SYS802 : Méthodes avancées de commande

Conception d'une loi de commande linéaire et non linéaire pour un sous-marin autonome

Présenté à :

David Bensoussan

Par :

LAMARRE, Alexandre

RAYEH, Lilia

SCHENA, Guilhem

Montréal, le 6 décembre 2021

Remerciements

Nous souhaitons premièrement remercier tous les membres du regroupement scientifique S.O.N.I.A, car sans leur contribution, le sous-marin ne serait pas au même niveau qu'il est aujourd'hui. De plus, leurs contributions ont été essentielles à la réalisation de notre projet, car, plusieurs ont travaillé sur l'intégration mécanique, électrique et logicielle des différents capteurs et actionneurs ainsi que la détermination des constantes mécaniques du modèle. Cela nous a permis de tester nos lois de commande sur le sous-marin réel et de rendre ce projet possible.

Nous souhaitons également remercier notre professeur, M. David Bensoussan pour toute l'aide qu'il nous a apportée au cours de la session. Il nous a grandement aidés à la compréhension de la matière. Sa disponibilité exceptionnelle a eu une grande valeur pour notre projet ainsi que pour le développement du club S.O.N.I.A.

Table des matières

Liste des figures

1. Modélisation du sous-marin

1.1. Présentation du sous-marin

Le dispositif étudié au cours de ce projet de session est un **sous-marin autonome**. Il a été développé par le club technique de l'ETS S.O.N.I.A (Système d'Opération Nautique Intelligent et Autonome). Ce type d'appareil est autrement appelé un AUV (Autonomous Underwater Vehicule) et a pour caractéristique d'être un dispositif aquatique voué à se déplacer sous l'eau de manière indépendante et sans interaction humaine. Cette définition implique donc la mise en place d'une loi de contrôle robuste, sachant qu'une fois submergée le sous-marin est virtuellement hors de contrôle des opérateurs.

Figure 1-Sous-marin autonome AUV8 [1]

D'autre part, l'AUV qui nous intéresse ici est développé dans le but de participer à une compétition internationale annuelle appelée RoboSub consistant à mettre à l'épreuve le sous-marin via une série d'actions à lui faire effectuer sans intervention humaine (suivi de trajectoire, alignement visuel, alignement acoustique, etc). De plus, il est à noter qu'il s'agit déjà de la 8^{e} version d'AUV mise au point par le club SONIA en 2020 et que ce dernier sera amené à en produire un nouveau tous les ans dans les prochaines années.

Le dispositif que nous étudions dans ce travail présente donc des particularités de conception et des d'objectifs de fonctionnalités précis qu'il nous faut prendre en compte lors du design de la loi de commande. Les contraintes que cela implique seront exposées plus en détail plus tard. Pour le moment, intéressons-nous à la structure de l'appareil en lui-même.

Le sous-marin est constitué majoritairement d'un caisson étanche d'environ 30'' x 15'' x 12'' d'aluminium 6061 T6 pour un poids total de 31kg. Son architecture a été conçue dans l'optique de

garantir la contrôlabilité du sous-marin en privilégiant les symétries axiales et un placement spécifique des actionneurs. En effet, le but est de minimiser la masse et l'inertie du sous-marin en centrant les composantes lourdes de la structure tout en compactant l'ensemble pour diminuer la masse d'eau déplacée lors mouvement et ainsi réduire la force de stabilisation nécessaire selon l'axe z (profondeur). Enfin, les symétries permettent de réduire le nombre de constantes des matrices d'inertie et d'amortissement que nous introduisons plus bas. Le sous-marin compte en effet 3 plans de symétrie et se déplace à basse vitesse ce qui nous permet de négliger les forces de couplages hydrodynamiques.

1.2. Actionneurs

D'autre part, l'AUV est composé de 8 moteurs BlueRobotics T-200 dont 4 moteurs planaires gérants les déplacements en x, y et yaw (rotation autour de z) et 4 moteurs, dits de profondeurs, responsables des déplacements en z, roll (rotation autour de x) et pitch (rotation autour de y). Remarquons ici que nous contrôlons 6 degrés de liberté avec 8 actionneurs, ce qui nous place face à un problème de « control allocation » puisque nous devons faire un choix d'affectation des efforts lié à nos 6 déplacements désirés. C'est pourquoi nous aurons tendance à nous orienter vers des lois du type commande optimales afin de trouver un équilibre de distribution optimal.

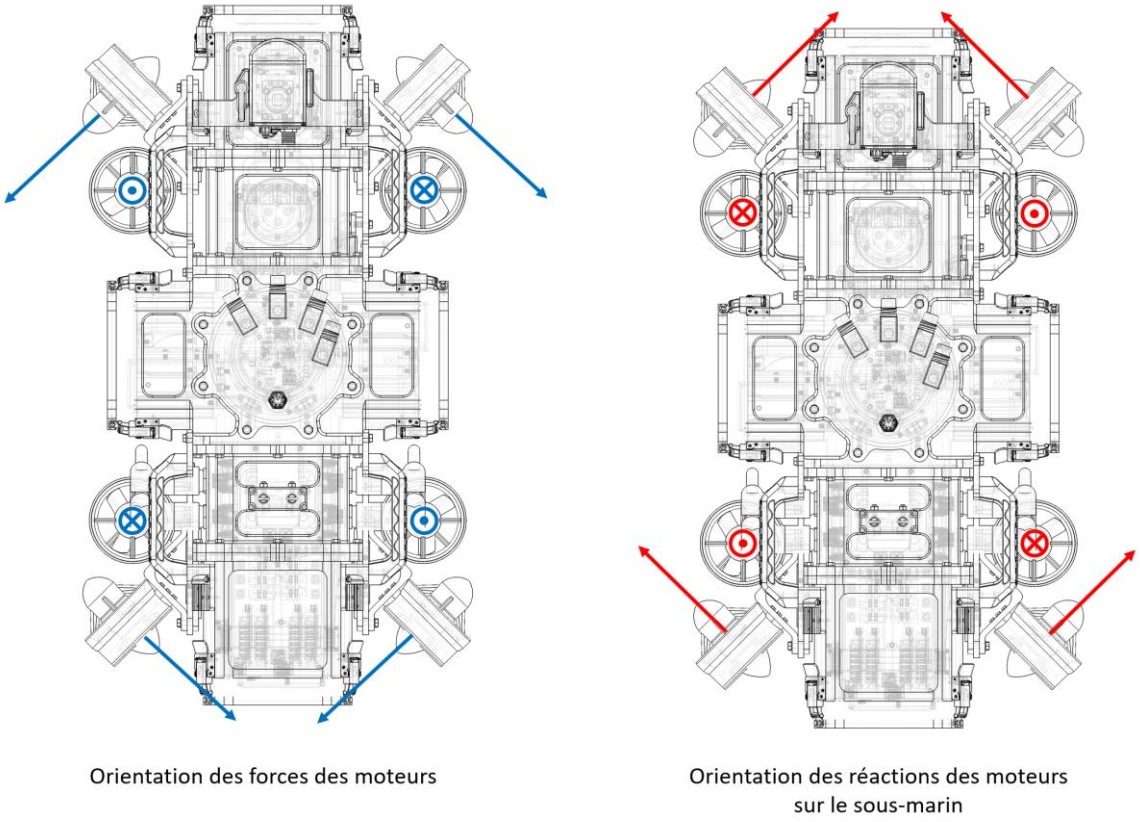

Figure 2-Schéma mécanique du sous-marin AUV8 [2]

1.3. Capteurs

Le sous-marin est de plus composé de 3 capteurs différents permettant la mesure des différents états.

Tout d'abord, nous avons un DVL (Doppler Velocity Log) qui mesure les vitesses linéaires x, y et z par effet doppler. Malheureusement il existe des positions singulières qui rendent le capteur imprécis ou même fautif : lorsque l'angle entre le sous-marin et le fond du bassin dépasse les 20°, l'angle de réflexion des ondes sur les parois du bassin ne permet pas au faisceau de parvenir jusqu'au capteur, qui renvoie alors la valeur maximale par défaut. Certaines surfaces, telle que la céramique, ne garantissent pas non plus une bonne lecture des mesures à cause de problèmes d'absorption et de dispersion. Nous avons donc accès à la mesure de nos états x, y et z, mais celle-ci est soumise à des restrictions de fiabilité dont il nous faut tenir compte dans le design de la commande comme nous le verrons plus tard.

Figure 3 -DVL Teledyne PathFinder[3]

Figure 4-AHRS VectorNAV VN-100 [8]

Nous avons également accès aux mesures d'un AHRS (Attitude and Heading Reference System) qui nous donne une valeur d'angles en quaternions (à quatre composantes contrairement aux angles d'Euler qui en présentent 3), les accélérations selon x, y et z et enfin les vitesses angulaires. Notons que l'AHRS est de plus constitué d'un filtre de Kalman intégré sur sa carte mère qui fusionne les données d'un accéléromètre, d'un gyroscope et d'un magnétomètre.

Enfin, nous avons une seconde mesure de la position en profondeur z grâce à un Depth Sensor. Celui-ci la calcule par différentiel de pression. La présence de ce dernier capteur nous permet de nous affranchir des aléas liés au DVL pour ce qui est du contrôle du déplacent en profondeur qui est particulièrement critique.

Figure 5-Depth Sensor ImpactSubsea ISD4000[5]

Chacun de ces capteurs nous envoie des mesures à une fréquence précise (10Hz pour le DVL et 50Hz pour les deux autres). Cela entraîne une répercussion sur le choix du temps d'échantillonnage Ts de notre représentation d'état discrete.

1.4. Compétition

Comme nous l'avons dit, la compétition RoboSub impose un certain nombre de contraintes sur la structure et l'asservissement du sous-marin. Des contraintes de l'ordre de la sécurité tout d'abord : le sous-marin doit flotter à l'arrêt et comporter un interrupteur physique d'arrêt d'urgence. De plus il ne pas dépasser une certaine taille et un certain poids fixé par le comité des jurés. D'autre part, le cadre de la compétition nous permet de limiter les fonctionnalités désirées pour le sous-marin : il est conçu pour naviguer dans un bassin d'eau douce avec de faibles perturbations (pas de vague, faible houle).

Les activités de la compétition sont de trois types : les tâches de positionnement visuel, les tâches de positionnement acoustique et enfin les tâches de trajectoire.

Les tâches visuelles consistent à détecter, à l'aide de caméras embarquées et d'algorithmes de machine learning, des repères visuels donnant des informations de positionnement et d'alignement que le sous-marin doit exécuter.

Les tâches acoustiques sont basées sur la présence de deux émetteurs disposés à des emplacements inconnus dans le bassin de test. Le sous-marin doit pouvoir les repérer puis atteindre leur position pour effectuer deux tâches : la tâche des torpilles et celle de l'octogone. Le sous-marin possède des hydrophones qui sont en mesure de capter les ondes acoustiques et de calculer l'angle d'élévation ainsi que l'angle de direction. Pour pouvoir capter les signaux des deux émetteurs, le sous-marin doit se déplacer rapidement dans le bassin sans accumuler trop d'erreurs. Pour la tâche des torpilles, le sous-marin doit seulement aller vers la direction de l'émetteur acoustique jusqu'à ce que la caméra détecte la présence de l'obstacle sur lequel il faut lancer les torpilles. La tâche de l'octogone est beaucoup plus exigeante, car elle nécessite de s'aligner au-dessus de l'émetteur et de faire surface dans un espace délimité par un octogone. Cette tâche est particulièrement importante, car si l'AUV fait surface en dehors de l'octogone, l'essai s'achève.

Enfin, les tâches de trajectoires sont nombreuses. La première consiste à passer à travers un portique flottant. Au cours de ce déplacement, il est possible d'obtenir des points supplémentaires de style si le sous-marin réussit à effectuer des figures (tonneau, chandelle, etc). Les tâches de trajectoires comprennent également le déplacement du sous-marin d'une tâche à l'autre. Pour effectuer ce genre de mouvements, nous avons bien entendu besoin d'un contrôleur adéquat.

1.5. Référentiels

Les mesures des trois capteurs nous sont données dans deux référentiels différents. Le premier est le référentiel fixe galiléen dans lequel est exprimée la position z. En revanche, les vitesses sont exprimées dans le référentiel du sous-marin. Il nous faut donc les transformer pour pouvoir les exploiter dans le référentiel galiléen. Pour ce qui est des conventions utilisées, pour le référentiel galiléen, nous

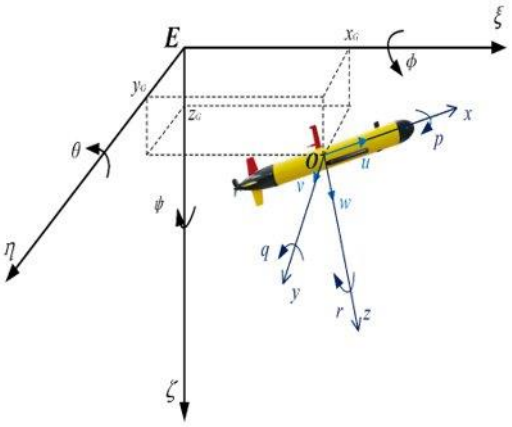

Figure 7- Référentiel du sous-marin [6]

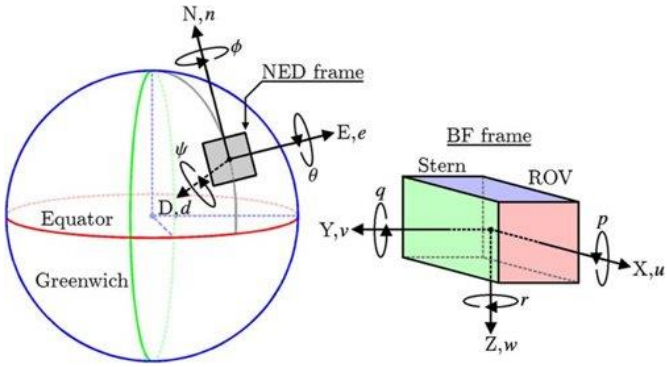

Figure 6-Référentiel inertiel [7]

nous basons sur le « NED » pour « North East Down » de sorte que les axes sont les suivants : X+ pour le nord, Y+ pour l'est et Z+ pour le bas. Pour le référentiel objet, l'origine est placée au centre de masse et les axes sont X : axe longitudinal, Y : axe transversal et Z : axe normal.

1.6. Équations d'état

L'AUV se déplace dans l'eau selon 3 déplacements linéaires en x, y et z et 3 déplacements angulaires en rotation autour de ces mêmes axes. Nous incluons de plus nos vitesses à nos états. Cela nous donne une représentation à 12 états en utilisant les angles d'Euler et 13 états en utilisant les quaternions. Le club SONIA a fait le choix récemment d'utiliser des quaternions afin de s'affranchir des positions singulières des angles d'Euler qui entrainent le problème de « wrap around » pour les angles de $\pm\pi$ radians. Or, puisque le quaternion possède 4 paramètres, il ne présente pas de discontinuités et pas de singularités. Cette particularité s'appelle le « Gimbal lock ».

Cela nous amène à la représentation d'état suivante :

$$x_q = [{n_q}^T v^T]^T = [x\ y\ z\ \eta\ \varepsilon_1\ \varepsilon_2\ \varepsilon_3\ u\ v\ w\ p\ q\ r]^T$$

Avec :

$[x\ y\ z]^T$: vecteur des positions linéaires

$[\eta\ \varepsilon_1\ \varepsilon_2\ \varepsilon_3]^T$: vecteur quaternions

$[u\ v\ w]^T$: vecteur des vitesses linéaires

$[p\ q\ r]^T$: vecteur des vitesses angulaires

Notons qu'il s'agit là de la représentation d'un AUV du SNAME (Society of Naval Architects and Marine Engineers).

Cependant, notons que dans notre cas, seule la position en z est mesurée par le Depth Sensor, les positions x et y ne sont qu'observables. Cela nous impose donc d'approximer ces valeurs en tenant compte de la dérive selon ces deux axes, c'est-à-dire, d'effectuer du « Dead Recknoning ».

Le sous-marin ainsi paramétré vérifie les équations de cinématique et de dynamique suivantes :

L'équation cinématique est :

$$\dot{n}_q = E(n_q)v$$

Avec :

E : Matrice de transformation $\mathbb{R}$7x6.

$$E = \begin{bmatrix} -2\,{\varepsilon_2}^2 - 2\,{\varepsilon_3}^2 + 1 & 2\,\varepsilon_1\,\varepsilon_2 - 2\,\varepsilon_3\,\eta & 2\,\varepsilon_1\,\varepsilon_3 + 2\,\varepsilon_2\,\eta & 0 & 0 & 0 \\ 2\,\varepsilon_1\,\varepsilon_2 + 2\,\varepsilon_3\,\eta & -2\,{\varepsilon_1}^2 - 2\,{\varepsilon_3}^2 + 1 & 2\,\varepsilon_2\,\varepsilon_3 - 2\,\varepsilon_1\,\eta & 0 & 0 & 0 \\ 2\,\varepsilon_1\,\varepsilon_3 - 2\,\varepsilon_2\,\eta & 2\,\varepsilon_2\,\varepsilon_3 + 2\,\varepsilon_1\,\eta & -2\,{\varepsilon_1}^2 - 2\,{\varepsilon_2}^2 + 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\varepsilon_1}{2} & -\frac{\varepsilon_2}{2} & -\frac{\varepsilon_3}{2} \\ 0 & 0 & 0 & \frac{\eta}{2} & -\frac{\varepsilon_3}{2} & \frac{\varepsilon_2}{2} \\ 0 & 0 & 0 & \frac{\varepsilon_3}{2} & \frac{\eta}{2} & -\frac{\varepsilon_1}{2} \\ 0 & 0 & 0 & -\frac{\varepsilon_2}{2} & \frac{\varepsilon_1}{2} & \frac{\eta}{2} \end{bmatrix}$$

Il s'agit de l'équation de transformation des vitesses en fonction du référentiel. Remarquons que la matrice E ne présente pas de singularités grâce à la représentation des angles en quaternions.

L'équation dynamique est, quant à elle, l'équation caractéristique d'un corps rigide dans un fluide idéal, calculée en utilisant la seconde loi de Newton :

$$M\dot{v} + C(v)v + D(v)v + g(n) = \tau + \tau e$$

Avec :

M : La matrice d'inertie du sous-marin (incluant la masse ajoutée d'eau déplacée) $\mathbb{R}$6x6

C : La matrice Coriolis $\mathbb{R}$6x6

D : Matrice d'amortissement $\mathbb{R}$6x6

g : Vecteur gravité et de flottaison (Hydrostatique) $\mathbb{R}$6x1

τ : Vecteur de la commande $\mathbb{R}$6x1

τe : Vecteur des perturbations environnantes $\mathbb{R}$6x1

La matrice M est ici diagonale, car nous négligeons les effets de couplages hydrodynamiques en raison des symétries de notre dispositif et de la faible vitesse de déplacement.

Il est important de savoir qu'à l'heure actuelle, les coefficients de la matrice D n'ont pas été calculés explicitement, mais approximés. En effet, il est très difficile de déterminer ces constantes, car elles sont non linéaires et fortement couplées. Il est en revanche possible de les approximer via des logiciels de simulation de type Ansys. C'est ce sur quoi travaille une partie du pôle mécanique du club SONIA. De plus, la matrice D peut également être approximée sous forme découplée en suivant les mêmes arguments que pour la matrice M.

Enfin, le vecteur des perturbations est modélisé à l'aide de superpositions de sinus et cosinus afin de garantir la robustesse du dispositif simulé. Cependant, la commande du sous-marin est aussi testée régulièrement en condition réelle afin de vérifier la robustesse des implémentations effectuées.

Ces deux équations nous donnent donc une représentation d'un **système non linéaire**.

$$\begin{pmatrix} \dot{n_q} \\ \dot{\mathrm{v}} \end{pmatrix} = \begin{pmatrix} \mathrm{E(e)\ v} \\ M^{-1}[-\mathrm{C(v)v} - \mathrm{D(v)v} - g_b(n_q) + \tau\mathrm{E} + \mathrm{B\ u}] \end{pmatrix}$$

1.7. Stabilité du système non-linéaire en boucle ouverte

Suivant les recommandations du livre *Guidance and Control of Ocean Vehicules de Thor I. Fossen* (1994) p102, le candidat de Lyapunov choisi est l'énergie totale du système :

$$V(\eta, \dot{\eta}) = \frac{1}{2}\dot{\eta}^T\, M_\eta(\eta)\dot{\eta} + \int_0^\eta g_\eta^T(z)\, dz$$

Avec M(n) la forme générale de notre matrice d'inertie et g notre matrice de gravité et de flottaison précédemment introduites.

Or, la fonction candidate est définie positive si et seulement si la matrice d'inertie M est positive, ce qui est vérifié pour les véhicules sous-marins tel que le nôtre où la masse ajoutée est constante.

Et sa dérivée par rapport au temps :

$$\dot{V} = \dot{\eta}^T\, [M_\eta(\eta)\ddot{\eta} + g_\eta(\eta)] + \frac{1}{2}\dot{\eta}^T M_\eta(\eta)\dot{\eta}$$

Qui se synthétise de la sorte :

$$\dot{V} = -\dot{\eta}^T D_\eta \dot{\eta} = -v^T D_\eta v$$

La dérivée est définie négative si et seulement si la matrice d'amortissement D est positive, ce qui est vérifié, car le système est dissipatif.

De plus, la fonction V est radialement non bornée.

Ainsi, il existe une fonction de Lyapunov qui garantit que le système en boucle ouverte est **globalement asymptotiquement stable (G.A.S)**

1.8. Linéarisation du système en boucle ouverte

Dans le cadre de ce rapport nous aurons besoin de linéariser le système en boucle ouverte afin d'appliquer des techniques de commande linéaire. De plus, nous verrons plus tard que posséder une matrice jacobienne analytique sera utile lors de l'implémentation du MPC.

