of an arithmetic expression involving rational numbers is always given precisely as a ratio of two integers. To enable this, activate the rational format with the command `format rat`. If the reader so wishes, MATLAB can also return the results in decimal form by activating any other type of format instead (e.g. `format short` or `format long`). MATLAB evaluates the above mentioned sum in exact mode as follows:

```
format rat
```

```
1/2 + 1/3 + 1/4
```

ans =

```
13/12
```

Unlike calculators, MATLAB ensures its operations with rational numbers are accurate by maintaining the rational numbers in the form of ratios of integers. In this way, calculations with fractions are not affected by rounding errors, which can become very serious, as evidenced by the theory of errors. Note that, once the rational format is enabled, when MATLAB adds two rational numbers the result is returned in symbolic form as a ratio of integers, and operations with rational numbers will continue to be exact until an alternative format is invoked.

A floating point number, or a number with a decimal point, is interpreted as exact if the rational format is enabled. Thus a floating point expression will be interpreted as an exact rational expression while any irrational numbers in a rational expression will be represented by an appropriate rational approximation.

```
format rat
```

```
10/23 + 2.45/44
```

```
ans =
```

```
1183 / 2412
```

The other fundamental subset of the real numbers is the set of irrational numbers, which have always created difficulties in numerical calculation due to their special nature. The impossibility of representing an irrational number accurately in numeric mode (using the ten digits from the decimal numbering system) is the cause of most of the problems. MATLAB represents the results with an accuracy which can be set as required by the user. An irrational number, by definition, cannot be represented exactly as the ratio of two integers. If ordered to calculate the square root of 17, by default MATLAB returns the number 5.1962.

```
sqrt(27)
```

```
ans =
```

```
5.1962
```

MATLAB incorporates the following common irrational constants and notions:

The following examples illustrate how MATLAB outputs these numbers and notions.

```
long format
```

```
pi
```

```
ans =
```

```
3.14159265358979
```

```
exp(1)
```

```
ans =
```

```
2.71828182845905
```

```
1/0
```

```
Warning: Divide by zero.
```

```
ans =
```

```
Inf
```

```
0/0
```

```
Warning: Divide by zero.
```

```
ans =
```

```
NaN
```

```
realmin
```

```
ans =
```

```
2.225073858507201e-308
```

```
realmax
```

```
ans =
```

1. 797693134862316e + 308

The disjoint union of the set of rational numbers and the set of irrational numbers is the set of real numbers. In turn, the set of rational numbers has the set of integers as a subset. All functions applicable to real numbers are also valid for integers and rational numbers. MATLAB provides a full range of predefined functions, most of which are discussed in the subsequent chapters of this book. Within the group of functions with real arguments offered by MATLAB, the following are the most important:

Trigonometric functions

Hyperbolic functions

Exponential and logarithmic functions

Numeric variable-specific functions

Here are some examples:

```
sin(pi/2)
```

```
ans =
```

```
1
```

```
asin(1)
```

```
ans =
```

```
1.57079632679490
```

```
log(exp(1)^3)
```

```
ans =
```

```
3.000000000000000
```

The function `round` is demonstrated in the following two examples:

```
round(2.574)
```

```
ans =
```

```
3
```

```
round(2.4)
```

```
ans =
```

```
2
```

The function `ceil` is demonstrated in the following two examples:

```
ceil(4.2)
```

```
ans =
```

```
5
```

```
ceil(4.8)
```

```
ans =
```

```
5
```

The function `floor` is demonstrated in the following two examples:

```
floor(4.2)
```

```
ans =
```

```
4
```

```
floor(4.8)
```

```
ans =
```

```
4
```

The `fix` function simply removes the fractional part of a real number:

```
» fix(5.789)
```

```
ans =
```

5

Operations on complex numbers are well implemented in MATLAB. MATLAB follows the convention that i or j represents the key value in complex analysis, the imaginary number $\sqrt{-1}$. All the usual arithmetic operators can be applied to complex numbers, and there are also some specific functions which have complex arguments. Both the real and the imaginary part of a complex number can be a real number or a symbolic constant, and operations with them are always performed in exact mode, unless otherwise instructed or necessary, in which case an approximation of the result is returned. As the imaginary unit is represented by the symbol i or j , the complex numbers are expressed in the form $a+bi$ or $a+bj$. Note that you don't need to use the product symbol (asterisk) before the imaginary unit:

$$(1-5i)*(1-i)/(-1+2i)$$

ans =

$$-1.6000 + 2.8000i$$

format rat

$$(1-5i)*(1-i)/(-1+2i)$$

ans =

$$-8/5 + 14/5i$$

Working with complex variables is very important in mathematical analysis and its many applications in engineering. MATLAB implements not only the usual arithmetic operations with complex numbers, but also various complex functions. The most important functions are listed below.

Trigonometric functions

Hyperbolic functions

Exponential and logarithmic functions

Specific functions for the real and imaginary part

Specific functions for complex numbers

Below are some examples of operations with complex numbers.

```
round(1.5-3.4i)
```

```
ans =
```

```
2 - 3i
```

```
real(i^i)
```

```
ans =
```

```
0.2079
```

```
(2+2i)^(2/(-3-3sqrt(3)i))90
```

```
ans =
```

```
0502e-085 - 1 + 7.4042e-070i
```

```
sin(1 + i)
```

```
ans =
```

```
1.2985 + 0.6350i
```

MATLAB easily handles vector and matrix calculus. Indeed, its name, MATrix LABoratory, already gives an idea of its power in working with vectors and matrices. MATLAB allows you to work with functions of a complex variable, but in addition this variable can even be a vector or a matrix. Below is a table of functions with complex vector arguments.

These functions also support a complex matrix as an argument, in which case the result is a vector of column vectors whose components are the results of applying the function to each column of the matrix.

Here are some examples:

```
V = 2:7, W = [5 + 3i 2-i 4i]
```

V =

2 3 4 5 6 7

W =

2.0000 - 1.0000i 0 + 4.0000i 5.0000 + 3.0000i

diff(V),diff(W)

ans =

1 1 1 1 1

ans =

-2.0000 + 5.0000i 5.0000 - 1.0000i

cumprod(V),cumsum(V)

ans =

2 6 24 120 720 5040

ans =

2 5 9 14 20 27

cumsum(W), mean(W), std(W), sort(W), sum(W)

ans =

2.0000 - 1.0000i 2.0000 + 3.0000i 7.0000 + 6.0000i

ans =

2.3333 + 2.0000i

ans =

3.6515

ans =

2.0000 - 1.0000i 0 + 4.0000i 5.0000 + 3.0000i

ans =

7.0000 + 6.0000i

mean(V), std(V), sort(V), sum(V)

ans =

4.5000

ans =

1.8708

ans =

2 3 4 5 6 7

ans =

27

fft(W), ifft(W), fft2(W)

ans =

7.0000 + 6.0000i 0.3660 - 0.1699i - 1.3660 - 8.8301i

ans =

2.3333 + 2.0000i - 0.4553 - 2.9434i 0.1220 - 0.0566i

ans =

7.0000 + 6.0000i 0.3660 - 0.1699i - 1.3660 - 8.8301i

Here are some examples :

A=[7 8 9; 1 2 3; 4 5 6], B=[1+2i 3+i;4+i,i]

A =

7 8 9

1 2 3

4 5 6

B =

1.0000 + 2.0000i 3.0000 + 1.0000i

4.0000 + 1.0000i 0 + 1.0000i

sin(A), sin(B), exp(A), exp(B), log(B), sqrt(B)

ans =

0.6570 0.9894 0.4121

0.8415 0.9093 0.1411

-0.7568 -0.9589 -0.2794

ans =

3.1658 + 1.9596i 0.2178 - 1.1634i

-1.1678 - 0.7682i 0 + 1.1752i

ans =

1.0e+003 *

1.0966 2.9810 8.1031

0.0027 0.0074 0.0201

0.0546 0.1484 0.4034

ans =

-1.1312 + 2.4717i 10.8523 + 16.9014i

29.4995 + 45.9428i 0.5403 + 0.8415i

ans =

0.8047 + 1.1071i 1.1513 + 0.3218i

1.4166 + 0.2450i 0 + 1.5708i

ans =

1.2720 + 0.7862i 1.7553 + 0.2848i

2.0153 + 0.2481i 0.7071 + 0.7071i

The exponential functions, square root and logarithm used above apply to the array elementwise and have nothing to do with the matrix exponential and logarithmic functions that are used below.

expm(B), logm(A), abs(B), imag(B)

ans =

-27.9191 +14.8698 i - 20.0011 +12.0638i

-24.7950 + 17.6831i -17.5059 + 14.0445i

ans =

11.9650 12.8038 -19.9093

-21.7328 -22.1157 44.6052

11.8921 12.1200 -21.2040

ans =

2.2361 3.1623

4.1231 1.0000

ans =

2 1

1 1

```
fix(sin(B)), ceil(log(A)), sign(B), rem(A,3*ones(3))
```

```
ans =
```

```
3.0000 + 1.0000i    0 - 1.0000i
```

```
-1.0000           0 + 1.0000i
```

```
ans =
```

```
2 3 3
```

```
0 1 2
```

```
2 2 2
```

```
ans =
```

```
0.4472 + 0.8944i 0.9487 + 0.3162i
```

```
0.9701 + 0.2425i    0 + 1.0000i
```

```
ans =
```

```
1 2 0
```

```
1 2 0
```

```
1 2 0
```

MATLAB can easily generate (pseudo) random numbers. The function `rand` generates uniformly distributed random numbers and the function `randn` generates normally distributed random numbers. The most interesting features of MATLAB's random number generator are presented in the following table.

Here are some examples:

```
[rand, rand(1), randn, randn(1)]
```

```
ans =
```

```
0.9501  0.2311 -0.4326 -1.6656
```

```
[rand(2), randn(2)]
```

```
ans =
```

```
0.6068  0.8913          0.1253 -1.1465
```

```
0.4860  0.7621          0.2877  1.1909
```

```
[rand(2,3), randn(2,3)]
```

```
ans =
```

```
0.3529 0.0099 0.2028 -0.1364 1.0668 -0.0956
```

```
0.8132 0.1389 0.1987  0.1139 0.0593 -0.8323
```

MATLAB features arithmetic, logical, relational, conditional and structural operators.

There are two types of arithmetic operators in MATLAB: matrix arithmetic operators, which are governed by the rules of linear algebra, and arithmetic operators on vectors, which are performed elementwise. The operators involved are presented in the following table.

Simple mathematical operations between scalars and vectors apply the scalar to all elements of the vector according to the defined operation, and simple operators between vectors are performed element by element. Below is the specification of these operators:

It must be borne in mind that the vectors must be of the same length and that in the product, quotient and power the first operand must be followed by a point.

The following example involves all of the above operators.

```
X = [5,4,3]; Y = [1,2,7]; a = X + Y, b = X-Y, c = x * Y, d = 2.*X, e =  
2./X, f = 2., g = X./Y, h =1., i = x^2, j = 2.^X, k = X.^ Y
```

```
a =
```

6 6 10

b =

4 2 -4

c =

5 8 21

d =

10 8 6

e =

0.4000 0.5000 0.6667

f =

0.5000 1.0000 3.5000

g =

5.0000 2.0000 0.4286

h =

5.0000 2.0000 0.4286

i =

25 16 9

j =

32 16 8

k =

5 16 2187

The above operations are all valid since in all cases the variable operands are of the same dimension, so the operations are successfully carried out element by element. For the sum and the difference there is no distinction between vectors and matrices, as the operations are identical in both cases.

The most important operators for matrix variables are specified below:

Here are some examples:

$$X = [5,4,3]; Y = [1,2,7]; l = X'Y, m = XY', n = 2*X, o = X/Y, p = Y$$

l =

5 10 35

4 8 28

3 6 21

m =

34

n =

10 8 6

o =

0.6296

p =

0 0 0

0 0 0

0.7143 0.5714 0.4286

All of the above matrix operations are well defined since the dimensions of the operands are compatible in every case. We must not forget that a

vector is a particular case of matrix, but to operate with it in matrix form (not element by element), it is necessary to respect the rules of dimensionality for matrix operations. For example, the vector operations $X \cdot Y$ and $X \cdot Y'$ make no sense, since they involve vectors of different dimensions. Similarly, the matrix operations XY , $2/X$, 2 , X^2 , 2^X and X^Y make no sense, again because of a conflict of dimensions in the arrays.

Here are some more examples of matrix operators.

$$M = [1,2,3;1,0,2;7,8,9]$$

M =

1 2 3

1 0 2

7 8 9

$$B = \text{inv}(M), C = M^2, D = M^{1/2}, E = 2^M$$

B =

-0.8889 0.3333 0.2222

0.2778 -0.6667 0.0556

0.4444 0.3333 -0.1111

C =

24 26 34

15 18 21

78 86 118

D =

0.5219 + 0.8432i 0.5793 - 0.0664i 0.7756 - 0.2344i

0.3270 + 0.0207i 0.3630 + 1.0650i 0.4859 - 0.2012i

1.7848 - 0.5828i 1.9811 - 0.7508i 2.6524 + 0.3080i

E =

1. 0e + 003 *

0.8626 0.9568 1.2811

0.5401 0.5999 0.8027

2.9482 3.2725 4.3816

MATLAB also provides relational operators. Relational operators perform element by element comparisons between two matrices and return an array of the same size whose elements are zero if the corresponding relationship is true, or one if the corresponding relation is false. The relational operators can also compare scalars with vectors or matrices, in which case the scalar is compared to all the elements of the array. Below is a table of these operators.

MATLAB provides symbols to denote logical operators. The logical operators shown in the following table offer a way to combine or negate relational expressions.

Here are some examples:

```
A = 2:7; P = (A>3)&(A<6)
```

P =

```
0 0 1 1 0 0
```

Returns 1 when the corresponding element of A is greater than 3 and less than 6, and returns 0 otherwise.

```
X = 3*ones(3,3); X>=[7 8 9; 4 5 6 ; 1 2 3]
```

ans =

```
0 0 0
```

0 0 0

1 1 1

The elements of the solution array corresponding to those elements of X which are greater than or equal to the equivalent entry of the matrix [7 8 9; 4 5 6; 1 2 3] are assigned the value 1. The remaining elements are assigned the value 0.

MATLAB implements logical functions whose output can take the value true (1) or false (0). The following table shows the most important logical functions.

Below are some examples using the above defined logical functions.

```
V=[1,2,3,4,5,6,7,8,9], isprime(V), isnumeric(V), all(V), any(V)
```

```
V =
```

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

```
ans =
```

```
0 1 1 0 1 0 1 0 0
```

```
ans =
```

```
1
```

```
ans =
```

```
1
```

```
ans =
```

```
1
```

```
B=[Inf, -Inf, pi, NaN], isinf(B), isfinite(B), isnan(B), isreal(B)
```

```
B =
```

```
Inf - Inf 3.1416 NaN
```

ans =

1 1 0 0

ans =

0 0 1 0

ans =

0 0 0 1

ans =

1

```
ismember([1,2,3], [8,12,1,3]), A = [2,0,1];B = [4,0,2]; isequal(2*A,  
B)
```

ans =

1 0 1

ans =

1

Exercise 2-1. Find the number of ways of choosing 12 elements from 30 without repetition, the remainder of the division of 2^{134} by 3, the prime decomposition of 18900, the factorial of 200 and the smallest number N which when divided by 16,24,30 and 32 leaves remainder 5.

```
factorial(30)/(factorial(12)*factorial(30-12))
```

ans =

8.6493e + 007

The command vpa is used to present the exact result.

```
vpa 'factorial(30)/(factorial(12)*factorial(30-12))'
```

ans =

225.

rem(2^134,3)

ans =

0

factor(18900)

ans =

2 2 3 3 3 5 5 7

factorial(100)

ans =

9. 3326e + 157

The command vpa is used to present the exact result.

vpa 'factorial(100)' 160

ans =

0.

N-5 is the least common multiple of 16, 24, 30 and 32.

lcm(lcm(16,24),lcm(30,32))

ans =

480

Then N = 480 + 5 = 485.

Exercise 2-2. In base 5 find the result of the operation defined by $a_{25} a_{a f f 6}$
 $16 + 6789_{ab} a_{12} + 3567_1 8 + 110022_1 3 - 1250$. In base 13 find the

result of the operation $(666551_7) * (aa199800a_{11}) + (fffaaa125_{16}) / (33331_4 + 6)$.

The result of the first operation in base 10 is calculated as follows:

$base2dec('a25aaf6',16) + base2dec('6789aba',12) + \dots$

$base2dec('35671',8) + base2dec('1100221',3) - 1250$

ans =

190096544

We then convert this to base 5:

$dec2base(190096544,5)$

ans =

342131042134

Thus, the final result of the first operation in base 5 is 342131042134.

The result of the second operation in base 10 is calculated as follows:

$base2dec('666551',7) * base2dec('aa199800a',11) + \dots$

$79 * base2dec('fffaaa125',16) / (base2dec('33331',4) + 6)$

ans =

2.7537e + 014

We now transform the result obtained into base 13.

$dec2base(275373340490852,13)$

ans =

BA867963C1496

Exercise 2-3. In base 13, find the result of the following operation:

$$(6665517)_7 * (aa199800a)_{11} + (fffaaa125)_{16} / (333314 + 6).$$

First, we perform the operation in base 10:

A more direct way of doing all of the above is:

$$\text{base2dec}('666551',7) * \text{base2dec}('aa199800a',11) + \dots$$

$$79 * \text{base2dec}('fffaaa125',16) / (\text{base2dec}('33331',4) + 6)$$

ans =

$$2.753733404908515e + 014$$

We now transform the result obtained into base 13.

$$\text{dec2base}(275373340490852,13)$$

ans =

BA867963C1496

Exercise 2-4. Given the complex numbers $X = 2 + 2i$ and $Y = -3 - 3\sqrt{3}i$, calculate $Y^3 X^2 / Y^{90}$, $Y^{1/2}$, $Y^{3/2}$ and $\ln(X)$.

$$X=2+2i; Y=-3-3\sqrt{3}i;$$

$$Y^3$$

ans =

216

$$X^{2/Y^{90}}$$

ans =

050180953422426e-085 - 1 + 7.404188256695968e-070i

$$\text{sqrt}(Y)$$

ans =

1.22474487139159 - 2.12132034355964i

$\text{sqrt}(Y^3)$

ans =

14.69693845669907

$\log(X)$

ans =

1.03972077083992 + 0.78539816339745i

Exercise 2-5. Calculate the value of the following operations with complex numbers:

$(i^{8-i}(-8))/(3-4*i) + 1$

ans =

1

$i^{\sin(1+i)}$

ans =

-0.16665202215166 + 0.32904139450307i

$(2+\log(i))^{(1/i)}$

ans =

1.15809185259777 - 1.56388053989023i

$(1+i)^i$

ans =

0.42882900629437 + 0.15487175246425i

$i^{\log(1+i)}$

ans =

0.24911518828716 + 0.15081974484717i

$$(1+\sqrt{3}i)^{(1-i)}$$

ans =

5.34581479196611 + 1.97594883452873i

Exercise 2-6. Calculate the real part, imaginary part, modulus and argument of each of the following expressions:

$$i^{3i}, (1 + \sqrt{3}i)^{1-i}, i^{ii}, i^i$$

$$Z1=i^{3i};Z2=(1+\sqrt{3}i)^{(1-i)};Z3=(i^i)^i ;Z4=i^i$$

format short

$$\text{real}([Z1 Z2 Z3 Z4])$$

ans =

1.0000 5.3458 0.0000 0.2079

$$\text{imag}([Z1 Z2 Z3 Z4])$$

ans =

0 1.9759 - 1.0000 0

$$\text{abs}([Z1 Z2 Z3 Z4])$$

ans =

1.0000 5.6993 1.0000 0.2079

$$\text{angle}([Z1 Z2 Z3 Z4])$$

ans =

0 0.3541 - 1.5708 0

Exercise 2-7. Generate a square matrix of order 4 whose elements are uniformly distributed random numbers from [0,1]. Generate another square matrix of order 4 whose elements are normally distributed random numbers from [0,1]. Find the present generating seeds, change their value to $\frac{1}{2}$ and rebuild the two arrays of random numbers.

```
rand(4)
```

```
ans =
```

```
0.9501 0.8913 0.8214 0.9218
```

```
0.2311 0.7621 0.4447 0.7382
```

```
0.6068 0.4565 0.6154 0.1763
```

```
0.4860 0.0185 0.7919 0.4057
```

```
randn(4)
```

```
ans =
```

```
-0.4326 -1.1465 0.3273 -0.5883
```

```
-1.6656 1.1909 0.1746 2.1832
```

```
0.1253 1.1892 -0.1867 -0.1364
```

```
0.2877 -0.0376 0.7258 0.1139
```

```
rand('seed')
```

```
ans =
```

```
931316785
```

```
randn('seed')
```

```
ans =
```

931316785

```
randn('seed', 1/2)
```

```
rand('seed', 1/2)
```

```
rand(4)
```

ans =

```
0.2190 0.9347 0.0346 0.0077
```

```
0.0470 0.3835 0.0535 0.3834
```

```
0.6789 0.5194 0.5297 0.0668
```

```
0.6793 0.8310 0.6711 0.4175
```

```
randn(4)
```

ans =

```
1.1650 -0.6965 0.2641 1.2460
```

```
0.6268 1.6961 0.8717 -0.6390
```

```
0.0751 0.0591 -1.4462 0.5774
```

```
0.3516 1.7971 -0.7012 -0.3600
```

Exercise 2-8. Given the vector variables $a = [\pi, 2\pi, 3\pi, 4\pi, 5\pi]$ and $b = [e, 2e, 3e, 4e, 5e]$, calculate $c = \sin(a) + b$, $d = \cos(a)$, $e = \ln(b)$, $f = c * d$, $g = c/d$, $h = d^2$, $i = d^2 - e^2$ and $j = 3d^3 - 2e^2$.

```
a=[pi,2pi,3pi,4pi,5pi],
```

```
b=[exp(1),2exp(1),3exp(1),4exp(1),5exp(1)],
```

```
c=sin(a)+b,d=cos(a),e=log(b),f=c.*d,g=c./d
```

```
h=d.^2, i=d.^2-e.^2, j=3d.^3-2e.^2
```

a =

3.1416 6.2832 9.4248 12.5664 15.7080

b =

2.7183 5.4366 8.1548 10.8731 13.5914

c =

2.7183 5.4366 8.1548 10.8731 13.5914

d =

-1 1 -1 1 -1

e =

1.0000 1.6931 2.0986 2.3863 2.6094

f =

-2.7183 5.4366 - 8.1548 10.8731 - 13.5914

g =

-2.7183 5.4366 - 8.1548 10.8731 - 13.5914

h =

1 1 1 1 1

i =

0 - 1.8667 - 3.4042 - 4.6944 - 5.8092

j =

-5.0000 - 2.7335 - 11.8083 - 8.3888 - 16.6183

Exercise 2-9. Given a uniform random square matrix M of order 3, obtain its inverse, its transpose and its diagonal. Transform it into a lower triangular matrix (replacing the upper triangular entries by 0) and rotate it

90 degrees counterclockwise. Find the sum of the elements in the first row and the sum of the diagonal elements. Extract the subarray whose diagonal elements are at 11 and 22 and also remove the subarray whose diagonal elements are at 11 and 33 .

```
M=rand(3)
```

```
M =
```

```
0.6868  0.8462  0.6539
```

```
0.5890  0.5269  0.4160
```

```
0.9304  0.0920  0.7012
```

```
A=inv(M)
```

```
A =
```

```
-4.1588  6.6947 -0.0934
```

```
0.3255  1.5930 -1.2487
```

```
5.4758 -9.0924  1.7138
```

```
B=M'
```

```
B =
```

```
0.6868  0.5890  0.9304
```

```
0.8462  0.5269  0.0920
```

```
0.6539  0.4160  0.7012
```

```
V=diag(M)
```

```
V =
```

```
0.6868
```

```
0.5269
```

0.7012

TI=tril(M)

TI =

0.6868 0 0
0.5890 0.5269 0
0.9304 0.0920 0.7012

TS=triu(M)

TS =

0.6868 0.8462 0.6539
0 0.5269 0.4160
0 0 0.7012

TR=rot90(M)

TR =

0.6539 0.4160 0.7012
0.8462 0.5269 0.0920
0.6868 0.5890 0.9304

s=M(1,1)+M(1,2)+M(1,3)

s =

2.1869

sd=M(1,1)+M(2,2)+M(3,3)

sd =

1.9149

$$SM = M(1:2, 1:2)$$

SM =

0.6868 0.8462

0.5890 0.5269

$$SM1 = M([1 \ 3], [1 \ 3])$$

SM1 =

0.6868 0.6539

0.9304 0.7012

Exercise 2-10. Given the following complex square matrix M of order 3, find its square, its square root and its base 2 and -2 exponential:

$$M = \begin{bmatrix} i & 2i & 3i \\ 4i & 5i & 6i \\ 7i & 8i & 9i \end{bmatrix}$$

$$M = [i \ 2i \ 3i; 4i \ 5i \ 6i; 7i \ 8i \ 9i]$$

M =

0 + 1.0000i 0 + 2.0000i 0 + 3.0000i

0 + 4.0000i 0 + 5.0000i 0 + 6.0000i

0 + 7.0000i 0 + 8.0000i 0 + 9.0000i

$$C = M^2$$

C =

-30 -36 -42

-66 -81 -96

-102 - 126 -150

$$D=M^{(1/2)}$$

D =

0.8570 - 0.2210i 0.5370 + 0.2445i 0.2169 + 0.7101i

0.7797 + 0.6607i 0.9011 + 0.8688i 1.0224 + 1.0769i

0.7024 + 1.5424i 1.2651 + 1.4930i 1.8279 + 1.4437i

$$2^M$$

ans =

0.7020 - 0.6146i - 0.1693 - 0.2723i -0.0407 + 0.0699i

-0.2320 - 0.3055i 0.7366 - 0.3220i - 0.2947 - 0.3386i

-0.1661 + 0.0036i - 0.3574 - 0.3717i 0.4513 - 0.7471i

$$(-2)^M$$

ans =

17.3946 -16.8443i 4.3404 - 4.5696i - 7.7139 + 7.7050i

1.5685 - 1.8595i 1.1826 - 0.5045i - 1.2033 + 0.8506i

-13.2575 +13.1252 i - 3.9751 + 3.5607i 6.3073 - 6.0038i

Exercise 2-11. Given the complex matrix M in the previous exercise, find its elementwise logarithm and its elementwise base e exponential. Also calculate the results of the matrix operations e^M and $\ln(M)$.

$$M=[i \ 2i \ 3i; \ 4i \ 5i \ 6i; \ 7i \ 8i \ 9i]$$

$$\log(M)$$

ans =

0 + 1.5708i 0.6931 + 1.5708i 1.0986 + 1.5708i

1.3863 + 1.5708i 1.6094 + 1.5708i 1.7918 + 1.5708i
1.9459 + 1.5708i 2.0794 + 1.5708i 2.1972 + 1.5708i

exp(M)

ans =

0.5403 + 0.8415i - 0.4161 + 0.9093i -0.9900 + 0.1411i
-0.6536 - 0.7568i 0.2837 - 0.9589i 0.9602 - 0.2794i
0.7539 + 0.6570i - 0.1455 + 0.9894i -0.9111 + 0.4121i

logm(M)

ans =

-5.4033 - 0.8472i 11.9931 - 0.3109i - 5.3770 + 0.8846i
12.3029 + 0.0537i -22.3087 + 0.8953i 12.6127 + 0.4183i
-4.7574 + 1.6138i 12.9225 + 0.7828i - 4.1641 + 0.6112i

expm(M)

ans =

0.3802 - 0.6928i - 0.3738 - 0.2306i -0.1278 + 0.2316i
-0.5312 - 0.1724i 0.3901 - 0.1434i - 0.6886 - 0.1143i
-0.4426 + 0.3479i - 0.8460 - 0.0561i -0.2493 - 0.4602i

Exercise 2-12. Given the complex vector $V = [1 + i, i, 1-i]$, find the mean, median, standard deviation, variance, sum, product, maximum and minimum of its elements, as well as its gradient, its discrete Fourier transform and its inverse discrete Fourier transform.

$V = [1 + i, i, 1-i]$

```
[mean(V),median(V),std(V),var(V),sum(V),prod(V),max(V),min(V)]'
```

```
ans =
```

```
0.6667 - 0.3333i
```

```
1.0000 + 1.0000i
```

```
1.2910
```

```
1.6667
```

```
2.0000 - 1.0000i
```

```
0 - 2.0000i
```

```
1.0000 + 1.0000i
```

```
0 - 1.0000i
```

```
gradient(V)
```

```
ans =
```

```
1.0000 - 2.0000i 0.5000 0 + 2.0000i
```

```
fft(V)
```

```
ans =
```

```
2.0000 + 1.0000i - 2.7321 + 1.0000i 0.7321 + 1.0000i
```

```
ifft(V)
```

```
ans =
```

```
0.6667 + 0.3333i 0.2440 + 0.3333i - 0.9107 + 0.3333i
```

Exercise 2-13. Given the arrays

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} i & 1-i & 2+i \\ 0 & -1 & 3-i \\ 0 & 0 & -i \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \sqrt{2}i & -\sqrt{2}i \\ 1 & -1 & -1 \end{bmatrix}$$

calculate $AB - BA$, $A^2 + B^2 + C^2$, ABC , $\sqrt{A} + \sqrt{B} + \sqrt{C}$, e^A ($e^B + e^C$), their transposes and their inverses. Also verify that the product of any of the matrices A , B , C with its inverse yields the identity matrix.

$$A = [1 \ 1 \ 0; 0 \ 1 \ 1; 0 \ 0 \ 1]; \quad B = [i \ 1-i \ 2+i; 0 \ -1 \ 3-i; 0 \ 0 \ -i]; \quad C = [1 \ 1 \ 1; 0 \ \sqrt{2}i \ -\sqrt{2}i; 1 \ -1 \ -1];$$

$$M1 = AB - BA$$

$$M1 =$$

$$\begin{bmatrix} 0 & -1.0000 - 1.0000i & 2.0000 \\ 0 & 0 & 1.0000 - 1.0000i \\ 0 & 0 & 0 \end{bmatrix}$$

$$M2 = A^2 + B^2 + C^2$$

$$M2 =$$

$$\begin{bmatrix} 2.0000 & 2.0000 + 3.4142i & 3.0000 - 5.4142i \\ 0 - 1.4142i & -0.0000 + 1.4142i & 0.0000 - 0.5858i \\ 0 & 2.0000 - 1.4142i & 2.0000 + 1.4142i \end{bmatrix}$$

$$M3 = ABC$$

M3 =

5.0000 + 1.0000i - 3.5858 + 1.0000i -6.4142 + 1.0000i

3.0000 - 2.0000i - 3.0000 + 0.5858i -3.0000 + 3.4142i

0 - 1.0000i 0 + 1.0000i 0 + 1.0000i

$$M4 = \sqrt{m(A)} + \sqrt{m(B)} - \sqrt{m(C)}$$

M4 =

0.6356 + 0.8361i - 0.3250 - 0.8204i 3.0734 + 1.2896i

0.1582 - 0.1521i 0.0896 + 0.5702i 3.3029 - 1.8025i

-0.3740 - 0.2654i 0.7472 + 0.3370i 1.2255 + 0.1048i

$$M5 = \expm(A) * (\expm(B) + \expm(C))$$

M5 =

14.1906 - 0.0822i 5.4400 + 4.2724i 17.9169 - 9.5842i

4.5854 - 1.4972i 0.6830 + 2.1575i 8.5597 - 7.6573i

3.5528 + 0.3560i 0.1008 - 0.7488i 3.2433 - 1.8406i

$$\text{inv}(A)$$

ans =

1 -1 1

0 -1 -1

0 0 1

$$\text{inv}(B)$$

ans =

0 - 1.0000i - 1.0000 - 1.0000i -4.0000 + 3.0000i

0 -1.0000 1.0000 + 3.0000i

0 0 0 + 1.0000i

inv(C)

ans =

0.5000 0 0.5000

0.2500 0 - 0.3536i - 0.2500

0.2500 0 + 0.3536i - 0.2500

[Ainv(A) Binv(B) C*inv(C)]

ans =

1 0 0 1 0 0 1 0 0

0 1 0 0 1 0 0 1 0

0 0 1 0 0 1 0 0 1

A'

ans =

1 0 0

1 1 0

0 1 1

B'

ans =

0 - 1.0000i 0 0

1.0000 + 1.0000i - 1.0000 0

2.0000 - 1.0000i 3.0000 + 1.0000i 0 + 1.0000i

C'

ans =

1.0000 0 1.0000

1.0000 0 - 1.4142i - 1.0000

1.0000 0 + 1.4142i - 1.0000

Chapter 3.

2-D Graphics IN MATLAB. explicit, PARAMETRIC and polar cURVES.
exploratory GRAPHS

MATLAB allows the representation of any mathematical function, even if it is defined piecewise or jumps to infinity in his field of definition, make graphs of Planar (two-dimensional) curves and surfaces (three-dimensional), group them and can overlap. You can combine colors, grids, frames, etc., in the graphics. It allows the representations of functions in implicit, explicit and parametric coordinates, and is without a doubt a mathematical software with high graphics performance. This is one of their differences with the rest of the symbolic calculation packages. Animations no more to combine different graphics with slight variations each others, so to display them quickly in succession, they give the impression of movement can be generated from these graphs.

On the other hand, it also allows the typical bar graphs, lines, Star and histograms. It also offers special possibilities of representation of Polyhedra with geographical maps. In the handling of graphics, it is very important to bear in mind the availability of memory on the computer. The graphics drawing consumes lots of memory and requires a high screen resolution.

The basic commands that Matlab uses to draw the graph of a function of a variable are as follows:

plot(X,Y)

draws the set of points (X, Y) , where X and Y are vectors row. For graphing a function $y = f(x)$, it is necessary to know a set of points $(X, f$

(X) , for what it initially set a range of variation vector X to the variable x . X and Y can be arrays of the same size, in which case a graph is made by each pair of rows and on the same axis. For complex values of X and Y , the imaginary parts are ignored

plot (Y)

draws the vector Y elements, i.e., gives the graph of the set of points (t, Y t) t = 1, 2,... n (n = length(Y)). It is useful for graphing time, series if Y is a matrix, plot (Y) made a graph for each column Y presenting all on the same axes. If the components of the vector Y are complex, plot (Y) is equivalent to plot (real (Y), imag (Y)).

plot (X, Y, S)

draws plot(X,Y) with the settings defined in S . Usually, S consists of two digits between single quotes, the first of which sets the color of the line of the graph, and the second sets the character to be used in the plotting. The possible values of colors and characters are, respectively, as follows:

y yellow . points

m magenta o circles

c cyan x x - brands

r Red + plus signs

g green - solid

b Blue * Star

w White : colon

k Black -.with dashes and dots

– semi-solid

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)

combines, on the same axes, graphs defined for the triples (Xi, Yi, Si). It is a way of representing various functions on the same graph.

`fplot('function' [xmin, xmax])`

graphs the function in the variation of x given interval

`fplot('función',[xmin, xmax], S)`

graphs the function at interval of variation of x given, with options for color and characters given by S

`fplot(['f1,f2,...,fn'],[xmin, xmax], S)`

graphs functions f_1, f_2, \dots, f_n on the same axes at interval of variation of x specified, and color and character options defined in S

`fplot(x(t),y(t), [tmin tmax])`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[tmin tmax]$

`ezplot ('function', [xmin xmax])`

graphs the function in the variation of x given interval

`ezplot ('function', [xmin xmax ymin ymax])`

graphs the function in the variations of x and y given intervals

`ezplot(x(t),y(t))`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[0,2\pi]$

`ezplot(x(t),y(t), [tmin tmax])`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[tmin tmax]$

Let's see some examples of 3-dimensional graphics:

We will begin representing function $f(x) = (\sin(x))^2 + 2x\cos(x)$ at $(-2\pi, 2\pi)$

» `x=(-2pi:0.1:2pi);`

» `y = sin(x).^2 + 2x.cos(x);`

» plot(x,y)

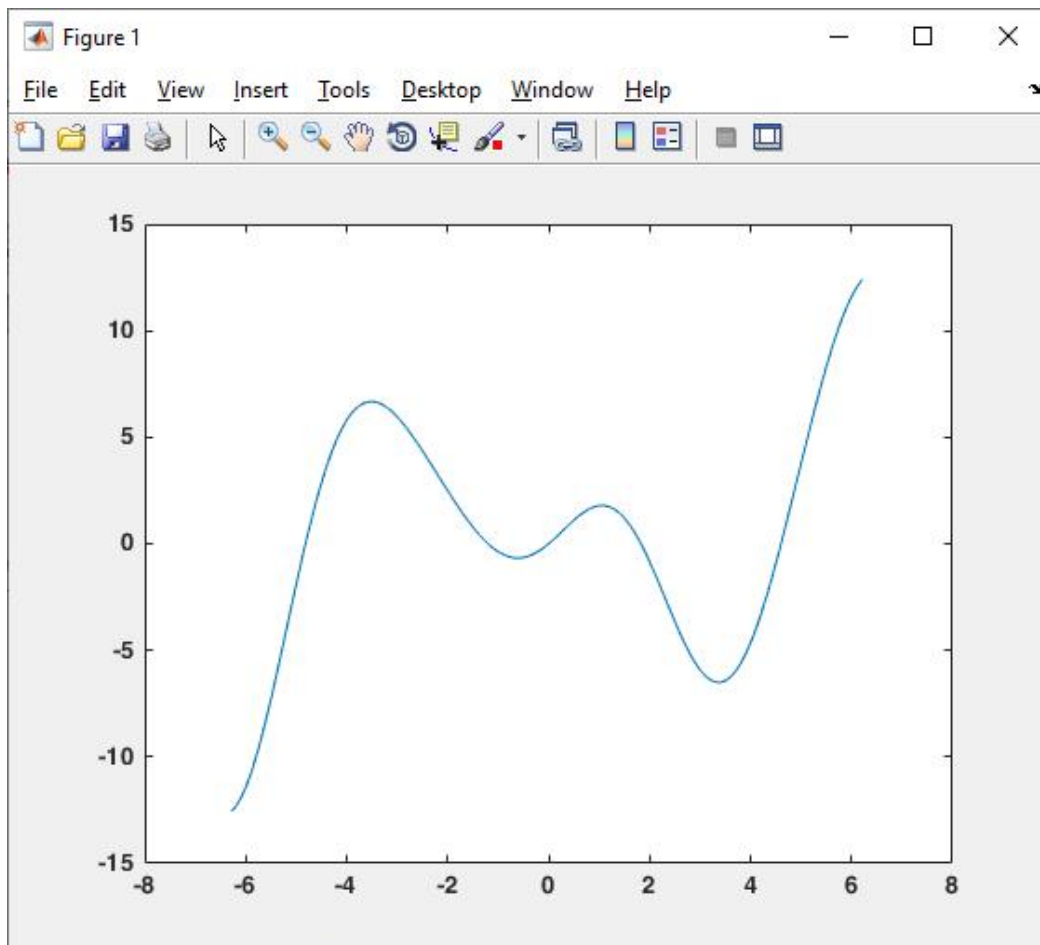


Figure 3-1

MATLAB returns the graph that is shown in Figure 3-1. It is notorious that the definition of the function has been made in vector form, using the notations for the purpose vector variables (already studied previously).

