# CS 6210: Homework 1

Instructor: Anil Damle
Due: September 7, 2018

## Policies

You may discuss the homework problems freely with other students, but please refrain from looking at their code or writeups (or sharing your own). Ultimately, you must implement your own code and write up your own solution to be turned in. Your solution, including plots and requested output from your code should be typeset and submitted via the CMS as a pdf file. Additionally, please submit any code written for the assignment via the CMS as well. This can be done by either including it in your solution as an appendix, or uploading it as a zip file.

---

## Question 1:

This question is intended to serve as a bit of a theoretical warm up for the course, matrix factorizations and norms will play a key role in much of our analysis and understanding of the algorithms we discuss.

1. For any $A \in \mathbb{R}^{n\times n}$ prove that there exists an orthogonal matrix $U \in \mathbb{R}^{n\times n}$ and symmetric positive semi-definite matrix $H \in \mathbb{R}^{n\times n}$ such that

   $$A = UH,$$

   you may use the existence of the SVD in your proof.

2. Prove that for any induced matrix norm $\|\cdot\|$ and $A \in \mathbb{R}^{n\times n}$

   $$\rho(A) \leq \|A\|,$$

   where $\rho(A)$ is the spectral radius of $A$.

3. Given a matrix $A \in \mathbb{R}^{m\times n}$ of rank $k \leq \min\{m, n\}$ prove that

   $$\|A\|_F \leq \sqrt{k}\|A\|_2.$$

4. Prove that the upper triangular matrix $T$ in the Schur decomposition of $A \in \mathbb{C}^{n\times n}$ (denoted here by $A = UTU^*$) is diagonal if and only if $A$ is normal (*i.e.* $AA^* = A^*A$).

5. **(An ungraded, slightly more challenging question)** Since real matrices can have complex eigenvalues, if we want to discuss the Schur form of a non-symmetric real matrix we necessarily have to consider complex numbers. However, it is useful to consider what we can accomplish with only real numbers.

   For the purposes of this problem you may only use the fact that for any matrix $A \in \mathbb{R}^{n\times n}$ there exists at least one scalar $\lambda \in \mathbb{C}$ and associated vector $v \in \mathbb{C}^n$ such that $Av = \lambda v$, and

not assume the existence of any other matrix factorizations. Prove that for for any matrix $A \in \mathbb{R}^{n\times n}$ there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that

$$Q^T A Q = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{mm} \end{bmatrix}$$

where each $R_{ij}$ is real and $R_{ii}$ is either $1 \times 1$ or $2 \times 2$ with complex eigenvalues. Such a decomposition is known are the real Schur decomposition.

## Question 2:

This question is meant to be a bit of a warm up on something we will do repeatedly in this course, which is implement, validate, and test algorithms we discuss. Since we have not really discussed any algorithms yet, we will start by simply exploring the variety of performance we can observe even with something as simple as matrix-matrix multiplication. Consider/implement the following four algorithms for computing $C = AB$:

1. $C(i,j) = \sum_k A(i,k)B(k,j)$, for this part you can only use built in scalar multiplication
2. $C(i,j) = A(i,:)B(:,j)$, you may now leverage your chosen languages calls to routines for computing inner products
3. $C = \sum_k A(:,k)B(k,:)$, you may now leverage your chosen languages calls to compute outer products and add matrices
4. $C(:,i) = AB(:,i)$, you may now leverage your chosen languages calls to compute matrix-vector products.
5. As a point of comparison we will also use the "built in" routine for computing matrix-matrix multiplication (*e.g.* simply writing $C = A^*B$ in Matlab), this is our way of accessing the routine for matrix-matrix multiplication from BLAS (`http://www.netlib.org/blas/`).

For all the above algorithms clearly illustrate that your implementation is $\mathcal{O}\left(n^3\right)$ (as part of this question, argue why your choice of plot clearly and unambiguously illustrates the correct scaling, think about the axes), compare and contrast their performance, and argue about why you believe you might be seeing such differences.

## Question 3:

Given two matrices $A, B \in \mathbb{R}^{n\times n}$ and let $\text{fl}(AB)$ be the result of computing $AB$ with floating point arithmetic using an algorithm that computes the entries of $\text{fl}(AB)$ either as inner products or as the sum of entries of appropriate outer products. Let

$$\text{fl}(AB) = AB + E,$$

determine an element wise bound on the entries of $E$ in terms of machine precision $\mu$, $n$, and the entries of $A$ and $B$. For this question you may only use floating point properties of the four basic arithmetic operations. For simplicity you may assume $A$ and $B$ are exactly representable in floating point.