REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

Département de Génie Electrique

Mémoire de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'Etat en Automatique

CONCEPTION ET REALISATION D'UNE CARTE DE COMMANDE A BASE DE MICROCONTROLLEUR POUR ROBOT SCARA

Proposé et dirigé par :
M^r O.STIHI
M^r M.TADJINE

Etudié par :
M^r ABDELKRIM Saïd
M^r BADJI Boualem

Promotion : Juin 2003

Laboratoire de Commande des Processus
E.N.P.10, Avenue Hassen-Badi, EL-HARRACH, ALGER

REMERCIEMENTS

Le travail présenté dans ce mémoire est mené au laboratoire de commande des processus de l'Ecole Nationale Polytechnique d'Alger.

Nous remercions nos promoteurs, Messieurs M. Tadjine et O. Stihi, pour l'ensemble de leurs conseils et les efforts consentis durant toute l'année.

De mêmes, nos remerciements vont à Monsieur R. Illoul pour nous avoir offert l'hospitalité et nous a ainsi permis de travailler dans de bonnes conditions.

Nous remercions aussi Madame L. Hammami et Monsieur H. Chekireb pour leurs précieux conseils et orientations lesquels nous ont été d'une très grande utilité.

Nos vifs remerciements vont à S.A Nedjari, chercheur au CDTA pour son accueil, son aide et sec nombreux conseils sans oublier Mr. Jean, gestionnaire du CCU.

Nos remerciements vont également à ceux qui ont participé à notre formation, du primaire à l'université.

Boualem & Saïd

DEDICACES

Je dédie ce travail avant tout, à ma mère ; qui m'a tout donné et à qui je doit tout, et j'espère que j'aurais un jour l'occasion de la remercier d'avantage.

A mon frère sofiane pour les moyens qu'il m'a fournis, et que sans sa contribution, ce travail n'aurait pas cette finition.

A tout les membres de ma famille ; mes sœurs Malya et Iméne, mes frères Fahim, Lyes et à mon père qui m'ont apporté leur soutien durant mes études.

A ma tante Fifi , Ratiba et mon cousin Hakim.

A Salim et Kenza, mes amis Kamel, Malek, Saïd, Adel , Brahim, Tarek Disquo, Grisso et plus spécialement à BADBOU,... MKS, les amis que je n'oublierai jamais

Merci.... ! ! !

Boualem BADJI

Sommaire

Partie 2 : Modélisation du moteur PITTMAN GM9236

Chapitre 3 : Poursuite de référence et prise en compte de la dynamique des actionneurs

Chapitre 4 : Discretisation et prise en compte de l'effet de la quantification

Chapitre 5 : Mise en œuvre pratique

Annexe 4 : Etude du PIC16F877

Annexe 5 : Modules utilisés dans la mise en œuvre pratique

INTRODUCTION

Depuis l'avènement des technologies, l'Homme a voulu se décharger des tâches répétitives et inintelligibles en cherchant à automatiser tout processus de production et cela en développant des systèmes de réglages automatiques et de gestion des tâches. L'avancement de telles idées a donné naissance à la robotique, cette branche a permis le regroupement de plusieurs domaines de connaissances tels que l'automatique, l'électronique, l'électrotechnique et la mécanique. Les objectifs principaux de la conception des robots industriels sont de réaliser des tâches avec rapidité et une grande précision. De nombreux travaux de recherche, basés essentiellement sur les stratégies de commande et de leur implémentation, ont vu le jour.

La commande de mouvement est d'une très grande importance dans le domaine de la robotique. Elle a fait l'objet d'une évolution des outils informatiques et numériques, permettant l'amélioration de la commande des actionneurs grâce à l'exploitation de leur grande vitesse de traitement de l'information. Ceci a permis d'appliquer les différentes techniques auparavant impossible à utiliser avec les anciens circuits de commande. Cela a abouti à l'élaboration de nouveaux matériels spécialisés dans la commande.

Le présent travail a pour objet l'étude du robot *SCARA* (Selective Compliance Assembly Robot Arm) dont le but est de concevoir et de réaliser une carte de commande pour l'asservissement articulaire en position.

Le traitement de robot du point de vue mécanique et structurel nous paraît important. Pour cela, nous détaillons les méthodes de modélisation des robots manipulateurs et nous les appliquons au robot *SCARA*. L'étude dimensionnelle porte sur le traitement du comportement de la structure vis-à-vis des charges appliquées sur elle et aussi des flexions et des oscillations propres au système articulaire. De même, un modèle mathématique du moteur DC "PITTMAN GM9236" est donné. Dans cette partie nous allons citer les avantages des actionneurs électriques, décrire en détail le moteur à courant continue en commençant par sa constitution et ses caractéristiques mécanique et électrique, lesquels sont nécessaires au dimensionnement du régulateur.

Les techniques de commande des robots manipulateurs sont nombreuses, leur utilisation dépend essentiellement de la structure du robot (complexité) et des tâches à effectuer. La

synthèse des différentes commandes est basée, principalement, sur le modèle dynamique du robot. Nous allons donc traiter deux types de commandes qui feront l'objet de simulations.

Ces commandes sont :

- la commande classique de type PID décentralisé ;
- la commande par découplage non linéaire ;

Les simulations ont été effectuées sous environnement *Matlab* dont nous donnons la représentation d'état du modèle du robot *Scara* ainsi que sa commande, en tenant compte de la dynamique des actionneurs.

La nécessité d'implémenter un algorithme de réglage sur calculateur de processus nous amène à mettre en évidence le fonctionnement échantillonné d'un tel procédé de réglage. Dans cette vision des choses, il est important de présenter les différentes méthodes de passages au domaine échantillonné pour le dimensionnement des régulateurs.

Enfin, une carte de commande, réalisée pour l'asservissement en position d'un moteur à courant continue est présentée afin de commander le robot *SCARA*. Une étude approfondie du PIC16F877 est établie dans le but d'utiliser toutes les ressources disponibles. Le PIC16F877 est chargé d'effectuer la tâche du calculateur de processus et d'exécuter l'algorithme du réglage .

CHAPITRE I :
MODELISATION DU ROBOT SCARA

1.1.Introduction

En robotique , on associe à tout élément du poste de travail un ou plusieurs repères. Ces repères sont généralement définis de telle sorte que leurs origines correspondent à des directions et à des points privilégiés ayant un rôle fonctionnel lors de l'exécution d'une tâche, telle que la direction d'insertion, le centre de gravité, orientation d'une articulation ou extrémité d'un outil .[1]

La notion de transformation de repère est donc fondamentale, Elle permet :

- d'exprimer la situation des différents corps du robot les uns par rapport aux autres ;
- de spécifier les situation que doit prendre le repère associé à l'organe terminal du robot pour réaliser une tâche donnée ainsi que les vitesses correspondantes.

Nous présentons dans ce chapitre une notation qui permet de décrire de façon homogène les différents systèmes de coordonnées .

1.2. Transformation homogène

Faisons subir une transformation quelconque au repère $\mathbf{R_i}$, cette transformation qui l'amène sur le repère $\mathbf{R_j}$, (figure 1.1). Cette transformation est définie par la matrice ${}^{i}\mathbf{T_j}$, appelée matrice de transformation homogène ,de dimension (4x4),telle que :

$$ {}^{i}T_j = [{}^{i}s_j \ {}^{i}n_j \ {}^{i}a_j \ {}^{i}P_j] = \begin{bmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.1) $$

Où ${}^{i}\mathbf{s_j}$, ${}^{i}\mathbf{n_j}$ et ${}^{i}\mathbf{a_j}$ désignent respectivement les vecteur unitaire suivant les axes $\mathbf{x_j}$,$\mathbf{y_j}$ et $\mathbf{z_j}$ du repère $\mathbf{R}_j$ exprimés dans le repère $\mathbf{R}_i$ et où ${}^{i}\mathbf{P_j}$ est le vecteur expriment l'origine du repère $\mathbf{R}_j$ dans le repère $\mathbf{R}_i$.

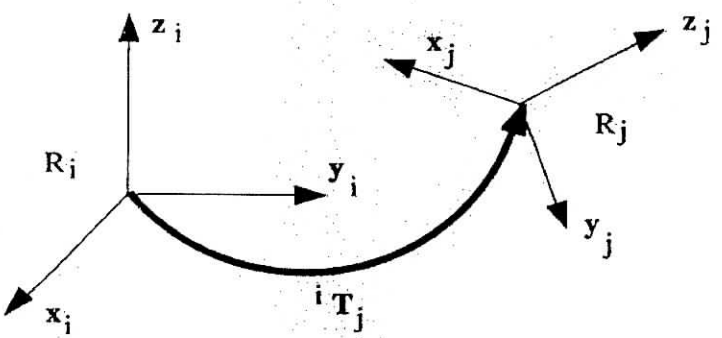


figure 1.1. *Transformation des repères.*

Par abus de notation la matrice de transformation homogène peut être exprimée de la manière suivante :

$$ {}^{i}T_j = \begin{bmatrix} {}^{i}A_j & {}^{i}P_j \\ 0 \ 0 \ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{i}s_j & {}^{i}n_j & {}^{i}a_j & {}^{i}P_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.2) $$

1.2.1. Matrice de transformation d'une translation pure

Soit cette transformation **Trans**(a,b,c), où a, b et c désigne les composantes de la translation le long de **x**, **y** et **z** respectivement.

$$ {}^{i}T_{j} = \mathrm{Trans}(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.3) $$

L'orientation étant conservée dans cette transformation, **Trans**(a,b,c) a pour expression (figure 1.2) :

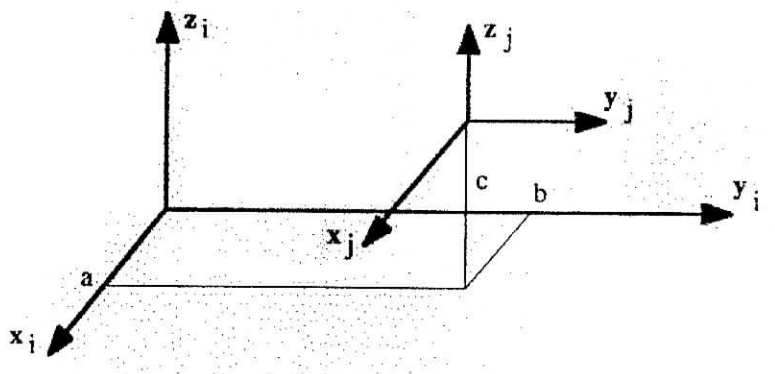


figure 1.2. *Transformation d'une translation pure.*

1.2.2. Matrice de transformation de rotation autour des axes principaux

1.2.2.1. Transformation correspondante à une rotation θ autour de l'axe x (figure 1.3)

Soit la rotation **Rot**(x,θ). On déduit les composantes des vecteurs unitaires ${}^{i}\mathbf{s}_{j}$, ${}^{i}\mathbf{n}_{j}$ et ${}^{i}\mathbf{a}_{j}$ portées respectivement par les axes $\mathbf{x}_j$, $\mathbf{y}_j$ et $\mathbf{z}_j$ du repère $\mathbf{R}_j$ et exprimés dans $\mathbf{R}_i$.
Si l'on note Sθ et Cθ les sinus et cosinus de θ respectivement nous obtenons:

$$ {}^{i}T_{j} = \mathrm{Rot}(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathrm{rot}(x,\theta) & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.4) $$

rot(**x**, θ) désigne la matrice d'orientation de dimension (3x3).

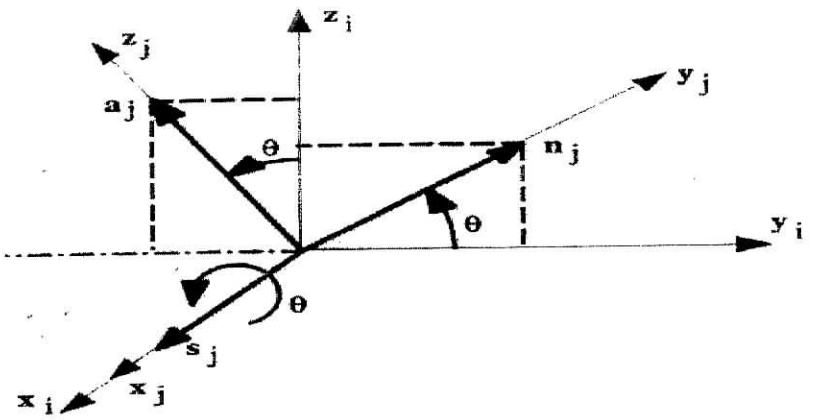


figure 1.3. *Transformation de rotation autour de l'axe x.*

1.2.2.2. Matrice de transformation correspondante à une rotation θ autour de l'axe y

avec un raisonnement analogue, nous obtenons :

$$ {}^{i}T_{j} = \mathrm{Rot}(y\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathrm{rot}(y\theta) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.5) $$

1.2.2.3. Matrice de transformation correspondante à une rotation θ autour de l'axe z

on obtient :

$$ {}^{i}T_{j} = \mathrm{Rot}(z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathrm{rot}(z,\theta) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.6) $$

1.3. Propriétés des matrices de transformation homogène

les matrices de transformation homogène possèdent certaines propriétés qui sont les suivantes :

Une matrice de transformation peut se mettre, d'après la relation (1.2), sous la forme :

$$ T = \begin{bmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & A & & P \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.7) $$

La matrice **A** représente la rotation alors que la matrice colonne **P** représente la translation. Pour une translation pure, $\mathbf{A}=\mathbf{I}_3$ ($\mathbf{I}_3$ représente la matrice identité d'ordre 3), tandis que pour une rotation pure, $\mathbf{P}=0$.

b) La matrice **A** est orthogonale, c à d $\mathbf{A}^{-1} = \mathbf{A}^T$.

c) L'inverse de la matrice de transformation représenté par la relation (1.7) peut être calculé par :

$$T^{-1} = \begin{bmatrix} & & & -s^T.P \\ & A^T & & -n^T.P \\ & & & -a^T.P \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A^T & -A^T P \\ 0\ 0\ 0 & 1 \end{bmatrix} \qquad (1.8)$$

l'inverse d'une transformation ${}^{i}\mathbf{T}_j$ est $({}^{i}\mathbf{T}_j)^{-1} = {}^{j}\mathbf{T}_i$.

d) Si un repère R_0 a subi k transformation consécutive (figure 1.4) et si chaque transformation i,(i=1,....,k),est définie par rapport au repère courant R_{i-1} , alors la transformation ${}^{0}\mathbf{T}_k$ peut être déduite de la composition des multiplication à droite de ces transformation :

$${}^{0}T_K = {}^{0}T_1\ {}^{1}T_2\ {}^{2}T_3 \ldots\ldots {}^{K-1}T_K \qquad (1.9)$$

e) Si un repère R_j , définie dans le repère R_i par la transformation ${}^{i}\mathbf{T}_j$,subit une transformation **T** exprimée dans le repère R_i , le repère R_j se transforme en $R_{j'}$ avec

${}^{i}T_{j'} = T\ {}^{i}T_j$ (figure 1.5).

A partir des propriétés e et d, on déduit que :

- une multiplication à droite de la transformation ${}^{i}\mathbf{T}_j$ signifie que la transformation est faite par rapport au repère courant R_j ;
- une multiplication à gauche signifie que la transformation est faite par rapport au repère de référence R_i .

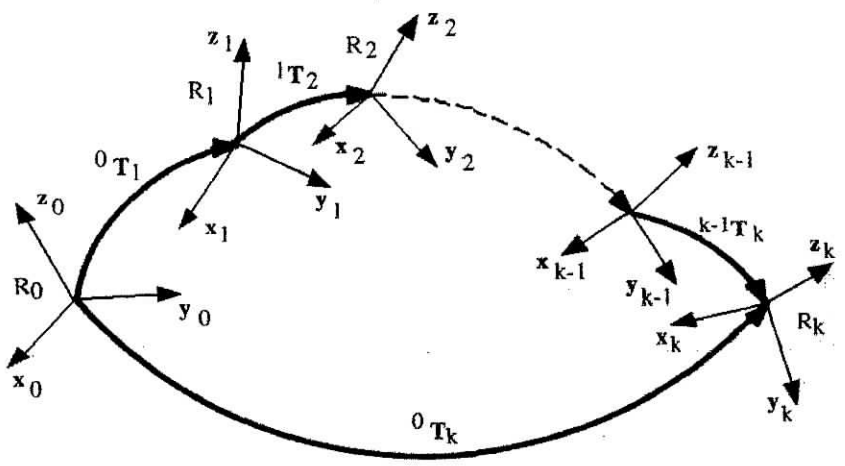


Figure 1.4. *Composition des transformations :multiplication à droite.*

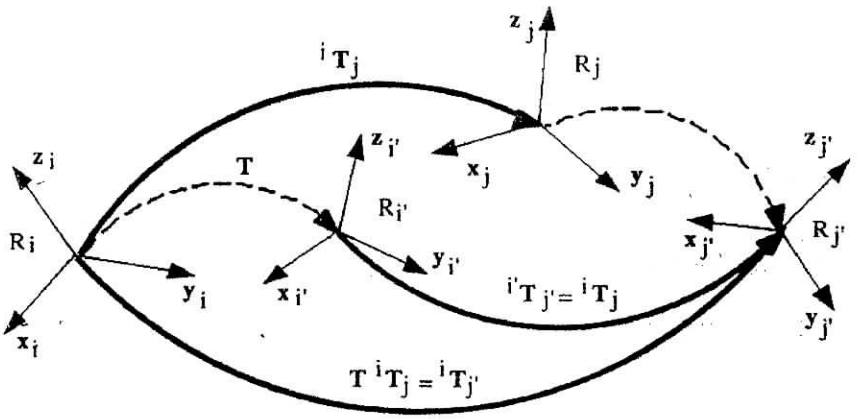


Figure 1.5. *Composition des transformations :multiplication à gauche.*

1.4. Modélisation des robots manipulateurs

1.4.1. Méthodes et notations

La modélisation des robots de façon systématique exige une méthode adéquate pour la description de leur morphologie. Plusieurs méthodes et notation ont été proposées ; la plus répandue est celle de Denavit-Hartenberg, cette méthode a été modifiée et cela en utilisant la notation de Khalil et Kleinfinger qui permet la description homogène, et avec un nombre minimum de paramètres. **[1]**

1.4.2. Description de la géométrie des robots à structure ouverte simple

Ce paragraphe présente la méthodologie à suivre pour décrire les robots à structure ouverte simple.

Une structure ouverte simple est composée de n+1 corps notés C_0,..... C_n, et de n articulation. Le corps C_0 désigne la base du robot et le corps C_n le corps qui porte l'organe terminal. L'articulation j connecte le corps C_j au corps C_{j-1} (figure 3.1) .la méthode de description est fondée sur les règles et convention suivantes :

- les corps sont supposés parfaitement rigides. Ils sont connectés par des articulations considérées comme idéales (pas de jeu mécanique, pas d'élasticité), soit rotoïdes, soit prismatiques ;
- le repère R_j est lié au corps C_j ;
- la variable de l'articulation j est notée q_j .

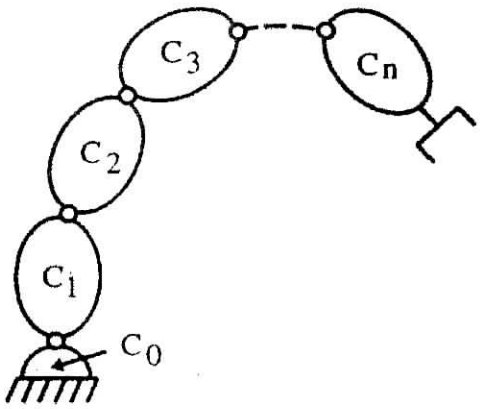


Figure 1.7.*Robot à structure ouverte simple.*

Le corps R_j ,fixé au corps C_j est défini de sorte que :

- l'axe $\mathbf{z}_j$ est porté par l'axe de l'articulation j ;
- l'axe $\mathbf{x}_j$ est porté par la perpendiculaire commune aux axes $\mathbf{z}_j$ et $\mathbf{z}_{j+1}$.

Le passage du repère R_{j-1} au repère R_j s'exprime en fonction des quatre paramètres géométriques suivants (figure 1.8) :

- α_j : angle entre les axes $\mathbf{z}_{j-1}$ et $\mathbf{z}_j$ correspondant à une rotation de $\mathbf{x}_{j-1}$;
- d_j : distance entre $\mathbf{z}_{j-1}$ et $\mathbf{z}_j$ le long de $\mathbf{x}_{j-1}$;
- θ_j : angle entre les axes $\mathbf{x}_{j-1}$ et $\mathbf{x}_j$ correspondant à une rotation de $\mathbf{z}_j$;
- r_j : distance entre $\mathbf{x}_{j-1}$ et $\mathbf{x}_j$ le long de $\mathbf{z}_{j-1}$.

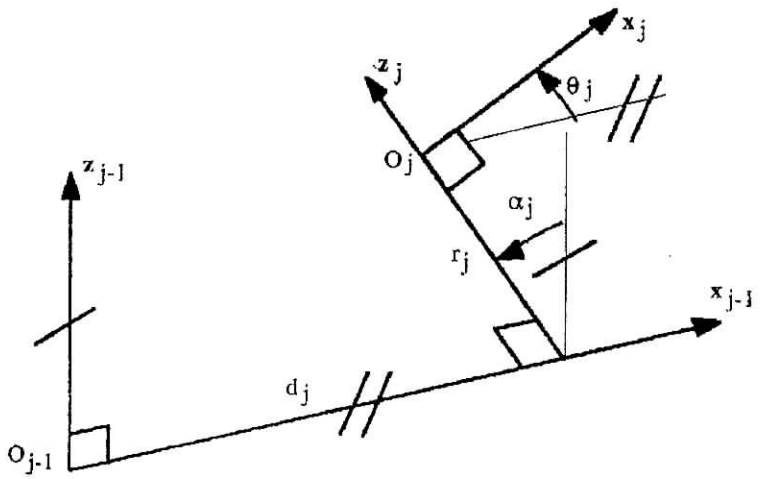


Figure 1.8 :*Paramètres géométrique dans le cas d'une structure ouverte simple.*

la variable articulaire q_j associée à la j ^{ième} articulation est soit θ_j ,soit r_j ,selon que cette articulation est de type rotoïde ou prismatique, ce qui se traduit par la relation .

$$q_j = \overline{\sigma}_j \theta_j + \sigma_j r_j \qquad \text{avec}$$

$$\sigma_j = 0 \text{ si l'articulation } j \text{ est prismatique}$$

$$\sigma_j = 1 \text{ si l'articulation } j \text{ est rotoïde.}$$

La matrice de transformation définissant le repère $\mathbf{R}_j$ dans le repère $\mathbf{R}_{j-1}$ est donnée par : (figure 1.8).

$$
\begin{aligned}
{}^{j-1}T_j &= \mathrm{Rot}(x\alpha_j)\,\mathrm{Tran}(xd_j)\,\mathrm{Rot}(z\theta_j)\,\mathrm{Tran}(zr_j) \\
&= \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \qquad (1.10)
$$

1.4.3. Modèle géométrique direct

le modèle géométrique direct (MGD) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c'est à dire les coordonnées opérationnels du robot, en fonction de ses coordonnées articulaires. Il peut être représenté par la matrice de passage ${}^0\mathbf{T}_n$:

$$
{}^0T_n = {}^0T_1(q_1)\,{}^1T_2(q_2)\,\ldots\,{}^{n-1}T_n(q_n) \qquad (1.11)
$$

le modèle géométrique direct du robot peut aussi être représenté par la relation :

$$
X = f(q) \qquad (1.12)
$$

q étant le vecteur des variables articulaires tel que :

$$
q = [\,q_1 \;\; q_2 \;\ldots\; q_n\,] \qquad (1.13)
$$

les coordonnées opérationnelles sont définies par :

$$
X = [\,x_1 \;\; x_2 \;\ldots\; x_n\,]^T \qquad (1.14)
$$

Application au robot SCARA :

le robot SCARA est un robot possédant quatre degré de liberté dont trois sont rotoïde et le quatrième prismatique(figure 1.6), il appartient a la catégorie des robots a structure ouverte simple

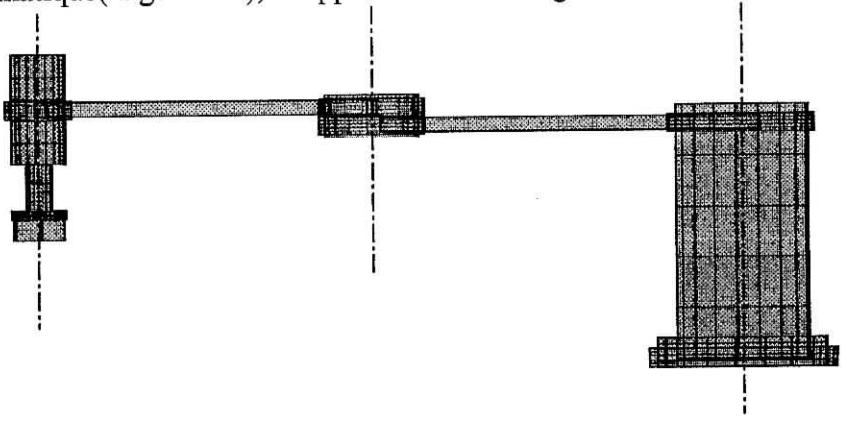


Figure 1.6. *Robot de type SCARA.*

En utilisant la méthode cité ci-dessus , les repère sont placés comme il est montré sur la figure 1.9.

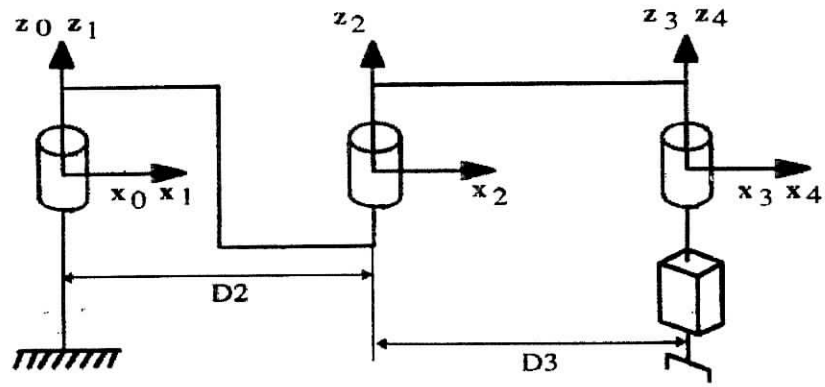


Figure 1.9. *Placement des repères.*

Les paramètres géométrique du robot SCARA sont donnés dans le tableau 1.1.

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	θ_1	0
2	0	0	D_2	θ_2	0
3	0	0	D_3	θ_3	0
4	1	0	0	0	r_4

Tableau 1.1. *Paramètres géométrique du robot SCARA.*

A partir de ce tableau et compte tenu de la relation (1.10), on déduit les matrices de transformation élémentaire ${}^{j-1}\mathbf{T}_j$ suivante :

$$ {}^{0}T_1 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}T_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & D2 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}T_3 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & D3 \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Le calcul de ${}^{0}\mathbf{T}_4$ donne le résultat suivant :

$$ {}^{0}T_4 = \begin{bmatrix} C123 & -S123 & 0 & C12\,D3 + C1\,D2 \\ S123 & C123 & 0 & S12\,D3 + S1\,D2 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.15) $$

Avec

$C123 = \cos(\theta_1 + \theta_2 + \theta_3)$ et $S123 = \sin(\theta_1 + \theta_2 + \theta_3)$

la relation (1.15) est la matrice de transformation définissant le repère R_4 dans le repère R_0.

1.4.4. Modélisation géométrique inverse du robot SCARA

Nous cherchons à résoudre le système d'équation de la relation (1.12) ; Le modèle géométrique direct d'un robot permet de calculer les coordonnées opérationnelles donnant la situation de l'organe terminal en fonction des coordonnées articulaires. Le problème inverse consiste à calculer les coordonnées articulaires correspondant à une situation donnée de l'organe terminal.

Lorsqu'elle existe, la forme explicite qui donne toutes les solutions possibles (il y a rarement unicité de solution) constitue ce que l'on appelle le modèle géométrique inverse (MGI). Les méthodes sont nombreuse , pour cela nous nous intéressons à une seule méthode celle de **Paul** qui traite séparément chaque cas particulier et convient à la plupart des robots industriels. Pour plus de detail voir **[1]**.

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^0T_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (1.16)$$

Pour la position nous avons :

Nous avons :

$$\begin{aligned} P_x &= C12\,D3 + C1\,D2 \\ P_y &= S12\,D3 + S1\,D2 \\ P_z &= r_4 \end{aligned} \qquad (1.17)$$

Nous obtenons directement :

$$r_4 = P_Z$$

Les deux premières expression de la relation (1.17) représente un système d'équation de type 8 en θ_1 et θ_2 **[1]** qui a pour solution :

$$\theta_2 = \text{ATAN2}[\pm\sqrt{1-(C2)^2}, C2] \qquad (1.18)$$

$$\theta_1 = \text{ATAN2}(S1, C2) \qquad (1.19)$$

avec :

$$C2 = \frac{P_x^{\,2} + P_y^{\,2} - (D3)^2 - (D2)^2}{2\,D3\,D2}$$

$$S1 = \frac{B1\,P_y - B2\,P_x}{B1^2 + B2^2} \qquad C1 = \frac{B1\,P_y + B2\,P_x}{B1^2 + B2^2}$$

$$B1 = D2 + D3\,C2 \qquad B2 = D3\,S2$$

Et ATAN2 est une fonction MATLAB qui nous permet de calculer l'arc tangente à partir de donnée de deux arguments.

Connaissant θ_1 et θ_2, on trouve enfin pour θ_3 :

$$\theta_3 = \text{ATAN2}(s_y, s_x) - \theta_2 - \theta_1 \qquad (1.20)$$

1.4.5. Modèle cinématique direct

Le modèle cinématique directe d'un robot manipulateur décrit les vitesses des coordonnées opérationnelles en fonction des vitesses articulaires. Il est noté :

$$\dot{X} = J(q)\,\dot{q} \qquad (1.21)$$

où **J(q)** désigne la matrice jacobienne de dimension (m x n) du mécanisme, égale à $\frac{\partial X}{\partial q}$ et fonction de la configuration articulaire **q** . La même matrice jacobienne intervient dans le calcul du modèle différentielle direct qui donne les variation élémentaires **dX** des coordonnées opérationnelles en fonction des variations **dq** , soit :

$$dX = J(q)\,dq \qquad (1.22)$$

L'intérêt de la matrice jacobienne est multiple :

- elle est à la base du modèle différentiel inverse, permettant de calculer une solution locale des variations articulaires **q** connaissant les coordonnées opérationnelles **X** ;
- en statique, on utilise le jacobien pour établir la relation liant les efforts exercés par l'organe terminal sur l'environnement aux forces et couples des actionneurs ;
- elle facilite le calcul des singularités et de la dimension de l 'espace opérationnel accessible du robot .

1.4.5.1 Matrice jacobienne de base

On indique dans ce paragraphe comment déterminer la matrice jacobienne d'un mécanisme à chaîne ouverte simple. Lorsqu'il y a des arborescences, on procède de façon analogue mais en traitant chaque chaîne de puis la base séparément.

On peut obtenir la matrice jacobienne par une méthode de calcul direct, fondée sur la relation entre les vecteurs des vitesses de translation et de rotation $\mathbf{V_n}$ et ω_n du repère $\mathbf{R_n}$, représentant les éléments de réduction du torseur cinématique du repère R_n , et les vitesses articulaires $\dot{q}$:

$$V_n = \begin{bmatrix} V_n \\ \omega_n \end{bmatrix} = J_n \dot{q} \qquad (1.23a)$$

On note que $\mathbf{V_n}$ est la dérivée par rapport au temps du vecteur $\mathbf{P_n}$. En revanche, ω_n ne représente pas la dérivée d'une représentation quelconque de l'orientation.

L'expression du jacobien est identique si l'on considère la relation entre les vecteurs de translation et de rotation différentielles ($\mathbf{d}_n$,δ_n) du repère R_n et les différentielles des coordonnée articulaires **dq** :

$$\begin{bmatrix} d_n \\ \delta_n \end{bmatrix} = \mathbf{J}_n \, \mathbf{dq} \qquad (1.23b)$$

1.4.5.2. Calcul du jacobien de base

Considérons la K^{ieme} articulation d'une chaîne articulée. La vitesse $\dot{q}_K$ induit sur le repère terminal $\mathbf{R}_n$ la vitesse de translation $\mathbf{V}_{K,n}$ et la vitesse de rotation $\omega_{K,n}$ deux cas se présentent :

- si l'articulation est prismatique ($\sigma_K = 1$, figure 1.10) :

$$\begin{cases} \mathbf{V}_{K,n} = a_K \, \dot{q}_K \\ \omega_{K,n} = 0 \end{cases} \qquad (1.24)$$

Où l'on rappelle que $\mathbf{a}_K$ est le vecteur unitaire porté par l'axe $\mathbf{z}_k$ de l'articulation K.

- si l'articulation est rotoïde ($\sigma_k = 0$, figure 1.11)

$$\begin{cases} \mathbf{V}_{K,n} = a_k \dot{q}_k \, x \, \mathbf{L}_{k,n} = (a_k \, x \, \mathbf{L}_{k,n}) \dot{q}_k \\ \omega_{k,n} = a_k \dot{q}_k \end{cases} \qquad (1.25)$$

le terme $\mathbf{L}_{k,n}$ désignant le vecteur d'origine O_k et d'extrémité O_n.

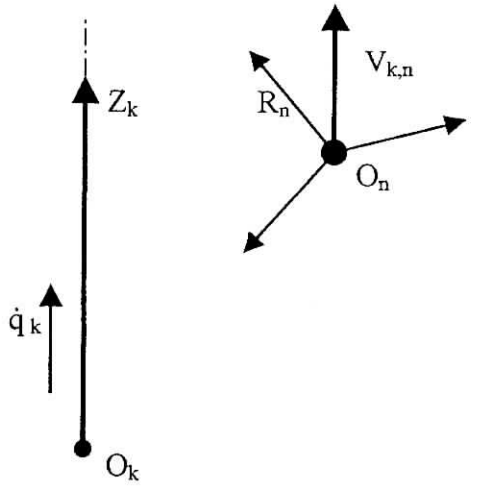


Figure 1.10. *Cas d'une articulation prismatique*

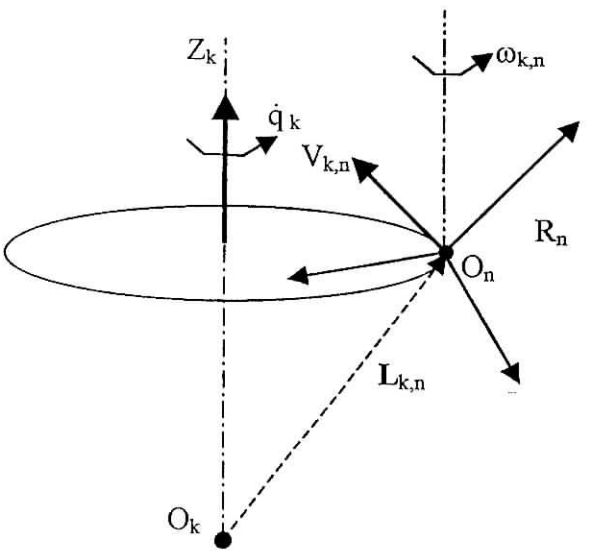


Figure 1.11. *Cas d'une articulation rotoïde*

Les vecteurs $\mathbf{V}_{k,n}$ et $\omega_{k,n}$ s'écrivent donc sous la forme générale suivante :

$$\begin{cases} \mathbf{V}_{k,n} = [\sigma_k a_k + \overline{\sigma}_k (a_k \times \mathbf{L}_{k,n})]\, \dot{q}_k \\ \omega_{k,n} = \overline{\sigma}_k a_k \dot{q}_k \end{cases} \tag{1.26}$$

En appliquant le théorème de composition des vitesses, les vitesses de translation et de rotation du repère terminal s'écrivent :

$$\begin{cases} \mathbf{V}_n = \sum_{k=1}^{n} V_{k,n} = \sum_{k=1}^{n} [\sigma_k a_k + \overline{\sigma}_k (a_k \times \mathbf{L}_{k,n})]\, \dot{q}_k \\ \omega_n = \sum_{k=1}^{n} \omega_{k,n} = \sum_{k=1}^{n} \overline{\sigma}_k a_k \dot{q}_k \end{cases} \tag{1.27}$$

En mettant ce système sous forme matricielle et en l'identifiant à la relation (1.23), on déduit que :

$$\mathbf{J}_n = \begin{bmatrix} \sigma_1 a_1 + \overline{\sigma_1}(a_1 \times L_{1,n}) & \dots & \sigma_n a_n + \sigma_n (a_n \times L_{n,n}) \\ \overline{\sigma_1} a_1 & \dots & \overline{\sigma_n} a_n \end{bmatrix} \tag{1.28}$$

Si l'on projette les élément de la relation(1.28)dans un repère R_i , on obtient le jacobien ${}^{i}\mathbf{J}_n$ de dimension (6 x n) tel que :

$${}^{i}V_n = {}^{i}\mathbf{J}_n\, \dot{q} \tag{1.29}$$

En général, on exprime $\mathbf{V}_n$ et ω_n soit dans le repère $\mathbf{R}_n$, soit dans le repère $\mathbf{R}_0$. La matrice jacobienne correspondante est notée ${}^{n}\mathbf{J}_n$ ou ${}^{0}\mathbf{J}_n$ respectivement. Ces matrices peuvent être calculées en utilisant une matrice ${}^{i}\mathbf{J}_n$, i=1,..., n, grâce à la relation de transformation de la matrice jacobienne entre repères suivante :

$${}^{s}\mathbf{J}_n = \begin{bmatrix} {}^{s}A_i & 0_3 \\ 0_3 & {}^{s}A_i \end{bmatrix} {}^{i}\mathbf{J}_n \tag{1.30}$$

où ${}^{s}\mathbf{A}_i$ est la matrice d'orientation , de dimension (3x3), du repère $\mathbf{R}_i$ exprimée dans le repère $\mathbf{R}_s$.

La matrice ${}^{s}\mathbf{J}_n$ peut être décomposée en deux matrices, la première étant toujours de rang plein.

Les deux matrices ${}^{i}\mathbf{J}_n$ et ${}^{s}\mathbf{J}_n$ ayant les mêmes positions singulières, on cherche pratiquement à utiliser le repère de projection $\mathbf{R}_i$ qui simplifie les éléments de la matrice ${}^{i}\mathbf{J}_n$ la plus simple lorsque l'on prend i=[partie entière de n/2].

