MATH-6600, CLA Problem Set No. 10, 12-10-15

Name: Michael Hennessey

1. GMRES code:

```
function [x,nit]=gmres0(A,b,x0,maxit,tol)
%This function implements the GMRES algorithm to determine an approximate
%solution to a linear system Ax=b.
%maxit defines the maximum number of iterations allowed
%tol is the residual error tolerance
%x0 is the initial guess of the solution
%The solution is returned as x along with the number of iteration nit
%required
x=x0;
[~,m]=size(A);
I=1:m;
Q(I,1)=b/norm(b);
residOld=1;
for n=1:maxit
    %Calculate the residual from step n-1
    resid=norm(b-A*x);
    if resid<tol
        break;
    end

    %Arnoldi iteration
    v=A*Q(I,n);
    for j=1:n
        H(j,n)=Q(I,j)'*v;
        v=v-H(j,n)*Q(I,j);
    end
    H(n+1,n)=norm(v);
    if norm(v)==0
        break;
    end
    Q(I,n+1)=v/H(n+1,n);
    %minimize
    [~,R]=qr(H,0);
    [Omega,~]=qr(H);
    [q,~]=size(Omega');
    e=eye(q,1);
    g=norm(b)*Omega'*e;
    [p,~]=size(R);
    y=R\g(1:p);

    x=Q(I,1:n)*y;
```

```
    fprintf('GMRES: n=%3d, ||r_n||_2 = %8.2e, ratio=%8.2e\n',n,resid,resid/residOld);
    residOld=resid;
end

nit=n;

%Check error
xTrue=A\b;

fprintf('FINAL: Solution from GMRES: (nit=%d)\n',nit);
fprintf(' 2-norm error=%8.2e\n',norm(x-xTrue,2));
fprintf('max-norm error=%8.2e\n',norm(x-xTrue,inf));
```

Tridiagonal matrix and b creation code:

```
%create an m by m tridiagonal matrix
m=100;
A=zeros(m,m);
A(1,1)=4; %avoids the issue of accessing A(1,0)
A(1,2)=1;
for i=2:m
    A(i,i)=4; %Diagonal entries
    A(i,i+1)=2; %Superdiagonal entries
    A(i,i-1)=1; %subdiagonal entries
end
A=A(1:m,1:m); %Gets rid of accidental m+1 column

%form b
b=zeros(m,1);
for i=1:m
    b(i)=1+i/m;
end
```

Results:

```
>> [x,nit]=gmres0(A,b,b,30,10^(-5))
GMRES: n=  1, ||r_n||_2 = 9.15e+01, ratio=9.15e+01
GMRES: n=  2, ||r_n||_2 = 5.88e-01, ratio=6.43e-03
GMRES: n=  3, ||r_n||_2 = 2.59e-01, ratio=4.40e-01
GMRES: n=  4, ||r_n||_2 = 1.39e-01, ratio=5.35e-01
GMRES: n=  5, ||r_n||_2 = 7.86e-02, ratio=5.68e-01
GMRES: n=  6, ||r_n||_2 = 4.56e-02, ratio=5.79e-01
GMRES: n=  7, ||r_n||_2 = 2.66e-02, ratio=5.84e-01
GMRES: n=  8, ||r_n||_2 = 1.55e-02, ratio=5.85e-01
GMRES: n=  9, ||r_n||_2 = 9.10e-03, ratio=5.86e-01
GMRES: n= 10, ||r_n||_2 = 5.33e-03, ratio=5.86e-01
GMRES: n= 11, ||r_n||_2 = 3.12e-03, ratio=5.86e-01
GMRES: n= 12, ||r_n||_2 = 1.83e-03, ratio=5.86e-01
GMRES: n= 13, ||r_n||_2 = 1.07e-03, ratio=5.86e-01
GMRES: n= 14, ||r_n||_2 = 6.28e-04, ratio=5.86e-01
GMRES: n= 15, ||r_n||_2 = 3.68e-04, ratio=5.86e-01
GMRES: n= 16, ||r_n||_2 = 2.15e-04, ratio=5.86e-01
```

```
GMRES: n= 17, ||r_n||_2 = 1.26e-04, ratio=5.86e-01
GMRES: n= 18, ||r_n||_2 = 7.39e-05, ratio=5.86e-01
GMRES: n= 19, ||r_n||_2 = 4.33e-05, ratio=5.86e-01
GMRES: n= 20, ||r_n||_2 = 2.54e-05, ratio=5.86e-01
GMRES: n= 21, ||r_n||_2 = 1.49e-05, ratio=5.86e-01
FINAL: Solution from GMRES: (nit=22)
 2-norm error=6.06e-06
max-norm error=3.79e-06
```

