# Numerical Analysis
# homework 07: Matrix Eigenvalues

Due on Tuesday, April 18, 2017

**102061149 Fu-En Wang**

# 1 Introduction

In previous homework, we had already use power method and inverse power method to find the largest and smallest eigenvalue. However, both of them cannot be used to find all the eigenvalue of a matrix $A$; therefore, it this project, we will use **QR Method** to find all the eigenvalues.

## 1.1 API

In **MAT.h** and **MAT.cpp**, I implement the following functions:

1. **void QRFact(const MAT &A, MAT &Q, MAT &R);**
2. **int EVqr(MAT &A, double tol, int maxiter);**
3. **int EVqrShifted(MAT &A, double mu, double tol, int maxiter);**

**QRFact** will apply **QR Decomposition** to A and store them into Q and R. **EVqr** and **EVqrShifted** is the implementation of **QR Iteration** and **Shifted QR Iteration**. In this assignment, **tol** should be set to $10^{-9}$ and **mu** should be 0.5. We have to find the three largest and the three smallest eigenvalues of **m3.dat**, **m4.dat**, **m5.dat**, **m6.dat**, **m7.dat**, **m8.dat**.

## 1.2 Error Calculation

According to **Gram-Schmidt Process**, we should get a final matrix with 0 in the non-diagonal from **QR Iteration**. Therefore, it makes sense that we can check whether all non-diagonal elements are zero or not. However, for saving execution time, we simply this checking to the following fomula:

$$error = \max_{2 \le i \le n} |a_{i,i-1}|$$

For each iteration, we only check the one-line elements below diagonal, which will be faster a lot than checking all non-diagonal elements.

# 2 Implementation

**Algorithm 1 QR Decomposition**

```
$A = \{a_1, a_2, ..., a_n\}, a_i$ is column vector,
$r_{11} = \sqrt{(a_1)^T a_1}$
$q_1 = \frac{a_1}{r_{11}}$
for each j ∈ {2, ..., n} do
    $q_j = a_j$
    for each i ∈ {1, ..., j-1} do
        $r_{ij} = (q_i)^T q_j$
        $q_j -= r_{ij} q_i$
    end for
    $r_{jj} = \sqrt{(q_j)^T q_j}$
    $q_j = \sqrt{sum((q_j)^2)}$
end for
```

**Algorithm 2 QR Iteration**

```
$T^{(0)} = A$
for each k ∈ {1, ..., maxIter} do
    $T^k = Q^k R^k$
    $T^{k+1} = R^k Q^k$
    if error < tol then
        break
    end if
end for
```

**Algorithm 3 Shifted QR Iteration**

```
$T^{(0)} = A$
for each k ∈ {1, ..., maxIter} do
    $T^k - \mu I = Q^k R^k$
    $T^{k+1} = R^k Q^k + \mu I$
    if error < tol then
        break
    end if
end for
```

## 2.1 Complexity

For **QR Decomposition**, the double for-loop do $\frac{n(n+1)}{2}$ times of operation, and $(q_i)^T q_j$ in each operation make **QR Decomposition** be a $\boldsymbol{O(n^3)}$ problem.
For each iteration of **QR Iteration**, there are **QR Decomposition** and **mat** $\times$ **mat**. So it is also $\boldsymbol{O(n^3)}$ problem.

## 3 Discussion

In the section, we will discuss the following topics:

1. Eigenvalues of m3, ..., m15.dat
2. Runtime of EVqr
3. Runtime of EVqrShifted
4. Overall complexity

### 3.1 Eigenvalues

Table 1 and 2 show eigenvalue calculated by **EVqr** and **EVqrShifted**, respectively.

| EVqr | N | E_large_1 | E_large_2 | E_large_3 | E_small_1 | E_small_2 | E_small_3 |
|---|---|---|---|---|---|---|---|
| m3 | 3 | 0.627719 | 2 | 6.372281 | 0.627719 | 2 | 6.372281 |
| m4 | 10 | 4.455992 | 20.431729 | 67.840399 | 0.512543 | 0.55164 | 0.629808 |
| m5 | 20 | 17.235222 | 81.223819 | 270.495189 | 0.503097 | 0.512479 | 0.528819 |
| m6 | 30 | 38.53868 | 182.544889 | 608.253606 | 0.501373 | 0.505511 | 0.512543 |
| m7 | 40 | 68.364136 | 324.394506 | 1081.115447 | 0.500772 | 0.503093 | 0.507004 |
| m8 | 50 | 106.711318 | 506.772618 | 1689.080688 | 0.500494 | 0.501978 | 0.504468 |

