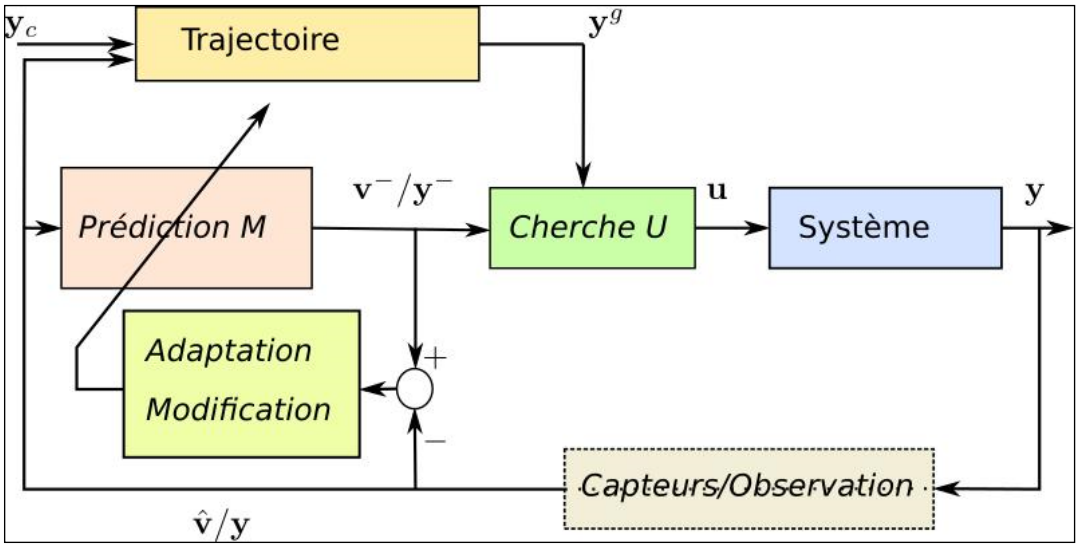
Développement d'une commande à modèle partiel appris : analyse théorique et étude pratique

Huu Phuc Nguyen

Par **Huu Phuc NGUYEN**

Développement d'une commande à modèle partiel appris : analyse théorique et étude pratique

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 16 décembre 2016
Spécialité : Technologies de l'Information et des Systèmes

D2323

Université de Technologie de Compiègne
Heudiasyc UMR CNRS/UTC 7253

Thèse

présentée pour l'obtention du grade de Docteur de l'UTC

par

Huu Phuc NGUYEN

Développement d'une commande à modèle partiel appris. Analyse théorique et Étude pratique

Thèse soutenue le 16 Décembre 2016 devant le jury composé de :

M. Luc Dugard	DR CNRS, Gipsa-lab, Grenoble	(Rapporteur)
M. Pascal Morin	Professeur, ISIR, UMPC	(Rapporteur)
M. Dominique Meizel	Professeur Émérite, Univ. de Limoges	(Examinateur)
Mme. Isabelle Fantoni	DR CNRS, Heudiasyc, UTC	(Examinatrice)
M. Ali Charara	Professeur, Heudiasyc, UTC	(Co-Directeur de thèse)
M. Jérôme De Miras	Maître de Conférences, Heudiasyc, UTC	(Directeur de thèse)

À mon fils…

Table des matières

Liste des figures

Notations

AWD	All Wheel Drive
FWD	Front Wheel Drive
EKF	Extended Kalman Filter
GP	Gaussian Process
GPS	Global Positioning System
KLSPI	Kernel Based Least Square Policy Iteration
KRLS	Kernel Recursive Least Square
KRR	Kernel Ridge Regression
LSPI	Least Square Policy Iteration
LWPR	Local Weighted Projection Regression
LTV	Linear Time Varying systems
MRAS	Model Reference Adaptive System
NMPC	Nonlinear Model Predictive Control
PI	Policy Iteration
PVTOL	Planar Vertical TakeOff and Landing
RBF	Radial basic function
RNA	Réseau de Neurones Artificiels
RTK	Real Time Kinematic
RWD	Rear Wheel Drive
SARSA	State-Action-Reward-State-Action
SOGP	Sparse Online Gaussian Process
VFA	Value Function Approximation

Remerciements

Tout d'abord, Je voudrais exprimer ma reconnaissance à mes directeurs de thèse Monsieur Jérôme DE MIRAS, Maître de Conférences et Monsieur Ali CHARARA, directeur de l'unité mixte de recherche CNRS 7253 Heudiasyc, Professeur à l'Université de Technologie de Compiègne. Ils m'ont donné la possibilité de développer ce travail de recherche et m'ont accueilli dans de bonnes conditions de travail au laboratoire Heudiasyc. Ils ont toujours eu la disponibilité pour discuter et me donner des idées, des solutions quand j'avais des difficultés. Ils m'ont aidé à développer de nouvelles perspectives sur mon métier dans le domaine de l'automatique.

Puis, je souhaite également exprimer mes remerciements aux personnes de l'équipe des drones et de la voiture intelligente qui m'ont aidé à réaliser des essais.

J'adresse ensuite mes remerciements aux membres du jury, au président Monsieur le Professeur émérite Dominique MEIZEL, à Madame la Directrice de recherche Isabelle FANTONI et particulièrement à Monsieur le Directeur de recherche Luc DUGARD et Monsieur le Professeur Pascal MORIN pour avoir accepté de rapporter ce travail, pour des remarques et des conseils.

Je remercie les doctorants du laboratoire, mes collèges à l'Université Technique Le Quy Don, Hanoï, Vietnam, et mes amis pour des discussions ouvertes où j'ai trouvé des idées et des outils très efficaces.

Finalement, un grand remerciement aux membres de ma famille, qui m'ont souvent aidé pour équilibrer mes efforts et avoir une bon rythme de travail.

Résumé de la thèse

Résumé En théorie de la commande, un modèle du système est généralement utilisé pour construire la loi de commande et assurer ses performances. Les équations mathématiques qui représentent le système à contrôler sont utilisées pour assurer que le contrôleur associé va stabiliser la boucle fermée. Mais, en pratique, le système réel s'écarte du comportement théorique modélisé. Des non-linéarités ou des dynamiques rapides peuvent être négligées, les paramètres sont parfois difficiles à estimer, des perturbations non maitrisables restent non modélisées.

L'approche proposée dans ce travail repose en partie sur la connaissance du système à piloter par l'utilisation d'un modèle analytique mais aussi sur l'utilisation de données expérimentales hors ligne ou en ligne. A chaque pas de temps la valeur de la commande qui amène au mieux le système vers un objectif choisi a priori, est le résultat d'un algorithme qui minimise une fonction de coût ou maximise une récompense.

Au centre de la technique développée, il y a l'utilisation d'un modèle numérique de comportement du système qui se présente sous la forme d'une fonction de prédiction tabulée ayant en entrée un n-uplet de l'espace joint entrées/état ou entrées/sorties du système. Cette base de connaissance permet l'extraction d'une sous-partie de l'ensemble des possibilités des valeurs prédites à partir d'une sous-partie du vecteur d'entrée de la table. Par exemple, pour une valeur de l'état, on pourra obtenir toutes les possibilités d'états futurs à un pas de temps, fonction des valeurs applicables de commande. Basé sur des travaux antérieurs ayant montré la viabilité du concept en entrées/état, de nouveaux développements ont été proposés. Le modèle de prédiction est initialisé en utilisant au mieux la connaissance a priori du système. Il est ensuite amélioré par un algorithme d'apprentissage simple basé sur l'erreur entre données mesurées et données prédites. Deux approches sont utilisées : la première est basée sur le modèle d'état (comme dans les travaux antérieurs mais appliquée à des systèmes plus complexes), la deuxième est basée sur un modèle entrée-sortie. La valeur de commande qui permet de rapprocher au mieux la sortie prédite dans l'ensemble des possibilités atteignables de la sortie ou de l'état désiré, est trouvée par un algorithme d'optimisation.

Afin de valider les différents éléments proposés, cette commande a été mise en œuvre sur différentes applications. Une expérimentation réelle sur un quadricoptère et des essais réels de suivi de trajectoire sur un véhicule électrique du laboratoire montrent sa capacité et son efficacité sur des systèmes complexes et rapides. D'autres résultats en simulation permettent d'élargir l'étude de ses performances. Dans le cadre d'un pro-

jet partenarial, l'algorithme a également montré sa capacité à servir d'estimateur d'état dans la reconstruction de la vitesse mécanique d'une machine asynchrone à partir des signaux électriques. Pour cela, la vitesse mécanique a été considérée comme l'entrée du système.

Mots-clés Commande non linéaire, commande par apprentissage, inversion numérique de modèle, approximation de fonction, commande en temps discret

Abstract In classical control theory, the control law is generally built, based on the theoretical model of the system. That means that the mathematical equations representing the system dynamics are used to stabilize the closed loop. But in practice, the actual system differs from the theory, for example, the nonlinearity, the varied parameters and the unknown disturbances of the system.
The proposed approach in this work is based on the knowledge of the plant system by using not only the analytical model but also the experimental data. The input values stabilizing the system on open loop, that minimize a cost function, for example, the distance between the desired output and the predicted output, or maximize a reward function are calculated by an optimal algorithm.
The key idea of this approach is to use a numerical behavior model of the system as a prediction function on the joint state and input spaces or input-output spaces to find the controller's output. To do this, a new non-linear control concept is proposed, based on an existing controller that uses a prediction map built on the state-space. The prediction model is initialized by using the best knowledge a priori of the system. It is then improved by using a learning algorithm based on the sensors' data. Two types of prediction map are employed : the first one is based on the state-space model, the second one is represented by an input-output model. The output of the controller, that minimizes the error between the predicted output from the prediction model and the desired output, will be found using optimal algorithm.
The application of the proposed controller has been made on various systems. Some real experiments for quadricopter, some actual tests for the electrical vehicle Zoé show its ability and efficiency to complex and fast systems. Other the results in simulation are tested in order to investigate and study the performance of the proposed controller. This approach is also used to estimate the rotor speed of the induction machine by considering the rotor speed as the input of the system.

Keywords Nonlinear control, learning control, numerical inversion model, function approximation, discret control

Les travaux de ce mémoire sont présentés partiellement dans les publications Nguyen et al. (2014), Nguyen et al. (2015a), Nguyen et al. (2015b) et Nguyen et al. (2015c), Nguyen et al. (2016) détaillées ci-dessous.

1. H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Nonlinear control of the PVTOL aircraft by numerical inversion of its behavioral model. Dans *23th IEEE International Conference on Control Application*, Antibe, France, October 2014.
2. H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Commande non linéaire en temps discret utilisant un modèle de référence d'un véhicule automobile. Dans *Sixième Journées Doctorales, Journées Nationales MACS*, Bourges, France, Juin 2015.
3. H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Nonlinear control for the vehicle by numerical inversion of its behavioral model. Dans *14th European Control Conference*, Linz, Austria, July 2015.
4. H-P. Nguyen, C. Lajnef, J. De Miras, S. Bonnet, A. Charara, et M. Eltabach. Numerical approach for estimation of the rotor speed of the asynchronous machine. Dans *8th Surveillance*, Roanne, France, October 2015.
5. H-P. Nguyen, J. De Miras, A. Charara, M. Eltabach, et S. Bonnet. How to estimate the rotor speed of the asynchronous machine using a numerical approach. *Submitted to Mechanical Systems and Signal Processing (MSSP)*, 2016.

Introduction générale

D'une manière générale, en théorie de commande, la connaissance à priori du système à contrôler est très importante pour développer une loi de commande. Les approches classiques sont essentiellement basées sur un modèle théorique du système. Le système est modélisé pour obtenir des équations mathématiques représentant le système afin de construire la loi de commande et l'appliquer au système réel. L'utilisation des informations théoriques n'empêche pas d'extrapoler des connaissances sur le système à partir des données de capteurs. Par exemple, la commande adaptative propose une solution pour faire évoluer automatiquement les paramètres du contrôleur. La commande par apprentissage ou par réseaux de neurones approxime la valeur de la commande en fonction des états à partir des données expérimentales.

Par conséquent, l'utilisation d'un modèle numérique tabulé pour la représentation du système est un choix très flexible. Les systèmes embarqués, de plus en plus puissants, permettent de sauvegarder ces données volumineuses pour la commande. La commande utilisant un modèle numérique tabulé de comportement du système a été développée dans une thèse précédente Bonnet (2008), où le modèle mathématique sous forme d'état du système est utilisé. Sachant que le système réel s'écarte du modèle théorique (variation des paramètres, dynamiques non-modélisées, perturbations, etc), on désire mettre en œuvre une commande qui ne nécessite que très peu de réglage, sans pour autant avoir un modèle précis du système à contrôler, et qui respecte la dynamique théorique définie. L'objectif de cette thèse est de contribuer au développement d'un concept de commande satisfaisant ces exigences.

En combinant l'apprentissage statistique avec la commande déjà développée, ce travail propose une amélioration des fonctionnalités. Le modèle mathématique du système peut-être utilisé comme donnée initiale de la commande, qui est une table de prédiction pour représenter la dynamique du système à contrôler sous la forme entrée-état ou entrée-sortie. Les données réelles peuvent être utilisées pour construire ou améliorer la table de prédiction. En utilisant cette table sous une forme entrée-sortie, on peut éviter l'utilisation d'un observateur et construire une nouvelle forme de la commande.

Organisation de ce document

Ce document est divisé en cinq chapitres, les deux premiers chapitres portent sur la conception de la commande, ses applications seront présentées dans les trois derniers chapitres.

Le premier chapitre porte sur l'approche numérique tabulée de la commande et sa conception.

Le deuxième chapitre, on expose la bibliographie sur l'apprentissage statistique et, en partant de ces éléments, on propose un stratégie d'apprentissage pour la commande. Cet algorithme sera utilisé pour modifier la table de prédiction.

Le troisième chapitre concerne une application à un véhicule aérien. Tout d'abord, la commande est appliquée pour le modèle simplifié du PVTOL. Ensuite, la commande est utilisée pour stabiliser l'attitude d'un quadricoptère complet. Des résultats réels sont également fournis.

Dans le quatrième chapitre, les résultats en simulation et essais réels pour un véhicule autonome sont présentés.

Le dernier chapitre porte sur l'observation de la vitesse d'une machine asynchrone. Dans ce chapitre, on introduit comme algorithme d'observation une version très proche de l'algorithme de commande. Les résultats en simulation montrent l'efficacité de la méthode.

Le mémoire se termine par un chapitre de conclusion et perspectives.

Chapitre 1

Conception de la commande

Dans ce chapitre, des rappels sont faits à partir de la thèse de Bonnet (2008) qui propose une approche numérique pour une commande en temps discret. Dans ces travaux, qui vont servir de base aux développements décrits dans ce mémoire, le modèle du système est décrit à partir d'une forme entrée-état discrète (table) comme les états atteignables à partir des points de la table en un pas de temps. Une simulation par intégration numérique est utilisée sur les équations différentielles du modèle pour calculer ces points. Une évolution de cette commande, basée sur la forme entrée-sortie, sera ensuite introduite.

Sommaire

1.1 Commande par modèle tabulé de comportement

1.1.1 Concept de commande

Le concept de commande non linéaire basée sur un modèle tabulé de comportement d'un système a été proposé dans les travaux de Bonnet (2008), De Miras et Bonnet (2014). L'idée principale de cette stratégie de commande en temps discret est d'utiliser le comportement d'un système de référence linéaire ou non linéaire stable comme comportement de référence défini à priori pour le système à piloter en boucle semi fermée. De cette façon, si les sorties du système sont capables, à chaque pas de temps, de correspondre à celles du système de référence, le système à commander présentera la même dynamique que la référence et convergera vers la consigne.

Dans ce qui suit, les vecteurs et les matrices sont indiqués en gras. Considérons un modèle non linéaire invariant par rapport au temps sous la forme :

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= h(\mathbf{x}(t)) \end{cases} \tag{1.1}$$

où $\mathbf{x} \in \mathcal{S} \subset \mathbb{R}^n$ est le vecteur d'état, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^d$ est le vecteur d'entrée et $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^m$ est le vecteur sortie. On suppose que la prédiction suivante est disponible :

$$\mathbf{x}_{k+1} = p_x(\mathbf{x}_k, \mathbf{u}_k) \tag{1.2}$$

où la fonction p_x est simplement un échantillonnage et une discrétisation de la fonction d'état f dans (1.1) utilisant un pas de temps Δt tel que $t_k = k \cdot \Delta t$, les vecteurs $\mathbf{x}_k = \mathbf{x}(k \cdot \Delta t), \mathbf{u}_k = \mathbf{u}(k \cdot \Delta t), \mathbf{y}_k = \mathbf{y}(k \cdot \Delta t)$ sont les vecteurs d'état, d'entrée et de sortie en temps discret au pas de temps k respectivement. Dans le cas général, il est difficile de trouver l'expression mathématique exacte de p_x. Mais, des valeurs approximatives pour un vecteur d'état et un vecteur d'entrée donnés peuvent être calculées par l'intégration numérique de f.

On souhaite stabiliser $\mathbf{y}$ autour du point $\mathbf{y}_c$. De façon équivalente, il faudra stabiliser $\hat{\mathbf{y}} = \mathbf{y} - \mathbf{y}_c$ autour de zero. Supposons le système linéaire stable homogène ci-dessous :

$$\dot{\hat{\mathbf{y}}}(t) = \mathbf{A} \cdot \hat{\mathbf{y}}(t), \hat{\mathbf{y}} \in \mathbb{R}^m, \mathbf{A} \in \mathcal{M}^{m \times m}. \tag{1.3}$$

La matrice $\mathbf{A}$ est construite en considérant les dynamiques estimées du système à piloter. Bien entendu toutes ses valeurs propres ont une partie réelle négative. Considérant (1.3), la trajectoire de $\mathbf{y}(t)$ convergera vers le point $\mathbf{y}_c$. La matrice $\mathbf{A}$ décrit alors la dynamique de la boucle fermée désirée. On a décrit un système linéaire stable mais de façon plus générale on peut choisir un système stable non-linéaire comme référence. L'objectif de commande est de trouver à chaque pas de temps k le vecteur d'entrée $\mathbf{u}_k$ telle que la sortie prochaine $\mathbf{y}_{k+1}$ soit aussi proche que possible de $\mathbf{y}^g_{k+1}$, avec $\mathbf{y}^g_{k+1}$ obtenue à partir de la discrétisation de (1.3) comme suit :

$$\mathbf{y}^g_{k+1} = e^{\mathbf{A} \cdot \Delta t}(\mathbf{y}_k - \mathbf{y}_c) + \mathbf{y}_c. \tag{1.4}$$

Étant donné la nature entrée/état de la représentation choisie, on est dans un cadre où la trajectoire de référence et la trajectoire de sortie ont des dynamiques du même ordre. A chaque pas de temps, le vecteur sortie $\mathbf{y}_{k+1}$ doit être calculé. Ainsi on calcule non seulement la prédiction d'état (1.2) mais aussi la prédiction de sortie par l'intermédiaire d'une fonction p_y :

$$\mathbf{y}_{k+1} = p_y(\mathbf{x}_k, \mathbf{u}_k). \tag{1.5}$$

Dans de nombreux exemples, on notera que p_y est un sous-ensemble de p_x, ce qui simplifie le problème. On introduit le vecteur $\mathbf{v} = \left[\mathbf{x}^T, \mathbf{y}^T\right]^T$ pour obtenir $\mathbf{x}$ et $\mathbf{y}$ comme suit : $\mathbf{x} = \mathbf{P}_x \cdot \mathbf{v}, \mathbf{y} = \mathbf{P}_y \cdot \mathbf{v}$. Ce vecteur sera utilisé dans la suite pour construire l'algorithme de commande. A l'étape $k+1$ ce vecteur est :

$$\mathbf{v}_{k+1} = \left[p_x(\mathbf{x}_k, \mathbf{u}_k), p_y(\mathbf{x}_k, \mathbf{u}_k)\right]^{\mathrm{T}} = p(\mathbf{x}_k, \mathbf{u}_k). \tag{1.6}$$

Finalement, les entrées du système au pas suivant résultent de la résolution du problème de minimisation suivant :

$$\mathbf{u}_k = \arg\min_{u \in \mathcal{U}} \left\| \mathbf{y}^g_{k+1} - \mathbf{P}_y \cdot p\left(\mathbf{x}_k, \mathbf{u}\right) \right\|_2. \tag{1.7}$$

Supposons qu'à l'étape k, l'estimation d'état du système notée par $\hat{\mathbf{x}}_k$ est disponible, et la sortie $\mathbf{y}_k$ est mesurée, cela va permettre de calculer le vecteur $\mathbf{v}_k$. De plus, afin de simplifier l'exposé, on suppose que les entrées du système et l'estimation d'état sont calculées sans délai : à partir des mesures au pas k, les entrées $\mathbf{u}_k$ peuvent être calculées et appliquées instantanément au système [1].

Le but de l'algorithme de commande est de déterminer la solution approchée du problème (1.7), même si p (i.e p_x et p_y) sont généralement mal connues. Nous supposons encore ici que des valeurs approximatives spécifiques de ces fonctions peuvent être calculées à tout moment dans l'ensemble continu $\mathcal{S} \times \mathcal{U}$. L'algorithme est divisé en quatre grandes étapes :

1. Mesure des sorties et estimation des états,
2. Estimation de l'erreur de prédiction,
3. Détermination de la référence basée sur (1.4)
4. Recherche des entrées à appliquer (1.7).

L'étape 1 joue un rôle important comme dans tous les systèmes de contrôle basés sur une représentation d'état. En général, des états du système sont difficiles à mesurer directement. A partir des mesures des sorties et d'un modèle du système, les techniques d'observation permettent la reconstruction et le filtrage de tous les états du système. Dans cette étape, on peut utiliser, sans limitation, un filtrage déterministe, un filtrage de Kalman, des observateurs à grand gain, etc. La contrainte essentielle est de pouvoir

1. Dans les faits, on peut se passer de cette hypothèse en travaillant sur une prédiction de l'état un pas de temps plus loin.

estimer le retard introduit entre les états, typiquement entre une variable et sa dérivée, afin de pouvoir compenser ce retard par une prédiction (De Miras et Bonnet (2014)).

L'étape 2 est nécessaire car la table de prédiction est construite en utilisant un modèle du système. Ce modèle contient nécessairement des approximations qui rendent la prédiction intrinsèquement différente de la réalité. En plus d'une modélisation simplificatrice, des perturbations externes non décrites dans le modèle peuvent ajouter un biais aux prédictions par rapport aux valeurs réelles. Donc pour fonctionner, l'algorithme doit compenser au mieux ces écarts de prédiction. Fondamentalement, l'erreur de prédiction au pas k est la différence entre la prédiction à partir des données au pas $k-1$ et l'estimation de l'état obtenue au pas k. On peut définir le vecteur d'erreur ϵ_k comme suit :

$$\epsilon_k = p\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) - \mathbf{v}_k.$$

Les prédictions utilisées dans l'algorithme pour trouver les valeurs de commande sont corrigées avec ce vecteur d'erreur ϵ_k en supposant que l'erreur au pas k est suffisamment proche de celle obtenue à l'étape $k-1$. On considère alors que la dynamique de l'erreur n'est pas plus rapide que celle à contrôler. En cas de bruits sur les mesures, il est utile d'utiliser un filtrage pour éliminer le contenu à haute fréquence injecté dans la mesure d'erreur. On peut utiliser un filtrage numérique classique du premier ordre passe-bas comme ci-après :

$$\varepsilon_k = \varepsilon_{k-1} + \alpha\left(p\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) - \mathbf{v}_k\right) \tag{1.8}$$

avec α inclus dans $[0,1]$. Ainsi, dans le cas d'un biais constant, l'erreur de prédiction converge vers zéro.

L'étape 3 se base sur l'équation (1.4) et est donc calculée en ligne en fonction du dernier état estimé. Dans les faits, le modèle de référence n'est pas indépendant du modèle qui le suit [2]. En effet, on utilise comme condition initiale du calcul l'état estimé du système et pas l'état précédent du modèle de référence. On n'obtient donc pas exactement la dynamique prévue mais, en respectant une cohérence entre dynamique de la référence et capacité du système, le système a la capacité de se déplacer vers l'objectif qu'on lui impose et on obtient la convergence.

On arrive à l'étape 4 de l'algorithme où on va appliquer la méthode numérique de recherche de la solution approchée $\mathbf{u}_k$ au problème (1.7). Cet algorithme utilise une discrétisation des entrées $\mathcal{U} = [\underline{\mathbf{u}}, \overline{\mathbf{u}}]$ pour générer des intervalles :

$$\mathcal{U}_{i\in 1,\ldots,d} = \left\{ u_j^i | u_j^i = \underline{u}_i + j\frac{\overline{u}_i - \underline{u}_i}{N_i}, j \in 0,\ldots,N_i \right\} \tag{1.9}$$

2. Si on a deux systèmes indépendants et qu'il y a accumulation d'erreurs à chaque pas, on finit par avoir des incohérences comme demander une vitesse nulle tout en n'ayant pas atteint la position désirée. Ou en fait, le but premier du modèle de référence est de conserver cette cohérence entre les états vis à vis de l'objectif.

où $u_i, i \in 1, \ldots, d$ sont les composantes du vecteur d'entrée $\mathbf{u}$. Chaque composant u_i appartenant à l'intervalle $[\underline{u}_i, \overline{u}_i]$ est discrétisé par $N_i + 1$ points tels que $u_j^i = \underline{u}_i + j\frac{\overline{u}_i - \underline{u}_i}{N_i}, j \in 0, \ldots, N_i$. Pour chaque point qui n'est pas un point de cette grille, il sera nécessaire de faire intervenir un algorithme d'interpolation pour définir la solution de 1.6. L'interpolation barycentrique est pour l'instant utilisée (Munos et Moore (1998)). Dans le cas où p n'est pas une fonction convexe, la solution du problème de minimisation peut être trouvée en utilisant un algorithme de recherche exhaustive [3]. Si p est une fonction convexe, l'algorithme du gradient basé sur les vertex de la discrétisation développé dans Bonnet (2008) pourra être mis en œuvre.

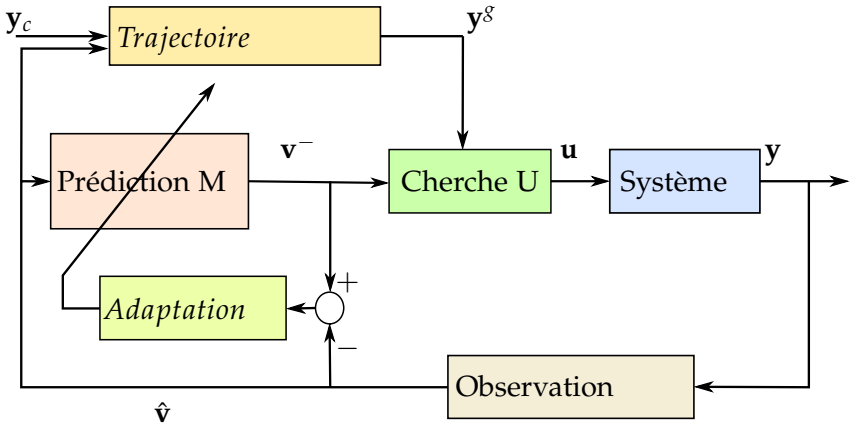


FIGURE 1.1 – *Le schéma de la commande*

Le schéma complet de la commande est montré sur la Figure 1.1. Au pas k le vecteur de sortie $\mathbf{y}_k$ est mesuré. Puis, le bloc "*Observation*" estime l'état et les sorties et renvoie $\hat{\mathbf{v}}_k$ une estimation du vecteur $\mathbf{v}_k$. Ensuite, l'erreur ε_k et la prédiction $\mathbf{v}_{k+1}^-$ de $\mathbf{v}$ sont calculées par les blocs "*Adaptation*" en utilisant l'équation (1.8) et par le bloc "Prédiction M" en utilisant l'équation (1.6) respectivement. A partir de $\hat{\mathbf{v}}_k$ on peut aussi calculer la référence interne $\mathbf{y}_{k+1}^g$ basée sur l'équation (1.4) dans le bloc "*Trajectoire*". Finalement le bloc "*Cherche U*" renvoie les valeurs de commande $\mathbf{u}_k$. Ce bloc représente l'algorithme qui résout le problème (1.7).

1.1.2 Algorithme d'interpolation

La fonction de prédiction est implémentée par une grille unitaire hypercubique dans l'espace joint formé des entrées et des états. Cette grille est illustrée sur la figure 1.2. Chaque point représente un ensemble d'entrées $\mathbf{u}_k$ et d'états $\mathbf{x}_k$ du système : $\mathbf{z} = [\mathbf{u}^T, \mathbf{x}^T]^T$. Le vecteur $\mathbf{v}_{k+1}$ dans l'équation (1.6) est calculé pour tous les points $\mathbf{z}_k = [\mathbf{u}_k^T, \mathbf{x}_k^T]^T$ de la grille. Pour accéder à une prédiction pour les points qui ne sont pas présents dans la grille, il est nécessaire de faire une interpolation par rapport aux

3. Bien entendu, plus la dimension augmente, plus le problème devient compliqué.

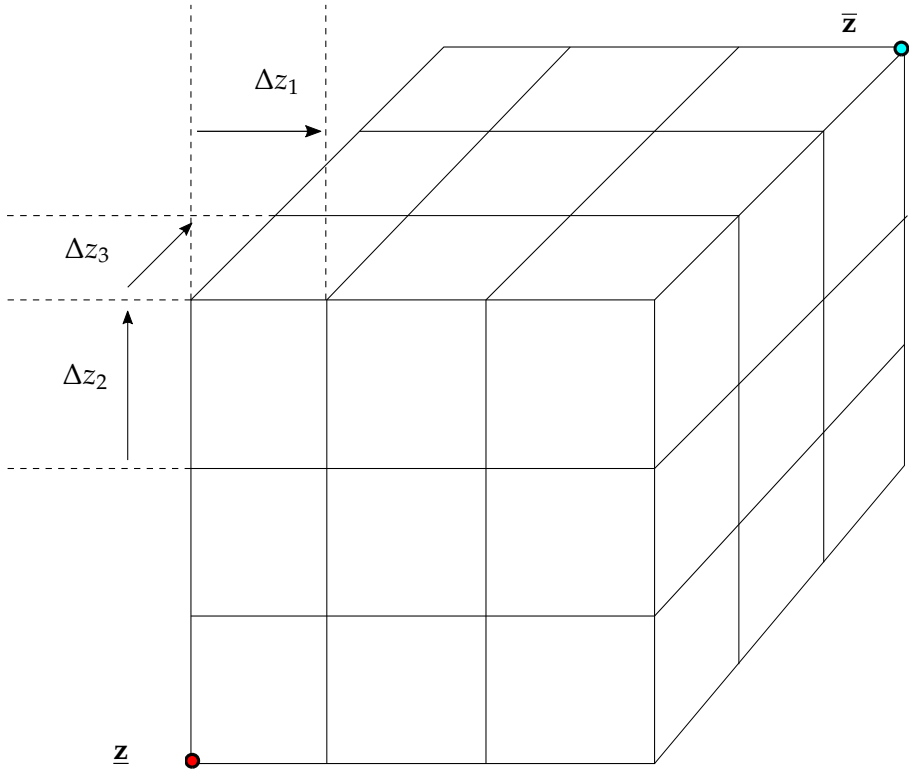


FIGURE 1.2 – *Discrétisation régulière dans l'espace $\mathcal{S} \times \mathcal{U}$ de dimension d*

plus proches voisins. Dans cette version de la commande on utilise l'interpolation barycentrique de Munos et Moore (1998). Cette interpolation est basée sur les coordonnées barycentriques qui déterminent la position d'un point $\mathbf{p}$ dans un simplexe par rapport à ses sommets $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_d$ (Figure 1.3).

$$\mathbf{p} = \omega_0 \mathbf{p}_0 + \omega_1 \mathbf{p}_1 + ... + \omega_d \mathbf{p}_d$$

avec $\sum_{i=0,d} \omega_i = 1$ et $\omega_i \geq 0$ pour tout i. La valeur d'une fonction Γ de $\mathbf{p}$ peut être approximée (interpolation barycentrique) par :

$$\Gamma(\mathbf{p}) = \omega_0 \Gamma(\mathbf{p}_0) + \omega_1 \Gamma(\mathbf{p}_1) + ... + \omega_d \Gamma(\mathbf{p}_d)$$

.

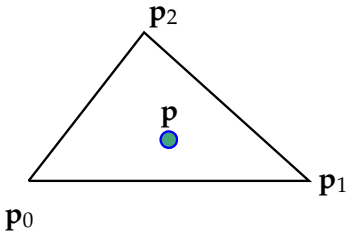


FIGURE 1.3 – *Coordonnée barycentrique dans l'espace 2D*

Pour utiliser cette interpolation, chaque sous-hypercube de la grille est divisée par une triangulation de COXETER-FREUDENTHAL-KUHN Moore (1992). $\mathbf{z}$ est un vecteur de dimension d. Un hypercube contient $d!$ simplexes. Le point $\mathbf{z}_k = [\mathbf{u}_k^T, \mathbf{x}_k^T]^T$ à interpoler est nécessairement contenu dans un des hypercubes définis par la grille et donc dans un simplexe de la triangulation. Ce simplexe est représenté par ses sommets $\sigma = \{\xi_0, \xi_1, ..., \xi_d\}$. Le point $\mathbf{z}_k$ est localisé dans l'hypercube par ses coordonnées locales $[\alpha_1, \alpha_2, ..., \alpha_d]^T$.

$$\mathbf{z}_k = \xi_0 + \sum_{i=1,d} \alpha_i (\Delta z_i)$$

avec $\Delta z_i = \xi_{i+1} - \xi_i$. Il existe une permutation π du vecteur $[1, 2, ..., d]^T$ telle que

$$1 \geq \alpha_{\pi(1)} \geq \alpha_{\pi(2)} \geq ... \geq \alpha_{\pi(d)} \geq 0$$

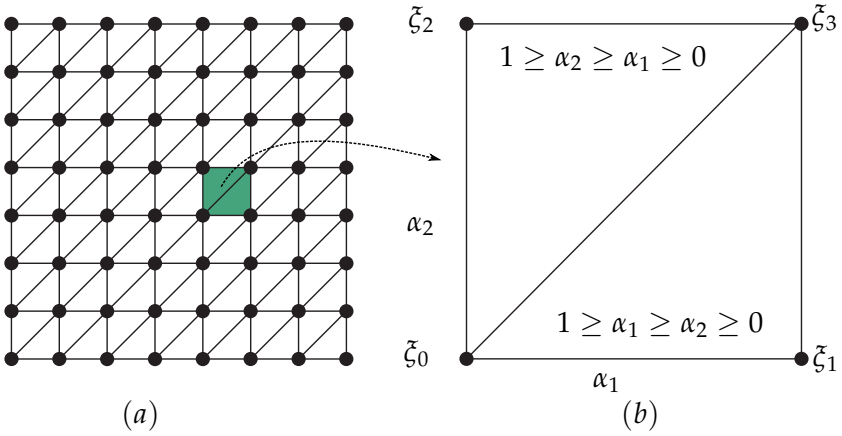


FIGURE 1.4 – *Triangulation de COXETER-FREUDENTHAL-KUHN*

La Fig. 1.4 montre le cas 2D, un rectangle contient deux simplexes. La Fig. 1.5 montre le cas 3D, chaque cube contient $3! = 6$ simplexes. Le point $\mathbf{z}_k$ est donc également déterminé par :

$$\mathbf{z}_k = \xi_0 + \sum_{i=1,d} \alpha_{\pi(i)} (\Delta z_{\pi(i)})$$

$\mathbf{z}_k$ est localisé dans le simplexe par ses coordonnées barycentriques $[\omega_0, \omega_1, ..., \omega_d]^T$

$$\mathbf{z}_k = \omega_0 \xi_0 + \sum_{i=1,d} \omega_i \xi_{\pi(i)}$$

donc, on a :

$$\begin{bmatrix} \omega_0 \\ \omega_1 \\ ... \\ \omega_d \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & ... & 0 \\ 0 & 1 & -1 & ... & 0 \\ ... & ... & ... & ... & -1 \\ 0 & 0 & ... & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \alpha_{\pi(1)} \\ ... \\ \alpha_{\pi(d)} \end{bmatrix}$$

La fonction $\mathbf{v}_k$ dans l'équation (1.6) est interpolée. L'erreur d'interpolation est proportionnelle à la taille de la grille.

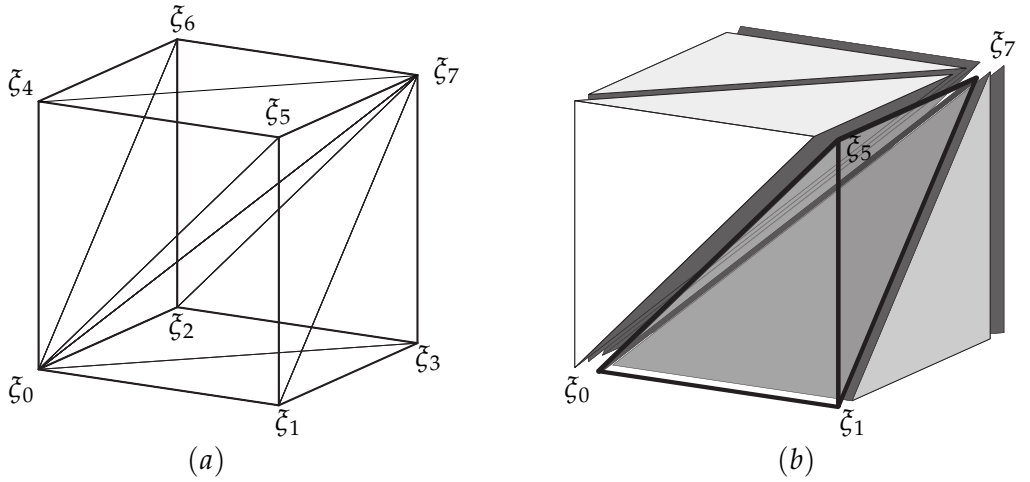


FIGURE 1.5 – *Triangulation de COXETER-FREUDENTHAL-KUHN d'un cube*

1.1.3 Algorithme d'optimisation

Pour résoudre le problème d'optimisation Eq. (1.7), il existe plusieurs méthodes dans la littérature. Les données utilisées pour la commande étant organisées sur une grille et l'interpolation étant basée sur les simplexes, les méthodes basées sur les simplexes ont été explorées. L'algorithme 1 a été proposé par Bonnet (2008). L'idée de cette méthode est de chercher le simplexe dans l'espace des sorties qui est le plus proche de la sortie désirée. Une fois ce simplexe trouvé, on remonte à l'espace des entrées afin de trouver la valeur de commande cherchée. Dans cet algorithme, un simplexe et une enveloppe affine sont définis par :

$$\mathbf{conv}\,(\mathcal{X}) = \left\{ \sum_{i=1}^{k} \omega_i \cdot \mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, \omega_i \in \mathbb{R}^+, \sum_{i=1}^{k} \omega_i = 1 \right\}$$

$$\mathbf{aff}\,(\mathcal{X}) = \left\{ \sum_{i=1}^{k} \omega_i \cdot \mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, \omega_i \in \mathbb{R}, \sum_{i=1}^{k} \omega_i = 1 \right\}$$

Au pas courant, l'état du système est $\mathbf{x}$. Cet algorithme commence par chercher un simplexe dans l'espace des entrées qui minimise la distance au point de référence $\mathbf{y}^g$. En premier lieu, un simplexe arbitraire est choisi après génération de simplexes de sortie en utilisant l'interpolation linéaire (lignes $3 - 9$). A l'étape suivante, le point de référence est projeté sur $\mathbf{aff}(\sigma')$. En utilisant les coordonnées de ce point $\mathbf{y}_\perp$, on peut détecter s'il est dans le simplexe σ' (ligne 17) et renvoyer un vecteur de booléens. Dans le cas où il existe un élément i étant faux, la solution doit être cherchée dans un sous-simplexe suivant. Les nouveaux simplexes des entrées et des sorties à considérer sont déterminés par une opération de pivot (ligne 20) par rapport à un vertex. L'algorithme est répété itérativement jusqu'à ce qu'on ne trouve plus de vertex de rotation. On a alors obtenu le simplexe désiré contenant la solution. Une autre partie de l'algorithme permet de

Algorithme 1 Recherche de la commande

Fonction CALCULATEACTION($\mathbf{x}, \mathbf{y}^g, \varepsilon_k$)
  $d \leftarrow dim(\mathcal{U})$ ▷ Inputs' dimension
  $s_i \leftarrow |\mathcal{U}_i|, i \in 1, \ldots, d$ ▷ Number of points in $\mathcal{U}_i$, and $\sigma = \mathbf{conv}\{\xi_0, \ldots, \xi_d\}$,$\sigma' = \mathbf{conv}\{\Gamma_0, \ldots, \Gamma_d\}$
  $\xi_0 \leftarrow \lfloor \frac{s}{2} \rfloor$ ▷ Initial vertex at center point for inputs
  $\Gamma_0 \leftarrow \mathbf{P}_y \cdot (p(\mathbf{x}, \xi_0) - \varepsilon_k)$ ▷ Initial vertex for outputs $\mathcal{Y}$
  pour $i \leftarrow 1, \ldots, d$ **faire**
    $\xi_i \leftarrow \xi_{i-1} + e_i$ ▷ $\mathbf{e}$ is canonical basis of $\mathbb{R}^d$
    $\Gamma_i \leftarrow \mathbf{P}_y \cdot (p(\mathbf{x}, \xi_i) - \varepsilon_k)$
  fin pour
  répéter
    pour $i \leftarrow 1, \ldots, d$ **faire**
      $\gamma_i \leftarrow \Gamma_i - \Gamma_0$ ▷ Calculate frame of $\mathbf{aff}(\sigma')$
    fin pour
    Calculate projection $\mathbf{y}_\perp$ of $\mathbf{y}^g$ onto $\mathbf{aff}(\sigma')$
    Solve α from $\Gamma'_0 + \sum_{i=1}^{d} \gamma_i \cdot \alpha_i = \mathbf{y}_\perp$
    $\alpha' \leftarrow \mathbf{T} \cdot \alpha$ ▷ Transform to $\mathbb{R}'_d$
    $\mathbf{r} \leftarrow$ TESTSIMPLEX(α')
    $n \leftarrow 0$ ▷ Number of pivots
    pour $i : \mathbf{r}_i = false$ **faire**
      $\tilde{\xi}_i \leftarrow$ PIVOTVERTEX(ξ_i)
      si $1 \leq \tilde{\xi}_i \leq s$ **alors** ▷ Check in $\mathcal{U}$
        $\xi_i \leftarrow \tilde{\xi}_i$ ▷ Recalculate σ and σ'
        $\Gamma_i \leftarrow \mathbf{P}_y \cdot (p(\mathbf{x}, \xi_i) - \varepsilon_k)$
        $n \leftarrow n + 1$
      fin si
    fin pour
  jusqu'à $n = 0$
  $\mathbf{y} \leftarrow$ PROJECTONSIMPLEX($\sigma', \mathbf{y}^g$)
  Calculate $\mathbf{u}$ as orthogonal projection of $\mathbf{y}$ onto $\mathbf{aff}(\sigma)$
  renvoie u
fin Fonction

trouver le point dans le simplexe solution, le point dont la distance à la référence est minimum (ligne 28). Finalement, le vecteur de commande est la projection orthogonale de ce point référence sur **aff**(σ).

Puisque les entrées de commande sont calculées par approximation sur une représentation discrète, le temps d'exécution est finie et dépend seulement du nombre de points utilisés pour construire la table de prédiction. Mais l'erreur d'approximation est linéairement proportionnelle au carré de la taille de la grille de discrétisation. A cela il faut ajouter le temps nécessaire pour calculer l'interpolation de p (p_x et p_y). Une autre limite correspond à l'espace nécessaire pour stocker la table.

Méthode de Nelder-Mead

Un autre algorithme utilisable a été proposé par Nelder et Mead (1965) pour le problème de minimisation sans contrainte. Dans cet algorithme, on n'a pas besoin du gradient et la taille du simplexe est variable. Des stratégies existent pour tenir compte des contraintes comme par exemple dans Luersen et al. (2004). La figure 1.6 montre les opérations utilisables pour rechercher la solution du problème de minimisation dans un espace 2D.

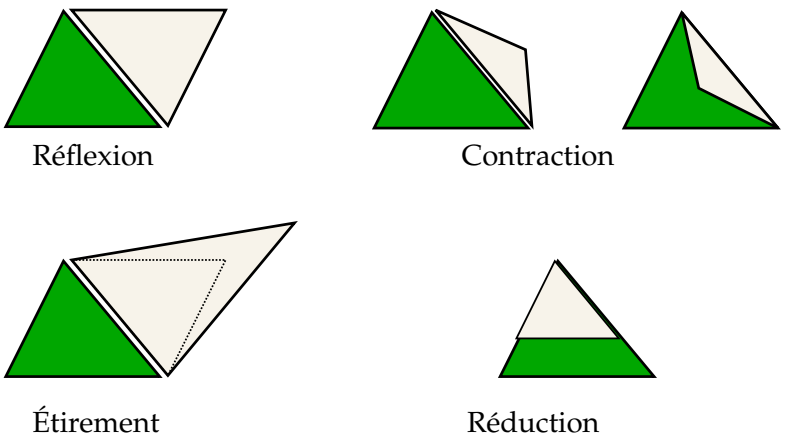


FIGURE 1.6 – *Les opérations du simplex utilisées par la méthode de Nelder-Mead*

Dans un espace N-D, un simplex est construit en utilisant $N+1$ points $\mathbf{x}_1, \cdots, \mathbf{x}_{N+1}$. Il y a quatre opérations primitives : réflexion, contraction, étirement et réduction. Un point centre est défini par $\mathbf{x}_0 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)$. La réflexion du simplex est faite en remplaçant un point $\mathbf{x}_{N+1}$ du simplexe par son image $\mathbf{x}_r = \mathbf{x}_0 + \alpha(\mathbf{x}_0 - \mathbf{x}_{N+1})$ avec $\alpha > 0$. La contraction du simplex est faite utilisant l'image $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_{N+1} - \mathbf{x}_0)$, avec $0 < \rho <= 0.5$. Pour l'étirement, on calcule la réflexion $\mathbf{x}_r$ afin de calculer l'étirement par $\mathbf{x}_e = \mathbf{x}_0 + \gamma(\mathbf{x}_r - \mathbf{x}_0)$, avec $\gamma > 0$. Ces trois opérations utilisent un point $\mathbf{x}_0$ comme le centre de gravité pour calculer l'image d'un seul point $\mathbf{x}_{N+1}$ à modifier. La quatrième

opération modifie tous les points du simplex sauf $\mathbf{x}_1$, $\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$ avec $0 < \sigma < 1$. Les valeurs des coefficients les plus utilisés sont $\alpha = 1, \gamma = 2, \rho = 0.5, \sigma = 0.5$.

Pour trouver la minimisation d'une fonction $f(\mathbf{x})$, l'algorithme commence par choisir un simplex de façon arbitraire. Ses valeurs $f(\mathbf{x}_i)$ sont indexés de manière croissante du $f(\mathbf{x}_1)$ au $f(\mathbf{x}_{N+1})$. La dernière valeur $f(\mathbf{x}_{N+1})$ sera remplacée par son image telle que la meilleure valeur est conservée. L'algorithme est répété jusqu'à ce que le simplex atteigne une taille inférieure à un seuil ou que le nombre maximal d'itérations soit atteint. L'algorithme est détaillé par l'algorithme 2.

La figure 1.7 montre la minimisation de la fonction $f(x_1, x_2) = (x_1 - 0.5)^2 + x_2^2$ autour du point $x_1 = 0.5$ et $x_2 = 0$. Le simplex est initialisé aléatoirement. La solution trouvée utilise moins de vingt itérations : étirement, réflexion, contraction, réflexion, réflexion, contraction, contraction, etc. L'avantage de cette méthode est la facilité de sa mise en œuvre. On n'a pas besoin de la dérivée de la fonction. Le simplexe de départ reste très important dans la performance de la recherche et, en pratique, pour l'application à la commande, il est judicieux d'initialiser le simplex autour de la solution précédente.

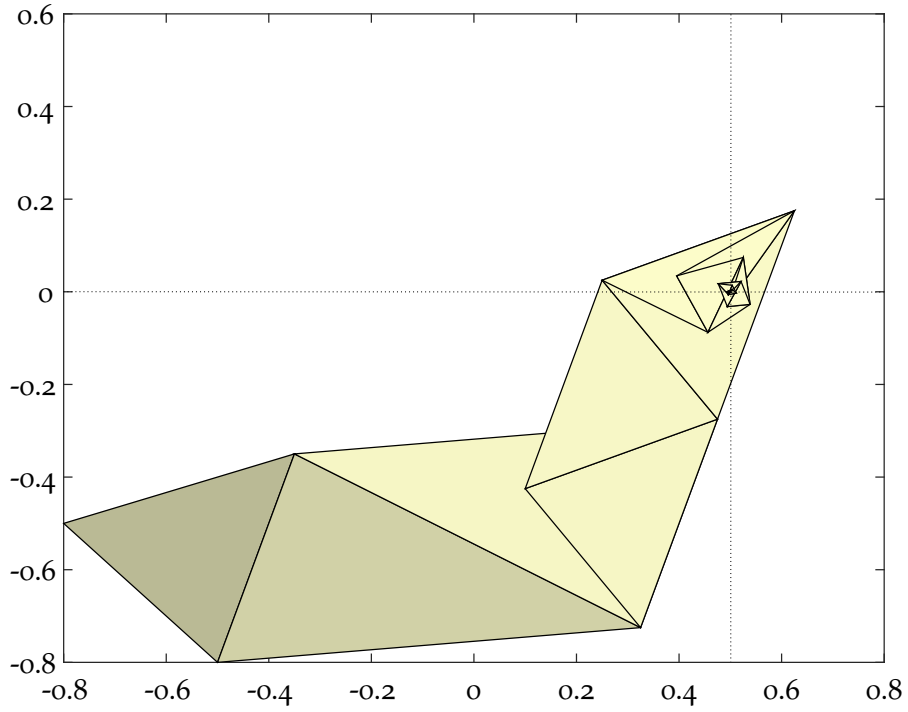


FIGURE 1.7 – *La minimisation de la fonction* $f(x_1, x_2) = (x_1 - 0.5)^2 + x_2^2$ *avec la méthode de Nelder-Mead*

1.1.4 Remarques

La commande décrite requiert une table de prédiction aussi précise que possible. Elle est générée par simulation pour tous les points liés à une grille régulière construite

Algorithme 2 Nelder-Mead algorithme

Fonction NmMinSearch($\varepsilon_f, \varepsilon_{\mathbf{x}}, n_{max}$)
Initialiser le simplex de départ
  $\Gamma_i \leftarrow f(\mathbf{x}_i), i = 1, \cdots, N$ ▷ Initialisation du simplex
  $f(\mathbf{x}_1) \leq \cdots \leq f(\mathbf{x}_{N+1})$ ▷ Réindexation
  $\mathbf{x}_0 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)$ ▷ Calcul du centre des points
  répéter
    $f(\mathbf{x}_1) \leq \cdots \leq f(\mathbf{x}_{N+1})$ ▷ Réindexation
    $\mathbf{x}_0 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)$ ▷ Calcul du centre des points
  Réflexion
    $\mathbf{x}_r = \mathbf{x}_0 + \alpha(\mathbf{x}_0 - \mathbf{x}_{N+1})$
    si $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{N+1})$ **alors**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
      Continuer
    fin si
  Étirement
    si $f(\mathbf{x}_r) < f(\mathbf{x}_1)$ **alors** ▷ Le meilleur trouvé par réflexion
      $\mathbf{x}_e = \mathbf{x}_0 + \gamma(\mathbf{x}_r - \mathbf{x}_0)$
      si $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ **alors**
        $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_e$
        Continuer
      sinon
        $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_r$
        Continuer
      fin si
    sinon
      Suite à la contraction
    fin si
  Contraction
    $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_{N+1} - \mathbf{x}_0)$
    si $f(\mathbf{x}_c) < f(\mathbf{x}_{N+1})$ **alors**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}_c$
      Continuer
    sinon
      Suite à la réduction
    fin si
  Réduction
    $\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$
  jusqu'à $max\|\mathbf{x}_i - \mathbf{x}'_j\| \leq \varepsilon_{\mathbf{x}}$ ou $max\|f(\mathbf{x}_i) - f(\mathbf{x}_j)'\| \leq \varepsilon_f$ ou le nombre d'itération égal n_{max} ▷ Critère de stop
  renvoie $\mathbf{x}_0$
fin Fonction

dans l'espace entrée-état. Si une solution exacte discrète des équations différentielles existe, elle peut aussi être utilisée pour calculer les points de la table, par exemple en l'initialisant par un système linéaire approché. Grâce à des mécanismes d'apprentissage, on peut aussi imaginer que ce modèle peut être appris à partir des données mesurées du système. Dans la formulation entrée-état, la commande a besoin d'un observateur pour reconstruire l'état du système. Cette commande utilise un algorithme d'optimisation pour chercher la valeur de la commande qui peut être remplacé par d'autres algorithmes.

1.2 Commande en mode entrée-sortie

1.2.1 Modèle de prédiction en mode entrée-sortie

Comme évolution de la commande, on se propose d'utiliser un modèle de prédiction construit sous forme entrée-sortie ; dans ce cas là, la commande n'a pas besoin d'observateur. Une des opérations importantes dans ce cadre est de conserver une mémoire des sorties et des entrées afin de les utiliser conjointement avec les valeurs de l'instant k. La construction sous cette forme se fait en temps discret comme décrit avec l'équation 1.10. Cette forme, connue sous le nom de NARX[4] (Leontaritis et Billings (1985a), Leontaritis et Billings (1985b)), est un modèle auto-régressif qui permet de trouver la sortie suivante à partir des entrées-sorties précédentes et des entrées-sorties courantes.

$$Y_{k+1} = F(Y_k, \cdots, Y_{k-N_1}, U_k, \cdots, U_{k-N_2}) \tag{1.10}$$

Cette fonction peut être apprise utilisant les mesures (réelles ou simulées) $Y_k, \cdots, Y_{k-N_1}, U_k, \cdots, U_{k-N_2}$. Les données collectés sont organisées en une chaine $Y_k, \cdots, Y_{k-N_1}, U_k, \cdots, U_{k-N_2}, Y_{k+1}$ comme illustré sur la figure 1.8. Ce modèle peut être

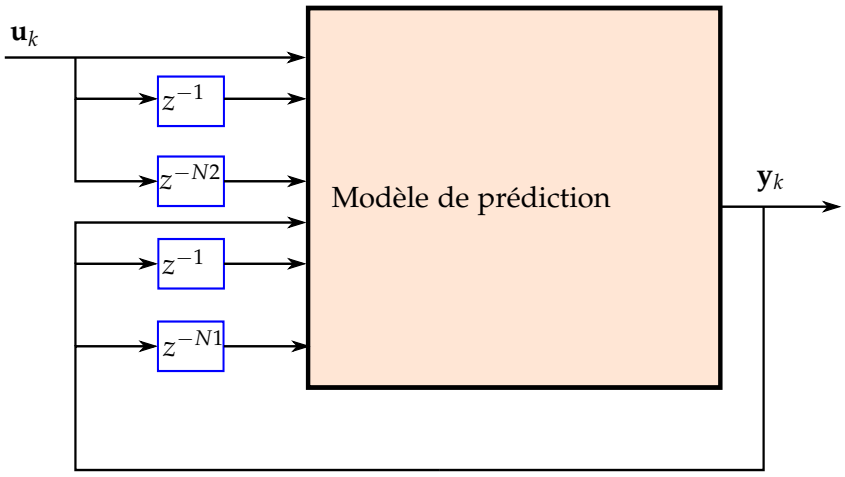


FIGURE 1.8 – *Le modèle de prédiction en mode entrée-sortie*

4. NARX : nonlinear autoregressive exogenous model

obtenu par discrétisation, identification ou apprentissage. Dans certains cas, comme les systèmes linéaires, à partir du modèle d'état on peut trouver le modèle entrée-sortie par analyse mathématique. Mais, la plupart des systèmes réels sont non linéaires, l'obtention du modèle entrée-sortie devient difficile. Cela restreint l'obtention du modèle à l'utilisation des techniques d'identification ou d'apprentissage. Les techniques d'identification donneront un modèle entrée-sortie approximé par une fonction de transfert à coefficients constants. La technique de régression donnera ce modèle par une fonction d'approximation en utilisant par exemple un réseau de neurones. Dans ce travail on se focalisera sur les techniques d'apprentissage présentées dans le chapitre suivant.

