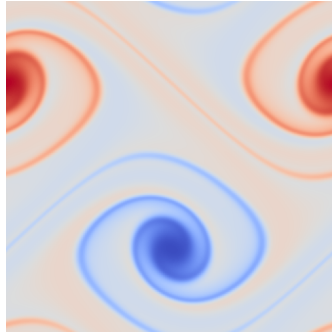


TP Méthodes Numériques : *Simulation d'écoulement fluide*



De nombreux processus industriels font intervenir des écoulements fluides. Des prévisions fiables sur les caractéristiques de ces écoulements sont nécessaires pour la conception, l'optimisation des processus et l'analyse de la sécurité des appareils et processus associés. Les études expérimentales sont coûteuses et, dans la plupart des cas, non transférables à des géométries modifiées, à différentes échelles ou systèmes de matériaux. Pour cette raison, il existe un besoin évident d'outils numériques.

Ce TP constitue une introduction à l'intégration en temps des équations aux dérivées partielles à travers l'étude d'un modèle physique d'écoulement de fluide de type Navier-Stokes incompressible en deux dimensions.

Ce problème sera découpé en plusieurs sous problèmes qui vous permettront de vous familiariser avec l'implémentation et l'utilisation de certains schémas d'intégration en temps, de schémas de différences finies explicites et implicites ainsi que d'un solveur spectral. Le logiciel `scilab` vous permettra d'expérimenter la résolution de systèmes linéaires creux et vous offrira différents moyens de visualiser les résultats numériques.

Informations Pratiques

Les consignes ci-dessous sont à respecter lors de votre TP :

- **Travail en binôme.**
- Rédiger un **compte-rendu**. Il n'y a qu'un seul rapport à rendre par binôme. Vous devez expliciter les méthodes employées, présenter et commenter avec soin les résultats obtenus. Le compte-rendu sera dactylographié, nous conseillons d'utiliser le logiciel `LATEX` qui est beaucoup employé pour la rédaction d'articles scientifiques. Ce compte-rendu ne comportera pas de programmes. Les programmes `scilab` seront fournis séparément **conformément au squelette de code fourni**. La qualité du rapport et des graphiques, la lisibilité du code et la pertinence des commentaires seront pris en compte dans la note du TP.
- **Remise du rapport et des programmes.** Le TP est à rendre au plus tard le **18 Mars 8h00**. Il faudra déposer sur `TEIDE` votre rapport sous la forme d'une archive `tpmn_nom1_nom2.tar.gz` contenant un fichier

`rapport.pdf` ainsi que vos programmes `scilab` dans le sous dossier `src` tel que fourni au début du TP.

- **Séances de TP.** Nous répondrons à vos questions sur ce TP durant une séance organisée pour chaque groupe.
- **Contacts :**
 - Jean-Baptiste.Keck@univ-grenoble-alpes.fr
 - Olivier.Ozenda@univ-grenoble-alpes.fr

Introduction

Dans tout le sujet, les variables notées en **gras** sont soit des éléments de \mathbb{C} , soit des vecteurs dans \mathbb{R}^2 ou \mathbb{R}^3 , $\mathbf{x} \cdot \mathbf{y}$ désigne le produit scalaire dans \mathbb{R}^2 ou \mathbb{R}^3 , et $\mathbf{x} \wedge \mathbf{y}$ désigne le produit vectoriel dans \mathbb{R}^3 .

On se place dans un domaine $\Omega \subset \mathbb{R}^n$ on note $\mathbf{x} \in \Omega$ la variable d'espace, $t \in [0, T]$ la variable de temps, $P(\mathbf{x}, t)$ le champ scalaire de pression instantané dans le fluide et $\mathbf{u}(\mathbf{x}, t)$ le champ de vecteur représentant la vitesse instantanée du fluide dans chaque direction de l'écoulement. La vitesse est donc décrite par n composantes scalaires $\mathbf{u} = [\mathbf{u}_x \ \mathbf{u}_y]^T$ en 2D et $\mathbf{u} = [\mathbf{u}_x \ \mathbf{u}_y \ \mathbf{u}_z]^T$ en 3D.

On considère les équations de Navier-Stokes incompressibles dans leur formulation vitesse-pression (\mathbf{u}, P) en dimension 2 ou 3 :

$$\nabla \cdot \mathbf{u} = 0 \quad (1a)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \Delta \mathbf{u} - \frac{\nabla P}{\rho} \quad (1b)$$

Le fluide considéré ici possède une densité ρ constante et une viscosité cinématique ν constante (ces deux constantes physiques sont strictement positives). On donne l'expression des opérateurs **nabla** ∇ et **laplacien** Δ en annexe.

