

IN103 : Introduction à Matlab

Instabilité de Kelvin Helmholtz

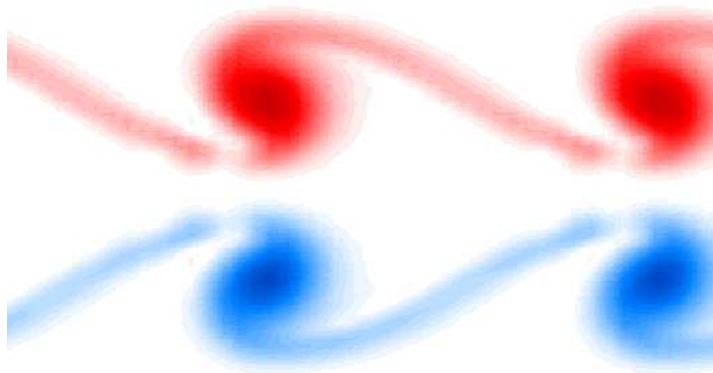


Figure : Instabilité de Kelvin Helmholtz

# Chapitre 1

## Introduction

### 1.1 Instabilité de Kelvin-Helmholtz

Les instabilités de Kelvin-Helmholtz apparaissent dans les fluides qui présentent un fort cisaillement entre deux couches (de densités différentes par exemple) de fluide. Il résulte alors un enroulement d'une couche avec l'autre. La compréhension de la génération de ces instabilités permet de mieux les représenter dans les modèles dont la résolution n'est pas suffisante pour permettre de les résoudre explicitement.

Dans cette étude, nous allons modéliser le phénomène physique en question pour la résolution numérique des équations de Navier Stokes qui régissent la dynamique des fluides incompressibles. Pour rendre le problème plus abordable, nous nous placerons dans le cadre simplifié des équations bidimensionnelles (2D), avec des conditions aux limites périodiques. Ce modèle va nous permettre de simuler les instabilités de Kelvin-Helmholtz d'une couche de mélange plane.

Sur le plan pratique, nous aborderons rapidement la discrétisation des équations différentielles par la méthode dite « aux différences finies », ainsi que plusieurs schémas numériques utiles. On verra la nécessité de construire un maillage adapté. Enfin on s'intéressera à la résolution d'un système tridiagonal, à l'utilisation de la transformée de Fourier rapide (FFT), ainsi qu'à l'utilisation d'options graphiques afin de visualiser les résultats.



FIG. 1.1 – Instabilités de Kelvin-Helmholtz

## 1.2 Matlab

Au fur et à mesure nous allons voir comment traduire en langage Matlab le problème physique. En général la construction d'un programme contient un script principal (fonction *main*) qui contient les variables de votre programme et appellera quand nécessaire les fonctions que vous aurez créées pour répondre à des questions spécifiques.

Pour vous aider il est indispensable d'utiliser l'aide de Matlab. Vous disposez également de documents en ligne sur le site internet de l'ENSTA.

Enfin pensez à écrire des commentaires dans vos scripts Matlab, cela vous permettra de suivre plus facilement votre propre progression dans l'écriture du programme.

# Chapitre 2

## Le problème physique

### 2.1 Définition du problème

On considère l'écoulement d'un fluide dans le plan horizontal  $(x,y)$ . On suppose que le fluide est incompressible :  $\rho(x,y,t) = \rho_0$ . L'écoulement du fluide est alors complètement décrit par son champ vectoriel de vitesse  $\mathbf{U}(x,y) = (u(x,y), v(x,y))$  et son champ scalaire de pression  $p(x,y)$ . On va déterminer ces grandeurs à partir des lois de conservation suivantes :

- la conservation de la masse (qui se réduit à la non-divergence de la vitesse) :

$$\nabla \cdot (\mathbf{U}) = 0 \quad (2.1)$$

- la conservation de la quantité de mouvement :

$$\rho_0 \left( \frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \cdot \nabla) \mathbf{U} \right) = -\nabla p + \mu \Delta \mathbf{U} \quad (2.2)$$

Afin de pouvoir représenter des phénomènes physiques sans tenir compte des dimensions du problème, on *adimensionne* les équations. On introduit donc les paramètres adimensionnés suivant :

$$x = \frac{x^*}{L}, \dots, u = \frac{u^*}{V_0}, \dots, t = \frac{V_0 t^*}{L}, p = \frac{p^*}{\rho V_0^2} \quad (2.3)$$

Les variables (\*) sont les grandeurs physiques.  $L, V_0$  sont les échelles de longueur et de vitesse caractéristique de l'écoulement. Un nombre sans dimension utile pour caractériser le comportement des fluides incompressibles est le nombre de Reynolds qui représente le rapport des effets d'inertie sur les effets de viscosité :

$$Re = \frac{V_0 L}{\nu}, \nu \text{ viscosité cinématique du fluide.} \quad (2.4)$$

On peut réécrire explicitement le système d'équations adimensionnées à résoudre :

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (2.5)$$

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (2.6)$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (2.7)$$

En conclusion, le système d'équations de Navier-Stokes à résoudre est constitué par le système d'équations (2.5) - (2.6) - (2.7). La condition initiale et les conditions limites seront précisées plus tard.

