Analyse Numérique I

Sup'Galilée, Ingénieurs MACS, 1ère année

François Cuvelier

Laboratoire d'Analyse Géométrie et Applications
Institut Galilée
Université Paris XIII.

2022/10/23

Chapitre IV

Résolution de systèmes linéaires

Plan

Méthodes itératives pour la résolution du système linéaire

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b}.$$

Trouver une **matrice d'itération** $\mathbb{B}$ et d'un vecteur $\boldsymbol{c}$ telles que

$$\boldsymbol{x}^{[k+1]} = \mathbb{B}\boldsymbol{x}^{[k]} + \boldsymbol{c},\ k \geqslant 0,\ \boldsymbol{x}^{[0]} \text{ arbitraire}$$

vérifie

$$\lim_{k\to\infty} \boldsymbol{x}^{[k]} = \tilde{\boldsymbol{x}} \text{ avec } \tilde{\boldsymbol{x}} = \mathbb{A}^{-1}\boldsymbol{b}$$

Plan

Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ une matrice régulière, avec $\forall i \in [\![1, n]\!],\ a_{i,i} \neq 0$

$$\mathbb{A} = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix} + \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{2,1} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$

$$= \mathbb{D} - \mathbb{E} - \mathbb{F} = \begin{pmatrix} \ddots & & -\mathbb{F} \\ & \mathbb{D} & \\ -\mathbb{E} & & \ddots \end{pmatrix}$$

Plan

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b} \iff \forall i,\ b_i = \sum_{j=1}^{n} \mathrm{A}_{i,j} x_j = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j + \mathrm{A}_{i,i} x_i + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j$$

La méthode itérative de **Jacobi** :

$$b_i = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j^{[k]} + \mathrm{A}_{i,i} x_i^{[k+1]} + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j^{[k]} \quad \forall i \in [\![1, n]\!]$$

ou encore

$$x_i^{[k+1]} = \frac{1}{\mathrm{A}_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} \mathrm{A}_{ij} x_j^{[k]} \right) \quad \forall i \in [\![1, n]\!]$$

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b} \iff \forall i,\ b_i = \sum_{j=1}^{n} \mathrm{A}_{i,j} x_j = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j + \mathrm{A}_{i,i} x_i + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j$$

La méthode itérative de **Jacobi** :

$$b_i = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j^{[k]} + \mathrm{A}_{i,i} x_i^{[k+1]} + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j^{[k]} \quad \forall i \in [\![1, n]\!]$$

$$\boldsymbol{b} = \mathbb{D}\boldsymbol{x}^{[k+1]} - \mathbb{E}\boldsymbol{x}^{[k]} - \mathbb{F}\boldsymbol{x}^{[k]}$$

ou encore

$$x_i^{[k+1]} = \frac{1}{\mathrm{A}_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} \mathrm{A}_{ij} x_j^{[k]} \right) \quad \forall i \in [\![1, n]\!]$$

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

Plan

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b} \iff \forall i,\ b_i = \sum_{j=1}^{n} \mathrm{A}_{i,j} x_j = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j + \mathrm{A}_{i,i} x_i + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j$$

La méthode itérative de **Gauss-Seidel** :

$$b_i = \mathrm{A}_{i,i} x_i^{[k+1]} + \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j^{[k+1]} + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j^{[k]} \quad \forall i \in [\![1, n]\!]$$

ou encore

$$x_i^{(k+1)} = \frac{1}{\mathrm{A}_{ii}} \left(b_i - \sum_{j=1}^{i-1} \mathrm{A}_{ij} x_j^{[k+1]} - \sum_{j=i+1}^{n} \mathrm{A}_{ij} x_j^{[k]} \right) \quad \forall i \in [\![1, n]\!]$$

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b} \iff \forall i,\ b_i = \sum_{j=1}^{n} \mathrm{A}_{i,j} x_j = \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j + \mathrm{A}_{i,i} x_i + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j$$

