EXAM 1

The material we have covered so far has been designed to support the following learning goals:

- understand sources of error in scientific computing (modeling, measurement, truncation, and rounding errors)
- know how to measure errors (absolute error, relative error, errors in different norms)
- be comfortable with floating point arithmetic
- know how errors propagate through arithmetic calculations
- understand forward and backward error analysis
- know the definition of a condition number
- understand pivoting with Gaussian elimination
- understand the singular value decomposition (SVD)
- understand the relation between the SVD and the pseudoinverse
- use the SVD in practical applications (e.g. image deblurring)
- learn how to write functional, well-documented Matlab code
- learn how to use Latex

This exam is designed to test your mastery of these goals. It is an open book, open note take home exam. It is not permissible to discuss this exam with any person other than your professor (you may feel free to ask your professor for hints to problems that are causing you difficulties, however.)

The solutions to this exam are to be typeset with Latex and submitted in both hard copy and electronic copy. **The exam is due Wednesday, October 10, at 9 a.m.** Any code that you use should be included with your write-up, either as an appendix or in-line via the `verbatim` command. (The latter is nicer.) Your code should also be submitted via dropbox in a format that is easy to run (i.e. as m-files.) Code should be tastefully commented and execute without errors.

Your work will be judged both for correctness (i.e. do you have the right answer?) and for style (i.e. do you express that answer efficiently, in full sentences, appropriately type-set, etc.?)

---

**Problem 1**

Let $A$ be the matrix whose singular value decomposition is $A = U \Sigma V^T$, where

$$U = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}, \quad V = \begin{pmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$  

(1) Show that $U$ and $V$ are unitary.

**Solution:**
To show that $U$ is unitary, we check that

$$UU' = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = U' * U.$$

Similarly, we show that $V$ is unitary by showing that $VV' = V'V = I$. We spare the reader the details.

(2) Let $x$ be the vector $(1, 1)^T$. On a single graph, plot vectors representing $x$, $Ax$, $A^2x$, and $A^3x$. What does $A$ seem to be doing? Could you have predicted this behavior from the SVD alone, i.e. without graphing?
Solution:
The Matlab code used to generate the figure is included in an appendix.

The matrix seems to be rotating the vector \( x \) by 15 degrees. We could have predicted this from the SVD because \( A \) takes \( V(:,1) = (\sqrt{3}/2 \, 1/2)^T \) to \( U(:,1) = (1/\sqrt{2} \, 1/\sqrt{2})^T \) with no contraction or expansion \((\sigma_1 = 1)\)–the angle between these vectors is 15 degrees. Ditto for \( V(:,2) \) and \( U(:,2) \).

(3) Let \( B \) represent a matrix with the singular value decomposition \( B = U\Sigma_B V^T \), where \( U \) and \( V \) are as above and

\[
\Sigma_B = \begin{pmatrix} 1 & 0 \\ 0 & 1/2 \end{pmatrix}.
\]

What do you think \( B \) “does?” Try to describe this behavior in words. On a separate graph, plot \( x, Bx, B^2x, \) and \( B^3x \) to verify your description, where again \( x = (1,1)^T \).

Solution:
Again, the Matlab code used to generate the figure is contained in an appendix.

This time, the matrix not only rotates, but slightly contracts (as evidenced by the fact that \( \sigma_2 = 1/2 \).) We see this behavior in the plot.

(4) Now let \( C \) represent the matrix

\[
C = \begin{pmatrix} 0.6830 & -0.1830 \\ 0.6830 & -0.1830 \\ 0.2588 & 0.9659 \end{pmatrix}.
\]
Find the singular value decomposition of $C$. (Use the `svd` command in Matlab. Make sure to enter the numbers accurately.) Using the SVD decomposition, describe what $C$ “does”.

**Solution:**

Issuing the command

$$[U, S, V] = \text{svd}(C)$$

yields

$$U = \begin{pmatrix}
-0.7057 & -0.0442 & -0.7071 \\
-0.7057 & -0.0442 & 0.7071 \\
-0.0625 & 0.9980 & 0.0000
\end{pmatrix}, \quad V = \begin{pmatrix}
-0.9802 & 0.1979 \\
0.1979 & 0.9802
\end{pmatrix}, \quad S = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}$$

The matrix $C$ apparently maps $\mathbb{R}^2$ onto the subspace of $\mathbb{R}^3$ perpendicular to the span of the vector $(-.7071, .7071)$. Since the singular values are all of magnitude 1, $C$ neither expands nor contracts the size of any vector in its domain.

(5) Suppose $y = (-1, 1, 0)$. Is there a vector $x \in \mathbb{R}^2$ such that $Cx = y$? Why or why not? (Hint: this information is contained in the SVD decomposition.)

**Solution:**

The vector $y$ lies in the space perpendicular to the range of $C$. Thus there is no vector $x \in \mathbb{R}^2$ such that $Cx = y$.

(6) Now let $y = (-1, 1, 1)$. Find the pseudo-inverse of $y$ under $C$, i.e. find

$$x^\dagger = C^\dagger y,$$

where $C^\dagger$ is the pseudo-inverse of $C$. (Hint: use the `pinv` command in Matlab.)

