République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université Abderrahmane Mira
Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Automatique

Spécialité : Automatique et Informatique Industrielle

Thème

Utilisation des algorithmes bio_inspirés pour la planification de Trajectoire d'un robot mobile a deux roues

Préparé par :

- BELAITOUCHE Ilyas
- FAOUZI Kamel

Dirigé par :

Mr. TIGHZERT Lyes

Examiné par :

Mr Nait Mohand Nacim (P)

Mr Lehouche Hocine

Année universitaire : **2023/2024**

Remerciements

Nous tenons à exprimer nos sincères remerciements à toutes les personnes qui nous ont soutenus et accompagnés tout au long de notre parcours et de la rédaction de ce mémoire de fin d'études.

Tout d'abord, nous souhaitons exprimer notre gratitude envers notre encadrant, Monsieur **TIGHZERT Lyes**, pour son précieux encadrement, ses conseils éclairés et son soutien constant. Ses orientations expertes ont été d'une importance capitale dans l'aboutissement de ce travail.

Nous tenons également à remercier nos enseignants et membres du corps professoral, De l'université de Abderrahmane Mira, pour leur expertise académique et leurs conseils pertinents. Leurs cours et discussions ont contribué à élargir nos connaissances et à approfondir notre compréhension du sujet.

Nous tenons à remercier tous ceux qui ont participé à notre recherche en contribuant de près ou de loin à la collecte de données et à la validation de nos résultats. Leur collaboration et leur engagement ont été d'une grande valeur pour la réussite de cette étude.

Enfin, nous sommes profondément reconnaissants envers toutes les personnes qui ont contribué à la réalisation de ce mémoire. Leur soutien inestimable a été une source de motivation et d'inspiration tout au long de ce processus. Nous sommes fiers de pouvoir partager ce travail avec vous et nous sommes reconnaissants de votre impact positif dans notre vie universitaire.

Dédicaces

Je dédie ce travail

À mes parents, à mes frères

Nacir, Toufik, Larbi , Fouad

à ma petite sœur Chaima

et *à toute ma famille*

et à mes amis

Abderahim, Lounes, Fahem, Amine, Zaki

et à une personne qui mérite aussi Rosa

Ilyas

Dédicaces

Je dédie ce travail

À mes parents, à mes frères,

à ma sœur et *à toute*

ma famille et à

mes ami(e)s

Faouzi

Table Des Matières

Chapitre 2 : Suivi Le Trajectoire d'un Robot Mobile a deux roues

Chapitre 3 : Algorithme D'optimisation Base Sur TLBO

Chapitre 4 : Conception Materiel Et Logiciel Du Robot Mobile

Liste de Figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

La liste des Tableaux

Abréviation

LAAS	*Laboratoire d'Analyses d'Architecture des Systèmes a Toulouse (France).*
WMR	*Wheeled Mobile Robot (Robot Mobile à Roues)*
L	*le Lagrangien*
Ec	*Energie cinétique*
E_M	*Energie cinétique du chariot en mouvement*
E_{cm}	*Energie cinétique du pendule*
V_C	*La vitesse de centre de gravité du pendule*
MCD	*Modèle Cinématique Directe*
TLBO	*Teaching-Learning Based Optimization*
PID	*Régulateur Proportionnel Intégral et Dérivé*
MCC	*Mobile Coordinate Control*
DDMR	*Dynamic Modeling and Control of Mobile Robots*

Introduction Générale

La robotique mobile est un domaine en pleine expansion et aux multiples applications, de la logistique à la surveillance, en passant par l'exploration et l'assistance personnelle. Parmi les problèmes techniques importants, il est nécessaire de souligner la planification de la trajectoire, qui assurer des déplacements efficaces et sécurisés des robots et au niveau optimal. Ce mémoire se concentre sur l'utilisation des algorithmes d'optimisation basés sur l'enseignement-apprentissage (TLBO) pour la planification de trajectoire d'un robot mobile à deux roues.

Les robots mobiles à deux roues, souvent appelés robots différentiels, sont largement utilisés en raison de leur simplicité mécanique et de leur maniabilité. Cependant, la planification de trajectoire pour ces robots pose des défis complexes, notamment en termes de modélisation cinématique et de gestion des obstacles. Les algorithmes d'optimisation, tels que le TLBO, offrent des solutions prometteuses pour surmonter ces défis en trouvant des trajectoires optimales de manière efficace.

Ce mémoire vise à explorer et à démontrer l'efficacité des algorithmes TLBO dans la planification de trajectoire pour un robot mobile à deux roues. Les objectifs spécifiques sont les suivants :

- **Modélisation** : Développer un modèle cinématique précise du robot mobile à deux roues en utilisant une approche mathématique fondée sur un modèle existant
- **Planification de Trajectoire** : Mettre en œuvre des techniques de planification de trajectoire adaptées aux contraintes du robot.
- **Algorithme TLBO** : Analyser et adapter l'algorithme TLBO pour l'optimisation des trajectoires.
- **Validation Pratique** : Tester et valider les solutions proposées à l'aide de simulations et d'expérimentations matérielles.

Après cette introduction, le travail est organisé comme suit :

Dans le chapitre 1 : Dans Ce chapitre on se focalise sur la modélisation géométrique, cinématique et dynamique d'un robot mobile de type uni cycle. incluant les équations de mouvement et les contraintes physiques.

Et dans le chapitre 2 : Ce chapitre présente un aperçu détaillé sur la Méthode du suivi de trajectoire pour les robots mobiles à deux roues.

Dans la partie suivante, Ce chapitre détaille l'algorithme TLBO, son fonctionnement, et son adaptation pour la planification de trajectoire. Une analyse de son comportement dans le système est également présentée.

La quatrième partie sera consacrer aux expérimentations pratiques, incluant les simulations et les tests matériels, pour valider l'efficacité des solutions proposées.

Ce mémoire se concentre sur la combinaison de la modélisation cinématique, de la planification de trajectoire et de l'optimisation à l'aide de l'algorithme TLBO, dans le but d'apporter des contributions significatives à la robotique mobile. Nous espérons que les résultats obtenus mettront en lumière l'efficacité des algorithmes TLBO pour la planification de trajectoire, ouvrant ainsi la porte à des applications plus avancées et robustes dans le domaine des robots mobiles.

CHAPITRE I : Modélisation du robot mobile a deux roues

I. Introduction :

Le développement et le contrôle d'un système mécatronique, comme un robot mobile, nécessitent une compréhension approfondie et une représentation précise de ce système. En particulier, un " **modèle mathématique"** est essentiel pour décrire avec précision le comportement du robot.

Dans ce chapitre, nous explorons comment le comportement d'un robot peut être décrit à l'aide d'équations **mathématiques**. Pour cela, nous utilisons des concepts mathématiques qui nous permettent de voir le robot sous différents angles : sa position (à travers un modèle géométrique), sa vitesse (via un modèle cinématique) ou les forces qu'il exerce (en se basant sur un modèle dynamique**). C'est** ainsi que nous pouvons comprendre et prédire son mouvement et son interaction avec l'environnement.

II. Robot mobile unicycle (différentiel) :

II.1 Définition :

Un robot mobile unicycle est un système robotique qui se déplace en utilisant une seule roue motrice. Contrairement aux robots à quatre roues, il n'a qu'une seule roue pour se déplacer.

En ce qui concerne le robot mobile à deux roues, il fonctionne avec deux roues motrices, chacune étant alimentée par un moteur distinct. Le changement de direction du robot est réalisé en ajustant les vitesses des deux roues motrices. Ces robots sont souvent équipés de roues folles pour garantir leur stabilité lors des déplacements.

Ce type de robot est souvent utilisé pour des applications de navigation autonome, de surveillance, d'exploration et de recherche. Le schéma des robots de type unicycle est donné en Figure I.1 **[1]**.

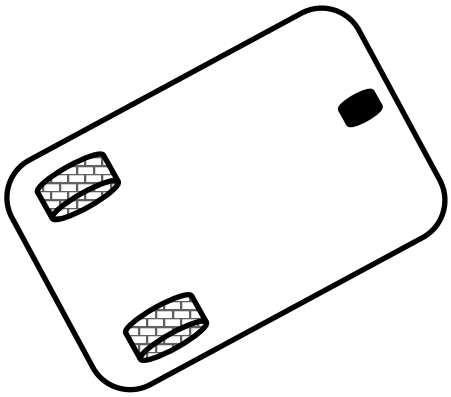


Figure I.1_Robot de type unicycle

II.2 Caractéristique de robot mobile unicycle :

Voici quelques caractéristiques importantes du robot mobile unicycle :

- **Configuration** : Le robot unicycle est actionné par une seule roue motrice. Il peut également posséder des roues folles (ou roulettes) pour assurer sa stabilité.
- **Centre de Rotation** : Le centre de rotation du robot est situé sur l'axe reliant la roue motrice.
- **Non-Holonome** : Le robot unicycle est un système non-holonome, ce qui signifie qu'il ne peut pas se déplacer latéralement (perpendiculairement à la direction de la roue motrice).
- **Commande** : Sa commande peut être relativement simple, ne dispose que de deux degrés de libertés sur un plan

  - Une translation : avance ou recule.
  - Une rotation : tourne vers la droite ou vers la gauche.

En résumé, le robot mobile unicycle offre une grande mobilité grâce à sa conception à une seule roue. Il occupe moins d'espace que les robots à plusieurs roues et peut être utilisé dans diverses applications **[2] [3]**.

II.3 Fonctionnement et mécanisme de déplacement :

Un robot mobile à deux roues utilise une conduite différentielle, avec deux roues motrices fixes, une de chaque côté de la plate-forme, chacune étant entraînée indépendamment. En contrôlant les vitesses de rotation des deux roues et notamment en jouant sur la différence de vitesse entre la roue droite et la roue gauche, il est possible de déplacer le robot dans différentes directions : vers l'avant, vers l'arrière, à droite, et à gauche. Il est même possible de réaliser un demi-tour sur place. Un schéma illustratif des mouvements du robot et des vitesses associées est donné à la Figure I.2 **[4]**.

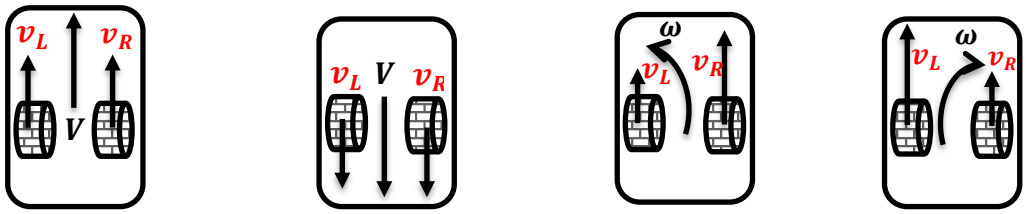


Figure I.2_Schéma de différentes déplacement possible avec Un robot mobile unicycle

Voici les mouvements possibles pour un robot mobile à deux roues avec un système de conduite différentielle:

- **Marche en avant,** où les deux roues tournent à **même vitesse** et dans la **même direction (avant)**, le robot avance en ligne droite.
- **Marche en arrière,** lorsqu'un robot roule sur ses roues à la **même vitesse** et dans le **même sens (arrière)**, il recule.
- **Tourner à droite**, où la roue **gauche** tourne **plus vite** que la roue droite et le robot effectue un virage vers la droite.
- **Tourner à gauche**, Indique que la roue **droite** tourne **plus vite** que la roue gauche et le robot effectue un virage vers la gauche.

II.3 Des exemples de robots mobile unicycle :

Le robot SHAKEY est considéré comme l'un des pionniers parmi les premiers robots mobiles autonomes, car il possède des capacités avancées en matière de perception, d'analyse et de planification. Équipé de deux roues motrices, il a été développé au Stanford Research Institute aux États-Unis en 1967 et est capable de se déplacer à une vitesse de 2 mètres par heure **[5]**.

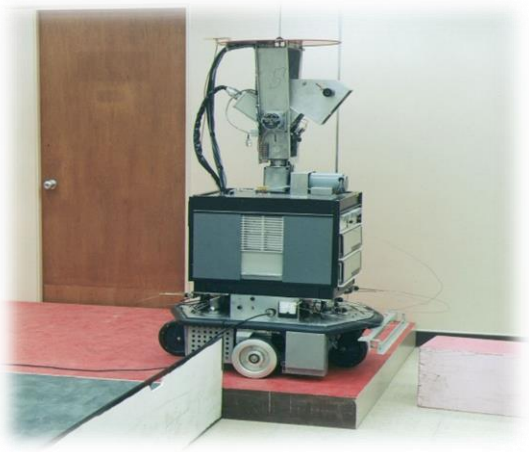


Figure I.3_ Le robot SHAKEY

Le robot Pioneer, développé au Stanford Research Institute (États-Unis, en 1993), est connu pour son coût abordable et sa gamme complète de capteurs internes et externes utilisés pour la planification de trajectoires. Cette information est illustrée dans la figure I.2 **[6]**.

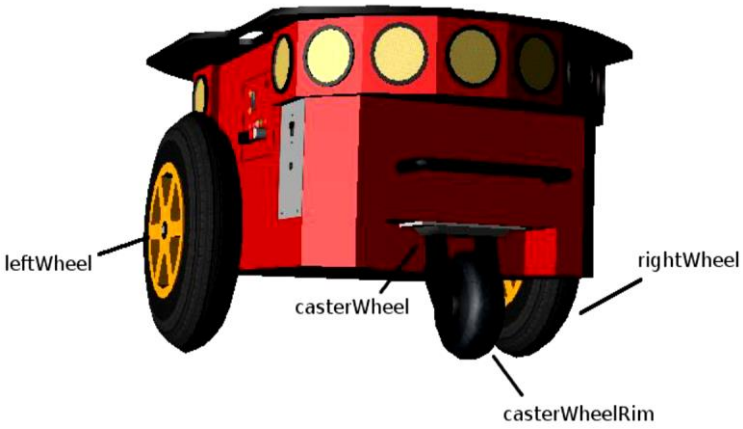


Figure I.4_Véhicule unicycle (Robot Pionner-3DX)

Le robot HILARE a été développé au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS) à Toulouse, en France, en 1979. Son objectif principal était de permettre la planification de trajectoires pour un robot mobile ponctuel **[3]**.

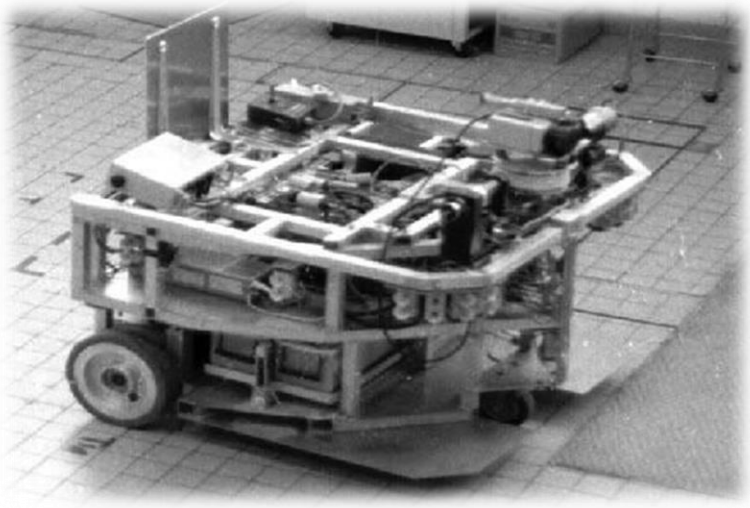


Figure I.5_Robot HILARE

III. Modélisation du robot unicycle [7] :

III.1 Systèmes de coordonnée :

Pour localiser le WMR dans son environnement, il faut établir deux repères de coordonnées différents :

- Repère universel (globale) : $\{ x_I, y_I \}$ est le système de coordonnées fixe dans le plan du robot.
- Repère du robot (local) : $\{ x_R, y_R \}$ est le système de coordonnées fixé sur le robot.

Les deux repères définis ci-dessus sont représentés sur la Figure I.6 Le point central A sur l'axe entre les roues est défini comme l'origine du repère du robot. Le centre de masse C du robot est situé sur l'axe de symétrie, à une distance d de l'origine A.

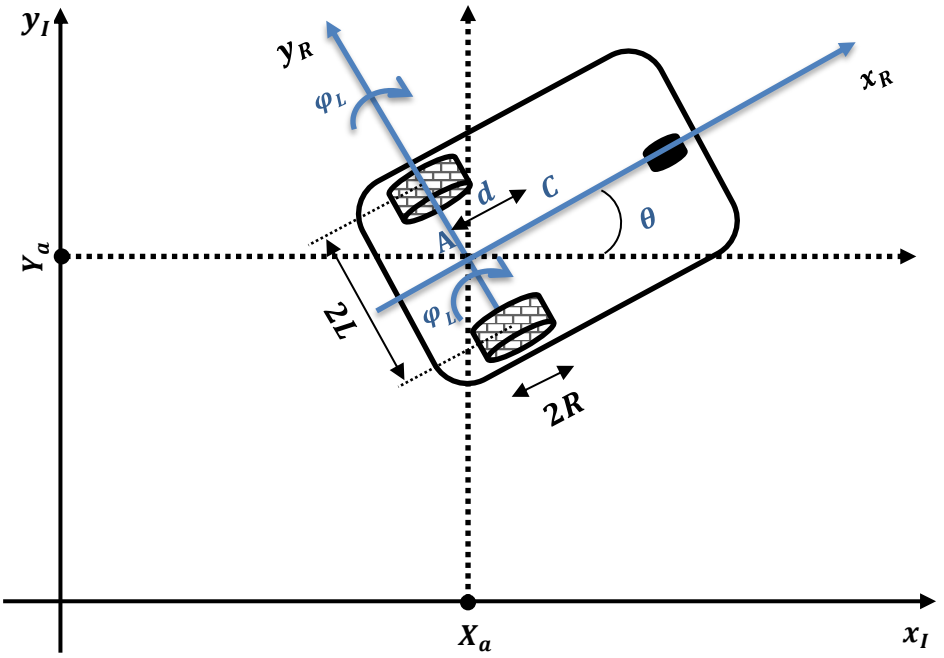


Figure I.6_Le robot mobile à entraînement différentiel dans la référence globale et locale

Avec :

$\boldsymbol{A}$: est le point milieu de l'axe des roues.

$\boldsymbol{2R}$: représente le diamètre de la roue du robot.

$\boldsymbol{2L}$: représente la largeur du robot.

$\boldsymbol{\varphi_R}$ *Et* $\boldsymbol{\varphi_L}$: représentent respectivement les vitesses de rotation de la roue droite et de la roue gauche.

$\boldsymbol{\theta}$: est l'angle d'orientation du robot.

$\boldsymbol{d}$: est la distance entre le point A et le point C (centre de masse).

Le robot est caractérisé par ses positions $\boldsymbol{X_a}$ et $\boldsymbol{Y_a}$, ainsi que son angle d'orientation θ, Comme indiqué dans la figure **I.6**, la position et l'orientation du robot par rapport au repère inertiel peuvent être exprimées par un vecteur $\boldsymbol{q}$ de ces trois éléments.

$$q = \begin{bmatrix} X_a \\ Y_a \\ \theta \end{bmatrix} \qquad (I.1)$$

À ce stade, il est essentiel d'expliquer le mappage entre les deux repères, celui du robot (local) et celui inertiel (global). La position d'un point quelconque du robot peut être définie dans ces deux repères de la manière suivante :

On utilise un repère fixe $\{x_I, y_I\}$ et un repère mobile $\{x_R, y_R\}$ attaché au robot pour décrire sa position et son orientation par rapport à l'environnement En prenant $X^I = \begin{bmatrix} x^I \\ y^I \\ \theta^I \end{bmatrix}$ comme un point de repère $\{x_I, y_I\}$ et $X^R = \begin{bmatrix} x^R \\ y^R \\ \theta^R \end{bmatrix}$ comme un point de repère $\{x_R, y_R\}$.

La relation entre ces deux repères est représentée par la transformation de rotation suivante :[11]

$$X^I = R(\theta)\, X^R \qquad (I.2)$$

Avec $R(\theta)$ est la matrice de rotation orthogonale :

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (I.3)$$

On verra dans la section suivante que l'équation (I.4) est très importante dans la dérivation des modèles cinématiques et dynamiques DDMR car elle décrit la relation entre les vitesses dans le cadre inertiel et le cadre du robot :

$$\dot{X}^I = R(\theta)\, \dot{X}^R \qquad (I.4)$$

III.2 Les contraintes de mouvement :

Le mouvement d'un robot mobile différentiel est caractérisé par deux équations de contrainte non-holonomes, qui sont obtenues par deux principales hypothèses (Dhaouadi R, 2013) :

Hypothèse I : Aucun glissement latéral. Cela signifie simplement que le robot ne peut se déplacer que dans un mouvement courbe (en avant et en arrière) mais pas latéralement. Dans le repère du robot (local), cette condition signifie que la vitesse du point central A est nulle le long de l'axe latéral, donc :

$$\dot{y}_a^R = 0 \qquad (I.5)$$

En utilisant la matrice de rotation R (θ) et d'après l'équation (I.2), les vitesses dans le repère mobile sont obtenues comme suit :

$$X^I = R\,(\theta)\,X^R$$

Donc

$$X_a^R = R^T\,(\theta)\,X_a^I \qquad (I.6)$$

$$\begin{bmatrix} x_a^R \\ y_a^R \\ \theta_a^R \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_a^I \\ y_a^I \\ \theta_a^I \end{bmatrix} \qquad (I.7)$$

$$\mathrm{y}_a^R = -sin(\theta)\mathrm{x}_a^{\mathrm{I}} + cos(\theta)\mathrm{y}_a^{\mathrm{I}}\,(\theta) \qquad (I.8a)$$

$$\frac{d}{dt}(\mathrm{y}_a^R) = -sin(\theta)\dot{\mathrm{x}}_a^{\mathrm{I}} + cos(\theta)\dot{\mathrm{y}}_a^{\mathrm{I}}\,(\theta) \qquad (I.8b)$$

$$\dot{\mathrm{y}}_a^{\mathrm{R}} = -sin(\theta)\dot{\mathrm{x}}_a^{\mathrm{I}} + cos(\theta)\dot{\mathrm{y}}_a^{\mathrm{I}}\,(\theta) \qquad (I.8c)$$

De Equation (1.5) et Eq. (1.8c) on trouve la première contrainte non-holonomie :

$$-sin(\theta)\dot{x}_a^I + cos(\theta)\dot{y}_a^I = 0 \qquad (I.9)$$

Hypothèse II : La contrainte de roulement pur (sans glissement) signifie que chaque roue du chariot maintient un point de contact fixe P avec la surface tout en se déplaçant, comme illustré dans la Figure I.7. Cela implique l'absence de glissement le long de l'axe longitudinal ($\boldsymbol{x_R}$) de la roue ainsi que de dérapage le long de son axe orthogonal ($\boldsymbol{y_R}$).

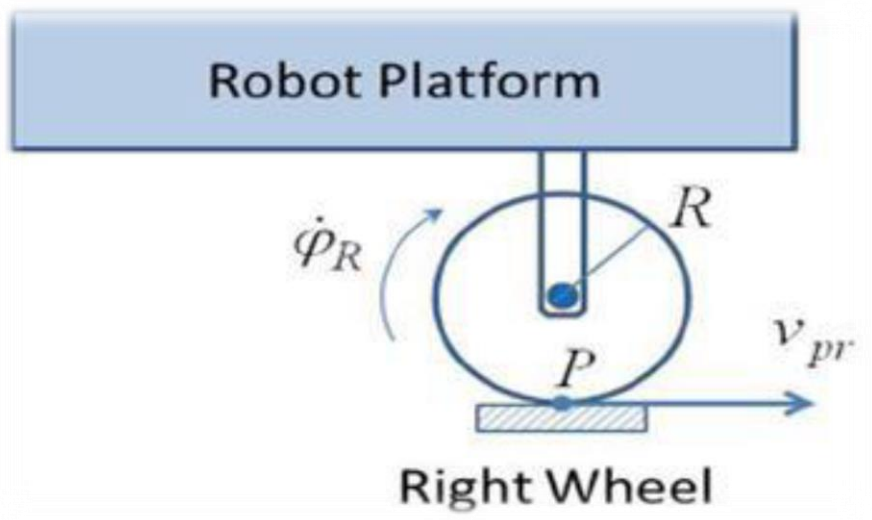


Figure I.7_Roulement sans glissement (La contrainte de roulement pur)

Les vitesses des points de contact dans le repère robot local sont liées aux vitesses des roues par :

$$\begin{cases} \boldsymbol{v_{pR}} = \mathrm{R}\ \boldsymbol{\dot{\varphi}_R} \\ \boldsymbol{v_{pL}} = \mathrm{R}\ \boldsymbol{\dot{\varphi}_L} \end{cases} \qquad \begin{matrix} (I.10a) \\ (I.10b) \end{matrix}$$

Où

$\boldsymbol{v_{pL}}$ est la vitesse linéaire de la roue gauche.
$\boldsymbol{v_{pR}}$ est la vitesse linéaire de la roue droite.

