Etude d'un modèle de mouvement collectif

Yuguang XIAO[1], Jean SIERES[2] et Albert LI[3]

[1] yuguang.xiao@etu.sorbonne-universite.fr
[2] jean.sieres@etu.sorbonne-universite.fr
[3] albert.li@etu.sorbonne-universite.fr

Fait à Paris, France
26 avril 2024

Table des matières

1 Introduction **4**

2 Modèles et Méthodes numériques **5**
2.1 Mise en contexte . . . 5
2.2 Modèle à trois zones (noté 3-Z) [1] . . . 5
2.2.1 3-Z appliqué dans l'espace $\mathbb{R}^2$ . . . 6
2.2.2 3-Z appliqué dans l'espace $\mathbb{R}^3$ . . . 6
2.3 Méthodes numériques . . . 6
2.3.1 Problème de Cauchy . . . 6
2.3.2 Méthode d'Euler explicite . . . 7
2.3.3 Méthode de Runge-Kutta d'ordre 4 (noté RK4) [2] . . . 7

3 Définitions, propriétés importantes et résultats théoriques[1] **8**

4 Analyse numérique des résultats **11**
4.1 Cas 1 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(ln(r)-1)$ . . . 11
4.1.1 Analyse des simulations dans l'espace $\mathbb{R}^2$ . . . 11
4.1.2 Analyse des résultats dans $\mathbb{R}^3$ . . . 12
4.2 Cas 2 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r^2 ln(r)$ . . . 12
4.2.1 Résultats des simultions dans $\mathbb{R}^2$ . . . 12
4.2.2 Analyse des résultats dans $\mathbb{R}^3$ . . . 14
4.3 Résultats de EMG du cas 1 et du cas 2 . . . 14

5 Ajout M prédateurs **17**
5.1 Modèle d'ajout de M prédateurs (noté 3-Z-P) . . . 17
5.2 Illustrations des résultats en utilisant 3-Z-P . . . 17
5.2.1 Cas : N = 50, M = 1 en utilisant Méthode d'Euler explicite . . . 19
5.2.2 Cas : N = 50, M = 3 en utilisant Méthode d'Euler explicite . . . 19

6 Ajout des conditions de bords **22**
6.1 Ajout d'un cadre fermé dans $\mathbb{R}^2$ . . . 22
6.1.1 Problème du mur carré et analyses . . . 22
6.1.2 Observations et analyse pour l'ajout d'un mur carré pour le cas 1 . . . 22
6.1.3 Résultats du Cas N =10 ; $\phi(r) = \frac{r}{2+r3}$; $V(r) = r^2 \ln(r)$ avec un mur carré . . . 23
6.1.4 Observations et analyse pour l'ajout d'un mur carré pour le cas 2 . . . 23
6.2 Que se passe-t-il lorsque l'on modifie la taille de la boîte . . . 23
6.2.1 Resultats du Cas N = 50 ; $\phi(r) = \frac{r}{2+r3}$; $V(r) = r(\log(r)-1)$ avec un mur carré . . . 24
6.3 Cas 1 . . . 24
6.4 Cas 2 . . . 25
6.4.1 Analyse des résultats . . . 25
6.5 Resultats du Cas 1 avec N = 5 ; $\phi(r) = \frac{r}{2+r^3}$; $V(r) = r(\log(r)-1)$ avec un mur circulaire . . . 25

7 Discussions **27**

8 Conclusion **27**

9 Annexes **29**
9.1 3-Z dans l'espace $\mathbb{R}^2$ . . . 29
9.1.1 Fonction dérivée . . . 29
9.1.2 Méthodes numériques : RK4+Euler explicite . . . 30
9.1.3 Programme de la Boucle (noté PB) . . . 31
9.1.4 Code des graphes (Définitions) . . . 32
9.1.5 Applications numériques . . . 34
9.2 3-Z dans l'espace $\mathbb{R}^3$ (visualisation) . . . 36
9.2.1 Fonction dérivée . . . 36
9.2.2 Code des graphes (Définitions) . . . 37
9.2.3 Applications numérique . . . 38
9.3 3-Z-P . . . 39
9.3.1 Fonction dérivée . . . 39
9.3.2 Applications numériques . . . 40
9.4 Problème du mur carré et circulaire . . . 44
9.4.1 Cas du mur carré . . . 44
9.4.2 Cas du mur circulaire . . . 45

1 Introduction

L'étude des systèmes biologiques vise à s'interroger sur les comportements des êtres vivants, qu'ils soient des animaux ou bien des êtres humains. Ce domaine comprend l'étude des mouvements collectifs qui sont omniprésents dans la nature. Par exemple, des individus se déplaçant de façon ordonnée dans des lieux publics en ce qui concerne les êtres humains, ou bien dans des phénomènes liés à la migration des oiseaux en passant par la formation des bancs de poissons pour ne citer qu'eux. Afin de mieux comprendre ces comportements, plusieurs modèles ont été proposé selon le champ de vision d'un individu parmi le groupe ou bien celui mis en évidence dans l'article intitulé *« Asymptotic flocking for the three-zone model »*, publié par *Fei Cao, Sebastien Motsch, Alexander Reamy and Ryan Theisen,* qui modélise les intéractions des entités selon trois cercles concentriques, appelé modèle tri-zonal. Nous avons fait le choix de s'intéresser à ce dernier et utiliserons l'abréviation 3-Z pour désigner ce modèle.

Tout d'abord, il est important de définir plusieurs termes, à commencer par celui du mouvement collectif, que l'on qualifie par un système constitué de nombreuses entités similaires, qui, sous certaines conditions, agissent ensemble de manière synchronisée. Le tout sans communication directe ou de hiérarchie. Ensuite, lorsque que tous les agents s'aligneront et auront la même vitesse, nous parlerons alors d'un*flock*.

Pour comprendre ce phénomène, nous nous appuierons sur l'article mentionné ci-dessus, qui analyse les interactions entre les agents au travers de trois différents principes. La répulsion à courte distance, l'alignement à moyenne distance, et enfin l'attraction pour une longue distance.

L'objectif principal de notre étude sera d'illustrer les paramètres propices à l'émergence d'un comportement de groupe cohérent, ou *flocking* et de s'interroger sur l'influence de différentes contraintes physiques sur le *flock*.

Expliquons de manière plus précise et mathématique ce phénomène.

Définition 1 (Flocking)**.** *Si au temps t, le poisson i est à la position $x_i(t)$, avec la vitesse $v_i(t)$ et si on note la position du centre de masse $x_c = \frac{1}{N}\sum_{i=1}^{N} x_i(t)$. On dit qu'il y a flocking lorsque les vitesses des individus s'alignent sur le temps. Mathématiquement, cela signifie que la somme des carrés des différences entre la vitesse de chaque individu $\mathbf{v}_i(t)$ et la vitesse moyenne $\mathbf{v}_{lim}$ doit tendre vers 0 lorsque t tend vers l'infini.*

$$\exists \mathbf{v}_{lim} \text{ telle que } \lim_{t\to+\infty} \sum_{i=1}^{N} \|\mathbf{v}_i(t) - \mathbf{v}_{lim}\|^2 = 0$$

et que la distance maximum (supremum) entre la position de chaque individu $x_i(t)$, et la position du centre de masse x_c pour tout t, doit être finie.

$$\sup_{0\le t<+\infty} \sum_{i=1}^{N} \|\mathbf{x}_i(t) - \mathbf{x}_c(t)\|^2 < +\infty$$

Ce rapport se concentrera donc sur l'étude des mouvements collectifs, en mettant l'accent sur le concept de *flocking* dans le cas des bancs de poissons auquel cas le *flocking* se produira lorsque le groupe de poissons s'alignera sur une vitesse commune et maintiendra leur cohésion. Pour comprendre ce phénomène, nous nous appuierons sur le modèle tri-zonal, qui analyse les interactions à travers trois principes fondamentaux : répulsion à courte distance, alignement à distance moyenne et attraction à longue distance.

L'objectif principal de cette étude est de déterminer les conditions propices à l'émergence d'un comportement de groupe cohérent, ou *flocking*. Pour ce faire, nous combinons une approche multidisciplinaire, incluant la résolution de systèmes d'équations différentielles, l'analyse numérique et graphique ainsi que la programmation en Python. Les résultats de l'étude nous aideront à mieux comprendre comment le *flocking* se forme et se déplace, ou bien comment ce dernier varie lorsqu'il y a une présence de prédateurs, ou bien de murs. Cela pourra éclairer de nouvelles pistes pour de futurs travaux de recherche dans ce domaine.

2 Modèles et Méthodes numériques

2.1 Mise en contexte

Dans l'étude des systèmes complexes, le modèle à trois zones joue un rôle crucial dans la compréhension des phénomènes de groupes. Ce modèle est particulièrement pertinent pour l'observation d'interactions entre les individus sur des règles simples tout en produisant des comportements de groupe émergents et complexes. Nos observations et nos études se baseront en partie sur l'étude réalisée par Fei Cao, Sebastien Motsch, Alexander Reamy et Ryan Theisen intitulée : « Asymptotic flocking for the three-zone model ».

2.2 Modèle à trois zones (noté 3-Z) [1]

Une illustration schématique de ce modèle est présentée dans la Figure 1. Chaque individu i est défini par un vecteur de position $\mathbf{x}_i$ et un vecteur de vitesse $\mathbf{v}_i$, tous appartenant à l'espace $\mathbb{R}^d$ (où d vaut 2 ou 3). Le comportement des N individus est déterminé par le système d'équations suivant :

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v_i}, \\ \dot{\mathbf{v}}_i = \dfrac{1}{N}\displaystyle\sum_{j=1}^{N} \phi_{ij}(\mathbf{v}_j - \mathbf{v}_i) + \dfrac{1}{N}\sum_{j\neq i}^{N} -\nabla_{\mathbf{x}_i} V(\|\mathbf{x}_j - \mathbf{x}_i\|). \end{cases} \tag{1}$$

Dans cette formule, $\phi_{ij} = \phi(\|\mathbf{x}_j - \mathbf{x}_i\|)$ indique la force d'alignement entre les individus i et j. Il est supposé que la fonction ϕ est toujours positive. Par ailleurs, $\nabla_{\mathbf{x}_i} V(\|\mathbf{x}_j - \mathbf{x}_i\|)$ exprime l'effet d'attraction ou de répulsion exercé par l'individu j sur l'individu i. L'explication du gradient révèle :

$$-\nabla_{\mathbf{x}_i} V(\|\mathbf{x}_j - \mathbf{x}_i\|) = V'(\|\mathbf{x}_j - \mathbf{x}_i\|)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$

Ainsi, la dynamique entre l'agent i et l'agent j est caractérisée par une attraction si $V' > 0$ et une répulsion si $V' < 0$. La Figure 1 illustre deux exemples de configurations pour les fonctions ϕ et V.

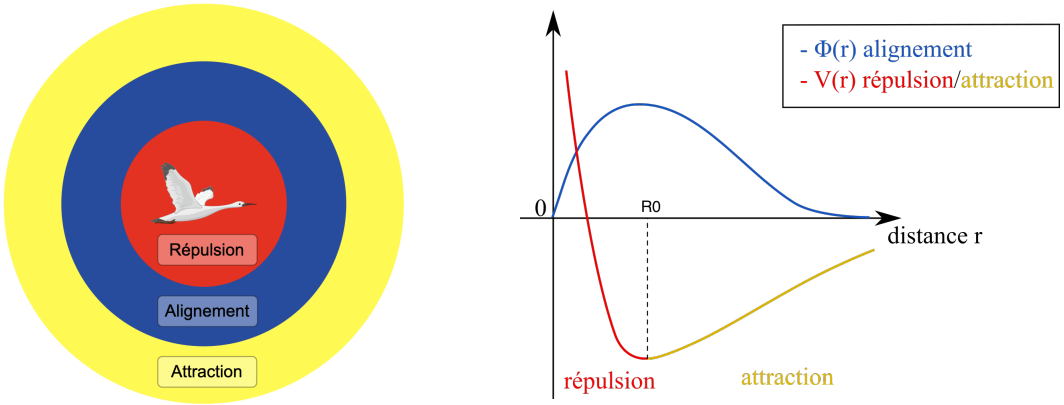

FIGURE 1 – Sur la gauche : Schéma du modèle tri-zonal illustrant les trois comportements principaux : l'attraction, l'alignement et la répulsion. Sur la droite : Les fonctions V et ϕ modélisent respectivement l'attraction et la répulsion ainsi que l'alignement.

2.2.1 3-Z appliqué dans l'espace $\mathbb{R}^2$

Introduisons ce modèle 3-Z à l'espace $\mathbb{R}^2$, où chaque individu est non seulement défini par sa position mais est également influencé par la position relative à ses voisins. Nous établissons alors un système de coordonnées cartésiennes où la position du i-ème individu est donnée par (x_i,y_i), où $i \in [\![1,N]\!]$. De plus, nous définissons la distance entre 2 individus dans l'espace $\mathbb{R}^2$: $r_{ij} = \sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$, où $j \in [\![1,N]\!]$. Détaillons alors le système (1) :

$$\begin{cases} \ddot{x_i} &= \frac{1}{N}\sum_{j=1}^{N}\phi(r_{ij})(\dot{x_j}-\dot{x_i}) + \frac{1}{N}\sum_{j\neq i}^{N}V'(r_{ij})\frac{x_j-x_i}{r_{ij}} \\ \ddot{y_i} &= \frac{1}{N}\sum_{j=1}^{N}\phi(r_{ij})(\dot{y_j}-\dot{y_i}) + \frac{1}{N}\sum_{j\neq i}^{N}V'(r_{ij})\frac{y_j-y_i}{r_{ij}} \end{cases} \tag{2}$$

2.2.2 3-Z appliqué dans l'espace $\mathbb{R}^3$

Introduisons ce modèle 3-Z à l'espace $\mathbb{R}^3$, où chaque individu est non seulement défini par sa position mais est également influencé par la position relative à ses voisins. Nous établissons alors un système de coordonnées cartésiennes où la position du i-ème individu est donnée par (x_i,y_i), où $i \in [\![1,N]\!]$. De plus, nous définissons la distance entre 2 individus dans l'espace $\mathbb{R}^2$: $R_{ij} = \sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2}$, où $j \in [\![1,N]\!]$. Détaillons alors le système (1) :

$$\begin{cases} \ddot{x_i} &= \frac{1}{N}\sum_{j=1}^{N}\phi(R_{ij})(\dot{x_j}-\dot{x_i}) + \frac{1}{N}\sum_{j\neq i}^{N}V'(R_{ij})\frac{x_j-x_i}{R_{ij}} \\ \ddot{y_i} &= \frac{1}{N}\sum_{j=1}^{N}\phi(R_{ij})(\dot{y_j}-\dot{y_i}) + \frac{1}{N}\sum_{j\neq i}^{N}V'(R_{ij})\frac{y_j-y_i}{R_{ij}} \\ \ddot{z_i} &= \frac{1}{N}\sum_{j=1}^{N}\phi(R_{ij})(\dot{z_j}-\dot{z_i}) + \frac{1}{N}\sum_{j\neq i}^{N}V'(R_{ij})\frac{z_j-z_i}{R_{ij}} \end{cases} \tag{3}$$

2.3 Méthodes numériques

2.3.1 Problème de Cauchy

Pour simuler le mouvement d'un groupe, il est courant de partir d'une situation initiale connue et d'utiliser des équations pour prédire les positions futures. Ce type de problème mathématique est appelé *problème de Cauchy*. Il est très utile dans notre étude puisqu'il nous permet de calculer les positions et les vitesses d'un ensemble d'individus à chaque instant, en connaissant leur état de départ. Cette approche facilitera notamment nos programmes Python et nos représentations graphiques. Pour un système d'équations différentielles, le problème de Cauchy de la forme $\dot{u} = f(t,u)$, où u est généralement un vecteur ou une matrice dépendant de la variable t. Dans ce projet, nous posons (par exemple en utilisant 3-Z dans l'espace $\mathbb{R}^2$) :

$$u = \begin{pmatrix} x_1 & x_2 & \cdots & x_N & y_1 & y_2 & \cdots & y_N & \dot{x}_1 & \dot{x}_2 & \cdots & \dot{x}_N & \dot{y}_1 & \dot{y}_2 & \cdots & \dot{y}_N \end{pmatrix}^T \tag{4}$$

Ainsi que la dérivée de u :

$$\dot{u} = \begin{pmatrix} \dot{x}_1 & \dot{x}_2 & \cdots & \dot{x}_N & \dot{y}_1 & \dot{y}_2 & \cdots & \dot{y}_N & \ddot{x}_1 & \ddot{x}_2 & \cdots & \ddot{x}_N & \ddot{y}_1 & \ddot{y}_2 & \cdots & \ddot{y}_N \end{pmatrix}^T \tag{5}$$

Nous posons donc $\dot{u}[0] = u[2N]$, $\dot{u}[1] = u[2N+1]$, ... , $\dot{u}[2N] = \ddot{x}_1 = \frac{1}{N}\sum_{j=1}^{N}\phi(r_{1j})(u[2N+j-1]-u[2N]) + \frac{1}{N}\sum_{j\neq i}^{N}V'(r_{1j})\frac{u[j-1]-u[0]}{r_{1j}}$ (par les formules de (2)), ... , et ainsi de suite pour les autres coordonnées.

