Méthodes et Pratiques
PSE : Capture de l'énergie des vagues

José Bonet Faus, Gabrielle Di Mauro, Éloi Vincent

Mai 2020

1 Introduction

La récupération de l'énergie houlomotrice est un enjeu environnemental de plus en plus important. Optimiser la récupération de cette énergie renouvelable permettrait de répondre en partie aux besoins énergétiques de la planète. Cet article présente les matériels et méthodes pour récupérer l'énergie produite par la houle, générée dans une cuve remplie de $20cm$ d'eau. La houle est générée par un batteur vertical, mis en mouvement par un vérin électrique piloté grâce à l'interface **Arduino**. La fréquence de la houle est maîtrisée et les ondes produites sont focalisées grâce à une lentille concave.

2 Montage : la génération de vagues

Le montage expérimental de ce projet est réalisé dans une cuve en plexiglas de dimensions $70 \times 85 \times 130cm$. Dans le fond de cette cuve est placé un quadrillage de dimensions $60 \times 85cm$, composé de carrés de $0.5cm$ alternativement noirs et blancs. Il faut que la taille des carrés soit petite devant la longueur d'onde λ des vagues et grande devant la capacité de résolution de la caméra pour permettre de traiter les données par imagerie Schlieren, une méthode qu'on expliquera en partie 3.
Dans cette cuve se trouve un vérin commandé par un module **ControlinoMini**, alimenté par un générateur de tension continue à 24V. Au bout du vérin on a installé un batteur, c'est la partie mobile qui va plonger dans l'eau pour générer des vagues. Sur la durée de notre projet, on utilise deux types de batteurs différents :

1. un batteur plat venant incider sur la surface de l'eau à 30^o et de dimensions $60 \times 15cm$.
2. un batteur circulaire de dimension de longueur $60cm$ et avec un rayon de courbure de $30cm$.

Les éléments du système d'action du batteur sont les suivants :
— Vérin électrique SMC 43N Tige filetée, 24V cc, 100mm/s, course 100mm, ref : LDZBL3M-100A3L
— Câble SMC, longueur 2m, à utiliser avec Série CN3, ref : LC3F2-1-C3-02-1
— Câble SMC, longueur 2m, à utiliser avec Série CN2, ref : LC3F2-1-C2-02-1
— Câble d'alimentation SMC, longueur 2m, à utiliser avec Série CN1, ref : LC3F2-1-C1-02-1
— Contrôleur de moteur c.c. Monophasé, 24 V c.c., 2,3 A, ref : LC3F212-5A5B

Pour piloter le batteur nous avons utilisé l'interface **Arduino** et pour traiter numériquement les données récupérées, on a utilisé le logiciel **MatlabR2018a** ainsi que le logiciel **ImageJ**. La récupération d'images se fait avec une caméra BASLER réf acA 1300-200uc munie d'un objectif de focale $1m$. La caméra est placée au dessus de la cuve à environ $2m$ de la surface de l'eau. Un schéma du montage est présent en *Figure* 1