Le but du TP est d'effectuer la simulation numérique d'un fluide incompressible en 2D dans une boîte unitaire périodique sur `scilab`. Pour cela, on découpe ce TP en 3 sections distinctes :

- La section 1 Introduit la résolution d'une équation d'advection-diffusion en différences finies, en 1D puis en 2D. Nous proposons également dans cette section d'implémenter un solveur linéaire.
- La section 2 est dédiée à l'implémentation d'un solveur spectral pour l'équation de Poisson sur un domaine périodique.
- La section 3 focalise sur la description et la résolution en temps du problème physique en réutilisant certaines routines implémentées dans les sections précédentes.

Les deux premières sections sont indépendantes. La section 3 utilise certaines routines implémentées dans la section 1 et 2. Les routines de test fournies vous permettront de valider votre implémentation au fur et à mesure sur un environnement de type linux, dans le cas où votre implémentation est conforme au squelette de code fourni (ce qui est requis pour la remise du code sur TEIDE).

Énoncé du problème à résoudre

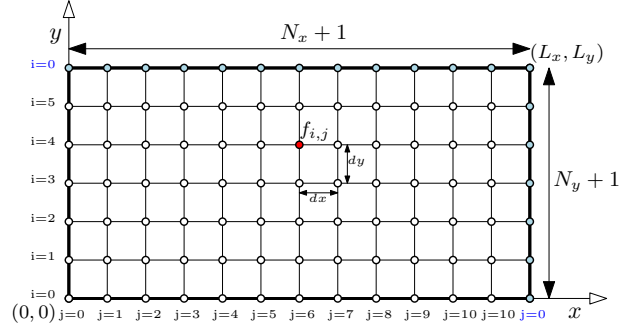
Nous allons travailler sur la formulation vitesse-vorticité (\mathbf{u}, ω) des équations de Navier-Stokes en dimension deux :

$$\frac{\partial \mathbf{u}_x}{\partial x} + \frac{\partial \mathbf{u}_y}{\partial y} = 0 \quad (2a)$$

$$\omega = \frac{\partial \mathbf{u}_y}{\partial x} - \frac{\partial \mathbf{u}_x}{\partial y} \quad (2b)$$

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = \nu \Delta \omega \quad (2c)$$

La construction de ce système est donnée à titre d'information dans l'annexe B. Dans la suite du TP, le domaine de simulation $\Omega = [0, L_x] \times [0, L_y]$ sera considéré rectangulaire et périodique dans chaque direction. La discrétisation spatiale des variables se fait sur une grille Cartésienne de $N = N_x \times N_y$ points :



On peut résoudre ce problème à l'aide de l'algorithme (A) :

1. On initialise ω sur tout le domaine de calcul à $t = 0$.
2. Tant que $t < T$:
 - (a) Calculer \mathbf{u} à partir de ω grâce aux équations (2a) et (2b).
 - (b) Résoudre l'équation de transport-diffusion sur ω (2c).
 - (c) $t = t + dt$.

On prendra un pas de temps dt assez petit de tel sorte à ce que l'on puisse résoudre les deux sous-problèmes (a) et (b) en fonction du pas d'espace dx choisi et on note N_t le nombre de pas de temps vérifiant $N_t dt = T$.

1 Résolution de l'équation de transport diffusion

On considère un problème d'advection-diffusion 1D avec des conditions aux bords périodiques

$$\begin{cases} \frac{\partial}{\partial t} \phi(x, t) + c(x) \frac{\partial}{\partial x} \phi(x, t) - \kappa \frac{\partial^2}{\partial x^2} \phi(x, t) = 0 & \forall (x, t) \in]0; 1[\times]0; 1[\\ \phi(0, t) = \phi(1, t) & \forall t \in]0; 1[\\ \phi(x, 0) = \phi_0(x) & \forall x \in]0; 1[\end{cases} \quad (3)$$

où c est une fonction bornée et κ une constante positive et la condition initiale est définie par :

$$\begin{cases} 0 \leq x < 0.25 & \Rightarrow \phi_0(x) = 0 \\ 0.25 \leq x < 0.375 & \Rightarrow \phi_0(x) = 2(x - 0.25) \\ 0.375 \leq x < 0.5 & \Rightarrow \phi_0(x) = 2(0.5 - x) \\ 0.5 \leq x \leq 1 & \Rightarrow \phi_0(x) = 0 \end{cases} \quad (4)$$

On choisit de discrétiser l'équation (3) en effectuant le schéma numérique suivant

$$\begin{aligned} \phi(x, t + dt) = & \phi(x, t) - c(x, t) \frac{dt}{2dx} (\phi(x + dx, t) - \phi(x - dx, t)) \\ & + c(x)^2 \frac{dt^2}{2dx^2} (\phi(x + dx, t) - 2\phi(x, t) + \phi(x - dx, t)) \\ & - \kappa \frac{dt}{dx^2} (\phi(x + dx, t + dt) - 2\phi(x, t + dt) + \phi(x - dx, t + dt)) \end{aligned} \quad (5)$$

