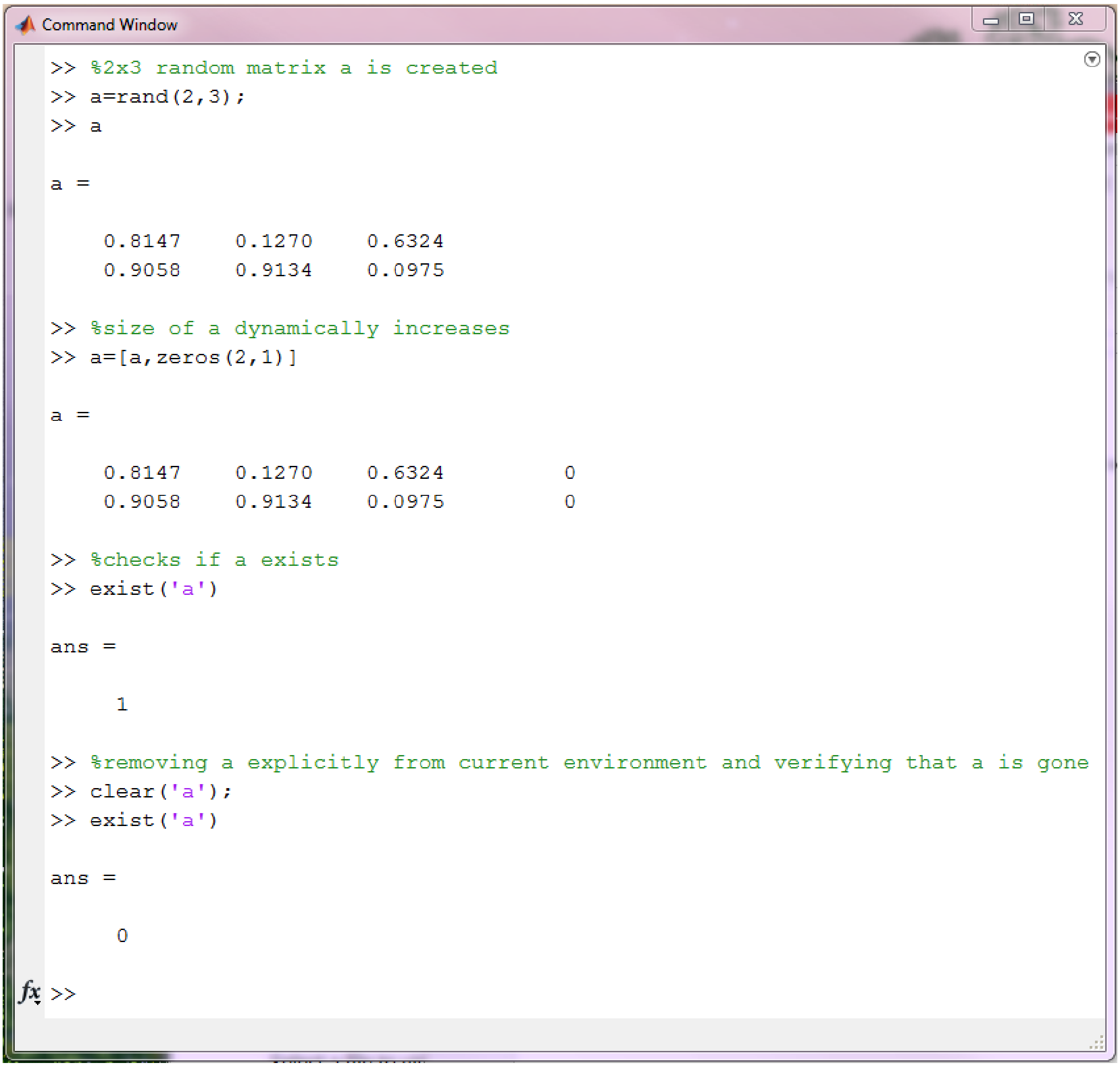
Introduction to Matlab (Practice 1)

Text comments

The text commentary in Matlab is a string that starts with the characters %. The text comments can be used in the command window, and in the text file script. The string after characters % is not perceived as a command and pressing the Enter key activates the following command line.

The Matlab language was initially devoted to matrix operations, and scientific and engineering applications were its main target. But over time, it has considerably evolved, and currently the Matlab language includes powerful operators for manipulating a large class of basic objects.



```
>> %2x3 random matrix a is created
>> a=rand(2,3);
>> a

a =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975

>> %size of a dynamically increases
>> a=[a,zeros(2,1)]

a =

    0.8147    0.1270    0.6324         0
    0.9058    0.9134    0.0975         0

>> %checks if a exists
>> exist('a')

ans =

     1

>> %removing a explicitly from current environment and verifying that a is gone
>> clear('a');
>> exist('a')

ans =

     0

fx >>
```

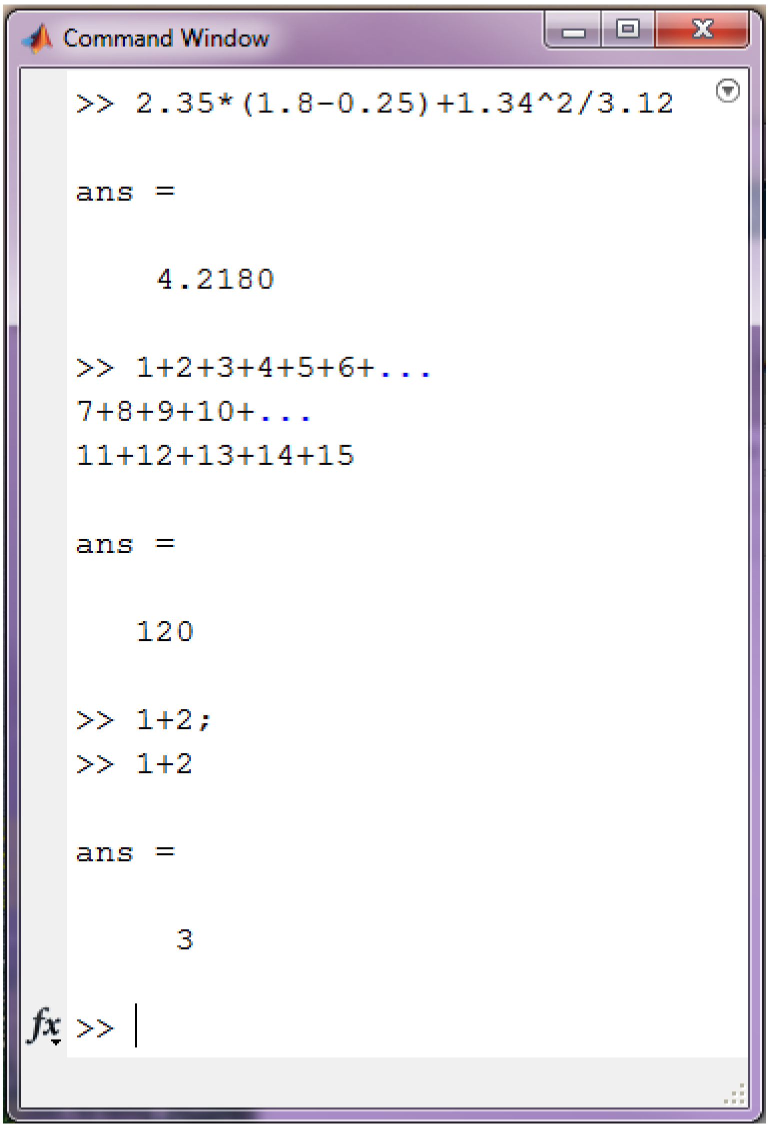
Elementary mathematical expressions

To perform simple arithmetic operations in Matlab use the following operators: + addition, - subtraction, * multiplication, / division from left to right, \ division from right to left, ^ exponentiation.

