SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

3A104
Projet Numérique

Rapport
Détection et suivi de particules

MONTREUIL Yannis
CHAPEL Antoine

Encadrant :
KRAWCZYNSKI Jean-François

16/04/2019

Table des matières

Introduction

À travers ce rapport, nous vous présentons ce que nous avons réalisé dans le cadre de la deuxième partie de l'unité d'enseignement 3A104. Le but de ce projet est d'étudier et de représenter numériquement un écoulement de particules autour d'un cylindre. Pour ce faire nous allons essayer de reconstituer les champs de vitesse de l'écoulement autour du cylindre. La reconstitution de ces champs sera effectuée à l'aide de la détection et du suivi de particules présentes dans cet écoulement. Dans un premier temps nous allons donc présenter ce projet de manière à bien en dégager les enjeux, puis nous expliciterons nos méthodes de résolutions pour finalement nous concentrer sur les résultats que nous aurons obtenu.

Présentation du projet

Ce projet se base sur une technique expérimentale utilisée encore aujourd'hui qui s'intitule : *Particle Tracking Velocimetry (PTV)*, ou encore *Vélocimétrie par suivie de particules (VSP)* en français. Cette technique permet de déterminer la vitesse des particules agissant à l'intérieur d'un fluide. Il semble que cette technique ai commencée à être utilisée dans les année 90 à l'aide de l'utilisation de films photographiques. L'avancée technologique à tout particulièrement favorisée le développement de cette technique puisque l'évolution des outils optiques à considérablement diminué le temps de prise de mesure entre deux images successives, ainsi que le développement de cette méthode par le biais numérique. Ce procédé de mesure repose donc sur le fait que chaque particule est suivie individuellement entre deux images successives. À l'inverse d'autres méthodes qui consistent à suivre un ensemble de particules dans leur mouvement, la PTV permet d'obtenir une meilleure résolution spatiale. Dans la plus grande majorité des cas, un laser illumine les particules. Ensuite une série de prise d'image est effectuée. L'écart de temps entre deux images est relativement petit de manière à obtenir finalement une représentation précise de l'écoulement considéré.

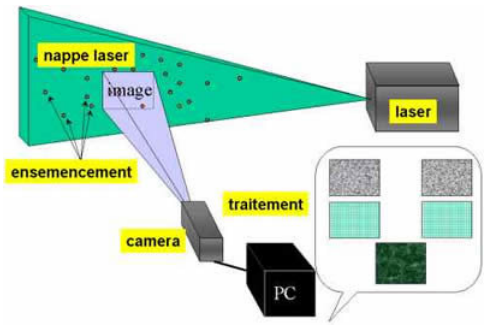

FIGURE 1 – Méthode PTV

Aujourd'hui les méthodes de mesures utilisant l'analyse d'images sont devenues des outils très utilisés pour la recherche expérimentale en mécanique des fluides. Cette méthode est composée de deux procédés bien différents puisque le premier est une approche bidimensionnel (2D) alors que le deuxième est tri-dimensionnel (3D)

- L'approche 2D étudie une coupe bidimensionnelle du fait qu'un seul plan de l'écoulement soit éclairé par le laser. Pour cela une lentille cylindrique peut être utilisée de manière à étaler le faisceau dans une direction, de plus des lentilles sphériques peuvent être utilisées pour focaliser l'énergie du laser sur la zone souhaitée.
- L'approche 3D repose quant à elle sur l'utilisation de plusieurs caméras et d'un éclairage en volume permettant une visualisation tridimensionnelle de l'écoulement des particules.

Dans notre cas, nous nous intéresserons tout particulièrement à l'utilisation de la méthode bidimensionnelle. L'écoulement considéré est un écoulement de particules autour d'un cylindre dont vont trouverez une représentation schématique ci-dessous.

