Auteur
LE SCOËZEC Mathis

Tuteur Entreprise
LEGRAND Julien

Tuteur Académique
JAULIN Luc

Conception, commande robuste et fusion de capteurs pour un robot sous-marin autonome résident

Rapport de Projet de Fin d'Études

Ifremer

ENSTA INSTITUT POLYTECHNIQUE DE PARIS

Master Robotique Mobile
Promotion 2026
08/2026

Résumé & Abstract

Résumé

Ce rapport de Projet de Fin d'Études porte sur la conception, la modélisation dynamique, la fusion de capteurs, le guidage et le contrôle robuste d'un véhicule sous-marin autonome (AUV) résident. Ce stage a été réalisé au sein de l'Ifremer.
Le travail présenté s'articule autour de trois axes principaux : la simulation dynamique de la plateforme sous Gazebo, l'estimation d'état par fusion de capteurs (IMU, DVL, USBL, vision), et la mise en œuvre de lois de guidage et de contrôle (Super-Twisting Algorithm) afin d'assurer le suivi de trajectoire ainsi que les manœuvres d'amarrage et de désamarrage.
Les performances des algorithmes développés ont été validées en simulation, puis évaluées lors d'essais réels en bassin et en mer.

Mots-clés : Robotique sous-marine, AUV résident, Commande robuste, Super-Twisting Algorithm, Fusion de capteurs, Docking.

Abstract

This Master's Thesis report focuses on the design, dynamic modeling, sensor fusion, guidance, and robust control of a resident Autonomous Underwater Vehicle (AUV). This internship was carried out at Ifremer.
The work presented revolves around three main axes : the dynamic simulation of the platform in Gazebo, state estimation via sensor fusion (IMU, DVL, USBL, vision), and the implementation of guidance and control laws (Super-Twisting Algorithm) to ensure trajectory tracking as well as docking/undocking maneuvers.
The performance of the developed algorithms was validated in simulation and subsequently evaluated through real-world experiments in test basins and at sea.

Keywords : Underwater robotics, Resident AUV, Robust control, Super-Twisting Algorithm, Sensor fusion, Docking.

Remerciements

Je tiens à exprimer ma profonde gratitude à mon tuteur d'entreprise, **Julien LEGRAND**, pour son encadrement, sa disponibilité et sa confiance tout au long de ce Projet de Fin d'Études. Je remercie bien évidemment l'ensemble du département RDT pour son accueil et son aide, et plus particulièrement l'équipe avec laquelle j'ai directement travaillé :

- **Damien LE VOURC'H**, **Nicolas MERTZ** et **Clara JACOB**, pour leur aide précieuse sur les parties mécanique et électronique de l'AUV, ainsi que durant les phases d'essais ;
- **Adrien CHAUVET** et **Maxime FERRERA**, pour leur aide sur la partie algorithmique ;
- **Sébastien CHALONY**, pour son accueil au bassin et son soutien indispensable lors des manipulations à réaliser en plongée ;
- **Jérome BLANDIN**, pour son rôle dans la gestion du projet ;
- **Jean-Yves LE COAIL**, pour son aide lors de la remise en état du câble de l'AUV.

Mes remerciements s'adressent également à mon tuteur académique, **Luc JAULIN**, ainsi qu'à mon professeur **Lionel LAPIERRE**, pour leurs précieux conseils scientifiques et leur suivi tout au long de ce projet.

Table des matières

Table des figures

Liste des acronymes

AHRS Attitude and Heading Reference System 12
AUV Autonomous Underwater Vehicle 8

CFD Computational Fluid Dynamics 21
CSV Comma-Separated Values 24

DVL Doppler Velocity Log 10

EKF Extended Kalman Filter 29

GNSS Global Navigation Satellite System 12

JSON JavaScript Object Notation 27

MPC Model Predictive Control 43

NED North-East-Down 18

PFE Projet de Fin d'Études 8
PID Proportionnel Intégral Dérivé 43
PWM Pulse Width Modulation 24

ROS Robot Operating System 15
ROV Remotely Operated Vehicle 8

SDK Software Development Kit 28
SMC Sliding Mode Control 43
STA Super-Twisting Algorithm 44

TCP Transmission Control Protocol 27

URDF Unified Robot Description Format 29
USBL Ultra-Short Baseline 10

Chapitre 1

Présentation du projet et du système

Ce premier chapitre pose le cadre général du Projet de Fin d'Études (PFE). Après une brève présentation de l'organisme d'accueil, le projet d'Autonomous Underwater Vehicle (AUV) résident est détaillé à travers ses enjeux techniques, la description du système (AUV et station d'amarrage) et la définition précise de mon périmètre d'action.

1.1 Présentation d'Ifremer

L'Institut Français de Recherche pour l'Exploitation de la Mer (Ifremer) est un Établissement Public à Caractère Industriel et Commercial (EPIC) créé en 1984. Sous la tutelle des ministères de la Transition écologique et de l'Enseignement supérieur, il mène des missions de recherche, d'expertise et d'innovation pour la protection de l'océan et la gestion durable de ses ressources. L'institut s'appuie sur 23 implantations en métropole et en Outre-mer, structurées autour de 5 centres principaux (Bretagne, Atlantique, Manche – Mer du Nord, Méditerranée, Pacifique).

1.1.1 Cadre du stage

Mon stage s'est déroulé sur le site de Plouzané, au sein du Service Ingénierie et Instrumentation Marine (SIIM), rattaché à l'unité de recherche Recherches et Développements Technologiques (RDT). Ce service regroupe des compétences multidisciplinaires (mécanique, électronique, informatique, optique) dédiées à la conception et à la fabrication de prototypes d'instrumentation marine.

1.1.2 Moyens techniques et infrastructures

Pour ses opérations et la validation de ses développements, le site de Plouzané dispose de moyens d'essai et de déploiement de pointe :

— **La Flotte Océanographique Française :** opérée par Genavir, elle comprend 4 navires hauturiers, 6 navires côtiers et des engins sous-marins profonds jusqu'à 6000 mètres (le sous-marin habité *Nautile*, les Remotely Operated Vehicle (ROV) *Victor 6000* et *Ariane*, ainsi que les AUV *Asterx*, *Idefx* et *Ulyx*).

— **Infrastructures d'essais :** le site est équipé de caissons hyperbares pour simuler les hautes pressions, d'un site de tests en mer facilement accessible et d'un bassin d'essais à houle.

1.2 Le projet AUV résident

1.2.1 But du projet et contexte

La nécessité d'une observation in situ, continue et à long terme des océans est reconnue depuis plusieurs décennies comme un enjeu fondamental pour de nombreuses disciplines scientifiques et études interdisciplinaires [1]. Pour répondre à ce besoin, diverses plateformes d'observation ont été déployées, chacune présentant des compromis spécifiques entre couverture spatiale et résolution temporelle :

— **Les flotteurs lagrangiens (Argo) :** Ils offrent une vision synoptique à grande échelle des phénomènes physico-chimiques, mais restent soumis aux courants et ne permettent pas de cibler une zone fixe.
— **Les observatoires fond de mer (points fixes) :** Ils constituent la solution privilégiée pour l'acquisition de longues séries temporelles de données. Cependant, leur emprise spatiale demeure extrêmement limitée et circonscrite au rayon d'action de leurs capteurs.
— **Les planeurs sous-marins (*gliders*) :** Outils précieux pour les mesures à grande échelle, leur autonomie sans intervention humaine reste limitée à quelques mois et leur conception ne leur permet pas d'évoluer à proximité immédiate du fond marin.
— **Les robots sous-marins autonomes conventionnels :** Ils offrent une excellente couverture spatiale, mais leur mise en œuvre dépend de la présence continue d'un navire de surface. Cela limite leurs missions à quelques jours d'affilée tout en engendrant des coûts financiers et une empreinte environnementale importants.

Pour surmonter ces limitations, et dans un contexte d'accroissement continu de l'autonomie des véhicules marins, l'Ifremer a engagé en 2023 le développement d'un **AUV résident** dans le cadre du projet **ScInObs** (*Science and Innovation for Observatories*). Le projet doit développer un système composé d'un AUV et d'une station d'amarrage. L'objectif majeur est de réduire l'empreinte carbone et les coûts opérationnels de l'acquisition de données, tout en étendant la portée spatiale des observatoires sous-marins permanents grâce à la mobilité d'un vecteur autonome capable d'opérer depuis une station d'accueil immergée. Cependant, le passage d'un AUV conventionnel à un système résident immergé pendant plusieurs semaines voire plusieurs mois sans intervention humaine locale soulève des verrous technologiques majeurs articulés autour de quatre axes principaux :

1. Tenue aux agressions du milieu marin et bio-salissures
L'immersion prolongée soumet la plateforme et sa station d'accueil à une dégradation continue induite par l'environnement sous-marin :

— **Dégradation des interfaces optiques :** L'opacification des hublots des capteurs (caméras, projecteurs) et des cibles optiques de guidage par les bio-salissures compromet les fonctions de vision et de reconnaissance de la base.

- **Encrassement des mécanismes :** Le dépôt de sédiments sur les surfaces critiques peut bloquer ou altérer les mécanismes d'arrimage et de verrouillage sur la base. Des algues peuvent également bloquer voire endommager les moteurs.
- **Variation de masse et de centrage :** L'accumulation de sédiments et le développement d'organismes marins modifient la masse, la densité et l'hydrodynamique du robot. Cela altère son équilibre et sa dynamique, ce qui peut perturber le contrôle du robot.
- **Courants marins :** Si le courant est supérieur à la poussée créée par les moteurs, l'AUV ne pourra pas réaliser les trajectoires souhaitées, ce qui est critique si ce dernier est proche d'obstacles.

2. Fiabilité matérielle, logicielle et gestion des modes dégradés

L'absence complète d'opérateur humain et la difficulté de réaliser des opérations de maintenance imposent une fiabilité inédite.

- **Intégrité logicielle :** L'architecture logicielle embarquée doit présenter une tolérance aux fautes absolue. En cas d'anomalie ou de plantage d'un nœud de calcul, le système doit être capable de s'auto-diagnostiquer et de se réinitialiser en toute sécurité sans intervention externe.
- **Analyse des pannes et modes dégradés :** Chaque sous-ensemble (propulsion, électronique, énergie) doit faire l'objet d'analyses d'impact de panne rigoureuses. Le système doit pouvoir basculer dynamiquement dans des modes de fonctionnement dégradés garantissant la préservation de l'engin et son retour à la base ou sa mise en sécurité.

3. Localisation, guidage terminal et amarrage répétable

Le succès de la résidence repose sur la garantie d'un amarrage sans faille (100 % de réussite) à l'issue de chaque mission :

- **Navigation et retour à la base :** Le robot doit maintenir une estimation d'état et un positionnement permanents par rapport à sa base, en faisant face à des dérives d'intégration, des pertes probables des signaux acoustiques (Ultra-Short Baseline (USBL), Doppler Velocity Log (DVL)), des courants marins variables et des conditions de visibilité changeantes (turbidité).
- **Manœuvre d'approche et d'amarrage :** La phase d'approche de la base nécessite un alignement précis guidé par la fusion de capteurs. Les lois de commande doivent contrer efficacement les perturbations hydrodynamiques aux abords de la structure de docking.

4. Interfaces sous-marines et transferts sans contact

L'intégration entre le vecteur mobile et la station d'accueil nécessite la mise au point de sous-systèmes spécifiques :

- **Transfert d'énergie et de données sans contact :** Pour éviter les connecteurs électriques humides sensibles à la corrosion et à l'usure mécanique, le rechargement de la batterie et les transferts de données haut débit s'effectuent via des liaisons inductives.
- **Verrouillage mécanique :** Intégration d'un dispositif robuste de verrouillage mécanique de l'AUV sur sa station.
- **Alimentation de la base :** Pour recharger l'AUV, il faut que la base soit alimentée en énergie, soit via un câble, soit grâce à une source d'énergie directement intégrée à la

base.

1.2.2 Présentation de l'AUV

Le Graaltech X-300

Pour répondre à ces problématiques, l'Ifremer a fait le choix d'utiliser un AUV X-300 de la société italienne Graaltech, illustré sur la Figure 1.1. De légères modifications ont été réalisées par Graaltech à la demande de l'Ifremer pour pouvoir intégrer des capteurs supplémentaires.

Il est important de noter qu'il n'est pas prévu que l'X-300 actuel soit le vecteur final. Si les premières phases de tests sont concluantes, l'AUV sera modifié pour ajouter de la charge utile et de la batterie.

FIGURE 1.1 – AUV X-300 de Graaltech.

L'AUV final mesure 2,80 m de long pour 15,5 cm de diamètre et pèse actuellement environ 40 kg. Il est équipé de 6 moteurs T200 de chez Blue Robotics qui lui permettent de contrôler 5 degrés de liberté (tous sauf le roulis), dont la nomenclature est détaillée dans le Tableau 1.1.

TABLE 1.1 – Nomenclature et abréviations des moteurs de l'AUV X-300.

Nom du moteur	**Abréviation**	**Actions principales**
Moteur arrière gauche	RL (Rear Left)	Cavalement / Lacet
Moteur arrière droit	RR (Rear Right)	Cavalement / Lacet
Moteur vertical arrière	UR (Up Rear)	Pilonnement / Tangage
Moteur vertical avant	UF (Up Front)	Pilonnement / Tangage
Moteur horizontal arrière	SR (Side Rear)	Embardée / Lacet
Moteur transversal avant	SF (Side Front)	Embardée / Lacet

Cet AUV a été choisi pour 3 raisons principales :

— **Coût modéré :** L'AUV et ses modifications ont coûté environ 100 k€ à l'Ifremer. C'est un prix très raisonnable si on le compare aux AUV industriels classiques qui peuvent facilement dépasser 1 M€. Cela permet à l'AUV d'être théoriquement perdable, ce qui permet d'aborder les phases de tests plus sereinement et constitue un avantage majeur pour un défi aussi exigeant que la résidence.

— **Forme de torpille :** De par sa géométrie, le X-300 présente un faible coefficient de traînée sur son axe principal. Cela diminue l'impact du courant marin et augmente l'autonomie de l'engin.

— **Possibilité de *station-keeping* et *hovering* :** La plupart des AUV en forme de torpille sont sous-actionnés ou ont des degrés de liberté couplés. Ils ne peuvent donc pas réaliser de *hovering* (survoler une cible) ou de *station-keeping* (se maintenir en position), ce qui est très préjudiciable pour les missions d'observation scientifique. Le X-300 en est capable grâce à ses 5 degrés de liberté.

Capteurs embarqués

L'X-300 est conçu avec des capteurs embarqués nativement. Il est équipé d'une Attitude and Heading Reference System (AHRS), d'un baromètre et d'un Global Navigation Satellite System (GNSS). Ces capteurs, couplés au logiciel propriétaire de Graaltech, permettent de réaliser des missions de base avec l'AUV.

Pour obtenir un contrôle plus fin et plus souple, ce logiciel n'est pas utilisé par l'Ifremer et les commandes sont envoyées directement aux moteurs.

Pour obtenir des mesures plus précises, mesurer d'autres états ou avoir des informations sur l'environnement du robot, l'Ifremer a décidé d'intégrer des capteurs complémentaires au vecteur, résumés dans le Tableau 1.2.

TABLE 1.2 – Suite de capteurs embarqués sur l'AUV et leurs rôles.

Capteur	**Marque / Modèle**	**Rôle**
AHRS	SBG Systems (Ellipse-A)	Mesure l'attitude, les vitesses de rotation et les accélérations du robot.
DVL	Water Linked (A50)	Mesure la vitesse linéaire et l'altitude par rapport au fond (jusqu'à 50 m du fond).
USBL	Blueprint Subsea (SeaTrac)	Positionnement relatif de l'AUV par rapport à la base (portée max. : 1000 m).
Caméra frontale 4K	Allied Vision	Positionnement visuel de précision grâce aux mires (*AprilTags*) de la base (voir Fig. 1.2 et 1.3).

Pour la position précise des capteurs et actionneurs de l'AUV, le lecteur pourra se référer à l'Annexe C.

L'AUV est uniquement alimenté grâce à son pack batterie. Ce dernier peut se recharger soit via un chargeur externe quand le robot est hors de l'eau, soit via un connecteur inductif présent sous le robot et sur la base (Figures 1.2).

Une pièce mécanique appelée *"mâchoire"* a été rajouté à l'avant du robot. C'est cette pièce qui permet la réalisation des premières phases de notre méthode d'amarrage (Figure 1.2). Référez vous à la Section 1.2.4 pour plus d'informations.

En plus de l'ordinateur embarqué présent par défaut et chargé de la gestion bas niveau, le robot est équipé d'une carte Jetson Orin. Cette dernière n'est utilisée que comme une interface pour l'instant, mais devra à terme embarquer toute l'intelligence du robot.

Pour le moment, un *tether* (câble Ethernet de 100 m) est utilisé pour faire la liaison entre l'ordinateur de commande et l'AUV (Figure 1.2) . Ces deux derniers

FIGURE 1.2 – X-300 modifié

partagent donc le même réseau.

1.2.3 Présentation de la station d'amarrage

La station d'accueil (dont une vue 3D est présentée en Figure 1.3) permet à l'AUV de se recharger et de stocker ou transmettre ses données.

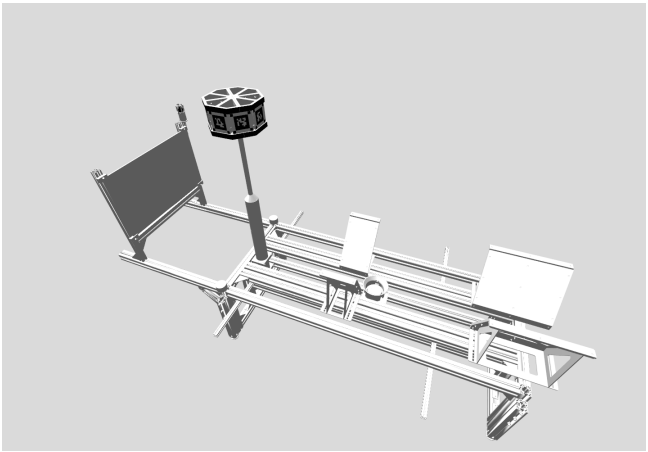


FIGURE 1.3 – Vue 3D de la station d'amarrage.

L'alimentation de la base et la communication avec la surface se font pour le moment grâce à un câble.

Il est envisagé de lui ajouter des batteries suffisantes pour recharger l'AUV durant toute la durée de la résidence. Cette solution permettrait de déployer facilement le système en eaux profondes, mais n'est pas envisageable actuellement au vu des barrières technologiques.

La station sera aussi une station scientifique, et sera donc équipée de divers capteurs dont l'architecture fonctionnelle est schématisée sur la Figure 1.4.

