ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Section Microtechnique 8e semestre

Projet de Semestre

Suivi de chemin par un robot

Damien PERRITAZ

Laboratoire d'Automatique

Semestre d'été 2005

Supervision du projet
MER : Denis Gillet
Assistant : Christophe Salzmann

Table des matières

Table des figures

Résumé

Le but de ce projet était de réguler un robot sur un chemin. Le robot et son système informatique étaient donnés.

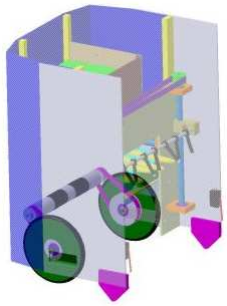

Fig. 1 – Le robot

Le robot est régulé en vitesse grâce à une méthode d'état. Un régulateur classique avec intégrateur a été synthétisé.

Le robot est régulé sur un chemin selon la loi de commande de C. Samson. Il s'agit d'une méthode très répandue pour le suivi de chemin, qui prend comme paramètres d'écarts au chemin : la distance à la droite et la différence entre l'angle du robot et l'angle de la droite.

Ces deux régulateurs en cascade permettent de suivre un chemin constitué de suite de droites. L'anticipation se fait grâce à la commande prédictive prenant l'angle des droites comme variable manipulée.

La modélisation du système a permis de faire une bonne simulation. Cette simulation était indispensable pour trouver les bons paramètres et sentir leur effet.

L'implémentation de l'algorithme sur le robot avec un système informatique biprocesseur a montré l'efficacité de l'approche choisie, après avoir résolu les problèmes liés aux différences entre la théorie et la pratique.

Les résultats sont très concluants et ce concept de régulation a été adopté pour la participation à un concours de robotique autonome.

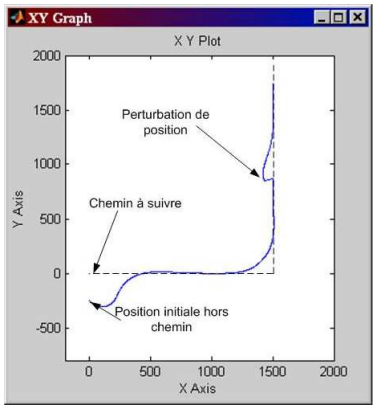

Fig. 2 – Résultats de simulation

Chapitre 1

Introduction

Dans le vaste domaine de la robotique mobile, l'étude du déplacement a une grande importance. De nombreuses approches du déplacement existent en utilisant différents types de capteurs.

Les robots mobiles, connaissant leur position, se déplacent d'un point à un autre avec diverses contraintes. Dans certains cas, seule l'atteinte du point final est important, sans se préoccuper du trajet. Dans d'autres cas, le trajet parcouru est important tout comme sa dépendance du temps ; la trajectoire à suivre par le robot doit être définie.

Mais généralement, seul le chemin à suivre est défini. Le robot doit atteindre le point final selon une route indépendante du temps. C'est le thème qui va nous intéresser dans ce rapport.

Motivations

Le besoin de faire suivre un chemin à un robot a été défini dans le cadre de l'équipe de robotique Team-ID pour l'implémentation sur un robot participant au concours SwissEurobot.

Le sujet de ce projet est dans un premier temps l'étude des diverses méthodes existantes pour la régulation d'un robot sur un chemin. Il faudra ensuite choisir et adaptée une méthode adéquate à l'application et enfin l'implémenter sur le robot existant en prenant en compte des contraintes physiques et informatiques, entre autres.

Contenu du rapport

Dans ce rapport, je présente le robot qui devra être régulé, ainsi que le chemin type qu'il sera amené à suivre.

Le gros chapitre de la régulation présente le concept retenu. Il se divise ensuite en deux parties principales :

1. la régulation en vitesse du robot
2. la régulation du robot sur le chemin

Il se termine par la présentation du schéma fonctionnel du système et par l'étude des résultats de la simulation.

L'implémentation pratique est ensuite expliquée, en commençant par introduire le système informatique.

Les problèmes liés aux différences entre la théorie et la pratique seront exposés, tout commme les solutions trouvées.

Enfin, les propositions d'amélioration et les perspectives possibles concluront le travail.

Chapitre 2

Le robot

Le robot sur lequel sera implémenté le suivi de chemin a été conçu lors d'un précédent projet de semestre au Laboratoire de Systèmes Autonomes[1] pour la participation à SwissEurobot'05. La conception mécanique, le choix des capteurs et actionneurs doivent être pris comme acquis. La construction du robot a été terminée au courant du mois d'avril, le robot n'était donc pas utilisable durant la première partie du semestre.

Ce robot différentiel non-holonome, se déplace à l'aide de deux roues motrices indépendantes. Le couple est transmis par les moteurs d'une puissance de 23 W grâce à des courroies crantées. Les moteurs sont équipés d'encodeurs magnétiques pour donner leur position angulaire.

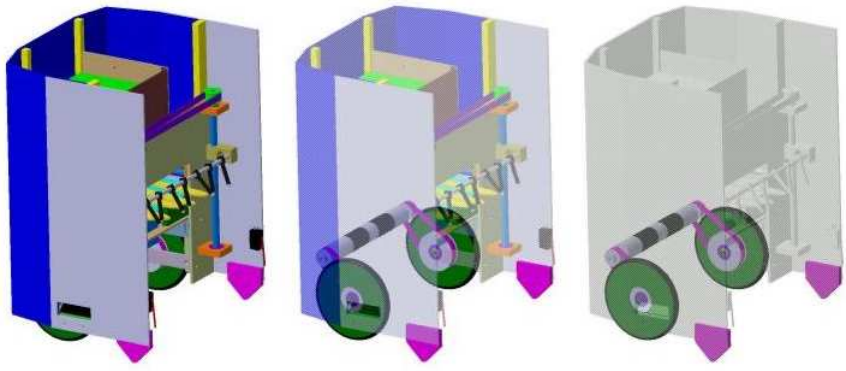

FIG. 2.1 – Vue générale du robot

Le robot a les dimensions proches d'une feuille A4 (largeur entre les roues : 240 mm) pour une masse d'environ 15 kg. L'inertie est évidemment

[1] Plus d'informations sont disponible en annexe dans le rapport du projet concerné

non négligeable. Le centre de gravité est plus haut et plus avancé que l'axe des roues : le troisième point d'appui est un frotteur situé à l'avant du robot.

La vitesse maximale du robot est de 80 $\frac{cm}{s}$, mais cette vitesse ne devrait vraisemblablement pas être nécessaire.

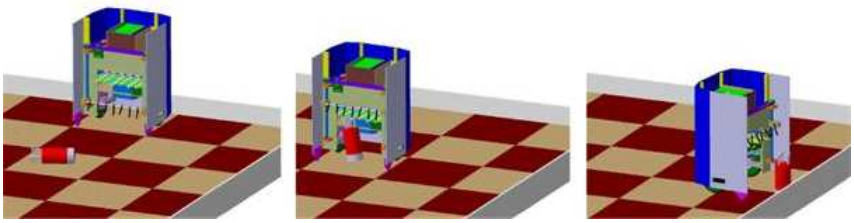

FIG. 2.2 – Action de jeu du robot

2.1 Modèle du robot

2.1.1 Modèle cinématique

Voici le modèle cinématique du robot, selon la contrainte non-holonomique 2.1 due au roulement sans glisser des roues.

$$\dot{y} \cdot cos(\theta) - \dot{x} \cdot sin(\theta) = 0 \tag{2.1}$$

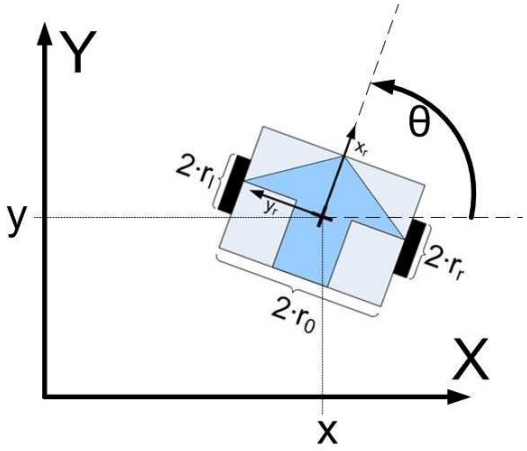

FIG. 2.3 – Modèle du robot

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{r_r}{2}cos(\theta) & \frac{r_l}{2}cos(\theta) \\ \frac{r_r}{2}sin(\theta) & \frac{r_l}{2}sin(\theta) \\ \frac{r_r}{2r_0} & \frac{r_r}{2r_0} \end{pmatrix} \cdot \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{pmatrix} \tag{2.2}$$

avec :
$\dot{\theta}_r$(t) : vitesse angulaire de la roue droite
$\dot{\theta}_l$(t) : vitesse angulaire de la roue gauche

2.1.2 Modèle odométrique

En discrétisant le modèle cinématique et en utilisant les approximations habituelles, le modèle odométrique devient[2] :

$$\theta(k) = \frac{r(\theta_r(k) - \theta_l(k))}{2r_0} \tag{2.3}$$

$$x(k) = x(k-1) + \frac{r(\Delta\theta_r + \Delta\theta_l)}{2} \cdot cos(\frac{\theta(k) - \theta(k-1)}{2}) \tag{2.4}$$

$$y(k) = y(k-1) + \frac{r(\Delta\theta_r + \Delta\theta_l)}{2} \cdot sin(\frac{\theta(k) - \theta(k-1)}{2}) \tag{2.5}$$

avec :

$$\Delta\theta_r = \theta_r(k) - \theta_r(k-1) \tag{2.6}$$

$$\Delta\theta_l = \theta_l(k) - \theta_l(k-1) \tag{2.7}$$

2.2 Le chemin à suivre

Ce robot doit se déplacer sur le terrain de jeu avec des routes simples mais efficaces, qui ne nécessitent pas d'être dépendantes du temps : il s'agit donc de chemin et non de trajectoire. A cause de la configuration du terrain (obstacles fixes, fossé), le suivi du chemin doit être précis.

Le chemin typique à suivre est une suite de droites, avec enchaînement fluide à vitesse non nulle. Le robot doit anticiper la droite suivante et tourner avec un rayon de courbure de l'ordre de 30 cm, afin de pouvoir tourner autour d'un obstacle fixe en suivant le mur. De plus, le déplacement doit pouvoir se faire en marche avant comme en marche arrière.

