QR method for symmetric matrix

Qin Ma 21307110024

December 3, 2023

Abstract

The QR algorithm computes a Schur decomposition of a matrix. It's espeicially efficient in being applied to dense matrices.

For a symmetric matrix, appropriately refining the QR algorithm can yield the eigen decomposition of the matrix.

The algorithm's accuracy, convergence behaviour, and sensitivity are analyzed.

Key Words:QR method; symmetric matrix; numerical analysis.

Contents

1 Theory and Algorithm

1.1 Eigen Decomposition

Suppose we have a symmetric matrix A and there exists an orthogonal matrix Q such that $D = QAQ^T$, where D is a diagonal matrix, hence $A = Q^T DQ$, the equation is called the eigen decomposition of a symmetric matrix. And Q is a matrix consisting of a set of orthogonal eigenvectors of A. The elements of D are eigenvalues of A.

The decomposition is the foundation of symmetric QR algorithm.

1.2 Tridiagonalization

Assuming that the upper Hessenberg decomposition of a real symmetric matrix A is $Q^T AQ = T$, it is easy to verify that T is a symmetric tridiagonal matrix.

Using householder reflections can transform A to T.

The householder reflection is defined by $H = I - \beta vv^T$. In the case of $n = k$, $H_k A_{k-1} H_k$ can set the elements outside the main diagonal and the secondary diagonal of the matrix A_{k-1} to 0 for the first row and the first column, hence $n - 1$ iterations are manage to tridiagonalize a symmetric matrix. Proof: See [2].

Suppose

$$\tilde{H}_k = I - \beta vv^T, v \in R^{n-k},$$

Easy to obtain that

$$\tilde{H}_k A_{k-1} \tilde{H}_k = A_{k-1} - vw^T - wv^T,$$

$$w = u - \frac{1}{2}\beta(v^T u)v, u = \beta A_{k-1} v.$$

We can use this equation to compute the tridiagonalization.

Listing 1: hessenberg.m

```
function [alpha,gamma,U_0]=hessenberg(A)

n=size(A,1);
U_0=eye(n,n);
d=zeros(n-2);

for k=1:n-2
    [v,beta]=house(A(k+1:n,k));
    u=beta*A(k+1:n,k+1:n)*v;
    w=u-(beta*u'*v/2)*v;
    A(k+1,k)=norm(A(k+1:n,k),2);
    A(k,k+1)=A(k+1,k);
    A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-v*w'-w*v';

    d(k)=beta;
    A(k+2:n,k)=v(2:n-k);
end

v=zeros(n,1);
for k=1:n-2
    v(n+1-k:n,1)=A(n+1-k:n,n-1-k);
    v(n-k,1)=1;
    U_0=(eye(n,n)-d(n-1-k)*(v*v'))*U_0;
end
```

This algorithm requires $\frac{4}{3}n^3$ multiplication operations to compute the tridiagonalization, and another $\frac{4}{3}n^3$ multiplication operations to accumulate the transformation matrices.

Applying the QR algorithm to a tridiagonal matrix can significantly reduce the number of QR iterations.

1.3 Implict Symmetric QR Iteration

The shift technique can make the convergence rate to the diagonal matrix cubic, by applying it, we can accelerate QR iterations.

Suppose

$$T_k(n-1:n, n-1:n) = \begin{bmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{bmatrix}$$

Take the shift as the one closer to α_n among the two eigen values.

$$\mu = a_n + d - \text{sign}(d)\sqrt{d^2 + b_{n-1}^2}$$

$$d \equiv (a_{n-1} - a_n)/2$$

Wilkinson[1] proved that this shift strategy has cubic convergence and gave reasons why this shift is better than $\mu = \alpha_n$.

Applying the method in the textbook "Numerical Linear Algebra"[2], we implicitly complete one symmetric QR iteration using Givens rotation. The algorithm can be found in page 209. However, we can improve it by reducing the computation operations.

It's unnecessary to compute $T = G_k T G_k^T$ explicitly. For instance, in case of $n = 4, k = 2$.

$$T = \begin{bmatrix} x & x & y & 0 \\ x & x & x & 0 \\ y & x & x & x \\ 0 & 0 & x & x \end{bmatrix}$$

And

$$T = G_2 T G_2^T = \begin{bmatrix} x & \hat{x} & 0 & 0 \\ \hat{x} & \hat{x} & \hat{x} & y \\ 0 & \hat{x} & \hat{x} & \hat{x} \\ 0 & y & \hat{x} & x \end{bmatrix}$$

Therefore, only the matrix of two dimensions in the center and four subdiagonal elements need to be updated. This technique can be applied in higher dimensional matrices.

In practice, the matrix T is stored by two vectors: alpha for diagonal elements and beta for subdiagonal elements.

