Modélisation de mouvements de groupe

TIPE mathématiques appliquées et informatique pratique

Omar Bennouna

11 juin 2019

Motivation

Figure – Drone Intel

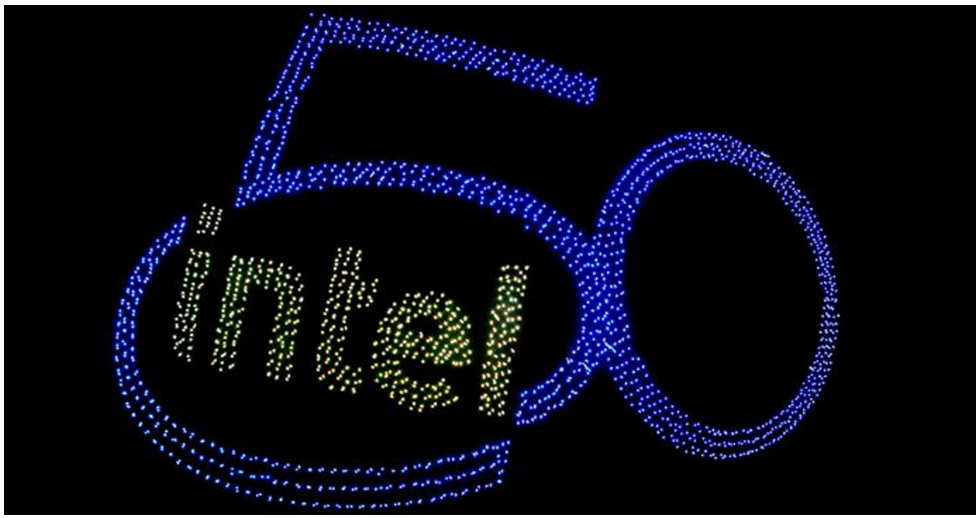

Figure – Spectacle de drones

- Spectacle de 2018 drones par Intel en Août 2018
- Comment gérer un tel mouvement ?

Motivation

Figure – Troupeau de poissons

Figure – Troupeau d'oiseaux

- On veut s'inspirer des mouvement observés dans la nature.

Préliminaires

- On cherche à gérer le mouvement d'un ensemble d'unités
- On veut s'inspirer des mouvements de groupe observés dans la nature : oiseaux, bancs de poissons, etc.
- Modèle discret : $\overrightarrow{OM}(t+1) = \overrightarrow{V}(t) + \overrightarrow{OM}(t)$
- Toutes les masses sont égales à 1
- 3 lois fondamentales proposées par Craig Reynolds
  1. **Séparation :** Garder une distance convenable des autres agents pour éviter les collisions et la surcharge
  2. **Alignement :** Se diriger vers la direction moyenne du troupeau
  3. **Cohésion :** Rester proche du troupeau pour éviter une dispersion des agents

Premier modèle et implémentation

- On commence par un premier modèle simple
  - 2 agents
  - Une force attractive : $\overrightarrow{F_{attr}} = \lambda \times \vec{r}_{i \to j}$
  - Une force répulsive $\overrightarrow{F_{rep}} = -\mu \times \frac{\vec{r}_{i \to j}}{\|\vec{r}_{i \to j}\|^2}$
- Positions stockées dans un `numpy.array list` tel que `pos[i][t]` représente la position de l'agent i à l'intant t.

Simulation et commentaires

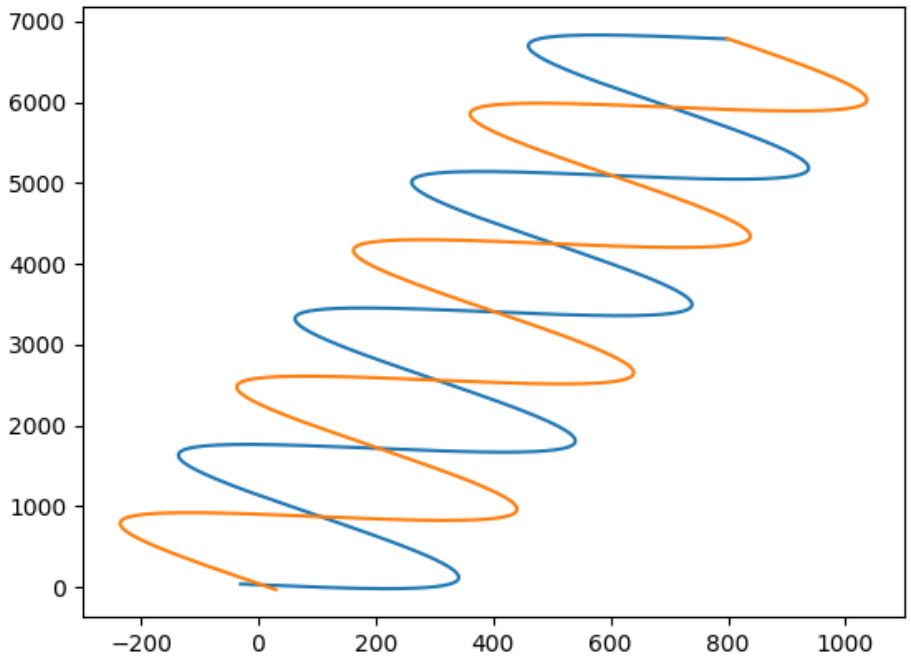

Figure – Simulation du premier algorithme
$\lambda = 5 \times 10^{-4}$ $\mu = 4.5 \times 10^{-4}$

- Modèle inutilisable : présence d'oscillations parasites

Amélioration

- On ajoute une force qui amortit les oscillations qui obéit à la loi suivante :

$$\overrightarrow{F_{am}} = \begin{cases} \overrightarrow{V} \times (f-1) & \text{si } \left\|\overrightarrow{V}\right\| \geq \left\|\overrightarrow{V_{lim}}\right\| \\ \overrightarrow{V} \times (f-1) + (\left\|\overrightarrow{V_{lim}}\right\| - \left\|\overrightarrow{V}\right\|) \times \overrightarrow{v_{inf}} & \text{sinon} \end{cases}$$

- Avec :
  - f coefficient d'amortissement inférieur à 1 strictement
  - $\overrightarrow{V_{lim}}$ la vitesse vers laquelle on veut converger
  - $\overrightarrow{v_{inf}} = \frac{\overrightarrow{V_{lim}}}{\left\|\overrightarrow{V_{lim}}\right\|}$ le vecteur directeur de cette vitesse

Simulation et commentaires

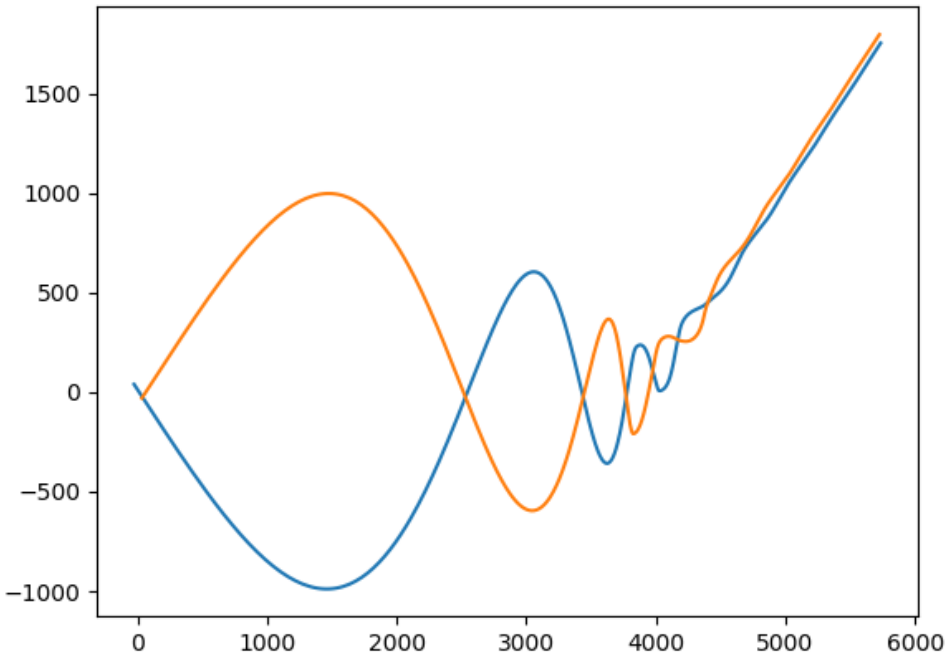

Figure – Simulation algorithme modifié
$\lambda = 5 \times 10^{-4}$ $\mu = 1$ $f = 9.9 \times 10^{-1}$