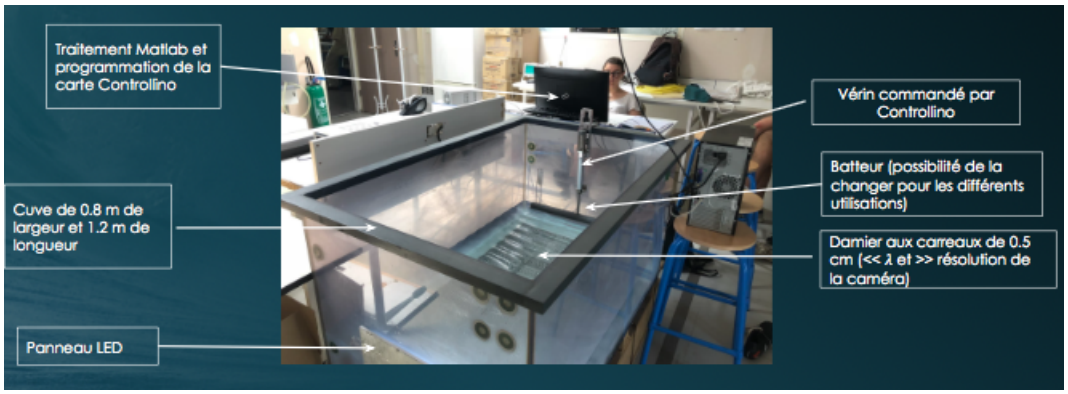

FIGURE 1 – Montage expérimental pour la génération de la houle

3 Protocole de mesure du champ des hauteurs

Dans cette première étude, la profondeur d'eau reste identique. Le code **Arduino** nous permet d'imposer la fréquence à laquelle le batteur tape sur la surface de l'eau. On génère ainsi des vagues de fréquence connue. De plus, sa course est également réglable, on peut donc contrôler l'amplitude des vagues produites. Dans un premier temps, on choisit ces paramètres arbitrairement de sorte à avoir des vagues bien visibles et d'amplitude considérable. Plus concrètement, c'est le paramètre *delay* de notre code **Arduino** que l'on fait varier pour contrôler la fréquence des vagues. Pour obtenir des vagues idéales, on s'est placé au maximum de la course du vérin et à une fréquence correspondant à $delay = 160ms$.

3.1 Analyse numérique par la méthode de Schlieren

Une fois qu'on a généré des vagues grâce à notre code **Arduino**, on va essayer de les analyser ; on utilise pour cela la méthode de Schlieren [1]. Cette méthode se base sur la déformation d'un motif lors du passage de l'onde, c'est pour cela que nous avons placé un quadrillage au fond de la cuve. Le principe de la méthode est le suivant : la déformation du quadrillage permet de retrouver la pente provoquée par chaque onde sur la surface de l'eau, on accède alors à la forme du champ des hauteurs par transformée de Fourier.

Le code principal permettant l'analyse de la houle se trouve en Annexe B. Il a été modifié pour ce projet, mais notre code fait quand même appel à des fonctions auxiliaires disponibles sur le site suivant : (**https** : //**github.com/swildman/fcd**).

Les images obtenues sont ensuite filtrées afin d'améliorer leur qualité. Il faut appliquer un filtre passe-bas pour éliminer le bruitage de haute fréquence puis un filtre passe haut pour retirer les longues oscillations non désirées. Le code est en Annexe B (de la ligne 1 à 62 pour le traitement

des images et de la ligne 64 à 100 pour le filtrage) et en Annexe C, la fonction filtrage. Ce programme qui donne le champs de hauteurs filtré sera par la suite appelé "analyse-vagues".

3.2 Application pour la relation de dispersion

Avant de continuer, il faut vérifier que les vagues générées respectent bien la relation de dispersion classique de la houle. Pour trouver la relation de dispersion expérimentale deux méthodes peuvent être utilisées se basant toutes deux sur la déformation du quadrillage.

3.2.1 Protocole pour trouver la relation de dispersion à l'aide de la méthode de Schlieren

La première option consiste à utiliser la méthode de Schlieren pour vérifier la relation de dispersion. Dans un premier temps, on génère dans la cuve des ondes de fréquences très différentes qui seront traitées par le programme "analyse-vagues". Dans un second temps, nous allons faire la transformée de Fourier spatiale du champ de hauteurs obtenu. Ceci nous permet d'avoir accès aux fréquences des vagues et aux longueurs d'onde correspondantes, on a donc la relation de dispersion. Il en faut ensuite tracer dans un même graphique la relation de dispersion expérimentale que l'on vient d'obtenir et la théorique pour les comparer.

3.2.2 Protocole pour trouver la relation de dispersion à l'aide d'Image J

Cette méthode n'utilise pas la méthode de Schlieren ni de traitement Matlab, mais le logiciel **ImageJ** et le code *Arduino*. Durant l'expérience, on règle le paramètre *delay* du vérin , ce qui nous permet de générer des vagues de fréquences connues. Ensuite, à l'aide du logiciel **ImageJ** on peut repérer les fronts d'onde de nos vagues (visibles grâce à la déformation du quadrillage) et trouver la longueur d'onde correspondant à chaque fréquence. Les longueurs obtenues seront en pixels, c'est pour cela qu'il est crucial de ne pas oublier, au préalable, de savoir à quoi équivaut en mètre la taille d'un pixel. Une fois la liste des fréquences et longueurs d'onde obtenues, celles des pulsations et des nombres d'ondes s'en déduient facilement et la relation de dispersion peut être tracée et comparée à la relation de dispersion théorique.