1.8.1. Jacobienne Analytique

Pour commencer, nous allons linéariser le système en boucle ouverte en trouvant les matrices jacobiennes analytiques. Il s'agit d'une matrice paramétrée qui prend en argument un point de linéarisation. Cela nous permet de trouver le modèle linéaire autour de n'importe quel point rapidement. Nous avons suivi la démarche du livre *Guidance and Control of Ocean Vehicules de Thor I. Fossen* (1994) p99-100,

Si nous perturbons les équations du système en boucle ouverte de sorte que :

$$\Delta v = v(t) - v_0(t)$$

$$\Delta \eta = \eta(t) - \eta_0(t)$$

Nous pouvons linéariser l'équation dynamique de la façon suivante :

$$M\,\Delta\dot{v} + \frac{\partial(C(v)v)}{\partial v}|_{v_0}\,\Delta v + \frac{\partial(D(v)v)}{\partial v}|_{v_0}\,\Delta v + \frac{\partial(g(\eta))}{\partial \eta}|_{\eta_0}\,\Delta\eta = tm\,\Delta u$$

Maintenant, si nous perturbons l'équation cinématique, nous avons le résultat suivant :

$$(\dot{\eta}_0 + \Delta\dot{\eta}) = E(\eta_0 + \Delta\eta)\,[v_0 + \Delta v]$$

Si nous remplaçons $\dot{\eta}_0$par $E(\dot{\eta}_0)v_0$, et que nous isolons $\Delta\dot{\eta}$, nous trouvons le résultat suivant.

$$\Delta\dot{\eta} = E(\eta_0 + \Delta\eta)\Delta v + [E(\eta_0 + \Delta\eta) - E(\eta_0)]v_0$$

Nous pouvons remarquer que l'équation contient une double perturbation. Si on néglige les termes du 2e ordre et que nous développons la série de Taylor, nous trouvons le résultat suivant (énorme merci à David Bensoussan pour la démonstration) :

$$\Delta\dot{\eta} = E(\eta_0)\,\Delta v + \left[\frac{\partial(E(\eta_0))}{\partial\eta}\Delta\eta\right] v_0$$

Dans la littérature, il est courant de substituer le deuxième terme de l'addition par une variable, ce qui donne :

$$\dot{E}(v_0, \eta_0) = \frac{\partial(E(\eta_0))}{\partial\eta}\, v_0$$

Nous pouvons donc réécrire l'équation de perturbation, ce qui donne :

$$\Delta\dot{\eta} = \dot{E}(v_0, \eta_0) * \Delta\eta + E(\eta_0)\,\Delta v$$

Si nous mettons les équations linéarisées sous la forme d'équation d'états, nous trouvons les matrices suivantes

$$A(\mathrm{x}_0) = \begin{bmatrix} \dot{E}(v_0, \eta_0) & E(\eta_0) \\ -M^{-1}g(\eta_0) & -M^{-1}(C(v_0) + D(v_0)) \end{bmatrix}, \mathbb{R}^{13\mathrm{x}13}$$

$$B = \begin{bmatrix} 0 \\ M^{-1}\,T_m \end{bmatrix}, \mathbb{R}^{13\mathrm{x}8}$$

Où :

$$\mathrm{x}_0 = [\eta_0\ v_0]^T \quad g(\eta_0) = \frac{\partial(g(\eta))}{\partial\eta}|_{\eta_0} \quad D(v_0) = \frac{\partial(D(v)v)}{\partial v}|_{v_0} \quad C(v_0) = \frac{\partial(C(v)v)}{\partial v}|_{v_0}$$

Vu que nous assumons que nous mesurons tous nos états, la matrice C peut s'écrire de la façon suivante :

$$C = I(13)$$

Nous avons été en mesure de linéariser notre système en boucle ouverte de façon analytique et nous pouvons donc remarquer que seulement notre matrice A est non linéaire.

1.8.2. Linéarisation autour d'un point d'équilibre

Maintenant que nous avons trouvé nos équations linéarisées, il est possible de calculer les matrices du système en boucle ouverte linéarisées en un point d'équilibre. On calcule le point correspondant au vecteur d'état x_{ss} tel que f(x_{ss}) = 0. Ce point correspond à un état pour lequel le sous-marin n'évolue plus, c'est-à-dire un état pour lequel le résultat de toutes les équations différentielles est nul.

On considère alors que l'entrée du système en boucle ouverte est nulle, soit que la commande d'équilibre est égale à un vecteur nul. On peut ainsi déduire par analyse numérique que le point d'équilibre est :

$$x_{ss} = [0\ 0\ 0.2243\ 0.9999\ -0.0118\ -0.0039\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

$$u_{ss} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

Maintenant que nous avons déterminé un point d'équilibre, nous pouvons linéariser notre représentation d'états. Cela se fait en évaluant les matrices jacobiennes analytiques au point d'équilibre. Ainsi, on obtient nos matrices A, B, C et D constituant notre modèle linéaire.

1.9. Modèle linéarisé discret

La commande du sous-marin est exécutée sur un ordinateur embarqué, ce qui nécessite la discrétisation du modèle. Pour ce faire, nous allons utiliser un bloqueur d'ordre 0. Nous verrons plus tard qu'on ne pourra pas toujours utiliser la fonction Matlab « c2d » pour arriver à nos fins. C'est pourquoi nous allons prendre le temps de bien définir nos matrices de transition. Nous pouvons commencer par la matrice de A_d qui ce calcul de la façon suivante :

$$A_d = e^{A*Ts}$$

Malheureusement, la matrice A n'est pas inversible, cependant nous savons que les moteurs n'ont pas d'impact direct sur l'équation cinématique. Le modèle complet comprend 13 états, mais pour l'équation dynamique, nous pouvons ne considérer que les 6 derniers états correspondants aux vitesses linéaires (u, v, w) et angulaires (p, q, r). Cette matrice A réduite est inversible et nous pouvons trouver la matrice de transition B_d :

$$B_d = \begin{bmatrix} \mathbf{0}_{7\text{x}8} \\ A_{réduit}^{-1}\left(A_{d_{réduit}} - \mathbf{I}(6)\right) B \end{bmatrix}$$

Les matrice C_d et D_d sont identiques à leurs homologues continues :

$$C_d = C \qquad D_d = D$$

Pour ce faire, on choisit une fréquence d'échantillonnage de 10Hz correspondant à la fréquence de notre capteur DVL : $f_s = 10\ Hz, T_s = 0.1\ sec$.

1.10. Stabilité du système linéarisé en boucle ouverte

Pour étudier la stabilité, on peut calculer les valeurs propres de notre matrice A linéarisée au point d'équilibre. On souhaite obtenir des valeurs propres dont la partie réelle est négative pour le système en boucle ouverte continu ou des valeurs propres situés à l'intérieur du cercle unitaire dans le plan complexe pour le système en boucle ouverte discret afin de garantir sa stabilité.

Le calcul des valeurs propres du modèle continu linéarisé et réduit aux états de vitesse donne :

$$\text{pôles}_c = \begin{pmatrix} -5.7070 \\ -2.6507 \\ -1.3659 \\ -0.4989 \\ -0.4607 \\ -0.4581 \end{pmatrix}$$

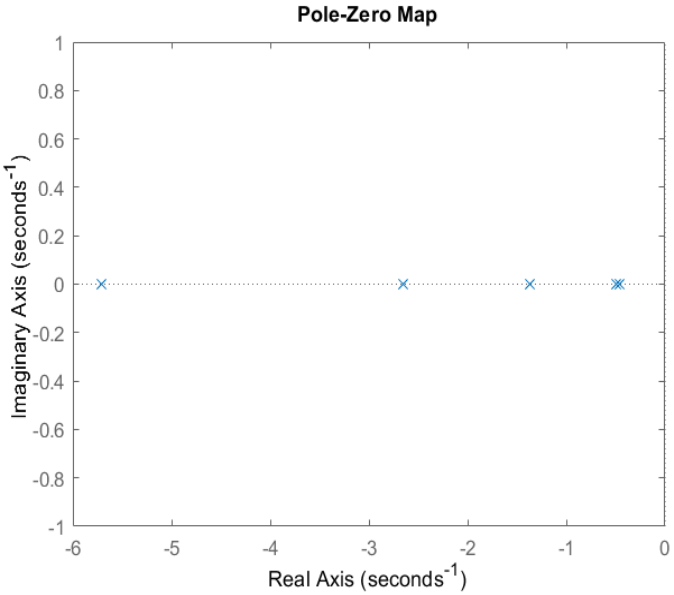

Figure 8 - Pôles du système BO continu

On constate que tous les pôles continus sont strictement négatifs, ce qui implique la stabilité du système en boucle ouverte.

On peut obtenir les pôles discrets en calculant les valeurs propres de la matrice A discrète, ou en les calculant à partir des pôles continus en utilisant la relation $p\hat{o}le_{discret} = e^{p\hat{o}le_{continu} * Ts}$. Dans les deux cas, nous obtenons :

$$p\hat{o}les_d = \begin{pmatrix} 0.5651 \\ 0.7671 \\ 0.8723 \\ 0.9513 \\ 0.9550 \\ 0.9552 \end{pmatrix}$$

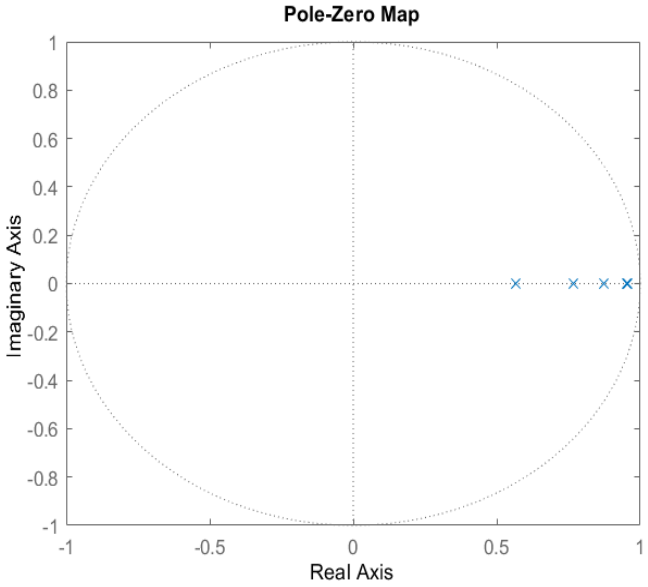

Figure 9 - Pôles du système BO discret

On constate que tous les pôles discrets sont à l'intérieur du cercle unitaire, ce qui montre que le système discret en boucle ouverte est lui aussi bien stable.

1.11. Contraintes

Comme précisé en début de section, le sous-marin a pour objectif de participer à une compétition durant laquelle il aura à effectuer un certain nombre d'activités différentes. L'AUV est donc équipé de divers modules qui tirent sur la batterie et l'espace du CPU embarqué. Ainsi, nous avons comme contraintes d'utiliser un minimum de capacité processeur, de l'ordre de 20% d'utilisation maximale du CPU. De plus, l'ensemble du contrôle se fera via le système embarqué sans connexion à un PC. Il est donc impératif que nous n'utilisions que des fonctions Matlab compilables en C ou en C++ afin de pouvoir les importer et les faire tourner sur le système embarqué. Nous avons également des contraintes d'effort au niveau des moteurs qui sont bornés à 50N dans le sens horaire et 40N dans le sens antihoraire. Nous devrons donc vérifier les gains maximum appliqués en entrée des moteurs par les différentes lois de commandes que nous allons tester. D'autre part, il est important pour le club SONIA que les méthodes de contrôle adoptées pour un sous-marin d'une génération puissent être migrés dans la version de l'AUV de génération suivante. Nous souhaitons donc développer une loi de commande robuste face aux changements d'architecture et de fonctionnalités de l'appareil.

1.12. Représentation commandable et observable

Le modèle réduit aux états de vitesse étant commandable et observable, il est possible de le mettre sous une forme canonique de commandabilité et d'observabilité. Il faut bien sûr prendre en compte le fait qu'il s'agit ici d'un système multientrées.

Notre sous-marin possède $m = 8$ entrées et $p = 6$ sorties et se présente sous la forme suivante :

$$\begin{bmatrix} y_1(s) \\ \dots \\ y_6(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & \cdots & G_{18}(s) \\ \vdots & \ddots & \vdots \\ G_{61}(s) & \cdots & G_{68}(s) \end{bmatrix}$$

Sur Matlab, il est possible d'afficher chaque élément de la matrice de fonction de transfert G(s) avec la fonction *ss2tf()* qui renvoie 6 fonctions de transferts (associées aux 6 sorties) pour chacune des 8 entrées. Ces fonctions de transfert possèdent un dénominateur commun de degré n=6 dont on peut extraire les coefficients :

$$D(s) = 1.0000\mathrm{s}^6 + 11.1413\mathrm{s}^5 + 40.9976\mathrm{s}^4 + 64.9066\mathrm{s}^3 + 48.0846\mathrm{s}^2 + 16.6262\mathrm{s} + 2.1754$$

Ces coefficients sont les coefficients $a_0, \dots, a_6$ qui seront utiles pour construire les matrices d'états de commandabilité et d'observabilité A_C et A_O.

On peut ensuite récupérer les coefficients des numérateurs des fonctions de transfert pour former les matrices Q_k. Ainsi, la matrice Q_0 est par exemple composée des coefficients des numérateurs associés aux puissances 0 de s. De manière plus générale, pour $k \in [\![0, \dots, 5]\!]$ la matrice Q_k s'écrit :

$$Q_k = \begin{bmatrix} G_{11k} & \cdots & G_{18k} \\ \vdots & \ddots & \vdots \\ G_{61k} & \cdots & G_{68k} \end{bmatrix}$$

Avec G_{ijk} le coefficient de la k-ième puissance du numérateur de la fonction de transfert G_{ij} (i étant le numéro de la sortie et j le numéro de l'entrée). Les matrices Q_k sont donc de dimension 6x8.

Notre fonction de transfert s'écrit alors :

$$G(s) = \frac{Q_0 + sQ_1 + \cdots + s^5 Q_5}{a_0 + a_1 s + \cdots + a_6 s^6}$$

La représentation commandable de notre sous-marin s'écrit :

$$A_C = \begin{bmatrix} O_8 & I_8 & O_8 & O_8 & O_8 & O_8 \\ O_8 & O_8 & I_8 & O_8 & O_8 & O_8 \\ O_8 & O_8 & O_8 & I_8 & O_8 & O_8 \\ O_8 & O_8 & O_8 & O_8 & I_8 & O_8 \\ O_8 & O_8 & O_8 & O_8 & O_8 & I_8 \\ -2.1754I_8 & -16.6262I_8 & -48.0648I_8 & -64.9066I_8 & -40.9976I_8 & -11.1413I_8 \end{bmatrix}, \mathbb{R}^{48x48}$$

$$B_C = \begin{bmatrix} O_8 \\ O_8 \\ O_8 \\ O_8 \\ O_8 \\ I_8 \end{bmatrix}, \mathbb{R}^{48x8}$$

$$C_C = [Q_0 \quad Q_1 \quad Q_2 \quad Q_3 \quad Q_4 \quad Q_5], \mathbb{R}^{6x48}$$

Comme le nombre d'entrées et le nombre de sorties sont différents, on ne peut pas transposer ces matrices pour obtenir la forme canonique d'observabilité. Pour ce faire, on écrit directement les matrices canoniques d'observabilité :

$$A_O = \begin{bmatrix} O_6 & O_6 & O_6 & O_6 & O_6 & -2.1754I_6 \\ I_6 & O_6 & O_6 & O_6 & O_6 & -16.6262I_6 \\ O_6 & I_6 & O_6 & O_6 & O_6 & -48.0648I_6 \\ O_6 & O_6 & I_6 & O_6 & O_6 & -64.9066I_6 \\ O_6 & O_6 & O_6 & I_6 & O_6 & -40.9976I_6 \\ O_6 & O_6 & O_6 & O_6 & I_6 & -11.1413I_6 \end{bmatrix}, \mathbb{R}^{36x36}$$

$$B_O = \begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \end{bmatrix}, \mathbb{R}^{36x6}$$

$$C_O = [O_6 \quad O_6 \quad O_6 \quad O_6 \quad O_6 \quad I_6], \mathbb{R}^{6x36}$$

2. Commandes du sous-marin

2.1. Objectifs de performance

Nos exigences en termes de performances découlent de l'analyse précédente de nos contraintes, mais également des besoins de l'AUV pour participer et se démarquer lors de la compétition RoboSub. Ainsi nous souhaitons que le système en boucle fermée et avec correcteur admette une **réponse nulle en régime permanents** et un **dépassement minimal**. De plus, l'AUV doit pouvoir se déplacer à une vitesse de croisière d'**1m/s** pour atteindre les différentes tâches de la compétition de manière efficace. D'autre part, le **temps de réponse** du sous-marin doit être dans un ordre de grandeur **de 1s**, sans quoi celui-ci réagira à retardement aux différentes perturbations aquatiques et sa précision de mouvement et de positionnement en sera largement dégradée. C'est pourquoi nous fixons notre objectif de temps de réponse à 5% (temps que le système met pour atteindre 95% de la valeur finale) à maximum 2s. Enfin comme nous l'avons dit plus tôt, nous souhaitons modéliser une loi de **commande discrète** afin de la migrer directement sur l'ordinateur embarqué de l'AUV.

2.2. Commande précédente

La commande précédente, mise en place sur le sous-marin AUV7, a été développée au sein d'un projet SYS802 de 2018 par Olivier Lavoie et Rafik Chennouf. A ce moment, les étudiants n'avaient pas accès à une représentation du système en boucle ouverte et avaient donc déterminé l'ensemble des fonctions de transfert de manière expérimentale suivant la méthode « Black-Box ». Cela consiste à modéliser le système en boucle ouverte comme une boite noire en approchant mathématiquement des résultats observés en tests piscine par des interpolations.

Au cours de ce projet, les élèves ont mis au point une commande PID découplée sur quatre degrés de liberté (trois degrés de position et une rotation autour de z) et deux contrôleurs (un en vitesse et un en position).

Cette loi de commande a cependant montré des performances assez faibles : la vitesse de pointe du sous-marin était bornée à 0.2m/s, la loi de commande ne stabilisait ni la rotation en y ni celle en x et négligeait tous les effets de couplages. Cela ne permettait donc au sous-marin de se déplacer qu'en lignes droites par à-coups de 3m et de manière assez oscillante.

2.3. Commandes en vitesse

Dans un premier temps, nous allons nous concentrer sur l'asservissement du sous-marin en vitesse. Nous allons tout d'abord effectuer un placement de pôles par retour d'état avant de concevoir un régulateur PID.

Le modèle complet comprend 13 états, mais pour l'asservir en vitesse, nous pouvons ne considérer que les 6 derniers états correspondants aux vitesses linéaires (u, v, w) et angulaires (p, q, r). De la sorte, nous pouvons nous affranchir des autres états et travailler avec des matrices ne comportant que 6 lignes. Ces 6 états sont suffisants pour faire un asservissement du sous-marin en vitesse.

Nous considérons alors les matrices réduites aux états de vitesse suivantes :

$$A_v = \begin{bmatrix} -0.465 & -1.9e-5 & 1.59e-4 & -4.35e-4 & 0.0234 & -0.00506 \\ -1.42e-5 & -0.464 & 6.78e-4 & -0.0729 & 2.4e-4 & 0.00214 \\ 1.02e-4 & 5.82e-4 & -0.499 & 0.00757 & -9.28e-4 & -2.64e-5 \\ -0.00196 & -0.438 & 0.053 & -5.7 & 0.0187 & 0.0199 \\ 0.15 & 0.00205 & -0.00928 & 0.0268 & -1.36 & 8.03e-6 \\ -0.0152 & 0.00856 & -1.23e-4 & 0.0133 & 3.75e-6 & -2.65 \end{bmatrix}$$

$$B_v = \begin{bmatrix} 0.0211 & 0.0211 & 0.0218 & 0.0218 & -0.00182 & -0.00181 & 0.00185 & 0.00178 \\ -0.0153 & 0.0156 & -0.0156 & 0.0153 & -0.00342 & 0.00339 & 0.00357 & -0.00353 \\ -7.08e-5 & 9.37e-5 & -6.74e-5 & 9.72e-5 & -0.021 & 0.0211 & -0.0218 & 0.0217 \\ 0.063 & -0.0609 & 0.0603 & -0.0635 & -0.268 & 0.265 & 0.279 & -0.276 \\ 0.019 & 0.0196 & 0.019 & 0.0196 & 0.106 & 0.105 & -0.108 & -0.103 \\ -0.172 & -0.173 & 0.176 & 0.175 & 6.2e-4 & -6.21e-4 & -6.47e-4 & 6.47e-4 \end{bmatrix}$$

$$C_v = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_v = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Étant donné que nous allons travailler dans le domaine discret, on peut calculer le système discret en boucle ouverte grâce à la fonction « c2d » de Matlab. Pour ce faire, on choisit une fréquence d'échantillonnage de 10Hz correspondant à la fréquence de notre capteur DVL : $f_s = 10\ Hz$, $T_s = 0.1\ sec$.

Ainsi, on obtient les matrices réduites discrètes suivantes :

$$A_d = \begin{bmatrix} 0.955 & -1.05e-6 & 1.41e-5 & -3.01e-5 & 0.00214 & -4.34e-4 \\ -7.84e-7 & 0.955 & 4.9e-5 & -0.00542 & 1.65e-5 & 1.78e-4 \\ 9.05e-6 & 4.2e-5 & 0.951 & 5.62e-4 & -8.4e-5 & -1.69e-6 \\ -1.35e-4 & -0.0325 & 0.00393 & 0.566 & 0.00133 & 0.00131 \\ 0.0137 & 1.42e-4 & -8.4e-4 & 0.00189 & 0.873 & -6.74e-7 \\ -0.0013 & 7.12e-4 & -7.9e-6 & 8.76e-4 & -3.15e-7 & 0.767 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.00207 & 0.00207 & 0.00213 & 0.00213 & -1.66e-4 & -1.66e-4 & 1.69e-4 & 1.63e-4 \\ -0.00151 & 0.00154 & -0.00154 & 0.00151 & -2.54e-4 & 2.52e-4 & 2.65e-4 & -2.63e-4 \\ -5.05e-6 & 7.23e-6 & -4.82e-6 & 7.45e-6 & -0.00205 & 0.00207 & -0.00212 & 0.00211 \\ 0.00481 & -0.00468 & 0.00464 & -0.00485 & -0.0204 & 0.0202 & 0.0213 & -0.021 \\ 0.0018 & 0.00184 & 0.0018 & 0.00184 & 0.00988 & 0.00988 & -0.0101 & -0.00969 \\ -0.0151 & -0.0152 & 0.0155 & 0.0154 & 4.1e-5 & -4.09e-5 & -4.27e-5 & 4.26e-5 \end{bmatrix}$$

$$C_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Il s'agit des matrices discrètes **réduites** aux états de vitesses, mais pour simplifier on ne rajoute pas de v pour le préciser.

2.3.1. Placement de pôles par retour d'état

Nous avons voulu mettre en place une commande de placement de pôles par retour d'états. En ajoutant un gain dans la boucle de retour du système en boucle fermée, il est facile de calculer la valeur de ce gain permettant de placer les pôles aux valeurs désirées. De cette manière, on peut rendre le sous-marin plus performant, toujours en essayant de concilier rapidité et dépassement minimal.

Cette méthode a l'avantage d'être relativement simple et d'offrir la possibilité de choisir directement les pôles du système en boucle fermée, ce qui permet notamment d'assurer la stabilité. Cependant, il faut tout de même noter qu'il s'agit d'une méthode nécessitant une grande précision du modèle et qui n'est donc pas forcément robuste vis-à-vis d'un éventuel changement physique du sous-marin. De plus, le calcul au niveau du gain dans la boucle de retour est susceptible de générer un retard.

Comme précisé plus haut, sur les 13 états du système, on ne s'intéresse qu'aux 6 états de vitesses, c'est-à-dire aux matrices A_d, B_d, C_d, D_d (matrices réduites aux états de vitesse discrètes).

Une remarque importante ici est de noter que le retour d'état est également un retour de sortie puisque la matrice C_d du système est une matrice diagonale et que la matrice D_d est nulle. On peut donc appliquer notre gain dans la boucle de retour au vecteur d'état aussi bien qu'au vecteur de sortie puisqu'ils sont égaux. On utilisera par la suite le terme retour d'état pour parler également de retour de sortie.

