RAGONNET Romain
JUMENTIER Romain

Master1 Ingénierie Mathématiques
Promotion 2012-2013

Travail Encadré de Recherche

Sujet :

La marche de l'empereur

Encadré par Monsieur
Bernhard Beckermann

Table des matières

1. INTRODUCTION

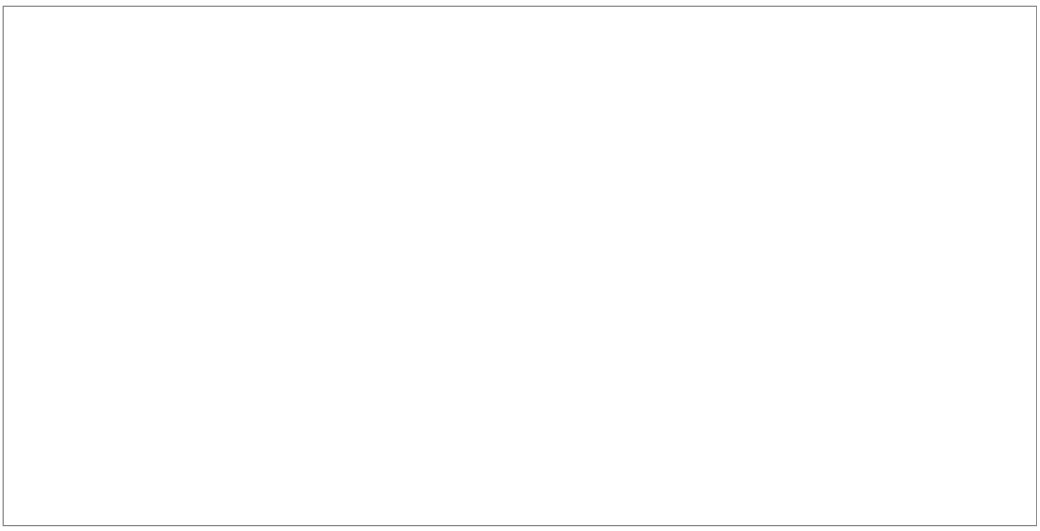
1.1 Présentation du problème

1.1.1 Le phénomène de la tortue

Dans ce Travail Encadré de Recherche, nous avons choisi de traiter le sujet intitulé ***" La marche de l'empereur "***. Ce titre fait référence au manchot empereur vivant en Antarctique. Nous allons plus précisément étudier un phénomène particulier qui est propre à cette espèce d'oiseau. Avant de présenter ce phénomène, intéressons-nous aux conditions de vie du manchot empereur.

Tout d'abord précisons que le manchot empereur est un oiseau qui vit en colonie. Cette dernière qui peut comporter plusieurs milliers d'individus est plus précisément appelée une horde. Étant donnée la zone géographique où il vit, le manchot est confronté durant les mois d'hiver à des conditions climatiques très rudes. En effet, la température atteint couramment -40°C et le vent peut souffler jusqu'à 150 km/h. Une des explications au fait qu'il survit très bien dans un domaine si hostile est la particularité de son plumage qui est le plus dense de tous les plumages d'oiseau (15 plumes par cm^2). Seulement, en cas de conditions extrêmes, le plumage ne suffit pas à lui seul à maintenir la température interne du manchot à 39°C. Ce dernier adopte alors une stratégie particulière durant les périodes de grand froid.

Nous en arrivons alors au sujet d'étude de ce TER. Afin de maintenir leur température corporelle, tous les manchots d' une horde se regroupent afin de former un amas compact d'oiseaux. Ce phénomène est appelé "tortue" par les ornithologues car comme nous le voyons sur la figure ci-dessous, la horde de manchots forme une espèce de grosse carapace très dense.



Au sein de cette tortue, la densité peut atteindre jusqu'à 8 oiseaux au m^2.

Parmi les manchots de la horde, certains sont beaucoup plus exposés au froid que d'autres. En effet, nous imaginons très bien qu'un manchot situé en bordure de horde aura plus froid qu'un manchot qui est situé au centre. De même, l' intuition nous fait penser que parmi les oiseaux du bord, ceux qui auront le plus froid sont ceux situés du côté le plus exposé au vent.

Les biologistes expliquent alors que lorsqu'un manchot a trop froid, il se déplace afin de trouver une position où les conditions sont plus acceptables. Ils constatent que suite aux différents déplacements individuels, la horde voit sa forme être sensiblement modifiée. De plus, ils remarquent que l'ensemble de manchots avance sur la banquise en suivant apparemment le sens du

vent. Ces déplacements peuvent prendre une ampleur conséquente étant donné que les manchots font parfois la tortue durant des temps très longs (plusieurs jours sans interruption).

1.1.2 Objectifs de l'étude

Au vu du phénomène présenté précédemment, l'idée est alors de modéliser la situation afin de pouvoir expliquer et prédire les différents mouvements de la horde. Les trois chercheurs américains Aaron Waters, François Blanchette et Arnold D. Kim se sont penchés sur ce problème avant nous et nous disposons d'un résumé de leur étude sur lequel nous pouvons nous baser. Le but ici est de refaire la modélisation par nous-même en s'inspirant de ce résumé, notamment pour ce qui est de la définition du modèle et de ses hypothèses. Dans l'article des chercheurs, sont exposées les méthodes utilisées de manière très générale. Notre tâche consistera dans un premier temps à développer la théorie qui permet de les mettre en application.

Ensuite, nous devrons mettre en place la partie programmation qui elle n'est pas mentionnée dans l'article. Le résumé qui est à notre disposition traite en grande partie les résultats obtenus après simulation. Cela nous permettra de comparer nos propres résultats avec ces derniers. Les chercheurs proposent un modèle particulier que nous tenterons de reproduire en nous posant systématiquement la question de sa cohérence et en tentant d'en améliorer certains points. Mais notre but est aussi de proposer un autre modèle qui nous semble plus naturel afin de comparer les conclusions obtenues.

L'objectif est donc de créer une simulation qui génère une tortue et la fait évoluer au cours du temps en fonction de différents paramètres comme par exemple la vitesse du vent. Afin d' y parvenir, nous devons être capable de déterminer la température aux alentours de la horde pour savoir quel manchot a le plus froid et se déplacera. La connaissance de la température nécessite bien sûr la connaissance du vent car il y a une interaction entre ces deux grandeurs.

Une grande partie du problème consiste alors à calculer ces quantités sur le domaine situé autour de la horde. Ce dernier est très particulier par deux aspects :

→ La forme même du domaine est irrégulière
→ Le domaine change à chaque fois qu'un manchot se déplace

1.2 Présentation des recherches

1.2.1 Modèle et hypothèses

Pour réaliser cette étude, nous avons besoin de définir le modèle utilisé, ainsi que les hypothèses nécessaires à la modélisation.

a. Modèle d'évolution de la horde

L' hypothèse principale sur laquelle reposera l'étude est la suivante :
Au sein de la horde, chaque individu souhaite augmenter sa propre température corporelle. Les manchots ne cherchent donc pas à augmenter la température globale de la horde. Ils raisonnent en tant qu'individus et non en tant que membres d'un groupe.

Pour cela, nous considérons la modélisation suivante concernant l'évolution de la horde :
→ Au début de l'expérience, la horde est constituée d'un ensemble de manchots qui sont tous en contact très serré les uns avec les autres. La horde d'oiseaux forme alors un ensemble compact de sorte à ne pas y laisser entrer le vent. De plus, chaque manchot est en contact avec au moins deux

autres manchots afin de se tenir au chaud.
→ Tous les manchots restent immobiles jusqu'à ce que l' un d'entre-eux ait trop froid. Il s'agit naturellement d'un manchot situé en bordure de horde. Lorsque le manchot en question ne supporte plus les conditions climatiques qui l'entourent, il décide de se déplacer afin de trouver une zone plus clémente. Il choisit alors la position la plus chaude sur le pourtour de la horde car il ne peut pénétrer en son intérieur du fait de la compacité de cette dernière.
→ Après ce déplacement, le groupe d'oiseaux connaît un nouvel état d'équilibre jusqu'à ce que le phénomène se reproduise.

Dans ce modèle, nous supposons qu'un seul manchot se déplace à la fois. Nous supposons également que tous les individus ressentent le froid de la même façon.

Le nombre de manchots considérés ne comptabilise que les individus adultes car les petits sont couvés au sein de la horde, sous les individus adultes. Ainsi, les manchots étudiés dans ce problème peuvent tous être considérés de même taille. Au cours du temps de simulation, le nombre d'oiseaux reste constant.

La horde forme toujours qu'une seule partie. Elle ne peut se scinder.

b. Hypothèses concernant le vent et la température

Tout d'abord, étant donné que les manchots vivent sur la banquise, on peut considérer qu'ils se trouvent sur un plan parfait. De plus, au vu de leur petite taille (<120cm), nous négligerons les effets liés à l'altitude concernant le vent et la température. Même s'il est clair que le vent qui passe au dessus de la horde contribue à refroidir les manchots, nous considérons qu'il y contribue de la même manière pour tous les individus. Ce n'est donc pas un phénomène à prendre en compte. Ainsi, notre domaine de résolution sera un domaine du plan.

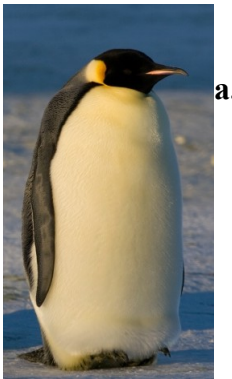
Dans cette étude, nous nous intéressons au cas où le vent est constant au large de la horde. De plus, nous supposons qu'il n'est pas tourbillonnant, même en présence de l'obstacle que représente la horde.

La température au sein de la horde est supposée être la même en tout point et ne varie pas au cours du temps de simulation. Nous verrons alors dans la partie 4.2 de quelle manière nous évaluons le sentiment de froid qu'a un manchot.

La température dans les zones éloignées de la horde connaît une répartition uniforme et sa valeur ne change pas au cours du temps.

Toujours concernant la température, nous considérons qu'il n'y a aucune source de chaleur en présence sur le domaine d'étude. Nous négligeons ainsi par exemple la chaleur que pourrait renvoyer les manchots à l'extérieur de la horde.

Enfin, nous considérons que la variable temps n'intervient pas dans la modélisation car au moment de la détermination des vents et températures, la horde est toujours à l'équilibre.



1.2.2 Structure de la horde virtuelle

a. Départ de constats simples

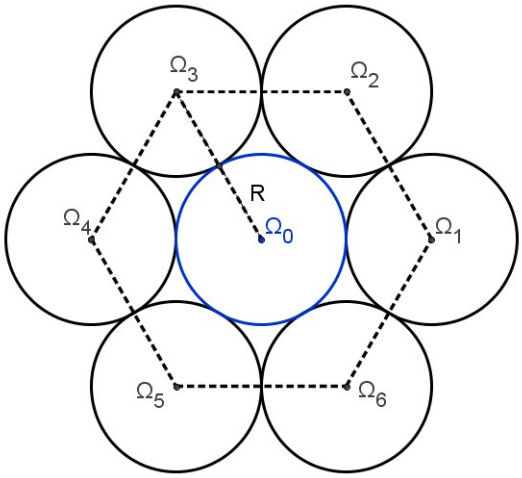
Le corps d'un manchot a une physionomie assez régulière, assimilable

à une forme cylindrique. Nous nous intéressons ici à une visualisation du manchot dans un plan horizontal. Il est alors naturel de modéliser un individu par un disque.

D'autre part, on considère dans ce modèle que les manchots sont tous de même taille. Notre horde sera donc constituée d'un ensemble de disques de même rayon.

Pour pouvoir augmenter leur température interne, les manchots se tiennent très serrés. De cette manière, chaque manchot est en contact direct avec ses voisins. La horde est alors un **ensemble connexe du plan, constitué de disques de même rayon et juxtaposés**.

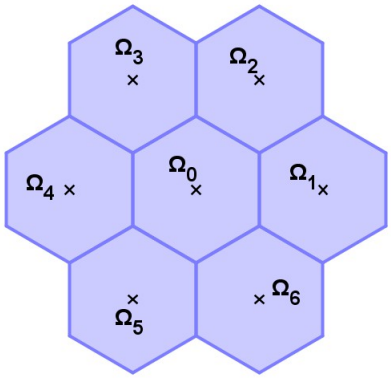
b. Structure de la horde



On considère un disque D_0 du centre de la horde. On note Ω_0 son centre et R son rayon. Étant au cœur du groupe d'oiseaux, D_0 est entouré de voisins qui sont des disques D_i de centre Ω_i. Les point Ω_i sont donc situés à une même distance de $2R$ de Ω_0 et sont entre-eux distants de cette même longueur (car les disques sont tous tangents). On obtient alors un hexagone $\Omega_1\Omega_2\Omega_3\Omega_4\Omega_5\Omega_6$ et on peut alors conclure qu'un manchot du centre de la horde possède 6 voisins.

Ainsi, la horde sera décrite par un ensemble de points situés sur les nœuds d'un maillage hexagonal.

Afin de rendre compte de la compacité de la horde, nous avons décidé de remplacer les disques par les hexagones présents sur la figure ci-contre. En effet, les manchots sont tellement serrés les uns contre les autres que leur corps épouse la forme des manchots voisins.



c. Système de repérage

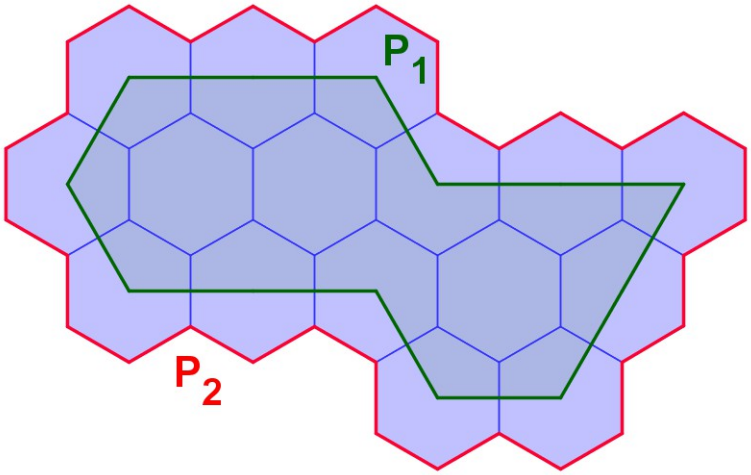
De par la forme du maillage, il est naturel d'introduire le repère $(\Omega_0, \overrightarrow{\Omega_0\Omega_1}, \overrightarrow{\Omega_0\Omega_2})$. Dans ce repère, les voisins de Ω_0 ont les coordonnées suivantes :

Point	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5	Ω_6
Voisin	E	NE	NW	W	SW	SE
Coordonnées	(1,0)	(0,1)	(-1,1)	(-1,0)	(0,-1)	(1,-1)

1.2.3 Domaine de résolution

Comme nous l'avons mentionné dans la partie 1.1.2, une grande partie de l'étude portera sur la détermination de la température et du vent autour de la horde de manchot. Ayant présenté la géométrie de la horde et au vu des hypothèses posées en 1.2.1, nous pouvons désormais présenter le domaine de résolution du problème.

Considérons la horde de manchots représentée ci-dessous :



Comme nous le verrons dans la suite de l'étude, nous avons choisi de tester deux modèles utilisant des domaines de résolution différents.

Pour le premier modèle, nous choisissons le polygone P_1 passant par les centres des manchots situés en bordure de horde.

Pour le second modèle, nous considérons le polygone P_2 qui décrit le contour de la horde.

Nous définissons alors le domaine de résolution Ω qui dans les deux modèles correspond à l'extérieur du polygone (P_1 ou P_2). Ce domaine est alors un sous ensemble connexe et non borné du plan. Nous observerons par la suite que grâce à l'hypothèse d'uniformité du vent et de la température au large de la horde, le fait de travailler sur un domaine non-borné n'est pas un obstacle à la résolution.

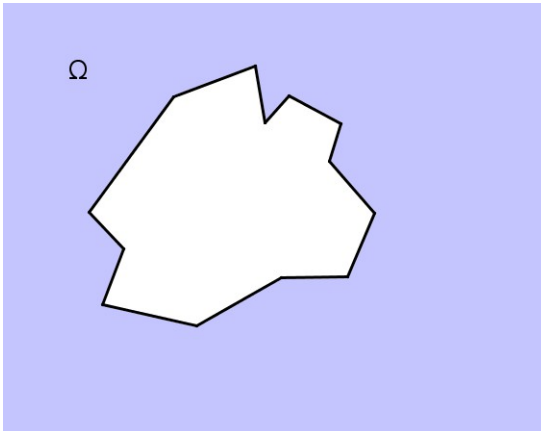
Ainsi, on constate que le problème possède deux particularités qu'il faudra savoir gérer :
---> Le domaine de résolution a une forme assez complexe et plutôt irrégulière
---> Le domaine de résolution change à chaque fois qu'un manchot modifie sa position.

2. LE PROBLÈME PHYSIQUE

2.1 Applications conformes

2.1.1 Motivations

Comme présentée dans la partie 1.2.3, une grande part du problème consiste à déterminer la distribution du vent et de la température dans un domaine particulier : l'extérieur d'un polygone. Dans la suite, nous noterons ce domaine Ω .



La horde de manchots ayant une forme très aléatoire, on ne peut se satisfaire de résoudre le problème sur l'extérieur de polygones ayant une géométrie simple tel que le carré par exemple. Nous avons donc besoin d'une méthode plus générale qui permette de travailler avec n'importe quel polygone non-croisé, quelque soit son nombre de sommets.

L'idée est donc de résoudre le problème sur un domaine simple, pour lequel la solution est accessible analytiquement, pour ensuite la transférer sur notre domaine d'étude par le biais d'une transformation du plan.

Par exemple, nous verrons par la suite que nous sommes en mesure de donner la solution pour le vent autour d'un obstacle de type segment. L'enjeu serait alors de trouver une application qui transforme l'extérieur d'un segment en l'extérieur de notre polygone. Il faudra de plus être capable de déterminer la solution sur Ω , connaissant la solution sur le domaine de départ.

Nous allons voir que le problème pourra être résolu grâce aux applications conformes.

2.1.2 Définitions et propriétés

Nous avons choisi pour l'étude des transformations du plan d'utiliser l'analyse complexe.

Définition :
Soit U un ouvert de C
Une application $f : U \subset C \rightarrow C$ est dite conforme si elle conserve les angles.

Propriété :
La composition de deux applications conformes est elle-aussi une application conforme.

Nous avons également la propriété suivante d'après le livre *"Introduction à la géométrie hyperbolique et aux surfaces de Riemann"* de Eric Toubiana et Ricardo Sá Earp (p45).

