# Implementation of Implicit Shift QR for Real Symmetric Matrices

Weihao Li, School of Data Science, Fudan University

22307140041@m.fudan.edu.cn

## Algorithm

Given a real $n \times n$ symmetric matrix $A$, first convert it to the tridiagonal form $T$. Then in each iteration we select an irreducible submatrix $\tilde{T}$ and perform a step of implicit QR iteration on it. We choose *the Wilkinson shift*, which is the eigenvalue of the lower-right corner $2 \times 2$ submatrix which is closer to the element of the lower-right corner. Wilkinson(1968) has shown that Wilkinson shift and the shift $\tilde{T_{nn}}$ both is cubically convergent, but gives heuristic reasons why the Wilkinson shift is preferred[4]. And the implicit shift is done by determining the first column of resulting $\tilde{T_2}$ and accordingly generating the whole matrix $\tilde{T_2}$ by Givens rotations. The main structure of the QR process can be stated as follows.

```
Algorithm 1: Implicit Shift Symmetric QR Algorithm
Input  : real symmetric matrix $A$
Output: $D$, $Q$ (if needed) such that $D$ is diagonal and $Q$ is real orthogonal.
1 $Q$,$T$= ToTridiagonal ($A$)
2 while not yet converge do
3     Determine $p, q$ that $T(p:q, p:q)$ is irreducible tridiagonal
4     ImplicitShift ($Q$, $T$, $p, q$)
5 end
```

The beginning part is to convert $A$ into the tridiagonal form by Householder transformations.

```
Algorithm 2: ToTridiagonal
Input  : real symmetric matrix $A$.
Output: real orthogonal $Q$, tridiagonal $T$.
1 Initialize the orthogonal matrix as an identity, $Q = I_n$
2 for $i = 1$ to $n - 1$ do
3     Compute the Householder transformation $H$ that makes column $i$ and
       row $i$ of $A$ become tridiagonal
4     $A(i+1:n, i:n) \leftarrow HA(i+1:n, i:n)$
5     $A(i:n, i+1:n) \leftarrow A(i:n, i+1:n)H$
6     $Q(i:n, i+1:n) \leftarrow Q(i:n, i+1:n)H$
7 end
```

Then we preform deflations and implicit shift in each step of the while-loop. The so called "bulge chasing" is also encapsulated in the implicit shift function.

**Algorithm 3:** ImplicitShift

```
Input: $Q$, $T$, $p, q$
1 $\mu \leftarrow$Compute the Wilkinson shift with $T(q-1:q, q-1:q)$
2 $x, z \leftarrow T(1,1) - \mu, T(2,1)$
3 for $i = p$ to $q-1$ do
4     $c, s \leftarrow$GivensRotation (v)
5     $T \leftarrow G_i T G_i^T$, where $G_i = G(i, i+1, c, s)$
6     if $i < q-1$ then
7         $x, z \leftarrow T(i+1, i), T(i+2, i)$
8     end
9 end
```

## Implementation Details

1. The implementation is based on *python 3.11*, with packages *numpy 1.26.0* and *numba 0.58.1*.
2. This symmetric QR algorithm is designed for real symmetric matrices, and all the computations are restricted over $R^{n \times n}$.
3. Each Householder transformation takes $O(n^2)$ time because we use the right hand side of $(I - 2vv^T)A = A - 2v(v^T A)$ to compute it, avoiding explicitly forming the Householder matrix.
4. Two 1d arrays, one with size $n$ and the other with size $n-1$, are used to save the diagonal and subdiagonal entries of $T$, which dramatically save the time and space usage during the QR iterations, and the algorithm return $D$ in the 1d array form too.
5. The convergence criterion for a subdiagonal entry used is

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

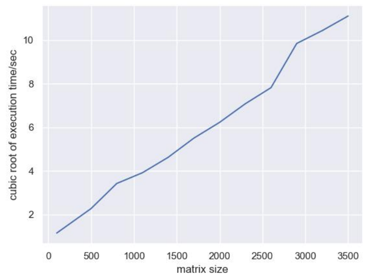
   The entry $T(i, i-1)$ is set to zero if the condition holds.

## Numerical Experiments

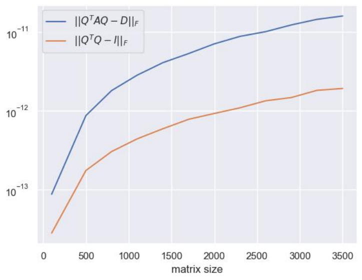
In this section we denote by $Q^T A Q = D$ the goal of our QR algorithm in which $D$ is diagonal and $Q$ is real orthogonal. The test matrices $A$ of different sizes have element-wise sampled from $(0, 1)$ with numpy's default rng. All experiments are carried out on an Apple M2 CPU and 16GB RAM.

### Execution time

We have in particular conducted experiments on matrices of order ranging from 100 to 3500. The cubic root of running time is approximately linear of the matrix size as shown in the figure, which is corresponding to the $O(n^3)$ asymptotic time of the algorithm.

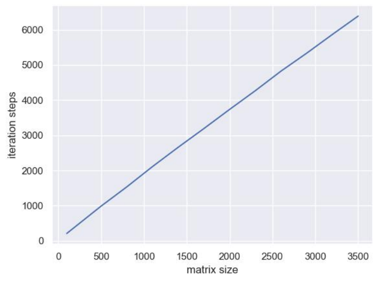


(a) execution times

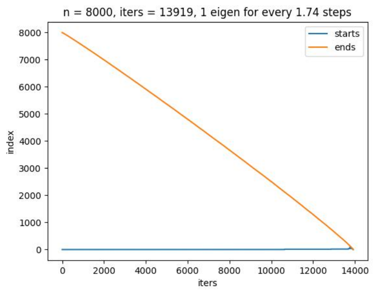


(b) numerical stability

Figure 1: Execution time and Numerical Stability



(a) Iteration steps is linear of matrix size



(b) Convergence history of an large matrix

Figure 2: Convergence Speed

## Numerical Stability

We measure the numerical stability by the orthogonal loss of $Q$: $||Q^TQ - I||_F$ and the forward error $||Q^TAQ - D||_F$.

## Convergence Speed

All the random generated matrices in the experiment converged, including matrices for drawing figure above and additional 100 random $100 \times 100$ matrices. We have visualized the linear connection between matrix sizes and iteration steps in figure below. We have also visualize the convergence history of an $8000 \times 8000$ matrix, in term of the start and end index of the irreducible submatrix, indicating that 1.74 Wilkinson shifts are required on average for one subdiagonal entry to deflat.

## Acknowledgements

The author would like to express appreciation to his teacher, Prof. Meiyue Shao, for his guidance and support provided through the numerical algorithm course. And the author would also appreciate TA Zhenyu Xu, who offer timely help to me in fixing a problem in the implementation. This project also get helpful inputs from student Zhang's project on the asymmtric QR implementation[3].

## References

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

[2] Shufang Xu, Li Gao, Pingwen Zhang(2013). Numerical Linear Algebra, 2th ed., Peking University Press, Beijing.

[3]https://github.com/ForeverHaibara/Fudan-Courses/tree/main/%E6%95%B0%E5%80%BC%E7%AE%97%E6%B3%95%E4%B8%8E%E6%A1%88%E4%BE%8B%E5%88%86%E6%9E%90/Project%201

[4]J.H. Wilkinson (1968). "Global Convergence of Tridiagonal QR Algorithm With Origin Shifts," *Lin. Alg. Applic. 1*, 409-420.