**Numerical Analysis**

**Homework 7. Matrix Eigenvalues**

***102061125 陳冠鈞***

1. **Objective**

In this assignment, I will apply QR iterations and shifted QR iterations to find all the eigenvalues of different size matrices, and observe the convergence behavior and the CPU time.

1. **Approach**

I implement the following functions:

**void qrFact(MAT &A, MAT &Q, MAT &R);**

**int EVqr(MAT &A, double tol, int maxiter);**

**int EVqrShifted(MAT &A, double tol, int maxiter);**

**qrFact** will apply QR decomposition to A and get Q and R matrices. **EVqr** and **EVqrShifted** use **qrFact** to do QR iterations and shifted QR iterations. For this homework, the tolerance is   
set to be

**2.1. Error**

In QR iterations and shifted QR iterations, if , the error for each iteration is defined as

**2.1. Algorithm**

|  |
| --- |
| **Algorithm. QR decomposition** |
| Input: a matrix **A**.  Output: Q and R matrices. (A = Q x R)  (where is the *i*-th column vector of matrix A, and is the *j*-th column vector of matrix Q) |

|  |
| --- |
| **Algorithm. QR iterations** |
| Input: a matrix **A**, converge criterion.  Output: solution when maxIteration is reached or the error is smaller than .  calculate error for  (where “” is the matrix QR decomposition, and “” is simply matrix multiplication.) |

|  |
| --- |
| **Algorithm. Shifted QR iterations** |
| Input: a matrix **A**, a real number , converge criterion.  Output: solution when maxIteration is reached or the error is smaller than .  calculate error for  (where “” is the matrix QR decomposition, and “” is simply matrix multiplication.) |

1. **Computational Complexity**

**3.1. QR Decomposition**

* Because of double loops, we have to do iterations 🡪
* In second loop, we have to do a vector-vector multiplication 🡪
* Overall computational complexity 🡪

1. **Results**

**4.1. Eigenvalues**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Evqr | N | largest\_1 | largest\_2 | largest\_3 | smallest\_1 | smallest\_2 | smallest\_3 |
| m3 | 3 | 6.37228 | 2 | 0.627719 | 0.627719 | 2 | 6.37228 |
| m4 | 10 | 67.8404 | 20.4317 | 4.45599 | 0.512543 | 0.55164 | 0.629808 |
| m5 | 20 | 270.495 | 81.2238 | 17.2352 | 0.503097 | 0.512479 | 0.528819 |
| m6 | 30 | 608.254 | 182.545 | 38.5387 | 0.501373 | 0.505511 | 0.512543 |
| m7 | 40 | 1081.12 | 324.395 | 68.3641 | 0.500772 | 0.503093 | 0.507004 |
| m8 | 50 | 1689.08 | 506.773 | 106.711 | 0.500494 | 0.501978 | 0.504468 |

Table 1. The three largest and smallest eigenvalues of m3 to m8, EVqr.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EVqrShifted | N | largest\_1 | largest\_2 | largest\_3 | smallest\_1 | smallest\_2 | smallest\_3 |
| m3 | 3 | 6.37228 | 2 | 0.627719 | 0.627719 | 2 | 6.37228 |
| m4 | 10 | 67.8404 | 20.4317 | 4.45599 | 0.512543 | 0.55164 | 0.629808 |
| m5 | 20 | 270.495 | 81.2238 | 17.2352 | 0.503097 | 0.512479 | 0.528819 |
| m6 | 30 | 608.254 | 182.545 | 38.5387 | 0.501373 | 0.505511 | 0.512543 |
| m7 | 40 | 1081.12 | 324.395 | 68.3641 | 0.500772 | 0.503093 | 0.507004 |
| m8 | 50 | 1689.08 | 506.773 | 106.711 | 0.500494 | 0.501978 | 0.504468 |
| m9 | 60 | 2432.15 | 729.679 | 153.58 | 0.500343 | 0.501373 | 0.503097 |
| m10 | 70 | 3310.32 | 993.114 | 208.971 | 0.500252 | 0.501008 | 0.502273 |
| m11 | 80 | 4323.6 | 1297.08 | 272.883 | 0.500193 | 0.500772 | 0.501739 |
| m12 | 90 | 5471.98 | 1641.57 | 345.316 | 0.500152 | 0.50061 | 0.501373 |
| m13 | 100 | 6755.46 | 2026.59 | 426.272 | 0.500123 | 0.500494 | 0.501112 |
| m14 | 120 | 9727.73 | 2918.22 | 613.747 | 0.500086 | 0.500343 | 0.500772 |
| m15 | 150 | 15199.4 | 4559.62 | 958.873 | 0.500055 | 0.500219 | 0.500494 |