La méthode itérative de **Gauss-Seidel** :

$$b_i = \mathrm{A}_{i,i} x_i^{[k+1]} + \sum_{j=1}^{i-1} \mathrm{A}_{i,j} x_j^{[k+1]} + \sum_{j=i+1}^{n} \mathrm{A}_{i,j} x_j^{[k]} \quad \forall i \in [\![1, n]\!]$$

$$\boldsymbol{b} = \mathbb{D}\boldsymbol{x}^{[k+1]} - \mathbb{E}\boldsymbol{x}^{[k+1]} - \mathbb{F}\boldsymbol{x}^{[k]}$$

ou encore

$$x_i^{(k+1)} = \frac{1}{\mathrm{A}_{ii}} \left(b_i - \sum_{j=1}^{i-1} \mathrm{A}_{ij} x_j^{[k+1]} - \sum_{j=i+1}^{n} \mathrm{A}_{ij} x_j^{[k]} \right) \quad \forall i \in [\![1, n]\!]$$

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

Plan

Soit $w \in \mathbb{R}^*$.

$$x_i^{[k+1]} = w\hat{x}_i^{[k+1]} + (1-w)x_i^{[k]}$$

où $\hat{x}_i^{[k+1]}$ est obtenu à partir de l'une des deux méthodes précédentes.
Avec la méthode de Gauss-Seidel : méthode S.O.R. (successive over relaxation)

$$x_i^{[k+1]} = \frac{w}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1}^{i-1}\mathrm{A}_{ij}x_j^{[k+1]} - \sum_{j=i+1}^{n}\mathrm{A}_{ij}x_j^{[k]}\right) + (1-w)x_i^{[k]} \quad \forall i \in [\![1,n]\!]$$

Exercice 1:

Déterminer la matrice d'itération $\mathbb{B}$ et le vecteur $\boldsymbol{c}$ tels que

$$\boldsymbol{x}^{[k+1]} = \mathbb{B}\boldsymbol{x}^{[k]} + \boldsymbol{c}$$

en fonction de $\mathbb{D}$, $\mathbb{E}$, $\mathbb{F}$, et $\boldsymbol{b}$.

Proposition:

On pose $\mathbb{L} = \mathbb{D}^{-1}\mathbb{E}$ et $\mathbb{U} = \mathbb{D}^{-1}\mathbb{F}$.
La matrice d'itération de la méthode de Jacobi, notée $\mathbb{J}$, est donnée par

$$\mathbb{J} = \mathbb{D}^{-1}(\mathbb{E}+\mathbb{F}) = \mathbb{L}+\mathbb{U}, \tag{1}$$

La matrice d'itération de la méthode S.O.R., notée $\mathcal{L}_w$, est donnée par

$$\mathcal{L}_w = \left(\frac{\mathbb{D}}{w} - \mathbb{E}\right)^{-1}\left(\frac{1-w}{w}\mathbb{D} + \mathbb{F}\right) = (\mathbb{I} - w\mathbb{L})^{-1}\left((1-w)\mathbb{I} + w\mathbb{U}\right). \tag{2}$$

et elle vérifie

$$\rho(\mathcal{L}_w) \geqslant |w-1|. \tag{3}$$

La matrice d'itération de Gauss-Seidel est $\mathcal{L}_1$ et elle correspond à

$$\mathcal{L}_1 = (\mathbb{D}-\mathbb{E})^{-1}\mathbb{F} = (\mathbb{I}-\mathbb{L})^{-1}\mathbb{U}. \tag{4}$$

Plan

Théorème:

Soit $\mathbb{A}$ une matrice régulière décomposée sous la forme $\mathbb{A} = \mathbb{M} - \mathbb{N}$ avec $\mathbb{M}$ régulière. On pose

$$\mathbb{B} = \mathbb{M}^{-1}\mathbb{N} \quad \text{et} \quad \boldsymbol{c} = \mathbb{M}^{-1}\boldsymbol{b}.$$

Alors la suite définie par

$$\boldsymbol{x}^{[0]} \in \mathbb{K}^n \quad \text{et} \quad \boldsymbol{x}^{[k+1]} = \mathbb{B}\boldsymbol{x}^{[k]} + \boldsymbol{c}$$

converge vers $\bar{\boldsymbol{x}} = \mathbb{A}^{-1}\boldsymbol{b}$ quelque soit $\boldsymbol{x}^{[0]}$ si et seulement si $\rho(\mathbb{B}) < 1$.

Lien avec les méthodes de Jacobi, Gauss-Seidel, S.O.R.?

Corollaire:

Soit $\mathbb{A}$ une matrice vérifiant $\mathrm{A}_{i,i} \neq 0 \ \forall i$. Une condition nécessaire de convergence pour la méthode S.O.R. est que $0 < w < 2$.