[2] Dans ce modèle, $r_r = r_l = r$

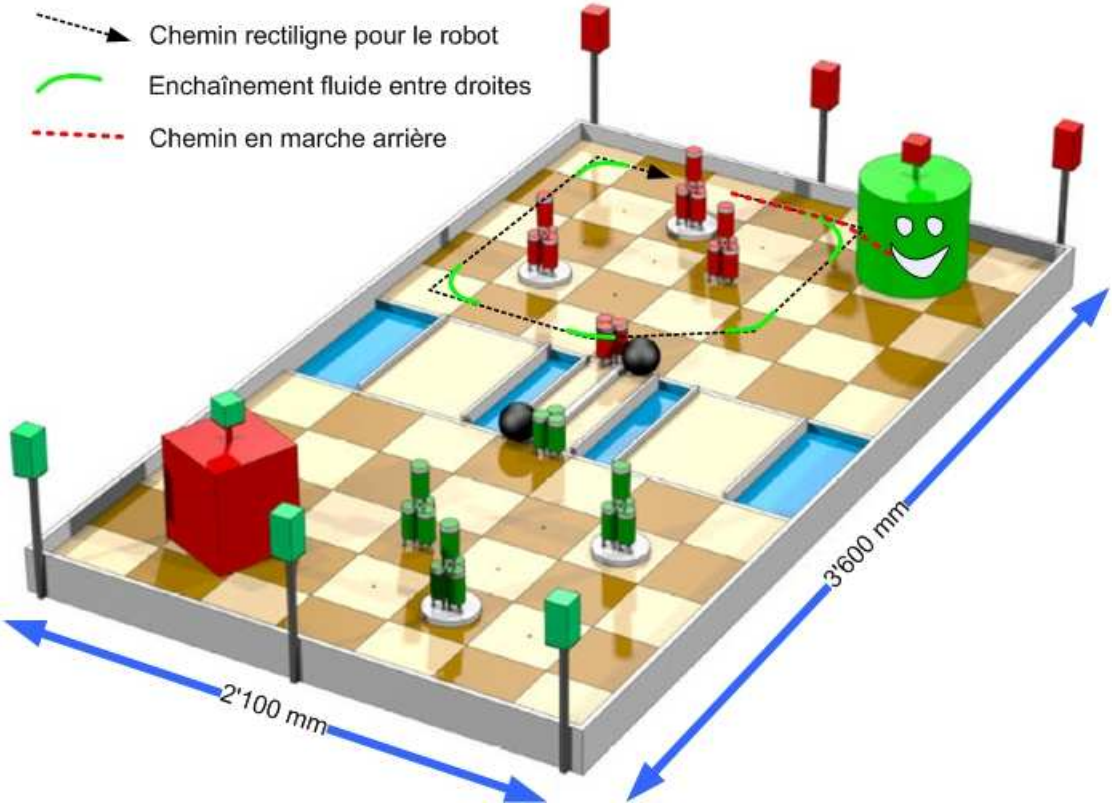

FIG. 2.4 – Déplacement typique du robot sur le terrain

Chapitre 3

La régulation

Dans la littérature, la loi de commande pour le suivi de chemin se base sur l'hypothèse que les vitesses imposées au robot sont directement transmises. Cette hypothèse correspond à une régulation en vitesse parfaite du robot.

Cette séparation entre le bas-niveau (régulation en vitesse du robot) et le haut-niveau (suivi de chemin) semble être une bonne solution pour plusieurs raisons :

- bonne séparation des tâches pour une meilleure lisibilité
- meilleur potentiel de réutilisabilité
- debuggage plus facile lors de l'implémentation

Ainsi, je vais garder cette séparation, pour avoir deux régulations en cascade.

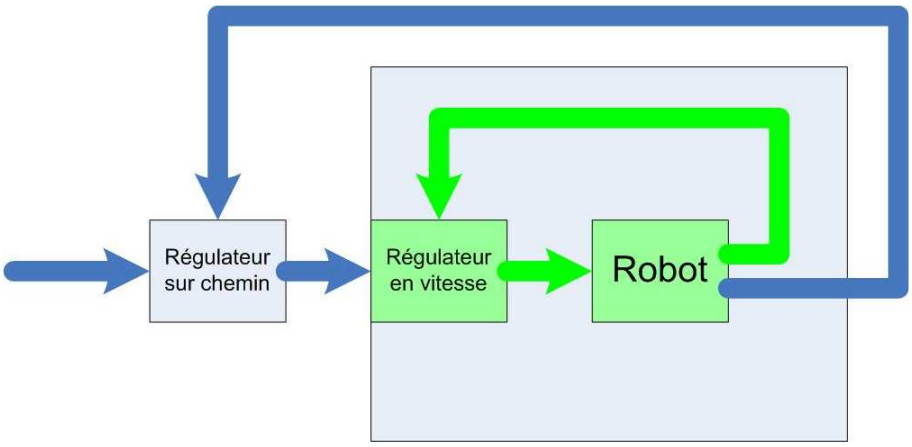

FIG. 3.1 – Les deux régulateurs en cascade

3.1 La régulation en vitesse du robot

Une bonne régulation en vitesse du robot est nécessaire pour espérer un suivi de chemin correct.

La régulation en vitesse doit avoir les caractéristiques suivantes :
- simplicité de conception et de mise en oeuvre
- robustesse, pour ne pas avoir de comportements imprévisibles
- peu de statisme, pour être proche du modèle idéal en régime permanent

L'approche standard pour la commande en vitesse d'un robot est l'utilisation d'un PID par roue. Etant donné le fort couplage dynamique entre les deux roues dû à la masse et l'inertie non négligeables, il paraît intéressant de réguler le robot à part entière, comme étant un système MIMO (Multiple Input Multiple Output).

3.1.1 Le système

Le système "robot" est un système MIMO :
- deux entrées : la tension appliquée à chaque moteur, en PWM pour ce robot
- deux sorties : la position angulaire de chaque moteur, donnée par un encodeur

Les valeurs des PWM sont sur 8 bits. Le sens de rotation du moteur est commandé séparément. Le moteur est commandé par une valeur comprise entre -255 et 255. La tension nominale d'alimentation des moteurs est de 24 Volt, mais elle peut être comprise entre 22 et 27 Volt.

Les encodeurs qui équipent les moteurs ont deux canaux en quadrature et 512 impulsions par tour d'axe moteur. Le facteur de réduction entre le moteur et la roue est d'environ 34 :1. Un tour de roue correspond donc à environ 69'262 impulsions.

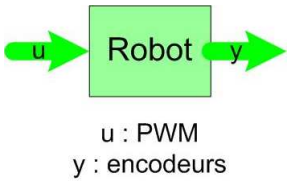

FIG. 3.2 – Système "robot"

3.1.2 Régulation d'état

Ce système MIMO pouvant être modélisé, la conception d'un régulateur d'état semble approprié.

3.1.3 Modélisation

Il est possible de modéliser le système décrit précédemment par la méthode de Newton-Euler. Les équations nécessaires sont :
- l'équation mécanique du moteur
- les équations du mouvement du robot

Equation mécanique du moteur

Les caractéristiques générales du moteur ont été données dans la section 3.1.1.

Hypothèses de modélisation :
- la tension d'alimentation est fixe à 24 Volt
- la caractéristique du moteur est linéaire par rapport à la tension appliquée
- le moteur filtre correctement le PWM, c'est-à-dire que le PWM a le même comportement que la tension correspondant au rapport cyclique
- le jeu de la transmission est nul (moteur, réducteur, courroie, roue)
- l'inertie du moteur lui-même est négligée par rapport à l'inertie de la roue
- l'inductance du moteur est négligée (L=0)
- les deux moteurs ont leurs paramètres identiques et correspondent à la datasheet [1]

$$U_i(t) = R_i \cdot i_i(t) \tag{3.1}$$

$$\ddot{\theta}_i(t) \cdot I_i = \sum_{i=1}^{n} M_i^j(t) \tag{3.2}$$

$$\sum_{i=1}^{n} M_i^j(t) = M_i^1(t) + M_i^2(t) + M_i^3(t) \tag{3.3}$$

$$M_i^1(t) = k_i \cdot n_i \cdot i_i(t) \tag{3.4}$$

$$M_i^2(t) = -F_i(t) \cdot r_i \tag{3.5}$$

$$M_i^3(t) = -f_i \cdot \dot{\theta}_i(t) \tag{3.6}$$

[1] Pour les équations suivante l'index i correspond soit à r pour droite soit à l pour gauche

avec[2] :
U_i(t) : tension appliqué au moteur
R_i : résistance du moteur
i_i(t) : courant dans le moteur
θ_i(t) : position angulaire de la roue
I_i : moment d'inertie rapporté à la roue
M_i^1(t) : couple moteur reporté sur la roue
M_i^2(t) : couple du frottement de la roue sur le terrain
M_i^3(t) : couple dû au frottement dynamique de rotation de la roue
k_i : constante du moteur
n_i : facteur de réduction
F_i(t) : force de frottement de la roue sur le terrain
r_i : rayon de la roue
f_i : coefficient de frottement dynamique

Equations du mouvement du robot

$$I_0 \cdot \ddot{\theta}_0(t) = r_0 \cdot (F_r(t) - F_l(t)) \tag{3.7}$$
$$m_0 \cdot \ddot{x}(t) = F_r(t) - F_l(t) \tag{3.8}$$
$$\ddot{\theta}_0(t) = \frac{\ddot{\theta}_r(t) \cdot r_r - \ddot{\theta}_l(t) \cdot r_l}{2 \cdot r_0} \tag{3.9}$$
$$\ddot{x}(t) = \frac{\ddot{\theta}_r(t) \cdot r_r + \ddot{\theta}_l(t) \cdot r_l}{2} \tag{3.10}$$

avec :
I_0 : moment d'inertie du robot, sur l'axe des roues, au centre
θ_0(t) : position angulaire du robot
r_0 : demie largeur entre les roues
F_i(t) : force de frottement de la roue sur le terrain
m_0 : masse du robot
x(t) : position linéique du robot
θ_i(t) : position angulaire de la roue
r_i : rayon de la roue

Modèle d'état continu

Les équations présentées ci-dessus nous montrent qu'il s'agit d'un système d'ordre 4.

Les états sont les suivants, dans l'ordre[3] :
θ_r : la position angulaire de la roue droite

[2] Les expressions suivantes sont en unités SI
[3] La dépendance du temps (t) a été omise par soucis de lisibilité.

$\dot{\theta}_r$: la vitesse angulaire de la roue droite
θ_l : la position angulaire de la roue gauche
$\dot{\theta}_l$: la vitesse angulaire de la roue gauche

Avec l'aide d'un logiciel de calcul symbolique (Mathematica 5.0), il est possible d'obtenir le modèle d'état algébrique continu.

