MATLAB

Assignment 1

This assignment is largely intended to introduce you to the computing system and Matlab basics. The Matlab Introduction slides as well as other references can be found on course website `https://mmedvin.math.ncsu.edu/Teaching/Matlab.html`.

1. Invoke MATLAB. On the left side of the window you should see the MATLAB prompt, $\gg$, which means that you are inside the MALTAB shell. Now you are ready to use MATLAB.

2. At the MATLAB prompt enter

   diary hw1.txt

   (Every command should be followed by hitting 'return', of course.) The command **diary** will record your MATLAB session in the text file hw1.txt (or whatever filename you choose). Now enter

   help diary

   to receive on-line information from MATLAB about **diary**. Note: throughout this exercise use the **help** command to get on-line information about any MATLAB command.

3. This exercise will introduce you to the utility of so called m-files. You will be using m-files a lot in this course. MATLAB is capable of executing sequences of commands that are stored in m-files. To see this, create a file called, for example, **test.m**, in your current directory. This can be done by entering

   edit mytest.m

   at the MATLAB prompt (mytest.m is supposed to be new, i.e. empty file change the name if you need). Then write the following commands in **mytest.m** (the words after the % symbol recognized as a comment in matlab - comments used to improve readability of the code):

   a=[1,2,3]; % create a row vector
   b=[1,2,3]'; % create a column vector
   A=b*a; % an outer product of column vector with row vector creates a matrix
   I=eye(3); % create identity matrix
   B=I+A; % add two matrices
   x=B\b; % solve equation $Bx = b$

   Save and quit **mytest.m**. To run **mytest.m** in the MATLAB shell, simply enter

   mytest

   (without the **.m**). It returns the value of **x**. Note: the inclusion of semicolon ";" after the statements suppresses the printing of the results after the statements are executed. To see what this means, go back to your **mytest.m** and delete a semicolon at the end of any statement, run the program again and see what happens.

4. Use MATLAB to create plots of the functions $\cos(1.7x)$ and $\sin(1.7x)$ for $x \in [0, 2\pi]$. Enter:

   x=0:0.1:2*pi;
   y1=cos(1.7*x);
   y2=sin(1.7*x);

   The first command creates the vector **x** with the values $0, 0.1, 0.2, ...$ up to 2π. The second and third commands create the vectors **y1, y2** of the same length as **x**, such that $y1(i) = \cos(1.7x(i))$ and $y2(i) = \sin(1.7x(i))$. Now use subplot and plot to prepare three graphs, *all in the same graphical window*:

   (a) A plot of the graph $(x, \cos(1.7x))$ in **subplot(2,2,1)**.

(b) A plot of the graph $(x, \sin(1.7x))$ in **subplot(2,2,2)**.

(c) A plot of $(x, \cos(1.7x))$ and $(x, \sin(1.7x))$ in **subplot(2,2,3)**. Use **legend** to show which is which.

The **plot** command creates a smooth graph that passes through all the points with coordinates $(x(i), y(i))$. Hence, this graph is only a discrete **approximation** of the accurate (i.e., continuous) graph. The finer spacing we use (say, 0.01 instead of 0.1 in the definition of **x**) the smoother (and more accurate) the graph we get.

Save your output as an JPG file. For more information use **help print**. Create a hardcopy of your output by printing the file to a printer. If you experience technical problems (e.g., the printer ran out of paper) please refer to the system staff.

Note: one thing that is emphasized in this class is a clear presentation of numerical and graphical results. Therefore:

- All plots should be created using **subplot** (remember the rain forests!).
- To make your plots clear, always use the commands **xlabel, ylabel** and **title**.
- If you have more then one graph in the same plot use **legend**. You can also add text anywhere inside your plots using **text** ot **gtext**.

5. In this exercise we compare the run time for various operations using the Matlab's pair of commands **tic - toc** (try **help tic** for documentation). Start by running the following program:

```
Num = 1000;
A = rand(Num,1);
B = rand(Num,1);
tic;
for i = 1:Num
     C(i)= A(i)+B(i);
end
T_add = toc
```

(a) Run the program a few times. Do you always get the same results? Why?

(b) Modify the program to compare the run time of +, -, *, /, exp(x), sin(x) and gamma(x) (do **help gamma** for a definition of this function). As the results may vary, run the program 10 times and give the average result.

(c) Summarize the results qualitatively.

(d) We can do much better in MATLAB by avoiding loops and using *vectorization*. Run the following program:

```
Num = 1000;
A = rand(Num,1);
B = rand(Num,1);
tic
C=A+B;      % for *,/ use A.*B,A./B (. means operating element by element)
T_add_vec = toc
```

Compare the run time of +, -, *, /, exp(x), sin(x) and gamma(x) in the (loopless) vectorial case, by showing the vectorization speedup factor for each operation.

(e) Summarize the results.

(f) Run the program

```
tic;
for i = 1:Num
   A(i); B(i);
end
T_loop = toc
```

How does the empty loop run-time compares with other run-times?

6. Enter

   diary off

   to terminate the record of your commands. Then enter

   exit

   to quit Matlab.

Hand in: the diary, the m-files and plots, the answers to questions in question (5).
Note: when handing in the diary, clean it from typos and make it readable. Use a marker to highlight the exercise number, final solution, and so on.

THE MATLAB DIGEST

Here are some additional functions in MATLAB and some useful tips:

- **max**(**x**), **min**(**x**) find the maximal and minimal values of a vector x. See help on them for more details. They are useful in many situations, e.g., calculating the maximal value of an error vector.
- **prod**(**x**) calculates the product (i.e., multiplication) of all the elements of the vector **x**.
- **sum**(**x**) adds the elements of the vector **x**.
- The command **format long** tells MATLAB to print 15 digits on the screen. The default is **format short**, which prints only the 5 significant digits. This command can be convenient when inspecting an error vector. For additional format options see **help format**.