4 Protocole pour la récupération de l'énergie des vagues

Une fois que nous avons la relation de dispersion, on se place dans son domaine de validité entre $3Hz$ et $12Hz$. Dans ce domaine là les vagues ont bien le comportement attendu, on est donc prêts à essayer de récupérer l'énergie de cette houle. Cependant, l'énergie de la houle est répartie sur tout le front d'onde, c'est à dire que notre système de récupération d'énergie devrait avoir des dimensions assez importantes pour limiter les pertes. Mais un système de grandes dimensions est encombrant, c'est pour cela que nous allons essayer de concentrer l'énergie en un point.

4.1 Focalisation

L'objectif de la focalisation est de faire converger les ondes en un même point pour augmenter l'amplitude et ainsi diminuer les pertes. La focalisation des ondes acoustiques marche selon le même principe que celles des ondes lumineuses [2], des lentilles concaves peuvent donc être utilisées. On utilise souvent des lentilles déformables en polymères qui flottent à la surface de l'eau, entre le batteur et le système de récupération d'énergie. Malheureusement, pour que la focalisation soit utile dans le contexte de notre expérience, il faut que le point focal ne soit pas

très proche des limites de la cuve. C'est pour cela qu'une véritable lentille avec un film épousant la forme des vagues (comme dans l'article [2]) ne peut pas être utilisée dans notre montage. On utilise alors le batteur circulaire décrit plus haut, qui génère un front d'ondes convergeant en un point, il simulera une focalisation.

L'amplitude des vagues au point focal dépend notamment de la fréquence d'excitation du batteur mais de nombreux paramètres rentrent en jeu. Pour observer et quantifier les résultats de la focalisation, on utilise l'imagerie Schlieren. Le code "analyse-vague" couplé à un filtre passe-bas (cf Annexe B) nous permet d'obtenir le champ des hauteurs. Si on réalise la transformée de Fourier de ce champ, on observe nettement un maximum d'amplitude ou niveau d'un point à $30cm$ du batteur (ce qui correspond au rayon de courbure) : c'est la tâche focale. Une coupe du champ des hauteurs nous montre que la tâche est en fait une tâche d'Airy, due à la diffraction.

On reprend alors cette expérience pour une gamme de fréquences de la houle, afin d'évaluer l'évolution de la tâche en fonction de la longueur d'onde. On constate d'abord que, la position de la tâche est indépendante de la longueur d'onde, elle est purement due à la géométrie du batteur. Et on constate également que la largeur à mi-hauteur de la tâche évolue linéairement avec λ à un préfacteur $\frac{1}{2}$ près. C'est un résultat classique de la diffraction, retrouvé dans l'article [2].

4.2 Récupération de l'énergie : observation à l'oscilloscope et traitement numérique

4.2.1 Montage : récupération de l'énergie des vagues

Après avoir réalisée la focalisation, on est prêt à récupérer l'énergie de notre houle. Pour cela on met en place un système de récupération d'énergie qui utilise le matériel suivant :

- une bobine de fil de cuivre de diamètre $0,8mm$ de longueur $22m$ et de résistance $0,9ohms$.
- des disques magnétiques en néodyme commandés sur supermagnet ref $S-10-05-N52N$ de diamètre $10mm$ et de hauteur $5mm$.
- un petit flotteur acheté chez *Decathlon* de masse $5g$ et de section perpendiculaire aux vagues $S = 10cm2$. Il constitue le corps du système de récupération.
- deux ressorts de $9cm$ de longueur et d'une raideur $k = 2N/m$. La raideur n'est pas très élevée pour permettre au flotteur d'épouser le mouvement des vagues sans dériver et s'éloigner de sa position initiale.
- une poulie et un fil de nylon qui transmettent le mouvement de translation du flotteur aux aimants de néodyme placés à l'intérieur de la bobine.
- oscilloscope et câbles électriques

Les éléments sont assemblés de telle sorte à réaliser le montage présenté en *Figure* 2.

4.2.2 Protocole de récupération de l'énergie

Pour récupérer l'énergie, on place 3 petits aimants de néodyme à l'intérieur de la bobine, et on les relie au corps du flotteur grâce à notre poulie et à un fil de nylon. Le corps du flotteur est placé précisément au centre de la tâche focale, de sorte à avoir une amplitude maximale des vagues. La tension aux bornes de la bobine est visualisée à l'oscilloscope.