1.4.5.3. Calcul de la matrice ${}^{i}\mathbf{J}_n$

En remarquant que le produit vectoriel $a_k \times \mathbf{L}_{k,n}$ peut se transformer en $\hat{\mathbf{a}}_k\, \mathbf{L}_{k,n}$, la K^{ieme} colonne de ${}^{i}\mathbf{J}_n$ notée ${}^{i}\mathbf{J}_{n;k}$ devient :

$${}^{i}\mathbf{J}_{n;k} = \begin{bmatrix} \sigma_K\, {}^{i}a_K + \overline{\sigma}_k\, {}^{i}A_K\, {}^{K}\hat{a}_K\, {}^{K}L_{K,n} \\ \overline{\sigma}_K\, {}^{i}a_K \end{bmatrix} \tag{1.31}$$

En développant, et en notant que :

- ${}^{k}a_k = [0\ 0\ 1]^T$
- ${}^{k}\mathbf{L}_k = {}^{k}\mathbf{P}_n$

on obtient :

$$ {}^{i}\mathbf{J}_{n;k} = \begin{bmatrix} \sigma_K\, {}^{i}a_K + \bar{\sigma}_k(-{}^{K}P_{ny}\, {}^{i}s_K + {}^{K}P_{nx}\, {}^{i}n_K) \\ \bar{\sigma}_K\, {}^{i}a_K \end{bmatrix} \qquad (1.32) $$

où ${}^{k}P_{nx}$ et ${}^{k}P_{ny}$ sont respectivement les composant x et y du vecteur ${}^{k}P_n$.

A partir de cette relation, on peut calculer la K^{ieme} colonne de ${}^{n}\mathbf{J}_n$:

$$ {}^{n}\mathbf{J}_{n;k} = \begin{bmatrix} \sigma_K\, {}^{n}a_K + \bar{\sigma}_k(-{}^{K}P_{ny}\, {}^{i}s_K + {}^{K}P_{nx}\, {}^{n}n_K) \\ \bar{\sigma}_K\, {}^{n}a_K \end{bmatrix} \qquad (1.33) $$

Les éléments de la colonne ${}^{n}\mathbf{J}_{n\,;\,k}$ se calculent à partir des éléments de la matrice ${}^{k}\mathbf{T}_n$, résultats intermédiaires obtenus lors du calcul du MGD.

De façon analogue, la K^{ieme} colonne de ${}^{0}\mathbf{J}_n$ s'écrit :

$$ {}^{0}\mathbf{J}_{n;k} = \begin{bmatrix} \sigma_K\, {}^{0}a_K + \bar{\sigma}_k\, {}^{0}\hat{a}_K({}^{0}P_n - {}^{0}P_K) \\ \bar{\sigma}_K\, {}^{0}a_K \end{bmatrix} \qquad (1.34) $$

Dans ce cas, les éléments de la colonne k s'obtiennent à partir de ceux de la matrice ${}^{0}\mathbf{T}_n$ et du vecteur ${}^{0}\mathbf{P}_n$. On doit donc calculer les matrices ${}^{0}\mathbf{T}_k$ pour k=1,…,n.

- REMARQUE : –Pour trouver le jacobien ${}^{E}\mathbf{J}_E$ définissant les vitesses du repère outil $\mathbf{R}_E$, on peut utiliser l'équation [5.9] après avoir remplacé $\mathbf{L}_{k,n}$ par $\mathbf{L}_{k,E}$, soit utiliser la relation exprimant la transformation entre torseurs cinématiques entre repères :

$$ {}^{E}\mathbf{J}_E = \begin{bmatrix} {}^{E}A_n & 0_3 \\ 0_3 & {}^{E}A_n \end{bmatrix} \begin{bmatrix} I_3 & -{}^{n}\hat{\mathrm{P}}_E \\ 0_3 & I_3 \end{bmatrix} {}^{n}\mathbf{J}_n = {}^{E}\mathrm{T}_n\, {}^{n}\mathbf{J}_n $$

Où ${}^{E}\mathrm{T}_n$ est la matrice (6x6) de transformation entre torseurs. Pour plus de détails voir **[1]**.

On utilisant les relations ci-dessus et ceux du modèle géométrique on peut procéder au calcule de la matrice jacobienne de base du robot SCARA (type RRRP) , ce qui donne:

Type RRRP: $(\sigma_1=0;\ \sigma_2=0;\ \sigma_3=0;\ \sigma_4=1)$,on sais dêja que: ${}^{0}a_i = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ $\forall\ i=1,..,4.$

$$ {}^{0}\mathrm{T}_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \Rightarrow {}^{0}\mathrm{P}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} $$

$$ {}^{0}T_2 = \begin{bmatrix} C12 & -S12 & 0 & a2.C1 \\ S12 & C12 & 0 & a2.S1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \Rightarrow {}^{0}P_2 = \begin{pmatrix} a2.C1 \\ a2.S1 \\ 0 \end{pmatrix} $$

$$ {}^{0}T_3 = \begin{bmatrix} C123 & -S123 & 0 & a2.C1 + a3.C12 \\ S123 & C123 & 0 & a2.S1 + a3.S12 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \Rightarrow {}^{0}P_3 = \begin{pmatrix} 0 \\ 0 \\ d4 \end{pmatrix} $$

jusqu'a la dernière articulation qui correspond a ${}^{0}T_4$ (déjà calculer auparavant) $\Rightarrow$

$$ {}^{0}P_4 = \begin{pmatrix} a2.C1 + a3.C12 \\ a2.S1 + a3.S12 \\ d4 \end{pmatrix} $$

$$ {}^{0}J_{4,1} = \begin{bmatrix} {}^{0}a_1 \times \left({}^{0}P_4 - {}^{0}P_1\right) \\ {}^{0}a_1 \end{bmatrix} = \begin{bmatrix} -a2.S1 - a3.S12 \\ a2.C1 + a3.C12 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}; \qquad {}^{0}J_{4,3} = \begin{bmatrix} {}^{0}a_3 \times \left({}^{0}P_4 - {}^{0}P_3\right) \\ {}^{0}a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}; $$

$$ {}^{0}J_{4,2} = \begin{bmatrix} {}^{0}a_2 \times \left({}^{0}P_4 - {}^{0}P_2\right) \\ {}^{0}a_2 \end{bmatrix} = \begin{bmatrix} -a3.S12 \\ a3.C12 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}; \qquad {}^{0}J_{4,4} = \begin{bmatrix} {}^{0}a_4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; $$

On déduit la matrice jacobienne directement, ce qui donne :

$$ \mathbf{J} = \begin{bmatrix} -a2.S1 - a3.S12 & -a3.S12 & 0 & 0 \\ a2.C1 + a3.C12 & a3.C12 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} $$

Pour ce qui concern l'etude de l'espace de travail du robot plus en detail , voir **[1]**.

1.4.6. Le modèle cinematique inverse

L'objectif du modèle cinématique inverse est de calculer à partir d'une configuration $\mathbf{q}$ donnée, les vitesses articulaires $\dot{q}$ qui assurent au repère terminal une vitesse opérationnelle $\dot{X}$ imposée. Pour obtenir le modèle cinématique inverse, on inverse le modèle cinématique direct en résolvant un système d'équations linéaires. La mise en œuvre peut être faite de façon analytique ou numérique :

- la solution analytique a pour avantage de diminuer considérablement le nombre d'opérations, mais on doit traiter séparément tous les cas singuliers.

- les méthodes numériques sont plus générales, la plus répandue étant fondée sur la notion de pseudo-inverse : les algorithmes traitent de façon unifiée les cas réguliers, singuliers et redondant. Elles nécessitent un temps de calcul relativement important.

Nous présentons dans cette section les techniques et méthodologies à mettre en œuvre pour établir un modèle cinématique inverse dans les cas réguliers et singuliers .**[1]**

1.4.6.1. Forme générale du modèle cinématique

Quelle que soit la méthode utilisée pour décrire les coordonnées opérationnelles, le modèle cinématique direct peut être mis sous la forme :

$$\dot{X}=\begin{bmatrix}\Omega_p & 0_3\\ 0_3 & \Omega_r\end{bmatrix}\begin{bmatrix}{}^0A_i & 0_3\\ 0_3 & {}^0A_i\end{bmatrix}\begin{bmatrix}I_3 & -{}^i\hat{L}_{j,n}\\ 0_3 & I_3\end{bmatrix}{}^i\mathbf{J}_{n,j}\,\dot{q} \qquad (1.35)$$

avec :

- Ω_p est la matrice de transformation entre le vecteur $\dot{X}_p$ et le vecteur ${}^0\mathbf{V}_n$
- Ω_r est la matrice de transformation entre le vecteur $\dot{X}_r$ et le vecteur ${}^0\omega_n$

t.q :

$$\begin{bmatrix}\dot{X}_p\\ \dot{X}_r\end{bmatrix}=\begin{bmatrix}\Omega_p & 0_3\\ 0_3 & \Omega_r\end{bmatrix}\begin{bmatrix}{}^0V_n\\ {}^0\omega_n\end{bmatrix}$$ / Xp et Xr étant une représentation quelconque dans le repère $\mathbf{R}_0$

ou, sous forme condensée :

$$\dot{X}={}^0J_X\dot{q} \qquad (1.36)$$

Etant donné la simplicité des éléments de ${}^i\mathbf{J}_{n,j}$ comparés à ceux de ${}^0\mathbf{J}_x$, il est préférable de chercher une solution analytique à partît de l'expression [1.35]. Celle-ci s'écrit encore :

$${}^i\dot{X}_{n,j}={}^iJ_{n,j}\dot{q} \qquad (1.37)$$

Si on utilise les coordonnées cartésiennes pour décrire la position, alors $\Omega_p = \mathbf{I}_3$. Pour alléger l'écriture, nous adopterons la forme suivante de l'équation [1.37] :

$$\dot{X} = J\,\dot{q} \qquad (1.38)$$

1.4.6.2. Modèle cinématique inverse dans le cas régulier

Dans ce cas, la matrice jacobienne J est carrée d'ordre n et son déterminant est non nul. Deux méthodes de calcul peuvent être mises en œuvre.

A). Première méthode

On calcule $\mathbf{J}^{-1}$, la matrice inverse de **J,** qui permet de déterminer les vitesses articulaires $\dot{q}$ grâce à la relation :

$$\dot{q} = J^{-1}\dot{X} \tag{1.39}$$

Lorsque la matrice **J** a la forme suivante :

$$\mathbf{J} = \begin{bmatrix} A & 0 \\ B & C \end{bmatrix} \tag{1.40}$$

Les matrices **A** et **C** étant carrées inversibles, il est facile de montrer que l'inverse de cette matrice s'écrit :

$$\mathbf{J}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{bmatrix} \tag{1.41}$$

La résolution du problème se ramène donc à l'inversion, beaucoup plus simple, de deux matrices de dimension moindre. Lorsque le robot manipulateur possède six degrés de liberté et un poignet de type rotule, la forme générale de **J** est celle de la relation (1.40), **A** et **C** étant de dimension (3x3), ce qui n'est pas le cas de notre robot.**[1]**

B). Seconde méthode

Dans cette méthode, on tient compte d'une éventuelle forme particulière de la matrice **J** permettant de réduire le nombre d'inconnues. Cette méthode donne, dans la plupart des cas, des solutions nécessitant moins d'opérations. Prenons par exemple, le cas d'un robot manipulateur à poignet rotule dont le jacobien a, comme on l'a déjà dit, a la structure de l'équation (1.40) :

$$\begin{bmatrix} \dot{X}_p \\ \dot{X}_r \end{bmatrix} = \begin{bmatrix} A & 0_3 \\ B & C \end{bmatrix} \begin{bmatrix} \dot{q}_a \\ \dot{q}_b \end{bmatrix} \tag{1.42}$$

A et **C** étant des matrices carrées de dimension (3x3), inversibles en dehors des positions. La solution $\dot{q}$ est donnée par :

$$\begin{cases} \dot{q} = A^{-1}\dot{X}_a \\ \dot{q} = C^{-1}\left[\dot{X}_b - B\dot{q}_a\right] \end{cases} \tag{1.43}$$

Solution qui est beaucoup plus simple que celle obtenue avec la première méthode.

1.4.6.3. Solution au voisinage des position singulières

Lorsque le robot et non redondant, les singularités d'ordre un sont solution de det(**J**)=0. Dans le cas redondant, elles sont données par det($\mathbf{JJ}^T$)=0. Les singularités d'ordre supérieur sont déterminées à partir des configurations singulières d'ordre un.

En une position singulière donnée, il n'existe pas de vitesse articulaires qui puisse engendrer une vitesse opérationnelle suivant cette dernière direction avec le modèle cinématique inverse classique car il peut donner des vitesses articulaires importantes, incompatibles avec les caractéristiques des actionneurs.

Le vecteur de vitesse $\dot{X}$ est constitué en général d'un ensemble formant le vecteur $I(\mathbf{J})$ et d'un vecteur orthogonal appartenant à $I(\mathbf{J})^{\perp}$

A). Utilisation de la pseudo inverse

Il est courant d'utiliser la pseudo inverse $\mathbf{J}^{+}$ de la matrice **J** (voir **[1]** pour la méthode de calcul la plus classique de la pseudo inverse)

$$\dot{q}=J^{+}\dot{X} \qquad (1.44)$$

Cette solution, minimise la norme euclidienne $\|\dot{q}\|^2$ et la norme de l'erreur $\|\dot{X}-J\dot{q}\|^2$. Dans une configuration singulière, on distingue les deux cas particuliers suivants :

- $\dot{X}$ appartient uniquement à $I(\mathbf{J})$. Alors, la solution (1.44) est exacte et l'erreur est nulle bien que l'inverse $\mathbf{J}^{-1}$ ne soit pas définie ;
- $\dot{X}$ appartient uniquement à $I(\mathbf{J})^{\perp}$. Alors, la solution (1.44) donne $\dot{q}=0$. Si la consigne de vitesse suivante est définie selon cette direction, le robot se bloque et il faut lui définir des stratégies de déblocage ;

On a montré que la pseudo inverse donne une solution discontinue lors du passage par une configuration singulière, on peut la vérifié par la formulation (1.45). En effet, en dehors des singularités :

$$\dot{q}=\sum_{i=1}^{m}\frac{1}{\sigma_i}V_i U_i^{T}\dot{X} \qquad (1.45)$$

σ_i : sont les valeurs singulières de **J** et sont égales aux racines carrées des valeurs propres du produits $\mathbf{J}^{T}\mathbf{J}$.

En s'approchant d'une singularité, σ_{min} devient petit ce qui conduit à une vitesse $\dot{q}$ élevée. Et quand elle s'annule elle n'est plus prise en compte, la somme s'arrête.

1.5. Modèle dynamique du robot Scara

le robot SCARA est un robot série possédant quatre degrés de liberté dont trois sont rotoïde et le quatrième prismatique comme le montre la figure 1.13 :

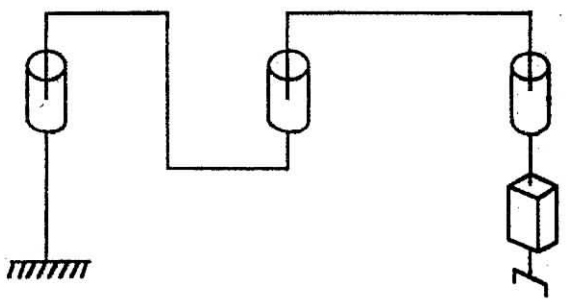


Figure 1.13. *Robot de type SCARA.*

Dans cette partie nous présentons le principe de calcul du modèle dynamique d'un robot Scara, Ce calcul se base sur le formalisme de Lagrange cité au annexe 1.

Le robot Scara a généralement pour tache le déplacement d'objet d'un point a un autre appartenant tous deux à son espace de travail, les deux premières articulations rotoïdes permettent le positionnement de l'organe terminal tandis que la troisième articulation rotoïde et l'articulation prismatique permettent respectivement l'orientation et le déplacement verticale de l'organe terminal. Pour cette raison deux modèles ont été calculés, le premier modèle tient compte des trois articulations rotoïdes, cela se justifie par le faite que la dynamique de la translation est indépendante des autre articulation, le détail du calcul sera donné. Le deuxième modèle tient compte seulement des articulations de positionnement, ce modèle plus simple nous servira pour effectuer les différentes simulations.

1.5.1. Premier modèle

Le premier modèle calculé est celui du robot représenté à la figure1.14, nous commencerons par la détermination des vitesses ainsi que l'énergie cinétique. A partir de cette énergie, qui est à la base du formalisme de Lagrange, nous obtenons le modèle dynamique de notre robot.

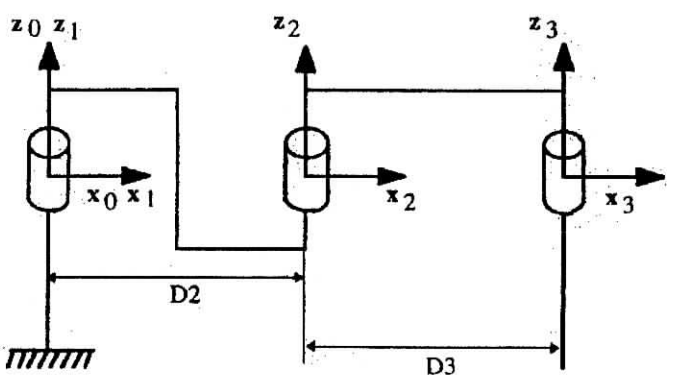


Figure 1.14. *Robot à trois degré de liberté.*

1.5.1.1. Calcul des vitesses

La détermination des vitesses est indispensable pour le calcul de l'énergie cinétique. A partir des relations (A1.13) et (A1.14), nous obtenons l'expression des vitesse ${}^{j}\omega_j$ et ${}^{j}\mathbf{V}_j$ de chaque articulation dans le repère $\mathbf{R}_j$ qui lui est lié, les vitesses ainsi obtenues sont les suivantes :

$$
\begin{aligned}
&{}^{0}\omega_0 = 0 \\
&{}^{1}\omega_1 = {}^{1}A_0\,{}^{0}\omega_0 + \dot{q}_1\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\\dot{q}_1\end{bmatrix} \\
&{}^{2}\omega_2 = {}^{2}A_1\,{}^{1}\omega_1 + \dot{q}_2\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\\dot{q}_1+\dot{q}_2\end{bmatrix} \\
&{}^{3}\omega_3 = {}^{3}A_2\,{}^{2}\omega_2 + \dot{q}_3\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\\dot{q}_1+\dot{q}_2+\dot{q}_3\end{bmatrix}
\end{aligned}
\qquad (1.46)
$$

Et nous obtenons aussi :

$$
\begin{aligned}
&{}^{0}V_0 = 0 \\
&{}^{1}V_1 = {}^{1}A_0({}^{0}V_0 + {}^{0}\omega_0 x\, {}^{0}P_1) = 0 \\
&{}^{2}V_2 = {}^{2}A_1({}^{1}V_1 + {}^{1}\omega_1 x\, {}^{1}P_2) = \begin{bmatrix}S2\,\dot{q}_1\,D2\\C2\,\dot{q}_1\,D2\\0\end{bmatrix} \\
&{}^{3}V_3 = {}^{3}A_2({}^{2}V_2 + {}^{2}\omega_2 x\, {}^{2}P_3) = \begin{bmatrix}C3\,S2\,\dot{q}_1\,D2 + S3[(C2\,D2+D3)\dot{q}_1 + D3\,\dot{q}_2]\\-S3\,S2\,\dot{q}_1\,D2 + C3[(C2\,D2+D3)\dot{q}_1 + D3\,\dot{q}_2]\\0\end{bmatrix}
\end{aligned}
\qquad (1.47)
$$

1.5.1.2. Calcul de l'énergie cinétique

En utilisant la relation (A1.12) nous obtenons l'expression de l'énergie cinétique de chaque articulation suivante :

$$E_1 = \frac{1}{2} ZZ_1\,\dot{q}_1^{\,2}$$

$$E_2 = \frac{1}{2}\left[ZZ_2(\dot{q}_1+\dot{q}_2)^2 + M_2\,D2^2\dot{q}_1^{\,2} + 2\,MX_2\,C2\,D2\,\dot{q}_1\,(\dot{q}_1+\dot{q}_2)\right]$$

$$E_2 = \frac{1}{2}[ZZ_3(\dot{q}_1+\dot{q}_2+\dot{q}_3)^2 + M_3[(D2^2+D3^2)\,\dot{q}_1^{\,2} + 2C2\,D2\,D3\,\dot{q}_1^{\,2} + 2C2\,D2\,D3\,\dot{q}_1\,\dot{q}_2 + 2\,D3^2\,\dot{q}_1\,\dot{q}_2 + D3^2\,\dot{q}_2^{\,2}]]$$

Connaissant l'expression des énergies cinétiques, nous obtenons la matrice d'inertie qui est donné par :

$$A = \begin{bmatrix} I_{a1} + ZZ_1 + ZZ_2 + M_2 D2^2 + 2\,MX_2 C2\,D2 + ZZ_3 + M_3(D2^2 + D3^2 + 2\,C2\,D2\,D3) & ZZ_2 + MX_2 C2\,D2 + ZZ_3 + M_3(C2\,D2\,D3 + D3^2) & ZZ_3 \\ ZZ_2 + MX_2 C2\,D2 + ZZ_3 + M_3(C2\,D2\,D3 + D3^2) & I_{a2} + ZZ_2 + ZZ_3 + M_3 D3^2 & ZZ_3 \\ ZZ_3 & ZZ_3 & I_{a3} + ZZ_3 \end{bmatrix}$$

La matrice C se calcul a partir du symbole de *Christophell*, nous obtenons :

$$C = \begin{bmatrix} -(MX_2\,S2\,D2 + M_3 S2\,D2\,D3)\dot{q}_2 & -(MX_2\,S2\,D2 + M_3 S2\,D2\,D3)(\dot{q}_1 + \dot{q}_2) & 0 \\ (MX_2\,S2\,D2 + M_3 S2\,D2\,D3)\dot{q}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

1.5.2. Second modèle

Comme il a été cité précédemment, ce second modèle est celui d'un robot à deux degrés de liberté comme le montre la figure suivante :

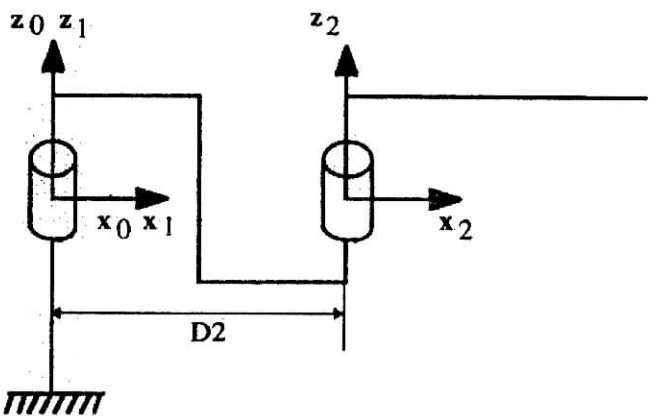


Figure 1.15. *Robot à deux degrés de liberté.*

Le modèle1 de ce robot est donné sans calcul, sa détermination est basée sur les mêmes étapes que celles décrites précédemment, le modèle de ce robot est représenté par la formule suivante :

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + \Gamma_f + T_{m0}$$

Les éléments du modèle sont donnés par :

$$A = \begin{bmatrix} ZZ1 + ZZ2 + M2\,D2^2 + 2MX_2 D2\,C2 + I_{a1} & ZZ2 + M2\,D2^2 + MX_2 D2\,C2 \\ ZZ2 + M2\,D2^2 + MX_2 D2C2 & ZZ2 + I_{a2} \end{bmatrix}$$

$$C = \begin{bmatrix} -MX_2\,S2\,D2\,\dot{q}_2 & -MX_2\,S2\,D2\,(\dot{q}_1 + \dot{q}_2) \\ MX_2\,S2\,D2\,\dot{q}_1 & 0 \end{bmatrix}$$

$$\Gamma_f = \begin{bmatrix} Fv_1\dot{q}_1 + Fs_1 sign(\dot{q}_1) \\ Fv_2\dot{q}_2 + Fs_2 sign(\dot{q}_2) \end{bmatrix}$$

T_{m0} est le couple dû à la charge, il est donné par la formule suivante :

$$T_{m0} = m_0\,J^T(q)[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g]$$

1.5.3. Modèle du robot pour la simulation

Pour la simulation nous avons considéré le robot à deux degrés de liberté. Les deux articulations sont considéré comme deux barres ayant une longueur L_i et une masse m_i se trouvant à son extrémité, le modèle ainsi obtenue se met sous la forme :

$$\Gamma = A(q)\ddot{q} + N(q,\dot{q}) + \Gamma_f + T_{m0}$$

avec

$$A(q) = \begin{bmatrix} (m_1 + m_2)L_1^2 + m_2 L_2^2 + 2m_2\,L_1 L_2\,C2 + I_{a1} & m_2 L_2^2 + m_2\,L_1 L_2\,C2 \\ m_2 L_2^2 + m_2\,L_1 L_2\,C2 & m_2 L_2^2 + I_{a2} \end{bmatrix}$$

$$N(q,\dot{q}) = C(q,\dot{q})\dot{q} = \begin{bmatrix} -m_2\,L_1\,L_2\,(2\dot{q}_1\dot{q}_2 + \dot{q}_1^2)S2 \\ m_2\,L_1\,L_2\,\dot{q}_1^2\,S2 \end{bmatrix}$$

$$\Gamma_f = \begin{bmatrix} Fv_1\dot{q}_1 + Fs_1 sign(\dot{q}_1) \\ Fv_2\dot{q}_2 + Fs_2 sign(\dot{q}_2) \end{bmatrix}$$

$$T_{m0} = m_0 J^T(q)[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g]$$

avec $g = [0\ 0]^T$

La matrice jacobienne du robot Scara à deux degrés de liberté est donnée par :

$$J(q) = \begin{bmatrix} -L_1 S1 - L_2 S12 & -L_2 S12 \\ L_1 C1 + L_2 C12 & L_2 C12 \end{bmatrix}$$

Les valeurs numériques des différents paramètres du robot Scara sont les suivantes **[5]** :

m_1=15.91 kg
m_2=11.36 kg
L_1=L_2=0.432 m
F_{v1}=F_{v2}=7 x 10^{-4}
F_s1=F_{s2}=10^{-3}
I_{a1}=I_{a2}=7.06 x 10^{-6}

1.6. Conclusion

Dans le présent chapitre, nous nous somme consacrés à l'étude de l'aspect géométrique, cinématique et dynamique ; pour cela nous avons adopté la convention de Denavit-Hartenberg avec la modification de Khalil et Kleinfinger, le calcul du MGD est basée sur la transformation homogène qui nous permet d'avoir la position du repère terminal par rapport a un repère de référence, pour la commande, c-à-d le calcul de **q=f(x,y,z)** l'inversion qui consiste à la détermination MGI en utilisant la méthode de Paul utilisée pour la plupart des robot industriel.

L'étude des espaces de travail des robots industriels et de leur configuration de singularité irréversible constituant un grand problème dans l'industrie, nous a mener a la détermination du MCD basée sur la matrice jacobienne de base, du MCI qui est obtenu directement par inversion matricielle simple du MCD dans le cas régulier ou par la pseudo-inverse dans le cas singulier.

La commande des actionneurs constituant les robots et la génération des trajectoire se base essentiellement sur le model dynamique, ce dernier peut être obtenue à partir de différent formalisme tel que Lagrange et Newton-Euler, le plus utilisé est le formalisme de Lagrange utilisant l'énergie cinétique et potentielle et les inerties de la structure, ces inerties doivent être minimales afin d'optimiser l'énergie que doit fournir les actionneurs en même temps garantir une stabilité au repos.

CHAPITRE II :
DIMENSIONNEMENT DU ROBOT SCARA ET DE CES ACTIONNEURS

2.1. Introduction

Regardons les principes guidant la conception d'un robot ou de façon plus générale. Un des problèmes principaux de l'utilisation des robots est qu'on désire amener un outil a une certaine position dans l'espace mais que dans la grande majorité des cas, on ne vérifie pas la position finale de l'outil. Ce qu'on mesure, c'est la position des actionneurs (moteurs, vérins, etc...). On peut dire en quelque sorte que l'extrémité du robot soit contrôlée en boucle ouverte bien que les actionneurs soient contrôlés en boucle fermée.

En supposant que les bras et les roulements sont infiniment rigides, la position de l'outil sera connue par les matrices de transformation du robot mais en pratique il y a toujours une certaine flexibilité. On voit encore aujourd'hui surtout des robots contrôlés aux articulations. Dans certaines applications, on ne peut s'empêcher d'avoir des bras flexibles, mais dans la plupart des cas il faut s'arranger pour minimiser cette flexibilité.

La rigidité de la structure est un aspect très important pour la plupart des robots **[10]**. Essentiellement, on conçoit la structure en fonction de la précision minimale demandée dans les spécifications initiales. Ici il est important de parler de deux types de rigidité. Une rigidité statique qui est essentiellement le calcul de déflexions dues `a une charge statique et de la rigidité dynamique qui est le calcul des fréquences naturelles du système. La première est directement reliée à la précision de positionnement exigée. La deuxième va influencer le comportement dynamique du système, a savoir si le système risque de bouger en oscillant ou non. En fait chaque axe est contrôlé par un système asservi caractérise par une certaine largeur de bande.

En pratique, on cherchera à avoir la fréquence naturelle minimum d'environ 3 fois la largeur de bande **[10]**.

En réalité du point vue statique, c'est la rigidité qui est importante, en effet on a que, de façon générale

$$k = \frac{\Delta F}{\Delta X} \qquad (2.1)$$

Donc, pour un effort donné, plus k est grand, plus les déflexions sont faibles. Par contre, en dynamique, ce qu'on désire c'est une grande fréquence naturelle, or, de façon générale, on aura

$$\omega^2 = \frac{k}{m} \qquad (2.2)$$

Il faut que la rigidité soit grande ou la masse petite ou les deux. Or les deux sont parfois incompatibles. En effet, si on décide par exemple de prendre une membrure en acier plutôt qu'en aluminium de manière a ce qu'elle soit plus rigide, on aura une membrure plus lourde ce qui ne sera pas nécessairement mieux.

2.2. L'étude des articulations en flexion :

Soit les dimensions proposées à une réalisation structurelle de notre robot (figure 2.1) :

- *1er Articulation :* M1=15,91 **Kg** ; L1=0,432 **m** ; H1=0,1315 **m ;** B1=0,1 **m**
- 2éme *Articulation :* M2=11,36 **Kg** ; L2=0.,32 **m** ; H2=0,1 **m ;** B2= 0,094 **m**

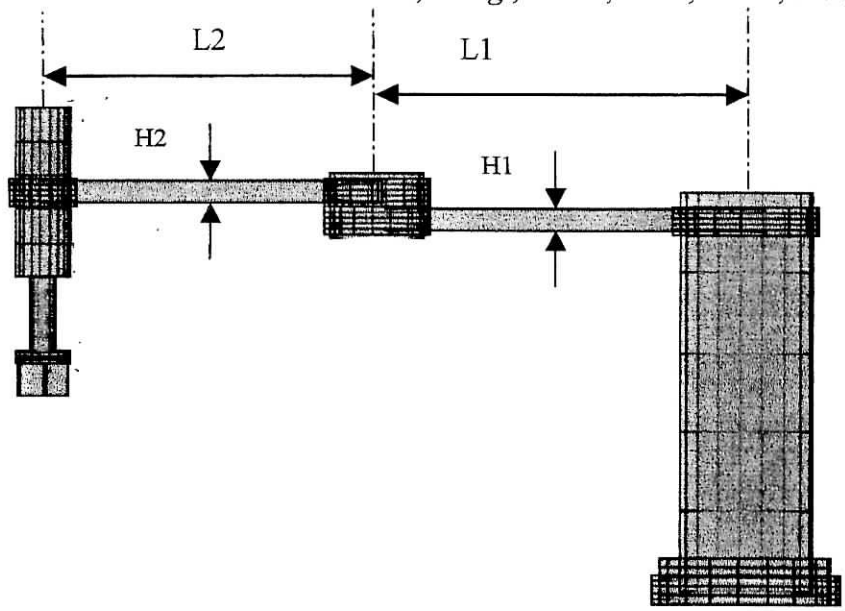


Figure 2.1. *dimensions réelles de la structure du robot SCARA.*

Pour une longueur donnée, on voudrait avoir un matériau ayant un bon rapport E/ρ , appelé la rigidité spécifique. Le tableau suivant donne certaines valeurs de métaux connus **[10]** :

Matérian	$E.(GPa)$	$\rho.(10^3 kg/m^3)$	$E/\rho.(10^7 m^2/s^2)$
Carbure de Bore	450	2.4	19.9
Berylium	290	1.9	15.3
Alumine	300-400	3.7-3.8	7.9-11
Carbure de Titane	400-450	5.7-6.0	7-9
Carbure de Tungsten	550	16	3.4
Acier	210	7.8	2.7
Aluminium	72	2.8	2.6
Magnesium	45	1.9	2.4

Tableau 2.1. *Rigidité spécifique des matériaux de construction en robotique*

L'utilisation de matériaux composites semble très appropriée pour réduire la rigidité spécifique.

En flexion on trouve une rigidité qui dépend du mode de fixation, pour une poutre encastrée, on a (Figure 2.2) :

$$\delta = \frac{F.L^3}{3.E.I} \tag{2.3}$$

$$k_{flexion} = \frac{F}{\delta} = \frac{3.E.I}{L^3} \tag{2.4}$$

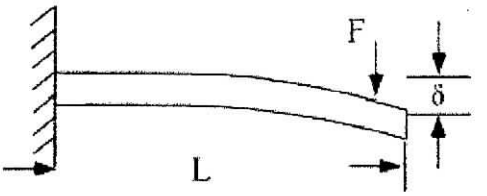


Figure 2.2. *Flexion d'une poutre en encastrement.*

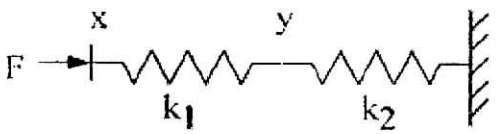
Dans le cas d'une poutre rectangulaire, on a que :

$$I = \frac{b.h^3}{12} \Rightarrow k_{fl} = \frac{3.E.b.h^3}{12.L^3} = \frac{E.b.h}{L}\left(\frac{h}{2.L}\right)^2 \tag{2.5}$$

2.3. rigidité équivalente

Les robots sont des structures à plusieurs membrures **[10]**. Lorsqu'on s'intéresse à la rigidité d'un ensemble complexe, on le décompose en structures simples et on évalue ensuite la rigidité de l'ensemble à partir des valeurs individuelles. Pour analyser les différentes configurations, on va assimiler les structures flexibles à des ressorts

Lorsqu'on a deux ressorts en série, on peut trouver facilement la rigidité équivalente. On trouve que :



$f = k_1 . (x - y)$
$k_1 . (x - y) = k_2 . y$

d'où

$$f = \frac{k_1.k_2}{k_1 + k_2}.x = k_{eq}.x \tag{2.6}$$

On remarque l'analogie avec un système électrique ou on a des résistances en parallèles. On pourrait d'ailleurs utiliser le résultat suivant :

$$\frac{1}{k_{eq}} = \frac{1}{k_1} + \frac{1}{k2} \tag{2.7}$$

Pour le calcul des fréquences propres de notre structure, on utilisera son modèle dynamique réduit, en considérons les articulations comme étant des poutres en série modélisé chacune comme un ressort et une masse en négligeant l'amortissement (cette approximation change la réponse réelle du système mais pas le calcul des fréquences propres), nous obtenons :

$$\begin{aligned} f - k_1.(x - y) &= m_1.\ddot{x} \\ k_1.(x - y) &= m_2.\ddot{y} \end{aligned} \qquad (2.8)$$

on trouve en faisant la transformée de Laplace :

$$\begin{aligned} F + k_1.Y &= (m_1.s^2 + k_1).X \\ k_1.X &= (m_2.s^2 + k_1 + k_2).Y \end{aligned} \qquad (2.9)$$

en isolant X, on trouve :

$$G(s) = \frac{X}{F} = \frac{m_2.s^2 + k_1 + k_2}{m_1 m_2.s^4 + (k_1.m_2 + (k_1 + k_2).m_1).s^2 + k_1 k_2} \qquad (2.10)$$

Prenons les valeurs numériques de notre structure on trouve, après calculs :
(Il faut faire attention car le calcul est fait en considérons pour une force de flexion latérale).

$I1=1{,}0961 .10^{-5} \Rightarrow k_1 = 2{,}9623 . 10^7$ N/m.
$I2=6.9215 .10^{-6} \Rightarrow k_2 = 1{,}875 . 10^7$ N/m.

Les racines du dénominateur donnent :

$\omega 1=1.9768.10^3$ rad/s; $\omega 2=0.8879.10^3$ rad/s;

- Ces fréquences représentent en quelques sorte la limite vibratoire que doit vérifier le régulateur, toutes fois, dépasser ces fréquences peut rendre instable le système, une valeur $\omega_j=\omega_{max} / 2$ représente un bon compromis ;

- Si la plus petite fréquence naturelle de la structure est du même ordre de grandeur que la largeur de bande du régulateur, on risque d'avoir un mouvement avec vibrations à moins d'avoir un amortissement adéquat.

