Homework set 2 — APPM4720/5720, Spring 2016

Problem 1: Recall the *single pass* RSVD shown in Figure 1. Consider the following piece of Matlab code:

```
m=12; n=20; b=3; k=5;
A  = rand(m,n);
Gc = randn(n,k);
Gr = randn(m,k);
Yc = zeros(m,k);
Yr = zeros(n,k);

for i = 1:(n/b)
  ind       = (i-1)*b + (1:b);
  A_slice   = A(:,ind);
  Yc        = Yc + A_slice*Gc(ind,:);
  ???????????????????????????????????
end

fprintf(1,'Error in Yc = %12.3e\n',max(max(abs(Yc - A *Gc))))
fprintf(1,'Error in Yr = %12.3e\n',max(max(abs(Yr - A'*Gr))))
```

This snippet of code emulates how a streaming algorithm would interact with $\mathbf{A}$ — you read a set of b columns at a time, and use the information in $\mathbf{A}_{\text{slice}}$ to build $\mathbf{Y}_{\text{c}}$ piece by piece.

(a) Write code to replace the question marks. The result of the new code should be that after the loop completes, the matrix `Yr` has also been computed. Note that this line should reference only `A_slice`, not `A` itself. (This can be solved by a single line of code, but if you use more, then that is fine too.)

(b) Currently, the code only works if `n` is an integer multiple of `b`. Modify the code so that it works for any block size `b`.

Hand in the code you write.

```
ALGORITHM: SINGLE-PASS RANDOMIZED SVD FOR A GENERAL MATRIX

Inputs: An $m \times n$ matrix $\mathbf{A}$, a target rank $k$, and an over-sampling parameter $p$ (say $p = 10$).
Outputs: Matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{D}$ in an approximate rank-$k$ SVD of $\mathbf{A}$. (I.e. $\mathbf{U}$ and $\mathbf{V}$ are ON and $\mathbf{D}$ is diagonal.)
Stage A:
  (1) Form two Gaussian random matrices $\mathbf{G}_{\text{c}} = \texttt{randn}(n, k+p)$ and $\mathbf{G}_{\text{r}} = \texttt{randn}(m, k+p)$.
  (2) Form the sample matrices $\mathbf{Y}_{\text{c}} = \mathbf{A}\,\mathbf{G}_{\text{c}}$ and $\mathbf{Y}_{\text{r}} = \mathbf{A}^*\,\mathbf{G}_{\text{r}}$.
  (3) Form ON matrices $\mathbf{Q}_{\text{c}}$ and $\mathbf{Q}_{\text{r}}$ consisting of the $k$ dominant left singular vectors of $\mathbf{Y}_{\text{c}}$ and $\mathbf{Y}_{\text{r}}$.
Stage B:
  (4) Let $\mathbf{C}$ denote the $k \times k$ least squares solution of the joint system of equations formed by the equations $\left(\mathbf{G}_{\text{r}}^*\mathbf{Q}_{\text{c}}\right)\mathbf{C} = \mathbf{Y}_{\text{r}}^*\mathbf{Q}_{\text{r}}$ and $\mathbf{C}\left(\mathbf{Q}_{\text{r}}^*\mathbf{G}_{\text{c}}\right) = \mathbf{Q}_{\text{c}}^*\mathbf{Y}_{\text{c}}$.
  (5) Decompose the matrix $\mathbf{C}$ in a singular value decomposition $[\hat{\mathbf{U}},\,\mathbf{D},\,\hat{\mathbf{V}}] = \texttt{svd}(\mathbf{C})$.
  (6) Form $\mathbf{U} = \mathbf{Q}_{\text{c}}\hat{\mathbf{U}}$ and $\mathbf{V} = \mathbf{Q}_{\text{r}}\hat{\mathbf{V}}$.
```

FIGURE 1. A basic randomized algorithm single-pass algorithm suitable for a general matrix.

Problem 2: On the course webpage, download the file `hw02p2.m`. This file contains an implementation of the basic RSVD scheme. It computes approximate matrix factorizations using the basic RSVD, and plots the approximation error versus the minimum error as produced by the truncated (full) SVD. The error is reported in both the spectral and the Frobenius norms.

For this problem, code up the single pass algorithm described in Figure 1 and include it in the comparison. Hand in a printout of your implementation of the single pass algorithm (you do not need to print out the driver code, etc, just the actual subroutine). Also hand in the error plots for three different sets of test matrices. In the code that you can download, two test cases are already included. You are welcome to use these two. For the third, come up with some matrix you find interesting yourself! It could be dense, sparse, etc. Just remember that for any of this to make sense, the singular values of the matrix you pick must show at least some degree of decay.

Note: In step (4) of the algorithm, the matrix $\mathbf{C}$ is determined by jointly solving two matrix equations. Note that this is a bit complicated to implement. For simplicity, simply pick one of the two equations and determine $\mathbf{C}$ by solving that one, ignoring the other.

Problem 3: Implement the SRFT code described in Figure 2. In your implementation, just apply the *full* FFT to the rows of the matrix $\mathbf{AD}$, and then extract $k + p$ randomly picked columns. This will in principle be slower than a properly implemented subsampled FFT that never evaluates the unneeded indices at all. But, our objective here is not to test speed, just to see how the method compares in terms of *accuracy*.

Hand in the same material that is specified for Problem 2 — printout of the subroutine, and three figures showing the errors for three different matrices. (You are welcome to plot the errors of both the SRFT and the single-pass algorithm in the same figure, and hand in one set of three plots that cover both Problem 2 and Problem 3.)

ALGORITHM: BASTARDIZED RANDOMIZED SVD BASED ON THE SRFT

Inputs: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = k$).

Outputs: Matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{D}$ in an approximate rank-k SVD of $\mathbf{A}$. (I.e. $\mathbf{U}$ and $\mathbf{V}$ are ON and $\mathbf{D}$ is diagonal.)

Note: This algorithm explicitly computes $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. This precludes $O(mn \log k)$ complexity, but lets us analyze how well the SRFT works as a random matrix. (Also, the way the application of $\mathbf{R} = \mathbf{DFS}$ is described is computationally inefficient, but is mathematically equivalent to a "proper" SRFT.)

```
Stage A:
    (1) Draw $n$ random numbers $\mathbf{d} = \{d_j\}_{j=1}^{n}$ and set $\mathbf{D} = \texttt{diag}(\mathbf{d})$. Draw at random without replacement
        an index vector $J$ of length $k + p$ from the set $\{1, 2, \dots, n\}$.
    (2) Compute $\mathbf{Z} = \mathbf{ADF}$ where $\mathbf{F}$ is the discrete Fourier transform. Then form $\mathbf{Y} = \mathbf{Z}(:, J)$.
    (3) Form an ON matrix $\mathbf{Q}$ by orthonormalizing the columns of $\mathbf{Y}$ so that $\mathbf{Q} = \texttt{orth}(\mathbf{Y})$.
Stage B:
    (4) Compute $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$.
    (5) Decompose the matrix $\mathbf{B}$ in a singular value decomposition $[\hat{\mathbf{U}}, \mathbf{D}, \mathbf{V}] = \texttt{svd(B,'econ')}$.
    (6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.
```

FIGURE 2. A randomized algorithm based on a subsampled Fourier transform. Observe that the output matrices $\mathbf{U}$ and $\mathbf{V}$ will be complex even when $\mathbf{A}$ is real.