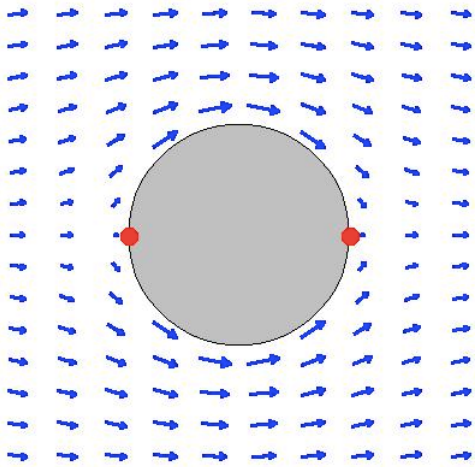

FIGURE 2 – Écoulement plan autour d'un cylindre

Nous disposons de cent images permettant d'analyser les particules en mouvement dans l'écoulement autour du cylindre. La partie pratique de cette méthode à donc déjà été effectuée et nous a été transmise, notre objectif est donc le traitement numériques des données collectées lors de cette première partie.

Première partie

Analyse des données

1 Objectifs du traitement numérique

Dans un premier temps, notre objectif est d'identifier et de déterminer la position des particules dans cet écoulement. Afin d'obtenir la meilleure représentation finale de cet écoulement il nous faut nous intéresser au plus grand nombre possible de particules puisqu'une petite partie seulement ne serait pas représentatif de l'écoulement. Le fait d'étudier le plus grand nombre possible de particules nous permet donc d'obtenir une bonne représentation spatiale de l'écoulement. La seconde partie de ce projet sera donc d'associer la corrélation temporelle des particules avec leur position, on déterminera alors les champs de vecteurs de l'écoulement. Les images étant séparées d'un pas de temps connu, il nous faut donc après avoir déterminé la position $x(t)$ d'une particule à l'instant t, faire correspondre cette position $x(t)$ avec la position suivante au temps suivant à l'instant $t + 1$. La variation de la position dans le temps d'une particule nous permet d'obtenir son vecteur vitesse.

2 Identification des centres des particules

2.1 Pixel tiré de l'image I

La toute première étape de cette partie consiste à créer une partie de code permettant de sélectionner et de déterminer la taille d'une des cents images de la série de mesure qui nous a été fournie. La taille de cette image est

de 640*860 px. À la suite de cela nous l'affichons à l'écran. Cette image nous permet de connaitre les différentes intensités des particules de l'image.

Il est aisé d'observer à l'œil nu le fait qu'il y ait un nombre très important de particules, il nous est cependant presque impossible de les distinguer individuellement ni de connaitre leur taille caractéristique. De manière à en savoir plus sur ces particules nous zoomons donc sur une petite partie de notre image, de manière à distinguer précisément une particule et d'en déterminer par la même occasion sa taille caractéristique. Nous zoomons donc sur l'image de manière à observer une seule particule dans une matrice 32×32 px. La taille caractéristique d'une particule est d'environ 4×4 px.

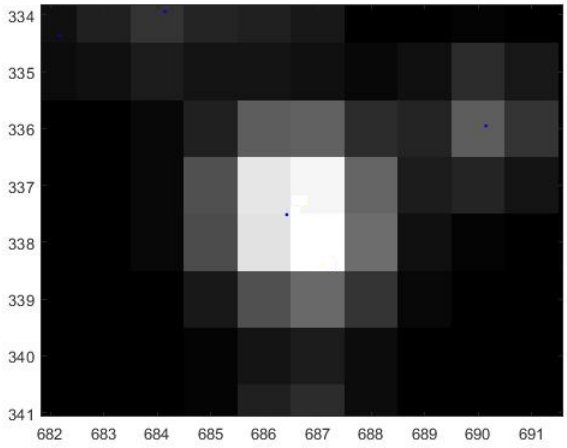

FIGURE 3 – Particule extraite de l'image avec son centre

Toutes les positions exactes des particules nous sont fournies dans un autre dossier et nous les affichons par des points bleues sur notre l'image de référence de manière à avoir une idée plus clair du nombre de particules. Nous avons à notre disposition les milieux de chaque particules afin de pouvoir comprendre plus facilement l'objectif de cette première étape.