Simulation et commentaires

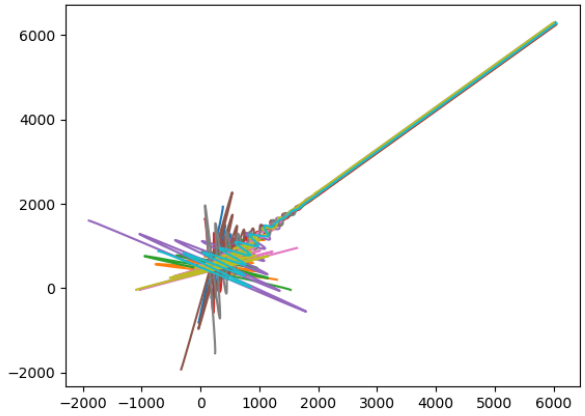

Figure – $\lambda = 5 \times 10^{-4}$ $\mu = 1$ $N = 10$

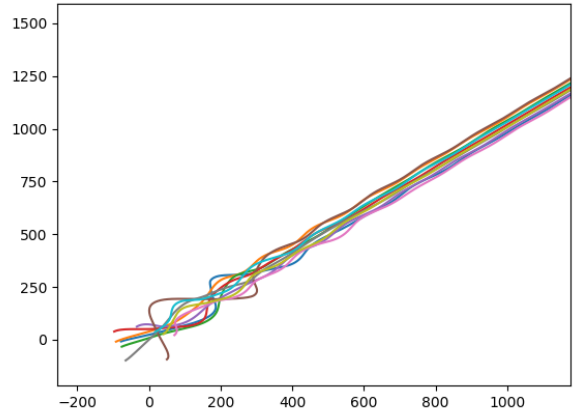

Figure – $\lambda = 5 \times 10^{-4}$ $\mu = 1$ $N = 10$

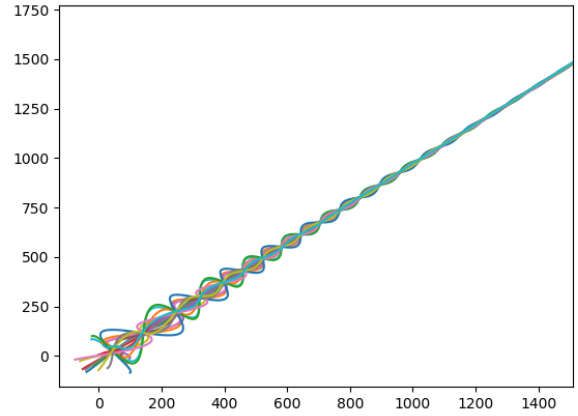

Figure – $\lambda = 5 \times 10^{-4}$ $\mu = 5 \times 10^{-3}$ $N = 5$

- Lois 1, 2 et 3 respectées
- Mais le modèle n'est pas très réaliste :
  - Portée d'interaction infinie
  - Norme usuelle non différentiable en 0 et donc divergence de la vitesse
  - Concerne essentiellement les mouvements d'oiseaux voyageurs

Modèle plus réaliste

- On remplace $\|.\|$ par $\|x\|_\sigma = \frac{\sqrt{1+\epsilon\|x\|^2}-1}{\epsilon}$ avec $\epsilon > 0$ qui est différentiable en 0.
- On utilise des graphes de proximité G(t)=(V,E) dépendant du temps
  - Implémentation par matrice d'adjacence `numpy.array` symétrique $A = (a_{i,j})$ (graphe non orienté)
  - Interaction entre l'agent i et l'agent j $\iff$ $a_{i,j} > 0$ $\iff$ d(i,j)<r, avec r rayon maximal d'interaction
  - Coefficients varient d'une façon continuement dérivable.
  - $a_{i,j} = C \times \rho_h(\frac{\|q_i - q_j\|_\sigma}{r_\sigma})$ avec

$$\rho_h(x) = \begin{cases} 1 & \text{si } x \in [0, h[\\ \frac{1}{2}(1 + cos(\pi(\frac{x-h}{1-h})) & \text{si } x \in [h, 1] \\ 0 & \text{sinon} \end{cases}$$

une fonction qui varie continuement de 1 à 0, C une constante grace à laquelle A est bistochastique, et q_i la position de l'agent i.

Courbe de ρ_h

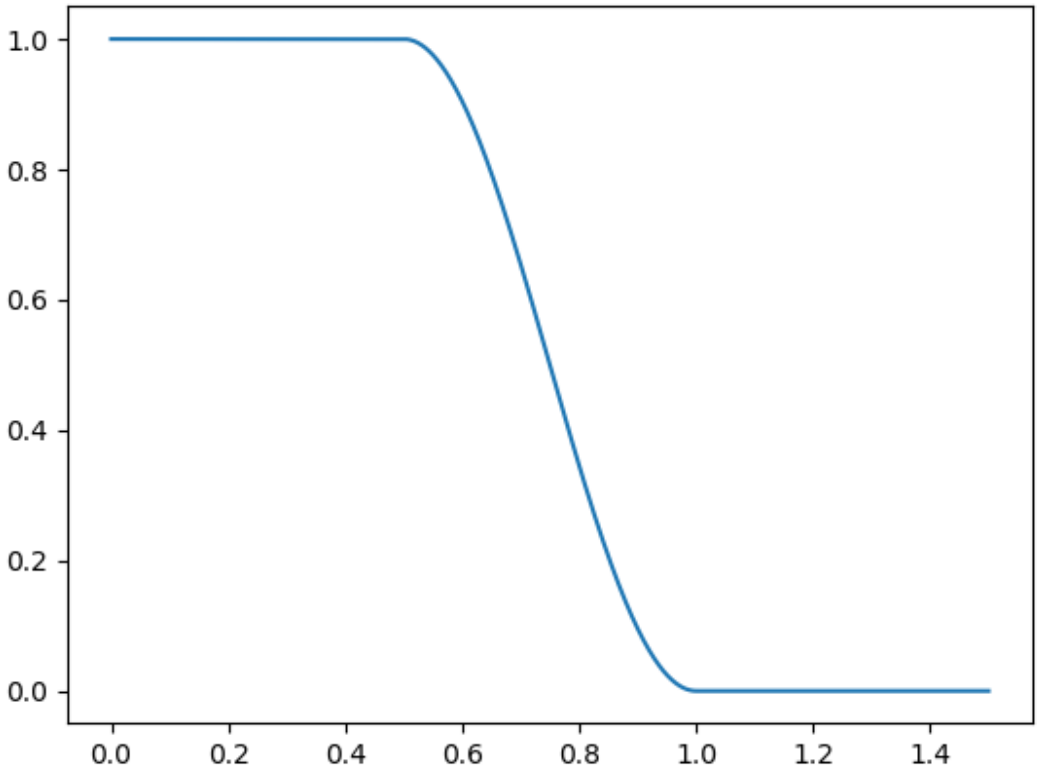

Figure – Fonction ρ_h pour $h = \frac{1}{2}$

Nouvelle démarche

- Le calcul de la vitesse à l'instant t+1 ayant celles de l'intant t :
  - $\overrightarrow{V_i(t+1)} = \overrightarrow{V_i(t)} + \sum_{j \in N_i} (\overrightarrow{V_j(t)} - \overrightarrow{V_i(t)}) a_{i,j} + \overrightarrow{F_{attr}} + \overrightarrow{F_{rep}}$
  - La relation précédente peut être écrite d'une façon plus concise : $V(t+1) = (M(t) + I)V(t) + F_a Q(t) + F_r(Q(t))$,
  - $V(t) = \begin{pmatrix} \overrightarrow{V_1(t)} \\ . \\ . \\ . \\ \overrightarrow{V_n(t)} \end{pmatrix}$ et $Q(t) = \begin{pmatrix} \overrightarrow{q_1(t)} \\ . \\ . \\ . \\ \overrightarrow{q_n(t)} \end{pmatrix}$

Nouvelle démarche

- $D = diag(\sum_{j=1,j\neq 1}^{n} a_{1,j}, ..., \sum_{j=1,j\neq n}^{n} a_{n,j})$
- $M(t) = A + D$
- $F_a = \lambda \times (n \times D - A)$
- $F_r(Q) = -\mu \times \begin{pmatrix} \sum_{j=1,j\neq 1}^{n} \frac{(\overrightarrow{q_j} - \overrightarrow{q_1})a_{1,j}}{\left\|\overrightarrow{q_j} - \overrightarrow{q_n}\right\|^2} \\ . \\ . \\ . \\ \sum_{j=1,j\neq n}^{n} \frac{(\overrightarrow{q_j} - \overrightarrow{q_n})a_{n,j}}{\left\|\overrightarrow{q_j} - \overrightarrow{q_n}\right\|^2} \end{pmatrix}$ opérateur non linéaire

- Calcul de la position à l'instant t+1
  - $\overrightarrow{q_i(t+1)} = \overrightarrow{q_i(t)} + \overrightarrow{V_i(t)}$
  - Vectoriellement Q(t+1)=Q(t)+V(t)

Simulation avec r=+∞

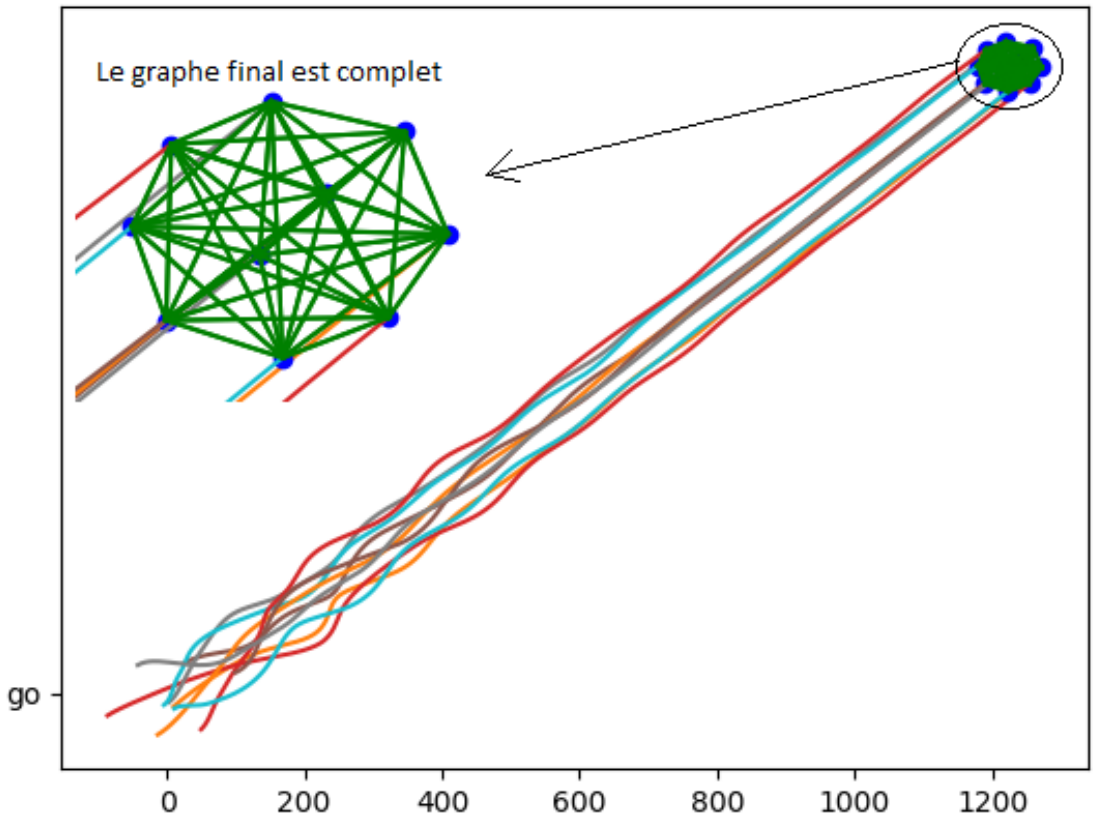

Figure – Test pour 10 agents
$\lambda = 5 \times 10^{-4}$ $\mu = 1$ $r = +\infty$