1.2.2 Schéma d'une commande en mode entrée-sortie

Le schéma de la commande est modifié comme indiqué sur la figure 1.9. Dans cette

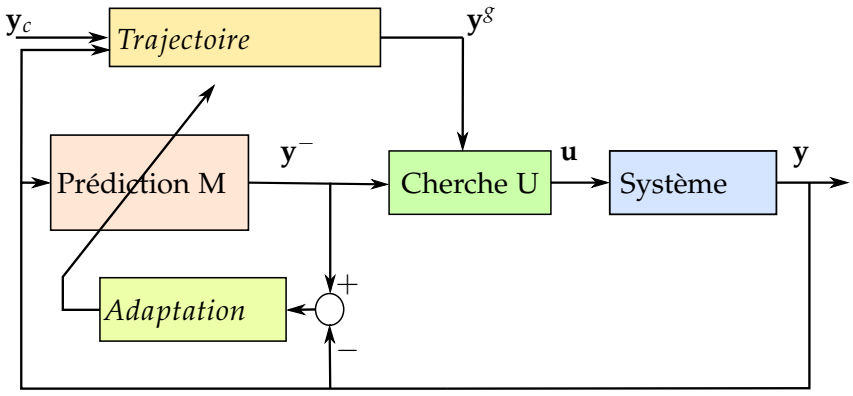


FIGURE 1.9 – *Le schéma de la commande en mode entrée-sortie*

évolution de la commande, les mesures sont de la même façon utilisées pour calculer la valeur des entrées à appliquer. L'algorithme d'optimisation présenté fonctionne sur cette commande, les données de la table n'ayant pas de signification à priori. Donc, état ou données retardées sont utilisables indifféremment. Pour représenter finement le système dynamique, il faut un nombre de délais N_1, N_2 le plus grand possible. Le nombre de dimensions de la table de prédiction peut donc être plus grand que pour le modèle entrée d'état, ce qui peut être gênant au delà d'une certaine valeur. Comme la commande basée sur le modèle d'état, le bloc "Trajectoire" joue le rôle d'un filtre à priori pour la consigne à suivre.

Conclusion du chapitre

La commande par modèle tabulé de comportement pour la commande non linéaire d'un système a été présentée. En considérant la conception de l'approche entrée-état, une évolution de la commande basée sur le modèle entrée-sortie a été proposée. Ceci permet d'éviter la construction d'un observateur et d'utiliser directement les données issues des capteurs. La table de prédiction basée sur le modèle entrée-sortie est difficile à obtenir par analyse mathématique. De même, une méthodologie de simulation initialisant une nouvelle condition initiale, pour chaque point de la table, avant de simuler un seul pas de temps est difficile, car le modèle de simulation n'a généralement pas une forme entrée-sortie. Une stratégie d'apprentissage sera proposée dans le chapitre suivant pour obtenir ce modèle.

Chapitre 2

Stratégie d'apprentissage

Dans ce chapitre, on va présenter quelques approches dans le domaine de l'apprentissage statistique. Les méthodes classiques seront introduites pour expliciter leur mise en œuvre. En second lieu, on proposera une interprétation de ces concepts afin de les adapter à la méthodologie de commande proposée dans ce mémoire. En dernier lieu, une version de la commande permettant l'apprentissage du modèle sera proposée.

Sommaire

2.1 Apprentissage de modèle

Pour apprendre le modèle d'un système en général ou plus précisément le modèle décrit par l'équation (1.10), la littérature sur l'apprentissage statistique expose nombre de méthodes efficaces. Une fonction arbitraire peut être approchée par une fonction interpolée grâce à des techniques d'approximation. Parmi les différentes méthodes, on s'intéressera ici aux réseaux à fonctions de base radiales RBF, aux méthodes à noyaux, au processus gaussien.

2.1.1 Réseaux à fonctions de base radiales RBF

Une fonction de base radiale est une fonction scalaire qui est symétrique autour d'un point $\mathbf{x}_{ci}$. $\phi_i(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{x}_{ci}\|)$, avec $\|\cdot\|$ la norme d'un vecteur (Powell (1987), Broomhead et Lowe (1988)). Cela peut par exemple être une fonction gaussienne :

$$\phi_i(\mathbf{x}, \sigma_i) = \exp(-\|\mathbf{x} - \mathbf{x}_{ci}\|^2/\sigma_i^2)$$

avec σ_i la largeur et $\mathbf{x}_{ci}$ le centre de la fonction. Un réseau RBF est une somme linéaire de fonctions de base radiales.

$$y(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \sigma_i) = \mathbf{w}^T \Phi(\mathbf{x})$$

avec le vecteur des poids $\mathbf{w} = [w_1, \cdots, w_n]^T$ et $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \cdots, \phi_n(\mathbf{x})]^T$. On peut considérer un réseau RBF comme un réseau de neurones à trois couches Viennet (2006). La figure 2.1 représente un réseau RBF avec une sortie y. Pour les systèmes avec plusieurs sorties, on utilise une combinaison des réseaux RBFs, chaque sortie étant associée à un réseau.

Connaissant les centres $\mathbf{x}_{ci}$ et les largeurs $\sigma_i, i = \overline{1, n}$, le problème d'approximation est un problème supervisé. Soient les vecteurs entrés $\mathbf{x}_j, j = \overline{1, m}$ et ses valeurs sorties $y_j, j = \overline{1, m}$, on détermine le vecteur des poids $\mathbf{w}$ qui minimise la somme suivante :

$$\sum_{j=1}^{m} (y_j - \mathbf{w}^T \Phi(\mathbf{x}_j))^2 + \lambda \mathbf{w}^T \mathbf{w} \tag{2.1}$$

avec $\Phi(\mathbf{x}_j) = [\phi_1(\mathbf{x}_j), \cdots, \phi_n(\mathbf{x}_j)]^T$. La condition de minimisation donne :

$$\lambda \mathbf{w} - \sum_{j=1}^{m} (y_j - \mathbf{w}^T \Phi(\mathbf{x}_j)) \Phi(\mathbf{x}_j) = 0$$

ou de façon équivalente :

$$(\Phi\Phi^T + \lambda \mathbf{I}_n)\mathbf{w} = \sum_{j=1}^{m} y_j \Phi(\mathbf{x}_j)$$

avec $\Phi = [\Phi(\mathbf{x}_1), \cdots, \Phi(\mathbf{x}_m)]$ et $\mathbf{y} = [y_1, \cdots, y_m]^T$, d'où la valeur de $\mathbf{w}$:

$$\mathbf{w} = (\Phi\Phi^T + \lambda \mathbf{I}_n)^{-1} \Phi \mathbf{y} \tag{2.2}$$

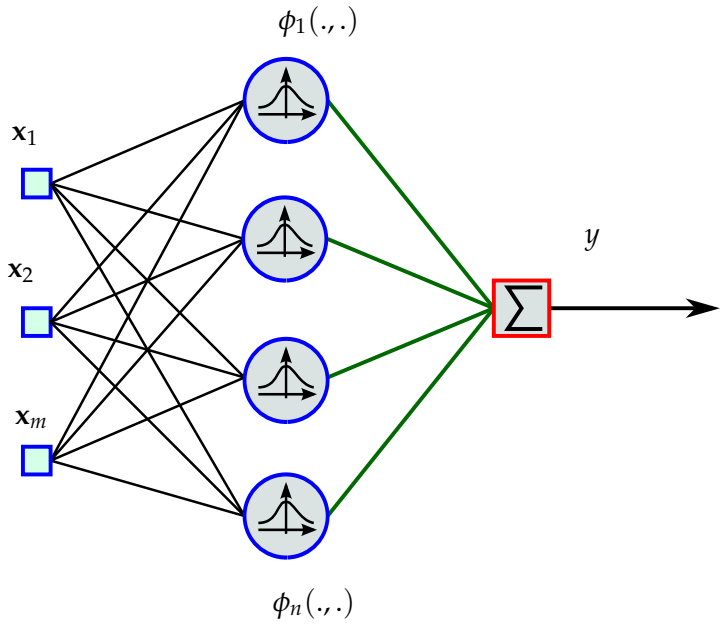


FIGURE 2.1 – *Réseau RBF*

En outre c'est une forme quadratique, la solution peut-être donc approchée en utilisant la méthode de descente de gradient ou d'autres algorithmes d'optimisation. Des centres et des largeurs optimales peuvent-être aussi calculés par un algorithme d'optimisation. Un exemple de régression par RBF pour la fonction $y = x \sin x$ est donné sur la figure 2.2.

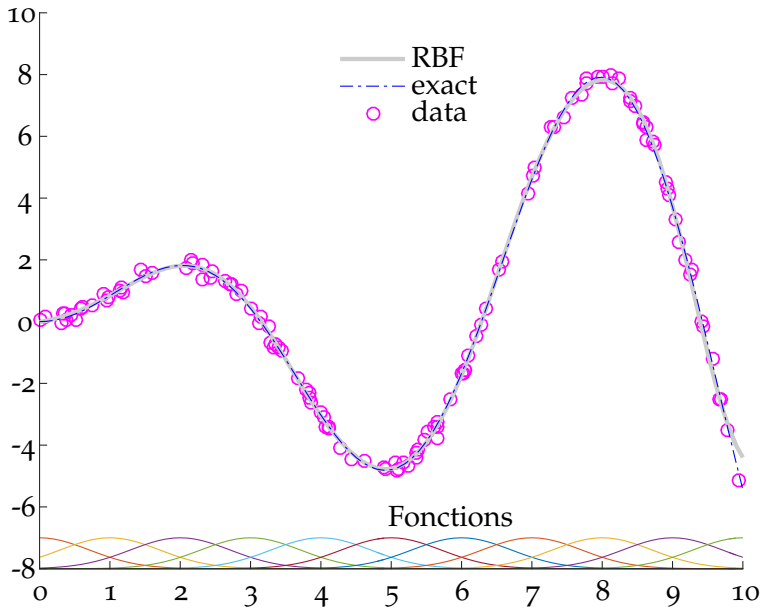


FIGURE 2.2 – *Régression par RBF avec* $y = x \sin x$. *Les centres sont* $x_{ci} = i, i = \overline{0,10}, \lambda = 0.02$ *et* $\sigma = 1$.

2.1.2 Méthode à noyaux

Un noyau sur l'ensemble χ est une fonction k symétrique : $\chi \times \chi \mapsto \mathbb{R}$.

$$\forall(\mathbf{x}, \mathbf{x}') \in \chi^2 : k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$$

Un noyau est dit défini positif s'il satisfait pour tout $n \in \mathbb{N}$, $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n) \in \chi^n$ et $(a_1, a_2, \cdot, a_n) \in \mathbb{R}^n$, la somme suivante non-négative :

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i a_j k(\mathbf{x}_i, \mathbf{x}_j))$$

De façon équivalente, la matrice de Gram suivante est définie positive :

$$\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=\overline{1,n}}$$

Un noyau défini positif est également appelé un noyau de Mercer. Les noyaux les plus connus et utilisés sont les suivants :

— Produit scalaire : $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$

— Polynômial : $k_m(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^m = (1 + \mathbf{x}_i^T \mathbf{x}_j)^m$

— Gaussien : $k_m(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$

Théorème du représentant (Schölkopf et al. (2001))

Il existe une fonction $\phi : \chi \mapsto \mathbb{R}^m$ telle que la fonction noyau est le produit scalaire de deux fonctions de bases.

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

Ce théorème permet d'approximer une fonction par un réseau de noyaux.

Régression par noyau

On détermine le vecteur des poids $\mathbf{w} = [w_1, \cdots, w_n]^T$ qui minimise la somme suivante :

$$\sum_{j=1}^{m}(y_j - \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}_j))^2 + \lambda \sum_{i=1}^{n} w_i^2 \tag{2.3}$$

$$\sum_{j=1}^{m}(y_j - \mathbf{w}^T \Phi(\mathbf{x}_j))^2 + \lambda \mathbf{w}^T \mathbf{w} \tag{2.4}$$

avec $\Phi(\mathbf{x}_j) = [\phi_1(\mathbf{x}_j), \cdots, \phi_n(\mathbf{x}_j)]^T$,$\Phi = [\Phi(\mathbf{x}_1), \cdots, \Phi(\mathbf{x}_m)]$ et $\mathbf{y} = [y_1, \cdots, y_m]^T$ par la solution simple suivante :

$$\mathbf{w} = (\Phi\Phi^T + \lambda \mathbf{I}_n)^{-1}\Phi\mathbf{y} \tag{2.5}$$

On note que cette solution est la même que celle de la méthode basée sur un réseau de fonctions à bases radiales, à la différence près de la fonction de base ϕ. En vérifiant que

$$(\Phi\Phi^T + \lambda \mathbf{I}_n)^{-1}\Phi = \Phi(\Phi^T\Phi + \lambda \mathbf{I}_m)^{-1}$$

on obtient

$$\mathbf{w} = \Phi(\Phi^T\Phi + \lambda\mathbf{I}_m)^{-1}\mathbf{y} \tag{2.6}$$

Donc, la valeur de prédiction de la sortie correspondant à l'entrée $\mathbf{x}$ est calculée par :

$$y(\mathbf{x}) = \mathbf{w}^T\Phi(\mathbf{x}) = \mathbf{y}^T(\Phi^T\Phi + \lambda\mathbf{I}_m)^{-1}\Phi^T\Phi(\mathbf{x})$$

$$y(\mathbf{x}) = \mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I}_m)^{-1}\mathbf{k}(\mathbf{x})$$

où la matrice $\mathbf{K} = \Phi^T\Phi$ est une matrice de Gram et le vecteur $\mathbf{k}(\mathbf{x}) = \Phi^T\Phi(\mathbf{x})$. C'est la méthode à noyau classique nommée KRR (Kernel Ridge Regression). En remplaçant $\mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I}_m)^{-1} = \alpha^T$ avec $\alpha = (\mathbf{K} + \lambda\mathbf{I}_m)^{-1}\mathbf{y}$, la valeur reconstruite au point $\mathbf{x}$ est donc calculée par

$$y(\mathbf{x}) = \alpha^T\mathbf{k}(\mathbf{x})$$

Cette forme vérifie le théorème du représentant.

Régression par noyau en ligne

La méthode précédente prend en compte toutes les données dans la solution. Cela pose un problème de calculabilité quand le nombre des données est grand. Des techniques ont été proposées pour élaguer des données afin d'obtenir un dictionnaire compact comme le critère de cohérence, la dépendance linéaire approximative, etc. Dans Engel et al. (2004), un algorithme a été proposé pour approximer une fonction en utilisant un noyau. Un vecteur d'entrée $\mathbf{x}_t$ est pris en compte si la condition linéaire suivante est satisfaite :

$$\min_{\mathbf{w}} \|\mathbf{w}^T\Phi(\mathbf{x}) - \Phi(\mathbf{x}_t)\| > \nu$$

où ν est un seuil positif. De plus, si le nombre des éléments dans le dictionnaire est supérieur à une valeur, la nouvelle entrée ne sera pas prise en compte. Des algorithmes ont été proposés pour contourner ce problème et respecter la nature non-linéaire du système. Par exemple, dans Van Vaerenbergh et al. (2010), un élément est supprimé et remplacé si la dimension est supérieure à un maximum. Dans Van Vaerenbergh et al. (2006), une fenêtre est appliquée pour déterminer les éléments du dictionnaire.

Un autre problème à maîtriser est la dimension de la matrice. Si la dynamique du système est non linéaire, il faut utiliser un dictionnaire de dimension N suffisamment grand. Donc l'inversion de la matrice reste une opération lourde. Heureusement, les données des mesures utilisées sont actualisées périodiquement ce qui permet l'utilisation d'algorithmes itératifs pour calculer et mettre à jour récursivement le vecteur des poids $\mathbf{w}$.

2.1.3 Processus gaussien

Un processus gaussien est un processus aléatoire où toutes les lois sont gaussiennes. Chaque loi de distribution est associée avec le centre $\mu(\mathbf{x})$ et la covariance est une fonction à noyau $k(\mathbf{x}, \mathbf{x}')$. Dans Rasmussen et Williams (2005) l'algorithme d'apprentissage

suivant a été proposé. Soit les vecteurs colonnes des entrées $\mathbf{x}_i$ et les sorties y_i avec $i = \overline{1,n}$. La matrice des entrées a priori $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ est utilisée comme un dictionnaire initial. On souhaite établir une relation entre les entrées et les sorties sous forme $y_i = f(\mathbf{x}_i) + \epsilon_i$, où ϵ_i est un bruit gaussien de moyenne nulle et de variance σ_n^2, noté par $\mathcal{N}\left(0, \sigma_n^2\right)$. La sortie prédite est $\mathbf{y} = \mathcal{N}\left(0, \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)$. Dans ce cas, un noyau gaussien est souvent utilisé :

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_s^2 \left(-\frac{1}{2} (\mathbf{x}_p - \mathbf{x}_q)^T \mathbf{W} (\mathbf{x}_p - \mathbf{x}_q) \right)$$

où $\mathbf{W}$ est la matrice de poids. La distribution de la sortie prédite $f(\mathbf{x}_*)$ pour un point $\mathbf{x}_*$ et les mesures $\mathbf{y}$ est donnée par :

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \mathbf{k}(\mathbf{X}, \mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*, \mathbf{X}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right) \tag{2.7}$$

où $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X})$, $\mathbf{k}_* = \mathbf{k}(\mathbf{X}, \mathbf{x}_*)$
La valeur moyenne est la meilleure valeur prédite de la sortie $f(\mathbf{x}_*)$. La solution s'écrit (Rasmussen et Williams (2005)) :

$$\begin{cases} f(\mathbf{x}_*) = \mathbf{k}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} = \mathbf{k}_*^T \alpha, \\ V(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_* \end{cases} \tag{2.8}$$

avec $V(\mathbf{x}_*)$ la covariance estimée et le vecteur des coefficients $\alpha = (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$. Les paramètres $[\sigma_n^2, \sigma_s^2, \mathbf{W}]$ sont calculés en se basant sur des entrées à priori $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ et $\mathbf{y} = [y_1, y_2, ..., y_n]^T$ par une méthode de quasi-Newton (Rasmussen et Williams (2005)). Le but est de maximiser la fonction de vraisemblance :

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2} \mathbf{y}^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log(|\mathbf{K} + \sigma_n^2 \mathbf{I}|) - \frac{n}{2} \log 2\pi$$

En pratique, la matrice inverse est calculée en se basant sur une décomposition de Cholesky.

Régression en ligne par processus gaussien

Pour une utilisation en ligne, il faut adapter les méthodes précédentes. On trouve dans la littérature différentes évolutions. Le but est de réduire la taille du dictionnaire et de limiter les opérations mathématiques à réaliser sur une période de temps. Les éléments $\mathbf{X}$ contenus dans le dictionnaire permettent de calculer la matrice $\mathbf{K}$ et le vecteur des coefficients α. Quand une nouvelle mesure est disponible, un algorithme choisit les éléments les plus importants à garder et recalcule $\mathbf{K}$ et α (Engel et al. (2002), Nguyen-Tuong et Peters (2011)). Une autre version utilisant un modèle local a été proposée dans Nguyen-Tuong et al. (2009). Kocijan et Murray-Smith (2005) proposent d'utiliser le modèle de régression pour faire de la commande prédictive, mais après l'apprentissage, le processus gaussien est fixé et il est utilisé comme un modèle de prédiction.

2.2 Commande par apprentissage

Dans le domaine de la commande par apprentissage, la programmation dynamique et l'apprentissage par renforcement utilisent des méthodes similaires : itération de valeur, itération de politique, recherche directe de la politique. La programmation dynamique est largement basée sur un modèle. Au contraire, l'apprentissage par renforcement peut se faire indépendemment d'un modèle. La politique ou la fonction de valeur sont construites en se basant sur les données de mesure. Si par ailleurs on connait bien le système et qu'on peut le modéliser suffisamment précisément, le modèle pourra être utilisé pour accélérer l'étape de recherche du processus d'apprentissage. On va tout d'abord décrire ce qu'est un processus de décision markovien. On donnera ensuite des éléments sur la commande par apprentissage sans modèle ou basée modèle. On peut trouver un état de l'art sur les méthodes de la commande par apprentissage dans Kober et Peters (2012).

2.2.1 Processus de décision markovien

Un processus de décision markovien est défini comme un quadruplé $(\mathcal{S}, \mathcal{U}, \mathcal{P}, \mathbb{R})$ dans Buşoniu et al. (2010)) ou un quintuplé $(\mathcal{S}, \mathcal{U}, \mathcal{P}, \mathbb{R}, \gamma)$ dans Lagoudakis et Parr (2003). A l'étape k, l'état du système est $\mathbf{x}_k \in \mathcal{S}$. Sous la commande appliquée $\mathbf{u}_k \in \mathcal{U}$, à l'étape suivante, l'état du système sera $\mathbf{x}_{k+1} \in \mathcal{S}$ et l'environnement donnera une récompense $r_{k+1} \in \mathbb{R}$. La probabilité que le processus arrive à l'état $\mathbf{x}_{k+1}$ est décrite par la fonction de transition d'états $\mathcal{P}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}) \in [0, 1]$. Ainsi, l'état suivant dépend seulement de l'état courant $\mathbf{x}_k$ et de l'action $\mathbf{u}_k$. Cette propriété est dite propriété de Markov.

La figure 2.3 décrit les composants du processus de décision markovien et leurs rôles. Le contrôle envoie une valeur $\mathbf{u}_k$ et observe l'état du processus $\mathbf{x}_k$. Il reçoit une récompense r_{k+1}. Selon la récompense reçue, il améliore la valeur envoyée au processus à l'étape suivante.

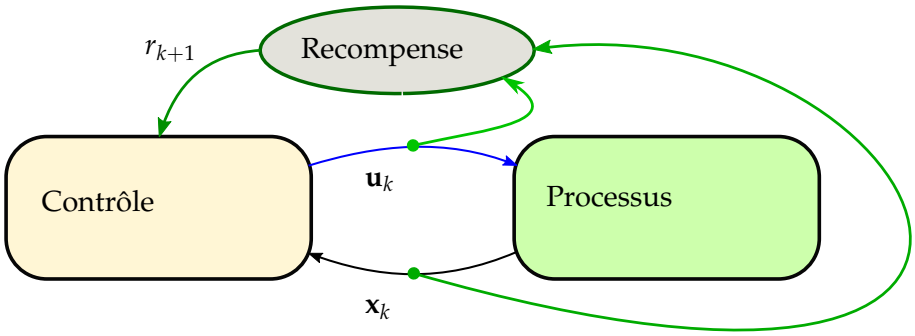


FIGURE 2.3 – *Le schéma d'un processus de décision markovien*

La récompense future

La récompense future pour l'état initial $\mathbf{x}_0$ est définie par la somme suivante, avec γ un coefficient de décroissance :

$$R(\mathbf{x}_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1}$$

En conséquence, la relation des récompenses sur deux pas est $R(\mathbf{x}_t) = r_{t+1} + \gamma R(\mathbf{x}_{t+1})$ avec le coefficient γ dans l'intervalle $[0, 1]$.

La politique de la commande

La politique de la commande d'un processus de décision markovien est définie par une fonction $\pi : \mathcal{S} \mapsto \mathcal{P}(\mathcal{U})$, avec $\mathcal{P}(\mathcal{U})$ la probabilité dans l'espace des entrées $\mathcal{U}$. Le but de la commande est de trouver la politique optimale ou gloutonne qui maximise la fonction de coût suivante :

$$J_{\pi *} = \max_{\pi} J_{\pi} = \max_{\pi} E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} \right] \tag{2.9}$$

avec E l'espérance mathématique. Dans le cas déterministe, on remplace $E(x)$ par x.

La fonction de valeur des états

La fonction de valeur des états pour une politique stationnaire π et la fonction optimale de valeur pour la politique optimale π_* sont définies par :

$$V^{\pi}(\mathbf{x}) = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} |_{\mathbf{x}_0 = \mathbf{x}} \right]$$

$$V^{*}(\mathbf{x}) = \max_{\pi} E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} |_{\mathbf{x}_0 = \mathbf{x}} \right]$$

La théorie de la programmation dynamique indique que la fonction optimale de valeur satisfait l'équation de Bellman :

$$V^{*}(\mathbf{x}) = \max_{\mathbf{u}} \left[r(\mathbf{x}, \mathbf{u}) + \gamma E[V^{*}(\mathbf{x}')] \right]$$

La fonction de valeur des états/actions

La fonction de valeur des états/actions est définie par

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} |_{\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = \mathbf{u}} \right]$$

donc, la fonction optimale de valeur des états/actions est définie par :

$$Q^{*}(\mathbf{x}, \mathbf{u}) = \max_{\pi} E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} |_{\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = \mathbf{u}} \right]$$

Cette fonction vérifie l'équation de Bellman :

$$Q^{\pi}(\mathbf{x},\mathbf{u}) = E_{\pi}\left[r(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{u}'} \mathcal{P}^{\pi}(\mathbf{x},\mathbf{u}',\mathbf{x}')Q^{\pi}(\mathbf{x}',\mathbf{u}')\right]$$

La politique optimale peut donc être facilement exprimée par :

$$\pi^{*}(\mathbf{x}) = \max_{\mathbf{u}} Q^{*}(\mathbf{x},\mathbf{u}) \tag{2.10}$$

2.2.2 Commande par apprentissage sans modèle

L'idée de la commande par apprentissage sans modèle est de trouver la politique optimale de commande pour maximiser la fonction de coût en se basant sur les données mesurées disponibles. Les données observées lors d'essais réels ou en simulation sont sauvegardées dans des épisodes, c'est-à-dire une chaîne de $\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}, r_{k+1}$. Trois approches sont développées pour chercher la politique de commande :

— itération de la politique,

— évaluation de la fonction de valeur,

— recherche direct de la politique.

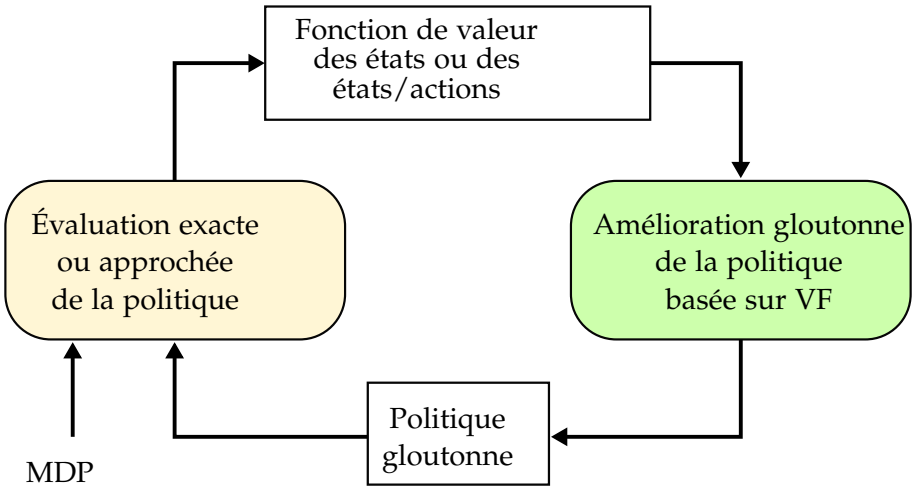


FIGURE 2.4 – *Le schéma d'un algorithme d'itération de la politique PI*

L'itération de la politique est montrée sur la figure 2.4. Les auteurs Lagoudakis et Parr (2003) ont proposé un algorithme pour calculer la politique de commande utilisant la méthode des moindres carrés. La fonction de valeur est approximée en utilisant une architecture linéaire.

$$\hat{Q}^{\pi} = \sum_{j=1}^{k} \phi_j(\mathbf{x},\mathbf{u}) w_j^{\pi} = \Phi(\mathbf{x},\mathbf{u})\mathbf{w}^{\pi} \tag{2.11}$$

Dans cette architecture, les fonctions de base $\Phi(\mathbf{x},\mathbf{u})$ sont fixées arbitrairement. Pour tout ensemble $(\mathbf{x},\mathbf{u})$, l'équation de Bellman donne la solution exacte pour la fonction de

valeur d'une politique π par

$$Q^{\pi}(\mathbf{x},\mathbf{u}) = r(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{x}'} \mathcal{P}^{\pi}(\mathbf{x},\mathbf{u}',\mathbf{x}') \sum_{\mathbf{u}'} \pi(\mathbf{u}',\mathbf{x}') Q^{\pi}(\mathbf{x}',\mathbf{u}') \tag{2.12}$$

La terme de droite appelé opération de Bellman est défini par

$$(T_{\pi}Q)(\mathbf{x},\mathbf{u}) = r(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{x}'} \mathcal{P}^{\pi}(\mathbf{x},\mathbf{u}',\mathbf{x}') \sum_{\mathbf{u}'} \pi(\mathbf{u}',\mathbf{x}') Q^{\pi}(\mathbf{x}',\mathbf{u}')$$

et sous la forme matricielle

$$Q^{\pi} = \mathbf{R} + \gamma \mathbf{P} \Pi_{\pi} Q^{\pi}$$

ou avec l'opération de Bellman

$$Q^{\pi} = T_{\pi} Q$$

On constate que Q^{π} est invariant par rapport à l'opération de Bellman. L'opération de Bellman est une fonction de contraction, elle a un seul point fixe. De plus, la solution analytique est de la forme :

$$(\mathbf{I} - \gamma \mathbf{P} \Pi_{\pi}) Q^{\pi} = \mathbf{R} \tag{2.13}$$

En grande dimension le calcul de la matrice inverse prend du temps. Des méthodes récentes proposent la solution approchée :

$$\hat{Q}^{\pi} \approx \mathbf{R} + \gamma \mathbf{P} \Pi_{\pi} \hat{Q}^{\pi}$$

En remplaçant $\hat{Q}^{\pi} = \Phi \mathbf{w}^{\pi}$, le vecteur des paramètres est solution de l'équation

$$(\Phi - \gamma \mathbf{P} \Pi_{\pi} \Phi) \mathbf{w}^{\pi} \approx \mathbf{R}$$

La méthode basée sur la projection permet de formuler la solution par

$$\Phi^{T} \Delta_{\mu} (\Phi - \gamma \mathbf{P} \Pi_{\pi} \Phi) \mathbf{w}^{\pi} \approx \Phi^{T} \Delta_{\mu} \mathbf{R}$$

avec Δ_{μ} la matrice diagonale des poids. Sous la forme matricielle, le vecteur $\mathbf{w}^{\pi}$ est la solution de l'équation ci-dessous

$$\mathbf{A} \mathbf{w}^{\pi} \approx \mathbf{b} \tag{2.14}$$

Finalement, l'algorithme 3 est utilisé pour déterminer la matrice $\mathbf{A}$, le vecteur $\mathbf{b}$ afin d'obtenir $\mathbf{w}^{\pi}$.

Une fois $\mathbf{w}^{\pi}$ calculé pour la politique arbitraire, la politique optimale sera trouvée par la stratégie d'itération de la politique montrée sur la figure 2.4. L'algorithme complet de la méthode LSPI est donné dans l'algorithme 4. LSTD-Q est répété jusqu'à ce que le critère d'arrêt soit satisfait, soit le vecteur $\mathbf{w}$ inchangé.

D'autres algorithmes LSTD-Q, dits optimaux et basés sur un modèle du système ont été également présentés par Lagoudakis et Parr (2003). Cette méthode est souvent utilisé hors ligne pour déterminer la politique optimale à partir de données collectées. La phase en ligne permet ensuite d'évaluer la politique trouvée en l'appliquant sur le système

Algorithme 3 LSTD-Q

1: **Fonction** LSTD-Q(D, k, Φ, γ, π)

Entrées:

//D : Les données

//k : Nombre des fonctions de base

//Φ : Fonctions de base

//γ : Coefficient dans [0,1)

//π : Politique courante

2: $\tilde{\mathbf{A}} \leftarrow \mathbf{0}$ ▷ Matrice de kxk

3: $\tilde{\mathbf{b}} \leftarrow \mathbf{0}$ ▷ Vecteur de $kx1$

4: **pour tout** $(\mathbf{x}, \mathbf{u}, r, \mathbf{x}') \in D$ **faire**

5: $\tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}} + \Phi(\mathbf{x}, \mathbf{u}) \left(\Phi(\mathbf{x}, \mathbf{u}) - \gamma\Phi(\mathbf{x}', \pi(\mathbf{x}'))\right)^T$

6: $\tilde{\mathbf{b}} \leftarrow \tilde{\mathbf{b}} + \Phi(\mathbf{x}, \mathbf{u})r$

7: **fin pour**

8: $\tilde{w}^\pi \leftarrow \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$

9: **renvoie u**

10: **fin Fonction**

Algorithme 4 LSPI

1: **Fonction** LSPI($D, k, \Phi, \gamma, \varepsilon_k, \pi_0$)

Entrées:

//D : Les données

//k : Nombre des fonctions de base

//Φ : Fonctions de base

//γ : Coefficient dans [0,1)

//ε : Critère d'arrêt

//π_0 : Politique initiale

2: $\pi' \leftarrow \pi_0$ ▷ Initialisation

3: **répéter**

4: $\pi \leftarrow \pi'$ ▷ $\mathbf{w} \leftarrow \mathbf{w}'$

5: $\pi' \leftarrow$ LSTD-Q(D, k, Φ, γ, π)

6: **jusqu'à** $\pi \approx \pi'$ ▷ $\|\mathbf{w} - \mathbf{w}'\| \leq \varepsilon$

7: **renvoie u**

8: **fin Fonction**

réel. Dans cette méthode, les fonctions de base sont des fonctions non linéaires de l'état et de l'entrée. L'approximation est habituellement linéaire mais elle peut être changée. Par exemple dans Jung et Polani (2007), Xin et al. (2007), la méthode d'approximation utilisée est basée sur les fonctions à noyaux afin de développer la méthode KLPSI pour l'itération de la politique.

L'évaluation d'itération de la politique en ligne $SARSA$ a été proposée par Rummery et Niranjan (1994). La politique est choisie de manière gloutonne. La commande essaie d'explorer l'environnement en générant une valeur de commande aléatoirement. Des méthodes pour accélérer le temps d'exécution en ligne $SARSA - \lambda$ ont été proposées par Singh et Sutton (1996).

La recherche directe de la politique requiert du temps d'exécution. Dans le cas déterministe, le coût total pour la recherche exhaustive est proportionnelle à $|\mathcal{U}|^{|\mathcal{S}|}|\mathcal{S}|$ (voir Buşoniu et al. (2010)). Cette méthode est donc souvent utilisée dans le cas où le nombre des états et des entrées est faible.

2.2.3 Commande par apprentissage basée sur le modèle

La connaissance du système peut accélérer le processus d'apprentissage par planification et simulation. La relation entre l'apprentissage, la plannification et la commande est donnée par Sutton (1998) et représentée sur la figure 2.5 . A partir des données ex-

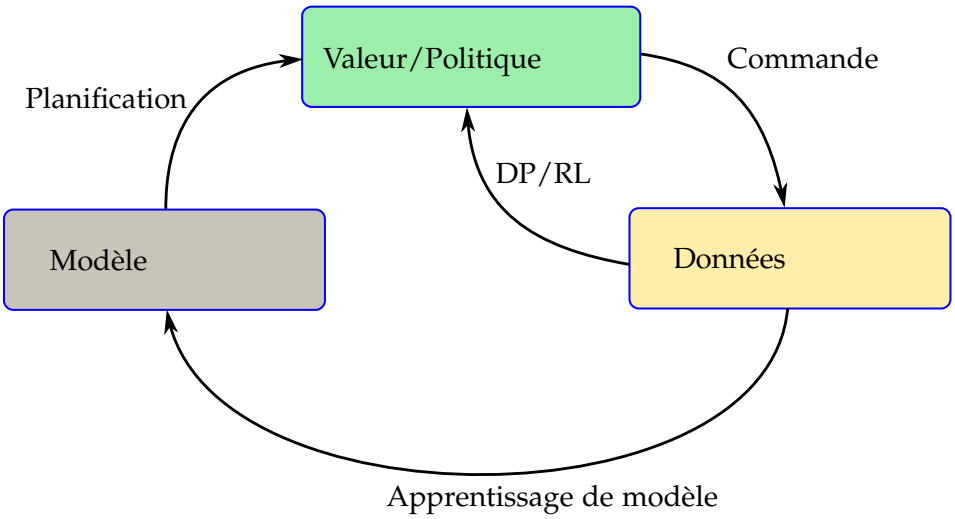


FIGURE 2.5 – *La relation entre l'apprentissage, la plannification et la commande*

périmentales, une stratégie d'apprentissage est utilisée pour reconstruire le modèle du système. Ce modèle sera utilisé pour améliorer la politique. La structure DYNA-Q sur la figure 2.6 permet d'utiliser la connaissance a priori du système (Sutton (1998)). L'algorithme 5 est initialisé avec une politique et un modèle initiaux. Pour chaque étape,

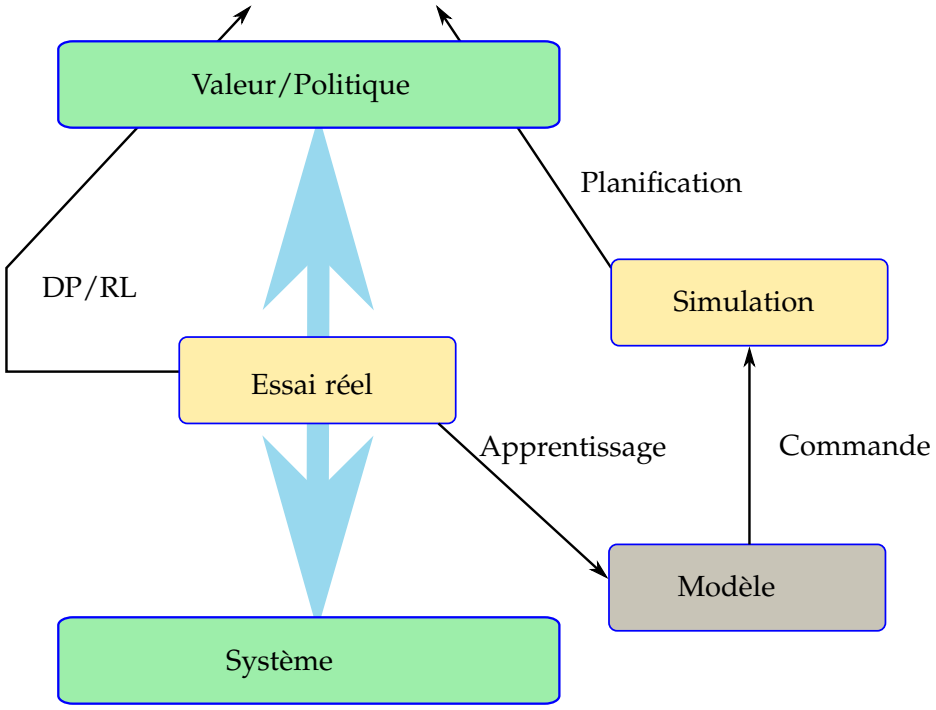


FIGURE 2.6 – *Commande par apprentissage basée sur modèle ou DYNA-Q*

l'entrée est sélectionnée grâce à une politique gloutonne, c'est-à-dire que l'entrée maximise la politique. Puis, la politique est mise à jour. L'état et la récompense observés sont utilisés pour l'apprentissage du modèle $Model(\mathbf{x}, \mathbf{u})$ du système. N-étapes pour Q-planning seront faites en se basant sur ce modèle pour accélérer l'obtention de la politique grâce au modèle appris.

2.3 Une commande par apprentissage de modèle

Les méthodes d'apprentissage dans la section précédente donnent directement la valeur de commande. Par rapport à la structure de commande décrite dans ce travail au chapitre 1, une combinaison entre l'apprentissage de modèle et la commande basée sur le modèle tabulé de comportement sera utilisée. L'idée proposée est d'utiliser l'apprentissage de modèle pour modifier ou construire la table de prédiction.

2.3.1 Correction itérative basée sur la fonction gaussienne

En considérant la structure des données qui constituent le modèle décrit au chapitre 1, on va choisir de modifier directement et localement la table de prédiction. La figure 2.7 montre la stratégie d'apprentissage. Une fonction *Correction* est ajoutée pour modifier la table. Cette fonction utilise l'erreur entre la sortie mesurée et la sortie prédite.

Algorithme 5 DyNa-Q

Fonction DYNA-Q
$Q(\mathbf{x}, \mathbf{u})$ et $Model(\mathbf{x}, \mathbf{u})$: la politique et le modèle initiaux
$Stop = 0$
  tantque $!Stop$ **faire**
    $\mathbf{x} \leftarrow \mathbf{x}_k$ ▷ État courant
    $\mathbf{u} \leftarrow \varepsilon - gloutonne(\mathbf{x}, Q)$ ▷ La politique gloutonne
    Appliquant $\mathbf{u}$, observer $\mathbf{x}'$ et r
    $Q(\mathbf{x}, \mathbf{u}) \leftarrow Q(\mathbf{x}, \mathbf{u}) + \alpha[r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}') - Q(\mathbf{x}, \mathbf{u})]$
    $Model(\mathbf{x}, \mathbf{u}) \leftarrow \mathbf{x}', r$
    pour $i = 1 : N$ **faire**
      $\mathbf{x} \leftarrow$ état observé précédent aléatoire
      $\mathbf{u} \leftarrow$ entrée précédente aléatoire dans $\mathbf{x}$
      $\mathbf{u}', r \leftarrow Model(\mathbf{x}, \mathbf{u})$
      $Q(\mathbf{x}, \mathbf{u}) \leftarrow Q(\mathbf{x}, \mathbf{u}) + \alpha[r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}') - Q(\mathbf{x}, \mathbf{u})]$
    fin pour
    $Stop \leftarrow$ entrée
  fin tantque
fin Fonction

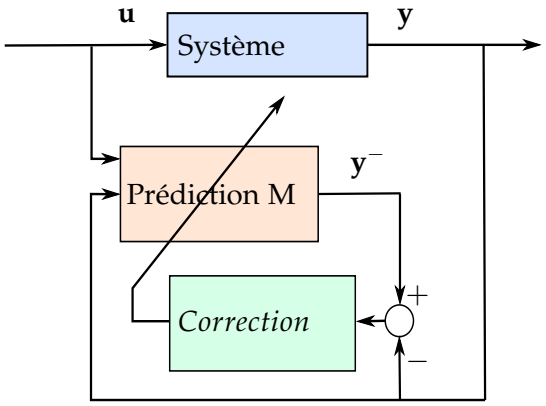


FIGURE 2.7 – *Le schéma d'apprentissage*

Pour commencer, on rappelle qu'une fonction arbitraire peut-être approximée par un réseau de fonctions de base radiales comme suit :

$$y(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) = \mathbf{y}^T (\Phi\Phi^T + \lambda \mathbf{I}_m)^{-1} \Phi^T \Phi(\mathbf{x})$$

ou par un réseau de fonctions à noyaux :

$$y(\mathbf{x}) = \mathbf{y}^T (\mathbf{K} + \lambda \mathbf{I}_m)^{-1} \mathbf{k}(\mathbf{x})$$

On suppose que la nouvelle mesure au point $\mathbf{x}_*$ est $y(\mathbf{x}_*)$ et sa valeur prédite est $\hat{y}(\mathbf{x}_*) = \mathbf{y}^T(\mathbf{K} + \lambda \mathbf{I}_m)^{-1}\mathbf{k}(\mathbf{x}_*)$, on cherche le nouveau vecteur $\mathbf{w}_n$ tel que $\Delta\mathbf{w} = \mathbf{w}_n - \mathbf{w}$ minimise $(\Delta y - \Delta\mathbf{w}^T\Phi(\mathbf{x}_*))^2 + \lambda\Delta\mathbf{w}^T\Delta\mathbf{w}$ avec $\Delta y_* = y(\mathbf{x}_*) - \hat{y}(\mathbf{x}_*)$. La condition est :

$$\left(\Phi(\mathbf{x}_*)\Phi^T(\mathbf{x}_*) + \lambda\mathbf{I}\right)\Delta\mathbf{w} = \Phi(\mathbf{x}_*)\Delta y_*$$

$$\Delta\mathbf{w} = \left(\Phi(\mathbf{x}_*)\Phi^T(\mathbf{x}_*) + \lambda\mathbf{I}\right)^{-1}\Phi(\mathbf{x}_*)\Delta y_*$$

L'erreur de prédiction est calculée par :

$$\Delta y = \Delta\mathbf{w}^T\Phi(\mathbf{x}) = \Delta y_*^T\left(\Phi^T(\mathbf{x}_*)\Phi(\mathbf{x}_*) + \lambda\mathbf{I}\right)^{-1}\Phi^T(\mathbf{x}_*)\Phi(\mathbf{x})$$

Sous la forme de noyau, cette relation devient :

$$\Delta y = \Delta y_*^T\left(k(\mathbf{x}_*, \mathbf{x}_*) + \lambda\mathbf{I}\right)^{-1}k(\mathbf{x}_*, \mathbf{x})$$

En utilisant un noyau gaussien pour $k(\mathbf{x}_*, \mathbf{x})$, cette équation signifie que chaque point de sortie de la grille doit être modifié en fonction de l'erreur de la sortie Δy_*^T et de la distance entre le point observé et le point de la grille. La modification des points de la grille est donc équivalent à la modification itérative des poids $\mathbf{w}$. Les autres algorithmes pour modifier itérativement les poids $\mathbf{w}$ peuvent être trouvés dans la littérature, par exemple Yu et al. (2014) a utilisé la méthode de Levenberg-Marquardt. Si le nombre de points de la grille n'est pas trop grand, on peut modifier tous les points. Pour limiter le temps d'exécution et mettre en œuvre cet algorithme d'apprentissage sans réduire la résolution de la grille dans le cas de dimensions de grille plus grandes, on ne modifiera pas tous les points de la grille. On va se limiter à un voisinage du point observé. Un simple algorithme détermine la zone à modifier en utilisant la distance entre le point lié à la grille et le point observé. Si la distance est inférieure à un rayon R, on va modifier la sortie associée à ce point. On note également que cette méthode ne conserve pas le vecteur $\mathbf{w}$. L'algorithme d'apprentissage est détaillé par l'algorithme 6. Dans le cas où la table de prédiction est construite à partir d'une forme d'état, il requiert un observateur *BuildInput* en ligne 5 pour reconstruire le vecteur d'état $\mathbf{x}$. Pour le modèle entrée-sortie, on utilise directement $\mathbf{x} = [\mathbf{y}_k, \mathbf{y}_{k-1}, \cdots, \mathbf{y}_{k-N_1}, \mathbf{u}_k, \mathbf{u}_{k-1}, \cdots, \mathbf{u}_{k-N_2}]^T$ comme entrée de la table de prédiction. La fonction *PredictOutput* est une fonction d'interpolation (ligne 6). La fonction *CorrectionTable* modifie localement la table de prédiction en se basant sur le point observé $\mathbf{x}$ et l'erreur $\mathbf{e}$ (ligne 8). Les paramètres N_1 et N_2 sont choisis en considérant la complexité de la dynamique du système. Si on se réfère aux travaux de Fliess et Join (2009), M. Fliess (2011) sur la commande sans modèle, il n'est pas nécessaire de connaitre exactement la dimension de chaque dynamique du système mais d'en capturer la partie la plus représentative, très souvent d'ordre 1 pour des systèmes réels classiques.

Algorithme 6 Apprentissage de modèle

Fonction LEARNINGMODEL($\mathbf{G}_0$)
  $\mathbf{G} \leftarrow \mathbf{G}_0$ ▷ Initialisation de la table de prédiction
  Envoyer la commande $\mathbf{u}_k$ au système
  $\mathbf{y} \leftarrow \mathbf{y}_k$ ▷ Mesure la sortie
  $\mathbf{x} \leftarrow$ BUILDINPUT($\mathbf{y}_k, \mathbf{y}_{k-1}, \cdots, \mathbf{u}_k, \mathbf{u}_{k-1}, \cdots$) ▷ Construire le vecteur entré à la table
  $\hat{\mathbf{y}}_{k+1|k} \leftarrow$ PREDICTOUTPUT($\mathbf{x}$) ▷ Prédire la sortie
  $\mathbf{e} \leftarrow \mathbf{y} - \hat{\mathbf{y}}_{k|k-1}$ ▷ Erreur de prédiction
  CORRECTIONTABLE(**x**,**e**) ▷ Modification de table de prédiction
  Sauvegarder $\mathbf{u}_k, \mathbf{y}_k$
  renvoie G
fin Fonction

Exemple 2.1 *Soit la fonction $y = x^2 + \delta_y$ avec δ_y un bruit gaussien.*

La fonction gaussienne utilisée pour modifier la table utilise $\sigma = 0.1$. La figure 2.8 montre la première itération. On a observé un point, son voisinage est modifié. La figure 2.9 montre la fonction apprise après 400 itérations. Elle converge vers la fonction réelle. Les courbes noires présentent les courbes apprises pour chaque itération. L'erreur de prédiction moyenne $MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$ est donnée sur la figure 2.10, elle tend vers zéro.

2.3.2 Évolution de la commande basée sur le modèle tabulé de comportement

En associant l'algorithme d'apprentissage de modèle, la commande peut être modifiée comme sur la figure 2.11. La modification de la table de prédiction peut être effectuée hors ligne sur des données, en ligne sur un système piloté par un moyen tiers (humain ou automatique) voire en même temps que la stabilisation du système si la qualité de la table au démarrage assure que la commande fonctionne. De plus, de par la nature interpolée des données à partir de la grille, il existe toujours une erreur de prédiction qu'il faut continuer à compenser.

On note que l'idée d'utiliser l'apprentissage pour construire le modèle du système est similaire à l'approche développée dans la commande prédictive proposée par Kocijan et Murray-Smith (2005) dans laquelle le modèle du système est représenté par un processus gaussien. La réalisation est par contre différente puisque les supports de l'information sont différents (une grille, en ce qui concerne ce travail).

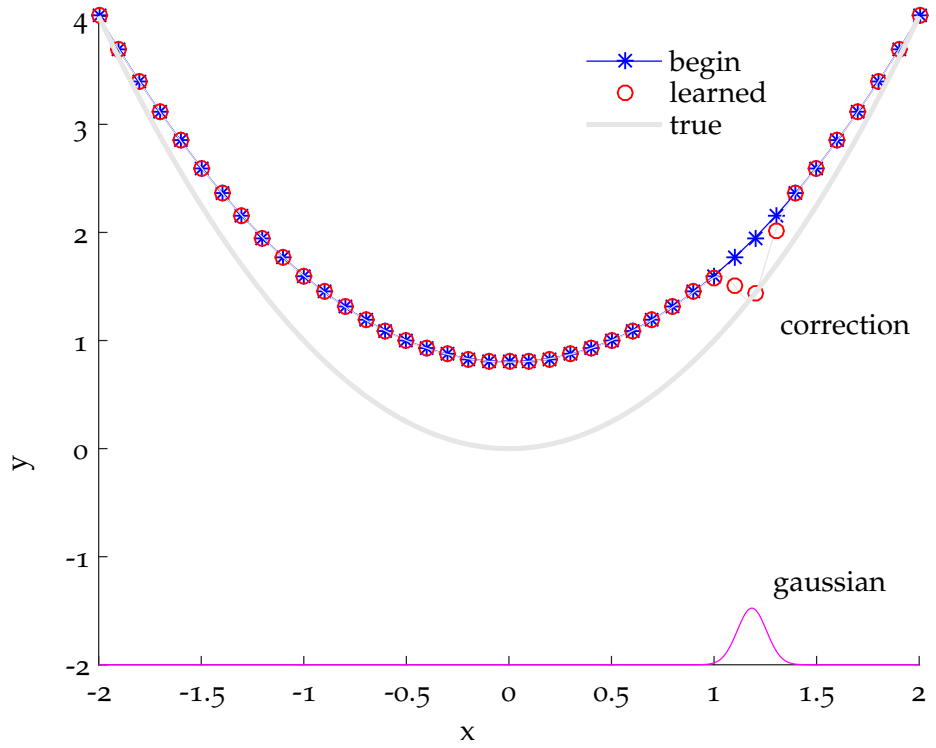


FIGURE 2.8 – *Apprentissage d'une fonction, la première itération*

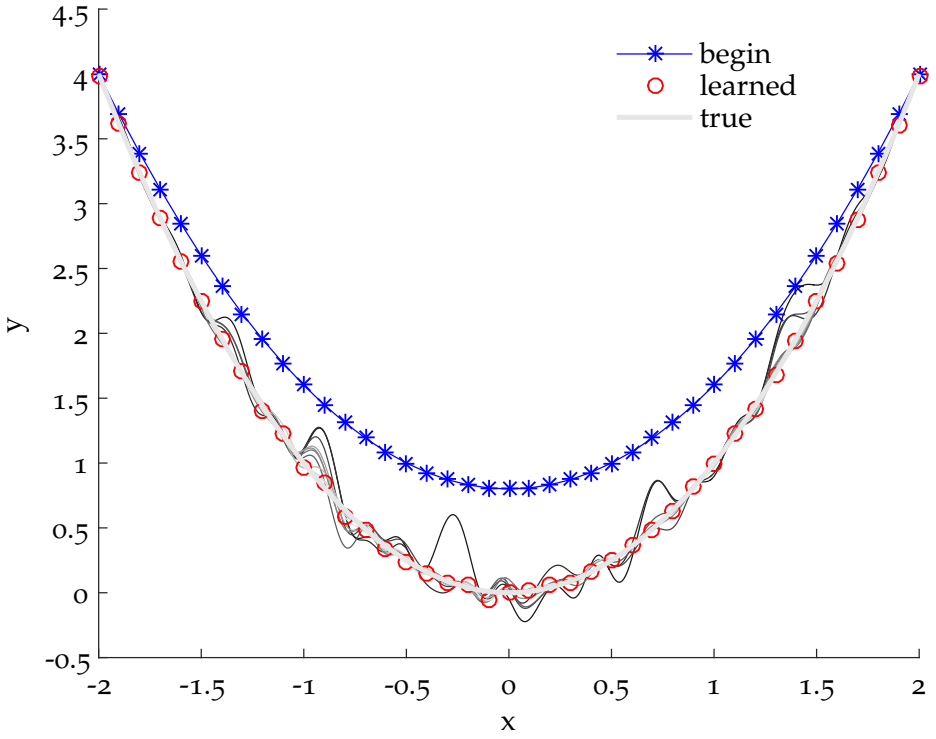


FIGURE 2.9 – *Apprentissage d'une fonction après* 400 *itérations, les courbes mince-solides présentent les courbes apprises pour chaque itération*

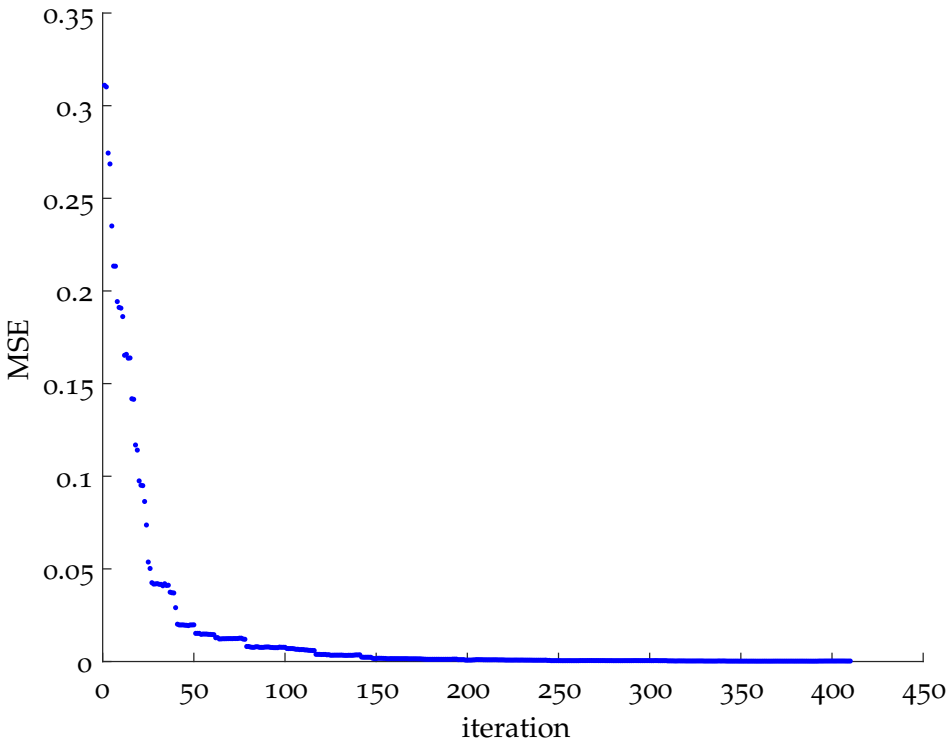


FIGURE 2.10 – *L'évoluation d'erreur de la fonction apprise avec le nombre d'itération, elle devient suffisamment petite*

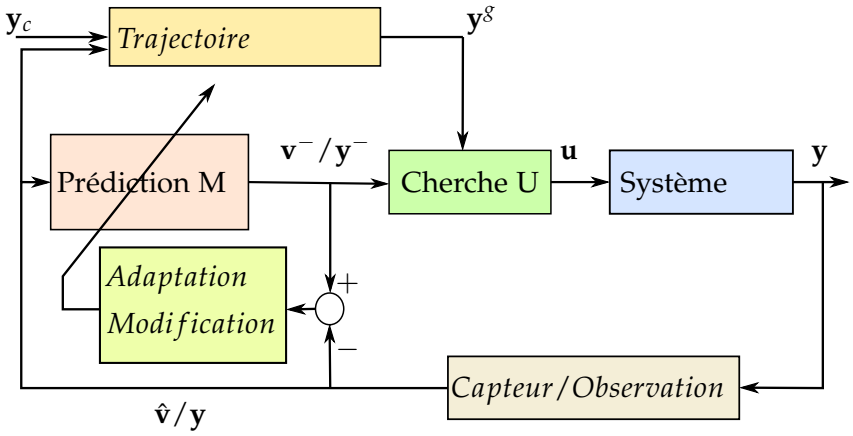


FIGURE 2.11 – *Le schéma d'une commande par apprentissage de modèle*

2.3.3 Exemples

Exemple 2.2 *Soit le système non linéaire d'ordre un*

$$\dot{y} = -a \tanh(y + u^3) \tag{2.15}$$

identique à celui proposé dans Kocijan et Murray-Smith (2005) avec $a = 1$.