Théorème: voir Lascaux-Théodor, vol.2, Théorème 19 et 20, pages 346 à 349

Soit $\mathbb{A}$ une matrice à diagonale strictement dominante ou une matrice inversible à diagonale fortement dominante alors

- la méthode de Jacobi est convergente,
- si $w \in]0,1]$ la méthode de Relaxation est convergente.

Théorème

Soit $\mathbb{A}$ une matrice hermitienne inversible en décomposée en $\mathbb{A} = \mathbb{M} - \mathbb{N}$ où $\mathbb{M}$ est inversible. Soit $\mathbb{B} = \mathbb{I} - \mathbb{M}^{-1}\mathbb{A}$, la matrice de l'itération. Supposons que $\mathbb{M}^* + \mathbb{N}$ (qui est hermitienne) soit définie positive. Alors $\rho(\mathbb{B}) < 1$ si et seulement si $\mathbb{A}$ est définie positive.

Preuve : voir exercice

Théorème

Soit $\mathbb{A}$ une matrice hermitienne définie positive, alors la méthode de relaxation converge si et seulement si $w \in]0, 2[$.

Preuve : voir Lascaux-Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, vol.2, Corollaire 24, page 351.

Plan

Principe de base

Résoudre :

$$\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$$

Méthodes itératives :

$$\boldsymbol{x}^{[0]} \in \mathbb{K}^n \text{ et } \boldsymbol{x}^{[k+1]} = \mathbb{B}\boldsymbol{x}^{[k]} + \boldsymbol{c}$$

Algorithme :

$\boldsymbol{x}^{[0]}$ donné
Pour $k = 0, 1, \cdots$ **faire**
  $\boldsymbol{x}^{[k+1]} \leftarrow \mathbb{B}\boldsymbol{x}^{[k]} + \boldsymbol{c}$
Fin Pour

Critère d'arrêt? Stockage de tous les $\boldsymbol{x}^{[k]}$?

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

$\Longrightarrow$ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\boldsymbol{x}^{[k]}$ suffisament proche de $\bar{\boldsymbol{x}} = \mathbb{A}^{-1}\boldsymbol{b}$

Comment choisir le critère d'arrêt pour la convergence?

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

$\Longrightarrow$ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\boldsymbol{x}^{[k]}$ suffisament proche de $\bar{\boldsymbol{x}} = \mathbb{A}^{-1}\boldsymbol{b}$

Comment choisir le critère d'arrêt pour la convergence?

Exemple de critère d'arrêt pour la convergence :
Soit $\boldsymbol{r}^{[k]} = \boldsymbol{b} - \mathbb{A}\boldsymbol{x}^{[k]}$ le résidu.

$$\frac{\|\boldsymbol{r}^{[k]}\|}{\|\boldsymbol{b}\|} \leqslant \varepsilon$$

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

$\Longrightarrow$ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\boldsymbol{x}^{[k]}$ suffisament proche de $\bar{\boldsymbol{x}} = \mathbb{A}^{-1}\boldsymbol{b}$

Comment choisir le critère d'arrêt pour la convergence?

Exemple de critère d'arrêt pour la convergence :
Soit $\boldsymbol{r}^{[k]} = \boldsymbol{b} - \mathbb{A}\boldsymbol{x}^{[k]}$ le résidu.

$$\frac{\|\boldsymbol{r}^{[k]}\|}{\|\boldsymbol{b}\|} \leqslant \varepsilon$$

Car dans ce cas, on a avec $\boldsymbol{e}^{[k]} = \bar{\boldsymbol{x}} - \boldsymbol{x}^{[k]}$

$$\frac{\|\boldsymbol{e}^{[k]}\|}{\|\bar{\boldsymbol{x}}\|} \leqslant \varepsilon \operatorname{cond}(\mathbb{A})$$

Algorithm Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

Données :

$\mathbb{A}$	:	matrice de $\mathcal{M}_n(\mathrm{K})$,
$\boldsymbol{b}$	:	vecteur de K^n,
$\boldsymbol{x}^0$	:	vecteur initial de K^n,
ε	:	la tolérence, $\varepsilon \in \mathbb{R}^+$,
kmax	:	nombre maximum d'itérations, $\mathrm{kmax} \in \mathbb{N}^*$

Résultat :

