Université de Picardie Jules Verne
Master 2 AAM
Calcul Scientifique
2019-2020

Méthodes de sous espaces de Krylov 2

1 GMRES redémarré

1- Calculer	$r^0 = b - Ax^0, \beta = \|r^0\|, v_1 = \frac{r^0}{\beta}$
2- Générer la base d'Arnoldi et	$\hat{H}_m$ en partant de v_1
3- Calculer y_m :	$y_m = \text{argmin}\|\beta e_1 - \hat{H}_m y_m\|$
4- Poser	$x_m = x_0 + V_m y_m$
5- Si précision atteinte	STOP
Sinon	Poser $x_0 = x_m$ aller en 1

2 Exercices

Exercice 1 (Comparaison de GMRES redémarré et du Gradient Conjugué (GC))
On considère la matrice de discrétisation du laplacien en dimension 2 :

```
Axx=2*eye(N,N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
d=speye(N,N);
A=kron(Id,Axx)+kron(Axx,Id);
```

1. Pour $N = 10$, $m = 1$ et `b=1-2*rand(N*N,1)`, comparer le nombre d'itérations nécessaires à la convergence pour GMRES et GC. Que constatez-vous ?

2. Recommencer l'expérience avec différentes valeurs de m et de N.

3. Mêmes questions avec les versions préconditionnées de GC et de GMRES, on utilisera ILU(0)

Exercice 2 (Résolution d'un système linéaire par GMRES avec redémarrage - cas non symétrique)
On considèr e l'EDP

$$-\Delta u + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = f \quad \text{Dans } \Omega =]0,1[^2, \tag{1}$$

$$u = 0 \qquad \text{sur } \partial\Omega \tag{2}$$

Pour construire les matrices de discrétisation de on procède comme suit :

- Construction de la matrice 1-D :
  `v=[0 1 zeros(1,N-2)]`, `Dx=toeplitz(-v',v)`,
- Version 2D à l'aide du produit de Kronecker :
  `D2x=Id.*.Dx/(2*h)` et de même `D2y=Dx.*.Id/(2*h)`

1. Pour $a = b = 1$, appliquer GMRES avec différentes valeurs de m
2. Recommencer avec $a = 10^6$ et $b = 1$. Que constatez-vous ?
3. Garder ces paramètres et utiliser la version préconditionnée de GMRES avec ILU(0). Conclusion ?
4. Représenter le spectre de la matrice dans sa version préconditionnée et non préconditionnée dans le plan complexe. Que constatez-vous ?

3 Annexe

3.1 Arnoldi pour les systèmes linéaires

```
function [x]=Arnoldi_schoen(A,b,x0,n,m)
epsilon=1.e-15;
r=b-A*x0; x=zeros(n,1); beta=norm(r,2);
v=r/beta;
V=[v];
H=zeros(m,m);
stop=0;
e1=[1 ; zeros(n-1,1)];
j=1;
while j< = m & stop==0
w=A*V(:,j);
for i=1:j
H(i,j)=w'*V(:,i);
w=w-H(i,j)*V(:,i);
end
H(j+1,j)=norm(w,2);

if abs(H(j+1,j)) >= epsilon
V(:,j+1)=w/H(j+1,j);
else
m=j;stop=1;
end
j=j+1;
end
[nr,nc]=size(H);
y=H(1:nc,1:nc)\(beta*e1);
x(1:nc,1)=x(1:nc,1)+V(1:nc,1:nc)*y;
endfunction
```

3.2 Householder

```
function [ c, s ] = rotmat( a, b )
//
// Calcul des paramtres de rotations de Givens pour a et b
//
if ( b == 0.0 ),
c = 1.0;
s = 0.0;
elseif ( abs(b) > abs(a) ),
temp = a / b;
s = 1.0 / sqrt( 1.0 + temp^2 );
c = temp * s; else temp = b / a; c = 1.0 / sqrt( 1.0 + temp^2 );
s = temp * c;
end
endfunction
```

