UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté de Technologies

Département d'Electronique

THESE DE DOCTORAT en SCIENCES

Spécialité : Contrôle

IMPLEMENTATION DE CONTROLEURS LINEAIRE ET NON-LINEAIRE POUR LE PILOTAGE D'UN ROBOT MOBILE AUTONOME

Présentée par

Boualem KAZED

Devant le jury composé de :

A. NAAMANE	Professeur, U. de Blida	Président
M. S. BOUCHERIT	Professeur, E. N. P. Alger	Examinateur
M. OULD ZEMIRLI	Professeur, U. de Médéa	Examinateur
A. GUESSOUM	Professeur, U. de Blida	Directeur de thèse

Blida, Décembre 2024

Résumé

La commande autonome des robots mobiles a été et reste le sujet d'un grand nombre de recherches durant les dernières décennies. L'intérêt grandissant porté à ce sujet vient du fait de la diversité des applications potentielles qui touchent aussi bien les domaines de services tels que les transports que ceux relatifs à la sécurité et la défense. Parmi les problèmes liés à la navigation de ces robots, la commande des mouvements constitue l'une des premières préoccupations des chercheurs dans ce domaine. Plusieurs approches ont été proposées avec plus ou moins de complexité quant à leur implémentation sur une plateforme mobile. Le but principal de ce travail consiste à élaborer des lois de commandes linéaire et non-linéaire à même de conduire un robot mobile, de type différentiel, d'une configuration initiale vers une position et orientation finales. La validation de ces commandes sera confirmée par leur implémentation matérielle sur un prototype de robot comprenant un minimum de composants mécaniques et électroniques mais suffisants pour réaliser les mouvements adéquats permettant à ce dernier d'atteindre l'objectif qui lui a été assigné. Tel que c'est le cas pour un système mécanique en mouvements, l'efficacité de la commande adoptée doit nécessairement tenir compte des contraintes dynamiques de ce système. Pour ce qui concerne le cas présent, cela a été pris en charge en considérant que la commande globale est constituée de deux parties distinctes ; Une commande cinématique prenant en compte les relations liant les vitesses linéaire et angulaire du robot à celles des vitesses de rotation de ses deux roues. Ces dernières étant asservies par une paire de commandes PID, elles prennent donc en considération la dynamique du robot puisqu'elles sont responsables de faire tourner les deux roues du robot en tenant compte de l'inertie de ces dernières. Le caractère non-linéaire et multi-variables du modèle cinématique nous a conduit à proposer une commande non-linéaire appropriée. Celle que nous avons retenu est basée sur la théorie de Lyapunov, à partir de laquelle deux lois de commandes seront déduites pour, respectivement, faire avancer et orienter le robot d'une manière à ce qu'il puisse atteindre une cible préalablement choisie. Afin d'assouplir la mise en œuvre de cette stratégie de commande, nous l'avons répartie en deux parties complémentaires ; La première dénommée, commande de haut niveau se charge de calculer les consignes des deux vitesses de rotation des moteurs gauche et droite du robot, celle-ci est basée sur le modèle cinématique directe

du robot. Cette dernière sera implémentée sous forme d'un programme Matlab/Simulink s'exécutant sur un PC fixe, connecté au robot par l'intermédiaire d'un port série sur lequel les vitesses de rotations désirées et mesurées sont échangées avec le contrôleur bas niveau. Celui-ci est implémenté sur une carte embarquée à bord du robot incluant un processeur 16 bits, suffisamment puissant pour exécuter les deux commandes PID prenant en charge l'asservissement, en temps réel, des vitesses des moteurs responsables des mouvements du robot. L'architecture ainsi adoptée nous offre une très grande souplesse non seulement pour le réglage des paramètres de la commande mais aussi d'avoir accès à tous les états internes du système.

Mots clés : Robots mobiles, commande non-linéaire, théorie de Lyapunov, Suivis de trajectoires, Systèmes temps réels.

Abstract

Designing self-controlled mobile robots has been and still the subject for many research works during the last decades. This growing interest comes from the benefits these robots can bring in a large number of applications such as in the transport domain, security or defence. Among the main problems faced in robot navigation, motion control is one if the most important concern for many researchers in this area. Several approaches have been proposed with different level of complexity related to their hardware implementation. The main goal of this work consists in designing some linear and non-linear control laws that will be used to drive a two-wheeled mobile robot to a desired posture starting from an initial known configuration. To validate these controllers their hardware implementation will be tested on a mobile robot prototype, constructed out of a minimal number of mechanical and electronic parts but enough to make this robot reach the goal assigned to it. As is the case for any moving mechanical system, the success of any applied controller is closely dependant on its ability to compensate for this system dynamics. In the present work this has been taken care of by considering that the overall control system is made of two distinct and complementary controllers; The first or high-level controller, based on the kinematic model, will compute the robot linear and heading speeds using the left and right rotational speeds of the robot wheels. To make sure that the required speeds have the required values, two low-level PID controllers are included in a pair of local closed loop subsystems to control the speed of each of these wheels. The parameters of these controllers have been tuned taking into account the wheels inertia. The set of these controllers can therefore be considered as the dynamic part of the overall control system. To make the robot execute the required movements the high-level controller has been designed based on the non-linear, multi-variable kinematic model. This has been achieved using the Lyapunov theory to derive two control laws for the robot longitudinal and angular speeds respectively. These will allow the robot to reach or follow the required assignments depending on the type of task submitted by the end user. This high-level controller has been implemented as a Matlab/Simulink program running on a stationary PC system, connected to the robot by means of a serial port on which the required and measured rotational speeds are being exchanged with the low-level controllers. These controllers, together with all the required input/output drivers will be running on an embedded

system equipped with a 16-bit processor having sufficient resources to have a real-time execution of the two low-level PID speed controllers making the robot move in the desired way. As will be shown later, this architecture provides a very flexible solution not only to tune the main controller parameters, but also to get access and record all the internal system states.

Keywords: Mobile robots, Non-linear control, Lyapunov theory, Trajectory tracking, Real time systems.

ملخص

لقد كان التحكم الذاتي في الروبوتات المتنقلة ولا يزال محل اهتمام عدد كبير من الأبحاث على مدار عقود .ولعل أهم أسباب هذا الاهتمام المتزايد هو تنوع تطبيقاته في مجالات مختلفة بدءا من قطاع الخادمات كالنقل وصولا إلى تلك المتعلقة بالأمن والدفاع .ومن بين المشاكل المتعلقة بملاحة هذه الروبوتات، يشكل التحكم في حركاتها أحد الاهتمامات الأساسية للباحثين في هذا المجال .تم اقتراح عدة طرق بدرجات متفاوتة من التعقيد فيما يتعلق بتنفيذها على منصة متنقلة .الهدف الرئيسي من هذا العمل هو تطوير قوانين التحكم الخطية وغير الخطية القادرة على قيادة الروبوت المتحرك، من النوع التفاضلي، وهذا من التشكيلة الأولية إلى الوضع والتوجه النهائيين .يتم إقرارصلاحية و فعالية هذه القوانين من خلال اختبارها على نموذج أولي للروبوت يشتمل على الحد الأدنى من المكونات الميكانيكية والإلكترونية الكافية لتنفيذ الحركات المناسبة التي تسمح له بتحقيق الهدف المسطر له .وكما هو الحال بالنسبة للنظام الميكانيكي المتحرك، فإن فعالية التحكم المعتمد يجب أن تأخذ بعين الاعتبار القيود الديناميكية لهذا النظام .فيما يتعلق بعملنا الحالي، قد تم الاهتمام بهذا الأمر من خلال اعتبار أن التحكم العام يتكون من جزأين متميزين؛ تحكم حركي يأخذ بعين الاعتبار العلاقات التي تربط السرعات الخطية والزاوية للروبوت بسرعات دوران عجلتيه. يتم التحكم في هذه الأخيرة بواسطة زوج من أوامر ب. أ. د. التي تأخذ في الاعتبار ديناميكية الروبوت لكونها مسؤولة عن تدوير عجلتيه مع مراعاة القصور الذاتي لهذه الأخيرة. إن الطبيعة الغير خطية والمتعددة المتغيرات للنموذج الحركي دفعتنا إلى اقتراح تحكم غير خطي مناسب .يعتمد القانون الذي اخترناه على نظرية ليابونوف، والتي يتم من خلالها استخلاص قانونين للتحكم على التوالي، لدفع الروبوت وتوجيهه بطريقة تمكنه من الوصول إلى الهدف الذي تم اختياره مسبقًا. ولجعل تنفيذ استراتيجية التحكم هذه أكثر مرونة، قمنا بتقسيمها إلى جزأين متكاملين؛ الأول يسمى التحكم عالي المستوى، وهو المسؤول عن حساب التعليمات الخاصة بسرعتي الدوران للمحركين الأيسر والأيمن للروبوت، وذلك بالاعتماد على النموذج ،الحركي المباشر للروبوت. سيتم تنفيذ هذا لأخيرفي شكل برنامج (متلاب سيمولينك) الذي يعمل على جهاز كمبيوتر ثابت متصل بالروبوت عبر منفذ تسلسلي يتم من خلاله تبادل سرعات الدوران المطلوبة والمقاسة مع وحدة التحكم ذات المستوى المنخفض .يتم تنفيذ ذلك على بطاقة مدمجة على متن الروبوت بما في ذلك معالج دقيق 16بيت، قوي بما يكفي لتنفيذ أمري ب. أ. د. اللذين يدعمان التحكم في الوقت الفعلي لسرعات المحركات المسؤولة عن حركات الروبوت .وبالتالي فإن البنية المعتمدة توفر لنا مرونة كبيرة ليس فقط لضبط معلمات التحكم ولكن أيضا للوصول إلى جميع الحالات الداخلية للنظام

الكلمات المفتاحية : الروبوتات المتنقلة، التحكم غير الخطي، نظرية ليابونوف، تتبع المسار،أنظمة الزمن الحقيقي

Remerciements

Je tiens à exprimer ma plus vive reconnaissance au Professeur Abderrezak GUESSOUM, directeur de thèse, pour ses qualités humaines et scientifiques, ses encouragements et surtout sa patience. J'ai aussi beaucoup apprécié sa confiance lors de l'élaboration de ce travail.

Mes remerciements s'adressent particulièrement aux honorables membres du jury ;

Mr. Abderrahmane NAAMANE, Professeur à l'université de Blida 1.

Mr. Mohamed Seghir BOUCHERIT, Professeur à l'Ecole Nationale Polytechnique d'Alger.

Mr. Mohamed OULD ZEMIRLI, Professeur à l'université de Médéa.

Qui ont gentiment accepté d'examiner ce travail et auxquels je tiens à exprimer ma plus profonde gratitude.

Je ne terminerai pas cette rubrique sans oublier de remercier tous les collègues enseignants des départements d'électronique et automatique, ainsi qu'à l'ensemble des personnels administratifs et techniques de l'université de Blida.

Table des Matières

Liste des figures

Liste des Tableaux

Liste des Abréviations

MCC : Moteur à Courant Continu

PID : Proportional Integral Controller

PWM : pulse Width Modulation

QEI : Quadrature Encoder Interface

UART : Universal Asynchronous Receive Transmit

$\boldsymbol{\omega_D}$: Vitesse angulaire de la roue droite du robot (rad/sec)

$\boldsymbol{\omega_G}$: Vitesse angulaire de la roue gauche du robot (rad/sec)

$\boldsymbol{v_D}$: Vitesse linéaire de la roue droite du robot (mètre/sec)

$\boldsymbol{v_G}$: Vitesse linéaire de la roue gauche du robot (mètre/sec)

$\boldsymbol{\Omega}$: Vitesse angulaire du robot autour de l'axe vertical (rad/sec)

$\boldsymbol{v}$: Vitesse de déplacement linéaire du robot (mètre/sec)

CRI : Centre de Rotation Instantané

PC : Personal Computer

RRT : Rapidly exploring Random Tree

PRM : Probabilistic Road Map

Introduction

L'introduction des systèmes autonomes ou automatiques dans une grande partie de l'activité humaine est une réalité depuis déjà très longtemps. L'utilisation de systèmes robotisés dans les domaines de la production industrielle constitue un exemple typique, qui ne cesse de prendre de l'ampleur. Les robots mobiles peuvent être complétement ou partiellement opérés à distance dans un certain nombre d'applications telles que l'accès à des endroits hostiles ou difficilement atteignables pour des humains. L'autonomie d'un robot mobile peut être mesurée par sa capacité à prendre les décisions adéquates sans aucune intervention humaine. Pour y arriver ce dernier doit être équipé d'un minimum d'éléments de perception, lui permettant de fournir les signaux de commandes nécessaires, à ses actionneurs, afin d'atteindre ou suivre les consignes dictées par les besoins des taches qu'il doit exécuter.

Parmi les configurations mécaniques des robots mobiles existants, celles utilisant une structure différentielle à deux roues est très largement utilisée, notamment, dans les applications d'intérieur. Indépendamment de cette structure, le problème commun à tous les robots mobiles reste la commande de mouvements. Le sujet principal de ce travail entre dans ce cadre en proposant un système de commande complet, permettant au robot d'exécuter les mouvements nécessaires le conduisant à une destination prédéfinie, ou bien suivre une trajectoire constituée de points de passages préalablement connus. La première partie de ce travail sera consacrée au problème similaire à celui de l'exécution automatique des manœuvres de stationnement. Ce dernier consiste à déterminer deux lois de commande agissantes, respectivement, sur les déplacements linéaire et angulaire du robot de sorte à le faire atteindre une posture, définie par les coordonnées cartésiennes et orientation finales. La conséquence de ces deux lois de commandes fera l'objet de la deuxième partie de ce travail, qui sera consacrée au problème de suivi de trajectoires. Une étude bibliographique montre que le nombre de travaux de recherches relatifs au suivis trajectoires [5] – [11] est bien supérieur à celui dédié à la commande de posture. Cet état de fait est probablement dû à la difficulté de ce dernier comparativement à celui traitant de la poursuite de trajectoires. Cela

peut aussi être expliqué par la diversité des besoins exprimés dans le monde réel tels que celui de l'agriculture, des transports, du nettoyage, de la sécurité ainsi que bien d'autres.

La stratégie adoptée pour résoudre le premier problème est basée sur une approche classique, qui consiste à construire un modèle mathématique régissant le fonctionnement du système à contrôler, pour ensuite élaborer des lois de commandes appropriées. L'essentiel de ce problème a été traité dans [1], sur lequel nous allons revenir pour montrer qu'une linéarisation du modèle utilisé peut, dans certains cas, donner de très bons résultats, notamment lorsque nous allons l'utiliser pour faire suivre, le robot utilisé, une trajectoire construite à partir d'un nombre limité de positions successives. La maitrise des mouvements d'un robot mobile passe nécessairement par une commande précise des vitesses des deux roues responsables de ces mouvements. Le problème est d'autant plus compliqué lorsque le robot doit constamment changer de direction avant d'atteindre la cible finale. Afin de résoudre ce problème nous allons dans un premier temps, agir au niveau local pour contrôler uniquement les vitesses de rotation des deux moteurs, couplés aux deux roues du robot. Pour décider des vitesses avec lesquelles ces deux roues doivent être mises en rotation, un autre contrôleur se charge de les calculer en fonction des vitesses longitudinale et angulaire que le robot doit subir pour réaliser les mouvements désirés. Pour mettre en œuvre ces contrôleurs, nous les avons organisées sous forme de deux boucles fermées, dont la première, elle-même constituée deux boucles internes, aura pour rôle l'asservissement des vitesses gauche et droite des roues du robot. La boucle externe, quant à elle, se chargera de fournir les consignes de vitesses nécessaires permettant au robot d'effectuer ses déplacements. Pour ce qui concerne les types de commandes utilisées, nous allons exploiter la théorie de Lyapunov pour montrer la manière avec laquelle nous avons élaborer les deux lois de commandes constituant la boucle externe, les deux boucles internes sont quant à elles basée sur une paire de commandes classiques de type PID.

Le modèle cinématique du robot nous permet d'obtenir les relations liant les vitesses linéaire et angulaire de ce dernier avec celles des vitesses de rotations des roues gauche et droite. La non-linéarité de ce modèle nous a conduit à proposer des lois de commandes non-linéaires à même de résoudre ce problème. La stratégie de commande adoptée dans ce travail sera basée sur le même modèle d'état utilisé dans [12] – [17], à partir duquel nous

avons établi une fonction de Lyapunov appropriée et surtout une nouvelle loi de commande pour l'asservissement de la vitesse avec laquelle le robot doit s'orienter. Avec le choix de cette commande il a été montré que le modèle proposé est asymptotiquement stable, ce qui n'a pas été clairement démontré dans la littérature. Plusieurs autres types d'approches ayant les mêmes objectifs ont été proposées [18] – [20]. D'autres auteurs [21] ont proposé une commande basée sur l'approche de Lyapunov utilisant les erreurs des coordonnées cartésiennes. Le même problème a été traité dans [22] avec une autre démarche proposant deux fonctions de Lyapunov différentes en fonction de la distance séparant le robot de la position cible. D'autres méthodes telles que celle basée sur un retour linéarisant [24] a été utilisée pour obtenir une commande de type PD visant à guider le robot vers une position et orientation désirées. Une commande linéaire avec l'ajout d'un point intermédiaire pour le réglage de l'orientation du robot a été proposée dans [25]. Une forme linéarisée du modèle cinématique du robot unicycle a été suggérée dans [26] pour obtenir trois commandes PID visant à résoudre le problème du suivi de trajectoire. Une technique de Commande à Temps Fini, basée sur une structure bilinéaire a été proposée dans [27], celle-ci s'articule autour de deux contrôleurs fonctionnant en alternance, pour stabiliser le robot sur la posture cible. Ce même problème a été traité dans [28] avec une stratégie de commande robuste, basée sur le principe de passivité. Plusieurs autres approches [23], [29] ont été proposées ont été proposées dans le même contexte. Des techniques basées sur l'Intelligence Artificielle (IA), utilisant l'Apprentissage Renforcé (AR)) ont été suggérées pour faire atteindre le robot une position et orientation prédéfinies [30], [31].

Le présent travail se distingue par le caractère modulaire de l'implémentation matérielle de la commande proposée. En effet, la partie prenant en charge la boucle externe, introduite précédemment, est exécutée séparément sur un PC fixe pour générer les deux consignes de vitesses pour chacune des deux roues dont est muni le robot mobile, conçu spécialement pour ce projet. Cette façon de procéder nous permet non seulement de réduire considérablement le temps de réglage des paramètres du contrôleur, mais surtout de pouvoir suivre, en temps réel, les variations de toutes les variables d'états y compris les signaux d'entrées/sorties ainsi que les signaux de contrôle calculés dans la boucle de commande principale.

L'organisation du reste de cette thèse se résume comme suit : Le chapitre 1 sera consacré à la description des modèles mathématiques utilisés à partir desquels les commandes proposées ont été obtenues. Le contenu du chapitre 2 sera entièrement dédié à la partie principale de ce sujet à savoir celle relative à l'élaboration des différentes commandes utilisées dans respectivement, les boucles interne et externe, citées précédemment. L'efficacité de ces commandes sera démontrée à travers plusieurs exemples d'applications, avec pour chaque cas, une comparaison des résultats des simulations exécutées sur le PC uniquement, avec ceux obtenus à partir des mesures acquises pendant les mouvements du robot réel soumis aux mêmes conditions que celles proposées pour les tests en simulation. Avant d'entamer la partie concernant le suivi de trajectoires, nous avons réservé le chapitre 3 pour une revue de certaines méthodes classiques, utilisées pour la planification de trajectoires dans un environnement comprenant des obstacles, pouvant altérer les déplacements du robot mobile. Après avoir planifié sa trajectoire, ce robot doit être capable de déterminer les mouvements à effectuer pour rester le plus proche possible de cette trajectoire. Le chapitre 4 sera donc consacré à ce sujet, avec quelques exemples issus de la vie réelle, montrant l'efficacité de l'approche proposée pour résoudre ce problème.

Chapitre 1

Modélisation du robot mobile

Afin d'obtenir un modèle représentatif d'un système donné, il est en général nécessaire de disposer de données entrées/sorties expérimentales, mesurées à partir du système physique. L'exploitation de ces données permet d'aboutir à deux types de modèles mathématiques ; Dans le cas d'un modèle de connaissance il est possible d'obtenir les valeurs des paramètres physiques du modèle, ceci en se basant sur une minimisation de la différence entre la réponse du modèle et celle mesurée expérimentalement. L'autre type de modélisation possible, grâce à ces données, consiste à estimer les paramètres d'un modèle de type ''boite noire'' ou modèle de représentation. Le but final de ce travail étant de contrôler les mouvements d'un robot mobile, la procédure adoptée sera, dans un premier, de trouver un model adéquat pour les moteurs responsables de ces mouvements, suivie par la suite par l'élaboration d'un modèle cinématique décrivant les relations entre les vitesses des deux moteurs avec celles des déplacements linéaires et angulaires du robot mobile.

1.1 Réalisation matérielle

Le but de ce paragraphe sera donner le principe de fonctionnement de la partie matérielle de ce travail permettant de réaliser l'acquisition de données citée précédemment. Le schéma bloc de la figure 1.1 représente le principe de fonctionnement de la partie électronique embarquée sur le robot mobile. Celle-ci se compose essentiellement d'un microcontrôleur de type DSPic33fj64mc802 [Annexe 1], comprenant tous les modules nécessaires à la commande d'un moteur de ce type. La zone du milieu de cette figure, dénommée *Firmware*, représente le programme de gestion du microcontrôleur, celle-ci a pour fonction d'exécuter toutes les opérations arithmétiques et logiques et de faire appel aux différents modules intégrés à ce microcontrôleur.

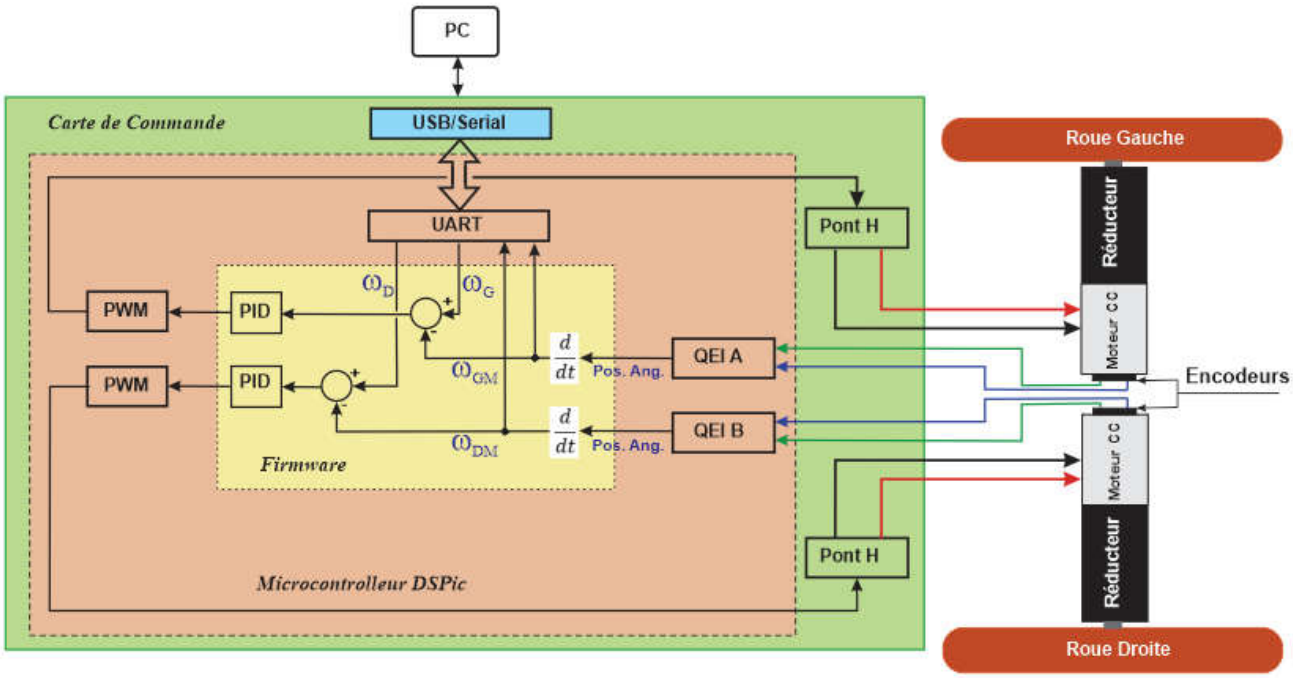
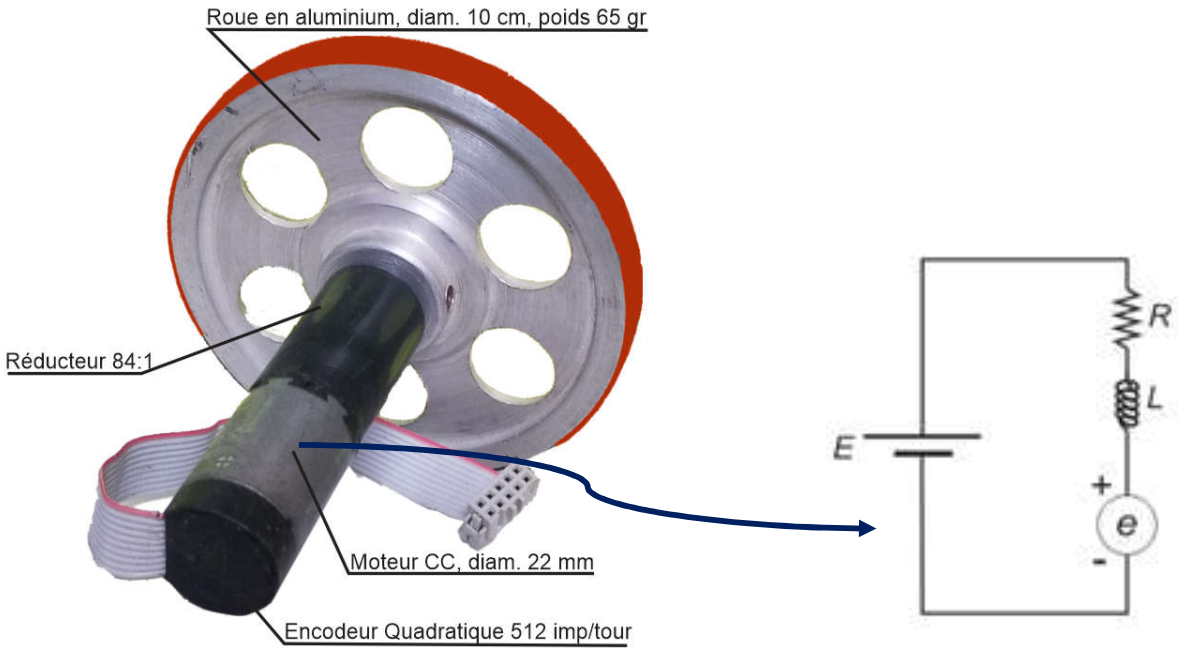


Figure 1.1 : Schéma bloc de la carte embarquée sur la plateforme mobile

En ce qui concerne cette partie du travail, il s'agit de contrôler l'un des deux moteurs CC en boucle ouverte, ceci afin de mesurer sa vitesse de rotation lorsque celui-ci reçoit le rapport cyclique d'une tension d'alimentation sous forme de signaux numériques modulés en largeur d'impulsion ou bien PWM (Pulse Width Modulation). L'étage de puissance est assuré par un pont en H de type L298N. Ce dernier permet d'amplifier le signal généré par l'unité PWM du microcontrôleur. L'utilisation du module série UART (Universal Asynchronous Transmit and Receive) du microcontrôleur permet de recevoir le rapport cyclique issu du programme Simulink et d'envoyer la vitesse mesurée à ce même programme grâce à l'unité QEI (Quadrature Encoder Interface), intégrée au microcontrôleur. Chaque module QEI reçoit deux signaux numériques déphasés de 90°, ces derniers proviennent des encodeurs montés sur les axes des deux moteurs du robot. Les modules QEI permettent non seulement de réaliser le comptage du nombre d'impulsions reçu mais aussi d'obtenir le sens de rotation des moteurs. Afin d'obtenir les vitesses de rotation, un calcul de la dérivée de ces signaux est effectué au niveau du *Firmware*, préalablement chargé sur la mémoire du microcontrôleur.

1.1.1 Modélisation du Moteur à Courant Continu (MCC)

Le moteur à courant continu est le moyen de base par lequel une énergie électrique est transformée en énergie mécanique. Les déplacements du robot mobile décrit précédemment sont assurés par une paire de moteurs à courant continu montés de part et d'autre du centre de ce robot. La photo de la figure 1.2 montre une partie de l'ensemble du système assurant la mobilité du robot ainsi que le schéma électrique équivalent du moteur CC. L'axe de rotation du moteur est connecté à un réducteur de vitesse qui est responsable de la mise en rotation de la roue réalisant les mouvements du robot. L'autre extrémité de l'axe du moteur est reliée à un encodeur quadratique permettant de mesurer l'angle de rotation effectué par l'axe du moteur.



(a) (b)

Figure 1.2 Motoréducteur avec roue (a), Schéma électrique équivalent du moteur CC (b)

1.1.2 Modèle dynamique du MCC

Le modèle global régissant le fonctionnement d'un MCC est donné par les deux équations mécanique et électriques suivantes :

$$\begin{cases} J\dfrac{d\omega}{dt} + f\omega(t) = \phi_0 i(t) \\ \dfrac{di}{dt} = \dfrac{1}{L}(E - Ri(t) - \phi_0\omega(t)) \end{cases} \quad (1.1)$$

Avec J : le moment d'inertie de l'axe de rotation du moteur, ω la vitesse de rotation, f les frottements visqueux s'opposant à la rotation de l'axe moteur, ϕ_0 la constante de couple, R la résistance électrique du bobinage de l'induit, L l'inductance de ce bobinage, i le courant traversant les bobines de l'induit lorsque le moteur est alimenté par la tension E. En utilisant la transformée de Laplace le système d'équations différentielles précédent peut se réécrire sous la forme :

$$\begin{cases} JS\omega(s) + f\omega(s) = \phi_0 i(s) \\ SI(s) = \dfrac{1}{L}(E(s) - RI(s) - \phi_0\omega(s)) \end{cases} \quad (1.2)$$

Donc

$$I = \frac{E - \phi_0\omega}{LS + R}$$

En remplaçant I dans l'équation mécanique nous obtenons :

$$(JS + f)\omega = \phi_0 \frac{E - \phi_0\omega}{LS + R} = \frac{\phi_0 E}{LS + R} - \frac{\phi_0^2\omega}{LS + R}$$

Par conséquent

$$\left(JS + f + \frac{\phi_0^2}{LS + R}\right)\omega = \frac{\phi_0 E}{LS + R}$$

Ce qui donne

$$\frac{\omega}{E} = \frac{\phi_0}{JLS^2 + (RJ + fL)S + Rf + \phi_0^2} \quad (1.3)$$

L'équation (1.3) représente la fonction de transfert du moteur CC ayant comme entrée la tension d'alimentation E et comme sortie la vitesse de rotation ω.

