La régulation d'un BlueBoat par apprentissage

Rapport - Stage 2A

Auteur :
Matheus BENIGNO DA SILVA

18 juillet 2026

Table des matières

1 Introduction **3**

2 Présentation du système **3**
2.1 Le BlueBoat 3
2.2 Objectif de contrôle 4
2.3 Variables utilisées par le contrôleur 4

3 Modèle de simulation **5**
3.1 État du bateau 5
3.2 Équations dynamiques 5
3.3 Intégration numérique 6

4 Contrôleur par champ vectoriel **6**
4.1 Principe général 6
4.2 Définition du champ 7
4.3 Orientation désirée 7
4.4 Commande différentielle 8

5 Génération de la base de données **8**
5.1 Principe 8
5.2 Structure des données 9
5.3 Taille de la base de données 9

6 Réseau de neurones **9**
6.1 Architecture 9
6.2 Propagation avant 10
6.3 Remarque sur les versions avec et sans biais 11

7 Apprentissage supervisé **12**
7.1 Formulation du problème 12
7.2 Fonction de coût 12
7.3 Optimisation 12
7.4 Exportation des poids 13

8 Validation en simulation **13**
8.1 Comparaison des trajectoires 13
8.2 Analyse de l'erreur de prédiction 14
8.3 Discussion sur la validation 15

9 Implémentation sur le BlueBoat réel **16**
9.1 Architecture logicielle 16
9.2 Communication MAVLink 16
9.3 Lecture de la position GPS 17
9.4 Conversion vers le repère local 17
9.5 Conversion de l'orientation 17
9.6 Commande par RC override 18
9.7 Sécurité pendant les essais 18

10 Essais réels en mer **18**
10.1 Essai avec le contrôleur par champ vectoriel 19
10.2 Essai avec le contrôleur neuronal 19
10.3 Généralisation hors de la zone d'apprentissage 19
10.4 Discussion des résultats expérimentaux 20

11 Discussion **20**

12 Conclusion **21**

1 Introduction

L'objectif de ce stage a été d'étudier, de simuler et de tester une méthode de régulation pour un BlueBoat autonome. Le problème considéré consiste à faire tourner le bateau autour d'une bouée en maintenant un rayon désiré. Cette trajectoire circulaire est appelée *carousel*. Dans le cadre du Blue Carousel Challenge, le bateau doit recevoir sa position et son orientation, puis produire deux commandes correspondant aux deux propulseurs du robot [1].

Le contrôle des véhicules marins autonomes est généralement étudié dans le cadre des systèmes de guidage, navigation et commande. Les approches classiques reposent souvent sur une modélisation dynamique du bateau, sur une loi de guidage définissant une trajectoire ou une direction à suivre, puis sur une loi de commande permettant de produire les efforts nécessaires sur les actionneurs [3, 4]. Dans le cas d'un bateau de surface à deux propulseurs, ces méthodes peuvent par exemple utiliser une commande de cap, une commande de vitesse ou une stratégie de suivi de trajectoire. Le travail présenté dans ce rapport s'inscrit dans ce contexte, mais adopte une approche plus géométrique : au lieu de définir uniquement une trajectoire de référence, un champ vectoriel est utilisé pour associer à chaque position du plan une direction désirée pour le bateau.

Le travail réalisé s'est organisé autour de deux approches complémentaires. La première approche repose sur un contrôleur géométrique par champ vectoriel. Ce contrôleur fournit une direction désirée en chaque point du plan et permet de générer les commandes des moteurs du bateau. La deuxième approche consiste à entraîner un réseau de neurones afin d'imiter le comportement de ce contrôleur. Le réseau est ensuite exporté sous forme d'une liste de poids afin de pouvoir être utilisé sans phase d'apprentissage pendant les essais.

Le rapport présente d'abord le problème de contrôle et le repère local utilisé. Ensuite, le modèle de simulation du bateau est décrit. Le contrôleur par champ vectoriel est ensuite détaillé, puis la génération des données, l'architecture du réseau de neurones et la procédure d'apprentissage sont présentées. Enfin, la dernière partie décrit l'implémentation sur le BlueBoat réel, notamment la communication MAVLink, la lecture des données GPS et d'attitude, ainsi que l'envoi des commandes aux moteurs par les canaux RC.

2 Présentation du système

2.1 Le BlueBoat

Le BlueBoat est un véhicule autonome de surface équipé de deux propulseurs. La commande du bateau est donc différentielle : une différence entre les deux commandes de propulsion permet de faire tourner le bateau, tandis qu'une commande identique sur les deux propulseurs produit principalement une translation vers l'avant.

On note les deux commandes moteur :

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1}$$

où u_1 correspond au propulseur gauche et u_2 au propulseur droit. Dans la suite du rapport, ces commandes sont aussi notées u_{left} et u_{right}.



FIGURE 1 – BlueBoat utilisé.

2.2 Objectif de contrôle

L'objectif est de faire suivre au bateau une trajectoire circulaire autour d'une bouée. La bouée est placée à l'origine du repère local :

$$B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2}$$

Le rayon désiré de la trajectoire est noté r_d. La distance du bateau à la bouée est :

$$R = \sqrt{(x - x_b)^2 + (y - y_b)^2} \tag{3}$$

Comme la bouée est placée à l'origine dans le repère local, cette expression devient :

$$R = \sqrt{x^2 + y^2} \tag{4}$$

Le but du contrôleur est donc double : faire converger R vers r_d et maintenir un mouvement tangentiel autour de la bouée.

2.3 Variables utilisées par le contrôleur

Le contrôleur utilise principalement la pose du bateau :

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5}$$

où x et y sont les coordonnées du bateau dans le repère local, et θ est son angle d'orientation dans ce même repère. L'angle $\theta = 0$ correspond à une orientation selon l'axe local.

Pour le réseau de neurones, l'angle n'est pas directement donné sous la forme θ. On utilise plutôt :

$$\cos(\theta), \quad \sin(\theta) \tag{6}$$

Cette représentation évite la discontinuité de l'angle entre $-\pi$ et π. Elle permet également de respecter la périodicité de l'orientation. En effet, deux angles séparés de 2π correspondent physiquement au même cap :

$$\theta \equiv \theta + 2\pi \tag{7}$$

Or, si l'angle θ était donné directement en entrée du réseau, celui-ci pourrait apprendre des comportements différents pour deux orientations équivalentes, par exemple pour θ et $\theta + 2\pi$. En utilisant $\cos(\theta)$ et $\sin(\theta)$, ces deux angles sont représentés de la même manière :

$$\cos(\theta + 2\pi) = \cos(\theta) \tag{8}$$

$$\sin(\theta + 2\pi) = \sin(\theta) \tag{9}$$

Cette représentation est donc plus adaptée pour l'apprentissage, car elle donne au réseau une description continue et périodique de l'orientation du bateau.