2.3.2 Méthode d'Euler explicite

Afin d'avoir un premier aperçu concernant la modélisation d'un mouvement collectif. Nous avons choisi d'utiliser dans un premier temps la méthode d'Euler explicite. Le choix de cette méthode est simple, c'est une méthode directe puisqu'elle prends en compte un seul pas à la fois et utilises seulement la première dérivée pour estimer la position du point suivant. Elle ne nécessite donc pas de résolution complexe de systèmes d'équations. Néanmoins, cela peut entraîner des erreurs importantes si le pas de temps est grand, car la pente(i.e la dérivée) peut changer considérablement. La première étape consiste à retranscrire les composantes du système différentiel fourni dans l'article "Asymptotic flocking for the three-zone model" publié par Fei Cao, Sebastien Motsch, Alexander Reamy et Ryan Theisen [1]. Nous commençons par écrire l'expression de la force d'accélération, qui n'est autre que la somme des forces d'alignement et d'attraction/répulsion. Une fois l'équation différentielle écrite. Nous pouvons appliquer la méthode du schéma d'Euler explicite. Cette méthode calcule une approximation de la solution au temps t_{n+1} à partir du temps précédent t_n. Cela se fait à partir de la relation suivante :

$$u_{n+1} = u_n + hf(t_n, u_n)$$

Avec u_n l'approximation de la solution au temps t_n et h le pas de temps

Concernant la précision de ce modèle, il est important de souligner qu'il est préférable d'avoir un pas de temps assez faible dans le but de réduire les erreurs d'approximation. Ainsi, la méthode d'Euler est dite de premier ordre, ce qui signifie que l'erreur accumulée pas par pas est proportionnelle au carré du pas de temps. Concernant les données initiales, nous générons aléatoirement une liste contenant les données de position et de vitesse. Les données de position sont générées selon la loi uniforme, et selon la loi normale pour les données de vitesse. Le choix de la distribution normale s'explique par le fait que cette dernière apparaît fréquemment dans la nature, notamment dans certains comportement de mouvement des animaux.

2.3.3 Méthode de Runge-Kutta d'ordre 4 (noté RK4) [2]

Une autre méthode plus complexe mais plus précise que la méthode d'Euler explicite est la méthode de Runge-Kutta 4, qui prends 4 différentes estimations de la pente à différents points entre l'instant t_n et t_{n+1} de la fonction $f(t,u)$. Les valeurs du vecteur u sont ensuite mises à jour à chaque itération en utilisant ces approximations de pente.

Plus précisément, la méthode RK4 calcule une approximation de la solution à l'instant t_{n+1} à partir de la solution à l'instant t_n en utilisant la formule suivante :

$$u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

u_n est l'approximation de la solution à l'instant t_n

$k_1 = hf(t_n, u_n)$

$k_2 = hf(t_n + \frac{h}{2}, u_n + \frac{k_1}{2})$

$k_3 = hf(t_n + \frac{h}{2}, u_n + \frac{k_2}{2})$

$k_4 = hf(t_n + h, u_n + k_3)$

h est le pas d'intégration, c'est-à-dire la distance entre deux instants consécutifs.

La méthode RK4 est une méthode d'ordre 4, ce qui signifie que l'erreur commise à chaque itération est proportionnelle à h^5. Elle est donc plus précise que les méthodes d'ordre inférieur comme la méthode d'Euler explicite.

Nous utiliserons la méthode RK4 pour résoudre des équations différentielles du mouvement collectif dans notre étude. Nous utiliserons de même ***np.random.uniform*** et ***np.random.normal*** pour définir les données initiales de position et de vitesse des individus.

3 Définitions, propriétés importantes et résultats théoriques[1]

Définition 2 (EMG)**.** *Nous définissons l'énergie moyenne du groupe ci-dessous :*

$$\mathscr{E}\left(\{\mathbf{x}_i,\mathbf{v}_i\}_i\right)=\frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{v}_i\|^2+\frac{1}{2N^2}\sum_{i,j,i\neq j}^{N}V\left(\left\|\mathbf{x}_j-\mathbf{x}_i\right\|\right) \tag{6}$$

Nous pouvons interpréter cette valeur comme la somme de l'énergie cinétique et de l'énergie potentielle du système.

Lemme 1. *Soit $\{\mathbf{x}_i,\mathbf{v}_i\}_i$ la solution du système des N-poisson avec le système 1. Alors l'énergie $\mathscr{E}$ de l'équation (6) satisfait :*

$$\frac{d}{dt}\mathscr{E}\left(\{\mathbf{x}_i,\mathbf{v}_i\}_i\right)=-\frac{1}{2N^2}\sum_{i,j=1}^{N}\phi_{ij}\|\mathbf{v}_j-\mathbf{v}_i\|^2.$$

Puisque ϕ est une fonction positive, l'énergie $\mathscr{E}$ est toujours décroissante.

Démonstration. En prenant la dérivée temporelle de l'énergie, nous obtenons :

$$\begin{aligned}\frac{d}{dt}\mathscr{E}\left(\{\mathbf{x}_i,\mathbf{v}_i\}_i\right)=&\frac{1}{N}\sum_{i=1}^{N}\dot{\mathbf{v}}_i\cdot\mathbf{v}_i+\frac{1}{2N^2}\sum_{i,j,i\neq j}^{N}\nabla_{\mathbf{x}_i}V\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\cdot\left(\mathbf{v}_j-\mathbf{v}_i\right)\\=&\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j,j\neq i}^{N}\left(\nabla_{\mathbf{x}_i}V\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\cdot\mathbf{v}_i+\phi_{ij}\left(\mathbf{v}_j-\mathbf{v}_i\right)\cdot\mathbf{v}_i\right)\\&+\frac{1}{2N^2}\sum_{i,j,i\neq j}^{N}\nabla_{\mathbf{x}_i}V\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\cdot\left(\mathbf{v}_j-\mathbf{v}_i\right).\end{aligned}$$

Grâce à un argument de symétrie, nous trouvons :

$$\begin{aligned}\sum_{i,j,i\neq j}^{N}\nabla_{\mathbf{x}_i}V\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\cdot\mathbf{v}_i&=\sum_{i,j,i\neq j}^{N}V'\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\frac{\mathbf{x}_j-\mathbf{x}_i}{\|\mathbf{x}_j-\mathbf{x}_i\|}\cdot\mathbf{v}_i\\&=-\sum_{i,j,i\neq j}^{N}V'\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\frac{\mathbf{x}_j-\mathbf{x}_i}{\|\mathbf{x}_j-\mathbf{x}_i\|}\cdot\mathbf{v}_j\\&=\frac{1}{2}\sum_{i,j,i\neq j}^{N}V'\left(\|\mathbf{x}_j-\mathbf{x}_i\|\right)\frac{\mathbf{x}_j-\mathbf{x}_i}{\|\mathbf{x}_j-\mathbf{x}_i\|}\cdot\left(\mathbf{v}_i-\mathbf{v}_j\right).\end{aligned}$$

Par conséquent, nous pouvons simplifier :

$$\frac{d}{dt}\mathscr{E}\left(\{\mathbf{x}_i,\mathbf{v}_i\}_i\right)=\frac{1}{N^2}\sum_{i\neq j}\phi_{ij}\left(\mathbf{v}_j-\mathbf{v}_i\right)\cdot\mathbf{v}_i.$$

En utilisant maintenant la symétrie $\phi_{ij}=\phi_{ji}$, nous concluons :

$$\frac{d}{dt}\mathscr{E}=\frac{1}{2N^2}\sum_{i,j=1}^{N}\phi_{ij}\left(\mathbf{v}_j-\mathbf{v}_i\right)\cdot\left(\mathbf{v}_i-\mathbf{v}_j\right)=-\frac{1}{2N^2}\sum_{i,j=1}^{N}\phi_{ij}\|\mathbf{v}_j-\mathbf{v}_i\|^2.$$

□

Comme l'énergie $\mathscr{E}$ est décroissante, Nous déduisons que l'énergie potentielle est bornée.

Lemme 2. *Soit $\{\mathbf{x}_i, \mathbf{v}_i\}_i$ une solution de 3-z avec le système 1. Il existe une constante C telle que pour tout temps $t \geq 0$:*

$$\sum_{i,j,i\neq j}^{N} V\left(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|\right) \leq C \tag{7}$$

Démonstration. Posons $C_0 = \mathscr{E}\left(\{\mathbf{x}_i(0), \mathbf{v}_i(0)\}_i\right)$. Puisque $\mathscr{E}$ décroît le long de la trajectoire de la solution, nous déduisons que pour tout t :

$$\frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{v}_i(t)\|^2 + \frac{1}{2N^2}\sum_{i,j,ineqj}^{N} V\left(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|\right) \leq C_0,$$

Étant donné que l'énergie cinétique $\frac{1}{2}\sum_{i=1}^{N}\|\mathbf{v}_i\|^2$ est toujours positive, nous en déduisons :

$$\sum_{i,j,i\neq j}^{N} V(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|) \leq 2C_0N^2.$$

En prenant $C = 2C_0N^2$, nous obtenons le résultat. □

Lemme 3. *Si $\lim_{r\to\infty} V(r) = +\infty$. Pour tous $i, j \in \mathbb{N}^*$ et $t \geqslant 0$, il existe un r_M tel que :*

$$\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\| \leqslant r_M \tag{8}$$

Démonstration. D'après le Lemme (2), nous savons que l'énergie potentielle est bornée, en particulier :

$$V\left(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|\right) \leq C,$$

Puisque V satisfait à l'équation (2.6), nous déduisons qu'il existe un r_M tel que :

$$V(r) > C \quad \text{si} \quad r > r_M.$$

Étant donné que $V\left(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|\right)$ est bornée par C, nous concluons que $\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|$ est borné par r_M. □

Théorème 1 (V diverge, minorée et ϕ strictement positive et bornée)**.** *Supposons que V satisfasse l'équation $\lim_{r\to\infty} V(r) = +\infty$ et qu'elle soit minorée. Supposons également que ϕ soit strictement positive et bornée. De plus, assumons que ϕ' et de V' soient également bornées. Alors, le modèle 3-Z converge vers une formation en flocking.*

Démonstration. En utilisant le Lemme (3), nous savons que la distance entre les agents reste bornée : $\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\| \leq r_M$. Puisque r_M est fini, nous pouvons prendre le minimum de ϕ sur cet intervalle :

$$m = \min_{s\in[0,r_M]} \phi(s).$$

Puisque ϕ est strictement positive, nous déduisons que $m > 0$. Ainsi,

$$\phi_{ij} = \phi\left(\|\mathbf{x}_j(t) - \mathbf{x}_i(t)\|\right) \geq m > 0.$$

Puisque l'énergie $\mathscr{E}(\{\mathbf{x}_i,\mathbf{v}_i\}_i)$ est décroissante et minorée, nous déduisons que $\frac{d}{dt}\mathscr{E}(\{\mathbf{x}_i,\mathbf{v}_i\}_i) \xrightarrow{t\to\infty} 0$ (si $\frac{d}{dt}\mathscr{E}$ est uniformément continue). Par conséquent, utilisant le Lemme (1),

$$\phi_{ij}\|\mathbf{v}_j-\mathbf{v}_i\|^2 \xrightarrow{t\to+\infty} 0.$$

Puisque $\phi_{ij}\geq m>0$, nous concluons que $\|\mathbf{v}_j-\mathbf{v}_i\| \xrightarrow{t\to\infty} 0$. De plus, la vitesse moyenne, $\overline{\mathbf{v}}=\frac{1}{N}\sum_i \mathbf{v}_i$, est préservée par la dynamique (par symétrie), donc

$$\mathbf{v}_i(t) \xrightarrow{t\to\infty} \overline{\mathbf{v}} \quad \text{pour tout } i$$

ce qui conclut la preuve. Pour terminer la preuve, il suffit d'établir la continuité uniforme de $\frac{d}{dt}\mathscr{E}$, qui est garantie si des bornes uniformes dans le temps sur la seconde dérivée temporelle de $\mathscr{E}$ peuvent être obtenues. En effet, nous calculons

$$\begin{aligned}\frac{d^2}{dt^2}\mathscr{E}(\{\mathbf{x}_i,\mathbf{v}_i\}_i) &= -\frac{1}{2N^2}\sum_{i,j=1}^N \phi'(\|\mathbf{x}_j(t)-\mathbf{x}_i(t)\|)\|\mathbf{v}_j-\mathbf{v}_i\|^2\frac{\mathbf{x}_j-\mathbf{x}_i}{\|\mathbf{x}_j-\mathbf{x}_i\|}\cdot(\mathbf{v}_j-\mathbf{v}_i)\\ &\quad -\frac{1}{N^2}\sum_{i,j=1}^N \phi(\|\mathbf{x}_j(t)-\mathbf{x}_i(t)\|)(\mathbf{v}_j-\mathbf{v}_i)\cdot(\dot{\mathbf{v}}_j-\dot{\mathbf{v}}_i) := \mathrm{I}+\mathrm{II},\end{aligned}$$

où II est donné par

$$\begin{aligned}\mathrm{II} = -\frac{1}{N^2}\sum_{i,j=1}^N \phi_{ij}(\mathbf{v}_j-\mathbf{v}_i)\cdot\Bigg(&\frac{1}{N}\sum_{k=1}^N\left[\phi_{kj}(\mathbf{v}_k-\mathbf{v}_j)-\phi_{ki}(\mathbf{v}_k-\mathbf{v}_i)\right]\\ &+\frac{1}{N}\sum_{k\neq j}V'(\|\mathbf{x}_k-\mathbf{x}_j\|)\frac{\mathbf{x}_k-\mathbf{x}_j}{\|\mathbf{x}_k-\mathbf{x}_j\|}-\frac{1}{N}\sum_{k\neq i}V'(\|\mathbf{x}_k-\mathbf{x}_i\|)\frac{\mathbf{x}_k-\mathbf{x}_i}{\|\mathbf{x}_k-\mathbf{x}_i\|}\Bigg)\end{aligned}$$

Dans la suite, nous désignerons par C une constante indépendante du temps dont la valeur peut varier d'une ligne à l'autre (et peut dépendre de N). En combinant la bornitude de ϕ et ϕ' avec le fait que $\frac{1}{2N}\sum_{i=1}^N\|\mathbf{v}_i(t)\|^2\leq C$ uniformément dans le temps, on peut facilement obtenir $\|\mathrm{II}\|\leq C$, où cette borne sur $\|\mathrm{II}\|$ est indépendante de N et de t. Pour le terme plus délicat I, nous avons

$$\|\mathrm{I}\| \leq \frac{C}{N^2}\sum_{i,j=1}^N\|\mathbf{v}_i(t)-\mathbf{v}_j(t)\|^3 \leq \frac{C}{N}\sum_{i=1}^N\|\mathbf{v}_i(t)\|^3.$$

Ainsi, la preuve sera complète une fois que nous aurons montré que $\frac{1}{N}\sum_{i=1}^N\|\mathbf{v}_i(t)\|^3\leq C$. Puis, nous calculons

$$\begin{aligned}\frac{1}{N}\frac{d}{dt}\sum_{i=1}^N\|\mathbf{v}_i\|^3 &= \frac{1}{N}\sum_{i,j=1}^N\phi_{ij}(\mathbf{v}_j-\mathbf{v}_i)\cdot\mathbf{v}_i\|\mathbf{v}_i\| + \frac{1}{N}\sum_{i=1}^N\sum_{j\neq i}V'(\|\mathbf{x}_j-\mathbf{x}_i\|)\frac{\mathbf{x}_j-\mathbf{x}_i}{\|\mathbf{x}_j-\mathbf{x}_i\|}\cdot\mathbf{v}_i\|\mathbf{v}_i\|\\ &:= A+B.\end{aligned}$$

Nous avons

$$A\leq\frac{C}{N}\sum_{j=1}^N\|\mathbf{v}_j\|\cdot\sum_{i=1}^N\|\mathbf{v}_i\|^2 - C\sum_{i=1}^N\|\mathbf{v}_i\|^3 \leq CN - C\sum_{i=1}^N\|\mathbf{v}_i\|^3$$

et

$$B\leq\frac{C}{N}\sum_{i,j=1}^N\|\mathbf{v}_i\|^2\leq CN.$$

Par conséquent, nous déduisons que

$$\frac{1}{N}\frac{d}{dt}\sum_{i=1}^{N}\|\mathbf{v}_i\|^3 \leq C - \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{v}_i\|^3, \quad \forall t \geq 0, \forall N,$$

d'où nous obtenons la borne uniforme $\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{v}_i(t)\|^3 \leq C$ comme souhaité. Ceci termine la preuve.

