

P S E

Matériel & Méthodes

ESPCI  PARIS

MODÉLISATION DE LA DYNAMIQUE D'UN

ICE MÉLANGE

DANS UN FJORD EN AVAL D'UN GLACIER

Professeurs encadrants:

Emmanuel FORT

Philippe NGHE

Suzie PROTIERE

Chloé GOUPY

Linh TRAN

Cédric LEAU

Promotion 137

M A I 2020

S O M M A I R E

1. Mise en place expérimental	p.1
1.1 Inventaire des matériaux utilisés	p.1
1.2 Le moteur pas à pas et Arduino	p.1
1.3 Fabrication des flotteurs et choix du matériau	p.1
1.4.Construction de la cuve.	p.1
1.5.Mise en place dispositif expérimental.....	p.1
2. Analyse des résultats.	p.2
2.1 Programmation Matlab.....	p.2
2.2 Expériences avec des paramètres variés.....	p.2
2.3 Limites des expériences	p.2
3. Bibliographie	p.2
3.1 Article de départ.	p.2
3.2 Arduino.	p.2
3. Annexes .	
3.1 Programme de traitement MatLab	p.2
3.2 Plan du montage	p.10
3.3 Figures.	p.11
4. Remerciements	p.11



DÉMARCHE ET OBJECTIFS

L'objectif de notre projet PSE est de reproduire et d'analyser à l'échelle du laboratoire la dynamique d'écoulement d'un glacier en amont d'un Fjord. Après avoir fait des recherches dans la littérature scientifique, nous avons décidé concevoir un dispositif en se basant sur les travaux de J.C.Burton et al. [2018]. Nous détaillerons ici les protocoles mis en œuvre, nous évoquerons succinctement les résultats obtenus.

1. MISE EN PLACE EXPÉRIMENTALE

1.1. Inventaire des matériaux utilisés

- Moteur pas à pas 28 BYJ-48
- Carte Arduino Uno
- Caméra
- Logiciel PylonViewer
- Logiciel Matlab 2019b
- Cuve étanche (100cm x 14,5cm x 14,5cm)
- Plaque blanche pour faire le blanc (100cm x 14,5cm)
- PS (Polystyrène extrudé), acheté chez Dalbe, découpé en plaques d'épaisseur 5mm
- POM(Polyoxymethylene), acheté chez Weber Metaux, plaque d'épaisseur 15,5mm
- Découpe-laser

1.2. Le moteur pas à pas et Arduino

Pour faire avancer la plaque qui nous servira de terminus glacier, nous avons utilisé un moteur pas à pas . Nous avons programmé sur Arduino un code permettant de le faire avancer à une vitesse constante. Le moteur peut être facilement contrôlé avec un interrupteur. Le code est détaillé en annexe.

1.2. Fabrication des flotteurs et choix du matériau

Nous réalisons un ensemble de 500 flotteurs de diamètres différents destinés à être placés dans une cuve remplie d'eau. Afin de se rapprocher au mieux des conditions réelles, la distribution des diamètres des flotteurs est déterminée à partir de la polydispersité observée expérimentalement pour les icebergs d'un ice mélange et trouvée dans la littérature (cf. Annexe). Pour qu'ils représentent au mieux les icebergs, nous fabriquons des flotteurs de densité semblable (d=0.9) avec deux matériaux : du polystyrène (PS) pour la partie flottante et une quille en Polyoxymethylene (POM). Pour cela, nous avons découpé à l'aide d'emporte-pièces des disques de polystyrène de taille variés. Pour le POM, nous avons utilisé une découpe laser.

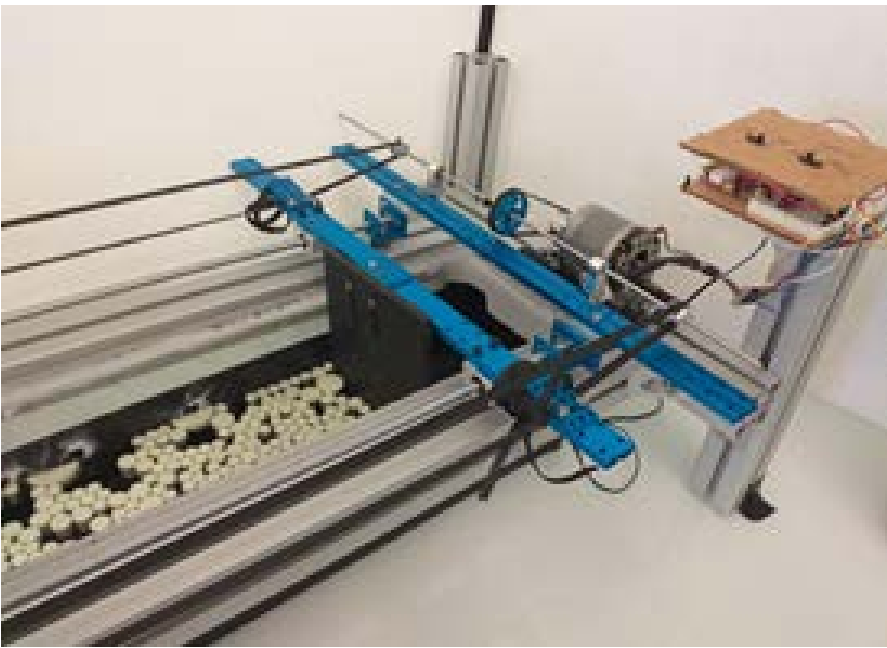


Figure 1. Photo du montage expérimental



Figure 2. Photo des flotteurs

MATERIAU	DIAMETRE (MM)	ÉPAISSEUR (MM)	QUANTITE
PS	10	5	300
PS	16	5	80
PS	18	5	20
PS	24	5	5
POM	5,8	15,5	300
POM	11	15,5	80
POM	12,4	15,5	20
POM	16,4	15,5	5
PS (OBSTACLES)	30	5	5
PS (OBSTACLES)	35	5	5
PS (OBSTACLES)	40	5	5

Figure 3. Dimensions et polydispersité des disques

1.3. Construction de la cuve

Les flotteurs sont ensuite placés dans un fjord représenté par une cuve en Plexiglas remplie d'eau de dimensions: 100cm x 14,5cm x 14,5cm (cf.annexe). Il s'agit d'une cuve récupérée dans les locaux de PSE. La cuve présentant quelques problèmes d'étanchéité, nous l'avons renforcée avec de la colle et des scotchs imperméables. Le fond de la cuve est tapissé de scotch noir, ce qui permet une meilleure identification des flotteurs lors du traitement d'image.

1.4. Mise en place dispositif expérimental

L'avancée du glacier par phénomène de calving est modélisée par l'avancée à vitesse constante d'une petite plaque partiellement immergée qui pousse les flotteurs. Des obstacles fixes sont collés au pistolet à colle sur les bords de la cuve pour modéliser l'irrégularité des rives d'un fjord naturel. L'acquisition des images se fait par une caméra fixée au dessus du montage. Celle-ci est connectée à l'ordinateur en USB et contrôlée avec le logiciel Pylon Viewer. Le contraste et la luminosité sont réglés de façon à obtenir l'acquisition la plus nette possible en noir et blanc.Au début de chaque série de mesure, nous réalisons l'image d'une planche blanche mate placée dans la cuve, qui sert de blanc pour le traitement. Enfin, nous lançons le moteur pas à pas et l'acquisition. La vidéo est enregistrée sous forme d'images successives, à 5 images par seconde. Une fois l'acquisition terminée, nous analysons les images par nos codes Matlab (cf. Annexe).

2. ANALYSE DES RÉSULTATS

2.1. PROGRAMMATION MATLAB

2.1.a. Structure des données

Les données nécessaires au bon fonctionnement des codes d'exploitation sont regroupées dans un sous-dossier /Données/foldername du dossier courant Matlab. Ce sous-dossier regroupe la suite d'images de la caméra, l'image de blanc, et un fichier de description de l'expérience.