2.2 Création d'un pixel suivant la loi Gaussienne

On considère à présent que les particules en mouvement dans cet écoulement ont des propriétés de diffusion générant une distribution gaussienne (2D) en intensité de niveaux de gris. La fonction gaussienne nous vient du mathématicien *Carl Friedrich Gauss*, cette loi peut aussi être retrouvée sous le nom de loi normal du fait qu'une majorité de phénomènes physiques suivent cette loi. Cette loi est une loi de probabilité continue. D'une manière générale on peut identifier les phénomènes physiques qui suivent cette loi, pour cela les phénomènes doivent suivre la condition de *Borel*. Une variable aléatoire suit la condition de Borel si cette variable dépend de plusieurs paramètres indépendant sans qu'aucun ne prédomine. C'est une loi très utilisée dans le calcul de probabilités. La fonction gaussienne est une fonction exponentielle négative représentative d'une courbe en cloche symétrique et de la forme analytique suivante :

$$f(x) = a \exp(-\frac{(x-b)^2}{2c^2}) \tag{1}$$

Dans de nombreux cas la fonction Gaussienne est utilisée pour calculer la densité de probabilité d'une variable aléatoire, normalement distribuée, avec une valeur attendue et une variance σ ($\sigma^2 = c^2$et $\mu = b$). Dans ce cas la fonction Gaussienne devient alors :

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma}) \tag{2}$$

Dans notre cas nous générons une particule à travers une fonction gaussienne (2D) en intensité de niveau de gris, la particule est encodée en 8 bits car il y a 256 niveaux de gris. La dimension de la matrice dans laquelle se trouve notre particule est de 33×33 px. L'objectif est de positionner aléatoirement cette particule à 5 px du centre de la matrice et de retrouver son centre. Pour ce faire nous créons un maillage avec un pas de petite taille ($pas = \frac{1}{32}$ et $\sigma = 1$) de manière à avoir une distribution gaussienne quasi continue (un pas de grande taille, et σ trop grand implique une distribution gaussienne très pixelisée). Nous incrémentons avec deux paramètres aléatoires rdx et rdy permettant de ne pas fixer l'image au centre de notre maillage. Nous intitulons cette image I_m . Ce maillage est défini par la ligne de code suivante :

Algorithme 1 Création du maillage

```
% Sx et Sy la taille de notre image 33 × 33
rdx=-5+rand*(5+5+1) %paramètre aléatoire(image non fixe)
rdy=-5+rand*(5+5+1)
[x,y]=meshgrid((1 :pas :Sx)-Sx/2+rdx,(1 :pas :Sy)-Sy/2+rdy) ; % création du maillage associé
```

Dans notre cas , la fonction Gaussienne 2D est définie de la manière suivante :

$$f(x,y) = I_{max} \exp(-\frac{1}{2}\frac{(x^2+y^2)}{\sigma}) \tag{3}$$

$\sigma = 1$ est notre paramètre de pixellisation et $I_{max} = 255$ est l'intensité la plus grande d'un pixel. Nous obtenons alors la particule suivante en y indiquant son centre :

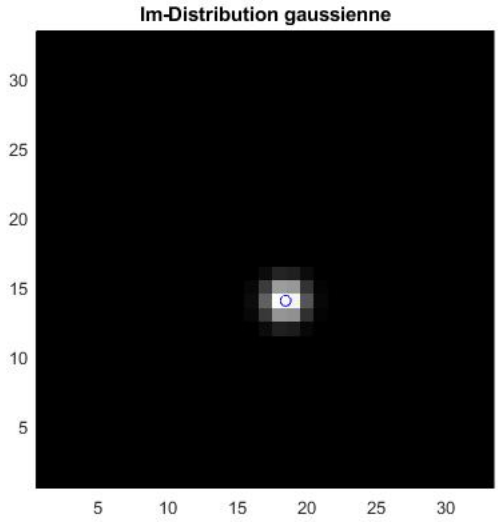

FIGURE 4 – Particule crée via la fonction Gaussienne (I_m)

2.3 Recherche des centres du pixel

Notre objectif à présent est de retrouver le centre de notre particule à l'aide d'une méthode. Pour cela nous utiliserons la fonction *lsqnonlin.m* qui permet de réaliser une minimisation de notre problème et ainsi de trouver notre milieu.