Dans le repère inertiel, ces vitesses peuvent être calculées en fonction des vitesses du point central $\boldsymbol{A}$ du robot, Suivant la figure I.7 on a :

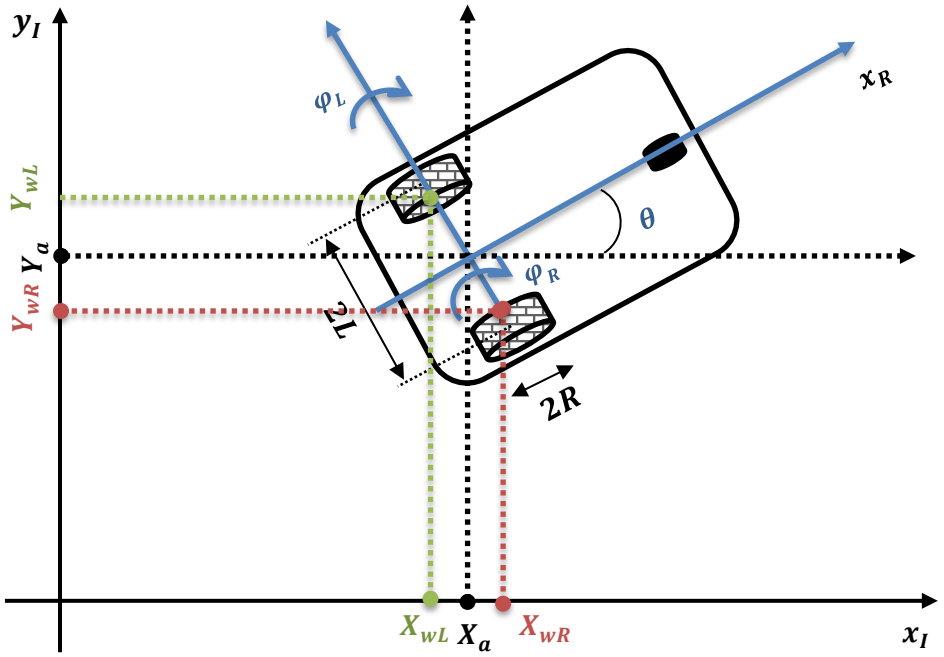


Figure I.8_Relation entre les coordonnées du point P et les point de contacts des roues

$$\text{Roue droite :} \begin{cases} {x^I}_{pR} = x_a^I + L\,sin(\theta) \\ {y^I}_{pR} = y_a^I - L\,cos(\theta) \end{cases} => \begin{cases} {\dot{x}^I}_{pR} = \dot{x}_a^I + L\dot{\theta}\,cos(\theta) \\ {\dot{y}^I}_{pR} = \dot{y}_a^I + L\dot{\theta}\,sin(\theta) \end{cases} \qquad (I.11a)$$

$$\text{Roue gauche :} \begin{cases} {x^I}_{pL} = x_a^I - L\,sin(\theta) \\ {y^I}_{pL} = y_a^I + L\,cos(\theta) \end{cases} => \begin{cases} {\dot{x}^I}_{pL} = \dot{x}_a^I - L\dot{\theta}\,cos(\theta) \\ {\dot{x}^I}_{pL} = \dot{y}_a^I - L\dot{\theta}\,sin(\theta) \end{cases} \qquad (I.11b)$$

En utilisant la matrice de rotation $R(\theta)$, on peut trouver les vitesses $\boldsymbol{v_{pR}}$ et $\boldsymbol{v_{pL}}$ dans le repère global :

$$\begin{bmatrix} \dot{x}_{pR}^I \\ \dot{y}_{pR}^I \\ \theta^I \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{pR}^R \\ \dot{y}_{pR}^R \\ \theta^R \end{bmatrix} \qquad (I.12)$$

Avec $\dot{y}_a^R = 0$ (car aucun glissement latéral). Ainsi :

$$\begin{bmatrix} \dot{x}_{pR}^I \\ \dot{y}_{pR}^I \\ \dot{\theta}_{pR}^I \end{bmatrix} = \begin{bmatrix} \dot{x}_{pR}^R \cos(\theta) \\ \dot{x}_{pR}^R \sin(\theta) \\ \dot{\theta}_{pR}^R \end{bmatrix} \qquad (I.13)$$

Où : $\vec{v}_{pR}^R = \dot{x}_{pR}^R \vec{\imath} + \dot{x}_{pR}^R \vec{\jmath} = R\,\dot{\varphi}_R \vec{\imath} \Rightarrow \boldsymbol{v_{pR}^R = \dot{x}_{pR}^R = R\,\dot{\varphi}_R}$

L'équation (I.13) devient :

$$\begin{cases} \dot{x}_{pR}^I \cos(\theta) = \dot{x}_{pR}^R \cos^2(\theta) \ldots . (b) \\ \dot{y}_{pR}^I \sin(\theta) = \dot{x}_{pR}^R \sin^2(\theta) \ldots . (a) \end{cases}$$

En additionnant les équations (a) et (b), on trouve la contrainte de roulement pour la roue droite :

$$\dot{x}_{pR}^I \cos(\theta) + \dot{y}_{pR}^I \sin(\theta) = R\,\dot{\varphi}_R \qquad (I.14a)$$

En suivant les mêmes étapes pour la roue gauche on trouve la contrainte de roulement pour la roue gauche :

$$\dot{x}_{pL}^I \cos(\theta) + \dot{y}_{pL}^I \sin(\theta) = R\,\dot{\varphi}_L \qquad (I.14b)$$

On peut former le système d'équation des deux roues :

$$\begin{cases} \dot{x}_{pR}^I \cos(\theta) + \dot{y}_{pR}^I \sin(\theta) = R\,\dot{\varphi}_R \\ \dot{x}_{pL}^I \cos(\theta) + \dot{y}_{pL}^I \sin(\theta) = R\,\dot{\varphi}_L \end{cases} \qquad (I.15)$$

Maintenant, en substituant (I.11a) et (I.11b) dans (I.15), on obtient :

$$\begin{cases} (\dot{x}_a^I + L\dot{\theta}\cos(\theta))\cos(\theta) + (\dot{y}_a^I + L\dot{\theta}\sin(\theta))\sin(\theta) = R\,\dot{\varphi}_R \\ (\dot{x}_a^I - L\dot{\theta}\cos(\theta))\cos(\theta) + (\dot{y}_a^I - L\dot{\theta}\sin(\theta))\sin(\theta) = R\,\dot{\varphi}_L \end{cases} \qquad (I.16)$$

Donc :

$$\begin{cases} \dot{x}_a^I cos(\theta) + \dot{y}_a^I sin(\theta) + d\dot{\theta} - R\dot{\varphi}_R = 0 \\ \dot{x}_a^I cos(\theta) + \dot{y}_a^I sin(\theta) - d\dot{\theta} - R\dot{\varphi}_L = 0 \\ -sin(\theta)\dot{x}_a^I + cos(\theta)\dot{y}_a^I = 0 \end{cases} \quad (I.17)$$

Les trois équations de contrainte peuvent être écrites dans la forme matricielle suivante :

$$\Lambda(q)\ \dot{q} = 0 \quad (I.18)$$

Où :

$A(q)$ est la matrice de contraintes non-holonomes donnée par :

$$\Lambda(q) = \begin{bmatrix} -sin(\theta) & cos(\theta) & 0 & 0 & 0 \\ cos(\theta) & sin(\theta) & L & -R & 0 \\ cos(\theta) & sin(\theta) & -L & 0 & -R \end{bmatrix} \quad (I.19)$$

Et $\dot{q}$ représente le dérivé de la coordonnée généralisée q :

$$\dot{q} = [\dot{x}_a, \dot{x}_a, \dot{\theta}, \dot{\varphi}_R, \dot{\varphi}_L]^T \quad (I.20)$$

Et

$$\begin{cases} \boldsymbol{v_R} = \mathrm{R}\ \boldsymbol{\dot{\varphi}_R} \\ \boldsymbol{v_L} = \mathrm{R}\ \boldsymbol{\dot{\varphi}_L} \end{cases} \quad (I.21)$$

III.3 Modèle cinématique :

La modélisation cinématiques concentre sur l'étude du mouvement des systèmes mécaniques sans prendre en compte les forces. Dans le cas d'un WMR (Robot Mobile à Roues), son objectif principal est de décrire les vitesses du robot en fonction des vitesses de ses roues motrices et les caractéristiques géométriques du robot.

La moyenne de la vitesse linéaire de chaque roue motrice dans le repère du robot est donc la vitesse linéaire du WMR dans le repère du robot.

III.3.1 Modèle cinématique direct [8] :

D'après ces deux hypothèses on peut alors déduire le modèle cinématique de notre robot : En utilise L'équation (I.10) qui nous permet d'obtenir l'expression des vitesses linéaires des roues droite et gauche au point de contact.

$$\begin{cases} v_{PR} = v_a + L\dot{\theta} \\ v_{PL} = v_a - L\dot{\theta} \end{cases} \quad (I.22)$$

Avec v_a la vitesse du point $\boldsymbol{A}$, v_{PR} est la vitesse de la roue droite au point $\boldsymbol{P}$ et v_{PL} est la vitesse de la roue gauche au point $\boldsymbol{P}$.

En posant :

$$\begin{cases} v_a = v \\ \dot{\theta} = \omega \end{cases} et \begin{cases} v_{PR} = v_R \\ v_{PL} = v_L \end{cases} \quad (I.23)$$

On obtient l'expression de la vitesse linéaire v et la vitesse angulaire ω du robot mobile en fonction des vitesses de rotation de la roue gauche $\dot{\varphi}_L$ et de la roue droite $\dot{\varphi}_R$.

$$\begin{cases} v = \dfrac{v_R + v_L}{2} \\ \omega = \dfrac{v_R - v_L}{2L} \end{cases} \quad (I.24)$$

Avec :

v: la vitesse linéaire du robot

ω: la vitesse angulaire du robot

Dans le repère mobile les coordonnées du point A sont :

$$\begin{cases} \dot{x}_A = v = \dfrac{R(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ \dot{y}_A = 0 \\ \dot{\theta} = \omega = \dfrac{R(\dot{\varphi}_R - \dot{\varphi}_L)}{2L} \end{cases} \quad (I.25)$$

En utilisant l'équation (I.2), il est possible d'écrire :

$$\begin{bmatrix} \dot{x}_A^I \\ \dot{y}_A^I \\ \dot{\theta}_A^I \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_A^R \\ \dot{y}_A^R \\ \dot{\theta}_A^R \end{bmatrix} \quad (I.26)$$

En remplaçant l'équation (I.25) dans (I.26), on obtient :

$$\begin{bmatrix} \dot{x}_A^I \\ \dot{y}_A^I \\ \dot{\theta}_A^I \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{R(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ 0 \\ \frac{R(\dot{\varphi}_R - \dot{\varphi}_L)}{2L} \end{bmatrix} \quad (I.27)$$

$$\dot{q}^I = \begin{bmatrix} \dot{x}_A^I \\ \dot{y}_A^I \\ \dot{\theta}_A^I \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos(\theta) & \frac{R}{2}\cos(\theta) \\ \frac{R}{2}\sin(\theta) & \frac{R}{2}\sin(\theta) \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \\ \dot{\varphi}_L \end{bmatrix} \quad (I.28)$$

L'équation **(I.28)** représente la cinématique directe différentielle d'un robot mobile à deux roues à entraînement différentiel.

A partir de ces relations nous pouvons construit notre modèle cinématique directe sous Simulink :

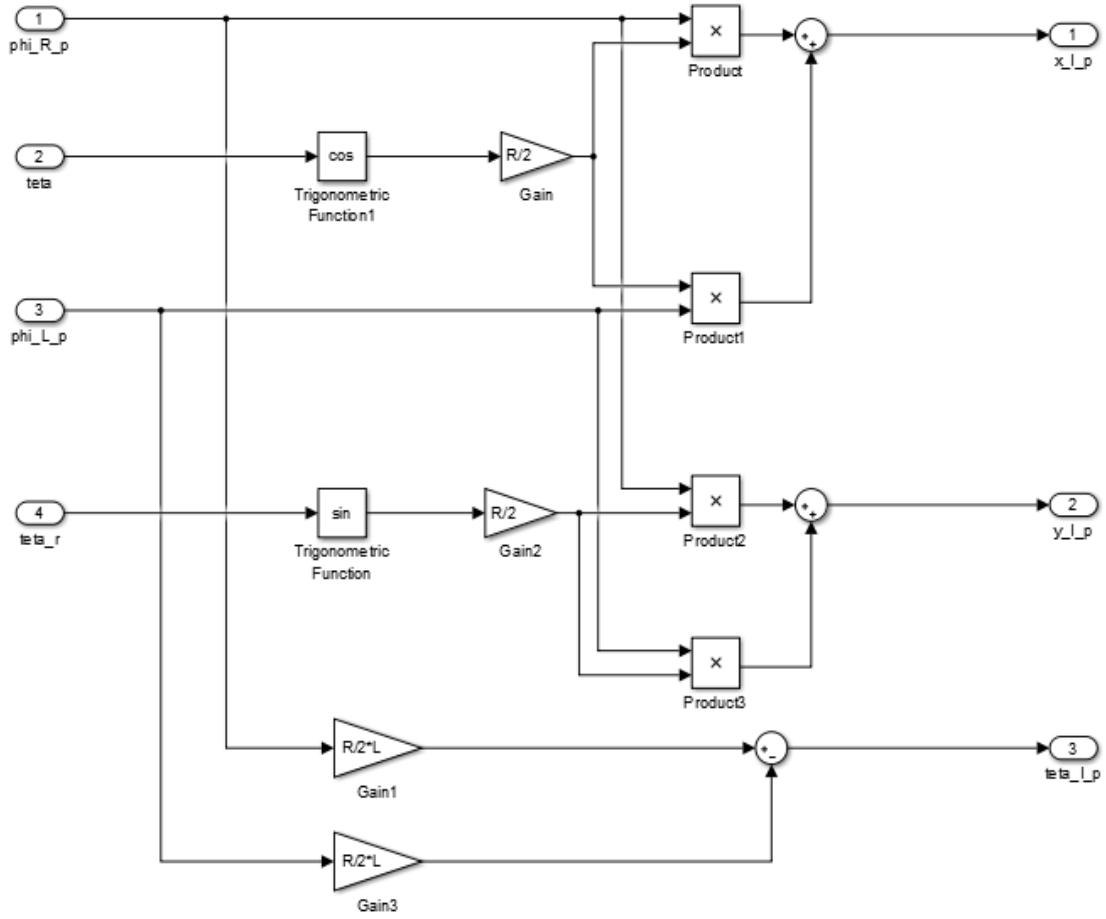


Figure I.9_ Modèle cinématique direct

Une autre forme alternative pour le modèle cinématique peut être obtenue en représentant les vitesses du WMR en termes de vitesses linéaires et angulaires du WMR dans le repère du robot.

$$\dot{q}^I = \begin{bmatrix} \dot{x}_A^I \\ \dot{y}_A^I \\ \dot{\theta}_A^I \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \qquad (I.29)$$

On obtient aussi un autre modèle cinématique du robot mobile unicycle :

$$\begin{cases} \dot{x}_A^I = v * cos\,(\theta) \\ \dot{y}_A^I = v * sin\,(\theta) \\ \dot{\theta}_A^I = \boldsymbol{\omega} \end{cases} \qquad (I.30)$$

A partir de ces équations nous pouvons **construire le** deuxième modèle cinématique directe (MCD) avec vitesses linéaires $\boldsymbol{v}$ et angulaires $\boldsymbol{\omega}$ sous Simulink/Matlab.

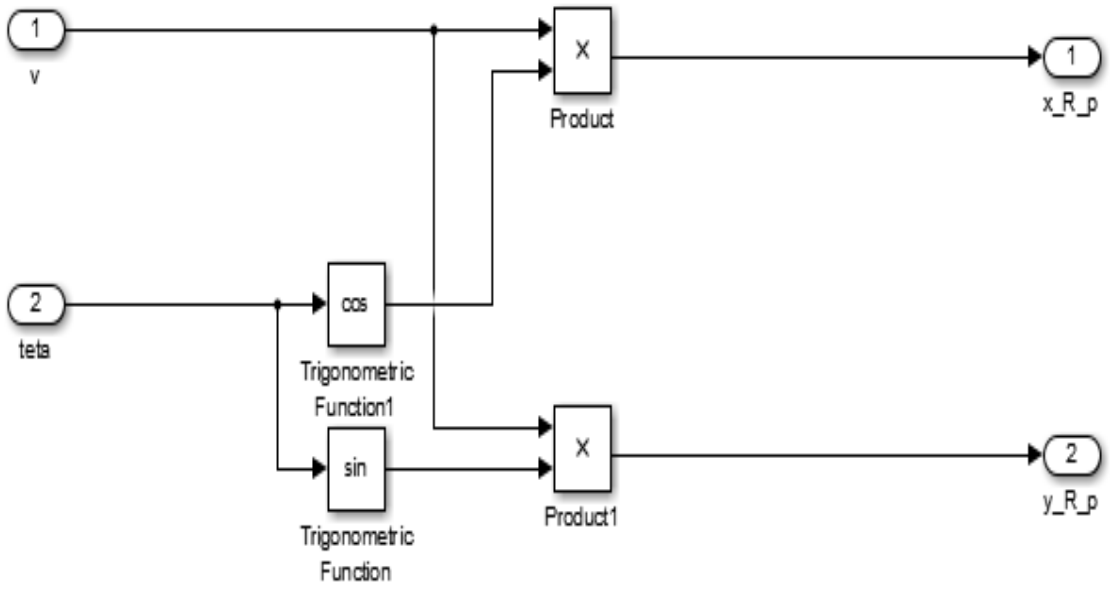


Figure I.10_ Modèle cinématique direct avec $\boldsymbol{v}$ et $\boldsymbol{\omega}$

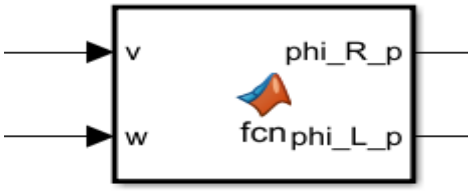
III.3.2 Modèle cinématique inverse [8] :

Le modèle cinématique inverse permet quant à lui, de calculer les vitesses de chaque roue à partir des vitesses opérationnelles V et θ. Les équations suivantes sont utilisées pour cela :

$$\dot{\varphi}_R = \frac{\mathrm{V} + \mathrm{L}\omega}{R} \qquad (I.31a)$$

$$\dot{\varphi}_L = \frac{\mathrm{V} - \mathrm{L}\omega}{R} \qquad (I.31b)$$

A partir de ces relations nous pouvons construit notre modèle cinématique inverse sous Simulink :




```
function [phi_R_p, phi_L_p] = fcn(v, w)
R = 0.1; % rayon de la roue en mètres
L = 0.2; % distance entre les roues en mètres


phi_R_p = (v + L*w)/(R);
phi_L_p= (v - L*w)/(R);
```

Figure I.11_Modèle cinématique inverse

III.4 Modèle dynamique :

Un modèle dynamique est essentiel pour décrire précisément le mouvement et pour l'élaboration d'algorithmes de contrôle des robots. Il y a un certain nombre de formalismes pour modéliser la dynamique du robot, les plus connus sont le formalisme d'Euler-Lagrange, le formalisme de Newton-Euler, ainsi que le principe de D'Alembert. Dans ce travail, nous utilisons le formalisme d'Euler-Lagrange.

Un robot non-holonome, soumis à des contraintes cinématiques, peut être décrit par l'équation de mouvement suivante :

$$\mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{V}(\mathrm{q}, \dot{\mathrm{q}})\,\dot{\mathrm{q}} + \mathrm{G}(\mathrm{q}) + \tau_d = \mathrm{B}(\mathrm{q})\tau - \mathrm{A}^{\mathrm{T}}(\mathrm{q})\lambda \qquad (I.32)$$

Avec :

M (q) : Matrice d'inertie du système "symétrique et définie positive de taille $R^{n\times n}$ "
V (q, $\dot{q}$) : la matrice des forces centrifuges et des forces de Coriolis
B(q) : la matrice de transformation d'entrée
A(q) : la matrice associée aux contraintes non-holonomes
λ : Vecteur des multiplicateurs de Lagrange
G(q) : est le vecteur gravitationnel

τ_d : le vecteur de limite perturbations inconnues incluant la dynamique non structurée
τ : est le vecteur d'entrée, $\tau = \begin{bmatrix}\tau_R \\ \tau_L\end{bmatrix}$

Dans cette étude, nous adoptons la méthode de Lagrange-Euler pour développer le modèle dynamique. Cette approche permet d'établir directement les relations entre les couples et les coordonnées généralisées du robot.

III.4.1 L'approche dynamique de Lagrange :

La méthode d'Euler-Lagrange est une approche analytique de la mécanique qui repose sur le principe de moindre action (minimisation de l'action). Elle utilise les énergies cinétique et potentielle d'un système pour dériver systématiquement les équations du mouvement.

Considérons un robot non-holonome avec n coordonnées généralisées (q_1,q_2,....., q_n) et soumis à m contraintes.

Le formalisme d'Euler-Lagrange est décrit par l'équation différentielle suivante :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F - A^T(q) * \lambda_k \quad i= 1,2...n \ , \ k=1,2...n-m \qquad (I.33)$$

Avec :

$L(q, \dot{q}) = T - V$: le Lagrangien

Tel que :

T : L'énergie cinétique du système ;

V : L'énergie potentielle du système ;

F : Le vecteur de force généralisée ;

λ_k : Le vecteur des multiplicateurs de Lagrange ;

q_i : la coordonnée généralisée et $q = [x_a, y_a, \theta, \varphi_R, \varphi_L]^T$, de dimension n =5.

L'énergie cinétique T du système est donnée par :

$$T=T_c+T_{wR}+T_{wL} \qquad (I.34)$$

Où

T_c Est l'énergie cinétique de la plateforme :

$$T_c = \frac{1}{2}m_c {v_c}^2 + \frac{1}{2}I_c \dot{\theta}^2 \qquad (I.35)$$

T_{wR} Est l'énergie cinétique de la roue droite :

$$T_{wR} = \frac{1}{2} m_w {v_{wR}}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_R^2 \qquad (I.36)$$

T_{wL} Est l'énergie cinétique de la roue gauche :

$$T_{wL} = \frac{1}{2} m_w {v_{wL}}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_L^2 \qquad (I.37)$$

Avec :

m_c : La masse de la plate-forme.
m_w: La masse de chaque roue plus la masse du moteur.
v_{wR}: La vitesse linéaire de la roue droite.
v_{wL} : La vitesse linéaire de la roue gauche.
I_m : Le moment d'inertie de chaque roue avec le moteur par rapport au diamètre de la roue.
I_w : Le moment d'inertie de chaque roue avec le moteur par rapport à l'axe de la roue.
I_c : Le moment d'inertie de la plate-forme du robot sans les roues motrices et les moteurs autour d'un axe vertical qui passe par le point C.

