Compte rendu du TP2 de Fortran

Laurent Valentin

31 janvier 2013

Table des matières

1.1 Objectifs de TP

On cherche à résoudre le système linéaire

$$Ax = b$$

On sait résoudre ce système grâce à des méthodes directes comme Gauss ou la décomposition LU. Pourtant ces methodes sont inefficaces pour des matrices creuses ou lorsque la taille de la matrice est grande.

Ce TP a pour objectif de découvrir les methodes itératives permettant de résoudre ces systèmes plus efficacement. Il s'agira de programmer ces methodes en Fortran et de les comparer afin de résoudre un systeme quelconque.

Ces méthodes fonctionnent sur le même principe : on part d'un vecteur x^0 et on calcule $r^0 = Ax^0 - b$ un vecteur de $(R)^n$ appelé résidu du système. On itère alors ce principe n fois jusqu'à ce qu'un critère d'arrêt soit vérifié. On a alors la suite de vecteurs $\{x^k\}_{k\geq 0}$.

Des théorèmes nous montrent alors que ces méthodes convergent vers un vecteur x qui est solution du système $Ax = b$. Ici, le critère d'arrêt sera choisi par l'utilisateur lors de l'initialisation de « eps »dans *syslin.f90.* On comparera cette valeur à

$$\frac{\|x^{k+1} - x^k\|}{\|x^{k+1}\|}$$

On choisira dans ce TP la norme 2 pour effectuer le test d'arrêt.

1.2 Algorithmes importants

On a 3 methodes importantes qui seront programmées dans ce TP : la methode de Jacobi, la methode de Gauss-Seidel et la methode de relaxation. Chacune de ces methodes respectent le principe décrit dans 1.1.

La methode de Jacobi

Algorithm 1 Methode de Jacobi

Require: la matrice A et le vecteur b
Ensure: le vecteur x tel que $Ax = b$
  $x^k \leftarrow x^0$
  while $\frac{\|x^{k+1}-x^k\|}{\|x^{k+1}\|} \geq \epsilon$ **do**
    for $i = 1 \rightarrow n$ **do**
      $x_i^{k+1} \leftarrow \frac{1}{a_{i,i}}[b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^k - \sum_{j=i+1}^{n} a_{i,j}x_j^k]$
    end for
  end while

La methode de Gauss-Seidel

Algorithm 2 Methode de Gauss-Seidel

Require: la matrice A et le vecteur b

Ensure: le vecteur x tel que $A * x = b$

$x^k \leftarrow x^0$

while $\frac{\|x^{k+1}-x^k\|}{\|x^{k+1}\|} \geq \epsilon$ **do**

for $i = 1 \rightarrow n$ **do**

$$x_i^{k+1} \leftarrow \frac{1}{a_{i,i}}[b_i - \sum_{j=1}^{i-1} a_{i,j} * x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j}x_j^k]$$

end for

end while

La methode de Relaxation

On introduit dans cette methode le coefficient ω de relaxation afin « d'améliorer »la methode de Gauss-Seidel

Algorithm 3 Methode de Relaxation

Require: la matrice A et le vecteur b

Ensure: le vecteur x tel que $Ax = b$

$x^k \leftarrow x^0$

while $\frac{\|x^{k+1}-x^k\|}{\|x^{k+1}\|} \geq \epsilon$ **do**

for $i = 1 \rightarrow n$ **do**

$$x_i^{k+1} \leftarrow (1-\omega)x_i^k + \frac{\omega}{a_{i,i}}[b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j}x_j^k]$$

