MAS565 Numerical Analysis HW8

20218125 채지석

#6.19.(a) By writing $R\Omega = \Omega \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix}$ and henceforth

$$R\Omega e_1 = \Omega \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix} e_1 = \lambda_2 \Omega e_1$$

and noting that Ω is invertible so $\Omega e_1 \neq 0$ we see that Ωe_1 is an eigenvector of R corresponding to eigenvalue λ_2. As $\lambda_1 \neq \lambda_2$ we have $R - \lambda_2 I = \begin{bmatrix} \lambda_1 - \lambda_2 & * \\ 0 & 0 \end{bmatrix}$ so $\begin{bmatrix} \frac{*}{\lambda_2 - \lambda_1} \\ 1 \end{bmatrix}$ becomes an eigenvector of R corresponding to λ_2 in $\mathbb{R}^2$. That is, we can set $\theta \in \mathbb{R}$ so that $\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ becomes an eigenvector of R corresponding to λ_2. We claim that $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is the desired Givens rotation Ω. To show this claim it suffices to show that the equation $R\Omega = \Omega \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix}$ holds. On the left hand side we have

$$R\Omega = \begin{bmatrix} \lambda_1 & * \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \lambda_1\cos\theta + *\sin\theta & -\lambda_1\sin\theta + *\cos\theta \\ \lambda_2\sin\theta & \lambda_2\cos\theta \end{bmatrix}$$

and since $R\Omega e_1 = \lambda_2 \Omega e_1 = \lambda_2 \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ we see that $\lambda_2\cos\theta = \lambda_1\cos\theta + *\sin\theta$. Now

$$\begin{aligned}\Omega \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix} &= \begin{bmatrix} \lambda_2\cos\theta & *\cos\theta - \lambda_1\sin\theta \\ \lambda_2\sin\theta & *\sin\theta + \lambda_1\cos\theta \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\cos\theta + *\sin\theta & *\cos\theta - \lambda_1\sin\theta \\ \lambda_2\sin\theta & \lambda_2\cos\theta \end{bmatrix} \\ &= R\Omega\end{aligned}$$

so indeed our assertion holds.

(b) Even if R was a complex 2×2 matrix, still we can make Ω to be a (complex) Givens rotation matrix $\begin{bmatrix} \cos\theta & -e^{-i\varphi}\sin\theta \\ e^{i\varphi}\sin\theta & \cos\theta \end{bmatrix}$ so that its first row Ωe_1 is parallel to $\begin{bmatrix} \frac{*}{\lambda_2-\lambda_1} \\ 1 \end{bmatrix}$ by adjusting the phase by φ and the ratio of entries by θ. Then, as we had $R\Omega = \Omega \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix}$ under the assumption that $R\Omega e_1 = \lambda_2 \Omega e_1$ only, we also would have $\Omega^H R \Omega = \begin{bmatrix} \lambda_2 & * \\ 0 & \lambda_1 \end{bmatrix}$.

Now we get back to the given situation where $R \in \mathbb{C}^{n\times n}$. For any $j \in \{1, \cdots, n-1\}$ partition R into $\begin{bmatrix} R_1 & * & * \\ 0 & R_2 & * \\ 0 & 0 & R_3 \end{bmatrix}$ so that $R_1 \in \mathbb{C}^{(j-1)\times(j-1)}$, $R_2 \in \mathbb{C}^{2\times 2}$, and $R_3 \in \mathbb{C}^{(n-j-1)\times(n-j-1)}$. As R_2 is in the form $\begin{bmatrix} \lambda_j & * \\ 0 & \lambda_{j+1} \end{bmatrix}$ there exists a Givens rotation Ω_j' such that $(\Omega_j')^H R_2 \Omega_j'$ is in the form $\begin{bmatrix} \lambda_{j+1} & * \\ 0 & \lambda_j \end{bmatrix}$. This is because if $\lambda_j = \lambda_{j+1}$ then we can let $\Omega_j' = I$. Now define a block diagonal matrix $\Omega_j = \begin{bmatrix} I & 0 & 0 \\ 0 & \Omega_j' & 0 \\ 0 & 0 & I \end{bmatrix}$ so that the block pattern matches that of R. Then by simple calculation we have

$$\Omega_j^H R \Omega_j = \begin{bmatrix} I & 0 & 0 \\ 0 & (\Omega_j')^H & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} R_1 & * & * \\ 0 & R_2 & * \\ 0 & 0 & R_3 \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & \Omega_j' & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} R_1 & * & * \\ 0 & (\Omega_j')^H R_2 \Omega_j' & * \\ 0 & 0 & R_3 \end{bmatrix}.$$

That is, $\Omega_j^H R \Omega_j$ is an upper triangular matrix with diagonal $\mathrm{diag}(\lambda_1, \cdots, \lambda_{j-1}, \lambda_{j+1}, \lambda_j, \lambda_{j+2}, \cdots, \lambda_n)$.