La figure suivante montre le schéma-bloc du système en boucle fermée avec retour d'état :

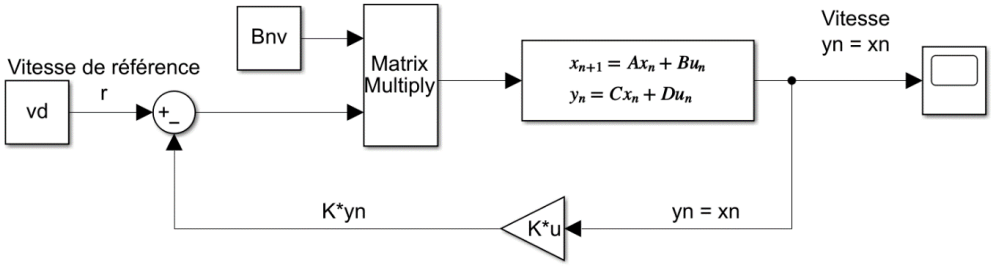

Figure 10 – Système en boucle fermée avec retour d'état

On note que le bloc de gain est noté K*u, mais qu'il correspond en réalité à K*yn (= K*xn), la notation u étant utilisée par Matlab pour désigner l'entrée du gain, c'est-à-dire ici yn (= xn).

On remarque ensuite qu'on n'applique pas directement l'erreur dans l'entrée du système. En effet, il faut rappeler que l'entrée U du système ne correspond pas à un vecteur vitesse, mais à un vecteur de force à appliquer aux 8 moteurs. Or, on désire commander le sous-marin en vitesse et non directement en force de moteur. Pour pallier ce problème, on multiplie en amont le signal par la matrice pseudo-inverse de B, ici notée Bnv. On obtient de la sorte un système en boucle fermée équivalent $(A_d, B_d Bnv \sim I_6, C_d)$, ce qui nous permet d'appliquer directement la commande en vitesse.

On rappelle que les pôles du système en boucle ouverte discret réduit sont les suivants, tous stable, car ils sont positionnés à l'intérieur du cercle unitaire dans le plan complexe :

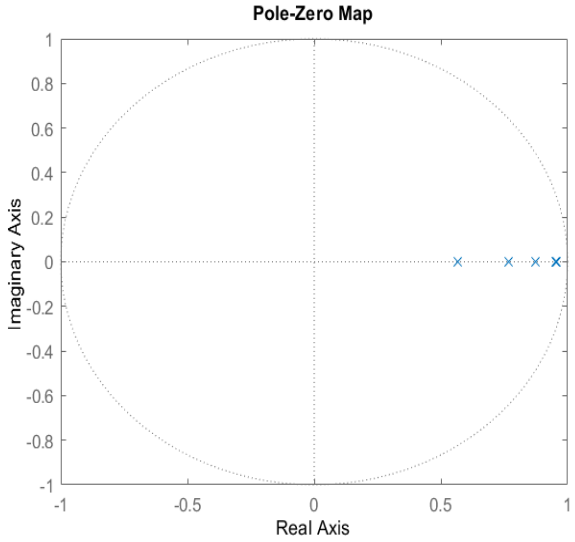

Figure 11 - Pôles discrets en BO

Le retour d'état permet d'obtenir un système en boucle fermée équivalent $(A_d - B_d * K_d, B_d * Bnv, C_d, D_d)$ avec de nouveaux pôles. Le gain K_d détermine ces nouveaux pôles, et on peut le calculer en utilisant la fonction $place(A_d, B_d Bnv)$ de Matlab qui permet de gérer les systèmes multivariables.

De manière expérimentale, on choisit des pôles différents pour améliorer la réponse temporelle. On a trouvé que les pôles $[0.9048 \quad 0.9048 \quad 0.9048 \quad 0.3679 \quad 0.9048 \quad 0.7408]$ (correspondant à des pôles continus $[-1 \quad -1 \quad -1 \quad -10 \quad -1 \quad -3]$) permettent d'avoir des résultats satisfaisants tout en gardant bien sûr le système en boucle fermée stable.

Les figures suivantes montrent la réponse du système en boucle ouverte et avec retour d'état pour une consigne en vitesse $v_d = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$.

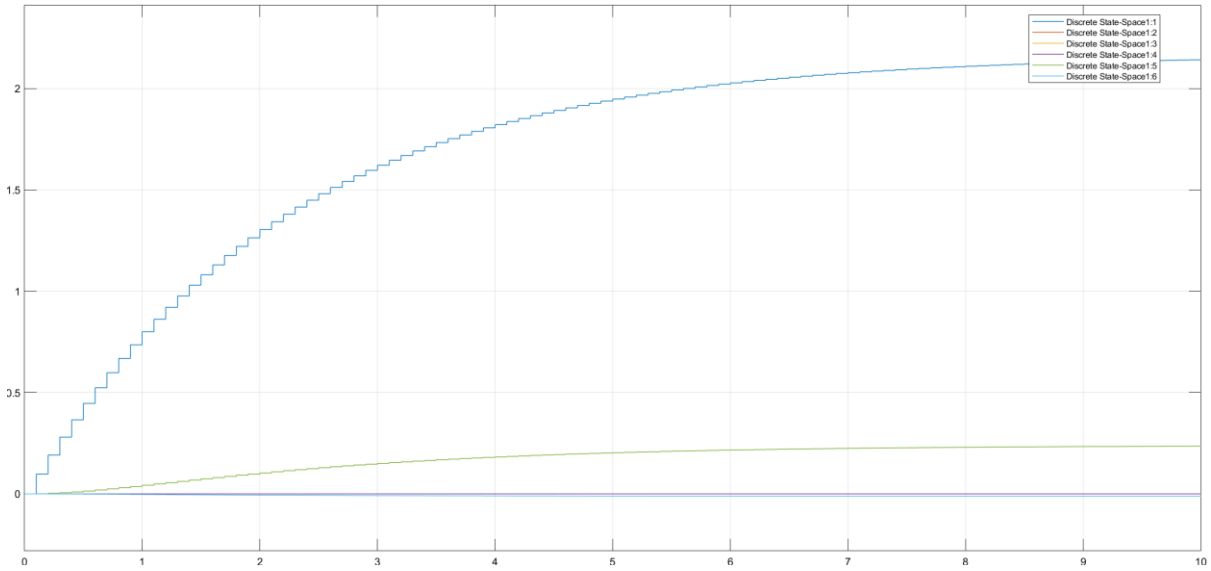

Figure 12 - Réponse du système en BO pour un consigne vd = [1 0 0 0 0 0]'

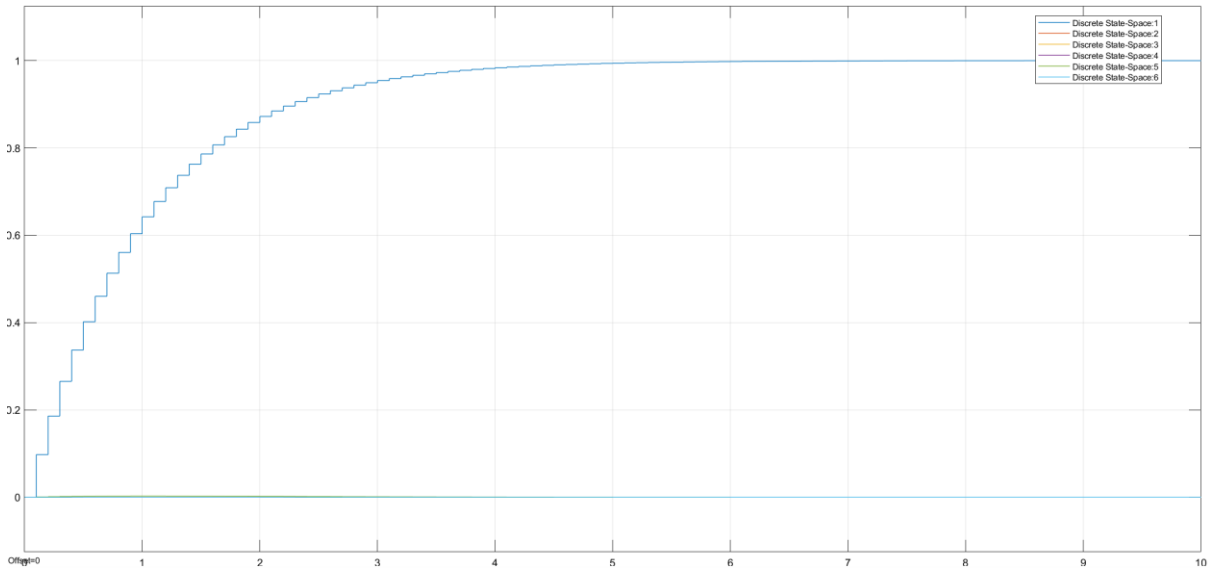

Figure 13 - Réponse du système BF avec retour d'état pour une consigne vd = [1 0 0 0 0 0]'

On remarque que le système en boucle ouverte ne converge pas vers la valeur désirée avec un **dépassement de plus de 100%** et est lent avec **un temps de réponse à 5% égal à 6,5s**. Il présente de plus un effet de couplage important.

Le retour d'état en revanche permet d'obtenir une réponse précise sans dépassement, rapide avec un **temps de réponse à 5% de 3s** et avec très peu d'effet de couplage. Cela reste néanmoins au-dessus de notre objectif de performance en vitesse.

Les figures suivantes montrent la réponse du système en boucle ouverte et en boucle fermée avec retour d'état pour une consigne en vitesse $v_d = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]^T$.

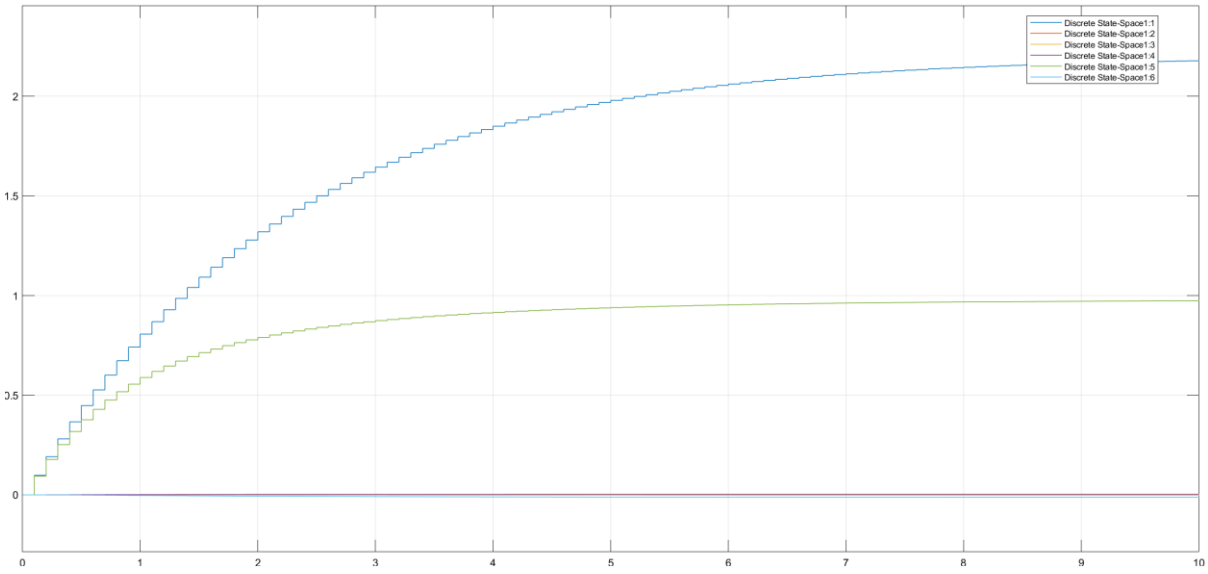

Figure 14 - Réponse du système en BO pour une consigne vd = [1 0 0 0 1 0]'

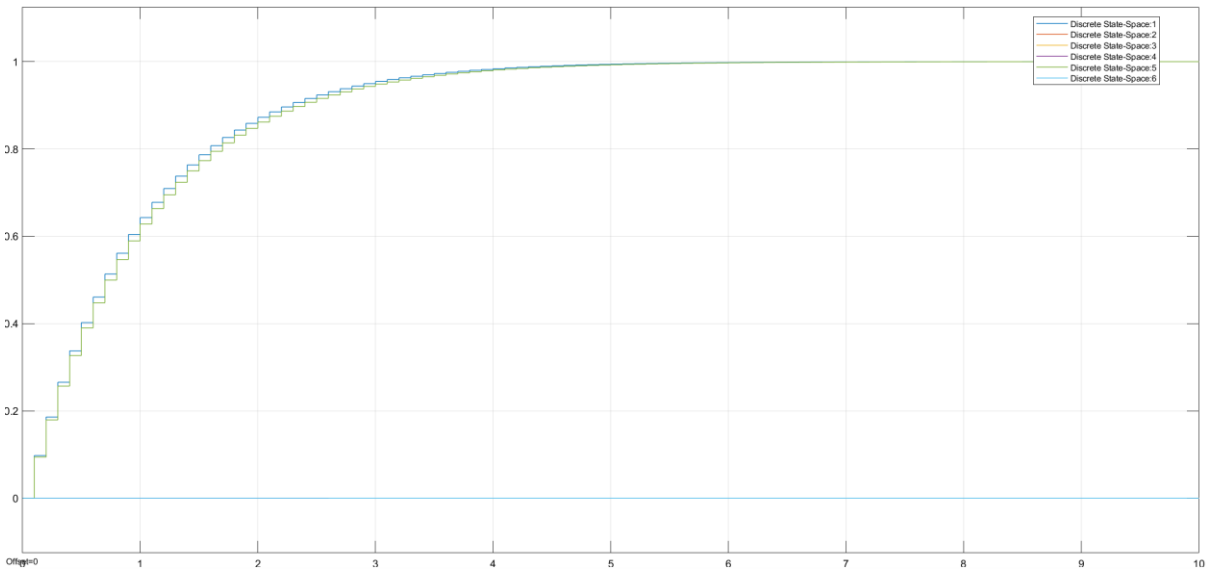

Figure 15 - Réponse du système BF avec retour d'état pour une consigne vd = [1 0 0 0 1 0]'

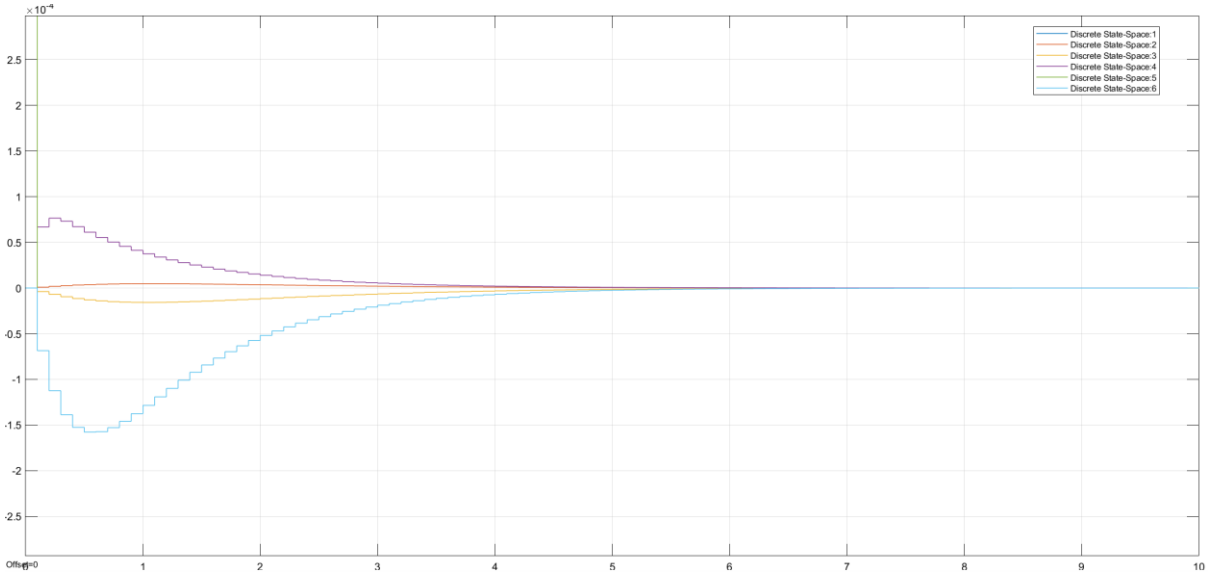

Figure 16 - Réponse du système BF avec retour d'état pour une consigne vd = [1 0 0 0 1 0]' (zoom)

On remarque que le système en boucle ouverte répond mieux à la consigne de vitesse angulaire puisqu'elle n'admet pas de dépassement et est légèrement plus rapide (**environ t5%=4,5s**). Les performances du système en boucle fermée ne changent pas sensiblement avec l'ajout de la vitesse angulaire. On voit notamment qu'il existe encore un effet de couplage, mais il reste négligeable puisqu'il ne dépasse pas **0.0001 rad/sec**.

En pratique, il est intéressant d'envoyer des consignes en vitesse de forme trapézoïdale pour que le sous-marin se déplace sans à-coups. On simule la réponse pour l'entrée suivante (trapèze selon x) :

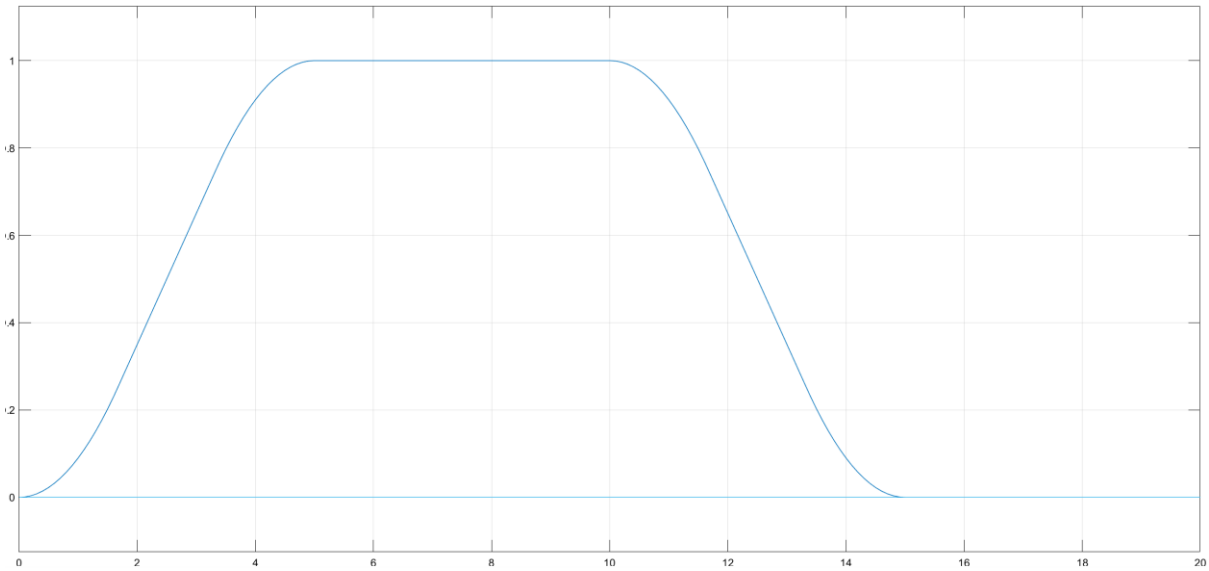

Figure 17 - Consigne en vitesse trapézoïdale

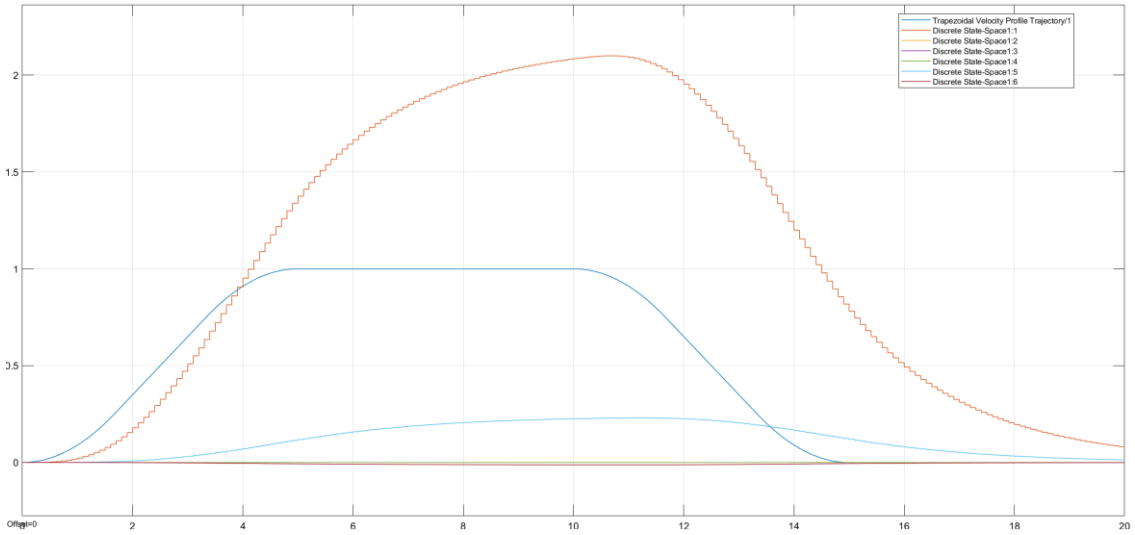

Figure 18 – Consigne (courbe bleue) et réponse du système en BO pour une consigne trapézoïdale

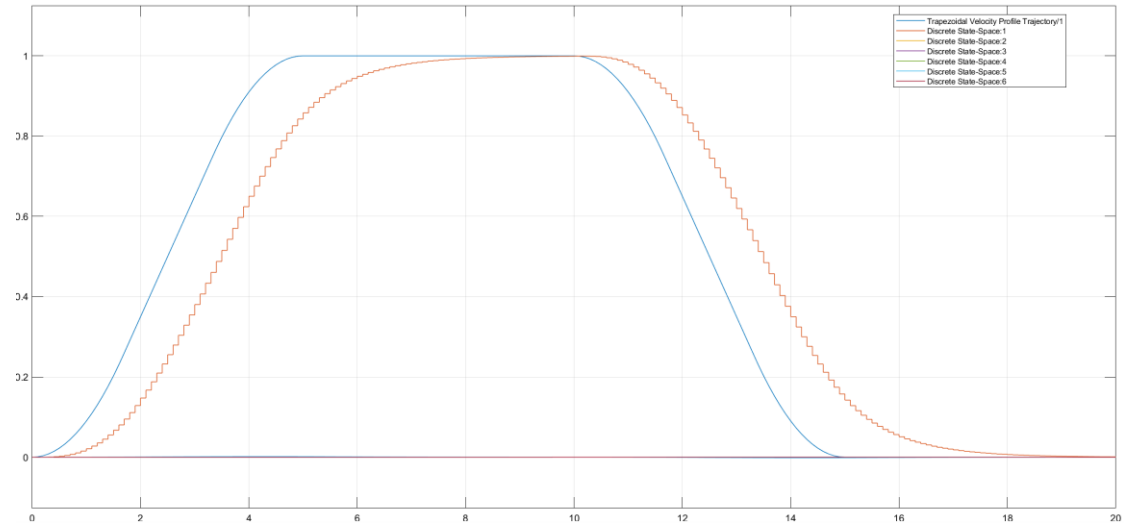

Figure 19 – Consigne (courbe bleue) et réponse du système en BF avec retour d'état pour une consigne trapézoïdale

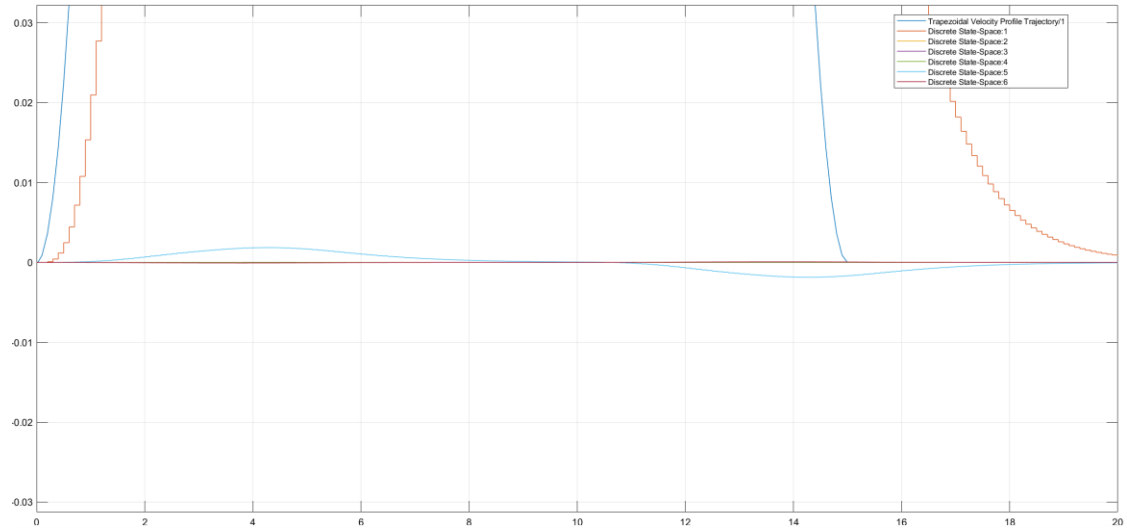

Figure 20 - Réponse du système en BF avec retour d'état pour une consigne trapézoïdale (zoom)

Encore une fois, le système en boucle ouverte admet un fort dépassement (120%) et de nombreux effets de couplages. En revanche, on constate que la réponse du système en boucle fermée avec retour d'état suit bien la consigne avec un **retard d'environ 1 seconde** pendant l'accélération et la décélération. La réponse atteint bien la valeur désirée **(pas de dépassement**) et on remarque que les effets de couplage sont très bien atténués (<1% de la valeur de consigne sur l'axe x).