2.1.b. Nettoyage de l'image

Nous supprimons les inhomogénéités d'éclairage en divisant toutes les images par l'image du blanc réalisée préalablement.

2.1.c. Trajectoire des flotteurs

Nous utilisons la fonction `imfindcircles` de Matlab afin de trouver les coordonnées (x,y) des centres et les rayons de tous nos cercles sur chaque image avec la boucle `parfor` (pour diminuer le temps de calcul). Nous stockons ces données dans une matrice $Nd \times [(x,y,rayon) \times Ni]$. Nd est le nombre total de disques. Ni est le nombre total d'images lors de l'acquisition. À chaque image, les données renvoyées par la fonction `imfindcircles` sont triées en fonction d'un score d'autant plus grand que le flotteur détecté est circulaire. On cherche ensuite à reconstituer les trajectoires de nos flotteurs. Se donnant un disque D dans l'image k , on cherche le disque D' qui en est le plus proche dans l'image $k+1$ (fonction `knnsearch`). Les déplacements entre deux images étant inférieurs aux rayons des flotteurs, on en déduit que D et D' sont le même disque qui s'est translaté entre les deux images. On reconstitue ainsi les trajectoires complètes.

2.1.c. Champ de vecteurs vitesses

Nous calculons ensuite les vecteurs de déplacement des disques entre 5 images. Les déplacements étant lents, nous les assimilons les vecteurs déplacement aux vecteurs vitesses des disques. Nous traçons ensuite sur Matlab les vecteurs vitesses au cours du temps. Nous superposons enfin l'évolution du champ de vecteur vitesse ainsi calculé à la vidéo initiale. Nous observons bien le phénomène de blocage-déblocage.

2.1.d. Histogramme des vitesses

Nous représentons la composante longitudinale de la vitesse en fonction de la position latérale des flotteurs dans la cuve. Ce choix est motivé par analogies avec les profils d'écoulement de Poiseuille observé dans les fluides visqueux. Pour cela, on regroupe les disques selon leur position latérale dans la cuve avec un pas d'environ 6 mm. On calcule moyenne et écart-type pour chacun des groupes. On trace ces grandeurs en fonction de la position latérale.

2.1.e. Evolution de la moyenne des vitesses (alternance de bandes et de pics)

Nous montrons d'abord que les phénomènes de blocage-déblocage concernent l'ensemble des disques. Pour cela, nous traçons une matrice d'évolution des normes des vitesses des disques. Chaque ligne représentant l'évolution temporelle de la norme de la vitesse. Ces lignes sont triées selon la position initiale des disques croissant. On observe des bandes colorées verticales, qui montrent que les variations de vitesses se font de manière concertée, simultanément dans tout la cuve. Cela justifie la pertinence de la moyenne de la norme de la vitesse pour décrire les phénomènes de blocages-déblocages.

2.1.f. Diagramme densité

L'objectif du diagramme densité est de représenter sur une unique image le déroulement de l'écoulement granulaire. Pour chaque photographie du milieu, on recrée une image binaire du milieu à partir des données de position et de rayon des disques issues de `imfindcircle`. On calcule alors la densité linéique de disque. Pour une position x dans la longueur de la cuve, la densité linéique de disques est le rapport de l'aire occupée par les disques dans la tranche en x sur l'aire totale de la tranche. Le calcul est en pratique effectué sur une longueur de 10 pixels dans la direction x , et sur toute la largeur de la cuve. La grandeur résultante est donc une moyenne glissante sur 10 pixels, qu'on représente par une ligne dans une échelle de couleur. L'empilement des ces lignes forme un diagramme densité (x,t) qui résume la mise en compression et les blocages/déblocages du milieu

2.2. EXPÉRIENCES AVEC DES PARAMÈTRES VARIÉS

Nous cherchons à modéliser ce qui se passe dans les différents fjords naturels. Différents paramètres peuvent donc être variés dans l'expérience. Nous avons notamment fait varier le nombre de flotteurs, leur polydispersité mais aussi la taille et le nombre des obstacles. Chaque expérience a été reproduite 3 fois afin pour s'assurer une possibilité de comparaison a posteriori en cas de résultats étonnants.

-Variation du nombre de flotteurs: 106, 205, 239 flotteurs.

-Variation de la polydispersité des flotteurs: flotteurs de toutes les tailles, flotteurs de 10 mm de diamètre uniquement, flotteurs de $D=24$ mm et $D=18$ mm uniquement.

-Variation de la taille des obstacles fixes: $D=40$ mm, $D=35$ mm, $D=30$ mm, 3 obstacles collées de façon à former une porte et donc réduire de fjord.

-Variation du nombre des obstacles fixes: 5 obstacles de $D=40$ mm, 3 obstacles de $D=40$ mm.

2.3. AUTRES EXPÉRIENCES ET LIMITES

Se donnant un unique obstacle dans la cuve, on cherche à évaluer une distance caractéristique d'influence de cet obstacle sur le milieu. On pourrait imaginer comparer cette distance caractéristique aux distances entre les obstacles, observer son évolution avec la taille de l'obstacle. Considérant les trajectoires des disques en bord de cuve, on cherche à modéliser les trajectoire des disques par une fonction `trajectoire_ideale(a,b,c,d,R,th)`, qui correspond à un arc de cercle de centre (a,b), de rayon R et d'angle th , raccordé de manière continue à deux demi-droites de pentes c et d . On sélectionne les trajectoires et on réalise pour chacun un fit par la fonction `trajectoire_ideale`. Nous avons pour idée de tirer la distance caractéristique recherchée de la courbe d'évolution de la courbure de la trajectoire au niveau de l'obstacle. Toutefois le modèle `trajectoire_ideale` ainsi mis en place n'est valide que pour les disques en bord de paroi. Au centre du milieu, les trajectoires sont trop éloignées du modèle pour que la procédure ait un sens. Faute de temps, nous n'avons pas pu aller plus loin dans la détermination de cette distance caractéristique d'influence.

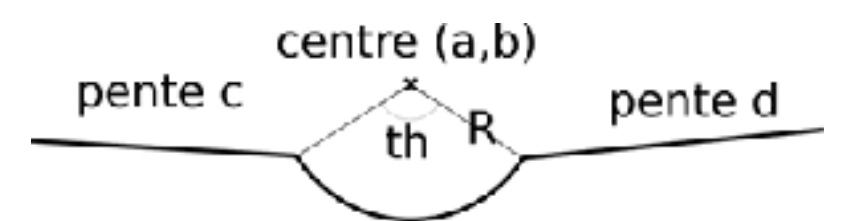


Figure 4. Schéma d'une courbe d'évolution de la courbure de la trajectoire au niveau de l'obstacle.

3. BIBLIOGRAPHIE

4.1. ARTICLE DE DÉPART

[1] Quantifying flow and stress in ice mélange, the world's largest granular material

Justin C. Burton, Jason M. Amundson, Ryan Cassotto, Chin-Chang Kuo, and Michael

Dennin <https://www.pnas.org/content/115/20/5105.short>

4.2. ARDUINO

[1] Vidéo youtube :

28BYJ-48 stepper motor and ULN2003 Arduino (Quick tutorial for beginners)

[2] http://tiptopboards.free.fr/arduino_forum/viewtopic.php?f=2&t=41

[3] <https://retroetgeek.com/geek/arduino/arduino-comment-utiliser-un-moteur-pas-a-pas-uln2003a-et-28byj-48/>