Quand notre batteur génère des ondes focalisées, le mouvement vertical des vagues met en mouvement le flotteur. Ce mouvement est transmis aux aimants grâce à la poulie et au fil de nylon.

Par induction, le mouvement de l'aimant va créer un courant qui circule dans la bobine. Une tension quasi sinusoïdale ayant une amplitude de l'ordre d'une centaine de mV est observable

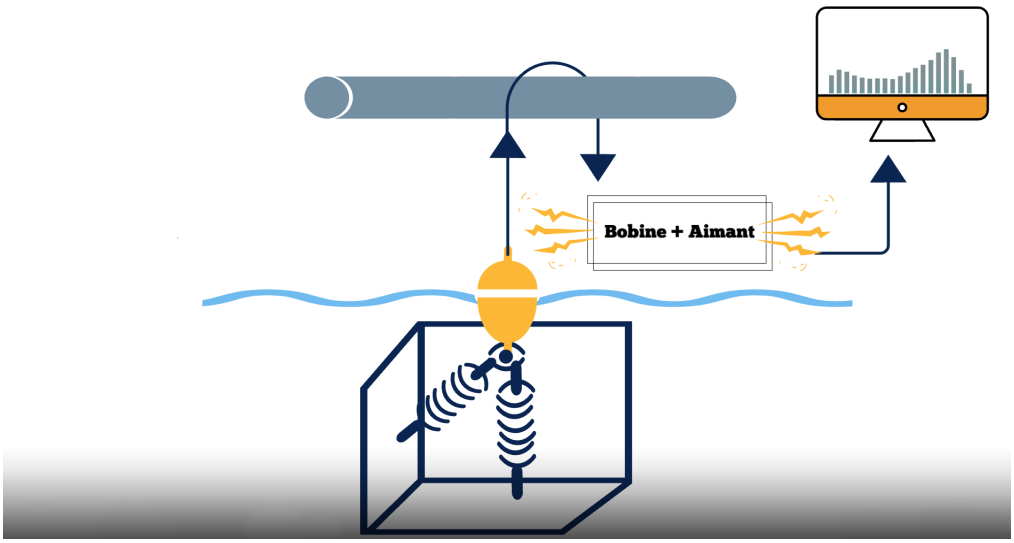

FIGURE 2 – Montage de récupération de l'énergie houlomotrice.

à l'oscilloscope. On sauvegarde ensuite les données de l'oscilloscope sous format *.csv* pour les exporter et les traiter sur **Matlab**.

4.3 Calcul de rendement

Une fois qu'on obtient des mesures de qualité et d'amplitude suffisante sur l'oscilloscope, on peut quantifier cette récupération d'énergie par un calcul de rendement.

Pour cela on calcule d'abord la puissance mesurée, c'est à dire la puissance électrique récupérée grâce au signal sur l'oscilloscope. On a :

$$P_{elec} = \frac{U_{eff}^2}{R}$$

Où U_{eff} est la tension efficace du signal obtenu et R la résistance de la bobine.

D'autre part on a besoin de calculer la puissance disponible notée P_{vague}, qui correspond à la puissance fournie par une vague captée par le flotteur sur sa section. On a la puissance au niveau du flotteur qui est :

$$P_{vague} = \frac{1}{2\sqrt{2}} \rho A^2 g^{3/2} \lambda^{1/2}$$

Avec A l'amplitude des vagues d'un pic à un creux (fournie par la mesure du champ des hauteurs et de l'ordre du cm) la longueur d'onde λ de nos ondes (calculée à partir d'Image J).

On peut alors calculer le rendement en effectuant $r = \frac{P_{elec}}{P_{vague}}$.

4.3.1 Rendement en fonction de la fréquence et de la profondeur

Notre rendement est assez faible. Le système de récupération rudimentaire comporte de très nombreuses pertes et les caractéristiques du vérin limitent très fortement l'amplitude des vagues. On a quand même essayé de réaliser des études de rendement en fonction de la fréquence des vagues choisie et en fonction de la hauteur d'eau.