3 Modèle de simulation

3.1 État du bateau

Avant les essais réels, le contrôleur et le réseau de neurones ont été testés dans un simulateur simple écrit en Python. Le modèle utilisé représente la position, l'orientation, les vitesses et la dynamique des propulseurs.

L'état du bateau est défini par :

$$X = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega & w_1 & w_2 \end{bmatrix}^T \tag{10}$$

où :

— x et y sont les coordonnées du bateau dans le repère local ;
— θ est l'orientation du bateau ;
— v_x et v_y sont les vitesses dans le repère du bateau ;
— ω est la vitesse angulaire ;
— w_1 et w_2 représentent la dynamique des deux propulseurs.

3.2 Équations dynamiques

La cinématique du bateau est donnée par :

$$\dot{x} = v_x \cos(\theta) - v_y \sin(\theta) \tag{11}$$

$$\dot{y} = v_x \sin(\theta) + v_y \cos(\theta) \tag{12}$$

$$\dot{\theta} = \omega \tag{13}$$

Les équations utilisées pour les vitesses sont :

$$\dot{v}_x = \omega v_y - p_5 v_x |v_x| + p_3(w_1|w_1| + w_2|w_2|) \tag{14}$$

$$\dot{v}_y = -\omega v_x - p_6 v_y |v_y| \tag{15}$$

$$\dot{\omega} = -p_7 \omega|\omega| + p_4(w_2|w_2| - w_1|w_1|) \tag{16}$$

Enfin, la dynamique des propulseurs est décrite par :

$$\dot{w}_1 = -p_1 w_1|w_1| + p_2 u_1 \tag{17}$$

$$\dot{w}_2 = -p_1 w_2|w_2| + p_2 u_2 \tag{18}$$

Ce modèle ne cherche pas à représenter parfaitement toutes les dynamiques hydrodynamiques du BlueBoat réel. Il sert surtout à tester la cohérence du contrôleur et à comparer la trajectoire obtenue avec le contrôleur par champ vectoriel et celle obtenue avec le réseau de neurones.

3.3 Intégration numérique

La simulation est réalisée avec une intégration d'Euler explicite :

$$x_{k+1} = x_k + \Delta t f(x_k, u_k) \tag{19}$$

avec un pas de temps Δt. Cette méthode est simple et suffisante pour obtenir une première validation du comportement du système.

4 Contrôleur par champ vectoriel

4.1 Principe général

Le contrôleur par champ vectoriel définit, en chaque position du plan, une direction désirée pour le bateau. L'idée générale consiste à guider le robot de manière à ce que son mouvement soit cohérent avec un champ vectoriel défini dans l'espace de travail. Cette approche s'inscrit dans les méthodes de guidage des robots mobiles, où l'on cherche à faire suivre au système une direction imposée par une loi géométrique [2].

Dans le cas du BlueBoat, cette direction combine deux effets :

— une composante tangentielle, qui permet au bateau de tourner autour de la bouée ;
— une composante radiale, qui corrige la distance du bateau par rapport au rayon désiré.

Ainsi, lorsque le bateau est trop loin de la bouée, le champ l'oriente vers l'intérieur. Lorsqu'il est trop proche, le champ l'oriente vers l'extérieur. Lorsque le bateau est proche du rayon désiré, le champ devient principalement tangentiel.

4.2 Définition du champ

On considère la position relative du bateau par rapport à la bouée :

$$p_x = x - x_b \tag{20}$$

$$p_y = y - y_b \tag{21}$$

La distance à la bouée est :

$$R = \sqrt{p_x^2 + p_y^2} \tag{22}$$

On définit ensuite l'angle polaire :

$$\alpha = \arctan 2(p_y, p_x) \tag{23}$$

Les variables normalisées utilisées pour le champ sont :

$$X_1 = \frac{R}{r_d} \cos(\alpha) \tag{24}$$

$$X_2 = \frac{R}{r_d} \sin(\alpha) \tag{25}$$

Le champ vectoriel est alors donné par :

$$F_x = -X_1^3 - X_1 X_2^2 + X_1 - X_2 \tag{26}$$

$$F_y = -X_2^3 - X_1^2 X_2 + X_1 + X_2 \tag{27}$$

Le vecteur $F = [F_x, F_y]^T$ donne la direction que le bateau doit suivre pour converger vers le mouvement circulaire autour de la bouée.

4.3 Orientation désirée

À partir du champ vectoriel, on calcule l'orientation désirée :

$$\theta_d = \arctan 2(F_y, F_x) \tag{28}$$

L'erreur angulaire entre l'orientation désirée et l'orientation réelle est :

$$e_\theta = \theta_d - \theta \tag{29}$$

Comme les angles sont définis modulo 2π, l'erreur est ramenée dans l'intervalle $[-\pi, \pi]$ avec la fonction :

$$\text{sawtooth}(a) = (a + \pi) \mod 2\pi - \pi \tag{30}$$

Ainsi, on utilise finalement :

$$e_\theta = \text{sawtooth}(\theta_d - \theta) \tag{31}$$

4.4 Commande différentielle

Le contrôleur calcule ensuite une correction différentielle à partir de l'erreur angulaire :

$$u_{\text{diff}} = K_\theta e_\theta \tag{32}$$

Dans cette version, la commande est proportionnelle à l'erreur d'orientation. Le gain K_θ permet de régler l'intensité de la correction appliquée pour aligner le bateau avec la direction donnée par le champ vectoriel.

Dans les scripts de test, une forme plus générale du contrôleur a également été prévue :

$$u_{\text{diff}} = K_\theta e_\theta - K_D \omega \tag{33}$$

où ω est la vitesse angulaire du bateau. Cependant, dans les essais présentés dans ce rapport, le terme dérivatif n'a pas été utilisé, car le gain K_D a été fixé à zéro. Cette forme proportionnelle-dérivative constitue donc principalement une piste d'amélioration pour la suite du travail.

Les commandes des deux propulseurs sont alors :

$$u_{\text{left}} = U_0 - u_{\text{diff}} \tag{34}$$

$$u_{\text{right}} = U_0 + u_{\text{diff}} \tag{35}$$

Ces commandes sont ensuite saturées dans l'intervalle autorisé :

$$u_{\text{left}}, u_{\text{right}} \in [-U_{\max}, U_{\max}] \tag{36}$$

Le terme U_0 correspond à une commande d'avance moyenne, tandis que u_{diff} crée la rotation du bateau.

5 Génération de la base de données

5.1 Principe

Le réseau de neurones n'a pas été entraîné directement sur le bateau réel. Il a d'abord été entraîné par apprentissage supervisé à partir d'exemples générés par le contrôleur par champ vectoriel. Le contrôleur joue donc le rôle d'un expert qui fournit la commande attendue pour un grand nombre de positions et d'orientations.