The same graph is obtained using the command `fplot` with the following syntax:

```
» fplot('sin(x)^2+2xcos(x)', [-2pi,2pi])
```

And also can be obtained by the same representation using the command `ezplot` using the following syntax:

```
» ezplot('sin(x)^2+2xcos(x)', [-2pi,2pi])
```

It is observed that in the last two cases function is expressed symbolically, and not of vector, as in the first case form.

MATLAB draws not only bounded functions, but it also represents features that have asymptotes and singularities. For example, Figure 3-2 shows the graph of the function $y = x^3 / (x^2 - 4)$ in the range of variation of x given by $(- 8.8)$ by using the command:

```
» ezplot('x^3/(x^2-4)', [- 8, 8])
```

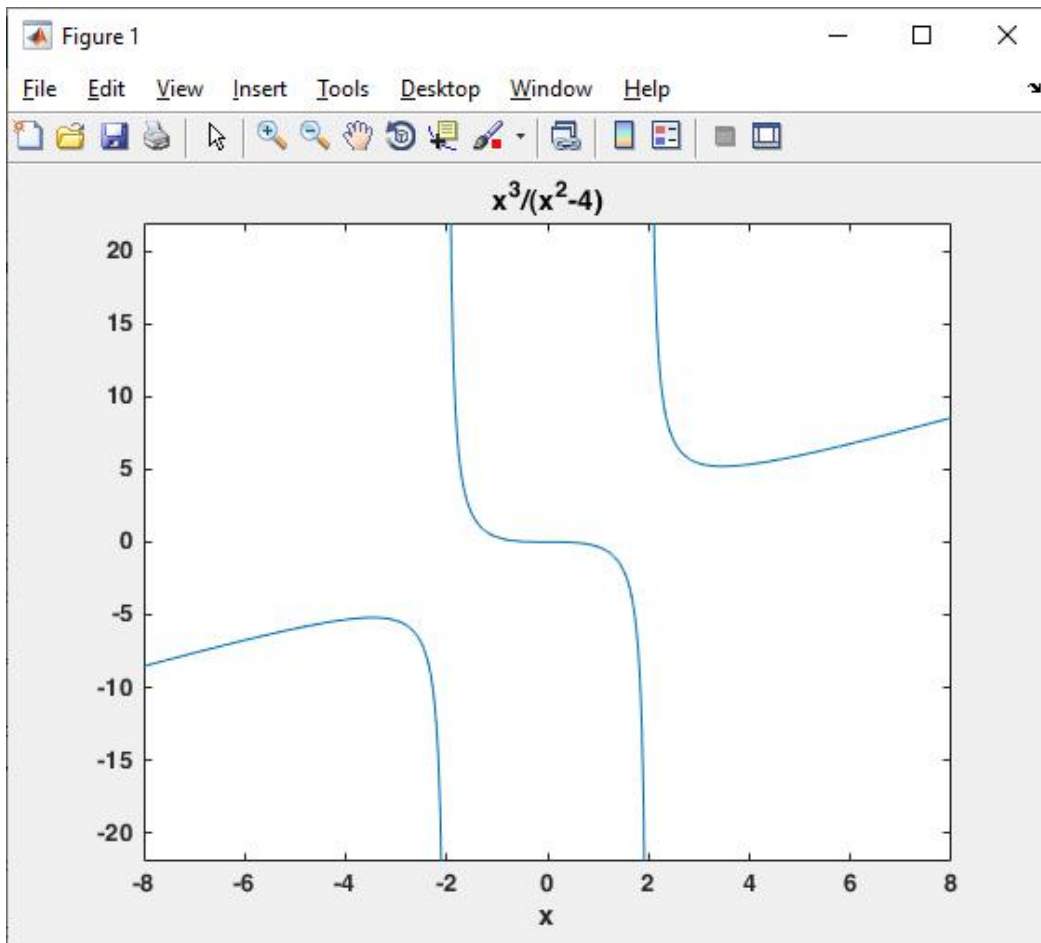


Figure 3-2

Exercise 3-1. Represent on the same axes the graphs of the functions $\sin(x)$, $\text{Sen}(2x)$ and $\text{Sen}(3x)$, for x varying in the range $(0, 2\pi)$.

The graphics, generated by the input is represented in Figure 3-3:

```
» fplot('sin(x),sin(2x),sin(3x)', [0, 2*pi])
```

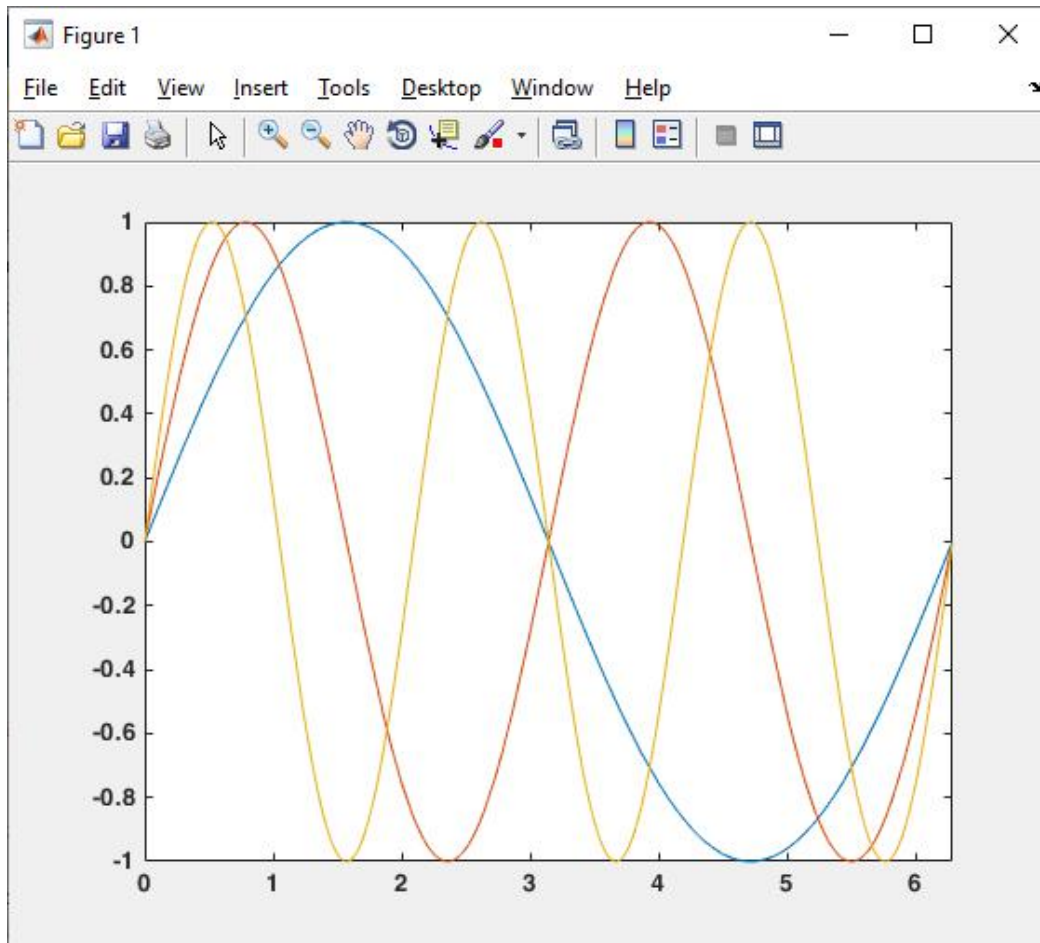


Figure 3-3

It may be interesting to differentiate between curves by their strokes. To do this, we will represent in Figure 3-4 the first function, $\sin(x)$, with black line, the second, $\sin(2x)$, Blue Star, and the third, $\sin(3x)$, with red circles. We use the following syntax:

```
» x=(0:0.05:2*pi);
```

```
» y1 = sin(x); y2 = sin(2x); y3 = sin(3x);
```

```
» plot(x,y1,'k-',x,y2,'b*',x,y3,'ro')
```

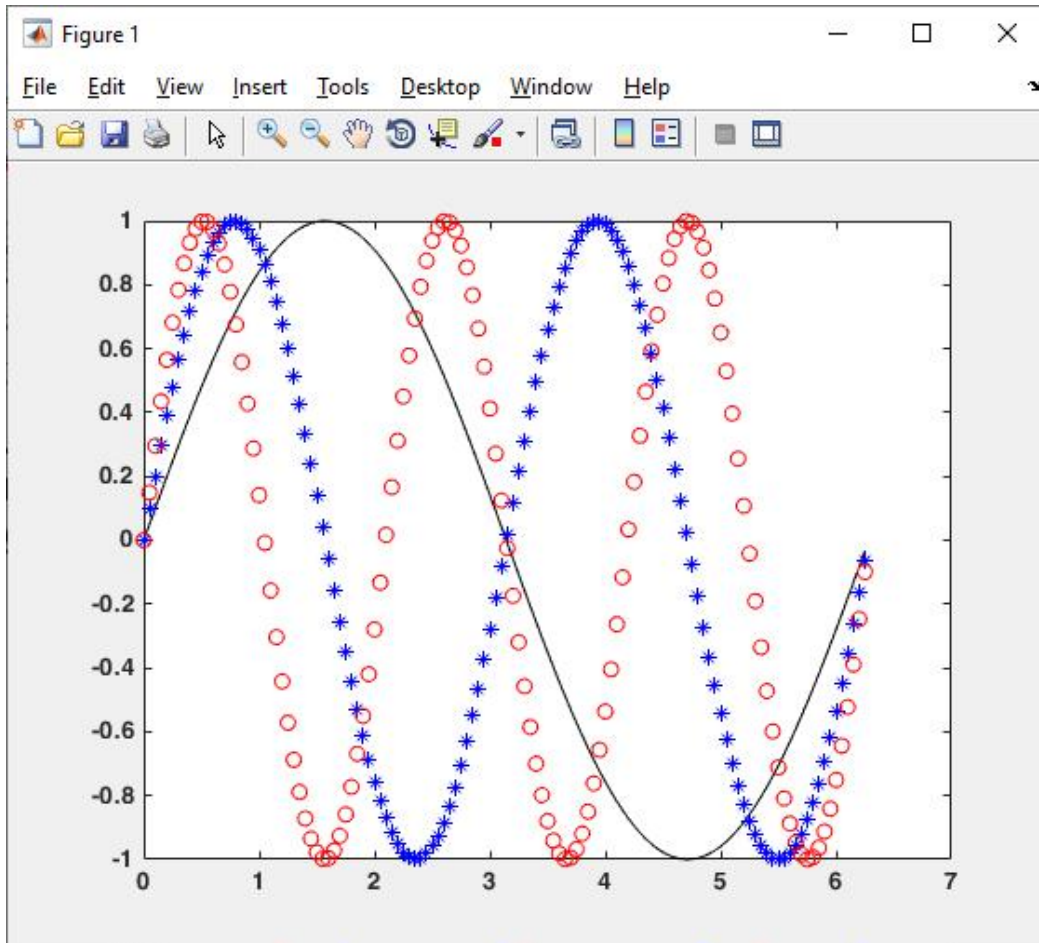


Figure 3-4

The commands available in Matlab for the purpose are as follows:

`title ('text')`

adds the text as the title of the graph at the top of the same in 3-D and 3-D graphics

`xlabel ('text')`

places the text next to the axis x in 3-D and 3-D graphics

`ylabel ('text')`

puts the text next to the axis and in 3-D and 3-D graphics

`zlabel ('text')`

places the text beside axis z in a 3-D chart

```
text (x, y, 'text')
```

places the text at the point (x, and) within the 3-D chart

```
text (x, and z, 'text')
```

places the text at the point (x, y, z) in 3D graphic

```
gtext ('text')
```

allows you to place text in a selected point with the mouse within a 3-D chart

grid

located grids in a 3-D or 3-D chart axes The option on grid place grates and greed off removes them. The option grid swap between on and off. hold keeps the graph with all its properties, so that the following graphic that is placed on the same axis and overlaps the existing. The option hold on active option and hold off delete. The option hold swap between on and off . Valid for 3-D and 3-D

Exercise 3-2. On the same axes represent the graphs of the functions $y = \sin(x^2)$ e $y = \log(\sqrt{x})$. The text of each equation is properly positioned within the graph and chart holder and two shafts.

We get the graph in Figure 3-5 considering MATLAB entry:

```
» x =linspace(0,2,30);
```

```
»y=sin(x.^2);
```

```
»plot(x,y)
```

```
»text(1,0.8,'y=sin(x^2)')
```

```
»hold on
```

```
»z=log(sqrt(x));
```

```
»plot(x,z)
»text(1,-0.1,'y=log(sqrt(x))')
» x label('Eje X');
»ylabel('Y axis');
»title('Graphic sine and logarithmic');
```

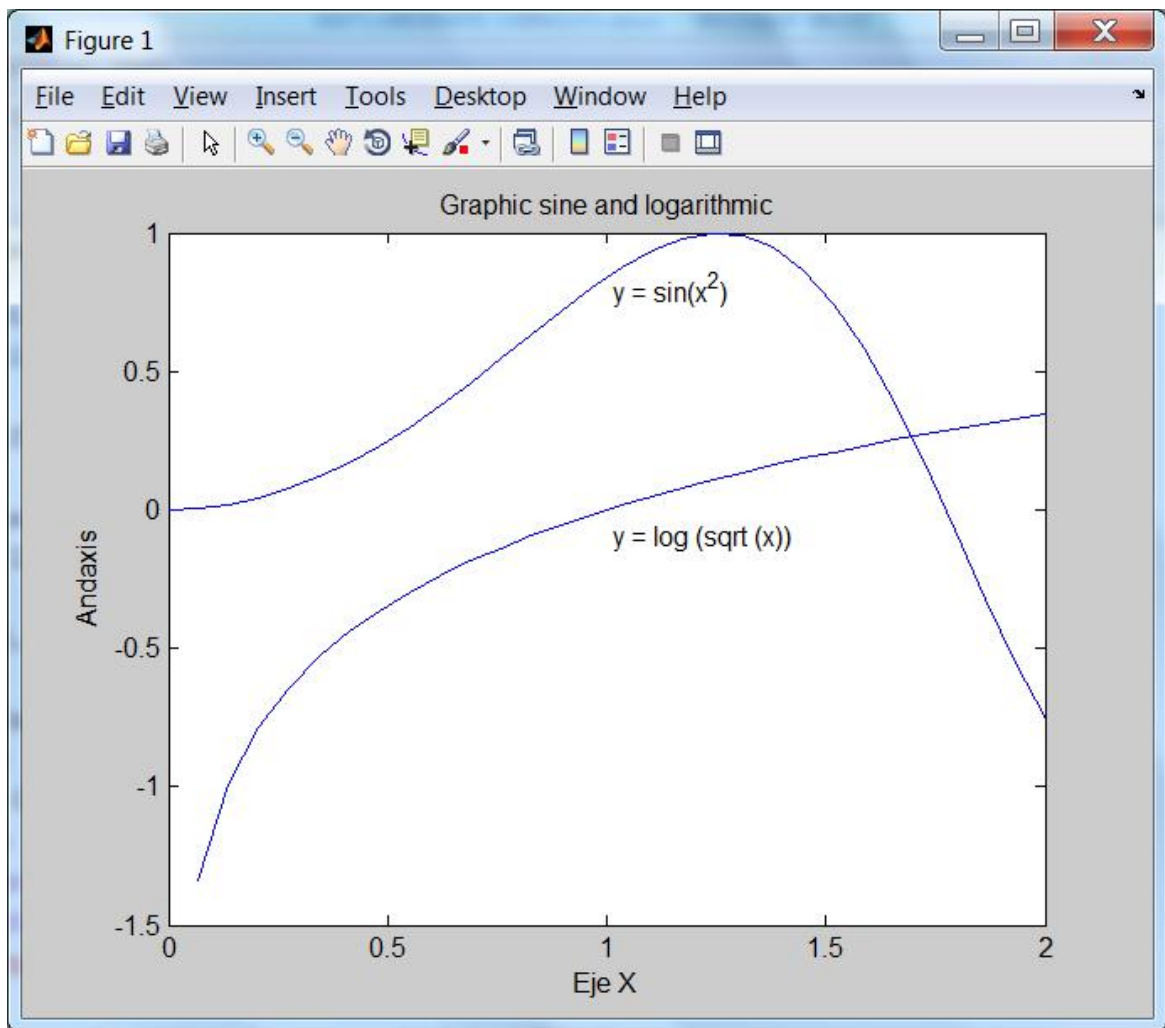


Figure 3-5

Below, are commands that allow you to manipulate the axes of a graph, placement of it within the screen, their appearance, their presentation from different points of view, etc.

`axis([xmin xmax ymin ymax])`

situates the minimum and maximum values for the X and Y axes in the graphic

`axis('auto')`

situates shafts in the Auto scale by default (the one given by $x_{min} = \min(x)$, $x_{max} = \max(x)$ e and free)

`axis(axis)`

freezes the scaling of axes in the currents, so that limits to placing other graphic on the same axes (with hold in on), the scale does not change

`V = axis`

gives the vector V of 4 elements, containing the scale of the current graphic

`axis('xy')`

situates Cartesian coordinates, with the origin at the bottom left of the graph

`axis('ij')`

set coordinates with the origin at the top left of the graph

`axis('square')`

the plotted rectangle becomes a square, so the figures will absorb

`axis('equal')`

puts the same factor of scale for both axes

`axis('normal')`

eliminates the options square and equal

`axis('off')`

eliminates labels and brands of the axes and grids, keeping the title of the chart and the texts within the text with gtext

axis('on')

reposition labels, marks and axes grids

subplot(m, n, p)

divides the graphics window in mxn subwindows and placed the current graphic window p-th , beginning to count from the left top and from left to right until the finish line, to go to the next

Exercise 3-3. Present in the same graph the graphs of the functions sin (x) and Cos (x), placed horizontally one next to each other with their names, and the x axis values between 0 and 2 * pi and the shaft and taking values between - 1 and 1. Also get the vertical representation, one under the other and slotted shafts.

MATLAB, we propose the following entry:

```
» x=(0:0.1:4*pi);
```

```
» y = sin(x);
```

```
» z = cos(x);
```

```
» subplot(1,2,1);
```

```
» plot(x,y), axis([0 2*pi -1 1]), title('sin(x)')
```

```
» subplot(1,2,2);
```

```
» plot(x,z), axis([0 2*pi -1 1]), title('cos(x)')
```

The result is presented in Figure 3-6:

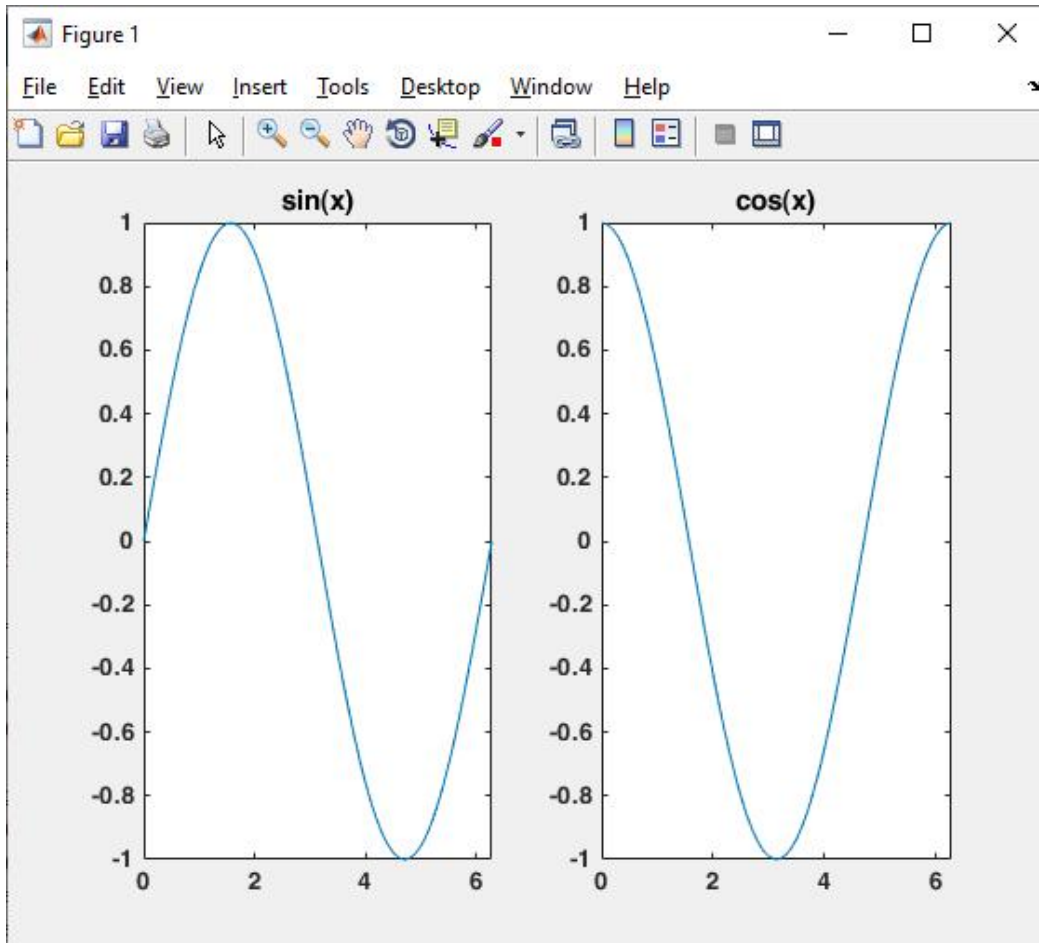


Figure 3-6

We now propose to Matlab the following entry:

» `x=(0:.1:4*pi);`

» `y = sin(x);`

» `z = cos(x);`

» `subplot(2,1,1);`

» `plot(x,y), axis([0 2*pi -1 1]), title('sin(x)'), grid`

» `subplot(2,1,2);`

» `plot(x, z), axis([0 2*pi-1 1]), title ('cos(x)'), grid`

The result is presented in Figure 3-7:

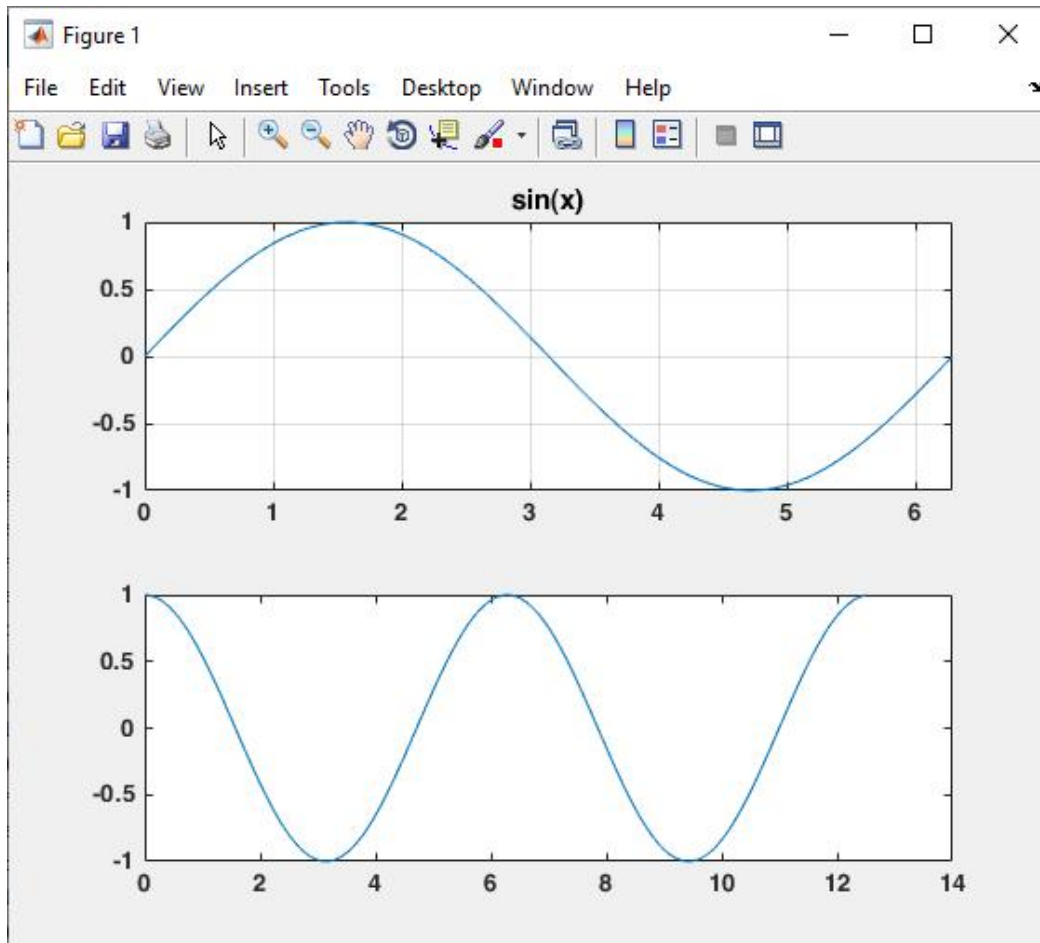


Figure 3-7

Exercise 3-4. Present in the same graph the graphs of the functions $\sin(x)$, $\cos(x)$, $\csc(x)$ and $\sec(x)$, placed in a matrix of four graphics, form under each function is its inverse for x ranging in $[-2\pi, 2\pi]$.

We use the command `subplot` to draw the four functions, in the appropriate order under $\sin(x)$ place $\csc(x)$, which under $\cos(x)$ will place $\sec(x)$. The syntax will be as follows:

```
»subplot(2,2,1);
```

```
»ezplot('sin(x)',[-2pi,2pi])
```

```
»subplot(2,2,2);
```

```
»ezplot('cos(x)',[-2pi,2pi])
```

```
»subplot(2,2,3);
```

```
»ezplot('csc(x)',[-2pi,2pi])
```

```
»subplot(2,2,4);
```

```
»ezplot('sec(x)',[-2pi,2pi])
```

MATLAB offers as a result the graph of Figure 3-8.

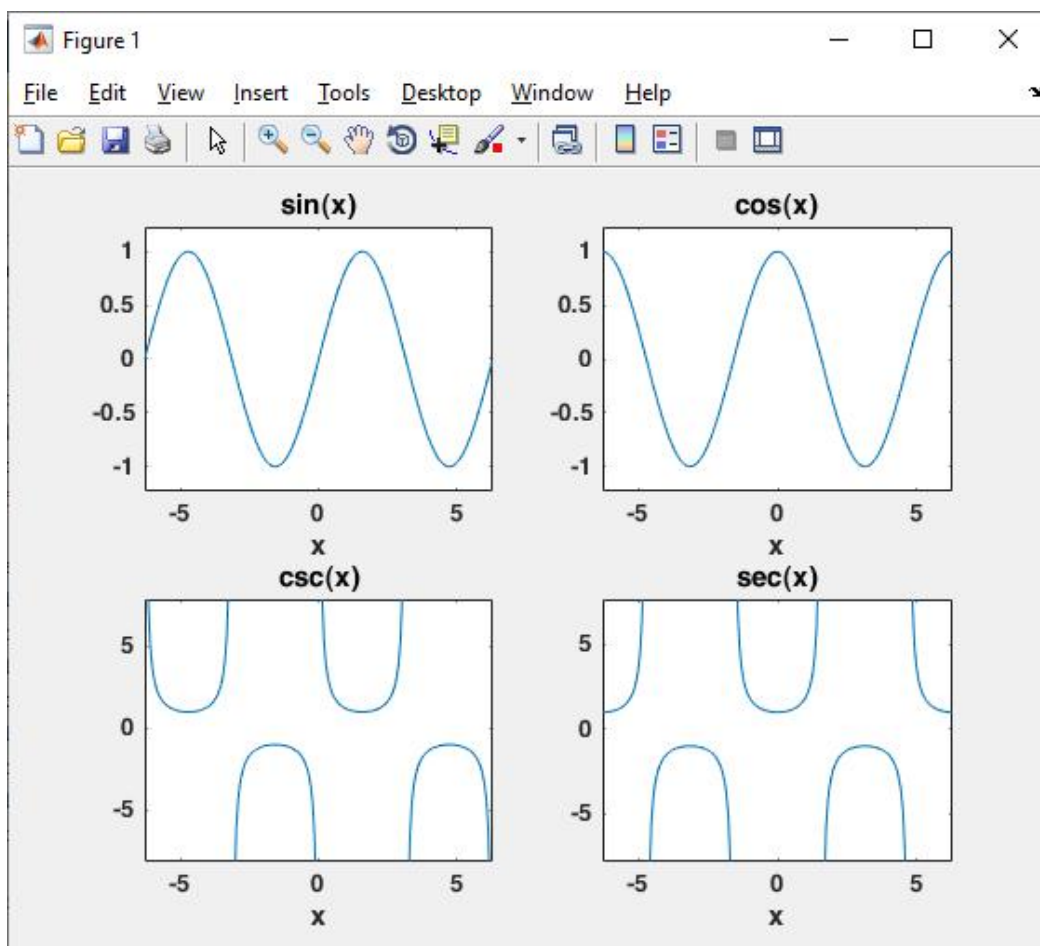


Figure 3-8

The commands that enable Matlab to represent graphs with logarithmic scales are the following:

```
loglog(X,Y)
```

performs the same graphics as `plot(X,Y)` , but with logarithmic scale on the two axes. The command presents the same variants and supports the same options as the command `plot`

`semilogx(X,Y)`

performs the same graphics as `plot(X,Y)` , but with logarithmic scale on the axis x , and normal scale on the axis and (semilogarithmic scale)

`semilogy(X,Y)`

performs the same graphics as `plot(X,Y)` , but with logarithmic scale on the axis and , and normal scale on the axis x (semilogarithmic scale)

Exercise 3-5. Present on the same graph the function $y = \text{abs}(e^{-1/2 x} \sin(5x))$ represented in normal scale, logarithmic scale and semilogarithmic scales.

The syntax presented here leads us to figure 3-9, which compares the graph of the same function for the different scales. It will be represented in the upper part of the figure scales and logarithmic, and normal at the bottom semilogarithmic scales.

» `x = 0:0.01:3;`

» `y = abs(exp(-0.5x).sin(5*x));`

» `subplot(2,2,1)`

» `plot(x,y)`

» `title('normal')`

» `hold on`

» `subplot(2,2,2)`

» `loglog(x,y)`

» `title('logarithmic')`

» `subplot(2,2,3)`

- » `semilogx(x,y)`
- » `title('semilogarithmic in X axis')`
- » `subplot(2,2,4)`
- » `semilogy(x,y)`
- » `title('semilogarithmic in Y axis')`

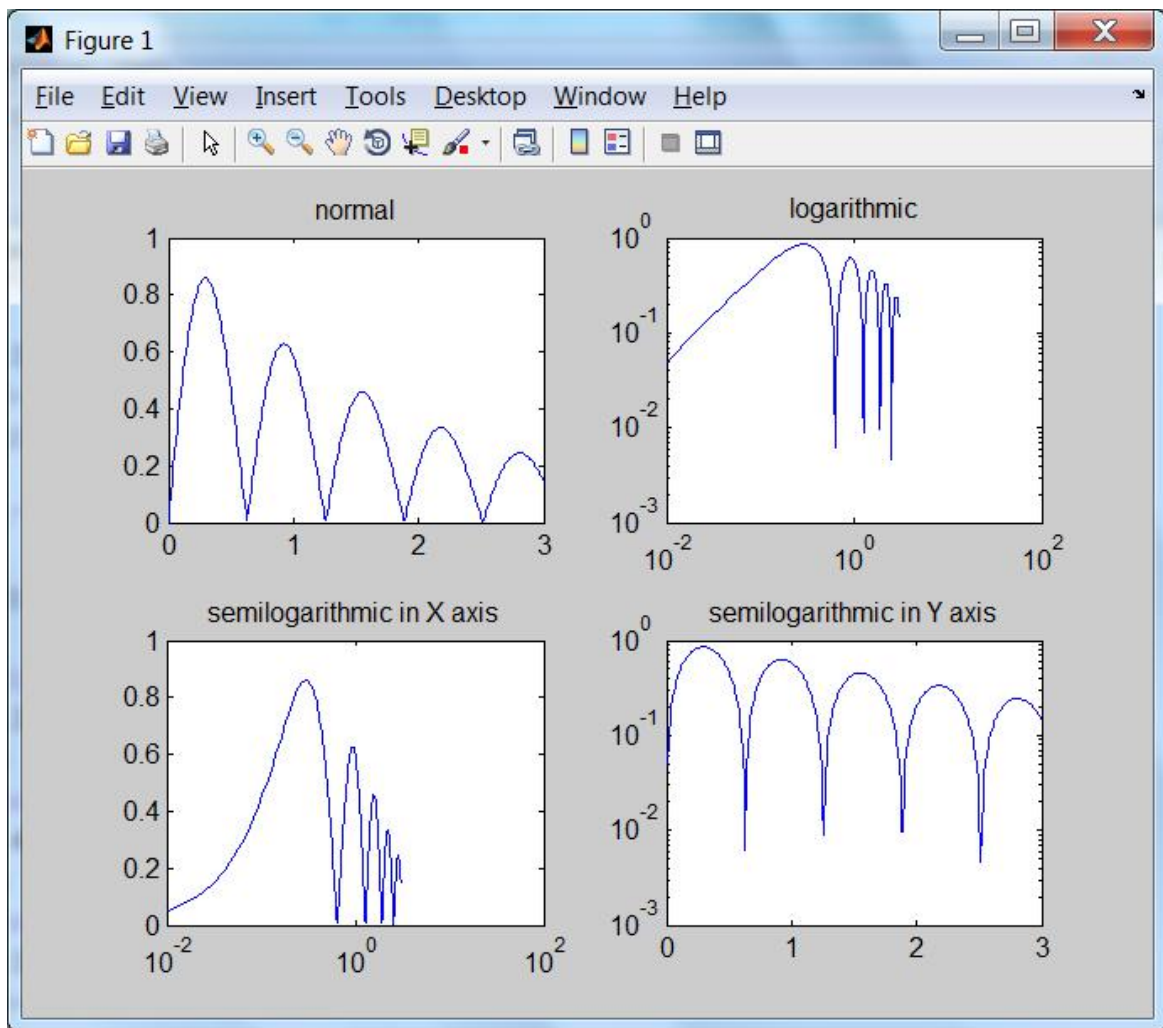


Figure 3-9

MATLAB also allows drawing polygons in two dimensions. To do this, use the following commands:

`fill (X, Y, C)`

draws the compact polygon whose vertices are pairs of components (X_i, Y_i) of the vectors X and Y column. C is a vector of the same size of X and Y which contains the colors C_i for each point (X_i, Y_i) . The C_i values may be: 'r', 'g', 'b', 'c', ' ', 'y', 'w', 'k', whose meanings we already know. If C is a single character, will be painted all the points of the polygon the color corresponding to the character. If X and Y are arrays of the same size, will be represented at the same time several polygons corresponding to each pair of vectors column (X_j, Y_j) . In this case, C can be a vector row C_j elements determine the unique color of each pair of vectors for polygon column (X_j, Y_j) . C can also be an array of the same dimension as X (e) Y , in which case its elements determine the color of each point (X_{ij}, Y_{ij}) of the set of polygons.

```
fill(X1,Y1,C1,X2,Y2,C2,...)
```

draws the compact polygon whose vertices are given by the points (X_i, Y_i, C_i) , the meaning of which we already know

Exercise 3-6. An octagon represent regular (square enclosure), whose vertices are defined by pairs of values $(\text{Sen}(t), \text{Cos}(t))$, for values of t varying between 8π and $15\pi/8$ separate $2\pi/8$. Use only the green color and put at the point $(-1/4,0)$ from the inside of the figure the 'Octagon' text.

Octagon ordering, product of the syntax is shown in Figure 3-10 si-following:

```
» t=(pi/8:2pi/8:15pi/8);
» x = sin(t); y = cos(t);
» fill(x,y,'g')
» axis('square')
» text(-0.25,0,'OCTOGON')
```

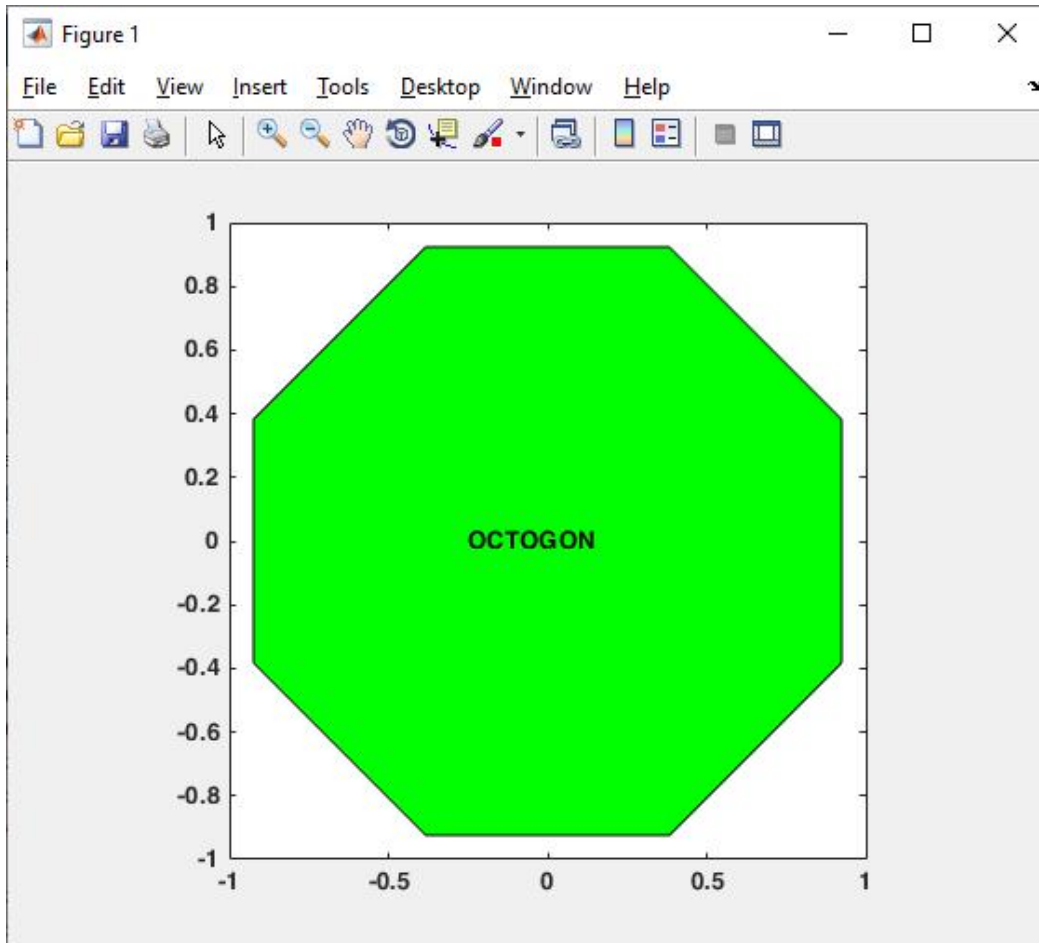


Figure 3-10

Following in the graphic power of Matlab line, we are going to see now how the program draws curves in parametric coordinates in the plane. Will discuss how you can get graphs of functions in which the variables x and y depend, in turn, of a parameter t . The commands to use are `plot`, `fplot` and `ezplot` and all its variants, conveniently defining intervals of variation parameter, and not the independent variable, as it was until now.

