Computational Methods and Data Analysis 2003

Ex. 1: Use of MATLAB

This exercise is meant to learn to use matlab. It can also be found in my dyrectory
~fasolino/CMDA/exer_2003
and in the home page of this course
http://www.snn.kun.nl/~tom/cmda2003.html.

It is better if each of you works alone. Only typing yourself you get to know the use of a computer. Of course you can exchange questions, comments and successful solutions. Actually those who find this exercise very easy are encouraged to help the others.

An interactive matlab course can be found at:
`http://www.mines.utah.edu/gg_computer_seminar/matlab/matlab.html?` ,
a rather complete manual can be downloaded as pdf file at:
`http://csp.tn.tudelft.nl/matlab/matlablong.pdf`.
If you open the latter as pdf file on your screen, you can easily search for a keyword, for instance the name of a function. The manual is in colour but a print out in gray-scale is just as good and much cheaper. Use the option -Zg in the lpr unix command.

It is useful to have all the programs of the course in one directory which is open for reading. To open a directory for reading use the unix command
`mkdir CMDA` (this creates the directory CMDA)
`chmod 755 CMDA` (this gives reading access to everybody)

Within this directory you can create other subdirectories for each exercise. Each directory has to be opened for reading.

To give reading access to a file myfile.m

```
chmod 644 myfile.m
```

To start matlab type

```
matlab -nojvm
```

the option -nojvm avoids the graphical (java) window which slows down the functioning. To close matlab you will type `quit` or `exit`.

When matlab starts type

```
helpwin
```

This opens an extra window with help information.

Working from keyboard

Assign:

- scalars: year=2003 ; conversion_factor=2.2 ; hbar=1.054e-34
- row vectors: v_1= (1 5 2 1 0); v_2= (7 -1 1 0 1)
- a 3×3 matrix with all elements equal 1 or all elements equal zero. Which is the easiest way? Write the 3×3 unit matrix by use of a loop.
  For the 2×2 matrix
  `A=[1 2;3 4]`
  calculate
  `A*A, A.*A , A^2, A.^2, A/A , A./A`.

  Check the meaning of each operation by calculating analytically the result.
- calculate the scalar product $v_1 \cdot v_2$ by use of a loop or by using the $.*$ operator of matlab. Check that the result is right.
- find the minimum and maximum component of vector v_1. This can be done by using the built-in matlab functions `min` and `max`. Use the help to find out how to use them, namely type `help min`.

Here it is suggested to look at similar functions, for instance have a look at `range`. In this way you come to learn other features of matlab.

Writing a script

Write a script (an m-file, you could call it basics.m) which performs the previous operations. When it is ready you just need to type the name of the file (without .m) to execute it. Make use of the function `input` to read the vector components from the keyboard. To ask for input you can use `input` with a string of characters. Put some comments in your script describing what you are doing. By typing `help basics` the comments which are at the top of the script (before the first command) appear on the screen. It is good practice to write comments and to name variables in English since international cooperation is likely to occur in your future. It is also better to start such a script by clearing the memory and all files and figures with

```
clear all
close all
```

Reading and writing from a file

To give all components of a vector by hand is boring and can lead to errors. It is often better to have the input on a file which is then read by matlab. For instance one often uses matlab to plot experimental or calculated data which is stored in a file.

Let's first create a file which contains the vectors
a=(1 2 3 ... 10) and the vector formed by taking the square of each component b=(1 4 9 .... 100). Instead of giving the components, find matlab commands which build these vectors.
The simplest way to write these vectors in a file is by use of the command `save`, namely
`save filename a b -ascii`. If you want to write with a specified format you can use `fopen` to create a file called, for instance, data, and write with `fprintf` the components, namely:

```
fid=fopen('data','w');
for i=1:10
 fprintf(fid,'%5i %5i\n',a(i),b(i));
end
fclose(fid);
```

which writes the vectors in the file data as follows

```
  1     1
  2     4
  3     9
  ..    ..
 10   100
```

Now create another script which reads the two vectors which you will call `x` and `y` from the file data. This can be done by use of `load` in several ways. For instance

```
load 'data'
[nrow, ncol] = size(data)
x=data(:,1)
y=data(:,2)
```

Check that the vectors have been read correctly. Check the size, with `size(x)`, the first component x(1), the last x(end). Plot them by using: `plot(x,y)`, `plot(x,y,'*')`, `plot(x,y,'g')`, `plot(x,y,'-',x,y,'+')` , `semilogx(x,y)`, `semilogy(x,y)`, `loglog(x,y)`. Plot also z=x+y in the same graph, by using `hold on`.
Try and label the axis, by learning to use `xlabel, ylabel, axis, legend.`