# .1 Méthode de Gauss-Seidel

## .1.1 Introduction à la méthode de Gauss-Seidel

La méthode de Gauss-Seidel est une méthode itérative pour résoudre les systèmes linéaires de la forme $Ax = b$, où $A$ est une matrice carrée d'ordre $n$ et $x, b$ sont des vecteurs de $\mathbb{R}^n$. C'est une méthode qui génère une suite qui converge vers la solution de ce système lorsque celle-ci en a une et lorsque les conditions de convergence suivantes sont satisfaites (quels que soient le vecteur $b$ et le point initial $x^0$) :

— Si la matrice $A$ est symétrique définie positive,
— Si la matrice $A$ est à diagonale strictement dominante.

## .1.2 Mise en place des matrices pour la méthode de Gauss-Seidel

Soit $Ax = b$ le système linéaire à résoudre, où $A \in \mathcal{M}_{n,n}$ et $b \in \mathcal{M}_{n,1}$. On cherche $x \in \mathcal{M}_{n,1}$ solution du système. Dans un premier temps, on va écrire $A$ sous la forme $A = D - E - F$ où $D$ est une matrice diagonale, $E$ est une matrice triangulaire inférieure, et $F$ est une matrice triangulaire supérieure.
On peut alors écrire :

$$Ax = b \tag{1}$$
$$\Leftrightarrow (D - E - F)x = b \tag{2}$$
$$\Leftrightarrow Dx = b - (E + F)x \tag{3}$$
$$\Leftrightarrow x = D^{-1}[b - (E + F)x] \tag{4}$$

On définit ensuite une suite de vecteurs $(x^k)$ en choisissant un vecteur $x^0$ et par la formule de récurrence :

$$x_i^{k+1} = \frac{1}{a_{i,i}} \left( b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) \tag{5}$$

## .1.3 Algorithme

Pour résoudre un système $Ax = b$, avec $A \in \mathcal{M}_n$ et $b \in \mathcal{M}_{n,1}$, on s'appuie sur l'algorithme suivant en posant :

— un vecteur initial $x^{(0)}$ choisi au préalable,
— l'erreur à l'itération k=0 calculée par $\varepsilon^{(0)} = \|Ax^{(0)} - b\|$,
— une variable $k$ qui sera notre compteur d'itération.

0 $x^{(0)} = x_0 \in \mathcal{M}_{n,1}$
1 $\varepsilon^{(0)} = \varepsilon$ (erreur)
2 $k = 0$
3 Tant Que $(\varepsilon^{(k)} >= \varepsilon)$ faire :
4 Pour $i = 1$ a $n$ :
5 $x_i^{(k+1)} = \frac{1}{a_{i,i}} \left[ b_i - \left( \sum_{j=i+1}^{n} a_{i,j} x_j^{(k)} + \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} \right) \right]$ pour $i = 1, ..., n$
6 $\varepsilon^{(k+1)} = \|Ax^{(k+1)} - b\|$
7 $k = k + 1$
8 Fin Tant Que

### .1.4 Résolution manuelle

Soit $A = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 3 & 3 \\ 3 & 7 & 8 \end{pmatrix} \in \mathcal{M}_3(\mathbb{R})$, et $b = \begin{pmatrix} 2 \\ 2 \\ 8 \end{pmatrix} \in \mathcal{M}_{3,1}(\mathbb{R})$

Calculons le vecteur $x^{(1)}$ (vecteur x trouvé après 1 itération de l'algorithme) solution du système $Ax = b$,

en prenant comme point initial $x^{(0)} = (0, 0, 0)$ :

**Résolution par le calcul itératif**

Dans cette sous-partie, nous résolverons le système de la même manière que le fait l'algorithme sus-cité.
Pour obtenir le vecteur $x^{(1)}$ (obtenu à l'itération $k = 1$), il nous faut obtenir $x_1^{(1)}$, $x_2^{(1)}$,$x_3^{(1)}$ par la formule suivante :

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left[ b_i - \left( \sum_{j=i+1}^{n} a_{i,j} x_j^{(k)} + \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} \right) \right] \text{ pour } i = 1, ..., 3$$