`plot(x(t),y(t))`

graphs the parametric curve $x=x(t)$ $y=y(t)$

`fplot(x(t),y(t), [tmin tmax])`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[tmin tmax]$

`ezplot(x(t),y(t))`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[0,2\pi)$

`ezplot(x(t),y(t), [tmin tmax])`

graphs the parametric curve $x=x(t)$ $y=y(t)$ for t en $[tmin tmax]$

Exercise 3-7. Represent the curve (Epicycloid) whose parametric coordinates are: $x = 4\cos[t] - \cos [4t]$ $y = 4\sin[t] - \sin [4t]$, for t varying between 0 and 2π .

The syntax will be as follows:

`t = 0:0.01:2*pi;`

`x =4cos(t)-cos(4t);`

`y=4sin(t)-sin(4t);`

`plot(x,y)`

The retrieved graph is presented in Figure 3-11, and represents the epicycloid .

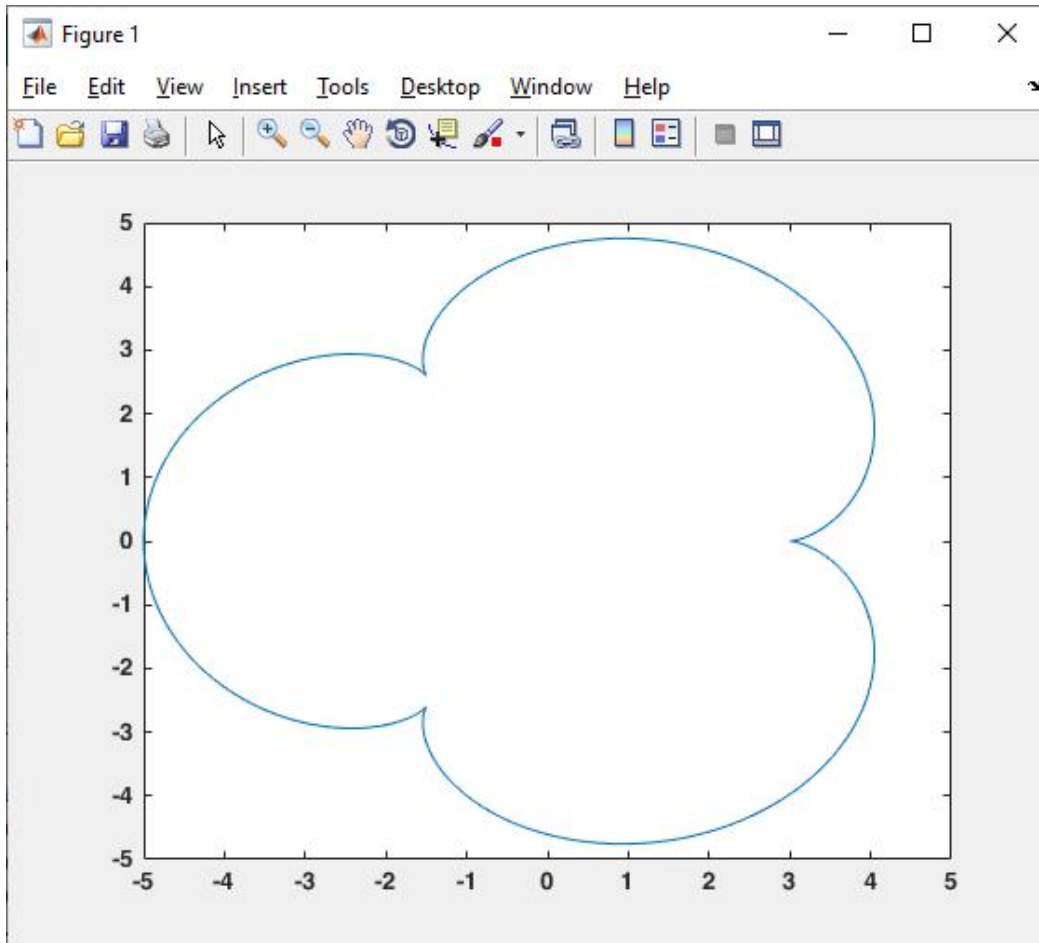


Figure 3-11

Exercise 3-8. Represent the graph of the Cycloid whose parametric equations are $x = t - 2\sin(t)$, $y = 1 - 2\cos(t)$, for t varying between -3π and 3π .

We will use the following syntax:

```
»t=-3pi:0.001:3pi;
```

```
»plot(t-2sin(t),1-2cos(t))
```

Gets the graph in Figure 3-12:

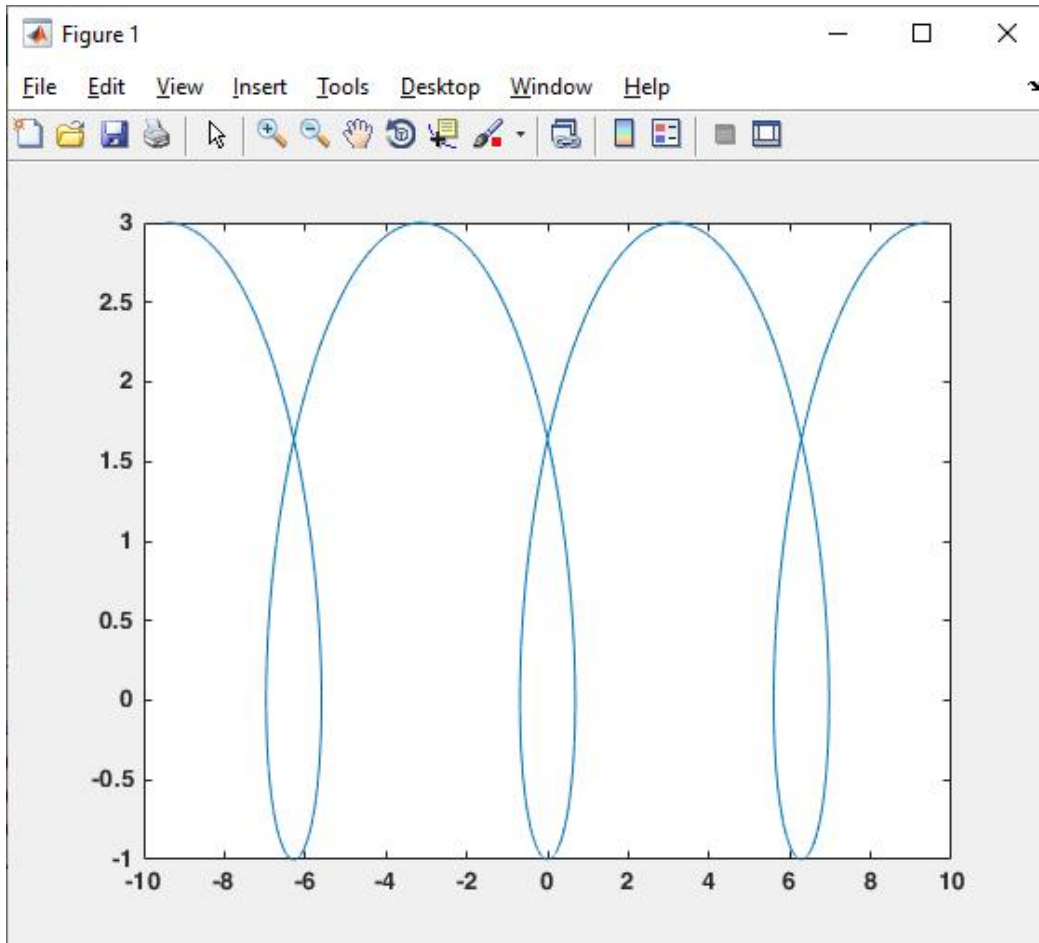


Figure 3-12

Exercise 3-9. Represent the graph whose parametric equations are $x = t + \sin(t)$, $y = t - \cos(t)$, for t varying between -4π and 4π .

In this case we use the `fplot` command to represent parametric curves. The graph of the figure 3-13 is obtained.

```
fplot(t+sin(t),t-cos(t), [-4pi,4pi])
```

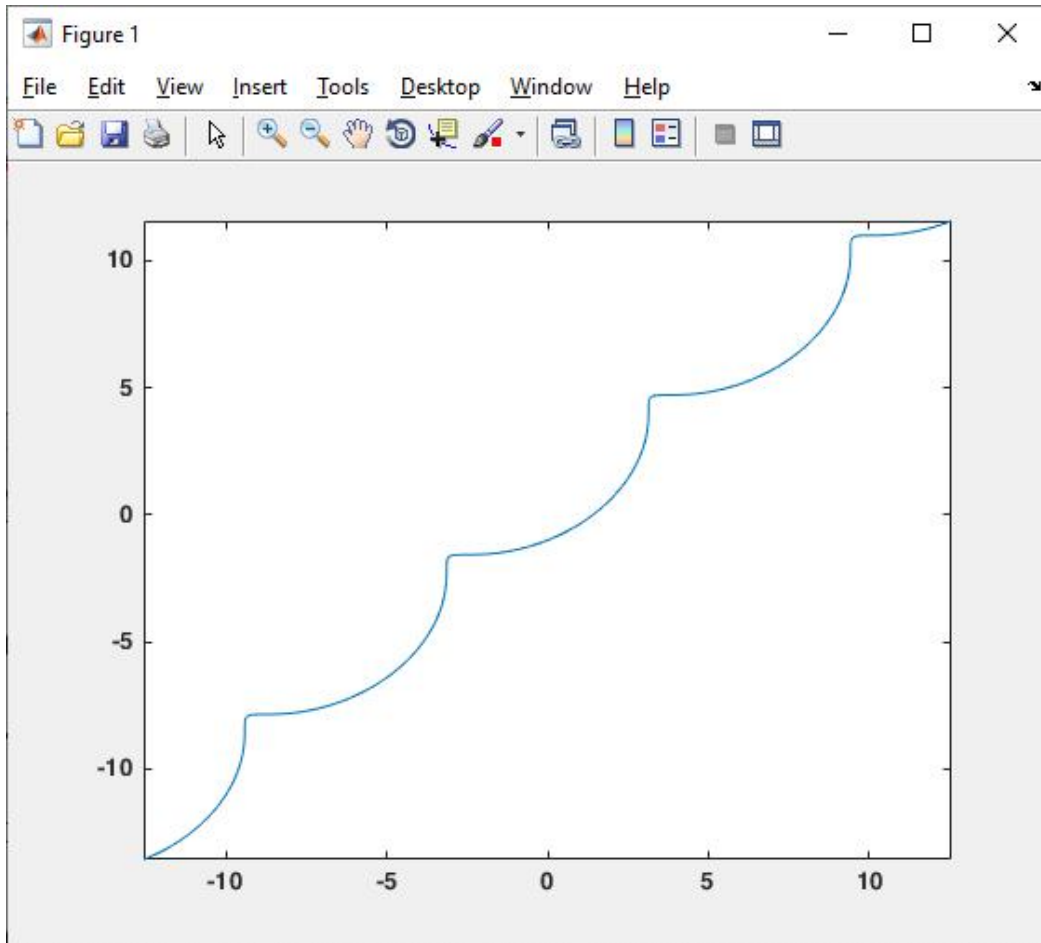


Figure 3-13

Exercise 3-10. Represent the graph whose parametric equations are $x = t + \sin^2(t)$, $y = t - \cos^2(t)$, for t varying between -4π and 4π .

In this case we use the `ezplot` command to represent parametric curves. The graph of the figure 3-14 is obtained.

```
ezplot(t + (sin(t))^2,t - (cos(t))^2*(t), [-4*pi,4pi])
```

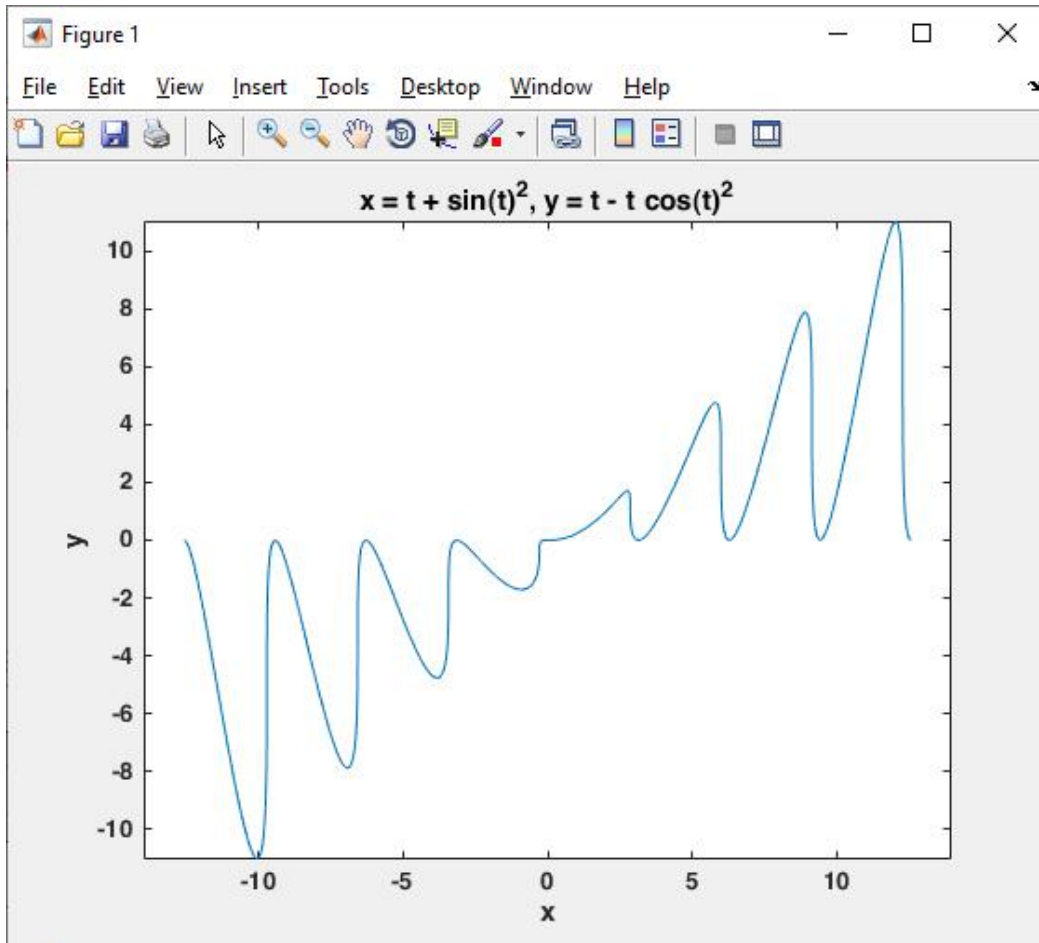


Figure 3-14

MATLAB enables the specific command `polar`, representing functions on coordinates polar. Its syntax is as follows:

`Polar (a , r)`

represents the curve in polar coordinates $r = r (a)$

`Polar (a , r , S)`

represents the curve in polar coordinates $r = r (a)$ with the style of line given by `S`, whose values were already specified in the command `plot`

Exercise 3-11. Represent the graph of the curve whose equation in polar coordinates is as follows: $r = \sin(2a)\cos(2a)$ to a between 0 and 2π .

The following syntax takes us to the graph in Figure 3-15:

```
»a=0:0.01:2*pi;
```

```
»r=sin(2a).cos(2*a);
```

```
»polar(a,r)
```

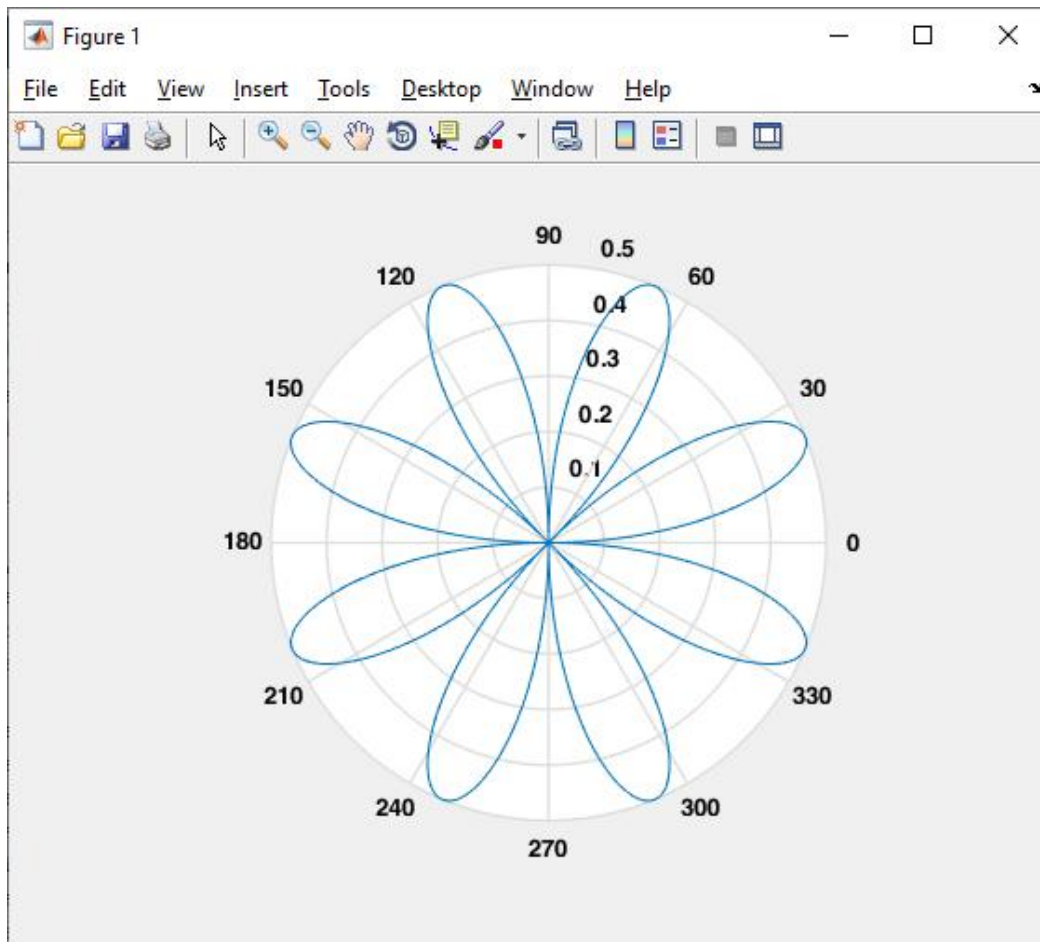


Figure 3-15

Exercise 3-12. Represent the graph of the curve whose equation in polar coordinates is as follows: $r = 4(1 + \cos(a))$ for a between 0 and 2π , called cardioid.

Gets the graph of Figure 3-14, representing the cardioid, using the following syntax:

```
»a=0:0.01:2*pi;
```

```
»r=4*(1+cos(a));
```

```
»polar(a,r)
```

```
»title('CARDIOID')
```

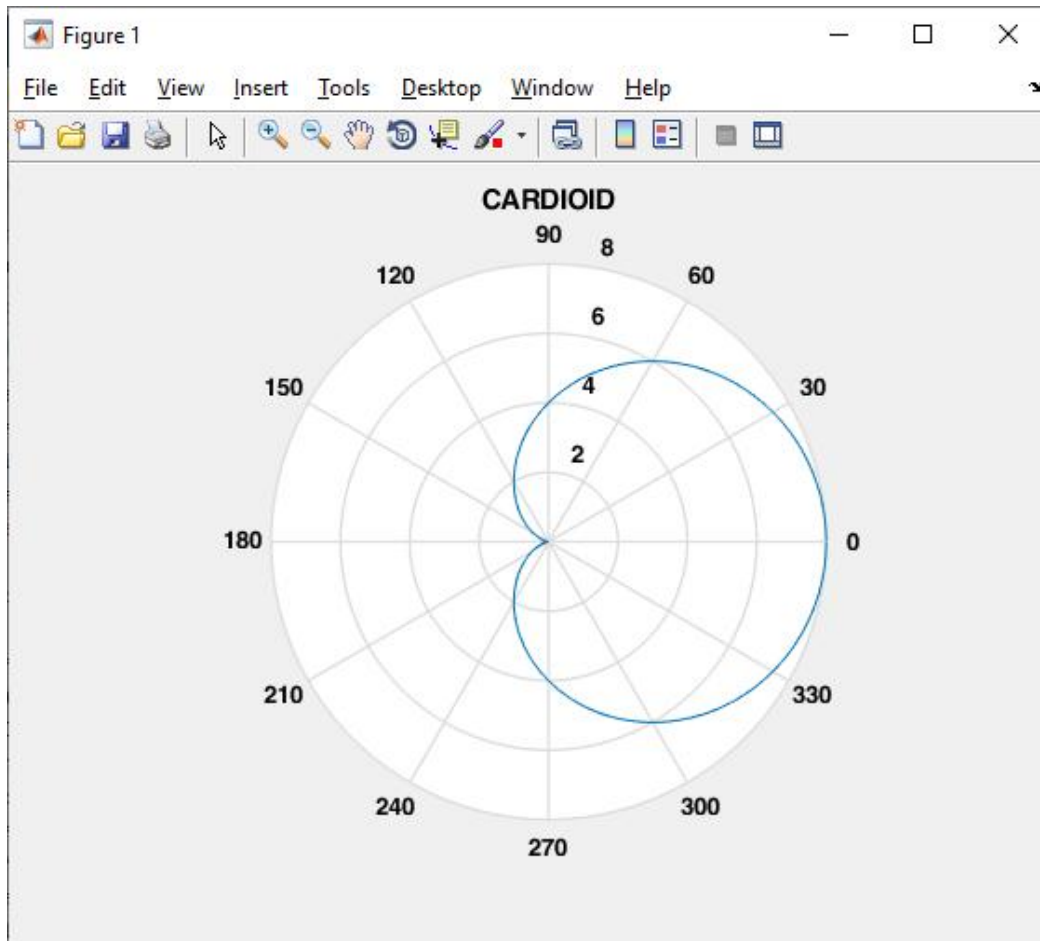


Figure 3-16

Exercise 3-13. Represent the graph of the lemniscata of Bernoulli whose equation is $r=4(\cos(2a)^{(1/2)})$ for between 0 and 2π , and the graph of the spiral of Archimedes whose equation is $r = 3a$, $-4\pi < a < 4\pi$.

The first curve is represented in Figure 3-17, and is obtained by the following syntax:

```
»a=0:0.01:2*pi;
```

```
»r=4(cos(2a).^(1/2));
```

```
»polar(a,r)
```

» title('Bernoulli LEMNISCATA')

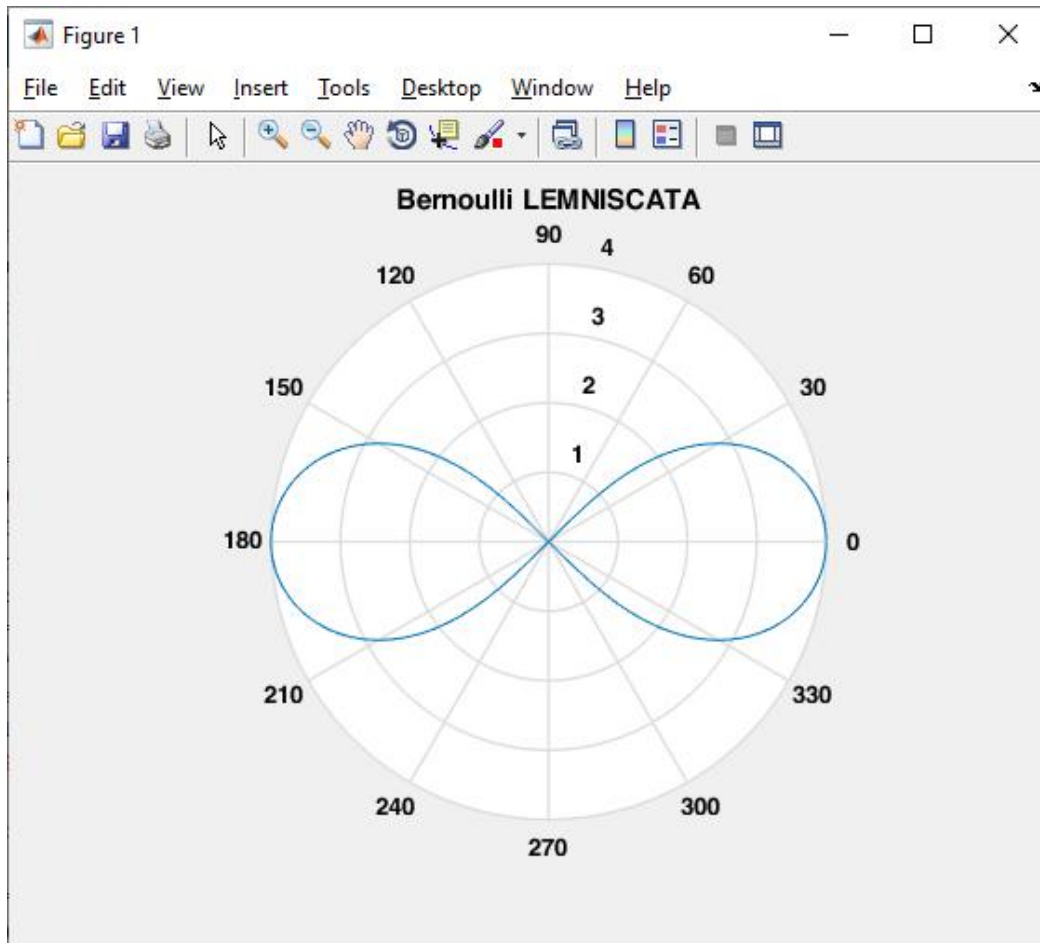


Figure 3-17

The second curve is represented in Figure 3-18, and is obtained by the following syntax:

```
»a=-4pi:0.01:4pi;
```

```
»r=3*a;
```

```
»polar(a,r)
```

```
»title('spiral of ARCHIMEDES');
```

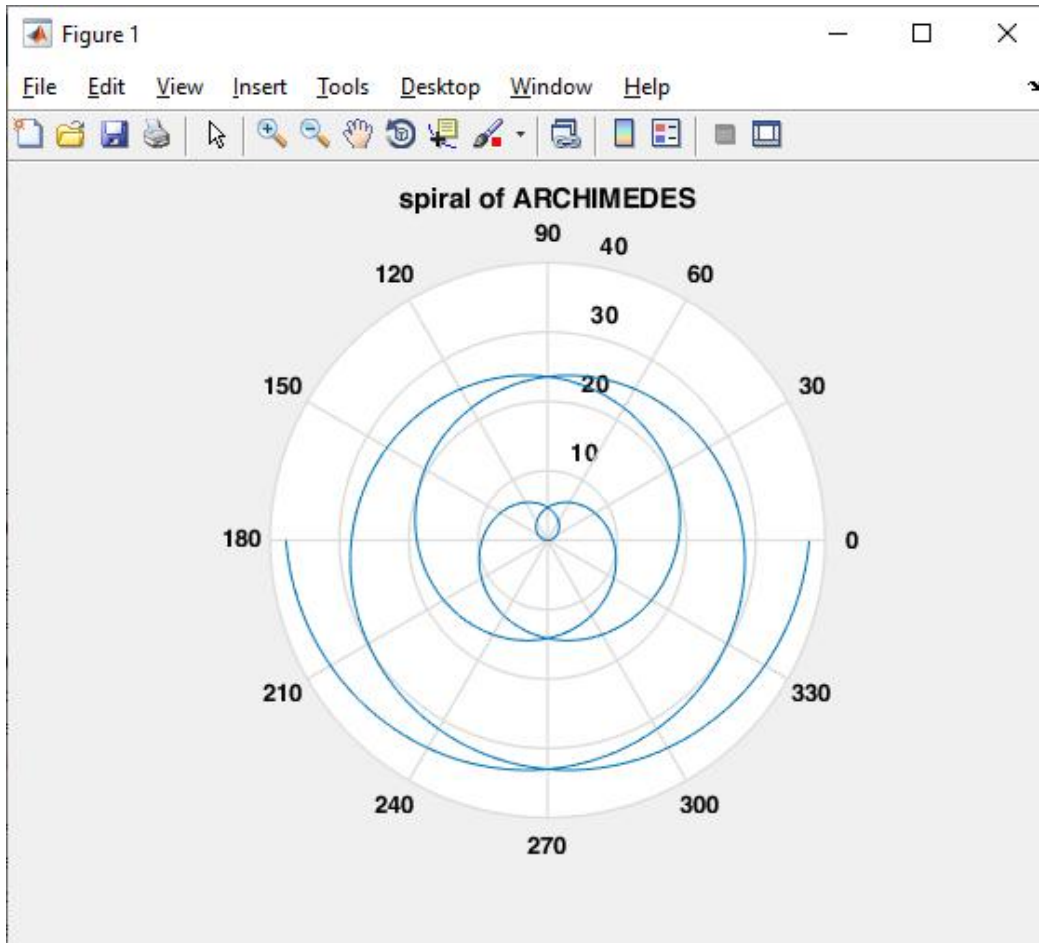



Figure 3-18

MATLAB constructed bar graphs, sectors, Pareto diagrams and histograms of frequencies through the following commands:

`bar (Y)`

draw bar graph relative to the vector of frequencies Y

`bar(X,Y)`

draws the bar graph on the vector of frequencies Y whose elements are given by the vector X

`stairs (Y)`

draw the relative to the vector and staggered graph

stairs(X,Y)

draw the ladder graph relative to the vector and whose elements are given by the vector X

hist (Y)

draws the histogram relative to the vector frequencies and using 10 vertical rectangles of equal base

hist(Y,n)

draws the histogram relative to the vector of frequencies and using vertical rectangles of equal base

hist(X,Y)

draws the histogram relative to the vector of frequencies and using rectangles vertical whose bases measure specified in the elements of the vector X

foot (X)

draws the pie chart relative to the vector of frequencies X

pie(X,Y)

draws the pie chart relative to the vector frequency X moving out the sectors in which $Y_i \neq 0$

pareto (X)

draws the Pareto graph relative to the vector X

Here are some examples:

» bar ([1, - 3, 4, 5, 2, 3])

» foot ([1, 3, 4, 5, 2, 3])

The graphics of the figures are 3-19 and 3-20:

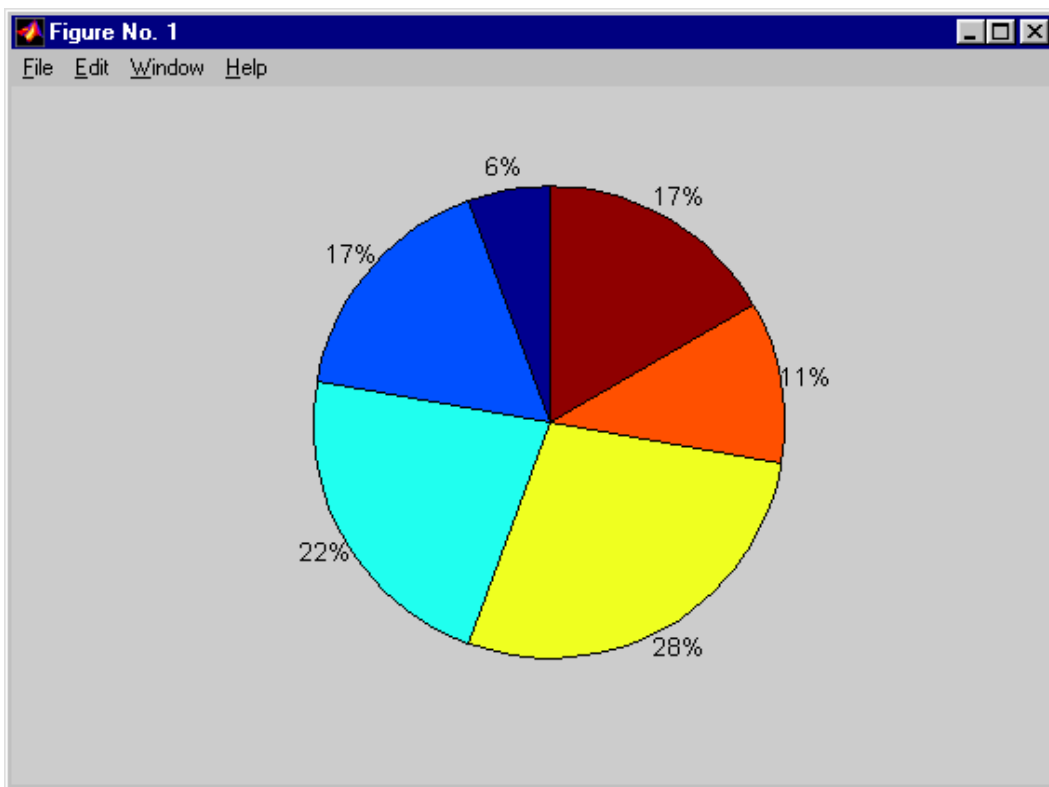
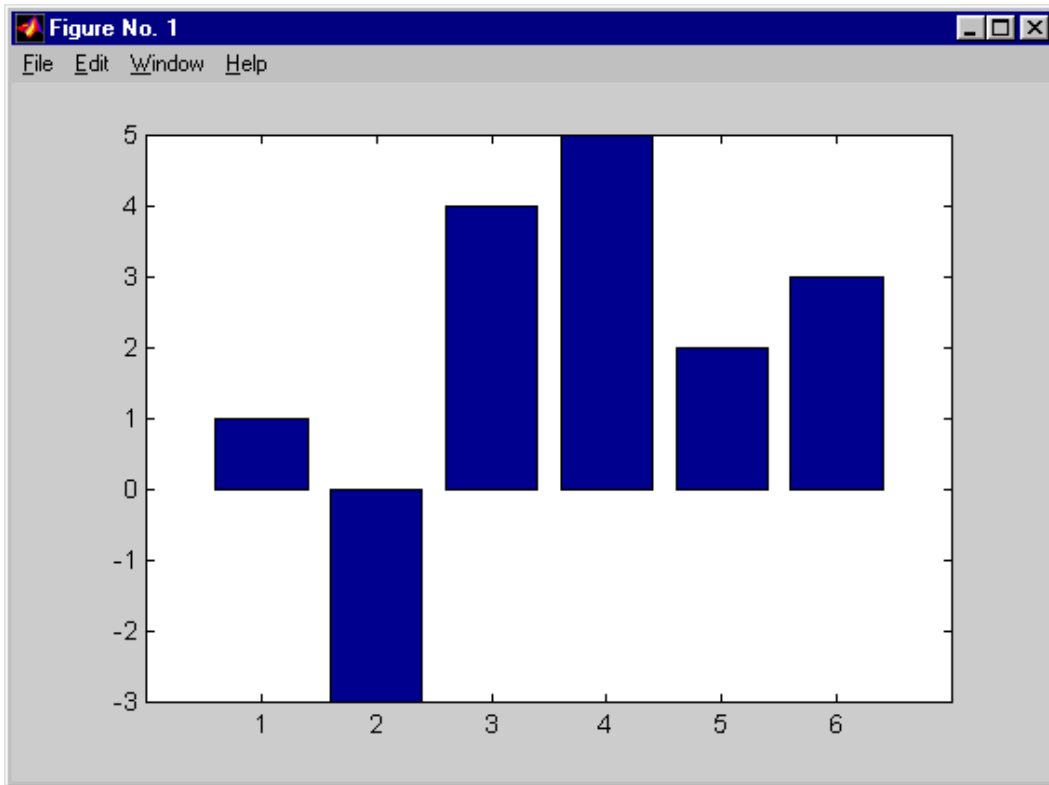


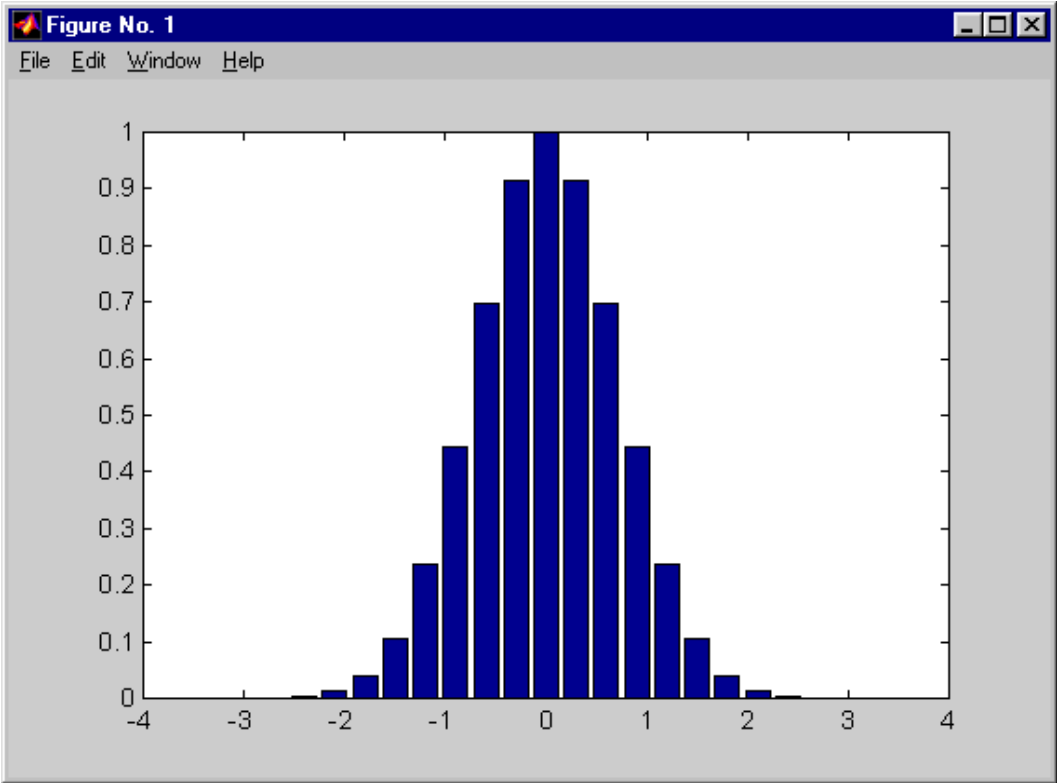
Figure 3-19

Below, is a bar chart for 20 values of a normal between - 3 and 3:

```
» x =-3:0.3:3;
```

```
» bar(x, exp(-x.^2))
```

Gets the graph in Figure 3-21:



