Compte-rendu – TP Méthodes Numériques :
Simulation d'écoulement fluide

Valentin Gautier
Thomas Bauer

17 mars 2019

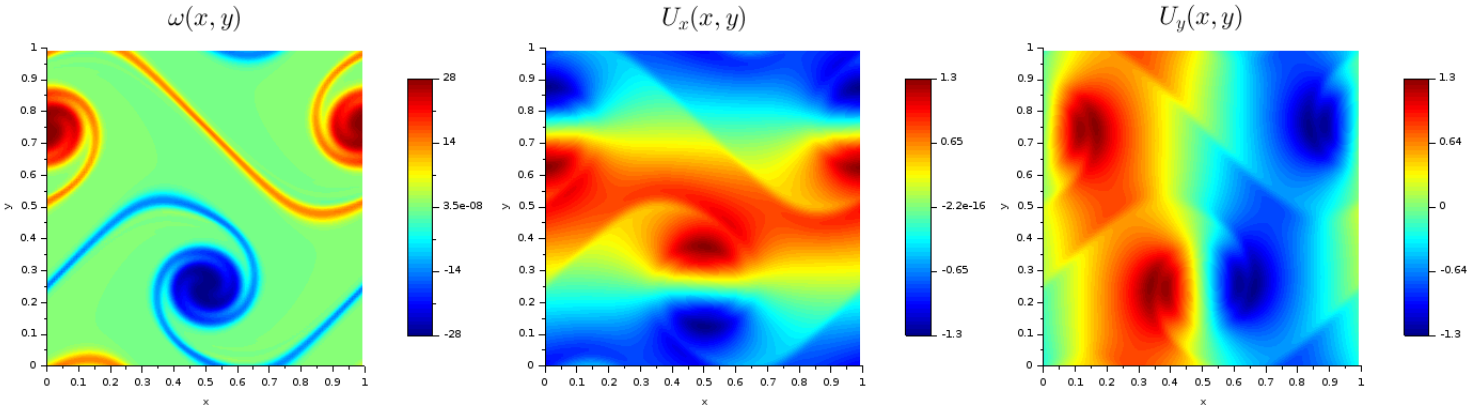


Table des matières

Résumé

Le but de ce TP de Méthodes Numériques est de nous faire manipuler les concepts et méthodes vus en cours à travers un problème de modélisation concret.

Nous nous intéresserons ici à l'étude d'un modèle d'écoulement fluide régi par les équations de Navier – Stokes. Afin d'effectuer la simulation numérique, l'utilisation du logiciel `scilab` ainsi que la subdivision en sous-problèmes nous permettront d'implémenter progressivement différentes routines de résolution.

Il s'agira en premier lieu d'implémenter une méthode de résolution par différences finies ainsi qu'un solveur linéaire, puis un solveur spectral utilisant la transformée de Fourier discrète. Ces routines seront enfin utilisées afin de résoudre en temps le problème de modélisation initial.

1 Résolution de l'équation de transport diffusion

Dans cette partie, nous nous proposons de résoudre une équation d'advection-diffusion unidimensionnelle avec une condition initiale donnée et des conditions aux bords périodiques.

1.1 Construction de la méthode de résolution

Étant donnée une équation aux dérivées partielles, sa discrétisation et l'approximation des opérateurs différentiels à l'aide de la formule de Taylor afin de ramener sa résolution à celle d'un système linéaire est un choix classique et pertinent. Aussi, la méthode numérique retenue ici est un schéma de différences finies.

Un tel schéma permettant d'approximer les dérivées spatiales et temporelles en un point de la discrétisation à partir du point précédent et suivant et le problème étant périodique, nous nous attendons à obtenir des matrices M et N circulantes dans l'écriture matricielle du système d'équations auquel nous nous ramenons.

Le schéma numérique proposé conduit en effet au système d'équations suivant :

$$M \begin{pmatrix} \phi(0, k\mathrm{dt}) \\ \vdots \\ \phi(n\mathrm{dx}, k\mathrm{dt}) \\ \vdots \\ \phi((N_x-1)\mathrm{dx}, k\mathrm{dt}) \end{pmatrix} = N \begin{pmatrix} \phi(0, (k+1)\mathrm{dt}) \\ \vdots \\ \phi(n\mathrm{dx}, (k+1)\mathrm{dt}) \\ \vdots \\ \phi((N_x-1)\mathrm{dx}, (k+1)\mathrm{dt}) \end{pmatrix},$$

où

$$M = \begin{pmatrix} 1-c(x)^2\frac{dt^2}{dx^2} & c(x)\frac{dt}{2dx}(c(x)\frac{dt}{dx}-1) & 0 & \cdots & 0 & c(x)\frac{dt}{2dx} \\ c(x)\frac{dt}{2dx} & 1-c(x)^2\frac{dt^2}{dx^2} & c(x)\frac{dt}{2dx}(c(x)\frac{dt}{dx}-1) & 0 & & 0 \\ 0 & c(x)\frac{dt}{2dx} & 1-c(x)^2\frac{dt^2}{dx^2} & c(x)\frac{dt}{2dx}(c(x)\frac{dt}{dx}-1) & \ddots & \vdots \\ \vdots & 0 & c(x)\frac{dt}{2dx} & 1-c(x)^2\frac{dt^2}{dx^2} & \ddots & 0 \\ 0 & & \ddots & \ddots & \ddots & c(x)\frac{dt}{2dx}(c(x)\frac{dt}{dx}-1) \\ c(x)\frac{dt}{2dx}(c(x)\frac{dt}{dx}-1) & 0 & \cdots & 0 & c(x)\frac{dt}{2dx} & 1-c(x)^2\frac{dt^2}{dx^2} \end{pmatrix},$$