Les équations compactes :

$$\dot{X} = A \cdot X + B \cdot U \tag{3.11}$$
$$Y = C \cdot X + D \cdot U \tag{3.12}$$

Les équations explicitées :

$$\begin{pmatrix} \dot{\theta}_r \\ \ddot{\theta}_r \\ \dot{\theta}_l \\ \ddot{\theta}_l \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & a_r^r & 0 & a_l^r \\ 0 & 0 & 0 & 1 \\ 0 & a_r^l & 0 & a_l^l \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ b_r^r & b_l^r \\ 0 & 0 \\ b_r^l & b_l^l \end{pmatrix} \cdot \begin{pmatrix} u_r \\ u_l \end{pmatrix} \tag{3.13}$$

$$\begin{pmatrix} \theta_r \\ \theta_l \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u_r \\ u_l \end{pmatrix} \tag{3.14}$$

avec :

$$a_r^r = -\frac{(4I_l r_0^2 + r_l^2(I_0 + m_0 r_0^2))(k_r^2 n_r^2 + f_r R_r)}{I_0 I_r r_l^2 + I_0(I_l + m_0 r_l^2) r_r^2 + r_r^2(4I_l I_r + I_r m_0 r_r^2)) R_r} \tag{3.15}$$

$$a_l^r = -\frac{k_l n_l n_r (I_0 - m_0 r_0^2) r_l}{I_0 I_l r_r^2 + I_0(I_r + m_0 r_r^2) r_l^2 + r_l^2(4I_r I_l + I_l m_0 r_l^2)) R_l} \tag{3.16}$$

$$a_r^l = -\frac{k_r n_r n_l (I_0 - m_0 r_0^2) r_r}{I_0 I_r r_l^2 + I_0(I_l + m_0 r_l^2) r_r^2 + r_r^2(4I_l I_r + I_r m_0 r_r^2)) R_r} \tag{3.17}$$

$$a_l^l = -\frac{(4I_r r_0^2 + r_r^2(I_0 + m_0 r_0^2))(k_l^2 n_l^2 + f_l R_l)}{I_0 I_l r_r^2 + I_0(I_r + m_0 r_r^2) r_l^2 + r_l^2(4I_r I_l + I_l m_0 r_l^2)) R_l} \tag{3.18}$$

$$b_r^r = \frac{k_r n_r (4I_l r_0^2 + r_l^2(I_0 + m_0 r_0^2))}{I_0 I_r r_l^2 + I_0(I_l + m_0 r_l^2) r_r^2 + r_r^2(4I_l I_r + I_r m_0 r_r^2)) R_r} \tag{3.19}$$

$$b_l^r = \frac{k_l n_l r_r r_l (I_0 - m_0 r_0^2)}{I_0 I_l r_r^2 + I_0(I_r + m_0 r_r^2) r_l^2 + r_l^2(4I_r I_l + I_l m_0 r_l^2)) R_l} \tag{3.20}$$

$$b_r^l = \frac{k_r n_r r_l r_r (I_0 - m_0 r_0^2)}{I_0 I_r r_l^2 + I_0(I_l + m_0 r_l^2) r_r^2 + r_r^2(4I_l I_r + I_r m_0 r_r^2)) R_r} \tag{3.21}$$

$$b_l^l = \frac{k_l n_l (4I_r r_0^2 + r_r^2(I_0 + m_0 r_0^2))}{I_0 I_l r_r^2 + I_0(I_r + m_0 r_r^2) r_l^2 + r_l^2(4I_r I_l + I_l m_0 r_l^2)) R_l} \tag{3.22}$$

Modèle d'état discret

Le modèle d'état doit ensuite être discrétisé sur la base des matrices numériques du modèle continu pour obtenir la forme :

$$X(k+1) = \Phi \cdot X(k) + \Gamma \cdot U(k) \quad (3.23)$$

$$Y(k) = C \cdot X(k) + D \cdot U(k) \quad (3.24)$$

La période d'échantillonnage du système a été choisie à 10 ms. La fréquence de 100 hz respecte le théorème de Shannon et est réalisable sur le système informatique.

Un logiciel de calcul numérique (Matlab 7.0.1) permet d'obtenir un résultat numérique.

Unités compatibles pour l'implémenation

Afin d'avoir une simulation cohérente avec les résultats expérimentaux, j'ai jugé intéressant de n'utiliser que des unités propres au système.

Ainsi, la tension appliquée au moteur devient la valeur PWM (entre -255 et 255) correspondant à un rapport cyclique. La position de la roue est la valeur du compteur des encodeurs du moteur.

Ces paramètres étant spécifiques à ce robot, je ne développe pas ce point.

3.1.4 Conception du régulateur

Sur la base du modèle d'état discret, il est possible de synthétiser un régulateur d'état. Le choix s'est arrêté sur la commande optimale qui répond parfaitement aux besoins.

Comme il s'agit d'un montage en asservissement et régulation, je travail en variable écart autour de la valeur nominale.

Commande optimale

La fonction coût à minimiser est la suivante :

$$J = \frac{1}{2} \sum_{k=0}^{N} [x^T(k) Q_1 x(k) + u^T(k) Q_2 u(k)] \quad (3.25)$$

Il faut ainsi choisir les matrices Q_1 (4×4) et Q_2 (2×2). Les matrices sont choisies diagonales positives.

Le poids à donner entre les éléments (moteurs, vitesses) droite et gauche logiquement choisis sont identiques.

Le poids de la position est nul.
Le poids des vitesses est beaucoup plus élevé que celui de la tension des moteurs.

Voici les matrices :

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 100'000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100'000 \end{pmatrix} \tag{3.26}$$

$$Q_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \tag{3.27}$$

La commande *dlqr* de Matlab donne le gain optimal.
Ainsi, la tension à appliquée en boucle fermée est :

$$U(k) = K_{\propto} \cdot X(k) \tag{3.28}$$

L'"observateur"

Tous les états ne sont pas directement mesurables, seuls les positions le sont, grâce aux encodeurs des moteurs.

Une solution aurait été de créer un observateur pour les états manquants (vitesses des roues) comme ça se fait traditionnellement.

Mais une solution plus simple réside en l'estimation de la vitesse instantanée par différenciation directe de la position :

$$vitesse(k) \cong \frac{position(k) - position(k-1)}{periode\_d'echantillonnage} \tag{3.29}$$

Ceci n'est qu'une approximation de la vitesse, mais elle suffisante pour notre application.

Action intégrale

Afin de diminuer le statisme en régime permanent, il est nécessaire d'ajouter des intégrateurs sur les vitesses.

L'action intégrale a été choisie itérativement en observant le résultat de la simulation, afin d'obtenir un statisme faible mais en évitant d'overshooter la valeur de consigne. La matrice est donc de faible valeur.

La matrice K_{I} (qui multiplie tous les états) est la suivante :

$$K_{\mathrm{I}} = \begin{pmatrix} 0 & k_{\mathrm{I}} & 0 & 0 \\ 0 & 0 & 0 & k_{\mathrm{I}} \end{pmatrix} \tag{3.30}$$

3.1.5 Schéma fonctionnel

La figure 3.3 représente le schéma fonctionnel de la régulation en vitesse du robot, synthétisant la présente section.

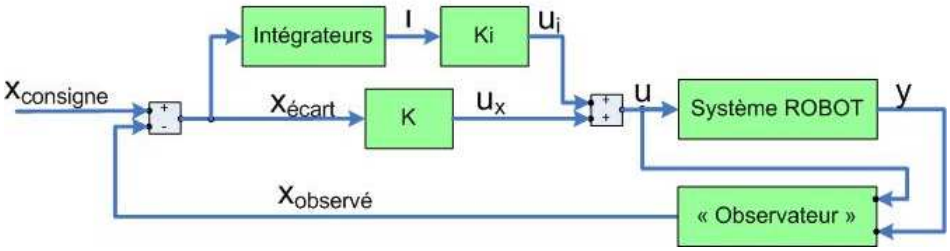

FIG. 3.3 – Schéma fonctionnel de la régulation en vitesse

3.2 La régulation sur le chemin

La partie principale du projet est le choix, la compréhension et la mise en oeuvre du suivi de chemin. Plusieurs approches du problème existent, mais la méthode de référence dans le domaine reste celle proposée par le Français Claude Samson[3].

Une hypothèse importante doit être respectée : le calcul odométrique de la position est exempt d'erreurs. Ceci est possible si les roues roulent sans glisser.

3.2.1 Loi de commande de C. Samson

Une explication simplifiée de cette loi de commande a été faite dans la thèse de E. Gauthier de Grenoble[5].

C. Samson propose une loi de commande permettant de suivre un chemin de manière géométrique indépendamment de la vitesse du robot. La seule variable de commande utilisée est la vitesse angulaire $\omega = \dot{\theta}$ du robot.

Soit C le chemin suivi et $\rho(s)$ sa courbure au point d'abscisse curviligne s, on note R' le point de C le plus proche de R (point de référence du robot) et d la distance $||RR'||$ (voir figure 3.4). De même on note θ_{des} l'angle de la tangente à la courbe en R' dans le repère principal. Cet angle représente l'orientation désirée pour le robot.

Le but de la loi de commande proposée est de réduire à la fois l'erreur en distance d et l'erreur en orientation :

$$\theta_e = \theta - \theta_{des} \tag{3.31}$$

Elle est donnée dans sa forme la plus simple par :

$$\omega = v \cdot (\rho(s_{R'}) - k_1 \cdot d - k_2 \cdot \theta_e) \tag{3.32}$$

où $s_{R'}$ est l'abscisse curviligne du point R' et k_1 et k_2 sont deux constantes positives. La vitesse de translation v du robot qui intervient comme un paramètre de la loi de commande est donc laissée libre et n'influe pas sur la convergence géométrique vers C (à condition de rester non nulle).

Des valeurs des constantes permettant de limiter les oscillations peuvent être obtenues dans le cas d'un chemin rectiligne :

$$k_1 = \xi^2 \tag{3.33}$$
$$k_2 = 2\zeta\xi \tag{3.34}$$

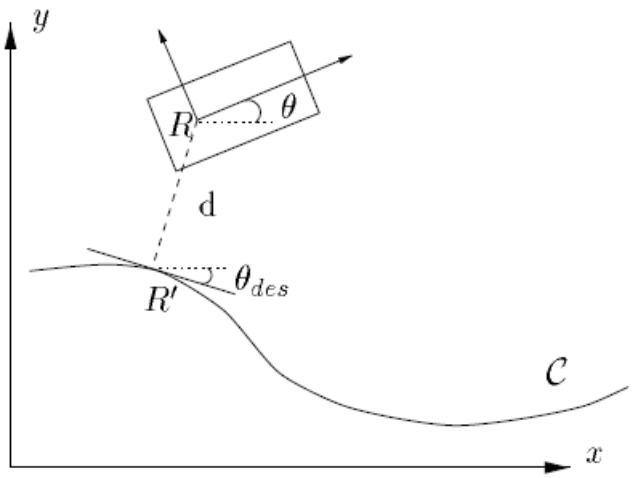

FIG. 3.4 – Suivi de chemin par la méthode de C. Samson

où ξ est la pulsation propre du système et ζ son amortissement. Ces valeurs sont également admises dans le cas général.