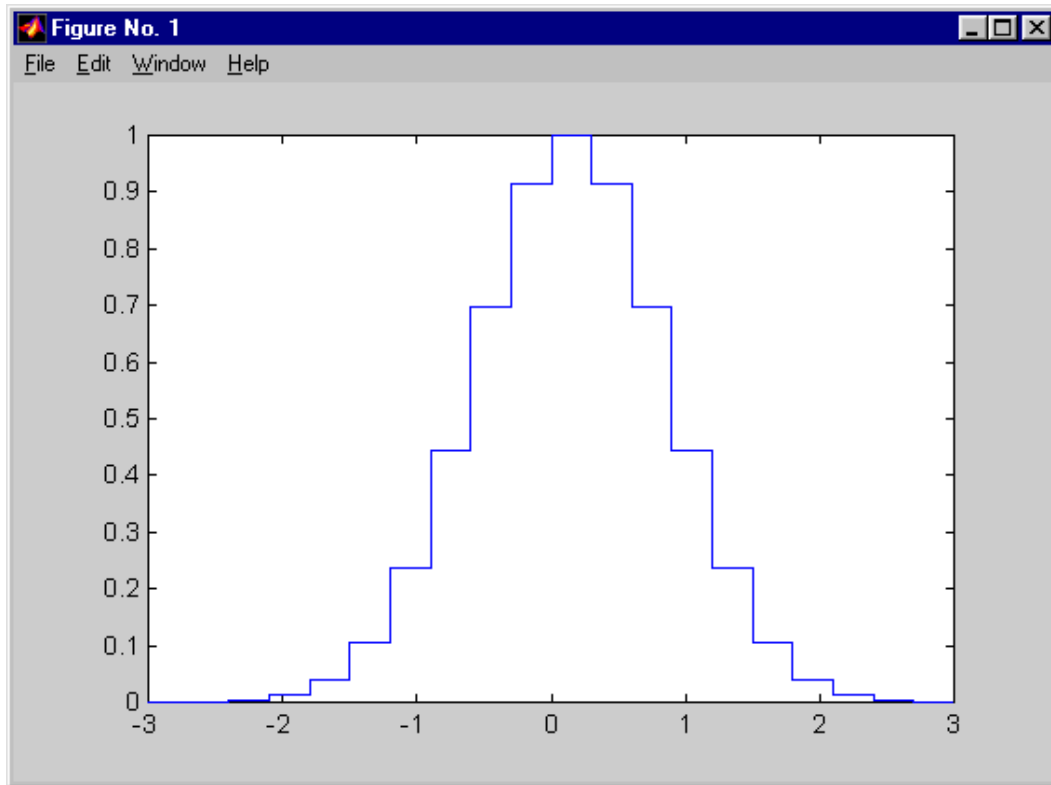


Figure 3-21

Figure 3-22

Figure 3-22 represents the step graph corresponding to the previous bar graph whose syntax is:

```
» x = -3:0.3:3;
```

```
» stairs(x,exp(-x.^2))
```

It is presented below, the histogram (see Figure 3-23) corresponding to a vector of 10,000 normal random values (0.1), 30 x between - 3 and 3 values:

```
» x = -3:0.1:3;
```

```
» y=randn(10000,1);
```

```
» hist(y,x)
```

In Figure 3-24 is a pie chart with two of its displaced areas, produced by using the syntax:

```
» pie([1, 3, 4, 5, 2, 3], [0,0,0,0,1,1])
```

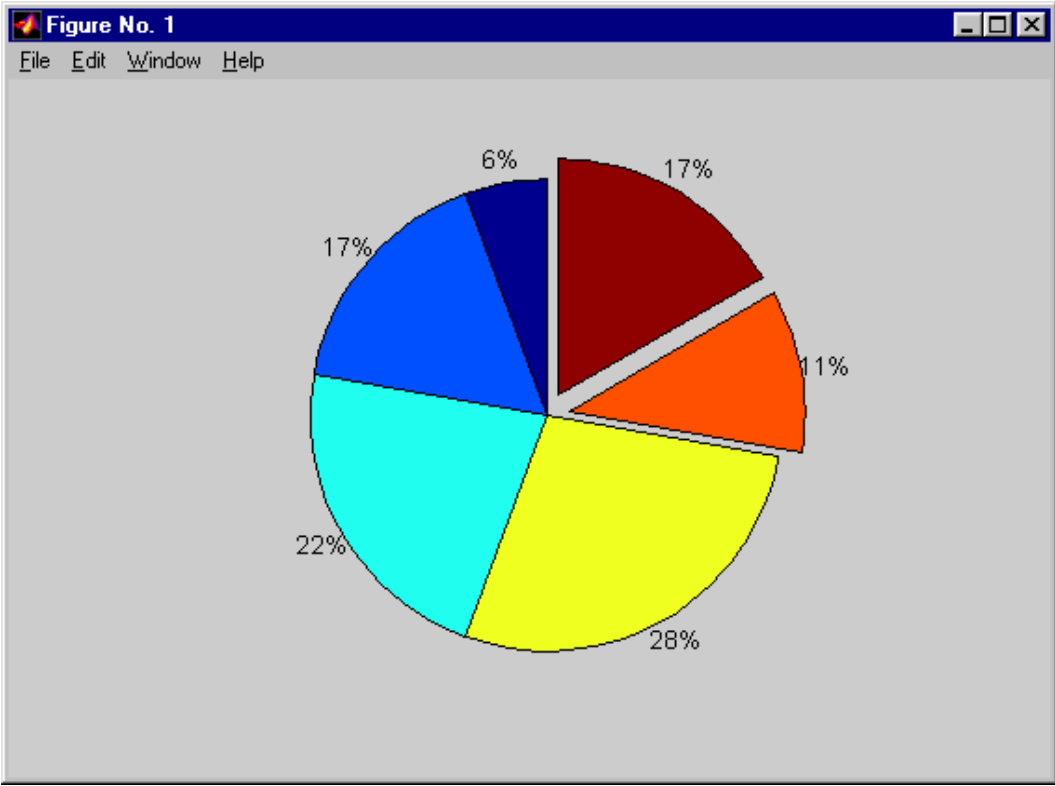
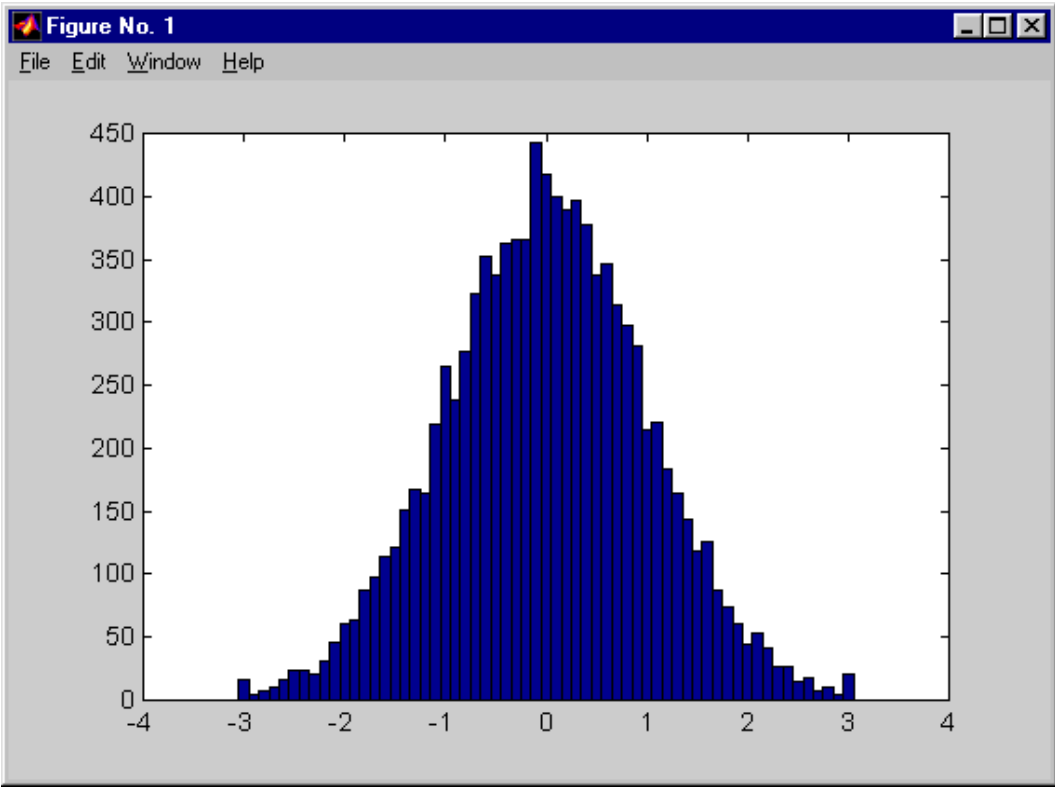


Figure 3-23

Figure 3-24

There are commands in Matlab which enable charting errors on a function, as well as certain types of graphics of arrows to be scanned now. Some of these commands are described below:

`errorbar(x,y,e)`

carries out the graph of the vector x against the vector and the errors specified in the vector e . Passing through each point (x_i, y_i) draws a vertical line length $2e_i$ whose centre is the point (x_i, y_i)

`stem (Y)`

draws the graph of the vector Y cluster. Each point and is attached to the axis x by a vertical line

`stem(X,Y)`

draw the graph of the vector Y cluster whose elements are given by the vector X

`rose(Y)`

draws the angular histogram relative to the vector Y angles in radians, using 20 equal radii

`rose(Y,n)`

draws on the vector and angular histogram, using equal radii

`rose(X,Y)`

draws the angle relative to the vector , and histogram using radios that are specified in the elements of the vector X

`compass (Z)`

carries out a diagram of arrows coming out of the origin and whose magnitude and direction are determined by the module and argument of the components of the vector Z complex numbers. Concerning the complex Z_i arrow joins the origin with the affix of Z_i

`compas(X,Y)`

is equivalent to `compas(X+i*Y)`

`compass (Z, S)` or `compass(X, Y, S)`

specifies in S line type to use on the arrows

`feather(Z)` o `feather(X,Y)` o `feather(Z,S)` o `feather(X,Y,S)`

is the same as `compass` , with the only difference that the origin of the arrows is not at the origin of coordinates, but that out of equally-spaced points of a horizontal line

`legend('leyenda1', 'leyenda2',..., 'leyendan')`

situated the legends given in n consecutive graphics

Here are some examples below:

First of all, let's represent in Figure 3-25 a chart of errors for the density of a normal distribution (0,1) function, with the variable defined in 40 points between - 4 and 4, errors being defined by 40 uniform random values (0.10):

```
» x = -4:0.2:4;
```

```
» y=(1/sqrt(2*pi))*exp(-(x.^2)/2);
```

```
» e=rand(size(x))/10;
```

```
» errorbar(x,y,e)
```

We also represent a graph of cluster corresponding to 50 normal random numbers (0.1) by using the syntax in Figure 3-26:

```
» y = randn(50,1); stem(y)
```

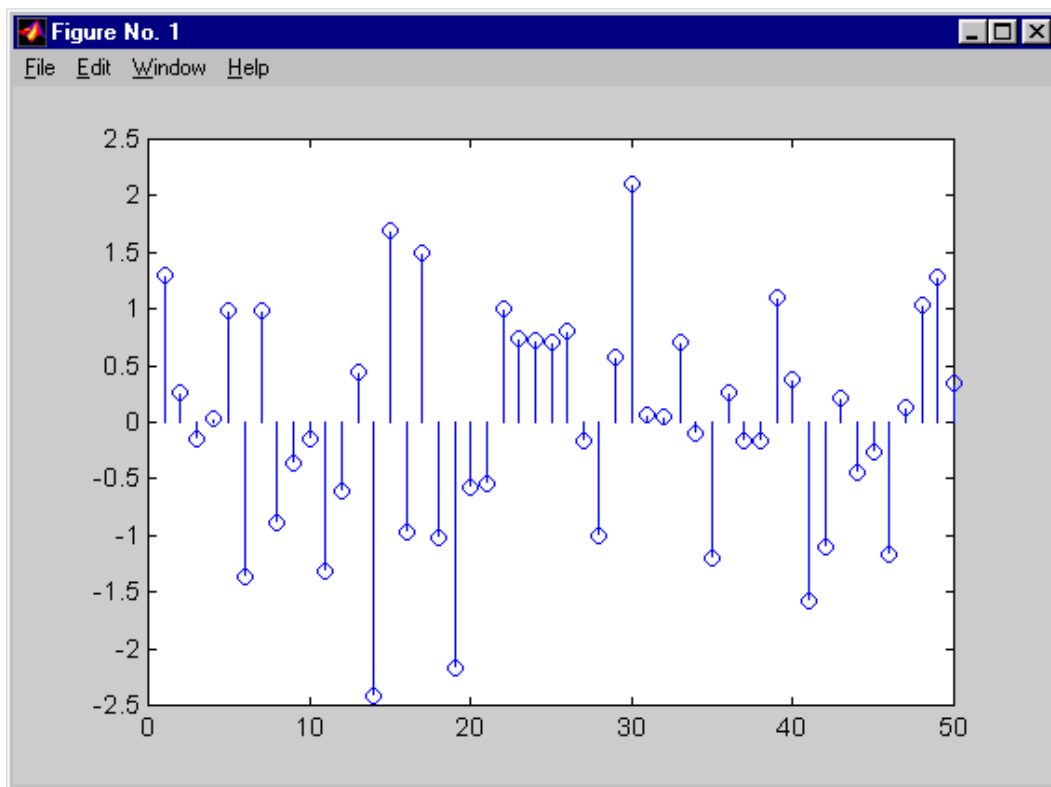
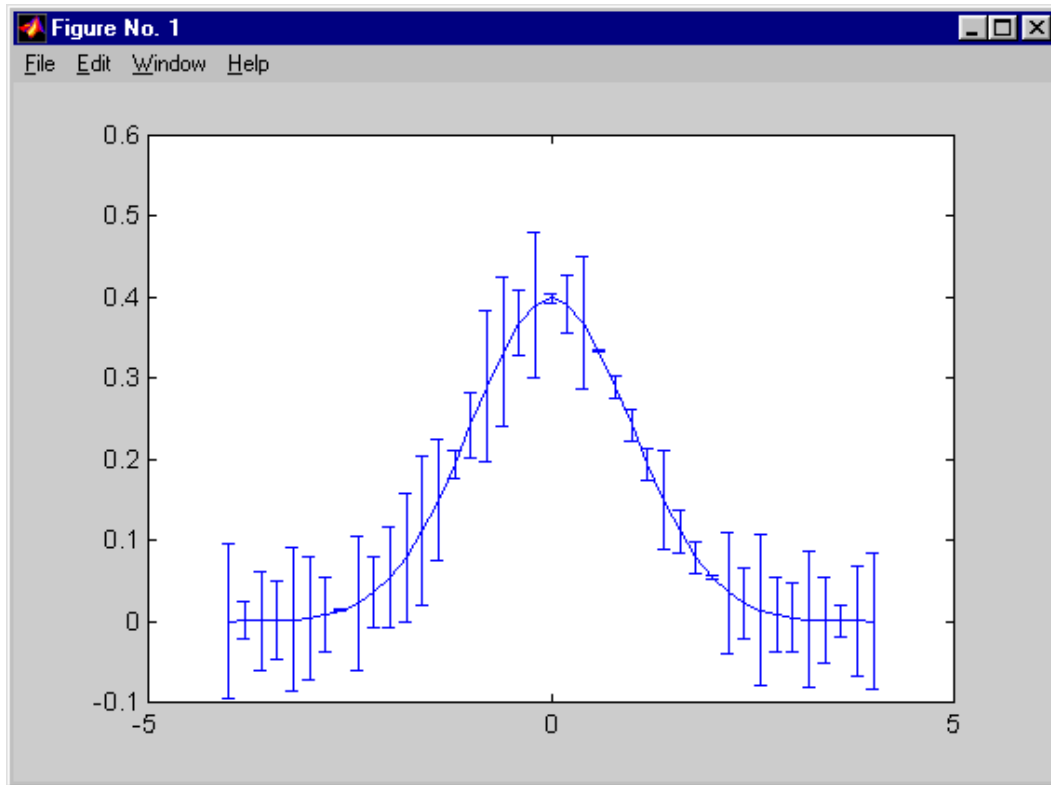



Figure 3-25

Now, presents a chart of arrows with center at the origin, corresponding to the eigenvalues of a normal (0,1) random square matrix of size 20 x 20 (see Figure 3-27). The syntax is as follows:

```
»z=eig(randn(20,20)); compass(z)
```

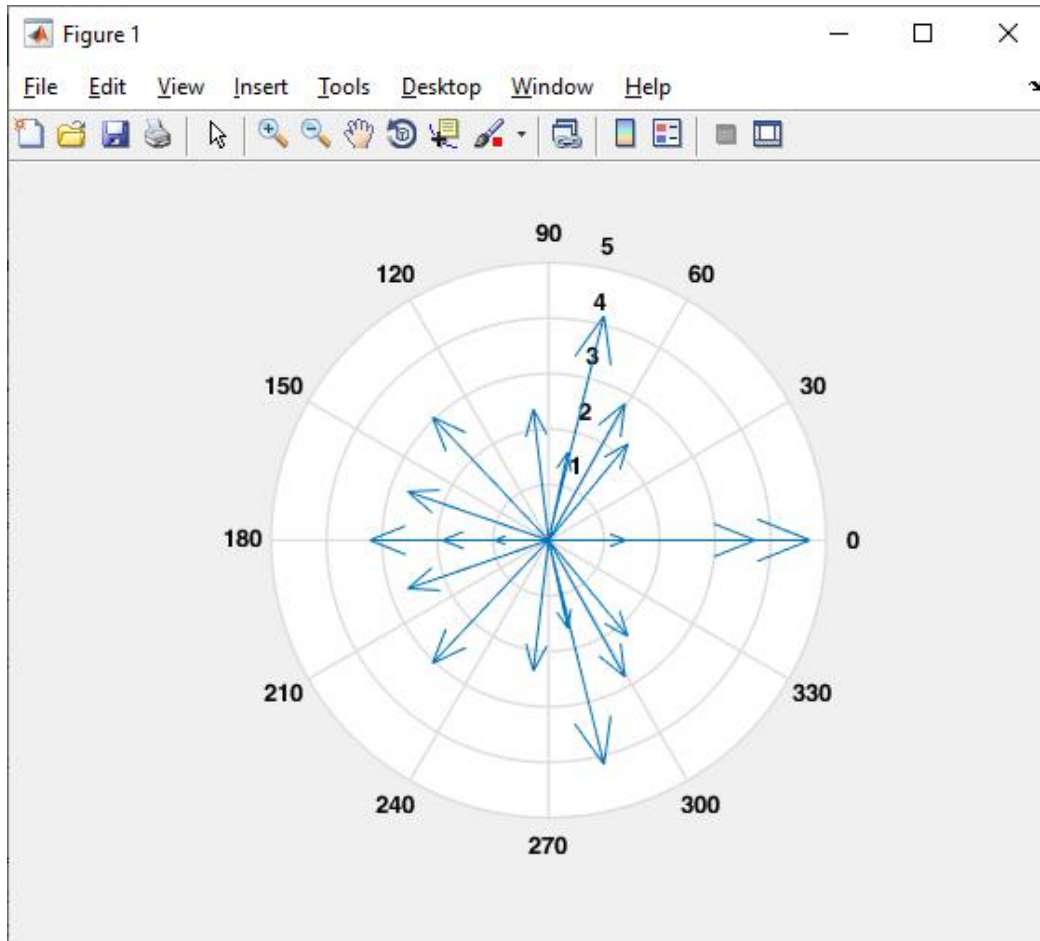


Figure 3-27

Chapter 4.

3-D Graphics IN MATLAB. 3-D curves ON SPACE, SURFACES, meshes AND contours. surfaces of revolution

The basic commands that Matlab uses to draw graphs that generate a line in three dimensions are the following:

```
plot3 (X, Y, Z)
```

draws the set of points (X, Y, Z) , where X , Y and Z are vectors row. X , Y and Z can be arrays of the same size, in which case a graph is made for each triplet of rows and on the same axis. For complex values of X , Y and Z are ignored the imaginary parts

`plot3(X,Y,Z,S)`

draws plot (X, Y, Z) with the settings defined in S . Usually S consists of two-digit quotes simple, the first of which sets the color of the line of the graph, and the second sets the character to be used in the plotting. The possible values of colors and characters have been already described to explain the command `plot`

`plot3(X1,Y1,Z1,S1,X2,Y2,Z2,S2,X3,Y3,Z3,S3,...)`

combined, on the same axes, graphs defined for the triples (X_i, Y_i, Z_i, S_i) . It is a way of representing various functions on the same graph

Here is an example:

» `x =0:pi/50:10*pi;`

» `y=sin(x);`

» `z=cos(x);`

» `plot3(x,y,z)`

Gets the graph in Figure 4-1:

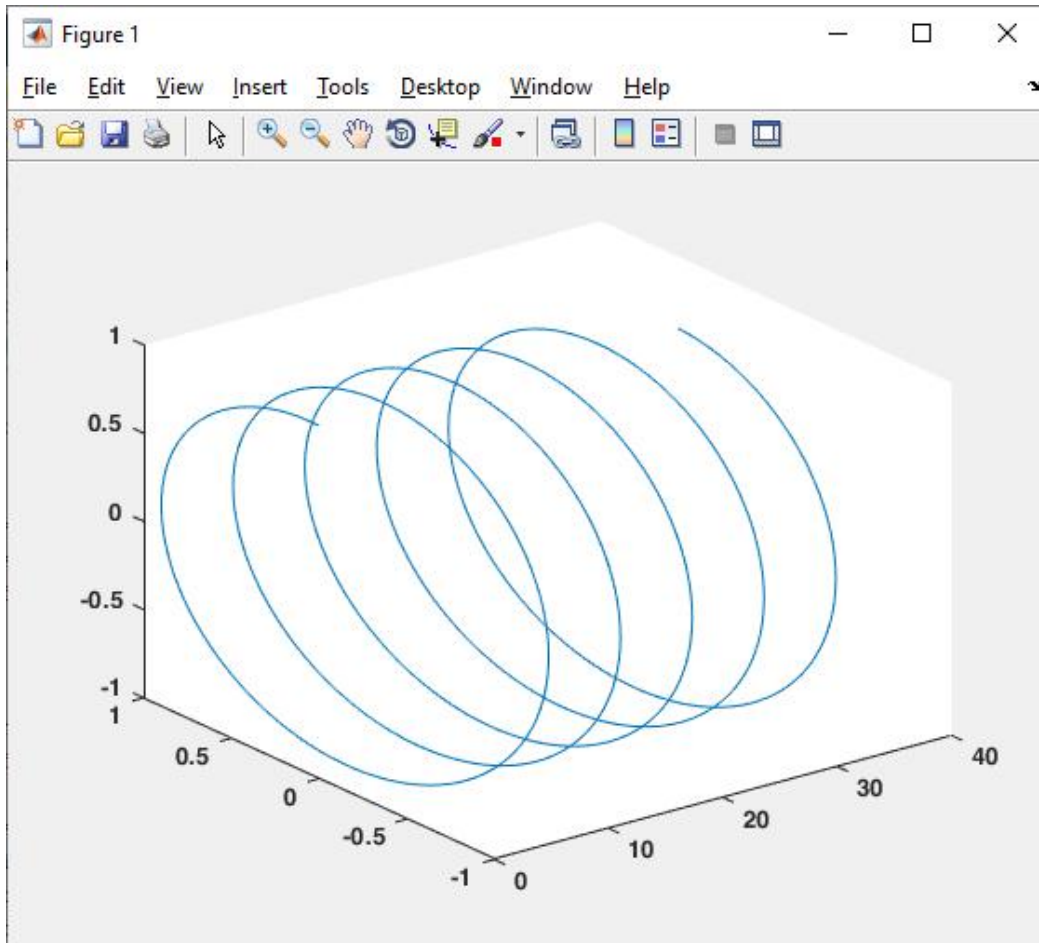


Figure 4-1

MATLAB also allows drawing polygons in three dimensions. To do this, use the following commands:

`fill3(X,Y,Z,C)`

draws the compact polygon whose vertices are the triples of components (X_i, Y_i, Z_i) of the column vectors X , Y and Z . C is a vector of the same size of X , Y and Z , which contains the colors of each point (X_i, Y_i, Z_i, C_i) . C_i values can be 'r', 'g', 'b', 'c', 'y', 'w', 'k', whose meanings we already know. If C is a single character, will be painted all the points of the polygon the color corresponding to the character. If X , Y and Z are of the same dimension, be represented matrices simultaneously several polygons corresponding to each triple vector column $(X.j, Y.j, Z.j)$. In this case, C can be a vector row C_j elements determine the unique color of each polygon corresponds to the triple of vector column $(X.j, Y.j, Z.j)$. C may

also be an array of the same dimension as X , Y and Z , in which case the elements determine the colors of each point (X_{ijk}, Y_{ijk}, Z_{ijk}) of the set of polygons.

```
fill3(X1,Y1,Z1,C1,X2,Y2, Z2, C2,...)
```

draws the compact polygon whose vertices are given by the points (X_i, Y_i, Z_i, C_i)

Here is an example:

```
» x =cos(0:0.01:8*pi);
```

```
»y=sin(0:0.01:8*pi);
```

```
»z=0:0.01:8*pi;
```

```
»fill3(x,y,z,'r')
```

Gets the graph of Figure 4-2:

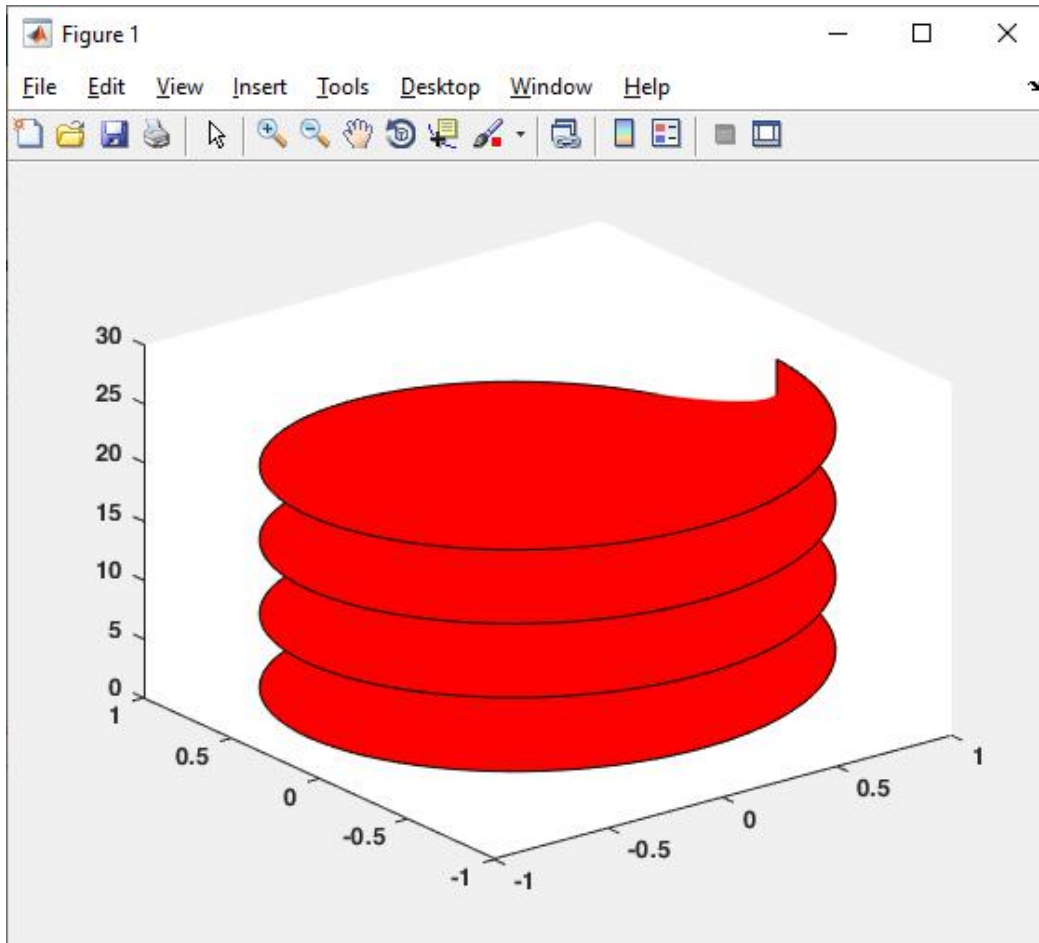


Figure 4-2

Let's see here how Matlab draws curves in parametric coordinates in space . The fundamental problem is to get graphs of functions tridimensional in which the variable x , y and z depend, in turn, of a parameter t .

The command that must be used is `plot3` and all its variants, suitably defining intervals of variation of the parameter, not the independent variable, as it was until now.

This type of graphics is very useful in certain matters, such as, for example, differential geometry.

Here is an example (Figure 4-3):

```
» t = 0:pi /50:10*pi; plot3(sin(t),cos(t),t);
```

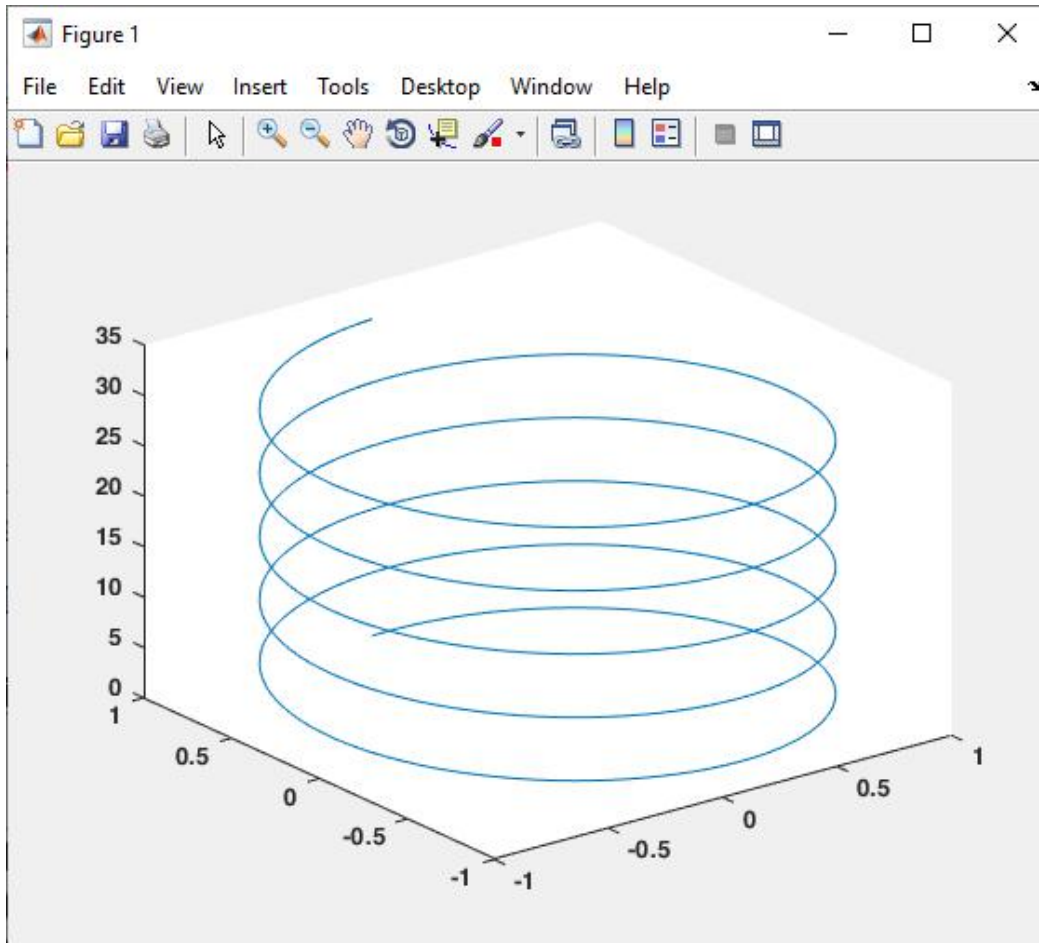


Figure 4-3

Below is another example:

```
»t=-4pi:0.01:4pi;
```

```
» x =cos(t).^2;
```

```
»y=sin(t).*cos(t);
```

```
»z=sin(t);
```

```
»plot3(x,y,z)
```

Gets the graph in Figure 4-4:

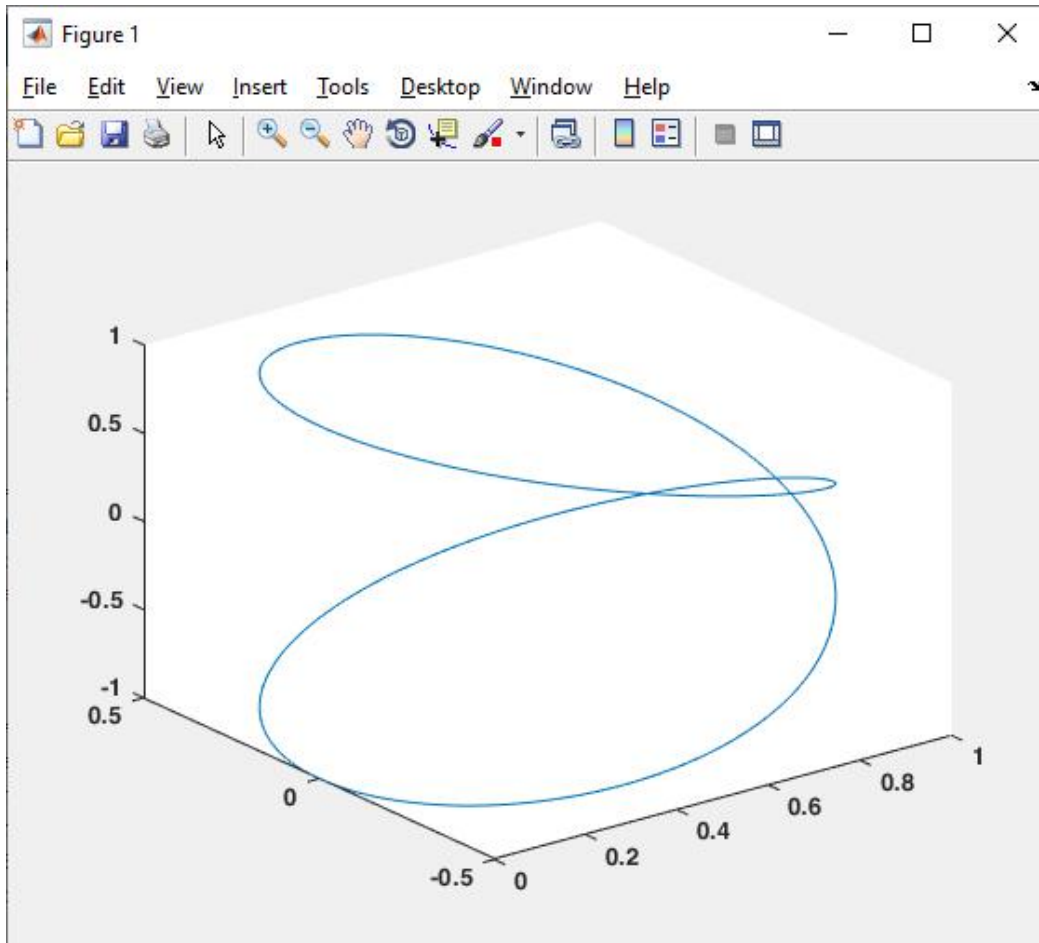


Figure 4-4

Surface graphics allow for dense representations of figures tridimensionales, and in particular of functions of two variables. The first step to represent a function of two variables $z = f(x,y)$ using his surface chart, is to use the command `meshgrid` , which basically defines the array of points (X, and) on which the function of two variables is evaluated for its graphical representation. Its syntax is as follows:

`[X, Y] = meshgrid(x,y)`

transforms definition field variable x given e and function to represent $z = f(x,y)$ in matrix arguments can be used by the command `mesh` for mesh chart

The second step is to use the available commands to effect, which are as follows:

`surf(X,Y,Z,C)`

represents the graph of function surface $z = f(x,y)$, doing the drawing with the colors specified in C . The C argument can be ignored

`surfc(X,Y,Z,C)`

represents the graph of function surface $z = f(x,y)$ with the chart's corresponding outline (contour lines projected onto the XY -plane)

`surf(X, Y, Z)`

represents the graph of function surface $z = f(x,y)$, making the drawing with shading

Exercise 4-1. Represent the surface of the slope-intercept form:

$$-14/2 < x, y < 14/2$$

Also represent the surface with its contour.