Avant cela non codons une fonction *ecart.m* que l'on va mettre en argument de notre fonction *lsqnonlin.m.*

Algorithme 2 Fonction *ecart.m*

```
function [evalecart] = ecart(x1,M,sig)
n=4*sig;
[X,Y] = meshgrid(-n :n,-n :n); % Génération du maillage associé à la fonction evalecart
evalecart=x1(3)*exp(-((X-x1(1)).^2+(Y-x1(2))^2)/(2*sig)) -M;
```

Une fois mise en argument de la fonction *lsqnonlin.m,* nous obtenons le résultat de notre minimisation indiquant les coordonnées du centre de la particule, ainsi que l'intensité du centre. Nous utilisons donc cette fonction sur le pixel crée via la loi Gaussienne, et nous obtenons son centre :

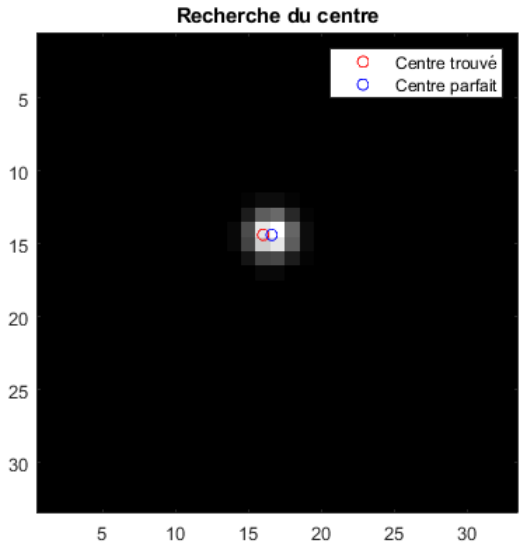

FIGURE 5 – Centre du pixel crée en rouge

Ces étapes nous ont permis d'obtenir un code viable et grâce auquel nous pouvons à présent retrouver la position d'une particule dans le cadre d'une distribution Gaussienne (2D) en ayant fixé la taille caractéristique d'une particule. L'intérêt d'avoir développé une telle méthode de résolution pour la détermination de la localisation d'une particule est de pouvoir l'adapter pour la détermination des localisations d'un grand nombre de particules (image entière).

Nous allons donc rechercher les centres des particules qui composent notre première image I. Nous commençons par réaliser le programme pour une seule image (nous généraliserons une fois que ce dernier est correct). Avant de commencer, nous devons définir des conditions de manière à filtrer les particules qui sont de " *bonnes candidates* " ou non.

Pour cette étape nous commençons par dilater les particules de notre image initiale pour nous permettre de détecter le plus grand nombre possible de particules. Pour ce faire nous utilisons la fonction *imdilate.m.* Cette fonction est un outil nous permettant de modifier la morphologie de nos particules. L'utilisation de cette fonction est très simple puisque qu'il nous suffit de définir la manière dont l'on dilate ainsi que l'image choisie. L'utilisation de cette fonction nous permet de définir une première condition. On crée une variable *dilate* dans laquelle se trouve les positions des particules dilatées de notre image. La première que nous imposons consiste donc à comparer les positions des particules dilatées ou non. Si la position d'une particule dilatée est la même que celle d'une particule non dilatée alors nous la considérons comme une "*bonne candidate*". Pour cette condition nous avons utilisé la fonction *find.* Cette première condition est définie par la ligne de code suivante :

Algorithme 3 Dilatation des particules

```
se=strel('sphere',3) ; %dilatation des particules
dilate=imdilate(I, se) ; % I désigne l'image
k=find(I==dilate) %comparaison entre particule dilatée ou non, k est un l'indice des points répondant à la comparaison
```

