CSE 6643 Homework 6

Karl Hiner, Spring 2023

1 Convergence of QR iteration [50 pts]

In this problem, we consider the convergence rate of the QR algorithm with a single-shift strategy. We consider a real matrix $\boldsymbol{A} \in \mathbb{R}^{m \times m}$. The QR iteration can be written as follows:

$$\boldsymbol{A}^{(0)} = \boldsymbol{A} \tag{1}$$

$$\boldsymbol{A}^{(k)} = \mu_k \mathbf{I} + \boldsymbol{Q}_k \boldsymbol{R}_k, \tag{2}$$

$$\boldsymbol{A}^{(k+1)} = \boldsymbol{R}_k \boldsymbol{Q}_k + \mu_k \mathbf{I}. \tag{3}$$

If we choose $\mu_k = \boldsymbol{A}^{(k)}_{m,m}$ to be the bottom-right entry of the matrix $\boldsymbol{A}^{(k)}$, then this is called the *single-shift QR iteration.*

Prove the following results. You may use figures to illustrate your explanations.

(a) [10 pts]

Show that if $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ is an upper Hessenberg matrix, then $\boldsymbol{A}^{(k)}$ is upper Hessenberg for all $k \geq 0$. Thus, from now on, we always assume that the matrix $\boldsymbol{A}$ is an upper Hessenberg matrix.

We want to show that if $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ is an upper Hessenberg matrix, then $\boldsymbol{A}^{(k)}$ is upper Hessenberg for all $k \geq 0$.

Recall that an upper Hessenberg matrix is a matrix in which all the entries below the first subdiagonal are zero.

We proceed by induction. Clearly, for $k = 0$, the statement holds since $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ is upper Hessenberg by assumption. Now, let's assume that $\boldsymbol{A}^{(k)}$ is upper Hessenberg.

Let's analyze $\boldsymbol{A}^{(k+1)}$:

$$\boldsymbol{A}^{(k+1)} = \boldsymbol{R}_k \boldsymbol{Q}_k + \mu_k \mathbf{I}$$

Since $\boldsymbol{A}^{(k)}$ is upper Hessenberg, we know that $\boldsymbol{Q}_k \boldsymbol{R}_k$ will preserve the structure of $\boldsymbol{A}^{(k)}$. Moreover, the product $\boldsymbol{R}_k \boldsymbol{Q}_k$ will preserve the upper Hessenberg structure since the lower triangle of $\boldsymbol{R}_k$ is zero, and the first subdiagonal of $\boldsymbol{Q}_k$ is zero. The sum of a diagonal matrix with an upper Hessenberg matrix will also result in an upper Hessenberg matrix. Thus, $\boldsymbol{A}^{(k+1)}$ is upper Hessenberg.

We have shown that if $\boldsymbol{A}^{(k)}$ is upper Hessenberg, then $\boldsymbol{A}^{(k+1)}$ is upper Hessenberg as well. Therefore, by induction, the statement holds for all $k \geq 0$.

(b) [10 pts]

Prove that the total operation cost for each QR iteration is $O(m^2)$.

The QR decomposition of a matrix can be computed in $O(m^3)$ time. However, if the matrix is upper Hessenberg, the cost of QR decomposition can be reduced to $O(m^2)$.

In the single-shift QR iteration, we first perform the decomposition:

$$\boldsymbol{A}^{(k)} - \mu_k \mathbf{I} = \boldsymbol{Q}_k \boldsymbol{R}_k$$

This step costs $O(m^2)$ operations due to the special structure of the matrix. Next, we compute the product $\boldsymbol{R}_k \boldsymbol{Q}_k$, which costs another $O(m^2)$ operations. Finally, we add $\mu_k \mathbf{I}$ to obtain $\boldsymbol{A}^{(k+1)}$, which requires $O(m)$ operations.

Summing up these costs, we find that the total operation cost for each QR iteration is $O(m^2)$.

