Résolution de Grands Systèmes Linéaires

Nicolas LIMARE

28 février 2005

Résumé

Cette étude consiste à implémenter, en fortran90, les trois algorithmes de résolution de système linéaire Lanczos, BiCGStab et GMRes, puis à en comparer les performances. Les trois méthodes sont ainsi testées sur des matrices creuses, grande taille (entre 67 et 1856 dans les exemples choisis).

Ces matrices sont lues à partir de fichiers texte au format *Harwell-Boeing*, puis stockées et utilisées dans un format de matrices creuses compressées (format *Morse*), dans un souci de ne pas encombrer la mémoire de travail.

L'historique de convergence de ces résolutions de système est alors représenté graphiquement afin de comparer les performances des trois méthodes.

Table des matières

1 Résultats numériques **2**

2 Implémentation **5**

2.1 Lecture et manipulation des données 5

2.2 Algorithmes 7

2.2.1 Lanczos 7

2.2.2 BiCGStab 7

2.2.3 GMRes 8

2.3 Programme principal 11

1 Résultats numériques

Les premières résolutions de systèmes $Ax = b$ étaient problématiques, car le plus souvent, aucun algorithme ne convergeait. Ce problème a été temporairement résolu par la modification de chacune des matrices A en lui ajoutant une diagonale de valeur suffisamment élevée. Cela résout probablement des problèmes de mauvais conditionnement, et pallie à la nécessité (bien réelle, si on souhaite résoudre un système linéaire donné) de prise en compte d'un préconditionneur.

Avec les matrices ainsi modifiées, le système converge très vite, et on obtient les figures des pages suivantes, en considérant l'ereur relative de Ax par rapport à b, et pour une valeur initiale $x_{0i} = 3/2\bar{x}_i$ ou $x_{0i} = 1/2\bar{x}_i$, alternativement dun indice au suivant, $\bar{x}$ étant la solution exacte. La convergence est stoppée lorsque $\epsilon < 10^{-8}$, et avec un espace de *Krylov* de taille 10 pour l'algorithme GMRes redémarré. Pour les convergences très rapides, lorsque plusieurs résultats sont fournis avec le même temps de calcul (intervalle trop courte), la moyenne de ces résultats est prise pour tracer la ligne de convergence.

Il apparait clairement que l'algorithme GMRes est le plus rapide des trois ; cependant, la stabilisation de la convergence attendue de BiCGStab par rapport à Lanczos n'est pas observable. Lanczos est en général, sur ces exemples, le plus lent des trois algorithmes ; dans les premiers essais, sans modification de A, on remarquait cependant que Laznczos parvenait à résoudre des systèmes où les autres échouaient.

Il serait intéressant d'effectuer ce test à partir de matrices dont on maitriserait mieux les propriétés, et le conditionnement, afin d'observer les liens entre la performance des algorithmes et le type de matrice ou le type de problème étudié.