Pour $i = 1$ :

$$x_1^{(1)} = \frac{1}{a_{1,1}} \left[ b_1 - \left( \sum_{j=2}^{3} a_{1,j} x_j^{(0)} + \sum_{j=1}^{0} a_{1,j} x_j^{(1)} \right) \right] \tag{6}$$

$$= \frac{1}{1} \left[ 2 - \left( a_{1,2} x_2^{(0)} + a_{2,2} x_2^{(0)} + 0 \right) \right] \tag{7}$$

$$= 2 - 2 \times 0 - 3 \times 0 = 2 \tag{8}$$

Pour $i = 2$ :

$$x_2^{(1)} = \frac{1}{a_{2,2}} \left[ b_2 - \left( \sum_{j=3}^{3} a_{2,j} x_j^{(0)} + \sum_{j=1}^{1} a_{2,j} x_j^{(1)} \right) \right] \tag{9}$$

$$= \frac{1}{3} \left[ 2 - \left( a_{2,3} x_3^{(0)} + a_{2,1} x_1^{(1)} \right) \right] \tag{10}$$

$$= \frac{1}{3} \left( 2 - 3 \times 0 - 1 \times 2 \right) \tag{11}$$

$$= \frac{1}{3} \times 0 = 0 \tag{12}$$

Pour $i = 3$ :

$$x_3^{(1)} = \frac{1}{a_{3,3}} \left[ b_3 - \left( \sum_{j=4}^{3} a_{3,j} x_j^{(0)} + \sum_{j=1}^{2} a_{3,j} x_j^{(1)} \right) \right] \tag{13}$$

$$= \frac{1}{8} \left[ 8 - \left( 0 + a_{3,1} x_1^{(1)} + a_{3,2} x_2^{(1)} \right) \right] \tag{14}$$

$$= \frac{1}{8}\left(8 - 3 \times 2 - 7 \times 0\right) \tag{15}$$

$$= \frac{1}{8} \times 2 = \frac{1}{4} \tag{16}$$

Conclusion :

Nous avons $x_1^{(1)} = 2$, $x_2^{(1)} = 0$, $x_3^{(1)} = \frac{1}{4}$. Et donc, $x^{(1)} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ \frac{1}{4} \end{pmatrix} 0$

**Résolution par le calcul matriciel**

Dans la section .1.2, nous avons vu que l'on pouvait décomposer la matrice $A$ par une matrice diagonale $D$, une matrice triangulaire inférieure $E$, et une matrice triangulaire supérieure $F$. Ceci fait, nous pouvons obtenir le vecteur $x$ par la formule suivante :

$$x^{(k+1)} = D^{-1}[b - (E + F)x^{(k)}]$$

Nous avons alors :

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 3 & 3 \\ 3 & 7 & 8 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 8 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & -2 & -2 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{pmatrix}}_{E} - \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -3 & -7 & 0 \end{pmatrix}}_{F}$$

Nous obtenons alors :

$$x^{(1)} = \begin{pmatrix} \frac{1}{1} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{8} \end{pmatrix} \left[ \begin{pmatrix} 2 \\ 2 \\ 8 \end{pmatrix} - \left( \begin{pmatrix} 0 & -2 & -2 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -3 & -7 & 0 \end{pmatrix} \right) \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right] \tag{17}$$

$$= \begin{pmatrix} \frac{1}{1} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{8} \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 8 \end{pmatrix} \tag{18}$$

$$= \begin{pmatrix} 2 \\ \frac{2}{3} \\ 1 \end{pmatrix} \tag{19}$$

*Remarque : Au fur et à mesure des itérations, le vecteur $x$ donné par le calcul itératif effectué dans la partie .1.4 se rapproche de la solution donnée par le précédent calcul. Il est alors normal que le vecteur trouvé au bout de la première itération soit différent du vecteur trouvé ci-dessus.*