Nous commencerons par exprimer toutes les vitesses en fonction des coordonnées généralisées, en utilisant l'équation générale de la vitesse dans le référentiel inertiel.

$$v_i^2 = \dot{x}_i^2 + \dot{y}_i^2 \qquad (I.38)$$

Le point C dans le repère fixe a pour coordonnées :

$$\begin{cases} x_c = x_a + d\ cos(\theta) \\ y_c = y_a + d\ sin(\theta) \end{cases} => \begin{cases} \dot{x}_c = \dot{x}_a - d\dot{\theta}\ sin(\theta) \\ \dot{y}_c = \dot{y}_a + d\dot{\theta}\ cos(\theta) \end{cases} \qquad (I.39)$$

Et Les roues droite et gauche dans le repère fixe a pour coordonnées :

$$\begin{cases} x_{wR} = x_a + L\ sin(\theta) \\ y_{wR} = y_a - L\ cos(\theta) \end{cases} => \begin{cases} \dot{x}_{wR} = \dot{x}_a + L\dot{\theta}\ cos(\theta) \\ \dot{y}_{wR} = \dot{y}_a + L\dot{\theta}\ sin(\theta) \end{cases} \qquad (I.40a)$$

$$\begin{cases} x_{wL} = x_a - L\ sin(\theta) \\ y_{wL} = y_a + L\ cos(\theta) \end{cases} => \begin{cases} \dot{x}_{wL} = \dot{x}_a - L\dot{\theta}\ cos(\theta) \\ \dot{y}_{wL} = \dot{y}_a - L\dot{\theta}\ sin(\theta) \end{cases} \qquad (I.40b)$$

L'énergie cinétique totale est donnée par :

$$\mathrm{T} = \frac{1}{2}m_c {v_c}^2 + \frac{1}{2}I_c\dot{\theta}^2 T_{wR} + \frac{1}{2}m_w {v_{wR}}^2 + \frac{1}{2}I_m\dot{\theta}^2 + \frac{1}{2}I_w\dot{\varphi}_R^2 + \frac{1}{2}m_w {v_{wL}}^2 + \frac{1}{2}I_m\dot{\theta}^2 + \frac{1}{2}I_w\dot{\varphi}_L^2 \qquad (I.41)$$

D'après les équations *(I.38) et (I.40),* on obtient :

$$\mathrm{T} = \frac{1}{2}m_c(\dot{x}_c^2 + \dot{y}_c^2) + \frac{1}{2}m_w(v_{wR}^2 + v_{wL}^2) + \frac{1}{2}(I_c + 2I_m)\dot{\theta}^2 + \frac{1}{2}I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) \qquad (I.42)$$

$$\vdots$$

$$T = \frac{1}{2}(m_c + 2m_w)[\dot{x}_a^2 + \dot{y}_a^2] + \frac{1}{2}[m_c d^2 + 2m_w L^2 + I_c + 2I_m]\dot{\theta}^2 + \frac{1}{2}I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) - m_c d\dot{\theta}(\dot{x}_a\, sin(\theta) - \dot{y}_a\, cos(\theta))$$

On pose :

$$\begin{cases} m = m_c + 2m_w \\ I = m_c d^2 + 2m_w L^2 + I_c + 2I_m \end{cases} \qquad (I.43)$$

Donc

$$T = \frac{1}{2}m(\dot{x}_a^2 + \dot{y}_a^2) - m_c d\dot{\theta}\left(\dot{x}_a\, sin(\theta) - \dot{y}_a\, cos(\theta)\right) + \frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) \qquad (I.44)$$

L'énergie potentielle étant nulle car le robot se déplace sur un plan horizontal.

Alors on a donc le Lagrangien : $L=T$

En utilisant l'équation (I.44), l'équation (I.33) devient :

$$\begin{cases} m\,\ddot{x}_a - m_c\, d\,\ddot{\theta}\, sin(\theta) - m_c\, d\,\dot{\theta}^2\, cos(\theta) = C_1 \\ m\,\ddot{y}_a + m_c\, d\,\ddot{\theta}\, cos(\theta) - m_c\, d\,\dot{\theta}^2\, sin(\theta) = C_2 \\ I\ddot{\theta} - m_c\, d\,\ddot{x}_a sin(\theta) + m_c\, d\,\ddot{y}_a cos(\theta) = C_3 \\ I_w\ddot{\varphi}_R = \tau_R + C_4 \\ I_w\ddot{\varphi}_L = \tau_L + C_5 \end{cases} \qquad (I.45)$$

Avec : C_1 , C_2 , C_3 , C_4 et C_5 sont les coefficients relatifs aux contraintes cinématiques qui peuvent être exprimés en fonction du vecteur de multiplicateurs de Lagrange et de la matrice de contrainte cinématique $A^T(q)$, où :

$$A^T(\mathrm{q}) = [C_1\ C_2\ C_3\ C_4\ C_5\ ]^T \qquad (I.46)$$

Le modèle dynamique du robot mobile à roue peut se mettre sous la forme:

$$M(q)\ \ddot{q} + v\ (q, \dot{q})\ \dot{q} = B(q)\ \tau - A^T(q)\ \lambda \qquad (I.47)$$

Avec :

$$M(q) = \begin{pmatrix} m & 0 & -m_c\ d\ sin(\theta) & 0 & 0 \\ 0 & m & m_c\ d\ cos(\theta) & 0 & 0 \\ -m_c\ d\ sin(\theta) & m_c\ d\ cos(\theta) & I & 0 & 0 \\ 0 & 0 & 0 & I_W & 0 \\ 0 & 0 & 0 & 0 & I_W \end{pmatrix} \qquad (I.48)$$

$$\mathrm{v}(\mathrm{q}, \dot{\mathrm{q}}) = \begin{pmatrix} 0 & 0 & -m_c\ d\dot{\theta}\cos(\theta) & 0 & 0 \\ 0 & 0 & -m_c\ d\dot{\theta}\sin(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad B(q) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (I.49)$$

$$A^T(q)\lambda = \begin{pmatrix} -sin(\theta) & cos(\theta) & 0 & 0 & 0 \\ cos(\theta) & sin(\theta) & L & -R & 0 \\ cos(\theta) & sin(\theta) & -L & 0 & -R \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{pmatrix} \qquad (I.50)$$

Dans le but d'obtenir une expression plus pratique du modèle dynamique du robot, adaptée aux besoins de contrôle et de simulation, en éliminant le terme $A^T(q)\ \lambda$ qui correspond aux forces de contraintes liées aux contraintes cinématiques.

En définissant $\dot{\eta} = \begin{pmatrix} \dot{\varphi}\mathrm{R} \\ \dot{\varphi}\mathrm{L} \end{pmatrix}$ comme un vecteur de vitesses auxiliaires η et en utilisant le modèle cinématique précédent (I.28), nous avons :

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_a \\ \dot{\theta} \\ \dot{\varphi}\mathrm{R} \\ \dot{\varphi}\mathrm{L} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} R\cos(\theta) & R\cos(\theta) \\ R\,sin(\theta) & R\,sin(\theta) \\ \frac{R}{L} & -\frac{R}{L} \\ 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} \dot{\varphi}\mathrm{R} \\ \dot{\varphi}\mathrm{L} \end{bmatrix} \qquad (I.51)$$

On pose

$$\dot{q} = S(q)\eta \qquad (I.52)$$

Et

$$\ddot{q} = \dot{s}(q)\eta + s(q)\dot{\eta} \qquad (I.53)$$

S(q) est une matrice de rang complet qui satisfait à la condition suivante :

$$S^T(q)\, A^T(q) = 0 \qquad (I.54)$$

En remplaçant *(I.52)* et *(I.53)* dans l'équation *(I.47)*, on obtient :

$$M(q)[\dot{s}(q)\eta + s(q)\dot{\eta}] + v(q,\dot{q})[S(q)\eta] = B(q)\tau - A^T(q)\lambda \qquad (I.55)$$

En multipliant l'équation *(I.55)* par $S^T(q)$ on a :

$$\begin{aligned} \left(S^T(q)\, M(q)\, \dot{s}(q) + S^T(q)\, v(q,\dot{q})S(q)\right)\eta + S^T(q)\, M(q)\, s(q)\dot{\eta} \\ = S^T(q).B(q)\tau - S^T(q)\, A^T(q)\lambda \end{aligned} \qquad (I.56)$$

On définit maintenant les nouvelle matrices :

$$\begin{cases} \bar{M}(q) = S^T(q)\, M(q)\, s(q) \\ \bar{V}(q,\dot{q}) = S^T(q)\, M(q)\, \dot{s}(q) + S^T(q)\, v(q,\dot{q})S(q) \\ \bar{B}(q) = S^T(q).B(q) \end{cases} \qquad (I.57)$$

L'équation dynamique *(I.47)* est réduite à la forme :

$$\bar{M}(q)\dot{\eta} + \bar{V}(q,\dot{q})\,\eta = \bar{B}(q)\tau \qquad (I.58)$$

Où

$$\bar{M}(q) = \begin{bmatrix} \frac{R^2}{4L^2}(mL^2 + I) + I_w & \frac{R^2}{4L^2}(mL^2 - I) \\ \frac{R^2}{4L^2}(mL^2 - I) & \frac{R^2}{4L^2}(mL^2 + I) + I_w \end{bmatrix} \quad (I.59)$$

$$\bar{V}(q, \dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{2L} m_c d\dot{\theta} \\ -\frac{R^2}{2L} m_c d\dot{\theta} & 0 \end{bmatrix}, \quad \bar{B}(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (I.60)$$

Nous pouvons également transformer l'équation de mouvement *(I.58)* en une forme alternative qui est représentée par les vitesses linéaires et angulaire (v, ω) du DDMR.

En insérant les équations de modèle cinématique *(I.25)* dans *(I.58)* on obtient :

$$\begin{cases} \left(m + \frac{2I_w}{R^2}\right)\dot{v} - m_c d\, \omega^2 = \frac{1}{R}(\tau_R + \tau_L) \\ \left(m + \frac{2L^2}{R^2} I_w\right)\dot{w} + m_c d\, \omega v = \frac{L}{R}(\tau_R - \tau_L) \end{cases} \quad (I.61)$$

Ensuite, ces deux équations peuvent être écrites sous forme matricielle comme suit :

$$\begin{bmatrix} m + \frac{2I_w}{R^2} & 0 \\ 0 & I + \frac{2L^2}{R^2} I_w \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 & -m_c d\omega \\ m_c d\omega & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & 0 \\ 0 & \frac{L}{R} \end{bmatrix} \begin{bmatrix} \tau_R + \tau_L \\ \tau_R - \tau_L \end{bmatrix} \quad (I.62)$$

Où

(τ_R, τ_L) est le couple d'entrée exprimé en Newton mètres (N.m).
m : est la masse totale du robot
I : est l'inertie totale équivalente du système
R : est le rayon de la roue
2L : est la largeur du robot

En remplaçant $(m + \frac{2I_w}{R^2})$ par m_0 et $(I + \frac{2L^2}{R^2} I_w)$ par I_0 dans l'équation *(I.61)* On obtient :

$$\begin{bmatrix} M_0 & 0 \\ 0 & I_0 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 & -m_c d\omega \\ m_c d\omega & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & 0 \\ 0 & \frac{L}{R} \end{bmatrix} \begin{bmatrix} \tau_R + \tau_L \\ \tau_R - \tau_L \end{bmatrix} \quad (I.63)$$

A partir de ces relations nous pouvons construit notre modèle dynamique directe Sous Simulink :

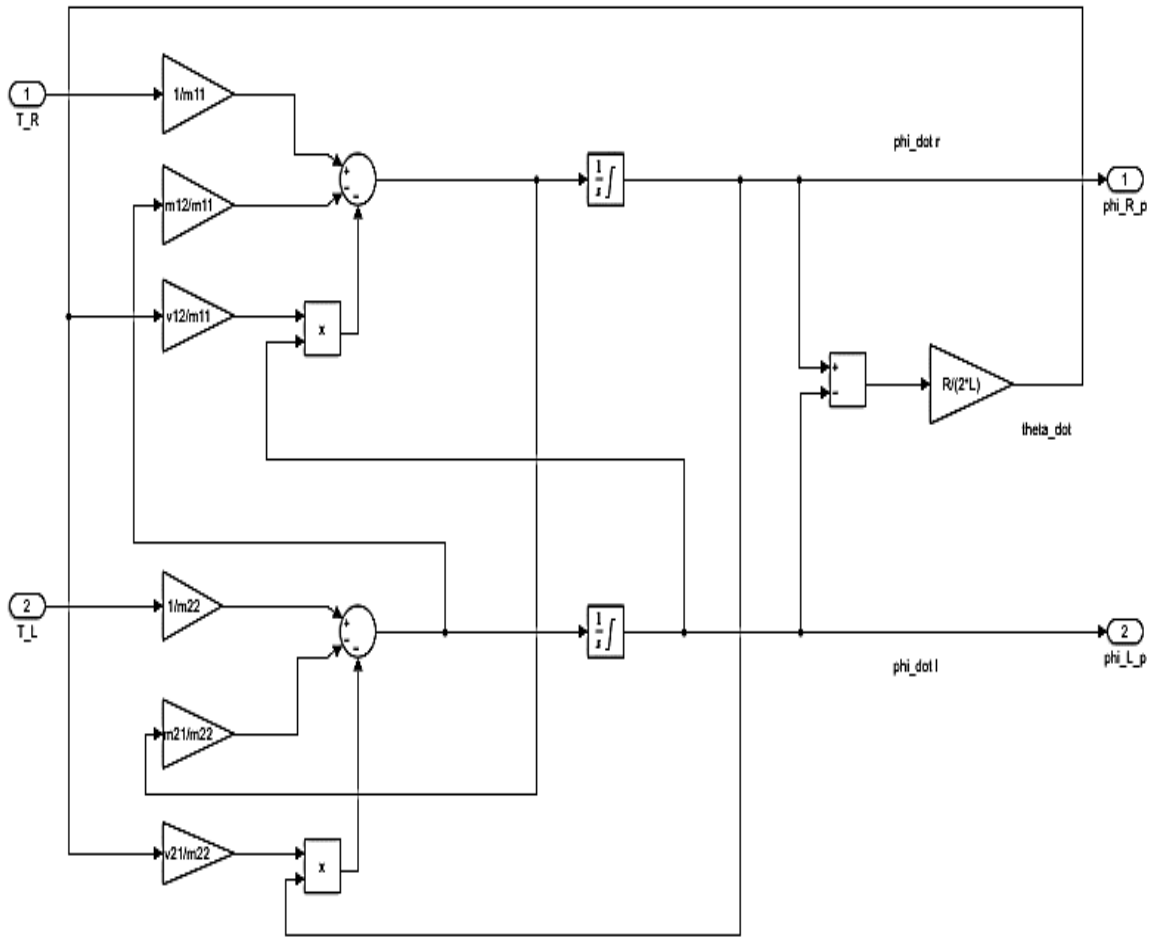


Figure I.12_Modèle dynamique direct

III.5 Modèle d'actionnement (comportement) :

III.5.1 Modélisation mathématique :

Nous utilisons des moteurs à courant continu pour propulser les roues de notre robot unicycle, considérées comme ses actionneurs. Les caractéristiques d'un moteur à courant continu sont :

$$\begin{cases} v_a(t) = R_a i_a(t) + L_a \dfrac{di_a(t)}{dt} + e_a(t) \\ e_a(t) = k_b w_m(t) \\ \tau_m = k_t i_a \\ \tau = N\tau_m \end{cases} \qquad (I.64)$$

Où

v_a : la tension est utilisée comme entrée du moteur[V]

i_a : est le courant d'induit[A].

R_a : résistance des armatures du moteur[Ω]

L_a : l'inductance des armatures du moteur[H]

e_a: la force électromotrice[V]

w_m: la vitesse angulaire du rotor[red/s]

τ_m : le couple du moteur[H]

k_t : constante de couple moteur [N.m/A]

k_b : constante de la force électromotrice [V.s/red]

N : le rapport de réduction

Transformée de Laplace :

Pour construire le modèle causal du MCC, on applique d'abord la transformée de Laplace aux équations du MCC.

$$(1) : v_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_a(t) \qquad (I.65)$$

La transformée de Laplace :

$$V = R_a I + S\,L_a\,I + E \qquad (I.66a)$$

$$V - E = (R_a + S\,L_a)\,I \qquad (I.66b)$$

Donc

$$\frac{1}{R_a + S\,L_a}(V - E) = I \qquad (I.67)$$

$$(2)\ F\ (C_m) = K_t\ I \qquad (I.68)$$

III.5.2 Le modèle Simulink d'actionnement :

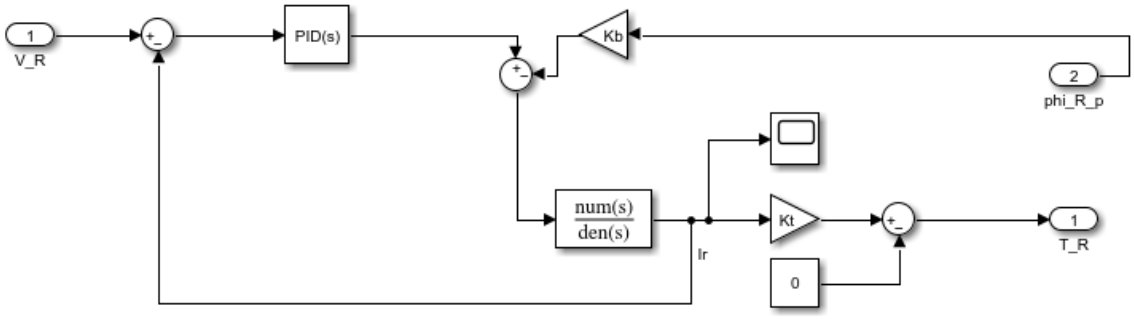


Figure I.13_Bloc moteur

Le modèle complet (modèle dynamique avec actionneur) :

Maintenant nous pouvons déterminer le modèle complet dynamique avec actionneur de notre robot comme nous montre la Figure I.14:

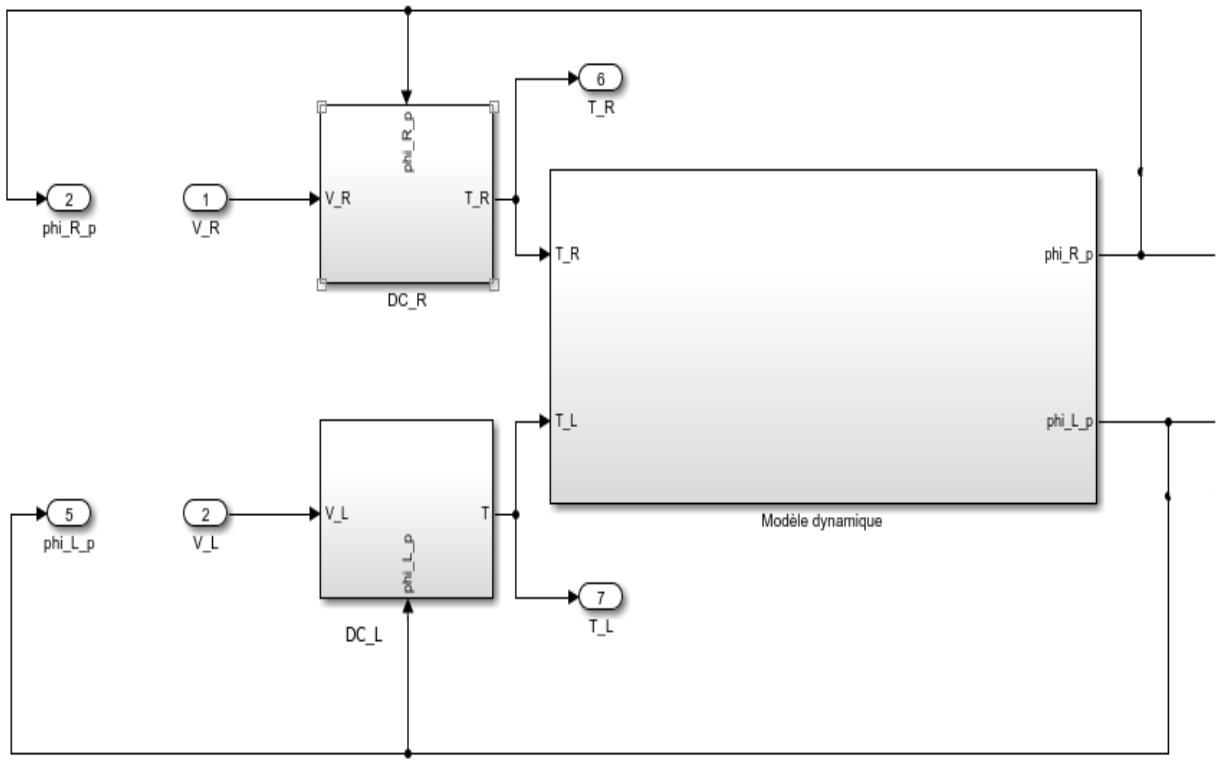


Figure I.14_DDMR modèle dynamique avec actionneur

IV. Conclusion

Dans **ce chapitre**, nous avons abordé différents aspects du robot mobile, notamment sa classification générale, ses composants essentiels et sa modélisation. Nous avons développé les équations cinématiques et dynamiques spécifiques au robot mobile unicycle. Ces modèles dynamiques et cinématiques devraient définir les fondements du contrôle du robot. La modélisation mathématique du robot qui a été utilisée pour le robot à roues différentielles au moyen de certaines hypothèses énoncées précédemment. La suite de l'analyse présente sous une forme mathématique la nature non linéaire des systèmes réels.

CHAPITRE II : Suivi Le Trajectoire d'un Robot Mobile a deux roues

---

I. Introduction

II. Méthode de Suivi de Trajectoire

III. Exemple de comportement

IV. Conclusion

I. Introduction

Les robots mobiles à deux roues, souvent appelés robots à équilibrage dynamique ou unicycles, sont devenus populaires en robotique en raison de leur simplicité de conception et de leur maniabilité optimale, ainsi que de leur large application. En revanche, le contrôle précis de ces robots pour suivre des trajectoires déterminées reste un défi majeur.

Dans le domaine de la navigation des robots mobiles autonomes, il existe deux grandes catégories de méthodes permettant à un robot d'atteindre une position spécifique :

- Méthodes sans trajectoire.
- Méthodes de suivi de trajectoire

Ce chapitre présente un aperçu détaillé sur la méthode du suivi de trajectoire pour les robots mobiles à deux roues, en décrivant les concepts fondamentaux et comment le comportement d'un robot peut être décrit à l'aide d'équation**s mathématiques**. Pour cela, nous utilisons des concepts mathématiques qui nous permettent de voir le robot sous différents angles : sa position (à travers un modèle géométrique), sa vitesse (via un modèle cinématique) ou les forces qu'il exerce (en se basant sur un modèle dynamique), Cela nous permet donc de mieux comprendre et anticiper ses mouvements ainsi que la façon dont il interagit avec son environnement.

II. Méthode de Suivi de Trajectoire :

II.1 Définition et Importance

Le suivi de trajectoire implique la gestion d'un robot pour qu'il suive une trajectoire prédéfinie, Cette trajectoire est exprimée avec des positions et des orientations dans un espace particulier. Le suivi de la trajectoire n'est pas simplement la planification de la trajectoire mais également le contrôle en temps réel pour ajuster les erreurs ou les déviations de la trajectoire planifiée. Une trajectoire peut être formulée de manière continue (via des courbes) ou discrète (via des points de passage) **[10]**.

L'importance du suivi de trajectoire réside dans la capacité du robot à accomplir des tâches complexes avec précision et efficacité. Par exemple, les robots mobiles dans les entrepôts suivent des trajectoires élaborées pour transport de marchandises tout en évitant les obstacles.

Pour les véhicules autonomes, un suivi de trajectoire précis est essentiel pour garantir la sécurité et l'efficacité de la navigation. **[11].**

II.2 Principes de comportement [12][13] :

Le suivi de trajectoire est un aspect essentiel de la robotique mobile, en particulier pour les robots à deux roues ou les unicycles. Il s'agit d'un processus en deux étapes : la génération de la trajectoire et le contrôle de la trajectoire.

II.2.1 Génération de la trajectoire :

La génération de plan consiste à déterminer comment un robot se déplace en fonction de connaissances préalables (statiques) ou acquises en temps réel (dynamiques). Ce processus repose sur trois concepts essentiels :**[14]**

- **Stratégie de navigation :** La stratégie de navigation commence par l'idée générale de quel est le plan approprié pour amener en toute sécurité le robot à partir d'un point à un autre. Elle peut inclure des décisions telles que l'utilisation d'une navigation réactive (basée sur les informations immédiates) ou d'une navigation planifiée (basée sur une trajectoire prédéfinie).

- **Modélisation de l'espace :** D'un autre côté, l'environnement du mobile robot ne peut jamais être connu à l'avance, mais il est obtenu après une campagne de mesures. Ce modèle de l'espace est utilisé pour représenter les obstacles, les zones libres et d'autres entités essentielles dans la navigation.

- **Planification de trajectoire :** Une fois que la carte de l'environnement est disponible et que la position actuelle du robot est connue, la planification de trajectoire consiste à anticiper les mouvements nécessaires pour atteindre un objectif fixe.