Question 1. Donner deux matrices M et N telles que pour $0 \leq k \leq N_t - 1$, le système

$$N \begin{bmatrix} \phi(0, (k+1)dt) \\ \vdots \\ \phi(ndx, (k+1)dt) \\ \vdots \\ \phi((N_x - 1)dx, (k+1)dt) \end{bmatrix} = M \begin{bmatrix} \theta(0, kdt) \\ \vdots \\ \theta(ndx, kdt) \\ \vdots \\ \theta((N_x - 1)dx, kdt) \end{bmatrix}$$

implémente le schéma (5).

On remarquera que $\phi(0, t) = \phi(1, t)$ par périodicité du domaine et plus généralement $\phi(kdx, t) = \phi(k \bmod N_x, t)$ pour tout entier relatif k .

Question 2. L'implémentation du schéma (5) nécessite la résolution d'un système linéaire à chaque pas de temps. En remarquant que la matrice N est symétrique définie positive, justifier le choix d'une méthode de type Cholesky pour effectuer cette résolution.

Question 3. Compléter le script `my_cholesky.sce` implémentant une fonction `my_cholesky` prenant en entrée une matrice carrée N et un vecteur S et renvoyant le vecteur U tel que $NU = S$. La fonction `my_choleski` fera appel à trois autres fonctions. Une fonction `cholesky_fact` réalisant la factorisation de Choleski et deux fonctions `up_sweep_cholesky` et `down_sweep_cholesky` résolvant des systèmes triangulaires.

Question 4. Compléter le script `diff-conv.sce` afin d'implémenter la résolution de l'équation (3) en utilisant la méthode (5) ainsi que le solveur linéaire `my_cholesky.sce`. Le script doit afficher sur le même graphique la condition initiale et la solution au temps final. Tester celui-ci avec la condition initiale (4), la vitesse de convection $c(x) = 0.4(x - 0.25)$ et pour différents coefficients de diffusion $\kappa \in \{10^{-4}; 10^{-3}; 10^{-2}\}$. Inclure vos résultats dans le rapport accompagné de vos commentaires éventuels. On utilisera les valeurs de dx et dt données dans le fichier `diff-conv.sce`, à noter le schéma numérique donné est stable si $\max_x |c(x)| \frac{dt}{dx} < 1$.

A partir de maintenant, on représentera, a chaque pas de temps, les versions discrétisées des fonctions scalaires par les tableaux $f_{i,j}^k \simeq f(jdx, idy, kdt)$. On va appliquer les résultats trouvés en 1D en réalisant un splitting directionnel. On considère le problème de transport-diffusion en 2D :

$$\left\{ \begin{array}{l} \frac{\partial}{\partial t} \phi(x, y, t) \\ + \mathbf{c}_x(x, y, t) \frac{\partial}{\partial x} \phi(x, y, t) - \kappa \frac{\partial^2}{\partial x^2} \phi(x, y, t) \\ + \mathbf{c}_y(x, y, t) \frac{\partial}{\partial y} \phi(x, y, t) - \kappa \frac{\partial^2}{\partial y^2} \phi(x, y, t) = 0 \quad \forall (x, y, t) \in]0; 1[\times]0; 1[\\ \phi(0, y, t) = \phi(1, y, t) \quad \forall (y, t) \in]0; 1[\times]0; 1[\\ \phi(x, 0, t) = \phi(x, 1, t) \quad \forall (x, t) \in]0; 1[\times]0; 1[\\ \phi(x, y, 0) = \phi_0(x, y) \quad \forall (x, y) \in]0; 1[\times]0; 1[\end{array} \right. \quad (6)$$

où \mathbf{c} est une fonction bornée et κ une constante positive. Les fonctions \mathbf{c}_x , \mathbf{c}_y et ϕ sont périodiques de période 1 selon chaque axe x et y .

Le principe du schéma de splitting consiste à résoudre (6) à chaque pas de temps en deux étapes.

- On omet d'abord les termes différentiels en y et on résout l'équation 1D résultante entre kdt et $(k+1)dt$:

$$\frac{\partial}{\partial t} \phi(x, y, t) + \mathbf{c}_x(x, y, t) \frac{\partial}{\partial x} \phi(x, y, t) - \kappa \frac{\partial^2}{\partial x^2} \phi(x, y, t) = 0 \quad (7)$$

- On omet ensuite les termes différentiels en x et on résout l'équation 1D résultante entre kdt et $(k+1)dt$:

$$\frac{\partial}{\partial t} \phi(x, y, t) + \mathbf{c}_y(x, y, t) \frac{\partial}{\partial y} \phi(x, y, t) - \kappa \frac{\partial^2}{\partial y^2} \phi(x, y, t) = 0 \quad (8)$$

On remarque tout de suite que les équations 1D résultantes sont les mêmes sur chaque axe (modulo le remplacement de \mathbf{c}_x par \mathbf{c}_y) et qu'il s'agit exactement du problème que l'on a résolu en question 4 et implémenté en tant que `solveur_1D`.