2.4. Limites articulaires du robot SCARA (butées mécaniques)

Le robot SCARA présente des limitations en vitesses liées directement aux types de moteurs utilisés, pour plus de détails voir chapitre sur le dimensionnement et étude du moteur **PITTMAN GM9236**, ainsi que des limitations mécaniques influençant directement l'espace de travail du robot appelées plus précisément butées mécaniques (Figure 2.3), qui sont :

$$\begin{cases} -150 \le \theta_1 \le 150 ; & \text{Angle de rotation de la } 1^{cr} \text{ articulation} \\ -150 \le \theta_2 \le 150 ; & \text{Angle de rotation de la } 2^{éme} \text{ articulation} \\ -135 \le \theta_3 \le 135 ; & \text{Angle de rotation de la } 3^{éme} \text{ articulation} \\ -14 \le Z_4 \le -29 ; & \text{Translation de la } 4^{éme} \text{ articulation} \end{cases}$$

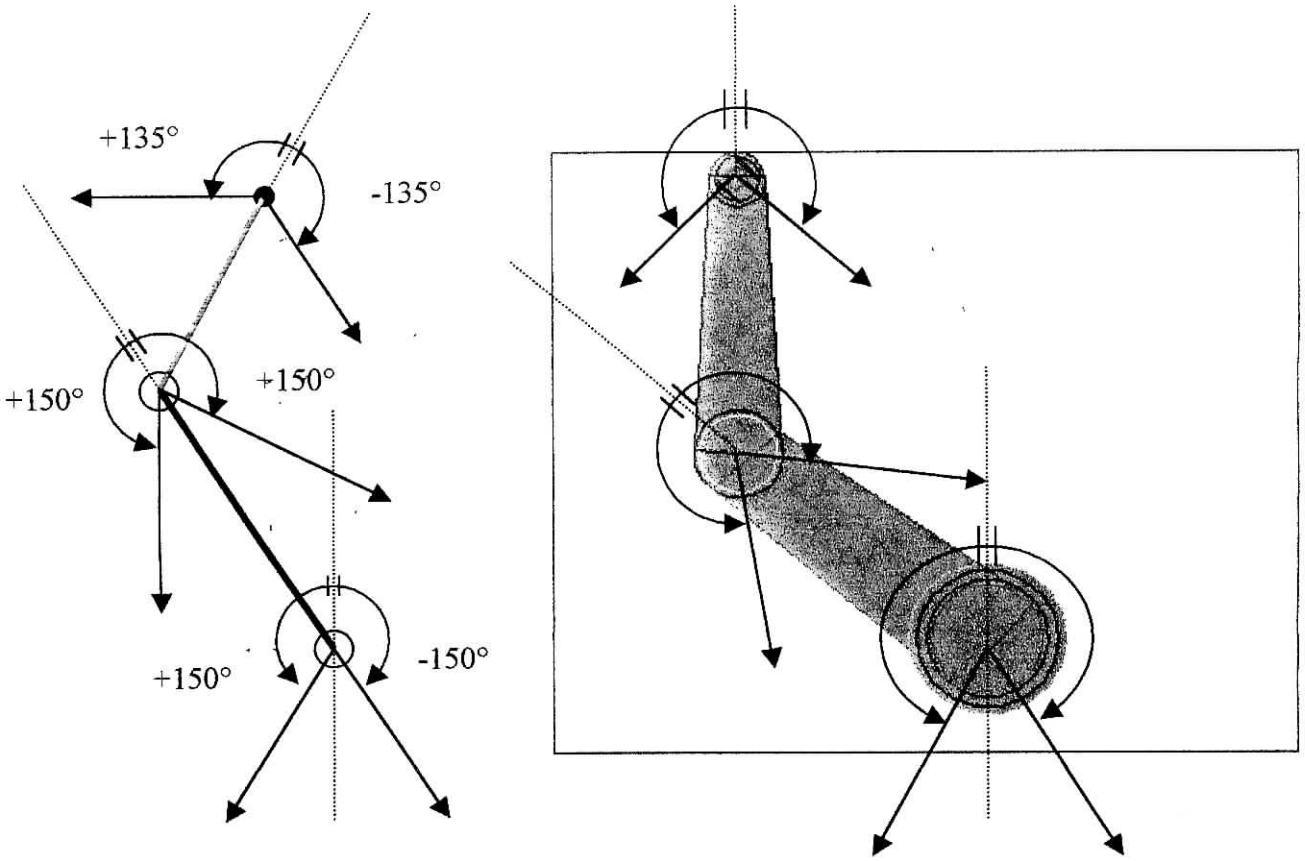


Figure 2.3. *Représentation des butées mécaniques.*

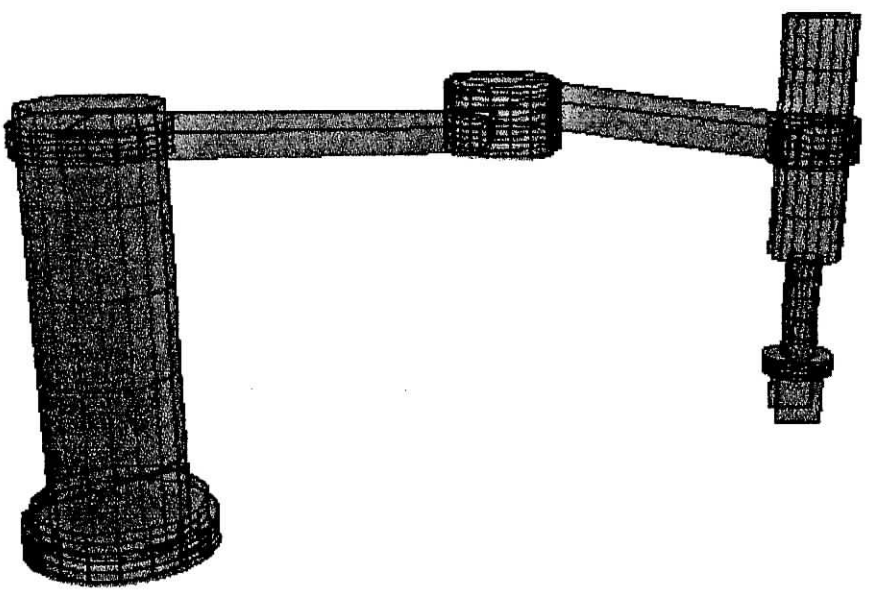


Figure 2.4. *Vue 3D du robot SCARA.*

2.5. Conclusion

Nous avons établie dans cette partie du chapitre les caractéristiques de la structure mécanique du robot SCARA soit : dimensions, masses, matières, inerties, compliances, charges maximales, fréquences de résonance et en dernier lieu les butées mécaniques que présente la structure, tout ces paramètres sont utilisés pour le dimensionnement du régulateur qui se charge de géré le mouvement de suivi d'une trajectoire de consigne sans se préoccuper des flexibilités articulaires ou des vibrations mécaniques et pour permettre la simulation de la réponse du système de réglage, voir chapitres 3 et 4.

2.6. Introduction

Un robot réalise des fonctions mécaniques telles qu'un déplacement ou un positionnement. Pour cela, il a besoin d'un actionneur qui va réaliser à la fois une transmission d'information et une conversion d'énergie **[12]**. Plusieurs technologies existent : on rencontre des actionneurs électriques, pneumatiques et hydrauliques. Les premiers sont le plus souvent des moteurs en rotation, un mécanisme permettant éventuellement de modifier la nature du mouvement. Ce sont ces machines qui nous intéressent ici.

Les moteurs employés font appel aux mêmes principes que ceux qu'on utilise en électrotechnique classique, mais leurs caractéristiques et leurs technologies sont différentes. Tout d'abord, la puissance en général modeste des machines fait qu'on fait souvent appel à des aimants permanents plutôt qu'à des inducteurs bobinés, ce qui simplifie la réalisation et élimine les pertes par effet joule correspondantes. Ensuite, on ne cherche pas à optimiser les mêmes paramètres. Les grandeurs liées à la conversion d'énergie (puissance, rendement) sont certes importantes, mais les grandeurs liées à l'information (précision, rapidité de réponse) sont primordiales. Ainsi, on minimise l'inertie des parties tournantes en adoptant des structures particulières ou une géométrie adaptée pour réaliser des moteurs à réponse rapide.

Les performances d'un actionneur électrique sont intimement liées à celles de son environnement : le convertisseur d'énergie et sa commande électronique, l'éventuel asservissement avec ses capteurs et les mécanismes associés à la charge. Il est important d'en tenir compte lors d'une comparaison de coût entre plusieurs solutions

Dans cette partie nous allons cité les avantage des actionneurs électrique, décrire en détail le moteur à courant continue en commençant par sa constitution ainsi que ses caractéristique mécanique et électrique.

2.7. Avantages des actionneurs électriques

Par rapport à leurs concurrents hydrauliques et pneumatiques, les actionneurs électriques présentent un certain nombre d'avantages parmi lesquels **[12]**:

- une énergie facilement disponible, soit à partir du secteur, soit à partir de batteries pour les engins autonomes ;

- une adaptation aisée de l'actionneur et de sa commande du fait de la nature électrique de l'ensemble des grandeurs.

Les progrès récents ont permis un élargissement du domaine d'emploi des actionneurs électriques. On peut citer :

- l'amélioration des performances des moteurs grâce en particulier à la disponibilité d'aimants plus efficaces ;

- la simplification de l'électronique associée grâce aux avancées dans le domaine des composants (circuits intégrés, transistors de puissance ...).

2.8.Moteur électrique à courant continu

Le moteur à courant continu est l'actionneur électrique le plus classique. Utilisé depuis longtemps, il est toujours présent dans de nombreuses applications. Beaucoup de structures différentes existent, mais le principe de base est le même.

2.8.1 Constitution d'un moteur à courant continu

On s'intéresse ici aux moteurs classiques, c'est-à-dire ceux qui sont conçus à peu près de la même façon que les machines de forte puissance.

Comme toute machine tournante, le moteur à courant continu comporte une partie fixe, le stator et une partie mobile, le rotor, séparées par un entrefer figure2.5 Le stator porte des aimants qui sont chargés de créer le champ magnétique dans l'entrefer, pour plus de détail sur le principe de fonctionnement des moteurs à courant continue reportez vous à **[12]**.

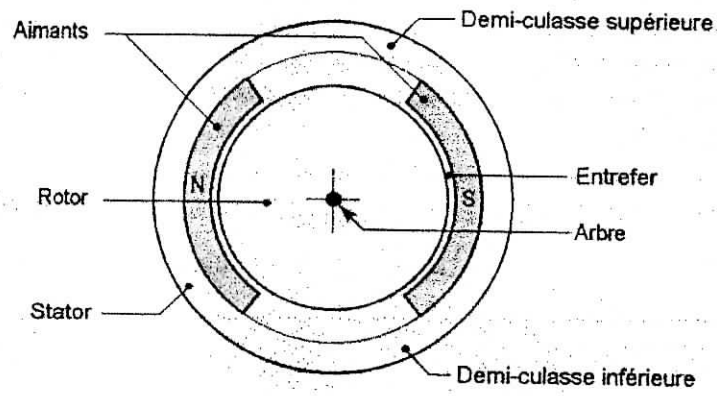


Figure 2.5. *Constitution d'un moteur à courant continue à aimant.*

2.8.2 Etude mécanique

Le phénomène essentiel dans un moteur à courant continue est la création d'un couple qui tend à faire tourner le rotor, nous avons l'expression du couple électromagnétique :

$$\Gamma_m = \frac{1}{2\pi}\frac{p}{a}N\Phi i \qquad (2.11)$$

Le moment s'exprime en newtons-métres (N·M). Φ est le flux sous un pôle, en weber(Wb), i l'intensité du courant dans l'induit exprimé en ampère(A), p est le nombre de paire de pôle, a le nombre de paires de voies de la machine et N, le nombre de conducteur actif. Pour un moteur donnée a, p et N sont fixes, on considère que le flux est pratiquement constant ; On pose donc :

$$Kt = \frac{1}{2\pi}\frac{p}{a}N\Phi \qquad (2.12)$$

Kt est la constante du moteur, avec cette définition, on a :

$$\Gamma_m = Kt\ i \qquad (2.13)$$

L'unité de Kt est le newton-mètre par empère ($N{\cdot}M{\cdot}A^{-1}$), la valeur de cette constante est généralement précisé dans la notice technique du moteur.
L'équation mécanique d'un moteur à courant continue découle du principe fondamental de la dynamique, pour un système en rotation qui est le cas du moteur nous avons :

$$\Gamma_m = J_T \frac{d\Omega}{dt} + B_T\Omega + \Gamma_s + \frac{\Gamma_p}{n} \qquad (2.14)$$

Où :
-J_T représente l'inertie totale vue par le moteur est égale à :

$$J_T = J_m + \frac{J_C}{n^2} \qquad (2.15)$$

J_m et J_C représenté respectivement l'inertie du rotor du moteur généralement considéré comme un cylindre et l'inertie de la charge externe.

- n est le rapport de réduction.
-B_T représente le coefficient de frottement visqueux totale vue par le moteur, il est donné par :

B_m et B_C représente respectivement le coefficient de frottement visqueux au niveau du rotor

$$B_T = B_m + \frac{B_C}{n^2} \qquad (2.16)$$

du moteur et le coefficient de frottement visqueux au niveau de la charge externe.
Le coefficient B_m a deus valeur selon que le moteur est commandé en tension ou en courant.

- Γ_s est le couple de frottement sec donné par :

$$\Gamma_S = \Gamma_0 \ \mathrm{Sign}(\Omega) \qquad (2.17)$$

la fonction sign étant le signe de la vitesse Ω et a pour valeur ± 1.

- Γ_P est le couple de perturbation que subit la charge externe.

2.8.3 Etude électrique

Sur le plan électrique , le moteur à courant continue à aiment est un dipôle, l'enroulement de l'induit a une résistance R et une inductance L. la loi des mailles donne :

$$u = E + R\,i + L\frac{d\,i}{d\,t} \qquad (2.18)$$

E est définie comme étant le force contre électromotrice, elle est donné par :

$$E = \frac{1}{2\pi}\frac{p}{a} N\Phi\,\Omega \qquad (2.19)$$

On peut mettre cette relation sou la forme :

$$E = K_E\,\Omega \qquad (2.20)$$

Avec $K_E = K_t$

On peut représenter le moteur par un schéma électrique équivalent (*figure2.6*).

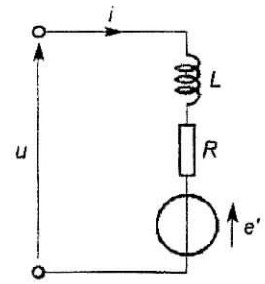


Figure 2.6. *Schéma équivalent .*

2.8.4 Fonction de transfert du moteur à courant continue

Afin d'étudier le comportement dynamique d'un moteur à courant continue, il est essentiel de déterminer sa fonction de transfert ou sa transmitance, elle définie le transfert entre la vitesse de rotation Ω et la tension de l'induit u, d'après les relations (2.13),(2.14),(2.17) et (2.19) obtenus précédemment nous avons le schéma suivant :

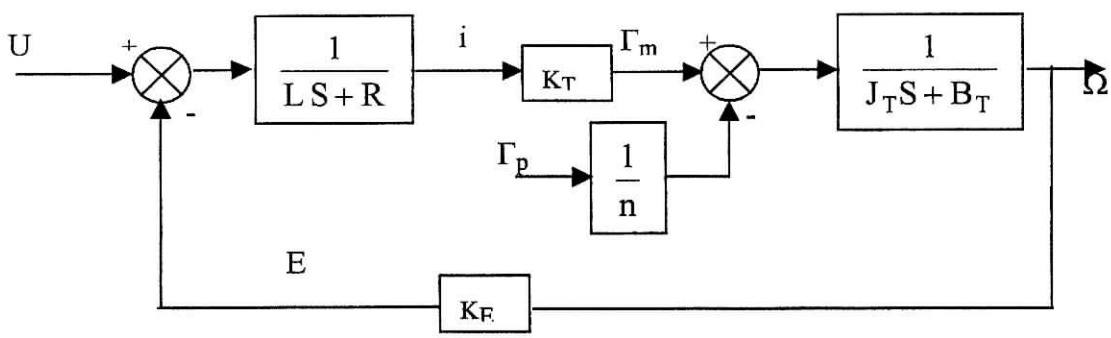


Figure 2.7. *Fonction de transfert d'un moteur à courant continu.*

On peut remarquer que le terme de Γ_s n'est pas représenté car il sera considéré comme une perturbation incluse dans le couple de perturbation externe. Les paramètres du moteur utilisé sont les suivants :

$L = 2.63$ mH
$R = 2.49\ \Omega$
$K_T = 45.8 \times 10^{-3}$ Nm/A
$n = 65.5$
$Jm = 7.06 \times 10^{-6}$ Kg m^2
$Bm = 3.54 \times 10^{-6}$ N m/rad/s
$K_E = 45.8 \times 10^{-3}$ V/ rad/s

2.9. Conclusion

Nous avons présenté dans cette partie le modèle représentant le moteur à courant continue, comme le moteur est l'actionneur utilisé ,il sera tenu compte pour le dimensionnement des régulateur.

CHAPITRE III :

POURSUITE DE REFERENCE ET PRISE EN COMPTE DE LA DYNAMIQUE DES ACTIONNEURS

3.1 Introduction

Avant toute implémentation d'une commande pour le contrôle des robot manipulateur ou tout autre système, il est nécessaire d'effectuer des simulations afin d'observer la manière dont va réagir le bras- manipulateur avec cette commande ; Dans cette partie nous allons présenter une séries de simulation sur le robot Scara, les commandes utilisées sont les celle décrites au annexe2, en premier nous appliquerons la commande par découplage non linéaire dont nous montrerons l'influence de la charge sur les performances puis les deux commandes PD et PID décentralisées ; ces simulations ont été réalisées en considérons la dynamique des actionneurs comme étant négligeable, mais en réalité ces actionneurs qui sont des moteurs dans notre cas ont une dynamique dont il faut en tenir compte, pour cette raisons, nous allons étudier la dynamique des moteurs PITTMAN puis effectuer une simulation du robot avec la commande PD seulement en incluant les équation représentant la dynamique des moteurs.

Les simulations ont été effectuées sous environnement Matlab en utilisant la fonction ode45 qui est une implémentation de la méthode d'intégration numérique de *Runge-Kutta* ;nous donnerons la représentation d'état du modèle du robot Scara avec sa commande.

3.2. Commande par découplage non linéaire (Couple Calculé)

Nous rappelons que la commande par découplage non linéaire nous permet par un retour d'état d'obtenir un système linéarisé ; nous avons la représentation d'état du robot Scara qui est la suivant **[17]**:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = [A(x_1) + m_0 J^T J]^{-1}(-N(x_1, x_2) - \Gamma_f - m_0 J^T \dot{J} x_2 + \Gamma) \end{cases} \tag{3.1}$$

avec $x_1 = q, x_2 = \dot{q}, J = J(x_1)$ et $\dot{J} = \dot{J}(x_1, x_2)$.

nous pouvant voir que le modèle contient le couple de charge qui est dû à la masse m_0, comme cette masse est variable, la commande qui permet la linéarisation de notre système ne tient pas compte de cette masse, nous verrons ainsi son influence , la commande qui permet la linéarisation donné par la relation (A2.13) est **[1]** :

$$\Gamma = A(x_1)\, w(t) + N(x_1, x_2) + \Gamma_f \tag{3.2}$$

le système ainsi linéarisé, nous obtenons avec un régulateur PD la dynamique de l'erreur qui est la suivant :

$$\ddot{e} + K_v \dot{e} + K_p e = 0 \tag{3.3}$$

où $e = x_1^d - x_1$.

Les gains de la commande sont choisis de façon à avoir en boucle fermée un amortissement critique, la bande passante du premier axe est fixée w_n= 5rad/s et w_n= 25 rad/s pour la deuxième axe, nous obtenons les matrices K_p et K_v suivante :

$$K_p = \begin{bmatrix} 25 & 0 \\ 0 & 625 \end{bmatrix} \quad , K_v = \begin{bmatrix} 10 & 0 \\ 0 & 50 \end{bmatrix}$$

nous avons effectué les simulations pour des faibles et grand déplacement, avec et sans charge, nous obtenons se qui suit :

3.2.1 Simulations

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

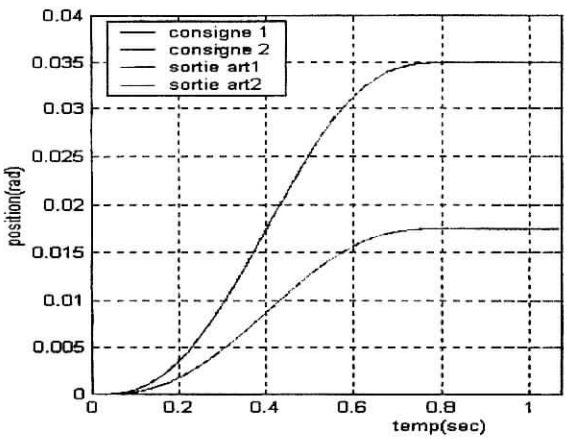


Figure 3.1.1.a *Suivi en position.*

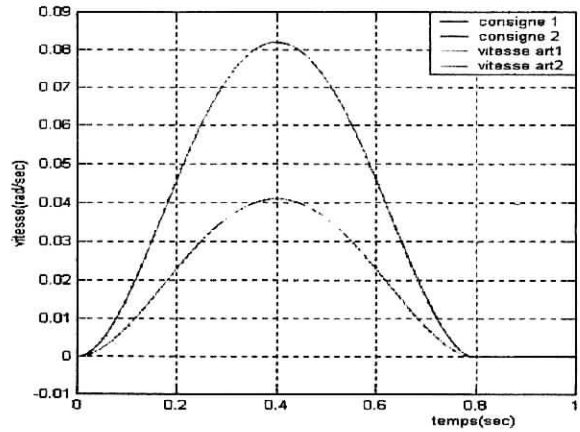


Figure 3.1.1.b *Suivi en vitesse.*

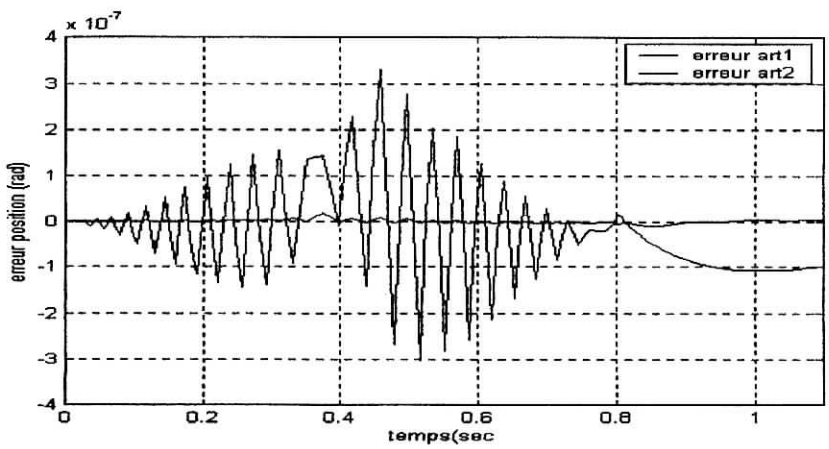


Figure 3.1.1.c *Erreur en position.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4\,kg$, nous obtenons les résultats suivant :

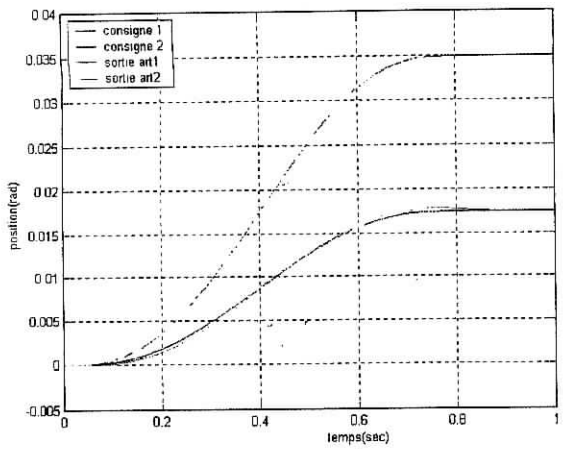


Figure 3.1.2.a *Suivit en position.*

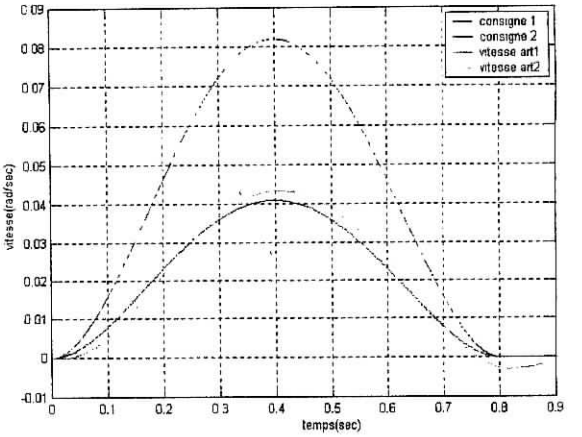


Figure 3.1.2.b *Suivi en vitesse.*

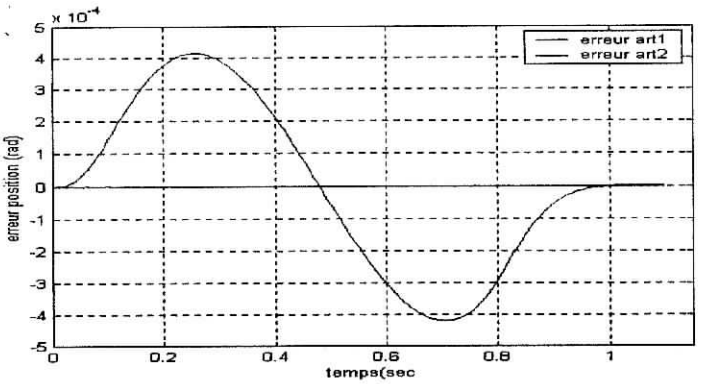


Figure 3.1.2.c *Erreur en position.*

Pour $q_i = [0 \quad 0]^T$, $q_f = [pi/3 \quad pi/4]^T$ et $m_0 = 0\,kg$,nous obtenons les résultats suivant :

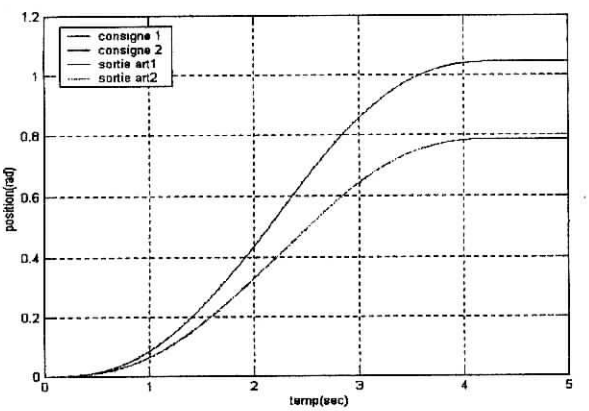


Figure 3.2.1.a *Suivi en position.*

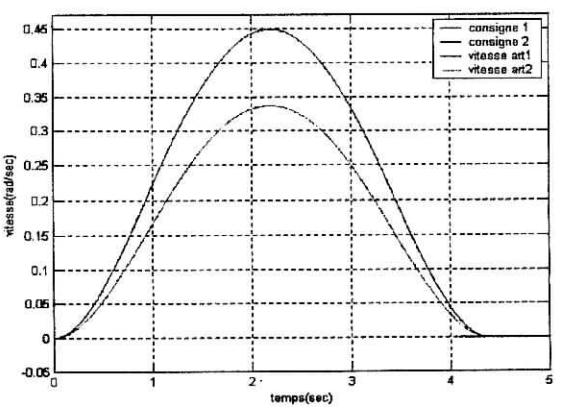


Figure 3.2.1.b *Suivi en vitesse.*

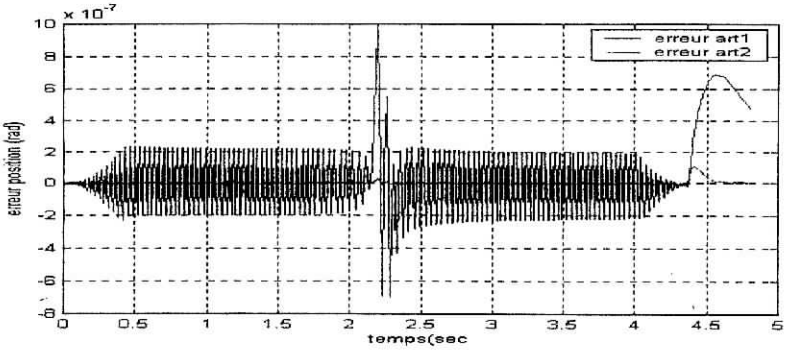


Figure 3.2.1.c *Erreur en position.*

Pour $q_i = [0 \quad 0]^T$, $q_f = [pi/3 \quad pi/4]^T$ et $m_0 = 4\,kg$,nous obtenons les résultats suivant :

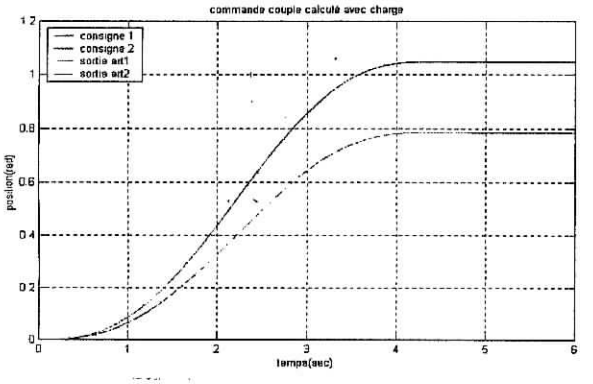


Figure 3.2.2.a *Suivi en position.*

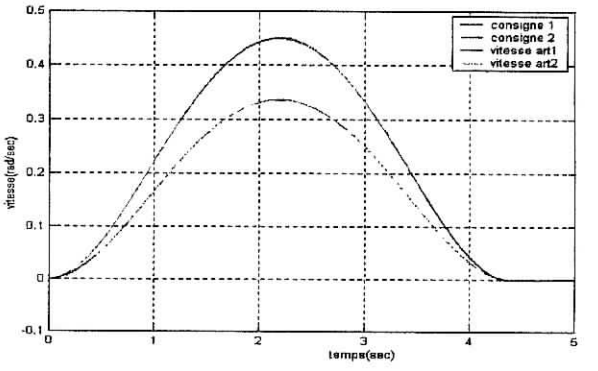


Figure 3.2.2.b *Suivi en vitesse.*

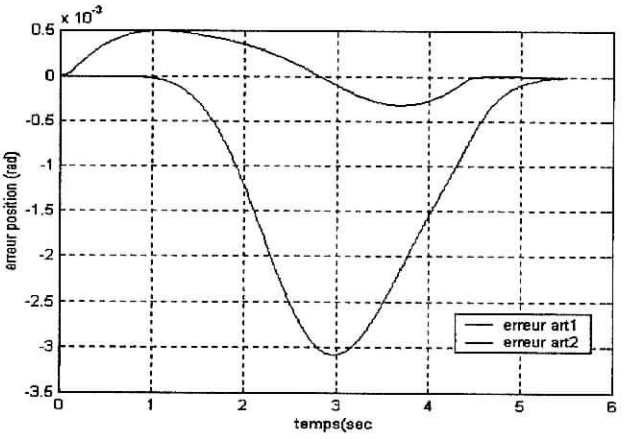


Figure 3.2.2.c *Erreur en position.*

Remarque

Nous pouvons voir que la commande par découplage non linéaire donne un bon suivit de la trajectoire dans le cas sans charge, mais cette performance se détériore dans le cas ou il y a présence de charge et surtout pour les faible déplacement figures 3.1.2, cela s'explique par le faite que la commande qui permet la linéarisation n'inclus pas la charge car elle est variable, pour palier ce problème il est nécessaire de déterminer le modèle dynamique réel, pour se faire le modèle est calculé en ligne, à partir de l'algorithme de Newton- Euler, on détermine ainsi les valeurs numérique des paramètres inertiels.

3.3 Commande Proportionnelle Dérivée

Pour un robot de type Scara une commande proportionnelle dérivée suffit pour effectuer la poursuite de trajectoire **[1]**, la commande est donnée par (A2.8) ; Dans se qui suit nous allons présenter les simulations utilisant se type de commande, nous ferons varier les pôles en boucle fermée pour voir l'influence que ça a sur notre robot et cela en absence et en présence de charge.

Nous prendrons les pôles en boucles fermée ω=20 et ω=50 et cela pour les deux articulation ; La méthode utilisée pour les calcul des coefficients des régulateur est celle citée au annexe2.

La valeurs des inerties maximales vue par chaque articulation est :

$$a_1 = \frac{(m_1 + m_2) L_1^2 + m_2 L_2^2 + 2 m_2 L_2 L_2}{n^2} \qquad a_2 = \frac{m_2 L_2^2}{n^2} \tag{3.4}$$

n est définie comme le rapport de réduction du moteur égale à 65,5 ; Après calcule nous obtenons :

a_1=0.0027 kg.m^2

a_2=4.9466x10^{-4} kg.m^2

Précédemment nous avons pris comme charge m_0 = 4 kg, ce choix n'est pas quelconque, nous avons fait en sorte que l'inertie additionnelle dû à l'effet de la charge corresponde à 25% de l'inertie maximale a_1 de la première articulation, cela représente environs 33% de l'inertie maximale a_2 de la deuxième articulation.

a_{1m} = (0.0027 x1.25) =0.0034 kg.m^2

a_{2m} = (4.9466 x10^{-4} x1.33) = 6.6165 x10^{-4} kg.m^2

D'après les relation (4.10) et (4.11), nous obtenons les matrices K_p et K_v suivante :

Pour ω=20 nous avons :

$$K_p = \begin{bmatrix} 1.36 & 0 \\ 0 & 0.2647 \end{bmatrix} \quad , K_v = \begin{bmatrix} 0.136 & 0 \\ 0 & 0.02647 \end{bmatrix}$$

Pour ω=50 nous avons :

$$K_p = \begin{bmatrix} 8.5 & 0 \\ 0 & 1.6541 \end{bmatrix} \quad , K_v = \begin{bmatrix} 0.34 & 0 \\ 0 & 0.0662 \end{bmatrix}$$

ainsi la commande est donné par :

$$\Gamma = n^2 (K_p e + K_v \dot{e}) \qquad (3.5)$$

où $e = x_1^d - x_1$.

Nous avons effectué les simulations pour des faibles et grand déplacement, avec et sans charge, nous commençons par prendre $\omega = 20$, nous obtenons se qui suit :

3.3.1 Simulations

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

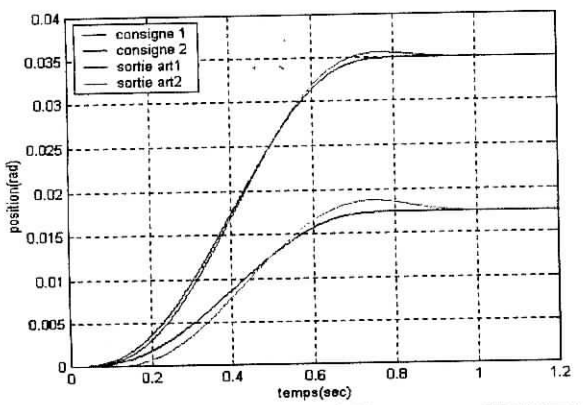


Figure 3.3.1.a *Suivi en position.*

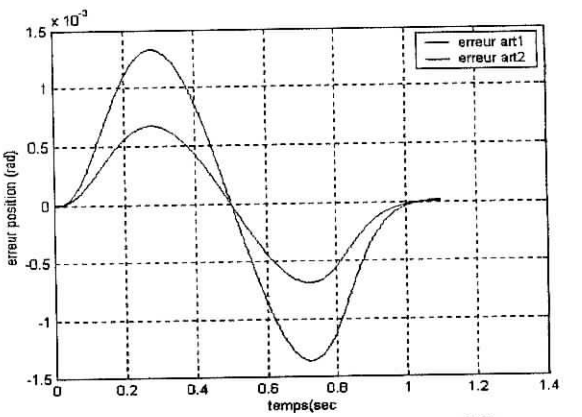


Figure 3.3.1.b *Erreur en position.*

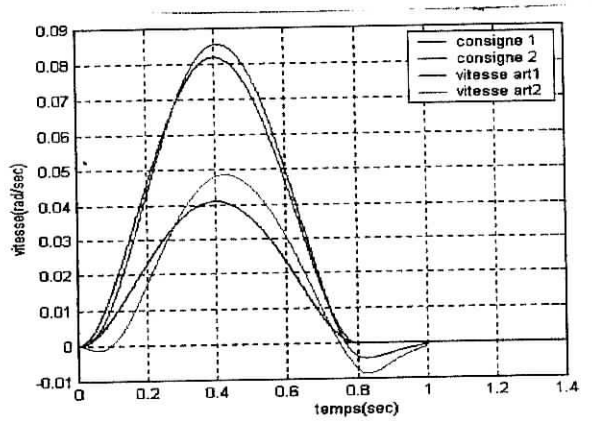


Figure 3.3.1.c *Suivi en vitesse.*

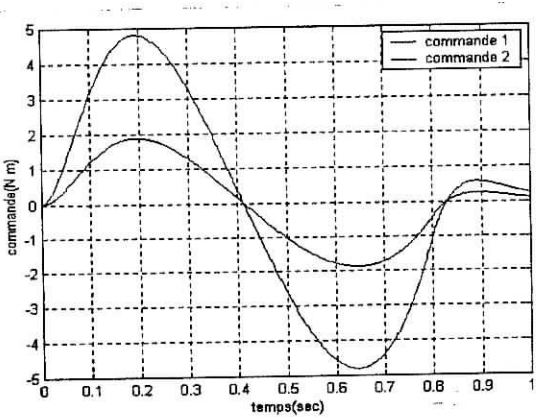


Figure 3.3.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4\,kg$,nous obtenons les résultats suivant :

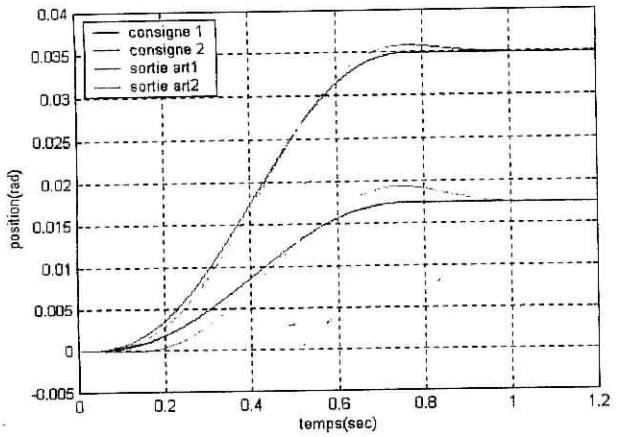


Figure 3.3.2.a *Suivi en position.*

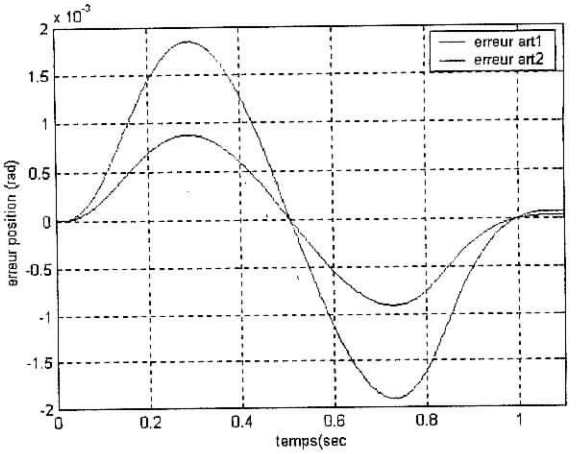


Figure 3.3.2.b *Erreur en position.*

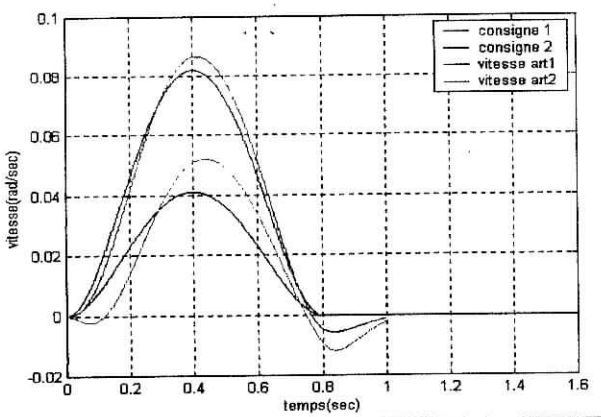


Figure 3.3.2.c *Suivi èn vitesse.*

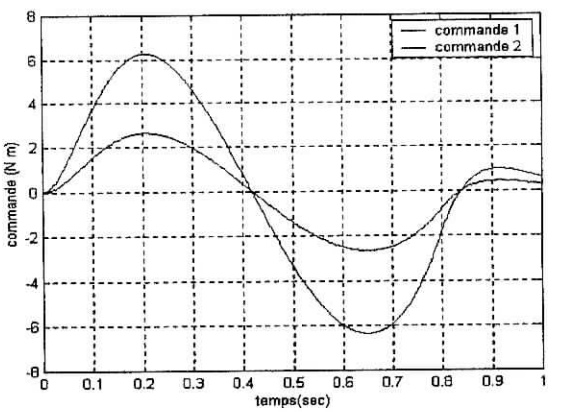


Figure 3.3.2.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/3 \quad pi/4]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