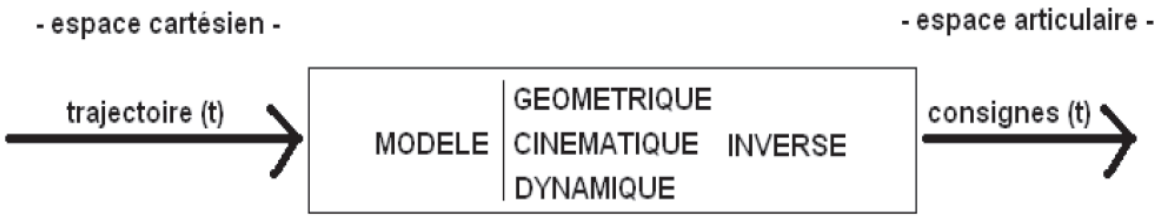


Figure II.1_Approche par modèle inverse pour effectuer un suivi de trajectoire

II.2.2 Contrôle de la trajectoire :

Une fois la trajectoire planifiée, le robot doit être capable de la suivre avec précision. Cela implique un contrôle en temps réel pour corriger toute déviation par rapport à la trajectoire planifiée. Le contrôleur de trajectoire utilise généralement des techniques de contrôle comme le contrôle PID (Proportionnel – Intégral - Dérivative) ou le contrôle par retour d'état pour assurer un suivi précis de la trajectoire.

Et voici une architecture de système de contrôle bien détailler au-dessous :

II.2.2.1 Conception du contrôleur de trajectoire :

En général, la conception du contrôleur de trajectoire implique les étapes suivantes :

- **Conception du contrôleur cinématique :** le but de ce contrôleur est d'obtenir l'erreur de position un comportement asymptotiquement stable. Ce contrôleur commande la position du robot en utilisant des techniques cinématiques pour réaliser un mouvement presque parfait en termes de vitesse et d'accélération.
- **Conception du contrôleur dynamique :** Ce contrôleur calcule le couple nécessaire pour que les vitesses du robot convergent vers les valeurs souhaitées par le contrôleur cinématique. En d'autres termes, il commande la vitesse du robot en utilisant les sorties du contrôleur cinématique.

II.2.2.2 Contrôle de trajectoire avec PID :

Le contrôleur PID est couramment utilisé pour réguler la trajectoire d'un robot mobile à deux roues, voici comment il fonctionne :

- **Proportionnel (P) :** Calcule l'erreur entre la position réelle et la position désirée. Le terme proportionnel ajuste la vitesse des roues en fonction de cette erreur.
- **Intégral (I) :** Accumule l'erreur au fil du temps et compense les erreurs systématiques. Il permet d'éliminer les erreurs de positionnement permanentes.
- **Dérivé (D) :** Anticipe les variations futures de l'erreur et réduit les oscillations. Il améliore la stabilité du système.

II.2.2.3 Approche de réglage :

Le réglage d'un contrôleur PID (Proportionnel-Intégral-Dérivé) est une étape essentielle pour assurer le bon fonctionnement et la stabilité d'un système, Dans cette section s'intéresse juste sur la méthode de réglage pas a pas pour régler correctement les gains (Kp), (Ki), et (kd) d'un contrôleur PID **[15]**:

Le Réglage Pas à Pas : En suivant ces étapes et en ajustant soigneusement les gains PID, on obtient un réglage optimal pour notre système de contrôle de robot mobile à deux roues :

1. Déterminer le comportement du système sans contrôle pour établir une ligne de base.
2. Configurer le PID en mode P seulement :
   - Commencez avec (Kp), (Ki), (Kd).
   - Augmentez progressivement (Kp) jusqu'à ce que le système commence à répondre correctement mais sans oscillations excessives.
3. Ajouter la dérivation :
   - Introduisez (Kd) pour réduire les oscillations et améliorer la stabilité.
   - Ajustez (Kd) jusqu'à obtenir une réponse douce et stable.

II.2.2.4 Avantages de l'utilisation d'un contrôleur PID :

- Simple à implémenter et à comprendre.
- Robuste aux perturbations.
- Efficace pour contrôler des systèmes non linéaires.

II.2.2.5 Inconvénients de l'utilisation d'un contrôleur PID :

- Peut être sensible au bruit.
- Les performances peuvent être dégradées si les gains du contrôleur ne sont pas réglés correctement.

II. Estimation de la Position et de l'Orientation :

Imaginons un robot à entraînement différentiel positionné en A (coordonnées (X_0, Y_0 , θ_0)) avec une distance non nulle par rapport à la position cible B (coordonnées (X_f, Y_f, θ_f))) définie dans le cadre inertiel global. En robotique mobile, il est essentiel de comprendre le comportement mécanique du robot. Cela nous permet de concevoir des robots adaptés aux tâches spécifiques et de développer des logiciels de contrôle efficaces. **[16]**

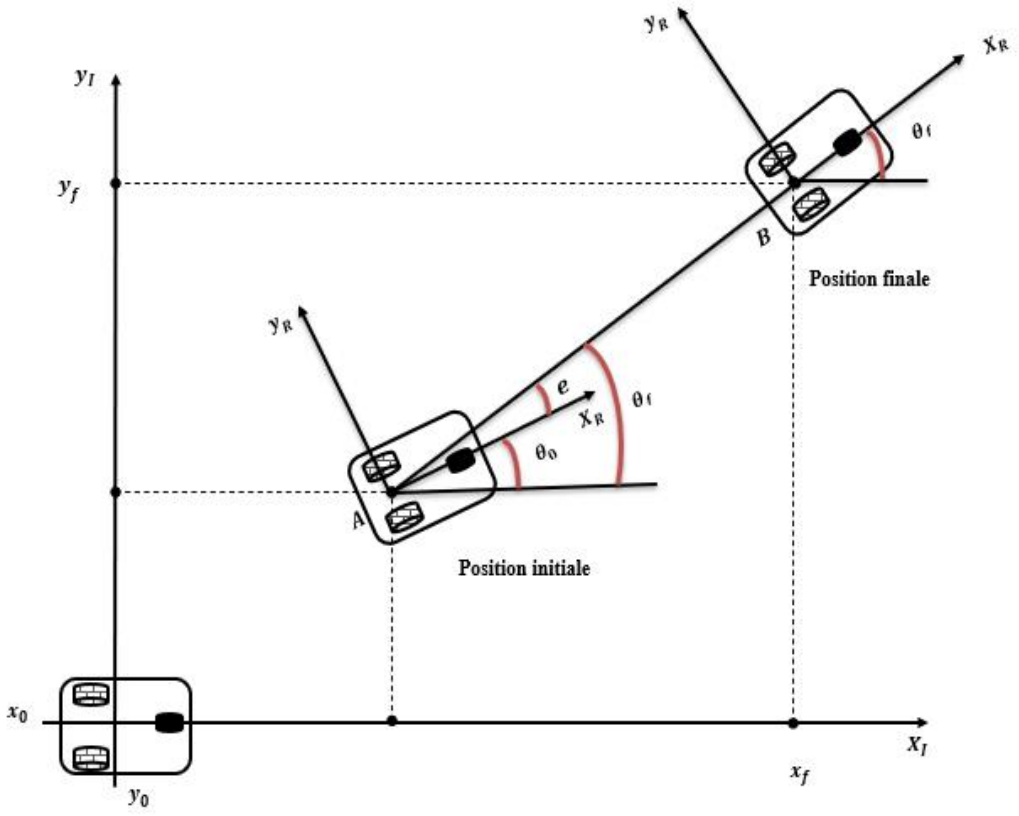


Figure II.2_ La position et orientation de robot mobile a deux roues

III. Exemple de comportement :

III. 1 Identification des paramètres :

Afin de simuler notre modèle, il est nécessaire de déterminer les paramètres du moteur. À cette fin, nous avons consulté la documentation fournie avec le moteur pour obtenir les informations nécessaires.

```
%% Paramètres

R=0.1              %% est le rayon de la roue
L=0.2              %% est la largeur de robot
Iw=0.0001          %% le moment d'inertie de chaque roue avec le moteur par rapport à l'axe de la roue
mw=0.5             %% la masse de la roue avec le moteur
mc=1               %% la masse de châssis
d=0.05             %% est la distance entre le point A et le point C
m=mc+2*mw          %% est la masse totale de robot
I=0.001            %% est l'inertie totale équivalent de système
La=0.001           %% l'inductance des armateurs du moteur [H]
Ra=1.9             %% résistance des armateurs de moteur [ohm]
Kb=4.1*10^-2       %% constante de la force électromotrice [V.s /rad]
Kt=0.16            %% constante de couple moteur [N.m/A]
N=1                %% le rapport de réduction

m11=Iw+(((R^2) / (4*L^2)) * (m*L^2+I))
m12=(((R^2) / (4*L^2)) * (m*L^2-I))
m21=m12
m22=Iw+(((R^2) / (4*L^2)) * (m*L^2+I))
M= [m11 m12 ; m21 m22]

%matrice de Coriolis.
v11=0
v12=((R^2) / (2*L)) * (mc*d)
v21=-((R^2) / (2*L)) * (mc*d)
v22=0
V= [v11 v12 ; v21 v22]

%PID
Kp=1
Ki=0.1
Kd=0.01

% Conditions initiales
x0=0
y0=0
θ0=0
```

III.2 Commande de mouvement :

A partir de ce schéma nous pouvons construit notre fonction Matlab Sous Simulink :

```
function [v, w, xf, yf, e] = fcn(t, x0, y0, theta0)

x=0.1*t+0.5*cos(2*pi*t/50)-0.5;
y=0.1*t+0.5*sin((2*pi*t/50));

dy=y-y0;
dx=x-x0;
xf=x;
yf=y;

d=sqrt(dx^2+dy^2);

alpha=atan2(dy,dx);

if dy<0
    alpha=2*pi+alpha;
end

e=alpha-theta0; %%l'erreur
v=0.5*d;
w=2*e;
```

Figure II.3_Fonction de commande de mouvement de robot mobile

Dans ce mode de commande, les mouvements sont prédéterminés et exécutés sans prendre en compte les informations de rétroaction de la disposition ou de l'état actuel du système. Ces mouvements sont plutôt bons pour être des actions simples et répétitives, mais ils sont rarement suffisamment précis ou suffisamment puissants pour fonctionner correctement dans les situations où l'environnement autour du robot est variant ou dynamique. Le système de coordonnées du robot est déterminé principalement par sa vitesse linéaire v et sa vitesse angulaire ω.

III.3. Méthode de Commande et de planification sur Matlab :

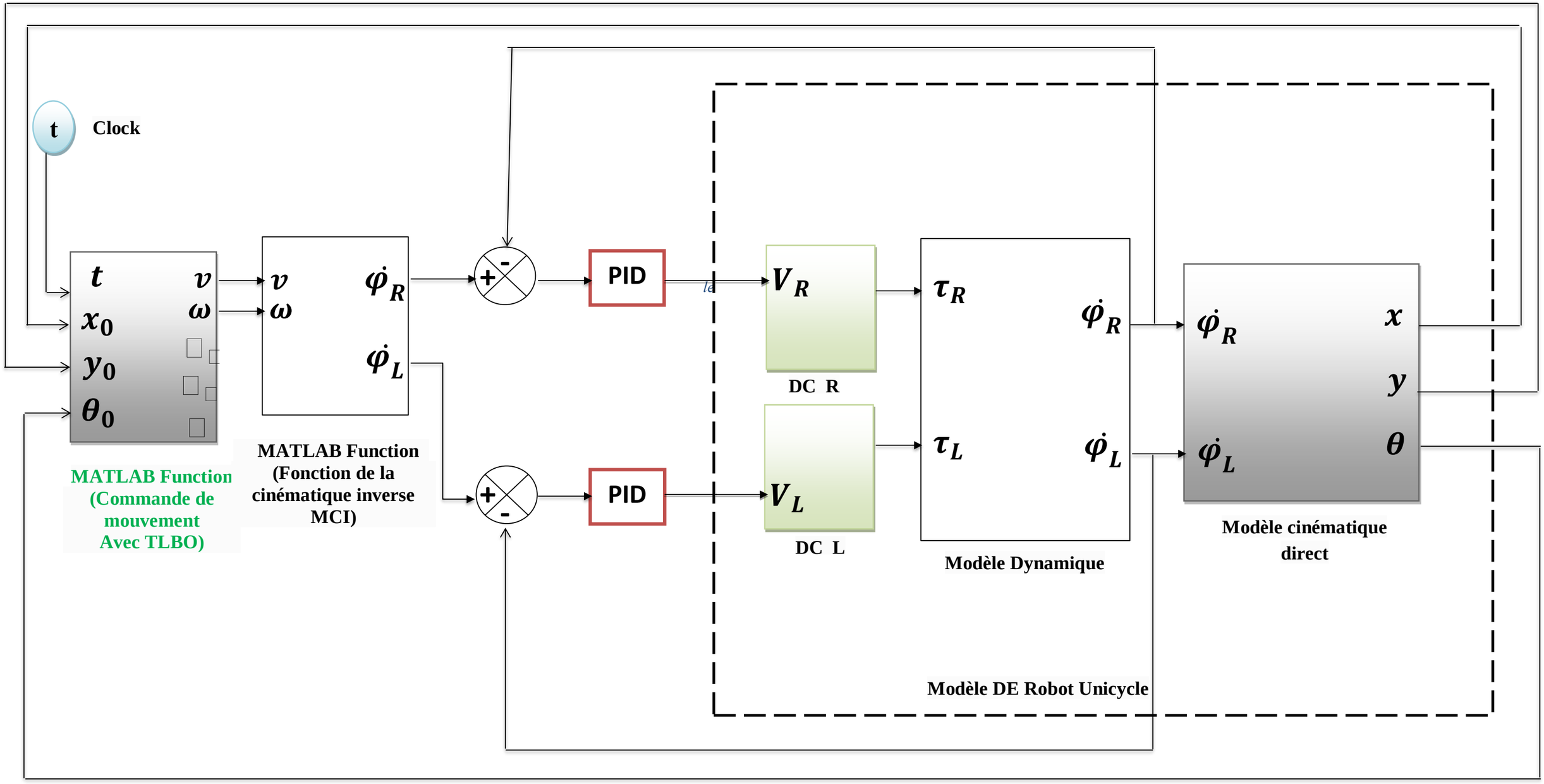


Figure II.4_Méthode de commande et de planification

III.4. Résultat de simulation :

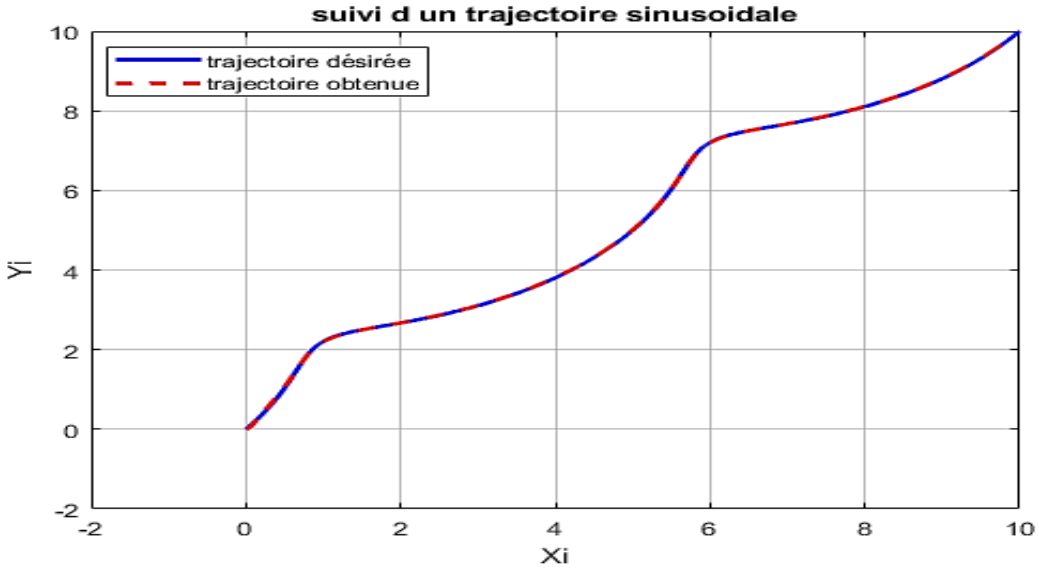


Figure II.5_Suivi la trajectoire sinusoïdale

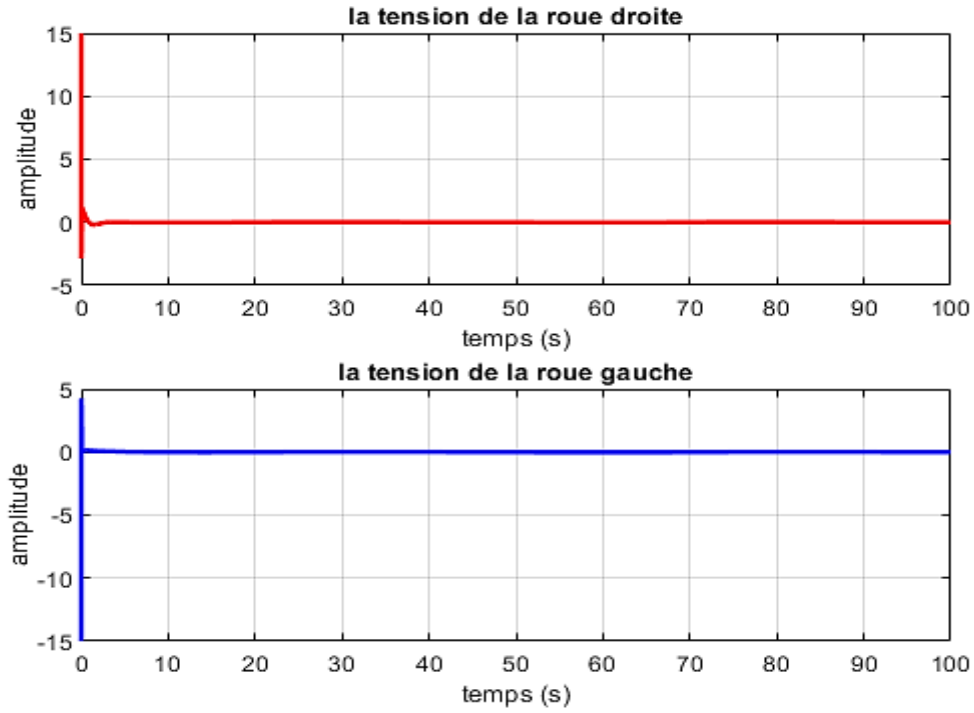


Figure II.6_La tension de la roue droite et gauche

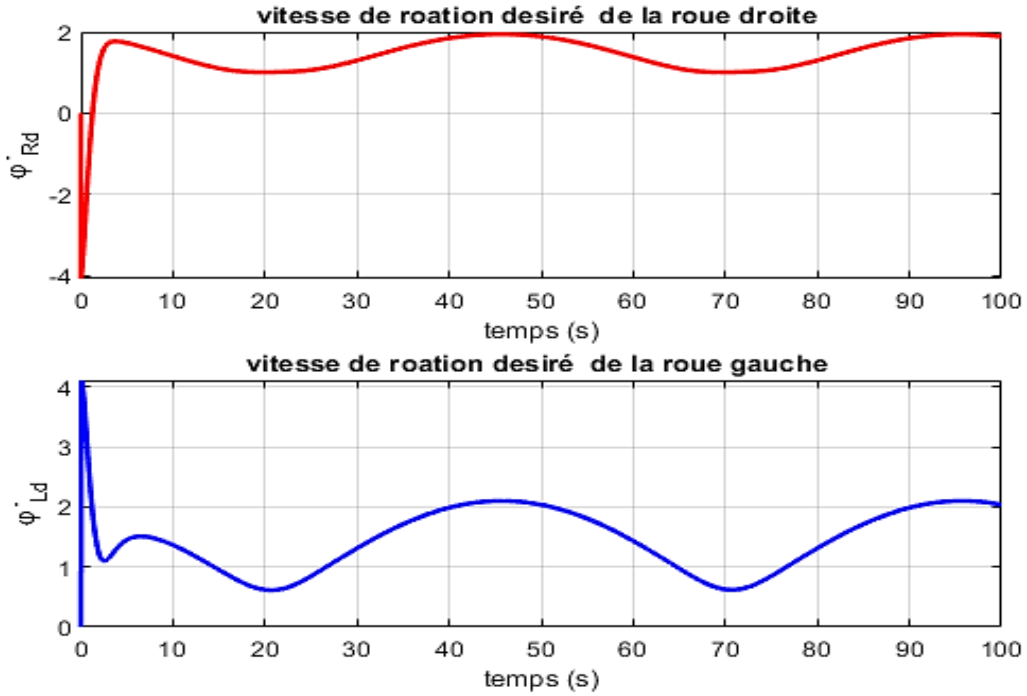


Figure II.7_Vitesses désirées des roues droites et gauche

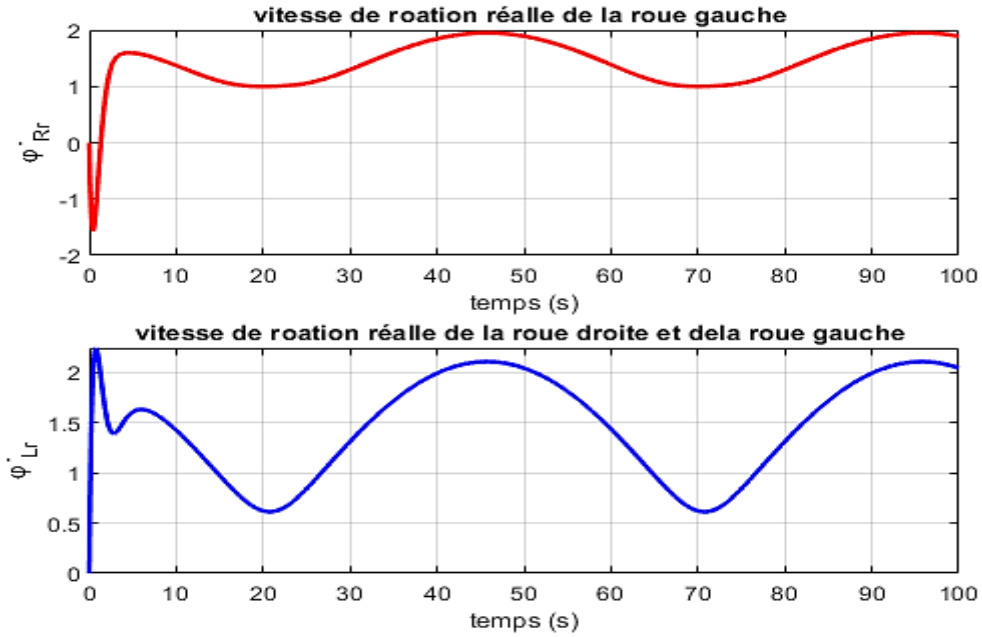


Figure II.8_Vitesse de rotation réelle de la roue droite et gauche

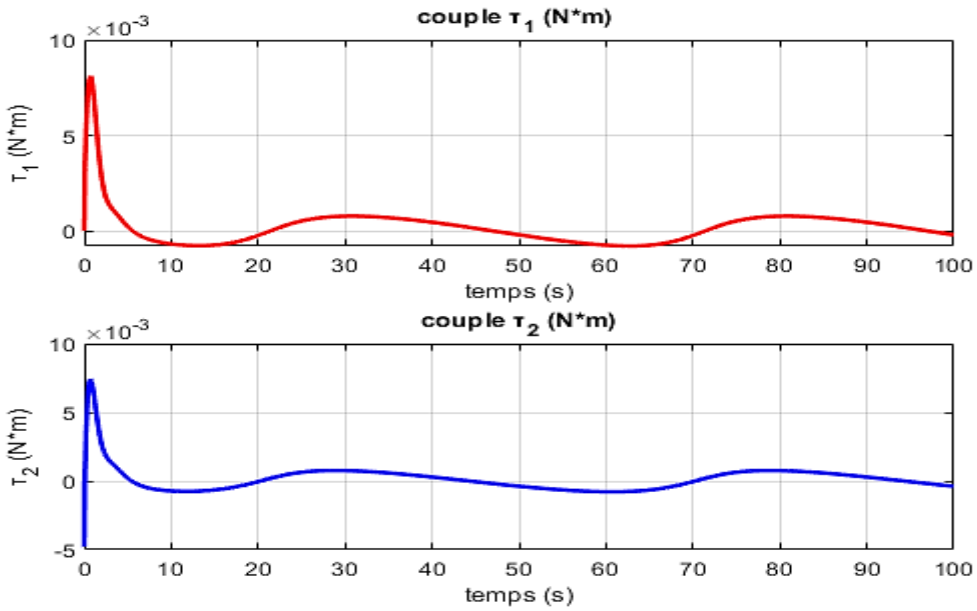


Figure II.9_Les couples de moteur de la roue droite et gauche τ_1 et τ_2

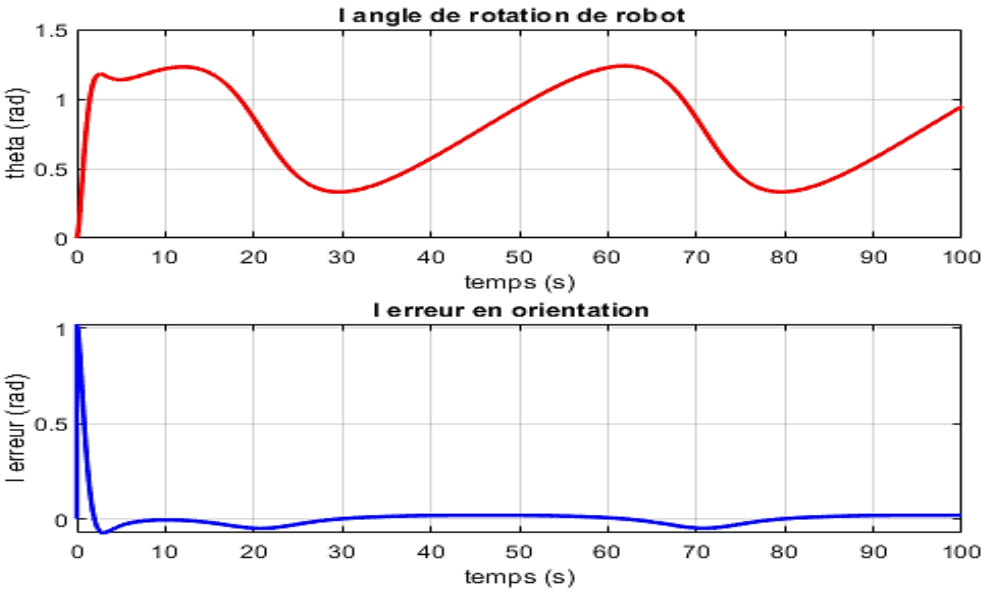


Figure II.10_L'angle de rotation et l'erreur d'orientation

III.5 Description des résultats obtenues :

La figure (II.5) présente la trajectoire obtenue et la trajectoire désirée, on remarque que le robot arrive à suivre la trajectoire. La figure (II.6) présente la tension appliquée aux deux moteurs et les valeurs de tension trouvée sont réalisables (Acceptables).

