Math 104A, Fall 2011

Introduction to Numerical Analysis
Guidelines for Programming Assignments

Instructor: Jingrun Chen.

Office: South Hall, 6705.
E-mail: `cjr@math.ucsb.edu`
URL: `http://www.math.ucsb.edu/~cjr`

As we advance in the course, the assignments become more computationally oriented. Part of the objectives of this course will be to develop adequate programming style and techniques. As important as these will be the presentation of numerical results in a manner that is clear and effective.

Programming

Your programs should be written with the reader in mind. This *reader* might be your professor, your boss, or it might even be you, two months after you wrote it. For this reason, your program should be well documented, and should be written with a good programming style. The general principle is that whoever reads your code should not have to go through all the computer instructions in order to know what the program does. This comes from practice, but there are several general principles.

The variables you use should have names that hint at what the values represent. For example, use names as `length`, `height`, and `VolumeOfSphere`, as opposed to `y`, `yy`, and `yyy`.

At the beginning of the program, you should briefly explain what the program does, and how. You should also describe the variables and data structures that you use: which ones are input, which ones are output, which ones are temporary, what kind of data they represent (real, double, integer, arrays, pointers, etc). If a variable is changed inside a subroutine, this should be made clear too. The dimensions of arrays should be specified also.

Each instruction (or group of instructions) in your program should be preceded by a brief description of what it does. Comments like `Initialization`, `Creation of coefficient matrix`, `Runge-Kutta iteration begins here`, or `Output to file`, can be very helpful when reading the program, and even when debugging your code.

Output

When talking about the output of a program, one must distinguish between what is the input data, and what constitute results. Each numerical simulation is unique, in the sense that the numerical results will change only if you use different parameters. Your results will depend only on a few parameters, such as the grid size and the time step size, and some physical parameters. The results can be completely reproduced by running your code using the same parameters. You should always print out the parameters that you are using for your simulation, since after a while you will have

completely forgotten what values you used. You should also print out the results in a form that is readable. For example, imagine the following computation: A ball is dropped from a height h, and one wants to know the speed of the ball when it hits the ground. The physical parameters would be the height h, the mass m of the ball, the acceleration of gravity g, the initial velocity v_0, and the friction coefficient of air, for example. The numerical parameter would be the time step size used in the numerical solution of Newton's Second Law. When solving this simple example, one might want to plot the whole trajectory of the ball. This trajectory could be plotted, and the parameters of the simulation could be printed separately.

The output of your programs should be very clear and concise. In each line printed, it should be specified what it is that you are printing, whether it is the total number of iterations, the final error, the number of points used, or the value of the different physical and numerical parameters employed. A good idea is to produce a log file in which you only save the parameters that describe your numerical simulation, as opposed to the output files, in which one can store all the data that will be plotted. In the log file, one can also write the **final** results, such as the error (in some appropriate norm), and things like that.

Plotting

When presenting numerical results, one should always stop for a minute and think what it is that one wants to convey to the reader, and what is the best way to do so: Printing an array of 320^2 (= 102400) numbers is clearly unacceptable, for a large number of reasons. Plotting is a clear alternative, and is an efficient way to present results, if done properly. Think what it is that you want to illustrate, and what plots would be effective in doing so. Then think what is a nice way to plot the results: the reader should not have to guess what it is that you are plotting.

As an example, assume that you are trying a new numerical method, and you want to show the order of convergence. Assume further that the error behaves like $O(N^{-2})$, where N is the number of data used in your computation. Printing the N values of the error is a very bad idea, and it is even worse to print the N values of the actual result, next to the N values computed. Doing so in a table like the following is an effective way of presenting the result:

Results of Newton's Method			
n	$x_{n+1} = g_4(x_n)$	$\lvert x_n - x^* \rvert$	$\frac{\log e_{n+1}}{\log e_n}$
0	10.0000000000	701.0000000000	
1	7.0791666667	205.4258086661	0.7310427008
2	5.1747623839	59.2361475383	0.5790093064
3	3.9728746222	16.3555930500	0.1075630702
4	3.2773048578	3.9784558101	-10.3106810980
5	2.9605072611	0.6538618129	2.7257585425
6	2.8838597713	0.0341026943	2.1535760044
7	2.8794000273	0.0001123173	2.0560403401
8	2.8793852417	0.0000000012	2.0268047419
9	2.8793852416	0.0000000000	Infinity

Figures (1) and (2) show the results of a numerical simulation in population dynamics. I have plotted the same results in both figures. Clearly the first plot is not acceptable, since one cannot

tell what is being plotted.

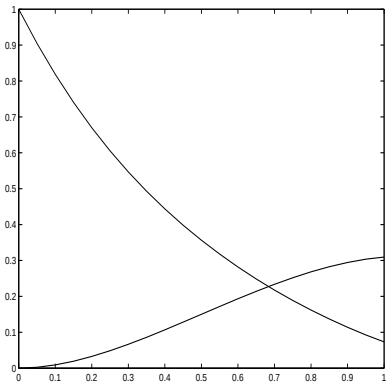

Figure 1: Example of what NOT to do.

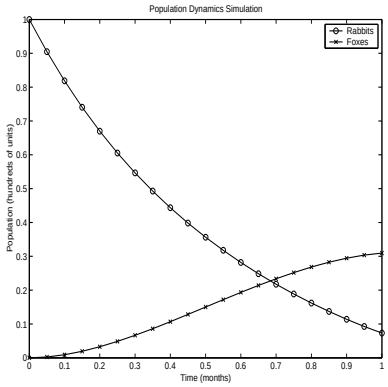

Figure 2: Example of what to do.

There are other things you should have in mind. For example, your plots should not be too crowded, and each curve in the plot should be distinguishable from the rest. Remember also that the plot might look different once you print it.

You should not use too many plots either. It is better to use a few plots that illustrate clearly a few ideas, than to show a whole bunch of them, with not a clear line of thought. Plots should simply be a visual help to understand the concepts you are introducing.

Some useful commands for plotting in Matlab are: `plot`, `subplot`, `surf`, `mesh`, `title`, `xlabel`, `ylabel`, and `legend`. Just do `help graph2d`, `help graph3d`, `help graphics`, `help specgraph`, or simply `help` for a list of commands. Note that Matlab accepts also the command `fscanf`, much like in C language. Try `help iofun` for a list of functions available for file input and output.

Note: Plots done by hand will not count. You must plot the data obtained in your numerical computation.