Simulation avec r fini

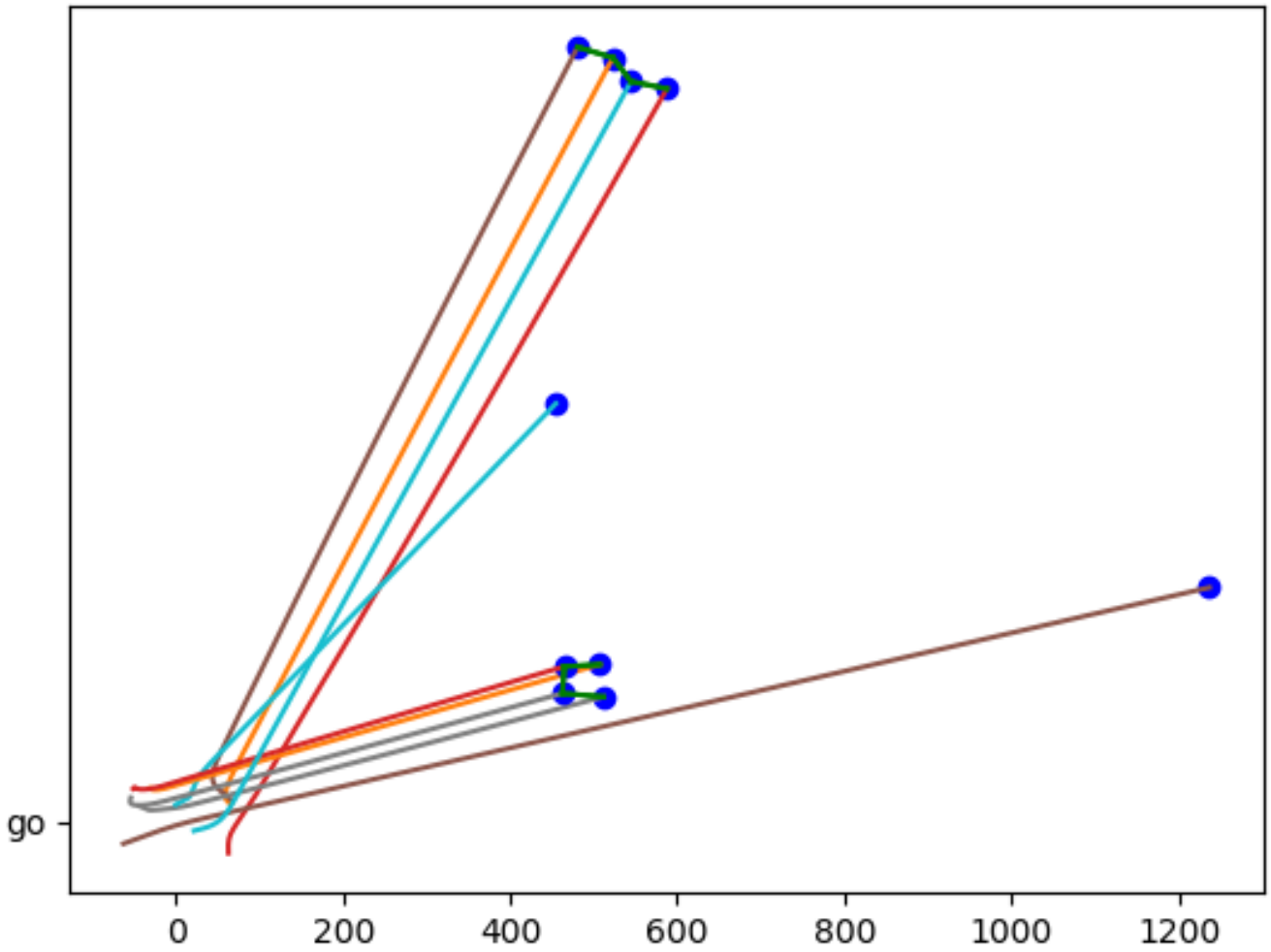

Figure – Test algorithme réaliste pour 10 agents
$\lambda = 5 \times 10^{-4}$ μ=1 $r = 50$

Ajout d'une cible

- On ajoute une cible que tous les agents doivent suivre de position $\overrightarrow{c(t)}$
- Exemples : leader du troupeau, proie, etc.
- La force exercée sur l'agent i sera $\overrightarrow{F_{cible}} = \gamma \times (\overrightarrow{c(t)} - \overrightarrow{q_i(t)})$ avec γ constante