4. ANNEXES

5.1. PROGRAMME DE TRAITEMENT MATLAB

PSE_traitement.m

```
clear all
clc
clf
close all
%%
%Choix de l'acquisition à traiter et importation des infos
foldername='2020-01-22_acq1'
dirdata=dir(sprintf('Données/%s', foldername));
fullname=dirdata(3).name;
fileFormat=fullname(1:length(fullname)-6);
index = 1;
%%
% Image pour faire le blanc :
FB=imread(sprintf('G:/MATLAB/Données/%s/blanc.tiff', foldername));
[imageSizeY, imageSizeX] = size(FB);
% Image initialisation pour compter le nombre de disques:
A=imread(sprintf('G:/MATLAB/Données/%s/%s%s.tiff', foldername, fileFormat,
num2str(index)));
DIV=uint8((double(A)./double(FB))*255);

[centers, radii] = imfindcircles(DIV, [6 30], 'Sensitivity', 0.85);

Nd=length(radii);
Ni=size(dir(sprintf('G:/MATLAB/Données/%s', foldername)), 1)-6; %Nombre
d'images max
%%
%Initialisation des tableaux
data=zeros(Nd,3*Ni);
tracks=zeros(Nd,3*Ni);
X=zeros(Nd,Ni);
Y=zeros(Nd,Ni);
R=zeros(Nd,Ni);
%%
%Lecture des images et extraction des disques. Calcul parallélisé.
parfor i=1:Ni
    A=imread(sprintf('G:/MATLAB/Données/%s/%s%s.tiff', foldername,
fileFormat, num2str(index+i-1)));
    DIV=uint8((double(A)./double(FB))*255);
    [centers, radii] = imfindcircles(A, [6 30], 'Sensitivity', 0.85);
    Ndi=length(radii)
    if (Ndi>Nd)
        X(:,i)=[centers(1:Nd,1)];
        Y(:,i)=imageSizeY-[centers(1:Nd,2)];
        R(:,i)=[radii(1:Nd,1)];
    else
        X(:,i)=[centers(:,1);zeros(Nd-Ndi,1)];
        Y(:,i)=imageSizeY-[centers(:,2);zeros(Nd-Ndi,1)];
        R(:,i)=[radii(:,1);zeros(Nd-Ndi,1)];
    end
end
end
%Remplissage de la matrice data avec X, Y et R
data(:,[1:3:3*Ni])=X;
data(:,1+[1:3:3*Ni])=Y;
data(:,2+[1:3:3*Ni])=R;

%%
%Reconstitution des trajectoires en recherchant le plus proche voisin dans
%l'image suivante
fprintf('Center detecting finished - Start tracking\n')
tracks(:,1:3)=data(:,1:3);
```

```

for i=1:(Ni-1)
    IDX=knnsearch(data(:,3*(i+1)-2:3*(i+1)-1), tracks(:,3*(i)-2:3*(i)-1));
    a=[data(:,3*(i+1)-2:3*(i+1))];
    tracks(:,3*(i+1)-2:3*(i+1))=a(IDX,:);
end
fprintf('Done\n')
%%
%Sauvegarde des données produites par le programme dans un dossier dédié.
mkdir(strcat("Traitement/", foldername));
save(strcat("Traitement/", foldername, "/savedata_", foldername), 'tracks',
'imageSizeX', 'imageSizeY', 'foldername', 'fileFormat', 'Ni', 'Nd')

```

densite.m

```

%clear all
%clc
clf
close all
%%
%Importation des données issues de PSE_traitement
%foldername='2020-02-07_acq1'
load(sprintf('Traitement/%s/savedata_%s.mat', foldername, foldername))
infos=fileread(strcat('Données/', foldername, '/infos.txt'))

X=tracks(:,[1:3:3*Ni]);
Y=tracks(:,1+[1:3:3*Ni]);
R=tracks(:,2+[1:3:3*Ni]);
%Conversion de l'espace en pixel à l'espace en mm.
X_mm=X*535/imageSizeX;
Y_mm=Y*147/imageSizeY;

[columnsInImage rowsInImage]=meshgrid(1:imageSizeX, 1:imageSizeY);
plage=20; %largeur sur de la plage de calcul de la densité
densiteM=zeros(imageSizeX-plage,Ni);
%Reconstitution de la densité linéaire en disque.
parfor i=1:Ni
    denstemp=zeros(imageSizeX-plage,1);
    B=zeros(imageSizeY,imageSizeX);
    for d=1:Nd
        % Recréation d'une image binaire du milieu à partir des disques
        % extraits dans PSE_traitement.
        B=(rowsInImage-Y(d,i)).^2+(columnsInImage-X(d,i)).^2<=R(d,i).^2|B==1;
    end

    for j=1:imageSizeX-plage
        %Calcul de la densité linéique en un point.
        denstemp(j,1)=sum(sum(B(:,j:j+plage-1)))/(imageSizeY*plage);
    end
    densiteM(:,i)=denstemp;
end
%%
%Affichage de la matrice densité (x,t)
fig1=figure(1);
imagesc(densiteM);
xlabel('x (pixel)')
ylabel('temps (image)')
title(strcat("Evolution de l'occupation par les disques - ",infos))
fig1.Renderer='painter';
set(fig1, 'Position',[30, 30, 1100, 800]);
savefig(strcat("Traitement\_", foldername, '\densite'))
saveas(gcf, strcat("Traitement\_", foldername, '\densite.svg'))
saveas(gcf, strcat("Traitement\_", foldername, '\densite.png'))
save(strcat("Traitement/", foldername, "/savedensite_", foldername), 'densiteM',
'imageSizeX', 'imageSizeY', 'foldername', 'fileFormat', 'Ni', 'Nd')