□

4 Analyse numérique des résultats

Selon la théorème 1, nous avons réussi à établir deux ensembles de fonctions ϕ et V, où pour le cas 1 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(\ln(r) - 1)$, et pour le cas 2 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r^2\ln(r)$ (Regardez la Figure 2). Ces deux ensembles de fonctions satisfont les conditions du théorème, donc les résultats obtenus convergent assurément vers un *flocking*.

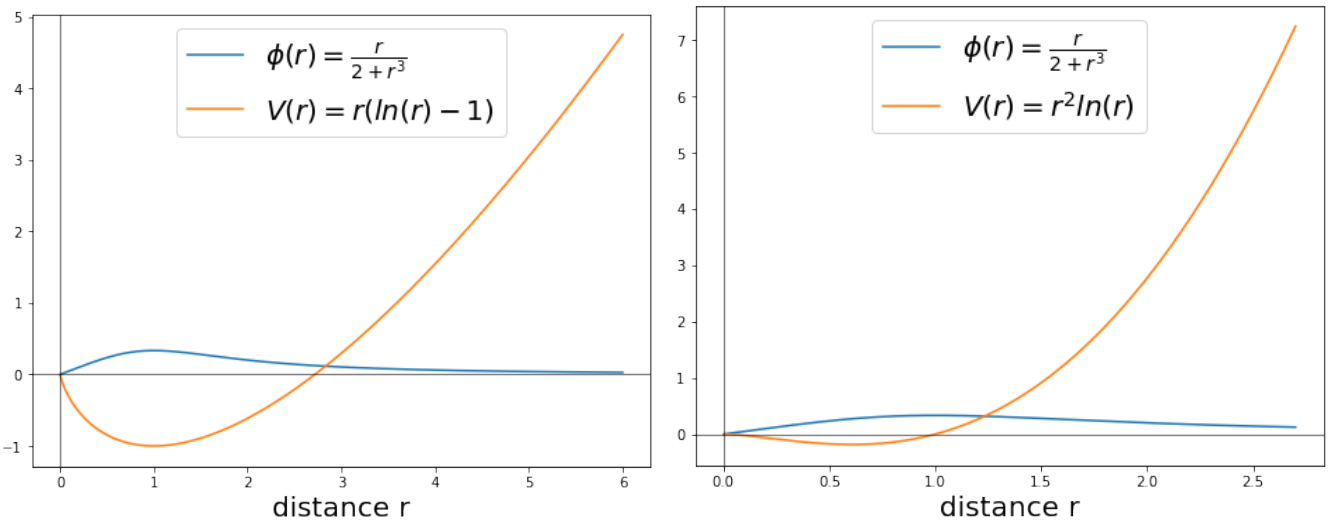

FIGURE 2 – Attraction-répulsion V et alignement ϕ utilisés pour les simulations du cas 1 et cas 2 (À gauche : l'image du Cas 1 ; À droite : l'image du Cas 2)

Durant toute cette partie, nous considérerons la durée totale de l'exécution est de 200, et le pas de temps est de 0,05. De plus, pour les conditions initiales de sorte que chaque individu se trouve à une position aléatoire dans un cube centré à l'origine, nous choisissons les limites de chaque dimension de -0,5 à 0,5 (soit $-0,5 \leq x \leq 0,5$, $-0,5 \leq y \leq 0,5$ et $-0,5$ et si on considère l'espace $\mathbb{R}^3$, nous rajouterons une troisième dimension par $-0,5 \leq z \leq 0,5$). La vitesse de chaque individu suit une distribution gaussienne $\mathcal{N}(0,5, 0,1)$, ce qui signifie que la moyenne des amplitudes de vitesse est de 0,5 et l'écart-type est de 0,1, avec des orientations aléatoires dans l'espace $\mathbb{R}^3$. Par ailleurs, dans toute la partie qui suit, nous utiliserons entièrement la méthode RK4 pour simuler ce modèle.

4.1 Cas 1 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(ln(r) - 1)$

4.1.1 Analyse des simulations dans l'espace $\mathbb{R}^2$

En regardant la Figure 3, ces images illustrent le processus par lequel les trajectoires et les orientations des agents s'alignent progressivement au fil du temps pour passer d'un état désordonné a un état ordonné. Afin de mieux visualiser l'émergence du regroupement, nous avons ajusté le nombre d'étapes k. Concernant les trajectoires des individus à la première étape, nous observons des motifs similaires pour les différents cas, N = 20 ; 50 et 200. Les

agents sont dispersés aléatoirement et leurs trajectoires montrent qu'ils commencent à interagir entre eux. On ne distingue aucun motif clair de mouvement collectif. Ensuite, pour le panel de droite,

1. pour N = 20, les agents semblent êtres alignés et pointent tous dans une même direction. On distingue la formation d'un motif de *flocking*.
2. pour N = 50, tous les agents sont alignés et forment un motif circulaire, ce qui indique la formation d'un mouvement collectif.
3. pour N = 200, tous les individus sont parfaitement alignés et forment une masse circulaire plus dense que pour les cas précédents. On distingue clairement la formation d'un *flocking*.

Selon les cas, les individus interagissent avec un nombre plus élevé d'autres agents. Le nombre d'interactions est alors plus important. Ceci conduit donc a une plus grande densité globale du flocking. Dès lors, l'écart mesuré entre les individus diminue lorsque l'on augmente le nombre N d'agents.

Nous avons réussi à simuler numériquement les résultats Figure 3 qui incluent à la fois l'état initial et l'état final. Sous les contraintes du modèle 3-Z, le mouvement des individus passe d'un état instable à ce qu'on pourrait considérer un *flocking* . Nous observons que la forme finale se rapproche d'un cercle où les individus sont répartis uniformément et où les directions de mouvement sont alignées.

4.1.2 Analyse des résultats dans $\mathbb{R}^3$

Pour nous rapprocher davantage de la réalité, nous avons également étudié le modèle 3-Z dans l'espace $\mathbb{R}^3$, où nous avons étudié le cas 1 avec les variables ϕ et V dans le but de mieux comprendre ce phénomène d'un autre point de vue, ce qui permettra peut être de trouver d'autres problématiques et observations intéressantes.

Expliquons alors les résultats finaux en regardant la Figure 4. Pour mieux se visualiser le *flocking*, on représente les trajectoires finales comme des "têtards", la tête indiquant la direction du mouvement. Pour augmenter la visibilité des trajectoires, nous n'avons pas laissé la représentation directement à la dernière étape, lors de l'apparition du *flock*, car cela aurait rendu les résultats difficiles à traiter, c'est-à-dire un rendu trop petit des têtes qui empêcherait de bien visualiser la direction du mouvement des "têtards". Par conséquent, nous avons introduit une condition selon laquelle pour les dernières k étapes, la longueur de la trajectoire de chaque objet doit être supérieure à 0,05. C'est pourquoi, pour différents N, les trajectoires que nous avons tracées ne se limitent pas à la dernière étape. Ces trois résultats distincts montrent clairement que, sous la condition du cas 1, dans l'espace $\mathbb{R}^3$, les mouvements de groupe finissent par se distribuer uniformément formant une sphère et se dirigeant tous dans la même direction, ce qui est particulièrement évident lorsque $N = 200$. Étant donné la nature tridimensionnelle de cette question, pour rendre les résultats plus visuels, nous avons également créé trois visualisations au format HTML, permettant une observation plus intuitive sous n'importe quel angle. Cliquez sur le lien suivant pour accéder aux visualisations : "cas1: 3-Z dans $\mathbb{R}^3$", vous trouverez un fichier des résultats au format HTML que vous pouvez télécharger ouvrir dans votre navigateur.

4.2 Cas 2 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r^2 ln(r)$

4.2.1 Résultats des simultions dans $\mathbb{R}^2$

En regardant la Figure 5, de la même façon que précédemment, dans le panel de gauche, les agents ne présentent aucune forme reconnaissable, et ne sont pas organisés indépendamment du nombre d'individus. Cependant, dans le panel de droite,

1. pour N = 20, les agents semblent alignés selon la même direction. Et, nous observons qu'ils sont disposés en cercle
2. pour N = 50, les agents sont arrangés de la même façon que pour N = 20.

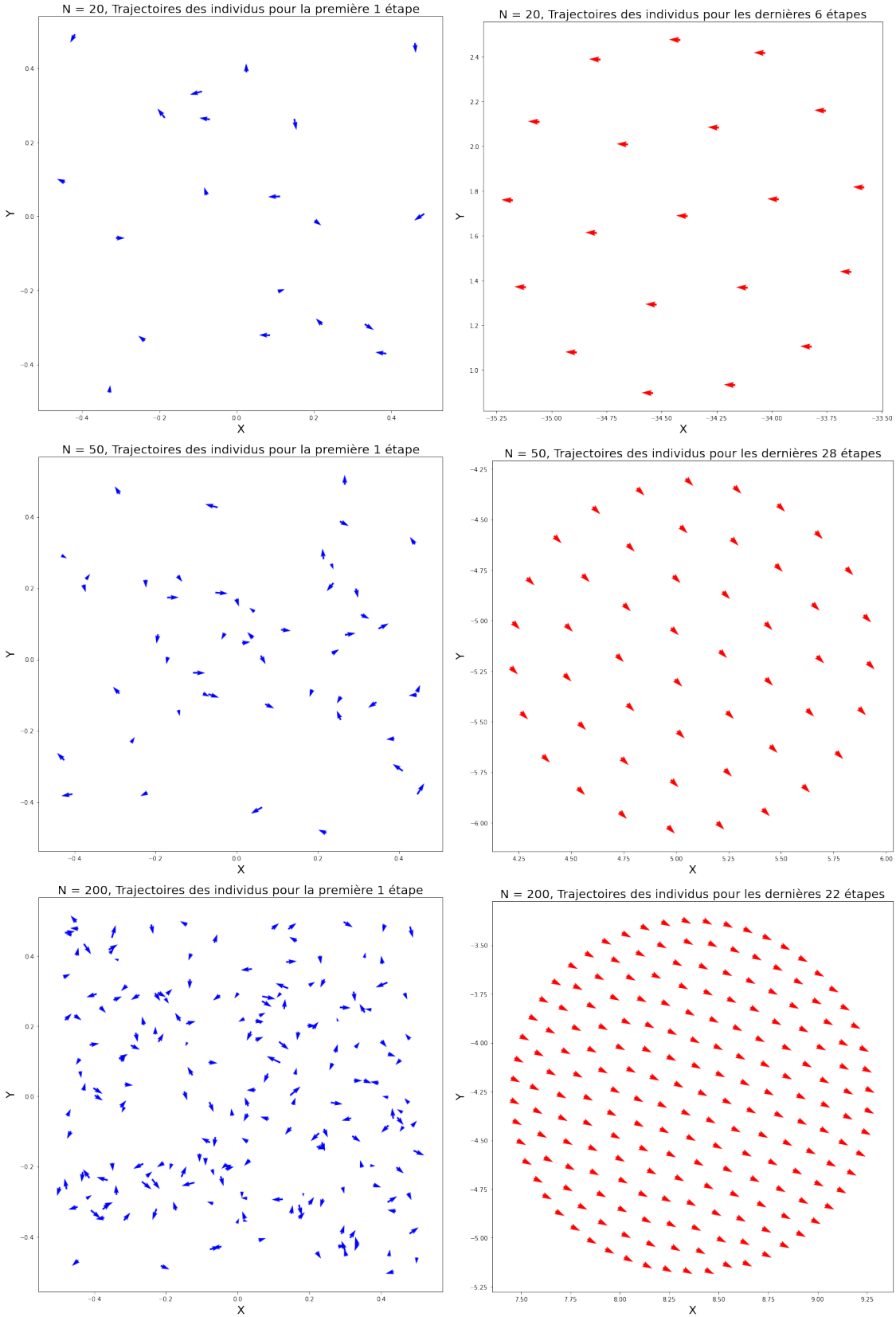

FIGURE 3 – Résultats du cas 1 dans 3-Z : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(\ln(r) - 1)$, avec les positions initiales déterminées par np.random.uniform(low = -0.5, high = 0.5), les amplitudes des vitesses par np.random.normal(loc = 0.5, scale = 0.1) avec des angles aléatoires das $\mathbb{R}^2$, la durée totale de l'exécution est de 200, et le pas de temps est de 0,05.

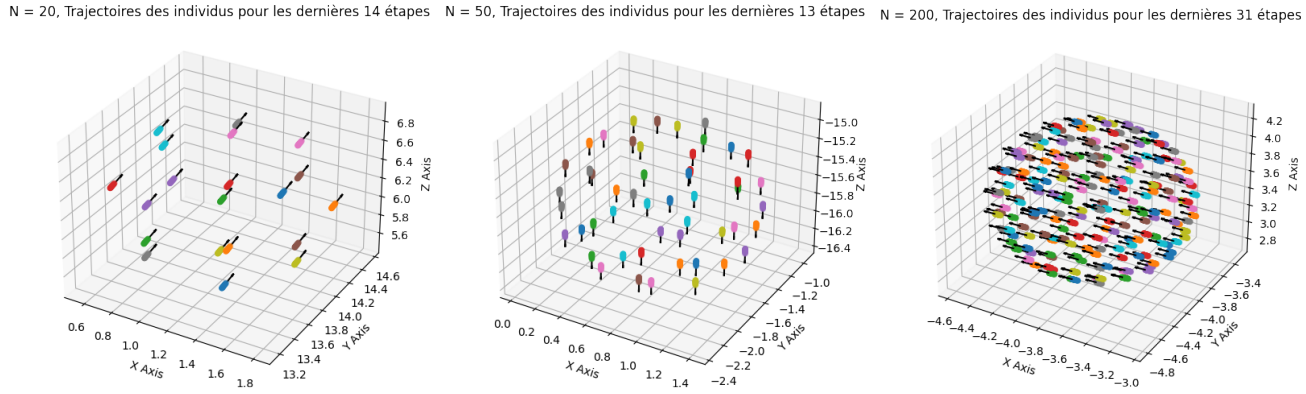

FIGURE 4 – Les résultats finaux en utilisant 3-Z du cas 1 : : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(\ln(r) - 1)$, avec les positions initiales déterminées par np.random.uniform(low = -0.5, high = 0.5), les amplitudes des vitesses par np.random.normal(loc = 0.5, scale = 0.1) avec des angles aléatoires dans $\mathbb{R}^3$, la durée totale de l'exécution est de 200, et le pas de temps est de 0,05.

3. Pour N = 200, les agents forment un cercle bien plus dense que précédemment, et, ils sont aussi alignés.

Au travers des 3 états nous constatons que les individus passent d'un modèle désordonné à stable. Nous pouvons constater qu'ils forment tous les trois une structure annulaire dans laquelle ils maintiennent une distance uniforme et s'alignent le long d'une direction.

4.2.2 Analyse des résultats dans $\mathbb{R}^3$

Avec les mêmes conditions initiales, les trajectoires, ressemblant à de petits têtards, représentent les chemins empruntés par les objets et leur direction de mouvement (vers la tête). Pour la même raison, afin de rendre les résultats plus intuitifs, nous n'avons pas strictement limité la représentation aux trajectoires de la dernière étape, mais nous avons choisi trois différentes étapes k, assurant que chaque trajectoire dépasse 0,05. D'après les résultats simulés sous les conditions du cas 2 dans le modèle 2D (Regardez la Figure 5), nous pouvons conjecturer que dans une situation 3D, le groupe devrait se distribuer uniformément sur une sphère et se déplacer dans la même direction. Bien que les résultats ne soient pas très intuitifs en regardant la Figure 6, nous pouvons supposer initialement que le groupe est uniformément concentré à l'intérieur d'une sphère. Pour vérifier davantage si le groupe est uniformément distribué sur la surface de la sphère, nous pouvons télécharger les résultats de visualisation au format HTML et observer les changements sous différents angles de vue. (Cliquez "cas2 : 3-Z dans $\mathbb{R}^3$" pour télécharger un fichier des résultats et ouvrir dans votre navigateur). Finalement, nous constatons que le groupe est parfaitement et uniformément distribué sur la surface de la sphère, et non à l'intérieur de toute la sphère.

4.3 Résultats de EMG du cas 1 et du cas 2

Selon le lemme 1, nous savons que l'EMG est nécessairement une fonction décroissante. Nous pouvons observer dans la Figure 7 que les graphiques de l'EMG pour différentes valeurs de N sont tous décroissants. Cette observation a été en outre renforcée par une analyse numérique, consolidant ainsi le lemme 1.