end for

end while

1.3 Listing en Fortran

1.3.1 Programme principal : syslin.f90

Le programme affiche le menu, effectue la demande "ichoix" et appelle les modules "methode.f90" et "affichage.f90".

```
!------------------------------------------------------------------------------
! TP2 : Méthodes itératives de résolution des systèmes linéaires
! Auteur : Valentin Laurent (ZZ1,G1)
! Date : 11/12/12
! Compilation : gfortran methode.f90 affichage.f90 syslin.f90 -o syslin
! Fichiers de tests : matrice4.txt et matrice9.txt
!------------------------------------------------------------------------------

PROGRAM syslin

USE methodes
USE affichage

IMPLICIT NONE

integer                                 :: ichoix   !choix de l'utilisateur
real(8),dimension(:,:),allocatable      :: A        !matrice A du système
real(8),dimension(:),allocatable        :: b        !second membre du système
real(8),dimension(:),allocatable        :: x        !inconnue du système
real(8)                                 :: omega    !coefficient de relaxation
integer                                 :: nb_iter  !nombre d'itération
integer                                 :: n        !rang de A
integer                                 :: i,j,k    !variable de boucle
integer                                 :: menu     !valeur de menu
character (len=20)                      :: nomfich  !nom du fichier
integer                                 :: ierr     !variable d'erreur
real(8)                                 :: x0       !valeur de x0
real(8)                                 :: eps      !valeur de test


!initialisation du menu
menu = 1

! Lecture du nom du fichier de donnees
print *,' Entrez le nom du fichier de donnees : '
read *,nomfich

!initialisation de x
print *,' Entrez la valeur de x0 (0.d0) :'
read *,x0

!initialisation de eps
print *,' Entrez la valeur de epsilon (1.d-12) :'
```

```
read *,eps

DO WHILE (menu /= 0)

    ! Ouverture du fichier en lecture
    open(1,file=nomfich,status='old',action='read',iostat=ierr)
    IF (ierr /= 0) THEN
        print *,'Fichier de donnees inexistant'
        stop
    END IF

    !lecture du rang
    read(1,*) n

    IF (menu == 1) THEN
        !allocation de A,b,x pour la première utilisation
        allocate(A(n,n))
        allocate(b(n))
        allocate(x(n))
    END IF

    !Lecture de la matrice A et de b dans "matrice.txt"
    DO i=1,n
        read(1,*) (A(i,j),j=1,n)
    END DO
    read(1,*) (b(k),k=1,n)

    !fermeture du fichier
    close(1)

    !initialisation de x
    x(1:n) = x0

    !affichage du menu
    print *,"----------------- tp2 Resolution de systemes lineaires -----------------"
    print *,"ichoix = 0 -> quitte le programme"
    print *,"ichoix = 1 -> Methode de Jacobi"
    print *,"ichoix = 2 -> Methode de Gauss-Seidel"
    print *,"ichoix = 3 -> Methode de relaxation"
    print *,"Le choix 3 effectue un tableau pour omega variant de 0.1 à 1.9"
    print*,"------------------------------------------------------------------------"

    !saisie utilisateur
    print *,"Veuillez saisir votre choix ?"
    read *,ichoix

    !choix des différentes methodes
    SELECT CASE(ichoix)
        CASE (0)
            !sortie du programme
```

```
        menu = -1

    CASE (1)
        !methode de Jacobi
        call resolution(1,A,b,omega,nb_iter,x,eps)

        !affichage
        call afficheMat(A,n)
        call afficheVect(x,n)
        print *,"Nombre d'iterations : ",nb_iter

    CASE(2)
        !methode de Gauss-Seidel
        omega = 1.d0
        call resolution(2,A,b,omega,nb_iter,x,eps)

        !affichage
        call afficheMat(A,n)
        call afficheVect(x,n)
        print *,"Nombre d'iterations : ",nb_iter

    CASE(3)
        !methode de relaxation
        omega = 1.d-1
        call afficheMat(A,n)

        DO k=1,19

            !initialisation de x
            x(1:n) = x0

            call resolution(3,A,b,omega,nb_iter,x,eps)

            !affichage
            print '(/)'
            print *,"Valeur d'omega : ",omega
            call afficheVect(x,n)
            print *,"Nombre d'iterations : ",nb_iter
            omega = omega + 1.d-1
        END DO

    CASE DEFAULT
        print *,"Evenement inconnu"

END SELECT

!affichage
menu = menu + 1
print '(/)'
```

```
END DO

print *,"Fin du programme"
print '(/)'
!fin du programme principal (execution)

END PROGRAM syslin
!fin du TP2
```

