Part A.

1. Using exact arithmetic, reduce the symmetric matrix

\[
A = \begin{bmatrix}
4 & -1 & -1 & 0 \\
-1 & 4 & 0 & -1 \\
-1 & 0 & 4 & -1 \\
0 & -1 & -1 & 4 \\
\end{bmatrix}
\]

to tridiagonal form using Householder similarity transformations. Be as efficient as possible.

2. (a) Write a simple Matlab script to perform the basic QR iteration (no shifts) based on repeated use of the built-in Matlab function \([Q,R] = qr(A)\). Use first the following stopping criterion:

\[
\max_{i \neq j} |a^{(k)}_{ij}| < 10^{-4}
\]

where \(a^{(k)}_{ij}\) denotes the \((i,j)\) entry of \(A_k\) (= the \(k\)th iterate in the QR process). Use the diagonal entries of the last iterate \(A_k\) as approximate eigenvalues of \(A\). Apply the algorithm to the symmetric matrix

\[
A = \begin{bmatrix}
4 & 3 & 2 & 1 \\
3 & 4 & 3 & 2 \\
2 & 3 & 4 & 3 \\
1 & 2 & 3 & 4 \\
\end{bmatrix}
\]

do not worry about preliminary reduction to tridiagonal form). Print out the intermediate \(A_k\) matrices and observe the decrease in the off-diagonal entries.

(b) Compute the eigenvalues of \(A\) using the \texttt{eig} function and take these to be the “exact” eigenvalues of \(A\) (they are, up to machine precision). How many accurate digits do the approximate eigenvalues computed by your QR algorithm have? You should use the “long” format.

(c) Determine (by experiment) how many iterations of the QR process are required to achieve \(t\) accurate digits in all of the eigenvalues, where \(t = 1, 2, \ldots, 8\). What is the rate of convergence: linear, superlinear, quadratic ...? Draw a plot with \(t\) displaying the number of iterations vs. \(t\).

Part B.

Here we consider approximations to the eigenvalues and eigenfunctions of the one-dimensional Laplace operator \(L[u] := -\frac{d^2u}{dx^2}\) on the unit interval \([0, 1]\) with boundary conditions \(u(0) = u(1) = 0\).

A scalar \(\lambda\) is said to be an eigenvalue of \(L\) (with homogeneous Dirichlet boundary conditions) if there exists a twice-differentiable function \(u: [0, 1] \rightarrow \mathbb{R}\), not identically zero in \([0, 1]\), such that

\[
-u''(x) = \lambda u(x) \quad \text{on } [0, 1] \quad \text{with } u(0) = u(1) = 0.
\]  

(1)

In this case \(u\) is said to be an eigenfunction of \(L\) corresponding to the eigenvalue \(\lambda\). Obviously, eigenfunctions are defined up to a nonzero scalar multiple.

The eigenvalues and eigenfunctions of \(L\) are easily found to be \(\lambda_j = j^2 \pi^2\) and \(u_j(x) = \alpha \sin(j \pi x)\) for any nonzero constant \(\alpha\), which we can take to be 1. Here \(j\) is a positive integer; hence, the operator \(L\) has an infinite set of (mutually orthogonal) eigenfunctions \(\{u_j\}_{j=1}^\infty\) corresponding to the discrete spectrum of eigenvalues \(\{\lambda_j\}_{j=1}^\infty\). Note that \(0 < \lambda_1 < \lambda_2 < \cdots < \lambda_j \rightarrow \infty\) as \(j \rightarrow \infty\). Also, each eigenvalue is simple in the sense that (up to a scalar multiple) there is a unique eigenfunction corresponding to it.

Approximations to the eigenvalues and eigenfunctions can be obtained by discretizing the interval \([0, 1]\) by means of \(N + 2\) evenly spaced points: \(x_i = ih\) where \(i = 0, 1, \ldots, N + 1\) and \(h = 1/(N + 1)\). The second derivative operator can then be approximated by centered finite differences:

\[
-\frac{d^2u}{dx^2}(x_i) \approx \frac{-u(x_{i-1}) + 2u(x_i) - u(x_{i+1})}{h^2}
\]
and therefore the continuous (differential) eigenproblem (1) can be approximated by the discrete (algebraic) eigenvalue problem

\[ h^{-2} T_N u = \lambda u \]  

(2)

where we have set

\[ T_N = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2 \end{bmatrix} \quad \text{and} \quad u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} \]

with \( u_i := u(x_i) \). It can be shown that the \( N \times N \) matrix \( T_N \) has eigenvalues \( \mu_j = 2(1 - \cos \frac{\pi j}{N + 1}) \) for \( j = 1 : N \), corresponding to the eigenvectors \( u_j \), where \( u_j(k) = \sqrt{\frac{2}{N+1}} \sin \left( \frac{k\pi}{N+1} \right) \) is the \( k \)th entry in \( u_j \).