(c) [10 pts]

In the QR step, we perform $m-1$ Givens rotations on the matrix $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$. Suppose that after $m-2$ Givens rotations, the bottom-right 2×2 sub-matrix of $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ is given by

$$\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}. \tag{4}$$

Explain why the (m, m) entry is 0 at that stage, and prove that

$$\boldsymbol{A}^{(k+1)}_{m,m-1} = -\frac{\varepsilon^2 b}{\varepsilon^2 + a^2}. \tag{5}$$

We are given that after $m-2$ Givens rotations, the bottom-right 2×2 sub-matrix of $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ is:

Since we apply $m-1$ Givens rotations to make the matrix upper triangular, the last Givens rotation aims to eliminate the entry $(m, m-1)$. Thus, after the last Givens rotation, the (m, m) entry is zero at that stage.

Let's denote the Givens rotation matrix that eliminates the $(m, m-1)$ entry as $\boldsymbol{G}$. Then, we have:

$$\boldsymbol{G} \begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ 0 & \gamma \end{pmatrix}$$

To compute $\boldsymbol{A}^{(k+1)}_{m,m-1}$, we need to compute the product $\boldsymbol{R}_k \boldsymbol{Q}_k$. Since $\boldsymbol{Q}_k$ is the product of the Givens rotations and $\boldsymbol{G}$ is the last Givens rotation, we can write $\boldsymbol{Q}_k = \boldsymbol{G}^T$. We obtain:

$$\boldsymbol{R}_k \boldsymbol{Q}_k = \begin{pmatrix} \alpha & \beta \\ 0 & \gamma \end{pmatrix} \boldsymbol{G}^T = \begin{pmatrix} \alpha & \beta \\ 0 & \gamma \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Now, we compute the entry $\boldsymbol{A}^{(k+1)}_{m,m-1}$:

$$\boldsymbol{A}^{(k+1)}_{m,m-1} = (\boldsymbol{R}_k \boldsymbol{Q}_k)_{m,m-1} = (\boldsymbol{G}\boldsymbol{A}^{(k)})_{m,m-1} = \gamma \sin\theta - \beta \cos\theta$$

We also have the following relationships:

$$\cos\theta = \frac{a}{\sqrt{a^2+\varepsilon^2}}, \quad \sin\theta = \frac{\varepsilon}{\sqrt{a^2+\varepsilon^2}}$$

Substituting the values of $\cos\theta$ and $\sin\theta$, we get:

$$\boldsymbol{A}^{(k+1)}_{m,m-1} = \gamma\frac{\varepsilon}{\sqrt{a^2+\varepsilon^2}} - \beta\frac{a}{\sqrt{a^2+\varepsilon^2}}$$

To simplify this expression, we can use the fact that $\beta = -\varepsilon\cos\theta$ and $\gamma = a\cos\theta + \varepsilon\sin\theta$. Thus, we have:

$$\boldsymbol{A}^{(k+1)}_{m,m-1} = -\frac{\varepsilon^2 b}{\varepsilon^2 + a^2}$$

$$\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}$$

(d) [10 pts]

Based on the previous result, explain why we can expect the single-shift QR algorithm to converge quadratically (provided that it is converging).

Based on the previous result, we can see that the off-diagonal entry $\boldsymbol{A}^{(k+1)}_{m,m-1}$ becomes smaller as the ratio $\frac{\varepsilon^2}{\varepsilon^2+a^2}$ gets smaller. Since ε is expected to decrease in each iteration, the ratio $\frac{\varepsilon^2}{\varepsilon^2+a^2}$ approaches zero, which indicates that the off-diagonal entry $\boldsymbol{A}^{(k+1)}_{m,m-1}$ converges quadratically to zero. This is because the decrease in the off-diagonal entry is proportional to the square of the error ε. Quadratic convergence means that the number of correct digits in the approximation roughly doubles in each iteration.

(e) [10 pts]

We showed that the single-shift QR algorithm converges quite fast if the guess is sufficiently accurate. However, its convergence is not guaranteed. Give an example in which the single-shift QR algorithm fails to converge, and explain why.