Decomposing a permutation $(1\ 2\ \cdots\ i)$ into composition of transpositions $(1\ 2)(2\,3)\cdots(i-1\ \ i)$, we see that $\tilde{R} = \Omega_1^H \Omega_2^H \cdots \Omega_{i-1}^H R \Omega_{i-1} \Omega_{i-2} \cdots \Omega_1$ suffices, and hence putting $U = \Omega_{i-1}\Omega_{i-2}\cdots\Omega_1$ we are done.

#6.20. We prove both by induction. That $A-k_1 I = P_1 U_1 = Q_1 R_1$ is immediate from the definitions, and $R_1 = Q^{-1}(A-k_1 I)$ so

$$A_2 = R_1 Q_1 + k_1 I = Q_1^{-1} A Q_1 - Q_1^{-1}(k_1 I) Q + k_1 I$$
$$= Q_1^{-1} A Q_1$$

follows from that Q_1 is orthogonal. Now, for any $i \in \mathbb{N}$, then as Q_i is also orthogonal we have $R_i = Q_i^{-1}(A_i - k_i I)$ so

$$A_{i+1} - k_i I = R_i Q_i = Q_i^{-1} A_i Q_i - k_i Q_i^{-1} Q_i$$
$$= Q_i^{-1} A_i Q_i - k_i I$$

hence $A_{i+1} = Q_i^{-1} A_i Q_i$, and by induction hypothesis we have

$$\begin{aligned} A_{i+2} &= Q_{i+1}^{-1} A_{i+1} Q_{i+1} \\ &= Q_{i+1}^{-1} P_i^H A P_i Q_{i+1} \\ &= Q_{i+1}^H Q_i^H \cdots Q_1^H A Q_1 Q_2 \cdots Q_i Q_{i+1} \\ &= P_{i+1}^H A P_{i+1}. \end{aligned}$$

For the second equation, first observe that for any $i \in \mathbb{N}$ it holds that

$$\begin{aligned} (A - k_{i+1} I) P_i U_i &= (A_1 - k_{i+1} I) Q_1 Q_2 \cdots Q_i R_i \cdots R_1 \\ &= (A_1 Q_1 - k_{i+1} Q_1) Q_2 \cdots Q_i R_i \cdots R_1 \\ &= (Q_1 A_2 - k_{i+1} Q_1) Q_2 \cdots Q_i R_i \cdots R_1 \\ &= Q_1 (A_2 - k_{i+1} I) Q_2 \cdots Q_i R_i \cdots R_1 \\ &= \cdots \\ &= Q Q_2 \cdots Q_i (A_{i+1} - k_{i+1} I) R_i \cdots R_1 \\ &= Q_1 Q_2 \cdots Q_i Q_{i+1} R_{i+1} R_i \cdots R_1 \\ &= P_{i+1} U_{i+1} \end{aligned}$$

so $P_{i+1} U_{i+1} = (A - k_{i+1} I)(A - k_1 I)(A - k_2 I) \cdots (A - k_i I)$. But each k_i are constants, and because $(x - k_{i+1})(x - k_1)(x - k_2) \cdots (x - k_i)$ and $(x - k_1)(x - k_2) \cdots (x - k_i)$ are same polynomials, we must have $P_{i+1} U_{i+1} = (A - k_1 I)(A - k_2 I) \cdots (A - k_i I)(A - k_{i+1} I)$.

\# 6.23. First we look at the example. Using numpy we get that $\delta_n' \approx 0.9973$ and $\gamma_n' \approx 7.445 \times 10^{-5}$. Indeed, numpy says that the minimum of $\lambda_i(B)$ is roughly 3.382, hence $d \approx 2.382$ and

$$\frac{|\gamma_n|^3}{d^2} \approx 1.762 \times 10^{-4} > \gamma_n',$$

$$\frac{|\gamma_n|^2}{d} \approx 4.198 \times 10^{-3} \geq 2.677 \times 10^{-3} = |\delta_n' - \delta_n|.$$

As indicated in the hint, we construct Q as a product of Givens rotations, mimicing the tridiagonalization by Givens rotation but here using Givens rotations in order to eliminate subdiagonal entries. Indexing according to the columns which a given Givens rotation affects, we name them so that $\Omega_{n-1,n} \cdots \Omega_{2,3}\, \Omega_{1,2}(A - \delta_n I) = R$ hence $Q^H = \Omega_{n-1,n} \cdots \Omega_{1,2}$.

This is the farthest I could get in this problem, and I tried only to fail devising a complete solution.