» `[X, Y] = meshgrid(-7.5:0.5:7.5);`

» `Z = sin(sqrt(X.^2+Y.^2))./sqrt(X.^2+Y.^2);`

» `surf(X, Y, Z)`

Gets the graph of Figure 4-5:

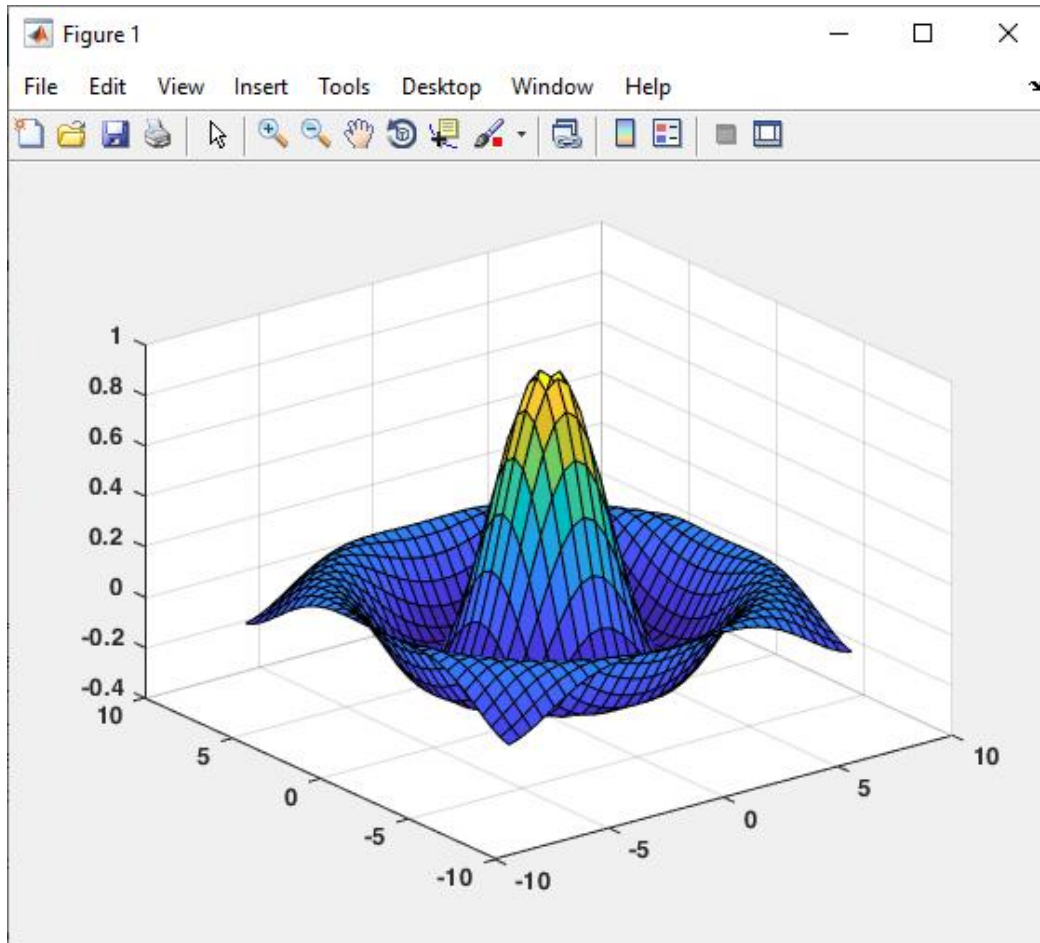


Figure 4-5

The surface with the outline (contour) graph is shown in Figure 4-6. The following syntax is used:

- » `[X, Y] = meshgrid(-7.5:0.5:7.5);`
- » `Z = sin(sqrt(X.^2+Y.^2))./sqrt(X.^2+Y.^2);`
- » `surf(X, Y, Z)`

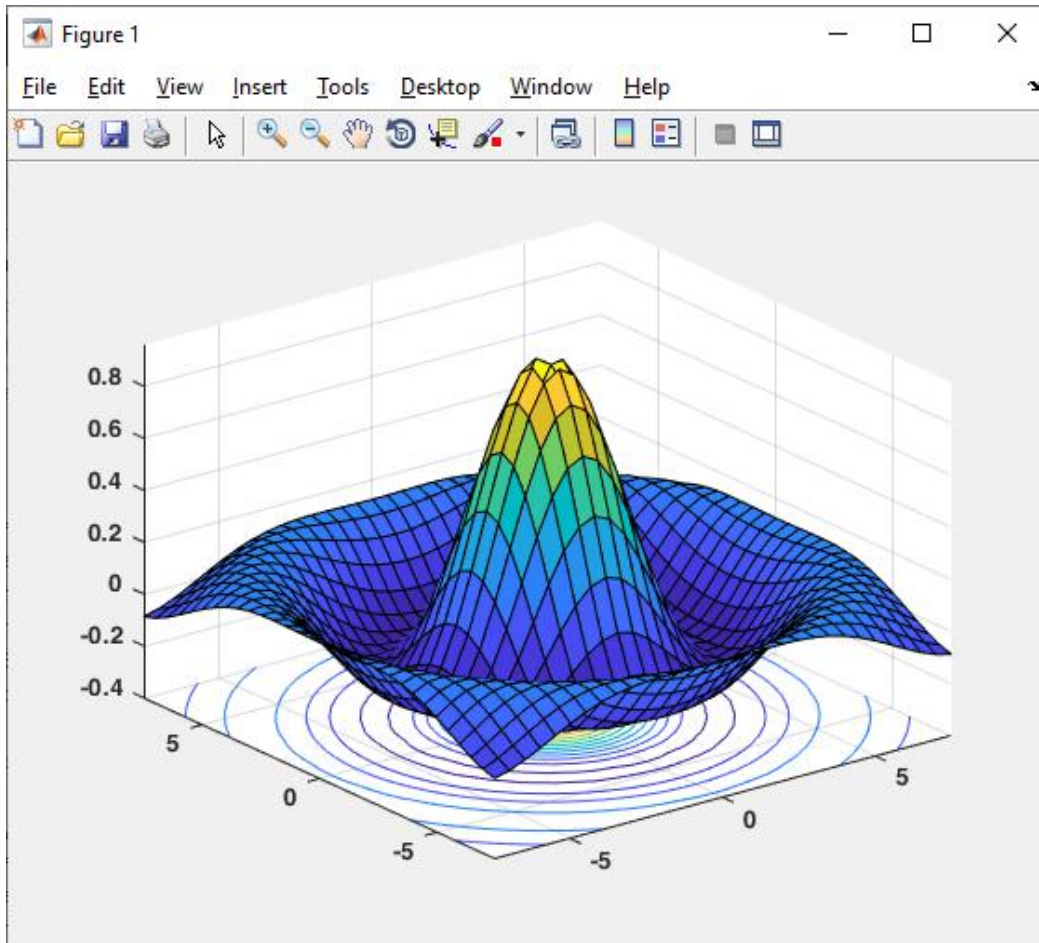


Figure 4-6

Figure 4-7 shows the chart shaded, using the syntax:

- » `[X, Y] = meshgrid(-7.5:0.5:7.5);`
- » `Z = sin(sqrt(X.^2+Y.^2))./sqrt(X.^2+Y.^2);`
- » `surf(X, Y, Z), shading interp`

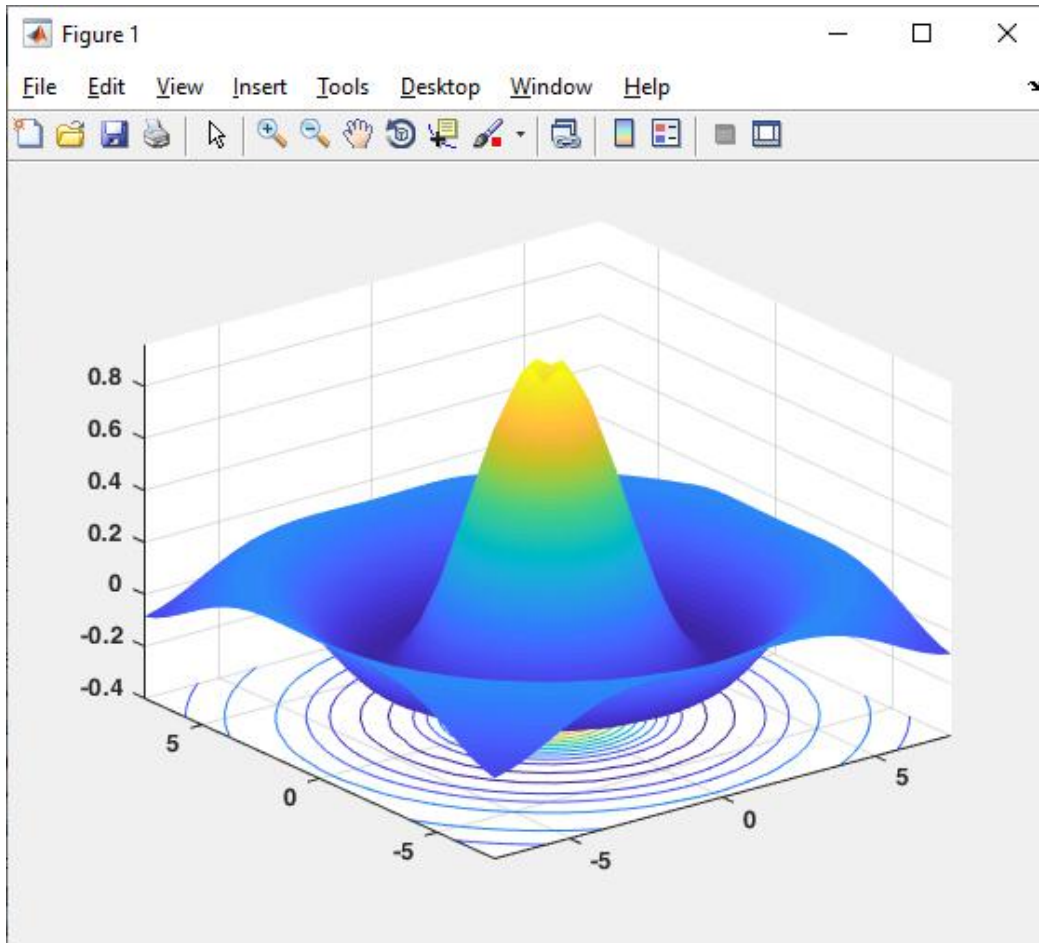


Figure 4-7

A three-dimensional graph of mesh is defined by a function $z = f(x,y)$, so that the points on the surface are represented on a grid, result of rise z values given by $f(x,y)$ on corresponding points of the plane $(x, \text{ and } y)$. The appearance of a mesh chart is like a fishing net, with points on the surface on the nodes of the network. Actually, it is a graph of surface whose graph has form of network.

To represent a graph of mesh, use the command `mesh` and its variants, whose syntax is as follows:

`mesh(X,Y,Z,C)`

represents the graph of the function $z = f(x,y)$, drawing the grid lines that compose the mesh with the colors specified in C . The C argument can be ignored

`meshz(X,Y,Z,C)`

represents the graph of the function $z = f(x,y)$ with a kind of curtain or curtain at the bottom

`meshc(X,Y,Z,C)`

represents the graph of the function $z = f(x,y)$ along with corresponding contour chart (contour lines projected onto the XY -plane)

Exercise 4-2. Represent the graph of mesh for the surface of equation:

$$z = x e^{-x^2 - y^2} - 2 < x, y < 2$$

Also be representation with their contours (contour chart) and representation with curtain.

The syntax presented here gives as a result the graph in Figure 4-8:

» `[X, Y] = meshgrid(-2:0.1:2,-2:0.1:2);`

» `Z = X.*exp(-X.^2- Y.^2);`

» `mesh(X, Y, Z)`

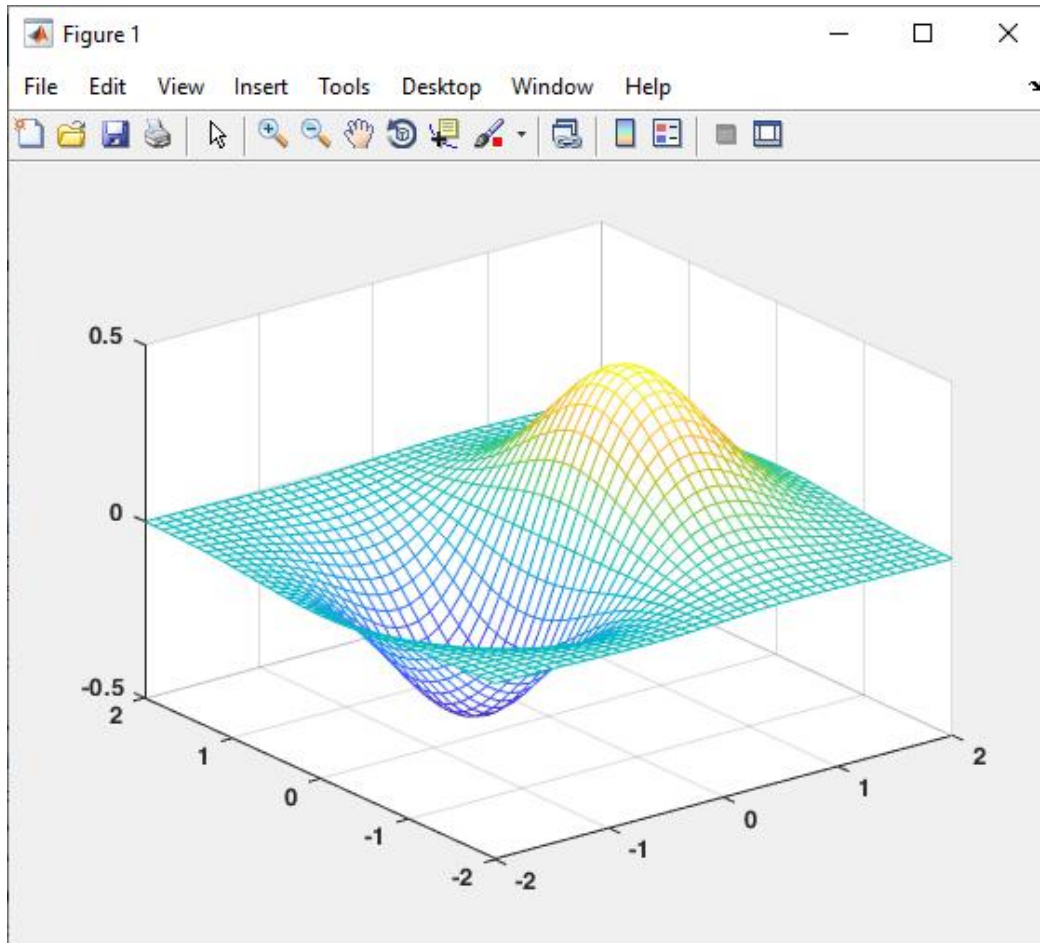


Figure 4-8

Figure 4-9 presents the mesh along with the contour graph (or contour chart). The syntax is as follows:

```
» [X, Y] = meshgrid(-2:0.1:2,-2:0.1:2);
```

```
» Z = x*exp(-X.^2- Y .^2);
```

```
» meshc(X, Y, Z)
```

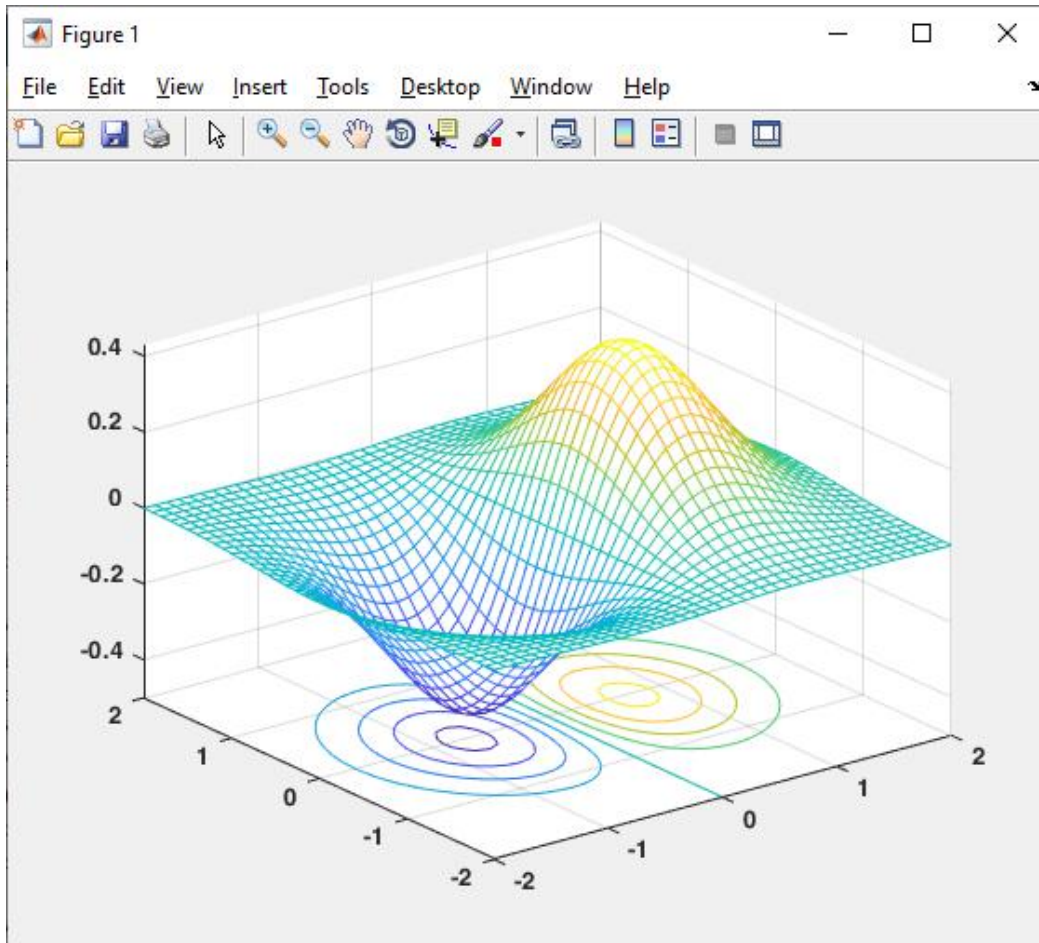


Figure 4-9

Finally, in Figure 4-10 represent graphic mesh curtain or curtain lower option. The syntax is as follows:

```
» [X, Y] = meshgrid(-2:0.1:2,-2:0.1:2);
```

```
» Z = x*exp(-X.^2- Y .^2);
```

```
» meshz(X, Y, Z)
```

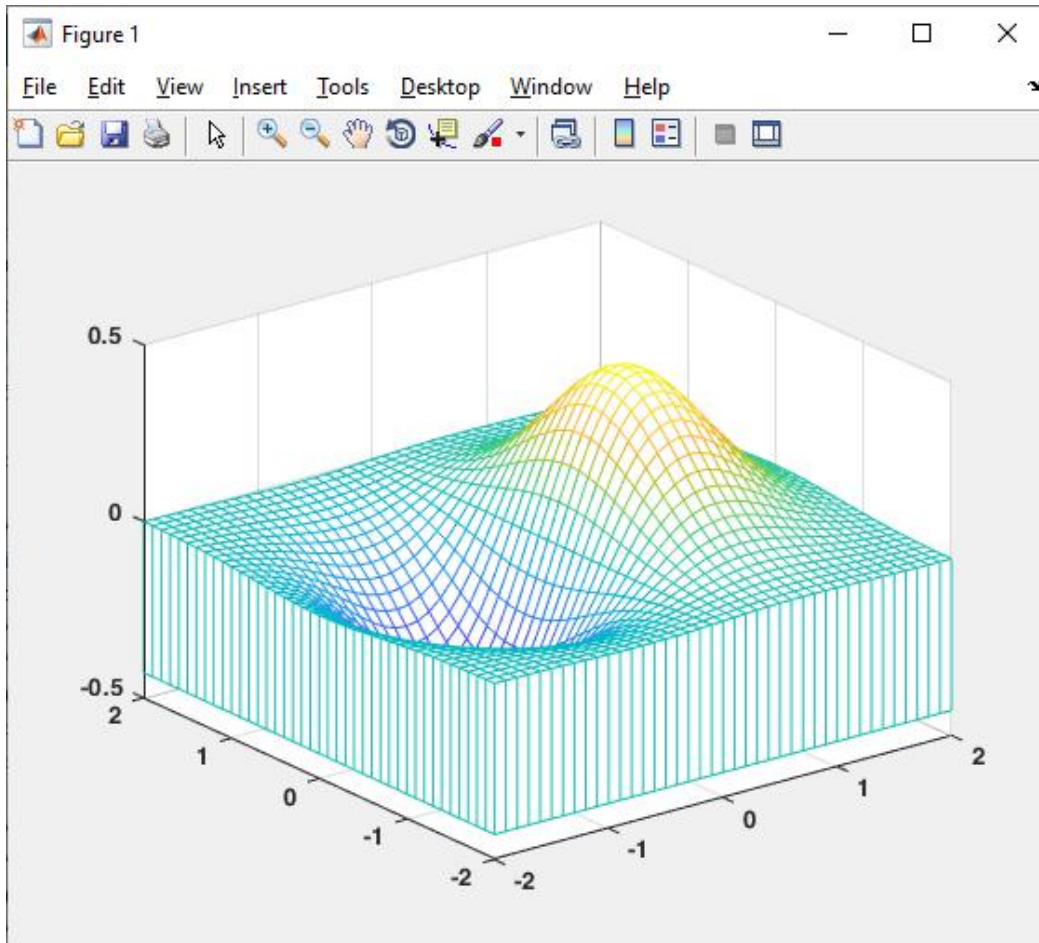



Figure 4-10

Another option of visualized a function of two variables is to use level curves calls or the system of dimensional planes. These curves are characterized because they are points (x, and) on which the value of $f(x,y)$ is constant. Thus, for example, in the weather charts, level curves that represent the same temperature points are called isotherms, and contour lines of equal pressure, isobars. Using contour lines, representing heights (values of $f(x,y)$) equal, can describe surfaces in space. Thus, drawing different contour corresponding to constant heights, can be described a map of lines on the surface level, Matlab called contour graph . The contour plots can be represented in two and three dimensions.

A map showing the regions of the Earth's surface, whose contour lines represent the height above the sea level, is called topographic map. In these maps is observed, therefore, the variance of $z = f(x,y)$ with respect to x and y . When the space between contour lines is large, it means

that the variation of the variable z is slow, while a small space indicates a rapid change of z .

Commands used Matlab for the representation of graphics outline (contour lines) are as follows:

`contour (Z)`

draws outline (contour lines) for the Z matrix graph. The number of contour lines to be used is chosen automatically

`contour(Z,n)`

draws the graph of outline (contour lines) for the Z matrix using contour lines

`contour (x, y, Z, n)`

draws the graph of outline (contour lines) for the Z matrix in the X and Y axes using scaling defined by vectors x and y (n contour lines)

`contour3 (Z), contour3(Z,n) and contour3 (x, y, Z, n)`

draws the contour in 4-dimensional plots

`pcolor (X, Y, Z)`

draws a graph of outline (contour lines) to the matrix (X, Y, Z) using a representation based on densities of colors. It is often called density chart

Exercise 4-3. Given the surface of equation:

$$z = \sin(x) \sin(y) \quad -2 < x, y < 2$$

represent it with their contour. Represent its outline two-dimensional with 20 lines graph and its three-dimensional outline with 50 lines chart. Also draw the corresponding density chart.

Figure 4-11 shows the graph of the surface with its contour. The syntax is as follows:

» `[X, Y] = meshgrid(-2:0.1:2);`

» $Z = \sin(X) \cdot \sin(Y)$;

» `surf(X, Y, Z)`

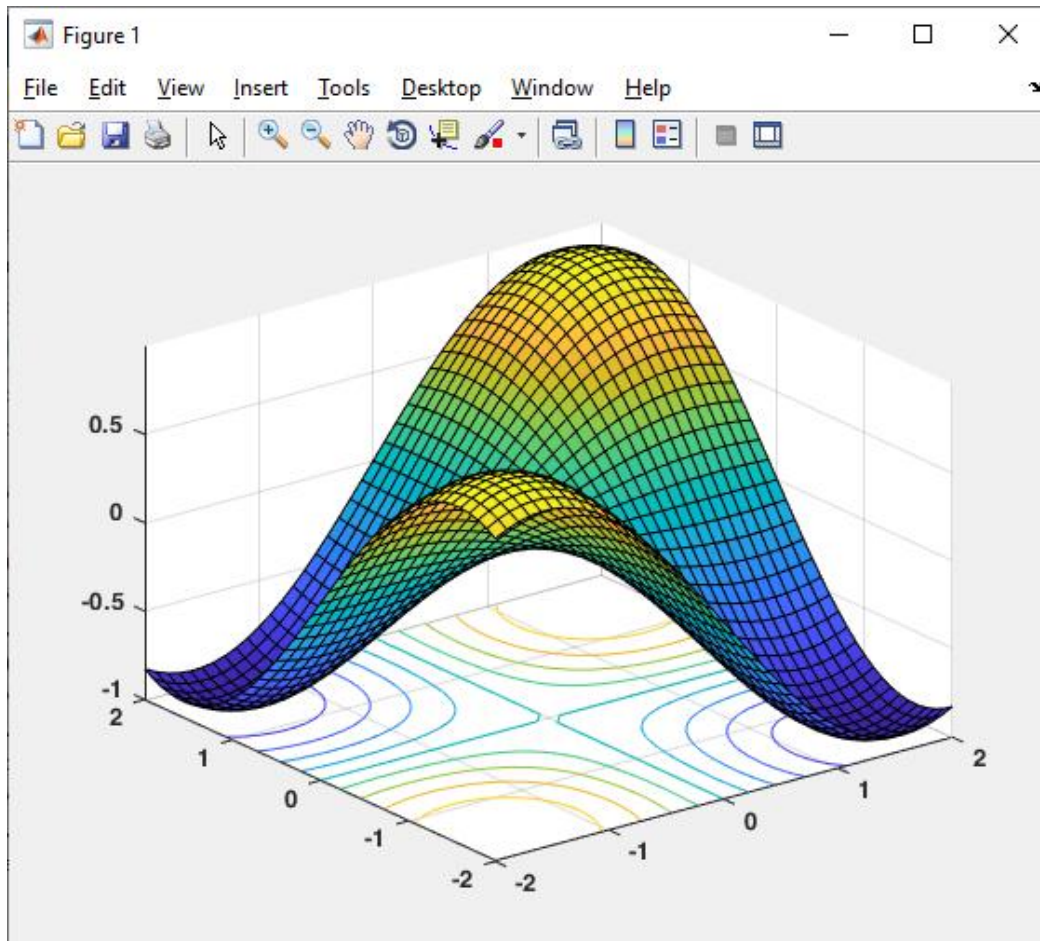


Figure 4-11

Figure 4-12 shows outline (contour) graph two-dimensional to 20 lines. The syntax is as follows:

» `[X, Y] = meshgrid(-2:0.1:2);`

» $Z = \sin(X) \cdot \sin(Y)$;

» `contour(Z, 20)`

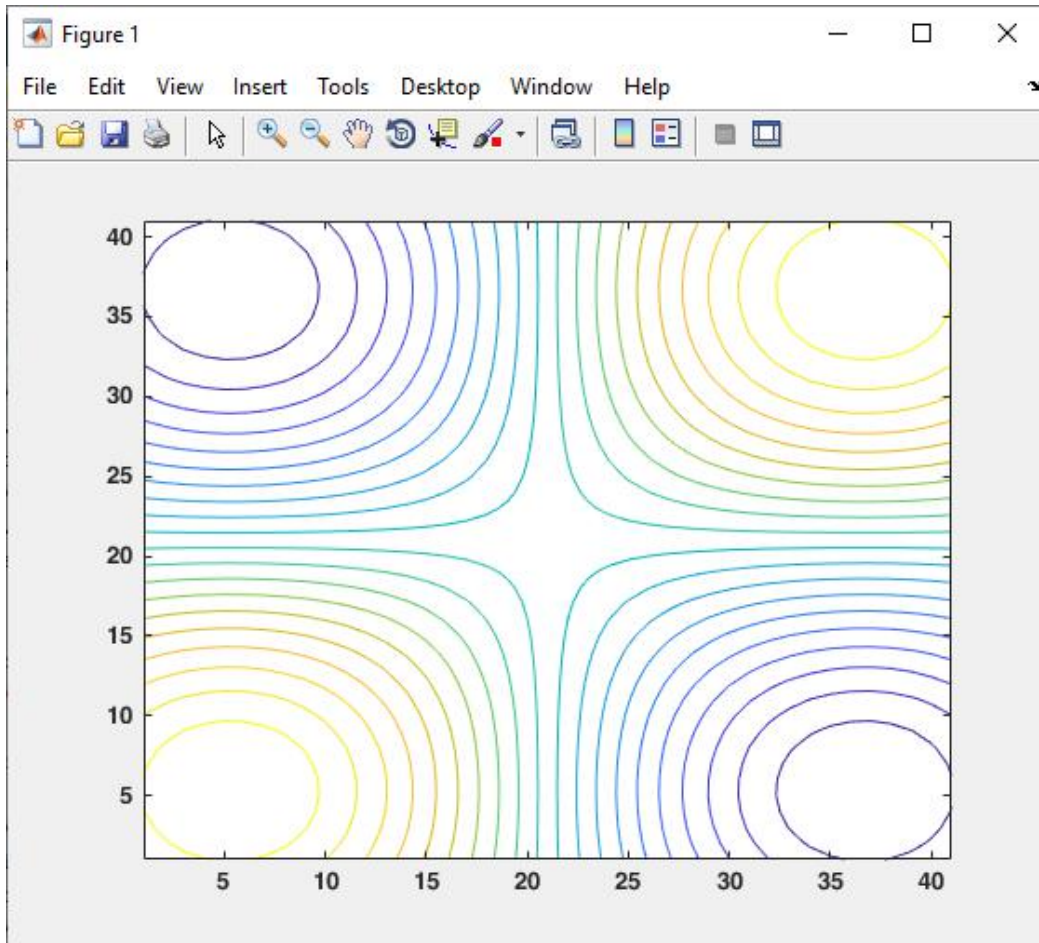


Figure 4-12

Figure 4-13 shows outline (contour) graphic three-dimensional to 50 lines. The syntax is as follows:

```
»[X,Y]=meshgrid(-2:0.1:2);
```

```
» Z =sin(X).*sin(Y);
```

```
»contour3(Z,50)
```

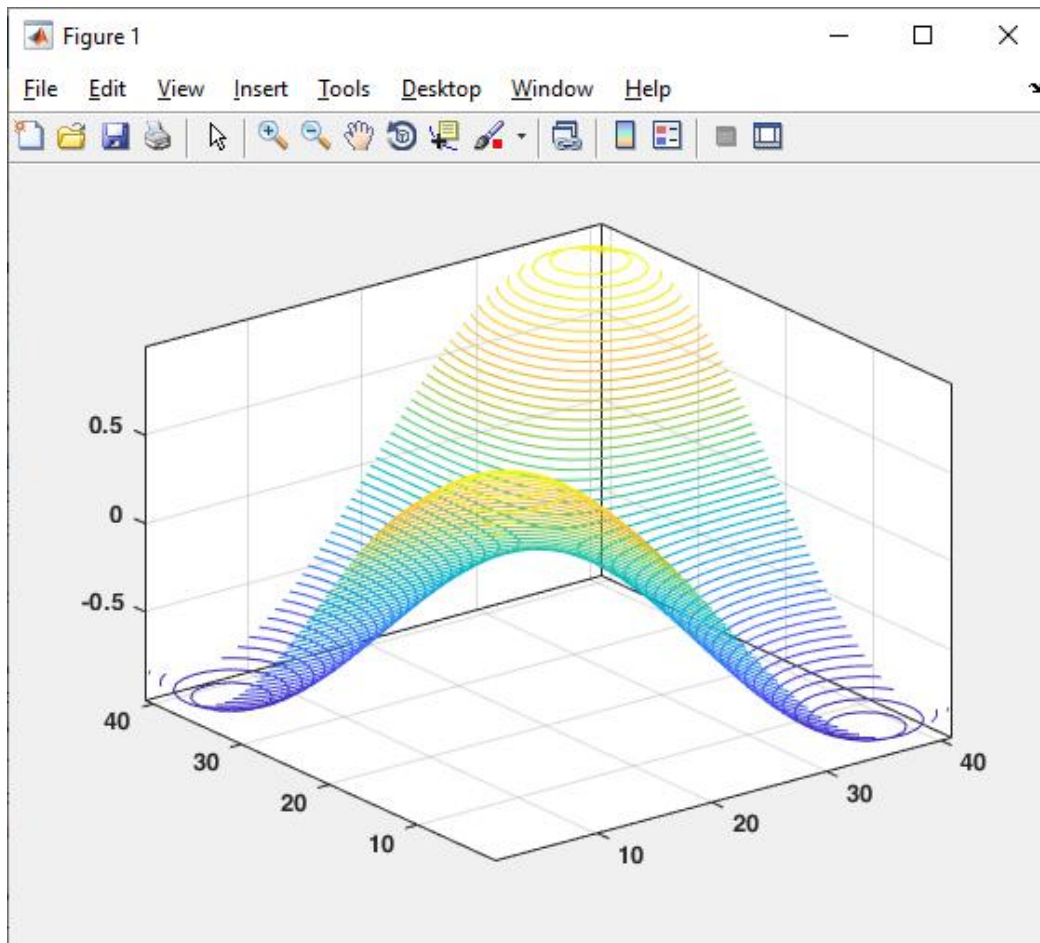


Figure 4-13

Finally, Figure 4-14 represents the graph of density (contour shaded according to different intensities of color). The syntax is as follows:

```
»[X,Y]=meshgrid(-2:0.1:2);
```

```
» Z =sin(X).*sin(Y);
```

```
»pcolor(X,Y,Z)
```

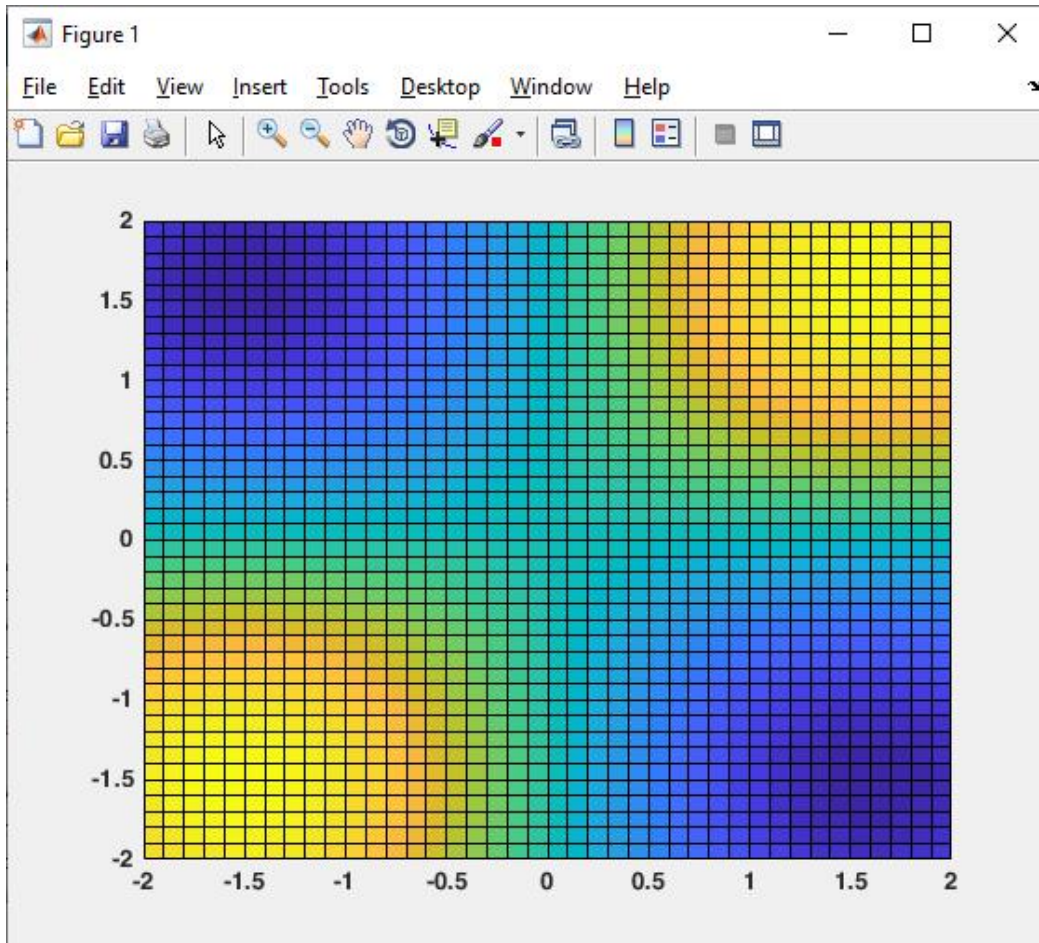


Figure 4-14

There are commands in Matlab which allow you to change the appearance of a same graph, either by your shader, the scale of its themes, colors, hidden lines, the point of view from which one can observe, etc. Below, some of these commands are:

`axis ([xmin ymin, ymax, zmin zmax max x])`

places intervals of variation of the axes in the indicated values. Also accepts the options 'ij' , 'square' , 'equal' , etc, identical to the already seen for two dimensions

`view ([x, y, z])`

places the point of view of the figure in the point's Cartesian coordinates (x, y, z)

`view([az, el])`

puts the angle of view of the figure in the point of azimuth (horizontal rotation) 'az' and elevation (vertical lift) 'el'

`hidden`

controls the presence of hidden lines in the graph. These lines come with `hidden on` and go with `hidden off`

`shading`

controls the type of shadow of a surface created with commands `surf`, `mesh`, `pcolor`, `fill` and `fill3`. Option `shading flat` situated a smooth shading option `shading interp` situated a dense shading and the option to `shading faceted` (default) puts a normal shader

`colormap (M)`

located the matrix `M` as the current color map. `M` must have three columns and only contain values between 0 and 1. It can also be a matrix whose rows are vectors RGB type `[r g b]`. All arrays have 3 columns and `p` rows.

`brighten(p)`

adjusts the lighting of the figure. If $0 < p < 1$, figure will be bright, and if $p < -1$, figure will be dark. The variation of `p` is the interval $(-1, 1)$, and to the values of `p` are approaching -1 , figure darkens, while as `p` values approaching 1, the figure illuminates

`image(A)`

produces a two-dimensional image with glitters proportional to the elements of the array `A`, and is used to display photographs and drawings adapted to specified in the parent to shines. Each element `(m, n)` of `A` matrix affects a cell of the drawing

`caxis([cmin cmax])`

places the minimum and maximum values of the color scale (defined by `colormap` and intrinsically related to the divisions that are made in the

axes via grids) for a chart. Therefore, it enables to use only a subset of colors defined by colormap to Figure .

caxis ([cmin cmax])

set minimum values and maximum color scale (defined by colormap and intrinsically linked to divisions making axes, via the grilles) for a chart. Therefore, it allows using only a subset of the defined colors for the colormap for the figure.