Notice that the eigenvectors \( u_j \) are normalized with respect to the 2-norm: \( u_j^T u_j = 1 \). Also notice that the eigenvalues of \( T_N \) lie in the interval \((0, 4)\). Hence, the eigenvalues of \( h^{-2} T_N \) lie in the interval \((0, 4(N+1)^2)\).

1. Show that for small \( j \) and large \( N \) (small \( h \)) the eigenvalues of \( h^{-2} T_N \), given by \( 2h^{-2}(1 - \cos \frac{\pi j}{N+1}) \), are good approximations of the smallest eigenvalues of the differential operator \( L \). (Hint: use Taylor expansion of the eigenvalues of \( T_N \) for small values of \( h \), and show that for small \( j \) the approximation error goes to 0 like \( h^2 \).) What can be said about the largest eigenvalues?

2. Investigate how the eigenvectors \( u_j \) of \( T_N \) are related to the eigenfunctions of \( L \).

3. Express the spectral condition number \( \kappa = \lambda_N/\lambda_1 \) of \( T_N \) (which is the same as the condition number of \( h^{-2} T_N \)) as a simple function of \( N \) for \( N \rightarrow \infty \).

4. Use Matlab to plot the eigenvalues of \( T_N \) for \( N = 21 \) (from smallest to largest).

5. Again using \( N = 21 \), use Matlab to plot the eigenvectors \( u_j \) of \( T_N \) for \( j = 1, 2, 3, 5, 11, 21 \). The plot should be of the form \((k, u_j(k)) \) where \( k = 1 \) : \( 21 \). Note that as \( j \) increases, the behavior of the eigenvectors (approximate eigenfunctions) becomes increasingly oscillatory; these eigenfunctions are often called “high energy modes”.

6. Write Matlab code implementing the power method to compute approximations of the largest eigenvalue \( \lambda_N \) of \( L \) and corresponding eigenfunction. Use \( N = 500 \) and apply the algorithm to \( h^{-2} T_N \). Implement the matrix–vector products as efficiently as possible: do not store \( h^{-2} T_N \) as a full \( N \times N \) matrix! Start with a random initial guess, use normalization in the 2-norm, and stop when the difference of two subsequent approximate eigenvectors has 2-norm less than some prescribed tolerance \( \tau \) (e.g., \( \tau = 10^{-6} \) or smaller if the approximation is not adequate). Report the number of iterations performed and compare the computed approximations with the exact eigenvalue/eigenvector pair. Use Matlab to plot the computed eigenvector.

7. Write Matlab code implementing the inverse power method to approximate \( \lambda_1 = \pi^2 \) (the smallest eigenvalue of \( L \)) and the corresponding eigenfunction. Use \( N = 500 \) and apply the algorithm to \( h^{-2} T_N \). To solve linear systems with \( h^{-2} T_N \) use the tridiagonal solver developed for the first homework assignment. Do not store \( h^{-2} T_N \) as a full \( N \times N \) matrix. Same initial guess, normalization, and stopping criterion as in 6. Report the number of iterations performed and compare the computed approximations with the exact eigenvalue/eigenvector pair. Use Matlab to plot the computed eigenvector.

8. Write Matlab code implementing inverse iteration to compute an approximation to the fifth eigenfunction of \( L \), i.e., the one corresponding to \( \lambda_5 \). You should use a decent approximation \( \lambda_5^{(0)} = \lambda_5 \), for example \( \lambda_5^{(0)} = 25\pi^2 \) where \( \pi \) is the Matlab approximation (\( \pi \)) to the true value of \( \pi \). Start from a random initial guess and iterate until the residual

\[ r^{(k)} = (h^{-2} T_N - \lambda_5^{(k)} I_N) u^{(k)} \]

satisfies

\[ \|r^{(k)}\|_\infty \leq 10^{-12} h^{-2} T_N \|_\infty \].

2
How many iterations are required to obtain an accurate approximate eigenvector? Report the number of iterations performed and compare the computed approximations with the exact eigenvector. Use Matlab to plot the computed eigenvector.