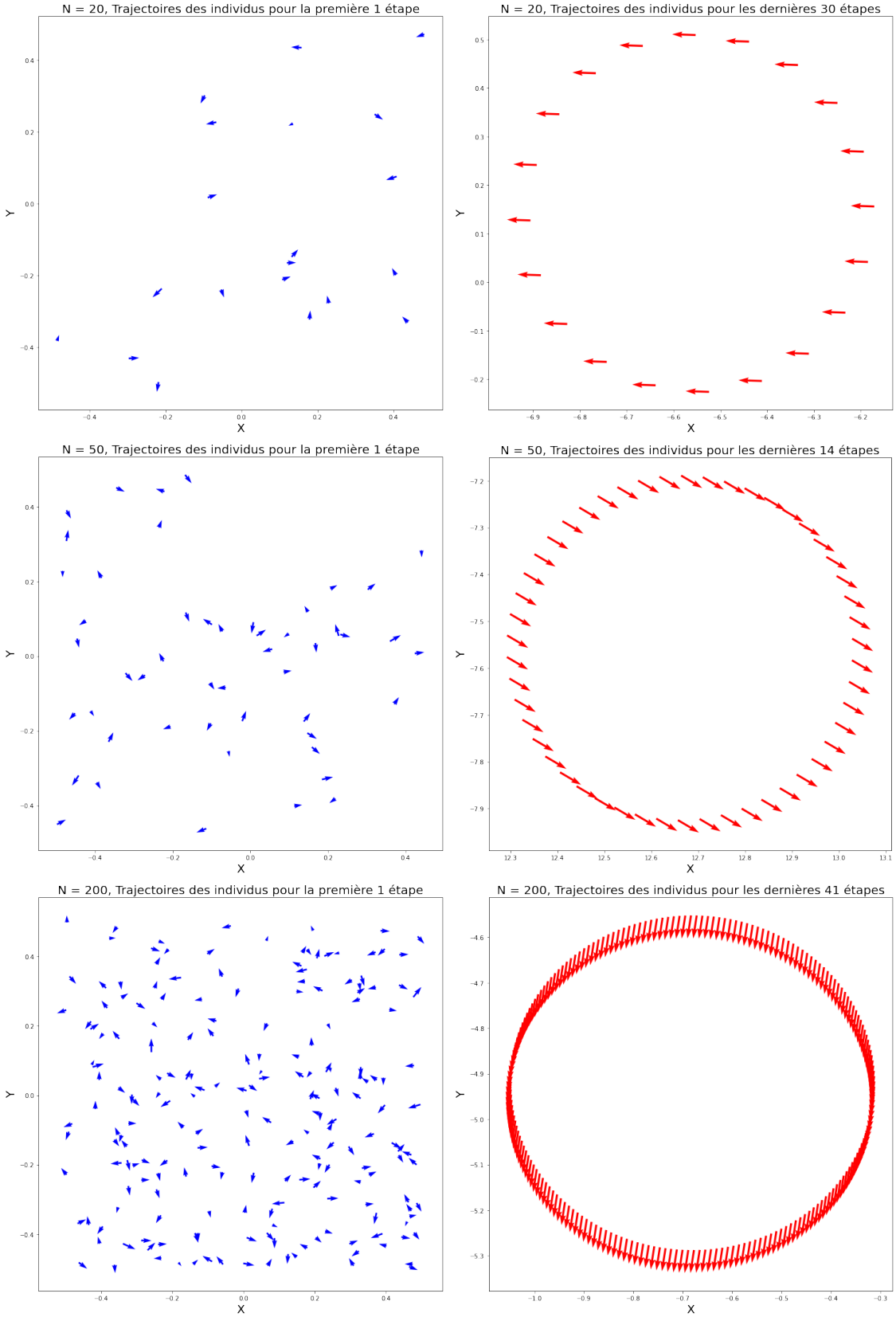

FIGURE 5 – Résultats du cas 2 dans 3-Z : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r^2 ln(r)$, avec les positions initiales déterminées par random.uniform(low = -0.5, high = 0.5), les amplitudes des vitesses par np.random.normal(loc = 0.5, scale = 0.1) avec des angles aléatoires dans $\mathbb{R}^2$, la durée totale de l'exécution est de 200, et le pas de temps est de 0,05.

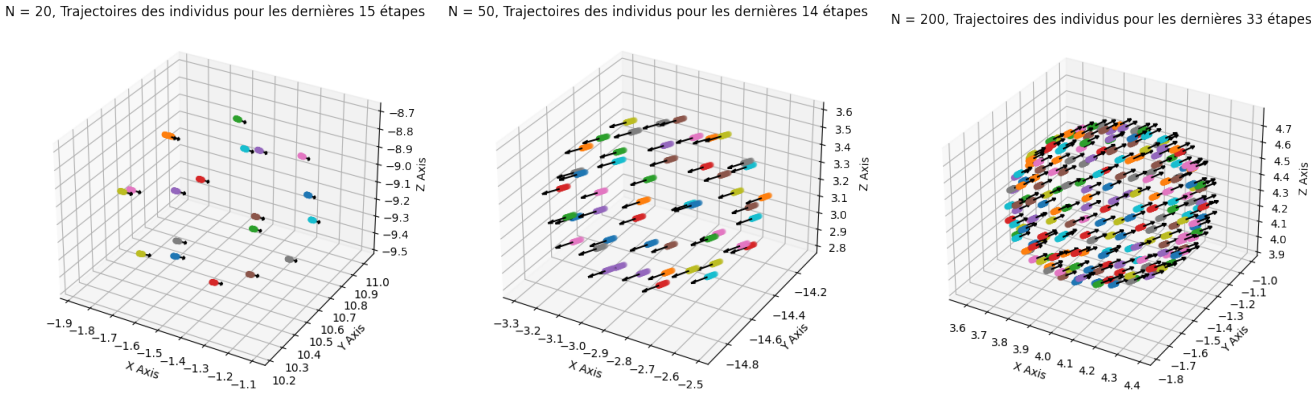

FIGURE 6 – Les résultats finaux en utilisant 3-Z du cas 2 : $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r^2 ln(r)$, avec les positions initiales déterminées par random.uniform(low = -0.5, high = 0.5), les amplitudes des vitesses par np.random.normal(loc = 0.5, scale = 0.1) avec des angles aléatoires dans $\mathbb{R}^3$, la durée totale de l'exécution est de 200, et le pas de temps est de 0,05.

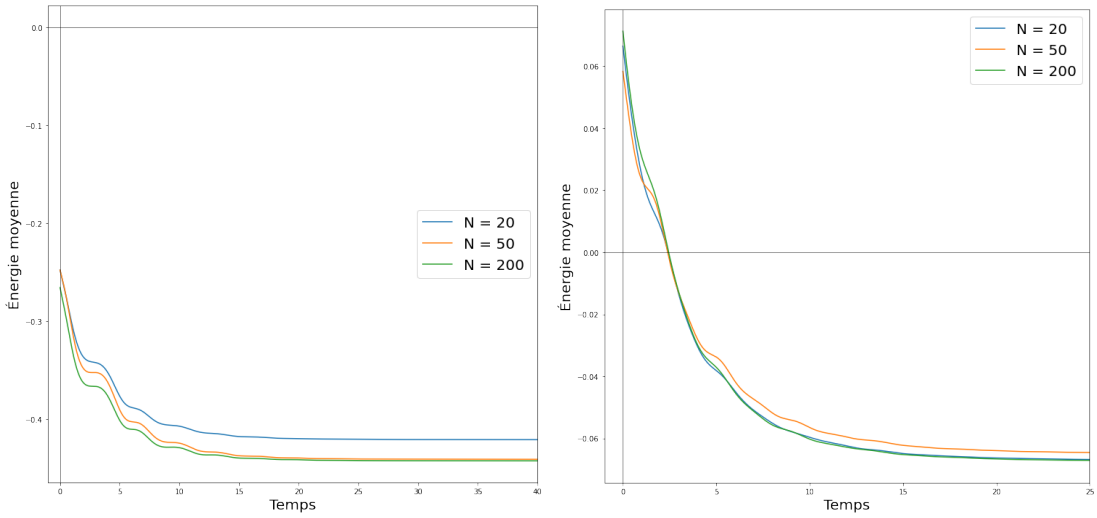

FIGURE 7 – Représentation graphiques de l'énergie moyenne en fonction du temps (À gauche : la figure du cas 1 ; À droite : la figure du cas 2).

5 Ajout M prédateurs

5.1 Modèle d'ajout de M prédateurs (noté 3-Z-P)

Nous supposons N proies (« preys » en anglais) et M prédateurs. Si au temps t, le prédateur p est à la position $\mathbf{w}_p(t)$, avec la vitesse $\mathbf{s}_p(t)$ et $\mathbf{s}_p^\perp$ désignant respectivement la vitesse ainsi que la vitesse perpendiculaire à $\mathbf{s}_p$. Nous remarquons que dans l'espace $\mathbb{R}^2$, il y a deux directions de vitesse perpendiculaire à $\mathbf{s}_p$ nous devons alors fixer l'une des deux pour $\mathbf{s}_p^\perp$. Dans ce modèle, $\mathbf{x}_i$ est la position de la proie i et $\mathbf{v}_i$ est la vitesse de la proie i, pour i et $j \in [\![1,N]\!]$, $\mathbf{w}_p$ est la position du prédateur p et $\mathbf{s}_p$ est la vitesse du prédateur p pour $p \in [\![1,M]\!]$, V est une fonction de potentiel influençant l'attraction et la répulsion entre les proies, et $\overline{V}$ est une fonction de potentiel influençant la répulsion entre les proies et les prédateurs pour les proies donc toujours décroissante, A est une fonction de potentiel influençant l'attraction entre les proies et les prédateurs pour les prédateurs donc toujours croissante.

Avec l'ajout de prédateurs et des proies, nous devons non seulement prendre en compte les interactions au sein des proies, mais aussi l'influence des prédateurs sur les proies (y compris la force d'alignement et la force de répulsion). En effet, lorsque les proies sont poursuivies par les prédateurs, ils tendent à se déplacer dans une direction perpendiculaire à la vitesse du prédateur pour échapper plus efficacement à ce dernier [3]. Cependant, pour les prédateurs, nous considérons simplement la force d'attraction qu'ils ont envers leurs proies. Dès lors, nous pouvons obtenir un nouvel modèle (un peu plus complexe) :

$$\begin{cases} \dot{\mathbf{x}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= \dfrac{1}{N}\displaystyle\sum_{j=1}^{N} \phi_{ij}(\mathbf{v}_j - \mathbf{v}_i) + \frac{1}{N}\sum_{j\neq i}^{N} -\nabla_{\mathbf{x}_{ji}} V(\|\mathbf{x}_j - \mathbf{x}_i\|) \\ &\quad + \dfrac{1}{M}\displaystyle\sum_{p=1}^{M} \psi_{pi}(\mathbf{s}_p^\perp - \mathbf{v}_i)\mathbb{1}(\mathbf{v}_i \cdot \mathbf{s}_p^\perp > 0) + \frac{1}{M}\sum_{p=1}^{M} -\nabla_{\mathbf{x}_{pi}} \overline{V}(\|\mathbf{w}_p - \mathbf{x}_i\|) \\ \dot{\mathbf{w}}_p &= \mathbf{s}_p \\ \dot{\mathbf{s}}_p &= -\nabla_{\mathbf{x}_{cp}} A(\|\mathbf{x}_c - \mathbf{w}_p\|) \end{cases} \tag{9}$$

Dans cette formule, $\phi_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$ et $\psi_{ip} = \psi(\|\mathbf{x}_i - \mathbf{w}_p\|)$ sont des fonctions déterminant l'intensité de l'alignement de vitesse affectant le mouvement des proies et l'intensité avec laquelle les prédateurs affectent l'alignement de la vitesse des proies. Par ailleurs, $-\nabla_{\mathbf{x}_{ji}} V(\|\mathbf{x}_j - \mathbf{x}_i\|)$ exprime l'effet d'attraction ou de répulsion exercé par l'individu j sur l'individu i, $-\nabla_{\mathbf{x}_{pi}} \overline{V}(\|\mathbf{w}_p - \mathbf{x}_i\|)$ représente la force de répulsion exercée par le prédateur p sur sa proie i et $-\nabla_{\mathbf{x}_{cp}} A(\|\mathbf{x}_c - \mathbf{w}_p\|)$ représente la force d'attraction exercée par le centre de masse des proies sur le prédateur p. Les formules du gradient seront donc :

$$-\nabla_{\mathbf{x}_{ji}} V(\|\mathbf{x}_j - \mathbf{x}_i\|) = V'(\|\mathbf{x}_j - \mathbf{x}_i\|)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$

$$-\nabla_{\mathbf{x}_{pi}} \overline{V}(\|\mathbf{w}_p - \mathbf{x}_i\|) = \overline{V}'(\|\mathbf{w}_p - \mathbf{x}_i\|)\frac{\mathbf{w}_p - \mathbf{x}_i}{\|\mathbf{w}_p - \mathbf{x}_i\|}$$

$$-\nabla_{\mathbf{x}_{cp}} A(\|\mathbf{x}_c - \mathbf{w}_p\|) = A'(\|\mathbf{x}_c - \mathbf{w}_p\|)\frac{\mathbf{x}_c - \mathbf{w}_p}{\|\mathbf{x}_c - \mathbf{w}_p\|}$$

5.2 Illustrations des résultats en utilisant 3-Z-P

En regardant la Figure 8, nous définissons les fonctions ϕ et V comme dans le cas 1 (c'est-à-dire $\phi(x) = \frac{r}{2+r^3}$ et $V(r) = r(ln(r) - 1)$). Nous définissons également $\psi(r) = \frac{2r}{2+r^{10}}$, où $\psi(r)$ représente la tendance du mouvement

des proies dans la direction perpendiculaire au prédateur. On remarque que plus la dérivée de ψ est grande, plus la tendance à ce que les proies se déplacent perpendiculairement par rapport au prédateur est accentuée. Ainsi, comparée à l'image de ϕ, nous avons configuré une fonction plus "abrupte", car lorsque la proie est de plus en plus proche du prédateur, sa tendance à se déplacer perpendiculairement est plus élevée que la force d'alignement des prédateurs. Par la suite, nous définissons $\overline{V}(r) = -\frac{1}{5}\ln(r)$, qui illustre la répulsion de la proie envers le prédateur, où plus r se rapproche de 0, plus l'effet répulsif est fort, reflété par une dérivée négative importante. C'est ainsi que nous établissons la fonction $\overline{V}$. Finalement, nous avons :

$$A(r) = \begin{cases} \sqrt{r} & \text{si } r \leqslant 2 \\ r^2 + \left(\frac{1}{2\sqrt{2}} - 4\right) r + \frac{2\sqrt{2}}{2} + 4 & \text{sinon} \end{cases}$$

Cette fonction représente l'attraction des proies pour les prédateurs. Comme $\overline{V}$, lorsque r est proche de 0, l'attraction est plus forte, comme en témoigne une dérivée plus grande. Cependant, avec une telle configuration, les prédateurs pourraient s'éloigner drastiquement de la proie si leur vitesse est trop élevée, réduisant ainsi la force d'attraction ressentie par ce dernier. Pour y remédier, nous introduisons un seuil à $r = 2$, de sorte que lorsque $r > 2$, plus les prédateurs s'éloignent, plus l'attraction est grande, et lorsque $r < 2$, plus les prédateurs sont proches, plus l'attraction est forte, tout en garantissant que A est une fonction continue et de classe C^1. Cette configuration a donc été adoptée pour répondre à ces critères.

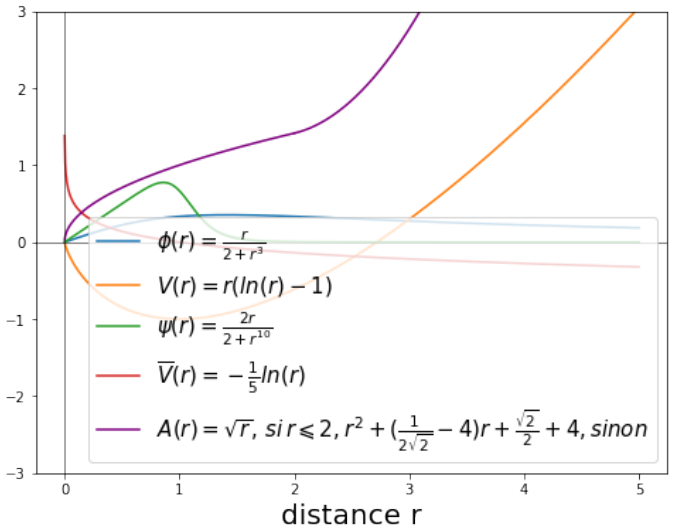

FIGURE 8 – Attraction-répulsion V, $\overline{V}$ et A et alignement ϕ et ψ pour les simulations en utilisant 3-Z-P

Nous définissons les conditions initiales, permettant aux proies de devenir des groupes cohésifs avant d'introduire les prédateurs. Selon les équations de ϕ et de V que nous avons établies, par exemple, dans le **cas 1**, les proies devraient être uniformément distribuées dans un cercle et se déplacer dans la même direction (figure 3). Nous définissons alors les conditions initiales, permettant aux proies d'entrer en phase de regroupement (*flocking*) avant d'introduire les prédateurs pour observer comment cela influencera le *flock*. Selon les équations définies pour ϕ et V, dans le **cas 1**, les proies doivent être uniformément réparties dans un cercle et se déplacer dans la même direction. Veuillez noter que dans certains cas, pour une meilleure représentation dans le GIF fournit, nous réduisons proportionnellement la vitesse initiale des proies par rapport à l'état finale obtenue dans la simulation du **cas 1** précédent. Ensuite, la vitesse des prédateur(s) est dirigée vers le centre de masse du *flocking* des proies configurées, comme leur vitesse est supérieure à celle des proies et leur position initiale est à l'extérieur du *flocking*, (Voir La Figure 3). Nous procéderons ensuite à une analyse numérique selon deux scénarios distincts : le premier cas consiste à se donner cinquante proies

et un seul prédateur, tandis que le deuxième cas sera composé de cinquante proies et trois prédateurs. D'ailleurs, nous considérerons que la durée totale de l'exécution est de 200, et le pas de temps est de 0,05 dans 2 Cas après.