$$N = \begin{pmatrix} 1+2\kappa\frac{dt}{dx^2} & -\kappa\frac{dt}{dx^2} & 0 & \cdots & 0 & -\kappa\frac{dt}{dx^2} \\ -\kappa\frac{dt}{dx^2} & 1+2\kappa\frac{dt}{dx^2} & -\kappa\frac{dt}{dx^2} & 0 & & 0 \\ 0 & -\kappa\frac{dt}{dx^2} & 1+2\kappa\frac{dt}{dx^2} & -\kappa\frac{dt}{dx^2} & \ddots & \vdots \\ \vdots & 0 & -\kappa\frac{dt}{dx^2} & 1+2\kappa\frac{dt}{dx^2} & \ddots & 0 \\ 0 & & \ddots & \ddots & \ddots & -\kappa\frac{dt}{dx^2} \\ -\kappa\frac{dt}{dx^2} & 0 & \cdots & 0 & -\kappa\frac{dt}{dx^2} & 1+2\kappa\frac{dt}{dx^2} \end{pmatrix}.$$

La résolution pratique du système linéaire nécessite de prendre en compte les propriétés des matrices à manipuler, afin d'être efficace. Remarquons en premier lieu que N est une matrice symétrique définie positive.

Démonstration.

On peut réécrire N sous la forme $N = I + \kappa\frac{dt}{dx^2}(2I - J - J^{n-1})$ avec J la matrice de permutation circulaire d'ordre 1.

Or J étant une matrice circulante, elle a pour valeurs propres $\{\omega^k \mid k \in [\![0, n-1]\!]\}$, donc $2I - J - J^{n-1}$ a pour valeurs propres $\{2 - \omega^k - \omega^{k(n-1)} \mid k \in [\![0, n-1]\!]\}$. De plus, ω est racine n-ième de l'unité donc ω^k et $\omega^{k(n-1)}$ sont conjugués. Donc, $\forall k \in [\![0, n-1]\!]$, $2 - \omega^k - \omega^{k(n-1)} = 0$.

Les valeurs propres de N sont donc $Sp(N) = Sp(I) = \{1\}$. Les valeurs propres de N sont donc toutes strictement positives et N est symétrique. Donc, N est symétrique définie positive.

La méthode de Cholesky est ici tout à fait adaptée car meilleure que d'autres schémas numériques (pivot de Gauss ...) sur cette classe de matrices. M et N étant creuses, les programmes n'en seront que plus efficaces et la résolution plus rapide.

La méthode de Cholesky repose sur le fait qu'une matrice A symétrique définie positive admet une décomposition de la forme $A = LL^T$ où L est une matrice triangulaire inférieure. La conséquence directe dont nous nous servons ici, est qu'il est possible de résoudre un système linéaire en résolvant deux systèmes triangulaires, ce qui est bien plus rapide. En effet, si $A = LL^T$ et que l'on cherche à résoudre le système $Ax = b$ où $x, b \in \mathbf{R}$, il suffit de résoudre les deux systèmes triangulaires suivants :

$$\left\{ \begin{array}{rcl} Ly & = & b \\ L^T x & = & y \end{array} \right.$$

Ainsi, notre routine de calcul est divisée en trois fonctions : la première (`cholesky_fact`) réalise en place la factorisation de N, la seconde (`up_sweep_cholesky`) et la troisième (`down_sweep_cholesky`) résolvent respectivement des systèmes triangulaires supérieurs et inférieurs (correspondant aux deux systèmes triangulaires de la méthode de Cholesky). Pour effectuer la résolution en pratique, il ne reste qu'à initialiser les fonctions et les matrices dans nos programmes.

1.2 Résultats et analyse

La première simulation, à l'aide de la fonction `my_cholesky`, a été effectuée pour différentes valeurs du coefficient de diffusion κ en utilisant la vitesse de convection $c(x) = 0.4(x - 0.25)$ sur $[0, 1]$ et la condition initiale définie par :

$$\left\{ \begin{array}{rcl} 0 \leq x < 0.25 & \Rightarrow & \phi_0(x) = 0 \\ 0.25 \leq x < 0.375 & \Rightarrow & \phi_0(x) = 2(x - 0.25) \\ 0.375 \leq x < 0.5 & \Rightarrow & \phi_0(x) = 2(0.5 - x) \\ 0.5 \leq x \leq 1 & \Rightarrow & \phi_0(x) = 0 \end{array} \right. .$$

Nous obtenons les résultats suivants :

