Introduction to Numerical Analysis I

Assignment 1

This assignment is largely intended to introduce you to the computing system and Matlab. The Matlab Introduction slides as well as other references can be found on CANVAS. The Matlab suite is available on the Math Department's Computer lab [1]

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 test.m

   at the MATLAB prompt. Then write the following commands in **test.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 **test.m**. To run **test.m** in the MATLAB shell, simply enter

   test

   (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 **test.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*:

[1] All registered students to this class have an account in The Math Department's Computer lab, which is located between LCB and JWB in the T. Benny Rushing Mathematics Center, room 155C. Matlab may also be available on other campus computer labs.

(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. Enter

   diary off

   to terminate the record of your commands. Then enter

   exit

   to quit Matlab.

6. Calculate absolute and relative errors in the following approximations for x by x_n:

   (a) $x = \pi, \quad x_n = 22/7$
   (b) $x = e, \quad x_n = 2.718$
   (c) $x = e/100, \quad x_n = 0.02718$
   (d) $x = 10^{\pi}, \quad x_n = 1400$

7. (a) Solve the quadratic equation $x^2 + 111.11x + 1.2121 = 0$ using the formula

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

   with 5 digit precision. Note the loss of significant digits in x_1. Do this twice: First, *round* the result after each calculation step. (that is, 123.456 becomes 123.46.) Then repeat the calculations but *chop* the result after each step. (that is, simply ignore extra digits, so that 123.456 becomes 123.45.) Note that usually, rounding gives more accurate results than chopping.

   (b) The loss of significant digits in x_1 in (a) can be avoided by using an alternative formula for x_1:

   $$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

   Use this formula to calculate x_1 more accurately. *(this formula can be obtained from the original one by multiplying the enumerator and the denominator by: $-b - \sqrt{b^2 - 4ac}$.)*

8. Let $S_x(N) = \sum_{n=0}^{N} \frac{x^n}{n!}$. Then $\lim_{N\to\infty} S_x(N) = e^x$ for any $x \in \mathbb{R}$.

   (a) Using Matlab, calculate $S_{x=10}(N)$ for N=[10:10:100]. You obtain a vector of numbers, $\vec{S}$. The elements $S(i)$ of $\vec{S}$ are partial sums that approximate the function e^x with different accuracies.

   (b) Using Matlab's built-in function **exp(x)** you can calculate e^x "exactly" (i.e., with double precision accuracy). Using **exp(x)** plot the graph of relative errors

   $$R_x(N) = \left| \frac{e^x - S_x(N)}{e^x} \right|$$

   for $x = 10$ as a function of N.

   (c) Repeat (a) and (b) with $x = -10$.

   (d) Do the graphs show convergence for both values of x?

   (e) Do the elements of the $\vec{S}$ converge to the correct value in both cases?

   (f) Explain your answers to (d) and (e).

9. A floating point number can be represented by $Fl(x) = \pm(1.d_1d_2...d_t)_2 2^e$, where the sequence $(.d_1d_2...d_t)_2$ is called *mantissa*, t is the number of digits in the mantissa, 2 is the *base* (or *radix*) and e is the *exponent.*

   (a) For a DEC-VAX using **single precision** we have $t = 24$ and $-128 \le e \le 127$. What are the smallest and largest numbers, m and M, that can be stored in this computer?

   (b) Repeat (a) for the case of **double precision**, where $t = 52$ and $-1024 \le e \le 1023$.

   (c) What is the smallest value of x in (a) and (b) for which x^{100} will overflow (i.e., $x^{100} > M$)?

   (d) The following calculations are done using single precision:

   $$\text{(i) } 10^2 - \sqrt{10^4 - 1}, \qquad \text{(ii) } 10^4 - \sqrt{10^8 - 1}, \qquad \text{(iii) } 10^8 - \sqrt{10^{16} - 1}.$$

   Determine whether each of the results of (i)–(iii) will be zero, nonzero or overflow.

10. Using Matlab, find the *smallest* n such that

   $$\underbrace{\frac{1}{3} \times \frac{1}{3} \times ... \times \frac{1}{3}}_{n} \times \underbrace{3 \times 3 \times ... \times 3}_{n} \neq 1 \ .$$

Hand in: the diary, the m-files and the answers to questions (6)–(10) (including plots). **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**. This function can be useful for (10).
- **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**.