신호 처리를 위한 행렬 계산 Programming Assignment 5

20180490 이재헌

[Code Implementation]

1. givens.m

2x2 givens rotation matrix는 Rotation matrix R_θ에 $-\theta$를 대입한 행렬과 같다.

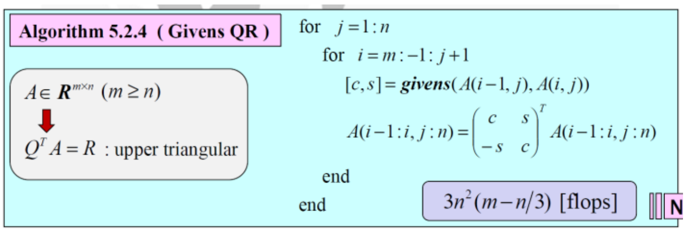
$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

$\cos(-\theta) = cos\theta, \sin(-\theta) = -sin\theta$임을 이용해 아래와 같이 c, s를 구할 수 있다.

```
% if arg(a,b) = t, c = cos(-t) = cos(t), s = sin(-t) = -sin(t)
r = sqrt(a^2 + b^2);
c = a/r;
s = -b/r;
```

2. GivensQR.m

강의에서 배웠던 Givens QR 알고리즘은 아래와 같다.



이를 매트랩 코드로 구현하면 다음과 같다.

```
[m,n] = size(A);
Q = eye(m);
for j=1:n
    for i=m:-1:j+1
        [c,s] = givens(A(i-1,j),A(i,j));
        G = eye(m);
        G(i-1:i,i-1:i) = [c s; -s c];
        Q = Q*G; % Q = G1*G2*...*Gn
        A(i-1:i,j:n) = [c s; -s c].'*A(i-1:i,j:n);
    end
end
R = A;
```

결국 A가 upper triangular matrix가 되는 것이므로 R = A이다. $G_n * \ldots * G_1 A = R$ 이므로 $Q = (G_n * \ldots * G_1)^T = G_1 * \ldots * G_n \quad since\ G_i\ s\ are\ symmetric\ and\ orthogonal$

따라서 코드처럼 Identity matrix 오른쪽에 m x m givens rotation matrix를 곱해가며 Q를 구할 수 있다.

3. house.m

강의에서 배웠던 householder matrix를 구하는 알고리즘은 다음과 같다.

Function $[v, \beta] =$ **Householder**(x)

$n =$ **length**(x);

$\sigma = x(2:n)^T x(2:n)$; $v = \begin{pmatrix} 1 \\ x(2:n) \end{pmatrix}$;

if $\sigma = 0$, $\beta = 2$

else

if $x(1)=0$, $\mu = \sqrt{\sigma}$ and $v(1) = \mu$

else, $\mu = \sqrt{x(1)^2 + \sigma}$

$v(1) = x(1) + \mu$

end

$\beta = 2|v(1)|^2/(\sigma + |v(1)|^2)$

$v = v/v(1)$

end

Real number version ➡ $3n$ [flops]

이 과정에서 v(1)을 정해줄 때, x(1)과 u의 부호를 같게 맞춰주면 x의 유효숫자를 더 잘 보존할 수 있다. 그 과정이 강의에도 아래의 그림처럼 나와있다.

if $x_1 > 0, \alpha = \|x\|_2$

else $\alpha = -\|x\|_2$

$v = x + \alpha e_1$ $\quad v_1 = x_1 + \alpha$

이 과정을 포함해 매트랩 코드를 작성하면 다음과 같다.

