Simulation numérique : sillage d'un obstacle confiné

Quelles sont les questions scientifiques ou techniques ?
Instabilité du sillage derrière un obstacle : seuil d'instabilité, fréquence et amplitude des oscillations au-delà du seuil.

Par quelles expériences y répondre ?
Expérience modèle avec un obstacle rectangulaire confiné dans un canal (comme pour l'expérience de vélocimétrie laser).

Quelles techniques expérimentales ?
Simulation numérique par éléments finis.
Comparaison avec les expériences réalisées par vélocimétrie laser.

Quels sont les résultats ?
À vous de les montrer à travers des graphes clairs.

Comment les interpréter ?
Ingrédients physiques, lois d'échelle, ajustement de courbes expérimentales : à vous de jouer !

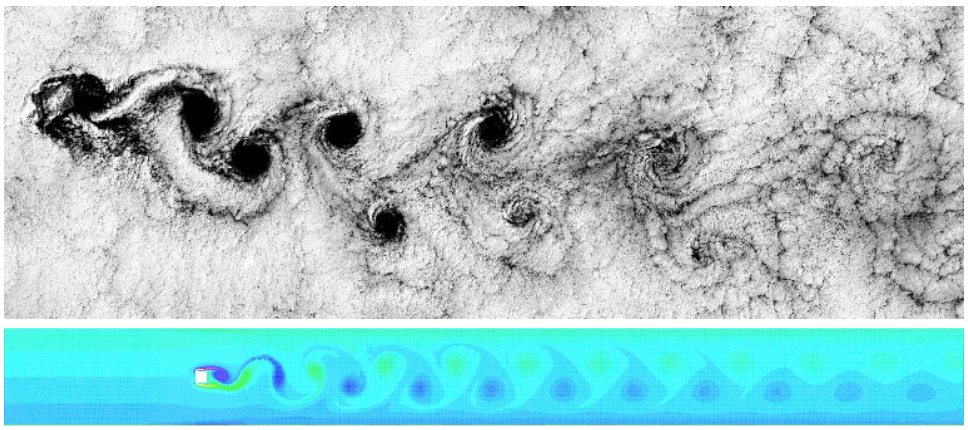

FIGURE 1 – Allée de vortex atmosphérique dans le sillage de l'île de Robinson Crusoé (archipel Juan Fernandez, Chili). Simulation numérique sous FreeFem (à un nombre de Reynolds modéré).

Le sillage d'un objet non profilé, de section circulaire ou carrée, présente une instabilité qui apparaît à des nombres de Reynolds de l'ordre de quelques dizaines. Cette instabilité, dite de Bénard-Von Karman, se manifeste par une allée de tourbillons alternés. Cette structure persiste jusqu'à des nombres de Reynolds extrêmement grands puisqu'elle est observée dans le sillage atmosphérique d'iles élevées, sur des échelles de plusieurs dizaines de kilomètres (photo ci-dessus). Le but de cette étude numérique est de caractériser cette instabilité près de son seuil et, accessoirement, de voir comment elle est influencée par le confinement entre deux parois, cette géométrie étant utilisée pour réaliser des débitmètres.

La manière la plus simple d'étudier cette instabilité est de faire varier le paramètre de contrôle (en l'occurrence le nombre de Reynolds) et d'observer la structure d'écoulement correspondante. On peut ainsi définir le nombre de Reynolds critique Re_c correspondant au seuil d'instabilité et les caractéristiques des modes instables (amplitude, fréquence temporelle et spatiale). Il est possible d'analyser plus finement la nature de l'instabilité en soumettant l'écoulement à des perturba-

tions bien contrôlées (impulsion localisée dans le temps, excitation sinusoïdale, créneau dans la vitesse moyenne) et en suivant la réponse du sillage à ces perturbations. C'est ce qui a été fait dans une étude expérimentale qui fait référence sur ce problème [1] et que nous allons reproduire numériquement.

Nous simulons ici un problème bidimensionnel, c'est à dire que nous supposons qu'il n'y a que deux composantes non nulles du champ de vitesse et que le problème est invariant dans la troisième dimension de l'espace. Le schéma du domaine de calcul est représenté sur la fig. 2. Le diamètre de l'obstacle est pris égal à 1. La largeur du canal *bloc* définit le rapport de blocage. Le centre de l'obstacle est placé à deux largeurs de canal en aval de l'entrée et la longueur du canal est au moins égale à 10 fois la largeur.