1.3.2 Module : methode.f90

Ce module effectue les methodes de Jacobi, Gauss-Seidel et de relaxation à selon la valeur de "ichoix".

```
MODULE methodes

CONTAINS

!------------------------------------------------------------------------------
! subroutine des différentes methodes de résolution du système linéaire Ax=b
!------------------------------------------------------------------------------

SUBROUTINE resolution(ichoix,A,b,omega,nb_iter,x1,eps)
IMPLICIT NONE

integer,intent(in)                      :: ichoix         !choix de l'utilisateur
real(8),dimension(:,:),intent(inout)    :: A              !matrice A
real(8),dimension(:),intent(inout)      :: x1             !inconnue
real(8),dimension(:),intent(in)         :: b              !second membre
real(8),intent(inout)                   :: omega          !coefficient de relaxation
real(8),dimension(:),allocatable        :: x2             !vecteur de la suite

integer,intent(out)                     :: nb_iter        !nombre d'itération
integer                                 :: i,j,k,im,ip    !variable de boucle
integer                                 :: n              !rang de A
real(8),intent(in)                      :: eps            !valeur de test
real(8)                                 :: arret          !critère d'arrêt
real(8)                                 :: v1, v2         !calcul des normes

!initialisation
nb_iter = 0
v1 = 0.d0
v2 = 0.d0
n = size(b)
allocate(x2(n))

arret = eps+1.d0

!methode de Jacobi
IF (ichoix == 1) THEN
```

```
    DO WHILE (arret > eps) !on utilise la norme 2

        !initialisation des variables de boucles (calcul de norme et itérations)
        v1 = 0.d0
        v2 = 0.d0
        nb_iter = nb_iter + 1

        !calcul des xk suivant la methode de Jacobi
        DO i=1,n
            ip = i+1
            im = i-1
            x2(i) = (b(i) - dot_product(A(i,1:im),x1(1:im)) &
                          - dot_product(A(i,ip:n),x1(ip:n)))/A(i,i)

            !calcul de la norme 2
            v1 = v1 + (x2(i)-x1(i))**2
            v2 = v2 + x2(i)**2
        END DO

        !calcul du critère d'arrêt (actualisé pour chaque itération)
        arret = sqrt(v1/v2)
        x1 = x2
    END DO

!methode de relaxation + Gauss-Seidel (omega = 1)
ELSE IF ((ichoix == 3) .or. (ichoix == 2)) THEN
    DO WHILE (arret > eps)

        !initialisation des variables de boucles (calcul de norme et itérations)
        v1 = 0.d0
        v2 = 0.d0
        nb_iter = nb_iter + 1

        !calcul des xk suivant la methode de Gauss-Seidel
        DO i=1,n
            ip = i+1
            im = i-1
            x2(i) = (1.d0-omega)*x1(i) + (b(i) - dot_product(A(i,1:im),x2(1:im)) &
                                         - dot_product(A(i,ip:n),x1(ip:n)))*omega/A(i,i)

            !calcul de la norme 2
            v1 = v1 + (x2(i)-x1(i))**2
            v2 = v2 + x2(i)**2
        END DO

        !calcul du critère d'arrêt (actualisé pour chaque itération)
        arret = sqrt(v1/v2)
        x1 = x2
    END DO
END IF
```

```
END SUBROUTINE resolution

!------------------------------------------------------------------------------
!------------------------------------------------------------------------------

END MODULE methodes
```

