4.2 Détermination du manchot qui se déplace et de sa nouvelle position

4.2.1 Modèle de détermination.

Ici, on se pose la question du choix du déplacement du manchot, c'est à dire : Lequel part ? Et où ira-t-il ?

L'idée principale est de faire bouger le manchot qui a le plus froid. Il ira se placer à la position qui est considérée comme la plus chaude. Nous avons donc besoin d'un critère qui définira le fait qu'un manchot ait plus ou moins froid.

Nous définissons alors le critère nommé ***Heatloss*** de la manière suivante :
On considère la dérivée normale de la température $\frac{\partial T}{\partial n}$ en tout point du polygone symbolisant la frontière de la horde. Ce polygone peut être découpé en plusieurs portions A_i qui sont chacunes propres à un manchot P_i . Pour chaque manchot de la frontière, on définit alors le critère par une intégrale curviligne sur la portion A_i :

$$\boxed{Heatloss(P_i) = -\int_{A_i} \frac{\partial T}{\partial n}}$$

Pour les manchots situés à l'intérieur de la horde, le critère ***Heatloss*** est considéré nul.

On obtient alors une quantité qui nous semble naturellement positive étant donné que la température devrait être maximale sur la frontière du domaine. On considère que plus le critère est élevé, plus la perte de chaleur est grande, et donc plus le manchot a froid.

Ainsi, nous souhaitons faire bouger le manchot ayant le Heatloss le plus élevé pour le placer auprès de celui dont le ***Heatloss*** est le plus faible (manchot noté P_{chaud}). Il nous reste plusieurs choix possibles pour la destination d'arrivée. Nous choisissons alors parmi les voisins frontaliers de P_{chaud} , celui dont le ***Heatloss*** est le plus bas. Ayant déterminé les deux manchots successifs qui accueilleront un nouveau voisin, la position est bien définie de manière unique.

N'oublions pas que le modèle inclut des contraintes quant à la constitution de la horde. Ainsi, la logique que nous définissions ci-dessus peut être sensiblement refrénée par les contraintes C_1 , C_2 et C_3 posées dans la partie 4.1.

En ce qui concerne le manchot qui se déplace, ceci implique deux cas à considérer :

- Le manchot partant ne doit pas laisser un manchot avec un seul voisin.
- Le départ du manchot ne doit pas créer un scindement de la horde en deux parties.

Si un manchot ne respecte pas ces deux contraintes, alors on choisit le manchot qui a la plus grosse perte de chaleur sans prendre en compte celui déjà analysé.

Vient ensuite le problème de l'arrivée d'un manchot :
Ce cas-là n'est pas très difficile car ses contraintes sont les mêmes que lors de la génération de la horde, c'est à dire :

- Ne pas créer de trou dans le polygone.
- Le manchot déplacé devra avoir au minimum deux voisins.
- Le manchot déplacé doit être adjacent à la horde.

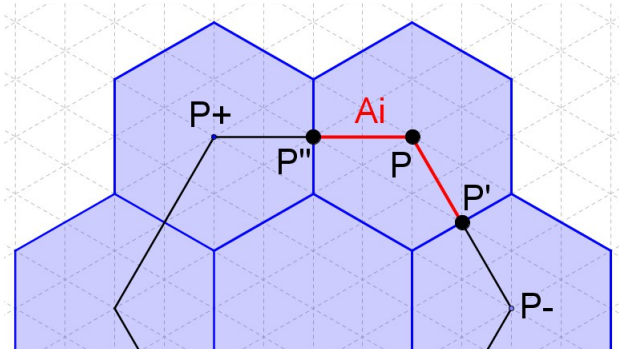
Ces contraintes ont déjà été gérées dans la partie 4.1.3.
De la même façon que lors du choix du manchot qui part, si nous rencontrons une position qui n'est pas acceptable, nous choisissons un autre manchot d'accueil P_{chaud} qui est celui qui a le plus chaud, hormis le manchot déjà analysé.

