CS513, Spring 13

Prof. Ron

Assignment #1

Due February 12 , 2013

Read carefully the introductory parts in the book: Lecture 1 and Lecture 2. Note that the notions of *orthogonal matrix* and *unitary matrix* are essentially the same: a unitary matrix is an orthogonal matrix with possibly complex entries (and then one needs to suitably apply conjugations to the complex numbers in the definitions).

(1) Do Q. 1.1, page 9. Use `Matlab` for the computations (and turn in the `Matlab` code you have used, too). Do also q. 2.3, page 15. For the latter, you may assume that the matrix is symmetric (i.e., A is real-valued, and $A' = A$), and may examine there expressions of the form $\langle Ax, y\rangle$.

(2) Let Q be an $m \times m$ real matrix that satisfies, for every vector $x \in \mathbb{R}^m$,

$$\|Qx\| = \|x\|. \qquad (\aleph)$$

Here, $\|x\| := \sqrt{\sum_{i=1}^m x_i^2}$.

(a) Show that 1 is the only eigenvalue of $Q'Q$, i.e., show that $\sigma(Q'Q) = \{1\}$.

(b) It is known that every *symmetric* matrix A is diagonalizable, i.e., can be written as $A = PDP^{-1}$, for some diagonal D, and invertible P. Prove that $Q'Q$ is symmetric (for any matrix Q), and use that in order to show that the matrix Q in this question (i.e., the one that satisfies $(\aleph)$) is *orthogonal*.

To this end, note: for every vectors $x, y \in \mathbb{R}^m$ and any real matrix $Q_{m\times m}$,

$$\|x\|^2 = \langle x, x\rangle, \quad \langle Qx, y\rangle = \langle x, Q'y\rangle.$$

(3)

Copy from our web site the file `demos/qr/sloppy_qr.m` into a directory in your account where you run `Matlab`. The code in that file QR factors a matrix $A_{m\times n}$ $(m \geq n)$ into $Q_{m\times n}R_{n\times n}$.

Assuming that A is square, you need to find the *complexity* of the algorithm used in that file for the QR factorization, i.e., the number of operations. For that, run the code on square matrices A of different sizes, and keep an operation count for each run (the number of operations of each line of code is listed as a comment in the .m file). Based on the experiment, determine the complexity in the form cn^k. (If you cannot do that, try at least to find the *order* of the algorithm i.e., determine the k in the $O(n^k)$ order of the algorithm. Take one of your test matrices to be of large order, say 100×100)