1.1.3 Simulation du moteur utilisé

L'équation (1.3) est une fonction de transfert d'un système linéaire de second ordre ayant deux pôles et pas de zéros. Afin de vérifier cette linéarité sur le moteur CC que nous avons utilisé, nous l'avons soumis à une alimentation variante entre -9.5 et +9.5 volts, augmentant et diminuant d'une manière linéaire, tel qu'illustré par la figure 1.3. Les résultats de la figure ci-dessous montrent que le moteur CC a bien un comportement linéaire en dehors de la zone morte, située au voisinage d'une tension d'alimentation nulle.

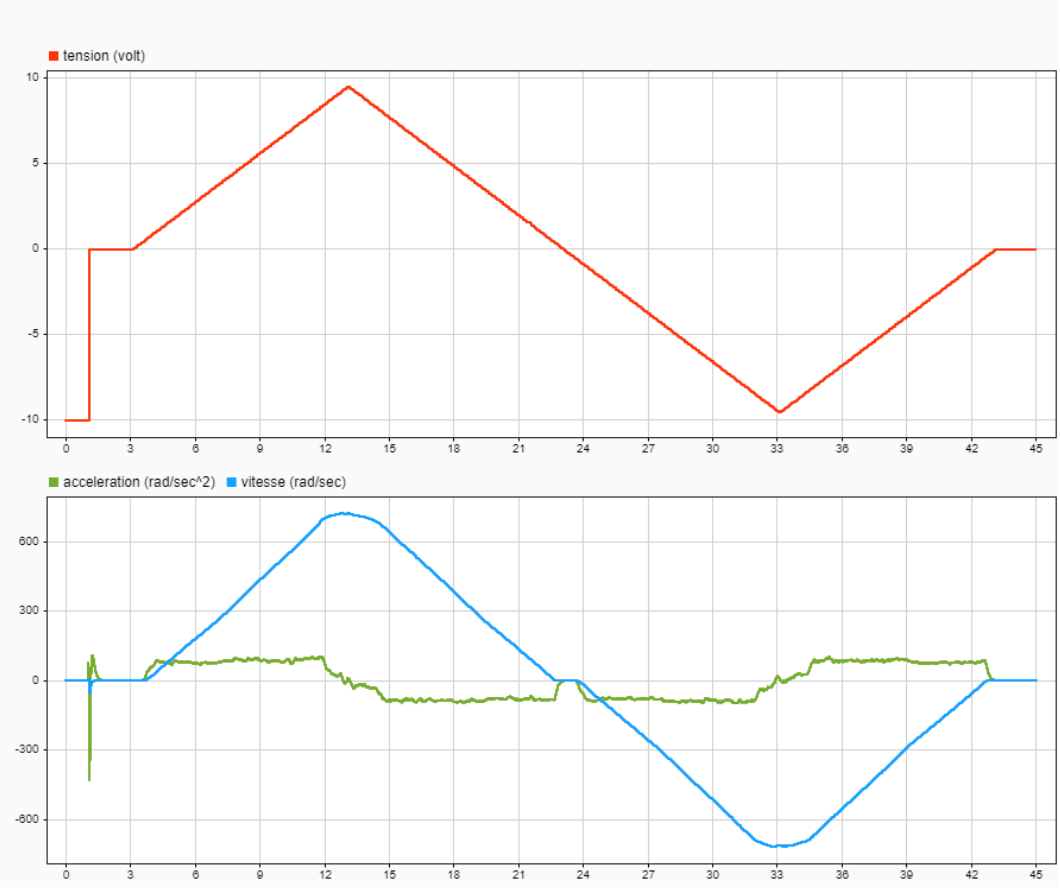


Figure 1.3 test de linéarité

Afin de mettre en évidence ce phénomène et représenter graphiquement la caractéristique vitesse/tension de ce moteur les données précédentes ont été réutilisées pour obtenir la figure 1.4. Nous pouvons aussi constater une hystérésis sur le graphe obtenu, cette dernière a pour effet de ralentir légèrement la rotation du moteur lorsque celui-ci est soumis à un changement brusque de l'accélération, ceci étant influencé par l'inertie du système, qui a pour effet de s'opposer aux variations de mouvements.

Un zoom sur le centre de cette figure révèle une zone morte, où le moteur est bloqué lorsque la tension d'alimentation se situe dans l'intervalle -0.5 volt et +0.5 volt, ceci est dû aux frottements secs qui empêchent la rotation du moteur. Cette figure nous permet aussi d'obtenir une estimation de la constante de vitesse $K_v \cong (\tan \alpha)^{-1} \cong \phi_0^{-1}$.

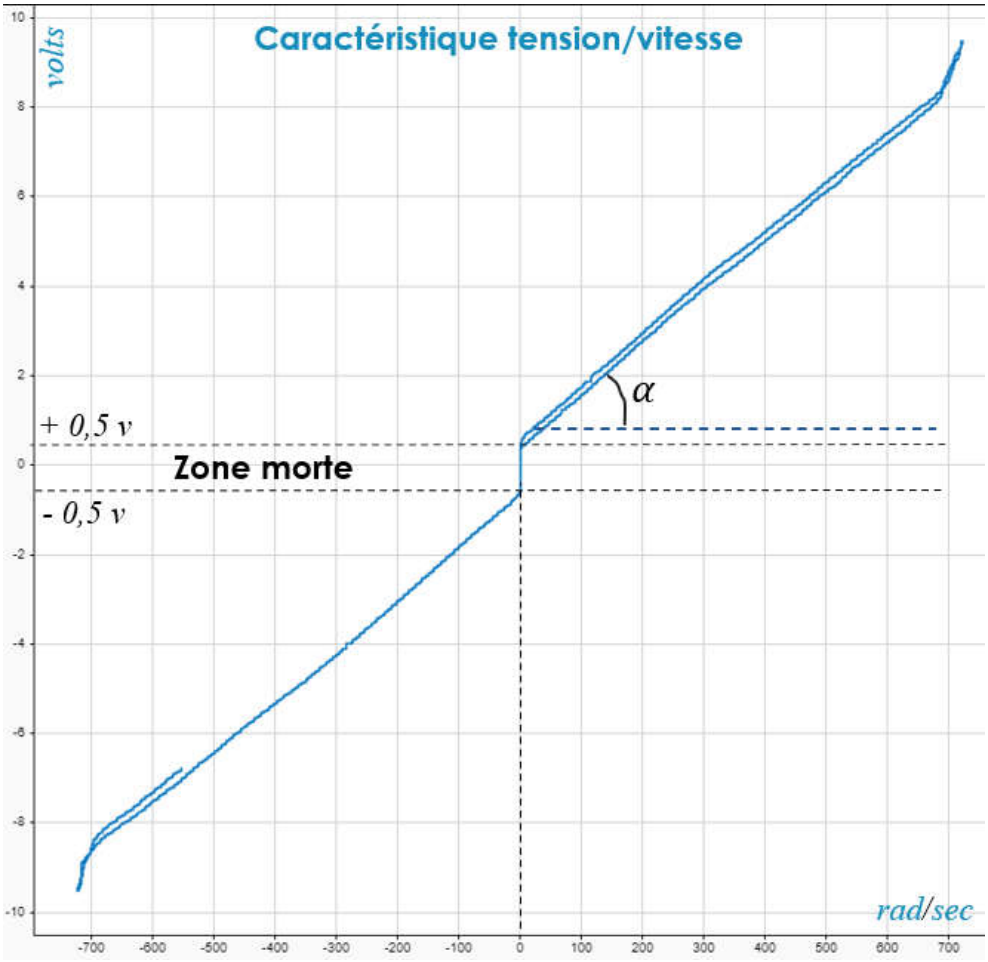


Figure 1.4 caractéristique vitesse/tension du moteur CC

La pente d'une droite parallèle à la courbe de cette figure donne une valeur très proche de la constante de vitesse qui peut être évaluée par $K_v \cong 84.74\ (\frac{rad}{s})/v$, ce qui n'est pas très loin de la valeur donnée sur la fiche du constructeur [Annexe 2] sur laquelle il est mentionné que $K_v = 875\left(\frac{tr}{min}\right)/v = 91.58\ \left(\frac{rad}{s}\right)/v$, la constante de couple mentionnée par le constructeur est $\phi_0 = K_v^{-1} = 0.0109\ nm/A$. Les autres données du constructeur sont comme suit : $R = 6.01\ \Omega,\ L = 0.362\ H,\ J = 4.29\ 10^{-7} kg\ m^2$, pour ce qui concerne le coefficient de frottements f il ne peut pas être mentionné car il n'existe pas de modèle permettent de le calculer précisément puisque ces frottements sont aléatoires et peuvent varier sans aucune information préalable.

Afin de tester le modèle précédent avec les données constructeur nous avons utilisé l'environnement Matlab/Simulink pour simuler le comportement du moteur. Pour réaliser cela il a fallu trouver une valeur convenable pour le coefficient de frottements visqueux f, celui-ci a été choisi en se basant sur le reste des données constructeur tels que la vitesse et le courant consommé lorsque le moteur tourne à vide, la valeur ainsi trouvée est $f = 7.66\ 10^{-7} Nm\ s/rad$. Avec l'ensemble de ces données et en utilisant l'équation (1.3) nous obtenons la fonction de transfert, du moteur à vide, suivante :

$$G_0(S) = \frac{\omega}{E} = \frac{0.0109}{1.553\ 10^{-7}\ S^2 + 2.856\ 10^{-6}\ S + 1.234\ 10^{-4}} \qquad (1.4)$$

Tel que cela est illustré par la figure 1.2, ce moteur doit faire tourner une charge constituée d'un réducteur de vitesse lui-même connecté à une roue en aluminium. Pour modéliser l'ensemble de ce système nous devons prendre en considération le moment d'inertie de la roue qui est assimilée à celle d'un disque plein, ce dernier est donné par $J_{Roue} = \frac{1}{2}MR^2$, avec $M = 65\ g$ et $R = 5\ cm$ étant la masse et le rayon de la roue respectivement, ceci nous donne $J_{Roue} = 81.25\ 10^{-6}\ kg\ m^2$. Pour ce qui concerne le moment d'inertie du réducteur le constructeur indique une valeur de $J_{Red} = 0.4\ 10^{-7}\ kg\ m^2$. Afin d'obtenir le moment d'inertie total subi par le moteur nous pouvons utiliser la formule suivante : $J_{Tot} = J_{Mot} + T_{Red} + \frac{J_{Roue}}{N^2}$ avec N le rapport de réduction de la vitesse, celui-ci constitue aussi un facteur multiplicateur

du couple disponible sur l'axe du moteur. En remplaçant par les valeurs numériques précédentes nous aurons : $J_{Tot} = 4.29\ 10^{-7} + 0.4\ 10^{-7} + \frac{81.25\ 10^{-6}}{84^2} = 4.80\ 10^{-7} kg\ m^2$. Bien que ces valeurs soient connues avec précision, elles ne peuvent pas être considérées comme étant des valeurs exactes sur lesquelles nous pouvons nous baser pour contrôler la vitesse du moteur dans le monde réel. Néanmoins cette première approximation va nous permettre d'utiliser ce modèle de simulation comme point de départ pour faire varier les paramètres du moteur de sorte à faire coïncider la vitesse de rotation du moteur simulé avec celle mesurée à partir du moteur réel. Cela est rendu possible grâce à l'utilisation de l'application ''*parameter estimator*'' faisant partie de l'environnement Matlab. Afin d'exploiter cet outil il faut disposer des entrées/sorties issues du modèle simulé ainsi que celles mesurées à partir du moteur réel. Une fois lancé, l'algorithme d'optimisation utilisé va faire varier les paramètres du modèle simulé pour faire en sorte que la vitesse de sortie de ce modèle se rapproche le plus possible de la vitesse mesurée. Les paramètres obtenus après cette optimisation sont :

$J = 4.67\ 10^{-7}\ kg\ m^2, \quad \phi_0 = 0.012\ \ Nm/A, \quad L = 0.488\ H, \quad R = 13.24\ \Omega$

$f = 1.25\ 10^{-7}\ Nm\ s/rad.$

Ainsi la fonction de transfert du système complet tel que représenté par l'équation 1.3 devient :

	$G_1(s) = \frac{0.012}{2.27\ 10^{-7}\ S^2 + 6.244\ 10^{-6}\ S + 1.457\ 10^{-4}}$	(1.5)

Cette équation représente le modèle du moteur exprimé dans le domaine continu, afin de l'exploiter dans le cadre d'une commande numérique, sujet du présent travail, il faut le convertir dans le domaine discret. Pour cela la fonction *c2d* (continue to discrete), disponible au sein de l'environnement Matlab, a été utilisée pour générer la transformée en z de la fonction de transfert de l'ensemble moteur-réducteur-roue :

	$H(z) = \frac{0.02609\ z + 0.02585}{z^2 - 1.972\ z + 0.973}$	(1.6)

Pour obtenir ce résultat il est nécessaire de préciser que la fonction *c2d* utilise un bloqueur d'ordre zéro pour échantillonner les données avec une fréquence de 1 kHz pour ce qui concerne le cas présent.

La figure 1.5 montre les réponses, en termes de vitesses de rotation pour le moteur simulé avant et après optimisation ainsi que celle mesurée à partir du moteur réel, lorsque ce moteur est soumis à une tension variable. D'après les courbes obtenues il est clair que la vitesse du moteur dont le modèle est donné par la fonction de transfert H(z) de l'équation (1.6), représentée par la courbe en couleur verte, est beaucoup plus rapprochée de celle mesurée à partir du moteur réel. Cela signifie donc que ce modèle peut être considéré comme étant une très bonne approximation du comportement de ce moteur. La courbe en rouge, représentant la vitesse du moteur à vide, utilisant les données du constructeur, sans tenir compte des pertes d'énergie, montre des différences notables avec les deux autres graphes et notamment durant les régimes transitoires.

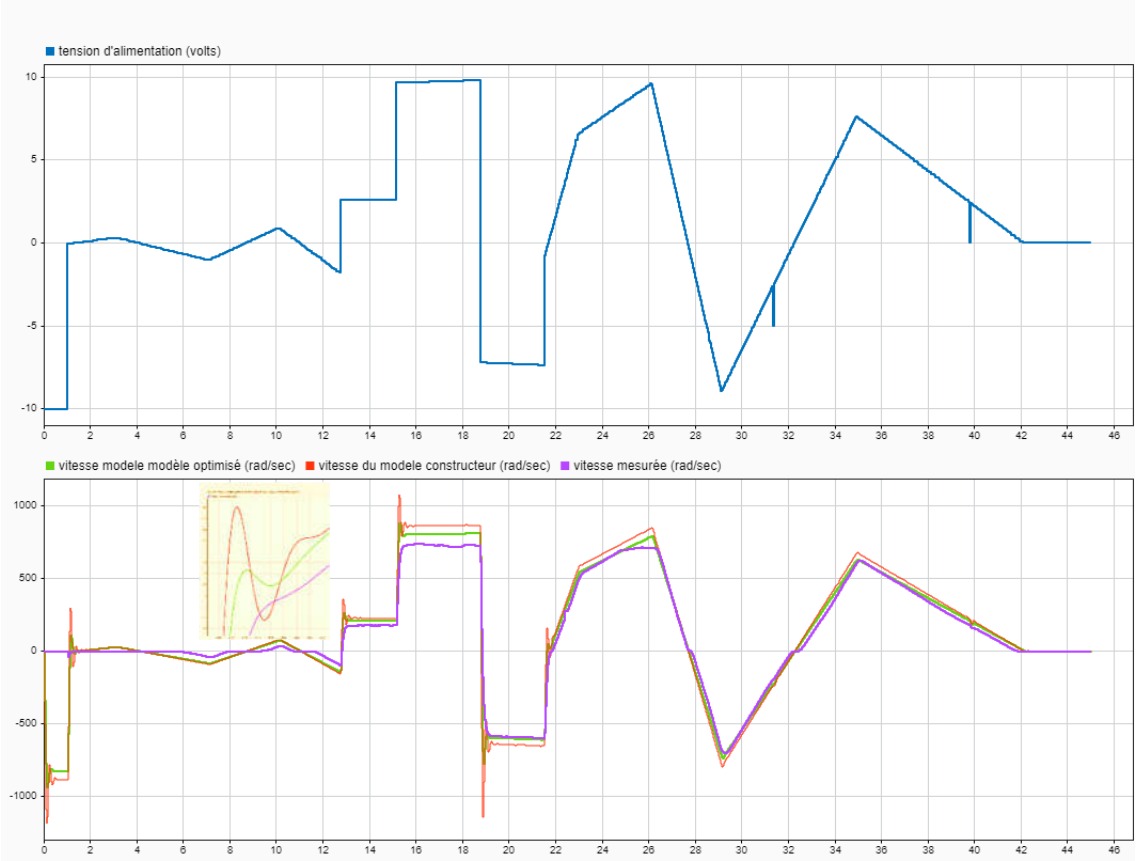


Figure 1.5 : Réponses du moteur avant et après optimisation

1.2 Description et Modélisation du Robot expérimental

La photo de la figure 1.6 montre une vue générale du robot spécialement conçu pour les besoins de ce projet. Nous pouvons ainsi distinguer la taille et la manière avec laquelle les différentes parties mécaniques et électroniques sont disposés sur cette plateforme.

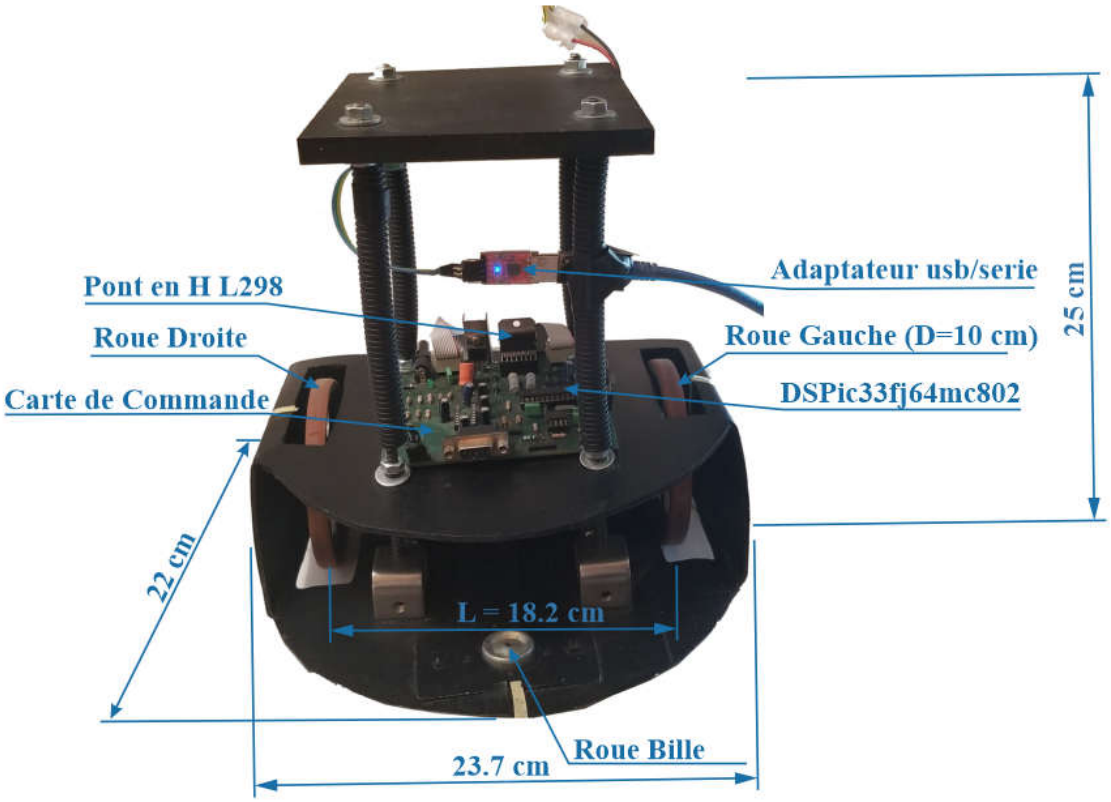


Figure 1.6 : Vue générale du robot utilisé dans ce travail

1.2.1 Modèle cinématique du Robot unicycle

La figure 1.7 montre l'exemple d'un robot qui s'est déplacé d'une posture (position et orientation) $P_1 = (x_1, y_1, \theta_1)$ à une posture $P_2 = (x_2, y_2, \theta_2)$. Ce déplacement étant considéré comme ayant eu lieu pendant une durée Δt suffisamment faible pour assimiler la portion de la trajectoire du robot à un arc de cercle dont le centre se situe au point CRI (Centre de Rotation Instantanée) et ayant comme rayon R_C. Les distances parcourues par les roues gauche et droite sont $S_G = R_G\alpha$ et $S_D = R_D\alpha$ respectivement. Au même moment, le robot (centre) s'est déplacé d'une distance $S_C = R_C\alpha$. En divisant tous les termes de ces deux équations par Δt

nous obtenons les vitesses linéaires de la roue gauche $v_G = R_G\dot{\alpha}$, celle de la roue droite $v_D = R_D\dot{\alpha}$ et $v = R_C\dot{\alpha}$, pour celle du centre du robot. Sachant que $R_D = R_G + L$, L étant la distance séparant les roues gauche et droite du robot, nous auront donc $v_D - v_G = ((R_G + L) - R_G)\dot{\alpha} = L\dot{\alpha}$. Tenant compte du fait que $\alpha = \theta_2 - \theta_1 = \Delta\theta$ nous avons $\dot{\alpha} = \dot{\theta}$. Si nous appelons ω_G et ω_D les vitesses angulaires des roues gauche et droite du robot nous pouvons calculer les vitesses linéaires de ces deux roues en utilisant $v_G = r\omega_G$ et $v_D = r\omega_D$ avec $r=D/2$ le rayon des deux roues motrices. Nous aurons ainsi $r\omega_D - r\omega_G = r(\omega_D - \omega_G) = L\dot{\theta}$ avec $\dot{\theta}$ la vitesse angulaire du robot autour du CRI qui sera notée Ω, ceci nous donne donc

$$\Omega = \frac{r}{L}(\omega_D - \omega_G) \tag{1.7}$$

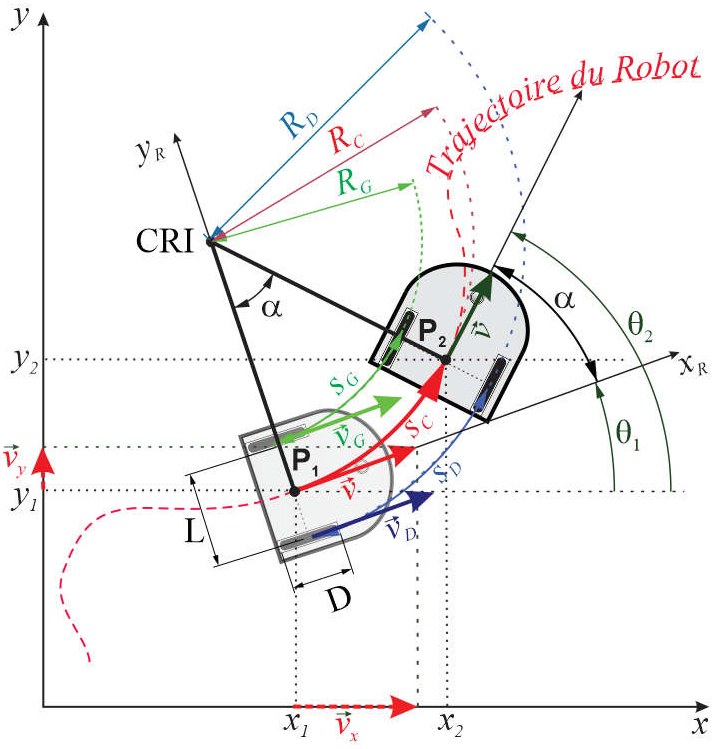


Figure 1.7. Déplacements élémentaires dans un repère (x, y) fixe

A partir de la figure 1.7 nous avons $R_C = R_G + L/2$ et $R_C = R_D - L/2$, par conséquent $2R_C = R_G + R_D$ ou bien $R_C = (R_G + R_D)/2$, en multipliant ces termes par $\dot{\alpha}$ nous obtenons $R_C\dot{\alpha} = (R_G\dot{\alpha} + R_D\dot{\alpha})/2$ c'est-à-dire $v = (v_G + v_D)/2$ ou bien $v = (r\omega_G + r\omega_D)/2$ finalement

$$v = \frac{r}{2}(\omega_D + \omega_G) \tag{1.8}$$

Les équations 1.7 et 1.8 donnent les relations des vitesses angulaires des deux roues d'un robot mobile différentiel et celles d'un robot de type unicycle qui serait constitué d'une roue unique pouvant tourner autour d'un axe vertical avec la vitesse angulaire Ω et se déplaçant avec la vitesse linéaire v.

De la figure 1.7 les projections de la vitesse v sur les axes x et y donnent, respectivement $v_x = \dot{x} = v\cos\theta$ et $v_y = \dot{y} = v\sin\theta$. L'ensemble de ces équations définissent le modèle cinématique du robot différentiel.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \Omega\end{bmatrix} \tag{1.9}$$

1.2.2 Contrainte non-holonome

Le long de la ligne passant par l'axe de rotation des deux roues du robot, les composantes de la vitesse v sont v_x' et v_y' comme le montre la figure 1.8 ci-dessous.

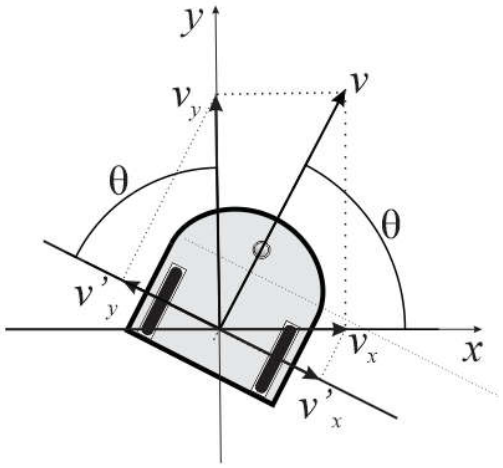


Figure 1.8: Projections des vitesses sur le plan fixe (x, y) sur l'axe latéral du robot

A partir de cette figure, nous avons $v'_x = v_x \cos\left(\frac{\pi}{2} - \theta\right) = v_x \sin\theta$ et $v'_y = v_y \cos\theta$ ou bien $v'_x = \dot{x}\sin\theta$ et $v'_y = \dot{y}\cos\theta$ la résultante de ces deux composantes est nulle car le robot ne peut pas se déplacer sur cette direction $\dot{x}\sin\theta - \dot{y}\cos\theta = 0$, cette équation traduit ce qui est connu sous l'appellation « *Contrainte non holonome* » de ce robot, elle peut aussi s'exprimer par $\tan\theta = \frac{\dot{y}}{\dot{x}}$.

1.2.3 Rayon de Rotation Instantanée

De la figure 1.7 nous avons $v_D = R_D\Omega$ et $v_G = R_G\Omega$ avec $R_G = R_C - \frac{L}{2}$ et $R_D = R_C + \frac{L}{2}$ R_C est le rayon de rotation du robot, donc $\frac{v_D}{v_G} = \frac{R_D}{R_G} = \frac{R_C + \frac{L}{2}}{R_C - \frac{L}{2}}$ par conséquent $v_D\left(R_C - \frac{L}{2}\right) = v_G\left(R_C + \frac{L}{2}\right)$ ce qui donne $R_C(v_D - v_G) = \frac{L}{2}(v_D + v_G)$ et finalement $R_C = \frac{\boldsymbol{L}}{\boldsymbol{2}}\left(\frac{\boldsymbol{v_D + v_G}}{\boldsymbol{v_D - v_G}}\right) = \frac{\boldsymbol{L}}{\boldsymbol{2}}\left(\frac{\boldsymbol{\omega_D + \omega_G}}{\boldsymbol{\omega_D - \omega_G}}\right)$, le robot suit une ligne droite si $\omega_G = \omega_D$ c'est-à-dire $R_C = \infty$ il tourne sur lui-même si $R_C = 0$ ou bien $\omega_G = -\omega_D$.

1.4 conclusion

Ce chapitre a été consacré à la modélisation du système de locomotion du robot mobile. Ce dernier est principalement constitué de deux motoréducteurs dont les axes de rotation sont solidaires des deux roues gauche et droite, assurant les mouvements de ce robot. Pour se faire il a fallu d'abord établir les équations dynamiques du moteur à courant continu pour ensuite déduire un modèle global représentatif de l'ensemble du système en rotation. Bien que les frottements statiques n'aient pas été pris en compte avec le modèle linéaire final, cela n'a pas eu de conséquences sur le résultat final de cette modélisation tel que montré sur la figure 2.5. La deuxième partie de ce chapitre a donné lieu au modèle cinématique permettant d'établir le lien existant entre les vitesses de rotation des deux moteurs avec les vitesses linéaire et angulaire de la plateforme mobile.

Chapitre 2

Pilotage d'un robot mobile autonome vers une cible fixe

Dans ce chapitre nous allons aborder la partie commande du robot mobile décrit précédemment. Le but de cela étant de faire en sorte que le robot puisse se déplacer, d'une manière autonome, pour atteindre une posture ou cible, définie par les coordonnées cartésiennes et orientation finales. L'approche adoptée pour y arriver, consiste à enchainer deux types de contrôleurs en série ; le premier étant de prendre en charge la partie dynamique du système en agissant directement sur les deux moteurs faisant déplacer le robot, le second aura pour rôle de fournir les consignes en termes de vitesses de rotation pour chacun de ces deux moteurs. Ce dernier sera calculé en tenant compte des équations cinématiques du robot présentées au chapitre précédent.

Figure 2.1: Schéma bloc de la commande globale

2.1 Contrôle de la vitesse d'un moteur CC

La partie dynamique de la commande proposée consiste en une paire de contrôleurs PID dont les paramètres ont été réglés en utilisant l'outil ou *toolbox pidtuner* disponible au sein de l'environnement Matlab/Simulink.

La figure 2.2 montre le programme de simulation représentant la commande en boucle fermée, utilisée pour le réglage des paramètres du contrôleur PID assurant l'asservissement,

en vitesse, de l'un des moteurs CC, dont la fonction de transfert est donnée par l'équation 1.6.