Test1 : Mouvement de la cible rectiligne uniforme

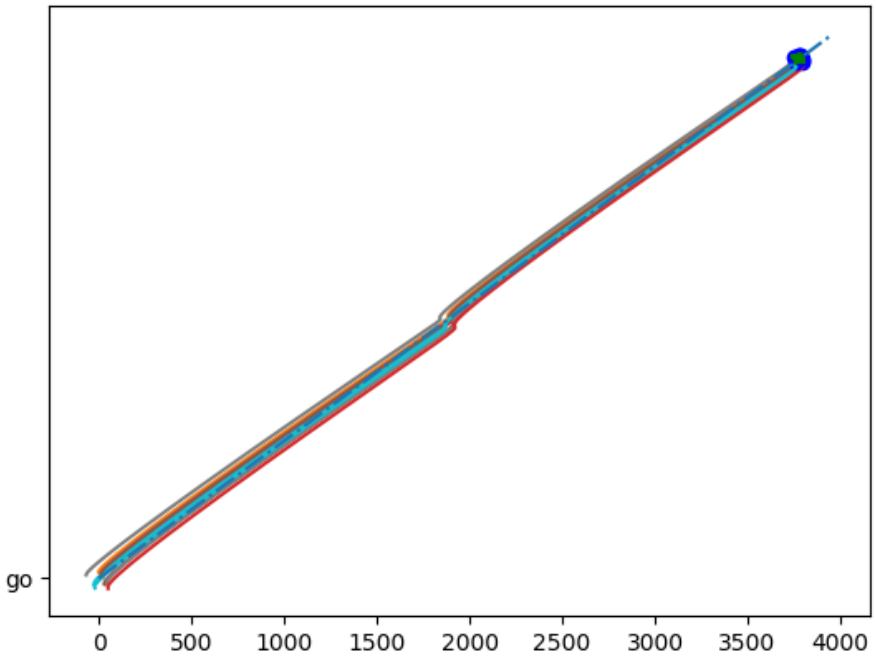

Figure – Test 1

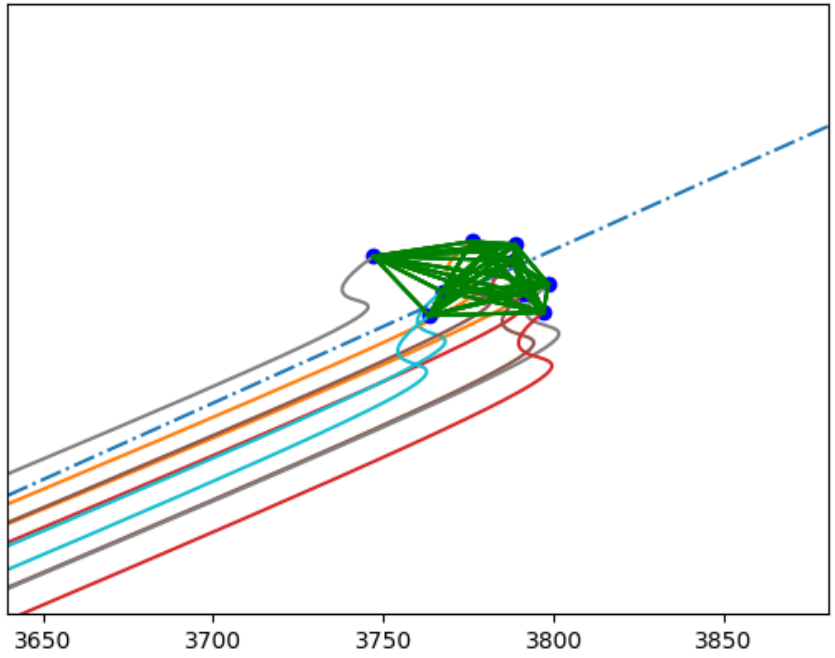

Figure – Zoom sur la figure précédente

- La trajectoire de la cible est en pointillés

Test2 : Mouvement de la cible circulaire uniforme

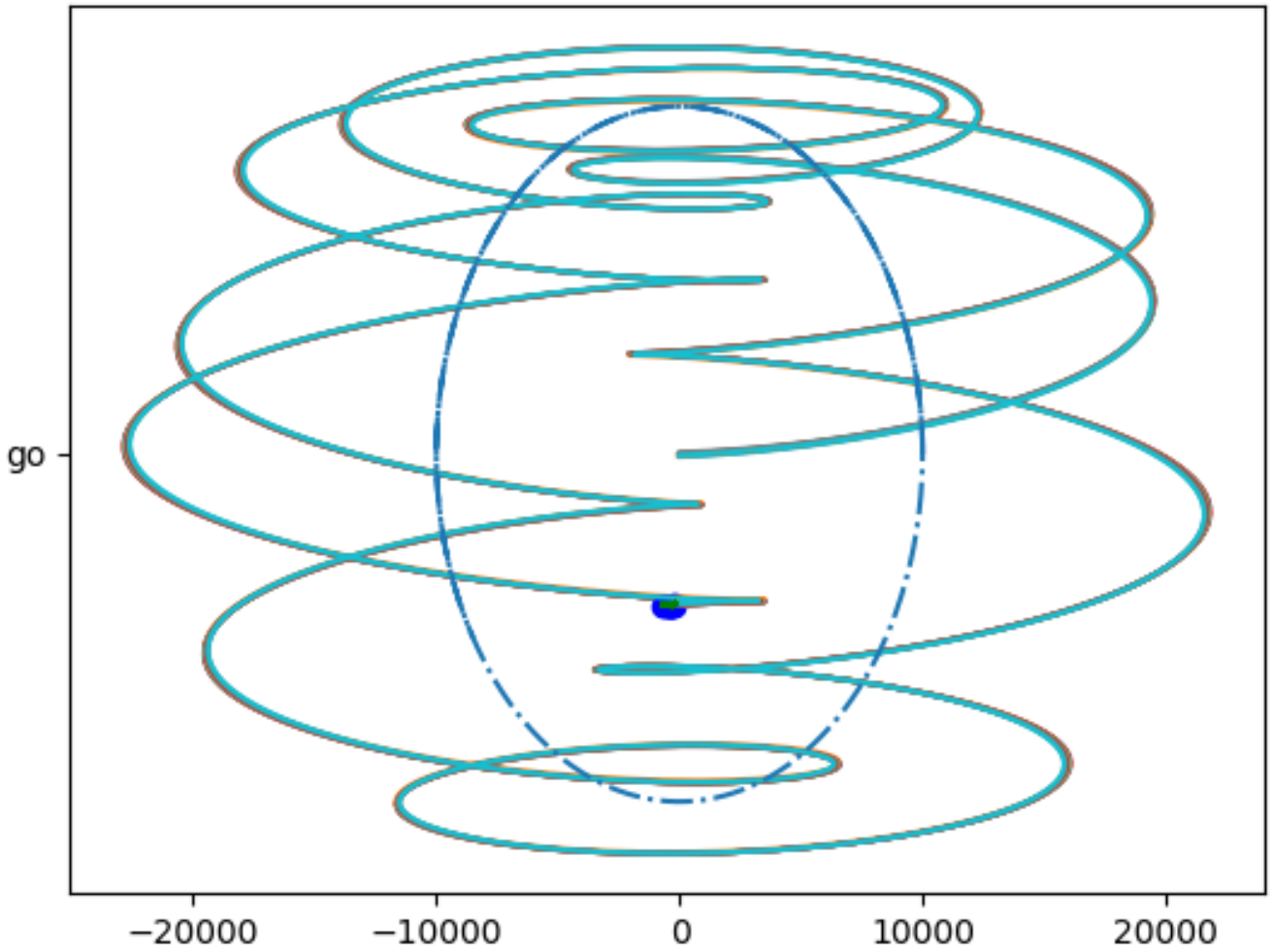

Figure – Test2

Remarques et amélioration

- Le modèle n'est pas optimal.
- On introduit une "Force attractive" dans l'espace des vitesses définie par $\overrightarrow{F_{stab}} = -k \times \left\| \overrightarrow{V_{cible}} - \overrightarrow{V_{agent}} \right\|^{\beta} \times (\overrightarrow{V_{cible}} - \overrightarrow{V_{agent}})$ avec k et β des constantes positive.
- β est décroissant avec les fluctuations.

Simulations

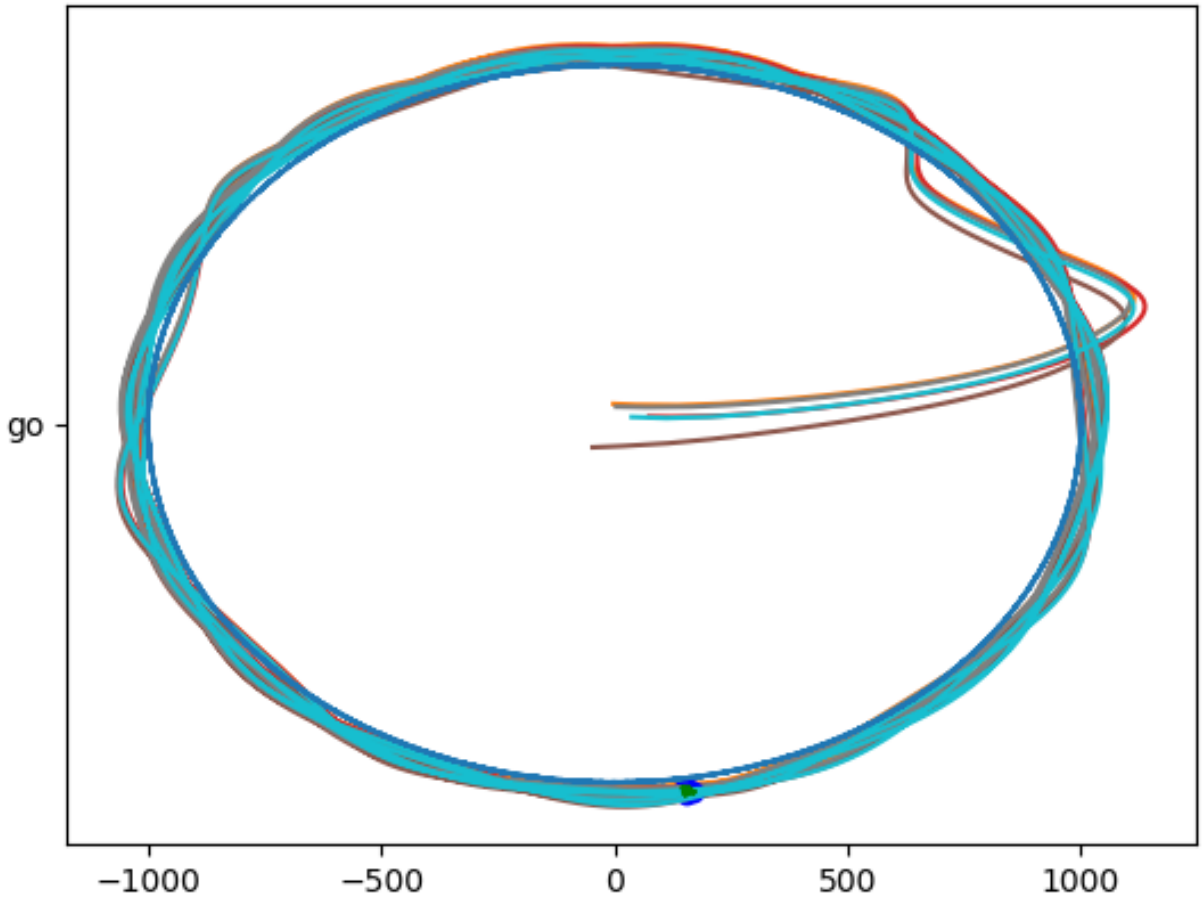

Figure – Simulation avec nouvelle règle
$\beta = 1.5$

- La vitesse des agents converge bien vers celle de la cible.