```

vitesse.m

```
%clear all
%clc
clf
close all
%%
%Importation des données issues de PSE_traitement
%foldername='2020-02-07_acq1'
load(sprintf('Traitement/%s/savedata_%s.mat', foldername, foldername))
dim=size(tracks);
Ni=dim(2)/3-30;
Nd=dim(1);
infos=fileread(strcat('Données/', foldername, '/infos.txt'))
X=tracks(:,[1:3:3*Ni]);
Y=tracks(:,1+[1:3:3*Ni]);
R=tracks(:,2+[1:3:3*Ni]);
X_mm=X*535/imageSizeX;
Y_mm=Y*147/imageSizeY;
NIm=repmat(1:Ni,Nd,1);
%%
% Calcul des vecteurs de déplacement qui seront assimilés aux vecteurs
% vitesse.
VX=[X(:,6:Ni)-X(:,1:(Ni-5)) repmat(zeros(Nd,1),1,5)];
VY=[Y(:,6:Ni)-Y(:,1:(Ni-5)) repmat(zeros(Nd,1),1,5)];
VX_mm_s=VX*535/(1.2*imageSizeX);
VY_mm_s=VY*147/(1.2*imageSizeY);
%%
% Tracé, affichage et sauvegarde des trajectoires
fig1=figure(1);
plot(X_mm', Y_mm', "-")
hold on
I = imread(strcat('Données\',foldername,'\',fileFormat, '1.tiff'));
h = image([0 535],[0 147],flip(I), 'AlphaData', 0.2);
uistack(h, 'bottom')
colormap(gray)
set(fig1, 'Position',[30, 30 , 1500, 500]);
axis equal
xlabel('x (mm)')
ylabel('y (mm)')
xlim([0 535])
ylim([0 147])
title(strcat("Trajectoires - ",infos))
fig1.Renderer='painter';

savefig(strcat("Traitement\ ",foldername,'\trajectoires'))
saveas(fig1,strcat("Traitement\ ",foldername,'\trajectoires.svg'))
saveas(fig1,strcat("Traitement\ ",foldername,'\trajectoires.png'))
saveas(fig1,strcat("Traitement\ ",foldername,'\trajectoires.jpeg'))

%%
%Tracé, affichage et sauvegarde du profil latéral de vitesse longitudinale
%(moyenne et écart-type)
fig4=figure(4);
pasY=15;
VXmoy=ones(floor(imageSizeY/pasY),1);
VXstd=ones(floor(imageSizeY/pasY),1);
for k=1:pasY:imageSizeY
    I=(NIm>90&NIm<Ni-20 & Y>k*ones(Nd,Ni) & Y<(k+pasY)*ones(Nd,Ni));
    VXmoy(1+(k-1)/pasY, 1)=mean(VX(I));
    VXstd(1+(k-1)/pasY, 1)=std(VX(I));
end
VXmoy_mm=VXmoy*535/(1.2*imageSizeX);
VXstd_mm=VXstd*535/(1.2*imageSizeX);
```

```

errorbar(((1:pasY:imageSizeY)+pasY/2*ones(1,
floor(imageSizeY./pasY)))*147/(1.2*imageSizeY), VXmoy_mm, VXstd_mm, 'o')
xlabel('y (mm)')
ylabel('Vx (mm/s)')

title(strcat("Profil de vitesse Vx en fonction de la position y dans la
largeur de la cuve - ", infos))
fig4.Renderer='painter';
set(fig4, 'Position', [30, 30 , 1100, 800]);

savefig(strcat("Traitement\ ", foldername, '\profilbouchon'))
saveas(fig4, strcat("Traitement\ ", foldername, '\profilbouchon.svg'))
saveas(fig4, strcat("Traitement\ ", foldername, '\profilbouchon.png'))
saveas(fig4, strcat("Traitement\ ", foldername, '\profilbouchon.jpeg'))
%%
%Tracé, affichage et sauvegarde de l'évolution de la moyenne de la norme de
la vitesse
fig5=figure(5);
normV=sqrt(VX_mm_s.^2+VY_mm_s.^2);
T=sortrows([X_mm(:, 1) normV], 1); %Tri des disques suivant X_mm initial
graph1=subplot(2,1,1);
imagesc([0 (Ni-6)*0.2],[0 Nd], T(:,2:(Ni-5)), [0 2*mean(mean(normV))])
colorbar()
set(fig5, 'Position', [30, 30 , 1100, 800]);
h=colorbar;
ylabel(h, 'normes des vitesses en mm/s')
ylabel('N° disque (triés par x initial)')
title(strcat("Norme de la vitesse des flotteurs en fonction du temps - ",
infos))
graph2=subplot(2,1,2);
plot([0:0.2:(Ni-6)*0.2], mean(normV(:, 1:(Ni-5))))
axis([0 (Ni-6)*0.2 0 6])

pos1=get(graph1, 'Position');
pos2=get(graph2, 'Position');
pos1(3)=pos2(3);
set(graph1, 'XTickLabel', '')
set(graph1, 'Position', pos1)
pos2(2)=pos1(2)-pos2(4)-0.05;
set(graph2, 'Position', pos2);
ylabel('Moyenne des normes de la vitesse')
xlabel('temps (s)')

hold on
fig5.Renderer='painter';
savefig(strcat("Traitement\ ", foldername, '\normevitesse'))
saveas(fig5, strcat("Traitement\ ", foldername, '\normevitesse.svg'))
saveas(fig5, strcat("Traitement\ ", foldername, '\normevitesse.png'))
saveas(fig5, strcat("Traitement\ ", foldername, '\normevitesse.jpeg'))

%%
%Tracé, affichage et sauvegarde d'une vidéo de l'évolution de la norme de
la vitesse
fig6=figure(6)
set(fig6, 'Position', [30, 30 , 1500, 500]);
k=1;
for i=1:1:Ni
    hold on
    quiver(X_mm(:,i),Y_mm(:,i),2*VX_mm_s(:,i),2*VY_mm_s(:,i),0)
    t=i*0.2;
    title(strcat(sprintf("Champ de vitesse - t=%.2f s - ",t),infos))
    axis equal
    I = imread(strcat('Données\ ',foldername, '\ ',fileFormat,
sprintf('%d',k+1), '.tiff'));
    h = image([0 535],[0 147],flip(I), 'AlphaData', 0.2);
    uistack(h, 'bottom')
    colormap(gray)

```

```

        xlabel('x (mm)')
        ylabel('y (mm)')
        xlim([0 535])
        ylim([0 147])

        pause(0.1)
        %savefig(strcat("G:\PSE\MATLAB\figure rapport\","
        sprintf("%s",fichier),"\",sprintf("vitesses_fig%d",k)))
        %saveas(gcf,strcat("G:\PSE\MATLAB\figure rapport\","
        sprintf("%s",fichier),"\",sprintf("vitesses_fig%d.svg",k)))
        saveas(fig6,
        strcat("Traitement/",foldername,sprintf("/film/vitesses_fig%03d.png",k)))
        k=k+1;
        clf
end

```

track_deviation.m

```

%clear all
%clc
clf
close all
%%
%Importation des données issues de PSE_traitement
%foldername='2020-01-21_acq2'
load(sprintf('Traitement/%s/savedata_%s.mat', foldername, foldername))
infos=fileread(strcat('Données/',foldername,'/infos.txt'))
dim=size(tracks);
Ni=dim(2)/3;
Nd=dim(1);
X=tracks(:,[1:3:3*Ni]);
Y=tracks(:,1+[1:3:3*Ni]);
R=tracks(:,2+[1:3:3*Ni]);
NIm= repmat(1:Ni,Nd,1);
X_mm=X*535/imageSizeX;
Y_mm=Y*147/imageSizeY;
%%
%Sélection des trajectoires comprises au niveau de l'obstacle unique.
J=(sum(X_mm<240,2)>0)&(sum(X_mm>310,2)>0);
Xdev_mm=X_mm(J,:);
Ydev_mm=Y_mm(J,:);

Fitdev=zeros(size(Xdev_mm,1),8);

for i=1:size(Xdev_mm,1)
    %Fit des trajectoires à la fonction trajectoire_ideale.
    [fitresult, gof] = fit_to_traj(Xdev_mm(i,:), Ydev_mm(i,:));
    Fitdev(i,:)=[coeffvalues(fitresult) gof.rsquare mean(Xdev_mm(i),2)];
end
%%
% Tracé, affichage et sauvegarde du rayon de courbure au niveau de
% l'obstacle en fonction de la position dans la largeur de la cuve.
figure()
K=Fitdev(:,7)>0.8; %Sélection des fits de haut r^2.
plot(Fitdev(K,8),1./Fitdev(K,1),'o');
xlabel("xmoyen")
ylabel("Rayon de courbure")

```

trajectoire_ideale.m

```
function y = trajectoire_ideale(x,a0,b0,c0,d0,R0,th0)
y = zeros(size(x));
for i = 1:length(x)
    a=250+2*(a0-0.5)*500;
    b=75+2*(b0-0.5)*150;
    c=-c0*0.1;
    d=d0*0.1;
    R=R0*50;
    th=th0*pi/2;
    t=R0.*sin(th);
    if x(i) <= a-R.*sin(th)
        y(i) = b-R.*cos(th) + c.*(x(i)-a+t);
    elseif x(i) >= a+R.*sin(th)
        y(i) = b-R.*cos(th) + d.*(x(i)-a-t);
    else
        y(i) = b-R.*cos(asin((x(i)-a)./R));
    end
end
end
```

fit_to_traj.m

```
function [fitresult, gof] = fit_to_traj(dataX, dataY)
% Data for 'untitled fit 1' fit:
%   X Input : dataX
%   Y Output: dataY
% Output:
%   fitresult : a fit object representing the fit.
%   gof : structure with goodness-of fit info.
[xData, yData] = prepareCurveData( dataX, dataY );

% Set up fittype and options.
ft = fittype( 'trajectoire_ideale(x,a0,b0,c0,d0,R0,th0)', 'independent',
'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares', 'Display', 'Off');
opts.Display = 'Off';
opts.Lower = [0.2 0.1 0.1 -0.7 -0.7 0.1];
opts.StartPoint = [9 mean(dataX)/500 4 0.488608973803579 0.5785
0.237283579771521];
opts.Upper = [30 0.9 5 0.7 0.7 1];

% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft, opts );
grid on
```