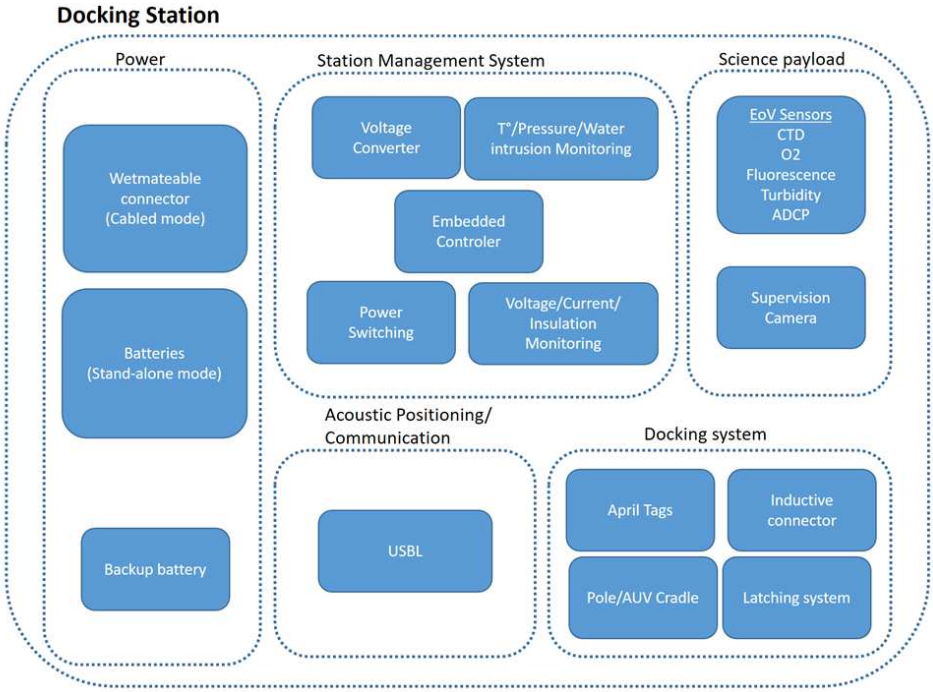


FIGURE 1.4 – Schéma des fonctions de la station d'amarrage.

La version actuelle de la base n'est pas la version finale, mais l'architecture globale ne devrait pas évoluer.

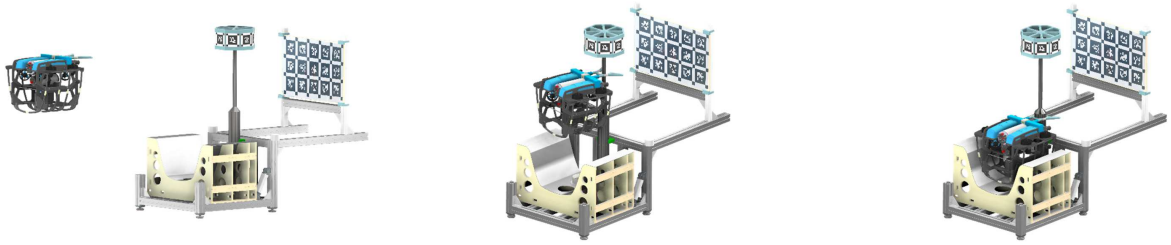
1.2.4 La méthode d'amarrage

Pour revenir vers la base, l'AUV utilisera principalement l'USBL ou la navigation à l'estime (*dead reckoning*, estimer sa position par l'intégration des vitesses) grâce à ses autres capteurs. Une fois proche de la base, il entrera en phase d'amarrage.

Le principe d'amarrage repose principalement sur la combinaison d'un guidage optique et d'un guidage mécanique passif. L'AUV utilise sa caméra frontale pour repérer des cibles visuelles (*AprilTags*) disposées sur la station, lui permettant d'estimer sa position relative et de s'aligner progressivement avec la base.

Une fois à proximité immédiate, le robot engage sa mâchoire autour d'un mât vertical de guidage (concept inspiré du *Kawasaki SPICE* [2]) qui bloque quatre de ses degrés de liberté pour contrer les courants marins. L'AUV descend ensuite le long de ce mât pour venir se caler naturellement dans un berceau d'accueil mécanique, où un loquet automatique le verrouille (fonctionnalité toujours en cours de développement). La procédure globale est illustrée sur la Figure 1.5 et la Figure 1.6. Cet amarrage physique garantit un alignement à quelques millimètres près, indispensable pour assurer le rechargement de la batterie et le transfert des données via le connecteur inductif.

La base est également équipée de mires (*AprilTags*) : un hexagone de 8 mires au-dessus du poteau d'amarrage et une plaque de 15 mires derrière le poteau d'amarrage. Lors des tests en pleine mer, nous utilisons pour des raisons de logistique une base simplifiée qui ne dispose que de l'hexagone.



(a) Phase d'approche (b) Alignement et descente (c) Verrouillage

FIGURE 1.5 – Procédure d'amarrage, exemple avec un BlueROV.

Pour cette raison, nous avons fait le choix de n'utiliser que l'hexagone pour se positionner visuellement durant la durée de mon stage. Cette décision a été prise par souci de cohérence et d'uniformité des algorithmes entre les tests en bassin et en pleine mer. Ceci rend la tâche plus difficile car ces mires ne sont pas visibles lors de la phase finale d'amarrage et de descente de l'AUV.

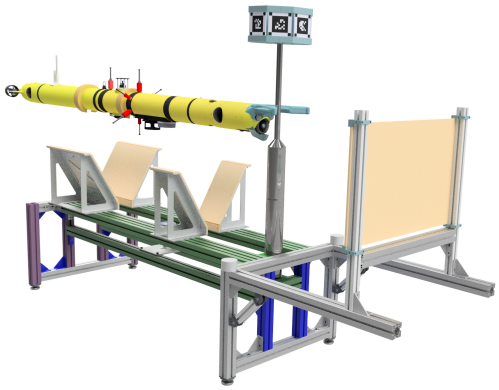


FIGURE 1.6 – Rendu 3D du X-300 en phase d'amarrage.

1.3 Architecture logicielle de l'AUV

L'architecture logicielle de l'AUV repose principalement sur l'environnement Robot Operating System (ROS) (ROS2 Humble dans notre cas), auquel s'intègrent la quasi-totalité de nos développements. Le langage privilégié est Python, bien que certains codes soient en C++.

Pour rappel, ROS repose sur une architecture modulaire et distribuée où des programmes indépendants, appelés nœuds, communiquent entre eux en publiant ou en s'abonnant à des flux de données nommés *topics*.

La schéma de la Figure 1.7 présente une architecture ROS simplifiée du projet.

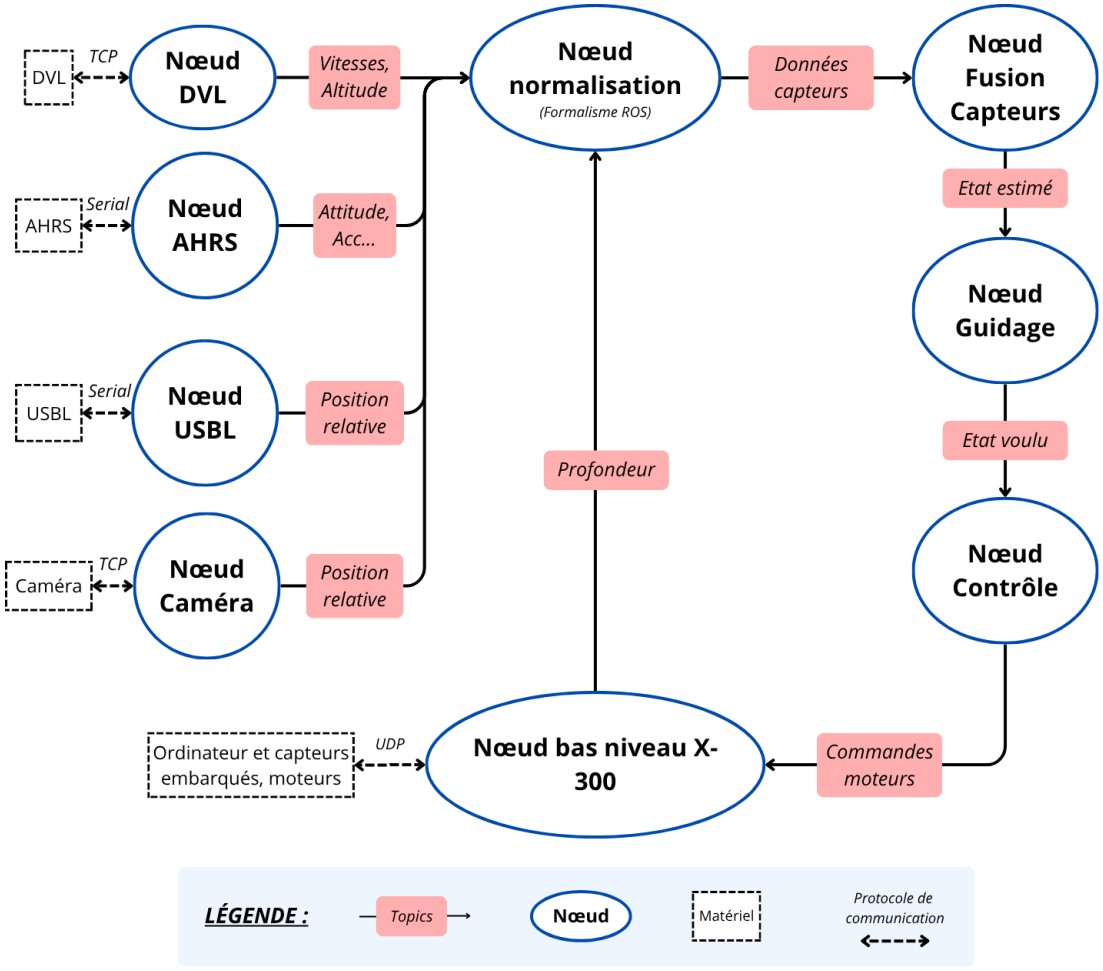


FIGURE 1.7 – Architecture ROS et logicielle simplifiée

1.4 Mon rôle et champ d'action

Mon intervention au sein de ce projet se concentre principalement sur l'intégration des capteurs, sur les briques d'estimation d'état, de guidage et de commande requises pour garantir des manœuvres d'amarrage et de désamarrage fiables et répétables ainsi que sur la réalisation de missions simples.

Étant arrivé en cours de projet, je n'ai pas pris part aux décisions sur la méthode générale d'amarrage, sur le choix de l'AUV, sur le design mécanique de la base ou de l'AUV, ni sur le choix des capteurs.

J'ai eu un accès quasi illimité au robot, mais un accès bien plus restreint au bassin (une vingtaine de jours au total, détaillés en Annexe A).

Mes principales tâches ont donc été :

1. **Intégration des capteurs :** programmation des nœuds ROS pour le DVL, l'AHRS SBG et l'USBL ;
2. **Modélisation et simulation :** modélisation des équations de l'AUV et création d'une

simulation ;

3. **Fusion de capteurs :** implémentation et réglage de la fusion de capteurs pour l'estimation d'état ;
4. **Guidage :** création d'un guidage par champ de vecteurs pour l'amarrage et implémentation d'un algorithme *pure pursuit* pour le suivi de points ;
5. **Contrôle :** implémentation d'un contrôle robuste (*Super-Twisting Algorithm*) pour l'AUV ;
6. **Validation expérimentale :** tests en bassin et en mer pour valider mes résultats de simulation.

Chapitre 2

Modélisation et simulation du système

Ce deuxième chapitre est consacré à la modélisation mathématique du robot sous-marin AUV X-300 ainsi qu'au développement de son environnement de simulation numérique. Les équations et la méthodologie sont fortement inspirés des travaux de Lapierre [3] et Borgognoni [4].

2.1 Modélisation dynamique de l'AUV X-300

Cette section se concentre sur les équations dynamiques de l'AUV. Ces équations ont par la suite été utilisées pour simuler l'AUV sous Gazebo et pour établir les lois de contrôle.

2.1.1 Introduction et Définition des Repères

On définit le repère monde et le repère local selon les conventions de la SNAME (*Society of Naval Architects and Marine Engineers*), schématisées sur la Figure 2.1 :

1. **Le repère Monde / Inertiel (*North-East-Down (NED)*) :** Noté $R_n = (O_n, \mathbf{x}_n, \mathbf{y}_n, \mathbf{z}_n)$, il est supposé galiléen. L'axe $\mathbf{x}_n$ pointe vers le Nord, $\mathbf{y}_n$ vers l'Est, et $\mathbf{z}_n$ pointe verticalement vers le bas (centre de la Terre).

2. **Le repère Corps (*Body-Fixed Frame*) :** Noté $R_b = (O_b, \mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b)$, il est solidaire du robot. L'origine O_b est au barycentre estimé du corps principal du sous-marin, qui est proche du centre géométrique. Les axes sont définis selon la convention nautique :
   — L'axe $\mathbf{x}_b$ pointe vers l'avant (cavalement / *surge*), associé à l'angle de roulis (ϕ, *roll*) autour de cet axe ;
   — L'axe $\mathbf{y}_b$ pointe vers la droite/tribord (embardée / *sway*), associé à l'angle de tangage (θ, *pitch*) autour de cet axe ;
   — L'axe $\mathbf{z}_b$ pointe vers le bas (pilonnement / *heave*), associé à l'angle de lacet (ψ, *yaw*) autour de cet axe.

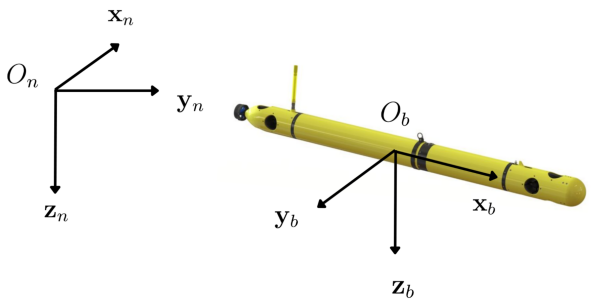


FIGURE 2.1 – Définition des repères Monde et Corps.

Les valeurs des différents paramètres et constantes utilisés lors ce chapitre sont disponibles en Annexe E.

2.1.2 Cinématique et Transformations de Repère

Vecteurs d'État Cinématiques Le mouvement du sous-marin est représenté par 6 degrés de liberté. On pose :

— **Vecteur de position et d'orientation dans le repère R_n :**

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \in \mathbb{R}^6, \quad \eta_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3, \quad \eta_2 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3 \tag{2.1}$$

Où (x, y, z) représentent les coordonnées cartésiennes, et (ϕ, θ, ψ) sont les angles d'Euler (Roulis, Tangage, Lacet).
Représenter les rotations par les angles d'Euler pose le problème du verrouillage de cardan (*gimbal lock*), avec une singularité autour de $\theta = \pm 90°$, mais on suppose que l'AUV n'atteindra jamais cet angle.

— **Vecteur de vitesse linéaire et angulaire dans le repère corps R_b :**

$$\nu = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \in \mathbb{R}^6, \quad \nu_1 = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3, \quad \nu_2 = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3 \tag{2.2}$$

Où (u, v, w) sont les vitesses translationnelles (surge, sway, heave) et (p, q, r) sont les vitesses angulaires (roll, pitch, yaw).

Prise en Compte du Courant Marin et Vitesse Relative Soit $\mathbf{v}_{c,n} \in \mathbb{R}^3$ le vecteur courant marin exprimé dans le repère Monde R_n. Sa projection exprimée dans le repère corps R_b s'écrit :

$$\mathbf{v}_{c,b} = \mathbf{R}_b^n(\eta_2)^T \mathbf{v}_{c,n} \tag{2.3}$$

On en déduit le vecteur de vitesse relative $\nu_r \in \mathbb{R}^6$ agissant sur le corps fluide :

$$\nu_c = \begin{bmatrix} \mathbf{v}_{c,b} \\ \mathbf{0}_{3\times 1} \end{bmatrix}, \quad \nu_r = \nu - \nu_c = \begin{bmatrix} u - v_{c,x} \\ v - v_{c,y} \\ w - v_{c,z} \\ p \\ q \\ r \end{bmatrix} \tag{2.4}$$

Jacobien de Transformation Cinématique $\mathbf{J}(\eta)$ La dérivation temporelle de la position et de l'attitude dans R_n s'exprime à partir des vitesses dans R_b via le Jacobien $\mathbf{J}(\eta) \in \mathbb{R}^{6\times 6}$:

$$\dot{\eta} = \mathbf{J}(\eta)\nu = \begin{bmatrix} \mathbf{R}_b^n(\eta_2) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{T}_b^n(\eta_2) \end{bmatrix} \nu \tag{2.5}$$

La matrice de rotation $\mathbf{R}_b^n(\eta_2) \in SO(3)$ (passage du repère corps au repère Monde) s'écrit :

$$\mathbf{R}_b^n(\eta_2) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\psi c\phi s\theta \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.6}$$

où $s(\cdot) = \sin(\cdot)$ et $c(\cdot) = \cos(\cdot)$.

La matrice de passage des vitesses angulaires $\mathbf{T}_b^n(\eta_2)$ est définie par :

$$\mathbf{T}_b^n(\eta_2) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}, \quad (\theta \neq \pm 90^\circ) \tag{2.7}$$

2.1.3 Équation Générale de la Dynamique de Fossen

On pose les équations de Fossen[5] :

$$\mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}(\nu)\nu + \mathbf{M}_A\dot{\nu}_r + \mathbf{C}_A(\nu_r)\nu_r + \mathbf{D}(\nu_r) + \mathbf{g}(\eta) = \tau_{\text{prop}} \tag{2.8}$$

Pour l'intégration numérique (résolution de l'équation d'état), l'accélération du corps est calculée en inversant la matrice d'inertie totale (en supposant un courant constant) :

$$\dot{\nu} = (\mathbf{M}_{RB} + \mathbf{M}_A)^{-1} \left[\tau_{\text{prop}} - \mathbf{g}(\eta) - \mathbf{C}_{RB}(\nu)\nu - \mathbf{C}_A(\nu_r)\nu_r - \mathbf{D}(\nu_r)\right] \tag{2.9}$$

2.1.4 Détail et Formulations Algébriques des Matrices Dynamiques

Matrice d'Opérateur Anti-symétrique Soit $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$, l'opérateur produit vectoriel anti-symétrique $S(\mathbf{x}) \in \mathbb{R}^{3\times 3}$ est défini par :

$$S(\mathbf{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.10}$$

Matrice d'Inertie du Corps Rigide $\mathbf{M}_{RB}$ et Coriolis $\mathbf{C}_{RB}$ Soit m la masse du sous-marin, $\mathbf{r}_g = [x_g, y_g, z_g]^T$ la position du centre de gravité par rapport au repère corps O_b, et $\mathbf{I}_o = \text{diag}(I_x, I_y, I_z)$ le tenseur d'inertie au centre géométrique. On choisit de modéliser le corps par un cylindre de longueur L et de diamètre $D = 2R$ pour simplifier le calcul :

$$I_x = \frac{1}{2}mR^2, \quad I_y = I_z = \frac{1}{4}m\left(R^2 + \frac{L^2}{3}\right), \tag{2.11}$$

La matrice d'inertie du corps rigide $\mathbf{M}_{RB} \in \mathbb{R}^{6\times6}$ prend la forme :

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3\times3} & -mS(\mathbf{r}_g) \\ mS(\mathbf{r}_g) & \mathbf{I}_o \end{bmatrix} \tag{2.12}$$

La matrice de Coriolis du corps rigide $\mathbf{C}_{RB}(\nu) \in \mathbb{R}^{6\times6}$ est construite selon les sous-blocs :

$$\mathbf{C}_{RB}(\nu) = \begin{bmatrix} \mathbf{0}_{3\times3} & -mS(\nu_1) - mS(\nu_2)\,S(\mathbf{r}_g) \\ -mS(\nu_1) + mS(\mathbf{r}_g)\,S(\nu_2) & -S(\mathbf{I}_o\nu_2) \end{bmatrix} \tag{2.13}$$

Matrice de Masse Ajoutée $\mathbf{M}_A$ et Coriolis Ajouté $\mathbf{C}_A$ Pour estimer la masse ajoutée, le corps de l'AUV est approximé par un ellipsoïde de demi-grands axes $a = L/2$, $b = R$, $c = R$. On utilise les théorèmes de Lamb [6] pour le calcul de la masse ajoutée :

$$\Delta(u) = \sqrt{(a^2+u)(b^2+u)(c^2+u)} \tag{2.14}$$

$$A_0 = abc\int_0^\infty \frac{du}{(a^2+u)\Delta(u)}, \quad B_0 = abc\int_0^\infty \frac{du}{(b^2+u)\Delta(u)}, \quad C_0 = abc\int_0^\infty \frac{du}{(c^2+u)\Delta(u)} \tag{2.15}$$

On pose $V = abc$. Les coefficients de masse ajoutée λ_{ii} s'expriment analytiquement :

$$\lambda_{11} = \frac{4}{3}\pi\rho V\frac{A_0}{2-A_0}, \quad \lambda_{22} = \frac{4}{3}\pi\rho V\frac{B_0}{2-B_0}, \quad \lambda_{33} = \frac{4}{3}\pi\rho V\frac{C_0}{2-C_0} \tag{2.16}$$

$$\lambda_{44} = 0 \quad (\text{car } b = c) \tag{2.17}$$

$$\lambda_{55} = \frac{4}{15}\pi\rho V\frac{(c^2-a^2)^2(A_0-C_0)}{2(c^2-a^2)+(C_0-A_0)(c^2+a^2)} \tag{2.18}$$

$$\lambda_{66} = \frac{4}{15}\pi\rho V\frac{(a^2-b^2)^2(B_0-A_0)}{2(a^2-b^2)+(A_0-B_0)(a^2+b^2)} \tag{2.19}$$

D'où $\mathbf{M}_A = \text{diag}(\lambda_{11}, \lambda_{22}, \lambda_{33}, \lambda_{44}, \lambda_{55}, \lambda_{66})$.