4.2.2 Calcul numérique du taux de perte de chaleur.

Pour calculer la perte de chaleur de chaque manchot afin de déterminer lequel bougera, nous avons créé une fonction sous Matlab appelée ***Heatloss***. Cette fonction renvoie dans un vecteur les pertes de chaleur pour chacun des manchots de la frontière.

a. Détermination de la portion de polygone A_i

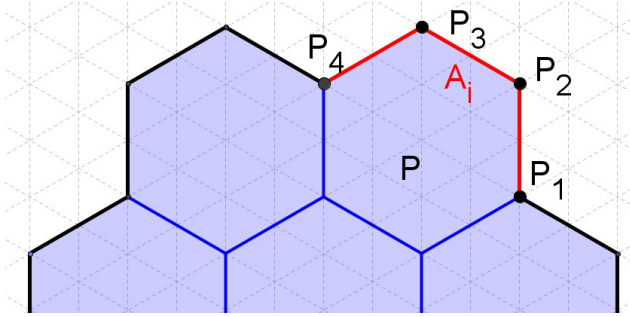
La nature de cette région dépend du modèle choisi pour tracer le polygone qui matérialise la horde.



------> Dans le cas du modèle où les sommets sont les centres des hexagones :

La portion de polygone A_i est constituée de deux segments $[P'P]$ et $[P\ P'']$.

------> Dans le cas du modèle où le polygone décrit le contour parfait de la horde :



La portion de polygone A_i est constituée de plusieurs segments de type $[P_j P_{j+1}]$. Leur nombre dépend du manchot P. Il varie entre 1 et 4.

Dans les deux cas, nous avons besoin du calcul numérique d'intégrales du type $\int_{B_1}^{B_2} \frac{\partial T}{\partial n}$ lorsque B_1 et B_2 sont des points situés sur la frontière de la horde.

b. Approximation de la dérivée normale

Afin de bien comprendre l'approximation réalisée, rappelons le résultat de la section 2.3.4 :

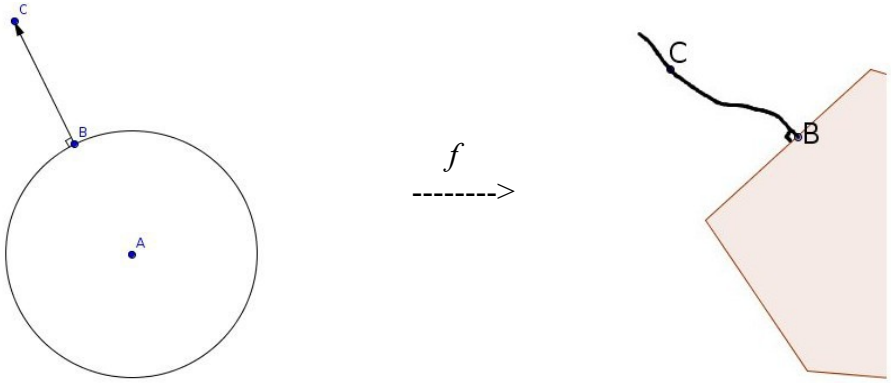
$$\forall \omega \in \Omega, \quad T(\omega) = \tilde{T}(z) \quad \text{si} \quad z = f^{-1}(\omega)$$

où, f est l'application qui envoie l'extérieur du disque unité sur l'extérieur du polygone(Ω),
T désigne la température autour du polygone,
$\tilde{T}$ désigne la température autour du disque unité.

Étant donné ce résultat, mais également la nature du maillage utilisé autour du disque, on souhaite approcher $\frac{\partial T}{\partial n}$ de la façon suivante : $\frac{\partial T}{\partial n}(\omega) \simeq \frac{\tilde{T}(1+\Delta r, \theta_\omega) - \tilde{T}(1, \theta_\omega)}{\Delta r}$

où Δr désigne le pas radial du maillage choisi et θ_ω l'argument correspondant au point z qui vérifie $z = f^{-1}(\omega)$.

En réalité, la normale en un point du polygone n'est pas accessible sur notre maillage à cause de la déformation visible sur le schéma de la section 3.1.
En effet, comme le montre le schéma ci-dessous, l'image C' d'un point C qui était situé sur la normale au cercle en un point B n'est pas forcément située sur la normale au polygone en $f(B)$.



