Numerical Methods I
Guideline for computer sessions
Mathematics Degree

Virginia Muto Foresi
Departamento de Matemática Applicada y
Estadística e Investigación Operativa
Facultad de Ciencia y Tecnología
Universidad del País Vasco/
Euskal Herriko Unibertsitatea

virginia.muto@ehu.eus
Office E.P0.20

Academic Year 2019/20

Contents

Computer Practise 1

Introductory Practise to MATLAB: basic concepts

1. Introduction.

   The scientific programming language MATLAB is used very frequently in the courses of Numerical Analysis.

   These notes give an introduction to basic notions of MATLAB with a tutorial and few examples, with special attention to how we built and treat matrices and how we do create graphics of functions representation.

   MATLAB (MAThematical LABoratory) is a language for scientific computing whose fundamental data are vectors and matrices. It is different from other programming languages as Fortran or C, since it works to higher level including hundreds of operation in just one command.

   Since the first versions of Cleve Moler an the end of 70's, MATLAB has became a powerful tool used in order to make research and to solve practical problems.

   In the page http://www.mathworks.es/ one can see the enormous quantity of information around MATLAB: published books, programs, links, news, scientific meetings...

   MATLAB is an interactive system. When we start it, we get in the screen a window divided into five zones:

   two of them on the left (up the ''Current Folder'' and ''Details'', down), other two in the central area y the last one on the right (''Workspace'', where the declaration of variables will appear).

   In the central area, if we have opened a ''script'', it will appear in the top zone and down the area for ''Commands''. Sometimes the script will be opened in a new window called ''Editor''.

   When one writes the name of a command after the prompt >>, and hits the ''Enter'' key, the system will run the instruction and if we have not assigned a particular

name, it will keep the result in a new variable called ‘‘ans’’.

The variables will be stored in the workspace that lies in the right side.

MATLAB has three important characteristics that make it different from other programming languages:

- One does not need to declare the variables.

- It contains a big collection of mathematical functions.

- The standard and fundamental type for data is vector (matrix) with complex numbers stored in floating point arithmetic with double precision.

Before starting, I want just to point out that the ‘‘help’’ command is useful to get information that is complete and clear enough with many examples.

- If you type ‘‘help command’’, after the prompt >> in the Command window, you will get the syntax and the options for the command ‘‘command’’.

- You can get even more information if you type ‘‘command’’ in the right up corner in the space denoted by ‘‘Search Documentation’’.

- Typing ‘‘lookfor subject’’ will search for everything related with ‘‘command’’.

Let us list same important characteristics:

- Capital letter and lower case are not equivalent.

- A semicolon ‘‘;’’ at the end of an instruction will suppress the output on the screen.

- The parenthesis () and square brackets [] are not interchangeable.

- The up arrow ↑ recovers previously used commands.

- In order to exit from MATLAB you have to type ‘‘exit’’ o ‘‘quit’’.

The following commands are very useful:

bench: makes a speed test of your computer.

computer: tells you the kind of PC you are running MATLAB.

demo: gives you access to a collection of examples.

pwd: actual directory.

dir: list the programs in the actual directory.

clear: removes the variables of the workspace (variables in the memory in the space we are working in).

Nan: indicates there is some indeterminate operation.

path: indicates the directories which are accessible in this moment.

ver: indicates the used MATLAB version.

who: gives an alphabetic list of all variables used in the workspace.

whos: gives an alphabetic list of all variables used in the workspace, with additional information about the type and the size.

2. Tutorial.

   During this first practise, we will use some MATLAB commands and we will see many examples.

   Interesting the page `https://es.mathworks.com/support/learn-with-matlab-tutorials.html`.

   The constants, variables y operators are written in a natural and intuitive form in the Command Windows.

   The operation/command is executed touching the Return or Enter key. Note that MATLAB makes difference between capital letter and lower case and they are considered as different characters.