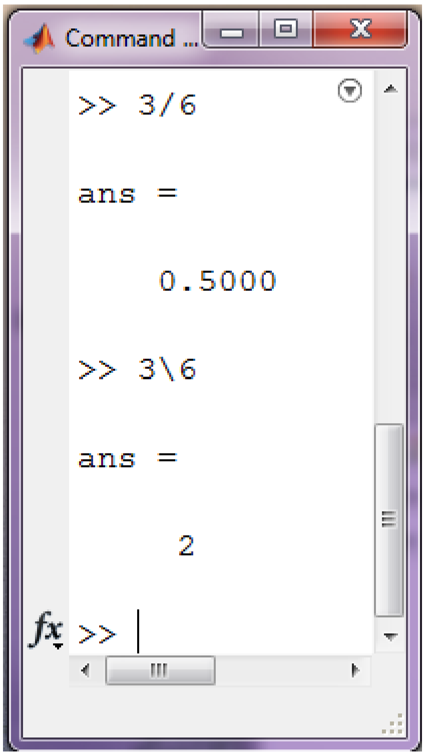
To Calculate the value of an arithmetic expression is possible if you enter it in the command prompt and pressing Enter. The work area will be the result of:

If the calculated expression is too long before pressing Enter should score three or more points. This will mean a continuation of the command line:

If a semicolon ";" placed at the end of the expression, the calculated result is not an output, so activate the next command line:



```
>> 2.35*(1.8-0.25)+1.34^2/3.12

ans =

    4.2180

>> 1+2+3+4+5+6+...
7+8+9+10+...
11+12+13+14+15

ans =

   120

>> 1+2;
>> 1+2

ans =

     3

fx >> 
```



```
>> 3/6

ans =

    0.5000

>> 3\6

ans =

     2

fx >> 
```

Variables in Matlab

In Matlab workspace, you can define variables and then use their in expressions. Any variable must be defined prior to be used in formulas and expressions. To determine the variable you need to dial name of the variable, then the symbol "=" and the value of the variable. Here the sign of equality is the assignment operator, the effect of which does not differ from similar operators of programming languages.

That is, if written as:

Variable_name = value_expressions

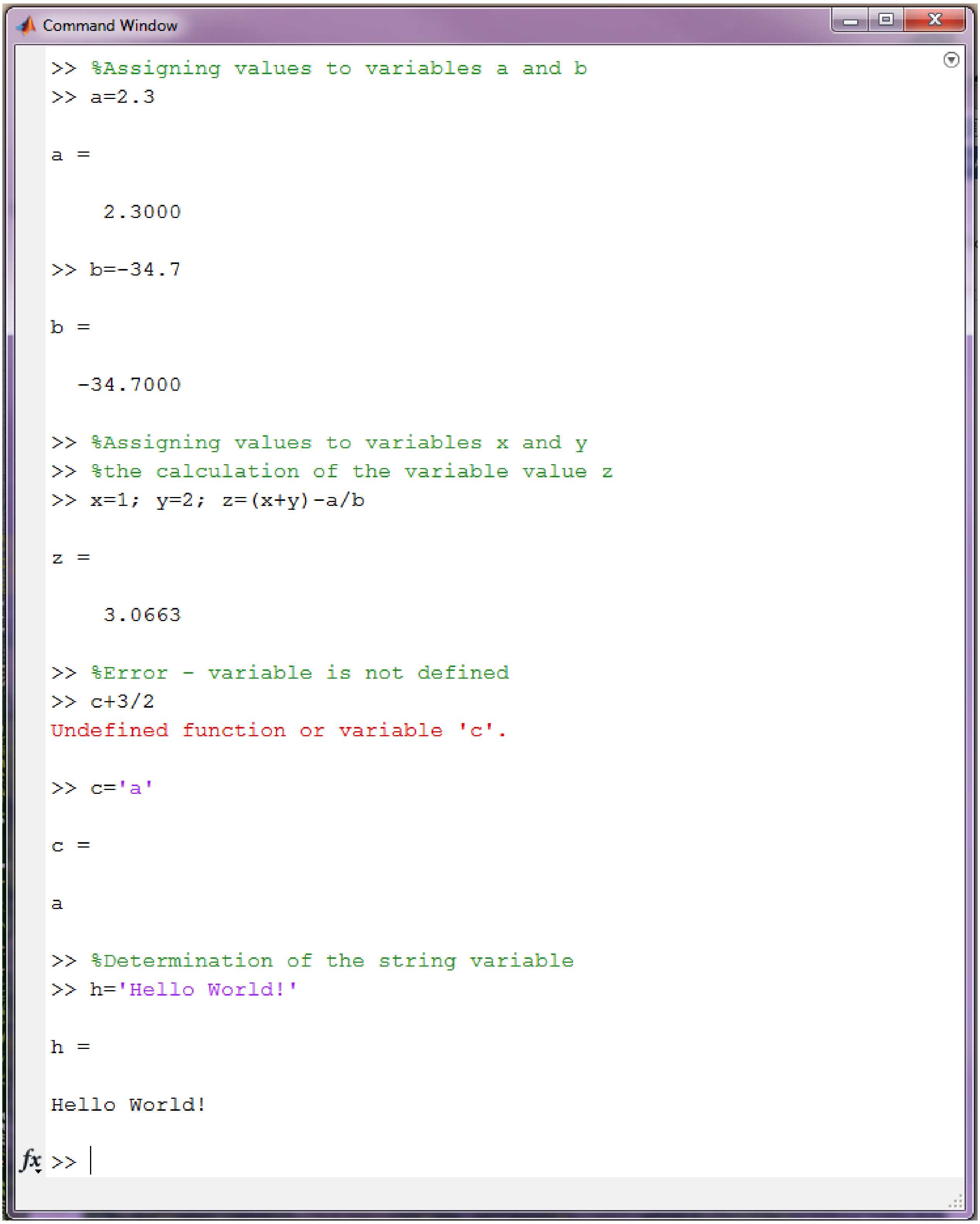
the variable which~~ose~~ name is indicated on the left, will be recorded as the value of the expression specified on the right.

The variable name must not be the same name as built in procedures, functions, and built-in system variables and can contain up to 24 characters.

The system distinguishes between large and small letters in variable names. That is ABC, abc, Abc, aBc - the names of different variables.

The expression on the right-hand side of an assignment operator can be a number, an arithmetic expression, a string of characters or a character expression. If it is a character or a string variable, the expression on the right side of an assignment must be taken in single quotes (‘’).