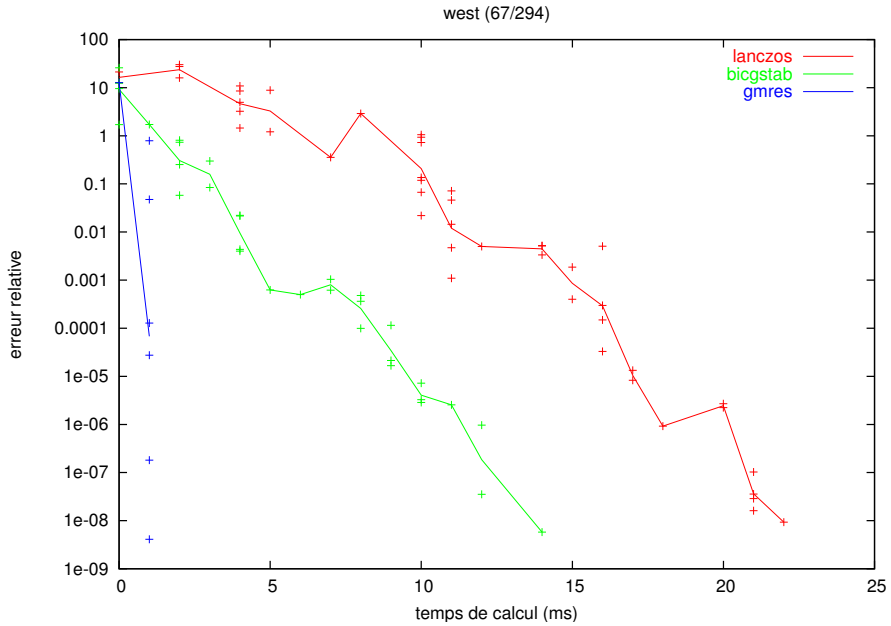

FIG. 1 – chimie, $n = 67$

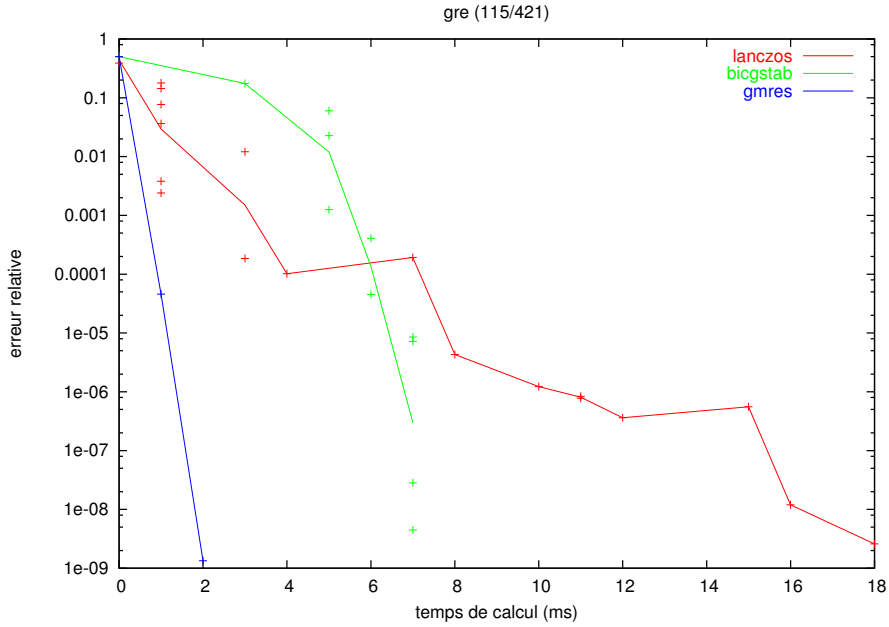

FIG. 2 – simulation de réseaux informatiques, $n = 115$

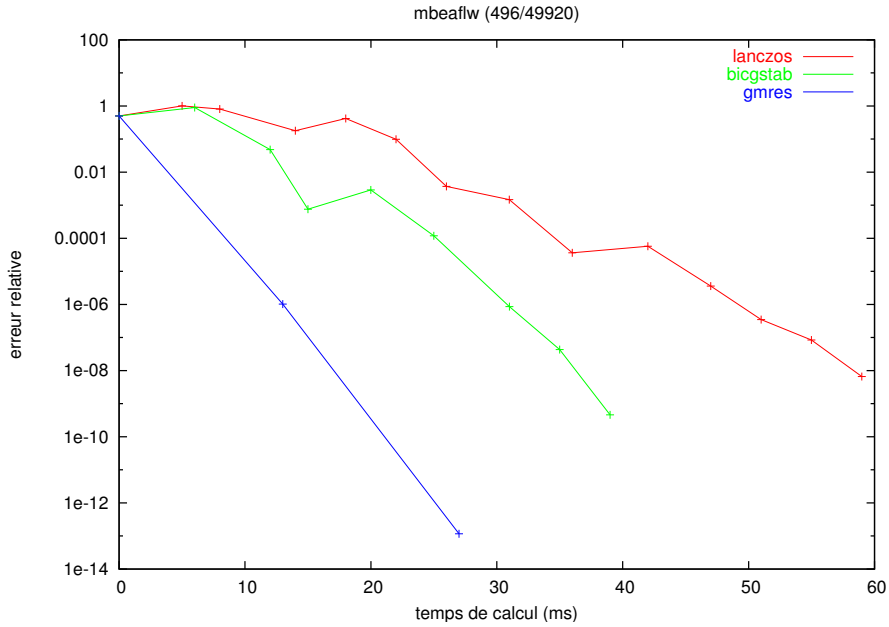

FIG. 3 – économie, $n = 496$

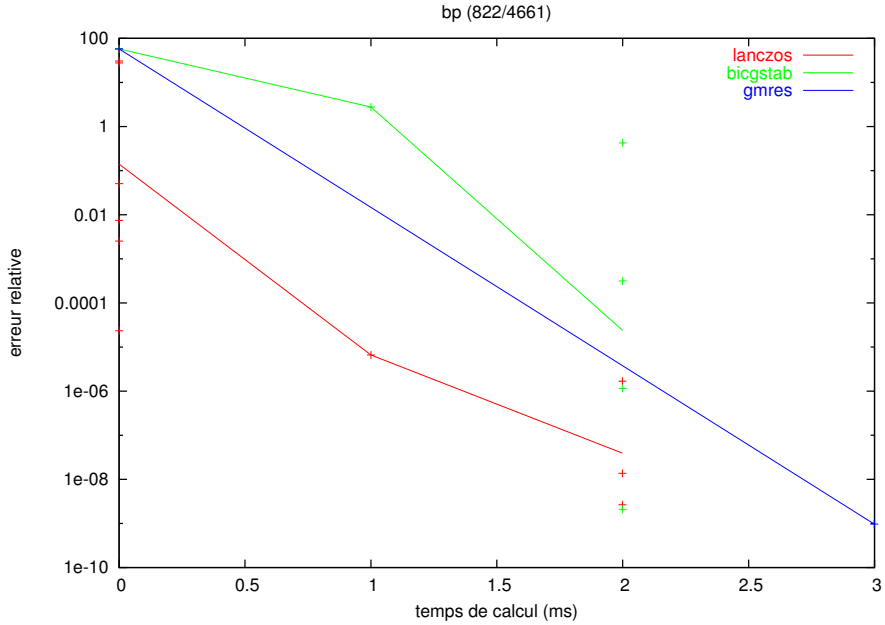

FIG. 4 – recherche opérationnelle, méthode du simplexe, $n = 822$

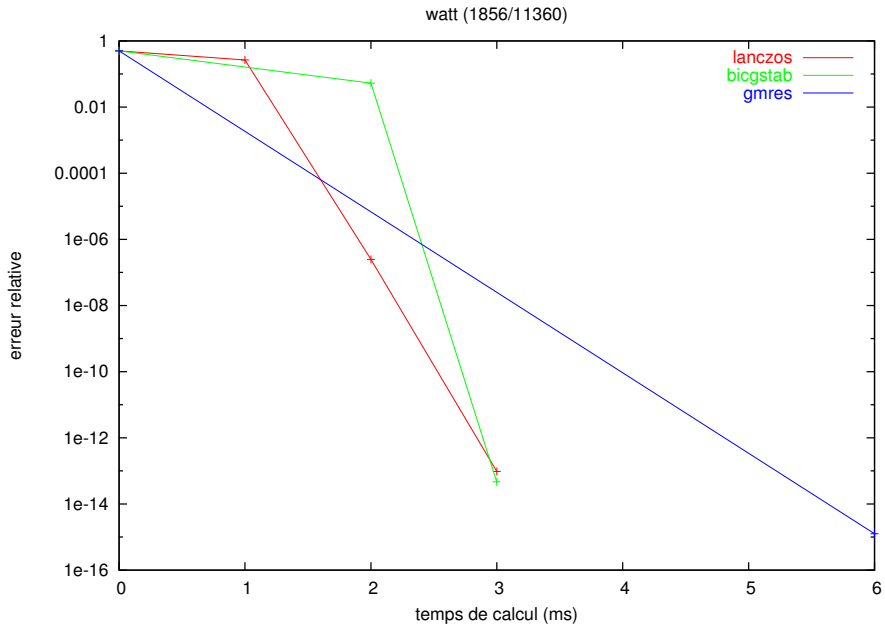

FIG. 5 – ingénierie pétrolière, $n = 1856$

2 Implémentation

Seulement des extraits des routines importantes sont reproduits ici. Le code source, avec les routines destinées à la gestion de l'historique de convergence, des erreurs ou des des paramètres de ligne de commande, est disponible dans l'archive jointe.

2.1 Lecture et manipulation des données

Bien que cela ne soit pas nécessaire à l échelle de ce projet, des types dérivés ont été créés pour gérer les fichiers au format *Harwell-Boeing* et les matrices compressées au format *Morse*. Le type `hbfile` est composé de variables contenant les informations figurant dans les fichiers de données (taille, type de données, format, etc.), à l'exception des valeurs que contient la matrice. Son chargement par la routine `hbf_load` permet de vérifier la qualité et le format du fichier fourni.

