REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Abbès LAGHROUR - Khenchela
Faculté des Sciences et de la technologie
Département de Génie Industriel

Mémoire de fin d'études

Pour l'Obtention de Diplôme Master

Filière : Automatique

Spécialité : Automatique et informatique industrielle

Réalisé par :

Aouragh Hadile et Messai Ahmed Taha

Intitulé :

Commande d'un robot mobile basée sur un algorithme d'optimisation de type métahistorique

Dirigée par : Dr. ALLOUANI Fouad

Membres du Jury :

Bououden Sofiane	Prof. à l'université de Khenchela	Président
Benfaroudj Hafiza	MA à l'université de Khenchela	Examinateur
Allouani Fouad	MCA à l'université de Khenchela	Rapporteur

Année universitaire 2020/2021

ملخص

يتعلق العمل المقدم في هذه المخطوطة بشكل أساسي بتشكيل وتطبيق قوانين التحكم ، استنادًا إلى النهج الذي يستند على استعمال خوارزمية التحسين المسمات بالإنكليزية 'Partial Swarm Optimization (PSO)'، للتتبع مسارات الروبوتات المتنقلة غير الشاملة .في هذه الأطروحة، تم تخصيص اهتمام خاص لنوع الروبوتات المتنقلة ذات الدراجة الأحادية .بالإضافة إلى ذلك، تستند النماذج المدروسة بتشكيل قوانين التحكم على النموذج الحركي للروبوت المتحرك.

يتم اختبار طريقة التحكم المقدمة من خلال محاكاة بواسطة برنامج MATLAB . تظهر جميع النتائج التي تم الحصول عليها فعالية طريقة التحكم المستعملة لمعالجة المشكلة المطروحة.

Résumé :

Le travail présenté dans ce mémoire de Master concerne principalement, la synthèse et l'application d'une loi de commande basée sur l'utilisation d'algorithme d'Optimisation par Essaim Particulaire OPE, pour le suivi de trajectoire des robots mobiles non-holonomes. Nous nous intéressons dans notre étude seulement aux robots mobiles de type unicycle. Il est à noter aussi que, les modèles considérés pour synthétiser les lois de commandes dans ce mémoire sont basés sur le modèle cinématique du robot mobile de type unicycle.

La méthode de commande présentée, est testée à travers une simulation par le logiciel MATLAB. L'ensemble des résultats obtenus, montre l'efficacité de cette approche pour le problème traité.

Abstract:

The work presented in this manuscript, is mainly concerns the synthesis and application of control laws, based on the use of the Partial Swarm Optimization (PSO) algorithm, for tracking the trajectories of non-holonomic mobile robots. In this thesis, particular interest is reserved for unicycle mobile robots' type. In addition, the models considered for the synthesis of the control laws are based on the kinematic model of the mobile robot.

The presented control method, is tested through a simulation by MATLAB software. All obtained results show the effectiveness of this approach for the treated problem.

Table des matières

Introduction générale

Introduction générale

La robotique mobile vise à rendre une machine autonome, c'est-à-dire lui donner les capacités de perception, de décision et d'action pour agir de manière autonome sans assistance ni intervention humaine avec son environnement, c'est un axe de recherche à la croisée de plusieurs disciplines scientifiques et techniques tel que la mécanique, l'électronique, l'électrotechnique et l'informatique…etc.

Les études de commande des robots mobiles à roues remontent aux années 80. Ces robots mobiles sont destinés à réaliser les tâches considérées comme pénibles ou les tâches s'accomplissant dans des milieux hostiles à l'être humain comme les milieux marin (tel que la recherche de la boite noire des avions), spatiale (tel que l'exploration planétaire) …etc. Le robot peut aussi être utile dans le domaine médical pour compenser les handicapes physiques.

Généralement, les modèles utilisés dans ce contexte sont non-linéaires et la propriété de non-holonomie qui caractérise un certain nombre de ces robots mobiles conduit à l'utilisation de techniques de commande non linéaire pour les piloter. En effet, la problématique générale se réduit dans la plupart des cas étudies à faire se déplacer le robot dans un environnement connu ou inconnu, tout en évitant d'éventuels obstacles fixes ou mobiles, pour réaliser une tâche prescrite. Il en découle qu'il faut pouvoir définir une stratégie de mouvement (planification), puis exécuter le déplacement prescrit.

En effet, la commande de suivi de trajectoires des robots mobiles non-holonomes (cas général) peut être considérée comme un problème d'optimisation, basé sur l'utilisation d'une objective écrite en fonction des objectives de commandes envisagées. Dans ce but, l'utilisation d'une méthode ou algorithme dit d'optimisation est devient indispensable.

Puisque les méthodes classiques (algorithmes conventionnels), ont montré leurs limites en termes de précision, temps de calcul, et univers d'applications, nous avons orienté nôtre intérêt à une catégorie d'algorithmes d'optimisation, dites méthodes méta-heuristiques.

En effet, les méta-heuristiques forment une famille d'algorithmes d'optimisation visant à ré-soudre des problèmes d'optimisation difficile, pour lesquels nous ne connaissons pas de méthodes classiques plus efficaces. Elles sont généralement utilisées comme des méthodes

génériques pouvant optimiser une large gamme de problèmes différents, d'où le qualificatif *méta*.

Parmi les algorithmes qui appartient à cette famille et qui connu une très large utilisation dans un grand nombre d'applications, l'algorithme dit en anglais : Particle Swarm Optimization (PSO) c-à-d. Optimisation par Essaim Particulaire (OEP), ce dernier a été proposé par les deux chercheurs Kennedy et Eberhart [35], qui ont été motivés par le comportement social du troupeau d'oiseaux ou du banc de poissons. En raison de ses caractéristiques spéciales, telles que la proximité, la qualité, la réponse diversifiée, la stabilité et l'adaptabilité, il a été mis en œuvre avec succès pour résoudre de nombreux problèmes d'ingénierie.

L'objectif de ce mémoire de Master trouve son intérêt dans la synthèse et l'application d'une loi de commande basée sur l'utilisation d'algorithme OPE, pour le suivi de trajectoire des robots mobiles non-holonomes. Nous nous intéressons dans notre étude seulement aux robots mobiles de type unicycle. Il est à noter aussi que, les modèles considérés pour synthétiser les lois de commandes dans ce mémoire sont basés sur le modèle cinématique du robot mobile de type unicycle.

Le premier chapitre de ce mémoire présente un aperçu général des concepts de base de la robotique mobile.

Le second chapitre parle ou profond sur la cinématique de quelques types de robots mobiles à savoir : les robots de type unicycle, les robots de type tricycle, et les robots de type voiture.

Le troisième chapitre est divisé en deux grande parties, dont le continu de la première est concentré sur l'environnement du robot, c-à-d. le traitement du problème de suivi d'une trajectoire donnée lors de la présence de deux scenarios différents (i) suivi d'une trajectoire avec contrôle/sans contrôle d'orientation et (ii) suivi d'une trajectoire stabilisation/sans stabilisation. Dans la seconde partie, quelques approches de commande relatives à ce problème seront montrées.

Dans le dernier chapitre de ce mémoire, nous parlerons au sens général sur l'algorithme OPE, puis sur la méthode de synthèse de la commande basée sur ce dernier, suivie par deux exemples de simulations.

Enfin, ce mémoire est achevé par une conclusion générale.

Chapitre I

Chapitre I : Généralité sur la robotique mobile

I.1. Introduction

La robotique mobile est un domaine dans lequel l'expérience pratique est particulièrement illustratrice et importante pour la compréhension des problèmes. Elle est la branche de l'intelligence artificielle concernée par l'étude de systèmes automatiques capables d'une interaction directe avec le monde physique. Elle est l'ensemble des techniques permettant la conception, la réalisation de machines automatiques ou de robots mobiles. La robotique mobile est un domaine de recherche essentiellement multidisciplinaire. Sa problématique porte sur la conception et l'étude de fonctions de *perception*, de *décision* et d'*action*, et sur l'*intégration* cohérente de ces fonctions en une machine physique. Cette machine devra mettre en œuvre ses fonctions sensori- motrices et décisionnelles pour réaliser de façon autonome une diversité de tâches dans un environnement dynamique imparfaitement modélisé, pour interagir avec d'autres machines et avec des humains, et pour améliorer ses propres performances par apprentissage. Ce premier chapitre présente un aperçu général des concepts de base de la robotique mobile.

I.2. Concepts sur la robotique mobile

La mobilité autonome des robots est devenue un sujet de recherche développé par tous les pays industrialisés. Qu'il s'agisse de robots mobiles à pattes, à roues ou même sous-marins et aériens, les applications sont vastes et multiples : robots de services, surveillance, construction, nettoyage, manipulation de charges, automobile intelligente, robots d'intervention, robots d'exploration planétaire ou de fonds marins, satellites, robots militaires, etc. Le marché potentiel de la robotique est considérable, même s'il faut pour cela résoudre des problèmes plus importants et plus fondamentaux que prévus initialement dans la quête vers la machine intelligente [1,2].

I.2.1. Définition d'un robot mobile

Un robot est une machine chargée d'effectuer une ou plusieurs tâches. Le mot vient d'une pièce de théâtre tchèque dans laquelle apparaissait un travailleur artificiel employé pour le « servage », désigné en tchèque par « *robota* ». L'appellation *Robot mobile* regroupe tous les types de robots qui ont la capacité de déplacement qui est la caractéristique commune entre eux, la différence réside dans la manière, qui dépend du domaine d'utilisation de robot, par laquelle le robot va atteindre cette faculté de mouvement. La mobilité par les roues est la structure mécanique la plus communément appliquée. Cette technique assure selon l'agencement et les dimensions des roues un déplacement dans toutes les directions avec une accélération et une vitesse importante.

Afin d'être autonome, un robot doit être équipé de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a et de ses objectifs [1], mais également de savoir comment réagir en conséquence, suivant le niveau d'autonomie. C'est à lui de planifier son parcours et de déterminer avec quels mouvements il va atteindre son objectif. Les recherches dans ce domaine portent principalement d'une part sur la localisation du véhicule autonome et la cartographie de son environnement, d'autre part sur le contrôle et la navigation autonome de tels véhicules (structure de contrôle, stratégies de commande, planification et navigation).

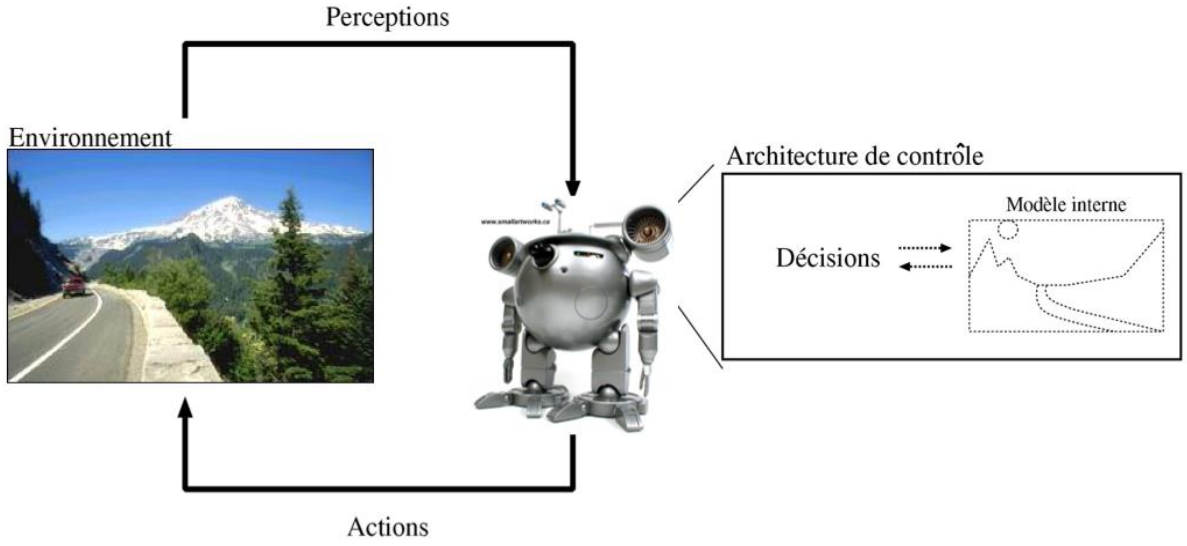


Figure I.1. Les interactions d'un robot avec son environnement.

I.2.2. Pourquoi la robotique mobile ?

D'après tout ce que nous voyons aujourd'hui, il est clair que dans un avenir proche, la robotique occupera une place importante dans notre quotidien. Elle possède de nombreux champs d'applications comme la robotique industrielle ou la robotique de service. Qu'il s'agisse de robot civil ou militaire, il existe désormais des robots capables d'étonnantes prouesses dans de nombreux secteurs : robots-compagnons assistant les personnes à domicile ou en charge de la surveillance et des soins, robots assurant la logistique dans les hôpitaux, robots assistant les industriels dans la réalisation de gestes pénibles et répétitifs, ou encore permettant le développement de prothèses ou d'orthèses intelligentes (Figure I.2).

Figure I.2. Différents types de robots mobiles selon le champ d'application.

Néanmoins, l'intérêt indéniable de la robotique mobile est d'avoir permettre d'augmenter considérablement nos connaissances sur la localisation et la navigation de systèmes autonomes. La gamme des problèmes potentiellement soulevés par le plus simple des robots mobiles à roues en fait un sujet d'étude à part entière et forme une excellente base pour l'étude de systèmes mobiles plus complexes.

I.2.3. Architecture des robots mobiles

L'architecture d'un robot mobile s'articule autour de trois modules fondamentaux (Figure I.3) [1,2] :

-Module locomotion,

-Module perception,

-Module décision.

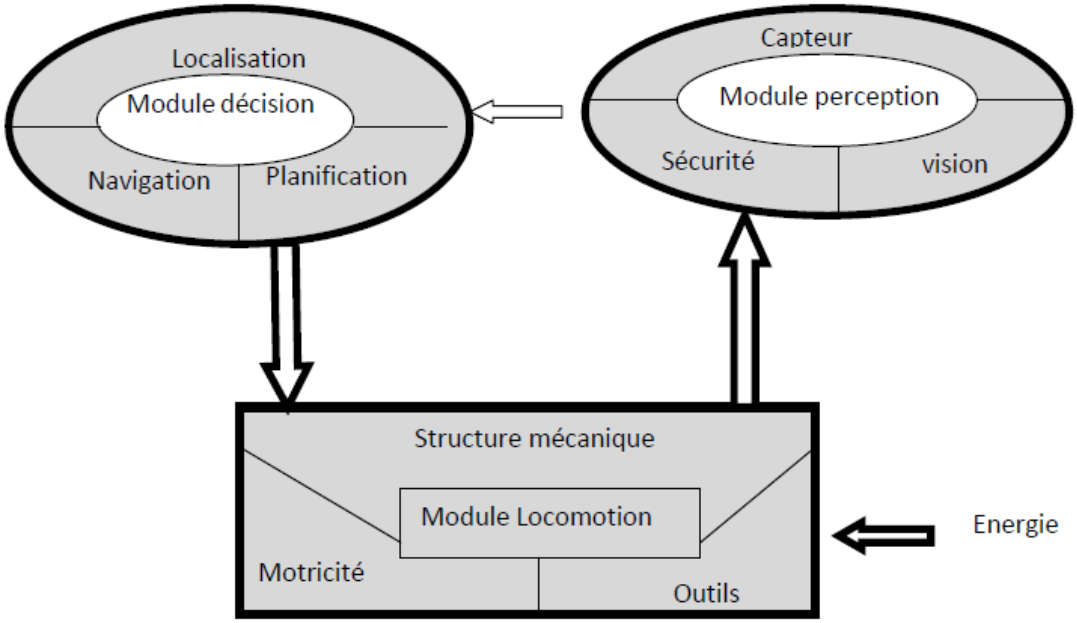


Figure I.3. Structure d'un robot mobile.

- Module locomotion

Le module *locomotion* comporte alors la structure mécanique, motricité et l'énergie utilisée dans le déplacement du robot mobile. Les robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient à roues, marcheurs, sous-marins ou aériens.

Actuellement plusieurs laboratoires de recherche contribuent jusqu'à présent à la construction des différents robots mobiles selon le type de locomotion.

- Module perception

La notion de *perception* en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée.

- Module Décision

Les informations en provenance des différents capteurs doivent être interprétées comme autant d'éléments utiles à la prise de décision sur l'action à faire, le but étant de délivrer les ordres corrects aux actionneurs, bras pinces ou moteurs des roues. C'est lors de cette phase de la conception d'un robot qu'il est nécessaire de lui donner une forme d'intelligence en lui laissant le choix sur l'action à entreprendre. Cette prise de décision est souvent arbitraire au début, mais elle permet de développer une forme d'apprentissage qui tient compte des résultats des décisions précédentes.

I.2.4. Types de plates formes mobiles

Les robots mobiles à roues peuvent être classés selon le type de la plateforme mobile utilisée dans les sous-classes suivantes :

I.2.3.1. Plates forme unicycle

C'est l'une des configurations les plus utilisées pour les robots mobiles d'intérieur par ce qu'elle nécessite un sol très plan et non accidenté.

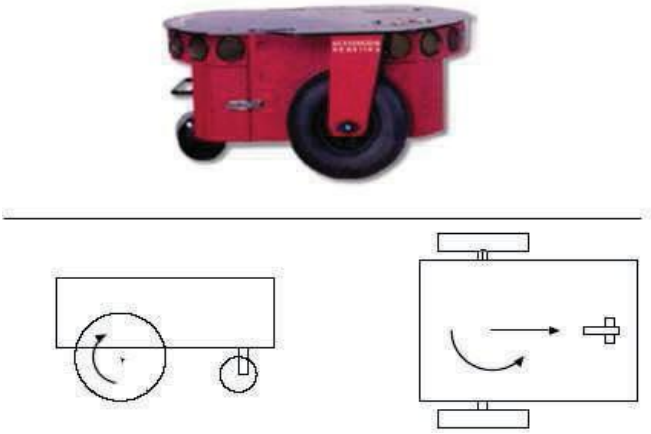


Figure I.4. Exemple de plateforme unicycle [4].

I.2.3.2. Plates formes holonomes

Les systèmes dits *holonome* sont beaucoup plus rares dans notre vie quotidienne. Ils ont une structure mécanique complexe qui leur permet de se déplacer dans toutes les directions sans manœuvre. La société *Nomadic* disparue en l'an 2000 ; a conçu un système mobile holonome : le XR4000 (Figure I.5). Il dispose de 4 roues motrices et directrices montées comme des roues de chariot. La synchronisation des 8 axes (2 par roue, rotation et orientation) est assurée par une carte dédiée basée sur le microcontrôleur Motorola 68332 et des circuits FPGAs et la structure mécanique est composée d'engrenages coniques [1,2].

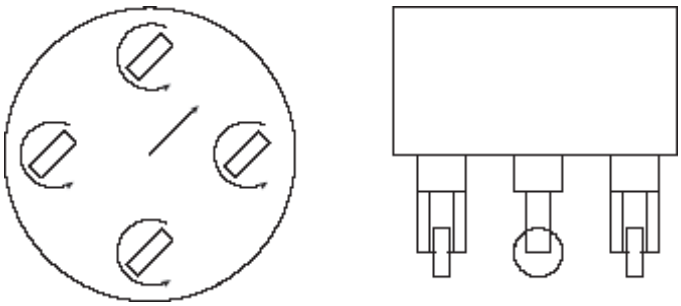


Figure I.5. Exemple de plate-forme holonome [4].

I.2.3.3. Plates formes non-holonomes

Les systèmes mobiles dits *non-holonomes* sont ceux que l'on rencontre le plus dans la vie courante (voiture particulière, bus, camion, …etc.). Ces systèmes ont une structure mécanique relativement simple (des roues motrices, des roues directrices et des roues libres). Une roue peut avoir une, deux ou trois fonctions. Mais tous ces systèmes ont une caractéristique commune : la direction de la vitesse d'entrainement (vitesse linéaire) est imposée par la direction des roues directrices.

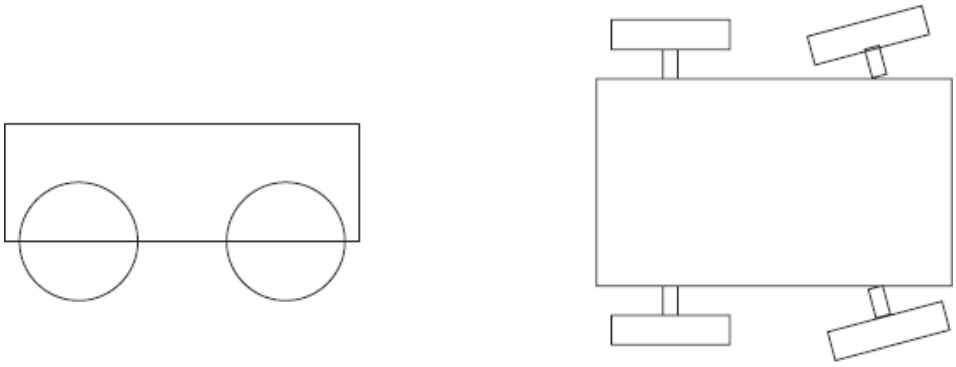


Figure I.6. Exemple de plateforme non holonome [4].

I.2.3.4. Robot omnidirectionnel

Un robot *omnidirectionnel* est un robot qui peut se déplacer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral. L'énorme avantage du robot omnidirectionnel est qu'il est holonome puisqu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande.

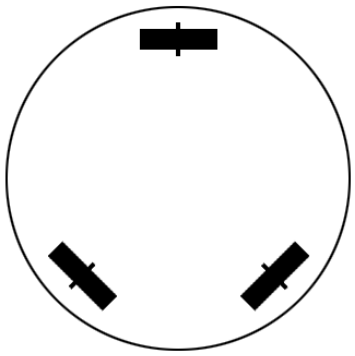


Figure I.7. Robot de type omnidirectionnel [4]

I.3. Les nouvelles approches de la robotique mobile

En 1983, Hans Moravec [5] développe un robot mobile minimaliste, le *Stanford Cart*, équipé d'une tourelle de vision stéréoscopique, Ce robot, bien que très lent, était capable de mettre à jour son modèle du monde et d'y inclure de nouveaux obstacles, sans pour autant recourir à une représentation symbolique. Son système de vision servait à localiser des espaces libres d'obstacles et aucune reconnaissance d'objets en tant que telle n'était réalisée [6].

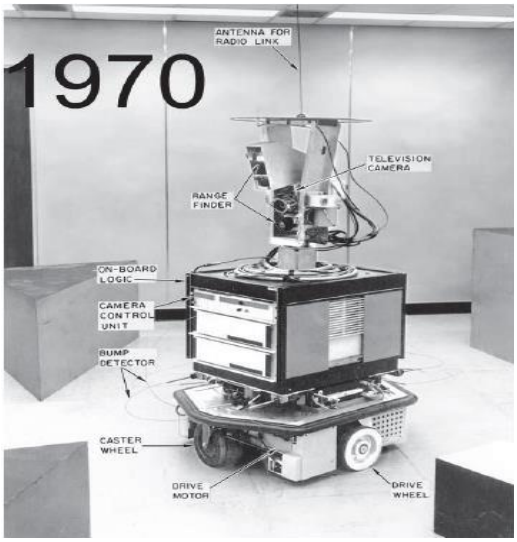


Figure I.8. Le robot Shakey de Stanford en [6].

C'est à cette période que certains chercheurs commencèrent à se douter que l'approche classique, plutôt que de résoudre et de simplifier les problèmes, les compliquait inutilement en concentrant les efforts sur le développement de systèmes de traitement symbolique de l'information. Le problème le plus critique était certainement celui du fondement des symboles, en anglais *"symbol grounding problem"*, qui se réfère à la difficulté de mettre en relation la signification d'un symbole avec l'objet réel ou l'événement qu'il représente. La description d'un symbole au moyen d'autres symboles conduit très souvent à des structures circulaires ou récursives. L'introduction de nouveaux symboles pour définir un symbole existant conduit à la prolifération de ceux-ci, sans pour autant leur attribuer une signification réelle. Les humains sont capables de fonder les symboles sur leurs perceptions sensorielles (c'est le fondement des symboles) : la signification des symboles émerge de l'interaction avec les objets

de l'environnement, elle ne leur est pas inhérente.

Un autre problème lié aux approches classiques est dû à la faible robustesse des systèmes de contrôle à l'égard du bruit et des événements inattendus. L'interaction d'un robot réel avec son environnement comporte un risque élevé de voir celui-ci se perdre s'il est piloté par un système classique. En effet, il lui est très difficile de tenir compte des écarts qui apparaissent entre sa perception du monde et la représentation symbolique qu'il en a, et de les corriger. Ainsi, un système de navigation par amers reposant sur une représentation symbolique se perdra ou ne saura pas réagir de manière appropriée lorsque ses capteurs lui fourniront des mesures inattendues ou entachées de bruit : si un amer perçu ne peut pas être mis en relation avec un symbole connu, ou qu'il est mis en relation avec un symbole incorrect, le système réagira de manière erronée. Un tel système ne peut donc pas être considéré comme entièrement autonome. D'autres approches s'avéraient nécessaires pour faire face aux nombreux problèmes inhérents au monde réel, tels que l'incertitude des connaissances, le bruit entachant les grandeurs mesurées par les capteurs, les modifications dynamiques de l'environnement, la nécessité de répondre rapidement et correctement à dos situations inattendues, etc [6].

Un certain nombre de nouvelles approches, en partie inspirées par les modèles cybernétiques et biologiques, ont donc été proposées :

1. Une approche sans représentations : les représentations internes, de forme purement symbolique, permettent de décrire un phénomène de façon abstraite et n'ont qu'un lien ténu avec la réalité. Une approche sans représentations permet, en s'acquittant d'un niveau d'abstraction élevé, de garder un lien direct entre le robot et le monde physique dans lequel celui-ci évolue.

2. Une approche réactive : pour de nombreuses tâches, la réponse directe et réactive à des événements extérieurs constitue une solution simple et efficace. Elle permet de s'acquitter des représentations symboliques complexes souvent nécessaires aux systèmes de planification classiques. Les comportements d'un robot basé sur une telle approche sont sélectionnés directement en fonction des stimuli perçus. Les comportements qui entraînent une modification du monde (ou une modification de sa perception) entraînent à leur tour l'activation de nouveaux comportements.

3. Une approche distribuée : la majorité des systèmes d'intelligence artificielle classique reposaient sur le principe qui dit que l'intelligence pouvait être assimilée à un processus séquentiel de traitement de l'information, basé principalement sur des inférences logiques.

L'approche distribuée combine les caractéristiques des deux approches décrites ci-dessus et répartit la prise de décision dans des modules séparés. Le choix final de l'action à exécuter résulte de la coopération ou de la compétition des divers modules. L'architecture à subsomption de Brooks est un exemple d'une telle approche, distribuée, réactive et sans représentations, Ces nouvelles approches s'appuient souvent sur des modèles biologiques.

De par leur lien étroit avec les recherches en vie artificielle, elles ouvrent de nouvelles perspectives pour l'intelligence artificielle, Elles ont entre autres donné le jour à la robotique comportementale, à l'approche animât et à la robotique animale.

I.4. L'autonomie d'un robot

L'autonomie par définition est la capacité à résister à des perturbations externes en utilisant les ressources internes. L'autonomie d'un robot est une faculté relative et non absolue, elle est liée aux capacités du robot, aux caractéristiques de l'environnement dans lequel il est plongé, à leurs variations, et enfin aux tâches qu'il doit effectuer. Le concept d'autonomie est complexe lorsqu'on considère l'interaction d'un robot avec son environnement.

Selon Steels [7], un système est autonome s'il développe les lois et les stratégies qui lui permettent de contrôler son comportement. En fait, l'autonomie est une capacité relative et on peut considérer qu'il existe une progression insensible du niveau le plus bas au plus élevé. Elle représente la capacité à choisir une stratégie, en termes de sous-buts pour arriver à un but fixé. La limite basse de l'autonomie est constituée par un système automatique qui s'autorégule en fonction de lois préétablies i.e. qui ne génère pas les lois que les activités de régulation cherchent à satisfaire.

L'autonomie désigne littéralement la capacité d'une entité à « *se gouverner par ses propres lois* ». Cela signifie entre autres le choix de ses buts. Dans la littérature robotique, on peut distinguer trois points de vue :

- L'autonomie au sens fort qui nous ramène à des questions de volonté et de but propre.
- L'autonomie au sens faible désigne la capacité de maintenir sa structure au sein d'un milieu complexe à travers des mécanismes tels que l'auto-organisation, l'évolution, l'adaptation et l'apprentissage.
- L'automatisme ou absence de contrôle extérieur est le sens implicite que l'on attribue

généralement au mot autonomie.

S'adapter consiste pour un robot à modifier son comportement pour faire face à des changements internes ou externes afin de maintenir certaines propriétés. L'adaptation se rapporte toujours à quelque chose : adaptation d'un groupe à la panne d'un individu, adaptation à la présence de passants dans l'environnement…etc. Les bases de l'autonomie sont concrètement les propriétés qu'un robot doit exhiber pour être autonome, par exemple :

-Navigation et localisation pour accéder à tous les points de l'environnement,

-Surveillance, i.e. détection d'évènements anormaux,

-Efficience dans son travail, en adoptant une stratégie de patrouille.

Un robot complètement autonome n'est pas forcément souhaitable. On peut vouloir en prendre le contrôle à certains moments ou intervenir indirectement à travers la modification de certains paramètres. Le terme d'autonomie ajustable désigne justement l'interruption temporaire de l'automatisme, du fait d'un superviseur ou de l'agent lui-même, afin d'accroître l'efficacité du système [8]. Le choix du degré d'automatisme dont doit être doté un agent autonome dépend de l'architecture de contrôleur.

I.4.1. Autonomie du mouvement et autonomie décisionnelle.

L'autonomie du mouvement passe par la détermination des déplacements, de manière planifiée et réactive, et par la commande des déplacements. Si les méthodes mises en jeu sont fortement dépendantes des modèles cinématiques et dynamique des engins considérés, des méthodes génériques commencent à apparaître, surtout pour la planification de trajectoire et la commande référencée sur des éléments de l'environnement. Dans ce contexte, la taille de l'environnement considéré nous amènera plus particulièrement à nous intéresser à l'intégration de méthodes locales de gestion des mouvements avec des méthodes plus globales, qui considèrent l'ensemble des informations disponibles sur l'environnement et la mission à réaliser. Ces derniers aspects relèvent surtout de l'autonomie décisionnelle : il s'agit de déterminer les modalités de déplacement à adopter et les informations à acquérir.

Les systèmes automatiques sont entièrement prévisibles, du moment que leur état interne est connu, alors que les systèmes autonomes ne le sont pas, puisqu'ils sont capables de prendre eux-mêmes des décisions en fonction de critères qui peuvent échapper à l'observateur.

L'autonomie implique la liberté de contrôle. Un système autonome prend ses décisions lui-même, il n'est pas contrôlé par un agent externe. La prise de décision implique une capacité à

évaluer des alternatives en fonction d'un état courant et de l'expérience acquise par le passé [6].

Posée dans le contexte d'environnements larges, éventuellement dynamiques, enrichie par la considération de systèmes multirobots pas nécessairement homogènes, et de contraintes de communication et de gestion des ressources, cette problématique est encore très ouverte. La possibilité d'interactions avec des opérateurs, distants ou non, doit aussi être considérée à différents niveaux au sein des robots. Ces problèmes d'autonomie ajustable passent notamment par le développement de concepts d'organisation des processus décisionnels (architecture de contrôle).

I.4.2. Autonomie envisageable pour les robots actuels

Il ne paraît pas suffisant de viser le développement d'un ensemble de fonctionnalités indépendantes, sous forme de comportements ou de modules, mais un ensemble cohérent où toutes les fonctionnalités puissent être mise en œuvre sur des démonstrateurs dans une même unité de lieu et de temps. La phase d'étude des missions a permis d'extraire des tâches *"robotisables"* que l'on peut décomposer en comportements sensorimoteurs ou en fonctions de surveillance. Les premiers se distinguent des suivants par la présence d'actions motrices. Ces éléments sont listés ci-après :

- faire un déplacement dans une direction et selon une distance donnée ;
- se diriger vers un amer visuel (désigné au robot par un opérateur humain) ;
- suivre un guide visuel au sol (bord de mur, de trottoir, d'accotement, couloir) ;
- s'orienter dans une direction donnée ;
- contourner un obstacle ;
- explorer une zone délimitée ;
- porter ou déposer une charge ;
- collecter un objet ;
- se servir d'un outil spécifique ;
- utiliser des armes non létales ;

I.5. Environnement dynamique et incertain

I.5.1. Notion d'environnement dynamique

Un environnement est dit dynamique s'il comporte des obstacles susceptibles de changer au cours du temps. Soit le cas des obstacles qui se déplacent (un piéton, un véhicule, …), ou ceux qui changent de forme ou de ceux qui apparaître/disparaître (une porte coulissante semble « disparaître » dans le mur quand elle s'ouvre).

