MATH 8610 (SPRING 2023) HOMEWORK 6

Assigned 03/13/23, due 03/20/23 at 11:59pm.

Instructor: Dr. Fei Xue, Martin O-203, fxue@clemson.edu.

1. **[Q1] (10 points)** Trefethen's book, Problem 6.5.

2. **[Q2] (10 points)** Review our analysis of the bound on the relative forward error of singular value computation by using a backward stable eigenvalue algorithm for A^TA. That is, $\frac{|\widetilde{\sigma}_k-\sigma_k|}{\sigma_k} \le \mathcal{O}\left(\frac{\sigma_1^2}{\sigma_k^2}\epsilon_{mach}\right)$, where $\widetilde{\sigma}_k = \sqrt{\widetilde{\lambda}_k}$ and $\widetilde{\lambda}_k$ denotes the computed k-th largest eigenvalue of A^TA. Instead, if we use a backward stable eigenvalue algorithm for $\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, show that the relative forward error of singular value computation would be bounded by $\mathcal{O}\left(\frac{\sigma_1}{\sigma_k}\epsilon_{mach}\right)$, assuming that square root computation is exact. Explain the advantage of the new error bound.

3. **[Q3] (10 points)** Read the introduction to the Golub-Kahan-Lanczos method, at `http://www.netlib.org/utk/people/JackDongarra/etemplates/node198.html` and the uploaded code implementation `HW6_GKLsvds.m`.
(a) Give a general description of the functionality of GKL; describe the main difference between the original GKL and the code.
(b) Download the zipped file `HW6_pics.zip`, unzip it, load the first jpeg file by

```
picA = double(imread('picA.jpg'));
```

and run

```
rk = 160;
tic; [Us1,Ss1,Vs1] = HW6_GKLsvds(picA(:,:,1),rk); toc;
tic; [Us2,Ss2,Vs2] = HW6_GKLsvds(picA(:,:,2),rk); toc;
tic; [Us3,Ss3,Vs3] = HW6_GKLsvds(picA(:,:,3),rk); toc;
tic; [U1,S1,V1] = svd(picA(:,:,1),0); toc;
tic; [U2,S2,V2] = svd(picA(:,:,2),0); toc;
tic; [U3,S3,V3] = svd(picA(:,:,3),0); toc;
```

Then, run MATLAB's command `whos` to see the memory used by `picA`, and by `Us1,Vs1,Us2,Vs2,Us3` and `Vs3` all together. Compare the timing used for computing and the memory used for storing the full and partial SVD of this picture.
(*Note: we are competing MATLAB code with the built-in C/FORTRAN* code *in timing, and our timing should improve considerably if our GKL code is in C/FORTRAN*)

(c) Finally, run MATLAB's command

```
picAh = zeros(size(picA));
picAh(:,:,1) = Us1*Ss1*Vs1';
picAh(:,:,2) = Us2*Ss2*Vs2';
picAh(:,:,3) = Us3*Ss3*Vs3';
disp([norm(picAh(:,:,1)-picA(:,:,1),'fro')/norm(picA(:,:,1),'fro') ...
norm(picAh(:,:,2)-picA(:,:,2),'fro')/norm(picA(:,:,2),'fro') ...
norm(picAh(:,:,3)-picA(:,:,3),'fro')/norm(picA(:,:,3),'fro')]);

figure(1);  imshow(uint8(picA));  axis equal;
figure(2);  imshow(uint8(picAh));  axis equal;
```

Are you satisfied with the quality of the image generated by `picAh`? If not, let `rk = 320`, rerun `HW6_GKLsvds`, compare the timing and memory cost for computing the partial SVD. Then show the images again. Are you satisfied now?

Repeat the above procedure for the other three pictures. Make some general comments on the computation and use of partial SVD for compressing images. In particular, give an estimate of the arithmetic cost of this partial SVD and full SVD applied to an image of dimension m-by-n, in a form of $\mathcal{O}(\cdot)$. For a given rank $\texttt{rk} \ll \min\{m,n\}$, how does the cost of partial SVD compared to full SVD as $\min\{m,n\}$ increases? In a recent development by random sketching, fulll orthogonalization of `rk` vectors of elements m or n ($\texttt{rk} \ll \min\{m,n\}$) needs only $\mathcal{O}(\texttt{rk}^3) + \mathcal{O}(\max\{m,n\}\texttt{rk})$ flops. Compare the cost of partial and full SVD if such a fast orthogonalization can be used.

(d) (**4 extra credit for fun, only for those who finished (a)(b)(c)**). Search the title of each artwork, the name of the artist, the approximate year of creation, and the current location of the artwork. Info of 4 paintings qualifies full extra credit. *Do your own research, instead of using others' findings, even for this leisure problem.*