Le type `csmat` est composé des trois vecteurs de stockage dela matrice, ainsi que de sa dimension et du nombre d'éléments non nuls de la matrice, afin de ne pas les recalculer à chaque fois. Ces vecteurs sont des tableaux à allocation dynamique, alloués par la routine `csm_init`. Les tableaux dynamiques dans des types dérivés ne sont pas inclus dans la norme fortran95, mais pris en charge par de nombreux compilateurs, et inclus dans des documents préliminaires du fortran2003.

```
type csmat
   integer :: n,nval
   integer,allocatable,dimension(:) :: colptr,rowind
   real(kind(0.d0)),allocatable,dimension(:) :: values
endtype csmat

subroutine csm_load(csm,filename)
  use hbf_mod
  use csm_mod
  use misc_interface
  use hbf_interface
  implicit none
  type(csmat),intent(out) :: csm
  character(len=*),intent(in) :: filename
  type(hbfile) :: hbf
  integer :: fileunit,ios,i,n

  call hbf_load(hbf,filename)
  call csm_init(csm,hbf\%ncol,hbf\%nnzero)

  fileunit=get_fileunit()
  open(unit=fileunit,file=hbf\%filename,status='old',iostat=ios)
  if(ios/=0) call error(201)

  n=4
  if(0<hbf\%rhscrd) n=n+1
  do i=1,n
     read(unit=fileunit,fmt='(a)',iostat=ios)
     if(ios/=0) call error(203)
  enddo
```

```
  read(unit=fileunit,fmt=hbf\%ptrfmt,iostat=ios) &
       csm\%colptr(1:csm\%n+1)
  if(ios/=0) call error(301)
  read(unit=fileunit,fmt=hbf\%indfmt,iostat=ios) &
       csm\%rowind(1:csm\%nval)
  if(ios/=0 ) call error(302)
  read(unit=fileunit,fmt=hbf\%valfmt,iostat=ios) &
       csm\%values(1:csm\%nval)
  if(ios/=0 ) call error(302)

  close(unit=fileunit)
endsubroutine csm_load
```

Enfin, les produits matrice-vecteur sont gérés par les routines `csm_vect_pdt` et `vect_csm_pdt` (cette dernière permet de calculer $A^t x = (x^t A)^t$).

```
subroutine vect_csm_pdt(x,csm,y)
  use csm_mod
  use misc_interface
  implicit none
  real(kind(0.d0)),dimension(:),intent(in) :: x
  type(csmat),intent(in) :: csm
  real(kind(0.d0)),dimension(:),intent(out) :: y
  integer :: i,j

  if((size(x)/=csm\%n).or.(size(y)/=csm\%n)) call error(410)
  y=0
  do j=1,csm\%n
     do i=csm\%colptr(j),csm\%colptr(j+1)-1
        y(j)=y(j)+x(csm\%rowind(i))*csm\%values(i)
     enddo
  enddo
return
endsubroutine vect_csm_pdt
```

2.2 Algorithmes

2.2.1 Lanczos

```
r=b-a_x
p=r
rbar=r
pbar=rbar

i=0
if(log_flag) call cpu_time(start_time)
do
   eps=rel_diff(a_x,b)
   if(log_flag) then
      call cpu_time(time)
      call log_add(trim(adjustl(int2char(i)))//&
           ' '//trim(adjustl(real2char(time-start_time)))//&
           ' '//trim(adjustl(double2char(eps))))
   endif
   if((i>=i_max).or.(eps<=eps_max)) exit

   call csm_vect_pdt(a,p,a_p)
   rbar_r=dot_product(rbar,r)
   beta=rbar_r/dot_product(pbar,a_p)
   r=r-beta*a_p
   x=x+beta*p
   call vect_csm_pdt(pbar,a,pbar_a)
   rbar=rbar-beta*pbar_a
   alpha=dot_product(rbar,r)/rbar_r
   p=r+alpha*p
   pbar=rbar+alpha*pbar

   call csm_vect_pdt(a,x,a_x)
   i=i+1
enddo
```

2.2.2 BiCGStab

On utilise l'algorithme suivant[1] :
.

pour $k = 1, ...$
$\beta_k = (y, r_{k-1})/(y, Ap_k)$
$u_k = r_{k-1} - \beta_k A p_k$
$\gamma_k = (u_k, Au_k)/(Au_k, Au_k)$
$x_k = x_{k-1} + \beta_k p_k + \gamma_k u_k$
$r_k = u_k - \gamma_k A u_k$
$\alpha_{k+1} = (y, r_k)/(y, r_{k-1})$
$p_{k+1} = r_k + \alpha(I - \gamma_k A)p_{k1}$
si $|Ax - b| < \epsilon$ la convergence est terminée