Pour chaque échantillon, on choisit une position (x, y) et un angle θ. On calcule ensuite les commandes du contrôleur :

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = C_{VF}(x, y, \theta) \tag{37}$$

où C_{VF} désigne le contrôleur par champ vectoriel.

5.2 Structure des données

Chaque ligne de la base de données est constituée de quatre entrées et de deux sorties :

$$[x, y, \cos(\theta), \sin(\theta), u_1, u_2] \tag{38}$$

Les variables x et y décrivent la position du bateau, tandis que $\cos(\theta)$ et $\sin(\theta)$ décrivent son orientation. Les sorties u_1 et u_2 correspondent aux commandes que le réseau doit apprendre à reproduire.

Cette base de données permet donc de transformer le problème de contrôle en un problème d'approximation de fonction :

$$f_{NN}(x, y, \cos\theta, \sin\theta) \approx C_{VF}(x, y, \theta) \tag{39}$$

5.3 Taille de la base de données

La base de données utilisée pour l'apprentissage a été générée à partir d'une grille régulière de positions et d'orientations. Les positions x et y ont été échantillonnées dans l'intervalle $[-10, 10]$ m, tandis que l'angle θ a été échantillonné dans l'intervalle $[-\pi, \pi]$.

Le nombre total d'exemples générés est donné par :

$$N = N_X \times N_Y \times N_\theta \tag{40}$$

Dans la configuration utilisée, les valeurs étaient :

$$N_X = 80, \quad N_Y = 80, \quad N_\theta = 180 \tag{41}$$

Le nombre total d'exemples était donc :

$$N = 80 \times 80 \times 180 = 1\,152\,000 \tag{42}$$

La base de données contenait ainsi environ 1,15 million d'exemples. Chaque exemple associe une configuration du bateau à la commande produite par le contrôleur par champ vectoriel.

6 Réseau de neurones

6.1 Architecture

L'architecture imposée pour le contrôleur neuronal du Blue Carousel Challenge est composée de quatre entrées, deux couches cachées de huit neurones et deux sorties [1]. Les entrées sont :

$$X_{NN} = \begin{bmatrix} x \\ y \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{43}$$

Les sorties sont :

$$Y_{NN} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{44}$$

La structure utilisée est donc présentée dans la figure 2.

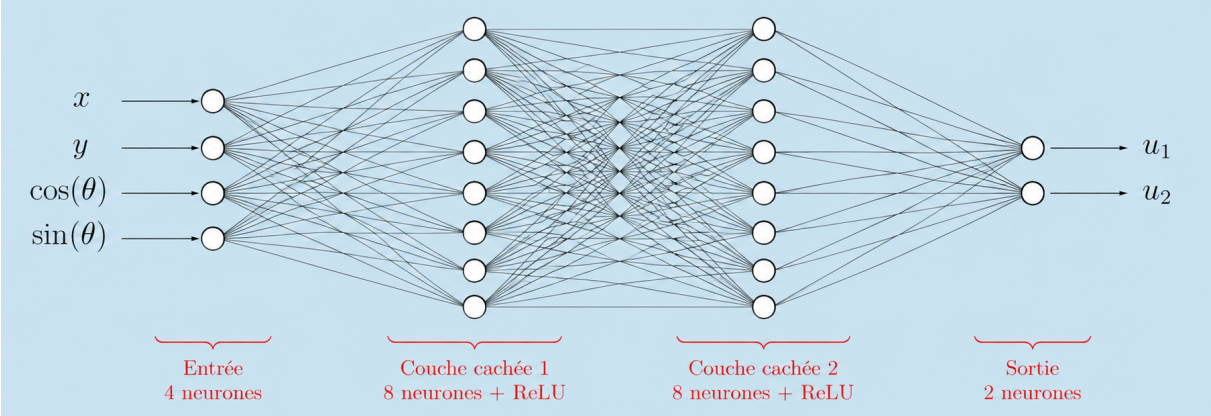


FIGURE 2 – Architecture du réseau de neurones utilisé pour le contrôle du BlueBoat.

6.2 Propagation avant

Dans la version sans biais, la propagation avant peut être écrite sous la forme :

$$z_1 = X_{NN} W_1 \tag{45}$$

$$h_1 = \text{ReLU}(z_1) \tag{46}$$

$$z_2 = h_1 W_2 \tag{47}$$

$$h_2 = \text{ReLU}(z_2) \tag{48}$$

$$\hat{u} = h_2 W_3 \tag{49}$$

avec la fonction d'activation :

$$\text{ReLU}(x) = \max(0, x) \tag{50}$$

Les dimensions des matrices sont :

$$W_1 \in \mathbb{R}^{4\times 8}, \quad W_2 \in \mathbb{R}^{8\times 8}, \quad W_3 \in \mathbb{R}^{8\times 2} \tag{51}$$

Le nombre total de poids est donc :

$$4 \times 8 + 8 \times 8 + 8 \times 2 = 112 \tag{52}$$

Ce nombre de paramètres correspond au format attendu pour le contrôleur neuronal du défi [1].

6.3 Remarque sur les versions avec et sans biais

Pendant le développement, deux versions du contrôleur neuronal ont été considérées afin de permettre plusieurs possibilités d'implémentation sur le BlueBoat. Les scripts utilisés pour les tests réels ont donc été adaptés pour pouvoir charger et exécuter une architecture sans biais ou une architecture avec biais.

Dans ce rapport, la version principale présentée reste cependant la version sans biais, car c'est cette architecture qui a été utilisée pour l'entraînement du réseau de neurones.

Une deuxième version, avec biais, peut également être utilisée dans les scripts de contrôle. Dans ce cas, chaque couche linéaire possède un vecteur de biais supplémentaire. La propagation avant devient alors :

$$z_1 = X_{NN} W_1 + b_1 \tag{53}$$

$$h_1 = \mathrm{ReLU}(z_1) \tag{54}$$

$$z_2 = h_1 W_2 + b_2 \tag{55}$$

$$h_2 = \mathrm{ReLU}(z_2) \tag{56}$$

$$\hat{u} = h_2 W_3 + b_3 \tag{57}$$

Les dimensions des matrices et des vecteurs de biais sont alors :

$$W_1 \in \mathbb{R}^{4\times 8}, \quad b_1 \in \mathbb{R}^8 \tag{58}$$

$$W_2 \in \mathbb{R}^{8\times 8}, \quad b_2 \in \mathbb{R}^8 \tag{59}$$

$$W_3 \in \mathbb{R}^{8\times 2}, \quad b_3 \in \mathbb{R}^2 \tag{60}$$

Le nombre total de paramètres de la version avec biais est donc :

$$4 \times 8 + 8 + 8 \times 8 + 8 + 8 \times 2 + 2 = 130 \tag{61}$$

Ainsi, la différence entre les deux versions ne concerne pas la structure générale du réseau, qui reste une architecture $4 - 8 - 8 - 2$, mais uniquement la présence ou non des vecteurs de biais dans les couches linéaires. Pour conserver une cohérence avec la base de données utilisée et avec l'entraînement réalisé, les résultats présentés dans la suite du rapport correspondent à la version sans biais.

