# TP3 : Équation de diffusion instationnaire en 2D

Mouhalhal Moussaab

## 1. Introduction

Dans ce TP, on étudie l'évolution de la vitesse $u(x, y, t)$ d'un écoulement dans une conduite rectangulaire. Ce mouvement est provoqué par un gradient de pression constant suivant la direction $z$ qui vaut $-\mu G$, et donc l'évolution de la vitesse est décrite par l'équation :

$$\frac{\partial u}{\partial t} = \nu G + \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

où $\nu$ est la viscosité cinématique. On impose que la vitesse soit nulle ($u = 0$) sur les bords de la conduite, et initialement l'écoulement est immobile, c'est-à-dire $u(x, y, 0) = 0$.

La conduite a des dimensions $L_x = 2\,\text{cm}$ et $L_y = 1\,\text{cm}$. La constante $G$ on peut la déterminer à partir du débit $Q$ que l'on impose ($Q = 10^{-6}\ m^3/s$).

L'objectif est de résoudre cette équation pour déterminer la vitesse $u$ en tout point de la conduite une fois que l'écoulement est stabilisé, en comparant des solutions numériques obtenues par différents schémas à une solution exacte connue.

## 2. Solution exacte

La solution stationnaire exacte une fois l'écoulement établi est donnée par :

$$u_{ex}(x, y) = \frac{4GL_x^2}{\pi^3} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n^3} \left( 1 - \frac{\cosh\left(\frac{n\pi(y - L_y/2)}{L_x}\right)}{\cosh\left(\frac{n\pi L_y}{2L_x}\right)} \right) \sin\left(\frac{n\pi x}{L_x}\right)$$

On détermine le coefficient $G$ à partir de l'expression du débit cible $Q$ par la formule suivante :

$$Q_{ex} = \frac{4GL_x^3 L_y}{\pi^4} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n^4} \left( 1 - \frac{1}{\cosh\left(\frac{n\pi L_y}{2L_x}\right)} \right)$$

## 3. Méthodes numériques

### 3.1 Discrétisation

Tout d'abord on discrétise uniformément le domaine (suivant $x$ et $y$) ainsi que le temps :

— $x_i = i\,\Delta x$, avec $\Delta x = \frac{L_x}{N_x - 1}$, où $N_x = 16$ est le nombre total de points de discrétisation suivant $x$.
— $y_j = j\,\Delta y$, avec $N_y = 8$.
— $t_n = n\,\Delta t$, avec $\Delta t = \frac{t_f}{N_t}$, où $N_t$ est le nombre de pas de temps pour une durée totale $t_f$.

La solution $u$ est associée à chaque point du domaine discrétisé par les indices $(i, j)$, où $i$ et $j$ désignent les indices selon $x$ et $y$. Pour simplifier l'écriture et la programmation, on utilise une numérotation linéaire sous la forme $u_k^n$ avec $k = i + jN_x$.

## 3.2 Schémas numériques

on résout le problème numériquement à l'aide du $\theta$-schéma dont l'équation discrétisée en notation indicielle s'écrit :

$$\begin{aligned}\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} &= \nu G + \theta\nu \left( \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) \\ &\quad + (1-\theta)\nu \left( \frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} \right)\end{aligned}$$

En notation linéaire ($k = i + jN_x$), cela devient :

$$\begin{aligned}\frac{u_k^{n+1} - u_k^n}{\Delta t} &= \nu G + \theta\nu \left( \frac{u_{k+1}^n - 2u_k^n + u_{k-1}^n}{\Delta x^2} + \frac{u_{k+N_x}^n - 2u_k^n + u_{k-N_x}^n}{\Delta y^2} \right) \\ &\quad + (1-\theta)\nu \left( \frac{u_{k+1}^{n+1} - 2u_k^{n+1} + u_{k-1}^{n+1}}{\Delta x^2} + \frac{u_{k+N_x}^{n+1} - 2u_k^{n+1} + u_{k-N_x}^{n+1}}{\Delta y^2} \right)\end{aligned}$$

Cette équation permet d'interpoler entre les schémas explicite ($\theta = 1$) et implicite ($\theta = 0$), où chaque schéma conduit à un système linéaire de la forme :

$$AU^{n+1} = BU^n + C$$

**Schéma explicite :**

Tout d'abord pour simplifier l'écriture on pose :

$$\alpha_x = \frac{\nu \Delta t}{\Delta x^2}, \qquad \alpha_y = \frac{\nu \Delta t}{\Delta y^2}$$