If the symbol ";" at the end of the expression is missing, the result is displayed as a name of the variable, and its value. The presence of the symbol ";" passes control to the next command line. This allows the use of variable names to record interim results in the computer's memory:

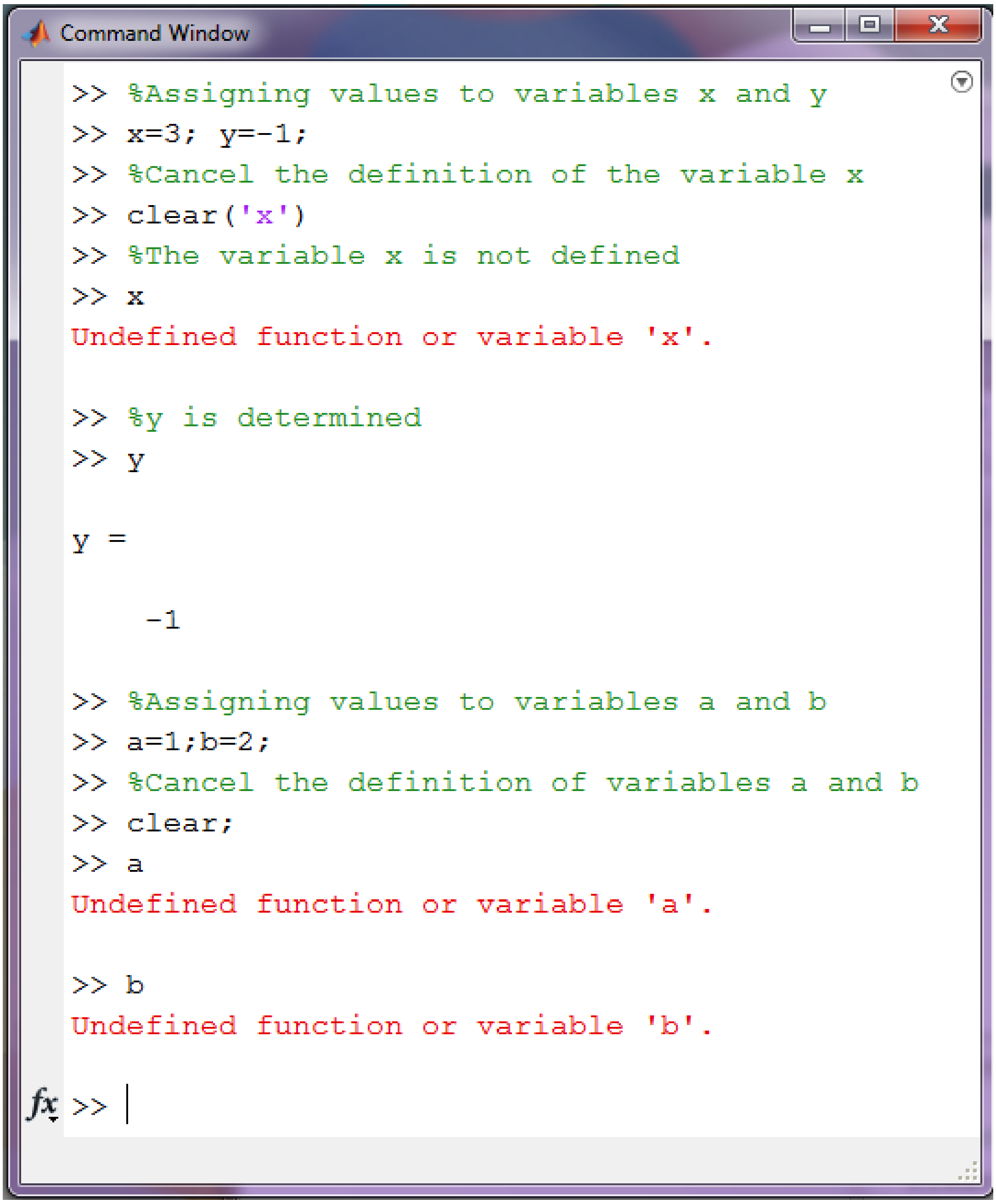


```
>> %Assigning values to variables a and b
>> a=2.3

a =

    2.3000

>> b=-34.7

b =

  -34.7000

>> %Assigning values to variables x and y
>> %the calculation of the variable value z
>> x=1; y=2; z=(x+y)-a/b

z =

    3.0663

>> %Error - variable is not defined
>> c+3/2
Undefined function or variable 'c'.

>> c='a'

c =

a

>> %Determination of the string variable
>> h='Hello World!'

h =

Hello World!

>> 
```

To clear the value of a variable, you can use the command

clear variable_name;

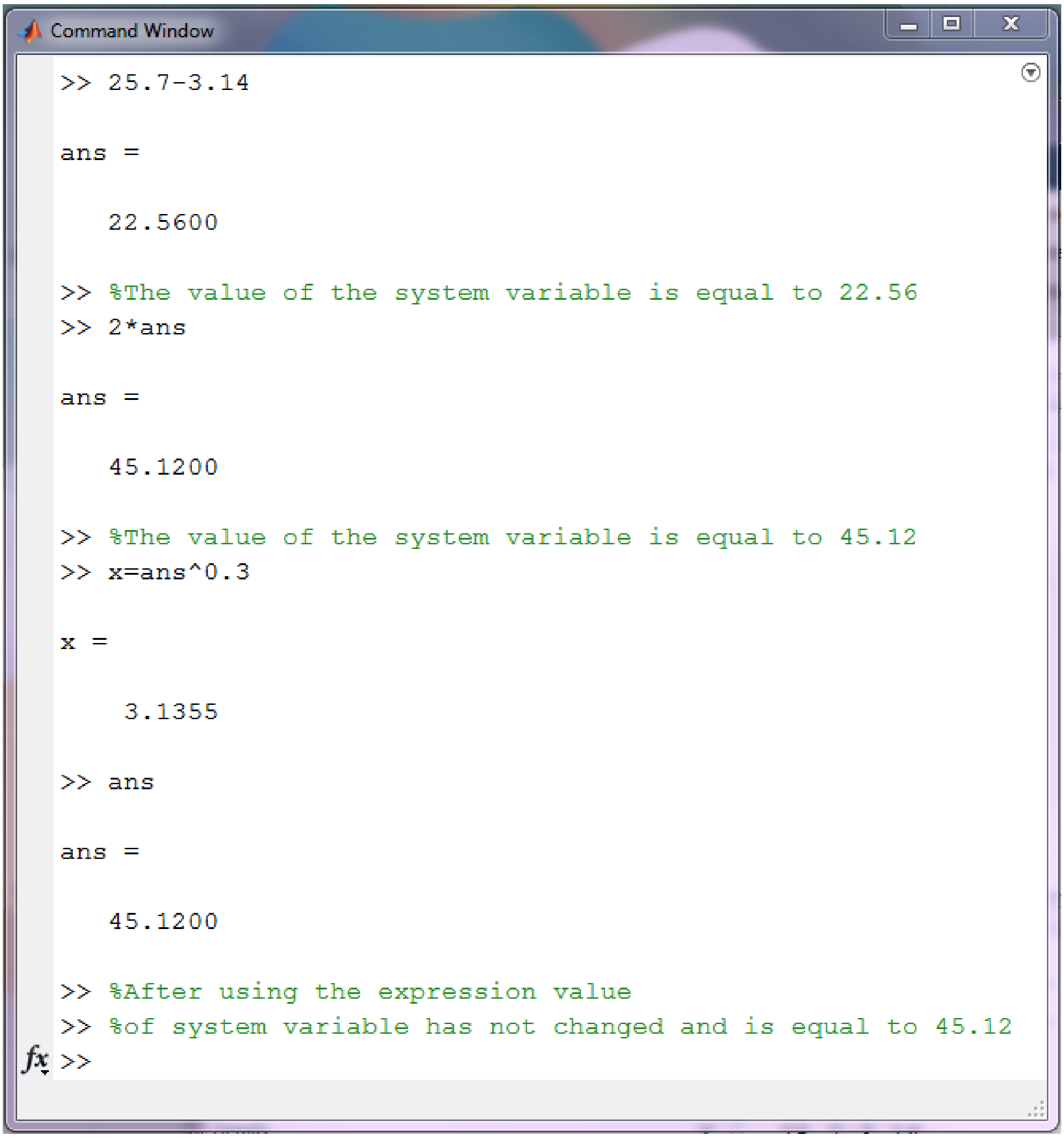
which cancels the value of all the variables of the session. The following are examples of the use of this command:



```
>> %Assigning values to variables x and y
>> x=3; y=-1;
>> %Cancel the definition of the variable x
>> clear('x')
>> %The variable x is not defined
>> x
Undefined function or variable 'x'.

>> %y is determined
>> y

y =

    -1

>> %Assigning values to variables a and b
>> a=1;b=2;
>> %Cancel the definition of variables a and b
>> clear;
>> a
Undefined function or variable 'a'.

>> b
Undefined function or variable 'b'.

fx >> 
```

