

# Projet : méthode itérative des directions alternées (ADI)

Nathan Toussaint

15 janvier 2024

## 1 Décomposition $LU$ d'une matrice tri-diagonale $T$

On considère une matrice tridiagonale  $T$  de dimension  $N \times N$  :

$$T = \begin{pmatrix} d_0 & u_0 & 0 & \cdots & 0 \\ l_1 & d_1 & u_1 & \ddots & \vdots \\ 0 & l_2 & d_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & u_{n-2} \\ 0 & \cdots & 0 & l_{n-1} & d_{n-1} \end{pmatrix} \quad (1)$$

dont les termes diagonaux sont supposés non nuls.

On la stocke en mémoire sous la forme de 3 tableaux de même taille  $N$  dont les indices commencent à 0 pour faciliter l'implémentation en C. La diagonale de  $T$  est stockée dans le tableau  $d$ , la première sous-diagonale dans  $l$  et la première sur-diagonale dans  $u$ . On note que les composantes 0 du vecteur  $l$  et la composante  $n - 1$  du vecteur  $u$  ne sont pas pertinentes.

La décomposition LU consiste à écrire la matrice tridiagonale  $T$  sous la forme d'un produit  $LU$  où

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \beta_1 & 1 & 0 & \ddots & \vdots \\ 0 & \beta_2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \beta_{n-1} & 1 \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} \alpha_0 & u_0 & 0 & \cdots & 0 \\ 0 & \alpha_1 & u_1 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \alpha_{n-2} & u_{n-2} \\ 0 & \cdots & 0 & 0 & \alpha_{n-1} \end{pmatrix} \quad (2)$$

Comme la première sur-diagonale supérieure de  $U$  s'identifie à celle de la matrice  $T$ , l'algorithme de décomposition LU se simplifie grandement. Une première version s'écrit :

```
1: function LU( $T$ )
2:    $\alpha_0 \leftarrow d_0$ 
3:   for  $k = 1..(n - 1)$  do
4:      $\beta_k \leftarrow l_k / \alpha_{k-1}$ 
5:      $\alpha_k \leftarrow d_k - u_{k-1} \beta_k$ 
6:   end for
7: end function
```

Grâce à la structure tri-diagonale de  $T$ , il n'est pas nécessaire d'allouer une matrice supplémentaire pour stocker la décomposition  $LU$ . Elle peut remplacer progressivement la matrice  $T$  ligne par ligne. L'algorithme optimisé en place mémoire s'écrit finalement :

```
1: function LU( $T$ )
2:   for  $k = 1..(n - 1)$  do
3:      $l_k \leftarrow l_k / d_{k-1}$ 
4:      $d_k \leftarrow d_k - u_{k-1} l_k$ 
5:   end for
```

▷ à la place de la matrice tridiagonale  $T$

6: **end function**

On note que la complexité de la décomposition  $LU$  pour une matrice tri-diagonale de rang  $N$  est  $\vartheta(N)$ . Elle est donc peu coûteuse par rapport à celle en  $\vartheta(N^3)$  pour une matrice pleine de rang  $N$ .

## 2 Résolution de $Tx = b$

On suppose que la matrice  $T$  a été décomposée en un produit  $LU$  où  $L$  est une matrice bi-diagonale inférieure avec des 1 sur la diagonale et  $U$  est une matrice bi-diagonale supérieure dont les termes diagonaux sont non nuls.

Résoudre  $LUx = b$  s'effectue en deux étapes : i) résolution de  $Ly = b$  par la méthode de descente puis ii) résolution de  $Ux = y$  par la méthode de montée.

