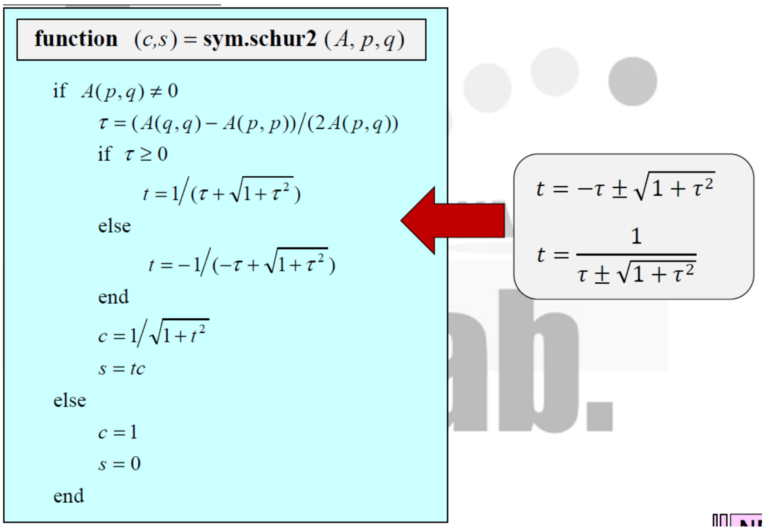
신호 처리를 위한 행렬 계산 Programming Assignment 6

20180490 이재헌

# [Code Implementation]

## 1. symSchur2.m

강의에서 배웠던 2x2 symmetric Schur decomposition 알고리즘은 아래와 같다.




알고리즘을 반복할수록 $A(p,q)$가 작아지기 때문에 $\tau$는 점점 큰 값이 나온다. $t = -\tau \pm \sqrt{1+\tau^2}$의 식을 쓰면 $\tau$가 커졌을 때 유효숫자의 손실이 커진다. 따라서 $t = \frac{1}{\tau \pm \sqrt{1+\tau^2}}$ 의 식을 쓰고, 유효숫자의 보존을 위해서 $\tau$가 양수와 음수일 때 분모가 커지도록 하는 식을 골라서 사용한다.

위의 식을 코드로 구현하면 아래와 같다.

```
if A(p,q) ~= 0
    tau = (A(q,q)-A(p,p))/(2*A(p,q));
    if tau >= 0
        t = 1/(tau + sqrt(1 + tau^2));
    else
        t = -1/(-tau + sqrt(1 + tau^2));

    end
    c = 1/sqrt(1 + t^2);
    s = t*c;
else
    c = 1;
    s = 0;
end
```

## 2. twosideJacobi.m

강의에서 배웠던 the cyclic-by-row Jacobi algorithm은 아래와 같다.

**Algorithm 8.4.3 (Cyclic Jacobi)**

$V = I_n$; $\text{eps} = \text{tol}\|A\|_F$

while $\text{off}(A) > \text{eps}$

  for $p = 1 : n-1$

    for $q = p+1 : n$

      $(c, s) = \textbf{sym.schur2}\ (A, p, q)$

      $A = J(p,q,\theta)^T AJ(p,q,\theta)$

      $V = VJ(p,q,\theta)$

    end

  end

end

While문 안의 코드를 구현하면 다음과 같다.

```
while offA>delta
    for p=1:n-1
        for q=p+1:n
            [c,s] = symSchur2(A,p,q);
            %%%%%%% [Write down your code in the following block]
            J = eye(n);
            J(p,p) = c;
            J(q,p) = -s;
            J(p,q) = s;
            J(q,q) = c;

            A = J'*A*J;
            V = V*J;
            %%%%%%% [Write down your code in the following block]
            offA = norm(A-diag(diag(A)),'fro');
            offVals = [offVals;offA];
        end
    end
end
```

Jocobi rotaion matrix를 givens rotation에서 사용했던 것과 비슷하게 설정해주고 이를 A의 양쪽에 곱해주면 A를 Schur Decomposition 할 수 있다.

## [Test.m Screenshot]

Test.m의 명령창 실행 결과는 아래와 같다.