System variables Matlab

If the command does not contain the mark assignment, the default calculated value is assigned to a special system variable ans. Moreover, the resulting value can be used in the following calculations, but it is important to remember that the value of ans changes after each call to the team without an assignment statement:



```
>> 25.7-3.14

ans =

   22.5600

>> %The value of the system variable is equal to 22.56
>> 2*ans

ans =

   45.1200

>> %The value of the system variable is equal to 45.12
>> x=ans^0.3

x =

    3.1355

>> ans

ans =

   45.1200

>> %After using the expression value
>> %of system variable has not changed and is equal to 45.12
>>
```

The result of the last operation unsigned assignment is stored in variable ans.

Note that some numerical constants are predefined in Matlab. In particular, π is pi, and e, the base of the natural log, is exp(1).

Other system variables in Matlab

i – the imaginary unit ($\sqrt{-1}$)

pi – number Pi π=3.1415927

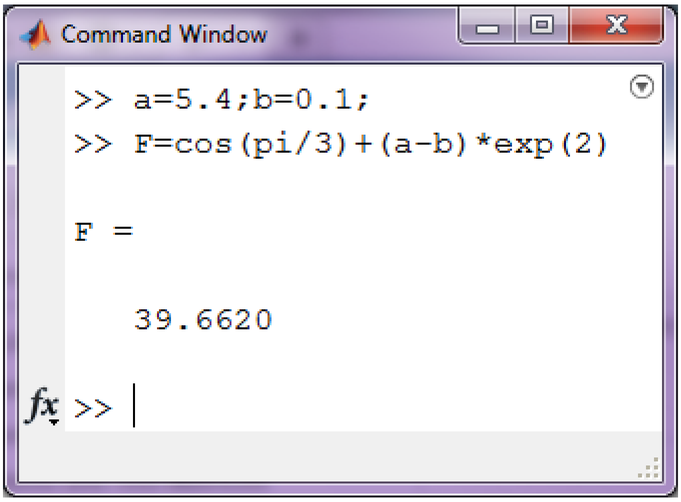
exp(a) – number e^a=2.7182818^a

inf – Machine symbol of infinity (∞)

nan – undefined result (0/0, ∞/∞, etc.)

eps – conditional zero %eps = 2.220D-16

All of these variables can be used in mathematical expressions:



```
>> a=5.4;b=0.1;
>> F=cos(pi/3)+(a-b)*exp(2)

F =

   39.6620

>> 
```

The functions in Matlab

All functions used in Matlab, can be divided into two classes:

- built-in
- user-defined

In general, the function call in Matlab is:

Variable_name = function_name (var1, [var2, …])

where variable_name - the variable which the results of the functions are saved in; This option can be omitted, then the value computed by the function will be assigned to the system variable ans;

function_name - the name of the built-in or previously created user-defined function;

var1, var2, ... - a list of function arguments.

Elementary math functions

The Package Scilab is equipped with a sufficient number of all kinds of built-in functions. There only the elementary mathematical functions are used most often:

Elementary math functions

Function	**Description of the function**
Trigonometric	

sin(x)
cos(x)
tan(x)
cotg(x)
asin(x)
acos(x)
atan(x)
	sine of number x
	cosine of number x
	tangent of umber x
	cotangent of number x
	arcsine of number x
	arccosine of number x
	arctangent of number x

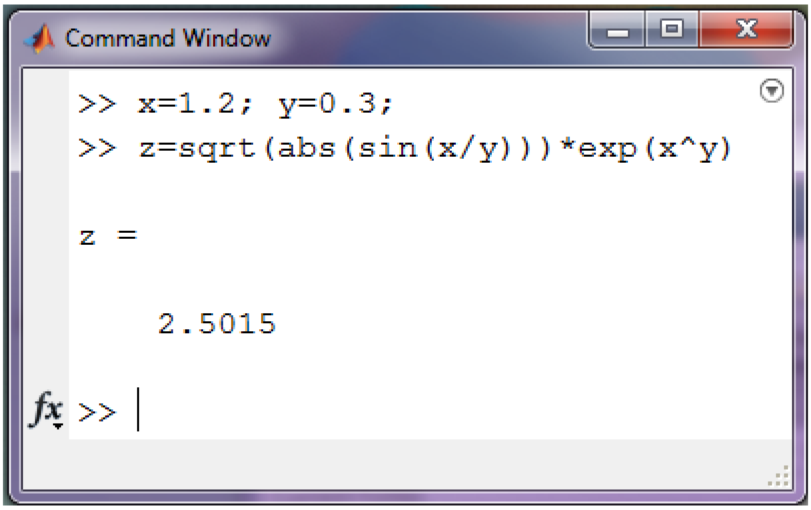
Exponential	

exp(x)
log(x)
	Exponent of number x
	natural logarithm of number x

Others	

sqrt(x)
abs(x)
log10(x)
log2(x)
	Square root of x
	Module number x
	Decimal logarithm of the number x
	Logarithm to the base two of the number x

The Example of calculating the value of the expression $z = \sqrt{\left|\sin\left(\frac{x}{y}\right)\right|} \cdot e^{x^y}$, x=1.2 and y=0.3.



```
>> x=1.2; y=0.3;
>> z=sqrt(abs(sin(x/y)))*exp(x^y)

z =

    2.5015

>> 
```

User-defined functions

For a start, get acquainted with the script file. The script file is a list of teams Matlab, stored on the disk. For the preparation, editing and debugging file scripts there is a special Editor, which can be called by click the **New Script** button on the **Home** tab. As a result of this command a new script file will be created. By default, it is named Untitled1.m.

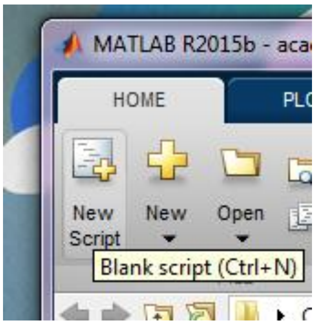
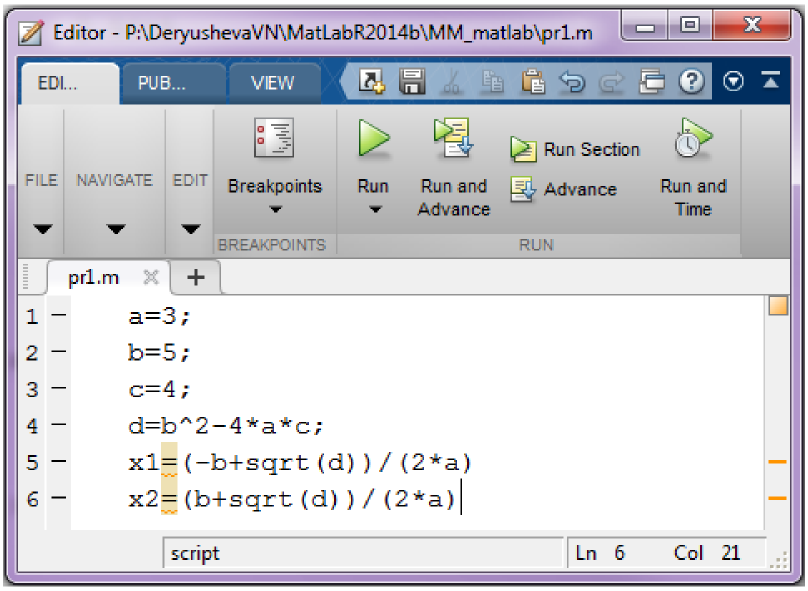


Figure 1.5.