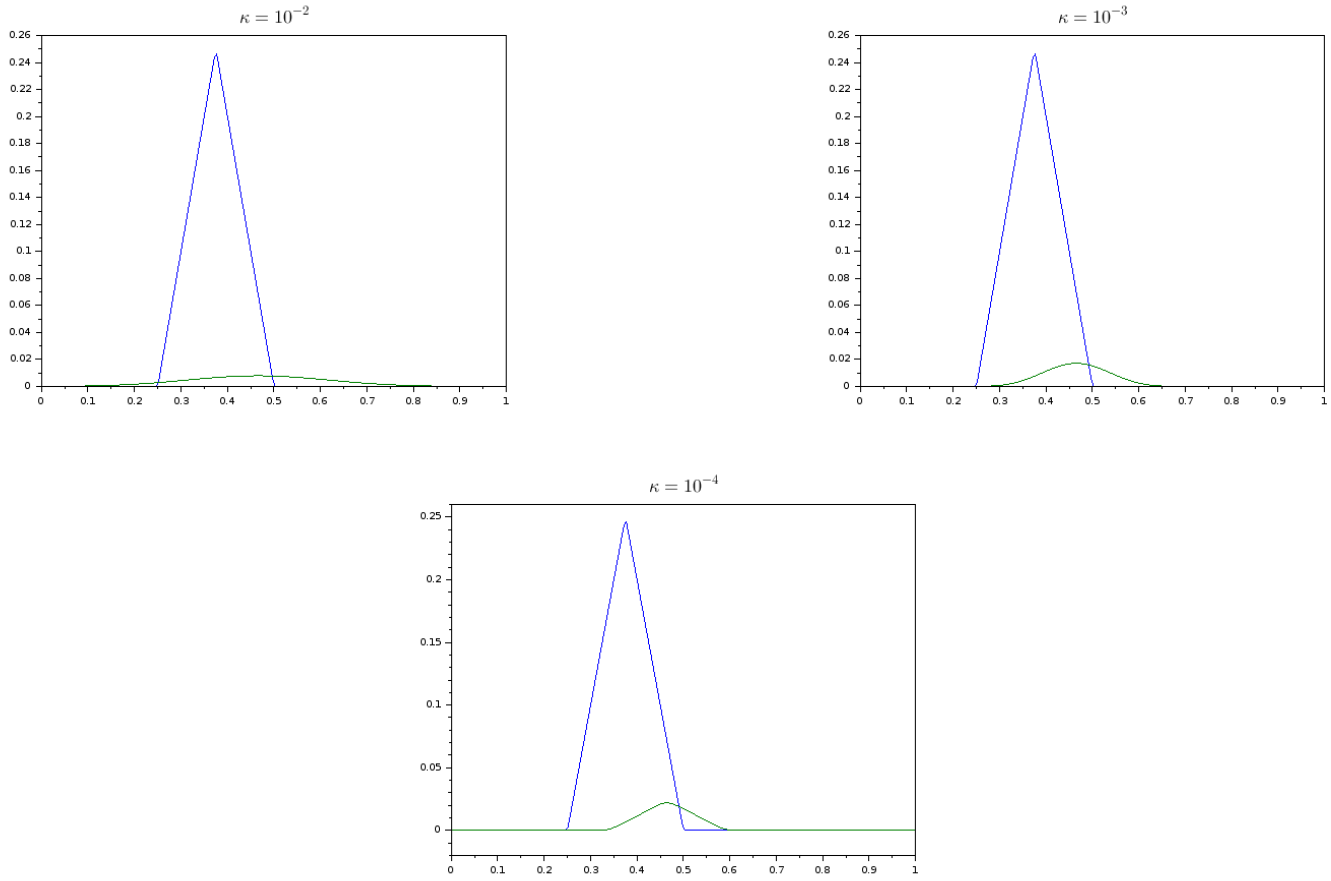


Fig. 1 – *Condition initiale (bleu) et solution au temps final (vert) de l'équation d'advection-diffusion 1D*

Nous pouvons constater que les trois graphiques présentent des solutions similaires, malgré la différence d'ordres de grandeur significative entre les différentes valeurs de κ (un rapport de 10^2 entre les deux valeurs extrêmes). Les solutions ont toutes une forme similaire à la condition initiale, sont atténuées et translatées selon les x croissants. En revanche, l'étendue sapatiale de la solution dépend de la valeur de κ : plus κ est proche de 0 et moins la solution est étendue. Du point de vue physique, nous pouvons émettre l'hypothèse que le coefficient de diffusion κ joue essentiellement sur l'étendue spatiale de la solution. En revanche, il ne semble pas influer l'atténuation. Par ailleurs, il serait intéressant de modéliser la situation plus finement et de manière plus réaliste, en utilisant notamment une représentation 3D.

Du point de vue informatique, les calculs ont été effectués très rapidement grâce à l'implémentation de la méthode de Cholesky. Cependant les fonctions pré-existantes en `scilab` sont optimales, et l'utilisation d'une telle fonction par la suite, comme `umfpack`, nous fera gagner du temps, surtout si nous voulons faire de meilleures simulations, c'est-à-dire plus de calculs.

1.3 Seconde approche

L'algorithme de splitting directionnel proposé par l'énoncé est bien plus gourmand en calculs que ce qui a été fait jusqu'à présent puisqu'il fait appel un grand nombre de fois au solveur 1D précédent, d'où la nécessité de réduire au maximum le temps de calcul. Nous nous proposons ici d'utiliser la fonction prédéfinie `umfpack` susmentionnée afin d'optimiser le solveur 1D.