Pour la résolution, on procédera donc de la manière suivante :

1. Itérer sur toutes les lignes de ϕ et \mathbf{c}_x et appeler indépendamment `solveur_1D`.
2. Itérer sur toutes les colonnes de ϕ et \mathbf{c}_y et appeler indépendamment `solveur_1D`. Ici il faudra faire attention puisque le solveur ne sait résoudre qu'avec des lignes, il faudra prendre le soin de transposer les colonnes en lignes au préalable avec l'opérateur ' de `scilab` et d'effectuer l'opération inverse sur le résultat afin de récupérer une colonne de ϕ .

On note que maintenant la contrainte de stabilité vient à la fois de l'advection-diffusion en x et en y . Il faut donc choisir un pas de temps qui soit le minimum des pas de temps donnés par les critères dans chaque direction.

Question 5. Compléter les scripts `dif-conv-f.sce` et `dif-conv-2D.sce` implémentant cette méthode et effectuer un test avec le script `test-2D.sce` que l'on aura complété pour afficher la condition initiale et la solution temps final sur deux graphiques séparés. Afin de diminuer le temps de calcul, on

utilisera la **fonction prédéfinie** `umfpack` pour effectuer les résolutions de systèmes linéaires nécessaires. Le script `dif-conv-f.sce` contiendra une fonction `solveur_1D` implémentant le schéma numérique vu à la question précédente.

2 Résolution du problème de Poisson

Dans cette partie nous allons utiliser les séries de Fourier afin de résoudre une équation de Poisson en 2D sur un domaine $\Omega = [0, L_x] \times [0, L_y]$ rectangulaire périodique dans chaque direction. Pour toute fonction $g(x, y)$ définie sur Ω on aura donc que $g(x, y) = g(x + L_x, y + L_y)$.

On considère l'équation scalaire suivante définie sur tout le domaine Ω :

$$\frac{\partial \psi}{\partial x^2}(x, y) + \frac{\partial \psi}{\partial y^2}(x, y) = f(x, y) \quad (9)$$

Définition 2.1. On définit la **série de Fourier** $S(f)$ d'une fonction périodique intégrable $f : \Omega \rightarrow \mathbb{R}$ la série de fonctions ayant pour somme partielle :

$$S_{nm}(f) = \sum_{p=-n}^n \sum_{q=-m}^m c_{pq}(f) e^{+\frac{2i\pi q}{L_x}x} e^{+\frac{2i\pi p}{L_y}y}$$

avec

$$c_{pq}(f) = \frac{1}{L_x L_y} \int_{\Omega} f(x, y) e^{-2i\pi \left(\frac{qx}{L_x} + \frac{py}{L_y} \right)} dx dy$$

Question 6. Soit $n > 1$ et $m > 1$, on pose $\mathbf{k}_x = \frac{2i\pi q}{L_x}$ et $\mathbf{k}_y = \frac{2i\pi p}{L_y}$.

$$S_{nm}(f) = \sum_{p=-n}^n \sum_{q=-m}^m \hat{f}_{pq} e^{+\mathbf{k}_y y} e^{+\mathbf{k}_x x}$$

$$S_{nm}(\psi) = \sum_{p=-n}^n \sum_{q=-m}^m \hat{\psi}_{pq} e^{+\mathbf{k}_y y} e^{+\mathbf{k}_x x}$$

Exprimer pour tout $(p, q) \in \llbracket -n, n \rrbracket \times \llbracket -m, m \rrbracket$ les coefficients de Fourier $\hat{\psi}_{pq}$ en fonction de \hat{f}_{pq} , \mathbf{k}_x et \mathbf{k}_y à partir de l'équation (9) en identifiant les coefficients des deux séries quand cela est possible. On imposera $\hat{\psi}_{00} = 0$.