Enter the text in the editor window, the script file is carried out according to the rules adopted for the teams Matlab. Fig. 1.6 shows an example of command input for solving a quadratic equation

$$3x^2 + 5x + 4 = 0.$$

It is easy to note that the semicolon ";" is placed after those teams that do not require output values.



```
a=3;
b=5;
c=4;
d=b^2-4*a*c;
x1=(-b+sqrt(d))/(2*a)
x2=(b+sqrt(d))/(2*a)
```

Figure 1.6.

To save and execute the script file you need to execute the command F5 or the Run button on the **Home** tab (fig. 1.7) .

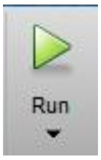
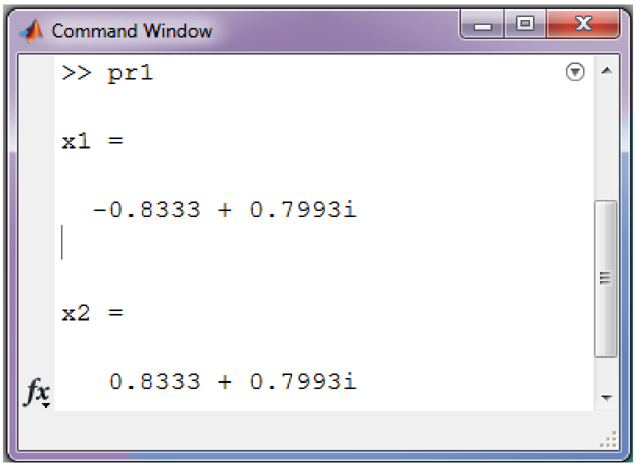


Figure 1.7.

If the information is saved for the first time, the window appears Save file As... Enter a name in the File Name field and click the Save button will store the information in the editor window. The script file is saved with the extension .m.

To view the result, you need to enter the command window variables x1, then x2 (fig. 1.8.)



```
>> pr1

x1 =

  -0.8333 + 0.7993i


x2 =

   0.8333 + 0.7993i
```

Figure 1.8.

For the convenience Editor set in one window with the command window (fig.1.10), you need to click ▼ then choose Dock Editor (fig.1.10).

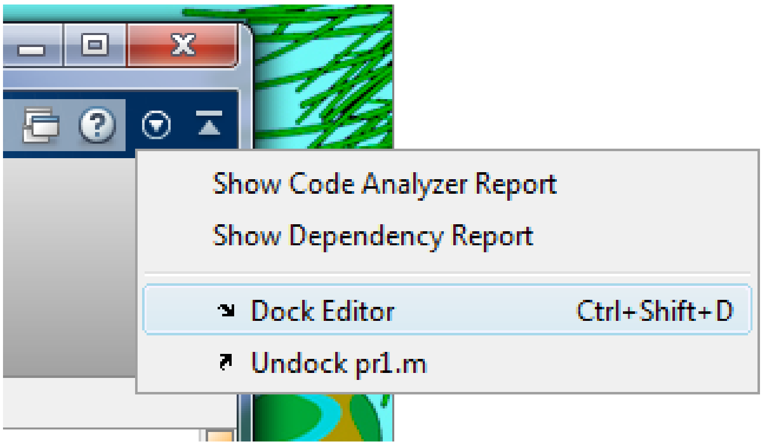
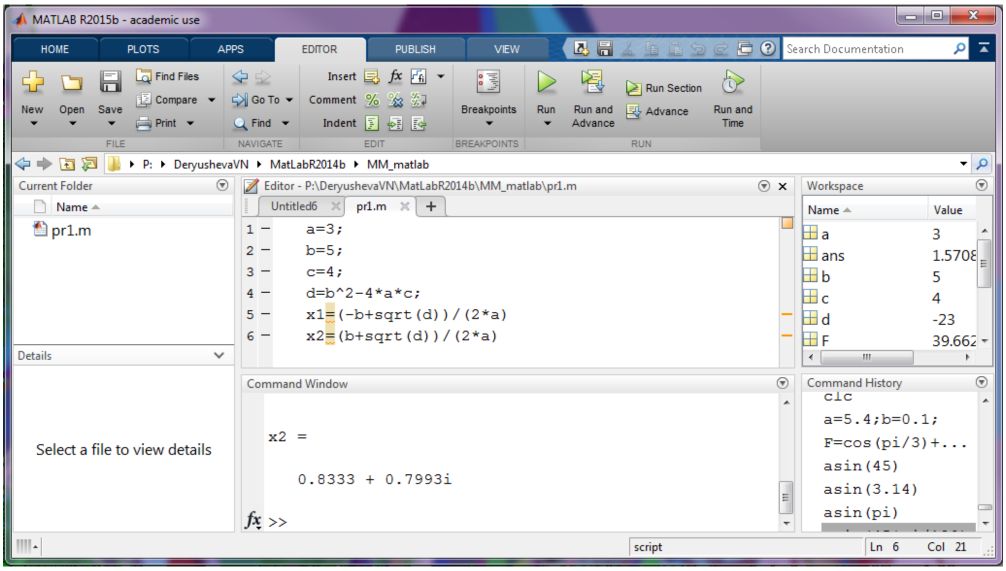


Figure 1.9.

(Video Pr1.001)



```
a=3;
b=5;
c=4;
d=b^2-4*a*c;
x1=(-b+sqrt(d))/(2*a)
x2=(b+sqrt(d))/(2*a)
```

```
x2 =

   0.8333 + 0.7993i

fx >>
```

Name	Value
a	3
ans	1.5708
b	5
c	4
d	-23
F	39.662

```
clc
a=5.4;b=0.1;
F=cos(pi/3)+...
asin(45)
asin(3.14)
asin(pi)
```

Figure 1.10.

$$5x^2 + 12x + 4 = 0.$$

Now back to User-defined Functions. It was not possible to pass the input parameters into the obtained script file , ie, this is the usual list of commands, the system is perceived as a single operator.

Write a Function

The function is usually intended for repeated use, it has no input parameters and executed without preset. Consider a few ways to create functions in Matlab.

The first way, You can create function handles to named and anonymous functions. You can store multiple function handles in an array, and save and load them, as you would any other variable.

What Is a Function Handle?

A function handle is a MATLAB® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from. Typical uses of function handles include:

- Pass a function to another function (often called function functions). For example, passing a function to integration and optimization functions, such as integral and fzero.
- Specify callback functions. For example, a callback that responds to a UI event or interacts with data acquisition hardware.
- Construct handles to functions defined inline instead of stored in a program file (anonymous functions).
- Call local functions from outside the main function.

