Note: This homework is a little different from the previous ones. There are 11 exercises but only 6 will be graded. We will reveal those that will be graded progressively in the announcements page. The idea is that you will at least try to answer each question to prepare for the possibility that you will be asked to answer it. It is to your advantage to proceed as though all questions will be graded and then focus progressively on those that are asked for sure.

1. Show that the Morse-Penrose pseudo-inverse \( X = A^+ \) of an \( m \times n \) matrix \( A \) minimizes the Frobenius norm \( \| AX - I \|_F \) over all \( n \times m \) matrices \( X \). [Hint: Use the fact that the Frobenius norm of a matrix is nothing but the 2-norm of the vector representation of the matrix. Then, you will show that the Pseudo-inverse satisfies the necessary and sufficient condition for a matrix to achieve the minimum of the equivalent least-squares problem \( \min \| AX - I \|_F \).]

2. Consider a nonsymmetric real matrix \( A \) which has the Schur decomposition \( A = QRQ^H \) where \( Q \) is unitary, and \( R \) upper triangular.
   (a) Show that \( q_1 \) the first column of \( Q \) is an eigenvector of \( A \). Are the other columns eigenvectors? Find a Schur decomposition of the matrix \( A_1 = A - \sigma q_1 q_1^H \). What are the eigenvalues of \( A_1 \)?
   (b) Assume that the eigenvalues of \( A \) are such that \( |\lambda_1| > |\lambda_2| > |\lambda_3| \geq |\lambda_4| \geq \cdots |\lambda_n| \)
   where for simplicity it is assumed that the eigenvalues are all real. Suppose you use the power method and have computed the eigenpair \( \lambda_1, u_1 \). Suggest a method for computing the next eigenvalue \( \lambda_2 \) by the power method applied to a certain matrix. You will find that this method will compute a vector \( w_2 \) which is not an eigenvector of \( A \). How can you obtain an eigenvector \( u_2 \) of \( A \) associated with \( \lambda_2 \)? [Hint: Look for \( u_2 \) in the form of a linear combination of \( u_1 \) and \( w_2 \)].

3. Given a matrix \( X \) of size \( m \times n \) of full column rank, find the inertia of a matrix of the form
   \[
   B = \begin{pmatrix}
   D & X \\
   X^T & 0
   \end{pmatrix}
   \]
   where it is assumed that \( D \) is a diagonal matrix (of size \( m \times m \)) with positive diagonal entries. [Hint: you can use block Gaussian Elimination and ideas from the LDLT factorization to reduce \( B \) to block-diagonal form]. Find the inertia when \( D \) is any nonsingular symmetric matrix (with say \( p \) positive eigenvalues and \( m - p \) negative eigenvalues) but \( X \) is an \( m \times m \) matrix (of full rank). Note: you can assume it is diagonal nonsingular with \( p \) positive entries on the diagonal.

4. Apply Gershgorin’s theorem to find a domain where the eigenvalues of \( A \) are located for the following matrices
   \[
   A_1 = \begin{pmatrix}
   1 & -1 & -1 \\
   1 & 2 & 3 \\
   2 & -4 & 1
   \end{pmatrix}
   \quad A_2 = \begin{pmatrix}
   -i & 0 & i \\
   1 & 0 & 1 \\
   0 & 1 + i & i
   \end{pmatrix}
   \quad A_3 = \begin{pmatrix}
   1 & i & -i \\
   -i & 2 & 0 \\
   i & 0 & 3
   \end{pmatrix}
   \]
   The region (complex or real) you find should be the smallest possible that can be determined by using (the row version of) the Gershgorin theorem.
5. The subspace iteration [called simultaneous iteration in the Trefethen-Bau text] is a generalization of the power method. Read relevant parts in Lecture 28 of the text. Then write a “subspace iteration” script in matlab. Your function should take a matrix \( A \) an initial block \( X_0 \), a tolerance for stopping (\( tol \)), and a maximum number of steps (\( maxits \)). Use the matlab supplied QR factorization function \([Q,R] = \text{qr}(X,0)\). Note also that Algorithm (7.3.6) in the text computes a basis \( Q_k \) of an approximate invariant subspace. The eigenvalues are extracted from this subspace as the eigenvalues of \( Q_k^H A Q_k \) as shown in (7.3.3). This step does not have to be done at each iteration. In fact it can be skipped to the end as is asked here. Apply the algorithm to compute the two eigenvalues of largest modulus of the matrix \( \text{Mark}(10) \) [see the class matlab page for the \text{Mark} script]. Stop the iteration whenever 
\[
\|AQ - QT\|_1 \leq 10^{-5}
\]
where \( T = Q^H A Q \). You need not show the intermediate results. Just show the eigenvalues obtained and the number of iterations required to converge.