Donc les matrices A, B et C seront exprimés de la manière suivante :

$$A = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}_{(N_x N_y \times N_x N_y)}$$

$$B = \begin{bmatrix} c & d & e & & & & \\ b & c & d & e & & & \\ a & b & c & d & e & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & a & b & c & d & e \\ & & & a & b & c & d \\ & & & & a & b & c \end{bmatrix}_{(N_x N_y \times N_x N_y)}$$

où :

$$a = \alpha_y, \quad b = \alpha_x, \quad c = 1 - 2\alpha_x - 2\alpha_y, \quad d = \alpha_x, \quad e = \alpha_y$$

$$C = \nu G \Delta t \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{(N_x N_y \times 1)}$$

Ce schéma est consistant d'ordre 1 en temps et d'ordre 2 en espace, et il est stable si $2\,\Delta t\,\nu\,\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \leq 1$, et donc ce dernier est conditionnellement convergent.

Pour implémenter numériquement le schéma explicite, on a créé une fonction qui construit la matrice $B$ avec **np.diag** à partir des coefficients $a$, $b$, $c$, $d$, $e$, puis qui initialise le vecteur $U$ à zéro. À chaque pas de temps, on met à jour la solution avec la relation $U^{n+1} = BU^n + C$ (A= matrice identité). Après chaque mise à jour, on applique les conditions aux limites en imposant des valeurs nulles sur les bords du domaine directement dans le vecteur $U$. Enfin, on transforme $U$ en une matrice 2D de dimensions $(N_y, N_x)$ pour pouvoir l'afficher.

**Schéma implicite :**

Pour le schéma implicite on exprime les matrices du système linéaire de la façon suivante :

$$A = \begin{bmatrix} c & d & e & & & & \\ b & c & d & e & & & \\ a & b & c & d & e & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & a & b & c & d & e \\ & & & a & b & c & d \\ & & & & a & b & c \end{bmatrix}_{(N_x N_y \times N_x N_y)}$$

$$B = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}_{(N_x N_y \times N_x N_y)}$$

$$C = \nu G \Delta t \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{(N_x N_y \times 1)}$$

où :

$$a = -\alpha_y, \quad b = -\alpha_x, \quad c = 1 + 2\alpha_x + 2\alpha_y, \quad d = -\alpha_x, \quad e = -\alpha_y$$

Ce schéma est consistant d'ordre 1 en temps et d'ordre 2 en espace, et il toujours stable et il est convergent.

Pour implémenter numériquement ce schéma, on a créé une fonction qui construit la matrice $A$ en utilisant des matrices creuses (**scipy.sparse.csr_matrix**) afin d'optimiser les calculs. À chaque pas de temps, on résout le système $AU^{n+1} = U^n + C$ (B= matrice identité) à l'aide de **scipy.sparse.linalg.spsolve**. Comme pour le schéma explicite, on impose les conditions aux limites en annulant directement les valeurs sur les bords du domaine. On a également créé une fonction équivalente utilisant des matrices pleines.

**Solution stabilisée** :On a choisi un petit pas de temps $\Delta t = 0{,}01$ s et un temps final suffisant $t_f = 60$ s (pas trop grand, car les calculs, surtout pour le cas des matrices pleines, prennent énormément de temps). Ce choix permet d'obtenir des solutions stables, avec un écart relatif par rapport à la solution stabilisée (obtenue avec un temps très grand, $t_f = 1000$ s) de **0,077 %** pour le schéma explicite et **0,079 %** pour le schéma implicite.

# 4. Résultats numériques

## 4.1 Vitesse exacte

On calcule la solution exacte à partir d'une série tronquée après 25 termes impairs. Elle sert de référence pour évaluer la précision des schémas numériques.

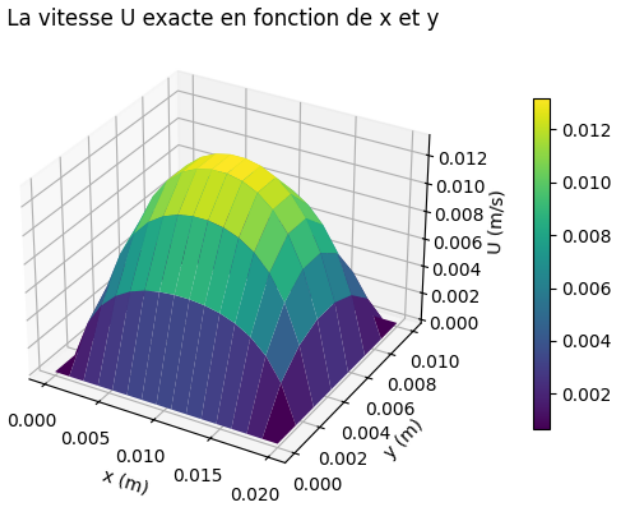



FIGURE 1 – Vitesse $U$ exacte en fonction de $x$ et $y$

On voit que la vitesse aux bords du domaine est nulle, et elle augmente jusqu'à atteindre une valeur maximale de 0,137 m/s au milieu du domaine.