7 Apprentissage supervisé

7.1 Formulation du problème

Le réseau de neurones est entraîné pour reproduire la commande générée par le contrôleur par champ vectoriel. Pour chaque échantillon i de la base de données, on dispose d'une entrée :

$$X_i = \begin{bmatrix} x_i & y_i & \cos(\theta_i) & \sin(\theta_i) \end{bmatrix} \tag{62}$$

et d'une sortie cible :

$$U_i = \begin{bmatrix} u_{1,i} & u_{2,i} \end{bmatrix} \tag{63}$$

L'objectif est de trouver les paramètres du réseau qui minimisent l'écart entre la sortie prédite $\hat{U}_i$ et la sortie cible U_i.

7.2 Fonction de coût

La fonction de coût utilisée est l'erreur quadratique moyenne :

$$J = \frac{1}{N} \sum_{i=1}^{N} \left\| \hat{U}_i - U_i \right\|^2 \tag{64}$$

avec :

$$\hat{U}_i = f_{NN}(X_i) \tag{65}$$

La minimisation de cette fonction pousse le réseau à produire des commandes proches de celles du contrôleur expert.

7.3 Optimisation

L'optimisation des paramètres du réseau est effectuée avec l'algorithme Adam. Le réseau a été entraîné pendant 10 000 époques. À chaque époque d'entraînement, les étapes suivantes sont réalisées :

1. calcul de la sortie prédite par le réseau ;
2. calcul de l'erreur quadratique moyenne ;
3. rétropropagation du gradient ;
4. mise à jour des poids du réseau.

L'entraînement de la version utilisée dans ce rapport a été réalisé avec une base de données de 1,15 million d'exemples. Pour cette configuration, la durée totale de l'entraînement était d'environ 12 minutes.

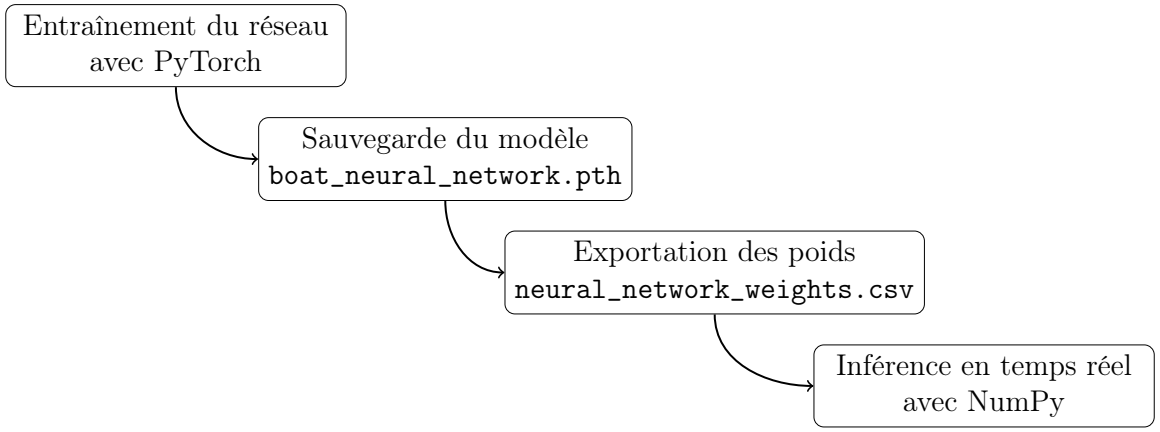
Cette durée dépend de plusieurs facteurs, notamment la taille de la base de données, le nombre d'époques, la puissance de calcul disponible et l'utilisation ou non d'un GPU.

À la fin de l'entraînement, le modèle est sauvegardé, puis les poids sont exportés dans un fichier CSV afin de pouvoir être relus par un programme de contrôle indépendant de PyTorch.

7.4 Exportation des poids

L'exportation des poids est une étape importante pour le passage de la phase d'apprentissage à la phase de test. En effet, le réseau est entraîné avec PyTorch, mais l'inférence peut ensuite être réalisée avec NumPy à partir d'une simple liste de poids.

Le processus est le suivant :



Cette séparation entre entraînement et utilisation permet de rendre le contrôleur plus simple à intégrer dans les scripts de test réel.

8 Validation en simulation

La validation en simulation a pour objectif de vérifier si le réseau de neurones est capable de reproduire le comportement du contrôleur par champ vectoriel. Pour cela, deux types de résultats sont analysés : d'abord les trajectoires obtenues avec les deux approches, puis la répartition spatiale de l'erreur de prédiction du réseau.

8.1 Comparaison des trajectoires

Une première comparaison est réalisée entre la trajectoire obtenue avec le contrôleur par champ vectoriel et celle obtenue avec le réseau de neurones. Les deux simulations utilisent le même modèle dynamique du bateau et les mêmes conditions initiales. La différence principale vient donc de la manière dont les commandes u_1 et u_2 sont calculées.

Dans le premier cas, les commandes sont calculées directement à partir des équations du champ vectoriel. Dans le second cas, elles sont obtenues par la propagation avant du réseau de neurones entraîné.