$\boldsymbol{x}^{\mathrm{tol}}$	:	un vecteur de K^n si convergence, sinon $\varnothing$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$ ▷ $\boldsymbol{p}$ contient le vecteur précédent
7: $\boldsymbol{x} \leftarrow$ calcul de l'itérée suivante en fonction de $\boldsymbol{p}$, $\mathbb{A}$, $\boldsymbol{b}$, ...
8: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
9: **Fin Tantque**
10: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors** ▷ Convergence
11: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
12: **Fin Si**

Pour Jacobi , la suite des itérées est définie par

$$x_i^{[k+1]} = \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} \mathrm{A}_{ij} x_j^{[k]}\right), \quad \forall i \in [\![1, n]\!].$$

Algorithme 2 $\mathcal{R}_0$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7:
8: $\boldsymbol{x} \leftarrow$ calcul par Jacobi
9: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
10: **Fin Tantque**
11: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors** ▷ Convergence
12: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
13: **Fin Si**

Algorithme 2 $\mathcal{R}_1$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} \mathrm{A}_{ij}\boldsymbol{p}_j\right)$
9: **Fin Pour**
10: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
11: **Fin Tantque**
12: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors** ▷ Convergence
13: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
14: **Fin Si**

Algorithme 2 $\mathcal{R}_1$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8:
9: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} \mathrm{A}_{ij}\boldsymbol{p}_j\right)$
10: **Fin Pour**
11: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
12: **Fin Tantque**
13: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**
14: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
15: **Fin Si**

Algorithme 2 $\mathcal{R}_2$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8: $\mathrm{S} \leftarrow 0$
9: **Pour** $j \leftarrow 1$ **à** n $(j \neq i)$ **faire**
10: $\mathrm{S} \leftarrow \mathrm{S} + \mathrm{A}_{i,j}\boldsymbol{p}_j$
11: **Fin Pour**
12: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}(b_i - \mathrm{S})$
13: **Fin Pour**
14: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
15: **Fin Tantque**
16: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**
17: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
18: **Fin Si**

Algorithm Méthode itérative de Jacobi pour la résolution d'un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

Données :

$\mathbb{A}$	:	matrice de $\mathcal{M}_n(\mathrm{K})$,
$\boldsymbol{b}$	:	vecteur de K^n,
$\boldsymbol{x}^0$	:	vecteur initial de K^n,
ε	:	la tolérence, $\varepsilon \in \mathbb{R}^+$,
kmax	:	nombre maximum d'itérations, $\mathrm{kmax} \in \mathbb{N}^*$

Résultat :

$\boldsymbol{X}$	:	un vecteur de K^n

1: **Fonction** $\boldsymbol{X} \leftarrow$ RSLJacobi ($\mathbb{A}, \boldsymbol{b}, \boldsymbol{x}^0, \varepsilon, \mathrm{kmax}$)
2: $k \leftarrow 0$, $\boldsymbol{X} \leftarrow \varnothing$
3: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A} * \boldsymbol{x}$,
4: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
5: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
6: $k \leftarrow k + 1$
7: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
8: **Pour** $i \leftarrow 1$ **à** n **faire**
9: $\mathrm{S} \leftarrow 0$
10: **Pour** $j \leftarrow 1$ **à** n $(j \neq i)$ **faire**
11: $\mathrm{S} \leftarrow \mathrm{S} + A(i,j) * p(j)$
12: **Fin Pour**
13: $x(i) \leftarrow (b(i) - \mathrm{S})/A(i,i)$
14: **Fin Pour**
15: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A} * \boldsymbol{x}$,
16: **Fin Tantque**
17: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**
18: $\boldsymbol{X} \leftarrow \boldsymbol{x}$
19: **Fin Si**
20: **Fin Fonction**

Pour Gauss-Seidel , la suite des itérées est définie par

$$x_i^{(k+1)} = \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1}^{i-1} \mathrm{A}_{ij} x_j^{[k+1]} - \sum_{j=i+1}^{n} \mathrm{A}_{ij} x_j^{[k]}\right) \quad \forall i \in [\![1, n]\!]$$

Algorithme 3 $\mathcal{R}_0$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7:
8: $\boldsymbol{x} \leftarrow$ calcul par Gauss-Seidel
9: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
10: **Fin Tantque**
11: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**
12: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
13: **Fin Si**