Définition 2.2. On discrétise toutes les fonctions $f : \Omega \rightarrow \mathbb{R}$ sur une grille Cartésienne doublement périodique de $N = N_x \times N_y$ points :

$$\begin{aligned} dx &= \frac{L_x}{N_x}, \quad x_n = ndx \quad \forall n \in \llbracket 0, N_x - 1 \rrbracket \\ dy &= \frac{L_y}{N_y}, \quad y_m = mdy \quad \forall m \in \llbracket 0, N_y - 1 \rrbracket \\ f_{mn} &= f(x_n, y_m) = f(ndx, mdy) \end{aligned}$$

L'expression de la **transformée de Fourier discrète** de f est donnée par :

$$\hat{f}_{pq} = \frac{1}{N} \sum_{m=0}^{N_y-1} \sum_{n=0}^{N_x-1} f_{mn} e^{-2i\pi \left[\frac{pm}{N_y} + \frac{qn}{N_x} \right]}$$

Cette transformée va nous permettre de calculer les coefficients de la série de Fourier de façon efficace en seulement $\mathcal{O}(N \log(N))$ opérations avec $N = N_x N_y$. On peut noter que dans notre cas f sera toujours une fonction à valeurs réelles tandis que \hat{f} sera une fonction à valeurs complexes. Dans ce cas, il est possible d'exploiter la symétrie hermitienne de \hat{f} afin d'en réduire l'empreinte mémoire d'un facteur deux mais pour simplifier nous ne le ferons pas dans ce TP.

Question 7. Dans le cas périodique discret, on doit calculer les nombres d'ondes \mathbf{k}_* dans chaque direction de la façon suivante :

$$k(N) = \begin{cases} [0, 1, \dots, N/2 - 1, & -N/2, & \dots, -1] & \text{si } N \text{ est pair} \\ [0, 1, \dots, (N-1)/2, & -(N-1)/2, & \dots, -1] & \text{sinon} \end{cases}$$

$$\mathbf{k}_x = \frac{2i\pi}{L_x} k(N_x)$$

$$\mathbf{k}_y = \frac{2i\pi}{L_y} k(N_y)$$

Implémenter la fonction `fftfreq` dans le fichier `poisson.sce` qui calcule les nombres d'ondes en fonction d'un nombre de points N et d'une longueur L donnés. On pourra s'aider des routines `modulo`, `linspace` et `cat` de `scilab`.

Question 8. Implémenter la routine `poisson_2d` dans le fichier `poisson.sce` qui résout le problème de Poisson en deux dimensions. Cette routine prend en entrée un tableau de réels f de taille (N_y, N_x) et effectue les tâches suivantes :

1. Génération des vecteurs \mathbf{k}_x et \mathbf{k}_y en appelant deux fois `fftfreq`.
2. Calcul de $\hat{f} = \text{fft}(f)$.
3. Calcul de $\hat{\psi}$ en fonction de \hat{f} , \mathbf{k}_x et \mathbf{k}_y avec $\hat{\psi}_{0,0} = 0$. L'expression pour calculer $\hat{\psi}$ est la même que celle trouvée en question 7.
4. Calcul de $\psi = \text{ifft}(\hat{\psi})$ par transformée de Fourier inverse.

On utilisera les routines de transformées de Fourier discrètes rapide `fft` et sa transformée inverse `ifft` fournies par `scilab` avec l'option `"nonsymmetric"`.

Question 9. Nous allons tester notre solveur de Poisson sur un cas analytique.

Trouver $\alpha \in \mathbb{R}$ tel que $\psi_\alpha(x, y) = \alpha \sin(2\pi x) \sin(2\pi y)$ soit une solution analytique de $\Delta \psi_\alpha(x, y) = \sin(2\pi x) \sin(2\pi y)$. Implémenter la fonction `solution_field` dans le fichier `test_poisson.sce` qui calcule $\psi_\alpha(x, y)$, solution de notre cas test.

Il est possible de tester le solveur avec la commande `make poisson` dans le dossier `src/poisson`. Si tout se passe comme prévu `TEST SUCCESS` est affiché.

Question 10. Compléter la routine `plot_error` du fichier `test_poisson.sce` afin d'afficher les dans une même image les quatre courbes suivantes :

1. $f(x, y) = \sin(2\pi x) \sin(2\pi y)$, la fonction analytique testée.
2. $\Psi_\alpha(x, y)$, la solution analytique de référence.
3. $\Psi_*(x, y)$, la solution obtenue avec le solveur de Poisson FFT.
4. $|\Psi_\alpha - \Psi_*(x, y)|$, l'erreur absolue commise par le solveur.

Inclure la sortie ainsi que vos commentaires sur l'erreur commise dans le rapport. On pourra s'aider des fonctions `subplot`, `plot3d` et `colorbar`.

Question 11. *Le véritable problème à résoudre est l'équation (16d). Une fois écrite sous forme scalaire, on obtient deux problèmes de Poisson indépendants :*

$$\Delta \mathbf{u}_x(x, y) = -\frac{\partial \omega}{\partial y}(x, y) \quad \text{et} \quad \Delta \mathbf{u}_y(x, y) = +\frac{\partial \omega}{\partial x}(x, y)$$

Exprimer les coefficients de Fourier de $\hat{\mathbf{u}}_x$ et $\hat{\mathbf{u}}_y$ en fonction de ceux de $\hat{\omega}$ et de \mathbf{k}_x et \mathbf{k}_y quand cela est possible. Implémenter la routine `poisson_curl_2d` dans le fichier `poisson.sce` qui résout le problème de Poisson avec rotationnel en deux dimensions avec $\hat{\mathbf{u}}_{x,00} = \mathbf{0}$ et $\hat{\mathbf{u}}_{y,00} = \mathbf{0}$.

