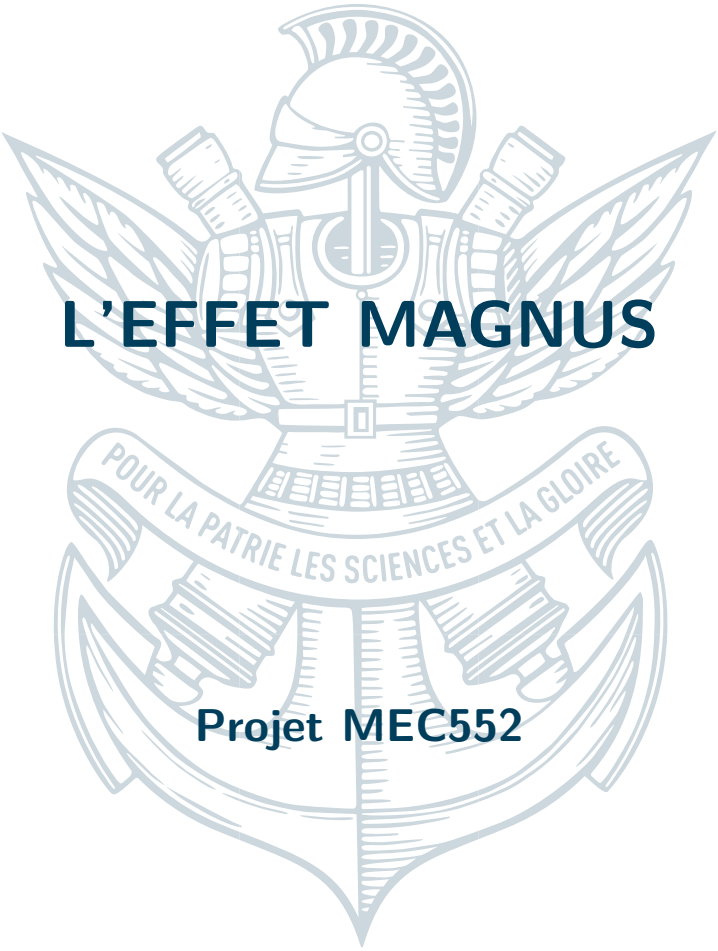

L'EFFET MAGNUS

Projet MEC552

Décembre 2019

Thibaut Badoual et Etienne Gourcerol

INTRODUCTION

En tant que joueurs de rugby, nous avons quotidiennement conscience des effets que l'on peut appliquer à une balle. Si cette dernière est lancée avec une rotation plus ou moins forte, sa trajectoire s'en trouve modifiée de manière souvent spectaculaire. Il suffit de visualiser les coups francs de footballeurs professionnels ou d'observer les coups coupés ou liftés des joueurs de tennis pour se rendre compte à quel point la rotation de la balle joue un rôle déterminant dans sa trajectoire.

A travers ce projet de mécanique des fluides numériques, nous avons alors cherché à modéliser numériquement l'effet majeur à l'origine de ces effets amusants : l'effet Magnus. Après une rapide description de l'effet Magnus, nous aborderons en détail la méthode numérique utilisée avant d'exposer nos résultats, leur sens physique, leurs limites et les améliorations possibles.

TABLE DES MATIÈRES

1 L'EFFET MAGNUS

1.1 Principe

Considérons un écoulement simple d'un fluide à la vitesse v autour d'un obstacle fixe. A bas nombre de Reynolds ($Re << 1$), le fluide contournera l'obstacle sans incident. Plus ce dernier croît, plus le fluide éprouvera des difficultés à joindre sans incidents les deux flux de fluide créés de part et d'autre de l'obstacle. A un certain seuil, une allée de Von Karman pourra alors être observée.

Que se passe-t-il si la balle - représentant l'obstacle - est en rotation ? De manière intuitive et si le fluide est visqueux, on peut facilement imaginer que le fluide au contact de la balle se mette à tourner lui aussi. Si ce dernier tourne dans le sens horaire, la vitesse apparente du fluide sera plus grande au dessus de la balle qu'en dessous. Un effet de portance vers le haut apparaît : **l'effet Magnus**.

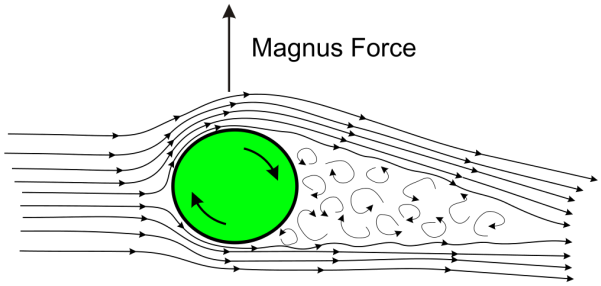

Figure 1 – L'effet Magnus

1.2 Bilan des forces

Plusieurs forces sont ici à prendre en compte dans le bilan des forces de l'obstacle.