Computer Assignment

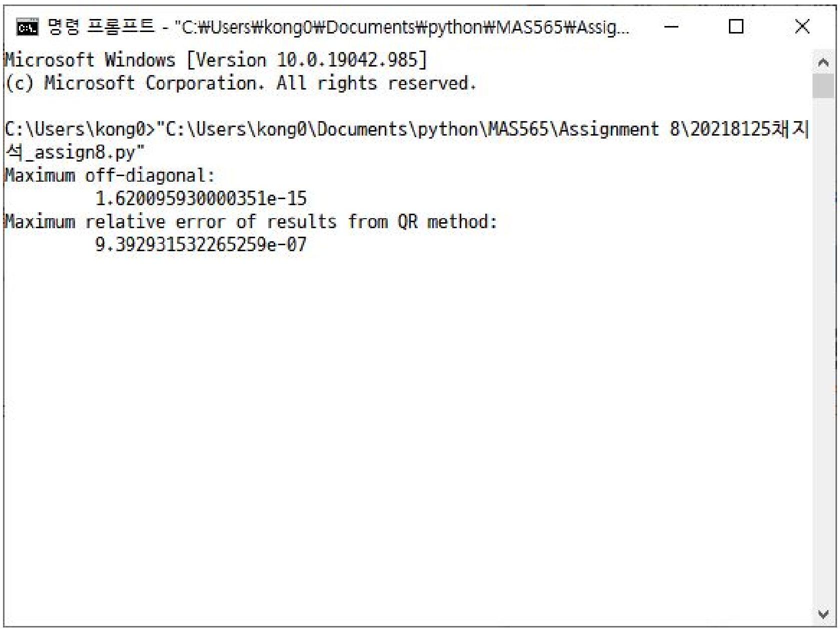
The program which does the required is submitted via KLMS along with this document. As it is a continuation of assignment 7, the first 130 lines of the code is almost identical with assignment 7.

The QR method is implemented, following the rules (6.6.4.3) of the textbook. Performing the QR decomposition is done using the `numpy` built-in function `numpy.linalg.qr`, although we also provide a self-made function `my_qr` which is capable of performing QR decomposition. However as expected, `my_qr` is much slower than `numpy.linalg.qr`, so practically we should use that instead of `my_qr`.

The stopping criteria is, as always, based on the relative error. This time we deal with matrices, so matrix norms are used in place of vector norms. Threshold is set to be 10^{-6}, also as always.

As mentioned in Theorem 6.6.4.15 of the textbook, QR iteration may not converge in general. Luckily in our case we see a convergence, but when we have no convergence the stopping criteria using relative error may not work properly. So this time, we also set a limit on the maximum number of iterations. Default value of this limit is set to be 1000.

To see if the QR iteration has converged or not we print out the maximum aboslute value of the off-diagonal entries. If the QR iteration has converged then the result will be close to a diagonal matrix, so this output indicates the convergence. Also it is required to compare the resulting eigenvalues with the results from assignment 7, so we compute the relative errors for each computed eigenvalues.



The results imply that the QR method has converged well, and the computed eigenvalues are not much different from the eigenvalues computed in assignment 7, which is where we computed the eigenvalues using the built-in method `numpy.linalg.eig`.

To have a better visual explanation on how well our QR method has performed we plot the two results, as the following figure. With our naked eye the two plots are nearly indistinguishable.

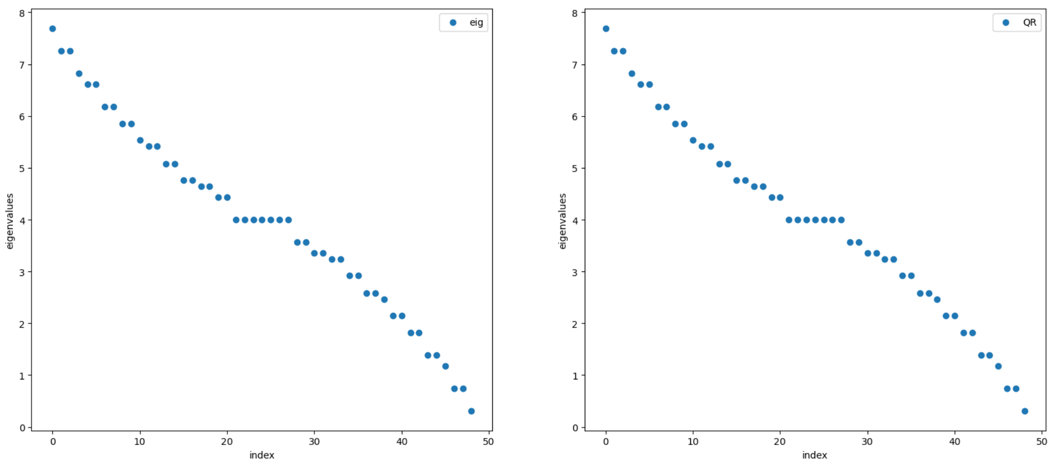


Figure 1: Scatter plots of the eigenvalues computed by `eig`(left) and the QR method(right)