4. IMPLÉMENTATION DE LA HORDE VIRTUELLE SOUS MATLAB

4.1 Génération aléatoire de la horde de départ

Pour débuter la simulation, nous avons besoin de générer une horde de départ constituée de N manchots (dans la réalité, la valeur de N peut aller jusqu'à plusieurs milliers). Pour que la simulation reflète au mieux la réalité, nous avons décidé de construire cette horde de manière aléatoire. Pour cela, nous avons initialisé la horde avec seulement 3 manchots, puis nous avons ajouté les autres oiseaux un par un de manière aléatoire en respectant les 3 contraintes suivantes :

- C_1 : Chaque manchot ajouté doit être adjacent à la horde
- C_2 : Un manchot doit toujours posséder au moins 2 voisins
- C_3 : L'ajout d' un manchot ne doit pas engendrer de trou dans le maillage

Dans la suite, les manchots seront repérés par les centres des disques dont il était question dans la partie précédente. On utilise le système de coordonnées issu du repérage non-orthogonal présenté en 1.2.2.

Nous avons construit la fonction Matlab ***horde***, qui permet de générer de manière aléatoire une horde de N manchots par l'appel ***horde(N)***.

4.1.1 Données nécessaires à la résolution du problème physique

Afin de savoir de quelle manière implémenter la fonction , il est indispensable d'identifier les données relatives à la horde qui seront nécessaires à la résolution du problème.

Tout d'abord, nous avons besoin de connaître la position de chaque manchot de la horde.

Ensuite, tous les manchots doivent avoir un identifiant qui nous permettra de les désigner, de les localiser ou de connaître l'évolution de leur position au cours du temps.

Puis, afin de pouvoir définir le polygone qui représentera la horde pour le problème physique, nous devons distinguer les individus qui se trouvent sur la bordure de la horde, des individus qui en sont à l'intérieur.

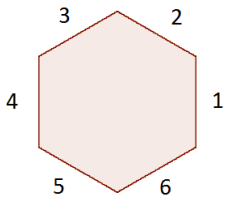
Enfin, pour les manchots situés sur la frontière, nous avons besoin de connaître les positions libres autour de chacun d'entre-eux, afin de définir les positions qui seraient susceptibles d'accueillir un nouveau manchot.

4.1.2 Méthode générale

a. Présentation de la classe 'ping'

Nous avons créé une nouvelle classe nommée ***ping*** qui regroupe les propriétés suivantes pour chaque manchot :

- ***ind*** est un entier qui permet d'identifier le manchot
- ***x*** correspond à la première coordonnée du manchot dans le repère non-orthogonal
- ***y*** correspond à la seconde coordonnée du manchot dans ce même repère
- ***libre*** est un vecteur ligne contenant les indices des positions libres autour du manchot (voir figure)



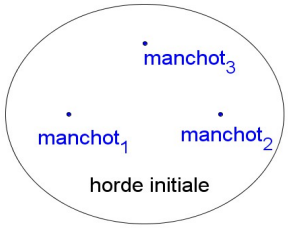
Exemple : Un manchot ***M*** ne possédant que deux voisins aux positions E et NE aura comme vecteur correspondant ***M.libre*** [3, 4, 5, 6]

Créer un nouveau manchot revient à créer un nouvel élément de type ***ping***. Déplacer un manchot revient à modifier les propriétés d'un élément de type ***ping*** déjà existant.

Le constructeur **ping(i)** permet de créer un manchot sans voisin, en position (0,0) et ayant pour indice $ind = i$.

Les 3 premiers oiseaux de la horde ont les propriétés suivantes :

1^{er} manchot : $ind = 1$, $x = 0$, $y = 0$, $libre = [3,4,5,6]$
$2^{ème}$ manchot : $ind = 2$, $x = 1$, $y = 0$, $libre = [1,2,5,6]$
$3^{ème}$ manchot : $ind = 3$, $x = 0$, $y = 1$, $libre = [1,2,3,4]$

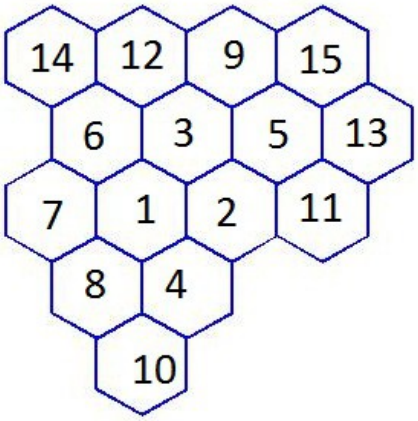


b. Présentation de la famille 'groupe' et du vecteur 'indfront'

Tout au long de la construction, nous avons besoin de travailler avec ces 2 éléments :

- ***groupe*** : contient tous les éléments de type ping déjà ajoutés à la horde
- ***indfront*** : contient les indices des manchots situés en bordure de la horde. Ce vecteur est ordonné de sorte à ce que lorsqu'on le parcourt, les manchots correspondants parcourent le contour de la horde dans le sens trigonométrique.