Les figures 3.5 et 3.6 illustrent les résultats de simulation issus de la publication de C. Samson [3] (il s'agit ici d'un véhicule unicycle, mais le principe est identique dans notre cas).

3.2.2 Enchaînement de droites

Le problème du suivi d'une droite étant théoriquement résolu, il faut s'attaquer à l'enchaînement de deux droites consécutives. Plusieurs approches sont possibles et sont décrites dans les paragraphes suivants.

Intersection brute des droites

La solution la plus simple serait de lier directement les droites et de donner ce chemin à suivre au robot.

Il paraît évident que le robot n'est pas capable de suivre un tel chemin sans s'arrêter à l'intersection puis se tourner en direction de la droite suivante. Cette solution d'enchaînement à vitesse nulle et rotation sur place n'est pas applicable dans notre cas. En effet, la forme du robot et la configuration des obstacles empêche un tel mouvement (blocage), principalement à cause du porte-à-faux du robot.

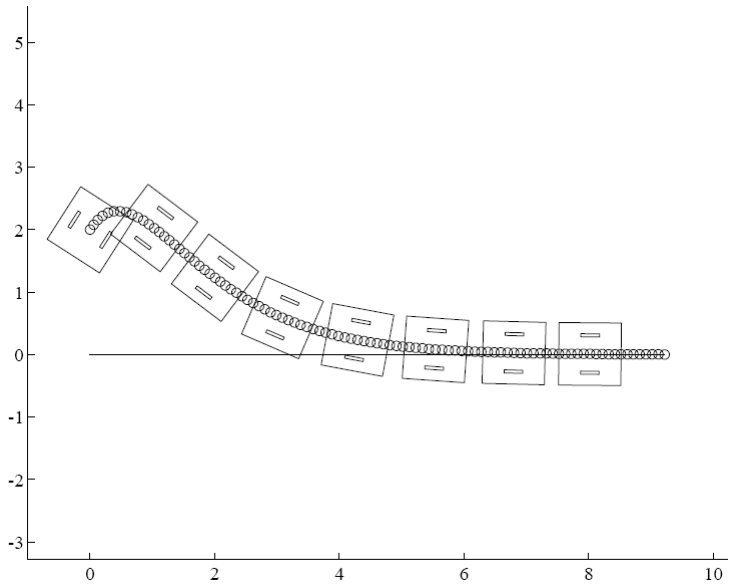

FIG. 3.5 – Simulation de suivi d'une droite

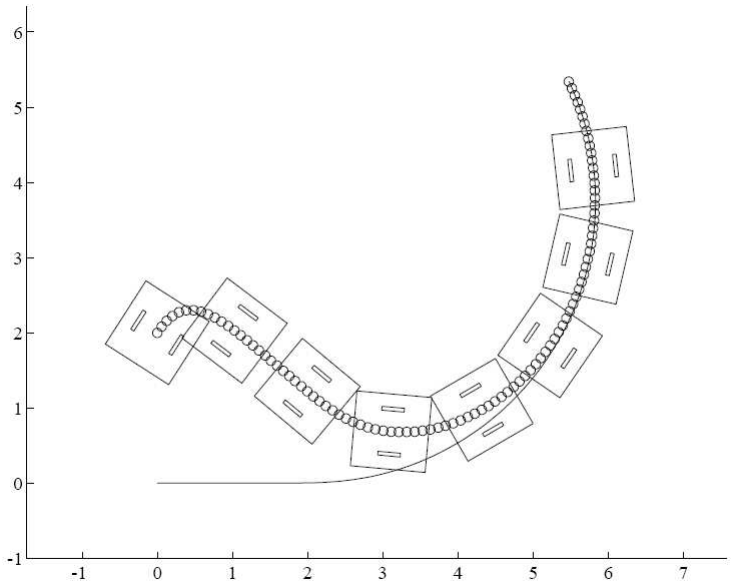

FIG. 3.6 – Simulation de suivi d'un arc de cercle

Dans le cas où le robot ne s'arrête pas à l'intersection, il va fortement overshooter la deuxième droite et prendre du temps pour l'intercepter à nouveau (figure 3.7 gauche).

Cette approche n'est donc pas adéquate.

Arc de cercle entre deux droites

Une solution qui vient rapidement à l'esprit est de relier les deux droites par un arc de cercle tangent aux droites.

Cette solution résout partiellement le problème de l'anticipation de droite. Cependant, la détermination de l'arc adéquat reliant les deux droites n'est pas triviale, et le robot overshootera malgré tout le chemin. Ceci vient du fait que la régulation attend une erreur avant de la corriger (figure 3.7 droite).

Cette approche est intéressante, mais n'a pas été choisie pour les raisons citées précédemment.

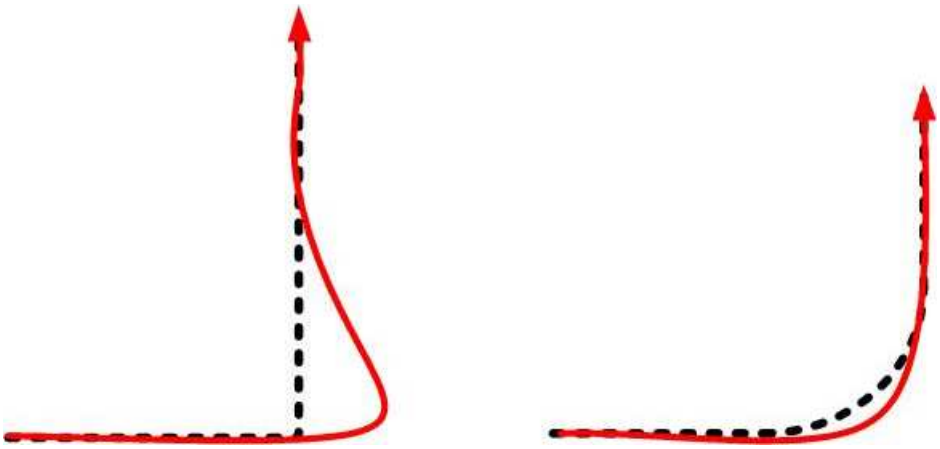

FIG. 3.7 – Enchaînement de droites : direct (figure gauche) et avec arc de cercle (figure droite)

Commande prédictive

Une approche intéressante a été présentée dans une publication de trois Danois[4].

Il s'agit d'une commande prédictive avec la différence d'angle au chemin comme variable manipulée.

Cette approche paraît efficace et élégante. Les résultats de simulation issus de la publication sont illustrés dans la figure 3.8. C'est la solution qui a été retenue pour ce travail. Elle est développée plus en profondeur ci-dessous.

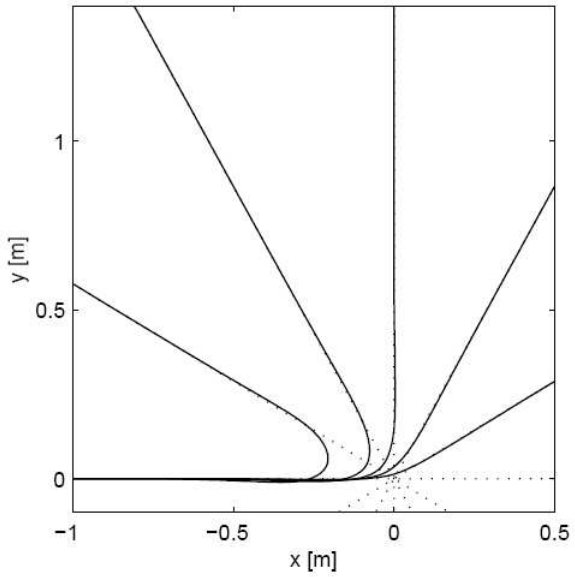

FIG. 3.8 – Simulation d'enchaînement de droites avec commande prédictive pour différents angles

3.2.3 Commande prédictive

Dans notre cas, la commande prédictive nous donne la vitesse angulaire ω en fonction des écarts à la trajectoire d et θ_e conformément à la notation de la section 3.2.1.

Le modèle

La commande prédictive nécessite un modèle cinématique linéaire comme référence. Il se base sur le modèle suivant, consistant pour le suivi d'une ligne droite :

$$\dot{s} = v \cdot cos(\theta - \psi) \quad (3.35)$$
$$\dot{d} = v \cdot sin(\theta - \psi) \quad (3.36)$$
$$\dot{\theta} = \omega \quad (3.37)$$

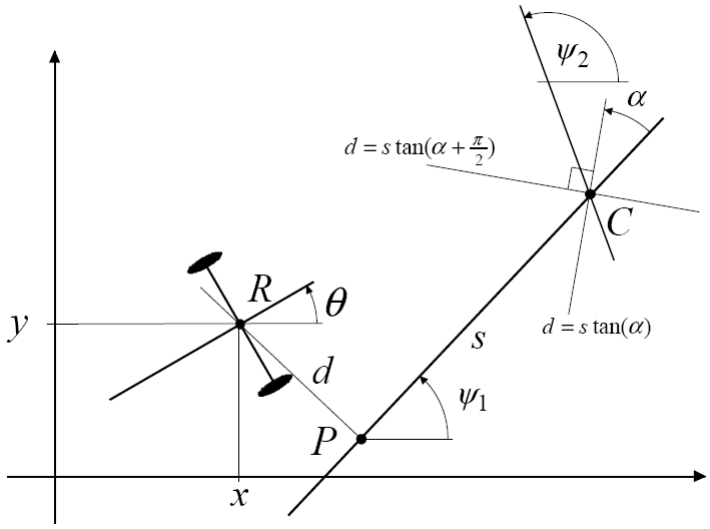

FIG. 3.9 – Définition des paramètres de la commande prédictive

où s est la distance à la prochaine intersection.