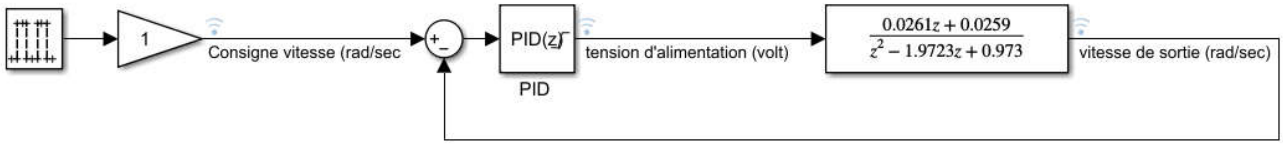


Figure 2.2: Programme Simulink de la commande PID

Après seulement quelques réglages au niveau des marges de gain et de phase du contrôleur PID il s'est avéré que seulement les deux termes P et I étaient nécessaires pour une commande suffisamment rapide et robuste tel que le montrent les résultats de la figure 2.3.

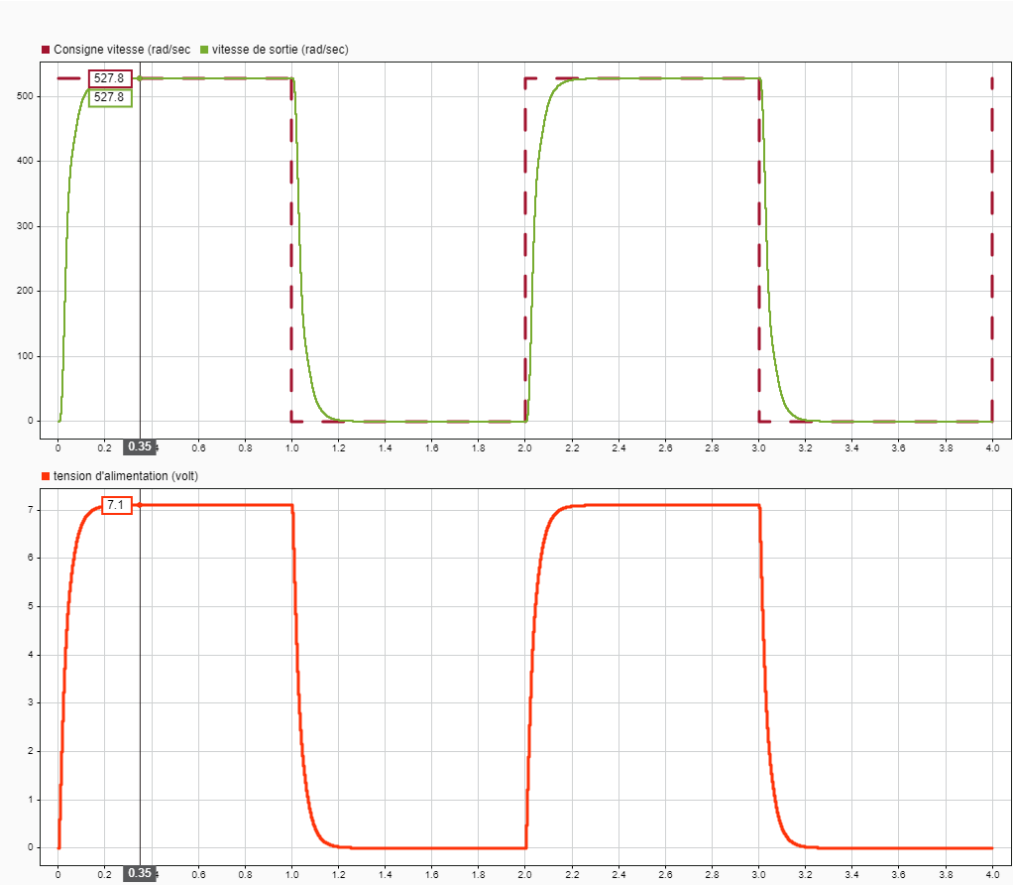


Figure 2.3 : Réponses du moteur CC à une consigne vitesse de 528 rad/sec

Table 2.1 Paramètres du Contrôleur PI (Période d'échantillonnage = 0.2 ms)

k_P	k_I
0.0000322	0.3226

Les résultats du de la figure 2.3 montrent que le contrôleur utilisé ayant pour paramètres ceux donnés par le tableau 2.1, est capable de faire atteindre au moteur la vitesse désirée en un peu moins de 0.35 seconde. La valeur de la consigne qui est de 528 rad/sec a été choisie de sorte que nous puissions obtenir une vitesse de rotation de l'une des deux roues du robot de 1 tour/sec ou bien 2π rad/sec, cela est dû au fait que le réducteur attaché à l'axe du moteur donne une vitesse de rotation 84 fois moins que celle de l'axe du moteur. Sachant que le rayon des deux roues du robot est de 5 cm nous pouvons déterminer que si les deux moteurs sont soumis à la même consigne de 2π rad/sec, la vitesse linéaire du robot sera donnée par l'équation : $v = 2\pi r = 0.314\ m/sec$, cela signifie que le robot est capable d'atteindre cette vitesse dans un intervalle de temps de 0.35 seconde.

2.2 Commandes cinématiques

2.2.1 Représentation dans l'espace d'état

Parmi les types de modèles mathématiques utilisés pour l'élaboration de commande de systèmes, la représentation sous forme de modèles d'état est probablement la plus utilisée. Afin de modéliser les déplacements du robot mobile utilisé dans ce travail nous avons illustré, sur la figure 2.4, une situation où le robot effectue une transition lui permettant de passer d'une posture $P_1 = (x_1, y_1, \theta_1)$ à une autre posture $P_2 = (x_2, y_2, \theta_2)$. Pour accomplir cette tâche le robot doit calculer les vitesses linéaire et angulaire à chaque période d'échantillonnage. La partie translation du mouvement est obtenue à partir de la projection de la vitesse linéaire v, sur la direction du segment de droite reliant les points de départ et d'arrivée, cette dernière est désignée par v_ρ. Les changements d'orientation se font en tenant compte du fait que l'angle du virage que le robot effectue est une combinaison des deux composantes angulaires ; α qui constitue la portion qui fait que le robot tourne vers la

direction passant par la position cible P_2 et β l'angle qui le fait tourner de sorte qu'il puisse atteindre l'orientation finale θ_2.

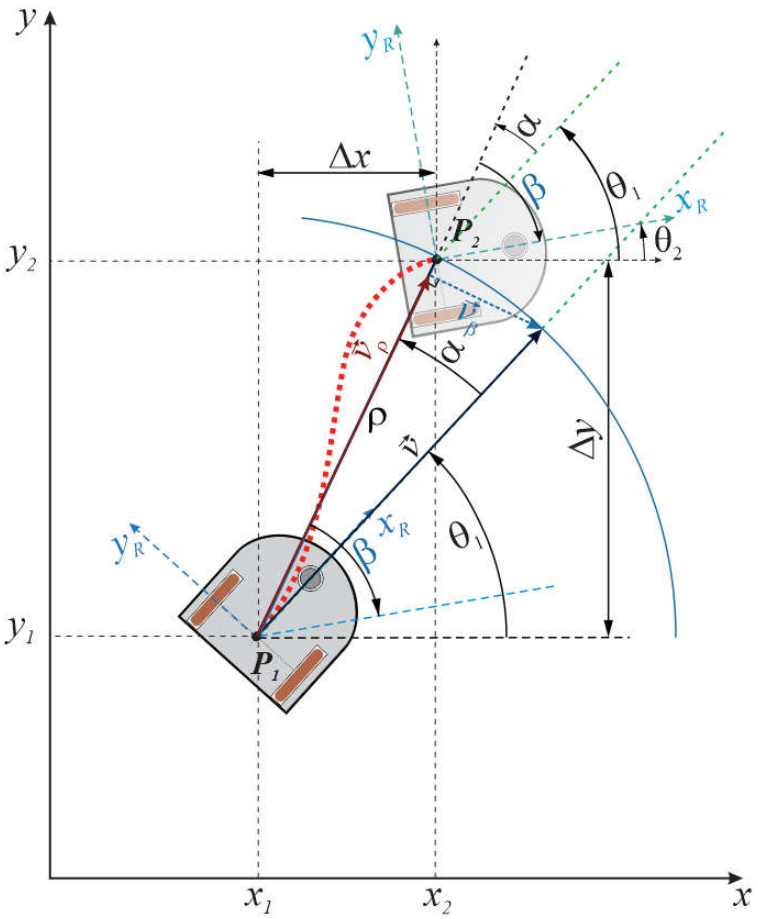


Figure 2.4 : Transition du robot mobile entre deux postures successives

Le comportement global du robot peut ainsi être modélisé par le système d'équations suivant :

$$\begin{cases} \rho = \sqrt{\Delta x^2 + \Delta y^2} \\ \alpha = \tan^{-1}\dfrac{\Delta y}{\Delta x} - \theta_1 \\ \beta = -\alpha + \theta_2 - \theta_1 \end{cases} \qquad (2.1)$$

Lorsque le robot se déplace vers sa position cible le taux de variations de la distance ρ doit diminuer. Cela peut se traduire par : $\dot{\rho} = v_\rho = -v\cos\alpha$. A partir de la figure 2.4 nous pouvons constater que le terme noté $\vec{v}_\beta$, peut être assimilé à la projection de la vitesse $\vec{v}$ sur l'arc de cercle de rayon ρ, par conséquent la vitesse angulaire avec laquelle l'angle β varie, peut être estimée par l'expression suivante : $\rho\dot{\beta} = v_\beta = -v\sin\alpha$. En utilisant la troisième partie des équations (2.1) nous obtenons : $\dot{\alpha} = -\dot{\beta} - \dot{\theta} = -\dot{\beta} - \Omega$ car $\dot{\theta}_2 = 0$ puisque θ_2 est constant. Finalement le système précédent peut être mis sous la forme du modèle d'état suivant :

$$\begin{cases} \dot{\rho} = -v\cos\alpha \\ \dot{\beta} = -\dfrac{v}{\rho}\sin\alpha \\ \dot{\alpha} = \dfrac{v}{\rho}\sin\alpha - \Omega \end{cases} \tag{2.2}$$

Qui peut aussi se mettre sous la forme matricielle :

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \dfrac{\sin\alpha}{\rho} & -1 \\ -\dfrac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \Omega \end{bmatrix} \tag{2.3}$$

2.2.2 Commande linéaire

La figure 2.4 montre que lorsque le robot atteint la posture finale $P_2 = (x_2, y_2, \theta_2)$ les trois états ρ, α et β du système décrit par (2.2) sont nuls. Afin d'arriver à ce résultat il faut agir, à chaque instant, sur les entrées de commande v et Ω de sorte à ce que le robot puisse suivre la trajectoire virtuelle illustrée par la couleur rouge sur cette figure.

Dans ce qui suit nous allons adopter une stratégie de commande qui s'appuie sur le fait que la vitesse v, responsable de la partie translation du mouvement, varie d'une manière proportionnelle à la distance ρ qui sépare le robot de la position finale. De la même manière nous allons considérer que, pour faire aligner le robot avec l'orientation finale θ_2, la vitesse angulaire Ω doit varier en tenant compte des deux composantes α et β. A partir de ces observations les lois de commandes suivantes sont proposées ;

$$v = k_\rho \rho \tag{2.4}$$

et

$$\Omega = k_\alpha \alpha + k_\beta \beta \tag{2.5}$$

En remplaçant les expressions de v et Ω dans (2.3) nous obtenons :

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \dfrac{\sin\alpha}{\rho} & -1 \\ -\dfrac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} k_\rho \rho \\ k_\alpha \alpha + k_\beta \beta \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos\alpha \\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin\alpha \end{bmatrix} \tag{2.6}$$

Sachant que les variations de trajectoire se font à chaque période d'échantillonnage les segments de droite constituant cette trajectoire sont de dimensions très réduites. Cela nous permet de les considérer comme étant proches de zéro et par conséquent nous pouvons faires les approximations : $\cos\alpha \simeq 1$ et $\sin\alpha \simeq \alpha$.

Ainsi la relation matricielle (2.6) devient :

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho & 0 & 0 \\ 0 & -k_\alpha + k_\rho & -k_\beta \\ 0 & -k_\rho & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} \tag{2.7}$$

Cette expression qui est de la forme $\dot{X} = AX$, représente un système linéaire dont la stabilité dépend des valeurs propres de la matrice A. Les parties réelles de ces valeurs propres doivent être négatives, sachant que ces dernières sont les racines de l'équation $det[\lambda I - A] = 0$, avec λ le vecteur contenant les valeurs propres. Dans ce cas nous avons :

$$\lambda I - A = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \begin{bmatrix} -k_\rho & 0 & 0 \\ 0 & -k_\alpha + k_\rho & -k_\beta \\ 0 & -k_\rho & 0 \end{bmatrix}$$

Donc :

$$\lambda I - A = \begin{bmatrix} \lambda + k_\rho & 0 & 0 \\ 0 & \lambda + k_\alpha - k_\rho & k_\beta \\ 0 & k_\rho & \lambda \end{bmatrix} \quad (2.8)$$

Ainsi nous aurons :

$$det[\lambda I - A] = (\lambda + k_\rho)\left[(\lambda + k_\alpha - k_\rho)\lambda - k_\beta k_\rho\right] = (\lambda + k_\rho)(\lambda^2 + \lambda(k_\alpha - k_\rho) - k_\beta k_\rho)$$

Les racines de cette équation sont :

$$\lambda_1 = -k_\rho,\ \lambda_2 = \frac{-(k_\alpha - k_\rho) + \sqrt{(k_\alpha - k_\rho)^2 + 4k_\rho k_\beta}}{2} \quad \text{et} \quad \lambda_3 = \frac{-(k_\alpha - k_\rho) - \sqrt{(k_\alpha - k_\rho)^2 + 4k_\rho k_\beta}}{2}$$

Nous pouvons donc déduire que pour que $\lambda_1 < 0$ il faut choisir $k_\rho > 0$, pour avoir $\lambda_2 < 0$ et $\lambda_3 < 0$ il suffit de choisir $k_\beta < 0$ et $(k_\alpha - k_\rho) > 0$.

2.2.3 Implémentation d'une commande linéaire

Avant de tester les performances de l'algorithme de control précédent, sur le robot réel, quelques tests en simulation sont proposés. Le schéma bloc de la figure 2.5 résume l'ensemble des opérations nécessaires à l'implémentation de cette commande, incluant tous les signaux d'entrées/sorties connectées avec le bloc Robot/Modèle. Ce bloc peut remplacer aussi bien le modèle mathématique des deux boucles fermées comprenant les commandes dynamiques des deux moteurs CC, ou bien le robot réel connecté à un ordinateur fixe exécutant le même algorithme de contrôle. Ce bloc accepte deux signaux d'entrée et génère deux signaux de sortie ; Ces derniers étant respectivement les vitesses de rotation des roues gauche et droite (ω_G, ω_D), calculées par le contrôleur et (ω_{GM}, ω_{DM}), celles mesurées sur des axes des deux moteurs simulés, ou à partir des encodeurs pour le cas du robot réel.

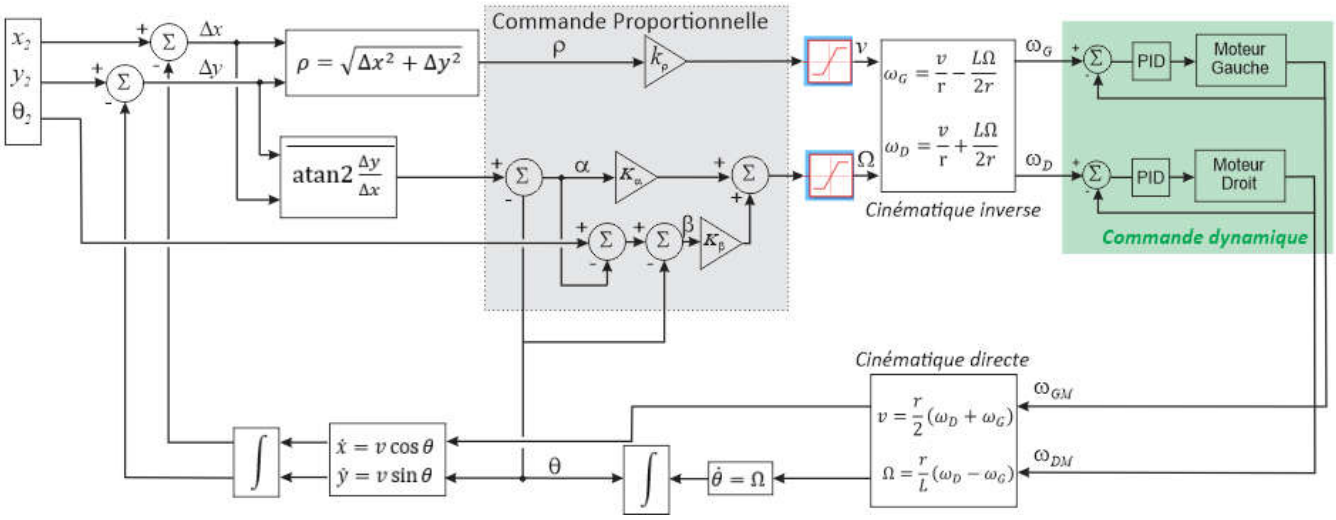


Figure 2.5 Commande linéaire dirigeant le robot vers une posture désirée (x_2, y_2, θ_2)

Sur la base du schéma bloc de la figure 2.5, ce contrôleur a été implémenté sous forme d'un programme Simulink sous l'environnement Matlab. Après quelques essai-erreurs les gains du contrôleur mentionnés sur le tableau 2.2 ont donné de très bons résultats, ceci indépendamment des choix de postures cibles sélectionnées.

Table 2.2 Gains du contrôleur linéaire

k_ρ	k_α	k_β
2.0	5.0	-1.0

2.2.4 Résultats et analyses en simulation

Parmi les nombreux tests effectués, nous avons sélectionné quelques échantillons pour des cibles situés sur les quatre quadrants du plan (x, y). La figure 2.6 montre les résultats obtenus pour les cas de cibles mentionnées sur cette figure.

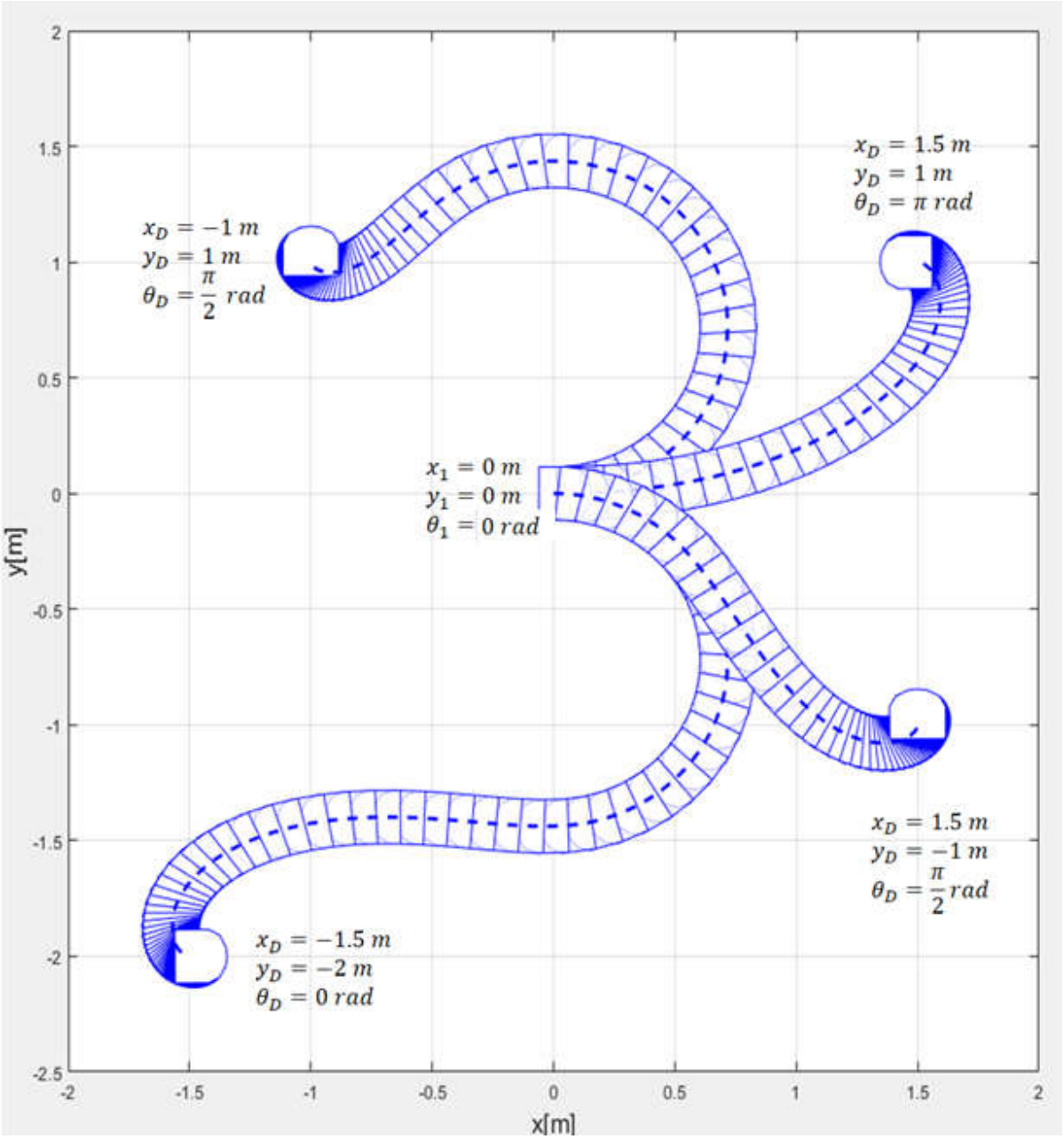


Figure 2.6 Trajectoires suivies par le robot pour atteindre 4 postures cibles

Les tracés de trajectoires ci-dessus montrent que le robot a pu atteindre les cibles proposées avec succès. Afin de donner plus de détails sur la manière et la rapidité avec laquelle le robot a pu accomplir ses missions, la figure 2.7 montre les variations temporelles des vitesses linéaires et angulaires pour le cas du bas à gauche de la figure 2.6.

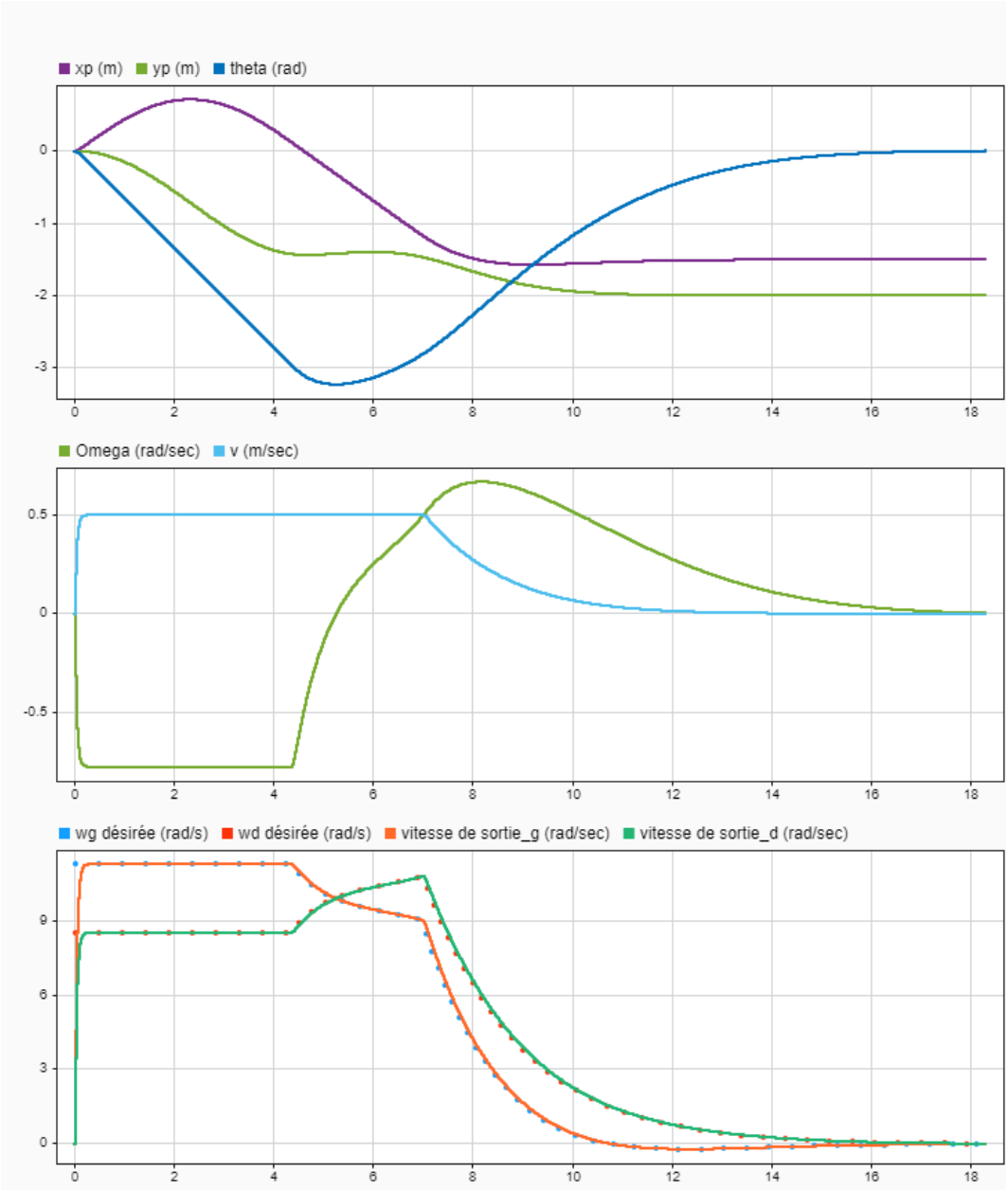


Figure 2.7: Vitesses linéaire et angulaire pour le cas en bas à gauche de la fig. 2.6

La partie haute de la figure ci-dessus montre, que pour cet exemple, le robot a pu atteindre la posture cible en un peu plus de 19 secondes. Pour d'autres situations cela peut être plus ou moins long, ceci dépend de l'instant où le robot est à moins de 0.1 mm de la position cible, bien que cela semble trop précis, cette valeur a été choisie afin de donner suffisamment de temps au robot pour qu'il puisse avoir un angle suffisamment proche de l'orientation ciblée, avant de s'arrêter.

Les graphes du milieu de la figure 2.7 montrent que les vitesses linéaire et angulaire ont été maintenues constantes pendant une certaine durée démarrant juste après le début du mouvement du robot. Cela est dû au fait que les lois de commandes (2.4) et (2.5) étant proportionnelles aux erreurs de positions et angulaire, celles-ci vont générer des commandes trop importantes durant cette phase. Pour remédier à cela nous avons introduit des saturations à ces deux signaux pour les maintenir à des valeurs acceptables qui sont de 0.5 mètre/seconde pour la vitesse linéaire et 0.77 rad/seconde pour la vitesse angulaire.

Les variations des vitesses de rotations des moteurs gauche et droite du robot sont tracées sur la partie basse de la figure 2.7, celles-ci montrent que les vitesses de consigne, en pointillés, calculées par le contrôleur cinématique sont très bien suivies par celles mesurées sur les axes des deux moteurs, cela montre que les contrôleurs PID précédemment conçus sont très efficaces.

2.2.5 Résultats des tests expérimentaux sur le robot réel

Dans le but de confirmer la validité des résultats obtenus en simulation nous allons reprendre les mêmes tests que précédemment pour les appliquer sur le robot réel. Pour cela le bloc contenant les deux boucles fermées de la commande dynamique sur la figure 2.5, est remplacé par la partie matérielle, composée d'une carte de commande embarquée sur le robot. Les données entrées/sorties, à savoir les vitesses désirées et celles mesurées, sont échangées avec le PC, grâce à une connexion série. L'algorithme de commande, qui s'exécute sur ce PC, calcule les vitesses de rotation des moteurs gauche et droite, celles-ci seront envoyées à la carte de commande, qui a pour tâche d'asservir ces vitesses avant de les appliquer aux deux moteurs pour donner au robot les mouvements adéquats. La figure 2.8 montre les résultats obtenus après avoir assigné au robot les mêmes postures que celles utilisées en simulation. Une comparaison visuelle des trajectoires obtenues en simulation et celles poursuivies par le robot réel montre qu'il y a une très grande similitude de ces trajectoires, ce qui démontre que les modèles utilisés en simulation sont une très bonne approximation du comportement du robot réel.

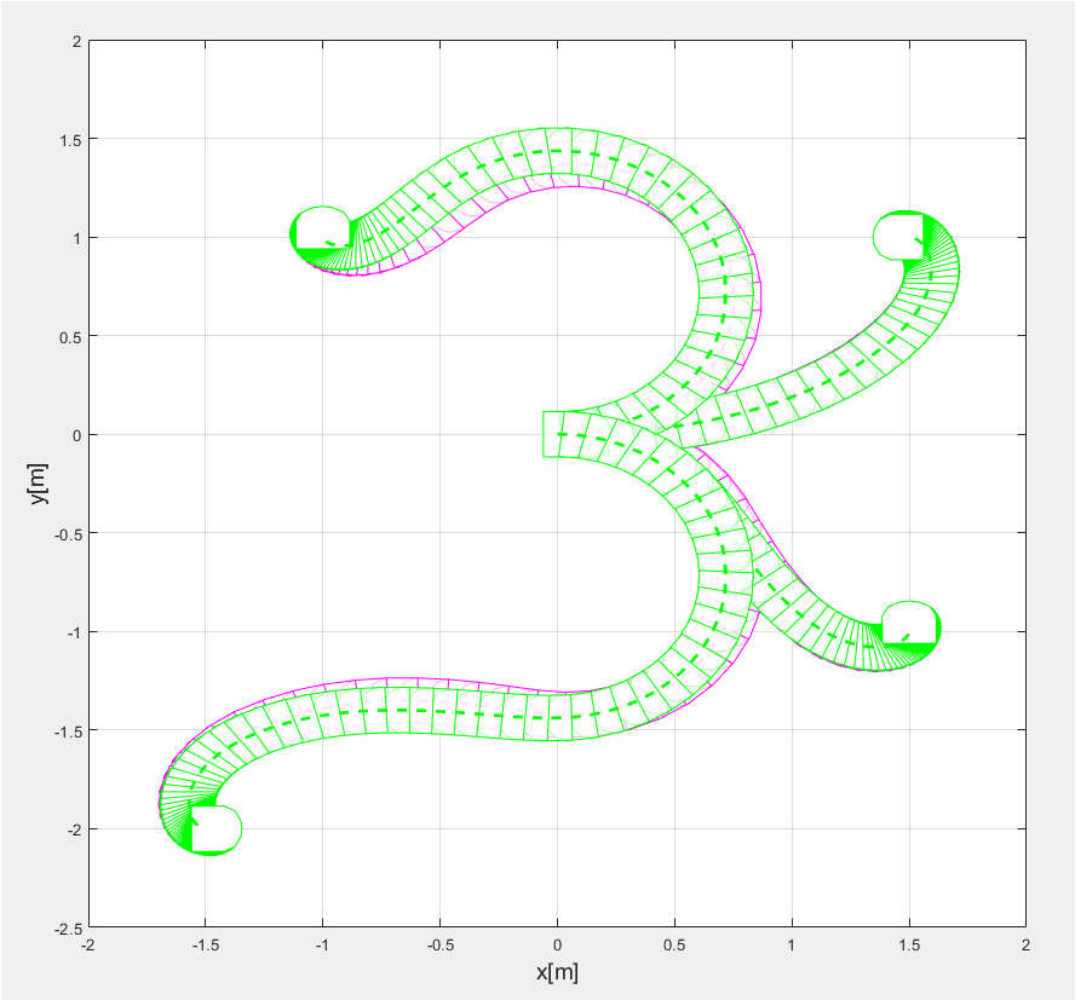


Figure 2.8 : Trajectoires du robot (magenta : robot réel, vert : simulation)

De même que pour les résultats des simulations précédentes, nous pouvons constater, sur la figure 2.9, une très grande ressemblance des graphes représentant les positions ainsi que vitesses linéaire et angulaire du robot. Avec les courbes des vitesses de rotations des moteurs gauche et droite, représentées en bas de la figure 2.9, nous pouvons constater l'apparition du signal nommé ‘’stop’’, qui monte à l'état haut après un peu moins de 19 secondes après de début du mouvement. Il s'agit d'un signal numérique à deux états, qui devient ‘’vrai’’ lorsque la position du robot est à moins de 0.1 mm de celle de la cible. Pour le cas des simulations précédentes ce même seuil a été choisi pour arrêter la simulation.