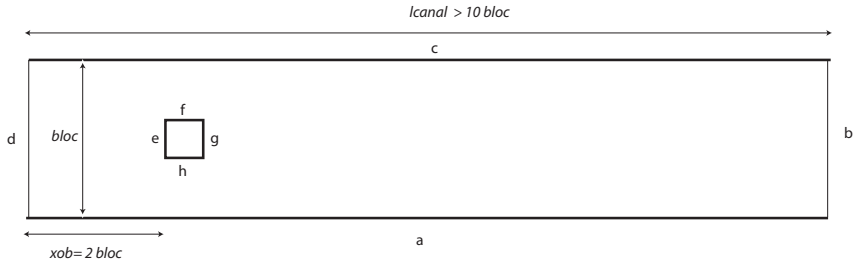

FIGURE 2 – Schéma du domaine de calcul pour un obstacle de section carrée, avec identification des différentes frontières (a-h).

Le logiciel utilisé pour le calcul est FreeFem++, un logiciel domaine public développé au laboratoire d'analyse numérique de l'UPMC. Les sources ainsi que les exécutables pour Unix, Windows et MacOsX sont disponibles gratuitement. FreeFem++ permet de résoudre des systèmes d'équations différentielles par les méthodes des éléments finis en utilisant une formulation variationnelle. Il permet également de générer automatiquement des maillages triangulés du domaine de calcul.

Expérience numérique

On se propose ici d'étudier quelques aspects de l'instabilité de Bénard-von Karman, en fixant le rapport de confinement, c'est-à-dire le rapport entre la largeur de l'obstacle et la largeur du canal :

- Quel est le seuil d'instabilité en nombre de Reynolds ? Ce seuil est-il comparable à celui déterminé expérimentalement par vos camarades ?

- Au delà du seuil d'instabilité, quelle est la fréquence des tourbillons émis et comment varie-t-elle avec le nombre de Reynolds ? Comment varie la force exercée par l'écoulement sur l'obstacle ? Les simulations rendent-elles compte des résultats expérimentaux obtenus pas vélocimétrie laser ? Plus généralement, comment passer des résultats numériques adimensionnés à des "vraies" grandeurs physiques ?

- Comment croît l'instabilité dans le sillage en fonction du temps ? Quel est le taux de croissance en fonction de l'écart (positif ou négatif) au seul d'instabilité ? Comment évolue-t-il avec Re au voisinage de Re_c ?

Finit-on par obtenir un régime permanent ? Dans ce cas, comment varie l'amplitude finale des oscillations en fonction du nombre de Reynolds ? Une théorie un peu poussée sur les instabilités suggère une évolution de l'amplitude finale sous la forme $A \sim A_0(Re - Re_c)^{1/2}$. Est-ce compatible avec vos résultats ? Pour répondre à cette question, vous pourrez utiliser la fonctionnalité du programme qui permet d'imposer divers types de perturbation sur le débit en entrée du canal.

- Comment pourriez-vous réaliser un débitmètre, ne comportant aucune partie mobile, en exploitant cette instabilité ?

Utilisation de FreeFem sous MacOs

Le calcul est entièrement défini dans un fichier texte "BVK_perturb.edp" détaillé ci-dessous. Ce fichier peut être modifié avec n'importe quel éditeur de texte. Sous MacOs, les éditeurs mi et Smultron présentent l'avantage de colorer le texte en fonction de la syntaxe du programme et d'être interfacés directement avec FreeFem.

Pour démarre le calcul, il suffit d'ouvrir le fichier BVK_perturb.edp soit :

– dans mi et d'exécuter la commande Tool/Run (ou le raccourci clavier cmd R).

– dans Smultron et exécuter la commande Commandes/Actions/FreeFem

FreeFem ouvre une fenêtre Terminal qui sert à l'interaction avec l'utilisateur. Lorsque la commande `plot` est utilisée, FreeFem ouvre une autre fenêtre pour afficher le maillage, les champs de pression, de vitesse, ...