On validera cette nouvelle routine avec $\omega(x, y) = 8\pi^2 \cos(2\pi x) \cos(2\pi y)$, $\mathbf{u}_x(x, y) = -2\pi \cos(2\pi x) \sin(2\pi y)$ et $\mathbf{u}_y(x, y) = +2\pi \sin(2\pi x) \cos(2\pi y)$ en implémentant un nouveau script de test `test_poisson_curl.sce` similaire à `test_poisson.sce`.

Après implémentation, il sera possible de lancer le script avec la commande `make curl` dans le dossier `src/poisson`. Inclure les graphes de l'erreur absolue commise pour la résolution de \mathbf{u}_x et \mathbf{u}_y dans votre rapport ainsi que vos commentaires éventuels.

3 Simulation numérique

Dans cette partie on propose d'implémenter l'algorithme de résolution (**A**) dans le fichier `src/simu.sce` sur un domaine périodique $\Omega = [0, 1]^2$ jusqu'à $T = 1.50$ avec une viscosité ν sur le problème d'une double couche de cisaillement. Ces couches de cisaillement sont initialement perturbés légèrement ce qui va provoquer un enroulement des couches formant de grandes structures tourbillonnaires.

Les conditions initiales sont les suivantes :

$$\mathbf{u}_x^0(x, y) = \begin{cases} \tanh(\rho[y - 0.25]) & \text{pour } y \leq 0.5 \\ \tanh(\rho[0.75 - y]) & \text{pour } y > 0.5 \end{cases}$$

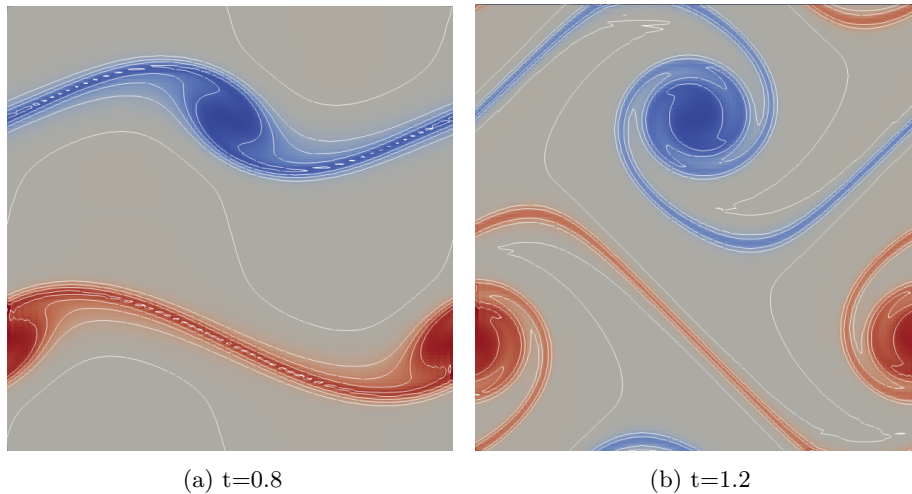
$$\mathbf{u}_y^0(x, y) = \delta \sin(2\pi x)$$

où ρ représente la largeur de la couche initiale et $\delta = 0.05$ la taille de la perturbation. On prendra dans un premier temps $\nu = 10^{-4}$ et $\rho = 30.0$.

Question 12. *Calculer le champ de vorticit   initial $\omega^0(x, y) = \frac{\partial \mathbf{u}_y^0}{\partial x} - \frac{\partial \mathbf{u}_x^0}{\partial y}$. Compl  ter la routine qui initialise ω avec son expression analytique    $t = 0$.*

Question 13. *En utilisant les fonctions impl  ment  es en section 2 et 3, impl  menter l'algorithme de r  solution d  crit en fin de section 1. On prendra un pas d'espace dx constant que l'on choisira uniforme dans chaque direction $N_x = N_y = 128$. On calculera un nouveau pas de temps dt    chaque it  ration apr  s le calcul de la vitesse en fonction de la contrainte de stabilit   du sch  ma de transport-diffusion. Afficher les isocontours de la vorticit   ω    $t = 0.80$ et $t = 1.20$ pour des valeurs allant de -36    $+36$ par pas de 6. Monter en r  solution spatiale si possible et inclure les r  sultats dans le rapport accompagn  s de vos commentaires.*

Les isocontours de r  f  rence    $t = 0.80$ et $t = 1.20$ sont donn  s pour la premi  re configuration ci-dessous. Ils ont   t   obtenus avec un autre solveur, il ne faudra donc pas s'attendre    une correspondance parfaite.