I.5.2. Notion d'incertitude

Une information est incertaine, si elle est bruitée (mauvaises conditions de mesures), incomplète (obstruction d'un capteur ou portée limitée, absence d'informations sur l'évolution d'un objet ou d'un phénomène) ou imprécise (les glissements des roues par rapport au sol sont observables mais rarement mesurables avec précision).

I.6. Problématiques de la robotique mobile

La robotique mobile cherche depuis des années à rendre une machine mobile autonome face à son environnement pour qu'elle puisse sans intervention humaine accomplir les missions qui lui sont confiées. Le spectre des missions que les roboticiens veulent voir accomplir par leurs machines est immense : exploration en terrain inconnu, manipulation d'objets, assistance aux personnes handicapées, transport automatisé, etc [9].

On distingue sans trop d'ambiguïté un certain nombre de problèmes en robotique mobile. En plus de pouvoir percevoir globalement son environnement, un robot doit souvent être capable d'identifier des objets, de reconnaître des personnes, de lire des indications, et même de repérer des symboles graphiques. Ces opérations sont effectuées en analysant les perceptions acquises par les capteurs.

Enfin, une autre capacité robotique tout aussi importante que celles énumérées précédemment, est la capacité pour un robot de prendre lui-même ses propres décisions nécessaires pour réaliser et coordonner des missions complexes. Cette capacité est d'une grande importance puisque, pour beaucoup de tâches robotiques, il peut exister de nombreuses façons de les réaliser. Par un raisonnement, le robot doit alors sélectionner les meilleures actions à effectuer pour réussir adéquatement sa mission [10]. Bien évidemment, l'aspect matériel, qui consiste à choisir et dimensionner aussi bien la structure mécanique du système que sa motorisation, son alimentation et l'architecture informatique de son système de contrôle-commande apparaît comme le premier point à traiter. Le choix de la structure est souvent effectué parmi un panel de solutions connues et pour lesquelles on a déjà résolu les

problèmes de modélisation, planification et commande.

Le choix des actionneurs et de leur alimentation est généralement assez traditionnel. La plupart des robots mobiles sont ainsi actionnés par des moteurs électriques à courant continu, alimentés par des convertisseurs de puissance fonctionnant sur batterie. De la même façon, les architectures de contrôle- commande des robots mobiles ne sont pas différentes de celles des systèmes automatiques ou robotiques plus classiques. On y distingue cependant, dans le cas général, deux niveaux de spécialisation, propres aux systèmes autonomes : une couche décisionnelle, qui a en charge la planification et la gestion (séquentielle, temporelle) des évènements et une couche fonctionnelle, chargée de la génération en temps réel des commandes des actionneurs [11].

Au fil des ans, plusieurs solutions ont été proposées afin de donner aux robots la capacité de coordonner leurs missions. L'une d'entre elles est l'utilisation d'outils de planification et navigation dans le domaine de l'intelligence artificielle.

Afin de faire fonctionner un robot mobile, plusieurs modules logiciels sont mis à contribution. Ces modules peuvent servir à interpréter les données perçues par les capteurs afin d'y extraire des informations ou à traiter des commandes de haut niveau et générer d'autres commandes à un niveau inférieur. Les modules les plus fréquemment utilisés sont les modules de localisation, de navigation, de vision, d'audio et de séquencement d'activités du robot.

Un robot mobile est commandé par une boucle de contrôle (ou sens général), cela est illustré à la figure I.4. De façon itérative, celle boucle fait une lecture des données reçues par les capteurs, les interprète, calcule les commandes motrices et les envoie aux actionneurs. Typiquement, cette boucle est exécutée environ dix (10) fois par seconde ; la fréquence peut varier selon les types de capteurs et d'actionneurs utilisés. La boucle de contrôle n'est pas unique ; selon l'architecture utilisée, elle peut être décomposée en plusieurs sous boucles de contrôle agencées de manières différentes [10].

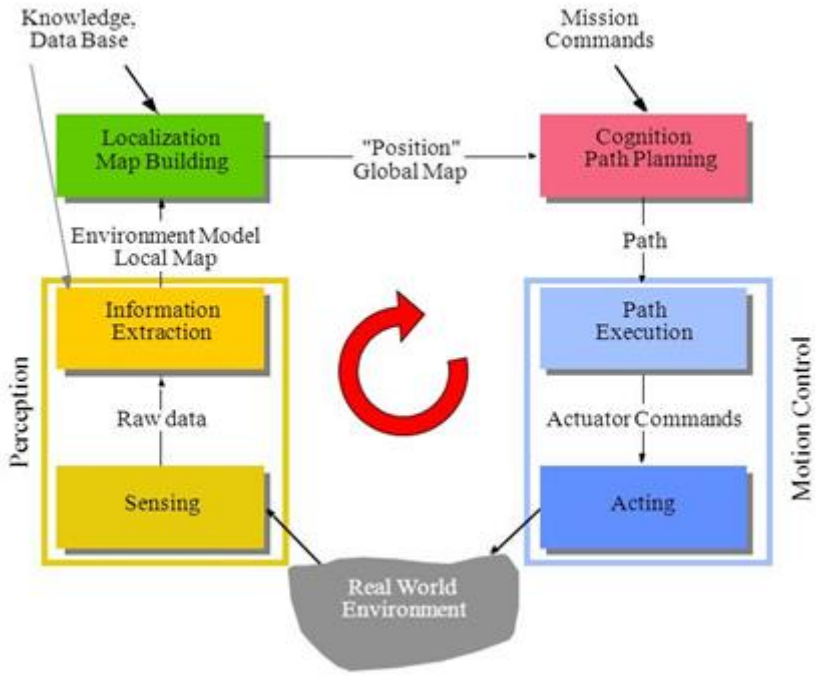


Figure I.9. Structure de commande générale d'un robot mobile.

I.7. Architectures de contrôle

Un robot est donc composé d'un ensemble de modules, chacun étant responsable d'une ou plusieurs fonctions. Un des premiers défis à résoudre est de déterminer comment relier efficacement les différents modules. Pour ce faire, il faut élaborer une architecture de contrôle qui dictera les responsabilités de chacun des modules et comment les informations circuleront entre ces derniers. Depuis les débuts de la robotique, beaucoup d'architectures ont été proposées. Elles peuvent être généralement classées en trois grandes catégories : délibérative, comportementale et hybride.

La définition des différents types d'architecture s'est faite au fil du temps, et s'est faite grâce à l'analyse des points faibles de chaque type d'architecture. La classification des architectures de contrôle repose sur la stratégie utilisée pour contrôler le système et pour atteindre ses objectifs :

- L'application directe des techniques d'intelligence artificielle telles que la planification dans l'école cognitive ;
- L'utilisation des comportements réactifs dans l'école réactive ;
- La centralisation de contrôle du groupe dans un superviseur externe ou la distribution.

I.7.1. Architectures de contrôle cognitives

Dans une architecture de contrôle cognitive, le robot cherche tout d'abord à modéliser les connaissances et les méthodes d'inférence associées permettant d'élaborer les actions que le robot doit entreprendre. Le robot commence par traiter les données recueillies par ses capteurs. Ensuite, il identifie les objets qui sont dans son environnement proche. Puis il construit une représentation de la scène dans son ensemble, et l'utilise pour générer un plan. Après quoi, il calcule au mieux une séquence de commandes vers les effecteurs pour exécuter le plan prévu (voir figure I.10).

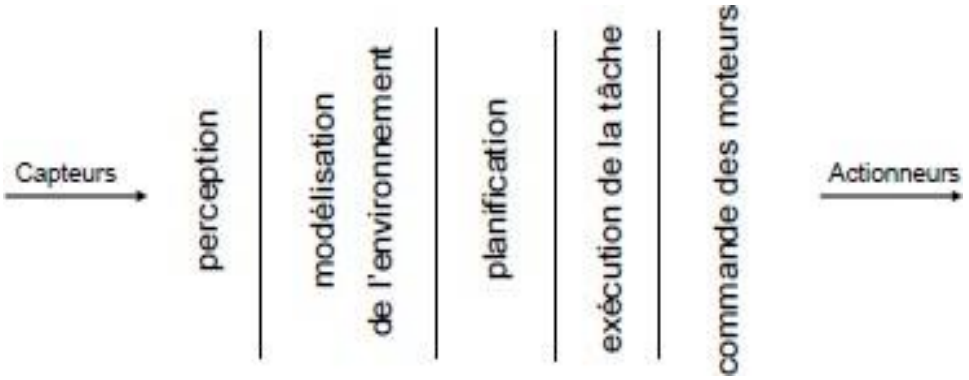


Figure I.10. Architecture de contrôle cognitive.

I.7.2. Architectures de contrôle réactives

Les architectures réactives ont pour but de concevoir et de réaliser un robot, capable d'apporter une réponse immédiate à toute nouvelle modification de l'environnement le concernant. Ces approches sont initialement fondées sur l'étude de l'interaction de l'animal avec son environnement naturel (l'éthologie) de manière à essayer de comprendre les mécanismes mis en œuvre et de les reproduire. Elles sont fondées sur le fait que lorsqu'un être humain évolue dans un environnement, il ne passe pas son temps à modéliser son environnement, et à planifier ses actions, mais qu'au contraire il réagit à de simples stimuli qui lui permettent de se déplacer vers son but tout en faisant face aux obstacles. Ainsi, contrairement aux systèmes hiérarchiques, le comportement global du robot apparaît comme le résultant de plusieurs comportements actifs simultanément. Dans ce type d'architectures il n'y a pas de phase de planification ou de modélisation de l'environnement (voir figure I.11).

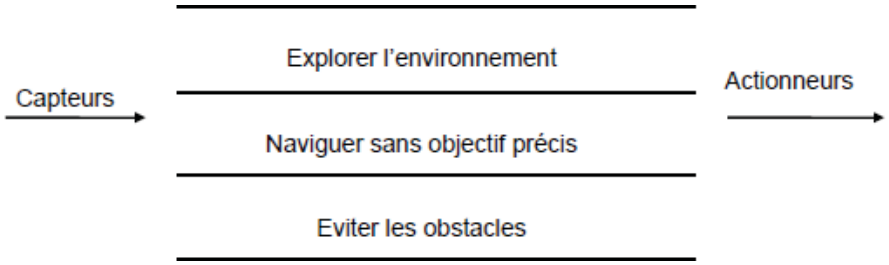


Figure I.11 Architecture de contrôle réactive.

I.7.3. Les architectures de contrôle comportemental réactif

Pour l'élaboration d'architecture de contrôle réactif de plus en plus complexe) propose de décomposer le comportement global du robot en un ensemble d'entrées élémentaires. Cette décomposition a comme principaux objectifs d'avoir une construction aisée de l'architecture de contrôle ainsi qu'une testabilité facile des comportements élémentaires. Ceci est rendu possible en isolant les comportements qui cohabitent dans une même structure de contrôle, impose l'adaptation de mécanismes approprié de coordination entre comportements.

Les architectures de contrôle réactives permettent d'avoir une réponse plus rapide grâce à un lien direct capteurs-actionneurs. Disposer des comportements élémentaires dans le cas des approches comportementales permet de les tester individuellement jusqu'à ce qu'ils soient adaptés à leurs taches. La mise à jour se fait alors simplement en les ajoutant aux comportements déjà existe. [12], [13] ont comparé les caractéristiques des deux types architectures de contrôle. La Figure I.12 illustre la comparaison entre les deux architectures selon les degrés d'intelligences.

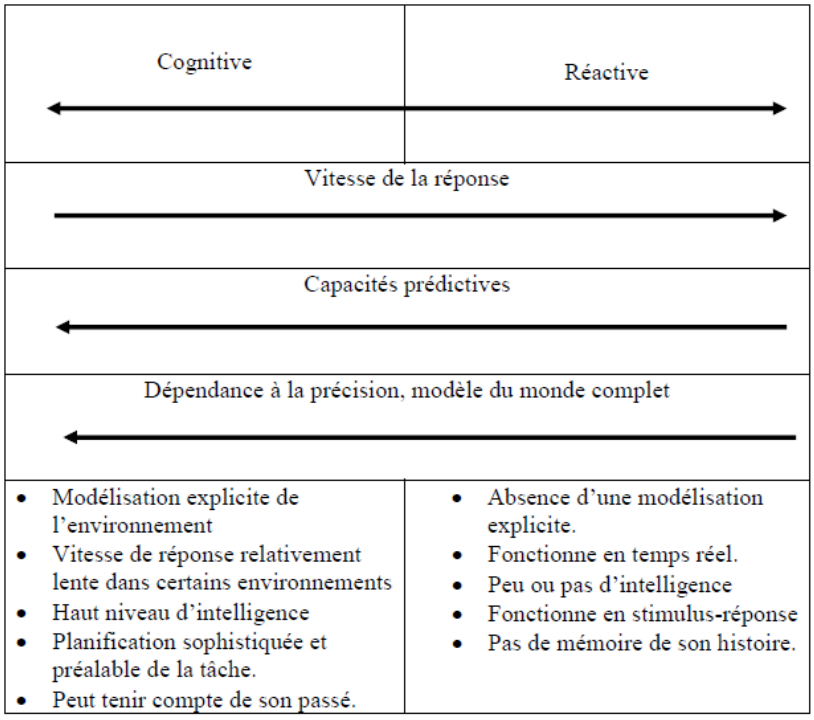


Figure I.12. Contrôle cognitif versus réactif.

I.7.3. Architectures Hybrides

Les approches cognitives et réactives sont diamétralement opposées. Cependant, chacune présente des caractéristiques intéressantes. Pour cela, les chercheurs ont essayé de les combiner en mettant au point des architectures hybrides permettant notamment d'allier des capacités de raisonnement et de décision de haut niveau, s'appuyant sur des représentations abstraites des connaissances, avec des comportements réactifs garantissant robustesse et flexibilité. En effet, l'exécution de plans est basée sur un contrôle réactif (ce que détectent les capteurs permet de réagir immédiatement en cas de nécessité), alors que la formulation de plan(s) est assurée par un système hiérarchique pouvant reposer sur un modèle de l'environnement et des obstacles connus.

Les développements les plus récents convergent vers l'utilisation d'architectures à deux ou trois niveaux. Dans ce type d'architecture, les niveaux bas sont consacrés à un ensemble de comportements sensorimoteurs et à leur mécanisme séquenceur, tandis que le niveau le plus élevé est chargé de la planification des enchaînements de comportements et le niveau intermédiaire, responsable de l'activation des comportements, est basé sur une approche appelée séquencement conditionnel. Ce niveau utilise les plans établis par le niveau supérieur.

Ces plans sont considérés comme des ressources (ou des conseils) et non comme des ordres par le séquenceur. La remise en question d'un plan fait suite à l'échec d'une action détectée par des routines de supervision implantées au-dessus des comportements.

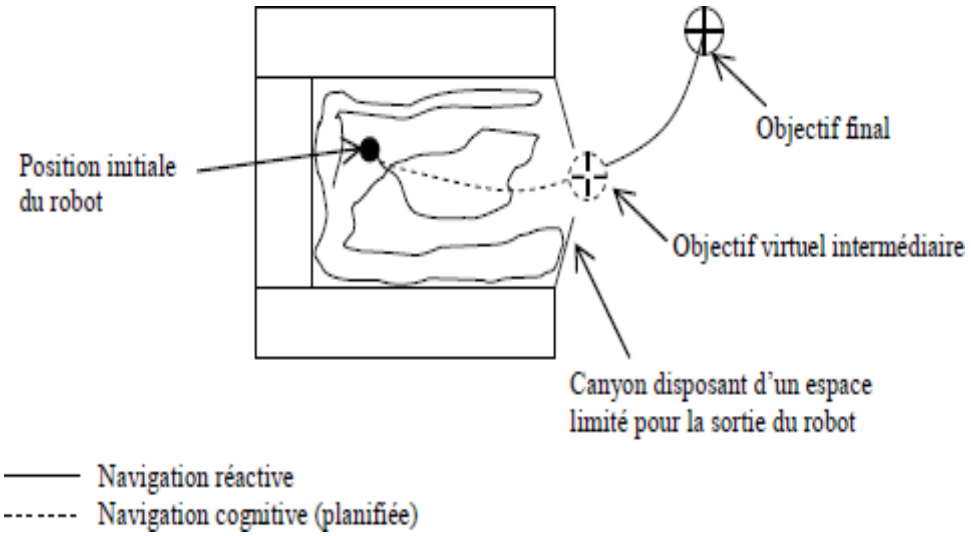


Figure I.13 Architecture de contrôle hybride.

I.8 Conclusion

Le développement de l'intelligence artificielle classique a conduit à la réalisation des premiers robots mobiles, évoluant dans des environnements strictement contrôlés. Leurs actions dans le monde réel n'étaient que l'expression physique d'opérations symboliques réalisées dans un modèle abstrait. Ces robots étaient incapables de réagir de manière appropriée à des perturbations ou à des événements inattendus, ce qui a rendu impraticable leur utilisation dans le monde réel. L'objectif de ce chapitre est de fournir un aperçu général sur la robotique mobile, ses problématiques et les solutions apportées actuelles. Nous avons présenté également un panorama des architectures de contrôle dont la portée va au-delà de leur application en robotique mobile.

La robotique mobile est un domaine dans lequel l'expérience pratique est primordiale. L'approche de la robotique présentée dans ce chapitre est essentiellement celle issue de l'intelligence artificielle. D'autres disciplines très importantes, telles que l'automatique sujet traité dans ce mémoire, sont toutefois impliquées et peuvent présenter un éclairage très

significatif et différent, mais ils ne changent pas fondamentalement les problèmes qui restent à résoudre.

Dans le chapitre suivant nous allons parler ou profond sur la cinématique de quelques types de robots mobiles, chose qui à nous permettre de comprendre le contenu des chapitres qui viennent par la suite.

Chapitre II

Chapitre II : La cinématique du robot mobile

II.1. Introduction

La modélisation mathématique est une étape très importante pour la commande des robots. Deux types de modèles sont généralement utilisés lors de la commande, à savoir : le modèle cinématique et le modèle dynamique. D'après la littérature, on rencontre plusieurs types de robots à savoir : les robots de type unicycle, les robots de type tricycle, et les robots de type voiture (vus au chapitre I). Dans le cadre de notre travail, nous utiliserons un robot de type unicycle à cause de sa simplicité de construction et de ses propriétés cinématiques intéressantes. Plusieurs robots ont fait l'objet d'étude de certains chercheurs, mais un a retenu leur attention : le robot mobile de type unicycle.

II.2. Définitions

On note $R = (O, \vec{x}, \vec{y}, \vec{z})$ un repère fixe quelconque, dont l'axe $\vec{z}$ est vertical et $\acute{R} = (\acute{O}, \vec{\acute{x}}, \vec{\acute{y}}, \vec{\acute{z}})$ un repère mobile lié au robot. On choisit généralement pour $\acute{O}$ un point remarquable de la plate-forme, typiquement le centre de l'axe des roues motrices s'il existe, comme illustrée sur la Figure II.1.

Par analogie avec la manipulation, on appelle situation [14] ou souvent posture [15] du robot le vecteur suivant :

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

Où x et y sont respectivement l'abscisse et l'ordonnée du point $\acute{O}$ dans R et θ l'angle $(\vec{x}, \vec{\acute{x}})$. La situation du robot est donc définie sur un espace $\mathcal{M}$ de dimension $m = 3$, comparable à l'espace opérationnel d'un manipulateur plan.

La configuration d'un système mécanique est connue quand la position de tous ses points dans un repère donnéé est connue [16]. Alors que pour un bras manipulateur cette notion est définie sans ambigüité par les positions angulaires des différentes articulations, on peut, dans le cas d'un robot mobile, donner une vision plus ou moins fine de la configuration, comme on le verra par la suite. Dans tous les cas, on définira la configuration du robot mobile par un vecteur :

$$q = \begin{pmatrix} q_1 \\ q_2 \\ .. \\ q_n \end{pmatrix}$$

de n coordonnées appelées coordonnées généralisées. La configuration est ainsi définie sur un espace N de dimension n, appelée l'espace des configurations.

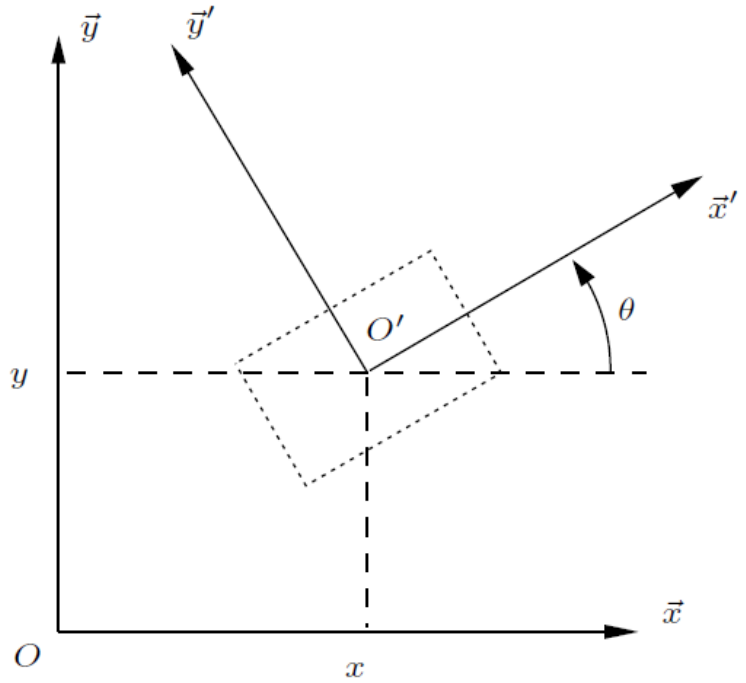


Figure II.1. Repérage d'un robot mobile.

II.3. Roulement sans glissement et contraintes non holonômes

II.3.1. Roulement sans glissement

La locomotion à l'aide de roues exploite la friction au contact entre roue et sol. Pour cela, la nature du contact (régulière, matériaux en contact) a une forte influence sur les priorités du mouvement relatif de la roue par rapport au sol. Dans de bonnes conditions, il y a roulement sans glissement (*r.s.g.*) de la roue sur le sol, c'est-à-dire que la vitesse relative de la roue par rapport au sol au point de contact est nulle. Théoriquement, pour vérifier cette condition, il faut réunir les hypothèses suivantes :

– le contact entre la roue et le sol est ponctuel ;

– les roues sont indéformables, de rayon r.

En pratique le contact se fait sur une surface, ce qui engendre bien évidement de llégers glissements. De même, alors qu'il est raisonnable de dire que des roues pleines sont indéformables, cette hypothèse est largement fausse avec des roues équilibrées de pneus.

Malgréé cela, on supposera toujours qu'il y a *r.s.g*. et, par ailleurs, que le sol est parfaitement plan.

Mathématiquement, on peut traduire la condition de *r.s.g*. sur une roue. Soit P le centre de la roue, Q le point de contact de la roue avec le sol, φ l'angle de rotation propre de la roue et θ l'angle entre le plan de la roue et le plan $(O, \vec{x}, \vec{z})$ comme indiquées à la Figure II.2. La nullité de la vitesse relative $\vec{v}_Q$ roue/sol au point de contact permet d'obtenir une relation vectorielle entre la vitesse $\vec{v}_P$ du centre P de la roue et le vecteur vitesse de rotation $\vec{\omega}$ de la roue :

$$\vec{v}_Q = \vec{v}_P + \vec{\omega} \wedge \overrightarrow{PQ} = \vec{0}$$

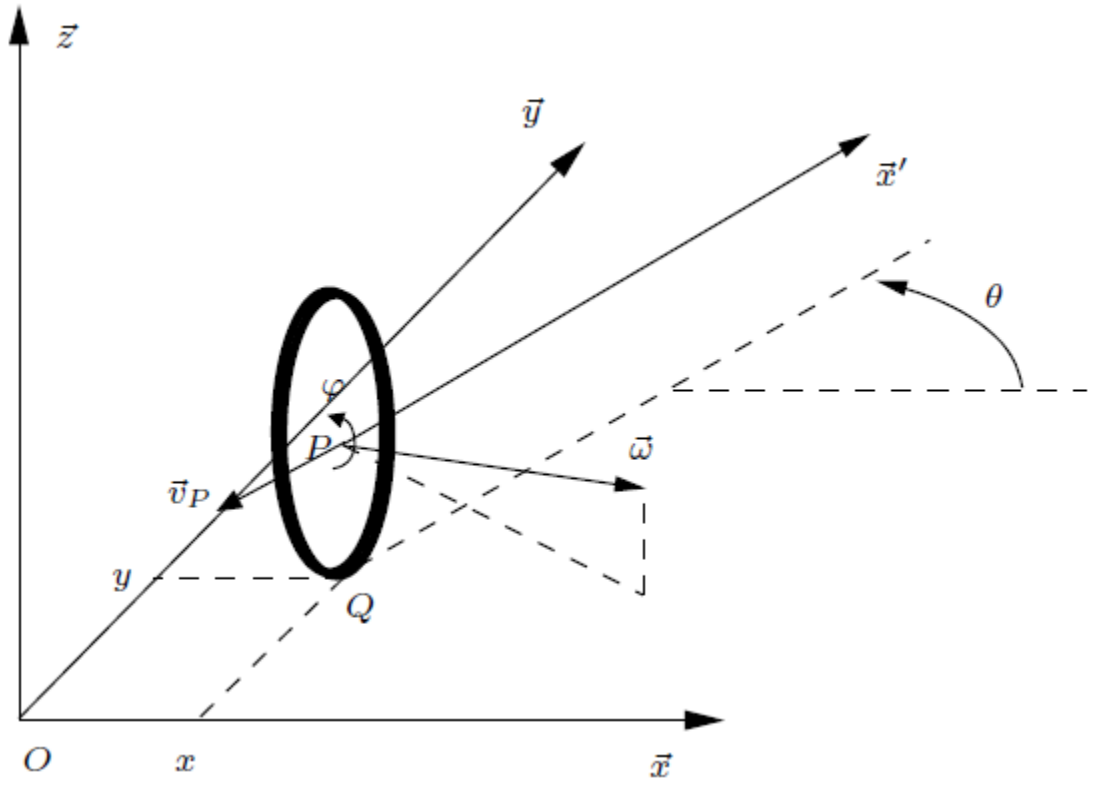


Figure II.2. Caractérisation du roulement sans glissement.

Les points P et Q ont pour coordonnées respectives $(x, y, r)^T$ et $(x, y, 0)^T$. Il vient alors :

$$\dot{x}\vec{x} + \dot{y}\vec{y} + \left(\dot{\theta}\vec{z} + \dot{\varphi}(sin\theta\vec{x} - cos\theta\vec{x})\right) \wedge (-r\vec{z}) = \vec{0},$$

$$(\dot{x} + r\dot{\varphi}cos\theta)\vec{x} + (\dot{y} + r\dot{\varphi}sin\theta)\vec{y} = \vec{0},$$

Ceci nous donne le système de contraintes scalaires :

$$\dot{x} + r\dot{\varphi}cos\theta = 0, \quad \text{(II.1)}$$

$$\dot{y} + r\dot{\varphi}sin\theta = 0, \quad \text{(II.2)}$$

Que l'on puisse transformer pour faire apparaitre les composantes de vitesse dans le plan de la roue d'une part et perpendiculairement à la roue d'autre part :

$$-\dot{x}sin\theta + \dot{y}cos\theta = 0, \quad \text{(II.3)}$$

$$-\dot{x}cos\theta + \dot{y}sin\theta = -r\dot{\varphi}, \quad \text{(II.4)}$$

Ces contraintes traduisent le fait que le vecteur $\vec{v}_P$ soit dans le plan de la roue et ait pour module $r\dot{\varphi}$.

II.3.2. Contraintes non holonômes

Les équations précédentes, caractérisant le *r.s.g*. d'une roue sur le sol, sont des contraintes non holonomes. Nous nous proposons dans ce paragraphe de préciser ce que recouvre ce terme et de caractériser les systèmes non holonomes.

Soit un système de configuration q soumis à des contraintes indépendantes sur les vitesses, regroupées sous la forme (pfaffienne) $A^T(q)\dot{q} = 0$. S'il n'est pas possible d'intégrer l'une de ces contraintes, elle est dite non intégrable ou non holonome. De manière concrète l'existence de contraintes non holonomes implique que le système ne peut pas effectuer certains mouvements instantanément. Par exemple, dans le cas de la roue, il ne peut y avoir de translation instantanée parallèlement à l'axe de la roue. Un tel déplacement n´nécessitera des manœuvres. De même, comme on le sait bien, une voiture ne peut se garer facilement sans effectuer de créneaux.

Il n'est pas évident de dire a priori si une contrainte est intégrable ou non. Pour cela, on a recours `a l'application du théorème de Frobenius, dont une version complète pourra être

trouvée dans un ouvrage de référence de géométrie différentielle [17] ou de commande non-linéaire [18]. Seule la connaissance du crochet de Lie est nécessaire à notre étude.

Pour deux vecteurs $b_i(q)$ et $b_j(q)$, cet operateur est défini par $\left[b_i(q), b_j(q)\right] = \frac{\partial b_i}{\partial q} b_i - \frac{\partial b_i}{\partial q} b_j$.

Théorème II.1 : Soit un système de configuration q, de dimension n, soumis à un ensemble de contraintes indépendantes s'écrivant sous la forme $A^T(q)\dot{q} = 0$. Soit $B(q) = [b_1(q), b_2(q), \ldots., b_m(q)\,]$ une matrice de rang plein m, orthogonale à $A(q)$ sur tout l'espace des configurations. Soit enfin l'algèbre de Lie de dimension p, avec$m \leq p \leq n$, engendrée par l'ensemble des colonnes de $B(q)$, auxquelles s'ajoutent les crochets de Lie successifs formés à partir de ces colonnes, à condition qu'ils augmentent la dimension de l'algèbre. Alors, parmi les contraintes auxquelles est soumis le système, $n - p$ sont intégrables.

Prenons l'exemple de la roue dont on a écrit le modèle précédemment. Sa configuration est entièrement définie par sa configuration $q = (x\ y\ \theta\ \varphi)^T$. D'après (II.3) et (II.4) on peut déterminer :

$$A(q) = \begin{pmatrix} -\sin\theta & \cos\theta \\ \cos\theta & \sin\theta \\ 0 & 0 \\ 0 & r \end{pmatrix}$$

Et on déduire :

$$B(q) = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \\ -\frac{1}{r} & r \end{pmatrix}$$

Matrice formée par les colonnes indépendantes $b_1(q) = \left(\cos\theta\ \sin\theta\ 0\ -\frac{1}{r}\right)^T$ et $b_2(q) = (0\ 0\ 1\ r)^T$et qui vérifie bien $A^T(q)B(q) = 0$ pour tout q. On calcule alors les crochets de Lie successifs à partir de $b_1(q)$ et $b_2(q)$:

$$[b_1(q), b_2(q)] = (\sin\theta \ \ -\cos\theta \ \ 0 \ \ 0)^T \qquad \text{(II.5)}$$

$$[b_1(q), b_2(q)] = 0$$

$$\left[b_2(q), [b_1(q), b_2(q)]\right] = (\cos\theta \ \ \sin\theta \ \ 0 \ \ 0)^T \qquad \text{(II.6)}$$

On constate que $b_1(q)$; $b_2(q)$; $[b_1(q), b_2(q)]$et $\left[b_2(q), [b_1(q), b_2(q)]\right]$ engendrent une algèbre de Lie de dimension 4, égale à celle de q. On arrête donc le calcul des crochets et on peut conclure qu'il n'existe pas de contrainte intégrable. Les contraintes (II.3) et (II.3) sont donc des contraintes non holonomes.

II.4. Les grandes classes de robots mobiles et leurs modèles

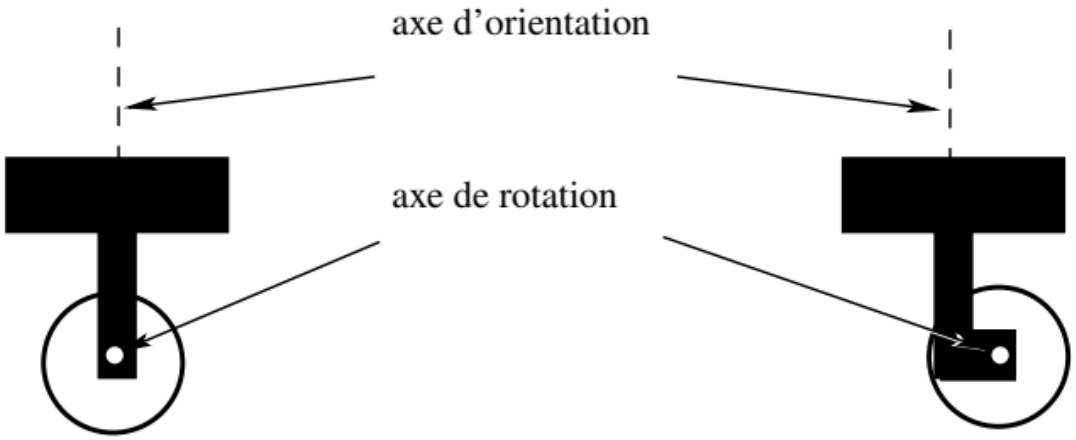
II.4.1. Disposition des roues et centre instantané de rotation

C'est la combinaison du choix des roues et de leur disposition qui confère à un robot son mode de locomotion propre. Sur les robots mobiles, on rencontre principalement trois types de roues (voir Figure II.3) :

– les *roues fixes* dont l'axe de rotation, de direction constante, passe par le centre de la roue ;

– les *roues centrées orientables*, dont l'axe d'orientation passe par le centre de la roue ;

– les *roues décentrées orientables*, souvent appelées *roues folles*, pour lesquelles l'axe d'orientation ne passe pas par le centre de la roue.