   Again, typing ''help'' followed by a word or sentence, we will get useful information about the operations we want to realize. The same if we type the command in the upper right part of ''Search Documentation''.

   Sum/subtraction y product/division con '' + −'' and '' ∗ /''. If one tries to use empty space for multiplication, as in Mathematica or other languages, MATLAB gives error.

```
2+4
3*5
3  5
5-2
6/2
```

   The assignment instructions of numerical values to variables are done using ''=''. If we end the instruction by a semicolon (;) the output is not shown.

   % indicates a comment line.

```
a=8;
b=5;
a/b    %  This is division
a^b    %  This is power
```

   By default, MATLAB shows numbers with 4 decimal digits, namely, it uses ''format short''. If we want to see more digits, we can use ''format long'', that shows up to 14 decimals, or we can use the options for the ''fprintf'' command as we will see in next practises. The commands ''format short g'' and ''format long g'' choose the best options among ''short'' or ''short e'' and ''long'' or ''long e''.

   In order to erase the value assigned to a variable, one can use ''clear variable-name''. The command ''clear'' by itself erases all the defined variables and defined functions.

```
format long e     % in scientific notation
a/b
format short
a/b
clear a
a+b
clear
```

"*pi*" is a predefined constant in the system. Moreover, there exist predefined functions as "exp", "sin", "cos", ...

```
pi
exp(1)
sin(pi)
cos(pi)
```

In the following we are going to show different examples of hoy we can define manually a row vector, a column vector and a bidimensional matrix. Moreover, we compute in two different manners the scalar product.

In order to know the number of elements (row or column) of a vector we use the command "*length*".

```
rowvec = [1 2 3]
colvec = [4;5;6]
length(rowvec)
length(colvec)
mat=[1 2 3; 4 5 6]
rowvec * colvec
dot(rowvec ,colvec)
```

Now we compute the product of the column vector times the row one in order to obtain a matrix.

The command "$size(A)$" gives the matrix's dimension.

The command "$ndims(A)$" gives how many dimensions have the matrix: in this case 2 (row and columns).

The command "$openvar('A')$" opens the variable in order to inspect its elements.

```
A = colvec * rowvec
size(A)
ndims(A)
openvar('A')
```

We may need to perform multiplication (or division) by or sum (or resting) a scalar to all the elements of a vector; this is done using standard mathematical operators $(*/ + -)$.

However, when we need to power all the elements of a vector to a fixed power, the operator ‘‘^’’ is not valid, since the program will understand that is the product of the row (or column) vector by itself, that is not possible in matrix algebra. Let us see with the following examples.

Let us compute the square, the exponential and the square root of each element of the row vector. In the mean time, we play with the ‘‘format’’ command.

```
b=5*rowvec
b = rowvec.^2
exp(rowvec)
sqrt(rowvec)
format long
sqrt(rowvec)
format
```

Now we want to compute the sum of the elements of the row vector and the column one.

```
sum(rowvec), sum(colvec)
sum(A)
```

Some specific operators for matrices are ‘‘inv’’, used in order to obtain the inverse matrix, ‘‘’’ to get the transpose and ‘‘det’’ for the determinant.

The operator that is new for many students is the division to the right ‘‘\’’, that implies that the dividend is placed to the right of the divisor, at the contrary of the conventional division.

```
aa=15/5
aa=5\15
```

In the next example, the ‘‘.\’’ MATLAB operator divides 2 by each element of colvec. And in the other example, it divides each element of colvec by 2.

```
colvec.\2
2.\colvec
```

The most important utility of this operator is the solution of systems of linear equations. Let us define a new matrix B and let us solve the linear system ‘‘$Bx = colvec$’’.

```
B=[-3 0 1; 2 5 -7;-1 4 8]
x = B\colvec
x = mldivide(B,colvec)
norm(B*x - colvec)     % norma euclidiana
```

Another example could be:

```
A=[3 2 -1; 8 -3 4; 5 2 1];
B=[10;21;13];
x = inv(A)*B
x=A\B
norm(A*x - B)
```

Let us compute now the eigenvalues and the eigenvectors of a matrix

```
e = eig(B)
[V,D] = eig(B)
```

Another way to define vectors is to specify the first term (a), the difference between two terms (d) and the last term (z), f.i. "$x = [a : d : z]$".