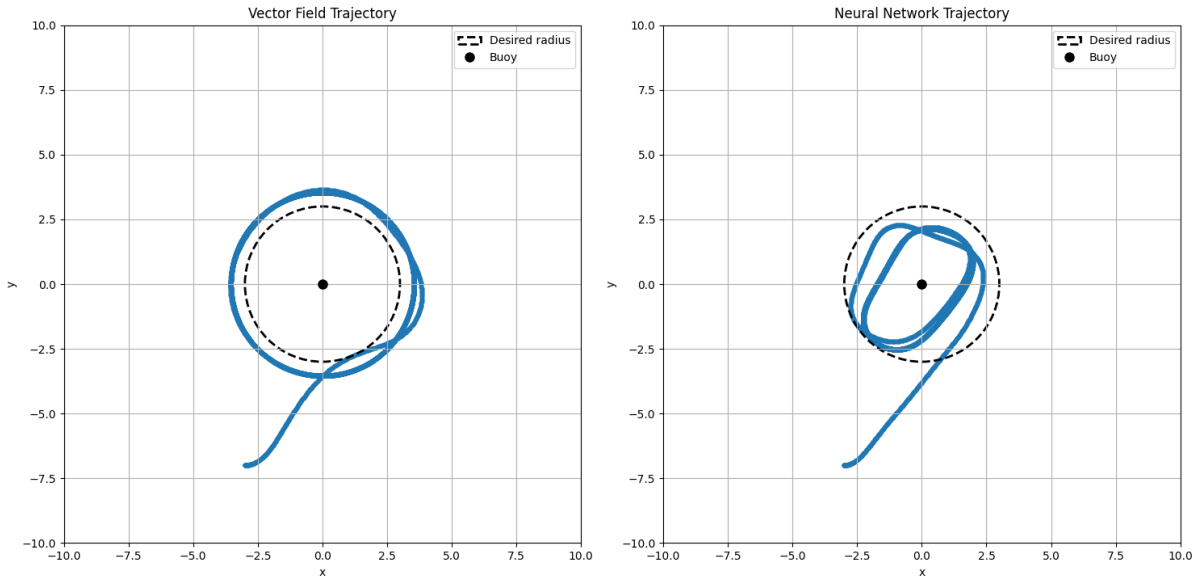


FIGURE 3 – Comparaison entre la trajectoire simulée avec le contrôleur par champ vectoriel et la trajectoire simulée avec le réseau de neurones.

La figure 3 montre que le contrôleur par champ vectoriel permet au bateau de rejoindre une trajectoire proche du cercle désiré autour de la bouée. La trajectoire obtenue avec le réseau de neurones présente également une convergence vers la zone du rayon désiré, mais elle est plus interne et moins circulaire que celle obtenue avec le contrôleur par champ vectoriel. Cette différence est attendue, car le réseau ne calcule pas explicitement le champ vectoriel : il approxime la commande du contrôleur à partir des exemples appris pendant l'entraînement.

Ainsi, la simulation montre que le réseau parvient à reproduire le comportement global du contrôleur expert, même si sa trajectoire n'est pas exactement identique à celle du contrôleur de référence.

8.2 Analyse de l'erreur de prédiction

Pour quantifier la qualité de l'apprentissage, on calcule l'erreur entre la commande prédite par le réseau de neurones et la commande cible donnée par le contrôleur par champ vectoriel. Pour chaque point de la base de données, cette erreur est définie par :

$$e_u = \begin{bmatrix} \hat{u}_1 - u_1 \\ \hat{u}_2 - u_2 \end{bmatrix} \tag{66}$$

où $\hat{u}_1$ et $\hat{u}_2$ sont les commandes prédites par le réseau, tandis que u_1 et u_2 sont les commandes générées par le contrôleur par champ vectoriel.

La norme de cette erreur est donnée par :

$$\|e_u\| = \sqrt{(\hat{u}_1 - u_1)^2 + (\hat{u}_2 - u_2)^2} \tag{67}$$

Cette erreur peut être représentée sous forme de carte dans le plan (x, y) afin d'identifier les zones du bassin où le réseau approxime le mieux le contrôleur.

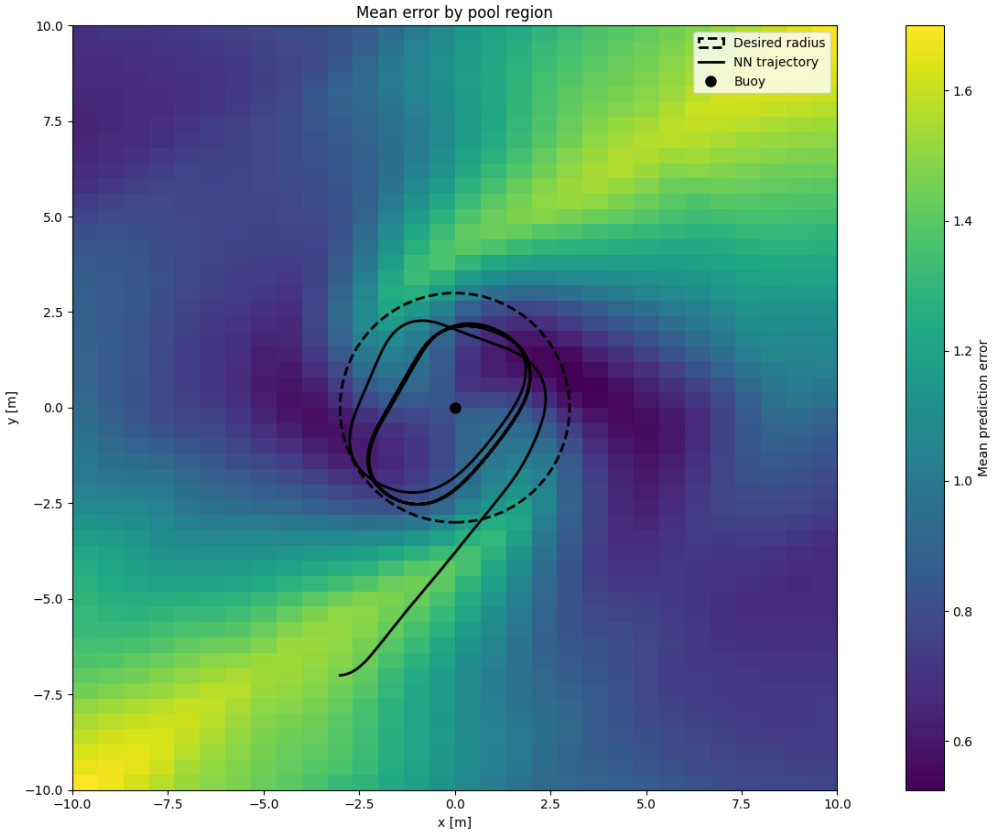


FIGURE 4 – Carte de l'erreur moyenne de prédiction du réseau de neurones en fonction de la position du bateau dans le bassin.

La figure 4 présente l'erreur moyenne de prédiction dans différentes régions du bassin. Les couleurs correspondent à l'erreur moyenne entre les commandes calculées par le réseau et celles données par le contrôleur expert. La trajectoire du réseau est également superposée à cette carte, ainsi que le cercle correspondant au rayon désiré autour de la bouée.

On observe que l'erreur n'est pas uniforme dans tout le bassin. Certaines zones présentent une erreur plus faible, tandis que d'autres régions, notamment plus éloignées de la trajectoire principale, présentent une erreur plus importante. Cela peut s'expliquer par le fait que le réseau de neurones utilise une architecture très compacte, avec seulement deux couches cachées de huit neurones. Il ne peut donc pas reproduire parfaitement toute la loi de commande du champ vectoriel sur l'ensemble de l'espace d'entrée.

8.3 Discussion sur la validation

Les résultats de simulation montrent que le réseau de neurones reproduit correctement le comportement général du contrôleur par champ vectoriel. Le bateau rejoint la zone du cercle désiré et reste autour de la bouée, ce qui confirme que le réseau a bien appris une approximation utilisable de la loi de commande.