4.2. CODE ARDUINO

Code de contrôle du moteur pas à pas

```
int pin1=2;
int pin2=3;
int pin3=4;
int pin4=5;
int inter=8;//interrupteur marche/arrêt
int sens=9;//interrupteur de sens
int vit = 20;//vitesse avant
int vitar =40;//vitesse arrière

void setup() {
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(13, OUTPUT); //led d'indication de fonctionnement
  pinMode(inter, INPUT_PULLUP);
  pinMode(sens, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(inter) == LOW){
    if (digitalRead(sens) == HIGH){
      digitalWrite(13,HIGH);

//Si l'interrupteur de sens est en position on, le moteur tourne dans un
sens
      //Etape1
      digitalWrite(pin1,LOW);
      digitalWrite(pin2,HIGH);
      digitalWrite(pin3,LOW);
      digitalWrite(pin4,HIGH);
      delay(vit);
      //Etape2
      digitalWrite(pin1,LOW);
      digitalWrite(pin2,HIGH);
      digitalWrite(pin3,HIGH);
      digitalWrite(pin4,LOW);
      delay(vit);
      //Etape3
      digitalWrite(pin1,HIGH);
      digitalWrite(pin2,LOW);
      digitalWrite(pin3,HIGH);
      digitalWrite(pin4,LOW);
      delay(vit);
      //Etape4
      digitalWrite(pin1,HIGH);
      digitalWrite(pin2,LOW);
      digitalWrite(pin3,LOW);
      digitalWrite(pin4,HIGH);
      delay(vit);
      digitalWrite(13,LOW);
    }

    else if (digitalRead(sens) == LOW){

//Si l'interrupteur de sens est en position off, le moteur tourne dans un
sens
      //Etape4
      digitalWrite(pin1,HIGH);
      digitalWrite(pin2,LOW);
      digitalWrite(pin3,LOW);
```

```

digitalWrite(pin4,HIGH);
delay(vitar);
//Etape3
digitalWrite(pin1,HIGH);
digitalWrite(pin2,LOW);
digitalWrite(pin3,HIGH);
digitalWrite(pin4,LOW);
delay(vitar);
//Etape2
digitalWrite(pin1,LOW);
digitalWrite(pin2,HIGH);
digitalWrite(pin3,HIGH);
digitalWrite(pin4,LOW);
delay(vitar);
//Etape1
digitalWrite(pin1,LOW);
digitalWrite(pin2,HIGH);
digitalWrite(pin3,LOW);
digitalWrite(pin4,HIGH);
delay(vitar);

digitalWrite(13,LOW);
if (digitalRead(inter) == HIGH){
    digitalWrite(13,LOW);
}
}
}