La période d'échantillonnage pour discrétiser cette équation est $Ts = 0.25s$. Une table de prédiction est générée utilisant l'intégration numérique de cette équation pour pouvoir comparer avec celle qui sera obtenue par apprentissage. Cette dernière est initialisée à zéro. Après avoir effectué quelques trajectoires par application d'entrées aléatoires, on obtient la table de prédiction représentée sur la figure 2.12. Dans la table de prédiction apprise "Learned", il y a des zones non apprises car la trajectoire parcourue n'est pas passée par ces points. Dans la zone déjà apprise, la table est très proche de la table construite par résolution de l'équation différentielle.

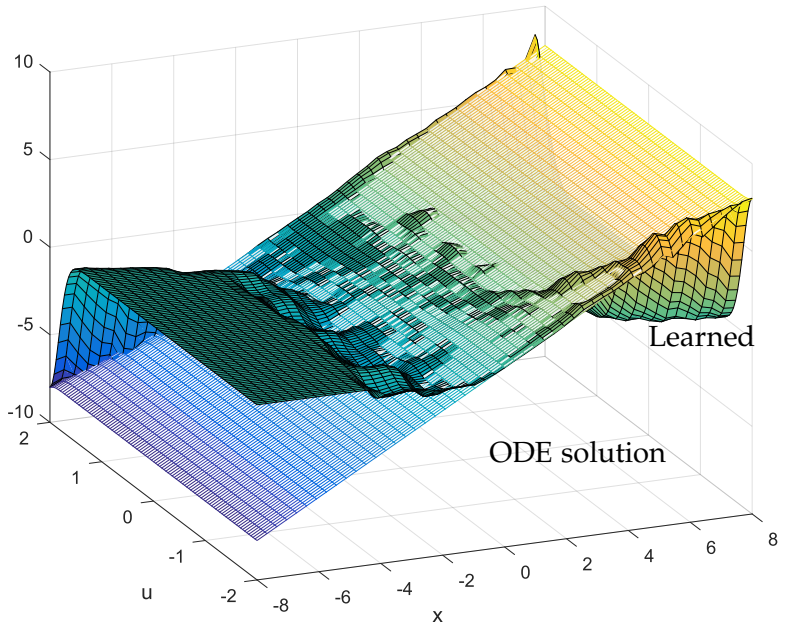


FIGURE 2.12 – *Comparaison entre table de prédiction apprise et simulée*

Les figures 2.13 et 2.14 montrent la stabilisation du système (2.15) en appliquant la table apprise. Dans cet exemple, la trajectoire de référence est un signal sinusoïdal $y = sin(0.1t)$. La table de prédiction a été apprise partiellement mais dans la zone utile pour réaliser la stabilisation du système par rapport à la trajectoire demandée.

Exemple 2.3 *Soit un pendule d'équation dynamique (Buşoniu et al. (2010))*

$$J\ddot{\theta} = mgl \sin(\theta) - b\dot{\theta} + c\Gamma \tag{2.16}$$

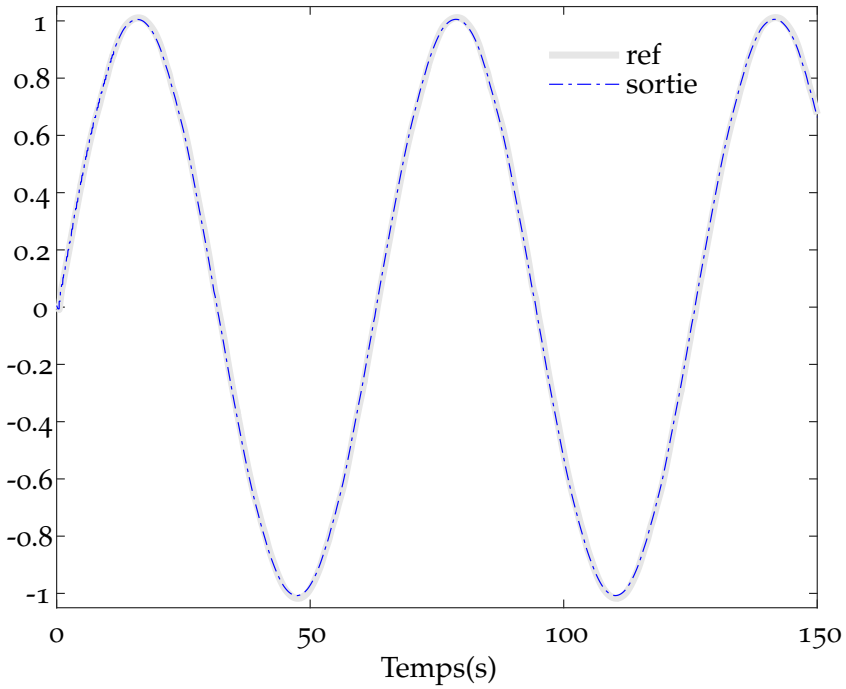


FIGURE 2.13 – *La sortie utilisant la table apprise, le système réel avec* $a = 1.2$

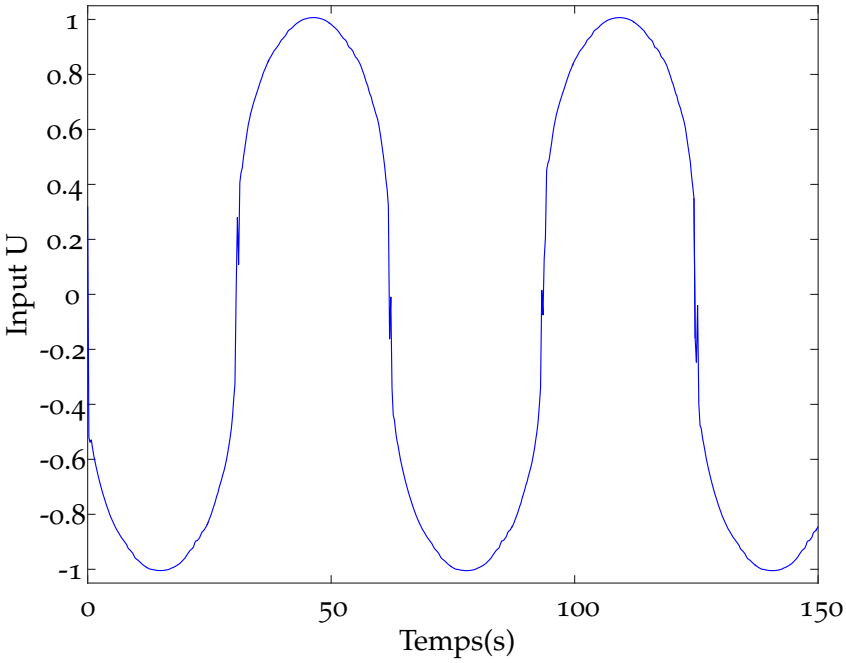


FIGURE 2.14 – *L'entrée utilisant la table apprise, le système réel avec* $a = 1.2$

avec $J = 2.0 \times 10^{-4}\ kg.m^2$, $m = 0.05\ kg$, $l = 0.042\ m$, $b = 3.0 \times 10^{-6}\ Nms.rad^{-1}$, $c = 5.6 \times 10^{-3}\ Nm.V^{-1}$, $g = 9.81\ m.s^{-2}$ *et l'entrée du système est* $\Gamma \in [-3.0, 3.0]\ V$

L'équation du système peut s'écrire sous forme entrée-état comme suit :

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \dfrac{mgl\sin(x_1) - bx_2 + c\Gamma}{J} \end{cases} \tag{2.17}$$

Ce système contient un point d'équilibre instable qui est $[x_1, x_2]^T = [0, 0]^T$. Dans cet exemple, la commande essaie de stabiliser le pendule vers le haut autour de deux points $[\pi/20, 0]$ et $[0, 0]^T$. Des tables de prédiction sont générées en utilisant l'équation d'état avec différentes masses m. On prend $m_1 = m_2 = 0.045\ kg$ pour la première et la deuxième table et $m_3 = 0.05\ kg$ pour la troisième table, avec un temps d'échantillonnage $T_s = 0.005s$. Seule la première commande intègre l'algorithme d'apprentissage en ligne. Ces trois commandes sont appliquées au système dont la masse est $m = 0.05\ kg$. Le temps de simulation est $40s$; on recommence six fois la simulation pour constater l'évolution de la réponse de la commande dont la table est modifiée en ligne.

Variable	Minimum	Maximum	Nombre de points
θ	$-2\pi[rad]$	$2\pi[rad]$	81
$\dot{\theta}$	$-15\pi[rad.s^{-1}]$	$15\pi[rad.s^{-1}]$	6
Γ	$-3V$	$3V$	7

TABLE 2.1 – *Paramètres des variables pour la génération des tables*

Le système de référence interne est un système linéaire du second ordre dont la fonction de transfert est de la forme :

$$H(p) = \frac{\omega_0^2}{p^2 + 2.\zeta_0.\omega_0.p + \omega_0^2}$$

avec $\omega_0 = 10, \zeta_0 = 1$.

Les figures 2.15 et 2.16 montrent qu'à chaque itération la table apprise est améliorée et permet l'obtention d'une meilleure réponse. La performance de la commande tend vers celle de la commande faite en utilisant la table construite avec la masse exacte. L'erreur est réduite à chaque nouvelle itération.

Exemple 2.4 *Soit un système dynamique suivant (Nguyen et al. (2014))*

$$\begin{cases} \ddot{x} &= -u_1 \sin\theta + \delta_w \\ \ddot{z} &= u_1 \cos\theta - 1 \end{cases} \tag{2.18}$$

avec $\theta \in [-\pi/2, \pi/2]$ *et* $u_1 \in [0, 2]$. δ_w *est une perturbation permanente comme le vent.*

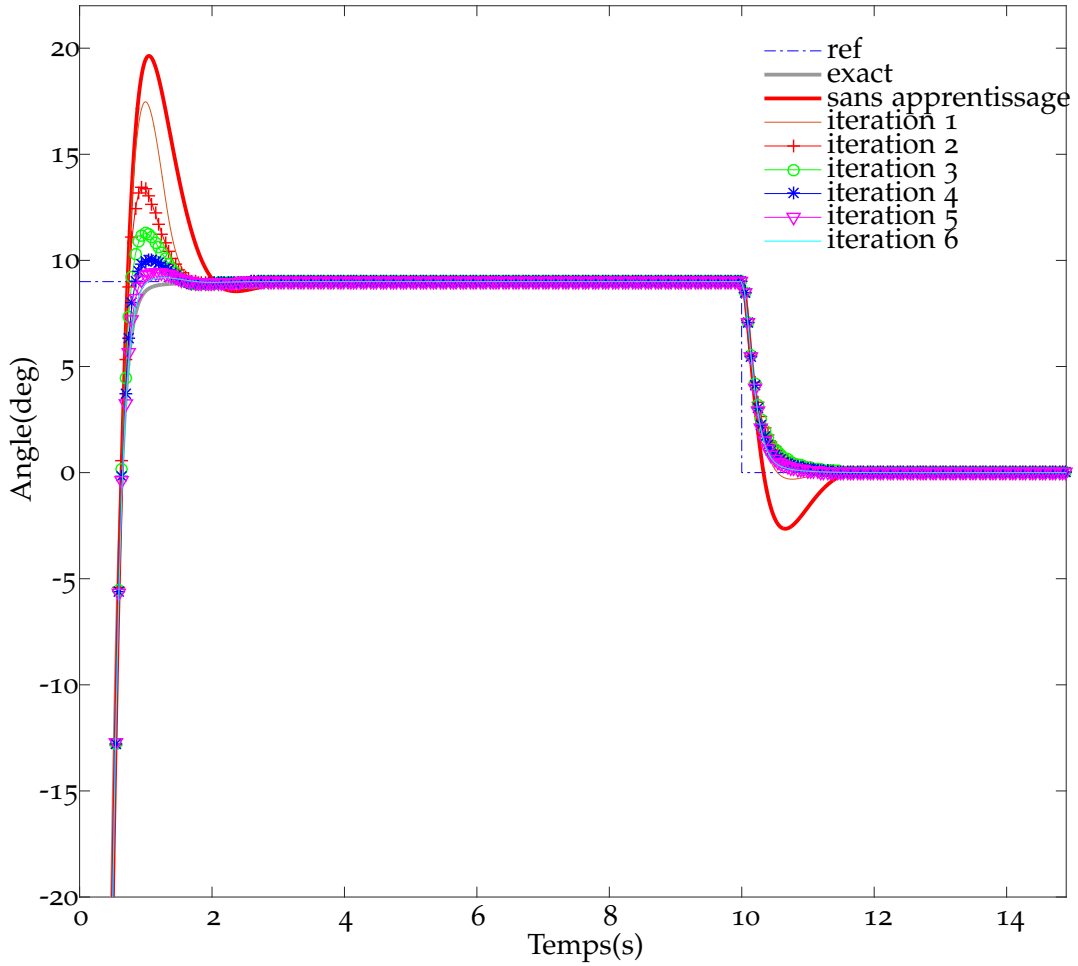


FIGURE 2.15 – *Apprentissage de modèle pour un pendule*

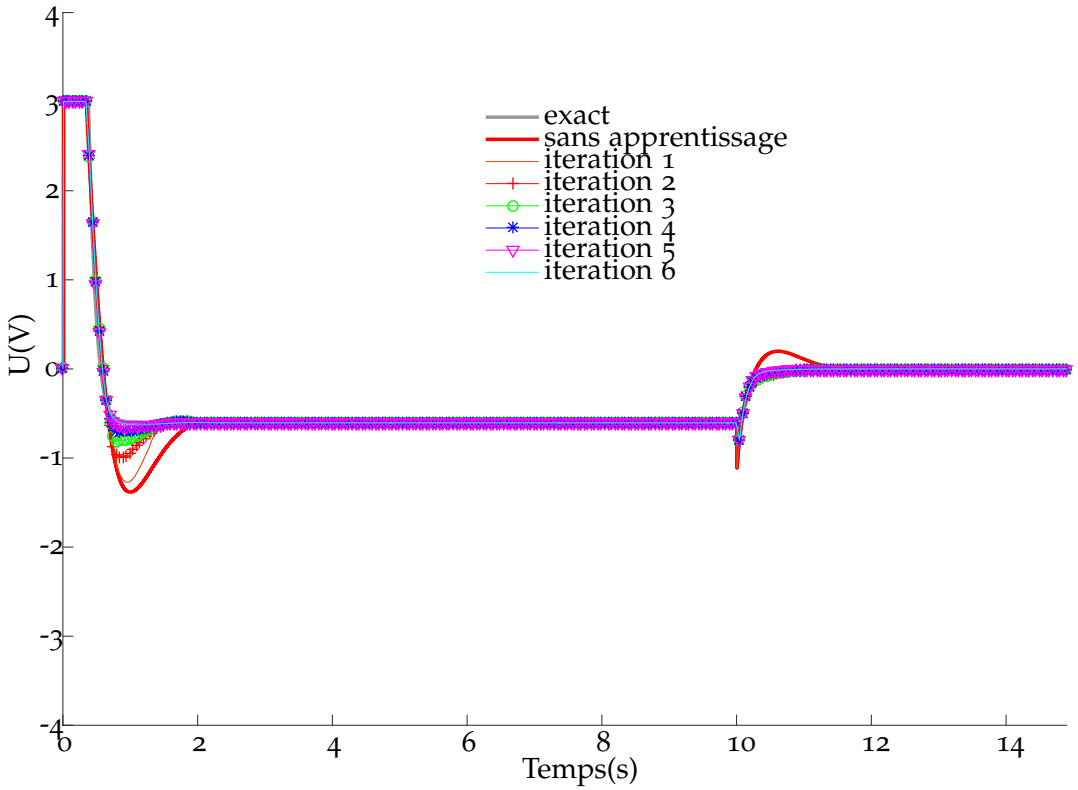


FIGURE 2.16 – *Entrée appliquée avec apprentissage de modèle pour un pendule*

Dans cet exemple, les sorties du système au pas k sont les variations des déplacements sur les axes x et z.

$$\begin{cases} \Delta x_k &= x_{k+1} - x_k \\ \Delta z_k &= z_{k+1} - z_k \end{cases}$$

Les équations ci-dessus peuvent être représentées au temps discret par :

$$\begin{cases} \Delta x_{k+1} &\approx F_1(\Delta x_k, u_{1k}, \theta_k, T_e) \\ \Delta z_{k+1} &\approx F_2(\Delta z_k, u_{1k}, \theta_k, T_e) \end{cases} \tag{2.19}$$

avec T_e la période d'échantillonnage. Dans cette simulation, on prend $T_e = 0.01\ s$. Δx_k et Δz_k sont dans l'intervalle $[-0.02,\ 0.02]$, ce qui veut dire qu'on suppose que la vitesse maximale sur chaque axe est 2.0 m/s. La table de prédiction est initialisée avec 21 points pour x, z et 81 points pour u_1, θ. La table est ensuite modifiée par l'algorithme d'apprentissage. On souhaite stabiliser le système au point $x = 5$, $z = 3$. Pour montrer la capacité de la commande à rejeter les perturbations, après $T = 25s$ une perturbation permanente $\delta_w = -0.2$ sera appliquée. Les figures 2.17, 2.18 et 2.19 présentent le comportement du système sur x et z et le signal de commande. Sous l'effet de la perturbation, ses entrées θ et u_1 ont évolué si rapidement qu'on ne perçoit pas l'effet de la perturbation sur la figure 2.17 .

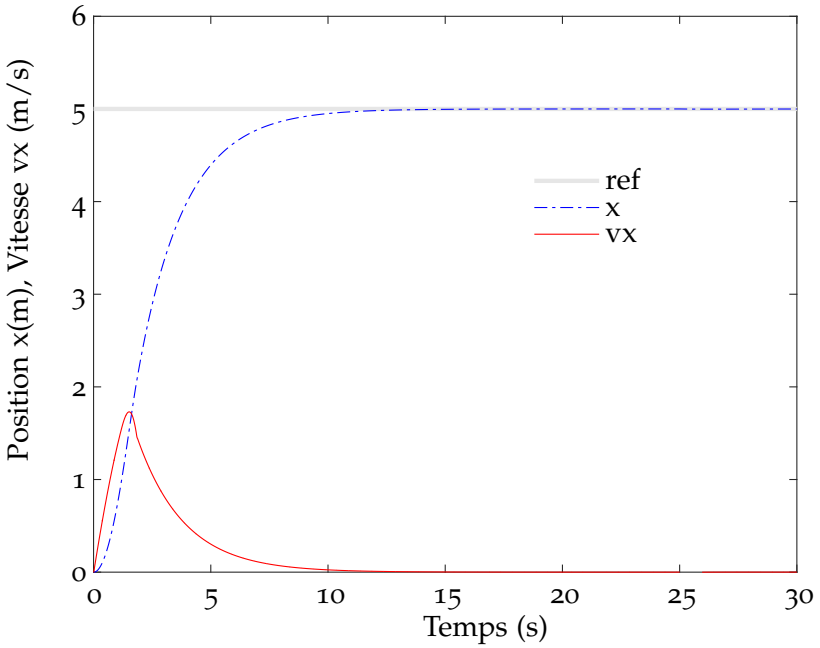


FIGURE 2.17 – *La position stabilisée sur l'axe x*

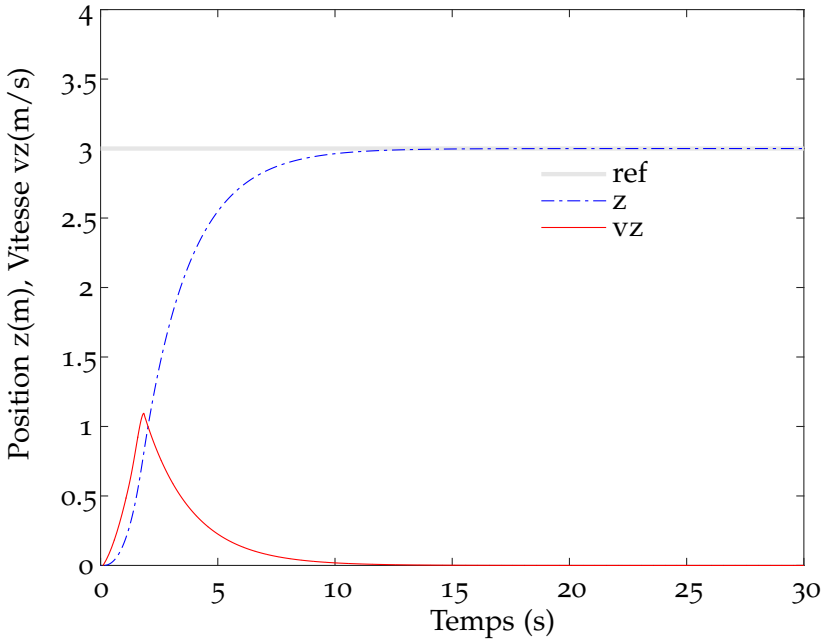


FIGURE 2.18 – *La position stabilisée sur l'axe z*

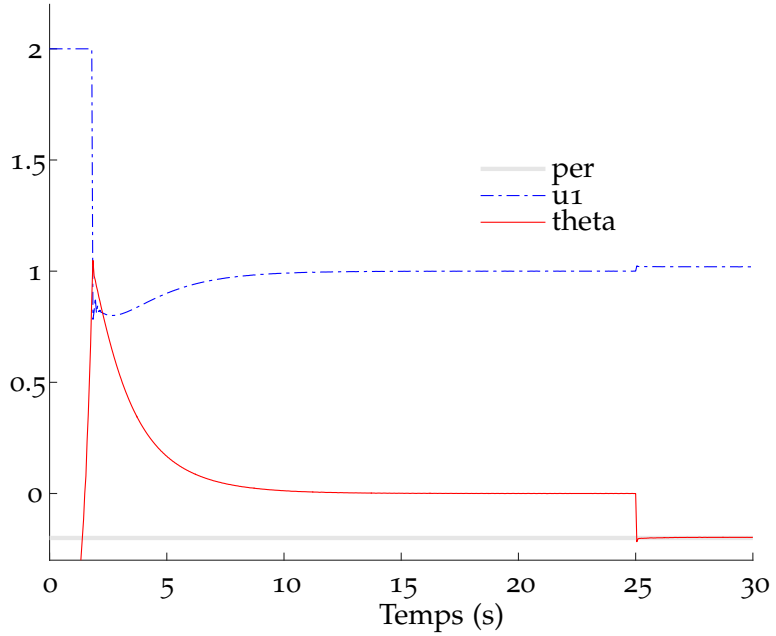


FIGURE 2.19 – *La valeur de commande*

Conclusion du chapitre

Dans le domaine de la commande, l'apprentissage est utilisé pour deux objectifs distinct et complémentaires. La première possibilité est d'utiliser les algorithmes d'apprentissage pour obtenir un modèle du système dynamique. Ce modèle sera utilisé pour construire une loi de commande. La deuxième possibilité est de construire la commande par apprentissage. La politique de commande est trouvée directement en se basant sur les données de façon à ce que que la récompense soit maximisée. Les méthodes d'apprentissage modernes utilisent la technique de l'approximation de fonction pour accélérer le processus d'apprentissage. On trouve de nombreux développements permettant une application en ligne de ces méthodes. Meilleure est la connaissance théorique du système, plus simple est l'émergence d'une politique de la commande. Ce travail s'est focalisé sur l'apprentissage de modèle du système à contrôler. Une stratégie d'apprentissage de modèle tabulé du système dynamique a été proposée. La table est modifiée et sauvegardée directement pour une utilisation au pas suivant. Ainsi la stratégie d'apprentissage proposée se concrétise par un algorithme simple dans le contexte d'un modèle tabulé. La stratégie est donc tout à fait réalisable et utilisable en ligne. Deux points n'ont pas été abordés :

— la question de l'intégrité des données ;

— la méthode de propagation de l'erreur pour corriger la table ne se limite pas à la fonction gaussienne, on peut utiliser également toutes les méthodes étudiées.

Chapitre 3

Application au véhicule aérien

Dans ce chapitre, nous présenterons les applications de la commande proposée sur les véhicules aériens. Le premier modèle utilisé est celui du PVTOL [1]. Le nombre de variables est restreint mais il présente la complexité nécessaire pour vérifier la robustesse de la commande. Ensuite, nous poursuivrons en réalisant la stabilisation d'un problème plus complexe : l'attitude d'un véhicule aérien. A la fin du chapitre, nous montrerons des résultats expérimentaux.

Sommaire

1. Planar Vertical Take-Off and Landing

3.1 Commande d'un PVTOL

Le modèle du PVTOL est représenté sur la figure 3.1. C'est un système sous-actionné qui présente une complexité intéressante pour étudier le problème de la commande. Il existe de nombreux travaux concernant le modèle PVTOL dans la littérature. On trouve par exemple, un algorithme de stabilisation globale étudié dans Fantoni et al. (2002a), Fantoni et al. (2002b), Lopez-Araujo et al. (2010), Fantoni et al. (2003), basé sur une fonction de Lyapunov en prenant en compte la saturation des entrées. Les auteurs Martin et al. (1994) ont utilisé l'approche des systèmes plats pour construire une commande. Le problème de délai des entrées a été traité dans le travail de Francisco et al. (2007). Une commande sans modèle est utilisée dans Riachy et al. (2010). Le contrôle par des techniques de back-stepping pour le PVTOL en tenant compte des composantes aérodynamiques a été proposé dans Wood et al. (2005). Les auteurs Benvenuti et al. (1996) ont présenté trois méthodes pour le suivi de trajectoire : la première méthode utilise le modèle linéarisé, la deuxième utilise la platitude du système et la troisième est basée sur une méthode numérique. Une commande non linéaire pour le système de phase non-minimale en utilisant la technique d'inversion dynamique et l'approche basée sur le critère de Lyapunov a été proposée dans Benosman et al. (2009) pour suivre une trajectoire lisse de la sortie. La commande par retour d'état saturé a été étudiée par Gruszka et al. (2011).

Dans cette partie du document, la commande proposée sera appliquée pour contrôler le PVTOL. La table de prédiction est initialisée en utilisant les équations mathématiques habituelles du système. Elle sera ensuite modifiée en ligne par apprentissage. La forme choisie pour la commande est une forme d'état avec un bloc observation d'état.

3.1.1 Equation dynamique

La figure 3.1 montre le schéma d'un modèle PVTOL simplifié. On donne les notations suivantes :

$[x, z]$	La position de PVTOL
θ	L'angle de roulis du PVTOL
u_1	La force de poussée
u_2	Le couple de rotation
$g = -1$	L'accélération de la gravité
ϵ	Le coefficient de couplage entre deux axes

En appliquant les équations de Newton-Euler on obtient :

$$\begin{cases} \ddot{x} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2 \end{cases} \tag{3.1}$$

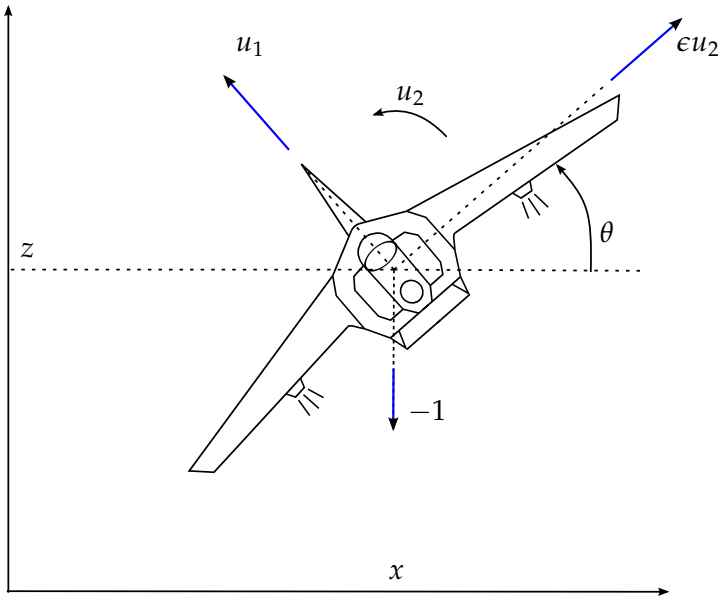


FIGURE 3.1 – *Le modèle simplifié du PVTOL*

Quand ϵ est très petit, ou en réalisant un changement de variables pour utiliser la sortie plate Sira-Ramirez et Fliess (1998), Fliess et al. (1998), on obtient :

$$\begin{cases} \ddot{x} &= -u_1 \sin\theta \\ \ddot{z} &= u_1 \cos\theta - 1 \\ \ddot{\theta} &= u_2 \end{cases} \tag{3.2}$$

3.1.2 Mise en œuvre de l'algorithme de commande

Dans cette section on souhaite élaborer la loi de commande par inversion numérique pour le modèle PVTOL. On introduit δ_w comme une perturbation du système due au vent (ici uniquement sur l'axe horizontal). On utilise l'équation (3.3) comme connaissance a priori du système pour construire des tables de comportements. En fonction de l'objectif de sortie spécifié, on cherche par l'algorithme d'optimisation, l'algorithme 1, présenté dans le premier chapitre, la meilleure commande permettant d'atteindre l'objectif à un pas d'échantillonnage pour le système :

$$\begin{cases} \ddot{x} &= -u_1 \sin\theta + \epsilon u_2 \cos\theta + \delta_w \\ \ddot{z} &= u_1 \cos\theta + \epsilon u_2 \sin\theta - 1 \\ \ddot{\theta} &= u_2 \end{cases} \tag{3.3}$$

Le PVTOL est un système sous-actionné dont la variable interne est l'angle θ, qu'il nous faut calculer pour piloter les positions. On va utiliser un modèle intermédiaire dans ce but. On commence par utiliser u_1 et θ comme entrées du système qui devient alors :

$$\begin{cases} \ddot{x} &= -u_1 \sin\theta \\ \ddot{z} &= u_1 \cos\theta - 1 \end{cases} \tag{3.4}$$

Variable	Minimum	Maximum	Nombre de points
$\dot{x}$	$-4[m.s^{-1}]$	$4[m.s^{-1}]$	11
$\dot{z}$	$-4[m.s^{-1}]$	$4[m.s^{-1}]$	11
θ	$-\pi[rad]$	$\pi[rad]$	41
$\dot{\theta}$	$-6.0[rad.s^{-1}]$	$6.0[rad.s^{-1}]$	11
u_1	$0[N]$	$4[N]$	11
u_2	$-40[Nm]$	$40[Nm]$	11

TABLE 3.1 – *Paramètres des variables*

La table 3.1 donne les paramètres retenus pour réaliser les simulations. Le système de référence sur chaque axe est défini comme un système linéaire du second ordre dont la fonction de transfert est de la forme :

$$H(p) = \frac{\omega_0^2}{p^2 + 2.\zeta_0.\omega_0.p + \omega_0^2}$$

Le schéma complet de la commande est donné par la figure 3.2. Dans ce schéma, le bloc *Cherche*$_\theta$ contient la commande pour le système (3.4). Le bloc **B** extrait $\mathbf{y}_\theta = [x, \dot{x}, z, \dot{z}]^T$ à partir du vecteur $\mathbf{y} = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^T$. A partir du calcul partiel qui permet d'extraire θ dans *Cherche*$_\theta$ et des mesures de position, on recommence avec une table qui représente le système complet dans "Prédiction M" afin de calculer les commandes effectives u_1 et u_2.

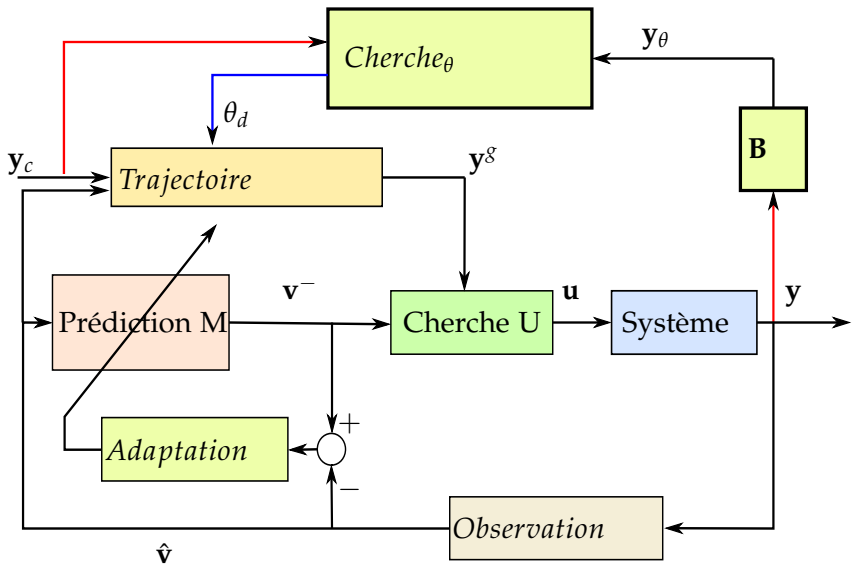


FIGURE 3.2 – *La commande pour le modèle simplifié du PVTOL*

3.1.3 Résultat de simulation

Les dynamiques des références pour les axes x et z sont choisies identiques avec une pulsation propre $\omega_0 = 1\ rad.s^{-1}$ et un coefficient d'amortissement $\zeta_0 = 1$. Pour l'angle θ, on a choisi $\omega_0 = 15\ rad.s^{-1}$ et $\zeta_0 = 1$. La table représentant le système dans "Prédiction M" est construite avec $\epsilon = 0$ et $\delta_w = 0$. Dans la première expérience, on souhaite suivre des références carrées. Dans la simulation, une perturbation constante a été ajoutée $\delta_w = -0.2\ N$ à partir de $t = 15\ s$. Le coefficient ϵ vaut zéro. Les figures 3.3, 3.4 et 3.5 montrent que la trajectoire du PVTOL converge vers la trajectoire de référence (ligne pointillée) sans erreur statique et sans dépassement. La perturbation est rejetée rapidement.

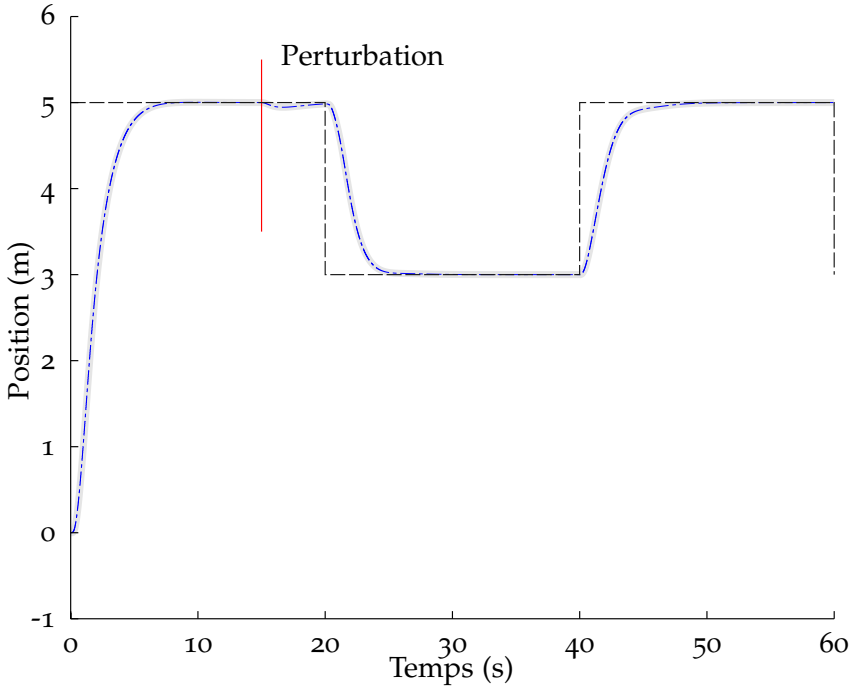


FIGURE 3.3 – *PVTOL, Simulation : Suivi d'un signal de référence carré, position sur l'axe x. Référence interne en pointillé noir, position obtenue en simulation en gris continu*

Dans la deuxième expérience, le système à piloter est modélisé avec $\epsilon = 0.1$, la perturbation ajoutée est $\delta_w = -0.2$ à partir de $t = 25\ s$. La trajectoire demandée est un cercle. Les figures 3.6, 3.7 et 3.8 montrent que la trajectoire du PVTOL converge vers la trajectoire de référence (ligne pointillée). La perturbation est rejetée rapidement. La figure 3.9 montre la trajectoire du PVTOL dans le plan. Le point étoile indique la position au temps $t = 25\ s$. Cette figure montre la robustesse de la commande proposée par rapport à la perturbation permanente, ici δ_w, et à la variation du paramètre du système ϵ.

La figure 3.10 illustre la comparaison entre la commande proposée (nommée Propose) et la commande par saturation (nommée sature) qui est proposée dans la thèse

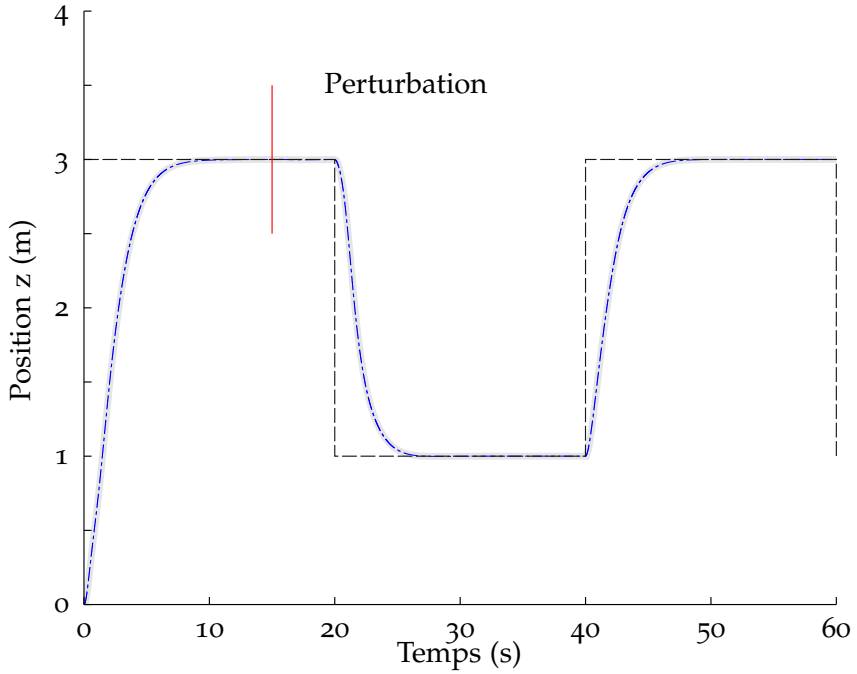


FIGURE 3.4 – *PVTOL, Simulation : Suivi d'un signal de référence carré, la position sur l'axe z. Référence interne en pointillé noir, position obtenue en simulation en gris continu*

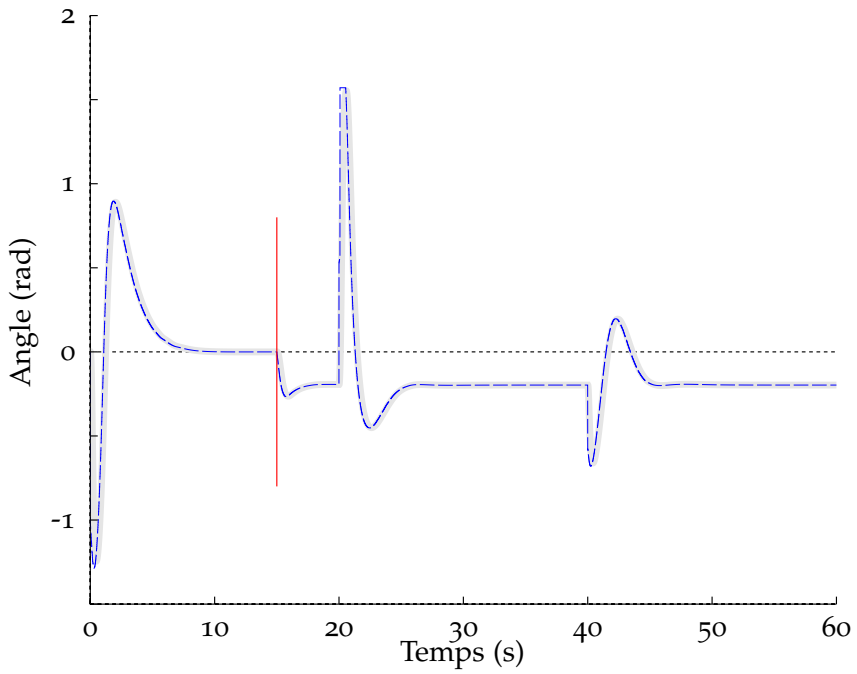


FIGURE 3.5 – *PVTOL, Simulation : L'angle θ et sa référence interne pour un signal de référence carré. Référence interne en pointillé noir, angle obtenu en simulation en gris continu*

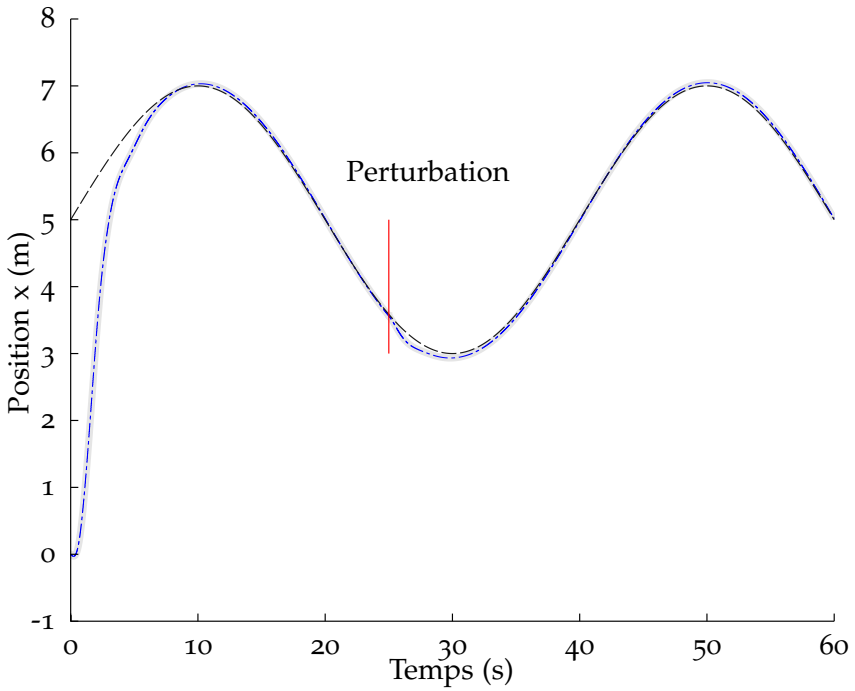


FIGURE 3.6 – *PVTOL, Simulation : Suivi d'un signal de référence sinusoïdale, la position sur l'axe x. Référence interne en pointillé noir, position obtenue en simulation en gris continu*

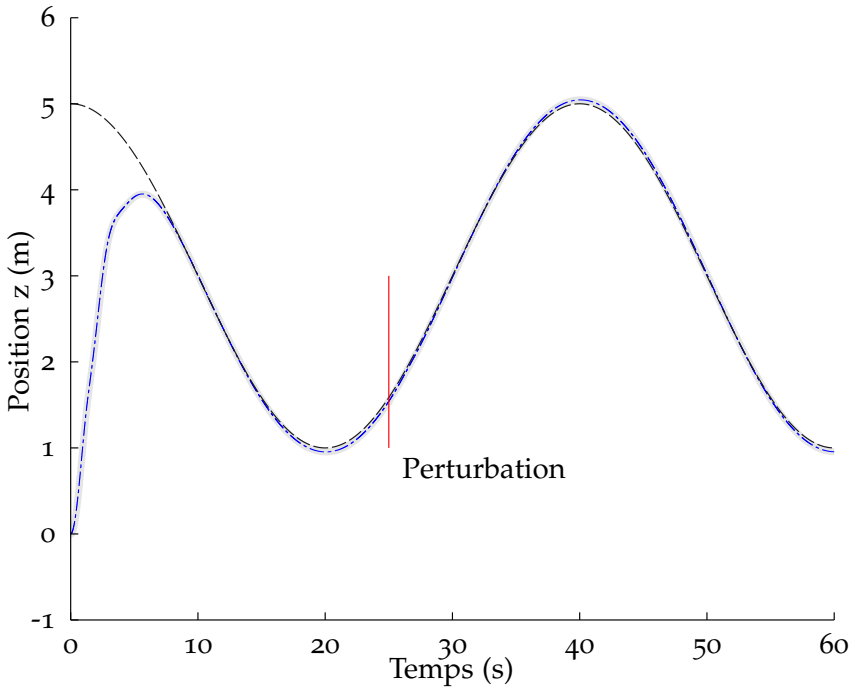


FIGURE 3.7 – *PVTOL, Simulation : Suivi d'un signal de référence sinusoïdale, la position sur l'axe z. Référence interne en pointillé noir, position obtenue en simulation en gris continu*

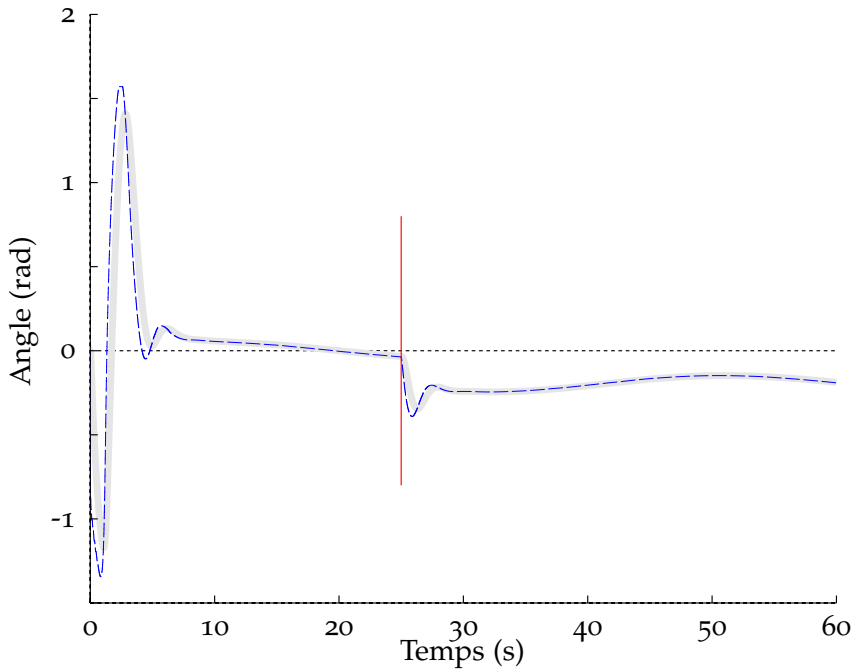


FIGURE 3.8 – *PVTOL, Simulation : L'angle θ et sa référence interne pour un signal de référence sinusoïdal. Référence interne en pointillé noir, angle obtenu en simulation en gris continu*

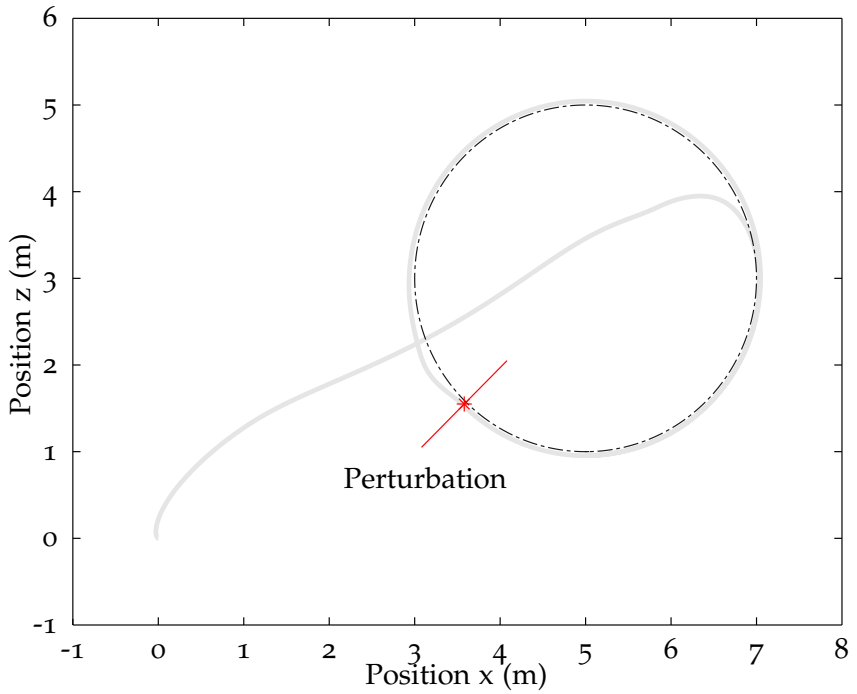


FIGURE 3.9 – *PVTOL, Simulation : Trajectoire de référence et résultat dans le plan Oxz*

Sanahuja (2010)[2]. La deuxième simulation présentée précédemment est refaite pour la commande par saturation. Dans cette commande, les deux entrées sont calculées indépendamment par :

$$u_1 = \frac{g - \sigma_a(k_a(z - z_d)) - \sigma_b(k_b(\dot{z} - \dot{z}_d))}{\cos(\theta)}$$

et

$$u_2 = \frac{\sigma_1(k_1(x - x_d)) + \sigma_r(k_2(\dot{x} - \dot{x}_d)) - \sigma_3(k_3 g\theta) - \sigma_4(k_4 g\dot{\theta})}{g}$$

avec la fonction de saturation définie par $\sigma_b(x) = x$ si $|x| < b$, sinon $\sigma_b(x) = bsign(x)$. Les paramètres pour la commande par saturation sont donnés dans la table 3.2.

b_a	1	b_1	1.36
k_a	1	b_2	1.61
b_b	2	b_3	2.98
k_b	2	b_4	6.55
k_1	0.06	k_3	2.22
k_2	0.52	k_4	2.2

TABLE 3.2 – *Paramètres de la commande par saturation*

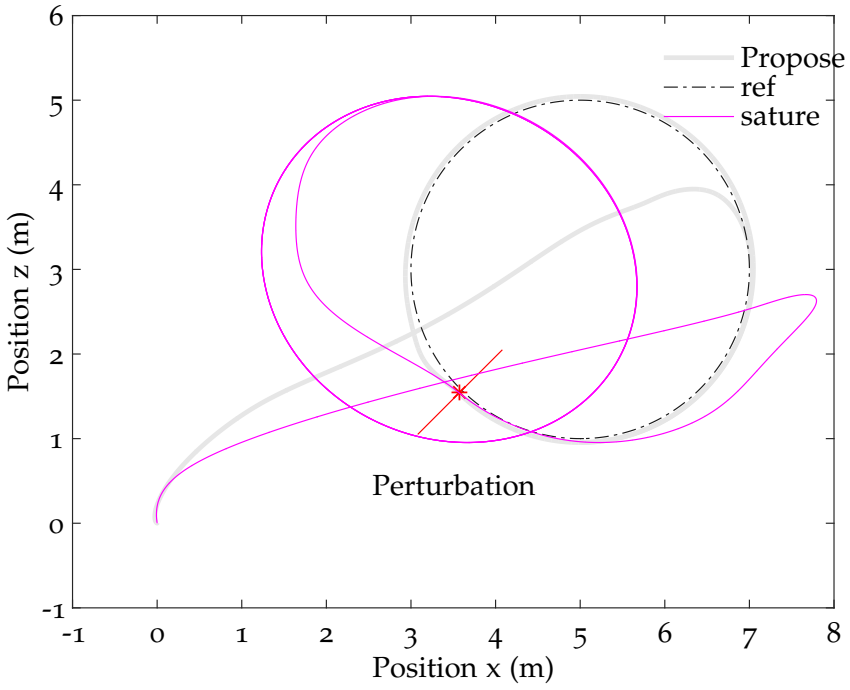


FIGURE 3.10 – *PVTOL, Simulation : Comparaison entre la commande proposée (Propose) avec la commande par saturation (sature) ; la commande proposée rejette bien la perturbation permanente tandis qu'elle ne peut pas être rejetée par la commande par saturation.*

2. A noter que cette commande n'est pas conçue pour rejeter une perturbation constante.

3.1.4 Temps de génération des tables

Cette commande nécessite le calcul d'une table de grande dimension à cause du maillage important choisi au départ pour l'espace conjoint état-entrée. Pour obtenir un maillage plus large, un deuxième essai a été effectué. Dans ce cas, le modèle du PVTOL est approximé par un système ultra-local (3.5) en supposant que θ est petit, $\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$ et $u_1 \approx 1$.

$$\begin{cases} \ddot{x} &= -\theta + F_1 \\ \ddot{z} &= u_1 + F_2 \\ \ddot{\theta} &= u_2 + F_3 \end{cases} \tag{3.5}$$

où F_1, F_2, F_3 sont des perturbations. Dans la méthode de commande sans modèle M. Fliess (2011), elles sont estimées en ligne.

De même façon, pour le système interne on prend :

$$\begin{cases} \ddot{x} &= -\theta + F_1 \\ \ddot{z} &= u_1 + F_2 \end{cases} \tag{3.6}$$

où u_1 et θ sont les entrées. Pour la génération des tables de prédiction, on néglige F_1, F_2, F_3. Le système devient alors un système linéaire et on peut utiliser la table des variables 3.1 avec un petit nombre de points. Ici, on prend trois points pour chaque variable comme la table 3.3.

Variable	Minimum	Maximum	Nombre de points
$\dot{x}$	$-4m.s^{-1}$	$4m.s^{-1}$	3
$\dot{z}$	$-4m.s^{-1}$	$4m.s^{-1}$	3
θ	$-\pi$	π	3
$\dot{\theta}$	$-6.0rad.s^{-1}$	$6.0rad.s^{-1}$	3
u_1	$0N$	$4N$	3
u_2	$-40Nm$	$40Nm$	3

TABLE 3.3 – *Paramètres des variables pour le système linéarisé*

Le temps pour génération des tables de comportements est inférieur de $5s$ tandis qu'il faut prendre au moins deux heures pour la génération des tables utilisant le modèle non linéaire dans la section précédente. On va vérifier que l'approximation de modèle va être compensée par le correcteur d'erreur de prédiction existant dans l'algorithme de commande (équation 1.8). Les deux essais précédents sont effectués à nouveau (figures 3.11, 3.12, 3.13, 3.14, 3.15, 3.16 et 3.17). La figure 3.12 présente le comportement du système sur l'axe z. Il existe un petit dépassement sur l'axe z pour la réponse au premier créneau. La commande arrive à rejeter la perturbation statique. De plus, le mécanisme d'adaptation globale arrive à corriger l'erreur de prédiction. Globalement on remarque

que l'angle est plus agité dans cette version de table qu'avec la précédente. Cela prouve qu'une bonne information de prédiction (corrigée par apprentissage si nécessaire) est plus efficace qu'un mécanisme de correction globale d'erreur de prédiction (équivalent à une fonction intégrale), même si cette dernière prouve son efficacité dans le cas du PVTOL.

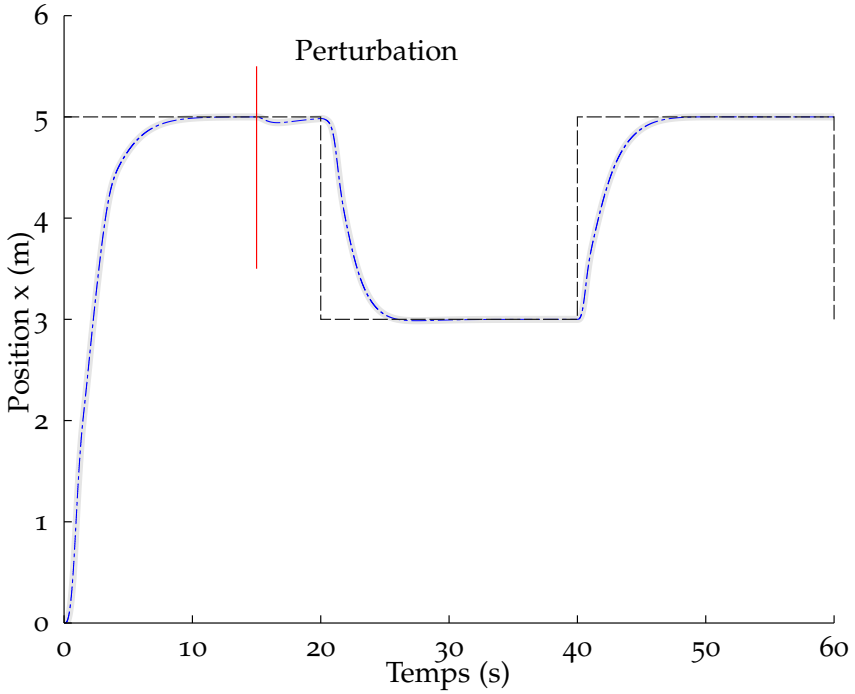


FIGURE 3.11 – *PVTOL, Simulation : Suivi d'un signal de référence carré, la position sur l'axe x. Référence interne en pointillé noir, position obtenue en simulation en gris continu*

3.2 Essais sur un X4 en mode PVTOL : le PVTOL-X4

On a mis en évidence, dans la section précédente, l'intérêt d'avoir un modèle tabulé de bonne qualité même si l'algorithme est capable de compenser suffisamment les erreurs pour assurer la stabilité. Dans ce qui suit, on va montrer que la table peut effectivement être améliorée en ligne par la mise en ?uvre de la partie apprentissage décrite au Chapitre 2. On montrera d'abord des simulations puis des essais réels.