## 4.2 Schéma explicite

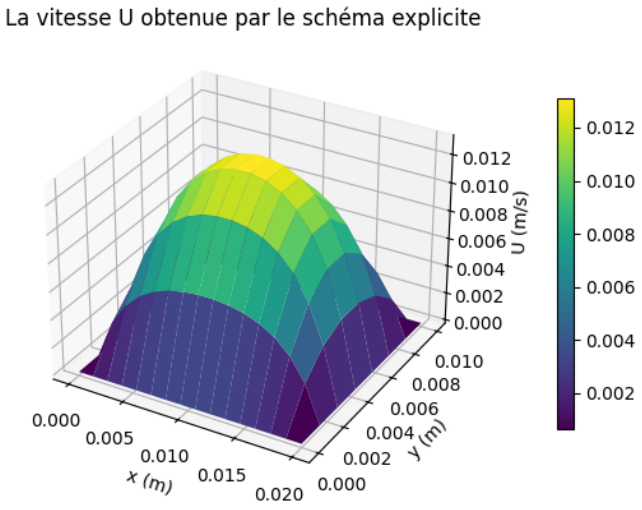



FIGURE 2 – Vitesse $U$ obtenue par le schéma explicite

On voit que la vitesse obtenue par le schéma explicite a exactement la même forme que la solution exacte, avec une vitesse nulle aux bords et également un maximum au centre du

domaine d'étude de 0,137 m/s.

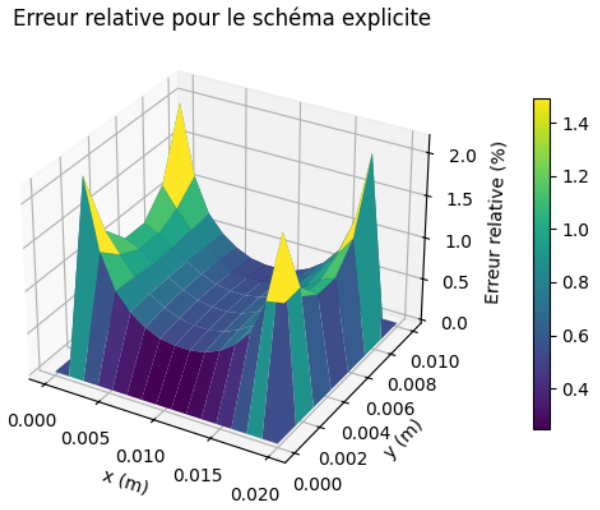



FIGURE 3 – Erreur relative du schéma explicite

L'erreur est nulle aux bords, ce qui est normal puisqu'on a imposé les mêmes conditions aux limites à la solution exacte et aux solutions numériques. Le maximum de l'erreur relative vaut 2.16% ce qui est très petit : cela montre que notre solution obtenue par le schéma explicite est très précise.

## 4.3 Schéma implicite (matrice creuse)

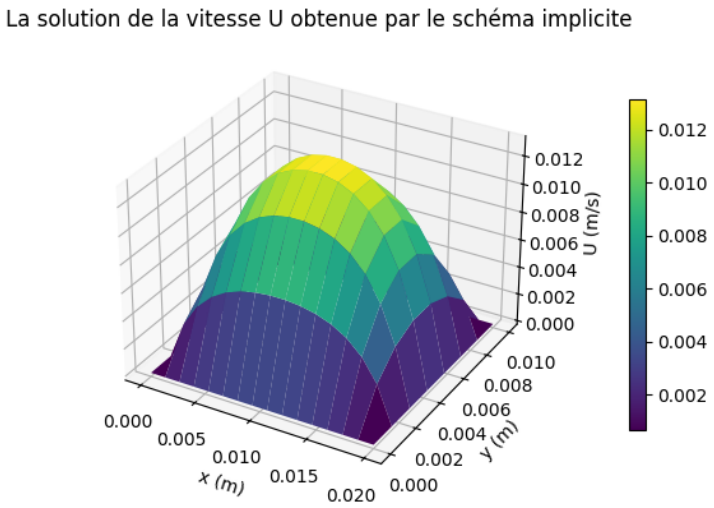



FIGURE 4 – Vitesse $U$ obtenue par le schéma implicite (matrice creuse)

La vitesse obtenue par le schéma implicite a aussi la même forme que la solution exacte. Les conditions aux limites sont bien appliquées et la valeur maximale au centre du domaine d'étude est bien conservée.