```

4.3. PLAN DES MONTAGES

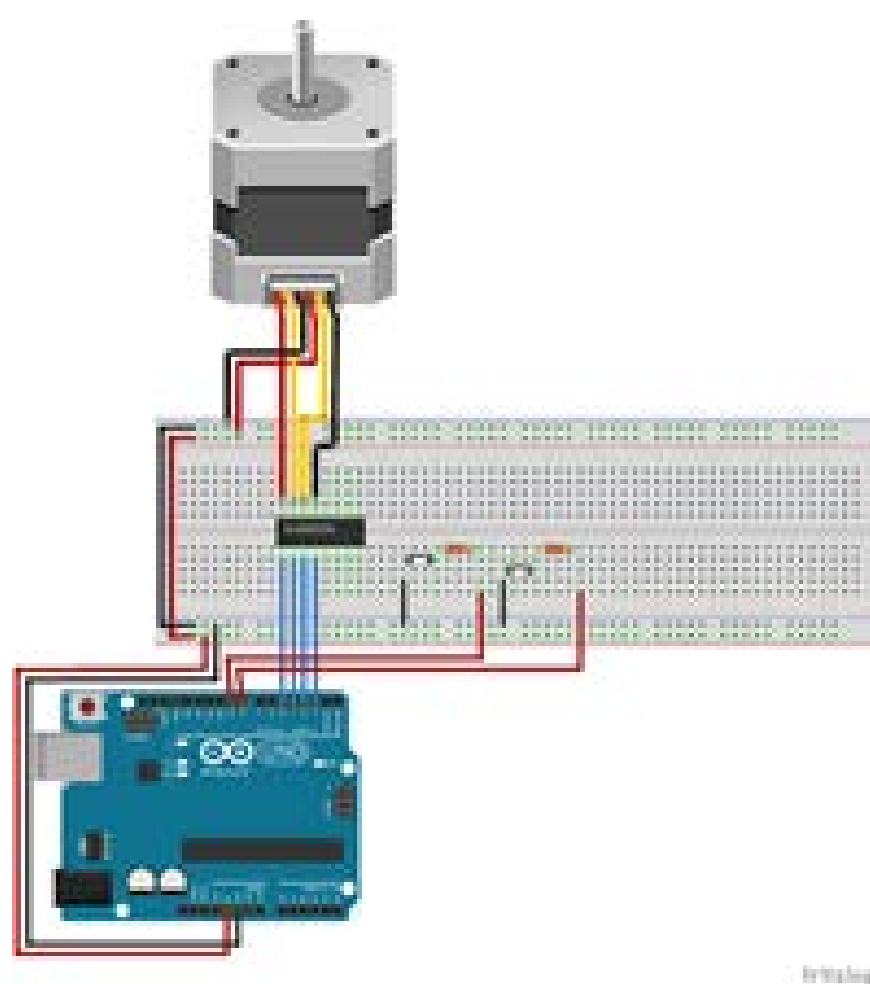


Figure 1. Schéma Montage Moteur Arduino

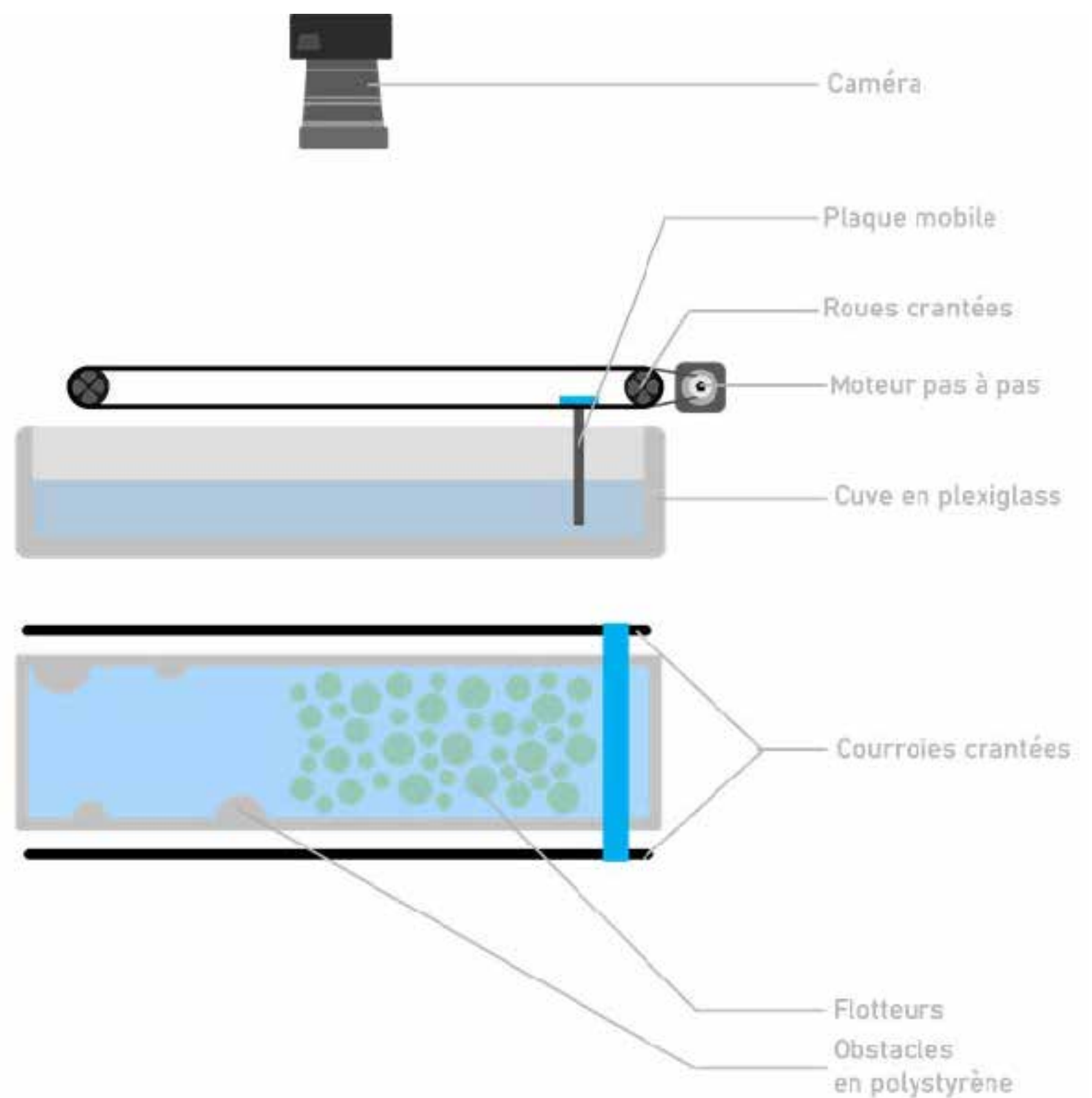


Figure 2. Schéma Montage Experimental

5. FIGURES

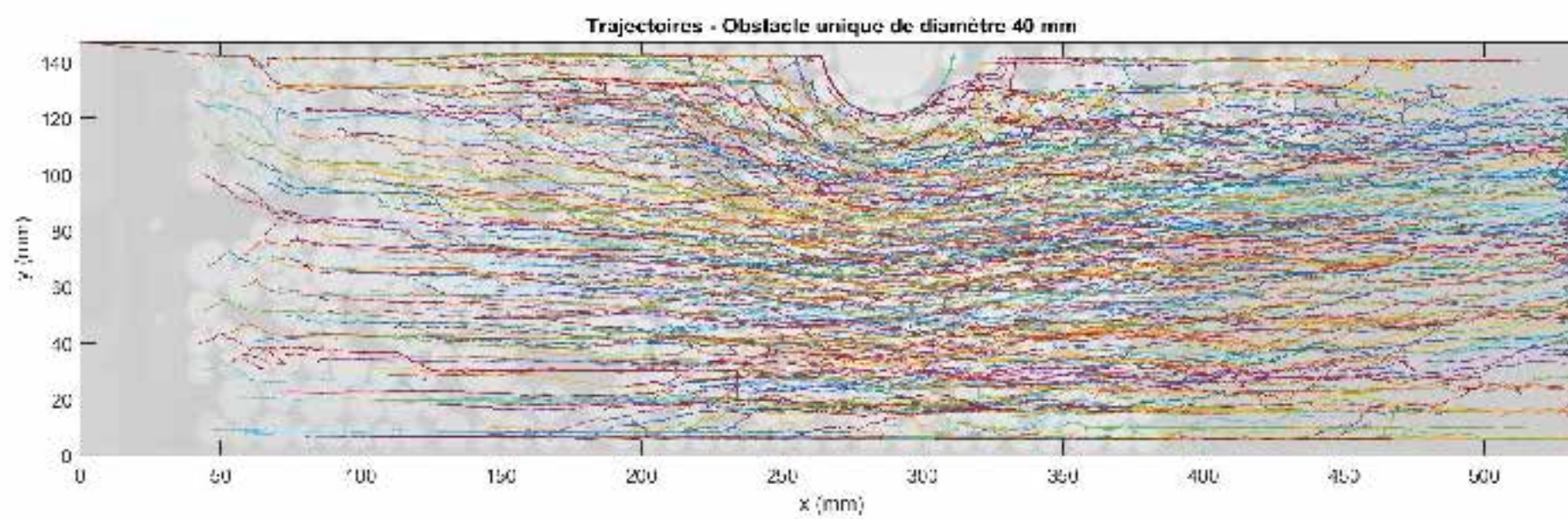


Figure 1. Trajectoire de tous les flotteurs, avec un obstacle de 40mm de diamètre tracé avec Matlab

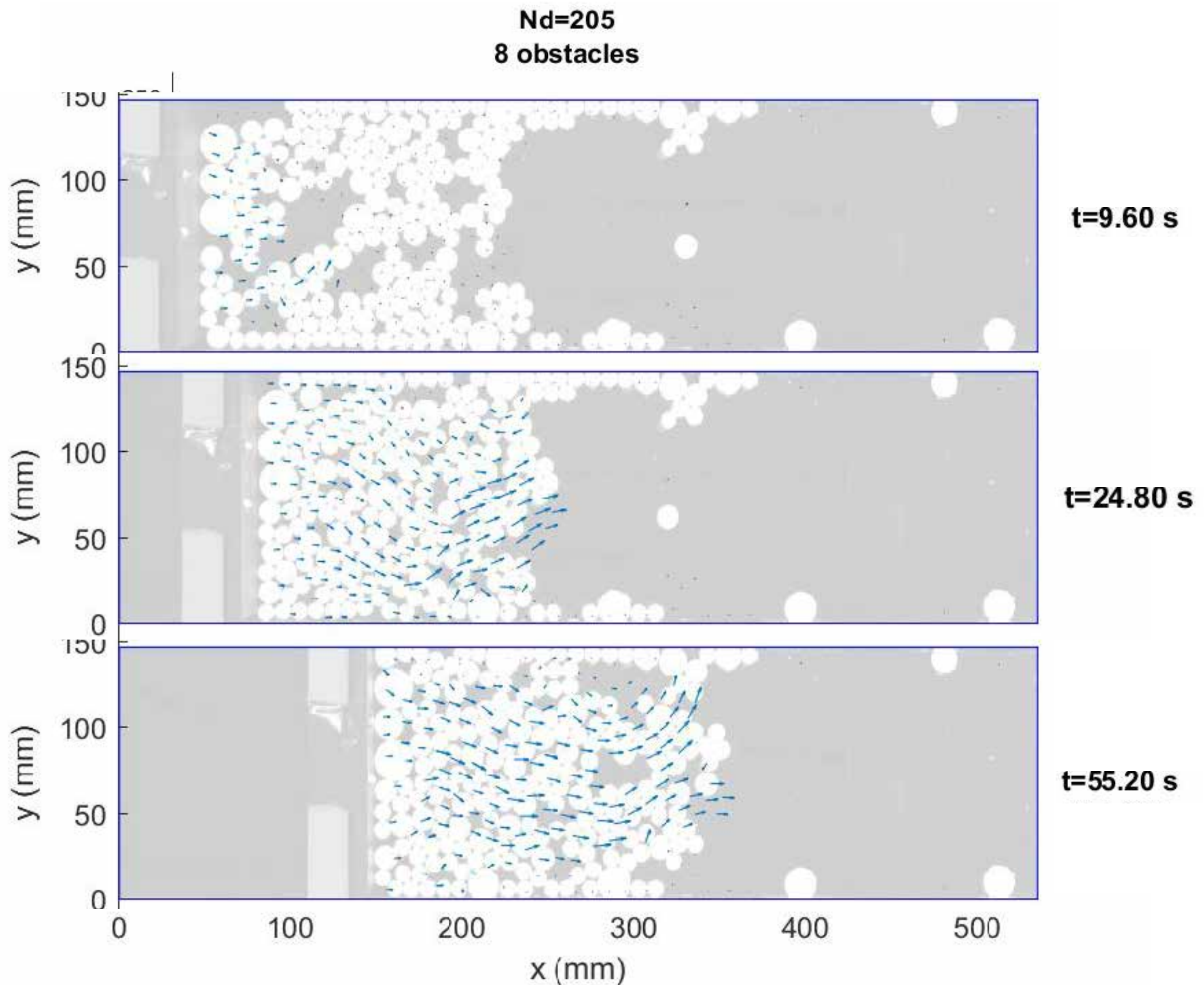


Figure 2. Champ de vecteur vitesse au cours du temps avec 8 obstacles et 205 flotteurs.