Table 2. The three largest and three smallest eigenvalues of m3 to m15, EVqrShifted

**4.2. CPU Time and Iterations**

I want to discuss the relationship between N vs. CPU time, iterations using EVqr and EVqrShifted. (**N** means the # of row (col) of the square matrix, **iter** means the iterations and **avg\_iter\_time** means the average iteration time.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Evqr | N | iter | time | avg\_iter\_time |
| m3 | 3 | 20 | 0 | 0 |
| m4 | 10 | 249 | 0.046875 | 0.000188253 |
| m5 | 20 | 909 | 1.07812 | 0.001186051 |
| m6 | 30 | 1942 | 7.26562 | 0.003741308 |
| m7 | 40 | 3325 | 30.6094 | 0.009205835 |
| m8 | 50 | 5041 | 97.9688 | 0.019434398 |

Table 3. Iterations and average iteration time, EVqr.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EVqrShifted | N | iter | time | avg\_iter\_time |
| m3 | 3 | 17 | 0 | 0 |
| m4 | 10 | 35 | 0 | 0 |
| m5 | 20 | 67 | 0.015625 | 0.000233209 |
| m6 | 30 | 100 | 0.140625 | 0.00140625 |
| m7 | 40 | 133 | 0.390625 | 0.00293703 |
| m8 | 50 | 167 | 0.9375 | 0.005613772 |
| m9 | 60 | 209 | 1.98438 | 0.009494641 |
| m10 | 70 | 244 | 4.39062 | 0.017994344 |
| m11 | 80 | 262 | 6.95312 | 0.026538626 |
| m12 | 90 | 310 | 11.2812 | 0.036390968 |
| m13 | 100 | 326 | 16.7969 | 0.051524233 |
| m14 | 120 | 383 | 35.5312 | 0.092770757 |
| m15 | 150 | 506 | 80.75 | 0.15958498 |

Table 4. Iterations and average iteration time, EVqrShifted.

**4.2.1. Average Iteration Time vs. N**

Figure 1. average iteration time vs. N, EVqr

Figure 2. average iteration time vs. N, EVqrShifted

In each iteration of EVqr and EVqrShifted, there is a **QR decomposition**. In Figure 1 and Figure 2, both the average iteration time grows with . These results and my analysis in session 3 match up.

**4.2.2. Iterations vs. N**

Figure 3. Iterations vs. N, EVqr

Figure 4. Iterations vs. N, EVqrShifted

In Figure 3, the iterations of **QR iteration** grow with , but in Figure 4, the iterations of **shifted QR iteration** grow with .

**4.3. Overall Computational Complexity**

|  |  |  |  |
| --- | --- | --- | --- |
| Complexity | Average Iteration Time | Total Iterations | Overall |
| EVqr |  |  |  |
| EVqrShifted |  |  |  |

Figure 5. CPU time vs. N, EVqr

Figure 6. CPU time vs. N, EVqrShifted

1. **Conclusion**

* The overall computational complexity of EVqr and EVqrShifted are and respectively.
* After A doing shifted QR iterations, the diagonal elements of A from left-top to right-bottom are the eigenvalues with difference between μ from largest to smallest.
* In this homework, the smallest eigenvalue of all the matrices are near to 0.5, so if choose , after matrix A doing shifted QR iterations, the diagonal elements of A from left-top to right-bottom are the largest eigenvalue to smallest eigenvalue.