1.3.3 Module d'affichage : affichage.f90

Le programme d'affichage provient du TP1, j'ai crée un module que je peux réutiliser dans chaque programme afin d'afficher un vecteur ou une matrice.

```
!------------------------------------------------------------------------------
! module d'affichage des matrices et des vecteurs
!------------------------------------------------------------------------------

MODULE affichage

CONTAINS

!------------------------------------------------------------------------------
!subroutine du l'affichage de la matrice a
!------------------------------------------------------------------------------

SUBROUTINE afficheMat(a,n)
IMPLICIT NONE

!déclaration des variables
real(8),dimension(:,:),intent(inout)        :: a    !matrice
integer, intent(inout)                      :: n    !rang de a
integer                                     :: i,j  !indice de boucle

!affichage de la matrice a
print *,"La matrice A :"
do i=1,n
   print *,(a(i,j), j=1,n)
end do

END SUBROUTINE afficheMat

!------------------------------------------------------------------------------
!subroutine de l'affichage du vecteur x
!------------------------------------------------------------------------------

SUBROUTINE afficheVect(x,n)
IMPLICIT NONE

!déclaration des variables
real(8),dimension(:),intent(inout)          :: x    !vecteur
```

```
integer,intent(inout)                        :: n    !rang de x
integer                                      :: i    !indice de boucle

!affichage du vecteur x
print *,"Le vecteur x :"
do i=1,n
    print *,x(i)
end do

END SUBROUTINE afficheVect

!------------------------------------------------------------------------------
!------------------------------------------------------------------------------

END MODULE affichage
```

1.4 Jeux d'essais

1.4.1 Matrice de taille 4*4 (matrice4.txt)

On utilise la matrice exmple :

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \text{ et } b = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

Methode de Jacobi

```
  Entrez le nom du fichier de donnees :
matrice4.txt
  Entrez la valeur de x0 (0.d0) :
0
  Entrez la valeur de epsilon (1.d-12) :
1.d-12
 ----------------- tp2 Resolution de systemes lineaires -----------------
 ichoix = 0 -> quitte le programme
 ichoix = 1 -> Methode de Jacobi
 ichoix = 2 -> Methode de Gauss-Seidel
 ichoix = 3 -> Methode de relaxation
 Le choix 3 effectue un tableau pour omega variant de 0.1 ? 1.9
 -------------------------------------------------------------------------
 Veuillez saisir votre choix ?
1
 Le vecteur x :
   1.9999999999981810
   1.9999999999981810
   1.9999999999981810
   1.9999999999981810
 Nombre d'iterations :           40
```

Methode de Gauss-Seidel

```
 Veuillez saisir votre choix ?
2
 Le vecteur x :
   1.9999999999996589
   1.9999999999998295
   1.9999999999998295
   1.9999999999999147
 Nombre d'iterations :           22
```