Algorithme 3 $\mathcal{R}_1$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1}^{i-1} \mathrm{A}_{ij} x_j - \sum_{j=i+1}^{n} \mathrm{A}_{ij}\boldsymbol{p}_j\right)$
9: **Fin Pour**
10: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
11: **Fin Tantque**
12: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**

Algorithme 3 $\mathcal{R}_1$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8:
9: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}\left(b_i - \sum_{j=1}^{i-1} \mathrm{A}_{ij} x_j - \sum_{j=i+1}^{n} \mathrm{A}_{ij}\boldsymbol{p}_j\right)$
10: **Fin Pour**
11: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
12: **Fin Tantque**
13: **Si** $\|\boldsymbol{r}\| \leqslant \mathrm{tol}$ **alors**
14: $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \boldsymbol{x}$
15: **Fin Si**

Algorithme 3 $\mathcal{R}_2$

1: $k \leftarrow 0$, $\boldsymbol{x}^{\mathrm{tol}} \leftarrow \varnothing$
2: $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$, $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
3: $\mathrm{tol} \leftarrow \varepsilon(\|\boldsymbol{b}\| + 1)$
4: **Tantque** $\|\boldsymbol{r}\| > \mathrm{tol}$ et $k \leqslant \mathrm{kmax}$ **faire**
5: $k \leftarrow k + 1$
6: $\boldsymbol{p} \leftarrow \boldsymbol{x}$
7: **Pour** $i \leftarrow 1$ **à** n **faire**
8: $\mathrm{S} \leftarrow 0$
9: **Pour** $j \leftarrow 1$ **à** $i - 1$ **faire**
10: $\mathrm{S} \leftarrow \mathrm{S} + \mathrm{A}_{i,j} x_j$
11: **Fin Pour**
12: **Pour** $j \leftarrow i + 1$ **à** n **faire**
13: $\mathrm{S} \leftarrow \mathrm{S} + \mathrm{A}_{i,j}\boldsymbol{p}_j$
14: **Fin Pour**
15: $x_i \leftarrow \frac{1}{\mathrm{A}_{ii}}(b_i - \mathrm{S})$
16: **Fin Pour**
17: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \mathbb{A}\boldsymbol{x}$,
18: **Fin Tantque**

Algorithm Méthode itérative de Gauss-Seidel pour la résolution d'un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

Données :
$\mathbb{A}$: matrice de $\mathcal{M}_n(\mathbb{K})$,
$\boldsymbol{b}$: vecteur de $\mathbb{K}^n$,
$\boldsymbol{x}^0$: vecteur initial de $\mathbb{K}^n$,
ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
kmax : nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :
$\boldsymbol{X}$: un vecteur de $\mathbb{K}^n$

```
1:  Fonction X ← RSLGaussSeidel ( 𝔸, b, x⁰, ε, kmax )
2:    k ← 0, X ← ∅
3:    x ← x⁰, r ← b − 𝔸 ∗ x,
4:    tol ← ε(‖b‖ + 1)
5:    Tantque ‖r‖ > tol et k ⩽ kmax faire
6:      k ← k + 1
7:      p ← x
8:      Pour i ← 1 à n faire
9:        S ← 0
10:       Pour j ← 1 à i − 1 faire
11:         S ← S + A(i, j) ∗ x(j)
12:       Fin Pour
13:       Pour j ← i + 1 à n faire
14:         S ← S + A(i, j) ∗ p(j)
15:       Fin Pour
16:       x(i) ← (b(i) − S)/A(i, i)
17:     Fin Pour
18:     r ← b − 𝔸 ∗ x,
19:   Fin Tantque
20:   Si ‖r‖ ⩽ tol alors
21:     X ← x
22:   Fin Si
23: Fin Fonction
```



```
Fonction X ← RSLJacobi ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n faire
      S ← 0
      Pour j ← 1 à n (j ≠ i) faire
        S ← S + A(i, j) ∗ p(j)
      Fin Pour
      x(i) ← (b(i) − S)/A(i, i)
    Fin Pour
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

```
Fonction X ← RSLGaussSeidel ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n faire
      S ← 0
      Pour j ← 1 à i − 1 faire
        S ← S + A(i, j) ∗ x(j)
      Fin Pour
      Pour j ← i + 1 à n faire
        S ← S + A(i, j) ∗ p(j)
      Fin Pour
      x(i) ← (b(i) − S)/A(i, i)
    Fin Pour
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```



```
Fonction X ← RSLJacobi ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n faire
      S ← 0
      Pour j ← 1 à n (j ≠ i) faire
        S ← S + A(i, j) ∗ p(j)
      Fin Pour
      x(i) ← (b(i) − S)/A(i, i)
    Fin Pour
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

