AMS 526 Homework 2

Due: Wednesday 09/26 in class

1. (15 points) Consider the matrix
$$A = \begin{bmatrix} 2 & 11 \\ 10 & 5 \end{bmatrix}.$$
   (a) (3 points) By hand calculation, determine the singular values of A. (Hint: Compute the eigenvalue of $A^T A$.)
   (b) (6 points) By hand calculation, compute the left singular vectors u_i and their corresponding right singular vectors v_i of A.
   (c) (3 points) Write down the SVD of A^T.
   (d) (3 points) Compute the condition number of A in 2-norm.

2. (15 points) Let $P \in \mathbb{R}^{n\times n}$ be an orthogonal projector.
   (a) (5 points) Show that $I-P$ is also an orthogonal projector. What space does $I-P$ project onto?
   (b) (5 points) Show that $I - 2P$ is orthogonal.
   (c) (5 points) Consider the matrix
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}.$$
   What is the orthogonal projector P onto range(A)? What is its complementary orthogonal projector?

3. (15 points) Each of the following problems describes an algorithm implemented on a computer satisfying the two axioms of floating point numbers (axioms (13.5) and (13.7) in the textbook). For each problem, answer whether the algorithm is *backward stable*, *stable but not backward stable*, or *unstable*. Prove your assertion or give a reasonably convincing argument.
   (a) (5 points) Data: $x, y \in \mathbb{R}^n$. Solution: $x^T y$, computed as $x_1 \otimes y_1 \oplus x_2 \otimes y_2 \oplus \cdots \oplus x_n \otimes y_n$, where $\oplus$ and $\otimes$ denotes floating-point addition and multiplication, respectively.
   (b) (5 points) Data: none. Solution: e, computed by summing $\sum_{k=0}^{\infty} 1/k!$ from left to right using floating-point multiplication $\otimes$ and floating-point addition $\oplus$, stopping when a summand is reached of magnitude $< \epsilon_{\text{machine}}$.
   (c) (5 points) Data: none. Solution: e, computed by the same algorithm as above except that the series is summed from right to left.

4. (15 points) Suppose $U = I - B$ is unit upper triangular, where $B \in \mathbb{R}^{n\times n}$.
   (a) (5 points) Show that
$$U^{-1} = I + B + B^2 + \cdots + B^{n-1}.$$

(b) (5 points) What is the value of $\|U^{-1}\|_F$ if $b_{ij} = 1$ for $i < j$?

(c) (5 points) What is the condition number of U in 1-norm if $b_{ij} = 1$ for $i < j$?

5. (10 points) Gaussian elimination can be used to compute the inverse A^{-1} of a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, though it is rarely really necessary to do so.

   (a) (5 points) Describe an algorithm for computing A^{-1} by solving n systems of equations, and show that its asymptotic operation count is $\frac{8}{3}n^3$ flops.

   (b) (5 points) Describe a variant of your algorithm, taking advantage of sparsity, that reduces the operations count to $2n^3$ flops.

6. (30 points) Write a routine in MATLAB to estimate the condition number of a real matrix A using 1-norm. You will need to compute $\|A\|_1$, which is easy, and estimate $\|A^{-1}\|_1$, which is more challenging. One way to estimate $\|A^{-1}\|_1$ is to take it as the ratio $\|z\|_1/\|y\|_1$, where z is the solution to $Az = y$ and y is picked by some heuristic to maximize the ratio.
   We choose y as the solution to the system $A^T y = c$, where c is a vector each of whose components is ± 1, with the sign for each component chosen by the following heuristic. Using the factorization $PA = LU$ (you may use MATLAB's routine `lu`), the system $A^T y = c$ is solved in two stages, successively solving the triangular systems $U^T v = c$ and $L^T P y = v$. At each step of the first triangular solution, choose the corresponding component of c to be 1 or -1, depending on which will make the resulting component of v larger in magnitude. You will need to write a custom triangular solution routine to implement the selection of c. Then solve the second triangular system $L^T P y = v$ in the usual way for y, for which you can use MATLAB's backslash "\" operator. The idea here is that any ill-conditioning in A will be reflected in U, resulting in a relatively large v. The relative well-conditioning unit triangular matrix L will then preserve this relationship, resulting in relatively large y.

   Test your program on the Hilbert matrix of order $n = 2, 3, \ldots, 12$, which has entries $h_{ij} = 1/(i + j - 1)$. For example, a 3×3 Hilbert matrix has entries

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}.$$

   To check the quality of your estimates, compute A^{-1} explicitly using the LU factorization that was already computed and then compute the condition number $\|A\|_1\|A^{-1}\|_1$. Plot the estimated condition numbers and the explicitly computed condition numbers for the Hilbert matrix of order $n = 2, 3, \ldots, 12$ (using the horizontal axis for n and the vertical axis for the condition numbers). Compare the required flops of the two approaches (i.e., estimation and explicit computation), and also plot their running times for different n. To obtain reliable timing of a procedure, you may need to run it repeatedly for hundreds of iterations and then take the average (use the `tic()` and `toc()` functions outside the iterations).

   Submit your programs, the plots, and a brief discussion of your results.