1 Élimination de Gauss-Jordan (avec pivot partiel)

On cherche à inverser la matrice carrée $n \times n$ M en procédant méthodiquement à des éliminations par combinaisons linéaires de lignes.

1.1 Le principe

Pour cela on utilise n étapes successives. L'étape numéro p (où $p = 1, \cdots, n$) se décompose ainsi :

1. recherche de l'élément maximum (en valeur absolue) dans la colonne p sur les lignes $q \geqslant p$: c'est le « pivot » ;
2. permutation des lignes q et p pour mettre le pivot sur la diagonale (si nécessaire) ;
3. division de la ligne p par le pivot, de sorte que $m_{pp} = 1$;
4. remplacement des lignes $q \neq p$ par la combinaison linéaire de la ligne q et de la ligne p : $\ell_q \leftarrow \ell_q - m_{pq}\ell_p$ (soit $m_{qr} \leftarrow m_{qr} - m_{qp} \times m_{pr}$) qui permet d'annuler les éléments m_{pq} pour $q \neq p$.

Il se trouve que chacune de ces opérations sur les lignes est obtenue en multipliant à gauche la matrice à inverser par une matrice « opérant sur les lignes ». Si l'on effectue cette opération en même temps sur M et sur la matrice identité $\mathbb{1}_n$, on obtient, progressivement à la place de M la matrice $\mathbb{1}_n$, et à la place de $\mathbb{1}_n$ une matrice N qui est l'inverse de M.

1.2 Illustration

Considérons par exemple la matrice suivante, avec $n = 4$:

$$\mathsf{M} = \begin{pmatrix} \boxed{1} & 2 & 3 & 2 \\ -1 & 2 & -2 & -1 \\ 0 & 3 & -1 & 1 \\ -1 & 3 & -2 & 0 \end{pmatrix} \qquad (\det \mathsf{M} = 5) \,. \tag{1}$$

Le premier pivot est l'élément m_{11}, ce qui rend inutiles les étapes 1 et 2. On construit aisément la matrice L_1 telle que :

$$\mathsf{M}' \leftarrow \mathsf{L}_1 \cdot \mathsf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \cdot \mathsf{M} = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 0 & 4 & 1 & 1 \\ 0 & 3 & -1 & 1 \\ 0 & \boxed{5} & 1 & 2 \end{pmatrix} \quad \text{et} \quad \mathsf{N} \leftarrow \mathsf{L}_1 \cdot \mathbb{1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

et M a déjà sa première colonne semblable à celle de $\mathbb{1}_4$.

Le *second pivot* est l'élément $m'_{52} = 5$. On doit donc multiplier par la matrice de permutation $\mathsf{S}(2,4)$ pour échanger les lignes, puis par la matrice $\mathsf{P}_2 = \mathrm{Diag}(1, 1/5, 1, 1)$ pour ramener le pivot à 1, enfin par la matrice L_2, où :

$$\mathsf{S}(2,4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad \text{et} \quad \mathsf{L}_2 = \begin{pmatrix} 1 & -2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -4 & 0 & 1 \end{pmatrix}$$

et donc :

$$\mathsf{M}' \leftarrow \mathsf{L}_2 \cdot \mathsf{P}_2 \cdot \mathsf{S}(2,4) \cdot \mathsf{M}' = \begin{pmatrix} 1 & 0 & 13/5 & 6/5 \\ 0 & 1 & 1/5 & 2/5 \\ 0 & 0 & \boxed{\text{-8/5}} & -1/5 \\ 0 & 0 & 1/5 & -3/5 \end{pmatrix} \quad \text{et} \quad \mathsf{N} \leftarrow \mathsf{L}_2 \cdot \mathsf{P}_2 \cdot \mathsf{S}(2,4) \cdot \mathsf{N} = \begin{pmatrix} 3/5 & -2/5 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 \\ -3/5 & -3/5 & 1 & 0 \\ 1/5 & -4/5 & 0 & 1 \end{pmatrix} . \tag{3}$$