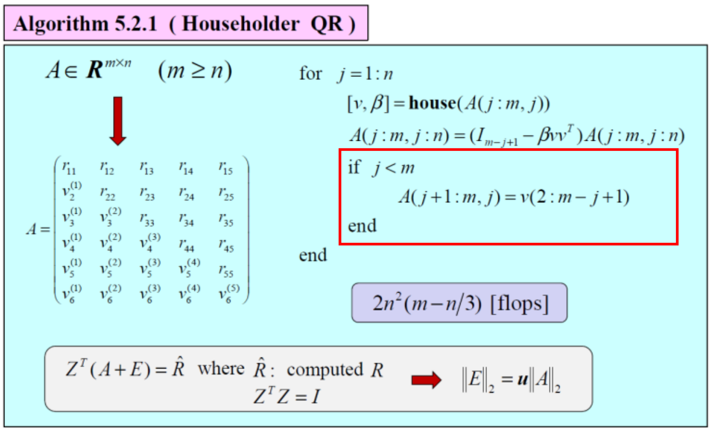
```
n = length(x);
s = x(2:n).'*x(2:n);
v = [1; x(2:n)];
u = sqrt(s + x(1)^2);
```

```
if s == 0
    beta = 2;
else
    if x(1) > 0 % to preserve significant digits
        v(1) = x(1) + u;
    else
        v(1) = x(1) - u;
    end
    beta = 2*abs(v(1))^2/(s + abs(v(1))^2);
    v = v/v(1);
end
```

강의 자료의 코드를 조금 수정해 구현하였다.

4. HouseholderQR.m

강의에 소개된 Householder QR decomposition 알고리즘은 다음과 같다.



코드의 중간 부분에 빨간색으로 표시한 부분은 householder vector를 A에 저장하는 부분인데, 이는 Q, R을 구하는 과정에서는 필수가 아니므로 생략하였다. 이 알고리즘을 매트랩 코드로 구현하면 다음과 같다.

```
[m,n] = size(A);
Q = eye(m);
for j=1:n-1
    [v, beta] = house(A(j:m,j));
    A(j:m,j:n) = (eye(m-j+1) - beta*(v*v.')) * A(j:m,j:n);
    H = eye(m);
    H(j:m,j:m) = eye(m-j+1) - beta*(v*v.');
    Q = Q*H;
end
R = A;
```

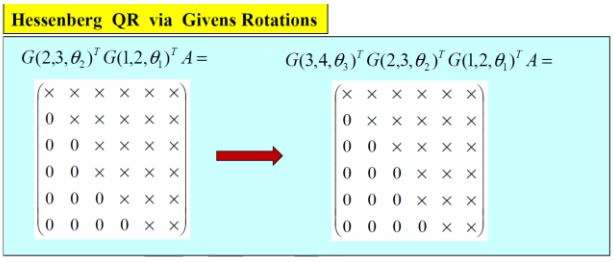
결국 A가 upper triangular matrix가 되는 것이므로 R = A이다. $H_n * \ldots * H_1 A = R$ 이므로 $Q = (H_n * \ldots * H_1)^T = H_1 * \ldots * H_n$ $\quad$ *since* H_i *s are symmetric and orthogonal*

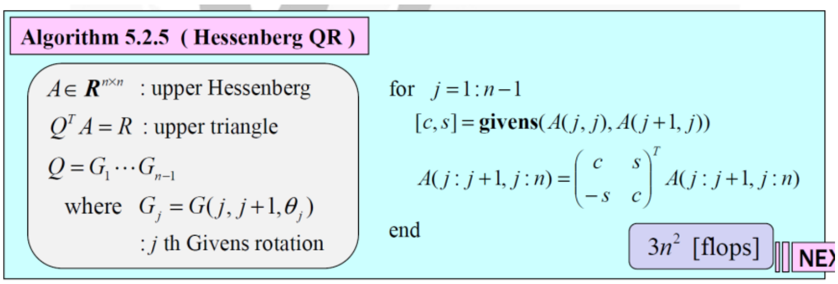
따라서 코드처럼 Identity matrix 오른쪽에 m x m givens rotation matrix를 곱해가며 Q를 구할 수 있다.

5. HessenbergQR.m

주어진 Hessenberg matrix를 QR decomposition하는 코드이다. 코드의 설명에는 A가 임의의 m x n matrix라고 되어있었지만 이는 오타인 것 같다. test 코드에서는 input으로 Hessenberg matrix를 대입하기에 주어지는 input이 Hessenberg matrix라고 가정하고 코드를 작성하였다.

Hessenberg matrix의 QR decomposition은 Givens rotation을 n-1번 하면 얻을 수 있다. 강의에서 배운 알고리즘은 다음과 같다.