(a) Roues fixes (axe d'orientation fixe) et roues centrées orientables.

(b) Roues decentrées orientables.

Figure II.3. Les principaux types de roues des robots mobiles.

De manière anecdotique on rencontrera aussi des systèmes particuliers, tels que les roues suédoises, les roues à plusieurs directions de roulement, etc. Bien évidemment, pour un ensemble de roues donné, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages. Par exemple, un robot équipé de deux roues fixes non parallèles ne pourrait pas aller en ligne droite ! Pour qu'une disposition de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantanée.

Ce point, lorsqu'il existe, est appelé *Centre Instantané de Rotation* (CIR). Les points de vitesse nulle lies aux roues se trouvant sur leur axe de rotation, il est donc nécessaire que le point d'intersection des axes de rotation des différentes roues soit unique. Pour cette raison, il existe en pratique trois principales catégories de robots mobiles à roues, que l'on va présenter maintenant.

II.4.2. Robots mobiles de type unicycle

II.4.2.1. Description

On désigne par *unicycle* un robot actionne par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Le schéma des robots de type unicycle est donné à la Figure II.4. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées.

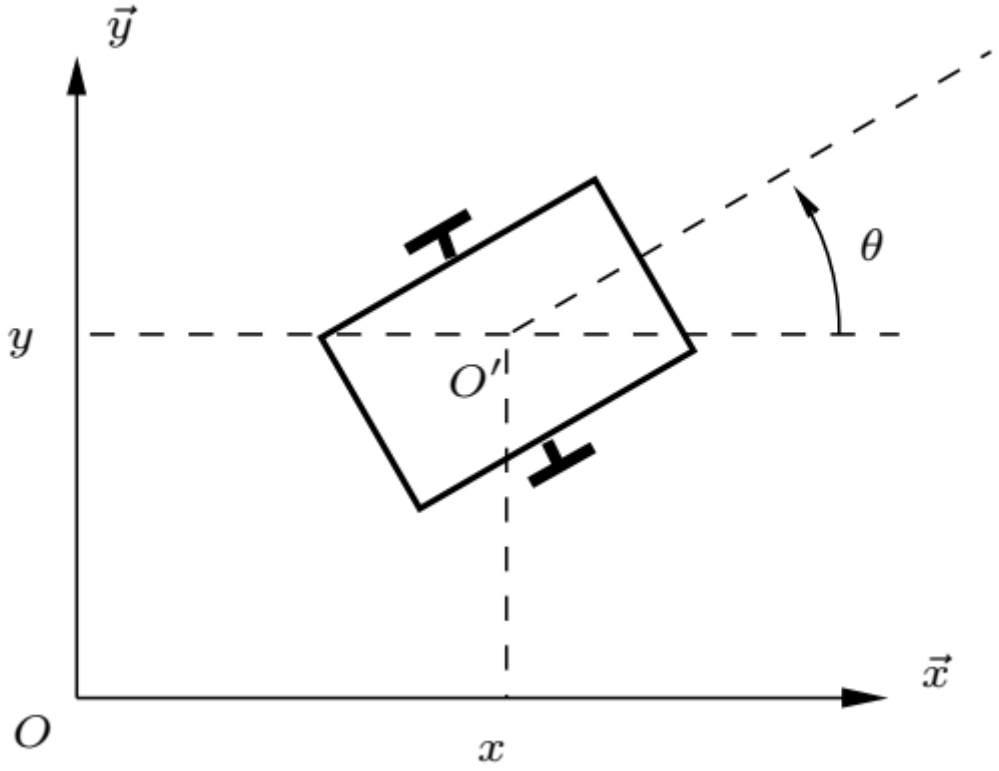


Figure II.4. Robot mobile de type unicycle.

Ce type de robot est très répandu en raison de sa simplicité de construction et de propriétés cinématiques intéressantes. La Figure II.5 présente différents robots de type unicycle, depuis Hilare, en 1977, jusqu'aux modelés actuels, qui, à l'instar du robot Khepera, tendent parfois vers l'extrême miniaturisation.

Ci-dessous, sont présentés comme exemples trois types de robots (célèbres) mobiles de type unicycle :

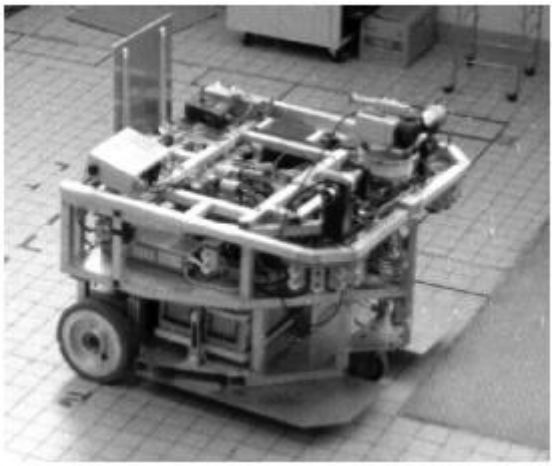


Figure II.5. Robot mobile *Hilare*, LAAS-CNRS, Toulouse, 1977 [19].

Ex.1. Informations techniques (robot Hilare, Figure II.5) ;

- Entraînement : batteries au plomb 24V, 2 moteurs DC avec codeurs incrémentaux,
- Calculateur : 4 processeurs Intel 80286, pas d'OS, multi-bus, modem série radio à 9600 bauds,
- Capteurs : odométrie, 16 capteurs US, un télémètre laser,
- Dimensions (L × l × h) : 80 cm × 80 cm × 60 cm,
- Poids : 400 kg.

Figure II.6. Robot mobile *Pioneer P3-DX*, ActiveMedia Robotics, 2004 [20].

Ex.2. Informations techniques (*Pioneer P3-DX*, ActiveMedia Robotics, 2004, Figure II.6);

- Entraînement : batteries 252 W h, 2 moteurs DC avec codeurs incrémentaux,
- Calculateur : microcontrôleur Hitachi HS-8, I/O Bus, 2 ports série,

- Capteurs : odométrie, 8 capteurs US en façade + options (dumpers, télémètre laser, gyroscope),
- Autonomie : 24 - 30 h,
- Vitesse : maximum 1,6 m/s,
- Dimensions (L × l × h) : 44 cm × 38 × 22 cm,
- Poids : 9 kg (charge admissible : 23 kg).

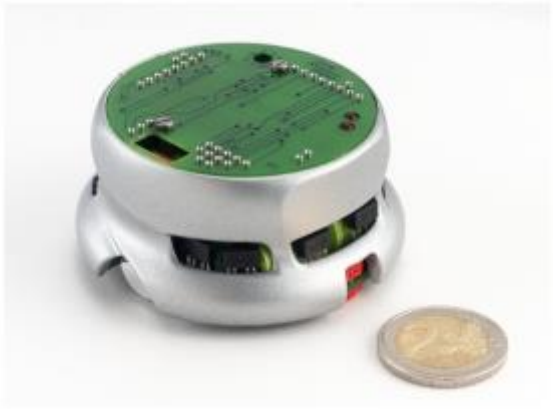


Figure II.7. Robot mobile *Khepera II*, K-team, EPFL, Lausanne, 2002 [21].

Ex.3. Informations techniques (*Khepera II*, K-team, EPFL, Lausanne, 2002, Figure II.7) ;

- Entraînement : adaptateur secteur ou accus NiMH, 2 moteurs DC avec codeurs incrémentaux,
- Calculateur : processeur Motorola 68331 @25 MHz, 512 Ko RAM, 512 Ko mémoire Flash programmable par le port série, communication par port série jusqu'à 115 Kbauds (limitée à 9600 bauds en liaison radio), 3 entrées analogiques 0 à 4.3 V, 8 bits,
- Capteurs : 8 capteurs infrarouges de luminosité et de proximité (100 mm de portée) et un détecteur de batteries faibles,
- Autonomie : 1 h en fonctionnement continu, l'ajout d'extensions diminuant l'autonomie,
- Vitesse: maximum 1 m/s, minimum 0; 02 m/s,
- Dimensions (D × h) : 7 cm × 3 cm,
- Pois: 80 g (charge admissible: 250 g).

II.4.2.2. Modélisation

Centre instantané de rotation : Les roues motrices ayant même axe de rotation, le CIR du robot est un point de cet axe. Soit ρ le rayon de courbure de la trajectoire du robot, c'est-à-dire la distance du CIR au point *O'* (voir Figure II.8). Soit *L* l'entre-axe et ω la vitesse de rotation du robot autour du CIR. Alors les vitesses des roues droite et gauche, respectivement notées v_d et v_g et définies à la Figure II.8, vérifient :

$$v_d = -r\dot{\varphi}_d = (\rho + L)\omega \quad \text{(II.6)}$$

$$v_g = r\dot{\varphi}_g = (\rho - L)\omega \quad \text{(II.7)}$$

Ce qui permet de déterminer ρ et ω à partir des vitesses des roues :

$$\rho = L\frac{\dot{\varphi}_d - \dot{\varphi}_g}{\dot{\varphi}_d + \dot{\varphi}_g} \quad \text{(II.8)}$$

$$\omega = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L} \quad \text{(II.9)}$$

L'équation (2.8) permet de situer le CIR sur l'axe des roues. Par ailleurs ces équations expliquent deux propriétés remarquables du mouvement des robots de type unicycle : si $\dot{\varphi}_d = -\dot{\varphi}_g$, le robot se deplace en ligne droite ; et si $\dot{\varphi}_d = \dot{\varphi}_g$, alors le robot effectue une rotation sur lui-même. L'utilisation de ces deux seuls modes de locomotion, bien que limitée, permet de découpler les mouvements et de fournir une solution simple pour amener le robot d'une posture a une autre. C'est sans doute là une des raisons du succès de ce type de robots. Pour élaborer une stratégie plus fine de déplacement, il est cependant intéressant de savoir comment la posture du robot est reliée à la commande de ` ses roues.

II.4.2.3. Choix de la commande

En ce qui concerne la commande, si l'on se contente de traiter le cas cinématique, on peut considérer que celle-ci est donnée, au plus bas niveau, par les vitesses de rotation des roues. Ceci étant, on préfère généralement exprimer cette commande par la vitesse longitudinale du robot, notée v (en *O'*) et sa vitesse de rotation $\dot{\theta}$ (autour de *O'*). Il y a en effet équivalence entre les deux représentations. D'une part, on a :

$$v = \frac{v_d + v_g}{2} = \frac{r(\dot{\varphi}_g - \dot{\varphi}_d)}{2} \quad \text{(II.10)}$$

D'autre part, la vitesse de rotation du robot est égale à la vitesse de rotation autour du CIR [25]:

$$\omega = \dot{\theta} = -\frac{r(\dot{\varphi}_g + \dot{\varphi}_d)}{2L} \quad \text{(II.11)}$$

Conformément à l'équation (2.9). On montre que ces relations sont parfaitement inversibles et qu'il y a ainsi équivalence entre les couples $(\dot{\varphi}_d, \dot{\varphi}_g)$ et (v, ω). Desormais, on utilise plutôt ce dernier couple de grandeurs, plus parlantes, quitte à calculer ensuite les angles ou vitesses de consigne des asservissements des roues.

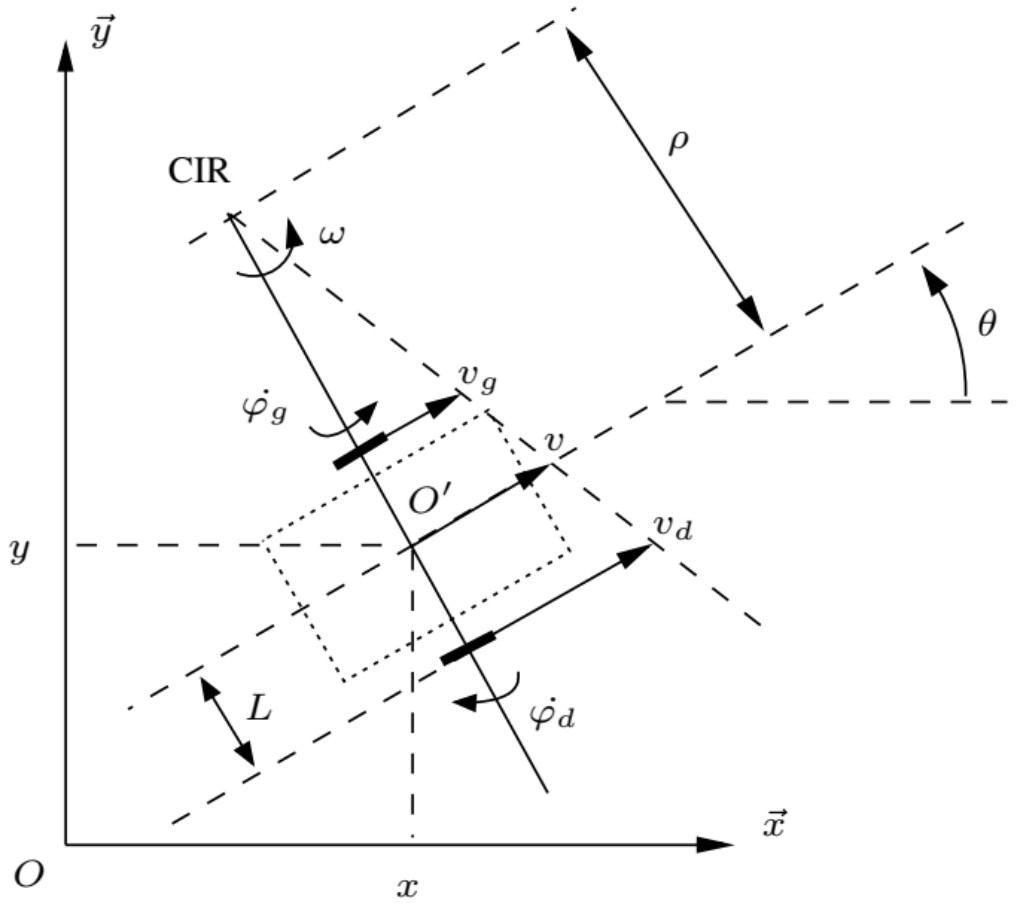


Figure II.8. Centre instantané de rotation d'un robot de type unicycle.

II.4.2.4. Modèle cinématique en posture

Relier la dérivée de la posture à la commande $u = (v, \omega)^T$ est facile. Une simple considération géométrique donne :

$$\dot{x} = v\,cos\,\theta$$

$$\dot{y} = v\,sin\,\theta$$

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

Ce qui s'écrit, sous forme matricielle :

$$\dot{\xi} = C(q)u \tag{II.12}$$

Avec ;

$$C(q) = \begin{pmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \end{pmatrix} \tag{II.13}$$

Ce modèle est appelé *modèle cinématique en posture* du robot [15].

II.4.2.5. Modèle cinématique en configuration

Considérons maintenant l'expression des conditions de *r.s.g.* des deux roues. On transpose les équations (II.1) et (II.2) aux deux roues de l'unicycle. Il vient alors :

$$\begin{cases} \dot{x} + L\dot{\theta}cos\,\theta + r\dot{\varphi}_d cos\,\theta = 0 \\ \dot{y} + L\dot{\theta}sin\,\theta + r\dot{\varphi}_d sin\,\theta = 0 \\ \dot{x} - L\dot{\theta}cos\,\theta - r\dot{\varphi}_g cos\,\theta = 0 \\ \dot{y} - L\dot{\theta}sin\,\theta - r\dot{\varphi}_g sin\,\theta = 0 \end{cases}$$

En choisissant $q = \left(x\ y\ \theta\ \varphi_d\ \varphi_g\right)^T$ comme vecteur de configuration, ces contraintes se regroupent sous la forme matricielle $A^T(q)\dot{q} = 0$, avec :

$$A^T(q) = \begin{pmatrix} 1 & 0 & Lcos\,\theta & rcos\,\theta & 0 \\ 0 & 1 & Lsin\,\theta & rsin\,\theta & 0 \\ 1 & 0 & -Lcos\,\theta & 0 & -rcos\,\theta \\ 0 & 1 & -Lsin\,\theta & 0 & -rsin\,\theta \end{pmatrix}$$

On constate que $A(q)$ n'est pas de rang plein. On peut donc, sans perte d'information, supprimer la dernière ligne et réécrire les contraintes avec :

$$A^T(q) = \begin{pmatrix} 1 & 0 & Lcos\,\theta & rcos\,\theta & 0 \\ 0 & 1 & Lsin\,\theta & rsin\,\theta & 0 \\ 1 & 0 & -Lcos\,\theta & 0 & -rcos\,\theta \end{pmatrix}$$

Par ailleurs il résulte des relations (2.9) et (2.11) la contrainte :

$$\dot{\theta} = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L}$$

Qui s'intègre en :

$$\theta = -\frac{r(\varphi_d + \varphi_g)}{2L} + \theta_0$$

Où θ_0 est la valeur de θ a l'initialisation (les angles des roues, généralement mesurés par des codeurs incrémentaux, étant alors choisis nuls). Cette contrainte intégrable est donc en fait une contrainte holonome. On peut donc éliminer une des quatre variables constituant le vecteur q choisi initialement, qui n'était donc pas constitué de grandeurs indépendantes (et n'était donc pas un vecteur de configuration au sens strict. . .). En prenant maintenant $q = (x\ y\ \theta\ \varphi_d\ )^T$ on trouve :

$$A^T(q) = \begin{pmatrix} 1 & 0 & Lcos\,\theta & rcos\,\theta \\ 0 & 1 & Lsin\,\theta & rsin\,\theta \end{pmatrix} \quad \text{et} \quad B(q) = \begin{pmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \\ -\frac{1}{r} & -\frac{L}{r} \end{pmatrix}$$

En distinguant, d'après ce qui précède, que :

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$
$$\dot{\varphi}_d = -\frac{v}{r} - \frac{L\omega}{r}$$

On constate que, sous forme vectorielle, la dérivée du vecteur de configuration s'écrit :

$$\dot{q} = B(q)u \qquad \text{(II.14)}$$

Avec pour vecteur de commande cinématique $u = (v\ \omega)^T$. La matrice $B(q)$ représente donc le *modèle cinématique (en configuration)* du robot mobile, ce résultat pouvant se généraliser [22]. On notera que la connaissance de ce modèle n'est pas toujours utile du point de vue pratique. Dans le cas de l'unicycle, étant donné que l'on cherche essentiellement à contrôler la posture du robot et que la variable φ_d n'apparait pas dans les équations régissant la dérivée de la posture, on se contentera généralement du modèle simplifie :

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega\end{aligned} \qquad \text{(II.15)}$$

On remarque que dans ce cas précis, le modèle simplifié est le modèle cinématique en posture. Il est à noter que, ce n'est cependant pas le cas pour tous les robots mobiles.

II.4.3. Robots mobiles de type tricycle et de type voiture

Ces robots partagent des propriétés cinématiques proches, raison pour laquelle on les regroupe ici.

II.4.3.1. Description

Considérons tout d'abord le cas du *tricycle*, représenté à la Figure II.9. Ce robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. De ce point dévoue, il est donc très proche d'une voiture. C'est d'ailleurs pour cela que l'on étudie le tricycle, l'intérêt pratique de ce type de robot (peu stable !) restant limite.

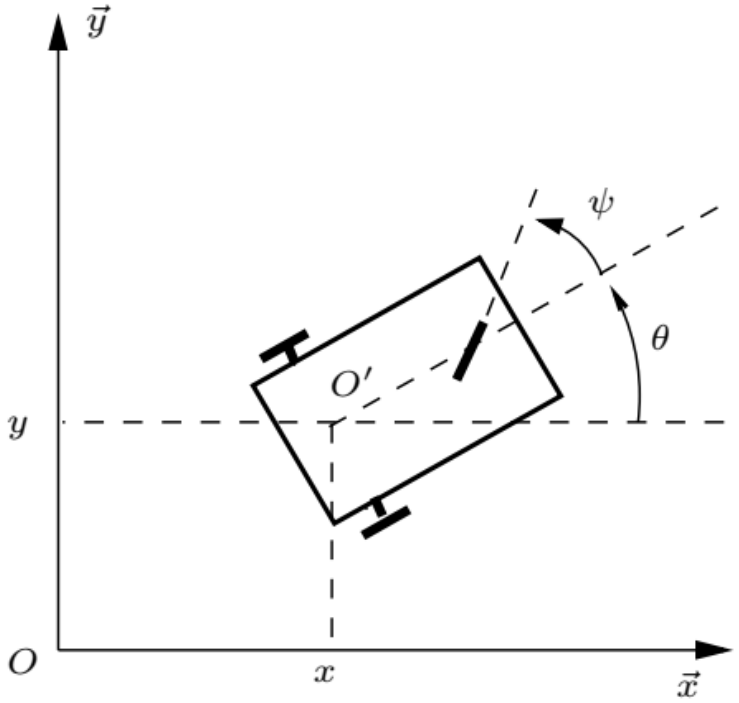


Figure II.9. Robot mobile de type tricycle.

Le cas du robot *de type voiture* est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une. Cela va de soi, on rencontre beaucoup plus souvent ce type de systèmes. On parle de robot dès lors que la voiture considérée est autonome [23,24], donc sans chauffeur ni télé-pilotage. Il s'agit là d'un des grands défis issus de la robotique mobile. Deux réalisations sont montrées à la Figure II.10, basées sur des voitures de série instrumentées.

Figure II.10. Projets de voitures autonomes à l'université de Carnegie Mellon [23,24].

II.4.3.2. Modélisation

Considérons tout d'abord le cas tricycle. Le CIR du robot se situé à la rencontre des axes des roues fixes et de la roue orientable, comme cela est représenté à la Figure II.11. On peut déterminer ρ de manière géométrique à partir de l'angle d'orientation de la roue avant et ω à partir de la vitesse linéaire v du véhicule (vitesse en *O'*) et de ρ :

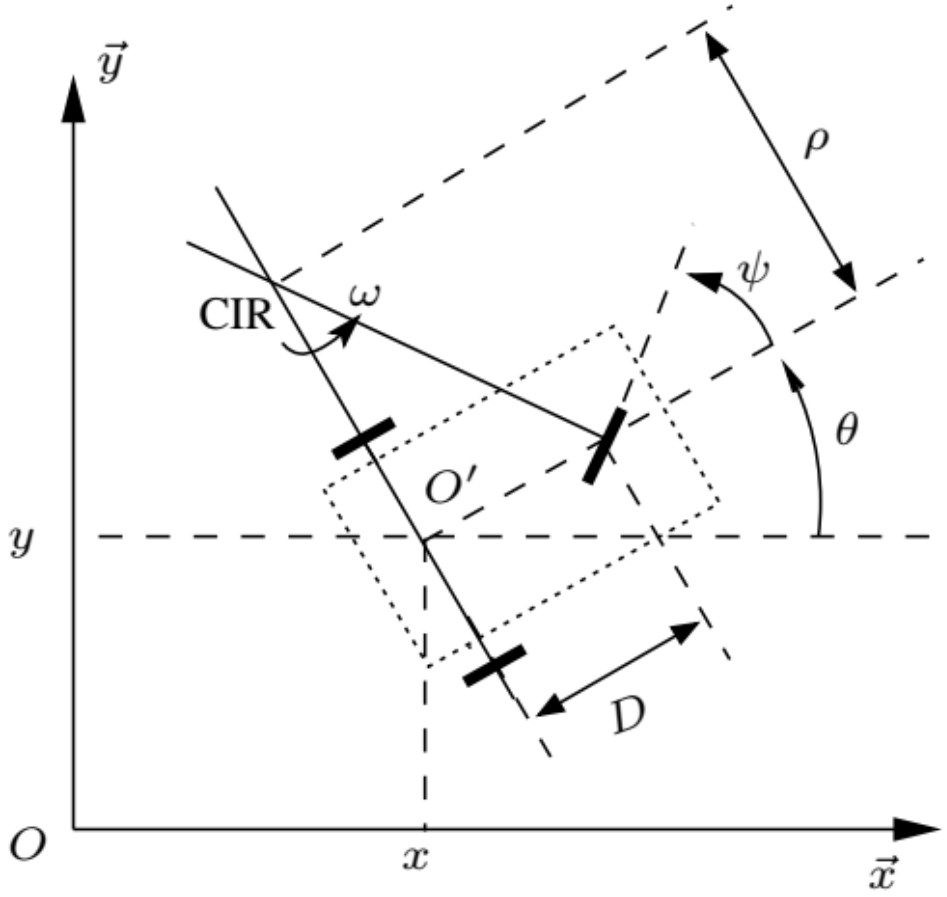


Figure II.11. Un robot mobile de type tricycle et son CIR.

$$\rho = \frac{D}{tan(\varphi)} \tag{II.16}$$

$$\omega = \frac{v}{D} tan(\varphi) \tag{II.17}$$

Ce type de robot peut se diriger en ligne droite pour $\varphi = 0$ et théoriquement tourner autour du point *O'* (on pourrait dire sur place) pour $\varphi = \frac{\pi}{2}$. Neanmoins, le rayon de braquage de la roue orientable, généralement limité, impose le plus souvent des valeurs φ de telles que $-\frac{\pi}{2} < \varphi < \frac{\pi}{2}$, interdisant cette rotation du robot sur lui-même.

L'écriture des contraintes sur chacune des roues et un raisonnement similaire à celui suivi dans le cas de l'unicycle permettent de déterminer les modèles cinématiques des robots de type tricycle. Toutefois, par un simple raisonnement géométrique, on établit les équations :

$$\begin{aligned} \dot{x} &= v cos\ \theta \\ \dot{y} &= v sin\ \theta \\ \dot{\theta} &= \frac{v}{D} tan(\varphi) \\ \dot{\varphi} &= \eta \end{aligned} \tag{II.18}$$

Où $u = (v\ \eta)^T$ est le vecteur de commande cinématique, η représentant la vitesse d'orientation imposée à la roue orientable. Ces équations sont celles du modèle cinématique en configuration simplifie, la configuration associée $q = (x\ y\ \theta\ \varphi)^T$ ne décrivant pas les rotations propres des différentes roues.

Comme on l'a vu précédemment, l'existence d'un CIR unique impose que les axes des roues du robot soient concourants. Dans le cas du robot de type voiture, cela impose aux roues du train avant de n'avoir pas la même orientation, comme illustré a la Figure II.12.

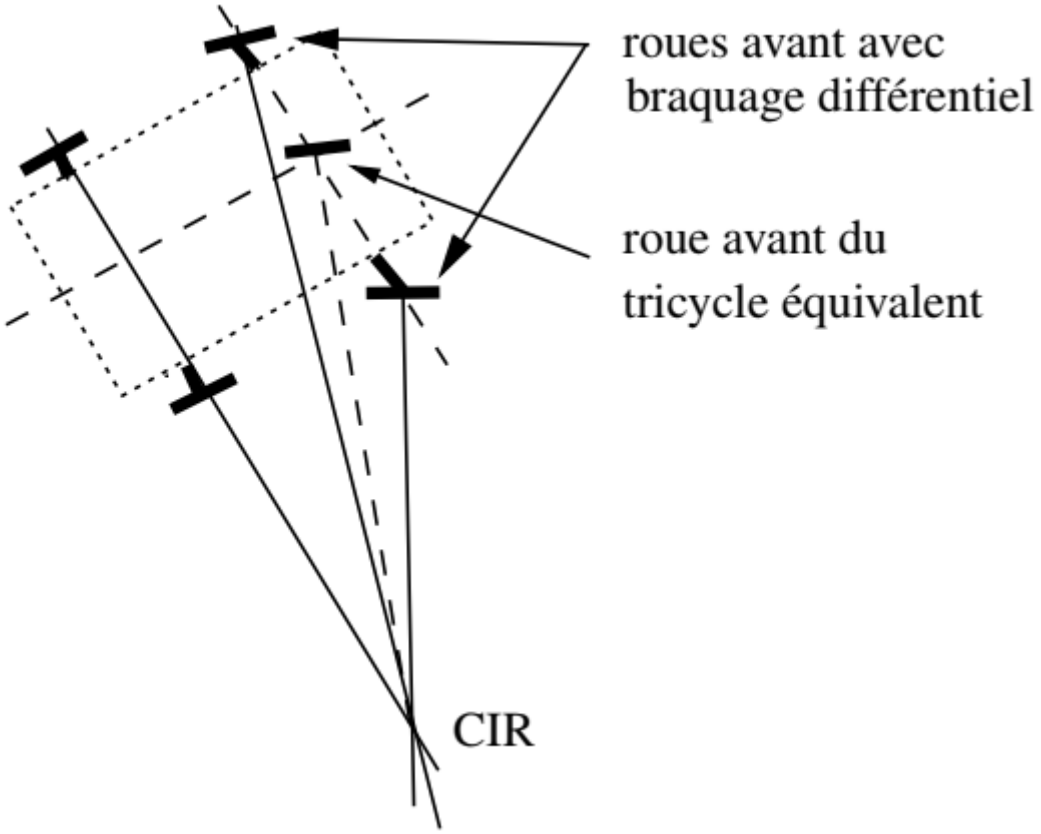


Figure II.12. Un robot mobile de type tricycle et son CIR.

Ainsi, les roues avant d'un robot de type voiture (et a fortiori d'une voiture) ne sont pas parallèles. Le roulement idéal, assurant que le CIR est bien unique, est réalisé sur une voiture par un système de braquage différentiel (dit d'Ackerman). Par ailleurs, les trajectoires des roues n'ayant pas même rayon de courbure, leurs vitesses sont également différentes (et liées évidemment).

L'équivalence entre tricycle et voiture est facile à montrer. Il suffit pour cela de figurer une roue virtuelle qui transformerait un robot de type voiture en tricycle en plaçant la roue orientable du tricycle au centre de l'axe des roues avant de la voiture, orientée de sorte que le CIR reste inchangé, conformément à la Figure II.12.

II.4.4. Robots mobiles omnidirectionnels

II.4.4.1. Description

Un robot mobile est dit *omnidirectionnel* si l'on peut agir indépendamment sur les vitesses : vitesse de translation selon les axes $\vec{x}$ et $\vec{y}$ et vitesse de rotation autour de $\vec{z}$.

D'un point de vue cinématique on montre que cela n'est pas possible avec des roues fixes où des roues centrées orientables [15]. On peut en revanche réaliser un robot omnidirectionnel en ayant recours à un ensemble de trois roues décentrées orientables où de trois roues suédoises disposées aux sommets d'un triangle équilatéral (voir Figure II.13). Du point de vue de la transmission du mouvement, ceci ne va pas sans poser de problème.

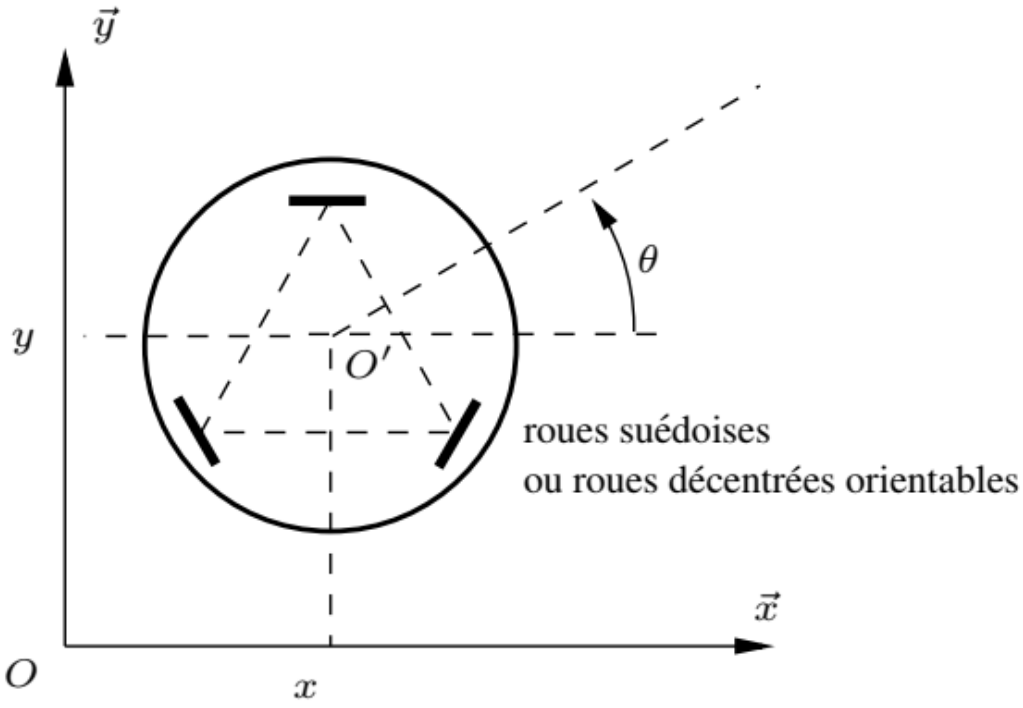


Figure II.13. Représentation d'un robot mobile omnidirectionnel.