Exercise 4-4. Given the surface of equation:

$$z = x^2 - y^2 - 2 \quad -2 < x, y < 2$$

represent it with strong lighting, dense shadows and gray colors. Represent about the same axis curve focused from four different points of view and with shading by default.

```
» [X, Y] = meshgrid(-2:0.05:2);
```

```
» Z=X.^2-Y.^2;
```

```
» surf(X,Y,Z),shading interp,brighten(0.75),colormap(gray(5))
```

Gets the graph of Figure 4-15:

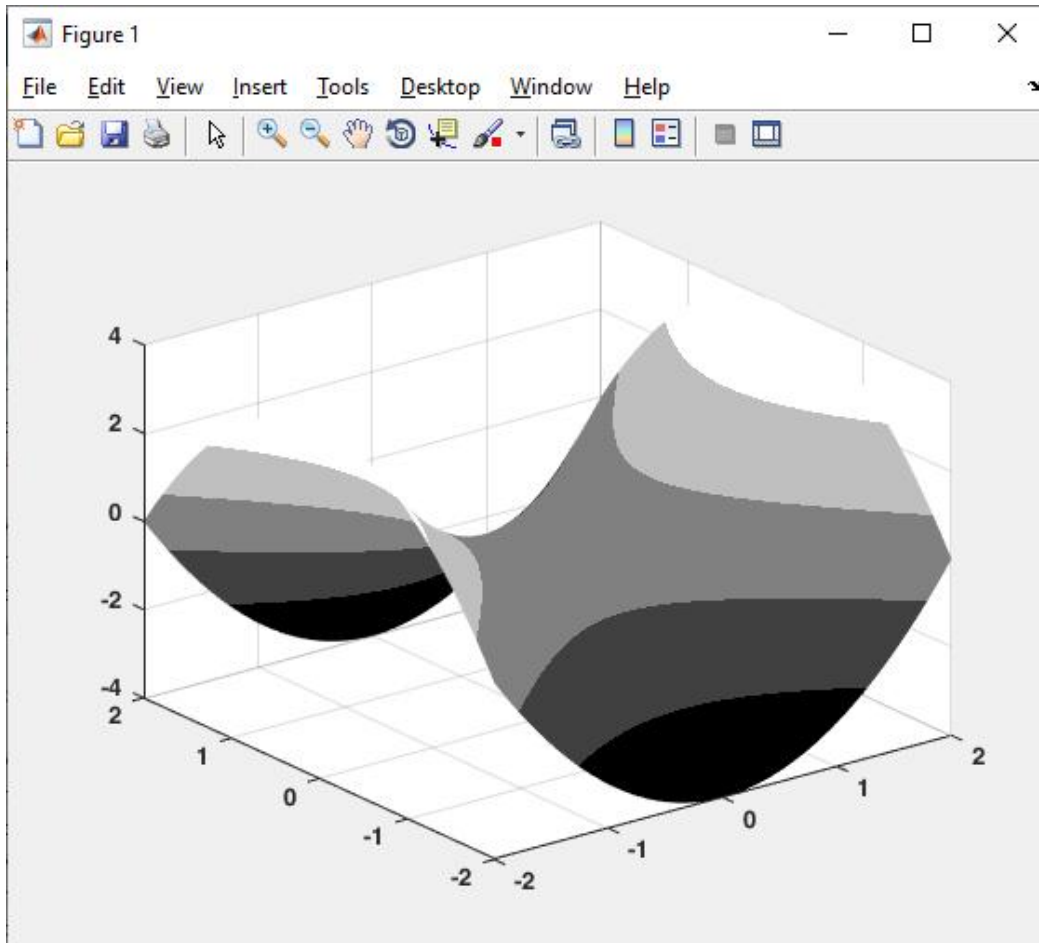


Figure 4-15

We will now present in Figure 4-16 surface focused from four different points of view. The syntax will be as follows:

» `[X, Y] = meshgrid(-2:0.05:2);`

» `Z = X.^2-Y.^2;`

» `subplot(2,2,1)`

» `surf(X,Y,Z)`

» `subplot(2,2,2)`

» `surf(X,Y,Z),view(-90,0)`

» `subplot(2,2,3)`

- » surf(X,Y,Z),view(60,30)
- » subplot(2,2,4)
- » surf(X, Y, Z), view(- 10, 30)

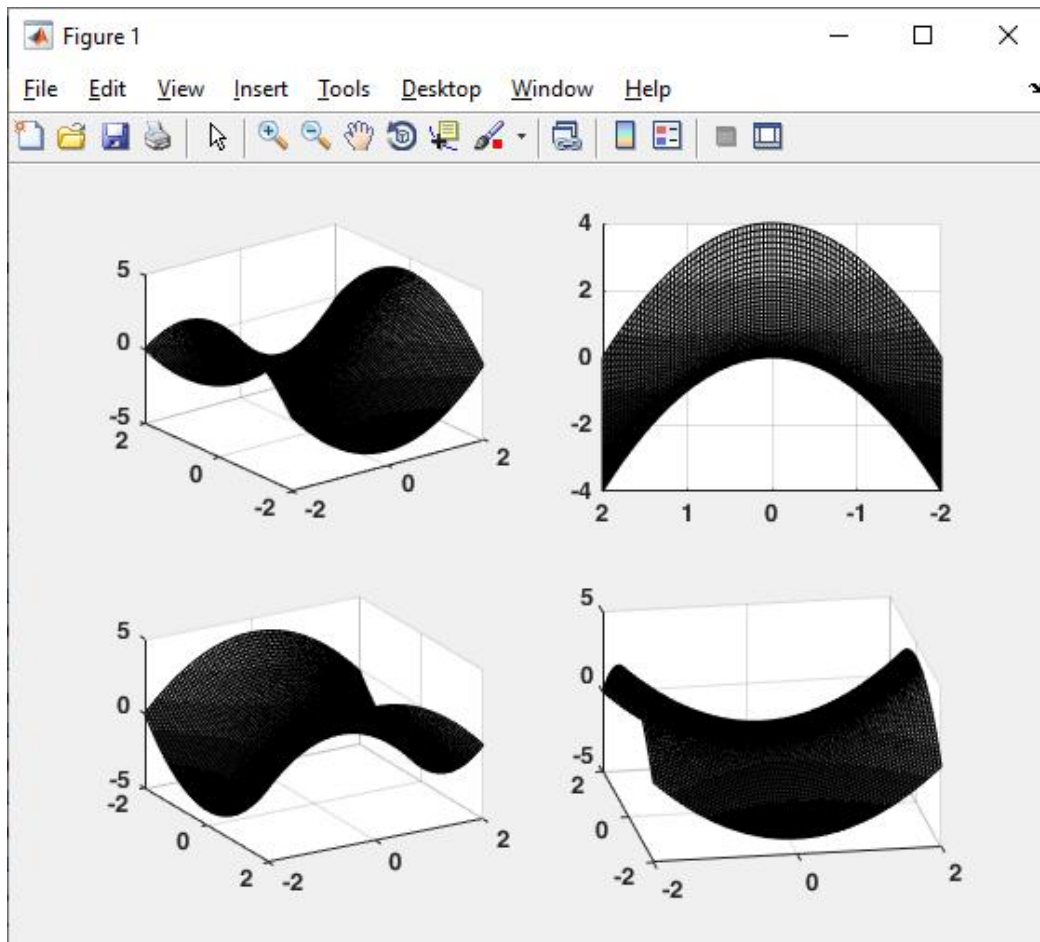


Figure 4-16

MATLAB allows you to represent surfaces whose components depend on specified variation parameter. To do this, you can use the commands surf and mesh , by properly defining the variable x, and z .

Implied, cylindrical and spherical coordinate surfaces are repre-tables in Matlab Parameterizing them previously.

In terms of surfaces of revolution, they are always ranges, which also allows its graphical representation with Matlab.

Exercise 4-5. Draw the surface of parametric coordinates:

$$x = 4\cos(r)\sec(t) \quad y = 2\sin(r)\sec(t) \quad z = \tan(t) \quad -2 < r < 2, \quad -\pi < t < \pi$$

```
» r=(-2*pi:0.1:2*pi)';
```

```
» t=(-pi:0.1:pi);
```

```
» X =4*cos(r)*sec(t);
```

```
» Y =2*sin(r)*sec(t);
```

```
» Z =ones(1,length(r))*tan(t);
```

```
» surf(X,Y,Z),shading interp
```

Gets the graph of Figure 4-17:

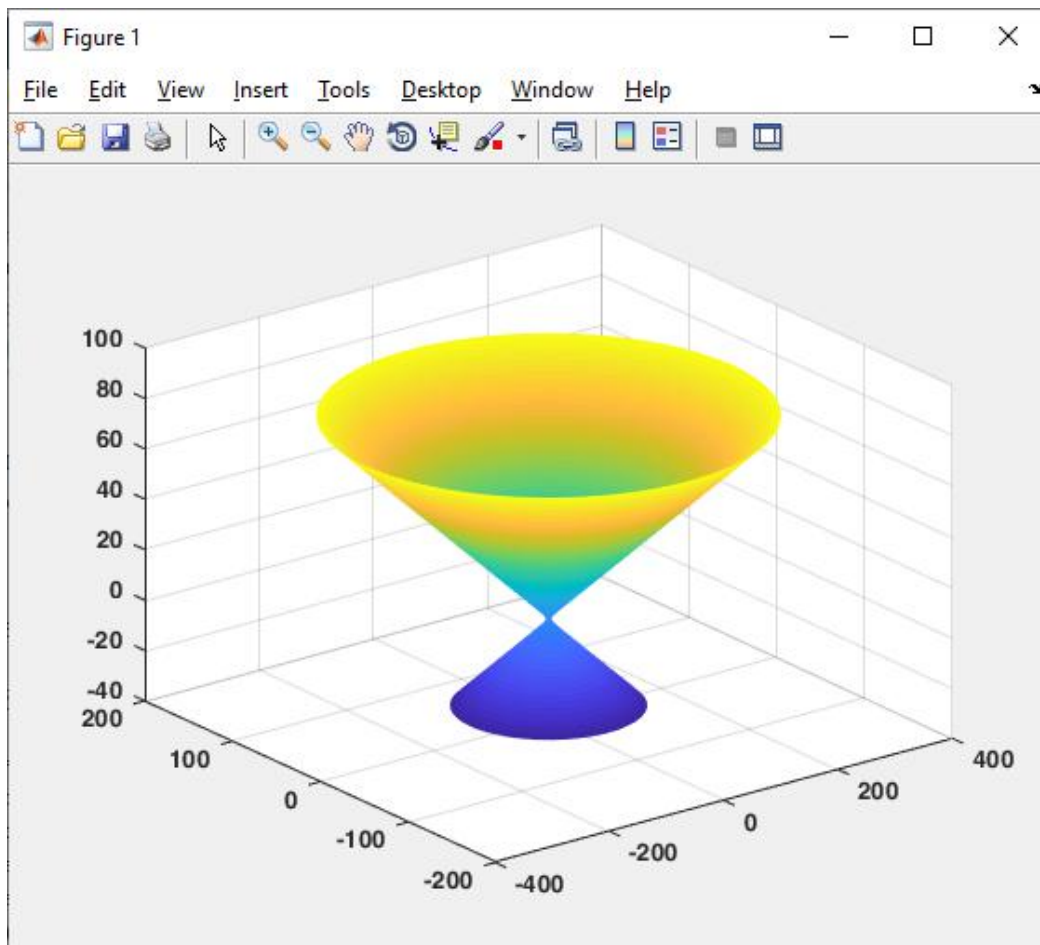


Figure 4-17

Exercise 4-6. Get the graph of the surface of revolution that is turning the function $\sin(x)$ around the axis z . also get the graph of the surface of revolution rotating function e^x around the axis y .

To obtain the equation of the surface, on the assumption that the rotation is around the axis Z consider the graph of the generating curve $y = r(z)$ in the plane YZ . To turn this graph around the axis Z , forms a surface of revolution. The sections with flat $z = z_0$ are circles whose RADIUS is $r(z_0)$ and equation $x^2 + y^2 = [r(z_0)]^2$. That means that the equation $x^2 + y^2 = [r(z)]^2$ describes the points on the surface of revolution. For our problem, is $r(z) = \sin(z)$ and the curve representing a is $x^2 + y^2 = \sin^2[z]$, which are parametric through the input of Matlab graphics:

```
» r=(0:0.1:2*pi)';  
» t=(-pi:0.1:2*pi);  
» X = cos(r)*sin(t);  
» Y = (r)*sin(t);  
» Z = ones(1, length(r))*t;  
» surf(X,Y,Z), shading interp
```

Gets the graph of Figure 4-18:

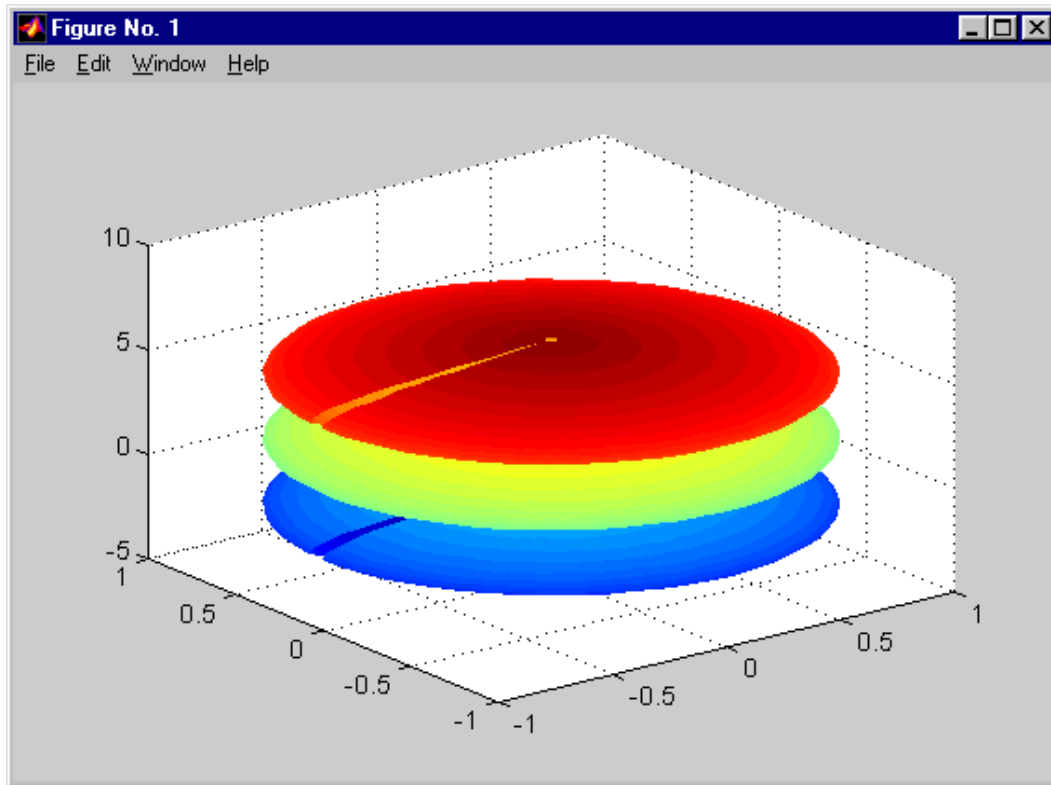


Figure 4-18

If we propose to Matlab entry:

```
»r=(0:0.1:2*pi)';
```

```
»t=(-2:0.1:2);
```

```
» X =cos(r)*exp(t);
```

```
»Y=ones(1,length(r))*t;
```

```
» Z =sin(r)*exp(t);
```

```
»surf(X,Y,Z), shading interp
```

Gets the graph of Figure 4-18, which has been found in the same way as the previous one, but rotating around the axis and exponential function (the generating curve is now the function e^x).

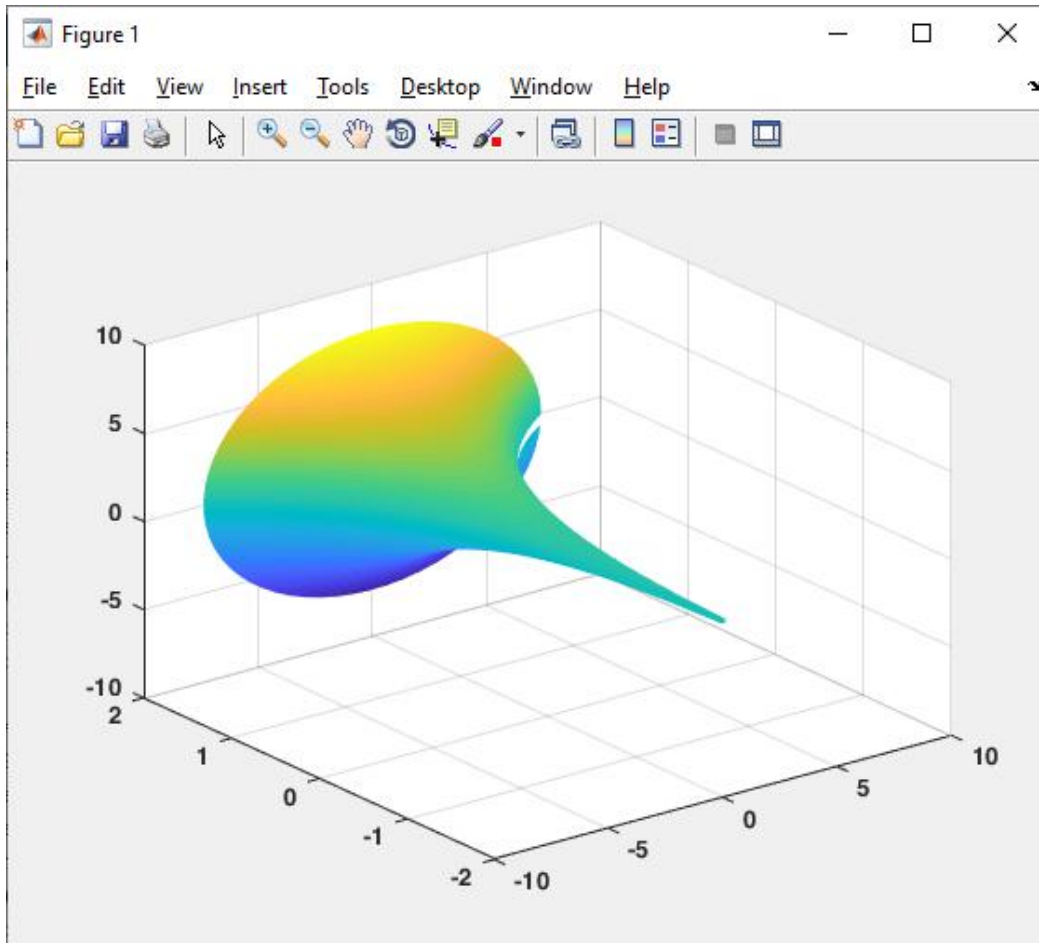


Figure 4-18

Exercise 4-7. Represent the cylinder $\{t, \text{Sen}[t], u\}$, with $\{t, 0, 2\text{Pi}\}$ and $\{u, 0, 4\}$ and the revolution bull $\{\text{Cost}, \text{sint}, \text{Sen}[u]\}$, with $\{t, 0, 2\text{Pi}\}$ and $\{u, 0, 2\text{Pi}\}$.

- » `t=(0:0.1:2*pi)';`
- » `r=(0:0.1:4);`
- » `X =sin(t)*ones(size(r));`
- » `Y =cos(t)*ones(size(r));`
- » `Z =ones(1,length(t))*r;`
- » `surf(X,Y,Z), shading interp`

Gets the graph of Figure 4-19:

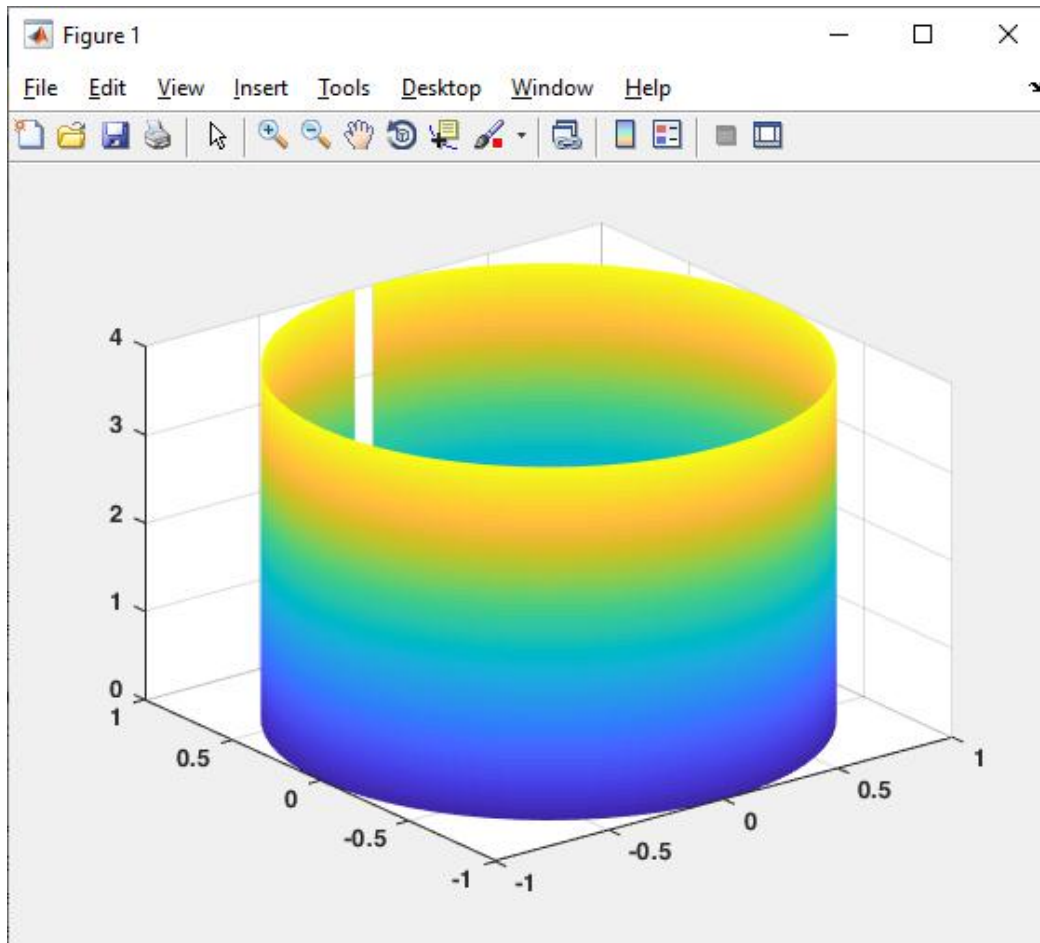


Figure 4-19

To represent the torus of revolution, we use the following syntax:

- » `t=(0:0.1:2*pi)';`
- » `r=(0:0.1:2*pi);`
- » `X=(3+cos(t))*cos(r);`
- » `Y=(3+cos(t))*sin(r);`
- » `Z= ones(1,length(t))*sin(r);`
- » `surf(X,Y,Z), shading interp`

We get the graph of Figure 4-20:

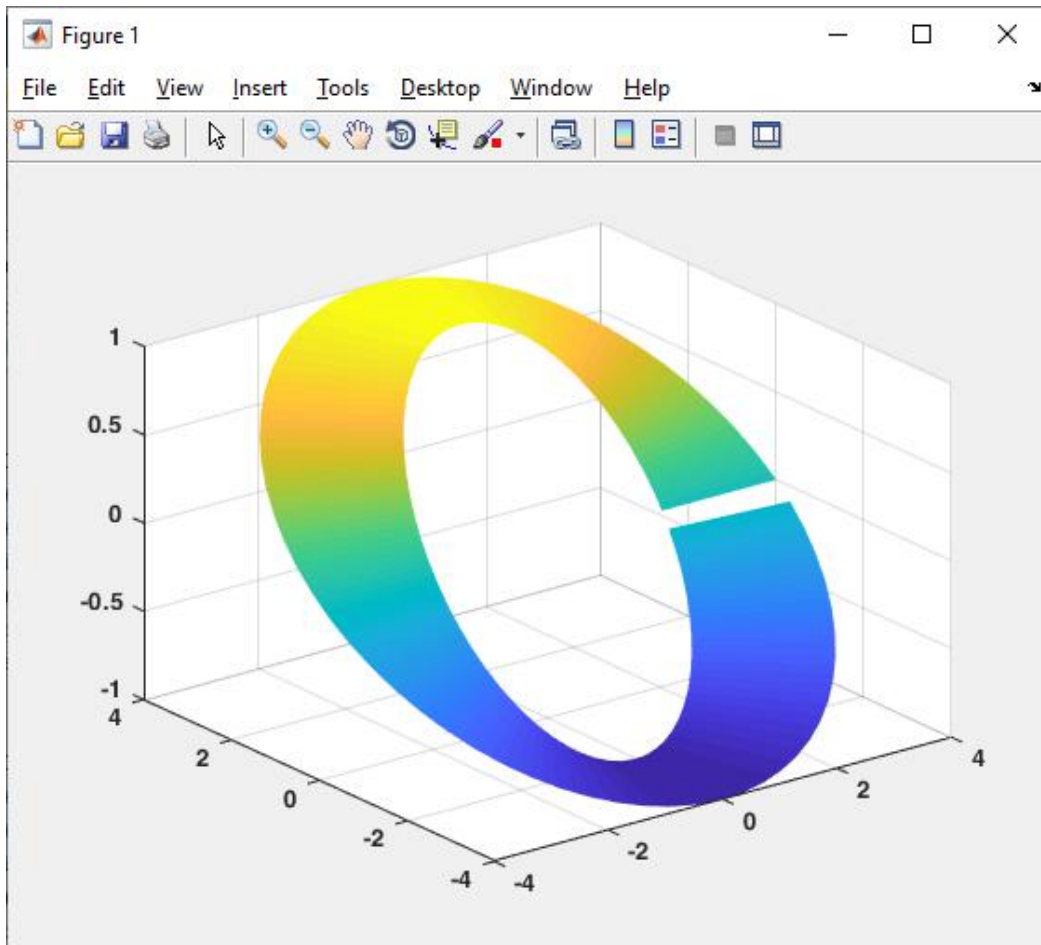


Figure 4-20

MATLAB enables commands to generate cylinders and spheres. We have:

$[X, Y, Z] = \text{cylinder}(r, n)$

draws the cylinder generated by the curve r , which has n points on the circumference horizontal section and that is aligned with ($n = 20$ per default) Z axis

$[X, Y, Z] = \text{sphere}(n)$

draws a sphere (by default $n = 20$)

As an example, let's represent the cylinder generated by the curve $4\cos(t)$ when t varies between 0 and 2π . The syntax will be as follows:

```
»t=0:pi/10:2*pi;
```

```
»[X,Y,Z]=cylinder(4*cos(t));
```

```
»surf(x,y,z)
```

Gets the graph of Figure 4-21:

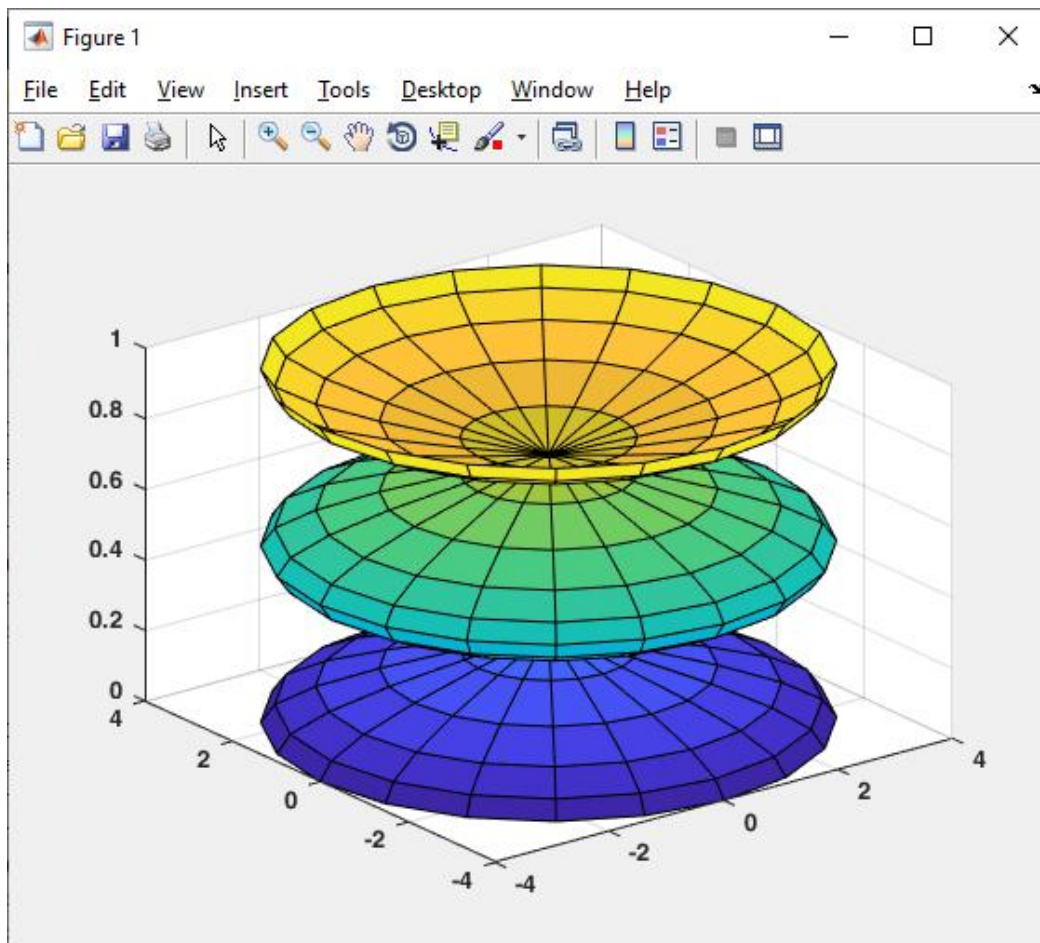


Figure 4-21

Graphic characteristics treated so far belong to the high level of Matlab GUI. However, there are low level (Handle Graphics) commands that allow creating and manipulating figures, axes, lines, surfaces, images, text, menus and other graphics objects. Between commands to create graphical objects are:

`figure (h)` or `h = figure`

creates the figure as an object of name `h` , and is located as a current figure. `Gcf (h)` command is used to mean any property to figure `h` . The command `close (h)` figure `close h` . The command `whitebg (h)` changes the color of the background of the figure `h` . The command `clf` closes the current figure. The command `graymon` located grayscale. The command `newplot` determines the axis to make a new figure. The command `refresh` redraws the figure

`axes (e)` or `e = axes`

creates shafts as an object of name `e` , in the current figure. Use the command `gca (e)` to refer any property to the axes `e`. Use the command `cla` is used to delete all the objects referring to the current axes.

`l = line(x,y)` or `l = line (x, y, z)`

creates, as an object of name `l` , the line joining the points `(X, and)` in the flat, or `(X, Y, Z)` space

`p = (X, Y, C) patch` or `patch(X,Y,Z,C)`

creates an opaque polygonal area that is defined by the set of points `(X, and)` in the flat, or `(X, Y, Z)` space, and whose color is given by `C` , as an object of name `p`

`s = surface(X,Y,Z,C)`

creates the parametric surface defined by `X` , `Y` and `Z` , and whose color is given by `C` as an object of name `(s)`

`i = image (C)`

creates the image defined by the colors in the array `C` as an object of name `i`

`t = text (x, y, 'string')` or `t = text (x, y, z, 'string')`

creates the text defined by the chain, located at the point `(x, y)` plane, or at the point `(x, y, z)` space

Each object has a level of hierarchy. The parents of an object are superior to the hierarchy, and children are the objects of lower hierarchy. Senior object is created with figure , then, is the one created by axes and, finally, and at the same level, are created by image , patch , surface , text and line . This means that if, for example, you want to create a surface, first has to create figure that is going to graph, then the axes and, finally, the surface itself.

So far we have seen commands that allow you to create objects, but, in addition, all these objects can have properties, as style of line, color, etc. List of possible properties to each object is a very long, and its full knowledge requires detailed consultation of MATLAB Reference manual. As a general rule, the name of a property of an object is a compound word whose components begin with capital letter. For example, the line style property has name LineStyle . The names of the properties that are mapped by default to an object are by Default , as, for example, DefaultFigureColor , which assigns the color by default to a figure. Below, are some of the most typical properties which must be seen in the different objects.

Among the commands that allow you to perform operations with graphical objects already created are as follows:

```
set(h, 'propiedad1', 'propiedad2',...)
```

puts the specified properties in the object h

```
get (h, 'property')
```

returns the current value of the specified property to the object h

```
object = gco
```

returns the current object of the current figure

```
rotate(h, [a, e], , [p,q,r])
```

rotates the object h angle , according to the axis of azimuth , and elevation and, being the origin point (p, q, r)

```
reset (h)
```

updates all properties assigned to the object h and set its properties by default

`delete(h)`

deletes the object h

Here are some examples:

The following syntax places the limits of variation of the current X , Y and Z axes to the specified values:

» `set(gca, 'Xlim', [0,10], 'Ylim', [-25, 25], 'Zlim' [-8,10])`

The following syntax places the color of the background of the current figure in white:

» `set(gcf, 'Color', 'w')`

The following syntax returns the current properties for a surface previously created named surfh :

» `get(surfh)`

The following syntax returns the line style of the surface surfh :

» `get('LineStyle' (surfh)`

The following syntax deletes the surface surfh :

» `delete (surfh)`

Exercise 4-8. Represent coordinates following parametric surface:

$$x(t)=4\cos(r)\cos(t), y(t)=2\sin(r)\cos(t), z(t) = \sin(t) - \pi < r < \pi, -\pi/2 < t < \pi/2$$

so that is presented in a figure with title "Surface parametric" and whose background color is white, being its black color axis. On the other hand, the surface presented their grids with yellow colours and must be enclosed in a cube.

»`r=(-pi:0.1:pi)';`

```
»t=(-pi/2:0.1:pi/2);  
» x =4cos(r)cos(t);  
»y=2sin(r)cos(t);  
»z=ones(1,length(r))*sin(t);  
»surface=surf(x,y,z);  
»set(surface,'EdgeColor','interp')  
»set(gcf,'Color','w','Name','SuperficieParamétrica');  
»set(gca,'XColor','k','YColor','k','ZColor','k','Box','on');
```

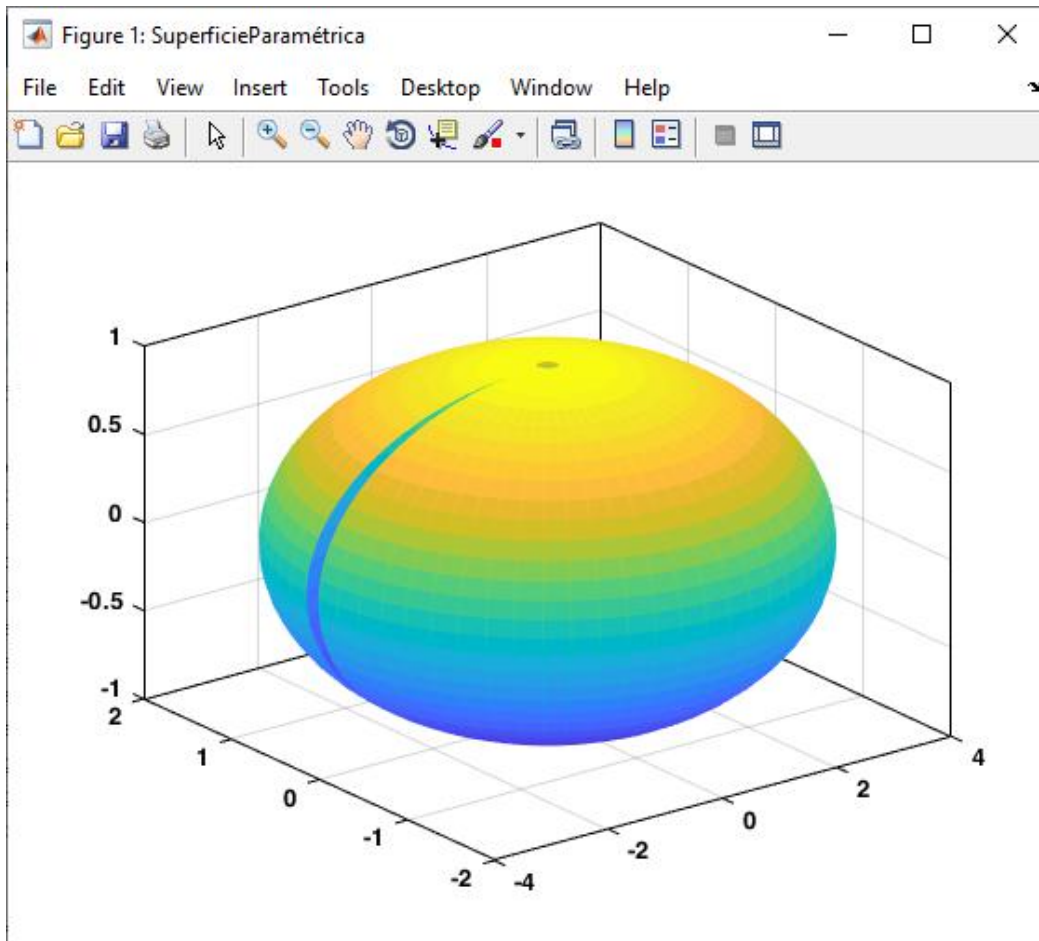


Figure 4-22

Figure 4-22 represents the ordered surface.

Chapter 5.

MATLAB LANGUAGE ELEMENTS. ALGEBRAIC EXPRESSIONS, POLYNOMIALS, EQUATIONS AND SYSTEMS

MATLAB incorporates a wide range of commands, including simplification, expansion and factorization, that allow you to work with algebraic expressions. The following table shows the most common commands used when working with algebraic expressions.

MATLAB implements specific commands for working with polynomials, such as finding their roots, differentiation and interpolation. The following table shows the syntax and examples of the most important of these commands.

MATLAB implements both algebraic and graphical commands for polynomial interpolation, the most important of which are summarized in the following table.

MATLAB includes multiple commands for solving equations and systems of equations. The following sections present the syntax and main features of these methods.

Below are the most common MATLAB commands used to solve equations and systems

Below are the MATLAB commands that can be used to solve equations and systems of equations by the biconjugate gradient method.

Below are the MATLAB commands that are used to solve equations and systems of equations by the method of conjugate gradients.

Below are the MATLAB commands that are used to solve equations and systems of equations by the residual method.

Below are the MATLAB commands that are used to solve equations and systems of equations by the symmetric and non-negative least squares methods.

In the previous sections we have studied equations and systems in general. We will now focus on linear systems of equations. To solve such systems we could simply use the commands we have seen so far, however MATLAB has a selection of special commands designed especially for linear systems. The following table lists these commands.

Systems of linear equations can be converted to array form and solved using calculations with matrices. A system can be written in the form $M \cdot X = B$, where X is the vector of variables, B the vector of independent terms and M the matrix of coefficients of the system. If M is a square matrix and the determinant of the matrix M is non-null, M is invertible, and the unique solution of the system can be written in the form: $X = M^{-1} B$. In this case, the command `solve`, `linsolve`, `lscov`, `bicg`, `pcg`, `lsqr`, `gmr`, `gmres`, `minres`, `symmlq` or `M`, already described above, offer the solution.

If the determinant of M is zero, the system has infinitely many solutions, since there are rows or columns in M that are linearly dependent. In this case, the number of redundant equations can be calculated to find out how many variables are needed to describe the solutions. If the matrix M is rectangular (not square), the system may be undetermined (the number of equations is less than the number of variables), overdetermined (the number of equations is greater than the number of variables) or non-singular (the number of equations is equal to number of variables and M has non-zero determinant). An indeterminate system can have infinitely many solutions, or none, and likewise for an overdetermined system. If a system has no solution, it is called inconsistent (incompatible), and if there is at least one solution, it is called consistent (compatible). The system $M \cdot X = B$ is called homogeneous when the vector B is the null vector, i.e. the system is of the form $M \cdot X = 0$. If the determinant of M is non-null, the unique solution of the system is the null vector (obtained with the command `linsolve`). If the determinant of M is zero, the system has infinitely many solutions. The solutions can be found using the commands `solve`, `linsolve`, `lsqr` or other commands described above for general linear systems.

A fundamental tool in the analysis and solution of systems of equations is the Rouché-Frobenius theorem. This theorem says that a system of m equations with n unknowns has a solution if, and only if, the rank of the matrix of coefficients coincides with the rank of the array extended with the vector column of the system-independent terms. If the two ranks are

equal, and equal to the number of unknowns, the system has a unique solution. If the two ranks are the same, but less than the number of unknowns, the system has infinitely many solutions. If they are different, the system has no solution.

In summary: Let A be the matrix of coefficients of the system and B the matrix A augmented by the column vector of independent terms.

If $\text{rank}(A) \neq \text{rank}(B)$, the system is incompatible (without solution).

If $\text{rank}(A) = \text{rank}(B) < n$, the system is indefinite (has infinitely many solutions).

If $\text{rank}(A) = \text{rank}(B) = n$, the system has a unique solution.

This theorem allows us to analyze the solutions of a system of equations before solving it.

We have already encountered homogeneous systems. A system $A \cdot X = B$ is said homogeneous if the vector of independent terms B is null, so every homogeneous system is of the form $A \cdot X = 0$. In a homogeneous system, the rank of the matrix of coefficients and the rank of the matrix augmented to include the column vector of independent terms always coincide. If we apply the Rouché-Frobenius theorem, a homogeneous system will have a unique solution when the determinant of the matrix A is non-zero. Since the null vector is always a solution of a homogeneous system, this must be the unique solution. A homogeneous system will have infinitely many solutions when the determinant of the matrix A is zero. In this case, the solutions are calculated as for general systems (using the command `solve`), or by using the function `null(A)`.

