Math. 512 Project 1 (Due Friday January 26)

Main Topics:

A brief introduction to numerical computations.

Main Objectives:

Gain familiarity with your software of choice, recognize the importance, presence and possible damaging effects of the inevitable numerical errors in scientific computations.

Main Tools:

Basic computer knowledge and basic numerical analysis.

1. (optional)

a) Install Matlab in your personal computer (most likely you already have it in your computer). It is available for free for USC students from the website:

https://software.usc.edu/matlab/

b) Open MATLAB and click on the learn MATLAB icon. Complete the Onramp course and print or save the completion certificate.

After completing the Onramp course, you should be familiar with the basics of:
Commands, Vectors and matrices, importing data, indexing and modifying arrays, array calculations, Calling functions, obtaining help, plotting data, Scripts, logical arrays, programming.

c) Vectorization

Generate the n-vector $y = \sin(x^2)$ at equidistant points in the interval $[0, \pi]$ in three ways:

1) sequential, using a 'for–loop'.
2) the same as in a) preceded by the initialization of y, i.e. setting $y = zeros(1, k)$ for appropriate k
3) vectorized, using the commands (familiarize yourself with Matlab 'dot' operations):

$x = linspace(0, \pi, k);\ \ y = \sin(x.* x);$

Verify experimentally using Matlab how the cost, c, of these operations depends on n, the size of the problem (suggestion: use 'tic' and 'toc' to measure the cpu time). Plot the cost c vs n for the three methods on the same plot. Label the axes, title the plot, and identify each set of data points using different lines for each plot. What are your observations from the results of this experiment?

2. (optional)

Open MATLAB and in the help icon explore documentation and examples. Get a brief look at some of the commands and examples of numerical applications. Note: there is an extensive set of examples and resources available at the Mathworks website. Take a look at some of them and discover the enormous applications of numerical algorithms in the industry and academia.

3. Let $f(x) = \cos x\ and\ x_0 = 1.2$. Write a routine to compute the absolute error of approximating $f'(x_0)$ by $\frac{f(x_0+h)-f(x_0)}{h}$. Experiment with values of h that approach zero. Plot your result on a log-log scale. Comment on your results.

4.

Consider the linear system

$$\begin{pmatrix} a & b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\ \ with\ a, b > 0; a \neq b.$$

a) If $a \approx b$, is there a numerical difficulty in solving this linear system.
b) Suggest a numerically stable formula for computing $z = x + y$ given a and b.
c) Determine whether the following statement is true or false, and explain why:
"When $a \approx b$, the problem of solving the linear system is ill conditioned, but the problem of computing $x + y$ is not ill conditioned."

5.
Write a routine (call it) bisect to find the root of the function $f(x) = \sqrt{x} - 2.1$ starting from the interval $[2,10]$, with $atol = 1.e - 8$.

a) How many iterations are required? Does the iteration count match the expectations, based on our convergence analysis?
b) What is the resulting absolute error? Could this absolute error be predicted by our convergence analysis?

6.
Write routines for problem 5 that implement Newton's and fixed point method, and compare results and stablish rate of convergence of each of the 3 methods (bisection, fixed point, Newton). Plot the function and show in this plot the progress of each of the methods.

7.
Write a routine (call it tri) that solves tridiagonal systems of equations of size n. Assume that no pivoting is needed, but do not assume that the tridiagonal matrix A is symmetric. Your program should expect as input four vectors of size n or $(n - 1)$: one right-hand-side b and the three nonzero diagonals of A. It should calculate and return $x = A^{-1}b$ using a Gaussian elimination method.