Table 1: Eigenvalue of EVqr

| EVqrShifted | N | E_large_1 | E_large_2 | E_large_3 | E_small_1 | E_small_2 | E_small_3 |
|---|---|---|---|---|---|---|---|
| m3 | 3 | 0.627719 | 2 | 6.372281 | 0.627719 | 2 | 6.372281 |
| m4 | 10 | 4.455992 | 20.431729 | 67.840399 | 0.512543 | 0.55164 | 0.629808 |
| m5 | 20 | 17.235222 | 81.223819 | 270.495189 | 0.503097 | 0.512479 | 0.528819 |
| m6 | 30 | 38.53868 | 182.544889 | 608.253606 | 0.501373 | 0.505511 | 0.512543 |
| m7 | 40 | 68.364136 | 324.394506 | 1081.115447 | 0.500772 | 0.503093 | 0.507004 |
| m8 | 50 | 106.711318 | 506.772618 | 1689.080688 | 0.500494 | 0.501978 | 0.504468 |
| m9 | 60 | 153.580161 | 729.679212 | 2432.149321 | 0.500343 | 0.501373 | 0.503097 |
| m10 | 70 | 208.970642 | 993.114283 | 3310.321344 | 0.500252 | 0.501008 | 0.502273 |
| m11 | 80 | 272.882751 | 1297.07783 | 4323.596758 | 0.500193 | 0.500772 | 0.501739 |
| m12 | 90 | 345.316483 | 1641.569852 | 5471.97556 | 0.500152 | 0.50061 | 0.501373 |
| m13 | 100 | 426.271836 | 2026.590348 | 6755.457752 | 0.500123 | 0.500494 | 0.501112 |
| m14 | 120 | 613.747401 | 2918.216761 | 9727.732302 | 0.500086 | 0.500343 | 0.500772 |
| m15 | 150 | 958.872886 | 4559.619934 | 15199.419543 | 0.500055 | 0.500219 | 0.500494 |

Table 2: Eigenvalue of EVqrShifted

From Table 2, we can plot the largest/smallest eigenvalue of m3 .... m15.dat as shown in Figure 1 and 2.

