Problem Set 1

Problem 1. *Complexity of common algorithms*

In series Calculus, we used the known convergence and divergence of some common, simple series as a comparison point for us to determine the convergence of other series. Analogously, it will be of use to us to know how many operations some common algorithms and operations take to perform.

(a) Find the number of flops and complexity (big-'0' notation) needed to compute:

(i) the dot product of two n-dimensional vectors.

(ii) the product of an $n \times n$ matrix with an n-dimensional vector.

(iii) the product of two $n \times n$ matrices.

(iv) the inverse of a non-singular $n \times n$ matrix (if it exists).

(v) the determinant of an $n \times n$ matrix via a cofactor expansion.

Is there an algorithm of smaller complexity that allows us to compute the determinant of a matrix? Explain.

(b) Pages 48-49 of the textbook outline the argument for why row reduction of a regular $n \times n$ matrix is $\mathcal{O}\left(\frac{1}{3}n^3\right)$. Apply the same methodology to compute the complexity of of row reducing an arbitrary, regular $m \times n$ matrix to row echelon form with '1's at the pivot entries.

Does your answer depend on the relative size of m and n (i.e., one answer for $m < n$ and another for $m > n$)?

Problem 2. *Getting familiar with MATLAB*

MATLAB (short for "**mat**rix **lab**oratory") is a scripting language that has become a common industrial tool for numerical simulation and computations since its initial release in 1984. We are going to be using it as our primary computational engine for this class, so we should start to get familiar with how we can use it. If this is your first exposure to the software, it codes at the same level of abstraction as C/C++.

(a) Your first job is to create a *function* called `myLU()` whose sole input parameter is an $n \times n$ matrix A. This function should return:

(1) if A is regular, the two matrices L and U corresponding to the LU-factorization of A

or

(2) An error message stating that A is not a regular matrix.

DO NOT implement partial pivoting.

(b) Consider symmetric matrices of the following form:

$$M_n = \begin{bmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n & n-1 & \cdots & 2 \\ n-2 & n-1 & n & \cdots & 3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & 3 & \cdots & n \end{bmatrix}, \quad \text{where } (M_n)_{ij} = n - |i-j|.$$

Use your function `myLU()` to compute what the smallest pivot value p_n is of M_n, and make a conjecture as to the value of $\lim_{n\to\infty} p_n$. There is no need to prove your assertion. Make sure to also comment on how much time it takes to run `myLU()` as you use larger and larger values of n.

(c) Consider the *Hilbert matrix*

$$H_n = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}, \quad \text{where } (H_n)_{ij} = \frac{1}{i+j-1}.$$

Use your function `myLU()` to determine if H_2, H_3, H_4, and H_5 are non-singular. Make a conjecture about the existence of H_n^{-1}, and use your function to test your hypothesis on larger values of n.

Carefully explain what you observe, and comment on what you might need to do to better justify your conjecture.