Formulation variationnelle

Techniquement, les calculs sous FreeFem sont basés sur la formulation variationelle d'une équation différentielle [2]. Pour décrire ce qu'est la formulation variationnelle d'une équation différentielle [2], considérons le problème d'une poutre flexible, de longueur unité soumise à une charge répartie f. Le moment fléchissant local $u(x)$ est décrite par l'équation différentielle de Poisson qui s'écrit, une fois adimensionnée :

$$-u"(x) = f(x) \tag{1}$$

avec les conditions aux limites $u(0) = u(1) = 0$. En multipliant cette équation par une fonction continue et différentiable $v(x)$, on obtient :

$$-\int_0^1 u"(x)v(x)dx = \int_0^1 f(x)v(x) \tag{2}$$

En intégrant par parties le membre de gauche, on a :

$$\int_0^1 u'(x)v'(x)dx - [u"v]_0^1 = \int_0^1 f(x)v(x) \tag{3}$$

Si v est choisi telle que $v(0) = v(1) = 0$:

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x) \tag{4}$$

Le problème variationnel consiste à trouver la fonction u qui satisfait l'équation 4 quelle que soit la fonction v.

La méthode des éléments finis consiste à trouver une solution sous la forme : $u(x) = \sum_{i=1}^N u_i\phi_i(x)$ où les fonctions ϕ_i sont N fonctions linéairement indépendantes et u_i sont N nombres réels à déterminer. En prenant $v = \phi_j$, le problème 4 se ramène à trouver les nombres u_i tels que :

$$\sum_{i=1}^N u_i \left(\int_0^1 \phi_i'(x)\phi_j'(x)dx \right) = \int_0^1 f(x)\phi_j'(x)dx \tag{5}$$

pour tout j compris entre 1 et N. On définit la matrice de rigidité $N \times N$ A telle que $A_{ji} = \int_0^1 \phi_i'(x)\phi_j'(x)dx$. Si $\mathbf{u}$ est le vecteur de composantes u_i et $\mathbf{f}$ le vecteur de composantes $f_j = \int_0^1 f(x)\phi_j'(x)dx$, le problème se ramène à résoudre le système linéaire :

$$A\mathbf{u} = \mathbf{f} \tag{6}$$

et donc à inverser la matrice A.

Dans la méthodes des éléments finis de degré 1, on divise l'intervalle $[0,1]$ en $N+1$ parties de largeur $h = 1/(N+1)$ et commençant respectivement en $x_i = ih$. On définit les fonctions $\phi_i(x)$ "triangulaires"telles que :

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{h} & \text{si } x_{i-1} < x < x_i \\ \frac{x-x_{i+1}}{h} & \text{si } x_i < x < x_{i+1} \\ 0 & \text{si } x <= x_{i-1} \text{ ou } x >= x_{i+1} \end{cases} \tag{7}$$

Ceci revient à faire une approximation de la solution par un ensemble de N fonctions affines par morceaux.

De la même manière, on peut définir des éléments de degré 2 où les fonctions ϕ_i sont des fonctions quadratiques de x, la solution approchée étant une succession d'arcs de parabole. Ces résultats à une dimension peuvent être étendus aux cas bidimensionnel et tridimensionnel.

Programme utilisé pour le calcul

```
// Calcul de l'écoulement autour d'un obstacle non profilé placé dans un canal
// la largeur de l'obstacle est fixée à 1 ainsi que la vitesse moyenne
// on peut faire varier le rapport de blocage de l'écoulement
real bloc, lcanal ;
cout << " Entrer le rapport de blocage (>1) :"; cin >> bloc;
assert (bloc>1);
// la longueur du canal est au moins 10 fois plus grande que sa largeur
cout << " Entrer la longueur du canal (>10 et < 30) :"; cin >> lcrel;
assert (lcrel>10);
assert (lcrel<30);
lcanal=lcrel*bloc ;
// definition du nombre de mailles en fonction de lcanal et bloc
int nbloc,ncanal ;
nbloc=floor(bloc);
ncanal=floor(lcanal);
// limites du canal
// frontières décrites dans le sens trigo
border a(t=0,lcanal) {x=t;y=0;};
border b(t=0,bloc) {x=lcanal;y=t;};
border c(t=0,lcanal) {x=lcanal-t;y=bloc;};
border d(t=0,bloc) {x=0;y=bloc-t;};
// obstacle
// frontière décrite dans le sens inverse
// le centre de l'obstacle est placé à 2 largeurs en aval de l'entrée du canal
int C1=99;
real xob,yob;
xob=2.*bloc;
yob=0.5*bloc;
// section carrée
border e(t=-0.5, 0.5) {x=xob;y=yob+t;label=C1;};
border f(t=0, 1) {x=xob+t;y=yob+0.5;label=C1;};
border g(t=0.5, -0.5) {x=xob+1;y=yob+t;label=C1;};
border h(t=0, 1) {x=xob+1-t;y=yob-0.5;label=C1;};
```