— la force de Magnus : F_m
— la force de traînée : F_t
— la pesanteur g négligée dans la suite du problème

On trouve dans la littérature les expressions suivantes pour la force de Magnus et celle de traînée :

$$F_t = \frac{1}{2} \times \rho \times S \times C_x \times V^2 \tag{1}$$

$$F_m = \frac{1}{2} \times \rho \times S \times C_m \times V^2 = \frac{C_m}{C_x} \times F_t \tag{2}$$

avec ρ la masse volumique du fluide, S la surface de référence, V la vitesse du fluide loin de l'obstacle, C_x le coefficient de trainée et C_m son analogue pour l'effet Magnus (ces deux coefficients dépendent de l'état de surface).

Remarques :

— Premièrement, on observe qu'en l'absence de frottements (de viscosité en somme), l'effet Magnus tout comme la force de traînée n'existe pas.

— la force de Magnus est dirigée perpendiculairement à la vitesse de rotation de la balle et à la vitesse du fluide au loin, c'est-à-dire selon $\vec{\omega} \wedge \vec{V}$.

— dans le cadre d'une balle : $S = \pi \times R^2$

— à haut nombre de Reynolds ($Re >> 1$) : $C_x \approx 0,4$ et $C_m \approx \frac{R \times \omega}{V}$

1.3 Équations du mouvement

En appliquant le principe fondamental de la dynamique à la balle, nous obtenons pour un nombre de Reynolds élevé :

$$m\frac{d^2\vec{OM}}{dt^2} = \vec{F_m} + \vec{F_x} = \frac{1}{2}\rho\pi R^3\vec{\omega} \wedge \vec{V} + \frac{1}{2}\rho\pi R^2 V \times 0,4 \times \vec{V} \tag{3}$$

Nous utiliserons cette équation dans notre méthode numérique afin de calculer l'avancée de la balle. Pour ce faire on calculera la résultante des forces de pression sur la balle qui comprend à la fois la force de Magnus et celle de traînée.

1.4 Problème étudié

Le problème fixé est le suivant. Une balle est lancée dans un fluide légèrement en mouvement avec une vitesse initiale non nulle et en rotation infinie. Pour toute la suite du problème, la vitesse de rotation de la balle est inchangée.

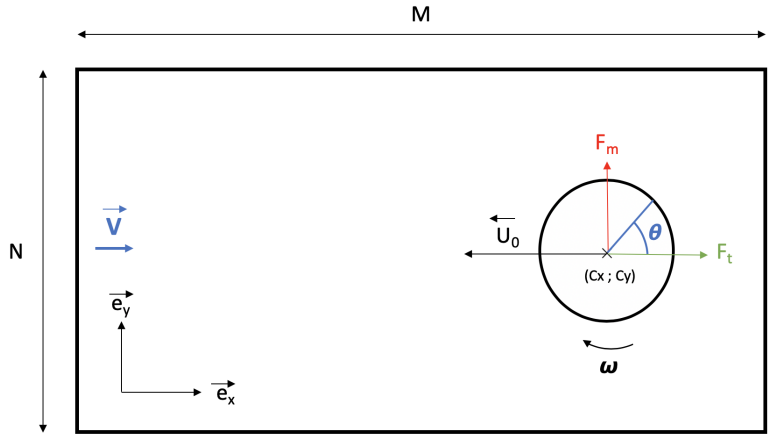

Figure 2 – Schéma de la situation

Pour un écoulement incompressible, les équations aux dérivées partielles (non linéaires) à résoudre en tous points du fluide sont :

$$\vec{\nabla}\vec{u} = 0 \tag{4}$$

$$\frac{\partial\vec{u}}{\partial t} + (\vec{u}.\vec{\nabla}).\vec{u} = -\vec{\nabla}p + \frac{1}{Re}\Delta\vec{u} \tag{5}$$

2
MÉTHODE NUMÉRIQUE UTILISÉE

La structure générale de notre algorithme est similaire à celle utilisée pour étudier les allées de Von Karman en PC. De nombreuses fonctions utilisées sont codées dans des fichiers distincts. Le code principal ainsi que le code de la résolution de l'équation de Poisson afin d'obtenir la pression sont exposés en Annexe.

La structure générale comporte les points suivants :

— définition des variables et initialisation
— définition de l'obstacle
— calcul du pas de temps
— résolution de l'équation d'advection et de diffusion
— pénalisation et correction
— intégration des forces de pression
— résolution du PFD et avancée de la balle
— affichage de la solution

2.1 Définitions des variables et initialisation

La première partie du code est dédiée à la définition de principales variables du problème. Nous définissons tout d'abord les variables de l'écoulement : le nombre de Reynolds Re, la masse volumique du fluide rho, la vitesse du fluide en amont U, la masse de la sphère m, le rayon et la vitesse de rotation de la balle (R et $Omega$) et la position et la vitesse initiale de la balle ($CX \times \mathrm{d}x, CY \times \mathrm{d}y$) et V_S.

Nous définissons ensuite la taille du domaine étudié avec $Long$ et $Larg$. Pour la grille numérique utilisée, nous fixons sa longueur M et sa largeur N avant de calculer les pas en espace $\mathrm{d}x$ et $\mathrm{d}y$. Les conditions aux limites sont implémentées avec un point fantôme, ainsi $\mathrm{d}x = \frac{Long}{M-2}$. On crée ensuite la matrice des points de la grille $[xx, yy]$.