3.2.1 Passage d'un quadricoptère à un PVTOL

Considérant un quadricoptère dans le plan XOZ comme sur la figure 3.18, on peut considérer cette partie du modèle du quadricoptère comme un PVTOL.

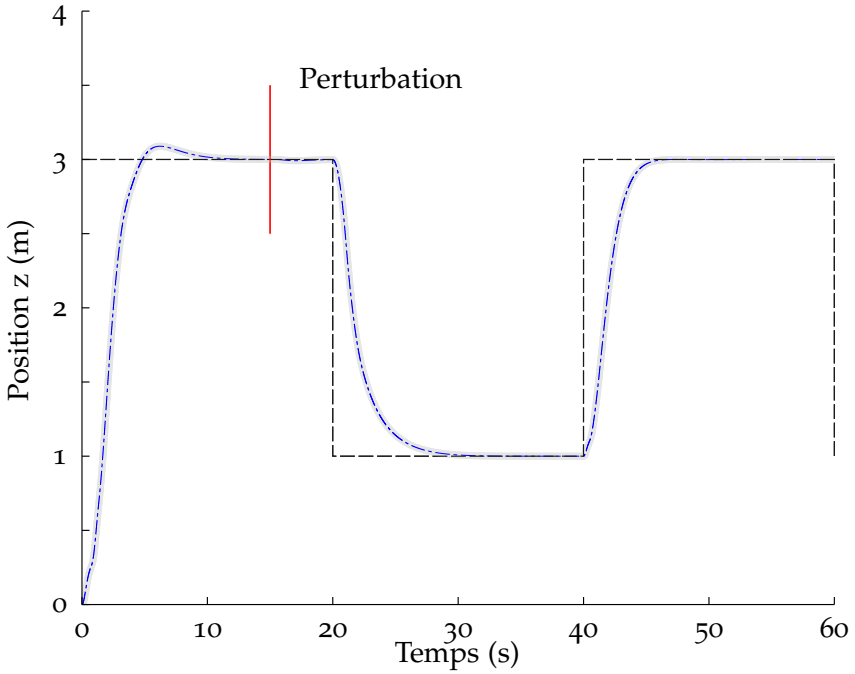


FIGURE 3.12 – *PVTOL, Simulation : Suivi d'un signal de référence carré, la position sur l'axe z. Référence interne en pointillé noir, position obtenue en simulation en gris continu.*

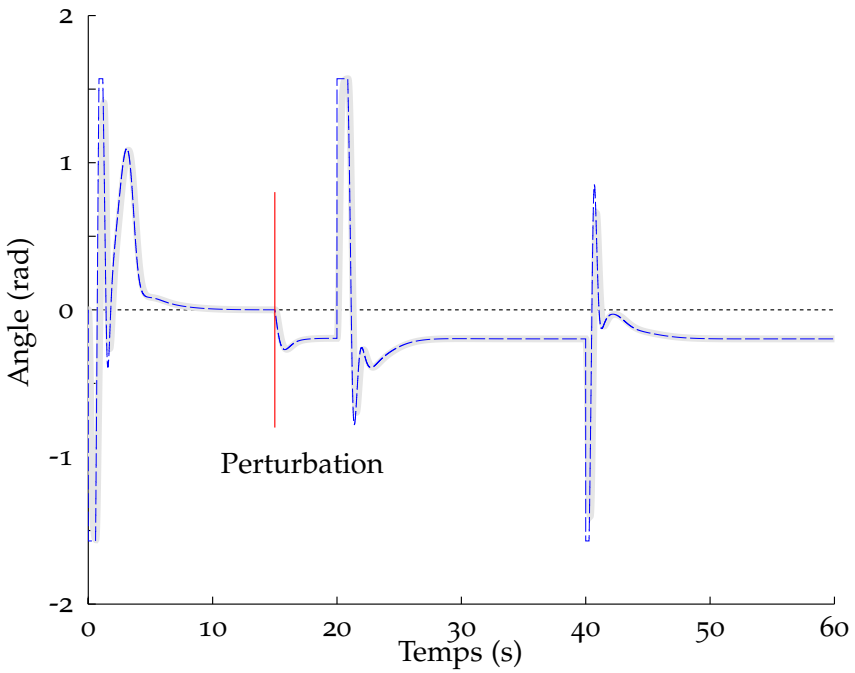


FIGURE 3.13 – *PVTOL, Simulation : L'angle θ et sa référence interne pour un signal de référence carré. Référence interne en pointillé noir, angle obtenu en simulation en gris continu.*

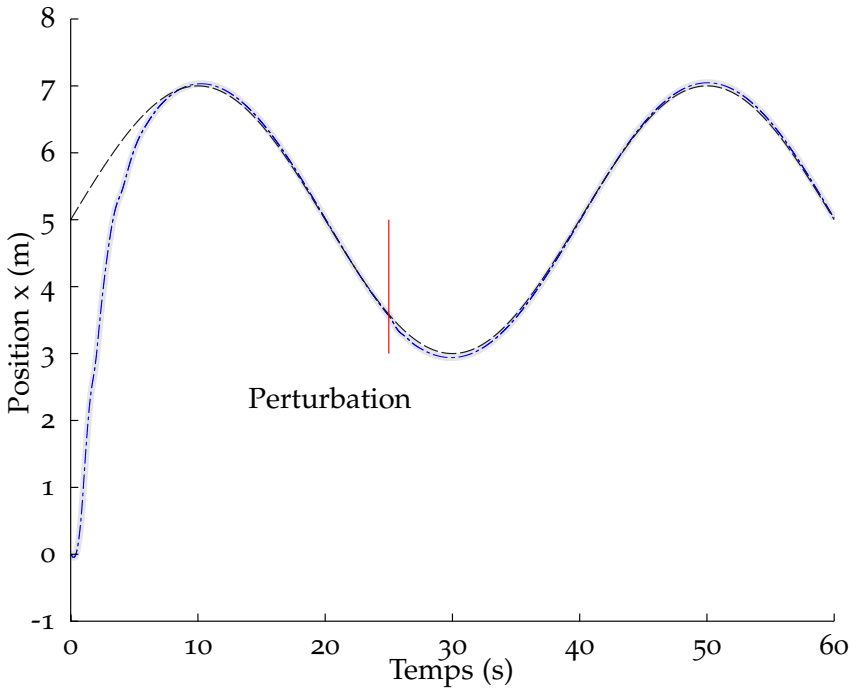


FIGURE 3.14 – *PVTOL, Simulation : Suivi d'un signal de référence sinusoïdale, la position sur l'axe x. Référence interne en pointillé noir, position obtenue en simulation gris continu.*

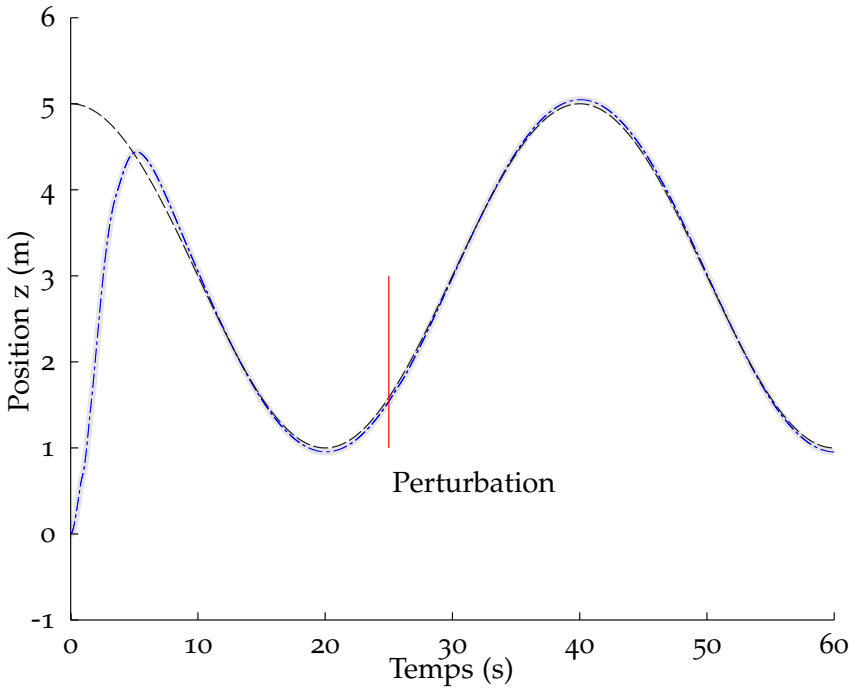


FIGURE 3.15 – *PVTOL, Simulation : Suivi d'un signal de référence sinusoïdale, la position sur l'axe z. Référence interne en pointillé noir, position obtenue en simulation en gris continu.*

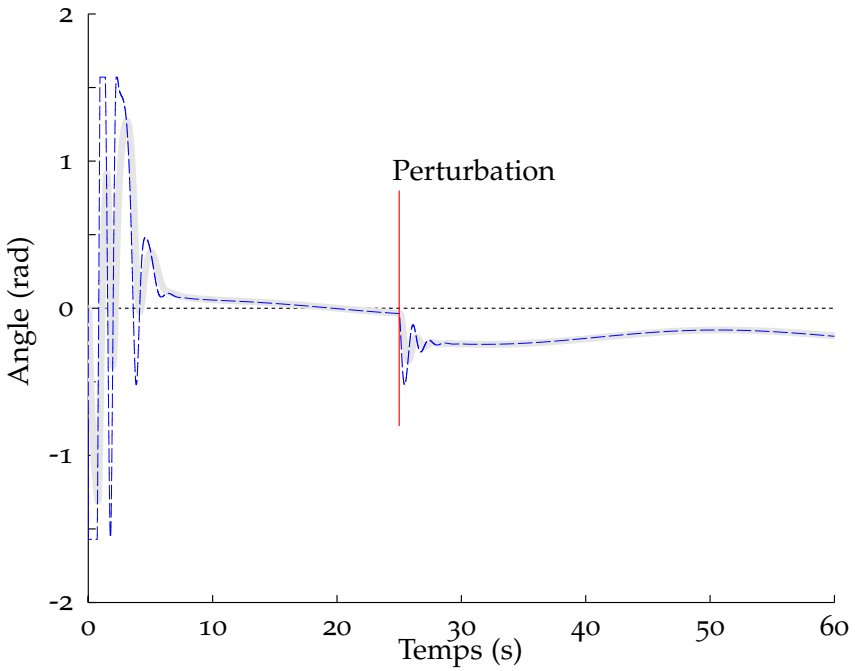


FIGURE 3.16 – *PVTOL, Simulation : L'angle θ et sa référence interne pour un signal de référence sinusoïdal. Référence interne en pointillé noir, angle obtenu en simulation en gris continu.*

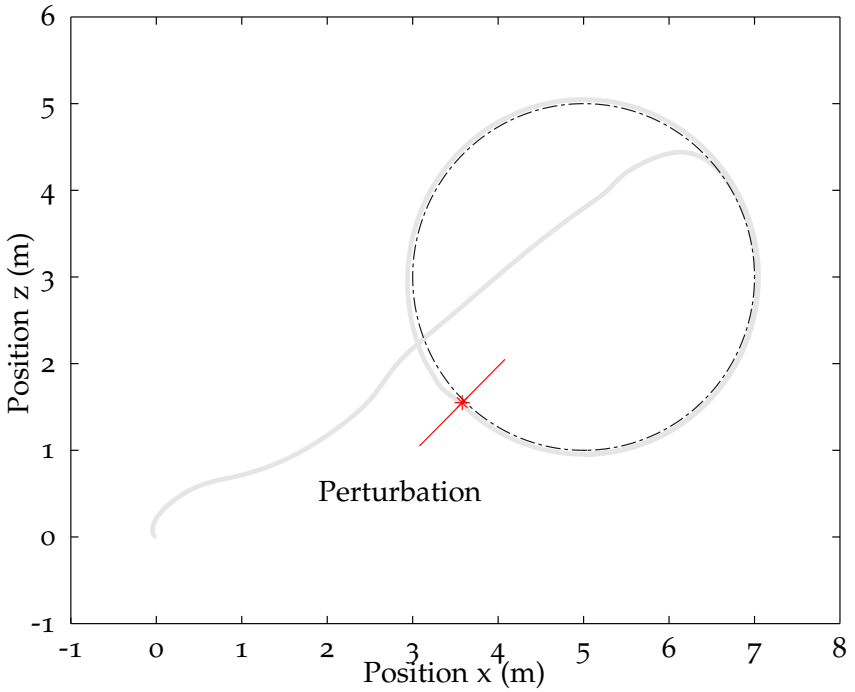


FIGURE 3.17 – *PVTOL, Simulation : Trajectoire de référence et résultat dans le plan Oxz*

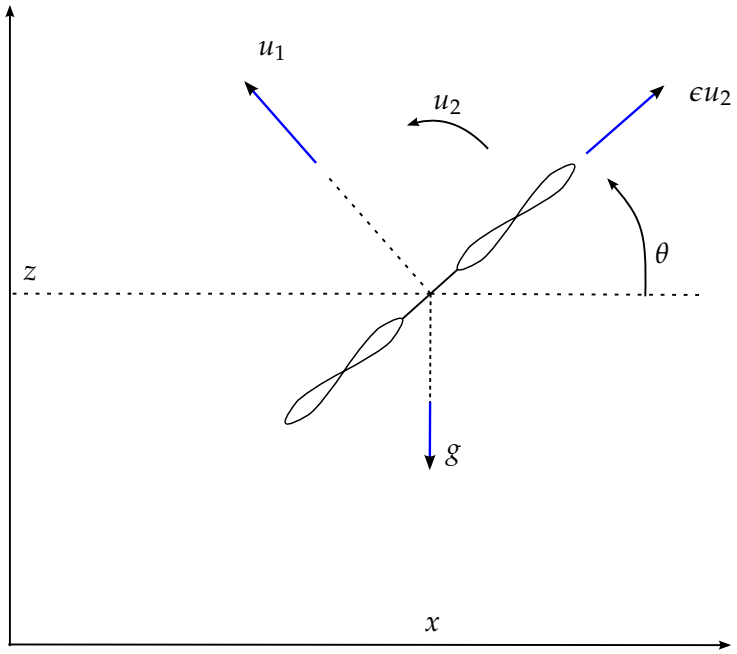


FIGURE 3.18 – *Le modèle du X4 dans le plan XOZ*

Les équations du système dans le plan XOZ peuvent s'écrire comme :

$$\begin{cases} m\ddot{x} &= -u_1 \sin\theta \\ m\ddot{z} &= u_1 \cos\theta - mg \\ I_z\ddot{\theta} &= u_2 \end{cases} \tag{3.7}$$

Dans ces équations, la force de poussée et le couple de roulis sont calculés par rapport aux vitesses des hélices comme suit :

$$\begin{cases} u_1 &= 2K(\omega_L^2 + \omega_R^2) \\ u_2 &= 2KL(\omega_L^2 - \omega_R^2) \end{cases} \tag{3.8}$$

où K est le coefficient de la poussée, L est la longueur des bras et ω_R, ω_L sont les vitesses angulaires des hélices à droite et à gauche respectivement. Supposons que ω_M est la vitesse maximale, on peut écrire u_1 et u_2 par :

$$\begin{cases} u_1 &= 2K\omega_M^2(\Omega_L^2 + \Omega_R^2) \\ u_2 &= 2KL\omega_M^2(\Omega_L^2 - \Omega_R^2) \end{cases} \tag{3.9}$$

Avec $\Omega_L = \dfrac{\omega_L}{\omega_M}$ et $\Omega_R = \dfrac{\omega_R}{\omega_M}$. En vol stationnaire la vitesse pour chaque hélice est égale à ω_0 telle que $mg = 4K\omega_0^2$. On obtient :

$$\begin{cases} u_1 &= \dfrac{2K\omega_M^2}{4K\omega_0^2} mg(\Omega_L^2 + \Omega_R^2) \\ u_2 &= \dfrac{2KL\omega_M^2}{4K\omega_0^2} mg(\Omega_L^2 - \Omega_R^2) \end{cases} \tag{3.10}$$

Les deux entrées ci-dessus deviennent :

$$\begin{cases} u_1 &= \dfrac{mg}{\Omega_0^2}\dfrac{1}{2}(\Omega_L^2+\Omega_R^2) \\ u_2 &= \dfrac{mgL}{\Omega_0^2}\dfrac{1}{2}(\Omega_L^2-\Omega_R^2) \end{cases} \tag{3.11}$$

Les équations du mouvement peuvent s'écrire comme pour le PVTOL avec :

$$\begin{cases} u_1^* &= \dfrac{u_1}{mg} \\ u_2^* &= \dfrac{u_2}{I_z} \\ x^* &= x/g \\ z^* &= z/g \end{cases} \tag{3.12}$$

Alors, la vitesse pour chaque hélice est calculée par :

$$\begin{cases} \Omega_L &= \Omega_0\sqrt{u_1^*+\dfrac{I_z}{mgL}u_2^*} \\ \Omega_R &= \Omega_0\sqrt{u_1^*-\dfrac{I_z}{mgL}u_2^*} \end{cases} \tag{3.13}$$

On introduit deux variables comme ceci :

$$\begin{cases} u_{1scale} &= \Omega_0^2 \\ u_{2scale} &= \Omega_0^2\dfrac{I_z}{mgL} \end{cases} \tag{3.14}$$

que l'on peut calculer aussi comme :

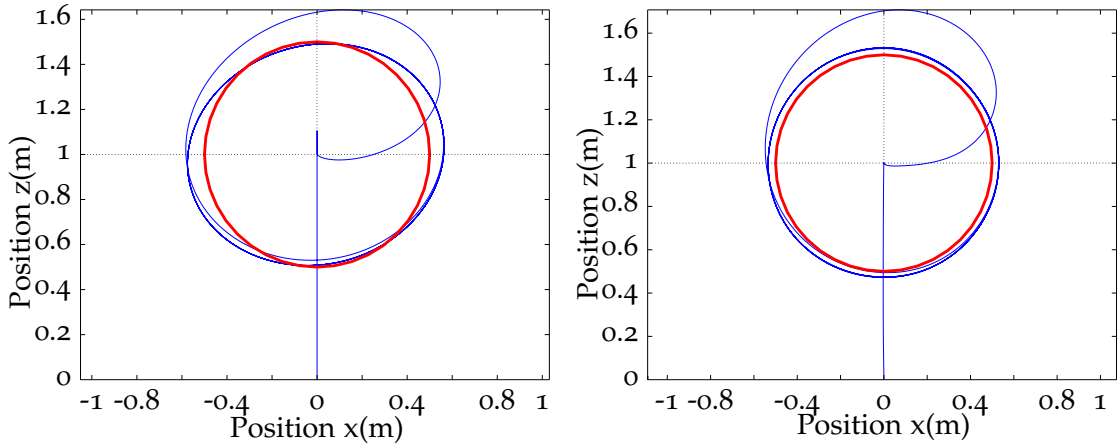
$$\begin{cases} u_{1scale} &= \dfrac{mg}{4K\omega_M^2} \\ u_{2scale} &= \dfrac{I_z}{4KL\omega_M^2} \end{cases} \tag{3.15}$$

Ce qui donne :

$$\begin{cases} \Omega_L &= \sqrt{u_1^* u_{1scale}+u_2^* u_{2scale}} \\ \Omega_R &= \sqrt{u_1^* u_{1scale}-u_2^* u_{2scale}} \end{cases} \tag{3.16}$$

3.2.2 Apprentissage en simulation sur le PVTOL-X4

Dans cette simulation, l'algorithme d'apprentissage, vu au Chapitre 2, est intégré à la commande. La table de prédiction est initialisée sur la base du modèle linéaire (3.5). La masse du PVTOL prend les valeurs $m = 0.496\ kg$ pour la table et $m = 0.65\ kg$ pour le système réel simulé. Le coefficient de poussée $\Delta F = 0.2F$ par changement du coefficient de la poussée $K_n = 1.2K$, donc $\Delta F = K_n\omega^2 - K\omega^2 = 0.2F$. La figure 3.19 montre l'effet de l'apprentissage. Un cercle est la trajectoire à suivre. Avant l'apprentissage, la commande ne peut pas suivre la trajectoire demandée avec deux dynamiques identiques sur deux axes. Il y a un dépassement sur l'axe z. Après l'apprentissage, le comportement du



(a) *Suivi d'un cercle sans apprentissage* (b) *Suivi d'un cercle après l'apprentissage*

FIGURE 3.19 – *Simulation : Effet d'apprentissage sur PVTOL-X4*

système s'est amélioré. La forme de la trajectoire est circulaire et le dépassement sur l'altitude a disparu. La différence entre la trajectoire demandée et la trajectoire obtenue est le résultat de l'existence de la référence interne qui prend la forme d'un filtre du second ordre. Sur chaque axe, la trajectoire est une sinusoïde qui est modifiée par le filtre.

3.2.3 Essai réel sur le PVTOL-X4

Afin d'effectuer des tests réels, on va reproduire l'essai de la simulation. Un simulateur 3D [3] permet de valider le code développé pour les drones avant de passer sur les machines réelles. La table reprend les paramètres retenus pour réaliser les essais précédents en simulation. Tout d'abord, pour commencer les essais, le drone décolle. On lui applique ensuite la trajectoire circulaire de rayon $r = 0.5\ m$ par suivi de deux consignes sinusoïdales avec $\omega = 0.8\ rad.s^{-1}$. Après plusieurs tours, il s'arrête. Le Parrot-AR2 [4] est utilisé pour valider la commande. Les essais sont faits dans la salle volière du laboratoire. Le système de localisation est basé sur un système Optitrack [5]. La figure 3.20 présente la trajectoire réelle avant et après l'apprentissage. L'effet de l'apprentissage se retrouve sur les essais réels. Les essais sont répétés afin de compléter l'apprentissage sur une plus longue période. La figure 3.21 présente la trajectoire réelle pour un autre essai. Les comportements sur deux axes sont identiques. Les figures 3.22 et 3.23 montrent que la commande a parfaitement stabilisé la trajectoire. Dans les figures 3.24 et 3.25, les vitesses sur deux axes x et z sont bien suivies. La figure 3.26 montre la force de poussée nominale appliquée au système. La force de poussée est modifiée de manière synchrone

3. Framework open source développé au laboratoire Heudiasyc, http ://www.hds.utc.fr/heudiasyc/production/logiciels-970
4. http ://www.parrot.com/fr/produits/ardrone-2/
5. http ://optitrack.com/

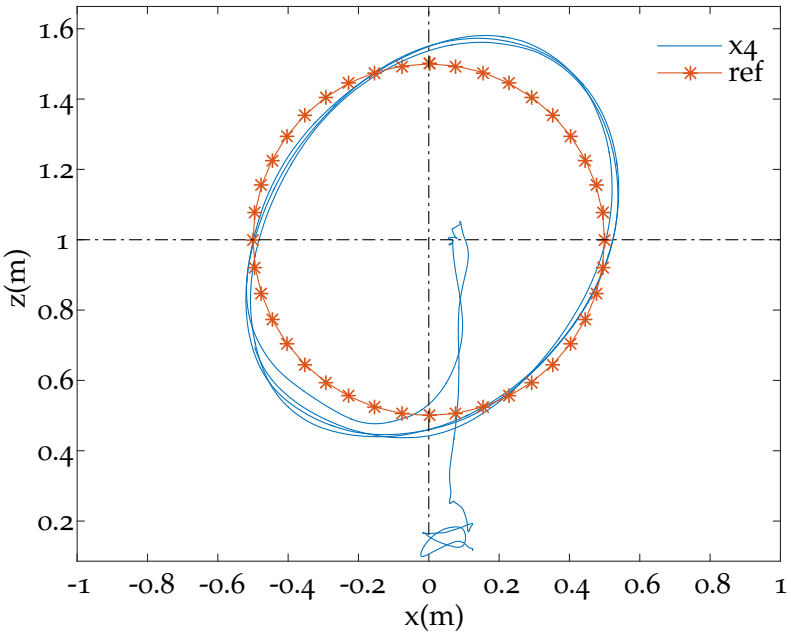
avec l'orientation. La figure 3.27 donne les angles de tangage-roulis-lacet que l'on peut mesurer avec la centrale inertielle du drone.

3.3 Commande d'attitude pour le quadricoptère

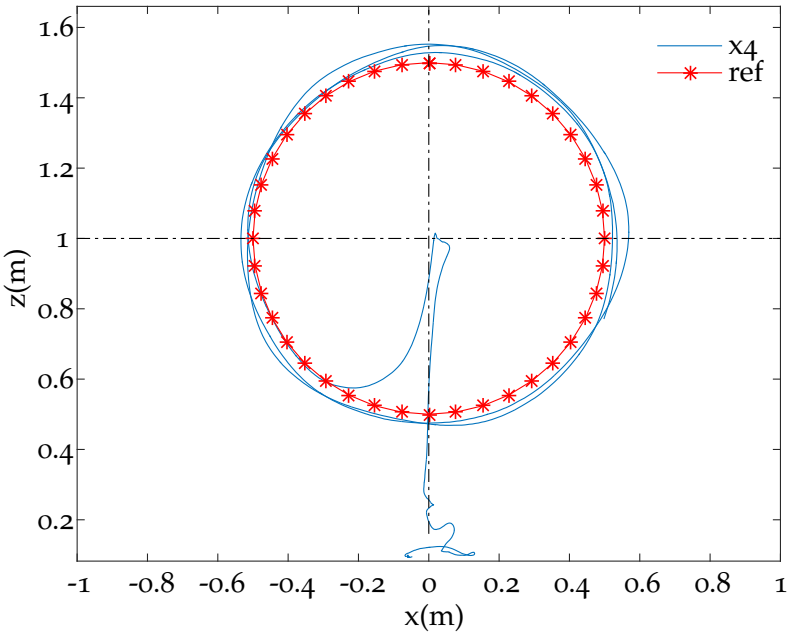
Le VTOL plan est un exemple prisé en automatique car il présente des caractéristiques utiles pour montrer l'efficacité d'une technique de commande. Pour autant, on ne pilote généralement pas un aéronef de type hélicoptère en se basant sur ce type de modèle. Il est plus logique de faire une boucle interne pour la stabilisation de l'attitude puis d'utiliser cet actionneur intermédiaire pour commander les translations. C'est cette structure qui va être étudiée dans ce qui suit.

La figure 3.28 présente un quadricoptère dans l'espace 3D. C'est un engin volant à quatre hélices. Chaque hélice i tourne à la vitesse ω_i et génère une force de poussée F_i et un couple de trainée. Le quadricoptère est donc un système sous-actionné avec quatre entrées et six degrés de liberté. La commande pour le quadricoptère peut se diviser en deux étapes : la commande de la translation et la commande d'attitude. De par la structure de système sous-actionné, la commande de la translation va générer une attitude désirée qui sera la trajectoire pour la commande d'attitude. Dans ce domaine beaucoup de techniques de commande ont été employée : la commande classique PID, la commande adaptative, la commande prédictive, la commande basée sur la fonction de Lyapunov, etc. Par rapport au défi de contrôler le quadricoptère en manœuvre agressive, on peut citer ici les travaux suivants dans la littérature. Une boucle ouverte optimisée comme solution d'un problème de minimisation, a été proposé dans le travail de Gillula et al. (2010). La trajectoire pour faire un looping est apprise par un algorithme d'apprentissage dans Lupashin et al. (2010). Pour la commande d'attitude seul, des méthodes de commande sont proposées dans l'état de l'art de Chaturvedi et al. (2011). Le temps minimal est considéré dans le travail de Mellinger et Kumar (2011). Basée sur la structure du quadricoptère qui peut voler sur le dos, dans le travail de Mark et Jonathan (2012), la trajectoire a été générée afin d'appliquer une commande par retour d'état basée sur une fonction de Lyapunov. La commande complète d'attitude en utilisant les quaternions a été proposée dans Fresk et Nikolakopoulos (2013). La commande basée sur les groupes de Lie, proposée par Goodarzi et al. (2013), permet de faire un looping sur le quadricoptère. Pour faire une manœuvre agressive Tomix et al. (2014) ont proposé un algorithme d'apprentissage, Spedicato et al. (2016) ont introduit une commande de type LQR .

Dans le contexte de ce travail, on va appliquer la commande proposée dans sa forme d'état sur le quadricoptère. Tout d'abord, il faudra définir les variables de la table de prédiction. Dans ce cadre, le choix du modèle du système est très important. La modélisation qui permet de discrétiser le modèle dynamique sera introduite. Ensuite, le schéma de la commande sera proposé. Finalement, les résultats et les commentaires termineront cette section.



(a) *Suivi d'un cercle sans apprentissage*



(b) *Suivi d'un cercle après l'apprentissage*

FIGURE 3.20 – *Essai réel : Effet d'apprentissage sur PVTOL-X4*

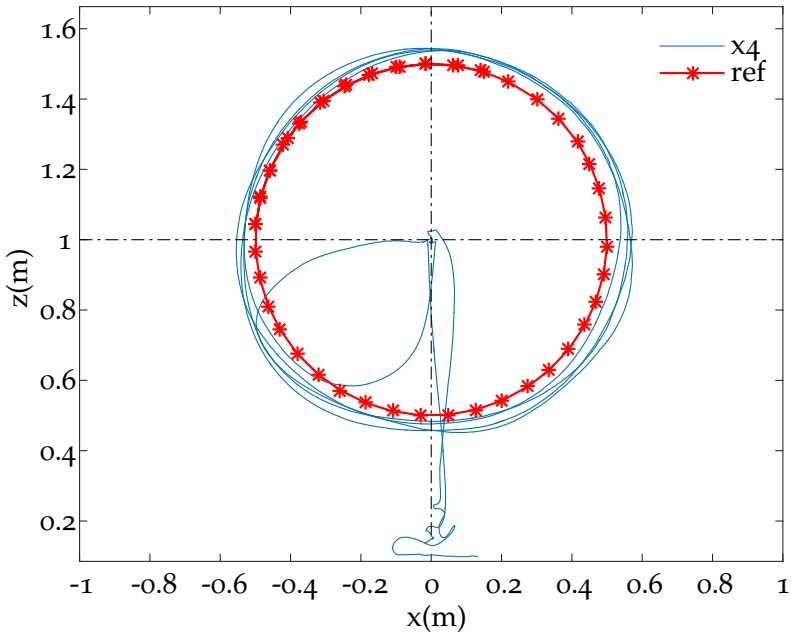


FIGURE 3.21 – *Essai réel : Suivi d'un cercle*

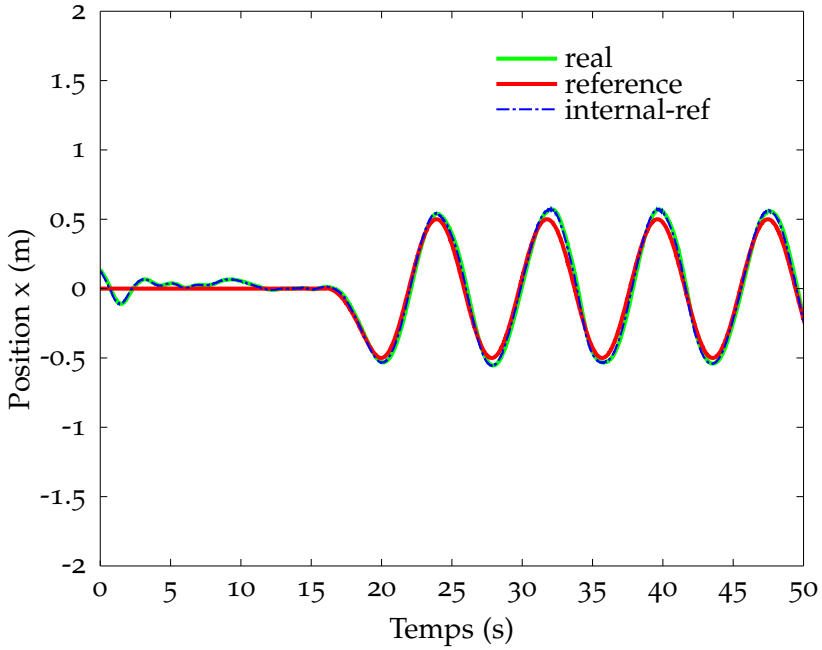


FIGURE 3.22 – *Essai réel : Suivi d'un cercle - position sur x*

3.3.1 Modélisation

On considère le quadricoptère montré sur la figure 3.28. Pour s'adapter aux définitions des référentiels capteur et simulation, le repère inertiel est choisi avec Oz vers le haut. On fait de même pour le repère fixé au système $Gx_by_bz_b$. G est le centre de gravité

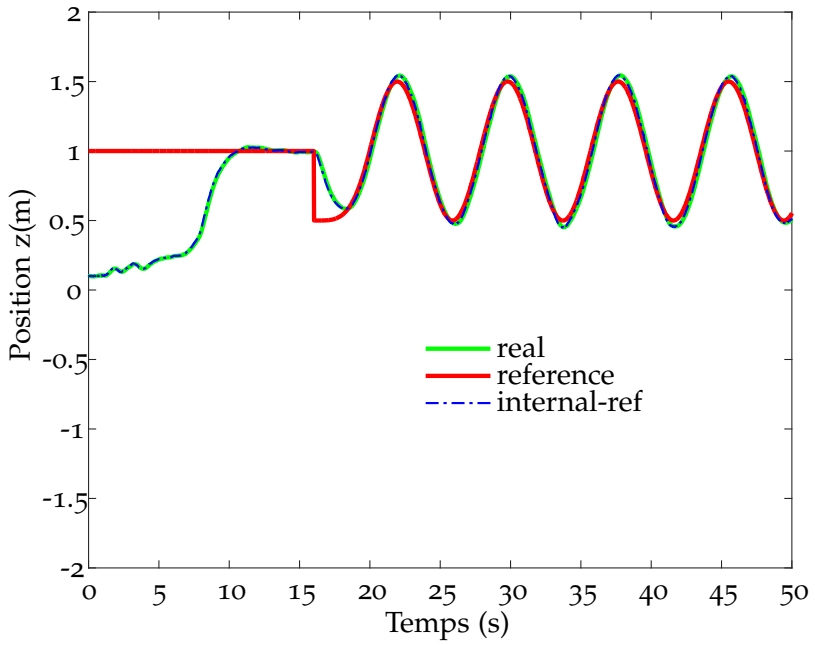


FIGURE 3.23 – *Essai réel : Suivi d'un cercle - position sur z*

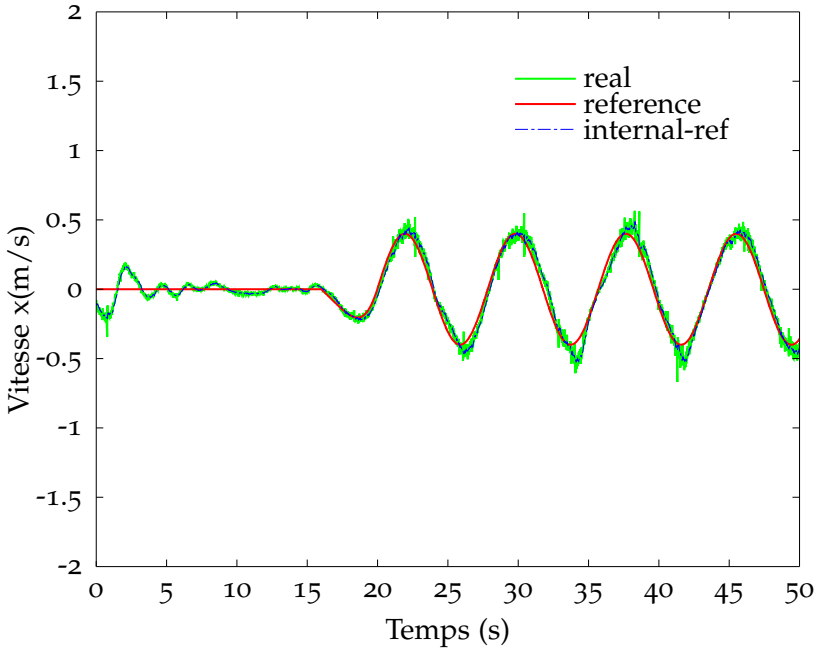


FIGURE 3.24 – *Essai réel : Suivi d'un cercle - vitesse sur x*

du système. On néglige également l'effet du sol, le vent et l'effet dit de flapping des hélices. On néglige les forces de trainée de translation générées par les hélices.

La force de poussée et le couple de trainée pour chaque hélice sont calculés approximativement par :

$$F_i = k_t \omega_i^2, \ \Gamma_i = k_c \omega_i^2$$

Les coefficients aérodynamiques k_t et k_c sont déterminés par des techniques d'identifi-

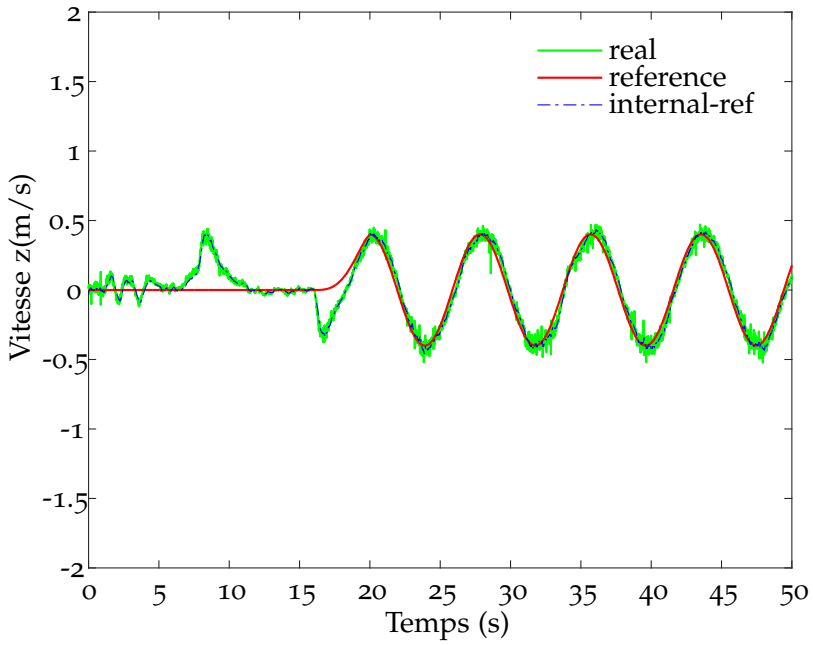


FIGURE 3.25 – *Essai réel : Suivi d'un cercle - vitesse sur z*

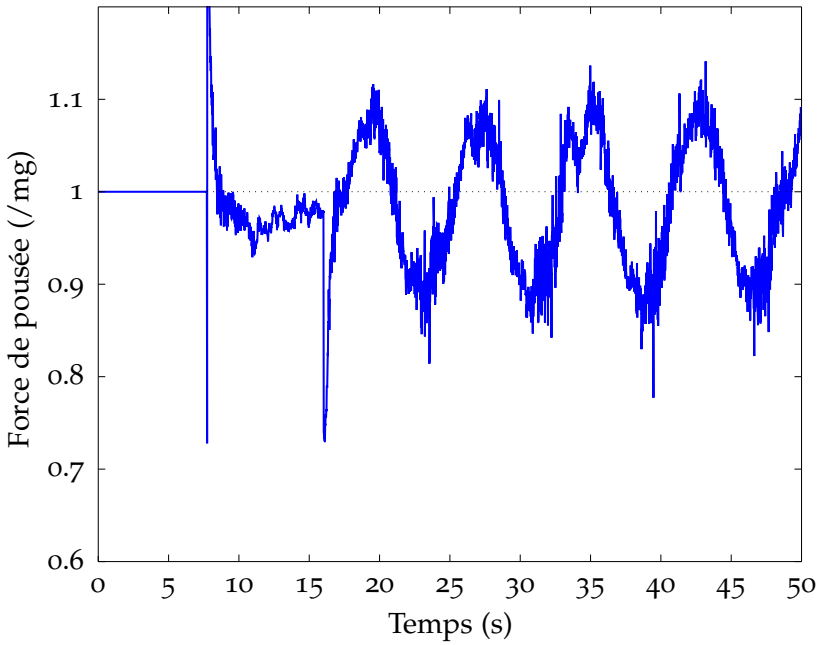


FIGURE 3.26 – *Essai réel : Suivi d'un cercle - Force de poussée*

cation. Donc la force de poussée totale f et les couples $\Gamma = [\Gamma_x, \Gamma_y, \Gamma_z]^T$ appliqués au quadricoptère sont définis par :

$$\begin{bmatrix} f \\ \Gamma_x \\ \Gamma_y \\ \Gamma_z \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ k_t L & k_t L & -k_t L & -k_t L \\ -k_t L & k_t L & k_t L & -k_t L \\ -k_c & k_c & -k_c & k_c \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.17}$$

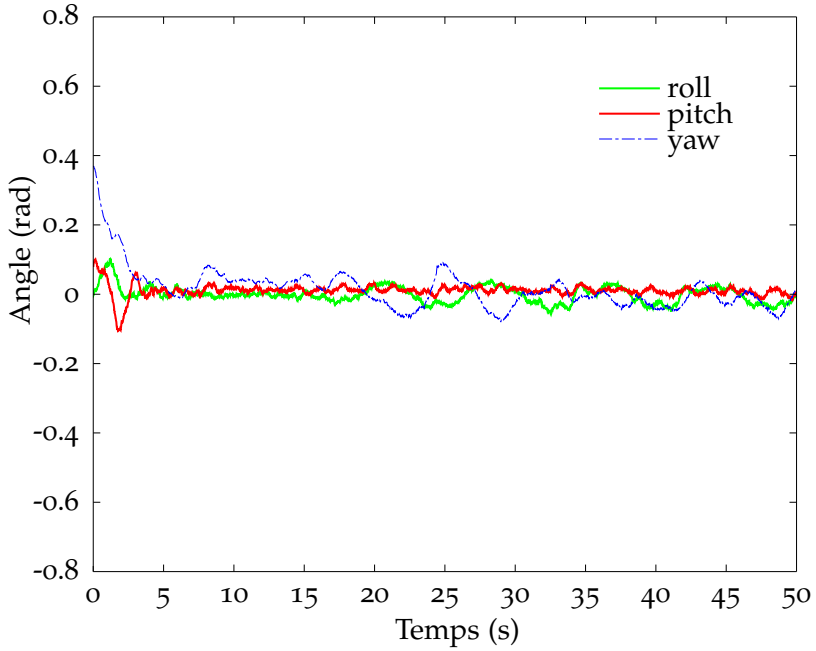


FIGURE 3.27 – *Essai réel : Suivi d'un cercle - RPY angles*

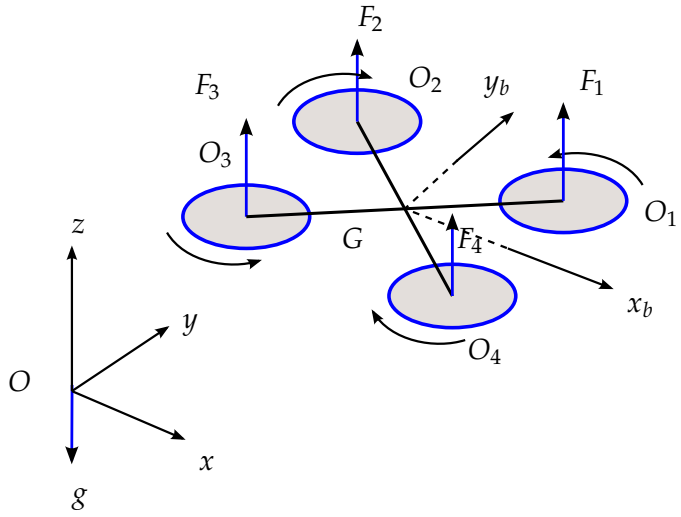


FIGURE 3.28 – *Quadricoptère complet*

L est la distance entre la position de l'hélice et Ox_b, $\omega_i, i = \overline{1,4}$ sont les vitesses angulaires des hélices. En appliquant les équations de Newton-Euler, on obtient les équations de mouvement suivantes :

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= f\mathbf{R}\mathbf{e}_3 - mg\mathbf{e}_3 \\ \dot{\mathbf{R}} &= \mathbf{R}[\Omega]^\times \\ \mathbf{J}\dot{\Omega} &= (\mathbf{J}\Omega) \times \Omega + \Gamma \end{cases} \tag{3.18}$$

avec $\mathbf{x}$ la position, $\mathbf{v}$ la vitesse de translation et $\mathbf{R}$ la matrice de rotation représentant l'attitude du système. f est la force de poussée totale et Γ les couples générés par les

forces de poussée de chaque hélice. $\mathbf{J}$ est la matrice d'inertie exprimée dans le repère $Gx_by_bz_b$ fixé au centre de gravité. Ω est la vitesse angulaire exprimée dans ce repère. $[\mathbf{x}]^\times$ est la matrice antisymétrique associée à un vecteur de dimension trois $\mathbf{x} = [x_1, x_2, x_3]^T$. Cette matrice représente le produit vectoriel du vecteur $\mathbf{x}$ et d'un vecteur arbitraire $\mathbf{x} \times \mathbf{y} = [\mathbf{x}]^\times \mathbf{y}$.

$$[\mathbf{x}]^\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

En utilisant le quaternion pour l'attitude, les équations dynamiques sont écrites sous la forme :

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= f\mathbf{R}(q)\mathbf{e}_3 - mg\mathbf{e}_3 \\ \dot{q} &= \frac{1}{2}\Xi(q)\Omega \\ \mathbf{J}\dot{\Omega} &= (\mathbf{J}\Omega) \times \Omega + \Gamma \end{cases} \tag{3.19}$$

avec

$$\Xi(q) = \begin{bmatrix} -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}^T$$

Le système étant sous-actionné, la commande va être souvent divisée en deux sous-systèmes : un pour la commande en translation et un autre pour la commande en attitude. On s'intéresse maintenant au problème de la commande d'attitude en combinaison avec la commande d'altitude. Par rapport à la commande proposée, basée sur le modèle tabulé, il est difficile d'utiliser directement le quaternion q ou la matrice de rotation $\mathbf{R}$ dans la table de prédiction parce qu'ils contiennent des contraintes entre les variables. Pour les trois angles d'Euler on rencontre le problème de singularité. Pour éviter ces problèmes, l'idée développée ici est d'utiliser des variables représentant l'erreur d'orientation entre deux pas de temps. Le modèle qui nous intéresse est :

$$\begin{cases} \dot{v_z} &= f\mathbf{R}(9) - mg \\ \dot{\mathbf{R}} &= \mathbf{R}[\Omega]^\times \\ \mathbf{J}\dot{\Omega} &= (\mathbf{J}\Omega) \times \Omega + \Gamma \end{cases} \tag{3.20}$$

où $\mathbf{R}(9)$ est le neuvième élément de la matrice $\mathbf{R}$. En temps discret, avec un temps d'échantillonnage de Ts, on utilise la formule de Rodrigue :

$$\mathbf{R}_{k+1} = \mathbf{R}_k \exp(T_s \Omega_k) \tag{3.21}$$

On définit l'erreur de rotation sous la forme :

$$\Delta\mathbf{R}_{k+1} = \frac{1}{2}\left(\mathbf{R}_{k+1}^T\mathbf{R}_k - \mathbf{R}_k^T\mathbf{R}_{k+1}\right) \tag{3.22}$$

Cette matrice est une matrice antisymétrique. Donc, on peut représenter $\Delta\mathbf{R}_{k+1}$ par un vecteur de trois composantes indépendantes $\mathbf{e}_R^{k+1}$ telles que $[\mathbf{e}_R^{k+1}]^\times = \Delta\mathbf{R}_{k+1}$. De cette manière on peut représenter (3.20) en temps discret sous la forme :

$$[v_z^{k+1}, \mathbf{e}_R^{k+1}, \Omega_{k+1}]^T \quad = \quad F(\Gamma_k, v_z^k, \mathbf{e}_R^k, \Omega_k, \mathbf{R}_k(9)) \tag{3.23}$$

Cette équation représente la table de prédiction utilisée pour la commande. En considérant la vitesse de rotation Ω_k constante au pas k, on peut calculer $\Delta\mathbf{R}_{k+1}$ à l'aide de la formule de Rodrigue :

$$\Delta\mathbf{R}_{k+1} \quad \approx \quad -T_s[\Omega_k]^\times \tag{3.24}$$

Cela donne

$$\mathbf{e}_R^{k+1} \approx -T_s\Omega_k$$

3.3.2 Modèle complet

Pour obtenir un modèle de simulation aussi complet et réaliste que possible, on a construit une réalisation multi-physique du quadricoptère sous MapleSim. On l'a ensuite exporté sous forme d'une S-Function dans Simulink. Les paramètres de ce modèle sont donnés dans la table 3.4. Ils correspondent aux caractéristiques d'un AR drone 2 de Parrot. Le temps d'échantillonnage utilisé pour le modèle discret est $Ts = 0.01\ s$.

			Paramètre
m	0.506	kg	Masse totale du modèle
I_x	2.38×10^{-3}	$kg.m^2$	Moment d'inertie sur l'axe x
I_y	3.85×10^{-3}	$kg.m^2$	Moment d'inertie sur l'axe y
I_z	5.9×10^{-3}	$kg.m^2$	Moment d'inertie sur l'axe z
L	0.15	m	Longueur des bras
k_t	2.3×10^{-5}		Coefficient aérodynamique de la poussée
k_c	2×10^{-6}		Coefficient aérodynamique du couple de trainée
Ω_{max}	300	$rad.s^{-1}$	Vitesse angulaire maximale des hélices

TABLE 3.4 – *Paramètres du quadricoptère*

3.3.3 Résultat en simulation

Dans les essais suivants, la table de prédiction est construite en utilisant le modèle d'erreur de rotation. La commande est appliquée sur le quadricoptère exporté par MapleSim. Dans la première simulation, on souhaite suivre une trajectoire q_d avec les trois angles de tangage-roulis-lacet comme sur la figure 3.29. Ce sont les signaux sinusoïdaux $\theta_d = 0.5\sin(0.5(t-i))$ avec $i = 0$ pour le tangage, $i = 1$ pour le roulis et $i = 2$ pour le lacet. Cette figure montre que l'attitude est bien suivi. La figure 3.30 montre le

vecteur O_bz pendant le suivi de la trajectoire. D'un point de vue pratique, la référence est construite en utilisant des angles d'Euler puis convertie en matrice de rotation ou quaternions selon les besoins.

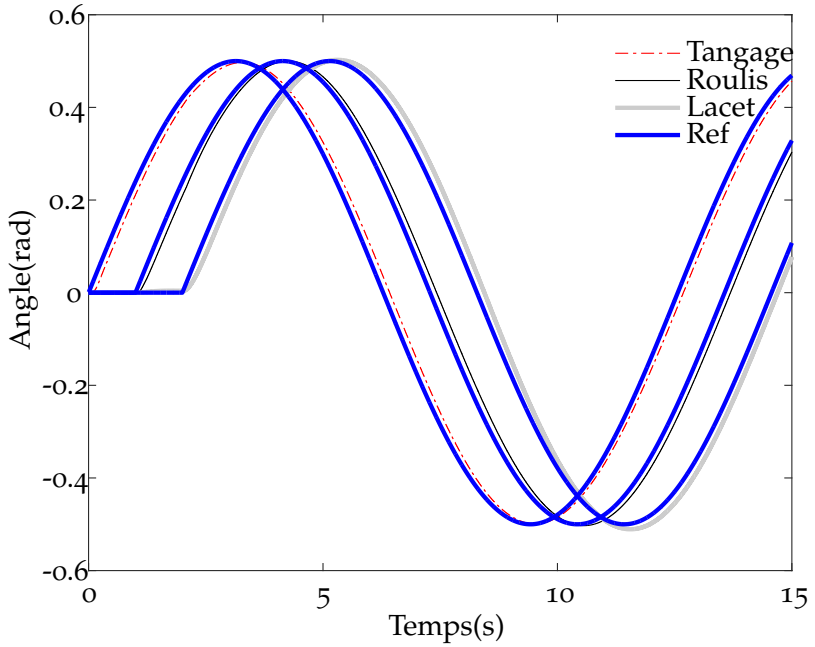


FIGURE 3.29 – *Quadricoptère, Simulation : Suivi d'une trajectoire d'attitude*

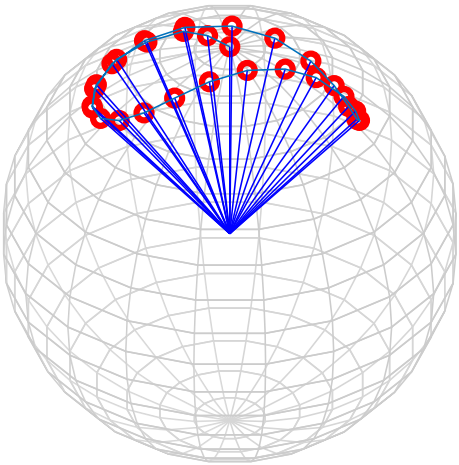


FIGURE 3.30 – *Quadricoptère, Simulation : Suivi d'une trajectoire d'attitude*

Dans la deuxième simulation, on a fait un looping sur l'angle de tangage, les deux autres angles étant stabilisés autour de zéro. La figure 3.32 montre que l'angle de tangage a bien suivi sa référence, avec un délai dû à l'erreur de poursuite. La figure 3.31 montre également que le vecteur O_bz a tourné un tour autour de l'axe x.

Pour stabiliser le quadricoptère à une hauteur z_d, la vitesse de référence sur l'axe z est générée par une commande classique $v_{zd} = K_p * (z_d - z)$. Dans cette simulation

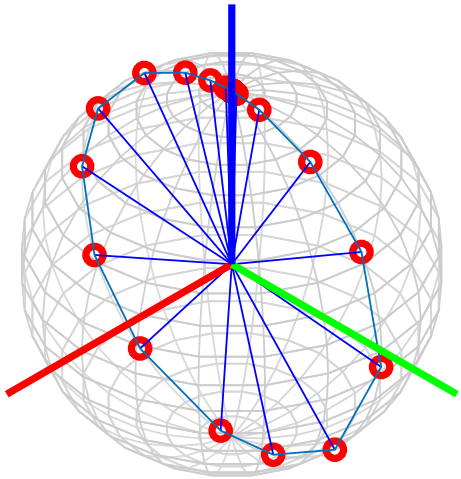


FIGURE 3.31 – *Quadricoptère, Simulation : Faire un looping sur l'axe x.*

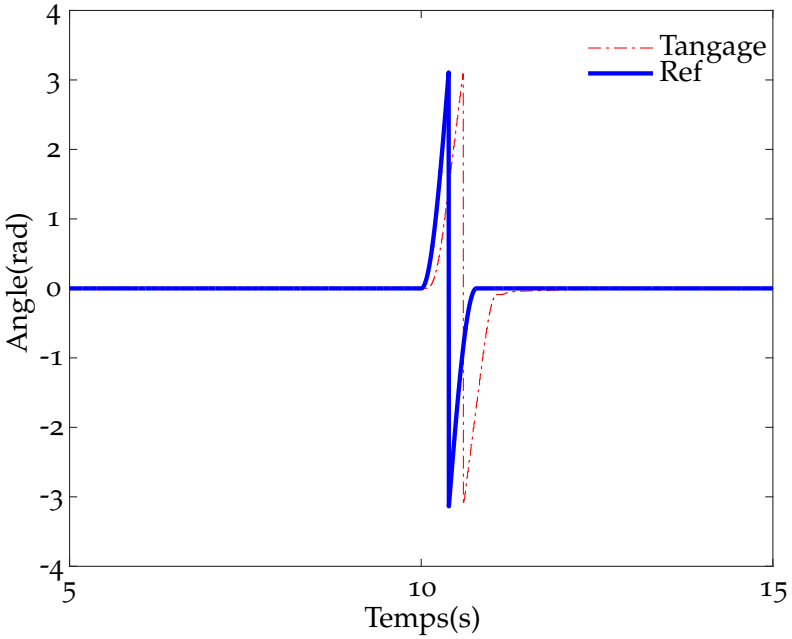


FIGURE 3.32 – *Quadricoptère, Simulation : Angle de tangage et sa référence pour un looping simple*

$z_d = 2.5m$ et K_p choisi est $K_p = 0.4.s^{-1}$. On trouve sur la figure figure 3.33, la position sur l'axe z, la vitesse et sa référence générée par la régulation de la position. Pendant le looping, le système descend. Pour la mise en pratique, il faudra donc augmenter la hauteur du système pour éviter la collision avec le sol.