En prenant en compte les formats des objets manipulés afin d'appliquer la routine de résolution unidimensionnelle selon x et selon y, nous obtenons les représentations 3D suivantes :

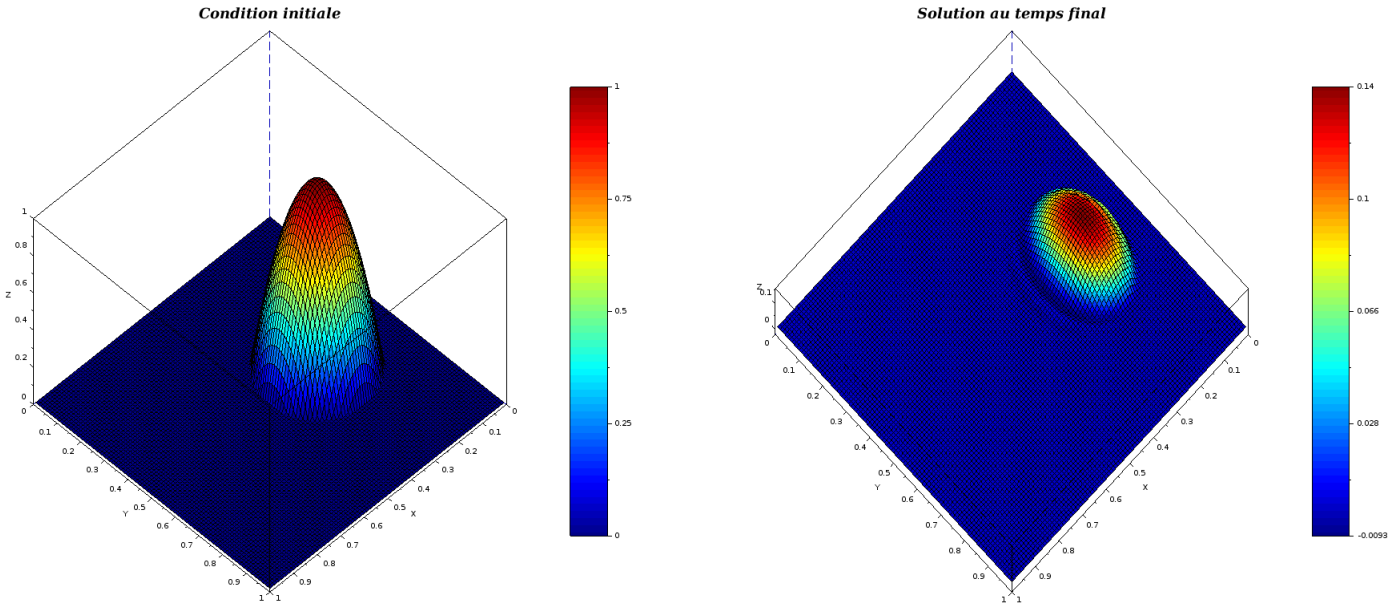


Fig. 2 – *Résolution avec le solveur 2D pour* $\kappa = 10^{-4}$

Les résultats sont ici plus visuels et permettent de mieux juger de ce qu'il se passe. Pour $\kappa = 10^{-4}$, nous pouvons tirer les mêmes conclusions que pour la simulation précédente : la solution est atténuée et légèrement translatée dans l'espace.

Une nouvelle simulation, prenant cette fois-ci $\kappa = 10^{-2}$, conduit au résultat suivant :

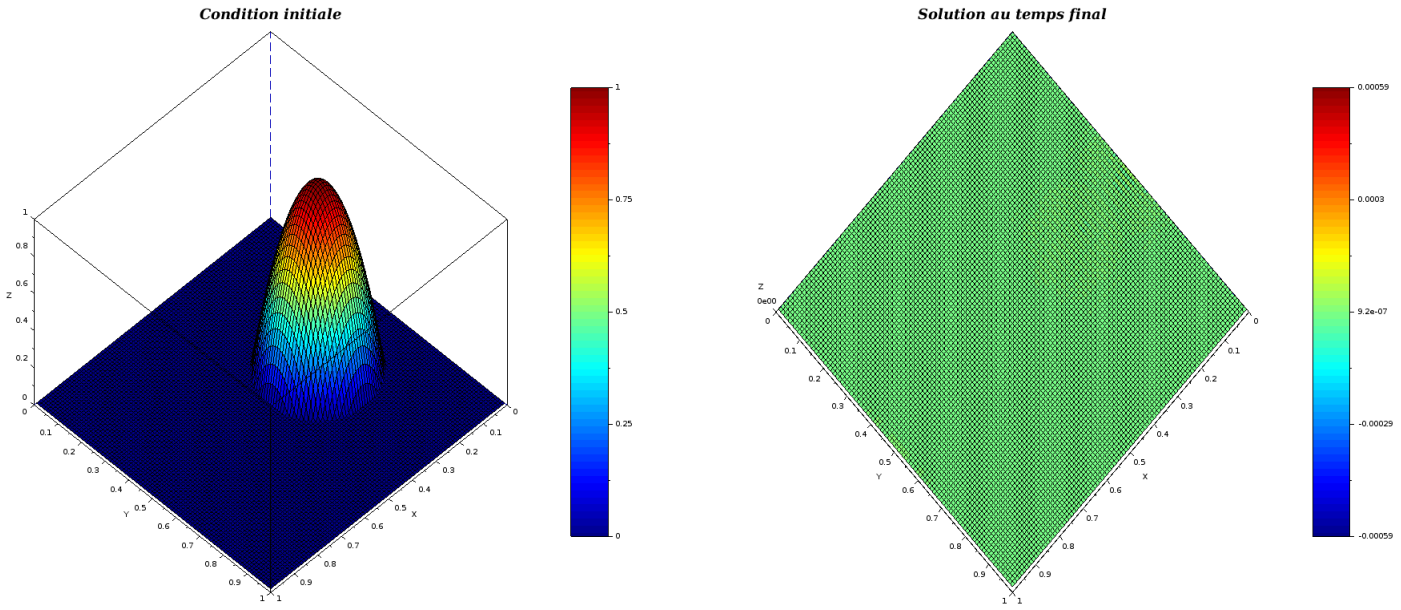


Fig. 3 – *Résolution avec le solveur 2D pour* $\kappa = 10^{-2}$

La solution semble être à peu près constante, environ égale à $9.2 \cdot 10^{-7}$. Elle est donc très proche de 0, comme cela était le cas sur le graphique correspondant de la figure 1. Il est possible que, suivant l'hypothèse faite précédemment, la solution soit très étendue spatialement et proche de 0, et qu'il est donc plus difficile de le voir sur l'affichage 3D. Cependant, il est aussi possible qu'il faille utiliser un pas de discrétisation plus fin encore pour obtenir des résultats précis.