## .1.5 Implémentation

**Commentaires fonctionnels**

*Note : L'implémentation qui suit utilise exactement les mêmes fonctions usuelles de manipulation de matrice que l'implémentation de l'algorithme de Gauss décrit dans la section* **??**. *De plus, dans cette implémentation, nous définirons une variable $k$ qui sera notre compteur d'itération et qui permettra l'arrêt de notre code si la suite ne converge pas. Enfin, notre variable erreur $\varepsilon$ sera mise à jour à chaque itération de la manière suivante :*

$$\varepsilon^{(k)} = p^{(k)} = Max_{i=1,\ldots,n}|\overline{x}_i - \widetilde{x}_i^k|$$

Nous détaillerons dans cette section uniquement les fonctions dites "non-usuelles" qui vont nous servir pour l'implémentation de l'algorithme de Gauss-Seidel. Il s'agit ici de la fonction de mise à jour de notre variable erreur et de la fonction implémentant l'algorithme de Gauss-Seidel.

**Fonction *majEpsilon :***

La fonction **majEpsilon** permet de mettre à jour la variable d'erreur $\varepsilon$ lors de l'exécution de notre algorithme. Grâce au vecteur $x^{(k)}$ qui représente la solution actuelle de notre système d'équations, la fonction calcule la différence absolue entre chaque élément $x_i^{(k)}$ et 1. Cela permet de mesurer à quel point les valeurs actuelles se rapprochent de 1, qui est notre valeur cible pour les solutions convergentes. La fonction conserve le maximum de ces différences absolues en tant que mesure d'erreur afin de mettre à jour notre variable $\varepsilon$. Cela permet de contrôler la précision de l'algorithme et de décider quand il a convergé de manière satisfaisante vers la solution recherchée. La fonction **majEpsilon** joue donc un rôle dans la détermination du critère d'arrêt de l'algorithme. Voici son implémentation en C :

```
0  float majEpsilon(float** matXk, int row){
1    float maxforEps=0;
2    for(int i=0; i<row; i++){
3        float soustr=fabs(1-matXk[i][0]);
4        printf("%f\n", matXk[i][0]);
5        if (soustr>maxforEps){
6            maxforEps=soustr;
7        }
8    }
9    return maxforEps;
10 }
```

**Fonction *gaussSeidel :***

La fonction **gaussSeidel** implémente l'algorithme de Gauss-Seidel tel que décrit précédemment dans la section .1.3. Par la programmation itérative, notre algorithme mettra à jour les solutions actuelles jusqu'à ce que l'erreur minimale définie soit atteinte ou que le nombre maximal d'itérations soit atteint. Voici son implémentation en C :

```
0    float** gaussSeidel(float **matA, float **matB, float **matXk, int row, int column,
1                        int nbIterMax)
2    {
3      //CREATING OUR SOLUTION VECTOR AT ITERATION k
4      float **matXk1=createMatrix(row, 1);
5
6      //INITIALIZING OUR ERROR VARIABLE AND ITERATION COUNTER
7      float epsilon=majEpsilon(matXk, row);;
8      int iter=0;
9
10     while ((epsilon>=pow(10,-6)) && (iter<nbIterMax)){
11         for(int i=0; i<row ; i++){
12             float sumF=0;
13             float sumE=0;
14
15             //CALCULATION OF F
16             for(int j=i+1; j<row ; j++){
17                 sumF+=matA[i][j]*matXk[j][0];
18             }
19
20             //CALCULATION OF E
21             for(int j=0; j<i ; j++){
22                 sumE+=matA[i][j]*matXk1[j][0];
23             }
24
25             //CALCULATION OF ELEMENT X_i^(k)
26             matXk1[i][0]=(matB[i][0]-sumF-sumE)/matA[i][i];
27
28         }
```

```
29
30          //UPDATING OUR SOLUTION VECTOR
31          for(int k=0; k<row ; k++){
32              matXk[k][0]=matXk1[k][0];
33          }
34
35          //UPDATING OUR ERROR VARIABLE AND ITERATION COUNTER
36          epsilon=majEpsilon(matXk, row);
37          iter+=1;
38      }
39
40      //RETURN THE SOLUTION VECTOR
41      return matXk;
42  }
```