Il faut enfin initialiser les matrice des vitesses, de la vorticité et du potentiel. Nous avons choisi de résoudre d'abord l'équation d'advection avec **une méthode semi-lagrangienne** avant de s'attaquer à celle de diffusion à l'aide de **la méthode des pas fractionnaires**. Ces deux méthodes nécessitent ainsi de définir une vitesse d'advection (uadv et vadv) tout comme une vitesse intermédiaire (ustar, vstar).

Il ne nous reste plus qu'à fixer le nombre d'itérations $nitermax$ avant de se plonger dans le calcul des vitesses, pression et potentiel à chaque itération.

2.2 Calcul des matrices Solid et S_{theta}

On utilise dans cet algorithme la méthode de pénalisation pour représenter notre balle en mouvement dans le fluide. C'est la matrice $Solid$ qui s'en charge en étant nulle à l'intérieur de l'obstacle et égale à 1 à l'extérieur. La balle étant en mouvement, il faut la redéfinir à chaque itération en annulant tous les points situés à moins de R du centre (X_i, Y_i) de la balle.

Les deux matrices $S_{theta,sin}$ et $S_{theta,cos}$ sont utilisées pour calculer la vitesse d'entraînement du fluide au bord de la balle ainsi que l'intégrale des forces de pression lors du PFD.

— $S_{theta,sin}$ est nulle partout sauf pour les points autour de la balle qui ont pour valeur l'angle formé entre le vecteur rayon et le vecteur [1 0] (comme sur le cercle trigonométrique) en degrés.

— $S_{theta,cos}$ vaut 90 partout sauf pour les points autour de la balle qui ont pour valeur l'angle formé entre le vecteur rayon et le vecteur [1 0] (comme sur le cercle trigonométrique) en degrés.

Pour ce faire on repère tous les points valant 1 qui ont au moins un voisin nul. Ceux-ci sont donc situés juste au bord de la balle.

2.3 Calcul de Δt pour satisfaire la condition de stabilité

En utilisant le schéma δ_- pour le terme d'advection, on trouve la condition CFL pour Δt et $\Delta x : 0 < c\frac{\Delta t}{\Delta x} < 1$ Or ici notre algorithme est en deux dimensions et les vitesses sont loin d'être uniformes. On choisit donc Δt tel que $c\frac{\Delta t}{\Delta x} = 0.8$ avec le minimum sur les deux dimensions du rapport $\frac{\Delta x}{c}$.

2.4 Pas d'advection

La première étape de notre calcul consiste à résoudre le pas d'advection suivant :

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u}.\vec{\nabla}).\vec{u} = 0 \tag{6}$$

Pour ce faire, nous avons utilisé le formalisme semi-lagrangien. En effet, l'équation (6) de notre problème devient alors :

$$\frac{D\vec{u}}{Dt} = 0 \tag{7}$$

Grâce à cette équation, nous avons l'évolution observée en un point fixe de l'espace. Si nous connaissons au temps t la valeur de la vitesse à chaque point de notre distribution, nous pouvons connaître en ces mêmes points la valeur de la vitesse au temps t+1. Pour réguler la densité des "traceurs", c'est-à-dire la disposition de nos points, et ainsi éviter une approximation numérique trop grande dans certaines régions, il faut réaliser une interpolation bilinéaire de la grille précédente. Un nouvel ensemble de traceurs est ainsi obtenu, il ne reste plus qu'à itérer.

2.5 Pas de diffusion

On résout l'équation de diffusion $\vec{u}^* = \vec{u}^a + \frac{\Delta t}{Re}\Delta\vec{u}$ avec les conditions aux limites suivantes pour lesquelles on utilise la méthode du point fantôme :

— **A gauche :** $u = u_0$ et $v = 0$
— **A droite :** $\delta_x u = 0$ et $\delta_x v = 0$
— **En haut :** $\delta_y u = 0$ et $v = 0$
— **En bas :** $\delta_y u = 0$ et $v = 0$

Pour calculer le laplacien de u et v on réalise un schéma du second ordre à 5 points :

$$\Delta u_j^i = \frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta x^2} + \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta y^2} \tag{8}$$

2.6 Pénalisation

On ajoute tout d'abord la vitesse d'entraînement $\vec{v_{ent}} = -R\omega\vec{e_\theta}$ de la balle en rotation à tous les éléments du fluide situés autour de celle-ci de la manière suivante :

- Pour u : $u = u + v_{ent}sin(\theta)$
- Pour v : $v = v - v_{ent}cos(\theta)$

On pénalise ensuite la vitesse avec la matrice *Solid*.