FIGURE 1 – Isocontours de $\omega(x)$ de -36 à 36 par pas de 6.

On pourra lancer la simulation avec `make simu`, créer une video de sa simulation en utilisant `make video` et supprimer tous les fichiers générés avec `make clean`.

Question 14. *Même question avec $\nu = 0.5 \cdot 10^{-4}$ et $\rho = 100.0$. On tracera les isocontours pour des valeurs de -70 à $+70$ par pas de 10. Comparer le nombre d'itérations nécessaires à la résolution du problème par rapport à la configuration précédente. Cette configuration est-elle plus facile à simuler que la première ?*

Question 15. *Pour la configuration 2, visualiser le champ de vecteurs vitesse à $t = 0.8$ et $t = 1.2$ avec la routine `fchamp`. Nous prendrons en compte les efforts effectués pour réaliser une visualisation plus poussée que celle suggérée.*

Annexes

A Installation sur une machine personnelle

Ce TP ne dépend que de `scilab` et `make`. Il faut `ffmpeg` afin de pouvoir générer les vidéos de simulation. Les personnes sur Windows ont deux possibilités :

1. Utiliser le Windows Subsystem for Linux (WSL) sur Windows 10.
2. Installer Docker sur Windows et :
 - soit utiliser l'image nommée `ubuntu` disponible sur `dockerhub` :
<https://cloud.docker.com/u/keckj/repository/docker/keckj/tpmn>
 - soit régénérer l'image localement avec le `Dockerfile` fourni.

Pour `ubuntu bionic 18.04 LTS` les dépendances sont : `make`, `scilab`, `openjdk-8-jdk`, `texlive-full` et `ffmpeg`. Le logiciel `evince` vous permettra d'ouvrir des fichiers pdf et `latexmk` vous permettra de compiler du \LaTeX facilement. Enfin le logiciel libre multi-plateforme `texmaker` vous offrira une interface graphique avec un afficheur pdf intégré pour faire du \LaTeX qui supporte la compilation `latexmk`.

B Construction de la formulation vitesse-vorticité des équations de Navier-Stokes

B.1 Reformulation de la formulation vitesse-pressure

On étudie initialement le modèle en dimension $n = 3$. On se place dans le domaine $\Omega \subset \mathbb{R}^3$ avec $\mathbf{x} = (x, y, z) \in \Omega$ et on définit les champs suivants, que l'on suppose suffisamment réguliers et **qui ne dépendent pas de z** :

$$P(\mathbf{x}, t) = P(x, y, t)$$

$$\mathbf{u}(\mathbf{x}, t) = \begin{bmatrix} u_x(x, y, t) \\ u_y(x, y, t) \\ u_z(x, y, t) \end{bmatrix}$$

$$\nabla \cdot \mathbf{u} = 0 \quad (13a)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \Delta \mathbf{u} - \frac{\nabla P}{\rho} \quad (13b)$$

À noter ici que si $\mathbf{u}_z(\mathbf{x}, 0) = 0 \ \forall \mathbf{x} \in \Omega$ alors $\mathbf{u}_z(\mathbf{x}, t) = 0 \ \forall (\mathbf{x}, t) \in \Omega \times [0, T]$. On se place dans un tel cas de figure et dans toute la suite des calculs, on suppose que $\mathbf{u}_z = 0$.

On définit un nouveau champ vectoriel $\boldsymbol{\omega} = \nabla \wedge \mathbf{u}$, appelé champ de vorticité. Il est défini comme étant le rotationnel de la vitesse du fluide \mathbf{u} . Dans notre cas, la vorticité est en réalité un champ scalaire ($\omega_x = \omega_y = 0$) et $\omega_z = \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}$.

On souhaite se débarrasser de la variable de pression P . En considérant $\mathbf{u} = [u_x \ u_y \ 0]^T$ et $\boldsymbol{\omega} = \nabla \wedge \mathbf{u} = [0 \ 0 \ \omega_z]^T$, on peut montrer assez facilement, en échangeant l'ordre des dérivées que l'on a les égalités suivantes :

$$\nabla \wedge \nabla P = \mathbf{0} \quad (14a)$$

$$\nabla \wedge \frac{\partial \mathbf{u}}{\partial t} = \frac{\partial \boldsymbol{\omega}}{\partial t} \quad (14b)$$

$$\nabla \wedge \Delta \mathbf{u} = \Delta \boldsymbol{\omega} \quad (14c)$$

En remarquant finalement que $\nabla \wedge [(\mathbf{u} \cdot \nabla) \mathbf{u}] = (\mathbf{u} \cdot \nabla) \boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \nabla) \mathbf{u}$, on peut montrer que la vorticité $\boldsymbol{\omega}$ vérifie l'équation d'advection-diffusion suivante :