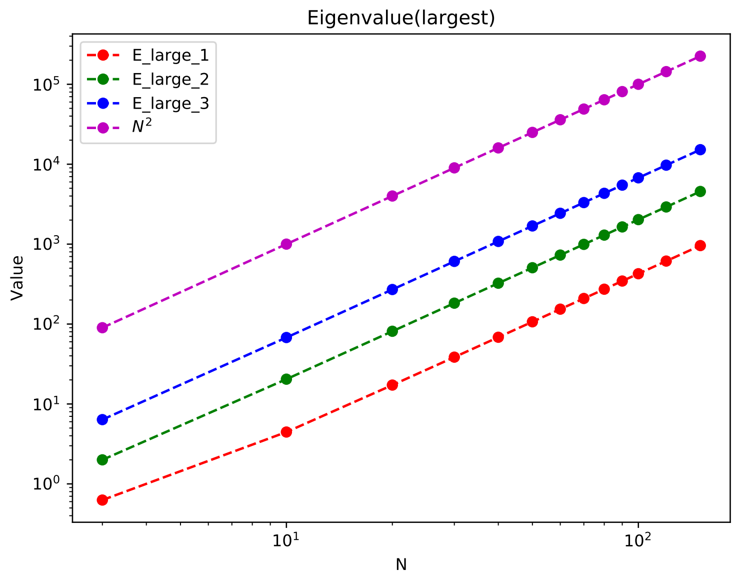

Figure 1: Largest eigenvalue

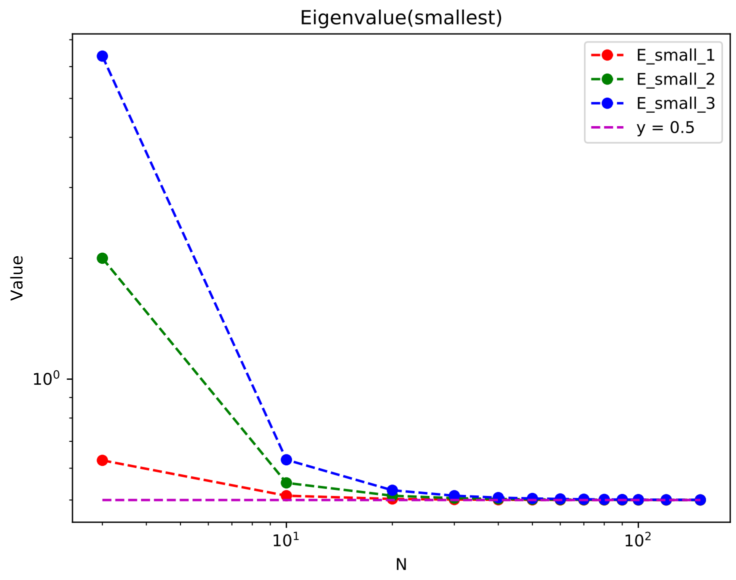

Figure 2: Smallest eigenvalue

In Figure 1 and 2, we can find that the three largest eigenvalues grow with $N^2$, while the three smallest eigenvalues converge to 0.5. As a result, in our case we can easily predict the eigenvalue for other N.

### 3.2 EVqr

Table 3 show the average iteration time(**iter_avg**) and iteration number(**iter_num**) for m3 ..... m8.dat.

| EVqr | N | iter_num | runtime(s) | iter_avg |
|---|---|---|---|---|
| m3 | 3 | 20 | 7.20E-05 | 3.6000E-06 |
| m4 | 10 | 249 | 0.005884 | 2.3631E-05 |
| m5 | 20 | 909 | 0.110081 | 1.2110E-04 |
| m6 | 30 | 1942 | 0.693329 | 3.5702E-04 |
| m7 | 40 | 3325 | 2.88519 | 8.6773E-04 |
| m8 | 50 | 5041 | 9.79014 | 1.9421E-03 |

Table 3: Itertion number and iter_avg of EVqr

From Table 3, we can plot iter_avg vs N and iter_num vs N as shown in Figure 3 and 4.

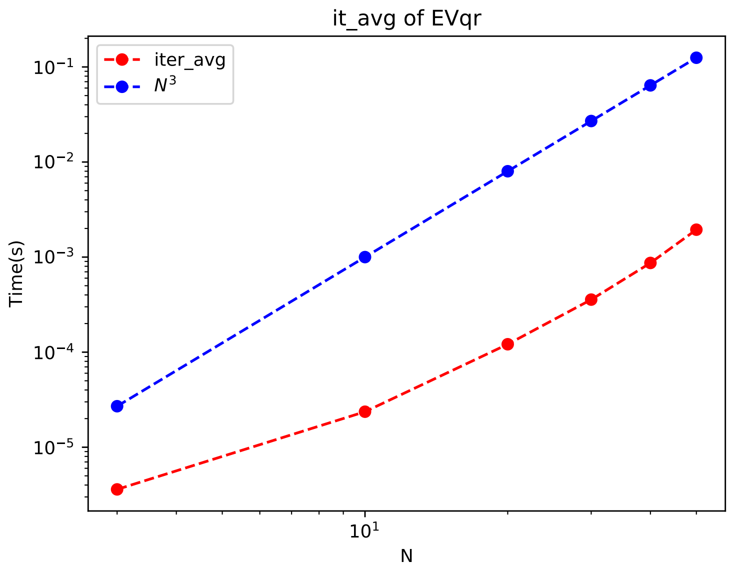

Figure 3: iter_avg vs N(EVqr)

In Figure 3, we can find that the slope of iter_avg is same as $N^3$, which means the complexity of each iteration of EVqr is $\boldsymbol{O(n^3)}$ and satisfies my analysis in Section 2.1.

