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 use induction. For $k = 0$, $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ is upper Hessenberg by assumption.

Now, assume that $\boldsymbol{A}^{(k)}$ is upper Hessenberg. To find $\boldsymbol{A}^{(k+1)}$, we perform a QR factorization of $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$.

Since $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ is an upper Hessenberg matrix, the resulting $\boldsymbol{Q}_k$ factor will also be upper Hessenberg.

To show this, multiply both sides of $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ by $\boldsymbol{R}_k^{-1}$ to get

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

$\boldsymbol{A}^{(k)}$ is upper Hessenberg by our inductive assumption. $\mu_k \mathbf{I}$ is a diagonal matrix, so $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ is also upper Hessenberg. $\boldsymbol{R}_k$ is an upper triangular matrix, so $\boldsymbol{R}_k^{-1}$ is also upper triangular.

Thus, $\boldsymbol{Q}_k$ is the product of an upper Hessenberg matrix and an upper triangular matrix, and so it is also upper Hessenberg.

Now, to show that $\boldsymbol{A}^{(k+1)} = \boldsymbol{R}_k \boldsymbol{Q}_k + \mu_k \mathbf{I}$ is upper Hessenberg, we need to prove that the entries below the first subdiagonal are zero.

Consider entry $a_{i,j}^{(k+1)}$ of $\boldsymbol{A}^{(k+1)}$, where $i > j+1$:

$$\begin{aligned}
\boldsymbol{A}^{(k+1)} &= \boldsymbol{R}_k \boldsymbol{Q}_k + \mu_k \mathbf{I} \\
a_{i,j}^{(k+1)} &= (\boldsymbol{R}_k \boldsymbol{Q}_k)_{i,j} + (\mu_k \mathbf{I})_{i,j} \\
&= (\boldsymbol{R}_k \boldsymbol{Q}_k)_{i,j} && (\mu_k \mathbf{I} \text{ diag.} \to (\mu_k \mathbf{I})_{i,j} = 0, \forall i \neq j) \\
&= \sum_{p=1}^{m} r_{i,p}^{(k)} q_{p,j}^{(k)}
\end{aligned}$$

Since $\boldsymbol{R}_k$ is upper triangular, $r_{i,p}^{(k)} = 0$ for all $i > p$. Also, since $\boldsymbol{Q}_k$ is upper Hessenberg, $q_{p,j}^{(k)} = 0$ for all $p > j+1$. Combining these observations, we can conclude that $(\boldsymbol{R}_k \boldsymbol{Q}_k)_{i,j} = 0$ for all $i > j+1$, and so $\boldsymbol{A}^{(k+1)}$ is upper Hessenberg.

Thus, we have shown that if $\boldsymbol{A}^{(k)}$ is upper Hessenberg, then $\boldsymbol{A}^{(k+1)}$ is also upper Hessenberg, which completes the induction.

(b) [10 pts]

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

A single step of QR iteration involves a QR factorization and computing $\boldsymbol{A}^{(k+1)}$ from the QR factors.

1. QR factorization:

   $\boldsymbol{A}^{(k)} - \mu_k \mathbf{I}$ is upper Hessenberg. We can compute its QR factorization using Givens rotations. Each Givens rotation zeros out one subdiagonal element in each column. Since there are $m-1$ nonzero subdiagonal elements, there will be $O(m-1)$ Givens rotations. Each Givens rotation requires $4m$ operations (2 multiplications and 2 additions per entry across two rows), giving a total cost of $4m(m-1) = O(m^2)$.

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

   $\boldsymbol{R}_k \boldsymbol{Q}_k$ is the product of an upper triangular matrix and an upper Hessenberg matrix, so we can compute the product using $O(m^2)$ operations. Adding the shift $\mu_k \mathbf{I}$ takes $O(m)$ operations.

Thus, the total operation cost for each QR iteration is $O(m^2) + O(m^2) + O(m) = O(m^2)$.

(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

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

Since we have applied $m-2$ Givens rotations, the (m, m) entry is 0 because each Givens rotation introduces a 0 entry below the diagonal, and we have only one non-zero entry left in the last subdiagonal.

We are looking for the lower-left entry of the bottom-right 2×2 submatrix of the product $G^T(\boldsymbol{A}^{(k)} - \mu_k \mathbf{I})G$, where G is the Givens rotation matrix:

$$G := \begin{pmatrix} c & -s \\ s & c \end{pmatrix},$$

with $c := \dfrac{a}{\sqrt{a^2+\varepsilon^2}}$ and $s := \dfrac{\varepsilon}{\sqrt{a^2+\varepsilon^2}}$.

For brevity, let's denote the subscript 2×2 as the lower-right 2×2 submatrix of a matrix.