Methode de Relaxation

```
 Veuillez saisir votre choix ?
3
```

```
Valeur d'omega :   0.10000000000000001
Le vecteur x :
  1.9999999999634070
  1.9999999999643583
  1.9999999999643583
  1.9999999999652842
Nombre d'iterations :          470


Valeur d'omega :   0.20000000000000001
Le vecteur x :
  1.9999999999829465
  1.9999999999836455
  1.9999999999836455
  1.9999999999843283
Nombre d'iterations :            7


Valeur d'omega :   0.30000000000000004
Le vecteur x :
  1.9999999999897882
  1.9999999999903864
  1.9999999999903864
  1.9999999999909703
Nombre d'iterations :            3


Valeur d'omega :   0.40000000000000002
Le vecteur x :
  1.9999999999936748
  1.9999999999941904
  1.9999999999941904
  1.9999999999946902
Nombre d'iterations :            2


Valeur d'omega :   0.50000000000000000
Le vecteur x :
  1.9999999999953850
  1.9999999999958546
  1.9999999999958546
  1.9999999999963087
Nombre d'iterations :            1


Valeur d'omega :   0.59999999999999998
Le vecteur x :
  1.9999999999969105
  1.9999999999973248
```

```
  1.9999999999973248
  1.9999999999977209
Nombre d'iterations :            1


Valeur d'omega :   0.69999999999999996
Le vecteur x :
  1.9999999999981368
  1.9999999999984723
  1.9999999999984723
  1.9999999999987816
Nombre d'iterations :            1


Valeur d'omega :   0.79999999999999993
Le vecteur x :
  1.9999999999990163
  1.9999999999992539
  1.9999999999992539
  1.9999999999994578
Nombre d'iterations :            1


Valeur d'omega :   0.89999999999999991
Le vecteur x :
  1.9999999999995659
  1.9999999999997056
  1.9999999999997056
  1.9999999999998133
Nombre d'iterations :            1


Valeur d'omega :   0.99999999999999989
Le vecteur x :
  1.9999999999998528
  1.9999999999999165
  1.9999999999999165
  1.9999999999999583
Nombre d'iterations :            1


Valeur d'omega :    1.0999999999999999
Le vecteur x :
  1.9999999999999689
  1.9999999999999885
  1.9999999999999885
  1.9999999999999980
Nombre d'iterations :            1
```

```
Valeur d'omega :    1.2000000000000000
Le vecteur x :
  1.9999999999999991
  2.0000000000000013
  2.0000000000000013
  2.0000000000000013
Nombre d'iterations :           1


Valeur d'omega :    1.3000000000000000
Le vecteur x :
  2.0000000000000018
  2.0000000000000009
  2.0000000000000009
  2.0000000000000000
Nombre d'iterations :           1


Valeur d'omega :    1.4000000000000001
Le vecteur x :
  1.9999999999999998
  1.9999999999999996
  1.9999999999999996
  1.9999999999999996
Nombre d'iterations :           1


Valeur d'omega :    1.5000000000000002
Le vecteur x :
  1.9999999999999998
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
Nombre d'iterations :           1


Valeur d'omega :    1.6000000000000003
Le vecteur x :
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
Nombre d'iterations :           1


Valeur d'omega :    1.7000000000000004
Le vecteur x :
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
```

```
  2.0000000000000000
Nombre d'iterations :            1


Valeur d'omega :    1.8000000000000005
Le vecteur x :
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
Nombre d'iterations :            1


Valeur d'omega :    1.9000000000000006
Le vecteur x :
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
  2.0000000000000000
Nombre d'iterations :            1
```

1.4.2 Matrice de taille 9*9 (matrice9.txt)

On utilise la matrice exmple :

$$A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \text{ et } b = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

Methode de Jacobi

```
  Entrez le nom du fichier de donnees :
matrice9.txt
  Entrez la valeur de x0 (0.d0) :
0
  Entrez la valeur de epsilon (1.d-12) :
1.d-12
 ---------------- tp2 Resolution de systemes lineaires -----------------
 ichoix = 0 -> quitte le programme
 ichoix = 1 -> Methode de Jacobi
 ichoix = 2 -> Methode de Gauss-Seidel
 ichoix = 3 -> Methode de relaxation
 Le choix 3 effectue un tableau pour omega variant de 0.1 ? 1.9
 ----------------------------------------------------------------------
 Veuillez saisir votre choix ?
1
 Le vecteur x :
   2.7499999999954525
   3.4999999999936335
   2.7499999999954525
   3.4999999999936335
   4.4999999999909051
   3.4999999999936335
   2.7499999999954525
   3.4999999999936335
   2.7499999999954525
 Nombre d'iterations :           78
```

Methode de Gauss-Seidel

```
 Veuillez saisir votre choix ?
2
 Le vecteur x :
   2.7499999999971578
   3.4999999999971578
```

```
   2.7499999999985789
   3.4999999999971578
   4.4999999999971578
   3.4999999999985789
   2.7499999999985789
   3.4999999999985789
   2.7499999999992895
 Nombre d'iterations :           41
```

