IMPLEMENTATION OF FRANCIS DOUBLE SHIFT QR FOR REAL SCHUR DECOMPOSITION

Zehao Zhang

Abstract. In this work we implement the QR algorithm for computing the real Schur decomposition of a general real matrix using the double-implicit-shift strategy. Several numerical experiments are also included.

1. Introduction

Solving eigenvalues of a matrix has long been a fundamental topic in the field of numerical computation. One of the oldest methods for the unsymmetric eigenvalue problems is the QR algorithm, whose basic idea is to iterate by

$$A^{(k)} = Q^{(k)}R^{(k)}, \quad A^{(k+1)} = R^{(k)}Q^{(k)}.$$

A shifted version is introduced for faster convergence,

$$A^{(k)} - \mu_k I = Q^{(k)}R^{(k)}, \quad A^{(k+1)} = \left(Q^{(k)}\right)^* A^{(k)}Q^{(k)}$$

where μ is an approximation to an eigenvalue. However, the $O(n^4)$-complexity of the algorithm barricades its application. Then Francis had [5], innovatively proposed his QR algorithm that cut the time complexity down to $O(n^3)$ by converting the matrix into Hessenberg form and applying a genious implicit shift strategy. Also, a double-shift strategy was introduced, where no complex arithmetic is necessary given a real input. A number of theories and improved algorithms on dense, unsymmetric eigenvalue problems are based on his work thereafter.

This paper is devoted to the implementation of the classical Francis double shift QR algorithm and to testing several numerical samples. We shall here assume that readers are familiar with the theory of Francis implicit double shift QR algorithm. For detailed description on how the algorithm works, see [5, 6].

2. Algorithm

Given a real matrix, it is first converted to the upper-Hessenberg form H. Then in each iteration we select an irreducible submatrix $\widetilde{H}$ and perform QR iterations on it. The implicit double shift strategy proposes that, by choosing the conjugated eigenvalues of the 2×2 submatrix in the lower-right corner we can merge the two complex shifts into one real shift. This is done by determining the first column of the resulting $\widetilde{H_2}$ and accordingly generating the whole matrix $\widetilde{H_2}$ by Householder transformations.

The main structure of the QR process can be stated as follows.

Algorithm 1: Francis Implicit Double Shift QR Algorithm

input : Arbitrary matrix A
output: Q, H such that $Q^T AQ = H$ where Q is orthogonal and H quasi-upper triangular.

```
Q,H = To-Upper-Hessenberg(A)
while Not yet converge do
    Determine e such that H(e+1:n, e+1:n) is quasi-diagonal that has deflated.
    Determine s such that H(s:e, s:e) is irreducible upper-Hessenberg.
    Implicit-Double-Shift(Q,H)
    Bulge-Chasing(Q,H)
end
```

The beginning part is to convert the input matrix A to upper-Hessenberg by Householder similarity transformations.

Algorithm 2: To-Upper-Hessenberg

```
Given a n × n matrix A.
Initialize the orthogonal matrix as an identity, Q = I_n.
for i = 1 to n − 1 do
    Compute the Householder transformation P that makes column i of A Hessenberg.
    A(i+1:n, i:n) ← PA(i+1:n, i:n).
    A(1:n, i+1:n) ← A(1:n, i+1:n)P.
    Q(1:n, i+1:n) ← Q(1:n, i+1:n)P.
end
```

Moving on to the while-loop, in each step a double-shift function is called to determine the first column after shifting the lower-right 2×2 block of $H(s:e, s:e)$.

Algorithm 3: Implicit-Double-Shift

```
u = H(e,e) + H(e−1,e−1).
v = H(e,e) * H(e−1,e−1) − H(e,e−1) * H(e−1,e).
x_1 = (H(s,s) * (H(s,s) − u) + H(s,s+1) * H(s+1,s) + v.
x_2 = H(s+1,s) * (H(s,s) + H(s+1,s+1) − u).
x_3 = H(s+1,s) * H(s+2,s+1).
Compute the Householder P that maps [x_1, x_2, x_3] to [×, 0, 0].
H(s:s+2, s:n) ← PH(s:s+2, s:n).
H(1:s+3, s:s+2) ← H(1:s+3, s:s+2)P.
Q(1:n, s:s+2) ← Q(1:n, s:s+2)P.
```

After the double-shift is done, a pack of Hessenberg similarity transformations turn the matrix back to the upper-Hessenberg form to preserve the structure. This is often called bulge chasing [1].