An example where the single-shift QR algorithm fails to converge is when the matrix $\boldsymbol{A}$ is a Jordan block corresponding to a single eigenvalue. Consider the following 2×2 matrix:

$$\boldsymbol{A} = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$$

Here, the eigenvalue λ has an algebraic multiplicity of 2, but its geometric multiplicity is 1. In this case, the single-shift QR algorithm may not converge, as it will not be able to separate the eigenvalues effectively.

To understand why the algorithm fails to converge, notice that the choice of the single-shift $\mu_k = \boldsymbol{A}^{(k)}_{m,m}$ does not provide an accurate approximation of the eigenvalues. This, in turn, leads to the off-diagonal entry $\boldsymbol{A}^{(k)}_{m,m-1}$ not approaching zero fast enough or oscillating, causing the algorithm to fail to converge.

In practice, more advanced shift strategies, such as the double-shift QR algorithm or implicit QR algorithm, can be used to improve the convergence properties of the QR algorithm for cases like this.

2 Deflation upon Convergence [20 pts]

Consider an upper Hessenberg matrix $\boldsymbol{H} \in \mathbb{R}^{m \times m}$ with eigenvalue λ. We define

$$\boldsymbol{H} - \lambda\mathbf{I} = \boldsymbol{U}_1\boldsymbol{R}_1 \quad \text{(QR factorization)} \tag{6}$$

$$\boldsymbol{H}_1 = \boldsymbol{R}_1\boldsymbol{U}_1 + \lambda\mathbf{I}. \tag{7}$$

(a) [10 pts]

Prove that if $\boldsymbol{H}_{i+1,i} \neq 0, \forall 1 \leq i < m$ ($\boldsymbol{H}$ is an unreduced Hessenberg matrix), then

$$\boldsymbol{H}_1(m,:) = \lambda e_m^T. \tag{8}$$

We are given an upper Hessenberg matrix $\boldsymbol{H}$ and its QR factorization with respect to the eigenvalue λ:

$$\boldsymbol{H} - \lambda\mathbf{I} = \boldsymbol{U}_1\boldsymbol{R}_1$$

We construct a new matrix $\boldsymbol{H}_1$:

$$\boldsymbol{H}_1 = \boldsymbol{R}_1\boldsymbol{U}_1 + \lambda\mathbf{I}$$

Our goal is to show that:

$$\boldsymbol{H}_1(m,:) = \lambda e_m^T$$

First, note that since $\boldsymbol{H}$ is an upper Hessenberg matrix, the last row of $\boldsymbol{H}$ can be written as:

$$\boldsymbol{H}(m,:) = (0,\ldots,0,h_{m,m-1},h_{m,m})$$

Now, let's analyze the last row of the matrix $\boldsymbol{H}_1$:

$$\boldsymbol{H}_1(m,:) = (\boldsymbol{R}_1\boldsymbol{U}_1 + \lambda\mathbf{I})(m,:)$$

Since $\boldsymbol{H} - \lambda\mathbf{I} = \boldsymbol{U}_1\boldsymbol{R}_1$, we can substitute $\boldsymbol{U}_1\boldsymbol{R}_1$ for $\boldsymbol{H} - \lambda\mathbf{I}$ in the last row of $\boldsymbol{H}$:

$$\boldsymbol{H}(m,:) = (0,\ldots,0,h_{m,m-1} + \lambda_{m-1},h_{m,m} - \lambda_m)$$

The last row of $\boldsymbol{H}_1$ can be computed as:

$$\boldsymbol{H}_1(m,:) = \left(\boldsymbol{R}_1(m,:)\boldsymbol{U}_1 + \lambda e_m^T\right)$$

Since the last row of $\boldsymbol{R}_1$ is all zeros except for the last element, and the last column of $\boldsymbol{U}_1$ is zero except for the last element, we have:

$$\boldsymbol{R}_1(m,:)\boldsymbol{U}_1 = (0,\ldots,0,h_{m,m-1} + \lambda_{m-1},h_{m,m} - \lambda_m)$$