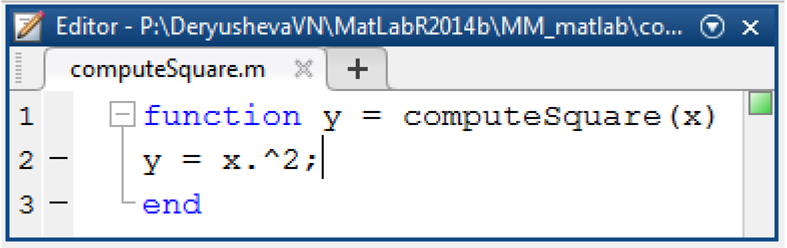
You can see if a variable, h, is a function handle using isa(h,'function_handle').

Creating Function Handles

To create a handle for a function, precede the function name with an @ sign. For example, if you have a function called myfunction, create a handle named f as follows:

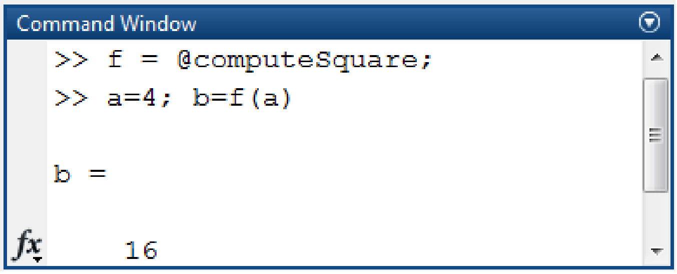
f = @myfunction;

You call a function using a handle the same way you call the function directly. For example, suppose that you have a function named computeSquare, defined as:



```
function y = computeSquare(x)
y = x.^2;
end
```

Create a handle and call the function to compute the square of four.



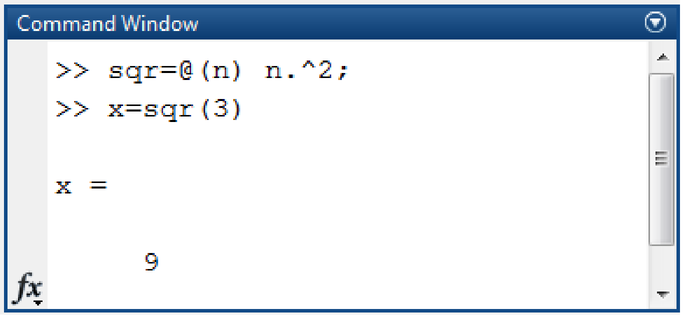
```
>> f = @computeSquare;
>> a=4; b=f(a)

b =

    16
```

Anonymous Functions

You can create handles to anonymous functions. An anonymous function is a one-line expression-based MATLAB function that does not require a program file. Construct a handle to an anonymous function by defining the body of the function, anonymous_function, and a comma-separated list of input arguments to the anonymous function, arglist. The syntax is:

h = @(arglist)anonymous_function

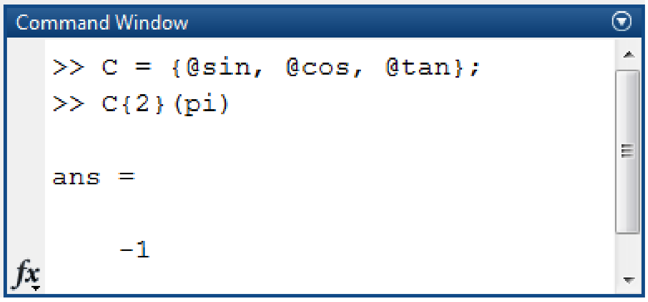
For example, create a handle, sqr, to an anonymous function that computes the square of a number, and call the anonymous function using its handle.



```
>> sqr=@(n) n.^2;
>> x=sqr(3)

x =

     9
```

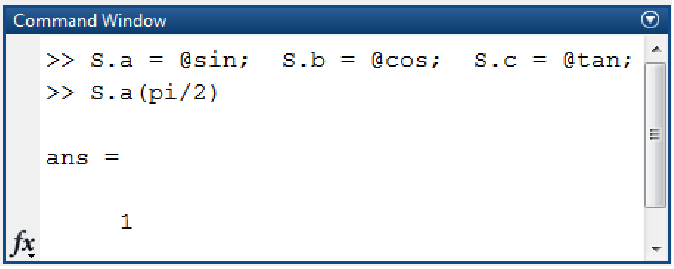
Arrays of Function Handles

You can create an array of function handles by collecting them into a cell or structure array. For example, use a cell array:



```
>> C = {@sin, @cos, @tan};
>> C{2}(pi)

ans =

    -1
```

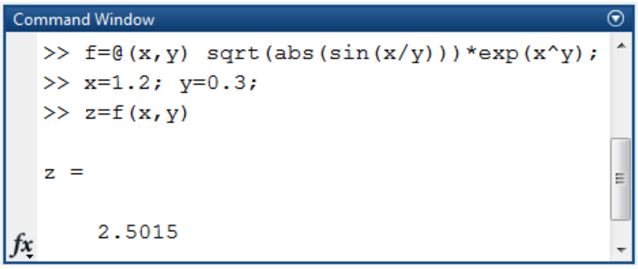
Or use a structure array:



```
>> S.a = @sin;  S.b = @cos;  S.c = @tan;
>> S.a(pi/2)

ans =

     1
```

Here's how to create and apply a Function Handle to evaluate the expression

$$z = \sqrt{\left|\sin\left(\frac{x}{y}\right)\right|} \cdot e^{xy}$$



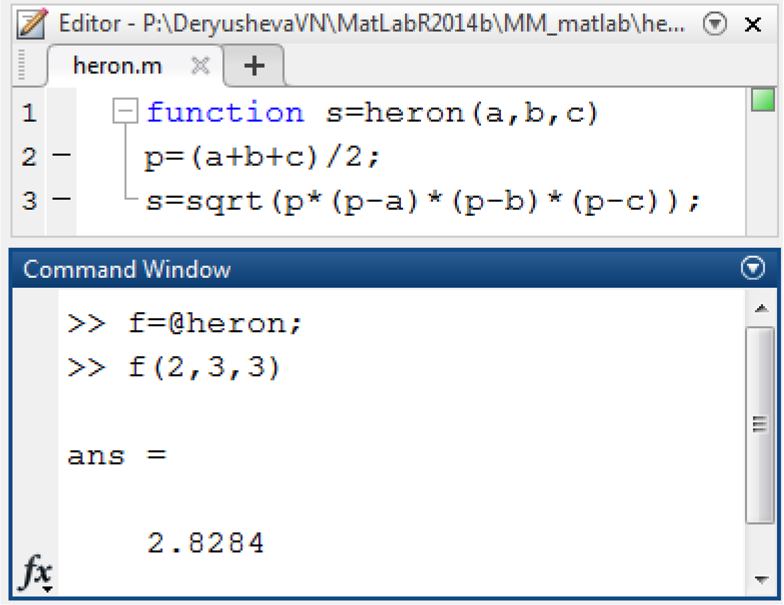
```
>> f=@(x,y) sqrt(abs(sin(x/y)))*exp(x^y);
>> x=1.2; y=0.3;
>> z=f(x,y)

z =

    2.5015
```

Consider the example of creation and application functions, calculates the area of a triangle with sides a, b, and c from the formula of Heron.