```
Fonction X ← RSLGaussSeidel ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n faire
      S ← 0
      Pour j ← 1 à i − 1 faire
        S ← S + A(i, j) ∗ x(j)
      Fin Pour
      Pour j ← i + 1 à n faire
        S ← S + A(i, j) ∗ p(j)
      Fin Pour
      x(i) ← (b(i) − S)/A(i, i)
    Fin Pour
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

Même ossature puisque toutes deux basées sur l'Algorithme générique

Peut-on simplifier, clarifier et racourcir les codes?


```
Fonction X ← RSLJacobi ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n (j ≠ i) faire
      S ← S + A(i, j) ∗ p(j)
    Fin Pour
    x(i) ← (b(i) − S)/A(i, i)
  Fin Pour
  r ← b − 𝔸 ∗ x,
Fin Tantque
Si ‖r‖ ⩽ tol alors
  X ← x
Fin Si
Fin Fonction
```

Algorithm Itération de Jacobi : calcul de $\boldsymbol{x}$ tel que

$$x_i = \frac{1}{\mathbb{A}_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} \mathbb{A}_{ij} y_j\right), \quad \forall i \in [\![1, n]\!].$$

Données :
A : matrice de $\mathcal{M}_n(\mathbb{K})$,
b : vecteur de $\mathbb{K}^n$,
y : vecteur de $\mathbb{K}^n$,

Résultat :
x : un vecteur de $\mathbb{K}^n$

```
1: Fonction x ← IterJacobi ( A, b, y )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à n (j ≠ i) faire
5:       S ← S + A(i, j) ∗ y(j)
6:     Fin Pour
7:     x(i) ← (b(i) − S)/A(i, i)
8:   Fin Pour
9: Fin Fonction
```



```
Fonction X ← RSLJacobi2 ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    x ← IterJacobi(𝔸, b, p)
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

Algorithm Itération de Jacobi : calcul de $\boldsymbol{x}$ tel que

$$x_i = \frac{1}{\mathbb{A}_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} \mathbb{A}_{ij} y_j\right), \quad \forall i \in [\![1, n]\!].$$

Données :
A : matrice de $\mathcal{M}_n(\mathbb{K})$,
b : vecteur de $\mathbb{K}^n$,
y : vecteur de $\mathbb{K}^n$,

Résultat :
x : un vecteur de $\mathbb{K}^n$

```
1: Fonction x ← IterJacobi ( A, b, y )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à n (j ≠ i) faire
5:       S ← S + A(i, j) ∗ y(j)
6:     Fin Pour
7:     x(i) ← (b(i) − S)/A(i, i)
8:   Fin Pour
9: Fin Fonction
```



```
Fonction X ← RSLGaussSeidel ( 𝔸, b, x⁰, ε, kmax )
  k ← 0, X ← ∅
  x ← x⁰, r ← b − 𝔸 ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    Pour i ← 1 à n faire
      S ← 0
      Pour j ← 1 à i − 1 faire
        S ← S + A(i, j) ∗ x(j)
      Fin Pour
      Pour j ← i + 1 à n faire
        S ← S + A(i, j) ∗ p(j)
      Fin Pour
      x(i) ← (b(i) − S)/A(i, i)
    Fin Pour
    r ← b − 𝔸 ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

Algorithm Itération de Gauss-Seidel : calcul de $\boldsymbol{x}$ tel que

$$x_i = \frac{1}{\mathbb{A}_{ii}}\left(b_i - \sum_{j=1}^{i-1} \mathbb{A}_{ij} x_j - \sum_{j=i+1}^{n} \mathbb{A}_{ij} y_j\right), \quad \forall i \in [\![1, n]\!].$$

Données :
A : matrice de $\mathcal{M}_n(\mathbb{K})$,
b : vecteur de $\mathbb{K}^n$,
y : vecteur de $\mathbb{K}^n$,