Propriété :
On a l' équivalence suivante :
f est conforme sur U *si et seulement si* f est holomorphe sur U et $\forall z \in U,\ f'(z) \neq 0$

Le fait que la transformation utilisée soit holomorphe est lourd de conséquences car on peut alors utiliser les équations de Cauchy-Riemann.

En notant $f(z)=P(x,y)+iQ(x,y),\quad \forall z=x+iy\in U$, on a :

$$P_x(x,y)=Q_y(x,y) \quad \text{et} \quad P_y(x,y)=-Q_x(x,y)$$

et les fonctions P et Q sont des fonctions harmoniques (leur Laplacien est nul).

Ce sont en grande partie ces équations qui nous permettront de transférer une solution d'un domaine à un autre à l'aide d'une application conforme. (voir 2.2.3 et 2.3.2)

2.1.3 Transformation de Joukowski

Définition :
La transformation de Joukowski est définie par :

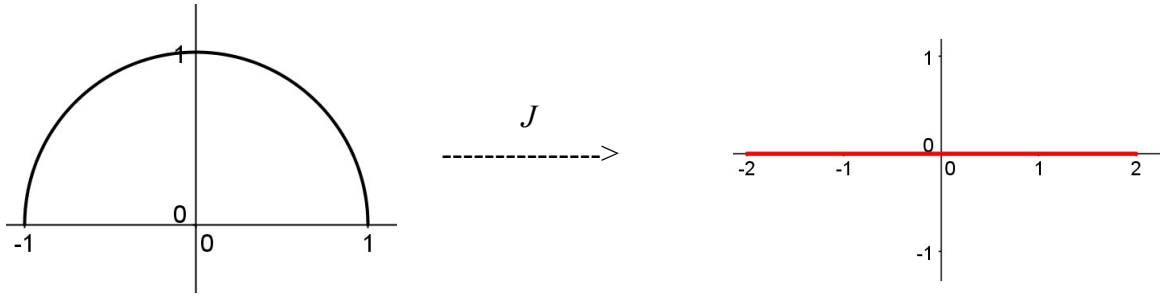
$$J(z)=z+\frac{1}{z} \quad , \quad \forall z\neq 0$$

Cette application est holomorphe sur $C-\{0\}$ car :

$$\forall z\neq 0 \quad , \quad J'(z)=1-\frac{1}{z^2}$$

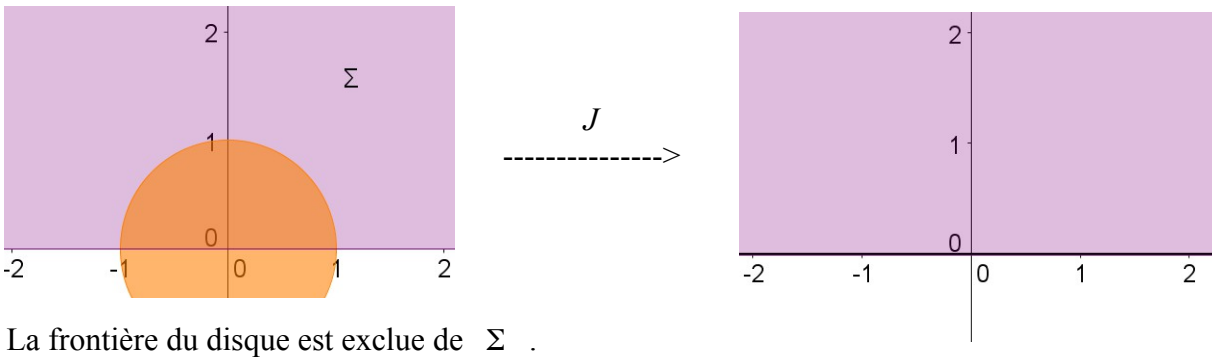
De plus, $\forall z\notin\{-1,0,1\}$, $J'(z)\neq 0$, donc cette transformation est une application conforme sur tous les domaines ouverts du plan complexe n'incluant ni 0 ni -1 ni 1 .

Propriété :
La transformation de Joukowski est une bijection qui transforme le demi-cercle unité supérieur en un segment borné aux points d'affixes respectives -2 et 2.



Preuve :
Soit z un point du demi-cercle unité supérieur. Alors, $\exists\theta\in[0,\pi]$, tel que $z=e^{i\theta}$.
Ainsi, $J(z)=e^{i\theta}+e^{-i\theta}=2\cos(\theta)$
Alors quand θ décrit $[0,\pi]$, $J(z)$ décrit $[-2,2]$ □

Propriété :
La transformation de Joukowski transforme la portion de plan Σ du schéma suivant en le demi-plan strictement supérieur.



La frontière du disque est exclue de Σ .

Preuve :
Notons que $\forall z = x + iy \neq 0$, on a $J(z) = \alpha(x, y) + i\beta(x, y)$
où :

$$\alpha(x, y) = \frac{x(x^2 + y^2 + 1)}{x^2 + y^2} \quad \text{et} \quad \beta(x, y) = \frac{y(x^2 + y^2 - 1)}{x^2 + y^2}$$

Si $M(z) \in \Sigma$, alors $x^2 + y^2 > 1$ et $y > 0$
Ainsi, on a bien $\beta(x, y) = \Im(J(z)) > 0$ □

Conclusion :
Nous avons démontré que l'application de Joukowski est une application conforme sur tous les domaines ouverts du plan complexe n'incluant ni 0 ni -1 ni 1 . De plus, les propriétés mises en évidence dans cette section serviront à déterminer la distribution du vent autour d'un disque, connaissant le vent autour d'un segment. (voir 2.2.5)

2.1.4 Transformation de Schwarz-Christoffel

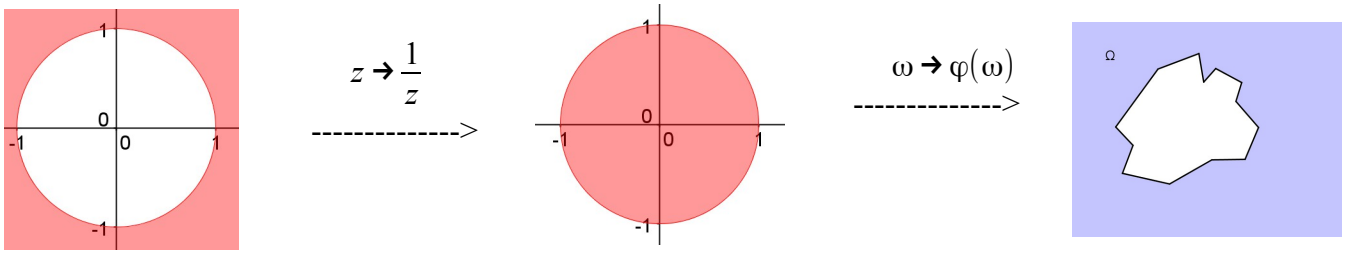
Étant donné la conclusion de la section 2.1.3, il serait désormais intéressant de mettre en évidence une application renvoyant l'extérieur du disque unité sur l'extérieur de notre polygone. Le théorème suivant nous permettra d'en assurer l'existence.

Théorème : Théorème de l'application conforme de Riemann
Soit U un ouvert simplement connexe non vide du plan complexe, distinct du plan tout entier. Alors, il existe une bijection holomorphe (application conforme) entre U et le disque unité (frontière exclue).

Notre domaine d'étude Ω vérifie bien toutes les conditions du théorème de Riemann. On peut donc affirmer qu'il existe une application conforme φ entre l'intérieur du disque unité et Ω .

De plus, par l'inversion $z \to \frac{1}{z}$ qui est aussi une application conforme, l'intérieur du disque unité

est renvoyé sur son extérieur (et inversement). Ainsi, l'application $z \to \varphi(\frac{1}{z})$ est l'application recherchée.



Le théorème de Riemann nous a donc permis de prouver l'existence de l'application φ mais ne nous permet pas de déterminer cette dernière.

Définition :
Soit P un polygone dont les sommets ordonnés ont pour affixes les $\{z_1, z_2, \ldots, z_n\}$ et dont les angles intérieurs correspondants sont les $\{\beta_1, \beta_2, \ldots, \beta_n\}$. On appelle ***application de Schwarz-Christoffel*** associée au polygone P une application de la forme suivante :

$$\forall z \in C,\ sc(z) = \omega_0 + K \int_0^z \prod_{k=1}^{n} \left(1 - \frac{\omega}{z_k}\right)^{1-\frac{\beta_k}{\pi}} . d\omega$$ où ω_0 et K sont des constantes complexes.

Propriété :
Une application de Schwarz-Christoffel définie comme ci-dessus est une application conforme qui envoie le disque unité sur l'extérieur du polygone P.

Cette transformation correspond donc parfaitement à l'application φ recherchée. Les constantes peuvent être choisies à l'aide de conditions supplémentaires du type $sc(z_0) = \omega_0$ ou encore $sc'(z_0) = \alpha_0$. Cette liberté dans le choix des constantes est intéressante car cela nous permet de conserver par exemple l'orientation d'une figure.

En pratique, nous utilisons un package permettant de générer cette application par la simple donnée des sommets du polygone P. Ce package est utilisable sous *Matlab* et a été créé par Tobin A. Driscoll de l'Université de Delaware (USA). Notons que ce package nous permet également d'obtenir la dérivée, ainsi que l'inverse de l'application.

2.2 Détermination du vent autour d'un obstacle

Nous étudions le cas où le vent s'écoule suivant un flux constant en l'absence d'obstacle. Nous cherchons à déterminer la distribution du vent sur le domaine Ω présenté précédemment. Pour cela, nous allons l'assimiler à une fonction u_Ω qui à tout point de Ω associe un élément de

R^2 . Le vent est supposé assez régulier pour que la fonction u_Ω soit de classe C^1 sur Ω .

2.2.1 Théorème de Helmholtz-Hodge

L'idée de départ est de vouloir écrire le vent comme une fonction qui dérive d'un potentiel scalaire. Nous verrons dans la partie 2.2.3 que c'est ce qui nous permettra de faire passer la solution pour le vent d'un domaine à un autre.

Une première approche est d'utiliser la décomposition suivante :

Théorème : Décomposition de Helmholtz-Hodge
Soit Ω un domaine compact et connexe du plan, de frontière régulière ou régulière par morceaux.
Soit $\vec{u}$ un champ de vecteurs de classe C^1 . $\vec{u}$: $\Omega \rightarrow R^2$

Alors : $\begin{cases} \exists \vec{A} : \Omega \rightarrow R^2 \\ \exists \psi : \Omega \rightarrow R \end{cases}$ tels que $\vec{u} = grad(\psi)^T + rot(\vec{A})$

Désormais, nous pouvons écrire le vent comme : $\vec{u_\Omega} = grad(\psi)^T + rot(\vec{A})$. En effet, toutes les hypothèses du théorème sont vérifiées, quitte à borner le domaine Ω . La connaissance de la température et du vent sont en effet inutiles pour le problème lorsqu'on est très éloigné de la horde.

En analysant la décomposition, on constate que le vent s'interprète alors comme la somme du gradient d'un potentiel scalaire et du rotationnel d'un champ de vecteurs. Sur le plan physique, on sait que le rotationnel induit des phénomènes de rotations.

Or, pour notre modèle, nous négligeons les phénomènes de tourbillonnement pour le vent. Dans notre cas on pourra alors écrire :

$$\vec{u_\Omega} = grad(\psi)^T$$

Dans la suite des travaux, nous nous attacherons à déterminer le potentiel ψ sur Ω pour ensuite en déduire le vent.

2.2.2 Équation du vent

Tout d'abord, utilisons l'hypothèse posée dans le modèle selon laquelle le vent ne pénètre pas dans la horde. Cela se traduit par le fait que le flux à travers la frontière de la horde est nul. Ainsi, on a : $\vec{u_\Omega}.\vec{n} = 0$ sur $\partial\Omega$.

On obtient donc la condition au bord suivante :

$$grad(\psi).\vec{n} = 0 \text{ sur } \partial\Omega$$

S'appuyant sur le fait que le vent est un phénomène de transport, on peut affirmer qu'il n'y a pas de source pour cette quantité. Alors pour tout domaine borné D , le flux à travers le bord de D est nul. Ainsi, on a :

$\int_{\partial D} \langle \vec{u_\Omega}.\vec{n} \rangle . d\sigma = 0$ où $\vec{n}$ désigne la normale unitaire en un point du bord.

Nous utilisons à présent le théorème suivant :

Théorème : ***Théorème de Gauss***
Soit D un domaine ouvert et borné du plan.
Soit $\vec{u}$ un champ de vecteurs de classe C^1 sur D .
Alors :

$$\int\int_D div\ (\vec{u}).dx.dy = \int_{\partial D} \langle \vec{u_\Omega}.\vec{n} \rangle . d\sigma$$

Ainsi, grâce à ce théorème, combiné à la remarque précédente, on obtient :

$\int\int_D div\ (\vec{u_\Omega}).dx.dy = 0$, $\forall D$ domaine ouvert borné inclus dans Ω .

Puisque l' intégrale doit s'annuler sur tout domaine $D \subset \Omega$, il faut forcément que :

$div\ (\vec{u_\Omega})(x,y) = 0$, $\forall (x,y) \in \Omega$

Enfin, on a vu en 2.2.1 que le vent s'écrit $\vec{u_\Omega} = grad(\psi)^T$. Alors, on obtient :

$div\ (grad(\psi))\ (x,y) = 0$, $\forall (x,y) \in \Omega$ et donc :

$\boxed{\Delta(\psi) = 0}$ sur Ω .

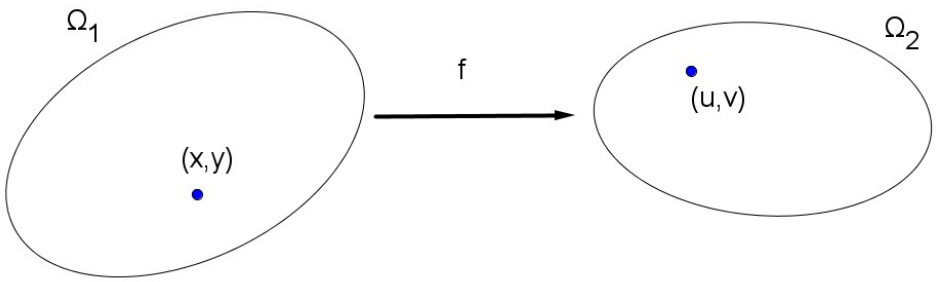
Pour déterminer le vent, il s'agira donc de chercher le potentiel ψ associé au vent en résolvant le problème de Poisson homogène suivant :

$$\boxed{\begin{cases} \Delta(\psi) = 0 \ \ sur\ \ \Omega \\ grad(\psi).\vec{n} = 0 \ \ sur\ \ \partial\Omega \end{cases}} \quad (*)$$

2.2.3 Passage de la solution d'un domaine à un autre

Dans cette section, on suppose que l'on connaît une solution du système (*) sur un certain domaine Ω_1 . Alors, on souhaite déterminer une solution du même système sur un domaine Ω_2 qui est l'image de Ω_1 par une application conforme f .

Nous utiliserons les notations présentes sur le schéma suivant :



Notons ψ_1 une solution du système (*) sur Ω_1 et considérons $\psi_2 = \psi_1 \circ f^{-1}$

Nous allons montrer que ψ_2 vérifie le système (*).

On a $\psi_1(x,y) = \psi_2 \circ f(x,y)$. Nous cherchons dans un premier temps à établir une relation entre $\Delta(\psi_1)$ et $\Delta(\psi_2)$.

Tout d'abord : $\nabla \psi_1(x,y) = \nabla \psi_2(f(x,y)).Jac(f)(x,y)$ où $Jac(f)$ désigne la matrice jacobienne de f .

On notera : $f(x,y) = (g(x,y), h(x,y))$

En développant la relation matricielle, on obtient :

$$\begin{cases} \frac{\partial \psi_1}{\partial x}(x,y) = \frac{\partial \psi_2}{\partial x}(f(x,y)).\frac{\partial g}{\partial x}(x,y) + \frac{\partial \psi_2}{\partial y}(f(x,y)).\frac{\partial h}{\partial x}(x,y) & (1) \\ \frac{\partial \psi_1}{\partial y}(x,y) = \frac{\partial \psi_2}{\partial x}(f(x,y)).\frac{\partial g}{\partial y}(x,y) + \frac{\partial \psi_2}{\partial y}(f(x,y)).\frac{\partial h}{\partial y}(x,y) & (2) \end{cases}$$

On dérive (1) par rapport à x :

$$\begin{aligned} \frac{\partial^2 \psi_1}{\partial x^2}(x,y) = & [\frac{\partial^2 \psi_2}{\partial x^2}(f(x,y)).\frac{\partial g}{\partial x}(x,y) + \frac{\partial^2 \psi_2}{\partial x \partial y}(f(x,y)).\frac{\partial h}{\partial x}(x,y)] \quad .\frac{\partial g}{\partial x}(x,y) + \\ & [\frac{\partial^2 \psi_2}{\partial x \partial y}(f(x,y)).\frac{\partial g}{\partial x}(x,y) + \frac{\partial^2 \psi_2}{\partial y^2}(f(x,y)).\frac{\partial h}{\partial x}(x,y)] \quad .\frac{\partial h}{\partial x}(x,y) + \\ & \frac{\partial \psi_1}{\partial x}(f(x,y)).\frac{\partial^2 g}{\partial x^2}(x,y) + \frac{\partial \psi_1}{\partial y}(f(x,y)).\frac{\partial^2 h}{\partial x^2}(x,y) \end{aligned}$$

$$\frac{\partial^2 \psi_1}{\partial x^2} = \frac{\partial^2 \psi_2}{\partial x^2}(f).(\frac{\partial g}{\partial x})^2 + 2.\frac{\partial g}{\partial x}.\frac{\partial h}{\partial x}.\frac{\partial^2 \psi_2}{\partial x \partial y}(f) + \frac{\partial \psi_2}{\partial x}(f).\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 \psi_2}{\partial y^2}(f).(\frac{\partial h}{\partial x})^2 + \frac{\partial \psi_2}{\partial y}(f).\frac{\partial^2 h}{\partial x^2}$$

De même, en dérivant (2) par rapport à y :

$$\frac{\partial^2 \psi_1}{\partial y^2} = \frac{\partial^2 \psi_2}{\partial x^2}(f).(\frac{\partial g}{\partial y})^2 + 2.\frac{\partial g}{\partial y}.\frac{\partial h}{\partial y}.\frac{\partial^2 \psi_2}{\partial x \partial y}(f) + \frac{\partial \psi_2}{\partial x}(f).\frac{\partial^2 g}{\partial y^2} + \frac{\partial^2 \psi_2}{\partial y^2}(f).(\frac{\partial h}{\partial y})^2 + \frac{\partial \psi_2}{\partial y}(f).\frac{\partial^2 h}{\partial y^2}$$

Puis en additionnant ces deux lignes, on obtient :

$$\Delta(\psi_1)=\frac{\partial \psi_2^2}{\partial x^2}(f).(g_x^2+g_y^2)+\frac{\partial \psi_2^2}{\partial y^2}(f).(h_x^2+h_y^2)+\Delta g.\frac{\partial \psi_2}{\partial x}(f)+\Delta h.\frac{\partial \psi_2}{\partial y}(f)$$

Puisque f est une application conforme, on peut utiliser les équations de Cauchy-Riemann énoncées en 2.1.2. Alors :

$g_x=h_y \qquad h_x=-g_y$ et donc $h_x^2+h_y^2=g_x^2+g_y^2$

De plus, g et h sont harmoniques. On obtient donc :

$$\Delta(\psi_1)=\Delta\psi_2(f).(g_x^2+g_y^2)$$

D'autre part, on remarque que, toujours grâce aux équations de Cauchy Riemann, la jacobienne de f est de la forme :

$$Jac(f)=\begin{pmatrix} g_x & g_y \\ -g_y & g_x \end{pmatrix}$$

Alors, on obtient la relation : $\Delta(\psi_1)=\Delta\psi_2(f).det(Jac(f))$

Enfin, notons que l'application f étant conforme, le déterminant de sa jacobienne ne s'annule jamais.