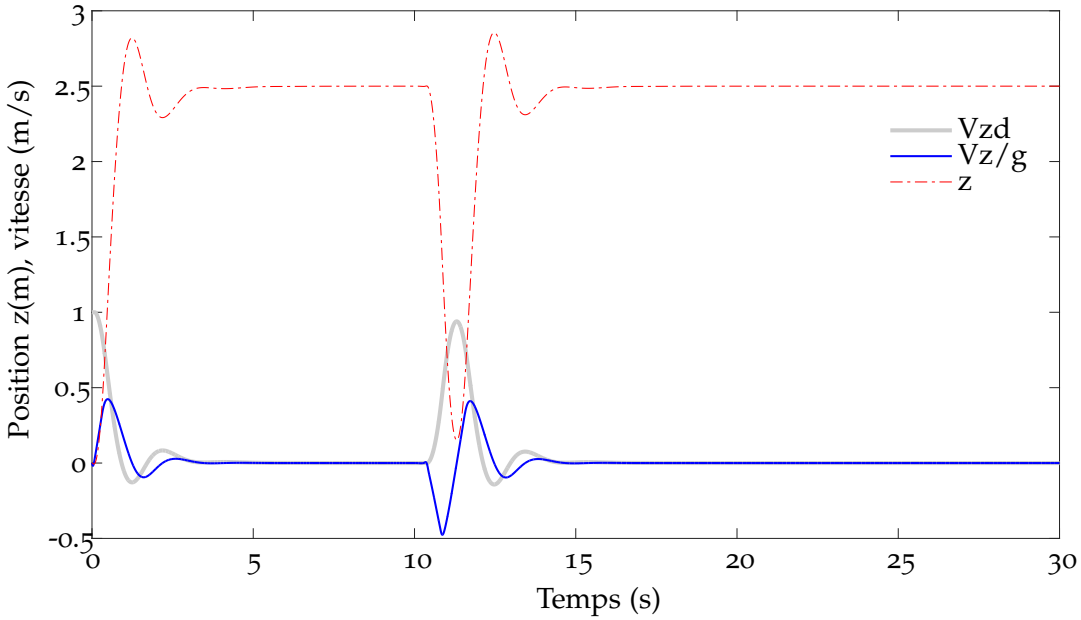


FIGURE 3.33 – *Quadricoptère, Simulation : La position, la vitesse et sa référence sur l'axe z. La position de référence en z est à la hauteur de* 2.5*m.*

Conclusion du chapitre

La commande proposée a été validée sur un système réel rapide, le PVTOL. Les tests sur le modèle simplifié du PVTOL montrent l'efficacité de la commande en terme de stabilité et de robustesse pour un système sous-actionné. On a aussi constaté que la compensation globale d'erreur de prédiction est fonctionnelle sur un modèle de mauvaise qualité mais qu'une bonne table de prédiction est plus efficace. On a alors appliqué l'algorithme d'apprentissage et montré sa capacité d'amélioration de la réponse. Le modèle linéarisé du système peut être utilisé pour générer la table de prédiction en vue d'accélérer le temps de préparation des données. En utilisant la table de prédiction générée par le modèle linéarisé avec l'apprentissage, on obtient une table plus précise qu'espéré. Pour la commande d'attitude du véhicule aérien, le modèle de prédiction est un modèle hybride avec l'état et la sortie. Il a été approximé grâce à une analyse mathématique. Il faudra ajouter un observateur pour reconstruire l'entrée de la table de prédiction et l'algorithme d'apprentissage pour modifier la table de prédiction en ligne. Bien que l'on ait généré seulement la vitesse de référence sur l'axe z pour stabiliser la hauteur, les tâches plus complexes peuvent être faites en ajoutant les régulations de la position sur les autres axes x et y. Cette commande doit encore être validée par des essais réels.

Chapitre 4

Application au véhicule intelligent

Dans ce chapitre, nous allons stabiliser un véhicule autour d'une trajectoire. Nous présenterons le modèle bicyclette qui est souvent utilisé pour l'étude de la dynamique d'un véhicule. Ce modèle sera utilisé pour construire la table de prédiction pour la commande. D'abord des rappels sur la dynamique des pneumatiques seront proposés afin de les incorporer dans le modèle. Ensuite, nous construirons le modèle bicyclette sous la forme d'équations différentielles permettant de générer des données. Finalement, les résultats de tests en simulation et sur un véhicule réel seront présentés.

Sommaire

4.1 Commande d'un véhicule autonome

Le modèle bicyclette représenté sur la figure 4.4 est souvent utilisé pour la prise en compte de la dynamique de véhicule dans un système de commande. C'est un système sous-actionné qui présente des caractéristiques dynamiques suffisantes pour le problème de commande de véhicule sans être trop complexe. Il est généralement utilisé en le séparant en deux parties, une pour la régulation de vitesse longitudinale et l'autre pour le contrôle latéral de position. L'objectif du contrôle longitudinal est de maintenir la vitesse du véhicule stable à une vitesse souhaitée, alors que le contrôle latéral maintient le véhicule dans la bonne direction et sur une trajectoire définie. On peut citer ici un grand nombre d'applications de contrôle concernant le véhicule dans la littérature. Par exemple, dans Tagne et al. (2013), les auteurs ont utilisé la commande par mode glissant d'ordre élevé pour la commande latérale. Les auteurs dans Fuchshumer et al. (2005) ont considéré le système comme un système plat afin de construire la loi de commande. Dans He et al. (2014), les auteurs ont proposé une stratégie de régulation de l'accélération utilisée dans le contrôle longitudinal. Un suivi robuste de vitesse longitudinale utilisant le contrôle de la traction et du freinage a été proposé dans Meihua et Tomizuka (2000). Dans Jian et al. (2014) une application de l'apprentissage a été utilisée pour choisir les paramètres d'un contrôle classique PID. Un contrôleur utilisant une fonction affine par morceaux a été utilisé dans Benine-Neto et Grand (2012) pour commander un véhicule autonome rapide tout terrain. Un contrôle combiné latéral et longitudinal de véhicule a été proposé dans Pham et al. (1994). Les auteurs dans Menhour et al. (2014) ont également proposé un contrôle combiné en appliquant des techniques d'estimation algébrique couplées avec la commande sans modèle.

Dans cette partie la commande combinée latérale-longitudinale sera introduite. Tout d'abord, on commencera par décrire le modèle non linéaire utilisé. On proposera ensuite une adaptation de l'algorithme de commande. Enfin on présentera les résultats des simulations obtenus à partir de Matlab/Simulink, MapleSim et les essais expérimentaux sur véhicule réel montrant la validation du modèle proposé et la robustesse de l'approche proposée. Dans tous les développements qui suivent, c'est un formalisme entrée-état qui a été appliqué au calcul de la table. Un observateur est donc requis pour les applications réels.

4.1.1 Pneumatique et dynamique

On utilise la formule magique de Pacejka (2006) pour calculer les forces et le moment appliqués à la roue F_x, F_y, M_z. Le glissement longitudinal est défini par (figure 4.1).

$$\sigma_x = \frac{R_{eff}\omega - V_x}{max(R_{eff}\omega, V_x)} \tag{4.1}$$

où R_{eff} est le rayon efficace, V_x est la vitesse longitudinale et ω est la vitesse angulaire de roue.

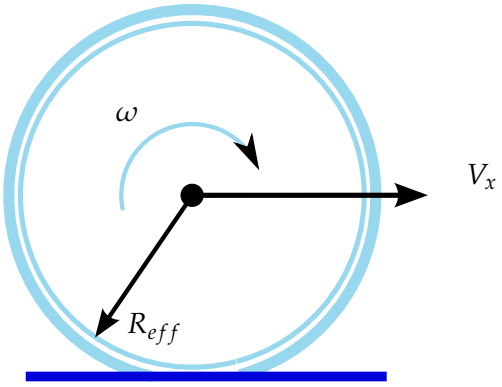


FIGURE 4.1 – *Le glissement longitudinal*

Dans les simulations, pour le cas où V_x est très faible, $max(R_{eff}\omega, V_x)$ est nul, il faut alors modifier le glissement longitudinal [1] afin de conserver la stabilité numérique :

$$\sigma_x = 2\frac{R_{eff}\omega - V_x}{V_{th} + \dfrac{max(R_{eff}\omega, V_x)^2}{V_{th}}} \tag{4.2}$$

avec V_{th} une constante de petite valeur, par exemple $V_{th} = 0.001$.
On définit la force longitudinale (glissement longitudinal pur) par :

$$F_{x0} = D_x \sin\left[C_x \arctan\left\{B_x\kappa - E_x(B_x\kappa - \arctan(B_x\kappa))\right\}\right] + S_{Vx} \tag{4.3}$$

avec

$$\begin{aligned}
\kappa &= \sigma_x + S_{Hx} \\
S_{Hx} &= b_{10}F_z + b_{11} \\
S_{Vx} &= b_{12}F_z + b_{13} \\
E_x &= (b_6F_z^2 + b_7F_z + b_8)(1 - b_9 sign(\kappa)) \\
D_x &= F_z(b_1F_z + b_2) \\
C_x &= b_0 \\
B_x &= \frac{1}{C_xD_x}(b_3F_z + b_4)F_z e^{-b_5F_z}
\end{aligned}$$

On définit la force latérale (glissement latéral pur) par :

$$F_{y0} = D_y \sin\left[C_y \arctan\left\{B_y\kappa - E_y(B_y\kappa - \arctan(B_y\kappa))\right\}\right] + S_{Vy} \tag{4.4}$$

1. http ://www.mathworks.com/

avec

$$\begin{aligned}
\kappa &= \alpha + S_{Hy} \\
S_{Hy} &= a_{11}F_z + a_{12} + a_{13}\gamma \\
S_{Vy} &= a_{14}F_z + a_{15} + \gamma(a_{16}F_z^2 + a_{17}F_z) \\
E_y &= (a_7F_z + a_8)(1 - (a_9\gamma + a_{10}sign(\kappa))) \\
D_y &= F_z(a_1F_z + a_2)(1 - a_3\gamma^2) \\
C_y &= a_0 \\
B_y &= \frac{1}{C_yD_y}a_4 \sin(2\arctan(\frac{F_z}{a_5}))(1 - a_6\|\gamma\|)
\end{aligned}$$

où $\alpha = -\arctan(\frac{V_y}{V_x})$ est l'angle de dérive de roue.
Il est aussi possible de définir les forces longitudinale et latérale (glissement combiné) en utilisant l'ellipse d'adhérence.

$$F_x = F_{x0}\sqrt{1 - \left(\frac{F_{y0}}{\mu F_z}\right)^2} \tag{4.5}$$

$$F_y = F_{y0}\sqrt{1 - \left(\frac{F_{x0}}{\mu F_z}\right)^2} \tag{4.6}$$

4.1.2 Equation dynamique

La figure 4.2 montre le schéma d'un modèle simplifié bicyclette. On y donne les différents paramètres pris en compte aussi reportés dans la table 4.1.

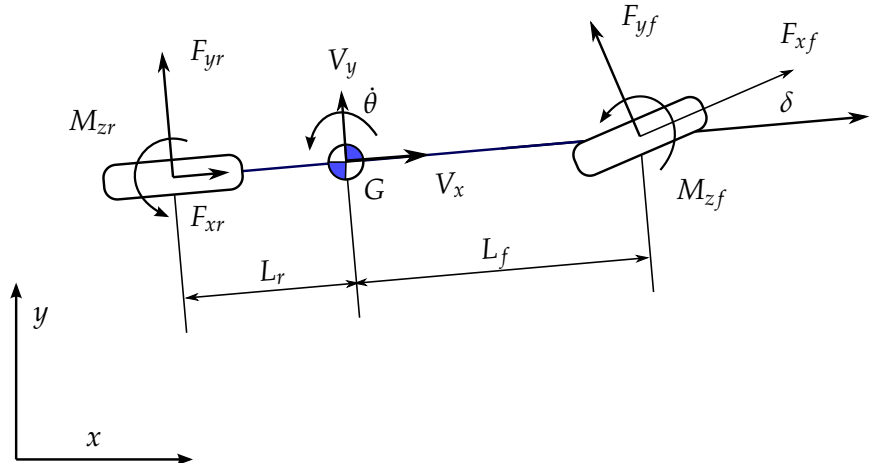


FIGURE 4.2 – *Le modèle bicyclette*

En appliquant les équations de Newton-Euler au centre de gravité G on obtient :

$$\begin{cases}
m\ddot{x} &= F_{xr}\cos\theta - F_{yr}\sin\theta + F_{xf}\cos(\theta+\delta) - F_{yf}\sin(\theta+\delta) \\
m\ddot{y} &= F_{xr}\sin\theta + F_{yr}\cos\theta + F_{xf}\sin(\theta+\delta) + F_{yf}\cos(\theta+\delta) \\
I_z\ddot{\theta} &= -F_{yr}L_r + (F_{xf}\sin\delta + F_{yf}\cos\delta)L_f + M_{zr} + M_{zf} \\
I_{yf}\ddot{\phi}_f &= -F_{xf}R_{eff} + M_f - M_{bf} \\
I_{yr}\ddot{\phi}_r &= -F_{xr}R_{eff} + M_r - M_{br}
\end{cases} \tag{4.7}$$

G	Le centre de gravité
$[x, y]$	La position du centre de gravité (m)
θ	L'angle de lacet du véhicule (rad)
$[Vx, Vy]$	La vitesse exprimée dans le repère local $(m.s^{-1})$
L_f, L_r	Les demi-empattements avant et arrière (m)
m	La masse du véhicule (kg)
I_z	Le moment d'inertie autour de l'axe z $(kg.m^2)$
δ	L'angle de braquage de roue (rad)
F_{xf}, F_{yf}, M_{zf}	Les forces et le moment pneumatique avant (N)
F_{xr}, F_{yr}, M_{zr}	Les forces et le moment pneumatique arrière (N)
I_{yf}	Le moment d'inertie de la roue avant autour de l'axe y $(kg.m^2)$
I_{yr}	Le moment d'inertie de la roue arrière autour de l'axe y $(kg.m^2)$
$\dot{\phi}_f, \dot{\phi}_r$	Les vitesses de rotation des roues $(rad.s^{-1})$
M_f, M_r	Les couples nets du moteur appliqués aux roues (Nm)
M_{bf}, M_{br}	Les couples de freinage (Nm)
R_{eff}	Le rayon efficace de la roue (m)

TABLE 4.1 – *Notation sur le véhicule*

La relation pour le passage du repère inertiel au repère local est donnée par :

$$\begin{cases} \ddot{x}\cos(\theta) + \ddot{y}\sin(\theta) &= \dot{V}_x - V_y\dot{\theta} \\ \ddot{y}\cos(\theta) - \ddot{x}\sin(\theta) &= \dot{V}_y + V_x\dot{\theta} \end{cases} \tag{4.8}$$

En combinant les équations ci-dessus et en prenant en compte les effets d'aérodynamique, cela donne :

$$\begin{cases} \dot{V}_x &= V_y\dot{\theta} + \frac{1}{m}(F_{xr} + F_{xf}\cos\delta - F_{yf}\sin\delta) - k_1 V_x^2 \\ \dot{V}_y &= -V_x\dot{\theta} + \frac{1}{m}(F_{yr} + F_{xf}\sin\delta + F_{yf}\cos\delta) - k_2 V_y^2 \\ \ddot{\theta} &= \frac{1}{I_z}(-F_{yr}L_r + (F_{xf}\sin\delta + F_{yf}\cos\delta)L_f + M_{zr} + M_{zf}) \\ I_{yf}\ddot{\phi}_f &= -F_{xf}R_{eff} + M_f - M_{bf} \\ I_{yr}\ddot{\phi}_r &= -F_{xr}R_{eff} + M_r - M_{br} \end{cases} \tag{4.9}$$

L'angle de dérive de la roue avant est calculé par :

$$\alpha_f = \delta - \arctan(\frac{V_y + L_f\dot{\theta}}{V_x}) \tag{4.10}$$

Pour la roue arrière on a :

$$\alpha_r = -\arctan(\frac{V_y - L_r\dot{\theta}}{V_x}) \tag{4.11}$$

Le glissement longitudinal s'exprime comme :

$$\sigma_r = \sigma_x(R_{eff}, \dot{\phi}_r, V_x) \tag{4.12}$$

$$\sigma_f = \sigma_x(R_{eff}, \dot{\phi}_f, V_x, \delta, \dot{\theta}) \tag{4.13}$$

Les forces verticales pour les roues avant et arrière sont données par :

$$F_{zf} = mg\frac{L_r}{L_r + L_f} \tag{4.14}$$

$$F_{zr} = mg\frac{L_f}{L_r + L_f} \tag{4.15}$$

Avec tous ces éléments, on peut constituer l'équation d'état du système dont le vecteur d'état est le suivant :

$$\mathbf{x} \quad = \quad [V_x, V_y, \dot{\theta}, \dot{\phi}_r, \dot{\phi}_f]^T$$

L'entrée du système est constituée de l'angle de braquage δ, des couples appliqués aux roues $M_r,\ M_f$ et des couples de freinage $M_{br},\ M_{bf}$. Le couple appliqué aux roues dépend du type de la voiture. Par exemple pour une voiture de type RWD [2], $M_f = 0$.

4.1.3 Modèle véhicule complet

Il existe de nombreux outils et environnements graphiques avec lesquels il est possible de modéliser la dynamique du véhicule en utilisant la théorie des systèmes multi-corps, comme par exemple SimMechanic de MathWorks ou MapleSim de MapleSoft. Dans ce travail, nous avons utilisé un modèle de véhicule à 4-roues construit sous MapleSim représenté sur la figure 4.3. Ce modèle utilise les bibliothèques "Tires" pour définir un modèle de Pacejka pour les pneus et "Drive Line" pour modéliser un différentiel complet pour une transmission réaliste du couple aux roues. Il contient quatre suspensions simples et des freins sur chaque roue. Les entrées du système sont les suivantes : le signal de braquage, le signal de freinage et le couple du moteur à la transmission mécanique. Des capteurs virtuels ont été utilisés pour obtenir l'état du véhicule : vitesse, position, vitesse angulaire, etc. Le modèle bicyclette équivalent a été construit en se basant sur les paramètres utilisés pour le modèle 4 roues. Ce modèle 4 roues a été exporté dans l'environnement Simulink pour réaliser les simulations.

4.1.4 Suivi d'une trajectoire

La figure 4.4 présente le modèle bicyclette et son interaction avec de la courbure de la route. La ligne centrale de la route est représentée par un repère Frenet G_r, i_s, j_s. La distance entre le modèle et cette ligne est notée par $d = GG_r$. On note aussi l'angle entre

2. Rear Wheel Drive

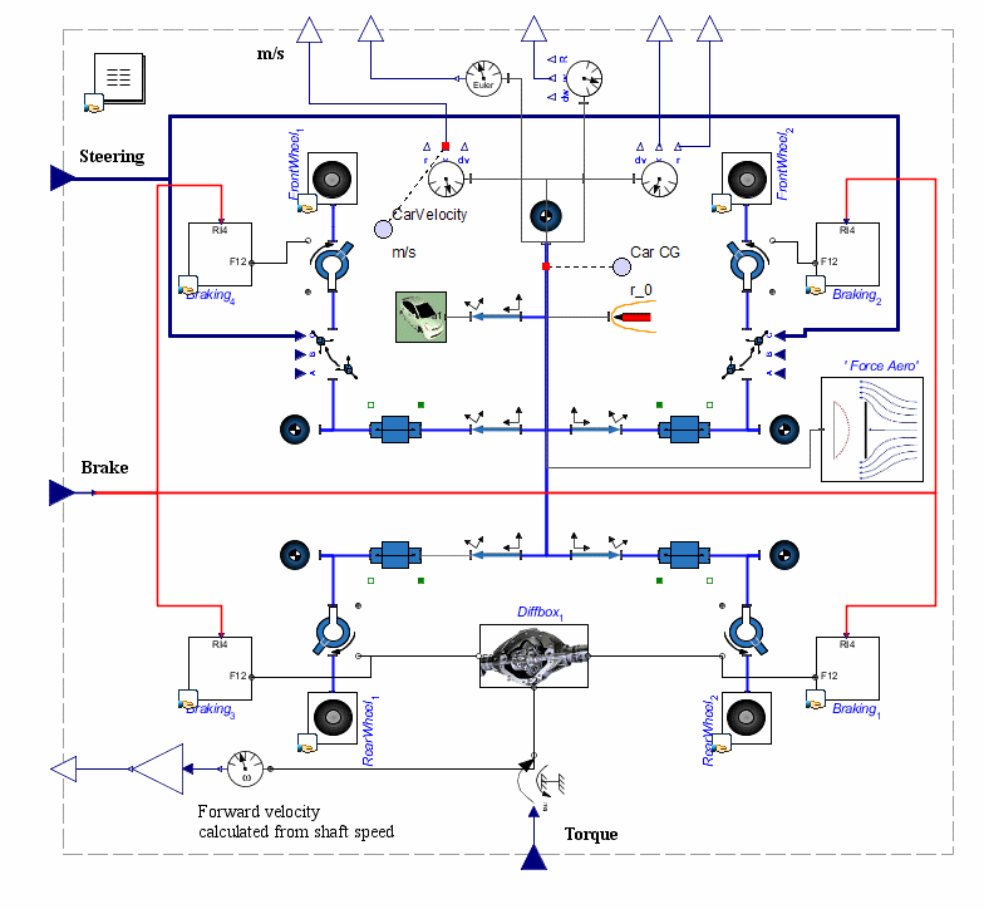


FIGURE 4.3 – *Le modèle de véhicule 4 roues de type RWD sous MapleSim*

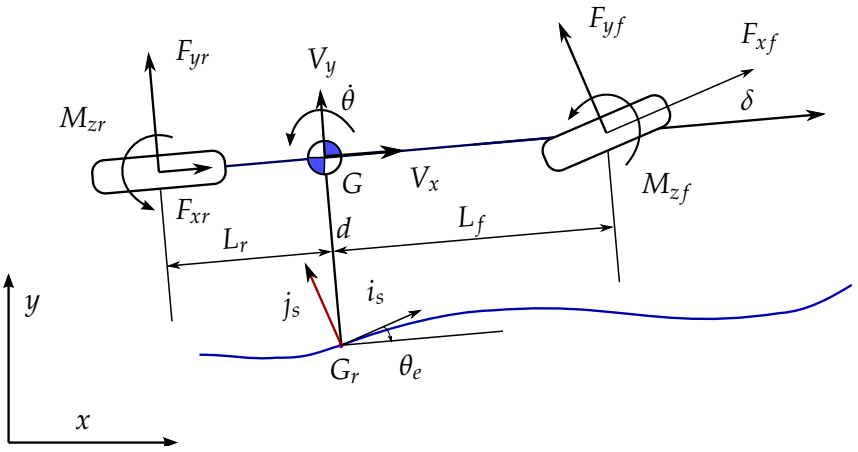


FIGURE 4.4 – *Le modèle bicyclette sur la route*

le véhicule et la route par θ_e, le rayon de courbure de la route par R_r. La dynamique latérale utilisée pour réaliser le suivi de trajectoire est donnée par :

$$\begin{cases} \dot{d} &= V_x \sin\theta_e + V_y \cos\theta_e \\ \dot{\theta_e} &= \dot{\theta} - \dfrac{V_x \cos\theta_e - V_y \sin\theta_e}{R_r - d} \end{cases} \tag{4.16}$$

4.1.5 Mise en œuvre de l'algorithme de commande

Dans cette section on considère un modèle de type RWD où la puissance du moteur est transmise seulement aux roues arrières. Pour simplifier le problème de commande, nous négligeons la dynamique de la roue avant. Le système (4.9) devient (4.17).

$$\begin{cases} \dot{V}_x &= V_y\dot{\theta} + \frac{1}{m}(F_{xr} + F_{xf}\cos\delta - F_{yf}\sin\delta) - k_1 V_x^2 \\ \dot{V}_y &= -V_x\dot{\theta} + \frac{1}{m}(F_{yr} + F_{xf}\sin\delta + F_{yf}\cos\delta) - k_2 V_y^2 \\ \ddot{\theta} &= \frac{1}{I_z}(-F_{yr}L_r + (F_{xf}\sin\delta + F_{yf}\cos\delta)L_f + M_{zr} + M_{zf}) \\ I_{yr}\ddot{\phi}_r &= -F_{xr}R_{eff} + M_r \end{cases} \tag{4.17}$$

Le vecteur d'état et de sortie du système sont $\mathbf{x} = [V_x, V_y, \theta, \dot{\theta}, \dot{\phi}_r]^T$ et $\mathbf{y} = [V_x, V_y, \theta, \dot{\theta}]^T$ respectivement. Le vecteur d'entrée est $\mathbf{u} = [\delta, M_r]^T$. On peut utiliser $\mathbf{x} = [V_x, V_y, \dot{\theta}, \dot{\phi}_r]^T$ et $\mathbf{y} = [V_x, V_y, \dot{\theta}]^T$ pour réduire la taille de la table. La trajectoire de référence dépend du temps et est donnée par $\mathbf{y}_c = [V_{xd}, V_{yd}, \dot{\theta}_d]^T$. La table 4.2 reprend les paramètres retenus pour réaliser les simulations.

Variable	Minimum	Maximum	Nombre de points
V_x	$0m.s^{-1}$	$40m.s^{-1}$	11
V_y	$-1.0m.s^{-1}$	$1.0m.s^{-1}$	11
$\dot{\theta}$	$-6.0rad.s^{-1}$	$6.0rad.s^{-1}$	11
d	$-50m$	$50m$	11
θ_e	$-\pi$	π	41
M_r	-1000	1000	5
δ	$-\frac{\pi}{6}$	$\frac{\pi}{6}$	5

TABLE 4.2 – *Paramètres des variables*

Le système de référence interne est un système linéaire du premier ordre pour toutes les sorties V_x, V_y et $\dot{\theta}$, dont la fonction de transfert est de la forme :

$$H(p) = \frac{1}{\tau_s.p + 1}$$

Si on utilise θ comme sortie, il faudra utiliser un système de référence du second ordre pour cette variable. La vitesse latérale désirée V_{yd} et la vitesse angulaire du véhicule $\dot{\theta}_d$ sont calculées comme références internes grâce aux équations (4.16). Le vecteur d'état et de sortie de ce système sont $\mathbf{x} = [d, \theta_e]^T$ et $\mathbf{y} = [d, \theta_e]^T$ respectivement, avec le vecteur d'entrée $\mathbf{u} = [V_y, \dot{\theta}]^T$, V_x utilisés comme des entrées internes. A partir de la table de prédiction pour ce système, les valeurs de V_{yd} et $\dot{\theta}_d$ peuvent être prédites, basées sur le vecteur d'état au pas courant et la cible à l'étape suivante. Le point désiré pour le

système bicyclette sera rassemblé sous la forme $\mathbf{y}_c = [V_{xd}, V_{yd}, \dot{\theta}_d]^T$. L'algorithme modifié est montré sur la figure 4.5. Des blocs ajoutés sont : bloc **B** pour l'intégration des valeurs $\mathbf{y}_{ir} = [d, \theta_e]^T$ à partir du vecteur $\mathbf{y} = [V_x, V_y, \dot{\theta}]^T$, bloc $Cherche_{traj}$ utilisé pour calculer $[V_{yd}, \dot{\theta}_d]^T$. Le bloc $Cherche_{traj}$ contient exactement l'algorithme montré sur la figure 1.1 appliqué au système (4.16). L'état désiré et les valeurs des entrées sont saturés car la table de prédiction est construite par discrétisation dans l'espace $\mathcal{S} \times \mathcal{U}$.

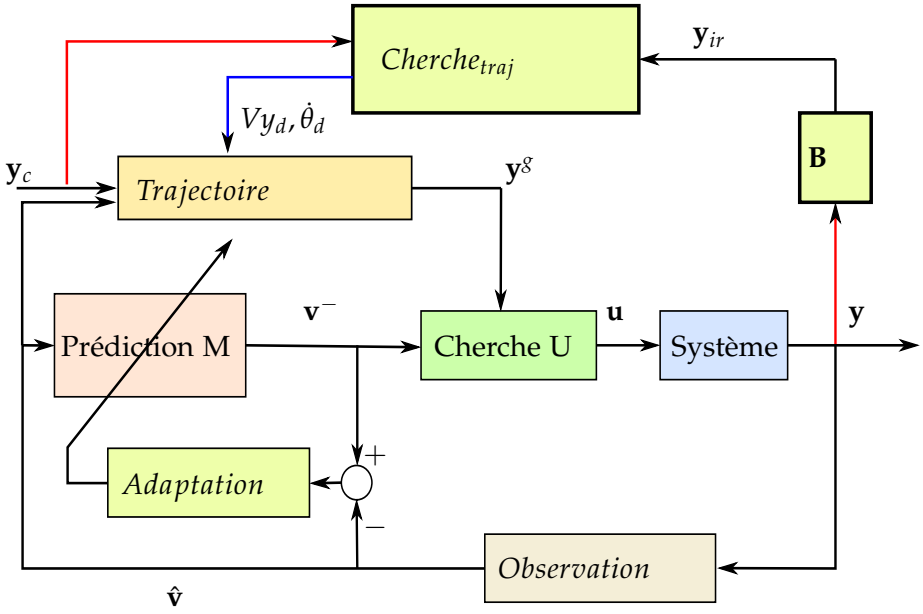


FIGURE 4.5 – *Le système complet pour le véhicule*

4.1.6 Résultat de simulation

Dans cette section des résultats de simulations obtenus à partir de Matlab et Simulink sont présentés, voir Nguyen et al. (2015b). On initialise le véhicule à la condition initiale $V_x(0) = 0, V_y(0) = 0, \theta(0) = 0, \dot{\theta}(0) = 0$. L'angle et la distance par rapport à la ligne centrale de route sont 0 et $3m$ respectivement. A partir de $t_0 = 10\ s$ le contrôle est utilisé. La référence est un système linéaire du premier ordre pour chaque axe $\dot{\theta}$ et Vy avec la constante de temps $\tau_s = 30\ s$ pour $\dot{\theta}$. Pour V_x nous avons utilisé $\tau_s = 5\ s$. Le couple à la transmission M_r est dans l'intervalle $[-1000; 1000]\ Nm$ et l'angle est saturé à $\frac{\pi}{6}\ rad$. Quand M_r est inférieur à zero, le freinage est activé. Pour les variables d'état, les intervalles sont $|V_x| \leq 40.0$, $|V_y| \leq 1.0$ et $|\dot{\theta}| \leq 6.0$. Pour valider le modèle et tester la robustesse du contrôleur proposé, nous avons généré la table de prédiction en utilisant le modèle bicyclette et réalisé des simulations sur le véhicule à 4-roues modélisé sous

MapleSim avec des conditions initiales équivalentes. Les paramètres utilisés pour la simulation sont donnés dans la table 4.3.

			Paramètre
m	1408	kg	Masse du modèle
I_z	1950	$kg.m^2$	Moment d'inertie du modèle
I_{yf}, I_{yr}	0.78	$kg.m^2$	Moment d'inertie de roue
L_r	1.490	m	Demi-empattement arrière
L_f	1.482	m	Demi-empattement avant
R_{eff}	0.344	m	Rayon efficace de roue
k_1	0.01	m^{-1}	Coefficient aérodynamique sur l'axe x
k_2	0	m^{-1}	Coefficient aérodynamique sur l'axe y

TABLE 4.3 – *Paramètres du véhicule*

Dans la première simulation, le véhicule suit une ligne droite puis il est stabilisé sur une trajectoire circulaire de rayon $R_r = 20\ m$ avec une vitesse longitudinale constante $V_{xd} = 10\ m.s^{-1}$. La figure 4.6 montre la position du véhicule et son équivalent bicyclette au cours de la trajectoire pour rejoindre le point de consigne. La référence est suivie et atteinte sans erreur statique, montrant la robustesse du contrôleur vis à vis des perturbations extérieures et des erreurs de modélisation. La figure 4.7 montre les distances du modèle véhicule et l'équivalent bicyclette par rapport à la ligne centrale de la route. Ces distances tendent vers zéro sans erreur statique.

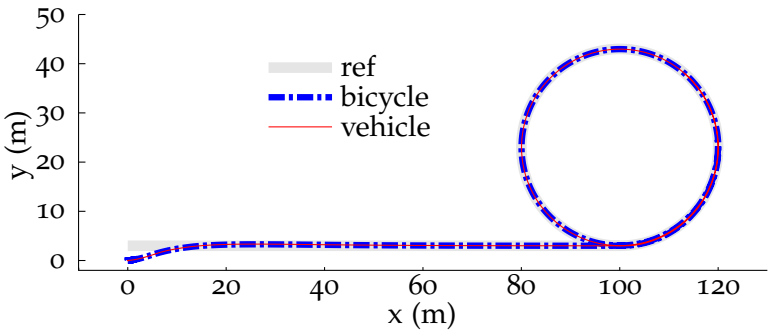


FIGURE 4.6 – *Suivi d'un cercle - xOy plan*

Les figures 4.8, 4.9 et 4.10 montrent la vitesse, l'angle de braquage et les moments appliqués aux roues respectivement. On observe sur la figure 4.8 que la vitesse latérale tend vers une vitesse constante de petite valeur car la trajectoire est un cercle. La vitesse longitudinale converge vers la valeur souhaitée $V_{xd} = 10\ m.s^{-1}$. L'angle de braquage devient constant pour le suivi du cercle (voir figure 4.9). Les figures 4.8 et 4.10 montrent également que les freins sont activés lorsque la vitesse longitudinale dépasse la valeur souhaitée. De plus, après que la vitesse longitudinale se soit stabilisée, les moments ap-

pliqués aux roues visibles sur la figure 4.10 sont non nuls car ils doivent compenser la force aérodynamique longitudinale.

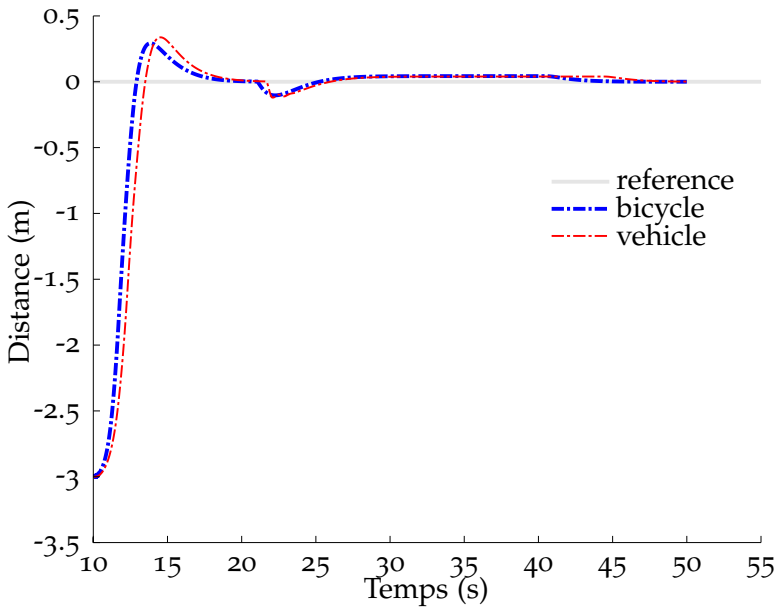


FIGURE 4.7 – *Distance à la route et sa référence*

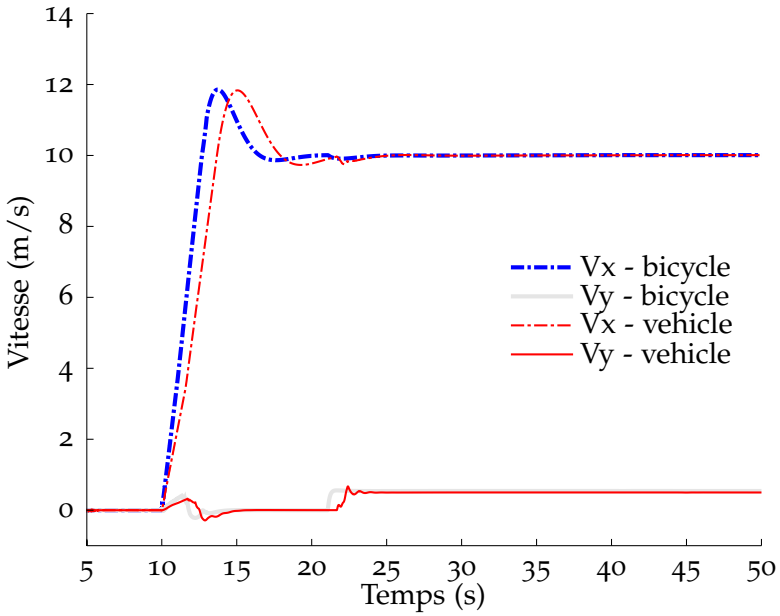


FIGURE 4.8 – *Vitesse longitudinale et latérale*

Dans la deuxième simulation, on souhaite suivre une trajectoire plus complexe (la courbe grise et claire sur la figure 4.11) avec plus de perturbations et des obstacles à éviter. La variation de la courbure de la trajectoire devrait provoquer des variations de la force verticale à cause du système des suspensions de véhicule (dans MapleSim) non modélisé dans le modèle bicyclette.
La figure 4.11 montre la trajectoire et la figure 4.12 montre la distance à la route. Quand

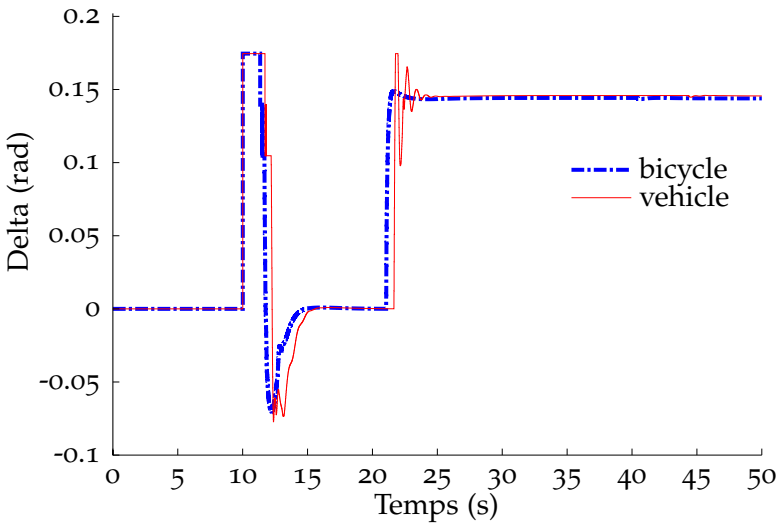


FIGURE 4.9 – *Angle de braquage*

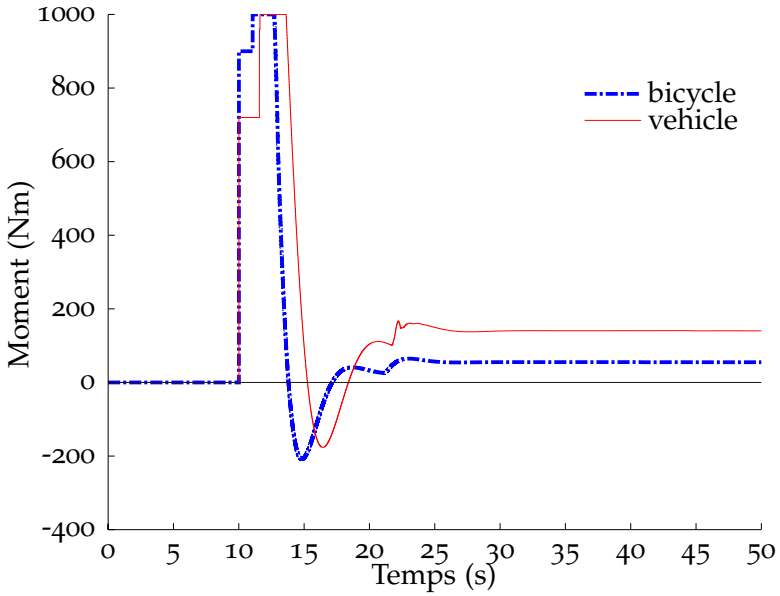


FIGURE 4.10 – *Moment en Nm*

la position x est dans l'intervalle $[150, 230]$ m, on fait faire au véhicule un écart de $d = 3\ m$ pour éviter un obstacle. Toutes ces figures montrent que la dynamique du véhicule est plus complexe que l'équivalent bicyclette, avec ainsi un temps de réponse plus long. Pour autant la stabilité est conservée.

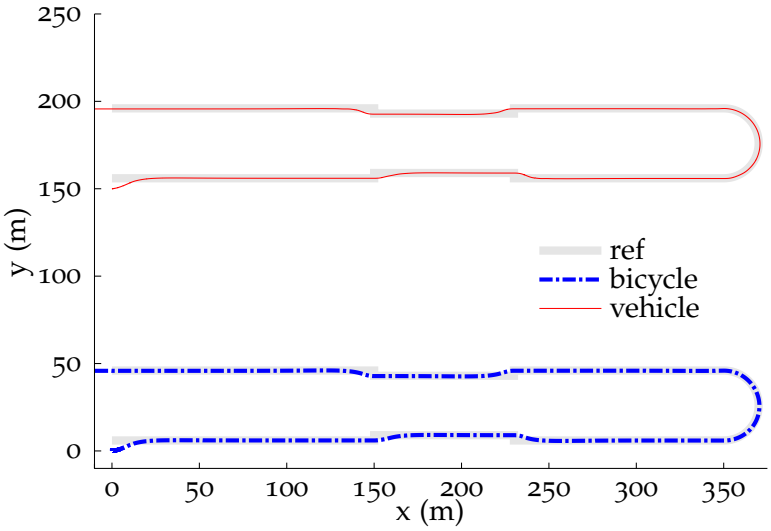


FIGURE 4.11 – *Trajectoire et Référence dans - xOy plan*

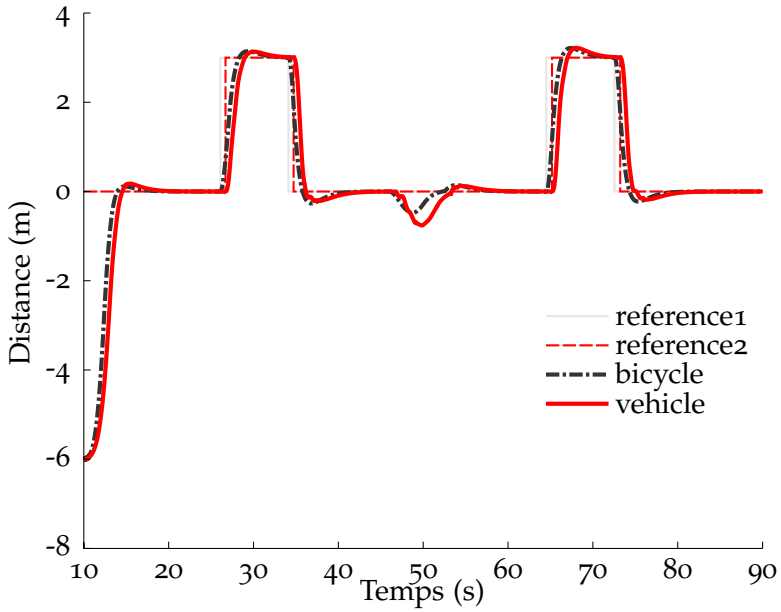


FIGURE 4.12 – *Distance à la route et sa référence*

La figure 4.13 montre les vitesses longitudinale et latérale du véhicule et de l'équivalent bicyclette. On constate qu'il existe des cas où il est nécessaire d'activer les freins pour conserver la vitesse longitudinale. Clairement, on trouve qu'il existe les valeurs négatives sur les courbes des valeurs de commande montrées sur la figure 4.14.

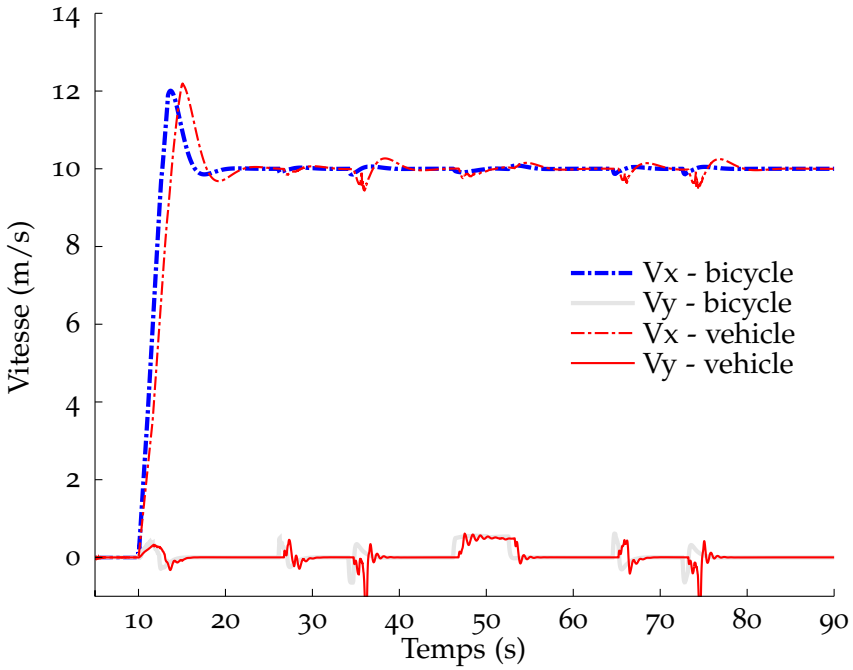


FIGURE 4.13 – *Vitesse longitudinale et latérale*

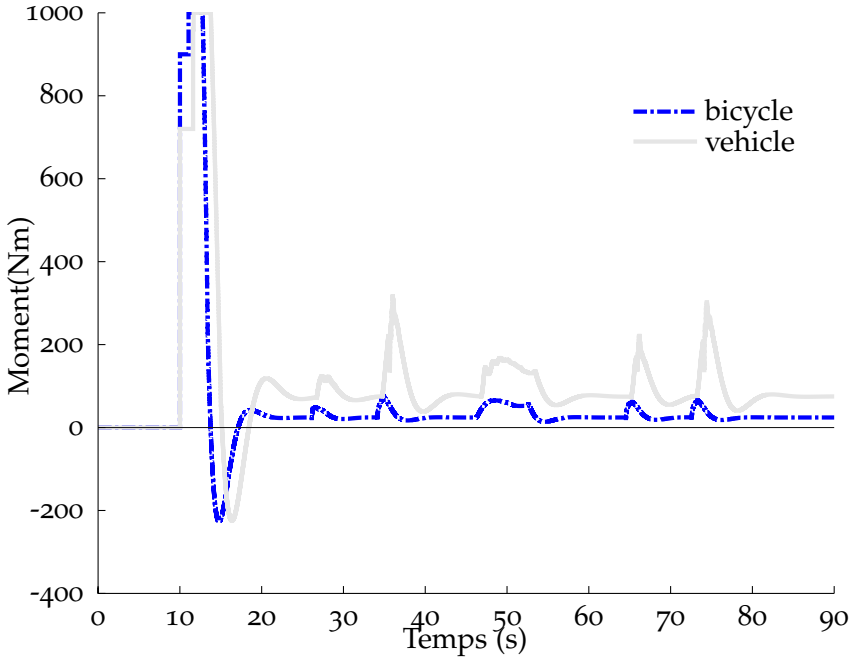


FIGURE 4.14 – *Moment en Nm*

4.2 Commande d'une voiture en drift

La conduite de la voiture en manœuvres agressives, par exemple en drift, est une technique avancée qui est un défi intéressant pour la synthèse de la commande. La commande nécessite d'atteindre la meilleure performance dans un mode fortement non linéaire. Dans la littérature on trouve des travaux concernant ce problème incluant l'étude de la dynamique du véhicule, l'optimisation de la structure de commande et son application à un véhicule en drift. En conduite normale, le glissement longitudinal σ et l'angle de dérive β sont petits : $\sigma \in [0, 0.1]$ et β ne dépasse pas $10°$. En drift, σ et β sont les plus grands possibles et la vitesse latérale ne doit pas être négligée. Pour plus de détails sur la dynamique du véhicule en drift, on peut consulter les travaux de Abdulrahim (2006). Du point vue de la commande, le point d'équilibre est instable. Quelques approches ont été proposées. Les auteurs Velenis et Tsiotras (2005) ont proposé une stratégie pour générer la trajectoire. Velenis et al. (2009; 2010), Voser et al. (2010), Velenis et al. (2011), Chaichaowarat et Wannasuphoprasit (2013), Olofsson et al. (2013), Ronnapee et Witaya (2015) ont analysé le point d'équilibre afin de construire une commande optimale LQR pour linéariser le système dynamique autour de ce point en utilisant différents modèles du véhicules. Katriniok et Abel (2011) ont proposé une commande prédictive basée sur le système avec des paramètres variants LTV [3]. Une commande par apprentissage a été proposée dans Lau (2011) pour une voiture à échelle réduite. Tavernini et al. (2013) a étudié le problème pour optimiser le temps de parcours.

Dans cette section, la commande développée sera adaptée pour contrôler un véhicule en drift. La table de prédiction est construite en se basant sur un modèle dynamique de type bicyclette. Des simulations seront introduites pour vérifier la capacité de robustesse et de stabilité de la commande.

4.2.1 Equation dynamique

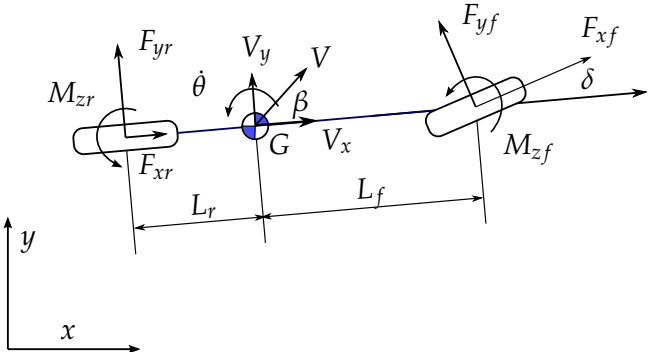


FIGURE 4.15 – *Modèle bicyclette*

La figure 4.15 montre le modèle bicyclette en utilisant $\mathbf{x} = [V, \beta, \dot{\theta}]^T$ comme vecteur

3. Linear Time Varying system

d'état du système. En appliquant les équations de Newton-Euler, on peut représenter ce système dans l'espace entrée-état comme suit :

$$\begin{cases} \dot{V} &= \frac{1}{m}(F_{xr}\cos\beta + F_{yr}\sin\beta + F_{xf}\cos(\delta-\beta) - F_{yf}\sin(\delta-\beta)) - kV^2 \\ \dot{\beta} &= -\dot{\theta} + \frac{1}{mV}(F_{yr}\cos\beta - F_{xr}\sin\beta + F_{xf}\sin(\delta-\beta) + F_{yf}\cos(\delta-\beta)) \\ \ddot{\theta} &= \frac{1}{I_z}(-F_{yr}L_r + (F_{xf}\sin\delta + F_{yf}\cos\delta)L_f + M_{zr} + M_{zf}) \\ I_{yr}\ddot{\phi}_r &= -F_{xr}R_{eff} + M_r \\ I_{yf}\ddot{\phi}_f &= -F_{xf}R_{eff} + M_f \end{cases} \tag{4.18}$$

			Paramètre
m	1450	kg	Masse du modèle
I_z	2741	$kg.m^2$	Moment d'inertie du modèle
I_{yf}, I_{yr}	1.8	$kg.m^2$	Moment d'inertie de roue
L_r	1.590	m	Demi-empattement arrière
L_f	1.19	m	Demi-empattement avant
R_{eff}	0.30	m	Rayon efficace de roue
k_1	0.01	m^{-1}	Coefficient aérodynamique sur l'axe x
k_2	0	m^{-1}	Coefficient aérodynamique sur l'axe y

TABLE 4.4 – *Paramètres du véhicule*

Afin de réaliser une comparaison, les paramètres du véhicule donnés dans la table 4.4 sont choisis identiques à ceux utilisés dans Velenis et al. (2009). Les vitesses longitudinales et latérales pour chaque roue sont calculées par :

$$V_{Fx} = V\cos(\beta-\delta) + \dot{\theta}L_f\sin(\delta), \quad V_{Fy} = V\sin(\beta-\delta) + \dot{\theta}L_f\cos(\delta)$$
$$V_{Rx} = V\cos(\beta), \quad V_{Ry} = V\sin(\beta) - \dot{\theta}L_r$$

La formule magique s'écrit sous la forme :

$$F(\kappa) = F_z D\sin[C\arctan\{B\kappa - E(B\kappa - \arctan(B\kappa))\}]$$

pour toutes les roues et prend pour coefficients $B = 7$, $C = 1.45$, $D = 1.0$, $E = 0$. Les forces longitudinales et latérales combinées sont montrées sur les figures 4.16 et 4.17. L'angle de dérive nécessaire pour faire un drift doit être situé dans la zone non linéaire qui génère une force latérale plus grande que lors de la conduite normale. Le glissement longitudinal est également dans la zone non linéaire, $|\sigma| > 10\%$.

4.2.2 Mise en œuvre de l'algorithme de commande

Nous comparons notre commande avec une commande quadratique LQR, telle que décrite dans Velenis et al. (2009; 2010). Une analyse mathématique du point d'équilibre

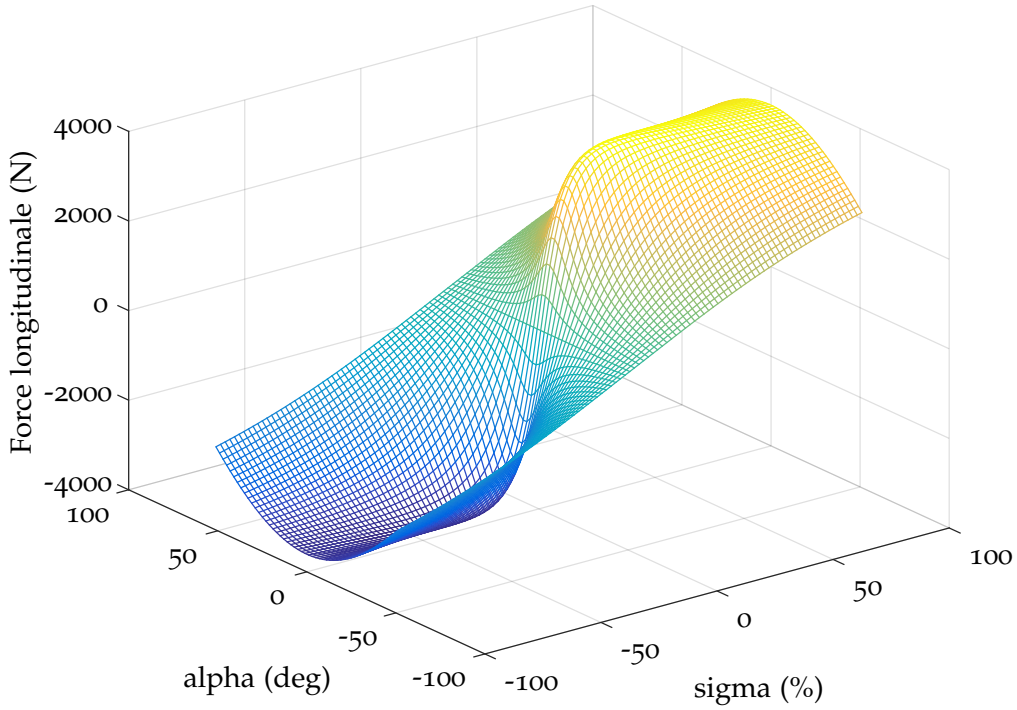


FIGURE 4.16 – *La force longitudinale avec* $Fz = 3.779kN$

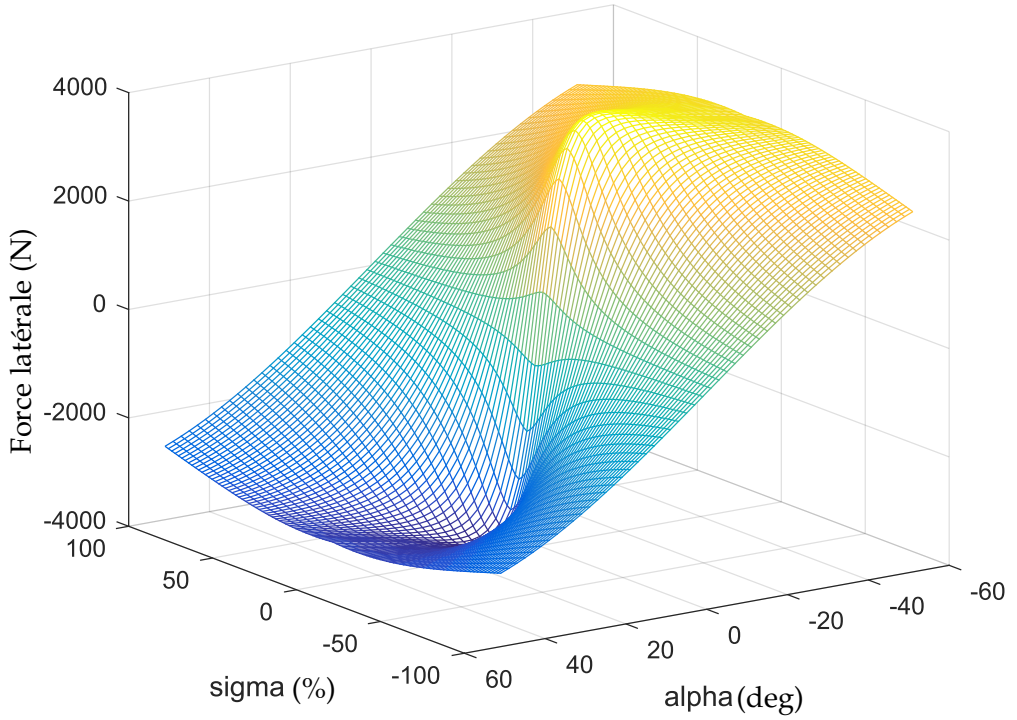


FIGURE 4.17 – *La force latérale avec* $Fz = 3.779kN$

des équations (4.18) peut-être trouvée dans les références ci-dessus. Dans ces travaux, la solution du point d'équilibre est déterminée sous Matlab utilisant l'outil pour le calcul symbolique[4]. Pour ce faire, on néglige la force aérodynamique longitudinale. Ce point d'équilibre est utilisé pour la commande quadratique LQR.