2.3.2. Correcteur PID

Bien que le sous-marin soit un système MIMO (multi-input multi-output) représenté par des équations d'états, il est possible d'utiliser une correction proportionnelle, intégrale et dérivée. En choisissant des gains adaptés, il est possible d'améliorer la performance du système en boucle fermée pour chacun des états de vitesse correspondant aux 6 derniers états du sous-marin. On peut ainsi avoir un asservissement sur les états u, v, w, p, q, r qui correspondent aux vitesses linéaires (u, v, w) et angulaires (p, q, r). On espère avec un tel correcteur pouvoir éliminer l'erreur statique et les effets de couplage tout en s'attachant à obtenir un temps de réponse rapide et un dépassement minimal.

Un tel correcteur est relativement simple à concevoir puisqu'une fois implémenté, il suffit de jouer avec les gains pour trouver une réponse satisfaisante. De plus, il n'est plus nécessaire d'avoir un modèle très précis comme pour le retour d'état étant donné que les gains ne dépendent pas directement du modèle.

Pour mettre en place une correction PID, on ne peut pas directement prendre en compte les 6 états de vitesse pour appliquer un seul correcteur. On a choisi d'appliquer 6 correcteurs PID, un par état, qu'on va paramétrer un par un. Notre modèle étant discret, chacun des six correcteurs est de la forme suivante :

$$P + I.T_s.\frac{1}{z-1} + D\frac{N}{1 + N.T_s.\frac{1}{z-1}}$$

Avec P, I et D les gains proportionnels, intégral et dérivé. On note également N le coefficient de filtre, lui aussi paramétrable et T_s la période d'échantillonnage.

Le schéma-bloc du système en boucle fermée avec correcteur est le suivant :

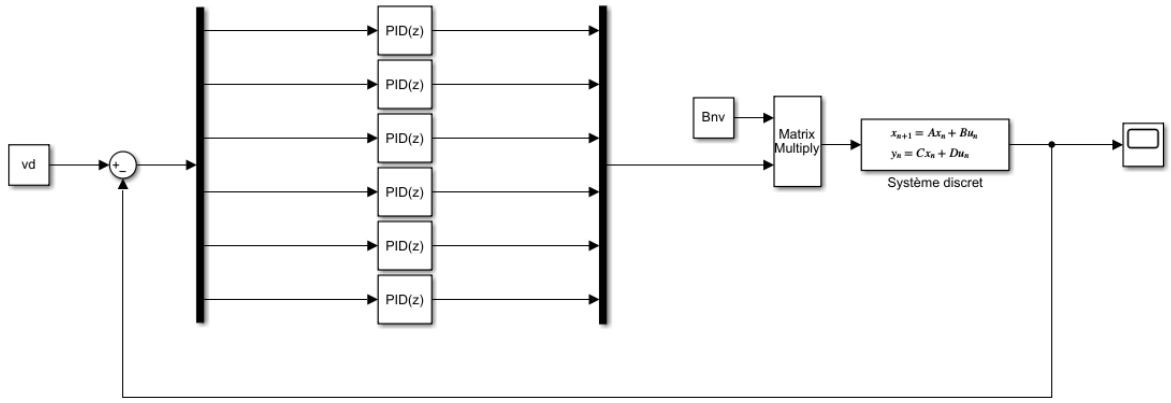

Figure 21 – Système en BF avec correcteur PID et retour unitaire

Le bloc vd correspond à la commande en vitesse, c'est-à-dire un vecteur de taille 6x1 contenant les vitesses désirées pour chaque état. A la sortie du sommateur, on applique les six correcteurs PID indépendants pour chaque vitesse.

De la même manière que pour le retour d'état, on ne contrôle ainsi pas directement les forces des moteurs, mais uniquement les vitesses, car c'est évidemment plus parlant. On utilise exactement de la même façon la matrice pseudo-inverse de B, nommée Bnv qu'on multiplie au signal en sortie de PID qui correspond à un vecteur de vitesses. Le système en boucle fermée équivalent obtenu est $(A_d, B_d Bnv \sim I_6, C_d)$.

De façon expérimentale, on peut choisir les gains des 6 correcteurs afin d'obtenir une réponse conciliant vitesse et dépassement minimal :

On a utilisé ici les gains suivants :

Tableau 1 - Gains PID

	P	I	D	N
PID1	2.43674325166001	1.09995357425763	0.2296764246937	2.3582948693606
PID2	3.85375412683343	2.71852154275701	0.190780814005723	3.46445143203079
PID3	4.7350716991649	7.84165762450361	0.286078254880993	6.55565795122902
PID4	7.28634765669471	45.6013589615017	0.0674891268309797	12.9101634602502
PID5	2.12272529254057	3.43691547038045	0.109013031693667	3.83268620776355
PID6	3.56542524168791	10.5921117462818	0.0790078771698945	6.35214865314064

On remarque que nos valeurs de gain restent bien dans le domaine d'application de 50N délimité par les moteurs.

On peut voir sur la figure suivante la réponse du système (vitesses u, v, w, p, q, r) pour le système en boucle ouverte pour un échelon en vitesse $v_d = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$ (m/s et rad/sec). Le temps de simulation est 5 secondes, et la période d'échantillonnage est de 0.1 seconde.

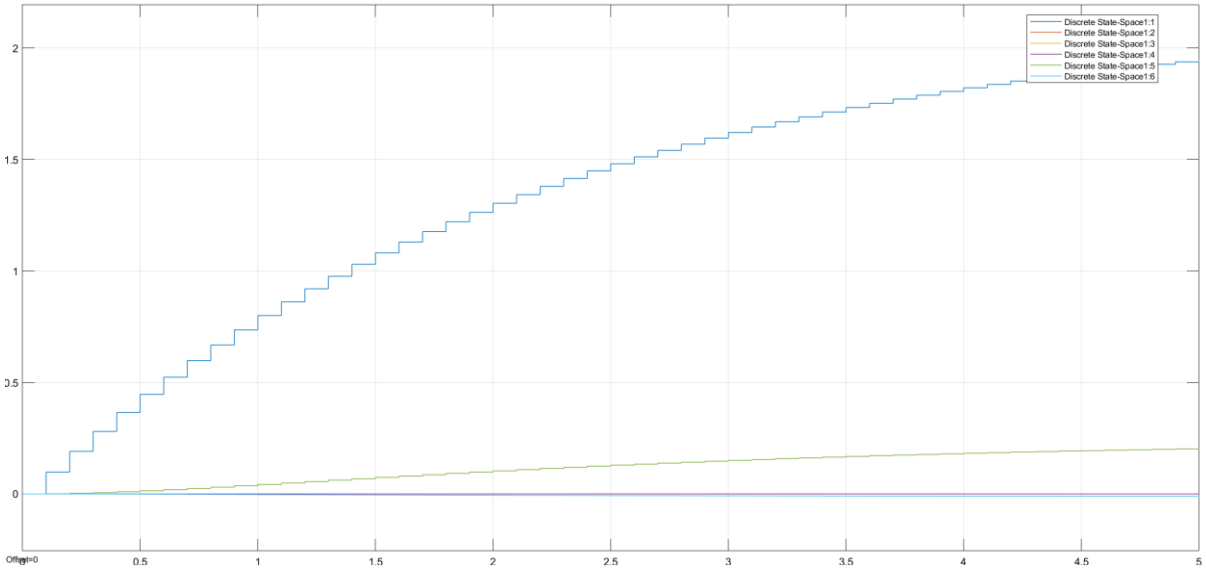

Figure 22 – Réponse (m/s – rad/s) du système en BO pour vd = [1 0 0 0 0 0]'

Le système en boucle ouverte n'atteint toujours pas les valeurs désirées puisqu'on voit clairement un **dépassement de 95%** de la valeur de consigne. De plus, on observe un effet de couplage puisqu'une rotation est introduite autour de l'axe y équivalente à 20% de la consigne selon l'axe x (courbe en vert correspondant à l'état p).

La figure suivante montre la réponse temporelle du système en boucle fermée avec correction PID et retour unitaire, pour la même consigne de vitesse :

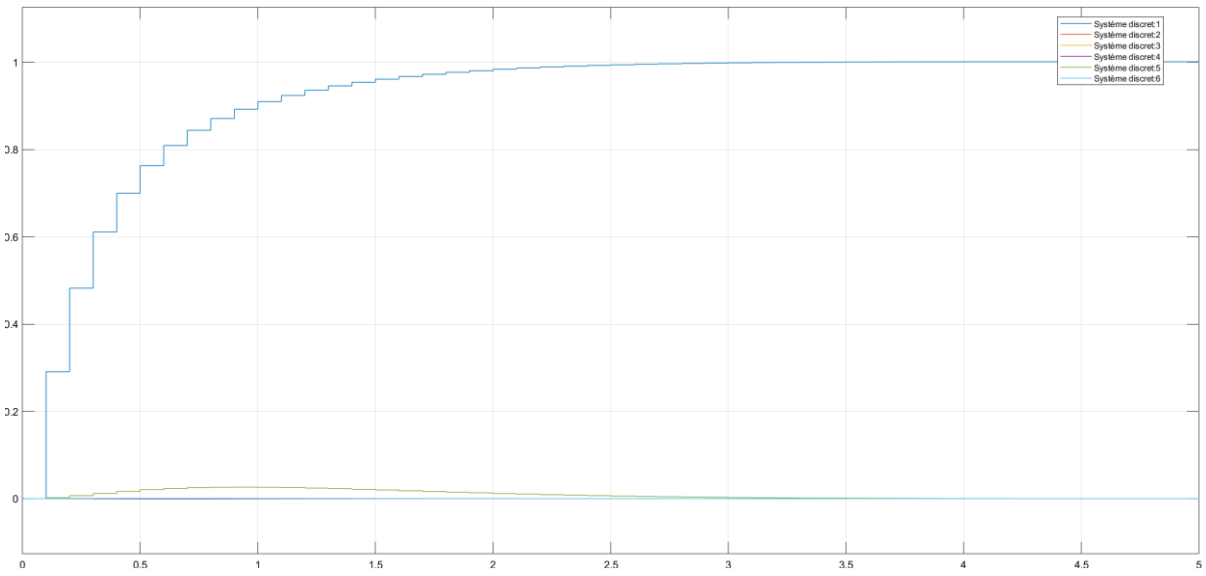

Figure 23 – Réponse (m/s-rad/s) du système en BF avec PID pour vd = [1 0 0 0 0 0]'

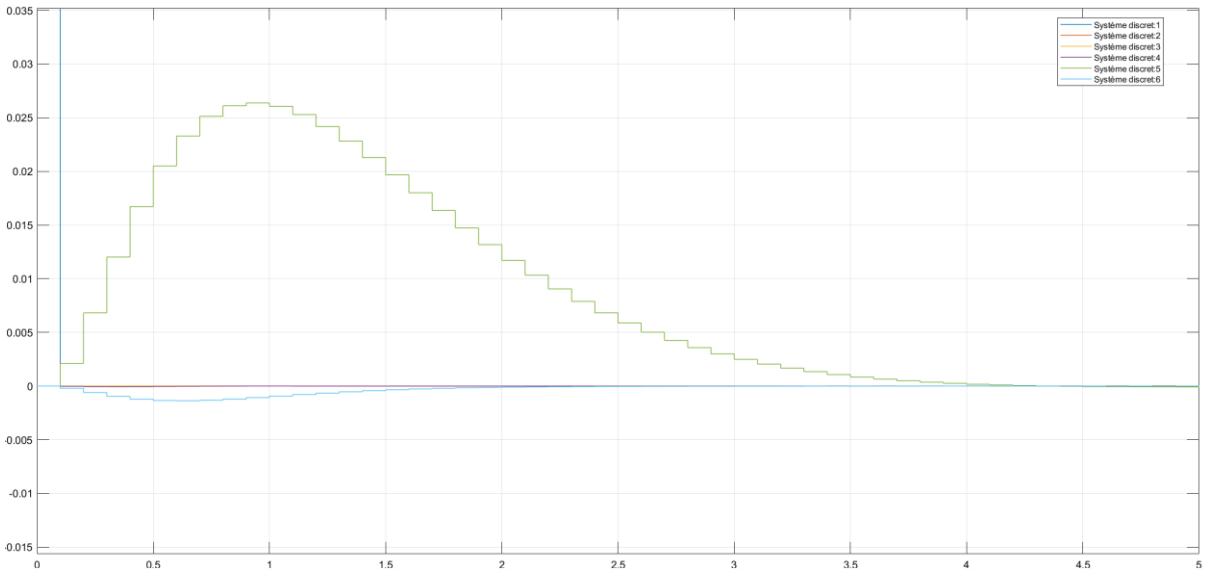

Figure 24 – Réponse (m/s-rad/s) du système en BF avec PID pour vd = [1 0 0 0 0 0]' (zoom)

On voit donc que la correction permet d'annuler l'erreur statique puisque les valeurs finales correspondent aux valeurs commandées (**pas de dépassement**). De plus, l'effet de couplage est fortement atténué puisque nous observons seulement une légère bosse avant la première seconde de 0,026 rad/s soit moins 3% de la valeur de consigne selon l'axe x (courbe verte correspondant à la vitesse angulaire q). Le correcteur PID nous permet également d'avoir un temps de réponse plus rapide avec **un temps de réponse à 5% égal à 1,5s**. Il s'agit d'une meilleure performance que pour le retour d'état et cela correspond à nos objectifs de performance, même si l'effet du couplage est moins atténué.

Nous pouvons voir sur les figures suivantes la simulation du modèle en boucle ouverte et avec correction PID (et retour unitaire) pour une consigne en vitesse $v_d = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]^T$. Là aussi, la correction permet d'avoir une réponse assez rapide, avec peu de dépassement : **10% pour la vitesse angulaire**. **Le temps de réponse à 5% est ici encore d'1,5s, ce qui nous convient.**

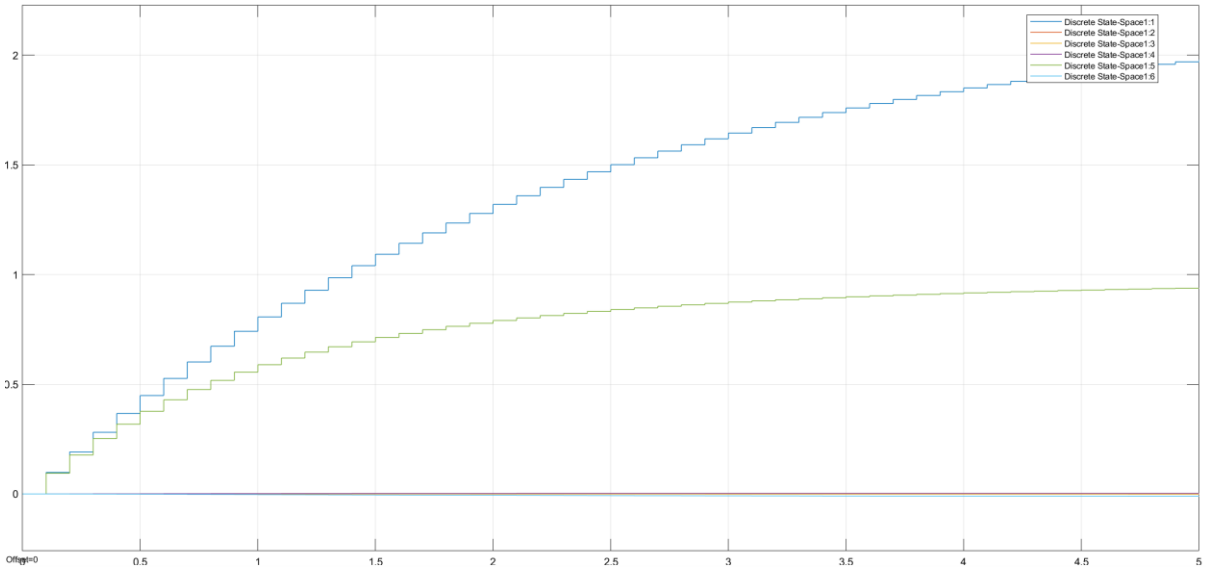

Figure 25 – Réponse (m/s-rad/s) du système en BO pour vd = [1 0 0 0 1 0]'

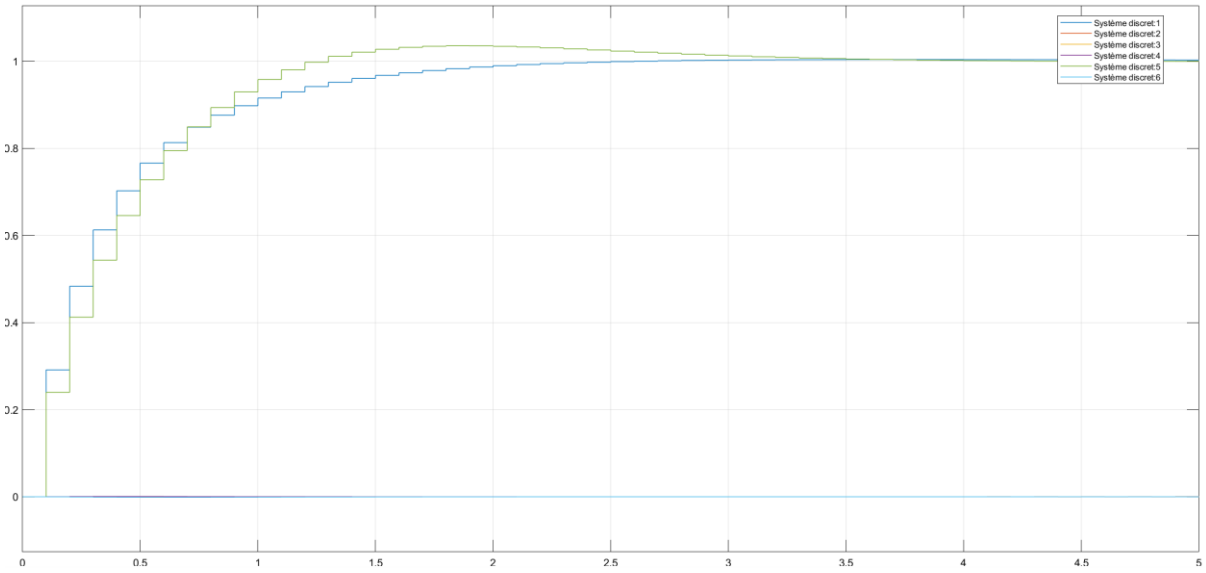

Figure 26 – Réponse (m/s-rad/s) du système en BF avec PID pour vd = [1 0 0 0 1 0]'

Comme pour le retour d'état, nous avons simulé la réponse du sous-marin pour une consigne en vitesse trapézoïdale (courbe bleue) sans et avec correction PID (et retour unitaire). La figure suivante montre la consigne (courbe bleue) et la réponse pour le système en boucle ouverte. On y remarque encore un fort dépassement ainsi que beaucoup d'effets de couplage.

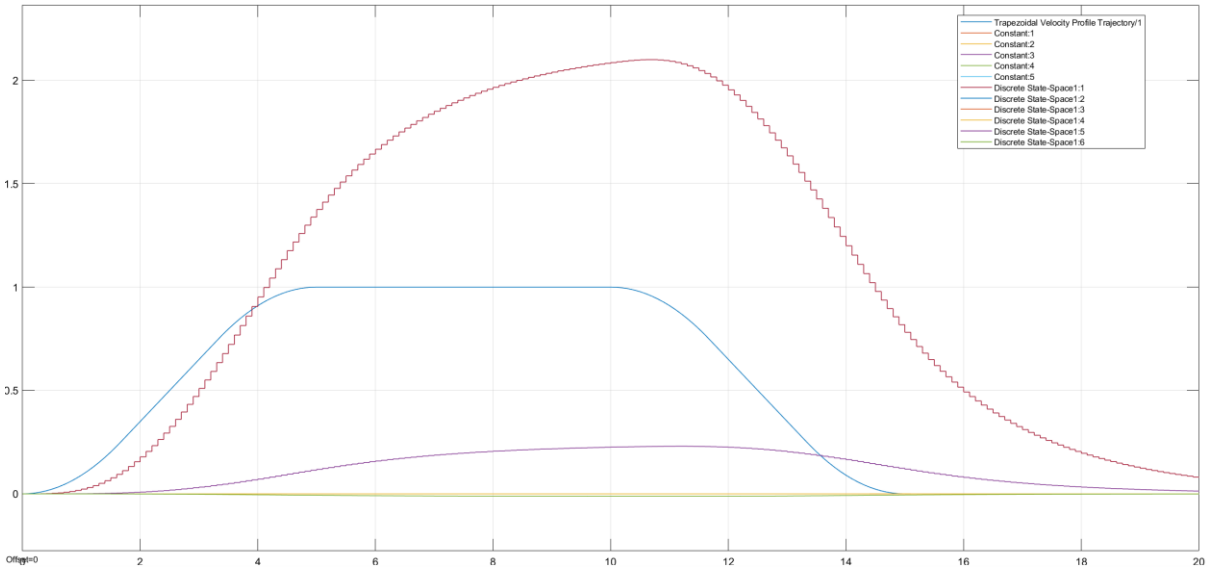

Figure 27 - Réponse (m/s-rad/s) du système en BO pour une consigne trapézoïdale

La figure suivante montre la consigne (courbe en bleu) ainsi que la réponse pour le système corrigé.