Cependant, la trajectoire obtenue avec le réseau n'est pas exactement identique à celle du contrôleur expert. Cette différence met en évidence les limites de l'approximation neuronale. Le réseau utilisé est volontairement petit afin de pouvoir être facilement exporté et exécuté dans les scripts de contrôle. Ce choix rend l'implémentation plus simple, mais

limite aussi la précision de l'approximation.

Un autre point observé pendant les simulations est la capacité du réseau à converger vers le cercle désiré même lorsque la position initiale du bateau se trouve en dehors de la région utilisée pour l'entraînement. Dans certains essais, malgré une condition initiale extérieure au domaine d'apprentissage, le bateau a tout de même été attiré vers la zone du rayon désiré et a fini par former un mouvement de type carrousel autour de la bouée. Ce résultat montre que le réseau a appris une approximation cohérente de la loi de commande, et qu'il possède une certaine capacité de généralisation au-delà des exemples directement présents dans la base de données.

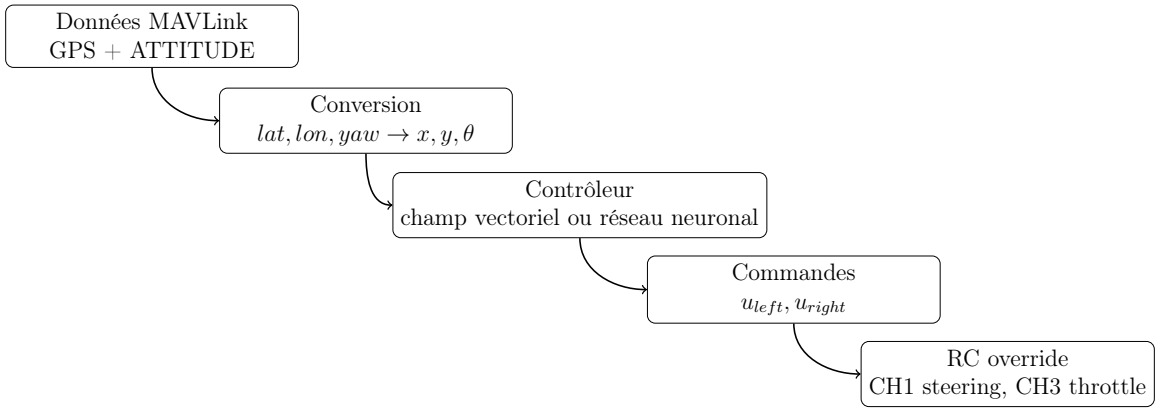
9 Implémentation sur le BlueBoat réel

9.1 Architecture logicielle

L'implémentation réelle est organisée en plusieurs modules Python. Cette organisation permet de séparer les responsabilités du programme :

— un fichier de configuration contenant les paramètres du test ;
— un module de communication MAVLink ;
— un module de conversion des coordonnées GPS vers le repère local ;
— un module contenant le contrôleur par champ vectoriel ;
— un module permettant de charger et d'exécuter le réseau de neurones ;
— des scripts principaux pour tester le contrôleur ou le réseau sur le bateau.

L'architecture générale est résumée par le schéma suivant :



9.2 Communication MAVLink

La communication avec le BlueBoat est réalisée avec le protocole MAVLink. Le script commence par établir une connexion avec le véhicule, puis attend la réception d'un *heartbeat*. Ce message permet de vérifier que la communication avec l'autopilote est active.

Après la connexion, le programme peut placer le bateau dans le mode souhaité, armer le véhicule, configurer les flux de messages et envoyer les commandes. Les messages utilisés pour le contrôle sont principalement :

— `GLOBAL_POSITION_INT`, pour obtenir la position GPS ;
— `ATTITUDE`, pour obtenir l'angle de lacet du bateau ;

— `AHRS`, utilisé notamment pour accéder à une information de vitesse angulaire lorsque nécessaire.

9.3 Lecture de la position GPS

La position GPS est reçue avec le message `GLOBAL_POSITION_INT`. Dans MAVLink, la latitude et la longitude sont envoyées sous forme d'entiers. Elles sont donc converties en degrés par :

$$lat = \frac{msg.lat}{10^7} \tag{68}$$

$$lon = \frac{msg.lon}{10^7} \tag{69}$$

Avant d'appliquer cette approximation locale, les latitudes et longitudes sont converties de degrés en radians.

Ces coordonnées sont ensuite transformées en coordonnées locales x, y en utilisant un point GPS de référence et une rotation du repère.

9.4 Conversion vers le repère local

À partir d'un point de référence (lat_0, lon_0), les déplacements Est/Nord sont approximés par :

$$x_{east} = R_E \cos(lat_0)(lon - lon_0) \tag{70}$$

$$y_{north} = R_E(lat - lat_0) \tag{71}$$

où R_E est le rayon de la Terre. Une rotation d'angle ψ_0 permet ensuite d'obtenir les coordonnées dans le repère local du bassin :

$$x = x_0 + \cos(\psi_0)x_{east} - \sin(\psi_0)y_{north} \tag{72}$$

$$y = y_0 + \sin(\psi_0)x_{east} + \cos(\psi_0)y_{north} \tag{73}$$

Cette transformation permet d'utiliser sur le bateau réel le même type de variables que dans la simulation et dans l'entraînement du réseau.

9.5 Conversion de l'orientation

Le message `ATTITUDE` fournit le yaw du bateau. Dans la convention MAVLink, le yaw est défini par rapport au Nord. Pour le contrôleur, on souhaite obtenir l'angle θ dans le repère local mathématique.

On commence par construire le vecteur d'orientation dans le repère Est/Nord :

$$v_{east} = \sin(yaw) \tag{74}$$

$$v_{north} = \cos(yaw) \tag{75}$$

Puis on applique la rotation vers le repère local :

$$v_x = \cos(\psi_0) v_{east} - \sin(\psi_0) v_{north} \tag{76}$$

$$v_y = \sin(\psi_0) v_{east} + \cos(\psi_0) v_{north} \tag{77}$$

L'orientation locale est alors :

$$\theta = \arctan 2(v_y, v_x) \tag{78}$$

9.6 Commande par RC override

Pour les essais réels, les commandes calculées par le contrôleur sont envoyées au bateau à travers les canaux RC. Les deux commandes u_{left} et u_{right} sont d'abord converties en une commande moyenne d'avance et une commande de direction :

$$u_{throttle} = \frac{u_{left} + u_{right}}{2} \tag{79}$$

$$u_{steering} = \frac{u_{left} - u_{right}}{2} \tag{80}$$

Ces valeurs sont ensuite converties en signaux PWM. La valeur neutre est de 1500 μs. Une commande positive augmente le PWM, tandis qu'une commande négative le diminue. Pour une commande normalisée dans $[-1, 1]$, la conversion peut être écrite :

$$PWM = PWM_{neutral} + u \cdot (PWM_{max} - PWM_{neutral}) \tag{81}$$

avec saturation entre PWM_{min} et PWM_{max}.
Dans les essais, les canaux utilisés sont :
— CH1 : direction / yaw ;
— CH3 : propulsion / throttle.