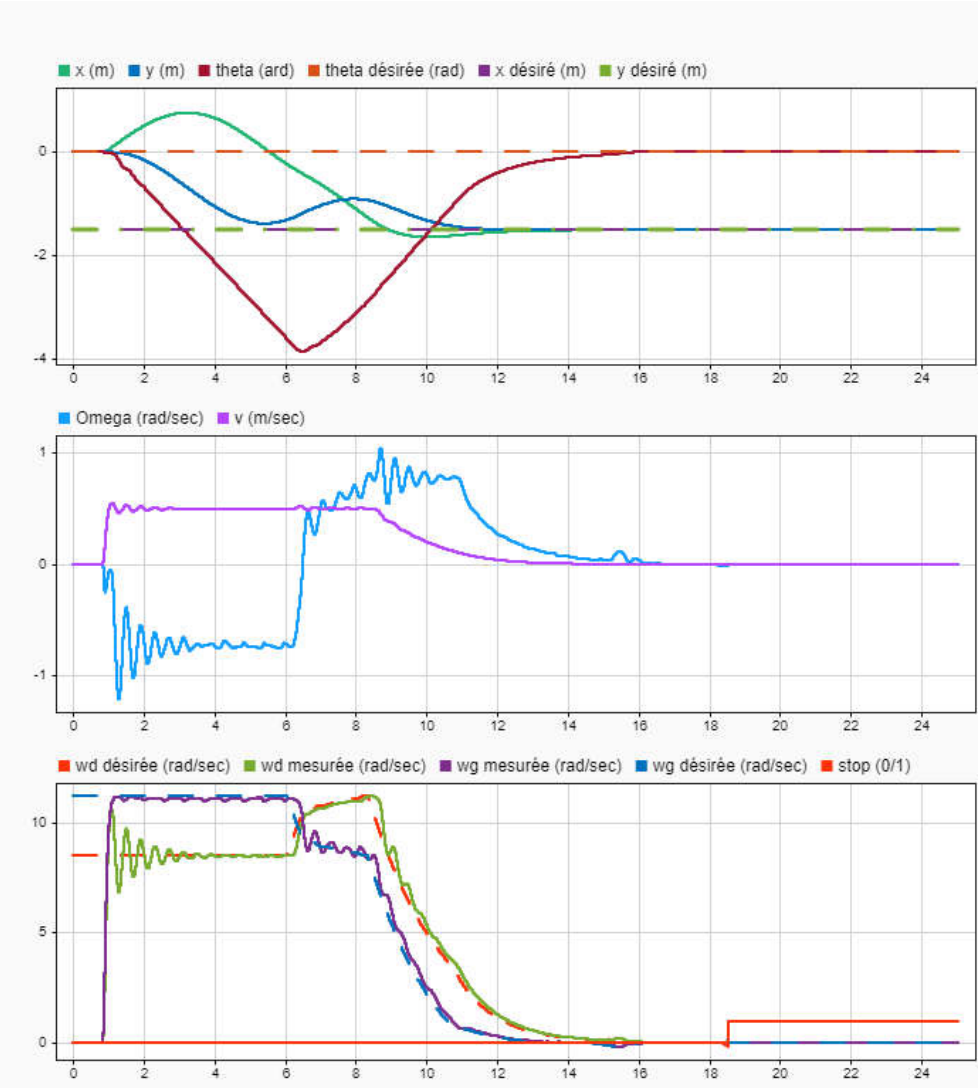


Figure 2.9 : Réponses temporelles des positions et vitesses du robot réel

Hormis les quelques perturbations observées sur les mesures des vitesses pendant l'exécution des mouvements du robot réel, nous pouvons observer une ressemblance évidente avec le cas de ce qui a été obtenu en simulation. Il faut toutefois signaler que pour obtenir un tel niveau de précision des positions et orientation d'un robot roulant sur le sol, la mécanique de ce dernier doit être mesurée avec un minimum d'erreurs surtout concernant le diamètre des deux roues ainsi que la distance L qui les sépare.

2.2.6 Commande non-linéaire

En se basant sur la même architecture illustrée par la figure 2.5 nous allons exploiter la théorie de Lyapunov pour concevoir une commande non-linéaire pour contrôler le robot dont les mouvements sont modélisés par le système d'équations d'état (2.2). Cette commande qui peut s'écrire sous la forme $u = f(v, \Omega)$ doit pouvoir diriger le robot de sorte à ce que les 3 variables d'état (ρ, β, α) convergent vers (0, 0, 0).

Pour y parvenir nous proposons la fonction de Lyapunov, positive définie, suivante :

$$V = \frac{1}{2}(\rho^2 + \alpha^2 + \beta^2) \tag{2.9}$$

Sa dérivée par rapport au temps est donc :

$$\dot{V} = \rho\dot{\rho} + \alpha\dot{\alpha} + \beta\dot{\beta} = \rho(-v\cos\alpha) + \alpha\left(\frac{v}{\rho}\sin\alpha - \Omega\right) + \beta(-\frac{v}{\rho}\sin\alpha)$$

Ou bien

$$\dot{V} = -v\rho\cos\alpha - \alpha\Omega + \alpha\frac{v}{\rho}\sin\alpha - \beta\frac{v}{\rho}\sin\alpha \tag{2.10}$$

Tel que mentionné précédemment, v et Ω sont les entrées de commande du système, pour ce qui concerne la vitesse linéaire v nous allons choisir une commande proportionnelle à la projection $\vec{v}_\rho$du vecteur $\vec{v}$ sur la droite reliant les points de départ P_1 à celui de l'arrivée P_2 tel mentionnés sur la figure 2.4. Avec k_ρ choisi comme constante de réglage celle-ci peut se mettre sous la forme :

$$v = k_\rho\rho\cos\alpha \tag{2.11}$$

Ce qui donne:

$$\dot{V} = -k_\rho\rho^2\cos\alpha^2 - \alpha\Omega + \alpha\frac{k_\rho\rho\cos\alpha}{\rho}\sin\alpha - \beta\frac{k_\rho\rho\cos\alpha}{\rho}\sin\alpha$$

$$\dot{V} = -k_\rho \rho^2 \cos\alpha^2 - \alpha\Omega + k_\rho \alpha \cos\alpha \sin\alpha - k_\rho \beta \cos\alpha \sin\alpha \qquad (2.12)$$

Si nous choisissons une autre constante positive k_α et l'expression de la vitesse angulaire Ω sous forme [1] :

$$\Omega = k_\alpha \alpha + k_\rho (\alpha - \beta) \cos\alpha \frac{\sin\alpha}{\alpha} \qquad (2.13)$$

Nous aurons:

$$\dot{V} = -k_\rho \rho^2 \cos\alpha^2 - k_\alpha \alpha^2 \qquad (2.14)$$

Avec les commandes (2.11) et (2.13) le système d'équations d'état (2.2) devient :

$$\begin{cases} \dot{\rho} = -k_\rho \rho \cos\alpha^2 \\ \dot{\beta} = -k_\rho \cos\alpha \sin\alpha \\ \dot{\alpha} = -k_\rho (\alpha - \beta) \cos\alpha \frac{\sin\alpha}{\alpha} - k_\alpha \alpha + k_\rho \cos\alpha \sin\alpha \end{cases} \qquad (2.15)$$

Selon la théorie de Lyapunov, si nous pouvons prouver que $\dot{V}$ est négative définie alors le système décrit par (2.15) est asymptotiquement stable autour de l'état donné par $(\rho, \alpha, \beta) = (0, 0, 0)$. A partir de (2.14) nous avons $\dot{V}(0, 0, \beta) = 0$, qui signifierait que $\dot{V}$ est négative semi-définie, nous pouvons aussi remarquer que lorsque $\alpha = 0$, $\frac{\sin\alpha}{\alpha} = 1$ et par conséquent la troisième équation dans (2.15) se réduit à $\dot{\alpha} = k_\rho \beta$, d'autre parts si $\rho = 0$ et $\alpha = 0$ cela voudrait dire que le système a atteint son état final ce qui implique que α reste constant et par conséquent $\dot{\alpha} = 0$, mais puisque $\dot{\alpha} = k_\rho \beta$ alors $\beta = 0$. Finalement nous pouvons conclure que $\dot{V} = 0$, uniquement lorsque $(\rho, \alpha, \beta) = (0, 0, 0)$, qui nous permet de confirmer qu'avec les lois de commande (2.11) et (2.13) ce système est effectivement asymptotiquement stable.

Avant de montrer les résultats des tests effectués, nous donnons, sur le tableau 2.3, les valeurs des constantes apparaissant dans les lois de commandes ci-dessus. Ces dernières ayant été obtenues après quelques essais-erreurs de ce contrôleur pour différents scénarios de cibles à atteindre.

Table 2.3 Paramètres du contrôleur non-linéaire

k_ρ	k_α
0.89	1.5

2.2.7 Résultats des tests en simulation de la commande non-linéaire

Afin de vérifier les performances de la commande non-linéaire proposée, et les comparer avec celles obtenues avec la commande linéaire, nous allons prendre les mêmes exemples, visant à faire atteindre au robot les mêmes postures cibles que précédemment. La figure 2.10 ci-dessous montre une nette amélioration des trajectoires parcourues par le robot et notamment lorsque les postures cibles sont derrières ce dernier. En effet nous constatons, que dans ces cas, la commande non-linéaire permet au robot d'effectuer un trajet plus court, lui permettant d'atteindre les positions et orientations désirées, cela étant dû au fait que le démarrage du robot s'effectue avec une vitesse linéaire négative, lui permettant de se rapprocher un peu plus de la cible. La figure 2.11, donne plus détails sur la manière avec laquelle les mouvements du robot évoluent dans le temps, pour le cas particulier où le robot est sollicité pour atteindre la posture (-1.5 m, -2.0 m, 0°). Sur cette même figure, nous avons aussi tracé les variations de la distance totale parcourue par le robot, qui est de 2.69 mètres, alors qu'elle était de 4.39 mètres pour le cas de la commande linéaire.

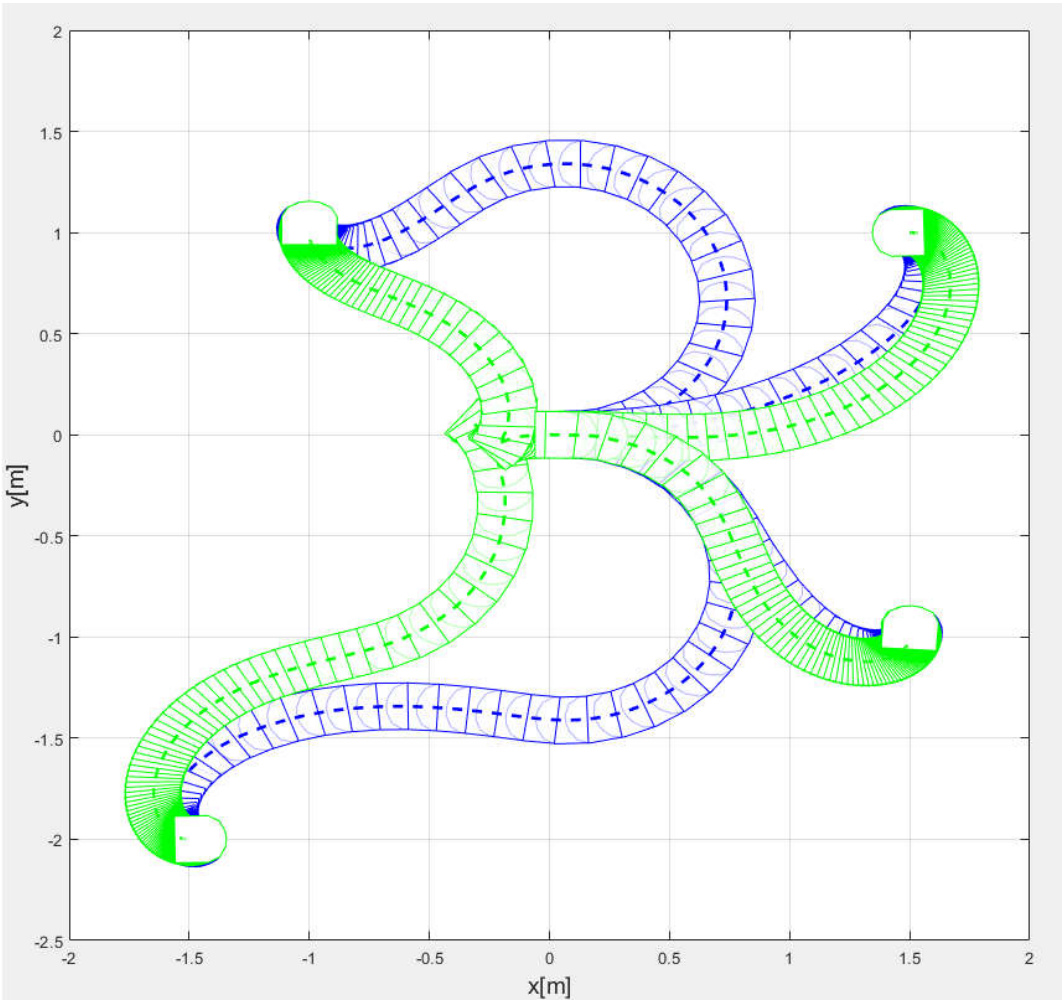


Figure 2.10 : Trajets du robot avec une commande linéaire (bleu) et non-linéaire (vert)

Bien qu'au niveau des variations de l'orientation du robot la tendance semble similaire pour les deux types de commande, nous constatons que pour le cas de la commande non linéaire la vitesse v, qui au départ est parfois négative, devient toujours positive lorsque le robot s'approche de sa position finale. Pour comprendre cela il faudra revenir sur l'équation (2.11) qui montre que v dépend de $cos\alpha$, sachant que l'angle α s'approche de zéro avant la fin du mouvements son cosinus est toujours positif. Par contre au début du mouvement, si la cible est à gauche de la position de départ du robot $cos\alpha$ est négatif puisque, dans ce cas, $\alpha \in \left[\frac{\pi}{2}, -\frac{\pi}{2}\right]$. Les 3 courbes de la partie inférieure de la figure 2.11 montrent que les 3 états (ρ, α, β) convergent tous vers (0, 0, 0), cela démontre bien que le contrôleur, ainsi défini,

entraine le système (2.15) vers un équilibre asymptotiquement stable, tel que mentionné précédemment.

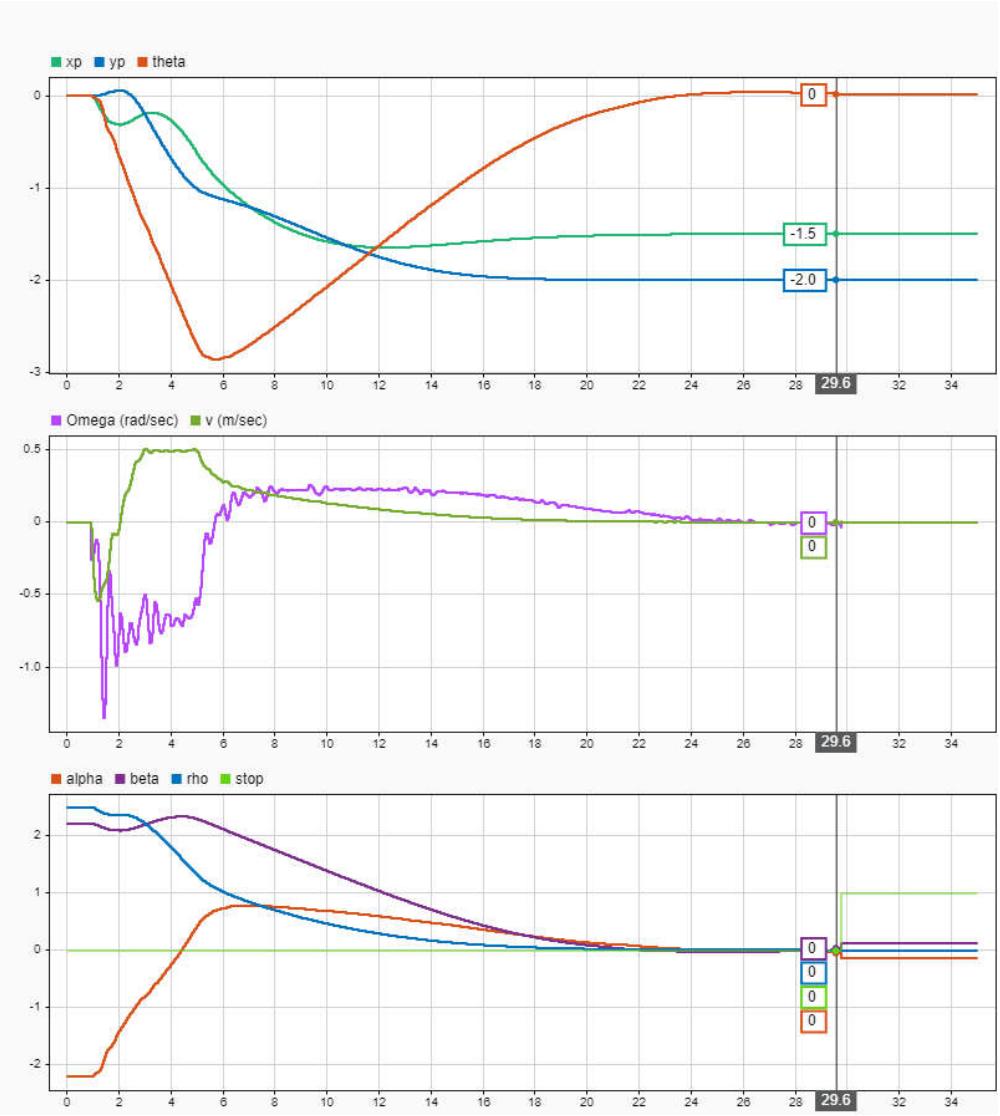


Figure 2.11 : Variations temporelles des paramètres du mouvements du robot

2.2.8 Résultats de la commande non-linéaire appliquée au robot réel

Afin de confirmer la validité des résultats de la commande non linéaire sur le robot réel nous lui avons appliqué les mêmes commandes que celles utilisées en simulation. La figure 2.12 ci-dessous montre une très grande similitude entre les trajectoires obtenues par le

contrôleur, lorsque celui-ci est appliqué aussi bien au robot réel, qu'à son image virtuelle représentée par le modèle développé précédemment.

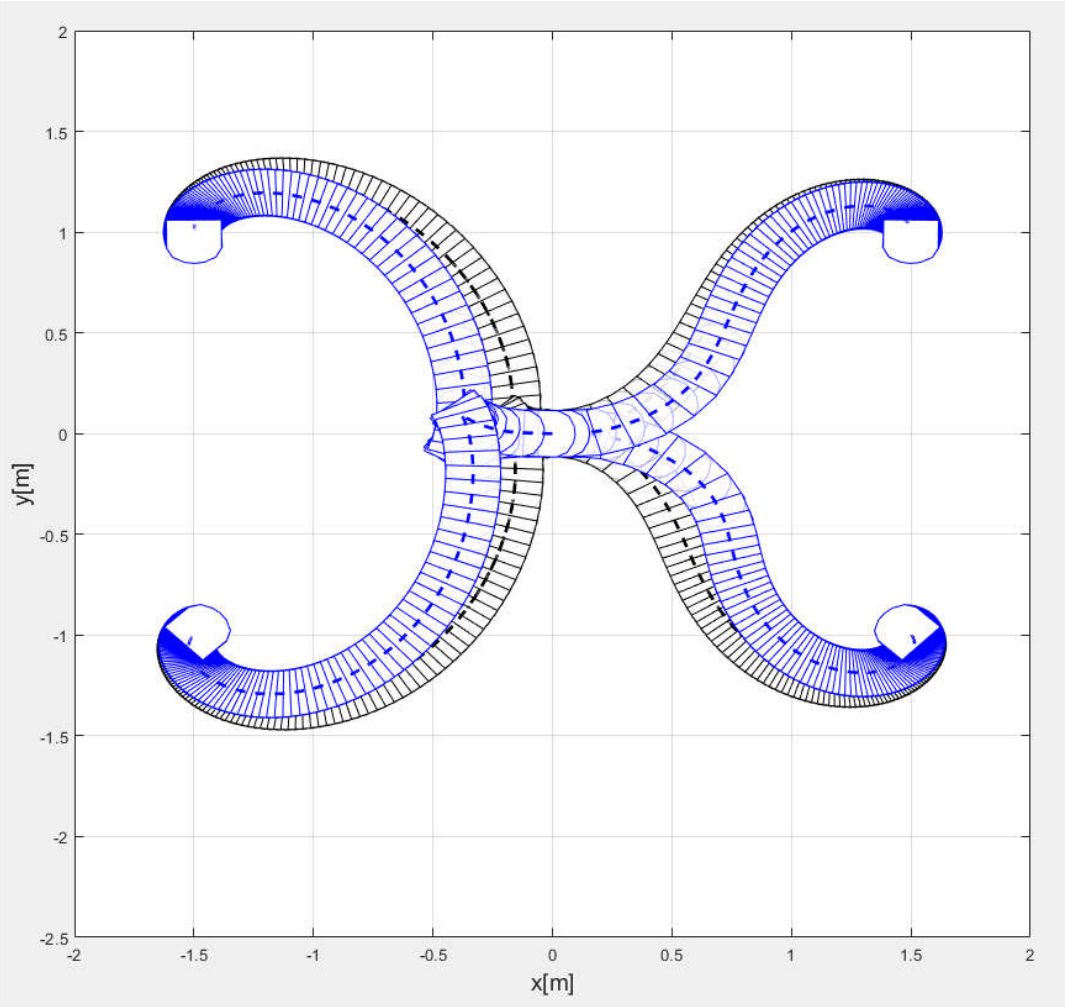


Figure 2.12 : Trajets du robot avec la commande non-linéaire (noir : modèle, bleu : réel)

Les tests expérimentaux précédemment cités ont tous été effectués avec le robot immobile, autrement dit, seules les deux roues étaient en rotation, sans être en contact avec le sol. Cette manière de procéder permet de réaliser différents types de scénarios sans trop de contraintes pratiques, telles que la nécessité de veiller à ce que la liaison filaire, avec le PC, soit suffisamment longue et ne gêne pas les mouvements du robot. La figure 2.13 montre que des données acquises pendant les mouvements du robot, sur le sol, sont très proches de celles mesurées lorsque ce dernier est immobile et que seules les deux roues sont mises en

rotation libre, soumises aux mêmes signaux de commandes, conduisant le robot à la posture cible P_2 = (-1.25 m, -0.75 m, 0°).

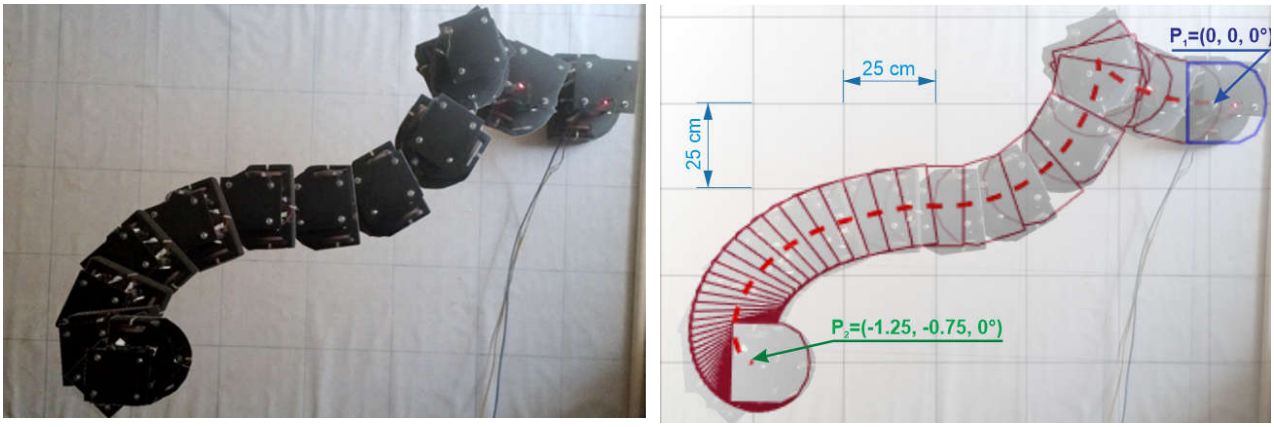


Figure 2.13 : Positions du robot (gauche), superposition avec positions en roues libres (droite)

Les graphes des mesures de vitesses et positions concernant les deux parties de la figure 2.13 sont montrées sur la figure 2.14, celles-ci confirment que le comportement du robot reste pratiquement inchangé pour les deux cas. Nous pouvons juste constater l'apparition de quelques perturbations ponctuelles, sur les signaux relatifs aux vitesses linéaires et angulaires mais qui sont sans influence sur le comportement global du robot.

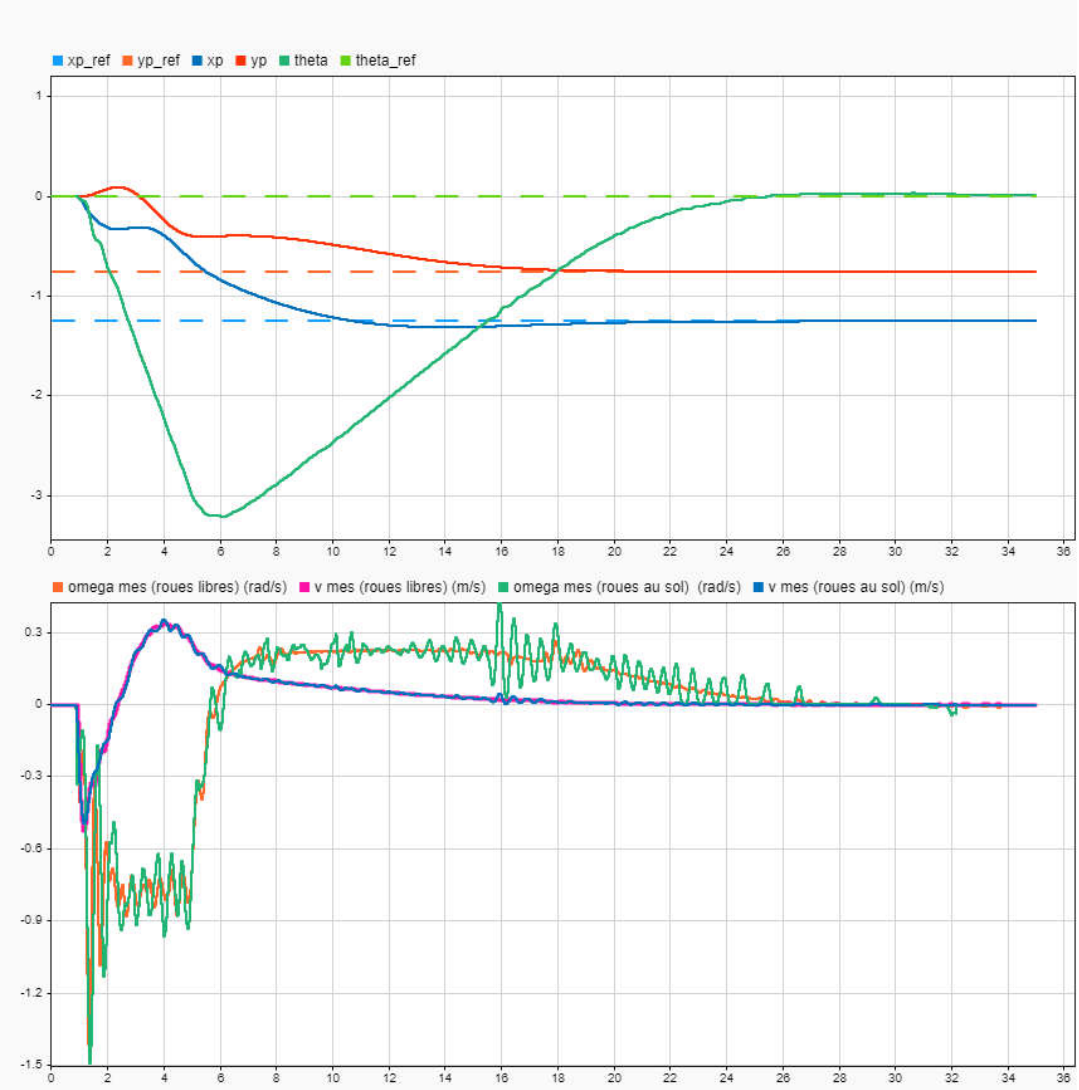


Figure 2.14 : Réponses temporelles avec (roues au sol : lignes, roues libres : pointillés)

2.3 Conclusion

L'essentiel de ce qui peut être retenu de ce chapitre est qu'une simple commande linéaire peut être utilisée pour conduire un robot mobile vers une posture donnée. Bien que celle-ci peut engendrer des trajectoires relativement longues, cela n'est pas tout à fait un inconvénient dans le cas où la distance séparant le point de départ à celui de l'arrivée est très importante. D'autres part les calculs nécessaires à son implémentation n'étant pas trop complexes, ils peuvent très bien être pris en charge par un simple microcontrôleur 8 bits. La

commande non-linéaire, quant à elle, génère des trajectoires plus ou moins optimales mais demande plus de ressources pour son implémentation étant donné qu'elle fait intervenir des calculs plus complexes et qui font appel à des fonctions trigonométriques. Pour ce qui concerne le cas présent, ce problème ne se pose pas puisque tous les calculs concernant ces commandes se font au niveau d'un ordinateur fixe, suffisamment puissant pour ne pas avoir à être confrontés ce genre de préoccupations.