Methode de relaxation

```
 Veuillez saisir votre choix ?
3


 Valeur d'omega :   0.10000000000000001
 Le vecteur x :
   2.7499999999190963
   3.4999999998873359
   2.7499999999215534
   3.4999999998873359
   4.4999999998431068
   3.4999999998907576
   2.7499999999215534
   3.4999999998907576
   2.7499999999239364
 Nombre d'iterations :          784


 Valeur d'omega :   0.20000000000000001
 Le vecteur x :
   2.7499999999622147
   3.4999999999479052
   2.7499999999640998
   3.4999999999479052
   4.4999999999281988
   3.4999999999505360
   2.7499999999640998
   3.4999999999505360
   2.7499999999659348
 Nombre d'iterations :           12


 Valeur d'omega :   0.30000000000000004
 Le vecteur x :
   2.7499999999769607
   3.4999999999685896
   2.7499999999786078
   3.4999999999685896
   4.4999999999572156
```

```
  3.4999999999708877
  2.7499999999786078
  3.4999999999708877
  2.7499999999802096
Nombre d'iterations :            5


Valeur d'omega :   0.40000000000000002
Le vecteur x :
  2.7499999999847571
  3.4999999999794960
  2.7499999999862332
  3.4999999999794960
  4.4999999999724674
  3.4999999999815494
  2.7499999999862332
  3.4999999999815494
  2.7499999999876605
Nombre d'iterations :            3


Valeur d'omega :   0.50000000000000000
Le vecteur x :
  2.7499999999872524
  3.4999999999829923
  2.7499999999886846
  3.4999999999829923
  4.4999999999773692
  3.4999999999849889
  2.7499999999886846
  3.4999999999849889
  2.7499999999900773
Nombre d'iterations :            1


Valeur d'omega :   0.59999999999999998
Le vecteur x :
  2.7499999999918918
  3.4999999999894573
  2.7499999999931850
  3.4999999999894573
  4.4999999999863691
  3.4999999999912426
  2.7499999999931850
  3.4999999999912426
  2.7499999999944111
Nombre d'iterations :            2


Valeur d'omega :   0.69999999999999996
```

```
Le vecteur x :
  2.7499999999938778
  3.4999999999921876
  2.7499999999950555
  3.4999999999921876
  4.4999999999901110
  3.4999999999937987
  2.7499999999950555
  3.4999999999937987
  2.7499999999961529
Nombre d'iterations :            1


Valeur d'omega :   0.79999999999999993
Le vecteur x :
  2.7499999999956506
  3.4999999999946012
  2.7499999999966911
  3.4999999999946012
  4.4999999999933822
  3.4999999999960050
  2.7499999999966911
  3.4999999999960050
  2.7499999999976326
Nombre d'iterations :            1


Valeur d'omega :   0.89999999999999991
Le vecteur x :
  2.7499999999971356
  3.4999999999965827
  2.7499999999980016
  3.4999999999965827
  4.4999999999960023
  3.4999999999977183
  2.7499999999980016
  3.4999999999977183
  2.7499999999987366
Nombre d'iterations :            1


Valeur d'omega :   0.99999999999999989
Le vecteur x :
  2.7499999999982911
  3.4999999999980735
  2.7499999999989480
  3.4999999999980735
  4.4999999999978959
  3.4999999999988951
  2.7499999999989480
```

```
  3.4999999999988951
  2.7499999999994476
Nombre d'iterations :           1


Valeur d'omega :    1.0999999999999999
Le vecteur x :
  2.7499999999991109
  3.4999999999990798
  2.7499999999995479
  3.4999999999990798
  4.4999999999990967
  3.4999999999995857
  2.7499999999995479
  3.4999999999995857
  2.7499999999998277
Nombre d'iterations :           1


Valeur d'omega :    1.2000000000000000
Le vecteur x :
  2.7499999999996256
  3.4999999999996647
  2.7499999999998659
  3.4999999999996647
  4.4999999999997309
  3.4999999999999098
  2.7499999999998659
  3.4999999999999098
  2.7499999999999805
Nombre d'iterations :           1


Valeur d'omega :    1.3000000000000000
Le vecteur x :
  2.7499999999998939
  3.4999999999999352
  2.7499999999999902
  3.4999999999999352
  4.4999999999999805
  3.5000000000000115
  2.7499999999999902
  3.5000000000000115
  2.7500000000000133
Nombre d'iterations :           1


Valeur d'omega :    1.4000000000000001
Le vecteur x :
  2.7499999999999969
```

```
  3.5000000000000151
  2.7500000000000133
  3.5000000000000151
  4.5000000000000266
  3.5000000000000142
  2.7500000000000133
  3.5000000000000142
  2.7500000000000044
Nombre d'iterations :            1


Valeur d'omega :    1.5000000000000002
Le vecteur x :
  2.7500000000000124
  3.5000000000000120
  2.7500000000000036
  3.5000000000000120
  4.5000000000000071
  3.4999999999999982
  2.7500000000000036
  3.4999999999999982
  2.7499999999999964
Nombre d'iterations :            1


Valeur d'omega :    1.6000000000000003
Le vecteur x :
  2.7500000000000027
  3.4999999999999987
  2.7499999999999964
  3.4999999999999987
  4.4999999999999929
  3.4999999999999951
  2.7499999999999964
  3.4999999999999951
  2.7499999999999982
Nombre d'iterations :            1


Valeur d'omega :    1.7000000000000004
Le vecteur x :
  2.7499999999999969
  3.4999999999999951
  2.7499999999999973
  3.4999999999999951
  4.4999999999999956
  3.4999999999999996
  2.7499999999999973
  3.4999999999999996
  2.7500000000000009
```

```
Nombre d'iterations :            1


Valeur d'omega :    1.8000000000000005
Le vecteur x :
  2.7499999999999978
  3.5000000000000004
  2.7500000000000018
  3.5000000000000004
  4.5000000000000027
  3.5000000000000022
  2.7500000000000018
  3.5000000000000022
  2.7500000000000009
Nombre d'iterations :            1


Valeur d'omega :    1.9000000000000006
Le vecteur x :
  2.7500000000000018
  3.5000000000000031
  2.7500000000000009
  3.5000000000000031
  4.5000000000000027
  3.5000000000000000
  2.7500000000000009
  3.5000000000000000
  2.7499999999999991
Nombre d'iterations :            1
```