Dans tous les cas, la résolution des calculs nécessite plusieurs minutes et si l'on veut atteindre une meilleure précision, il faut augmenter encore le nombre de calculs à effectuer. Cela confirme l'importance d'optimiser les routines utilisées.

2 Résolution du problème de Poisson

Nous cherchons ici à résoudre une équation de Poisson en deux dimensions sur un domaine Ω rectangulaire périodique. Plus précisément, il s'agit de résoudre une équation d'inconnue ψ de la forme :

$$\frac{\partial^2 \psi}{\partial x^2}(x,y) + \frac{\partial^2 \psi}{\partial y^2}(x,y) = f(x,y) \tag{1}$$

où $f, \psi : \Omega \to \mathbf{R}$.

2.1 Construction de la méthode de résolution

Pour résoudre (1) efficacement et différemment de la partie précédente nous utiliserons principalement deux outils, les séries de Fourier et la transformée de Fourier discrète. En effet, si l'on calcule la décomposition de f et ψ en série de Fourier et qu'on l'injecte dans (1), l'on obtient par identification en passant à la limite une relation entre les coefficients de la série de ψ et celle de f. En effet, en notant $\hat{\psi}_{pq}$ et $\hat{f}_{pq}$ ces coefficients, où $p, q \in \mathbf{N}$, $(p,q) \neq (0,0)$, il vient :

$$\hat{\psi}_{pq} = \frac{\hat{f}_{pq}}{\boldsymbol{k_x^2} + \boldsymbol{k_y^2}}. \tag{2}$$

Démonstration.

> Écrivons les sommes partielles des décompositions en série de Fourier de et f et des dérivées partielles de ψ. Soient $m, n \in \mathbf{N}^*$. Il vient :

$$S_{nm}(f) = \sum_{p=-n}^{n} \sum_{q=-m}^{m} \hat{f}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x} \tag{3}$$

$$\frac{\partial^2 S_{nm}(\psi)}{\partial x^2} = \sum_{p=-n}^{n} \sum_{q=-m}^{m} \boldsymbol{k_x^2} \hat{\psi}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x} \tag{4}$$

$$\frac{\partial^2 S_{nm}(\psi)}{\partial y^2} = \sum_{p=-n}^{n} \sum_{q=-m}^{m} \boldsymbol{k_y^2} \hat{\psi}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x} \tag{5}$$

En sommant (4) et (5), nous obtenons :

$$\frac{\partial^2 S_{nm}(\psi)}{\partial x^2} + \frac{\partial^2 S_{nm}(\psi)}{\partial x^2} = \sum_{p=-n}^{n} \sum_{q=-m}^{m} (\boldsymbol{k_x^2} + \boldsymbol{k_y^2}) \hat{\psi}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x}. \tag{6}$$

En passant à la limite sur n et sur m dans (3) et dans (6), nous retrouvons l'équation (1) écrite sous une forme équivalente :

$$\sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} (\boldsymbol{k_x^2} + \boldsymbol{k_y^2}) \hat{\psi}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x} = \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} \hat{f}_{pq} e^{\boldsymbol{k_y} y} e^{\boldsymbol{k_x} x}. \tag{7}$$

Ainsi, en identifiant les coefficients (lorsque $(p,q) \neq (0,0)$, auquel cas on impose $\hat{\psi}_{00} = 0$), nous obtenons la relation suivante :

$$\hat{\psi}_{pq} = \frac{\hat{f}_{pq}}{\boldsymbol{k_x^2} + \boldsymbol{k_y^2}}.$$

L'intérêt d'utiliser la transformée de Fourier discrète est de pouvoir calculer les coefficients de la série de Fourier de f sans avoir à passer par du calcul intégral. f étant connue, il est aisé de calculer $\hat{f}_{pq}$ à l'aide de la transformée discrète et d'exprimer ceux de ψ à partir de la formule (2). Au lieu de calculer explicitement la série de Fourier de ψ à partir des coefficients obtenus, il suffit de prendre la transformée de Fourier discrète inverse, qui nous donne directement la solution attendue. Un autre argument en faveur de l'utilisation de la transformée de Fourier discrète est que nous disposons d'un algorithme de calcul efficace, la transformée de Fourier rapide, et que cet algorithme est lui-même implémenté de manière optimale dans `scilab` (fonction `fft`).

2.2 Résultats et analyse

Commençons d'abord par tester sur un cas unidimensionnel, avec une fonction de référence, le solveur Poisson FFT que nous avons implémenté. Tentons de résoudre le problème $\Delta\psi_\alpha(x,y) = f(x,y)$, où $f(x,y) = sin(2\pi x)sin(2\pi y)$ et ψ_α est de la forme $\psi_\alpha = \alpha sin(2\pi x)sin(2\pi y)$.