5.2.1 Cas : N = 50, M = 1 en utilisant Méthode d'Euler explicite

Les proies sont représentées en vert et les prédateurs en rouge, tandis que les trajectoires, similaires à de petits têtards, indiquent la direction du mouvement, orientée par la tête. En observant la Figure 9, nous remarquons une intrusion de prédateur au sein d'un ensemble de proies déjà bien ordonnés. Nous pouvons constater que dès que le prédateur s'approche des proies, celles à proximité réagissent rapidement, accélérant vers les côtés, perpendiculairement au prédateur entraînant un effet de chaîne. En effet, à mesure que le prédateur traverse le groupe de proies, ces dernières réussissent à se scinder en deux parties, une stratégie similaire et cohérente avec la réalité. Puis, lorsque le prédateur s'éloigne à nouveau, les deux groupes séparés tendent à se réunir, se déplaçant presque dans la même direction et se rapprochant progressivement jusqu'à fusionner à nouveau en un seul ensemble. Et ainsi de suite, le prédateur, attiré par les proies, se retourne pour envahir à nouveau le groupe. Compte tenu de la dynamique particulière de cette interaction, nous avons créé une visualisation (cliquez ici) au format GIF pour une compréhension plus intuitive et compréhensible du phénomène. On remarque alors que le prédateur effectue un mouvement oscillatoire, envahissant à plusieurs reprises le groupe de proies, puis faisant demi-tour pour entamer une nouvelle percée. De leur côté, les proies se séparent à chaque invasion puis se regroupent de nouveau.

5.2.2 Cas : N = 50, M = 3 en utilisant Méthode d'Euler explicite

En regardant la Figure 10, comme dans le cas précédent, les prédateurs sont représentés en rouge et les proies en vert, tandis que les trajectoires, semblables à de petits têtards, indiquent la direction du mouvement. Lorsque les proies adoptent un comportement de *flocking*, l'intervention de trois prédateurs rend la situation encore plus intéressante. Par rapport au scénario précédent, l'ajout de deux prédateurs supplémentaires complexifie la dynamique. Nous observons que l'introduction de trois prédateurs divise le groupe de proies en trois parties. Dans ce cas, nous constatons des dynamiques similaires à celles du scénario précédent : les proies se séparent puis se rassemblent de nouveaux, tandis que les prédateurs effectuent une percée, se retournent et ainsi de suite. Or, certaines proies ne parviennent pas à emprunter des trajectoires d'évasion efficaces, ce qui peut être dû à la présence de multiples prédateurs annulant les forces dirigées perpendiculairement aux autres prédateurs, empêchant ainsi les proies de choisir la meilleure voie de fuite, à savoir l'accélération perpendiculaire. On peut voir ceci comme un effet *Sandwich*, les proies qui sont encerclés auront du mal à emprunter la meilleure fuite. Dans ce modèle 3-Z-P, bien que nous n'ayons pas intégré un mécanisme de prédation conduisant à la disparition des proies(c'est-à-dire lorsque les prédateurs mangent les proies qu'ils atteignent), nous pouvons déduire qu'avec plusieurs prédateurs, le taux de succès de la prédation pourrait être accru. Finalement, nous avons également créé une visualisation (cliquez ici) des résultats au format GIF de ce dernier.

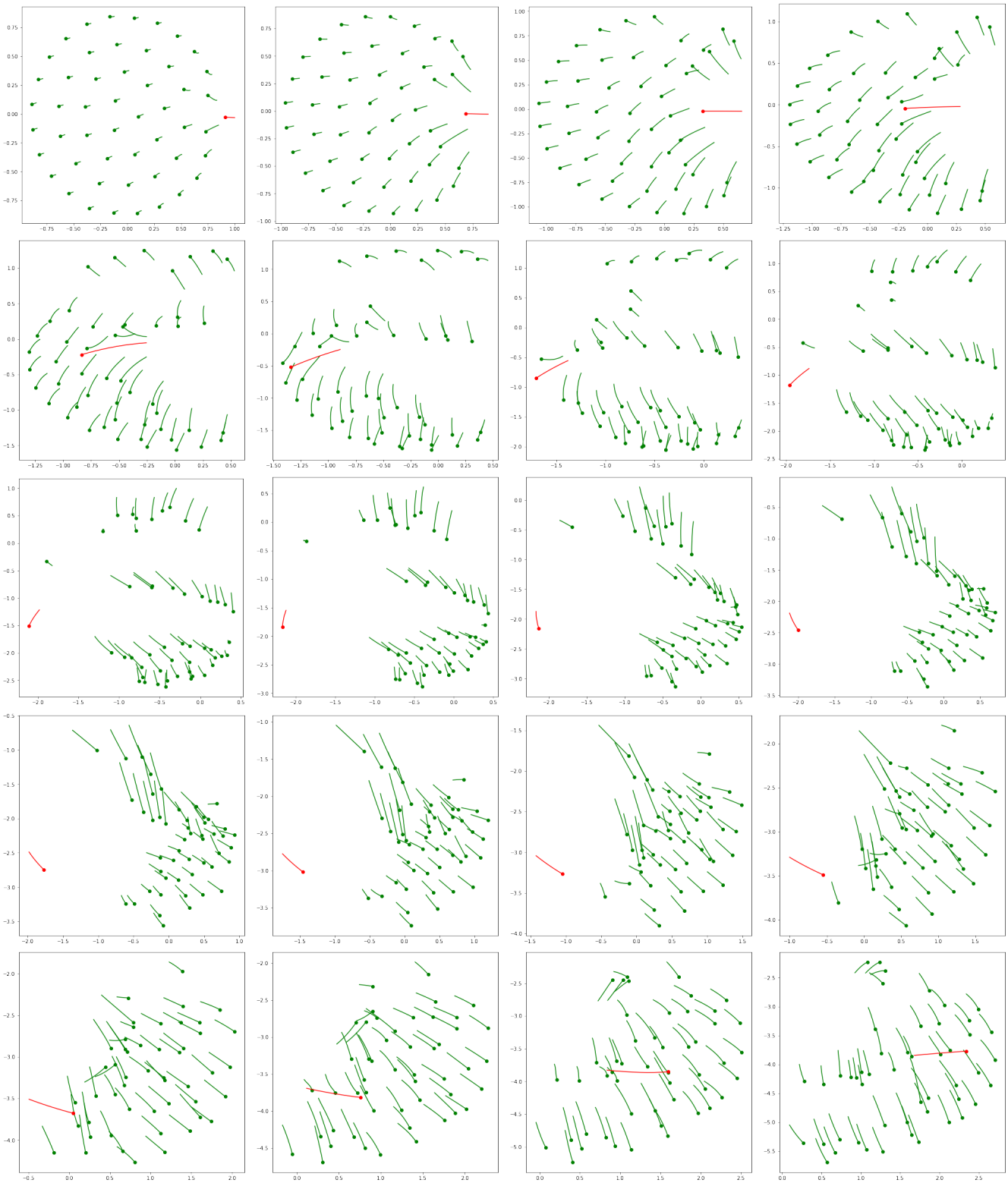

FIGURE 9 – Résultats du Cas N = 50, M = 1 en appliquant les foctions attraction-répulsion ci-dessus (Regardez la Figure 8) et les conditions initiales décrites dans le paragraphe précédent

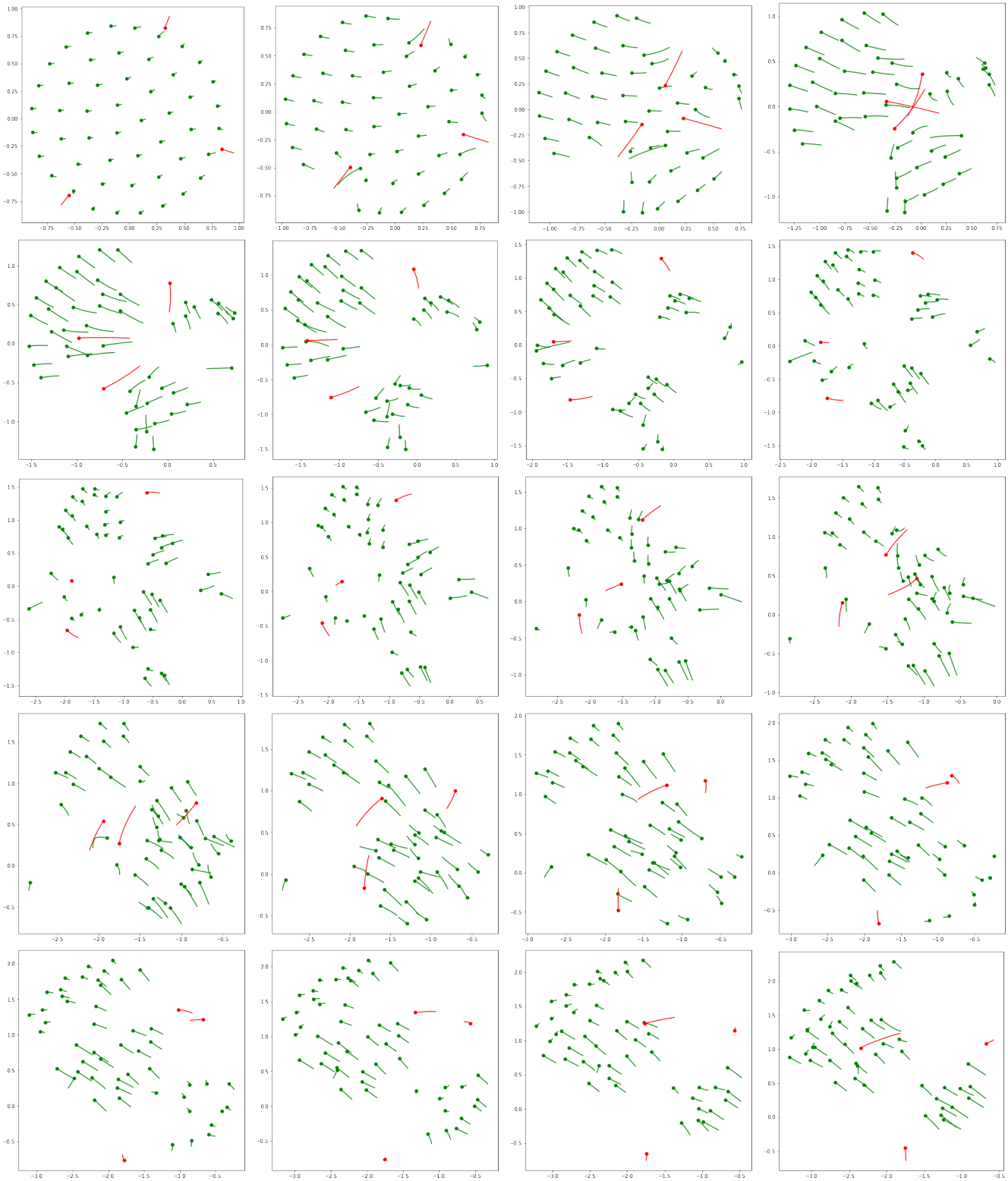

FIGURE 10 – Résultats du Cas N = 50, M = 3 en appliquant les foctions attraction-répulsion ci-dessus (Regardez la Figure 8) et les conditions initiales décrites dans le paragraphe précédent

6 Ajout des conditions de bords

6.1 Ajout d'un cadre fermé dans $\mathbb{R}^2$

Dans un premier temps, nous avons fait le choix d'ajouter une boite rectangulaire fermée dans laquelle les entités restent et ne peuvent pas en sortir. Dans un second temps, nous mettrons en place un mur circulaire. Tout cela dans le but d'observer comment les différentes contraintes physiques influencer le *flocking*. Afin d'incorporer les conditions de bords dans la simulation, nous avons modifié le programme de la méthode d'Euler explicite. L'approche resterait similaire si nous avions choisi d'utiliser la méthode de Runge-Kutta d'ordre 4 (RK4). Pour cela, nos conditions de réflexivité sont intégrées directement dans la fonction calculant les positions selon la méthode d'Euler explicite (ME). Nous commençons par définir le cadre de notre boite fermée en posant les coordonnées xmin, xmax, ymin, ymax. Ensuite, lorsque les positions sont calculées, si un agent se trouve en dehors du cadre. Sa nouvelle position est ajustée en prenant en compte le rebond. Celle-ci est calculée en faisant la différence entre la distance totale s'il n'y avait pas de contraintes et la distance au temps t_n vers l'extrémité. Ce calcul est possible car nous sommes à vitesse constante. Et donc nous pouvons additionner les composantes des forces. Concernant la vitesse, elle est simplement remplacée par son opposée en fonction de l'axe sur lequel se trouve l'agent. Par exemple, si un agent se déplace vers la droite (positivement selon x) et touche le bord droit , alors sa vitesse v_x sera inversée.

6.1.1 Problème du mur carré et analyses

Afin d'illustrer au mieux les rebonds, nous avons affiché deux graphiques issues de la simulation avec les conditions de bords.(Les premières figures dans la Figure 11). Nous avons délibérément choisi de mettre un faible nombre d'agents, N = 10 afin de mettre en évidence les trajectoires. Les paramètres utilisés sont les suivants.

- Limites du cadre
  - xmax = 10
  - ymax = 10
  - -xmin = -10
  - -ymin = -10
- nombre d'itérations n = 2000

représente la formation d'un mouvement collectif dans un boite fermée. A la fin de la simulation, les agents se regroupent dans le coin inférieur droit.

6.1.2 Observations et analyse pour l'ajout d'un mur carré pour le cas 1

Étant donné qu'il est difficile d'observer les directions des organismes, nous réalisons un grossissement de la région ou ils sont concentrés (Regardez la Figure 11). Nous constatons qu'ils suivent tous la même direction et ont un écart qui semble égal. Nous en déduisons donc à première vue qu'il y a bien l'apparition d'un *flocking*.

L'illustration suivante représente l'ensemble des trajectoires effectués par les individus tout au long de la simulation. La course se termine aux alentours des coordonnées (9.5,-7.5). Initialement, les agents se déplacent aléatoirement et ont des difficultés à se regrouper. Ensuite, au fil du temps, nous pouvons observer que lorsqu'ils rebondissent sur une paroi, les acteurs tendent à essayer de se regrouper en haut de la fenêtre Lorsqu'ils rebondissent de nouveau, nous pouvons constater la présence d'oscillations dans une zone autour des coordonnées (5, 10). Ces perturbations diminuent au cours du temps jusqu'à arriver à un état stable.