Algorithm 4: Bulge-Chasing

for $i = s+1$ **to** $e-1$ **do**
  Compute the Householder transformation P that makes column i of $H(s:e, s:e)$ Hessenberg.
  $j_1 = \min\{i+2, e\}$, $j_2 = \min\{i+3, e\}$.
  $H(i:j_1, i:n) \leftarrow PH(i:j_1, i:n)$.
  $H(1:j_2, i:j_1) \leftarrow H(1:j_2, i:j_1)P$.
  $Q(1:n, i:j_1) \leftarrow Q(1:n, i:j_1)P$.
end

3. Implementation Details

Here are some details that we need to clarify for implementing an efficient eigensolver.

1. Francis implicit double shift algorithm is designed for real matrix and all the computations are restricted over the field $\mathbb{R}^{n\times n}$. Yet the algorithm also works for complex matrices if the Householder transformation gets slightly modified.

2. Each Householder transformation takes $O(n^2)$ time because it can be computed by $(I - 2vv^T)A = A - 2v(v^TA)$, where there is no need to explicitly construct the Householder matrix. Also, rather than spanning and substracting a rank-1 matrix from A, overwriting A by updating $-x_iy_j$ in each entry shall be the favorable choice. We have discovered by experiment that this cute overwriting trick has sped up the whole process for about 7 times on 1000×1000 matrices.

3. The convergence criterion for a subdiagonal entry we use is

$$|A(i, i-1)| < \mathbf{u} \cdot (|A(i, i)| + |A(i-1, i-1)|) . \tag{1}$$

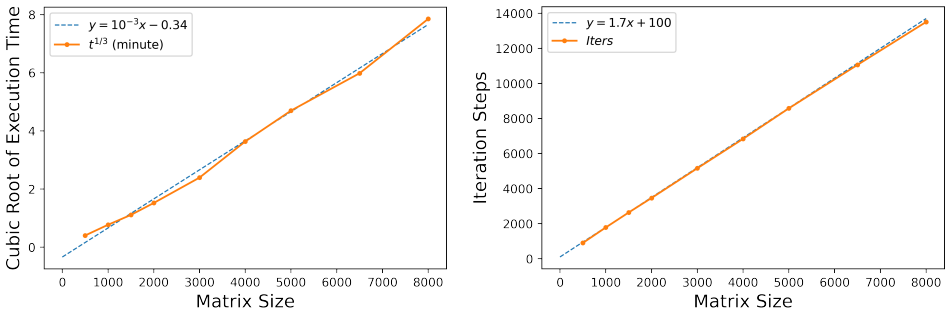
If the condition holds, then the entry $A(i, i-1)$ is set to zero. This works for all random cases we have experimented. But it fails for a particular example demonstrated afterwards where $A(i, i) = A(i-1, i-1) = 0$ but $A(i, i-1) \neq 0$.

4. Numerical Experiments

In this section we denote by $Q^TAQ = T$ the target of the QR algorithm where T is quasi-upper triangular and Q is orthogonal. The test matrices A of different sizes have each entry sampled from the uniform distribution $U(-0.5, 0.5)$ or the normal distribution $N(0, 1)$. All experiments are carried out on an Intel i7-1065G7 1.30GHz CPU and 16GB RAM.

4.1. **Execution Time.** We have in particular conducted experiments on matrices of order ranging from 500 to 8000. The program runs 25 seconds on average for an 1000×1000 matrix while 484 minutes for an 8000×8000. For further analysis, as the algorithm runs in $O(n^3)$ time, the cubic root of the running time is studied and the cubic root is approximately linear as shown in the figure. We have also found that it takes asymptotically $1.7n$

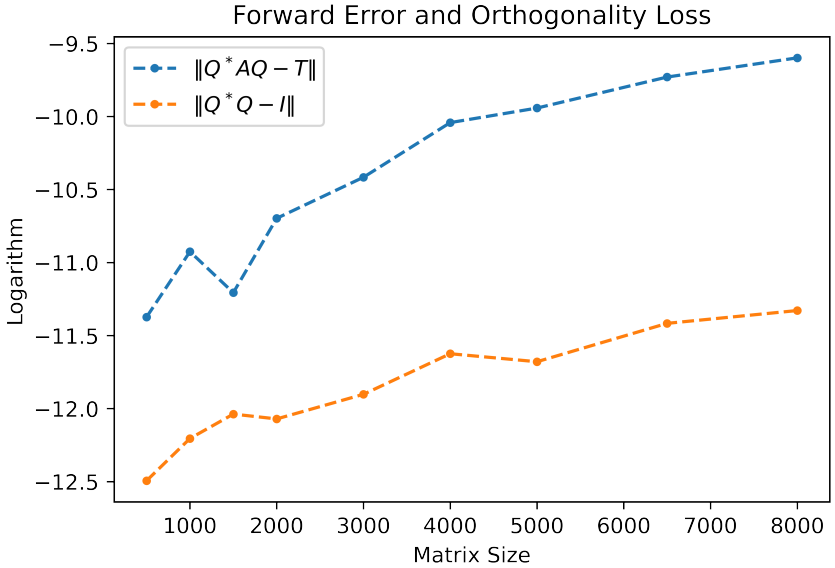
double-shifts for a random $R^{n\times n}$ matrix, indicating that 1.7 double-shifts are required on average for an eigenvalue to deflat.