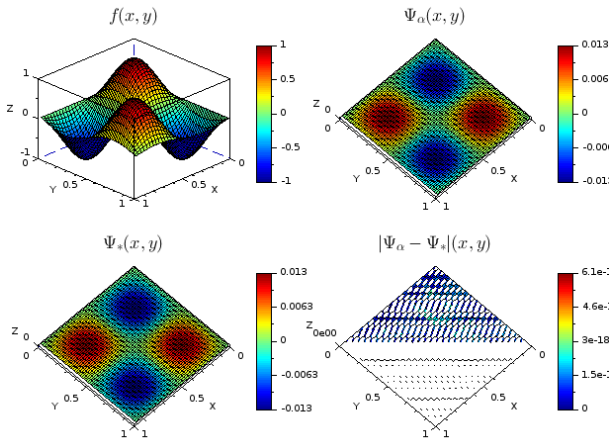


FIG. 4 – *Solveur de Poisson sur le cas analytique* $\Delta\psi_\alpha(x,y) = sin(2\pi x)sin(2\pi y)$

Nous avons représenté sur la figure 4 la fonction f, la solution analytique Ψ_α de l'équation, la solution approchée Ψ_* obtenue grâce au solveur Poisson FFT, et l'erreur absolue $|\Psi_\alpha - \Psi_*|$ commise par notre solveur.

A l'aide de l'échelle de couleurs, il est aisé de constater que la solution analytique et la solution approchée sont de même nature et similaires à f. En outre, le graphe de l'erreur absolue montre que l'approximation n'est pas parfaite, mais tout de même extrêmement satisfaisante : l'erreur maximale n'est que de l'ordre de 10^{-17}. Cela montre que la méthode retenue pour la résolution du problème est efficace.

Maintenant que nous nous sommes assurés de la précision et de la justesse de notre solveur, il s'agit de résoudre le problème plus général suivant dans $\mathbf{R}^2$:

$$\Delta \boldsymbol{u} = -\nabla \wedge \boldsymbol{\omega}, \tag{8}$$

où $\boldsymbol{u}$ désigne le vecteur vitesse et $\boldsymbol{\omega}$ la vorticité. Puisque nous travaillons dans $\mathbf{R}^2$, la projection de l'équation selon x et selon y nous donne deux problèmes de Poisson, indépendants, que nous savons donc résoudre grâce au solveur Poisson FFT :

$$\Delta \boldsymbol{u}_x(x,y) = -\frac{\partial \omega}{\partial y}(x,y), \tag{9}$$

$$\Delta \boldsymbol{u}_y(x,y) = \frac{\partial \omega}{\partial x}(x,y). \tag{10}$$

Malgré le test précédent, mieux vaut tracer le graphe d'une solution de référence ainsi que l'erreur absolue commise afin de s'assurer de la justesse des résultats. Cherchons la relation entre les coefficients de Fourier de $\hat{\boldsymbol{u}}_x$ et $\hat{\boldsymbol{u}}_y$ en fonction de ceux de $\hat{\boldsymbol{\omega}}$ et de $\boldsymbol{k_x}$ et $\boldsymbol{k_y}$ pour tous p,$q \in \mathbf{N}$, $(p,q) \neq (0,0)$ sinon les coefficients sont initialisés à 0. Nous avons :

$$\hat{\boldsymbol{u}}_{x,pq} = -\frac{\boldsymbol{k_y}\hat{\boldsymbol{\omega}}_{pq}}{\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2} \tag{11}$$

$$\hat{\boldsymbol{u}}_{y,pq} = \frac{\boldsymbol{k_x}\hat{\boldsymbol{\omega}}_{pq}}{\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2}. \tag{12}$$

Démonstration.

Procédons comme précédemment. Passons à la limite dans les sommes partielles des séries de Fourier de $\boldsymbol{u}_x$ et $\boldsymbol{u}_y$ et substituons le résultat dans les équations (9) et (10) :

$$\sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty}(\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2)\hat{\boldsymbol{u}}_{x,pq}e^{\boldsymbol{k_y}y}e^{\boldsymbol{k_x}x} = -\sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty}\boldsymbol{k_y}\hat{\boldsymbol{\omega}}_{pq}e^{\boldsymbol{k_y}y}e^{\boldsymbol{k_x}x}$$

et

$$\sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty}(\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2)\hat{\boldsymbol{u}}_{y,pq}e^{\boldsymbol{k_y}y}e^{\boldsymbol{k_x}x} = \sum_{p=-\infty}^{\infty}\sum_{q=-\infty}^{\infty}\boldsymbol{k_x}\hat{\boldsymbol{\omega}}_{pq}e^{\boldsymbol{k_y}y}e^{\boldsymbol{k_x}x}.$$

D'où en identifiant les coefficients :

$$\hat{\boldsymbol{u}}_{x,pq} = -\frac{\boldsymbol{k_y}\hat{\boldsymbol{\omega}}_{pq}}{\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2}$$

$$\hat{\boldsymbol{u}}_{y,pq} = \frac{\boldsymbol{k_x}\hat{\boldsymbol{\omega}}_{pq}}{\boldsymbol{k_x}^2 + \boldsymbol{k_y}^2}.$$

D'où le résultat.