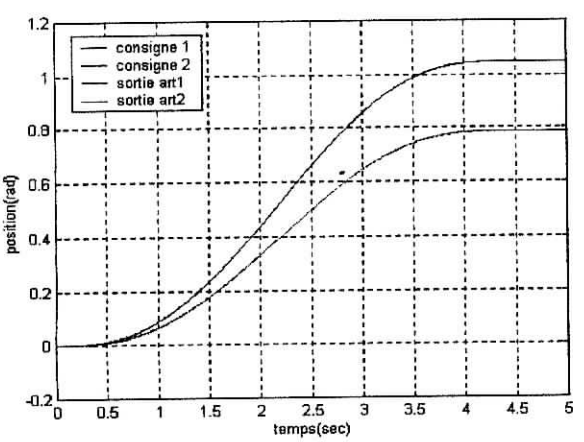


Figure 3.4.1.a *Suivi en position.*

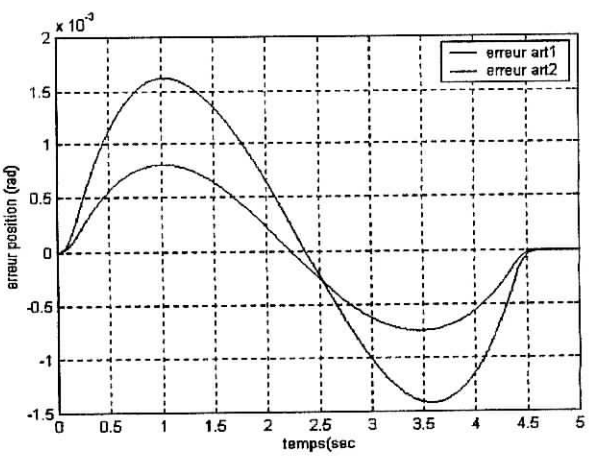


Figure 3.4.1.b *Erreur en position.*

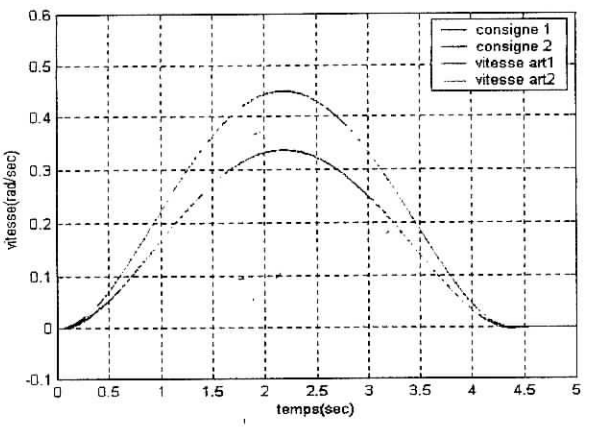


Figure 3.4.1.c *Suivi en vitesse.*

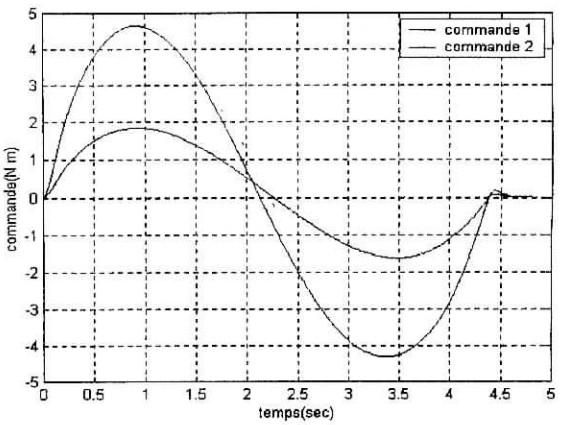


Figure 3.4.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/3 \quad pi/4]^T$ et $m_0 = 4\text{kg}$,nous obtenons les résultats suivant

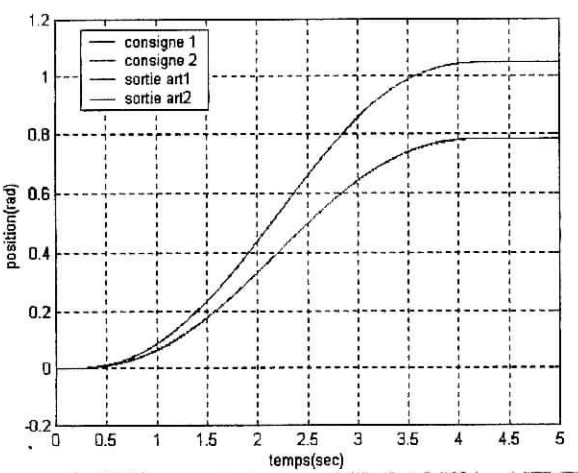


Figure 3.4.2.a *Suivi en position.*

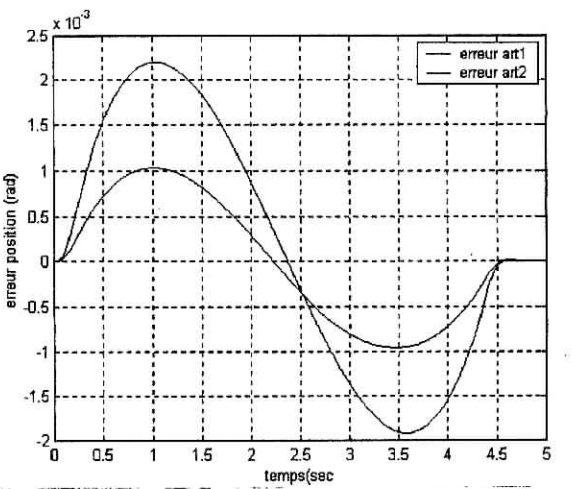


Figure 3.4.2.b *Erreur en position.*

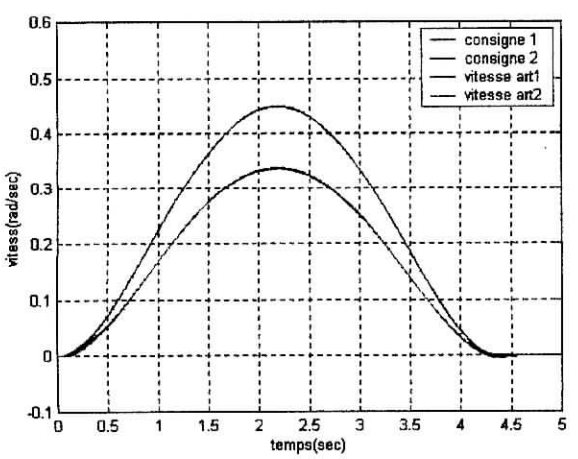


Figure 3.4.2.c *Suivi en vitesse.*

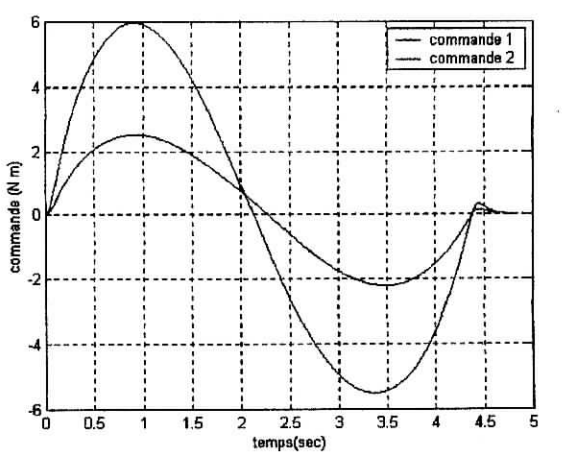


Figure 3.4.2.d *Signal de commande.*

Remarques

Nous pouvons constater des figures 3.1 que nous avons un mauvais suivit de la trajectoire pour des faibles déplacements et surtout en présence de charge mais par contre un bon suivit pour des grand déplacement , pour cette raison nous introduisons un pôle plus rapide qui a pour valeur 50 et nous simuleront pour des faible déplacement, ainsi nous nous obtenons les résultats suivant :

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

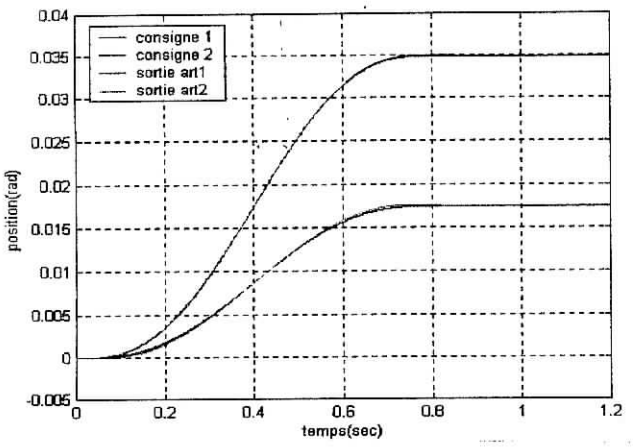


Figure 3.5.1.a *Suivi en position.*

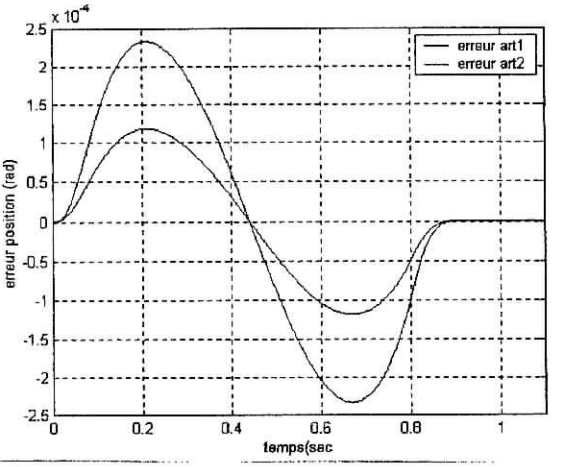


Figure 3.5.1.b *Erreur en position.*

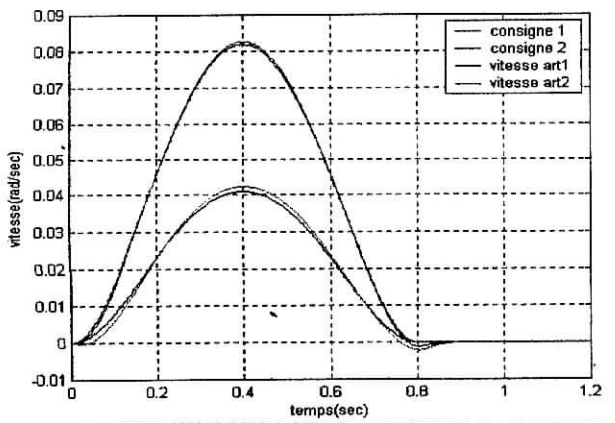


Figure 3.5.1.c *Suivi en vitesse.*

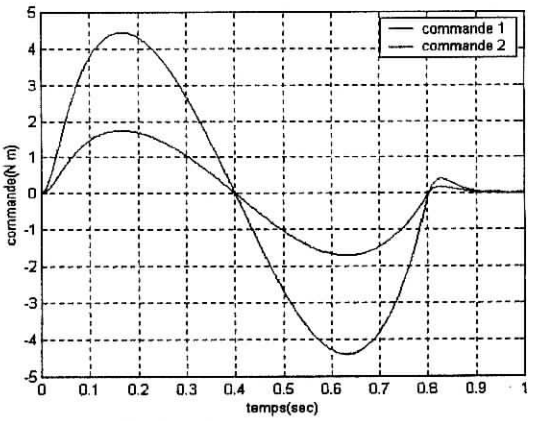


Figure 3.5.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4kg$,nous obtenons les résultats suivant :

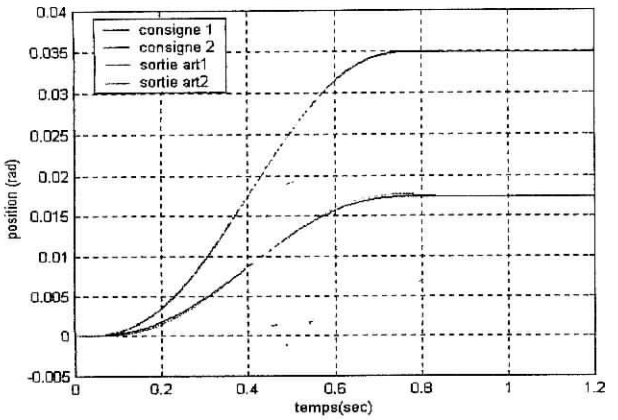


Figure 3.5.2.a *Suivi en position.*

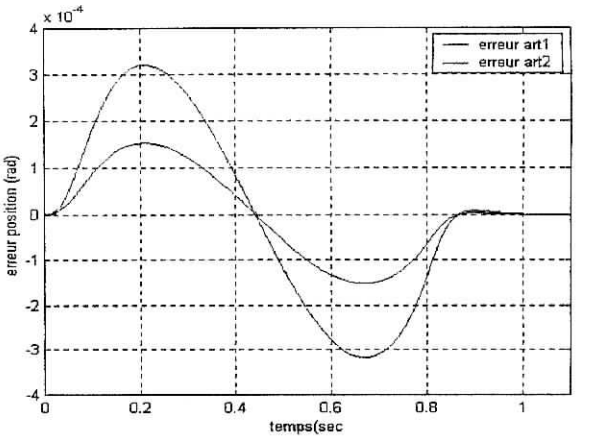


Figure 3.5.2.b *Erreur en position.*

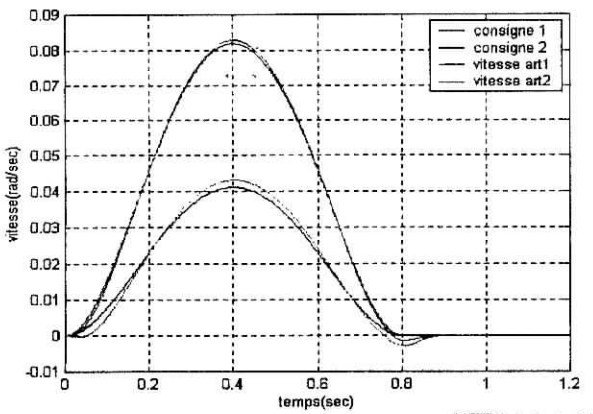


Figure 3.5.2.c *Suivi en vitesse.*

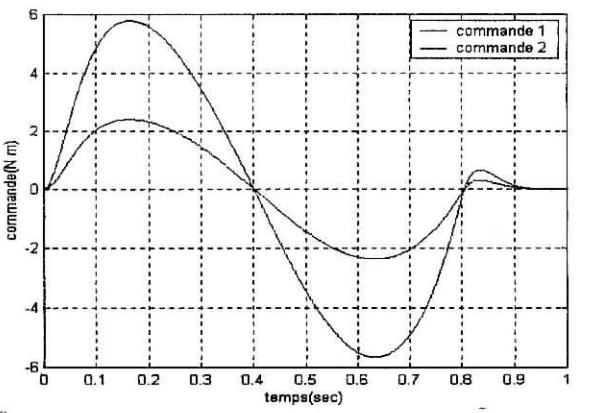


Figure 3.5.2.d *Signal de commande.*

Nous voyons des figures 3.5.1 et 3.5.2 une amélioration du suivit de trajectoire et cela même en présence de charge ; Le régulateur PD a donnée des résultats acceptable, mais nous allons essayer d'améliorer le performances en introduisant un action intégrale.

3.4 Commande Proportionnelle intégrale et Dérivée

Précédemment nous avons constaté que nous avions un bon suivit de trajectoire dans cette section nous allons utilisée une commande de type PID et voir l'amélioration qui peut en résulter **[1]**, le calcul des matrice K_p et K_v et K_I se fait a partir des relations (A2.6) et (A2.7), les pôles sont choisit comme pour la commande PD, nous obtenons les matrices suivantes :

Pour ω=20 nous avons :

$$K_p = \begin{bmatrix} 4.08 & 0 \\ 0 & 0.794 \end{bmatrix} \quad , K_v = \begin{bmatrix} 0.204 & 0 \\ 0 & 0.02397 \end{bmatrix} \quad K_I = \begin{bmatrix} 27.2 & 0 \\ 0 & 5.2932 \end{bmatrix}$$

Pour ω=50 nous avons :

$$K_p = \begin{bmatrix} 25 & 0 \\ 0 & 4.962 \end{bmatrix} \quad , K_v = \begin{bmatrix} 0.51 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad K_I = \begin{bmatrix} 425 & 0 \\ 0 & 82.706 \end{bmatrix}$$

3.4.1 Simulations

nous commencerons par prendre un pôle ω=20 ; Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

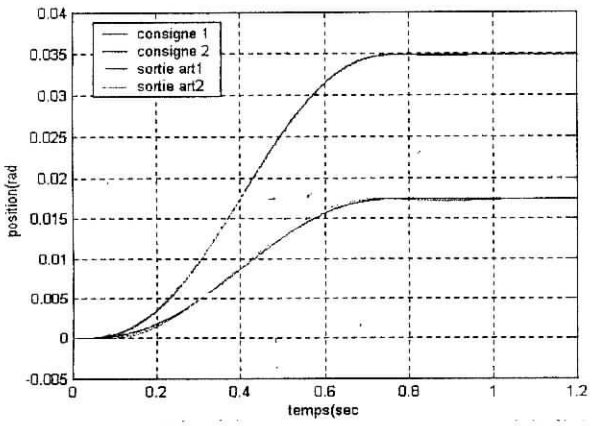


Figure 3.6.1.a *Suivi en position.*

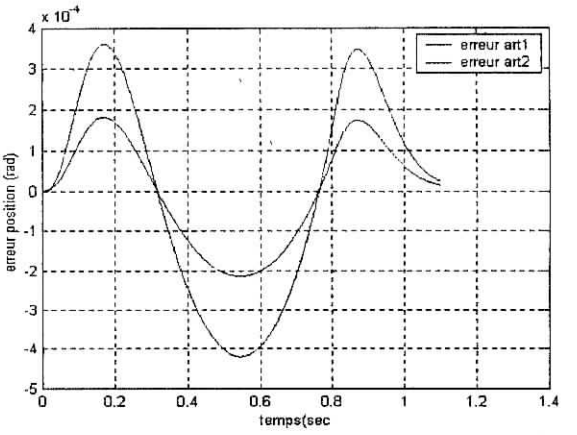


Figure 3.6.1.b *Erreur en position.*

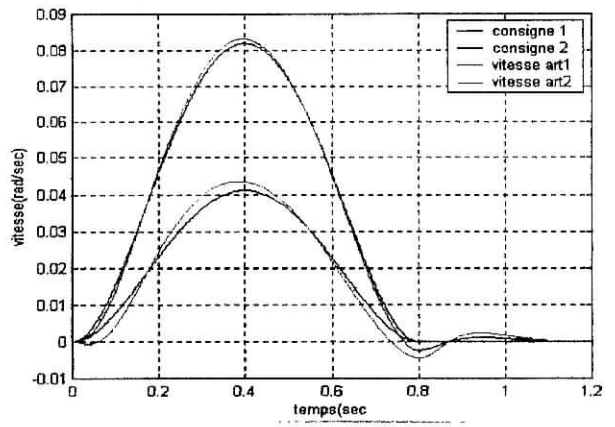


Figure 3.6.1.c *Suivi en vitesse.*

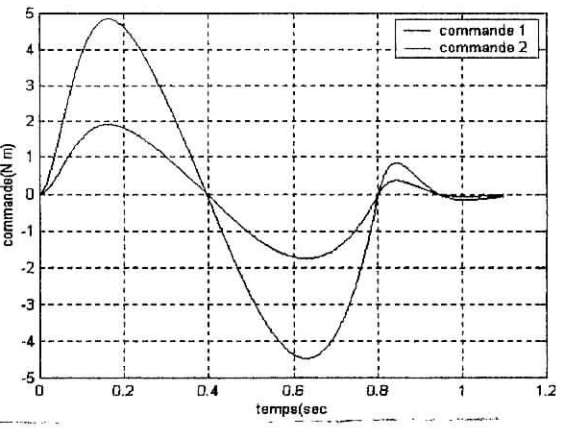


Figure 3.6.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4\,kg$,nous obtenons les résultats suivant :

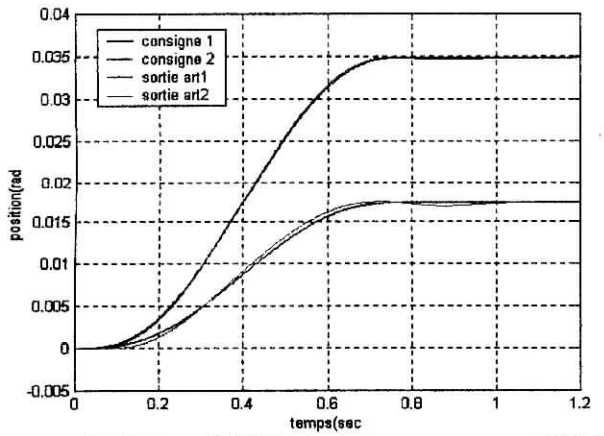


Figure 3.6.2.a *Suivi en position.*

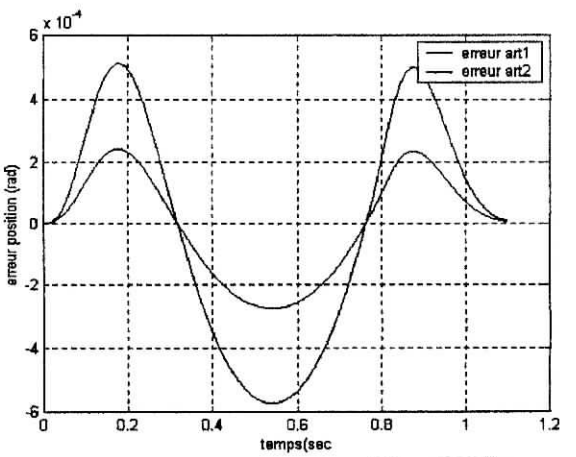


Figure 3.6.2.b *Erreur en position.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4\,kg$,nous obtenons les résultats suivant :

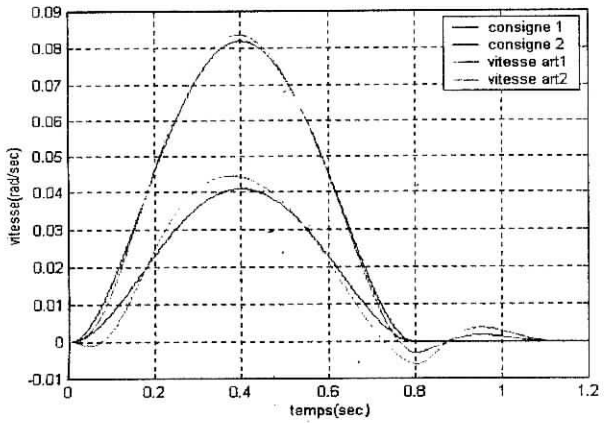


Figure 3.6.2.c *Suivi en vitesse.*

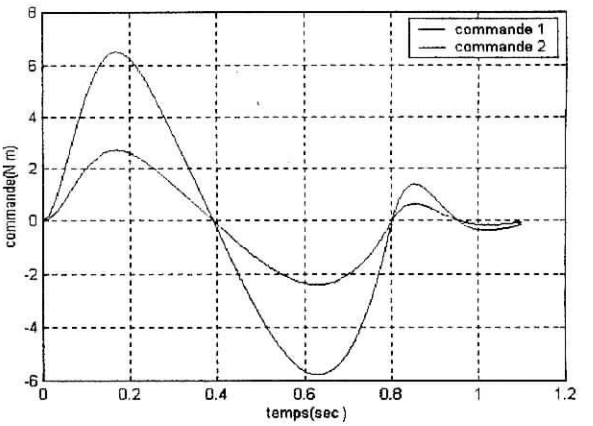


Figure 3.6.2.d *Signal de commande.*

Le pôle ω=50 ; Les données $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$,nous obtenons les résultats suivant :

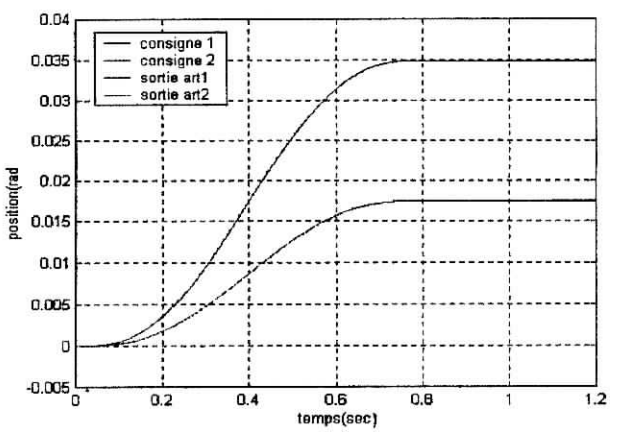


Figure 3.7.1.a *Suivi en position.*

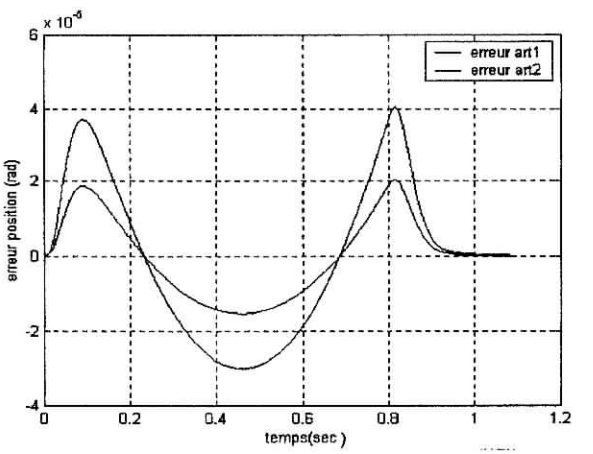


Figure 3.7.1.b *Erreur en position.*

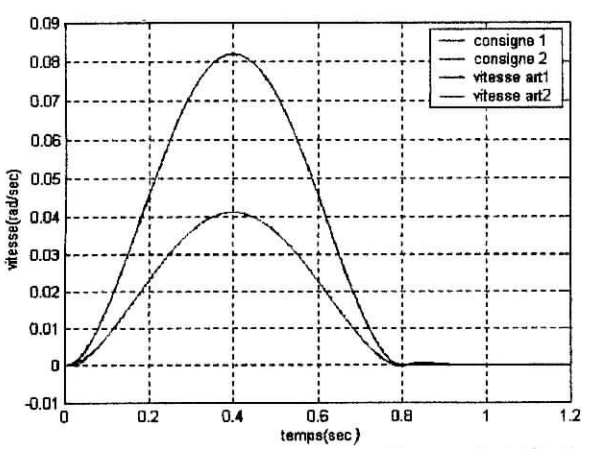


Figure 3.7.1.c *Suivi en vitesse.*

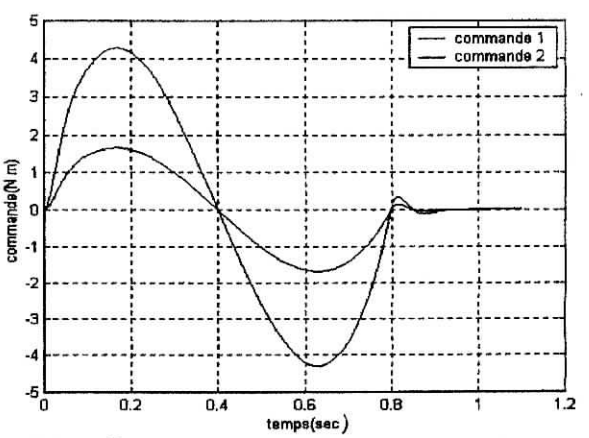


Figure 3.7.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4\,kg$,nous obtenons les résultats suivant :

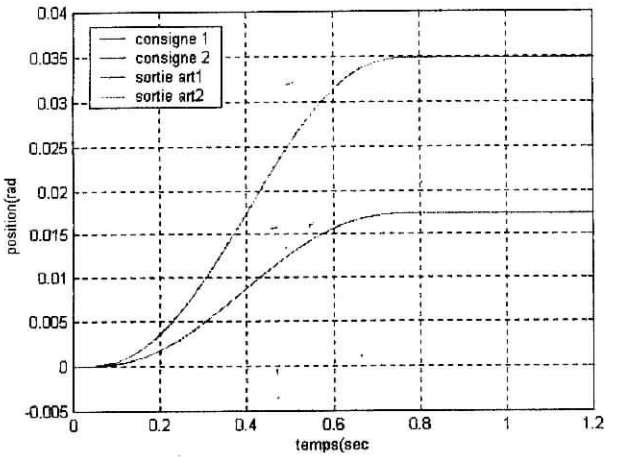


Figure 3.7.2.a *Suivi en position.*

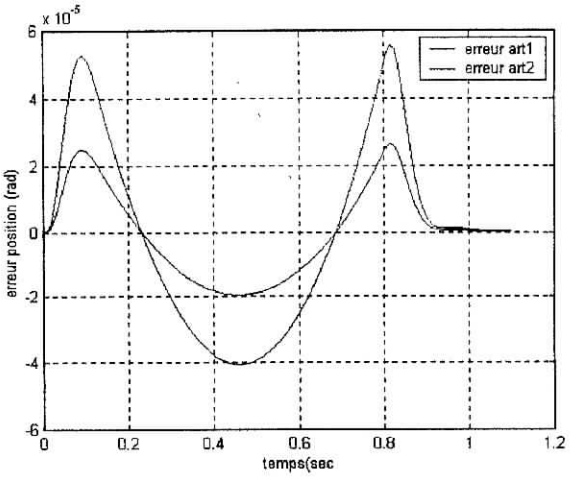


Figure 3.7.2.b *Erreur en position.*

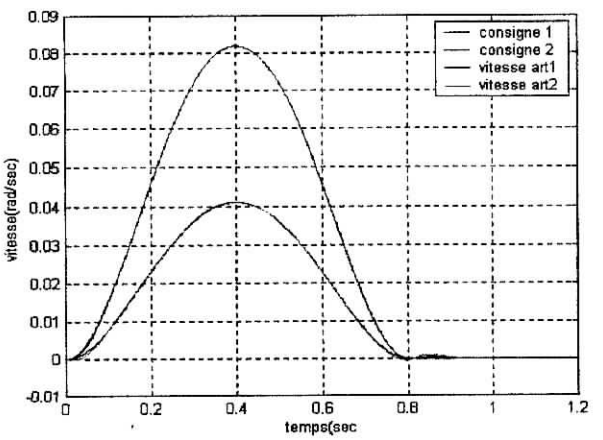


Figure 3.7.2.c *Suivi en vitesse.*

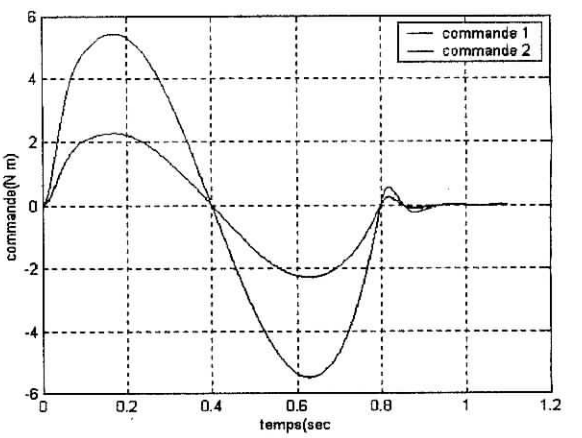


Figure 3.7.2.d *Signal de commande.*

Remarques :

Les figures 3.7.1 et 3.7.2 montre que le régulateur PID donne un très bon suivit de trajectoire avec un pôle qui est assez rapide, et cela même pour des très faible déplacement et en présence de charge.

3.4.2 Conclusion

La commande décentralisée du type PD et PID appliquée sur un robot de type Scara a donnée de bons résultat pour le suivit de trajectoire et est resté robuste vis à vis des variations d'inertie dû aux termes couplages, nous pouvons constater que pour des faibles et grands déplacements, la commande à la même grandeur, cela est dû par le faite que pour une trajectoire donnée, le temps de parcours est optimal, donc nous pouvons conclure des simulations effectuées précédemment qu'une commande de type PD ou PID est une commande satisfaisante. et permet une implémentation moins coûteuse assai simple pour suivre une trajectoire de consigne.

3.5. Influences des actionneurs

Précédemment nous avons effectué les différentes simulations en ne tenant pas compte des effets ou plus exactement de la dynamique des actionneurs ; Les moteurs étant des système électromécanique leurs dynamique ne doit pas être négligés **[15]**, pour cette raison, il est nécessaire de tenir compte dans la représentation d'état du modèle de notre robot ; Dans cette partie nous allons étudier le moteur PITTMAN qui servira comme actionneur et le commander afin qu'il fournissent le couple nécessaire pour suivre la consigne de trajectoire.

A partir de la figure 2.7 (§2.8.4) nous obtenons les deux fonctions de transfert qui représente respectivement le transfert entre la vitesse de rotation Ω et la tension au borne de l'induit **U**, et le transfert entre le courant **i** dans l'induit et la tension **U** , elles sont :

$$F_{\Omega}(S) = \frac{K_T}{(J_T S + B_T)\cdot(L S + R) + K_T K_E} \tag{3.6}$$

et

$$F_i(S) = \frac{(J_T S + B_T)}{(J_T S + B_T)\cdot(L S + R) + K_T K_E} \tag{3.7}$$

En utilisant les valeur des différents paramètres du moteur, nous traçons la réponse à un échelon de tension des deux fonctions de transfert, nous obtenons les graphes suivant :

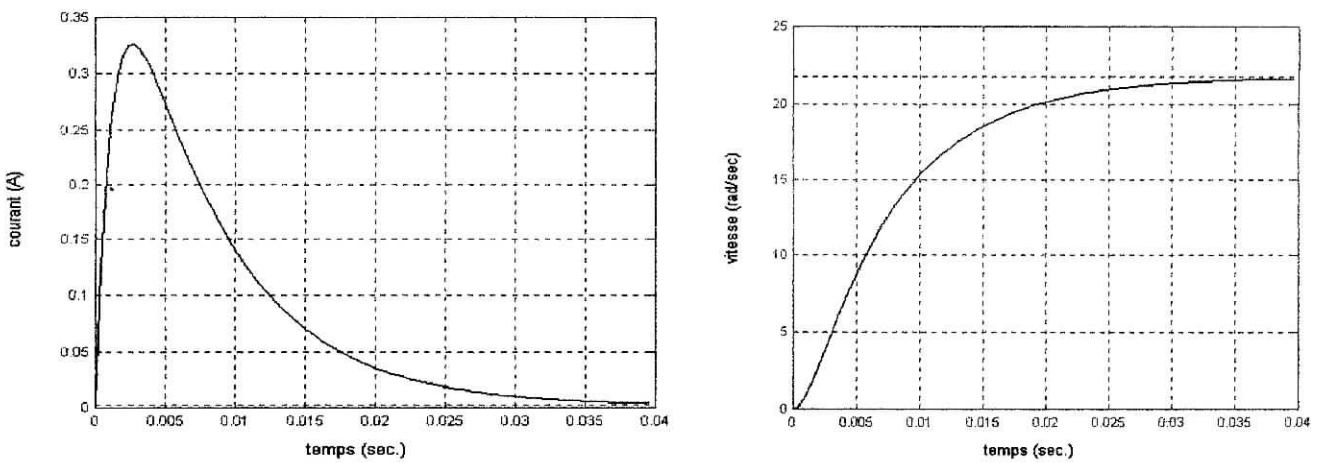


Figure 3.8.1 *Réponse indicielle du courant et de vitesse .*

Commentaire :

Le moteur est un système stable du deuxième ordre car il possède une constante de temps électrique et une mécanique, à partir du graphe de courant nous constatons que le courant passe par un pic, cela s'explique par le faite que qu'au démarrage le la fem est nulle et le courant évolue avec la tension U d'alimentation au borne de l'induit, cette évolution fait tourner le rotor qui lui génère un fem qui réduit la tension au borne de cette induit, d'ou une décroissance du courant, en terme de transfert cela s'explique par la présence d'un zéro dans

$F_i(S)$; En robotique le couple appliqué sur chaque articulation doit être celui issu de la commande, en générale la fem peut être négligé mais dans notre cas il ne serait pas correcte de le faire, pour cette raison il est nécessaire de réguler le courant à l'intérieur du moteur donc un commande en cascade s'impose.

3.6. Commande en cascade

La commande en cascade est une commande très répondue, il offre plusieurs avantages comme la décomposition d'un système complexe en sous système et la limitation de grandeur interne **[3]**. La figure 3.8.2 donne le principe de réglage en cascade pour la commande en position d'un moteur à courant continue.