Now, we add the last row of $\lambda\mathbf{I}$ to the last row of $\boldsymbol{R}_1(m,:)\boldsymbol{U}_1$:

$$\boldsymbol{H}_1(m,:) = (0,\ldots,0,h_{m,m-1} + \lambda_{m-1} + \lambda_m,h_{m,m} - \lambda_m + \lambda_m)$$

Since λ_m is an eigenvalue of $\boldsymbol{H}$, we obtain:

$$\boldsymbol{H}_1(m,:) = (0,\ldots,0,\lambda) = \lambda e_m^T$$

(b) [10 pts]

Explain the connection between this result and the process of deflation in the QR iteration algorithm.

The result we just proved is related to the process of deflation in the QR iteration algorithm. Deflation is a technique used to simplify the QR algorithm by reducing the size of the matrix on which the QR iteration is applied. When an off-diagonal entry of the matrix converges to zero (or becomes very small), the matrix can be divided into smaller matrices (deflated), and the QR iteration is then applied to each of these smaller matrices separately.

In the context of the result from part (a), if $\boldsymbol{H}_{m,m-1}$ converges to zero (or becomes very small), the last row of the matrix $\boldsymbol{H}_1$ becomes λe_m^T. This means that the matrix $\boldsymbol{H}_1$ can be partitioned into two smaller matrices, one of size $(m-1) \times (m-1)$ and the other of size 1×1. The 1×1 matrix contains the converged eigenvalue λ and is deflated from the rest of the matrix. The QR iteration can then be applied to the smaller $(m-1) \times (m-1)$ matrix, resulting in a more efficient algorithm.

Deflation can significantly speed up the QR iteration algorithm, especially when dealing with large matrices. By breaking the matrix into smaller submatrices and applying the QR iteration to each submatrix separately, the computation becomes more efficient and the algorithm can converge faster.

3 An implicit QR Factorization [15 bonus pts]

Denote $\boldsymbol{H} = \boldsymbol{H}_1$, and assume we generate a sequence of matrices $\boldsymbol{H}_k$ via

$$\boldsymbol{H}_k - \mu_k \mathbf{I} = \boldsymbol{U}_k \boldsymbol{R}_k, \quad \boldsymbol{H}_{k+1} = \boldsymbol{R}_k \boldsymbol{U}_k + \mu_k \mathbf{I}. \tag{9}$$

Prove that

$$(\boldsymbol{U}_1 \cdots \boldsymbol{U}_j)(\boldsymbol{R}_j \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_j \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I}). \tag{10}$$

This result shows that we are implicitly computing a QR factorization of

$$(\boldsymbol{H} - \mu_j \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I}). \tag{11}$$

We will prove this result by induction. Base case ($j = 1$):

$$(\boldsymbol{U}_1)(\boldsymbol{R}_1) = (\boldsymbol{H} - \mu_1 \mathbf{I})$$

This is true by the definition of the QR factorization in the problem statement.

Inductive step: Assume the result is true for $j = n$:

$$(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n)(\boldsymbol{R}_n \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_n \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I})$$

Now, we need to prove the result for $j = n + 1$:

$$(\boldsymbol{U}_1 \cdots \boldsymbol{U}_{n+1})(\boldsymbol{R}_{n+1} \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_{n+1} \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I})$$

Let's multiply both sides of the assumed equation by $(\boldsymbol{H} - \mu_{n+1} \mathbf{I})$:

$$(\boldsymbol{H} - \mu_{n+1} \mathbf{I})(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n)(\boldsymbol{R}_n \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_{n+1} \mathbf{I})(\boldsymbol{H} - \mu_n \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I})$$

Now, by the definition of $\boldsymbol{H}_{n+1}$:

$$\boldsymbol{H}_{n+1} - \mu_{n+1} \mathbf{I} = \boldsymbol{U}_{n+1} \boldsymbol{R}_{n+1}$$