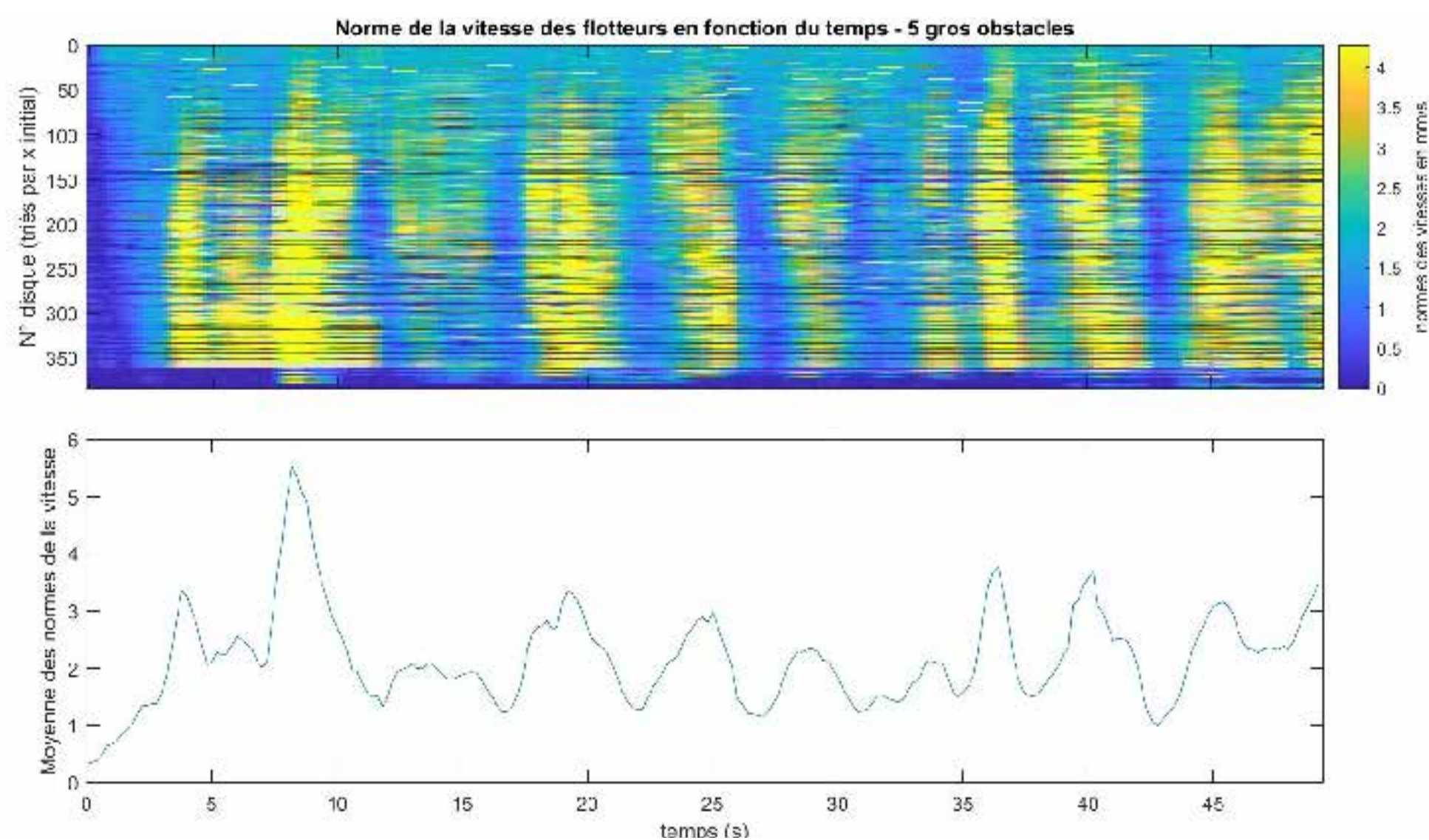
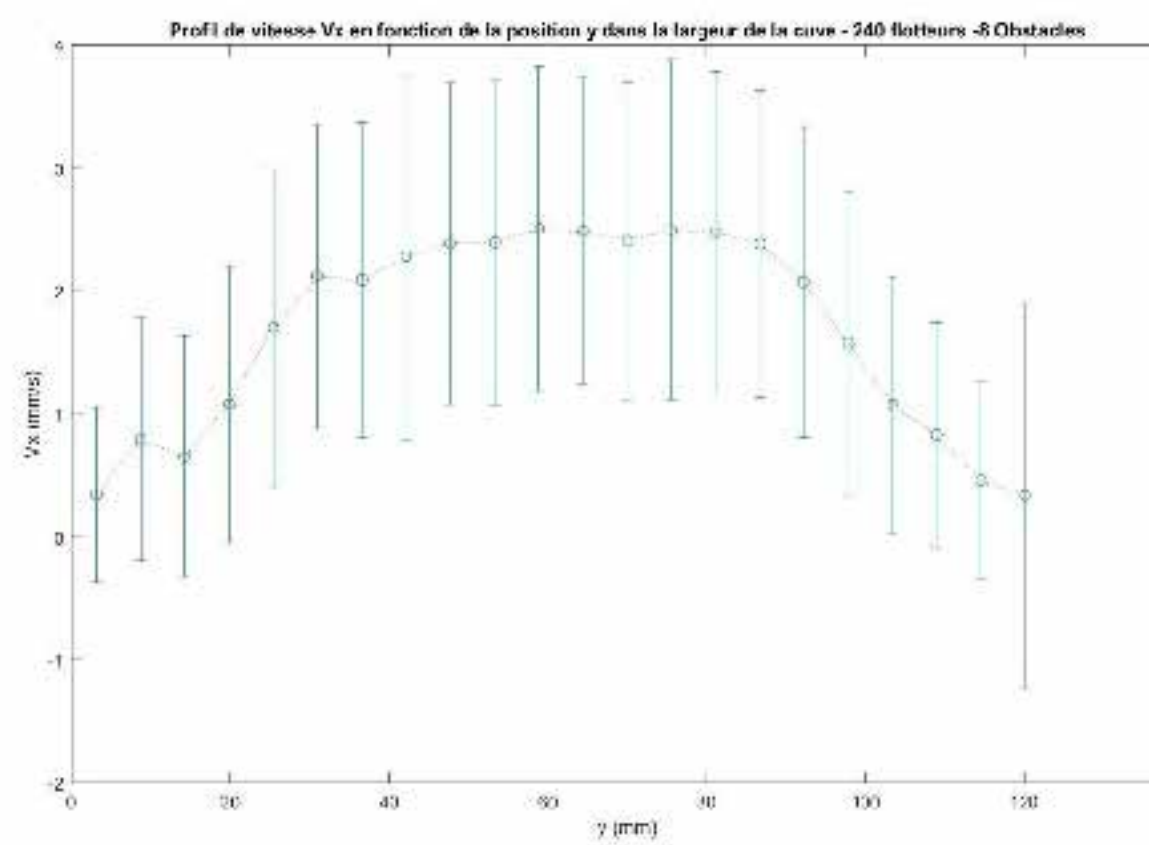
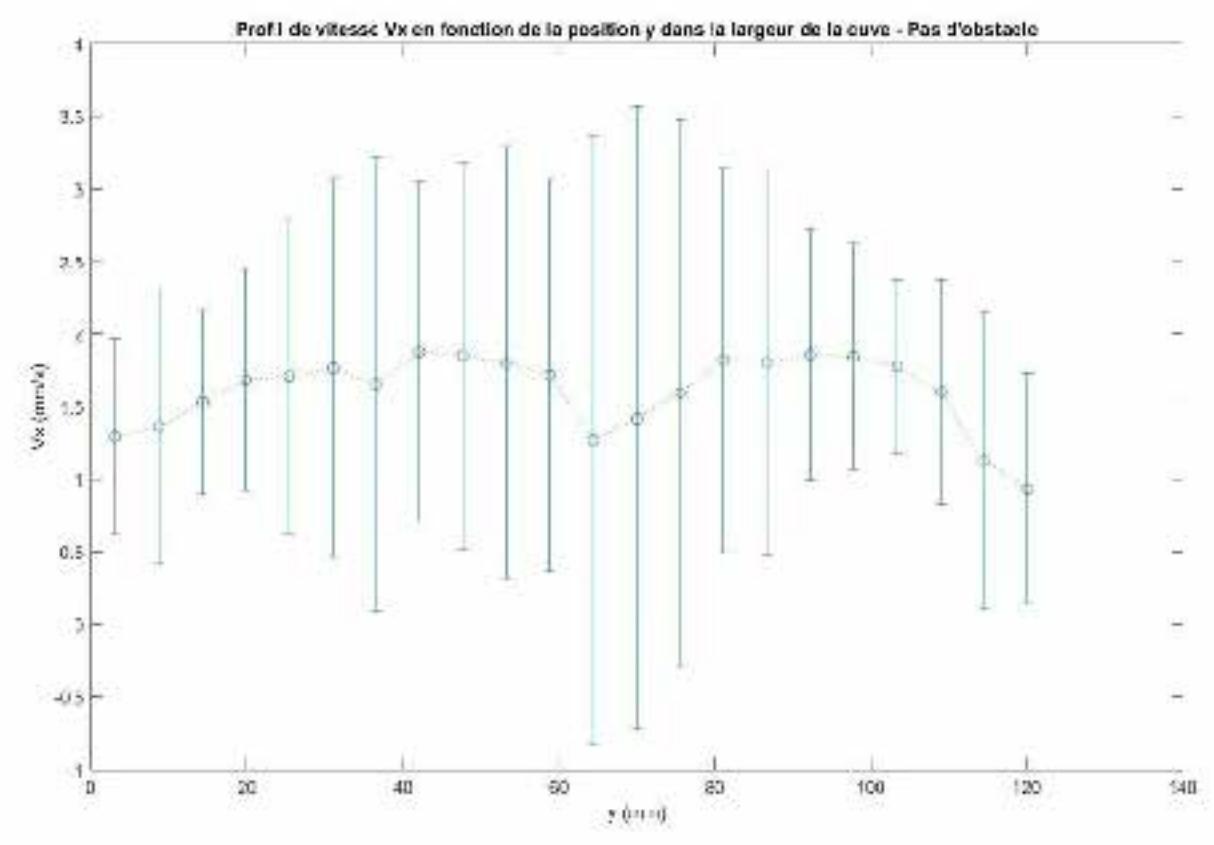