#### **Application :**

- Sous quelle forme exprimer les champs  $u, v$  et  $p$  sur le plan  $(x,y)$  du point de vue Matlab ?

## 2.2 Domaine et Maillage

Avant de s'attaquer aux équations et à leur résolution à proprement parler, on doit définir un domaine d'étude. Nous allons donc construire le maillage qui va servir à discrétiser les équations. Le domaine de calcul est rectangulaire de dimension  $L_x \times L_y$  (voir fig.2.1). Quant aux conditions de bords, on introduit des conditions de périodicité qui simplifient grandement la résolution des équations. De plus ces conditions périodiques vont nous permettre d'utiliser une fonctionnalité de Matlab : le calcul matricielle. Les boucles du type "for i=1 :n .... end" seront donc à proscrire le plus possible. A leur place il faudra de préférence utiliser le calcul matriciel. On considère donc que les champs  $\mathbf{U}(x, y)$  et  $p(x, y)$  sont périodiques en x et y :

$$\mathbf{U}(0, y) = \mathbf{U}(L_x, y) \quad p(0, y) = p(L_x, y) \quad \forall y, \quad (2.8)$$

$$\mathbf{U}(x, 0) = \mathbf{U}(x, L_y) \quad p(x, 0) = p(x, L_y) \quad \forall x, \quad (2.9)$$

En tenant compte des conditions périodiques, les équations seront donc discrétisées sur un maillage comportant  $n_x - 1$  points de discrétisation suivant  $x$  et  $n_y - 1$  points de discrétisation suivant  $y$ . Afin d'assurer la stabilité de la méthode numérique utilisée, on utilise un *maillage décalé*.

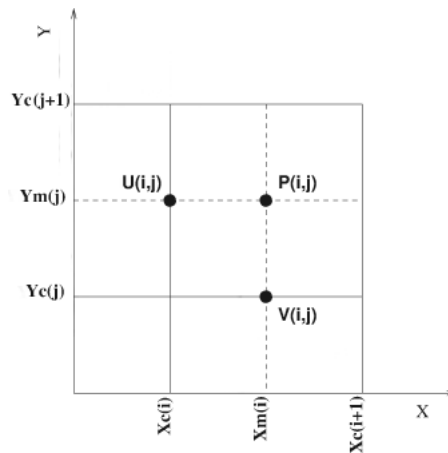


FIG. 2.1 – Maillage du domaine.

– Maillage *primaire* :

$$x_c(i) = (i - 1) \delta x, \quad \delta x = \frac{L_x}{n_x - 1}, \quad i = 1..n_x - 1, \quad (2.10)$$

$$y_c(i) = (j - 1) \delta y, \quad \delta y = \frac{L_y}{n_y - 1}, \quad j = 1..n_y - 1, \quad (2.11)$$

– Maillage *secondaire* :

$$x_m(i) = (i - \frac{1}{2}) \delta x, \quad i = 1..n_x - 1, \quad (2.12)$$

$$y_m(i) = (j - \frac{1}{2}) \delta y, \quad j = 1..n_y - 1, \quad (2.13)$$

Dans la maille  $(i, j)$ , c'est-à-dire le rectangle  $[x_c(i), x_c(i + 1)] \times [y_c(j), y_c(j + 1)]$ , les inconnues  $u, v, p$  seront évaluées en des points distincts :

- $u(i, j)$  est défini au point de coordonnées  $(x_c(i), y_m(j))$ ,
- $v(i, j)$  est défini au point de coordonnées  $(x_m(i), y_c(j))$ ,
- et  $p(i, j)$  est défini au point de coordonnées  $(x_m(i), y_m(j))$ .

La taille des tableaux (u,v,p,...) sera donc  $[n_x - 1, n_y - 1]$ .

Lors de la programmation, vous pourrez aussi définir certaines variables en tant que variables *globales*. Cf aide Matlab.

On considère un temps  $T$ , durée totale de la simulation, ainsi que le pas de temps  $\delta t$  pour le passage d'un état donné du fluide à un autre état. On définit l'instant :  $t_n = n\delta t$ . Le nombre maximal d'itérations est alors  $n_{max} = T/\delta t$ .

**Application :**

- Créer un script principal (main) dans lequel vous allez définir les différentes variables.
- Définir dans Matlab  $Rey$  (nombre de Reynolds),  $n_x, n_y, L_x, L_y, \delta x, \delta y, n_{max}$ .
- Créer dans Matlab les vecteurs  $i = [1 : n_x - 1]$  et  $j = [1 : n_y - 1]$ , quelle taille a la matrice  $u$ ?
- Définir et initialiser les matrices  $u, v$ , et  $p$
- Créer dans Matlab les vecteurs  $x_c, y_c, x_m$  et  $y_m$ .

