CS 475/675 - Spring 2015: Assignment 3

Due July 8, 2015, 5:00pm.
Written/analytical work should be submitted to the physical assignment box on 4th floor MC (Q1-4).
MATLAB code and any associated output and commentary should be submitted to the DropBox on LEARN (Q5G and Q6).
CS675 students should do all questions; CS475 need only do Q1-4, and Q6.
For full marks be sure to show all your work!

1. (5 marks) Let v be a given non-zero column vector.

   (a) Show algebraically that the orthogonal projection matrix $P = I - \frac{vv^T}{v^Tv}$ is idempotent, i.e. $PP = P$. (This property ensures that applying such a projection matrix repeatedly has no additional effect after the first application.)

   (b) Show algebraically that the Householder matrix $F = I - 2\frac{vv^T}{v^Tv}$ is indeed an orthogonal matrix, i.e. $F^T = F^{-1}$.

2. (10 marks) Adapt the Householder QR algorithm so that it computes the factorization $A = QL$ where L is lower triangular and Q is orthogonal. Assume that A is square and full-rank. Give a text description of your algorithm supported by illustrations and pseudocode. (Hint: modify the Householder vector so that $(I - 2vv^T/v^Tv)x$ is zero everywhere but its *bottommost* component, rather than its topmost.)

3. (5 marks) Use Householder transformations to perform a QR factorization of the following matrix by hand.

$$A = \begin{bmatrix} 2 & -1 & 12 \\ 2 & 4 & -6 \\ 2 & 4 & 6 \\ 2 & -1 & 0 \end{bmatrix} \tag{1}$$

   For simplicity, always use the "positive" reflector given by using the vector $v = x - \|x\|e_1$ (so that the result is $Fx = \|x\|e_1$). Show your work, and give the resulting factors.

4. (15 marks) Let A be a symmetric tridiagonal matrix.

   (a) In the QR factorization of $A = QR$, which entries of R are in general nonzero? Which entries of Q? Explain your answer.

   (b) Show that the tridiagonal structure is recovered when the product RQ is formed. (Hint: Show that (i) RQ is upper Hessenberg, and (ii) RQ is symmetric.)

   (c) Explain how the 2×2 Householder transformation can be used in the computation of the QR factorization of a tridiagonal matrix. Estimate the complexity of your resulting algorithm.

5G. [CS675 students only] (10 marks)

   (a) Implement the QL factorization method you derived in Q2. Create a MATLAB function:

```
[Q,L] = QL_Factor(A)
```

   You may assume the input is a full-rank square matrix A. The outputs are the Q and L factors of A. Submit your code for `QL_Factor.m`.

(b) Consider the problem of finding an $A = RQ$ factorization of a matrix A, where R is upper triangular and Q is an orthogonal matrix. (Note: these factors *will* be different from the usual $A = QR$ factors.) Create another MATLAB function:

```
[R,Q] = RQ_Factor(A)
```

The input is again a full-rank square matrix A. The outputs are the R and Q factors. Rather than design a new algorithm to form the factors from scratch, your function should reuse `QL_Factor` from part (a) in constructing R and Q. (Hint: Think about the QL factorization of A^T.) Submit your code for `RQ_Factor.m`

6. (30 marks) Consider the $n \times n$ tridiagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

Let $v^{(k)}$ be the k-th eigenvector of A and $\lambda^{(k)}$ the corresponding eigenvalue. Then they are given by:

$$v_j^{(k)} = \sin\left(\frac{kj\pi}{n+1}\right) \qquad \text{and} \qquad \lambda^{(k)} = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right),$$

where $v_j^{(k)}$ is the j-th component of $v^{(k)}$. Note: $\lambda^{(1)} < \lambda^{(2)} < \cdots < \lambda^{(n)}$.

(a) Implement the following numerical methods: power iteration, Rayleigh quotient iteration, and QR algorithm (no shift). Create the following MATLAB functions:

```
[v,lambda,iter] = PowerIteration(A,v0,maxiter,tol)
[v,lambda,iter] = RayleighQuotient(A,v0,maxiter,tol)
[V,Lambda,iter] = QRIteration(A,maxiter,tol)
```

The first two MATLAB functions take as inputs the matrix A, the initial vector $v0$, the maximum number of iterations $maxiter$ and the tolerance tol, and compute the approximate eigenvector v, approximate eigenvalue $lambda$, and the number of iterations to convergence, $iter$. The third MATLAB function computes all the eigenvectors and eigenvalues of A. The approximate eigenvectors are stored in matrix V and eigenvalues are stored in vector $Lambda$.

For `RayleighQuotient`, you can use MATLAB backslash \ to solve linear systems. For `QRIteration`, use MATLAB's built-in `qr` to to compute the QR factorization of A.

For all these methods, the stopping criterion is:

$$\|A\tilde{v} - \tilde{\lambda}\tilde{v}\|_2 < tol,$$

where $\tilde{v}$ and $\tilde{\lambda}$ are the approximate eigenvector and eigenvalue, respectively. For `QRIteration`, the stopping criterion applies to *all* eigenvectors and eigenvalues. Submit all your code.

(b) Compute the eigenvectors and eigenvalues of A using the above methods for $n = 100$. Set the maximum iteration number, $maxiter = 10000$, and the tolerance, $tol = 10^{-4}$. Create a MATLAB program, `EigenMethods.m`, which performs the following:

(i) Use `PowerIteration` to compute the largest eigenvector and eigenvalue of A. The initial vector $v0 = [1, 0, \ldots, 0]^T$.

(ii) Use `RayleighQuotient` to compute an eigenvector and eigenvalue of A. The initial vector $v0 = [1, 1, \ldots, 1]^T$.

(iii) Use `QRIteration` to compute all eigenvectors and eigenvalues of A.

For `PowerIteration`, make a plot of the computed eigenvector. Display the value of the computed eigenvalue and the number of iterations on the title of the plot. Do the same for `RayleighQuotient`. For `QRIteration`, make a plot of all eigenvalues. Also, plot the column vectors v_{20}, v_{40}, v_{60}, v_{80} of V. The title of the plots are the corresponding eigenvalues and number of iterations. Submit `EigenMethods` and the outputs.