이를 매트랩 코드로 구현하면 다음과 같다.

```
n = length(A(:,1));
Q = eye(n);
for j=1:n-1
    [c,s] = givens(A(j,j),A(j+1,j));
    A(j:j+1,j:n) = [c s; -s c].'*A(j:j+1,j:n);
    G = eye(n);
    G(j:j+1,j:j+1) = [c s; -s c];
    Q = Q*G;
end
R = A;
```

Givens rotation을 n-1번 적용하여 Q, R을 구하였다.

[Test.m Screenshot]

1. Test case with the built-in QR function

```
Test case
A =
   -0.9472    1.3170    0.8338    0.4571
    0.5401   -0.4056    0.6044    0.9217
   -0.2166   -0.4449   -0.1067   -0.1973
    1.1890    1.3284    0.0030    0.0755

QR Factorization using MATLAB build-in function 'qr'
Q =
   -0.5819    0.7098    0.3848    0.0974
    0.3318   -0.2286    0.9146    0.0350
   -0.1331   -0.2182   -0.0432    0.9658
    0.7305    0.6296   -0.1167    0.2376

R =
    1.6278    0.1285   -0.2682    0.1213
         0    1.9609    0.4788    0.2043
         0         0    0.8779    1.0186
         0         0         0   -0.0959
```

2. Householder QR

```
QR Factorization based on Householder Vector
Q =
   -0.5819    0.7098    0.3848    0.0974
    0.3318   -0.2286    0.9146    0.0350
   -0.1331   -0.2182   -0.0432    0.9658
    0.7305    0.6296   -0.1167    0.2376

R =
    1.6278    0.1285   -0.2682    0.1213
    0.0000    1.9609    0.4788    0.2043
   -0.0000   -0.0000    0.8779    1.0186
    0.0000    0.0000   -0.0000   -0.0959

||Q'Q - I|| = 0.00000
||A - QR|| = 0.00000
```

3. Givens QR

```
QR Factorization based on Givens Rotations
Q =
   -0.5819    0.7098    0.3848   -0.0974
    0.3318   -0.2286    0.9146   -0.0350
   -0.1331   -0.2182   -0.0432   -0.9658
    0.7305    0.6296   -0.1167   -0.2376

R =
    1.6278    0.1285   -0.2682    0.1213
    0.0000    1.9609    0.4788    0.2043
         0         0    0.8779    1.0186
   -0.0000         0         0    0.0959

||Q'Q - I|| = 0.00000
||A - QR|| = 0.00000
```

4. Hessenberg QR

```
QR Factorization for an upper Hessenberg matrix based on Givens Rotation.
Hessenberg matrix, H
   -0.9472   -0.8115    0.1798   -1.3956
   -1.3238    0.8331   -0.3752    0.8211
         0   -0.3336   -0.9525    0.0796
         0         0    0.8738   -0.3174

Q =
   -0.5819   -0.7808    0.1726   -0.1483
   -0.8132    0.5587   -0.1235    0.1061
         0   -0.2798   -0.7280    0.6259
         0         0    0.6519    0.7583

R =
    1.6278   -0.2053    0.2005    0.1444
   -0.0000    1.1924   -0.0834    1.5261
         0         0    1.3404   -0.6070
         0         0         0    0.1033

||Q'Q - I|| = 0.00000
||H - QR|| = 0.00000
```

- Householder QR, Givens rotation QR, Hessenberg QR 방법 모두 R이 upper triangular matrix가 되었다.
- Householder QR, Givens rotation QR, Hessenberg QR 방법 모두 $||Q^TQ - I|| = 0$이 되었으므로 Q가 orthogonal 함을 확인할 수 있다.
- Householder QR, Givens rotation QR, Hessenberg QR 방법 모두 $||A - QR|| = 0$이 되었으므로 QR factorization이 성공적으로 이루어졌음을 확인할 수 있다.
- Givens rotation의 Q의 마지막 column vector가 built-in function으로 구한 Q의 마지막 column vector와 부호가 다르다.

  이는 R(n,n)과 부호가 다른 것으로 상쇄가 되었다.