2.7 Pas de correction

Le second pas de la méthode des pas fractionnaires consiste à corriger le champ de vitesse obtenu jusqu'à la. L'écoulement étant incompressible, on va forcer le résultat : $\vec{\nabla}\vec{u} = 0$. Pour ce faire on calcule d'abord le potentiel des vitesses ϕ avec l'équation $\Delta\phi = \vec{\nabla}\vec{u}$ (Équation de Poisson). Pour calculer la divergence, on utilise un schéma δ_0 du second ordre en espace :

$$\vec{\nabla}\vec{u}^i_j = \frac{u^{i+1}_j - u^{i-1}_j}{2\Delta x} + \frac{u^i_{j+1} - u^i_{j-1}}{2\Delta y} \tag{9}$$

Pour résoudre l'équation en ϕ, on utilise un schéma à 5 points pour le laplacien. Le calcul revient à résoudre le système linéaire d'inconnu U : $LAPL \times U = f$ avec $f = \vec{\nabla}\vec{u}$ avec des conditions aux limites de Neumann à gauche, en haut et en bas et de Dirichlet à droite.
Une fois ϕ obtenu, on corrige le champ de vitesse par le gradient de ϕ.

2.8 Calcul de la résultante des forces de pression et PFD

La première étape est de calculer la pression en tout point du fluide. L'équation à résoudre est la suivante :

$$\Delta p = \frac{1}{\Delta t} * \vec{\nabla}.\vec{u} = f \tag{10}$$

De la même manière que précédemment, nous résolvons cette équation de poisson à l'aide de la fonction $Poiss_p$. Les conditions aux limites pour la pression sont des conditions de Neumann.

Une fois la pression calculée en tout point, et particulièrement autour de la balle, nous calculons l'intégrale des forces de pression par unité de longueur autour de la balle F_l.

$$\vec{F}l = R\int_0^{2\pi} \vec{p}(\theta)\mathrm{d}\theta \tag{11}$$

Remarques : l'intégrale des forces de pression correspond à la somme de la force de traînée et à celle de Magnus.

Il ne nous reste plus qu'à résoudre (3) à l'aide d'un schéma d'Euler explicite pour obtenir V la vitesse de la balle, puis (Xi, Yi) sa position à l'itération suivante.

3
CRITIQUES ET RÉSULTATS

3.1 Résultats

Trois codes similaires ont été écrits afin de visualiser correctement les résultats obtenus :

- **MagnusEffect** : renvoie l'évolution de la balle dans le fluide, sa trajectoire marquée en rouge et la résultante des forces de pression en vert.
- **MagnusEffect-Re** : renvoie la trajectoire de la balle pour quatre nombre de Reynolds différents.
- **MagnusEffect-Omega** : renvoie la trajectoire de la balle pour quatre rotations de la balle différentes.

Dans toutes les simulations numériques réalisées, le fluide est de l'eau, la pesanteur est négligée, le rayon de la balle est fixe égale à 1, la vitesse initiale de la balle $V_0 = [-5, 0]$, la masse m fixée à 10 et la vitesse du fluide ambiant U fixée à 2.

Les deux paramètres pertinents à étudier et à faire évoluer sont : **le nombre de Reynolds** *Re* et **la vitesse de rotation de la balle** *Omega*.

3.1.1 • MagnusEffect

Étudions l'effet de rotation de la balle sur l'écoulement fluide et sur sa propre trajectoire. La simulation numérique donne le figure suivante.

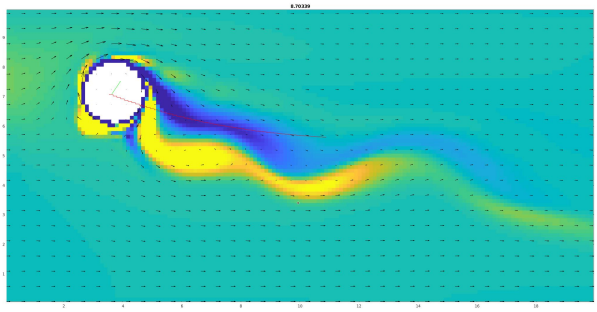

Figure 3 – Trajectoire de la balle : Re=1e6, Omega=2pi/5

La trajectoire est marquée en rouge et la résultante des forces de pression en vert. Comme prévu en première partie, les forces qui s'appliquent sur la balle ont une composante horizontale due à la traînée de la balle mais aussi verticale grâce à l'effet Magnus. Ainsi, non seulement la balle est freinée par le fluide mais aussi dévié par sa rotation.

Afin de mieux percevoir l'effet de rotation sur la balle, effectuons maintenant différentes simulations pour une balle fixe à Reynolds fixé mais dont la rotation évolue (Figures 4 à 7). On observe bien, pour une vitesse de rotation nulle, l'allée de Von Karman déjà étudiée. La couche limite de la balle se décolle bien. Augmentons la vitesse de rotation. Plus la vitesse de rotation augmente, plus la trajectoire de ces tourbillons créés par la balle est modifiée et plus ces tourbillons perdent en intensité. A forte rotation (Figure 7), la simulation numérique fait même quasiment disparaître cette allée de Von Karman.

Tout cela est parfaitement logique et se comprend physiquement. En effet, plus la vitesse de rotation de la balle augmente, plus la couche limite est énergétique, donc moins elle a tendance à se décoller. Une conséquence immédiate de cet effet est une diminution de la force de traînée de la balle. Cette dernière est en effet constituée d'une composante liée au frottement de la balle avec le fluide, et d'une autre liée aux instabilités tourbillonnaires. C'est cette dernière composante qui diminue fortement si les tourbillons sont moindres, c'est-à-dire si la balle est en rotation. Cet effet bien connu justifie d'ailleurs les trous dans les balles de golf.