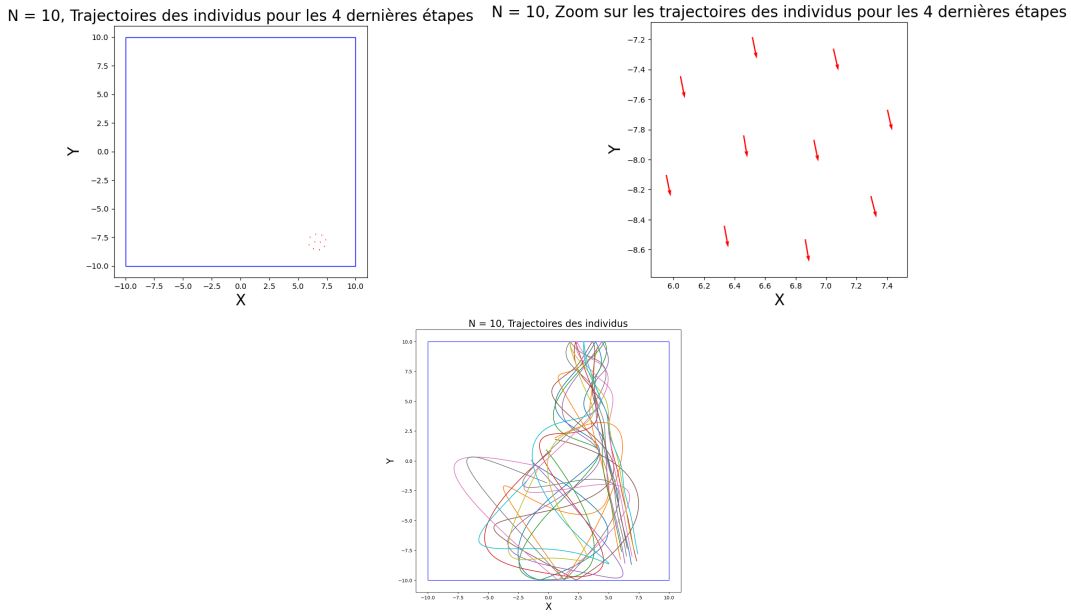

FIGURE 11

6.1.3 Résultats du Cas N =10 ; $\phi(r) = \frac{r}{2+r3}$; $V(r) = r^2 \ln(r)$ avec un mur carré

6.1.4 Observations et analyse pour l'ajout d'un mur carré pour le cas 2

De la même façon que précédemment, nous réalisons la même modélisation en utilisant les paramètres identiques à ceux utilisés précédemment, mis à part la fonction V qui diffère. La première figure illustre le regroupement des individus dans le coin inférieur gauche. Le second graphique représente un agrandissement du premier dans la zone ou se trouvent les entités. Dans ce même graphique, nous pouvons observer que les agents suivent une tendance générale pour leurs directions allant de gauche à droite. Cependant, ils ne sont pas alignés de façon à constituer un flock. La dernière figure illustre les trajectoires effectués par chaque élément tout au long de la simulation. Initialement, ils sont regroupés aléatoirement dans une zone à proximité du centre de la figure. Par la suite, ils se dirigent vers le bord gauche. Lorsqu'un rebond se produit, nous constatons qu'ils sont toujours désorganisés et ne parviennent pas à former un flock. En examinant les trajectoires à la suite du premier rebond, aux coordonnées (10,-6), et ce jusqu'au second en (-5,-8), on observe que les chemins semblent s'entortiller et se croiser entre eux avant de s'aligner, créant un motif torsadé. Par ailleurs, une fois que les particules sont regroupés, les rebonds ne semblent pas affecter la structure générale. La largeur du groupe conserve ses caractéristiques, à savoir ses dimensions qui sont de l'ordre d'une unité de mesure. Enfin, nous pouvons conclure par le fait que suivant la fonction V utilisée, les éléments ne réagissent pas de la même façon lorsqu'ils rebondissent sur un bord. Ceux suivant le premier cas sont plus sujets aux perturbations extérieures. Alors que dans le cas que nous venons d'évoquer ci-dessus, ils semblent êtres moins impactés par les conditions de bords.

6.2 Que se passe-t-il lorsque l'on modifie la taille de la boîte

Plus généralement, nous cherchons à savoir si les dimensions de la boite fermée ont une influence sur la formation d'un flocking. Pour cela, nous testons numériquement en diminuant progressivement les dimensions de la boite. Or, sachant que l'objectif n'est plus d'illustrer les trajectoires, nous augmentons le nombre d'agents afin d'avoir N = 50. Le nombre d'itérations reste inchangé. Les figures représentées ci-dessous illustrent l'alignement des individus pour des boites de dimensions 9x9, 5x5 et 2x2.

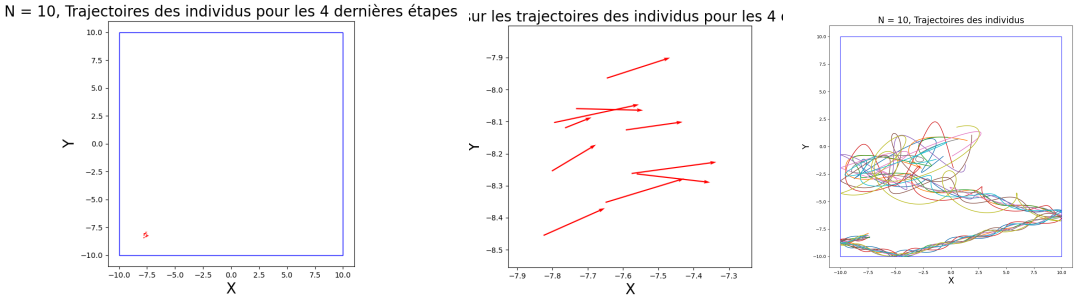

FIGURE 12 – Observations de l'ajout d'un cadre fermé de dimension 10x10 pour N = 10, pour le cas 2

6.2.1 Resultats du Cas N = 50 ; $\phi(r) = \frac{r}{2+r3}$; $V(r) = r(\log(r) - 1)$ avec un mur carré

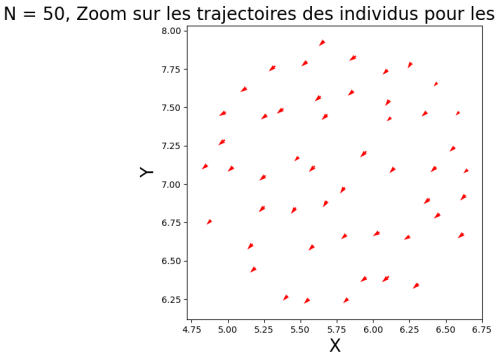

FIGURE 13 – Cas1 : Boite de taille 9x9

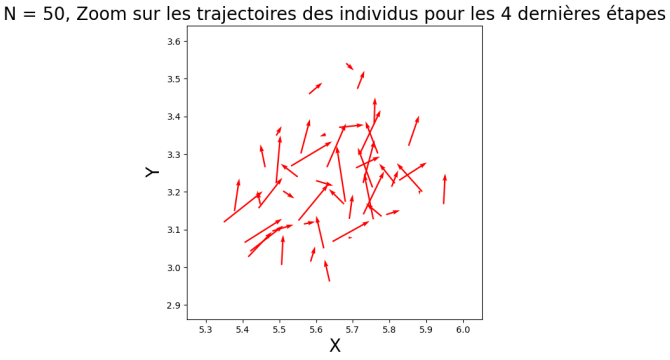

FIGURE 14 – Cas2 : Boite de taille 9x9

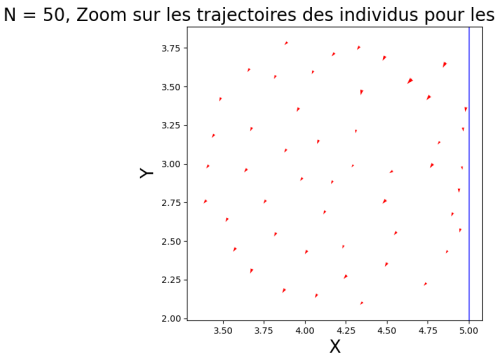

FIGURE 15 – Cas1 : Boite de taille 5x5

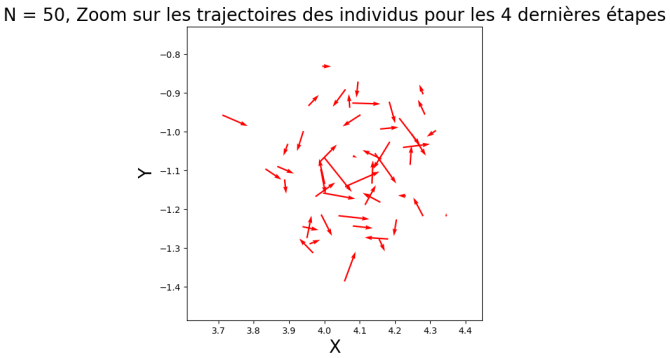

FIGURE 16 – Cas2 : Boite de taille 5x5

6.3 Cas 1

Dans la Figure 12 , les agents sont éparpillés sur un espace relativement large, ils ne semblent pas avoir de direction suffisamment claire. Cela nous amène donc à penser qu'il n'y a pas de comportement de flocking à ce stade. Concernant la boite de taille moyenne, l'espace est plus réduit et donc les agents sont plus concentrés. Nous pouvons constater qu'ils présentent tous un alignement suivant un direction similaire mais leur espacement n'est pas équidistant. Cela laisse penser que si le temps de la simulation est plus long, un mouvement collectif pourrait se former. Enfin, dans la plus petite boite,la Figure 17 , les individus sont forcés de se rapprocher mais, comme à l'illustration précédente, l'alignement n'est pas parfait. Ce qui suggère de nouveau que les agents se trouvent dans

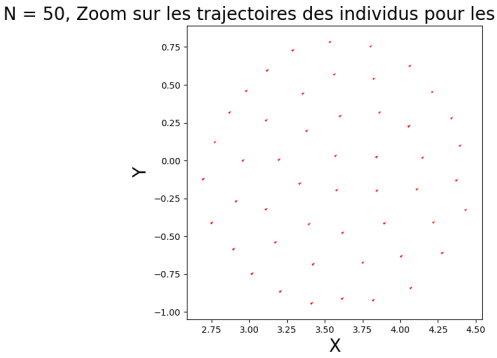

FIGURE 17 – Cas1 : Boite de taille 2x2

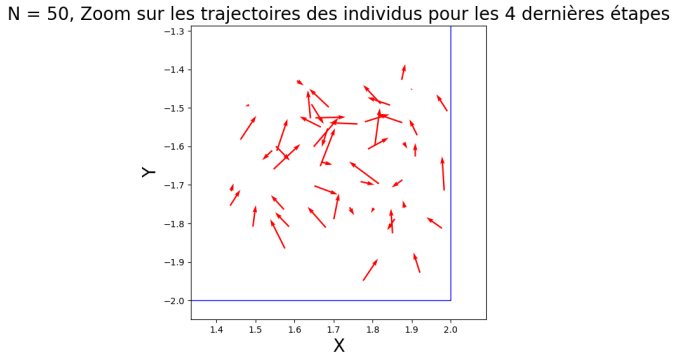

FIGURE 18 – Cas2 : Boite de taille 2x2

un état transitoire. Plus généralement, il est clair que la taille de la boite influence la formation d'un mouvement collectif. Plus l'espace est restreint, plus les agents ont tendance à être plus proche les uns des autres ce qui favorise la formation d'un flocking.

6.4 Cas 2

6.4.1 Analyse des résultats

Contrairement au cas précédent où nous pouvions percevoir une état transitoire vers un mouvement collectif pour les 3 figures. Ici, l'ensemble des illustrations ne nous permettent en aucun cas d'apercevoir un état ordonné. Les agents sont dispersés avec des vecteurs vitesse pointant dans des directions multiples. Par ailleurs, plus la taille de l'ensemble est faible, plus la densité des individus sera élevée. Plus généralement, nous pouvions penser qu'en réduisant la taille de l'espace, l'apparition d'un flocking serait favorisée. Or, non. Cela signifie que la taille de l'espace n'est pas le seul facteur pour favoriser un mouvement ordonné, ce qui laisse penser qu'en augmentant la durée de la simulation, les particules pourront se trouver dans un état stable à partir d'un certain moment.

6.5 Resultats du Cas 1 avec N = 5 ; $\phi(r) = \frac{r}{2+r^3}$; $V(r) = r(\log(r) - 1)$ avec un mur circulaire

Nous allons désormais nous intéresser à l'influence du mur circulaire sur la dynamique de *flocking*. Dans de telles simulations, le comportement collectif des entités est non seulement caractérisé par les interactions entre elles, mais aussi par leurs interactions avec l'environnement extérieur où bien les contraintes physiques associées, comme le mur carré étudié précédemment. Ici, concernant l'ajout d'un mur circulaire, son effet sur la trajectoire des individus est crucial pour comprendre et prédire le comportement du système dans son ensemble. Nous simulons le mouvement autour d'un mur circulaire en utilisant la méthode de Runge-Kutta 4 (RK4) pour résoudre les équations différentielles.

Paramètres de simulation

- Nombre de particules (N) : 5
- Rayon du cercle : 20
- Temps de fin de la simulation : 100
- Pas de temps (Δt) : 0.1

Conditions initiales aléatoires

Les positions et les vitesses des particules sont initialisées de manière aléatoire dans l'intervalle $[-10, 10]$ pour les positions et $[-1, 1]$ pour les vitesses.

Fonctions de calcul

— `distance(x1, y1, x2, y2)` : Calcule la distance euclidienne entre deux points.
— `deriv(u)` : Calcule les dérivées par rapport au temps des positions et des vitesses des particules.
— `RK4_step(u, dt)` : Effectue une étape de la méthode de Runge-Kutta 4 pour mettre à jour les positions et les vitesses des particules.
— `reflect(u, i)` : Gère la réflexion des particules sur le mur circulaire.

Simulation

Nous itérons sur chaque pas de temps en utilisant la méthode RK4 pour mettre à jour les positions et les vitesses des particules. Après chaque mise à jour, nous vérifions si une particule dépasse le mur circulaire. Si c'est le cas, nous appliquons la réflexion.

Affichage des trajectoires

Nous traçons les trajectoires des particules à chaque pas de temps. Le mur circulaire est également tracé pour visualiser les rebonds des particules.

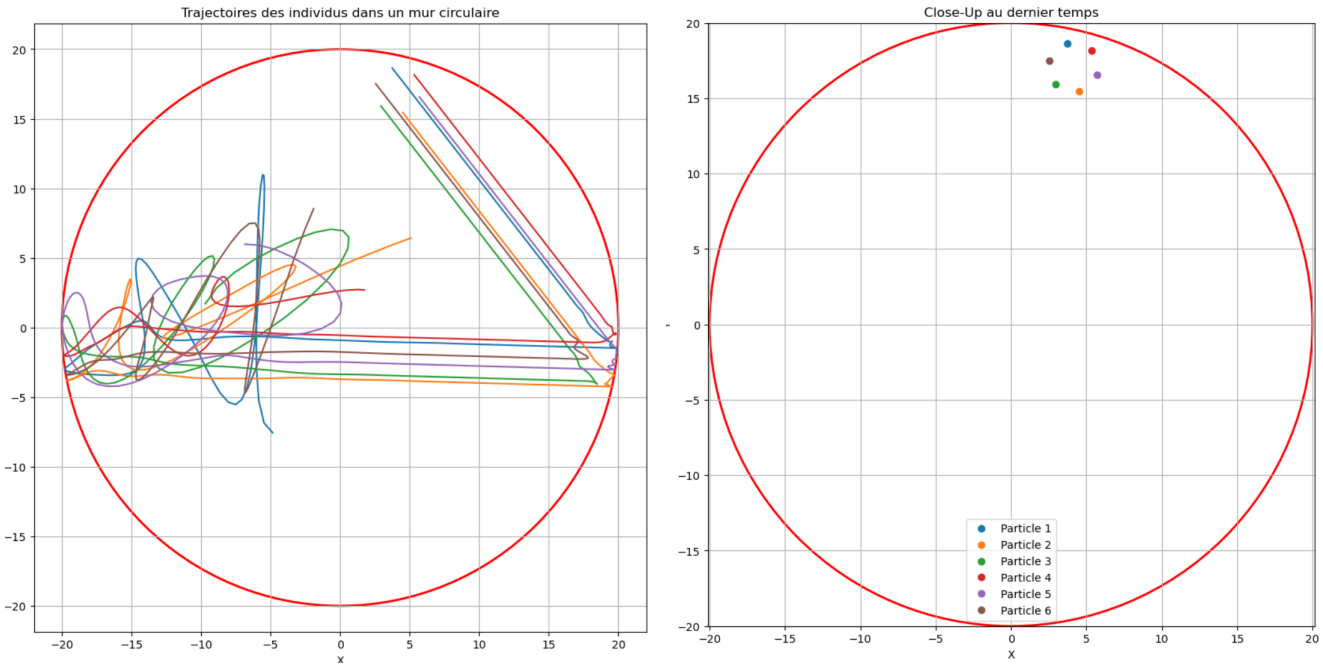

FIGURE 19 – Illustration du *flocking* pour les paramètres suivants : N = 5 avec $\phi(r) = \frac{r}{2+r3}$; $V(r) = r(\log(r) - 1)$; instant : t = 200 et représentation close-up au dernier temps

Figure 19 montre les résultats d'une simulation de dynamique de groupe, ou "flocking", où N = 5 particules interagissent entre elles et avec un mur circulaire. Les trajectoires des particules sont tracées dans un plan 2D et nous pouvons observer leur comportement collectif ainsi que leur interaction avec la limite circulaire externe.

Les fonctions $\phi(r)$ et V(r) décrivent les forces d'interaction entre les particules. La fonction $\phi(r)$ semble être une fonction qui influence l'alignement des particules entre elles, tandis que V(r) est potentiellement liée à la force d'attraction ou de répulsion en fonction de la distance r entre deux particules. La forme spécifique de V(r) suggère qu'elle pourrait modéliser une force attractive à longue distance et répulsive à courte distance, car elle devient négative lorsque r est petit (ce qui correspond à une force répulsive en raison du terme log(r)) et positive à mesure que r augmente.