Chapitre 3
Planifications de Trajectoires

Comme pour tout autre projet, avant d'entamer sa réalisation il faudra passer par une planification. Ce chapitre sera consacré à la description de quelques-unes des méthodes classiques de planification de trajectoires. Planifier une trajectoire consiste à identifier un chemin par lequel le robot doit passer pour atteindre sa position cible. Cette trajectoire doit tenir compte de la présence d'éventuels obstacles que le robot doit éviter d'une manière aussi efficace que possible, ceci, pour que le chemin suivi soit optimal en termes de distance parcourue.

3.1 Recherche de graphes

Connaissant la carte de son l'environnement le robot doit pouvoir construire un graphe construit à partir des connexions entre des nœuds et des sommets d'objets constituant l'environnement. Une génération de trajectoire est considérée comme « sure » si une solution existe, elle est « optimale » si elle aboutit à la meilleure solution.

3.1.1 Graphes de visibilité

La figure 3.1 représente une illustration de cette méthode ; Cette dernière consiste à relier tous les sommets visibles des objets présents dans l'environnement. La trajectoire que le robot va suivre sera le chemin le plus court reliant ses positions de départ et d'arrivée. C'est une solution optimale mais peut être très couteuse dans le cas où l'environnement est très complexe avec un nombre important d'objets. Elle peut aussi être dangereuse car le robot peut se trouver trop proche des objets de l'environnement.

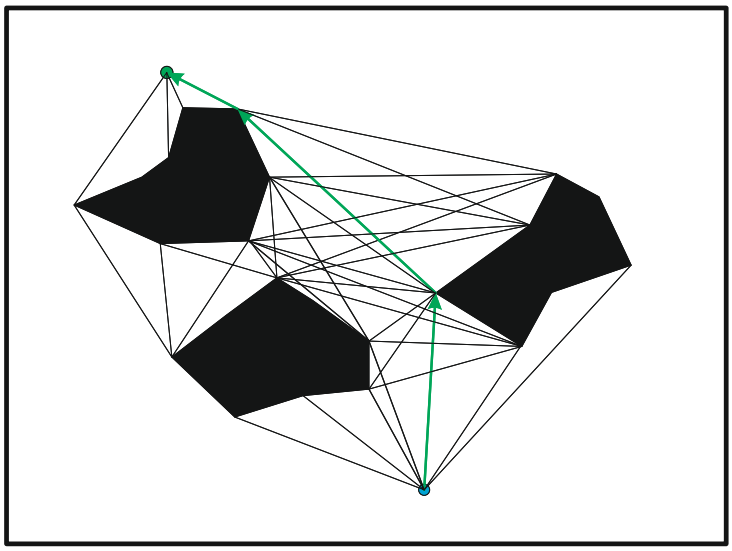


Figure 3.1 **:** Graphe de visibilité

3.1.2 Graphes de Voronoi

Cette méthode se base sur le principe de maximiser la distance entre le robot et les obstacles lors de son déplacement. Les possibilités de trajectoires sont calculées en traçant des lignes équidistantes entre les obstacles. La figure 3.2 montre un cas de figure illustrant cette méthode, le robot va suivre le chemin le plus court en choisissant des parties de ces lignes, l'existence d'une solution est garantie, cette méthode peut facilement être implémentée et permet aussi au robot de réaliser une cartographie de l'environnement, néanmoins elle n'est pas optimale et peut engendrer des problèmes d'ordre pratique, telle que la limitation des télémètres installés à bord du robot.

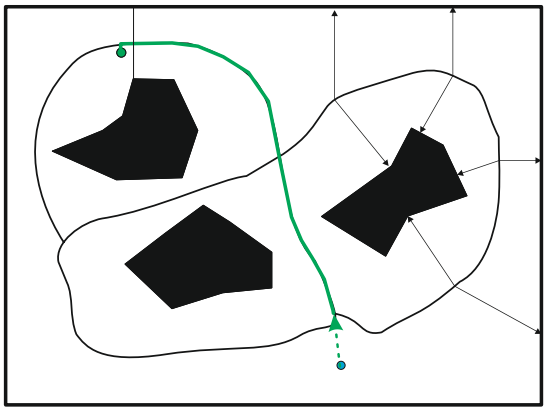


Figure 3.2 **:** Graphe de Voronoi

3.1.3 Décomposition en cellules

Cette approche consiste à diviser l'environnement en une grille de cellules fixes et de tailles égales. Ces cellules sont ensuite marquées comme occupées ou non selon qu'elles sont recouvertes par la présence d'un objet ou pas. La figure 3.3 reprend la configuration précédente et donne un exemple de partage de l'environnement en cellules de tailles égales. L'inconvénient de cette méthode provient de la granularité des contours des objets qui font perdre des espaces pour le déplacement du robot. L'avantage vient du fait de la simplicité de la planification de la trajectoire ; la figure 3.4 montre le cas d'utilisation de l'algorithme ''grassfire'' [36], ce dernier consiste à affecter un nombre à chaque cellule ; celles proches de la position de la cible ont les valeurs les plus faibles. Le déplacement du robot s'effectue en passant par les cellules avec des valeurs les plus en plus faibles, cette approche permet au robot d'établir une cartographie de son environnement grâces aux télémètres embarqués.

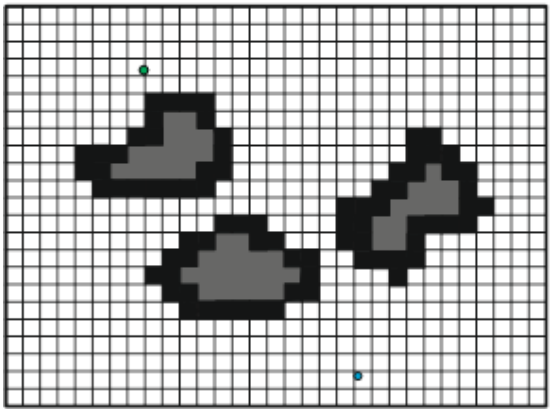


Figure 3.3 : Décomposition en cellules fixes

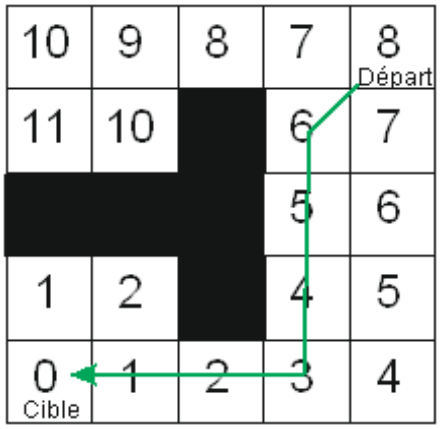


Figure 3.4 : Exemple de déplacement sur une grille d'occupation

3.2 Planification par champs de potentiel

Cette méthode s'appuie sur la création d'un champ de potentiel virtuel permettant au robot de s'approcher vers la cible en étant attiré par celle-ci et repoussé par les obstacles présents dans l'environnement. La figure 3.5 montre une représentation expliquant cette approche.

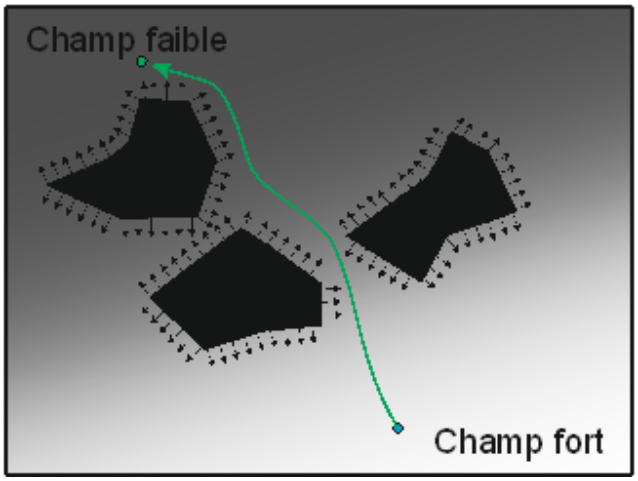


Figure 3.5 : Déplacement d'un robot à travers un champ de potentiel

3.3 Les algorithmes ''Bug''

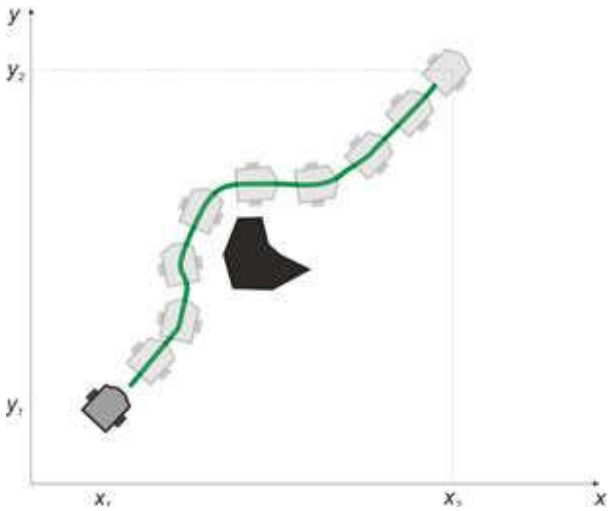


Figure 3.6 : Evitement d'obstacle typique

Ces algorithmes sont inspirés par le comportement des insectes lorsqu'ils se déplacent pour atteindre un objectif. Il existe plusieurs variantes ; connues sous l'appellation : *Bug 0*, *Bug 1*,

Bug 2, *DistBug*, *I-Bug*, *Tangent Bug*, … L'idée derrière ces méthodes est basée sur le suivi du périmètre de l'objet pour le contourner.

3.3.1 L'algorithme Bug 0

Cet algorithme représente la version la plus simple des algorithmes ''Bug''. La figure 3.7 explique le principe de cette idée. A chaque fois que le robot se trouve en face d'un obstacle il subit une rotation à gauche ou à droite pour contourner cet obstacle, dans le cas où il n'y a pas d'obstacle le robot s'oriente vers la position de la cible et avance en ligne droite.

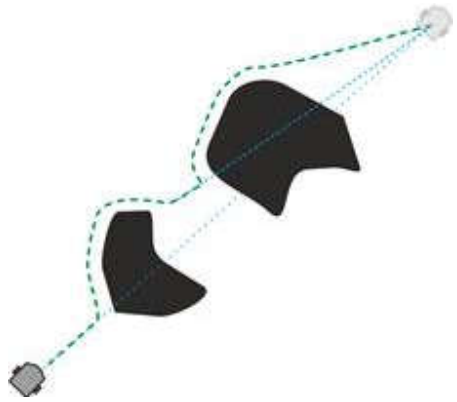


Figure 3.7 : Evitement d'obstacle basé sur l'algorithme Bug 0

Dans cette méthode il n'y a pas de mémorisation de trajectoire, c'est-à-dire que le robot ne retient pas l'historique de ses déplacements. La figure 3.8 montre une autre configuration de l'environnement dans lequel le robot va essayer d'atteindre sa cible avec le même algorithme.

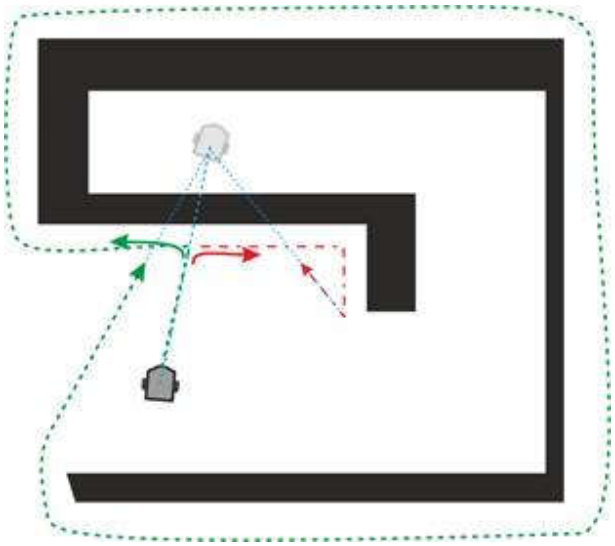


Figure 3.8 : ‘’Bug 0’’ ne donne pas de solution

Tel que le montre l’exemple de la figure 3.8 la méthode Bug 0 n’est pas sûre dans la mesure où elle ne garantit pas une solution pour certaines situations.

3.3.2 L’algorithme Bug 1

Dans cette méthode le robot commence par effectuer un tour complet de l’obstacle pour mémoriser le point le plus proche de la cible, par la suite et à partir de ce point, il se dirige vers cette cible, il reprend le même scénario pour chaque obstacle rencontré, la figure 3.9 illustre un exemple pour ce genre de situations. Cet algorithme assure qu’une solution existe pour n’importe quelle situation, néanmoins celle-ci n’est pas très efficace puisque le chemin parcouru risque d’être très long et donc prend beaucoup de temps avant que le robot atteint sa cible.

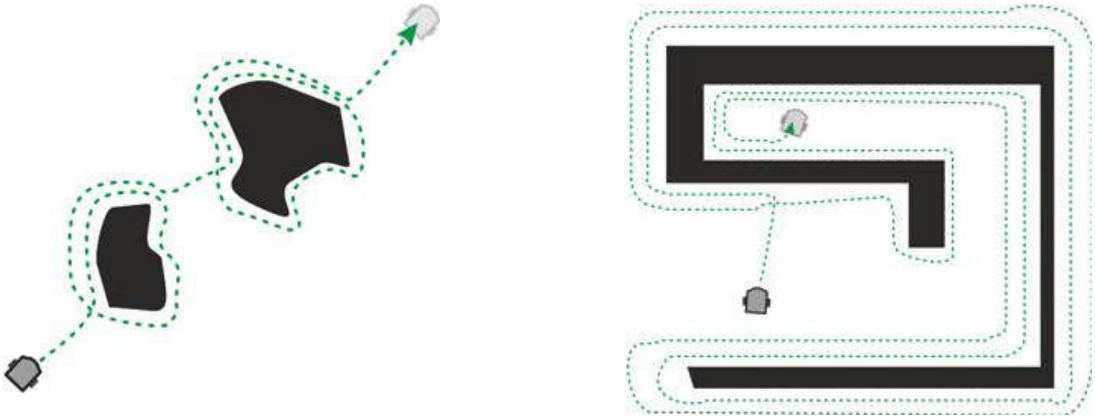


Figure 3.9 : Evitement d'obstacle basé l'algorithme Bug 1

3.3.3 L'algorithme Bug 2

Cette approche est une autre version qui consiste à faire en sorte que lorsqu'il n'y a pas d'obstacle en vue, le robot suit la ligne reliant le point de départ à la cible, en cas de présence d'obstacle le robot effectue un suivi de contour de la même manière que les deux cas précédents, la figure 3.10 montre deux exemples de parcours du robot utilisant cette méthode.

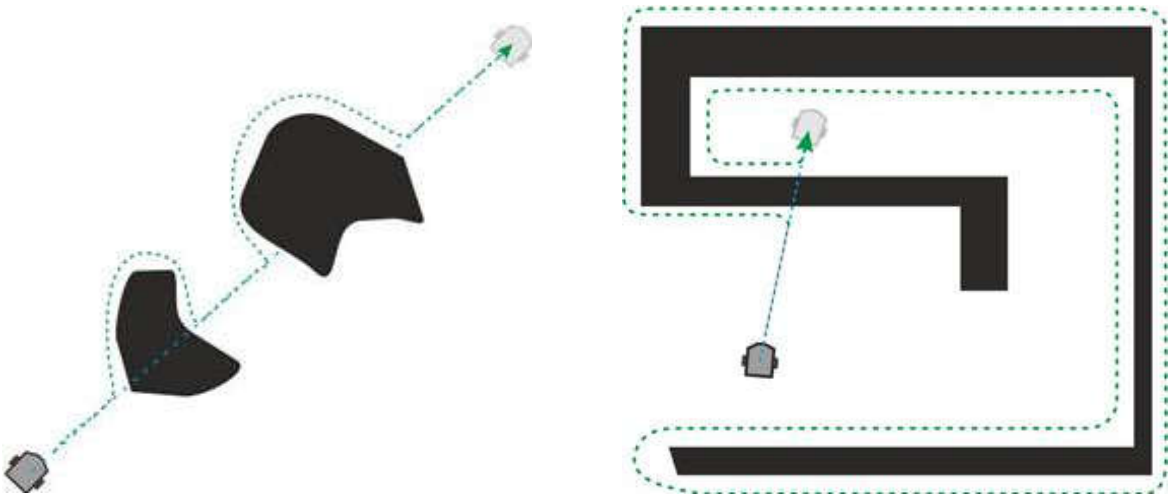


Figure 3.10 : Evitement d'obstacle basé l'algorithme Bug 2

Cet algorithme donne une solution sûre mais peut, dans certains cas, être moins efficace que celui basé sur Bug 1.

3.4 Histogramme des Champs de Forces Virtuelles (CFV)

L'un des problèmes des algorithmes ‘’*Bug’’* provient de leur dépendance exclusive des données récentes issues des télémètres, si ces données ne sont pas très fiables le comportement du robot en serait très affecté. La technique basée sur les CFV corrige cette limitation puisqu'elle se base sur la construction d'une carte locale de l'environnement autour du robot, cette carte consiste en une grille d'occupation telle que celle décrite au paragraphe 3.1.3, cette dernière est alimentée par les mesures récentes provenant des capteurs à bord du robot au fur et à mesure qu'il se déplace. Cet algorithme génère un histogramme polaire construit à partir des probabilités qu'un obstacle se situe à un angle donné, la figure 3.11 explique la manière de réaliser ce genre de graphes.

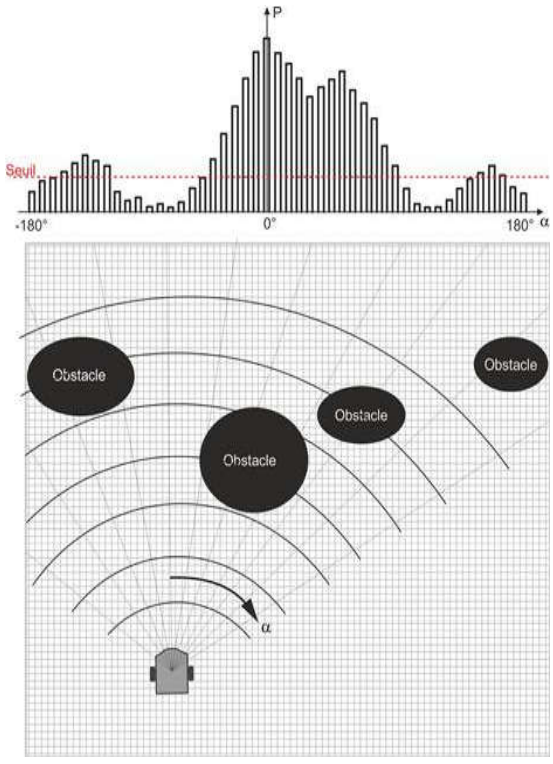


Figure 3.11 : Histogramme polaire

Afin d'effectuer des mouvements dans une direction donnée l'algorithme propose de réduire la carte de la figure 3.11 en une zone plus petite, à proximité du robot, de telle sorte que ce dernier puisse avancer vers sa cible sans percuter des obstacles, la figure 3.12 donne un aperçu sur cette étape.

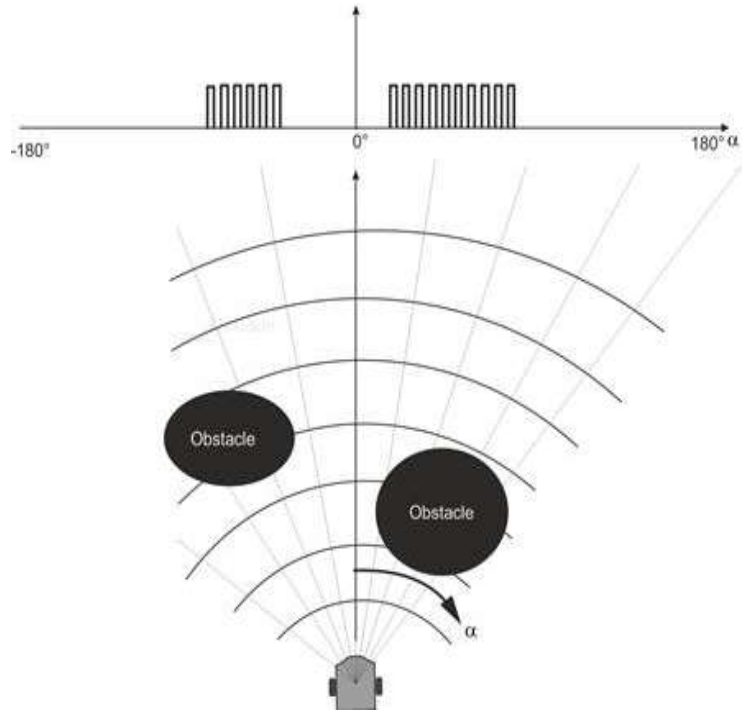


Figure 3.12 : Obstacles à proximité du robot

Avant d'atteindre sa cible le robot va effectuer des changements de directions en fonction de la disponibilité de passages plus ou moins larges entre les différents obstacles identifiés. Ces passages sont représentés par les zones où la probabilité de présence d'obstacles est inférieure à un certain seuil, le choix de la zone par laquelle le robot va passer sera déterminé en fonction de la taille des zones libres et l'orientation du robot avant le début de la manœuvre.

3.5 Recherche et planification de trajectoires

3.5.1 Recherche de trajectoire par expansion de nœuds

Considérons le cas de la figure 3.13 où le robot doit calculer sa trajectoire pour atteindre la position cible à partir de la position de départ. Dans ce cas le robot doit traverser le chemin décrit par 7 étapes avant d'arriver à la cible.

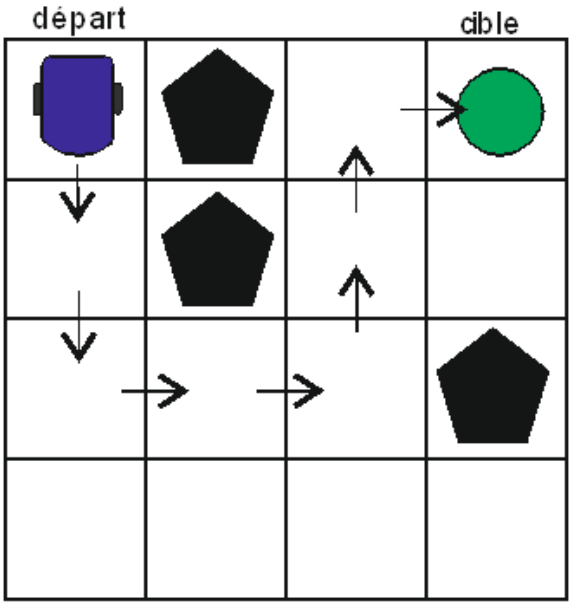


Figure 3.13 : Cas d'une trajectoire optimale

Afin d'arriver à un résultat optimal le robot doit utiliser une méthode pour décider des étapes à suivre l'une après l'autre. Pour cela considérons la situation illustrée par la figure 3.14. L'idée consiste à établir une liste de cellules ou nœuds à traverser d'une manière séquentielle. Le choix de la cellule suivante, adjacente à celle occupée par le robot, à un instant donné, s'effectue en tenant compte du fait que celle-ci doit être libre (pas d'obstacle) et qu'elle n'ait pas déjà été visitée par le robot. Dans le cas où plusieurs possibilités existent le robot doit choisir la cellule qui va le conduire à effectuer le minimum d'étapes durant ses déplacements.

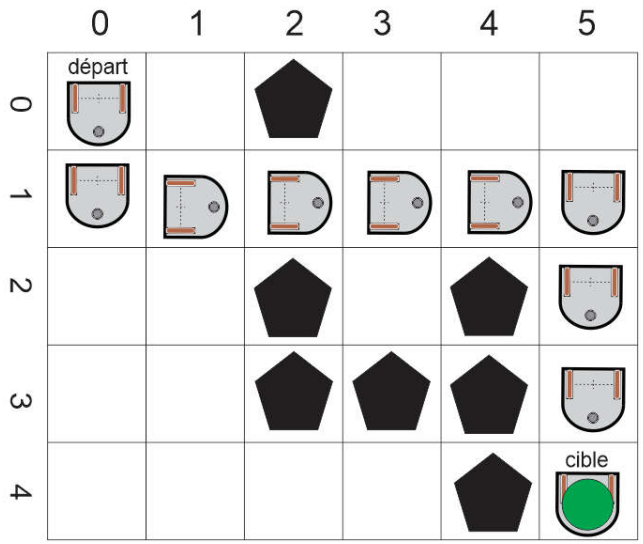


Figure 3.14 : Recherche par expansion de cellules

Ainsi l'expansion des cellules peut s'effectuer de la manière suivante ;

[1, 0, 1] [0, 1, 1] [1, 1, 2] [2, 0, 2] [2, 1, 3] [1, 2, 3] [3, 0, 3] [1, 3, 4] [3, 1, 4] [4, 0, 4] [0, 3, 5]
[2, 3, 5] [1, 4, 5] [4, 1, 5] [0, 4, 6] [1, 5, 6] [4, 2, 6] [0, 5, 7] [2, 5, 7] [4, 3, 7] [3, 5, 8] [4, 5, 9]

Ceci est obtenu en adoptant la notation (x, y, Nombre d'étapes) :

0,0,**0**	0,1,**1**		0,3,**5**	0,4,**6**	0,5,**7**
1,0,**1**	1,1,**2**	1,2,**3**	1,3,**4**	1,4,**5**	1,5,**6**
2,0,**2**	2,1,**3**		2,3,**5**		2,5 ,**7**
3,0,**3**	3,1,**4**				3,5,**8**
4,0,**4**	4,1,**5**	4,2,**6**	4,3,**7**		4,5,**9**

Figure 3.15 : Résultat de la recherche par expansion de cellules

Avec cette approche le robot va donc parcourir le chemin le plus court (cellules vertes).

3.5.2 L'Algorithme A*

Cette méthode est connue pour être la plus efficace dans le domaine de la planification de trajectoire en robotique. Il s'agit d'une extension de l'approche précédente dans le sens où l'expansion des cellules sera réduite afin de minimiser le nombre d'étapes que doit parcourir le robot. Pour cela une fonction ''heuristique'' est définie comme contrainte supplémentaire permettant cette minimisation. Cette fonction peut, par exemple, être choisie comme étant la distance séparant le robot de la position cible. La figure 3.16 illustre un exemple de choix de cette fonction heuristique. Sans tenir compte des obstacles cette fonction doit vérifier la condition suivante :

$$h(x,y) \leq distance\ de\ la\ cible$$

La différence essentielle de l'algorithme A* par rapport à celle basée sur l'expansion de cellules est que la minimisation s'applique sur une fonction $f = N + h(x,y)$.

9	8	7	6	5	4
8	7	6	5	4	3
7	6	5	4	3	2
6	5	4	3	2	1
5	4	3	2	1	0

Figure 3.16 : Exemple de fonction heuristique

La figure 3.17 montre l'efficacité de l'algorithme A*, le nombre d'étapes effectuées est très réduit du fait qu'un grand nombre de cellules n'ont pas été visitées. Sur un total de 30 cellules, 5 sont occupées par des obstacles. Parmi des 25 cellules restantes, le trajet du robot comprend uniquement les 12 étapes mentionnées sur cette figure.

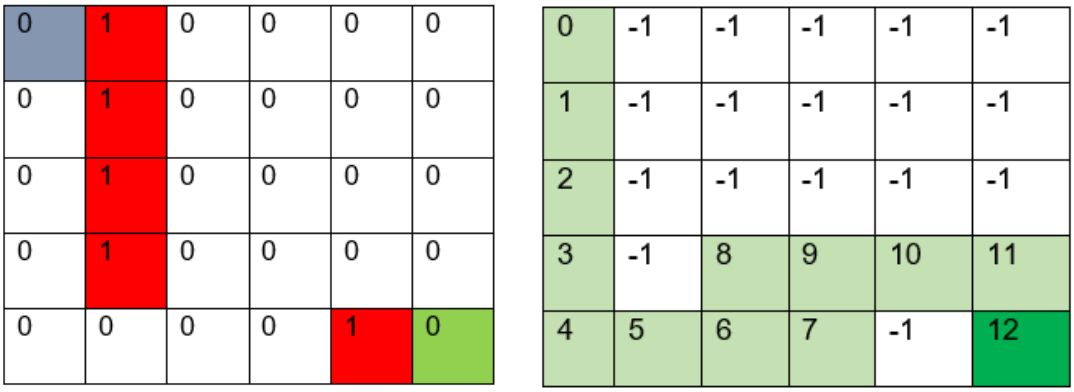


Figure 3.17 : Résultat obtenu avec A*

3.5.3 L'Algorithme Rapidly exploring Random Tree (RRT) [34]

Cette approche est basée sur la sélection d'un certain nombre de points par lesquels le robot va passer pour atteindre la position cible choisie. L'ensemble de ces points est défini par une arborescence grandissante d'une manière aléatoire, tout en évitant les zones occupées par des obstacles. Le départ d'une branche donnée se situe sur celle qui la précède à partir d'une distance minimale d préalablement choisie. La sélection des positions intermédiaires définissant le chemin final reliant la position initiale du robot à sa position cible est réalisée en effectuant le chemin inverse ; c'est-à-dire que pour une position i donnée le choix de la position i-1 se fait en tenant compte du fait que cette dernière est celle à partir de laquelle la position i a été ajoutée. La figure 3.18 illustre un exemple de chemin obtenu grâce à cette approche.