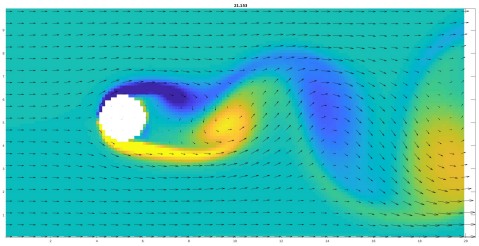

Figure 4 – Re=1e6, Omega=0

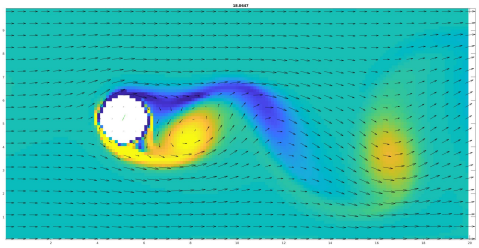

Figure 5 – Re=1e6, Omega=pi/10

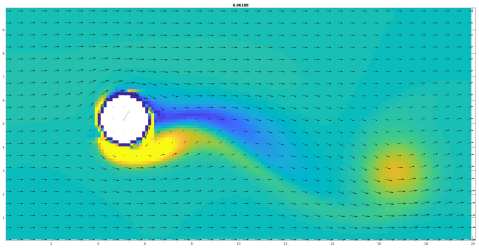

Figure 6 – Re=1e6, Omega=pi/5

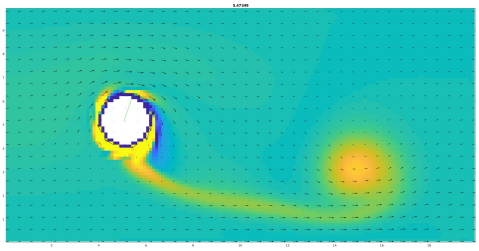

Figure 7 – Re=1e6, Omega=2pi/5

3.1.2 • MagnusEffect-Re

Étudions maintenant l'effet du changement du nombre de Reynolds sur la trajectoire de la bille (Figure 8). Conformément à notre intuition, pour un nombre de Reynolds faible, la viscosité du fluide est importante ce qui ralentit considérablement notre fluide. On observe bien sur notre simulation numérique une courbe bleue dont la longueur est très faible. Comme vu dans l'équation (1) et (2), pour Re faible, Cx est élevé donc F_m est faible au contraire de F_t élevée.

Pour des nombres de Reynolds élevés, la viscosité est plus faible, la force de traînée aussi (Cx diminue donc F_t aussi), la trajectoire est donc plus longue. La force de Magnus augmente en revanche étant inversement proportionnel à Cx. Les trajectoires orange, jaune et violette sont donc plus longues et subissent une déviation verticale plus élevée. On observe que pour $Re > 1e4$ les trajectoires sont moins longues que celle à $Re = 1e2$ mais ont une pente plus forte.

Deux facteurs expliquent cela :

— l'évolution de Cx par rapport au nombre de Reynolds révèle que $Cx(Re = 1e2) > Cx(Re > 1e4)$. Le coefficient de traînée se stabilisant pour Re compris entre 1e3 et 1e6. Ainsi, en accord avec l'équation (2), l'effet Magnus pour $Re = 1e4$ et $Re = 1e6$ est le même (même Cx) mais est inférieur pour $Re = 1e2$ (Cx étant plus élevé). Les courbes jaune et violette subissent donc la même déviation, plus importante que celle de la courbe orange.

— les mêmes arguments sur le coefficient de traînée et son évolution par rapport au nombre de Reynolds permettent de justifier que les courbes jaune et violette soient plus courtes que celle orange. La force de traînée en accord avec l'équation (1) est en effet plus importante pour les nombres de Reynolds supérieur à $1e4$.

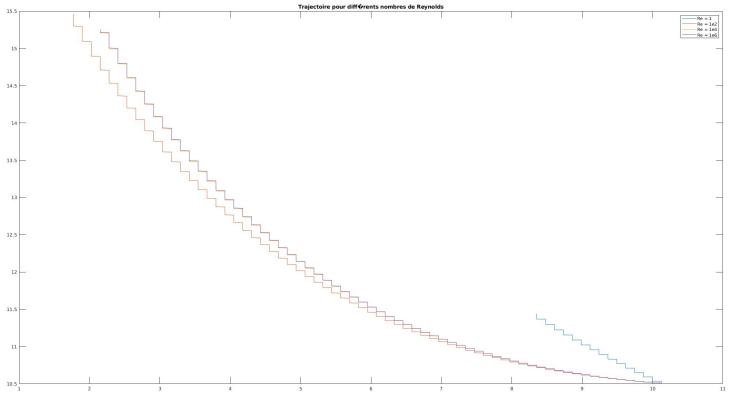

Figure 8 – Trajectoire de la balle pour différents nombres de Reynolds, Omega=2pi/5