Listing 2: QRstep.m

```
function [alpha,beta,Q]=wilkinson_QR_step(alpha,beta)
n=length(alpha);
Q=eye(n,n);
d=(alpha(n-1)-alpha(n))/2;
mu=alpha(n)-beta(n-1)^2/(d+sign(d)*sqrt(d^2+beta(n-1)^2));
x=alpha(1)-mu;
z=beta(1);
[c,s]=givens(x,z);
G=[c,s;-s,c];
Q(1:n,1:2)=Q(1:n,1:2)*G';
if n==2
        T=[alpha(1),beta(1);beta(1),alpha(2)];
        T=G*T*G';
        alpha(1)=T(1,1);
        beta(1)=T(2,1);
        alpha(2)=T(2,2);
else
```

```
        T=[alpha(1),beta(1);beta(1),alpha(2)];
        T=G*T*G';
        gamma=s*beta(2);
        beta(2)=c*beta(2);
end

for k=2:n-1
        x=T(2,1);
        z=gamma;
        [c,s]=givens(x,z);
        G=[c,s;-s,c];
        Q(1:n,k:k+1)=Q(1:n,k:k+1)*G';

        if k<n-1
                alpha(k-1)=T(1,1);
                beta(k-1)=c*T(2,1)+s*gamma;
                gamma=s*beta(k+1);
                beta(k+1)=c*beta(k+1);
                T=[T(2,2),beta(k);beta(k),alpha(k+1)];
                T=G*T*G';
        else
                alpha(k-1)=T(1,1);
                beta(k-1)=c*T(2,1)+s*gamma;
                T=[T(2,2),beta(k);beta(k),alpha(k+1)];
                T=G*T*G';
                alpha(k)=T(1,1);
                alpha(k+1)=T(2,2);
                beta(k)=T(2,1);
        end
```

```
end
end
```

The transformation matrix Q is also computed in the algorithm.

1.4 Deflation

Usually, we use $|t_{i+1,i}| \leq u * (|t_{ii}| + |t_{i+1,i+1}|)$ as the judgemental signs of convergence (u is machine precision). Proof:See [2].

Since the matrix usually converges from the tails (considering our shift strategy), we obtain the first algorithm.

Listing 3: convergenceJduge1.m

```
m=n;
while m>1
    [alpha(1:m),gamma(1:m-1),~]=wilkinson_QR_step(alpha(1:
       m),gamma(1:m-1));
    if abs(gamma(m-1))<=u*(abs(alpha(m-1))+abs(alpha(m)))
        m=m-1;
    end
end
```

An issue not treated in this algorithm is deflation. Deflation is of big practical importance. Let us consider the following 5 * 5 situation.

$$T = \begin{bmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & 0 & & \\ & 0 & a_3 & b_4 & \\ & & b_4 & a_4 & b_5 \\ & & & b_5 & a_5 \end{bmatrix}$$

The shift for the next step is determined from elements a_4, a_5, and b_5. The first givens rotation is determined from the shift and the elements a1 and b1. The implicit shift algorithm then chases the bulge down the diagonal. In this particular situation, the procedure finishes already in row/column 3 because $b_3 = 0$.

This shift does not improve convergence.

We can solve this problem by finding the largest irreducible tridiagonal matrix in T.

Listing 4: convergenceJduge2.m

```
while q<n
    for k=1:n-1
        if abs(gamma(k))<=u*(abs(alpha(k))+abs(alpha(k+1))
            )
             gamma(k)=0;
        end
    end
    [p,q]=Find_Reducible(alpha,gamma);
    if q<n
        [alpha(p+1:n-q),gamma(p+1:n-q-1),G]=
            wilkinson_QR_step(alpha(p+1:n-q),gamma(p+1:n-q
            -1));
        Q(1:n,p+1:n-q)=Q(1:n,p+1:n-q)*G;
    end
end
```

2 Numerical Experiments and Performance

2.1 Deflation

We initialise the matrix A in the following way.

$$a = \begin{bmatrix} 2 & 4 & 6 & 8 & 7 \end{bmatrix}$$

$$A = a' * a = \begin{bmatrix} 4 & 8 & 12 & 16 & 14 \\ 8 & 16 & 24 & 32 & 28 \\ 12 & 24 & 36 & 48 & 42 \\ 16 & 32 & 48 & 64 & 56 \\ 14 & 28 & 42 & 56 & 49 \end{bmatrix}$$

Although the matrix has only 5 dimensions. Our first algorithm does not work because deflation occurs.

$$gamma = \begin{bmatrix} 2.80000000000000e - 322 \\ -5.73437552744881e - 23 \\ -1.56800960362859e - 23 \\ -1.78319247829147e - 27 \end{bmatrix}$$

Deflation occurred in the first subdiagonal element unfortunately.

After testing, the first algorithm is able to handle randomly generated matrices up to order 120, with numerical underflow invalidating the algorithm at higher orders.

The second algorithm is good enough to avoid deflation.

The results can be verified by running test1.m

2.2 Complexity

Changing the matrix order to get the corresponding running time, we can observe the efficiency of the algorithm.