La résolution numérique pour la condition $\boldsymbol{\omega}$ et les solutions de référence $\boldsymbol{u}_x$ et $\boldsymbol{u}_y$

$$\begin{cases} \boldsymbol{\omega}(x,y) = 8\pi^2 cos(2\pi x)cos(2\pi y) \\ \\ \boldsymbol{u}_x(x,y) = -2\pi cos(2\pi x)sin(2\pi y) \\ \boldsymbol{u}_y(x,y) = 2\pi sin(2\pi x)cos(2\pi y) \end{cases}$$

donne en sortie les graphes suivants :

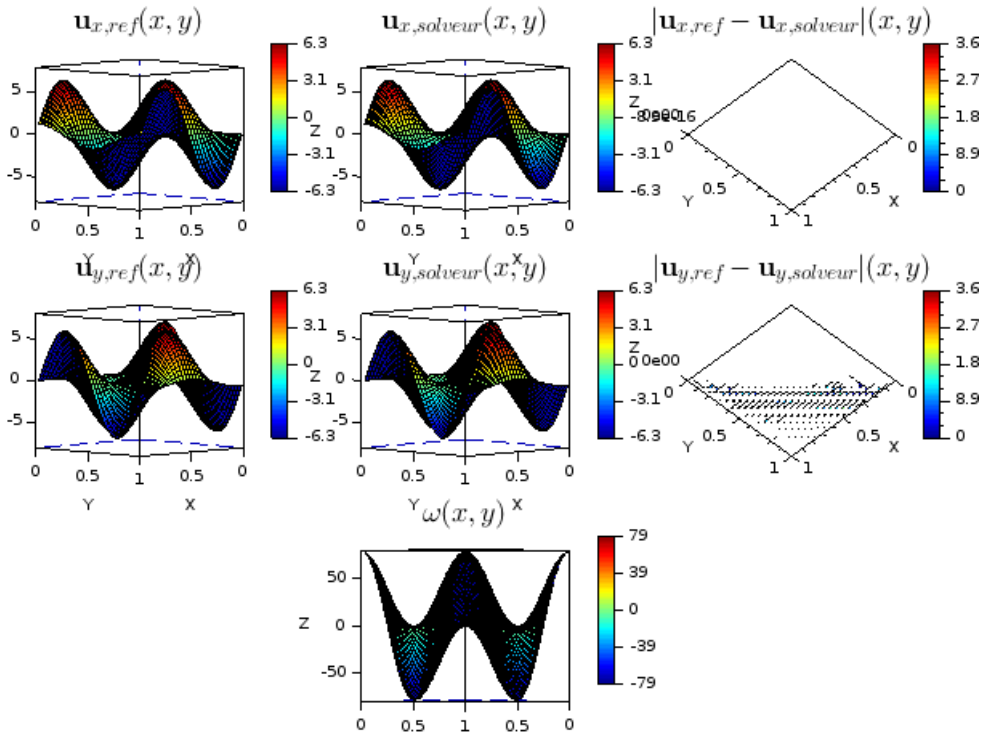


Fig. 5 – *Utilisation du solveur Poisson curl 2D pour la résolution de* $\Delta \boldsymbol{u} = -\nabla \wedge \boldsymbol{\omega}$

Nous pouvons donc constater que les solutions de référence et du solveur sont similaires. En outre, le graphe représentant l'erreur absolue permet de conclure que le solveur Poisson FFT a bien fonctionné et est plutôt efficace.

3 Simulation numérique

Il s'agit à présent de résoudre le problème initialement posé et d'en faire une simulation numérique sous forme de vidéo, en réutilisant les routines déjà définies et en implémentant un algorithme de splitting directionnel présenté dans l'énoncé. Pour ce faire, nous nous plaçons sur $\Omega = [0,1]^2$ et sur un intervalle de temps $[0, 1.50]$.

3.1 Construction de la méthode de résolution

Afin d'initialiser nos fonctions `scilab`, nous devons en premier lieu déterminer le champ de vorticité initial. Celui-ci s'écrit à partir des conditions initiales $\boldsymbol{u_x^0}$ et $\boldsymbol{u_y^0}$ pour tout $(x,y) \in \Omega$:

$$\omega^0(x,y) = \frac{\partial \boldsymbol{u_y^0}}{\partial x}(x,y) - \frac{\partial \boldsymbol{u_x^0}}{\partial y}(x,y), \tag{13}$$

où

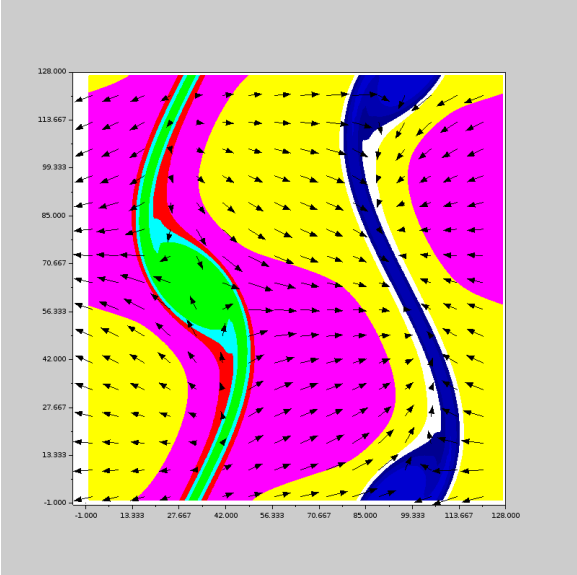
$$\frac{\partial \boldsymbol{u_x^0}}{\partial x}(x,y) = 2\pi\delta cos(2\pi x) \tag{14}$$

$$\frac{\partial \boldsymbol{u_y^0}}{\partial y}(x,y) = \begin{cases} \frac{\rho}{cosh(\rho(y-0.25))^2} & \text{si } y \leq 0.5, \\ \frac{-\rho}{cosh(\rho(0.75-y))^2} & \text{sinon.} \end{cases} \tag{15}$$