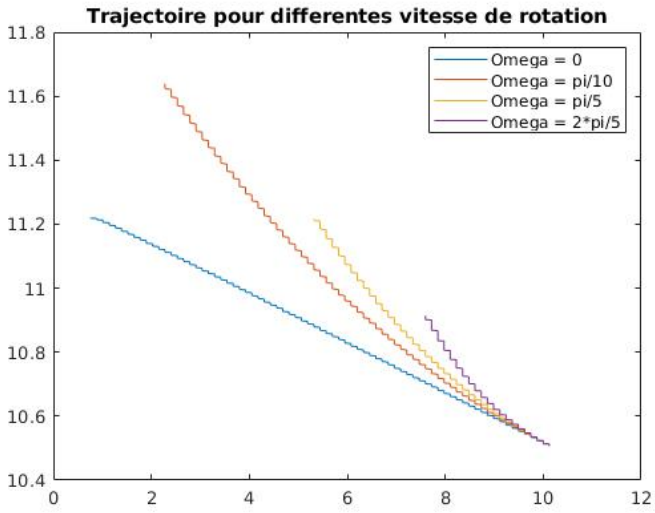

Figure 9 – Trajectoire de la balle pour différentes vitesse de rotation, Re=1e6

3.1.3 • MagnusEffect-Omega

Nous avons enfin simulé numériquement les trajectoires d'une balle dotée d'une vitesse de rotation variable (Figure 9). Le premier constat (développé dans la partie améliorations) est que même à rotation nulle, la trajectoire de la balle n'est pas horizontale, ce qui semble absurde. Sans rotation, la résultante des forces de pression a une composante verticale nulle par symétrie. Comment expliquer ce résultat ?

La raison de cette incohérence réside dans nos matrices $S_{theta,sin}$ et $S_{theta,cos}$ qui sont asymétriques : les angles $\theta = 0$ et $\theta = pi$ ne sont pas situés sur l'axe de symétrie horizontale de la balle mais un peu au-dessus. Ainsi, à rotation nulle, la composante du bas vers le haut des forces de pression est plus importante par pure calcul numérique.

Modifions donc notre repère pour rendre nos résultats analysables physiquement et choisissons comme horizontale cette trajectoire à rotation nulle. Plus la vitesse de rotation est importante, plus la pente de la trajectoire est plus, plus la déviation est élevée. Ceci est tout à fait logique puisque Cm dépend directement de la vitesse de rotation. Plus cette dernière est élevée, plus F_m (équation (2)) est élevée, donc plus la déviation

est importante.

3.2 Limites de la modélisation numérique

Un point important peut être aussi soulevé : comme le schéma que nous avons utilisé pour l'advection est stable même en l'absence de diffusion physique, nous pouvons en pratique augmenter Re indéfiniment sans observer d'instabilité. Cependant il est clair que la solution développe des échelles de plus en plus petites à mesure que Re est augmenté. Ce sont donc les effets de troncature numérique (comme la diffusion numérique) qui la régularisent en pratique. Les solutions à grande valeur de Re n'ont donc pas grand sens sur une grille réduite. Il faudrait augmenter la résolution numérique (donc la taille de la grille) pour les cas à grand nombres de Reynolds.

3.3 Autres possibilités et améliorations

Une première modification possible se situe dans la méthode des pas fractionnaires. En effet on passe dans cet algorithme par le potentiel des vitesses pour corriger *uadv* et *vadv* car l'écoulement étant irrotationnel on a $\Delta\phi = 0 = \vec{\nabla}\vec{u}$. La modification consisterait à résoudre le pas de diffusion en itérant 4 ou 5 fois avec le champ de vitesse et la pression. On calcule implicitement la vitesse en résolvant un système linéaire à partir des anciennes vitesses et pressions, puis on calcule la pression à l'aide de l'équation de Poisson (comme dans cet algorithme) puis on boucle. Ensuite on retire le gradient de la pression au champ de vitesse pour correction.

Cette méthode aurait eu l'avantage de donner directement la pression sans avoir à la calculer en plus pour calculer la résultante de pression sur la balle. Cependant, elle aurait demander de rajouter 4 ou 5 itérations dans lesquelles on résout trois fois un systèmes linéaires et on calcule une divergence contre seulement 2 systèmes, une divergence et un laplacien dans notre méthode.

Une autre amélioration possible serait de calculer les couples de frottements sur la balle qui ralentirait ainsi sa rotation au fur et à mesure. Pour ce faire, il faudrait calculer les forces de viscosité en jeu tout autour de la balle à l'aide de la viscosité du fluide et du gradient de vitesse normal. Ensuite, on aurait pu appliquer un théorème du moment cinétique à la balle en rotation et déterminer ainsi l'évolution de $\Omega(t)$.
Ceci nous permettrait sans doute de simuler l'écoulement pour des faibles nombres de Reynolds, ceux-ci divergeant pour le moment.