D'après laFigure 19, le mur circulaire force les individus à changer de direction lorsqu'ils entrent en contact avec lui, ce qui peut mener à des comportements intéressants tels que le regroupement le long du mur, la réflexion coordonnée, ou des changements dans le comportement de flocking. L'analyse de ces interactions nous permet d'explorer comment des bords peuvent définir ou bien modifer les dynamiques de groupe pré-définies dans le but de peut-être contrôler une population et de prédire des mouvements de groupe.

7 Discussions

Néanmoins le modèle 3-Z-P admets des inconvénients. Le modèle de prédation des prédateurs est trop simpliste, en effet, dans la vie réelle, chaque prédateur adopte souvent différentes stratégies de prédation. Par exemple, certains prédateurs peuvent être chargés de diviser le groupe de proies, tandis que d'autres entourent le groupe de proies, augmentant ainsi le taux de réussite des prédateurs. Bien sûr, dans notre modèle, nous n'avons également pas pris en compte les scénarios ou les proies prennent la fuite. Dans certaines situations particulières, lorsque le nombre de proies est nettement supérieur au nombre de prédateurs, le rapport de force bascule entre les deux groupes. Dans des études ultérieures, nous pourrons donc envisager des modèles plus complexes et plus proche de la réalité. Concernant les murs circulaires, le modèle n'est pas encore réaliste puisque les individus ne touchent pas réellement le mur. Il est donc important de relever le fait que ce n'est qu'une simulation qui pourra être davantage développer dans la suite de l'étude.

8 Conclusion

Finalement, ce rapport aura exploré en détail la dynamique complexe du *flocking*. Au travers de modèles et méthodes numériques, notamment l'utilisation de la méthode d'Euler explicite et de la méthode de Runge-Kutta d'ordre 4. Nous avons pu observer comment les interactions simples entre les agents peuvent aboutir à des comportements collectifs émergents dans un espace.

L'analyse numérique des résultats a révélé des différences significatives dans le comportement du *flocking* selon la nature des forces d'interaction entre les individus, modélisées par les fonctions $\phi(r)$ et V(r), et la forme des conditions aux limites.

L'ajout de prédateurs a introduit une nouvelle couche de complexité, démontrant comment la présence de menaces externes peut altérer la dynamique de groupe et mener à de nouvelles stratégies d'évitement et d'adaptation. Ces observations soulignent l'importance d'une compréhension approfondie des principes sous-jacents au *flocking* pour la conception de systèmes autonomes et l'étude de la dynamique des populations.

L'introduction de murs carrés et circulaires a mis en lumière l'influence des conditions de bord sur le *flocking*, montrant des changements notables dans les trajectoires et les configurations de groupe.

En résumé, ce rapport met en exergue la richesse des systèmes dynamiques auto-organisés et l'impact des conditions environnementales sur leur comportement. Il ouvre également la voie à de futures recherches sur l'interaction entre agents autonomes et leur environnement, avec des implications potentielles dans des domaines aussi variés que

la robotique, la biologie ou bien la gestion des foules.

Références

[1] Fei CAO et al. “Asymptotic flocking for the three-zone model”. en. In : Mathematical Biosciences and Engineering 17.6 (2020), p. 7692-7707. ISSN : 1551-0018. DOI : 10.3934/mbe.2020391. URL : http://www.aimspress.com/rticle/doi/10.3934/mbe.2020391 (visité le 13/02/2024).

[2] Liviu Gr. IXARU et Guido VANDEN BERGHE. “Runge-Kutta Solvers for Ordinary Differential Equations”. en. In : Exponential Fitting. Sous la dir. de Liviu Gr. IXARU et Guido VANDEN BERGHE. Mathematics and Its Applications. Dordrecht : Springer Netherlands, 2004, p. 223-304. ISBN : 9781402021008. DOI : 10.1007/978-1-4020-2100-8_6. URL : https://doi.org/10.1007/978-1-4020-2100-8_6 (visité le 17/02/2024).

[3] Alan WILSON et al. “Biomechanics of predator–prey arms race in lion, zebra, cheetah and impala”. In : Nature (fév. 2018). DOI : 10.1038/nature25479.

9 Annexes

Dans cette partie, nous allons montrer le code de chaque partie du projet. Ces paquets ci-dessous sont les bases utilisées par ce code.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import os
import plotly.graph_objects as go
```

9.1 3-Z dans l'espace $\mathbb{R}^2$

9.1.1 Fonction dérivée

Face au problème de Cauchy $\dot{u} = f(u,t)$, où u peut être un vecteur ou une matrice. Dans le modèle 3-Z, nous définissons la fonction f (*deriv* ci-dessous) ainsi par le système (2).

```
def d(xi,yi,xj,yj):
    return np.sqrt((xi-xj)**2+(yi-yj)**2)
def deriv(t,u):
    du = np.empty(u.size)
    # Dérivée de la vitesse
    for i in range(N):
        du[i] = u[2*N+i]
        du[N+i] = u[3*N+i]
        a = 0
        b = 0
        for j in range(N):
            rij = d(u[i],u[N+i],u[j],u[N+j])
            if i != j:
                a += 1/N*phi(rij)*(u[2*N+j]-u[2*N+i])+1/N*dV(rij)*(u[j] - u[i])/(rij)
                b += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])+1/N*dV(rij)*(u[N+j] - u[N+i])/(rij)
            else:
                a += 1/N*phi(rij)*(u[2*N+j]-u[2*N+i])
                b += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])
        du[2*N+i] = a
        du[3*N+i] = b
    return du
```

9.1.2 Méthodes numériques : RK4+Euler explicite

Voilà le code de RK4 et Euler explicite. Dans le code suivant, nous utiliserons RK4.

```
def rk4(x, dx, y, deriv):
    """
       /*----------------------------------------
       sous programme de resolution d'equations
       differentielles du premier ordre par
       la methode de Runge-Kutta d'ordre 4
       x = abscisse, une valeur scalaire, par exemple le temps
       dx = pas, par exemple le pas de temps
       y = valeurs des fonctions au temps t(i), c'est un tableau numpy de taille n
       avec n le nombre d'équations différentielles du 1er ordre

       rk4 renvoie les nouvelles valeurs de y pour t(i+1)

       deriv = variable contenant le nom du
       sous-programme qui calcule les derivees
       deriv doit avoir deux arguments: deriv(x,y) et renvoyer
       un tableau numpy dy de taille n
       ----------------------------------------*/
     """
    #  /* d1, d2, d3, d4 = estimations des derivees
    #    yp = estimations intermediaires des fonctions */
    ddx = dx/2.       #         /* demi-pas */
    d1 = deriv(x,y)   #       /* 1ere estimation */
    yp = y + d1*ddx
    #    for  i in range(n):
    #         yp[i] = y[i] + d1[i]*ddx
    d2 = deriv(x+ddx,yp)     #/* 2eme estimat. (1/2 pas) */
    yp = y + d2*ddx
    d3 = deriv(x+ddx,yp)  #/* 3eme estimat. (1/2 pas) */
    yp = y + d3*dx
    d4 = deriv(x+dx,yp)     #  /* 4eme estimat. (1 pas) */
    #/* estimation de y pour le pas suivant en utilisant
    #  une moyenne ponderee des derivees en remarquant
    #  que : 1/6 + 1/3 + 1/3 + 1/6 = 1 */
    return y + dx*( d1 + 2*d2 + 2*d3 + d4 )/6
```

```
def ME(x, dx, y, deriv): #Méthode d'Euler explicite
    return y+dx*deriv(x,y)
```

9.1.3 Programme de la Boucle (noté PB)

Après avoir bien défini la fonction dérivée et choisi une méthode numérique appropriée, nous présenterons la partie cruciale de la résolution du problème de Cauchy, à savoir le code pour obtenir le résultat final de l'équation.

```
def integrationEDO(start,end,step, v_ini, deriv):
    n = v_ini.size
    # nombre d'equa-diff du 1er ordre

    # Création du tableau temps
    interval = end - start                    # Intervalle
    num_points = int(interval / step) + 1     # Nombre d'éléments
    t = np.linspace(start, end, num_points)   # Tableau temps t

    # Initialisation du tableau v
    v = np.empty((n, num_points))
    # tableau y qui contient les valeurs actuelles de v au temps t[i]
    y = np.empty(n)

    # Condition initiale
    v[:, 0] = v_ini
    y[:] = v_ini # important de mettre [:] pour y,
                 # sinon y serait juste un alias de v_ini et non une copie
                 # ce qui ferait qu'on modifie v_ini quand on modifie y et
                 # donc aussi le u_ini donné à la fonction lors de son appel.

    # Boucle four
    for i in range(num_points - 1):
        y = rk4(t[i], step, y, deriv) # modifie le tableau y
        v[:, i + 1] = y

    # Argument de sortie
    return t, v
def resultat(tmax, dt, u_ini):
    # Méthode d'Euler
    t, u = integrationEDO(0, tmax, dt, u_ini, deriv)
```

```
32      return u
```

9.1.4 Code des graphes (Définitions)

Nous alors définissons conditions initiales.

```
#Conditions initiales
X0 = np.random.uniform(low = -2, high = 2, size = 2*N)
V0 = np.random.normal(loc = 4, scale = 1 , size = N)
theta0 = np.random.uniform(low=0, high=2*np.pi, size=N) # liste de thetas aléatoires
Vx = V0*np.cos(theta0)
Vy = V0*np.sin(theta0)
u_ini = np.concatenate((X0,Vx,Vy))
```

Voilà le code des trajectoires globaux

```
def traj_liste(N):
    return resultat(tf,dt,u_ini)
def draw(N):
    fig = plt.figure(figsize = (12, 12)) #la taille de la figure
    plt.xlabel("X",fontsize = 20)
    plt.ylabel("Y",fontsize = 20)
    for i in range(N):
        plt.plot(u[i,:],u[N+i,:],"-")
    plt.title(f"N = {N}, les trajectoires des individus",fontsize=20)
    plt.show()
```

Voilà le code des trajectoires intiniales.

```
def intial(N, k):
    fig = plt.figure(figsize=(12, 12))
    plt.xlabel("X", fontsize=20)
    plt.ylabel("Y", fontsize=20)

    for i in range(N):
        x = u[i, :][:k]
        y = u[N+i, :][:k]

        for j in range(k-1):
            dx = x[j+1] - x[j]
```

```
            dy = y[j+1] - y[j]
            plt.plot(x,y,color='none')
            plt.quiver(x[j], y[j], dx, dy, angles="xy", scale_units="xy",
                       scale=1, color = "blue", width=0.004)

    plt.title(f"N = {N}, Trajectoires des individus pour la première {k-1} étape",fontsize=20)
    plt.show()
```

Voilà des trajectoires finales.

```
def final(N):
    fig = plt.figure(figsize=(12, 12))
    plt.xlabel("X", fontsize=20)
    plt.ylabel("Y", fontsize=20)

    k = 2

    for i in range(N):
        k_temp = k
        while True:
            x = u[i, :][-k_temp:]
            y = u[N+i, :][-k_temp:]
            total_length = sum(np.sqrt(np.diff(x)**2 + np.diff(y)**2))

            if total_length > 0.05 or k_temp >= len(u[i, :]):
                break
            k_temp += 1

        k = max(k, k_temp)

    for i in range(N):
        x = u[i, :][-k:]
        y = u[N+i, :][-k:]

        dx = x[-1] - x[0]
        dy = y[-1] - y[0]
        plt.plot(x,y,color='none')
        plt.quiver(x[0], y[0], dx, dy, angles="xy", scale_units="xy", scale=1,
                   color="red", width=0.005, headwidth=3, headlength=5)

```

```
    plt.title(f"N = {N}, Trajectoires des individus pour les dernières {k-1} étapes",fontsize=20)
    plt.show()

```

Voilà le code de EMG et leurs trajectoires.

```
def E(n,liste_u,liste_N):
    arr = np.empty((3, n))
    for k in range(len(liste_u)):
        u = liste_u[k]
        N = liste_N[k]
        liste_energie = []
        for t in range(n):
            res = 0
            for i in range(N):
                res += 1/(2*N)*(u[2*N+i,:][t]**2+u[3*N+i,:][t]**2)
                for j in range(N):
                    rij = d(u[i,:][t],u[N+i,:][t],u[j,:][t],u[N+j,:][t])
                    if i != j:
                        res += 1/(2*N**2)*V(rij)
                    else:
                        res += 0
            liste_energie.append(res)
        arr[k, :] =liste_energie
    return arr
def draw_E(liste_E,xlim):
    fig = plt.figure(figsize = (12, 12)) #la taille de la figure
    plt.xlim(-1,xlim)
    plt.xlabel("Temps",fontsize = 20)
    plt.ylabel("Énergie moyenne",fontsize = 20)
    for i in range(len(liste_E)):
        plt.plot(T,liste_E[i,:],label = f"N = {liste_N[i]}")
    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)
    plt.legend(fontsize=20)
    plt.show()
```

9.1.5 Applications numériques

Dans cette partie, nous allons introduire des valeurs concrètes pour obtenir le résultat final de manière intuitive.

```
ti = 0            # Temps initial
tf = 200          # Temps final
n = 4000         # Nombre d'itérations temporelles
dt = (tf-ti)/n               # Pas temporel
T = np.linspace(ti, tf, n)    # Liste des temps
liste_N = [20,50,200]
```

Nous définissons ϕ, V et V' du Cas 1 ou Cas 2, et tracer leur graphes.

```
# Cas 1
def phi(r):
    return r/(2+r**3)

def V(r):
    return r*(np.log(r)-1)

def dV(r):
    return np.log(r)

"""
# Cas 2
def phi(r):
    return r/(2+r**3)

def V(r):
    return r**2*np.log(r)

def dV(r):
    return 2*r*np.log(r)+r
"""


r = np.linspace(0.001, 6, 1000)
fig = plt.figure(figsize=(8, 6))
plt.xlabel("distance r", fontsize=20)
plt.plot(r,phi(r),label = r"$\phi(r) = \frac{r}{2+r^3}$ ")
plt.plot(r,V(r),label = r"$V(r) = r(ln(r)-1)$")
plt.plot(r,dV(r),label = r"$V'(r) = ln(r)$")
plt.axhline(0, color='black',linewidth=0.5)
```

```
plt.axvline(0, color='black',linewidth=0.5)
plt.legend(fontsize=20)
plt.show()
```

Nous n'affichons que le code de cas en $N = 20$. Pour $N=50$, $N = 200$, nous ne changeons que le paramètre de N.

```
"""N = 20
k = 2
u = traj_liste(N)

array_data = u
file_path = "~/N_20_1.npy"
os.makedirs(os.path.dirname(file_path), exist_ok=True)
np.save(file_path, array_data)"""
liste_u = []
u = np.load('~/N_20_1.npy')
liste_u.append(u)
N = 20
k = 2
draw(N)
intial(N,k)
final(N)
```

EMG et les trajectoires

```
"""array_data = E(n,liste_u,liste_N)
#liste_u est la liste des résultats en N = 20,50 et 200
#liste_N = [20,50,200]
file_path = "~/Cas1_listes_energie.npy"
os.makedirs(os.path.dirname(file_path), exist_ok=True)
np.save(file_path, array_data)"""
liste_E = np.load('~/Cas1_listes_energie.npy')
xlim = 40
draw_E(liste_E,xlim)
```

9.2 3-Z dans l'espace $\mathbb{R}^3$ (visualisation)

9.2.1 Fonction dérivée

Dans cette partie, nous redéfinissons la fonction

```
def d(xi,yi,zi,xj,yj,zj): # la distance entre 2 individus dans R^3
    return np.sqrt((xi-xj)**2+(yi-yj)**2+(zi-zj)**2)
def deriv(t,u):
    du = np.empty(u.size)

    # Dérivée de la vitesse
    for i in range(N):
        du[i] = u[3*N+i]
        du[N+i] = u[4*N+i]
        du[2*N+i] = u[5*N+i]
        a = 0
        b = 0
        c = 0
        for j in range(N):
            rij = d(u[i],u[N+i],u[2*N+i],u[j],u[N+j],u[2*N+j])
            if i != j:
                a += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])+1/N*dV(rij)*(u[j] - u[i])/(rij)
                b += 1/N*phi(rij)*(u[4*N+j]-u[4*N+i])+1/N*dV(rij)*(u[N+j] - u[N+i])/(rij)
                c += 1/N*phi(rij)*(u[5*N+j]-u[5*N+i])+1/N*dV(rij)*(u[2*N+j] - u[2*N+i])/(rij)
            else:
                a += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])
                b += 1/N*phi(rij)*(u[4*N+j]-u[4*N+i])
                c += 1/N*phi(rij)*(u[5*N+j]-u[5*N+i])
        du[3*N+i] = a
        du[4*N+i] = b
        du[5*N+i] = c
    return du
```

9.2.2 Code des graphes (Définitions)

Ce code est similaire à celui de la section précédente (3-Z dans $\mathbb{R}^2$), mais cette fois, nous présentons uniquement le code pour afficher les résultats finaux en format html.