II.4.4.2. Modélisation

Dans ce cas on peut considérer qu'il est possible de commander directement la posture et le modèle cinématique en posture est donc :

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2 \\ \dot{\theta} &= u_3 \end{aligned} \qquad \text{(II.19)}$$

Où $u = (u_1\ u_2\ u_3)^T$ représente le vecteur de commande. On choisit ainsi généralement ce type de robot pour se dispenser des problèmes de planification et de commande liés à la non-holonomie. L'avantage d'une cinématique extrêmement simple est cependant à mettre en balance avec les inconvénients liés à une localisation odométrique déficiente et à une plus grande complexité mécanique, généralement responsable d'un surcout. On citera pour mémoire le cas du XR4000 Nomadic représenté à la Figure II.14, disparu en 2000…

Figure II.14. Robot mobile omnidirectionnel Nomadic XR4000.

II.5. Propriétés du modèle cinématique d'un robot mobile

II.5.1. Représentation d'état

A partir de la modélisation que nous avons présentée tout au cours de ce chapitre il résulte, dans tous les cas, un modèle dynamique (au sens de l'Automatique) sous la forme (II.14). Si l'on considère que la configuration (réduite) q fait office de variable d'état du système on a :

$$\dot{x} = B(x)u \qquad \text{(II.20)}$$

Avec $B(x)$ de dimension $n \times m$. Il s'agit d'une représentation non linéaire sans terme de dérivé, en comparaison de la représentation d'état classique d'un système linéaire invariant $\dot{x} = Ax + Bu$.

C'est le cas des robots mobiles de type unicycle ou voiture, qui par ailleurs font partie d'une classe de système non linéaires particuliers dits *systèmes chaines*, comme on le précisera au paragraphe consacré à la commande des robots.

II.5.2. Commandabilité des robots mobiles

Pour qu'un robot mobile soit utile (ou utilisable) il faut en premier lieu s'assurer de sa *commandabilité*. Cette propriété signifie qu'il existe toujours une loi de commande $u(t)$ amenant le robot d'un état initial à un état final quelconque. Pour caractériser la commandabilité d'un système non linéaire, on peut tout d'abord se demander si son linéarise autour de tout point d'équilibre est commandable. En x_0 quelconque, le modèle linéarisé issu de (2.20) s'écrit :

$$\dot{x} = B(x_0)u$$

Avec $B(x_0) \in \mathbb{R}^{n \times m}$. Pour les robots de type unicycle et voiture, la non-holonomie va de pair avec une forme de sous-actionnement qui se traduit par le fait que $m < n$. Appliquée a un tel système, la condition de rang (critère de commandabilité de Kalman) :

$$Rang(B, AB, \ldots\ldots, A^{n-1}B) = n$$

Se traduit comme suit :

$$Rang(x_0) = n$$

Cette condition n'est jamais remplie puisque $m < n$. Le linéarisé du système n'est donc pas commandable autour d'un point d'équilibre quelconque. Il faut alors, pour statuer sur la commandabilité du système, utiliser le théorème de Chow. On donne ici (sans preuve) son interprétation pour un système dont le modèle cinématique est donné par (II.20).

Théorème II.1 : *Commandabilité d'un robot mobile non holonome*

On note $B(x) = (b_1(x)b_2(x) \ldots\ldots b_m(x))$ *la matrice du modèle cinématique (II.20), de dimension* $n \times m$*. Un robot mobile est commandable si les colonnes de* $B(x)$ *et leurs crochets de Lie successifs forment un ensemble de* n *colonnes indépendantes.*

On considère le cas de l'unicycle. A partir du modèle cinématique (II.14) et des crochets (II.5) et (II.6) on obtient 4 vecteurs indépendants pour un système de dimension 4. Le modèle cinématique de l'unicycle est donc commandable. On montrerait de même que celui de la voiture l'est aussi. La conséquence de cette propriété est l'existence de retours d'états pour commander le système, comme on le verra au chapitre 3.

II.6. Conclusion

Dans ce chapitre, nous avons présenté des notions très importantes relatives aux robots mobiles au sens général. Nous avons parlé sur l'état général lorsque nous avons le cas des roulements sans glissement ainsi la signification de ce que on appels contraints non holonômes.

En outre, trois grandes classes de robots mobiles et leurs modèles ont été présenté ; à savoir ; *le type unicycle, le type tricycle et le type omnidirectionnel. C*e chapitre est terminé par la présentation de quelques propriétés du modèle cinématique d'un robot mobile.

Pareillement, il est à noter que, dans ce mémoire le type de robot sur lequel articulé l'objectif envisagé et le type *unicycle.*

Chapitre III

Chapitre III : Problème de suivi de trajectoire

III.1. Introduction

Le problème de suivi d'une trajectoire de référence pour un robot mobile non-holonome est apparu comme un problème de premier ordre pour la communauté roboticienne dans ces dernières années. En effet, la forte utilisation des robots mobiles dans les domaines où l'être humain ne peut pas être présent, notamment dans les sites nucléaires à haut risque ou dans le cas de l'exploration spatiale, nécessite la mise en œuvre de lois de commande autonomes et performantes pour assurer les tâches assignées aux robots. Plusieurs travaux concernant la poursuite de trajectoire ont été développés dans ce contexte [1,2].

Dans ce chapitre, la présentation du problème de suivi de trajectoire sans exception (glissements, contraintes externes,….) sera divisée en deux parties principales. Plus précisément, nous parlerons dans un premier temps (dans la 1er partie) sur ce problème aux deux sens généraux *(i) contrôle d'orientation* et *(ii) stabilisation de mouvement* avec leurs inverses *(contrôle/sans contrôle et stabilisation/sans stabilisation).* Dans un second temps (dans la 2^{ième} partie), nous montrons en détaille quelques approches de commande (les plus connues) relatives à ce problème.

III.2. Partie I : Problème de suivi de trajectoire avec quelques conditions environnementales

III.2.1. Les différents problèmes

III.2.1.1. Définitions et hypothèses

On se réfère souvent au terme de commande pour décrire ce problème. Il s'agit bien évidemment d'un problème de commande dans la mesure où l'on cherche à déterminer les lois qui permettent au robot de suivre une trajectoire donnée. Ceci étant il est prudent d'être plus précis, bien que le vocabulaire en la matière soit relativement peu unifié.

Nous adopterons la terminologie suivante pour désigner les deux problèmes que l'on va évoquer.

On distinguera :

– le *suivi de chemin (*suivi d'une trajectoire*)*, problème qui consiste à trouver la commande pour asservir la distance d'un point du robot a une trajectoire de référence, la vitesse longitudinale de déplacement du robot étant donnée ;

– *la stabilisation de mouvement,* qui consiste à déterminer la commande du système permettant de stabiliser asymptotiquement à zéro l'erreur de suivi du robot par rapport à un robot fictif de référence.

Dans les deux cas, on recherche une commande par retour d'état, puisque la commande en boucle ouverte n'offrant que des possibilités limitées dans un environnement aussi incertain que celui envisage. Les solutions envisageables sont différentes selon que l'on souhaite ou non contrôler l'orientation du robot.

III.2.1.2. Paramétrage

Il est à noter que, le paramétrage le plus général des problèmes de commande traités ici est donné à la Figure III.1. Le point O', origine du repère R'est le milieu de l'axe des roues et P est un point de coordonnées $(a, b, 0)^T$ dans R'.

On considère un repère de référence $R_r = (O_r, \overrightarrow{x_r}, \overrightarrow{y_r}, \overrightarrow{z_r})^T$ tel que $\overrightarrow{z_r} = z_r$. On note aussi, θ_r l'angle $(\vec{x}, \overrightarrow{x_r})$ définissant l'orientation de R_r dans R et $\theta_e = \theta - \theta_r$.

Selon les cas R_r est sera un *repère de Frènet* sur la trajectoire (suivi de chemin) ou le repère associé à un robot de référence (stabilisation de mouvement).

III.2.2. Suivi de chemin (suivi d'une trajectoire)

III.2.2.1. Problématique

Ce problème consiste à asservir la distance d'un point P du robot a un chemin (trajectoire) de référence C que l'on souhaite suivre. Pour cela, on cherche la commande par retour d'état ω, la vitesse longitudinale v étant donnée (cas particulier).

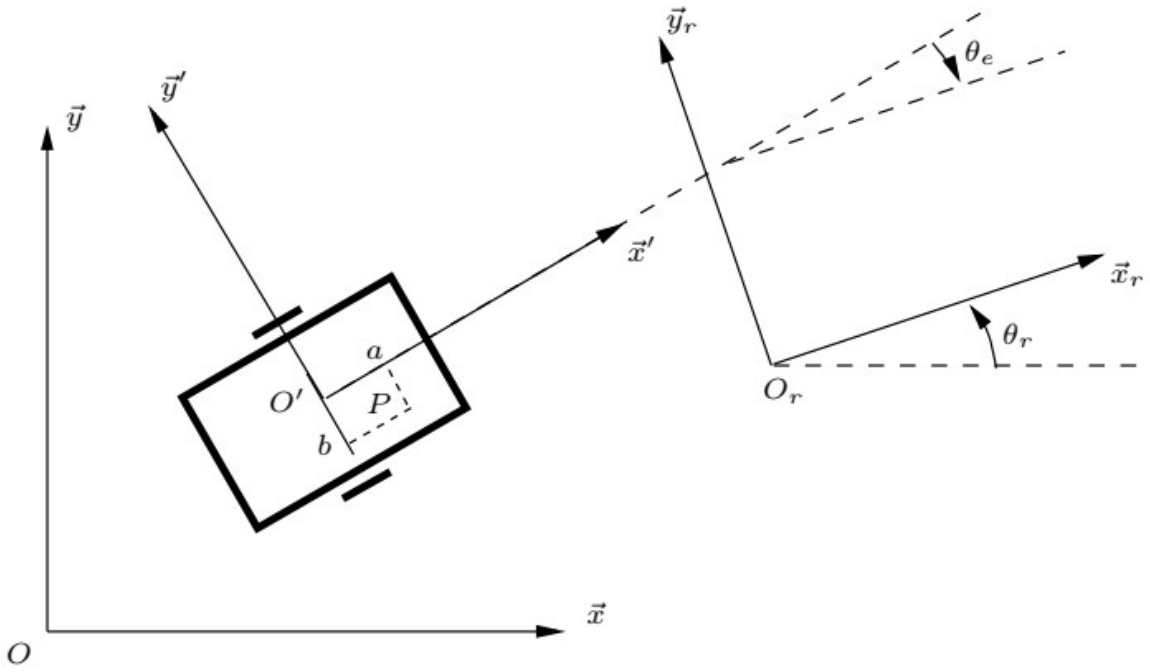


Figure III.1. Paramétrage pour les problèmes de commande.

III.2.2.2. Paramétrage

Le paramétrage du problème est donné à la Figure III.2. Le repère de référence est choisi de façon a ce que O_r soit le point de C le plus proche de P. Alors R_r est tel que $\overrightarrow{x_r}$ soit tangent à C en O_r et $\overrightarrow{y_r}$ normal à C en O_r. On note aussi, d la distance algébrique de O_r à P. Aussi, Le repère R_r mobile sur la trajectoire de référence C est un repère de Frènet, le point O_r étant repéré sur la trajectoire par son abscisse curviligne s, que l'on suppose normalisée.

Pour que O_r existe toujours et soit unique, il faut que P soit toujours à une distance de C inferieure au rayon de courbure minimum de C, ce qui s'écrit $|d| < |p(s)|, \forall\, s \in [0,1]$.

D'une manière équivalente, on a :

$$|dc(s)| < 1, \forall\, s \in [0,1], \qquad \text{(III.1)}$$

Où $c(s)$ est la courbure de C en O_r.

III.2.2.3. Modélisation dans le repère de Frènet

Pour le problème de suivi de chemin, la modélisation consiste à déterminer les mouvements de P et R' dans R_r et celui de R_r sur C, qui sont complètement caractérisés par la donnée de $\dot{s}$, $\dot{d}$ et $\dot{\theta}_e$.

La courbure $c(s)$ de C en O_r est définie par $c(s) = \frac{d\theta_e}{ds}$ donc :

$$\begin{aligned}\dot{\theta}_e &= \dot{\theta} - \dot{\theta}_r \\ &= \dot{\theta} - \frac{d\theta_r}{ds}\frac{ds}{dt}\end{aligned}$$

Soit :

$$\dot{\theta}_e = \omega - \dot{s}c(s) \quad , \tag{III.2}$$

La vitesse du point P vaut d'une part :

$$\overrightarrow{v_p} = \overrightarrow{v_{O'}} + \vec{\omega} \wedge O'P$$

$$= \overrightarrow{v_{x'}} + \omega\vec{z} \wedge (a\vec{x} + b\vec{y})$$

$$= (v - b\omega)(cos\theta_e\vec{x}_r + sin\theta_e\vec{y}_r) + a\omega(-sin\theta_e\vec{x}_r + cos\theta_e\vec{y}_r)$$

Soit :

$$\overrightarrow{v_p} = \big((v - b\omega)cos\theta_e - a\omega sin\theta_e\big)\vec{x}_r + \big((v - b\omega)sin\theta_e + a\omega cos\theta_e\big)\vec{y}_r \tag{III.3}$$

D'autre part, on peut exprimer la vitesse du point P par :

$$\overrightarrow{v_p} = \frac{d}{dt}\big(\overrightarrow{OP}\big)$$

$$= \frac{d}{dt}\big(\overrightarrow{OO_r} + \overrightarrow{O_rP}\big)$$

$$= \dot{s}\vec{x}_r + \frac{d}{dt}(d\overrightarrow{y_r})$$

$$= \dot{s}\vec{x}_r + \dot{d}\overrightarrow{y_r} - d\dot{\theta}_r\vec{x}_r$$

Soit :

$$\overrightarrow{v_p} = \dot{s}\big(1 - dc(s)\big)\vec{x}_r + \dot{d}\overrightarrow{y_r} \quad \text{(III.4)}$$

On déduit de (III.2), (III.3) et (III.4) que :

$$\dot{s} = \frac{(v - b\omega)cos\theta_e - a\omega sin\theta_e}{1 - dc(s)} \quad \text{(III.5)}$$

$$\dot{d} = (v - b\omega)sin\theta_e + a\omega cos\theta_e \quad \text{(III.6)}$$

$$\dot{\theta}_e = \omega - \dot{s}c(s) \quad \text{(III.7)}$$

III.2.2.4. Suivi de chemin sans contrôle d'orientation

On recherche une loi ω permettant de stabiliser la distance à la courbe de référence sans contrôle particulier de l'orientation du robot.

On choisit P situe à l'avant du robot sur l'axe longitudinal et donc $b = 0$. Alors (III.6) s'écrit :

$$\dot{d} = vsin\theta_e + a\omega cos\theta_e \quad \text{(III.8)}$$

La loi :

$$\omega = -\frac{vsin\theta_e}{acos\theta_e} - \frac{v}{cos\theta_e}k(d, \theta_e)d \quad \text{(III.9)}$$

Avec $k(d, \theta_e) \geq 0$ continu tel que $k\left(d, \mp\frac{\pi}{2}\right) = 0$ permet de linéariser (III.8) de sorte que la distance d du point P à la trajectoire vérifie :

$$\dot{d} = vak(d, \theta_e)d \quad \text{(III.10)}$$

Si a, v et $k(d,\theta_e)$ sont strictement positifs, $|d|$ est décroissante le long de toute trajectoire. Par définition de $k(d,\theta_e)$, ceci implique que l'erreur angulaire d'orientation doit rester strictement dans l'intervalle $\left]-\frac{\pi}{2},\frac{\pi}{2}\right[$.

On montre que si v est de signe constant et suffisamment régulière, d tend asymptotiquement vers 0. Il reste donc à vérifier que si θ_e est initialement dans l'intervalle $\left]-\frac{\pi}{2},\frac{\pi}{2}\right[$ il y reste. Dans le cas où l'erreur d'orientation initiale n'est pas dans l'intervalle en question le robot aura à faire une manœuvre lui permettant de s'y replacer. Avec la loi (III.9), (III.7) `s'écrit :

$$\dot{\theta}_e = v\left(-\frac{c(s)cos\theta_e}{1-dc(s)} - \left(1+\frac{ac(s)sin\theta_e}{1-dc(s)}\right)\left(\frac{tan\theta_e}{a} + + \frac{signe(v)k(d,\theta_e)d}{cos\theta_e}\right)\right)$$

Donc lorsque θ_e tend vers $-\frac{\pi}{2}$ par valeurs supérieures, $\dot{\theta}_e$ est du signe de :

$$\left(1+\frac{ac(s)}{1-dc(s)}\right)\frac{v}{a}$$

Et donc, $\dot{\theta}_e > 0$ si :

$$\frac{ac(s)}{1-d\ (s)} < 1$$

De la même manière on montre que lorsque θ_e tend vers $\frac{\pi}{2}$ par valeurs inferieures $\dot{\theta}_e < 0$ si :

$$\frac{ac(s)}{1-dc(s)} > -1$$

Si bien que finalement θ_e reste dans l'intervalle $\left]-\frac{\pi}{2},\frac{\pi}{2}\right[$ si :

$$\left|\frac{ac(s)}{1-dc(s)}\right| < 1$$

Le long de la trajectoire.

III.2.2.5. Suivi de chemin avec contrôle d'orientation

L'objectif cette fois est de déterminer une loi ω qui permette maintenant de stabiliser à la fois la distance a la trajectoire de référence et l'erreur en orientation durant le suivi.

On considère pour simplifier (et parce que cela suffit par la suite) que $a = b = 0$. On pose $x_1 = s$, et $u_1 = \dot{s}$, si bien que. $\dot{x}_1 = u_1$. Alors (III.5) s'écrit :

$$u_1 = \frac{v cos\theta_e}{1 - dc(s)}$$

Et donc (III.6) devient :

$$\dot{d} = v sin\theta_e$$

$$= u_1\big(1 - dc(s)\big)tan\theta_e$$

En posant $x_2 = d$, $x_3 = (1 - dc(s))tan\theta_e$ et $u_2 = \dot{x}_3$, il vient finalement la forme chaînée :

$$\dot{x}_1 = u_1$$
$$\dot{x}_2 = u_1 x_3$$
$$\dot{x}_3 = u_2$$

Note explicative

Soit un système représenté par son état $x = (x_1\, x_2\, x_3 \ldots\ldots x_n)^T$ et ayant pour commande $u = (u_1\, u_2)^T$. La représentation d'état du système est dite sous forme chaînée si elle s'écrit :

$$\dot{x}_1 = u_1$$
$$\dot{x}_2 = u_1 x_3$$
$$\ldots\ldots$$
$$\dot{x}_{n-1} = u_1 x_n$$
$$\dot{x}_n = u_2$$

L'intérêt de cette représentation est qu'il existe des techniques de commande non-linéaires adaptées à cette classe de système.

Le problème de commande initial qui consistait à rechercher un retour d'état ω à v donné est transformé par ce changement de variables. On recherche maintenant un retour d'état u_2, avec u_1 donné déduit de v. L'obtention de u_2 permet ensuite de recalculer ω.

Si l'on choisit le retour d'état proportionnel :

$$u_2 = -u_1 k_2 x_2 - |u_1| k_3 x_3$$

Avec k_2 et k_3 deux constantes strictement positives, alors :

$$\dot{x}_3 = -u_1 k_2 x_2 - |u_1| k_3 x_3$$

Et donc,

$$\ddot{x}_3 + |u_1| k_3 \dot{x}_3 + u_1^2 k_2 x_3 = 0$$

Ce qui signifie que pour u_1 constant quelconque x_3 et donc x_2 sont stables, le point de stabilité étant l'origine $x_2 = x_3 = 0$. Le système de d'épart est donc stable en $d = 0$ et $\theta_e = 0$, ce qui résout notre problème.

`On peut montrer selon [23] que le système est stable pour u_1 quelconque. En effet, pour des conditions initiales sur d et θ_e vérifiant :

$$x_2^2(0) + \frac{x_3^2}{k_2} < \frac{1}{c_{max}^2}$$

Où c_{max} est la courbure maximale le long de C, la condition sur la distance $|dc(s)| < 1$ est vérifiée et la fonction :

$$V(x) = \frac{1}{2}\left(x_2^2 + \frac{x_3^2}{k_2}\right)$$

tend vers zéro si u_1 est suffisamment régulière (bornée, dérivable et de dérivée bornée et ne tendant pas vers zéro lorsque t tend vers l'infini).

III.2.3. Stabilisation de mouvement

III.2.3.1. Problématique

Ce problème consiste à asservir la posture d'un robot mobile par rapport à un robot virtuel de référence.

III.2.3.2. Paramétrage

Le paramétrage du problème est donne à la Figure III.2.

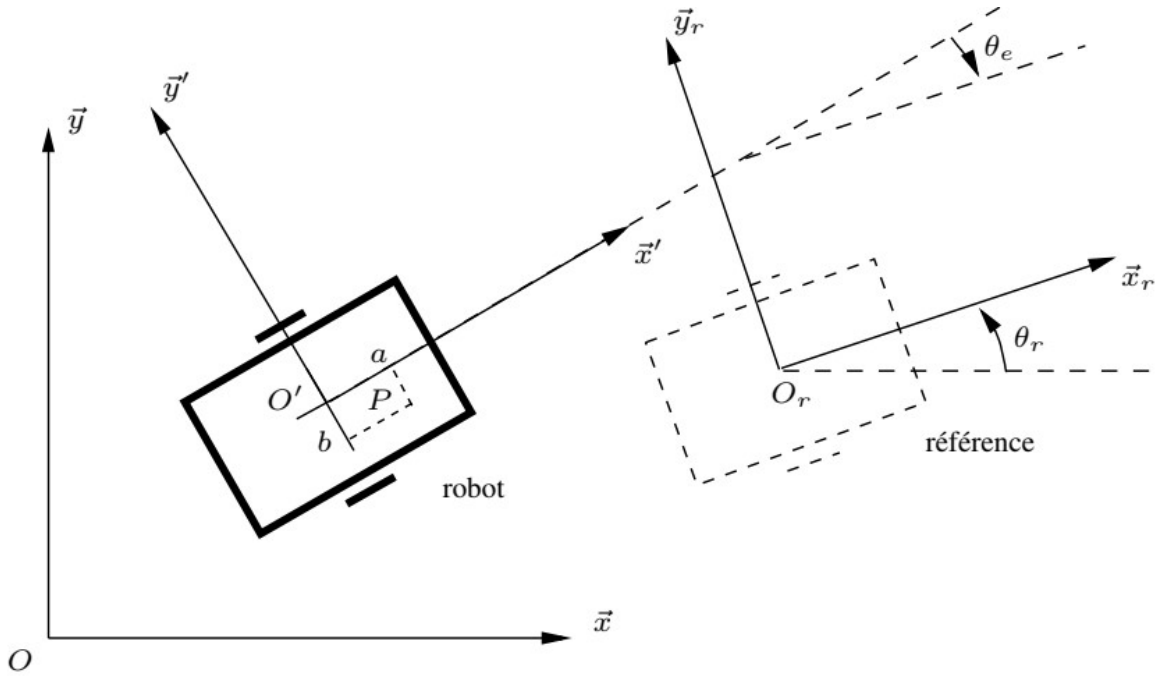


Figure III.2. Paramétrage pour la stabilisation de mouvement.

On considère dans la suite que le mouvement que d'écrit le robot de référence est admissible. Pour un robot dont le mouvement est décrit par la représentation d'état $\dot{\boldsymbol{x}} = B(\boldsymbol{x})\boldsymbol{u}$, un mouvement admissible est une loi $\boldsymbol{x_r}$ sur un intervalle de temps $[0, T]$ telle que $\dot{\boldsymbol{x_r}} = B(\boldsymbol{x_r})\boldsymbol{u_r}$.

Le problème de stabilisation consiste alors à trouver la commande par retour d'état $\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{x_r}, t)$ telle que l'origine du système d'erreur :

$$\dot{\boldsymbol{x_e}} = B(\boldsymbol{x})\boldsymbol{u} - B(\boldsymbol{x_r})\boldsymbol{u_r}$$

soit asymptotiquement stable.

III.2.3.3. Stabilisation sans contrôle d'orientation

L'objectif est de déterminer une loi (ω, v) *qui permette de stabiliser l'erreur entre la position courante et la position de référence sans contrainte particulière sur l'erreur en orientation.*

On choisit pour cela un point P situe sur l'axe longitudinal du robot, c'est-à-dire tel que $b = 0$. On peut calculer la dérivée de l'écart entre les positions de P et O_r dans R.

Soit $\boldsymbol{p_e}$ l'écart de position et $\boldsymbol{p_r}$ la position de O_r dans R :

$$\dot{\boldsymbol{p}}_{\boldsymbol{e}} = \overrightarrow{v_{P/R}} - \dot{\boldsymbol{p}}_{\boldsymbol{r}}$$

$$= \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} v - b\omega \\ a\omega \end{pmatrix} - \dot{\boldsymbol{p}}_{\boldsymbol{r}}$$

Par analogie avec (III.3). En posant le changement de variables :

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} 1 & -b \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

Inversible car par hypothèse $a \neq 0$, on obtient la representation :

$$\dot{\boldsymbol{p}}_{\boldsymbol{e}} = u - \dot{\boldsymbol{p}}_{\boldsymbol{r}}$$

Il est alors facile de stabiliser ce système par un simple retour d'état :

$$\boldsymbol{u} = \dot{\boldsymbol{p}}_{\boldsymbol{r}} - \boldsymbol{K}\dot{\boldsymbol{p}}_{\boldsymbol{e}}$$

Avec K définie positive.

Dans ce qui précède c'est le point P qui est asservi à la position de O_r et il persiste une erreur constante de $a \neq 0$. On va voir par la suite comment asservir l'origine du repère associée au robot à celle du robot virtuel de référence.

III.2.3.4. Stabilisation de mouvements admissibles avec contrôle d'orientation

L'objectif est de déterminer une loi (ω, v) *qui permette de stabiliser l'erreur entre la posture courante et la posture de référence.*

´On exprime l'erreur de posture ξ_e dans le repère R_r du robot de référence :

$$\xi_e = \begin{pmatrix} x_e \\ y_e \\ \theta_e \end{pmatrix} = \begin{pmatrix} cos\theta_r & sin\theta_r & 0 \\ -sin\theta_r & cos\theta_r & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x - x_r \\ x - y_r \\ \theta - \theta_r \end{pmatrix} \quad \text{(III.11)}$$

On montre en dérivant la relation (III.11) que :

$$\begin{pmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{pmatrix} = \begin{pmatrix} cos\theta_r \\ sin\theta_r \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega - \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} v_r - \begin{pmatrix} -y_e \\ x_e \\ 1 \end{pmatrix} \omega_r$$

Si l'on pose le changement de variables :

$$\begin{aligned} x_1 &= x_e \\ x_2 &= y_e \\ x_3 &= tan\theta_2 \\ u_1 &= vcos\theta_e - v_r \\ u_2 &= \frac{\omega - \omega_r}{cos^2\theta_e} \end{aligned}$$

Défini partout sauf en $\theta_e = \pm\frac{\pi}{2}$, la nouvelle représentation d'état du système s'écrit :

$$\begin{aligned} \dot{x}_1 &= u_1 + \omega_r x_2 \\ \dot{x}_2 &= u_1 x_3 - \omega_r x_1 + v_r x_3 \\ \dot{x}_3 &= u_2 \end{aligned}$$

On reconnait une forme chaînée si $v_r = \omega_r$. Alors, la loi de commande par retour d'état : ´

$$u_1 = -k_1|v_r|(x_1 + x_2x_3)$$

$$u_2 = -k_2v_rx_2 - k_3|v_r|x_3$$

Avec k_1, k_2 et k_3 trois constantes strictement positives rend le système asymptotiquement stable si v_r est suffisamment régulière (bornée, dérivable et de dérivée bornée et ne tendant pas vers zéro lorsque t tend vers l'infini) [26].

Une version linéarisée autour de zéro (le terme x_2x_3 disparait alors) de cette commande stabilise également l'origine de l'état.

III.3. Partie II : Quelques approches de commandes relatives au problème de suivi de trajectoire

III.3.1. Introduction

Lorsqu'on envisage de mettre en œuvre une commande de suivi de trajectoire, la première l'idée serait probablement d'imaginer la trajectoire de référence comme une position de référence en mouvement. Après, à chaque période d'échantillonnage dans le processus de commande, le point de référence se déplace vers le point actuel de la trajectoire de référence $(x_{ref}(t), y_{ref}(t))$, et l'action de commande de la position de référence est appliqué via deux les lois de commande indiquées dans ([2] page 71, (3.11) et (3.12)). Il est à noter ici que, le robot doit être déplacé à côté (le plus possible) de ce point de référence imaginaire. De plus, lorsque la vitesse est faible et les mesures de position sont bruités, la mesure de position du robot peut être trouvé décalée de la trajectoire. Il est donc extrêmement important de gérer correctement ces situations, p.ex. en utilisant une loi de commande soumis à des mis à jour ([2] page 71, (3.13)). En effet, cette approche souffre du fait qu'elle nécessite des gains de commande relativement importants, afin de réduire les erreurs mesurées, chose qui rend cette dernière fortement sensible aux perturbations. Il est donc très indispensable d'utiliser ce qu'en appelle en anglais le *'Feedforward compensation*' qui sera introduite dans ce qui suit.

III.3.2. Décomposition de la commande en action de Feedforward et de rétroaction (Feedback Action)

En réalité, la commande de suivi de trajectoire est importante non seulement d'un point de vue pratique, mais aussi d'un point de vue théorique. Son importance découle du fameux résultat basé sur la condition dite de *Brockett's*, d'où les systèmes non-holonomes (*Non-holonomic Systems*) ne peuvent pas être stabilisés asymptotiquement autour du point d'équilibre en utilisant une rétroaction régulière (ou lisse) invariante dans le temps (*smooth time-invariant feedback*) [24]. Ce résultat peut être vérifié facilement pour le cas d'une conduite différentielle, il peut être étendu à d'autres cinématiques y compris la cinématique Ackermann. Tout d'abord, nous vérifions si le système est (*driftless*) [2]. Puisque la dérivée du champ vectoriel $\dot{q}(t)$ est nulle en cas d'absence de commande $(v = 0, \omega = 0)$, le système est en effet (*driftless*). *Brockett* [24] a montré qu'une des conditions pour qu'un système a la caractéristique précédente (*a driftless system*), soit stabilisé par un retour continu invariant

dans le temps (*continuous time-invariant feedback*) est que le nombre d'entrées doit être au moins égal au nombre d'états. Dans le cas d'une conduite différentielle, cette condition est clairement violée et d'autres types de rétroaction doivent être recherchés. Il est à noter aussi que, des systèmes totalement non-holonomes est (*driftless*), sont toujours contrôlables dans un sens non-linéaire. Par conséquent, une stabilisation asymptotique peut être obtenue en utilisant une loi de commande variante dans le temps, discrète ou hybride.

L'une des possibilités de contourner la limitation imposée par La condition de *Brockett* est d'introduire une structure de commande complètement différente. Dans le cas d'une commande de suivi de trajectoire, très souvent à deux degrés de liberté cette dernière est utilisée lors l'un des deux chemins est correspond au chemin d'anticipation (*feedforward path*) et l'autre correspond au chemin de retour (*feedback path*).

Avant d'aborder l'étude (présentation) de ces deux types de commandes *feedforward* et *feedback*, il est très indispensable d'introduire une autre propriété de système très importante.

Un système est appelé différentiellement plat (*differentially flat*), s'il existe un ensemble de sorties plates (flat outputs), et tous les états et les entrées du système peuvent être réécrits en tant que fonctions de ces sorties plates et d'un nombre fini de leurs dérivées temporelles [2].

Cela signifie que certaines fonctions non linéaires f_x et f_u doivent exister satisfaisant les relations suivantes :

$$x = f_x\left(z_f, \dot{z}_f, \ddot{z}_f, \ldots\ldots, \frac{d^p}{dt^p} z_f\right)$$

$$u = f_u\left(z_f, \dot{z}_f, \ddot{z}_f, \ldots\ldots, \frac{d^p}{dt^p} z_f\right)$$

Où les vecteurs x, u et z_f représentent respectivement les états du système, ses entrées et ses sorties plates, tandis que p est un entier fini. Nous devrions également mentionner que les sorties plates devraient être des fonctions des états du système, de ses entrées et d'un nombre fini de dérivées d'entrée. Cela signifie que pour des sorties plates (cas général) sont fictives ; c.à.d. ils ne correspondent pas aux sorties réelles.

Dans le cas d'un robot mobile à roues a le modèle (modèle cinématique différentiel) exprimé dans le section II.4.2.4, les sorties plates sont les sorties réelles du système x et y. En effet, on

peut facilement montrer que toutes les entrées (les deux vitesses) et la troisième état (orientation du robot) peuvent être représentés comme des fonctions de x et y, ainsi leurs dérivées.

