Math 401 - Groupwork #12
Matrix Approximation and Image Compression

Instructions: Work through the problems below in order. Write down all of your answers on paper, to be turned in at the end of class. I do not want you to turn in any MATLAB code. Make sure that the name of every student in the group is on the paper. Some problems ask you to save images and email them to me (`ayashins@math.umd.edu`) at the end.

Here are some MATLAB remarks:

- The command `norm(A,'fro')` will give the Frobenius norm $\|A\|_F$.
- We run `[U,S,V] = svd(A)` to get the SVD of A. If we want to delete certain singular values in `S`, we can do it manually by typing things like `S(3,3) = 0`. This changes the third singular value to 0.
- Many singular values can be set to 0 by using a `for` loop. Alternatively, one can assign a block of zeros to a chunk of `S`. For example, if `S` is 8×10 and we want to get rid of all singular values except for the first three, we could enter `S(4:8, 4:10) = zeros(5,7)`, which assigns a 5×7 block of 0's to the submatrix formed by rows 4 through 8 and columns 4 through 10.
- When setting singular values to be 0, it may be a good idea to copy `S` first, say `NS = S`. Then one can edit `NS`, and construct the approximation by `NA = U*NS*transpose(V)`

1. Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.
   (a) Compute $\|A\|_F$ by doing arithmetic with the entries. Give the exact answer, and then a decimal approximation.
   (b) Compute $\|A\|_F$ by using the `norm(A,'fro')` command.
   (c) Compute $\|A\|_F$ by using the singular values of A. Be clear about what you are doing.
   (d) Let A_1 denote the best rank 1 approximation to A. Without computing A_1, determine the error $\|A - A_1\|_F$ in this approximation.
   (e) Compute A_1. Then compute $\|A - A_1\|_F$ directly to confirm your prediction.

2. The columns of $B = \begin{bmatrix} 2.1 & 8.4 & 12.6 & 18.9 & 27.3 & 31.5 \\ 1.23 & 4.94 & 7.35 & 10.99 & 15.78 & 18.24 \end{bmatrix}$ *almost* lie on the same line through the origin.
   (a) Find the best rank 1 approximation of B.
   (b) Use the rank 1 approximation of B to determine a line $y = mx$ which best fits the data. (Note: this is different than the curve fitting with least-squares that we did in the past. The line we get in this problem is an *orthogonal least-squares line*, which minimizes the orthogonal distances from the points to the line. In the past, our lines minimized the vertical distance from the points to the line.)

More on the back!

3. Your friend did some MATLAB computations and found equations for the following three planes

$$\begin{cases} 1.993x - 2.657y + 4.650z = 7.307 \\ -0.246x + 0.737y - 0.368z = -0.491 \\ 0.407x + 0.815y + 1.629z = 2.851 \end{cases}$$

Your friend tells you that the intersection of these three planes is a line. The equations were copied down by hand from the MATLAB output, and no mistakes were made in transcribing them.

(a) Let C denote the augmented matrix of this linear system. Based on what your friend told you, what is $\operatorname{rank} C$?

(b) To try to find the line of intersection, compute `rref(C)`. Also compute `rank(C)`. Something is not quite right here. Can you explain a reason for the discrepancy? (It's not your friend, who is very trustworthy and does excellent work in MATLAB.)

(c) Find the singular values of C.

(d) Make an adjustment so that your reduced row echelon form has the free variable that you expect, and then describe the solution to this system in parametric vector form.

4. Download the script `image_compression.m`, which is posted in ELMS. Also download the images `dog.jpg`, `android.jpg`, `homer.jpg`. Make sure you put the script and the images in the same folder. All images are 200×200 pixels.

(a) Start with the image `dog.jpg`. Experiment with the script by running it for different values of k, the number of singular values that we *keep*. Run it for $k = 100, 75, 50, 25, 10, 5, 1$. Save the images for $k = 50, 10, 1$ as `dog50.jpg`, `dog10.jpg`, `dog1.jpg`. (Save them as you usually do in MATLAB. Click the save button from the figure window. Switch the format to jpg. Note we are not actually compressing the image when we save it, so don't expect the file to be smaller.)

(b) Write down the first five singular values, as well as the last five. Then also write down $\sigma_{25}, \sigma_{50}, \sigma_{100}$.

(c) We measured image quality by the quantity

$$\frac{\sigma_1^2 + \ldots + \sigma_k^2}{\sigma_1^2 + \ldots + \sigma_k^2 + \ldots + \sigma_r^2}$$

(The denominator has all the singular values.) Note that this quantity can be computed in MATLAB by computing certain Frobenius norms (don't forget they have square roots!) Compute this quantity for $k = 50, 25, 10, 1$.

(d) Find the smallest value of k for which this image quality quantity is at least 0.997.

5. Repeat parts (a) through (d) of the previous problem for the image `android.jpg`.

(e) Comment on the differences you noticed between your answers for the dog and for android.

6. Download the script `compression_animation.m` to your computer. Recall that the compressed image (finite rank approximation) has the form

$$A_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T.$$

This script provides an animation in which the rank 1 matrices in this formula are added one at a time. That is, the first frame is $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$, the second frame is $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T$, and so on until the last frame, which is $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T$. Apply this script to the images `dog.jpg`, `android.jpg`, `homer.jpg`, and enjoy! Try downloading a jpeg image from Google and applying the script. (Don't have to turn anything in for this question.)