La figure (II .7) et (II .8) présente les vitesses des deux roues, la figure (II .9) présente les couples moteurs générés.

La figure (II .10) montre l'erreur de poursuite on remarque que l'erreur est faible.

IV. Conclusion

Le suivi de trajectoire est un élément fondamental de la navigation autonome pour les robots mobiles à deux roues. En combinant les concepts théoriques et les considérations pratiques présentées dans ce chapitre, les chercheurs, les ingénieurs et les passionnés de la robotique peuvent concevoir et développer des systèmes autonomes, performants, et capables de relever les défis du monde réel et d'accomplir des missions complexes avec une efficacité remarquable.

En résumé, la modélisation cinématique et dynamique est fondamentale pour planifier la trajectoire et assurer un contrôle précis d'un robot mobile à deux roues. On peut, avec ces équations et modèles, développer des algorithmes de commande performants qui permettent au robot de se déplacer efficacement vers sa destination.

CHAPITRE III : Algorithme D'optimisation TLBO

I. Introduction

II. Introduction sur les algorithmes

III. L'Algorithm TLBO (Teaching-Learning-Based Optimization algorithm)

IV. Planification de trajectoires guidées par TLBO

V. Conclusion

I. Introduction :

Dans ce chapitre, nous présentons un algorithme d'optimisation métaheuristique nommé Teaching-Learning Based Optimization (TLBO). Cet algorithme est conçu pour obtenir des solutions globales à des problèmes d'optimisation avec une précision satisfaisante tout en minimisant la charge de calcul. Le TLBO simule le processus d'enseignement-apprentissage pour résoudre des problèmes multidimensionnels, linéaires et non linéaires de manière efficace **[17]**.

Dans ces contextes, Un ajustement incorrect des paramètres spécifiques d'un algorithme peuvent augmenter la charge de calcul ou conduire à des optimums locaux. Par conséquent, Rao et al. ont élaboré l'algorithme Teaching-Learning-Based Optimization en 2011. Au lieu d'un ensemble spécifique de paramètres, le TLBO est efficace et fonctionne bien sans eux, en fonction des paramètres de contrôle habituels tels que la taille de la population et le nombre de générations.

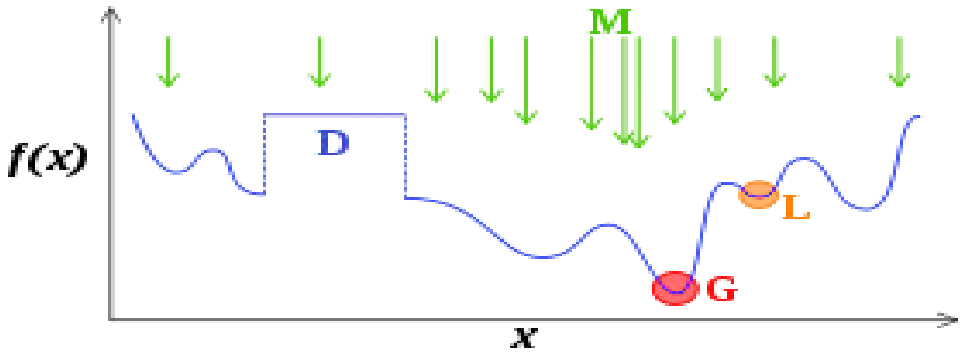
II. Introduction sur les algorithmes :

II.1. Optimisation métaheuristiques :

Les métaheuristiques, terme issu de la combinaison des mots grecs "méta" signifiant "au-delà" et "heuristique" pour "je trouve", ont été introduites par Fred Glover en 1986 **[18]**. Ces algorithmes d'optimisation, souvent inspirés de la nature, sont conçus pour résoudre des problèmes complexes où d'autres méthodes échouent. Les métaheuristiques sont des approches «génériques» utilisées pour optimiser une variété de problèmes. Ils ne garantissent pas de trouver la solution optimale; cependant, la plupart de ces méthodes cherchent une approximation rapide de l'optimum global plutôt que de rechercher une solution exacte, ce qui prendrait trop de temps.

Parmi les métaheuristiques, celles-ci basées sur la population sont les plus courantes. Ils peuvent être divisés en deux groupes principaux: la première solution travaille avec un groupe de solutions en parallèle (algorithme génétique, colonies de fourmis, optimisation de l'essaim particulaire – PSO, TLBO, etc.), la deuxième solution se concentre sur l'évolution itérative d'une seule solution (comme la méthode Tabou et le Recuit Simulé) **[19]**.

Ces algorithmes manipulent plusieurs solutions à la recherche de l'optimum et s'arrêtent selon des critères tels que le temps d'exécution ou la précision désirée.



Les métaheuristiques (M) sont souvent des algorithmes utilisant un échantillonnage probabiliste. Elles tentent de trouver l'optimum global (G) d'un problème d'optimisation difficile (avec des discontinuités — D —, par exemple), sans être piégé par les optima locaux (L) [20].

II.2. Rappel sur le problème d'optimisation :

Avant de commencer avec le TLBO, il est important de bien comprendre le problème d'optimisation.

II.2.1. Définition :

L'optimisation est un processus qui cherche à maximiser ou minimiser un critère spécifique, qu'il soit économique (comme la productivité, le coût d'investissement, le coût de gestion, etc.), technique (comme la précision, la stabilité, etc.) ou physique (comme la quantité de chaleur, l'énergie, le couple, etc.). Ce processus doit respecter certaines contraintes liées au problème. L'objectif est de trouver les valeurs optimales des variables pour maximiser les performances et minimiser le coût d'un critère dépendant de ces variables **[21]**.

II.2.2. Fonction objectifs et variable de son expression :

La fonction objective est le critère du problème à optimiser, en le diminuant ou en l'augmentant selon ce qui est nécessaire. En termes mathématiques, la fonction objective est exprimée en fonction de variables, qui sont encore appelées variables d'optimisation. C'est l'ensemble de variables d'optimisation qui sont tirées parmi ces mesures qui affectent la valeur de la fonction objective et en même temps, peuvent prendre des valeurs optimales dans les conditions réelles **[22]**.

II.2.3. Formulation mathématique d'un problème d'optimisation :

Le problème d'optimisation est représenté mathématiquement par une fonction objective *f* définie dans l'espace *R*, ou dans une partie de R^n vers R^m, où l'on cherche la solution X^* qui minimise cette fonction. En général, les problèmes d'optimisation peuvent être classés en deux grandes catégories selon l'existence ou l'absence de contraintes, le problème d'optimisation peut être sans contraintes ou sous contraintes. Dans les deux cas, il peut s'agir de problèmes mono-objectifs ou multi-objectifs **[22]**.

II.2.4. Problème d'optimisation sans contrainte :

On dit qu'un problème d'optimisation est sans contraintes si aucune contrainte ou fonction contrainte est présente. La formulation d'un tel problème est écrite mathématiquement suivant la formule ci-dessous **[22]** :

$$\begin{cases} min\, f(x) \notin x \in R^n \\ x_{k\,min} \leq x_k \leq x_{k\,max} \quad k = 1, \ldots, n \end{cases} \qquad (III.1)$$

II.2.5. Problème d'optimisation avec contrainte :

Un problème d'optimisation est dit contraint ou avec contraintes, si des fonctions contraintes, d'égalité ou d'inégalité, ou les deux types au même temps, existent. La formule générale d'un problème d'optimisation contraint est donnée par l'équation suivante.

$$\begin{cases} min\, f(x) \notin x \in R^n \\ g_i(x) \leq 0 \quad i = 1 \cdots p \\ h_j(x) \leq 0 \quad j = 1 \cdots m \\ x_{k\,min} \leq x_k \leq x_{k\,max} \quad k = 1 \cdots n \end{cases} \qquad (III.2)$$

Où :

- ✓ $f\,(x)$ est le critère à minimiser, s'appelle la fonction objectif ou fonction coût.
- ✓ $X = [x1, x2, x3, \ldots, x_n]$, est le vecteur des variables d'optimisation de dimension n, représentant les paramètres à optimiser.
- ✓ $x_{k\,min}$ et $x_{k\,max}$ sont les valeurs minimale et maximale de chaque variable, indiquant aussi les contraintes du domaine.
- ✓ Les expressions $g_i\,(x)$ et $h_j\,(x)$ sont des contraintes d'inégalité.

Dans le cas d'un problème de maximisation (maximiser un critère), il suffit de le transformer en un problème de minimisation en utilisant l'inverse ou l'opposé de f(x). C'est-à dire : min (– f(x)), cela permet de conserver toutes les généralités mentionnées précédemment.

III. L'algorithme TLBO (Teaching-Learning-Based Optimization algorithm) :

Un algorithme d'optimisation basé sur l'enseignement-apprentissage est un algorithme de nature sociale. Il imite le processus d'enseignement-apprentissage en classe, introduit sous une forme mono-objective. Cependant, à la différence des algorithmes traditionnels basés sur la population, l'algorithme TLBO est principalement composé de deux phases : la phase d'enseignement (Teaching phase) et la phase d'apprentissage (Learning phase). Les apprenants acquièrent des connaissances non seulement à partir de l'enseignement dispensé par un enseignant, mais aussi à travers leur interaction mutuelle via des discussions de groupe, des présentations et des communications formelles. Chaque individu de la population est considéré comme un apprenant et les variables sont considérées comme des sujets à apprendre. Après évaluation des apprenants, le meilleur apprenant est choisi pour devenir un enseignant pour la prochaine phase d'enseignement. La phase d'apprentissage, quant à elle, sert à échanger des connaissances entre les apprenants. Plusieurs variantes de TLBO nécessitent uniquement des paramètres communs, tels que le critère d'arrêt (nombre d'évaluations ou nombre de générations) et la taille de la population **[23], [24], [25]**.

III.1. Principe de l'algorithme TLBO :

L'algorithme se compose de deux phases principales : la phase de l'enseignant et la phase de l'élève. Voici les détails de ces deux phases :

a) Phase de l'enseignant (Teacher) :

La phase enseignant est la première partie de l'algorithme où les apprenants améliorent leur niveau grâce à l'enseignant. Durant cette phase, l'enseignant cherche à augmenter le résultat moyen de la classe dans la matière enseignée. À chaque itération *i*, avec *m* matières et *n* apprenants, l'algorithme désigne le meilleur apprenant comme enseignant. La différence entre le résultat moyen des apprenants dans chaque matière et celui de l'enseignant est calculée par :

$$Difference\_Mean_i = r_i\,(M_{new} - TFM_i) \quad (III.3)$$

Où

- $r_i \epsilon [0,1]$ Est un nombre aléatoire de la distribution uniforme.
- TF Est le facteur d'enseignement qui influence le taux de variation de la nouvelle moyenne et ne peut prendre que deux valeurs $[1,2]$.

$$TF = round[1 + rand(0,1)\{2-1\}] \quad (III.4)$$

Sur la base de $Difference\_Mean_i$, la solution existante est mise à jour dans la phase de l'enseignant selon l'expression suivante :

$$X_{new,i} = X_{old,i} + Difference\_Mean_i \quad (III.5)$$

$X_{new,i}$ Est la mise à jour de la valeur $X_{old,i}$, La solution $X_{new,i}$ est accepté si elle donne une meilleure valeur de la fonction objective. A la fin de la phase professeur, toutes les valeurs acceptées sont conservées et deviennent les entrées de la phase apprenants. La phase apprenants dépend de la phase professeur.

b) Phase des élèves (Learners) :

Un apprenant augmente son savoir dans deux cas : à l'aide de l'enseignant et en interagissant avec d'autres étudiants. ou un apprenant se connecte au hasard à d'autres apprenants et échange des informations via des discussions de groupe, des présentations, ou des communications formelles. Un apprenant apprend quelque chose de nouveau si l'autre apprenant a plus de connaissances. La mise à jour des connaissances de l'apprenant se fait comme suit :

Pour chaque apprenant i de la population Pn :

- Sélectionner aléatoirement deux apprenants X_i et X_j, où $i \neq j$.
- Si $f(X_i) < f(X_j)$ alors :

$$X_{new,i} = X_{old,i} + r_i\,(X_j - X_i) \quad (III.6)$$

Sinon :

$$X_{new,j} = X_{old,j} + r_i (X_i - X_j) \quad \textit{(III.7)}$$

- Accepter X_{new} si cela donne une meilleure valeur de la fonction objective.

III.2. Paramètres de TLBO :

- **Taille de la population (N) :** Nombre d'individus (ou solutions) dans la population.
- **Dimension du problème (D) :** Nombre de variables à optimiser.
- **Limites de l'espace de recherche ([min, max]) :** Les bornes inférieure et supérieure pour chaque dimension.
- **Nombre d'itérations :** Nombre maximum d'itérations ou critère d'arrêt.
- **Facteur d'enseignement (TF) :** Valeur aléatoire (1 ou 2) déterminant l'influence du meilleur individu sur les autres.
- **Variable aléatoire ri :** Nombre aléatoire [0,1] utilisé pour les mises à jour.

III.3. L'algorithme TLBO :

L'algorithme de la méthode TLBO peut être résumée comme ci-dessous **[26]** :

Pseudo code de l'algorithme de TLBO

1. Définir le problème d'optimisation, sa dimension D et l'espace de recherche min et max ($[\boldsymbol{min}, \boldsymbol{max}]^{\boldsymbol{D}}$)
2. Initialiser la population de N individus uniformément dans l'espace ($[\boldsymbol{min}, \boldsymbol{max}]^{\boldsymbol{D}}$)
3. évaluer le fitness des particules
4. **while** Condition d'arrêt n'est pas satisfaite **do**
5. $\boldsymbol{X_B}$= meilleur individu de la population
6. **/* Phase d'enseignement */**
7. **for i=1 : N**
8. $TF = round[1 + rand(0,1)\{2-1\}]$
9. $\boldsymbol{Difference\_Mean_i} = r_i (M_{new} - TFM_i)$
10. $\boldsymbol{X_{new,i}} = X_{old,i} + Difference\_Mean_i$
11. Vérifier si la position des élèves $X_{new,i}$ n'a pas dépassé les limites min et max
12. **if fitness** $X_{new,i}$ n'est pas meilleure que **fitness** ($X_{old,i}$) **then**
13. $X_{new,i} = X_{old,i}$
/* la valeur de $X_{new,i}$ obtenue en ligne 10 n'est gardée que si elle est */

/* meilleur que $X_{old,i}$*/
14. **end if**
15. **end for**
16. **/* Phase d'apprentissage */**
17. **for i= 1 : N**
18. **j**= nombre entier aleatoire entre 1 et N
19. **if fitness** ($X_{old,i}$) est meilleure que **fitness** ($X_{new,i}$) **then**
20. $\boldsymbol{X_{new,i} = X_{old,i} + r_i (X_j - X_i)}$
21. **if fitness** ($X_{new,i}$) n'est pas meilleure que **fitness** ($X_{old,i}$)
22. $X_{new,i} = X_{old,i}$
23. **end if**
24. **else**
25. $\boldsymbol{X_{new,j} = X_{old,j} + r_i (X_i - X_j)}$
26. **if fitness** ($X_{new,j}$) n'est pas meilleure que **fitness** ($X_{old,j}$)
27. $X_{new,i} = X_{old,i}$
28. **end if**
29. **end if**
30. **end for**
31. **end while**
32. **Return the best particle.**

Pseudo-code de l'algorithme Teaching-leaming-based optimization

IV. Planification de trajectoires guidées par TLBO :

IV.1. Planifications de la trajectoire avec TLBO :

L'algorithme TLBO présenté précédemment est utilisé ici pour rechercher un chemin permettant au robot de se déplacer d'un point initial à un point final tout en évitant les collisions avec des obstacles. La position initiale du robot ($x0 = 0, y0 = 0$) et la position de destination ($x = 10, y = 10$). Plusieurs obstacles sont présents sur le chemin du robot.

IV.2. Résultats et discussion :

Les figures suivantes illustrent les résultats obtenus lors de la planification d'une trajectoire en utilisant l'algorithme précédent (TLBO). Ces résultats ont été obtenus en variant les ensembles de paramètres appliqués à l'algorithme. Et voici les trois expériences qu'on a fait :

- **Expérience 1 (3 obstacles) :**

La taille de la population est N=150. A chaque itération, TLBO dicte au robot le point suivant qu'il doit atteindre. Ce point est le meilleur individu dans la population.

Paramètres	Expériences 1		
	test 1	test 2	test 3
Nombre d'itération	500	200	50
Nombre des points intermediare	3	3	3
taille de population	150	150	150
Nombre d'obstacles	3	3	3

Tableau III.1: Paramètre des expériences 1.

La figure suivante représente la première expérience de la trajectoire prévue à l'aide de l'algorithme TLBO et sa poursuite par le robot.

Pour 500 itérations :

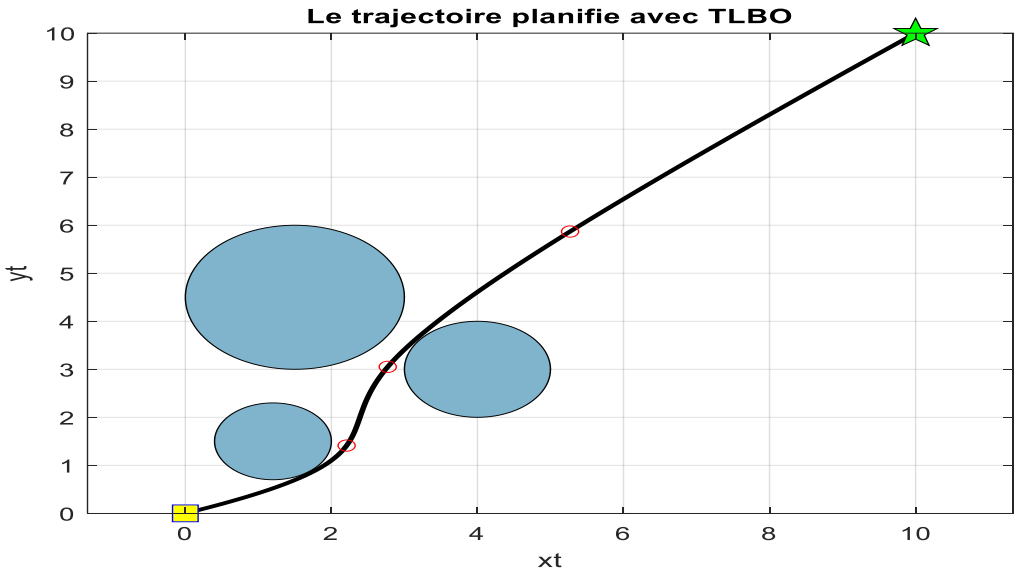


Figure III.1_La Trajectoire planifie avec le TLBO avec 3 obstacles et avec 500 itérations)

D'après la figure ci-dessus on remarque que le robot se dirige vers le but avec une trajectoire de forme rectiligne qui a pratiquement l'allure d'une droite et cela suite à la commande imposée.

Cependant quand il arrive à l'obstacle, il dévie avec un certain angle de sa trajectoire pour éviter la collision avec l'obstacle puis il suit sa trajectoire vers le but sous forme d'une droite. Ce comportement répond bien à l'influence de la technique proposée.

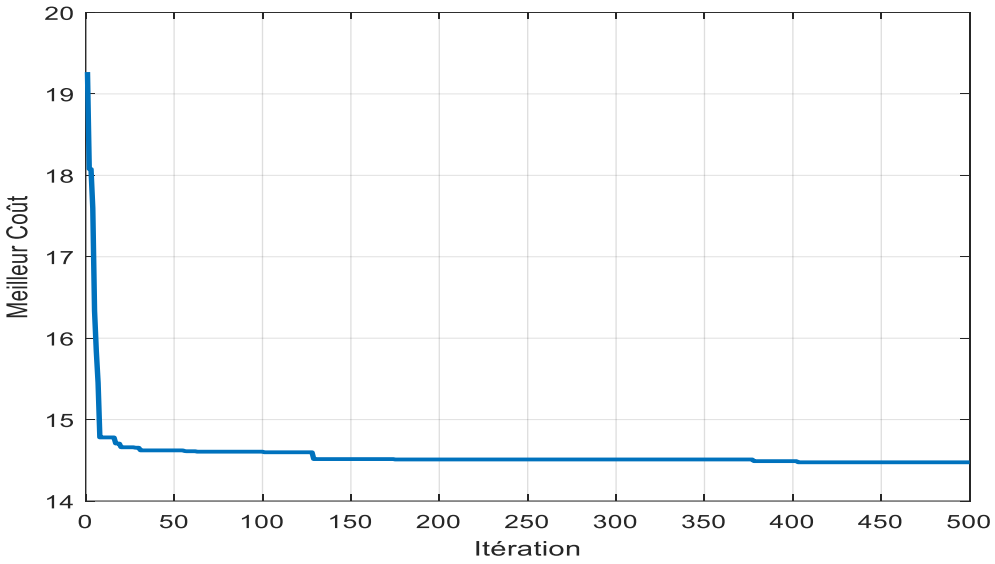


Figure III.2_Visualisation du BestCost

Visualisation du BestCost :

D'après la figure ci-dessus on remarque que le meilleur coût diminue quand le robot s'approche du but.