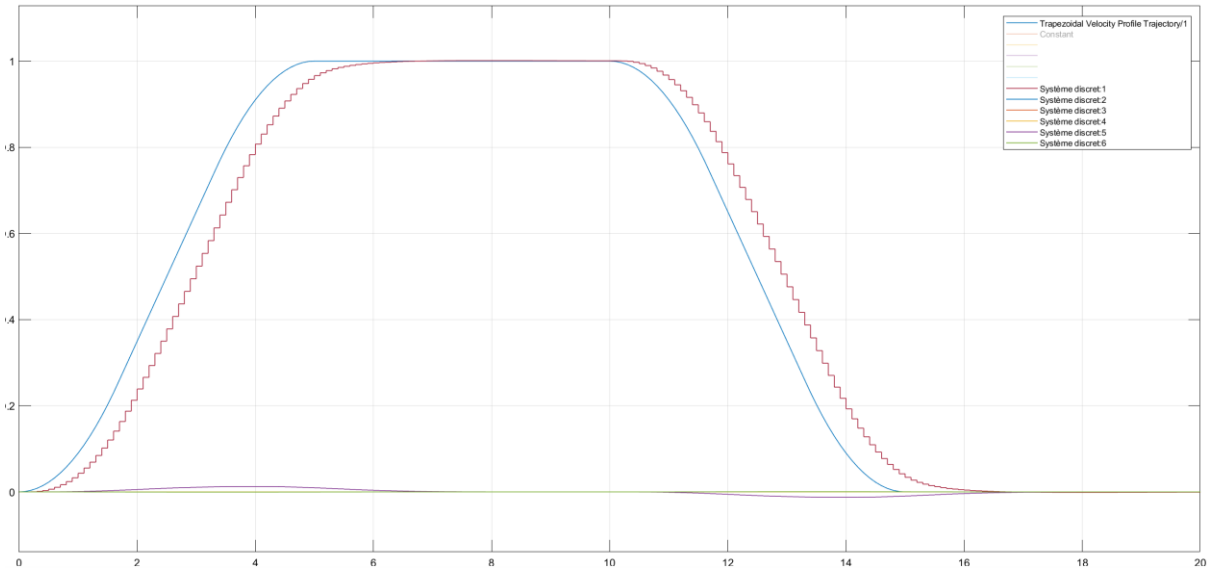

Figure 28 – Réponse (m/s-rad/s) du système en BF avec PID pour une consigne trapézoïdale

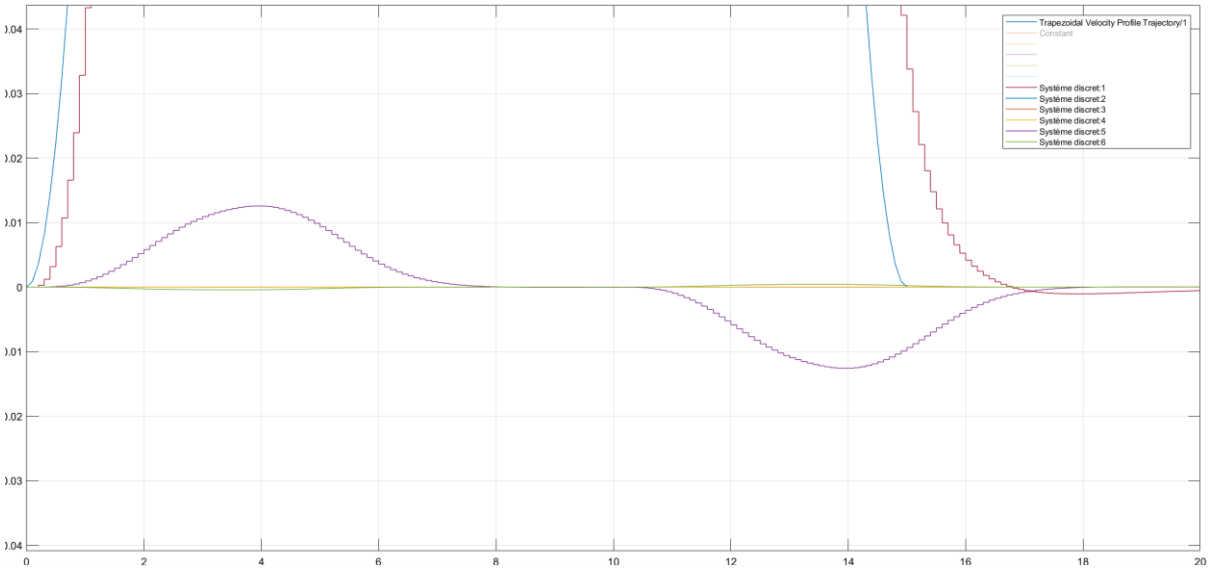

Figure 29 – Réponse (m/s-rad/s) du système en BF avec PID pour une consigne trapézoïdale (zoom)

La correction fonctionne également bien dans cette situation comme le montre la différence entre le modèle corrigé en boucle fermée et en boucle ouverte. Le système en boucle fermée suit la consigne avec un retard d'environ 0.5 seconde pendant l'accélération et la décélération sans dépassement. On observe toujours un léger effet de couplage correspondant à une rotation selon l'axe y pendant l'accélération et la décélération, mais cela reste acceptable puisqu'aux alentours de 1% de la valeur de consigne selon l'axe x.

3. Conception d'un contrôleur position avec suivi de trajectoire

Jusqu'à maintenant, nous n'avons que présenté des contrôleurs vitesse. Or pour le club Sonia, il serait intéressant d'être en mesure de contrôler le sous-marin en positions afin de pouvoir effectuer des tâches d'alignements.

À la session d'hiver 2021, Alexandre Desgagné et Alexandre Lamarre ont déjà exploré la commande MPC dans le cadre de leur projet spécial (GPA791). Cependant, par contrainte de temps, car le projet était plus focalisé sur la modélisation du système en boucle ouverte, ils ont utilisé la version MPC non linéaire. Cette version proposée par Matlab est « magique », car nous n'avons qu'à lui fournir le modèle non linéaire continue et le bloc s'occupe de faire la conception pour nous. Ce bloc calcule même la jacobienne de façon numérique. Cela a permis de faire des tests rapidement sur Simulink. Ce projet a été effectué durant la pandémie où il était très difficile d'accéder aux locaux des clubs étudiants et que toutes les piscines intérieures étaient fermées. Il a donc fallu attendre au mois de juillet avant d'effectuer nos premiers tests sur le sous-marin. Nous avons remarqué que même si le mpc non linéaire est compilable, il demande une quantité de ressource processeur bien trop importante pour rouler en temps réel sur notre ordinateur embarqué.

Dans la prochaine section, nous aborderons l'implémentation de contrôleur MPC (Adaptive et LTV) basée sur un problème d'optimisation quadratique convexe. Même s'il s'agit d'un problème d'optimisation non linéaire, il est facile de la résoudre, car la fonction coût ne possède pas de minimum local, ce qui indique qu'il y existe au plus 1 solution optimale. La figure suivante montre comment une fonction convexe est plus simple à résoudre qu'une fonction non convexe.

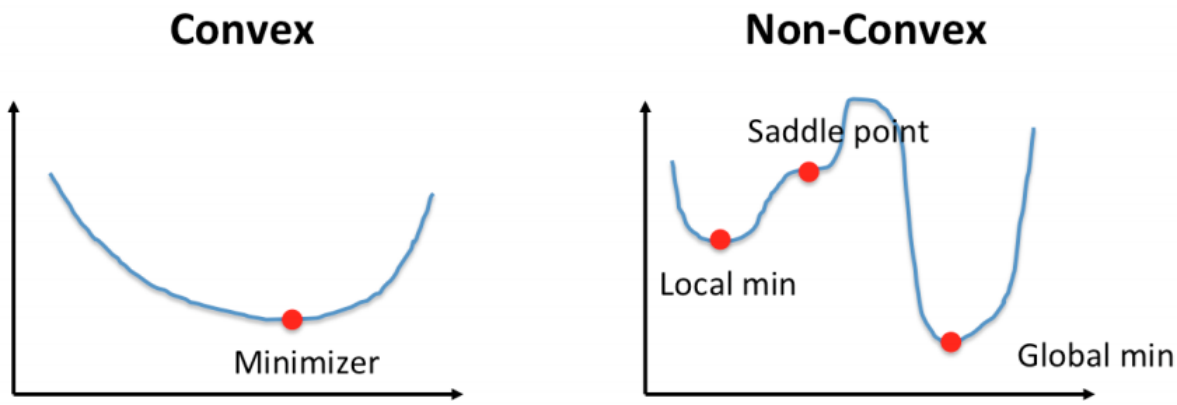

Figure 30 -Comparaison entre fonction convexe et non convexe [8]

Pour ces raisons, ces contrôleurs demandent beaucoup moins de ressources processeur et peuvent être exécuté en temps réel sur notre ordinateur embarqué. De plus nous savons que notre modèle ne comporte pas de haute non-linéarité, nous devrions obtenir des performances similaires.

3.1. Critère de performance

Pour le contrôleur position, nous allons envoyer une trajectoire polynomiale du 3^{e} ordre afin d'asservir le régime transitoire du sous-marin. Le but est de s'assurer que les vitesses et les accélérations demandées par la trajectoire soient réalisable pour le sous-marin. Dans notre cas, le temps de réponse n'a pas besoin n'a pas besoin d'être le plus rapide possible. Nous voulons que le sous-marin respecte au maximum la trajectoire demandée. Dépendamment de l'application, nous voulons que le sous-marin aille plus lentement. Par exemple lorsqu'on fonce dans un obstacle, nous voulons y aller doucement.

De plus, nous désirons avoir un dépassement qui se rapproche au maximum de 0%. Nous savons que nous sommes dans l'eau où les perturbations sont grandes. Ce qui rend un dépassement nul en pratique presque impossible.

Pour terminer, nous aimerions avoir une erreur presque nulle en régime permanent ainsi qu'une erreur constante au régime transitoire.

3.2. Définition d'un contrôleur MPC.

MPC pour « Model Predictive Control » ou (RHC) pour « Receding horizon control » est une technique qui fait partie de la famille des commandes optimales. Elle se base sur un modèle dynamique pour prédire et anticiper le comportement d'un sous-marin. Cette commande est principalement utilisée sur des systèmes complexes MIMO et elle a prouvé de bonnes performances lorsque le dispositif est soumis à de nombreuses perturbations [1]. Nous avons été intéressés par cette commande, car le sous-marin opère dans des conditions où il y a beaucoup de perturbations (vague, courant, etc).

L'algorithme de la commande s'effectue à chaque échantillonnage et il peut être vulgarisé selon les 3 étapes suivantes.

1. Fournir la référence désirée à chaque échantillon sur une période définie soit de k à k+p.
2. Prédire les états du système en boucle fermée à chaque échantillon sur une période définit soit de k à k+p.
3. Calculer une commande pour chaque échantillon sur une période définit soit de k à k+m.
4. Retourner uniquement la commande de k+1.

Voici une figure qui illustre le fonctionnement d'un MPC

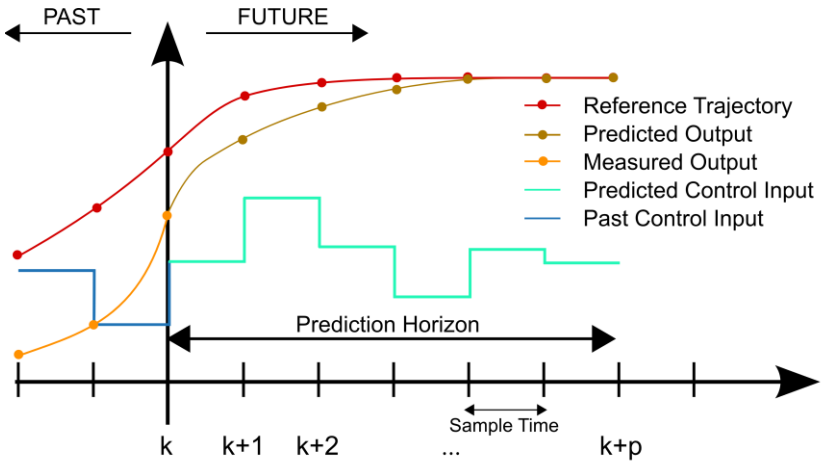

Figure 31-Schéma de la commande MPC [9]

Le contrôleur MPC vient avec une collection de gains et de paramètres que nous pouvons ajuster pour avoir une bonne réponse. Nous allons commencer par les paramètres statiques du contrôleur, c'est-à-dire ceux qui ne peuvent pas être changés en temps réel.

- Ts : Le temps d'échantillonnage
- P : le nombre d'échantillons utilisés pour la prédiction.
- M : le nombre d'échantillons pour lesquels le contrôleur prédira une commande.

Puisque la commande est non linéaire, il est possible de changer la majorité des gains en temps réel. Voici ceux qu'on a utilisés dans notre projet.

- Output Variable (OV) : Il s'agit d'un vecteur de la même taille que le vecteur d'état (x) dans lequel la valeur indique le niveau de performance de chaque état à asservir. Plus la valeur est grande, plus le contrôleur va donner de l'importance à l'état en question dans sa commande.
- Manipulated variable (MV) : Le contrôleur MPC permet de donner une référence désirée pour les entrées u. cela est pratique lorsque le système est à l'équilibre avec un u non nul. Le vecteur MV de taille identique au vecteur d'entrée U est le critère de performance des entré. Cela permet aux entrées de s'éloigner temporairement de la commande désirée afin d'améliorer la réponse globale du système en boucle fermée. Plus le chiffre est grand plus l'entrée sera en mesure de s'éloigner.
- Manipulates variable rate (MVR) : Ce vecteur de taille identique au vecteur d'entrée U permet de définir l'agressivité de la commande. Plus le chiffre est petit, plus la commande qui sera envoyée à l'entrée pourra varier rapidement.
- Umin, Umax : Deux vecteurs de taille U qui déterminent la commande maximale et minimale de chaque entrée.

3.3. Fonctionnement d'un contrôleur MPC

Comme pour la commande LQR, la commande MPC minimise un critère de performance quadratique (J).

Dans le cadre du cours, nous avons vu la commande optimale avec un horizon infini. Cette commande a pour but de trouver les gains optimaux pour l'ensemble du système en boucle fermé. On dit donc une optimisation globale. Nous pouvons remarquer cela, car l'intégrale de la fonction coût est borné de zéro à l'infini. Voici un exemple de fonction coût tiré du livre *Commande Moderne : Approche par modèles continus et discrets*, Bensoussan, (2008).

$$J = \int_0^\infty [y^T(t)\, Qy(t) + u^T(t)\, R\, u(t)]dt$$

Étant donné qu'il s'agit d'une optimisation globale, nous pouvons calculer les gains à l'avance et ceux-ci resteront constants sur tout l'intervalle de temps. Le système en boucle fermé reste donc linéaire.

En revanche le contrôleur MPC utilise un horizon finit variable qui est son horizon de prédiction. Cela veut dire que l'horizon de la fonction coût change à chaque itération et que l'intégrale sera bornée avec des valeurs non infinies. La fenêtre de prédiction est de l'instant k à k+p. Étant donné qu'on optimise sur une période définie, nous trouvons une solution optimale locale à la zone de prédiction. Il faut donc calculer une nouvelle solution en temps réel à chaque itération. De plus, vu que l'optimisation est effectuée dans le domaine discret, l'intégrateur est remplacé par un sommateur. Voici un exemple de fonction coût pour le MPC présenté par Wikipédia auquel nous avons utilisé la notation de Matlab.

$$J = \sum_{j=1}^{n_y} \left(\sum_{i=1}^{p} Q_j \left[y_j(k+i) - r_j(k+i) \right]^2 \right) + \sum_{j=1}^{n_u} \left(\sum_{i=1}^{m} R_j \left[u_j(k+i) - u_j(t+i-1) \right]^2 \right)$$

Où :

n_y : nombre variable de sortie.

n_u : nombre variable d'entré.

k : l'itération présente.

p : nombre d'intervalles de la plage de prédiction de variable de sortie.

m : nombre d'intervalles de la plage de prédiction de variable d'entrée.

Q_j : Poids de pénalité de performance pour chaque variable de sortie j (OV).

$r_j(k)$: Référence de la sortie j pour chaque instant k.

$y_j(k)$: valeur prédite de la sortie j pour chaque instant k.

R_j : Poids de pénalité de variation de la commande (énergie) (MVR).

$u_j(k)$: Commande de l'entré J au temps k.

Ce problème d'optimisation peut être résolu en temps réel avec un algorithme de programmation quadratique. De plus, vu que J ne sera minimisé de façon analytique, il est possible de rajouter des contraintes linéaires à notre problème d'optimisation sans que celui-ci devient non-convexe. Dans notre cas, nous avons rajouté une contrainte afin de saturer l'effort de nos moteurs de sorte que :

$$\forall_j \in n_u : \; U_{min} \leq \; U_j \leq U_{max}$$

Dans notre cas, les blocs MPC de Simulink crée les matrices de programmation quadratique en fonction du système en boucle ouverte et utiles l'algorithme « active-set » pour minimiser J. Il est possible de définir une fonction coût J personnalisé. Cependant, dans la version 2021a, seule la fonction coût par défaut est supportée pour la génération de code C++. Nous n'avons donc pas le choix du l'utiliser. Voici la fonction coût que nous allons utiliser dans ce projet.

$$J = J_y + \; J_u + J_{\Delta u} + J_\varepsilon$$

Où :

J_y : est la fonction quadratique qui pénalise l'erreur des variables de sortie en fonction des poids OV.

$$J_y = \sum_{j=1}^{n_y} \left(\sum_{i=1}^{p} OV_j \left[y_j(k+i) - r_j(k+i) \right]^2 \right)$$

J_u : est la fonction quadratique qui pénalise l'erreur de la commande selon les poids MV

$$J_u \; = \sum_{j=1}^{n_u} \left(\sum_{i=1}^{p-1} MV_j \left[u_j(k+i) - u_{j,désiré}(t+i) \right]^2 \right)$$

$J_{\Delta u}$: est la fonction quadratique qui pénalise l'énergie consommée par le système en boucle fermé selon les poids MVR :

$$J_{\Delta u} \; = \sum_{j=1}^{n_u} \left(\sum_{i=1}^{p-1} MVR_j \left[u_j(k+i) - u_j(t+i-1) \right]^2 \right)$$

J_ε : Pénalise les contraintes souples du problème d'optimisation, Or on n'utilise pas genre de contrainte dans notre projet. Nous pouvons la négliger.

Si vous regardez la documentation de Matlab, vous pouvez remarquer que nous négligeons le facteur d'échelle S_j. Nous le négligeons, car nos variables d'états ont des unités physiques (m, m/s, rad/sec) et que le facteur est donc de 1.

3.4. Avantages et inconvénients de la méthode MPC

Plusieurs avantages, techniques ou non, ont guidé notre choix vers ce type contrôleur. Nous allons prendre le temps dans cette section d'identifier ces différents points.

Premièrement, même si la méthode n'est pas simple à implémenter, elle est simple et intuitive à maintenir. Comme nous l'avons dit dans la section des contraintes, ce point est très important pour nous, car le taux de roulement dans les clubs est très grand. Il en résulte que tous ceux qui travaillent sur le projet actuellement ne seront plus membres du club d'ici la fin de l'année. Il est important que les nouveaux membres en début de baccalauréat soient en mesure de maintenir le contrôleur au fur à mesure des modifications du modèle (ajout ou modification de modules). Heureusement les gains sont intuitifs et ne demandent pas une grande connaissance en théorie de l'asservissement pour les comprendre.

Deuxièmement, nous avons mentionné plus tôt que notre sous-marin est « sur actionné », car nous avons plus de moteurs que de degrés de liberté. Cela crée un problème d'optimisation pour distribuer la commande sur les huit moteurs. Étant donné que le MPC est une commande optimale, nous pouvons intégrer ce problème d'optimisation à même le contrôleur. De plus, en utilisant une commande optimale, nous cherchons à réduire l'énergie des actionneurs afin d'augmenter l'autonomie de nos batteries.

Un troisième avantage comparé au LQR est qu'avec un MPC, il est possible d'appliquer des contraintes au problème d'optimisation. Ce qui s'avère pratique pour saturer nos moteurs. De plus le MPC est plus performant que LQR loin du point d'équilibre.

Dernièrement, la commande MPC est bien implémentée dans Matlab et est majoritairement compatible avec la génération de code C++. Ceci facilitera l'intégration de notre loi de commande sur notre ordinateur embarqué.

Cependant un désavantage de la commande est MPC est que la performance du contrôleur est directement reliée à la précision du modèle dynamique interne. Or, comme nous l'avons dit plus tôt, notre modèle n'est pas encore très précis, car les paramètres hydrodynamiques sont très difficiles à déterminer.

Un autre désavantage de la commande MPC lorsque nous utilisons un algorithme de programmation quadratique est que nous ne pouvons pas assurer que la solution va converger.

3.5. Adaptive MPC

Originellement, la commande MPC est une commande linéaire. Cependant, il existe plusieurs variantes offertes pour nous aider à travailler avec des non-linéarités. Puisque notre modèle est non linéaire, nous allons tirer avantage de ces variantes. L'une de ses solutions « adaptive mpc » est une variante qui, à chaque échantillon, linéarise le système en boucle ouverte autour des conditions actuelles. Or pour appliquer cette méthode, il faut que le modèle ne contienne pas de hautes non-linéarités afin que l'approximation linéaire soit représentative durant la période de prédiction. Les matrices d'états vont donc varier avec le temps. On dit que le système en boucle ouverte est LTV (Linear Time-Varying).

De plus, Matlab a uniquement implémenté le contrôleur dans le domaine discret. Vu que nous voulons générer du code C++ ultérieurement, il est important de ne pas utiliser des fonction Matlab ne supportant pas la génération de code. Les fonctions du type (ss, c2d, quatrotate, lqr, place, etc) ne pourront pas être utilisées.

C'est pour cette raison que nous avons pris le temps de bien définir notre jacobienne analytique et notre discrétisation, car elles vont nous être très utiles.

Pour commencer, il faut créer un objet MPC afin de définir les paramètres statiques du modèle (nombre d'entrées, nombre de sorties, temps d'échantillonnage). Nous devons aussi fournir les conditions initiales ainsi que le système en boucle ouverte linéarisé à ce point.