Nous définissons ensuite une seconde condition de filtrage. Cette condition repose sur le fait que nous choisissons une valeur minimale d'intensité (ici 42). Cette condition a pour objectif de ne pas considérer les pixels noires de notre image comme des particules appartenant à nos deux images. Une fois ces deux conditions remplis nous plaçons les indices des "*candidats*" dans deux vecteurs ($[Row, Col]$) de manière à pouvoir réutiliser ces coordonnées dans la suite du programme. Une fois les positions de nos "*candidats*" déterminés nous devons à présent en déterminer le centre de manière à pouvoir les suivre au fil du temps, soit entre deux images successives ($t-1$ et t)

Pour définir avec exactitude le centre de nos particules "*candidates*" nous utilisons la méthode précédente et de manière à être sur de nos résultats nous allons superposer sur la même image les résultats que nous trouverons avec les positions des particules qui nous ont été fournies dans le fichier *parameters.m.*

Nous obtenons un problème sur les bords de notre images lors de l'extraction de nos sous matrices de l'image I. Nous faisons donc appel à la fonction *wextend.m,* qui nous permet de prolonger l'image avec des pixels noirs et donc de paramétrer l'image à la taille souhaitée de manière à pouvoir analyser les particules situées aux bords de notre matrice. Nous décidons d'augmenter la taille de notre matrice de 4 pixels ce qui est suffisant pour que toutes les particules soient bien définies lors de l'extraction des sous matrices. Nous adaptons alors le code précédent grâce auquel nous avons déterminé le centre d'une particule avec l'utilisation de la fonction *lsqnonlin.m.* Nous plaçons les positions des centres de nos particules calculés à l'aide de la fonction *lsqnonlin.m* dans un vecteur nommé $x2sol$. Enfin nous stockons la positions de ces centres dans deux matrices $ord(y_{centre}, N_{image})$ et $abs(x_{centre}, N_{image})$ de la manière suivante :

Algorithme 4 Stockage des positions des centres sur l'image I

```
[x2sol]=lsqnonlin(@(x1)ecart(x1,M,sigma2),x0) ; % Positions des centres
abs(i,j)=x2sol(1)+Col(i) ; %Stockage des positions des centres
ord(i,j)=x2sol(2)+Row(i) ;
avec Col(i) et Row(i) les positions de nos particules "candidates" préalablement déterminés.
```

Nous obtenons donc les positions des centres plus ou moins précisément. Les positions trouvées sont relativement en accord avec ceux fournit. Nous considérons donc nos résultats comme valables pour la détermination des vecteurs vitesse qui sera notre dernière étape de ce projet.

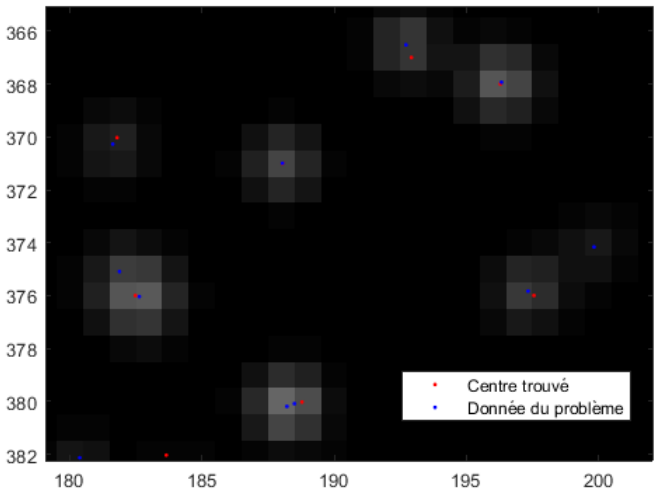

FIGURE 6 – Comparaison entre les données du problème et nos centres

3 Calcul des champs de vecteurs

3.1 Compréhension du problème

Nous possédons maintenant tous les centres des particules pour plusieurs images. Nous pouvons donc en déduire les vecteurs vitesses associés à chaque particule. Mais d'abord nous devons mettre en place un algorithme permettant d'associer une particule entre deux images successives, afin d'y obtenir le vecteur déplacement comme montrer sur l'image ci-dessous :