La fonction ***horde*** renvoie ces deux éléments une fois la construction terminée.
Voici un exemple de horde de 15 manchots avec le vecteur ***indfront*** correspondant :



Dans chaque hexagone apparaît l'indice du manchot correspondant.

Le vecteur ***indfront*** est dans cet exemple :

$[4 \;\; 2 \;\; 11 \;\; 13 \;\; 15 \;\; 9 \;\; 12 \;\; 14 \;\; 6 \;\; 7 \;\; 8 \;\; 10]$

c. Ajout d'un manchot à la horde

Pour ajouter un manchot à la horde, il s'agit donc d'ajouter un élément dans la famille ***groupe***, et de mettre à jour le vecteur ***indfront***. Il faudra veiller à conserver le bon ordre dans le vecteur ***indfront*** après l'ajout d'un manchot. Voici la méthode utilisée :

- On commence par choisir un manchot au hasard parmi ceux situés sur la frontière. Ceci équivaut donc à choisir aléatoirement une position i dans le vecteur ***indfront***.
- On essaye de placer un nouveau manchot qui serait le voisin du manchot choisi et du manchot correspondant à la position $(i+1)$ dans le vecteur ***indfront***. On utilise pour cela la fonction ***ajoute*** décrite dans la section 4.1.3 .

- Si la position ne génère pas de trou dans la horde, on y place un nouveau manchot
- Sinon, on choisit un autre indice i de manière aléatoire

Cette méthode assure bien le respect des contraintes C_1 , C_2 et C_3 .

4.1.3 Présentation de quelques méthodes d'implémentation

a. Description de la fonction *ajoute*

Lorsque nous souhaitons ajouter un manchot à la horde, on utilise la fonction ***ajoute*** ayant comme paramètres d'entrées les éléments suivants :

- ***indnew*** est l'indice du manchot que l'on ajoute à la horde
- ***indv1*** est l' indice d'un manchot de la horde auquel on souhaite ajouter le nouveau manchot
- ***dirv2*** permet de connaître le second manchot qui va accueillir un nouveau voisin. Il s'agit de la direction à choisir en partant du manchot ***indv1*** pour atteindre le deuxième voisin d'accueil. La direction est donnée par un entier (1 : Est, 2 : Nord-Est, ….. 6 : Sud-Ouest)
- Les deux éléments qui caractérisent la horde ***groupe*** et ***indfront***
- Un paramètre ***param*** qui vaut 1 ou 2 dont le rôle sera décrit dans cette section

Cette fonction retourne trois éléments :

- Les éléments ***groupe*** et ***indfront*** actualisés
- Un booléen ***ok*** qui permet de dire si un manchot a pu être ajouté ou non (ok=1 dans l'affirmative)

Il est important de noter que cette fonction sera utilisée dans deux configurations différentes :

Cas 1 : Lorsqu'on souhaite déplacer un manchot d'une position à une autre au cours de la simulation

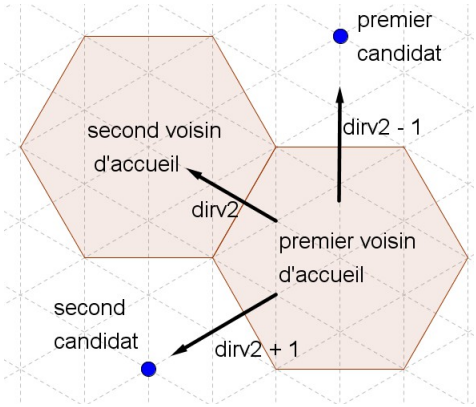
Cas 2 : Lors de la génération initiale de la horde

C'est le paramètre d'entrée ***param*** qui indique dans quel cas on se trouve au moment de faire appel à la fonction.

------> Connaissant ***indv1*** et ***dirv2***, on commence par déterminer la position d'accueil du nouveau manchot.

- Si nous sommes dans le cas 2, cette position se détermine très facilement. En effet, prenons l'exemple de la figure du 4.1.2.b. Supposons que l'on souhaite ajouter un seizième manchot à la horde et que le manchot choisi aléatoirement soit le manchot 9. Alors, avec notre méthode, le deuxième manchot à accueillir un nouveau voisin est le manchot d'indice 12 (celui qui suit 4 sur la frontière en la parcourant dans le sens trigonométrique). On a alors : $indnew = 16 \quad indv1 = 9 \quad dirv2 = 4$.
  On cherche désormais à déterminer la position ***dir*** qui définira le déplacement à effectuer en partant de ***indv1***, pour atteindre la position de ***indnew***.
  Étant donné le sens de rotation choisi, cette direction est définie de manière unique par la relation suivante : $dir = dirv2 - 1$ si $dirv2 > 1$ et $dir = 6$ si $dirv2 = 1$.
  Alors pour notre exemple, on a $dir = 3$.
- Si nous sommes dans le cas 1, la situation est un peu plus complexe car le deuxième voisin d'accueil ne sera pas forcément le manchot qui suit le premier voisin d'accueil sur la

frontière. En effet, il peut s'agir également du précédent. Ainsi, pour trouver la position qui soit à la fois voisine des deux manchots d'accueil, il y a deux possibilités, comme le montre la figure suivante :



