MATH 270A – NUMERICAL LINEAR ALGEBRA – PROGRAMMING HOMEWORK 1

Due Friday, November 2nd. Please show your teaching assistant. Groups of up to three people.

Exercise 1 (QR Decomposition and Random Matrices)
We perform some numerical experiments with an implementation of the QR factorization based on Gram-Schmidt. The programming assignment can be solved in any imperative programming language (such as C, C++, Fortran, JavaScript, Pascal, Python, Visual Basic, MATLAB, ...).

(1) We should test our implementation with sufficiently test problems. For example, we can produce a number of matrices with random entries. Implement an algorithm that constructs a random real $n \times n$ matrix A whose entries are Gaussian variables centered at 0 and with standard deviation $\sigma > 0$. Your programming language may already offer such a random floating-point generator, otherwise you have to build one for by yourself.

(2) Implement the QR decomposition via Gram-Schmidt process with the algorithm from the lecture:

FOR $j = 1, \dots, n$ DO
  $q^j = a^j$
  FOR $k = 1, \dots, j-1$ DO
    $R_{kj} = \tilde{q}^k \cdot q^j$
    $q^j = q^j - R_{kj}\tilde{q}^k$
  END FOR
  $\tilde{q}^j = q^j / \|q^j\|$
END FOR

Your implementation should produce the matrices Q and R.

(3) Have the implementation run for different matrix sizes (say, $n = 2, 3, 4, 10, 20$) for a number of randomly generated test cases (say, $M = 10$). How can you reasonably assess whether the matrices satisfy $A = QR$? How can you reasonably assess whether Q is orthogonal?

(4) The algorithm in the lecture is based on what is known as modified Gram-Schmidt process. This is the standard way of implementing this process. Alternatively, we could implement the Gram-Schmidt process as follows:

FOR $j = 1, \dots, n$ DO
  $q^j = a^j$
  FOR $k = 1, \dots, j-1$ DO $R_{kj} = \tilde{q}^k \cdot q^j$ END FOR
  FOR $k = 1, \dots, j-1$ DO $q^j = q^j - R_{kj}\tilde{q}^k$ END FOR
  $\tilde{q}^j = q^j / \|q^j\|$
END FOR

That is, given our current column a^j, we first compute the scalar products $a^j \cdot \tilde{q}^j$ and then subtract the corresponding multiples of $\tilde{q}^j$. Implement this algorithm; your implementation should produce the matrices Q and R. Run the same tests as for your previous implementation. Do you notice a difference?

Remark 1
The above procedure has the following interesting side effect: if we take the QR decomposition $A = QR$ of a matrix $A \in \mathbb{R}^{n \times n}$ whose entries are Gaussian variables around zero and fixed standard deviation then we produce an orthogonal matrix Q that is a random matrix by itself. It turns out that the probability distribution of Q is the so-called *Haar measure*.

The Haar measure of the orthogonal group $\mathcal{O}(n)$ has the nice property of being invariant under the group multiplication: the probability of a set $S \subseteq \mathcal{O}(n)$ is the same as the probability of $P \cdot S$ for any $P \in \mathcal{O}(n)$. The Haar measure is the "natural" probability distribution over $\mathcal{O}(n)$.