Cependant, une des caractéristiques de notre système est que, contrairement à la mer, nous ne sommes pas dans un milieu infini. La cuve est petite et les parois réfléchissent les ondes. On a donc décidé, dans le but d'augmenter l'amplitude des vagues afin d'améliorer notre rendement, de se placer en régime d'ondes stationnaires. Dans ce régime, où la fréquence choisie est liée aux dimensions de la cuve, une résonance a lieu et l'amplitude des vagues augmente considérablement. C'est dans ces conditions là que nous avons obtenu nos meilleurs résultats de rendement.

L'extraction des signaux de l'oscilloscope et le calcul du rendement peuvent se faire numériquement avec le code **Matlab** en Annexe E.

Ce code filtre les signaux de l'oscilloscope pour éliminer le bruit et calcule le rendement en fonction des paramètres cités ci-dessus.

Dans le cas de la fréquence, nous voyons que le rendement a une forme de Gaussienne, avec un pic à 10% de rendement autour de $7,5Hz$. Ceci est bien cohérent puisque cette fréquence correspond à un mode propre de la cuve, le rendement diminue de part et d'autre.

Dans le cas de la profondeur, le comportement est beaucoup moins exploitable. Le protocole consiste se placer à la fréquence propre de $7,5Hz$, et à faire varier la hauteur d'eau. Malheureusement les résultats obtenus pour le rendement ne semblent pas suivre d'évolution logique. En réalité, cela était déjà observable à l'oeil nu avant le traitement numérique. En fait, ajouter de l'eau modifie la façon dont le batteur interagit avec l'eau. Par exemple il suffit d'ajouter $1cm$ d'eau pour que le batteur plonge entièrement dans l'eau au lieu de taper sa surface . Les mesures à différentes profondeurs ne sont donc pas comparables puisque les vagues générées ne sont pas de la même nature.

Annexe

A Code Arduino

```
    #include <Controllino.h>
    void setup () {
// put your setup code here, to run once :
pinMode (CONTROLLINO_D3, OUTPUT); // ON
pinMode (CONTROLLINO_D5, OUTPUT); // SET
pinMode (CONTROLLINO_D6, OUTPUT); // A-PHAS
}

    void loop () {
// put your main code here, to run repeatedly :
digitalWrite(CONTROLLINO_D3, HIGH); // Allume le moteur
digitalWrite(CONTROLLINO_D5, HIGH); // Vitesse max
digitalWrite(CONTROLLINO_D6, LOW); // Sort le vérin entièrement
delay (160); // À modifier pour modifier la fréquence de l'onde
digitalWrite(CONTROLLINO_D6, HIGH); // Rentre le vérin
delay (160); // À modifier pour modifier la fréquence de l'onde
}
```

