CS 598AK Project Report

My project consisted of learning and implementing both Householder QR factorization with column-pivoting variants and state-of-the-art randomized QR factorization with column-pivoting algorithms [2, 4, 5, 3]. I started by re-learning Householder QR factorization with column-pivoting [1], since my understanding of this algorithm was never satisfactory. I then progressed to understanding the rich algorithmic differences and tradeoffs that occur when this algorithm is re-organized to use BLAS level-1, level-2, and level-3 kernels [6]. To understand the BLAS level-3 variant, I needed to first understand the efficient storage schemes needed for aggregating Householder updates [7]. Finally, before implementing the different randomized QR factorization algorithms, I needed to understand the general idea of randomized QR factorization [3]. After reading the recent papers, I obtained a strong understanding of the most recent techniques for efficient QR factorization with column pivoting using random sampling [2, 4, 5].

I have implemented the following algorithms in python: Householder QR with BLAS level 1, Householder QR with BLAS level 2, Householder QR with BLAS level 3, Householder QR with column pivoting with BLAS level 1, Householder QR with column pivoting with BLAS level 2, Householder QR with column pivoting with BLAS level 3, Single-sampled randomized Householder QR with column pivoting, Repeated-sampled randomized Householder QR with column pivoting, Randomized Householder QR with column pivoting, and Truncated Randomized Householder QR with column pivoting and no trailing matrix update. For the column-pivoting algorithms, I have implemented them both with rank-detection and user-specified rank approximation capabilities. If you glance at the most recent paper pertaining to randomized QR factorization [2], you will notice that only high-level pseudocode is given. Therefore, I needed to have a strong understanding of the details of each algorithm. I encountered many bugs during implementation that greatly helped my understanding.

The differences in the non-randomized QR factorization with column-pivoting (QRCP) algorithms are explained by the BLAS level they are organized to exploit and are well documented. The randomized algorithms each use the non-randomized QRCP and QR as building blocks. The non-randomized QRCP is performed on the sampled matrix in order to choose pivot columns. The non-randomized QR then blindly factors the panel once the pivoted columns have been swapped to the front of the trailing matrix and updated with the delayed Householder reflectors. The single-sampled randomized QRCP allows for a single approximation rank and performs a single sample to choose the specified number of pivots. The repeated-sampled randomized QRCP allows for rank detection because it can choose the next pivots one block size at a time until it detects a column norm below tolerance. The randomized QRCP advances the repeated-sample variant in its ability to downsample the sample matrix and avoid recomputation. The truncated QRCP further advances the randomized QRCP in its ability to avoid the trailing matrix update. These differences are explained at a very high level. For sake of brevity in this report, for more details please see the cited papers or talk to me.

I did not come close to implementing a distributed-memory version of the truncated randomized QRCP algorithm. I greatly underestimated the time that would take, but that is the next step in my line of research so I would be happy to talk to you about those results in the future if interested. In terms of contribution, I think the testing toolbox I have generated in python is very useful in understanding and comparing the results of each of the different algorithms. Of course there is still some small implemetation details I want to clean up and optimize, but what I have implemented allows the interested user to play around and experiment with the most recent algorithms for randomized QRCP. I myself find it very fun and useful and am still in the process of understanding the results that my code generates. This detailed analysis alone will take a long time and will be very interesting in the near future. I have not included a plot because you can create any plot you want when you run the code.

I have provided all of the code in a python Jupiter notebook. I have commented out extensive testing code, so all you need to do is proceed down the code boxes/cells and run each, since they are all either function definitions or strings. Finally, you can run the last two cells and follow the instructions to obtain the results. The first is a test that tests individual methods while the second will produce plots for a range of matrix ranks. Have fun playing around with it and please offer feedback! I have done extensive testing and most if not all corner cases have been addressed. Note that you cannot specify an underdetermined matrix. I want to add the final algorithm explored in [2], which details a truncted SVD algorithm that uses the truncated QRCP and can factor matrices of any dimension.

References

[1] Peter Businger and Gene H Golub. Linear least squares solutions by householder transformations. *Numerische Mathematik*, 7(3):269–276, 1965.

[2] Jed A Duersch and Ming Gu. Randomized qr with column pivoting. *SIAM Journal on Scientific Computing*, 2017.

[3] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[4] Per-Gunnar Martinsson, Gregorio Quintana-Orti, Nathan Heavner, and Robert van de Geijn. Householder qr factorization with randomization for column pivoting (hqrrp). flame working note# 78. *arXiv preprint arXiv:1512.02671*, 2015.

[5] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM Journal on Scientific Computing*, 38(5):S485–S507, 2016.

[6] Gregorio Quintana-Ortí, Xiaobai Sun, and Christian H Bischof. A blas-3 version of the qr factorization with column pivoting. *SIAM Journal on Scientific Computing*, 19(5):1486–1494, 1998.

[7] Robert Schreiber and Charles Van Loan. A storage-efficient wy representation for products of householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 10(1):53–57, 1989.