Nous savons que nous allons toujours activer le contrôleur lorsque le sous-marin est au repos. Nous allons donc choisir le point d'équilibre comme point initial.

$$x_0 = x_{ss} = [n_{ss}^T \; v_{ss}^T]^T = [0\ 0\ 0.2243\ 0.9999\ -0.0118\ -0.0039\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

Étant donné que le sous-marin flotte au repos, le sous-marin doit combattre la poussé d'Archimède pour rester à l'équilibre dès qu'il n'est pas à la surface de l'eau. Cela résulte que la commande u convergera vers une valeur non nulle. Nous pouvons résoudre la commande pour maintenir le sous-marin à l'équilibre de la façon suivante :

$$solve\big(g_b(n_{ss}) - \mathrm{B\ u} = 0\big)$$

Par la suite, nous pouvons définir comme u désiré de sorte que

$$u_{désiré} = u_{ss} = [0\ 0\ 0\ 0\ 3.45 - 3.25\ \ 3.17 - 3.30]^T$$

De plus, nous allons utiliser un temps d'échantillonnage de 10hz comme nous l'avons dit précédemment. Notre système en boucle ouverte discrétisé aux conditions d'équilibre est le suivant :

$$A_d = \begin{bmatrix} 1.0 & 0 & 0 & 0 & -4.6e-6 & 0.002 & 0 & 0.0977 & -6.25e-8 & 7.26e-7 & -1.76e-6 & 1.43e-4 & -2.29e-5 \\ 0 & 1.0 & 0 & 0 & -0.00227 & 3.5e-6 & 0 & -4.67e-8 & 0.0977 & 2.69e-6 & -3.33e-4 & 9.84e-7 & 9.44e-6 \\ 0 & 0 & 1.0 & 0 & 1.36e-4 & -2.74e-5 & 0 & 4.7e-7 & 2.32e-6 & 0.0975 & 3.31e-5 & -4.79e-6 & -9.77e-8 \\ 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.948 & 2.24e-4 & 0 & -3.78e-6 & -8.89e-4 & 1.08e-4 & 0.0373 & 4.07e-5 & 3.75e-5 \\ 0 & 0 & 0 & 0 & 2.32e-4 & 0.979 & 0 & 3.53e-4 & 3.96e-6 & -2.16e-5 & 5.66e-5 & 0.0464 & -2.15e-9 \\ 0 & 0 & 0 & 0 & 8.5e-4 & 0.00149 & 1.0 & -3.42e-5 & 1.88e-5 & -2.19e-7 & 3.8e-5 & 2.46e-5 & 0.0439 \\ 0 & 0 & 0 & 0 & -7.46e-5 & 0.0392 & 0 & 0.955 & -9.56e-7 & 1.38e-5 & -3.12e-5 & 0.00309 & -4.34e-4 \\ 0 & 0 & 0 & 0 & -0.0424 & 5.66e-5 & 0 & -7.11e-7 & 0.955 & 4.73e-5 & -0.00636 & 1.76e-5 & 1.77e-4 \\ 0 & 0 & 0 & 0 & 0.00243 & -5.28e-4 & 0 & 8.98e-6 & 4.11e-5 & 0.951 & 6.18e-4 & -9.7e-5 & -1.65e-6 \\ 0 & 0 & 0 & 0 & -1.88 & 0.00784 & 0 & -1.31e-4 & -0.0318 & 0.00385 & 0.523 & 0.00151 & 0.00128 \\ 0 & 0 & 0 & 0 & 0.00813 & -0.801 & 0 & 0.0136 & 1.37e-4 & -8.33e-4 & 0.00207 & 0.853 & -5.42e-7 \\ 0 & 0 & 0 & 0 & 0.0318 & 0.057 & 0 & -0.0013 & 7.01e-4 & -7.02e-6 & 0.00158 & 0.00143 & 0.767 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00207 & 0.00207 & 0.00213 & 0.00213 & -1.62e-4 & -1.62e-4 & 1.65e-4 & 1.59e-4 \\ -0.00152 & 0.00154 & -0.00154 & 0.00152 & -2.45e-4 & 2.43e-4 & 2.56e-4 & -2.53e-4 \\ -4.92e-6 & 7.09e-6 & -4.71e-6 & 7.31e-6 & -0.00205 & 0.00207 & -0.00212 & 0.00211 \\ 0.00471 & -0.00458 & 0.00455 & -0.00475 & -0.02 & 0.0198 & 0.0208 & -0.0206 \\ 0.00179 & 0.00183 & 0.00179 & 0.00183 & 0.0098 & 0.00981 & -0.00998 & -0.00963 \\ -0.0151 & -0.0152 & 0.0155 & 0.0154 & 3.93e-5 & -2.9e-5 & -4.09e-5 & 3.05e-5 \end{bmatrix}$$

$$C_d = \mathbf{I}(13)$$

$$D_d = \mathbf{0}_{13x8}$$

Nous pouvons implémenter le contrôleur sous Simulink. Ce qui donne :

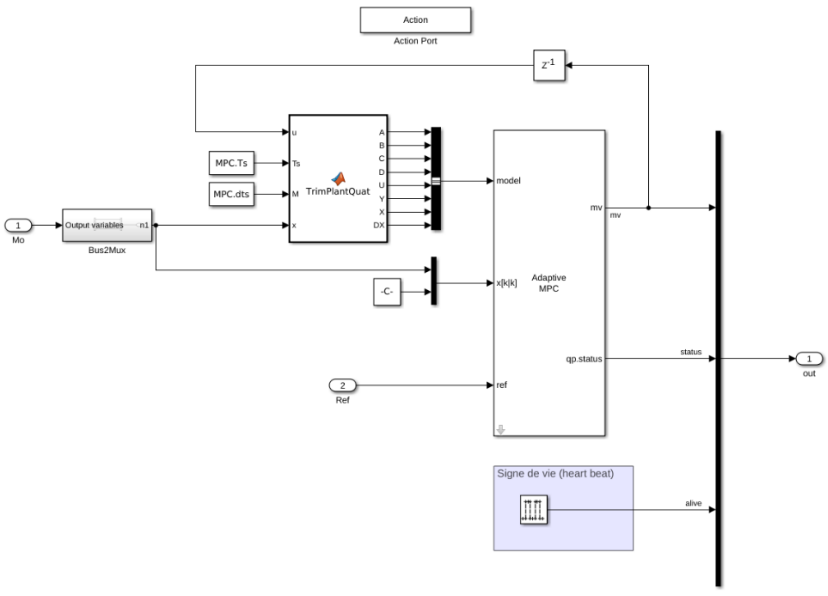

Figure 32 - Schéma du système avec MPC Adaptive

La fonction « TrimPlantQuat » s'occupe de linéariser et de discrétiser le système en temps réel. Elle retourne donc une nouvelle matrice A à chaque itération. Les matrices B, C et D sont linéaires, et donc restent constantes.

Par la suite X, Y et U sont les points d'opération des matrices linéarisées. Pour terminer, Dx est la variation d'états entre l'instant k et k+1.

$$D_x = f(x_k, u_k) - x_k$$

Sachant que notre modèle non linéaire $f(x_k, u_k)$ est continu, nous allons utiliser la méthode d'Euler Explicite « Foward Euler » comme technique d'intégration numérique afin d'avoir les états à l'échantillon k+1. Afin d'avoir une estimation plus précise de $f(x_k, u_k)$, nous allons itérer le calcul de manière à ce qu'il soit dix fois plus rapide que le temps d'échantillonnage du contrôleur, soit 100hz. Le paramètre dts permet de changer ce ratio.

Voici le code Matlab de la fonction « TrimPlantQuat »

```
function [A,B,C,D,U,Y,X,DX] = TrimPlantQuat(u,Ts,M,x)


% Lineariser le système
    [Ac, Bc, C, D]=AUVQuatJacobianMatrix(x,u);


% Discrétiser le système.
    A = expm(Ac*Ts); % Fossen Eq B.10/B.9 page 662

    BT = Ac(8:13,8:13)\(A(8:13,8:13)-eye(6))*Bc(8:13,1:8); % Fossen Eq B.11 p 662
    B = [zeros(7,8); BT];

% Calculer F(x(k),u(k))
    xk = x;

    % Intégration avec Euler Explicite (foward euler)
    for i=1:M
      xk = xk + AUVQuatSimFcn(xk,u)*(Ts/M);
       % Normaliser le Quaternion
          d=norm(xk(4:7));
          xk(4:7) = xk(4:7)./d;
    end


% Conditions Nominal du système discret
    U=u.';
    Y= x.';% (x==y);
    X=x.';
    DX= (xk-x).' ;
End
```

3.6. Détermination des gains

Comme avec le LQR, Il n'y pas de « recette » pour déterminer les poids de la fonction coût. Nous avons utilisé les valeurs de départ recommandées par Matlab, et nous avons itéré avec le modèle de simulation afin de trouver nos gains. Voici les gains que nous avons déterminé :

$$OV = [70\ 60\ 60\ 90\ 90\ 90\ 90\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$MV = [0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2]$$

$$MVR = [0.1\ 0.1\ 0.1\ 0.1\ 0.3\ 0.3\ 0.3\ 0.3]$$

$$U_{max} = [40\ 40\ 40\ 40\ 40\ 40\ 40\ 40]\ \ U_{min} = -[30\ 30\ 30\ 30\ 30\ 30\ 30\ 30]$$

Ts :10hz, P :10, M : 2

Nous avons eu beaucoup de problèmes avec l'intégration de nos moteurs et de nos capteurs. Il a donc fallu attendre le début du mois de novembre avant de pouvoir tester nos lois de contrôle. Or nous savions que le vrai sous-marin n'allait pas se comporter de la même façon qu'en simulation à cause du manque de précision dans la détermination des constantes mécaniques.

Pour notre premier essai, nous avons décidé d'avancer de 10 mètres en 20 secondes, voici les réponses que nous avons observé. En orange nous avons la référence et bleu la réponse.

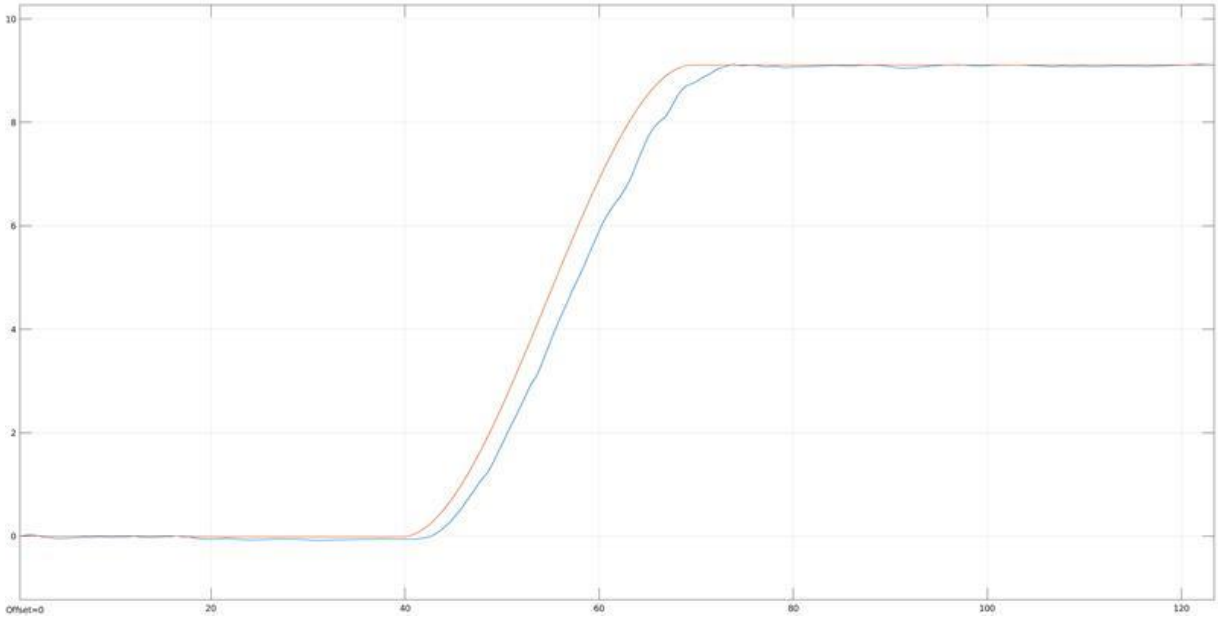

Figure 33-Réponse (m) du système en BF avec MPC selon x

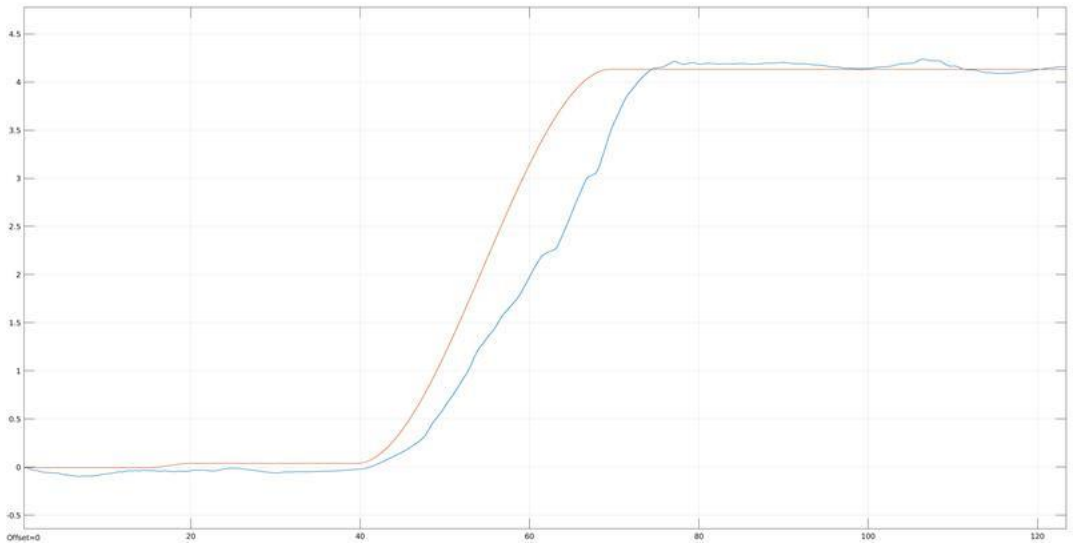

Figure 34-Réponse (m) du système en BF avec MPC selon y

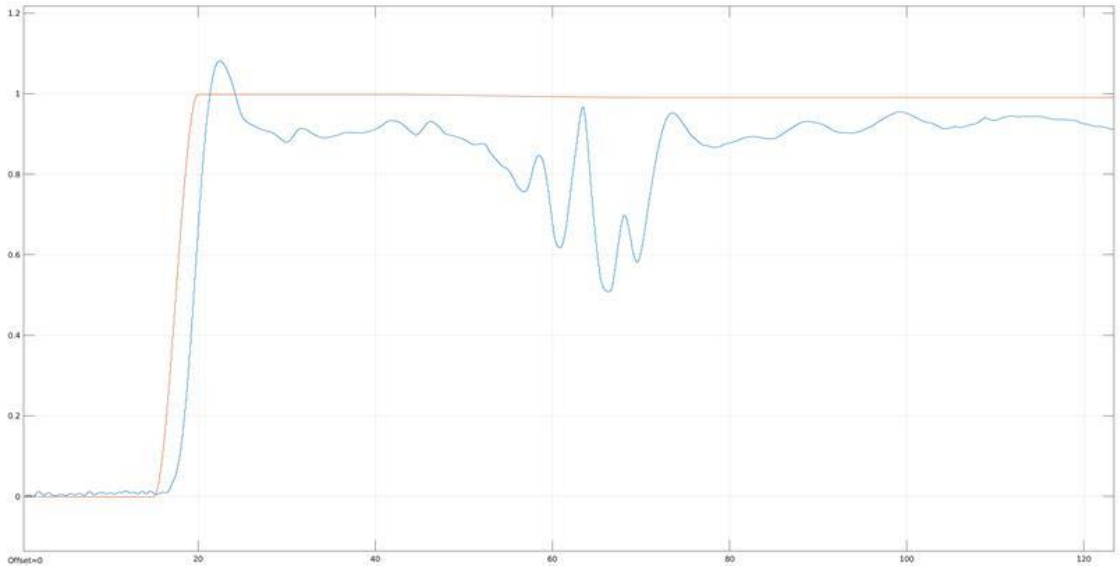

Figure 35-Réponse (m) du système en BF avec MPC selon z

Comme nous pouvons le constater que les réponses en x et y ont une erreur nulle en régime permanent, mais la réponse transitoire en y n'est pas constante et tente de diverger. De plus, la réponse en z fluctue beaucoup au cours du déplacement. Cela entraine un déplacement en dauphin.

Nous savons qu'il y a des effets de couplage non négligeables entre la vitesse en Z et la vitesse du « pitch » (rotation autour de x). En effet, le centre de masse et le centre de flottaison du sous-marin ne sont pas alignés, ce qui crée un moment de rotation autour de x lorsque le sous-marin avance et descend en z. Or dans nos OV, le poids de z est à 60 et le poids des angles est à 90. Le contrôleur priorise donc la performance en « pitch » au détriment de la position en z. Cela n'est pas une mauvaise chose, car l'on veut garder l'assiette du sous-marin la plus droite possible pour ne pas perdre les mesures de notre DVL comme nous l'expliquions plus tôt. Cependant, la différence entre les 2 poids est peut-être trop

grande. De plus, la différence entre la réponse en x et en y est déjà visible avec un delta de 10 dans nos OV. Enfin, lors des tests nous avons pu remarquer que le sous-marin oscille pendant le déplacement. Il serait donc intéressant d'augmenter les MVR pour améliorer le comportement du système en boucle fermée.

Pour ajuster nos gains de façon optimale, nous avons effectué 2 heures de tests à la piscine et sommes arrivés à la solution suivante :

$$OV = [30\ 30\ 30\ 45\ 45\ 45\ 45\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$MV = [0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2\ 0.2]$$

$$MVR = [0.4\ 0.4\ 0.4\ 0.4\ 0.5\ 0.5\ 0.5\ 0.5]$$

Ts :10hz, P :10, M : 2

Nous testons maintenant ces gains avec la même trajectoire que précédemment pour comparer nos résultats. Notons que les courbes x et y ne sont pas identiques, car les instructions dont données selon le référentiel objet soit « avance de 10m par rapport à ta position ». Or les positions linéaires et angulaires de départ du sous-marin étaient différentes. Cependant, le mouvement reste le même.

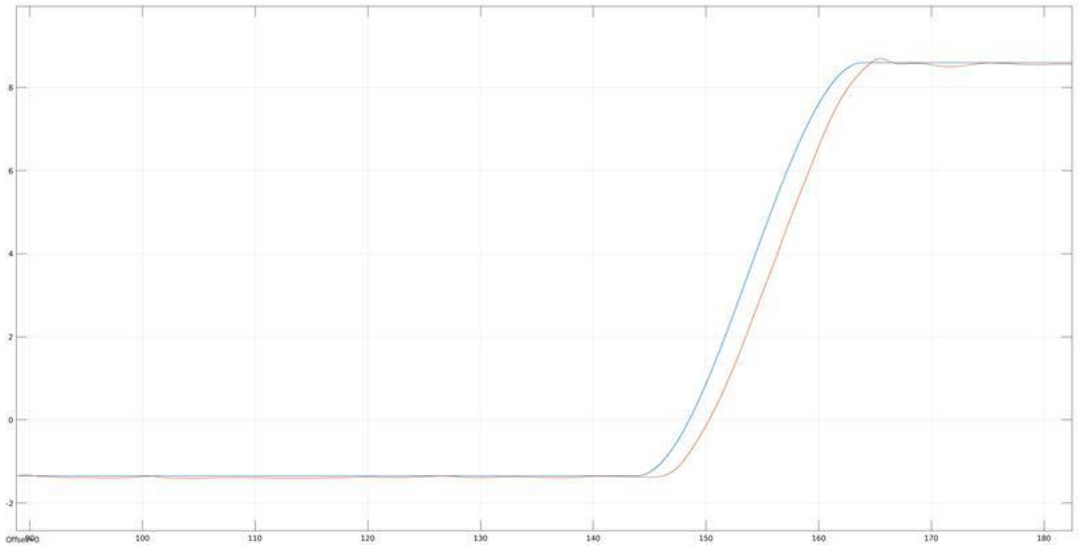

Figure 36-Réponse (m) du système en BF avec MPC réglé en x

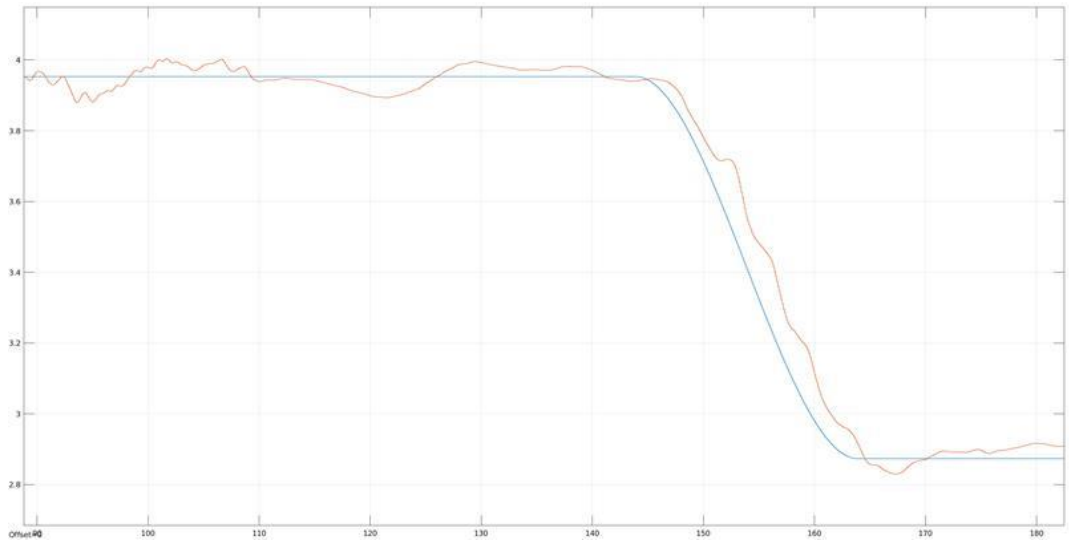

Figure 37-Réponse (m) du système en BF avec MPC réglé en y

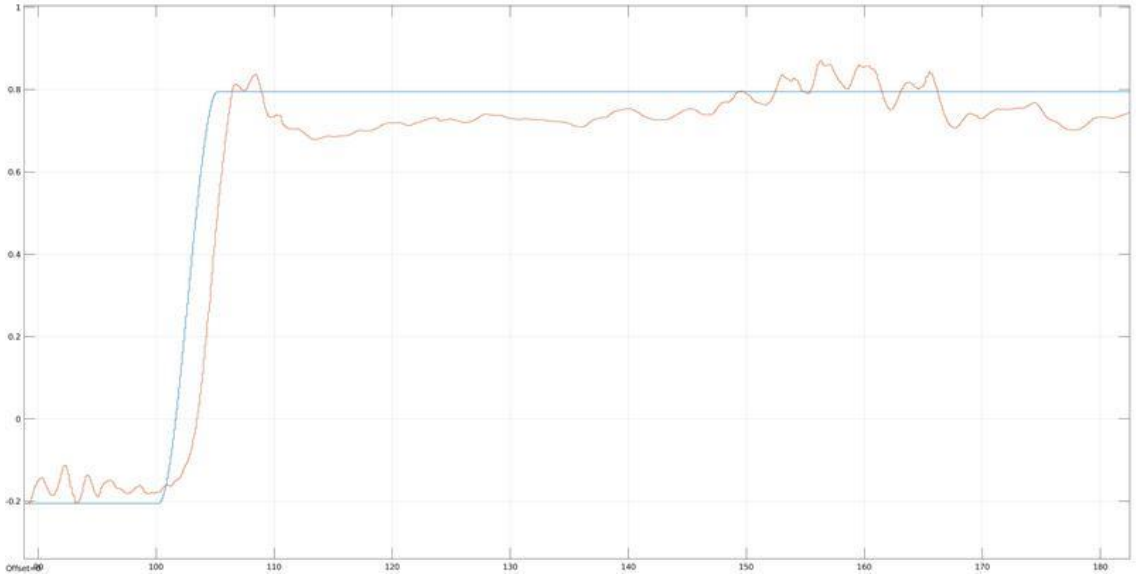

Figure 38-Réponse (m) du système en BF avec MPC réglé en z

Nous pouvons remarquer que nos réponses en x et y sont très similaire bien qu'elles n'en aient pas l'air. En effet, les échelles des graphiques sont différentes ce qui peut biaiser l'impression : présence d'un facteur d'échelle verticale de 5 sur la réponse en y par rapport à la réponse en x. Les réponses en x et y ont une erreur nulle en régime permanent et constante en régime transitoire. De plus, on peut remarquer que la réponse en z s'est grandement améliorée pour une perte de 1.5 degré en pitch (rotation autour de x). Enfin, le sous-marin est beaucoup plus stable en régime permanent. Malheureusement, l'enregistrement sur le sous-marin a été corrompu et les courbes angulaires de ces tests ont été perdues.