Ainsi, si $\Delta(\psi_1)=0$ sur Ω_1 , alors $\Delta(\psi_2)=0$ sur $\Omega_2=f(\Omega_1)$

Comme ψ_1 est une solution du système (*) sur Ω_1 , si $\vec{n}_1$ désigne la normale unitaire au bord $\partial\Omega_1$, on a :

$grad(\psi_1).\vec{n}_1=0$ *sur* $\partial\Omega_1$ et alors les vecteurs $grad(\psi_1)^T$ et $\vec{n}_1$ sont orthogonaux.

De plus, nous rappelons que l' application f est conforme donc elle conserve les angles. Ainsi, les vecteurs $grad(\psi_2)^T$ et $\vec{n}_2$ sont également orthogonaux. On a donc :

$$grad(\psi_2).\vec{n}_2=0 \quad sur \quad \partial\Omega_2$$

Les deux conditions du système (*) sont bien vérifiées. Alors, nous pouvons transformer une solution ψ_1 de (*) sur Ω_1 en une solution ψ_2 de (*) sur Ω_2 par la formule :

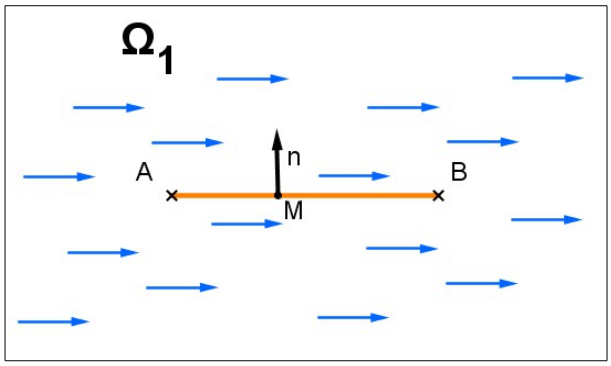
$$\psi_2=\psi_1\circ f^{-1}$$

2.2.4 Vent autour d'un segment

Nous cherchons dans un premier temps à trouver le vent dans une configuration très simple :

- ➔ Le vent, en l'absence d'obstacle, est constant et suit la direction (O,x) dans le sens positif.
- ➔ On considère un obstacle d'épaisseur nulle, représenté par le segment $[(-2,0)\ ;\ (2,0)]$.

Voici une représentation de la situation :



On remarque qu'en tout point M du segment où la normale existe, la normale est toujours égale au vecteur :

$\vec{n}=(0,1)$

L'obstacle ayant une épaisseur nulle, il ne génère aucune résistance au vent. Le champ de vecteurs est donc constant partout sur Ω_1 . Si l'on note U la vitesse constante du vent, on obtient l'expression du vent : $\forall(x,y)\in\Omega_1,\quad \vec{u}_1(x,y)=(U,0)$.

Vérifions que l'on peut trouver un potentiel scalaire générant $\vec{u}_1$ qui vérifie le système (*) sur Ω_1 . Étant donné l'expression de $\vec{u}_1$, il est naturel de s'intéresser au potentiel suivant :

$\forall(x,y)\in\Omega_1 \quad ,\psi_1(x,y)=U.x$

On a bien $\vec{u}_1=grad(\psi_1)^T$.

Les vecteurs $\vec{u}_1$ et $\vec{n}$ sont orthogonaux donc la première équation de (*) est bien vérifiée. De plus, le gradient de ψ_1 étant constant, on a bien un laplacien nul sur Ω_1 .

Ainsi, le potentiel scalaire ψ_1 est solution de (*) sur Ω_1 .

2.2.5 Vent autour du disque unité

Nous disposons désormais d'une solution pour le potentiel du vent sur le domaine Ω_1 . De plus, comme nous l'avons vu en 2.1.3, nous connaissons une application conforme qui envoie Ω_1 sur l'extérieur du disque unité (domaine noté Ω_2). Il s'agit de l'inverse de l'application de Joukowski. Nous allons alors considérer F_1 la version complexifiée du potentiel ψ_1 , définie par :

$\forall z\in\Omega_1,\quad F_1(z)=U.\Re(z)$ où $\Re(z)$ désigne la partie réelle de z .

D'après la section 2.2.3, nous pouvons déduire une solution F_2 pour le potentiel du vent sur le domaine Ω_2 par : $\forall\omega\in\Omega_2,\quad F_2(\omega)=F_1(J(\omega))$.

Alors, pour $\omega=x+iy$, on a $F_2(x+iy)=U.Re\left(x+iy+\frac{1}{x+iy}\right)=U.x\left(1+\frac{1}{x^2+y^2}\right)$

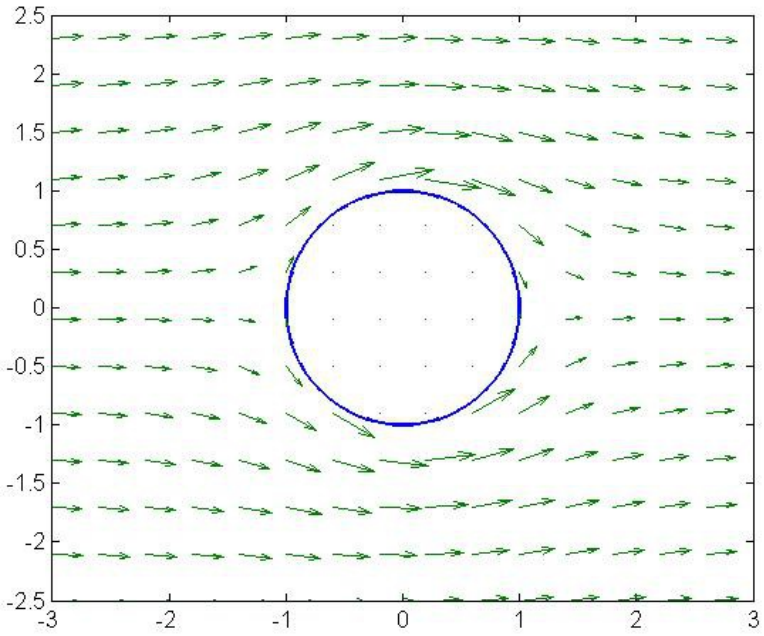
On associe à F_2 sa version réelle ψ_2 , définie par : $\psi_2(x,y)=F_2(x+iy)$

On détermine alors le vent $\vec{u}_2$ sur l'extérieur du disque unité par :

$\forall (x,y)\in\Omega_2,\quad \vec{u_2}(x,y)=grad(\psi_2)^T(x,y)$

Ainsi, $\forall (x,y)\in\Omega_2,\quad \vec{u_2}(x,y)=\left(U\left(1+\frac{y^2-x^2}{x^2+y^2}\right)\ ,\ \frac{-2xy}{x^2+y^2}.U\ \right)^T$

Voici la représentation du champ de vecteurs obtenu :

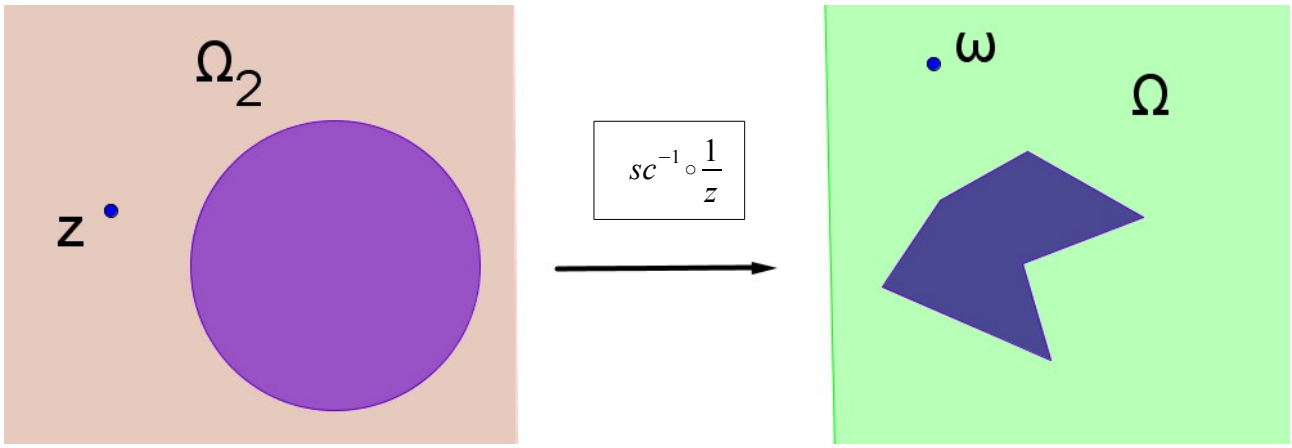


2.2.6 Vent autour d'un polygone

Connaissant désormais le vent et son potentiel autour du disque unité, nous cherchons à le déterminer sur notre domaine d'étude Ω. Pour cela, nous utilisons l'inverse de l'application de Schwarz-Christoffel présentée en 2.1.4. composée avec une inversion.

Ainsi, deux éléments quelconques $z\in\Omega_2$ et $\omega\in\Omega$ sont en bijection par la relation :

$\omega=sc^{-1}\left(\frac{1}{z}\right)$ où sc désigne l'application de Schwarz-Christoffel.



On note F le potentiel du vent sur le domaine Ω en version complexifiée.
Alors on a la relation :

$$\forall \omega \in \Omega, \quad F(\omega) = F_2(h^{-1}(\omega)) \quad \text{avec} \quad h(z) = sc^{-1} \circ \frac{1}{z}$$

et pour rappel : $F_2(z) = U.\Re(z + \frac{1}{z})$

On a : $h^{-1}(\omega) = \frac{1}{sc(\omega)}$ ainsi, $F(\omega) = U.\Re(sc(w) + \frac{1}{sc(w)}) = U.\Re(\Phi(\omega))$

Notons alors ψ la version réelle du potentiel F : $\psi(x, y) = F(x + iy)$.

On cherche à déterminer le vent $\vec{u_\Omega} = grad(\psi)^T$.

Alors : $\vec{u_\Omega}(x, y) = U.grad(\Re(A(x, y) + iB(x, y)))^T = U.grad(A)(x, y)$
en posant : $A(x, y) = \Re(\Phi(x + iy))$ et $B(x, y) = \Im(\Phi(x + iy))$

On peut alors noter le vent complexifié comme : $V(x + iy) = U.(A_x(x, y) + i A_y(x, y))$

La fonction A étant holomorphe, on peut utiliser les équations de Cauchy Riemann:

$A_x + iA_y = A_x - i B_x = \overline{A_x + iB_x} = \overline{\Phi'}$

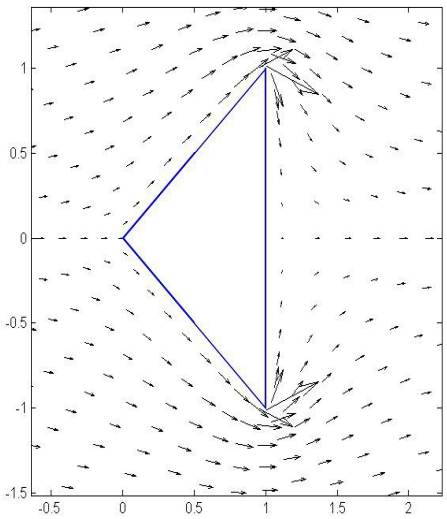
On a alors : $\boxed{V(x + iy) = U.(\overline{\Phi'(x + iy)})}$ avec $\Phi'(\omega) = sc'(\omega)(1 - \frac{1}{sc^2(\omega)})$ (#)

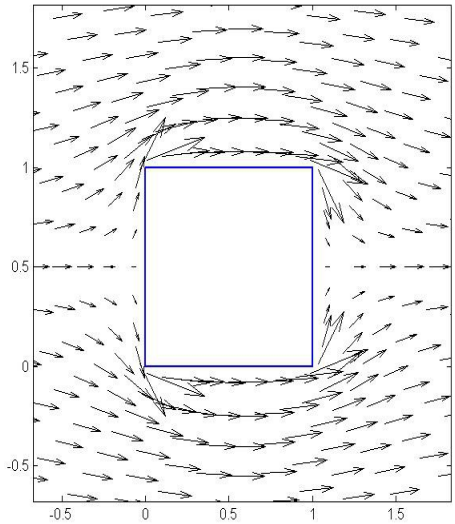
Finalement, on utilise la relation suivante : $\vec{u_\Omega}(x, y) = (\ \Re(V(x + iy))\ ,\ \Im(V(x + iy))\ )$

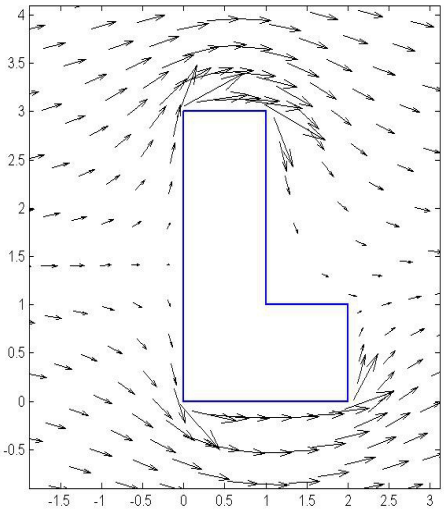
Remarque :

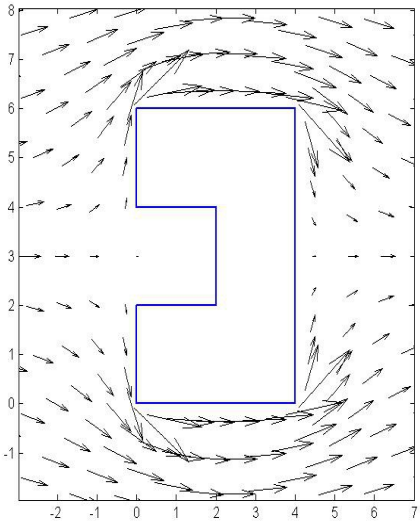
En pratique, lorsqu'on utilise le package de Tobin A. Driscoll, l' application de Schwarz-Christoffel qui est générée ne conserve pas forcément l'orientation du vent dans les zones éloignées de la horde. Au vu de la formule (#), et sachant que le vent dans les zones éloignées de la horde est de la forme (U , 0) avec $U > 0$, il faudra déterminer la rotation qui permet d'obtenir la condition suivante : $\|X\| = +\infty$ implique $f'(X) > 0$.

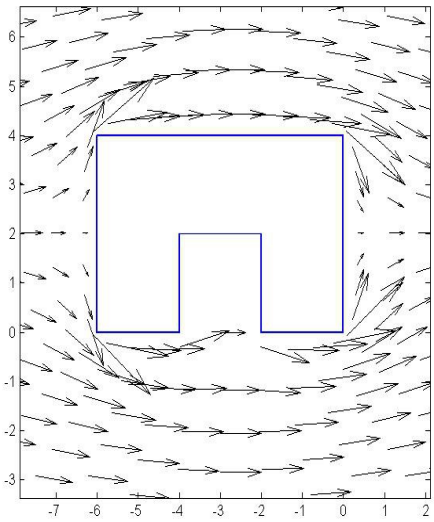
Voici quelques représentations du vent obtenu autour de différents types de polygones.

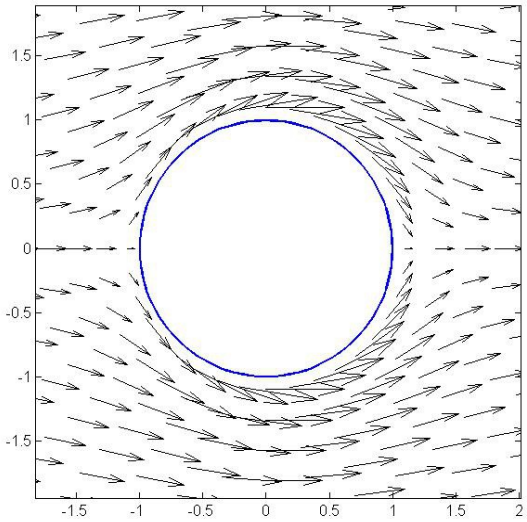

1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5
1
1.5
2


1.5
1
0.5
0
-0.5
-0.5
0
0.5
1
1.5


4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3


8
7
6
5
4
3
2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7


6
5
4
3
2
1
0
-1
-2
-3
-7
-6
-5
-4
-3
-2
-1
0
1
2


1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5
2

La dernière image correspond à un polygone ayant 100 sommets situés sur le cercle unité. Cette configuration nous permet de montrer la cohérence de la méthode en comparant le résultat obtenu à celui de la partie 2.2.5 .

Ces exemples permettent de mettre en évidence quelques phénomènes :

* Le vent est nul pour les points situés dans les zones concaves de Ω (voir figures 4 et 5).
* Lorsque le vent fait face à un mur qui est perpendiculaire à sa direction d'origine, la direction des flèches se modifie assez rapidement en amont de l'obstacle.
* Au niveau de certaines irrégularités de la frontière, le vent est beaucoup plus fort. Il s'agit des sommets du polygone pour lesquels le vent n'as pas beaucoup dévié en amont. En effet, pour le sommet de gauche du triangle, le vent arrive avec une intensité très faible car il a été dévié en amont vers les deux autres sommets. Par conséquent, l'intensité du vent au niveau de ces deux autres sommets devient très grande.

2.3 Détermination de la température

Le but de cette partie est de mettre en place une méthode permettant d'obtenir la température en tout point du domaine Ω . Pour cela, nous réutiliserons la théorie des applications conformes déjà utilisées dans le cas du vent.

