AMath 584, Autumn 2023

Assignment 3.

Due Thurs., Oct. 19, at 11:59 pm.

Reading: Lectures 6-7 in textbook

1. p. 47, Exercises 6.1, 6.2
2. p. 55, Exercise 7.5
3. **Even skinnier QR.** Let $A \in \mathbb{C}^{m\times n}$, $m \geq n$, and suppose $\text{rank}(A) = k < n$. Prove that there exists a factorization $A = Q_k R_k P^T$, where Q_k is an $m \times k$ matrix with orthonormal columns, R_k is an upper triangular $k \times n$ matrix (think of this as "upper trapezoidal"), and P is a permutation matrix. *Note: Algorithms that compute this factorization are called pivoted QR decomposition algorithms.*
4. **SVD from low rank factors.** (All the .m files mentioned in this problem can be compiled into a single file if you prefer.) Let $A \in \mathbb{C}^{m\times m}$ be a rank k matrix, and suppose we know and can access factors $X, Y \in \mathbb{C}^{m\times k}$ such that $A = XY^*$.
   (a) Using skinny QR decompositions (in MATLAB call `qr(·, 0)`), devise an algorithm `[u, s, v] = LRsvd(X,Y)` that computes the skinny singular value decomposition of A from X, Y in $\mathcal{O}(mk^2 + k^3)$ floating point operations. Write down your algorithm in pseudocode in your homework solution sheet.
   (b) Implement the algorithm in MATLAB (or language of your choice). You will turn in your code as a file: `LRsvd.m`.
   (c) To test the accuracy of the code, you will create another file. In addition to the `LRsvd.m` function, create a `testLRsvd.m` file that when executed, does the following:
      i. **Constructs and runs an accuracy test:** To do this, write code that populates X, Y for $m = 2048$, $k = 50$, with random entries, and then computes and prints the error $\|A - USV^*\|_2/\|A\|_2$, where A is computed via the product XY^*. This should give you around 15 digits of accuracy.
      ii. **Contstructs and runs a timing test:** Fix $k = 50$. For $m_j = 2^j$, starting with $j = 6$ and going up to $j = 12$, compute random $X, Y \in \mathbb{C}^{m_j\times k}$, and use these matrices to test how long it takes to run your algorithm for computing the skinny SVD, vs. the "naive" algorithm for SVD computation: forming A and computing its svd via MATLAB's `svd(·)` command (you don't need to go further and reduce the svd by chopping off the parts associated with the null space of A, though in principle this added step would be needed to get a truly "skinny" svd). Create a compelling plot that compares the two

methods as m_j grows. Summarize the results from the test in your PDF or written HW solutions.

The basic code for testing how long something takes is as follows (this doesn't show how to store the result/use a loop):

```
t = tic; runmyop; totaltime=toc(t).
```

This `tic` command starts a stopwatch timer and also names it `t`, the command `runmyop` is then executed (this is a stand-in for the process you want to time). After the process terminates, the command `totaltime = toc(t)` stops the stopwatch `t` and records the running time as the variable `totaltime`.