Le *troisième pivot* est alors $m'_{33} = -8/5$ (pas de permutation), il est ramené à 1 avec la matrice $\mathsf{P}_3 = \mathrm{Diag}(1, 1, -5/8, 1)$ et il vient :

$$\mathsf{L}_3 = \begin{pmatrix} 1 & 0 & -13/5 & 0 \\ 0 & 1 & -1/5 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/5 & 1 \end{pmatrix} ,$$

d'où :

$$\mathsf{M}' \leftarrow \mathsf{L}_3 \cdot \mathsf{P}_3 \cdot \mathsf{M}' = \begin{pmatrix} 1 & 0 & 0 & 7/8 \\ 0 & 1 & 0 & 3/8 \\ 0 & 0 & 1 & 1/8 \\ 0 & 0 & 0 & \boxed{-5/8} \end{pmatrix} \quad \text{et} \quad \mathsf{N} \leftarrow \mathsf{L}_3 \cdot \mathsf{P}_3 \cdot \mathsf{N} = \begin{pmatrix} -3/8 & 0 & 13/8 & -11/8 \\ 1/8 & 0 & 1/8 & 1/8 \\ 3/8 & 0 & -5/8 & 3/8 \\ 1/8 & 1 & 1/8 & -7/8 \end{pmatrix} . \tag{4}$$

Le *quatrième pivot* est alors $m'_{44} = -5/8$, il est ramené à 1 avec la matrice $\mathsf{P}_4 = \mathrm{Diag}(1, 1, 1, -8/5)$ et il vient :

$$\mathsf{L}_4 = \begin{pmatrix} 1 & 0 & 0 & -7/8 \\ 0 & 1 & 0 & -3/8 \\ 0 & 0 & 1 & -1/8 \\ 0 & 0 & 0 & 1 \end{pmatrix} ,$$

si bien que :

$$\mathsf{M}' \leftarrow \mathsf{L}_4 \cdot \mathsf{P}_4 \cdot \mathsf{M}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbb{1}_4 \quad \text{et} \quad \mathsf{N} \leftarrow \mathsf{L}_4 \cdot \mathsf{P}_4 \cdot \mathsf{N} = \frac{1}{5} \begin{pmatrix} -1 & 7 & 9 & -13 \\ 1 & 3 & 1 & -2 \\ 2 & 1 & -3 & 1 \\ -1 & -8 & -1 & 7 \end{pmatrix} , \tag{5}$$

cette dernière étant la matrice inverse de M.

1.3 Implémentation en C

`gaussj_double.c`, version en double précision, d'après *Numerical recipes*

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <math.h>
4  #include "mnitab.h"
5  #include "gaussj_double.h"
6  #define SWAP(a,b) {temp=(a);(a)=(b);(b)=temp;}
7
8  void gaussj_double(double **mat, double *v, double **matinv, double *u, int n) {
9     int  col, ligne, lmax , k ;
10    double **mtmp = NULL;
11    double big , pivinv , dum , temp ;
12    mtmp=double2d(n,n) ;
13    /* creation de la matrice de travail mtmp         */
14    /* et initialisation de l'inverse matinv a l'identite */
15    for (ligne=0;ligne<n;ligne++) {
16         u[ligne]=v[ligne] ;
17         for (col=0;col<n;col++) { mtmp[ligne][col]=mat[ligne][col] ;
18                                   matinv[ligne][col]=0.0 ;
19         }
20    }
21    for (col=0;col<n;col++) { matinv[col][col]=1.0 ; }
22
23    /* boucle principale sur les colonnes */
24    for (col=0;col<n;col++) {
25       /* recherche de la ligne du max. dans la partie inf. de la colonne en cours */
26       big=0.0;
27       for (ligne=col;ligne<n;ligne++) {
28         if ( fabs(mtmp[ligne][col]) >= big ) {
29            big=fabs( mtmp[ligne][col] );
30            lmax=ligne ;
31         }
32       }
33       /* le pivot est mtmp[lmax][col]   */
```

```
      printf(" pivot %d %d %f \n",col,lmax,mtmp[lmax][col]);
      /* test de regularite de la matrice */
      if( mtmp[lmax][col] == 0.0) {
        fprintf(stderr,"gaussj: matrice singuliere\n");
        exit(1);
      }
      pivinv = 1.0/mtmp[lmax][col] ;
      /* echange des lignes lmax et col pour placer le pivot sur la diagonale */
      for (k=0;k<n;k++) {
        SWAP( mtmp[col][k] , mtmp[lmax][k] );
        SWAP( matinv[col][k] , matinv[lmax][k] );
      }
        SWAP( u[col] , u[lmax] );
      /* ramener le pivot a 1 (apres echange) */
      u[col] *= pivinv ;
      for (k=0;k<n;k++) {
         mtmp[col][k] *= pivinv ;
         matinv[col][k] *= pivinv ;
      }
      /* elimination dans les lignes d'indice different de l'indice de colonne */
      for (ligne=0;ligne<n;ligne++) {
          if( ligne != col ) {
             dum = mtmp[ligne][col] ;
             u[ligne] -= u[col]*dum ;
             for (k=0;k<n;k++) {
                  matinv[ligne][k] -= matinv[col][k]*dum ;
                  mtmp[ligne][k] -= mtmp[col][k]*dum ;
             }
          }
      }
   }
 }
#undef SWAP
```

1.4 Implémentation en fortran 90

gaussj.f90, d'après *Numerical recipes*