À noter que représenter l'AUV par un ellipsoïde constitue une simplification et introduit des erreurs. Cependant, la masse ajoutée a un impact limité à faible vitesse et l'obtention de cette matrice par calculs de Computational Fluid Dynamics (CFD) aurait été trop complexe dans le temps imparti du stage.

Pour la matrice de Coriolis associée aux masses ajoutées $\mathbf{C}_A(\nu_r)$, en notant $\mathbf{A}_{11} = \text{diag}(\lambda_{11}, \lambda_{22}, \lambda_{33})$ et $\mathbf{A}_{22} = \text{diag}(\lambda_{44}, \lambda_{55}, \lambda_{66})$:

$$\mathbf{C}_A(\nu_r) = \begin{bmatrix} \mathbf{0}_{3\times3} & S(\mathbf{A}_{11}\nu_{r1}) \\ S(\mathbf{A}_{11}\nu_{r1}) & S(\mathbf{A}_{22}\nu_{r2}) \end{bmatrix} \tag{2.20}$$

Forces et Moments Restituants Hydrostatiques $\mathbf{g}(\eta)$ Soit $W = m \cdot g$ le poids total du sous-marin et $B = \rho \cdot g \cdot V_{\text{deplace}}$ la poussée d'Archimède exercée sur le volume immergé. Soient $\mathbf{r}_g = [x_g, y_g, z_g]^T$ et $\mathbf{r}_b = [x_b, y_b, z_b]^T$ les positions respectives du centre de gravité et de flottabilité.

Le vecteur des forces et moments restituants $\mathbf{g}(\eta) \in \mathbb{R}^6$ s'exprime par :

$$\mathbf{g}(\eta) = \begin{bmatrix} (W-B)\sin\theta \\ -(W-B)\cos\theta\sin\phi \\ -(W-B)\cos\theta\cos\phi \\ -(y_gW - y_bB)\cos\theta\cos\phi + (z_gW - z_bB)\cos\theta\sin\phi \\ (z_gW - z_bB)\sin\theta + (x_gW - x_bB)\cos\theta\cos\phi \\ -(x_gW - x_bB)\cos\theta\sin\phi + (y_gW - y_bB)\sin\theta \end{bmatrix} \tag{2.21}$$

Frottements et Amortissement Hydrodynamique $\mathbf{D}(\nu_r)$ On suppose un écoulement laminaire et on ignore le phénomène de portance.

Classiquement, la modélisation de la traînée dans les équations de Fossen repose sur l'identification d'une matrice d'amortissement diagonale $\mathbf{D}(\boldsymbol{\nu})$ décomposée en un terme linéaire constant $\mathbf{D}_{\text{lin}}$ et un terme quadratique $\mathbf{D}_{\text{quad}}(\boldsymbol{\nu}_r) = \text{diag}\left(X_{u|u|}|u_r|, Y_{v|v|}|v_r|, \dots\right)$. Ces coefficients sont classiquement déterminés à l'aide de CFD ou d'expérimentations en bassin. Cependant, ces méthodes d'identifications sont compliquées à mettre en place et demande des compétences particulières.

Ici, on opte pour la théorie des tranches (*strip theory*) en simplifiant l'AUV par un cylindre. Cette méthode approxime l'amortissement par un terme purement quadratique. Elle présente l'avantage de prendre en compte les effets de couplage entre les vitesses de translation et de rotation.

Cependant, sous l'hypothèse d'un amortissement uniquement quadratique, cette méthode s'avère peu fidèle à la réalité aux faibles vitesses. On ajoute donc des coefficients constants déterminés empiriquement $\mathbf{D}_{\text{lin}}$ pour simuler un amortissement linéaire et stabiliser la simulation :

$$\mathbf{D}(\nu_r) = \mathbf{D}_{\text{lin}}\nu_r + \mathbf{D}_{\text{quad}}(\nu_r) \tag{2.22}$$

1. **Traînée axiale (Surge) :** Calculée sur la surface frontale projetée $S_{\text{ellipse}} = \pi R^2$, sans nécessiter l'usage de la théorie des tranches :

$$X_{u|u|} = -\frac{1}{2}\rho S_{\text{ellipse}} C_{d,\text{ellipse}} \implies D_{\text{quad, surge}} = X_{u|u|}u_r|u_r| \tag{2.23}$$

2. **Théorie des tranches pour Sway, Heave, Pitch, Yaw :** Le corps du sous-marin est subdivisé en $N_s = 25$ tranches transversales de longueur $dx = L/N_s$ et de surface projetée $S_{\text{tranche}} = 2Rdx$. Pour chaque tranche localisée en $x_i \in [-L/2, L/2]$, les vitesses locales sont :

$$v_{y,i} = v_r + x_i \cdot r, \quad v_{z,i} = w_r - x_i \cdot q \tag{2.24}$$

Les forces transversales élémentaires de traînée $df_y(x_i)$ et $df_z(x_i)$ s'écrivent :

$$df_y(x_i) = -\frac{1}{2}\rho S_{\text{tranche}} C_{d,\text{cyl}} \cdot v_{y,i}|v_{y,i}| \tag{2.25}$$

$$df_z(x_i) = -\frac{1}{2}\rho S_{\text{tranche}} C_{d,\text{cyl}} \cdot v_{z,i}|v_{z,i}| \tag{2.26}$$

Par intégration sur la longueur totale, les termes quadratiques globaux deviennent :

$$\begin{cases} D_{\text{quad, sway}} = \sum_{i=1}^{N_s} df_y(x_i) \\ D_{\text{quad, heave}} = \sum_{i=1}^{N_s} df_z(x_i) \\ D_{\text{quad, pitch}} = -\sum_{i=1}^{N_s} df_z(x_i) \cdot x_i \\ D_{\text{quad, yaw}} = \sum_{i=1}^{N_s} df_y(x_i) \cdot x_i \end{cases} \tag{2.27}$$

Modèle de Propulsion et Configuration Moteurs Les 6 moteurs créent une force globalisée $\tau_{\text{prop}} \in \mathbb{R}^6$. τ_{prop} est calculé via la matrice d'allocation des propulseurs $\mathbf{A} \in \mathbb{R}^{6\times 6}$ et le vecteur de commande des moteurs $\mathbf{u}_{\text{moteurs}} \in \mathbb{R}^6$:

$$\tau_{\text{prop}} = \mathbf{A} \cdot \mathbf{u}_{\text{moteurs}} \tag{2.28}$$

où le vecteur de commande regroupe les poussées générées individuellement par chaque moteur (voir le Chapitre 1, Tableau 1.1) :

$$\mathbf{u}_{\text{moteurs}} = \begin{bmatrix} u_{\text{RL}} & u_{\text{RR}} & u_{\text{UR}} & u_{\text{UF}} & u_{\text{SR}} & u_{\text{SF}} \end{bmatrix}^T \tag{2.29}$$

L'AUV X-300 possède 6 propulseurs positionnés aux distances géométriques d_1, d_2, d_3. La matrice d'allocation $\mathbf{A}$ s'écrit alors :

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & d_2 & -d_2 & 0 & 0 \\ -d_1 & d_1 & 0 & 0 & d_3 & -d_3 \end{bmatrix} \tag{2.30}$$

2.2 Simulation Gazebo

En raison de la faible disponibilité du bassin d'essais et des risques d'endommagement de l'AUV, le développement d'un environnement de simulation s'est avéré indispensable pour le bon déroulement du stage.

L'objectif est de mettre en place un jumeau numérique suffisamment fidèle au système réel pour valider les lois de commande, les algorithmes de guidage et les séquences d'amarrage et de désamarrage avant leur déploiement sur le robot. Disposer d'une vérité terrain — non accessible

lors des essais avec le robot réel — facilite grandement le paramétrage des fusions de capteurs et de la commande.

Une première simulation simplifiée du robot a été réalisée en Python afin de valider la stabilité des équations dynamiques. Néanmoins, une simulation dépourvue d'environnement 3D reste insuffisante pour évaluer le positionnement visuel.

Le choix du simulateur s'est orienté vers une solution open-source afin de s'affranchir de coûts de licence. Trois options principales ont été envisagées :

— **Isaac Sim :** Écarté car il nécessite impérativement une carte graphique Nvidia ;
— **Stonefish :** Écarté en raison de sa prise en main complexe ;
— **Gazebo :** Retenu car open-source, flexible et déjà maîtrisé.

La version retenue pour ce développement est *Gazebo Harmonic*.

2.2.1 Plugins créés

En plus des fonctionnalités natives de Gazebo, plusieurs plugins ont été développés spécifiquement :

— **Plugin Baromètre :** Capteur non présent nativement dans Gazebo.
  — Simulé à partir de la position en Z inversée, avec possibilité d'ajouter du bruit sur la mesure ;
  — Publie également la pression correspondant à la profondeur.
— **Plugin DVL :** Capteur non présent nativement dans Gazebo.
  — Simulé en récupérant les vitesses de l'objet dans le repère monde, puis en les exprimant dans le repère du DVL ;
  — Utilise un capteur Lidar natif de Gazebo pour estimer l'altitude, ce qui permet d'interrompre la publication si le fond est trop éloigné ;
  — Permet l'ajout de bruit sur la mesure.
— **Plugin USBL :** Capteur non présent nativement dans Gazebo.
  — Simulé en convertissant la position de l'émetteur (sur l'AUV) dans le repère du récepteur (sur la base) ;
  — Adapte la fréquence de publication en fonction de la distance entre les deux modules ;
  — Permet l'ajout de bruit sur la mesure.
— **Plugin *Thruster* :** Actionneur présent nativement dans Gazebo, mais réécrit pour offrir plus de flexibilité.
  — Écoute un *topic* de commande moteur acceptant des consignes en Newtons, en PWM (entre 1100 et 1900) ou en pourcentage ;
  — Calcule la poussée produite (pour le Pulse Width Modulation (PWM) et le pourcentage) via une approximation linéaire ou par interpolation à partir d'un fichier Comma-Separated Values (CSV) de caractéristiques moteur ;
  — Intègre la simulation d'un temps de réponse dynamique des moteurs.
— **Plugin Hydrodynamique :** Réécrit afin de réutiliser directement les équations hydrodynamiques présentées dans la Section 2.1.
  — Applique globalement les équations définies précédemment ;
  — Injecte la masse ajoutée sous forme de forces extérieures, le moteur physique de Gazebo ne permettant pas de configurer une masse anisotrope ou d'imposer directement

une accélération ;
— Applique un filtre passe-bas adaptatif axe par axe ($\alpha = 0{,}1$) sur la masse ajoutée ($\tau_{am} = -\mathbf{M}_a \dot{\nu}_{\text{processed}}$) afin d'éliminer les instabilités numériques lorsque $M_a(i) > \text{Inertia}_{\text{dry}}(i)$, tout en maintenant la réactivité du système.

2.2.2 Présentation de la simulation

L'AUV simulé (comme représenté en rendu 3D sur la Figure 2.2) dispose des mêmes capteurs et actionneurs que l'AUV réel. Son comportement dynamique présente toutefois des écarts en raison des simplifications effectuées lors du calcul de certains coefficients. Faute de temps, ces différences de comportement dynamique entre la simulation et le réel n'ont pas fait l'objet d'une évaluation quantitative.

La simulation est conçue pour fonctionner de manière transparente avec les mêmes codes que ceux embarqués sur le robot réel.

Les principales approximations de la simulation découlent des simplifications suivantes :
— **Modélisation du capteur :** Les bruits de mesure sont modélisés par des lois gaussiennes.
— **Optimisation graphique :** Le nombre de facettes des modèles 3D a été réduit afin d'optimiser le temps de chargement.
— **Hypothèse de corps rigide :** Le corps de l'AUV est considéré comme un solide parfaitement indéformable.
— **Propriétés de surface :** Aucun coefficient de frottement spécifique n'a été appliqué sur les surfaces de l'engin.

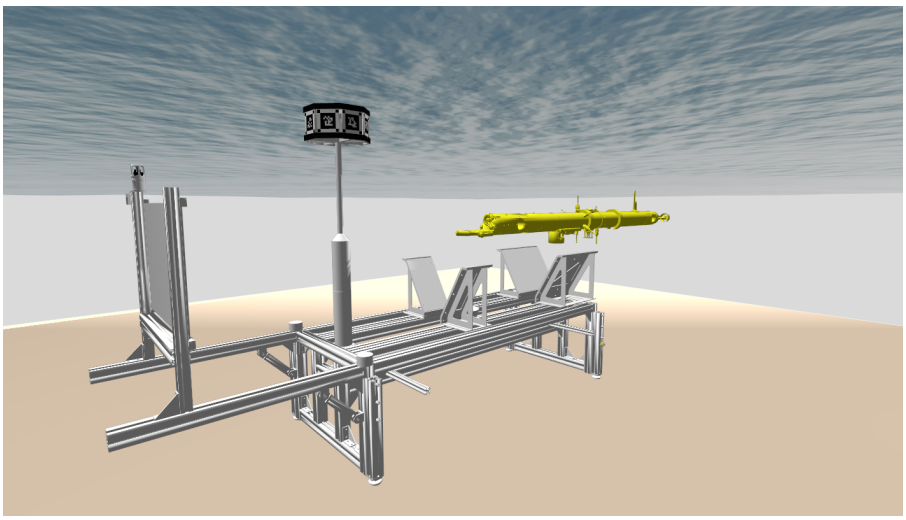


FIGURE 2.2 – Environnement de simulation développé sous Gazebo Harmonic.

Il est possible d'élaborer un environnement 3D plus complexe et réaliste ; cependant, ces modèles se révèlent peu pratiques lors des phases actives de développement (ralentissement de la simulation, diagnostic des erreurs rendu plus complexe).

Sur certains aspects, la simulation s'est avérée plus exigeante que les essais réels, notamment lors des phases d'amarrage. En simulation, l'alignement entre la pince du robot et le mât

d'amarrage nécessite une précision millimétrique pour réussir l'amarrage, alors que le système réel offre une tolérance mécanique bien plus importante. La réduction de la résolution géométrique du modèle 3D, l'absence de souplesse mécanique et l'application des coefficients de frottement par défaut constituent les causes probables de cet écart.

Enfin, la simulation s'est montrée très efficace lors des phases de débogage. Elle a permis de modifier rapidement des paramètres et configurations pour reproduire des comportements indésirables observés sur le robot réel, facilitant ainsi l'identification des erreurs de logique. Cet outil a ainsi permis un gain de temps significatif tout en évitant de placer l'AUV réel dans des situations à risque.

Chapitre 3

Capteurs et Fusion de Données

Ce troisième chapitre traite de l'intégration des capteurs à l'écosystème ROS de l'AUV ainsi que de la fusion de données, dont l'objectif est d'estimer en continu l'état du robot.

3.1 Capteurs intégrés à l'AUV

Cette section présente brièvement les capteurs intégrés au sous-marin (non présents nativement sur l'AUV), leurs protocoles de communication associés, ainsi que les difficultés techniques rencontrées pour chacun d'eux.

3.1.1 DVL A50 de Waterlinked

Principe et protocoles Le DVL communique via une connexion Transmission Control Protocol (TCP). Chaque paquet transmis adopte le format JavaScript Object Notation (JSON), ce qui a grandement simplifié son intégration au sein de l'écosystème ROS.

Problématiques rencontrées De nombreuses difficultés sont apparues lors des essais expérimentaux :

- **Défaillance matérielle :** Sur les deux exemplaires à disposition, l'un a été diagnostiqué défaillant par le constructeur WaterLinked.
- **Perte de signal récurrente :** Le DVL perd régulièrement son signal à seulement quelques mètres sous la surface. Après élimination des autres causes matérielles et logicielles, nous suspectons une interaction néfaste liée à la géométrie du bassin d'essai d'Ifremer.
- **Bruit et valeurs aberrantes :** Les mesures transmises demeurent fortement bruitées et renvoient ponctuellement des valeurs aberrantes.

3.1.2 AHRS Ellipse-A de SBG

Principe et protocoles L'AHRS est fourni avec un package ROS prêt à l'emploi, rendant son intégration directe. Le capteur est raccordé à un port série de la Jetson, lui-même redirigé vers un port TCP.

Cet AHRS fournit les mesures brutes de ses capteurs (accéléromètres, gyromètres, magnétomètres) ainsi qu'une estimation d'orientation calculée via un filtre de Kalman interne. Ce filtre est préconfiguré par le constructeur et n'est pas modifiable.

Problématiques rencontrées L'environnement du bassin génère de fortes perturbations ferromagnétiques (armatures en béton armé, structures métalliques environnantes) :

- **Erreurs de calibration hors bassin :** Contrairement à notre hypothèse initiale, l'éloignement par rapport aux parois s'est avéré insuffisant. Les calibrations effectuées en bassin se sont révélées erronées dès l'utilisation du robot en milieu extérieur.
- **Dérive du cap :** En milieu perturbé, le filtre de Kalman interne génère une dérive continue du cap, y compris lorsque l'AUV reste immobile dans un champ magnétique constant.

Solution développée : Nous avons implémenté notre propre module de détection des perturbations magnétiques (basé sur la norme et l'inclinaison du champ). La confiance accordée à la mesure de cap dans notre propre filtre de Kalman principal est ajustée dynamiquement en fonction de cet indicateur. En bassin, l'estimation du cap repose ainsi principalement sur l'intégration des gyromètres corrigés.

3.1.3 USBL SeaTrac de Blueprint

Principe et protocoles L'USBL communique également via une liaison série avec la Jetson. Contrairement à l'AHRS, son Software Development Kit (SDK) est de très bas niveau et plus complexe, ce qui a nécessité un temps de développement conséquent avant d'aboutir à un nœud d'interface ROS fonctionnel.