4.2. **Numerical Stabiliy.** Considering the roundoff error, now we denote by $\hat{Q}$ the orthogonal matrix computed in finite precision arithmetic. The result we obtain can be formulated as $\hat{Q}^T A\hat{Q} \approx \hat{T}$. When the orthogonality loss $\|\hat{Q}^T\hat{Q} - I\|_F = o(\epsilon)$ is small, there exists some 'truly' unitary matrix Q such that $\|\hat{Q} - Q\|_F = o(\epsilon)$ by the polar decomposition. And thus, the Hoffman-Wielandt theorem [2] gives a bound on the error of eigenvalues by

$$\sum_{k=0}^{n} \left|\sigma_k(\hat{Q}^T A\hat{Q}) - \sigma_k(A)\right|^2 \leqslant \|\hat{Q}^T A\hat{Q} - Q^* AQ\|_F^2 = \|A\|_F^2 o(\epsilon^2) \qquad (2)$$

Therefore, if both $\|\hat{Q}^T\hat{Q} - I\|_F$ and $\|\hat{Q}^T A\hat{Q} - \hat{T}\|_F$ are tiny, the eigenvalues of $\hat{T}$ are acceptable. The following figure showcases the logarithm of the Frobenius norms of the two metrics on matrices of different sizes.



4.3. **Convergence.** In order to test the convergence, we have studied 400000 random 4×4 matrices and 10000 random 100×100 matrices. The QR algorithm converged for all these cases. This, unsurprisingly, is supported by the theorem in [7] that an unreduced Hessenberg matrix tends to quasi-upper triangular if and only if there are at most two eigenvalues of even multiplicity and two of odd.

A non-trivial case presented in [3] where the QR algorithm fails to converge is as follows.

$$H(\epsilon) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & \epsilon & 0 \\ 0 & -\epsilon & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

The four eigenvalues of $H(\epsilon)$ are given by $\frac{1}{2}\left(\pm\sqrt{4-\epsilon^2} \pm \epsilon i\right)$ when $\epsilon^2 < 4$.

We have tested that after 10^6 iterations $H(3,2)$ becomes $10^{-2} - 10^{-7}$ for $\epsilon = 10^{-2}$. And for $\epsilon = 10^{-4}$ the Hessenberg matrix seems almost invariant after 10^6 iterations.

When $\epsilon^2 > 4$, the matrix indeed converges to quasi-upper triangular but the program fails to detect the convergence due to the criterion (1).

5. Acknowledgements

The author would like to acknowledge his teacher, Meiyue Shao, for the guidance on numerical linear algebra. And the author appreciates his classmate Zirui Liu, with whom he had discussions over execution time analysis.

References

[1] Bai, Z. and Demmel, J. (1989). On a Block Implementation of Hessenberg Multishift QR Iteration. *Int. J. High Speed Comput.*, **1**, 97-112

[2] Bhatia, R. and Elsner, L. (1994). The Hoffman-Wielandt inequality in infinite dimensions :. *Proceedings Mathematical Sciences*, **104**, 483-494.

[3] Day, D. (1996). How the QR algorithm fails to converge and how to fix it.

[4] Eraña Robles, G. (2018). Implementing the QR algorithm for efficiently computing matrix eigenvalues and eigenvectors.

[5] Francis, J. (1961). The QR transformation: a unitary analogue to the LR transformation. Parts I and II. *Comput. J.* **4**, 265-272, 332-345

[6] G. Golub and C. Van Loan (2013). *Matrix Computations*, 4th ed., The Johns Hopkins University Press, Baltimore, Maryland.

[7] Parlett, B.N. (1968). Global convergence of the basic algorithm on Hessenberg matrices. *Mathematics of Computation*, **22**, 803-817.

School of Data Science, Fudan University
Shanghai, People's Republic of China
E-mail address: 20307130201@fudan.edu.cn