```
module gaussj
implicit none
private :: echange, echange_scal
contains
  subroutine gaussj_real(mat, v, matinv, u)
    real, dimension(:,:), intent(in)    :: mat ! matrice carree n x n
    real, dimension(:),   intent(in)    :: v  ! second membre
    real, dimension(:,:), intent(inout) :: matinv     ! matrice inverse
    real, dimension(:),   intent(out)   :: u     ! solution
    real, dimension(size(mat,1),size(mat,1))  :: mtmp  ! matrice auxiliare
    ! en entree : mat est la matrice a inverser
    !             matinv une matrice de meme taille
    ! en sortie : matinv est l'inverse de mat
    ! mise sous forme diagonale (Gauss-Jordan) par pivot partiel
    ! (echange de lignes sans echange de colonnes)
    !  u est la solution de mat u = v
    integer                             :: n
    real                                :: pivot
    integer                             :: ligne, col, lmax
    integer, dimension(1)               :: vlmax
    n = size(mat, 1)
```

```
22      if ( size(mat, 2) /= n ) stop 'mat non carree'
23      if ( size(matinv, 1) /= n ) stop 'matinv de dimension incorrecte'
24      if ( size(matinv, 2) /= n ) stop 'matinv de dimension incorrecte'
25      ! remplir matinv avec la matrice identite n x n
26      mtmp(:,:) = mat(:,:)
27      u(:) = v(:)
28      matinv(:,:) = 0.
29      do ligne=1, n
30        matinv(ligne, ligne) = 1.
31      end do
32      do col = 1, n  ! boucle principale sur les colonnes
33        ! recherche de la ligne du max dans la partie inferieure de la colonne
34        vlmax = maxloc(abs(mtmp(col:n, col)))
35        ! attention, position relative a la borne inferieure = col
36        ! par exemple: vlmax(1) = 2 => max sur la ligne (col+1)
37        lmax = vlmax(1) + col - 1
38        pivot = mtmp(lmax, col)
39        ! print *, 'pivot : ', col, lmax, pivot
40        if (pivot == 0.) stop 'matrice singuliere'
41        ! echange des lignes lmax et col pour placer ce max sur la diagonale
42        call echange(mtmp(col,:), mtmp(lmax,:))
43        call echange(matinv(col,:), matinv(lmax,:))
44        call echange_scal(u(col), u(lmax))
45        ! ramener le pivot a 1 (apres echange)
46        mtmp(col,:) = mtmp(col, :) / pivot
47        matinv(col,:) = matinv(col,:) / pivot
48        u(col) = u(col) / pivot
49        do ligne = col + 1, n  ! elimination dans les lignes plus basses
50          ! attention a l'ordre: ne pas modifier mtmp(ligne, col) avant !
51          matinv(ligne, :) = matinv(ligne, :) - mtmp(ligne, col) * matinv(col, :)
52          u(ligne) =    u(ligne)    - mtmp(ligne, col) * u(col)
53          mtmp(ligne, :) = mtmp(ligne, :) - mtmp(ligne, col) * mtmp(col, :)
54        end do
55        do ligne = 1, col - 1  ! elimination dans les lignes plus hautes
56          ! attention a l'ordre: ne pas modifier mtmp(ligne, col) avant !
57          matinv(ligne, :) = matinv(ligne, :) - mtmp(ligne, col) * matinv(col, :)
58          u(ligne) =    u(ligne)    - mtmp(ligne, col) * u(col)
59          mtmp(ligne, :) = mtmp(ligne, :) - mtmp(ligne, col) * mtmp(col, :)
60        end do
61      end do
62    end subroutine gaussj_real
63
64    subroutine echange(lig1, lig2)
65      real, dimension(:), intent(inout) :: lig1, lig2
66      real, dimension(size(lig1))       :: tmp
67      tmp(:) = lig1(:)
68      lig1(:) = lig2(:)
69      lig2(:) = tmp(:)
70    end subroutine echange
71
72    subroutine echange_scal(x1, x2)
73      real,  intent(inout) :: x1, x2
74      real                 :: tmp
75      tmp = x1
76      x1 = x2
77      x2 = tmp
78    end subroutine echange_scal
79  end module gaussj
```