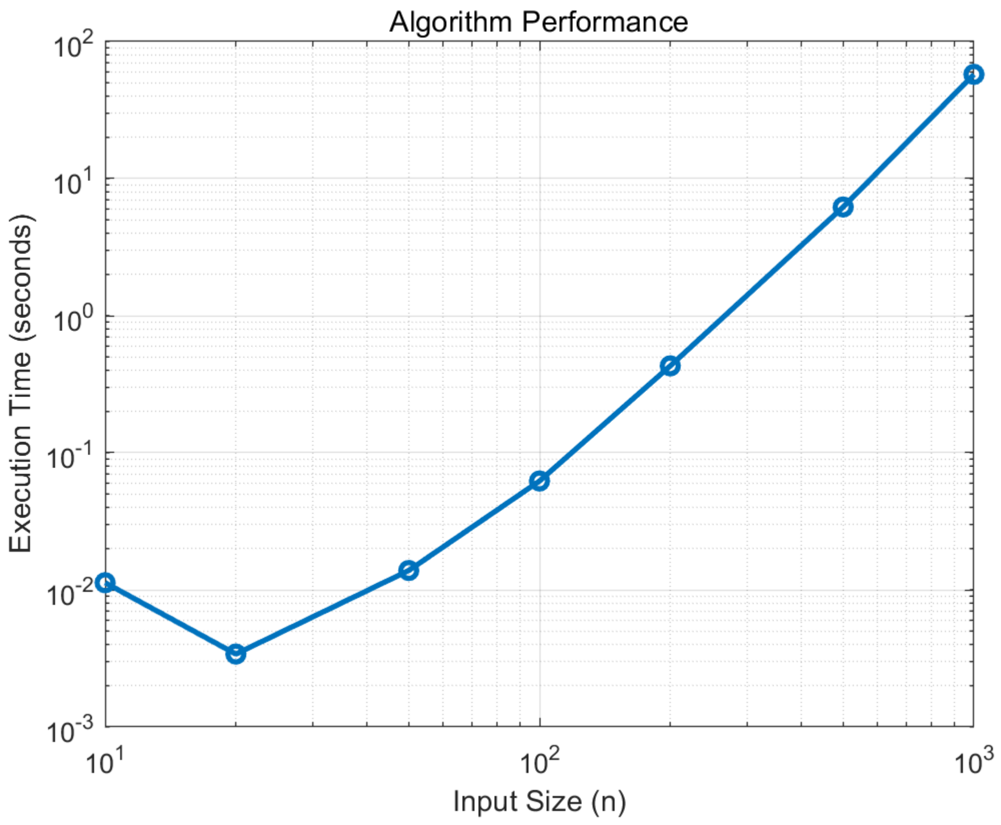


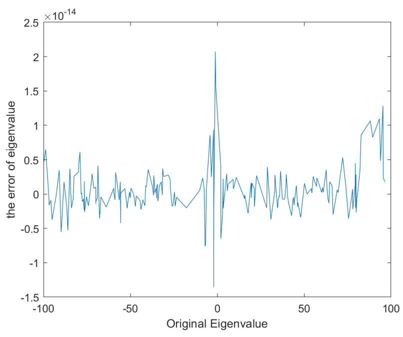
Figure 1: Time Complexity

And it takes roughly less than 60 seconds to compute a 1000-order matrix.

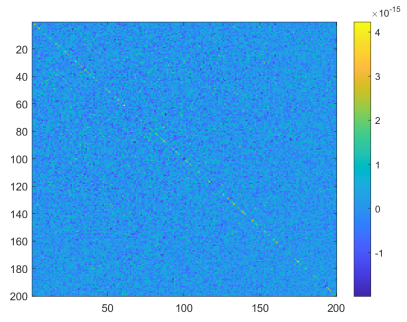
2.3 Accuracy

Take a 200-order matrix as an example.

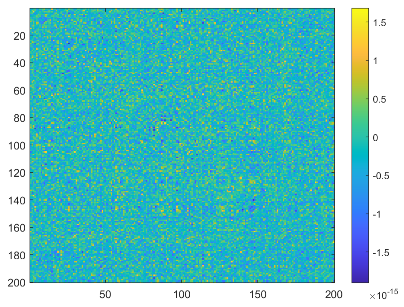
As shown in the figure below. The tolerance is one to two orders of magnitude higher than machine precision. (The machine precision of matlab is 2.2204e-16).



(a) Eigenvalue



(b) Eigenvector



(c) The change of A

Figure 2: Error Analysis

2.4 Convergence Behaviour

Take a 5-order matrix as an example.

The gamma obtained from each iteration is stored in the convergence history matrix to annalysis convergence behaviour.

$$H = \begin{bmatrix} 70.44 & -21.25 & 12.94 & 12.74 & 12.56 & 12.13 & 11.48 & 10.09 & 8.53 & 0 \\ 28.92 & 75.71 & -34.63 & -7.67 & 2.0056 & 0.74 & 0.29 & -3.4e-08 & 0 & 0 \\ 67.13 & -32.81 & -31.62 & 29.81 & -18.97 & 0.0008 & 0 & 0 & 0 & 0 \\ -36.39 & -9.44 & -0.18 & 9.3e-06 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A total of 10 iterations are manage to solve a 5-order matrix. On average, two iterations yield an eigenvalue.

The convergence behaviour is even better than 3rd order convergence.

References

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

[2] Shufang Xu. *Numerical linear Algebra*[M]. PEKING UNIVERSITY PRESS.