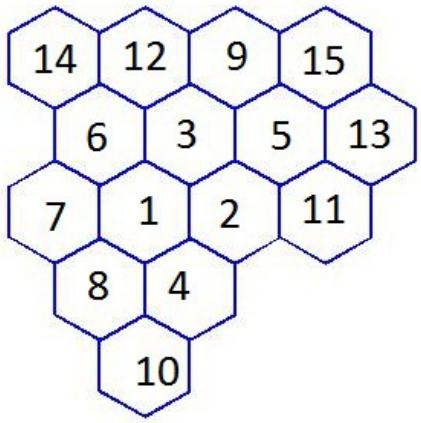
Parmi ces deux possibilités, seule une est à retenir car il y en a forcément une des deux qui est déjà occupée par un autre manchot. Il s'agit alors de tester laquelle des deux est libre afin de connaître la position d'accueil.

------> Nous connaissons désormais la position où nous allons tenter d'ajouter un manchot. Il faut maintenant vérifier qu'en y ajoutant un manchot, on ne génère pas de trou dans la horde (voir paragraphe **b** de cette partie)

- Si la position génère un trou, on sort de la fonction avec le booléen ***ok*** qui vaut 0. Les éléments ***groupe*** et ***indfront*** restant inchangés.
- Si la position ne génère pas de trou, alors on y place le manchot d'indice ***indnew*** en lui associant les coordonnées correspondantes. On passe alors à la phase de mise à jour de la horde.

-----> Mise à jour des éléments ***groupe*** et ***indfront***

- Les deux manchots d'accueil possèdent désormais un nouveau voisin. Il faut alors mettre à jour leur vecteur ***libre*** en y retirant la direction qui est maintenant occupée. En s'inspirant de la figure ci-dessous, et en supposant que l'on ajoute un manchot aux côtés des manchots 9 et 12, il faut alors effectuer les modifications suivantes :



$groupe(9).libre$ valait $[2\ \ 3]$. Il vaudra désormais : $[2]$

$groupe(12).libre$ valait $[2\ \ 3]$. Il vaudra désormais : $[3]$

- On met à jour le vecteur ***libre*** du manchot d'indice ***indnew.*** Il vaut dans cet exemple : $[1\ \ 2\ \ 3\ \ 4]$

- Il faut également mettre à jour le vecteur ***indfront*** :
  Pour expliquer la méthode utilisée, nous nous basons désormais sur un autre exemple :
  En considérant la figure ci-dessus, nous ajoutons un manchot aux côtés des manchots d'indice 14 et 6. Nous remarquons alors que le manchot 6 doit quitter indfront et que le nouveau manchot doit s'intercaler entre le 14 et le 7.

La transformation du vecteur ***indfront*** est alors la suivante :

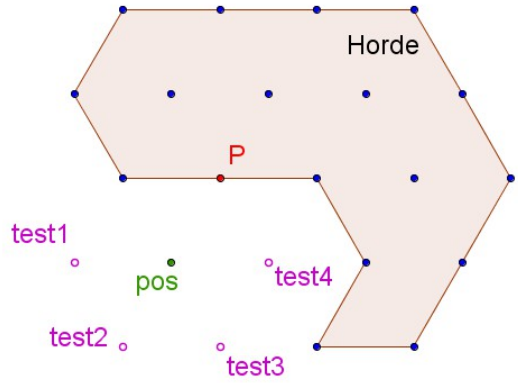
[4 2 11 13 15 9 12 14 6 7 8 10] devient [4 2 11 13 15 9 12 14 16 7 8 10]

Il faut trouver une méthode qui soit compatible avec toutes les situations. En effet, il peut y avoir parfois un nombre différent de manchots qui quittent la frontière (0 ou 2).
Nous commençons par détecter l'ensemble des manchots qui sont voisins de ***indnew*** (ici les manchots 14, 6 et 7) à l'aide de la fonction ***detect***. Parmi ces manchots, nous repérons les deux qui se trouvent aux extrémités du chapelet formé par l'ensemble des manchots détectés (ici les manchots 14 et 7).