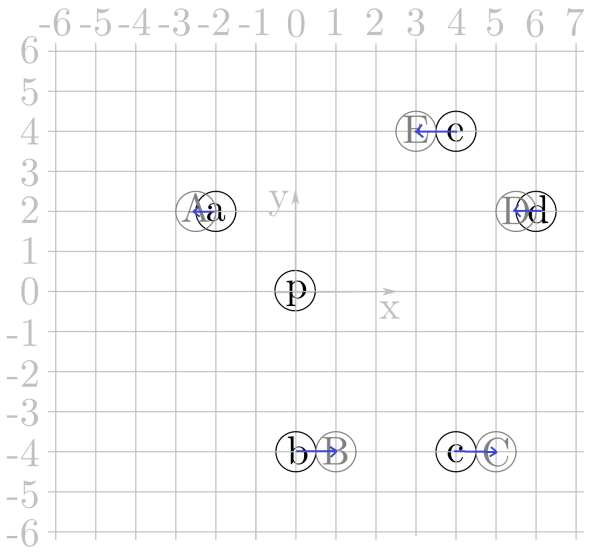

FIGURE 7 – Champs de Vecteurs

Nous avons à les particules à l'instant $t-1$ symbolisées par les lettres en minuscule. Les particules en majuscule sont celle de l'image 2 c'est à dire à l'instant t. Nous pouvons alors tracer les vecteurs vitesses.

3.2 Élaboration de l'algorithme et critiques des résultats

Tout d'abord, nous commençons par réfléchir localement. C'est à dire, que nous allons développer notre algorithme sur une particule de l'image $t-1$. Nous possédons bien-sur le centre de cette dernière, nous allons alors rechercher dans l'image à l'instant t la particule correspondante. Pour cela, nous créons arbitrairement une disque de recherche autour de notre centre de particule de $d_{max} = 30$ pixels. Nous calculons alors les distances entre la particule de l'image $t-1$ et ses candidates de l'image t. Nous obtenons donc plusieurs distance via la fonction *fonction_sort.m*, nous garderons les distances les plus petites ainsi que les indices associées aux particules proche de l'image t. Nous obtenons donc :

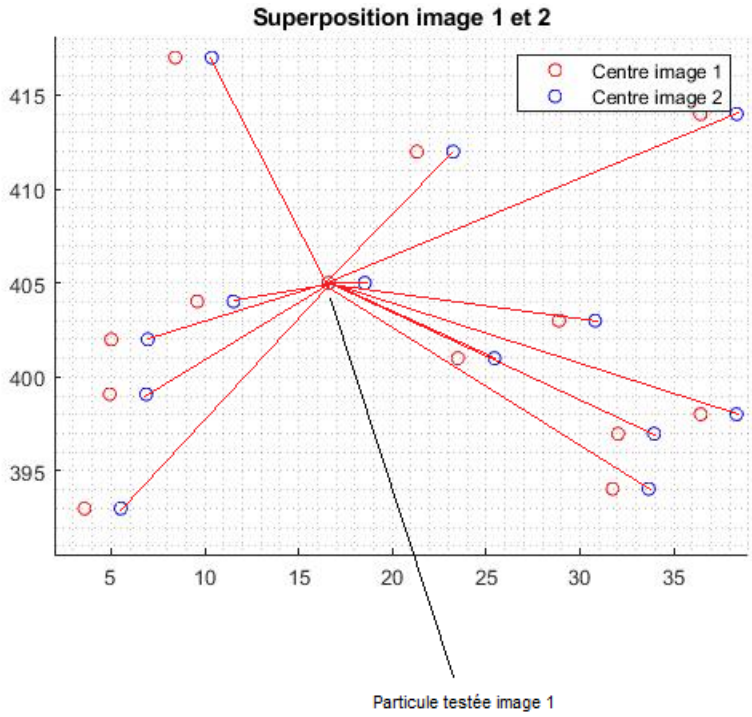

FIGURE 8 – Disque de recherche

Nous utilisons alors la fonction intégrée *corr2.m,* permettant d'effectuer la corrélation entre notre particule test et ses candidates. Nous retenons alors les coordonnées de la particule candidate ayant la plus haute corrélation avec notre particule test. Nous stockons toute ces informations dans notre vecteur appelé $POS(indice_{part-corré})$. Il nous suffit donc maintenant de construire nos vecteurs vitesses (dX et dY sont les coordonnées du vecteurs vitesse) :