### 2.3 Méthode aux différences finies

Compte tenu de la forme des équations, on ne peut résoudre analytiquement le système (2.5) - (2.6) - (2.7). On utilise donc le calcul numérique. Dans un premier temps on exprime de façon discrète les grandeurs physiques concernées par notre problème.

**Expression d'une grandeur continue sous forme discrète :** Soit une fonction  $f$  dépendant de la variable  $x$ . Il nous faut représenter les fluctuations de  $f$  de manière à retenir son comportement global. Pour cela, on ne conserve que certaines valeurs de cette fonction. On réalise en fait un "échantillonnage". D'où :

$$\underbrace{f(x) \rightsquigarrow f(i \cdot \delta x)}_{\text{discrétisation}} \rightsquigarrow \underbrace{f(i) \text{ ou } f_i}_{\text{notation}} \text{ avec } i \text{ un entier.}$$

Si bien qu'une fonction  $f = f(x, y, t)$  pourra être formulée sous forme discrète par :

$$f(x,y,t) \rightsquigarrow f(i \cdot \delta x, j \cdot \delta y, n \cdot \delta t) \rightsquigarrow f_{i,j}^n$$

**Schéma aux différences finies :** En utilisant le développement de Taylor on peut exprimer les dérivés d'une fonction  $f$ . Pour voir comment ces résultats sont obtenus vous pouvez vous reporter à l'annexe A. On a alors :

Dérivée d'ordre 1 :

- Schéma Euler arrière :

$$f'(i) = \frac{f(i) - f(i - 1)}{\delta x} \tag{2.14}$$

- Schéma Euler avant :

$$f'(i) = \frac{f(i + 1) - f(i)}{\delta x} \tag{2.15}$$

Dérivée d'ordre 2 :

– Schéma Euler centré :

$$f''(i) = \frac{f(i+1) - 2f(i) + f(i-1)}{\delta x^2} \quad (2.16)$$

**Application :**

- Pour prendre en compte les conditions de périodicité (définies précédemment), définir les vecteurs  $i_m = [n_x - 1, 1 : n_x - 2]$ ,  $j_m = [n_y - 1, 1 : n_y - 2]$ ,  $i_p = [2 : n_x - 1, 1]$ ,  $j_p = [2 : n_x - 1, 1]$ .
- En vous référant aux schémas ci-dessus et à l'annexe (A.1), exprimer  $\frac{\partial u}{\partial x}$ ,  $\frac{\partial u}{\partial y}$  sous forme discrète avec un schéma Euler avant et/ou arrière, et  $\frac{\partial^2 u}{\partial x^2}$  avec un schéma Euler centré.

# Chapitre 3

## Résolution numérique

### 3.1 Algorithme de résolution

Afin d'observer, numériquement, l'évolution spatio-temporelle du fluide, le système d'équations de Navier-Stokes sera résolu par une *méthode de projection* comportant deux pas :

- Un pas de prédiction : l'équation (2.2) sera résolue suivant un schéma explicite d'Adams-Bashfort (pour les termes convectifs  $(\mathbf{U} \cdot \nabla) \cdot \mathbf{U}$ ) et un schéma de Crank-Nicolson (pour les termes diffusifs  $\Delta \mathbf{U}$ ) :

$$\frac{\mathbf{U}^* - \mathbf{U}^n}{\delta t} = -\nabla p^n + \underbrace{\frac{3}{2}\mathbf{H}^n - \frac{1}{2}\mathbf{H}^{n-1}}_{\text{Adams-Bashfort}} + \underbrace{\frac{1}{Re}\Delta\left(\frac{\mathbf{U}^* + \mathbf{U}^n}{2}\right)}_{\text{Crank-Nicolson}} \quad (3.1)$$

- Un pas de correction : en effet le champ  $\mathbf{U}^*$  n'est pas de divergence nulle, on applique une correction afin d'assurer cette condition pour le champ  $\mathbf{U}^{n+1}$  :

$$\mathbf{U}^{n+1} - \mathbf{U}^* = -\delta t \nabla \phi \quad (3.2)$$

La variable  $\phi$  (sans signification physique) sera déterminée en prenant la divergence de l'équation (3.2), ce qui donne :

$$\Delta \phi = \frac{1}{\delta t} \nabla \cdot \mathbf{U}^* \quad (3.3)$$

Pour finir, l'algorithme est bouclé en réactualisant la pression (ainsi que son gradient) pour le pas suivant par :

$$p^{n+1} = p^n + \phi - \frac{\delta t}{2 Re} \Delta \phi \quad (3.4)$$

Cette progression permet donc le passage de l'état à l'instant  $t_n$  à l'état à l'instant  $t_{n+1}$ .

#### Application :

- Ecrire, de manière la plus détaillée possible, les étapes nécessaires pour passer de l'état à l'instant  $t_n$  à l'état à l'instant  $t_{n+1}$  à l'aide des quatre équations ci-dessus, c'est-à-dire comment passer de  $\mathbf{U}^n, p^n$  à  $\mathbf{U}^{n+1}, p^{n+1}$ .

