MA398 Matrix Analysis and Algorithms: Exercise Sheet 6

1. Prove the uniqueness of the singular values in the Singular Value Decomposition (SVD) of a matrix. That is, show that for every matrix $A \in \mathbb{R}^{m\times n}$, the singular values in the SVD are uniquely determined.

   Answer: Here is a sketch of the proof:

   The Singular Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ is given by $A = U\Sigma V^T$, where:

   $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix whose elements σ_i are the singular values of A, and The rows and columns of U and V are the left and right singular vectors of A, respectively. First, observe that if $A = U\Sigma V^T$ is an SVD of A, then $A^T A = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^T\Sigma V^T = V(\Sigma^2)V^T$ is an eigendecomposition of $A^T A$. Here, Σ^2 is a diagonal matrix whose entries are the squares of the singular values of A. The entries of V are the eigenvectors of $A^T A$.

   Now, the eigenvalues of a matrix are uniquely determined, up to order (as can be proved via the characteristic polynomial). This means that the singular values of A, being the square roots of the eigenvalues of $A^T A$, are also uniquely determined, up to order.

   Note, however, that by convention, the singular values in an SVD are always arranged in descending order along the diagonal of Σ. Therefore, the ordering of the singular values is fixed in the SVD, and they are uniquely determined.

2. Implement a function in Python that computes the Singular Value Decomposition (SVD) of a given matrix A. The function should take the matrix A as input and return the matrices U, Σ, and V^T. You can make use of the NumPy library for matrix operations. You can also use compute_svd(A) of the NumPy library to compute the singular values.

3. Consider a 2-dimensional LSQ problem, with:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

   (a) Write a Python function to compute the pseudo-inverse of A, and use it to find the solution $x = A^\dagger b$.

   (b) Using the numpy library, verify that your computed pseudo-inverse is correct and your solution x matches the one obtained from numpy's built-in least squares solver numpy.linalg.lstsq.

   (c) Compute the condition number $\kappa_2(A)$ of matrix A according to the provided definition, and confirm your computation with the function numpy.linalg.cond.

   (d) Generate a small perturbation $\Delta b = \begin{bmatrix} \delta & \delta \end{bmatrix}$ for a range of δ values and compute the corresponding Δx for each. Plot $\frac{|\Delta x|_2}{|x|_2}$ versus $\frac{\kappa_2(A)}{\eta\cos(\theta)} \frac{|\Delta b|_2}{|b|_2}$ to observe the bound provided in the theorem. For this part, you need to compute η and $\cos(\theta)$ based on the given formulas. Note, you may need to be careful about the value of δ to ensure that Δx and x are not zero vectors (which could lead to division by zero in your calculations.

   (e) Write a Python function implementing LSQ_SVD algorithm, and verify it gives the same solution as before for the given A and b. Analyze the complexity of this function based on the operations it performs.

   Hints:

- For the SVD of a matrix, you can use numpy.linalg.svd.
- For matrix multiplication, use the @ operator or numpy.dot.
- For transposing a matrix, use .T attribute.
- For computing the 2-norm of a vector, use numpy.linalg.norm.