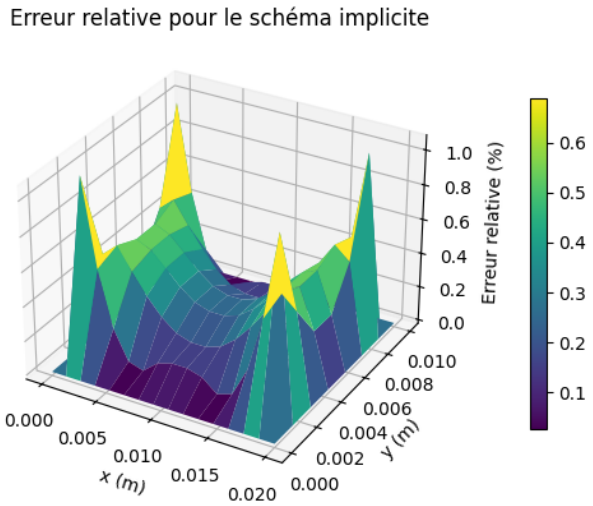
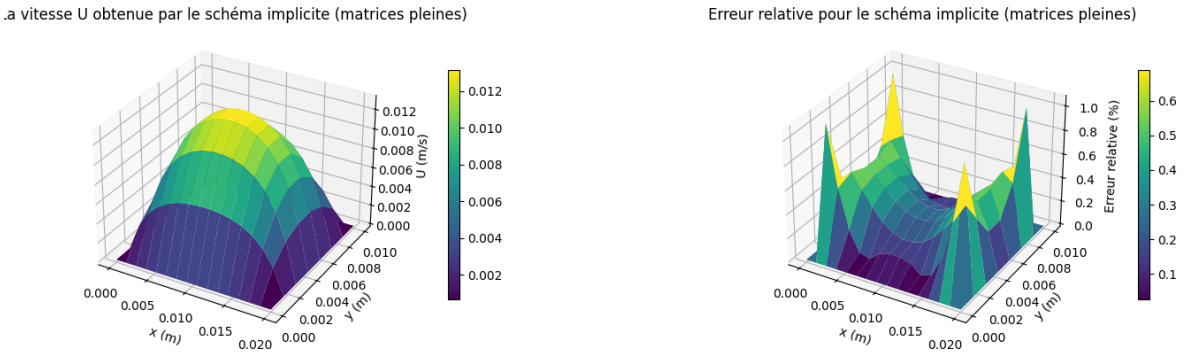



FIGURE 5 – Erreur relative du schéma implicite

L'erreur relative est cette fois légèrement plus faible avec un maximum de 1,06 %. Cela

montre que la solution obtenue avec le schéma implicite est un peu plus précise que celle du schéma explicite pour les mêmes conditions de temps.

### 4.4 Schéma implicite (matrice pleine)




(a) Vitesse obtenue (matrice pleine)

(b) Erreur relative (matrice pleine)

FIGURE 6 – Résultats du schéma implicite avec matrice pleine

On obtient les mêmes résultats que la simulation avec matrices creuses, mais avec un temps de calcul (mesuré avec **process_time()**) de 1609,33 s (soit environ 26 minutes) pour l'un des essais, ce qui est extrêmement long comparé à celui de la méthode avec matrices creuses (3,05 s), ce qui donne un rapport de $\approx 527$ entre les deux !

Ce temps très élevé confirme l'intérêt d'utiliser des structures creuses pour la résolution de systèmes linéaires de grande taille.

## 5. Conclusion

Ce TP nous a permis de mieux comprendre comment résoudre une équation de diffusion en deux dimensions avec différentes méthodes numériques. Nous avons utilisé un schéma explicite et un schéma implicite pour simuler l'évolution de la vitesse dans une conduite rectangulaire. En comparant les résultats obtenus avec la solution exacte, on peut vérifier la précision de chaque méthode. Le schéma explicite est plus simple mais nécessite un petit pas de temps pour rester stable, tandis que le schéma implicite est plus stable mais demande plus de calculs.Ces calculs peuvent toutefois être optimisés en exploitant la structure creuse des matrices utilisées. Ce travail nous aide à mieux comprendre les différences entre les méthodes numériques et leur application à des problèmes physiques.