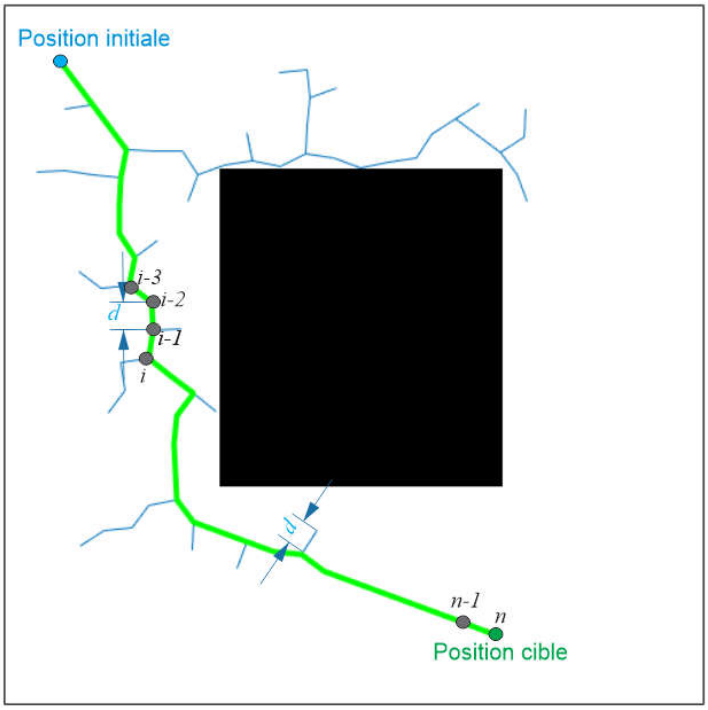


Figure 3.18 : Exemple de chemin obtenu avec la méthode RRT

3.5.4 L'Algorithme Probabilistic RoadMap (PRM) [33]

La planification basée sur cette approche se compose de deux phases : Une phase de construction suivie par une phase de recherche du chemin liant les positions de départ et d'arrivée du robot. Durant la phase de construction l'algorithme génère une feuille de route approximative donnant un maximum de possibilités de déplacements que le robot pourrait effectuer entre un ensemble de positions sélectionnées aléatoirement à l'intérieure des zones non-occupées par des obstacles. Une fois avoir obtenu une feuille de routes suffisamment dense et couvrant un maximum de l'espace libre, l'algorithme utilise une méthode de recherche du chemin le plus court, parmi toutes les possibilités résultant de ce maillage, telle que celle proposée dans [32]. La figure 3.19 montre un exemple de parcours obtenu en utilisant cet algorithme.

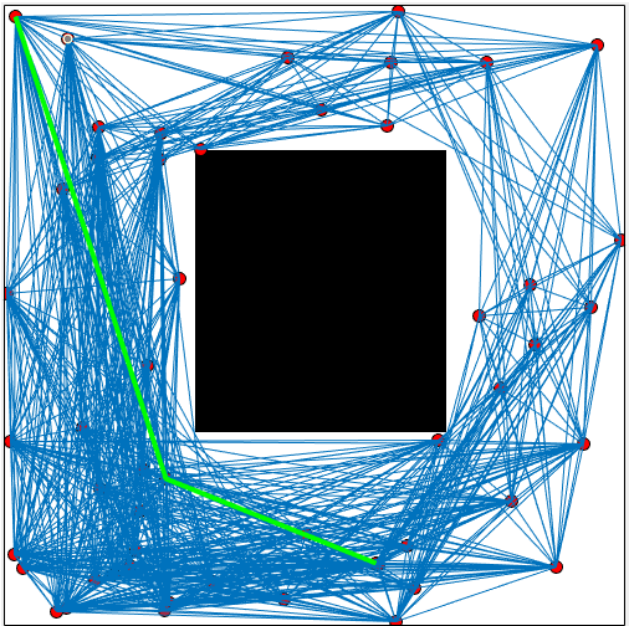


Figure 3.19 : Exemple de chemin obtenu avec la méthode PRM

3.6 Conclusion

Ce chapitre a été consacré à la présentation de quelques-unes des méthodes classiques proposées pour planifier une trajectoire d'un robot mobile évoluant dans environnement statique et contenant d'éventuels obstacles. Dans cette présentation nous avons donné uniquement les principes de base sur lesquelles ces méthodes ont été développées. Plusieurs autres améliorations ainsi que de nouvelles approches [35] ont été développées durant ces dernières décennies dans un seul but, celui d'obtenir une trajectoire plus courte et plus rapide.

Chapitre 4
Suivis de trajectoires prédéfinies

Parmi les nombreuses applications des robots mobiles, le suivi de trajectoires s'avère être l'un des problèmes majeurs et notamment dans des domaines impliquant le déplacement de différents types d'objets ou de personnes. Dans le cas où le robot doit suivre un chemin déjà connu dans le repère de référence, et est capable de régler sa propre vitesse, tout en restant sur ce chemin, la stratégie de commande ainsi définie est connue sous l'appellation ''suivi de chemin''. Cependant, si chacune des positions successives du robot doit être connue à chaque instant du parcourt, le type de mouvements, dans ce cas, est qualifié de ''suivi de trajectoire''.

Dans ce chapitre nous allons, dans un premier temps, décrire l'une des méthodes classiques relative au suivi de trajectoire, à savoir celle dénommée ''pure poursuit'', le but étant de comparer les performances de cette dernière avec les méthodes proposées dans notre travail. Les paragraphes qui suivront seront consacrés à la présentation des lois de commande étudiées précédemment adaptées au problème de suivi de trajectoires prédéfinies. En effet, considérant qu'une trajectoire est composée d'un ensemble de points consécutifs, l'application directe de ces commandes aboutit à l'une des solutions à ce problème.

4.1 La méthode ''pure pursuit''

Le principe de base sur lequel repose cette méthode consiste à calculer le rayon de courbure d'un arc de cercle dont l'une des deux extrémités coïncide avec le point de la trajectoire que le robot doit atteindre en suivant cet arc de cercle. La figure 4.1 montre les éléments nécessaires au calcul du rayon **R** connaissant la distance d'anticipation $\mathbf{L_D}$ et les coordonnées du point **T** dans le repère (x_R, y_R), attaché au robot mobile. Ainsi nous avons :

$$L_D^2 = x_T^2 + y_T^2 \qquad (4.1)$$

$$R^2 = d^2 + x_T^2 \tag{4.2}$$

$$R = d + y_T \tag{4.3}$$

Ce qui donne:

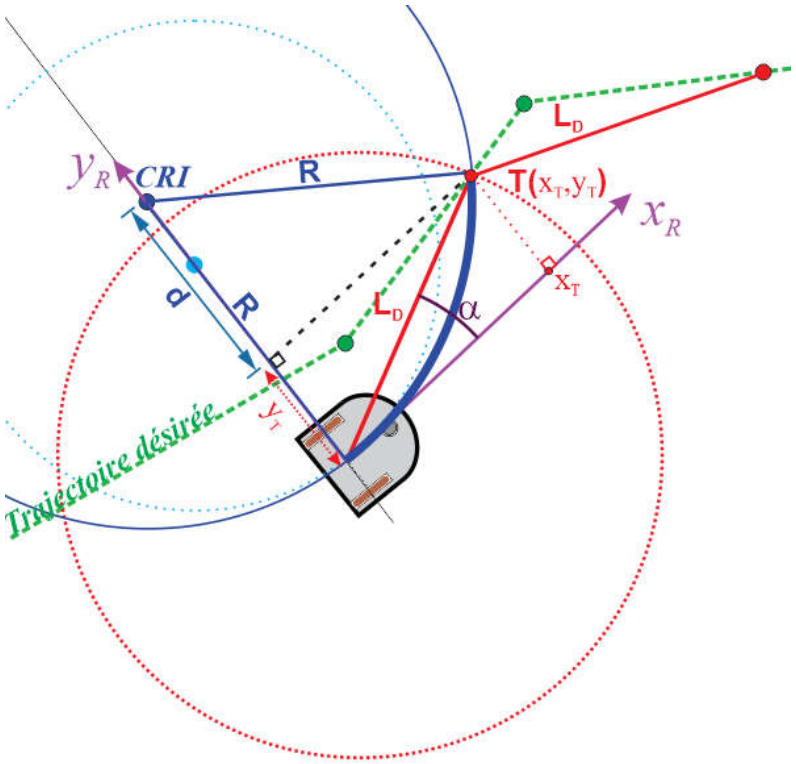


Figure 4.1 : Illustration de l'approche ''pure pursuit''

$$d = R - y_T$$

Donc

$$R^2 = (R - y_T)^2 + x_T^2 = R^2 - 2Ry_T + y_T^2 + x_T^2$$

En utilisant (4.1) nous aurons:

$$R^2 = R^2 - 2Ry_T + L_D^2$$

Par conséquent

$$R = \frac{L_D^2}{2y_T} \tag{4.4}$$

Pour s'orienter vers le point **T** appartenant à la trajectoire cible, le robot doit donc subir une rotation d'un angle α. Ce dernier peut se calculer de la manière suivante :

$$\sin \alpha = \frac{y_T}{L_D} \tag{4.5}$$

Dans cette approche, le but est de contrôler le mouvement angulaire en gardant une vitesse longitudinale constante, il est donc préférable de ne pas effectuer des changements angulaires brusques. De ce fait les variations de l'angle α doivent être faibles et par conséquent nous pouvons considérer que $\sin \alpha \approx \alpha$, c'est-à-dire :

$$\alpha = \frac{y_T}{L_D} \tag{4.6}$$

Ou bien $y_T = \alpha L_D$, en remplaçant dans (4.4) nous aurons :

$$\alpha = \frac{L_D}{2R} \tag{4.7}$$

Nous constatons ainsi que les changements d'orientation dépendent directement de la distance $\mathbf{L_D}$ (distance d'anticipation) dont la valeur est choisie par l'utilisateur, ceci en fonction du type de trajectoire et aussi de la taille du robot, ainsi que les vitesses pouvant être supportées par le robot mobile. Une valeur trop faible de $\mathbf{L_D}$ permettrait d'obtenir une réponse rapide, où le robot rejoint rapidement la trajectoire, par contre cela peut provoquer une instabilité du mouvement. Une valeur trop grande de ce paramètre donnerait un mouvement plus fluide, sans oscillations, mais peut considérablement retarder la convergence du robot vers la trajectoire désirée.

4.2 Commandes linéaires appliquées aux suivis de trajectoires

A partir de la figure 2.4 nous pouvons constater qu'en annulant l'angle β nous aurons un robot dont l'orientation dépendra directement de l'angle α, qu'il faudra convenablement contrôler pour faire avancer le robot vers le point se situant sur la position suivante de la trajectoire. La commande pour la vitesse linéaire, quant à elle, reste inchangée et dépendra uniquement de la distance séparant deux points consécutifs sur la même trajectoire. Les lois de commandes linéaires proposées se réduisent donc à l'équation (2.4) pour la vitesse linéaire, et (4.8) pour la vitesse angulaire.

$$\Omega = k_\alpha \alpha \tag{4.8}$$

Ainsi donc, l'architecture du système global représentée par la figure 2.5 reste inchangée et sera donc utilisée pour l'implémentation de cette commande pour faire suivre le robot une trajectoire prédéfinie. Afin de pouvoir juger de l'efficacité des commandes proposées, nous allons choisir des trajectoires avec des courbures plus ou moins accentuées avec des points de départ et d'arrivée placés sur différentes zones d'un plan de repère fixe, définissant les coordonnées cartésiennes ainsi que l'orientation du robot par rapport à ce plan.

4.2.1 Trajectoires de formes circulaires

Le premier exemple consiste à générer une trajectoire dont le point de départ se situe sur les parties négatives des axes (x, y) du repère fixe et dont la forme est construite de sorte que la fréquence de variation du déplacement sur l'axe x et deux fois supérieure à celle des variations sur l'axe y, ce genre de courbes est connu sous l'appellation ''courbes de Lissajous''. Les équations paramétriques de cette trajectoire sont données par:

$$\begin{cases} x(t) = 2\sin\dfrac{2\pi}{50}t - 5 \\ y(t) = 2\cos\dfrac{2\pi}{100}t - 2 \end{cases} \tag{4.9}$$

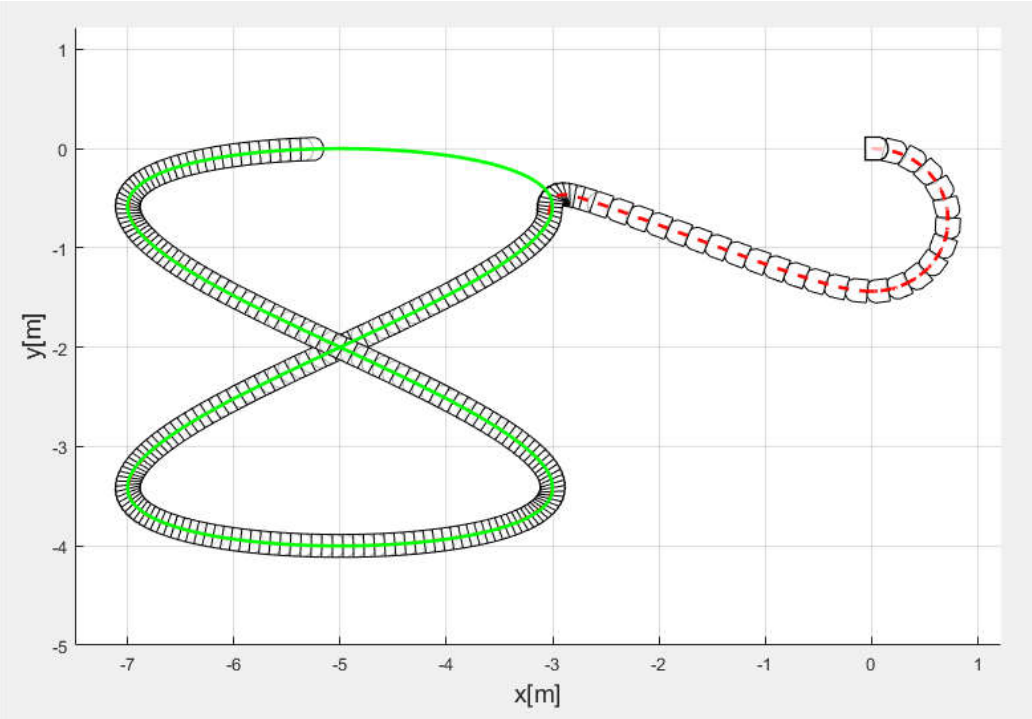


Figure 4.2 : Suivi d'une trajectoire périodique basé sur la commande linéaire

A partir de la figure ci-dessus nous pouvons constater qu'après avoir atteint de point le plus proche, situé sur la trajectoire, le robot est resté sur cette même trajectoire tout le long de son parcours. En effet et malgré sa simplicité, cette commande a donné de très bons résultats comparés à certains travaux récents ayant des objectifs similaires mais utilisant des lois de commandes non-linéaires bien plus compliquées telles que celles basées sur la théorie de Lyapunov [2-3], ou encore celle utilisant des réseaux de neurones [9] ainsi qu'une autre basée sur la commande en mode glissant [11]. Les positions successives du robot sont pratiquement confondues avec celles de la trajectoire prédéfinie, représentée par le tracé de couleur verte. Ces observations sont valables aussi bien pour le cas où les commandes sont appliquées au robot virtuel, simulé par les deux fonctions de transfert des deux moteurs, que pour le cas du robot réel, connecté au PC sur lequel ces mêmes commandes sont implémentées.

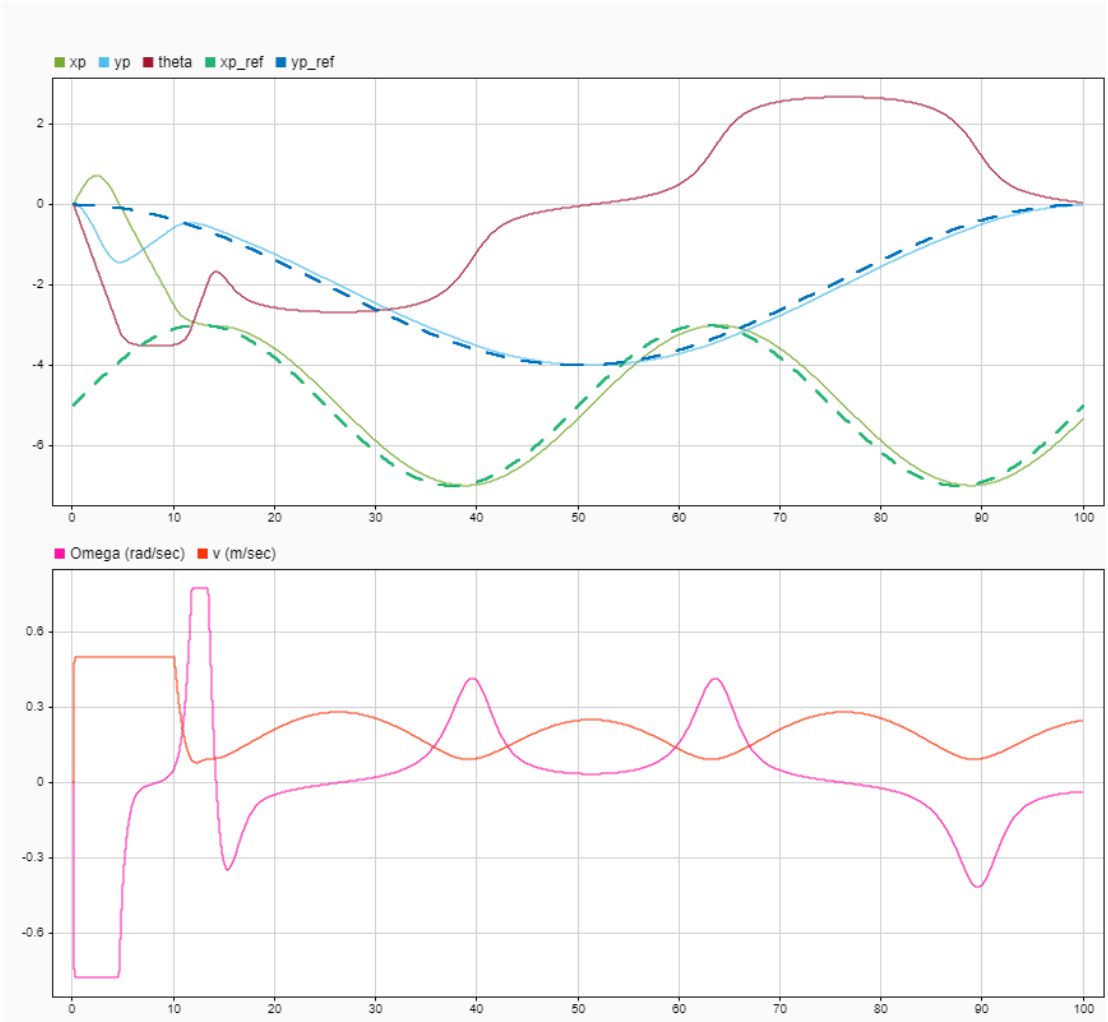


Figure 4.3 : Données mesurées durant les mouvements du robot réel poursuivant la trajectoire définie par les équations (4.9)

Pour une meilleure interprétation du comportement du robot lors des mouvements précédents, les variations temporelles des paramètres essentiels sont représentées sur la figure 4.3. Nous pouvons vérifier, sur la partie supérieure de cette figure, qu'une fois avoir rejoint la trajectoire désirée, les positions du robot sur les axes x et y, du repère fixe, sont conformes aux équations paramétriques (4.9). En effet, les projections des positions du robot sur les axes x et y ont bien une forme sinusoïdale, d'amplitude 2 mètres, avec une période de 50 secondes sur l'axe x, et 100 secondes sur l'axe y. La partie inférieure de cette figure montre que les vitesses linéaire et angulaire varient constamment et sont en opposition de phase durant les intervalles de temps où les courbures de la trajectoire sont relativement

accentuées. Cela peut s'expliquer par le fait que pour réussir à garder son cap sur cette trajectoire, le robot doit augmenter sa vitesse de rotation tout en réduisant sa vitesse de translation. Nous pouvons aussi observer que, pour les parties de la trajectoire où les courbures sont moindres la vitesse linéaire est à son maximum, puisque son l'orientation est constante et donc la vitesse de rotation est proche de zéro.

4.2.2 Trajectoires de forme rectangulaires

Les observations citées précédemment sont encore plus visibles sur les figures 4.4 et 4.5 qui montrent le comportement du robot réel poursuivant une trajectoire de forme carrée. Cette trajectoire a été générée en définissant les quatre sommets du carré comme points intermédiaires, comme cela est illustré par le tracé de couleur verte sur la figure 4.4. Une comparaison des deux trajectoires effectuées par le robot pour deux vitesses de déplacement différentes montrent qu'une meilleure poursuite est obtenue lorsque le robot se déplace plus lentement. Dans le cas où nous voulons obtenir un suivi rapide tout en restant très proche de la trajectoire il faudrait partager cette trajectoire en plusieurs parties tout en imposant des intervalles de temps adéquats, qui dépendent de la courbure de la portion de trajectoire concernée.

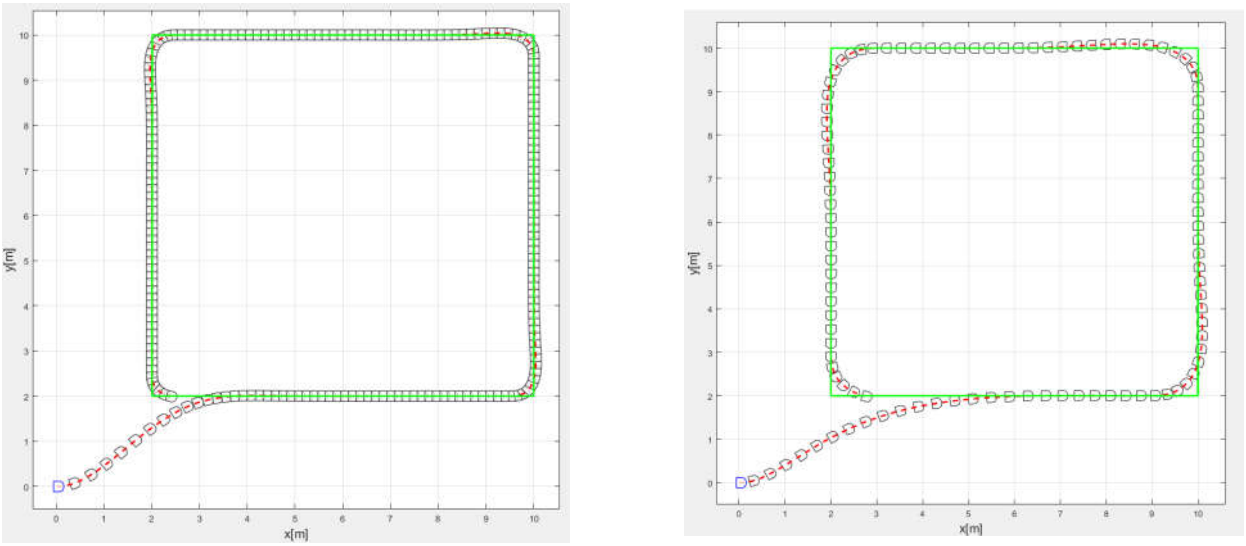


Figure 4.4 : Trajectoires du robot réel poursuivant une trajectoire carrée (Gauche : v = 0.2 m/sec, Droite : v =0.4 m/sec)

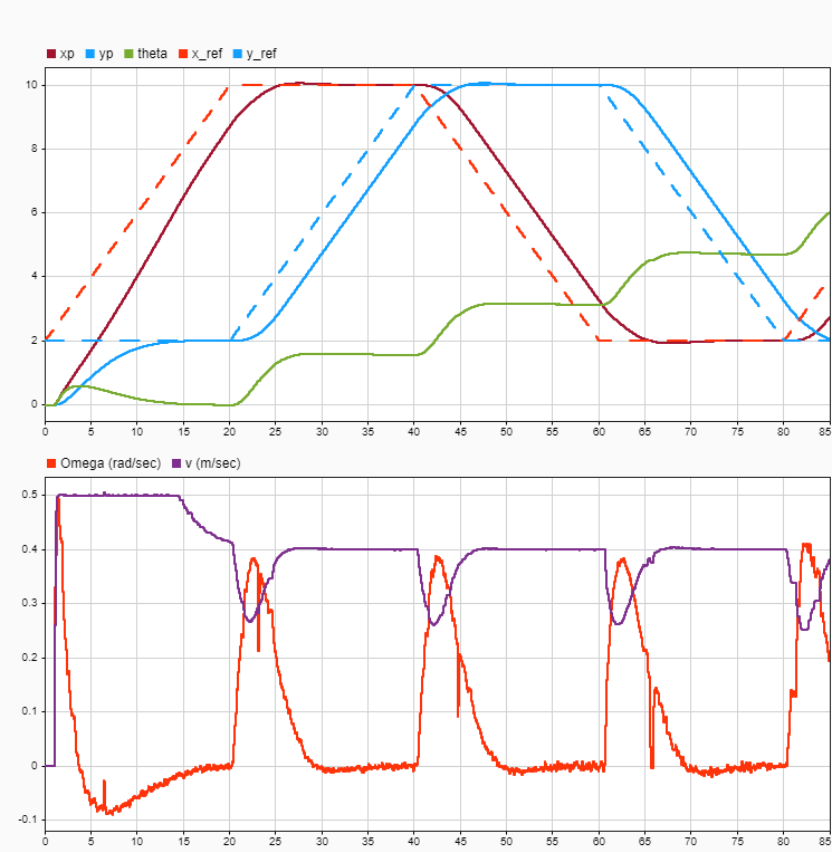


Figure 4.5 : Données temporelles du suivi de la trajectoire carrée avec v = 0.4 m/sec

4.3 Commandes non linéaires appliquées aux suivis de trajectoires

L'application de la commande non linéaire pour faire atteindre le robot une posture désirée, présentée au chapitre 2, peut aisément être adaptée à la poursuite d'une trajectoire préalablement définie. Pour cela il suffit de modifier l'équation (2.13) en annulant l'angle β qui servait à orienter le robot vers la direction finale choisie. Dans le cas présent ceci n'est pas nécessaire, puisque nous voulons garder la même direction que celle définie par les positions successives de la trajectoire à suivre. Ainsi donc la loi de commande donnant la vitesse de rotation du robot devient :

$$\Omega = k_\alpha \alpha + k_\rho \cos\alpha \sin\alpha \qquad (4.10)$$

La commande pour la vitesse linéaire reste inchangée et est donnée par l'équation (2.11), $v = k_\rho \rho \cos \alpha$.

4.3.1 Exemples d'applications pratiques

Les résultats obtenus en utilisant la commande non linéaire, pour le suivi de trajectoires, montrent qu'à part la phase de départ, pendant laquelle le robot débute ses mouvements, il n'y a pas une grande différence avec ceux donnés par la commande linéaire, une fois que ce dernier se trouve sur cette trajectoire. Comme cela a été observé sur l'exemple de la figure (2.10), la manière avec laquelle le robot réagit est très différente, dans le cas où la position à atteindre se trouve à l'arrière de la position initiale de ce dernier. L'avantage de la commande non linéaire réside dans le fait que l'espace requis, pour atteindre cette position, est beaucoup plus faible que celui nécessaire au mouvement généré par la commande linéaire.

4.3.1.1 Suivi d'une trajectoire à lignes parallèles

L'exemple typique pour ce genre de trajectoire peut être illustré dans le domaine de l'agriculture où des mouvements d'aller-retour sont nécessaires pour permettre à un tracteur de balayer un terrain agricole afin de procéder, par exemple à, un arrosage, un labourage ou encore diffuser des pesticides etc. Les photos des figures 4.6 et 4.7 montrent quelques exemples pour ce genre d'applications.

Figure 4.6 : Engin agricole pendant effectuant une moisson

Figure 4.7 : Arrosage d’un terrain agricole

Pour ce qui concerne le cas des applications ci-dessus les deux commandes proposées peuvent être utilisées pour générer la trajectoire permettant au robot de couvrir la totalité du terrain agricole concerné. Néanmoins, étant donné la taille et la nature de la surface à parcourir, les erreurs de mesure de la position du robot sont inévitables. Afin de remédier à cela, il sera nécessaire d’ajouter d’autres capteurs tels qu’un récepteur GPS (Global Positioning System), boussole électronique ou

encore une centrale inertielle comme sources d'informations supplémentaires. Un filtre de Kalman ou complémentaire pourront ensuite être exploités pour fusionner les données émanant de ces capteurs et obtenir des mesures plus précises. L'implémentation des commandes linéaire et non linéaire, sur le robot réel, permettant au robot d'effectuer des aller retours sont illustrées par la figure 4.8.

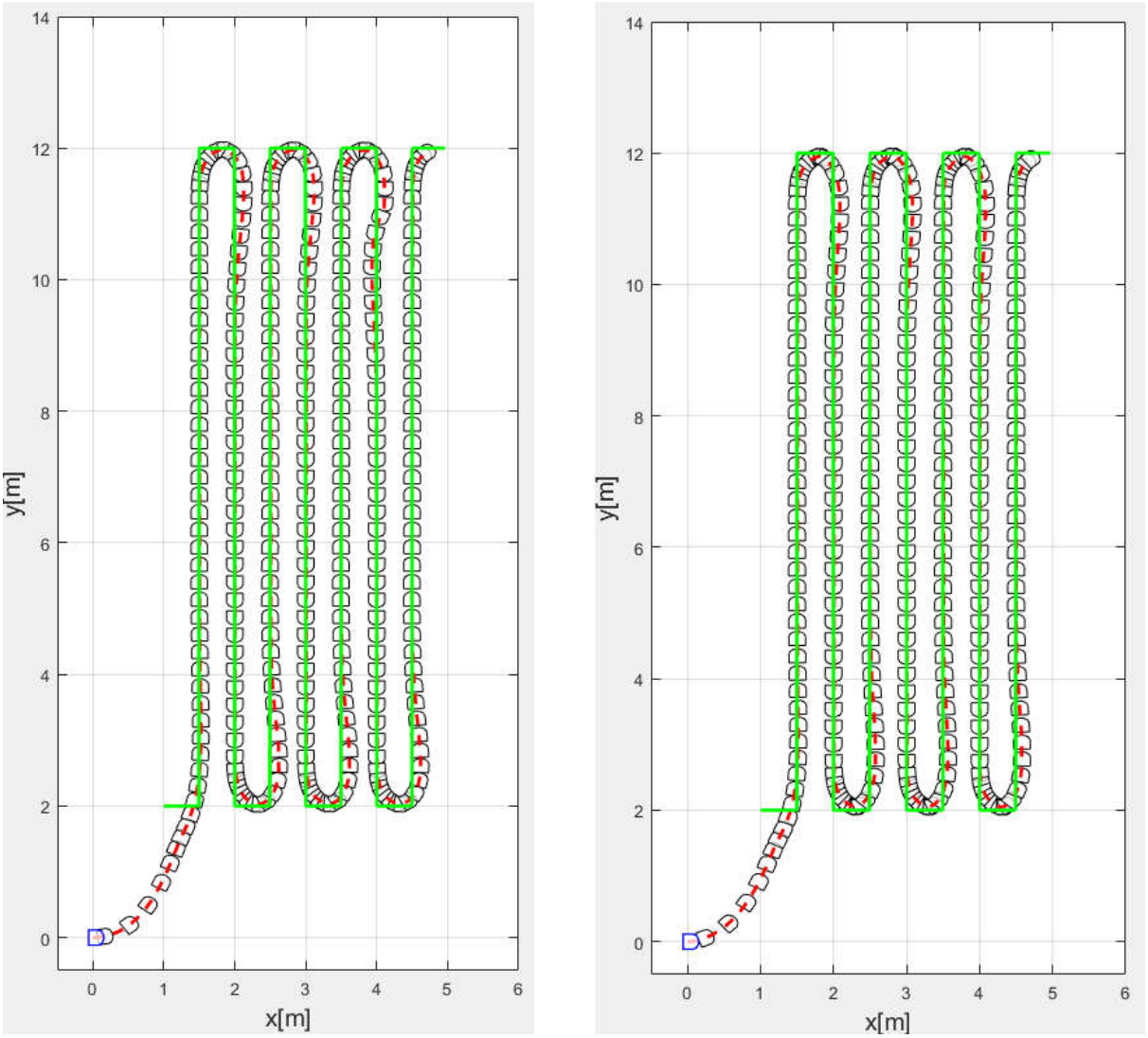


Figure 4.8 : Suivi de trajectoires avec une commande linéaire (gauche) et non-linéaire (droite)

4.3.1.2 Suivi d'une trajectoire de forme spirale

Une autre possibilité permettant au robot de couvrir une surface agricole consiste à lui faire suivre une trajectoire de forme spirale. Cette solution permet d'éviter au robot d'effectuer des virages brusques et donc de rester sur la trajectoire tout le long du parcours. La photo de la figure 4.9 montre l'exemple d'un arrosage assuré par des pivots tournants dont les centres sont fixés à des endroits immobiles, d'où la nécessité d'en avoir plusieurs pour couvrir une superficie plus importante.