### .1.6 Exemples d'exécution

Soient les matrices $A$ données dans le TP et dans l'annexe du document (section **??**). En résolvant le système $Ax = b$ en question, nous obtenons respectivement les résultats suivants :

Listing 1 – $A_1X = B$ results

```
0                   A matrix
1   3.000000    0.000000    4.000000
2   7.000000    4.000000    2.000000
3   -1.000000    1.000000    2.000000
4                   B matrix
5   7.000000
6   13.000000
7   2.000000
8                   SOLVING
9                   SOLUTION VECTOR X
10  -nan
11  -nan
12  -nan
13  Temps d'execution : 0.000515 secondes
```

Listing 2 – $A_2X = B$ results

```
0                   A matrix
1   -3.000000    3.000000    -6.000000
2   -4.000000    7.000000    8.000000
3   5.000000    7.000000    -9.000000
4                   B matrix
5   -6.000000
6   11.000000
7   3.000000
8                   SOLVING
9                   SOLUTION VECTOR X
10  -nan
11  -nan
12  -nan
13  Temps d'execution : 0.000419 secondes
```

Listing 3 – $A_3X = B$ results

```
0                   A matrix
1   4.000000    1.000000    1.000000
2   2.000000    -9.000000    0.000000
3   0.000000    -8.000000    6.000000
4                   B matrix
5   6.000000
```

```
 6   −7.000000
 7   −2.000000
 8                    SOLVING
 9                    SOLUTION VECTOR X
10   1.000000
11   1.000000
12   1.000000
13   Temps d'execution : 0.000399 secondes
```

Listing 4 – $A_4X = B$ results

```
 0                    A matrix
 1   7.000000    6.000000    9.000000
 2   4.000000    5.000000    −4.000000
 3   −7.000000    −3.000000    8.000000
 4                    B matrix
 5   22.000000
 6   5.000000
 7   −2.000000
 8                    SOLVING
 9                    SOLUTION VECTOR X
10   1.000000
11   0.999999
12   1.000000
13   Temps d'execution : 0.000408 secondes
```

Listing 5 – $A_5X = B$ results

```
 0                    A matrix
 1   1.000000    0.500000    0.250000
 2   0.500000    1.000000    0.000000
 3   0.250000    0.000000    1.000000
 4                    B matrix
 5   1.750000
 6   1.500000
 7   1.250000
 8                    SOLVING
 9                    SOLUTION VECTOR X
10   1.000001
11   1.000000
12   1.000000
13   Temps d'execution : 0.000431 secondes
```

Listing 6 – $A_6X = B$ results

```
 0                    A matrix
 1   1.000000    0.500000    0.250000    0.125000    0.062500    0.031250
 2   0.500000    1.000000    0.000000    0.000000    0.000000    0.000000
 3   0.250000    0.000000    1.000000    0.000000    0.000000    0.000000
 4   0.125000    0.000000    0.000000    1.000000    0.000000    0.000000
 5   0.062500    0.000000    0.000000    0.000000    1.000000    0.000000
 6   0.031250    0.000000    0.000000    0.000000    0.000000    1.000000
 7                    B matrix
 8   1.968750
 9   1.500000
10   1.250000
11   1.125000
12   1.062500
13   1.031250
14                   SOLVING
15                   SOLUTION VECTOR X
16   1.000001
17   1.000000
18   1.000000
```

```
 1.000000
 1.000000
 1.000000
Temps d'execution : 0.000467 secondes
```

Listing 7 – $A_7X = B$ results

```
                A matrix
1.000000  0.500000  0.250000  0.125000  0.062500  0.031250  0.015625  0.007812
0.500000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.250000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.125000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000
0.062500  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000
0.031250  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000
0.015625  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000
0.007812  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
                B matrix
1.992187
1.500000
1.250000
1.125000
1.062500
1.031250
1.015625
1.007812
                SOLVING
                SOLUTION VECTOR X
1.000001
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
Temps d'execution : 0.000526 secondes
```