Figure 3. Evolution de la moyenne de la norme des vitesses des flotteurs en fonction du temps

Dans la zone du haut du diagramme (x entre 0 et 50 px), la plaque impose une vitesse constante aux flotteurs qui en sont le plus proche. Lorsqu'on s'en éloigne on observe une alternance de périodes d'arrêt des disques et de plus grande vitesse ce sont les phénomènes de blocage déblocage. Ce sont les mêmes pour tous les disques : il agit d'un mouvement d'ensemble et non pas de zones isolées.



a)



b)

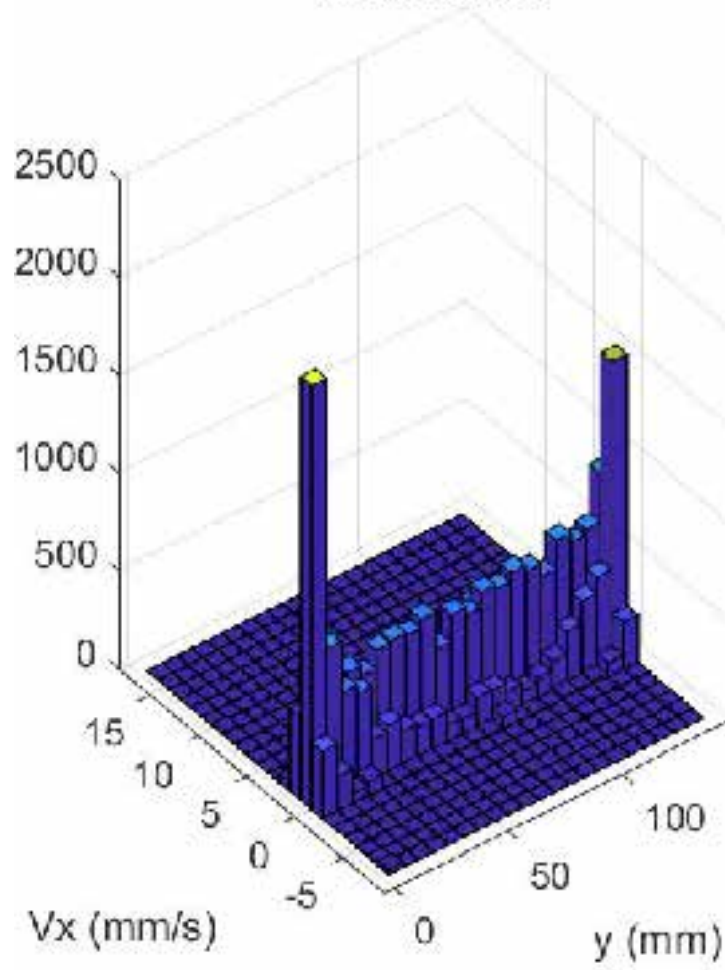
Figure 3. Distribution des vitesses sur la largeur de la cuve (a) avec 8 obstacles (b) sans obstacles

Lorsqu'il n'y a pas d'obstacle, le milieu avance d'un seul bloc. Lorsque les obstacles sont présents, la vitesse est plus faible sur les bords de la cuve, où les flotteurs sont retenus.

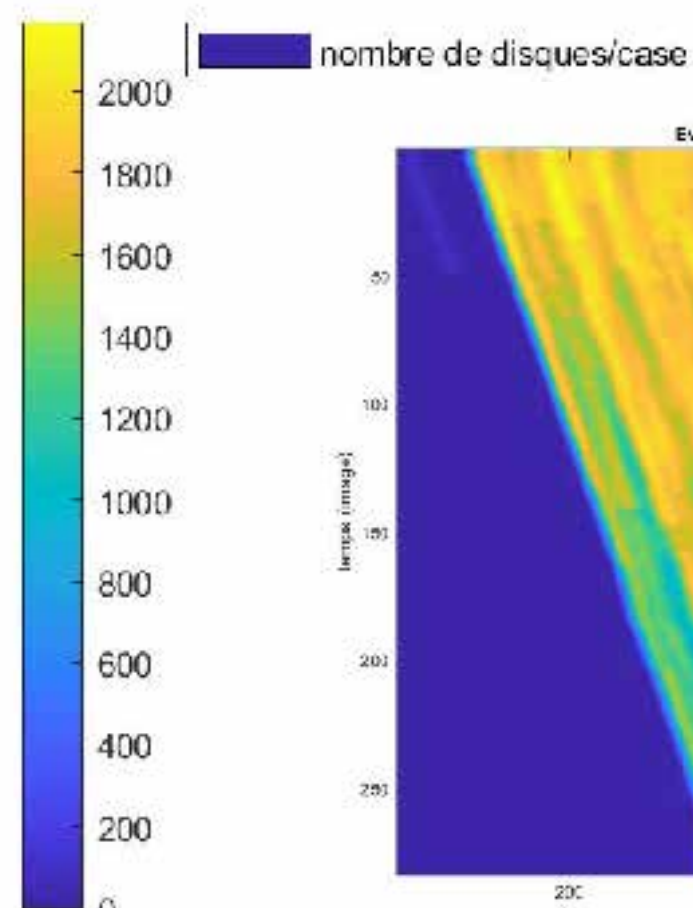
Histogramme de la vitesse V_x par rapport à la position y

Nd=205

8 obstacles



a)



b)

Figure 4. Histogramme de la vitesse V_x par rapport à la position y avec 8 obstacles (a) et Diagramme densité représentant la répartition des disques à chaque coordonnée et à chaque instant (b)