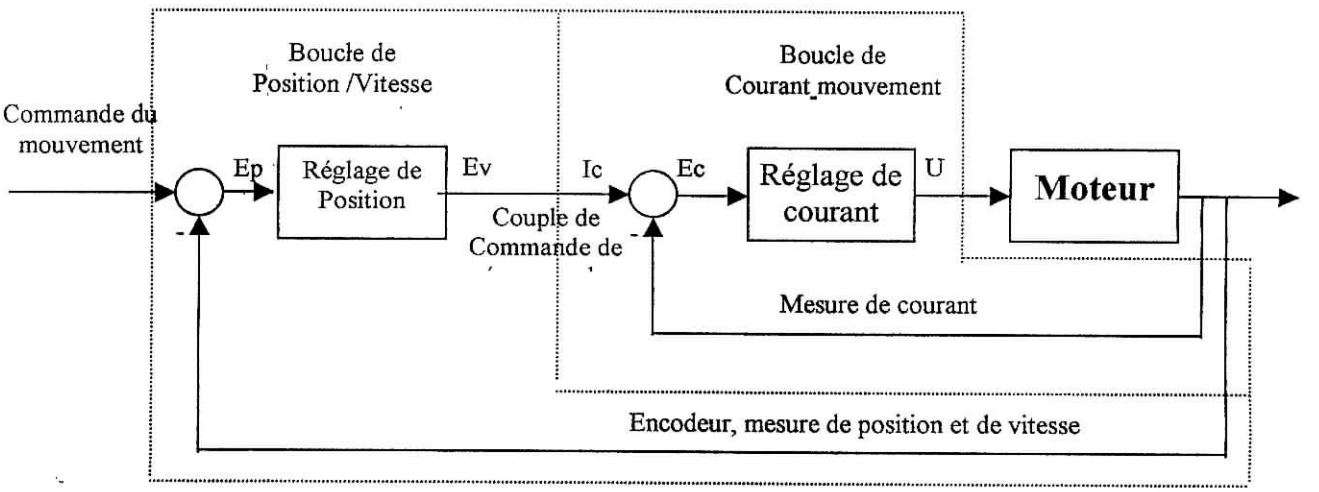


Figure 3.8.2 : *schéma fonctionnelle Boucle d'asservissement en cascade*

3.7. Réglage du courant

. Le couple appliquer sur chaque articulation doit correspondre au couple du régulateur, pour cette raison nous introduirons un régulateur interne de type Proportionnel Intégral, de régulateur s'écrit :

$$C_i(S) = K_{pi}\frac{(1+T_iS)}{T_iS} \qquad (3.8)$$

Une méthode simple pour le calcul de ce régulateur qui consiste à prendre la partie intégrale T_i égale à la constante de temps électrique et choisir le gain K_{pi} pour avoir une réponse en courant sans dépassement **[13]**. Après un petit calcul nous obtenons :

$$T_i = 1.06\times10^{-3} \quad \text{et} \quad K_{pi} = 1$$

nous faisons une simulation de la fonction $F_i(S)$ avec le régulateur calculé précédemment.

nous traçons ainsi la réponse à un échelon unitaire en boucle fermé en ayant une charge sur le rotor une inertie qui a pour valeur a_{1m} calculé précédemment, on obtient :

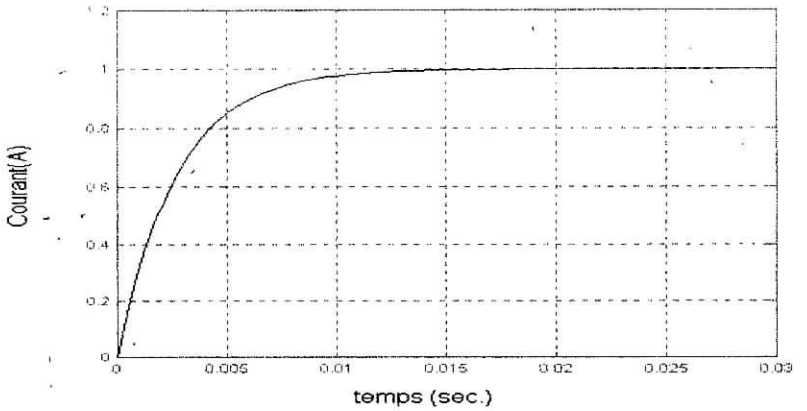


Figure 3.8.3 *Réponse indicielle en boucle fermée du courant.*

Remarques :

Nous observons que la réponse de la boucle de courant est réponse sans dépassement ;Nous avons constaté que l'erreur statique ne s'annule qu'à l'infinie, cela s'explique par la nature du système à régler, lequel présente un comportement dérivateur compensant le terme intégrateur, mais cette erreur reste acceptable qui est de l'ordre de 0.1%.

Le courant étant régulé, nous pouvant dire que la couple qui sera appliqué sur l'articulation sera le couple (courant) de consigne provenons du régulateur de la boucle externe, nous allons effectuer une simulation du moteur avec les deux boucles de régulation, en prenant ici ω=40, une charge correspondante à a_{1m} et un régulateur PD, le schéma de commande est le suivant :

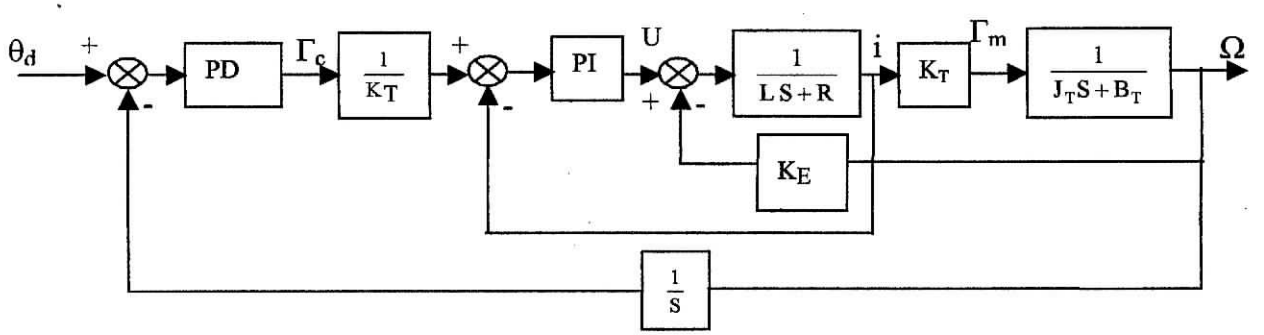


Figure 3.8.4 *Schéma de commande en cascade d'un moteur* .

nous obtenons à partir des relations (3.10) et (3.11) les coefficients de régulateur suivants :

$Kp = 5.44 \qquad Kv = 0.272$

nous effectuons la simulation sous Simulink avec les données suivantes :
q_i=0 et q_f=5 rad ; La valeur de k_a est choisit égale à 15 afin que le courant dans l'induit du moteur ne dépasse pas 1.5A.

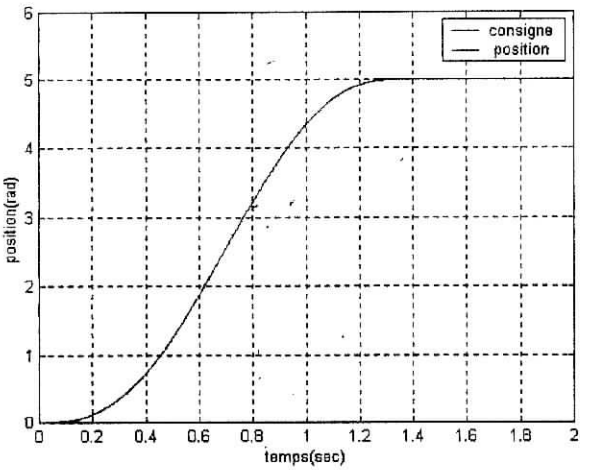


Figure 3.9.1.a *Suivi en position.*

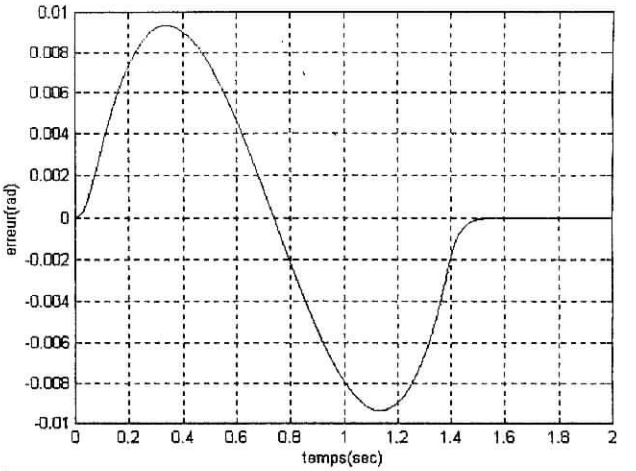


Figure 3.9.1.b *Erreur en position.*

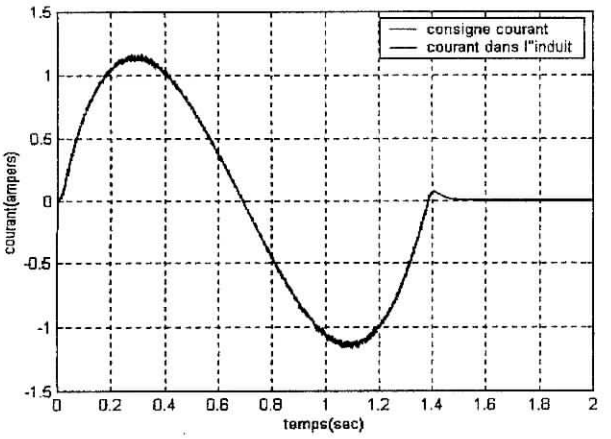


Figure 3.9.1.c *Suivi en courant.*

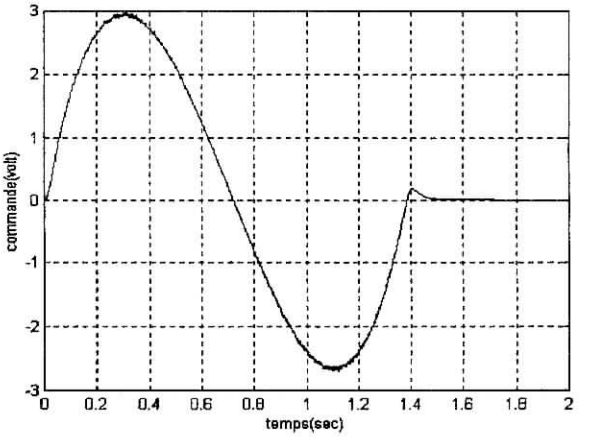


Figure 3.9.1.d *Signal de commande.*

Les figures 3.9.1 montre que le réglage en cascade donne de très bon résultats, en plus du suivit de trajectoire on peut voir que le courant dans l'induit du moteur suis biens le courant de référence donné par le régulateur PD, cette simulation concerne la commande en position d'un moteur ayant une charge constante, donc il est nécessaire de faire de nouvelles simulation du modèle du robot Scara en incluant cette fois ci la dynamique du moteur ; Nous rappelons la dynamique électrique d'un moteur à courant continue est donnée par la relations :

$$U = K_E \Omega + R i + L \frac{d i}{d t} \tag{3.9}$$

On sait que le couple réel est donné par :

$$\Gamma_r = K_T \, i \tag{3.10}$$

d'où nous obtenons la dynamique du couple qui est la suivante :

$$\dot{\Gamma}_r = -\frac{R}{L}\Gamma_r - \frac{K_E K_T}{L}\Omega + K_T \, U \tag{3.11}$$

la tension de commande U provient du régulateur PI , d'où nous avons l'expression suivante :

$$U = \frac{1}{K_T}\left[K_{Pi}(\Gamma_c - \Gamma_r) + K_i \int_0^t (\Gamma_c - \Gamma_r)\, d\tau\right] \qquad (3.12)$$

avec :

$$K_{pi}=1 \text{ et } K_i = \frac{1}{T_i} = 943.4$$

Nous rappelons que :

$$\Gamma_c = n^2 (K_p e + K_v \dot{e}) \qquad (3.13)$$

Après avoir établie les équation relatives à la dynamique électrique du moteur, nous obtenons les résultats de simulation effectuées pour des faibles déplacement ;

nous prenons ω=50 ; Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 0$

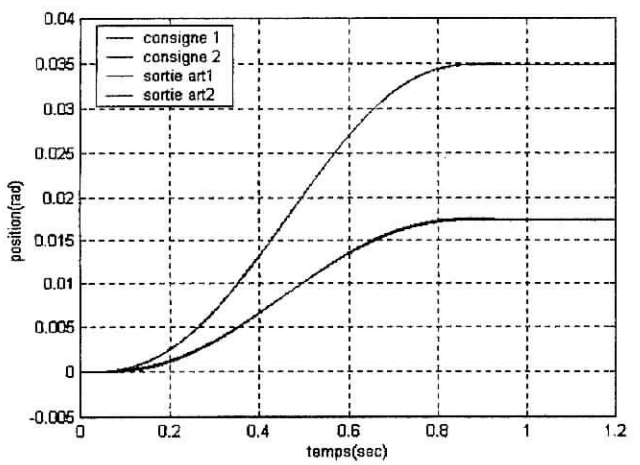


Figure 3.10.1.a *Suivi en position.*

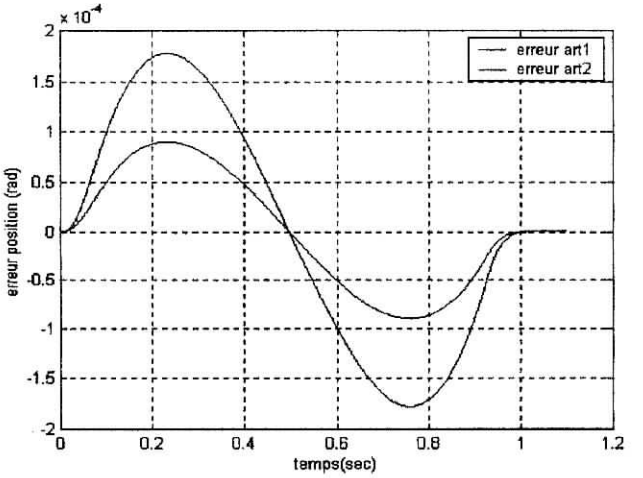


Figure 3.10.1.b *Erreur en position.*

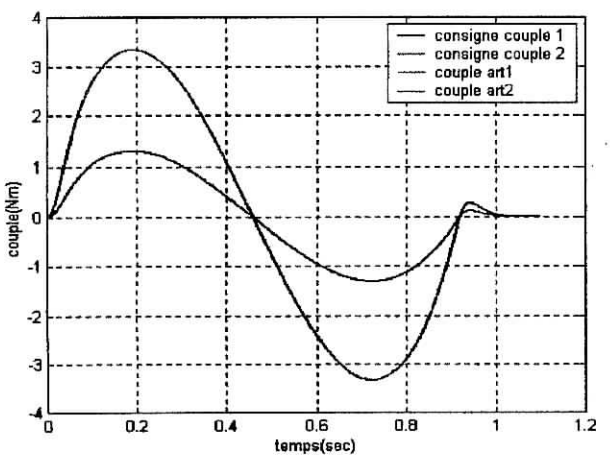


Figure 3.10.1.c *Suivi en courant.*

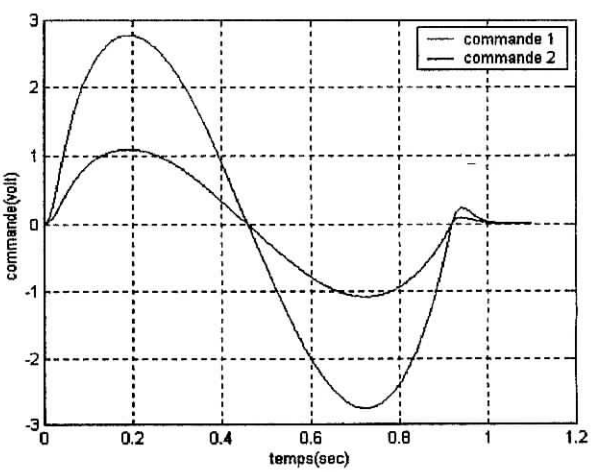


Figure 3.10.1.d *Signal de commande.*

Les données sont $q_i = [0 \quad 0]^T$, $q_f = [pi/90 \quad pi/180]^T$ et $m_0 = 4kg$, nous obtenons :

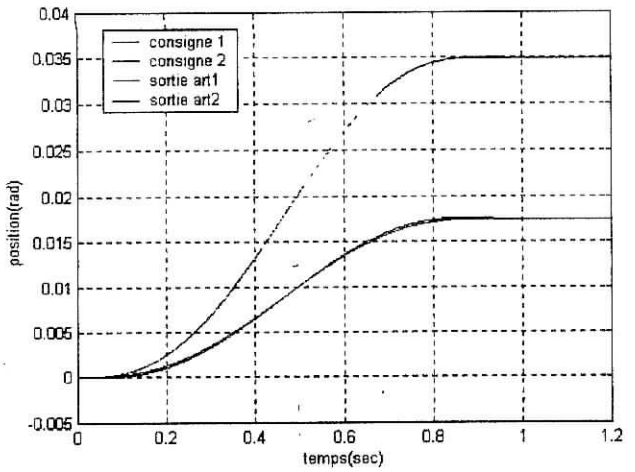


Figure 3.11.1.a *Suivi en position.*

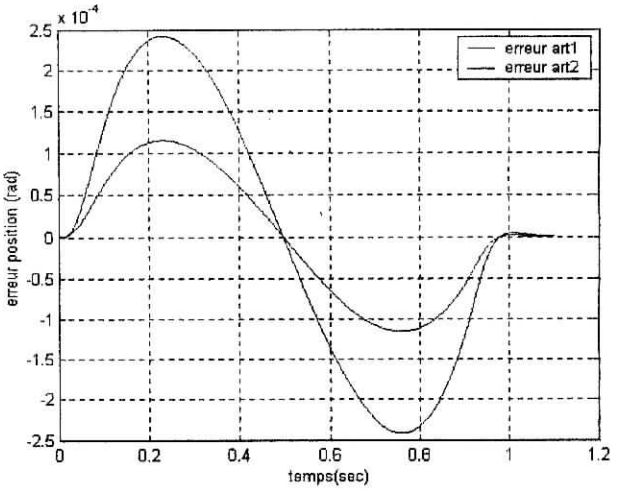


Figure 3.11.1.b *Erreur en position.*

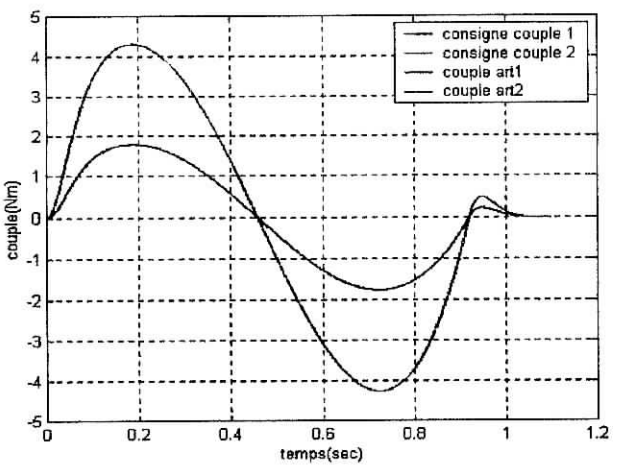


Figure 3.11.1.c *Suivi en courant.*

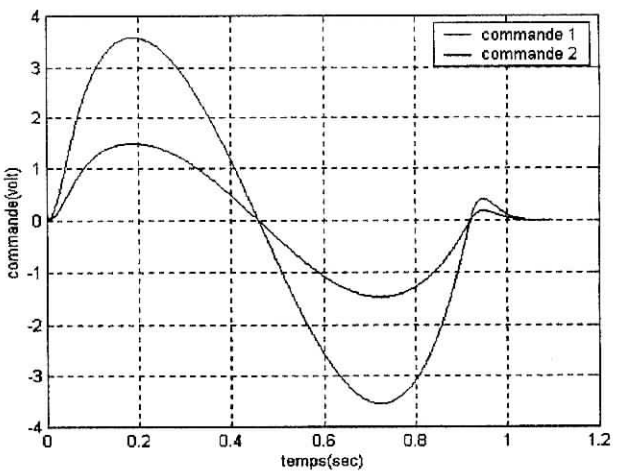


Figure 3.11.1.d *Signal de commande.*

3.8. Conclusion

Les simulations précédentes montrent l'importance de la commande en cascade , en robotique en cherche à ce qu'une articulation suive une trajectoire bien définie, cela ne signifie pas seulement une commande en position, il est nécessaire que le couple appliqué à l'articulation soit bien celui calculé par le régulateur ; comme les actionneur qui sont des moteur dans notre cas, ont une dynamique dont il ne faut pas négligée leurs influence sur la commande ; Sur se qui a été fait précédemment, nous avons montré qu'une boucle interne de commande du courant permettait d'approchée le plus du système réel en incluant la dynamique des moteur dans le modèle du robot manipulateur.

CHAPITRE IV :
DISCRITISATION ET PRISE EN COMPTE DE L'EFFET DE LA QUANTIFICATION

4.1. Introduction

La nécessité d'implémenter un algorithme de réglage sur calculateur de processus, nous amène à prendre en considération plusieurs approches et mettre en évidence le fonctionnement échantillonné d'un tel procédé de réglage, dans cette vision des choses, il nous est important de dimensionner les différents régulateurs (déjà vu dans la partie traitant l'influence de la dynamique des moteurs sur la boucle de réglage). Néanmoins, le faite d'utiliser un calculateur nous oblige a choisir un format adéquat pour représenter nos variables de travail, constantes de réglage et résolution d' E/S (quantification sur un certain nombre de Bits)

4.2. Echantillonnage

L'échantillonnage d'un signal entraîne, sous sa forme la plus simple, une transformation d'un signal continu en une série d'impulsion dont l'amplitude correspond a la valeur du signale continu à l'instant d'échantillonnage **[16]**, ce signal échantillonné étant non définie entre ces instants, ce qui correspond a une perte d'informations (informations énergétiques), donc l'utilisation d'un bloqueur s'avère nécessaire, il faut tenir compte du temps de calcul nécessaire pour une boucle de réglage (on l'introduira comme retard).

Les fréquences d'échantillonnages sont a choisir de manière à respecter le théorème de Shanon ($f_E \geq 2.\frac{1}{\tau}$) et sachant que la commande se fait en cascade pour prévenir d'une éventuel perturbation en couple qui correspond a un coefficient prés a la valeur du courant ($C_m = K_T.I_m$), nous somme obligé à travailler en double échantillonnage, l'un pour la boucle de position, et l'autre en boucle de courant (Figure 4.1).

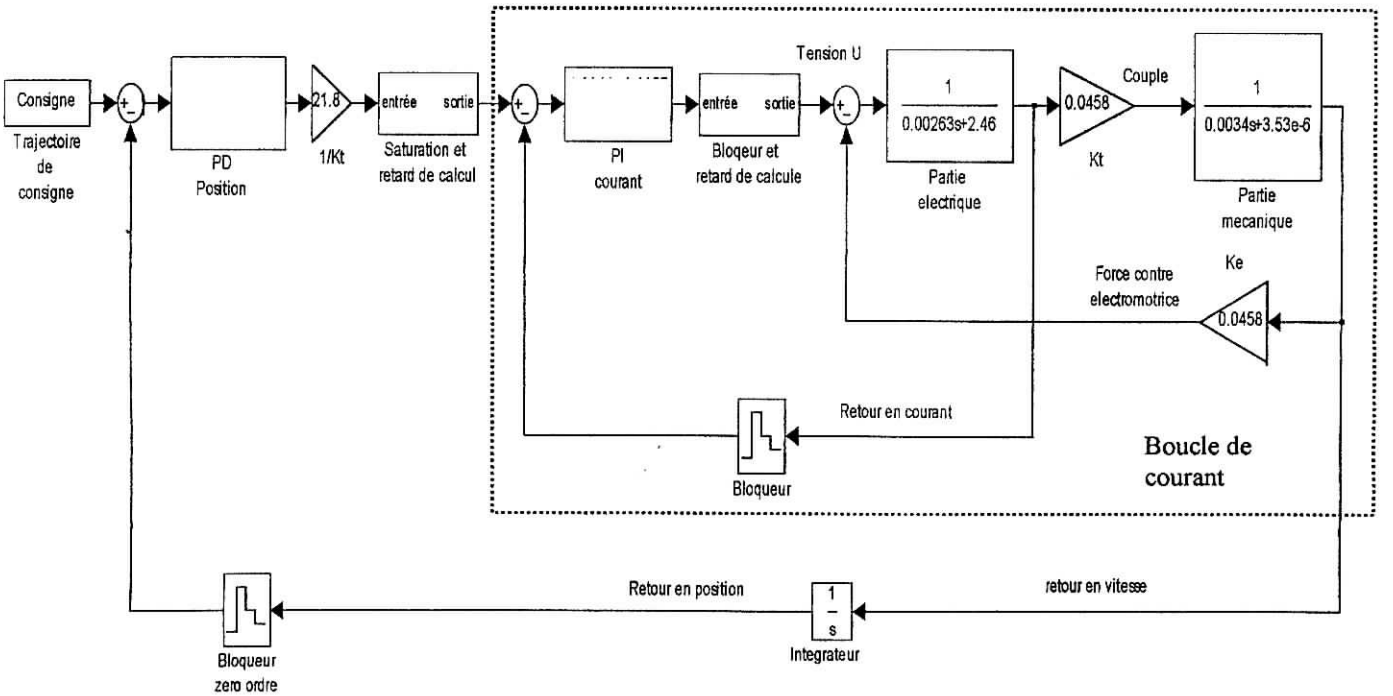


Figure 4.1. *Schéma bloc du réglage échantillonné en cascade.*

Sachant que les constantes de temps électrique et mécanique de notre actionneur (moteur DC, PITTMAN GM 9236) sont :

- $\tau_E = 1{,}06 \cdot 10^{-3}$ S
- $\tau_M = 8{,}5 \cdot 10^{-3}$ S

On choisi $T_{E,pi} = 0{,}2 \cdot 10^{-3}$ s ; pratiquement égale1/5 de τ_E
$T_{E,pd} = 2 \cdot 10^{-3}$ s ; pratiquement égale 1/4 de τ_M

On obtient 10 boucle de réglage en courant pour une seul en position.

4.2.1. Etude du régulateur PID échantillonné

Pour les régulateur digitaux, on peut travailler directement sur l'équation aux différences, pour laquelle, l'intégrale est remplacée par une somme, la dérivée par une différence et la deuxième dérivé (si elle existe) par une différence du deuxième ordre.

A l'instant k, on donne la sortie d'un régulateur PID par **[3]** :

$$y_r = K_p.e[k] + K_i.\sum_{i=0}^{k} e[i] + K_d.(e[k] - e[k-1]) \qquad (4.1)$$

Les coefficients k_p, k_i et k_d du régulateur continu sont remplacés par K_p, K_i et K_d .

Pour établir l'algorithme a implémenter sur microcontrôlleur, on doit élaborer la somme de manière récursive. Dans ce but, on introduit une variable auxiliaire pour la composante intégrale **[2]**:

$$X_R[k] = K_i \sum_{i=0}^{k-1} e[i] \qquad (4.2)$$

On obtient ainsi :

$$X_R[k+1] = X_R[k] + K_i.e[k] \qquad (4.3)$$

A l'aide de cette grandeur auxiliaire, la relation (4.1) devient :

$$y_R[k] = X_R[k] + K_{pid}.e[k] - K_d e[k-1] \qquad (4.4)$$

avec :

$$K_{pid} = K_p + K_i + K_d \qquad (4.5)$$

Tout les autres régulateurs sont une forme dérivé du régulateur PID ainsi détaillé.

L'algorithme de réglage qu'on doit utiliser devra contenir un élément saturateur correspondant a la limitation matérielle des composants utilisés, cette saturation ce fait par programme, pour éviter toute surcharge ou éventuelle surtension dû aux perturbation :

4.2.1.1. Algorithme PID avec saturation incorporée

Lire y
$e = w - y$
$y'_R = x_R + K_{PID}\, e - K_d\, e_{-1}$
si $y'_R > y_{Rmax}$
alors $y_R = y_{Rmax}$
sinon si $y'_R < y_{Rmax}$
alors $y_R = y_{Rmin}$
sinon $y_R = y'_R$
sortir y_R
$e_{lim} = e - (y'_R - y_R)/K_{PID}$
$x_R = x_R + K_i\, e_{lim}$
$e_{-1} = e$
fin

On a vu au chapitre sur la simulation de la dynamique du robot que les régulateur utilisé était des régulateur continu sauf que le calculateur travail en mode échantillonnage, qui va correspondre a des régulateur en équations aux différences, on montre qu'il existe un passage entre la fonction continu du régulateur et sa fonction échantillonnée **[3]** ; sur ce qui suit.

4.2.2. Fonction de transfert pseudo-continu du régulateur PD

La fonction de transfert d'un régulateur digitale doit être exprimée par la transformation en Z. Cependant avec un peu moins de rigueur on l'exprimer aussi par la transformation de Laplace **[2]**, soit :

$$d[k] = e[k] - e[k-1] \tag{4.6}$$

la transformée de Laplace de la différence :

$$d(s) = e(s) - e(s).\, e^{-s.T_E} = (1 - e^{-s.T_E})\, e(s) \tag{4.7}$$

ou approximativement

$$d(s) \cong \frac{s.T_E}{1 + s.T_E}.e(s) \tag{4.8}$$

Lorsqu'on introduit cette relation dans (4.1), on obtient :

$$R(s) = \frac{y_R(s)}{e(s)} \cong \frac{s.T_E}{1 + s.T_E/2}.K_d + K_p \tag{4.9}$$

$$\Rightarrow R(s) = \left(\frac{s.T_E.(K_d + \frac{K_p}{2}) + K_p}{1 + s.T_E/2} \right) \tag{4.10}$$

$$\Rightarrow R(s) = K_p.\left(\frac{1+s.T_v}{1+s.T_E/2}\right) \qquad (4.11)$$

On obtient directement les coefficients du régulateur digitale équivalent :

- $K_p = k_p$
- $K_d = \frac{T_v - T_E/2}{T_E}.k_p$ (4.12)

Note:
On doit noter que l'approximation (4.10) introduit une constante de temps $T_E/2$ supplémentaire qui doit être prise en considération comme petite constante lors de la simulation du circuit de réglage, cette constante de temps peut être négligé que si la dynamique du système est très lente par rapport a cette valeur.

4.2.3. Fonction de transfert pseudo-continu du régulateur PI

De la même manière on calcule la pseudo-continu du transfert du régulateur PI **[2]**

$$f[k] = \sum_{i=0}^{k} e[i] = f[k-1] + e[k] \qquad (4.13)$$

par la transformée de Laplace de cette somme donne :

$$f(s) = f(s).e^{-s.T_E} + e(s) \qquad (4.14)$$

$$\Rightarrow \quad f(s) = \frac{1}{1-e^{-s.T_E}}.e(s) \qquad (4.15)$$

Après approximation on obtient :

$$\Rightarrow \quad f(s) = \frac{1+s.T_E/2}{s.T_E}.e(s) \qquad (4.16)$$

Lorsqu'on introduit cette relation dans (4.1), pour un régulateur PI, on obtient :

$$R(s) = \frac{y_R(s)}{e(s)} \cong \frac{1+s.T_E/2}{s.T_E}.K_i + K_p \qquad (4.17)$$

$$\Rightarrow \quad R(s) = \frac{K_i + s.T_E(K_i/2 + K_p)}{s.T_E} \qquad (4.18)$$

$$\Rightarrow \quad R(s) = \frac{1+s.T_n}{s.T_i} \qquad (4.19)$$

On obtient directement les coefficients du régulateur digitale équivalent :

- $K_i = \frac{T_E}{T_i}$

- $K_p = \dfrac{T_n - T_E/2}{Ti}$ (4.20)

RMQ:
On voit que le terme intégrale du régulateur PI n'introduit aucune constante de temps, ce qui était le cas du terme dérivé dans le régulateur PD.

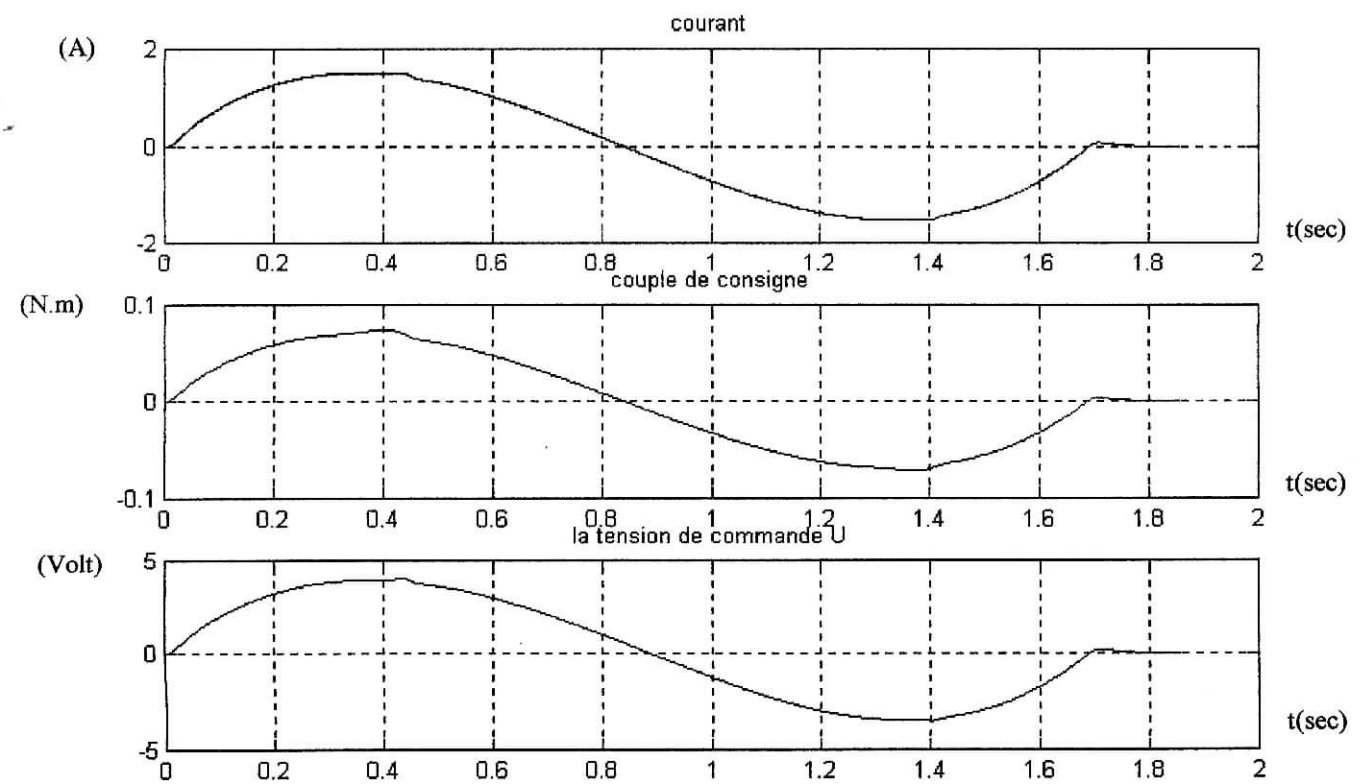
En se basant sur les précédentes relations et après calculs, on obtient :

$$R_{PD}(Z) = \frac{138{,}72.Z - 133{,}28}{Z} \quad (4.21)$$

$$R_{PI}(Z) = \frac{1{,}0944.Z - 0{,}9057}{Z - 1} \quad (4.22)$$

On peut voir l'effet de l'échantillonnage sur la réponse du système a une trajectoire polynomiale de degré trois.

$C_{pert} = 0$; pos_init=0; pos_fin=10; $I_{max} = 1{,}5A$



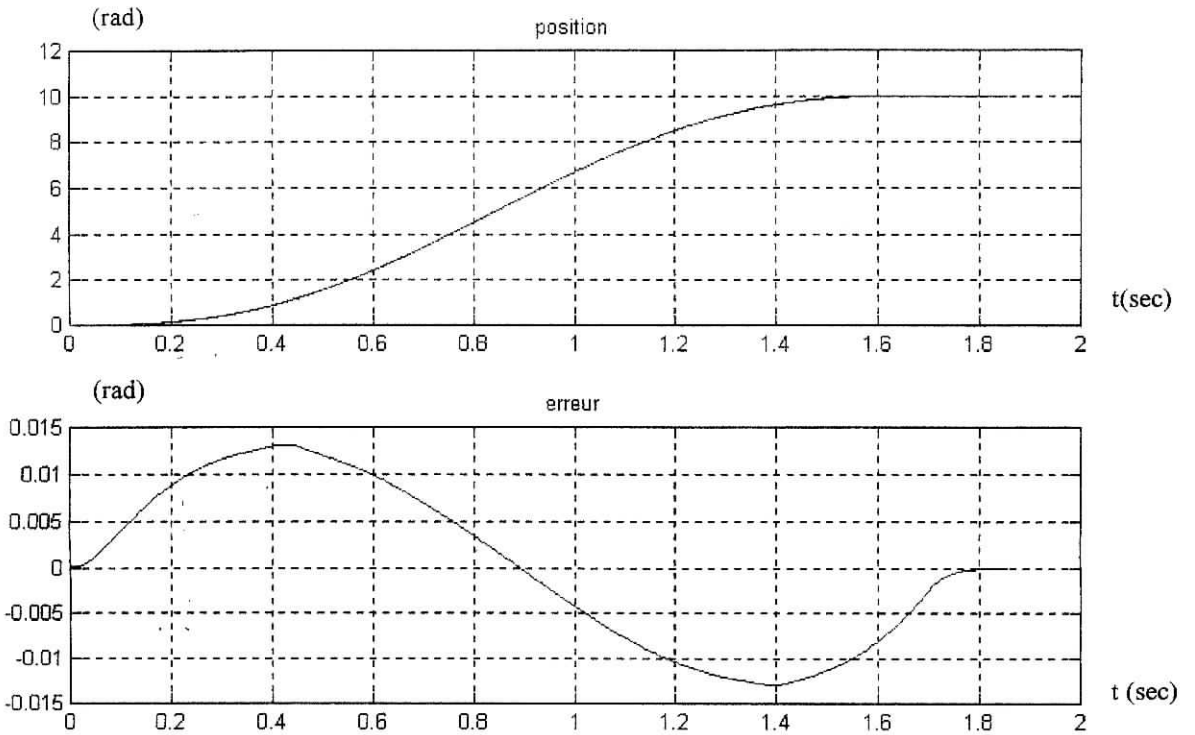
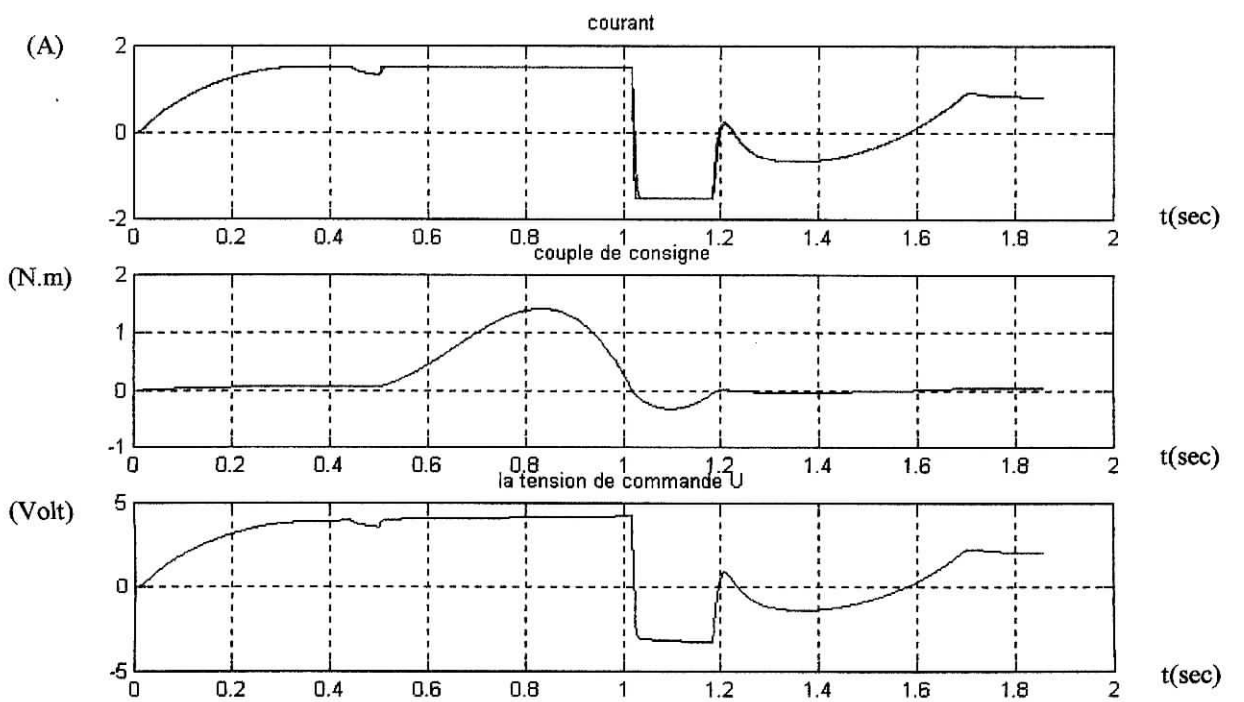


Figure 4.2. *Réponse en commande échantillonné, sans perturbation.*

C_{pert} = 0,0382 N.m (à t = 0.5 S) ; pos_init=0 ; pos_fin=10 ; I_{max} = 1,5A ;

RMQ: le couple perturbateur exerce son effet sur le rotor du moteur, ce qui donne un équivalent de $C_{pert\ /\ Articulation} = C_{pert} . 65,5 = 2,5$ N.m ;



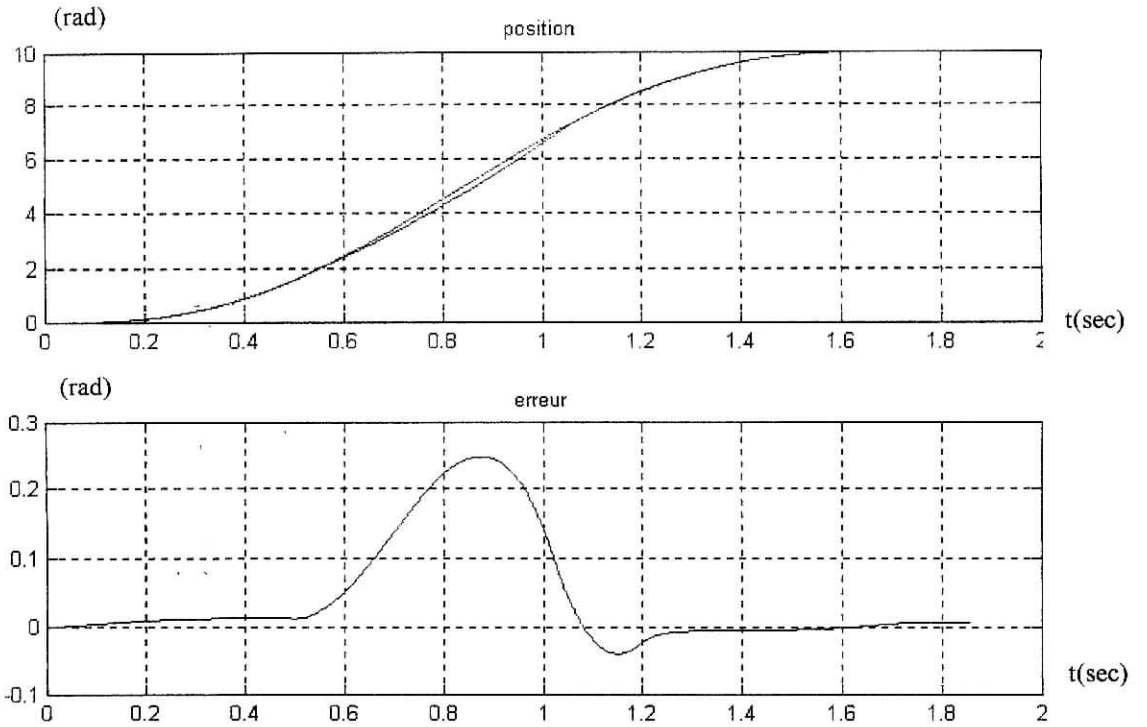


Figure 4.3. *Réponse en commande échantillonné, avec perturbation.*

On constate que l'effet de l'échantillonnage et pratiquement inexistant, tant qu'on respecte les conditions d'utilisation (théorème de Shanon), par contre on remarque la saturation apparue en simulation avec perturbation, ce détail ne peut être négligé, car il est a la base de la protection du matériel utilisé sinon cette saturation n'existera pas, le régulateur compense cette perturbation avec une très bonne réponse.

