프로젝트

조별로 두 가지 Gram-Schmidt 알고리즘을 파이썬 코드로 구현해서 jinseong@snu.ac.kr (박진성 조교) 이메일로 10월 15일까지 ipynb 파일을 보내주세요. 아래 코드는 Matlab코드로 이를 참고하셔서 구현하셔도 됩니다.

프로젝트-1

1. implement the following MATLAB programs **clgs** in Python.

[기존 그람-슈미츠 알고리즘 Classical Gram-Schmidt orthogonalization]

```
function [Q, R] = clgs(A)
[m, n] = size(A);
V=A; Q=eye(m,n);
R=zeros(n,n);
for j=1:n
  for i=1:j-1
       R(i,j)=Q(:,i)'*A(:,j);
       V(:,j)=V(:,j)-R(i,j)*Q(:,i);
  end
  R(j,j)=norm(V(:,j));
  Q(:,j)=V(:,j)/R(j,j);
end
```

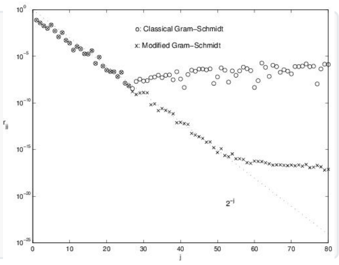
2. Implement the following MATLAB programs **mgs** in Python.

[수정 그람-슈미츠 알고리즘 Modified Gram-Schmidt orthogonalization]

```
function [Q, R] = mgs(A)
 [m, n] = size(A);
 Q = A;
 R=zeros(n,n);
 for i = 1:n-1
    R(i,i)=norm(Q(:,i));
    Q(:,i)=Q(:,i)/R(i,i);
    R(i,i+1:n)=Q(:,i)'*Q(:,i+1:n);
    Q(:,i+1:n)=Q(:,i+1:n)-Q(:,i)*R(i,i+1:n);
 end
 R(n,n)=norm(Q(:,n));
 Q(:,n)=Q(:,n)/R(n,n);
```

3. Implement the following MATLAB code in Python and test it. Plot the graph(right).

```
[U,X]=qr(randn(80));
[V,X]=qr(randn(80));
J=1:80; S=diag(2.^(-J));
A=U*S*V;
[Qc,Rc]=clgs(A);
[Qm,Rm]=mgs(A);
```



Implementing the MATLAB program **semilogy** in Python, plot the diagonal elements r_{jj} produced by both computations with s_{jj} in one figure. Which is more numerically stable, classical or modified?