Problématiques rencontrées

- **Précision limitée en bassin :** Les essais ont montré une précision médiocre (de l'ordre du mètre). Ces résultats restent à nuancer car les réverbérations acoustiques sur les parois du bassin altèrent le fonctionnement du capteur.
- **Incident matériel :** Une voie d'eau survenue dans le corps du robot a endommagé l'USBL, interrompant prématurément les campagnes de tests avec ce capteur. L'USBL ne sera donc pas utilisé dans la fusion de capteur.

3.1.4 Système de vision avec caméra d'Allied Vision

La chaîne de traitement visuel a été développée par Maxime FERRERA et Adrien CHAUVET. Un premier nœud ROS assure l'acquisition et la publication du flux vidéo de la caméra frontale, tandis qu'un second nœud effectue la détection des mires connues (*AprilTags*) pour estimer la pose relative (position et orientation) du robot par rapport à la station.

Limites et problématiques

- **Portée et précision :** La détection n'est effective que lorsque la caméra se situe à moins de 4 m d'une mire. La précision se dégrade fortement avec la distance, atteignant des incertitudes de plusieurs dizaines de centimètres au-delà de 2 m.

— **Turbidité et éclairage :** Les performances restent dépendantes de la visibilité globale et de la turbidité de l'eau. Pour pallier cette limitation, une évolution envisagée consiste à équiper la base d'amarrage de projecteurs dédiés à l'éclairage direct des mires, en complément des projecteurs embarqués sur l'AUV.

3.2 Estimation d'état

Pour pallier les défaillances individuelles des capteurs et garantir une estimation d'état robuste et continue, une architecture de fusion fondée sur le Filtre de Kalman Étendu (Extended Kalman Filter (EKF)) a été mise en œuvre. L'EKF est un algorithme populaire, largement utilisé et avec un faible coût computationnel. Nous n'avons pas testé d'autres algorithmes d'estimation d'état.

Cette étape du développement d'un AUV est l'une des plus cruciales : si nous n'arrivons pas à estimer correctement l'état du robot, toutes les autres couches (guidage, contrôle, etc.) seront également faussées. Nous avons donc consacré un temps non négligeable à corriger les problématiques des capteurs et à paramétrer le filtre de Kalman afin d'obtenir les meilleurs résultats possibles.

Utilisation du paquet ROS *robot_localization* Plutôt que de développer un filtre de Kalman nous-mêmes, nous avons utilisé *robot_localization* [7]. C'est un paquet conçu pour simplifier la fusion de données capteurs sous ROS, permettant de fusionner un nombre arbitraire de capteurs et d'estimer l'état de systèmes évoluant en 2D ou en 3D.

Son principal avantage réside dans le fait qu'il ne nécessite pas de manipuler directement les équations mathématiques de l'EKF :

— **Configuration simplifiée :** Le réglage du filtre, l'ajout des capteurs à fusionner et la modification des paramètres s'effectuent simplement via un fichier de configuration.

— **Respect des standards ROS :** Ce paquet n'accepte que quatre types de messages, et chaque message doit impérativement être horodaté et inclure la matrice de covariance associée au capteur.

robot_localization suppose un modèle cinématiques (vitesses constantes) pour la phase de prédiction de l'EKF. Cette approximation entraîne une dérive rapide de l'estimation en cas de perte prolongée des signaux capteurs, mais s'avère amplement suffisante en fonctionnement nominal.

Mise en conformité des données et repères Afin de respecter ces exigences, le travail a consisté à :

— **Créer un nœud ROS dédié :** Ce nœud reçoit les messages des capteurs, ajuste les covariances dynamiquement si besoin, et les republie dans le formalisme demandé par *robot_localization* ;

— **Définir la géométrie du robot :** Pour que *robot_localization* connaisse les transformations géométriques entre les origines de chaque capteur et l'origine de l'AUV, un fichier Unified Robot Description Format (URDF) contenant l'ensemble de ces informations a été conçu.

3.2.1 Stratégie et choix d'architecture de fusion

Face aux contraintes du bassin et aux limitations de certains capteurs, les choix d'intégration suivants ont été arrêtés :

- **Capteurs intégrés dans la fusion :**
  - Baromètre : Fournit la profondeur absolue (z).
  - DVL : Mesure les vitesses linéaires relatives (v_x, v_y, v_z).
  - AHRS : Fournit l'orientation (roulis et tangage ϕ, θ) ainsi que les vitesses angulaires à haute fréquence.
  - Caméra (Traitement d'image) : Assure le positionnement relatif par rapport aux mires de la base (x, y et cap ψ).
- **Capteurs et données rejetés :**
  - USBL : Écarté en raison d'une défaillance matérielle.
  - Accéléromètres bruts : Non utilisés en double intégration directe pour éviter une dérive importante en quelques secondes.
  - GNSS : Inexploitable en immersion.
  - AHRS interne du robot : Rejetée au profit de la centrale SBG en raison d'une fréquence d'échantillonnage trop faible et d'un bruit important.
- **Gestion spécifique des mesures de la caméra :**
  - Axes filtrés : La profondeur (z) ainsi que le roulis et le tangage (ϕ, θ) issus de la caméra sont ignorés, car déjà mesurés avec précision par le baromètre et l'AHRS SBG.
  - Estimation du cap (ψ) : En raison de la désactivation du magnétomètre en bassin, le cap absolu est corrigé principalement par le nœud de vision.
  - Repère de référence : Les mesures visuelles étant exprimées dans le repère de la base d'amarrage, ce dernier est choisi comme repère global pour nos essais. Un passage au repère Monde absolu nécessiterait d'équiper la base d'une centrale inertielle et de connaître sa position exacte.

Comme nous n'utilisons pas l'USBL, lorsque les mires ne sont pas détectables par l'AUV, le robot évolue en *dead-reckoning* (estime à l'aveugle).

3.3 Résultats de la fusion

Cette section présente les résultats obtenus par la fusion de données, en simulation dans un premier temps, puis lors des essais réels.

3.3.1 En simulation

En simulation sous Gazebo, la présence d'une vérité terrain permet d'évaluer quantitativement l'erreur d'estimation. Cependant, en simulation, nous connaissons parfaitement tous les paramètres des capteurs, leurs bruits et leurs transformations géométriques par rapport à l'origine du robot, ce qui implique que la fusion de capteurs est censée être quasi parfaite.

Nous aurions pu simuler d'autres incertitudes sur les capteurs que du bruit gaussien, mais il était complexe de reproduire fidèlement le bruit exact de certains capteurs, comme celui des

mesures du DVL.

Nous avons néanmoins modifié certains paramètres en simulation, tels que l'alignement DVL-AHRS ou la position du baromètre. Cela nous a permis d'éliminer ces causes potentielles lors de l'identification des erreurs de fermeture de boucle pendant les tests réels.

Résultats On observe sur les Figures 3.1 et 3.2 que l'estimation d'état est quasi parfaite (les courbes se confondent), malgré le bruit gaussien appliqué sur les capteurs. La position et le cap sont mis à jour grâce au positionnement visuel vers $t = 112\,\mathrm{s}$.

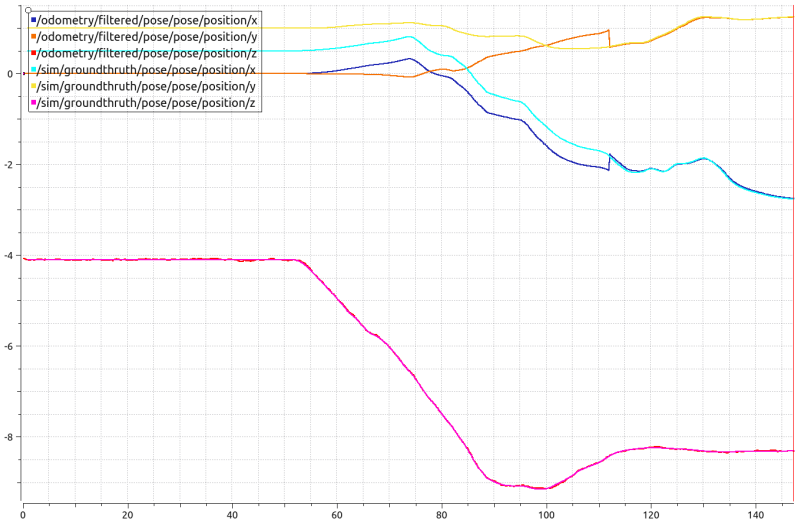


FIGURE 3.1 – Valeurs réelles de la position sur les 3 axes vs valeurs estimées par l'EKF (axe x en s, axe y en m).

FIGURE 3.2 – Valeurs réelles de l'orientation sur les 3 axes vs valeurs estimées par l'EKF (axe x en s, axe y en rad).

La simulation nous a donc principalement permis de valider le bon paramétrage des transformations géométriques des capteurs ainsi que la configuration de *robot_ localization*.

3.3.2 Tests réels

En l'absence de vérité terrain lors de nos tests en bassin, il a été difficile d'évaluer avec précision l'efficacité absolue de notre fusion de capteurs.

Nous avons donc réalisé des trajectoires avec fermeture de boucle afin d'estimer les erreurs de notre filtre. Le but de ces tests était de caractériser les erreurs et bruits des capteurs pour paramétrer l'EKF de la manière la plus optimale possible.

Tous ces tests ont été réalisés sans la mesure de position fournie par la caméra et avec un DVL qui a été identifié ultérieurement comme défaillant (sans savoir s'il l'était déjà au moment des essais). La caractérisation des capteurs et de la fusion de données a en effet été réalisée relativement tôt durant le stage, avant que la caméra ne soit intégrée à l'écosystème ROS. De nouveaux tests devront donc être menés pour qualifier la fusion de capteurs complète.

Nous avons réalisé 3 types de trajectoires : des cercles, des allers-retours en ligne droite et des parallélogrammes. Toutes ces trajectoires restent approximatives car le robot était téléopéré par un utilisateur n'ayant qu'un retour visuel direct sur le bassin. À chaque essai, la procédure reste identique : amarrage manuel à la base, démarrage du filtre de Kalman et de l'enregistrement des données, désamarrage, réalisation de la trajectoire, nouvel amarrage à la base, puis arrêt de l'enregistrement.

Trois essais représentatifs sont détaillés ci-après, comparant à chaque fois deux configurations de filtres :

- **EKF basique :** intègre la mesure de cap brute fournie par l'AHRS et ne filtre quasiment pas les données du DVL ;
- **EKF corrigé :** s'appuie uniquement sur les gyromètres pour estimer le cap (les biais ayant été préalablement estimés et compensés) et applique un filtre de rejet (distance de Mahalanobis) sur les données du DVL.

La métrique d'évaluation retenue est l'erreur relative de fermeture de boucle, exprimée sous la forme du pourcentage de l'écart géométrique final rapporté à la distance totale parcourue.

Une étude statistique globale menée sur l'ensemble de nos tests ainsi que ses conclusions sont synthétisées dans le Tableau 3.1.

Trajectoire de type cercle La trajectoire en cercle est celle où les erreurs de cap et du DVL ont eu le moins d'impact, comme l'illustre la Figure 3.3. Cela peut être dû à la zone du bassin où le test a été réalisé (relativement éloignée des murs et des structures métalliques) ainsi qu'à la géométrie même de la trajectoire.

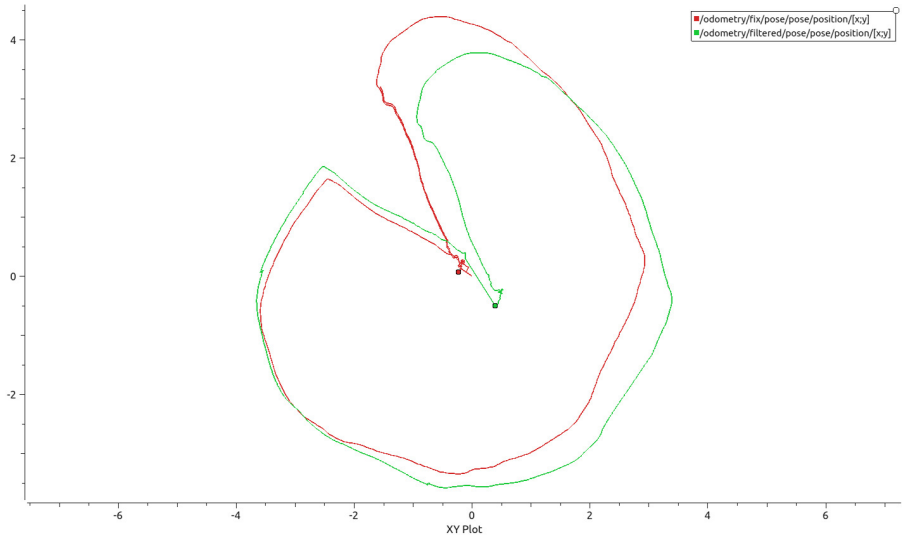


FIGURE 3.3 – Comparaison de la trajectoire en cercle vue du dessus (axes x et y en m) entre l'EKF basique (vert) et l'EKF corrigé (rouge).

On obtient une erreur de 2,33% avec la fusion de capteurs basique contre 0,77% avec la version corrigée.

Trajectoire de type parallélogramme La trajectoire en parallélogramme est celle qui met le plus en évidence les erreurs de cap. C'est ici que les différences entre les deux filtres sont les plus flagrantes (voir la Figure 3.4), ce qui nous a permis de confirmer que l'information de cap fournie par l'AHRS était erronée.

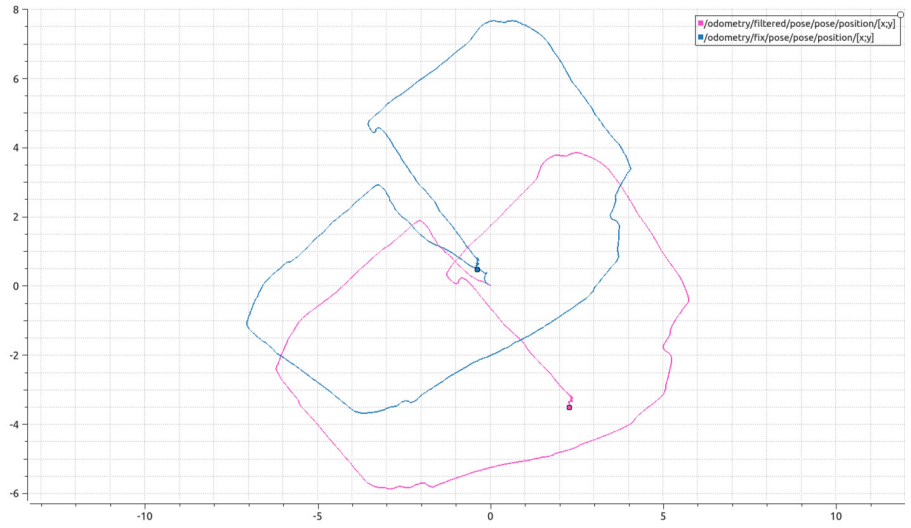


FIGURE 3.4 – Comparaison de la trajectoire en parallélogramme vue du dessus (axes x et y en m) entre l'EKF basique (rose) et l'EKF corrigé (bleu).

On obtient une erreur de 9,52% sur la fusion basique contre seulement 1,07% avec la version corrigée.

Trajectoire de type aller-retour en ligne droite La trajectoire en aller-retour permet de limiter l'influence des erreurs de cap afin d'isoler d'autres dysfonctionnements potentiels. Lors des différents essais (exemple sur la Figure 3.5), nous ne parvenions pas à obtenir une fermeture de boucle satisfaisante sur ce parcours.

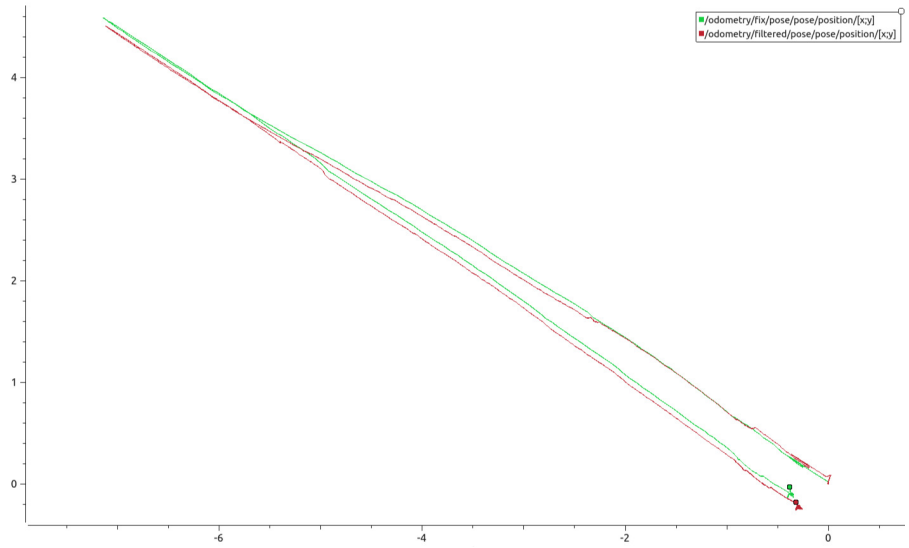


FIGURE 3.5 – Comparaison de la trajectoire en aller-retour vue du dessus (axes x et y en m) entre l'EKF basique (rouge) et l'EKF corrigé (vert).

L'utilisation exclusive des gyromètres et le filtrage du DVL n'améliorent pas les résultats. Nous avons réalisé que le DVL présentait des biais de mesure non constants et non répétables.

Étant placés dans des conditions non optimales (bassin) et obtenant des résultats néanmoins exploitables pour de l'estime sur de courtes distances, nous n'avons pas cherché à résoudre davantage ce problème.

De plus, il est prévu d'ajouter des caméras stéréoscopiques sous l'AUV afin d'effectuer de l'odométrie visuelle, ce qui fournira des mesures de vitesse complémentaires pour fiabiliser la fusion.

On obtient une erreur de 2,03% sur la fusion de capteurs basique contre 1,91% avec la version corrigée.

Étude statistique de l'ensemble des essais et conclusion

Comme indiqué précédemment et étayé par les données du Tableau 3.1, on observe une nette amélioration de l'estimation d'état lorsque le cap fourni par l'AHRS est ignoré au profit des gyromètres et que les données DVL sont filtrées.

Le constructeur du DVL annonce une précision à long terme de ±1,01%. Avec une erreur moyenne mesurée de notre côté à ±0,89%, nous pouvons estimer que l'erreur résiduelle provient principalement des imperfections de mesure de ce capteur.

TABLE 3.1 – Comparaison des performances entre l'EKF basique et l'EKF corrigé.

Test réalisé	**Erreur EKF basique (%)**	**Erreur EKF corrigé (%)**
a) 2 cercles sens horaire	0,39	0,29
b) 3 cercles sens horaire	0,57	0,30
c) 3 cercles sens trigo	0,31	0,18
d) Parallélogramme sens horaire	4,82	1,47
e) Parallélogramme sens trigo	7,34	0,58
f) Aller-retour ligne droite	1,56	1,78
g) Aller-retour ligne droite	2,04	1,92
h) Translation gauche puis droite	0,99	0,69
i) Cercle sens trigo	2,33	0,78
j) Parallélogramme sens trigo	9,52	1,07
k) Parallélogramme sens trigo	8,98	0,76
Moyenne cercles	0,90	0,39
Moyenne parallélogrammes	7,66	0,97
Moyenne allers-retours	1,53	1,46
Moyenne totale	**3,53**	**0,89**

Au vu du faible nombre de capteurs disponibles (aucune redondance sur l'estimation de la position ou de la vitesse selon les axes x et y), il s'avère difficile de réduire davantage l'impact de ces biais. Les autres sources d'incertitude principales résident dans l'alignement géométrique des capteurs, les biais du gyroscope et le bruit de mesure global.