Substitute $\boldsymbol{H}_{n+1}$ with $\boldsymbol{U}_{n+1}\boldsymbol{R}_{n+1} + \mu_{n+1}\mathbf{I}$:

$$(\boldsymbol{U}_{n+1}\boldsymbol{R}_{n+1} + \mu_{n+1}\mathbf{I} - \mu_{n+1}\mathbf{I})(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n)(\boldsymbol{R}_n \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_{n+1}\mathbf{I})(\boldsymbol{H} - \mu_n\mathbf{I}) \cdots (\boldsymbol{H} - \mu_1\mathbf{I})$$

Simplify the expression:

$$(\boldsymbol{U}_{n+1}\boldsymbol{R}_{n+1})(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n)(\boldsymbol{R}_n \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_{n+1}\mathbf{I})(\boldsymbol{H} - \mu_n\mathbf{I}) \cdots (\boldsymbol{H} - \mu_1\mathbf{I})$$

Using the associative property, we can rewrite this as:

$$(\boldsymbol{U}_{n+1}(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n))(\boldsymbol{R}_{n+1}(\boldsymbol{R}_n \cdots \boldsymbol{R}_1)) = (\boldsymbol{H} - \mu_{n+1}\mathbf{I})(\boldsymbol{H} - \mu_n\mathbf{I}) \cdots (\boldsymbol{H} - \mu_1\mathbf{I})$$

Now, observe that:

$$\boldsymbol{U}_{n+1}(\boldsymbol{U}_1 \cdots \boldsymbol{U}_n) = (\boldsymbol{U}_1 \cdots \boldsymbol{U}_{n+1})$$
$$\boldsymbol{R}_{n+1}(\boldsymbol{R}_n \cdots \boldsymbol{R}_1) = (\boldsymbol{R}_{n+1} \cdots \boldsymbol{R}_1)$$

Thus, we have shown that the result is true for $j = n + 1$, which completes the inductive step. Therefore, the result holds for all j:

$$(\boldsymbol{U}_1 \cdots \boldsymbol{U}_j)(\boldsymbol{R}_j \cdots \boldsymbol{R}_1) = (\boldsymbol{H} - \mu_j\mathbf{I}) \cdots (\boldsymbol{H} - \mu_1\mathbf{I})$$

This result shows that we are implicitly computing a QR factorization of $(\boldsymbol{H} - \mu_j\mathbf{I}) \cdots (\boldsymbol{H} - \mu_1\mathbf{I})$.

4 QR with Shifts [30 pts]

(a) Almost upper triangular [7.5 pts]

Go to section (a) of the file `HW6_your_code.jl` and implement a function that reduces a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ to Hessenberg form using Householder reflections. You should end up with a matrix $\boldsymbol{T}$ in Hessenberg form. Your algorithm should operate in place, overwriting the input matrix and not allocating additional memory.

(b) Givens [7.5 pts]

Go to section (b) of the file `HW6_your_code.jl` and implement a function that runs a single iteration of the unshifted QR algorithm. Your function should take $\boldsymbol{T}_k$ in Hessenberg form as an input and compute $\boldsymbol{T}_{k+1}$ also in Hessenberg form. You should use Givens rotations to implement QR-factorization on $\boldsymbol{T}_k$.

(c) Single-Shift vs. Wilkson Shifts [7.5 pts]

Go to section (c) of the file `HW6_your_code.jl` and implement a function that runs the practical QR iteration with both the Single-Shift and Wilkinson Shift. Your function should have an input that allows you to select which type of shift you want to use. Your implementation should include deflation and a reasonable criteria for when to implement deflation and terminate your QR iterations. You can use your function from part (b) to do the QR iteration at each step.

(d) Breaking symmetry [7.5 pts]

Go to section (d) of the file `HW6_your_driver.jl` and design an experiment that evaluates your practical QR algorithm with shifts. You should include a semi-log plot showing the rate of convergence of your algorithm using Single-Shift and Wilkinson Shift. Compare the results with the rate of convergence you expected to see for both cases. Do you have a preference between the Wilkinson shift or the Rayleigh shift? If so which one do you prefer and why?