2.3.1 Équation de convection-diffusion

Nous allons utiliser dans cette partie l'équation de convection-diffusion qui relie le vent et la température :

$$\frac{\partial \tau}{\partial t} = D.\Delta\tau - \nabla_{x,y}\tau.\vec{u} + R$$

avec les notations suivantes :

τ : Température $\vec{u}$: Vent t : variable temporelle x, y : variables spatiales
R : Source de chaleur D : Coefficient de diffusion thermique

D'après les hypothèses du modèle, il n'y a pas de source de chaleur dans le domaine Ω .
Ainsi, $R=0$.

De plus, à chaque étape de résolution, le système est à l'équilibre et donc ne dépend pas du temps. Ainsi, l' équation devient :

$$\nabla_{x,y}\tau.\vec{u} = D.\Delta\tau$$

Pour simplifier l'étude, nous normalisons l'équation de la façon suivante, en notant :

$\vec{u}' = \frac{\vec{u}}{U}$, $T = \frac{\tau - \tau_\infty}{\tau_H - \tau_\infty}$ où τ_H est la température au sein de la horde.

τ_∞ est la température aux points éloignés de la horde

De cette façon, la variable T est un nombre sans unité compris entre 0 et 1.
Il vaudra 1 sur la frontière de la horde et 0 pour les points éloignés de la horde.

On introduit également : $x' = \frac{x}{L}$ et $y' = \frac{y}{L}$ où L désigne le diamètre de la horde dans le cas ou elle aurait une forme circulaire (L dépend alors du nombre de manchots).

On introduit également la constante $Pe = \frac{U.L}{D}$ nommée nombre de Péclet.
On remarque que si l'on souhaite modifier l'intensité du vent, il suffit de modifier cette dernière constante.

L'équation devient alors $Pe . \nabla_{x,y} T . \vec{u} = \Delta T$ où toutes les quantités sont sans unité.
Étant données les hypothèses énoncées quant à la température constante au sein de la horde, on ajoute à cette équation la condition au bord de type Dirichlet suivante :

$$T(x, y) = 1 \quad , \quad \forall (x, y) \in \partial\Omega$$

Remarque :
Nous aurons besoin par la suite d'utiliser l'autre hypothèse émise pour la température disant qu'au loin de la horde, la température est également constante. Dans ce cas on aura la condition $T = 0$.

Le système qui caractérise la température pour notre problème est donc :

$$\begin{cases} Pe . \nabla_{x,y} T . \vec{u} = \Delta T \quad sur \quad \Omega \\ T(x, y) = 1 \quad , \quad \forall (x, y) \in \partial\Omega \\ \lim\limits_{x,y \to +\infty} T(x, y) = 0 \end{cases} \qquad (**)$$

Méthode de résolution du problème pour la température :
Étant donné que nous disposons de formules explicites pour le vent sur le domaine Ω_2 correspondant à l'extérieur du cercle, nous allons chercher à résoudre la première équation du système (**) par une méthode de différences finies. Ensuite, nous déterminerons la température sur le domaine final Ω à l'aide de la transformation conforme qui envoie Ω_2 sur Ω .

2.3.2 Passage de la solution d'un domaine à un autre

On suppose connue une solution de (**) sur un domaine Ω_1 notée T_1 . On cherche à démontrer que si f est une application conforme qui envoie Ω_1 sur un autre domaine Ω_2 , alors $T_2 = T_1 \circ f^{-1}$ est solution de (**) sur Ω_2 .

Comme nous l'avons déjà démontré pour le vent dans la section 2.2.3, on réutilise la formule qui relie les Laplaciens et les gradients des deux fonctions T_1 et T_2 :

$$\nabla(T_1) = \nabla T_2(f) . Jac(f) \qquad \text{et} \qquad \Delta(T_1) = \Delta T_2(f) . det(Jac(f))$$

Alors : $\Delta T_2(f) = \frac{Pe}{det(Jac(f))} . \nabla T_1 . \vec{u}_1$ car T_1 solution de (**)

$$\Delta T_2(f) = \frac{Pe}{det(Jac(f))} . \nabla T_2 . Jac(f) . \vec{u}_1$$

On rappelle que $\vec{u_1}=\nabla\psi_1$ et que $\vec{u_1}=\nabla\psi_1$ avec $\psi_2=\psi_1\circ f^{-1}$

Alors, on a également la relation : $\vec{u_1}={}^t\nabla(\psi_1)=(\nabla\psi_2(f).Jac(f))^T=(Jac(f)^T).\vec{u_2}$

Ainsi : $\Delta T_2(f)=\frac{Pe}{det(Jac(f))}.\nabla T_2.Jac(f).(Jac(f))^T\vec{u_2}$

Or, par les équations de Cauchy-Riemann, on a $Jac(f).(Jac(f))^T=det(Jac(f)).I$

En effet, $Jac(f)$ est de la forme $\begin{pmatrix} g_x & g_y \\ -g_y & g_x \end{pmatrix}$ et donc on a :

$$Jac(f).(Jac(f))^T=\begin{pmatrix} g_x{}^2+g_y^2 & 0 \\ 0 & g_x{}^2+g_y^2 \end{pmatrix}=det(Jac(f)).I$$

On obtient donc finalement : $\Delta T_2(f)=Pe.\nabla T_2.\vec{u_2}$
Ainsi, $\boxed{T_2=T_1\circ f^{-1}}$ est solution de la première équation du système (**) sur Ω_2 .

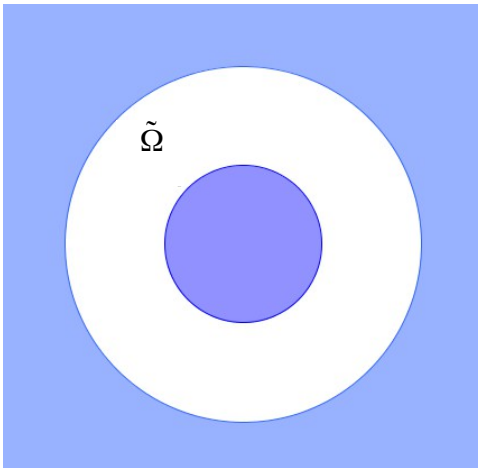
Remarque :

1. On remarque dans cette démonstration, comme dans la démonstration du 2.2.3, que ce qui nous permet d'obtenir une solution sur le nouveau domaine est l'utilisation des équations de Cauchy-Riemann. On constate alors tout l'intérêt des applications holomorphes.

2. Afin de vérifier que le système (**) tout entier est bien vérifié pour T_2 , il faudra s'assurer que les deux autres conditions sont toujours valables. Autrement dit, il faut avoir $f(\partial\Omega_1)=\partial(\Omega_2)$ et $\forall X,\ tel\ que\ \|X\|=+\infty,\ \|f(X)\|=+\infty$. Or nous verrons que ces conditions sont bien vérifiées pour l'application conforme que l'on va utiliser.

2.3.3 Température autour du disque unité

Pour résoudre l' équation aux dérivées partielles de (**) autour du disque unité, nous nous plaçons sur le domaine $\tilde{\Omega}$ décrit sur la figure suivante :



$\tilde{\Omega}$ est le domaine situé entre le cercle unité et un deuxième cercle de même centre et de rayon $r_{max}>1$. Il est raisonnable de se limiter à ce domaine car la température dans les zones éloignées de la horde est constante et connue.

Sur ce domaine, il est naturel d'utiliser les coordonnées polaires, en notant :

$x=r.cos(\theta)$ et $y=r.sin(\theta)$
où $(r,\theta)\in[1,r_{max}]\times[0,2\pi]$

Pour pouvoir utiliser l'équation (**), nous avons besoin de la traduire en coordonnées polaires.

Tout d'abord, nous connaissons le vent $\vec{u}_2$ d'après la partie 2.2.5 :

$$\forall (x,y)\in\tilde{\Omega},\quad \vec{u}_2(x,y)=(1+\frac{y^2-x^2}{(x^2+y^2)^2}\ ,\ \frac{-2xy}{(x^2+y^2)^2}\ )^T$$

Ce qui donne pour la version polaire $\vec{\mu}_2$:

$$\vec{\mu}_2(r,\theta)=(\ \frac{r^2+\sin^2\theta-\cos^2\theta}{r^2}\ ,\frac{-2\cos\theta\sin\theta}{r^2}\ )^T \qquad (2.0)$$

On note $\tilde{T}$ et τ la température sur $\tilde{\Omega}$ respectivement en coordonnées cartésiennes et polaires.

Alors : $\tau(r,\theta)=\tilde{T}(r.cos(\theta),r.sin(\theta))$ et en dérivant, on obtient :

(1) $\frac{\partial\tau}{\partial r}(r,\theta)=\cos(\theta)\frac{\partial\tilde{T}}{\partial x}+\sin(\theta)\frac{\partial\tilde{T}}{\partial y}$ et (2) $\frac{\partial\tau}{\partial\theta}(r,\theta)=-r.\sin(\theta)\frac{\partial\tilde{T}}{\partial x}+r.\cos(\theta)\frac{\partial\tilde{T}}{\partial y}$

Cherchons tout d'abord à exprimer les dérivées partielles de $\tilde{T}$ en fonction de celle de τ.

Par $(1)\times r.sin(\theta)+(2)\times\cos(\theta)$, on obtient :

$$\frac{\partial\tilde{T}}{\partial y}=\frac{\cos(\theta)\frac{\partial\tau}{\partial\theta}}{r}+\sin(\theta)\frac{\partial\tau}{\partial r} \qquad (2.1)$$

Puis par $(1)\times r.cos(\theta)-(2)\times\sin(\theta)$: $\frac{\partial\tilde{T}}{\partial x}=-\frac{\sin(\theta)\frac{\partial\tau}{\partial\theta}}{r}+\cos(\theta)\frac{\partial\tau}{\partial r}$ (2.2)

On a alors $(\nabla T)^T=\begin{pmatrix}\frac{\partial T}{\partial x}(r\cos(\theta),r\sin(\theta))\\ \frac{\partial T}{\partial y}(r\cos(\theta),r\sin(\theta))\end{pmatrix}=\begin{pmatrix}\frac{-\sin(\theta)\frac{\partial\tau}{\partial\theta}}{r}+\cos(\theta)\frac{\partial\tau}{\partial r}\\ \frac{\cos(\theta)\frac{\partial\tau}{\partial\theta}}{r}+\sin(\theta)\frac{\partial\tau}{\partial r}\end{pmatrix}$

On peut alors transposer dans la formule précédente, on écrira les calculs avec l'opérateur de dérivée pour alléger la notation.

$$\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{-\sin(\theta)\frac{\partial \tau}{\partial \theta}}{r} + \cos(\theta)\frac{\partial \tau}{\partial r} \\ \frac{\cos(\theta)\frac{\partial \tau}{\partial \theta}}{r} + \sin(\theta)\frac{\partial \tau}{\partial r} \end{pmatrix} = \begin{pmatrix} \cos\theta & \frac{-\sin\theta}{r} \\ \sin\theta & \frac{\cos\theta}{r} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial \theta} \end{pmatrix}$$

Par produit :

$$\begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} = \frac{1}{r} \begin{pmatrix} \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} \end{pmatrix} \begin{pmatrix} r\cos\theta & r\sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & \frac{-\sin\theta}{r} \\ \sin\theta & \frac{\cos\theta}{r} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial \theta} \end{pmatrix}$$

$$= \frac{1}{r} \cdot \begin{pmatrix} \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} \end{pmatrix} \cdot \begin{pmatrix} r & 0 \\ 0 & \frac{1}{r} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial \theta} \end{pmatrix}$$

$$= \frac{1}{r} \begin{pmatrix} \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} \end{pmatrix} \cdot \begin{pmatrix} r\frac{\partial}{\partial r} \\ \frac{1}{r}\frac{\partial}{\partial \theta} \end{pmatrix}$$

Et on a donc :

$$D = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial \theta^2} = \frac{1}{r}\left(\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right)` + \frac{\partial}{\partial \theta}\left(\frac{1}{r}\frac{\partial}{\partial \theta}\right)\right) = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2} + \frac{\partial^2}{\partial \theta^2}$$

On applique alors l'opérateur à τ et on obtient le Laplacien de la température en coordonnées polaires :

$$\Delta \tilde{T}(r.cos(\theta), r.sin(\theta)) = \frac{\partial \tau}{\partial r^2}(r,\theta) + \frac{1}{r}\frac{\partial \tau}{\partial r}(r,\theta) + \frac{1}{r^2}\frac{\partial \tau}{\partial \theta^2}(r,\theta) \quad (2.3)$$

Ensuite, on injecte les relations (2.0), (2.1), (2.2) et (2.3) dans (**),
On obtient alors après simplification:

$$\boxed{\tau_\theta \cdot \left(-\sin\theta \cdot \left(\frac{1}{r} + \frac{1}{r^3}\right)\right) + \tau_r \cdot \left(\cos\theta \cdot \left(1 - \frac{1}{r^2}\right) - \frac{1}{Pe \cdot r}\right) - \frac{1}{Pe}\cdot\tau_{rr} - \frac{1}{Pe \cdot r^2}\cdot\tau_{\theta\theta} = 0}$$

L' équation est couplée avec les conditions aux limites suivantes :

$$\begin{cases} \tau(1,\theta) = 1, \ \tau(r_{max},\theta) = 0 & \forall \ \theta \in [0, 2\pi] \\ \tau(r, 2\pi) = \tau(r, 0) & \forall r \in [1, r_{max}] \end{cases}$$

Cette équation sera résolue numériquement par une méthode de différences finies. (voir section 3.2)

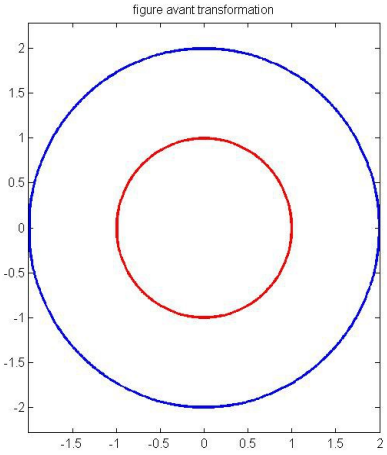
2.3.4 Température autour d'un polygone

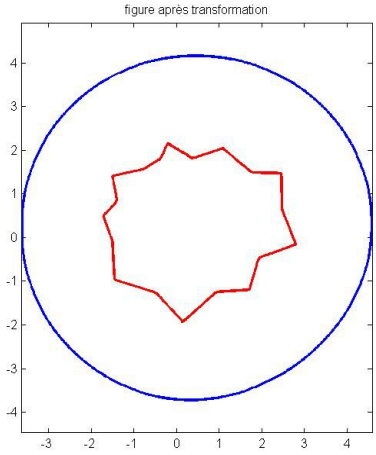
On note T la température sur le domaine final Ω , et $\tilde{T}$ la température autour du disque unité (domaine Ω_2). D'après 2.3.2 et 2.1.4,

$T_2 = T_1 \circ f^{-1}$ avec $\forall z \in \Omega_2, \quad f(z) = sc^{-1}(\frac{1}{z})$

Ainsi, on a : $\forall \omega \in \Omega$, $T(\omega) = \tilde{T}(\frac{1}{sc(\omega)}) = \tilde{T}(z)$ si $f(z) = \omega$.

Étant donné la forme du domaine sur lequel est connue $\tilde{T}$, nous nous intéressons à l'image de ce dernier afin de vérifier qu'il est suffisant pour la résolution du problème.

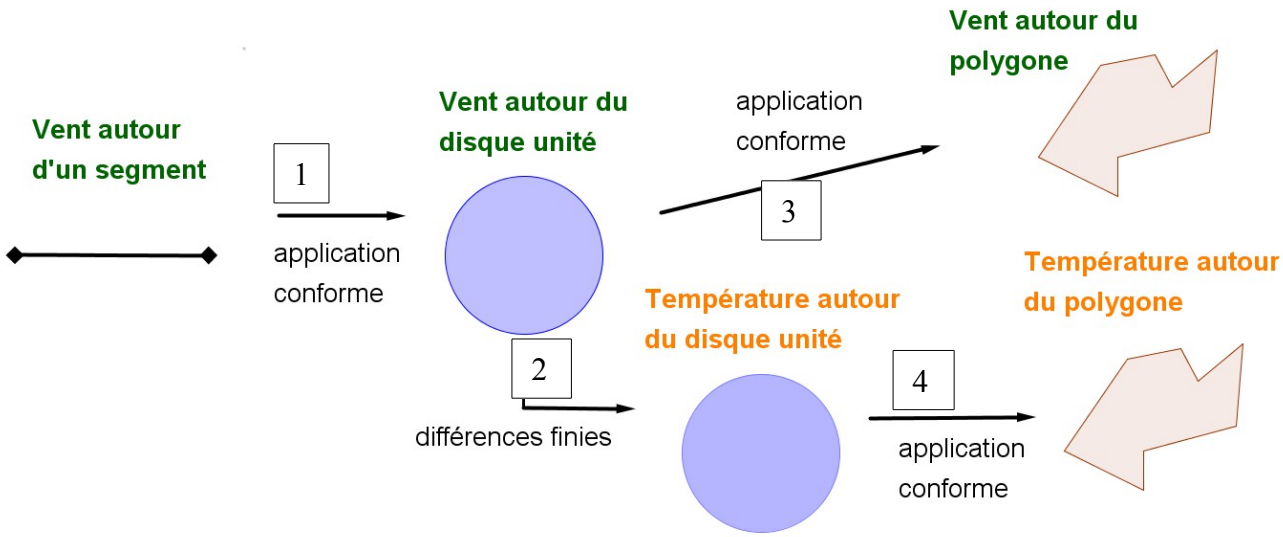




Ci-contre, nous pouvons voir l'image du domaine de résolution par l'application f pour un polygone quelconque. On peut remarquer que la transformation a conservé la forme du cercle extérieur mais aussi les proportions de départ. Le domaine choisi en forme d'anneau convient donc parfaitement à la résolution autour du disque.

3. RÉSOLUTION NUMÉRIQUE

Après avoir démontré dans les parties précédentes les méthodes utilisées, il est temps de passer à la résolution numérique du problème de détermination de la température et du vent sur le domaine extérieur de la horde de manchots. Afin d'illustrer les méthodes utilisées, nous proposons le diagramme suivant :



Parmi les quatre repères indiquant des problèmes à résoudre, trois correspondent à des situations déjà traitées dans les sections précédentes :

→ repère 1 : Nous avons obtenu des formules explicites pour le vent autour du disque en 2.2.5.