9.7 Sécurité pendant les essais

Plusieurs précautions sont prises dans les scripts de test réel. Avant de commencer le contrôle, une commande neutre est envoyée au bateau. Pendant la boucle de contrôle, si les données GPS ou d'attitude ne sont pas valides pendant une durée trop longue, le programme envoie également une commande neutre.

À l'arrêt du programme, le script envoie plusieurs commandes neutres, libère le RC override et désarme le véhicule. Cette logique est importante pour éviter que le bateau continue à recevoir une commande non désirée après l'interruption du programme.

10 Essais réels en mer

Après les validations en simulation, des essais réels ont été réalisés avec le BlueBoat en mer. L'objectif principal de ces essais était de vérifier si les comportements observés en simulation étaient également retrouvés dans des conditions réelles, avec les incertitudes liées au GPS, aux perturbations de l'environnement et à la dynamique réelle du bateau.

Les essais ont été réalisés autour d'une bouée utilisée comme point de référence. Le bateau était placé à une certaine distance de cette bouée, puis le contrôleur était activé afin d'observer si le BlueBoat convergeait vers un mouvement de type carrousel autour du rayon désiré.

10.1 Essai avec le contrôleur par champ vectoriel

Le premier essai réel a été réalisé avec le contrôleur par champ vectoriel. Le bateau a été placé en mer à une certaine distance de la bouée, puis le contrôleur a été lancé. Le comportement observé a montré que le BlueBoat était bien attiré vers la zone du cercle désiré et qu'il commençait à former un mouvement de type carrousel autour de la bouée.

Cependant, la trajectoire suivie n'était pas parfaitement superposée au cercle de référence. Le bateau a eu tendance à suivre une trajectoire plus ouverte, avec un décalage radial par rapport au rayon désiré. Autrement dit, le BlueBoat réalisait bien un mouvement autour de la bouée, mais sur un rayon légèrement différent de celui imposé par le contrôleur.

Ce comportement avait déjà été observé en simulation. Dans les trajectoires simulées avec le contrôleur par champ vectoriel, le bateau converge également vers un mouvement circulaire, mais il existe un écart entre la trajectoire obtenue et le cercle désiré. L'essai réel a donc confirmé qualitativement le comportement observé en simulation. La principale différence est que le décalage constaté en mer semblait plus important que celui obtenu dans le simulateur.

Cette différence peut s'expliquer par plusieurs facteurs : les perturbations de l'environnement marin, les imprécisions du modèle dynamique utilisé en simulation, les délais de communication, ainsi que la précision limitée du GPS.

10.2 Essai avec le contrôleur neuronal

Un second essai a été réalisé avec le contrôleur basé sur le réseau de neurones. Dans ce cas, les commandes des moteurs n'étaient plus calculées directement à partir des équations du champ vectoriel, mais à partir de la propagation avant du réseau entraîné.

Le comportement observé en mer a été satisfaisant. Le bateau a également convergé vers la zone du carrousel autour de la bouée. De plus, la trajectoire obtenue avec le contrôleur neuronal semblait plus fermée que celle obtenue avec le contrôleur par champ vectoriel. Le décalage entre la trajectoire suivie et le cercle désiré était donc plus faible.

Ce résultat est cohérent avec les simulations réalisées précédemment. En simulation, la trajectoire produite par le réseau de neurones était également plus interne et présentait un écart plus réduit par rapport au cercle désiré. Ainsi, même si le réseau de neurones n'a pas été conçu à partir d'une expression analytique du champ, il a permis d'obtenir un comportement réel proche, voire plus satisfaisant dans ce cas particulier, que celui du contrôleur de référence.

10.3 Généralisation hors de la zone d'apprentissage

Un autre point important observé pendant les essais concerne la capacité de généralisation du réseau de neurones. Même lorsque le bateau était initialement placé dans une zone correspondant à des coordonnées en dehors de la région utilisée pour l'entraînement, il a tout de même eu tendance à rejoindre le cercle désiré et à former un mouvement de carrousel autour de la bouée.

Ce comportement avait également été observé en simulation. Il indique que le réseau n'a pas simplement mémorisé les exemples de la base de données, mais qu'il a appris une approximation suffisamment cohérente de la loi de commande pour produire un comportement correct dans certaines situations extérieures au domaine d'apprentissage.

Il faut toutefois rester prudent avec cette interprétation. La généralisation d'un réseau de neurones n'est pas garantie pour toutes les conditions initiales possibles. Ce résultat montre seulement que, dans les essais réalisés, le réseau a conservé un comportement cohérent même en dehors de la région utilisée pour la génération des données.

10.4 Discussion des résultats expérimentaux

Les essais réels ont permis de confirmer plusieurs tendances observées en simulation. Dans les deux cas, le BlueBoat a été attiré vers la zone du rayon désiré et a réalisé un mouvement de type carrousel autour de la bouée. Le contrôleur par champ vectoriel a montré un comportement cohérent avec la simulation, mais avec une trajectoire plus ouverte que le cercle de référence. Le contrôleur neuronal a, quant à lui, produit une trajectoire plus fermée, avec un décalage plus faible par rapport au rayon désiré.

Le décalage observé entre la trajectoire réelle et le cercle désiré peut avoir plusieurs origines. Une première cause possible est liée à la structure du contrôleur utilisé. Dans la version testée, la commande dépend principalement de l'erreur angulaire entre l'orientation désirée et l'orientation du bateau. Un terme dérivatif utilisant la vitesse angulaire du bateau pourrait améliorer l'amortissement du système et réduire l'écart observé.

Une perspective d'amélioration serait donc d'utiliser un contrôleur de type proportionnel-dérivatif pour le calcul de la commande différentielle :

$$u_{diff} = K_\theta e_\theta - K_D \omega \tag{82}$$

où e_θ est l'erreur d'orientation et ω la vitesse angulaire du bateau. Ce terme dérivatif permettrait de tenir compte de la rotation réelle du bateau et pourrait limiter les oscillations ou les dépassements autour du cercle désiré.

Une nouvelle base de données pourrait ensuite être générée à partir de ce contrôleur amélioré. Le réseau de neurones serait alors réentraîné afin d'apprendre le comportement du nouveau contrôleur par champ vectoriel, intégrant à la fois une action proportionnelle et une action dérivative.

Enfin, il faut également tenir compte de la précision du GPS. Comme la position du bateau est utilisée directement pour calculer sa distance à la bouée et son erreur radiale, une erreur de localisation peut provoquer un décalage apparent ou réel de la trajectoire. Cette limite est particulièrement importante lors des essais en mer, où les conditions expérimentales sont moins contrôlées que dans le simulateur.