Dans l'ensemble, nous sommes parvenus à mettre en place une estimation d'état fonctionnelle. Bien qu'elle reste perfectible, nous avons choisi de ne pas consacrer plus de temps au réglage du filtre afin de privilégier le développement des autres briques algorithmiques. Les capteurs n'ayant pas été évalués dans un environnement nominal (essais en bassin), la phase de réglage de la fusion de données devra nécessairement être réitérée lors des essais à la mer.

Chapitre 4

Guidage

Ce chapitre est consacré à l'élaboration des lois de guidage pour les phases de suivi de trajectoire et d'amarrage/désamarrage.

4.1 Guidage et suivi de trajectoire

Le suivi de trajectoire n'étant pas le sujet principal de mon stage, j'ai choisi d'utiliser l'algorithme de poursuite pure (*pure pursuit algorithm*), qui est une méthode éprouvée. Il permet de générer des consignes cinématiques afin de guider l'AUV le long des segments reliant les différents points de passage (*waypoints*), comme le représente le schéma de la Figure 4.1.

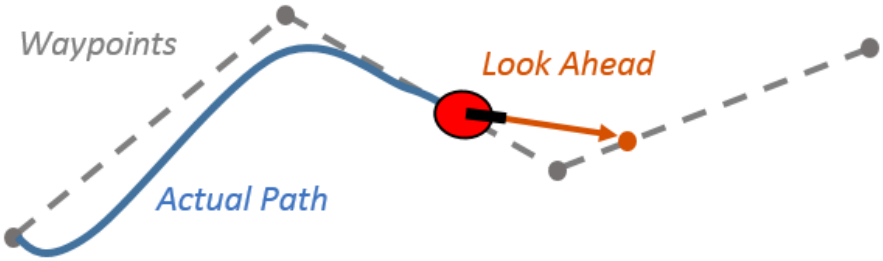


FIGURE 4.1 – Schéma explicatif de l'algorithme de poursuite pure (*pure pursuit*).

4.1.1 Gestion de la trajectoire et commutation de *waypoint*

Soit une liste ordonnée de N *waypoints* $W = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N-1}\}$ avec $\mathbf{p}_i \in \mathbb{R}^3$. La transition de l'indice courant vers le segment suivant $(\mathbf{p}_A, \mathbf{p}_B) = (\mathbf{p}_k, \mathbf{p}_{k+1})$ s'effectue dès que la distance euclidienne entre la position réelle de l'AUV $\mathbf{p}_R = \boldsymbol{\eta}_{1..3}$ et le point cible $\mathbf{p}_B$ passe sous le rayon de la sphère d'acceptation R_{accept} :

$$\|\mathbf{p}_R - \mathbf{p}_B\| < R_{\text{accept}} \implies k \leftarrow k + 1 \tag{4.1}$$

4.1.2 Projection géométrique et point visé (*Look-ahead*)

Soient $\mathbf{p}_A, \mathbf{p}_B \in \mathbb{R}^3$ les coordonnées des deux *waypoints* définissant le segment courant. Le vecteur directeur unitaire de la droite s'écrit :

$$\mathbf{u}_{\text{line}} = \frac{\mathbf{p}_B - \mathbf{p}_A}{\|\mathbf{p}_B - \mathbf{p}_A\| + \varepsilon} \tag{4.2}$$

La position projetée de l'AUV $\boldsymbol{\eta}_{1..3} = [x, y, z]^T$ sur le segment est donnée par le produit scalaire :

$$x_{\text{local}} = (\boldsymbol{\eta}_{1..3} - \mathbf{p}_A) \cdot \mathbf{u}_{\text{line}} \tag{4.3}$$

Le point cible visé $\mathbf{p}_{\text{target}}$ est déterminé en avançant d'une distance de visée L_d (*look-ahead*), puis est saturé sur la longueur du segment $[0, \|\mathbf{p}_B - \mathbf{p}_A\|]$:

$$\mathbf{p}_{\text{target}} = \mathbf{p}_A + \text{clip}\left(x_{\text{local}} + L_d,\, 0,\, \|\mathbf{p}_B - \mathbf{p}_A\|\right) \mathbf{u}_{\text{line}} \tag{4.4}$$

Le vecteur de poursuite 3D reliant le robot au point visé est défini par :

$$\mathbf{v}_{\text{pursuit}} = \mathbf{p}_{\text{target}} - \boldsymbol{\eta}_{1..3} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \tag{4.5}$$

4.1.3 Consignes d'orientation (Cap et Tangage)

Consigne de cap (ψ_d) : Le cap brut est déterminé dans le plan horizontal (x, y) par :

$$\psi_{\text{raw}} = \text{atan2}(e_y, e_x) \tag{4.6}$$

Un filtre passe-bas du premier ordre de coefficient $\alpha_\psi \in]0, 1]$ est appliqué avec prise en compte de la discontinuité 2π via la fonction dent de scie (*sawtooth*) :

$$\psi_d^{(k)} = \psi_d^{(k-1)} + \alpha_\psi \cdot \text{sawtooth}\left(\psi_{\text{raw}} - \psi_d^{(k-1)}\right) \tag{4.7}$$

Consigne de tangage (θ_d) : L'angle d'assiette brut est calculé dans le plan vertical, puis filtré et bridé par la saturation mécanique $\theta_{\max}$:

$$\theta_{\text{raw}} = -\text{atan2}\left(e_z,\, \sqrt{e_x^2 + e_y^2} + \varepsilon\right) \tag{4.8}$$

$$\theta_d^{(k)} = \text{clip}\left(\theta_d^{(k-1)} + \alpha_\theta \left(\theta_{\text{raw}} - \theta_d^{(k-1)}\right),\, -\theta_{\max},\, \theta_{\max}\right) \tag{4.9}$$

Cette saturation permet d'éviter de générer des consignes associées à un angle de tangage trop important, ce qui pourrait provoquer des instabilités.

4.1.4 Génération du profil de vitesse et intégration de consigne

Afin d'éviter des virages trop brusques en cas de fort désalignement, la vitesse linéaire est pondérée par un facteur d'alignement k_{align} basé sur l'erreur de cap $\tilde{\psi} = \text{sawtooth}(\psi_{\text{raw}} - \psi_{\text{réel}})$:

$$k_{\text{align}} = \max\left(0,\, \cos(\tilde{\psi})\right)^3 \tag{4.10}$$

La vitesse désirée modulée s'exprime alors par :

$$v_{d,\text{prop}} = v_{\text{desired}} \cdot \tanh(d) \cdot k_{\text{align}} \quad \text{avec} \quad d = \min\left(d_{\text{start}},\, d_{\text{target}}\right) \tag{4.11}$$

Les composantes de la vitesse linéaire dans le repère global $\mathbf{v}_d = [\dot{x}_d, \dot{y}_d, \dot{z}_d]^T$ sont obtenues par projection trigonométrique :

$$\begin{cases} \dot{x}_d = v_{d,\text{prop}} \cdot \cos(\psi_d)\cos(\theta_d) \\ \dot{y}_d = v_{d,\text{prop}} \cdot \sin(\psi_d)\cos(\theta_d) \\ \dot{z}_d = -v_{d,\text{prop}} \cdot \sin(\theta_d) \end{cases} \tag{4.12}$$

La nouvelle consigne de position est intégrée au cours du temps. Un mécanisme de sécurité bride la distance maximale $d_{\max}$ autorisée entre la consigne intégrée $\mathbf{p}_{\text{int}}$ et la position réelle du robot $\mathbf{p}_R$, empêchant l'emballement des erreurs si le robot subit un retard important :

$$\mathbf{p}_{\text{int}} = \mathbf{p}_d^{(k-1)} + \mathbf{v}_d \cdot \Delta t \tag{4.13}$$

$$\mathbf{p}_d^{(k)} = \begin{cases} \mathbf{p}_{\text{int}} & \text{si } \|\mathbf{p}_{\text{int}} - \mathbf{p}_R\| \leq d_{\max} \\ \mathbf{p}_R + \dfrac{\mathbf{p}_{\text{int}} - \mathbf{p}_R}{\|\mathbf{p}_{\text{int}} - \mathbf{p}_R\|} \cdot d_{\max} & \text{si } \|\mathbf{p}_{\text{int}} - \mathbf{p}_R\| > d_{\max} \end{cases} \tag{4.14}$$

L'algorithme génère et retourne finalement le vecteur de pose désiré $\boldsymbol{\eta}_d$ combinant la position filtrée et l'orientation calculée, ainsi que le vecteur des vitesses dérivées $\dot{\boldsymbol{\eta}}_d$:

$$\boldsymbol{\eta}_d = \begin{bmatrix} \mathbf{p}_d^{(k)} \\ 0 \\ \theta_d^{(k)} \\ \psi_d^{(k)} \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ z_d \\ 0 \\ \theta_d \\ \psi_d \end{bmatrix}, \quad \dot{\boldsymbol{\eta}}_d = \begin{bmatrix} \mathbf{v}_d \\ \mathbf{0}_{3\times 1} \end{bmatrix} = \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{z}_d \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.15}$$

4.2 Guidage d'amarrage et désamarrage

Pour la phase d'amarrage, la précision du guidage est d'une importance primordiale.

Le guidage par champs de vecteurs est une méthode reconnue pour sa garantie de convergence (et donc sa robustesse face aux perturbations), garantissant des commandes continues et fluides pour un faible coût algorithmique. L'amarrage étant la phase la plus critique — exposant le sous-marin aux risques de collision et nécessitant une forte réjection des perturbations sans

mouvements brusques —, c'est cette méthode que nous avons choisi d'implémenter pour notre AUV.

L'objectif du processus d'amarrage est le suivant : l'AUV doit contourner la station, s'aligner progressivement selon un axe prédéfini, positionner sa mâchoire au centre du poteau d'amarrage, puis descendre le long de celui-ci jusqu'au verrouillage final.

Le champ de vecteurs conçu est ainsi composé de trois forces (voir Figure 4.2 pour une représentation graphique) :

1. **Une sphère de répulsion :** Définie autour du dock, elle empêche l'AUV de s'approcher directement de la cible tant qu'il n'est pas positionné dans le bon axe, évitant ainsi toute collision latérale.
2. **Une force de glissement :** Elle force le robot à contourner la sphère de sécurité en glissant le long de sa surface pour se placer progressivement dans l'axe d'approche.
3. **Un champ d'attraction axial :** Dès que l'AUV est suffisamment aligné avec l'axe d'approche du dock, le champ attractif devient prépondérant et le guide vers le poteau d'amarrage.

Le passage progressif entre le mode de contournement (répulsion + glissement) et le mode d'attraction axiale est lissé par une fonction de pondération basée sur l'erreur d'alignement.

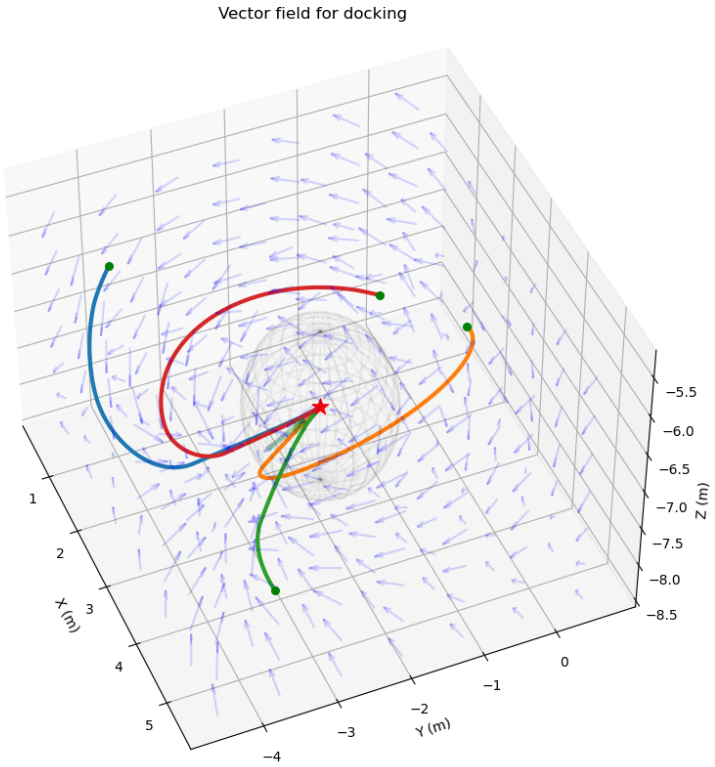


FIGURE 4.2 – Champ de vecteurs avec exemples de trajectoires.

Une fois le robot stabilisé face au poteau d'amarrage, l'algorithme valide la transition vers la phase de descente verticale le long de celui-ci, jusqu'au verrouillage mécanique.

4.2.1 Machine à états et stratégie d'amarrage

La procédure d'amarrage se décompose en trois phases :

— **Phase 0 (Approche et Alignement) :** L'AUV suit le champ de vecteurs (voir Figure 4.2) et rejoint la position visée (poteau au centre de la mâchoire).

— **Phase 1 (Descente verticale) :** Une fois l'alignement horizontal validé face au poteau, le robot verrouille ses consignes x et y. Il effectue une descente purement verticale jusqu'au réceptacle de la station, tout en régulant son assiette (*pitch*) pour éviter tout coincement mécanique.

— **Phase 2 (Verrouillage & Fin) :** L'AUV est correctement positionné dans le réceptacle, les consignes de vitesse sont coupées et la mission d'amarrage est validée.

L'orientation et la position absolues de la base n'étant pas connues a priori, le guidage s'effectue directement dans son repère local.

Le vecteur d'erreur positionnelle $\mathbf{e}$, la distance d et la direction unitaire $\mathbf{u}_{\text{dir}}$ sont définis par :

$$\mathbf{e} = \mathbf{p} - \mathbf{p}_{\text{target}}, \quad d = \|\mathbf{e}\| + \varepsilon, \quad \mathbf{u}_{\text{dir}} = \frac{\mathbf{e}}{d} \tag{4.16}$$

4.2.2 Génération du champ de vitesse (Phase 0 : Approche)

Soit $\mathbf{u}_{\text{axis}}$ le vecteur unitaire de l'axe d'approche. L'alignement relatif de l'AUV est donné par $a = \mathbf{u}_{\text{dir}} \cdot \mathbf{u}_{\text{axis}}$. La fonction de pondération d'attraction $w(a) \in [0, 1]$ repose sur une fonction sigmoïde :

$$w(a) = \frac{1}{1 + \exp\left(-\gamma_{\text{stiff}} \cdot (a - a_{\text{thresh}})\right)} \tag{4.17}$$

Les composantes du champ de forces artificiel s'expriment par :

1. **Force d'attraction axiale :**

$$\mathbf{f}_{\text{attractor}} = k_{\text{attractor}} \cdot \mathbf{e} \tag{4.18}$$

2. **Forces de la sphère de sécurité (Répulsion et Glissement) :**

$$\mathbf{f}_{\text{repulsion}} = k_{\text{repulsion}}(R_{\text{sphere}} - d) \cdot \mathbf{u}_{\text{dir}} \tag{4.19}$$

$$\mathbf{f}_{\text{gliding}} = k_{\text{gliding}} \left(\mathbf{u}_{\text{axis}} - (\mathbf{u}_{\text{axis}} \cdot \mathbf{u}_{\text{dir}})\mathbf{u}_{\text{dir}}\right) \tag{4.20}$$

$$\mathbf{f}_{\text{sphere}} = \mathbf{f}_{\text{repulsion}} + \mathbf{f}_{\text{gliding}} \tag{4.21}$$

Le champ de vitesse brut $\mathbf{v}_{\text{raw}}$ résulte de la combinaison de ces forces. Deux régimes particuliers sont toutefois introduits :

— **Protection anti-collision inférieure :** Si l'AUV évolue à une immersion trop importante par rapport à la cible à proximité de la station, une force de rappel verticale exécute une remontée d'urgence pour prévenir tout impact avec la structure.

— **Convergence terminale :** À très courte distance de la cible et sous réserve d'un alignement satisfaisant, la dynamique est simplifiée au profit d'une force attractive pure afin d'assurer une convergence fluide et d'éviter les oscillations résiduelles.

Dans les autres configurations, la vitesse brute provient de la combinaison de $\mathbf{f}_{\text{attractor}}$ et $\mathbf{f}_{\text{sphere}}$:

$$\mathbf{v}_{\text{raw}} = \begin{cases} \begin{bmatrix} 0 & 0 & k_{\text{floor}}(z_{\text{target}} + z_{\text{limit}} - z) \end{bmatrix}^T & \text{si } z < z_{\text{target}} + z_{\text{limit}} \text{ et } d < 4{,}0\,\text{m} \\ k_{\text{point}} \cdot \mathbf{e} & \text{si } d < 0{,}3\,\text{m et } a > 0{,}90 \\ w(a) \cdot \mathbf{f}_{\text{attractor}} + (1 - w(a)) \cdot \mathbf{f}_{\text{sphere}} & \text{sinon} \end{cases} \tag{4.22}$$

4.2.3 Génération du champ de vitesse (Phase 1 : Descente verticale)

En Phase 1, les coordonnées horizontales désirées sont immédiatement verrouillées sur la position du réceptacle ($x_d = x_{\text{dock}}, y_d = y_{\text{dock}}$). La vitesse de descente selon z est modulée par un facteur de pénalité f_{pitch} basé sur l'erreur d'assiette $\tilde{\theta} = |\theta - \theta_{\text{dock}}|$, évitant ainsi de forcer la descente si le robot se cabre :

$$f_{\text{pitch}} = \begin{cases} 1{,}0 & \text{si } d < 0{,}2\,\text{m} \\ \exp\left(-\gamma_{\text{pitch}} \cdot \tilde{\theta}\right) & \text{sinon} \end{cases} \tag{4.23}$$

$$v_{z,\text{unscaled}} = \text{clip}\left(k_{\text{point}}(z - z_{\text{dock}}),\, -v_{\max},\, v_{\max}\right), \quad \mathbf{v}_{\text{raw}} = \begin{bmatrix} 0 & 0 & v_{z,\text{unscaled}} \cdot f_{\text{pitch}} \end{bmatrix}^T \tag{4.24}$$

4.2.4 Saturations dynamiques et limitation de retard

Quelle que soit la phase, la vitesse maximale autorisée $v_{\max,\text{d}}$ varie selon la distance d :

$$v_{\max,\text{d}} = \begin{cases} \max\left(0{,}015,\, v_{\max} \cdot \tanh(d/3{,}0)\right) & \text{si phase } = 0 \\ \max\left(0{,}015,\, v_{\max} \cdot \tanh(1{,}2 \cdot d)\right) & \text{si phase } \geq 1 \end{cases} \tag{4.25}$$

Après saturation de la norme de $\mathbf{v}_{\text{raw}}$ vers $\mathbf{v}_{\text{sat}}$, la vitesse désirée $\mathbf{v}_d^{(k)}$ est bridée par l'accélération maximale $a_{\max}$ afin d'éviter les sauts de consigne brutaux sur les propulseurs :

$$\Delta\mathbf{v} = \mathbf{v}_{\text{sat}} - \mathbf{v}_d^{(k-1)}, \quad \delta_v = \|\Delta\mathbf{v}\| + \varepsilon \tag{4.26}$$

$$\mathbf{v}_d^{(k)} = \begin{cases} \mathbf{v}_{\text{sat}} & \text{si } \delta_v \leq a_{\max}\Delta t \\ \mathbf{v}_d^{(k-1)} + \dfrac{\Delta\mathbf{v}}{\delta_v} \cdot a_{\max}\Delta t & \text{sinon} \end{cases} \tag{4.27}$$

Enfin, la consigne de position $\mathbf{p}_d^{(k)}$ est intégrée et maintenue à proximité immédiate de l'AUV de manière similaire à celle du suivi de chemin (Équations (4.13) et (4.14)).