Résultat :
x : un vecteur de $\mathbb{K}^n$

```
1:  Fonction x ← IterGaussSeidel ( A, b, y )
2:    Pour i ← 1 à n faire
3:      S ← 0
4:      Pour j ← 1 à i − 1 faire
5:        S ← S + A(i, j) ∗ x(j)
6:      Fin Pour
7:      Pour j ← i + 1 à n faire
8:        S ← S + A(i, j) ∗ y(j)
9:      Fin Pour
10:     x(i) ← (b(i) − S)/A(i, i)
11:   Fin Pour
12: Fin Fonction
```

```
Fonction X ← RSLGaussSeidel2 ( A, b, x^0, ε, kmax )
  k ← 0, X ← ∅
  x ← x^0, r ← b − A ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    x ← IterGaussSeidel(A, b, p)
    r ← b − A ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

Algorithm Itération de Gauss-Seidel : calcul de $\boldsymbol{x}$ tel que

$$x_i = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{i,j}x_j - \sum_{j=i+1}^{n} A_{i,j}y_j\right), \quad \forall i \in [\![1,n]\!].$$

Données :

A	:	matrice de $\mathcal{M}_n(\mathbb{K})$,
b	:	vecteur de $\mathbb{K}^n$,
y	:	vecteur de $\mathbb{K}^n$,

Résultat :

x	:	un vecteur de $\mathbb{K}^n$

```
1: Fonction x ← IterGaussSeidel ( A, b, y )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à i − 1 faire
5:       S ← S + A(i, j) ∗ x(j)
6:     Fin Pour
7:     Pour j ← i + 1 à n faire
8:       S ← S + A(i, j) ∗ y(j)
9:     Fin Pour
10:    x(i) ← (b(i) − S)/A(i, i)
11:  Fin Pour
12: Fin Fonction
```



```
Fonction X ← RSLGaussSeidel2 ( A, b, x^0, ε, kmax )
  k ← 0, X ← ∅
  x ← x^0, r ← b − A ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    x ← IterGaussSeidel(A, b, p)
    r ← b − A ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

```
Fonction X ← RSLJacobi2 ( A, b, x^0, ε, kmax )
  k ← 0, X ← ∅
  x ← x^0, r ← b − A ∗ x,
  tol ← ε(‖b‖ + 1)
  Tantque ‖r‖ > tol et k ⩽ kmax faire
    k ← k + 1
    p ← x
    x ← IterJacobi(A, b, p)
    r ← b − A ∗ x,
  Fin Tantque
  Si ‖r‖ ⩽ tol alors
    X ← x
  Fin Si
Fin Fonction
```

Les deux codes sont fortement similaires!

Peut-on éviter les copier/coller et gagner encore en lisibilité?

Ecriture Algorithme générique sous forme d'une fonction et on ajoute aux paramètres d'entrées une fonction formelle IterFonc calculant une itérée :

$$\boldsymbol{x} \leftarrow \text{IterFonc}(\mathbb{A}, \boldsymbol{b}, \boldsymbol{y}).$$

Algorithm Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

Données :

$\mathbb{A}$	:	matrice de $\mathcal{M}_n(\mathbb{K})$,
$\boldsymbol{b}$	:	vecteur de $\mathbb{K}^n$,
IterFonc	:	fonction de paramètres une matrice d'ordre n, et deux vecteurs de $\mathbb{K}^n$. retourne un vecteur de $\mathbb{K}^n$.
$\boldsymbol{x}^0$	:	vecteur initial de $\mathbb{K}^n$,
ε	:	la tolérence, $\varepsilon \in \mathbb{R}^+$,
kmax	:	nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :

$\boldsymbol{x}^{\text{tol}}$	:	un vecteur de $\mathbb{K}^n$ si convergence, sinon $\emptyset$

```
1: Fonction X ← RSLMethIter (A, b, IterFonc, x^0, ε, kmax)
2:   k ← 0, x^tol ← ∅
3:   x ← x^0, r ← b − Ax,
4:   tol ← ε(‖b‖ + 1)
5:   Tantque ‖r‖ > tol et k ⩽ kmax faire
6:     k ← k + 1
7:     p ← x
8:     x ← IterFonc(A, b, p)
9:     r ← b − Ax,
10:  Fin Tantque
11:  Si ‖r‖ ⩽ tol alors
12:    x^tol ← x
13:  Fin Si
14: Fin Fonction
```



```
Fonction X ← RSLJacobi3 ( A, b, x^0, ε, kmax )
  X ← RSLMethIter(A, b, IterJacobi, x^0, ε, kmax)
Fin Fonction
```

```
Fonction X ← RSLGaussSeidel3 ( A, b, x^0, ε, kmax )
  X ← RSLMethIter(A, b, IterGaussSeidel, x^0, ε, kmax)
Fin Fonction
```