As a first example we solve the system:

$$2x + y + z + t = 1$$

$$x + 2y + z + t = 1$$

$$x + y + 2z + t = 1$$

$$x + y + z + 2t = 1$$

We will find the rank of the matrix of the system and the rank of the augmented matrix obtained by extending the matrix by the column vector of independent terms.

$$A = [2,1,1,1;1,2,1,1;1,1,2,1;1,1,1,2];$$

$$B = [2,1,1,1,1;1,2,1,1,1;1,1,2,1,1;1,1,1,2,1];$$

$$[\text{rank}(A), \text{rank}(B)]$$

ans =

4 4

We note that the ranks of the two matrices coincide with the number of unknowns. The Rouché-Frobenius theorem then tells us that the system is compatible with a unique solution. We can calculate the solution in the following way:

$$B = [1 \ 1 \ 1 \ 1]';$$

$$\text{linsolve}(A, B)$$

ans =

0.2000

0.2000

0.2000

0.2000

The solution could also have been found using the following commands:

$$\text{lscov}(A, B)$$

ans =

0.2000

0.2000

0.2000

0.2000

bicg(A, B)

bicg converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

pcg(A, B)

PCG converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

lsqr(A, B)

lsqr converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

qmr(A, B)

QMR converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

gmres(A, B)

gmres converged at iteration 1 to a solution with relative residual 1.5e-016

ans =

0.2000

0.2000

0.2000

0.2000

symmlq(A, B)

symmlq converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

As a second example, we solve the system:

$$x + 2y + 3z = 6$$

$$x + 3y + 8z = 19$$

$$2x + 3y + z = -1$$

$$5x + 6y + 4z = 5$$

We find the rank of the matrix of the system and the rank of the augmented matrix.

$$A=[1,2,3;1,3,8;2,3,1;5,6,4];$$

$$B=[1,2,3,6;1,3,8,19;2,3,1,-1;5,6,4,5];$$

$$[\text{rank}(A), \text{rank}(B)]$$

ans =

3 3

We note that the ranks of the two matrices coincide with the number of unknowns. The Rouché-Frobenius theorem then tells us that the system is compatible with a unique solution. We can calculate the solution in the following way:

$$A = [1,2,3;1,3,8;2,3,1;5,6,4];$$

$$B = [19-6 - 5-1]';$$

$$\text{linsolve}(A, B)$$

ans =

1.0000

-2.0000

3.0000

As a third example, we solve the system:

$$x + 2y - z = 0$$

$$2x - y + z = 0$$

$$3x + y = 0$$

As we have a homogeneous system, we will calculate the determinant of the matrix of coefficients of the system.

$$A = [1, 2, -1; 2, -1, 1; 3, 1, 0];$$

$$\det(A)$$

ans =

$$5.5511e-016$$

This answer is very close to zero, in fact the determinant is actually zero, thus the homogeneous system will have infinitely many solutions, which are calculated with the command solve as shown below.

$$[x, y, z] = \text{solve}('x+2y-z, 2x-y+z, 3*x+y', 'x,y,z')$$

x =

$$-z/5$$

y =

$$(3 * z) / 5$$

z =

$$z$$

Thus the infinite set of solutions depend on a parameter z and are described as $\{(-z/5, 3z/5, z)\}$, $z \in \mathbb{R}$.

Exercise 1. Expand the following algebraic expressions:

$$(x + 1)(x + 2), \frac{x + 1}{x + 2},$$

$$\sin(x + y), \cos(2x), e^{a+\ln(b)}, \ln\left(\frac{x}{(1-x)^2}\right), (x + 1)(y + z).$$

» syms x y z b t

» pretty(expand((x+1)*(2.4.x+2)))

2

x + 3 x + 2

» pretty(expand((x+1)/(2.4.x+2)))

x 1

—— + ——

x + 2 x + 2

» pretty(expand(sin(x+y)))

sin(x) cos (y) + cos (x) sin(y)

» pretty(expand(cos(2*x)))

2

2 cos (x) - 1

» pretty(expand(exp(a+log(b))))

exp (a) b

» pretty(expand(log(x)/(1-x)^2)))

log (x) - 2-log(1-x)

» pretty(expand((x+1)*(y+z)))

$$x y + x z + y + z$$

Exercise 2. Factorize the following algebraic expressions:

$$6x^2 + 18x - 24, \quad x^4 - y^4, \quad x^3 + y^3, \quad \frac{x^3 - y^3}{x^4 - y^4}$$

syms x y

pretty(factor(6x^2+18x-24))

$$6 (x + 4) (x - 1)$$

pretty(factor(x^4-y^4))

$$2 \quad 2$$

$$(x + y) (x + y) (x + y)$$

pretty(factor(x^3+y^3))

$$2 \quad 2$$

$$(x + y) (x - x y + y)$$

pretty(factor((x^3-y^3)/(x^4-y^4)))

$$2 \quad 2$$

$$x + x y + y$$

$$2 \quad 2$$

$$(x + y) (x + y)$$

Exercise 3. Simplify the following algebraic expressions:

$$\sin^2(x) + \cos^2(x), e^{a+\ln(be^c)}, \cos(3a \cos(x)), \frac{x^2 - y^2}{(x - y)^3}$$

syms x y b c

simplify(sin(x)²+cos(x)²)

ans =

1

pretty(simplify(exp(a+log(b*exp(c)))))

b exp(a + c)

pretty(sym(simplify(cos(3*acos(x)))))

3

4 x - 3 x

pretty(simplify((x²-y²)/(x-y)³))

1

•

x

Exercise 4. Rewrite the following algebraic expressions in terms of powers of x:

$$f(x) = a^3x - x + a^3x + a, \quad p(x) = y/x + 2z/x + x^{1/3} - y^*x^{1/3}, \quad q(x) = (x+1)(x+2)$$

Rewrite the following expression in terms of powers of sin(x): $y(\sin(x) + 1) + \sin(x)$

Rewrite the following expression in terms of powers of ln(x): $f = a \ln(x) - x \ln(x) - x$

Rewrite the following expression in terms of powers of x and y: $p = xy + zxy + yx^2 + zyx^2 + x + zx$

```
syms x y z
```

```
pretty(collect(a^3*x-x+a^3+a, x))
```

3 3

$(a - 1) x + a + a$

```
pretty(collect(y/x+2z/x+x^(1/3)-y^(1/3)x,x))
```

$y + 2 z - x^{4/3} y + x^{4/3}$

x

```
pretty(collect((x+1)*(x+2)))
```

2

$x + 3 x + 2$

```
p=xy+zxy+yx^2-zyx^2+x+z*x;
```

```
pretty(collect(p, [x,y]))
```

2

$(1-z) x y + (z + 1) x y + (z + 1) x$

```
f=a*log(x)-log(x)x-x;
```

```
pretty(collect(f,log(x)))
```

$(a - x) \log (x) - x$

Exercise 5. Combine the terms as much as possible in the following expression:

$a \ln(x) + 3 \ln(x) - \ln(1-x) + \ln(1+x) / 2$

Simplify it assuming that a is real and $x > 0$.

```
pretty(sym(simplify(a*log(x)+3*log(x)-log(1-x)+log(1+x)/2)))
```

$\log(x + 1)/2 - \log(1-x) + 3 \log(x) + \log(x)$

```
x = sym('x', 'positive')
```

x =

x

```
a = sym('a', 'real')
```

a =

a

```
pretty(sym(simplify(a*log(x)+3*log(x)-log(1-x)+log(1+x)/2)))
```

/ x - 1

-log| - ————— |

| 3 a 1/2 |

x x (x + 1) /

Exercise 6. Expand and simplify the following trigonometric expressions:

a. $\sin [3x] \cos [5 x]$

b. $[(\cot[a])^2 + (\sec[a])^2 - (\csc[a])^2]$

c. $\sin [a] / (1 + \cot[a]^2) - \sin [a]^3$

```
pretty(simplify(expand(sym(sin(3x)cos(5*x))))))
```

$\sin(8 x) \sin (2 x)$

pretty(simplify(expand(((cot(a))^{2+(sec(a))2-(csc(a))^2}))))

1

----- - 1

2

cos (a)

pretty(simplify(expand(sin(a)/(1+cot(a)^{2-sin(a)3}))))

0

Exercise 7. Simplify the following algebraic expressions as much as possible:

$x^2 - y^2 - \frac{2xy}{x^2 + y^2}$

$\frac{1+a}{1-b} - \frac{a-b}{ab}$

• ◦ , + -

$\frac{x+y}{x-y} - \frac{2}{b} + \frac{a}{ab}$

$x - y$

pretty(simplify(expand(x/(x+y)-y/(x-y)+2xy/(x²-y²))))

1

pretty(simplify(expand(((1+a^{2/b+(1-b)2})/a-(a^{3-b)3})/(a*b))))

1 1

• ◦ ■

a b

Exercise 8. Simplify the following algebraic fractions as much as possible:

$$a^3 - a^2b + ac^2 - bc^2 \quad (x - 9)(x^2 - 2x + 1)(x - 3)$$

,

$$a^3 + ac^2 + a^2b + b^2c \quad (x - 6x + 9)(x - 1)(x - 1)$$

$$\text{pretty(simplify(expand(a^3-a^2b+ac^2-bc^2)/(a^3+ac^2+a^2b+bc^2)))}$$

a - b

a + b

$$\text{pretty(simplify(expand((x^2-9)(x^2-2x+1)(x-3))/((x^2-6x+9)(x^2-1)(x-1))))}$$

2

— + 1

x + 1

Exercise 9. Calculate the roots of the following polynomials:

$$x^3 - 6x^2 - 72x - 27, \quad 2x^4 - 3x^3 + 4x^2 - 5x + 11, \quad x^{11} - 1$$

Evaluate the first polynomial at the identity matrix of order 3, the second at the unit matrix of order 3 and the third at a uniformly random matrix of order 3.

Find the coefficients of the derivatives of the given polynomials and display the results in polynomial form.

$$p1 = [1 -6 -72 -27]; r = \text{roots}(p1)$$

r =

12.1229

-5.7345

-0.3884

$p2 = [2 -3 4 -5 11]; r = \text{roots}(p2)$

r =

1.2817 + 1.0040i

1.2817 - 1.0040i

-0.5317 + 1.3387i

-0.5317 - 1.3387i

$p3 = [1 0 0 0 0 0 0 0 0 0 0 1]; r = \text{roots}(p3)$

r =

-1.0000 + 0.0000i

-0.8413 + 0.5406i

-0.8413 - 0.5406i

-0.4154 + 0.9096i

-0.4154 - 0.9096i

0.1423 + 0.9898i

0.1423 - 0.9898i

0.6549 + 0.7557i

0.6549 - 0.7557i

0.9595 + 0.2817i

0.9595 - 0.2817i

```
Y1=polyval(p1,eye(3))
```

```
Y1 =
```

```
-104 - 27 - 27
```

```
-27 -104 - 27
```

```
-27 - 27 -104
```

```
Y2=polyval(p2,ones(3))
```

```
Y2 =
```

```
9 -9 -9
```

```
9 -9 -9
```

```
9 -9 -9
```

```
Y3=polyval(p3,rand(3))
```

```
Y3 =
```

```
1.1050 1.3691 1.0000
```

```
1.3368 1.0065 1.0013
```

```
1.0000 1.0000 1.6202
```

```
d1 = polyder(p1)
```

```
D1 =
```

```
3 -12 -72
```

```
pretty(poly2sym(d1,x))
```

```
2
```

```
3 x - 12 x - 72
```

```
d2 = polyder(p2)
```

D2 =

8 - 9 8 - 5

pretty(poly2sym(d2,x))

3 2

8 x - 9 x + 8 x - 5

d3 = polyder(p3)

D3 =

11 0 0 0 0 0 0 0 0 0 0

pretty(poly2sym(d3,x))

10

11 x

Exercise 10. Consider the equally spaced set of points in the interval [0,5] separated by one tenth. Interpolate the error function at these points and adjust a polynomial of degree 6. Represent the original curve and the interpolated on the same graph.

x = (0:0.1:5)';

p = polyfit(x,y,6);

y = erf(x);

f = polyval(p,x);

p = polyfit(x,y,6)

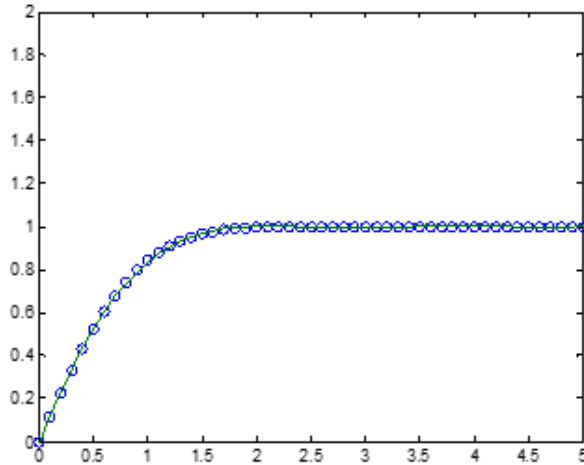
p =

0.0012 - 0.0173 0.0812 - 0.0791 - 0.4495 1.3107 - 0.0128

f = polyval(p,x);

```
plot(x,y,'o',x,f,'-')
```

```
axis([0 5 0 2])
```



Exercise 11. Calculate the second degree interpolating polynomial passing through the points (- 1.4), (0,2), and (1.6) in the least squares sense.

```
» x = [-1, 0, 1]; y=[4,2,6]; p=poly2sym(polyfit(x,y,2))
```

p =

```
3 * x ^ 2 + x + 2
```

Exercise 12. Represent 200 points of cubic interpolation between the points (x, y) given by $y= e^x$ for x values in 20 equally spaced intervals between 0 and 2.

First, we define the 20 points (x, y) , for x equally spaced between 0 and 2:

```
» x = 0:0.1:2;
```

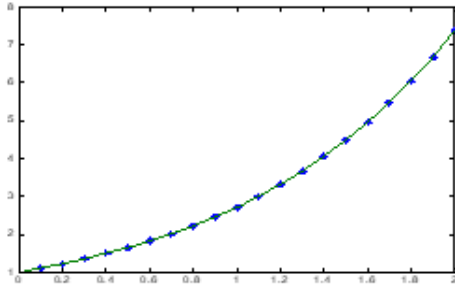
```
» y = exp(x);
```

Now we find cubic interpolation points (xi, yi) , for x values in 200 equally spaced between 0 and 2, and represent them on a graph together with the initial points (x, y) (indicated by asterisks).

```

» xi = 0:0.01:2;
» yi = interp1(x,y,xi,'cubic');
» plot(x,y,'*',xi,yi)

```



Exercise 13. Find interpolation points of the parametric function $X = \cosh(t)$, $Y = \sinh(t)$, $Z = \tanh(t)$ for values of t between 0 and $\pi/6$ in 25 equally spaced intervals..

First, we define the given points (x, y, z) , for equally spaced values of t between 0 and $\pi/6$.

```

t=0:pi/150:pi/6;
x=cosh(t); y=sinh(t); z=tanh(t);

```

Now we find the 26 points of interpolation (x_i, y_i, z_i) , for values of the parameter t equally spaced between 0 and $\pi/6$.

```

» xi = cosh(t); yi = sinh(t);
» zi = griddata(x,y,z,xi,yi);
» points = [xi, yi, zi]

```

points =

1.0000 0 0

1.0002 0.0209 0.0209

1.0009 0.0419 0.0419

1.0020 0.0629 0.0627
1.0035 0.0839 0.0836
1.0055 0.1049 0.1043
1.0079 0.1260 0.1250
1.0108 0.1471 0.1456
1.0141 0.1683 0.1660
1.0178 0.1896 0.1863
1.0220 0.2110 0.2064
1.0267 0.2324 0.2264
1.0317 0.2540 0.2462
1.0373 0.2756 0.2657
1.0433 0.2974 0.2851
1.0498 0.3194 0.3042
1.0567 0.3414 0.3231
1.0641 0.3636 0.3417
1.0719 0.3860 0.3601
1.0802 0.4085 0.3782
1.0890 0.4312 0.3960
1.0983 0.4541 0.4135
1.1080 0.4772 0.4307
1.1183 0.5006 0.4476
1.1290 0.5241 0.4642

1.1402 0.5479 0.4805

Exercise 14. Using fast Fourier transform (FFT) interpolation, find the 30 points (x_i, y_i) approximating the function $y = \sinh(x)$ for values of x that are in equally spaced intervals between 0 and 2π , interpolating them between values of (x, y) given by $y = \sinh(x)$ for x values in 20 evenly spaced intervals in $(0, 2)$. Graph the points.

First, we define the x values equally spaced in 20 intervals between 0 and 2 .

```
» x =(0:pi/10:2*pi);
```

Now we find the interpolation points (x, y) orders.

```
y = interpft(sinh(x), 30);
```

```
points = [y', (asinh(y))']
```

points =

-0.0000 - 0.0000

-28.2506 - 4.0346

23.3719 3.8451

-4.9711 - 2.3067

-7.7918 - 2.7503

14.0406 3.3364

-4.8129 - 2.2751

-0.8717 - 0.7877

11.5537 3.1420

-3.3804 - 1.9323

4.4531 2.1991

11.8616 3.1682

-0.2121 - 0.2105

10.9811 3.0914

15.1648 3.4132

6.1408 2.5147

21.2540 3.7502

23.3792 3.8455

18.5918 3.6166

39.4061 4.3672

40.6473 4.3982

42.8049 4.4499

73.2876 4.9876

74.8962 5.0093

89.7159 5.1898

139.0371 5.6279

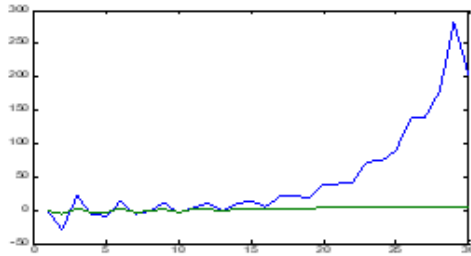
139.3869 5.6304

180.2289 5.8874

282.4798 6.3368

201.7871 6.0004

plot(points)



Exercise 15. Find the polynomial of degree 3 which is the best fit through the points (i, i^2) $1 \leq i \leq 7$, in the least squares sense. Evaluate this polynomial at $x = 10$ and graphically represent the best fit curve.

```
x=(1:7);y=[1,4,9,16,25,36,49];p=vpa(poly2sym(polyfit(x,y,2)))
```

p =

```
x^2-
0.000000000000009169181662272871686413366801652*x+0.00000000
0000020761891472015924526365781895317
```

Now we calculate the numerical value of the polynomial p at $x = 10$.

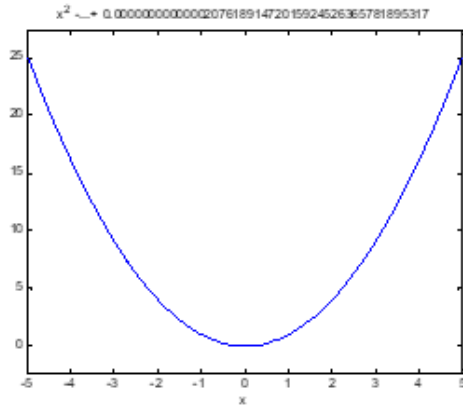
```
subs(p,10)
```

ans =

```
100.0000
```

Next we graph the polynomial:

```
» ezplot(p,[-5,5])
```



Exercise 16. Find the solutions to the following equations:

$$\sin(x) \cos(x) = 0 \text{ and } ax^2 + bx + c = 0$$

» solve('sin(x)*cos(x)=0')

ans =

pi/2

otherwise

$$\text{solve}(\sin(x) \cdot \cos(x) = 0)$$

ans =

pi/2

» pretty(solve('ax^2+bx+c=0','x'))

/ 2

| b + sqrt(b - 4 a c) |

- ————— |

| 2 a |

| |

| 2 |

$$\frac{|b - \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b + \sqrt{b^2 - 4ac}|}{2a}$$

otherwise

```
pretty(solve(ax^2+bx+c==0,x))
```

/ 2

$$\frac{|b - \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b + \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b - \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b + \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b - \sqrt{b^2 - 4ac}|}{2a}$$

$$\frac{|b + \sqrt{b^2 - 4ac}|}{2a}$$

Exercise 17. Find at least two solutions for each of the following two trigonometric and exponential equations:

$$x \sin(x) = 1/2 \text{ and } 2^{x^3} = 4(2^{3x})$$

First, we use the command solve :

```
vpa(solve('x*sin(x)=1/2','x'))
```

ans =

21.968386631500002609599321864459

otherwise

```
vpa(solve(x*sin(x)==1/2,x))
```

ans =

```
21.968386631500002609599321864459
```

```
vpa(solve('2^(x3)=42^(3x)', 'x'))
```

ans =

```
2.0
```

otherwise

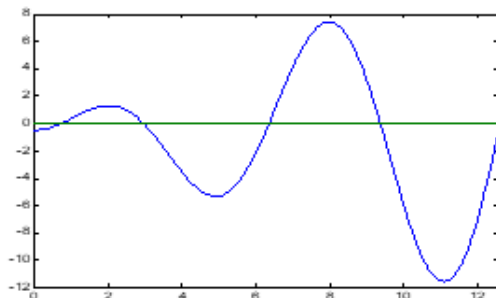
```
vpa(solve(2^(x3)==42^(3x),x))
```

ans =

```
2.0
```

To better analyze the first equation, we graphically represent the function to determine approximate intervals where the possible solutions can be found.

```
» fplot('xsin(x)-1/2',[0,4pi])
```



We observe that there is a solution between 0 and 2, another between 2 and 4, another between 4 and 8, and so on. We can calculate three of them with the command `fzero`.

```
s1=fzero('x*sin(x)-1/2',2)
```

s1 =

0.7408

```
s2=fzero('x*sin(x)-1/2',4)
```

s2 =

2.9726

```
s3=fzero('x*sin(x)-1/2',6)
```

S3 =

6.3619

Exercise 18. Solve each of the following two logarithmic and surd equations:

$$x^{3/2} \log(x) = x \log(x^{3/2}), \quad \sqrt{1-x} + \sqrt{1+x} = a.$$

```
vpa(solve('x^(3/2)/log(x)=xlog(x)(3/2)'))
```

ans =

1.0

otherwise

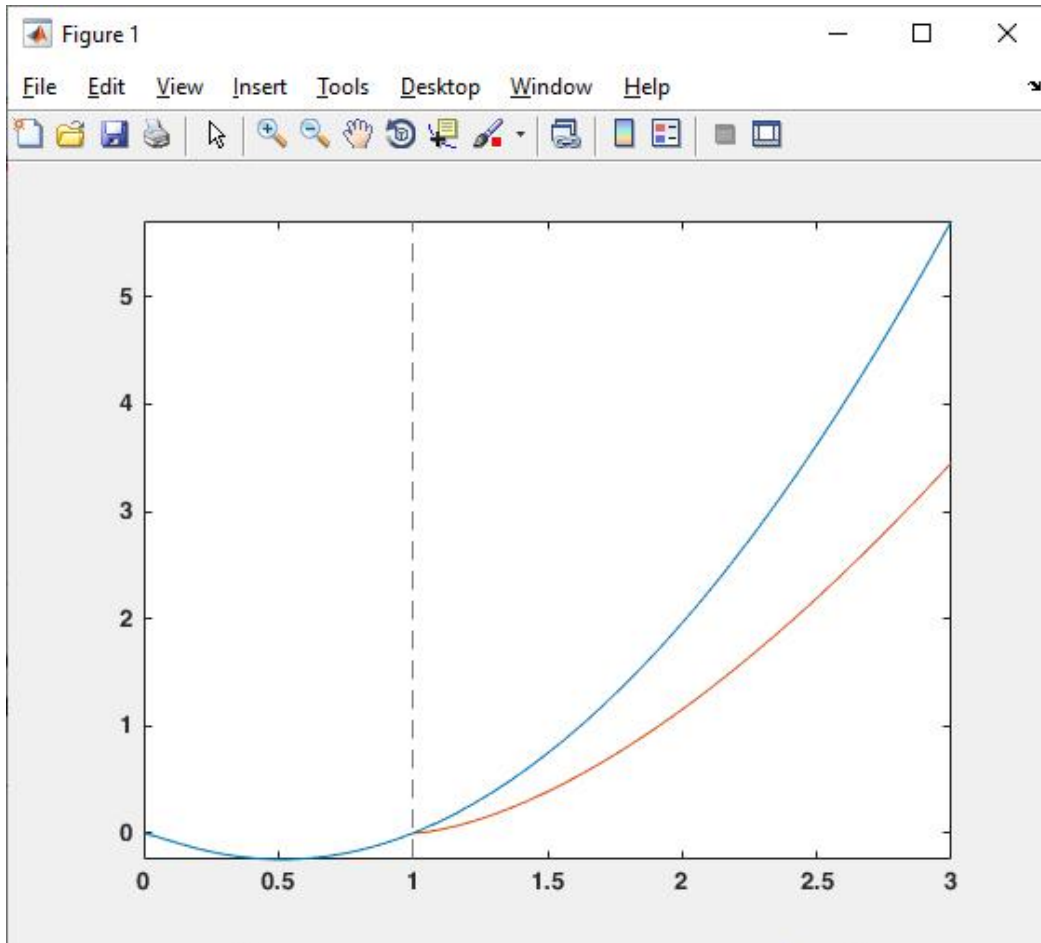
```
vpa(solve('x^(3/2)/log(x)==xlog(x)(3/2)'))
```

ans =

1.0

We graph the function to determine in which intervals a solution can be found. The plot indicates that $x=1$ is the only real solution.

```
fplot('x^(3/2)/log(x),xlog(x)(3/2)',[0,3])
```

Now, we solve the surd equation:

```
pretty(sym(solve('sqrt(1-x)+sqrt(1+x)=a','x')))
```

$$\begin{array}{c}
 +- \qquad \qquad -+ \\
 | \quad 2 \frac{1}{2} | \\
 | \quad a(4 - a) \quad | \\
 | \quad \text{—————} | \\
 | \quad 2 \quad | \\
 | \quad | \\
 | \quad 2 \frac{1}{2} |
 \end{array}$$

$$| a(4 - a) |$$

$$| - \frac{\quad}{\quad} |$$

$$| 2 |$$

Exercise 19. Solve the following system of two equations:

$$\cos(x/12) / \exp(x^2/16) = y$$

$$5/4 + y = \sin(x^{3/2})$$

$$[x,y]=\text{solve}(\cos(x/12)/\exp(x^2/16)==y,5/4+y==\sin(x^{3/2}))$$

x =

$$1.4871476048289696408542396856279$$

$$0.7552602772328636111717863015393i$$

y =

$$0.88755081697835547124465602184681$$

$$0.13253150350741418925867594825393i$$

Exercise 20. Find the intersection of the hyperbolas with equations $x^2 - y^2 = r^2$ and $a^2 x^2 - b^2 y^2 = a^2 b^2$ with the parabola $z^2 = 2px$.

$$[x, y, z] = \text{solve}('a^{2x^2-b^2y^2}=a^2*b^2', 'x^2-y^2=r^2', 'z^2=2px',$$

'x,y,z')

x =

$$((4a^{2b^2p^2-4b^2p^2r^2}) / (a^2-b^2))^{1/2} / (2*p)$$

$$((4a^{2b^2p^2-4b^2p^2r^2}) / (a^2-b^2))^{1/2} / (2*p)$$

$$((4a^{2b^2p^2-4b^2p^2r^2}) / (a^2-b^2))^{1/2} / (2*p)$$

$$-((4a^{2b^2p^2-4b^2p^2r^2}) / (a^2-b^2))^{1/2} / (2*p)$$

$$\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/2} / (2^*p)$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/2} / (2^*p)$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/2} / (2^*p)$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/2} / (2^*p)$$

y =

$$a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$-a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$-a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$-a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

$$-a * \left(\frac{b^2-r^2}{a^{2-b2}}\right)^{1/2}$$

z =

$$\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4}$$

$$\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4}$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4}$$

$$\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4} * i$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4}$$

$$-\left(\frac{4a^{2b}2^{2p^2-4b}2p2r^2}{a^{2-b2}}\right)^{1/4} * i$$

$$\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/4} * i$$

$$-\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/4} * i$$

otherwise

syms x y z a b r p

$$[x, y, z] = \text{solve}(a^{2x^2-b}y^2==a^2b^2,x^2-y^2==r^2, z^2==2p*x,[x,y,z])$$

x =

$$\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$-\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$-\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$-\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

$$-\left(\frac{4a^{2b^2p^2-4b^2p}r^2}{a^2-b^2}\right)^{1/2}/(2*p)$$

y =

$$a\left(\frac{(b+r)(b-r)}{(a+b)(a-b)}\right)^{1/2}$$

$$-a\left(\frac{(b+r)(b-r)}{(a+b)(a-b)}\right)^{1/2}$$

$$a\left(\frac{(b+r)(b-r)}{(a+b)(a-b)}\right)^{1/2}$$

$$-a\left(\frac{(b+r)(b-r)}{(a+b)(a-b)}\right)^{1/2}$$

$$a\left(\frac{(b+r)(b-r)}{(a+b)(a-b)}\right)^{1/2}$$

$$a(((b+r)(b-r))/((a+b)*(a-b)))^{1/2}$$

$$-a(((b+r)(b-r))/((a+b)*(a-b)))^{1/2}$$

$$-a(((b+r)(b-r))/((a+b)*(a-b)))^{1/2}$$

z =

$$((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4}$$

$$((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4}$$

$$-((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4}$$

$$-((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4}$$

$$-((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4} * i$$

$$((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4} * i$$

$$-((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4} * i$$

$$((4a^{2b^{2p^2} - 4b^{2p} 2r^2})/(a^2 - b^2))^{1/4} * i$$

Exercise 21. Study and solve the system:

$$\begin{aligned} x_1 - x_2 + x_3 &= 1 \\ 4x_1 + 5x_2 - 5x_3 &= 4 \\ 2x_1 + x_2 - x_3 &= 2 \\ x_1 + 2x_2 - 2x_3 &= 1 \end{aligned}$$

$$\gg A = [1, -1, 1; 4, 5, -5; 2, 1, -1; 1, 2, -2]$$

A =

$$1 \ -1 \ -1$$

$$5 \ -4 \ -5$$

$$2 \ - \ 1 \ -1$$

$$2 \ - \ 1 \ -2$$

$$\gg B = [1, -1, 1, 1; 4, 5, -5, 4; 2, 1, -1, 2; 1, 2, -2, 1]$$

$$B =$$

$$1 \ -1 \ -1 \ -1$$

$$5 \ 4 \ -5 \ 4$$

$$1 \ 2 \ -1 \ 2$$

$$2 \ 1 \ -2 \ 1$$

$$\gg [\text{rank}(A), \text{rank}(B)]$$

$$\text{ans} =$$

$$2 \ 2$$

We see that the ranks of A and B coincide and its value is 2, which is less than the number of unknowns in the system (3). Therefore, the system will have infinitely many solutions. We try to solve it with the command solve :

$$\text{syms } x1 \ x2 \ x3$$

$$[x1,x2,x3]=\text{solve}(x1-x2+x3==1,4x1+5x2-5x3==4,2x1+x2-x3==2, \\ x1+2x2-2x3==1,x1,x2,x3)$$

$$x1 =$$

$$1$$

$$x2 =$$

$$0$$

$$x3 =$$

$$0$$

Infinitely many solutions are obtained in terms of the parameter z , namely $\{1, z, z\}$, $z \in \mathbb{R}$. Note that the trivial solution $\{1, 0, 0\}$ is obtained by setting the parameter equal to zero.

Exercise 22. Study and solve the system:

$$x + 2y + 3z + t = 6$$

$$x + 3y + 8z + t = 19$$

$$A = [1, 2, 3, 1; 1, 3, 8, 1]$$

$$A =$$

$$1 \ 2 \ 3 \ 1$$

$$1 \ 3 \ 8 \ 1$$

$$\gg B = [1, 2, 3, 1, 6; 1, 3, 8, 1, 19]$$

$$B =$$

$$1 \ 2 \ 3 \ 1 \ 6$$

$$1 \ 3 \ 8 \ 1 \ 19$$

$$\gg [\text{rank}(A), \text{rank}(B)]$$

$$\text{ans} =$$

$$2 \ 2$$

We see that the ranks of A and B coincide, and their common value is 2, which is less than the number of unknowns for the system (4). Therefore, the system has infinitely many solutions. We try to solve it:

$$[x, y, z, t] = \text{solve}('x+2y+3z+t=6', 'x+3y+8z+t=19', 'x', 'y', 'z', 't')$$

$$x =$$

$$7.0z1 - 1.0z - 20.0$$

y =

$$13.0 - 5.0 \cdot z_1$$

z =

z₁

t =

z

otherwise

```
syms x y z t
```

```
[x, y, z, t] = solve(x+2y+3z+t==6,x+3y+8z+t==19,[x,y,z,t])
```

x =

$$7.0z_1 - 1.0z_2 - 20.0$$

y =

$$13.0 - 5.0 \cdot z_1$$

z =

z₁

t =

z

This time the solution depends on two parameters z_1 and z_2 . As these parameters vary over the real numbers (x, y, z, t) varies over all solutions of the system. These solutions form a two-dimensional subspace of the four dimensional real vector space which can be expressed as follows:

$$\{7z_1 - z_2 - 20, z_2, 13 - 5z_1, z_1\}, z_1, z_2 \in \mathbb{R}$$

Exercise 23. Study and solve the system:

$$\begin{aligned}
3x_1 + x_2 + x_3 - x_4 &= 0 \\
2x_1 + x_2 - x_3 + x_4 &= 0 \\
x_1 + 2x_2 + 4x_3 + 2x_4 &= 0 \\
2x_1 + x_2 - 2x_3 - x_4 &= 0
\end{aligned}$$

» det([3,1,1,-1;2,1,-1,1;1,2,4,2;2,1,-2,-1])

ans =

-30

As the determinant of the coefficient matrix is non-zero, the system has only the trivial solution:

```
[(x1,x2,x3,x4)=solve('3x1+x2+x3-x4=0','2x1+x2-x3+x4=0','x1+2x2-4x3-2*x4=0','x1-x2-3x3-5x4=0','x1','x2','x3','x4')]
```

x 1 =

0

x 2 =

0

x 3 =

0

x 4 =

0

Otherwise

```
syms x1 x2 x3 x4
```

```
[x1 x2 x3 x4]=solve(3x1+x2+x3-x4==0,2x1+x2-x3+x4==0,x1+2x2-4x3-2x4==0,x1-x2-3x3-5*x4==0,[x1,x2,x3,x4])
```

$$x_1 =$$

$$0$$

$$x_2 =$$

$$0$$

$$x_3 =$$

$$0$$

$$x_4 =$$

$$0$$

Exercise 24. Study and solve the following system, according to the values of m :

$$m x + y + z = 1$$

$$x + m y + z = m$$

$$x + y + m z = m^2$$

syms m

$$A=[m,1,1;1,m,1;1,1,m]$$

$$A =$$

$$[m, 1, 1]$$

$$[1, m, 1]$$

$$[1, 1, m]$$

det(A)

$$\text{ans} =$$

$$m^3 - 3m + 2$$

`solve('m^3 - 3*m + 2=0','m')`

ans =

-2

1

1

The values of m which determine the rank of the matrix are - 2 and 1.

We now consider the augmented matrix extended to include a fourth column with values 1 , m and m^2 :

`B=[m,1,1,1;1,m,1,m;1,1,m,m^2]`

B =

[m, 1, 1, 1]

[1, m, 1, m]

[1, 1, m, m^2]

We will study the case $m = -2$:

`rank(subs(A,{m},{-2}))`

ans =

2

`rank(subs(B,{m},{-2}))`

ans =

3

We see that the ranks of the two arrays are different, hence the system is inconsistent (i.e. it has no solution) if $m = -2$.

Now we study the case $m = 1$:

```
rank(subs(A,{m},{1}))
```

```
ans =
```

```
1
```

```
rank(subs(B,{m},{1}))
```

```
ans =
```

```
1
```

Now the rank of both matrices is 1, which is less than the number of unknowns. In this case, the system has infinitely many solutions. We find them by substituting $m = 1$ into the initial system:

```
[x,y,z]=solve('x+y+z=1','x','y','z')
```

Warning: 1 equation in 3 variables. New variables might be introduced.

```
x =
```

```
1 - z2 - z1
```

```
y =
```

```
z2
```

```
z =
```

```
z1
```

Thus the solutions are given in terms of two parameters. The two-dimensional subspace of solutions is:

```
{1-z2-z1, z2, z1}, z1, z2 R
```

If we consider the case where m is neither -2 nor 1 , the system has a unique solution, which is given by the command solve :

```
[x,y,z]=solve('mx+y+z=1','x+my+z=m','x+y+m*z=m^2','x','y','z')
```

$$x =$$

$$-(m + 1)/(m + 2)$$

$$y =$$

$$1/(m + 2)$$

$$z =$$

$$(m^2 + 2 * m + 1)/(m + 2)$$

Exercise 25. Study and solve the following system, according to the values of m:

$$m y = m$$

$$(1 + m) x - z = m$$

$$y + z = m$$

syms m

$$A = [0, m, 0; m + 1, 0, -1; 0, 1, 1]$$

$$A =$$

$$[0, m, 0]$$

$$[m + 1, 0, - 1]$$

$$[0, 1, 1]$$

det (A)

$$\text{ans} =$$

$$-m^2 - m$$

solve('-m^2-m=0','m')

$$\text{ans} =$$

-1

0

We see that the values of m which determine the rank of the matrix of coefficients of the system are $m = 1$ and $m = 0$.

We now consider the augmented matrix:

$$B = [0, m, 0, m; m + 1, 0, -1, m; 0, 1, 1, m]$$

$B =$

$$[0, m, 0, m]$$

$$[m + 1, 0, -1, m]$$

$$[0, 1, 1, m]$$

$$\text{rank}(\text{subs}(A, \{m\}, \{-1\}))$$

ans =

2

$$\text{rank}(\text{subs}(B, \{m\}, \{-1\}))$$

ans =

3

If $m = -1$, we see that the system has no solution because the rank of the matrix of coefficients of the system is 2 and the rank of the augmented matrix is 3.

Now, we analyze the case $m = 0$:

When m is zero the system is homogeneous, since the independent terms are all null. We analyze the determinant of the matrix of coefficients of the system.

$$\det(\text{subs}(A, \{m\}, \{0\}))$$

ans =

0

Since the determinant is zero, the system has infinitely many solutions:

$$[x, y, z] = \text{solve}('x-z=0', 'y+z=0', 'x', 'y', 'z')$$

Warning: 2 equations in three variables. New variables might be introduced.

x =

z1

y =

-z1

z =

z1

Thus the solutions are given in terms of one parameter. The one-dimensional subspace of solutions is:

$$\{z1, -z1, z1\}, z1 \in \mathbb{R}$$

If m is neither 0 nor -1, the system has a unique solution, since the ranks of the matrix of the system and of the augmented matrix coincide. The solution, using the function solve, is calculated as follows.

$$[x, y, z] = \text{solve}('m * y = m', '(1+m) * x-z = m', 'y + z = m', 'x', 'y', 'z')$$

x =

$$(2 * m - 1) / (m + 1)$$

y =

1

z =

m - 1

Exercise 26. Study and solve the system:

$$2x + y + z + t = 1$$

$$x + 2y + z + t = 1$$

$$x + y + 2z + t = 1$$

$$x + y + z + 2t = 1$$

$$A = [2, 1, 1, 1; 1, 2, 1, 1; 1, 1, 2, 1; 1, 1, 1, 2];$$

$$B = [2, 1, 1, 1, 1; 1, 2, 1, 1, 1; 1, 1, 2, 1, 1; 1, 1, 1, 2, 1];$$

$$[\text{rank}(A), \text{rank}(B)]$$

ans =

4 4

$$b = [1, 1, 1, 1]';$$

We see that the matrices A and B (the augmented matrix) both have rank 4, which also coincides with the number of unknowns. Thus the system has a unique solution. To calculate the solution we can use any of the commands shown below.

$$x = \text{nls}(A, b)$$

x =

0.2000

0.2000

0.2000

0.2000

`x = bicg(A,b)`

bicg converged at iteration 1 to a solution with relative residual 0

`x =`

0.2000

0.2000

0.2000

0.2000

`x = bicgstab(A,b)`

bicgstab converged at iteration 0.5 to a solution with relative residual 0

`x =`

0.2000

0.2000

0.2000

0.2000

`x = pcg(A,b)`

pcg converged at iteration 1 to a solution with relative residual 0

`x =`

0.2000

0.2000

0.2000

0.2000

`gmres(A,b)`

gmres converged at iteration 1 to a solution with relative residual 0

ans =

0.2000

0.2000

0.2000

0.2000

x = lsqr(A,b)

lsqr converged at iteration 2 to a solution with relative residual 0

x =

0.2000

0.2000

0.2000

0.2000

A

ans =

0.2000

0.2000

0.2000

0.2000

.