Cette première partie permet de définir le domaine de calcul. Les différentes frontières sont définies par des équations paramétriques. Le domaine de calcul se trouve à gauche de la frontière décrite dans le sens trigonométrique. Notez que les sens de parcours permettent d'exclure l'intérieur de l'obstacle du domaine de calcul.

```
int i=0;
// Définition du maillage
int n=2;
cout << " Entrer la resolution du maillage (>1) :"; cin >> n;
mesh th = buildmesh(a(ncanal*n)+b(nbloc*n)+c(ncanal*n)+d(nbloc*n)+e(2*n)+f(2*n)+g(2*n)+h(2*n));
plot (th,wait=,ps='maillage.ps") ;

// définition des espaces d'éléments finis Xh et Mh sur le maillage th
// P2 éléments quadratiques continus par morceaux
// P1 éléments affines continus par morceaux
fespace Xh(th,P2); // velocity space
fespace Mh(th,P1); // pressure space
// u1,u2,p sont les deux composantes de vitesse et la pression calculées
// v1,v2,q sont les quantités conjuguées utilisées dans le calcul variationnel
Xh u2,v2;
Xh u1,v1;
Xh vor, sigma11, sigma22, sigma12 ;
Mh p,q;
```

Cette seconde partie permet de définir le maillage et les espaces d'éléments finis. Le maillage triangulaire est défini à partir du nombre de noeuds sur chacune des frontières du domaine de calcul. Ici un paramètre défini par l'utilisateur (n) permet de raffiner plus ou moins le maillage.

On définit les champs de vitesse ainsi que le champ de vorticité et les composantes du tenseur des contraintes sur l'espace Xh et le champ de pression sur l'espace Mh.

```
// définition de la condition aux limites en entrée du canal
// profil de vitesse parabolique
// u1(y=0) = 0 ; u1(y=bloc) = 0 : vitesse moyenne <u1> = 1
func uin=6*y*(bloc-y)/(bloc*bloc) ;

// calcul de la solution de Stokes

solve Stokes ([u1,u2,p],[v1,v2,q],solver=Crout) =
    int2d(th)( ( dx(u1)*dx(v1) + dy(u1)*dy(v1)
             +  dx(u2)*dx(v2) + dy(u2)*dy(v2) )
             + p*q*(0.000001)
             + p*dx(v1)+ p*dy(v2)
             + dx(u1)*q+ dy(u2)*q
            )
  + on(b,d,u1=uin,u2=0)
  + on(a,c,C1,u1=0,u2=0);
```

On définit le profil de vitesse en entrée du canal. On prend un profil parabolique, solution du problème à petit nombre de Reynolds. La vitesse moyenne est prise égale à 1, de manière à normaliser correctement les quantités calculées ultérieurement. On définit la forme variationnelle de l'équation de Stokes ($\eta\Delta\mathbf{u} = \nabla p$), le champ de vitesse étant à divergence nulle pour respecter la condition d'incompressibilité (cf § 9.6.1 de la documentation de FreeFem). La commande `solver=Crout` définit l'algorithme utilisé pour la résolution du système linéaire. On ajoute les conditions aux limites : profil parabolique définit par la fonction uin en entrée et en sortie (frontières b et d), vitesse nulle sur les parois du canal et sur l'obstacle (frontières a,c et C1).

```
// calcul et affichage de la fonction de courant
```

```
Xh psi,phi;
problem streamlines(psi,phi) =
int2d(th)( dx(psi)*dx(phi) + dy(psi)*dy(phi))
+ int2d(th)( -phi*(dy(u1)-dx(u2)))
+ on(a,psi=0);

streamlines ;
plot(psi,nbiso=50,wait=1);
```