On s'intéresse maintenant à représenter les équations (4.18) par une table de prédiction en temps discret. On prend $[V, \beta, \dot{\theta}, \dot{\phi}_r, \dot{\phi}_f]^T$ comme vecteur d'état et $[\delta, M_r, M_f]^T$ pour le vecteur des entrées. On suppose que les couples M_r et M_f peuvent être positifs ou négatifs, dans l'intervalle $[-4000;\ 4000]\ Nm$ et l'angle de braquage est saturé pour $\frac{\pi}{4}\ rad$. La vitesse totale V se situe dans l'intervalle $[0;\ 20]\ m/s$. L'angle de dérive du modèle bicyclette β, on a utilisé β dans $[-\pi/2; \pi/2]$. Les vitesses de rotation des roues sont limitées dans $[0,\ 120]\ rad/s$. La vitesse angulaire $\dot{\theta}$ est saturée à 2 rad/s. Le temps d'échantillonnage est $Ts = 0.01\ s$. Une fois la table de prédiction obtenue, la commande proposée dans le chapitre 1 sera appliquée pour stabiliser $[V, \beta, \dot{\theta}]^T$ vers le point d'équilibre.

La référence "Trajectoire" dans le schéma de la commande proposée (la figure 1.1) est un système linéaire du premier ordre pour toutes les sorties V, β et $\dot{\theta}$, dont la fonction de transfert est de la forme :

$$H(p) = \frac{1}{\tau_s.p + 1} \tag{4.19}$$

4.2.3 Résultat de simulation

Les valeurs des entrées du point d'équilibre du système sont utilisées comme des valeurs de références dans les figures suivantes pour comparaison. Dans les deux essais suivants, les constantes de temps pour la référence interne (4.19) sont $\tau_s = 10$ pour V, β et $\tau_s = 30$ pour $\dot{\theta}$.

Dans la première simulation, le véhicule est initialisé avec des conditions initiales suffisantes pour faire du drift. La vitesse initiale est $V = 8.4\ m.s^{-1}$ et l'angle de dérive est égal à $-25°$ pour stabiliser le véhicule au point $V_e = 7\ m.s^{-1}, \beta_e = -51°$. Le diamètre de la trajectoire est $d_e = 14.0\ m$, la vitesse angulaire $\dot{\theta}_e = 1\ rad.s^{-1}$.

La figure 4.18 illustre la trajectoire du véhicule, la courbe en pointillé présente la trajectoire par la commande LQR et la courbe en continu rouge générée par la commande proposée. Les courbes ne sont pas les mêmes car les deux méthodes ont des performances dynamiques différentes. Les figures 4.19 et 4.20 montrent que la vitesse et l'angle de dérive tendent vers le point d'équilibre. Il y a un grand dépassement sur la vitesse générée par notre commande (Propose) car le temps d'adaptation du compensateur d'erreur et la référence linéaire ne donnent pas la meilleure performance. Il y a trois constantes de temps τ_s à régler.

L'angle de braquage nécessaire dans cet essai est présenté sur la figure 4.21. On constate que la commande proposée peut atteindre la valeur précise de l'angle de bra-

4. http ://fr.mathworks.com/products/symbolic/

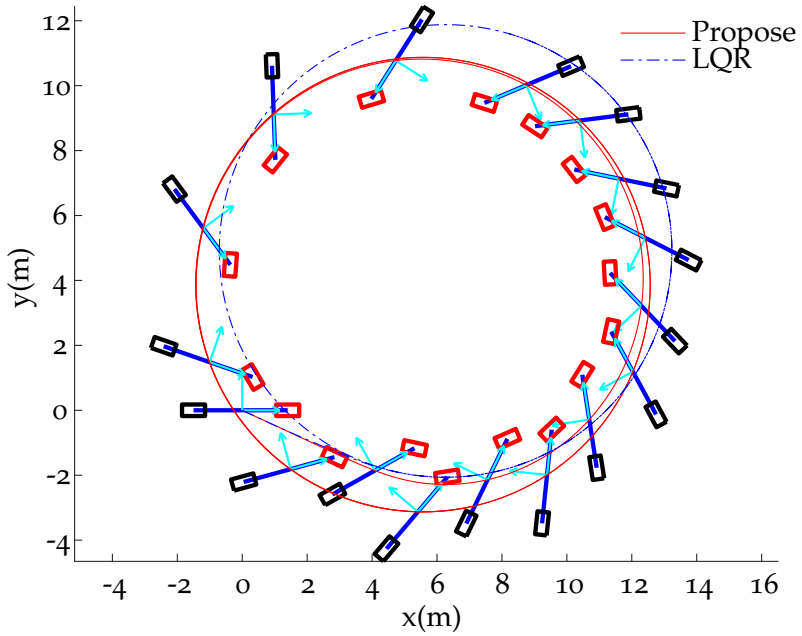


FIGURE 4.18 – *Drift, Simulation 1 : Trajectoire d'un modèle bicyclette en drift, la courbe en pointillé présente la trajectoire par la commande LQR et l'autre courbe en continue rouge est générée par la commande proposée.*

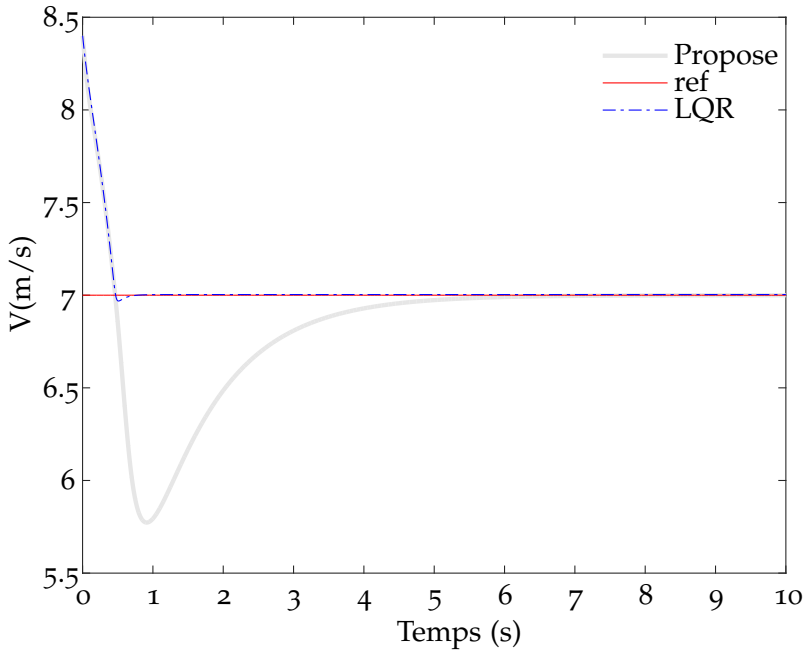


FIGURE 4.19 – *Drift, Simulation 1 : Vitesse totale et sa référence*

quage par ailleurs définie analytiquement pour la commande LQR. De même, la figure 4.22 montre les couples appliqués au roues. Les couples pour la commande développée sont bruités car elle n'assure pas une continuité locale complète lorsqu'on change de vertex dans la table. Si la commande optimale LQR nécessite Les valeurs du

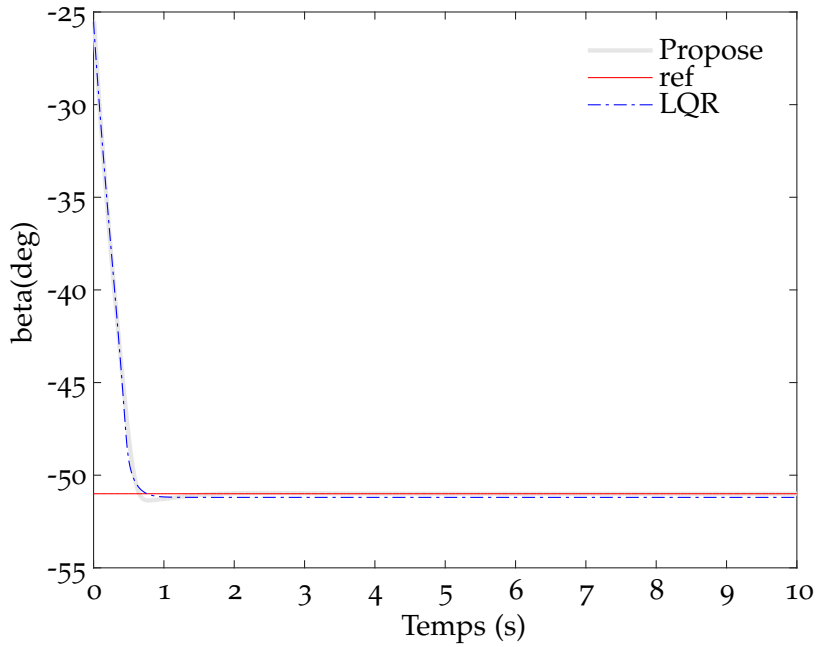


FIGURE 4.20 – *Drift, Simulation 1 : Angle de dérive β*

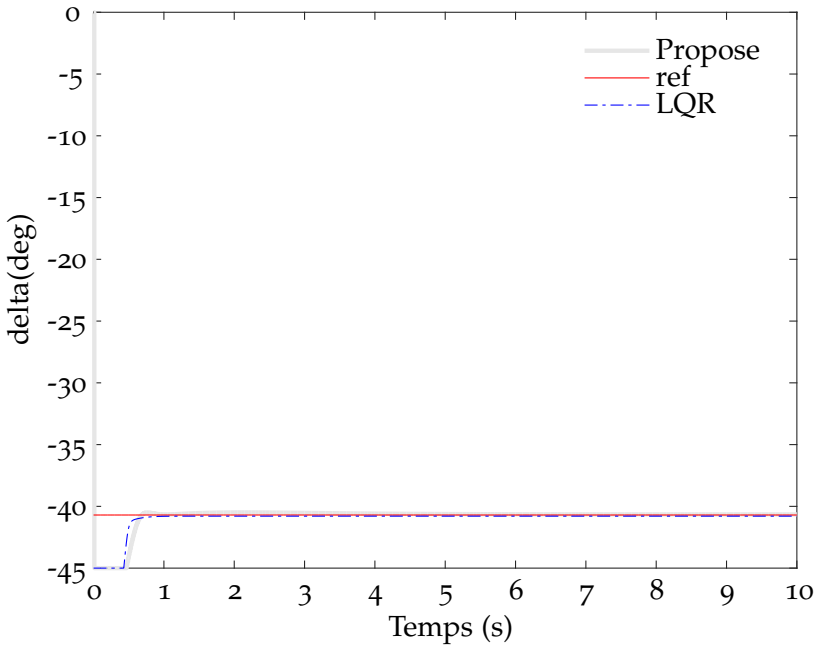


FIGURE 4.21 – *Drift, Simulation 1 : Angle de braquage, la commande proposée (Propose) donne la valeur de l'angle de braquage désiré -40.7°*

point d'équilibre, la commande proposée atteint d'elle même ces valeurs. Il existe une erreur statique pour β avec la commande LQR. Pour la performance au niveau de la vitesse totale V, la figure 4.19 montre que la commande optimale LQR répond plus vite que la commande proposée (Propose) parce que sa performance dépend du système de référence interne et du filtre définie sur la compensation de l'erreur.

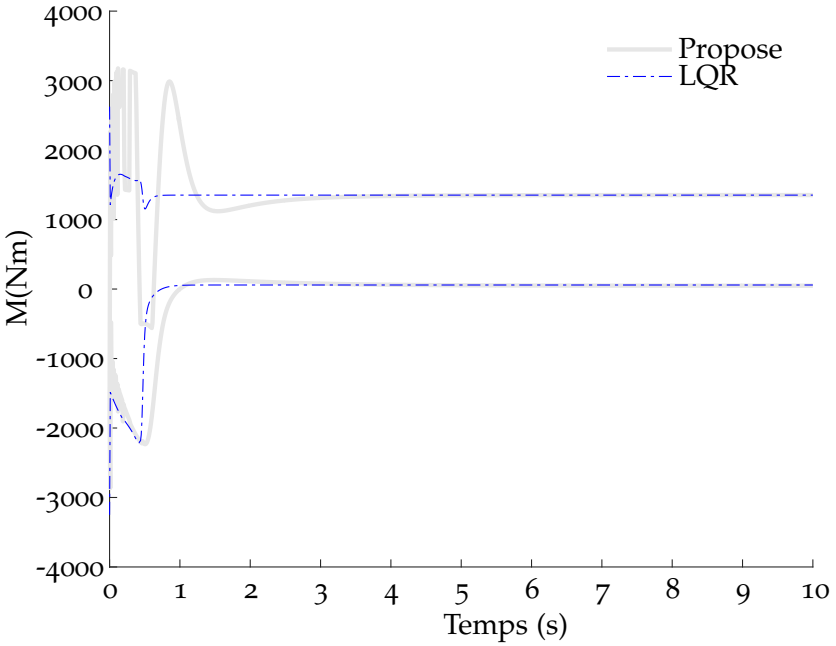


FIGURE 4.22 – *Drift, Simulation 1 : Moments appliqués aux roues*

Dans la deuxième simulation, le modèle bicyclette est initialisé à la vitesse initiale $V = 8.4m/s$, l'angle de dérive à $-20°$ et la vitesse angulaire à 1.2 rad/s pour stabiliser le modèle bicyclette autour du point $V_e = 7m.s^{-1}$, $\beta_e = -10.4°$, d'un diamètre $d_e = 14.0\ m$, de la vitesse angulaire $\dot{\theta}_e = 1rad.s^{-1}$. Les états du système sont reportés sur les figures 4.23 et 4.24.

Les valeurs de commande sont illustrées sur les figures 4.25 et 4.26. On remarque que les couples appliqués aux roues sont positifs pour la commande LQR autour du temps $t = 1\ s$. C'est-à-dire qu'il faudrait utiliser une voiture de type $4WD$ ou AWD avec une traction appliquée à toutes les roues. Mais la commande proposée (Propose) a donné un résultat réalisable pour la voiture RWD. On applique un freinage aux roues avants.

La figure 4.27 affiche les trajectoires obtenues par ces deux méthodes : la courbe en pointillés est obtenue pour la commande LQR. On voit que les comportements ne sont pas identiques. La commande optimale LQR peut atteindre sa consigne plus rapidement. Mais comme on l'a déjà souligné, notre approche n'a pas besoin des valeurs du point d'équilibre. Par contre elle a besoin de temps pour converger. Ce phénomène est également observé sur les figures 4.23, 4.24 et 4.25.

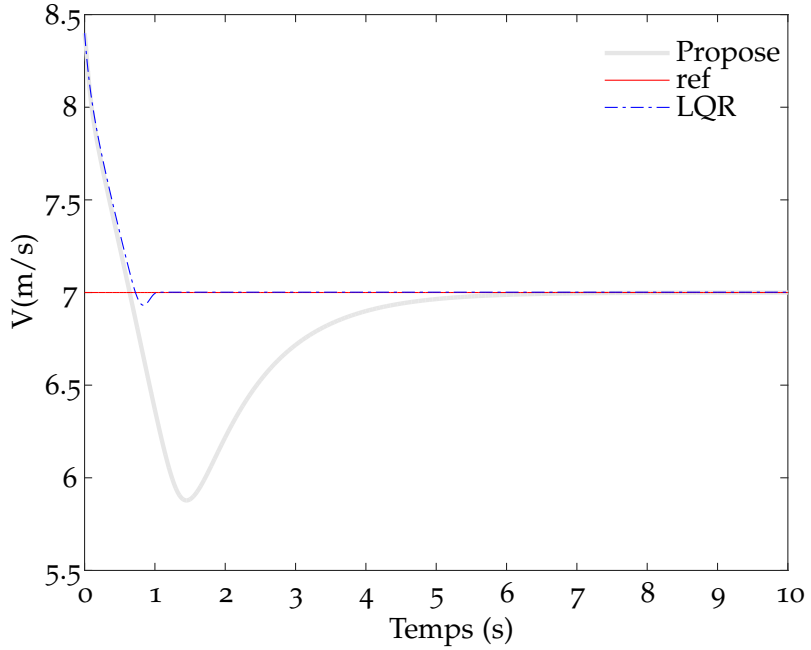


FIGURE 4.23 – *Drift, Simulation 2 : Vitesse totale et sa référence*

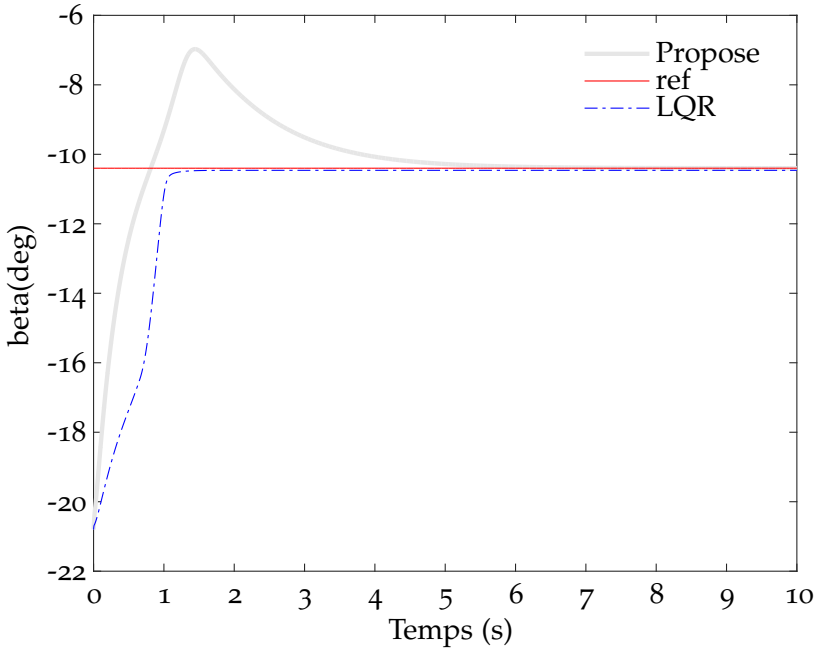


FIGURE 4.24 – *Drift, Simulation 2 : Angle de dérive β*

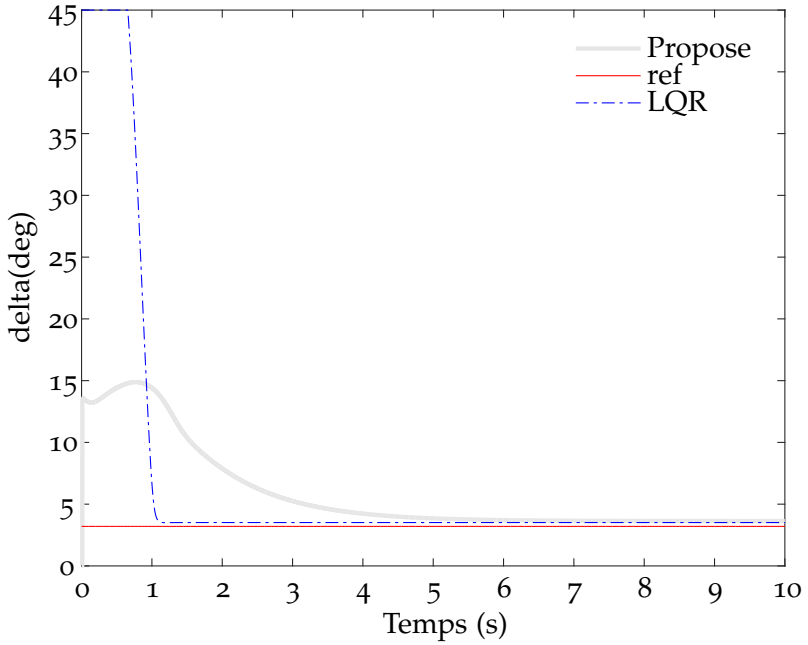


FIGURE 4.25 – *Drift, Simulation 2 : Angle de braquage δ*

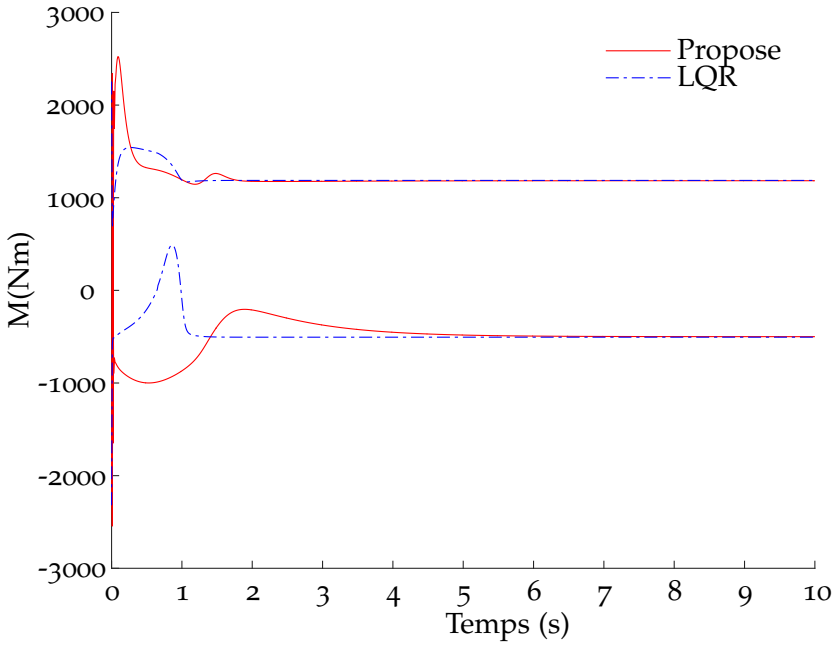


FIGURE 4.26 – *Drift, Simulation 2 : Moments appliqués aux roues*

4.3 Essais sur le véhicule réel

Afin de valider expérimentalement l'étude de la commande d'un véhicule, des essais ont été réalisés au laboratoire Heudiasyc sur une voiture électrique Zoé. Elle est équipée de capteurs pour mesurer ses états : l'angle volant, l'accélération, le couple de freinage, la vitesse angulaire, la position et l'orientation. Pour la position, on utilise un système

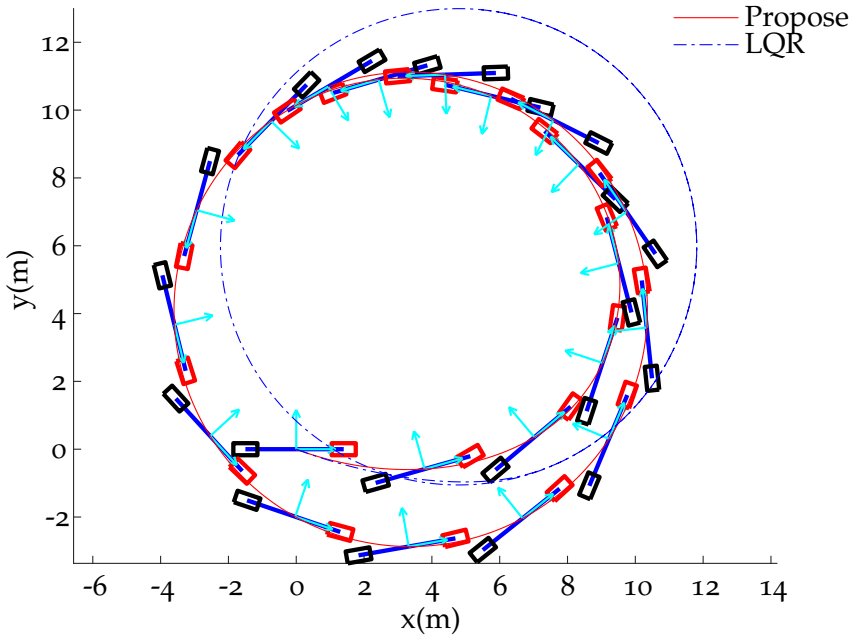


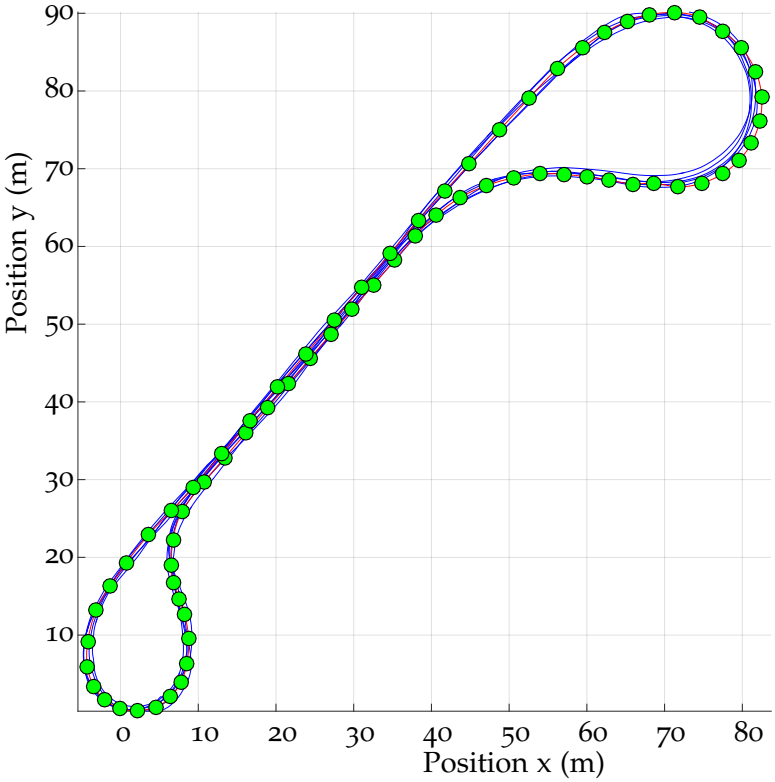
FIGURE 4.27 – *Drift, Simulation 2 : Trajectoire d'un bicyclette en drift*

GPS RTK, qui permet avec deux antennes de mesurer l'angle de lacet. Les couples sont appliqués aux roues avant (FWD). Il faut modifier le modèle du véhicule pour générer la table de prédiction et vérifier à nouveau en simulation avant d'effectuer les tests réels.

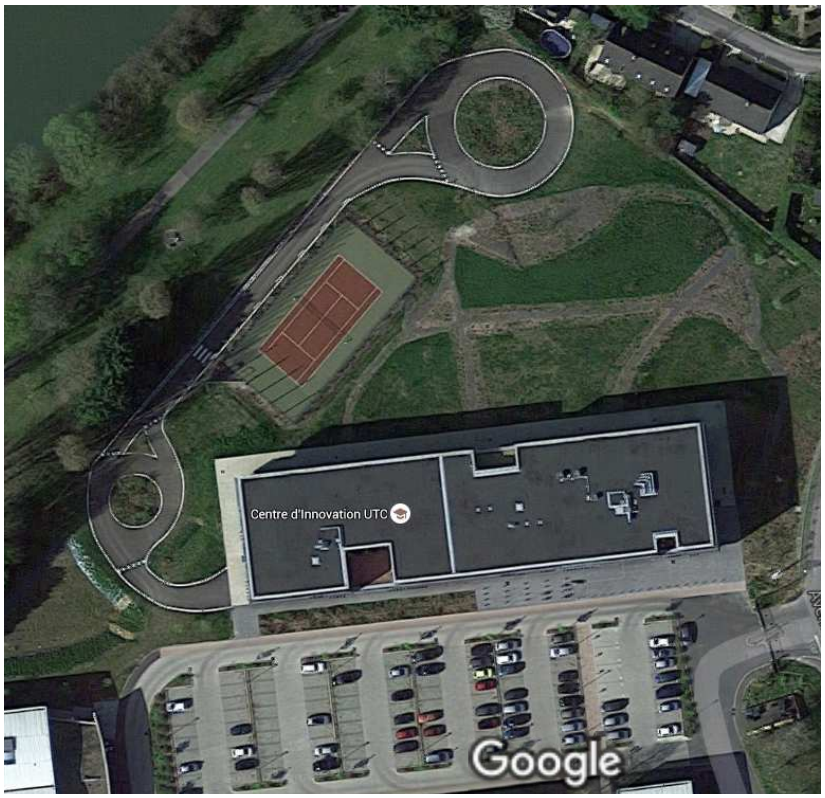
4.3.1 Planification de la trajectoire et localisation

Il y a des limites dans la performance de la voiture, il faut donc générer une trajectoire qui peut être suivie. La courbure de la trajectoire doit être continue et la plus lisse possible. On utilise souvent des courbes dites Clothoide. C'est une courbe dont la courbure est proportionnelle à l'abscisse curviligne. La planification de trajectoire et la localisation du véhicule présentent des problèmes intéressants et complexes, mais dans ces essais, la trajectoire est construite a priori. A partir des données mesurées avec le capteur GPS (relevé en conduite manuelle) et une carte numérique, (Google Map, OpenStreetMap par exemple), les segments de la trajectoire sont choisis. Puis, chaque segment est approximé par une courbe Clothoide. Finalement, la trajectoire est sauvegardée. La figure 4.28 montre les points de la trajectoire choisie sur le circuit d'essai du laboratoire.

Pour déterminer la position du véhicule par rapport à la carte et à la trajectoire, la position du véhicule est projetée sur les segments au voisinage de la position P_i pour calculer la distance entre le véhicule et chaque segment. La projection est le point H_i lié à la distance minimale (figure 4.29). Une fois cette projection effectuée, les deux variables d et θ_e sont déterminés.



(a) *La trajectoire Séville*



(b) *La trajectoire Séville réelle en vue satellite (Google Map)*

FIGURE 4.28 – *La trajectoire Séville*

4.3.2 Estimation d'état

Pour contrôler le système avec la commande basée sur l'état, il faut estimer son état à partir des capteurs. Dans ce cas, on souhaite suivre une trajectoire dans l'espace 2D, donc, il faut estimer la pose du véhicule (la position et l'angle de lacet). Plusieurs travaux concernant le problème d'estimation d'état du véhicule peuvent être trouvés dans la littérature, par exemple Doumiati et al. (2012). Cette partie porte sur une méthode simplifiée pour estimer la pose du véhicule basée sur l'équation cinématique et un filtre de Kalman étendu. Les données du GPS-RTK contiennent la position et l'angle de lacet en utilisant deux antennes. On va estimer la position du centre de gravité.

La figure 4.30 présente le schéma cinématique d'un modèle bicyclette sans drift. On suppose que l'antenne du GPS est attachée sur le point A. Ce point est lié au véhicule par rapport au centre de gravité comme l'indique l'équation (4.20).

$$\begin{cases} x_A &= x_G - Lr\cos(\theta) \\ y_A &= y_G - Lr\sin(\theta) \end{cases} \tag{4.20}$$

Les équations cinématiques du système sont données par :

$$\begin{cases} \dot{x}_A &= V_f\cos(\delta)\cos(\theta) \\ \dot{y}_A &= V_f\cos(\delta)\sin(\theta) \\ \dot{\theta} &= \dfrac{V_f}{L_f+L_r}\sin(\delta) \end{cases} \tag{4.21}$$

En combinaison avec l'équation (4.20), on obtient :

$$\begin{cases} \dot{x}_G &= V_f\cos(\delta)\cos(\theta) - V_f\dfrac{Lr}{L_f+L_r}\sin(\delta)\sin(\theta) \\ \dot{y}_G &= V_f\cos(\delta)\sin(\theta) + V_f\dfrac{Lr}{L_f+L_r}\sin(\delta)\cos(\theta) \\ \dot{\theta} &= \dfrac{V_f}{L_f+L_r}\sin(\delta) \end{cases} \tag{4.22}$$

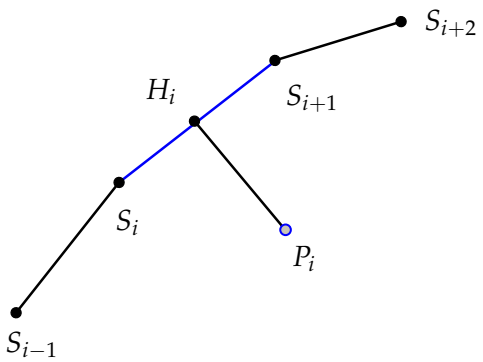


FIGURE 4.29 – *Localisation Simple*

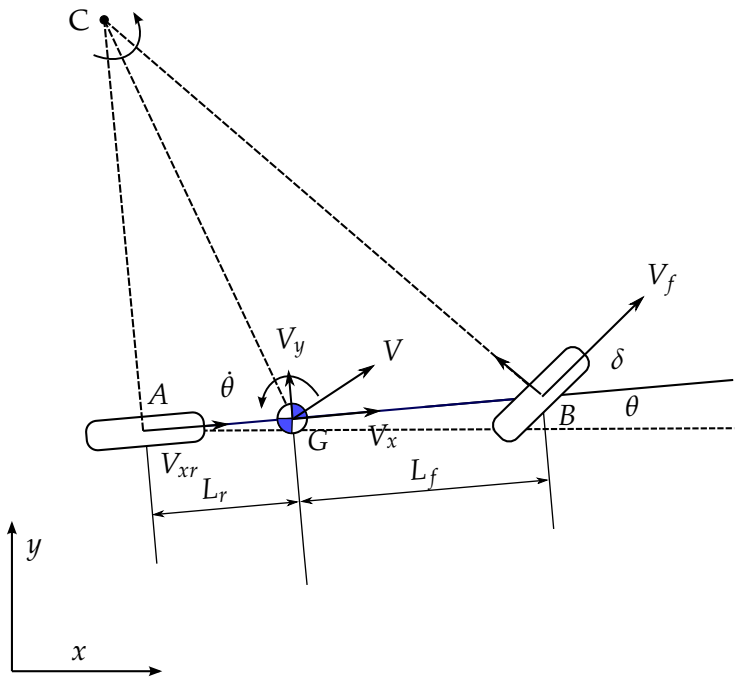


FIGURE 4.30 – *Schéma cinématique*

Le vecteur d'état est $\mathbf{x} = [x_G, y_G, \theta]^T$, la sortie est $\mathbf{y} = [x_A, y_A, \theta]^T$ et l'entrée est $\mathbf{u} = [V_f, \delta]$. L'équation (4.22) est discrétisée en utilisant l'approximation d'Euler à l'ordre un afin de mettre en forme un filtre de Kalman étendu.

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + T_s F(\mathbf{x}_k, \mathbf{u}_k)$$

Une fois les états estimés obtenus, la vitesse latérale sera approximée par $V_y \approx L_r \dot{\hat{\theta}}$. On note que la relation entre l'angle de volant en (°) et l'angle de braquage en (rad) est $\delta_{volant} = 15.625 \times \delta \dfrac{180}{pi}$ pour la Zoé. Cette relation permet de calculer δ à partir de l'angle de volant. Elle peut être trouvée en utilisant

$$\dot{\theta} = \frac{V_f}{L_f + L_r} \sin(\delta) = \frac{V_f}{L_f + L_r} \sin(\delta_{volant}/k_v)$$

d'où

$$k_v = \frac{\delta_{volant}}{\arcsin((L_r + L_f)\dot{\theta}/V_f)}$$

L'angle volant, la vitesse angulaire, la vitesse moyenne des roues avant sont disponibles grâce au bus CAN. La figure 4.31 montre l'angle volant divisé par $k_v = 15.625$ (la courbe "calcul") et l'angle de braquage calculé par $\delta = \arcsin((L_r + L_f)\dot{\theta}/V_f$ (la courbe "from wheel"). Ces données sont enregistrés lors d'un essai réel sur la ZOÉ. Le coefficient k_v a été vérifié par des données expérimentales.

Les figures 4.32, 4.33 et 4.34 présentent le résultat de l'estimation. Ces données sont collectées à partir d'un essai en conduite manuelle. La position et l'angle de lacet obtenus par GPS sont à $10Hz$ mais la vitesse angulaire et l'angle de volant sont à $100Hz$. L'estimation a été réalisée à $100Hz$ pour la commande de la Zoé. La figure 4.34 montre la trajectoire estimée au centre de gravité avec $L_r = 1.4$.

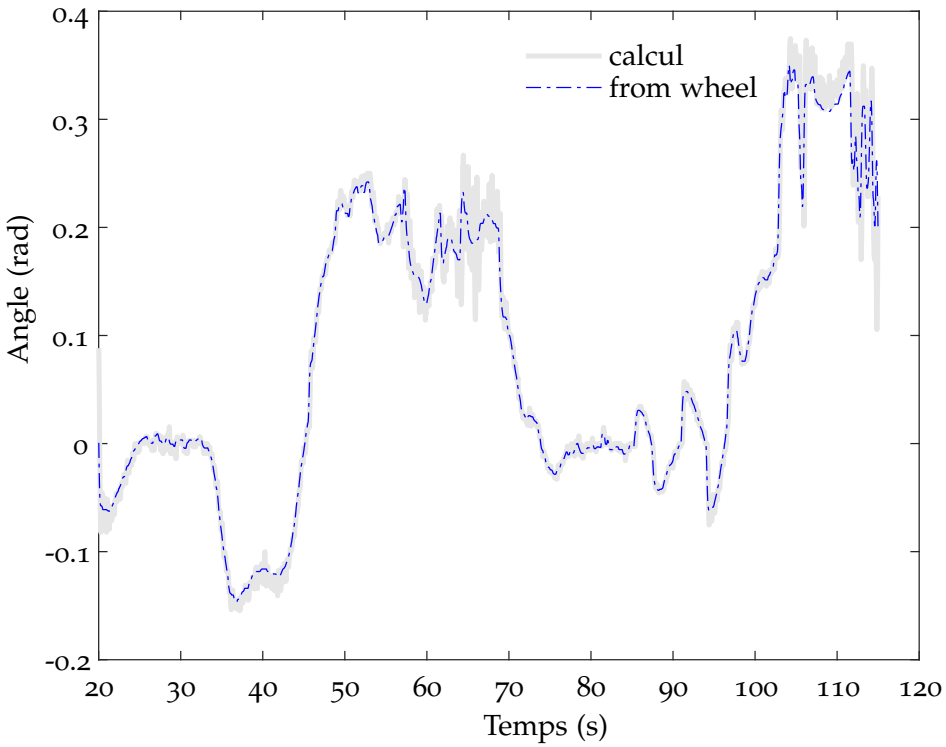


FIGURE 4.31 – *L'angle au volant divisé par k_v et l'angle de braquage calculé*

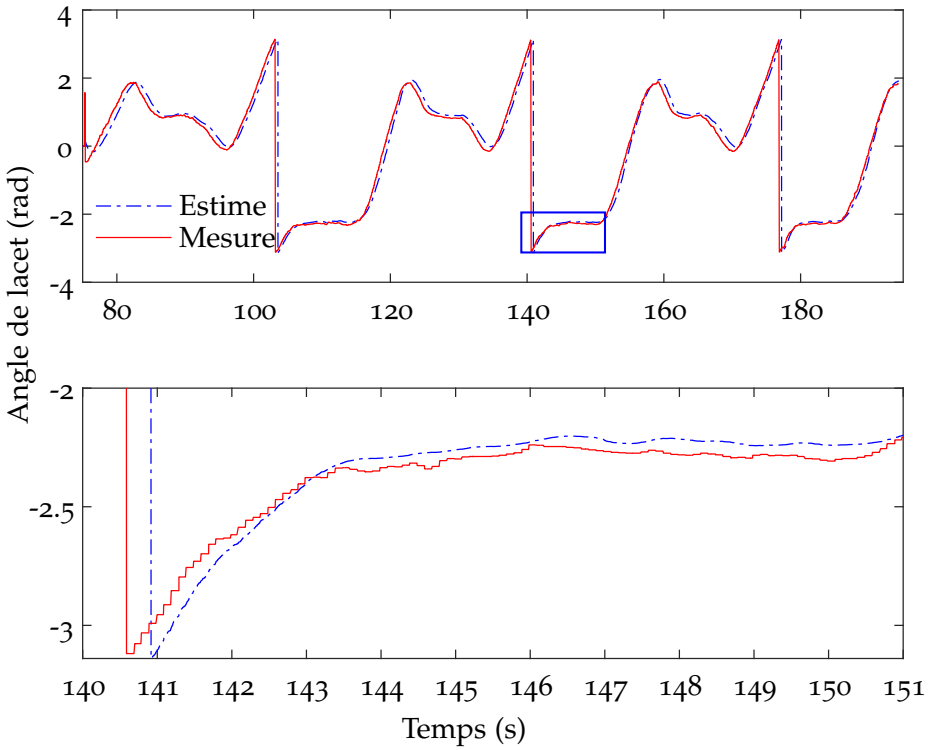


FIGURE 4.32 – *L'angle de lacet θ et sa valeur estimée*

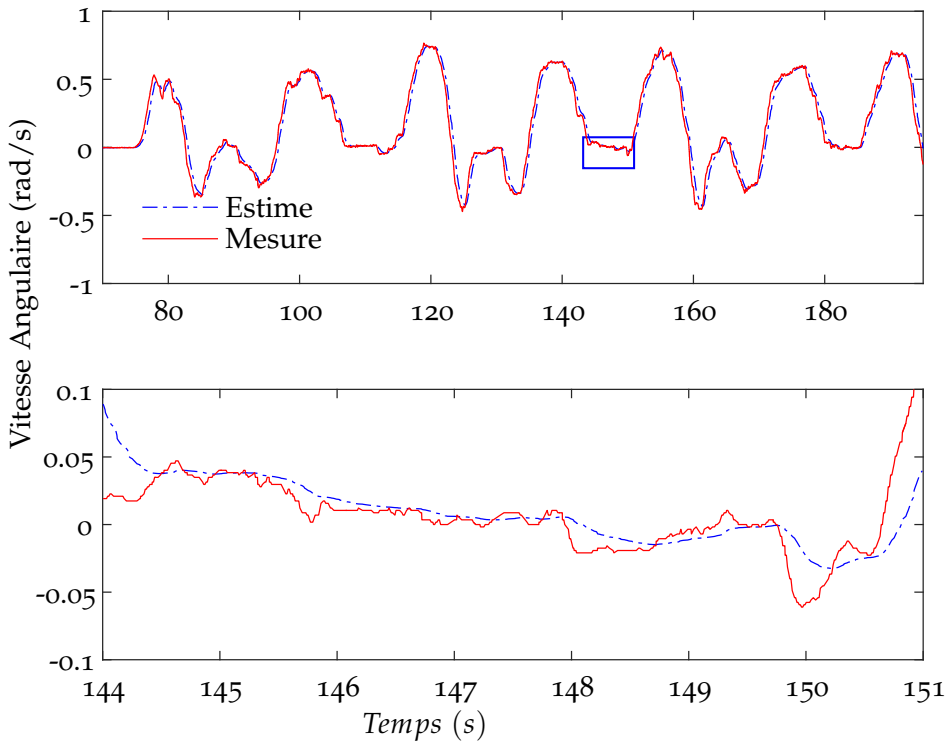


FIGURE 4.33 – *La vitesse angulaire et sa valeur estimée*

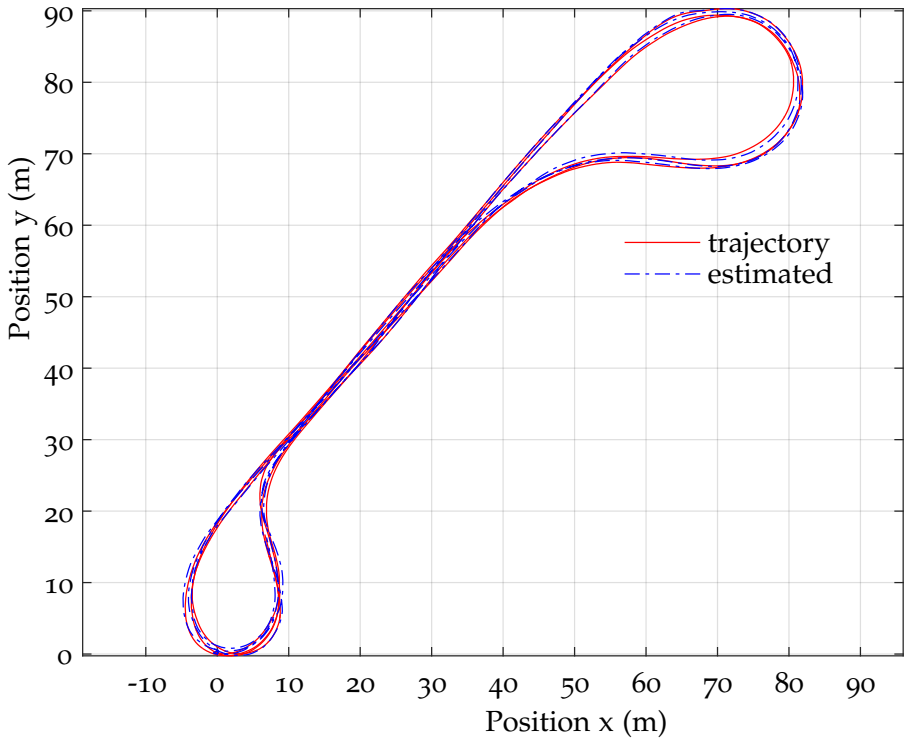


FIGURE 4.34 – *La trajectoire et son estimée*

4.3.3 Résultat et commentaire

Afin de réaliser des tests, on a fait l'expérimentation sur le circuit Séville du laboratoire, le modèle bicyclette de type FWD[5] et les paramètres de la Zoé. La figure 4.35 montre la trajectoire suivi par la Zoé (la courbe bleue) et sa référence (la courbe rouge). Dans ce test, on souhaite stabiliser la vitesse longitudinale à 10 km/h.

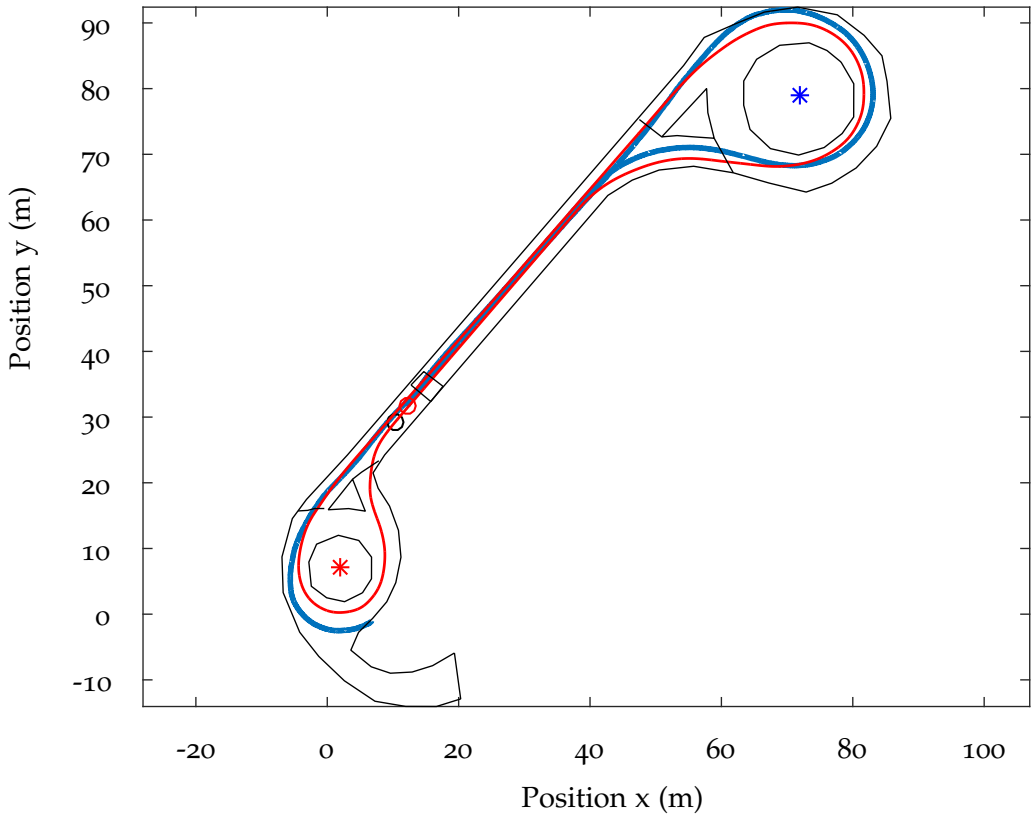


FIGURE 4.35 – *Essai réel : Suivi de la trajectoire Séville*

L'angle au volant est affiché sur la figure 4.36, la figure 4.37 montre la vitesse longitudinale en km/h et l'accélération en Nm. On constate que l'erreur sur la vitesse longitudinale est parfois importante, de même pour l'erreur latérale. Pour la sécurité, le véhicule a parfois été freiné par le pilote ce qui provoque des variation de la force verticale au cours du parcours de la trajectoire Séville. L'estimation d'état est approximative en utilisant le modèle cinématique. Les paramètres pour générer la table de prédiction utilisée pour la commande sont trop différents des valeurs réelles. Tous ces éléments offrent des perspectives pour améliorer la performance sur des prochains essais.

5. Front Wheel Drive

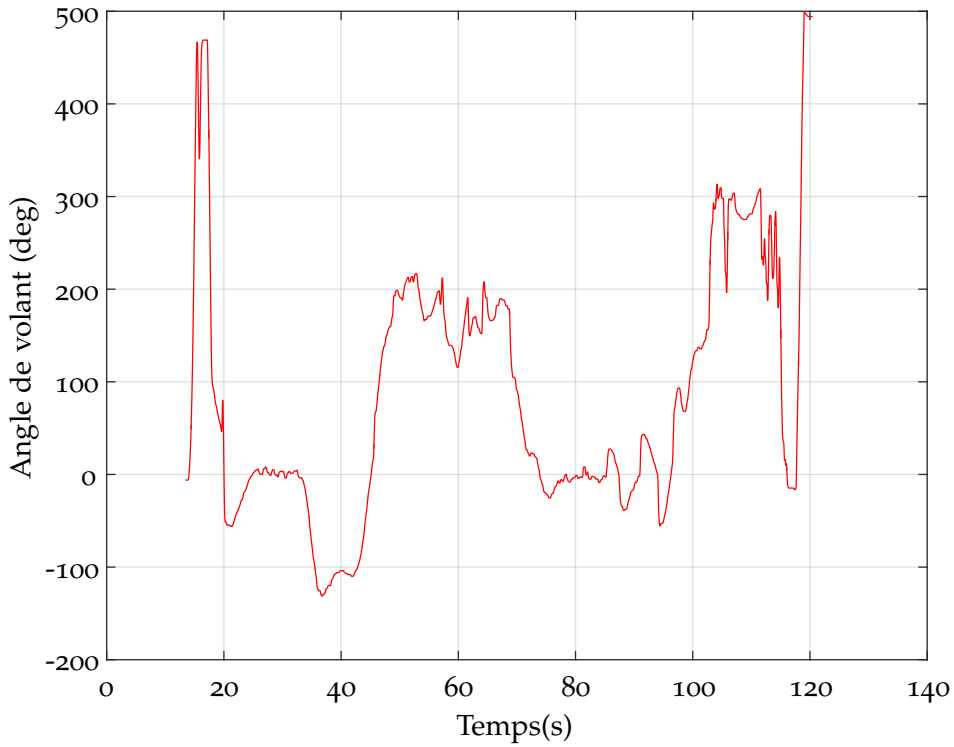


FIGURE 4.36 – *Essai réel : L'angle volant*

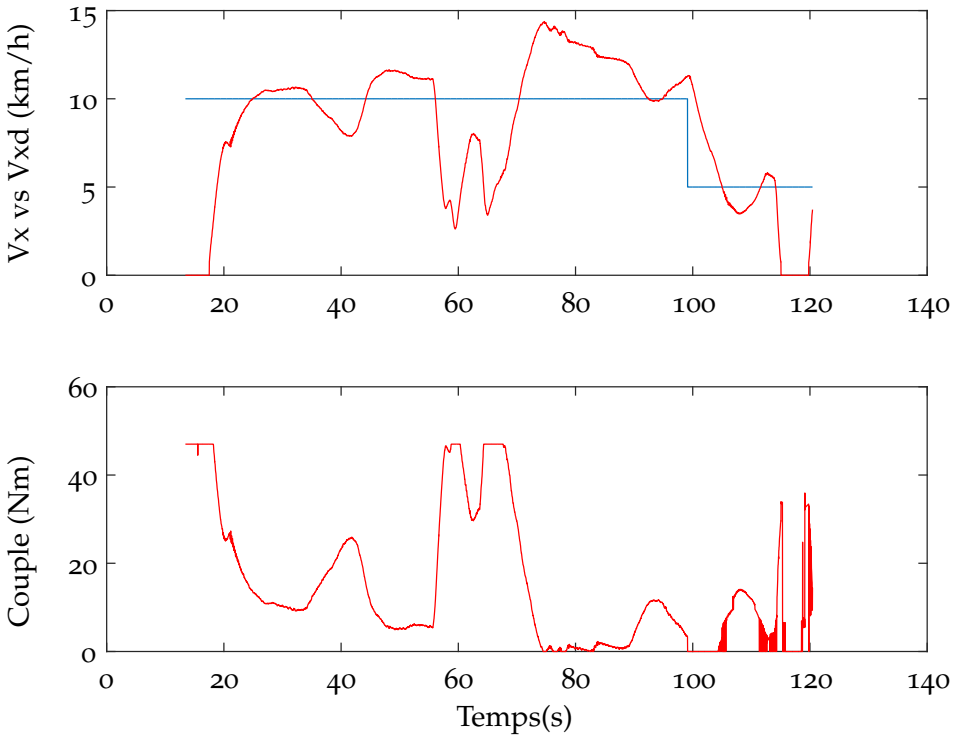


FIGURE 4.37 – *Essai réel : La vitesse longitudinale et le couple aux roues*

Conclusion du chapitre

La commande appliquée dans ce chapitre a été construite en utilisant le modèle d'état du système simplifié dit bicyclette. On a étudié la dynamique du véhicule et modélisé le système simplifié pour générer la table de prédiction. La commande a été utilisée pour conduire une voiture autonome suivant une trajectoire et également utilisée en simulation pour une voiture en drift. Dans ce cas, la commande peut trouver automatiquement le point d'équilibre du système. En terme d'amélioration, on envisage d'appliquer l'algorithme d'apprentissage pour adapter la table de comportement du véhicule réel. L'estimation d'état du véhicule ainsi que sa localisation ont été obtenues avec des modèles simples ne tenant pas compte de plusieurs phénomènes et variables (par exemple, la dynamique verticale). L'amélioration de résultats nécessite une meilleure précision du modèle utilisé pour la table de comportement.

Chapitre 5

Application à la machine asynchrone

Dans ce chapitre, nous présentons deux méthodes pour l'observation de la vitesse d'une machine asynchrone basée sur la structure de commande détaillée dans ce document. Après avoir développé le modèle mathématique de la machine asynchrone, nous construisons le problème d'observation en considérant qu'il est équivalent à un problème de commande. Finalement, des résultats de simulation sont présentés.

Sommaire

5.1 Modélisation mathématique

La machine asynchrone ou machine à induction, est une machine électrique à courant alternatif dont le fonctionnement est basé sur le couple électromagnétique généré par un champ magnétique. Ce champ tournant induit le courant électrique dans les enroulements du rotor. Les courants alternatifs circulant dans les enroulements au stator génèrent ce champ magnétique tournant. La fréquence de rotation de ce champ est imposée par la fréquence des courants statoriques. Sa vitesse de rotation est appelée vitesse de synchronisme. Une force électromotrice induite apparaît et crée des courants rotoriques qui mettent le rotor en mouvement afin de s'opposer à la variation de flux d'après la loi de Lenz-Faraday. Si le rotor tourne à la même vitesse que le champ magnétique du stator, il n'y a pas de variation de champ magnétique et donc le couple électromagnétique est nul. La machine est donc appelée asynchrone en ce sens qu'il existe une différence entre la vitesse du rotor et la vitesse de rotation du champ statorique.

Le stator de la machine asynchrone est souvent constitué par trois bobines alimentées par un système de tensions triphasées placées à 120°. Le rotor peut être bobiné ou à cage d'écureuil pour générer les courants induits. La figure 5.1 montre le schéma d'une machine asynchrone triphasée avec rotor bobiné. Dans cette partie on se focalise sur ce type de machine [1] mais le principe de la méthode peut-être utilisé pour d'autres machines électriques. La modélisation mathématique de la machine asynchrone se divise en trois étapes : mise en équation électrique, mise en équation magnétique et établissement des équations mécaniques (Séguier et Notelet (2006)). Les équations sont d'abord écrites pour le modèle triphasé puis dérivées pour un modèle biphasé équivalent afin de réduire le nombre des variables.