### 2.1 Résolution de $Ly = b$ par la méthode de descente

Soit le système d'équations linéaires où la matrice associée est bi-diagonale inférieure :

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_1 & 1 & 0 & \ddots & \vdots \\ 0 & l_2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & l_{n-1} & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \quad (3)$$

La méthode de résolution est celle de la descente. De nouveau, dans le cas de matrice bi-diagonale, l'algorithme se simplifie et s'écrit :

```
1: function DESCENTE( $LU, b$ )
2:    $y_0 \leftarrow b_0$ 
3:   for  $k = 1..(n-1)$  do
4:      $y_k \leftarrow b_k - l_k y_{k-1}$ 
5:   end for
6: end function
```

La complexité de cet algorithme est  $\vartheta(N)$ .

### 2.2 Résolution de $Ux = y$ par la méthode de montée

Soit le système d'équations linéaires où la matrice associée est bi-diagonale supérieure :

$$U = \begin{pmatrix} d_0 & u_0 & 0 & \cdots & 0 \\ 0 & d_1 & u_1 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & d_{n-2} & u_{n-2} \\ 0 & \cdots & 0 & 0 & d_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} \quad (4)$$

L'algorithme de la méthode de la montée s'écrit dans notre cas :

```
1: function MONTEE( $LU, y$ )
2:    $x_{n-1} \leftarrow y_{n-1}/d_{n-1}$ 
3:   for  $i = (n-2)..1$  do
4:      $x_k \leftarrow (y_k - u_k x_{k+1})/d_k$ 
5:   end for
6: end function
```

La complexité de cet algorithme est  $\vartheta(N)$ .

### 2.3 Complexité

La complexité de résolution d'un système tri-diagonale en utilisant la méthode de décomposition  $LU$  et les méthodes de descente et de montée est  $\vartheta(N)$  et peu coûteuse par rapport à la méthode de résolution de Gauss qui a une complexité  $\vartheta(N^3)$  pour une matrice dense de rang  $N$ .

## 3 Méthode de résolution ADI

On cherche à résoudre en 2D, l'équation de Poisson :  $-\Delta u(x, y) = f(x, y)$  sur un domaine carré  $\Omega = ]0, 1[^2$ . Elle est discrétisée en différences finies sur une grille est de taille  $n_x \times n_y$ . On impose des conditions de Dirichlet sur le contour  $\partial\Omega$  épousant le contour de la grille :  $u(x, y) = u^d(x, y)$  (*notation différente du sujet pour réduire les ambiguïtés de notations avec les indices*).

### 3.1 Discrétisation du domaine carré

On repère chaque noeud de la grille par ses coordonnées entières  $(i, j)$  comme dans l'exemple ci-dessous d'une grille  $5 \times 4$  :

	$(0, 3)$	$(1, 3)$	$(2, 3)$	$(3, 3)$	$(4, 3)$
	$(0, 2)$	$(1, 2)$	$(2, 2)$	$(3, 2)$	$(4, 2)$
	$(0, 1)$	$(1, 1)$	$(2, 1)$	$(3, 1)$	$(4, 1)$
	$(0, 0)$	$(1, 0)$	$(2, 0)$	$(3, 0)$	$(4, 0)$

TABLE 1 – coordonnées entières des noeuds de la grille  $5 \times 4$

Dans l'approximation des différences finies, on définit :

$$\hat{L}_x u|_{(i,j)} = -\partial_x^2 u|_{(i,j)} = \frac{-u(i-1, j) + 2u(i, j) - u(i+1, j)}{h_x^2} + \vartheta(h_x^2) \quad (5)$$

et

$$\hat{L}_y u|_{(i,j)} = -\partial_y^2 u|_{(i,j)} = \frac{-u(i, j-1) + 2u(i, j) - u(i, j+1)}{h_y^2} + \vartheta(h_y^2) \quad (6)$$

où  $h_x = \frac{1}{n_x-1}$  et  $h_y = \frac{1}{n_y-1}$  sont les pas de la grille.

Dans la suite, on introduit les notations  $m_x = h_x^{-1} = n_x - 1$  et  $m_y = h_y^{-1} = n_y - 1$ .

Notons que la fonction  $f(x, y)$  est évaluée aux noeuds intérieurs  $(i, j)$  de la grille.