→ repère 3 et 4 : Le vent autour du polygone est obtenu à l'aide de la composition de deux applications conformes (fonction inverse et réciproque de l'application de Schwarz-Christoffel). On utilise alors le package de Tobin A. Driscoll. Voir paragraphes 2.2.6 et 2.3.4

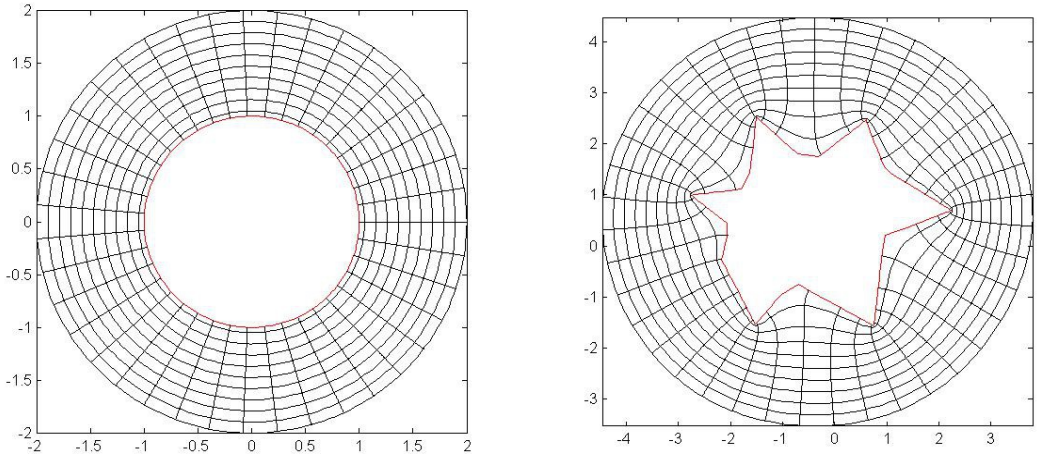
Il nous reste désormais à résoudre le problème d'obtention de la température autour du disque unité. Pour cela nous résolvons numériquement l'équation obtenue en 2.3.3 par une méthode de différences finies.

Remarque :

En analysant le diagramme ci-dessus, il est clair que nous aurions pu nous passer de la connaissance du vent autour du polygone pour déterminer la température autour du polygone. Seulement, il nous semble indispensable de le déterminer également afin de bien comprendre les phénomènes que l'on pourrait observer.

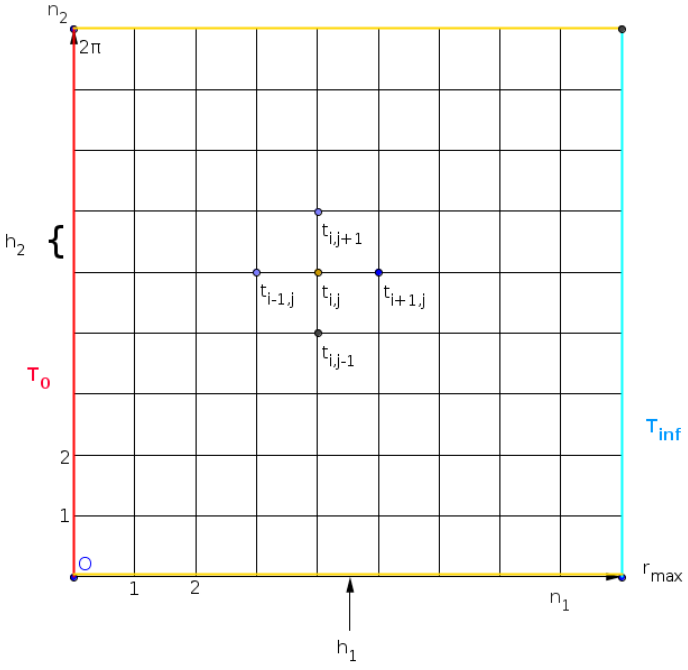
3.1 Maillage du domaine de résolution

Nous avions vu dans la partie 2.3.3 que le domaine de résolution $\tilde{\Omega}$ a la forme d'un anneau compris entre deux cercles centrés en l'origine et de rayons respectifs 1 et r_{max} . Le maillage qui s'impose donc naturellement est un maillage polaire. Toutefois, il est important de vérifier qu'après transformation par application conforme, les points où l'on connaîtra la solution sur le domaine final soient bien répartis. Examinons alors la figure suivante représentant le maillage autour du disque et sa transformation autour d' un polygone.



Nous remarquons que les nœuds du maillage d'arrivée sont répartis de manière similaire aux nœuds du maillage de départ, sauf pour les points situés dans les concavités du domaine Ω . En effet, à ces endroits les nœuds sont clairement plus éloignés de la frontière qu'ils ne l'étaient avant transformation. Pour remédier à cela, nous prendrons soin de choisir un pas radial très fin afin d'assurer que l'on atteint même les zones proches de la frontière du domaine Ω .

Sur la figure ci-dessous, nous présentons les notations utilisées dans la suite. Notons que nous avons choisi de représenter le maillage sous forme rectangulaire par soucis de clarté mais cela ne modifie pas le raisonnement pour autant.



Sur ce schéma, $t_{i,j}$ désigne l'approximation de la température au point (r_i, θ_j) vérifiant :

$$r_i = h_1 . i \quad \text{avec} \quad h_1 = \frac{r_{max} - 1}{n_1 + 1}$$

$$\theta_j = h_2 . j \quad \text{avec} \quad \theta_j = \frac{2\pi}{n_2}$$

Notons que T_0 désigne la température au sein de la horde et T_{inf} la température au large de la horde. Nous rappelons que $T_0 = 1$ et $T_{inf} = 0$.

On a alors les conditions au bord de type Dirichlet suivantes :

→ $t_{0,j}=T_{inf} \qquad \forall j\in\{0,1,.....,n_2\}$ (ligne rouge)

→ $t_{n1+1\ ,j}=T_0 \quad \forall j\in\{0,1,.....,n_2\}$ (ligne bleue)

Nous avons également la condition périodique suivante :

$t_{i,0}=t_{i,n_2} \qquad \forall i\in\{0,1,.....,n_1+1\}$ (lignes jaunes)

3.2 Schéma numérique utilisé pour la température

On rappelle l'équation de la température trouvée auparavant :

$$\tau_\theta\cdot(-\sin\theta\cdot(\frac{1}{r}+\frac{1}{r^3}))+\tau_r\cdot(\cos\theta\cdot(1-\frac{1}{r^2})-\frac{1}{Pe.r})-\frac{1}{Pe}\cdot\tau_{rr}-\frac{1}{Pe.r^2}\cdot\tau_{\theta\theta}=0 \qquad (\#\#)$$

On pose alors :

$$a_{i,j}=\cos\theta_j\cdot(1-\frac{1}{r_i^2})-\frac{1}{Pe.r_i} \quad , \quad b_{ij}=-\frac{\sin\theta_j}{r_i}\cdot(1+\frac{1}{r_i^2}) \quad , \quad c=-\frac{1}{Pe} \quad \text{et} \quad d_i=-\frac{1}{Pe.r_i^2}$$

Le schéma d'approximation par différences finies utilisé ici sera le schéma centré suivant :

$$a_{ij}\cdot\frac{u_{i+1,j}-u_{i-1,j}}{2h_1}+b_{ij}\cdot\frac{u_{i,j+1}-u_{i,j-1}}{2h_2}+c\cdot\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h_1^2}+d_i\cdot\frac{u_{i,j-1}-2u_{i,j}+u_{i,j+1}}{h_2^2}=0$$

$\forall 1\le i\le n_1, \qquad \forall 1\le j\le n_2$ en affectant à la notation u_{i,n_2+1} le terme $u_{i,1}$

Ce schéma numérique peut encore s'écrire sous la forme :

$$\eta_{i,j}\cdot u_{i+1,j}+\alpha_{i,j}\cdot u_{i-1,j}+\gamma_{i,j}\cdot u_{i,j}+\beta_{i,j}\cdot u_{i,j-1}+\mu_{i,j}\cdot u_{i,j+1}=0$$

en introduisant les notations suivantes :

$$\gamma_{i,j}=-\frac{2c}{h_1^2}-\frac{2d_i}{h_2^2} \quad , \qquad \beta_{i,j}=(-\frac{b_{ij}}{2h_2}+\frac{d_i}{h_2^2}) \quad , \qquad \alpha_{i,j}=(-\frac{a_{ij}}{2h_1}+\frac{c}{h_1^2}) \quad ,$$

$$\mu_{i,j}=(\frac{b_{ij}}{2h_2}+\frac{d_i}{h_2^2}) \quad , \qquad \eta_{i,j}=(\frac{a_{ij}}{2h_1}+\frac{c}{h_1^2})$$

Il s'agira alors de résoudre le système matriciel $A.U=B$ avec $A\in R^{n_1.n_2\times n_1.n_2}$ et $U, B\in R^{n_1.n_2}$ définis par : (Pour $n_1=3$ et $n_2=3$)

$$A=\begin{pmatrix} \gamma_{11} & \mu_{11} & \beta_{11} & \eta_{11} & 0 & 0 & 0 & 0 & 0 \\ \beta_{12} & \gamma_{12} & \mu_{12} & 0 & \eta_{12} & 0 & 0 & 0 & 0 \\ \mu_{13} & \beta_{13} & \gamma_{13} & 0 & 0 & \eta_{13} & 0 & 0 & 0 \\ \alpha_{21} & 0 & 0 & \gamma_{21} & \mu_{21} & \beta_{21} & \eta_{21} & 0 & 0 \\ 0 & \alpha_{22} & 0 & \beta_{22} & \gamma_{22} & \mu_{22} & 0 & \eta_{22} & 0 \\ 0 & 0 & \alpha_{23} & \mu_{23} & \beta_{23} & \gamma_{23} & 0 & 0 & \eta_{23} \\ 0 & 0 & 0 & \alpha_{31} & 0 & 0 & \gamma_{31} & \mu_{31} & \beta_{31} \\ 0 & 0 & 0 & 0 & \alpha_{32} & 0 & \beta_{32} & \gamma_{32} & \mu_{32} \\ 0 & 0 & 0 & 0 & 0 & \alpha_{33} & \mu_{33} & \beta_{33} & \gamma_{33} \end{pmatrix}, \quad B=\begin{pmatrix} -\alpha_{11} \\ -\alpha_{12} \\ -\alpha_{13} \\ 0 \\ 0 \\ 0 \\ -\eta_{31} \\ -\eta_{32} \\ -\eta_{33} \end{pmatrix} \quad \text{et} \quad U=\begin{pmatrix} u_{1,1} \\ \dots \\ u_{1,n_2} \\ u_{2,1} \\ \dots \\ u_{2,n_2} \\ \dots \\ \dots \\ u_{n_1,n_2} \end{pmatrix}$$

3.3 Consistance du schéma numérique

Nous allons maintenant nous intéresser à la consistance de notre schéma numérique. Pour cela, nous calculons l'erreur de troncature.

On utilise les développements limités suivants :

1) $$u(r_i \pm h_1, \theta_j) = u(r_i, \theta_j) \pm h_1 \cdot u_r(r_i, \theta_j) + \frac{h_1^2}{2} \cdot u_{rr}(r_i, \theta_j) \pm \frac{h_1^3}{6} \cdot u_{rrr}(r_i, \theta_j) + O(h_1^4)$$

2) $$u(r_i, \theta_j \pm h_2) = u(r_i, \theta_j) \pm h_2 \cdot u_\theta(r_i, \theta_j) + \frac{h_2^2}{2} \cdot u_{\theta\theta}(r_i, \theta_j) \pm \frac{h_2^3}{6} \cdot u_{\theta\theta\theta}(r_i, \theta_j) + O(h_2^4)$$

On a alors : $$\text{✳}_1 : \quad a_{ij} \cdot \frac{u_{i+1}^j - u_{i-1}^j}{2h_1} = a_{ij} \cdot \left[u_r(r_i, \theta_j) + \frac{h_1^2}{3} u_{rrr}(r_i, \theta_j) + O(h_1^3) \right]$$

puis $$\text{✳}_2 : \quad b_{ij} \cdot \frac{u_i^{j+1} - u_i^{j-1}}{2h_2} = b_{ij} \cdot \left[u_\theta(r_i, \theta_j) + \frac{h_2^2}{3} u_{\theta\theta\theta}(r_i, \theta_j) + O(h_2^3) \right]$$

et $$\text{✳}_3 : \quad c \cdot \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h_1^2} = c \cdot \left[u_{rr}(r_i, \theta_j) + \frac{h_1}{3} u_{rrr}(r_i, \theta_j) + O(h_1^2) \right]$$

et finalement $$\text{✳}_4 : \quad d_i \cdot \frac{u_i^{j-1} - 2u_i^j + u_i^{j+1}}{h_2^2} = d_i \cdot \left[u_{\theta\theta}(r_i, \theta_j) + \frac{h_2}{3} u_{\theta\theta\theta}(r_i, \theta_j) + O(h_2^2) \right]$$

Alors on obtient par addition et en utilisant l'équation (##) :

$$\begin{aligned} \text{✳}_1 + \text{✳}_2 + \text{✳}_3 + \text{✳}_4 = 0 + {} & a_{ij} \cdot \left[\frac{h_1^2}{3} u_{rrr}(r_i, \theta_j) + O(h_1^3) \right] + b_{ij} \cdot \left[\frac{h_2^2}{3} u_{\theta\theta\theta}(r_i, \theta_j) + O(h_2^3) \right] \\ & + c \cdot \left[\frac{h_1}{3} u_{rrr}(r_i, \theta_j) + O(h_1^2) \right] + d_i \cdot \left[\frac{h_2}{3} u_{\theta\theta\theta}(r_i, \theta_j) + O(h_2^2) \right] \\ = {} & \epsilon_{i,j} \end{aligned}$$

Alors :

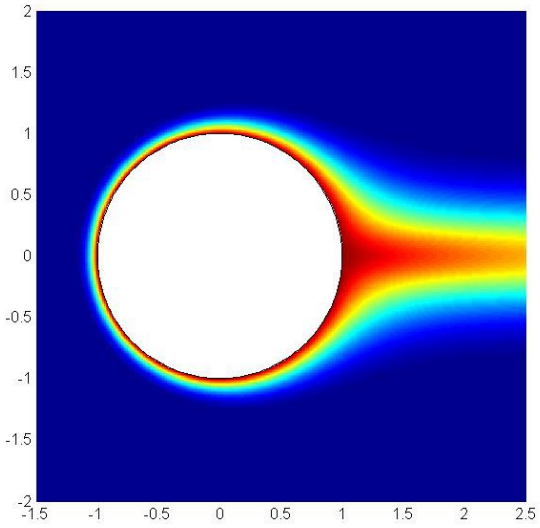
$$\|\epsilon_{i,j}\|_\infty \leqslant \left(\frac{h_1^2}{3}\cdot a_{ij} + \frac{h_1}{3}\cdot c\right)\cdot \left\|u_{rrr}(r,\cdot)\right\|_{L^\infty(\Re)} + \left(\frac{h_2^2}{3}\cdot b_{ij} + \frac{h_2}{3}\cdot d_i\right)\cdot \left\|u_{\theta\theta\theta}(\cdot,\theta)\right\|_{L^\infty(\Re)} + O(h_1^2) + O(h_2^2)$$

Ainsi, $\|\epsilon_{i,j}\|_\infty \to 0 \quad quand \quad h_1 \to 0 \quad et \quad h_2 \to 0.$

On peut alors affirmer que l'on a un ordre de consistance de 2 en rayon, et de 2 en angle.

3.4 Résultats

Nous avons implémenté cette méthode sous Matlab dans la fonction tempdisk. Voici la représentation graphique obtenue pour la solution numérique :



Résultat obtenu pour :
$r_{max} = 3, \quad n_1 = 150, \quad n_2 = 300, \quad Pe = 100$

Nous rappelons que le vent vient de la gauche de la figure, il est donc logique que les zones les plus chaudes soient situées sur la partie droite.

En récupérant les résultats de la fonction tempdisk avec différents jeux de paramètres, nous avons vérifié que les températures étaient toujours comprises entre 0 et 1. Ainsi, le schéma numérique utilisé semble générer un principe du maximum discret. Ce résultat qui était prévisible sera réutilisé dans la définition d'un critère d'évaluation de perte de chaleur pour les manchots.

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.

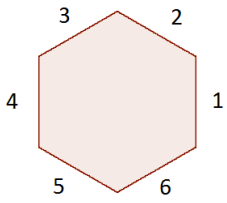
Enfin, pour les manchots situés sur la frontière, nous avons besoin de connaître les positions libres autour de chacun d'entre-eux, afin de définir les positions qui seraient susceptibles d'accueillir un nouveau manchot.

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)



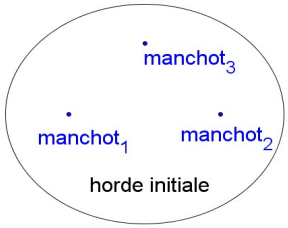
Exemple : Un manchot ***M*** ne possédant que deux voisins aux positions E et NE aura comme vecteur correspondant ***M.libre*** [3, 4, 5, 6]

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 : $ind = 3$, $x = 0$, $y = 1$, $libre = [1,2,3,4]$

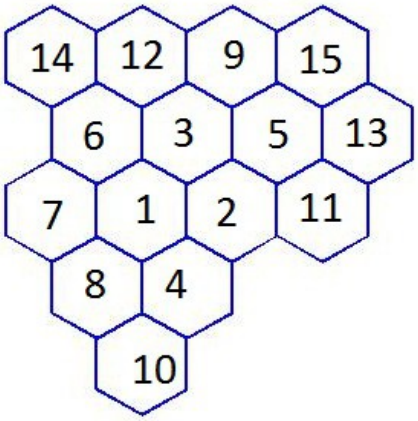


b. Présentation de la famille 'groupe' et du vecteur 'indfront'

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

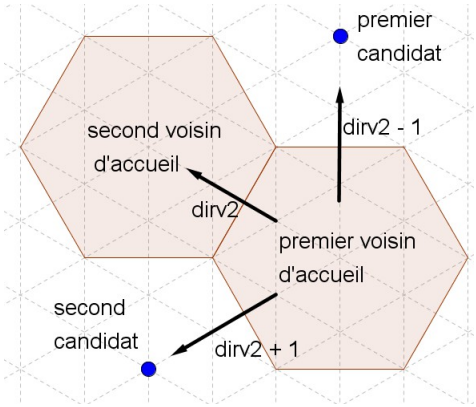
Cas 2 : Lors de la génération initiale de la horde

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 :



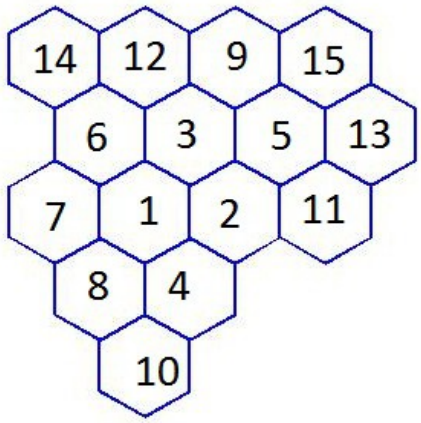
Parmi ces deux possibilités, seule une est à retenir car il y en a forcément une des deux qui est déjà occupée par un autre manchot. Il s'agit alors de tester laquelle des deux est libre afin de connaître la position d'accueil.