Pour visualiser le champ de vitesse, il est commode de tracer les lignes de courant plutôt que d'afficher les vecteurs vitesse sur chaque élément du maillage. Dans un écoulement bidimensionnel incompressible, on peut définir une fonction de courant ψ telle que : $u_x = \partial\psi/\partial y$ et $u_y = -\partial\psi/\partial x$ ou encore : $-\Delta\psi = \text{rot}\mathbf{u}$. Les lignes de courant sont les lignes d'égale valeur de ψ. La commande `plot` permet d'afficher ces lignes de courant.

```
// attention à la définition de Reynolds.
// nu est l'inverse du nb de Reynolds
// vitesse moyenne = 1, taille de l'obstacle = 1

real reynolds ;
cout << " Entrer le nombre de Reynolds :"; cin >> reynolds;
real nu=1./reynolds;
// définition du pas de temps pour le problème instationnaire
real dt=0.1;
real alpha=1/dt;
// définition du problème de Navier-Stokes
Xh up1,up2;
problem NS (u1,u2,p,v1,v2,q,solver=Crout,init=i) =
int2d(th)(
alpha*( u1*v1 + u2*v2)
+ nu * ( dx(u1)*dx(v1) + dy(u1)*dy(v1)
+ dx(u2)*dx(v2) + dy(u2)*dy(v2) )
- p*q*(0.000001)
- p*dx(v1) - p*dy(v2)
- dx(u1)*q - dy(u2)*q
)
+ int2d(th) ( -alpha*
convect([up1,up2],-dt,up1)*v1 -alpha*convect([up1,up2],-dt,up2)*v2 )
+ on(d,u1=uin,u2=0)
+ on(b,u2=0)
  + on(a,c,C1,u1=0,u2=0);
//
```

Dans cette partie, on définit le problème de Navier-Stokes instationnaire $\rho(\partial_t u + u.\nabla u) = -\nabla p + \eta\Delta u$. La taille de l'obstacle et la vitesse moyenne ayant été prises égales à 1, le nombre de Reynolds du problème se réduit à : $Re = 1/\nu$. L'équation de Navier-Stokes, s'écrit donc, en variables adimensionnées : $\partial_t u + u.\nabla u = -\nabla p + \nu\Delta u$, la pression étant normalisée par la pression dynamique ρU^2. De la même façon que pour l'équation de Stokes, FreeFem résoud une forme variationnelle de l'équation (cf § 9.6.1 de la documentation de FreeFem). Pour calculer le terme instationnaire, il faut définir un pas de temps `dt` qui est ici pris égal à 0,1 (en variable adimensionnée).

```
// définition des valeurs pour la visualisation de la vorticité de -vmax à vmax
real[int] vorval(50);
real vmax=10;
for (i=0;i<50;i++){
vorval[i]=vmax*(-1.+0.04*i);
}
// fichier de sortie pour l'évolution temporelle de la vitesse derrière l'obstacle
string uxvstfile="ux_vs_t_b"+bloc+"_re"+reynolds+".txt" ;
ofstream uxvst(uxvstfile,append);
// fichier de sortie pour l'évolution temporelle de la force sur l'obstacle
string fvstfile="f_vs_t_b"+bloc+"_re"+reynolds+".txt" ;
ofstream fvst(fvstfile,append);
//
// définition de la position du point de mesure de la vitesse
// real xmes,ymes,xrel,yrel ;
//xmes = -lcanal ;
//ymes = 2*bloc ;
//cout << " Position x du point de mesure de vitesse par rapport à l'obstacle :"; cin >> xrel;
//xmes=xob+xrel;
// définition des paramètres pour les sorties graphiques
//string store_dir,base_dir ;
string yn;
bool calcf ;
real fx,fy ;
cout << "calcul des forces sur l'obstacle ? (oui non)"; cin >> yn;
calcf = (yn == "oui");
bool sortiep, sortiev, zoom ;
cout << "enregistrement des champs de pression ? (oui non)"; cin >> yn;
sortiep = (yn == "oui");
cout << "enregistrement des champs de vitesse ? (oui non)"; cin >> yn;
sortiev = (yn == "oui");
cout << " sorties graphiques sur une partie seulement du domaine de calcul ? (oui non)"; cin >> y
zoom = (yn == "oui");
real llx,lly,urx,ury;
llx=0. ;
lly=0. ;
urx=lcanal ;
ury=bloc ;
if (zoom) {
cout << "coordonnee x du point inferieur gauche :"; cin >> llx;
cout << "coordonnee y du point inferieur gauche :"; cin >> lly;
cout << "coordonnee x du point superieur droit :"; cin >> urx;
cout << "coordonnee y du point superieur droit :"; cin >> ury;
}
```