### 3.2 Les pas fractionnaires de la méthode

La méthode de résolution ADI est une méthode itérative composée de deux pas fractionnaires. Elle se caractérise par une approche qui alterne entre les directions  $x$  et  $y$ , en résolvant chaque sous-problème de manière implicite :

$$\begin{cases} (L_x + \omega_k I)u^* &= -(L_y - \omega_k I)u^k + b \\ (L_y + \omega_k I)u^{k+1} &= -(L_x - \omega_k I)u^* + b \end{cases} \quad (7)$$

Cette alternance permet de transformer la résolution de l'équation bidimensionnelle en deux problèmes unidimensionnels plus simples, qui peuvent être résolus efficacement. En pratique, la résolution s'arrête lorsque  $\|(L_x + L_y)u^k - b\|_\infty < \epsilon$  avec  $\epsilon \approx 10^{-6}$  par exemple, voire plus petit.

### 3.3 Application à la grille $5 \times 4$

On propose de détailler la méthode ADI dans le cas d'une grille de taille  $n_x \times n_y = 5 \times 4$ . Le premier pas fractionnaire de la méthode ADI consiste à résoudre, pour chacune des lignes  $j \in ]0, n_y - 1[ = \{1, 2\}$ . Pour  $i \in [1, 3]$ , il s'écrit :

$$-m_x^2 u_{i-1,j}^* + (2m_x^2 + \omega)u_{i,j}^* - m_x^2 u_{i+1,j}^* = m_y^2 u_{1,j-1}^k - (2m_y^2 - \omega)u_{1,j}^k + m_y^2 u_{1,j+1}^k + b_{i,j} \quad (8)$$

Viennent ensuite s'ajouter les conditions de dirichlet en  $i = 0$  et  $i = n_x - 1$ . De manière matricielle, le système d'équations à résoudre s'écrit :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -m_x^2 & 2m_x^2 + \omega & -m_x^2 & 0 & 0 \\ 0 & -m_x^2 & 2m_x^2 + \omega & -m_x^2 & 0 \\ 0 & 0 & -m_x^2 & 2m_x^2 + \omega & -m_x^2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{0,j}^* \\ u_{1,j}^* \\ u_{2,j}^* \\ u_{3,j}^* \\ u_{4,j}^* \end{pmatrix} = \begin{pmatrix} 0 \\ m_y^2 u_{1,j-1}^k - (2m_y^2 - \omega)u_{1,j}^k + m_y^2 u_{1,j+1}^k \\ m_y^2 u_{2,j-1}^k - (2m_y^2 - \omega)u_{2,j}^k + m_y^2 u_{2,j+1}^k \\ m_y^2 u_{3,j-1}^k - (2m_y^2 - \omega)u_{3,j}^k + m_y^2 u_{3,j+1}^k \\ 0 \end{pmatrix} + \begin{pmatrix} u_{0,j}^d \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \\ u_{4,j}^d \end{pmatrix} \quad (9)$$

où  $u_{i,j}^d$  est la valeur de dirichlet au noeud de coordonnées  $(i, j)$ . Cette première étape revient à résoudre  $n_y - 2$  petits systèmes linéaires dont la matrice est tri-diagonale de rang  $n_x$ .

De manière similaire, le second pas fractionnaire revient à résoudre, pour chacune des colonnes  $i \in ]0, n_x - 1[ = \{1, 2, 3\}$  :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_y^2 & 2m_y^2 + \omega & -m_y^2 & 0 \\ 0 & -m_y^2 & 2m_y^2 + \omega & -m_y^2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{i,0}^{k+1} \\ u_{i,1}^{k+1} \\ u_{i,2}^{k+1} \\ u_{i,3}^{k+1} \end{pmatrix} = \begin{pmatrix} 0 \\ m_x^2 u_{i-1,1}^* - (2m_x^2 - \omega)u_{i,1}^* + m_x^2 u_{i+1,1}^* \\ m_x^2 u_{i-1,2}^* - (2m_x^2 - \omega)u_{i,2}^* + m_x^2 u_{i+1,2}^* \\ 0 \end{pmatrix} + \begin{pmatrix} u_{i,0}^d \\ b_{i,1} \\ b_{i,2} \\ u_{i,4}^d \end{pmatrix} \quad (10)$$

De même, cette seconde étape revient à résoudre  $n_x - 2$  petits systèmes linéaires dont la matrice est tri-diagonale de rang  $n_y$ .