Figure 4.9 : Pivots d'arrosage circulaire

Afin de générer une trajectoire en spirale nous avons utilisé les équations paramétriques (4.11). Celles-ci permettent au robot de suivre un chemin circulaire dont le rayon augmente de 1 mètre à pour chaque tour effectué. Il faut noter que, pour cette application, la largeur de la partie arrière du robot a été modifiée du fait de l'ajout d'un arrosoir pouvant couvrir une zone plus importante durant ses déplacements.

$$\begin{cases} x(t) = (1 + 0.02t)\sin\dfrac{2\pi}{50}t \\ y(t) = (1 + 0.02t)\cos\dfrac{2\pi}{50}t \end{cases} \quad (4.11)$$

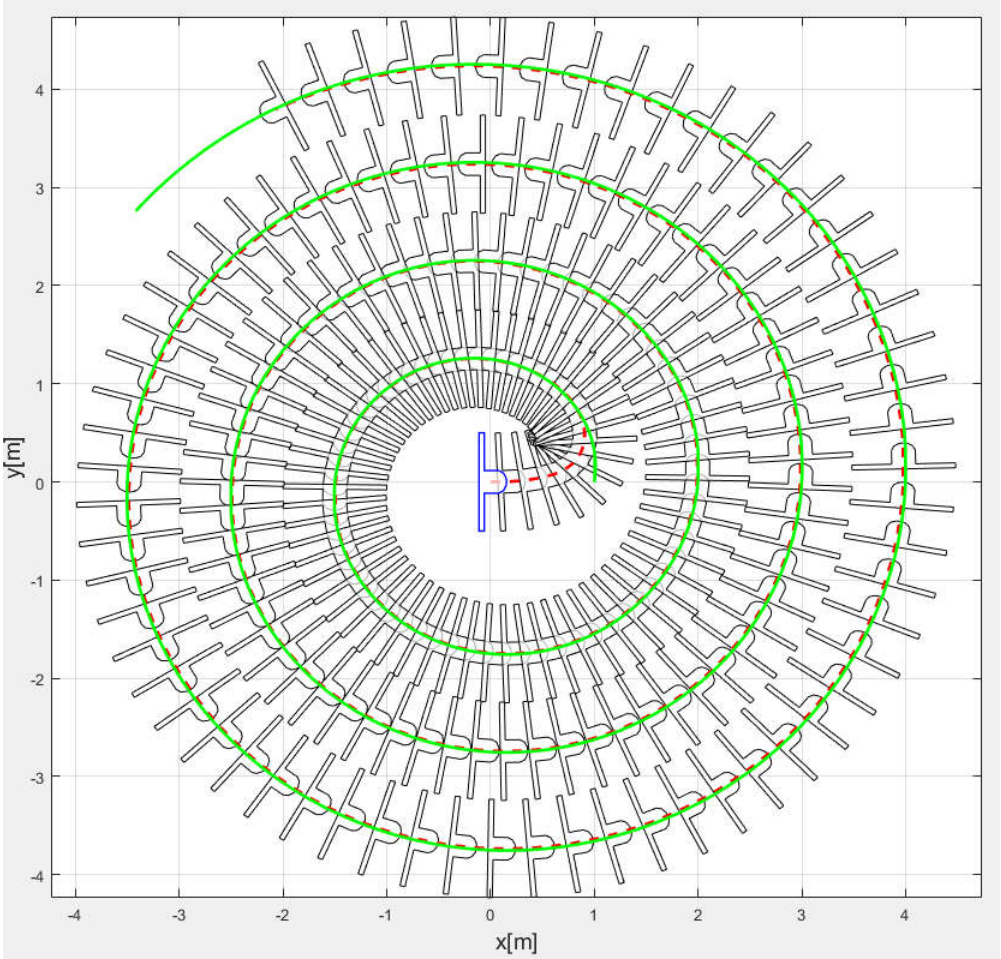


Figure 4.10 : Suivi du robot réel d'une trajectoire en spirale à l'aide de la commande linéaire

L'application des commandes linéaire et non-linéaire, pour le suivi de la trajectoire définie par les équations (4.11), a donné des résultats très proches. Les figures 4.10 et 4.11 représentent les résultats spatio-temporels d'un échantillon des tests réalisés sur le robot réel utilisant la commande linéaire.

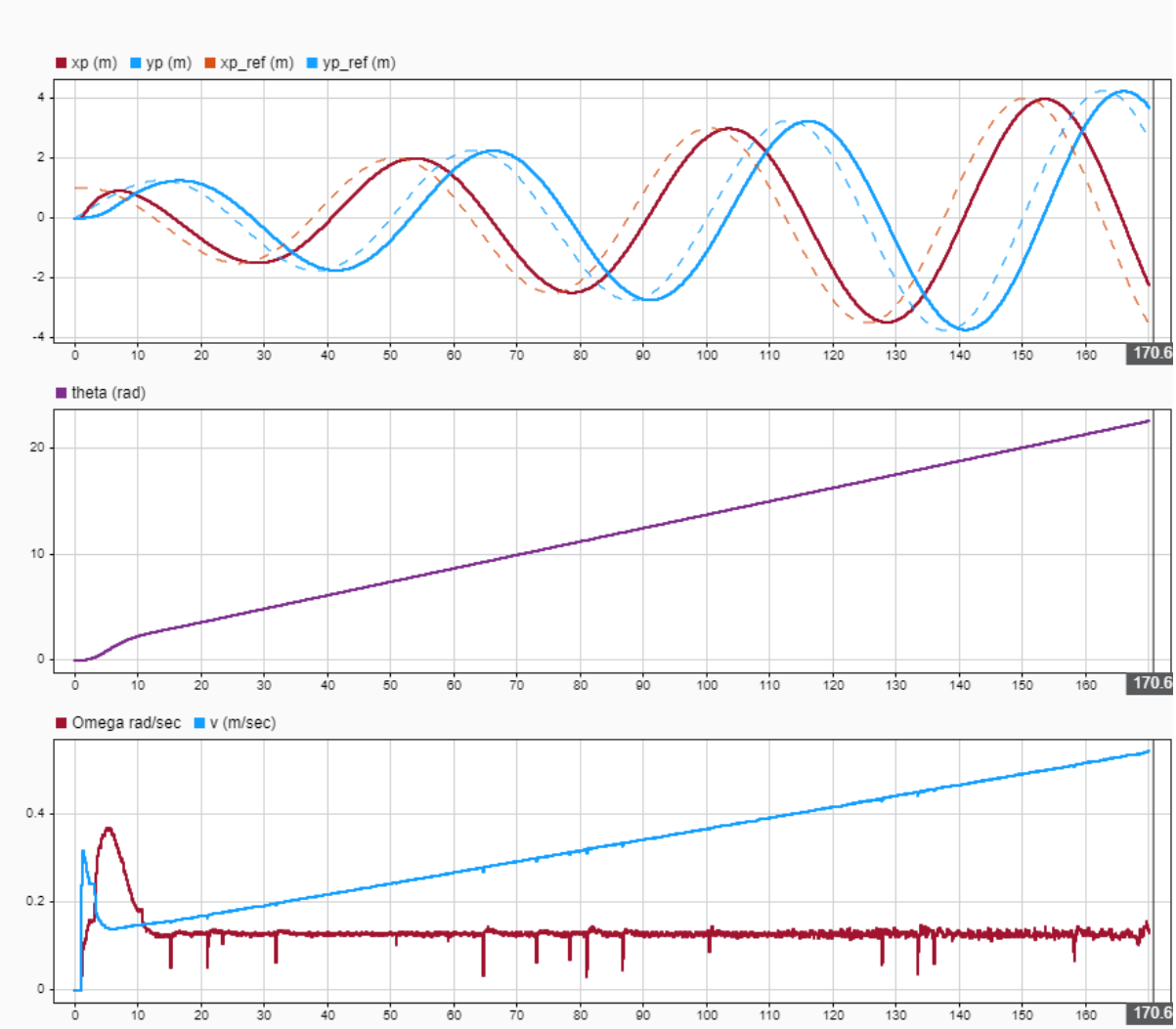


Figure 4.11 : Vitesses et positions du robot durant la trajectoire en spirale

Le suivi illustré par la figure 4.10 montre que la trajectoire du robot est très satisfaisante, néanmoins cela ne sera plus le cas lorsque la surface à couvrir est plus importante, car au fur et à mesure que le rayon de courbure augmente la vitesse linéaire augmente aussi, puisque le robot a besoin de rouler plus vite pour parcourir un périmètre de plus en plus long durant la même période de temps. Une solution possible pour résoudre ce problème serait d'utiliser les équations (4.11) pour générer la trajectoire complète, pour ensuite échantillonner ce parcours en un ensemble de segments de droites de longueurs

variables, que le robot pourra ensuite suivre en utilisant les mêmes stratégies de commandes discutées précédemment.

4.3.2 Suivi de trajectoires pour des applications d'intérieur

Dans un autre contexte, il est tout à tout fait possible d'envisager l'exploitation des résultats précédents pour des applications telles que le nettoyage des sols dans des endroits très spacieux et où cela doit se faire en permanence, tels que c'est le cas dans les aérogares ou autres espaces publics comme illustré par les photos de la figure 4.12.

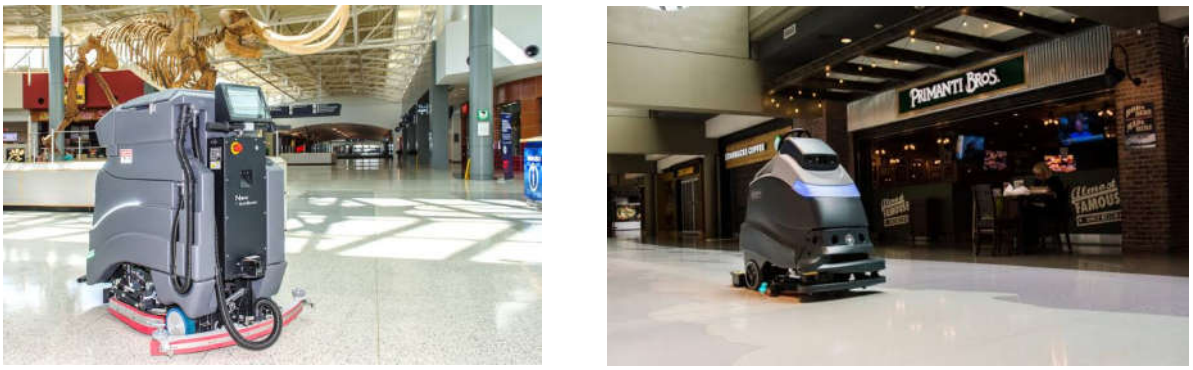


Figure 4.12 : Robots de nettoyage

Pour des raisons évidentes, lors de la pandémie du covid'19 les applications robotiques dans le domaine médical se sont multipliées. Cette situation a permis aux responsables de la santé de se rendre compte de l'importance de l'utilisation des robots mobiles autonomes pour s'occuper de certaines tâches, notamment celles qui nécessitent des déplacements vers des endroits à risques. La figure 4.13 montre des exemples de robots mobiles munis de systèmes de désinfection de l'air, au sein d'une structure de santé. Il est clair que, pour être performants, ce genre de robots doivent disposer d'un système de navigation très efficace leur permettant de couvrir totalement la zone à protéger.

Figure 4.13 : Robots de désinfection à l'ultra-violet contre le covid'19

Dans ce qui suit nous allons comparer les lois de commandes proposées lorsque celles-ci sont utilisées pour faire suivre le robot des trajectoires lui permettant de se déplacer dans des couloirs et les chambres d'un centre de santé. La figure 4.14 montre un exemple de scénario où le robot est programmé pour faire un balayage total d'un service de santé, en passant par toutes les chambres, l'une après l'autre. Une comparaison des mouvements résultants des deux commandes utilisées montre la supériorité de la commande non-linéaire. En effet, pour le cas de la commande linéaire, l'espace nécessaire pour effectuer ses manœuvres ne permet pas au robot de se déplacer sans risques de collision avec les murs de séparation des chambres et ne convient donc pas pour ce genre d'environnement. La commande non-linéaire, quant à elle, donne un résultat très satisfaisant, puisque le robot a pu exécuter cette mission avec succès. Restant dans le même contexte le robot peut donc être utilisé pour assurer d'autres taches, telles que la livraison de médicaments ou de plateaux repas aux patients occupant les chambres de ce service, il suffit pour cela, de modifier les différents points de passage par lesquels le robot doit passer pour atteindre les coordonnées de la cible visée.

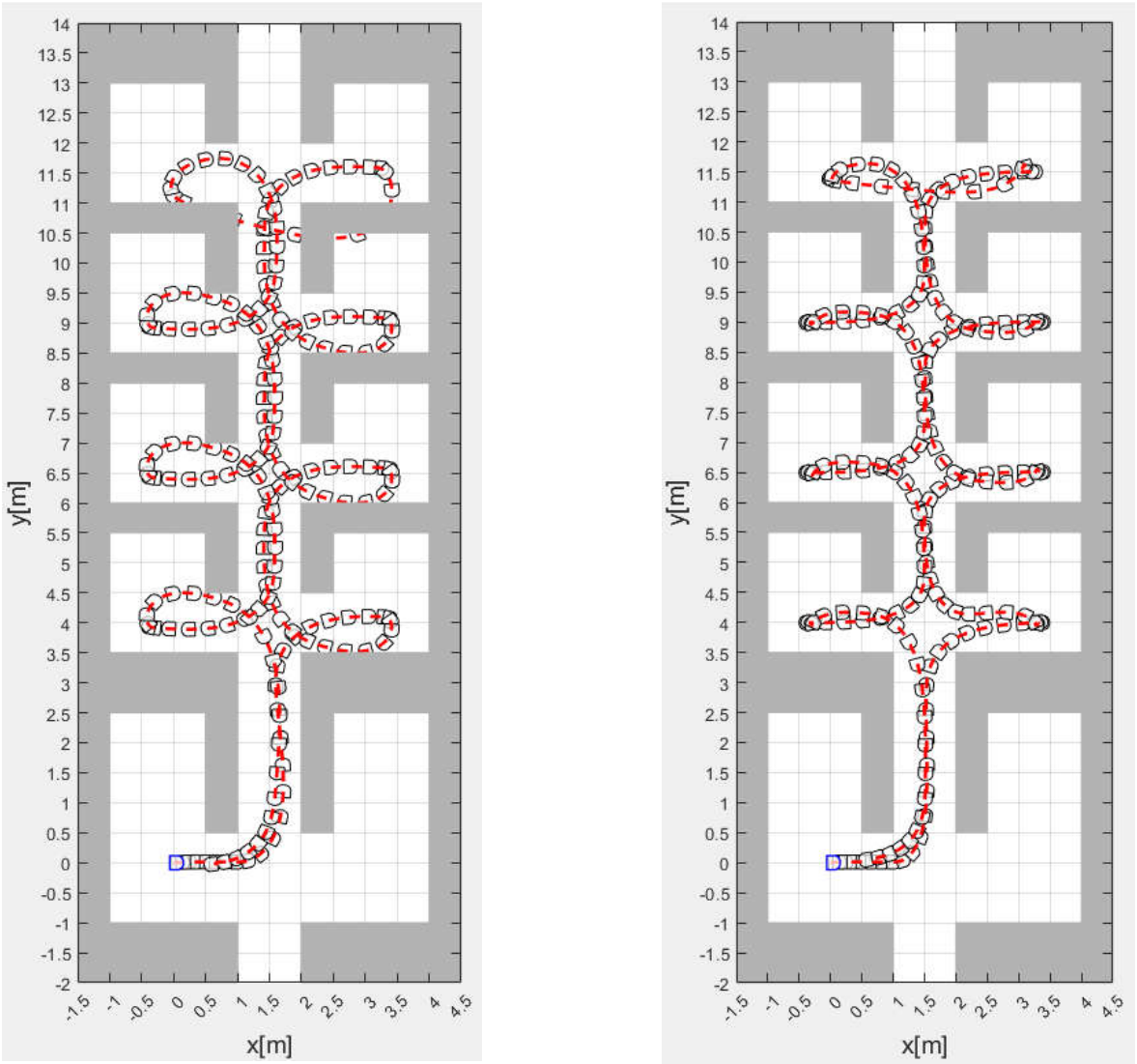


Figure 4.14 : Déplacements du robot dans un endroit restreint (gauche : commande linéaire, droite : commande non-linéaire)

4.3.3 Etude comparative avec la méthode ‘’pure pursuit’’

Afin de mettre évidence les différences du comportement du robot piloté à l’aide des commandes proposées avec celui obtenu en utilisant l’approche ‘’pure poursuit’’, nous l’avons soumis au même cas de la trajectoire à lignes parallèles. Les résultats de ces tests sont montrés sur les figures 4.15. Ce que nous pouvons constater de ce suivi est que cette méthode n’est pas convenable pour ce type de trajectoire, qui présente des variations assez

marquées au niveau des changements de directions, et ce pour différents choix de la distance d'anticipation $\mathbf{L_D}$. Afin d'améliorer le suivi de cette trajectoire il faudrait diminuer la vitesse longitudinale et la distance d'anticipation $\mathbf{L_D}$ tout en augmentant vitesse angulaire du robot. Sachant que cette dernière est limitée par les vitesses de rotation maximales, des moteurs gauche et droite du robot, nous ne pouvons donc pas satisfaire à ces hypothèses en même temps. Pour ce qui concerne l'exemple de la figure 4.15, les vitesses linéaire et angulaire ont été choisies conformément à ce qui est admissible par notre robot; $v = 0.5\ m/sec$, $\Omega = 0.77\ rad/sec$.

Dans le cas où la trajectoire à suivre ne présente pas de discontinuités accrues, cette méthode donne de très bons résultats. Son utilisation est aussi beaucoup mieux adaptée pour des véhicules de type voiture, où de rayon de braquage est beaucoup plus important que celui produit par les robots unicycle, utilisé dans ce travail.

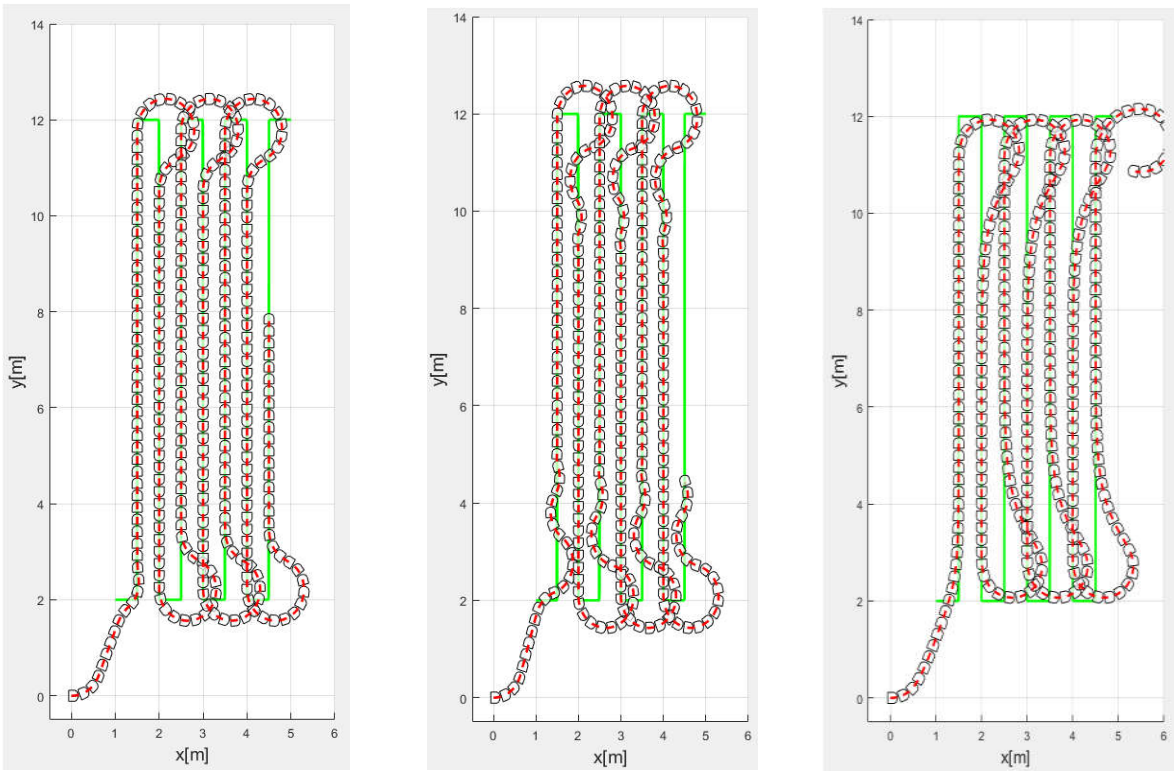


Figure 4.15 : Suivi de trajectoire utilisant la commande ‘’pure pursuit’’
(Gauche : L = 0.25 m, Centre : L = 0.1 m, Droite : L = 1m)

4.4 Conclusion

Dans le chapitre nous avons montré comment réutiliser les commandes proposées au chapitre 2 pour les adapter au suivi de trajectoires. Les résultats obtenus à travers les exemples utilisés ont démontré l'applicabilité de ces commandes sans aucune difficulté. De même que pour le cas des tests utilisés pour faire déplacer le robot vers une posture désirée, les différences des trajectoires poursuivies par le robot réel et celles produites par le modèle cinématique sont pratiquement négligeables. Cela nous offre donc une très grande flexibilité pour tester, en simulation, le comportement du robot réel avant de le brancher avec le PC sur lequel sont exécutés les calculs essentiels pour l'une ou l'autre des commandes utilisées. Il a été aussi montré que les approches de commande proposées, et notamment, la commande non-linéaire, donnent des résultats très prometteurs quant à leur utilisation dans des applications réelles, et ce dans de nombreux domaines socio-économiques.

Conclusion générale

Le thème principal de cette thèse concerne la mise en œuvre pratique d'un système de commande spécialement conçu pour le pilotage d'un robot mobile de type différentiel. Cela a été effectué en passant par différentes étapes, dont les premières ont été consacrées à une étude théorique concernant la modélisation des sous-ensembles constituant ce système. Pour ce faire nous avons considéré que le système global est constitué de deux parties complémentaires, dont la première, qualifiée de partie décisionnelle, est basée sur le modèle cinématique du robot pour calculer les vitesses de rotation des deux moteurs assurant la mobilité du robot. La deuxième partie, quant à elle, considérée comme partie exécutive, a comme fonction l'asservissement de ces deux vitesses. Cette dernière a été réalisée en tenant compte de l'inertie des deux roues qui y sont fixées, prenant ainsi, en charge une grande partie des contraintes dynamiques du système mobile. L'adoption de cette approche nous a permis d'implémenter les commandes proposées sur deux supports différents, prenant respectivement en charge, les parties décisionnelles et exécutives. Avec cette architecture modulaire nous pouvons très facilement modifier la nature de la commande au niveau décisionnel, puisque dans notre cas, cette dernière s'exécute sous l'environnement Matlab/Simulink sur un PC fixe. Avec la disponibilité d'une grande variété d'outils mathématiques et graphiques sous cet environnement, la plateforme conçue dans le cadre de ce travail, peut très bien être adaptée pour un usage pédagogique ou de recherche, permettant non seulement d'implémenter et tester différents types de contrôleurs, mais aussi, de pouvoir changer le type de système à contrôler, en apportant les modifications nécessaires au niveau des entrées/sorties de ce dernier. Pour ce qui concerne le cas particulier du présent travail, l'exploitation de la théorie de Lyapunov pour implémenter une commande non linéaire pour contrôler les mouvements d'un robot mobile a été réalisée avec succès, puisque les résultats présentés ont montré que cela a été le cas pour deux types d'objectifs différents ; à savoir celui pour le cas où le robot devait atteindre une posture finale, ainsi que pour le cas où ce dernier devait suivre une trajectoire, constituée d'un ensembles de points de passages prédéfinis. Les résultats des commandes proposées sur le modèle de simulation ainsi que sur le système réel ont montré une très grande similitude, cela prouve la bonne représentabilité des modèles utilisés, et surtout que de cette modélisation est très utile pour le développement d'autres commandes dans un minimum de temps. Bien que pour ce

qui concerne le suivi de trajectoires, les commandes linéaires et sa version non-linéaire ont donné des résultats comparables, la supériorité de la commande non linéaire proposée a été bien observée dans le cas où le robot est contraint d'évoluer dans un espace restreint. De ce fait le choix de l'une ou l'autre de ces commandes dépendra des conditions de leur utilisation et en fonction de la puissance de calcul dont dispose le processeur sur lequel cette commande sera implémentée. Pour le cas présent cela ne présente pas de difficultés particulières puisque cette commande est exécutée sur un PC avec suffisamment de ressources permettant de réaliser tous les calculs trigonométriques avec une très grande précision. Parmi les perspectives de ce travail, la possibilité d'utiliser une autre carte embarquée sur le robot pour remplacer le PC est tout à fait envisageable. Plusieurs tests doivent néanmoins être effectués pour décider du type de processeur dont doit être munie cette carte. A l'aide d'une caméra placée au-dessus de l'environnement dans lequel évolue le robot il serait possible de transformer ce travail en un système de navigation complet. Pour cela il suffira d'inclure une partie logicielle prenant en charge l'implémentation d'algorithmes de vision adéquats, capables de reconnaitre et isoler les éventuels obstacles présents sur cet environnement. A partir de cela il deviendra possible de prévoir l'implémentation des différentes méthodes de planification de trajectoires, telles que celles présentées au chapitre 3, et rendre ainsi le déplacement du robot complètement automatisé, puisque dans ce cas, les seules données à fournir au système seraient les coordonnées cartésiennes de la position finale que doit atteindre le robot.

Bibliographie

[1] B. Kazed, A. Guessoum, “A Lyapunov based posture controller for a differential drive mobile robot”, IAES International Journal of Robotics and Automation (IJRA) Vol. 13, No. 1, March 2024, pp. 1~10 ISSN: 2722-2586, doi:10.11591/ijra.v13i1.pp1-10

[2] M. Elsayed, A. Hammad, A. Hafez, and H. Mansour, “Real Time Trajectory Tracking Controller based on Lyapunov Function for Mobile Robot,” International Journal of Computer Applications, vol. 168, no. 11, pp. 1–6, Jun. 2017, doi: 10.5120/ijca2017914540.

[3] A. M. Ali, A. H. Mohammed, H. M. Alwan, “Path Tracking for Nonholonomic Mobile Robot by using Advance Lyapunov-based control laws”, Int. J. Cur. Eng. Tech., Vol.9, No.2 (March/April 2019) doi: https://doi.org/10.14741/ijcet/v.9.2.1.

[4] R. Kubo, Y. Fujii, H. Nakamura, “Control Lyapunov Function Design for Trajectory Tracking Problems of Wheeled Mobile Robot”, IFAC PapersOnLine 53-2 (2020) 6177–6182, doi: 10.1016/j.ifacol.2020.12.1704.

[5] L. Xiao-Feng, Z. Ren-Fang, C. Guo-Ping, “A New Non-Time-Based Tracking Control for Mobile Robot” Int. J. Robot. Eng. 5:025, 2020, doi: 10.35840/2631-5106/4125.

[6] S. Cao, Y. Jin, T. Trautmann, K. Liu, “Design and Experiments of Autonomous Path Tracking Based on Dead Reckoning”, Appl. *Sci.* 2023, *13*, 317. https://doi.org/10.3390/app13010317.

[7] J. Zhang, Q. Gong, Y. Zhang, J. Wang, “Finite-Time Global Trajectory Tracking Control for Uncertain Wheeled Mobile Robots”, IEEE Access, 2020, doi: 10.1109/ACCESS.2020.3030633.

[8] Y. M. Choi, J. H. Park, “Game-Based Lateral and Longitudinal Coupling Control for Autonomous Vehicle Trajectory Tracking” EEE Access, 2021, doi: 10.1109/ACCESS.2021.3135489.

[9] N. Hassan, A. Saleem, “Neural Network-Based Adaptive Controller for Trajectory Tracking of Wheeled Mobile Robots”, IEEE Access, doi: 10.1109/ACCESS.2022.3146970

[10] M. J. Rabbani, A. Y. Memon, “Trajectory Tracking and Stabilization of Nonholonomic Wheeled Mobile Robot Using Recursive Integral Backstepping Control”, Electronics 2021, 10, 1992, https://doi.org/10.3390/electronics10161992.

[11] L. Song, J. Huang, Q. Liang, L. NIE, X. Liang, J. ZHU, “Trajectory Tracking Strategy for Sliding Mode Control with Double Closed-Loop for Lawn Mowing Robot Based on ESO”, IEEE Access, 2022, doi: 10.1109/ACCESS.2022.3166816.