Ce modèle peut être linéarisé autour du point de fonctionnement ($d = 0$, $\theta - \psi = \theta_e = 0$) :

$$\dot{d} = v \cdot \theta_e \tag{3.38}$$

$$\dot{\theta}_e = \omega \tag{3.39}$$

Ce modèle peut être discrétisé à une période d'échantillonnage h :

$$d(k+1) = d(k) + h \cdot v \cdot [\theta(k) - \psi(k) + \frac{h}{2}] \tag{3.40}$$

$$\theta(k+1) = \theta(k) + h \cdot \omega(k) \tag{3.41}$$

Ce modèle peut être sous forme de représentation d'état :

$$\begin{pmatrix} d(k+1) \\ \theta(k+1) \end{pmatrix} = \begin{pmatrix} 1 & hv \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2}v \\ h \end{pmatrix} \omega(k) + \begin{pmatrix} 0 & -hv \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix} \tag{3.42}$$

Sous une forme compacte, ceci donne :

$$z(k+1) = A \cdot z(k) + B_\omega \cdot \omega(k) + B_r \cdot r(k) \tag{3.43}$$

avec :

$z(k) = \begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix}$ l'état du système (écarts au chemin) au temps kh

$r(k) = \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix}$ le vecteur de référence (mis sous cette forme pour la consistance des calculs)

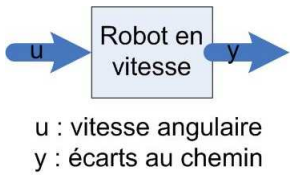

FIG. 3.10 – Système SIMO "robot régulé en vitesse"

Le critère

Le critère à minimiser sur l'horizon choisi est :

$$J(k) = \sum_{n=0}^{N} [\hat{z}(k+n) - r(k+n)]^T Q [\hat{z}(k+n) - r(k+n)] + \lambda \omega^2(k+n) \quad (3.44)$$

où $\hat{z}$ est la sortie prédite, Q une matrice de pondération, λ un poids scalaire et N l'horizon.

En introduisant la notation matricielle, il vient :

$$J(k) = [\hat{Z}(k) - R(k)]^T I_Q [\hat{Z}(k) - R(k)] + \lambda \Omega^T(k) \Omega(k) \quad (3.45)$$

avec :

$$\hat{Z}(k) = \begin{pmatrix} \hat{z}(k|k) \\ \vdots \\ \hat{z}(k+N|k) \end{pmatrix} = Fz(k) + G_\omega \Omega(k) + G_r r(k) \quad (3.46)$$

$$I_Q = \begin{pmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & Q & 0 \\ 0 & \dots & \dots & 0 & Q \end{pmatrix} \quad (3.47)$$

où :

$$\Omega(k) = \begin{pmatrix} \omega(k) & \dots & \omega(k+N) \end{pmatrix}^T \quad (3.48)$$

$$R(k) = \begin{pmatrix} r(k) & \dots & r(k+N) \end{pmatrix}^T = \begin{pmatrix} \mathbf{0} \\ \Psi(k) \end{pmatrix}^T = \begin{pmatrix} 0 & \dots & 0 \\ \psi(k) & \dots & \psi(k+N) \end{pmatrix}^T \quad (3.49)$$

et :

$$F = \begin{pmatrix} I & A & \dots & A^N \end{pmatrix}^T \tag{3.50}$$

$$G_i = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ B_i & 0 & \dots & 0 & 0 \\ AB_i & B_i & \dots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ A^{N-1}B_i & \dots & AB_i & B_i & 0 \end{pmatrix} \tag{3.51}$$

où l'index i doit être substitué par ω et r.

L'algorithme prédictif

L'objectif est de trouver la séquence de commande $\omega(k)$, $\omega(k+1)$, ... qui minimise $J(k)$.

$$\Omega(k) = -L_z z(k) - L_r R(k) \tag{3.52}$$

avec :

$$L_z = (\lambda + G_\omega^T I_Q G_\omega)^{-1} G_\omega^T I_Q G_\omega F \tag{3.53}$$

$$L_r = (\lambda + G_\omega^T I_Q G_\omega)^{-1} G_\omega^T I_Q (G_r - 1) \tag{3.54}$$

En fait seul la prochaine commande $\omega(k)$ est appliquée au système. Donc seule la première ligne de L_z et les deux[4] premières lignes de L_r nous intéressent. la commande à appliquer devient :

$$\omega(k) = -k_d d(k) - k_\theta \theta(k) - K_\Psi \Psi(k) \tag{3.55}$$

où k_d et k_θ sont des scalaire et K_Ψ un vecteur $[1 \times (N+1)]$ multipliant le vecteur de référence $\Psi(k)$.

La référence

L'algorithme prédictif nécessite le vecteur $\Psi(k) = \begin{pmatrix} \psi(k) & \dots & \psi(k+N) \end{pmatrix}^T$ des futurs angles de référence.

Dans le cas d'une droite unique :

$$\Psi(k) = \begin{pmatrix} \psi_1 & \dots & \psi_1 \end{pmatrix}^T \tag{3.56}$$

où ψ_1 est l'angle de la droite en question.

[4] Rappelons que $r(k) = \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix}$

Dans le cas d'un virage (enchaînement de deux droites), le vecteur de référence présente la forme suivante :

$$\Psi(k) = \begin{pmatrix} \psi_1 & \dots & \psi_1, & \psi_2 & \dots & \psi_2 \end{pmatrix}^T \tag{3.57}$$

ou représenté sous forme plus compacte

$$\Psi(k) = \begin{pmatrix} \Psi_1^{\eta} & \Psi_2^{N+1-\eta} \end{pmatrix}^T \tag{3.58}$$

où Ψ_i^x représente un vecteur comprenant x fois la valeur ψ_i.

Il faut alors déterminer le point de basculement entre les ψ_1 et les ψ_2. Ce point est situé sur les bissectrices entres les deux droites. Cette limite est illustrée sur la figure 3.11, où les bissectrices (en traitillé) délimitent les zones de références entre la première et la deuxième droite.

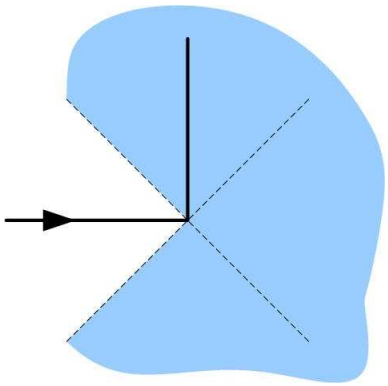

FIG. 3.11 – Changement de référence aux bissectrices

La difficulté réside dans le fait que le vecteur de référence doit être fonction du temps, alors que l'angle des deux droites et dépendant de l'espace.

Cette incohérence spatio-temporelle peut malgré tout est surpassée en estimant le moment où le robot arrivera à l'intersection des deux droites, en utilisant l'approximation suivante :

$$\eta = \frac{s(k)}{\upsilon(k)h} \tag{3.59}$$

3.2.4 Choix des paramètres

Les degrés de libertés laissés par l'algorithme prédictif doivent être fixés. Ainsi, un logiciel de calcul (Matlab 7.0.1) sera capable de nous calculer les paramètres nécessaires à l'implémentation : L_z et L_r.

La vitesse v

Bien que la vitesse soit un paramètre de l'algorithme, le robot ne se déplacera pas toujours avec la même consigne de vélocité. Il faut malgré tout fixer ce paramètre pour que le calcul de la commande soit unique. La vitesse choisie est de :

$$v = 30[\frac{cm}{s}] \tag{3.60}$$

Ceci correspond à une valeur moyenne dans notre application.

L'horizon Nh

L'horizon a été fixé à 45 [cm] devant le robot. Sachant que la vitesse v est de 30 $[\frac{cm}{s}]$, ceci correspond à un un temps de 1.5 [s].

Afin de ne pas avoir de trop grandes valeurs à stocker, le compromis suivant a été choisi :

$$N = 15 \tag{3.61}$$
$$h = 0.1[s] \tag{3.62}$$

Les pondération λ et Q

Les valeurs de pondération ont été choisis par un processus itératif sur la simulation, avec comme point de départ, les valeurs présente dans la publication de référence [4].

$$\lambda = 3'000 \tag{3.63}$$
$$Q = \begin{pmatrix} 1 & 0 \\ 0 & \delta \end{pmatrix} \tag{3.64}$$
$$\delta = 0.0001 \tag{3.65}$$

3.3 Mise en commun des régulations

La synthèse des deux régulateurs en cascade a été faite et commentée. Il faut encore les mettre en commun et faire toutes les adaptations requises.

Le schéma fonctionnel de la figure 3.12 représente le principe général de la régulation en vitesse du robot. Le bloc "robot régulé en vitesse" correspond au schéma fonctionnel 3.3 de la page 14.

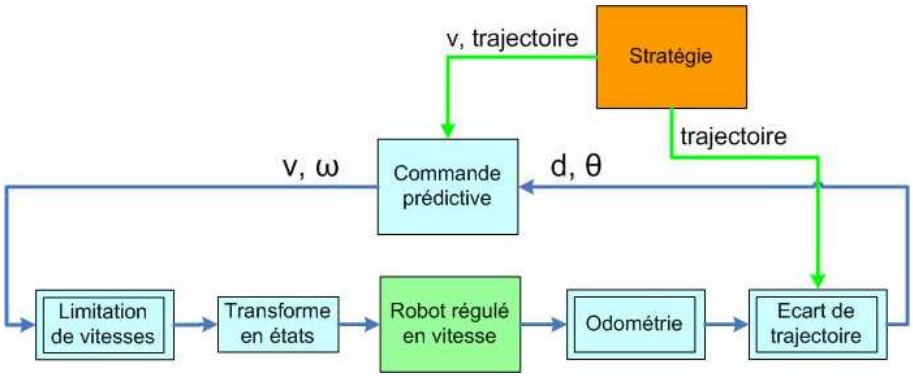

FIG. 3.12 – Schéma fonctionnel général de la régulation

Le bloc "stratégie" représente l'intelligence artificielle du robot. Il dicte le chemin à suivre et la vélocité désirée.

Le bloc non-linéaire "odométrie" calcule la position du robot, conformément au paragraphe 2.1.2.

Les blocs restants seront présentés dans les paragraphes suivants.

3.3.1 De vitesses à états

L'algorithme de la commande prédictive calcule la vitesse angulaire à appliquer au robot. Il transmet également la vélocité qui lui a été donnée par la partie stratégie.

La régulation en vitesse a comme consigne, les états qui correspondent aux vitesse et position angulaires de chaque roue.

Ces vitesses ne sont pas directement compatibles, une adaptation doit être faite. Il faut convertir le binôme vélocité-vitesse angulaire en vitesses de chaque roue, en respectant les unités de chaque module.

En faisant abstraction des unités respectives :

$$\theta_r = \frac{\upsilon + \omega \cdot r_0}{r} \tag{3.66}$$

$$\theta_l = \frac{\upsilon - \omega \cdot r_0}{r} \tag{3.67}$$

ou sous forme matricielle :

$$\begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 0 & 0 \\ 1 & r_0 \\ 0 & 0 \\ 1 & -r_0 \end{pmatrix} \begin{pmatrix} \upsilon \\ \omega \end{pmatrix} \tag{3.68}$$

3.3.2 Limitation de vitesses

Les vitesses de consignes ne peuvent pas être imposées telles quelles au robot pour respecter l'hypothèse de roulement sans glisser.