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

where

$$p = \frac{a+b+c}{2}$$



```
function s=heron(a,b,c)
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c));
```

```
>> f=@heron;
>> f(2,3,3)

ans =

    2.8284
```

The second way to create an application of the design features. Open a file in a text editor. Within the file, declare the function and add program statements:

function f = fact(n)
f = prod(1:n);

function [name1,..., name N]=function_ name(var_1,...,var_M)

body of function;

where name1, ..., nameN - a list of output parameters, ie the variables that will be assigned to the final result of the calculation;

function name - the name with which the function is invoked,

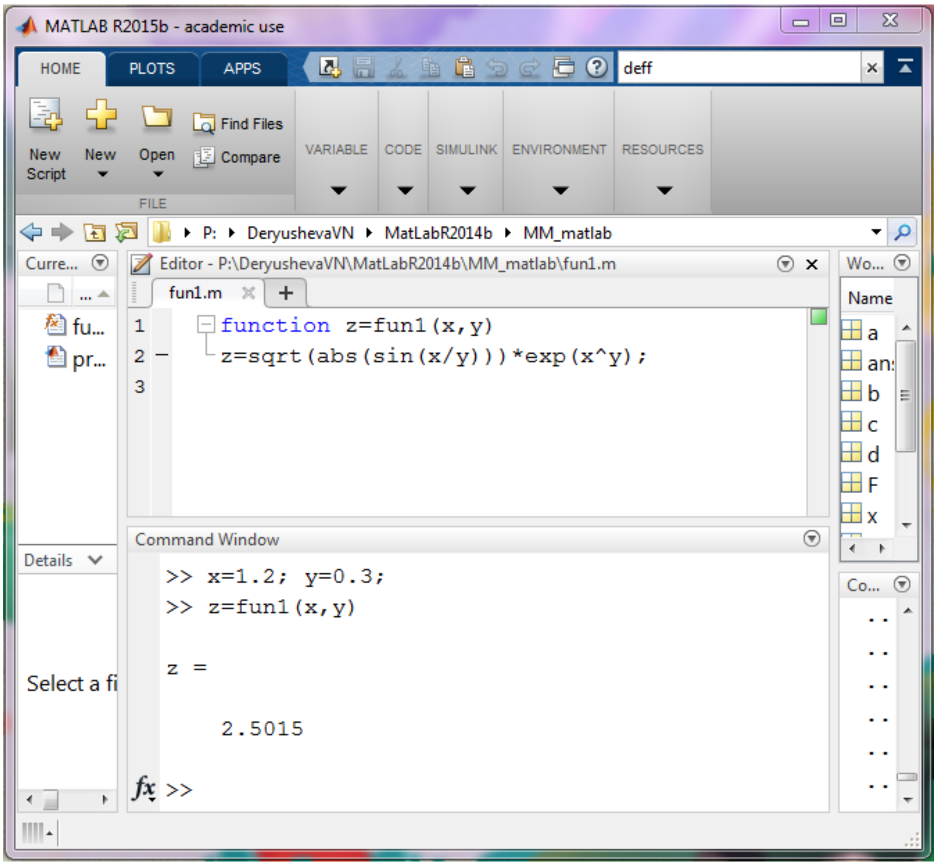
variable_1, ..., variable _M - input parameters.

All variable names inside the function, and the names from the list of input and output parameters perceived by system as local, ie considered defined within the function.

Function play a role in Matlab routines. Therefore, it is advisable to dial them in text editor and save them as separate files. And the file name must necessarily coincide with the name of the function. The file extension-functions are usually assigned m.

Call of the function in the same way as any other built-in system, i.e. from the command line. However, the functions are stored in separate files, should be pre-loaded into the system.

The following is the easiest way to use the operator function. Here's how to create and apply a function to evaluate the expression $z = \sqrt{\left|\sin\left(\frac{x}{y}\right)\right|} \cdot e^{xy}$ (the value of this expression has already been calculated in the above listing). (video Pr1.002)



```
function z=fun1(x,y)
z=sqrt(abs(sin(x/y)))*exp(x^y);
```

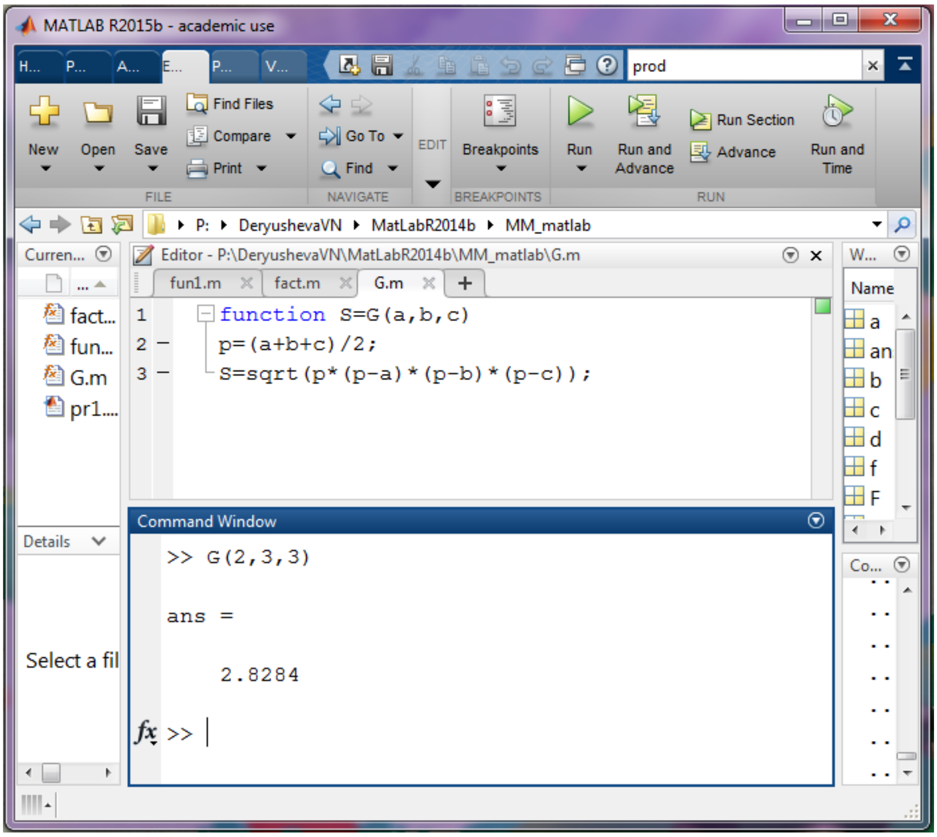
```
>> x=1.2; y=0.3;
>> z=fun1(x,y)

z =

    2.5015

>>
```

Consider the example of creation and application functions, calculates the area of a triangle with sides a, b, and c from the formula of Heron.

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

where

$$p = \frac{a+b+c}{2}$$



```
function S=G(a,b,c)
p=(a+b+c)/2;
S=sqrt(p*(p-a)*(p-b)*(p-c));
```