B Code Matlab Analyse-vague

```
%Script principal de traitement par méthode de shclieren

close all
clear all
%On recupère les données .tiff préparés dans le même dossier que ce code
%on les stocke dans f
% l'image de référence f1 correspondant à une surface de l'eau non perturbée
f=rdir('*.tiff*');
f1=rdir('ref.*');
% Charge l'image de référence
Iref=imread(f1.name);
% On croppe l'image sur la partie quadrillée
Iref = rgb2gray(Iref);
Iref = double(Iref);
Iref= Iref-mean(mean(Iref));

% Ici, on va boucler sur les images

for i=1:length(f)-1

% Charge l'image de travail
Idef=imread(f(i).name);

% On tramsforme les image en double pour éviter les erreurs d'arrondi
Idef = rgb2gray(Idef);
```

```
Idef = double(Idef);

Idef= Idef-mean(mean(Idef));
% Trouve 2 carrier peaks indépendants (orth) dans l'image de référence, grâce à la
% fonction findorthcarrierpks
[kr, ku] = findorthcarrierpks(Iref, 4*pi/min(size(Iref)), Inf);

% ectraction de ces signaux de l'image de référence
krad = sqrt(sum((kr-ku).^2))/2;
fIref = fft2(Iref);
cr = getcarrier(fIref, kr, krad);
cu = getcarrier(fIref, ku, krad);

% Obtient le champ de déplacement et le profil des hauteurs
fIdef = fft2(Idef);

[u,v] = fcd_dispfield(fIdef,cr,cu);

%on enléve la pente de l'image, on retire la moyenne du gradient
u=u-mean(mean(u));
v=v-mean(mean(v));

h = fftinvgrad(-u,-v);
%h = invgrad2(-u,-v);


%sauver le résultat (avant le déjeuner!!)
a=f(i).name;
nom_fichier=strcat('heigth',a(end-9:end-5),'.mat');
save(nom_fichier,'h');
 fft2Dh=fftshift(fft2(h));
 fft2Dh=fft2Dh(513-300:513+300,433-300:433+300);
 nom_fichier=strcat('fft2d',nom_fichier);
 save(nom_fichier,'fft2Dh')

disp(i)
end

%
f=rdir('heigth*.mat');
h;
order=-3;
filtsize=10;

for i=1:length(f)
    load(f(i).name);
    H=filtrage(order,filtsize,h);
    nom_fichier=strcat('Hfiltre',a(end-9:end-5),'.mat');
    save(nom_fichier,'H');
    disp(i)
    imagesc(H);
    caxis([-2 2])
    colorbar;
    i
    pause(0.01);
end
```

```
83
84  %f=rdir('*.tiff*');
85  WriterObj = VideoWriter('video_Sclieren.avi');
86  WriterObj.FrameRate = 7;
87  open(WriterObj);
88  for i=1:length(f)
89       Frame = imread(f(i).name);
90       writeVideo(WriterObj, Frame);
91       i
92  end
93  close(WriterObj);
94
95
96  %f=rdir('heigth*.mat');
97  h;
98  order=-3;
99  filtsize=10;
100 %On applique la fonction filtrage à toutes les images
101
102 for i=1:length(f)
103      load(f(i).name);
104      test(:,:,i)=h;
105      H=filtrage(order,filtsize,h);
106      testfiltre(:,:,i)=H;
107      i
108 end
109 %xfocal=740;
110 yfocal=504;
111 % évolution temporelle du signal en un point
112 signal=squeeze(testfiltre(xfocal,yfocal,:));
113
114 % transformée de fourier
115 Y=fft(signal);
116 M=max(Y);
117 Y=Y./M; %normalisation
118 freq=linspace(0,25,length(f));
119
120
121 figure(3)
122 plot(freq,(abs(Y)).^2);
123 title('Transform e de fourier temporelle du champ des hauteurs au point focal')
124 xlabel('fr quence en Hz')
125 ylabel('module au carr  de la FFT normalis e')
126
127
128 %Rfocal=500;
129
130 F=fft(testfiltre,[],3);
131 F_bis=abs(F(:,:,81)).^2;
132 W=F_bis(100:end,:);
133 W(W>9000) = [9000];
134 tic
135 for i=1:size(W,1)
136      for j=1:size(W,2)
137           if ( sqrt((i-xfocal)^2+(j-yfocal)^2)> Rfocal)
138           W(i,j)=0;
```

```
            end
        end
    end
    toc
    W=W+6000;
    %figure(4)
    s=surf(W,'FaceAlpha',1);
    view(145,45)
    s.EdgeColor = 'none';
    xlabel(' x en px')
    ylabel('y en px')
    title('Amplitude au carr  de la FFT en 3D filtr  e')

    %Tache d'airy
    figure (5)
    V=W(672:673,100:900);
    L=V(1,:);
    L=L/8936.8;
    abscisse=ones(1,length(L));
    ordonnee=ones(1,length(L));
    for i=1: length(L)
        abscisse(i)=i*58/1024;
        ordonnee(i)=L(i);
    end

    abscisse=abscisse*58/1024;
    plot(abscisse,ordonnee)

    xlabel('largeur de la t che focale en cm')
    ylabel('hauteur normalis e')
    title('T che Airy ')

    %
    %Crée une structure V qui stocke toutes les images

    for n=1:272
        o=figure(9);
        axis tight manual

        Vid = testfiltre(:,:,n);
        clims=[0,10000];
        Vid=Vid(100:end,:);
        Vid=smoothn(Vid);
        Vid(Vid>9000) = [9000];

     % filtrage
        for i=1:size(Vid,1)
            for j=1:size(Vid,2)
                if ( sqrt((i-xfocal)^2+(j-yfocal)^2)> Rfocal)
                Vid(i,j)=0;
                end
            end
        end

        %acquisition
        s= surf(Vid,'FaceAlpha',1);
```

```
195         s.EdgeColor = 'none';
196         shading interp;
197         colorbar;
198         caxis manual
199         caxis([-2.5 2.5]);
200 
201         view(145,60);
202         title(num2str(n));
203         xlabel(' x en px')
204         ylabel('y en px')
205         xlim([0,1000]);
206         ylim([200,700]);
207         zlim([-4,4]);
208 
209         n
210         Video(n) = getframe(gcf) ;
211         drawnow
212 
213 end
214 
215 %
216 writerObj = VideoWriter('video_focalisation.avi');
217 writerObj.FrameRate = 5;
218 open(writerObj);
219 for i=1:length(Video)
220         frame = Video(i) ;
221         writeVideo(writerObj, frame);
222 end
223 close(writerObj);
224 
225 
226 
227 %
228 %la section "TRAITEMENT DE L'IMAGE FILTRÉE", nous la fréquence à laquelle la fft a une amplitude
    maximale
229 %Elle correspond à la fréquence d'excitation : environ 3-4 Hz
230 %le 81 représente les 3-4 Hz dans l'échelle de la fft
231 f_excitation=81;
232 
233 %Conversion Pixels en cm
234 size_x_ref=58*(825/1024);
235 size_y_ref=63*(981/1280);
236 
237 F=fft(testfiltre,[],3); %TF temporelle
238 image=(abs(F(:,:,f_excitation)).^2);
239 clims=[0,10000];
240 imagesc(image,clims);
241 colorbar
242 colormap parula
243 
244 %modification des axes
245 
246 xticklabels= 0:10:size_x_ref;
247 xticks = linspace(0, size(image, 2), numel(xticklabels));
248 set(gca, 'XTick', xticks, 'XTickLabel', xticklabels);
```

```
yticklabels= 0:10:size_y_ref;
yticks = linspace(0, size(image, 1), numel(yticklabels));
set(gca, 'YTick', yticks, 'YTickLabel', yticklabels);
xlabel(' x en cm')
ylabel('y en cm')
title('Amplitude du pic    Fexct de la FFT en tout point de la cuve')
```