We have:

$$\begin{aligned} \left(G^T(\boldsymbol{A}^{(k)} - \mu_k \mathbf{I})G\right)_{2\times 2} &= \left(G^T \boldsymbol{A}^{(k)} G\right)_{2\times 2} - \left(G^T \mu_k \mathbf{I} G\right)_{2\times 2} \\ &= \begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} + \begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} \mu_k & 0 \\ 0 & \mu_k \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \end{aligned}$$

Note that we ultimately only care about the lower-left entry of the bottom-right 2×2 submatrix.

We can see the right term will contribute nothing to this entry:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} \mu_k & 0 \\ 0 & \mu_k \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} \cdots & \cdots \\ \mu_k(-sc+cs) & \cdots \end{pmatrix} = \begin{pmatrix} \cdots & \cdots \\ 0 & \cdots \end{pmatrix}$$

This leaves us with the left term to compute:

$$\begin{gathered} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \\ \begin{pmatrix} ac+s\varepsilon & cb \\ -sa+c\varepsilon & -sb \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \\ \begin{pmatrix} \cdots & \cdots \\ (-sa+c\varepsilon)c + (-sb)s & \cdots \end{pmatrix} \end{gathered}$$

Simplifying the lower-left entry and substituting for c and s,

$$\begin{aligned} (-sa+c\varepsilon)c + (-sb)s &= -csa + c^2\varepsilon - s^2 b \\ &= -\frac{a^2\varepsilon}{a^2+\varepsilon^2} + \frac{a^2\varepsilon}{a^2+\varepsilon^2} - \frac{\varepsilon^2 b}{\varepsilon^2+a^2} \\ &= -\frac{\varepsilon^2 b}{\varepsilon^2+a^2}, \end{aligned}$$

which is our desired result.

(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).

In part (c), we derived that the off-diagonal entry of the bottom-right 2×2 sub-matrix of $\boldsymbol{A}^{(k+1)}$ is given by

$$-\frac{\varepsilon^2 b}{\varepsilon^2+a^2}.$$

The ε^2 is in the numerator means that as $\varepsilon \to 0$, the off-diagonal entry $\boldsymbol{A}^{(k+1)}_{m,m-1}$ will also approach 0. That is, the off-diagonal entry decreases quadratically with respect to ε. Since the QR algorithm aims to drive the off-diagonal entries to 0 to reveal the eigenvalues along the diagonal, this suggests the single-shift QR algorithm will converge quadratically if it converges at all.

(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.

Consider the following matrix:

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

The true eigenvalues of $\boldsymbol{A}$ are $\lambda_1 = 1$ and $\lambda_2 = -1$, and so $|\lambda_1| = |\lambda_2| = 1$.

Thus, the single-shift QR algorithm cannot decide which direction to move its estimate for the eigenvalues, and so it cannot proceed forward.

To see this, consider the first iteration of the single-shift QR algorithm applied to $\boldsymbol{A}$.

- Initialize $\boldsymbol{T}_0 = \boldsymbol{A}$.
- Since the diagonal entries are both zero, we choose $\mu_1 = 0$ as the shift (our "eigenvalue estimate").
- Then, we take the QR factorization of $\boldsymbol{T}_0 - \mu_1 \mathbf{I} = \boldsymbol{T}_0$.

We can already see that we can make no progress. Furthermore, note that the off-diagonals are not zero (or near-zero), and never will be, and so we cannot use deflation to split up the problem and make progress that way either.

As explained in class, we need a way to break the symmetry of the problem, and this is why we introduce the Wilkinson Shift.

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

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

$$
\begin{aligned}
\boldsymbol{Hx} &= \lambda \boldsymbol{x} && \text{(eigenvector equation for } \boldsymbol{H}) \\
\boldsymbol{U}_1^T(\boldsymbol{H} - \lambda \mathbf{I})\boldsymbol{x} &= \boldsymbol{U}_1^T \boldsymbol{Hx} - \lambda \boldsymbol{U}_1^T \boldsymbol{x} = \boldsymbol{R}_1 \boldsymbol{y} && (\boldsymbol{y} \coloneqq \boldsymbol{U}_1^T \boldsymbol{x}) \\
\boldsymbol{H}_1 \boldsymbol{y} &= (\boldsymbol{U}_1^T(\boldsymbol{H} - \lambda \mathbf{I})\boldsymbol{x} + \lambda \mathbf{I})\boldsymbol{y} && \text{(substitute } \boldsymbol{R}_1 \boldsymbol{y}) \\
\boldsymbol{H}_1 \boldsymbol{y} &= \boldsymbol{U}_1^T \boldsymbol{Hx} - \lambda \boldsymbol{U}_1^T \boldsymbol{x} + \lambda \boldsymbol{y} && \text{(simplify)} \\
\boldsymbol{H}_1 \boldsymbol{y} &= \boldsymbol{U}_1^T \lambda \boldsymbol{x} - \lambda \boldsymbol{U}_1^T \boldsymbol{x} + \lambda \boldsymbol{y} && \text{(substitute } \boldsymbol{Hx} = \lambda \boldsymbol{x}) \\
\boldsymbol{H}_1 \boldsymbol{y} &= \lambda \boldsymbol{y} && \text{(simplify)} \\
h_{m,m-1} y_{m-1} + h_{m,m} y_m &= \lambda y_m && \text{(examine the last element)}
\end{aligned}
$$