Ensuite, nous enlevons de ***indfront*** les éventuels manchots qui parmi le chapelet ne sont pas aux extrémités (ici le manchot 6).
Enfin, nous ajoutons le manchot d'indice ***indnew*** entre les deux extrémités obtenues précédemment.

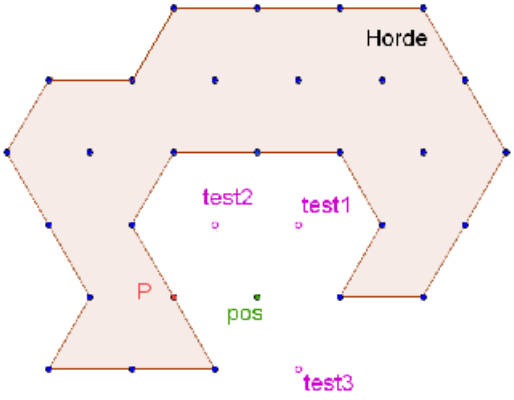
b. Détection d'un trou dans le maillage

Ce qui est présenté ci-dessous correspond au contenu de la fonction ***trou***.
On considère la horde ci-dessous. Les points bleus représentent les manchots de la horde. ***P*** est le manchot de la horde auquel on souhaite fixer un nouveau voisin en position ***pos***.



L'idée est de parcourir toutes les positions libres autour de ***pos*** afin de vérifier qu'elles ne sont pas "enfermées" par les autres manchots de la horde. Pour cela, on utilise la fonction ***enferme*** qui fonctionne de la manière suivante :
Pour savoir si la position ***test4*** serait enfermée après l'arrivée d'un manchot en ***pos***, on procède ainsi :
→ On explore les 6 directions autour de ***test4***
→ On regarde si en se déplaçant en ligne droite dans une direction donnée, on finit par rencontrer un manchot (voir fonction ***testdir***)
→ Si toutes les directions mènent à un manchot, alors ***test4*** est enfermée (Ce n'est pas le cas ici car la direction SW est libre)
Pour ce premier exemple, la position ***pos*** serait acceptée car les 4 places testées ne sont pas enfermées.

Sur le second exemple à droite, on remarque que les positions ***test1*** et ***test2*** seraient enfermées. Il y a création d'un trou dans le maillage, la position ***pos*** est donc refusée.



c. Test de la présence d'un manchot dans une direction donnée

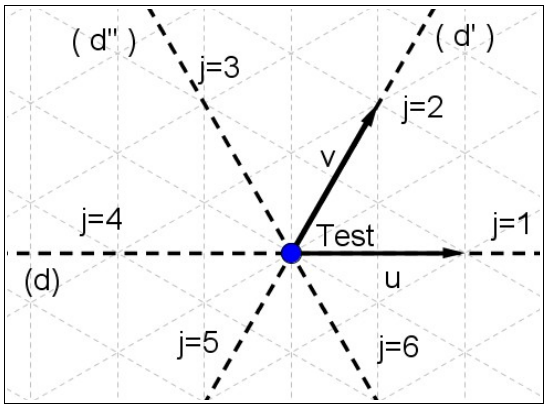
Comme nous l'avons dit précédemment, la construction de la fonction trou nécessite de savoir résoudre le problème suivant :

Partant d'une position *test* ***, si l'on se déplace dans une direction donnée, allons-nous rencontrer un manchot sur notre chemin ?***

Pour répondre à cette question, la fonction ***testdir*** nécessite les arguments suivants :

- *test* est un point de R^2 correspondant à la position de départ.
- *j* est un entier correspondant à une direction (1 : Est, 2 : Nord-Est, … 6 : Sud-Est)
- les éléments ***groupe*** et ***indfront*** définis précédemment

Cette fonction retourne un booléen qui vaut 1 si un manchot est détecté, 0 sinon.
La figure suivante illustre les notations :



Le repère $(Test, \vec{u}, \vec{v})$ correspond au repère présenté dans la partie 1.2.2.

On remarque que dans ce repère, on obtient les équations de droites suivante :

$$\begin{cases} (d): & y=0 \\ (d'): & x=0 \\ (d''): & y=-x \end{cases}$$

Ainsi, nous regardons pour chaque manchot *P* de position *(P.x , P.y)* situé sur la frontière, si il vérifie la condition suivante (la position de *Test* est notée *(Tx,Ty)*) :

Direction *j*	**Condition**
1	$P.y = Ty$ et $P.x > Tx$
2	$P.x = Tx$ et $P.y > Ty$
3	$P.y - Ty = -(P.x - Tx)$ et $P.y > 0$
4	$P.y = Ty$ et $P.x < Tx$
5	$P.x = Tx$ et $P.y < Ty$
6	$P.y - Ty = -(P.x - Tx)$ et $P.y < 0$

Si la condition correspondant à la direction ***j*** donnée est vérifiée par un manchot *P*, la fonction ***testdir*** renvoie le booléen 1.