## 3.2 Pré-requis

### 3.2.1 Conditions initiales

Afin d'observer numériquement ce problème, nous allons imposer un cisaillement dans le champ fluide en introduisant un profil de vitesse avec un point d'inflexion. Seule la vitesse initiale  $u_0$  sera non-nulle ( $v_0 = 0$  et  $p_0 = 0$ ) tel que présentée sur la figure suivante.

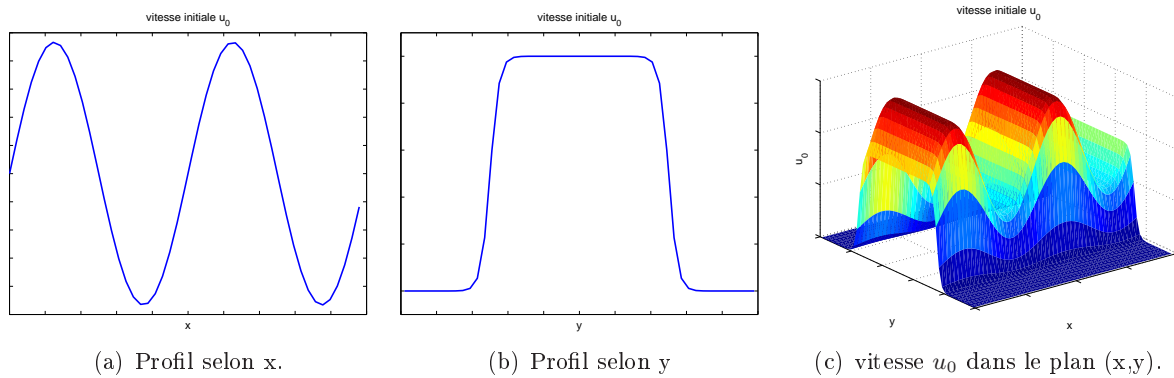


FIG. 3.1 – Vitesse initiale  $u_0$ .

#### **Application :**

- Ecrire une fonction matlab qui définit les conditions initiales ( $u_0$ ) en fonction des paramètres du domaine ( $x_c, y_m, L_x, L_y$ ). Vous pourrez utiliser la fonction matlab pré-existante *tanh* pour obtenir un champ initial  $u_0$  semblable à la figure (3.1). La fonction *ndgrid* pourra vous être utile pour les coordonnées 2D.

### 3.2.2 Calcul du pas de temps

Pour que notre modélisation du problème soit complète, il nous faut indiquer comment calculer le pas de temps  $\delta t$  par lequel sera incrémenté le schéma itératif. La valeur du pas de temps sera limitée par une condition de stabilité dite CFL calculée à partir de la relation générale :

$$\delta t = \frac{cfl}{\max\left(\left|\frac{u_0}{\delta x}\right| + \left|\frac{v_0}{\delta y}\right|\right)} \quad (3.5)$$

On impose par ailleurs  $cfl < 1$  (par exemple  $cfl = 0.2$ ).

#### **Application :**

- Ecrire une fonction matlab qui calcule  $\delta t$ .

### 3.2.3 Boucle temporelle

Avant de coder les équations pour résoudre le problème physique il s'agit de définir la boucle temporelle dans laquelle ces équations vont être résolues. En effet nous avons précédemment défini les conditions initiales. Nous allons maintenant pour chaque pas de temps résoudre les équations à partir des valeurs calculées au pas de temps précédent.

**Application :**

- Ecrire une boucle temporelle sur t dans le main, après les définitions et initialisations des variables.

### 3.3 Calcul du champ $\mathbf{U}^*$

#### 3.3.1 Calcul des termes explicites de convection

Les deux composantes du terme de convection  $\mathbf{H}=[H_u, H_v]$  seront calculées aux mêmes point de grille que les vitesses correspondantes. Ainsi  $H_u(i, j)$  est défini aux points  $(x_c(i), y_m(j))$ . Pour se faire, nous utiliserons un schéma aux différences finies tel que chaque dérivée représente le même poids dans le calcul du terme  $\mathbf{H}$ .

$$H_u = -\left(\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y}\right) \quad (3.6)$$

$$H_v = -\left(\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y}\right) \quad (3.7)$$

**Application :**

- En vous référant à l'annexe (A.2), écrire une fonction Matlab qui calcule  $H_u$  et  $H_v$ .