Simulations

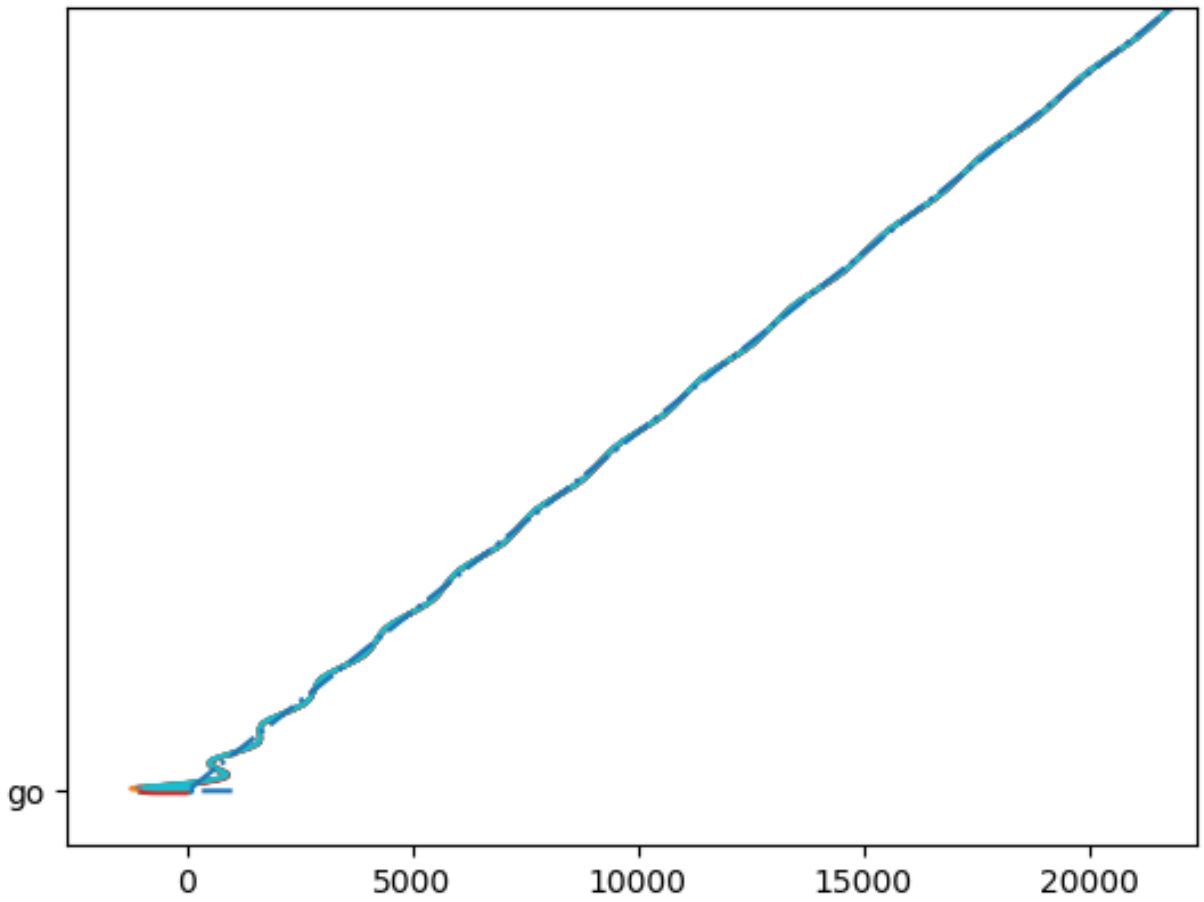

Figure – Simulation avec nouvelle règle

Evolution des vitesses

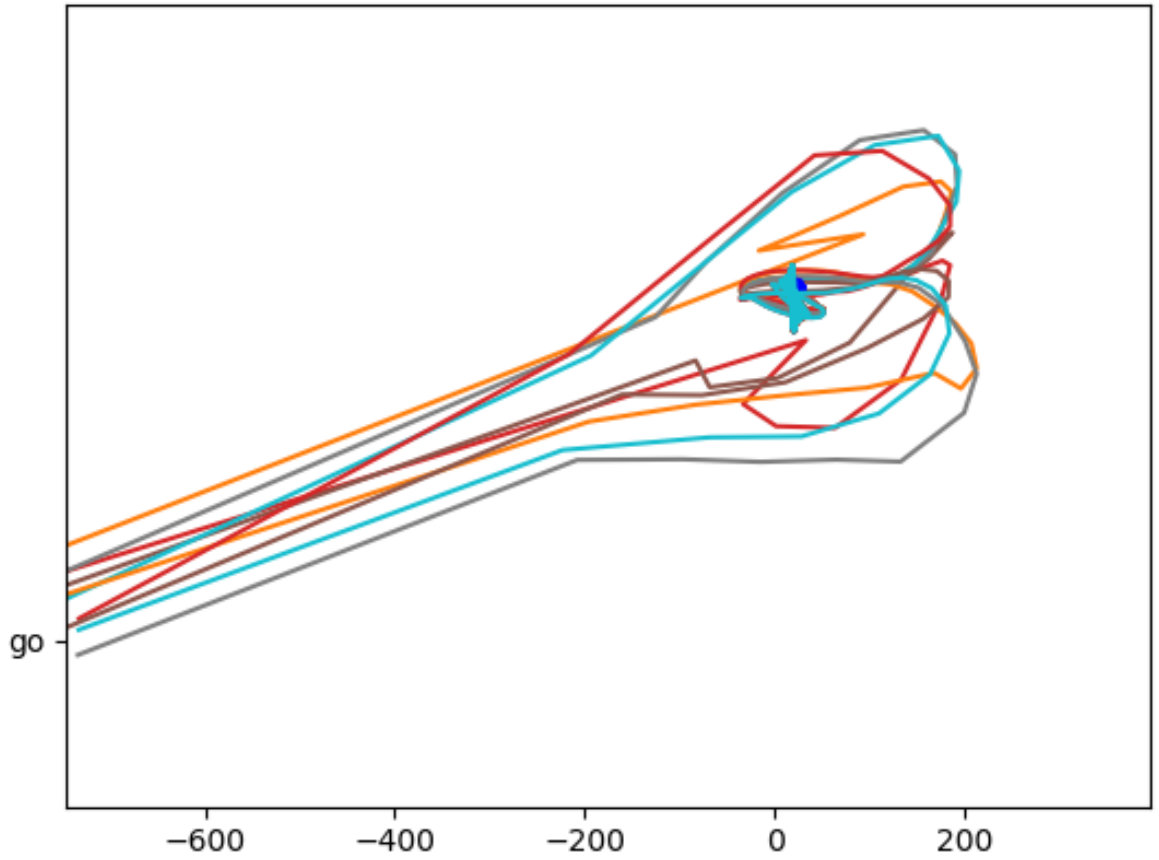

Figure – Simulation avec nouvelle règle

Evolution des vitesses

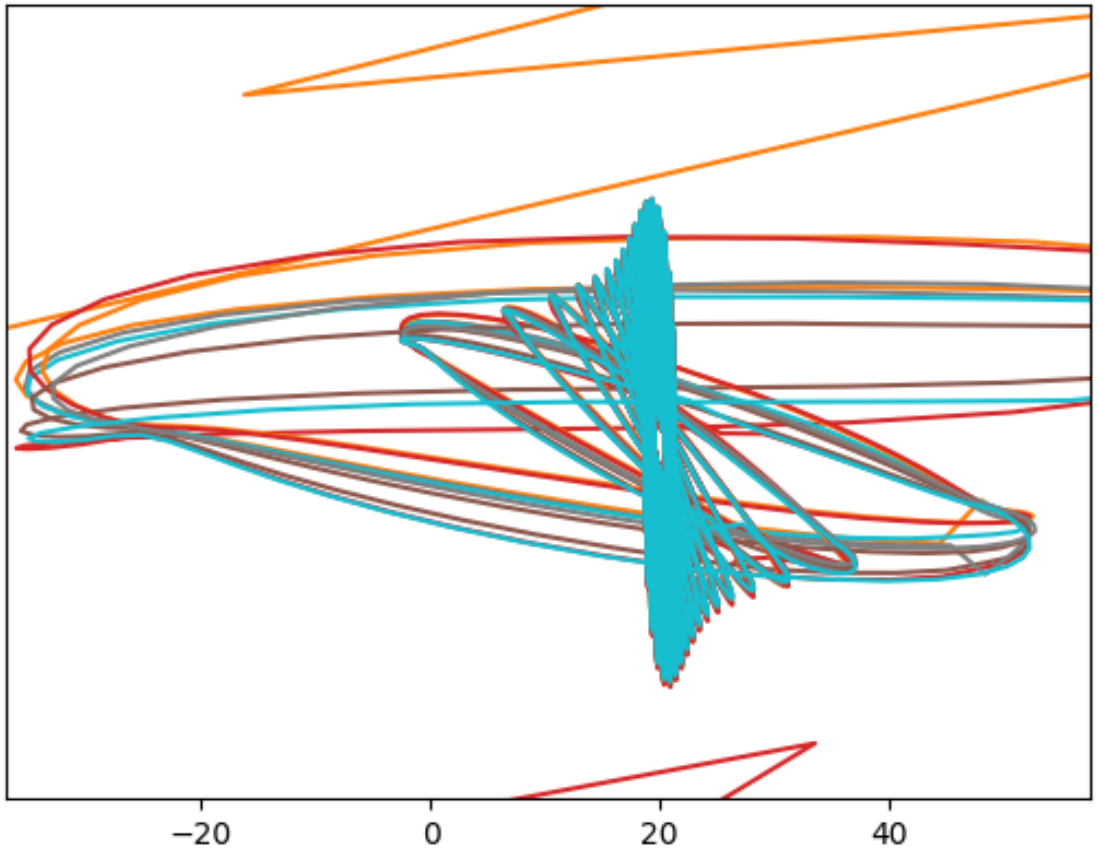

Figure – Zoom sur la figure précédente

Evolution des vitesses

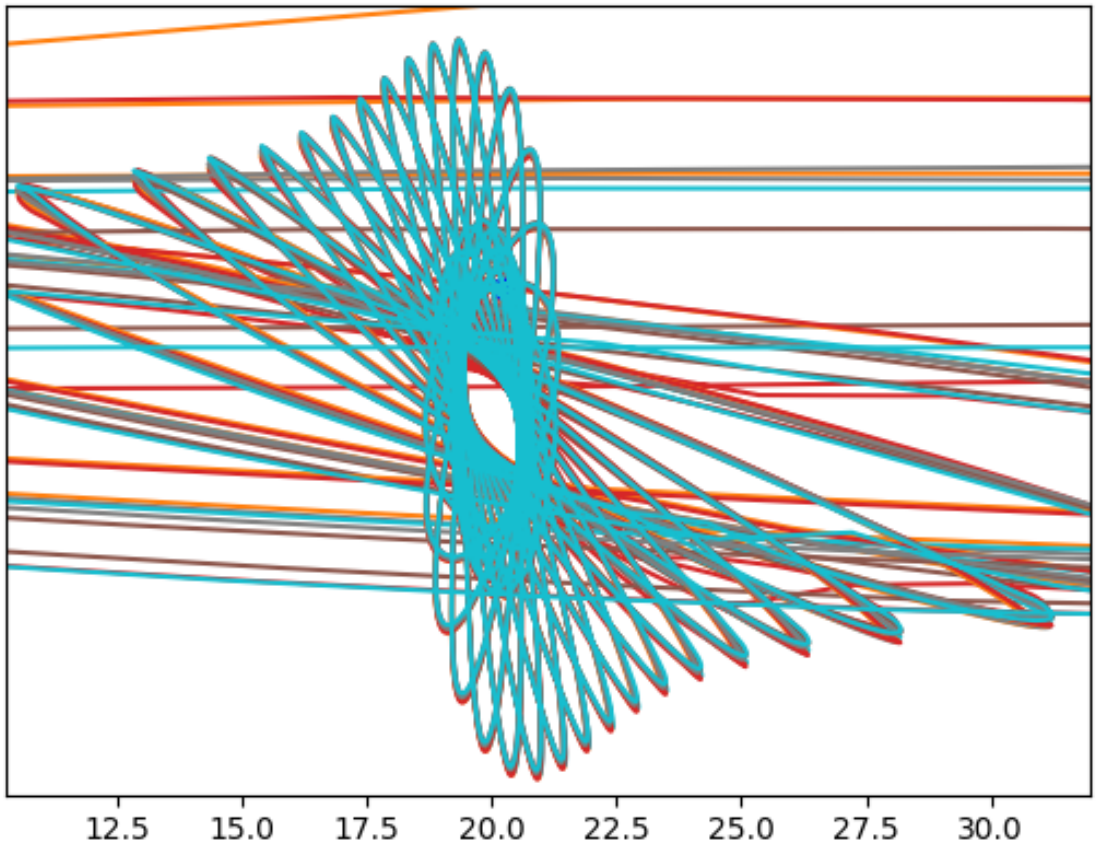

Figure – Zoom sur la figure précédente

Evolution des vitesses

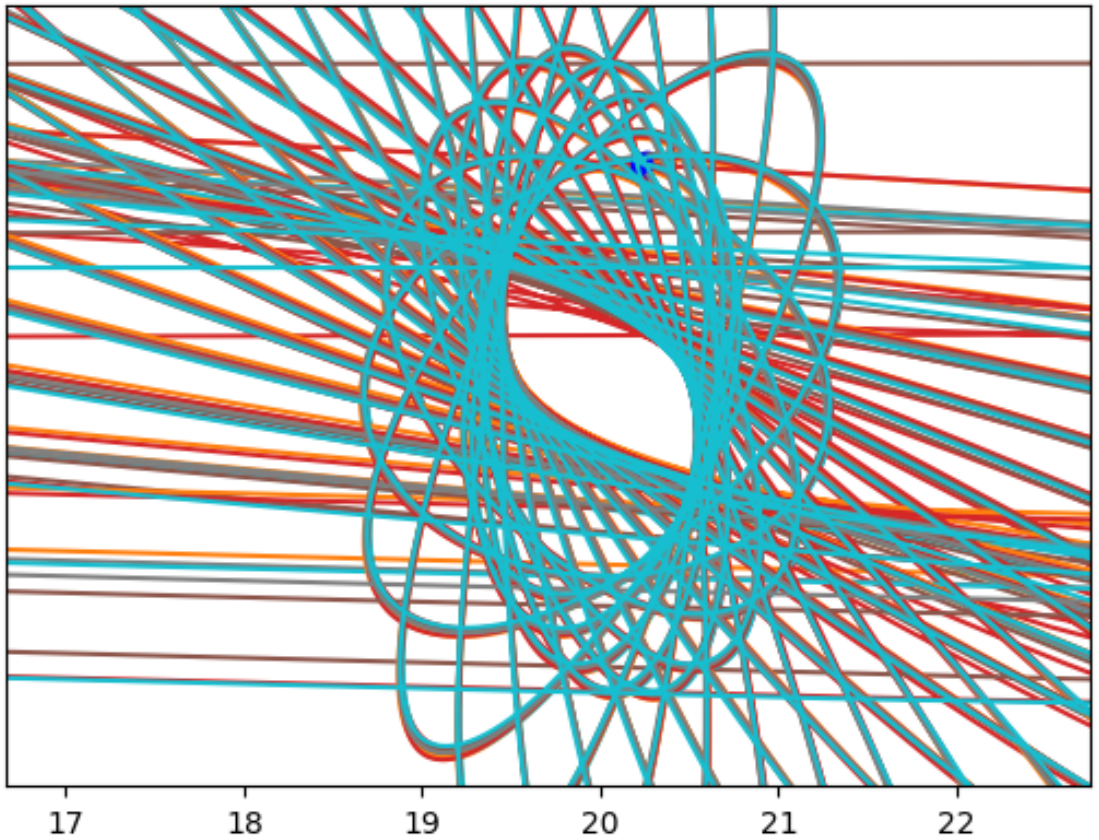

Figure – Zoom sur la figure précédente

Evolution des vitesses

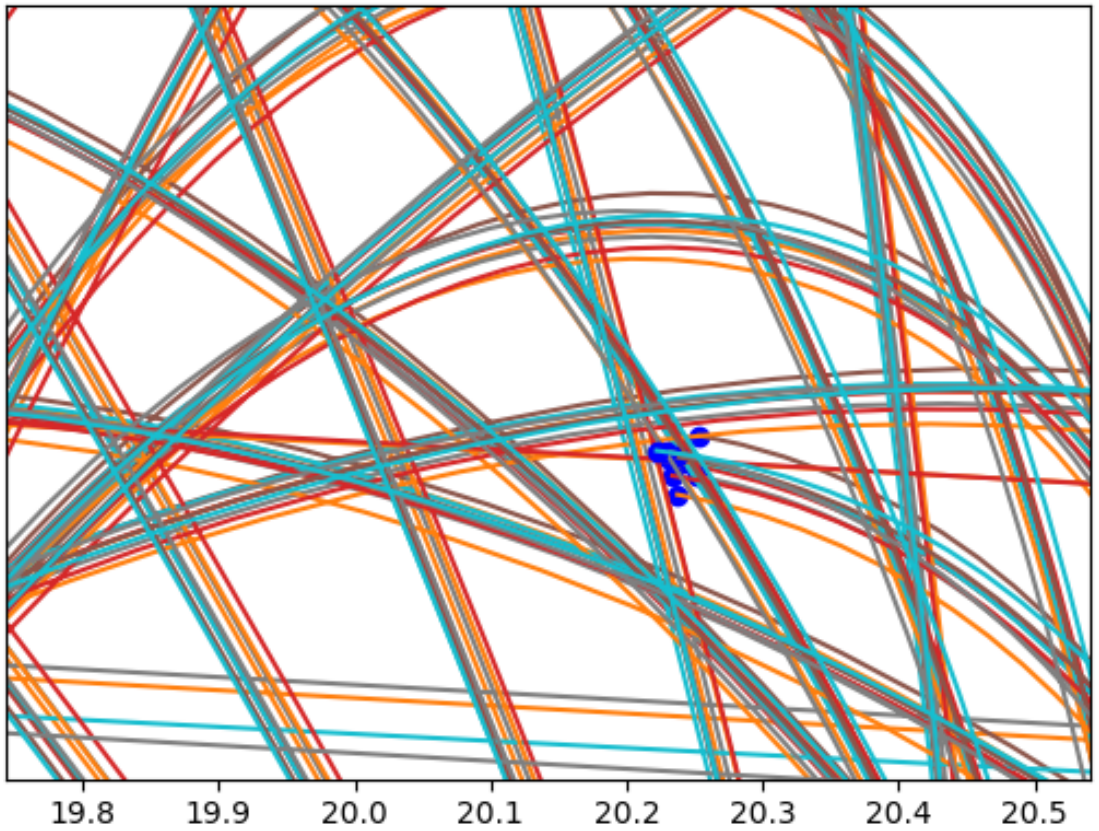

Figure – Zoom sur la figure précédente

Ajout d'obstacles

- Disques définis par la position de leur centre et leur rayon.
- Force répulsive infinie lorsque l'agent est tangent à l'obstacle.
- Force répulsive en $\frac{1}{d}$.
- Donc force de la forme $\overrightarrow{F_{obs}} = -\omega \times \frac{\widetilde{\overrightarrow{q_{obs}}} - \overrightarrow{q_i}}{\left\| \widetilde{\overrightarrow{q_{obs}}} - \overrightarrow{q_i} \right\|^2} \times \mathbb{1}_{\left\| \overrightarrow{q_{obs}} - \overrightarrow{q_i} \right\| < \delta}$ avec $\omega > 0$, $\widetilde{\overrightarrow{q_{obs}}} = \overrightarrow{q_{obs}} - \frac{\widetilde{\overrightarrow{q_{obs}}} - \overrightarrow{q_i}}{\left\| \widetilde{\overrightarrow{q_{obs}}} - \overrightarrow{q_i} \right\|} \times r_{obs}$, δ une distance arbitrairement choisie et r_{obs} le rayon de l'obstacle.
- L'introduction de δ a pour but d'éviter une perturbation du mouvement.