Accélération maximale

Il est nécessaire de limiter les accélérations de chaque roue. L'accélération maximale a été calculée sur la base des équations suivantes :

$$\ddot{x}_i = \ddot{\theta}_i \cdot r \tag{3.69}$$

$$\sum_j F_i^j = m \cdot \ddot{x}_o \tag{3.70}$$

$$F_i^f = \mu \cdot m \cdot g \tag{3.71}$$

où :

x_i : est la position linéaire de la roue correspondante, l'index i devant être remplacé par r pour droite et l pour gauche

F_i^f : est la forme de frottement entre la roue et le sol

μ : est le coefficient de frottement entre le caoutchouc et le bois, soit environ 0.5

Il en résulte :

$$\ddot{\theta}_i^{max} = \frac{\mu g}{r} \tag{3.72}$$

Un facteur de sécurité de cinq a été pris pour cette limitation de l'accélération angulaire maximale d'une roue.

Ces limitations ont été réparties sur les accélérations linéaire et angulaire du robot.

$$\ddot{x} = \frac{\mu g}{2\nu} \tag{3.73}$$

$$\ddot{\theta} = \frac{\mu g}{2 r_0 \nu} \tag{3.74}$$

où ν est le facteur de sécurité choisi.

Vitesse maximale

La vitesse maximale des roue est limitée physiquement à environ 80 $[\frac{cm}{s}]$. Afin de simplifier l'utilisation des vitesses, il a été choisi de séparer en deux les vitesses des roues dues aux contributions de la vélocité et de la vitesse angulaire du robot.

Ce choix implique que la vélocité maximale du robot est de 40 $[\frac{cm}{s}]$. Ceci est amplement suffisant pour notre application. La vitesse angulaire maximale du robot est de 3.3 $[\frac{rad}{s}]$.

Le robot peut ainsi avancer à vélocité maximale et toujours tourner autour d'une roue à l'arrêt.

Vitesse angulaire en fonction de la vélocité

Afin que le sens de rotation des roues ne change pas durant le déplacement, la vitesse angulaire a été bornée en fonction de la vélocité selon la relation suivante :

$$\omega_{max} = \frac{\upsilon}{r_0} \tag{3.75}$$

Cette restriction empêche par contre le robot de tourner sur lui-même durant un suivi de chemin.

3.3.3 Ecarts au chemin

Les droites sont représentées par leurs points extrêmes. Ainsi, la droite actuelle part du point (x_0, y_0) pour aller au point (x_1, y_1).

Le robot connaissant sa position (x, y), il est facile de calculer les écarts au chemin, conformément à la méthode de C. Samson.

$$d = \frac{DY \cdot dx - DX \cdot dy}{\sqrt{(dx^2 + dy^2}} \tag{3.76}$$

$$\theta_e = \theta - atan2(dy, dx) \tag{3.77}$$

où

$$dx = x_1 - x_0 \tag{3.78}$$

$$dy = y_1 - y_0 \tag{3.79}$$

$$DX = x - x_1 \tag{3.80}$$

$$DY = y - y_1 \tag{3.81}$$

avec $atan2$ la fonction trigonométrique arctangeante améliorée bien connue.

3.4 Résultats de simulation

Une simulation de ce principe de régulation a été faite sur le logiciel de simulation Simulink de Matlab. Ceci permet d'optimiser certains paramètres afin de gagner du temps sur l'implémentation réelle.

3.4.1 Les meilleurs résultats

Les meilleurs résultats de simulations sont montrés sur la figure 3.13. On y voit le chemin de référence en traitillé (deux droites avec un angle droit entre elles) et le chemin parcouru par le robot en trait continu bleu.

On remarque que même avec une position initiale hors chemin (l'angle est également initialisé à une valeur divergente au chemin), le robot intercepte bien le chemin, avec peu d'overshoot. Il reste sur le chemin jusque peu avant le virage.

La commande prédictive fait son effet, le robot anticipe le virage et se retrouve sur la deuxième droite avec peu d'overshoot.

Une perturbation de position (dans notre cas, il s'agit d'une recalibration dynamique : la position absolue est donnée au robot grâce à un système de balises). Le robot rejoint son chemin de belle manière.

3.4.2 Autres résultats

D'autres résultats intéressants de simulation sont présentés dans les paragraphes suivants.

Effet de l'anticipation

Un ajout a été fait par rapport à la théorie pour obtenir le bon résultat montré précédemment. Il a fallu ajouté une anticipation de 80 mm. Cela correspond en fait à modifier l'horizon du robot, qui doit réagir plus tôt. Il se croit donc 80 mm plus proche du point de basculement. Ceci permet d'éviter l'overshoot présent sinon (voir figure 3.14).

Angles plus aigues

Dans notre application, les angles entre les droites sont généralement droits. Cependant, il est intéressant d'évaluer la limite lors d'angles plus aigues. Nous voyons sur les figure 3.15 et 3.16 que le comportement est correct pour un angle de 60 degrés, mais devient limite pour 45 degrés. La

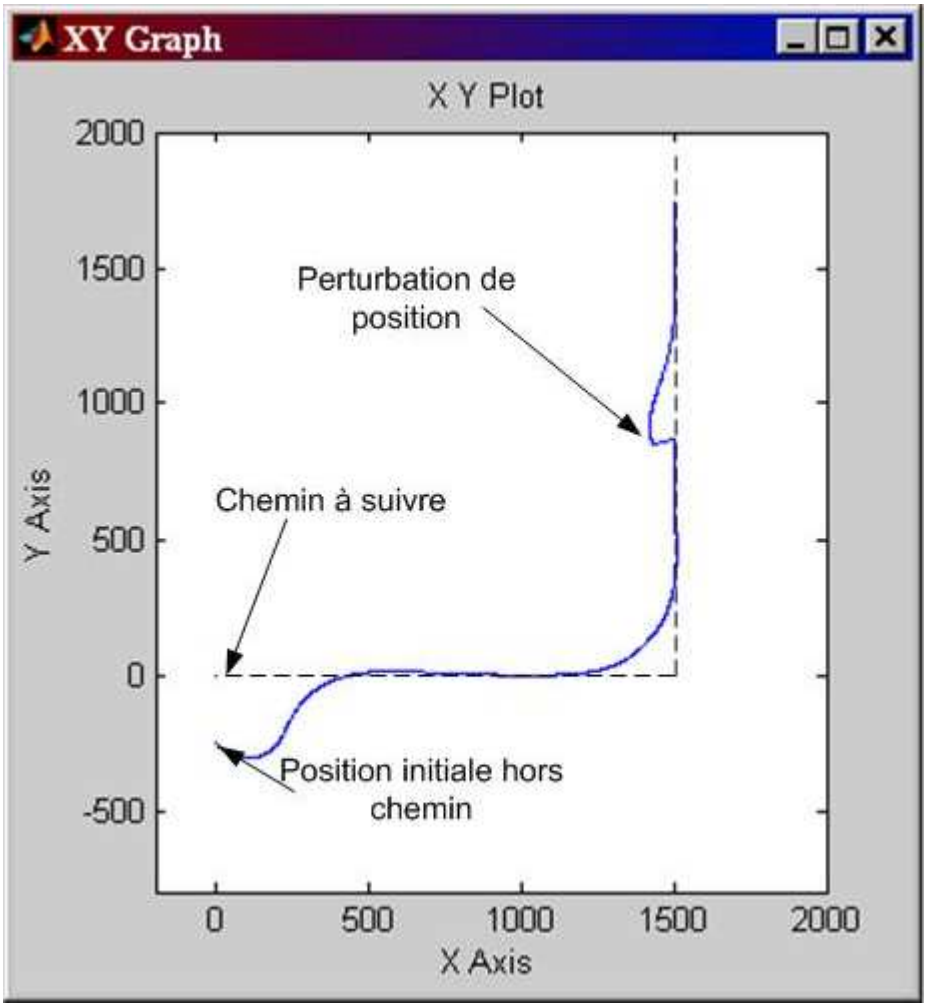

FIG. 3.13 – Résultats de simulation (unités en mm)

solution serait de ralentir avant le virage, mais cela compliquerait passablement les choses.

3.4.3 Schéma Simulink

Le schéma Simulink qui a été utilisé pour les simulation est présenté sur la figure 3.17. Il s'agit du schéma général, il comporte de nombreuses sous parties qui ne sont pas présentés dans le rapport. Tous les fichiers Matlab et Simulink sont disponibles en annexe sur CD-ROM.

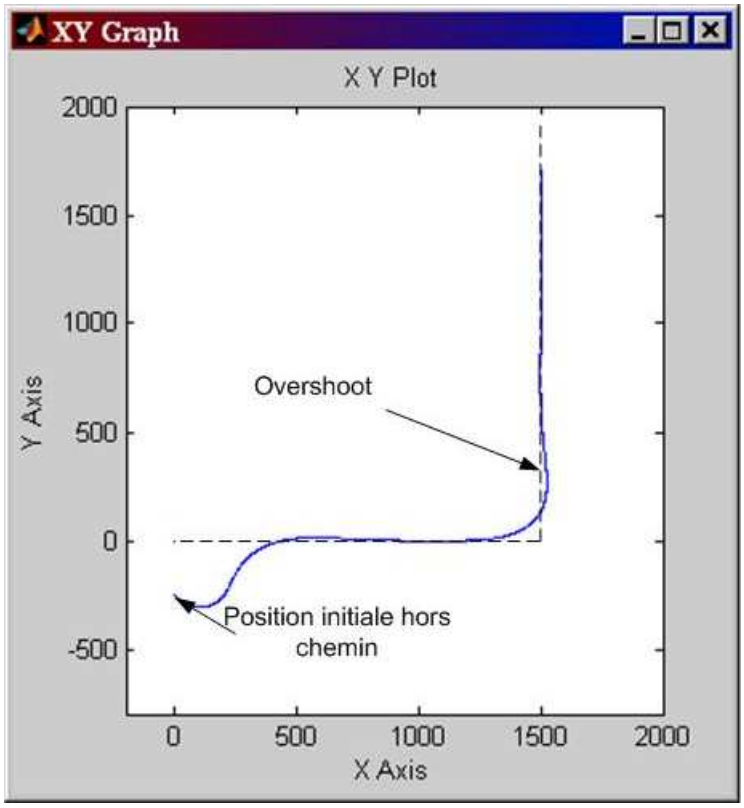

FIG. 3.14 – Résultats de simulation sans anticipation

3.4.4 Mesure de la vitesse angulaire

Des mesures de la vitesse angulaire ont été faites sur le système simulé. La figure 3.18 représente les vitesses angulaires désirée, bornée et mesurée.