#### 3.3.2 Calcul du champ $\mathbf{U}^*$

On peut premièrement remarquer que l'équation (3.1) peut s'écrire sous la forme suivante :

$$\underbrace{\left(I - \frac{\delta t}{2 Re} \Delta\right)}_{\text{opérateur de Helmholtz}} \delta \mathbf{U}^* = \delta t \underbrace{\left[-\nabla p^n + \frac{3}{2} H^n - \frac{1}{2} H^{n-1} + \frac{1}{Re} \Delta \mathbf{U}^n\right]}_{RHS^n} \quad (3.8)$$

où  $\delta \mathbf{U}^* = \mathbf{U}^* - \mathbf{U}^n$ . Pour résoudre cette équation nous allons utiliser la *factorisation approximative* (ou ADI - Alternating Direction Implicit en anglais). L'opérateur de Helmholtz est approché, avec une précision à l'ordre deux en espace, sous la forme :

$$\left(I - \frac{\delta t}{2 Re} \Delta\right) \approx \left(I - \frac{\delta t}{2 Re} \frac{\partial^2}{\partial x^2}\right) \left(I - \frac{\delta t}{2 Re} \frac{\partial^2}{\partial y^2}\right) \quad (3.9)$$

Cette factorisation est utilisée pour résoudre l'équation (3.8) en deux étapes :

$$\underbrace{\left(I - \frac{\delta t}{2 Re} \frac{\partial^2}{\partial x^2}\right)}_{A1} \overline{\delta \mathbf{U}^*} = RHS^n \quad (+ \text{condition de périodicité en } x) \quad (3.10)$$

$$\underbrace{\left(I - \frac{\delta t}{2 Re} \frac{\partial^2}{\partial y^2}\right)}_{A2} \delta \mathbf{U}^* = \overline{\delta \mathbf{U}^*} \quad (+ \text{condition de périodicité en } y) \quad (3.11)$$

Où  $\overline{\delta \mathbf{U}^*}$  est une variable intermédiaire introduite afin d'explicitier la démarche.

En discrétisant, ces systèmes sont de la forme :

$$\alpha f_{i-1} + \beta f_i + \gamma f_{i+1} = g_i \text{ (+ conditions de périodicité)} \quad (3.12)$$

Si on note  $(A_{ij})$  la matrice des coefficients  $(\alpha, \beta, \gamma)_i$ , on peut écrire le système suivant :

$$\begin{bmatrix} \beta & \gamma & & & & & & & \alpha \\ \alpha & \ddots & \ddots & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \alpha & \beta & \gamma & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \gamma & \\ \gamma & & & & & & \alpha & \beta & \end{bmatrix} \begin{bmatrix} \vdots \\ f_{i-1} \\ f_i \\ f_{i+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ g_i \\ \vdots \end{bmatrix}$$

Ces système linéaires à matrice tridiagonale et périodique pourront être résolus à l'aide la fonction *chol*.

#### Application :

- Calculer  $RHS_n$  à partir de  $H_u$  et  $H_v$ , du gradient de  $p$  (utiliser un schéma Euler arrière), et du Laplacien de  $\mathbf{U}$  (utiliser le schéma d'Euler d'ordre 2).
- En discrétisant (3.10) et (3.11) montrer qu'on obtient effectivement dans chacun des deux cas un système de la forme (3.12).
- Que valent alors les coefficients  $\alpha$ ,  $\beta$  et  $\gamma$  pour les matrices A1 et A2 ? Définir les matrices A1 et A2 avant la boucle temporelle.
- Calculer  $\overline{\delta\mathbf{U}^*}$  en résolvant le système linéaire  $A1\overline{\delta\mathbf{U}^*} = RHS^n$
- Calculer  $\delta\mathbf{U}^*$  en résolvant le système linéaire  $A2\delta\mathbf{U}^* = \overline{\delta\mathbf{U}^*}$
- Pour ceux qui ont du temps, résoudre le système avec la méthode Choleski. Pour cela, écrire une fonction matlab, prenant en entrée une matrice A et un vecteur x, qui résout un système du type (3.12) à l'aide de la fonction matlab préexistante *chol*. Modifier cette fonction afin de pouvoir résoudre un système du type (3.10).
- Calculer enfin  $\mathbf{U}^*$  à partir de la définition de  $\delta\mathbf{U}^*$ .

### 3.4 Résolution de l'équation de Poisson

Nous nous intéressons maintenant à l'équation (3.3) :

$$\Delta\phi = \underbrace{\frac{1}{\delta t} \nabla \cdot \mathbf{U}^*}_{Q(i,j)}$$

A ce stade, on connaît  $\mathbf{U}^*$ . On peut donc estimer sa divergence. *La divergence de  $\mathbf{U}^*$  est calculée avec un schéma Euler avant.* De plus, compte tenu de la périodicité de  $u$ ,  $\phi$  est aussi périodique. On développe, dans un premier temps, la dérivée seconde en "x" du laplacien de  $\phi$  par un schéma aux différences finies ( cf. Annexe.(A.1) ), puis on décompose en série de Fourier (en utilisant la périodicité suivant la direction x) :

$$\phi(i, j) = \sum_{l=1}^{n_x-1} \hat{\phi}_l(j) e^{i \frac{2\pi}{n_x-1} (i-1)(l-1)}, \quad \forall i = 1 \dots n_x - 1 \quad (3.13)$$

L'utilité de cette décomposition est de diagonaliser l'opérateur laplacien et de ramener le problème initial (3.3) à un problème 1D.