```
def final_3D(N, u):
    fig = go.Figure()
    colors = [f'rgba({r}, {g}, {b}, 0.8)' for r, g, b in np.random.randint(0, 255, (N, 3))]
    k = 2
    for i in range(N):
        k_temp = k
```

```
        while True:
            x = u[i, :][-k_temp:]
            y = u[N+i, :][-k_temp:]
            z = u[2*N+i, :][-k_temp:]
            total_length = sum(np.sqrt(np.diff(x)**2 + np.diff(y)**2 + np.diff(z)**2))
            if total_length > 0.05 or k_temp >= len(u[i, :]):
                break
            k_temp += 1
        k = max(k, k_temp)
    for i in range(N):
        x = u[i, :][-k:]
        y = u[N+i, :][-k:]
        z = u[2*N+i, :][-k:]
        dx = x[-1] - x[0]
        dy = y[-1] - y[0]
        dz = z[-1] - z[0]
        fig.add_trace(go.Scatter3d(x=x, y=y, z=z, mode='lines',
                                   line=dict(color=colors[i], width=4),name=f'individu {i+1}'))
        fig.add_trace(go.Cone(x=[x[-1]], y=[y[-1]], z=[z[-1]], u=[dx], v=[dy], w=[dz],
                              colorscale=[[0, colors[i]], [1, colors[i]]],
                              showscale=False,sizemode="absolute", sizeref=0.1))

    fig.update_layout(title=f"N = {N}, Trajectoires des individus pour les dernières {k-1} étapes",
                      scene=dict(xaxis_title='X Axe', yaxis_title='Y Axe', zaxis_title='Z Axe'),
                      scene_aspectmode='auto',width=800, height=600)

    fig.show()
    fig.write_html(f"3D_trajectories_{N}.html")
```

9.2.3 Applications numérique

Similaire aux conditions initiales établies dans la section précédente (3-z dans $\mathbb{R}^2$), cette section ajoute uniquement des conditions initiales supplémentaires sur l'axe des y, incluant la position et la vitesse.

```
#Conditions initiales
X0 = np.random.uniform(low=-2, high=2, size=3*N)
V0 = np.random.normal(loc=4, scale=1, size=N)
theta0 = np.random.uniform(low=0, high=np.pi, size=N)
phi0 = np.random.uniform(low=0, high=2*np.pi, size=N)
Vx = V0 * np.sin(theta0) * np.cos(phi0)
```

```
Vy = V0 * np.sin(theta0) * np.sin(phi0)
Vz = V0 * np.cos(theta0)
u_ini = np.concatenate((X0, Vx, Vy, Vz))
#Résultats
u = resultat(tf, dt, u_ini)
#Tracer
final_3D(N,u)
```

9.3 3-Z-P

9.3.1 Fonction dérivée

Au début, nous définissons la distance, d, entre 2 individus dans l'espace $\mathbb{R}^2$, et la fonction dérivée par le système (5) :

```
def d(xi,yi,xj,yj):
    return np.sqrt((xi-xj)**2+(yi-yj)**2)

def deriv(t,u):
    du = np.empty(u.size)

    # Dérivée de la vitesse
    for i in range(N):
        du[i] = u[2*N+i]
        du[N+i] = u[3*N+i]
        a = 0
        b = 0
        for j in range(N):
            rij = d(u[i],u[N+i],u[j],u[N+j])
            if i != j:
                a += 1/N*phi(rij)*(u[2*N+j]-u[2*N+i])+1/N*dV(rij)*(u[j] - u[i])/(rij)
                b += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])+1/N*dV(rij)*(u[N+j] - u[N+i])/(rij)
            else:
                a += 1/N*phi(rij)*(u[2*N+j]-u[2*N+i])
                b += 1/N*phi(rij)*(u[3*N+j]-u[3*N+i])
        for p in range(M):
            rip = d(u[i],u[N+i],u[4*N+p],u[4*N+M+p])
            if u[2*N+i]*u[4*N+3*M+p] > u[3*N+i]*u[4*N+2*M+p]:
                a += 1/M*psi(rip)*(u[4*N+3*M+p]-u[2*N+i])+1/M*dVbar(rip)*(-u[i]+u[4*N+p])/rip
                b += 1/M*psi(rip)*(-u[4*N+2*M+p]-u[3*N+i])+1/M*dVbar(rip)*(-u[N+i]+u[4*N+M+p])/rip
```

```
            elif u[3*N+i]*u[4*N+2*M+p]>u[2*N+i]*u[4*N+3*M+p]:
                a += 1/M*psi(rip)*(-u[4*N+3*M+p]-u[2*N+i]) + 1/M*dVbar(rip)*(-u[i]+u[4*N+p])/rip
                b += 1/M*psi(rip)*(u[4*N+2*M+p]-u[3*N+i]) + 1/M*dVbar(rip)*(-u[N+i]+u[4*N+M+p])/rip
        du[2*N+i] = a
        du[3*N+i] = b
    for q in range(M):
        du[4*N+q] = u[4*N+2*M+q]
        du[4*N+M+q] = u[4*N+3*M+q]
        x_c = 1/N * sum(u[0:N])
        y_c = 1/N * sum(u[N:2*N])
        rpc = d(u[4*N+q],u[4*N+M+q],x_c,y_c)
        du[4*N+2*M+q] = dA(rpc)*(-u[4*N+q]+x_c)/rpc
        du[4*N+3*M+q] = dA(rpc)*(-u[4*N+M+q]+y_c)/rpc
    return du
```

9.3.2 Applications numériques

Nous définissons aussi ϕ, V, V', A, A', $\overline{V}$ et $\overline{V}'$ et traçons leur images.

```
def phi(r):
    return r/(2+r**2)

def psi(r):
    return 2*r/(2+r**10)

def V(r):
    return r*(np.log(r)-1)

def dV(r):
    return np.log(r)

def A1(r):
    return np.sqrt(r)

def dA1(r):
    return 1/2*r**(-1/2)

def A2(r):
    return r**2 + (np.sqrt(2)/4 - 4)*r + 4 + np.sqrt(2)/2

```

```
def dA2(r):
    return 2*r + (np.sqrt(2)/4 - 4)

def Vbar(r):
    return -1/5*np.log(r)

def dVbar(r):
    return -1/(5*r)

r = np.linspace(0.001, 5, 1000)
r1 = np.linspace(0.001, 2, 400)
r2 = np.linspace(2.001, 5, 600)
fig = plt.figure(figsize=(8, 6))
plt.xlabel("distance r", fontsize=20)
plt.ylim([-3, 3])
plt.plot(r,phi(r),label = r"$\phi(r) = \frac{r}{2+r^3}$ ")
plt.plot(r,V(r),label = r"$V(r) = r(ln(r)-1)$")
plt.plot(r,psi(r),label = r"$\psi(r) = \frac{2r}{2+r^{10}}$")
plt.plot(r,Vbar(r),label = r"$\overline{V}(r) = -\frac{1}{5}ln(r)$")
plt.plot(r1,A1(r1),color = "purple", label = r"$A(r) = \sqrt{r},\,si\,r\,\leqslant\,2, r^2 + (\frac{1}{2\sq
plt.plot(r2,A2(r2),color = "purple")
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.legend(fontsize=15)
plt.show()
```

Nous affichons le code dans le Cas M = 1 ci-dessous.

```
# M =1
N = 50
M = 1
u_1 = np.load('~/N_50_1.npy')
u_ini_proies = [sous_liste[-1] for sous_liste in u_1]
x_c_ini = 1/N * sum(u_ini_proies[0:N])
y_c_ini = 1/N * sum(u_ini_proies[N:2*N])
#On mettre (0,0) comme le central de masse
k = 1
"""
Veuillez noter que dans certains cas,
pour une meilleure représentation dans le GIF produit
```

```
nous réduisons proportionnellement la vitesse initiale des proies
"""
u_ini_proies0 = [x - x_c_ini if i < N else x - y_c_ini if i < 2*N
                 else x/k for i, x in enumerate(u_ini_proies)]
d1 = np.ones(M) # la distance entre un prédateur et le centre de masse des prores
Vp0 = np.random.normal(loc=0.1, scale=0, size=M)
theta0_p = np.random.uniform(low=0, high=2*np.pi, size=M)

Xp0 = d1 * np.cos(theta0_p)
Yp0 = d1 * np.sin(theta0_p)
Vx0 = Vp0 * np.cos(np.pi + theta0_p)
Vy0 = Vp0 * np.sin(np.pi + theta0_p)

u_ini = np.hstack((u_ini_proies0, Xp0, Yp0, Vx0, Vy0))
u = resultat(tf,dt,u_ini)
```

Cette partie du code permet de présenter les trajectoires de mouvement des proies et des prédateurs à chaque intervalle de temps, afin d'améliorer la présentation dans le rapport.

```
for j in range(20):
    fig = plt.figure(figsize=(8, 8))
    for i in range(N):
        plt.plot(u[i,:][10*j:10*(j+1)],u[N+i,:][10*j:10*(j+1)],'-',color = "green")
        plt.plot(u[i,:][10*j:10*(j+1)][-1],u[N+i,:][10*j:10*(j+1)][-1],'o',color = "green")
    for k in range(M):
        plt.plot(u[4*N+k,:][10*j:10*(j+1)],u[4*N+M+k,:][10*j:10*(j+1)],'-',color = "red")
        plt.plot(u[4*N+k,:][10*j:10*(j+1)][-1],u[4*N+M+k,:][10*j:10*(j+1)][-1],'o',color = "red")
```

Ce code est utilisé pour générer le GIF. Le code suivant sert à afficher le GIF ainsi qu'à enregistrer le GIF et les données associées.

```
def update(frame, N, M, u, scat_proies, scat_predateurs, lines_proies, lines_predateurs):
    for line in lines_proies + lines_predateurs:
        line.set_data([], [])

    x_proies = u[0:N, frame]
    y_proies = u[N:2*N, frame]
    scat_proies.set_offsets(np.c_[x_proies, y_proies])

    x_predateurs = u[4*N:4*N+M, frame]
```

```
    y_predateurs = u[4*N+M:4*N+2*M, frame]
    scat_predateurs.set_offsets(np.c_[x_predateurs, y_predateurs])

    for i in range(max(0, frame-4), frame+1):
        for j in range(N):
            lines_proies[j].set_data(u[j, max(0, frame-4):frame+1], u[N+j,
                                     max(0, frame-4):frame+1])
        for j in range(M):
            lines_predateurs[j].set_data(u[4*N+j, max(0, frame-4):frame+1],
                                         u[4*N+M+j, max(0, frame-4):frame+1])

    return scat_proies, scat_predateurs, *lines_proies, *lines_predateurs

def create_animation(u, N, M):
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.set_xlim((-5, 15))
    ax.set_ylim((-13, 8))
    scat_proies = ax.scatter([], [], color='green',s =10)
    scat_predateurs = ax.scatter([], [], color='red',s=10)

    lines_proies = [ax.plot([], [], color='green', linewidth=0.5)[0] for _ in range(N)]
    lines_predateurs = [ax.plot([], [], color='red', linewidth=0.5)[0] for _ in range(M)]

    anim = FuncAnimation(fig, update, frames=350, fargs=(N, M, u, scat_proies,
                         scat_predateurs, lines_proies, lines_predateurs),
                         blit=True, interval=20)

    anim.save('N = {N}, M = {M}.gif', writer='pillow')
```

```
"""
create_animation(u, N, M)
array_data = u
file_path = "~/N={N},M={M}.npy" # Dans Autre M, on doit changer "1" ici.
os.makedirs(os.path.dirname(file_path), exist_ok=True)
np.save(file_path, array_data)
"""
```

9.4 Problème du mur carré et circulaire

9.4.1 Cas du mur carré

Ci dessous, nous avons modifié la fonction ME de la méthode d'euler afin d'y ajouter les conditions de reflexivité. Les simulations sont réalisés à partir d'un faible nombre d'agents car nous souhaitons pouvoir observer sans grandes difficulté les trajectoires des agents.

```
def ME(x, dx, y, deriv): #Méthode d'Euler explicite avec le mur

  #On calcul les nouvelles positions
  new_positions = y+dx*deriv(x,y)

  #On vérifie si la position est en dehors du mur:
  #D'abord selon l'axe x
  for i in range(N):
      if new_positions[i] > xmax:
          new_positions[i] = xmax - (new_positions[i]- xmax)

          new_positions[2*N+i] = -new_positions[2*N+i]
      elif new_positions[i] < xmin: #On fait la même chose pour le coté gauche
          new_positions[i] = xmin+(xmin-new_positions[i])
          new_positions[2*N+i] = -new_positions[2*N+i]
   #Ensuite selon l'axe y
  for i in range(N,2*N): #On commence à N car on a déjà traité les positions x
      if new_positions[i] > ymax:
          new_positions[i] = ymax - (new_positions[i] - ymax)

          #Ensuite, on inverse la direction de la vitesse de l'individu
          new_positions[3*N+i-N] = -new_positions[3*N+i-N]
          """
            On soustrait par N car les composantes de vitesse selon y sont stockées
            à partir de 3*N jusqu'a 4*N. Donc, on doit indexer les valeurs
            de 0 à N-1 pour acceder aux éléments
            """
      elif new_positions[i] < ymin:
          new_positions[i] = ymin + (ymin - new_positions[i])
          new_positions[3*N+i-N] = -new_positions[3*N+i-N]
  return new_positions
```

9.4.2 Cas du mur circulaire

```python
import numpy as np
import matplotlib.pyplot as plt

# Fonction pour calculer la distance entre deux points
def distance(x1, y1, x2, y2):
    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)

# Fonction dérivée pour la mise à jour des vitesses et positions
def deriv(u):
    du = np.zeros_like(u)
    for i in range(N):
        # Mise à jour de la position en fonction de la vitesse
        du[i] = u[2*N+i]  # dx/dt = vx
        du[N+i] = u[3*N+i]  # dy/dt = vy
        acc_x = 0
        acc_y = 0
        for j in range(N):
            if i != j:
                rij = distance(u[i], u[N+i], u[j], u[N+j])
                acc_x += phi(rij) * (u[2*N+j] - u[2*N+i]) + dV(rij) * (u[j] - u[i]) / rij
                acc_y += phi(rij) * (u[3*N+j] - u[3*N+i]) + dV(rij) * (u[N+j] - u[N+i]) / rij
        # Mise à jour des vitesses
        du[2*N+i] = acc_x
        du[3*N+i] = acc_y
    return du

# Étape de la méthode de Runge-Kutta 4
def RK4_step(u, dt):
    k1 = dt * deriv(u)
    k2 = dt * deriv(u + 0.5 * k1)
    k3 = dt * deriv(u + 0.5 * k2)
    k4 = dt * deriv(u + k3)
    return u + (k1 + 2*k2 + 2*k3 + k4) / 6

# Réflexion des particules sur le mur circulaire
def reflect(u, i):
    pos = u[i], u[N+i]
```

```
    vel = u[2*N+i], u[3*N+i]
    dist = np.hypot(*pos)
    # Si la distance est supérieure au rayon, on fait réfléchir la particule
    if dist > rayon_cercle:
        norm = pos / dist
        # Changement de la direction de la vitesse
        u[2*N+i], u[3*N+i] = vel - 2 * np.dot(vel, norm) * norm
        # Réinitialisation de la position à la frontière du cercle
        u[i], u[N+i] = norm * rayon_cercle
    return u

# Paramètres de simulation
N = 6  # Nombre de particules
rayon_cercle = 20  # Rayon du cercle
temps_fin = 100  # Temps de fin de la simulation
dt = 0.1  # Pas de temps

# Conditions initiales aléatoires avec des vitesses réduites
u = np.random.uniform(-10, 10, 4 * N)

# Trajectoires calculées pendant la simulation
trajectoires = []
temps_simulation = np.arange(0, temps_fin, dt)
for t in temps_simulation:
    u = RK4_step(u, dt)
    for i in range(N):
        u = reflect(u, i)
    trajectoires.append(u[:2*N])

# Affichage des trajectoires
plt.figure(figsize=(10, 10))
for i in range(N):
    plt.plot(np.array(trajectoires)[:, i], np.array(trajectoires)[:, N+i])

# Dessin du mur circulaire
cercle = plt.Circle((0, 0), rayon_cercle, color='red', fill=False, linewidth=2)
plt.gca().add_patch(cercle)

plt.xlabel('X')
```

```
77 plt.ylabel('Y')
78 plt.title('Trajectoires avec rebonds sur un mur circulaire')
79 plt.grid(True)
80 plt.axis('equal')
81 plt.show()
82
83
```