This assignment is due on Thursday, May 11. It should be uploaded to Canvas at the Homework 4 submission.

For this assignment, you will write a MATLAB program that will compute the eigenvalues and eigenvectors of a given symmetric matrix. That is, it must compute the Schur decomposition

\[ A = QDQ^T \]

where \( Q \) is a real orthogonal matrix whose columns are the normalized eigenvectors of \( A \), and \( D \) is a diagonal matrix whose diagonal entries are the eigenvalues of \( A \). Your program must consist of the following functions:

- **function \([c,s]=\text{givens}(a,b)\)**
  
  This function can be downloaded from the Assignments page of the course web site. It computes \( c \) and \( s \) such that
  
  \[
  \begin{bmatrix}
    c & -s \\
    s & c
  \end{bmatrix}
  \begin{bmatrix}
    a \\
    b
  \end{bmatrix}
  =
  \begin{bmatrix}
    r \\
    0
  \end{bmatrix}, \quad r = \sqrt{a^2 + b^2}.
  \]

- **function \(v=\text{house}(x)\)**
  
  This function can be downloaded from the Assignments page of the course web site. It computes a vector \( v \) such that
  
  \[
  Px = (I - cvv^T)x = \alpha e_1, \quad \alpha = \pm\|x\|_2, \quad c = \frac{2}{v^Tv}.
  \]

- **function \([T,U]=\text{tridiag}(A)\)**
  
  This function uses Householder reflections (which must be computed using\ house\) to compute an orthogonal matrix \( U \) such that \( U^TAU = T \), where \( T \) is tridiagonal. Note that \( U \) can only reduce a general matrix to upper Hessenberg form, but because \( A \) is assumed to be symmetric, \( U^TAU \) will automatically turn out to be tridiagonal. It is recommended that you examine the function \text{houseqr}, which can be downloaded from the assignment in Canvas, or the Assignments page of the course web site, as this function will be similar.

- **function \([T,G]=\text{qrstep}(A)\)**
  
  This function uses Givens rotations (which must be computed using\ givens\) to perform the operations
  
  \[
  A - \mu I = GR, \quad T = RG + \mu I
  \]

  that constitute a single iteration of the QR algorithm. The first step is the QR factorization of \( A - \mu I \), where the shift \( \mu \) is the Wilkinson shift, for which the formula can be found in the Lecture 16 Notes, or on page 420 of Golub and van Loan (to be distributed in class). Multiplication of \( A \) on the left by \( G^T \) to perform the QR factorization of \( A \), or multiplication of \( R \) on the right by \( G \) to complete the orthogonal similarity transformation \( T = G^T A G \), must be performed by Givens row rotations (for left multiplication by \( G^T \)) or Givens column rotations (for right multiplication by \( G \)). The matrix \( G \) must be computed as an accumulation of Givens column rotations before it is returned.
\begin{itemize}
  \item \textbf{function} \[D,Q\]=symqr(T)

  This function is the main function for the symmetric QR algorithm applied to the symmetric tridigonal matrix \(T\). It must proceed as in Algorithm 8.3.3 on page 421 of Golub and van Loan (to be distributed in class), except for the first step, the tridiagonalization, which will have already been performed by \textbf{tridiag}.

  In each iteration of a \textbf{while} loop, this function must find indices \(p\) and \(q\) such that (1) \(T(q+1 : n, q+1 : n)\) is diagonal, and (2) \(T(p+1 : n-q, p+1 : n-q)\) is unreduced, meaning all of its off-diagonal elements are nonzero. Furthermore, it must be a \textit{maximal} unreduced block, meaning that either \(q = 0\) or \(t_{n-q+1,n-q} = 0\), and either \(p = 0\) or \(t_{p+1,p} = 0\). In other words, \(T(p+1 : n-q, p+1 : n-q)\) must be \textit{decoupled} from other diagonal blocks of \(T\), thus leading to a block upper-triangular structure, to ensure that the eigenvalues of \(T(p+1 : n-q, p+1 : n-q)\) are also eigenvalues of \(A\).

  Instead of checking if a subdiagonal entry is \textit{exactly} zero, which rarely occurs in practice, we “declare” \(t_{i+1,i}\) to be zero if

\[
|t_{i+1,i}| \leq TOL(|t_{i,i}| + |t_{i+1,i+1}|), \quad TOL = 10^{-10}.
\]

  If all subdiagonal entries are sufficiently small, then the loop exits, and \(T\), that has been reduced to diagonal form, is returned as \(D\).

  Otherwise, once a suitable, unreduced diagonal block of \(T\) is chosen, then \textbf{qrstep} must be applied to it. The matrix \(G\) returned by \textbf{qrstep} must then be applied to the appropriate \textit{columns} of \(Q\) (which should initially be \(I\)) in order to correctly compute the orthogonal matrix \(Q\) such that \(Q^T TQ\) is diagonal, to within the specified tolerance.

  \item \textbf{function} \[D,Q\]=symeig(A)

  This is the main function of the entire program. Its job is very simple: to call \textbf{tridiag} to tridiagonalize \(A\), yielding a tridiagonal matrix \(T\), and then call \textbf{symqr} to compute a diagonal matrix \(D\) whose diagonal entries are the eigenvalues of \(T\) (and thus \(A\) as well). The similarity transformations used to reduce \(A\) to \(T\), and \(T\) to \(D\), must be properly accumulated in order to obtain an orthogonal matrix \(Q\) such that \(Q^T A Q = D\).

  Test your code on a randomly generated symmetric matrix. It is recommended that you have your code print out intermediate results, such as the tridiagonal matrix \(T\) in \textbf{symqr} after each application of \textbf{qrstep} to an unreduced diagonal block.

  In addition to your code, submit the answers to the following questions:

  1. Explain what changes would have to be made to the code to make it compute the real Schur form \(A = QTQ^T\) of a general (unsymmetric) matrix, where \(Q\) is a real orthogonal matrix and \(T\) is a \textit{block} upper-triangular matrix, with \(1 \times 1\) or \(2 \times 2\) diagonal blocks, depending on whether the eigenvalues are real or complex.

  2. In what ways can your algorithm, or its implementation, be made more efficient, based on the structure of the matrices involved?

  3. How would the code change if only the eigenvalues were required, and not the eigenvectors? How can you make the code flexible so that the commands \(D=symeig(A)\) and \([D,Q]=symeig(A)\) both work as intended, and the first command does not perform any unnecessary computations? \textit{Hint:} consult the MATLAB documentation for the function \textbf{nargout}.
\end{itemize}