UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Homework.

Introduction to Matlab: Matrices, Graphics and programming.

1. Objectives

In this first Matlab Practice students will get an overview of the possibilities that this environment can offer.

At the end of this practice, the student will be able to:

1. Think in a vectorial manner using Matlab, and its benefits.
2. Explain the differences between script and function and to know to create them.
3. Create multidimensional matrices and to access at any element of them.

Do not hesitate to use HELP command/menu whenever you need it. Remember that the semicolon, at the end of an expression in the command window, forces the result NOT TO appear albeit it is evaluated.

2. Practice Development

Open the Demo window.

Type *demo* in the MATLAB Command window. Open the following demos and see them.

Desktop Tools and Development Environment

1) ***Tutorials about the environment***

1.1 Follow the *Getting Started with MATLAB* video demo. Pay particular attention to the use of vectors and matrices.

1.2 Follow the *Working in the development environment* video demo.

1.3 Indicate what the *command history* is and list two ways to repeat the evaluation of a previously executed expression.

1.4 Indicate what the *current folder* window is used for. How can be some visualization options changed?.

Matrices

2) ***Basic matrix operations, Matrix manipulation***

(Follow the tutorial and then answer the questions typing the corresponding expressions in the command window.)

2.1 Write a row vector, called "v1" with the following five numbers: 12, 23, 54, 8, 6.

2.2 Add 10 to each element of "v1" and store it at "v2".

2.3 Visualize a two-dimension graphic of "v2".

2.4 Create the following matrix of 4x4 elements and call it M:

1	4	22	7
9	2	3	11
49	55	6	3
24	7	9	12

2.5 Obtain the transposed matrix of M and call it Mt.

2.6 Obtain the inverse of M and call it Mi.

2.7 Proof that the product of a matrix by its inverse is the identity matrix.

2.8 Indicate the contents of the "ans" variable.

2.9 Visualize a 3D graphic of the M matrix in wire as well as a surface (type: >>mesh(M), >> surf(M) – ">>" is the prompt symbol–).

(If you need more help about matrix manipulation you can see the *Working with Arrays* video demo and read the *Matrices and Arrays* help in the *Getting Started* entrance in MATLAB Help)

Graphics

(For more information, see the *Using Basic Plotting Functions* video demo and read the *Graphics* help in the *Getting Started* entrance in MATLAB Help).

3) ***2D Plots***

3.1 ¿What does the "*x = 0:0.05:5;*" expression?

3.2 ¿What does the "*bar*" function?

3.3 ¿What does the "*stairs*" function?

4) ***3D Plots***

4.1 Indicate and proof what does the "*z = peaks(25)*" expression.

4.2 Indicate the difference between *mesh* and *waterfall.*

4.3 Indicate the difference between *surf* and *surfl.*

4.4 Observe the effects of *contour, quiver* and *slice.*

5) ***Images and Matrices***

5.1 Observe the different colormaps that can be applied to indexed images. Indicate three of them and remark the kind of the obtained color when used.

6) ***Line plotting***

6.1 Explain what does each one of the demo expressions:

```
x=0:0.02:1;
hndl=plot(x,humps(x));
set(hndl,'Color','cyan')
```

6.2 What is stored in *hndl*?

6.3 What sentence must be written to change to green the color line?

6.4 What sentence must be written to change the line width to 2?

6.5 What sentence must be written to change the line symbol to "o"?

6.6 What sentence must be written to change the line symbol size to 12?

7) ***3D Surface plots***

7.1 Open a new figure (type *figure*), evaluate one by one the following demo lines and indicate what are they used for.

```
z=peaks(25);
 surfl(z)
 shading interp
 colormap(hot)
 axis off
```

Programming

8) ***Matlab -> Getting Started -> Programming***

8.1 Follow the *Writing a MATLAB Program* video demo.

8.2 Follow the *Introducing MATLAB Fundamental Classes* video demo.

8.3 Create an M-file with the following code and check how it works:

```
function [area] = calculate_rectangle_area (side1, side2)
area=side1*side2;
```

9) ***Getting Started -> Programming -> Vectorization***

One way to make your MATLAB programs run faster is to vectorize the algorithms you use in constructing the programs.

9.1 A simple example involves creating a table of logarithms:

```
x = .01;
for k = 1:1001
    y(k) = log10(x);
    x = x + .01;
end
```

9.2 Test results and execution time with a vectorized version (use functions *tic* and *toc*)

```
x = .01:.01:10;
y = log10(x);
```

10) ***Getting Started -> Programing -> Preallocation***

If you cannot vectorize a piece of code, you can make your for loops go faster by preallocating any vectors or arrays in which output results are stored.

10.1 Run the next code :

```
for n = 1:32
    r(n) = rank(magic(n));
end
```

10.2 See the run time benefits of inserting the next sentence before the loop. Test with higher n values.

```
r = zeros(32,1);
```

11) ***Getting Started -> Programming -> Other Data Structures -> Multi-dimensional arrays***

11.1 Read what a multidimensional array is.

11.2 Check and indicate what do the next expressions:

```
A=[5 7 8; 0 1 9; 4 3 6]
A(:,:,2)=[1 0 4;3 5 6; 9 8 7]
```

11.3 Next, check the following expressions, write the returned value and indicate what are the differences among them:

```
A(1,2)
A(1,2,1)
A(1,2,2)
```

11.4 Next, how do you access to the element in the third row, second column in the second dimension?