3.3 GMRES - redémarré

```
//
// Fonction GMRES avec reémarrage adaptée depuis Matlab
//
function [x, error, kiter, flag, residu, mvpvec] = gmres( A, x, b, M, restrt, max_it,
tol )
// -- Iterative template routine --
// Univ.  of Tennessee and Oak Ridge National Laboratory
// October 1, 1993
// Details of this algorithm are described in "Templates for the
// Solution of Linear Systems:  Building Blocks for Iterative
// Methods", Barrett, Berry, Chan, Demmel, Donato, Dongarra,
// Eijkhout, Pozo, Romine, and van der Vorst, SIAM Publications,
// 1993.  (ftp netlib2.cs.utk.edu; cd linalg; get templates.ps).
//
// gmres.m solves the linear system Ax=b
// using the Generalized Minimal residual ( GMRESm ) method with restarts .
//
// input A REAL nonsymmetric positive definite matrix
// x REAL initial guess vector
// b REAL right hand side vector
// M REAL preconditioner matrix
// restrt INTEGER number of iterations between restarts
// max_it INTEGER maximum number of iterations
// tol REAL error tolerance
//
// output x REAL solution vector
// error REAL error norm
// iter INTEGER number of iterations performed
// flag INTEGER: 0 = solution found to tolerance
// 1 = no convergence given max_it
// resvec REAL (iter x 1) vector of norm2 solution residuals at each iteration
// mvpvec INTEGER (iter x 1) vector of ACTUALLY computed MVPs at each iteration
//
//iter = 0; //initialization
kiter=1;
ndim=max(size(x));
//resvec=zeros(ndim,1);
resvec=[];
mvpvec=zeros(ndim,1);
flag = 0;
mvp_count =0;

bnrm2 = norm( b );
if ( bnrm2 == 0.0 ), bnrm2 = 1.0; end
```

```
r = M \ ( b-A*x ); // not recorded and MVP is not counted
error = norm( r ) / bnrm2;
residu=[error];
if ( error < tol ) return, end

   [n,n] = size(A);
// initialize workspace
m = restrt;
V(1:n,1:m+1) = zeros(n,m+1);
H(1:m+1,1:m) = zeros(m+1,m);
cs(1:m) = zeros(m,1);
sn(1:m) = zeros(m,1);
e1 = zeros(n,1);
e1(1) = 1.0;


   for iter = 1:max_it, // begin iteration
r = M \ ( b-A*x ); mvp_count = mvp_count+1;
V(:,1) = r / norm( r );
s = norm( r )*e1;
for i = 1:m, // construct orthonormal
w = M \ (A*V(:,i)); mvp_count = mvp_count+1; // basis using Gram-Schmidt
for k = 1:i,
H(k,i)= w'*V(:,k);
w = w - H(k,i)*V(:,k);
end
H(i+1,i) = norm( w );
V(:,i+1) = w / H(i+1,i);
for k = 1:i-1, // apply Givens rotation
temp = cs(k)*H(k,i) + sn(k)*H(k+1,i);
H(k+1,i) = -sn(k)*H(k,i) + cs(k)*H(k+1,i);
H(k,i) = temp;
end
[cs(i),sn(i)] = rotmat( H(i,i), H(i+1,i) ); // form i-th rotation matrix
temp = cs(i)*s(i); // approximate residual norm
s(i+1) = -sn(i)*s(i);
s(i) = temp;
H(i,i) = cs(i)*H(i,i) + sn(i)*H(i+1,i);
H(i+1,i) = 0.0;
error = abs(s(i+1)) / bnrm2;
residu=[residu error];
if ( error <= tol ), // update approximation
y = H(1:i,1:i)  s(1:i); // and exit
x = x + V(:,1:i)*y;
break;
end
end
```

```
if ( error <= tol ), break, end
y = H(1:m,1:m) \ s(1:m);
x = x + V(:,1:m)*y; // update approximation
r = M \( b-A*x ); mvp_count = mvp_count+1; // compute residual

resvec= [resvec, r]; // store the residual AT the iter's iteration
mvpvec(iter) = mvp_count; // store ACTUALLY performed MVP count BY iters //iteration
s(i+1) = norm(r);
error = s(i+1) / bnrm2; // check convergence

if ( error <= tol ), break, end;
end

if ( error > tol ) flag = 1; end; // converged
kiter=iter;
endfunction
```