------> 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 :

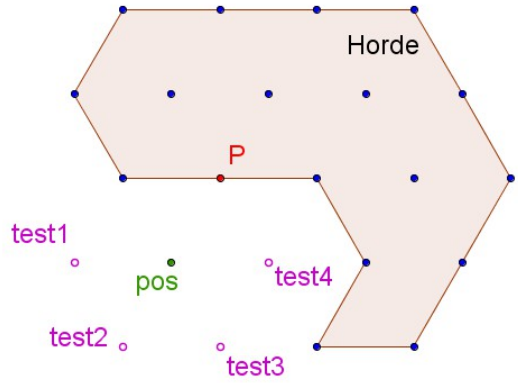
[4 2 11 13 15 9 12 14 6 7 8 10] devient [4 2 11 13 15 9 12 14 16 7 8 10]

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.

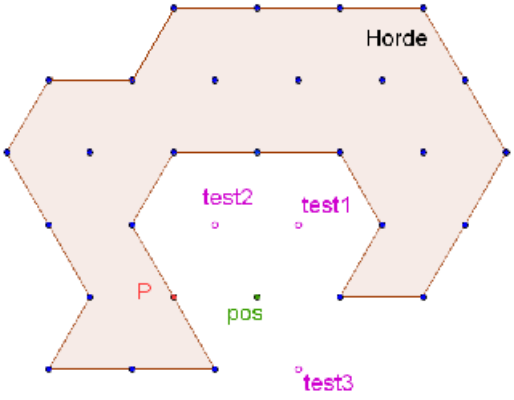
b. Détection d'un trou dans le maillage

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

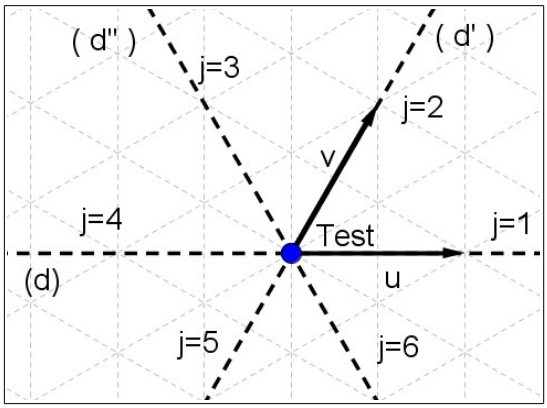
Comme nous l'avons dit précédemment, la construction de la fonction trou nécessite de savoir résoudre le problème suivant :

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.

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 :

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

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.

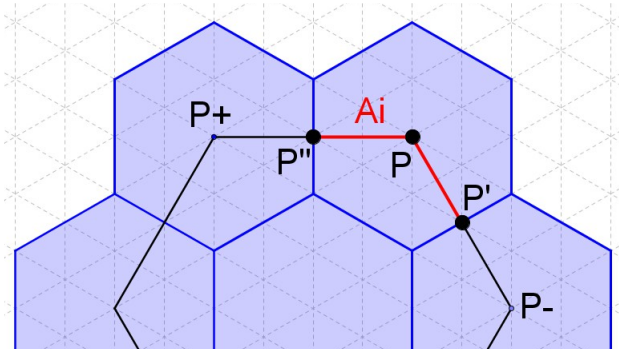
Ces contraintes ont déjà été gérées dans la partie 4.1.3.
De la même façon que lors du choix du manchot qui part, si nous rencontrons une position qui n'est pas acceptable, nous choisissons un autre manchot d'accueil P_{chaud} qui est celui qui a le plus chaud, hormis le manchot déjà analysé.

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

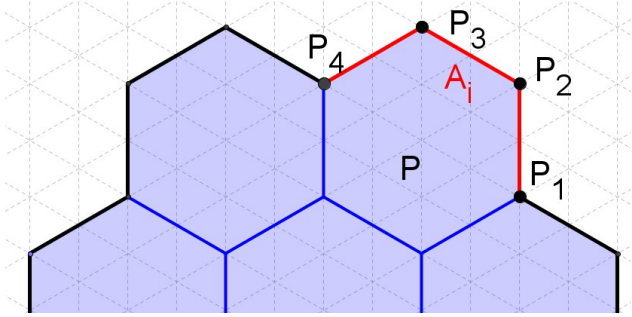
La nature de cette région dépend du modèle choisi pour tracer le polygone qui matérialise la horde.



------> 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 :

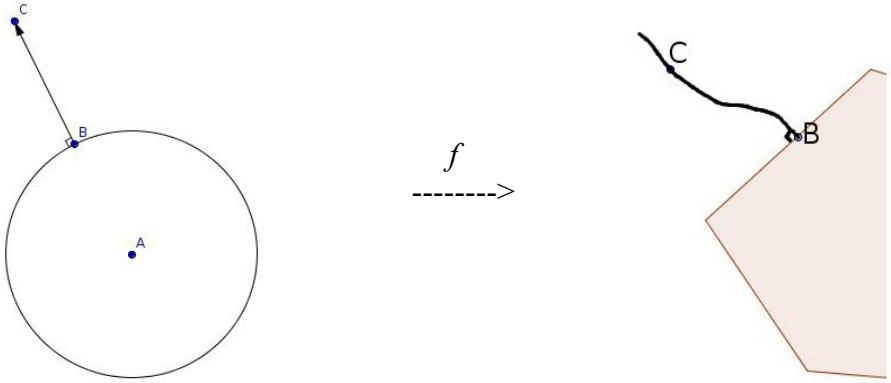
$$\forall \omega \in \Omega, \quad T(\omega) = \tilde{T}(z) \quad \text{si} \quad z = f^{-1}(\omega)$$

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.

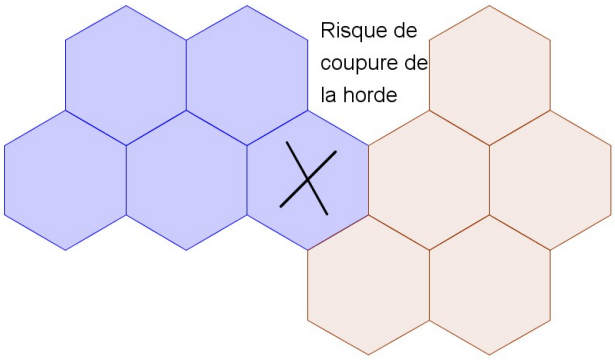
Dans la pratique, ce déplacement se fait en deux étapes. Avec Matlab, on a alors créé une fonction ***enleve*** pour enlever un manchot de sa position et mettre à jour les données de ses anciens voisins. Puis on a utilisé la fonction ***ajoute*** présentée dans la partie 4.1, pour le mettre à la place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

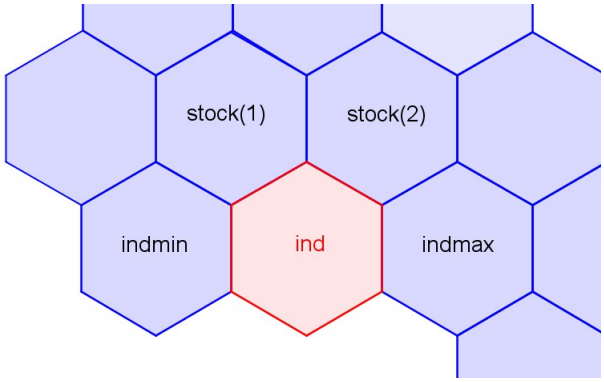
En effet, une fois ordonné, ce vecteur serait le suivant : $libre = [2 \;\; 5]$. On remarque qu'il y a un écart supérieur à 1 entre deux valeurs successives de ***libre***. Ceci est caractéristique de la situation à exclure, mis à part quand cet écart est dû à la présence simultanée de 1 et 6 dans le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

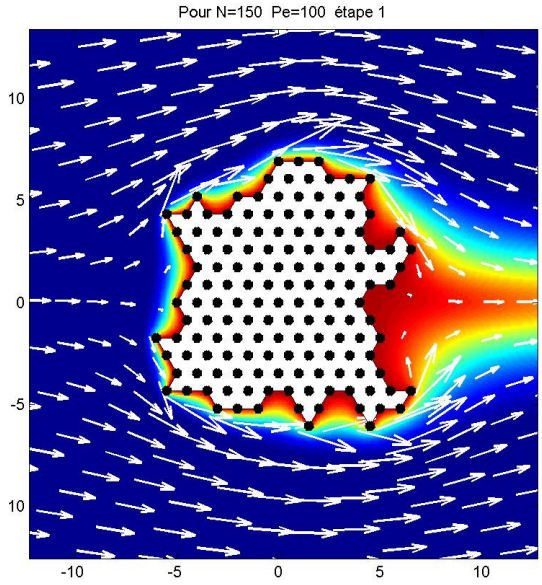
$$indfront = [(stock_2),\ indmax\ \ ..\ ..\ ..\ indmin,\ (stock_1),\ (stock_3)]$$

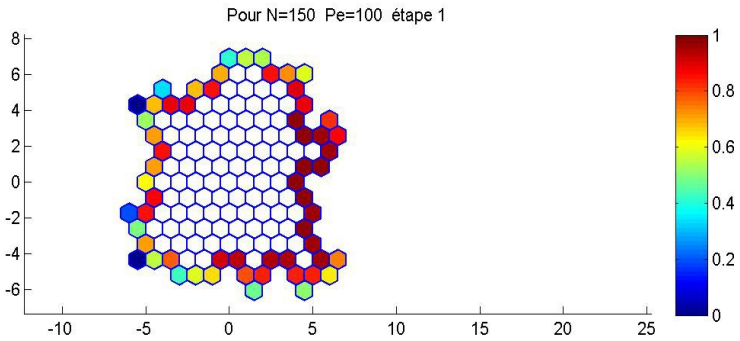
5. RÉSULTATS

5.1 Tendances générales

Nous avons réalisé les simulations en considérant les deux types de polygones présentés dans la section 1.2.3. En effet, l'article de Aaron Waters, François Blanchette, Arnold D. Kim montrait l'utilisation du premier modèle avec le polygone passant par les centres des hexagones. Il nous est toutefois paru plus naturel de considérer le polygone du second modèle traçant le contour de la horde (suivant les sommets des hexagones). Nous avons remarqué que ce second modèle n'aboutissait à aucune conclusion significative. En effet, la horde de manchots adopte une forme hexagonale assez rapidement. Nous distinguons deux cas différents, parfois la horde continue à avancer en conservant la même forme, et parfois elle reste bloquée (avec toujours le même manchot qui se déplace entre les deux mêmes positions). Ce phénomène ne reflète pas du tout la réalité, d'autant plus que même en faisant varier la force du vent, les résultats restent inchangés.

Nous choisissons alors de réaliser toutes les simulations avec le modèle n°1 comme l'ont fait les chercheurs. Pour interpréter les résultats, nous avons utilisé deux types de représentations. Une première figure nous permet de visualiser la répartition du vent et de la température autour de la horde à chaque étape. Une seconde figure nous permet de visualiser l'évolution des manchots sur la banquise ainsi que le taux de perte de chaleur associé à chaque individu frontalier. Nous avons utilisé un code couleur s'échelonnant entre le rouge et le bleu afin de montrer les manchots qui ont chaud et ceux qui ont froid.





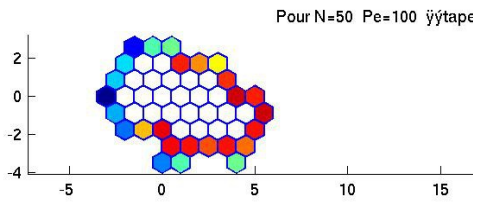
5.1.1 Les déplacements prioritaires.

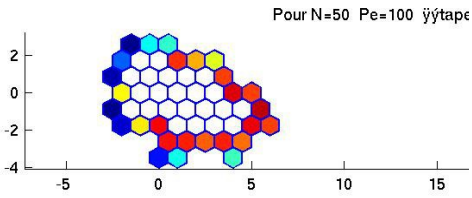
On va ici mettre en lumière les déplacements prioritaires observés, c'est à dire, grâce aux simulations faites, quels manchots sont les plus susceptibles de bouger ? Et à quels endroits ?

Dans nos simulations, on a supposé que le vent venait de gauche, ainsi, le manchot le plus susceptible de partir est situé parmi les manchots du côté gauche de la frontière. En effet, en analysant la première figure de la partie 5.1, on constate clairement que les zones situées à droite de la horde sont beaucoup plus chaudes que les autres.

De plus, nous constatons qu'un deuxième paramètre rentre en compte, c'est celui du nombre de voisins. En effet, au moins un manchot a de voisins, au plus il ressent le froid.

Sur la figure ci-dessous est représentée la valeur du Heatloss pour chacun des manchots frontaliers à l'aide du code couleur. On peut remarquer le déplacement du manchot qui avait le plus grand Heatloss (bleu foncé) vers l'emplacement ayant pour voisin deux manchots de Heatloss faible (rouge foncé).



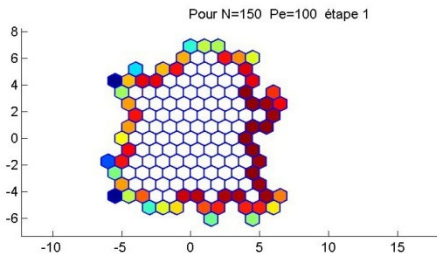


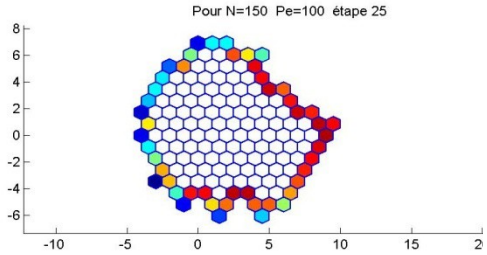
On remarque clairement que le manchot qui s'est déplacé est celui qui était le plus situé à gauche, il ne possédait d'ailleurs que trois voisins. Les deux voisins qui l'ont accueilli sont les plus à droite.

5.1.2 Évolution de la forme et de la position de la horde

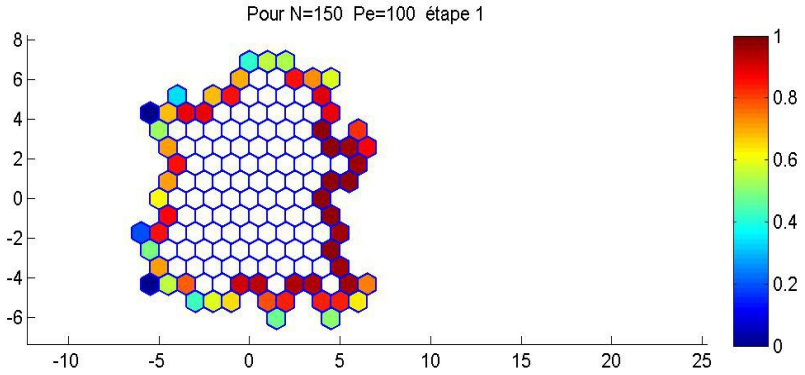
Avec les constatations faites ci-dessus, et se faisant à répétitions, la horde a alors tendance à s'affiner, à s'allonger. En effet, basons-nous sur l'exemple suivant pour constater les résultats. Il s'agit d'une simulation réalisée avec 150 manchots, un nombre de Péclet de 100, et un rayon maximal de 3.

Nous constatons dans un premier temps que la horde cherche à se régulariser. Comme nous le voyons sur les deux figures ci-dessous, la frontière de la horde est plus régulière au bout de 25 étapes et n'a pas avancé.

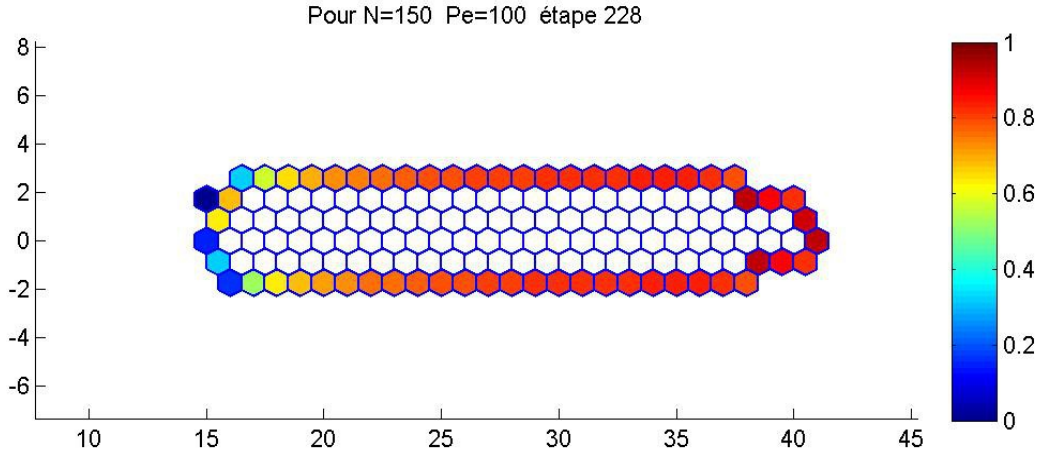




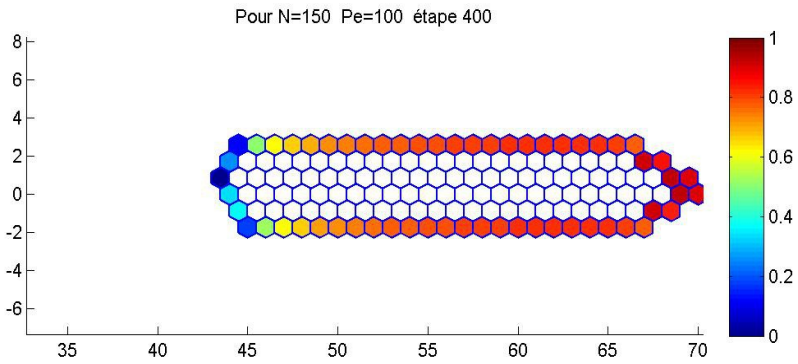
Nous voyons de même dans les graphiques ci-dessous le phénomène d'allongement de la horde :
Étape 1 :



Étape 228 :



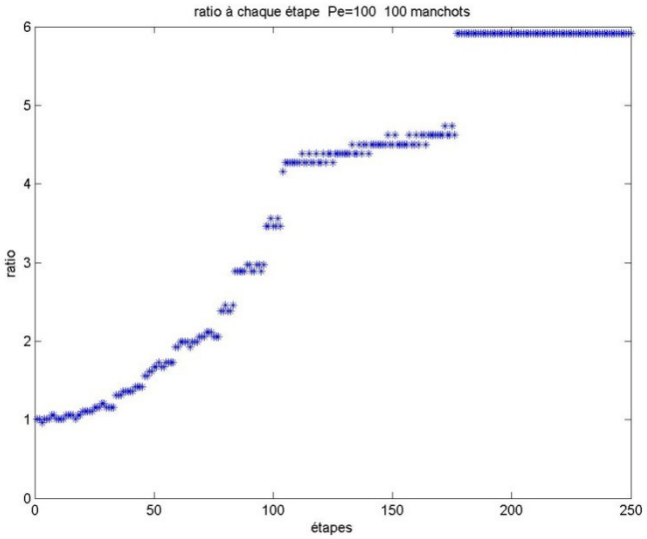
Dans un second temps, comme les manchots situés sur la gauche de la horde ont tendance à aller vers la droite, on peut alors observer que la horde se déplace globalement vers la droite avec le temps (en se référant aux axes). Mais même une fois la forme allongée, au fur et à mesure du temps, la horde continue d'avancer. Sur la figure ci-dessous, on y voit que la horde a progressé encore à la 400e étape.