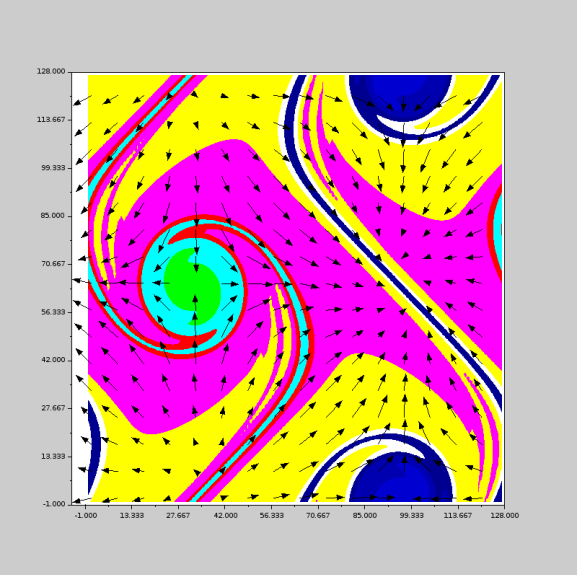
On résout ensuite en appliquant l'algorithme vu en fin de partie 1.A de l'énoncé chaque itération temporelle :

- On ajuste le pas de temps dt pour garantir le critère de stabilité,
- On calcule Ux et Uy à partir du champ de vorticité,
- On calcule le champ de vorticité en réalisant un splitting directionnel et en utilisant les fonctions des parties précédentes.

3.2 Résultats et analyse

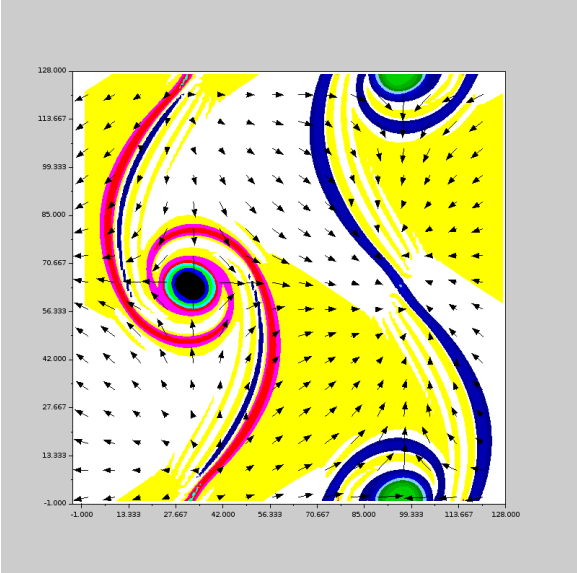


(a) *t=0.8*

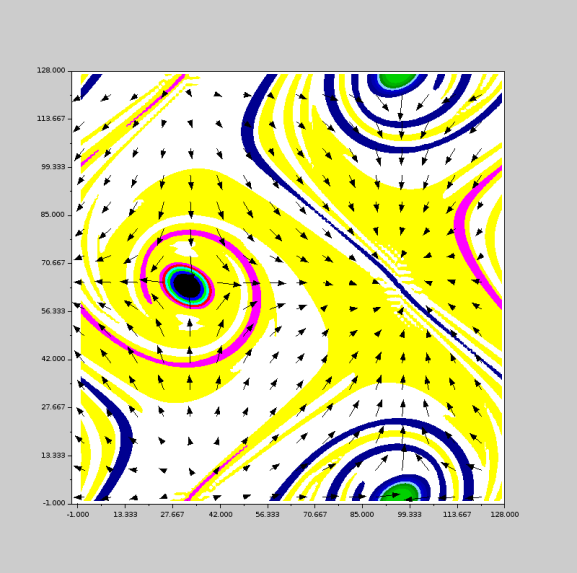


(b) *t=1.2*

Fig. 6 – *Isocontours du champ de vorticité – Simulation avec* $\nu = 10^{-4}$ *et* $\rho = 30.0$



(a) *t=0.8*



(b) *t=1.2*

Fig. 7 – *Isocontours du champ de vitesse – Simulation avec* $\nu = 0.5 \cdot 10^{-4}$ *et* $\rho = 100.0$

La simulation 1 (figure 6) s'effectue en 250 itérations alors que la 2 (figure 7) en nécessite 294. Cela semble assez logique car les valeurs choisies pour ν et ρ entraînent une convergence plus rapide dans le temps. Ainsi, le pas de temps est adapté et diminue afin de garantir la stabilité de la méthode.

4 Conclusion

Ce TP a été pour nous l'occasion de mettre en application nos cours et d'avoir des retours concrets et chiffrés sur les méthodes employées en termes de vitesse de calcul et de précision. Cela nous a également permis de faire de gros progrès

en ce qui concerne la manipulation de `scilab`. Pource qui est du format, cela nous a permis de développer une démarche complète de résolution et simulation numérique d'un problème à partir de sa mise en équation mathématique.

Bien que réalisé sur des cas particuliers, ce TP nous a montré qu'il était relativement aisé de donner vie aux équations et de visualiser des phénomènes physiques en faisant varier bon nombre de paramètres, chose très difficile à faire expérimentalement.