3.7. MPC LTV

Comme nous venons de le discuter, l'adaptive MPC linéarise le système en boucle ouverte à chaque itération et suppose que ce système est constant sur toute la plage de sa prédiction. Dans notre cas nous avons une prédiction de 10 échantillons avec un temps d'échantillonnage de 10hz pour une prédiction de 1 seconde dans le futur. Cependant, nous sommes capables de tourner de 180 degrés en « yaw » en 3 secondes et nous soupçonnons que la prédiction du système en boucle fermée peut s'avérer aberrante dans ce genre de cas où les non-linéarités varient rapidement.

Le LTV MPC fonctionne de la même façon que l'adaptive MPC à la seule différence qu'il n'assume pas que le modèle est constant sur toute la plage de sa prédiction. Le MPC LTV permet de faire varier le système en boucle ouverte dans son intervalle de prédiction. Pour ce faire, il linéarise un système en boucle ouverte pour chaque échantillon de la plage de prédiction. Dans notre cas, cela équivaut à 10 linéarisations par itération. Voici l'implémentation du contrôleur dans Simulink.

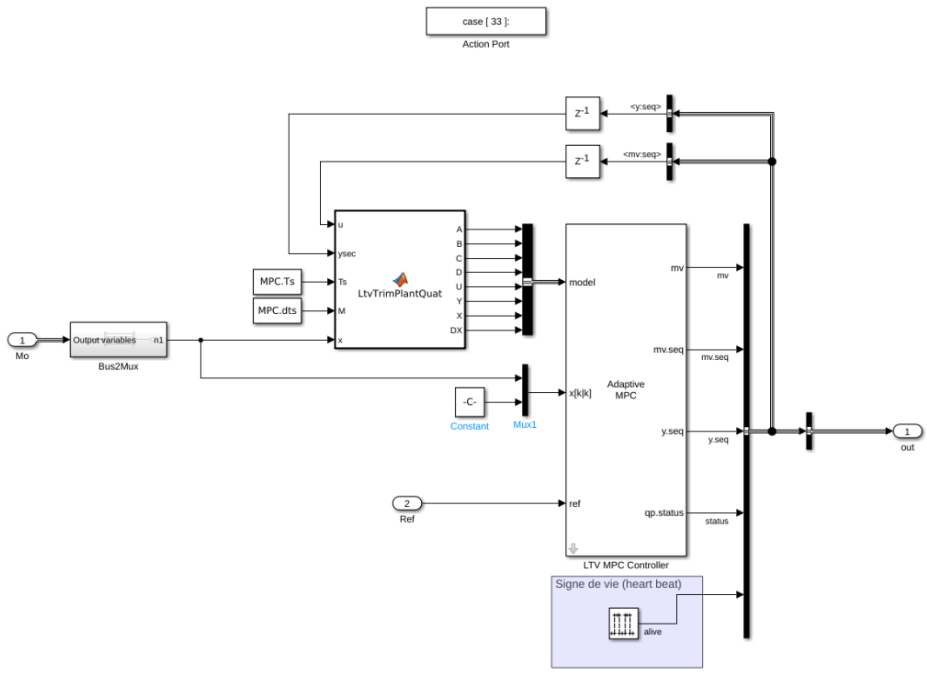

Figure 39-Schéma du contrôleur LTV MPC

Y.seq est une matrice 13X10 qui contient la prédiction de chaque état pour chaque échantillon de l'intervalle de prédiction. Mv.seq est une matrice de 8X2 qui contient les commandes prédites pour chaque échantillon de l'intervalle de commande. Maintenant, la fonction « LtvTrimPlantQuat » s'occupe de linéariser et de discrétiser en fonction de chaque prédiction. Elle retourne donc 10 nouvelles matrices A à chaque itération. Les matrices B, C et D sont linéaires, elles restent donc constantes.

Voici le code Matlab de la fonction « LtvTrimPlantQuat » :

```
function [A,B,C,D,U,Y,X,DX] = LtvTrimPlantQuat(u,ysec,Ts,M,x)


% Déterminer les dimentions
    nx = size(ysec,2);
    nu = size(u,2);
    p = size(u,1);

% Initialiser les matrices
    A = zeros(nx, nx, p);
    B = zeros(nx, nu, p);
    C = zeros(nx, nx, p);
    D = zeros(nx, nu, p);
    U = zeros(nu, 1, p);
    Y = zeros(13, 1, p);
    X = zeros(13, 1, p);
    DX = zeros(13, 1, p);

 xkk = x;

for i=1:p

    % Actualiser les états
        xk=xkk;
        uk=u(i,:).';

    % Lineariser le model dynamique
        [Ac, Bc, C(:,:,i), D(:,:,i)]=AUVQuatJacobianMatrix(xk,uk);

    % Discrétiser le système
        A(:,:,i) = expm(Ac*Ts); % Fossen(2021) Eq B.10/B.9 page 662


        BT = Ac(8:13,8:13)\(A(8:13,8:13)-eye(6))*Bc(8:13,1:8); % Fossen(2021) Eq B.11 p 662
        B(:,:,i) = [zeros(7,8); BT];

    % Calculer F(x(k),u(k))

        % Intégration avec Euler Explicite
        for j=1:M
          xkk = xkk + AUVQuatSimFcn(xkk,uk)*(Ts/M);
          % Normaliser le Quaternion
          d=norm(xkk(4:7));
          xkk(4:7) = xkk(4:7)./d;
        end


    % Condition nominal du système discret
        U(:,:,i)  = uk.';
        Y(:,:,i)  = xk.'; % (x==y);
        X(:,:,i)  = xk.';
        DX(:,:,i) = (xkk-xk).' ;

end

end
```

Nous avons effectué des tests sur notre modèle de simulation, car nous n'avons pas encore eu la chance de tester le MPC LTV en piscine. Pour tester la plus-value du LTV MPC par rapport à l'adaptative MPC, nous effectuons des trajectoires agressives qui feront changer la dynamique du modèle plus rapidement. Comme premier test nous effectuons une rotation du sous-marin de 90 degrés en 2 seconde. Afin de mieux interpréter les graphiques, nous avons converti la réponse en quaternion en angle d'Euler.

Les premières courbes montrent la réponse du contrôleur MPC adaptative.

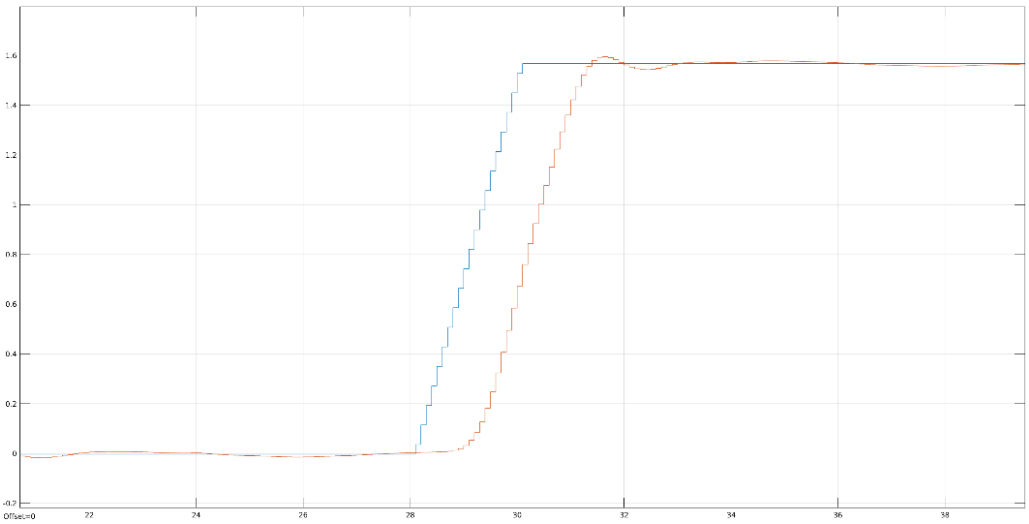

Figure 40-Réponse(rad) du système en BF avec Adaptative MPC en « yaw »

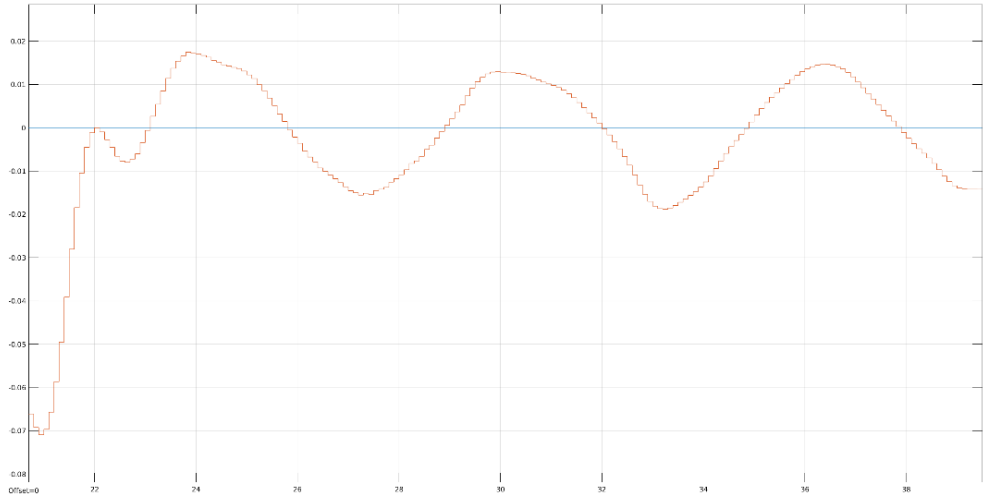

Figure 41-Réponse (rad) du système en BF avec Adaptative MPC en « pitch »

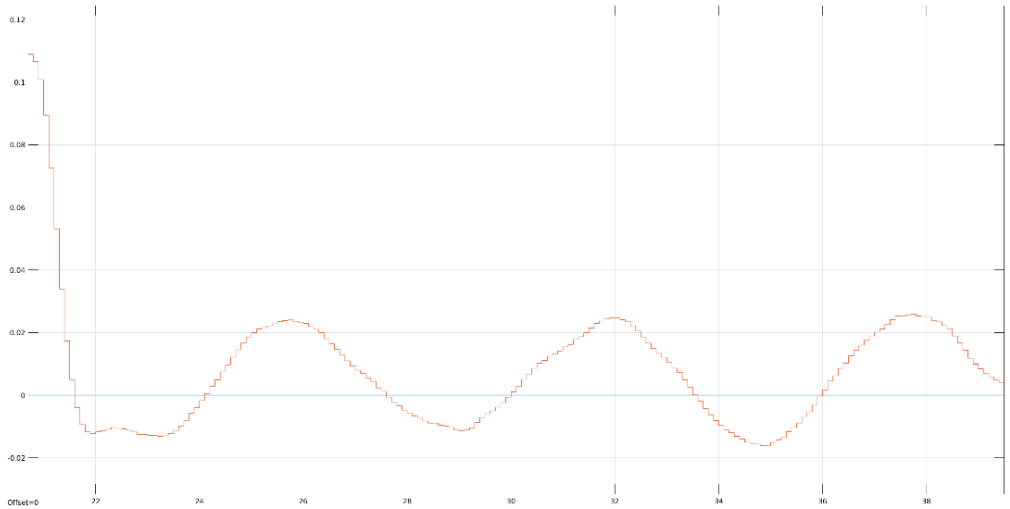

Figure 42-Réponse (rad) du système en BF avec Adaptative MPC en « roll »

Voici les courbes obtenues avec le LTV MPC :

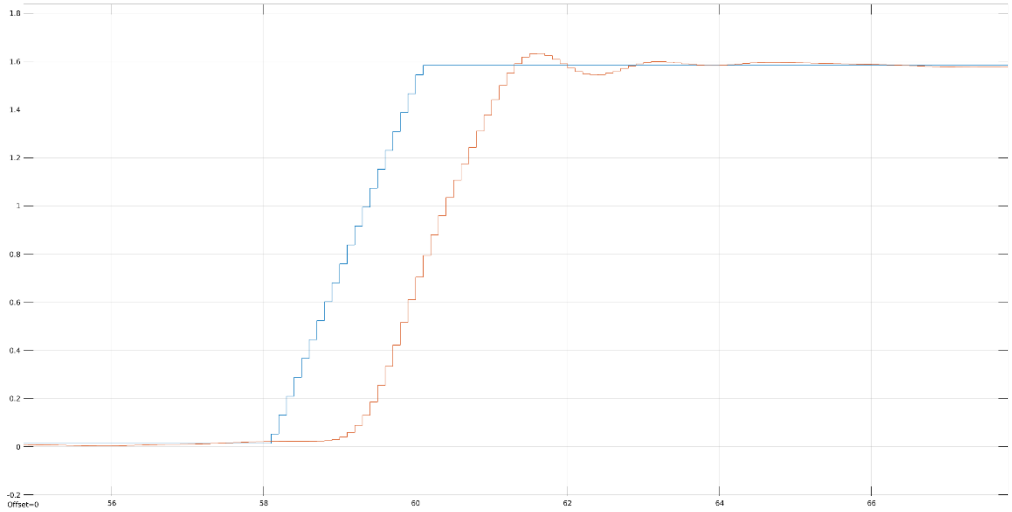

Figure 41-Réponse (rad) du système en BF avec LTV MPC en « yaw »

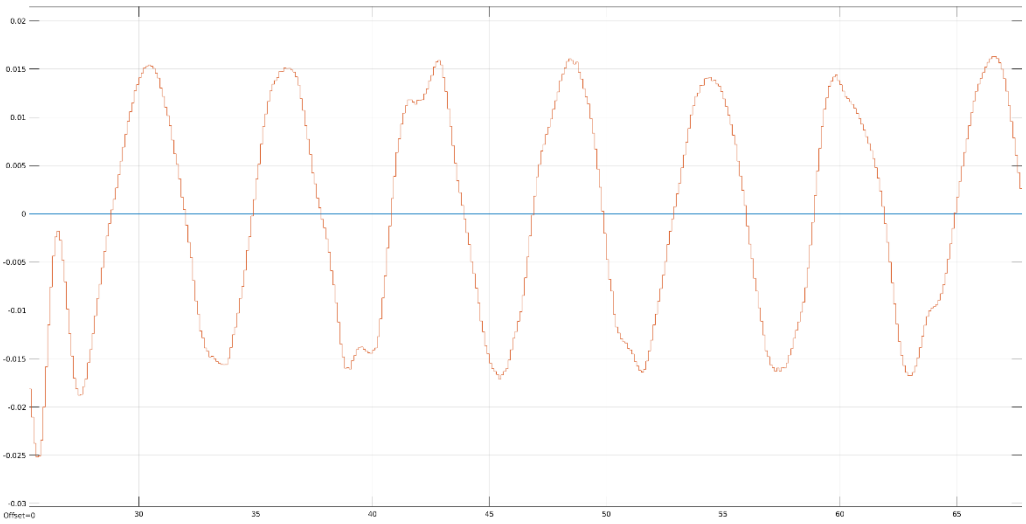

Figure 44-Réponse (rad) du système en BF avec LTV MPC en « pitch »

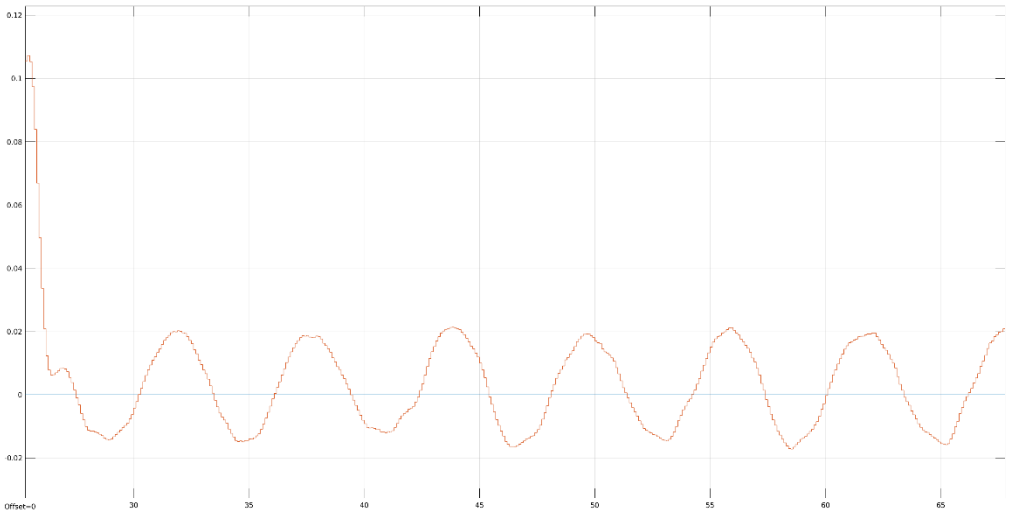

Figure 43-Réponse(rad) du système en BF avec LTV MPC en « roll »

Nous pouvons remarquer qu'il n'y a pas d'amélioration majeure de la réponse pour ce type de trajectoire. Nous pouvons remarquer que malgré les perturbations, le système oscille autour de la référence Comme deuxième essai, nous avons décidé de tester notre fonction « hard brake ». Cette fonction consiste à arrêter brusquement la trajectoire en cours lorsque la vision détecte un obstacle. Cela est pratique lorsque le sous-marin est mode « recherche ». Cette commande change la dynamique du modèle rapidement. Pour cet essai nous avons envoyé une commande de déplacement « avancer de 10 mètres en 20 secondes » et nous avons envoyé la commande « hard brake » autour du 7e mètre. Voici la réponse qu'a généré le MPC adaptive.

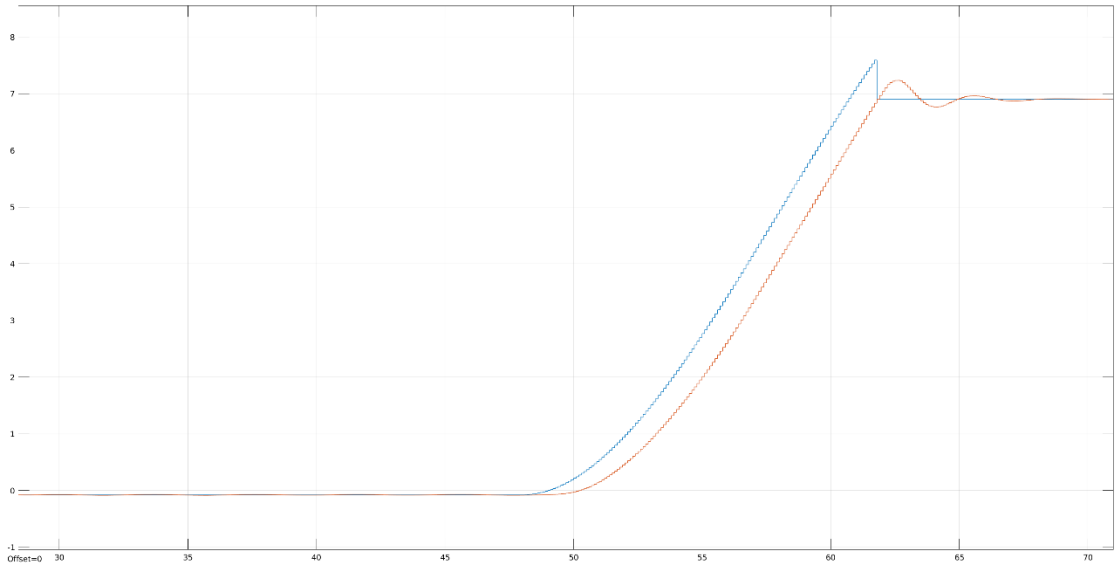

Figure 45—Réponse(m) du système en BF avec Adaptative MPC en x

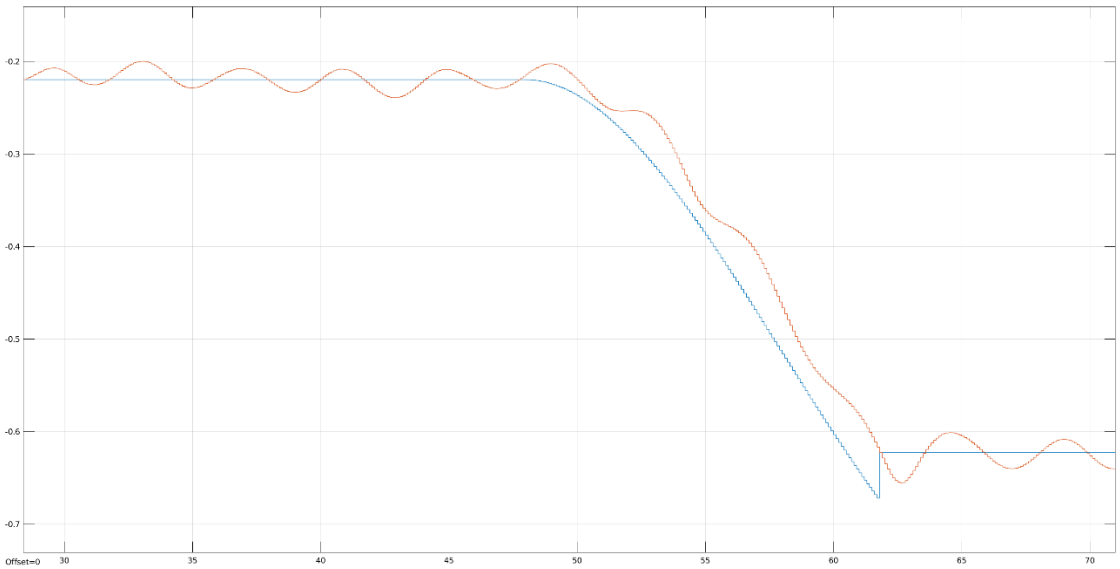

Figure 46-Réponse(m) du système en BF avec Adaptative MPC en y

Maintenant, voici la même trajectoire avec le MPC LTV

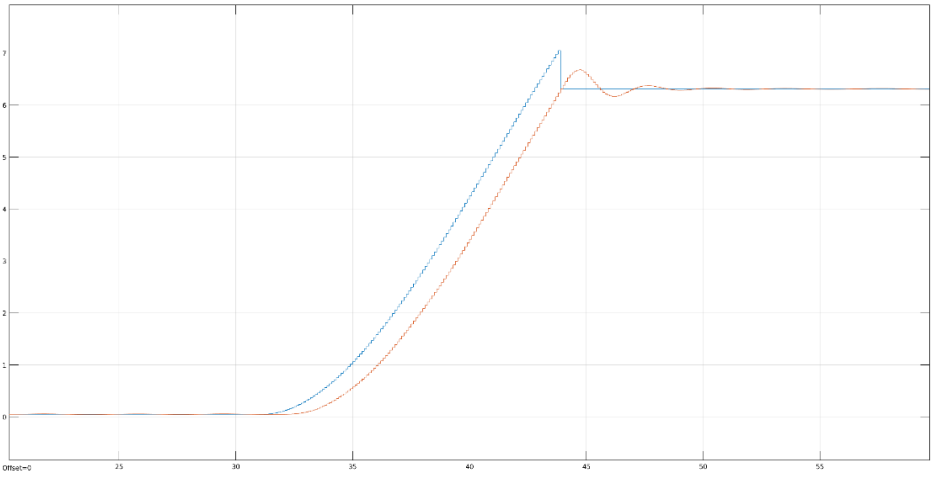

Figure 47-Réponse (m) du système en BF avec LTV MPC en x

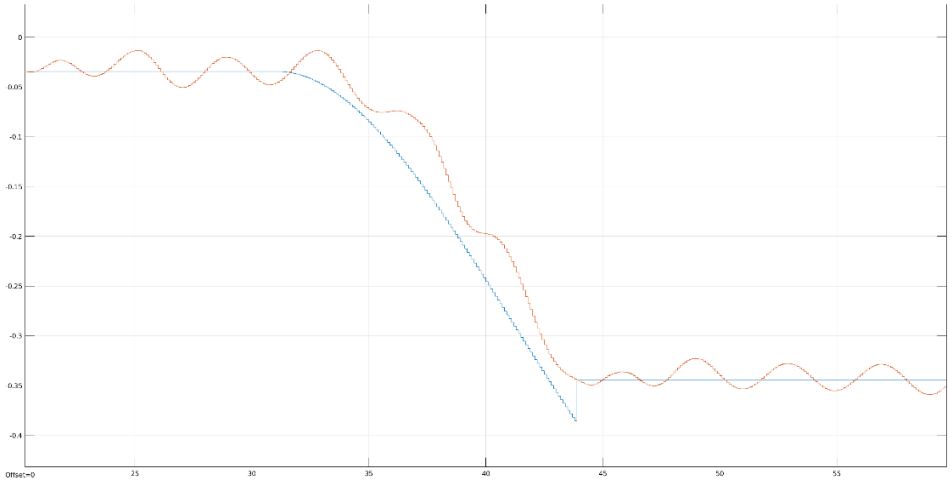

Figure 48-Réponse(m) du système en BF avec LTV MPC en y

Là encore, nous pouvons remarquer que la différence entre les 2 contrôleurs est très faible, bien que nous puissions voir une légère amélioration en y. Nous pouvons remarquer qu'un dépassement c'est introduit, ce qui normale, car les prédictions qu'ont été drastiquement changées et qu'on demande au sous-marin d'arrêter instantanément.