Une autre amélioration possible serait d'avoir un maillage plus fin notamment autour de la balle. En effet, comme on l'a vu dans les résultats de **MagnusEffect-Omega**, à rotation nulle la trajectoire de la balle n'est quand même pas horizontale à cause d'une asymétrie des matrices angulaires. Pour tendre vers une symétrie parfaite, une solution consiste à augmenter la densité des points du maillage autour du solide uniquement, pour éviter un calcul numérique trop coûteux.

Enfin, une dernière amélioration possible serait de rentrer d'utiliser des variables dimensionnées afin de pouvoir utiliser cet algorithme pour obtenir de vrais résultats quantitatifs et non qualitatifs. Pour ce faire, il faudrait d'abord augmenter la taille de l'espace pour le passer à quelques dizaines de mètres et ensuite réduire la taille de la balle à 10cm. Afin d'avoir une bonne résolution de la balle (à au moins 1 cm près), il faudrait avoir N et M de l'ordre de 1000 ce qui rallongerait énormément le temps de calcul. On pourrait alors remonter aux coefficients C_m et C_x et vérifier ainsi si les valeurs théoriques, à haut Re notamment, sont valides.

CONCLUSION

Cette simulation numérique sur l'effet de Magnus semble ainsi avoir rempli ses objectifs. Non seulement nous avons réussi à simuler cet effet et à en comprendre les causes et les conséquences, mais nous avons aussi commencé à percevoir les avantages et les inconvénients de la simulation numérique. Rien ne peut être laisser au hasard. Tout résultat incohérent peut tout aussi bien provenir d'une erreur dans les équations tout comme d'une approximation numérique.

Notre simulation fonctionne ainsi bien dans une plage de Re donné mais montre des dysfonctionnements à très faible ou très haut nombre de Reynolds, soit menant à des solutions divergentes, soit en lissant excessivement les résultats. Quoi qu'il en soit, nous avons ainsi un peu commencé à appréhender l'utilisation complexe et délicate de la simulation numérique sous MatLab.

ANNEXE

3.4 Code principal

```
%=========================================
%
%        Navier-Stokes flow opened domain
%        Magnus Effect with high Re
%
%=========================================

clear all
close all

% Global variables
global M;
global N;
global dx;
global dy;
global uadv;
global vadv;
global gradphix;
global gradphiy;
global vort;

%Fluid properties
L=1;
U=2;
Re = 1e3;
rho = 1.293e-3;

%Ball in rotation properties
R = 1;
Omega = 2*pi/5;
m = 10;
V_S = [-5 0];

% Domain size
Long = 20*L;
Larg = 10*L;

% Grid size
M = 200;
N = 100;

dx = Long/(M-2);
dy = Larg/(N-2);

% Gridding
x = linspace(dx/2,Long-dx/2,M-2);
y = linspace(dy/2,Larg-dy/2,N-2);
[xx,yy] = meshgrid(x,y);