Algorithm Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

Données :

$\mathbb{A}$	:	matrice de $\mathcal{M}_n(\mathbb{K})$,
$\boldsymbol{b}$	:	vecteur de $\mathbb{K}^n$,
IterFonc	:	fonction de paramètres une matrice d'ordre n, et deux vecteurs de $\mathbb{K}^n$. retourne un vecteur de $\mathbb{K}^n$.
$\boldsymbol{x}^0$	:	vecteur initial de $\mathbb{K}^n$,
ε	:	la tolérence, $\varepsilon \in \mathbb{R}^+$,
kmax	:	nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :

$\boldsymbol{x}^{\text{tol}}$	:	un vecteur de $\mathbb{K}^n$ si convergence, sinon $\emptyset$

```
1: Fonction X ← RSLMethIter (A, b, IterFonc, x^0, ε, kmax)
2:   k ← 0, x^tol ← ∅
3:   x ← x^0, r ← b − Ax,
4:   tol ← ε(‖b‖ + 1)
5:   Tantque ‖r‖ > tol et k ⩽ kmax faire
6:     k ← k + 1
7:     p ← x
8:     x ← IterFonc(A, b, p)
9:     r ← b − Ax,
10:  Fin Tantque
11:  Si ‖r‖ ⩽ tol alors
12:    x^tol ← x
13:  Fin Si
14: Fin Fonction
```


Méthode de relaxation utilisant Gauss-Seidel,avec $w \in \mathbb{R}^*$,

$$x_i^{[k+1]} = \frac{w}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^{n} A_{ij}x_j^{[k]}\right) + (1-w)x_i^{[k]} \quad \forall i \in [\![1,n]\!]$$

Algorithm Itération S.O.R.

Données :

A	:	matrice de $\mathcal{M}_n(\mathbb{K})$,
b	:	vecteur de $\mathbb{K}^n$,
y	:	vecteur de $\mathbb{K}^n$,
w	:	réel non nul.

Résultat :

x	:	un vecteur de $\mathbb{K}^n$

```
1: Fonction x ← IterSOR ( A, b, y, w )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à i − 1 faire
5:       S ← S − A(i, j) ∗ x(j)
6:     Fin Pour
7:     Pour j ← i + 1 à n faire
8:       S ← S − A(i, j) ∗ y(j)
9:     Fin Pour
10:    x(i) ← w ∗ (b(i) − S)/A(i, i) + (1 − w) ∗ y(i)
11:  Fin Pour
12: Fin Fonction
```

Paramètre w "en trop" dans l'appel de la fonction IterSOR pour pouvoir utiliser la fonction générique RSLMethIter !

Méthode de relaxation utilisant Gauss-Seidel,avec $w \in \mathbb{R}^*$,

$$x_i^{[k+1]} = \frac{w}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^{n} A_{ij}x_j^{[k]}\right) + (1-w)x_i^{[k]} \quad \forall i \in [\![1,n]\!]$$

Algorithm Itération S.O.R.

Données :

A	:	matrice de $\mathcal{M}_n(\mathbb{K})$,
b	:	vecteur de $\mathbb{K}^n$,
y	:	vecteur de $\mathbb{K}^n$,
w	:	réel non nul.

Résultat :

x	:	un vecteur de $\mathbb{K}^n$

```
1: Fonction x ← IterSOR ( A, b, y, w )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à i − 1 faire
5:       S ← S − A(i, j) ∗ x(j)
6:     Fin Pour
7:     Pour j ← i + 1 à n faire
8:       S ← S − A(i, j) ∗ y(j)
9:     Fin Pour
10:    x(i) ← w ∗ (b(i) − S)/A(i, i) + (1 − w) ∗ y(i)
11:  Fin Pour
12: Fin Fonction
```

Paramètre w "en trop" dans l'appel de la fonction IterSOR pour pouvoir utiliser la fonction générique RSLMethIter !

```
Fonction X ← RSLSOR3 ( A, b, w, x^0, ε, kmax )
  IterFun ← ((M, r, s) ↦ IterSOR(M, r, s, w))
  X ← RSLMethIter(A, b, IterFun, x^0, ε, kmax)
Fin Fonction
```