Chapter 6.

MATRICES, VECTOR SPACES, LINEAR APPLICATIONS AND QUADRATIC FORMS

We have already seen how vectors and matrices are represented in MATLAB in the chapter dedicated to variables, however we shall recall here the notation.

Consider the matrix

You can enter this in MATLAB in the following ways:

$$A=[a_{11} \ a_{12} \ \dots \ a_{1n} ; a_{21} \ a_{22} \ \dots \ a_{2n} ; \dots ; a_{m1} \ a_{m2} \ \dots \ a_{mn}]$$

$$A=[a_{11} \ a_{12} \ \dots \ a_{1n} ; a_{21} \ a_{22} \ \dots \ a_{2n} ; \dots ; a_{m1} \ a_{m2} \ \dots \ a_{mn}]$$

On the other hand, a vector $V=(v_1, v_2, \dots, v_n)$ is introduced as a special case of a matrix with a single row (i.e. a matrix of dimension $1 \times n$) in the following form:

$$V=[v_1, v_2, \dots, v_n]$$

$$V=[v_1 \ v_2 \ \dots \ v_n]$$

MATLAB includes commands that allow you to perform the most common symbolic and numerical operations with matrices. The following table shows the most important such operations.

MATLAB implements commands for the majority of known matrix decompositions and enables you to work with eigenvalues and eigenvectors with ease. The syntax for the most common commands is presented in the following table.

The matrix commands presented above enable you to work with vector spaces, linear applications and quadratic forms. Using these commands one can determine dependence and linear dependence of sets of vectors, change bases and work in general in two and three-dimensional vector geometry. We illustrate these applications in the following examples.

As a first example we determine whether the vectors $\{1, 2, -3, 4\}$, $\{3, -1, 2, 1\}$, $\{1, -5.8, -7\}$, $\{2, 3, 1, -1\}$ are linearly independent.

$$A=[1,2,-3,4;3,-1,2,1;1,-5,8,-7;2,3,1,-1]$$

A =

1 2 -3 -4

3 -1 2 1

1 -5 8 -7

2 3 1 -1

det(A)

ans =

0

As the determinant of the matrix having the vectors as rows is zero, the vectors are linearly independent.

As a second example, we determine if the set of three vectors of \mathbb{R}^4 $\{\{1,2,2,1\},\{3,4,4,3\},\{1,0,0,1\}\}$ are linearly independent.

B = [1,2,2,1;3,4,4,3;1,0,0,1]

B =

1 2 2 1

3 4 4 3

1 0 0 1

rank (B)

ans =

2

Since we have three vectors in \mathbb{R}^4 , they would be linearly independent if the rank of the matrix having these vectors as rows was 3. However, since this rank is 2, the vectors are linearly dependent.

As a third example we find the dimension and a basis of the linear subspace generated by the vectors $\{\{2,3,4,-1,1\},\{3,4,7,-2,-1\},\{1,3,-1,1,8\},\{0,5,5,-1,4\}\}$.

To find the dimension of the linear space we calculate the rank of the matrix formed by the vectors that generate it. That rank will be the required dimension.

$$A=[2,3,4,-1,1;3,4,7,-2,-1;1,3,-1,1,8;0,5,5,-1,4]$$

A =

$$2 \quad 3 \quad 4 \quad -1 \quad 1$$

$$3 \quad 4 \quad 7 \quad -2 \quad -1$$

$$1 \quad 3 \quad -1 \quad 1 \quad 8$$

$$0 \quad 5 \quad 5 \quad -1 \quad 4$$

rank (A)

ans =

3

Thus the dimension of the linear space is 3, and a basis will be formed by the row vectors corresponding to any non-singular minor of order 3 of the matrix A.

$$\det([3 \ 4 \ 7; 1 \ 3 \ -1; 0 \ 5 \ 5])$$

ans =

75

Thus a basis will be formed by the vectors $\{\{3,4,7,-2,-1\}, \{1,3, - 1, 1, 8\}, \{0,5,5,-1,4\}\}$.

As a fourth example will check if the vectors $\{\{2,3, - 1\}, \{0,0,1\}, \{2,1,0\}\}$ form a basis in \mathbb{R}^3 and find the components of the vector $\{3,5,1\}$ in terms of this basis.

Given that these are three vectors in three-dimensional space, a sufficient condition for them to form a basis in \mathbb{R}^3 is that the determinant of the matrix having these vectors as rows is non-zero.

$$\det([2,3,-1;0,0,1;2,1,0])$$

ans =

4

The vectors form a basis. To find the components of the vector $\{3,5,1\}$ in terms of this basis, we do the following:

$$\text{inv}([2,0,2;3,0,1;-1,1,0]) * [3,5,1]'$$

ans =

1.7500

2.7500

-0.2500

In our fifth example we consider the the bases of \mathbb{R}^3 defined as $B = \{\{1,0,0\}, \{-1, 1, 0\}, \{0,1, -1\}\}$ and $B1 = \{\{1,0, -1\}, \{2,1,0\}, \{-1, 1, 1\}\}$, find the change of basis matrix of B into $B1$, and calculate the components of the B -basis vector $\{2,1,3\}$ in base $B1$.

The operations to be carried out are as follows:

$$B = [1,0,0;-1,1,0;0,1,-1]$$

B =

1 0 0

-1 1 0

0 1 -1

$$B1 = [1, 0, -1; 2, 1, 0; -1, 1, 1]$$

B1 =

1 0 -1

2 1 0

-1 1 1

$$A = \text{inv}(B1') * B'$$

A =

-0.5000 1.5000 2.5000

0.5000 -0.5000 -0.5000

-0.5000 1.5000 1.5000

To find the components of the base-B vector {2,1,3} in base-B1 using the change of basis matrix A, we perform the following operation:

$$\text{inv}(B1') * B'[2,1,3]'$$

ans =

8

-1

5

For our sixth example we find the scalar triple product of the vectors $\{\{1,1,2\}, \{0,1,0\}, \{0,1,1\}\}$ and calculate the area of the triangle whose vertices have coordinates the points (0,0), (5,1) and (3,7).

$$\text{dot}([1,1,2], \text{cross}([0,1,0], [0,1,1]))$$

ans =

1

$$(1/2) * \text{det}([0,0,1; 5,1,1; 3,7,1])$$

ans =

16

As our seventh example we consider a linear transformation of \mathbb{R}^5 to \mathbb{R}^3 whose matrix with respect to the canonical bases is as follows:

$$\begin{pmatrix} 0 & -3 & -1 & -3 & -1 \\ -3 & 3 & -3 & 3 & -1 \\ 2 & 2 & -1 & 1 & 2 \end{pmatrix}$$

We find a basis for its kernel (and hence its dimension) and find the image of the vectors $\{4,2,0,0,-6\}$ and $\{1.2, -1, -2, 3\}$. We also find a basis for the image of the transformation.

$$A=[0,-3,-1,-3,-1;-3,3,-3,-3,-1;2,2,-1,1,2]$$

A =

0 -3 -1 -3 -1

-3 3 -3 -3 -1

2 2 -1 1 2

null(A)

ans =

-0.5397 - 0.1251

-0.2787 - 0.2942

-0.0266 - 0.6948

0.0230 0.6021

0.7936 - 0.2292

These two column vectors form a basis for the null space of A . Thus the kernel of the transformation has dimension 2. Two previous output vectors

form the core of the nucleus and therefore the dimension of the kernel is 2.

To find the image of any column vector v via the linear transformation we simply compute $A*v$.

$$A*[4\ 2\ 0\ 0\ -6]'$$

ans =

0

0

0

$$A*[1\ 2\ -1\ -2\ 3]'$$

ans =

-2

9

11

The dimension of the image of the linear transformation is equal to the rank of A .

$$\text{rank}(A)$$

ans =

3

Thus the dimension of the image of the transformation is 3, and a basis of the image will be given by any three linearly independent columns of A .

$$\det([0\ -3\ -2;\ -3\ 3\ 2;\ -1\ -3\ -1])$$

ans =

-9

Therefore, the vectors $\left\{ \begin{bmatrix} 0 \\ -3 \\ -2 \end{bmatrix}; \begin{bmatrix} -3 \\ 3 \\ 2 \end{bmatrix}; \begin{bmatrix} -1 \\ -3 \\ -1 \end{bmatrix} \right\}$ form a basis of the image.

As an eighth example we consider the linear transformation f between two vector subspaces U and V of real three-dimensional space, such that $f(a, b, c) = (a + b, b + c, a + c)$, for (a, b, c) in U . We find the matrices corresponding to the transformations f , f^5 , and $e f$.

To find the matrix of f , we find the images of the canonical basis vectors under f :

f

$f =$

$[a + b, b + c, a + c]$

$\text{subs}(f, \{a, b, c\}, \{1, 0, 0\})$

ans =

1 0 1

$\text{subs}(f, \{a, b, c\}, \{0, 1, 0\})$

ans =

1 1 0

$\text{subs}(f, \{a, b, c\}, \{0, 0, 1\})$

ans =

0 1 1

The matrix A associated with the linear transformation f will then have as columns the images of the basis vectors found above. Thus: $A =$

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

The matrix associated to f^5 will be A^5 and the matrix associated to $e f$ will be $e A$.

$$A = ([1 \ 0 \ 1; 1 \ 1 \ 0; 0 \ 1 \ 1])'$$

A =

$$1 \ 1 \ 0$$

$$0 \ 1 \ 1$$

$$1 \ 0 \ 1$$

$$A^5$$

ans =

$$11 \ 10 \ 11$$

$$11 \ 11 \ 10$$

$$10 \ 11 \ 11$$

$$\text{expm}(A)$$

ans =

$$3.1751 \ 2.8321 \ 1.3819$$

$$1.3819 \ 3.1751 \ 2.8321$$

$$2.8321 \ 1.3819 \ 3.1751$$

For our ninth example we classify the bilinear form $f:U \times V \rightarrow R$ and the quadratic form $g:U \rightarrow R$ defined as follows:

$$f[(a, b, c), (d, e, f)] = (a, b, c) \begin{pmatrix} 1 & -2 & 0 \\ 0 & 0 & 4 \\ -1 & 0 & 3 \end{pmatrix} \begin{pmatrix} d \\ e \\ f \end{pmatrix}$$

$$g(a, b, c) = (a, b, c) \begin{pmatrix} 1 & -1 & 3 \\ -1 & 1 & -3/2 \\ 3 & -3/2 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$A = [1, -2, 0; 0, 0, 4; -1, 0, -3]$$

A =

1-2-0

0 0 4

-1 0 - 3

det (A)

ans =

8

As the determinant of the matrix of f is non-zero, the bilinear form is regular non-degenerate.

$$B = [1, -1, 3; -1, 1, -3/2; 3, -3/2, 4]$$

B =

1.0000 - 1.0000 3,0000

-1.0000 1.0000 - 1.5000

3,0000 - 1.5000 4.0000

To classify the quadratic form, we calculate its diagonal determinants.

det(B)

ans =

-2.2500

$\det([1,-1;-1,1])$

ans =

0

It turns out that the quadratic form is negative semi-definite.

We can also obtain the classification via the eigenvalues of the matrix of the quadratic form.

A quadratic form is defined to be positive if and only if all its eigenvalues are strictly positive. A quadratic form is defined to be negative if and only if all its eigenvalues are strictly negative.

A quadratic form is positive semi-definite if and only if all its eigenvalues are non-negative. A quadratic form is negative semi-definite if and only if all its eigenvalues are not positive.

A quadratic form is indefinite if there are both positive and negative eigenvalues.

$\text{eig}(B)$

ans =

-0.8569

0.4071

6.4498

There are positive and negative eigenvalues, so the quadratic form is indefinite.

Exercise 1. Consider the following matrix:

$$M = \begin{pmatrix} 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \\ 1/5 & 1/6 & 1/7 \end{pmatrix}$$

Find its transpose, its inverse, its determinant, its rank, its trace, its singular values, its condition number, its norm, M^3 , e^M , $\log(M)$ and \sqrt{M} .

» $M = [1/3.1/4.1/5; 1/4.1/5.1/6; 1/5.1/6.1/7]$

$M =$

0.3333 0.2500 0.2000

0.2500 0.2000 0.1667

0.2000 0.1667 0.1429

» transpose = M'

transpose =

0.3333 0.2500 0.2000

0.2500 0.2000 0.1667

0.2000 0.1667 0.1429

» inverse = $\text{inv}(M)$

inverse =

1. 0e + 003 *

0.3000 - 0.9000 0.6300

-0.9000 2.8800 - 2.1000

0.6300 - 2.1000 1.5750

To verify that this is indeed the inverse, we multiply it by M to obtain the identity matrix of order 3:

» $M \cdot \text{inv}(M)$

ans =

1.0000 0.0000 0.0000

0.0000 1.0000 0.0000

0.0000 0.0000 1.0000

» determinant = $\det(M)$

determinant =

2.6455e-006

» rank = $\text{rank}(M)$

rank =

3

» trace = $\text{trace}(M)$

trace =

0.6762

» vsingular = $\text{svd}(M)$

vsingular =

0.6571

0.0189

0.0002

» condition = $\text{cond}(M)$

condition =

3.0886e + 003

For the calculation of the norm, we find the standard norm, the 1-norm, the infinity norm and the Frobenius norm:

» norm(M)

ans =

0.6571

» norm(M,1)

ans =

0.7833

» norm(M,inf)

ans =

0.7833

» norm(M,'fro')

ans =

0.6573

» M ^ 3

ans =

0.1403 0.1096 0.0901

0.1096 0.0856 0.0704

0.0901 0.0704 0.0578

» logm(M)

ans =

-2.4766 2.2200 0.5021

2.2200 - 5.6421 2.8954

0.5021 2.8954 - 4.7240

» sqrtm(M)

ans =

0.4631 0.2832 0.1966

0.2832 0.2654 0.2221

0.1966 0.2221 0.2342

To calculate e M we try the eigenvalue, Padé approximant, Taylor expansion and condition number variants:

» expm(M)

ans =

1.4679 0.3550 0.2863

0.3550 1.2821 0.2342

0.2863 0.2342 1.1984

» expm1(M)

ans =

1.4679 0.3550 0.2863

0.3550 1.2821 0.2342

0.2863 0.2342 1.1984

» expm2(M)

ans =

1.4679 0.3550 0.2863

0.3550 1.2821 0.2342

0.2863 0.2342 1.1984

» expm3(M)

ans =

1.4679 0.3550 0.2863

0.3550 1.2821 0.2342

0.2863 0.2342 1.1984

As we see, all methods yield the same exponential matrix.

Exercise 2. Consider the following matrix:

$$M = \begin{pmatrix} 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \\ 1/5 & 1/6 & 1/7 \end{pmatrix}$$

Find its transpose, its inverse, its determinant, its rank, its trace, its singular values, M3, log (M) and sqrt (M).

$$M = \text{sym} ('[1/3,1/4,1/5; 1/4,1/5,1/6; 1/5,1/6,1/7]')$$

M =

[1/3, 1/4, 1/5]

[1/4, 1/5, 1/6]

[1/5, 1/6, 1/7]

$$\text{transpose} = M'$$

$$\text{transpose} =$$

$$[1/3, 1/4, 1/5]$$

$$[1/4, 1/5, 1/6]$$

$$[1/5, 1/6, 1/7]$$

$$\text{inverse} = \text{inv}(M)$$

$$\text{inverse} =$$

$$[300, -900, 630]$$

$$[-900, 2880 - 2100]$$

$$[630, -2100, 1575]$$

$$\text{determinant} = \text{det}(M)$$

$$\text{determinant} =$$

$$1/378000$$

$$\text{rank} = \text{rank}(M)$$

$$\text{rank} =$$

$$3$$

$$\text{trace} = \text{trace}(M)$$

$$\text{trace} =$$

$$71/105$$

$$\text{vsingular} = \text{svd}(M)$$

$$\text{vsingular} =$$

$$\frac{(12703/88200 - (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3})/2 - 1030177 / (99574272 * (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3}) - (1030177 / (49787136 * (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3})) - ((102103^{1/2} * i) / 49787136 + 1045602865/351298031616)^{1/3}}{2}^{1/2}$$

$$\frac{(12703/88200 - (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3})/2 - 1030177 / (99574272 * (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3}) + (3^{1/2} * (1030177 / (49787136 * (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3})) - ((102103^{1/2} * i) / 49787136 + 1045602865/351298031616)^{1/3}) * i}{2}^{1/2}$$

$$\frac{(1030177 / (49787136 * (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3})) + (1045602865/351298031616 + (102103^{1/2} * i) / 49787136)^{1/3} + 12703/88200)^{1/2}$$

M^3

ans =

$$[10603/75600, 1227/11200, 26477/294000]$$

$$[1227/11200, 10783/126000, 74461/1058400]$$

$$[26477/294000, 74461/1058400, 8927/154350]$$

log(M)

ans =

$$[-\log(3), -\log(4), -\log(5)]$$

$$[-\log(4), -\log(5), -\log(6)]$$

$$[-\log(5), -\log(6), -\log(7)]$$

sqrt(M)

ans =

$[3^{1/2}/3, 1/2, 5^{1/2}/5]$

$[^{1/2}/5, 1/2, 5, 6^{1/2}/6]$

$[5^{1/2}/5, ^{1/2}/6, 6, 7^{1/2}/7]$

Exercise 3. Consider the following symbolic matrix:

$$A = \begin{bmatrix} a & b & c \\ 3c & a-3c & b \\ 3b & -3b+3c & a-3c \end{bmatrix}$$

Calculate A' , A^{-1} , determinant (A), trace (A), rank (A), inv (A) and A^2 .

$A = \text{sym}(['a,b,c; 3c,a-3c,b; 3b,-3b+3c,a-3c'])$

$A =$

$[a, \quad b, \quad c]$

$[3 * c, a - 3 * c, b]$

$[3 * b, 3 * c - 3 * b - 3 * c]$

$\text{transpose}(A)$

$\text{ans} =$

$[a, 3 * c, 3 * b]$

$[(b) - 3 * c, 3 * c - 3 * b]$

$[c, b, a - 3 * c]$

$\text{pretty}(\text{det}(A))$

$3^2 \quad 2 \quad 2 \quad 3 \quad 2 \quad 3$

$a^3 - 6 a^2 c + 3 a b^2 - 9 a b c + 9 a^2 c + 3 b^3 + 9 b^2 c + 9 c^3$

pretty(trace(A))

3 a - 6 c

rank(A)

ans =

3

simplify(inv(A))

ans =

$$\left[\frac{(a^2 - 6ac + 3b^2 - 3bc + 9c^2)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3}, \frac{-(ab - 3c^2)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3}, \frac{(b^2 + 3c^2 - ac)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3} \right]$$

$$\left[\frac{(3b^2 + 9c^2 - 3ac)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3}, \frac{-(c(3a + 3b) - a^2)}{9bc^2 - 6a^2c + a(3b^2 - 9bc + 9c^2) + a^3 + 3b^3 + 9c^3}, \frac{-(ab - 3c^2)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3} \right]$$

$$\left[\frac{-(3ab - 9c^2)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3}, \frac{(3b^2 + 3ab - 3ac)}{c^2(9a + 9b) - c(6a^2 + 9ba) + 3ab^2 + a^3 + 3b^3 + 9c^3}, \frac{-(c(3a + 3b) - a^2)}{9bc^2 - 6a^2c + a(3b^2 - 9bc + 9c^2) + a^3 + 3b^3 + 9c^3} \right]$$

pretty(simplify(A^2))

+-

-+

2	2	2	2	
a + 6 b c,	3 c - 6 b c + 2 a b,	b - 3 c + 2 a c		
2	2	2	2	

[4cosh(a)sinh(a), 2cosh(a)^2 + 2sinh(a)^2]

pretty(simplify(M1))

+-		-+
	2	
	4 sinh (a) + 2,	2 sinh (2 a)
	2	
	2 sinh(2 a),	4 sinh (a) + 2
+-		-+

M2=A²-B²

M2 =

[0, 0]

[0, 0]

To calculate A n and B n , we first find their successive powers to try to see the general rule:

[simplify(A²),simplify(A³),simplify(A⁴)]

ans =

[cosh(2a), sinh(2a), cosh(3a), sinh(3a), cosh(4a), sinh(4a)]

[sinh(2a), cosh(2a), sinh(3a), cosh(3a), sinh(4a), cosh(4a)]

[simplify(B²),simplify(B³),simplify(B⁴)]

ans =

[cosh(2a), sinh(2a), sinh(3a), cosh(3a), cosh(4a), sinh(4a)]

[sinh(2a), cosh(2a), cosh(3a), sinh(3a), sinh(4a), cosh(4a)]

The form of the general rule is now evident:

$$A^n = B^n = \begin{bmatrix} \cosh(na) & \sinh(na) \\ \sinh(na) & \cosh(na) \end{bmatrix}$$

simplify(inv(A))

ans =

[cosh(a), -sinh(a)]

[-sinh(a), cosh(a)]

simplify(inv(B))

ans =

[-sinh(a), cosh(a)]

[cosh(a), -sinh(a)]

simplify(det(A))

ans =

1

simplify(det(B))

ans =

-1

simplify(trace(A))

ans =

2*cosh(a)

```
simplify(trace(B))
```

```
ans =
```

```
2*sinh(a)
```

```
simplify(exp(A))
```

```
ans =
```

```
[ exp(cosh(a)), exp(sinh(a))]
```

```
[ exp(sinh(a)), exp(cosh(a))]
```

```
simplify(exp(B))
```

```
ans =
```

```
[ exp(sinh(a)), exp(cosh(a))]
```

```
[ exp(cosh(a)), exp(sinh(a))]
```

Exercise 5. Consider a normally distributed random square matrix A of order 3. Calculate the diagonal matrix D with diagonal entries the eigenvalues of A and the matrix V whose columns are the corresponding eigenvectors (if the output is complex, transform it to real form).

Find the balanced matrix of A , and real and complex forms of its Schur decomposition.

Find the coefficients of the characteristic polynomial of the matrix A .

Calculate the upper triangular matrix R of the same dimension as the matrix A , the permutation matrix E and the orthogonal matrix Q such that $A * E = Q * R$ and check the result.

Consider the Hessenberg matrix B of A and calculate the diagonal matrix D of generalized eigenvalues of A and B , and a matrix V whose columns are the corresponding eigenvectors, satisfying $A * V = B * V * D$. Also calculate the vector of generalized singular values of A and B .

```
A=randn(3)
```

A =

-0.4326 0.2877 1.1892

-1.6656 -1.1465 -0.0376

0.1253 1.1909 0.3273

[V,D] = eig(A)

V =

0.2827 0.4094 - 0.3992i 0.4094 + 0.3992i

0.8191 -0.0950 + 0.5569i - 0.0950 - 0.5569i

-0.4991 0.5948 0.5948

D =

-1.6984 0 0

0 0.2233 + 1.0309i 0

0 0 0.2233 - 1.0309i

[V,D] = cdf2rdf(V,D)

V =

0.2827 0.4094 - 0.3992

0.8191 -0.0950 0.5569

-0.4991 0.5948 0

D =

-1.6984 0 0

0 0.2233 1.0309

0 -1.0309 0.2233

$$[T, B] = \text{balance}(A)$$

T =

1 0 0

0 1 0

0 0 1

B =

-0.4326 0.2877 1.1892

-1.6656 -1.1465 -0.0376

0.1253 1.1909 0.3273

$$[U, T] = \text{schur}(A)$$

U =

0.2827 0.2924 0.9136

0.8191 -0.5691 -0.0713

-0.4991 -0.7685 0.4004

T =

-1.6984 0.2644 - 1.2548

0 0.2233 0.7223

0 -1.4713 0.2233

$$[U, T] = \text{rsf2csf}(U, T)$$

U =

0.2827 -0.7482 + 0.1678i 0.2395 - 0.5242i

0.8191 0.0584 - 0.3266i - 0.4661 + 0.0409i

-0.4991 -0.3279 - 0.4410i - 0.6294 - 0.2298i

T =

-1.6984 1.0277 + 0.1517i 0.2165 + 0.7201i

0 0.2233 + 1.0309i 0.7490 - 0.0000i

0 0 0.2233 - 1.0309i

poly(A)

ans =

1.0000 1.2517 0.3540 1.8895

Next we calculate the upper triangular matrix R of the same dimension as the matrix A of the above example, the permutation matrix E and the orthogonal matrix Q such that $A * E = Q * R$ and check the result.

[Q, R, E] = qr (A)

Q =

-0.2507 0.4556 - 0.8542

-0.9653 - 0.0514 0.2559

0.0726 0.8887 0.4527

R =

1.7254 1.1211 - 0.2380

0 1.2484 0.8346

0 0 - 0.8772

E =

1 0 0

0 1 0

0 0 1

$A * E$

ans =

-0.4326 0.2877 1.1892

-1.6656 -1.1465 -0.0376

0.1253 1.1909 0.3273

$Q * R$

ans =

-0.4326 0.2877 1.1892

-1.6656 -1.1465 -0.0376

0.1253 1.1909 0.3273

Thus the matrices do indeed satisfy $A * E = Q * R$.

Now we consider the Hessenberg matrix B of A , we calculate the diagonal matrix D of generalized eigenvalues of A and B , and a matrix V whose columns are the corresponding eigenvectors, so that $A * V = B * V * D$. In addition we calculate the vector of generalized singular values of A and B .

$B = \text{hess}(A)$

$B =$

-0.4326 - 0.1976 1.2074

1.6703 - 1.2245 0.1544

0 - 1.0741 0.4053

$[V, D] = \text{eig}(A, B)$

V =

0.0567 1.0000 1.0000

-0.0354 - 0.4998 0.5297

-1.0000 0.4172 0.3785

D =

1.0000 0 0

0 - 0.4722 0

0 0 - 2.1176

A * V

ans =

-1.2245 - 0.0803 0.1699

-0.0137 - 1.1082 - 2.2872

-0.3649 - 0.3334 0.8801

B * V * D

ans =

-1.2245 - 0.0803 0.1699

-0.0137 - 1.1082 - 2.2872

-0.3649 - 0.3334 0.8801

sigma = gsvd (A, B)

sigma =

0.2874

1.0000

3.4799

Exercise 6. Consider the 3×3 matrix A below:

$$\begin{pmatrix} 1 & 5 & -2 \\ -7 & 3 & 1 \\ 2 & 2 & -2 \end{pmatrix}$$

Find the LU, QR, Cholesky, Schur, Hessenberg and singular value decompositions of A, checking that the results are correct. Also find the pseudoinverse of A.

First, we find the Schur decomposition, checking that the result is correct.

$$A = [1, 5, -2; -7, 3, 1; 2, 2, -2];$$

$$[U, T] = \text{schur}(A)$$

U =

-0.0530 - 0.8892 - 0.4544

-0.9910 - 0.0093 0.1337

0.1231 - 0.4573 0.8807

T =

2.4475 - 5.7952 - 4.6361

5.7628 0.3689 2.4332

0 0 - 0.8163

Now we check that $U * T * U' = A$ and that $U * U' = \text{eye}(3)$:

$$[UTU', U*U']$$

ans =

1.0000 5.0000 - 2.0000 1.0000 0.0000 0.0000

-7.0000 3,0000 1.0000 0.0000 1.0000 0.0000

2.0000 2.0000 - 2.0000 0.0000 0.0000 1.0000

Now we find the LU, QR, Cholesky, Hessenberg and singular value decompositions, checking the results for each case:

» [L, U, P] = lu(A)

L =

1.0000 0 0

-0.1429 1.0000 0 lower triangular matrix

-0.2857 0.5263 1.0000

U =

-7.0000 3,0000 1.0000

0 5.4286 - 1.8571 upper triangular matrix

0 0 - 0.7368

P =

0 1 0

1 0 0

0 0 1

[PA,LU]

ans =

-7 3 1 -7 3 1

5 1 -2 1 5 -2 we have P * A = L * U

2 2 -2 2 2 -2

$$[Q, R, E] = \text{qr}(A)$$

Q =

-0.1361 - 0.8785 - 0.4579

0.9526 - 0.2430 0.1831

-0.2722 - 0.4112 0.8700

R =

-7.3485 1.6330 1.7691

0 - 5.9442 2.3366 upper triangular matrix

0 0 - 0.6410

E =

1 0 0

0 1 0

0 0 1

$$[AE, QR]$$

ans =

1.0000 5.0000 - 2.0000 1.0000 5.0000 - 2.0000

-7.0000 3.0000 1.0000 - 7.0000 3.0000 1.0000

2.0000 2.0000 - 2.0000 2.0000 2.0000 - 2.0000

Then, $A * E = Q * R$.

$$R = \text{chol}(A)$$

??? Error using == > chol

Matrix must be positive definite.

An error message is returned because the matrix is not positive definite.

$$[H, p] = \text{hess}(A)$$

P =

1.0000 0 0

0 - 0.9615 0.2747

0 0.2747 0.9615

H =

1.0000 - 5.3571 - 0.5494

7.2801 1.8302 - 2.0943

0 - 3.0943 - 0.8302

$$[P * H * P', P' * P]$$

ans =

1.0000 5.0000 - 2.0000 1.0000 0 0

-7.0000 3.0000 1.0000 0 1.0000 0

2.0000 2.0000 - 2.0000 0 0 1.0000

Then $PHP' = A$ and $P'P = I$.

$$[U, S, V] = \text{svd}(A)$$

U =

-0.1034 - 0.8623 0.4957

-0.9808 0.0056 - 0.1949

0.1653 - 0.5064 - 0.8463

S =

7.8306 0 0

0 6.2735 0 diagonal matrix

0 0 0.5700

V =

0.9058 - 0.3051 0.2940

-0.3996 - 0.8460 0.3530

-0.1411 0.4372 0.8882

USV'

ans =

1.0000 5.0000 - 2.0000

-7.0000 3,0000 1.0000 we see that USV'= A

2.0000 2.0000 - 2.0000

Now we calculate the pseudoinverse of the matrix A :

X=pinv(A)

X =

0.2857 - 0.2143 - 0.3929

0.4286 - 0.0714 - 0.4643

0.7143 - 0.2857 - 1.3571

[AXA,XAX]

ans =

1.0000 5.0000 - 2.0000 0.2857 - 0.2143 - 0.3929

-7.0000 3,0000 1.0000 0.4286 - 0.0714 - 0.4643

2.0000 2.0000 - 2.0000 0.7143 - 0.2857 - 1.3571

Thus, we see that $AXA = A$, $XAX = X$.

Exercise 7. Consider the following matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) \\ 0 & \sin(a) & \cos(a) \end{bmatrix}$$

Calculate its eigenvalues, its characteristic polynomial, its Jordan canonical form and its singular values.

We start by defining the matrix A as a symbolic matrix:

```
» A=sym('[100;0cos(a)-sin(a);0sin(a)cos(a)]')
```

A =

```
[ 1,  0,  0]
```

```
[0, cos (a) - sin (a)]
```

```
[0, sin (a), cos (a)]
```

```
» eigensys(A)
```

ans =

```
[
```

```
cos (a) + 1/2 * (- 4 * sin (a) ^ 2) ^(1/2)]
```

```
cos (a) - 1/2 * (- 4 * sin (a) ^ 2) ^(1/2)]
```

```
» pretty(simple(poly(A)))
```

```
3 2 2
```

```
x - 2 x cos (a) + x - x + 2 x cos (a) - 1
```

» jordan(A)

ans =

[1, 0, 0]

[0, cos(a) + 1/2 * (-4 * without(a)^2)^(1/2), 0]

[0, 0, cos(a) - 1/2 * (-4 * without(a)^2)^(1/2)]

» simple(svd(A))

ans =

[1]

[1/2 * (4 * cos(a-comp(a)) + 2 * (-2 + 2 * cos(2 * a-2 * conj(a))))^(1/2))
^(1/2)]

[1/2 * (4 * cos(a-comp(a)) - 2 * (-2 + 2 * cos(2 * a-2 * conj(a))))^(1/2))
^(1/2)]

Exercise 8. Diagonalize the symmetric matrix whose rows are the vectors:

(3, -1, 0), (-1, 2, -1), (0, -1, 3).

Find the similarity transform V , confirm that the eigenvalues of the original matrix are the diagonal elements of the diagonal matrix and that the diagonal matrix and the original matrix are similar.

We calculate the diagonal matrix J of A , which will consist of the eigenvalues of A on its diagonal and at the same time find the similarity transform V . To do this, we use the command $[V, J] = \text{jordan}(A)$:

$$A = [3, -1, 0; -1, 2, -1; 0, -1, 3]$$

A =

3 -1 0

-1 -2 -1

0 -3 -1

[V, J] = jordan(A)

V =

1 -1 -1

2 0 -1

1 -1 -1

J =

1 0 0

0 3 0

0 0 4

We now confirm that the diagonal matrix J has the eigenvalues of A on its diagonal:

eig(A)

ans =

1.0000

3.0000

4.0000

The matrices A and J are similar because the matrix V satisfies the relationship $V^{-1} * A * V = J$:

inv(V) * A * V

ans =

1.0000 0 - 0.0000

0 3.0000 0

0 0 4.0000

Exercise 9. Find a diagonal matrix similar to each of the following arrays:

$$A = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & -\sin(a) \\ -1 & 0 & \cos(a) \\ -\sin(a) & \cos(a) & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix}.$$

Diagonalize the matrices. Find the characteristic polynomial of each matrix.

$$A = \text{sym}(['0,-r,q;r,0,-p;-q,p,0']);$$

$$[V, J] = \text{jordan}(A)$$

V =

$$\left[\frac{(q \sqrt{-p^2 - q^2 - r^2})^{1/2}}{(p^2 + q^2) - (p r)} \frac{1}{(p^2 + q^2)}, \frac{-(q \sqrt{-p^2 - q^2 - r^2})^{1/2}}{(p^2 + q^2) - (p r)} \frac{1}{(p^2 + q^2)}, \frac{p}{r} \right]$$

$$\left[\frac{-(p \sqrt{-p^2 - q^2 - r^2})^{1/2}}{(p^2 + q^2) - (q r)} \frac{1}{(p^2 + q^2)}, \frac{(p \sqrt{-p^2 - q^2 - r^2})^{1/2}}{(p^2 + q^2) - (q r)} \frac{1}{(p^2 + q^2)}, \frac{q}{r} \right]$$

$$[1, 1, 1]$$

J =

$$\left[\frac{-(-p^2 - q^2 - r^2)^{1/2}}{2}, 0, 0 \right]$$

$$\left[0, \frac{(-p^2 - q^2 - r^2)^{1/2}}{2}, 0 \right]$$

$$\left[0, 0, 0 \right]$$

Now, we analyze the matrix B :

$$\gg B = \text{sym}(['0,1,-sin(a); -1, 0, cos(a)-sin(a), cos(a), 0'])$$

» J = simple (jordan(B))

J =

[0, 0, 0]

[0, 0, 0]

[0, 0, 0]

This shows that the matrix B has a single eigenvalue zero of multiplicity 3. In addition, the kernel of $B - 0 * \text{eye}(3) = B$ has dimension less than three, as the determinant of B is zero. In particular, it has dimension one (as we see, calculating a basis with the command null (B) below) . As the multiplicity and the dimension of the kernel differ, we conclude that the matrix B is not diagonalizable:

» null(B)

ans =

[cos (a)]

[sin (a)]

[1]

We have calculated a basis for the kernel of B , which is formed by a single vector, hence the dimension of the kernel of B is 1:

» det(B)

ans =

0

We now analyze the matrix C :

C = sym ('[cos(a), - sin(a); sin(a), cos(a)]');

[V, J] = jordan (C)

V =

[1/2, 1/2]

[1/2, -1/2]

J =

[cos (a) - sin (a) * i, 0]

[0, cos (a) + sin (a) * i]

We now calculate the characteristic polynomial of the three matrices:

pretty (poly (A))

3 2 2 2

x + (p + q + r) x

pretty(simple(sym(poly(B))))

3

x

pretty(simple(sym(poly(C))))

2

x - 2 cos (a) x + 1

Exercise 10. Find the eigenvalues of the Wilkinson matrix of order 8, the magic square magic(8) of order 8 and the Rosser matrix.

[eig (wilkinson (8)), eig (rosser), eig (magic (8))]

ans =

1. 0e + 003 *

-0.0010 -1.0200 0.2600

0.0002 0.0000 0.0518

0.0011 0.0001 -0.0518

0.0017 1.0000 0.0000

0.0026 1.0000 -0.0000 + 0.0000i

0.0028 1.0199 -0.0000 - 0.0000i

0.0042 1.0200 -0.0000 + 0.0000i

0.0043 1.0200 -0.0000 - 0.0000i

We note that the Wilkinson matrix has pairs of eigenvalues that are close, but not equal. The Rosser matrix has a double eigenvalue, three nearly equal eigenvalues, dominant eigenvalues of the opposite sign, a zero eigenvalue and a small, non-zero eigenvalue.

Exercise 11. Consider the linear transformation f between two vector subspaces U (contained in \mathbb{R}^3) and V (contained in \mathbb{R}^4), such that for any point (a, b, c) in U :

$$f(a, b, c) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Find the kernel and the image of f .

$$A = ([1,0,0;0,0,0;0,0,1;0,0,0]);$$

The kernel is the set of vectors of U with null image:

$$\text{null}(A)$$

ans =

0

1

0

Hence the kernel is the set of vectors $\{0, b, 0\}$ with varying b . Moreover, the kernel obviously has dimension 1, since it has $\{0, 1, 0\}$ as a basis.

rank(A)

ans =

2

The dimension of the image of f must match the rank of the matrix A , which we have just seen is 2. The columns of a two column submatrix of A which has a non-singular two by two submatrix will form a basis of the image of f .

det([1,0;0,1])

ans =

1

Thus a basis of the image of f is given by $\{\{1, 0, 0, 0\}, \{0, 0, 1, 0\}\}$.

Exercise 12. Given the quadratic form $g:U \rightarrow R$ defined as follows

$$g(a, b, c) = (a, b, c) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

classify and find its reduced equation, its rank and its signature.

To classify the quadratic form, we calculate its diagonal determinants.

$$G = [1, 0, 0; 0, 2, 2; 0, 2, 2]$$

G =

1 0 0

0 2 2

0 2 2

det (G)

ans =

0

det([1,0;0,2])

ans =

2

The quadratic form is degenerate positive semidefinite.

To find the reduced equation we diagonalize the matrix.

J = jordan(G)

J =

0 0 0

0 1 0

0 0 4

The reduced quadratic form equation is then:

$$h(x, y, z) = (x, y, z) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = y^2 + 4z^2$$

rank(J)

ans =

2

The rank of the quadratic form is 2, since the rank of the matrix is 2. The signature is also 2, since the number of positive terms in the diagonal matrix is 2.