[1] ...dont je n'ai pu redémontrer la validité.

```
fail_flag=.false.
call csm_vect_pdt(a,x,a_x)
r=b-a_x
y=r

i=0
if(log_flag) call cpu_time(start_time)
do
   eps=rel_diff(a_x,b)
   if(log_flag) then
      call cpu_time(time)
      call log_add(trim(adjustl(int2char(i)))//&
           ' '//trim(adjustl(real2char(time-start_time)))//&
           ' '//trim(adjustl(double2char(eps))))
   endif
   if((fail_flag).or.(i>=i_max).or.(eps<=eps_max)) exit

   rhobar=rho
   rho=dot_product(y,r)
   if(rho==0) then
      fail_flag=.true.
   else
      if(i==0) then
         p=r
      else
         alpha=(rho/rhobar)*(beta/gamma)
         p=r+alpha*(p-gamma*a_p)
      endif
      call csm_vect_pdt(a,p,a_p)
      beta=rho/dot_product(y,a_p)
      u=r-beta*a_p
      call csm_vect_pdt(a,u,a_u)
      gamma=dot_product(u,a_u)/dot_product(a_u,a_u)
      x=x+beta*p+gamma*u
      r=u-gamma*A_u
   endif

   call csm_vect_pdt(a,x,a_x)
   i=i+1
enddo
```

2.2.3 GMRes

```
h=0
r=b-a_x
beta=sqrt(dot_product(r,r))
xbar=x+1

i=0
if(log_flag) call cpu_time(start_time)
```

```
do

   eps=rel_diff(a_x,b)
   if(log_flag) then
      call cpu_time(time)
      call log_add(trim(adjustl(int2char(i)))//&
           ' '//trim(adjustl(real2char(time-start_time)))//&
           ' '//trim(adjustl(double2char(eps))))
   endif

   delta=max(maxval(x-xbar),-minval(x-xbar))
   if((delta==0).or.(i>=i_max).or.(eps<=eps_max)) exit
   xbar=x

! arnoldi method
   v(:,1)=r/beta
   do j=1,m
      call csm_vect_pdt(a,v(:,j),omega)
      do k=1,j
         h(k,j)=dot_product(omega,v(:,k))
         omega=omega-h(k,j)*v(:,k)
      enddo
      h(j+1,j)=sqrt(dot_product(omega,omega))
      if(h(j+1,j)==0) exit
      if(j<m) v(:,j+1)=omega/h(j+1,j)
   enddo

! triangulation of h and y update
   y=0
   y(1)=beta
   do j=1,m
      alpha=sqrt(h(j,j)*h(j,j)+h(j+1,j)*h(j+1,j))
      if(alpha==0) alpha=1
      c_alpha=h(j,j)/alpha
      s_alpha=-h(j+1,j)/alpha
      do k=1,m
         hjk=h(j,k)
         hj1k=h(j+1,k)
         h(j,k)=c_alpha*hjk-s_alpha*hj1k
         h(j+1,k)=s_alpha*hjk+c_alpha*hj1k
      enddo
      if(j<m) y(j+1)=s_alpha*y(j)
      y(j)=c_alpha*y(j)
   enddo

! solve hx=y->y
   do j=m,1,-1
      if(h(j,j)==0) h(j,j)=1
      y(j)=y(j)/h(j,j)
      y(1:j-1)=y(1:j-1)-y(j)*h(1:j-1,j)
```

```
    enddo

    x=x+matmul(v,y)

    call csm_vect_pdt(a,x,a_x)
    r=b-a_x
    beta = sqrt(dot_product(r,r))
    i=i+1
enddo
```

2.3 Programme principal

Le programme principal, `msolve` est assez simple, et se contente de lire les paramètres de ligne de commande, charger les fichiers de données et y appliquer chacun des trois algorithmes. L'essentiel (précision, itérations, log, fichiers de données) est paramétrable en ligne de commande ; la syntaxe est indiquée lors d'un appel avec l'option `-help` :

```
> msolve -help
 syntax: msolve [options] filenames

 options:
  N: integer, D: double, B: bool(1,0,yes,no), S: string
  -imax=N, -eps=D, -dimk=N
   parameters for the solvers (will be prompted if not provided)
  -lanczos=B, -bicgstab=B, -gmres=B
   selection of the solvers to use (default: no)
  -clean=B, -sym=B, -inv=B
   preliminary operations on the matrix (default: no)
  -info=B, -log=B
   ouputs (default: info=no, log=yes)
  -logtag=S
   tag to the output log filenames
  -help
   command-line instructions help
```