5.1.1 Modèle triphasé

Équations électriques

D'après la loi des mailles d'un circuit électrique et la loi de Faraday pour la force électromotrice, la relation entre les tensions, les courants et les flux statoriques peut s'écrire sous la forme suivante :

$$\begin{bmatrix} V_{sa} \\ V_{sb} \\ V_{sc} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix} \begin{bmatrix} i_{sa} \\ i_{sb} \\ i_{sc} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \Phi_{sa} \\ \Phi_{sb} \\ \Phi_{sc} \end{bmatrix}$$

réécrite sous forme matricielle comme suit :

$$\mathbf{V}_s = R_s \mathbf{i}_s + \frac{d}{dt} \mathbf{\Phi}_s \tag{5.1}$$

1. Il s'agit d'une étude demandée et financée en partie par le CETIM (Centre d'Etude Technique des Industries Mécaniques)

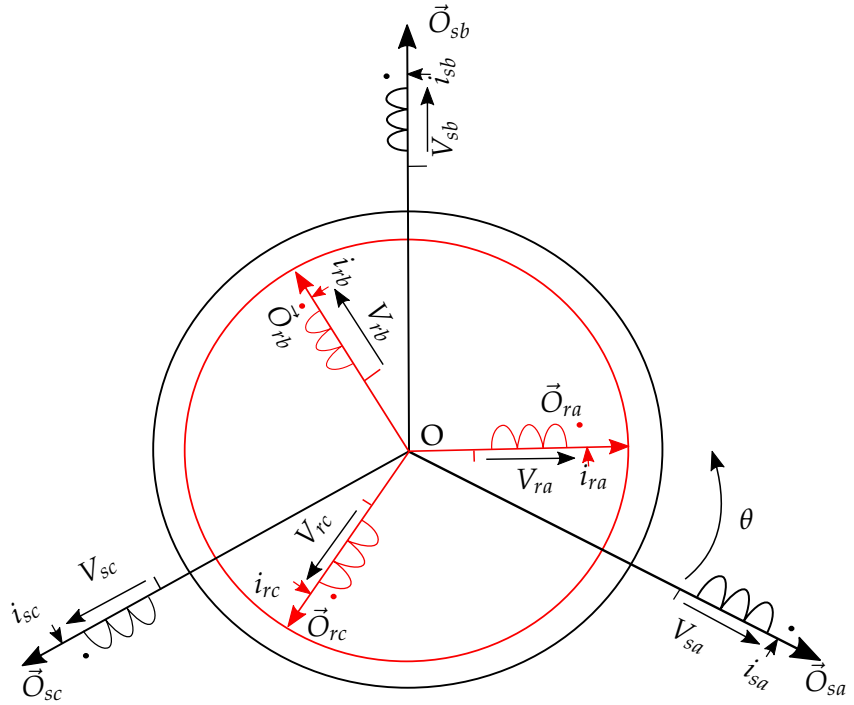


FIGURE 5.1 – *Modèle d'une machine asynchrone triphasé*

Pour le rotor on a :

$$\mathbf{V}_r = R_r \mathbf{i}_r + \frac{d}{dt}\mathbf{\Phi}_r = \mathbf{0} \tag{5.2}$$

Équations magnétiques

Pour chaque phase, le flux total est égal au flux généré par le courant et le flux mutuel. La relation entre le flux et le courant qui génère ce flux est supposé linéaire, d'où les équations magnétiques suivantes :

$$\begin{bmatrix} \Phi_{sa} \\ \Phi_{sb} \\ \Phi_{sc} \\ - \\ \Phi_{ra} \\ \Phi_{rb} \\ \Phi_{rc} \end{bmatrix} = \left[\begin{array}{ccc|ccc} l_s & M_s & M_s & M_1 & M_3 & M_2 \\ M_s & l_s & M_s & M_2 & M_1 & M_3 \\ M_s & M_s & l_s & M_3 & M_2 & M_1 \\ - & - & - & - & - & - \\ M_1 & M_2 & M_3 & l_r & M_r & M_r \\ M_3 & M_1 & M_2 & M_r & l_r & M_r \\ M_2 & M_3 & M_1 & M_r & M_r & l_r \end{array}\right] \begin{bmatrix} i_{sa} \\ i_{sb} \\ i_{sc} \\ - \\ i_{ra} \\ i_{rb} \\ i_{rb} \end{bmatrix}$$

avec la définition suivante des variables :

l_s	l'inductance propre d'une phase statorique
l_r	l'inductance propre d'une phase rotorique
M_s	l'inductance mutuelle entre deux phases statoriques
M_r	l'inductance mutuelle entre deux phases rotoriques
M_{sr}	l'inductance mutuelle maximale entre une phase statorique et une phase rotorique

$$\begin{aligned} M_1 &= M_{sr}cos\left(\theta\right) \\ M_2 &= M_{sr}cos\left(\theta - \tfrac{4\pi}{3}\right) \\ M_3 &= M_{sr}cos\left(\theta - \tfrac{2\pi}{3}\right) \end{aligned}$$

Ces relations s'expriment sous une forme matricielle par :

$$\mathbf{\Phi}_{s(abc)} = \mathbf{L}_{s(abc)}\mathbf{i}_{s(abc)} + \mathbf{M}_{sr}\mathbf{i}_{r(abc)} \tag{5.3}$$

$$\mathbf{\Phi}_{r(abc)} = \mathbf{M}_{rs}\mathbf{i}_{s(abc)} + \mathbf{L}_{r(abc)}\mathbf{i}_{r(abc)} \tag{5.4}$$

avec

$$\mathbf{M}_{sr} = \mathbf{M}_{rs}^{t} = M_{sr}\begin{bmatrix} \cos\left(\theta\right) & \cos\left(\theta + \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) \\ \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ \cos\left(\theta + \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta\right) \end{bmatrix}$$

Couple électromagnétique

$$C_e = n_p\left(\frac{\partial \mathbf{W}_{co}}{\partial\theta}\right)[i] = n_p\left[i_s\right]^t\frac{\partial}{\partial\theta}\mathbf{M}_{sr}\left(\theta\right)\left[i_r\right] \tag{5.5}$$

Équations mécaniques

$$J\frac{d}{dt}\Omega_m = C_e - C_r - f\Omega_m \tag{5.6}$$

$$\Omega_m = \left(1 - g\right)\frac{\omega_s}{p} \tag{5.7}$$

$$g = \frac{\omega_s - \omega}{\omega_s} = \frac{\omega_r}{\omega_s} \tag{5.8}$$

J	le moment d'inertie des masses tournantes
g	le glissement
f	le coefficient de frottement visqueux
C_r	le couple résistant imposé par l'arbre de la machine
C_e	le couple électromagnétique
Ω_m	la vitesse angulaire mécanique
$f\Omega_m$	le couple de frottement visqueux
ω_s	la vitesse angulaire du champ statorique

5.1.2 Modèle biphasé

Transformation triphasé-biphasé

L'objectif de cette transformation est de passer d'un système triphasé (abc) vers un système diphasé $(\alpha\beta)$ en alignant l'axe (α) avec l'axe (a) comme présenté sur la figure 5.2. Il existe principalement deux transformations : Clarke qui conserve l'amplitude des grandeurs mais pas la puissance et le couple, et la transformation de Concordia qui conserve la puissance mais pas les amplitudes.

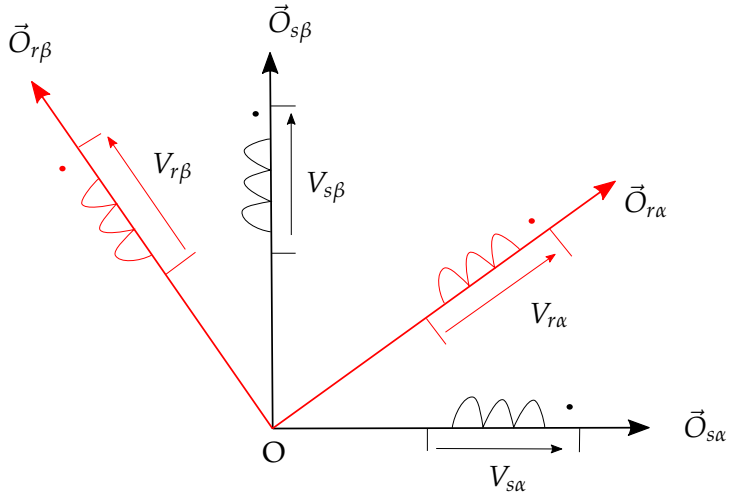


FIGURE 5.2 – *Transformation triphasé-biphasé*

Transformation de Concordia

Le passage du repère abc au repère $\alpha\beta$ se fait en utilisant la matrice de passage $\mathbf{T}_{23}$ représentée comme suit :

$$\mathbf{x}_{\alpha\beta} = \mathbf{T}_{23}\mathbf{x}_{abc} \tag{5.9}$$

avec $\mathbf{T}_{23} = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$

De la même façon, le passage du repère $\alpha\beta$ au repère abc se fait en utilisant la matrice de passage $\mathbf{T}_{32}$

$$\mathbf{x}_{abc} = \mathbf{T}_{32}\mathbf{x}_{\alpha\beta} \tag{5.10}$$

avec $\mathbf{T}_{32} = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$

Transformation de Clarke

Pour la transformation de Clarke, la matrice de passage est $\mathbf{C}_{23}$:

$$\mathbf{x}_{\alpha\beta} = \mathbf{C}_{23}\mathbf{x}_{abc} \tag{5.11}$$

avec $\mathbf{C}_{23}^{T} = \mathbf{C}_{32} = \frac{2}{3}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$

Transformation de Park

La transformation de Park permet de passer du repère fixe (abc) au repère tournant à

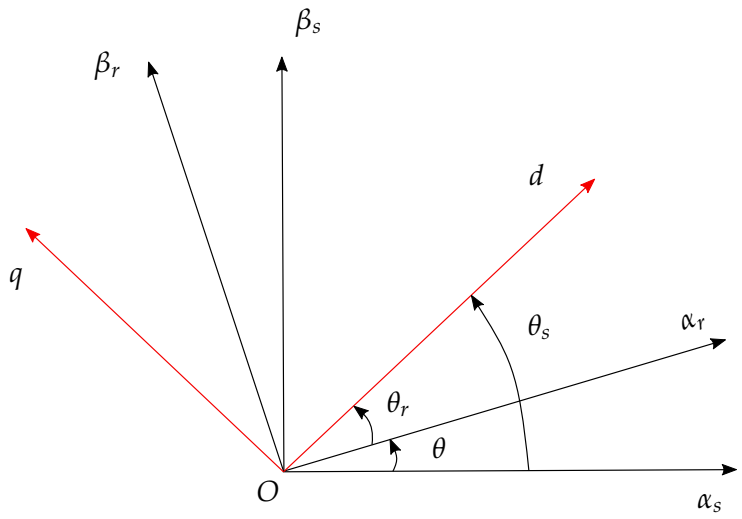


FIGURE 5.3 – *Transformation de Park*

vitesse quelconque $\omega_s = \frac{d\theta_s}{dt}$. La relation est :

$$\mathbf{x}_{abc} = \mathbf{P}_{32}(\theta_s)\mathbf{x}_{dq} \tag{5.12}$$

avec la matrice de passage

$$\mathbf{P}_{32}(\theta_s) = \sqrt{\frac{2}{3}}\begin{bmatrix} \cos(\theta_s) & -\sin(\theta_s) \\ \cos(\theta_s - \frac{2\pi}{3}) & -\sin(\theta_s - \frac{2\pi}{3}) \\ \cos(\theta_s + \frac{2\pi}{3}) & -\sin(\theta_s + \frac{2\pi}{3}) \end{bmatrix} = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} \cos(\theta_s) & -\sin(\theta_s) \\ \sin(\theta_s) & \cos(\theta_s) \end{bmatrix}$$

$$\mathbf{P}_{32}(\theta_s) = \mathbf{T}_{32}\mathbf{P}(\theta_s) \tag{5.13}$$

où la matrice de rotation dans l'espace 2D est définie par :

$$\mathbf{P}(\theta_s) = \begin{bmatrix} \cos(\theta_s) & -\sin(\theta_s) \\ \sin(\theta_s) & \cos(\theta_s) \end{bmatrix}$$

5.1.3 Mise en équation d'état

Dans le repère fixé au stator $\alpha\beta$, pour le stator,

$$\mathbf{T}_{23}\mathbf{\Phi}_{s(abc)} = \mathbf{T}_{23}\left[\mathbf{L}_{s(abc)}\mathbf{i}_{s(abc)} + \mathbf{M}_{sr}\mathbf{i}_{r(abc)}\right]$$

En utilisant $\mathbf{\Phi}_{s\alpha\beta} = \mathbf{T}_{23}\mathbf{\Phi}_{s(abc)}$,$\mathbf{i}_{s(abc)} = \mathbf{T}_{32}\mathbf{i}_{s\alpha\beta}$ et $\mathbf{i}_{r(abc)} = \mathbf{T}_{32}\mathbf{P}(-\theta)\mathbf{i}_{r\alpha\beta}$ on a

$$\mathbf{\Phi}_{s\alpha\beta} = \mathbf{T}_{23}\mathbf{L}_{s(abc)}\mathbf{T}_{32}\mathbf{i}_{s\alpha\beta} + \mathbf{T}_{23}\mathbf{M}_{sr}\mathbf{T}_{32}\mathbf{P}(-\theta)\mathbf{i}_{r\alpha\beta}$$

d'où

$$\mathbf{\Phi}_{s\alpha\beta} = L_s\mathbf{i}_{s\alpha\beta} + M\mathbf{i}_{r\alpha\beta} \tag{5.14}$$

où $L_s = l_s - M_s, M = L_m = \frac{3}{2}M_{sr}$ Pour le rotor, on a :

$$\mathbf{P}(\theta)\mathbf{T}_{23}\mathbf{\Phi}_{r(abc)} = \mathbf{P}(\theta)\mathbf{T}_{23}\left[\mathbf{L}_{r(abc)}\mathbf{i}_{r(abc)} + \mathbf{M}_{sr}^T\mathbf{i}_{s(abc)}\right]$$

Cela donne le flux :

$$\mathbf{\Phi}_{r\alpha\beta} = \mathbf{P}(\theta)\mathbf{T}_{23}\mathbf{L}_{r(abc)}\mathbf{T}_{32}\mathbf{P}(-\theta)\mathbf{i}_{r(\alpha\beta)} + \mathbf{P}(\theta)\mathbf{T}_{23}\mathbf{M}_{sr}^{T}\mathbf{T}_{32}\mathbf{i}_{s\alpha\beta}$$

L'équation magnétique du rotor s'écrit sous la forme :

$$\mathbf{\Phi}_{r\alpha\beta} = L_r\mathbf{i}_{r\alpha\beta} + M\mathbf{i}_{s\alpha\beta} \tag{5.15}$$

où $L_r = l_r - M_r, M = L_m = \frac{3}{2}M_{sr}$
Les équations 5.1 de la machine asynchrone avec la transformée de Concordia, et la transformée de Park s'écrivent donc sous la forme suivante :

$$\mathbf{T}_{23}\mathbf{V}_{s(abc)} = \mathbf{T}_{23}\left[R_s\mathbf{i}_{s(abc)} + \frac{d}{dt}\mathbf{\Phi}_{s(abc)}\right]$$

On obtient ainsi les équations au stator :

$$\mathbf{V}_{s\alpha\beta} = R_s\mathbf{i}_{s\alpha\beta} + \frac{d}{dt}\mathbf{\Phi}_{s\alpha\beta} \tag{5.16}$$

et au rotor :

$$\mathbf{V}_{r(abc)} = R_r\mathbf{i}_{r(abc)} + \frac{d}{dt}\mathbf{\Phi}_{r(abc)}$$

La dérivée du flux rotorique est calculée par :

$$\frac{d}{dt}\mathbf{\Phi}_{r(abc)} = \frac{d}{dt}\left[\mathbf{T}_{32}\mathbf{P}(-\theta)\mathbf{\Phi}_{r\alpha\beta}\right] = \mathbf{T}_{32}\mathbf{P}(-\theta)\frac{d}{dt}\mathbf{\Phi}_{r\alpha\beta} + \omega\mathbf{T}_{32}\frac{\partial}{\partial\theta}\mathbf{P}(-\theta)\mathbf{\Phi}_{r\alpha\beta}$$

La tension rotorique s'écrit alors comme :

$$\mathbf{V}_{r(abc)} = R_r\mathbf{i}_{r(abc)} + \mathbf{T}_{32}\mathbf{P}(-\theta)\frac{d}{dt}\mathbf{\Phi}_{r\alpha\beta} + \omega\mathbf{T}_{32}\frac{\partial}{\partial\theta}\mathbf{P}(-\theta)\mathbf{\Phi}_{r\alpha\beta}$$

En multipliant à gauche par $\mathbf{P}(\theta)\mathbf{T}_{23}$, cela donne :

$$\mathbf{P}(\theta)\mathbf{T}_{23}\mathbf{V}_{r(abc)} = \mathbf{P}(\theta)\mathbf{T}_{23}\left[R_r\mathbf{i}_{r(abc)} + \mathbf{T}_{32}\mathbf{P}(-\theta)\frac{d}{dt}\mathbf{\Phi}_{r\alpha\beta} + \omega\mathbf{T}_{32}\frac{\partial}{\partial\theta}\mathbf{P}(-\theta)\mathbf{\Phi}_{r\alpha\beta}\right]$$

Si on considère que la tension appliquée au rotor est nulle :

$$\mathbf{0} = \mathbf{V}_{r\alpha\beta} = R_r\mathbf{i}_{r\alpha\beta} + \frac{d}{dt}\mathbf{\Phi}_{r\alpha\beta} + \omega\mathbf{P}(\theta)\mathbf{T}_{23}\mathbf{T}_{32}\frac{\partial}{\partial\theta}\mathbf{P}(-\theta)\mathbf{\Phi}_{r\alpha\beta}$$

On obtient :

$$\mathbf{0} = R_r\mathbf{i}_{r\alpha\beta} + \frac{d}{dt}\mathbf{\Phi}_{r\alpha\beta} + \omega\mathbf{P}(-\pi/2)\mathbf{\Phi}_{r\alpha\beta} \tag{5.17}$$

Le système de trois équations est réduit à un système à deux équations. C'est identique pour l'écriture des flux en fonction des courants. L'objectif pour les flux est de réduire les matrices 3×3 des inductances à des matrice 2×2, ce qui fait disparaître les inductances cycliques :

$$L_s = l_s - M_s, L_r = l_r - M_r, M = L_m = \frac{3}{2}M_{sr}$$

Couple électromagnétique

Le couple électromagnétique s'écrit :

$$C_e = n_p \mathbf{i}_{s(abc)}^T \frac{\partial}{\partial \theta} \mathbf{M}_{sr} \mathbf{i}_{r(abc)}$$

$$C_e = n_p \mathbf{i}_{s\alpha\beta}^T \mathbf{T}_{23} \frac{\partial}{\partial \theta} \mathbf{M}_{sr} \mathbf{T}_{32} \mathbf{P}(-\theta) \mathbf{i}_{r\alpha\beta}$$

Finalement, en combinant l'équation du couple C_e avec les équations qui lient les flux aux courants, on obtient un ensemble d'expressions possibles pour le couple :

$$C_e = n_p M \left(i_{s\beta} i_{r\alpha} - i_{s\alpha} i_{r\beta} \right) \tag{5.18}$$

$$C_e = n_p \left(\phi_{s\alpha} i_{s\beta} - \phi_{s\beta} i_{s\alpha} \right) \tag{5.19}$$

$$C_e = n_p \left(\phi_{r\beta} i_{r\alpha} - \phi_{r\alpha} i_{r\beta} \right) \tag{5.20}$$

$$C_e = n_p \frac{M}{L_s} \left(-\phi_{s\alpha} i_{r\beta} + \phi_{s\beta} i_{r\alpha} \right) \tag{5.21}$$

$$C_e = n_p \frac{M}{L_r} \left(-\phi_{r\beta} i_{s\alpha} + \phi_{r\alpha} i_{s\beta} \right) \tag{5.22}$$

Le modèle simplifié habituellement utilisé pour le modèle $\alpha\beta$ est obtenu à partir des équations (5.14), (5.15), (5.16), (5.17) et (5.22) :

$$\begin{bmatrix} \frac{di_{s\alpha}}{dt} \\ \frac{di_{s\beta}}{dt} \\ \frac{d\phi_{r\alpha}}{dt} \\ \frac{d\phi_{r\beta}}{dt} \\ \frac{d\Omega}{dt} \end{bmatrix} = \begin{bmatrix} \eta\mu\phi_{r\alpha} + \mu n_p \Omega \phi_{r\beta} - \gamma i_{s\alpha} + V_{s\alpha} / (\sigma L_s) \\ \eta\mu\phi_{r\beta} - \mu n_p \Omega \phi_{r\alpha} - \gamma i_{s\beta} + V_{s\beta} / (\sigma L_s) \\ -\eta\phi_{r\alpha} - n_p \Omega \phi_{r\beta} + M\eta i_{s\alpha} \\ -\eta\phi_{r\beta} + n_p \Omega \phi_{r\alpha} + M\eta i_{s\beta} \\ n_p \left(M/(JL_r) \right) \left(i_{s\beta}\phi_{r\alpha} - i_{s\alpha}\phi_{r\beta} \right) - Cr/J \end{bmatrix} \tag{5.23}$$

avec Rr, Lr, Rs, Ls les résistances et inductances rotorique et statorique, respectivement. Les courants statoriques exprimés dans le repère $\alpha\beta$ sont notés par $i_{s\alpha}$ et $i_{s\alpha}$. M est l'inductance mutuelle. $\phi_{r\alpha}$ et $\phi_{r\beta}$ sont les composants du flux rotorique. J est l'inertie du moteur. $\Omega = \omega / n_p$ est la vitesse mécanique qu'on veut observer. $V_{s\alpha}, V_{s\beta}$ sont les tensions du stator. Les autres paramètres du modèle sont :

$\eta = R_r / L_r = 1/\tau_r$

$\mu = M / (\sigma L_r L_s)$

$\gamma = M^2 R_r / \left(\sigma L_r^2 L_s \right) + R_s / (\sigma L_s)$

$\sigma = 1 - \frac{M^2}{L_s L_r}$

On note que toutes les équations sont indépendantes de la position θ du rotor.

5.2 Observation de la vitesse mécanique

5.2.1 État de l'art

Les méthodes pour observer la vitesse mécanique de la machine asynchrone ont un grand intérêt depuis quelques années pour développer des commandes performantes à

moindre coût en éliminant le capteur de vitesse coûteux, fragile et difficile à entretenir. On peut les classifier dans deux catégories : les méthodes directes et les méthodes indirectes. Les méthodes indirectes utilisent le modèle du système pour l'estimation. Entre autres techniques, on trouve celles utilisant un système adaptatif à modèle de référence (MRAS) Schauder (1992), Comanescu et Xu (2006). On trouve aussi des techniques utilisant le filtre de kalman étendu Pena et Asher (1993), Kim et al. (1994), Forgez (2005). L'intelligence artificielle (RNA) a aussi été proposée par Ben-Brahim et Kurosawa (1993), Kulkarni et El-Sharkawi (1997), Burton et al. (1997). Dans les méthodes directes on peut trouver des techniques basées sur l'injection soit d'un signal de tension soit d'un courant à haute fréquence Ha et Sul (1999),Ribeiro et al. (1998),Testa et al. (2005). Une autre technique exploite les composantes homo-polaires. Enfin une technique fonctionne en extrayant de l'information de la mesure de la variation du courant Schroedl (1996).

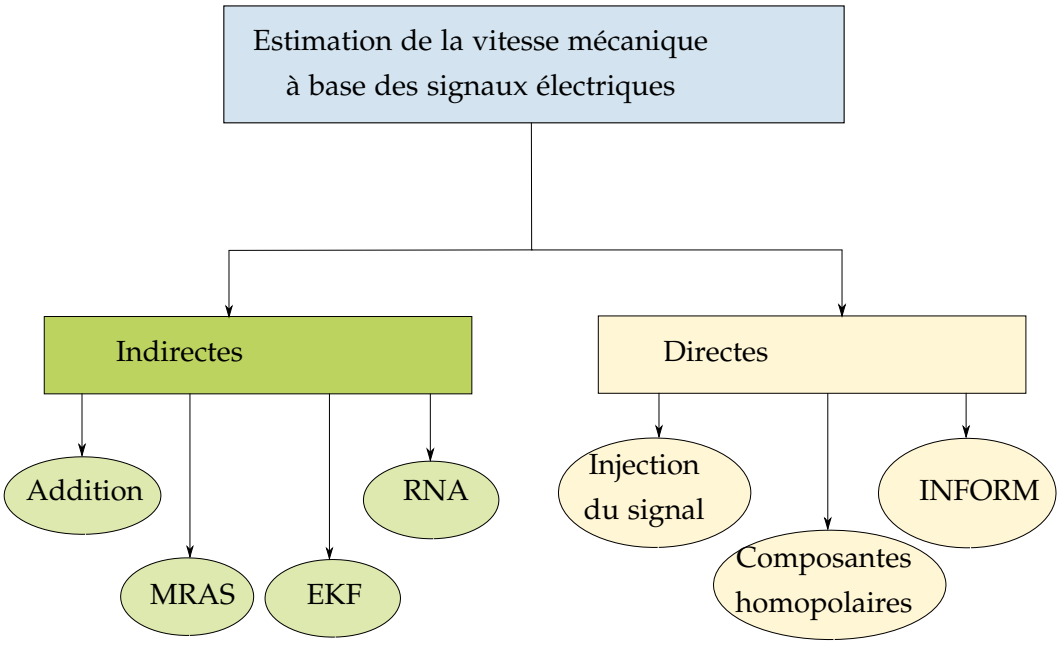


FIGURE 5.4 – *Catégorie des méthodes*

5.2.2 Algorithmes d'observation

Dans ce travail, l'observateur de la vitesse mécanique est construit en utilisant notre algorithme de commande. La vitesse mécanique est considérée comme l'entrée du système et la référence à atteindre correspond à la mesure à l'instant k. Les résultats sont comparés avec une méthode basée sur le filtre de Kalman étendu et une méthode basée sur le calcul des flux magnétiques. Ces méthodes seront décrites dans les paragraphes suivants.

Calcul des flux magnétiques

L'intégration de l'équation (5.16) permet d'estimer les flux en boucle ouverte mais donne des valeurs biaisées à cause de la condition initiale. Pour éviter ce problème, dans certains travaux issus de la littérature, des filtres passe-bas sont ajouté avec un correcteur PID, par exemple Hu et Wu (1998). Cette méthode est détaillée dans la figure 5.5, avec $\mathbf{v}_{emf} = \mathbf{V}_s - R_s\mathbf{I}_s$, et w_c la fréquence de coupure du filtre passe-bas. A partir des flux statoriques estimés, les flux rotoriques peuvent être calculés par :

$$\hat{\Phi}_r = \frac{L_r}{M}(\hat{\Phi}_s - \sigma L_s \mathbf{I}_s)$$

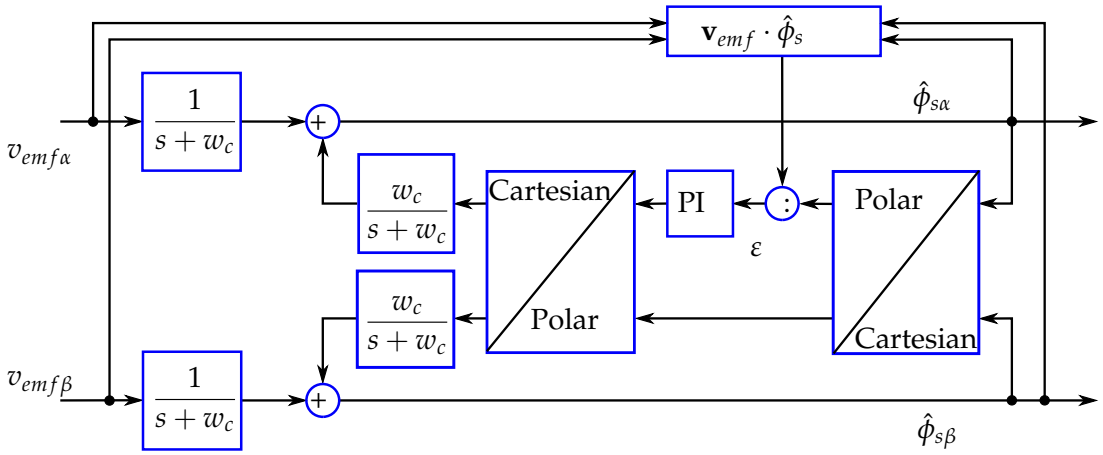


FIGURE 5.5 – *Calcul des flux utilisant filtre de passe base et correcteur PI*

A coté des méthodes pour estimer la vitesse mécanique, l'estimation des flux joue un rôle important pour la commande d'une machine asynchrone car c'est une composante essentielle pour le calcul du couple et le positionnement relatif des vecteurs en commande vectorielle. A partir des flux estimés, on peut estimer la vitesse mécanique. Schauder (1992) a proposé une méthode pour déterminer la vitesse mécanique, connaissant les flux rotoriques, de la façon suivante :

$$\frac{d\Phi_r}{dt} = \frac{L_r}{M}(\mathbf{V}_s - R_s\mathbf{I}_s - \sigma L_s \frac{d\mathbf{I}_s}{dt})$$

L'angle du vecteur des flux rotoriques est défini par $\theta_m = \arctan(\frac{\phi_{r\beta}}{\phi_{r\alpha}})$, donc :

$$\dot{\theta}_m = \frac{\dot{\phi}_{r\beta}\phi_{r\alpha} - \dot{\phi}_{r\alpha}\phi_{r\beta}}{\phi_{r\alpha}^2 + \phi_{r\beta}^2}$$

De plus, on a :

$$\begin{cases} \dot{\phi}_{r\alpha} &= -\eta\phi_{r\alpha} - n_p\Omega\phi_{r\beta} + M\eta i_{s\alpha} \\ \dot{\phi}_{r\beta} &= -\eta\phi_{r\beta} + n_p\Omega\phi_{r\alpha} + M\eta i_{s\beta} \end{cases}$$

Finalement, on obtient :

$$\Omega = \frac{1}{n_p}\left(\frac{\dot{\phi}_{r\beta}\phi_{r\alpha} - \dot{\phi}_{r\alpha}\phi_{r\beta}}{\phi_{r\alpha}^2 + \phi_{r\beta}^2} - M\eta\frac{\phi_{r\alpha}i_{s\beta} - \phi_{r\beta}i_{s\alpha}}{\phi_{r\alpha}^2 + \phi_{r\beta}^2}\right) \tag{5.24}$$

En combinant avec l'estimation des flux, la vitesse mécanique estimée sera :

$$\hat{\Omega} = \frac{1}{n_p}\left(\frac{\dot{\hat{\phi}}_{r\beta}\hat{\phi}_{r\alpha} - \dot{\hat{\phi}}_{r\alpha}\hat{\phi}_{r\beta}}{\hat{\phi}_{r\alpha}^2 + \hat{\phi}_{r\beta}^2} - M\eta\frac{\hat{\phi}_{r\alpha}i_{s\beta} - \hat{\phi}_{r\beta}i_{s\alpha}}{\hat{\phi}_{r\alpha}^2 + \hat{\phi}_{r\beta}^2}\right) \tag{5.25}$$

Cette méthode requiert une bonne connaissance de tous les paramètres de la machine.

Filtre de Kalman étendu

Si on considère la vitesse mécanique comme un état du système, et si on ne connait pas les paramètres mécaniques ainsi que les charges, les équations peuvent s'écrire comme l'équation (5.26).

$$\begin{bmatrix} \frac{di_{s\alpha}}{dt} \\ \frac{di_{s\beta}}{dt} \\ \frac{d\phi_{r\alpha}}{dt} \\ \frac{d\phi_{r\beta}}{dt} \\ \frac{d\Omega}{dt} \end{bmatrix} = \begin{bmatrix} \eta\mu\phi_{r\alpha} + \mu n_p\Omega\phi_{r\beta} - \gamma i_{s\alpha} + V_{s\alpha}/\left(\sigma L_s\right) \\ \eta\mu\phi_{r\beta} - \mu n_p\Omega\phi_{r\alpha} - \gamma i_{s\beta} + V_{s\beta}/\left(\sigma L_s\right) \\ -\eta\phi_{r\alpha} - n_p\Omega\phi_{r\beta} + M\eta i_{s\alpha} \\ -\eta\phi_{r\beta} + n_p\Omega\phi_{r\alpha} + M\eta i_{s\beta} \\ 0 \end{bmatrix} \tag{5.26}$$

Le vecteur d'état du système est $\mathbf{x} = [i_{s\alpha}, i_{s\beta}, \phi_{r\alpha}, \phi_{r\beta}, \Omega]^T$, le vecteur de sortie est $\mathbf{y} = [i_{s\alpha}, i_{s\beta}]^T$, les entrées sont $\mathbf{u} = [V_{s\alpha}, V_{s\beta}]^T$. Le système s'écrit sous la forme :

$$\begin{cases} \frac{d\mathbf{x}}{dt} &= \mathbf{A}(\Omega)\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{cases} \tag{5.27}$$

avec les matrices suivantes :

$$\mathbf{A}(\Omega) = \begin{bmatrix} -\gamma & 0 & \eta\mu & \eta n_p\Omega & 0 \\ 0 & -\gamma & -\eta n_p\Omega & \eta\mu & 0 \\ M\eta & 0 & -\eta & -n_p\Omega & 0 \\ 0 & M\eta & n_p\Omega & -\eta & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Le système en temps discret avec le temps d'échantillonnage T_e est obtenu en utilisant l'approximation de Taylor à l'ordre un :

$$\begin{cases} \mathbf{x}_{k+1} &= \mathbf{x}_k + T_e\left[\mathbf{A}(\Omega_k)\mathbf{x}_k + \mathbf{B}\mathbf{u}_k\right] \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k \end{cases} \tag{5.28}$$

L'observation est faite en utilisant un filtre de Kalman étendu comme décrit dans l'annexe A.1

5.3 Approche par formalisme d'état pour extraire la vitesse mécanique

5.3.1 Formulation du problème

Pour considérer la vitesse mécanique comme l'entrée, les quatre premières équations dans (5.23) sont utilisées.

$$\begin{bmatrix} \frac{di_{s\alpha}}{dt} \\ \frac{di_{s\beta}}{dt} \\ \frac{d\phi_{r\alpha}}{dt} \\ \frac{d\phi_{r\beta}}{dt} \end{bmatrix} = \begin{bmatrix} \eta\mu\phi_{r\alpha} + \mu n_p\Omega\phi_{r\beta} - \gamma i_{s\alpha} + V_{s\alpha}/(\sigma L_s) \\ \eta\mu\phi_{r\beta} - \mu n_p\Omega\phi_{r\alpha} - \gamma i_{s\beta} + V_{s\beta}/(\sigma L_s) \\ -\eta\phi_{r\alpha} - n_p\Omega\phi_{r\beta} + M\eta i_{s\alpha} \\ -\eta\phi_{r\beta} + n_p\Omega\phi_{r\alpha} + M\eta i_{s\beta} \end{bmatrix} \tag{5.29}$$

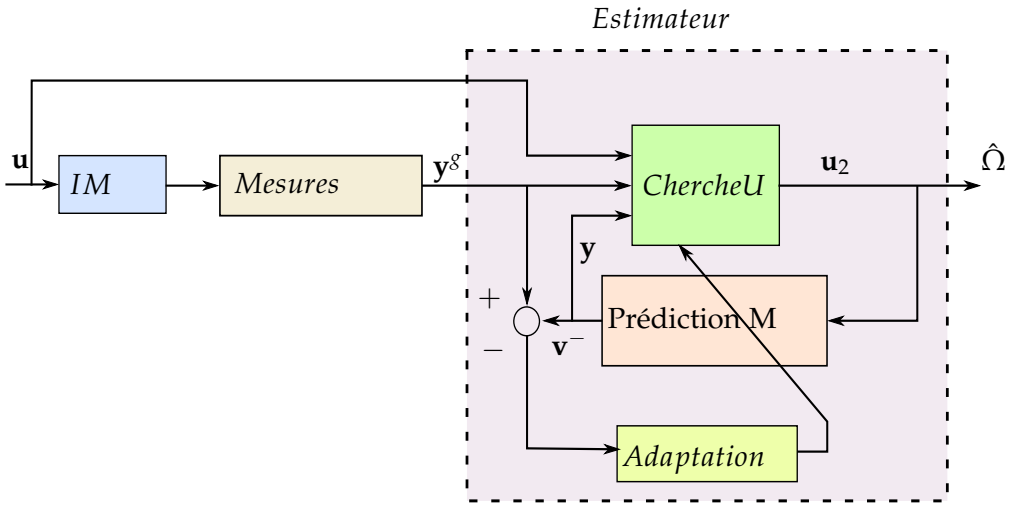


FIGURE 5.6 – *Algorithme d'observation de la vitesse d'une machine asynchrone*

Le vecteur d'état et des sorties sont $\mathbf{x} = [i_{s\alpha}, i_{s\beta}, \phi_{r\alpha}, \phi_{r\beta}]^T$ et $\mathbf{y} = [i_{s\alpha}, i_{s\beta}]^T$ respectivement. Le vecteur des entrées est $\mathbf{u}_2 = [V_{s\alpha}, V_{s\beta}, \Omega]^T$. L'algorithme est modifié comme indiqué sur la figure 5.6. Dans cette figure, le bloc "Prédiction M" contient la table de prédiction pour le système (5.29). L'algorithme de recherche de la commande "ChercheU" est appliquée connaissant les entrées $\mathbf{u} \subset \mathbf{u}_2$. Les mesures $\mathbf{y}^g = [i_{s\alpha}, i_{s\beta}]^T$ sont directement utilisées. Au pas courant, l'état estimé précédent $\hat{\mathbf{x}}_{k-1}$, le vecteur des entrées prédites $\hat{\mathbf{u}}_{2(k-1)}$ et la sortie prédite $\mathbf{y}_k^-$ sont disponibles. L'adaptation est effectuée en utilisant l'erreur entre la mesure courante $\mathbf{y}_k^g$ et sa valeur prédite $\mathbf{y}_k^-$. L'algorithme de recherche de la commande "ChercheU" permet de résoudre le problème de minimisation suivant :

$$\hat{\mathbf{u}}_{2,k} = \arg\min_{\mathbf{u}_2 \in \mathcal{U}_2} \left\| \mathbf{y}_k^g - \mathbf{P}_y \cdot p\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_2\right) \right\|_2 . \tag{5.30}$$

En appliquant $\hat{\mathbf{u}}_{2,k}$, l'état correspondant $\hat{\mathbf{x}}_k$ et la sortie prédite au pas suivant $\mathbf{y}_{k+1}^-$ sont calculés grâce au bloc "Prédiction M" et corrigés en utilisant l'adaptation. Clairement, le principe d'estimation est équivalent au problème de commande.

5.3.2 Résultats en simulation

Dans ce paragraphe, les résultats obtenus sous Matlab et Simulink sont présentés. Pour générer le profil de la vitesse, une commande scalaire (V/f) a été implémentée. Le modèle est construit sous Simulink avec les paramètres sont donnés dans la table 5.1 et le temps d'échantillonnage est $Ts = 600\mu s$. Ces paramètres sont également utilisés pour générer la table de prédiction. Les résultats comparés avec le filtre de Kalman étendu permettent d'évaluer la capacité de l'estimation proposée.

$J[kg * m^2]$	Rr[Ω]	Rs[Ω]	Lr[H]	Ls[H]	M[H]	np
$1.6e^{-2}$	3.75	11.0	0.47	0.04	0.47	2

TABLE 5.1 – *Paramètre de la machine asynchrone*

Dans la première simulation, la machine démarre à vitesse nulle. Ensuite, la vitesse est amenée à la fréquence de $50Hz$. Finalement, la machine est arrêtée. La figure 5.7 présente les tensions et les courants statoriques. La figure 5.8 montre la vitesse estimée en utilisant la méthode proposée (nommée Propose, la courbe en traits pointillés), la vitesse donnée par l'EKF (nommée EKF, la courbe en mince et solide) et la valeur réelle de la vitesse (nommée Mesure, la courbe solide et gris).

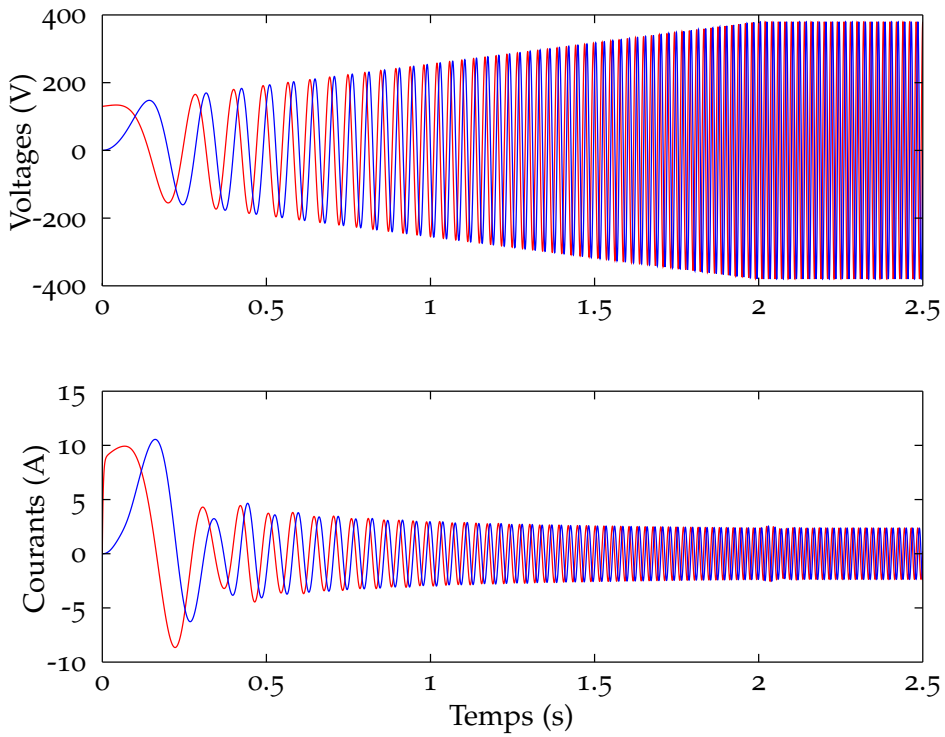


FIGURE 5.7 – *Tensions $V_{s\alpha}$, $V_{s\beta}$ et courants statoriques $i_{s\alpha}$, $i_{s\beta}$*

Dans le deuxième test, on va modifier quelques paramètres de la machine asynchrone. Le résistance rotorique est modifiée en prenant $Rr = 2 * Rr$, les mesures des

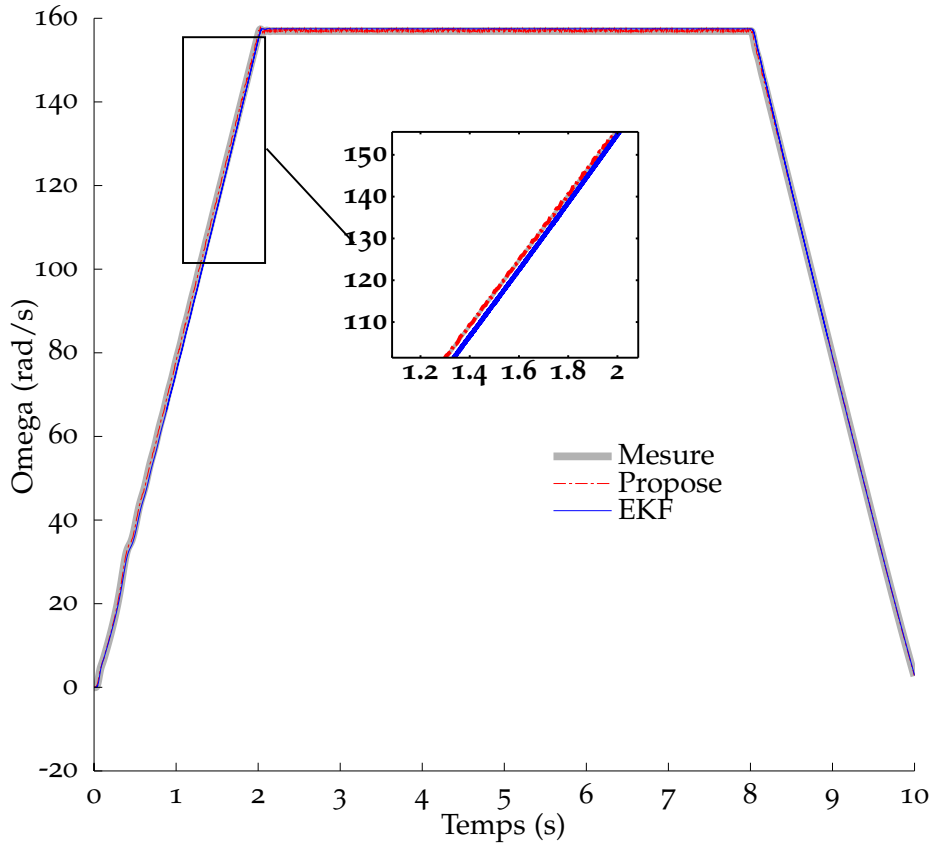


FIGURE 5.8 – *Vitesses estimées et ses références* Ω

courants statoriques $i_{s\alpha}$, $i_{s\beta}$ sont bruités. Le profil de la vitesse est plus complexe avec les fréquences de $30Hz, 40Hz$ et $50Hz$. La configuration du filtre EKF n'est pas modifiée, ainsi que la table de prédiction précédente pour vérifier la robustesse. Les courbes de résultats sur la figure 5.9 montrent que l'estimateur a la capacité de converger. De plus, on constate que l'erreur à vitesse faible est plus petite pour la méthode proposée que celle du filtre EKF. La figure 5.10 montre l'erreur de la vitesse estimée pour chaque méthode. Le méthode proposée contient un filtre de passe-bas qui filtre un peu le bruit.

5.4 MRAS basée sur les flux

5.4.1 Formulation du problème

Considérons maintenant le système (5.23), à partir des équations, on obtient :

$$\mu \frac{d\Phi_r}{dt} = -\frac{d\mathbf{I}_s}{dt} + \frac{1}{\sigma L_s}(\mathbf{V}_s - R_s \mathbf{I}_s) \tag{5.31}$$

En utilisant $\mathbf{x} = \mu \frac{d\Phi_r}{dt}$, le modèle peut s'écrire :

$$\begin{cases} \mathbf{x} &= -\frac{d\mathbf{I}_s}{dt} + \frac{1}{\sigma L_s}(\mathbf{V}_s - R_s \mathbf{I}_s) \\ \frac{d\mathbf{x}}{dt} &= -\mathbf{A}\mathbf{x} + \mu M \eta \frac{d\mathbf{I}_s}{dt} \end{cases} \tag{5.32}$$

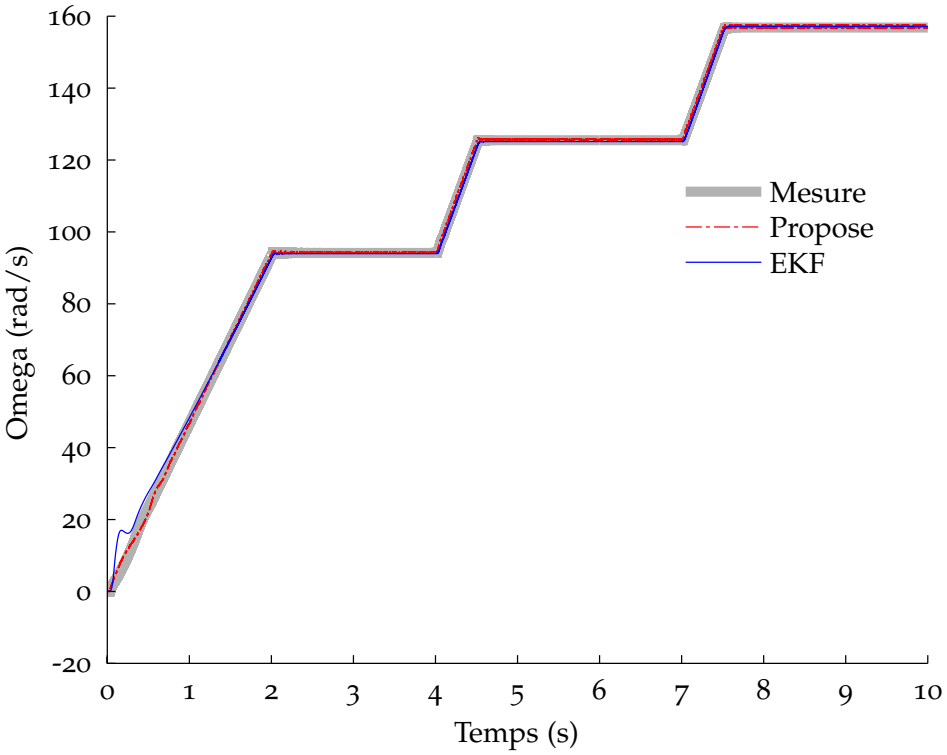


FIGURE 5.9 – *Vitesses mécaniques estimées et ses références* Ω

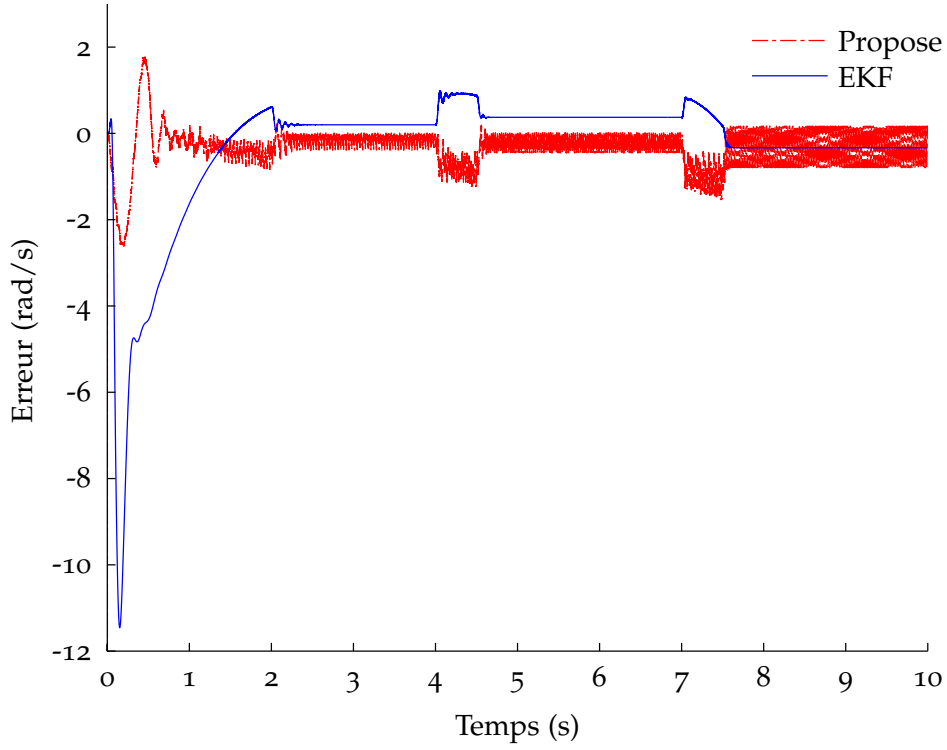


FIGURE 5.10 – *Erreur des vitesses estimées*

Cette variable est similaire à celle de Peng et Fukao (1994) pour le schéma MRAS.

$$\begin{cases} \mathbf{x} &= -\dfrac{d\mathbf{I}_s}{dt} + \dfrac{1}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s) \\ \dfrac{d\mathbf{x}}{dt} &= -\mathbf{A}\mathbf{x} + \mu M\eta\left(-\mathbf{x} + \dfrac{1}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s)\right) \end{cases} \tag{5.33}$$

A chaque étape, Ω, l'entrée du système, est calculée telle que le système

$$\frac{d\mathbf{x}}{dt} = -\mathbf{A}\mathbf{x} + \mu M\eta\left(-\mathbf{x} + \frac{1}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s)\right)$$

suive bien la trajectoire $\mathbf{x} = -\dfrac{d\mathbf{I}_s}{dt} + \dfrac{1}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s)$. C'est la différence entre la méthode proposée ici et le MRAS (Peng et Fukao (1994)).

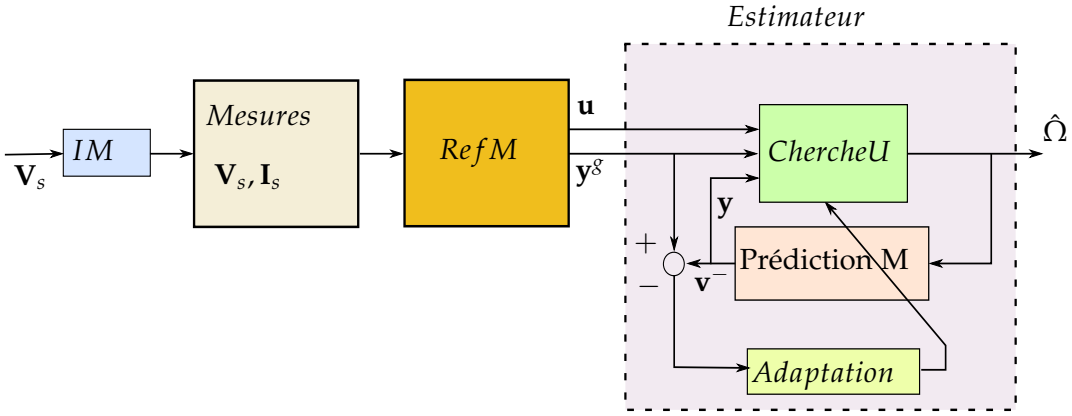


FIGURE 5.11 – *Observation de la vitesse basée sur MRAS*

L'état et la sortie sont $\mathbf{x}$ et $\mathbf{y} = \mathbf{x}$ respectivement. Le vecteur $\mathbf{u} = \mathbf{V}_s - R_s\mathbf{I}_s$ est l'entrée externe du système. L'algorithme est modifié comme indiqué sur la figure 5.11. Sur cette figure, les tensions et les courants statoriques sont disponibles à partir des capteurs de la machine "IM". Le bloc "Prédiction M" contient la table de prédiction pour le système (5.33). L'algorithme "ChercheU" est appliqué connaissant les entrées externes $\mathbf{u}$. Les mesures $\mathbf{y}^g = -\dfrac{d\mathbf{I}_s}{dt} + \dfrac{1}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s)$ sont les sorties du bloc "Ref M" qui calcule $\mathbf{y}^g$ et $\mathbf{u}$. Au pas courant, l'état estimé précédent $\hat{\mathbf{x}}_{k-1}$, la vitesse estimée $\hat{\Omega}_{(k-1)}$ et la sortie prédite $\mathbf{y}_k^-$ sont disponibles. Puis, l'adaptation est effectuée en utilisant l'erreur entre $\mathbf{y}_k^g$ et $\mathbf{y}_k^-$. L'algorithme "ChercheU" permet de trouver la solution du problème de minimisation suivant :

$$\hat{\Omega}_k = \arg\min_{\Omega\in\mathcal{U}} \left\|\mathbf{y}_k^g - \mathbf{P}_y \cdot p\left(\hat{\mathbf{x}}_{k-1}, \Omega\right)\right\|_2. \tag{5.34}$$

Quand la vitesse $\hat{\Omega}_k$ est calculée, l'état correspondant $\hat{\mathbf{x}}_k$ et la sortie prédite au pas suivant $\mathbf{y}_{k+1}^-$ sont déterminés avec l'aide du bloc "Prédiction M" et corrigés en utilisant l'adaptation de l'équation (1.8.

5.4.2 Résultats en simulation

Les paramètres de la machine sont donnés dans la table 5.1. Dans la première simulation, la machine démarre à la vitesse nulle. Ensuite, la vitesse sera amenée à la fréquence de $50Hz$. Finalement, la machine est arrêtée. Une charge égale à $T_L = Cr = 10 \ \ Nm$ est appliquée à partir $T = 5.5\ s$. La figure 5.12 présente les tensions et les courants statoriques. Au temps $Ts = 5.5\ s$ les courants changent à cause de la charge. La figure 5.13 montre la vitesse estimée en utilisant la méthode proposée (nommée Propose, la courbe en traits pointillés), la vitesse donnée par la méthode MRAS citée (nommée OpenLoop, la courbe en mince et solide) et la valeur vraie de la vitesse (la courbe solide et gris).