On a en fait de même avec le terme  $Q$  qui lui aussi étant périodique (puisque  $U^*$  est périodique) se décompose en série de Fourier :

$$Q(i, j) = \sum_{l=1}^{n_x-1} \hat{Q}_l(j) e^{i \frac{2\pi}{n_x-1} (i-1)(l-1)}, \forall i = 1 \dots n_x - 1 \quad (3.14)$$

Si bien que l'équation initiale (3.3) est équivalente, pour chaque nombre d'onde  $l = 1 \dots n_x - 1$ , à une équation de la forme :

$$\frac{\partial^2 \hat{\phi}_l(j)}{\partial y^2} + k_l \hat{\phi}_l(j) = \hat{Q}_l(j), \text{ avec } k_l = \frac{2}{\delta x^2} \left[ \cos\left(\frac{2\pi}{n_x-1}(l-1)\right) - 1 \right] \quad (3.15)$$

A partir de ce point, nous discrétisons directement la dérivée seconde suivant "y" par un schéma aux différences finies centrées. On obtient alors de nouveau un système linéaire à matrice tridiagonale. Cependant ces systèmes ne sont pas positifs, on ne peut donc utiliser l'algorithme de Choleski. On utilisera un algorithme de Thomas. Le détail vous est fourni en annexe. Les fonctions matlab correspondantes vous seront fournies durant les séances.

Une fois  $\hat{\phi}$  résolu dans l'espace de Fourier, on revient dans l'espace physique pour obtenir  $\phi$ .

**Application :**

- Calculer la divergence de  $U^*$  et en déduire  $Q$ .
- Exprimer l'équation (3.15) sous la forme (3.12) en discrétisant. En déduire  $\alpha$ ,  $\beta$  et  $\gamma$ .
- Utiliser  $\alpha$ ,  $\beta$  et  $\gamma$  comme arguments d'entrées des fonctions fournies résolvant l'algorithme de Thomas.

### 3.5 Calcul des champs de vitesse corrigés

Une fois les champs  $U^*$  et  $\phi$  calculés, on corrige les termes de vitesse en utilisant (3.2).

**Application :**

- Ecrire en matlab le calcul de  $U^{n+1}$  à partir de (3.2)

### 3.6 Calcul du champ de pression

Le champ de pression s'obtient directement avec (3.4).

**Application :**

- Ecrire en matlab le calcul du champ de pression à partir de (3.4). *Le gradient de pression sera calculé avec un schéma Euler arrière.*

# Chapitre 4

## Visualisation

Afin d'observer la formation de l'instabilité de Kelvin-Helmholtz, on visualise la vorticité.

### 4.1 Visualisation de la vorticité

Nous disposons de plusieurs moyens rapides de visualisation de l'écoulement. Dans cette étude nous allons calculer le champ de vorticité d'après la définition suivante :

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \quad (4.1)$$

#### Application :

- Ecrire une fonction matlab qui affiche  $\omega$  dans une fenêtre graphique tous les  $N$  pas de temps.

### 4.2 Interface graphique

Pour aller plus loin, et par exemple faciliter l'utilisation de votre programme par une autre personne, il peut être intéressant de créer une mini interface graphique pour la visualisation. Vous prendrez exemple sur le TD n°3. Pensez aussi à consulter l'aide Matlab : section "gui".

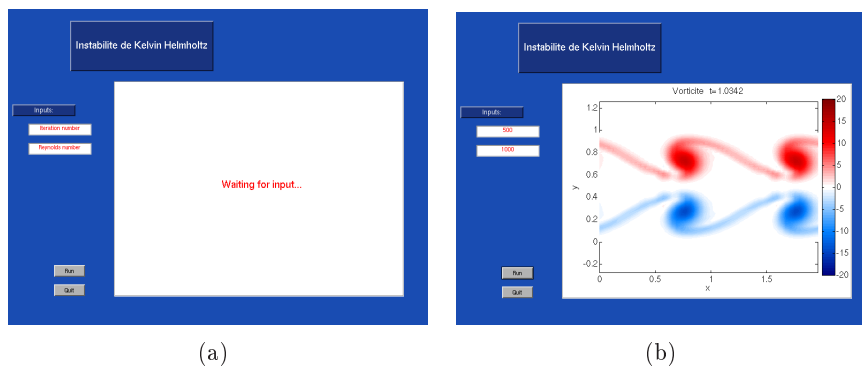


FIG. 4.1 – Exemple d'interface graphique

# Annexe A

## A.1 Méthodes aux différences finies

**Introduction :** Le développement en série de Taylor s'écrit pour une fonction  $f$  continue sur l'intervalle  $[a, b]$  :  $\forall (x, x + \Delta x) \in [a, b] \times [a, b]$ ,

$$f(x + \Delta x) = f(x) + \sum_{k=1}^{\infty} \frac{(\Delta x)^k}{k!} \frac{d^k f}{dx^k}(x)$$

On utilise cette formule dans sa forme discrète pour exprimer de manière approchée les dérivées intervenants dans les équations. Si bien que :

$$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \dots$$

s'écrit en "sous forme discrète" :

$$f(i\Delta x + \Delta x) = f(i\Delta x) + \Delta x f'(i\Delta x) + \frac{\Delta x^2}{2!} f''(i\Delta x) + \dots$$

et enfin du point de vue programmation :

$$f(i + 1) = f(i) + \Delta x f'(i) + \frac{\Delta x^2}{2!} f''(i) + \dots$$

On peut alors définir de manière approchée (en fait à un ordre près)  $f'(i)$ ,  $f''(i)$ , etc...