C Code Matlab pour la fonction filtrage

```
function [ H ] = filtrage( order, filtsize, data )


% code pour filtrer spatialement, et çeliminer les effets de bord au
% limites de la cuve.
% order=-3
% filtsize=100

%On définit un rectangle correspondant à l'image qu'on veut exploiter, ces
%paramêtres changent pour chaque acquisition
x0=200;
xf=990;
y0=150;
yf=1150;

nx=size(data,1);
ny=size(data,2);

n=min(nx,ny);
kx0=nx/2+1; % mode 0
ky0=ny/2+1;


[ky,kx] = meshgrid(1:ny,1:nx);
k = sqrt((kx-kx0).^2+(ky-ky0).^2);


T = 1./(1+(k/(n/filtsize)).^(order/2));
sp=fftshift(fft2(data));
H = real(ifft2(ifftshift(sp.*T)));

%on crop la matrice H
H=H(x0:xf,y0:yf);

end
```

D Code Matlab pour la largeur de la tâche de focalisation en fonction de la fréquence

```
%
%delay 80,90,100,110,120,140
%freq=(4.265,4.853,5.357,5.882,6.548,7.5) ; largeur=(78,73,65,53,52,45) ; l=zeros(1,length(freq)) ;
error=zeros(1,length(freq));
```

```
5
6  %conversion pixels/mètres
7  a=0.58/1024;
8  largeur_m=largeur.*a; %largeur de la tâche en m
9
10
11 for i=1:length(freq)
12    l(i)=2*9.81/((2*pi)*freq(i))^2;
13    error(i)=1*(0.58/1024*sqrt(2));
14 end
15 % On s'attend à avoir une évolution linéaire avec lambda
16 %les taches on une allure de tache d'Airy, la largeur correspond à leur
17 %larguer à mi-hauteur.
18
19
20
21 figure(1)
22 plot(freq,largeur_m,'o')
23 hold on;
24 p = polyfit(freq,largeur_m,1);
25 largeur_fit = polyval(p,freq);
26 plot(freq,largeur_fit);
27 errorbar(freq,largeur_m,error,'bx');
28 xlim([3, 8]);
29 xlabel("fr quence en Hz");
30 ylabel("Largeur de la t che focale en m");
31 title("Largeur de la t che en fonction de la fr quence");
32
33 figure(2)
34 plot(l,largeur_m,'o')
35 hold on;
36 q = polyfit(l,largeur_m,1);
37 largeur_fit = polyval(q,l);
38 plot(l,largeur_fit);
39 errorbar(l,largeur_m,error,'bx');
40 xlim([0, 0.035]);
41 xlabel("longueur d'onde en m ");
42 ylabel("Largeur de la t che focale en m");
43 title("largeur de la t che en fontion de la longueur d'onde");
```