Simulation avec obstacle

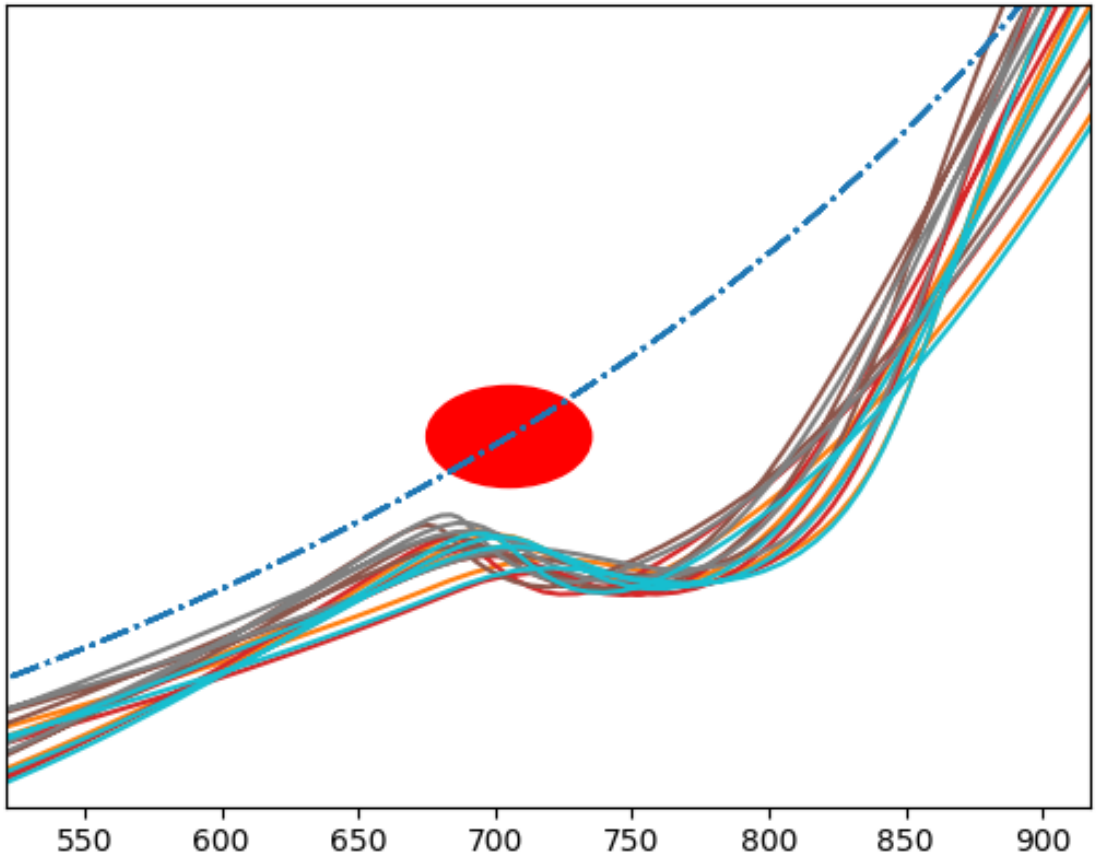

Figure – Simulation avec vitesse circulaire

Simulation avec obstacle

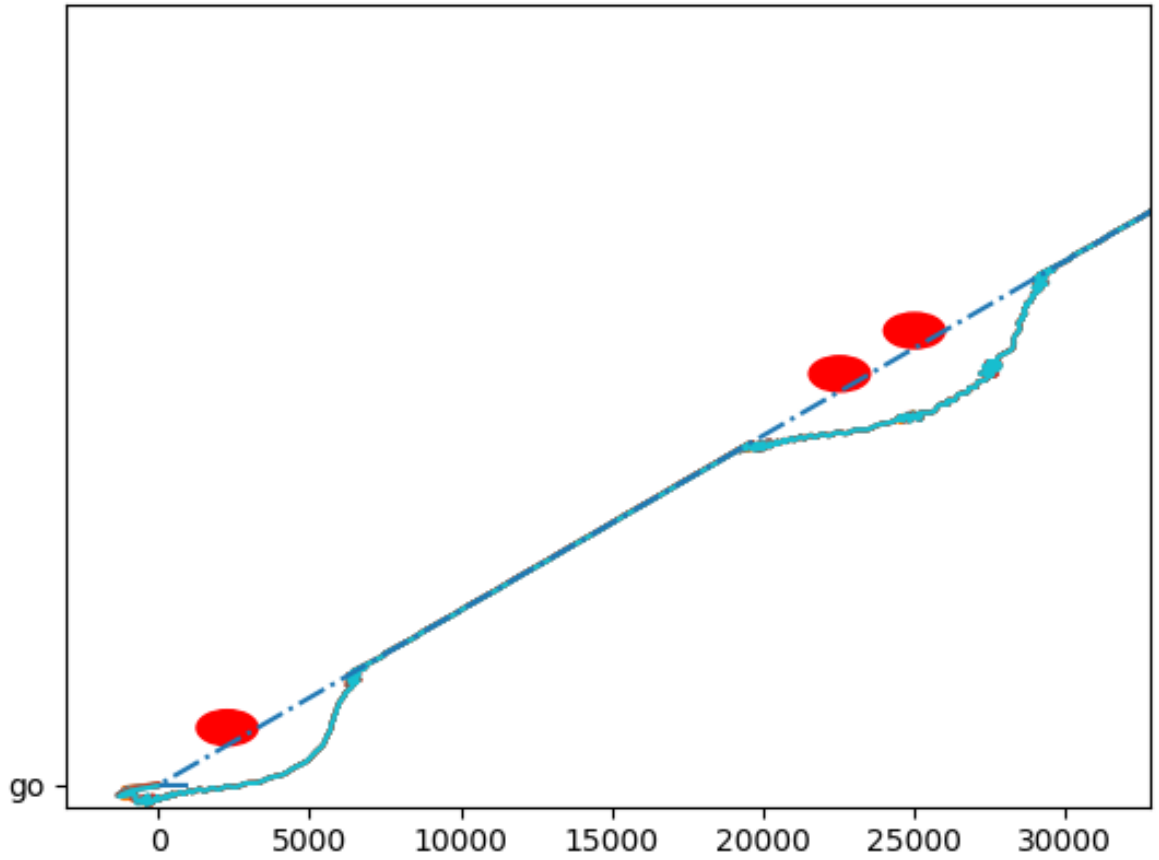

Figure – Simulation avec vitesse rectiligne uniforme

Simulation avec obstacle

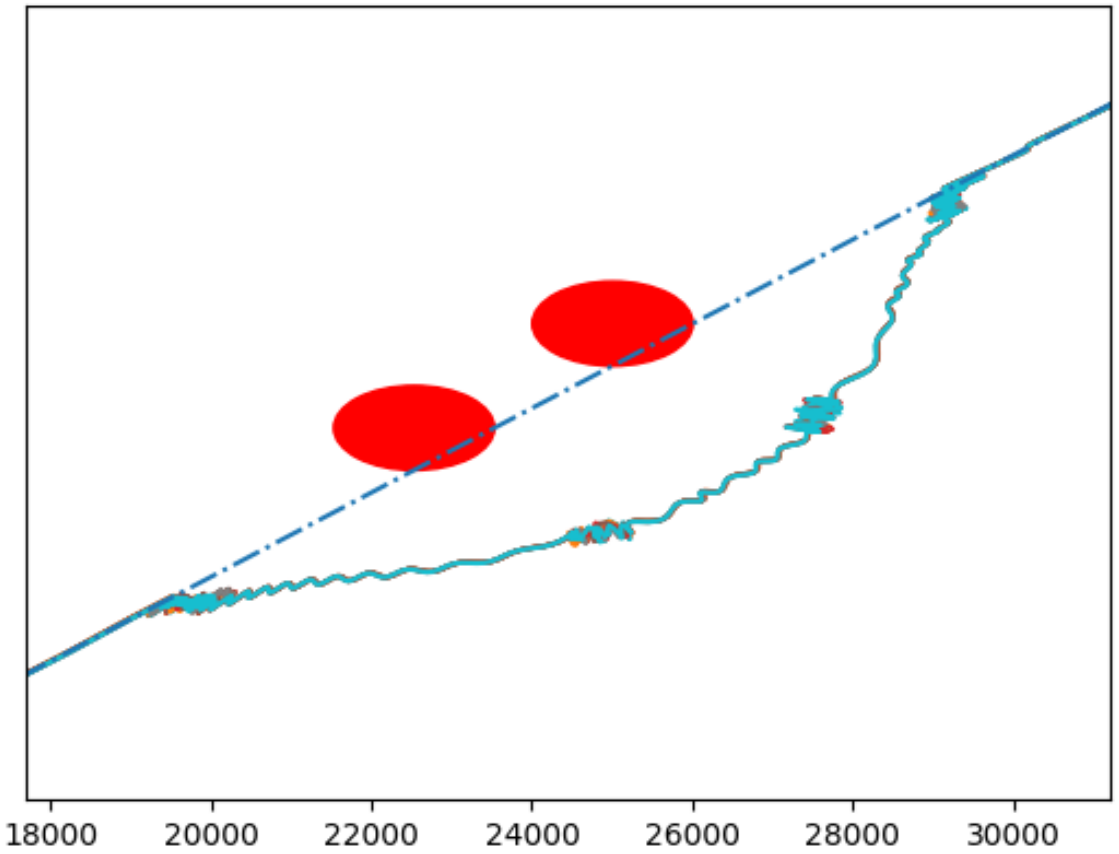

Figure – Zoom sur la figure précédente

Simulation avec obstacle

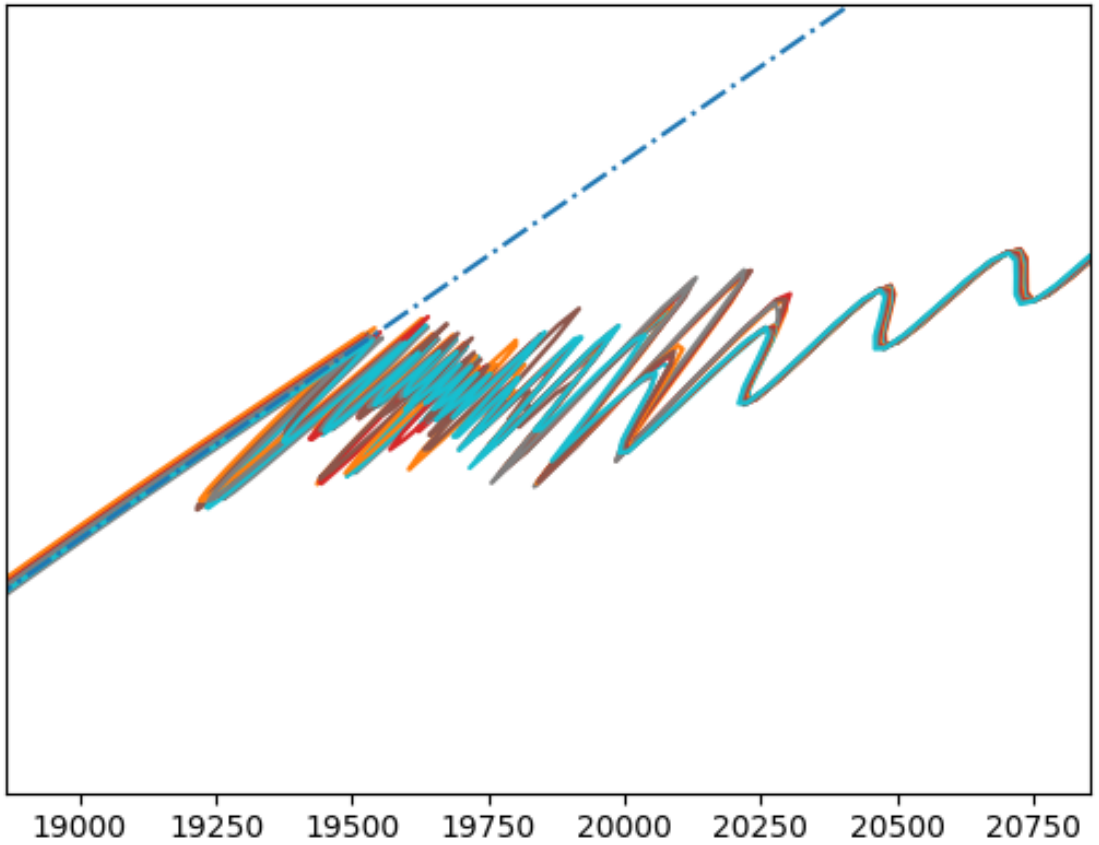

Figure – Zoom sur la figure précédente

- Oscillations dues à 2 causes.

Peturbations aléatoires

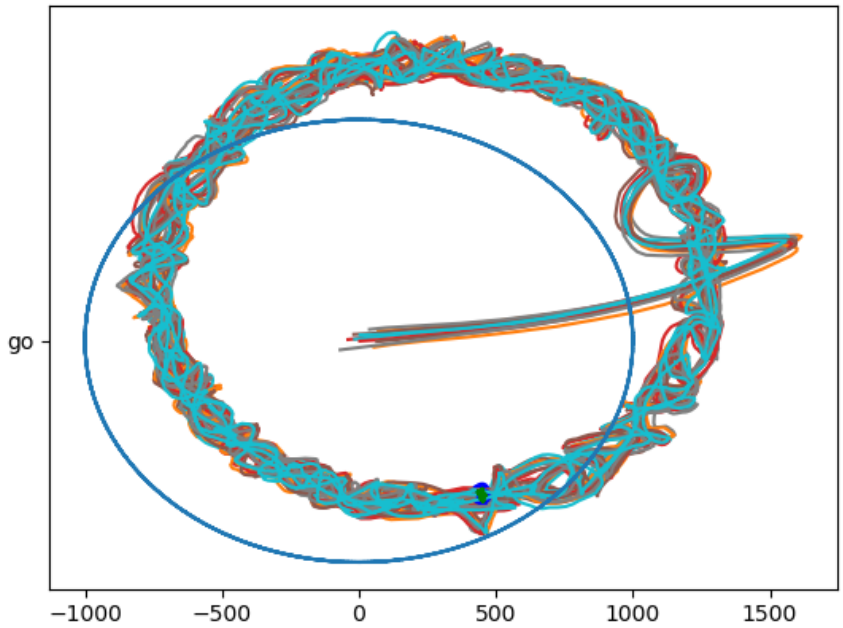

Figure – Perturbations d'amplitude 10.

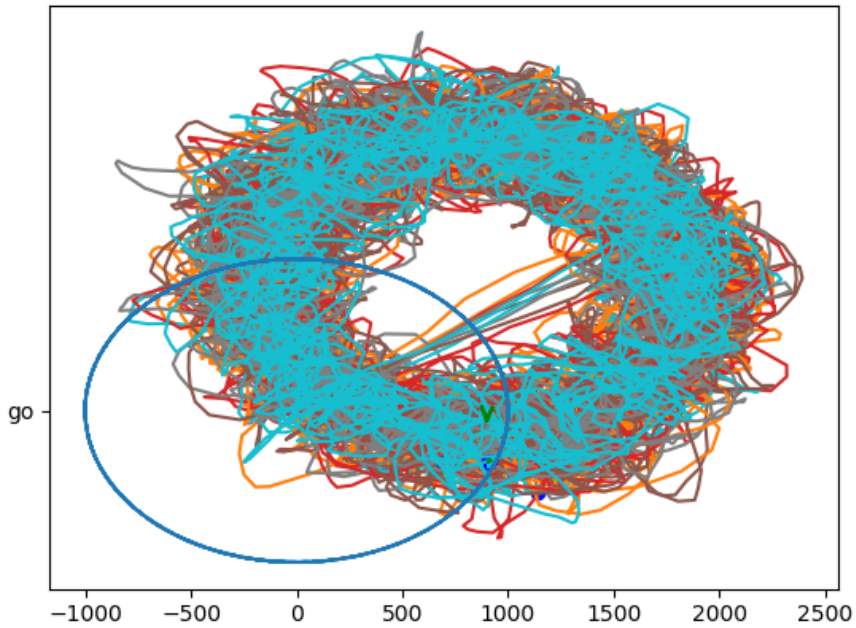