Nous savons que $\dot{x}$ et $\dot{y}$ peuvent être interprétés comme des coordonnées cartésiennes de la vitesse de translation du robot. Par conséquent, la vitesse de translation est obtenue par la somme dite *Pythagorean sum* de sa composante :

$$v = \sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \tag{III.12}$$

En raison de contraintes non-holonomes (*nonholonomic constraints*), un robot à roues à entraînement différentiel (*Differentially Driven Wheeled Robot*) se déplace toujours dans le sens de son orientation, ce qui signifie que la tangente de l'orientation est égale au quotient des composantes cartésiennes de la vitesse de translation :

$$\varphi(t) = arctan\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right) \tag{III.13}$$

Finalement, la vitesse angulaire $\omega(t)$ est déterminée comme la dérivée temporelle de l'orientation $\varphi(t)$ donnée par l'équation (III.13), elle n'est pas définie uniquement dans le cas où la vitesse de translation devient 0.

$$\omega(t) = \frac{d}{dt}\left[arctan\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right)\right] = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}^2(t) + \dot{y}^2(t)} \tag{III.14}$$

III.3.3. Technique de linéarisation de retour d'état (State Feedback Linearization)

La linéarisation de retour d'état (*State Feedback Linearization*) est une technique de commande des systèmes non linéaires très puissante. Ainsi, l'idée principale de cette dernière est de faire transformer algébriquement la dynamique d'un système non linéaire donné en un système entièrement ou partiellement linéarisé afin que l'ensemble de méthode de commande par retour d'état puissent être appliqué [2]. Il est à noter que, cette technique, qui est une transformation exacte d'un état (*State*) et un retour (*Feedback*), est entièrement différente de la linéarisation conventionnelle basée sur des approximations de la série de Taylor.

Selon [2], les étapes qui doivent être suivies lors de la conception d'une linéarisation de retour d'état pour intérêt de commande sont les suivantes :

- Les sorties appropriées (dites ; flat outputs) doivent être choisies. Leur nombre est le même que le celui des d'entrées de système,
- Les sorties (dites ; flat outputs) doivent être dérivées ainsi leurs dérivés doivent vérifiées la présence fonctionnelle des entrées. Cette étape est répétée jusqu'à toutes les entrées (ou leurs dérivées) apparaissent dans les dérivées des sorties (dites ; flat outputs). De plus, si toutes les entrées (plus précisément leurs dérivées les plus élevées) peuvent être dérivées à partir de ce système d'équations, nous pouvons passer à l'étape suivante.
- Le système d'équations est résolu pour les dérivées les plus élevées des entrées individuelles. Pour obtenir les entrées réelles du système, une chaîne d'intégrateurs (*integrators*) doit être utilisé sur les dérivés d'entrée. D'autre part, les dérivés de sortie, servir comme de nouvelles entrées d'système.

On considère le cas d'un robot mobile à roues a le modèle exprimé dans le section II.4.2.4, les sorties (dites ; flat outputs) sont respectivement $x(t)$ et $y(t)$. Ainsi, les dérivées premières de ces dernières sont données comme suit :

$$\dot{x} = v cos\theta$$
$$\dot{y} = v sin\theta$$

Dans les premières dérivées, seule la vitesse de translation v est apparaît ainsi la différenciation peut être contenue :

$$\ddot{x} = \dot{v} cos\theta - v\dot{\theta} sin\theta$$
$$\ddot{y} = \dot{v} sin\theta + v\dot{\theta} sin\theta$$

Il est clair que, dans les expressions des dérivées secondes de x et y, les deux vitesses (v et $\omega = \dot{\theta}$) sont apparues. Maintenant, le système d'équations est réécrit pour que les dérivées secondes des sorties sont décrites comme des fonctions des dérivées les plus élevées des entrées individuelles ($\dot{v}$ et ω dans ce cas) :

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} = \begin{bmatrix}cos\theta & -vsin\theta\\ sin\theta & vcos\theta\end{bmatrix}\begin{bmatrix}\dot{v}\\ \omega\end{bmatrix} = F\begin{bmatrix}\dot{v}\\ \omega\end{bmatrix} \quad \text{(III.15)}$$

La matrice F dans (III.15), qui est non singulière si $v = 0$. Ainsi, le système d'équations précédent peut donc être résolu pour en cherchant $\dot{v}$ et ω:

On obtient ainsi :

$$\begin{bmatrix}\dot{v}\\ \omega\end{bmatrix} = F^{-1}\begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} = \begin{bmatrix}cos\theta & sin\theta\\ -\frac{sin\theta}{v} & \frac{cos\theta}{v}\end{bmatrix}\begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} \quad \text{(III.16)}$$

En effet, la grandeur ω est la solution de l'équation (III.16) qui est en réalité l'entrée réelle du robot, tandis que la solution $\dot{v}$ doit être intégrée avant d'être pouvoir utilisée comme entrée.

De plus, le système linéaire obtenu a les entrées suivantes $[u_1, u_2] = [\ddot{x}, \ddot{y}]^T$, et comme états $z = [x, y, \dot{x}, \dot{y}]^T$ (notant que le modèle cinématique exprimé dans le section II.4.2.4 a trois états ; le quatrième est dû à une intégration supplémentaire).

De ce fait, La dynamique du nouveau système peut être décrit aisément par la représentation dans l'espace d'état suivante :

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{y}\\ \ddot{y}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ y\\ \dot{y}\end{bmatrix} + \begin{bmatrix}0 & 0\\ 1 & 0\\ 0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \quad \text{(III.17)}$$

Ou sous la forme compacte suivante :

$$\dot{z} = Az + Bu \quad \text{(III.18)}$$

Le système (III.18) est contrôlable puisque sa matrice de contrôlabilité donnée dans (III.19) a un rang plain, chose qui implique que le contrôleur d'état existe pour un polynôme caractéristique choisi arbitrairement dans la boucle fermée.

$$Q_C = [B, AB] \quad \text{(III.19)}$$

Aussi, une exigence supplémentaire est de concevoir une loi de commande pour que le robot suive une trajectoire de référence. Dans le cadre des systèmes plats (flat systems), une trajectoire de référence est donnée pour les sorties (dites ; flat outputs), dans ce cas, ils sont respectivement $x_{\mathrm{ref}}(t)$ et $y_{\mathrm{ref}}(t)$.

La référence peut alors être facilement obtenue pour l'état du système $z_{\mathrm{ref}}(t) = [x_{\mathrm{ref}}, \dot{x}_{\mathrm{ref}}, y_{\mathrm{ref}}, \dot{y}_{\mathrm{ref}}]^T$ ainsi l'entrée du système $u_{\mathrm{ref}}(t) = [\ddot{x}_{\mathrm{ref}}, \ddot{y}_{\mathrm{ref}}]$.

L'équation (III.18) peut également être écrite pour les signaux de référence :

$$\dot{z}_{\mathrm{ref}} = Az_{\mathrm{ref}} + Bu_{\mathrm{ref}} \tag{III.20}$$

L'erreur entre les états réels et les états de référence est définie comme par $\tilde{z} = z - z_{\mathrm{ref}}$. Soustraire l'équation (III.20) de (III.18), nous donne :

$$\dot{\tilde{z}} = A\tilde{z} - B(u - u_{\mathrm{ref}}) \tag{III.21}$$

Il est à noter que, cette dernière équation (III.21) décrit la dynamique de l'erreur d'état. Cette dynamique doit être stable et suffisamment rapide. Une façon de prescrire la dynamique en boucle fermée est d'imposer des pôles spécifiques en boucle fermée.

Nous avons déjà montré que la paire (A, B) est contrôlable, chose qui nous permettre de faire la sélection correcte d'une matrice de gain de contrôle K (de dimension 2×4), des emplacements arbitraires des pôles en boucle fermée dans le demi-plan gauche du plan complexe peut être réalisé.

Par conséquent, (III.21) devient :

$$\dot{\tilde{z}} = (A - BK)\tilde{z} - BK\tilde{z} + B(u - u_{\mathrm{ref}})$$

$$\dot{\tilde{z}} = (A - BK)\tilde{z} + B(K\tilde{z} + u - u_{\mathrm{ref}}) \tag{III.22}$$

Si le dernier terme de l'équation (III.21) est nul, les erreurs d'état convergent vers 0 avec la dynamique prescrite, donnée par la matrice du système en boucle fermée $(A - BK)$.

En effet, forcer ce terme à 0 définit la loi de commande relative à cette approche :

$$u(t) = K\big(z_{\text{ref}}(t) - z(t)\big) + u_{\text{ref}}(t) \qquad \text{(III.22)}$$

Une représentation sous forme d'un diagramme fonctionnel du système de commande conçu est illustrée sur la Figure III.3.

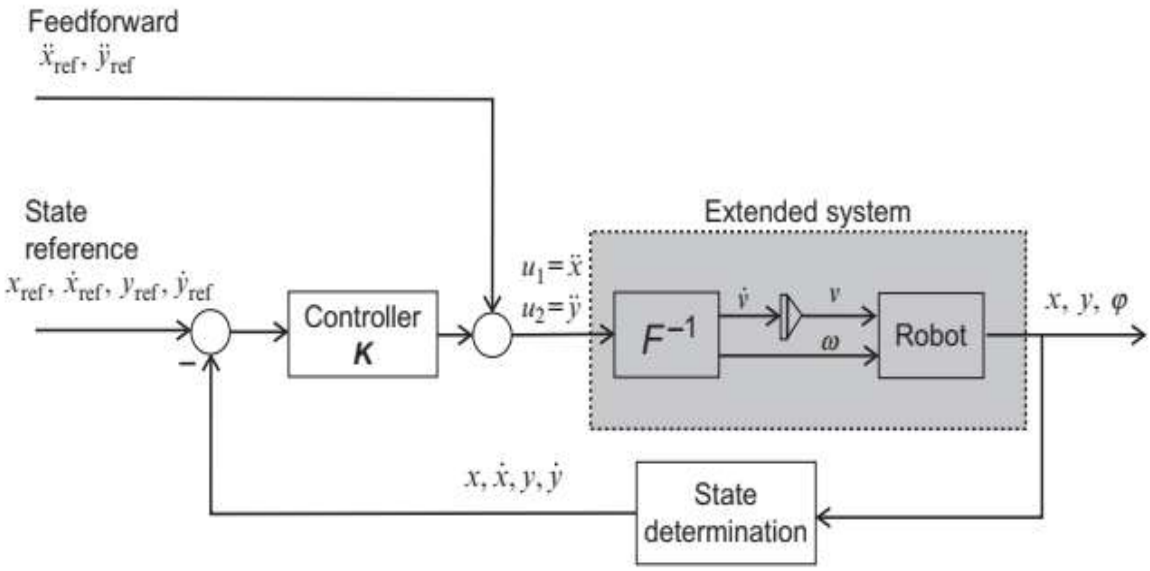


Figure III.3. Commande pour suivi d'une trajectoire à base d'une linéarisation de retour d'état [2].

En raison d'une forme spécifique des deux matrices A et B dans (III.17), d'où u_1 a uniquement une influence sur les deux états z_1 et z_2 tandis que u_1 n'influence seulement que sur les deux états z_3 et z_4, de ce fait la matrice de gain de contrôle prend la forme spéciale suivante :

$$K = \begin{bmatrix} k_1 & k_2 & 0 & 0 \\ 0 & 0 & k_3 & k_4 \end{bmatrix} \qquad \text{(III.23)}$$

La loi de commande (III.21) peut donc être complètement décomposée comme suit :

$$u_1(t) = \ddot{x}(t) = k_1\big(x_{\text{ref}}(t) - x(t)\big) + k_2\big(\dot{x}_{\text{ref}}(t) - \dot{x}(t)\big) + \ddot{x}_{\text{ref}}(t) \qquad \text{(III.24)}$$

$$u_2(t) = \ddot{y}(t) = k_3\big(y_{\text{ref}}(t) - y(t)\big) + k_4\big(\dot{y}_{\text{ref}}(t) - \dot{y}(t)\big) + \ddot{y}_{\text{ref}}(t)$$

Il est à noter que, l'approche de commande proposée nécessite la connaissance de tous les états. En, effet, malgré que x et y sont généralement mesurés, leurs dérivés ne le sont pas.

De plus, La dérivée numérique conduit pratiquement à une amplification de bruit chose qui doit être évitée. A cet effet, Deux solutions possibles sont proposées :

- Les états non mesurés peuvent être estimés par des observateurs d'État.
- Si l'orientation du robot θ est mesurée, les dérivées peuvent être calculées comme à travers les deux équations $\dot{x} = vcos\theta$ et $\dot{y} = vsin\theta$.

L'exemple de simulation illustratif donné dans ce qui suit, nous permettre de prendre une idée très profonde sur cette approche.

Exemple de simulation illustratif - 1 [2] :

Un robot mobil à conduite différentielle doit être commandé d'une façon qui permet de suivre la trajectoire de référence définie par $x_{\text{ref}} = 1{,}1 + 0{,}7sin(\frac{2\pi t}{30})$ et $y_{\text{ref}} = 0{,}9 + 0{,}7sin(\frac{4\pi t}{30})$. Le pas de simulation est égal à $T_s = 0{,}033\ s$. La position initiale est $[x(0), y(0), \theta(0)] = [1{,}1, 0{,}8, 0]$.

Le détail de cet exemple est illustré dans script MATLAB (voir le code MATLAB M-1 dans l'annexe).

On Implémente l'algorithme de commande présenté en utilisant le Matlab, ainsi les tous résultats obtenus sont illustrés dans les figures Figure III.4, Figure III.5 et Figure III.6.

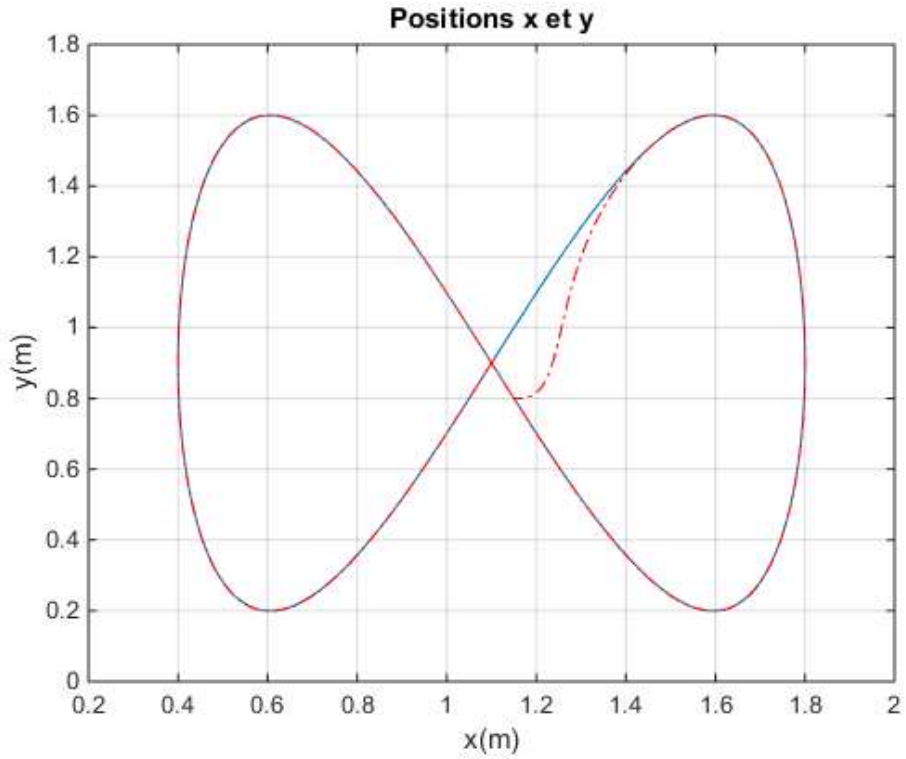


Figure III.4. Potions $x(t)$ et $y(t)$ ($x_{\text{ref}}(t)$ et $y_{\text{ref}}(t)$ trajectoire de référence en bleu, $x_{\text{réelle}}(t)$ et $y_{\text{réelle}}(t)$ en rouge discontinu).

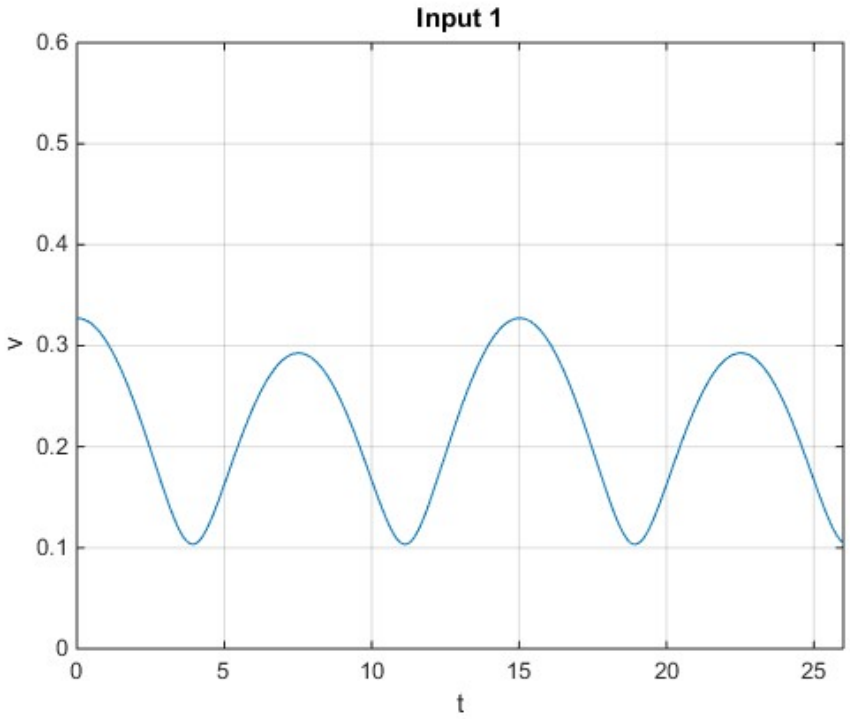


Figure III.5. Sortie 1 du régulateur (v).

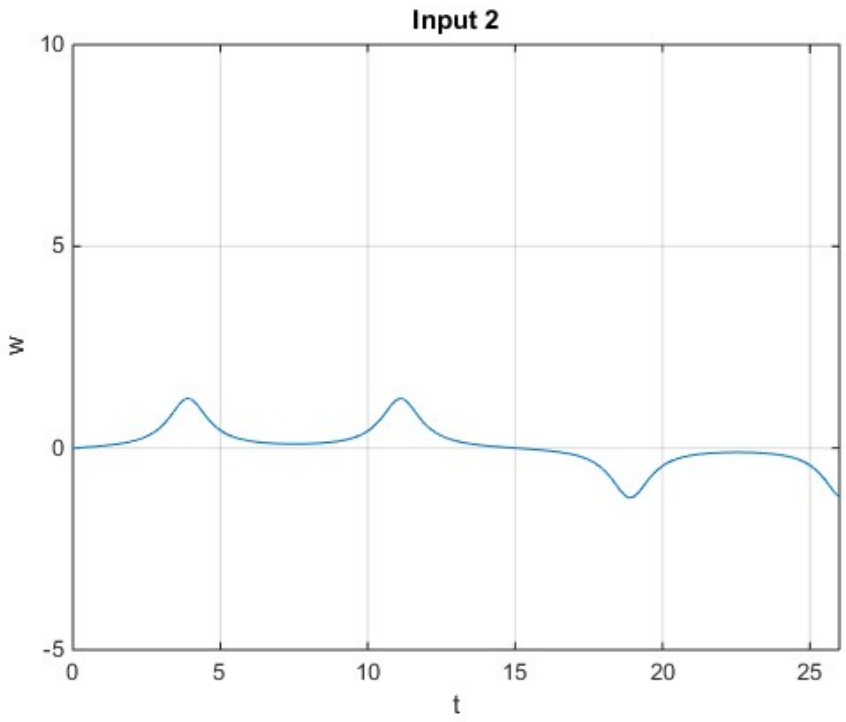


Figure III.6. Sortie 2 du régulateur (ω).

III.3.4. Développement d'un modèle cinématique d'erreur dédié au problème de suivi de trajectoire (Kinematic Trajectory-Tracking Error Model) [2]

Généralement, afin de résoudre le problème de commande (le cas traité- chez les robots mobiles étudiés dans ce mémoire), la transformation des coordonnées du robot est souvent effectuée sous une forme très adaptée aux objectives envisagées (de commande).

La posture de l'erreur n'est pas donnée dans les coordonnées globales du système, mais plutôt comme une erreur dans le système de coordonnées local du robot qui est aligné avec le mécanisme de mouvement. Cette erreur est exprimée comme l'écart entre un robot de référence dit virtuel et celui du robot réel comme il est illustré sur la Figure III.7.

Les erreurs obtenues sont les suivantes : e_x qui donne l'erreur dans la direction de conduite, e_y qui donne l'erreur dans la direction perpendiculaire, et e_φ qui donne l'erreur d'orientation. Ces erreurs sont bien illustrées sur la Figure III.7. Il est à noter que, cette approche a été adoptée la première fois dans [25].

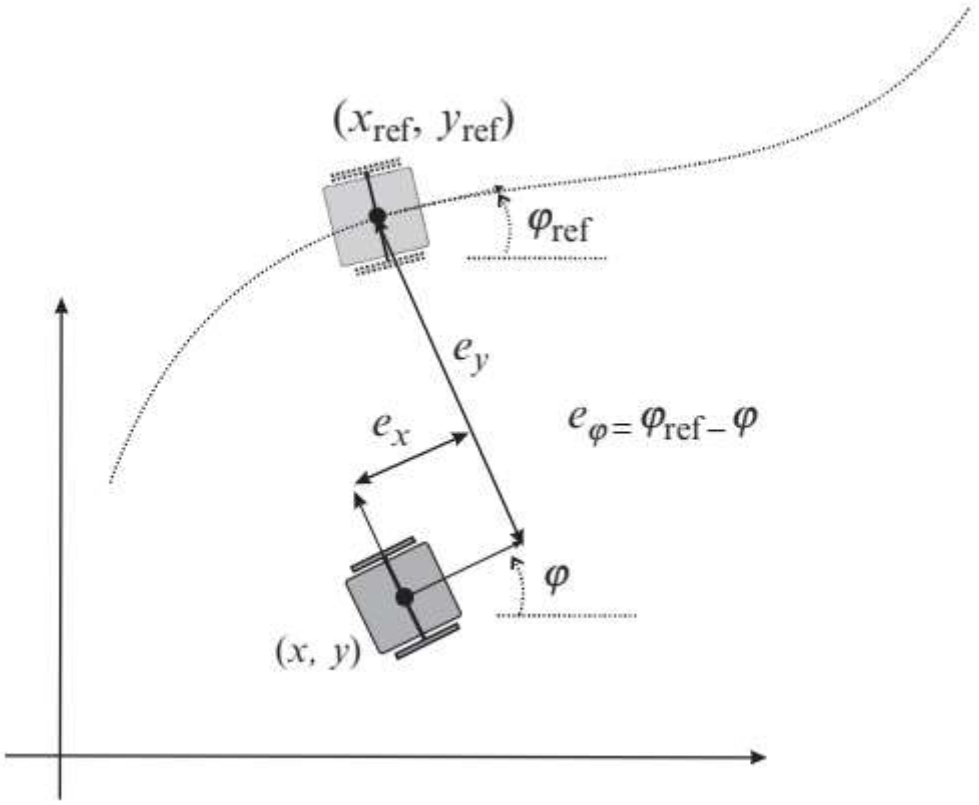


Figure III.7. Illustration de l'erreur de posture en coordonnées locales [2].

En effet, l'erreur de posture $e(t) = \left[e_x(t), e_y(t), e_\varphi(t) \right]^T$ est déterminée en utilisant la posture réelle $q(t) = [\, x(t), y(t), \varphi(t)]^T$ du robot réel et la posture de référence $q_{ref}(t) = \left[x_{ref}(t), y_{ref}(t), \varphi_{ref}(t) \right]^T$ du robot de référence virtuel :

$$\begin{bmatrix} e_x(t) \\ e_y(t) \\ e_\varphi(t) \end{bmatrix} = \begin{bmatrix} cos(\varphi(t)) & sin(\varphi(t)) & 0 \\ -sin(\varphi(t)) & cos(\varphi(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(q_{ref}(t) - q(t) \right) \qquad \text{(III.25)}$$

En supposant que le robot réel et le robot de référence ont le même modèle cinématique donné dans la section II.4.2.4 et en tenant compte de la transformation (III.25), le modèle d'erreur de posture peut s'écrire comme suit :

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\varphi \end{bmatrix} = \begin{bmatrix} cos(e_\varphi) & 0 \\ sin(e_\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} -1 & e_y \\ 0 & -e_x \\ 0 & -1 \end{bmatrix} \qquad \text{(III.26)}$$

Avec v_{ref} et ω_{ref} sont les vitesses de référence linéaire et angulaire données par les deux équations suivantes (voir [2], section 3.3.2) :

$$v_{ref} = \sqrt{\dot{x}_{ref}^2(t) + \dot{x}_{ref}^2(t)} \tag{III.27}$$

$$\omega_{ref} = \frac{\dot{x}_{ref}(t)\ddot{y}_{ref}(t) - \dot{y}_{ref}(t)\ddot{x}_{ref}(t)}{{v_{ref}}^2} \tag{III.28}$$

L'entrée $u = [v, \omega]^T$ doit être fournie par le régulateur. Très souvent (voir [26]) la commande u est décomposée comme suit :

$$u = [v, \omega]^T = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_{ref} cos(e_\varphi) + v_{fb} \\ \omega_{ref} + \omega_{fb} \end{bmatrix} \tag{III.29}$$

D'où v_{fb} et ω_{fb} sont respectivement les signaux de retour (*feedback signals*) à définir ultérieurement, tandis que $v_{ref} cos(e_\varphi)$ et ω_{ref} sont les signaux d'anticipation (*feedforward signals*), bien que techniquement parlant, $v_{ref} cos(e_\varphi)$ est modulé avec l'erreur d'orientation qui provient de la sortie. Par contre, lorsque l'erreur d'orientation est amenée à 0, $v_{ref} cos(e_\varphi)$ devient une « vraie » anticipation.

L'application de la commande (III.29) dans l'équation (III.26) aboutit au modèle d'erreur de suivi :

$$\begin{aligned} \dot{e}_x &= \omega_{ref} e_y - v_{fb} + e_y \omega_{fb} \\ \dot{e}_y &= -\omega_{ref} e_x + v_{ref} sin(e_\varphi) - e_x \omega_{fb} \\ \dot{e}_\varphi &= -\omega_{fb} \end{aligned} \tag{III.30}$$

En effet, le but de la commande ici est de faire conduire les erreurs du modèle d'erreur (III.30) vers 0 en choisissant correctement les commandes v_{fb} et ω_{fb}. Cette dernière constatation sera détaillée dans les sections qui viennent.

III.3.5. Régulateur linéaire

En réalité, le modèle d'erreur (III.30) obtenu est non linéaire. Dans ce qui suit, ce modèle sera linéarisé pour admettre l'utilisation d'un régulateur linéaire. La linéarisation doit avoir lieu autour d'un certain point d'équilibre. Ici, le choix évident est une erreur égale à zéro c.à.d. $(e_x = 0,\ e_y = 0, e_\varphi = 0)$. Ce dernier point est considéré comme point d'équilibre du modèle (III.30) si les deux vitesses de rétroaction sont également égales à 0 (v_{fb}=0, $\omega_{fb} = 0$).

Ainsi, la linéarisation de (III.30) autour d'une erreur égale à zéro nous donne ce qui suit :

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\varphi \end{bmatrix} = \begin{bmatrix} 0 & \omega_{ref} & 0 \\ -\omega_{ref} & 0 & v_{ref} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_{fb} \\ \omega_{fb} \end{bmatrix} \quad \text{(III.31)}$$

Le système ci-dessus est en réalité de nature linéaire et variable dans le temps puisque $v_{ref}(t)$ et $\omega_{ref}(t)$ sont des fondations temporelles. De plus, ce dernier représente un modèle d'état d'un système d'erreur de type dynamique où tous les états (erreurs dans ce cas) sont accessibles. Un retour d'état complet est donc possible si le système est contrôlable.

Si on suppose que v_{ref} et ω_{ref} sont constants (le chemin de référence est constitué de segments de droite et d'arcs de cercle), il est facile de confirmer que la matrice de contrôlabilité définie par (III.19) est du rang complet, et par conséquent toutes les erreurs peuvent être forcées à 0 par un retour d'état statique. Aussi, si v_{ref} et ω_{ref} ne sont pas constants, la contrôlabilité est conservée si l'un des signaux de référence est différent de 0, bien que l'analyse devienne plus compliquée.

Puisque le système (III.31) a une structure spéciale, un retour d'état statique avec une forme simple de matrice de gain est souvent utilisé :

$$\begin{bmatrix} v_{fb} \\ \omega_{fb} \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & k_\varphi \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} \quad \text{(III.32)}$$

Nous observons clairement de (III.32), que l'erreur dans le sens de la marche est corrigée par v_{fb}, tandis que les erreurs dans l'orientation ainsi dans les directions latérales sont corrigées par ω_{fb}. Il est à noter que, les gains (k_x, k_y, k_φ) du régulateur peuvent être déterminés soit par tâtonnement, ou par leur optimisation dans un modèle de système ou à

travers l'emploi de l'approche du placement des pôles, ou autres,…… Ici, les gains de régulateur sont choisis de telle sorte que les pôles du système de commande se trouvent dans des positions appropriés dans le plan complexe p. Le système a trois pôles dont l'un doit être réel tandis que les deux autres seront des complexes conjugués. Premièrement, les pôles seront placés sur des emplacements fixes $-2\zeta\omega_n$, et $-\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}$ où la fréquence propre non amortie $\omega_n > 0$ et le coefficient d'amortissement $0 < \zeta < 1$ sont des paramètres de conception.

Si le polynôme caractéristique du système en boucle fermée (donnée comme suit) :

$$\left| p\boldsymbol{I}_{\mathbf{3\times 3}} - \begin{bmatrix} 0 & \omega_{ref} & 0 \\ -\omega_{ref} & 0 & v_{ref} \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & k_\varphi \end{bmatrix} \right|$$

Il est comparé à celui souhaité :

$$(p + 2\zeta\omega_n)(p^2 + 2\zeta\omega_n p + \omega_n^2)$$

La valeurs gains du régulateur peut être obtenue [27] :

$$k_x = k_\varphi = 2\zeta\omega_n$$

$$k_y(t) = \frac{\omega_n^2 - \omega_{ref}^2(t)}{v_{ref}(t)} \qquad \text{(III.33)}$$

Il est à noter que, ω_n doit être plus grand que le maximum de $|\omega_{ref}|$. Les gains du régulateur exprimés par l'équation (III.33), ne sont pas applicables dans le cas pratique puisque le gain $k_y(t)$ devient extrêmement grande lorsque la vitesse de référence $v_{ref}(t)$ est faible. Pour surmonter ce problème, la fréquence propre non amortie ω_n deviendra variable dans le temps.

Puisqu'il est naturel d'adapter les temps de stabilisation de la réponse transitoire aux vitesses de référence, le choix suivant semble très judicieux :

$$\omega_n(t) = \sqrt{\omega_{ref}^2(t) + g v_{ref}^2(t)} \quad \text{avec} \quad g > 0$$

En répétant une procédure similaire à celle illustrée ci-dessus, les gains du régulateur suivants sont obtenus :

$$k_x = k_\varphi = 2\zeta\sqrt{\omega_{ref}^2(t) + gv_{ref}^2(t)}$$

$$k_y(t) = gv_{ref}(t) \tag{III.34}$$

Deux remarques importantes relatives aux méthodes de commande présentés jusqu'à présent, doivent être indiquées :

- Les lois de commande présentées, sont conçues en basant sur des modèles linéarisés. Il est à noter que, un modèle linéarisé n'est valable qu'au voisinage du point de fonctionnement (erreur nulle dans ce cas), et les performances de commande peuvent ne pas être celles attendues en cas d'erreurs importantes de commande,
- Pareillement, si nous avons travaillé avec à un système linéaire mais variable dans le temps, certains des résultats connus des systèmes invariants linéaires dans le temps ne sont plus valides. Nous devons mentionner ici que même si tous les pôles se trouvent dans des emplacements (fixes) du demi-plan gauche du plan complexes, le système pourrait être instable. Malgré les difficultés potentielles mentionnées ci-dessus, les lois de commande linéaires sont souvent utilisées dans la pratique en raison de leur simplicité, de leur réglage relativement facile et de leurs performances et robustesse acceptables.

Afin de monter, l'efficacité de cette méthode de commande détaillée ci-dessus, nous montrons dans ce qui suit, les résultats de simulations obtenus sous les conditions illustrées dans l'exemple de simulation illustratif 1 (voir page 65).

Le détail de cet exemple est illustré dans le script MATLAB (voir le code MATLAB M-2 dans l'annexe).

On Implémente l'algorithme de commande présenté en utilisant le Matlab, ainsi les tous résultats obtenus sont illustrés dans les figures Figure III.8, Figure III.9 et Figure III.10.