**Solution:**

The Matlab command

$$x = \text{pinv}(y)$$

yields the solution

$$x^\dagger = (0.2588, 0.9660)^T.$$
Solution:
The floating point numbers are of the form
\[ x = 2^p \cdot z, \]
where \( z \in [1,2) \). On our machine, \( p \) can be \( \pm 1 \) and \( \pm 0 \). Assuming only the decimal is stored in the mantissa, \( z \) can be \( \pm 1, \pm 1.25, \pm 1.5, \) and \( \pm 1.75 \). Ignoring the numbers corresponding to \( p = -0 \) for a moment, we can store
\[ \pm 0.5, 0.625, 0.75, 0.875, 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 3.5 \]
Note that allowing \( p = -0 \) gives us 8 more numbers we can store. What numbers we choose to store will be machine dependent. It would not be unreasonable to include among these numbers 0, nan, and inf, as Matlab does.

(2) What is the relative error in the machine representation of the real number 3?

Solution:
0 (since 3 is machine representable.)

(3) What is machine \( \epsilon \) for this machine? What is the smallest positive number it can store? Are these two things the same?

Solution:
The gap between 0 and the next largest number is 0.25. The smallest positive number storable is 0.5. These two things are not the same.

(4) Suppose you want to evaluate the function \( f(x) = x^2 \) with this machine for \( x = 3 \). Calculate the forward and backward errors for this problem.

Solution:
Forward error: \( f(3) = 9 \), which is stored as 3.5. So forward error is 5.5.
Backward error: \( \sqrt{3.5} = 1.8078 \), so backward error is \( |3 - 1.8078| = 1.1292 | \).

(5) Suppose you want to calculate \( x \cdot y \) for \( x \) and \( y \) in the range \([1,2]\). What is maximum absolute error of your results? (Remember that the machine must first store the numbers before it can manipulate them.)

Solution:
I solved this problem numerically by writing a script to search through all possible combinations of \( x \) and \( y \), calculating for each combination the value
\[ |xy - r(x) \cdot r(y)|, \]
where \( r \) is the “rounding” function. The code shows that under the assumption that the machine rounds up when a number is directly between two machine representable numbers, the maximum error is 0.6094 and occurs where \( x = y = 1.375 \).

Problem 3
Download “smileyface.mat” from the class webpage to do this problem. The file ‘data.mat’ contains two variables: \( F \) is an unblurred image, and \( G \) is a blurred image.

(1) Suppose the blur is generated by replacing each pixel by the sum of the 5 pixels directly above and below it. (If a given pixel is less than pixels away from top or bottom border, this average simply extends to the pixels within the range of the image. Note too that this is the sum, not the average.) Write down what the blurring kernel would look like. (You can do this with words: how big is it? where is it non-zero? how are the non-zero entries arranged?)
Solution:
The blurring kernel will be a 2500 × 2500 matrix with ones along the diagonal, the first four super diagonals, and the first four subdiagonals.

(2) Write Matlab code to actually form $K$.

Solution:
See the Matlab code.

(3) Once you’ve formed $K$, time how long it takes your computer to compute its SVD. (Suggestion: use the tic and toc commands.)

Solution:
See Matlab code.

(4) Prove formula (9) on page 15 of my notes. (Hint: show that in general you can write

$$AB^T y = \sum_i [B(:,i)^T y] A(:,i),$$

where $y$ is a vector, $A$ and $B$ are matrices of appropriate dimensions, and, e.g., $A(:,i)$ represents the $i$th column of $A$.)

Solution:
Checking the formula

$$AB^T y = \sum_i [B(:,i)^T y] A(:,i),$$

is merely a matter of keeping track of indices (details suppressed.) In order to apply this to the SVD reconstruction $x = V \Sigma^\dagger U^T y$, it suffices to show that the $i$th column of $V \Sigma^\dagger$ is just $v(:,i)/\sigma_i$. This is a straightforward exercise.

(5) Use the same formula (9) to reconstruct the unblurred image. Note that you will need the $U$ and $V$ you calculated when you obtained the SVD of $K$. How many terms should be in the sum? Add some logic to pause the reconstruction every once in a while so you can see how it’s coming along. (Useful Matlab commands: keyboard, pause.) Does the best reconstruction use all the terms, or few than all the terms?

Solution:
See Matlab code. Adding the last term causes weird noise artifacts, but you it seems like you can add all but the last, and the reconstruction gets better the more you add. Note that the norm errors get large if you add the last term.

(6) Now add some noise to the blurred image. (Matlab command: $G = G + \text{randn(size}(G))$.) Reconstruct the image as before, again pausing once in a while to see how you’re doing. How many terms leads to the best reconstruction?

Solution:
See Matlab code. Even with just a little bit of noise, the level of blowup the occurs when you add the last term is extravagant. Still, you can add most of the terms. As you increase the noise level, however, you can add few and fewer terms. The cutoff threshold for singular values seems to depend on the noise level.

(7) The unblurred image $F$ is also included in the .mat file. Redo the last subproblem, only this time, each time you add a term to the reconstruction, calculate $||F - \hat{F}||_F^2$, where $|| \cdot ||_F^2$ refers to the Frobenius norm of a matrix, defined as the square sum of the elements. Make a plot that shows the number of terms included.
in the sum vs. the Frobenius norm of the reconstruction. Does the “best” reconstruction correspond to smallest Frobenius error norm?

**Solution:**
This problem is built into my solutions for the last two parts.

1. **Appendix: MATLAB code**