Figure – Perturbations d'amplitude 100.

- Implémentée par l'ajout d'un vecteur `[(2*random()-1)*amplitude,(2*random()-1)*amplitude]` à la vitesse à chaque itération.
- Trajectoire transaltée
- L'allure reste correcte pour une amplitude raisonnable.

Vent

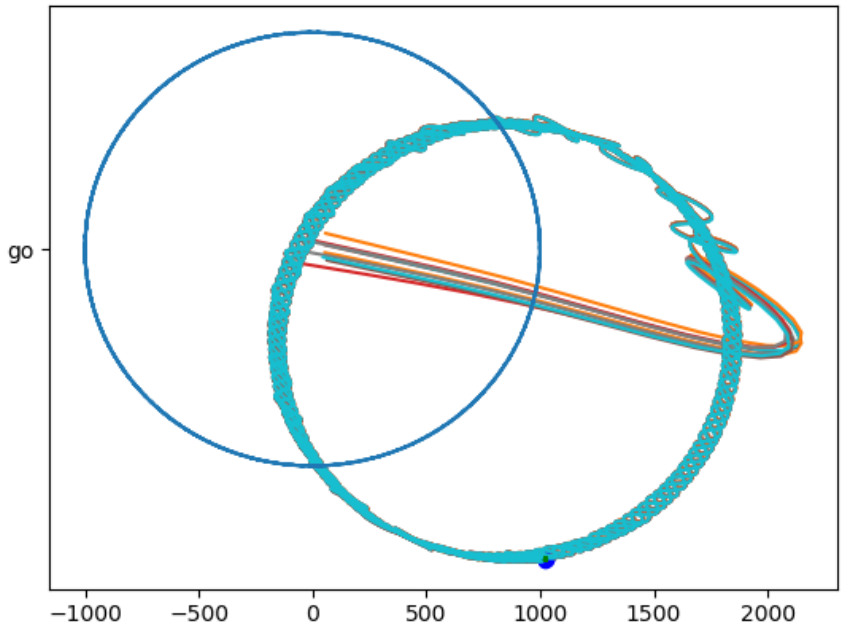

Figure – Vent de vecteur `[40,-10]`

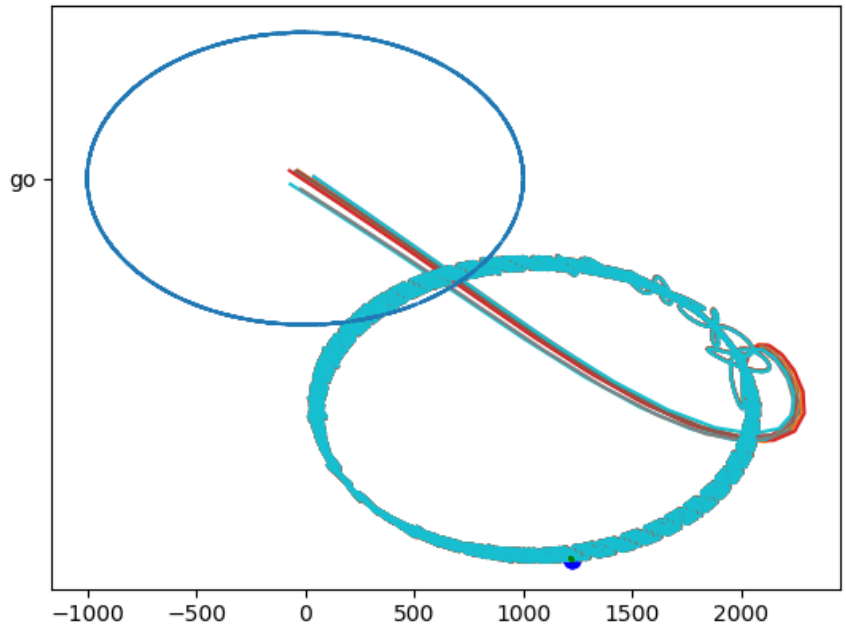

Figure – Vent de vecteur `[100,-150]`

- Implémentation par ajout d'un vecteur constant `[x,y]` à la vitesse à chaque itération.
- Même remarques qu'avant.
- Solution éventuelle : adaptation de la vitesse de la cible à celle des agents.