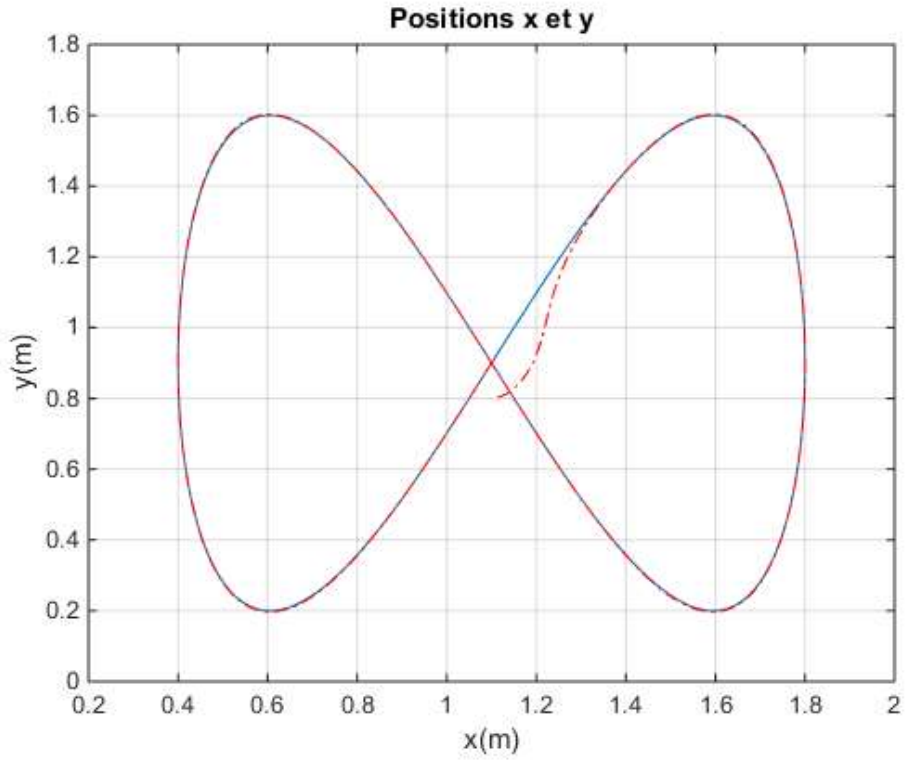


Figure III.8. Potions $x(t)$ et $y(t)$ ($x_{\text{ref}}(t)$ et $y_{\text{ref}}(t)$ trajectoire de référence en bleu, $x_{\text{réelle}}(t)$ et $y_{\text{réelle}}(t)$ en rouge discontinu).

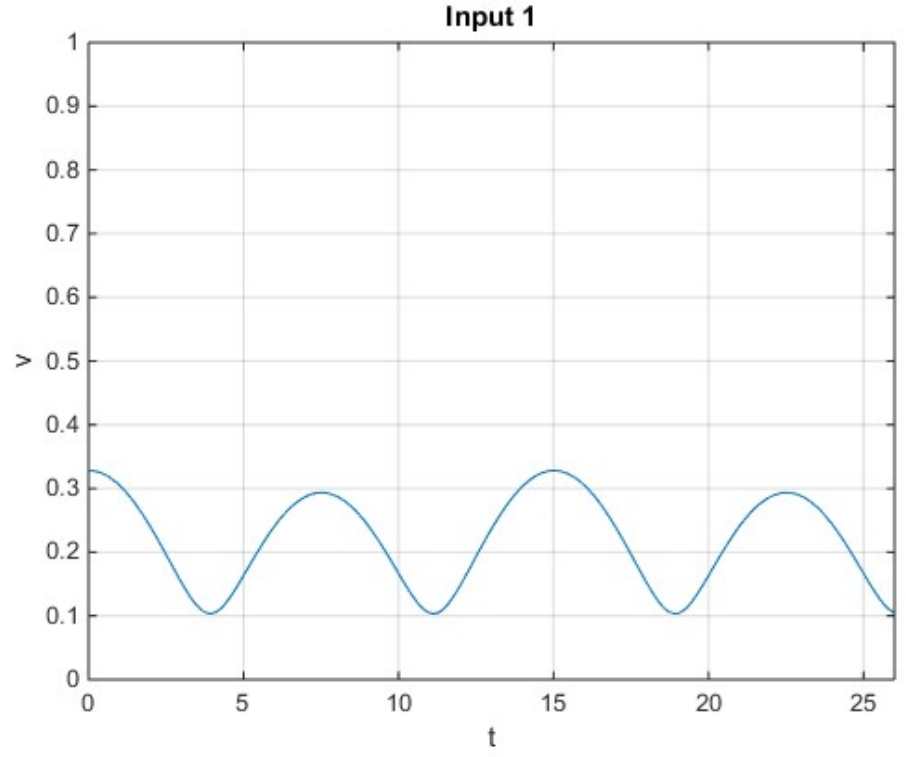


Figure III.9. Sortie 1 du régulateur (v).

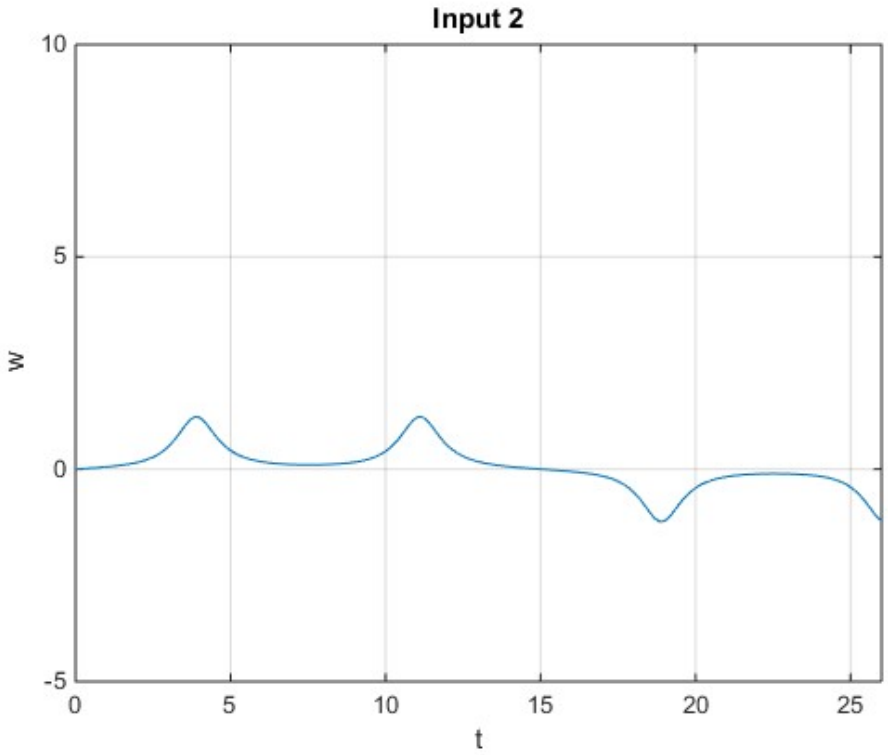


Figure III.10. Sortie 2 du régulateur (ω).

On observe clairement d'après les résultats obtenus, l'efficacité de l'approche présentée.

III.3.6. Développement d'un régulateur flou de type Takagi-Sugeno [2]

Comme nous l'avons déjà dit, le modèle d'erreur donné par (III.30) est non linéaire. En effet, les modèles de *Takagi-Sugeno* sont en réalité conçus afin de décrire la dynamique des systèmes non linéaires. Dans cette petite section, le modèle (III.30) sera réécrit sous la forme d'un Modèle de type *Takagi-Sugeno (TS)*. Cette forme permet la conception et la synthèse d'une loi de commande basée sur ce qu'on appelle ; Une compensation distribuée en parallèle (*Parallel Distributed Compensation-PDC*) dans la structure d'une inégalité matricielle linéaire (*Linear Matricial Inequality-LMI*).

III.3.6.1. Modèle d'erreur de type Takagi-Sugeno d'un robot mobile à roues différentielles (Differentially Driven Wheeled Mobile Robot)

Les modèles de type *Takagi-Sugeno (ou TS)* sont basés sur la logique floue où le modèle est exprimé sous la forme d'un ensemble de règles floues de type Si-Alors (*If-Then rules*). Un

modèle de type TS peut également être représenté sous une forme plus compacte dite en anglais *(polytopic form*) [28] donnée comme suit :

$$\dot{\xi} = \sum_{i=1} h_i\big(z(t)\big)\big(A_i\xi(t) + B_i u(t)\big)$$

$$v(t) = \sum_{i=1}^{r} h_i\big(z(t)\big)\big(C_i\xi(t)\big) \tag{III.35}$$

D'où $\xi(t) \in \mathbb{R}^n$ est un vecteur d'état, $v(t) \in \mathbb{R}^p$ est un vecteur de sortie, et $z(t) \in \mathbb{R}^q$ est le vecteur de prémisse qui dépend du vecteur d'état (ou autre quantité), A_i, B_i et C_i sont des matrices constantes. Et enfin, le les fonctions de pondération non linéaires $h_i\big(z(t)\big)$ sont toutes non négatives et telles que $\sum_{i=1}^{r} h_i\big(z(t)\big) = 1$ pour un $z(t)$ arbitraire. Pour tout modèle non linéaire, un peut trouver un modèle TS flou équivalent dans une région compacte de l'état variable spatiale en utilisant l'approche de non-linéarité sectorielle, qui consiste à décomposer chaque terme non linéaire borné en une combinaison convexe de ses limites [29].

Le nombre de règles r est alors lié au nombre de non-linéarités du modèle, comme il sera indiqué par la suite. Dans ce qui suit, basant sur le fait que dans un modèle d'erreur d'un robot mobile (spécifié) les fonctions non linéaires sont connues a priori, ce qui rend l'utilisation du concept susmentionné est très possible. Le modèle d'erreur non linéaire de suivi exprimé par (III.30) sera donc réécrit sous une forme matricielle équivalente comme suit :

$$\begin{bmatrix}\dot{e}_x\\ \dot{e}_y\\ \dot{e}_\varphi\end{bmatrix} = \begin{bmatrix}0 & \omega_{ref} & 0\\ -\omega_{ref} & 0 & v_{ref}\dfrac{\sin(e_\varphi)}{e_\varphi}\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}e_x\\ e_y\\ e_\varphi\end{bmatrix} + \begin{bmatrix}-1 & e_y\\ 0 & -e_x\\ 0 & -1\end{bmatrix}\begin{bmatrix}v_{fb}\\ \omega_{fb}\end{bmatrix} \tag{III.36}$$

Quatre fonctions non linéaires bornées apparaissent dans ce modèle : ω_{ref}, $v_{ref}\dfrac{\sin(e_\varphi)}{e_\varphi}$ (ou avec une notation différente $v_r sinc(\varphi)$), e_y et e_x. Le vecteur de prémisse est donc exprimé comme suit :

$$z(t) = \begin{bmatrix} \omega_{ref} \\ v_{ref} sinc(\varphi) \\ e_y \\ e_x \end{bmatrix} \quad \text{(III.37)}$$

Tout d'abord, la contrôlabilité de l'équation (III.36) au sens linéarisé sera analysé. Il est banal de voir qu'au voisinage de l'erreur zéro, le système (III.36) est contrôlable si v_{ref} est différent de 0 et $|e_\varphi|$ est différent de π, ou ω_{ref} est différent de 0. Dans les cas pratiques, ω_{ref} croise souvent 0, donc v_{ref} ne peut pas devenir 0, et $|e_\varphi|$ ne peut pas devenir π. Les hypothèses suivantes sont donc nécessaires pour éviter la perte de contrôlabilité et pour restreindre notre attention à une certaine région compacte de l'espace d'erreur [2] :

$$\begin{gathered} \underline{\omega}_{ref} \le \omega_{ref} \le \bar{\omega}_{ref} \\ |e_\varphi| \le \bar{e}_\varphi < \pi, 0 \le \underline{v}_{ref} < v_{ref} \le \bar{v}_{ref} \Rightarrow \bar{v}_{ref} sinc(\bar{e}_\varphi) \le v_{ref} sinc(\varphi) \le \bar{v}_{ref} \\ |e_y| \le \bar{e}_y \\ |e_x| \le \bar{e}_x \end{gathered} \quad \text{(III.38)}$$

Les bornes imposées sur v_{ref} et ω_{ref} sont obtenues à partir de la trajectoire de référence réelle, tandis que les limites de l'erreur de suivi sont sélectionnées sur la base de toute connaissance disponible à priori. Il est à noter que, que ces limites (bornes) doivent être inférieures à l'erreur due au bruit de mesure, aux erreurs initiales, etc. les limites mentionnées dans (III.38) sont notés $\underline{z}_j$ et $\bar{z}_j$, $j = 1, 2, 3, 4$. En réalité, Il existe quatre non-linéarités dans le système, donc le nombre de règles floues r est égal à $2^4 = 16$. Le modèle de TS de l'équation (III.38) est donc exprimé comme suit :

$$\dot{e}(t) = A_{z(t)} e(t) + A_{z(t)} u_{fb}(t) \quad \text{(III.39)}$$

Avec :

$$A_i = \begin{bmatrix} 0 & \varepsilon_i^1 & 0 \\ -\varepsilon_i^1 & 0 & \varepsilon_i^2 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_i = \begin{bmatrix} -1 & \varepsilon_i^3 \\ 0 & -\varepsilon_i^4 \\ 0 & -1 \end{bmatrix} \quad \text{(III.40)}$$

L'indice i parcourt de manière arbitraire tous les sommets du hypercube défini par l'équation (III.38). Il est à noter que, Le moyen le plus courant consiste à utiliser une

énumération binaire comme suit :

$$i_1 = \begin{cases} 0 & i \le \frac{r}{2} \\ 1 & si\ non \end{cases}$$

$$i_2 = \begin{cases} 0 & i - \frac{r}{2} i_2 \le \frac{r}{4} \\ 1 & si\ non \end{cases}$$

$$i_3 = \begin{cases} 0 & i - \frac{r}{2} i_1 - \frac{r}{4} i_2 \le \frac{r}{8} \\ 1 & si\ non \end{cases} \quad \text{(III.41)}$$

$$i_3 = \begin{cases} 0 & i - \frac{r}{2} i_1 - \frac{r}{4} i_2 - \frac{r}{8} i_3 \le \frac{r}{16} \\ 1 & si\ non \end{cases}$$

Ainsi, ε_i^j dans l'équation (III.38) est défini comme :

$$\varepsilon_i^j = \underline{z}_j + i_j\left(\overline{z}_j - \underline{z}_j\right),\ i = 1,2,\dots.,16,\ j = 1,2,3,4 \quad \text{(III.42)}$$

Et enfin les fonctions d'appartenance doivent être définies :

$$h_i(z) = w_{i1}^1(z_1) w_{i2}^2(z_2) w_{i3}^3(z_3) w_{i4}^4(z_4),\ i = 1,2,\dots.,16$$

$$w_1^j\left(z_j\right) = \frac{z_j - \underline{z}_j}{\overline{z}_j - \underline{z}_j},\ w_0^j\left(z_j\right) = 1 - w_1^j\left(z_j\right),\ j = 1,2,3,4 \quad \text{(III.43)}$$

$$\varepsilon_i^j = \underline{z}_j + i_j\left(\overline{z}_j - \underline{z}_j\right),\ i = 1,2,\dots.,16,\ j = 1,2,3,4$$

Le modèle de TS exprimé par (III.39) du modèle d'erreur de suivi représente en réalité le modèle exact du système (III.36), c'est-à-dire il prend en considération toutes les non-linéarités connues qui surviennent dans le système. La forme représentée par l'équation (III.39) est très appropriée à la conception et l'analyse comme il sera montré ci-dessous.

III.3.6.2. Commande basée sur l'approche PDC d'un robot mobile à roues différentielles

Afin de stabiliser le modèle de type TS exprimé dans (III.39), une loi de commande basée sur l'approche *PDC* qui a la forme donnée par (III.44) est utilisée.

$$u_{fb}(t) = -\sum_{i=1}^{r} h_i\big(z(t)\big)F_I e(t) = -F_{z(t)}e(t) \qquad \text{(III.44)}$$

Le problème de stabilisation en utilisant l'approche *PDC* est bien connu. En effet, ceci est dû à une structure spécifique qui caractérise l'emploi de cette approche, d'où le modèle du système à contrôler ainsi le régulateur répartissent les mêmes fonctions d'appartenance. De plus, cette approche offre la possibilité de traiter la stabilité du système d'une manière formelle et très simple. En somme, le système décrit par les deux l'équations (III.39) et (III.44) est asymptotiquement stable si pour chaque i et j la matrice $(A_i - B_i F_j)$ est dite matrice de *Hurwitz* (une matrice est dite de *Hurwitz* si toutes les valeurs propres se trouvent dans le demi-plan gauche du plan complexe en fonction de p). Il est à noter que, par conséquent le nombre de matrices à analyser augmente très rapidement ce qui rend l'emploi d'une approche systématique chose très nécessaire. Également, basant sur le travail publié dans [6-local], on peut observer clairement l'importance de l'outil *IMT*. La formulation des paramètres de système à commander sous forme de matrices A_i et B_i permet d'extraire l'ensemble des paramètres F_j du régulateur qui permet de stabiliser asymptotiquement le système à commander.

Une remarque importante doit être ajoutée, c'est que cette approche ne tient pas en compte les propriétés spécifiques du système à commander, ceci est apparu p.ex. sur la forme des fonctions d'appartenance utilisées.

Dans ce qui suit, est illustré l'algorithme de la méthode :

Le modèle de TS donné par (III.39), peut être stabilisé en utilisant la loi de commande basée sur l'approche PDC (voir (III.44)), s'elles existes des matrices M_i *avec* $(i = 1,2, \dots, r)$ *et un* $X > 0$*, telle que la condition LMI suivante est vérifiée :*

$\gamma_{ii} < 0 \quad i = 1,2, \dots, r$

$\frac{2}{r-1}\gamma_{ii} + \gamma_{ij} + \gamma_{ji} < 0, i, j = 1,2, \dots, r$ et $i \neq j$

Avec $\gamma_{ii} = XA_i^T + A_iX - M_j^T B_I^T - B_iM_j$ *ainsi les gains* F_i *de la loi de commande PDC (voir (III.44)) sont définis par l'expression* $F_i = M_iX^{-1}$.

III.3.7. Commande prédictive

Les approches de commandes dites prédictive basées sur un modèle (*Model-based Predictive Control-MPC*), sont des méthodes avancées qui peuvent être appliqués à divers domaines parmi eux le problème de poursuite d'une trajectoire de référence (connue préalablement) relatif à la robotique mobile. L'utilisation de cette approche de commande, plus précisément en robotique mobile repose généralement sur une linéarisation du modèle cinématique ou du modèle dynamique pour prédire les futurs états du système.

En trouve p.ex. dans [30], l'application d'une commande prédictive généralisée sur un robot mobile. Aussi, les auteurs de [31] ont réussi à appliquer une telle commande avec l'ajout d'un prédicteur dit en anglais (*Smith predictor*) afin de gérer les retards résides dans la dynamique du système commandé (robot mobile). Une commande *MPC* a été appliquée aussi en basant sur la description d'un système linéaire variant dans le temps dans [32]. Les auteurs de [33], ont appliqué un régulateur prédictif non-linéaire basé sur un modèle neuronal, le lecteur peut trouver autan d'applications de cette approche de commande dans la robotique mobile et plus précisément le problème de suivi d'une trajectoire.

Dans la plupart de ces approches lors de leur application, les solutions qui construisent la loi de commande sont obtenues généralement par une optimisation d'une fonction de coût donnée. D'autres approches, sont basées sur une recherche analytique des solutions, ces dernières sont considérées plus efficaces en termes de temps de calcul et facilité d'implémentation en temps réel (voir [34]).

Dans cette petite section, nous présentons la résolution du problème de suivi d'une trajectoire relative à un robot mobile à roues différentielles en utilisant une commande prédictive. Il est à noter que, la trajectoire de référence doit être une fonction temporelle lisse et deux fois différentiable. Pour la prédiction, un modèle dynamique a erreur linéarisée est obtenu autour de la trajectoire de référence utilisé. L'objective de la commande synthétisée, est de faire minimiser la différence entre l'erreur de suivi relative au robot (future erreur) et l'erreur de référence définie avec la dynamique souhaitée. En effet, la commande *MPC* synthétisée combine une solution anticipée (*feedforward solution*) et une action de retour (*feedback action*) dans le vecteur d'entrée u, elle est exprimée comme suit :

$$u = u_{ff} + u_{fb} = \begin{bmatrix} v_{ref} cos(e_\varphi) + v_{fb} \\ \omega_{ref} + \omega_{fb} \end{bmatrix} \quad \text{(III.45)}$$

D'où le vecteur d'entrée dit (*feedforward input*) $u_{ff} = \left[v_{ref} cos(e_\varphi) \omega_{ref}\right]^T$ est calculé à partir de la trajectoire de référence en utilisant les deux relations ([2] page 94, (3.22) et (3.23)). Le vecteur d'entrée de retour $u_{fb} = \left[v_{fb} \omega_{fb}\right]^T$ représente la sortie (commande générée) du régulateur *MPC*.

Le problème de commande est donné par l'erreur dynamique de suivi linéaire (III.36) définie sous forme courte par :

$$\dot{e} = A_c(t)e + B_C u_{fb} \quad \text{(III.46)}$$

D'où $A_c(t)$ et B_C sont des matrices du modèle d'espace d'états continu et e est l'erreur de suivi définie dans les coordonnées locales du robot, définie aussi par transformation (III.25), elle est illustrée sur la Figure III.7.

III.3.7.1. Commande MPC discrète

Une description d'une telle commande est présentée dans ce qui suit, en basant sur celle présentée dans [34]. En effet, cette dernière a une nature discrète. Par conséquent, le modèle (III.45) doit être discrétiser comme suit :

$$\dot{e}(k+1) = A(k)e(k) + Bu_{fb}(k) \quad \text{(III.47)}$$

Avec, $A(k) \in \mathbb{R}^n \times \mathbb{R}^n$, n est le nombre de variables d'état et $B \in \mathbb{R}^m \times \mathbb{R}^m$ et m est le nombre de variables d'entrée. Les matrices discrètes $A(k)$ et B peuvent être obtenus comme suit :

$$A(k) = I + A_c(t)T_s$$

$$B = B_c T_s \quad \text{(III.48)}$$

Cette dernière représente une très bonne approximation pour une période d'échantillonnage T_s courte. En effet, l'idée principale d'une commande *MPC* est de faire calculer des commandes optimales qui permettent de minimiser une fonction objectif définie dans un intervalle de prédiction d'horizon h. La fonction objectif est une fonction de coût de type quadratique, exprimée comme suit :

$$J(u_{fb}, k) = \sum_{i=1}^{h} \epsilon^T(k,i) Q \epsilon(k,i) + u_{fb}^T(k+i-1)\, R u_{fb}(k+i-1) \quad \text{(III.49)}$$

Analysant cette fonction, on trouve qu'elle est composée de l'erreur future de suivi de référence $e_r(k+i)$, l' erreur de suivi prédite $e_r(k+i|k)$, une différence entre ces deux erreurs $\epsilon^T(k,i) = e_r(k+i) - e_r(k+i|k)$, et les futures commande $u_{fb}(k+i-1)$, d'où le i désigne l'nième pas en avant de prédiction $(i = 1, \ldots, h)$ et de Q et R qui sont des matrices de pondération.

Pour la prédiction de l'état $e(k+i|k)$, le modèle d'erreur (III.47) est appliqué comme suit :

$$e(k+1|k) = A(k)e(k) + Bu_{fb}(k)$$
$$e(k+2|k) = A(k+1)e(k+1|k) + Bu_{fb}(k+1)$$
....
$$e(k+i|k) = A(k+i-1)e(k+i-1|k) + Bu_{fb}(k+i-1) \quad \text{(III.50)}$$
....
$$e(k+h|k) = A(k+h-1)e(k+h-1|k) + Bu_{fb}(k+h-1)$$

Les éléments de prédictions $e(k+i|k)$ apparus dans l'équation (III.50) sont réorganisés de telle manière à être dépendants de l'erreur actuelle $e(k)$, les entrées actuelles et futures $u_{fb}(k+i-1)$ ainsi que les matrices $A(k+i-1)$ et B. Par conséquent, la prédiction de sortie du modèle (sortie prédite) à l'instant h peut alors être écrite comme :

$$e(k+i|k) = \prod_{j=1}^{h-1} A(k+1)\, e(k) + \sum_{i=1}^{h} \left(\prod_{j=1}^{h-1} A(k+1) \right) B u_{fb}(k+1-1) + B u_{fb}(k+h-1) \quad \text{(III.51)}$$

L'erreur de référence future $e_r(k+i)$ définit comment l'erreur de suivi devrait diminuer lorsque le robot n'est pas sur la trajectoire. Définissons l'erreur de référence future pour décroître de manière exponentielle à partir de l'erreur de suivi actuelle $e(k)$, comme suit :

$$e_r(k+i) = A_r^i e(k) \quad \text{(III.52)}$$

Pour $i = 1, \ldots, h$, la dynamique de l'erreur de référence est définie par la matrice du modèle de référence A_r. Ainsi, selon les deux équations (III.50) et (III.51) le vecteur d'erreur de prédiction de suivi du robot le est défini comme suit :

$$E^*(k) = [e(k+1|k)^T\ e(k+2|k)^T \ldots\ e(k+h|k)^T]^T$$

D'où E^* est donné pour tout l'intervalle de l'observation (h) où le vecteur de commande est donné comme suit :

$$U_{fb}(k) = \left[u_{fb}^T(k)\ \ u_{fb}^T(k+1) \ldots\ u_{fb}^T(k+h-1)\right]^T \quad \text{(III.53)}$$

Ainsi,

$$\Lambda(k,i) = \prod_{j=1}^{h-1} A(k+j)$$

Le vecteur d'erreur de prédiction de suivi du robot peut être écrite en format compact comme suit :

$$E^*(k) = F(k)e(k) + G(k)u_{fb}(k) \tag{III.54}$$

D'où :

$$F(k) = [A(k)A(k+1)A(k) \ldots\ldots \Lambda(k,0)]^T \tag{III.55}$$

et

$$G(k) = \begin{bmatrix} B & 0 & \cdots & 0 \\ A(k+1)B & B & \ldots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda(k,1)B & \Lambda(k,2)B & \ldots & B \end{bmatrix} \tag{III.56}$$

D'où les dimensions de $F(k)$ et $G(k)$ sont $(n \cdot\ h\ \times n)$ et $(n\ \cdot\ h \times\ m \cdot h)$ respectivement. Le vecteur d'erreur de suivi de référence du robot est :

$$E_r^*(k) = [e_r(k+1)^T\ e_r(k+2)^T \ldots\ldots\ e_r(k+h)^T]^T$$

$E_r^*(k)$ est sous forme compacte, elle est calculée comme suit :

$$E_r^*(k) = F_r e(k) \tag{III.57}$$

Avec ;

$$F_r = [A_r A_r A_r^2 \ldots\ldots A_r A_r^h] \tag{III.58}$$

D'où F_r et une matrice de dimension $(n \cdot\ h\ \times n)$.

Le vecteur de commande optimal (III.53) est obtenu par optimisation de l'équation (III.49), ce qui peut être fait numériquement ou analytiquement.

La solution analytique est déterminée comme suit :

Premièrement, la fonction objectif (III.49) définie sous forme matricielle, devient :

$$J(U_{fb}) = (E_r^* - E^*)^T \bar{Q}(E_r^* - E^*) + U_{fb}^t \bar{R} U_{fb} \quad \text{(III.59)}$$

Le minimum de l'équation (III.58) est exprimé comme suit :

$$\frac{\partial J}{\partial U_{fb}} = -2\bar{Q}G^T E_r^* + 2G^T \bar{Q}E^* + 2\bar{R}U_{fb} = 0 \quad \text{(III.60)}$$

Ainsi, le vecteur de commande optimale est obtenu comme suit :

$$U_{fb}(k) = (G^T \bar{Q} G + \bar{R})^{-1} G^T \bar{Q}(F_r - F)e(k)$$

D'où : $\bar{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix}$ et $\bar{R} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}$ (III.61)

La solution (III.61) ($\bar{R}$) contient les vecteurs de commandes $u_{bt}^T(k+i-1)$ pour tout l'intervalle de prédiction $(i = 1, \dots, h)$. Afin d'appliquer une commande avec retour (feedback control action) à l'instant k, seul le premier vecteur $u_{bt}^T(k)$ (m premières lignes de $u_{bt}(k)$) est appliqué au robot. Il est à noter que, la solution est obtenue analytiquement ce qui offre une bonne possibilité d'une implémentation en temps réel qui ne pas être possible si l'optimisation numérique de l'équation (III.49) est utilisée.

Pour maitre les choses relatives à cette approche très claire, nous présentons dans ce qui suit un exemple illustratif avec simulation sous les mêmes conditions des exemples illustratifs précédents (voir pages 65 et 72).

Plus précisément, on implémente la loi de commande définie dans (III.61) pour un robot mobile à roues différentielles afin de suivre la trajectoire définie dans les exemples illustratifs précédents.

Lors de l'implémentation, nous avons utilisé un horizon de prédiction égal $h = 4$, la matrice de modèle de référence $A_r = I_{3\times3} \cdot 0.65$, les matrices Q et R sont données comme suit :

$$Q = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \text{ et } R = I_{2\times 2} \cdot 10^{-3}$$

Le détail de cet exemple est illustré dans le script MATLAB (voir le code MATLAB M-3 dans l'annexe).

On Implémente l'algorithme de commande présenté en utilisant le Matlab, ainsi les tous résultats obtenus sont illustrés dans les figures Figure III.11, Figure III.12 et Figure III.13.

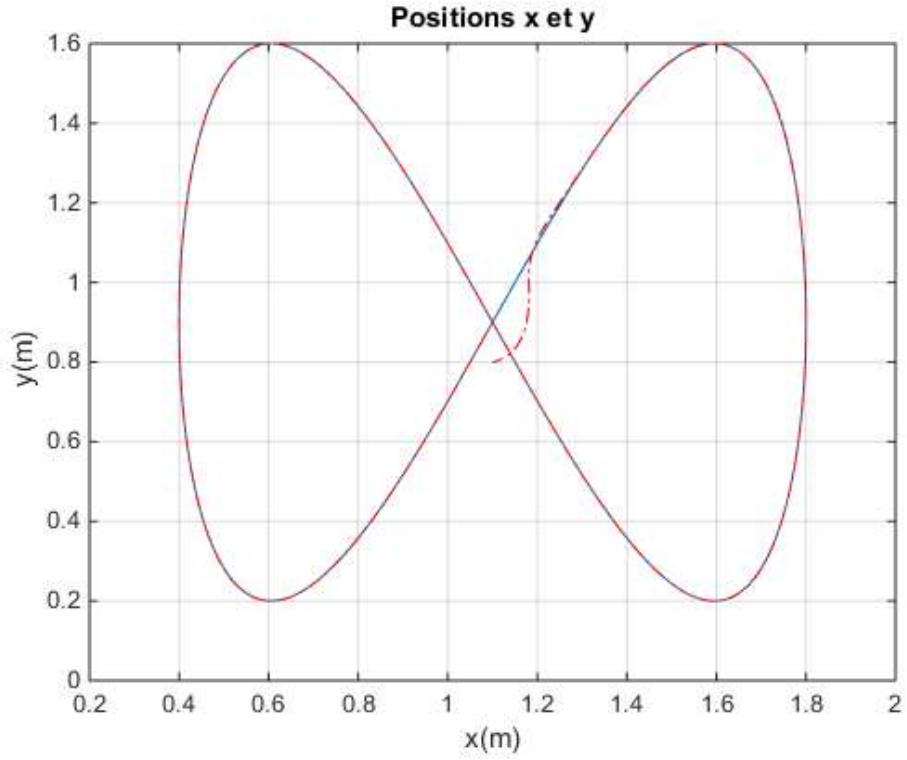


Figure III.11. Potions $x(t)$ et $y(t)$ ($x_{\text{ref}}(t)$ et $y_{\text{ref}}(t)$ trajectoire de référence en bleu, $x_{\text{réelle}}(t)$ et $y_{\text{réelle}}(t)$ en rouge discontinu).