Mais étant donné que l'application est conforme, même si les lignes de maillage sont déformées, nous conservons l'angle droit en $f(B)$. Ainsi, en choisissant un pas radial très petit, l'approximation sera très fiable.
Le calcul de l'intégrale aura donc lieu sur le domaine extérieur au cercle.

c. Approximation numérique de l'intégrale

Au vu de l'approximation décrite dans le paragraphe précédent, il nous reste à définir la méthode permettant d'approcher la quantité :

$$I = \int_{\theta_1}^{\theta_2} N(\theta).d\theta \qquad \text{où} \qquad N(\theta)=\frac{\tilde{T}(1+\Delta r,\theta)-\tilde{T}(1,\theta)}{\Delta r}$$

et où θ_1 et θ_2 désignent les arguments respectifs de z_1 et z_2 vérifiant :
$z_1=f^{-1}(\omega_{B_1})$ et $z_2=f^{-1}(\omega_{B_2})$

Tout d'abord, notons que nous ne disposons des valeurs des températures que pour les points du maillage. Ainsi, nous définissons les valeurs θ_1' et θ_2' qui correspondent aux angles des points du maillage les plus proches de θ_1 et θ_2 .

Nous représentons la situation sur le schéma suivant. Nous avons choisi de représenter la portion du cercle par un segment dans un souci de clarté :

Dans un premier temps, nous calculons une première approximation de l'intégrale par la méthode des rectangles sur le segment $[\theta_1', \theta_2']$:

$$I_0=\sum_{\theta=\theta_1'}^{\theta_2''} N(\theta).\Delta\theta$$

Ensuite, afin de tenir compte du décalage entre θ_1' et θ_1 et de celui entre θ_2' et θ_2 , nous allons ajouter ou soustraire, selon les cas, les quantités suivantes :

$$écart_1=N(\theta_1').(\theta_1'-\theta_1) \qquad \text{et} \qquad écart_2=N(\theta_2').(\theta_2'-\theta_2)$$

En procédant ainsi, la méthode est au moins exacte pour les fonction constantes.

4.2.3 Modifications de la horde virtuelle.

Le critère ***Heatloss*** étant connu pour chaque manchot de la frontière, nous devons à présent appliquer la méthode de déplacement de manchot définie en 4.2.1.

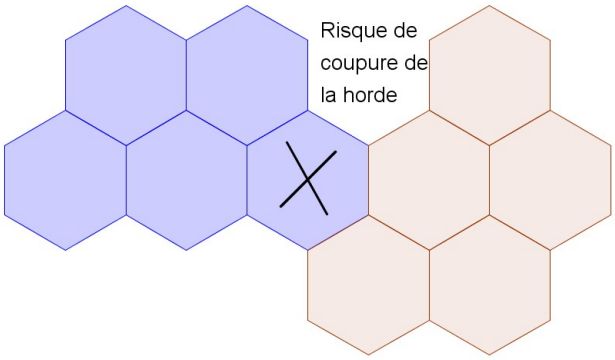
Dans la pratique, ce déplacement se fait en deux étapes. Avec Matlab, on a alors créé une fonction ***enleve*** pour enlever un manchot de sa position et mettre à jour les données de ses anciens voisins. Puis on a utilisé la fonction ***ajoute*** présentée dans la partie 4.1, pour le mettre à la place où il doit aller.

Comme nous l'avons vu dans la section 4.2.1, la fonction ***ajoute*** a déjà été créée et est prévue pour répondre à toutes les contraintes. Voici les grandes lignes de la méthode utilisée au sein de la fonction ***enleve***.

Cette fonction prend comme arguments d'entrée l'indice ***ind*** du manchot qu'il faut enlever, ainsi que les éléments caractéristiques de la horde ***groupe*** et ***indfront***. Elle retourne ces deux derniers éléments après les avoir mis à jour mais également un booléen ***ok*** qui permet de savoir si le manchot ***ind*** a bien pu être retiré de la horde.

------> Tout d'abord, nous vérifions que le fait d'enlever le manchot ***ind*** n'entraînerait pas qu'un de ses voisins se retrouve avec un seul voisin. La condition C_2 serait alors violée. Pour cela, nous détectons l'ensemble des voisins du manchot ***ind*** et pour chacun d'entre-eux, nous vérifions que le vecteur ***libre*** correspondant est constitué d'au plus 3 éléments. Si cette condition n'est pas vérifiée pour chaque voisin, la fonction retourne le booléen 0.