4.2.5 Consignes de sortie et conditions de transition

L'algorithme génère et transmet au contrôleur bas niveau le vecteur de pose $\boldsymbol{\eta}_d$ et le vecteur de vitesses $\dot{\boldsymbol{\eta}}_d$:

$$\boldsymbol{\eta}_d = \begin{bmatrix} x_d^{(k)} \\ y_d^{(k)} \\ z_d^{(k)} \\ 0 \\ \theta_{\text{dock}} \\ \psi_{\text{dock}} \end{bmatrix}, \quad \dot{\boldsymbol{\eta}}_d = \begin{bmatrix} v_{x,d}^{(k)} \\ v_{y,d}^{(k)} \\ v_{z,d}^{(k)} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.28}$$

Les transitions de phase s'effectuent selon les conditions d'erreurs de position et d'orientation.

4.2.6 Logique du guidage de désamarrage

Les équations du désamarrage étant très proches de certaines phases de l'amarrage, elles ne sont pas détaillées à nouveau ici.

Pour se désamarrer, le robot effectue une phase d'ascension verticale guidée (correspondant à la Phase 1 d'amarrage exécutée en sens inverse), puis recule pour rejoindre un point de sortie donné.

Ce point de sortie est choisi de manière à éviter toute collision avec la base, ce qui rend superflu l'usage d'une sphère de répulsion similaire à celle de la phase d'amarrage.

4.2.7 Conclusion

Une méthode de guidage fiable et robuste a ainsi été établie. L'ensemble des cas limites n'a pas pu être traité dans le cadre de ce travail, et il conviendrait à terme de développer un arbre de décision couvrant chaque scénario de défaillance.

Cependant, l'élaboration d'un tel système requiert une longue campagne d'essais afin de répertorier les modes de défaillance possibles, d'y associer des stratégies de repli et de les valider expérimentalement. Cela dépassait le cadre de ce stage ; l'effort s'est donc concentré sur le mode de fonctionnement nominal en concevant une méthode de guidage minimisant les risques de situations critiques pour l'AUV.

Les performances des algorithmes de guidage étant indissociables de celles de la commande, les résultats globaux seront analysés conjointement dans le chapitre 6.

Chapitre 5

Contrôle

Ce chapitre est consacré à l'élaboration des lois de commande de l'AUV. Un contrôleur unique est exploité pour l'ensemble des modes de fonctionnement (suivi de trajectoire, amarrage, désamarrage, etc.).

Le rôle de ce contrôleur est d'asservir le système : il calcule et ajuste en continu les commandes envoyées aux moteurs afin que l'AUV suive fidèlement les consignes générées (trajectoire, vitesse, position et attitude).

5.1 Choix de l'architecture de contrôle

Actuellement, quatre méthodes de contrôle sont principalement employées pour les AUV [8, 9] :

— Le contrôle proportionnel-intégral-dérivé (Proportionnel Intégral Dérivé (PID)) ;
— La commande par mode glissant (Sliding Mode Control (SMC)) ;
— La commande prédictive basée sur un modèle (Model Predictive Control (MPC)) ;
— La commande par réseaux de neurones.

Les principaux avantages et inconvénients de ces méthodes sont synthétisés dans le Tableau 5.1.

5.1.1 Analyse comparative et sélection

Dans le cas d'un AUV résident, les incertitudes et les perturbations environnementales sont nombreuses et évoluent au cours du temps (encrassement biologique, sédiments, courants marins, défaillances matérielles). Le régulateur PID a donc été écarté en raison de sa faible robustesse face à ces variations.

Bien que la commande prédictive (MPC) soit particulièrement performante pour des phases de manœuvre de haute précision telles que l'amarrage, cet algorithme nécessite une connaissance complète des états du robot et, idéalement, une mesure ou une estimation précise des perturbations extérieures. La modélisation dynamique disponible reposant sur des approximations significatives, cette méthode n'a pas été retenue.

Les approches basées sur les réseaux de neurones et l'apprentissage par renforcement représentent des alternatives modernes et efficaces. Toutefois, en raison de la faible précision du

TABLE 5.1 – Synthèse comparative des méthodes de contrôle pour AUV.

Méthode	Avantages	Inconvénients
PID	Structure simple. Facilité d'implémentation et de réglage.	Performances limitées sur les systèmes non linéaires. Faible robustesse aux perturbations.
SMC	Temps de réponse rapide. Forte robustesse aux incertitudes du modèle et aux perturbations.	Phénomène de *chattering*.
MPC	Gestion explicite des contraintes. Excellente précision de poursuite.	Coût calculatoire élevé. Nécessite un modèle dynamique très précis.
Réseaux de neurones	Compensateur d'incertitudes très efficace. Adaptabilité aux dynamiques complexes.	Complexité d'apprentissage en temps réel. Preuve de convergence complexe.

modèle initial et de la complexité de mise en œuvre de ces méthodes, nous n'avons pas retenu cet algorithme.

La commande par mode glissant (SMC) classique est une méthode éprouvée, reconnue pour sa forte robustesse face aux erreurs de modélisation et aux perturbations. Elle garantit théoriquement la convergence de l'état du système vers une surface de glissement en temps fini. Néanmoins, cette méthode induit fréquemment du broutement (*chattering*, commutation rapide des moteurs) au voisinage de la surface de glissement. Ces oscillations à haute fréquence sollicitent brutalement les actionneurs, ce qui accélère l'usure mécanique et augmente la consommation électrique. De plus, les moteurs ne pouvant pas suivre ces variations rapides, ce phénomène peut générer des instabilités dans la boucle de régulation.

5.1.2 Option retenue : l'algorithme Super-Twisting (STA)

Afin de pallier l'effet de *chattering* tout en conservant les propriétés de robustesse de la commande par mode glissant, plusieurs extensions ont été développées dans la littérature. Parmi elles, l'algorithme Super-Twisting Algorithm (STA) est une commande par mode glissant d'ordre 2 introduite par Levant (1993) [10]. Cet algorithme préserve la robustesse du SMC classique tout en lissant le signal de commande, réduisant ainsi drastiquement le phénomène de *chattering*.

C'est donc le contrôleur STA qui a été choisi et implémenté pour l'asservissement de l'AUV X-300. Bien qu'il existe des variantes plus avancées du STA [11], une version simple a été privilégiée afin de faciliter son intégration et d'obtenir rapidement de bons résultats expérimentaux.

5.2 Équations du STA

5.2.1 Définition de l'erreur et surface de glissement

Soient $\eta_d(t) \in \mathbb{R}^6$ la pose souhaitée et $\eta(t) \in \mathbb{R}^6$ la pose mesurée de l'AUV. L'erreur $e(t)$ et sa dérivée temporelle $\dot{e}(t)$ sont définies par :

$$e(t) = \eta_d(t) - \eta(t), \quad \dot{e}(t) = \dot{\eta}_d(t) - \dot{\eta}(t) \tag{5.1}$$

La surface de glissement $\sigma(t) \in \mathbb{R}^6$ est choisie comme :

$$\sigma(t) = T \cdot e(t) + \dot{e}(t) \tag{5.2}$$

où $T = \text{diag}(\gamma_1, \gamma_2, \ldots, \gamma_6) > 0$ est une matrice de gains définie positive paramétrant la dynamique de convergence sur la surface.

Bien que d'autres surfaces de glissement existent, ce choix s'appuie sur la formulation la plus couramment adoptée dans la littérature.

5.2.2 Loi de commande STA

La loi de commande STA $\tau_{STA} \in \mathbb{R}^6$ s'articule autour de deux termes : une composante proportionnelle non-linéaire et une composante intégrale λ :

$$\tau_{STA} = -K_1\sqrt{|\sigma|}\text{sign}(\sigma) + \lambda \tag{5.3}$$

$$\dot{\lambda} = -\frac{1}{2}K_2\text{sign}(\sigma) \tag{5.4}$$

où $K_1 = \text{diag}(k_{11}, \ldots, k_{16})$ et $K_2 = \text{diag}(k_{21}, \ldots, k_{26})$ sont les matrices de gains constants du STA, à régler empiriquement.

Quasi-Super Twisting Bien que le STA soit conçu pour éliminer le *chattering*, en pratique des oscillations restent présentes dans la commande. Pour limiter ce phénomène lors des applications sur des robots réels, on fait l'approximation suivante :

$$\text{sign}(s) \approx \tanh\left(\frac{s}{\delta}\right) \tag{5.5}$$

où δ est un scalaire positif. δ doit être choisi de manière à trouver un compromis entre lissage de la commande et performance.

Le *Quasi-Super Twisting* perd la propriété de convergence vers la surface de glissement en un temps donné, mais présente une convergence asymptotique vers une zone bornée (proche de la surface de glissement)[12].

Si δ est bien choisi, les performances du *Quasi-Super Twisting* sont très proches de celles du STA. Dans la suite de ce rapport, l'algorithme utilisé sera simplement désigné par STA par souci de simplicité.

5.2.3 Calcul de la commande STA

La dynamique complète de l'AUV exprimée directement dans le repère fixe terrestre η (autre forme de l'équation de Fossen (2.8)) s'écrit :

$$M_\eta(\eta)\ddot{\eta} + C_\eta(\nu,\eta)\dot{\eta} + D_\eta(\nu,\eta)\dot{\eta} + g_\eta(\eta) = \tau_\eta \tag{5.6}$$

avec les matrices projetées via $J(\eta)$, définie ici (2.5) :

$$M_\eta(\eta) = J^{-T}(\eta) M_{\text{tot}} J^{-1}(\eta), \quad \text{avec } M_{\text{tot}} = M_A + M_{RB} \tag{5.7}$$

$$C_\eta(\nu,\eta) = J^{-T}(\eta)\left(C(\nu) - M_{\text{tot}} J^{-1}(\eta)\dot{J}(\eta)\right) J^{-1}(\eta) \tag{5.8}$$

$$D_\eta(\nu,\eta) = J^{-T}(\eta) D(\nu) \tag{5.9}$$

$$g_\eta(\eta) = J^{-T}(\eta) g(\eta) \tag{5.10}$$

$$\tau_\eta = J^{-T}(\eta)\tau \tag{5.11}$$

On pose :

$$\ddot{\eta}(t) = F(\eta,\dot{\eta},\nu) + G(\eta)\tau(t) \tag{5.12}$$

où :

— $F(\eta,\dot{\eta},\nu) = -\mathbf{M}_\eta^{-1}(\eta)\left[\mathbf{C}_\eta(\nu,\eta)\dot{\eta} + \mathbf{D}_\eta(\nu,\eta)\dot{\eta} + \mathbf{g}_\eta(\eta)\right]$,

— $G(\eta) = \mathbf{M}_\eta^{-1}(\eta)$

En injectant l'équation (5.12) dans la dérivée de la surface de glissement, nous obtenons :

$$\begin{aligned}\dot{\sigma}(t) &= \mathbf{T}\dot{\mathbf{e}} + \ddot{\mathbf{e}} \\ &= \mathbf{T}\dot{\mathbf{e}} + \ddot{\eta}_d - \ddot{\eta} \\ &= \mathbf{T}\dot{\mathbf{e}} + \ddot{\eta}_d - [F(\eta,\dot{\eta},\nu) + G(\eta)\tau_\eta]\end{aligned} \tag{5.13}$$

Pour imposer la dynamique du STA, on égalise cette dérivée avec le terme de commande τ_{STA} :

$$\mathbf{T}\dot{\mathbf{e}} + \ddot{\eta}_d - [F(\eta,\dot{\eta},\nu) + G(\eta)\tau_\eta] = \tau_{STA} \tag{5.14}$$

En isolant la contribution de la commande $G(\eta)\tau_\eta$:

$$G(\eta)\tau_\eta = \ddot{\eta}_d(t) + \mathbf{T}\dot{\mathbf{e}} - F(\eta,\dot{\eta},\nu) - \tau_{STA} \tag{5.15}$$

D'où la loi de commande en force (N) τ exprimée dans le repère de l'AUV :

$$\tau = J^T \mathbf{M}_\eta(\eta)\left[\ddot{\eta}_d + \mathbf{T}\dot{\mathbf{e}} - F(\eta,\dot{\eta},\nu) - \tau_{STA}\right] \tag{5.16}$$

Remarquons que les performances de ce contrôleur STA sont donc liées à la précision de la modélisation de l'AUV. Au vu des approximations faites dans nos équations, c'est un point d'amélioration futur important de nos algorithmes. L'identification des paramètres hydrodynamiques par CFD ou par expérimentations sera donc nécessaire pour obtenir des résultats optimaux.

5.2.4 Conversion en commandes moteurs

Pour déterminer la poussée que doivent produire les moteurs, on utilise la matrice d'allocation des moteurs $\mathbf{A}$ (2.30) :

$$\tau = \mathbf{A}\mathbf{u}_{\text{mot}} \tag{5.17}$$

$$\mathbf{u}_{\text{mot}} = \mathbf{A}^{\dagger}\tau = \mathbf{A}^{T}\left(\mathbf{A}\mathbf{A}^{T}\right)^{-1}\tau \tag{5.18}$$

Enfin, la consigne de poussée individuelle $u_{\text{mot},i}$ est convertie en signal de commande moteur grâce aux tables de correspondance expérimentales fournies par BlueRobotics pour les propulseurs T200. Une interpolation linéaire associe chaque force en Newton à sa consigne PWM (puis au pourcentage de puissance correspondant) envoyée directement aux ESC.

Les résultats du contrôle sont présentés dans le chapitre suivant.

Chapitre 6

Expérimentations et résultats

Ce chapitre a pour objectif de présenter les résultats de nos méthodes de guidage et de contrôle. Des tests ont été réalisés en simulation et en bassin. Dans certains cas, une comparaison avec un régulateur PID sera réalisée.

Pour étudier la robustesse de nos algorithmes face aux perturbations en simulation, deux conditions de tests seront étudiées :

- **Sans perturbation :** les équations de l'AUV seront parfaitement connues du contrôleur et le courant sera nul.
- **Avec perturbation :** certains paramètres de l'AUV seront changés dans la simulation et un courant sera appliqué.

Les paramètres de chaque cas sont détaillés en annexe E.

6.1 Contrôle d'état

Ce contrôle simple a pour but de tester le contrôleur STA sans guidage. L'entrée du contrôleur est uniquement une pose désirée.

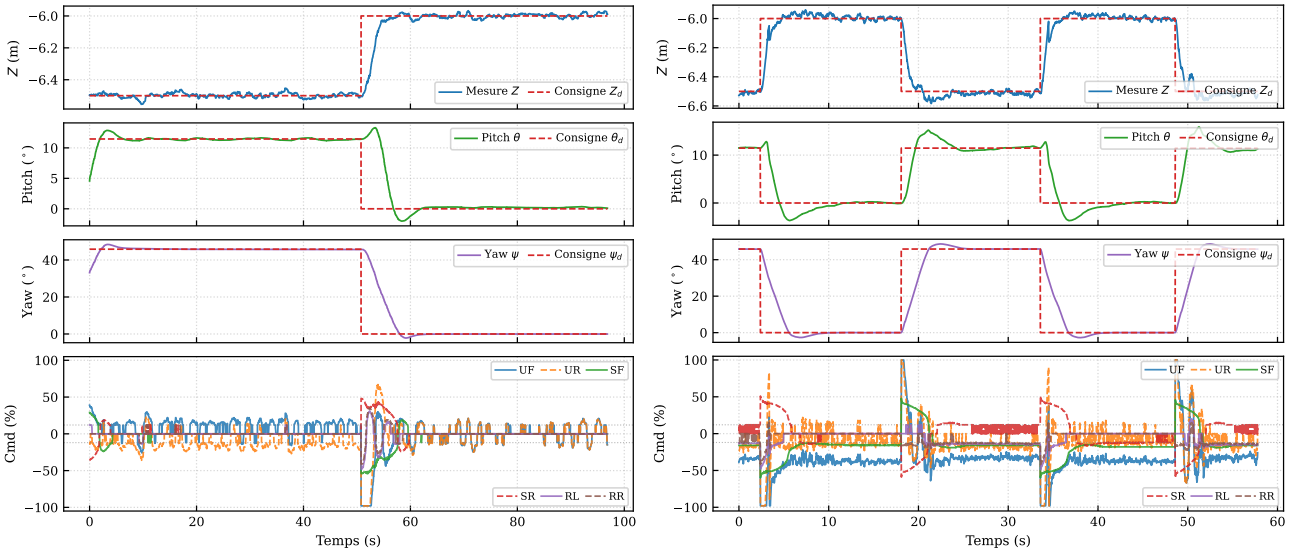
Ces tests permettent de facilement observer :

- Les oscillations ;
- Les erreurs statiques ;
- Le temps de convergence en fonction des paramètres du contrôleur.

6.1.1 En simulation

Tests sans perturbation Comme illustré sur la Figure 6.1a (issue des simulations globales de la Figure 6.1), l'AUV parvient à suivre la commande et l'on obtient une erreur satisfaisante sur les 3 axes contrôlés. Après 60 s, on observe de légères oscillations dans les commandes moteurs qui pourraient correspondre à un léger *chattering*, mais celui-ci reste négligeable.

Tests avec perturbation Au vu de la Figure 6.1b, on ne remarque presque aucune différence dans le suivi de consigne par rapport au test sans perturbation. Le tangage met tout de même plus de temps à converger.



(a) Sans perturbation (b) Avec perturbation

FIGURE 6.1 – Simulations du contrôleur STA.

6.1.2 En bassin

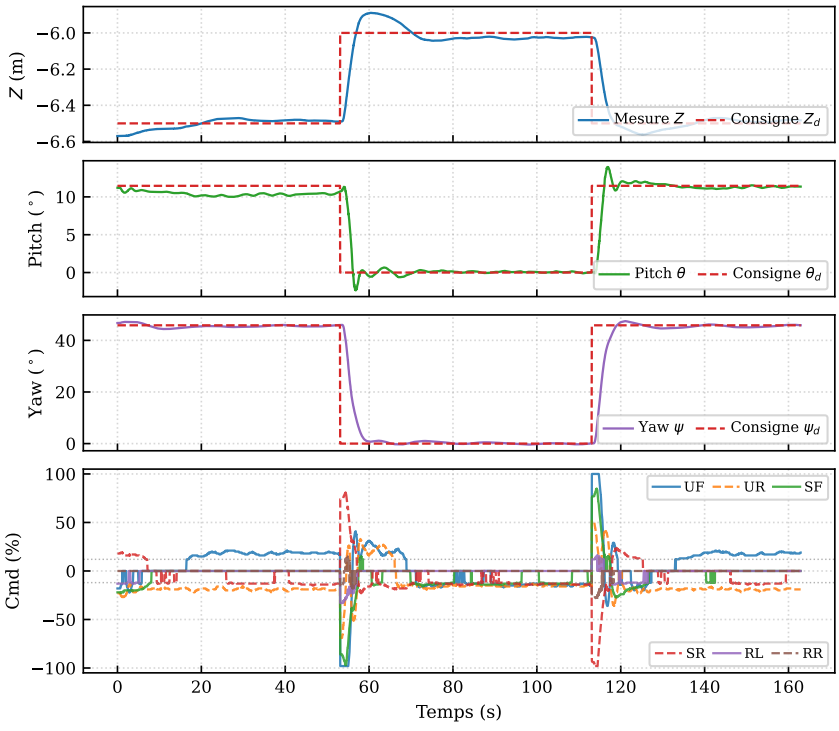


FIGURE 6.2 – Résultats expérimentaux du contrôleur STA en bassin.