It is also possible to build a vector specifying the first and the last element (a and b), as well as the total number of elements, (n). In this case the distance (difference between two terms) will depend on the total number of terms we wish to built. In order to do this we will use the command "$linspace(a, b, n)$". If the number n is not specified, it will be considered as 100 for default.

```
x = [0:0.1:1]
x = linspace(0,1,11)
```

Another way of building vectors and matrices:

```
v=1:6
w=2:3:10, y=1:-.25:0
C=[A,[8;9;10]]
D=[B;rowvec]
```

In order to obtain the i-th element of a vector we write "$v(i)$".

If we are working with a matrix we will have to specify the row and the column.

If we want to specify a range of elements of a vector or a matrix we will use "colon".

The command "$C(:, n)$" refers to all elements of the "n-th" column of matrix C. Similarly, the command "$C(m, :)$" refers to all elements of the "m-th" row of C.

```
C(2,3), C(2:3,1:2)
C(:, 3), C(2,:)
```

Other matrices

eye: identity matrix.

zeros: matrix whose elements are all zero.

ones: matrix whose elements are all one.

rand: matrix whose elements are random numbers with standard uniform distribution in (0,1).

randn: matrix whose elements are random numbers with standard normal distribution.

```
I3=eye(3), Y=zeros(3,5), Z=ones(2)
F=rand(3), G=randn(1,5)
```

Let us look at our workspace

```
who
whos
```

Example of sum of continuous fractions.

It is interesting to see the possibility of performing a loop:

```
g=2; for k=1:1,g=1+1/g; end
g
%
g=2; for k=1:2,g=1+1/g; end
g
%
g=2; for k=1:10,g=1+1/g; end
g
```

Graphics of curves in the plane.

We can generate bidimensional plots using the command "plot", whose syntax is

```
plot(x,y,'LineSpec','properties','values').
```

"x" and "y" are two vectors with the same number of elements; the first one contains the x abscissa values and the second the y ordinate values. The "LineSpec" are the arguments that define the type and the appearance of the line and/or markers that will be used for the graphic representation. The "properties" and "values" allow to modify the appearance of the plotted curve or used markers.

In the next computer practise we will see plot in space (3-D).

```
t=0:.005:1;
z=exp(10*t.*(t-1));
plot(t,z)
%
plot(t,z,'g--')
xlabel('t')
ylabel('sin(12 pi t)')
title('Plot of the Sine Function')
%
t=0:.005:1;
z=exp(10*t.*(t-1)).*sin(12*pi*t);
plot(t,z,'g:*')
```

Graphics of two curves in the same plot.

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
%
hold on
%
y2 = cos(x);
plot(x,y2,':')
legend('sin','cos')
```

3. References

1. V. Dominguez and M.L. Rapún, available in *MATLAB en cinco lecciones de Numérico*
2. A. Gilat, MATLAB. Una introducción con ejemplos prácticos, Reverté, 2006.
3. D.J. Higham and N.J. Higham, MATLAB Guide, SIAM, 2000.
4. J.F. Mathews and K.D. Fink, Métodos Numéricos con MATLAB.Tercera edición, Prentice Hall, 1999.
5. C.B. Moler, Numerical Computing with MATLAB, SIAM, 2004.
6. C.F. Van Loan, Introduction to Scientific Computing, Prentice-Hall, 2000.
7. W.Y. Yang, T.S. Chung W. Cao, and J. Morris, Applied Numerical Methods Using MATLAB, Wiley Interscience, 2005.