4.3. Quantification des grandeurs :

Une grandeur physique ne peut être représentée par une grandeur digitale qu'avec une précision limitée car elle ne peut varier que par un pas correspondant à un certain nombre de bits "n_q", la variation par un pas est appelé quantification, l'erreur de quantification relative et donnée par $\frac{\Delta U_q}{U_{max}} = \frac{1}{2^{n_q}-1}$; ce qui correspond exactement a un pas de quantification au maximum **[2]**.

4.3.1. Quantification de l'entrée de consigne :

La trajectoire générée comme consigne d'entrée doit être mise a l'échelle du microcontrôlleur, si on considère les butée mécanique que présente le robot SCARA, par ex. $-150° \leq \theta_1 \leq 150°$, qui donne un maximum de déplacement de 300° sur l'articulation et donc l'équivalent de: 342,9572 rad (19650°) sur l'axe du rotor du moteur.

Le script MATLAB permettant de passer d'une grandeur réelle U en grandeur quantifié Y selon un nombre de bits nq et un Umax est donné par :

```
>> pas = Umax/((2^nq)-1);
>> Y = pas*fix(U/pas);        %la commande fix permet de tronqué la valeur en son nbre entier
```

- Soit pour : Umax = 342,9572 rad ; nq = 16 bits ; U est une trajectoire polynomiale d°5 de 0 rad ➔ 10 rad ;

La visualisation de l'effet d'échantillonnage et de quantification donne :

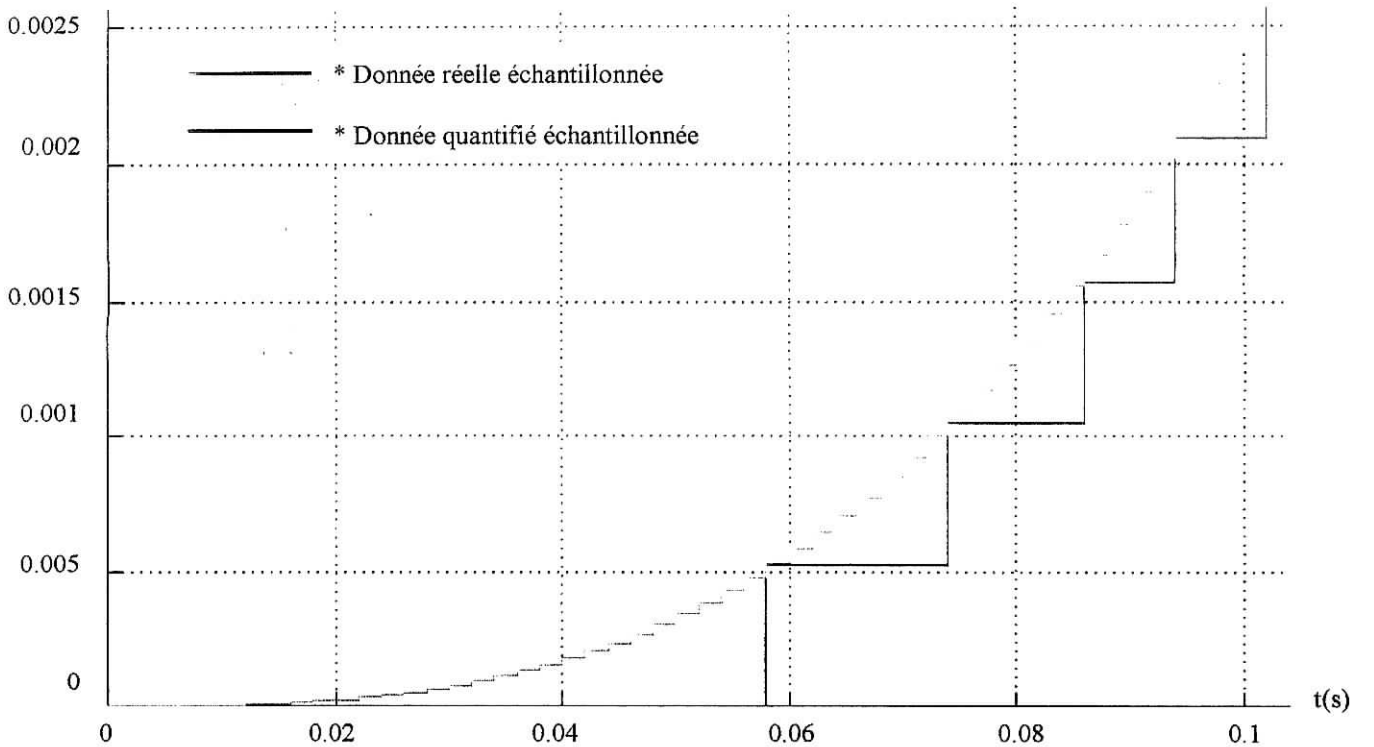


Figure 4.4. *Effet échantillonnage – quantification.*

4.3.2. Quantification de l'information de la boucle de retour :

Pour un retour simple de données comme pour un convertisseur A/N les données seront quantifiées de la même manière que pour les données d'entré, ce qui est le cas des grandeurs de la boucle de courant, l'exception faite pour le capteur de position, car tout simplement il s'agit d'un capteur incrémentale qui génère deux signaux carrées en quadrature de phase (Figure 4.5) et un signale d'index qui génère une impulsion pour chaque révolution (tour), ces signaux notés Ch.A ; Ch.B ; Ch.I, l'unité est en CPR (count per revolution), Figure 4.5.

Celui du PITTAMN GM 9236 est un codeur de la série 91XX avec une résolution de 500 CPR/Ch.x , ce qui nous permet de multiplier cette précision par 2 ou par 4, c'est possible par simple considération des deux front montant et descendant des deux canaux A et B. voir figure 4.6.

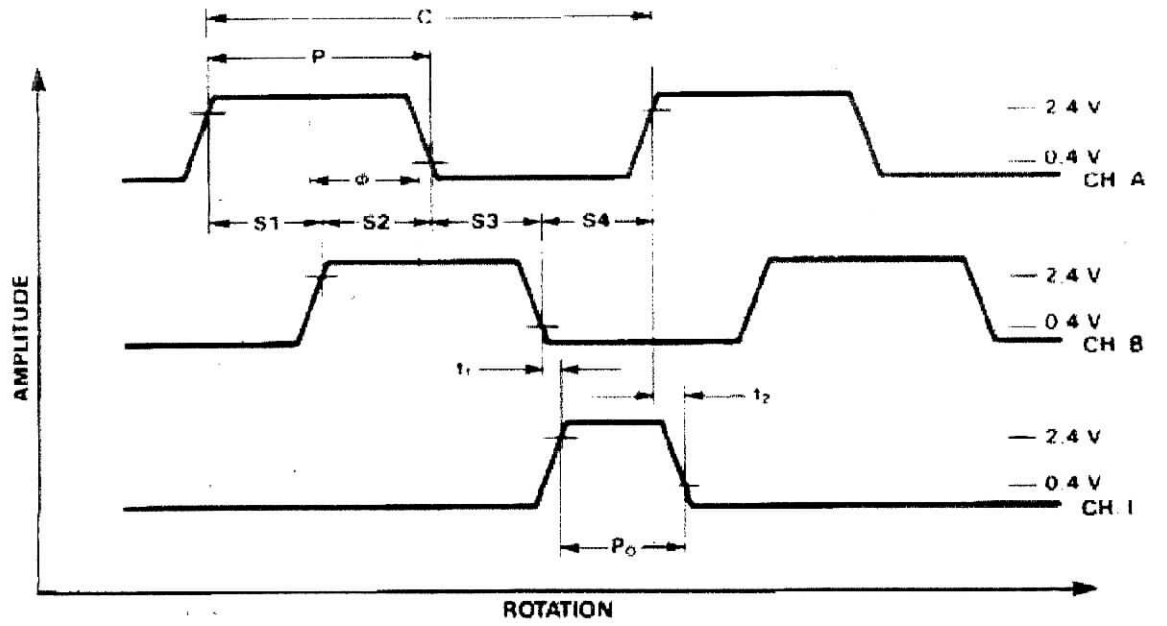


Figure 4.5. *Format d'impulsion généré par le codeur incrémentale.*

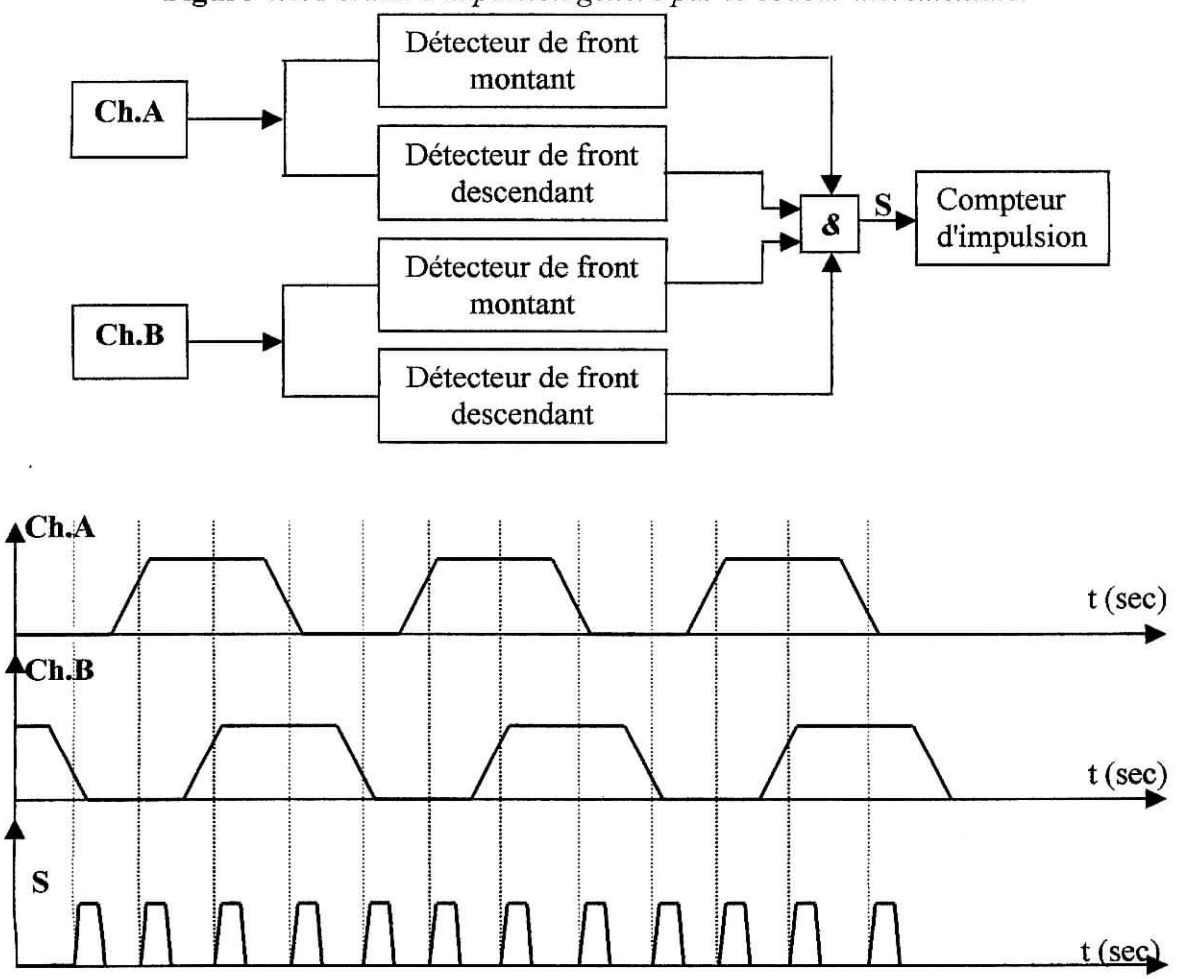


Figure 4.6. *Principe d'augmentation de la résolution sur double canal.*

Si on compte pour ce codeur qui a comme résolution 500 CPR/Ch.x, on obtient 2000 CPR pour les deux canaux, ce qui peut être codée pour une rotation articulaire maximale de 300° "butée mécanique" (l'équivalent de 342,9572 rad ou 19650° à l'arbre du moteur) sur 17 bits, car tout simplement ce système générera exactement $1{,}0917.10^5$ impulsions durant cette traversé.

On adopte le même principe si on veut avoir une résolution de 1000CPR sauf qu'on considère seulement les fronts montants et descendants d'un seul canal. L'information sur la position peut ainsi être codée sur 16 bits, car le nombre d'impulsions généré est de $5{,}4583.10^4$

Voir annexe concernant l'étude du capteur incrémentale et la circuiterie correspondante, pour conditionnement, gestion du sens et filtrage digitale.

4.3.3. Effet de la quantification Consigne/Capteur

On peut voir l'effet de la quantification sur la réponse du système par simulation discrète selon les paramètres suivants:

- pos_init = 0 ; pos_fin = 10 ; I_{max} = 1,5A ; I_{sat} = ±3A ; capteur = 500CPR ; nq = 16 bits ; $T_{E,pi} = 0{,}2 .10^{-3}$ s ; $T_{E,pd} = 2 . 10^{-3}$ s ; C_{pert} = 0 N.m ;

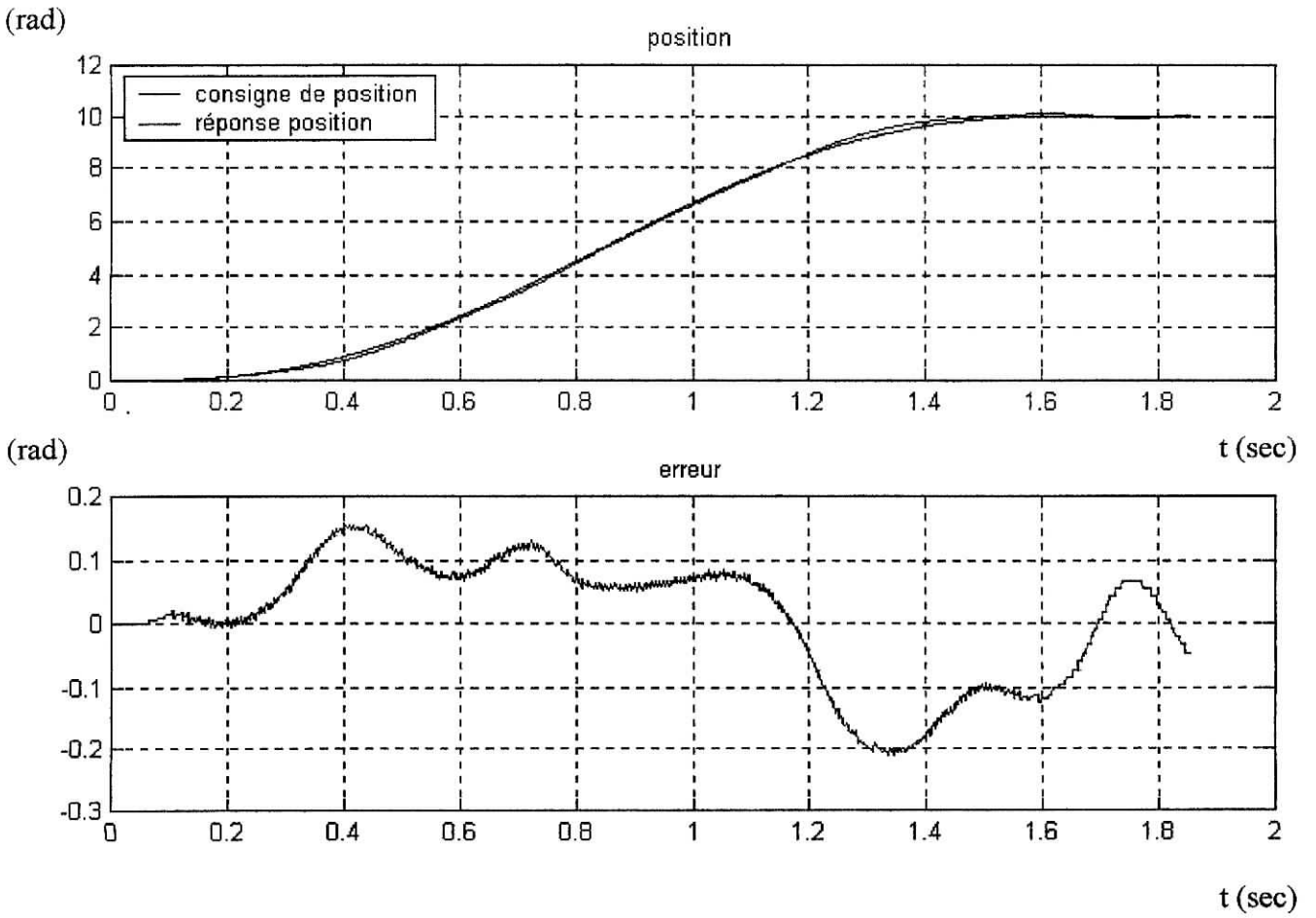


Figure 4.7. *Sortie en position.*

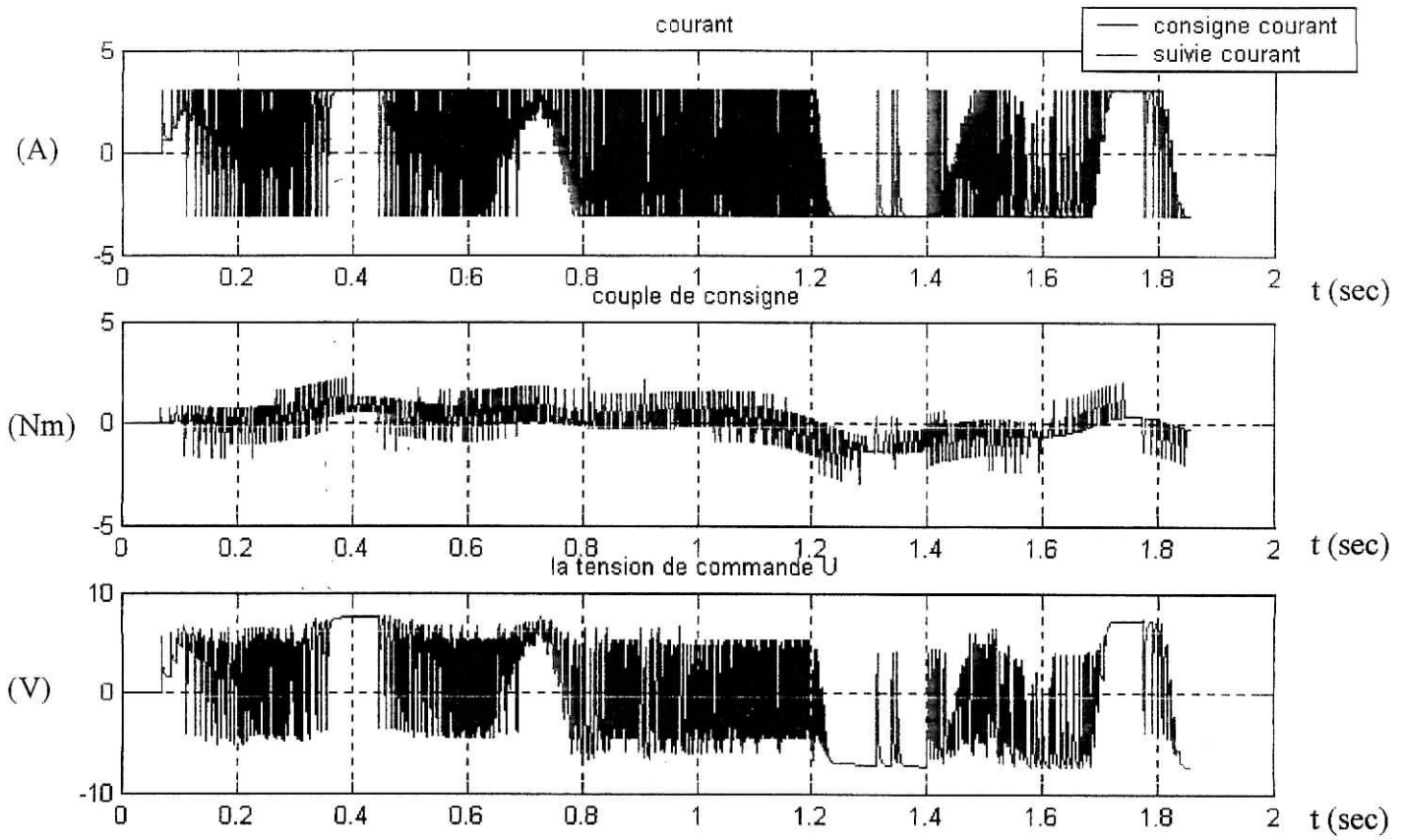


Figure 4.8. *Grandeurs de commande.*

On peut voir aussi la perte d'informations concernant l'erreur entre la vraie position et la position vu par le microcontrôlleur à partir du compteur lié aux canaux du capteur incrémentale, voir figure 4.9.

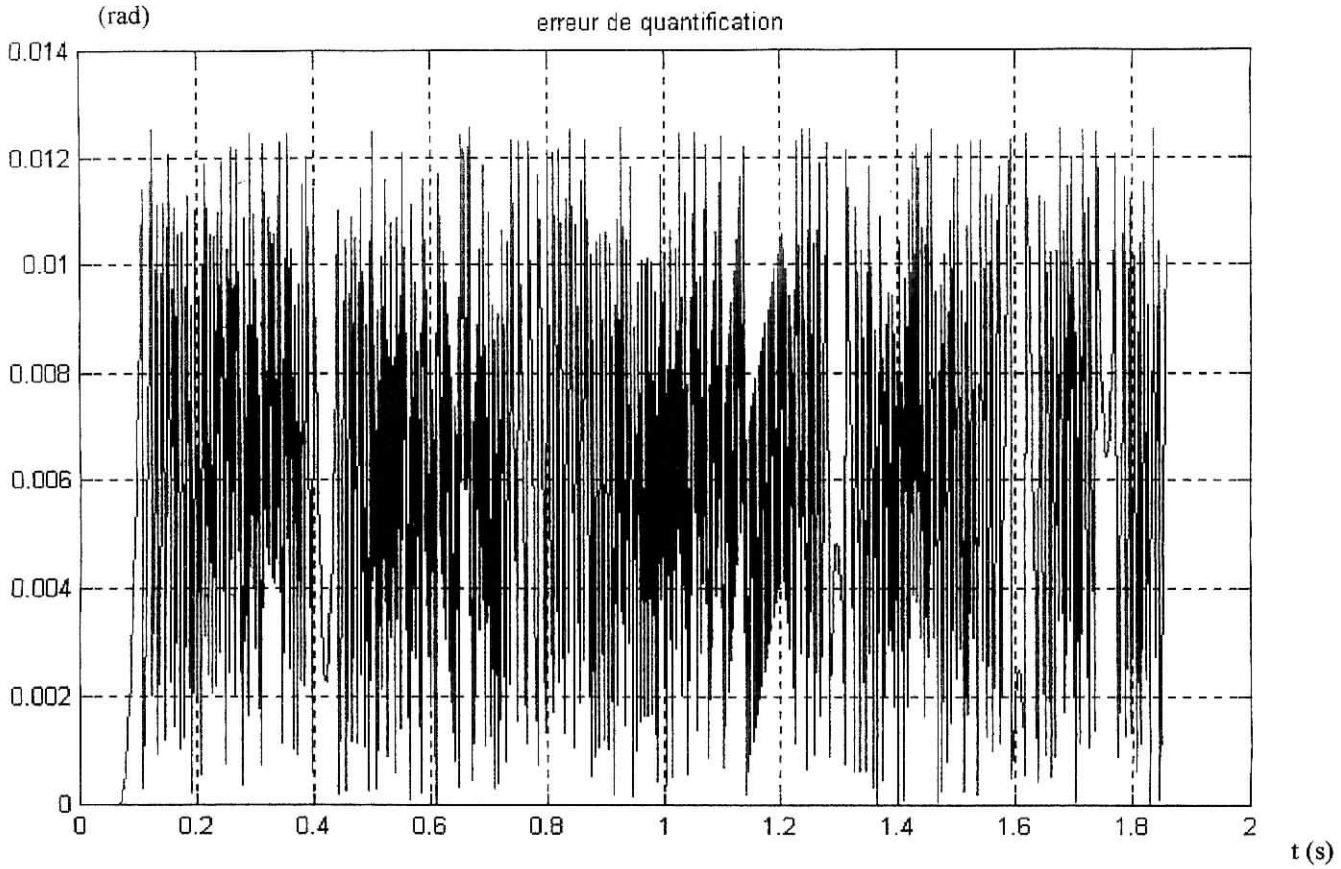


Figure 4.9. *Erreur du retour en position "erreur dû a la quantification"*

Effectivement, cette erreur ne pourra pas dépasser un pas de résolution, certes, car toute variation détectable sera lue par le calculateur le pas de résolution est donné par :

$\text{Pas} = \frac{360.65{,}5.\text{pi}}{180.\text{rés}}$; pour un rés de 500 on obtient : Pas = 0.0126 rad/incrément

Pratiquement irréalisable, la commande destinée a mettre en évidence le suivi en vitesse–position, toute fois, elle reste ce qui doit être envoyer par le calculateur de commande si on désire travailler avec les performances exigées, on verra dans la prochaine section la position du problème et les solutions proposées pour y remédier.

- pos_init = 0 ; pos_fin = 10 ; I_{max} = 1,5A ; I_{sat} = ± 3A ; capteur = 2000CPR ; nq = 17 bits ; $T_{E,pi} = 0{,}2 .10^{-3}$ s ; $T_{E,pd} = 2 . 10^{-3}$ s ; C_{pert} = 0 N.m ;

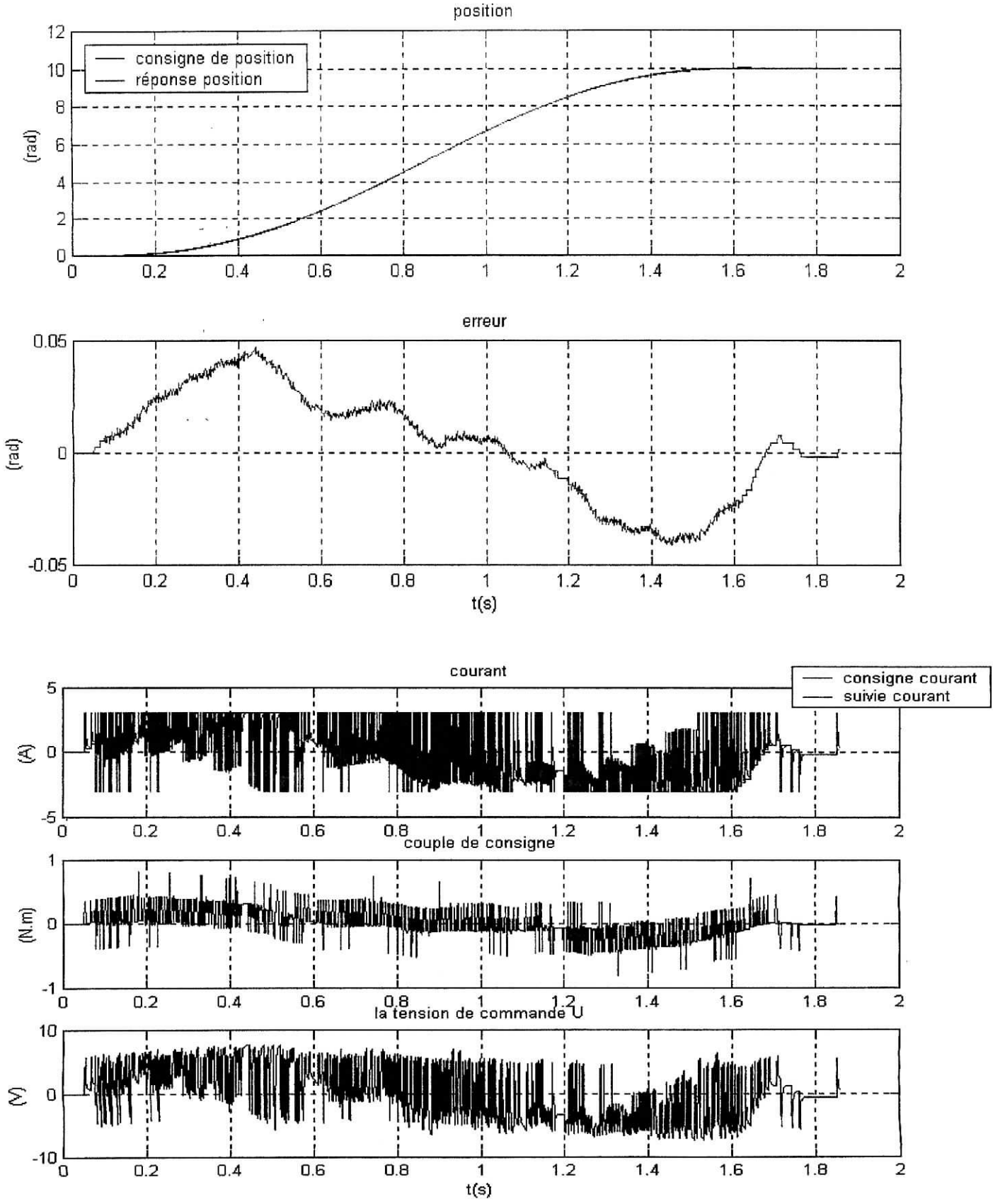


Figure 4.10. *Sortie en position, erreur de suivi, profil de commande.*

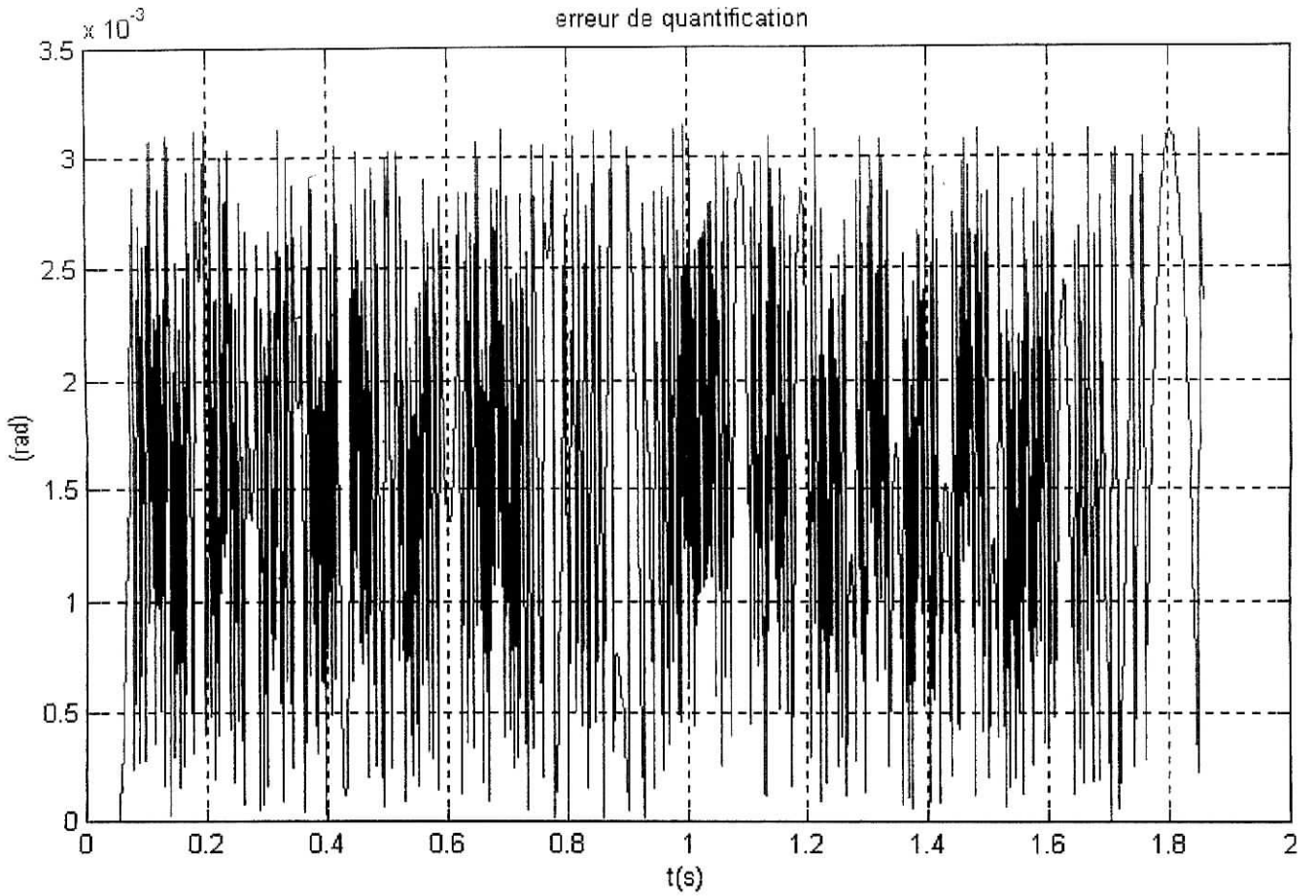


Figure 4.11. *Erreur du retour en position "erreur dû a la quantification".*

De même, cette erreur ne peut dépasser le pas de résolution qui est de 0,0031 rad/incrément ; l'élaboration de la commande pour chaque cycle de réglage par le calculateur est introduite sous forme de retard pur estimé à **0,5 ms** pour la boucle de position et de **0,1 ms** pour la boucle de courant.

On voit aussi que le faite d'augmenter la résolution, les performances deviennent meilleurs vis à vis de l'erreur de poursuite et l'élimination du profil oscillatoire, aussi connu sous le nom de "cycle limite" ;

4.3.4. Position du problème :

l'erreur en position dû au capteur et la résolution adopté pour le codage de la consigne d'entrée rend le système très sensible aux variation brusque. Le régulateur est à réponse très rapide donc, notre boucle de réglage du courant qui est nécessaire à la compensation des effets perturbateurs, ne peut pas atteindre sa valeur en consigne, cette accumulation génère à son tour des gradients de consigne en courant très importants (limité à ± 3A maximum), de ce faite il en résulte une commande U qui a un spectre fréquentiel très riche.

Cette commande ne peut être implémenter sachant qu'on est limité a travailler au-dessous de 5MHz , fréquence limite du microcontrôlleur (fréquence en cycle instruction, pour plus de

détails, voir chapitre sur 16F877). De plus, la commande U sera appliqué sous forme d'une MLI (PWM), des conditions dépassants les performances de notre microcontrôlleur.