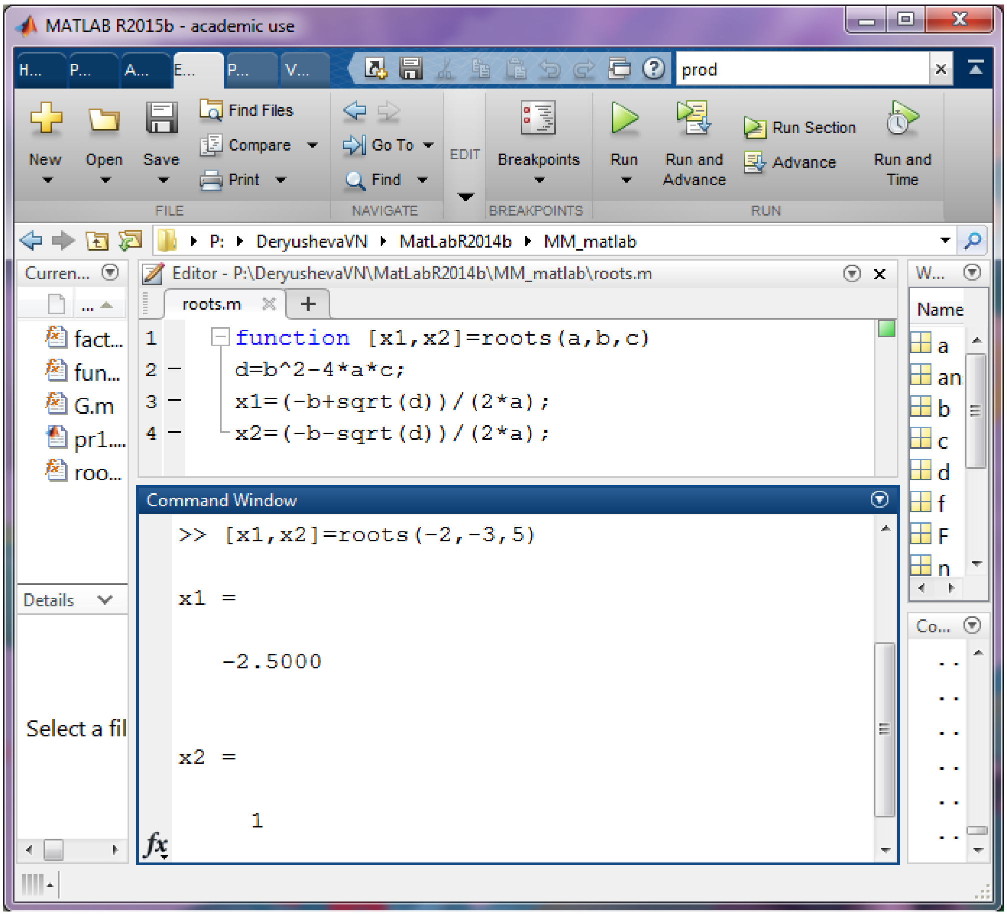
```
>> G(2,3,3)

ans =

    2.8284

>> 
```

The following listing is an example of creation and application functions, which can be used to find the roots of a quadratic equation of the form

$ax^2+bx+c=0$

using formulas

$$D = b^2 - 4ac; x_{1,2} = \frac{-b \pm \sqrt{D}}{2}a$$



```
function [x1,x2]=roots(a,b,c)
d=b^2-4*a*c;
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
```

```
>> [x1,x2]=roots(-2,-3,5)

x1 =

   -2.5000


x2 =

     1
```

As an example, consider the following problem. (Video 1.003)

Task 2.1.

Solve the cubic equation.

Cubic equation

$$ax^3 + bx^2 + cx + d = 0 \tag{1.1}$$

After division by a takes the canonical form:

$$x^3 + rx^2 + sx + t = 0 \tag{1.2}$$

Where

$$r = \frac{b}{a}; \quad s = \frac{c}{a}; \quad t = \frac{d}{a}$$

In equation (1.2) we make the change

$$x = y - \frac{r}{3}$$

and obtained the following equation

$$y^3 + py + q = 0 \tag{1.3}$$

Where

$$p=\frac{(3s-r^2)}{3}, \qquad q=2\frac{r^3}{27}-\frac{rs}{3}+t$$

The number of real roots of the above equation (1.3) depends on the sign of the discriminant $D=\left(\frac{p}{3}\right)^3+\left(\frac{q}{2}\right)^3$ (Table 1.2)

Table 2.2. The number of roots of the cubic equation

The discriminant	The number of real roots	The number of complex roots
$D\geq 0$	1	2
$D<0$	3	-

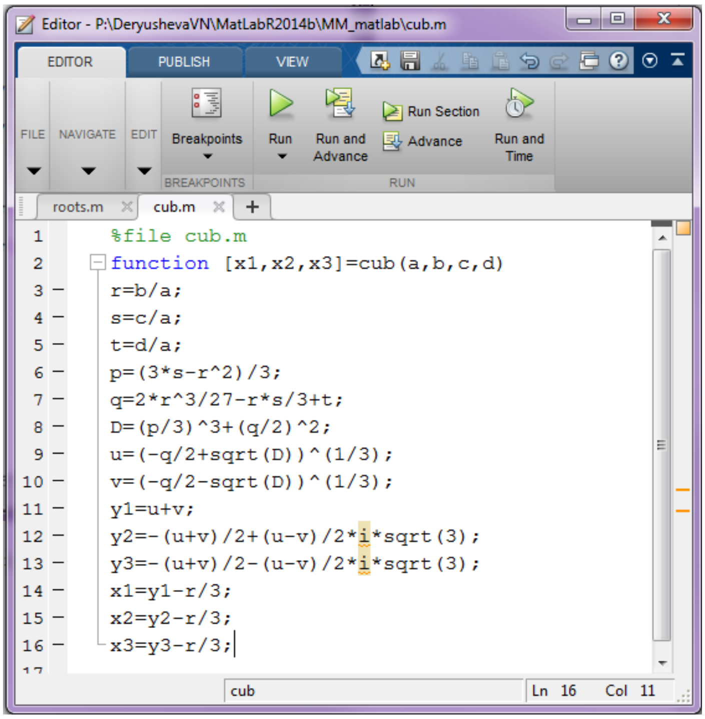
The roots of the above equation can be calculated using the formula Cardano:

$$y_1=u+v, \qquad y_2=\frac{-(u+v)}{2}+\frac{(u-v)}{2}i\sqrt{3}, \qquad y_3=\frac{-(u+v)}{2}-\frac{(u-v)}{2}i\sqrt{3},$$

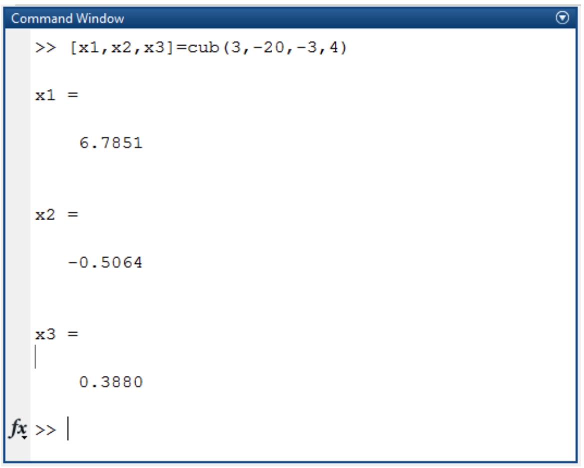
Here

$$u=\sqrt[3]{\frac{-q}{2}+\sqrt{D}}, \qquad v=\sqrt[3]{\frac{-q}{2}-\sqrt{D}},$$

The following is the text of the function that implements the above-described method of solving cubic equations:



```
%file cub.m
function [x1,x2,x3]=cub(a,b,c,d)
r=b/a;
s=c/a;
t=d/a;
p=(3*s-r^2)/3;
q=2*r^3/27-r*s/3+t;
D=(p/3)^3+(q/2)^2;
u=(-q/2+sqrt(D))^(1/3);
v=(-q/2-sqrt(D))^(1/3);
y1=u+v;
y2=-(u+v)/2+(u-v)/2*i*sqrt(3);
y3=-(u+v)/2-(u-v)/2*i*sqrt(3);
x1=y1-r/3;
x2=y2-r/3;
x3=y3-r/3;
```



```
>> [x1,x2,x3]=cub(3,-20,-3,4)

x1 =

    6.7851


x2 =

   -0.5064


x3 =

    0.3880

>> 
```