CS513, Spring 21

Prof. Ron

HW #5, Factor=1 (or 1.1, see below)

Due Feb. 27, 2021

(1), Factor=.2 QR factor the matrix Z on page 76 of the book. You may use `Matlab` to this end, but must show all the intermediate results of the your work. (So, you cannot use the `qr` routine of `Matlab`.) You may use any code you wish, including everything used in the class demos, but you must document such usage. Your answer will be upgraded to .3 factor (50% bonus), if you compute efficiently the QR-factorization, along the lines of the next question.

(2), Factor=.8

(a) Describe an efficient algorithm for computing the product Hv, with H a Householder matrix, and v a vector. You should assume that H is not given explicitly, and that, instead, the input on H is the corresponding vector w.

Note: The multiplication of a generic square matrix of size m by a vector consumes $O(m^2)$ operations, which is pretty obvious from the definition of such multiplication (and surely you cannot do better, since the matrix has m^2 entries and you need to process each one of them at least once). However, the vector w has only m entries, and the vector v also has only m entries, so a smart algorithm should be able to compute Hv, directly from w, v, using $O(m)$ operations. This is exactly what you are asked to establish!! Note that once you constructed the matrix H from the vector w, you already lost the game (since, once you compute H, your only option it to treat it as a generic matrix), even if you ignore the fact that it already took you $O(m^2)$ operations to compute all the entries of H.

(b) Using (a) above, write a short code that *efficiently* solves a square linear system of equations using QR factorization.

Guideline: Never compute any Householder matrix, and do not compute the Q factor, only the R factor. Simply save the vector w of each Householder. So, for example, when you need to compute $Q'b$, then, since

$$Q = H_1 H_2 \ldots H_{m-1},$$

you have

$$Q'b = (H_{m-1} \ldots H_1)b = ...(H_2(H_1 b)).$$

(c) Run your algorithm on two examples of your choice. The two matrices you choose should have at least order 5 each, and should be invertible. Do not choose special matrices (i.e., your matrix should not be upper triangular, or orthogonal, not even symmetric).

(d) Estimate the complexity of your algorithm, i.e., the number of operations it uses as a function of the order of the system. Note: if your algorithm is designed correctly, the complexity is determined by the QR factorization itself, and not by the subsequent need to solve the two derived systems.