1.5 Conclusions

On observe une nette décroissance du nombre d'itération effectués selon les methodes utilisées. En effet, la methode de Jacobi utilise plus d'itération que la methode de Gauss-Seidel qui est elle même moins efficace que la methode de relaxation. Pourtant, même si ces methodes gagnent en efficacité elles présentent quelques inconvénients :

- La methode de Jacobi nécessite le stockage de x^k et x^{k+1}, ce qui prend plus de place dans la mémoire.
- La methode de Gauss-Seidel est encore trop séquentielle et perd en efficacité.
- La methode de relaxation est optimisée pour une certaine valeur de ω qu'il faut trouver.

On remarque, de plus, que la methode de relaxation est plus optimisée pour une valeur de ω proche de $1.d0$.

Le programme Fortran facilite la manipulation des matrices et des formules mathématiques complexes (surtout au niveau du séquntiel). Il pose néammoins des problèmes lors de l'affichage. Par exemple, l'affichage des matrices de taille 4 et de taille 9 impose 8 chiffres après la virgule, j'ai essayé de changer de format (F12.8) mais l'affichage se fait alors par colonnes.

On peut comparer ces methodes avec la methode LU présentée dans le TP2 pour la matrice de taille 4 et celle de taille 9, on remarque alors que le nombre d'itération est très réduit.