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla) \boldsymbol{\omega} = \nu \Delta \boldsymbol{\omega} \quad (15)$$

On déduit de ce qui précède la formulation vitesse-vorticité $(\mathbf{u}, \boldsymbol{\omega})$ des équations de Navier-Stokes :

$$\nabla \cdot \mathbf{u} = 0 \quad (16a)$$

$$\boldsymbol{\omega} = \nabla \wedge \mathbf{u} \quad (16b)$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla) \boldsymbol{\omega} = \nu \Delta \boldsymbol{\omega} \quad (16c)$$

Il reste maintenant à calculer la vortacité ω à partir de la vitesse \mathbf{u} . Les équations (16a) et (16b) impliquent :

$$\Delta \mathbf{u} = - \nabla \wedge \omega \quad (16d)$$

Le cas 2D correspond précisément au cas où $\mathbf{u}_z = 0$ et où les champs ne dépendent pas de z . En notant $\omega = \omega_z$ on peut donc déduire la formulation vitesse-vortacité (\mathbf{u}, ω) des équations de Navier-Stokes en dimension deux :

$$\frac{\partial \mathbf{u}_x}{\partial x} + \frac{\partial \mathbf{u}_y}{\partial y} = 0 \quad (17a)$$

$$\omega = \frac{\partial \mathbf{u}_y}{\partial x} - \frac{\partial \mathbf{u}_x}{\partial y} \quad (17b)$$

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = \nu \Delta \omega \quad (17c)$$

B.2 Compléments d'analyse vectorielle

On définit les opérateurs nabla ∇ et laplacien Δ en dimension 3 :

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} \quad \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Soit $f : \Omega \rightarrow \mathbb{R}$ un champ scalaire et $\mathbf{v} : \Omega \rightarrow \mathbb{R}^3$ un champ vectoriel.

Par abus de notation, on pourra utiliser ∇ comme un vecteur classique :

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \quad \begin{aligned} \nabla \cdot \mathbf{v} &= \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \\ \mathbf{v} \cdot \nabla &= v_x \frac{\partial}{\partial x} + v_y \frac{\partial}{\partial y} + v_z \frac{\partial}{\partial z} \end{aligned} \quad \nabla \wedge \mathbf{v} = \begin{bmatrix} \frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} \\ \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x} \\ \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \end{bmatrix}$$

On pourra également utiliser Δ comme un scalaire :

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \quad \Delta \mathbf{v} = \begin{bmatrix} \frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2} \\ \frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} + \frac{\partial^2 v_y}{\partial z^2} \\ \frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} + \frac{\partial^2 v_z}{\partial z^2} \end{bmatrix}$$

Ces deux opérateurs se généralisent en dimension n quelconque avec les

expressions suivantes : $\nabla = \left(\frac{\partial}{\partial x_i} \right)_{i \in \llbracket 0, n-1 \rrbracket}$ et $\Delta = \nabla \cdot \nabla = \sum_{i=0}^{n-1} \frac{\partial^2}{\partial x_i^2}$.

B.3 Complément concernant le modèle de Navier-Stokes

Cette annexe fournit des sources complémentaires sur le modèle de Navier-Stokes incompressible. Pour plus de détails sur le modèle on pourra par exemple consulter cet article de vulgarisation scientifique : <https://sciencetonnante.wordpress.com/2014/03/03/la-mysterieuse-equation-de-navier-stokes>.

The diagram illustrates the Navier-Stokes equation with physical interpretations for each term:

$$\rho \left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} \right) = \nabla P + \rho \mathbf{g} + \mu \nabla^2 \mathbf{V}$$

- MASS** (Density of the fluid): ρ
- ACCELERATION** (How velocity experienced by a particle changes with time): $\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V}$
 - $\frac{\partial \mathbf{V}}{\partial t}$: Change in velocity over time
 - $\mathbf{V} \cdot \nabla \mathbf{V}$: The speed and direction which the fluid is moving
- FORCE** (All the forces that are acting on the fluid): $\nabla P + \rho \mathbf{g} + \mu \nabla^2 \mathbf{V}$
 - ∇P : Internal pressure gradient of the fluid (the change in pressure)
 - $\rho \mathbf{g}$: External forces acting on the fluid (such as gravity)
 - $\mu \nabla^2 \mathbf{V}$: Internal stress forces acting on the fluid (taking into consideration viscous effects)

La simulation que l'on résout en section 4 est tirée du [papier suivant](#) : *David L. Brown et Michael L. Minion, Performance of Underresolved Two Dimensional Incompressible Flow Simulations, Journal of Computational Physics, 1995, vol. 122, no 1, p. 165-183.*