4.3.5. Solutions proposées :

- 1). Filtrage de la commande :

on se basant sur le principe que le moteur constitue en lui même un filtre passe bas en considérant le transfert tension/vitesse donné par G(s) et comme le montre le diagramme de Bode associer Figure 7.12.

$$G(s) = \frac{0{,}0458}{8{,}942.10^{6}.s^{2} + 0{,}008364.s + 0{,}002106} \qquad (4.23)$$

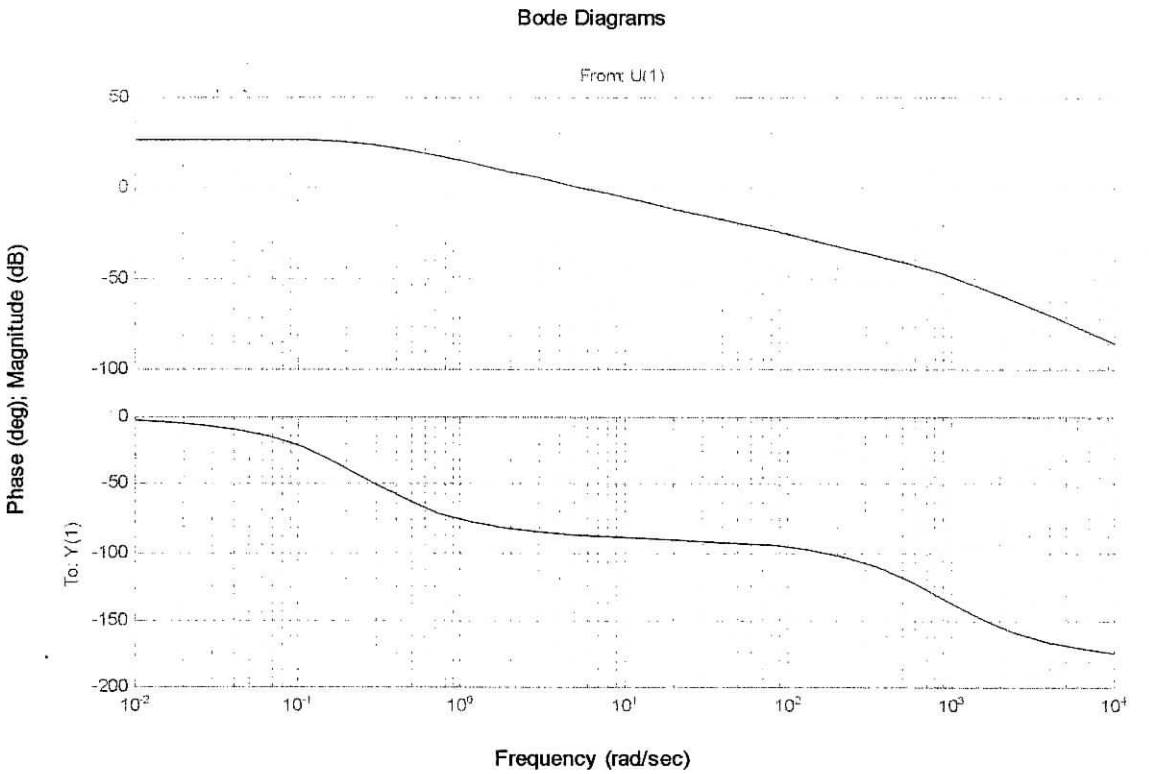


Figure 4.12. Diagramme de Bode du transfert tension/vitesse.

L'idée est d'introduire un filtre qui réduit l'effet des signaux dépassant $\omega = 5.47$ rad/sec, pour les attenuation de moins de 0 dB ou plus tolérant encore $\omega = 7.73$ rad/sec à -3 dB.

Les résultats de la simulation du premier cas donne :

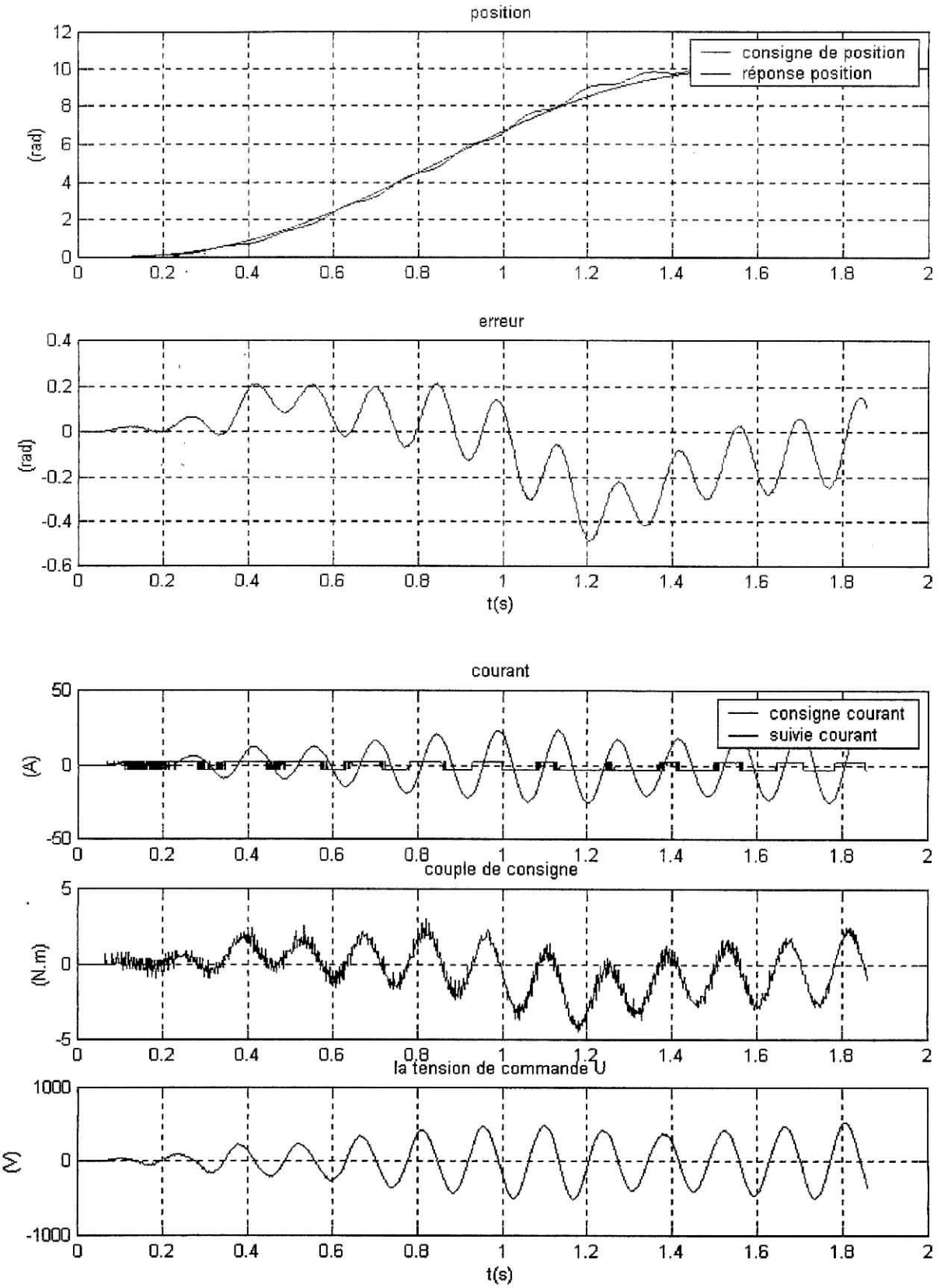


Figure 4.13. *Réponse et états interne du système de réglage .*

Pour le deuxième cas la simulation donne :

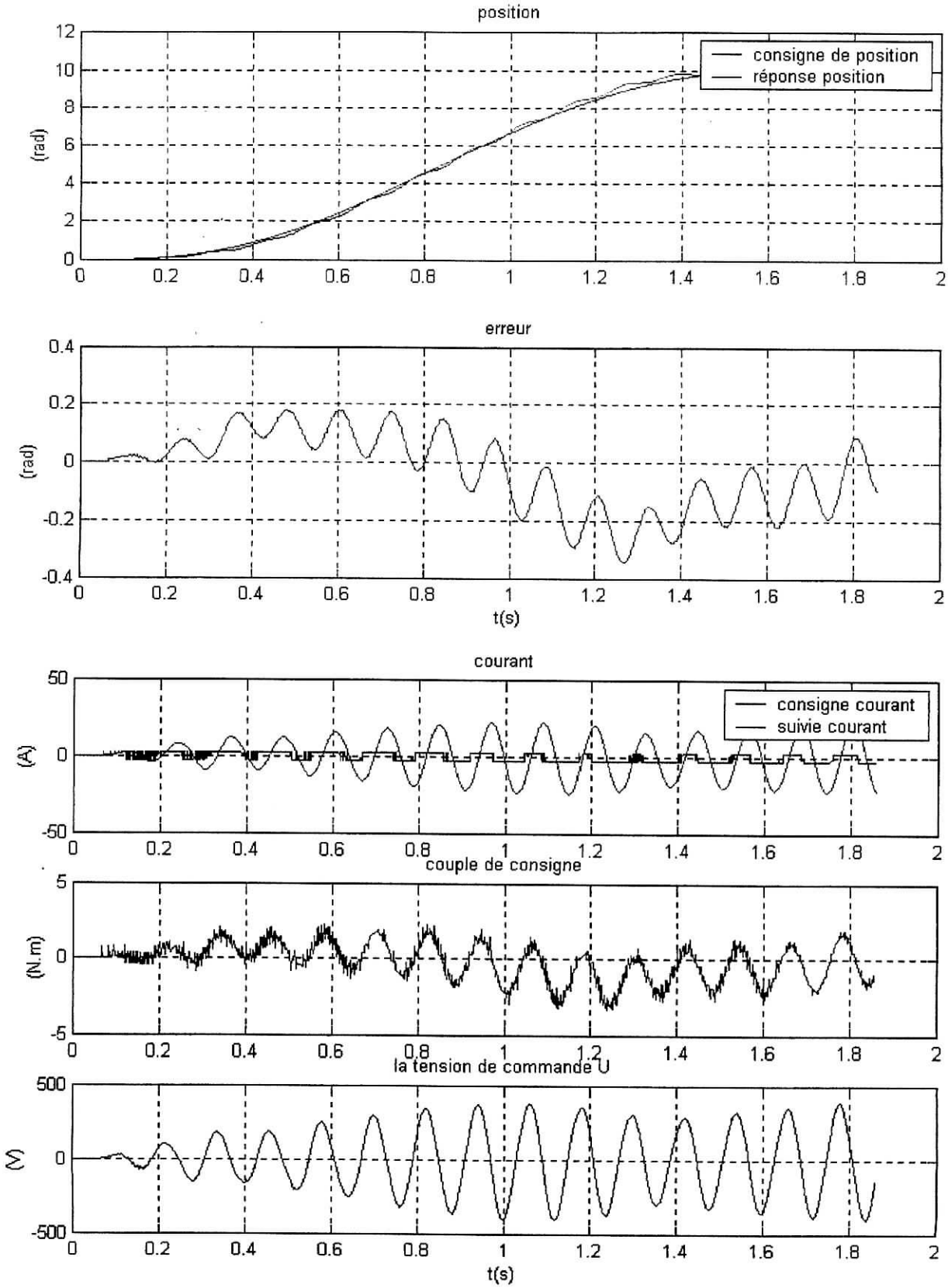


Figure 4.14. Réponse et états interne du système de réglage.

Interprétation des résultats :

On remarque tout d'abord que le suivi de trajectoire est oscillatoire non amorti, aussi on voit que la commande est très grande dépassant de loin la réalité de fonctionnement du moteur, que ça soit pour 0dB d'atténuation ou -3dB, pratiquement le filtrage change complètement la dynamique de réglage.

Le régulateur compense l'énergie du signale haute fréquence filtré par la même quantité en basse fréquence, sachant qu'un signale HF est très énergique par rapport a un signale BF, cela explique la grande valeur du signale destiné a piloté le moteur.

Donc, on peut pas filtré ce signale car toute ses fréquences sont utiles même si il est atténué par la bonde passante du moteur .

- 2). Augmenter la période d'échantillonnage de la boucle de position :

On a mentionné auparavant, que le courant n'atteint jamais sa valeur de consigne, ceci dû a la variation pour chàque période de réglage en position de cette consigne, alors on est dans l'obligation d'augmenter cette période d'échantillonnage, prenons une période de 10ms.

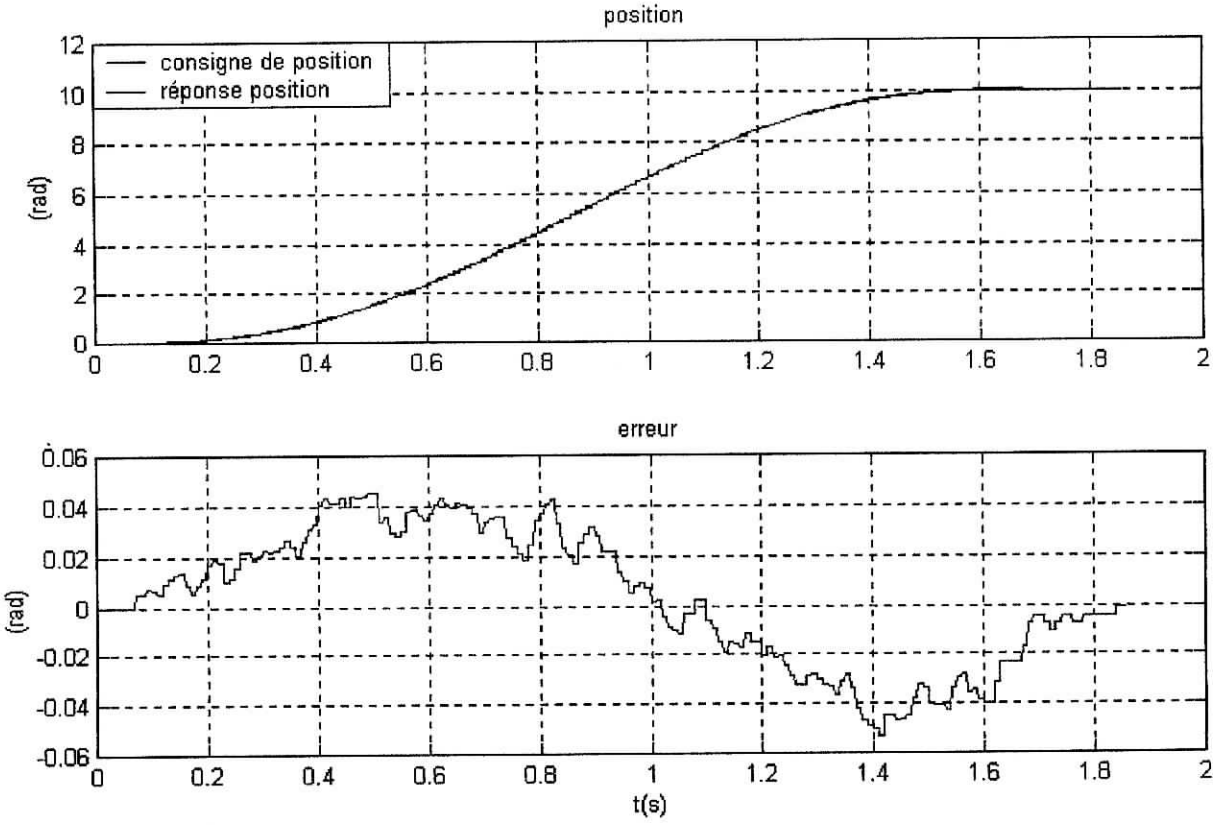


Figure 4.15. *Suivi en positon avec* $T_{E,mec} = 10ms$.

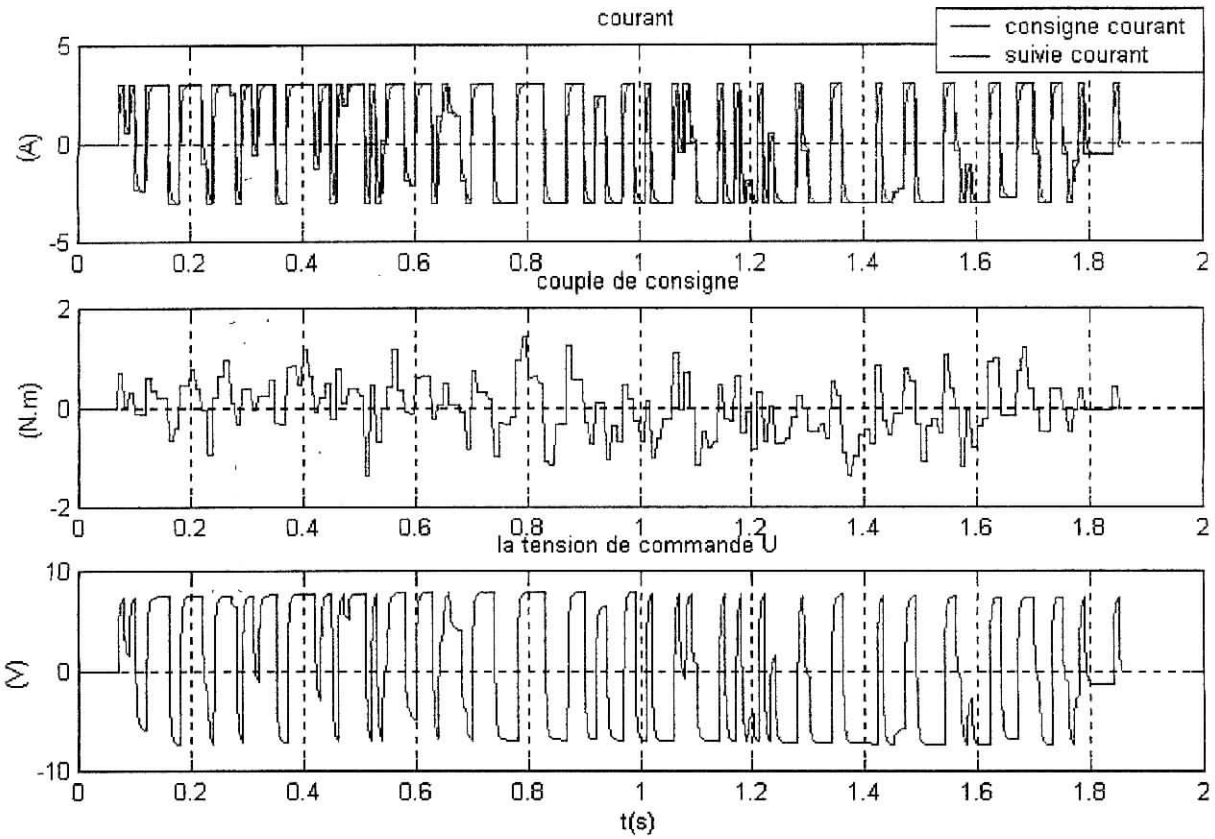


Figure 4.16. *Commande pour $T_{E,mec}$ = 10ms.*

On remarque que cette solution minimise le bruitage de commande vu précédemment, néanmoins l'existence de saturation dû au variation brusque de consigne (position et courant) génère un cycle limite en commande comme vous pouvez le constater sur la figure4.16, même si l'erreur de suivie tend vers 0, ce phénomène et le résultats de premièrement la non satisfaction de la condition de shanon sur les fréquences d'échantillonnages et deuxièmement a cause de la dynamique très rapide de la boucle de courant.

4.4. Conclusion :

Nous avons présenté dans ce chapitre le régulateur PID échantillonné et nous avons établie aussi la méthode permettant le passage du domaine continu au domaine discret moyennant un certain nombre d'approximations, la simulation des régulateurs digitaux équivalents donnent de très bonnes performances néanmoins toute grandeur digitale peut être représenté qu'avec un certain nombre limité de bits, ce qui nous a amené à prendre en considération l'effet de la quantification sous ses différents aspects, compte tenu de ces étapes, nous avons effectué les simulations et traitant à la fois les pertes d'informations résultantes, à la fin nous avons adopté l'approche pratique pour résoudre les problèmes de réglage (voir la procédure au chapitre 5).

CHAPITRE V :

MISE EN ŒUVRE PRATIQUE DE LA CARTE DE COMMANDE

5.1. Introduction

La commande en mouvement des robots manipulateurs est essentiellement réalisée par des circuits numériques programmables tel que les microprocesseurs, les microcontrôlleurs et les DSP où les lois de commande sont implémentées sous forme d'algorithmes , leurs principaux avantages sont la rapidité de développement, leurs flexibilité et leurs faibles espace d'occupation.

Dans ce chapitre, nous présentons la réalisation d'une carte de commande pour l'asservissement en position d'un moteur à courant continue afin de commander le robot SCARA.

5.2. Présentation et principes

le choix du microcontrôlleur PIC16F877 pour réalisée la commande en position d'un moteur à courant continue n'est pas quelconque, ce circuit de la firme Microchip **[9],[11]** répond aux exigences d'une telle application, nous proposons un circuit d'interfaçage de ce microcontrôlleur dont nous détaillerons les circuits indispensables et les différentes ressources dont il a besoin pour cette application.

Comme nous avons pu le constaté, le PIC16F877 offre une multitude de fonctions pour effectuer la commande d'un moteur à courant continue telle que :

- Mesure de la position du moteur.
- Calcul du profil du mouvement (génération de la trajectoire).
- Calcul du signal d'erreur et implémentation de l'algorithme de régulation (PID).
- Génération du signal de commande du moteur (PWM).

la figure 5.1 représente le synoptique de la carte de commande du moteur à courant continue.

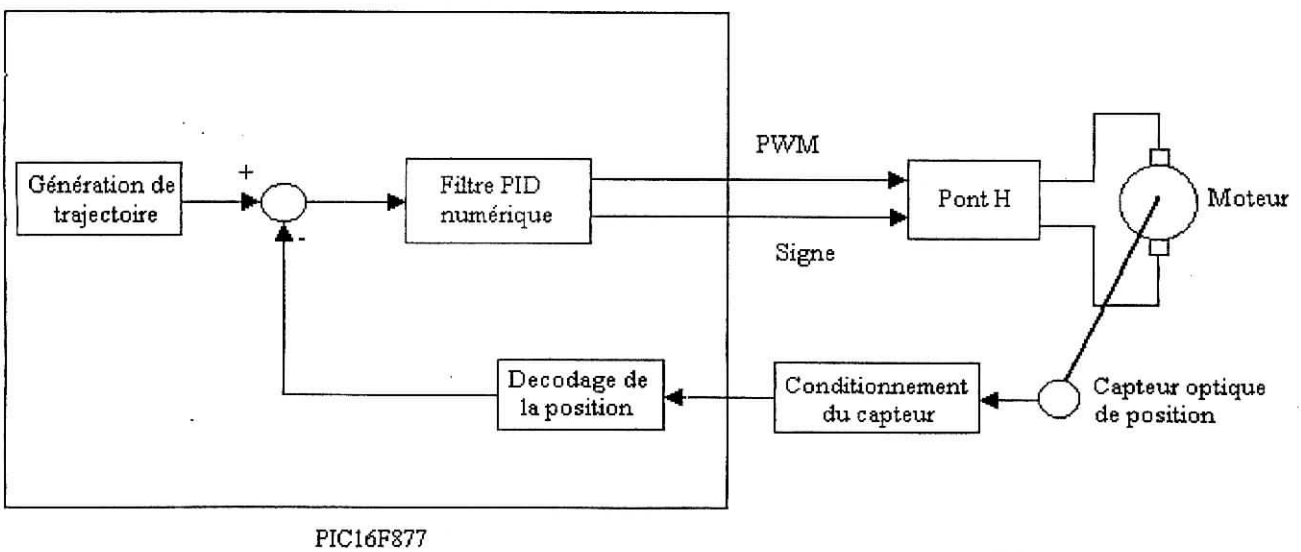


Figure 5.1. *Bloc diagramme du système de commande.*

5.3. Mesure de la position du moteur

Afin de pouvoir agir sur la commande, le calculateur doit disposer de la position actuelle du moteur, étant donnée que le capteur fourni deux signaux en quadrature, un circuit de décodage permet de générer deux trains d'impulsions haut et bas à la sortie des bascules D (figure A.3.3.a), ces dernières sont connectées aux bornes RA4/T0CKL et RC0/T1CKL afin d'accumuler les impulsions hautes et basses, à partir des deux valeurs obtenues, nous déterminons la position réelle du moteur, cette position est représentée dans un registre **POSR** ayant une taille de 16Bits.

l'organigramme suivant donne la façon dont la position est calculée.

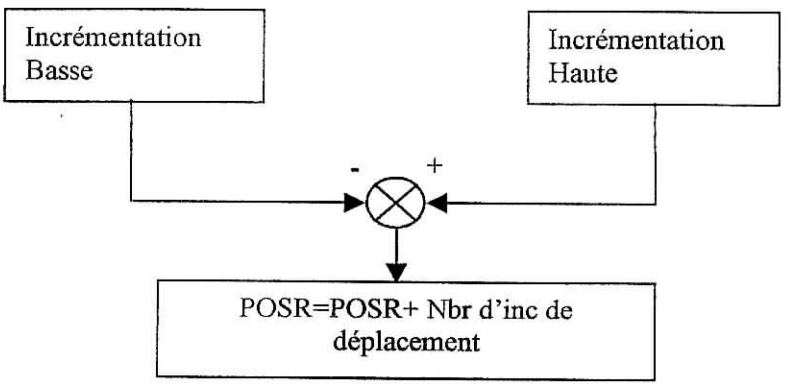


Figure 5.2. *Organigramme de calcul de la position du moteur.*

5.4. Génération de la trajectoire

La position réelle du moteur ainsi connue, nous allons déterminer la consigne de position du moteur , cette valeur est stockée dans une variable POS, la taille de cette variable et de 32 bits, seule le 16 bits les plus significatifs sont pris en compte, les 16 bits les moins significatifs représentent la partie fractionnaire de la position désirée et cela afin d'avoir une meilleure précision.

Un profil trapézoïdal de vitesse a été choisit pour le calcul de la trajectoire désirée, la raison est la facilité d'implémentation et le faible temps de calcul ; Pour générer cette trajectoire il est nécessaire d'introduire plusieurs paramètres qui sont **[11]**:

- Position final POSF ;
- La vitesse maximal Kv ;
- L'accélération Ka ;
- Nombre d'échantillons de la phase vitesse constante Nbr2 et la phase de décélération Nbr3.

Le Timer2 représente la base de temps de notre circuit de commande, les différents paramètres ont un nombre d'incrément comme grandeur de position et un nombre d'échantillons comme grandeur de temps.

A chaque cycle d'échantillonnage, quand le profil demande une nouvelle accélération le registre d'accélération est additionné avec le registre de vitesse qui à son tour additionné au registre de position. Quand la demande d'augmenté la vitesse s'arrête seul le registre de

vitesse est additionné au registre de position, cela représente l'implémentation d'une génération de trajectoire de type trapézoïdal **[11]**.
la figure 5.4 donne l'organigramme de la génération de la trajectoire.

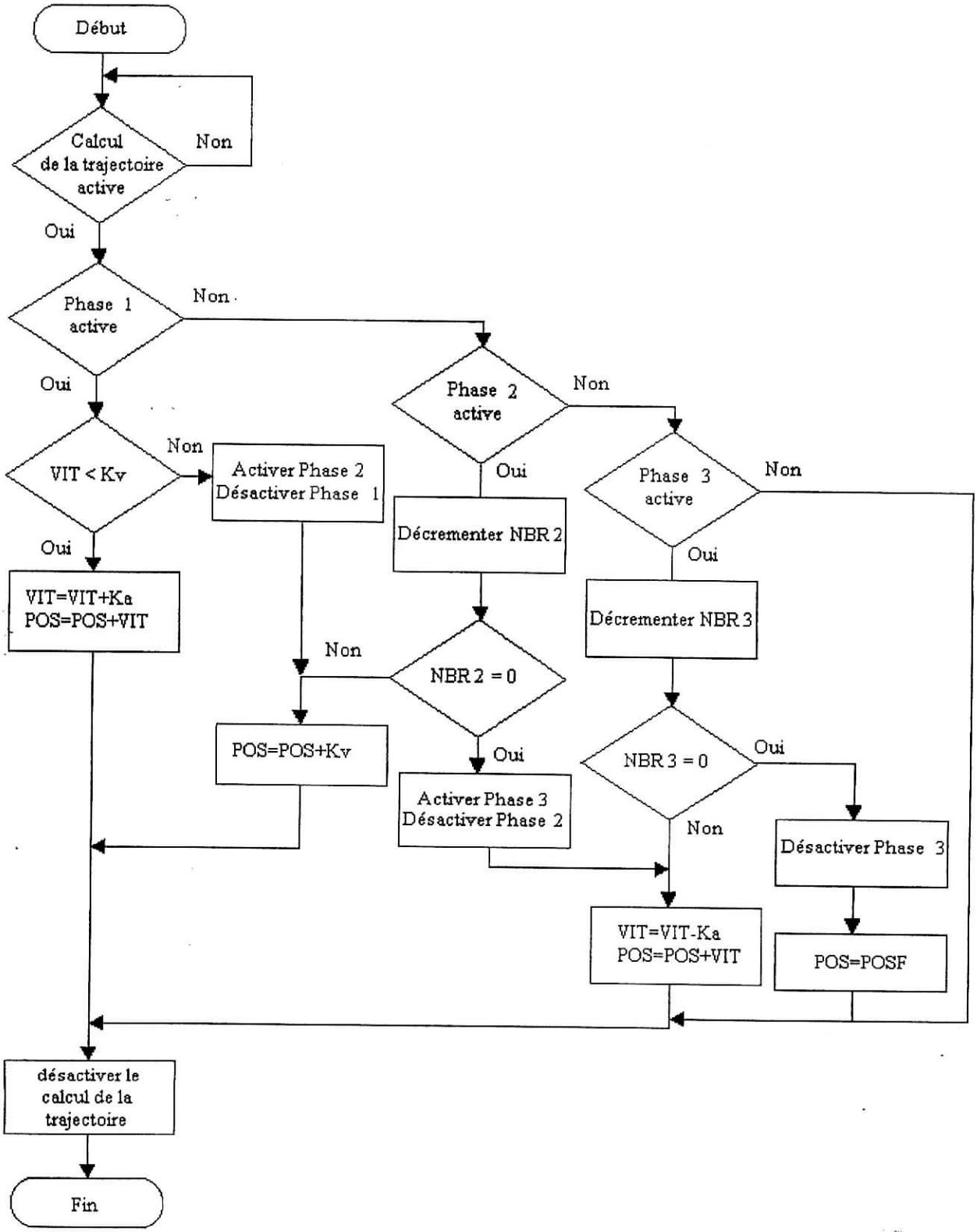


Figure 5.3. *Organigramme de génération de trajectoire.*

La génération de la trajectoire se fait en trois phase, la phase 1 est la phase d'accélération, la valeur de l'accélération Ka est additionné au registre de vitesse VIT, ce dernier est additionné à son tour au registre de position POS, cette phase dure jusqu'à ce que la vitesse soit supérieur ou égale à la vitesse maximale Kv, puis la phase 2 s'active, durant cette phase seule la valeur Kv est additionné au registre de position, une variable NBR2 correspondant au nombre d'échantillons que dure cette phase se décrémente,, arrivée NBR2 à zéro la phase 3 qui est la phase de décélération s'active, la valeur Ka est soustraite au registre de vitesse puis ce dernier est additionné au registre de position, une variable NBR3 se décrémente, elle correspond au nombre d'échantillons que dure cette phase, arrivé NBR3 à zéro, la valeur de la position final POSF est charger dans le registre de position.

La position désirée est calculée à chaque fois que le calcul de la trajectoire est activé **[11]**, cette portion de code calcule la prochaine position que le moteur doit occuper. Après calculs la position obtenue a une taille de 32 bits.

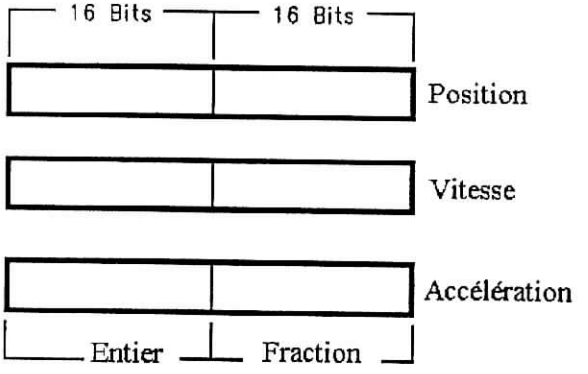


Figure 5.4. *Représentation du format des variables de consigne.*

5.5. Implémentation de l'algorithme de régulation

la position réelle et désirée du moteur connues, l'algorithme de régulation est exécuté pour calculer la prochaine valeur du rapport cyclique. L'algorithme implémenté est un régulateur proportionnel- intégral- dérivé (PID).

. Le terme proportionnel fournie une action directement liée à l'erreur, le terme intégral accumule les erreurs en position successives calculée à chaque itération de la boucle de réglage et le terme dérivée correspond à une action qui est liée au variation de l'erreur en position.

A l'instant k, on donne **[3]**:

$$U = K_p.e[k] + K_i.\sum_{i=0}^{k} e[i] + K_d.(e[k] - e[k-1]) \qquad (5.1)$$

cette expression représente la loi de commande d'un régulateur classique PID, l'erreur en position est en radians et le signal de commande U est en volts, donc il est nécessaire de transformer cette loi de commande afin quelle soit adaptée à notre calculateur, nous savons que la position du moteur est fournie en nombre d'incréments et la tension U doit être une valeur entière qui représente le rapport cyclique, donc nous obtenons :

$$e[k] = \Delta q\, e_{INC}[k] \qquad (5.2)$$

où Δq représente le pas de quantification et $e_{INC}[k]$ l'erreur en nombre d'incrément.

$$U = \alpha E \Rightarrow U = \frac{t_f}{T} E \tag{5.3}$$

T : temps de découpage t_f : temps de fermeture

A partir de la relation (5.3) nous obtenons :

$$t_f = \frac{T}{E} U \tag{5.4}$$

nous savons du datasheet du microcontrôlleur que la valeur du registre PR2 correspond au temps de découpage et le registre CCPR1L correspond au temps de fermeture, la relation (5.4) devient :

$$CCPR1L = \frac{PR2}{E} U \tag{5.6}$$

A partir des relations (5.1), (5.2) et (5.6) nous obtenons une nouvelle expression de la loi de commande qui est :

$$CCPR1L = k_p . e_{INC}[k] + k_i . \sum_{i=0}^{k} e_{INC}[i] + k_d . (e_{INC}[k] - e_{INC}[k-1]) \tag{5.7}$$

avec :

$$k_j = K_j \frac{PR2 \ \Delta q}{E} \qquad j = p, i, d \tag{5.8}$$

la figure suivante donne l'organigramme de l'algorithme de réglage :

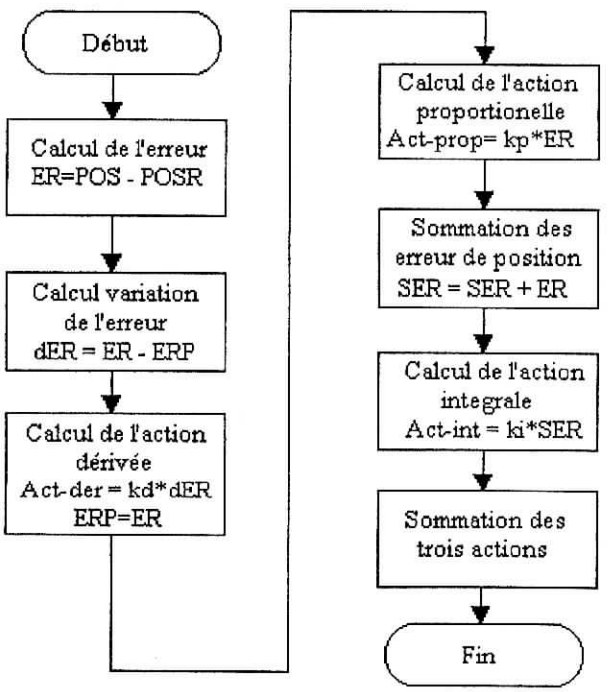


Figure 5.5. *Organigramme de l'algorithme de régulation PID.*

L'organigramme de l'algorithme de réglage PID est linéaire, au début l'erreur de position ER est calculée, ensuite la variation de l'erreur qui représente la différence entre l'erreur en position actuelle ER et l'erreur précédente ERP est calculée à son tour, a partir de cette variation la valeur du terme différentiel (Act-der) est déterminée.

Le terme proportionnel (Act-prop) est ensuite calculé, puis nous terminons par le calcul du terme intégral (Act-int) en sommant les erreur de position dans une variable SER et en le multipliant par le coefficient correspondant.

Les trois termes qui ont une taille de 32 bits sont additionnés ensemble, seul les 8 bits les plus significatifs des 16 bits les moins significatifs du résultat de l'addition est pris en compte car il représente la valeur en temps du signal de commande.

5.6. Génération du signal de commande du moteur (PWM)

la commande ayant été calculée, nous obtenons une valeur correspondante au temps de fermeture du circuit de commutation, la valeur de la commande peut prendre des valeurs positive ou des valeurs négative, mais comme nous utilisons une seule PWM, il est donc nécessaire de trouver une manière de faire fonctionner le moteur dans les deux sens, pour cela nous avons rajouter un circuit qui utilise un bit de signe pour inverser la tension de commande (voir figure A.3.2).

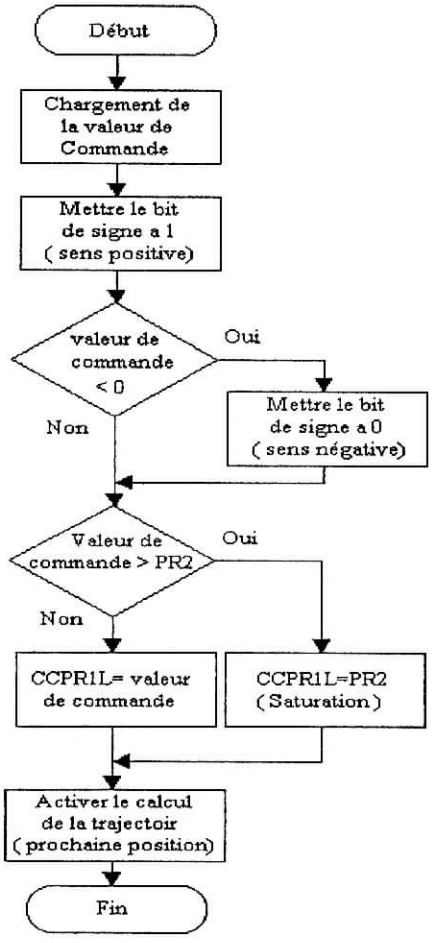


Figure 5.3. Organigramme de génération du signal de commande.

L'organigramme précédent nous permet de tester le signe de la valeur de commande, il a pour action de mettre à 1 ou à 0 le bit de signe pour inverser la tension de commande, une saturation de la valeur de commande est envoyée au registre CCPR1L de la PWM dans le cas où la valeur de commande calculé est supérieur à PR2.
Après l'envois de la valeur de commande dans le registre correspondant, nous activons le calcul de la trajectoire afin de déterminer la prochaine position désirée du moteur.

5.7. Calibrage des coefficients

La figure 5.6 donne le détail sur la multiplication pour les coefficients k_p, k_d et k_i [11], ces coefficients étant réels, ils sont représenté sous un format de 16 bits, les 8 bits les moins significatifs représentent la partie fractionnaire, donc afin de les représenter sous le format voulu, ils sont multipliés par 256 et seule la partie entière du résultat est convertie pour former les coefficients à implémenter dans le calculateur.