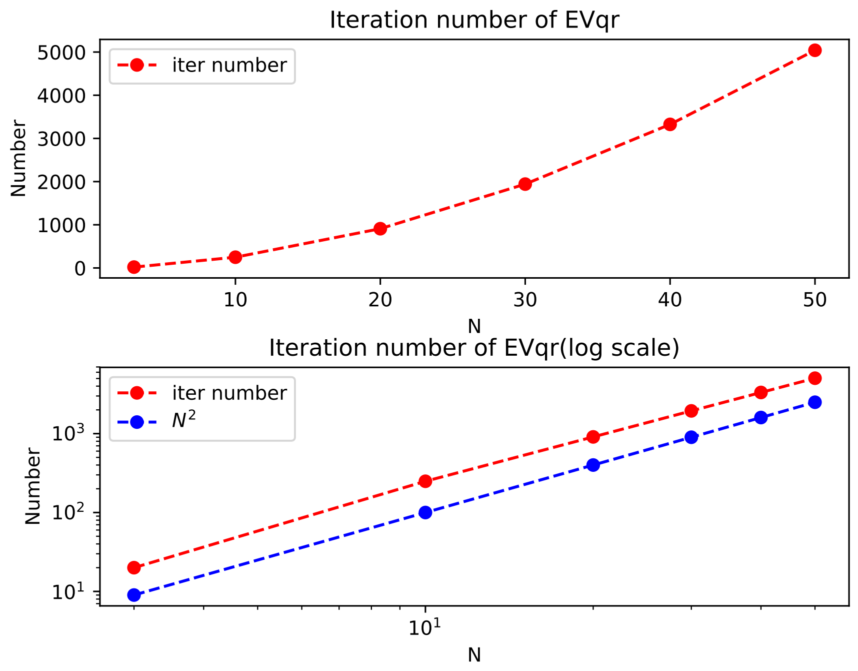

Figure 4: iter_num vs N(EVqr) in normal scale(top) and log scale(bottom)

From Figure 4, we can find that the iteration number of EVqr grows with $N^2$, which mean the complexity of iteration number in EVqr is $\boldsymbol{O(n^2)}$. Because the complexity of iter_avg and iter_num is $O(n^3)$ and $O(n^2)$, repectively, the overall complexity of EVqr will be $\boldsymbol{O(n^5)}$.

## 3.3 EVqrShifted

Table 4 show the average iteration time(**iter_avg**) and iteration number(**iter_num**) for m3 ..... m15.dat.

| EVqrShifted | N | iter_num | runtime(s) | iter_avg |
|---|---|---|---|---|
| m3 | 3 | 17 | 7.60E-05 | 4.4706E-06 |
| m4 | 10 | 35 | 0.000616 | 1.7600E-05 |
| m5 | 20 | 67 | 0.01132 | 1.6896E-04 |
| m6 | 30 | 104 | 0.039096 | 3.7592E-04 |
| m7 | 40 | 133 | 0.079233 | 5.9574E-04 |
| m8 | 50 | 167 | 0.17786 | 1.0650E-03 |
| m9 | 60 | 209 | 0.386447 | 1.8490E-03 |
| m10 | 70 | 244 | 0.724597 | 2.9697E-03 |
| m11 | 80 | 277 | 1.13667 | 4.1035E-03 |
| m12 | 90 | 310 | 1.81915 | 5.8682E-03 |
| m13 | 100 | 346 | 2.72404 | 7.8729E-03 |
| m14 | 120 | 425 | 5.60207 | 1.3181E-02 |
| m15 | 150 | 489 | 12.2082 | 2.4966E-02 |

Table 4: Itertion number and iter_avg of EVqrShifted

From Table 4, we can plot iter_avg vs N and iter_num vs N as shown in Figure 5 and 6.

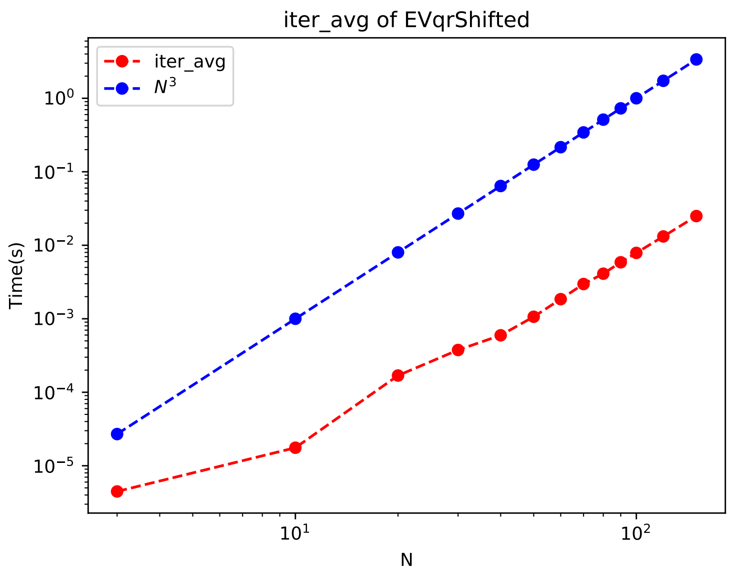

Figure 5: iter_avg vs N(EVqrShifted)

In the beginning of Figure 5, the slope is unstable because the runtime calculation has large error when N is small. After N is large enough. the slope is same as $N^3$, which means the complexity of each iteration in EVqrShifted is $\boldsymbol{O(n^3)}$ and satisfies my analysis in Section 2.1.