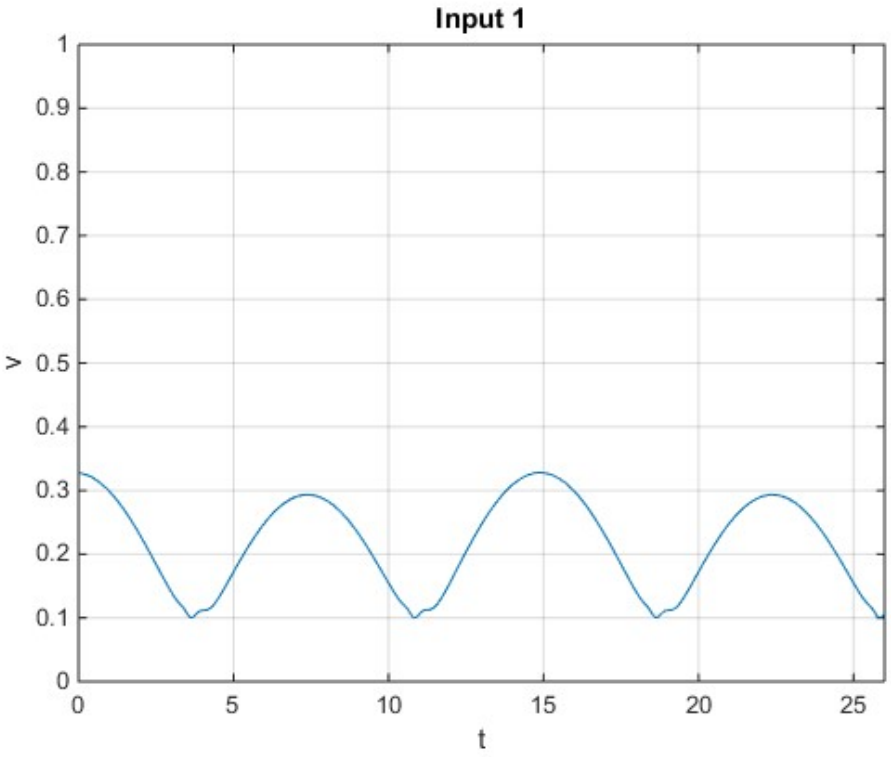


Figure III.12. Commande MPC 1 (v).

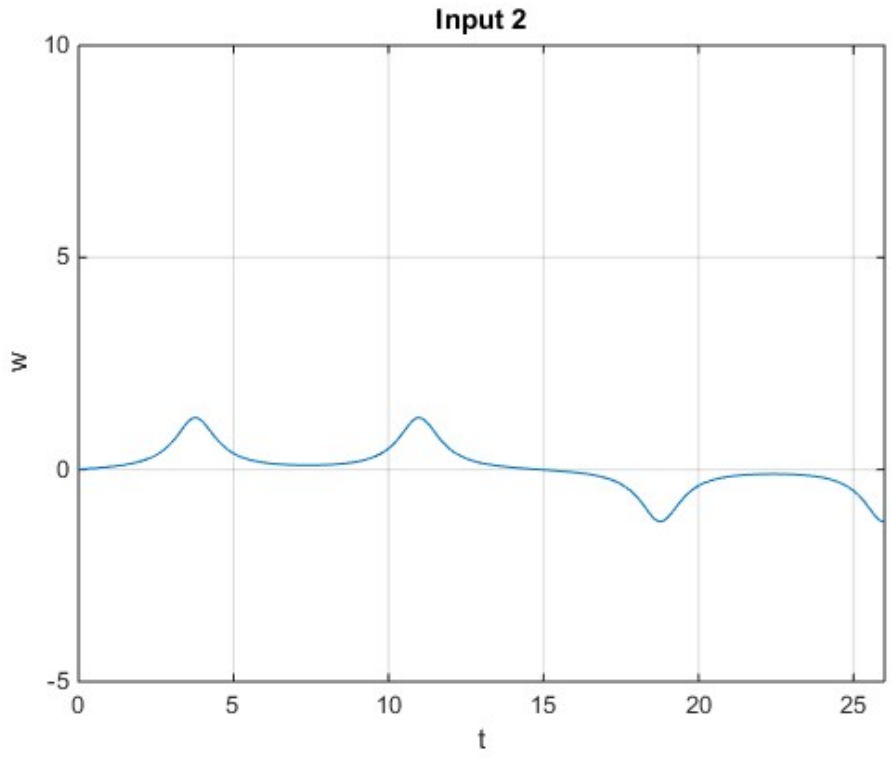


Figure III.13. Commande MPC 2 (ω).

III.4. Conclusion

Le problème de suivi d'une trajectoire de référence pour un robot mobile non-holonome au sens général a été présenté dans chapitre. En effet, le sujet suivi d'une trajectoire pour un robot mobile à travers une loi de commande synthétisée est considéré très important vu l'importance de l'utilisation des robots mobiles dans des différentes tâches relatives aux activités spécifiques dans notre vie.

Dans ce chapitre, nous avons présenté le problème en question en deux grandes parties principales, dans la première nous avons concentré sur l'environnement du robot, c-à-d. le traitement de ce problème lors de la présence de deux scenarios différents *(i) suivi d'une trajectoire avec contrôle/sans contrôle d'orientation* et *(ii) suivi d'une trajectoire stabilisation/sans stabilisation.* Dans la seconde partie, nous avons montré en détaille quelques approches de commande relatives à ce problème.

Ce chapitre est doté d'un ensemble d'exemples illustratifs de simulations, dont le but est de faire prouver et clarifier les différentes approches présentées.

Dans le chapitre suivant, nous serons plus pointés dans notre présentation. Nous nous parlerons sur le problème de suivi d'une trajectoire de référence pour un robot mobile non-holonome en synthétisant une loi de commande basée sur l'emploi d'un algorithme de type métaheuristique dit ; algorithme d'Optimisation par Essaim Particulaire (OEP) et en anglais Particle Swarm Optimization (PSO). Le chapitre suivant présente ainsi tous les résultats de simulations lors de l'application de cette méthode de commande.

Chapitre IV

Chapitre IV :

Suivi de trajectoire pour un robot mobile en utilisant un algorithme PSO

IV.1. Introduction

Le commande d'un robot mobile peut être aussi définie comme un problème d'optimisation où la meilleure solution parmi toutes les solutions possibles (dans un espace de recherche) est trouvée à chaque temps d'échantillonnage lors du processus de réglage (commande). En effet, la commande obtenue n'a pas une structure explicite, ce qui signifie que la loi de commande ne peut pas être donnée comme une fonction de cartographie (*mapping function*) des états du système aux commande générée. Une solution optimale qui minimise une fonction objectif donnée peut être trouvée par un algorithme d'optimisation de type itératif quel que soit son type, classique tels que les méthodes de Newton, les méthodes de descentes ou telles que les méthodes stochastiques comme p.ex. les algorithmes génétiques [36], les algorithmes de colonies de fourmis [37] et l'algorithme d'Optimisation par Essaim Particulaire (OEP) [35] avec lequel on cherche à synthétiser (dans ce mémoire) une loi de commande appliquée au problème de suivi d'une trajectoire pour un robot mobile non-holonome à roues différentielles.

IV.1. Applications de l'algorithme OPE aux problèmes de commandes (aspect général)

L'algorithme OPE a connu une large application dans le domaine de la commande des systèmes. On cite dans ce qui suit quelques travaux dans les quels cet algorithme a été utilisé.

Les deux auteurs Cai et Yang [38] ont proposé une version améliorée de OPE pour un ensemble de robots mobiles utilisés afin de chercher des cibles sous une forme coopérative dans des environnements inconnus. Les règles de coopération améliorées pour le système multirobots ont été appliquées dans la fonction de champ potentiel, qui a agi comme la fonction objectif utilisée par le l'algorithme OPE.

Le problème de commande de la fréquence de charge (*Load Frequency Control LFC*) en utilisant différents types d'algorithmes d'optimisation pour deux types de système de configuration de puissance a été résolu dans [39]. Le schéma de commande proposée est basé sur l'emploi des inégalités matricielles linéaires (Linear Matrix Inequalities LMIs) regelées par

l'algorithme OPE.

Également, l'algorithme OPE a été utilisé dans [40] pour régler un régulateur PID, en utilisant un palier magnétique actif qui a suspendu l'arbre rotatif et le maintenu en position d'élévation en appliquant des forces électromagnétiques contrôlées sur le rotor dans les directions radiale et axiale.

Les chercheurs Nedic et ses collègues [41], ont proposé une nouvelle commande de force de charge en cascade conçue pour une plateforme a robot parallèle (a parallel robot platform). L'algorithme OPE est utilisé dans ce dernier travail, afin de chercher les paramètres optimaux des régulateurs placés en cascade. Les résultats de simulation obtenus lors de l'application de cette méthode, montrent l'efficacité de cette dernière (entrainement optimal des régulateurs placés en cascade) dans le problème de suivi des trajectoires données.

On trouve aussi dans [42], l'optimisation des paramètres d'un régulateur PID en utilisant un algorithme OPE amélioré (l'équation de la mise-à-jour des vitesses des particules) appliqué pour le réglage des système multivariables de type (Multi-inputs Multi outputs MIMO).

Le groupe de Xiang [43], ont proposé un nouveau schéma de commande sans l'emploi des capteurs de retour (capteurs de positions). En effet, la commande synthétisée avec l'aide d'un algorithme OPE et quelques techniques d'indentification, dont le but est de faire satisfaire les performances de suivi relatives aux actionneurs de conduction de polymère sous des perturbations, a prouvé son efficacité.

Le problème de stabilisation d'un hélicoptère autonome avec la présence des perturbations dû au vent a été traité avec l'emploi d'un algorithme OPE dans [44]. Plus précisément, le OPE est utilisé afin d'automatiser le processus de réglage basé sur un régulateur conçu à la base d'une régulation non linéaire adaptative de la sortie et d'une stabilisation robuste d'une chaîne d'intégrateurs.

Une commande basée sur une nouvelle version de OPE dite (MOPSO), appliquée à un robot bipède marchant dans un plan latéral en pente, est présentée dans [46].

Dans le même stade, Zhong et ses collègues [47], ont synthétisé un régulateur PID modifié appliqué à un régulateur de pression à gaz. Plus précisément, la méthode proposée est basée sur l'utilisation des réseaux de neurones de type RBF (Radial Basis Function) basées sur un algorithme OPE, dont le but (algorithme OPE) est de faire ajuster on-line les paramètres du régulateur PID.

L'algorithme OPE, est utilisé aussi (voir [48]) en combinaison avec un réseaux RBF récurant flou (RBFNN) afin de déterminer le point opérationnel pour un régulateur PID appliqué à une turbine d'aire et de faire identifier la région de stabilité.

IV.3. Algorithme d'Optimisation par Essaim Particulaire (OEP)

L'optimisation par essaim particulaire (OEP), ou Particule Swarm Optimisation (PSO) en anglais, est un algorithme évolutionnaire qui utilise une population de solutions candidates pour développer une solution optimale au problème. Cet algorithme a été proposé par *Russel Eberhart* (ingénieur en électricité) et *James Kennedy* (socio-psychologue) en 1995 [35]. Il s'inspire à l'origine du monde du vivant, plus précisément du comportement social des animaux évoluant en essaim, tels que les bancs de poissons et les vols groupés d'oiseaux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une « intelligence » limitée, et ne dispose que d'une connaissance locale de sa situation dans l'essaim. L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que « rester proche des autres individus », « aller dans une même direction » ou « aller à la même vitesse », suffisent pour maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. L'essaim de particules correspond à une population d'agents simples, appelés particules. Chaque particule est considérée comme une solution du problème, où elle possède une position (le vecteur solution) et une vitesse. De plus, chaque particule possède une mémoire lui permettant de se souvenir de sa meilleure performance (en position et en valeur) et de la meilleure performance atteinte par les particules « voisines » (informatrices) : chaque particule dispose en effet d'un groupe d'informatrices, historiquement appelé son voisinage.

IV.3.1. Principe de l'algorithme

Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, « survole » l'espace de recherche, à la recherche de l'optimum global. Le déplacement d'une particule est influencé par les trois composantes suivantes :

1. Une composante d'inertie : la particule tend à suivre sa direction courante de déplacement ;

2. Une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée.

3. Une composante sociale : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

Dans un espace de recherche de dimension D, la particule i de l'essaim est modélisée par son vecteur position $\overrightarrow{x_i}$ = (x_{i1} , x_{i2} , …, x_{iD}) et par son vecteur vitesse $\overrightarrow{v_i}$ = (v_{i1} ,v_{i2} , … ,v_{iD}). La qualité de sa position est déterminée par la valeur de la fonction de coût en ce point. Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note $\vec{P}best_i$ = ($pbest_{i1}$, $pbest_{i2}$, … , $pbest_{iD}$). La meilleure position atteinte par les particules de l'essaim est notée $\vec{G}best$ = ($gbest_1$, $gbest_2$, … , $gbest_D$). Nous nous référons à la version globale de PSO, où toutes les particules de l'essaim sont considérées comme voisines de la particule i, d'où la notation $\vec{G}best$ ($global\ best$).

La figure ci-dessous (Figure IV.1), montre le modèle du mouvement d'une particule donnée.

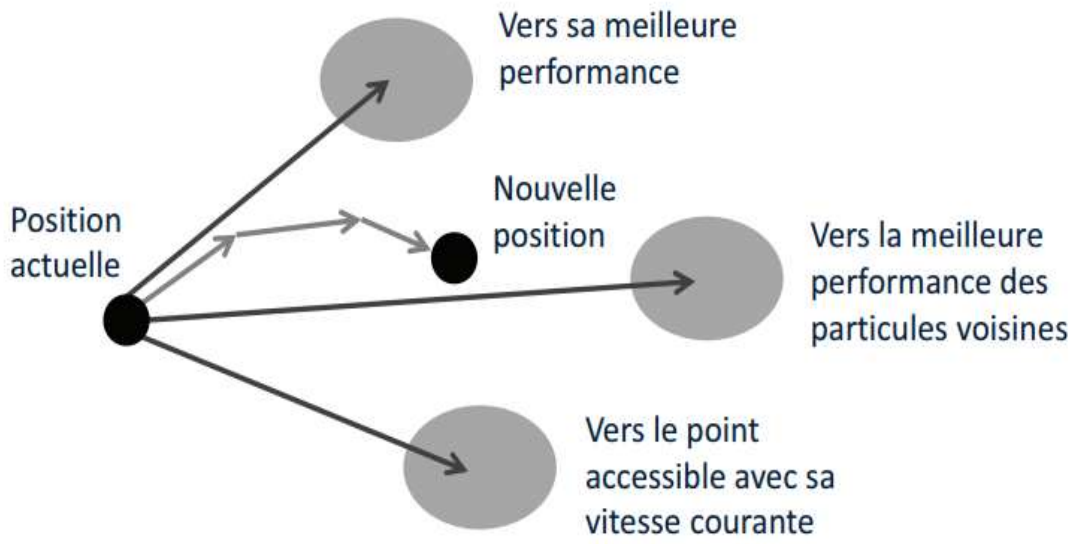


Figure IV.1 : Modèle de déplacement d'une particule.

Remarque

Le terme de « vitesse » est ici abusif, car les vecteurs $\overrightarrow{v_i}$ ne sont pas homogènes à une vitesse. Il serait plus approprié de parler de « direction de déplacement ». Cependant, pour respecter

l'analogie avec le monde animal, les auteurs ont préféré utiliser le terme de « vitesse ». Au départ de l'algorithme, les particules de l'essaim sont initialisées de manière aléatoire/régulière dans l'espace de recherche du problème. Ensuite, à chaque itération, chaque particule se déplace, en combinant linéairement les trois composantes citées ci- dessus.

En effet, à l'itération $t+1$, le vecteur vitesse et le vecteur position sont calculés à partir de l'équation (IV.1) et de l'équation (IV.2), respectivement.

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 r_{1_{i,j}}^{t} [\, pbest_{i,j}^{t} - x_{i,j}^{t} \,] + c_2 r_{2_{i,j}}^{t} [\, gbest_{j}^{t} - x_{i,j}^{t} \,] \,, j \in \{1\,, 2\,, \dots\,, D\} \qquad \text{(IV.1)}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \,, j \in \{1\,, 2\,, \dots\,, D\} \qquad \text{(IV.2)}$$

Où w est une constante, appelée coefficient d'inertie ; c_1 et c_2 sont deux constantes, appelées coefficients d'accélération ; r_1 et r_2 sont deux nombres aléatoires tirés uniformément dans $[0,1]$, à chaque itération t et pour chaque dimension j.

Les trois composantes mentionnées ci-dessus (i.e. d'inertie, cognitive et sociale) sont représentées dans l'équation (IV.1) par les termes suivants :

1. $wv_{i,j}^{t}$ correspond à la composante d'inertie du déplacement, où le paramètre w contrôle l'influence de la direction de déplacement sur le déplacement futur,
2. $c_1 r_{1_{i,j}}^{t} [\, pbest_{i,j}^{t} - x_{i,j}^{t} \,]$ correspond à la composante cognitive du déplacement, où le paramètre c_1 contrôle le comportement cognitif de la particule,
3. $c_2\, r_{2_{i,j}}^{t}\ [gbest_{j}^{t} - x_{i,j}^{t}\ ]$ correspond à la composante sociale du déplacement, où le paramètre c_2 contrôle l'aptitude sociale de la particule.

Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées et les deux vecteurs $\vec{P}best_i$ et $\vec{G}best$ sont mis à jour, à l'itération $t+1$, suivant les deux équations (IV.3) (dans le cas d'une minimisation) et (IV.4) (dans une version globale de PSO), respectivement. Cette procédure est présentée dans l'Algorithme ci-dessous, où N est le nombre de particules de l'essaim.

$$\vec{P}best_i(t+1) = = \begin{Bmatrix} \overrightarrow{P}best_i(t), \ si\ f\big(\overrightarrow{x_i}(t+1)\big) \geq \vec{P}best_i(t) \\ \overrightarrow{x_i}\,(\,t+1\,)\,, sinon \end{Bmatrix} \qquad \text{(IV.3)}$$

$$\vec{G}best(t+1) = argmin_{\vec{P}best_i} f\left(\vec{P}best_i\,(t+1)\right), 1 \leq i \leq N \quad \text{(IV.4)}$$

Algorithme PSO de base

1 **Initialiser** aléatoirement N particules : position et vitesse,
2 **Evaluer** les positions des particules,
3 **Pour** chaque particule i, $\vec{P}best_i = \overrightarrow{x_i}$
4 **Calculer** $\vec{G}best$ selon (IV.4)
5 **Tant que** le critère d'arrêt n'est pas satisfait faire :
6 Déplacer les particules selon (IV.1) et (IV.2),
7 Evaluer les positions des particules,
8 Mettre à jour $\vec{P}best_i$ et $\vec{G}best$ selon (IV.3) et (IV.4),
9 **Fin**

IV.4. Suivi de trajectoire pour un robot mobile en utilisant un algorithme OPE

Dans ce qui suit, nous présentons le problème de suivi d'une trajectoire de référence pour un robot mobile non-holonome en utilisant un algorithme OPE. En effet, on travaille à développer une loi de commande basée sur cette approche.

Afin de permettre au lecteur de comprendre aisément cette méthode c.à.d. la synthèse d'une commande pour l'objective expliquée ci-dessus à base d'un algorithme OPE, nous commençant par une application sur la même trajectoire imposée dans les exemples illustratifs présentés au troisième chapitre.

Plus détaillé, l'algorithme OPE est appliqué à chaque période d'échantillonnage $t = kT_s$ afin de trouver les commandes optimales (vitesse linéaire et triangulaire c.à.d. v et ω) qui permettent au robot de suivre avec précision la trajectoire de référence imposé à lui ($x_{ref}(t)$ et $y_{ref}(t)$). Par la suite, la position prédire de robot est calculée a base de sa cinématique ainsi la meilleure solution trouvée par l'algorithme OPE (commandes optimales trouvées).

Il est à noter que, nous avons implémenté une commande prédictive (voir la section III.3.7) avec l'emploi d'un algorithme OPE.

Les commandes optimales générées, ont alors le but de minimiser la fonction objectif suivante ; $J(t) = e(t)^T Q e(t) + u_{fb}^T R u_{fb}$ formulée à la base de l'erreur de poursuite

$e(t) = [e_x(t), e_y(t), e_\varphi(t)]^T$ et les deux commandes dite de retour $u_{fb}^T = [v_{fb}\ \omega_{fb}]$, avec les matrices Q et R sont diagonales prédéfinies.

La loi de commande synthétisée, comprend une commande anticipée (*feedforward action*) et une autre de retour (*feedback action*). Il est à noter que, la commande anticipée est obtenue à partir d'une trajectoire connue tandis que l'action de retour est calculée à l'aide du OPE. De plus, l'erreur de suivi est exprimée en coordonnées locales du robot (voir la Figure III.7).

En effet, l'erreur dans la coordonnée x locale peut tout simplement compensée par la vitesse de translation v et les deux erreurs sur y et φ peuvent être compensées par la vitesse angulaire ω. Cependant, ce n'est pas le cas lors de l'utilisation d'une erreur de suivi globale due à la transformation de rotation non-linéaire exprimée par (III.24).

Ci-dessous, est représenté le processus de générations des deux commandes optimales v^*, ω^* générées après un cycle d'entrainement complet (itération courante = $Iter_{max}$) à chaque instant d'échantillonnage.

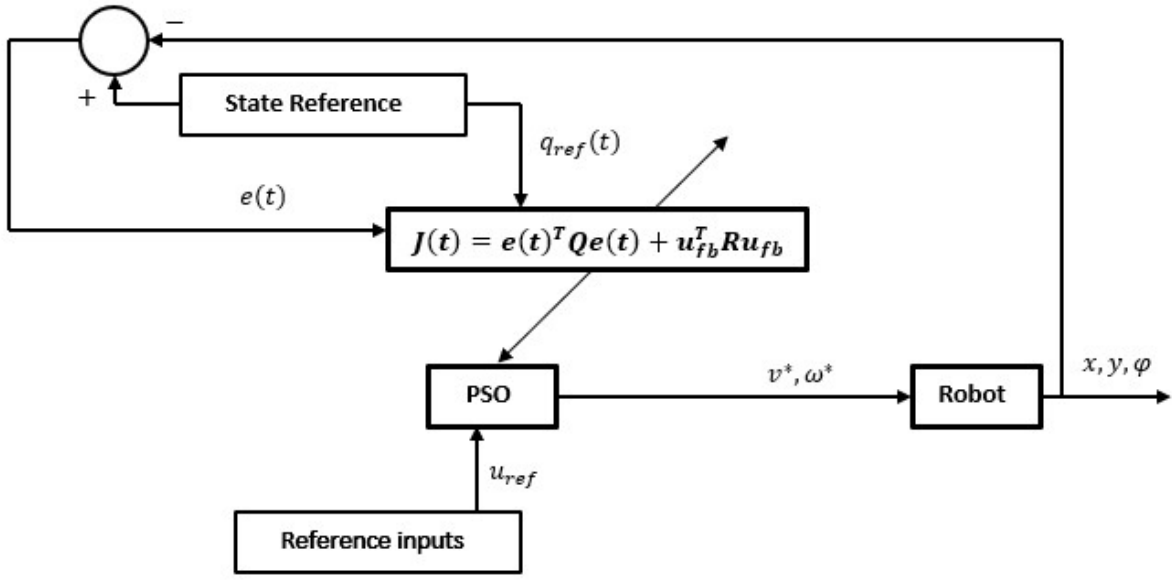


Figure IV.2 : Commande d'un robot mobile basée sur un algorithme OPE (PSO).

Le détail de cet exemple est illustré dans le script MATLAB (voir le code MATLAB M-4 dans l'annexe). En effet, le lecteur peut voir clairement sur ce script l'objectif de l'algorithme OPE.

Le tableau ci-dessous, donne les différents paramètres de contrôle du PSO utilisé :

Paramètres	Valeurs
w	0.25
r_1 et r_2	1.5
$Iter_{max}$ (nombre maximal d'itérations)	20
$N_{pop} = N$	25

Tableau IV.1 : Paramètres du PSO.

On Implémente l'algorithme de commande présenté en utilisant le MATLAB, ainsi tous les résultats obtenus sont illustrés dans les figures Figure IV.3, Figure IV.4 et Figure IV.5.

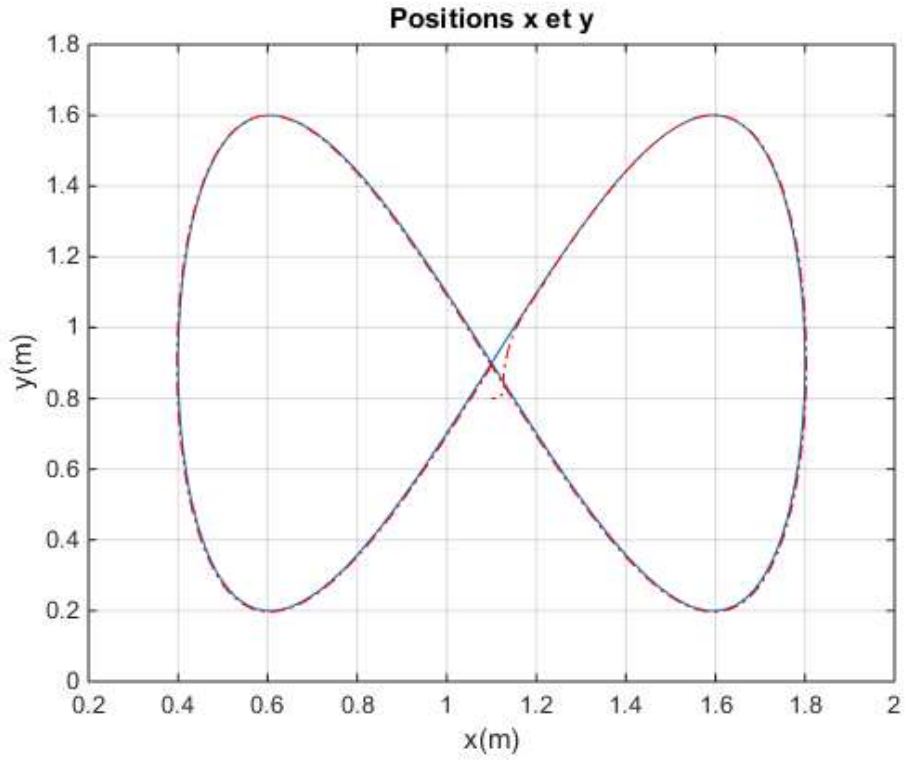


Figure IV.3. Potions $x(t)$ et $y(t)$ ($x_{\text{ref}}(t)$ et $y_{\text{ref}}(t)$ trajectoire de référence en bleu, $x_{\text{réelle}}(t)$ et $y_{\text{réelle}}(t)$ en rouge discontinu).

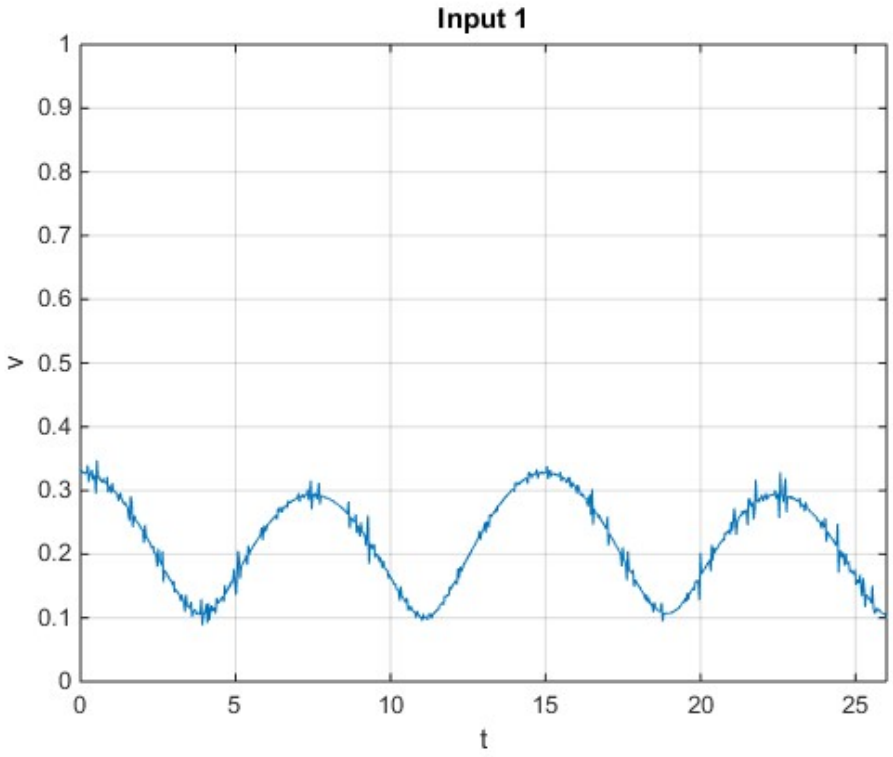


Figure IV.4. Entrée 1 du robot, générée par l'algorithme OPE (v).

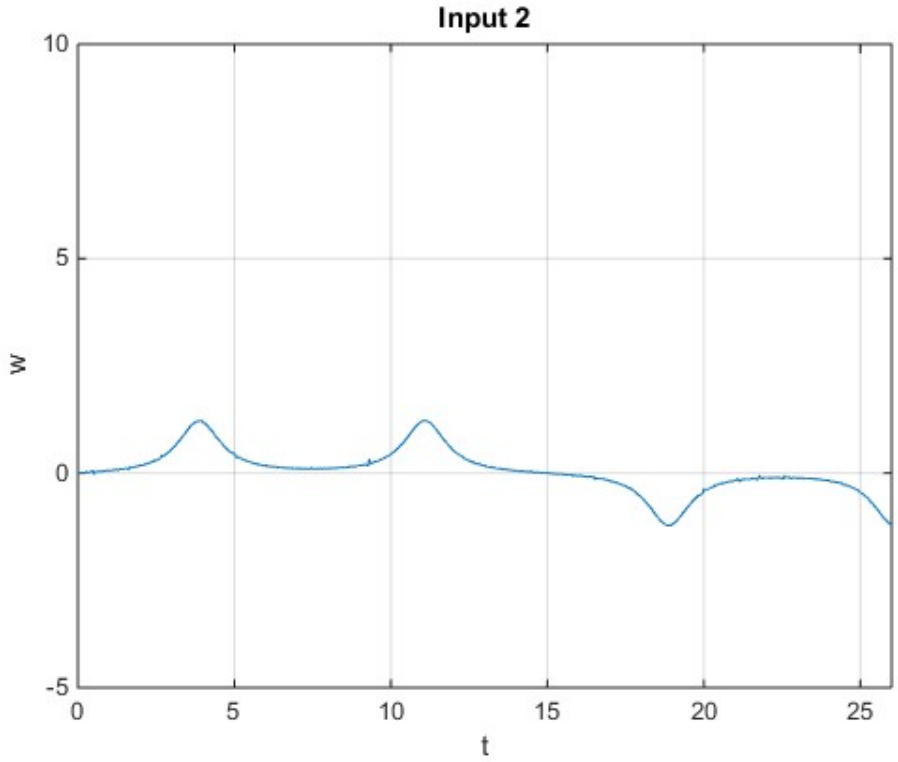


Figure IV.5. Entrée 2 du robot, générée par l'algorithme OPE (ω).

Afin de prouver l'efficacité de cette méthode, on refaire la simulation en utilisant une trajectoire différente (relatives aux x_{ref} et y_{ref} respectivement), son expression est la suivante :

Trajectoire de test (2) :

$x_{\text{ref}} = 2{,}7 + 0{,}25sin(\frac{2\pi t}{30})$ et $y_{\text{ref}} = 1{,}9 + 0{,}2sin(\frac{4\pi t}{30})$.

Les résultats obtenus, sont illustrés sur les figures Figure IV.6, Figure IV.7 et Figure IV.8.

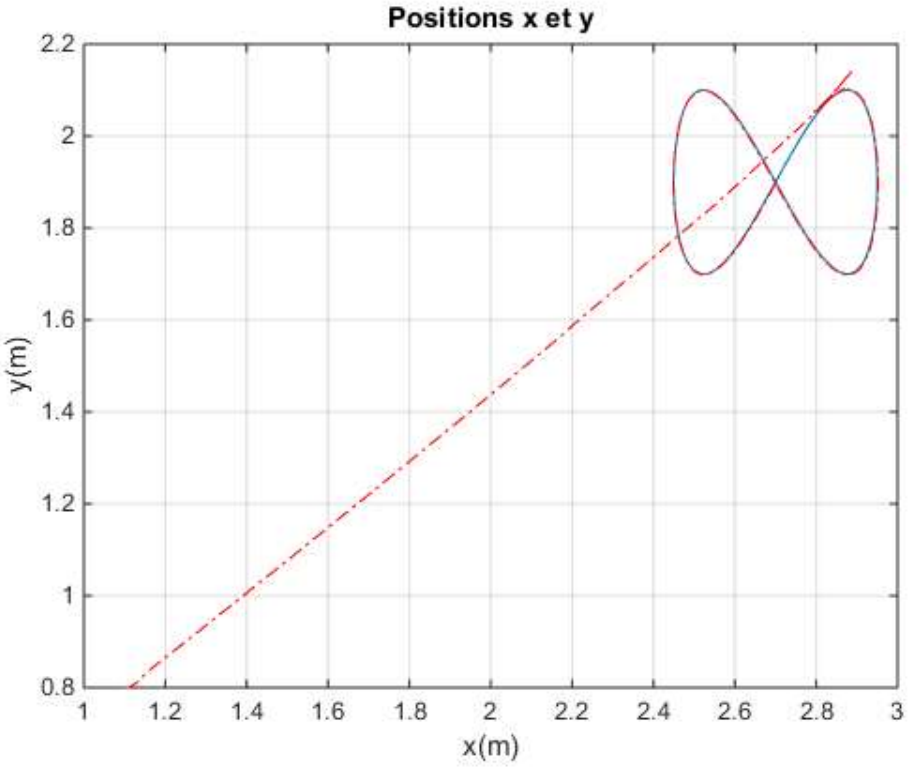


Figure IV.6. Potions $x(t)$ et $y(t)$ ($x_{\text{ref}}(t)$ et $y_{\text{ref}}(t)$ trajectoire de référence en bleu, $x_{\text{réelle}}(t)$ et $y_{\text{réelle}}(t)$ en rouge discontinu).