```

```
% Initializations
u = zeros(N,M);
v = zeros(N,M);

vort = zeros(N,M);

uadv = zeros(N,M);
vadv = zeros(N,M);

ustar = zeros(N,M);
vstar = zeros(N,M);

divstar = zeros(N,M);

phi = zeros(N,M);
gradphix = zeros(N,M);
gradphiy = zeros(N,M);


niter = 0;
nitermax = 2000;

CX = M;
CY = N/2;

Xi = CX;
Yi = CY;
L_X = zeros(nitermax,1);
L_Y = zeros(nitermax,1);

% ITERATIONS
while ((niter<nitermax))
  if((mod(niter,50)==0)&&(niter>1))
      fprintf('Iteration: %d', niter);
      fprintf('\n');
  end

    % Define the obstacle
    cx = (Xi)*dx;
    cy = (Yi)*dy;

    % Mask for the code (0 in the solid, 1 outside)
    Solid=ones(N,M);
    Solid(2:N-1,2:M-1)=0.5+0.5*sign(sqrt((xx-cx).^2+(yy-cy).^2)-R);
    % Mask for plotting (NaN in the solid, 1 outside)
    Solid2=Solid;
    Solid2(Solid==0)=NaN;

    %Matrix of the angles around the solid
    S_theta_sin= zeros(N,M);
    S_theta_cos= ones(N,M)*pi/2;
    comp = 0;
    for k=2:M-1
      for i=2:N-1
        if Solid(i,k) == 1
          if (Solid(i-1,k) == 0)|| (Solid(i+1,k) == 0) || (Solid(i,k-1) == 0) || (Solid(i,k+1) == 0)
            S_theta_sin(i,k) = atan2d(i-Yi, k-Xi)*pi/180;
            S_theta_cos(i,k) = atan2d(i-Yi, k-Xi)*pi/180;
            comp = comp + 1;
```

```
            end
        end
    end
end

dt = CFL(u,v);
% Advection step
SemiLagUV(u,v,dt);
% Diffusion step (explicit)
ustar = dt/Re*Laplacien(u)+uadv;
vstar = dt/Re*Laplacien(v)+vadv;

% Boundary conditions
% LEFT
ustar(1:N,1)=-ustar(1:N,2) + 2.0*U;
vstar(1:N,1)=-vstar(1:N,2);
% TOP
ustar(N,1:M)=ustar(N-1,1:M);
vstar(N,1:M)=-vstar(N,1:M);
% BOTTOM
ustar(1,1:M)=ustar(2,1:M);
vstar(1,1:M)=-vstar(2,1:M);
% RIGHT
ustar(1:N,M)=ustar(1:N,M-2);
vstar(1:N,M)=vstar(1:N,M-2);

% Immerged solid
ustar = ((ustar + R * Omega * sin(S_theta_sin)).* Solid);
vstar = ((vstar - R * Omega * cos(S_theta_cos)).* Solid);

% Compute divergence
divstar=div(ustar,vstar);
%Resolve \Delta phi = div ustar
phi = Poiss(divstar,N,M);

% Boundary conditions
% LEFT
phi(2:N-1,1)=phi(2:N-1,2);
% RIGHT
phi(2:N-1,M)=-phi(2:N-1,M-1);
% TOP
phi(N,2:M)=phi(N-1,2:M);
% BOTTOM
phi(1,2:M)=phi(2,2:M);

% Projection step
grad(phi);
u = ustar + gradphix;
v = vstar + gradphiy;

%Compute pressure with Neumann BCs
d = div(u,v);
pnew = Poiss_p(N,M,dx,dy,d/dt);
p = pnew;

%Pressure force
Fl = - R*2*pi/comp*[sum(sum(p.*cos(S_theta_cos))) sum(sum(p.*sin(S_theta_sin)))];

%PFD
```

```
169       V_S = V_S + dt/m * (Fl);
170
171       %Moving the ball forward
172       Xi = Xi + dt*V_S(1);
173       Yi = Yi + dt*V_S(2);
174
175
176       niter = niter+1;
177       L_X(niter) = Xi;
178       L_Y(niter) = Yi;
179
180       if((mod(niter,50)==1)&&(niter>1))
181           Fl
182           Xi
183           Yi
184               figure(1)
185           % Compute vorticity
186           curl(u,v);
187           Svort=Solid2.*vort;
188           pcolor(xx,yy,Svort(2:N-1,2:M-1));
189           caxis([-4,4])
190           shading flat
191
192           hold on
193               quiver(xx(1:4:N-2,1:4:M-2),yy(1:4:N-2,1:4:M-2),...
194                   u(2:4:N-1,2:4:M-1),v(2:4:N-1,2:4:M-1),0.9,'k');
195              quiver(Xi*dx,Yi*dy,Fl(1),Fl(2),0.6,'g');
196
197           axis image
198           plot(floor(L_X(1:niter))*dx,L_Y(1:niter)*dy,color='r')
199           hold off
200               title(niter*dt)
201           drawnow
202
203
204       end
205
206   end
```

3.4.1 • Équation de poisson pour la pression

```
1
2  function pnew = Poiss_p(N,M,dx,dy,f)
3
4
5  %==============================================
6  %
7  %  routine solves pressure Poisson equation
8  %  (with Neumann boundary conditions)
9  %
10 %==============================================
11
12 NX=N-2;
13 NY=M-2;
14
15 %%% LAPLACIAN WITH NEUMANN BCs (pressure laplace operator)
16
17 %%% derivee seconde en x
18 a11=-1;   % Neumann
19 b11=-1;   % Neumann
20 C1 = [1 a11 0];
21 C2 = ones(NX-2,1)*[1 -2 1];
22 C3 = [0 b11 1];
23 DXX = spdiags([C1; C2; C3],-1:1,NX,NX)'/dx^2;  %'
24
25 clear C1;
26 clear C2;
27 clear C3;
28
29 %%% derivee seconde en y
30 a11=-1;   % Neumann
31 b11=-1;   % Neumann
32 C1 = [1 a11 0];
33 C2 = ones(NY-2,1)*[1 -2 1];
34 C3 = [0 b11 1];
35 DYY = spdiags([C1; C2; C3],-1:1,NY,NY)'/dy^2;  %'
36
37 clear C1;
38 clear C2;
39 clear C3;
40
41 LAPLN =  kron(speye(NY),DXX) + kron(DYY,speye(NX));
42
43 % Fixes the value at one point to ensure uniqueness
44 LAPLN(1,2:end)=0.0;
45
46
47
48
49 % Enforce the solvability condition with Neumann BCs
50
51 mean_val=sum(sum(f(2:N-1,2:M-1)))/(N-2)/(M-2);
52 f(2:N-1,2:M-1) = f(2:N-1,2:M-1) - mean_val;
53
54 %%% Membre de droite
55 f2D=f(2:N-1,2:M-1);
56 f1D = reshape(f2D,NX*NY,1);
```

```
57
58 %%% Resolution
59 u1D = LAPLN\f1D;
60
61 u2D=reshape(u1D,NX,NY);
62
63 pnew=zeros(N,M);
64 pnew(2:N-1,2:M-1)=u2D;
65
66 % ghost cell mapping
67 pnew(1,:)=pnew(2,:);
68 pnew(end,:)=pnew(end-1,:);
69 pnew(:,1)=pnew(:,2);
70 pnew(:,end)=pnew(:,end-1);
```