Avec le vrai robot et en bassin (Figure 6.2), on obtient des résultats inférieurs à la simulation, mais qui restent satisfaisants.

On n'observe aucun *chattering*, mais on remarque tout de même des oscillations sur le lacet. Ces dernières peuvent être dues :

— À l'absence de gestion de la zone morte des moteurs ;
— À un mauvais choix de δ (5.5).

Un paramétrage plus poussé devrait donc permettre d'améliorer ces résultats.

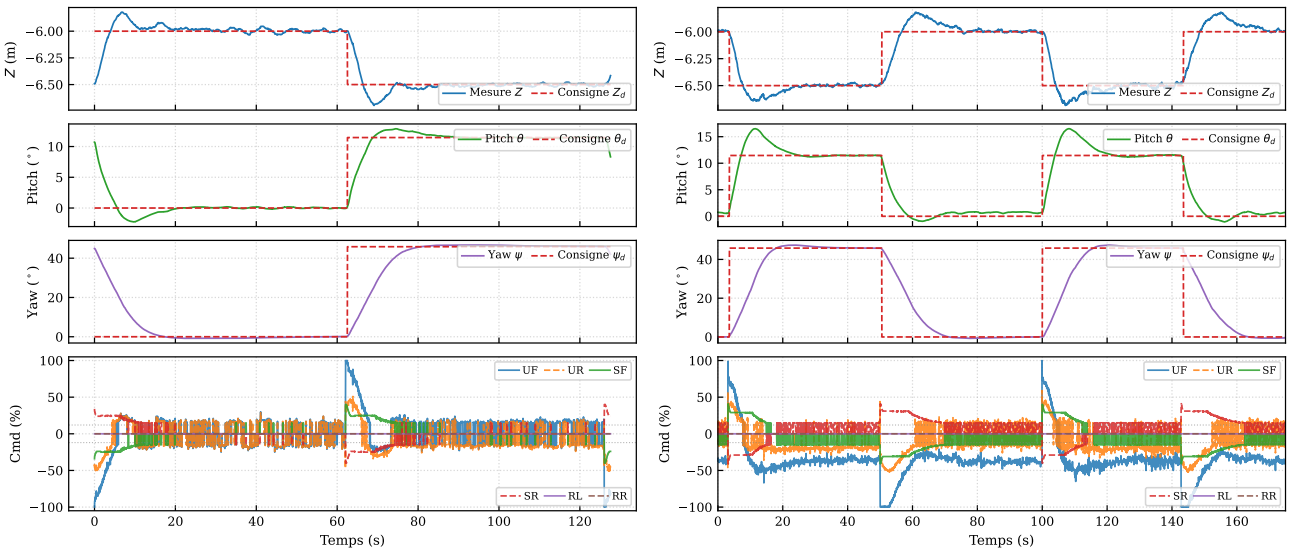
6.1.3 Comparaison avec le PID

Un PID en cascade (PID sur la position qui commande un PID sur la vitesse) a été implémenté pour le contrôle de 3 degrés de liberté (cap, tangage, profondeur). Par conséquent, la comparaison avec le STA ne se fera que sur ces degrés de liberté.

Une gestion simple de la zone morte des moteurs a été implémentée : si une commande est inférieure à la zone morte, on ajoute la zone morte à la commande.

Le but de cette comparaison est de valider notre contrôleur STA et de comparer ses performances et sa robustesse face aux perturbations avec celles d'un contrôleur populaire.

En simulation



(a) Sans perturbation (b) Avec perturbation

FIGURE 6.3 – Simulations du contrôleur PID.

Tests sans perturbations Comme attendu sur la Figure 6.3a, le PID suit presque parfaitement la consigne. Le temps de réponse peut probablement être amélioré avec un paramétrage plus poussé.

Tests avec perturbations D'après la Figure 6.3b, le contrôle reste performant malgré les perturbations. On observe tout de même des dépassements bien plus importants ainsi qu'une erreur statique sur le tangage.

En bassin

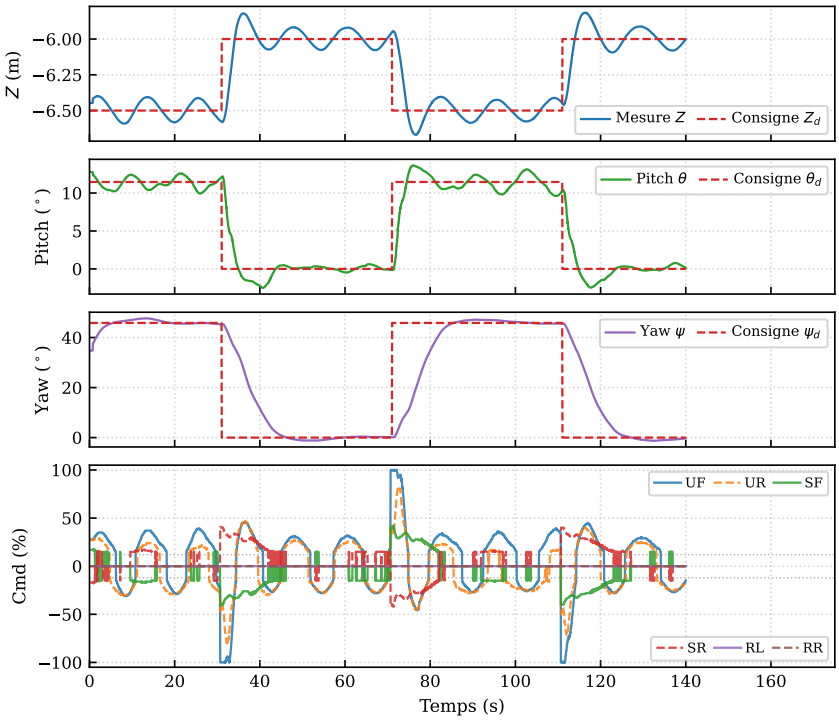


FIGURE 6.4 – Résultats expérimentaux du contrôleur PID en bassin.

Comme le montre la Figure 6.4, le contrôle est ici bien moins précis qu'en simulation. Les paramètres optimaux pour le contrôle en profondeur n'ont pas pu être identifiés rapidement, ce qui explique ces oscillations qui se répercutent sur le tangage. Le but de cette comparaison étant d'évaluer des contrôleurs avec un paramétrage rapide, nous n'avons pas consacré davantage de temps à ce problème. Cependant, cela démontre qu'un PID n'est pas toujours simple à paramétrer.

Aucun test avec perturbation n'a été réalisé en bassin, faute de temps. On peut tout de même considérer l'impact du *tether* (câble) comme une perturbation non constante.

Analyse de l'efficacité énergétique et de l'effort de commande

Afin de comparer quantitativement la sollicitation des actionneurs et la fluidité des commandes entre le STA et le PID, trois indicateurs ont été calculés sur l'ensemble des essais :

— L'effort moyen absolu (L_1) ;
— L'énergie cumulée du signal (L_2) ;
— La variabilité moyenne des commandes (TV).

Ces métriques ne sont que des indicateurs mais permettent d'évaluer la demande énergétique de chaque contrôleur en l'absence de mesures réelles. Le Tableau 6.1 résume l'ensemble de ces métriques.

On constate dans le Tableau 6.1 que dans la majorité des cas, le STA nécessite une énergie de commande cumulée (L_2) nettement inférieure à celle du PID.

TABLE 6.1 – Synthèse des métriques d'effort et d'énergie de commande.

Configuration	**Effort** L_1 **(%)**		**Effort** L_2 **(%²)**		**Variabilité (TV) (%/pas)**	
	STA	PID	STA	PID	STA	PID
Simu. sans pert.	36,00	46,72	1145,88	1270,48	1,15	2,64
Simu. avec pert.	106,79	87,89	3841,90	3004,70	1,34	2,20
Essais bassin	56,41	75,89	1451,53	2509,55	0,31	0,94

De plus, la variabilité TV du STA reste toujours inférieure à celle du PID, ce qui confirme l'absence de sollicitation inutile des moteurs et le faible *chattering*.

Ces valeurs plus élevées pour le PID peuvent s'expliquer par la gestion de la zone morte et par des gains mal paramétrés. Ces résultats n'indiquent donc pas que le STA est un contrôleur particulièrement économe en énergie, mais montrent au moins que sa consommation énergétique est comparable à celle d'un contrôleur classique.

Conclusion

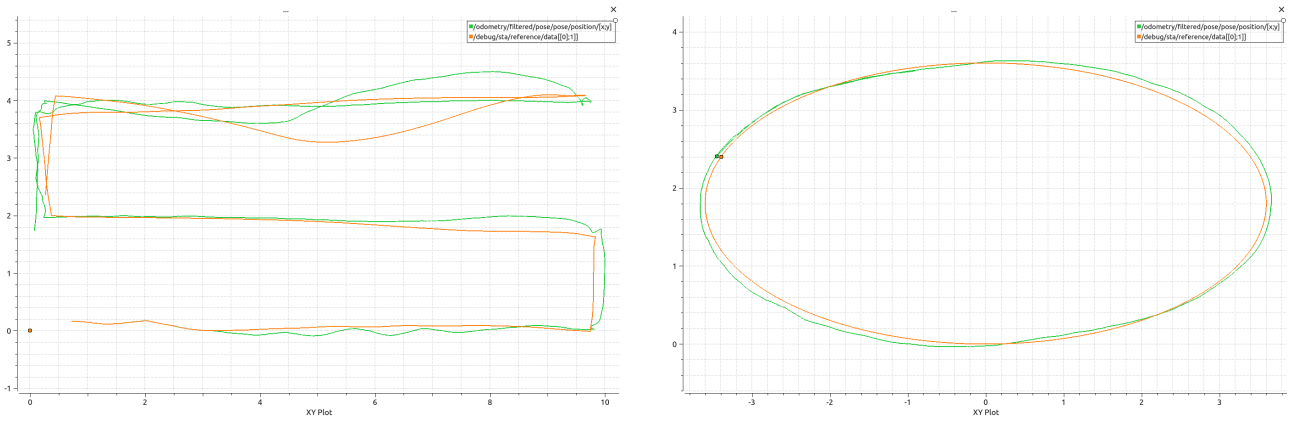
- **Précision du suivi de commande** : Le STA et le PID obtiennent tous deux des résultats satisfaisants. Le PID devrait même être plus précis dans des environnements non perturbés, car il ne souffre pas d'un possible *chattering*.
- **Robustesse** : Le STA est plus résistant à de faibles perturbations que le PID, même si ce dernier obtient de très bons résultats. Des tests plus poussés doivent être effectués pour clarifier ce point.
- **Facilité de paramétrage** : Ce critère est un ressenti purement personnel. Malgré un nombre supérieur de paramètres, le STA s'est révélé plus simple à régler que le PID. On obtient très rapidement de bons résultats avec le STA ; un paramétrage poussé est surtout demandé pour obtenir une meilleure précision.
- **Efficacité énergétique** : Bien que la métrique de la moyenne des commandes ne soit pas représentative de l'énergie réellement consommée par le robot (Tableau 6.1), elle nous permet au moins d'avancer qu'en moyenne le STA ne demande pas plus d'effort aux moteurs que le PID lors de nos tests.

Remarques : Il est important de noter que ces tests n'avaient pas pour vocation de mener une comparaison poussée entre le STA et le PID. Ces essais nous ont surtout permis de valider l'efficacité du STA et de montrer que, dans certaines conditions, il peut être plus performant qu'un PID. Cependant, un PID bien paramétré fonctionnera sûrement dans la majeure partie des cas d'utilisation d'un AUV résident.

6.2 Suivi de chemin et de trajectoire

6.2.1 En bassin

Lors des tests en bassin, on observe un suivi de chemin et de trajectoire satisfaisant. Ces tests nous ont aussi permis de valider l'algorithme de *pure pursuit*, comme illustré sur la Figure 6.5.



(a) Suivi de chemin (*pure pursuit*) (b) Suivi de trajectoire

FIGURE 6.5 – Vue du dessus des essais de suivi de chemin et de trajectoire en bassin (consigne en orange, position estimée en vert, les 2 axes sont en m)

Les oscillations sur la trajectoire de l'AUV pour le suivi de chemin (Figure 6.5a) sont principalement dues à un mauvais paramétrage du STA sur le cap. Ces erreurs ont été corrigées, mais nous avons malheureusement manqué de temps pour réaliser de nouveaux tests.

6.2.2 En pleine mer

Fin juin, nous avons réalisé une journée de tests sur le site d'essais en mer de Sainte-Anne-du-Portzic (Figure 6.6).

Cette journée nous a permis de :

— Valider le déploiement de l'AUV en pleine mer ;
— Valider les algorithmes de fusion de capteurs ;
— Tester le contrôleur STA (qui était en début de phase de test à ce moment).

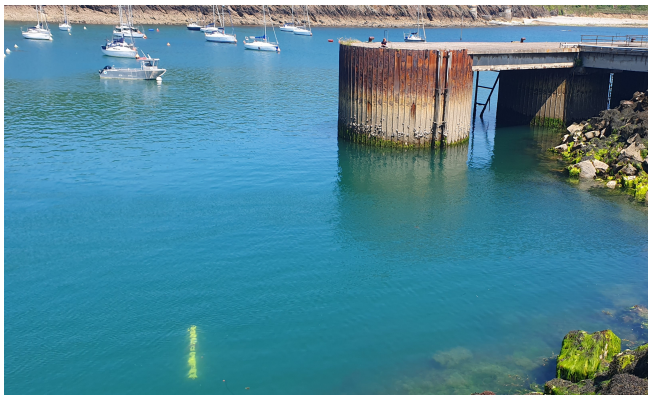


FIGURE 6.6 – Déploiement de l'AUV X-300 sur le site d'essais de Sainte-Anne-du-Portzic.

Avec seulement quelques minutes de paramétrage, nous avons réussi à obtenir un suivi de trajectoire correct avec le STA (Figure 6.7).

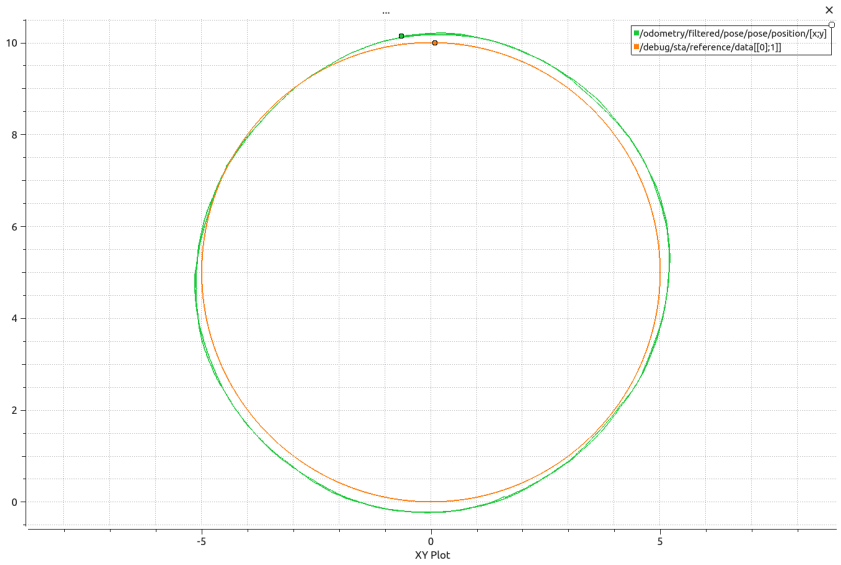


FIGURE 6.7 – Suivi de trajectoire en pleine mer (consigne en orange, position estimée en vert, les 2 axes sont en m).

6.3 Amarrage et désamarrage

6.3.1 En simulation

La simulation (Figure 6.8) nous a permis de valider nos algorithmes de guidage et de contrôle pour ces phases critiques de la résidence.

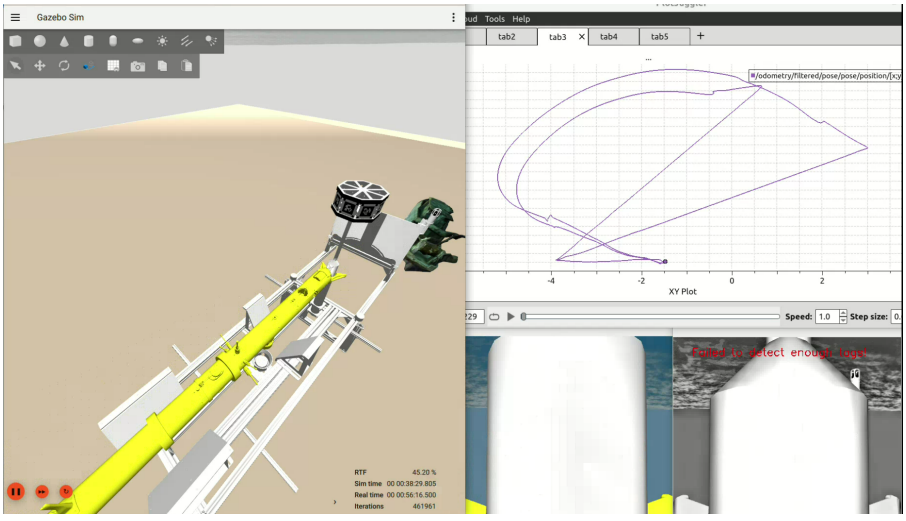


FIGURE 6.8 – Opérations d'amarrage et désamarrage en simulation, ici en phase de descente.

Les résultats sont concluants : l'AUV est capable de s'amarrer, même en présence de courant et après une perte temporaire des mires de la base, de se désamarrer, de rejoindre un point différent de son point de départ et de répéter l'opération.

Lors de la phase de tests la plus longue (avec des incertitudes sur le modèle de l'AUV et un courant constant), **l'AUV X-300 simulé a effectué près de 120 amarrages sans échec**. En simulation, un amarrage prend environ deux minutes si l'AUV démarre à environ 3 m de la station d'amarrage.

6.3.2 En bassin

Nous avons réalisé des tests d'amarrage en bassin. Une partie des amarrages a réussi, mais le contrôle était trop peu précis et le positionnement par vision renvoyait occasionnellement des valeurs aberrantes.

Il a donc été décidé de revoir et corriger ces points, ce qui a été fait. Malheureusement, le robot a subi de nombreux problèmes matériels juste avant nos nouveaux tests (*tether* endommagé, DVL défectueux, court-circuit sur la carte SD contenant les codes propriétaires bas niveau de l'AUV), ce qui nous a empêchés de les réaliser dans le temps imparti au stage.

En raison de ces incidents matériels et à défaut d'enregistrements exploitables lors des premiers essais concluants, aucun jeu de données représentatif ne peut être présenté ici. Néanmoins, au vu des corrections apportées, la validation expérimentale complète en bassin est estimée à moins d'une semaine de travail une fois le robot opérationnel.