Afin de quantifier l'allongement au cours du temps, nous avons défini la grandeur suivante :

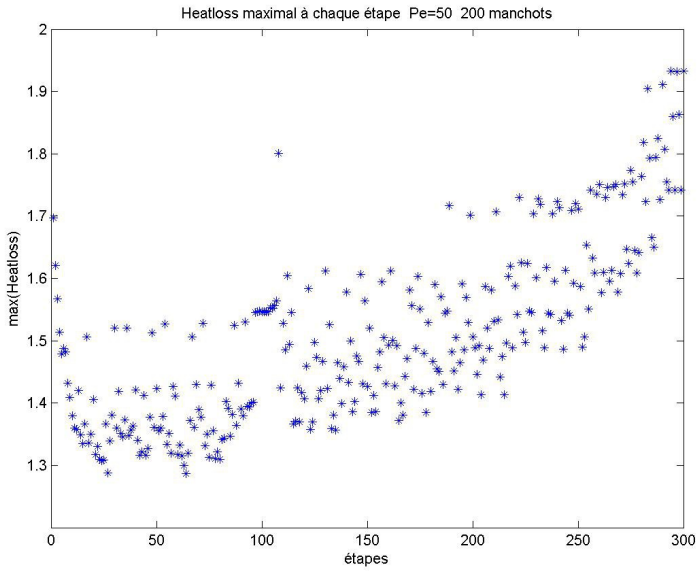
$$Ratio = \frac{largeur\ de\ la\ horde}{hauteur\ de\ la\ horde}$$

Voici pour une simulation un graphique représentant l'évolution de ce ratio en fonction des étapes.



On remarque que l'allongement augmente au cours du temps pour ensuite se stabiliser. Ce graphe nous montre notre résultat cité plus haut expliquant que la horde cherche tout d'abord à adopter une forme régulière. En effet, le ratio n'augmente pas beaucoup en début de simulation, mais par la suite, le ratio augmente plus franchement. Nous notons l'existence de paliers qui peuvent s'expliquer ainsi : Imaginons que la horde ait plus d'étages que le nombre auquel elle doit se stabiliser. Il va alors falloir retirer un étage d'hexagones, mais les manchots s'en iront un par un et cela prend un certain nombre d'étapes. Voilà pourquoi le ratio peut rester constant pendant un cours instant pour ensuite augmenter brutalement. En effet, le calcul du ratio se fait à l'aide des coordonnées des manchots les plus excentrés. Alors, même s'il ne reste qu'un manchot sur une ligne donnée, le ratio est calculé de la même manière que si la ligne était complète.

Sur le graphique suivant, nous avons représenté l'évolution du Heatloss maximal obtenu à chaque étape pour une simulation donnée.



Nous constatons que le Heatloss maximal tend à augmenter au cours du temps. Cela est dû au fait que la horde est plus allongée et que souvent le manchot se trouvant le plus à gauche ne possède plus beaucoup de voisins au bout d'un certain temps de simulation.

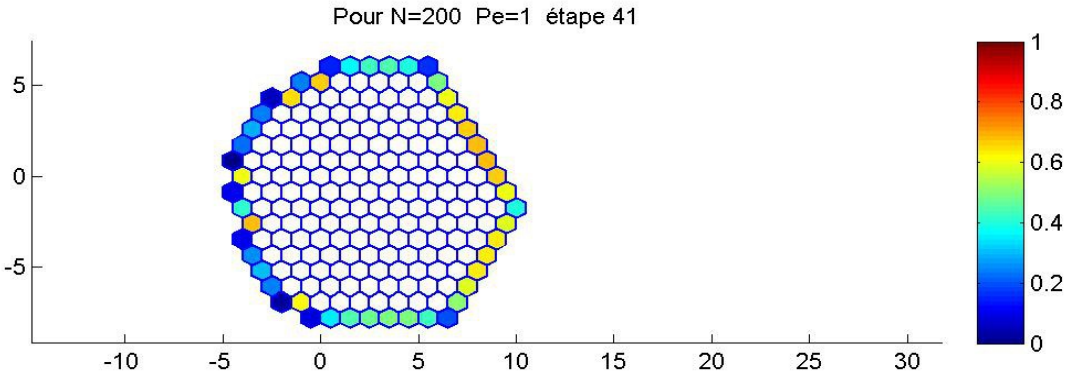
Nous nous sommes intéressés de la même manière à la valeur du Heatloss moyen obtenu à chaque étape au sein de la horde. Nous n'avons remarqué aucune tendance explicite car ce taux était trop fluctuant. On peut alors en conclure que la stratégie individuelle de chaque manchot ne permet pas à la horde de réduire son exposition au vent de manière globale.

5.2 Influence des paramètres sur les résultats

Nous nous sommes demandés si les phénomènes mis en évidence précédemment dépendaient de certains paramètres. Nous avons alors réalisé plusieurs simulations en modifiant les paramètres suivants :

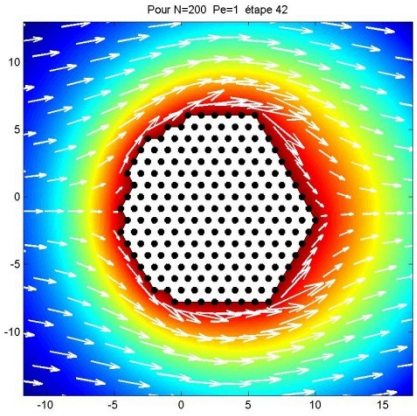
- Nombre de manchots
- Constante de Péclet
- Rayon maximal du domaine de résolution (r_{max})

Nous avons alors constaté que pour une valeur de la constante de Péclet très faible (autrement dit un vent faible). Il n'y avait pas d'avancée de la horde. On arrive à un état d'équilibre qui est le même que celui constaté avec le modèle numéro 2 (voir partie 6.2).



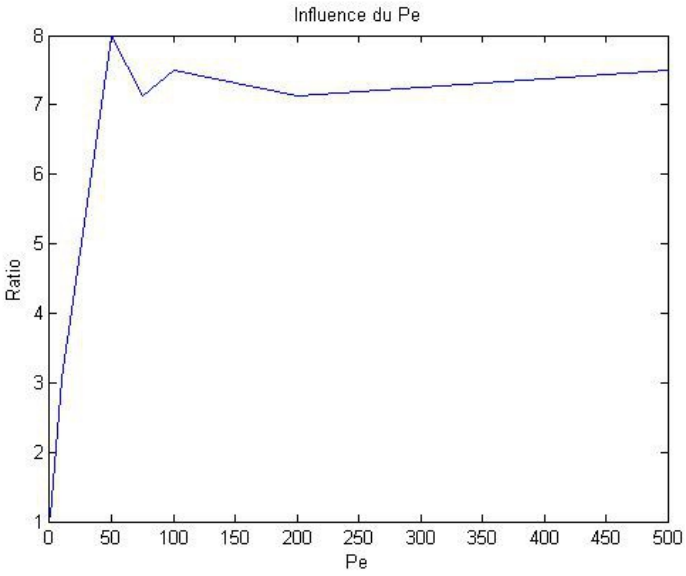
Nous constatons dans ce cas que le ratio est égal à 1.

Dans cette situation, un manchot part de sa place pour y revenir à l'étape d'après.
En regardant de plus près à la figure suivante, on remarque bien qu'une valeur de la constante de Péclet faible implique que la chaleur se distribue plus largement et de manière plus symétrique entre le côté gauche et le côté droit. Ce phénomène s'explique par une force du vent qui est alors faible.

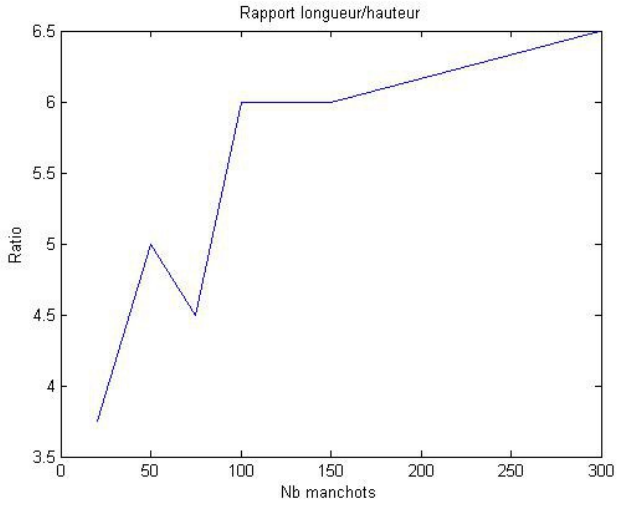


En ce sens, nous avons cherché à déterminer la valeur minimale de la constante de Péclet pour laquelle la horde commence à se déplacer. Nous avons commencé à observer des déplacements à partir de $Pe=8$ pour une horde de 100 manchots.

C'est alors que nous avons étudié l'impact que la constante de Péclet pouvait avoir sur nos simulations. Nous avons donc pris différentes valeurs de la constante et effectué plusieurs simulations pour un même nombre de manchots. Et nous avons pu en tirer le graphe suivant :



Le graphe (obtenu avec des simulations à 200 manchots) nous montre bien qu'à vent faible la horde ne s'affine pas significativement et garde à peu près son ratio d'origine. Mais qu'avec un Péclet plus fort (Pe > 50) , le vent se fait sentir et alors le ratio est plus élevée. La horde devient alors plus allongée. On remarque qu'à partir d'une certaine valeur de la constante (environ 100), le ratio reste quasiment le même.



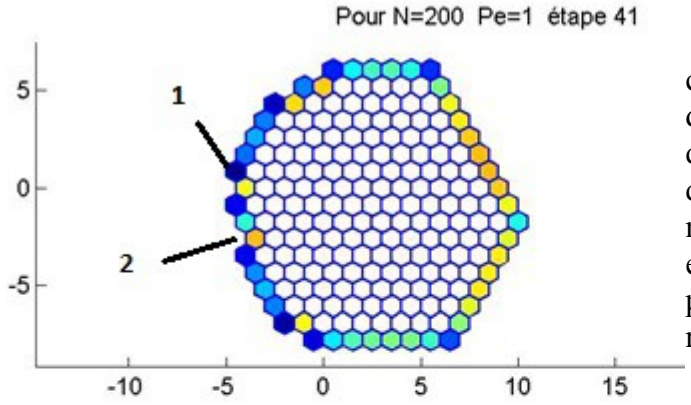
Outre la constante de Péclet, nous nous sommes demandé quel impact avait le nombre de manchots sur nos résultats, de la même façon, nous avons fixé les autres paramètres et fait de multiples simulations en changeant le nombre de manchots pour obtenir le graphe ci-contre.

Le graphe nous montre que le nombre de manchots a tendance à faire augmenter le ratio, même si il a l'air de se stabiliser lorsque le nombre de manchots devient grand.

Pour ce qui est du choix du r_{max} , nous avons testé différentes valeurs. Nous remarquons que lorsque ce nombre dépasse 3, les résultats restent inchangés. Par contre, il ne faut pas prendre une valeur trop petite car le changement de température en bordure de horde serait trop brutal. Nous choisissons alors de le prendre égal à 3. En effet, ayant le choix, nous préférons la valeur la plus petite possible car, à nombre de nœuds de calculs égal, elle offre une discrétisation plus fine.

5.3 Explications des résultats obtenus avec le second modèle

Tout d'abord, rappelons que dans un premier temps la horde cherche à adopter une forme régulière. Le problème dans ce modèle est que lorsque la forme est devenue régulière, elle ressemble à un grand hexagone. Alors, tous les manchots situés sur un même côté de l'hexagone sont confrontés exactement aux mêmes conditions climatiques. Sauf pour les manchots situés aux sommets de l'hexagone ou ceux situés sur un côté qui n'est pas parfaitement lisse. En effet, ceux-là possèdent moins de voisins que les autres et sont donc plus exposés au froid. La figure ci-dessous présente un exemple de situation pour laquelle la horde ne se déplace plus.



Le manchot numéroté 1 sur la figure ci-contre est celui qui a le plus froid. Il se déplace alors vers la position numérotée 2 qui est la plus chaude. Le problème est qu'après ce déplacement, ce même manchot redevient celui ayant le plus froid et la position qu'il avait quitté devient la position la plus chaude. Il va alors regagner sa position d'origine.

Nous arrivons alors à une situation de blocage où le même déplacement aura lieu en boucle.

6. BILAN DES RECHERCHES

6.1 Discussion au terme de l'étude

Au cours de ce TER, nous avons pu prendre conscience des différents aspects d'une modélisation. En effet, nous avons eu la chance de pouvoir créer un modèle de simulation dans son intégralité. Cela nous a permis de bien en comprendre les enjeux et les points qui nécessitent une attention particulière.

Tout d'abord, nous avons réalisé l'importance de la partie théorique pré-programmation. Nous pensons en effet qu'il est indispensable de bien maîtriser toutes les notions utilisées afin de pouvoir interpréter correctement les résultats obtenus, et surtout de pouvoir les discuter.

Ensuite, nous avons pu réaliser l'influence des hypothèses du modèle sur les résultats obtenus. Dans notre cas, nous avons en effet observé des divergences très significatives entre les deux modèles utilisés pour définir la frontière. De plus nous avons réalisé qu'un modèle pour lequel nous n'aurions pas forcément opté à l'origine peut s'avérer être plus réaliste par la suite.

Au niveau de la phase de programmation, ce qui nous a posé le plus de soucis et donc pris le plus de temps était la génération aléatoire de la horde ainsi que sa mise à jour automatique à chaque étape. Ces faits ne sont pourtant pas les plus compliqués au niveau théorique, mais ce qui nous a parfois apporté des difficultés est que l'on devait toujours être capable d'anticiper toutes les configurations possibles, même les plus inattendues au niveau de la géométrie de la horde.

Durant notre étude, nous avons rencontré un problème au niveau de l'implémentation de la génération aléatoire de la horde. Nous avions commencé la partie programmation par cette phase et voyant que tout fonctionnait parfaitement, nous sommes passé à la phase de programmation de la partie résolution numérique. Seulement, au moment de faire la liaison entre ces deux phases principales, nous avons réalisé que nous avions omis un besoin nécessaire permettant de faire cette liaison. En effet, la horde était générée grâce aux éléments ***groupe*** et ***indfront*** présentés dans la partie 4.1, seulement le vecteur ***indfront*** n'était pas ordonné et donc on ne pouvait pas créer le polygone nécessaire à la résolution. Nous avons tenté en vain de créer une fonction permettant de le réordonner, mais cela s'est avéré trop compliqué. Nous avons alors dû reprendre notre méthode de génération de la horde afin que le vecteur ***indfront*** soit ordonné systématiquement à chaque ajout d'un manchot. En pratique, cela s'est traduit par l'implémentation de la fonction ***ajoute*** qui nous sert au moment de la génération de la horde ainsi qu'au moment de sa mise à jour durant la simulation. Ceci a bien sûr engendré une perte de temps conséquente qui aurait été évitée si nous avions mieux anticipé les besoins.

Nous en retenons alors la leçon qu'il est indispensable d'avoir une vision globale sur notre travail et éviter de traiter des parties de manière trop indépendantes les unes des autres.

Enfin, nous constatons grâce à ce travail l'importance d'optimiser les coûts de calcul car nous réalisons au moment de lancer la simulation que nous sommes limités dans le nombre de manchots et dans le nombre de nœuds du maillage si l'on ne veut pas des temps de calcul trop longs. Par exemple, une simulation avec 400 manchots et des valeurs $n_1 = 150$ et $n_2 = 300$ dure environ 4 heures.

Notons également qu'après s'être inspirés de l'article fait par les chercheurs, nous avons cherché à poser notre propre modèle et notre propre façon de faire en mettant en lien la théorie et la pratique. Nous avons pu ajouter des illustrations et démontrer des résultats évoqués implicitement. Par exemple, un fait notable est que nous avons ajouter sur nos représentations de la température en dehors de la horde l'affichage du vent. Nous avons remarqué qu'il était primordial dans cette étude, c'est alors que nous avions cherché plus d'explications de ce facteur et voulu voir le lien serré entre le vent et la perte de chaleur des manchots.
Nous avons aussi, repris toute la théorie et cité les théorèmes utilisés pour mener à bien la cohérence entre les hypothèses et la pratique. Nous avons par exemple démontré un fait très important quant aux applications conforme qui est le passage de la solution d'un domaine à un autre. En voulant approfondir l'étude et proposer une modélisation plus réaliste, nous avions aussi mis en place le modèle numéro 2 qui prenait comme polygone frontalier le contour des hexagones.

6.2 Améliorations possibles

À la fin de nos travaux, nous nous posons la question de savoir comment nous pourrions améliorer le modèle à l'avenir et comment modifier certaines hypothèses dans le but d'avoir des résultats encore plus réalistes et concluants.

La première idée qui nous vient en tête est de remettre en question l' hypothèse principale posée sur le vent disant qu' il n' y avait pas de tourbillons. En effet, comme nous pouvons le voir sur la figure ci-dessous, en présence d' un obstacle, le champ de vecteurs devient en réalité rotationnel dans certaines zones.

Nous devinons que la prise en compte des tourbillons modifierait considérablement les températures en certains lieux, notamment dans les zones de concavité du domaine de résolution pour lesquelles, dans notre modèle, le vent est nul. Seulement, si nous prenons en compte les tourbillons, il y aura un deuxième terme présent dans la décomposition de Helmholtz-Hodge et le vent ne s'écrira plus comme le gradient d'un potentiel. Il faudrait alors rechercher une autre méthode qui nous permettrait de faire passer la solution d'un domaine à un autre.