**Schéma aux différences finies :**

Dérivée d'ordre 1 :

– Euler arrière :

$$f'(i) = \frac{f(i) - f(i - 1)}{\Delta x} \tag{A.1}$$

– Euler avant :

$$f'(i) = \frac{f(i + 1) - f(i)}{\Delta x} \tag{A.2}$$

Dérivée d'ordre 2 :

– Euler centrée :

$$f''(i) = \frac{f(i + 1) - 2f(i) + f(i - 1)}{\Delta x^2} \tag{A.3}$$

## A.2 Calcul des termes d'advection

Nous avons choisi d'utiliser deux maillages entrelacés (un maillage primaire et un maillage secondaire) dans le but de pouvoir évaluer simplement les termes non-linéaires sans compromettre numériquement le calcul des équations de Navier-Stokes. L'écriture des termes  $Hu$  et  $Hv$  n'est donc pas trivial du fait de ce double maillage.

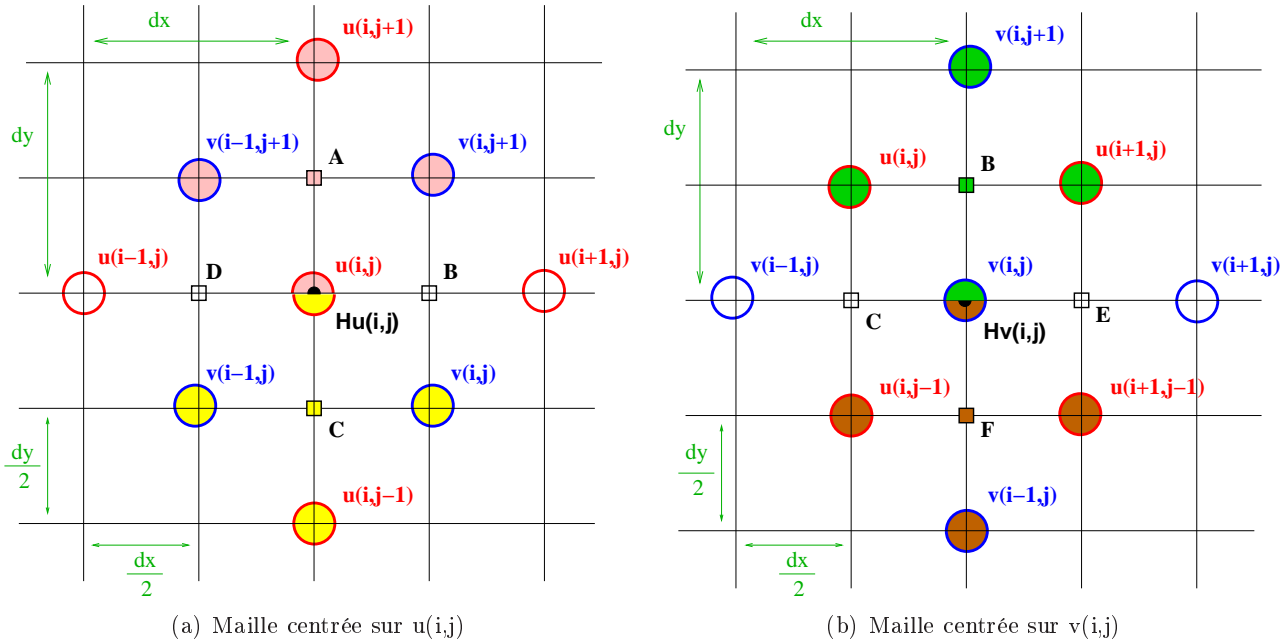


FIG. A.1 – Maillages primaire et secondaire.

### Calcul de $Hu$ :

A partir de la figure (A.1(a)), on calcule le terme  $\frac{\partial uv}{\partial y}$ .

- Evaluer le produit discrétisé " $u*v$ " en A à l'aide des quatre points roses. Pour cela calculer la moyenne de  $u$  et celle de  $v$  en A.
- Evaluer le produit discrétisé " $u*v$ " en C à l'aide des quatre points jaunes de nouveau avec la moyenne.
- Calculer la dérivée discrétisée centrée de " $u*v$ " à partir des résultats précédemment obtenus.

Puis, faites de même pour  $\frac{\partial u^2}{\partial x}$  aux points B et D (*indice* :  $u^2 = u * u$ ).

Créer une fonction Matlab qui calcule  $Hu$ .