------> Ensuite, nous souhaitons éviter la configuration représentée sur la figure suivante au sein de la horde, car cela serait propice à une séparation de cette dernière en deux parties.



Remarquons que l'analyse du vecteur ***libre*** du manchot marqué d'une croix nous permet de repérer automatiquement une telle situation.

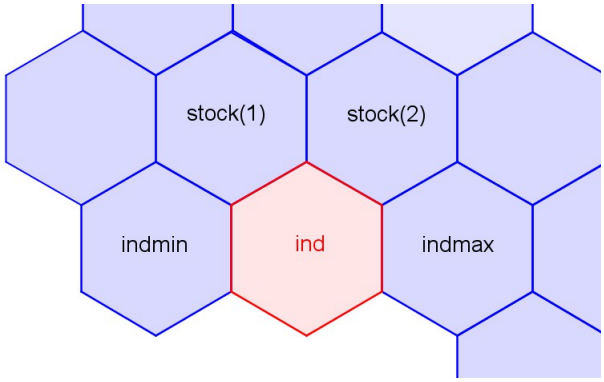
En effet, une fois ordonné, ce vecteur serait le suivant : $libre = [2 \;\; 5]$. On remarque qu'il y a un écart supérieur à 1 entre deux valeurs successives de ***libre***. Ceci est caractéristique de la situation à exclure, mis à part quand cet écart est dû à la présence simultanée de 1 et 6 dans le vecteur libre. (exemple : $libre = [1 \;\; 5 \;\; 6]$ pour le manchot situé en bas à droite n'est pas une situation à exclure)

Dans le cas où la situation est à exclure, la fonction ***enleve*** retourne le booléen 0.

------> Nous sommes désormais dans le cas où le fait de retirer le manchot ***ind*** ne crée pas de dysfonctionnement au sein de la horde. Ce dernier est donc effectivement retiré. Il faut désormais mettre à jour les éléments ***groupe*** et ***indfront***. La première mise à jour évidente est de retirer l'indice ***ind*** du vecteur ***indfront***. Les autres caractéristiques de ce manchot seront modifiées en fin de fonction : ***libre*** devient [1 2 3 4 5 6] et on attribue aux coordonnées de ***ind*** des valeurs aberrantes ($groupe(ind).x = groupe(ind).y = 99\,999$)
Il faut également modifier les vecteurs ***libres*** des manchots qui se voient enlever un voisin.

Pour bien comprendre la mise à jour à effectuer au sein du vecteur ***indfront***, nous considérons ici l'exemple décrit sur la figure suivante :

On remarque que plusieurs éléments doivent venir s'ajouter au vecteur ***indfront*** (ici stock(1) et stock(2)).
Leur nombre dépend de la configuration obtenue et varie entre 0 et 3.



Tous ces éléments doivent s'intercaler entre les indices qui étaient déjà frontaliers : ***indmin*** et ***indmax***.

La méthode est donc la suivante :
Après avoir enlevé l'élément sortant, le vecteur ***indfront*** sera de la forme :
$indfront = [..\ ..\ ..\ ..\ indmin\ \ indmax\ \ ..\ ..]$ ou encore $indfront = [indmax\ \ ..\ ..\ ..\ indmin]$

Afin de simplifier le problème, on se ramènera systématiquement à la deuxième forme en réordonnant le vecteur ***indfront***.

Le manchot d'indice stock(1) (s'il existe) doit être le nouveau voisin frontalier du manchot d'indice ***indmin***. Il se trouvera dans la position qui suit ***indmin*** dans le vecteur ***indfront***. Le manchot d'indice stock(2) devrait quant à lui précéder celui d'indice ***indmax***. Enfin, s'il existe un troisième manchot noté stock(3), alors il devra se trouver entre stock(1) et stock(2).

Voici donc le nouveau vecteur ***indfront*** après mise à jour :

$$indfront = [(stock_2),\ indmax\ \ ..\ ..\ ..\ indmin,\ (stock_1),\ (stock_3)]$$