Pour 200 itérations :

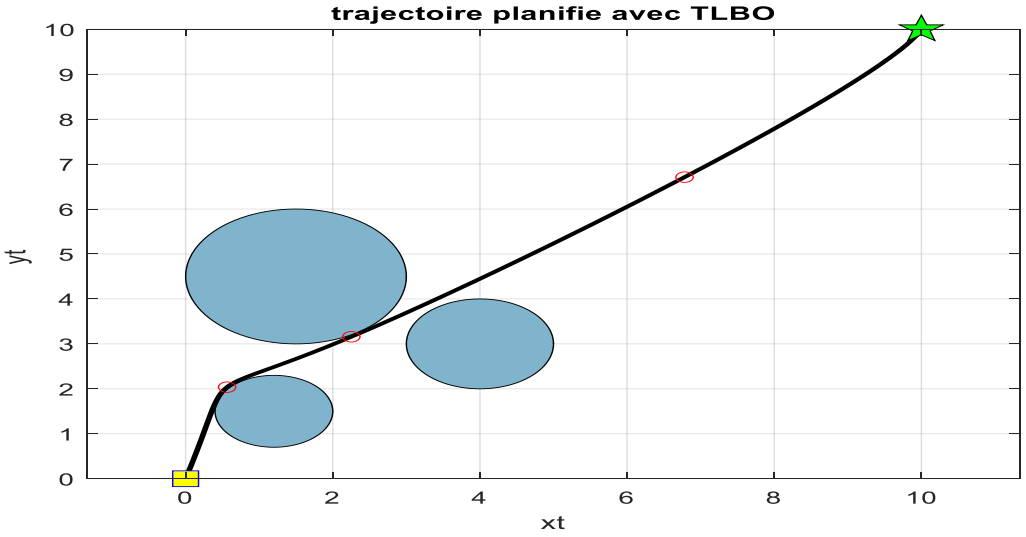


Figure III.3_La Trajectoire planifie avec le TLBO avec 3 obstacles en 200 itérations)

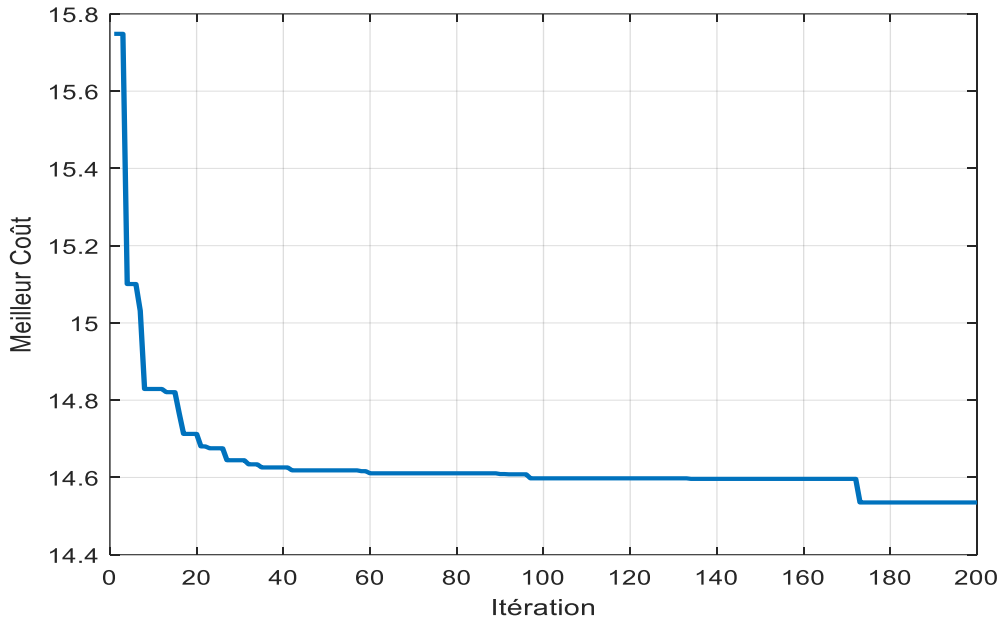


Figure III.4_Visualisation du BestCost

Pour 50 itérations :

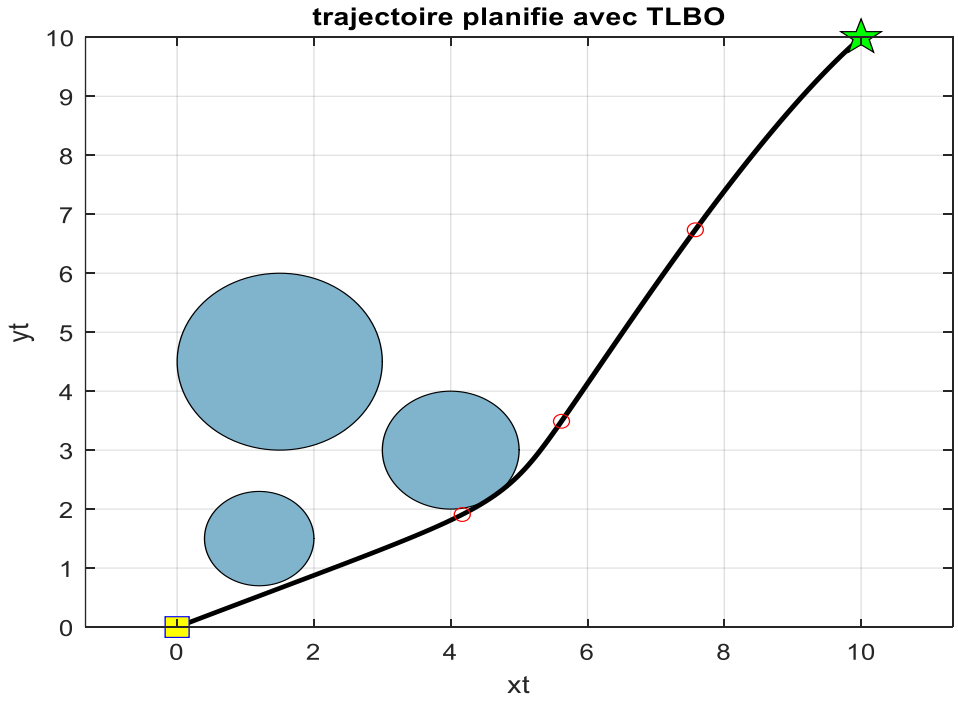


Figure III.5_La Trajectoire planifie avec le TLBO en présence de 3 obstacles avec 50 itérations)

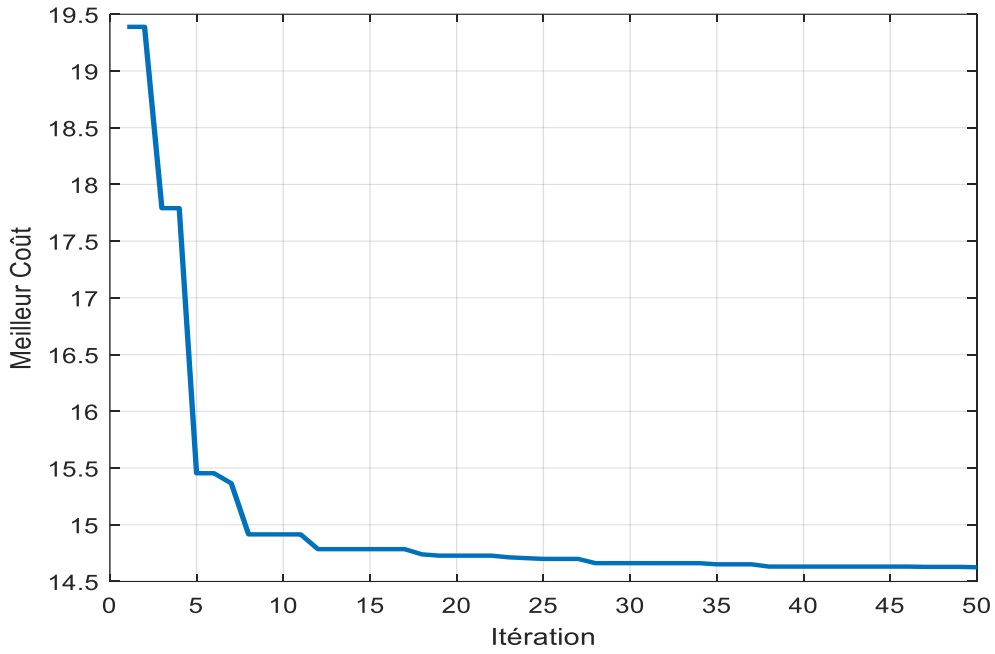


Figure III.6_Visualisation du BestCost

On constate un mouvement pseudo aléatoire et une trajectoire non lisse. Ce qui n'est pas vraiment stable.

D'après les trois figures ci-dessus on constate que la meilleure trajectoire ou le meilleur résultat dépend du nombre d'itérations η (il stable avec l'augmentation de η).

- **Expérience 2 (5 obstacles) :**

La taille de la population est N=150. A chaque itération, TLBO dicte au robot le point suivant qu'il doit atteindre. Ce point est la moyenne de la population. On constate que la trajectoire est beaucoup plus lisse que le cas précédent.

Paramètres	Expériences 2
Nombre d'itération	500
taille de population	150
Nombre d'obstacles	5
Nombre des points intermediare	5

Tableau III.2_Paramètre des expériences 2.

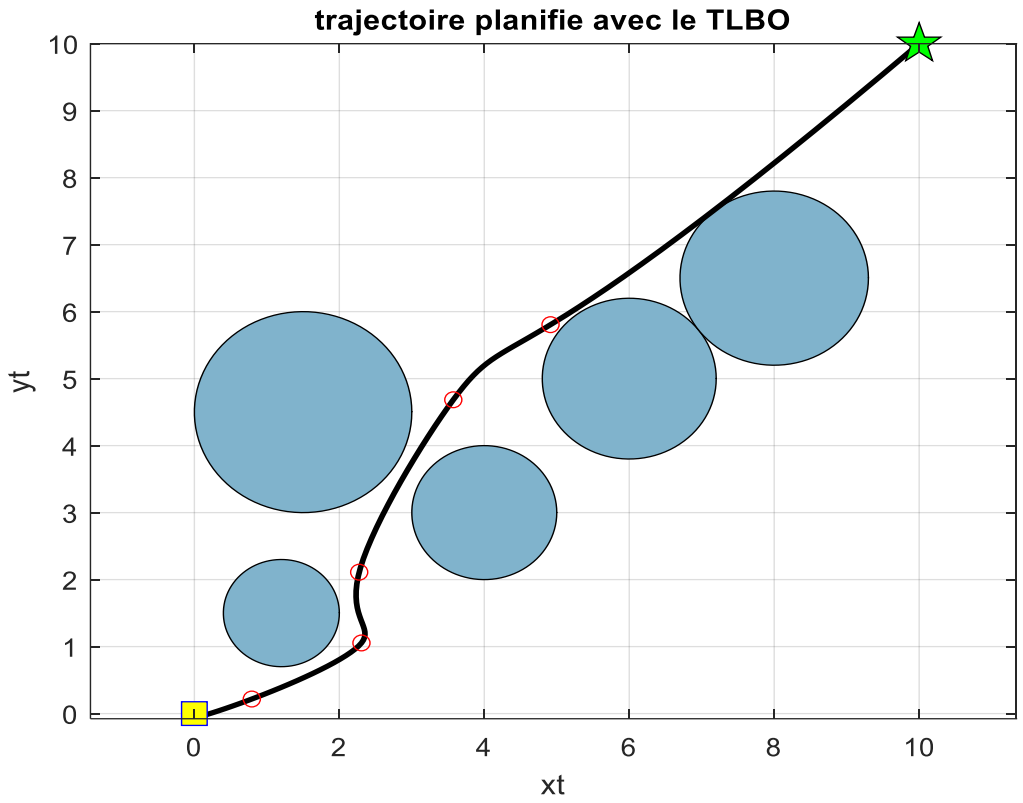


Figure III.7_La Trajectoire planifie avec le TLBO en présence de 5 obstacles avec 500 itérations)

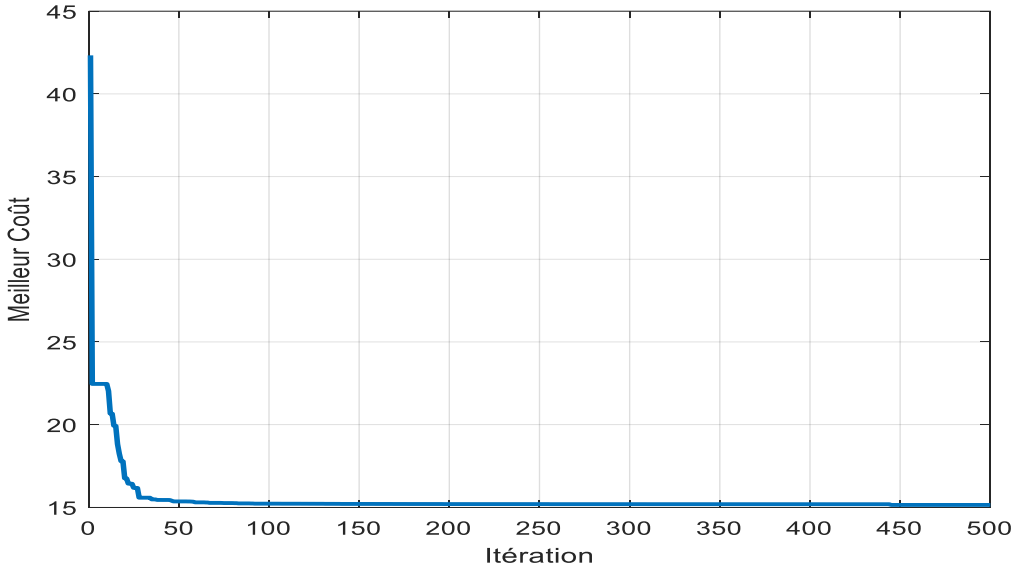


Figure III.8_Visualisation du BestCost

- **Expérience 3 (plusieurs obstacles)** :

La taille de la population est N=150. Pour 500 itérations, On suppose que notre espace de travail contient neuf obstacles, ces derniers genent le robot mobile dans son évolution du point de départ à l'arrivée au but. Apres certaines itérations on voit l'influence de la technique TLBO qui dicte au robot le point suivant qu'il doit atteindre.

Paramètres	**Expériences 2**
Nombre d'itération	500
taille de population	150
Nombre d'obstacles	9
Nombre des points intermediare	5

Tableau III.3_Paramètre des expériences 3.

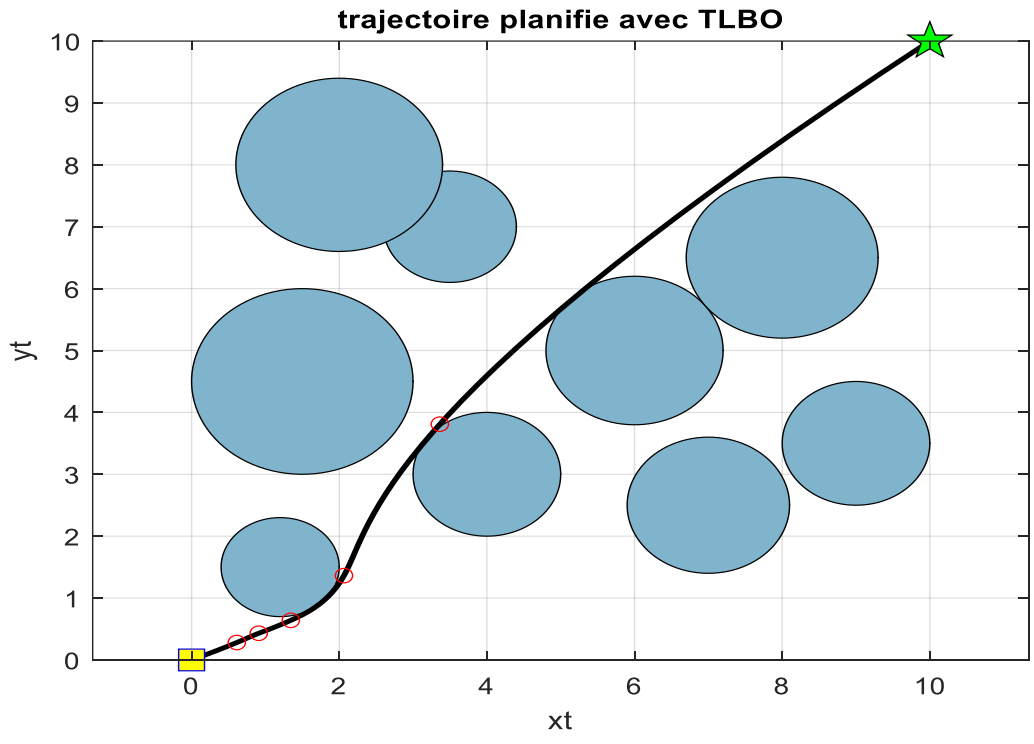


Figure III.9_La Trajectoire planifie avec le TLBO en présence de plusieurs obstacles avec 500 itérations)

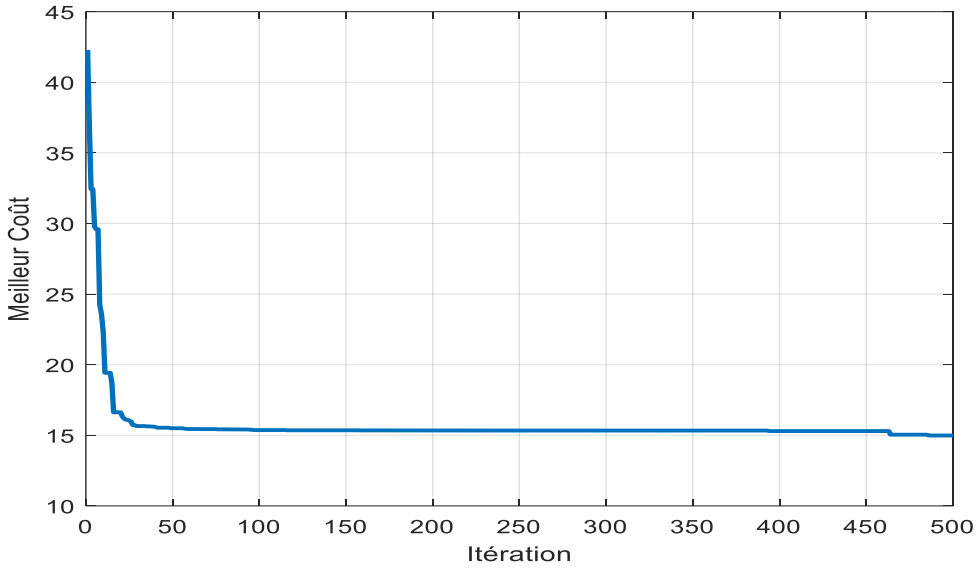


Figure III.10_Visualisation du BestCost

Visualisation du potentiel attractif :

D'après la figure ci-dessous on voit que le BestCost tend vers une valeur au voisinage du but.

V. Conclusion

La planification de trajectoire est un défi majeur dans le domaine de la robotique mobile, surtout lorsqu'il s'agit de trouver des trajectoires optimales dans des environnements complexes avec des obstacles. L'algorithme d'optimisation basé sur l'enseignement-apprentissage (TLBO) s'est révélé être une approche efficace et prometteuse pour aborder ce problème. Sa simplicité, combinée à son absence de paramètres spécifiques à ajuster, le rend particulièrement attrayant par rapport à d'autres méthodes méta-heuristiques.

Dans ce travail, nous avons démontré que TLBO permet de générer des trajectoires optimales **et lisses** tout en évitant efficacement les obstacles. L'algorithme exploite l'interaction entre les solutions pour raffiner les trajectoires, minimisant à la fois la distance parcourue et les risques de collisions avec des obstacles. Les simulations effectuées ont montré que TLBO est non seulement capable de résoudre le problème de planification de trajectoire avec succès, mais qu'il peut également être appliqué à des environnements de plus en plus complexes.

CHAPITRE IV : CONCEPTION MATERIELLE ET LOGICIEL DU ROBOT MOBILE

I. Introduction

II. Description matériel

III. Branchement globale du système

IV. La forme finale du robot

V. Implementation de programme

VI. Conclusion

I. Introduction

Un robot est un système complexe réalisé à partir de pièces mécaniques, électromécaniques et électroniques. Les robots peuvent être contrôlés au moyen d'un équipement électronique central unique appelé système embarqué qui peut être une simple séquence d'automatisme, un programme d'ordinateur informatique ou une véritable intelligence artificielle en fonction du degré de perfectionnement nécessaire pour accomplir les tâches planifiées. Les robots autonomes mobiles disposent également d'une source d'énergie embarquée, généralement une batterie d'accumulateurs électriques ou, pour les robots plus énergivores, un générateur électrique couplé à un moteur à essence.

Un robot se compose de trois parties essentielles :

1. **La partie mécanique** : Inclut toutes les pièces et structures mécaniques du robot.
2. **L'électronique de commande** : Se divise en trois unités :
    - **Unité de commande et de décision** : Généralement un microcontrôleur.
    - **Unité de perception de l'environnement** : Composée de capteurs.
    - **Unité de mouvement des moteurs** : Inclut le circuit de puissance.
3. **La partie de programmation** : Englobe le logiciel qui pilote le robot.

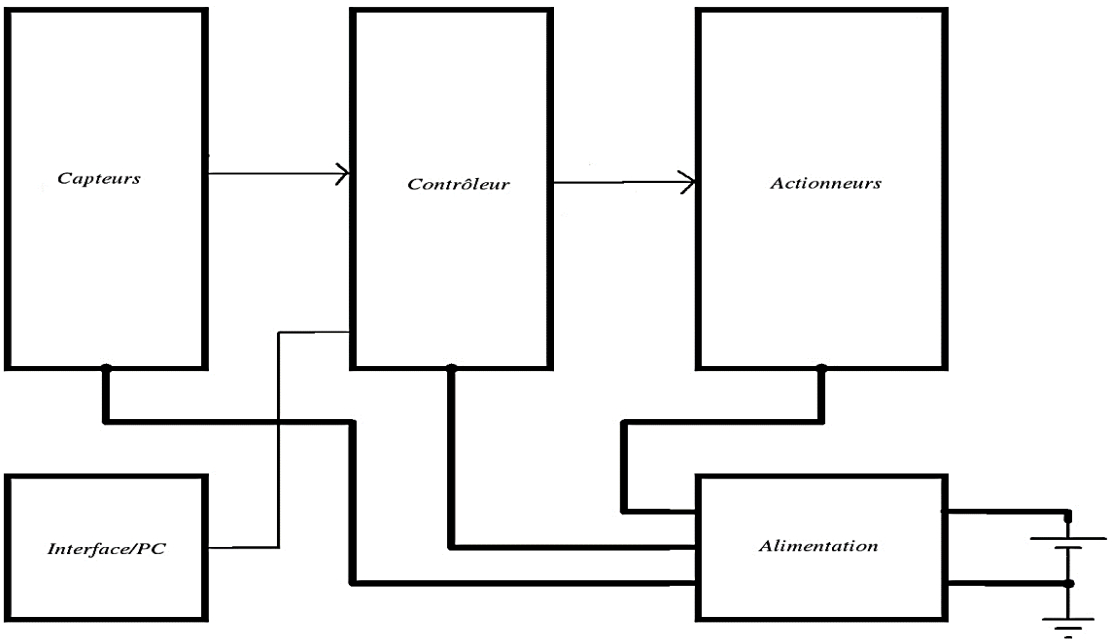


Figure IV.1_Schéma des différentes unités du robot

L'objet de ce chapitre est de décrire tout d'abord les éléments principaux constitutifs de notre projet, la partie matérielle de la commande (le choix du matériel). Ensuite, nous aborderons le logiciel que nous utiliserons pour programmer le robot.

II. Description matériel:

Avant de donner les différentes unites de robot on presenter Notre chassis de robot:

II.1. Kit chassis Robot à 2 roues 2WD:

Figure IV.2_Kit chassis Robot à 2 roues 2WD[27]

Caractéristiques Techniques:

1. Châssis robuste en plastique
2. Deux roues motrices pour une mobilité optimale
3. Supports pour fixer des composants électroniques
4. Facile à assembler sans outils spéciaux
5. Dimensions: 20cm x 15cm x 5cm
6. Alimentation du moteur : 3V-6V.
7. Batterie: 2x AA, 7.4V avec des piles lithium's

II.2. La carte Arduino UNO :

Une carte Arduino est une petite (5,33 x 6,85 cm) carte électronique équipée d'un micro-contrôleur. Le micro-contrôleur permet, à partir d'événements détectés par des capteurs, de programmer et commander des actionneurs ; la carte Arduino est donc une interface programmable **[28]**.

Figure IV.3_Système Arduino Uno [29]

L'Arduino est une plateforme de contrôle, elle est constituée de deux choses:

- **Le logiciel (Software):** gratuit et open source, développé en Java, dont la simplicité d'utilisation relève du savoir cliquer sur la souris.

- **Partie Hardware (Le matériel):** cartes électroniques dont les schémas sonten libre circulation sur internet.

Le système Arduino permet à l'utilisateur de réaliser un grand nombre de projetspuisque l'étendue de l'utilisation de l'Arduino est gigantesque. Voici quelques exemples:

- Contrôler les appareils domestiques.
- Fabriquer son propre robot.
- Faire un jeu de lumières.
- Communiquer avec l'ordinateur.
- Télécommander un appareil mobile.