2. Conjugate Gradient code:

```
function [x,nit]=cg0(A,b,x0,maxit,tol)
%This function applies the conjugate gradient iteration to determine an
%approximate solution to the system Ax=b
%maxit defines the maximum number of iterations allowed
%tol is the residual error tolerance
%x0 is the initial guess of the solution
%The solution is returned as x along with the number of iteration nit
%required


[~,m]=size(A);
I=1:m;
x=x0;
r(I,1)=b-A*x0;
p(I,1)=r(I,1);
residOld=norm(r(I,1),2);
%note we start at 2 so as to allow a more convenient notation
for n=2:maxit+1
    alpha(n-1)=(r(I,n-1)'*r(I,n-1))/(p(I,n-1)'*A*p(I,n-1)); %step length
    x=x+alpha(n-1)*p(I,n-1); %approximate solution
    r(I,n)=r(I,n-1)-alpha(n-1)*A*p(I,n-1); %residual
    resid=norm(r(I,n),2);
    fprintf('CG: n=%3d, || r_n ||_2 = %8.2e, ratio=%8.2e\n',n-1,resid,resid/residOld)
    if resid<tol
        break;
    end
    residOld=resid;
    beta(n-1)=(r(I,n)'*r(I,n))/(r(I,n-1)'*r(I,n-1)); %improvement
    p(I,n)=r(I,n)+beta(n-1)*p(I,n-1); %search direction
end
nit=n-1;
%Adjust the solution
x=x;
xTrue=A\b;
fprintf('FINAL: Solution from CG: (nit=%d)\n',nit);
fprintf(' 2-norm error=%8.2e\n',norm(x-xTrue,2));
fprintf('max-norm error=%8.2e\n',norm(x-xTrue,inf));
```

Tridiagonal matrix and b creation code:

```
%create an m by m tridiagonal matrix
```

```
m=100;
A=zeros(m,m);
A(1,1)=4; %avoids the issue of accessing A(1,0)
A(1,2)=1;
for i=2:m
    A(i,i)=4; %Diagonal entries
    A(i,i+1)=1; %Superdiagonal entries
    A(i,i-1)=1; %subdiagonal entries
end
A=A(1:m,1:m); %Gets rid of accidental m+1 column

%form b
b=zeros(m,1);
for i=1:m
    b(i)=1+i/m;
end
```

Results:

```
[x,nit]=cg0(A,b,zeros(100,1),30,10^(-6));
CG: n=  1, || r_n ||_2 = 3.71e-01, ratio=2.42e-02
CG: n=  2, || r_n ||_2 = 9.29e-02, ratio=2.50e-01
CG: n=  3, || r_n ||_2 = 2.48e-02, ratio=2.67e-01
CG: n=  4, || r_n ||_2 = 6.64e-03, ratio=2.68e-01
CG: n=  5, || r_n ||_2 = 1.78e-03, ratio=2.68e-01
CG: n=  6, || r_n ||_2 = 4.76e-04, ratio=2.68e-01
CG: n=  7, || r_n ||_2 = 1.28e-04, ratio=2.68e-01
CG: n=  8, || r_n ||_2 = 3.42e-05, ratio=2.68e-01
CG: n=  9, || r_n ||_2 = 9.16e-06, ratio=2.68e-01
CG: n= 10, || r_n ||_2 = 2.45e-06, ratio=2.68e-01
CG: n= 11, || r_n ||_2 = 6.58e-07, ratio=2.68e-01
FINAL: Solution from CG: (nit=11)
 2-norm error=2.04e-07
max-norm error=1.69e-07
```