3.4.5 Résultats concluants

Les résultats de la simulation sont suffisamment concluants pour permettre de passer à l'implémentation sur le système réel.

Ils auront permis de sentir l'effet des différents paramètres liés à la régulation et de déterminer les coefficients en boucle fermée.

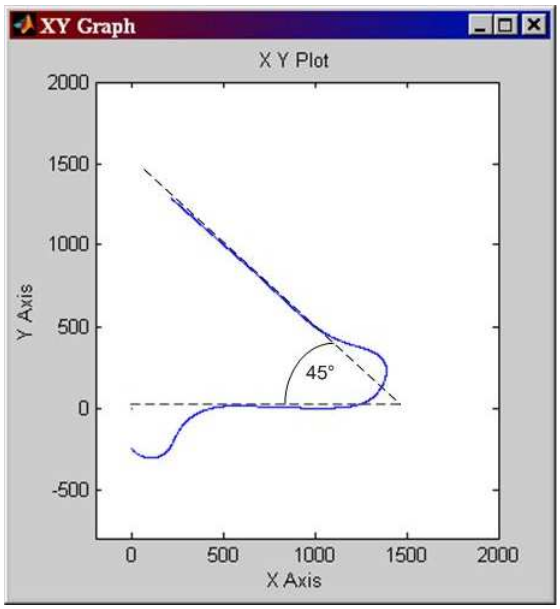

FIG. 3.15 – Résultats de simulation pour un angle de 45 degrés

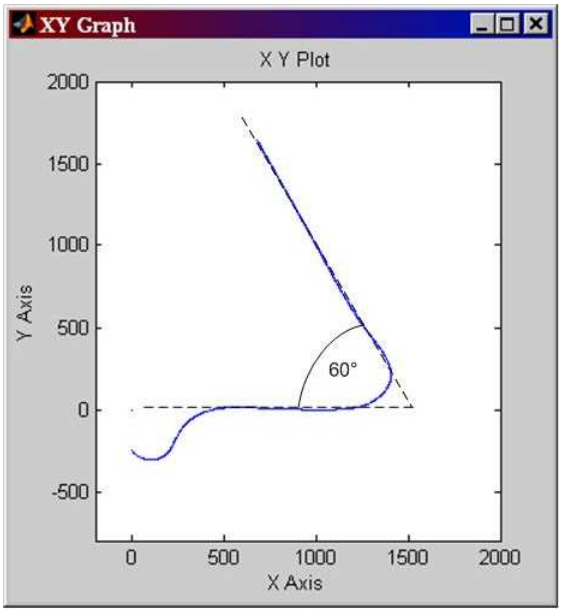

FIG. 3.16 – Résultats de simulation pour un angle de 60 degrés

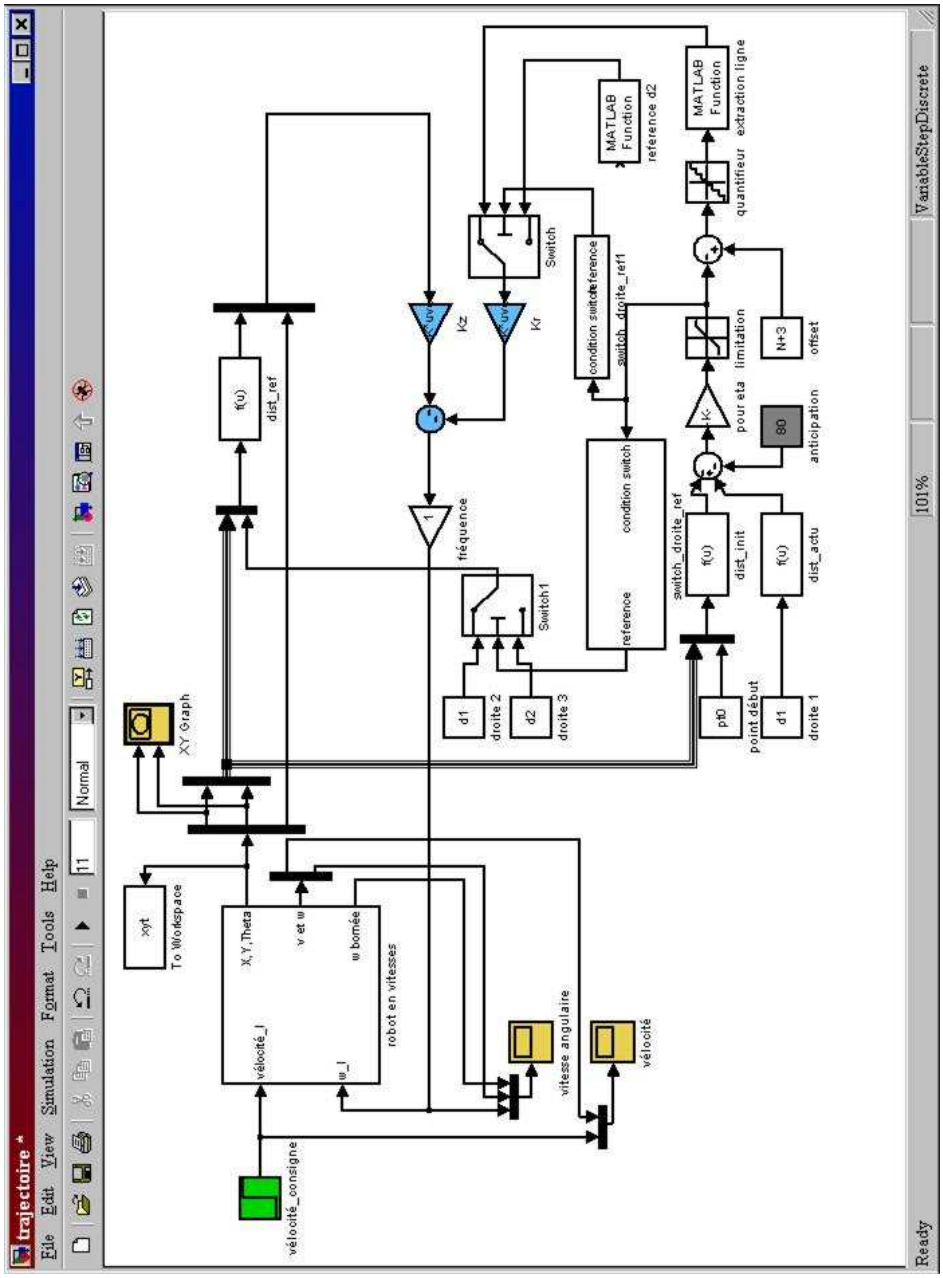

FIG. 3.17 – Schéma de simulation

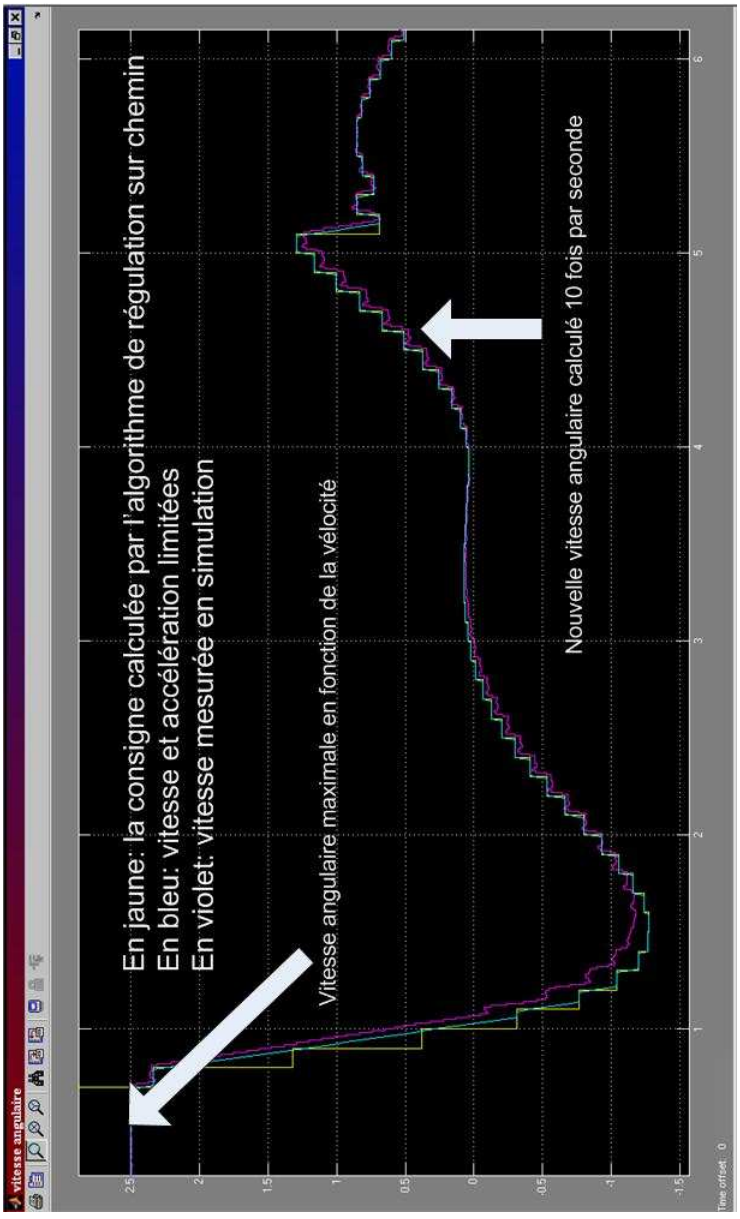

FIG. 3.18 – Mesures de la vitesse angulaire

Chapitre 4

Implémentation pratique

Après une partie théorique, puis la simulation, il est possible d'implémenter les différents algorithmes de régulation sur le système réel.

4.1 Le système informatique

Le système informatique est la plateforme sur laquelle seront programmés les algorithmes, il est essentiel de bien le connaître.

Il s'agit de la rokEPXA, une carte développée au Laboratoire d'Architecture des Processeurs (LAP) à l'EPFL. Elle comporte un processeur ARM et une logique programmable de type FPGA.

A l'intérieur de la logique programmable, un microcontrôleur de type NIOS a été implémenté lors d'un projet du projet de semestre au LAP[1]

Processeur	ARM - 32 bits	NIOS - 16 bits
Fréquence	122 Mhz	32 Mhz
Mémoire	64 Mo	12 ko
Point fort	Puissance de calcul élevée	Temps réel garanti

Tab. 4.1 – Les processeurs de la rokEPXA

Une communication à 100 hz intervient entre les deux processeurs.