E Code Matlab pour le calcul du rendement

```
1  %FILTRAGE DU SIGNAL OSCILLO
2
3  %
4  %
5  % Fabrication d'un signal d'exemple
6  %phi=0.63;
7  %test=2.43*cos(2*pi*f*X+phi)+2*(rand(length(X),1)-0.5);
8  %plot(X,test)
9
10
11 %f=7.602; R=0.9;
12 %On charge le fichier de données de l'oscillo (format .csv)
13 %On rétire les premierès valeurs (Nan)
```

```
%Les données sont en V pour la tension et en s pour le temps
X=X(3:end);
CH1=CH1(3:end);
test=CH1;

% On extrait la composante de la fft correspondant à f
analyseur=exp(-sqrt(-1)*2*pi*f*X);
amp=test.*analyseur;

%amplitude=sum(abs(amp));
amplitude = amplitude/length(X)*2;
phase = angle(amplitude);

%subplot(2,1,1)
plot(X,real(amp))
hold on
%plot(X,imag(amp))
title('tension aux bornes de la bobine filtr e')
xlabel('temps en secondes')
ylabel('tension en volts')

subplot(2,1,2)
plot(X,CH1)
xlabel('temps en secondes')
ylabel('tension en volts')
title('tension aux bornes de la bobine')

%
F=[5.882, 8.529, 6.6180,7.1640, 8.97,7.602]; %en Hz
Profondeur=[15,16,16.5,17,17.5,18] %en Cm

A=[0.0502,0.1107,0.0799,0.1369,0.0393,0.1369];
Abis = [0.1369,0.2028,0.1503,0.0898,0.2280,0.1122]

R=0.9;
h=0.005; %hauteur d'une vague en m

r=zeros(1,length(A));
rbis=zeros(1,length(Abis));
error=zeros(1,length(A));
errorbis=zeros(1,length(Abis));
s=0.001

for i=1:length(A)
    P_elec =(((A(i))/sqrt(2))^2)/R;
    l=(2*9.81/(F(i)^2)); %longueur d'onde d'une vague en m.
    Pth=(1/2*sqrt(2))*1000*9.81^(3/2)*h^2*sqrt(l);
    r(i)=(P_elec/Pth)*100;
end

for i=1:length(Abis)
    P_elec =(((Abis(i))/sqrt(2))^2)/R;
    Pth=(1/2*sqrt(2))*1000*9.81^(3/2)*h^2*sqrt(l);
    rbis(i)=(P_elec/Pth)*100;
end

```

```
%incertitude pour la fréquence
moy=sum(r)/6;
ecart_type=sqrt(0.2*sum((r-moy).^2));
incertitude_type=ecart_type/sqrt(6);


%incertitude pour la profondeur
moybis=sum(rbis)/6;
ecart_typebis=sqrt(0.2*sum((rbis-moybis).^2));
incertitude_typebis=ecart_typebis/sqrt(6);

error = error + incertitude_type;
errorbis = errorbis + incertitude_typebis;



figure(1)
plot(F,r,'o');
hold on;
errorbar(F,r,error,'bx');
xlim([5.6, 9]);
title('Rendememement en fonction de la fr quence')
xlabel('fr quence en Hz')
ylabel('rendement en %')

figure (2)
plot(Profondeur,rbis,'o');
hold on;
errorbar(Profondeur,rbis,errorbis,'bx');
xlim([14.5, 18.5]);
title('Rendement en fonction de la profondeur')
xlabel('profondeur en cm')
ylabel('rendement en %')
```

Références

[1] Sander Wildeman. Real-time qauntitative schlieren imaging by fast fourier demodulation of checkered backdrop. Experiments in Fluids, 2018.

[2] Lucie Domino. Contrôle et manipulation d'ondes hydroélastiques. Thèse en Ligne, 2018.