6. Consider the matrix
\[
A = \begin{pmatrix}
1 & 2 & 3 \\
2 & 1 & 0 \\
3 & 0 & 0
\end{pmatrix}
\]
Determine the \( c,s \) pair of the Jacobi rotation needed to annihilate the entry (1,3) (and (3,1)) of the matrix. Then show the result of applying the transformation.

7. What is the cost of using the QR algorithm for computing the eigenvalues (only) of a tridiagonal matrix \( T \)? [You can make the assumption that the QR-iteration algorithm with Wilkinson shifts requires about 2 steps per eigenvalue]. What about the cost of computing both the eigenvalues and eigenvectors?

8. We are given a square matrix \( A \) that has the SVD \( A = U \Sigma V^T \). Let \( U = [u_1, u_2, \ldots, u_n] \), \( V = [v_1, \ldots, v_n] \) and \( \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n) \). Prove that the \( 2n \) eigenvalues of the matrix:
\[
H = \begin{pmatrix}
0 & A^T \\
A & 0
\end{pmatrix}
\]
are \( \pm \sigma_i \) with the corresponding unit eigenvectors \( \frac{1}{\sqrt{2}} \left[ v_i \pm u_i \right] \). Extend to the general case of a rectangular matrix \( A \).

9. This is about the QRSVD algorithm. For an \( m \times n \) matrix \( A \) (assume \( m > n \)) the algorithm is as follows:
\[
A = Q_0 R_0 \quad \text{// QR factorization of } A
\]
For \( k = 0,1,2,\ldots \), Do:
\[
R_k^T = Q_{k+1} R_{k+1} \quad \text{// QR factorization of } R_k^T
\]
EndDo

[In the QR factorizations it is assumed that diagonal entries or \( R \) are positive]. If \( A = U \Sigma V^T \) is the ‘thin’ SVD of \( A \), then it is observed that the product \( Q_1 Q_3 \cdots Q_{2k+1} \) converges to \( V \), \( Q_0 Q_2 Q_4 \cdots Q_{2k} \) converges to \( U \), and \( R_{2k} \) converges to \( \Sigma \).
(a) Try the algorithm on a random 6 \( \times \) 3 matrix.
(b) Show that \( Q_1 (R_1 R_0) \) is the QR decomposition of \( R_0^T R_0 \) (\( Q_1 \) is the Q factor, \( R_1 R_0 \) the R
factor)
(c) Show that \((R_1 R_0) Q_1 = R_1 R_1^T\) (Hint: focus first on the middle matrix \((R_0)\) on the left hand side) and then that \(R_1 R_1^T = R_2^T R_2\). Going from \(A_0 = R_0^T R_0\) to \(A_2 = R_2^T R_2\) represents a step of the QR algorithm. Explain.
(d) More generally consider the matrices \(A_k = R_k^T R_k\) for \(k = 0, 2, 4, \cdots\). Show that these are the iterates of the QR algorithm applied to the matrix \(A_0 = R_0^T R_0\).

10. (You may find it useful to first read beginning of Lecture 30 of text. The notes also provide enough information). Write a matlab script \([s, c] = \text{jacrot}(A, p, q)\) to calculate the rotation needed to annihilate the term in position \((p, q)\) of a matrix \(A\) by the Jacobi algorithm. Make sure to avoid divisions by zero. The script should work in *all* cases. Apply your script to determine the \(c, s\) pair of the Jacobi rotation needed to annihilate the entry \((1,3)\) (and \((3,1)\)) of the \(3 \times 3\) matrix seen in Exercise ??.

11. Write a script of the form \(\text{function} \ [Q, B] = \text{jacmeth}(A, \text{nit, tol})\) that implements the Jacobi algorithm. Here \(\text{nit}\) is the max number of iterations (whole sweeps) allowed, and \(\text{tol}\) is the tolerance for stopping. Your algorithm will stop when either the maximum number of outer sweeps is reached or when \(\|B - \text{diag}(B)\|_F \leq \|A\|_F \ast \text{tol}\) where \(B\) is the matrix at the current step. You need to implement the trick of computing these frobenius norms by the successive updates which requires with very little arithmetic. The returned matrix \(Q\) should be the matrix of eigenvectors.

You will find in the class web-site a data set called \text{tenxten} containing a symmetric matrix of dimension 10. Apply your script to compute the eigenvalues and eigenvectors of \(A\). Use a tolerance of 1.e-10 and a max. number of iterations of 10.