Listing 8 – $A_8X = B$ results

```
                A matrix
1.000000  0.500000  0.250000  0.125000  0.062500  0.031250  0.015625  0.007812  0.003906  0.001953
0.500000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.250000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.125000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.062500  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.031250  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000
0.015625  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  0.000000
0.007812  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  0.000000
0.003906  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  0.000000
0.001953  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
                B matrix
1.998046
1.500000
1.250000
1.125000
1.062500
1.031250
1.015625
1.007812
1.003906
1.001953
                SOLVING
                SOLUTION VECTOR X
1.000001
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
Temps d'execution : 0.000572 secondes
```

Listing 9 – $A_9X = B$ results

```
                    A matrix
 3.000000   -1.000000    0.000000
-2.000000    3.000000   -1.000000
 0.000000   -2.000000    3.000000
                    B matrix
 2.000000
 0.000000
 1.000000
                    SOLVING
                    SOLUTION VECTOR X
 1.000000
 0.999999
 1.000000
Temps d'execution : 0.000424 secondes
```

Listing 10 – $A_10X = B$ results

```
                    A matrix
 3.000000   -1.000000    0.000000    0.000000    0.000000    0.000000
-2.000000    3.000000   -1.000000    0.000000    0.000000    0.000000
 0.000000   -2.000000    3.000000   -1.000000    0.000000    0.000000
 0.000000    0.000000   -2.000000    3.000000   -1.000000    0.000000
 0.000000    0.000000    0.000000   -2.000000    3.000000   -1.000000
 0.000000    0.000000    0.000000    0.000000   -2.000000    3.000000
                    B matrix
 2.000000
 0.000000
 0.000000
 0.000000
 0.000000
 1.000000
                    SOLVING
                    SOLUTION VECTOR X
 1.000000
 0.999999
 0.999999
 0.999999
 0.999999
 0.999999
Temps d'execution : 0.000376 secondes
```

Listing 11 – $A_11X = B$ results

```
                    A matrix
 3.000000   -1.000000    0.000000    0.000000    0.000000    0.000000    0.000000
-2.000000    3.000000   -1.000000    0.000000    0.000000    0.000000    0.000000
 0.000000   -2.000000    3.000000   -1.000000    0.000000    0.000000    0.000000
 0.000000    0.000000   -2.000000    3.000000   -1.000000    0.000000    0.000000
 0.000000    0.000000    0.000000   -2.000000    3.000000   -1.000000    0.000000
 0.000000    0.000000    0.000000    0.000000   -2.000000    3.000000   -1.000000
 0.000000    0.000000    0.000000    0.000000    0.000000   -2.000000    3.000000
                    B matrix
 2.000000
 0.000000
 0.000000
 0.000000
 0.000000
 0.000000
 0.000000
 1.000000
                    SOLVING
                    SOLUTION VECTOR X
 1.000000
 1.000000
 1.000000
 0.999999
 0.999999
 0.999999
```

```
0.999999
0.999999
Temps d'execution : 0.000587 secondes
```

Listing 12 – $A_1 2X = B$ results

```
                A matrix
3.000000  -1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
-2.000000  3.000000  -1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000  -2.000000   3.000000  -1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000   0.000000  -2.000000   3.000000  -1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000   0.000000   0.000000  -2.000000   3.000000  -1.000000   0.000000   0.000000   0.000000   0.000000
0.000000   0.000000   0.000000   0.000000  -2.000000   3.000000  -1.000000   0.000000   0.000000   0.000000
0.000000   0.000000   0.000000   0.000000   0.000000  -2.000000   3.000000  -1.000000   0.000000   0.000000
0.000000   0.000000   0.000000   0.000000   0.000000   0.000000  -2.000000   3.000000  -1.000000   0.000000
0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000  -2.000000   3.000000  -1.000000
0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000  -2.000000   3.000000
                B matrix
2.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
1.000000
                SOLVING
                SOLUTION VECTOR X
1.000000
1.000000
1.000000
1.000000
0.999999
0.999999
0.999999
0.999999
0.999999
1.000000
Temps d'execution : 0.000439 secondes
```