Notons encore que la programmation se fait en C sur les deux processeurs. Les détails concernant le programmation en C sortent du contexte de ce projet et ne sont pas détaillés dans le present rapport.

[1]Pour plus d'information, prière de se référer au rapport de V. Longchamp en annexe sur CD-ROM.

4.1.1 Emplacement des calculs

Afin d'exploiter au maximum les capacités des processeurs, le choix de l'emplacement des calculs est très important. Comme illustré sur la figure 4.1, la régulation en vitesse du robot se fait dans le NIOS afin d'être le plus proche possible de la machine (temps réel), alors que tout le reste se fait dans l'ARM qui permet de faire des calculs demandant plus de ressources, sans contraintes temps réelles trop serrées.

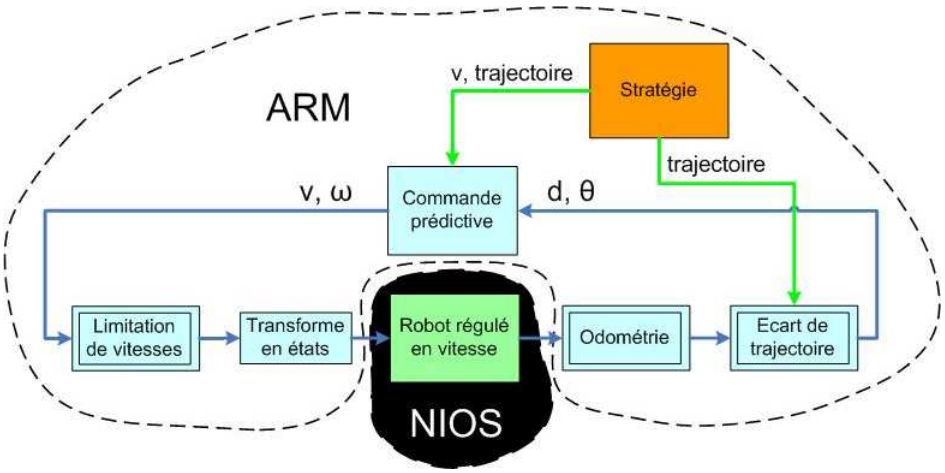

Fig. 4.1 – Choix de l'emplacement des calculs

4.1.2 Difficultés d'implémentation

Travailler en entier

Une des difficultés réside dans le fait de l'utilisation de variables de type entier (*int*) uniquement. N'ayant pas de Floating Point Unit (FPU), les processeurs ne sont pas efficaces avec des variables à virgules flottantes (*float*).

Ceci impose de revoir certains calculs et de faire des adaptations. Nous avons par exemple travaillé en millièmes de degré pour les angles afin de garder une précision suffisante.

Fonctions mathématiques

Certaines fonction mathématiques (sinus, cosinus, racine) sont trop gourmandes en calcul. Il a fallu trouver des astuces pour pouvoir malgré tout les utiliser.

Les fonctions trigonométriques utilisées par le programme (sinus et cosinus) ont été tabélisés. Une partie de la mémoire flash du système a été sacrifiée pour contenir les valeurs des sinus et cosinus.

La fonction racine carrée a été recodée pour n'utiliser que des entiers. Il s'agit d'un algorithme qui travaille bit à bit, connu sous le nom de $isqrt()$.

Ces optimisations étaient absolument nécessaires pour garantir le fonctionnement du programme (un sinus prenait plus de 4 ms). Elles ont permis de gagner un facteur 10 en temps sur chacune des fonctions adaptées.

4.2 Différences entre théorie et pratique

Quelques différences ont été observées entre la théorie et la pratique. Il fallait faire avec, et modifier certains paramètres pour limiter leur effet.

4.2.1 Odométrie imparfaite

L'odométrie n'était, comme pour tous les robots, par parfaite. Et l'erreur grandissait avec la distance parcourue.

Ceci est dû à de nombreuses causes, dont on peut citer : l'imperfection de la mécanique, le glissement des roues au sol malgré les précautions prises, la limite de rafraîchissement de la position par rapport au modèle simplifié utilisé.

Aucune de ces causes n'est supprimable, il faut donc trouver des astuces. L'ajout d'un système de balise repositionnant le robot de manière absolu était le moyen de supprimer l'erreur grandissante et repartir avec une position connue.

4.2.2 Jeu dans la transmission

Un problème majeur et inattendu était le jeu dans la transmission qui était tout sauf négligeable.

Ceci implique la transformation du système linéaire en un système non linéaire. Il y a une dynamique différente lorsque le moteur est accéléré sans qu'il ne soit couplé avec l'inertie de la roue.

Ce problème avait plusieurs effets : la vitesse mesurée sur le moteur n'était pas celle de la roue. Ceci entraînait une réaction trop brusque du

régulateur et une erreur dans le calcul de l'odométrie.

Ce problème a pu être résolu en utilisant un filtre numérique passe-bas. Il aurait aussi été possible de faire de l'observation de la vitesse en prenant en considération les vitesses sur une période plus longue.

$$\Delta\theta_{filtre}(k) = \Delta\theta(k) \cdot \alpha + (1 - \alpha) \cdot \Delta\theta_{filtre}(k - 1) \tag{4.1}$$

La réaction trop brusque sur le système a pu être diminuée grâce à une limitation de pente dans les PWM. Ceci a pour effet de limiter le couple imposé aux roues.

4.3 Résultats obtenus

Après l'évaluation des sources de problèmes et leur résolution, le système fonctionnait bien. Les paramètres utilisés sur le système réel étaient similaires à ceux de la simulation.

Le robot était capable de suivre son chemin de manière propre. Les erreurs odométriques, de l'ordre du centimètre d'écart pour deux mètres parcoururs, restaient la plus grande source de problèmes.

Le robot était prêt pour le concours. Ces déplacements étaient efficaces, mais d'autres évenements ne nous ont pas permis d'obtenir un bon classement. Le concept de déplacement était l'un des plus poussés qui soit. Notre robot précédent ayant participé à l'édition 2004 du concours n'avait même pas de régulation en vitesse, mais se déplaçait uniquement par rapport à son but. Mon approche du suivi de déplacement est un grand progrès par rapport à ce qui a été fait pour un tel concours.

Le robot reste fonctionnel et capable de faire une démonstration de ces talents sur demande...

Chapitre 5

Conclusion et perspectives

Le but de ce projet était de réguler un robot sur un chemin. Le robot et son système informatique étaient donnés.

Le robot est régulé en vitesse grâce à une méthode d'état. Le robot est régulé sur un chemin selon la loi de commande de C. Samson.

Ces deux régulateurs en cascade permettent de suivre un chemin constitué de suite de droites. L'anticipation se fait grâce à la commande prédictive prenant l'angle des droites comme variable manipulée.

Une simulation a permis de trouver les bons paramètres, nécessaires à l'implémentation sur le système réel.

La solution adoptée suivait parfaitement le cahier des charges défini pour le concours. Le robot se déplaçait sur un chemin prédéfini de manière effiace et élégante.

5.1 Perspectives

Il serait intéressant de valider la méthode en quantifiant la robustesse et la similarité à la simulation.

Le suivi d'un chemin plus complexe avec, par exemple, un autre robot pourrait être mis en oeuvre pour pouvoir tester la portabilité du principe.

L'approche de suivi de chemin a un avenir pour la robotique, que ce soit pour différents concours de robotique autonome, comme pour le transport de marchandise dans les milieux industriels.

5.2 Conclusion du projet

Ce projet était intégré dans le projet plus global de la conception de deux robot participant au concours SwissEurobot'05. Il s'inscrit dans un cadre pluridisciplinaire impliquant un groupe d'une dizaine de personne, dont j'étais responsable. De plus, il m'a permis de me familiariser avec des outils de calcul numérique et de simulation.

L'intégration de la théorie, à l'implémentation sur le système réel en passant par la simulation était un travail très complet. Le respect des contraintes, tant temporelles (délai du concours proche), financières que matérielle ont rendu la tâche plus ardue, mais le résultat est vraiment très concluant.

Le fruit de ce travail pourra certainement être réutilisé pour la commande d'autres robots mobiles, vu la qualité du résultat obtenu.

Lausanne, le 23 juin 2005 Damien PERRITAZ

Remerciements

Je tiens à remercier les personnes qui ont participé de près ou de loin à ce projet :

- MER D. Gillet, qui a accepté de suivre le projet que je lui ai proposé
- C. Salzmann, assistant responsable, qui a suivi mon travail
- P. Mülhaupt, pour ses explications sur la commande prédictive
- Toute l'équipe de robotique team-ID, qui a permis la conception et la mise en oeuvre de ce robot
- C. Braillon, qui m'a orienté vers la commande de Samson lors d'échanges de mails
- ... tous ceux que j'ai oubliés

Annexes

Les annexes suivantes sont disponibles sur le CD-ROM :

- Dossier *Mathematica* comprenant les fichiers Mathematica pour le calcul du modèle du système[1]
- Dossier *Matlab* comprenant les fichiers Matlab nécessaires au développement et à la simulation (fichier principal à lancer : commande.m) et les fichiers Simulink de la simulation
- Dossier *Rapport* comprenant le rapport final en format pdf, ainsi que le fichier source LaTeX
- Dossier *Présentations* comprenant les fichiers powerpoint des présentations intermédiaire et finale en format ppt et pps
- Dossier *Robots* comprenant le rapport et le résumé du projet de semestre de D. Perritaz, S. Progin et D. Sonney "Robot autonome pour le concours Eurobot05"
- Dossier *rokEPXA* comprenant le rapport et le résumé du projet de semestre de V. Longchamp "Contrôle d'un robot mobile par un système embarqué"
- Dossier *Bibliographie* comprenant les deux publications et la thèse

[1] il est important de changer le répertoire de travail, il doit être cohérent avec le path de Matlab

Bibliographie

Ouvrages

[1] GILLET D. *Systèmes multivariables I : méthodes d'état*, Lausanne : Laboratoire d'Automatique de l'EPFL, 2004.

[2] LONGCHAMP R. *Commande numérique de systèmes dynamiques*, Lausanne : Presses Polytechniques et Universitaires Romandes, 1995.

Publications

[3] SAMSON C. MICAELLI A. *Trajectory tracking for unicycle-type and two-steering-wheels mobile robots*, Sophia-Anitpolis, France : Institut National de Recherche en Automatique er en Automatique, 1993.

[4] BAK M. POULSEN N.K. RAVN O. *Path Following Mobile Robot in the Presence of Velocity Constraints*, Kongens Lyngby, Denmark : Technical University of Denmark, 2000.

Thèse

[5] GAUTHIER E. *Utilisation des Réseaux de Neurones Artifciels pour la Commande d'un Véhicule Autonome*, Grenoble : Institut National Polytechnique, 1999.