We can now conclude that $h_{m,m} = \lambda$ and $h_{m,m-1} = 0$, and so

$$
\boldsymbol{H}_1(m, :) = \begin{bmatrix} 0 & \cdots & 0 & 0 & \lambda \end{bmatrix} = \lambda e_m^T.
$$

(b) [10 pts]

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

When an off-diagonal entry of the matrix in the QR iteration converges to near-zero, we can "deflate" the matrix by dividing it into two smaller matrices, and applying the QR iteration to each smaller matrix separately.

In problem (a), we found that if $\boldsymbol{H}_{m,m-1}$ converges to zero, 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.

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 proove this result by induction on k.

Base case ($k = 1$):

$$
\begin{aligned}
\boldsymbol{U}_k \boldsymbol{R}_k &= \boldsymbol{H}_k - \mu_k \mathbf{I} && \text{(given)} \\
\boldsymbol{U}_1 \boldsymbol{R}_1 &= \boldsymbol{H}_1 - \mu_1 \mathbf{I} && \text{(set } k = 1) \\
(\boldsymbol{U}_1)(\boldsymbol{R}_1) &= (\boldsymbol{H} - \mu_1 \mathbf{I}) && \text{(since } \boldsymbol{H} \coloneqq \boldsymbol{H}_1\text{. QED for base case)}
\end{aligned}
$$

Inductive step: Assume that the result holds for $k > 0$.

Define $\boldsymbol{Q}_k \coloneqq (\boldsymbol{U}_1 \cdots \boldsymbol{U}_k)$, $\boldsymbol{P}_k \coloneqq (\boldsymbol{R}_k \cdots \boldsymbol{R}_1)$, and $\boldsymbol{G}_k \coloneqq (\boldsymbol{H} - \mu_k \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I})$.

Then, our inductive assumption is

$$\begin{aligned} (\boldsymbol{U}_1 \cdots \boldsymbol{U}_k)(\boldsymbol{R}_k \cdots \boldsymbol{R}_1) &= (\boldsymbol{H} - \mu_k \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I}) \\ \boldsymbol{Q}_k \boldsymbol{P}_k &= \boldsymbol{G}_k. \end{aligned}$$

We want to show that the result also holds for $k+1$. That is, we want to show that

$$\begin{aligned} (\boldsymbol{U}_1 \cdots \boldsymbol{U}_{k+1})(\boldsymbol{R}_{k+1} \cdots \boldsymbol{R}_1) &= (\boldsymbol{H} - \mu_{k+1} \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I}) \\ (\boldsymbol{U}_1 \cdots \boldsymbol{U}_k \boldsymbol{U}_{k+1})(\boldsymbol{R}_{k+1} \boldsymbol{R}_k \cdots \boldsymbol{R}_1) &= (\boldsymbol{H} - \mu_{k+1} \mathbf{I})(\boldsymbol{H} - \mu_k \mathbf{I}) \cdots (\boldsymbol{H} - \mu_1 \mathbf{I}) \\ \boldsymbol{Q}_k \boldsymbol{U}_{k+1} \boldsymbol{R}_{k+1} \boldsymbol{P}_k &= (\boldsymbol{H} - \mu_{k+1} \mathbf{I}) \boldsymbol{G}_k. \end{aligned}$$

First, observe the following relation:

$$\begin{aligned} \boldsymbol{H}_k - \mu_k \mathbf{I} &= \boldsymbol{U}_k \boldsymbol{R}_k && \text{(given)} \\ (\boldsymbol{U}_k)^T \boldsymbol{H}_k \boldsymbol{U}_k &= (\boldsymbol{U}_k)^T (\boldsymbol{U}_k \boldsymbol{R}_k + \mu_k \mathbf{I}) \boldsymbol{U}_k && \text{(mult. left and right by } (\boldsymbol{U}_k)^T \text{ and } \boldsymbol{U}_k) \\ &= \boldsymbol{R}_k \boldsymbol{U}_k + \mu_k \mathbf{I} && \text{(since } \boldsymbol{U}_k \text{ is orthonormal)} \\ (\boldsymbol{U}_k)^T \boldsymbol{H}_k \boldsymbol{U}_k &= \boldsymbol{H}_{k+1} && \text{(by definition of } \boldsymbol{H}_{k+1}) \end{aligned}$$

I am not sure how to progress from here. Looking forward to reading the answer!

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?

I was not able to complete this. I struggled to ensure that the two shifts both resulted in the same selected eigenvalue to converge for comparison, and I wasn't able to produce anything meaningful in the time I had.