[12] M. Aicardi, G. Casalino, A. Bicchi and A. Balestrino, “Closed loop steering of unicycle like vehicles via Lyapunov techniques”, in IEEE Robotics & Automation Magazine, vol. 2, no. 1, pp. 27-35, March 1995, https://doi.org/10.1109/100.388294.

[13] S. K. Malu and J. Majumdar, “Kinematics, localization and control of differential drive mobile robot” *Glob. J. Res. Eng.*, 2014.

[14] T. T. Hoang, P.M. Duong, N.T.T. Van, T. Q. Vinh, “Stabilization control of the differential mobile robot using Lyapunov function and extended Kalman filter”. *J. Sci. Technol.* 2012, *50*, 441–452.

[15] C. Canudas de Wit, H. Khennouf, C. Samson, and O. J. Sørdalen, “Nonlinear control design for mobile robots,” in *Recent Trends in Mobile Robots*, Y. F. Zheng, Ed. Singapore: World Scientific, 1993, vol. 11, pp. 121–156.

[16] K. Amar, S. Mohamed, “Stabilized feedback control of unicycle mobile robots”, Int. J. Adv. Rob. Syst. 10 (4) (2013) 187, https://doi.org/10.5772/51323.

[17] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Noguchi, “A stable tracking control method for an autonomous mobile robot”, in Proceedings of the 1990 IEEE International Conference on Robotics and Automation 13-18 May 1990, vol. 381, pp. 384–389 (1990), https://doi.org/10.1109/robot.1990.126006.

[18] E. Fabregas, G. Farias, E. Aranda-Escolástico, G. Garcia, D. Chaos, S. Dormido-Canto, S. D. Bencomo, “Simulation and experimental results of a new control strategy for point stabilization of nonholonomic mobile robots”, *IEEE Trans. Ind. Electron.* 2019, *67*, 6679–6687, https://doi.org/10.1109/tie.2019.2935976.

[19] H. S. Shim and Y. G. Sung, “Stability and four-posture control for nonholonomic mobile robots”, *IEEE Trans. Robot. Autom.*, vol. 20, no. 1, pp. 148–154, Feb. 2004, https://doi.org/10.1109/tra.2003.819730.

[20] H. S. Shim and Y. G. Sung, “A Posture Control for Two Wheeled Mobile Robots”, Trans. On Control, Automation and Systems Engineering, Vol. 2 No. 3, pp. 201- 206, September 2000.

[21] P. Panahandeh, K. Alipour, B. Tarvirdizadeh, A. Hadi, “A kinematic Lyapunov-based controller to posture stabilization of wheeled mobile robots”, *Mech. Syst. Signal Process.* 2019, *134*, 106319. https://doi.org/10.1016/j.ymssp.2019.106319.

[22] A. Widyotriatmo, K. S. Hong, and L. H. Prayudhi, “Robust stabilization of a wheeled vehicle: Hybrid feedback control design and experimental validation”, *J. Mech. Sci. Technol.*, vol. 24, no. 2, pp. 513–520, Feb. 2010, doi:10.1007/s12206-010-0105-1.

[23] K. Kozłowski, J. Majchrzak, M. Michałek, and D. Pazderski, “Posture stabilization of a unicycle mobile robot – two control approaches”, in *Robot Motion Control*, vol. 335, pp. 25–54, Springer, Berlin, 2006. https://doi.org/10.1007/978-1-84628-405- 2_2.

[24] G. Oriolo, A. D. Luca, and M. Vendittelli, “WMR control via dynamic feedback linearization: design, implementation, and experimental validation”, *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002. https://doi.org/10.1109/tcst.2002.804116.

[25] F. C. Vieira, A. A. D. Medeiros, P. J. Alsina, and A. P. A. Jr., “Position and orientation control of a two-wheeled differentially driven nonholonomic mobile robot,” *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pp. 256-262, 2004. https://doi.org/10.5220/0001138702560262.

[26] P. K. Padhy, T. Sasaki, S. Nakamura, H. Hashimoto, “Modelling and position control of mobile robot”, in Advanced motion control, 2010 11th IEEE international workshop on, pp 100–105, March (2010), https://doi.org/10.1109/amc.2010.5464018.

[27] M. Thomas, B. Bandyopadhyay, L. Vachhani, “Posture stabilization of unicycle mobile robot using finite time control techniques”, *IFAC-Pap*. 2016; 49(1) :379-384. https://doi.org/10.1016/j.ifacol.2016.03.083.

[28] D. Jung, S. Bang, “Posture Stabilization of Wheeled Mobile Robot Based on Passivity-Based Robust Switching Control with Model Uncertainty Compensation”, *Appl. Sci.* 2019, *9*, 5233, https://doi.org/10.3390/app9235233.

[29] G. Artus, P. Morin, C. Samson, “Tracking of an omnidirectional target with a unicycle-like robot: control design and experimental results”. Research Report INRIA 4849, INRIA (2003).

[30] F. Quiroga, G. Hermosilla, G. Farias, E. Fabregas, G. Montenegro, “Position control of a mobile robot through deep reinforcement learning”, *Appl. Sci.* 2022, *12*, 7194, https://doi.org/10.3390/app12147194.

[31] G. Farias, G. Garcia, G. Montenegro, E. Fabregas, S. Dormido-Canto, “Position control of a mobile robot using reinforcement learning”, *IFAC-PapersOnLine* 2020, pp 17393–17398. https://doi.org/10.1016/j.ifacol.2020.12.2093.

[32) E.W Dijkstra, ‘’A note on two problems in connexion with graphs’’. *Numer. Math.* 1959, *1*, 269–271. [CrossRef]

[33] L. E. Kavraki, P. Svestka, J. C. Latombe, M. H. Overmars, 5 ‘’Probabilistic roadmaps for path planning in high-dimensional configuration spaces’’, IEEE Transactions on Robotics and Automation, V. 12, Issue 4, (June 1996), pages 566-580.

[34] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning" (http://msl.cs.uiuc/edu/~lavalle/papers/Lav98c.pdf). Technical Report (TR 98–11). Computer Science Department, Iowa State University.

[35] Qin, H.; Shao, S.; Wang, T.; Yu, X.; Jiang, Y.; Cao, Z. Review of Autonomous Path Planning Algorithms for Mobile Robots. *Drones* 2023, *7*, 211. https://doi.org/10.3390/drones7030211, pages 1603-1609.

[36] D. Sutherland, J. J. Sharples, K. A. Moinuddin, “The effect of ignition protocol on grassfire development”. International journal of wildland fire. 2020 Feb 4;29(1):70-80.

[37] Al-Araji, Ahmed Sabah. “A Comparative Study of Various Intelligent Algorithms Based Nonlinear PID Neural Trajectory Tracking Controller for the Differential Wheeled Mobile Robot Model.” Journal of Engineering, vol. 20, no. 05, July 2023, pp. 44–60. https://doi.org/10.31026/j.eng.2014.05.03

[38] G. Allibert, E. Courtial, and Y. Touré, “Real-time visual predictive controller for image-based trajectory tracking of a mobile robot,” IFAC Proceedings Volumes, vol. 41, no. 2, pp. 11244–11249, 2008, doi: 10.3182/20080706-5-kr-1001.01905

[39] M. Zhong, H. Zhao, Y. Yang, and J. Zhang, “Design of Trajectory Tracking Controller for Four Wheel Mobile Robot Based on Lyapunov Direct Method,” 2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI), Sep. 2018, doi: 10.1109/issi.2018.8538155.

[40] L. Du, “Double closed loop controller of wheeled mobile robot for trajectory tracking based on back-stepping and Lyapunov method,” Proceedings of the 2017 2nd International Conference on Materials Science, Machinery and Energy Engineering (MSMEE 2017), 2017, doi: 10.2991/msmee-17.2017.254.

[41] A. Khadhraoui and M. Saad, “Barrier Lyapunov Function Based Trajectory Tracking Controller of a Quadrotor UAV,” 2023 IEEE Canadian Conference on

Electrical and Computer Engineering (CCECE), Sep. 2023, doi: 10.1109/ccece58730.2023.10289068.

[42] M. Hussein, “A Non-linear Dynamic Controller for Differential Drive Mobile Robot Trajectory Tracking,” Journal of Advanced Research in Dynamical and Control Systems, vol. 12, no. SP8, pp. 837–845, Jul. 2020, doi: 10.5373/jardcs/v12sp8/20202587.

[43] M. Cui, J. Zhao, and H. Liu, “Design and Implementation of Trajectory Tracking Controller for Nonholonomic Mobile Robots Based on the Lyapunov Method,” 2019 Chinese Control Conference (CCC), Jul. 2019, doi: 10.23919/chicc.2019.8865395.

[44] Siegwart R. and Nourbakhsh I. R. and Scaramuzza D., “Introduction to Autonomous Mobile Robots”, Livre, MIT Press, (2011).

[45] X. Yang and A. Tayebi, “Vision based trajectory tracking controller for a B21R mobile robot,” 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2006, doi: 10.1109/iros.2006.282504.

[46] D. Xie, S. Wang, and Y. Wang, “Trajectory Tracking Control of Differential Drive Mobile Robot Based on Improved Kinematics Controller Algorithm,” 2018 Chinese Automation Congress (CAC), Nov. 2018, doi: 10.1109/cac.2018.8623764.

[47] Y. Kumar, S. B. Roy, and P. B. Sujit, “Barrier Lyapunov Function based Trajectory Tracking Controller for Autonomous Vehicles with Guaranteed Safety Bounds,” 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Sep. 2020, doi: 10.1109/icuas48674.2020.9214007.

[48] K. Benbouabdallah and Q. Zhu, “Bacterial foraging oriented by particle swarm optimization of a Lyapunov-based controller for mobile robot target tracking,” 2013 Ninth International Conference on Natural Computation (ICNC), Jul. 2013, doi: 10.1109/icnc.2013.6818029.

[49] Y. Mutoh and N. Kogure, “Experimental Results of Trajectory Tracking Control of Robot Manipulator using Time Varying Sliding Mode Controller,” Proceedings of the

13th International Conference on Informatics in Control, Automation and Robotics, 2016, doi: 10.5220/0005961304390446.

[50] A. Alouache and Q. Wu, “Genetic Algorithms for Trajectory Tracking of Mobile Robot Based on PID Controller,” 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Sep. 2018, doi: 10.1109/iccp.2018.8516587.

[51] T. Y.Abdalla and Abdulkareem. A. A, “PSO-based Optimum Design of PID Controller for Mobile Robot Trajectory Tracking,” International Journal of Computer Applications, vol. 47, no. 23, pp. 30–35, Jul. 2012, doi: 10.5120/7497-0601.

[52] F. Hmeyda and F. Bouani, “Camera-based autonomous mobile robot path planning and trajectory tracking using PSO algorithm and PID controller,” 2017 International Conference on Control, Automation and Diagnosis (ICCAD), Jan. 2017, doi: 10.1109/cadiag.2017.8075657.

[53] S. Xiong, M. Chen, and Z. Wei, “Air-ground Trajectory Tracking for Discrete-Time Autonomous Mobile Robot Based on Model Predictive Hybrid Tracking Control and Multiple Harmonics Time-varying Disturbance Observer,” Apr. 2022, doi: 10.21203/rs.3.rs-1480180/v1.

[54] O. Meiying, H. Sun, Z. Zhang, and S. Gu, “Fixed-Time Trajectory Tracking Control for Nonholonomic Mobile Robot Based on Visual Servoing,” Sep. 2021, doi: 10.21203/rs.3.rs-795447/v1.

[55] Y. Yan, M. Tang, W. Wang, Y. Zhang, and B. An, “Predictive Control of Robot Arm Trajectory Tracking Based on Control Lyapunov Function,” 2023 International Conference on Advanced Robotics and Mechatronics (ICARM), Jul. 2023, doi: 10.1109/icarm58088.2023.10218800.

[56] N. K. Goswami and P. K. Padhy, “Gain tuning of Lyapunov function-based controller using PSO for mobile robot control,” 2016 11th International Conference on Industrial and Information Systems (ICIIS), Dec. 2016, doi: 10.1109/iciinfs.2016.8262954.

[57] R. Safaric and K. Jezernik, “Trajectory tracking neural network controller for a robot mechanism and Lyapunov theory of stability,” Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94), doi: 10.1109/iros.1994.407366.

[58] A. Dumlu and M. R. Yildirim, “Real-Time Implementation of Continuous Model Based Sliding Mode Control Technique for Trajectory Tracking Control of Mobile Robot,” Balkan Journal of Electrical and Computer Engineering, vol. 6, no. 4, pp. 211–216, Oct. 2018, doi: 10.17694/bajece.459568.

[59] B. Jung and G. S. Sukhatme, “Real-time Motion Tracking from a Mobile Robot,” Defense Technical Information Center, Jan. 2005. doi: 10.21236/ada459071.

[60] H.-M. Wu and M. Q. Zaman, “LiDAR Based Trajectory-Tracking of an Autonomous Differential Drive Mobile Robot Using Fuzzy Sliding Mode Controller,” IEEE Access, vol. 10, pp. 33713–33722, 2022, doi: 10.1109/access.2022.3162244.

[61] Tai-Yu Wang and Ching-Chih Tsai, “Adaptive trajectory tracking control of a wheeled mobile robot via Lyapunov techniques,” 30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004, doi: 10.1109/iecon.2004.1433342.

[62] Z. Yu and S. F. Wong, “Autonomous mobile robot nonlinear trajectory tracking controller design and implementation on ROS-MATLAB system,” 2021 China Automation Congress (CAC), Oct. 2021, doi: 10.1109/cac53003.2021.9727795.

[63] K. S. Hatamleh, M. A. Jaradat, M. Hayajneh, and M. A. Momani, “Intelligent Adaptive Input-Output State Feedback Controller for Mobile Robot Trajectory Tracking,” 2024, doi: 10.2139/ssrn.4734841.

[64] H. A.R. Akar and F. R. Mahdi, “Trajectory Tracking Controller of Mobile Robot under Time Variation Parameters based on Neural Networks and Stochastic Fractal Algorithm,” Research Journal of Applied Sciences, Engineering and Technology, vol. 13, no. 11, pp. 871–878, Dec. 2016, doi: 10.19026/rjaset.13.3429.

[65] K. Benbouabdallah and Z. Qi-Dan, “Improved Genetic Algorithm Lyapunov-Based Controller for Mobile Robot Tracking a Moving Target,” Research Journal of Applied Sciences, Engineering and Technology, vol. 5, no. 15, pp. 4023–4028, Apr. 2013, doi: 10.19026/rjaset.5.4471.

[66] C. U. Dogruer, “Trajectory Tracking Control of a Mobile Robot with Model Predictive Controller and Observer,” 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA), Nov. 2019, doi: 10.1109/iccma46720.2019.8988644.

[67] M. Mohamed and M. Hamza, “Design PID Neural Network Controller for Trajectory Tracking of Differential Drive Mobile Robot Based on PSO,” Engineering and Technology Journal, vol. 37, no. 12A, pp. 574–583, Dec. 2019, doi: 10.30684/etj.37.12a.12.

[68] Z. Tao, Y. Wang, G. Li, and G. Hou, “Lyapunov based global trajectory tracking control of wheeled mobile robot,” Journal of Physics: Conference Series, vol. 2478, no. 10, p. 102018, Jun. 2023, doi: 10.1088/1742-6596/2478/10/102018.

[69] L. Añorve, M. Mera, H. Rios, M. Ballesteros, and I. Salgado, “Time-Varying Tracking Control of a Unicycle Mobile Robot: A Composite Lyapunov Function Approach,” 2023 11th International Conference on Control, Mechatronics and Automation (ICCMA), Nov. 2023, doi: 10.1109/iccma59762.2023.10374848.

[70] Y. Dai et al., “Backstepping Controller for Trajectory Tracking of Wheeled Mobile Robot Based on Particle Swarm optimization,” 2019 Chinese Control Conference (CCC), Jul. 2019, doi: 10.23919/chicc.2019.8866027.

[71] W. Yang and W. Liu, “Delta Robot Trajectory Tracking Based on Fuzzy Adaptive Sliding Mode Controller,” 2021 China Automation Congress (CAC), Oct. 2021, doi: 10.1109/cac53003.2021.9727620.

[72] L. Wang, T. Han, and L. Zhang, “Trajectory tracking for nonholonomic mobile robot using the sliding mode controller,” 2018 IEEE International Conference on Information and Automation (ICIA), Aug. 2018, doi: 10.1109/icinfa.2018.8812526.

[73] Z. Liu, W. Chen, J. Lu, and H. Wang, “Synchronous trajectory tracking for mobile robot network without velocity measurements between coupling robots,” 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), Jun. 2016, doi: 10.1109/rcar.2016.7784102.

[74] D. Pazderski, P. Szulczyński, and K. R. Kozłowski, “Kinematic Tracking Controller for Unicycle Mobile Robot Based on Polar-like Representation and Lyapunov Analysis,” Lecture Notes in Control and Information Sciences, pp. 45–56, 2009, doi: 10.1007/978-1-84882-985-5_5.

[75] K. Puentes and L. Morales, “Trajectory Tracking of a Mobile Robot Using a PID Controller Combined with Neural Networks,” 2023 IEEE Seventh Ecuador Technical Chapters Meeting (ECTM), Oct. 2023, doi: 10.1109/etcm58927.2023.10309094.

[76] F. DEMİRBAŞ and M. KALYONCU, “Differential drive mobile robot trajectory tracking with using pid and kinematic based backstepping controller,” Selcuk University Journal of Engineering, Science and Technology, vol. 5, no. 1, pp. 1– 15, Mar. 2017, doi: 10.15317/scitech.2017.65.

[77] K. E. Dagher and A. Al-Araji, “Design of a Nonlinear PID Neural Trajectory Tracking Controller for Mobile Robot based on Optimization Algorithm,” Engineering and Technology Journal, vol. 32, no. 4, pp. 973–985, May 2014, doi: 10.30684/etj.32.4a.13.

[78] M. J. Mohamed and M. Y. Abbas, “Design a Fuzzy PID Controller for Trajectory Tracking of Mobile Robot,” Engineering and Technology Journal, vol. 36, no. 1A, pp. 100–110, Jan. 2018, doi: 10.30684/etj.2018.136785.

[79] A. S. Al-Araji and N. Q. Yousif, “A Cognitive Nonlinear Trajectory Tracking Controller Design for Wheeled Mobile Robot based on Hybrid Bees-PSO Algorithm,” Engineering and Technology Journal, vol. 35, no. 6, pp. 609–616, Jun. 2017, doi: 10.30684/etj.2017.131978.

[80] X. Lu, D. Ren, and S. Yu, “FPGA-based real-time object tracking for mobile robot,” 2010 International Conference on Audio, Language and Image Processing, Nov. 2010, doi: 10.1109/icalip.2010.5685091.

[81] M. Fu, T. Zhang, F. Ding, and D. Wang, “Appointed-Time Integral Barrier Lyapunov Function-Based Trajectory Tracking Control for a Hovercraft with Performance Constraints,” Applied Sciences, vol. 10, no. 20, p. 7381, Oct. 2020, doi: 10.3390/app10207381.

[82] Navin Chandra P. and Mija S.J., “Robust controller for trajectory tracking of a Mobile Robot,” 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), Jul. 2016, doi: 10.1109/icpeices.2016.7853137.

[83] N. Khlif, N. Khraief, and S. Belghith, “Mobile Robot Modelling and Design of Kinematic Controller for trajectory tracking,” 2023 20th International Multi- Conference on Systems, Signals & Devices (SSD), Feb. 2023, doi: 10.1109/ssd58187.2023.10411315.

[84] N. Ali, “Fuzzy Logic based Real Time Go to Goal Controller for Mobile Robot,” International Journal of Computer Applications, vol. 176, no. 11, pp. 32–36, Apr. 2020, doi: 10.5120/ijca2020920078.

[85] M. E. Hedroug, E. Bdirina, and K. Guesmi, “Fuzzy Predictive Controller for Trajectory Tracking of Differential-Drive Mobile Robot,” 2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM), Nov. 2023, doi: 10.1109/ic2em59347.2023.10419477.

[86] E. Dönmez, A. F. Kocamaz, and M. Dirik, “A Vision-Based Real-Time Mobile Robot Controller Design Based on Gaussian Function for Indoor Environment,” Arabian Journal for Science and Engineering, vol. 43, no. 12, pp. 7127–7142, Dec. 2017, doi: 10.1007/s13369-017-2917-0.

[87] S. Islam, “Lyapunov-based hybrid control for robust trajectory tracking of robotic manipulators”, doi: 10.22215/etd/2010-09448.

[88] D. Ben Halima Abid, N. Y. Allagui, and N. Derbel, “Navigation and trajectory tracking of mobile robot based on kinematic PI controller,” 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Dec. 2017, doi: 10.1109/sta.2017.8314966.

[89] X. Yu, W. Zhu, and L. Xu, “Real-time Motion Planning and Trajectory Tracking in Complex Environments based on Bézier Curves and Nonlinear MPC Controller,” 2020 Chinese Control and Decision Conference (CCDC), Aug. 2020, doi: 10.1109/ccdc49329.2020.9163994.

[90] A. Alouache and Q. Wu, “Fuzzy logic PD controller for trajectory tracking of an autonomous differential drive mobile robot (i.e. Quanser Qbot),” Industrial Robot: An International Journal, vol. 45, no. 1, pp. 23–33, Jan. 2018, doi: 10.1108/ir-07-2017-0128.

[91] J. ZHONG and J. SU, “A Real-time Moving Object Tracking System Based on Visual Prediction,” ROBOT, vol. 32, no. 4, pp. 516–521, Aug. 2010, doi: 10.3724/sp.j.1218.2010.00516.

[92] N. H. Thai, T. T. K. Ly, and L. Q. Dzung, “Trajectory tracking control for mecanum wheel mobile robot by time-varying parameter PID controller,” Bulletin of Electrical Engineering and Informatics, vol. 11, no. 4, pp. 1902–1910, Aug. 2022, doi: 10.11591/eei.v11i4.3712.

[93] L. T. K. Trịnh and T. Hoang, “Bézier trajectory tracking control of The Omnidirectional Mobile Robot based on a linear time- varying state feedback controller,” Science and Technology Development Journal, 2022, doi: 10.32508/stdj.v25i2.3914.

[94] G. Haifeng and L. Nmgbo, “Robust H<inf>∞</inf> controller based on variable gain linear parameter varying and pole assignment of nonholonomic mobile robot trajectory tracking,” 2018 Chinese Control And Decision Conference (CCDC), Jun. 2018, doi: 10.1109/ccdc.2018.8407179.

[95] K. Singhirunnusorn, F. Fahimi, and R. Aygun, “A single camera 360-degree real time vision-based localization method with application to mobile robot trajectory tracking,” IET Cyber-Systems and Robotics, vol. 3, no. 3, pp. 185–198, May 2021, doi: 10.1049/csy2.12021.

[96] M. R. Azizi and J. Keighobadi, “Robust Sliding Mode Trajectory Tracking Controller for a Nonholonomic Spherical Mobile Robot,” IFAC Proceedings Volumes, vol. 47, no. 3, pp. 4541–4546, 2014, doi: 10.3182/20140824-6-za-1003.01430.

[97] K. Singhirunnusorn, F. Fahimi, and R. Aygun, “A single camera 360-degree real time vision-based localization method with application to mobile robot trajectory tracking,” IET Cyber-Systems and Robotics, vol. 3, no. 3, pp. 185–198, May 2021, doi: 10.1049/csy2.12021.

[98] A. Andreev and O. Peregudova, “On Time-Delayed Feedback Trajectory Tracking Control of a Mobile Robot with Omni-Wheels,” 2019 12th International Workshop on Robot Motion and Control (RoMoCo), Jul. 2019, doi: 10.1109/romoco.2019.8787373.

Annexe 1

MICROCHIP

dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 AND dsPIC33FJ128MCX02/X04

High-Performance, 16-bit Digital Signal Controllers

Operating Range:

- Up to 40 MIPS operation (at 3.0V -3.6V):
  - Industrial temperature range (-40°C to +85°C)
  - Extended temperature range (-40°C to +125°C)
- Up to 20 MIPS operation (at 3.0V -3.6V):
  - High temperature range (-40°C to +140°C)

High-Performance DSC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set
- 16-bit wide data path
- 24-bit wide instructions
- Linear program memory addressing up to 4M instruction words
- Linear data memory addressing up to 64 Kbytes
- 83 base instructions: mostly 1 word/1 cycle
- Two 40-bit accumulators with rounding and saturation options
- Flexible and powerful addressing modes:
  - Indirect
  - Modulo
  - Bit-Reversed
- Software stack
- 16 x 16 fractional/integer multiply operations
- 32/16 and 16/16 divide operations
- Single-cycle multiply and accumulate:
  - Accumulator write back for DSP operations
  - Dual data fetch
- Up to ±16-bit shifts for up to 40-bit data

Direct Memory Access (DMA):

- 8-channel hardware DMA
- Up to 2 Kbytes dual ported DMA buffer area (DMA RAM) to store data transferred via DMA:
  - Allows data transfer between RAM and a peripheral while CPU is executing code (no cycle stealing)
- Most peripherals support DMA

Timers/Capture/Compare/PWM:

- Timer/Counters, up to five 16-bit timers:
  - Can pair up to make two 32-bit timers
  - One timer runs as a Real-Time Clock with an external 32.768 kHz oscillator
  - Programmable prescaler
- Input Capture (up to four channels):
  - Capture on up, down or both edges
  - 16-bit capture input functions
  - 4-deep FIFO on each capture
- Output Compare (up to four channels):
  - Single or Dual 16-bit Compare mode
  - 16-bit Glitchless PWM mode
- Hardware Real-Time Clock and Calendar (RTCC):
  - Provides clock, calendar and alarm functions

Interrupt Controller:

- 5-cycle latency
- Up to 53 available interrupt sources
- Up to three external interrupts
- Seven programmable priority levels
- Five processor exceptions

Digital I/O:

- Peripheral pin Select functionality
- Up to 35 programmable digital I/O pins
- Wake-up/Interrupt-on-Change for up to 31 pins
- Output pins can drive from 3.0V to 3.6V
- Up to 5V output with open drain configuration
- All digital input pins are 5V tolerant
- 4 mA sink on all I/O pins

On-Chip Flash and SRAM:

- Flash program memory (up to 128 Kbytes)
- Data SRAM (up to 16 Kbytes)
- Boot, Secure, and General Security for program Flash

dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 AND dsPIC33FJ128MCX02/X04

System Management:

- Flexible clock options:
  - External, crystal, resonator, internal RC
  - Fully integrated Phase-Locked Loop (PLL)
  - Extremely low jitter PLL
- Power-up Timer
- Oscillator Start-up Timer/Stabilizer
- Watchdog Timer with its own RC oscillator
- Fail-Safe Clock Monitor
- Reset by multiple sources

Power Management:

- On-chip 2.5V voltage regulator
- Switch between clock sources in real time
- Idle, Sleep, and Doze modes with fast wake-up

Analog-to-Digital Converters (ADCs):

- 10-bit, 1.1 Msps or 12-bit, 500 Ksps conversion:
  - Two and four simultaneous samples (10-bit ADC)
  - Up to nine input channels with auto-scanning
  - Conversion start can be manual or synchronized with one of four trigger sources
  - Conversion possible in Sleep mode
  - ±2 LSb max integral nonlinearity
  - ±1 LSb max differential nonlinearity

Audio Digital-to-Analog Converter (DAC):

- 16-bit Dual Channel DAC module
- 100 Ksps maximum sampling rate
- Second-Order Digital Delta-Sigma Modulator

Comparator Module:

- Two analog comparators with programmable input/output configuration

CMOS Flash Technology:

- Low-power, high-speed Flash technology
- Fully static design
- 3.3V (±10%) operating voltage
- Industrial and Extended temperature
- Low power consumption

Motor Control Peripherals:

- 6-channel 16-bit Motor Control PWM:
  - Three duty cycle generators
  - Independent or Complementary mode
  - Programmable dead-time and output polarity
  - Edge-aligned or center-aligned
  - Manual output override control
  - One Fault input
  - Trigger for ADC conversions
  - PWM frequency for 16-bit resolution (@ 40 MIPS) = 1220 Hz for Edge-Aligned mode, 610 Hz for Center-Aligned mode
  - PWM frequency for 11-bit resolution (@ 40 MIPS) = 39.1 kHz for Edge-Aligned mode, 19.55 kHz for Center-Aligned mode
- 2-channel 16-bit Motor Control PWM:
  - One duty cycle generator
  - Independent or Complementary mode
  - Programmable dead time and output polarity
  - Edge-aligned or center-aligned
  - Manual output override control
  - One Fault input
  - Trigger for ADC conversions
  - PWM frequency for 16-bit resolution (@ 40 MIPS) = 1220 Hz for Edge-Aligned mode, 610 Hz for Center-Aligned mode
  - PWM frequency for 11-bit resolution (@ 40 MIPS) = 39.1 kHz for Edge-Aligned mode, 19.55 kHz for Center-Aligned mode
- 2-Quadrature Encoder Interface module:
  - Phase A, Phase B, and index pulse input
  - 16-bit up/down position counter
  - Count direction status
  - Position Measurement (x2 and x4) mode
  - Programmable digital noise filters on inputs
  - Alternate 16-bit Timer/Counter mode
  - Interrupt on position counter rollover/underflow

dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 AND dsPIC33FJ128MCX02/X04 PRODUCT FAMILIES

The device names, pin counts, memory sizes, and peripheral availability of each device are listed below. The following pages show their pinout diagrams.

dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 and dsPIC33FJ128MCX02/X04 Controller Families

Device	Pins	Program Flash Memory (Kbyte)	RAM (Kbyte)(1)	Remappable Peripheral										RTCC	I^2C™	CRC Generator	10-bit/12-bit ADC (Channels)	6-pin 16-bit DAC	Analog Comparator (2 Channels/Voltage Regulator)	8-bit Parallel Master Port (Address Lines)	I/O Pins	Packages
				Remappable Pins	16-bit Timer(2)	Input Capture	Output Compare Standard PWM	Motor Control PWM (Channels)(3)	Quadrature Encoder Interface	UART	SPI	ECAN™	External Interrupts(4)									
dsPIC33FJ128MC804	44	128	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ128MC802	28	128	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ128MC204	44	128	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ128MC202	28	128	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC804	44	64	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ64MC802	28	64	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC204	44	64	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ64MC202	28	64	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ32MC304	44	32	4	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ32MC302	28	32	4	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S

Note 1: RAM size is inclusive of 2 Kbytes of DMA RAM for all devices except dsPIC33FJ32MC302/304, which include 1 Kbyte of DMA RAM.
2: Only four out of five timers are remappable.
3: Only PWM fault pins are remappable.
4: Only two out of three interrupts are remappable.

Annexe 2