Dans l'implémentation, on ne traitera que le cas où les valeurs de  $u$  au bord sont strictement nulles.

### 3.4 Stockage des valeurs aux noeuds sous la forme d'un vecteur

Du point de vue de l'implémentation, on stockera les approximations de la solution dans un tableau unidimensionnel de taille  $n_x \times n_y$  en utilisant la relation  $n(i, j) = i + j n_x$  qui numérote de manière unique chaque noeud  $(i, j)$ .

### 3.5 Convergence

On note  $\bar{u}$  la solution de  $(L_x + L_y)\bar{u} = b$ . On définit les erreurs  $\epsilon^k = u^k - \bar{u}$  et  $\epsilon^{k+1/2} = u^{k+1/2} - \bar{u}$ . En soustrayant  $(L_x + L_y)\bar{u} = b$  des équations (7), on obtient :

$$\begin{cases} (L_x + \omega_k I)\epsilon^{k+1/2} &= -(L_y - \omega_k I)\epsilon^k \\ (L_y + \omega_k I)\epsilon^{k+1} &= -(L_x - \omega_k I)\epsilon^{k+1/2} \end{cases} \quad (11)$$

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

TABLE 2 – Numérotation unique des noeuds de la grille  $4 \times 4$

Après élimination de  $e^{k+1/2}$ , on obtient :

$$\epsilon^{k+1} = (L_y + \omega I)^{-1}(L_x - \omega I)(L_x + \omega I)^{-1}(L_y - \omega I) \epsilon^k = T(\omega) \epsilon^k \quad (12)$$

Comme les matrices  $L_x$  et  $L_y$  commutent, on a :

$$T(\omega) = (L_x + \omega I)^{-1}(L_x - \omega I)(L_y + \omega I)^{-1}(L_y - \omega I) \quad (13)$$

A titre d'exemple, prenons  $n_x = n_y = 4$ , les matrices  $L_x$  et  $L_y$  en tenant compte des conditions de dirichlet sur les bords sont de la forme et **commutent** entre elles :

	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
0	1															
1		1														
2			1													
3				1												
4					1											
5					$-m_x^2$	$2m_x^2$	$-m_x^2$									
6						$-m_x^2$	$2m_x^2$	$-m_x^2$								
7								1								
8									1							
9								$-m_x^2$	$2m_x^2$	$-m_x^2$						
10									$-m_x^2$	$2m_x^2$	$-m_x^2$					
11												1				
12													1			
13														1		
14															1	
15																1

$L_x =$

$$L_y =$$

	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
0	1															
1		1														
2			1													
3				1												
4					1											
5		$-m_y^2$				$2m_y^2$				$-m_y^2$						
6			$-m_y^2$				$2m_y^2$				$-m_y^2$					
7								1								
8									1							
9						$-m_y^2$				$2m_y^2$				$-m_y^2$		
10							$-m_y^2$				$2m_y^2$				$-m_y^2$	
11												1				
12													1			
13														1		
14															1	
15																1

Les valeurs propres de  $T(\omega)$  valent  $\frac{\lambda_n - \omega}{\lambda_n + \omega} \frac{\gamma_n - \omega}{\gamma_n + \omega}$  et son rayon spectral vaut  $\max_{\lambda_n, \gamma_n} \left| \frac{\lambda_n - \omega}{\lambda_n + \omega} \times \frac{\gamma_n - \omega}{\gamma_n + \omega} \right|$ .  
 Pour  $\omega > 0$ , la fonction  $\varphi(\lambda) = \frac{\lambda - \omega}{\lambda + \omega}$