Dans cette partie, on définit les paramètres pour les sorties graphiques. Par défaut, on affichera le champ de vorticité $\omega_z = \partial_y u_x - \partial_x u_y$ de manière à visualiser les tourbillons dans le sillage de l'obstacle. On définit le fichier qui permettra d'enregistrer l'évolution temporelle de la vitesse en un point du domaine de calcul. Ensuite, on spécifie si on calcule les forces sur l'obstacle, si on enregistre le champ de pression, le champ de vitesse.

Par défaut les sorties graphiques affichent tout le domaine de calcul, mais on peut spécifier une zone plus petite.

```
// nombre d'itérations en temps
```

```
int nbiter ;
cout << " Entrer le nombre d iterations :"; cin >> nbiter;
for (i=0;i<nbiter;i++){
up1=u1;
up2=u2;
NS;
x=xob+2. ; y=yob ;
uxvst << i << "," << u1 << "," << u2 << "\n";
if (calcf){
sigma11=nu*dx(u1)-p;
sigma22=nu*dy(u2)-p ;
sigma12=nu*(dy(u1)+dx(u2));
fx=int1d(th,C1) (sigma11*N.x+sigma12*N.y);
fy=int1d(th,C1) (sigma12*N.x+sigma22*N.y);
fvst << i << "," << fx << "," << fy << "\n";
}
if ( !(i % 10)) {
vor = dy(u1)-dx(u2);
if (sortiep) {
plot(cmm="iteration no "+i,p,fill=1,bb=[[llx,lly],[urx,ury]],ps="p_b"+bloc+"_re"+reynolds+"_t"+i+
}
if (sortiev) {
// calcul de la fonction de courant
streamlines ;
plot (cmm="iteration no "+i,psi,nbiso=50,bb=[[llx,lly],[urx,ury]],ps="stream_b"+bloc+"_re"+reynol
}
plot(cmm="iteration no "+i,vor,viso=vorval,fill=1,bb=[[llx,lly],[urx,ury]],ps="vor_b"+bloc+"_re"+
}
}
```

Boucle de calcul proprement dite. Le temps total de calcul est `nbiter*dt`.

Noter qu'on réutilise le champ de vitesse déterminé à l'itération n (`u1,u2`) pour calculer la vitesse à l'itération $n+1$.

Pour calculer la force sur l'obstacle, on calcule les trois composantes indépendantes du tenseur des contraintes $\sigma = -pI + 2\eta(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$. En variables adimensionnées (contraintes normalisées par la pression dynamique ρU^2), on a :

$$\begin{aligned} \sigma_{xx} &= -p + \nu\frac{\partial u_x}{\partial x} \\ \sigma_{xy} &= \sigma_{yx} = \nu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \\ \sigma_{yy} &= -p + \nu\frac{\partial u_y}{\partial y} \end{aligned} \tag{8}$$

Pour afficher les lignes de courant, on doit recalculer la fonction de courant, par la commande `streamlines` définie plus haut.

```
// sortie de la vitesse longitudinale le long de l'axe
// pour la mesure de la longueur de recirculation
string uaxevsxfile="uaxe_vs_x_b"+bloc+"_re"+reynolds+".txt";
{
ofstream uaxef(uaxevsxfile,append);
real long,ddx;
long=lcanal-(xob+1);
ddx=0.01*long ;
for (i=0;i<100;i++){
x=xob+1+ddx*i ;
```

```
uaxef << x << "," << u1 << "\n";
}
}
```

Enfin, on enregistre dans un fichier le profil de vitesse de long de l'axe du canal. Ceci est utile pour mesure la longueur de la zone de recirculation derrière l'obstacle.

Références

[1] M. Provansal, C. Mathis & L. Boyer, Benard-von Karman instability : transient and forced regimes, J. Fluid Mech. 182, 1 (1987)

[2] Introduction à l'analyse numérique, J. Rappaz, M. Picasso, Presses Polytechniques Romandes (1998)