Fiche n^0 8. Résolution d'un système linéaire - stockage des matrices creuses

Exercice 1 – Résolution d'un système linéaire par une méthode itérative

On cherche à résoudre un système du type $Ax = b$ (on suppose A inversible) : Soit D la diagonale, supposée à coefficients tous non-nuls, de la matrice A, soit L sa partie sous-diagonale ($L_{i,j} = 0$ pour $j \geq i$), et soit U sa partie sur-diagonale ($U_{i,j} = 0$ pour $j \leq i$).

On établit une famille de méthodes itératives de résolution du système $Ax = b$ par décomposition de A sous la forme $A = M - N$, avec M inversible. Le système est équivalent à

$$Mx = Nx + b$$

et un alogrithme itératif que l'on peut envisager est le suivant : Soit $x^{(0)}$ un vecteur quelconque. Pour tout $k \geq 0$, on résout :

$$Mx^{(k+1)} = Nx^{(k)} + b$$

a) Identifier les matrices M et N dans l'algorithme de Jacobi vu dans la fiche n^0 7.

b) On pose $e^{(k)} = x^{(k)} - x^*$. Ecrire la relation linéaire qui lie $e^{(k+1)}$ à $e^{(k)}$ (sous la forme $e^{(k+1)} = Qe^{(k)}$, où Q est une matrice à préciser).

c) La méthode de Gauss-Seidel consiste à choisir $M = (L + D)$ et $N = -U$. Justifier que M est inversible et que résoudre les systèmes du type $Mx = y$ est aisé. Proposer un algorithme qui réalise cette résolution et implémenter la méthode de Gauss-Seidel. On utilisera cette méthode pour résoudre le problème de différences finies vu dans la fiche n^0 7.

Exercice 2 – Stockage d'une matrice creuse

Dans la fiche n^0 7, nous avons utilisé une matrice pleine pour stocker la matrice associée à la discrétisation différences finies étudiée.

Or, ceci n'est pas optimal, car cette matrice contient essentiellement des termes nuls et seulement quelques termes non-nuls. Il est donc beaucoup moins coûteux de ne stocker que les termes non-nuls de la matrice, mais il faut dans le même temps pouvoir se répérer dans celle-ci : Appelons A_{math} une matrice au sens mathématique (tableau de $p \times q$ éléments) et A_{inf} sa représentation informatique ; nous allons stocker dans $A_{\text{inf}}[i][\,]$ les éléments non-nuls de la $i$ème ligne de A_{math}, pour $i \in [0, p-1]$. Il faut donc pouvoir associer $A_{\text{inf}}[i][j]$ à $(A_{\text{math}})_{i,k}$ pour un certain $k \in [1, q]$.

Nous proposons pour cela de d'utiliser :

1. *Un tableau d'entiers de taille p indiquant le nombre de termes non-nuls de la ligne i de A_{math}. Appelons ce tableau NNZ. Informatiquement, $A_{\text{inf}}[i][\,]$ devra donc être un tableau de taille $NNZ[i]$ dans sa deuxième composante.*
2. *Pour chaque ligne $i \in [0, p-1]$, un tableau d'entiers indiquant les numéros des colonnes de A_{math} où se situent les éléments $A_{\text{inf}}[i][j]$ pour $j \in [0, NNZ[i] - 1]$.*

Le code va donc ressembler à ceci

```
double**  Ainf = NULL;
int*      NNZ  = NULL;
int**     NC   = NULL;

Ainf = malloc( p * sizeof( double* ) );
NNZ  = malloc( p * sizeof( int     ) );
NC   = malloc( p * sizeof( int*    ) );
```

```
// il faut se débrouiller pour connaitre NNZ[i]

for(i = 0 ; i < p ; i++){
    Ainf[i] = malloc( NNZ[i] * sizeof ( double ) ); // il faut connaitre NNZ[i]
    NC[i]   = malloc( NNZ[i] * sizeof ( int    ) );
}
```

Pour tout i de $[0, p-1]$ et tout j de $[0, NNZ[i]-1]$, la valeur $A_{\text{inf}}[i][j]$ est égale à $(A_{\text{math}})_{i,NC[i][j]}$.

Reprendre la matrice liée à la discrétisation par différences finies de la fiche n^0 7 et la stocker sous forme creuse. Écrire un algorithme de Jacobi et de Gauss-Seidel utilisant cette forme de stockage ; Comparer les temps de calcul des deux algorithmes, entre eux, et lorsque le stockage est plein ou creux.

Exercice 3 – Algorithmes Gradient conjugué et BICGSTAB

Lorsque la matrice du système linéaire à résoudre n'est pas à diagonale dominante, il peut arriver que la méthode de Jacobi ou de Gauss-Seidel ne converge pas. Dans ce cas, lorsque la matrice est symétrique, un algorithme qui peut convenir est la méthode du Gradient Conjugué. Lorsque la matrice n'est pas symétrique, un algorithme populaire est l'algorithme BICGSTAB, dont on peut trouver la formulation sur Internet.

Ces algorithmes nécessitent à chaque itération un produit matrice - vecteur. Proposer une implémentation de ce produit lorsque la matrice est stockée de façon creuse comme expliqué dans l'exercice précédent.

Proposer une implémentation de ces algorithmes et tester la résolution du problème de différences finies.