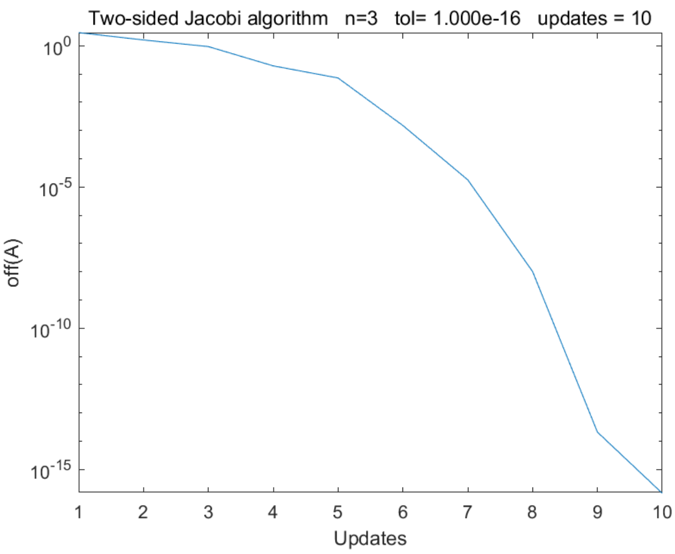
```
Test case
A =
   -1.8945    1.7292   -0.6615
    1.7292    2.6340    0.9228
   -0.6615    0.9228    1.6676

||A - V*D*V^T|| = 0.00000

||I - V^T*V|| = 0.00000

D =
   -2.6784   -0.0000   -0.0000
   -0.0000    3.4812         0
   -0.0000   -0.0000    1.6043

||A - USV^T|| = 0.00000

S =
    3.4812         0         0
         0    2.6784         0
         0         0    1.6043
```

A와 V'DV를 비교하는 것이 아니라 A와 VDV'을 비교하도록 코드의 오류를 수정하였다.

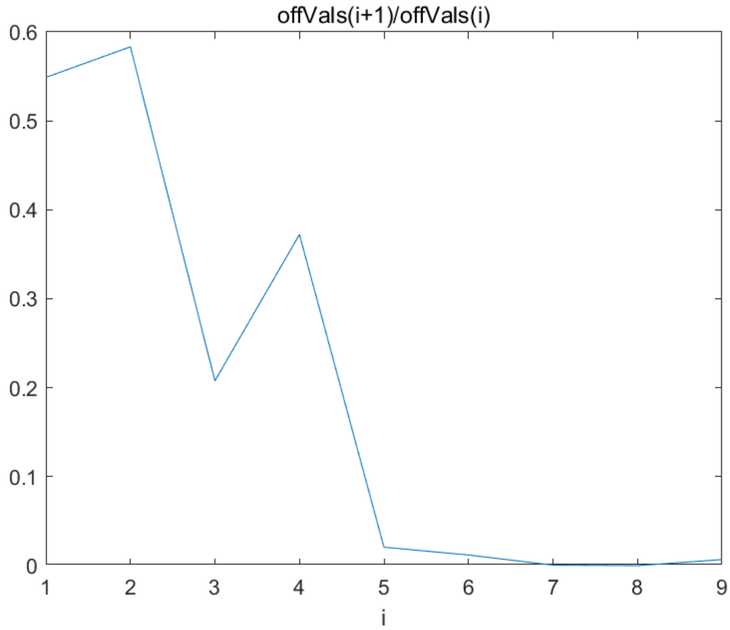
실행 결과, VDV'이 A와 같게 나왔고 V도 orthogonal하게 나왔다. A의 Schur decomposition이 잘 이루어진 것을 확인하였다.

D의 diagonal element들의 절댓값이 SVD를 통해 구한 Singular value와 같게 나왔다. A가 대칭이므로 SVD를 하면 U = V로 나온다. 그러면 A = VDV'이므로 Schur decomposition과 SVD가 본질적으로는 같은 것이라고 할 수 있다.

또한 Cyclic Jacobi algorithm의 수렴 속도 그래프는 아래와 같았다.



수렴의 속도를 알아보기 위해서 offVals의 요소별 비율을 확인하였다.




위 그래프의 가로축은 요소의 인덱스이고, 세로축은 $\frac{offVals(i+1)}{offVals(i)}$ 이다.

$$\frac{offVals(i+1)}{offVals(i)} \leq C \cong 0.6$$

Therefore, it converges at a linear rate.