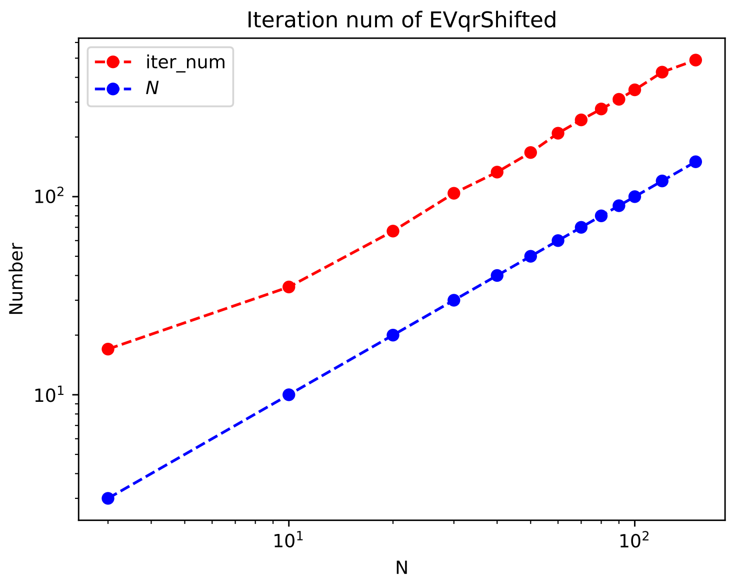

Figure 6: iter_num vs N(EVqrShifted)

In Figure 6, we can find that the iteration number grows with N, which means complexity of iteration number in EVqrShifted is $\boldsymbol{O(n)}$. Because the complexity of iter_avg and iter_num is $O(n^3)$ and $O(n)$, the overall complexity of EVqrShifted is $\boldsymbol{O(n^4)}$.

## 3.4 Overall Complexity

From Section 3.2 and 3.3, the overall complexity is listed in Table 5.

| Complexity | Evqr | EvqrShifted |
|---|---|---|
| iter_avg | $O(n^3)$ | $O(n^3)$ |
| iter_num | $O(n^2)$ | $O(n)$ |
| overall | $O(n^5)$ | $O(n^4)$ |

Table 5: Overall comlexity of EVqr and EVqrShifted

From Table 3 and 4, we can plot (iter_avg x iter_num) vs N as shown in Figure 7.

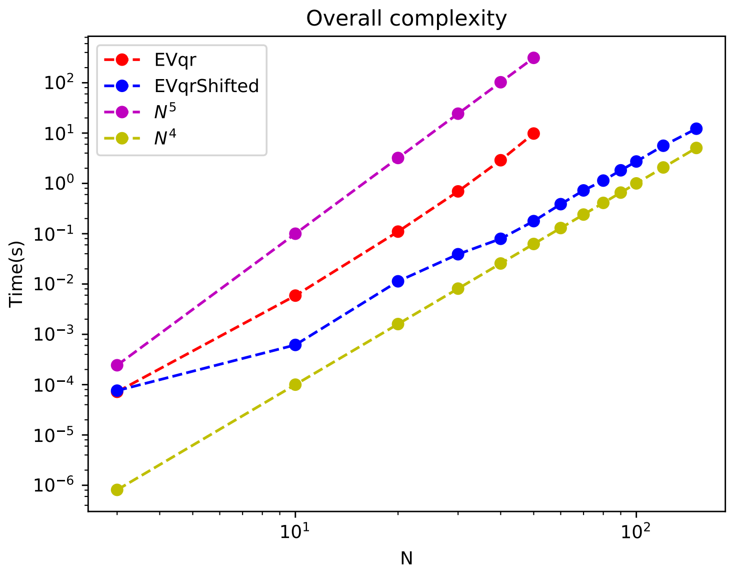

Figure 7: (iter_avg x iter_num) vs N

In Figure 7, the slope of EVqr is same as $N^5$, while the slope of EVrShifted is same as $N^4$. Therefore, the overall complexity satisfies my analysis in Section 3.2 and 3.3.
Because the overall complexity of EVqr and EVqrShifted is $\boldsymbol{O(n^5)}$ and $\boldsymbol{O(n^4)}$, respectively, EVqrShifted will be much faster when N is large.