11 Discussion

Les résultats obtenus montrent que la démarche suivie pendant le stage est cohérente : le contrôleur par champ vectoriel a d'abord permis de définir une loi de commande interprétable, puis cette loi a été utilisée pour générer une base de données et entraîner un réseau de neurones compact.

Le contrôleur par champ vectoriel présente l'avantage d'être explicite. Chaque étape du calcul peut être analysée : calcul du champ, orientation désirée, erreur angulaire et commande différentielle. Cette propriété a été importante pour construire une première solution de contrôle et pour comprendre le comportement du bateau en simulation comme pendant les essais réels.

Les essais ont cependant montré que la trajectoire obtenue avec ce contrôleur n'était pas parfaitement superposée au cercle désiré. Le bateau avait tendance à réaliser un mouvement de carrousel avec un certain décalage radial. Ce comportement avait déjà été observé en simulation, ce qui montre que le simulateur permettait de retrouver qualitativement une partie du comportement réel du BlueBoat. En mer, ce décalage semblait toutefois plus important, ce qui peut être lié aux perturbations extérieures, aux limites du modèle dynamique, aux délais de communication ou encore à la précision du GPS.

Le réseau de neurones, de son côté, permet d'approximer la loi de commande sous une forme compacte. Une fois entraîné, il ne nécessite plus l'exécution explicite des équations du contrôleur par champ vectoriel. Il suffit de réaliser une propagation avant à partir des quatre entrées : x, y, $\cos(\theta)$ et $\sin(\theta)$.

Les résultats de simulation et les observations en mer ont montré que le réseau de neurones reproduisait le comportement général du contrôleur expert. Le bateau était attiré vers la zone du rayon désiré et formait un mouvement de carrousel autour de la bouée. Dans les essais réalisés, la trajectoire obtenue avec le réseau semblait même plus fermée que celle obtenue avec le contrôleur par champ vectoriel, avec un décalage plus faible par rapport au cercle désiré.

Un autre résultat intéressant concerne la généralisation du réseau. Même lorsque le bateau était placé initialement dans une région située en dehors du domaine utilisé pour l'entraînement, il a tout de même eu tendance à rejoindre le cercle désiré. Cette observation indique que le réseau n'a pas seulement mémorisé les exemples de la base de données, mais qu'il a appris une approximation cohérente de la loi de commande. Il faut cependant rester prudent, car cette généralisation n'est pas garantie pour toutes les conditions initiales possibles.

Enfin, le décalage observé autour du cercle désiré suggère une piste d'amélioration importante. Le contrôleur utilisé pourrait être amélioré en ajoutant un terme dérivatif basé sur la vitesse angulaire du bateau. Une commande de type proportionnel-dérivatif,

$$u_{diff} = K_\theta e_\theta - K_D \omega, \tag{83}$$

permettrait de mieux amortir la rotation du bateau et pourrait réduire les oscillations ou le décalage par rapport au rayon désiré. Une nouvelle base de données pourrait ensuite être générée à partir de ce contrôleur amélioré, puis un nouveau réseau de neurones pourrait être entraîné pour reproduire ce comportement.

12 Conclusion

Ce stage a permis de construire une chaîne complète de contrôle pour le BlueBoat, depuis la modélisation en simulation jusqu'aux premiers essais réels en mer. Le travail a d'abord consisté à définir un contrôleur par champ vectoriel capable de faire converger le bateau vers une trajectoire circulaire autour d'une bouée. Ce contrôleur a ensuite

été utilisé comme contrôleur expert pour générer une base de données d'apprentissage supervisé.

Une architecture de réseau de neurones compacte a été étudiée. La version principale utilisée dans ce rapport est une architecture sans biais de type $4 - 8 - 8 - 2$, composée de quatre entrées, deux couches cachées de huit neurones avec activation ReLU, et deux sorties correspondant aux commandes des propulseurs. La base de données finale contenait environ 1,15 million d'exemples, obtenus à partir d'une grille de positions et d'orientations. Le réseau a ensuite été entraîné pour reproduire les commandes générées par le contrôleur par champ vectoriel.

Les simulations ont permis de comparer le comportement du contrôleur par champ vectoriel et celui du réseau de neurones. Elles ont montré que les deux approches permettaient au bateau de rejoindre la zone du cercle désiré et de former un mouvement de carrousel autour de la bouée. La carte d'erreur a également montré que l'approximation du réseau n'était pas uniforme dans tout l'espace, ce qui est cohérent avec le choix d'une architecture volontairement compacte.

Une architecture logicielle a ensuite été mise en place pour tester les contrôleurs sur le BlueBoat réel. Cette architecture repose sur la communication MAVLink pour lire les données GPS et d'attitude, convertir ces informations dans un repère local, calculer les commandes du contrôleur ou du réseau, puis envoyer les ordres au bateau par RC override. Des mécanismes de sécurité ont également été intégrés afin d'envoyer une commande neutre en cas de problème de données ou à l'arrêt du programme.

Les essais réels en mer ont confirmé qualitativement plusieurs comportements observés en simulation. Le contrôleur par champ vectoriel a permis au bateau de former un mouvement de carrousel, mais avec une trajectoire plus ouverte que le cercle désiré. Le contrôleur neuronal a également permis de rejoindre la zone du rayon désiré, avec une trajectoire plus fermée dans les essais observés. Le réseau a aussi montré une certaine capacité de généralisation lorsque le bateau était placé en dehors de la région utilisée pour l'entraînement.

Les perspectives de ce travail concernent principalement l'amélioration de la robustesse en conditions réelles. Il serait intéressant de refaire des essais avec une acquisition complète des données afin de comparer quantitativement les résultats réels et simulés. Il serait également utile d'étudier l'influence de la précision GPS, des délais de communication, du choix du rayon désiré et des conditions initiales. Enfin, une amélioration naturelle consisterait à ajouter un terme dérivatif au contrôleur par champ vectoriel, à générer une nouvelle base de données avec ce contrôleur proportionnel-dérivatif, puis à entraîner un nouveau réseau de neurones pour reproduire cette loi de commande améliorée.

Références

[1] L. Jaulin, N. Caouren, T. Bourin, M. Benigno, Q. Brateau, A. Morvan et R. Bornier, *Blue Carousel Challenge*. Disponible en ligne : `https://webperso.ensta.fr/Jaulin/carousel.html`.

[2] L. Jaulin, *Mobile robotics : Guidance*. ENSTA Bretagne, 24 avril 2025. Disponible en ligne : `https://www.ensta-bretagne.fr/robmooc/`.

[3] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2^{e} édition, John Wiley & Sons, 2021.

[4] T. I. Fossen, *Guidance and Control of Ocean Vehicles*, John Wiley & Sons, 1994.