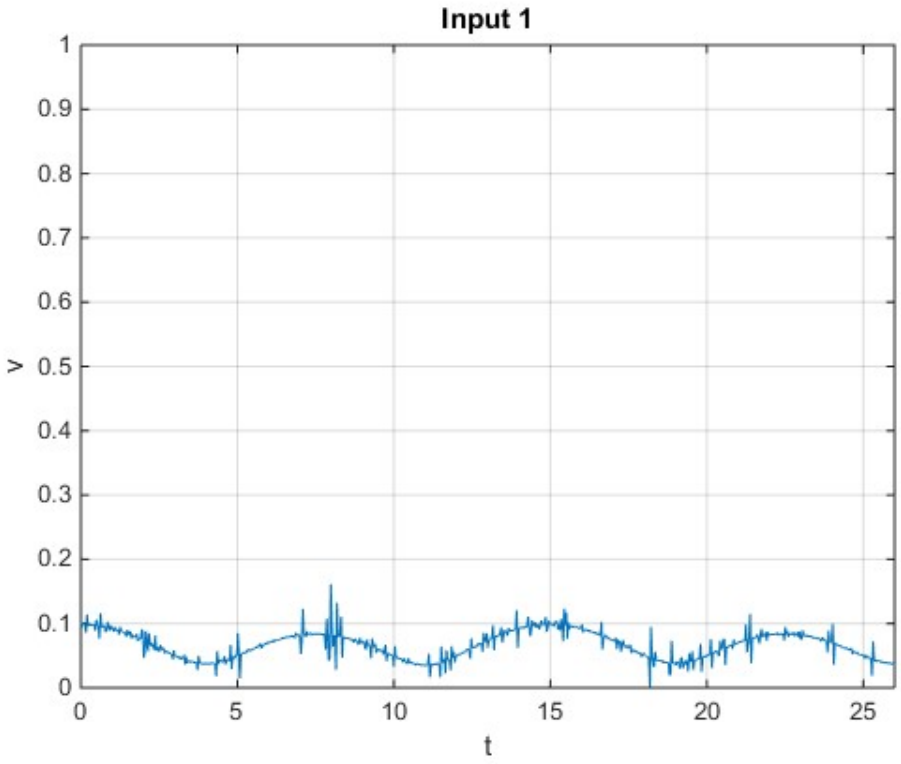


Figure IV.7. Entrée 1 du robot, générée par l'algorithme OPE (v).

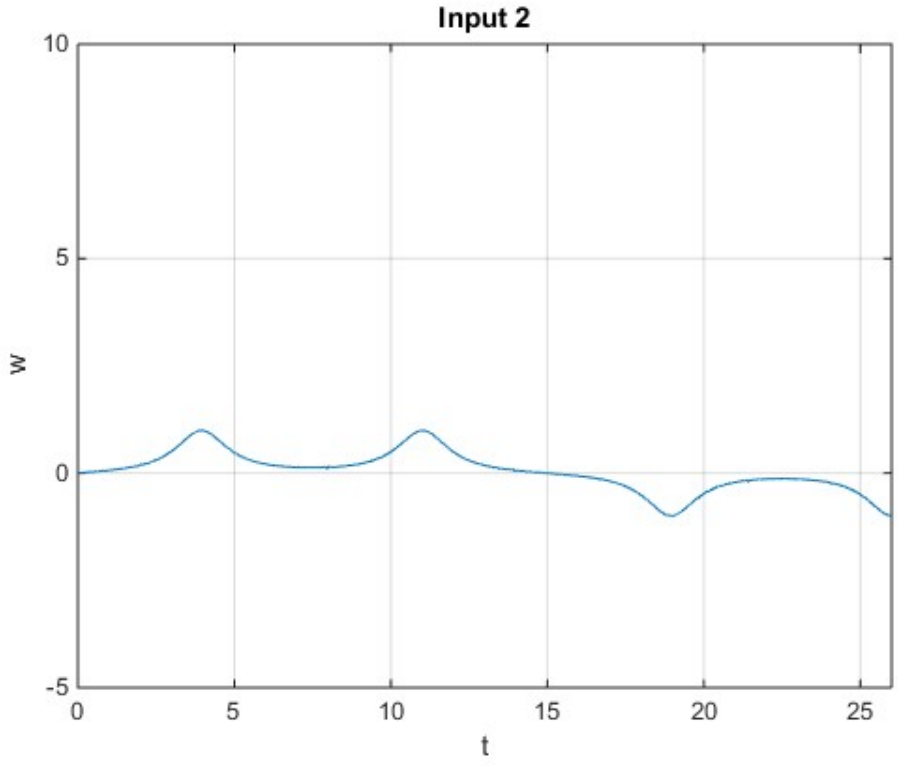


Figure IV.5. Entrée 2 du robot, générée par l'algorithme OPE (ω).

Il est à noter que, les résultats de simulation obtenus, ne sont pas déterministes puisque l'algorithme OPE, lors de sa recherche (optimisation) utilise des particules distribuées aléatoirement afin de trouver des meilleurs solutions (commandes optimales) à chaque période d'échantillonnage.

IV.5. Conclusion

Dans ce chapitre, nous avons parlé sur le problème de suivi d'une trajectoire de référence pour un robot mobile non-holonome en utilisant un algorithme OPE.

Les résultats de simulations montrent clairement que, cette méthode de commande permette d'atteindre les objectives envisagées relatives au problème de poursuite d'une trajectoire donnée.

Il est à noter que, ce type de synthèse est très sensible (n'est pas déterministe), est ceci est dû principalement au comportement aléatoire de l'algorithme OPE, plus l'obligation d'un choix soigneux de ses paramètres de contrôle.

Conclusion générale

Conclusion générale

Le travail présenté dans ce mémoire et porté sur le suivi de trajectoire d'un robot mobile non-holonome de type unicycle. En effet, basant sur le fait que le problème de synthèse d'une loi de commande peut être traiter comme un problème d'optimisation, nous avons travaillé à commander le robot en question (en simulation), en utilisant un algorithme de type méta-heuristique dit en anglais : Particle Swarm Optimization (PSO) c-à-d. Optimisation par Essaim Particulaire (OEP). L'intervention de ce dernier est basée sur la minimisation d'une fonction objectif définie d'une façon spécifique, qui permet d'atteindre les objectives envisagées.

Un des avantages principaux de cette méthode par rapport aux méthodes de commande conventionnelles est sa capacité de générer des commandes de qualité (commandes optimales) vu la puissance de l'algorithme OPE à résoudre ce genre de problèmes (problèmes d'optimisation de type combinatoire).

Afin de pouvoir examiner l'adaptation de cette méthode au problème de suivi d'un robot mobile, des essais de simulations sur MATLAB ont été réalisés. En effet, nous avons obtenu de bons résultats dans le suivi d'une trajectoire imposée.

Cependant, et comme nous avons signalé au chapitre quatre, les résultats obtenus, ne sont pas déterministes, puisque l'algorithme OPE est basé dans son processus de recherche sur un mécanisme sélectif déclenché au début d'une manière aléatoire.

Le travaille présenté dans ce mémoire, peut être complété par des études comparatives basées sur l'emploi d'autres algorithmes méta-heuristique, d'autres techniques de commandes conventionnelles et d'autre type de robots.

Annexe

Annexe

Code M-1

```
clc
clear all
close all

Ts = 0.033; % Sampling time
t = 0:Ts:30; % Simulation time
L_velocity=[];
A_velocity=[];

% Reference
freq = 2*pi/30;
xRef = 1.1 + 0.7*sin(freq*t); yRef = 0.9 + 0.7*sin(2*freq*t);
dxRef = freq*0.7*cos(freq*t); dyRef = 2*freq*0.7*cos(2*freq*t);
ddxRef =-freq^2*0.7*sin(freq*t); ddyRef =-4*freq^2*0.7*sin(2*freq*t);
qRef = [xRef; yRef; atan2(dyRef, dxRef)];
uRef = [ddxRef; ddyRef];

q = [xRef(1)+.05; yRef(1)-0.1; 0]; % Initial robot pose
z1 = [q(1); dxRef(1)]; % Initial state [x, x']
z2 = [q(2); dyRef(1)]; % Initial state [y, y']
v = sqrt(z1(2)^2+z2(2)^2); % Initial state of velocity integrator

x_q=[q(1)];
y_q=[q(2)];

% Matrices of linearized system
A = [0, 1; 0, 0]; B = [0; 1]; C = [1, 0];
% State feedback controller
desPoles = [-2-1i; -2+1i]; % Desired poles (of the controller)
K = acker(A, B, desPoles); % Control gains obtained by pole placement

for k = 1:length(t)
% Reference states
zRef1 = [xRef(k); dxRef(k)];
zRef2 = [yRef(k); dyRef(k)];

% Error and control
ez1 = zRef1 - z1;
ez2 = zRef2 - z2;
uu = [ddxRef(k); ddyRef(k)] + [K*ez1; K*ez2];

% Compute inputs to the robot
F = [cos(q(3)), -v*sin(q(3)); ...
sin(q(3)), v*cos(q(3))];
vv = F\uu; % Translational acceleration and angular velocity
v = v + Ts*vv(1); % Integrate translational acceleration
u = [v; vv(2)]; % Robot input

L_velocity=[v L_velocity];
A_velocity=[vv(2) A_velocity];
```

```
% Robot motion simulation
dq = [u(1)*cos(q(3)); u(1)*sin(q(3)); u(2)];
noise = 0.00; % Set to experiment with noise (e.g. 0.001)
q = q + Ts*dq + randn(3,1)*noise; % Euler integration
q(3) = wrapToPi(q(3)); % Map orientation angle to [-pi, pi]

x_q=[q(1) x_q];
y_q=[q(2) y_q];

% Take known (measured) orientation and velocity to compute states
z1 = [q(1); u(1)*cos(q(3))];
z2 = [q(2); u(1)*sin(q(3))];
end

figure (1)
plot(t,L_velocity)
axis([0 26  0 0.6]);
grid

title('Input 1')
xlabel('t')
ylabel('v')

figure (2)
plot(t,A_velocity)
axis([0 26  -5 10]);
grid

title('Input 2')
xlabel('t')
ylabel('w')


figure (3)
plot(xRef,yRef)
hold on
plot(x_q,y_q,'-. r')
grid

title('Positions x et y')
xlabel('x(m)')
ylabel('y(m)')
```

Code M-2

```
clc
clear all
close all

Ts = 0.033; % Sampling time
t = 0:Ts:30; % Simulation time

q = [1.1; 0.8; 0]; % Initial robot pose
x_q=[q(1)];
y_q=[q(2)];

L_velocity=[];
A_velocity=[];

% Reference
freq = 2*pi/30;
xRef = 1.1 + 0.7*sin(freq*t); yRef = 0.9 + 0.7*sin(2*freq*t);
dxRef = freq*0.7*cos(freq*t); dyRef = 2*freq*0.7*cos(2*freq*t);
ddxRef =-freq^2*0.7*sin(freq*t); ddyRef =-4*freq^2*0.7*sin(2*freq*t);
qRef = [xRef; yRef; atan2(dyRef, dxRef)];

vRef = sqrt(dxRef.^2+dyRef.^2);
wRef = (dxRef.*ddyRef-dyRef.*ddxRef)./(dxRef.^2+dyRef.^2);
uRef = [vRef; wRef]; % Reference inputs

for k = 1:length(t)

   e = [cos(q(3)), sin(q(3)), 0; ...
      -sin(q(3)), cos(q(3)), 0; ...
       0, 0, 1]*(qRef(:,k) - q); % Error vector

   e(3) = wrapToPi(e(3)); % Correct angle

   % Current reference inputs
    vRef = uRef(1,k);
    wRef = uRef(2,k);

   % Control
   eX = e(1); eY=e(2); ePhi=e(3);
   zeta = 0.9; % Experiment with this control design parameter
   g = 85; % Experiment with this control design parameter
   Kx = 2*zeta*sqrt(wRef^2+g*vRef^2);
   Kphi = Kx;
   Ky = g*vRef;
   % Gains can also be constant e.g.: Kx = Kphi = 3; Ky = 30;

   % Control: feedforward and feedback
   v = vRef*cos(e(3))+ Kx*e(1);
   w = wRef + Ky*e(2) + Kphi*e(3);

   L_velocity=[v L_velocity];
   A_velocity=[w A_velocity];
```

```
 % Robot motion simulation
    dq = [v*cos(q(3)); v*sin(q(3)); w];
    noise = 0.00; % Set to experiment with noise (e.g. 0.001)
    q = q + Ts*dq + randn(3,1)*noise; % Euler integration
    q(3) = wrapToPi(q(3)); % Map orientation angle to [-pi, pi]

    x_q=[q(1) x_q];
    y_q=[q(2) y_q];

end

figure (1)
plot(t,L_velocity)
axis([0 26  0 1]);
grid

title('Input 1')
xlabel('t')
ylabel('v')

figure (2)
plot(t,A_velocity)
axis([0 26  -5 10]);
grid

title('Input 2')
xlabel('t')
ylabel('w')

figure (3)
plot(xRef,yRef)
hold on
plot(x_q,y_q,'-. r')
grid

title('Positions x et y')
xlabel('x(m)')
ylabel('y(m)')
```

Code M-3

```
clc
clear all
close all

Ts = 0.033; % Sampling time
t = 0:Ts:30; % Simulation time

q = [1.1; 0.8; 0]; % Initial robot pose
x_q=[q(1)];
y_q=[q(2)];

L_velocity=[];
A_velocity=[];

% Reference
freq = 2*pi/30;
xRef = 1.1 + 0.7*sin(freq*t); yRef = 0.9 + 0.7*sin(2*freq*t);
dxRef = freq*0.7*cos(freq*t); dyRef = 2*freq*0.7*cos(2*freq*t);
ddxRef =-freq^2*0.7*sin(freq*t); ddyRef =-4*freq^2*0.7*sin(2*freq*t);
qRef = [xRef; yRef; atan2(dyRef, dxRef)];
vRef = sqrt(dxRef.^2+dyRef.^2);
wRef = (dxRef.*ddyRef-dyRef.*ddxRef)./(dxRef.^2+dyRef.^2);
uRef = [vRef; wRef]; % Reference inputs

for k = 1:length(t)-4

   e = [cos(q(3)), sin(q(3)), 0; ...
      -sin(q(3)), cos(q(3)), 0; ...
       0, 0, 1]*(qRef(:,k) - q); % Error vector

   e(3) = wrapToPi(e(3)); % Correct angle
   A0 = [1, Ts*uRef(2,k), 0;-Ts*uRef(2,k), 1, Ts*uRef(1,k); 0,0,1];
   A1 = [1, Ts*uRef(2,k+1), 0;-Ts*uRef(2,k+1), 1, Ts*uRef(1,k+1); 0,0,1];
   A2 = [1, Ts*uRef(2,k+2), 0;-Ts*uRef(2,k+2), 1, Ts*uRef(1,k+2); 0,0,1];
   A3 = [1, Ts*uRef(2,k+3), 0;-Ts*uRef(2,k+3), 1, Ts*uRef(1,k+3); 0,0,1];
   A4 = [1, Ts*uRef(2,k+4), 0;-Ts*uRef(2,k+4), 1, Ts*uRef(1,k+4); 0,0,1];
   B = [Ts, 0; 0, 0; 0, Ts];
   C = eye(3);
   Z = zeros(3,2);
   Hm = [C*A0*B, Z, Z, Z; ...
        C*A0*A1*B, C*A0*B, Z, Z; ...
        C*A0*A1*A2*B, C*A0*A1*B, C*A0*B, Z; ...
        C*A0*A1*A2*A3*B, C*A0*A1*A2*B, C*A0*A1*B, C*A0*B];

   Fm = [C*A0*A1, C*A0*A1*A2, C*A0*A1*A2*A3, C*A0*A1*A2*A3*A4].';

   ar = 0.65;
   Ar = eye(3)*ar; % Reference error dynamics
   H = 0;
   Fr = [Ar^(H+1), Ar^(H+2), Ar^(H+3), Ar^(H+4)].';

   % Weight matrices
   Qt = diag(repmat([1; 40; 0.1], 4, 1));
   Rt = diag(repmat([0.001; 0.001], 4, 1));
```

```
    % Optimal control calculation
    KKgpc = (Hm.'*Qt*Hm + Rt)\(Hm.'*Qt*(-Fm));
    KK = KKgpc(1:2,:); % Take current control gains

    v = -KK*e;
    uF = [uRef(1,k)*cos(e(3)); uRef(2,k)];
    u = v + uF;

    vMAX = 1; wMAX = 15; % Max velocities

    if abs(u(1))>vMAX, u(1) = sign(u(1))*vMAX; end
    if abs(u(2))>wMAX, u(2) = sign(u(2))*wMAX; end

    L_velocity=[u(1) L_velocity];
    A_velocity=[u(2) A_velocity];

    % Robot motion simulation
    dq = [u(1)*cos(q(3)); u(1)*sin(q(3)); u(2)];
    noise = 0.00; % Set to experiment with noise (e.g. 0.001)
    q = q + Ts*dq + randn(3,1)*noise; % Euler integration
    q(3) = wrapToPi(q(3)); % Map orientation angle to [-pi, pi]

    x_q=[q(1) x_q];
    y_q=[q(2) y_q];

end

figure (1)
plot(t(1:size(t,2)-4),L_velocity)
axis([0 26  0 1]);
grid

title('Input 1')
xlabel('t')
ylabel('v')

figure (2)
plot(t(1:size(t,2)-4),A_velocity)
axis([0 26  -5 10]);
grid

title('Input 2')
xlabel('t')
ylabel('w')

figure (3)
plot(xRef,yRef)
hold on
plot(x_q,y_q,'-. r')
grid

title('Positions x et y')
xlabel('x(m)')
ylabel('y(m)')
```

Code M-4 (Principal)

```
clc
clear all
close all

Ts = 0.033; % Sampling time
t = 0:Ts:30; % Simulation time

q = [1.1; 0.8; 0]; % Initial robot pose
x_q=[q(1)];
y_q=[q(2)];

L_velocity=[];
A_velocity=[];

% Reference
freq = 2*pi/30;

%xRef = 1.1 + 0.7*sin(freq*t); yRef = 0.9 + 0.7*sin(2*freq*t);
%dxRef = freq*0.7*cos(freq*t); dyRef = 2*freq*0.7*cos(2*freq*t);
%ddxRef =-freq^2*0.7*sin(freq*t); ddyRef =-4*freq^2*0.7*sin(2*freq*t);

xRef = 2.7 + 0.25*sin(freq*t); yRef = 1.9 + 0.2*sin(2*freq*t);
dxRef = freq*0.25*cos(freq*t); dyRef = 2*freq*0.2*cos(2*freq*t);
ddxRef =-freq^2*0.25*sin(freq*t); ddyRef =-4*freq^2*0.2*sin(2*freq*t);

qRef = [xRef; yRef; atan2(dyRef, dxRef)];

vRef = sqrt(dxRef.^2+dyRef.^2);
wRef = (dxRef.*ddyRef-dyRef.*ddxRef)./(dxRef.^2+dyRef.^2);
uRef = [vRef; wRef]; % Reference inputs

vMax = 1; wMax = 15; % Velocity constraints

% Swarm initialization
iterations = 20; % Number of iterations
omega = 0.5*0.5; % Inertia
c1 = 0.5*1; % Correction factor
c2 = 0.5*1; % Global correction factor
N = 25; % Swarm size
swarm = zeros([2,N,4]);
uBest = [0; 0];

for k = 1:length(t)-1
   % Initial swarm position
   swarm(:,:,1) = repmat(uBest, 1, N) + diag([0.1; 3])*randn(2,N);
   swarm(:,:,2) = 0; % Initial particle velocity
   swarm(1,:,4) = 1000; % Best value so far

   for iter = 1:iterations % PSO iteratively find best solution
      % Evaluate particles parameters
      for i = 1:N
         % Compute new predicted pose of the robot using i-th particle
         % parameters (input velocities) and compare obtained predicted
         % pose to the reference pose.
         vwi = swarm(:,i,1);
```

```
        ui = vwi + uRef(:,k); % Feedback and feedforward
        qk = q; % Current robot pose

        % Predict robot pose using particle parameters (velocities)

        qk = qk + Ts*[cos(qk(3)), 0; sin(qk(3)), 0; 0, 1]*ui;
        qk(3) = wrapToPi(qk(3)); % Correct angle range

        e = [cos(qk(3)), sin(qk(3)), 0; ...
           -sin(qk(3)), cos(qk(3)), 0; ...
           0, 0, 1]*(qRef(:,k+1)-qk); % Error

        e(3) = wrapToPi(e(3)); % Correct angle range

        Qt = diag([4; 80; 0.1]); Rt = diag([1; 1]*0.0001); % Weights

        J = e.'*Qt*e + vwi.'*Rt*vwi; % Cost function

        if J<swarm(1,i,4) % if new parameter is better, update:
            swarm(:,i,3) = swarm(:,i,1); % param values (v and w)
            swarm(1,i,4) = J; % and best criteria value.
        end

    end

    [~, gBest] = min(swarm(1,:,4)); % Global best particle parameters

    % Updating parameters by velocity vectors

    a = omega*swarm(:,:,2) + ...
    c1*rand(2,N).*(swarm(:,:,3) - swarm(:,:,1)) + ...
    c2*rand(2,N).*(repmat(swarm(:,gBest,3), 1, N) - swarm(:,:,1));

% Max param increment, acceleration: aMax=3 ==> 3*Ts=0.1

    a(1,a(1,:)>0.1) = 0.1; a(1,a(1,:)<-0.1) = -0.1;

% Max param increment, angular acceleration: aMax=60 ==> 60*Ts=2

    a(2,a(1,:)>2) = 2; a(2,a(1,:)<-2) = -2;

    v = swarm(:,:,1) + a; % Update velocity

% Limit velocity to preserve curvature ...
    [m, ii] = max([v(1,:)/vMax; v(2,:)/wMax; ones(1,N)]);
    i = ii==1; v(1,i) = sign(v(1,i))*vMax;
    v(2,i) = v(2,i)./m(i);
    i = ii==2; v(2,i) = sign(v(2,i))*wMax;
    v(1,i) = v(1,i)./m(i);
    swarm(:,:,2) = a; % Updated particle velocities (acceleration)
    swarm(:,:,1) = v; % Updated particle positions (velocities)

end

% Take the best particle to get robot command velocities
uBest = swarm(:,gBest,1);
u = uBest + uRef(:,k); % Feedback and feedforward
```

```
    % Velocity constraints
    if abs(u(1))>vMax, u(1) = sign(u(1))*vMax; end
    if abs(u(2))>wMax, u(2) = sign(u(2))*wMax; end

    L_velocity=[u(1) L_velocity];
    A_velocity=[u(2) A_velocity];

    % Robot motion simulation
    dq = [u(1)*cos(q(3)); u(1)*sin(q(3)); u(2)];
    noise = 0.00; % Set to experiment with noise (e.g. 0.001)
    q = q + Ts*dq + randn(3,1)*noise; % Euler integration
    q(3) = wrapToPi(q(3)); % Map orientation angle to [-pi, pi]
    x_q=[q(1) x_q];
    y_q=[q(2) y_q];

end


figure (1)
plot(t(1:size(t,2)-1),L_velocity)
axis([0 26  0 1]);
grid

title('Input 1')
xlabel('t')
ylabel('v')

figure (2)
plot(t(1:size(t,2)-1),A_velocity)
axis([0 26  -5 10]);
grid

title('Input 2')
xlabel('t')
ylabel('w')

figure (3)
plot(xRef,yRef)
hold on
plot(x_q,y_q,'-. r')
grid

title('Positions x et y')
xlabel('x(m)')
ylabel('y(m)')
```

Références bibliographiques

Références Bibliographiques

[1] Siegwart, R., Nourbakhsh, I.: Introduction to Autonomous Mobile Robots. MIT Press (2004).

[2] G. Klancar, A. Zdešar, S. Blaži ˇ c, and I. Škrjanc, ˇ Wheeled mobile robotics - From fundamentals towards autonomous systems. Elsevier, Butterworth-Heinemann, 2017.

[3] B.Bayle « Robotique Mobile » , chapter lessons, école nationale superieur de physiquede strasbourg 2010-2011.

[4] S. Lens. *Locomotion d'un robot mobile*. Thèse de doctorat, Faculté des Sciences Appliquées, Université de Liege, 2008.

[5] H. Moravec, *Stanford card and the CMU rover,* proceeding of IEEE vol 71, N° 7, pp 872-884, juillet 1983.

[6] Pierre Arnaud, *Des moutons et des robots : architecture de contrôle réactive et déplacements collectifs de robots*, PPUR presses polytechnique, 2000.

[7] L. Steels. *When are robots inteligent autonomous agents?* journal of robotics, Volume 15 pp 3-9, 1995.

[8] F. Sempe, *Auto-organisation d'une collectivité de robots : Application à l'activité de* patrouille en présence de perturbations. Thèse de Doctorat de l'Université Pierre et Marie Curie, Paris 6, 2004.

[9] D. Bonnafous, *Exécution réactive de trajectoires pour robots mobiles non-holonomes*. Thèse de doctorat. Institut National Polytechnique de Toulouse, Systèmes Automatiques. France, 2003.

[10] Éric Beaudry, *Planification de tâches pour un robot mobile autonome*, Thèse de Maître de sciences, Université de Sherbrooke, Québec, Canada. Août 2006.

[11] Bernard BAYLE, *Robotique mobile*, Thèse de doctorat, Ecole Nationale Supérieure de Physique de Strasbourg, Université de Strasbourg 2008.

[12] R.C.Arkin, *Formation Behaviors, behavior-Base robotics*, The MIT Press. Cambridge, MA, USA, 1998.

[13] L. Adouane, *Architecture de contrôle comportementale etréactive pour la coopération d'un groupe de robots mobiles.* Thèse de doctorat, Laboratoire d'automatique de Besançon UMR (CNRS 6589) France, Avril 2005.

[14] J.-Y. Fourquet et M. Renaud. Coordinated Control of a Non-Holonomic Mobile Manipulator. In ISER'1999, pages 115–125, Sydney, Australie, mars 1999.

[15] G. Campion, G. Bastin et B. D'Andrea-Novel. ´ Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pages 47–62, 1996.

[16] J. Neimark et N. Fufaev. Dynamics of nonholonomic systems, volume 33. Translations of Mathematical Monographs, 1972.

[17] F. W. Warner. Foundations of differentiable manifolds and lie groups. Graduate Texts in Mathematics. Springer-Verlag, 1983.

[18] H. Nijmeijer et A. J. Van der Shaft. Nonlinear dynamical control systems. Springer Verlag, New York, 1990.

[19] G. Giralt, R. Chatila et M. Vaisset. An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots. In First International Symposium on Robotics Research, pages 191–214, Cambridge, 1984.

[20] ActivMedia. P3-DX: World's Most Popular Intelligent Wheeled Robot, 2004. http://www.activrobots.com.

[21] K-team. Khepera II user manual, 2002. http ://www.k-team.com.

[22] P. Morin et C. Samson. Introduction à la commande par retour d'etat ´ des systemes non-holon ` omes. Ecole des Mines de Paris, 2003.

[23] J.-P. Laumond, editeur. La robotique mobile. Hermes Sciences, 2001.

[24] R.W. Brockett, Asymptotic stability and feedback stabilization, in: R.S. Millman, H.J. Sussmann (Eds.), Differential Geometric Control Theory, Birkhuser, Boston, MA, 1983, pp. 181–191

[25] Y. Kanayama, A. Nilipour, C.A. Lelm, A locomotion control method for autonomous vehicles, in: Proceedings of the 1988 IEEE International Conference on Robotics and Automation, 1988, vol. 2, April, 1988, pp. 1315–1317.

[26] 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, IEEE, Cincinnati, OH, 1990, pp. 384–389.

[27] A. De Luca, G. Oriolo, M. Vendittelli, Control of wheeled mobile robots: an experimental overview, in: S. Nicosia, B. Siciliano, A. Bicchi, P. Valigi (Eds.), Ramsete, Lecture Notes in Control and Information Sciences, vol. 270, Springer, Berlin, 2001, pp. 181–226.
[28] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, IEEE Trans. Syst. Man. Cybern. SMC-15 (1985) 116–132.
[29] H.O. Wang, K. Tanaka, M.F. Griffin, An approach to fuzzy control of nonlinear systems: stability and design issues, IEEE Trans. Fuzzy Syst. 4 (1996) 14–23.
[30] A. Ollero, O. Amidi, Predictive path tracking of mobile robots. application to the CMU navlab, in: Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments, ICAR, vol. 91, 1991, pp. 1081–1086.
[31] J.E. Normey-Rico, J. Gómez-Ortega, E.F. Camacho, A Smith-predictor-based generalised predictive controller for mobile robot path-tracking, Control. Eng. Pract. 7(1999) 729–740.
[32] F. Kuhne, W.F. Lages, J.M.G. da Silva Jr, Model predictive control of a mobile robot using linearization, in: Proceedings of Mechatronics and Robotics, 2004, pp. 525–530.
[33] D. Gu, H. Hu, Neural predictive control for a car-like mobile robot, Rob. Autom. Syst. 39 (2) (2002) 73–86.
[34] G. Klancar, I. Škrjanc, Tracking-error model-based predictive control for mobile robots in real time, Rob. Autom. Syst. 55 (2007) 460–469.
[35] J. Kennedy and R. C. Eberhart. "Particle Swarm Optimization". In: Proceedings of the IEEE International Conference on Neural Networks IV, pp. 1942–1948, Perth, Australia, November 1995.
[36] D. Pham and D. Karaboga, Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. London, U.K.: Springer, 2012
[37] C. Blum, Theoretical and Practical Aspects of Ant Colony Optimization, Dissertations in Artificial Intelligence, Vol. 282, Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany, 2004.
[38] Y. F. Cai and S. X. Yang, "An improved PSO-based approach with dynamic parameter tuning for cooperative multi-robot target searching in complex unknown environments," International Journal of Control, vol. 86, no. 10, pp. 1720–1732, 2013.
[39] S. K. Pandey, S. R. Mohanty, N. Kishor, and J. P. S. Catalao, ˜ "Frequency regulation in hybrid power systems using particle swarm optimization and linear matrix inequalities based robust controller design," International Journal of Electrical Power & Energy Systems, vol. 63, pp. 887–900, 2014.
[40] G. Stimac, S. Braut, and R. ˇ Ziguli ˇ c, "Comparative analysis of ´ PSO algorithms for PID controller tuning," Chinese Journal of Mechanical Engineering, vol. 27, no. 5, pp. 928–936, 2014.
[41] N. Nedic, D. Prsic, L. Dubonjic, V. Stojanovic, and V. Djordjevic, "Optimal cascade hydraulic control for a parallel robot platform by PSO," International Journal of Advanced Manufacturing Technology, vol. 72, no. 5–8, pp. 1085–1098, 2014.
[42] W.-D. Chang and C.-Y. Chen, "PID controller design for MIMO processes using improved particle swarm optimization," Circuits, Systems, and Signal Processing, vol. 33, no. 5, pp. 1473– 1490, 2014.
[43] X. Xiang, R. Mutlu, G. Alici, and W. Li, "Control of conducting polymer actuators without physical feedback: simulated feedback control approach with particle swarm optimization," Smart Materials and Structures, vol. 23, no. 3, Article ID 035014, 2014.
[44] K. A. Danapalasingam, "Robust autonomous helicopter stabilizer tuned by particle swarm optimization," International Journal of Pattern Recognition and Artificial Intelligence, vol. 28, no. 1, Article ID 1459002, 2014.
[45] M. J. Mahmoodabadi, M. Taherkhorsandi, and A. Bagheri, "Optimal robust sliding mode tracking control of a biped robot based on ingenious multi-objective PSO," Neurocomputing, vol. 124, pp. 194–209, 2014.
[46] M. J. Mahmoodabadi, M. Taherkhorsandi, and A. Bagheri, "Optimal robust sliding mode tracking control of a biped robot based on ingenious multi-objective PSO," Neurocomputing, vol. 124, pp. 194–209, 2014.
[47] Y. Zhong, X. Huang, P. Meng, and F. Li, "PSO-RBF neural network PID control algorithm of electric gas pressure regulator," Abstract and Applied Analysis, vol. 2014, Article ID 731368, 7 pages, 2014
[48] J.-W. Perng, G.-Y. Chen, and S.-C. Hsieh, "Optimal pid controller design based on PSO-RBFNN for wind turbine systems," Energies, vol. 7, no. 1, pp. 191–209, 2014.