### Calcul de $Hv$ :

A partir de la figure (A.1(b)), on calcule le terme  $\frac{\partial uv}{\partial x}$  :

- Evaluer le produit discrétisé " $u*v$ " en C à l'aide de la moyenne.
- Evaluer le produit discrétisé " $u*v$ " en E à l'aide de la moyenne.
- Calculer la dérivée discrétisée centrée de " $u*v$ " à partir des résultats précédemment obtenus.

Puis, faites de même pour  $\frac{\partial v^2}{\partial y}$  aux points B et F.

Créer une fonction Matlab qui calcule  $Hv$ .

### A.3 Algorithme de Thomas

**Algorithme 12.5.** Algorithme de Thomas pour la résolution d'un système linéaire à matrice tridiagonale<sup>5</sup>.

Le système :

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdot & \cdot & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdot & a_n & b_n \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ X_{n-1} \\ X_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_{n-1} \\ f_n \end{pmatrix}$$

est résolu en introduisant la récurrence :

$$\begin{cases} X_k = \gamma_k - \frac{c_k}{\beta_k} X_{k+1}, & k = 1 \dots (n-1) \\ X_n = \gamma_n \end{cases} \quad (12.71)$$

En utilisant ces relations dans le système initial, on peut déterminer les coefficients  $\gamma_k$  et  $\beta_k$  :

$$\begin{cases} \beta_1 = b_1 \\ \beta_k = b_k - \frac{c_{k-1}}{\beta_{k-1}} a_k, & k = 2 \dots n \\ \gamma_1 = \frac{f_1}{\beta_1} = \frac{f_1}{b_1} \\ \gamma_k = \frac{f_k - a_k \gamma_{k-1}}{\beta_k}, & k = 2 \dots n \end{cases}$$

Une fois les coefficients  $\gamma_k$  et  $\beta_k$  calculés, les inconnues  $X_k$  seront calculées par substitution inverse en (12.71), à partir de la valeur de  $X_n$ .

**Algorithme 12.6.** Algorithme de Thomas pour la résolution d'un système linéaire à matrice tridiagonale et périodique.

Le système

$$\left( \begin{array}{cccccc|c} b_1 & c_1 & 0 & \cdot & 0 & 0 & a_1 \\ a_2 & b_2 & c_2 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & \cdot & a_{n-1} & b_{n-1} & c_{n-1} \\ \hline c_n & 0 & 0 & \cdot & 0 & a_n & b_n \end{array} \right) \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ X_{n-1} \\ X_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_{n-1} \\ f_n \end{pmatrix}$$

est écrit sous la forme<sup>B</sup> :

$$\left( \begin{array}{cccccc|c|c} b_1^* & c_1 & 0 & \cdot & 0 & 0 & 0 & v_1 \\ a_2 & b_2 & c_2 & \cdot & 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & \cdot & a_{n-1} & b_{n-1} & c_{n-1} & 0 \\ 0 & 0 & 0 & \cdot & 0 & a_n & b_n^* & v_n \\ \hline -1 & 0 & 0 & \cdot & 0 & 0 & -1 & 1 \end{array} \right) \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ X_{n-1} \\ X_n \\ X^* \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_{n-1} \\ f_n \\ 0 \end{pmatrix} \quad (12.72)$$

avec :

$$\begin{cases} v_1 = a_1 \\ v_n = c_n \\ b_1^* = b_1 - a_1 \\ b_n^* = b_n - c_n \end{cases} \quad \text{et} \quad X^* = X_1 + X_n$$

Une forme équivalente de (12.72) est :

$$\underbrace{\begin{pmatrix} b_1^* & c_1 & 0 & \cdot & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & a_n & b_n^* \end{pmatrix}}_{M^*} \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ X_n \end{pmatrix} + \begin{pmatrix} v_1 \\ 0 \\ \cdot \\ v_n \end{pmatrix} X^* = \begin{pmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_n \end{pmatrix}$$

et

$$X^* = X_1 + X_n.$$

La solution est cherchée sous la forme :

$$X_k = X_k^{(1)} - X_k^{(2)} \cdot X^*, \quad k = 1 \dots n \quad (12.73)$$

ce qui nous amène à résoudre deux systèmes à matrice tridiagonale de dimension  $n$  :

$$\begin{cases} M^* \cdot X^{(1)} = (f_1 f_2 \dots f_{n-1} f_n)^T \\ M^* \cdot X^{(2)} = (v_1 0 \dots 0 v_n)^T \end{cases} \quad (12.74)$$

L'inconnue supplémentaire introduite est calculée par :

$$X^* = \frac{X_1^{(1)} + X_n^{(1)}}{1 + X_1^{(2)} + X_n^{(2)}} \quad (12.75)$$

Pour résumer, les étapes de l'algorithme seront :

- résoudre les deux systèmes (12.74) par l'algorithme de Thomas (12.5) (cette étape peut être optimisée car les deux systèmes ont la même matrice  $M^*$ );
- calculer  $X^*$  de (12.75);
- trouver la solution par (12.73).