Homework set 1 — APPM4720/5720, Spring 2016

Problem 1: Let $\mathbf{A}$ be an $m \times n$ matrix, set $p = \min(m, n)$, and suppose that the singular value decomposition of $\mathbf{A}$ takes the form

$$(1) \qquad \underset{m\times n}{\mathbf{A}} = \underset{m\times p}{\mathbf{U}}\;\underset{p\times p}{\mathbf{D}}\;\underset{p\times n}{\mathbf{V}^*}.$$

Let k be an integer such that $1 \le k < p$ and let $\mathbf{A}_k$ denote the truncation of the SVD to the first k terms:

$$\mathbf{A}_k = \mathbf{U}(:, 1 : k)\,\mathbf{D}(1 : k, 1 : k)\,\mathbf{V}(:, 1 : k)^*.$$

Recall the definitions of the spectral and Frobenius norms:

$$\|\mathbf{A}\| = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}, \qquad \text{and} \qquad \|\mathbf{A}\|_{\mathrm{F}} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |\mathbf{A}(i,j)|^2\right)^{1/2}.$$

Prove directly from the definitions of the norms that

$$\|\mathbf{A} - \mathbf{A}_k\| = \sigma_{k+1}$$

and that

$$\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}} = \left(\sum_{j=k+1}^{p} \sigma_j^2\right)^{1/2}.$$

Problem 2: On the course webpage, download the file `hw01p2.m`. This file contains an implementation of the column pivoted QR algorithm that computes a rank-k approximation to a matrix, for any given k. Your task is now to do two modifications to the code:

(a) Starting with the function `CPQR_given_rank` write a new function with calling sequence

```
[Q,R,ind] = CPQR_given_tolerance(A,acc)
```

that takes as input an accuracy, and computes a low-rank approximation to a matrix that is accurate to precision "acc".

(b) Write a function with calling sequence

```
[U,D,V] = SVD_given_tolerance(A,acc)
```

that computes a diagonal matrix $\mathbf{D}$, and orthonormal matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$\|\mathbf{A} - \mathbf{U}\mathbf{D}\mathbf{V}^*\| \le \varepsilon,$$

where ε is the given tolerance. The idea is to use the function `CPQR_given_tolerance(A,acc)` that you created in part (a).

Please hand in a print-out of the code that you created.

Extra problem: The file `hw01p2.m` creates a plot that shows the accuracies of two low-rank approximations: The truncated SVD on the one hand, and the truncated QR on the other. Let me encourage you to play around with this a bit, try different matrices and see how the approximations errors compare. There is no need to hand anything in!

Problem 3: In this example, we investigate the effect *blocking* has on execution time of matrix computations.

(a) Suppose that we are given two $n \times n$ matrices $\mathbf{B}$ and $\mathbf{C}$ and that we seek to compute $\mathbf{A} = \mathbf{BC}$. We could do this in Matlab either by just typing `A = B*C`, or, we could write a loop

for $i = 1 : n$
```
A(:,i) = B*C(:,i)
```
end for

The code `hw01p3.m` illustrates the two techniques. It turns out that while the two methods are mathematically equivalent, doing it via a loop is much slower. In this problem, please measure the time T_n required for several different values of n. Test the hypothesis that $T_n = C\,n^3$ by plotting your measure valued of T_n versus n in a log-log-diagram. Fit a straight line through the points, and estimate C. Hand in the graph and the values of C that you estimate for the two methods.

(b) Repeat the problem in (a), but now compare three different matrix factorization algorithms:
- `[L,U] = lu(A)`
  This factorization can be blocked. It is fast, but not good for low-rank approximation.
- `[Q,R,J] = qr(A,'vector')`
  Column pivoted QR factorization — intermediately fast, and good for low-rank approximation.
- `[U,D,V] = svd(A)`
  Singular value decomposition — slowest, but excellent for low-rank approximation.

(We used LU here as a stand-in for non-pivoted QR factorization, since I think there is no non-pivoted QR factorization built in to Matlab. If I am wrong and you find it, then please use that instead!)