Toujours concernant le vent, nous pourrions ajouter des variations d'intensité et de direction plutôt que de le garder constant.

Dans notre modélisation, nous pourrions également introduire le facteur temps, ce qui permettrait d'obtenir une évolution de la horde en temps réel.

Une autre hypothèse qui fonde le modèle est celle disant qu'un seul manchot à la fois se déplace pour regagner la position la plus chaude parmi celles de la frontière. Cette hypothèse est tout à fait discutable car en réalité chaque manchot raisonnant de manière individuelle, il se pourrait que plusieurs manchots se déplacent simultanément. Un autre modèle aurait aussi pu être testé en supposant qu'un manchot qui a froid pénètre à l'intérieur de la horde plutôt que de rechercher une position sur la frontière.

Ensuite, il est important de constater que toutes nos hypothèses font de notre modèle un modèle déterministe qui ne dépend que de la horde générée au départ. Ceci est très discutable car les modifications au sein de la horde dépendent à tout instant de la décision individuelle de chaque oiseau. Il serait alors intéressant d'ajouter à notre modèle un facteur permettant de prendre en compte l'aspect aléatoire. Ce facteur pourrait être par exemple l'ajout à la valeur du Heatloss de chaque manchot frontalier d'une variable aléatoire de nature bien choisie.

Enfin, nous pensons qu'il pourrait être intéressant d'implémenter les programmes à l'aide d'un langage de programmation plus performant comme le C++ afin d'être gagnant en terme de temps de calcul. Seulement, nous faisons appel dans ce travail à des fonctions Matlab très utiles, notamment au niveau des représentations graphiques. L' idée serait alors de créer des programmes en C++ pour l'aspect résolution numérique, puis de récupérer les résultats pour ensuite les transmettre à Matlab dans un but de représentation.

6.3 Enrichissements personnels

Au terme de ce Travail Encadré de Recherche, il est intéressant de faire le point sur ce que cela nous a apporté au niveau personnel. En effet, nous avons tous les deux le sentiment d'avoir beaucoup appris de ce travail.

Tout d'abord, comme nous le disions dans la section 6.1, il s'agit de notre première réalisation d'une modélisation complète. Nous en retirons alors beaucoup de conclusions quant aux points pour lesquels il faut être très attentifs et rigoureux que nous avons déjà détaillé auparavant.

Ensuite, ce travail nous a permis d' appliquer les théories et méthodes étudiées au cours des deux semestres de Master 1 mais également au cours de nos années de Licence. En particulier, nous avons retrouvé beaucoup de notions traitées dans les modules de MADF, TIAN et POOCN concernant la méthode des différences finies ainsi que les théories étudiées en cours d'analyse complexe de Licence 3. Notre étude nous a permis de bien comprendre les enjeux exprimés durant les cours et surtout de réaliser la puissance des outils que l'on nous a appris à utiliser. Ce travail nous aide alors à réaliser la diversité des domaines d'applications possibles des méthodes que l'on nous apprend.

Enfin, nous constatons que l'implémentation des méthodes nous a permis de beaucoup progresser au niveau de notre logique de programmation. Nous avons d'ailleurs appris à utilisé le logiciel Matlab qui nous était jusqu'alors complètement étranger.

6.4 Remerciement

Nous tenons à remercier particulièrement Monsieur Bernhard Beckermann pour toute l'aide qu'il a pu nous apporter en nous accompagnant dans notre Travail Encadré de Recherche.

7. SOMMAIRE DES FONCTIONS CRÉÉES SOUS MATLAB

Voici ci-dessous la liste des fonctions que nous avons dû créer. Dans ce tableau figurent les arguments d'entrée et sortie pour chacune d'entre-elles, ainsi que la description de leur rôle.

Fonctions concernant la température et le vent :

Nom	Arguments d'entrée	Arguments de sortie	Rôle
tempdisk	r_{max} : rayon extérieur de l'anneau n_1 : nombre de subdivisions pour le rayon n_2 : nombre de subdivisions pour l'angle Pe : Constante de Péclet Th : Température au sein de la horde Ti : Température au loin de la horde	- z contient les affixes des points de calcul - x contient les solutions aux points dans z - znormal contient les affixes des points sur le cercle - normal approche la dérivée normale aux points dans znormal	Détermine la température en tout point du maillage de l'anneau
temppoly	- zdisk est un vecteur qui contient les coordonnées des points où nous avons effectué nos calculs dans l'anneau - xdisk est un vecteur qui contient les températures aux points dans zdisk - sc est l'application de Schwarz-Christoffel	- zpoly est un vecteur qui contient les affixes concernées autour de p - xpoly est un vecteur qui contient les températures aux points d'affixes contenues dans zpoly	Après avoir obtenu nos résultats autour du disque avec la fonction tempdisk. Cette fonction nous permet via l'application de Schwarz-Christoffel d'obtenir les températures correspondantes aux points correspondants autour du polygone
evaluer	- chaine qui sert à choisir le type d'évaluation que l'on souhaite effectuer (dérivée, inverse…) - sc1, c'est l'application crée par le package Tolbin - x est l'affixe du point où l'on souhaite évaluer l'application	- y est l'image de x	Permet d'évaluer l'application Schwarz-Christoffel , ainsi que son inverse, sa dérivée… l'application correspond à celle générée par le package Tolbin rectifiée par la composition avec une rotation

heatloss	- indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique - p est le polygone correspondant à la frontière, tel que p(i) contienne l'affixe du manchot d'indice indfront(i) -sc est l'application de Schwarz-Christoffel - znormal, vecteur qui contient les affixes des points de calcul - normal contient les dérivées normales en chaque point de znormal	- Heatloss est un vecteur qui contient les pertes de chaleur des manchots de indfront	Permet de calculer la perte de chaleur de chaque manchot situé sur la frontière
heatloss2	- Mêmes arguments que pour heatloss avec celui-ci en plus : - der est un vecteur tel que der(i) contient pour un manchot indfront(i) l'indice du polygone p correspondant au dernier sommet du manchot indfront(i)	- Heatloss est un vecteur qui contient les pertes de chaleur des manchots de indfront	A la même fonction que heatloss mais est adaptée au 2^e modèle étudié
integrale	- zmoins et zplus sont les bornes de l'intégrale - sc est l'application de Schwarz-Christoffel - znormal, vecteur qui contient les affixes des points de calcul - normal est le vecteur qui contient les approximations des dérivées normales en chaque point de znormal	- Int est la valeur de l'intégrale calculée	Cette fonction permet de calculer l'intégrale entre le long du polygone de la dérivée normale de la température

Notons que c'est en grande partie au sein des fonctions ci-dessus que nous avons besoin du package de Tobin A. Driscoll concernant les applications de Schwarz-Christoffel.

Les fonctions utilisées dans ce package sont les suivantes :

- ***polygon*** : génère un polygone grâce aux données des affixes des sommets
- ***extermap*** : construit l'application de Schwarz-Christoffel envoyant l'extérieur d' un polygone donné sur l'extérieur du disque unité
- ***evaldiff*** : évalue la dérivée de l'application de Schwarz-Christoffel créée en un point
- ***evalinv*** : évalue l' inverse de l'application de Schwarz-Christoffel créée en un point

Fonctions concernant la génération et les modifications de la horde :

Nom	Arguments d'entrée	Arguments de sortie	Rôle
ping			Ping est la classe sur laquelle on va travailler. Un manchot de la classe ping a son indice, sa position en abscisse x, sa position en ordonnée y, et le vecteur « libre » qui contient les indices des voisins libres
horde	- N est le nombre de manchots que l'on souhaite avoir dans la horde que nous créons	- groupe est un vecteur contenant les caractéristiques de tous les manchots de notre horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	Créer une horde de N manchots en suivant nos conditions de construction
ajoute	- indnew est l'indice du manchot que l'on souhaite ajouter - indv1 et indv2 sont les manchots auxquels on compte coller le manchot arrivant indnew - dirv2 est la direction (en partant de indv1) dans laquelle on va placer indnew - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique - param vaut 1 si il s'agit de son utilisation pour la simulation, vaut 2 si il s'agit de son utilisation pour la génération de la horde	- groupe est la horde mise à jour après ajout de indnew - indfront est mis à jour lui aussi - ok est un booléen égal à 1 si l'ajout s'est effectué, 0 sinon	Ajoute un manchot à la horde en mettant à jour la horde et le vecteur indfront. (Peut ne pas ajouter de manchot si les 3 conditions pour placer un manchot ne sont pas satisfaites, dans ce cas là il renvoie un booléen ok=0 sans changer la horde)

enleve	- ind est l'indice du manchot que l'on souhaite enlever - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- H est la horde mise à jour - b est le vecteur indfront mis à jour - ok est un booléen qui vaut 1 si le manchot a bien été enleveé, 0 sinon	Permet de retirer un manchot de la horde tout en respectant nos conditions de construction de la horde
deplacement	- pos est la position du manchot d'où l'on part -dir est la direction dans laquelle on part du manchot qui se situe en position pos	- position contient les coordonnées de là où l'on arrive en étant parti du manchot en pos dans la direction dir	- Permet d'obtenir la position d'arrivée après être parti d'un manchot en allant dans une direction choisie
detect	- ind est l'indice d'un manchot - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- u est le vecteur contenant les indices dans indfront des voisins frontaliers de ind	- Donne les indices des manchots frontaliers de celui rentré en argument
detect2	- pos est la position (x,y) du manchot que l'on étudie - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés	u est le vecteur contenant les indices dans indfront des voisins frontaliers de ind	- Donne les indices des manchots frontaliers de celui rentré en argument
present	- pos est la position (x,y) que l'on teste - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- test est un booléen de valeur 1 si il y a un manchot, 0 sinon	Teste si un manchot occupe la position pos
testdir	- j est la direction dans laquelle on veut vérifier si il y a un manchot - test est le manchot duquel on part - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés.	- bool est un booléen de valeur 1 si un manchot est présent dans la direction j en partant de test, 0 sinon	Vérifie si un manchot est présent dans une direction j en partant d'un manchot test

trou	- pos est la position test - u contient les indices des voisins libres de pos - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- bool est un booléen de valeur 1 si l'on créer un trou, vaut 0 sinon	Vérifie si l'ajout d'un manchot à la position pos nous fait créer un trou dans la horde ou non
localise	- H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique - P est le manchot duquel on part - j est la direction dans laquelle on part depuis le manchot P	- manchot est de type ping, il s'agit du manchot voisin de P dans la direction j (on travaille uniquement sur les frontaliers)	Détecte le manchot qui est le voisin de P dans la direction j
enferme	- pos est la position que l'on veut tester - j est la direction dans laquelle on part pour nos tests (partant de pos) - H est la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- bool est un booléen égal à 1 si la position pos est dans un trou du maillage, 0 sinon	Teste si la position pos est dans un trou du maillage ou non
admi	- pos contient les coordonnées (x,y) de l'endroit où l'on veut tester si un manchot est admissible (selon nos critères de modélisation) - H la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- bool, qui a pour valeur 1 si pos peut accueillir un manchot, 0 sinon - u est un vecteur qui contient les indices correspondants aux voisins libres de pos	Teste si la position pos respecte les 3 conditions pour accueillir un manchot, renvoie le booléen correspondant
oppose	- i la direction de base	- j la direction opposée à i	Renvoie la direction opposée à la direction i

poly_contour	- H la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique - p le polygone passant par les centres des manchots frontaliers obtenu par la fonction hordepolycentre.	- p2 le polygone passant par les sommets des hexagones - der , vecteur contenant les indices dans p2 du dernier sommet de l'hexagone du manchot dans p	- Construit le polygone décrivant le contour de la horde (passant par le contour des hexagones et non pas par les centres des hexagones)
hordepolycentre	- H la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- p le polygone dont les sommets sont les affixes des centres des manchots frontaliers	Construction du polygone décrivant la frontière. Il s'agit du polygone qui sera soumis souvent aux applications Schwarz-Christoffel
newhorde	- H la horde - Heatloss, le vecteur contenant les pertes de chaleur de tous les manchots frontaliers. - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique	- Hnew la horde mise à jour après déplacement du manchot - indfrontnew, le vecteur indfront mis à jour après déplacement du manchot	Déplace le manchot ayant la plus grande perte de chaleur et le place à côté de celui ayant la plus petite (en respectant les conditions de notre modèle). Puis met à jour la horde et le vecteur indfront
Position	- pos correspond aux coordonnées (x,y) du manchot de référence - dir est la direction dans laquelle on part depuis pos	- position contient les coordonnées de là où l'on arrive en étant parti du manchot pos dans la direction dir	Permet d'avoir la coordonnée de l'emplacement potentiel d'un manchot en partant d'un manchot pos dans une une direction dir

Fonctions permettant la visualisation des résultats :

Nom	Arguments d'entrée	Arguments de sortie	Rôle
dessin	- H la horde - indfront, le vecteur qui contient les manchots frontaliers ordonnés dans le sens trigonométrique - p le polygone représentant la frontière de la horde		Représente la horde de manchots (avec les hexagones)

dessin2	- H la horde - p le polygone représentant la frontière de la horde - s la taille du point au centre de l'hexagone représentant le manchot (dépend du nombre de manchots)		Représente la horde de manchots (en dessinant le contour et représentant chaque manchot avec un point)
hexagone	- x la partie réelle de z - y la partie imaginaire de z		Crée un hexagone situé autour du point d'affixe z de coordonnées (x,y)
hexagone2	- x la partie réelle de z - y la partie imaginaire de z - v vecteur à trois composantes servant à coder une couleur		Crée un hexagone situé autour du point d'affixe z de coordonnées (x,y), et coloriant l'intérieur d'une couleur codée par v. (utilisée pour représenter les manchots frontalier avec leur heatloss)
point	- z_1 affixe du centre de l'hexagone dans lequel on travaille - dir la direction du sommet dont on veut avoir l'affixe	- z_2 l'affixe du sommet de l'hexagone centré en z_1 en étant parti dans la direction dir	Donne l'affixe du sommet choisi d'un hexagone en ayant rentré en paramètre le centre de l'hexagone
dessinventpoly	- p le polygone représentant la horde - sc l'application de Schwarz-Christoffel		Représente le vent autour du polygone (La force du vent ne rentre pas dans les paramètres parce que la représentation resterait inchangée car le champ de vecteur serait proportionnel et la fonction quiver donnerait le même résultat)
dessintemppoly	- z contient tous les points du maillage - t est un vecteur contenant les températures des affixes stockés dans z - n représente le nombre de subdivisions pour les deux axes - H la horde		Représente la température en couleur autour du polygone (rouge pour les zones chaudes, bleu pour les zones froides)

Heat_color	- p le polygone - Heatloss le vecteur contenant les pertes de chaleur de tous les manchots frontaliers		Représente par une échelle de couleurs les valeurs du Heatloss pour les manchots de la frontière. (rouge = Heatloss faible, bleu = Heatloss élevé)
circus	- n le nombre de côtés que l'on souhaite que le polygone ait	- p le polygone dessiné	Construit un polygone régulier à n côtés
maillage	- p le polygone frontalier		Construit un maillage autour du disque et renvoie le dessin de l'image du maillage autour du polygone p
simulation	- nb_ping le nombre de manchots que l'on souhaite avoir dans la horde -nb_etapes le nombre d'étapes que l'on souhaite exécuter - Pe la valeur de la constante de Péclet	- M1 et M2 génèrent toutes les images permettant de faire une animation. M1 stocke les images avec les hexagones, M2 stocke les images avec représentation du vent et de la température à l'extérieur de la horde	Fait une simulation de nb_etapes étapes avec nb_ping manchots et une constante de Péclet égale à Pe Ci-dessous, nous allons plus développer cette fonction

<u>Fonction simulation</u> :

La fonction simulation commence par générer une horde avec le nombre de manchots rentré en paramètre et crée aussi le vecteur indfront stockant les indices des manchots frontaliers. Puis appelle la fonction hordepolycentre pour créer le polygone frontalier.
Ensuite nous effectuons la discrétisation de l'équation de la température grâce à la fonction tempdisk.
Comme nous allons nous intéresser à certaines statistiques de nos simulations, nous créons aussi les vecteurs heatmat, heatmean, heatmax, front_compteur et ratio qui nous permettent d'extraire des données intéressantes (heatloss moyen à chaque étape, etc…).
On entre maintenant dans la boucle qui va effectuer le nombre d'étapes demandé.
Une étape se détaille de la façon suivante :
- On calcule les température autour du polygone à l'aide de la fonction temppoly.
- On calcule le vecteur Heatloss stockant les perte de chaleur des manchots frontaliers.
- On entre dans les vecteurs heatmat, heatmean, heatmax, front_compteur et ratio les informations dont nous avons besoin pour pouvoir les récupérer en sortie de boucle.
- On régle les axes pour avoir notre horde bien dessinée dans nos fenêtres.
  - On représente sur une figure la horde avec les hexagones, ceux frontaliers étant coloriés avec des couleurs montrant le Heatloss de chaque manchot. Sur une deuxième figure nous représentons le polygone avec les manchots représentés par des points, et autour du polygone sont affichés le vent et la température hors de la horde.

8. BIBLIOGRAPHIE

→ Article publié par les chercheurs Aaron Waters, François Blanchette et Arnold D. Kim http://www.plosone.org/article/authors/info%3Adoi2F10.1371%2Fjournal.pone.0050277;jsessionid=B5F2D0EBE2B8838553A7588A89A58400

→ Article " Les manchots font la tortue " http://recherchespolaires.inist.fr/?Les-manchots-font-la-tortue

→ *"Analyse numérique matricielle appliquée à l'art de l'ingénieur, tome 1: Méthodes directes"* de Patrick Lascaux, Raymond Théodor

→ Livre *"Introduction à la géométrie hyperbolique et aux surfaces de Riemann"* de Eric Toubiana et Ricardo Sá Earp (p45)

→ Numerical Computation of the Schwarz-Christoffel Transformation de Lloyd N. Trefethen Computer Science Department, Stanford University, California (USA) ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/79/710/CS-TR-79-710.pdf

→ Cours de Licence 3 (année 2008-2009) d' Analyse Complexe de Johannes Huebschmann Laboratoire Paul Painlevé, Université Lille 1

→ Cours de Master 1 (année 2012-2013) de Modélisation et Approximation par Différences Finies de Monsieur Emmanuel Creusé, Laboratoire Paul Painlevé, Université Lille 1

→ Cours de Master 1 (année 2012-2013) de Traîtement Informatique de l' Analyse Numérique de Madame Ana Matos, Laboratoire Paul Painlevé, Université Lille 1

→ Cours de Master 1(année 2012-2013) de Programmation Orientée Objet pour le Calcul Numérique de Madame Caterina Calgaro, Laboratoire Paul Painlevé, Université Lille 1