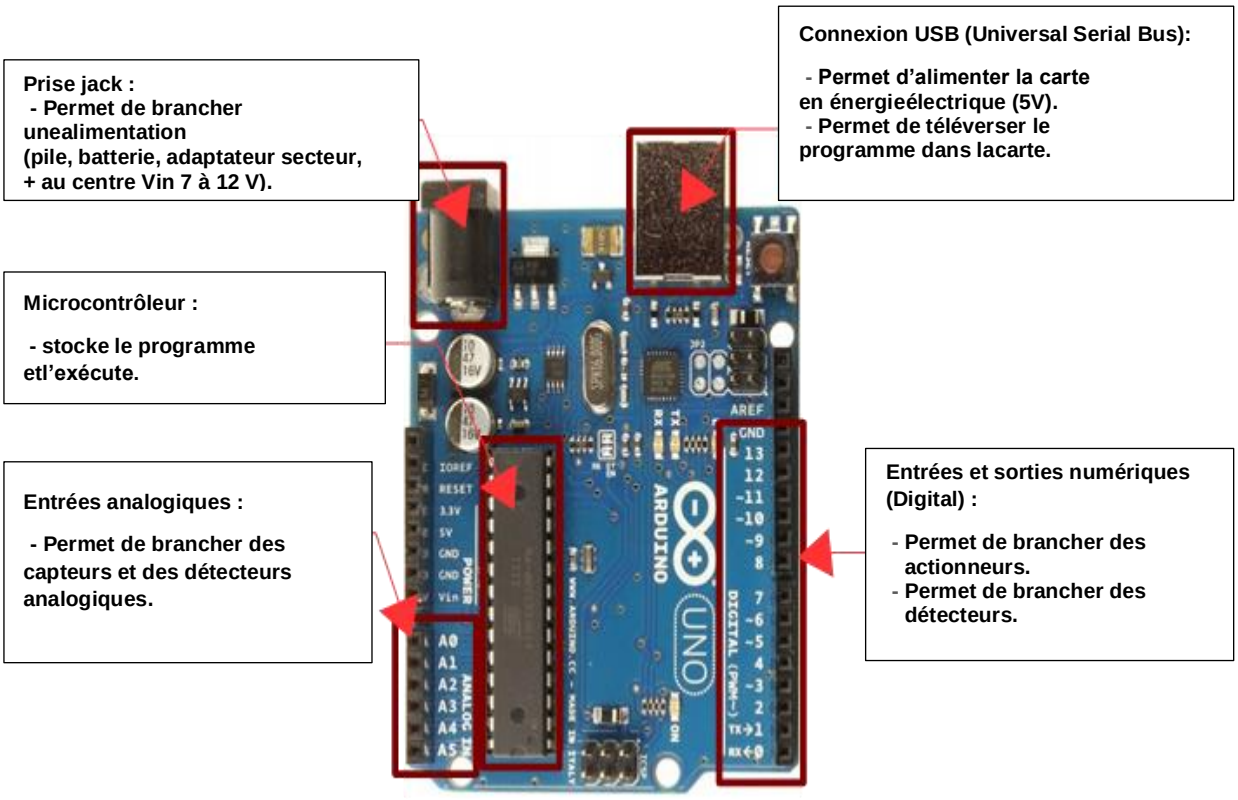


Figure IV.4_Carte Arduino Uno [29]

II.3. Moteurs à courant continu:

Pour notre travail on utilise deux moteurs à courant continu pour commander les roues. Évidemment, pour pouvoir valider un moteur, il faut connaître les spécifications que nous voulons atteindre.

On utilise les moteurs à CC, ils sont facile à utiliser car ils sont programmables depuis une très large gamme de langages de programmation (C#, Java,Python, Labview, Matlab…)

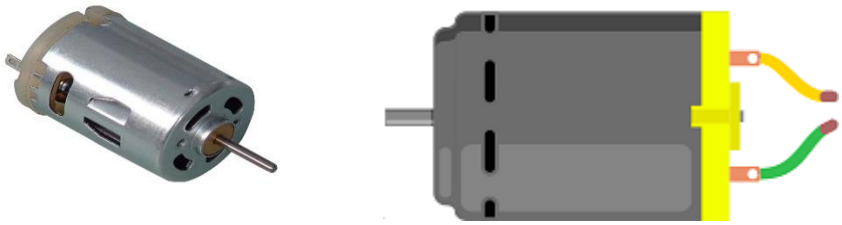


Figure IV.5_Moteur à Courant Continu DC

Un **moteur à courant continu (CC)** de 5V est un type de moteur électrique qui utilise un courant électrique continu de 5V pour produire un mouvement rotatif. Les **moteurs à courant continu de 5V** sont généralement utilisés dans des projets de programmation et de robotique, car ils sont compatibles avec de nombreux microcontrôleurs et cartes de développement, tels que Arduino, qui fonctionnent à des tensions de 5V.

Ce moteur à courant continu DC de type 130 dispose d'une plage de fonctionnement plus large que la plupart de spetits moteurs avec une alimentation entre 4,5V et 9V DC, il sont donc idéal pour être piloté avec un shield moteur, surtout si vous diposez d'une alimentation externe entre 5V et 9V **[30]**.

Caractéristiques :

- Tension de fonctionnement du moteur : 3-6V
- Type de moteur : Moteur à deux axes avec un taux de réduction de 1:48
- Diamètre du pneu : 65 mm
- Vitesse à vide à 3V : 125 m/min, ce qui correspond à une vitesse de pneu de 65 mm : 26 m/min
- Vitesse à vide à 5V : 208 m/min, ce qui correspond à une vitesse de pneu de 65 mm : 44 m/min

II.4. Le Driver L298:

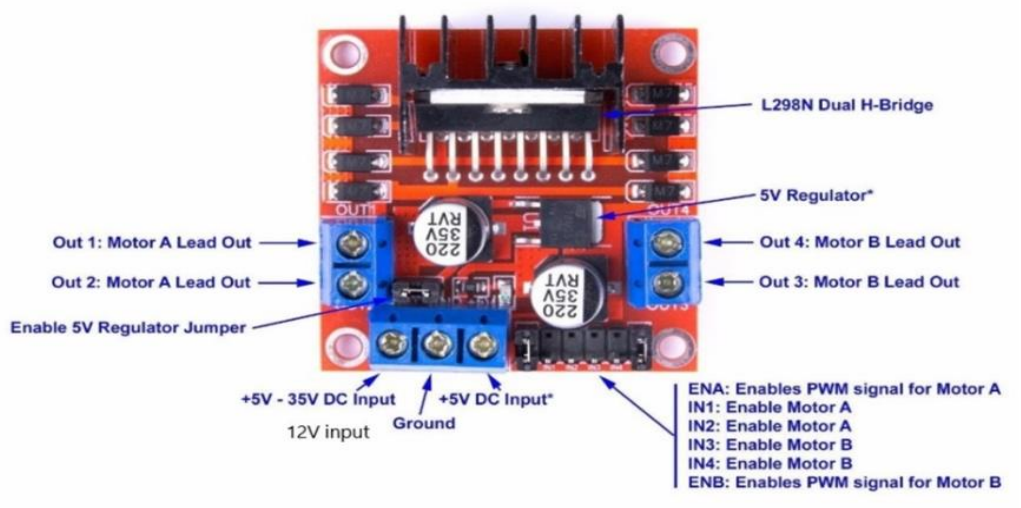


Figure IV.6_Module a base de circuit L298[31]

II.4.1. Description L298xx (H-Bridge Motor Driver):

Le Pont H, est une structure électronique servant à contrôler la polarité aux bornes d'un dipôle. Il est composé de quatre éléments de commutation généralement disposés schématiquement en une forme de H d'où il porte le nom. Les commutateurs peuvent être des relais, des transistors, ou autres éléments de commutation en fonction de l'application visée **[32]**

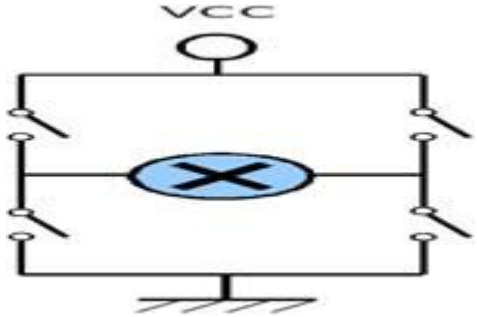


Figure IV.7_Circuit d'un pont en H.

II.4.2. Principe de fonctionnement du pilote (Command du L298N):

Le L298N est un double pont en H, c'est à dire qu'il permet de faire tourner lesmoteurs dans un sens ou dans l'autre sans avoir à modifier les branchements, grâce à sa forme de H, d'où il tient son nom et qui lui permet de faire passer le courant soit dans un sens ou dans l'autre.

Les ports **ENA** et **ENB** permettent de gérer l'amplitude de la tension délivréeau moteur, grâce à un signal PWM. Les ports **In1**, **In2** pour le moteur A et **In3**, **In4** pour le moteur B, permettent de contrôler le pont en H et par conséquent le sens de rotation des moteurs. Comme expliqué sur le tableau suivant :

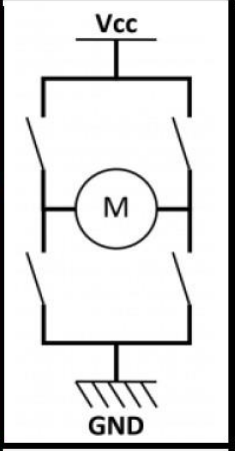
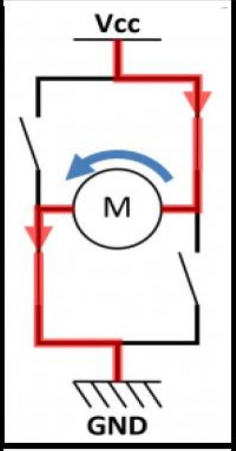
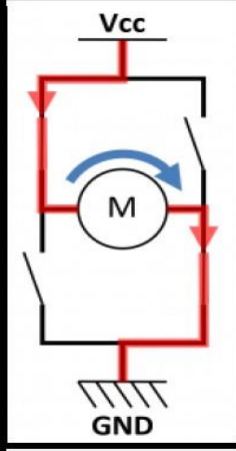
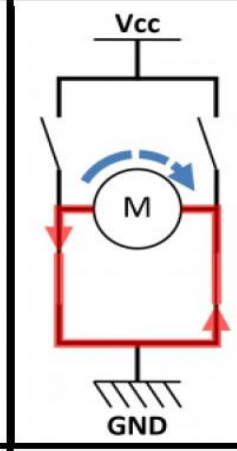
Moteur A	Moteur B	Arrêt (moteur libre)	Sens +	Sens –	Arrêt (moteur freiné)
					
In1	In3	LOW	HIGH	LOW	HIGH
In2	In4	LOW	LOW	HIGH	HIGH

Tableau IV.1_Principe de fonctionnement du pilote (Command du L298N)

II.5. Le capteur de Vitesse LM393

II.5.1. Fonctionnement :

Mesurer la vitesse d'un moteur peut être très utile dans les projets de robotique, et vous pouvez le faire à l'aide du capteur de vitesse de l'encodeur LM393. Avec lui, vous pouvez mesurer la rotation des moteurs, calculer la vitesse des robots, définir des limites de déplacement, entre autres applications.

Ce module peut également être utilisé pour le comptage d'impulsions ou comme interrupteur de fin de course. Il utilise la **puce de comparaison LM393** (fiche technique), et à son extrémité, il dispose d'un opto-interrupteur d'une portée de 5 mm dans lequel vous pouvez utiliser un disque d'encodeur ou un autre dispositif pour interrompre le faisceau lumineux **[33]**.

II.5.2. Structure LM393 Module de capteur de Vitesse :

Le module dispose de 4 broches dont deux sont une alimentation (3 à 5V et GND), une sortie numérique (**DO)** et une analogique (**A0**).

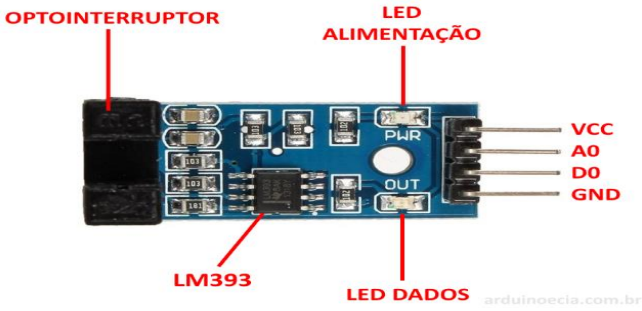


Figure IV.8_Structure LM393 Module de capture de Vitesse

L'optoswitch MOCH22A dispose d'une LED infrarouge d'un côté et d'un phototransistor de l'autre. Lorsque le faisceau lumineux infrarouge est arrêté, la sortie numérique D0 envoie le signal 1, sinon la sortie reste au niveau 0.

II.5.3. Caractéristiques:

- Alimentation: 3,3 à 5 Vcc
- Interface: digitale
- Led d'indication
- Perforation du disque: double méplat Ø5 x 4 mm (voir photo 2)
- Dimensions: 32 x 14 mm

II.5.4. Application:

Pour tester le circuit, nous utilisons un **disque d'encodeur** connecté au moteur. Le disque comporte plusieurs « ouvertures » à travers lesquelles le faisceau de lumière passera et générera une impulsion dans la sortie numérique.

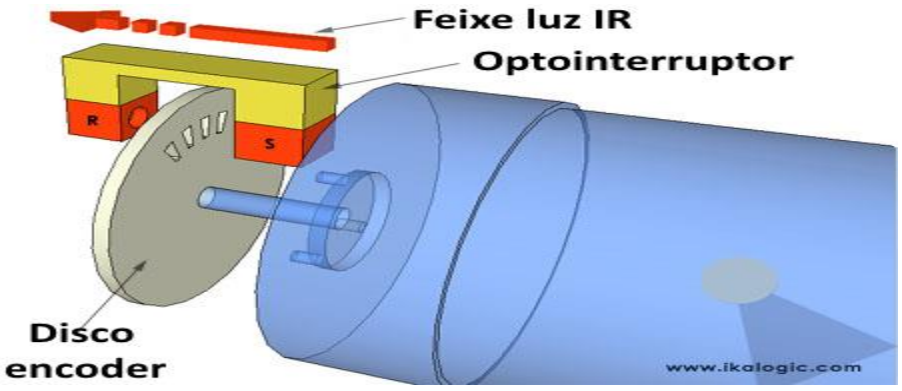


Figure IV.9_le disque d'encodeur brancher au moteur

Le disque que nous utilisons a 20 ouvertures, et ce numéro doit être entré dans le programme, qui utilisera ces informations pour calculer le régime moteur.

III. Branchement globale du système:

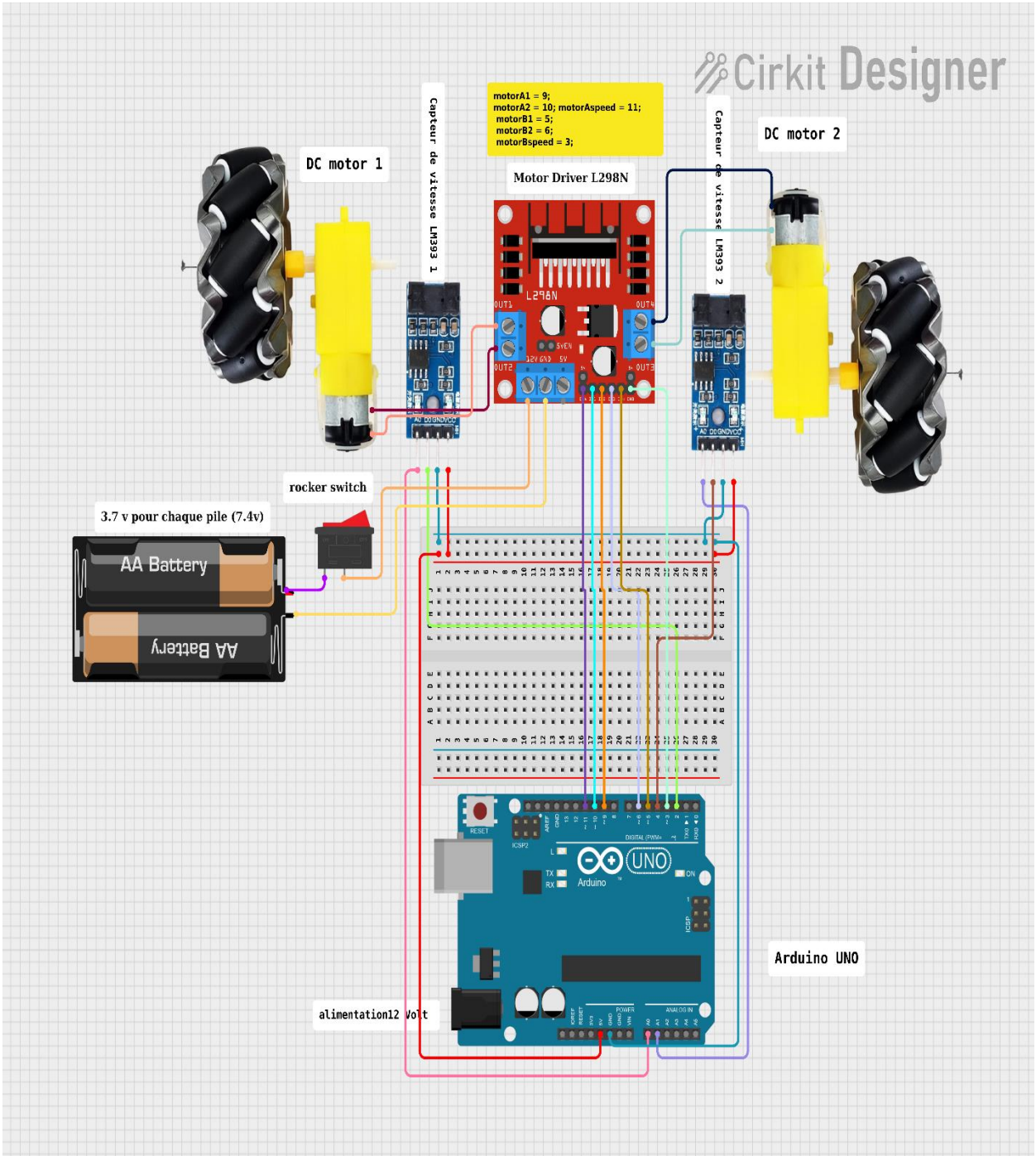


Figure IV.10_Schéma électronique global du robot

IV. La forme finale du robot et le test final:

Le montage final de robot mobile a deux roues réalisées est présenté dans la figure IV.11. Durant la réalisation de projet nous avons rencontré des difficultés mais on a réussi à les surmonter. Notre robot est maintenant prêt à se simuler après avoir était programmé à l'aide d'un logiciel de programmation nommé Arduino IDE, il fonctionne avec les capteurs de vitesse.

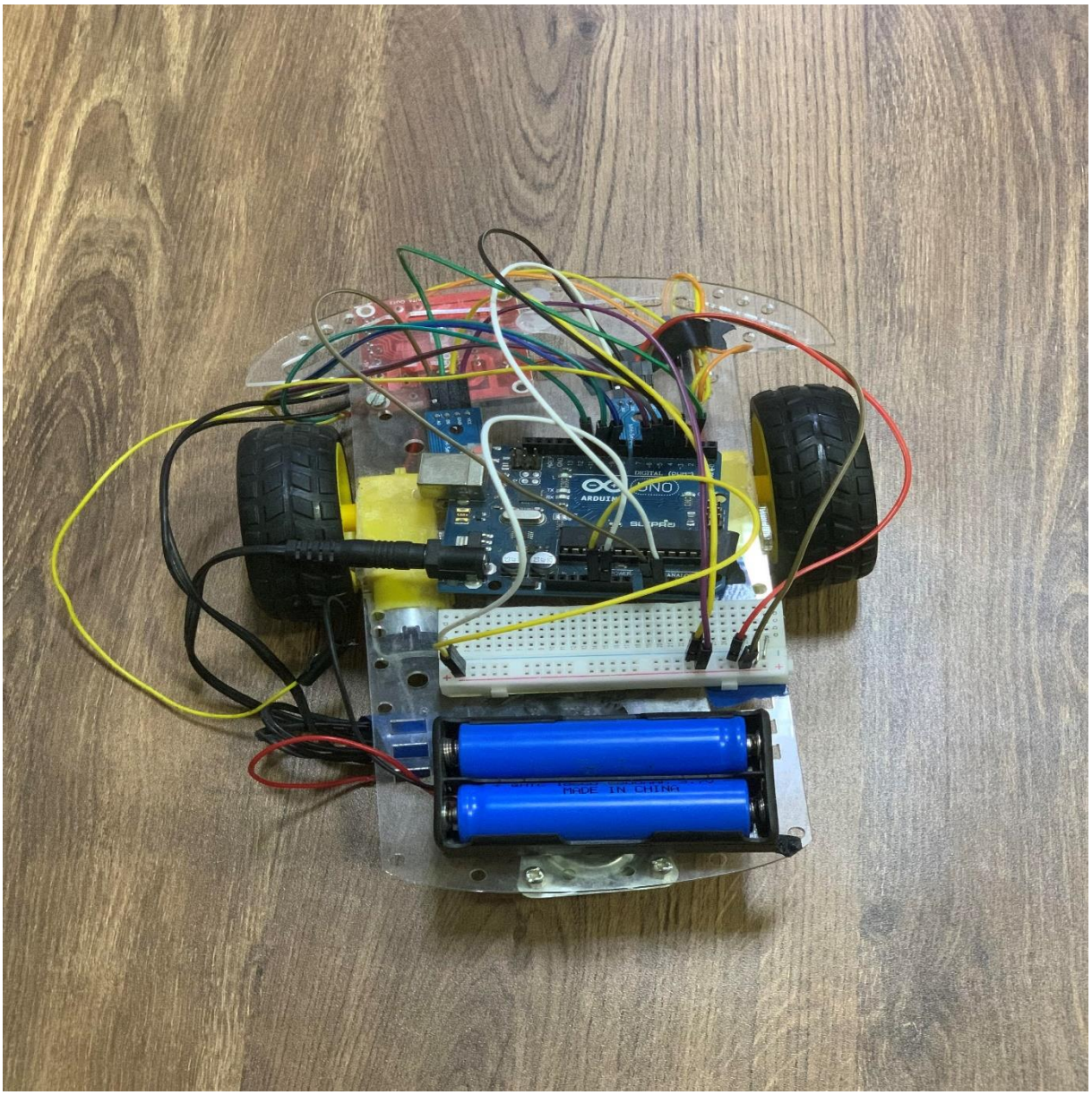


Figure IV.11_La forme finale du robot

Test final :

Apres finalisation du montage, nous avons testé notre robot dans un trajectoire.

