Nicole Deyerl
Numerical Methods
9/24/15

# Project 1

```
%classical gram Schmidt code

function [q,r] = ndcgs(A)
[m,n]=size(A);
for j=1:n
    v(:,j)=A(:,j);
    for i=1:j-1
        r(i,j)=transpose(q(:,i))*A(:,j);
        v(:,j)=v(:,j)-r(i,j).*q(:,i);
    end
    r(j,j)=norm(v(:,j)); %2norm for a matrix is just norm;
    q(:,j)=v(:,j)./r(j,j);
end

end



%modified gram Schmidt code

function [q,r] = ndmgs(A)
[m,n]=size(A);
for i=1:n
    v(:,i)=A(:,i);
end
for i=1:n
    r(i,i)=norm(v(:,i));
    q(:,i)=v(:,i)/r(i,i);
    for j=i+1:n
        r(i,j)=transpose(q(:,i))*v(:,j);
        v(:,j)=v(:,j)-r(i,j).*q(:,i);
    end
end
end
```

```
EDU>> A=rand(20,10);
EDU>> [Q,R]=ndcgs(A);
EDU>> norm(A-Q*R)

ans =

   3.5058e-16

EDU>> I=eye(10);
EDU>> norm(Q'*Q-I)

ans =

   1.8927e-15
EDU>> [Q,R]=ndmgs(A);
EDU>> norm(A-Q*R)

ans =

   4.5616e-16
EDU>> norm(Q'*Q-I)

ans =

   7.8309e-16
```

| | \|\|A-QR\|\| | | | \|\|Q*Q-I\|\| | | |
|---|---|---|---|---|---|---|
| | CGS | MGS | MatlabQR | CGS | MGS | MatlabQR |
| n=5 | 5.60E-17 | 6.58E-17 | 5.63E-16 | 9.36E-10 | 2.89E-13 | 6.71E-16 |
| n=10 | 6.30E-17 | 8.01E-17 | 5.78E-16 | 2.850558 | 5.38E-06 | 6.11E-16 |
| n=15 | 7.21E-17 | 9.01E-17 | 5.88E-16 | 7.817911 | 0.897559 | 9.32E-16 |
| n=20 | 7.50E-17 | 9.19E-17 | 5.99E-16 | 12.80726 | 0.999221 | 1.53E-15 |

The modified and classical gram Schmidt perform worse for larger values of n (the classical gram Schmidt more so). The Matlab economy sized QR does not carry the least amount of error for all values of n compared to the classical and modified gram Schmidt, but it performs the most consistently for all values of n.

```
A=hilb(20);
a=[]; b=[];
n=5;
A=h(:,1:n);
[Q,R]=ndcgs(A);
x=norm(A-Q*R);
I=eye(n);
y=norm(Q'*Q-I);
a=[a,x];
b=[b,y];
%repeated for n=10, 15, 20 then repeated for MGS and qr(A,0).
```