Pour donner suite à ces deux essais, nous pouvons conclure que les non-linéarités apparaissant dans la période de prédiction ont un impact négligeable sur la loi de commande. Ainsi, la commande LTV MPC n'améliore que très faiblement les performances du système en boucle fermée. De plus, le LTV MPC est plus gourmand en termes de ressources CPU de l'ordinateur embarqué que l'Adaptative MPC. Il est donc plutôt dans notre intérêt de conserver la commande Adaptative MPC. Néanmoins, avant de conclure quant à la performance du correcteur LTV MPC, il nous faudra refaire ces mêmes tests sur le sous-marin réel en janvier 2022.

4. Compétition Inter-Québec

Nous avons organisé une compétition amicale à l'Aqua dôme LaSalle durant la fin de semaine du 20-21 novembre. Nous avons eu la chance de tester ce que nous avons développé dans le cadre de ce projet en affrontant l'équipe de l'UQTR et McGill. Nous avons même eu la chance d'avoir la marine royale canadienne pour escorter nos sous-marins.

Figure 49 - participant à gauche McGill, au centre Sonia et à droite l'UQTR

Problèmes survenus

Étant un prototype, il est certain que plusieurs problèmes mécanique, électrique et logiciel vont faire surface. Dans ce rapport nous allons discuter des problèmes qui ont impacté l'asservissement du sous-marin. Il faut savoir que les tests à l'eau sont rares (2 par jour) et d'une durée de 30 minutes. Lorsqu'un problème arrive, il faut réagir très vite si on veut avoir le temps de régler.

4.1.3. Canaux de puissance brulée

À la première journée de la compétition, on a remarqué que le sous-marin se déplaçait de façon inhabituelle. Après Investigation avec les plongeurs, on a remarqué que le moteur #5 ne tournait plus. L'équipe électrique a investi via un terminal de commande pour remarquer que l'Esc du moteur 5 ne répondait plus.

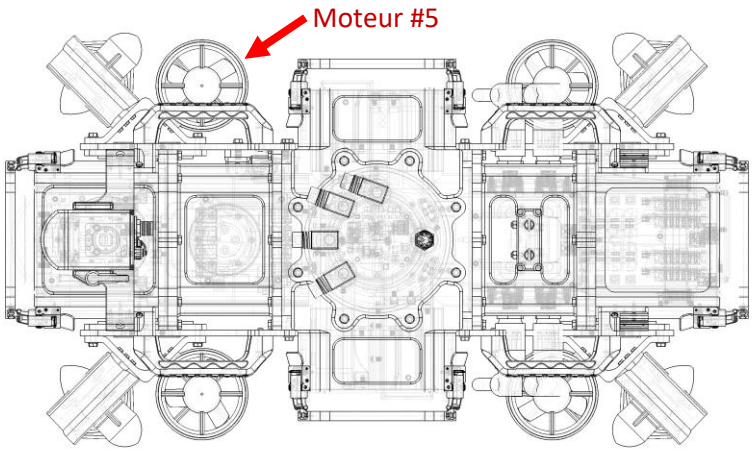

Figure 50- Identification du moteur 5

Nous savions que si on sortait le sous-marin de l'eau à la moitié de notre premier test, pour que l'équipe électrique essaye de réparer le problème, nous ne pourrions pas retourner à l'eau de la journée. Nous savions aussi qu'à cause de de la redondance de moteur, qu'il serait possible d'asservir le sous-marin avec un moteur en moins. Après avoir pris la décision de modifier le contrôleur, nous avons pris le 15 minutes restant pour modifier le contrôleur.

Malheureusement, c'est l'un des pires moteurs à perde, car c'est un moteur qui asservie l'assiette (les axes verticaux z, roll, pitch) et qu'il sert à combattre l'inclinaison du sous-marin quand il avance.

Aux lieux de générer une nouvelle matrice B en retirant le moteur #5, nous avons décidé de modifier avec les contraintes de la fonction coût du MPC. L'idée derrière est d'être capable dans le futur de désactivé des moteurs en temps réel lorsqu'on détecte des fautes et qu'il n'est pas possible de changer la taille des matrices en temps réel. Nous avons donc changé Umin et Umax de sorte qu'on désactive le moteur 5 :

$$U_{max} = [40\ 40\ 40\ 40\ 0\ 40\ 40\ 40]\ \ U_{min} = -[30\ 30\ 30\ 30\ 0\ 30\ 30\ 30]$$

Nous étions en mesure de nous déplacer, mais nous avions une erreur de 10 degrés en « Roll » et « Pitch » au régime permanent. Nous avons donc modifié les poids du contrôleur afin de réduire cette erreur statique.

$$OV = [30\ 30\ 30\ 55\ 65\ 55\ 45\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$MV = [0.2\ 0.2\ 0.2\ 0.2\ 0.4\ 0.4\ 1.1\ 0.4]$$

L'idée était d'augmenter la priorité des angles de l'assiette pour réduire l'erreur statique. Aussi nous avons permis au moteur opposé au moteur 5 de sortir de sa référence pour essayer de compenser le moteur défectueux. Nous avons été capables de réduire l'erreur statique à 4 degrés et cela ne nous a pas empêchés de continuer la compétition. Nous avons juste dû réduire les vitesses de déplacements.

4.1.4. DVL dans des conditions non optimales

Le DVL est conçu pour travailler dans des endroits naturels comme l'océan, rivière, lac, etc. Cependant il est écrit dans la documentation qu'il n'est pas recommandé pour des piscines. Or, les piscines que nous testons à l'habitude sont soit recouvertes d'une toile en vinyle ou possèdent un fond en ciment et nous n'avons jamais eu de problème. À notre surprise, le fond de la piscine à l'aqua dôme LaSalle est en tuile de céramique et nous avons remarqué que notre DVL ne retournait pas toujours de bonne mesure. Par exemple, le DVL à un échenillions de 10hz et en une seconde, il pouvait retourner qu'une seule bonne mesure. Parfois, on pouvait n'avoir aucune donnée pendant au moins 5 secondes.

Malheureusement, si le DVL ne retourne pas de valeur, le système n'est plus observable. Il ne nous est donc pas possible d'estimer les états manquants. Comme le dit Thor Fossen dans « Handbook of Marine craft hydrodynamics and motion control » (2021), la meilleure chose à faire lorsqu'un capteur fait défaut c'est de confiance au modèle dynamique.

Nous avons commencé à implémenter un filtre de Kalman étendue qui n'est pas complètement fonctionnelle. Nous avons utilisé le bloc Simulink EKF pour implémenter le filtre rapidement. Voici le schéma Simulink du filtre.

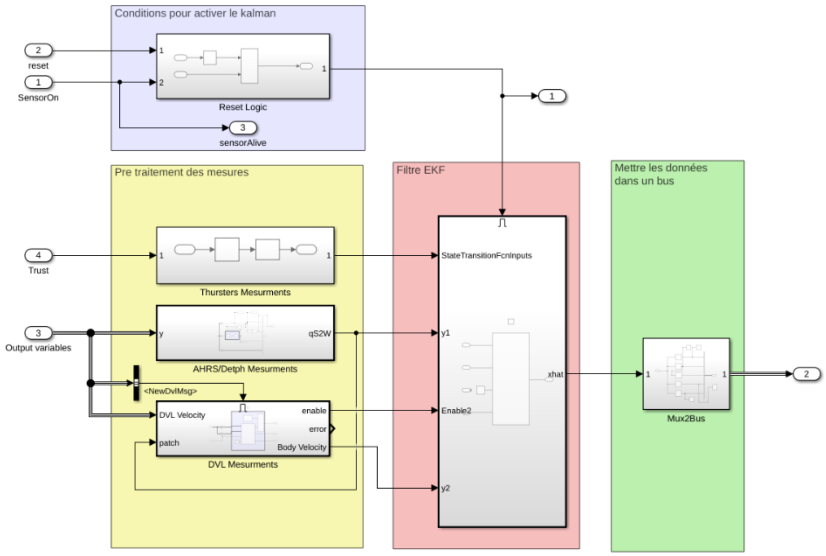

Figure 51 - Shémat du filtre de Kalman

La section en bleu s'assure que tous les capteurs sont allumés avant de démarrer le filtre de Kalman. La section en jaune traitre les données des capteurs et vérifie que celui ne présente pas de faille. La section en rouge est le filtre de Kalman et la section en vert organise l'information dans un « bus ».

L'idée de fonctionnement qu'on a implémenté est assez simple. Vu qu'on sait que notre modèle n'est pas précis, nous y avons pris une covariance très grande et nous avons mis une covariance très petite aux capteurs. De plus, le bloc Simulink nous permet d'activer ou désactiver les entrées du capteur.

Dans une utilisation normale, si tous les capteurs fonctionnent, leurs covariances sont tellement petites que le filtre retourne presque la valeur exacte du capteur. Cependant si le capteur devient fautif, on le désactive et le filtre n'a que le modèle pour estimer les états manquants.

Actuellement cette logique n'est qu'appliquée sur le DVL. De plus Matlab automatise énormément la tâche. Comme modèle de transition, nous avons simplement donné le modèle d'état non linéaire et laissons Simulink trouve la jacobienne de façon numérique. Sinon les équations de mesures font juste assigner la valeur des capteurs à leur état respectif. Voici les paramètres Simulink de notre filtre.

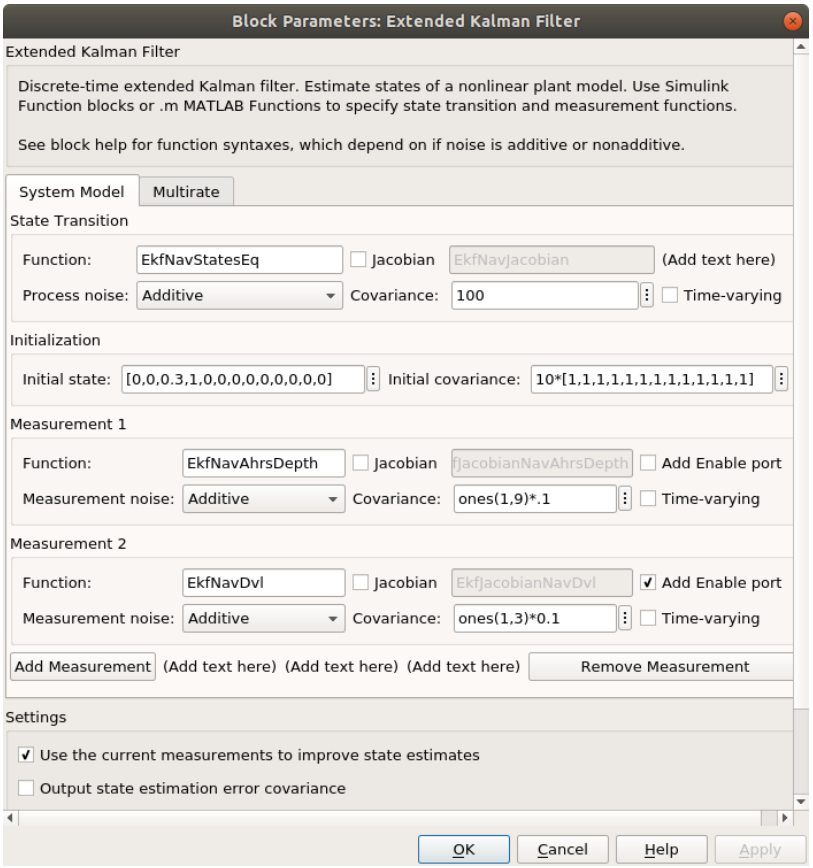

Figure 52 - Paramètre du filtre de Kalman

Même si nous savons que notre implémentation n'est pas rigoureuse, elle nous a quand même permis de naviguer même si notre DVL ne fonctionnait pas très bien.

5. Comparaison des résultats

Au cours de ce projet, nous avons donc pu comparer quatre lois de commandes différentes. Deux permettent l'asservissement du système en boucle fermée en vitesse et deux autres en position. Nous avons testé chacun de ces correcteurs en simulation ou, quand cela a été possible, sur le sous-marin réel via des tests piscine. Le but de cette section est de faire un retour sur ce qui a été observé au cours de ces différents tests. Cependant, l'idée n'est pas de choisir une loi de commande unique. En effet, un asservissement en position sera très utile pour les tâches de positionnement tandis qu'un asservissement en vitesse sera plus intéressant pour du déplacement en trajectoires. Ainsi il apparait comme une bonne idée d'implémenter ses deux modes de contrôle sur le sous-marin afin de permettre à l'équipe du club SONIA de pouvoir changer de mode en fonction de leurs besoins.

En ce qui concerne les lois de commande en vitesse, le retour d'état comme le PID ont montré des améliorations intéressantes par rapport au système en boucle ouverte. En effet, les deux ont permis d'atteindre la valeur de consigne sans dépassement par rapport à celle-ci. En revanche, les effets de couplages restent présents notamment avec le PID, bien qu'ils soient globalement très affaiblis (<1%). Le retour d'état permet de retrancher les effets de couplage aux alentours de 0.001rad/sec, ce qui est largement satisfaisant pour notre utilisation. En revanche, le PID permet d'atteindre un temps de réponse faible de t5%=1.5s tandis que le retour d'état reste sur t5%=3s. Notons qu'il s'agit déjà d'une amélioration majeure par rapport à la loi de commande implémentée sur l'AUV7. Néanmoins, pour nos fonctionnalités, il est intéressant de privilégier un temps de réponse le plus faible possible, quitte à garder quelques artefacts de couplage (qui restent très faibles) pour l'asservissement. Le PID semble donc une meilleure loi de commande que le retour d'état par rapport à ns objectifs. Cependant, avant de trancher sur la question, il sera nécessaire d'effectuer des tests piscine pour étudier les réponses du sous-marin réel.

En ce qui concerne l'asservissement en position, comme nous l'avons dit plus tôt, l'Adaptative MPC semble être notre meilleure solution puisque le LTV MPC n'apporte pas de majeure différence de performance et a pour inconvénient d'être très demandant en ressource CPU, déjà très demandé par les autres modules du sous-marin. Enfin, l'Adaptative MPC s'est montré à la fois performant et robuste ors de la compétition Inter-Québec et ce malgré la panne d'un moteur. Il s'agit là d'une situation exceptionnelle certes, mais qui sert de très bon exemple du type contraintes que doit pouvoir surmonter la loi de commande implémentée sur le sous-marin si le club veut pouvoir performer en compétition.

Projets Futurs

Ce projet de SYS802 nous a permis de grandement améliorer la commande actuelle de notre sous-marin, mais il y a encore des projets qu'on aimera accomplir pour la compétition à San Diego.

- Premièrement on aimerait implanter le même contrôleur MPC dans notre ancien sous-marin AUV7, car nous avons comme but de déployer 2 sous-marins en simultané dans le bassin de la compétition.
- Deuxièmement, nous aimerions compléter notre filtre de Kalman pour augmenter la précision de notre estimation de position et d'être en mesure de gérer en temps réel les fautes de nos capteurs.
- Troisièmement, nous aimerions implémenter une gestion qui regarde l'état des moteurs en temps réel et qui modifie les gains lorsqu'il détecte des fautes de moteurs. L'équipe électrique publie déjà l'état des canaux de puissance.
- Quatrièmement, nous aimerions améliorer notre génération de trajectoire, pour être en mesure de générer des courbes fluides entre plusieurs points.
- Dernièrement, nous aimerions implémenter notre sonar, afin d'être en mesure d'identifier les obstacles plus précisément et d'améliorer notre estimation de notre position en vitesse.

6. Conclusion

Le sous-marin AUV8 est un modèle complexe regroupant différents domaines de l'ingénierie et est conçu pour être le plus performant possible. Dans le cadre de ce cours, nous nous sommes penchés sur l'aspect commande du système en boucle fermée en étudiant différentes solutions permettant le contrôle du sous-marin.

En premier lieu, nous avons étudié la modélisation du système en boucle ouverte en présentant la linéarisation, la discrétisation, la stabilité ainsi que la commandabilité et l'observabilité. À partir de ces bases, nous avons conçu deux commandes en vitesse que sont la commande par retour d'état et la commande avec PID ainsi qu'une commande en position correspondante au MPC.

Nous avons pu implémenter le MPC sur le sous-marin réel, ce qui nous a permis de tester cette commande dans des conditions réelles en piscine lors des tests, mais aussi pendant la compétition Inter-Québec. Cette compétition a de plus permis de voir à quel point le MPC rendait le système en boucle fermée performant lors des différentes épreuves prévues.

Un tel sous-marin ne demande qu'à être amélioré dans le but d'être plus performant. Ce cours nous a permis de surmonter de nombreux défis techniques, même s'il en reste encore. Nous avons tous pu en apprendre plus sur la commande et l'application de nos connaissances sur le sous-marin nous a stimulés pour concrétiser notre apprentissage.

Dans ce rapport, nous avons apporté au club SONIA les éléments suivants

- Développer la jacobienne analytique continue de façon plus rigoureuse.
- Développer le modèle discret.
- Conception d'un contrôleur vitesse 6DDL avec PID discret basé sur un modèle dynamique.
- Conception d'un contrôleur vitesse 6DDL avec retours d'état discret basé sur un modèle dynamique.
- Conception et implémentation d'un contrôleur MPC Adaptive en position supportant la génération de code C++.
- Conception et implémentation d'un contrôleur MPC LTV en position supportant la génération de code C++.
- Commencer l'implémentation d'un filtre de kalman étendue pour filtrer nos capteurs.
- Commencer l'implémentation d'un détecteur de faute de moteur et prouvez que nous pouvons quand même fonctionner à 7 moteurs.

Bibliographie

Images

[1] : SONIA AUV, (Dernière modification en 2021). « **À propos** » repéré à < https://sonia.etsmtl.ca/fr/about-us/ >

[2] : Desgagné, A., Lamarre A., (2021). « **Identification d'un sous-marin autonome représenté à l'aide du quaternion et asservissement avec un contrôleur MPC** » Montréal, École de technologie Supérieure, Université du Québec pour le cours GPA791 (Projet Spécial), 104 p.

[3] : TeledyneMarine, (Dernière modification en 2021). « **PATHFINDER DVL** », repéré à <http://www.teledynemarine.com/Pathfinder_DVL >

[4] : VectorNav, (Dernière modification en 2021). « **VN-100** », repéré à < https://www.vectornav.com/products/detail/vn-100>

[5] : Impact Subsea, (Dernière modification 2021), « **ISD4000 Underwater depth and temperature sensor** », repéré à <https://www.impactsubsea.co.uk/isd4000/>

[6]: Liu, F., Shen, Y., He, B., Wan, J., Wang, D., Yin, Q., Qin, P., (2019). « **3DOF Adaptive Line-Of- Sight Based Proportional Guidance Law for Path Following of AUV in the Presence of Ocean Currents** ». Repéré à < https://www.mdpi.com/2076-3417/9/17/3518/htm#B31-applsci-09-03518 >

[7] : De A. Fernandes, D., J. Sørensen A., C. Donha, D., (2015). « **Path Generation for High-Performance Motion of ROVs Based on a Reference Model** » Repéré à <https://www.researchgate.net/publication/277013046_Path_Generation_for_High-Performance_Motion_of_ROVs_Based_on_a_Reference_Model>

[8] : Pinterest, (Dernière modification en 2021). « **Convex & Non-convex functions** », repéré à <https://www.pinterest.com/pin/672232681861372201>

[9] : Wikipédia, (Dernière modification en 2021). « **Model Predictive Control** », repéré à <https://en.wikipedia.org/wiki/Model_predictive_control>

Documentation

Aubin-Perron, F., Lemay, L., (2019). « **Étude Inertielle et Hydrodynamique d'un Sous-Marin Autonome pour l'intégration d'un Modèle de Contrôle** ». Montréal : École de technologie Supérieure, Université du Québec pour le cours MEC791 (Projet Spécial), 53 p.

Bensoussan, David., (2008). « **Commande Moderne : Approche par modèles continus et discrets** ». Presse internationales Polytechnique., 381 p.

Fossen, Thor I., (1994). « **Guidance and Control of Ocean Vehicles** ». Chichester: John Wiley & Sons Ltd., 494 p.

Fossen, Thor I., (2021). « **Handbook of marine craft hydrodynamics and motion control** ». Chichester: John Wiley & Sons Ltd., 710 p.

Desgagné, A., Lamarre A., (2021). « **Identification d'un sous-marin autonome représenté à l'aide du quaternion et asservissement avec un contrôleur MPC** » Montréal, École de technologie Supérieure, Université du Québec pour le cours GPA791 (Projet Spécial), 104 p.

Lavoie, O., Chennouf, R., (2018). « **Contrôle pour Sous-Marin Autonome** ». Montréal : École de technologie Supérieure, Université du Québec pour le cours SYS802, 53 p.

MathWorks, (Dernière modification en 2021). **« Optimisation Problem »** repéré à < https://www.mathworks.com/help/mpc/ug/optimization-problem.html >