Algorithme 5 Vecteurs vitesse

```
for k=1 :size(abs,1)
 dX(k,j)=abs(POS(k),j+1)-abs(k,j) ; % calcul des vecteurs positions
 dY(k,j)=ord(POS(k),j+1)-ord(k,j) ;
end
```

j correspond à l'image t, et k aux particules de l'image t-1

Il nous faut maintenant tracer ces vecteurs vitesses sur notre image. Pour cela, nous créons un maillage adapté à notre image avec la fonction *meshgrid*. Il nous faut maintenant interpoler les points sur notre maillage.

Algorithme 6 Interpolation

```
[xq,yq]=meshgrid(0 :0.5 :size(I,2),0 :0.5 :size(I,1)) ;
dxq=griddata(abs( :,1),ord( :,1),dX,xq,yq) ;
dyq=griddata(abs( :,1),ord( :,1),dY,xq,yq) ;
```

Une fois l'extrapolation réaliser, nous obtenons donc :

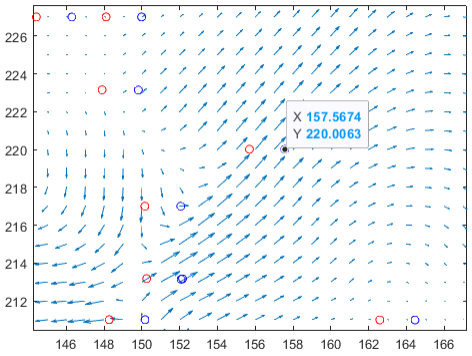

FIGURE 9 – Vecteurs vitesse sur l'image

Le résultat nous parait très approximatif. En effet, les vecteurs vitesses ne sont pas en accord avec ce que nous pensions trouver. Cela peut s'expliquer par plusieurs chose :

— Une erreur sur l'extrapolation ? En effet, nous avons en amont vérifié l'exactitude des distances entre les particules. Malgré cela, les vecteurs retranscrit sur notre image ne semble pas correct. Nous avons donc exploré plusieurs méthodes (*'nearest', 'linear', 'cubic'*), mais le résultat semble toujours incorrect.

— Une condition manquante ? Nous avons l'impression que la fonction *gridddata* ne prend pas totalement en compte les points de notre maillage.

— Nous ne prenons absolument pas en compte qu'une particule se déforme légèrement. Mais notre problème avec nos vecteurs se trouvent en amont, zone ou les particules sont trop peu déformée pour autant agir sur nos résultats.

Conclusion

Ce projet nous a donc permis de découvrir notre première méthode numérique sur un écoulement. Nous avons dû beaucoup nous documenter afin de bien comprendre les concepts et l'application de cette méthode. Les connaissances que nous avons pu développer autour de cette technique enrichissent donc notre bagage en mécanique des fluides, et notamment en méthode numérique. Malgré les nombreux problèmes dûent au codage de ces méthodes ainsi que de les problèmes de compréhension, ce projet nous a beaucoup intéressé étant donnée que ce dernier pouvait nous permettre d'obtenir « *un produit fini* ». C'est aussi le premier projet portant réellement sur un problème physique que nous aurions pu plus ou moins résoudre analytiquement. Nous avons pu nous rendre compte des différentes difficultés pouvant être présente lors d'analyse d'un écoulement.

Références

[1] W. Brevis, Y. Niño et G.H. Jirka. "Integrating cross-correlation and relaxation algorithms for particle tracking velocimetry"

[2] https ://www.lavision.de/fr/techniques/piv-ptv/

[3] « Vélocimétrie laser pour la mécanique des fluides « - Alain BOUTIER

[4] https ://link.springer.com/article/10.1007/s00348-018-2660-7