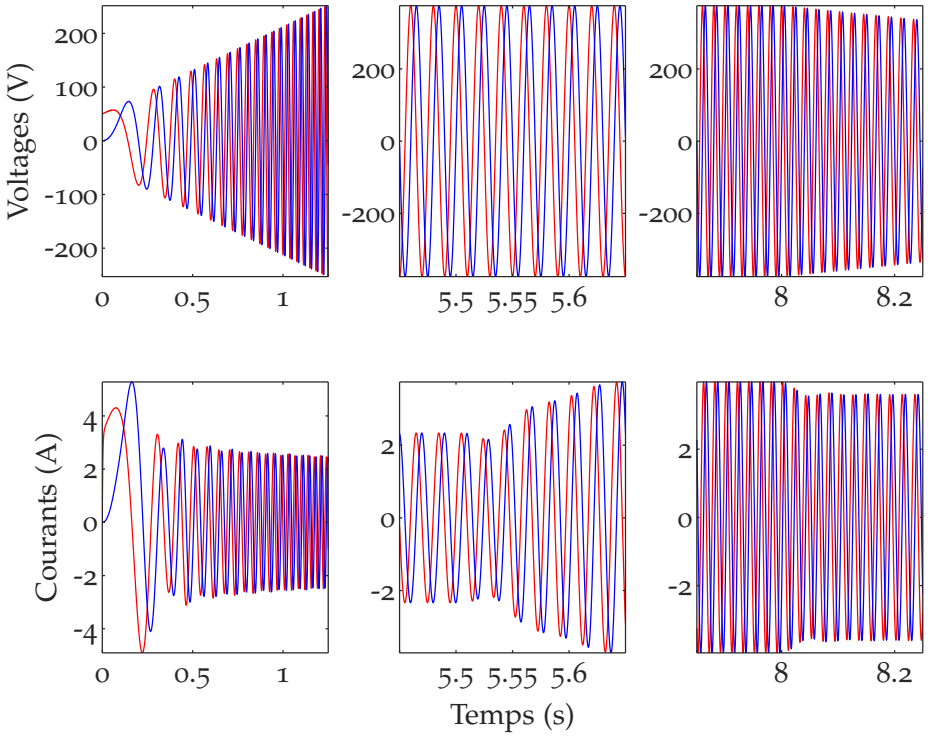


FIGURE 5.12 – *Tensions $V_{s\alpha}$, $V_{s\beta}$ et courants $i_{s\alpha}$, $i_{s\beta}$*

Dans le deuxième test, on modifie quelque paramètres de la machine asynchrone. Le résistance statorique est modifiée pour $Rs = 2 * Rs$. La charge $T_L = Cr = 10\ Nm$ est appliquée à partir $T = 6\ s$. Le profil de la vitesse est plus complexe avec les fréquences de $30Hz$, $40Hz$ et $50Hz$. La table de prédiction reste identique à l'essai précédent pour vérifier la robustesse. Les courbes de résultats sur la figure 5.14 montrent que l'estimateur a la capacité de converger. De plus, on constate que l'erreur à vitesse faible pour la méthode proposée est plus petite que celle calculée par la méthode MRAS. La figure 5.15 montre l'erreur de la vitesse estimée pour chaque méthode. Ce test montre que la performance dépend du Rs et que la méthode proposée est robuste par rapport à la variation de la résistance statorique.

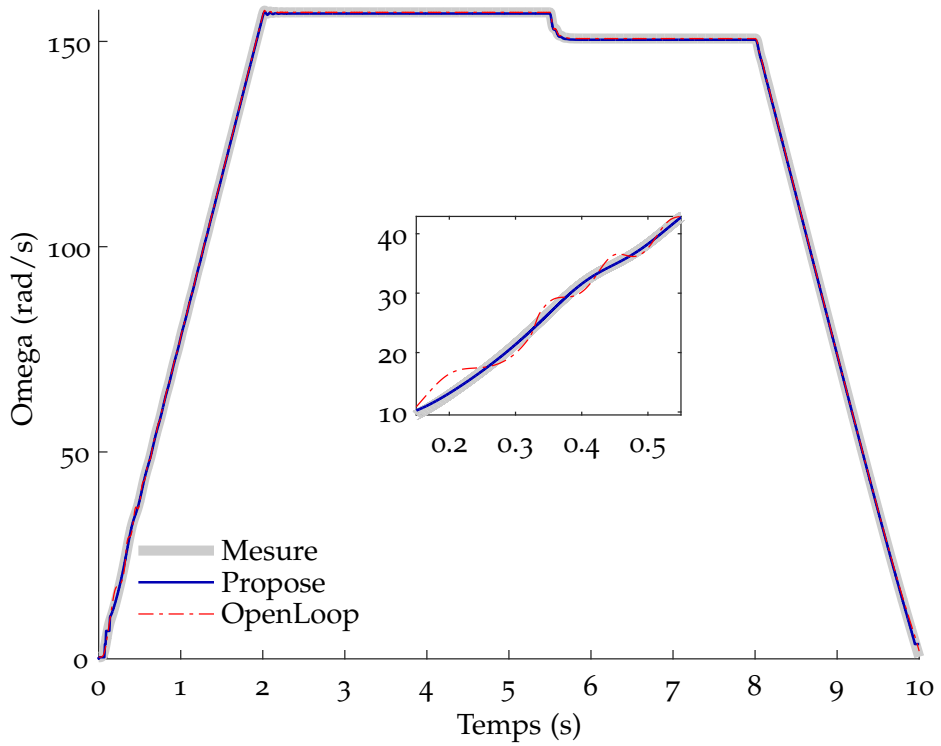


FIGURE 5.13 – *Vitesses mécaniques estimées et ses références* Ω

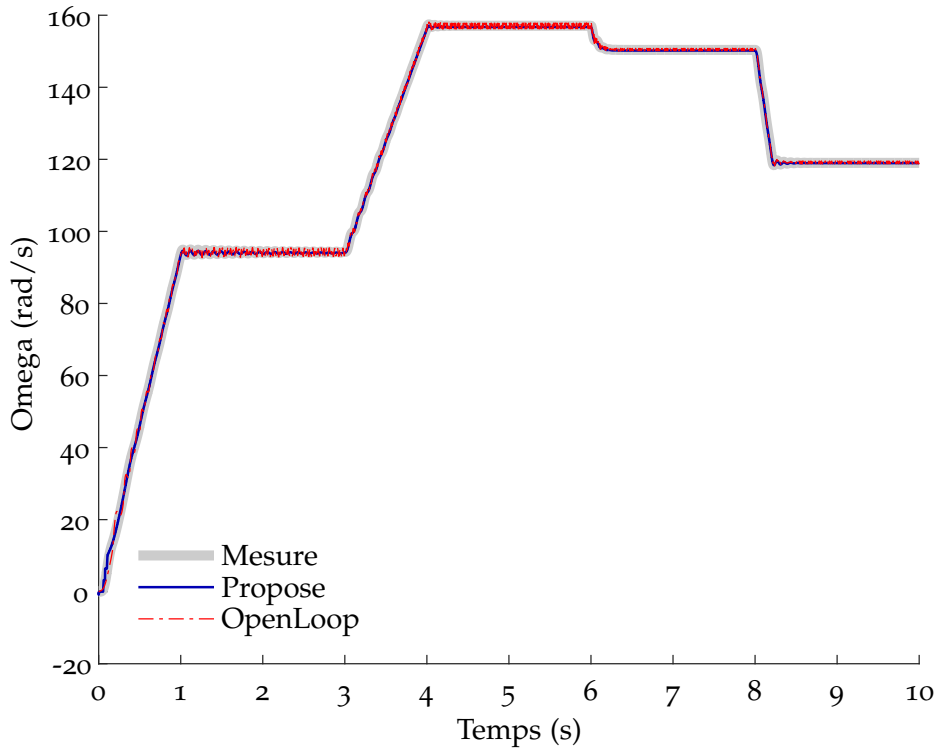


FIGURE 5.14 – *Vitesses mécaniques estimées et ses références* Ω

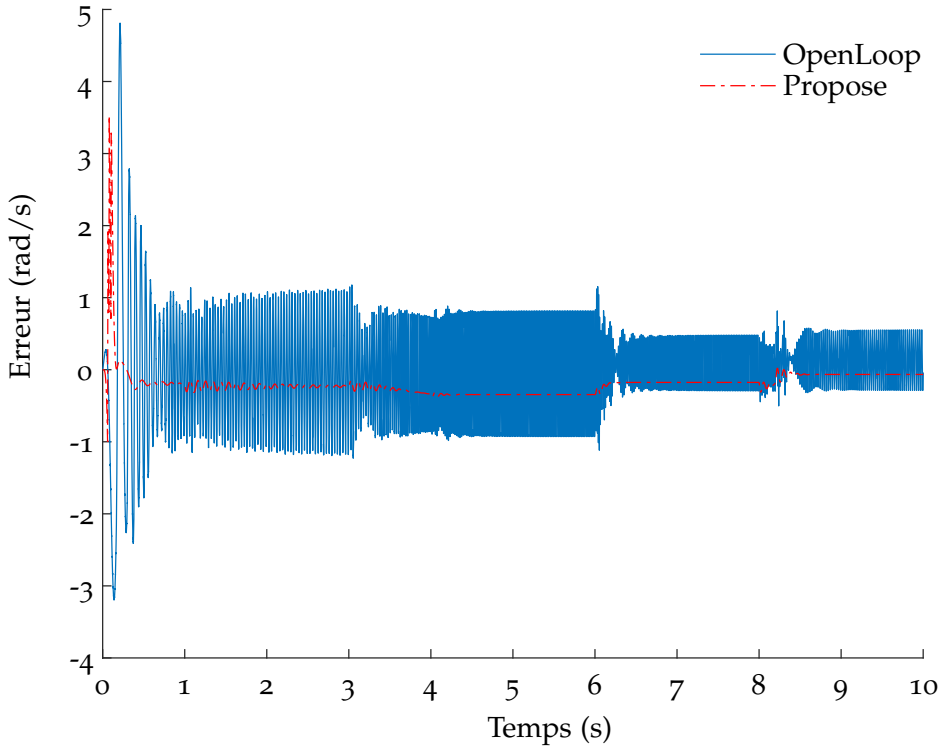


FIGURE 5.15 – *Erreur des vitesses estimées avec la variation de la résistance statorique*

Dans le troisième test, la résistance rotorique est modifiée pour $Rr = 1.25 * Rr$. Ce changement provoque un écart important entre le modèle réel et la table de prédiction. Les autres conditions sont les mêmes que dans le deuxième test. La figure 5.16 affiche les vitesses estimées par les deux méthodes. On trouve qu'il existe une erreur statique en régime permanent en charge. De plus, on constate que l'erreur de vitesse est plus faible pour la méthode proposée que pour la méthode de comparaison.

5.5 MRAS basé sur le modèle en courants

Dans cette section, on essaie d'apprendre la table de comportement. Pour ce faire, le modèle de la machine est représenté en utilisant les équations statoriques. Les signaux de référence sont les courants statoriques qui sont mesurés directement. Les signaux de référence ne dépendent donc pas des paramètres de la machine comme pour la proposition précédente. A partir des équations (5.14) et (5.15), les flux rotoriques peuvent s'écrire :

$$\mu\Phi_r = -\mathbf{I}_s + \frac{1}{\sigma L_s}\Phi_s \tag{5.35}$$

De plus

$$\frac{d\mathbf{I}_s}{dt} = \mu \begin{bmatrix} \eta & n_p\Omega \\ -n_p\Omega & \eta \end{bmatrix} \Phi_r - \gamma\mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{V}_s$$

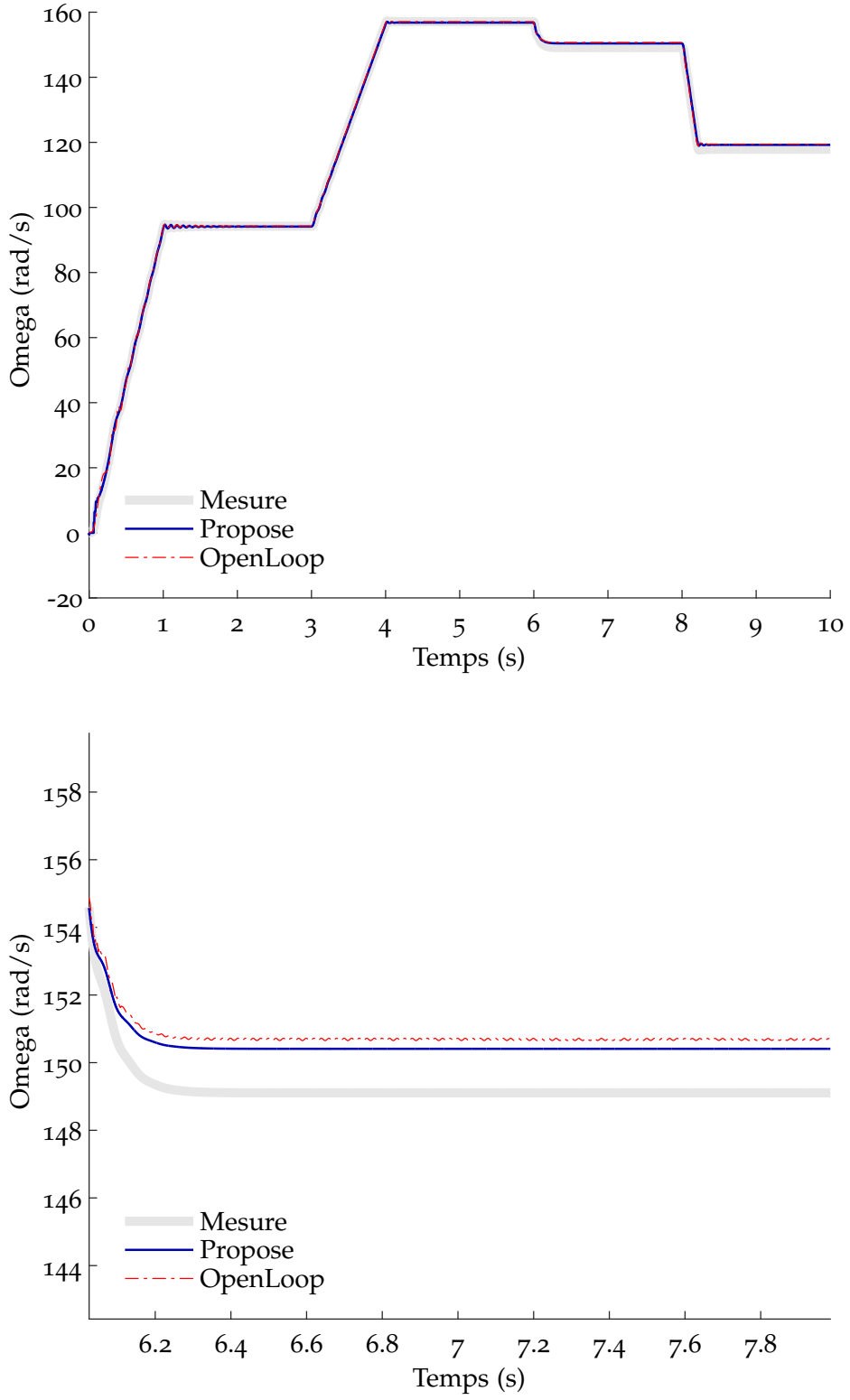


FIGURE 5.16 – *Vitesses mécaniques estimées et ses références Ω, le zoom sur la vitesse montre l'erreur statique.*

Donc, l'équation ramenée au stator est obtenue comme suit :

$$\frac{d\mathbf{I}_s}{dt} = \begin{bmatrix} \eta & n_p\Omega \\ -n_p\Omega & \eta \end{bmatrix} (-\mathbf{I}_s + \frac{1}{\sigma L_s}\Phi_s) - \gamma \mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{V}_s \tag{5.36}$$

Les flux statoriques peuvent être calculés par l'intégration de l'équation :

$$\frac{d\Phi_s}{dt} = \mathbf{V}_s - R_s\mathbf{I}_s$$

Cependant, il faut avoir connaissance des flux initiaux. Pour éviter ce problème, dans le cas où les signaux sont sinusoïdaux en régime permanent (ce qui est une hypothèse faible pour la plupart des fonctionnements industriels), on peut calculer l'intégrale du vecteur par une rotation de $-\pi/2$ et une division par la pulsation électrique. Φ_s est donc directement extrapolé de $\frac{d\Phi_s}{dt}$. L'approximation de flux s'écrit :

$$\Phi_s \approx \frac{1}{\omega_e}\begin{bmatrix} V_{s\beta} - R_s I_{s\beta} \\ -V_{s\alpha} + R_s I_{s\alpha} \end{bmatrix} \tag{5.37}$$

où ω_e est la pulsation électrique. Les deux équations (5.36) et (5.37) permettent de décrire l'évolution des courants en fonction des tensions et de la vitesse électrique ω_e. La table du comportement de la machine est représentée par la relation suivante :

$$\mathbf{I}_{s(k+1)} \approx F(\mathbf{I}_{s(k)}, \mathbf{V}_{s(k)}, \omega_{e(k)}, T_s) \tag{5.38}$$

avec T_s le temps d'échantillonnage. Cette relation est valide dans un repère arbitraire. On va utiliser les deux repères les plus connus : le repère fixe au stator $\alpha\beta$ et le repère tournant à la vitesse ω_e. La machine est contrôlée par une commande scalaire V/f avec la charge variable à chaque essai pour générer des données $\mathbf{I}_{s(k+1)}, \mathbf{I}_{s(k)}, \mathbf{V}_{s(k)}, \omega_{e(k)}$. La table de prédiction présentant l'équation (5.38) est initialisée à zéro. Chaque essai et chaque itération de l'algorithme d'apprentissage vont modifier cette table.

Après plusieurs itérations d'apprentissage, l'observation est effectuée. Par exemple, la machine asynchrone est démarrée jusqu'à la pulsation $\omega_e = 100\pi\ rad.s^{-1}$, à partir de $t = 5.5\ s$, une charge $T_L = 10\ Nm$ est appliquée. La figure 5.17 montre les courants dans le repère tournant à la vitesse de synchronisation ω_e. Dans ce repère les signaux ne varient que très peu en régime permanent. Les courants prédits suivent bien leurs références. La figure 5.18 donne les vitesses prédite et mesurée. Il existe une erreur statique de prédiction. Ceci peut s'expliquer du fait que l'algorithme doit utiliser des valeurs où les éléments de la table n'ont pas été suffisamment modifiés lors de l'apprentissage. Ainsi on peut penser qu'une adaptation continue de la table – qui n'a pas été mise en place dans cet essai – pourrait permettre une convergence pour les points peu impactés lors de la phase hors ligne.

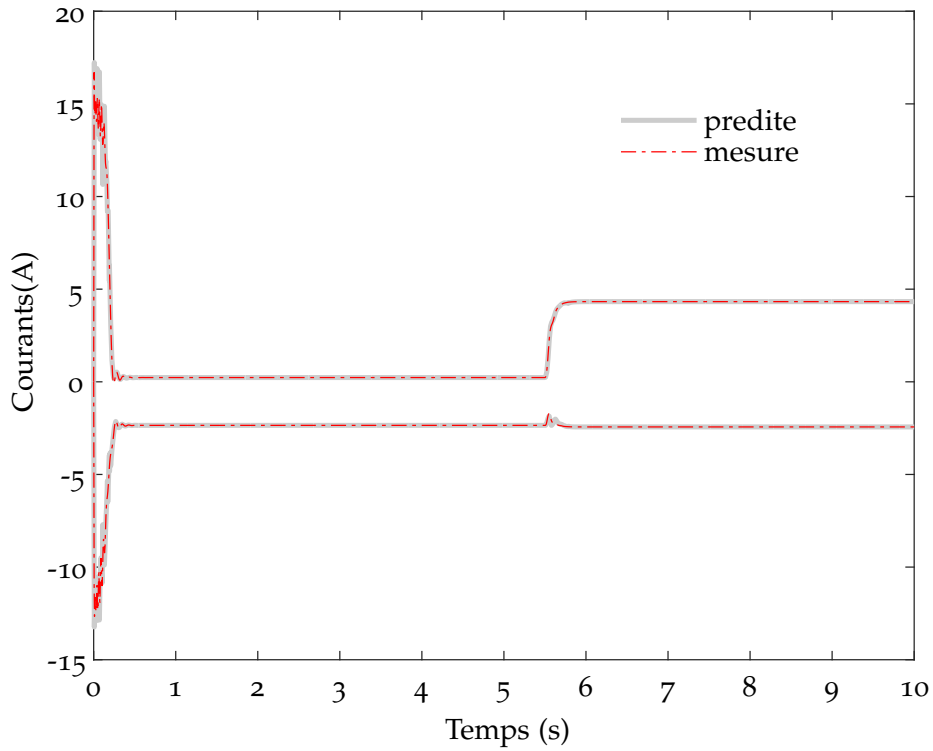


FIGURE 5.17 – *Courants prédits i_{sd}, i_{sq} et leurs références*

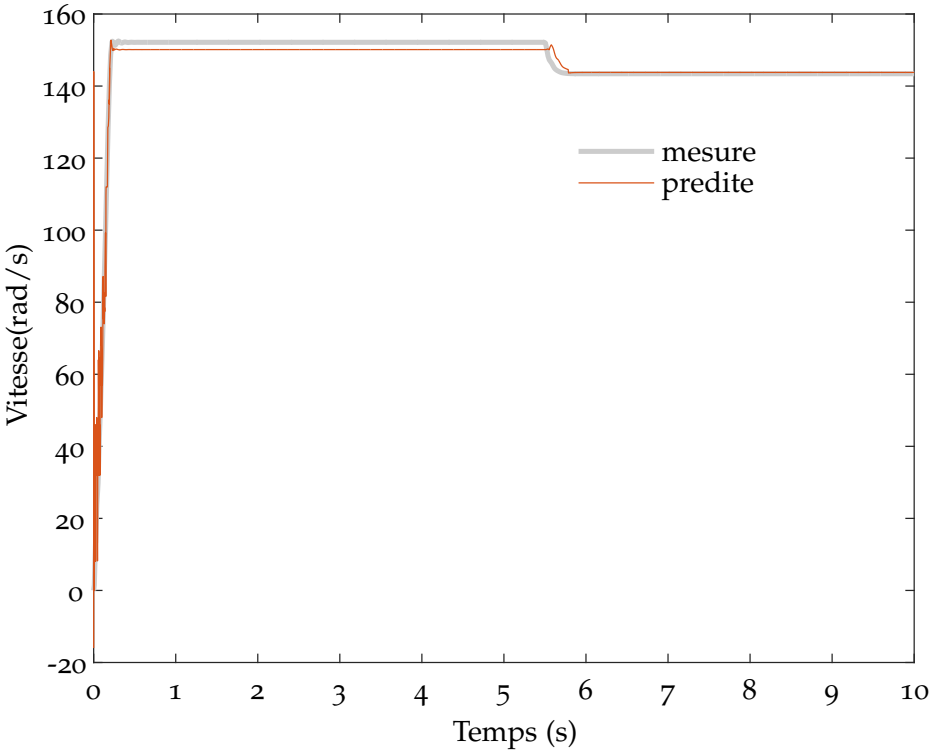


FIGURE 5.18 – *Vitesses mécaniques estimées et leurs références* Ω

Conclusion du chapitre

Dans ce chapitre, les méthodes pour modéliser la machine asynchrone ont été étudiées. L'obtention les équations dynamiques a permis de proposer une première version d'une nouvelle méthode pour extraire la vitesse mécanique de la machine asynchrone. D'autres versions pour estimer la vitesse mécanique ont également été étudiées. Dans certaines configurations il est nécessaire d'avoir une estimation des paramètres la plus précise possible. Dans ce cas, il faudra identifier les paramètres de la machine. Comme méthode d'identification on peut, par exemple, utiliser la méthode des moindres carrés en alignant les données comme montré dans l'annexe (A.2). De nouvelles méthodes ont été proposées pour observer la vitesse mécanique de la machine sans capteur mécanique. L'observateur proposé est basé sur le modèle de comportement tabulé où l'erreur peut être corrigée en ligne. Donc, il semblerait qu'il ne dépende pas explicitement des paramètres de la machine et pourrait être appliqué sur certaines machines sans calibration. Le modèle de comportement a été premièrement construit par simulation. Il peut être amélioré par apprentissage comme dans la dernière section. Cette méthode semble suffisamment robuste pour se comparer aux différentes propositions de la littérature. Pour autant, il sera nécessaire de valider les résultats par des essais réels avec des données bruitées.

Conclusion générale et Perspectives

L'objectif de cette thèse a été de contribuer au développement d'une commande non linéaire qui ne nécessite que très peu de réglage, sans pour autant avoir un modèle précis du système à contrôler, et qui respecte la dynamique théorique définie par les spécifications quelles que soient les dynamiques négligées, les variations paramétriques du modèle et les perturbations. L'idée de départ a été d'utiliser la commande par modèle de comportement numérique tabulé déjà développée, basée sur le modèle d'état du système. Après étude des méthodes pour obtenir le modèle approximé du système par apprentissage, en utilisant la connaissance réelle du système à partir des mesures, la commande a été modifiée au niveau de son utilisation de la table de prédiction. Un mécanisme a été ajouté pour modifier la table de prédiction directement, ce qui permet d'obternir la table de prédiction le plus précise. La commande basée sur le modèle d'état requiert un observateur pour construire l'état du système. Cela introduit des retards et nécessite l'utilisation d'une référence interne stable. Le passage à l'utilisation d'une représentation entrée-sortie permet d'éviter ces problèmes et d'utiliser directement les mesures de capteurs. Une nouvelle version de la commande basée sur le modèle entrée-sortie a été développée.

Un apprentissage simple de la table de prédiction a été proposée. Mais la représentation des données par une grille (ou tout autre moyen), appris à partir du système, présente le défaut de posséder des portions de l'espace non utilisable car non parcourues précédemment par les données disponibles. Dans ce cas, il est toujours possible d'initialiser la table avec un comportement approché comme on a pu le voir sur l'exemple du PVTOL.

La commande a été appliquée sur les véhicules aériens et les véhicule autonomes dans le cardre de suivi de trajectoire et également à une machine asynchrone pour pour estimer sa vitesse mécanique.

Perspectives

Pour la commande d'attitude du quadricoptère, il faudra ajouter une table pour la stabilisation de la position et un observateur pour reconstruire la matrice de rotation du système afin de faire les tests réels. Pour la commande appliquée à la voiture en drift, il faudra avoir accès à une mesure/estimation de la vitesse latérale. La commande pour la voiture Zoé sera améliorée en utilisant un meilleur algorithme de localisation. Bien que l'observation de la vitesse mécanique de la machine asynchrone a été effectuée

en simulation et seulement pour extraire la vitesse mécanique, les utilisateurs de la commande pourront manipuler cet algorithme pour attaquer le problème d'ajustement dynamique afin trouver les paramètres d'un processus dynamique.

Un des axes non exploré offert par l'utilisation d'un table de prédiction est de travailler sur un horizon fini par tirages aléatoires successifs d'autant de séries de commandes possibles sur une période d'échantillonnage et de ne garder que celle qui minimise un critère spécifique. On pourrait alors commuter entre les deux modes de commande en fonction des besoins sans changer les données d'entrée.

Enfin, une des améliorations les plus importantes qu'il faudra mettre au point sera de moduler la performance du système commande/observateur en fonction d'un critère de qualité des données de la table afin d'éviter des comportements instables dus à l'utilisation de données peu apprises.

Chapitre A

Annexes

Sommaire

A.1 Filtre de Kalman étendu

Le filtre de Kalman étendu est utilisé pour estimer l'état d'un système non linéaire sous la forme en temps discret suivante :

$$\begin{cases} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k \\ \mathbf{y}_k &= g(\mathbf{x}_k) + \beta_k \end{cases} \tag{A.1}$$

avec α_k, β_k les bruits gaussiens et les matrices de covariance $\mathbf{Q}$ et $\mathbf{R}$. Le principe est d'utiliser une linéarisation autour d'un point de référence ou autour des valeurs prédites à l'étape précédente. Il consiste en deux étapes : étape d'estimation et étape de prédiction.

Estimation

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{C}_k^T(\mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}_k^T + \mathbf{R})^{-1}$$
$$\mathbf{C}_k = \frac{\partial g}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k|k-1})$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{y}_k - g(\hat{\mathbf{x}}_{k|k-1})\right)$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)^T + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^T$$

Prédiction

$$\hat{\mathbf{x}}_{k+1|k} = f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)$$
$$\mathbf{P}_{k+1|k} = \mathbf{A}_k\mathbf{P}_{k|k}\mathbf{A}_k + \mathbf{Q}$$
$$\mathbf{A}_k = \frac{\partial f}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)$$

A.2 Identification des paramètres d'une machine asynchrone

Rappelons que :

$$\begin{bmatrix} \frac{di_{s\alpha}}{dt} \\ \frac{di_{s\beta}}{dt} \\ \frac{d\phi_{r\alpha}}{dt} \\ \frac{d\phi_{r\beta}}{dt} \end{bmatrix} = \begin{bmatrix} \eta\mu\phi_{r\alpha} + \mu n_p\Omega\phi_{r\beta} - \gamma i_{s\alpha} + V_{s\alpha}/(\sigma L_s) \\ \eta\mu\phi_{r\beta} - \mu n_p\Omega\phi_{r\alpha} - \gamma i_{s\beta} + V_{s\beta}/(\sigma L_s) \\ -\eta\phi_{r\alpha} - n_p\Omega\phi_{r\beta} + M\eta i_{s\alpha} \\ -\eta\phi_{r\beta} + n_p\Omega\phi_{r\alpha} + M\eta i_{s\beta} \end{bmatrix} \tag{A.2}$$

avec

$$\mathbf{A} = \begin{bmatrix} \eta & n_p\Omega \\ -n_p\Omega & \eta \end{bmatrix}$$

$$\begin{cases} \frac{d\mathbf{I}_s}{dt} &= \mu\mathbf{A}\Phi_r - \gamma\mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{V}_s \\ \frac{d\Phi_r}{dt} &= -\mathbf{A}\Phi_r + M\eta\mathbf{I}_s \end{cases} \tag{A.3}$$

$$\frac{d\Phi_r}{dt} = \frac{1}{\mu}\left(-\frac{d\mathbf{I}_s}{dt} + (\mu M\eta - \gamma)\mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{V}_s\right)$$

donc, si $\omega = const$

$$\frac{d^2\mathbf{I}_s}{dt^2} = \mu\mathbf{C}\frac{d\Phi_r}{dt} - \gamma\frac{d\mathbf{I}_s}{dt} + \frac{1}{\sigma L_s}\frac{d\mathbf{V}_s}{dt}$$

d'où

$$\frac{d^2\mathbf{I}_s}{dt^2} = (-\gamma\mathbf{I_2} - \mathbf{A})\frac{d\mathbf{I}_s}{dt} + (\mu M\eta - \gamma)\mathbf{A}\mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{A}\mathbf{V}_s + \frac{1}{\sigma L_s}\frac{d\mathbf{V}_s}{dt}$$

En remplaçant $\gamma = \mu M\eta + \frac{R_s}{\sigma L_s}$, on obtient :

$$\frac{d^2\mathbf{I}_s}{dt^2} = (-\gamma\mathbf{I_2} - \mathbf{A})\frac{d\mathbf{I}_s}{dt} - \frac{R_s}{\sigma L_s}\mathbf{A}\mathbf{I}_s + \frac{1}{\sigma L_s}\mathbf{A}\mathbf{V}_s + \frac{1}{\sigma L_s}\frac{d\mathbf{V}_s}{dt}$$

Cela donne :

$$\frac{d^2\mathbf{I}_s}{dt^2} + n_p\Omega\mathbf{P}(-\pi/2)\frac{d\mathbf{I}_s}{dt} = -(\eta+\gamma)\frac{d\mathbf{I}_s}{dt} + \frac{\eta}{\sigma L_s}(\mathbf{V}_s - R_s\mathbf{I}_s) + \frac{1}{\sigma L_s}\left[n_p\Omega\mathbf{P}(-\pi/2)(\mathbf{V}_s - R_s\mathbf{I}_s) + \frac{d\mathbf{V}_s}{dt}\right]$$

En considérant les données à partir de cette forme, les paramètres peuvent trouvés utilisant une méthode des moindres de carrés.

Bibliographie

Mujahid Abdulrahim. On the dynamics of automobile drifting, 2006/04/03 2006. URL `http://dx.doi.org/10.4271/2006-01-1019`. (Cité page 90.)

Lazhar Ben-Brahim et Ryoichi Kurosawa. Identification of induction motor speed using neural networks. Dans *Conference Record of the Power Conversion Conference, Yokohama*, pages 689–694. IEEE, 1993. (Cité page 118.)

A. Benine-Neto et C. Grand. Piecewise affine control for fast unmanned ground vehicles. Dans *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3673–3678, 2012. (Cité page 77.)

M. Benosman, Fang Liao, Kai-Yew Lum, et Jian Liang Wang. Nonlinear control allocation for non-minimum phase systems. *IEEE Transactions on Control Systems Technology*, 17(2) :394–404, March 2009. ISSN 1063-6536. (Cité page 47.)

L. Benvenuti, P. Di Giamberardino, et L. Farina. Trajectory tracking for a PVTOL aircraft : a comparative analysis. Dans *Proceedings of the 35th IEEE Conference on Decision and Control*, volume 2, pages 1563–1568, Dec 1996. (Cité page 47.)

Stéphane Bonnet. *Appoches numériques pour la commande des systèmes dynamiques*. Thèse de doctorat, Université de Technologie de Compiègne, France, 2008. (Cité pages 1, 3, 5, 8 et 11.)

D.S. Broomhead et D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2 :321–355, 1988. (Cité page 21.)

Bruce Burton, Farrukh Kamran, Ronald G Harley, Thomas G Habetler, Martin A Brooke, et Ravi Poddar. Identification and control of induction motor stator currents using fast on-line random training of a neural network. *IEEE Transactions on Industry Applications*, 33(3) :697–704, 1997. (Cité page 118.)

L. Buşoniu, R. Babuška, B. De Schutter, et D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010. (Cité pages 26, 31 et 38.)

Ronnapee Chaichaowarat et Witaya Wannasuphoprasit. 7th ifac symposium on advances in automotive controloptimal control for steady state drifting

of rwd vehicle. *IFAC Proceedings Volumes*, 46(21) :824–830, 2013. ISSN 1474-6670. URL http://www.sciencedirect.com/science/article/pii/S1474667016384774. (Cité page 90.)

N. A. Chaturvedi, A. K. Sanyal, et N. H. McClamroch. Rigid-body attitude control. *IEEE Control Systems*, 31(3) :30–51, 2011. ISSN 1066-033X. (Cité page 63.)

Mihai Comanescu et Longya Xu. Sliding-mode mras speed estimators for sensorless vector control of induction machine. *IEEE Transactions on Industrial Electronics*, 53(1) : 146–153, 2006. (Cité page 118.)

J. De Miras et S. Bonnet. Nonlinear control of a magnetic levitation shaft by numerical inversion of its behavioral model. Dans *13th IEEE European Control Conference*, Strasbourg, France, June 2014. (Cité pages 5 et 7.)

Moustapha Doumiati, Ali Charara, Alessandro Victorino, Daniel Lechner, et Bernard Dubuisson. *Vehicle Dynamics Estimation using Kalman Filtering*. Wiley-Blackwell, dec 2012. URL http://dx.doi.org/10.1002/9781118578988. (Cité page 101.)

Y. Engel, S. Mannor, et R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8) :2275–2285, 2004. ISSN 1053-587X. URL <GotoISI>://WOS:000222760500012. 839CI Times Cited :188 Cited References Count :31. (Cité page 24.)

Yaakov Engel, Shie Mannor, et Ron Meir. Sparse online greedy support vector regression. Dans Tapio Elomaa, Heikki Mannila, et Hannu Toivonen, éditeurs, *Machine Learning : ECML 2002*, volume 2430 de *Lecture Notes in Computer Science*, book section 8, pages 84–96. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44036-9. URL http://dx.doi.org/10.1007/3-540-36755-1_8. (Cité page 25.)

I. Fantoni, R. Lozano, et P. Castillo. A simple stabilization algorithm for the PVTOL aircraft. Dans *15th IFAC World Congress*, Barcelona, Spain, July 2002a. (Cité page 47.)

I. Fantoni, R. Lozano, et A. Palomino. Global stabilizing control design for the PVTOL aircraft using saturation functions on the inputs. Dans *European Control Conference, ECC2003*, Cambridge, UK, September 2003. (Cité page 47.)

I. Fantoni, A. Zavala, et R. Lozano. Global stabilization of a PVTOL aircraft with bounded thrust. Dans *IEEE Conference on Decision and Control, CDC 02*, Las Vegas, USA, December 2002b. (Cité page 47.)

M. Fliess et C. Join. Model free control and intelligent pid controllers: towards a possible trivialization of nonlinear control?. Dans 15^{th} *IFAC Symp. System Identif.*, Saint-Malo, 2009.
http://hal.archives-ouvertes.fr/inria-00372325/fr/. (Cité page 34.)

Michel Fliess, Hebertt Sira-Ramirez, et Richard Márquez. Regulation of non-minimum phase outputs : a flatness based approach. Dans D. Normand-Cyrot, éditeur, *Perspectives in Control - Theory and Applications : A Tribute to Ioan Dore Landau*, pages 143–164. Springer-Verlag, London, June 19–23 1998. (Cité page 48.)

Christophe Forgez. Identification paramétrique de la machine asynchrone. *UV.UTC*, 12 (6) :3–5, 2005. (Cité page 118.)

Rogelio Francisco, Frederic Mazenc, et Sabine Mondie. Global asymptotic stabilization of a PVTOL aircraft model with delay in the input. Dans John Chiasson et JeanJacques Loiseau, éditeurs, *Applications of Time Delay Systems*, volume 352 de *Lecture Notes in Control and Information Sciences*, pages 343–356. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-49555-0. (Cité page 47.)

E. Fresk et G. Nikolakopoulos. Full quaternion based attitude control for a quadrotor. Dans *Control Conference (ECC), 2013 European*, pages 3864–3869, 2013. (Cité page 63.)

S. Fuchshumer, K. Schlacher, et T. Rittenschober. Nonlinear vehicle dynamics control - a flatness based approach. Dans *44th IEEE Conference on Decision and Control, European Control Conference. CDC-ECC*, pages 6492–6497, 2005. (Cité page 77.)

J. H. Gillula, H. Huang, M. P. Vitus, et C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets : Autonomous quadrotor aerobatics in theory and practice. Dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1649–1654, 2010. ISBN 1050-4729. (Cité page 63.)

F. Goodarzi, D. Lee, et T. Lee. Geometric nonlinear pid control of a quadrotor uav on se(3). Dans *Control Conference (ECC), 2013 European*, pages 3845–3850, 2013. (Cité page 63.)

A. Gruszka, M. Malisoff, et F. Mazenc. On tracking for the PVTOL model with bounded feedbacks. Dans *American Control Conference (ACC)*, pages 1428–1433, June 2011. (Cité page 47.)

Jung-Ik Ha et Seung-Ki Sul. Sensorless field-orientation control of an induction machine by high-frequency signal injection. *IEEE Transactions on Industry Applications*, 35(1) : 45–51, 1999. (Cité page 118.)

Hongwen He, Jiankun Peng, Rui Xiong, et Hao Fan. An acceleration slip regulation strategy for four-wheel drive electric vehicles based on sliding mode control. *Energies*, 7(6) :3748–3763, 2014. ISSN 1996-1073. (Cité page 77.)

Jun Hu et Bin Wu. New integration algorithms for estimating motor flux over a wide speed range. *IEEE Transactions on Power Electronics*, 13(5) :969–977, Sep 1998. ISSN 0885-8993. (Cité page 119.)

Wang Jian, Xu Xin, Liu Daxue, Sun Zhenping, et Chen Qingyang. Self-learning cruise control using kernel-based least squares policy iteration. *IEEE Transactions on Control Systems Technology*, 22(3) :1078–1087, 2014. ISSN 1063-6536. (Cité page 77.)

T. Jung et D. Polani. Kernelizing lspe. Dans *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 338–345, 2007. (Cité page 31.)

A. Katriniok et D. Abel. Ltv-mpc approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics. Dans *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6828–6833, 2011. ISBN 0191-2216. (Cité page 90.)

Y-R Kim, Seung-Ki Sul, et M-H Park. Speed sensorless vector control of induction motor using extended kalman filter. *IEEE Transactions on Industry Applications*, 30(5) :1225–1233, 1994. (Cité page 118.)

Jens Kober et Jan Peters. Reinforcement learning in robotics : A survey. Dans Marco Wiering et Martijn van Otterlo, éditeurs, *Reinforcement Learning : State-of-the-Art*, pages 579–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. URL http://dx.doi.org/10.1007/978-3-642-27645-3_18. (Cité page 26.)

Juš Kocijan et Roderick Murray-Smith. Nonlinear predictive control with a gaussian process model. Dans Roderick Murray-Smith et Robert Shorten, éditeurs, *Switching and Learning in Feedback Systems : European Summer School on Multi-Agent Control, Maynooth, Ireland*, pages 185–200. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-30560-6. URL http://dx.doi.org/10.1007/978-3-540-30560-6_8. (Cité pages 25, 35 et 38.)

Amol S Kulkarni et MA El-Sharkawi. Speed estimator for induction motor drives using an artificial neural network. Dans *IEEE International Electric Machines and Drives Conference Record*, pages MD2–2. IEEE, 1997. (Cité page 118.)

Michail G. Lagoudakis et Ronald Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4 :1107–1149, 2003. ISSN 1532-4435. (Cité pages 26, 28 et 29.)

T. K. Lau. Learning autonomous drift parking from one demonstration. Dans *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1456–1461, 2011. (Cité page 90.)

I.J. Leontaritis et S.A. Billings. Input-output parametric models for non-linear systems. part i : deterministic non-linear systems. *Int. J. Control*, 41 :303 – 328, 1985a. (Cité page 16.)

I.J. Leontaritis et S.A. Billings. Input-output parametric models for non-linear systems. part ii : stochastic non-linear systems. *Int. J. Control*, 41 :329 – 344, 1985b. (Cité page 16.)

D. Lopez-Araujo, A. Zavala-Rio, I. Fantoni, S. Salazar, et R. Lozano. Global stabilization of the PVTOL aircraft with lateral force coupling and bounded inputs. *International Journal of Control*, 83(7) :1427–1441, 07 2010. (Cité page 47.)

M.A. Luersen, R. Le Riche, et F. Guyon. A constrained, globalized, and bounded nelder–mead method for engineering optimization. *Structural and Multidisciplinary Optimization*, 27(1) :43–54, 2004. ISSN 1615-1488. URL http://dx.doi.org/10.1007/s00158-003-0320-9. (Cité page 13.)

S. Lupashin, A. Schollig, M. Sherback, et R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. Dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1642–1648, 2010. ISBN 1050-4729. (Cité page 63.)

S. Riachy M. Fliess, C. Join. Rien de plus utile qu'une bonne théorie : la commande sans modèle. Dans *JD-JN MACS*, Marseille, 2011. http://hal.archives-ouvertes.fr/hal-00581109/fr/. (Cité pages 34 et 55.)

Cutler Mark et How Jonathan. Actuator constrained trajectory generation and control for variable-pitch quadrotors. Dans *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2012. URL http://dx.doi.org/10.2514/6.2012-4777. doi :10.2514/6.2012-4777. (Cité page 63.)

P. Martin, S. Devasia, et B. Paden. A different look at output tracking : control of a VTOL aircraft. Dans *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 3, pages 2376–2381, Dec 1994. (Cité page 47.)

Tai Meihua et M. Tomizuka. Robust longitudinal velocity tracking of vehicles using traction and brake control. Dans *Advanced Motion Control, 2000. Proceedings. 6th International Workshop on*, pages 305–310, 2000. (Cité page 77.)

D. Mellinger et V. Kumar. Minimum snap trajectory generation and control for quadrotors. Dans *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525, 2011. ISBN 1050-4729. (Cité page 63.)

L. Menhour, B. d'Andréa Novel, C. Boussard, M. Fliess, et H. Mounier. Coupled nonlinear vehicle control : Flatness-based setting with algebraic estimation techniques. *Control Engineering Practice*, 22 :135 – 146, january 2014. ISSN 0967-0661. URL http://www.sciencedirect.com/science/article/pii/S0967066113001846. (Cité page 77.)

Douglas William Moore. *Simplicial mesh generation with applications*. Thèse de doctorat, Cornell University, Ithaca, NY, USA, August 1992. (Cité page 10.)

Rémi Munos et Andrew Moore. Barycentric interpolators for continuous space & time reinforcement learning. Dans *Advances in Neural Information Processing Systems*. MIT Press, 1998. (Cité pages 8 et 9.)

J. A. Nelder et R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313, 1965. URL `http://comjnl.oxfordjournals.org/content/7/4/308.abstract`. (Cité page 13.)

H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Nonlinear control of the pvtol aircraft by numerical inversion of its behavioral model. Dans *23th IEEE International Conference on Control Application*, Antibe, France, October 2014. (Cité pages xx et 40.)

H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Commande non linéaire en temps discret utilisant un modèle de référence d'un véhicule automobile. Dans *Sixième Journées Doctorales, Journées Nationales MACS*, 2015a. (Cité page xx.)

H-P. Nguyen, J. De Miras, S. Bonnet, et A. Charara. Nonlinear control for the vehicle by numerical inversion of its behavioral model. Dans *14th European Control Conference*, Linz, Austria, July 2015b. (Cité pages xx et 84.)

H-P. Nguyen, J. De Miras, A. Charara, M. Eltabach, et S. Bonnet. How to estimate the rotor speed of the asynchronous machine using a numerical approach. *Submitted to Mechanical Systems and Signal Processing (MSSP)*, 2016. (Cité page xx.)

H-P. Nguyen, C. Lajnef, J. De Miras, S. Bonnet, A. Charara, et M. Eltabach. Numerical approach for estimation of the rotor speed of the asynchronous machine. Dans *8th Surveillance*, Roanne, France, October 2015c. (Cité page xx.)

D. Nguyen-Tuong, M. Seeger, et J. Peters. Local gaussian process regression for real time online model learning and control. Dans *Advances in Neural Information Processing Systems 22 (NIPS 2008), Cambridge, MA : MIT Press*, 2009. URL `http://www-clmc.usc.edu/publications/N/nguyen_NIPS2008.pdf`. (Cité page 25.)

Duy Nguyen-Tuong et Jan Peters. Incremental online sparsification for model learning in real-time robot control. *Neurocomput.*, 74(11) :1859–1867, 2011. ISSN 0925-2312. (Cité page 25.)

Bjorn Olofsson, Kristoffer Lundahl, Karl Berntorp, et Lars Nielsen. An investigation of optimal vehicle maneuvers for different road conditions. *IFAC Proceedings Volumes*, 46(21) :66–71, 2013. ISSN 1474-6670. URL `http://dx.doi.org/10.3182/20130904-4-jp-2042.00007`. (Cité page 90.)

Hans B. Pacejka. *Tyre and Vehicle Dynamics*. Butterworth-Heinemann, Oxford, second edition édition, 2006. ISBN 978-0-7506-6918-4. (Cité page 77.)

RS Pena et GM Asher. Parameter sensitivity studies for induction motor parameter identification using extended kalman filters. Dans *Fifth European Conference on Power Electronics and Applications*, pages 306–311. IET, 1993. (Cité page 118.)

Fang-Zheng Peng et T. Fukao. Robust speed identification for speed-sensorless vector control of induction motors. *IEEE Transactions on Industry Applications*, 30(5) :1234–1240, Sep 1994. ISSN 0093-9994. (Cité page 125.)

Hung Pham, K. Hedrick, et M. Tomizuka. Combined lateral and longitudinal control of vehicles for ivhs. Dans *American Control Conference, 1994*, volume 2, pages 1205–1206 vol.2, 1994. (Cité page 77.)

M. J. D. Powell. Algorithms for approximation. Chapitre Radial Basis Functions for Multivariable Interpolation : A Review, pages 143–167. Clarendon Press, New York, NY, USA, 1987. ISBN 0-19-853612-7. URL `http://dl.acm.org/citation.cfm?id=48424.48433`. (Cité page 21.)

Carl Edward Rasmussen et Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X. (Cité pages 24 et 25.)

Samer Riachy, Michel Fliess, Cédric Join, et Jean-Pierre Barbot. Vers une simplification de la commande non linéaire : l'exemple d'un avion à décollage vertical. Dans *Sixième Conférence Internationale Francophone d'Automatique, CIFA 2010*, page CDROM, Nancy, France, Juin 2010. (Cité page 47.)

LAS Ribeiro, MW Degner, F Briz, et RD Lorenz. Comparison of carrier signal voltage and current injection for the estimation of flux angle or rotor position. Dans *Industry Applications Conference, Thirty-Third IAS Annual Meeting*, volume 1, pages 452–459. IEEE, 1998. (Cité page 118.)

Chaichaowarat Ronnapee et Wannasuphoprasit Witaya. Linear quadratic optimal regulator for steady state drifting of rear wheel drive vehicle. *Journal of Robotics and Mechatronics*, 27(3) :225–234, 2015. ISSN 1883-8049 0915-3942. URL `http://dx.doi.org/10.20965/jrm.2015.p0225`. (Cité page 90.)

G. A. Rummery et M. Niranjan. On-line q-learning using connectionist systems. Rapport technique, Cambridge University Engeneering Department, 1994. URL `ftp://svr-ftp.eng.cam.ac.uk/reports/rummery_tr166.ps.Z`. (Cité page 31.)

Guillaume Sanahuja. *Commande et localisation embarquée d'un drone aérien en utilisant la vision*. Thèse de doctorat, Université de Technologie de Compiègne, France, 2010. (Cité page 54.)

Colin Schauder. Adaptive speed identification for vector control of induction motors without rotational transducers. *IEEE Transactions on Industry applications*, 28(5) :1054–1061, 1992. (Cité pages 118 et 119.)

Bernhard Schölkopf, Ralf Herbrich, et Alex J. Smola. A generalized representer theorem. Dans David Helmbold et Bob Williamson, éditeurs, *Computational Learning Theory : 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001 Amsterdam, The Netherlands, July 16–19, 2001 Proceedings*, pages 416–426. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44581-4. URL http://dx.doi.org/10.1007/3-540-44581-1_27. (Cité page 23.)

Manfred Schroedl. Sensorless control of ac machines at low speed and standstill based on the inform method. Dans *Conference Record of the Industry Applications Conference, Thirty-First IAS Annual Meeting*, volume 1, pages 270–277. IEEE, 1996. (Cité page 118.)

G. Séguier et F. Notelet. *Électrotechnique industrielle*. Tec & Doc Lavoisier, 2006. ISBN 9782743007911. (Cité page 111.)

Satinder P. Singh et Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1) :123–158, 1996. ISSN 1573-0565. URL http://dx.doi.org/10.1007/BF00114726. (Cité page 31.)

H. Sira-Ramirez et M. Fliess. Regulation of non-minimum phase outputs in a PVTOL aircraft. Dans *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 4, pages 4222–4227, Dec 1998. (Cité page 48.)

Sara Spedicato, Giuseppe Notarstefano, Heinrich H. Bulthoff, et Antonio Franchi. Aggressive maneuver regulation of a quadrotor uav. Dans Masayuki Inaba et Peter Corke, éditeurs, *Robotics Research : The 16th International Symposium ISRR*, pages 95–112. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28872-7. URL http://dx.doi.org/10.1007/978-3-319-28872-7_6. (Cité page 63.)

Richard Sutton. *Reinforcement learning : An introduction*. MIT Press, Cambridge, Mass, 1998. ISBN 0262193981. (Cité page 31.)

Gilles Tagne, Reine Talj, et Ali Charara. Higher-Order Sliding Mode Control for Lateral Dynamics of Autonomous Vehicles, with Experimental Validation. Dans *IEEE Intelligent Vehicles Symposium (IV)*, pages 678–683, Gold Coast, Australia, Juin 2013. (Cité page 77.)

Davide Tavernini, Matteo Massaro, Efstathios Velenis, Diomidis I. Katzourakis, et Roberto Lot. Minimum time cornering : the effect of road surface and car transmission layout. *Vehicle System Dynamics*, 51(10) :1533–1547, 2013. ISSN 0042-3114. URL http://dx.doi.org/10.1080/00423114.2013.813557. (Cité page 90.)

A Testa, D Triolo, A Consoli, G Scarcella, et G Scelba. Sensorless airgap flux position estimation by injection of orthogonal stationary signals. Dans *Power Electronics Specialists Conference, 2005. PESC'05. IEEE 36th*, pages 1567–1573. IEEE, 2005. (Cité page 118.)

T. Tomix, M. Maier, et S. Haddadin. Learning quadrotor maneuvers from optimal control and generalizing in real-time. Dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1747–1754, 2014. ISBN 1050-4729. (Cité page 63.)

S. Van Vaerenbergh, I. Santamaria, Liu Weifeng, et J. C. Principe. Fixed-budget kernel recursive least-squares. Dans *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 1882–1885, 2010. ISBN 1520-6149. (Cité page 24.)

S. Van Vaerenbergh, J. Via, et I. Santamaria. A sliding-window kernel rls algorithm and its application to nonlinear channel identification. Dans *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V, 2006. ISBN 1520-6149. (Cité page 24.)

E. Velenis, E. Frazzoli, et P. Tsiotras. On steady-state cornering equilibria for wheeled vehicles with drift. Dans *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 3545–3550, 2009. ISBN 0191-2216. (Cité pages 90 et 91.)

E. Velenis et P. Tsiotras. Minimum time vs maximum exit velocity path optimization during cornering. Dans *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, volume 1, pages 355–360, 2005. ISBN 2163-5137. (Cité page 90.)

Efstathios Velenis, Emilio Frazzoli, et Panagiotis Tsiotras. Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions. *IJVAS*, 8(2/3/4) :217, 2010. ISSN 1471-0226 1741-5306. URL http://dx.doi.org/10.1504/ijvas.2010.035797. (Cité pages 90 et 91.)

Efstathios Velenis, Diomidis Katzourakis, Emilio Frazzoli, Panagiotis Tsiotras, et Riender Happee. Steady-state drifting stabilization of rwd vehicles. *Control Engineering Practice*, 19(11) :1363–1376, 2011. ISSN 0967-0661. URL http://www.sciencedirect.com/science/article/pii/S0967066111001614. (Cité page 90.)

Emmanuel Viennet. Réseaux à fonctions de base radiales. Dans Younès Bennani, éditeur, *Apprentissage connexionniste*, I2C Hermès, page 105. Lavoisier, 2006. URL https://hal.archives-ouvertes.fr/hal-00085092. 18 pages. (Cité page 21.)

Christoph Voser, Rami Y. Hindiyeh, et J. Christian Gerdes. Analysis and control of high sideslip manoeuvres. *Vehicle System Dynamics*, 48(sup1) :317–336, 2010. ISSN 0042-3114. URL http://dx.doi.org/10.1080/00423111003746140. (Cité page 90.)

R. Wood, B. Cazzolato, et D. Halim. A global non-linear control design for a PVTOL vehicle with aerodynamics. Dans *44th IEEE Conference on Decision and Control, European Control Conference. CDC-ECC '05*, pages 7478–7483, Dec 2005. (Cité page 47.)

Xu Xin, Hu Dewen, et Lu Xicheng. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4) :973–992, 2007. ISSN 1045-9227. URL http://www.ncbi.nlm.nih.gov/pubmed/17668655. Xu, Xin Hu, Dewen Lu, Xicheng eng Research Support, Non-U.S. Gov't 2007/08/03 09 :00 IEEE Trans Neural Netw. 2007 Jul ;18(4) :973-92. (Cité page 31.)

H. Yu, P. D. Reiner, T. Xie, T. Bartczak, et B. M. Wilamowski. An incremental design of radial basis function networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10) :1793–1803, Oct 2014. ISSN 2162-237X. (Cité page 34.)

Titre Développement d'une commande à modèle partiel appris.
Analyse théorique et Étude pratique

Résumé En théorie de la commande, un modèle du système est généralement utilisé pour construire la loi de commande et assurer ses performances. Les équations mathématiques qui représentent le système à contrôler sont utilisées pour assurer que le contrôleur associé va stabiliser la boucle fermée. Mais, en pratique, le système réel s'écarte du comportement théorique modélisé. Des non-linéarités ou des dynamiques rapides peuvent être négligées, les paramètres sont parfois difficiles à estimer, des perturbations non maitrisables restent non modélisées.
L'approche proposée dans ce travail repose en partie sur la connaissance du système à piloter par l'utilisation d'un modèle analytique mais aussi sur l'utilisation de données expérimentales hors ligne ou en ligne. A chaque pas de temps la valeur de la commande qui amène au mieux le système vers un objectif choisi a priori, est le résultat d'un algorithme qui minimise une fonction de coût ou maximise une récompense.
Au centre de la technique développée, il y a l'utilisation d'un modèle numérique de comportement du système qui se présente sous la forme d'une fonction de prédiction tabulée ayant en entrée un n-uplet de l'espace joint entrées/état ou entrées/sorties du système. Cette base de connaissance permet l'extraction d'une sous-partie de l'ensemble des possibilités des valeurs prédites à partir d'une sous-partie du vecteur d'entrée de la table. Par exemple, pour une valeur de l'état, on pourra obtenir toutes les possibilités d'états futurs à un pas de temps, fonction des valeurs applicables de commande. Basé sur des travaux antérieurs ayant montré la viabilité du concept en entrées/état, de nouveaux développements ont été proposés. Le modèle de prédiction est initialisé en utilisant au mieux la connaissance a priori du système. Il est ensuite amélioré par un algorithme d'apprentissage simple basé sur l'erreur entre données mesurées et données prédites. Deux approches sont utilisées : la première est basée sur le modèle d'état (comme dans les travaux antérieurs mais appliquée à des systèmes plus complexes), la deuxième est basée sur un modèle entrée-sortie. La valeur de commande qui permet de rapprocher au mieux la sortie prédite dans l'ensemble des possibilités atteignables de la sortie ou de l'état désiré, est trouvée par un algorithme d'optimisation.
Afin de valider les différents éléments proposés, cette commande a été mise en œuvre sur différentes applications. Une expérimentation réelle sur un quadricoptère et des essais réels de suivi de trajectoire sur un véhicule électrique du laboratoire montrent sa capacité et son efficacité sur des systèmes complexes et rapides. D'autres résultats en simulation permettent d'élargir l'étude de ses performances. Dans le cadre d'un projet partenarial, l'algorithme a également montré sa capacité à servir d'estimateur d'état dans la reconstruction de la vitesse mécanique d'une machine asynchrone à partir des signaux électriques. Pour cela, la vitesse mécanique a été considérée comme l'entrée du système.

Mots-clés Commande non linéaire, commande par apprentissage, inversion numérique de modèle, approximation de fonction, commande en temps discret