Chapitre 7

Conclusion et perspectives

Lors de ce stage, j'ai implémenté l'ensemble de la chaîne algorithmique nécessaire au suivi de chemin et à l'amarrage autonome d'un AUV. Mon travail a couvert l'intégralité de quatre domaines essentiels à la robotique mobile autonome : l'intégration matérielle et logicielle des capteurs au sein du système ROS du robot, la fusion de données pour l'estimation d'état, ainsi que les couches de guidage et de contrôle-commande. J'ai également créé une simulation qui pourra servir au développement des futurs algorithmes.

Même si la validation expérimentale de l'amarrage en bassin a été freinée par des aléas matériels, illustrant la complexité du passage de la simulation au monde réel, l'efficacité des algorithmes a été démontrée. La robustesse de la simulation et la validation des essais en bassin pour le contrôle d'état et le suivi de trajectoire constituent des preuves de concept solides pour la suite du projet.

Toutefois, la validation en bassin et en mer de mes algorithmes ne représente qu'une étape vers l'autonomie résidentielle complète. D'un point de vue purement algorithmique et logiciel, plusieurs étapes critiques restent à franchir :

- Modélisation plus précise de l'AUV (CFD, essais en bassin) ;
- Validation des algorithmes en mer plus agitée ;
- Passage des algorithmes sur la Jetson embarquée ;
- Tests sans le *tether*, d'abord en bassin, puis en mer ;
- Caractérisation des cas de défaillance et solutions à développer puis tester pour chacun d'entre eux ;
- Phases de tests longues durées et progressives pour fiabiliser le système, éliminer les cas limites et minimiser les risques opérationnels avant un test de résidence en conditions réelles.

Sur le plan personnel, ce stage s'est révélé particulièrement enrichissant. Il m'a permis de mettre en pratique des notions théoriques complexes de théorie du contrôle, de guidage et de filtrage, tout en me confrontant aux contraintes concrètes de la robotique sous-marine. Faire face aux imprévus matériels et à la réalité du milieu marin m'a appris à adapter ma démarche méthodologique, à faire preuve de rigueur dans le diagnostic de pannes et à gagner en autonomie.

Je suis particulièrement reconnaissant envers l'équipe de m'avoir accordé sa confiance sur une problématique aussi exigeante. Ce stage a confirmé mon intérêt pour la robotique autonome, en particulier la robotique marine et sous-marine.

Annexe A

Chronologie des essais expérimentaux

TABLE A.1 – Chronologie des essais expérimentaux et campagnes de tests de l'AUV.

Date	**Objectifs et activités expérimentales**
27/03	Qualification du système de positionnement USBL hors station d'amarrage et tests PID.
03/04	Validation du positionnement USBL sur station d'amarrage et caractérisation du DVL.
22/05	Trajectoires manuelles pour l'évaluation de l'EKF.
28/05	Caractérisation des repères capteurs, évaluation de la centrale inertielle (AHRS SBG) et tests des algorithmes de fusion de données.
04/06	Évaluation préliminaire du contrôleur STA et acquisition de données pour la fusion capteurs.
05/06	Phase de débogage et d'expertise sur la centrale inertielle.
08/06	Résolution de dysfonctionnements sur l'AHRS et ajustement du filtre EKF.
12/06	Optimisation et débogage du filtre EKF.
18/06	**Campagne en mer (Sainte-Anne-du-Portzic) :** validation des capteurs réels et de la fusion, acquisition de jeux de données (amarrage et trajectoires), exécution de trajectoires élémentaires en mode autonome et évaluation de la loi de commande.
09/07	Réglage des gains de la loi de commande.
15/07 – 17/07	Diagnostic et résolution de problèmes d'intégration sur le DVL.
21/07 – 22/07	Affinement des performances du système de contrôle et débogage.
23/07 – 27/07	Campagne d'essais d'amarrage et suivi de chemin.
28/07 – 06/08	Diagnostic et prise en charge d'une défaillance du *tether*.
07/08 – 10/08	Optimisation finale des lois de commande et étude comparative STA / PID.
10/08	Incident matériel : court-circuit critique entraînant la destruction de la carte mémoire du calculateur bas niveau.

Annexe B

Diagramme de Gantt

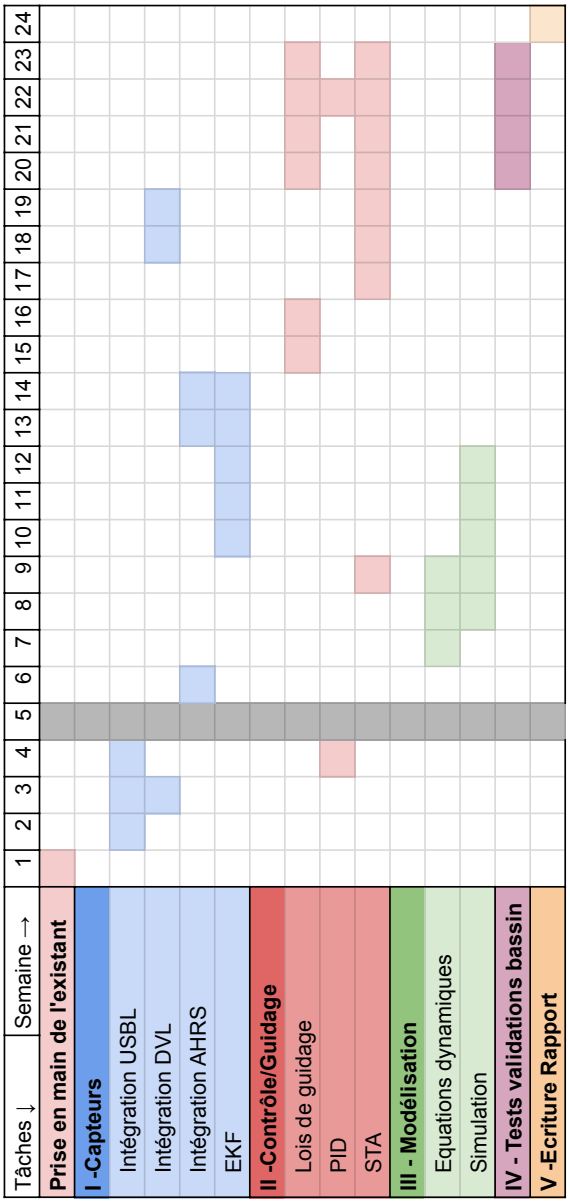


FIGURE B.1 – Diagramme de Gantt prévisionnel et déroulement du projet.

Annexe C

Plan de l'AUV et positions des capteurs et actionneurs

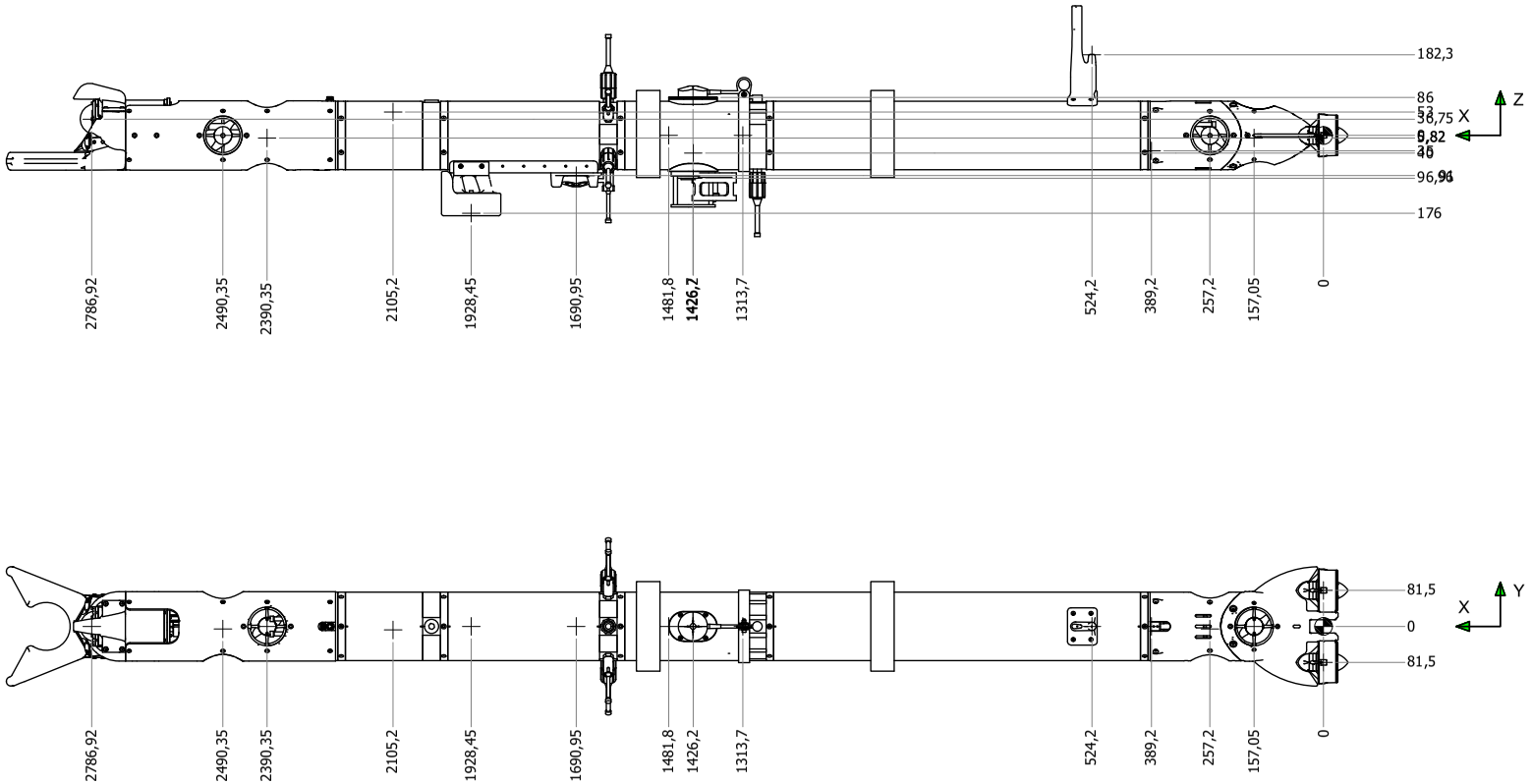


Table			
Désignation	X	Y	Z
Origine	0	0	0
Centre de poussée	1481.8	0	0
Centre de gravité estimé	1427	0	-40
Thruster long Bbd	0	-81.5	0
Thruster long Tbd	0	81.5	0
Thruster trans AR	257.2	-5.82	0
Thruster trans AV	2490.2	-5.82	0
Thruster Vert AR	157.05	0	-5.82
Thruster Vert AV	2390.2	0	-5.82
Pression	389.2	0	-35
GPS	524.2	0	182.3
IMU SBG	1481.8	0	0
IMU PNI	2105.2	-8	53
USBL	1426.2	0	-96.96
DVL 2	1426.7	0	86
DVL 95	1690.95	0	-91
Inductif	1928.45	0	-176
Caméra	2787	0	36.75

Rév.	Description	Date	Dessiné par	Vérifié par	Approuvé par
A					

Matière : | Tol. générales :

Traitement : | Rugosité:

Masse: 39,720 kg | eotp :

Ensemble :

Sous-ensemble :

Titre: H41BEAIA00 - Vecteur intermédiaire coordonnées

RDT/SIM | Planche: 1 /1 | Format: A2 | Echelle: 1 : 8 | Indice: A

Ifremer
CENTRE DE BRETAGNE
Technopôle de Brest Iroise
CS 10070
29280 PLOUZANE
Tel : 02 98 22 40 40

FIGURE C.1 – Plan de l'AUV et positions des capteurs et actionneurs

Annexe D

Paramètres numériques du modèle de l'AUV X-300

1. Constantes physiques et éléments de calcul

— Accélération de la pesanteur : $g = 9{,}81\ \mathrm{m/s^2}$
— Masse volumique de l'eau : $\rho = 1000\ \mathrm{kg/m^3}$

2. Géométrie et masse du robot

— Longueur du tube : $L_{\text{tube}} = 2{,}8\ \mathrm{m}$
— Diamètre du tube : $D_{\text{tube}} = 0{,}155\ \mathrm{m} \quad \Longrightarrow$ Rayon $R = 0{,}0775\ \mathrm{m}$
— Masse totale : $m = 40{,}05\ \mathrm{kg}$
— Volume total : $V = 0{,}040\ \mathrm{m^3}$
— Centre de gravité : $\mathbf{r}_g = [0{,}0,\ 0{,}0,\ -0{,}04]^T\ \mathrm{m}$
— Centre de flottabilité : $\mathbf{r}_b = [0{,}0,\ 0{,}0,\ 0{,}0]^T\ \mathrm{m}$

3. Inertie du corps rigide ($\mathbf{M}_{rb}$)

$$I_x = \frac{1}{2} m \left(\frac{D_{\text{tube}}}{2} \right)^2 \approx 0{,}120\ \mathrm{kg \cdot m^2}$$

$$I_y = I_z = \frac{m}{4} \left[\left(\frac{D_{\text{tube}}}{2} \right)^2 + \frac{L_{\text{tube}}^2}{3} \right] \approx 26{,}233\ \mathrm{kg \cdot m^2}$$

$$\mathbf{I}_o = \mathrm{diag}(I_x,\ I_y,\ I_z)$$

4. Hydrodynamique et traînée

— Coefficient de traînée du cylindre (corps) : $C_{d,\text{cyl}} = 1{,}3$

— Coefficient de traînée de l'ellipse (tête) : $C_{d,\text{ell}} = 0{,}6$
— Matrice d'amortissement linéaire ($\mathbf{D}_{\text{lin}}$) :

$$\mathbf{D}_{\text{lin}} = \text{diag}(-20,\ -150,\ -100,\ -3,\ -120,\ -100)$$

5. Matrice d'allocation des propulseurs (A)

Basée sur les bras de levier $d_1 = 0{,}0815$ m, $d_2 = 1{,}26995$ m et $d_3 = 0{,}9632$ m :

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1{,}26995 & 1{,}26995 & 0 & 0 \\ 0{,}0815 & -0{,}0815 & 0 & 0 & -0{,}9632 & 0{,}9632 \end{bmatrix}$$

Annexe E

Paramètres cas nominal vs cas perturbé

TABLE E.1 – Comparaison des paramètres du modèle de l'AUV entre le cas nominal (non perturbé) et le cas perturbé.

Paramètre	**Symbole / Unité**	**Cas nominal**	**Cas perturbé**
Propriétés massiques et géométriques			
Masse du robot	m [kg]	40,00	41,00
Inertie suivant x (roulis)	I_{xx} [kg · m^2]	0,12	0,12
Inertie suivant y (tangage)	I_{yy} [kg · m^2]	26,20	26,20
Inertie suivant z (lacet)	I_{zz} [kg · m^2]	26,20	26,20
Longueur du corps	L [m]	2,80	3,00
Rayon du corps	R [m]	0,07525	0,10000
Volume immergé	V [m^3]	0,04	0,04
Masse volumique du fluide	ρ [kg/m^3]	1000	1000
Centre de gravité	$\mathbf{r}_g$ [m]	$[0{,}0\,;\,0{,}0\,;\,-0{,}04]^T$	$[0{,}02\,;\,0{,}01\,;\,-0{,}04]^T$
Centre de poussée	$\mathbf{r}_b$ [m]	$[0{,}0\,;\,0{,}0\,;\,0{,}0]^T$	$[0{,}0\,;\,0{,}0\,;\,0{,}0]^T$
Coefficients d'amortissement linéaire ($\mathbf{D_{lin}}$)			
Avance (*Surge*)	D_u [N · s/m]	−20,0	−50,0
Dérive (*Sway*)	D_v [N · s/m]	−150,0	−300,0
Pilonnement (*Heave*)	D_w [N · s/m]	−100,0	−200,0
Roulis (*Roll*)	D_p [N · s/rad]	−3,0	−6,0
Tangage (*Pitch*)	D_q [N · s/rad]	−120,0	−240,0
Lacet (*Yaw*)	D_r [N · s/rad]	−100,0	−200,0
Perturbations environnementales			
Courant marin	$\mathbf{v}_c$ [m/s]	$[0{,}0\,;\,0{,}0\,;\,0{,}0]^T$	$[0{,}2\,;\,0{,}1\,;\,0{,}0]^T$

Bibliographie

[1] Jérôme Blandin, Adrien Chauvet, Maxime Ferrera, Morgann Bot, Damien Vourc'h, Julien Legrand, and Nicolas Mertz. Development of a resident auv for scientific observation. pages 1–7, 06 2025. doi : 10.1109/OCEANS58557.2025.11104752.

[2] Jixin Liu, Fei Yu, Bo He, and C. Guedes Soares. A review of underwater docking and charging technology for autonomous vehicles. *Ocean Engineering*, 297 :117154, 2024. ISSN 0029-8018. doi : https://doi.org/10.1016/j.oceaneng.2024.117154. URL `https://www.sciencedirect.com/science/article/pii/S0029801824004918`.

[3] Lionel Lapierre. Développement d'un simulateur pour la conception de la commande d'un auv. Technical report, ENSTA, 2025. URL `https://webperso.ensta.fr/lapierre/Teaching/Marine_Robots/PDFs/3-AUV%20Simulator.pdf`.

[4] Raffaele Borgognoni. Parameter identification of the fossen model applied to the uuv blucy. Technical report, Università di Bologna, 2023. URL `https://amslaurea.unibo.it/id/eprint/28011/1/Thesis_Borgognoni_.pdf`.

[5] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley & Sons, 1994. ISBN 0 471 94113 1.

[6] Horace Lamb. *Hydrodynamics*. Dover Publications, 6th edition, 1945.

[7] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 335–348, Cham, 2016. Springer International Publishing. ISBN 978-3-319-08338-4.

[8] Auwal Shehu Tijjani, Ahmed Chemori, and Vincent Creuze. A survey on tracking control of unmanned underwater vehicles : Experiments-based approach. *Annual Reviews in Control*, 54 :125–147, 2022. ISSN 1367-5788. doi : https://doi.org/10.1016/j.arcontrol.2022.07.001. URL `https://www.sciencedirect.com/science/article/pii/S1367578822000864`.

[9] Dong Ma, Ye Li, Teng Ma, and António M. Pascoal. The state of the art in key technologies for autonomous underwater vehicles : a review. *Engineering*, 2025. ISSN 2095-8099. doi : https://doi.org/10.1016/j.eng.2025.08.002. URL `https://www.sciencedirect.com/science/article/pii/S209580992500445X`.

[10] Arie Levant. Sliding order and sliding accuracy in sliding mode control. *International Journal of Control*, 58 :1247–1263, 12 1993. doi : 10.1080/00207179308923053.

[11] Emmanuel Gamero, Ahmed Chemori, Jorge Torres, Jesús Guerrero, and Vincent Creuze. Robust adaptive sta-based tracking control of auvs with real-time experiments. *IFAC-PapersOnLine*, 59(18) :313–318, 2025. ISSN 2405-8963. doi : https://doi.org/10.1016/j.ifacol.2025.10.239. URL `https://www.sciencedirect.com/science/article/pii/S2405896325016386`. 14th IFAC Symposium on Robotics ROBOTICS 2025.

[12] Michael Alibani, Carmelo Ferrara, and Lorenzo Pollini. Super twisting sliding mode control for precise control of intervention autonomous underwater vehicles. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–7, 2018. doi : 10.1109/OCEANS.2018.8604839.