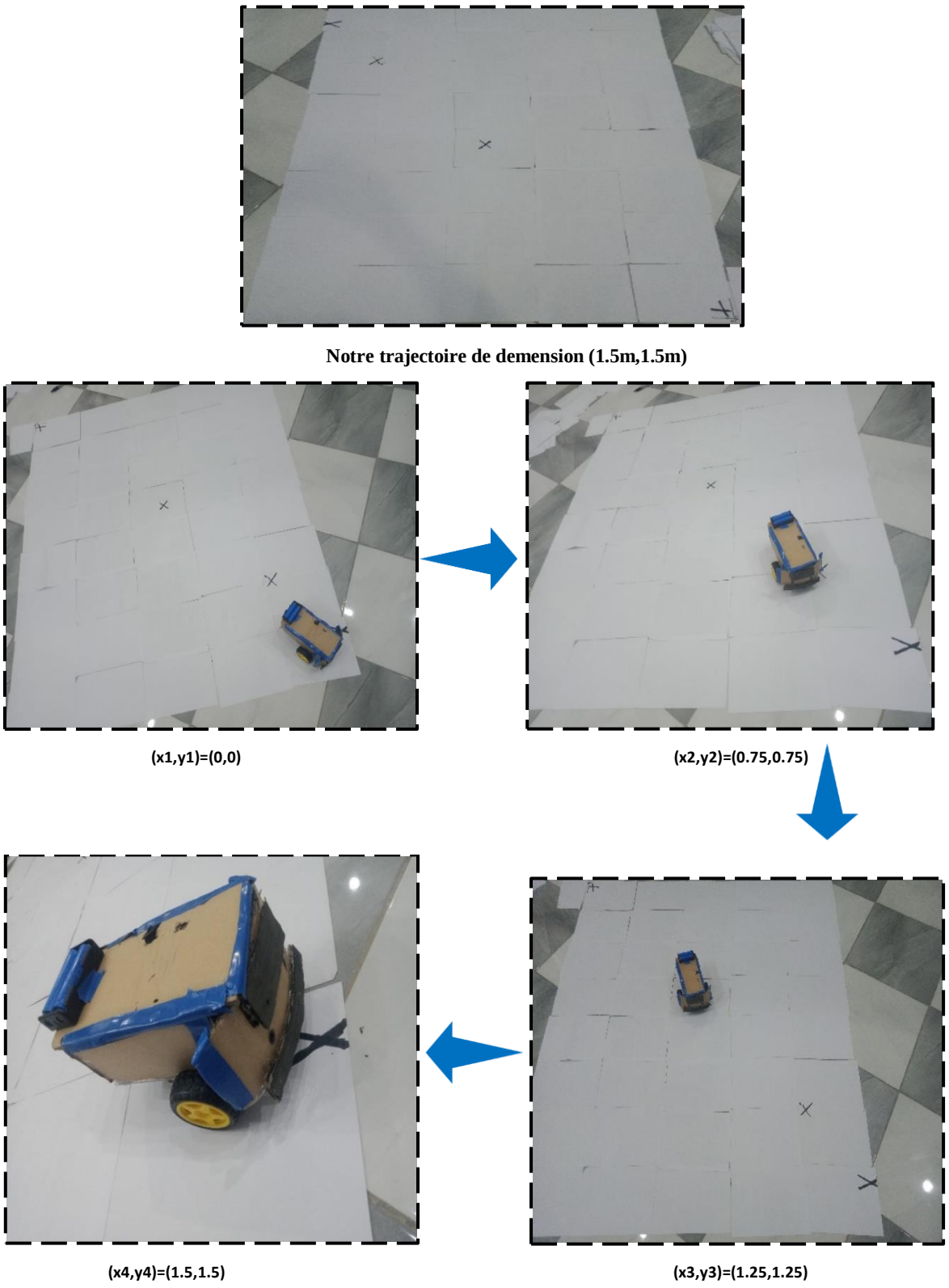


Figure IV.12_Les test final pour un suivre de trajectoire

V. Implémentation de programme :

//Inclusion des bibliothèques standard d'Arduino

```
#include <Arduino.h>
```

//Déclaration des broches

```
// Broches du capteur de vitesse roue droite
const int capteurDroit_A0 = A0;   // Broche analogique du capteur droit (facultatif si non utilisé)
const int capteurDroit_D0 = 2;    // Broche numérique du capteur droit (interruption)

// Broches du capteur de vitesse roue gauche
const int capteurGauche_A0 = A1; // Broche analogique du capteur gauche (facultatif si non utilisé)
const int capteurGauche_D0 = 4;   // Broche numérique du capteur gauche (interruption)

// Broches de commande du pont en H L298N
const int ENA = 3;   // PWM pour vitesse moteur droit
const int IN1 = 5;   // Direction moteur droit
const int IN2 = 6;
const int ENB = 11;  // PWM pour vitesse moteur gauche
const int IN3 = 9;   // Direction moteur gauche
const int IN4 = 10;
```

- ✓ Ces lignes définissent les broches utilisées pour les capteurs de vitesse et les moteurs. Les broches analogiques (A0, A1) sont facultatives et peuvent être utilisées pour des mesures analogiques. Les broches numériques (D0) sont utilisées pour les interruptions.

//Variables de direction et compteur de vitesse

```
volatile int compteurDroite = 0;  // Compteur d'impulsions pour la roue droite
volatile int compteurGauche = 0;  // Compteur d'impulsions pour la roue gauche
```

- ✓ Ces variables comptent les impulsions des capteurs de vitesse, ce qui permet de mesurer la vitesse des roues.

```
//Caractéristiques du capteur et de la roue
const int impulsionsParTour = 20; // Nombre d'impulsions par tour de roue (adapter selon votre capteur)
const float rayonRoue = 0.05;     // Rayon de la roue en mètres (adapter selon votre robot)
```

✓ Ces constantes définissent les caractéristiques physiques des roues et des capteurs.

.....

```
//Modèle du robot

float x = 0.0;  // Position x du robot
float y = 0.0;  // Position y du robot
float theta = 0.0;  // Orientation du robot (en radians)

const float L = 0.14;  // Distance entre les deux roues (en mètres)
const float r = 0.032; // Rayon des roues (en mètres)
```

✓ Ces variables et constantes définissent la position et l'orientation du robot ainsi que les dimensions physiques du robot.

.....

```
//Commande du robot

float x_d = 5.0;  // Position cible en x
float y_d = 5.0;  // Position cible en y

// Gains proportionnels pour la vitesse linéaire et angulaire
const float Kp_linear = 1.0;   // Gain proportionnel pour la distance
const float Kp_angular = 2.0;  // Gain proportionnel pour l'orientation

const float vitesseMax = 255;  // Valeur PWM maximale pour les moteurs (0-255)
```

✓ Ces variables définissent la position cible du robot et les gains proportionnels pour le contrôle de la vitesse linéaire et angulaire.

```
//Planification des points de la trajectoire

float pointsX[] = {0.0, 2.0, 4.0, 5.0};  // x0, x1, x2, xf obtenus par TLBO
float pointsY[] = {0.0, 3.0, 2.0, 5.0};  // y0, y1, y2, yf obtenus par TLBO
```

```
// Temps associés à chaque point
float tempsPoints[] = {0.0, 3.0, 6.0, 10.0};  // t0, t1, t2, tf obtenus par
TLBO
```

✓ Ces tableaux définissent les points de la trajectoire que le robot doit suivre, ainsi que les temps associés à chaque point.

.....

//Temps de la dernière lecture

```
unsigned long dernierTemps = 0;   // Temps de la dernière mesure
```

✓ Cette variable stocke le temps de la dernière mesure pour calculer les intervalles de temps.

.....

//Fonction setup

```
void setup() {
  // Initialisation des broches
  pinMode(capteurDroit_D0, INPUT);
  pinMode(capteurGauche_D0, INPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Initialisation de la communication série
  Serial.begin(9600);

  // Configurer les interruptions pour compter les impulsions
  attachInterrupt(digitalPinToInterrupt(capteurDroit_D0),
incrementerCompteurDroite, RISING);
  attachInterrupt(digitalPinToInterrupt(capteurGauche_D0),
incrementerCompteurGauche, RISING);

  // Temps initial
  dernierTemps = millis();
}
```

✓ La fonction setup initialise les broches, configure la communication série et les interruptions pour compter les impulsions des capteurs de vitesse.

.....

```
//Fonction loop

void loop() {
  unsigned long tempsActuel = millis();  // Temps actuel en millisecondes
  float t = (float)(tempsActuel - debutTemps) / 1000.0;  // Temps en secondes
(t0 = 0s)
```

Explication :

- `millis()` retourne le temps écoulé depuis le démarrage du programme en millisecondes.
- `t` est le temps actuel en secondes depuis le début du programme.

Formule :

$$t = \frac{\text{tempsActuel} - \text{debutTemps}}{1000.0}$$

```
// Limiter t à la durée totale de la trajectoire (tf)
  if (t > 10.0) t = 10.0;
```

Explication :

- Cette condition limite `t` à 10 secondes, la durée totale de la trajectoire.

Formule :

$$t = \min(t, 10.0)$$

```
// Obtenir la position cible (x, y) à l'instant t via la spline cubique
   float x_cible, y_cible;
  getPositionSpline(t, x_cible, y_cible);
```

Explication :

- getPositionSpline(t, x_cible, y_cible) calcule la position cible (x_cible, y_cible) à l'instant t en utilisant une spline cubique.

Formule :

$$(x_{\text{cible}}, y_{\text{cible}}) = \text{getPositionSpline}(t)$$

.....

```
  // Mise à jour de la position actuelle du robot (cinématique)
  unsigned long deltaTemps = tempsActuel - dernierTemps;  // Temps écoulé depuis la dernière lecture (en ms)
  if (deltaTemps >= 1000) {  // Mettre à jour toutes les secondes (1000 ms)
```

Explication :

- deltaTemps est le temps écoulé depuis la dernière mise à jour.
- La mise à jour se fait toutes les secondes (1000 ms).

Formule :

$$\Delta t = \text{tempsActuel} - \text{dernierTemps}$$

```
// Calcul des vitesses angulaires pour les deux roues
float frequenceDroite = (float)compteurDroite / (deltaTemps / 1000.0);
float frequenceGauche = (float)compteurGauche / (deltaTemps / 1000.0);
float omegaDroite = (2 * PI * frequenceDroite) / impulsionsParTour;
float omegaGauche = (2 * PI * frequenceGauche) / impulsionsParTour;
```

Explication :

- frequenceDroite et frequenceGauche sont les fréquences des impulsions des roues droite et gauche.

omegaDroite et omegaGauche sont les vitesses angulaires des roues.

$$\text{frequenceDroite} = \frac{\text{compteurDroite}}{\Delta t/1000.0}$$

$$\text{frequenceGauche} = \frac{\text{compteurGauche}}{\Delta t/1000.0}$$

$$\omega_{\text{Droite}} = \frac{2\pi \times \text{frequenceDroite}}{\text{impulsionsParTour}}$$

$$\omega_{\text{Gauche}} = \frac{2\pi \times \text{frequenceGauche}}{\text{impulsionsParTour}}$$

```
// Mise à jour de la position et orientation du robot
miseAJourCinematique(omegaGauche, omegaDroite, deltaTemps / 1000.0);
```

Explication :

- miseAJourCinematique met à jour la position (x, y) et l'orientation theta du robot en utilisant les vitesses angulaires et le temps écoulé.

Formules :

- Vitesse linéaire moyenne :

$$v = \frac{r}{2}(\omega_{\text{Gauche}} + \omega_{\text{Droite}})$$

- Vitesse angulaire du robot :

$$\omega = \frac{r}{L}(\omega_{\text{Droite}} - \omega_{\text{Gauche}})$$

- Mise à jour de la position et orientation (intégration d'Euler) :

$$x = x + v\cos(\theta)\Delta t$$

$$y = y + v\sin(\theta)\Delta t$$

$$\theta = \theta + \omega\Delta t$$

```
// Réinitialiser les compteurs d'impulsions
 compteurDroite = 0;
compteurGauche = 0;
dernierTemps = tempsActuel;
```

Explication :

- Les compteurs d'impulsions sont réinitialisés pour la prochaine mesure.
- dernierTemps est mis à jour avec le temps actuel.

```
// Afficher la position et orientation actuelles
    Serial.print("Position actuelle (x, y, theta): (");
    Serial.print(x);
    Serial.print(", ");
    Serial.print(y);
    Serial.print(", ");
    Serial.println(theta);
```

Explication :

- La position actuelle (x, y) et l'orientation theta du robot sont affichées sur le moniteur série.

.....

```
  // Appeler la fonction de contrôle pour suivre la position cible

  controleVersCible(x_cible, y_cible);
```

Explication :

- Cette fonction ajuste les commandes des moteurs pour suivre la position cible (x_cible, y_cible).

Détails Mathématiques :

Calcul de l'erreur de position :

$$erreurX = xcible - x$$

$$erreurY = ycible - y$$

Calcul de la distance à la cible :

$$\text{distance} = \sqrt{\text{erreurX}^2 + \text{erreurY}^2}$$

Calcul de l'angle désiré vers la cible :

$$\text{angleDesire} = \arctan\left(\frac{\text{erreurY}}{\text{erreurX}}\right)$$

Calcul de l'erreur d'orientation :

$$\text{erreurTheta} = \text{angleDesire} - \theta$$

Normalisation de l'erreur d'orientation :

$$\text{erreurTheta} = \text{erreurTheta} \bmod 2\pi$$

- Si erreurTheta > π, alors erreurTheta -= 2π.
- Si erreurTheta < -π, alors erreurTheta += 2π.

```
// Pause légère pour éviter de saturer la boucle
  delay(100);
```

Explication :

- delay(100) ajoute une pause de 100 ms pour éviter de saturer la boucle loop

}

```
// Fonction d'interruption pour la roue droite
void incrementerCompteurDroite() {
  compteurDroite++;  // Incrémenter le compteur d'impulsions de la roue droite
}

// Fonction d'interruption pour la roue gauche
void incrementerCompteurGauche() {
  compteurGauche++;  // Incrémenter le compteur d'impulsions de la roue gauche
}
```

Explication :

- Ces fonctions sont appelées à chaque impulsion des capteurs de vitesse des roues, incrémentant les compteurs d'impulsions

```
// Fonction pour commander le moteur droit
void commandeMoteurDroit(int pwm) {
  // Exemple simple de commande de moteur
  digitalWrite(IN1, HIGH);  // Avancer
  digitalWrite(IN2, LOW);
  analogWrite(ENA, pwm);    // Ajuster la vitesse avec PWM
}

// Fonction pour commander le moteur gauche
void commandeMoteurGauche(int pwm) {

// Exemple simple de commande de moteur
  digitalWrite(IN3, HIGH);  // Avancer
```

```
  digitalWrite(IN4, LOW);
  analogWrite(ENB, pwm);    // Ajuster la vitesse avec PWM
}
```

Explication :

- Ces fonctions contrôlent les moteurs droit et gauche en ajustant la direction et la vitesse via PWM.

VI. Conclusion:

Ce chapitre était le noyau de notre projet fin d'étude, nous avons présenté dans la premières partie les différents composants de chaque unité qui constitue le robot à réaliser. Nous avons aussi présenté le fonctionnement et le branchement du circuit global du robot.

Et dans la deuxième partie qui a été consacré pour la partie programmation et réalisation finale de robot mobile.

Conclusion

Ce projet de fin d'études a été un projet ambitieux, couvrant plusieurs disciplines et nécessitant des compétences multidisciplinaires. Il nous a permis de mettre en pratique les connaissances acquises tout au long de notre formation et de concrétiser nos compétences en construisant et en contrôlant un robot mobile. De plus, il a enrichi notre compréhension de la robotique mobile ainsi que de la programmation embarquée.

Au cours de ce projet, nous avons exploré la puissance de la carte Arduino UNO, qui a servi de cœur au robot. La programmation de cette carte a permis de commander deux moteurs à courant continu, rendant le robot capable de se déplacer de manière autonome dans son environnement. L'intégration de capteurs de vitesse a renforcé l'autonomie du robot, lui permettant de planifier sa trajectoire et d'éviter les obstacles sans intervention humaine.

Le mémoire a également exploré l'utilisation des algorithmes d'optimisation basés sur la méthode d'enseignement-apprentissage (TLBO) pour la planification de trajectoire d'un robot mobile à deux roues. Les quatre chapitres du mémoire ont couvert des aspects essentiels de la robotique mobile. Le premier chapitre a détaillé la modélisation cinématique du robot, basée sur les équations de mouvement et les principes de la cinématique. Le deuxième chapitre a porté sur les méthodes de suivi de trajectoire, en mettant l'accent sur les techniques adaptées aux robots mobiles et la gestion des obstacles. Le troisième chapitre a examiné et adapté l'algorithme TLBO pour l'optimisation des trajectoires, prouvant son efficacité dans ce contexte. Enfin, le quatrième chapitre a décrit les expérimentations pratiques, combinant simulations et tests matériels pour valider les solutions proposées.

Les résultats obtenus montrent que l'algorithme TLBO peut améliorer de manière significative la planification de trajectoire, ouvrant ainsi la voie à des applications plus avancées et robustes dans le domaine de la robotique mobile. Les compétences acquises à travers ce projet, allant de la modélisation théorique à la validation expérimentale, constituent une base solide pour des recherches futures ainsi que des applications industrielles.

Perspectives

- **Intégration de capteurs supplémentaires :** Ajouter des LIDARs ou des caméras pour obtenir des données environnementales plus précises, améliorant ainsi la planification de trajectoire.
- **Optimisation multi-objectifs:** Explorer des algorithmes prenant en compte plusieurs critères de performance, tels que la consommation d'énergie et le temps de parcours.
- **Application dans des environnements dynamiques:** Adapter les algorithmes TLBO pour fonctionner dans des environnements où les obstacles et les cibles sont en mouvement.
- **Implémentation sur des plateformes robotiques réelles:** Tester les algorithmes en dehors des simulations pour évaluer leur robustesse et efficacité dans des conditions réelles.

En conclusion, ce mémoire a démontré le potentiel des algorithmes TLBO pour la planification de trajectoire des robots mobiles à deux roues. Les perspectives de recherche identifiées offrent de nombreuses opportunités pour approfondir et étendre ces travaux, contribuant ainsi à l'avancement de la robotique mobile.

Bibliographie :

[1] Bouali, A. (2012). Planification de trajectoire pour un robot mobile (Doctoral dissertation, Université de Batna 2).

[2] B. Bayle, *"Robotique mobile "Cours option ISAV, Télécom Physique,* Strasbourg 3A., pp. Cours option ISAV, Télécom Physique.

[3] J. P. Laumond, La robotique mobile, Paris,: Edition Hermes Science Publication, 2001.

[4] N. C. e. M. Quigley, Introduction to Autonomous Robots.

[5] N. Nilsson, *"The Quest for Artificial Intelligence: A History of Ideas and Achievements,",* Cambridge University Press, 2009.

[6] Apriaskar, E., Fahmizal, F., Cahyani, I., & Mayub, A. (2020). Autonomous Mobile Robot based on BehaviourBased Robotic using V-REP Simulator–Pioneer P3-DX Robot. Jurnal Rekayasa Elektrika, 16(1)

[7] H. A. Dhaouadi R, «Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified framework,» college of Engineering, American University of Sharjah, UAE, sahrjah, 2013.

[8] B. J. El Mouderrib Iliasse, «Commande cinématique en vitesse/Cap d'un robot unicycle, Mouderrib Iliasse, Belmiloud Jedjiga».

[9] https://globaljournals.org/GJRE_Volume14/1-Kinematics-Localization-and-Control.pdf, s.d.

[10] Wang, S., Yin, X., Li, P., Zhang, M., & Wang, X. (2020). Trajectory tracking control for mobile robots using reinforcement learning and PID. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 44, 1059-1068

[11] Smith, J., & Doe, A. (2021). Importance of trajectory tracking in autonomous robots. *Journal of Robotics and Automation*, 35(4), 567-578.

[12] C. G. Faïz BenAmar, « Mobile robotics : design, modeling and control, » 2016.

[13] J. Courbon, Y. Mezouar, L. Eck, P. Martinet, "*A generic framework for topological navigation of urban vehicle*", ICRA09 Workshop on Safe navigation in open and dynamic environments Application to autonomous vehicles, ICRA09, Kobe, Japan, May 12th, 2009

[14] Pradalier, C., Hermosillo, J., Koike, C., Braillon, D., Bessière, P., Laugier, C. *"The CyCab: a car-like robot navigating autonomously and safely among "*.Robotics and Autonomous Systems (RAS) 50(1):51-67 (2005)

[15] S. YAHIAOUI, S. MERROUCHI. « Planification de mouvement d'un robot mobile » Mémoire d'ingénieur d'état en électronique, Université de Batna Juin 1997.

[16] B. Atmani, "Optimisation des paramètres du Contrôleur PID par Algorithme Génétique Multi-objectifs", Mémoire de Master, Université Abderrahmane Mira de Béjaïa, 2011.

[17] Hachimi, H. (2013). *Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications*. Maroc, INSA de Rouen, École Mohammadia d'ingénieurs Rabat.

[18] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.

[19] Boussaid, I. (2013). *Perfectionnement de métaheuristiques pour l'optimisation continue*. Université Paris Est, France.

[20] Wikimedia Commons. (n.d.). *Optimisation difficile* [Image]. Retrieved from https://commons.wikimedia.org/wiki/File:Optimisation_difficile.svg?uselang=fr

[21] Bourahla, K. (2019). *Adaptation des méthodes d'optimisation par intelligence artificielle pour la conception optimale des machines électriques* (Thèse de Doctorat). Université Mohammed Seddik, Algérie.

[22] Venkata Rao, R. (n.d.). *Teaching-Learning-Based Optimization (TLBO) algorithm and its codes*. S.V. National Institute of Technology.

[23] Venkata Rao, R., & Patel, V. (2012). Department of Mechanical Engineering, S.V. National Institute of Technology, Ichchanath, Surat, Gujarat – 395 007, India. *Received 11 June 2012; revised 16 August 2012; accepted 9 October 2012*.

[24] Chen, X., Xu, B., Mei, C., Ding, Y., & Li, K. (2018). Teaching–learning–based artificial bee colony for solar photovoltaic parameter estimation. *Applied Energy, 212*, 1578-1588.

[25] Jamil, M., & Yang, X. S. (2013). A Literature Survey of Benchmark Functions For Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150-194.

[26] Chen, X., Xu, B., Yu, K., & Du, W. (2018). Teaching-Learning-Based Optimization with Learning Enthusiasm Mechanism and Its Application in Chemical Engineering. *Journal of Applied Mathematics*, 1-15.

[27] Gotronic. (n.d.). **Châssis à 2 roues motrices**. Retrieved September 18, 2024, from https://www.gotronic.fr/cat-chassis-a-2-roues-motrices-1268.htm

[28] Arduino. (n.d.). **UNO R3**. Arduino Documentation. Retrieved September 18, 2024, from https://docs.arduino.cc/hardware/uno-rev3/

[29] Google Images. (n.d.). **Image**. Retrieved September 18, 2024, from https://images.app.goo.gl/rjQE2L25Ag59ud6N6

[30] SparkFun Electronics. (n.d.). **5V DC Motor - 130 Size**. SparkFun Documentation. Retrieved September 18, 2024, from https://www.sparkfun.com/products/11696

[31] Google Images. (n.d.). **Image**. Retrieved September 18, 2024, from https://images.app.goo.gl/VQ4aCuphygStnds5A

[32] STMicroelectronics. (n.d.). **L298 Dual Full-Bridge Driver**. STMicroelectronics Documentation. Retrieved September 18, 2024, from https://www.st.com/resource/en/datasheet/cd00000240.pdf

[33] Electropeak. (n.d.). **Interfacing LM393 Infrared Speed Sensor with Arduino**. Electropeak Documentation. Retrieved September 18, 2024, from https://electropeak.com/learn/interfacing-lm393-infrared-speed-sensor-with-arduino/

ملخص:

تم إنجاز هذا البحث في إطار دراسة وتحقيق روبوت متنقل ذو عجلتين يتمتع بنظام ملاحة ذاتي باستخدام خوارزميات (التعليم والتعلم المستند إلى التحسين) بهدف تحسين TLBO مستوحاة من الطبيعة لتخطيط المسار. تم اختيار خوارزمية متابعة المسار المحدد مسبقا.

أتاحت الدراسة الحركية للروبوت تحديد العلاقات بين الموقع والسرعة مع العجلتين، وذلك لضمان التحكم الدقيق فيهما. لنمذجة وتخطيط مسارات الروبوت مع مراعاة القيود البيئية. يتضمن MATLAB كما تم إجراء محاكاة باستخدام برنامج هذا العمل أيضًا مرحلة تنفيذ عملي للتحقق من صحة النتائج التي تم الحصول عليها من خلال المحاكاة في ظروف واقعية.

الكلمات المفتاحية: روبوت متنقل ذو عجلتين، روبوت وحيد العجلة، الملاحة الذاتية، تطوير الروبوت المتنقل، MATLAB, TLBO.

Résumé :

Ce mémoire est réalisé dans le cadre de l'étude et de la réalisation d'un robot mobile à deux roues avec navigation autonome en utilisant des algorithmes bio-inspiré pour la planification de trajectoire. L'algorithme TLBO Teaching-Learning-Based Optimization a été choisi pour viser l'optimisation de suivi de la trajectoire prédéfinie.

L'étude cinématique du robot a permis d'établir les relations entre la position, la vitesse avec les deux roues, afin de garantir le contrôle des deux roues. Une simulation a été réalisée sous MATLAB pour modéliser et planifier les trajectoires du robot en prenant en considération des contraintes environnementales. Ce travail inclut également une phase de réalisation pratique, permettant de valider les résultats obtenus par simulation dans des conditions réelles.

Mots clés: robot mobile à deux roues, unicycle, navigation autonome, réalisation de robot mobile, MATLAB, TLBO.

Abstract :

This thésis focuses on the study and development of a two-wheeled mobile robot with autonomous navigation using bio-inspired algorithms for trajectory planning. The TLBO (Teaching-Learning-Based Optimization) algorithm was selected to optimize the tracking of a predefined path.

The kinematic analysis of the robot helped establish the relationships between position and speed with both wheels, ensuring precise control. A simulation was conducted using MATLAB to model and plan the robot's trajectories while considering environmental constraints. This work also includes a practical implementation phase to validate the simulation results under real-world conditions.

Keywords: two-wheeled mobile robot, unicycle, autonomous navigation, mobile robot development, MATLAB, TLBO.