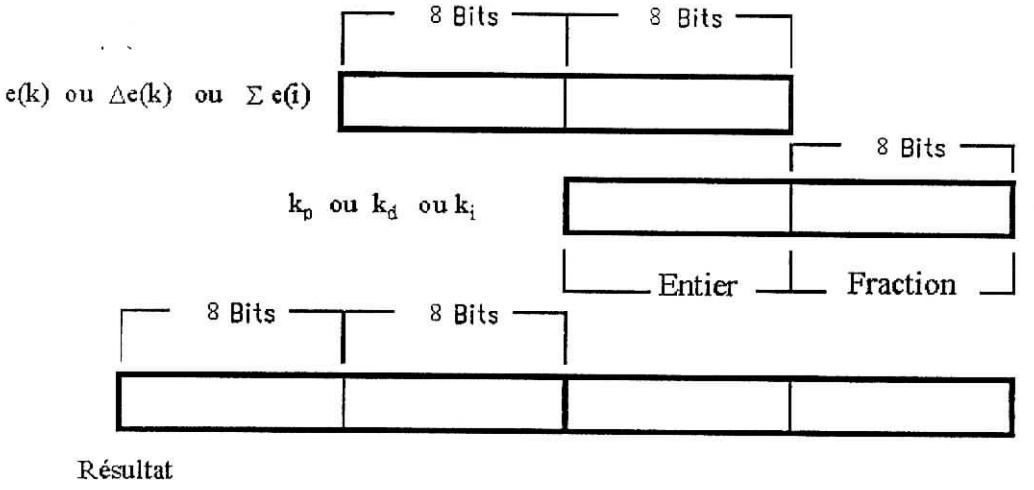


Figure 5.6. *Calibrage des coefficients* .

5.8. Réalisation

nous avons réalisé le circuit prévu à la commande en position d'un moteur à courant continue, le schéma de est représenté sur la figure 5.7 principe et le circuit imprimé est donnée en annexe 5 (figure A5.10)

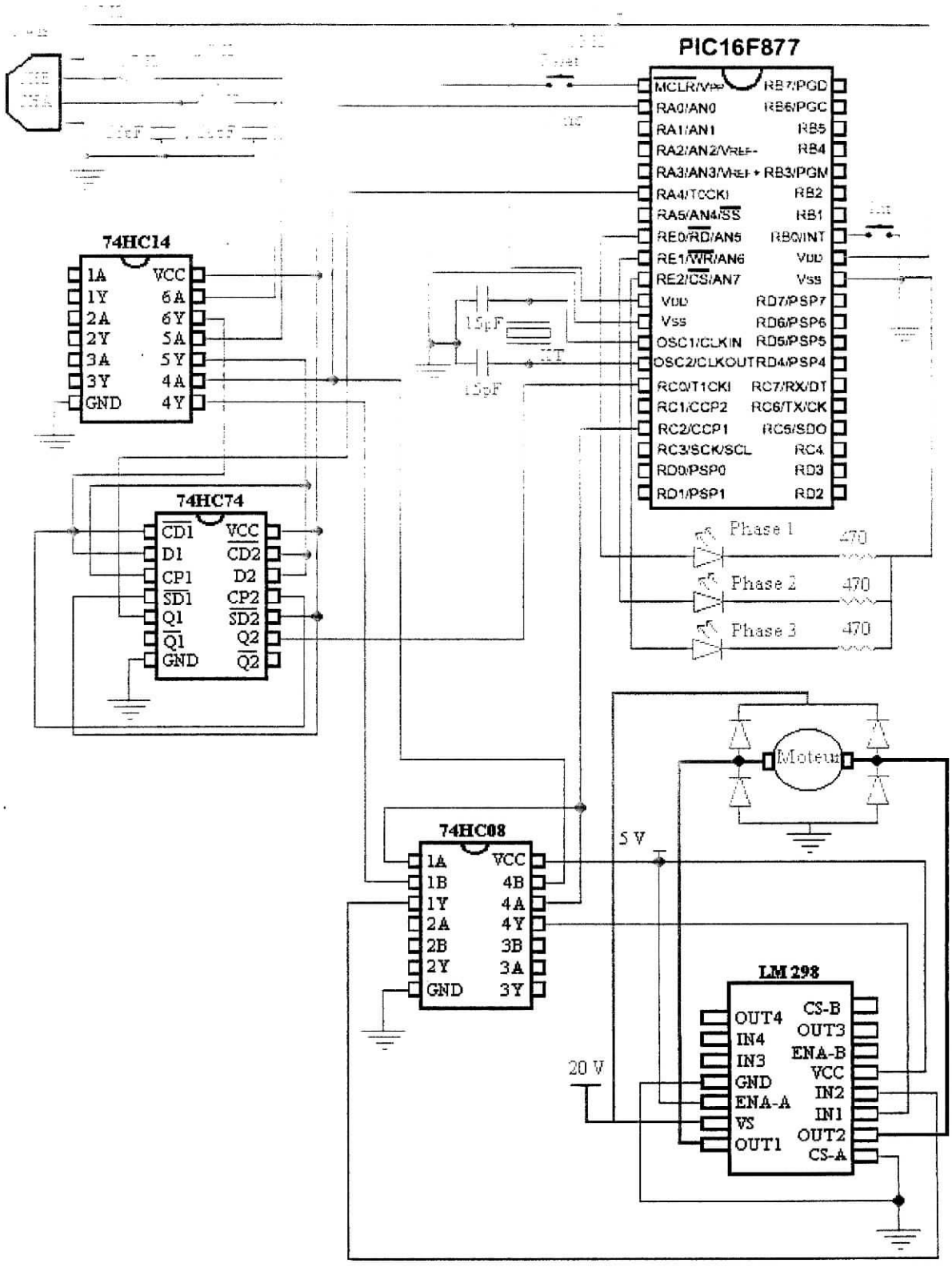


Figure 5.7. *Schéma électrique*

5.9. Résultats d'application

Dans cette partie, nous avons calculer les coefficients du régulateur PID à implémenter dans le calculateur pour faire la poursuite en position du moteur, pour notre application nous avons choisit un régulateur PD, ainsi les coefficients obtenus dans le domaine discret sont :

$K_p = 21$ et $K_d = 91$

En utilisant la relation (5.8), nous obtenons les coefficients du régulateur à implémenter dans le microcontrôlleur sont les suivants :

$k_p = 3.3778$ et $k_d = 14.6373$

ces coefficients représentés sous un format de 16 bits deviennent en binaires, comme suit :

k_p = B'00000011 01100000'

k_d = B'00001110 10100011'

La période d'échantillonnage a été choisie égale à 0.8192 ms ce qui correspond à une valeur du registre PR2 égale à 256 et un post diviseur de 16.

La fréquence du signal PWM qui commande le moteur doit être assez élevée afin que la composante ondulé du courant induit dans le bobinage soit faible, le microcontrôlleur fonctionne avec une fréquence d'horloge de 20 Mhz et le signal PWM avec une fréquence égale à 19.53 Khz ce qui correspond à une interruption du Timer2 chaque 51.2 µs.

Pour la démonstration nous avons pris une consigne de trajectoire de +150° ce qui correspond aux paramètres suivants :

$Ka = 1000$ rad/sec^2 , $Kv = 292.8127$ rad/sec , $\tau = 0.2928$ sec , $t_{final} = 0.8784$ sec

A partir de ces paramètres nous obtenons :

Ka = 0.0534 incréments/échantillon^2 , Kv = 19.0884 incréments/échantillon

NBR2 = NBR3 = 357 échantillons

POSF = 13646 incréments.

5.10. Conclusion

Dans ce qui a été fait précédemment, nous avons montré comment implémenter la commande en position d'un moteur a courant continue en intégrant une routine de génération de trajectoire de type trapézoïdale ainsi qu'un algorithme de régulation PID générant la commande sous forme d'une PWM.

CONCLUSION GENERALE

Nous avons eu, dans notre travail, à faire l'étude du robot série *SCARA*. Comme pour tous robot manipulateur, nous avons effectué une modélisation du point de vue géométrique en se basant sur la notation de Denavit-Hartenberg, une modélisation cinématique et une modélisation dynamique, qui elle est fondée sur le formalisme de *Lagrange*. L'étude structurelle et dimensionnelle étant essentielle, nous nous sommes penchés sur le calcul des modes propres du robot en faisant certaine approximation conceptuelle afin que les commandes appliquées sur ce robot soient performantes. Nous avons appliqué deux types de commande : la commande par découplage non linéaire et la commande classique (PID). Les résultats, obtenues en simulation, ont montré la robustesse de celle ci devant les variations d'inertie dues au robot lui même et aux charges. Ces simulation ont été faites en tenant compte de la dynamique des actionneurs qui sont des moteurs à courant continu.

Une étude dans le domaine discret a été réalisée sur le système de commande afin de pouvoir implémenter les algorithmes de réglage. Nous avons observé, en simulation, l'influence de l'échantillonnage et de la quantification et nous avons proposé des solutions pour palier cette influence. Toutefois, nous avons réalisé et conçu une carte de commande à base du microcontrôlleur 16F877 qui effectue l'asservissement en position d'un moteur à courant continue. Cette réalisation, destinée à la commande du robot *Scara*, nous a permis de comprendre et de résoudre les problèmes liés à la pratique, tels que la synchronisation , l'élaboration de programme et l'implémentation d'algorithme.

A la lumière des résultats obtenus, de nombreuses perspectives s'ouvrent à nous. En collaboration avec le département de Génie mécanique, la conception de la structure du robot *Scara* ainsi que l'étude liée à l'aspect mécanique et structurelle sont à envisager. De même, la réalisation de plusieurs cartes de commande interfacées par ordinateur est à prendre en considération.

REFERENCES BIBLIOGRAPHIQUES

REFERENCES BIBLIOGRAPHIQUES

[1] **W. Khalil, E. Dombre** ; « Modélisation, identification et commande des robot »
© *HERMES Sciences Publications*, Paris 1988, 1999.

[2] **H. Bühler** ; « Conception des systèmes automatiques »
Presse Polytechnique Romandes, 1988.

[3] **H. Bühler** ; « Réglages échantillonnés » – Volume 1.
Presse Polytechnique Romandes, 1986.

[4] **L. Badji, F. Bouziani** ; « Etude et mise en fonction de l'interface d'axes industriels Turbo UMAC »
E.N.P, Département du génie électrique, Projet de fin d'études, 2002.

[5] **K. Djabella, B.Lassami** ; « Dévellopement d'un logiciel d'animation et de commande pour bras manipulateurs »
E.N.P, Département du génie électrique, Projet de fin d'études, 2002.

[6] **A. Berni, L. Bentmohamed, A. Ouali** ; « Conception d'un robot mobile non-holonome »
U.S.T.H.B, Institut d'électronique, Projet de fin d'études, 2000.

[7] **C. Bigonoff** ; « La programmation des PICs par Bigonoff – seconde partie – la gamme Midrange par l'étude des 16F87X (16F876-16F877) », format pdf
http://www.abcelectronique.com/bigonoff/ , 2003.

[8] « PIC 16F87x datasheet 28/40-Pin 8-bits CMOS Flash Microcontrolers », DS30292C, Microchip Technology. Inc, 2002.
http://microchip.com/

[9] **S. Boeling** ; « Pic 18CXXX / Pic16CXXX DC servomotor application », DS006961, Microchip Technology. Inc Chandler, AZ, 2003.
http://microchip.com/

[10] **Louis Lamarche**; « Notes de cours – MEC-741 »
Département de génie mécanique, Rédigé: 1997/08 – Révisé: 2002/08
www.mec.etsmtl.ca/cours/mec741/notes.pdf

[11] « LM628/629 User Guide »
National Semiconductor, Application Note 706, October 1993
http://www.national.com/an/AN/AN-706.pdf

[12] **P.Mayé** ; « Moteur électrique pour la robotique »
© Edition DUNOD, Paris, 2000

[13] **M. Correvon** ; « Systèmes électronique – Chapitre 5 – Les régulateur standards ».
http://iese.eivd.ch/Enseignement/enseignement.html, 2002

[14] « Datasheet L298 - DUAL FULL-BRIDGE DRIVER », Format pdf
www.hvwtech.com/downloads/datasheets/L298.pdf

[15] **S. Bourin** ; « Régulation de courant pour la commande des moteurs DC »
http://iai1.eivd.ch/cours/cours_er/chap_03/html/node2.htm , 2003

[16] « Echantillonnage et quantification »
http://www.atela.uhp-nancy.fr/tisserand/sma/HTML/experience.htm , 2003

[17] **B.CHERKI** ; « Commande des robots manipulateurs par retour d'état »
Thèse doctorat, Ecole Centrale de Nantes, 1996.

[18] Datasheet « Fasr Fairchild Advanced Schottky TTL »
Fairchild, Schlumberger Company

[19] Technical Data HEDS9100 « Three Channel Optical Incremental Encoder Modules »
© Agilent Technologies, Inc. 2002.
www.agilent.com/semiconductors

ANNEXE 1

MODELISATION
DYNAMIQUE

A1. Modélisation dynamique

A1.1. Introduction

Le model des bras manipulateur sont décrits par un système d'équation mathématique qui décrivent sa dynamique, Les robots étant des structures mécanique, l'ensembles des équations dynamique peuvent être déterminé par les lois de la mécanique classique ; le modèle dynamique décrit la relation entre les couples (et/ou forces) appliqués aux actionneurs et les position, vitesse et accélération articulaires. On représente le modèle dynamique par une relation de la forme **[1]**, **[17]**:

$$\Gamma = f(q, \dot{q}, \ddot{q}, f_e) \qquad \text{(A1.1)}$$

- Γ : vecteur des couples/forces des actionneurs, selon que les articulation est rotoïde ou prismatique. Dans la suite, on écrira tout simplement *couples* ;

- q : vecteur des positions articulaire ;

- $\dot{q}$: vecteur des vitesses articulaires ;

- $\ddot{q}$: vecteur des accélération articulaires ;

- f_e : vecteur représentant l'effort extérieur (forces et moments) qu'exerce le robot sur l'environnement.

On convient d'appeler *modèle dynamique inverse*, ou tout simplement *modèle dynamique*, la relation de la forme (A1.1).

Plusieurs formalisme ont été utilisés pour obtenir le modèle dynamique des robots, les formalismes les plus utilisés sont le formalisme de *Lagrange* et le formalisme de *Newton-Euler* qui ferons l'objet de ce chapitre.

A1.2 Formalisme de Lagrange

Le but de ce paragraphe est d'étudier la forme générale du modèle dynamique, de mettre en évidence les différents termes qui y interviennent **[1]**.

Le formalisme de Lagrange décrit les équations du mouvement en termes de travail et énergie du système ; ce qui se traduit par l'équation suivante :

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad i = 1, \ldots, n \qquad \text{(A1.2)}$$

- L : lagrangien du système égale à L=E-U ;

- E : énerigie cinétique totale du système ;

- U : énerigie potentielle totale du système.

A1.2.1. Forme générale des équation dynamique

L'équation dynamique peut se mettre en générale sous la forme :

$$\Gamma = A(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + Q(q) + \Gamma_f + \Gamma_e + T_{m0} \qquad \text{(A1.3)}$$

où :

A(q) est la matrice (n x n) de l'énergie cinétique, appelée aussi *matrice d'inertie* du robot; $C(q,\dot{q})\,\dot{q}$ vecteur de dimension (n x 1)représentant les couples/forces de Coriolis et des forces centrifuges ; **Q(q)** vecteur des forces/couples de gravité.

A1.2.2. Calcul de l'énergie cinétique

l'énergie cinétique du système est une fonction quadratique des vitesses articulaires :

$$E = \frac{1}{2}\dot{q}^T A(q)\,\dot{q} \qquad \text{(A1.4)}$$

l'énergie cinétique totale du système est donnée par la relation :

$$E = \sum_{j=1}^{n} E_j \qquad \text{(A1.5)}$$

où E_j désigne l'énergie cinétique du corps C_j, qui s'exprime par :

$$E_j = \frac{1}{2}(\omega_j^T\, I_{Gj}\, \omega_j + M_j V_{Gj}^T V_{Gj}) \qquad \text{(A1.6)}$$

Etant donnée que (figure 1.12) :

$$V_{Gj} = V_j + \omega_j \times S_j \qquad \text{(A1.7)}$$

et en sachant que :

$$J_j = I_{Gj} - M_j \hat{S}_j \hat{S}_j \qquad \text{(A1.8)}$$

la relation (A1.6) devient :

$$E_j = \frac{1}{2}[\omega_j^T\, J_j\, \omega_j + M_j V_j^T V_j + 2 * MS_j^T (V_j \times \omega_j)] \qquad \text{(A1.9)}$$

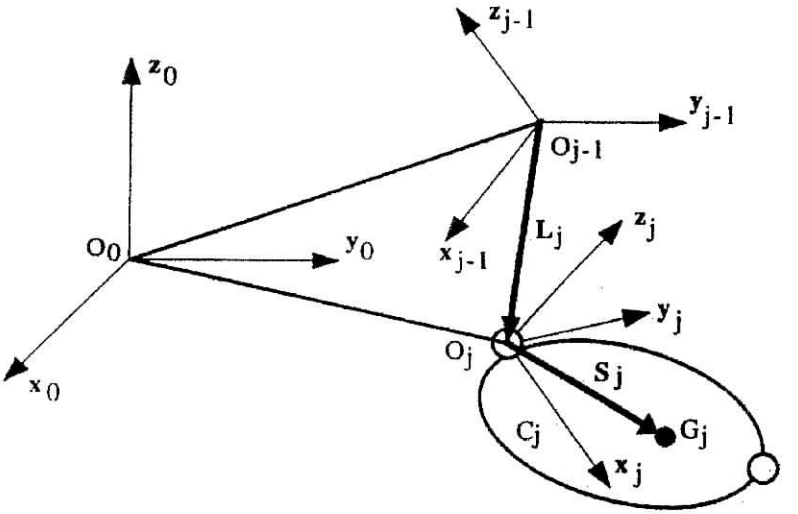


Figure A1.1 :*Composition des vitesses.*

Le calcul de $\mathbf{V}_j$ et de ω_j se fait par les équation de composition des vitesses(figure A1.1).

$$\omega_j = \omega_{j-1} + \bar{\sigma}_j \dot{q}_j a_j \qquad \text{(A1.10)}$$

$$V_j = V_{j-1} + \omega_{j-1} \times L_j + \sigma_j \dot{q}_j a_j \qquad \text{(A1.11)}$$

Dans l'équation (A1.9), tous les éléments doivent être exprimés dans le même repère . la façon la plus simple est de l'exprimer dans le repère R_j . On réécrit donc les équations (A1.9), (A1.10) et (A1.11) donnant E_j, ${}^j\omega_j$ et ${}^j\mathbf{V}_j$ comme suit :

$$E_j = \frac{1}{2}[{}^j\omega_j{}^T \; {}^jJ_j \; {}^j\omega_j + M_j \, {}^jV_j{}^T \, {}^jV_j + 2 * {}^jMS_j{}^T ({}^jV_j \times {}^j\omega_j)] \qquad \text{(A1.12)}$$

$${}^j\omega_j = {}^jA_{j-1} \; {}^{j-1}\omega_{j-1} + \bar{\sigma}_j \dot{q}_j \; {}^ja_j \qquad \text{(A1.13)}$$

$${}^jV_j = {}^jA_{j-1}({}^{j-1}V_{j-1} + {}^{j-1}\omega_{j-1} \times {}^{j-1}P_j) + \sigma_j \dot{q}_j \; {}^ja_j \qquad \text{(A1.14)}$$

- La matrice **A** est une matrice d'élément générique A_{ij}. L'élément A_{ii} est égal au coefficient de $(q_i^2/2)$ dans l'expression de l'énergie cinétique, tandis que l'élément A_{ij} si $i \neq j$, est égale au coefficient de $q_i \, q_j$.

- Les éléments de la matrice **C** peuvent être calculés à partir du *symbole de Christophell* $c_{i,jk}$ tel que :

$$C_{ij} = \sum_{k=1}^{n} c_{i,jk} \dot{q}_k \quad \text{avec} \quad c_{i,jk} = \frac{1}{2}\left[\frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} - \frac{\partial A_{jk}}{\partial q_i}\right] \qquad \text{(A1.15)}$$

A1.2.3. Calcul de l'énergie potentielle

L'énergie potentielle s'écrit :

$$U = \sum_{K=1}^{n} U_j = \sum_{K=1}^{n} - M_j \, g^T (L_{0,j} + S_j) \qquad \text{(A1.16)}$$

$\mathbf{L}_{0,j}$ désignant le vecteur d'origine $\mathbf{O}_0$ et d'extrémité $\mathbf{O}_j$ .En projetant les vecteurs de cette relation dans $\mathbf{R}_0$,on obtient :

$$U_j = -M_j \; {}^0g^T ({}^0P_j + {}^0A_j \, {}^jS_j) \qquad \text{(A1.17)}$$

L'expression précédente peut se mettre sous la forme

$$U_j = - \; {}^0g^T (M_j \, {}^0P_j + {}^0A_j \, {}^jMS_j) = -[{}^0g^T \quad 0] \; {}^0T_j \begin{bmatrix} {}^jMS_j \\ M_j \end{bmatrix} \qquad \text{(A1.18)}$$

- les éléments de $\mathbf{Q}$ tel que $\mathbf{Q}=[Q_1 \ldots Q_n]^T$ se calculent en écrivant que :

$$Q_i = \frac{\partial U}{\partial q_i} \qquad \text{(A1.19)}$$

A1.2.4 Couples de frottements et des efforts statique

Le vecteur Γ_f représente les forces/couples de frottement, il est donné par la relation suivante :

$$\Gamma_f = \mathrm{Diag}(\dot{q}) \, F_v + \mathrm{Diag}[\mathrm{Sign}(\dot{q})] \, F_s \qquad \text{(A1.20)}$$

avec :

$$F_s = [F_{s1} \ \ldots \ F_{sn}]^T$$
$$F_v = [F_{v1} \ \ldots \ F_{vn}]^T$$

Le vecteur Γ_e représente les couples ou forces que doivent fournir les actionneurs d'un robot pour que son organe terminal puisse exercer un effort statique f_e sur l'environnement ; il est donné par la relation suivante :

$$\Gamma_e = J_n^{\,T} \, f_e \qquad \text{(A1.21)}$$

A1.2.5. Prise en compte des inerties des actionneurs

Le plus souvent il est nécessaire de tenir compte des inerties des actionneurs. En représente l'énergie cinétique de l'actionneur j par un terme de la forme :

$$\frac{1}{2} Ia_j \dot{q}_j^2$$

Le paramètre inertiel Ia_j peut s'écrire :

$$Ia_j = N_j^2 J_{mj} \qquad \text{(A1.22)}$$

Où J_{mj} est le moment d'inertie du rotor de l'actionneur j, N_j est le rapport de réduction de l'axe j égal à $\dot{q}_{mj} / \dot{q}_j$ et $\dot{q}_{mj}$ désigne la vitesse du rotor de l'actionneur j.

A1.2.6. Prise en compte des couples de charge

Le vecteur des couples additifs T_{m0} représente l'effet de la charge **[5]**, il set calculé à partir de la matrice jacobienne J, ce couple est donné par la relation suivante :

$$T_{m0} = m_0 J^T(q)[J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} + g] \qquad \text{(A1.23)}$$

A.1.3 Formalisme de Newton-Euler

Les équation de Newton –Euler expriment le torseur dynamique en Gj des efforts extérieur sur un corps j par les équations :

$$F_j = M_j \dot{V}_{Gj} \qquad \text{(A1.24)}$$

$$M_{Gj} = I_{Gj} \dot{\omega}_j + \omega_j \times (I_{Gj} \omega_j) \qquad \text{(A1.25)}$$

Cette méthode permet de calculer le modèle dynamique des robots en ligne, elle est fondé sur une double récurrence. La récurrence avant, de base du robot à vers l'effecteur , calcule successivement les vitesses et accélérations des corps, puis leur torseur dynamique. Une récurrence arrière de l'effecteur vers la base , permet le calcul des couples des actionneurs en exprimant pour chaque corps le bilans des efforts.

Cette méthode permet d'obtenir directement le modèle dynamique inverse sans avoir à calculer explicitement les matrices **A**,**C** et **Q**. Les paramètres inertiels utilisés sont $\mathbf{M}_j$, $\mathbf{S}_j$ et $\mathbf{I}_{Gj}$. Le modèle obtenue n'est pas linéaire par rapport aux paramètres inertiels.

A1.3.1. Equations de Newton_Euler linéaires par rapport aux paramètres inertiels

Dans ce paragraphe il est présenté un algorithme de Newton-Euler fondé sur la double récurrence de la méthode de **Luh** et **al [1],** mais expriment le torseur dynamique des effort extérieurs en O_j plutôt en G_j , en utilisant les paramètres inertiels $\mathbf{M}_j$, $\mathbf{MS}_j$ et $\mathbf{J}_j$ .Le modèle ainsi obtenu est linéaire par rapport aux paramètres inertiels.

Les équations de Newton-Euler ainsi modifiées s'écrivent :

$$F_j = M_j \dot{V}_j + \dot{\omega}_j \times MS_j + \omega_j \times (\omega_j \times MS_j) \quad \text{(A1.26)}$$

$$M_j = J_j \dot{\omega}_j + \omega_j \times (J_j \omega_j) + MS_j \times \dot{V}_j \quad \text{(A1.27)}$$

A1.3.1.1 récurrence avant

Elle permet de calculer $\mathbf{F}_j$ et $\mathbf{M}_j$ à partir des relations (A1.69) et (A1.70). Pour cela il faut calculer ω_j , $\dot{\omega}_j$ et $\dot{V}_j$. Les formules de composition des vitesses donnent :

$$\omega_j = \omega_{j-1} + \bar{\sigma}_j \dot{q}_j a_j \quad \text{(A1.28)}$$

$$V_j = V_{j-1} + \omega_{j-1} \times L_j + \sigma_j \dot{q}_j a_j \quad \text{(A1.29)}$$

La dérivée de ces équations par rapport au temps s'écrit :

$$\dot{\omega}_j = \dot{\omega}_{j-1} + \bar{\sigma}_j (\ddot{q}_j a_j + \omega_{j-1} \times \dot{q}_j a_j) \quad \text{(A1.30)}$$

$$\dot{V}_j = \dot{V}_{j-1} + \dot{\omega}_{j-1} \times L_j + \omega_{j-1} \times (\omega_{j-1} \times L_j) + \sigma_j (\ddot{q}_j a_j + 2 \omega_{j-1} \times \dot{q}_j a_j) \quad \text{(A1.31)}$$

On calcule finalement $\mathbf{F}_j$ et $\mathbf{M}_j$ grâce aux relation A1.30 et A1.31 en initialisant cette récurrence par $\omega_0 = 0$, $\dot{\omega}_0 = 0$ et $\dot{V}_0 = 0$.

A1.3.1.2 Récurrence arrière

Les équations composant la récurrence arrière sont obtenues à partir du bilan des efforts sur chaque corps , écrit à l'origine O_j , on obtient (figure A1.2) :

$$F_j = f_j - f_{j+1} + M_j g - f_{ej} \quad \text{(A1.32)}$$

$$M_j = m_j - m_{j+1} - L_{j+1} \times f_{j+1} + S_j \times M_j g - m_{ej} \quad \text{(A1.33)}$$

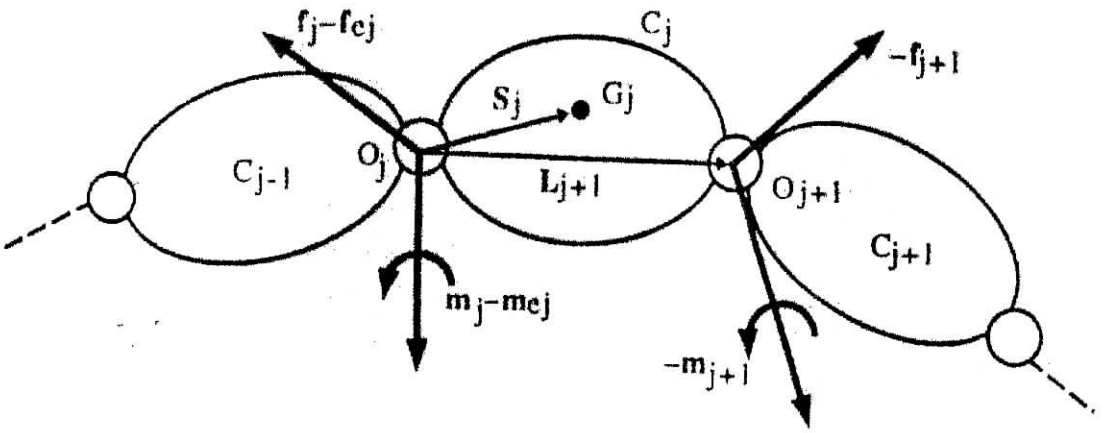


Figure A1.2. *Bilan des efforts au centre de gravité.*

On peut faire intervenir l'effet de la gravité sans avoir à la prendre en compte dans le bilan des effort , pour cela, on prend :

$$\dot{V}_0 = -g$$

d'où l'on tire les équations suivantes :

$$f_j = F_j + f_{j+1} + f_{ej} \qquad \text{(A1.34)}$$

$$m_j = M_j + m_{j+1} + L_{j+1} \times f_{j+1} + m_{ej} \qquad \text{(A1.35)}$$

Cette récurrence est initialisée par les effort. $f_{n+1} = 0$ et $m_{n+1} = 0$.

On obtient alors les couples Γ_j en projetant , suivant la nature de l'articulation j, les vecteur f_j ou m_j sur l'axe du mouvement. On ajoute les termes correctifs représentant l'effet des frottements et des inerties des actionneurs , ce qui donne :

$$\Gamma_j = (\sigma_j f_j + \bar{\sigma}_j m_j)^T a_j + F_{Sj} \, \mathrm{Sign}(\dot{q}_j) + F_{Vj} \dot{q}_j + Ia_j \ddot{q}_j \qquad \text{(A1.36)}$$

On déduit directement des équation (A1.31) et (A1.32) que les termes $\mathbf{f}_j$ et $\mathbf{m}_j$ ne dépendent que des paramètres inertiels du corps j et des corps situés en aval qui sont introduit par les termes $\mathbf{f}_{j+1}$ et $\mathbf{m}_{j+1}$ de la récurrence.

A1.3.2. Forme pratique des équations de Newton-Euler [1]

Pour utiliser pratiquement l'algorithme de Newton-Euler, il faut projeter dans un même repère les vecteurs et tenseurs qui apparaissent dans une même équation. Nous prenons ici le choix de Luh et al **[1]** qui consiste à projeter les grandeurs relatives à un corps dans le repère qui lui est lié.

Les équations de la récurrence avant deviennent, pour j=1,... n :

$$ {}^{j}\omega_{j-1} = {}^{j}A_{j-1}\ {}^{j-1}\omega_{j-1} \tag{A1.37} $$

$$ {}^{j}\omega_{j} = {}^{j}\omega_{j-1} + \overline{\sigma}_{j}\,\dot{q}_{j}\ {}^{j}a_{j} \tag{A1.38} $$

$$ {}^{j}\dot{\omega}_{j} = {}^{j}A_{j-1}\ {}^{j-1}\dot{\omega}_{j-1} + \overline{\sigma}_{j}\,(\ddot{q}_{j}\ {}^{j}a_{j} + {}^{j}\omega_{j-1} \times \dot{q}_{j}\ {}^{j}a_{j}) \tag{A1.39} $$

$$ {}^{j}\dot{V}_{j} = {}^{j}A_{j-1}({}^{j-1}\dot{V}_{j-1} + {}^{j-1}U_{j-1}\ {}^{j-1}P_{j}) + \sigma_{j}(\ \ddot{q}_{j}\ {}^{j}a_{j} + 2\ {}^{j}\omega_{j-1} \times \dot{q}_{j}\ {}^{j}a_{j}) \tag{A1.40} $$

$$ {}^{j}F_{j} = M_{j}\ {}^{j}\dot{V}_{j} + {}^{j}U_{j} \times {}^{j}MS_{j} \tag{A1.41} $$

$$ {}^{j}M_{j} = {}^{j}J_{j}\ {}^{j}\dot{\omega}_{j} + {}^{j}\omega_{j} \times ({}^{j}J_{j}\ {}^{j}\omega_{j}) + {}^{j}MS_{j} \times {}^{j}\dot{V}_{j} \tag{A1.42} $$

avec $\omega_0 = 0$, $\dot{\omega}_0 = 0$ et $\dot{V}_0 = g$ .et :

$$ {}^{j}U_{j} = {}^{j}\hat{\dot{\omega}}_{j} + {}^{j}\hat{\omega}_{j}\ {}^{j}\hat{\omega}_{j} \tag{A1.43} $$

L'introduction de la matrice ${}^{j}U_{j}$, ainsi que l'utilisation de certain de ces éléments dans le calcul de ${}^{j}\mathbf{M}_{j}$,permet un gain de calcul.

Pour la récurrence arrière, lorsque j =n,... 1 :

$$ {}^{j}f_{j} = {}^{j}F_{j} + {}^{j}f_{j+1} + {}^{j}f_{ej} \tag{A1.44} $$

$$ {}^{j-1}f_{j} = {}^{j-1}A_{j}\ {}^{j}f_{j} \tag{A1.45} $$

$$ {}^{j}m_{j} = {}^{j}M_{j} + {}^{j}A_{j+1}\ {}^{j+1}m_{j+1} + {}^{j}P_{j+1} \times {}^{j}f_{j+1} + {}^{j}m_{ej} \tag{A1.46} $$

$$ \Gamma_{j} = (\sigma_{j}\ {}^{j}f_{j} + \overline{\sigma}_{j}\ {}^{j}m_{j})^{T}\ {}^{j}a_{j} + F_{sj}\ \mathrm{Sign}(\dot{q}_{j}) + F_{vj}\ \dot{q}_{j} + Ia_{j}\ \ddot{q}_{j} \tag{A1.47} $$

ANNEXE 2

TECHNIQUES DE COMMANDE

A2.1. Introduction

Les technique de commande des robot manipulateur sont nombreuses, leurs utilisation dépend essentiellement de la structure du robot(complexité) et des taches à effectuer ; La synthèse des différentes commande se base principalement sur le modèle dynamique du robot **[1]**, on peut citer :

- la commande classique de type PID décentralisé ;
- la commande par découplage non linéaire ;
- la commande passive ;
- la commande adaptative ;
- la commande à structure variable (modes glissants).

Dans ce chapitre nous allons traiter les deux premières commandes et feront l'objet de notre étude sur le robot Scara.

A2.2. Equation du mouvement

Afin de bien appréhender la problématique de la commande des robot-manipulateur, il est utile de rappeler les équation du modèle dynamique du robot vue au §1.5 dont la forme général pour un robot à n degré de liberté est la suivante :

$$\Gamma = A(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + Q(q) + \Gamma_f \qquad \text{(A2.1)}$$

Ou sous un forme plus compacte :

$$\Gamma = A(q)\,\ddot{q} + H(q,\dot{q}) \qquad \text{(A2.2)}$$

La synthèse de la commande consiste à calculer le couple transmis par l'actionneur à l'articulation j dont notre cas un moteur à courant continue, puis d calculer la tension u_j à appliquer au borne du moteur permettant de suivre la consigne désirée.

A2.3. Commande classique

Le modèle dynamique décrit un système de n équations différentielles du second ordre non linéaires et couplées **[1]**, n étant le nombre d'articulations. Pourtant, dans une commande classique, qui est celle de la plupart des robots industriels actuels, le mécanisme est considéré comme un système linéaire et chacune de ses articulations est asservie par une commande décentralisée de type PID à gains constants.

Leurs avantages est la facilité d'implémentation et le faible coût en calcul par contre, on constate que la réponse temporelle varie suivant la configuration du robot et des dépassement de consigne. Dans beaucoup d'application ces inconvénients ne représente pas un gros problème.

Une telle commande est représenté sur suivant :

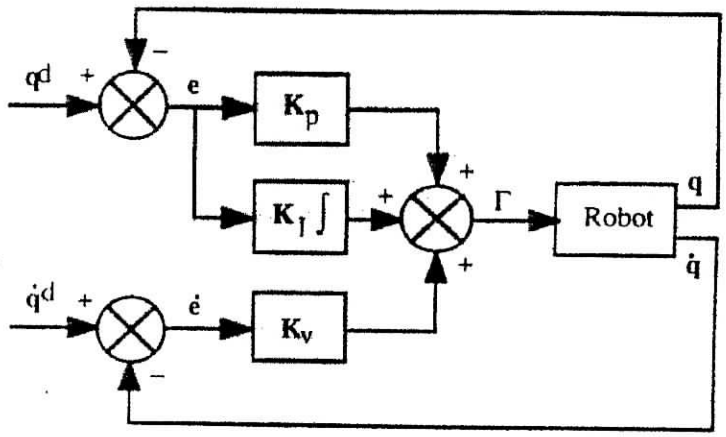


Figure A2.1. *Schéma classique d'une commande PID.*

La loi de commande est donnée par :

$$\Gamma = K_p(q^d - q) + K_v(\dot{q}^d - \dot{q}) + K_I \int_{t0}^{t} (q^d - q)\, d\tau \qquad \text{(A2.3)}$$

où $\dot{q}^d(t)$ et $q^d(t)$ représentes les vitesse et position désirées dans l'espace articulaire et $\mathbf{K_p}$, $\mathbf{K_v}$ et $\mathbf{K_I}$ sont des matrices diagonales d'élément K_{pj} , K_{vj} et K_{Ij} , le calcul de ces gains est effectué en considérant le modèle de l'articulation j comme un système linéaire du deuxième ordre à coefficients constants, elle est donnée par :

$$\Gamma = a_j \ddot{q}_j + F_{vj} \dot{q}_j + \gamma_j \qquad \text{(A2.4)}$$

Le coefficient $a_j = A_{jj\,max}$ désigne la valeur maximal de l'élément A_{jj} de la matrice d'inertie du robot et γ_j représente un couple perturbateur **[1]**.

La fonction de transfert en boucle fermée représente le transfert entre la position réelle $q(t)$ et la position désirée $q^d(t)$, elle est donnée par :

$$\frac{q_j(S)}{q^d{}_j(S)} = \frac{K_{vj} S^2 + K_{pj} S + K_{Ij}}{a_j S^3 + (K_{vj} + F_{vj}) S^2 + K_{pj} S + K_{Ij}} \qquad \text{(A2.5)}$$

La solution la plus courante en robotique est de choisir les gains de manière à obtenir les pôles en boucles fermé comme étant un pôle triple réel négatif, se qui donne une réponse la plus rapide possible sans oscillation.
Par conséquent, l'équation caractéristique est :

$$\Delta(S) = a_j (S + \omega_j)^3 \qquad \text{(A2.6)}$$

avec $\omega_j > 0$. Les expressions des gains des régulateurs sont donnée par :

$$\begin{cases} K_{pj} = 3 a_j \omega_j^2 \\ K_{vj} + F_{vj} = 3 a_j \omega_j \\ K_{Ij} = a_j \omega_j^3 \end{cases} \qquad \text{(A2.7)}$$

Pour se qui a été dit précédemment, nous faisons les remarque suivantes :

- ω_j est choisi le plus grand possible, mais il est important de ne pas la choisir supérieure à la pulsation de résonance ω_{jr} correspondante aux modes de vibration mécanique afin de ne pas déstabiliser le système, en général on prend $\omega_j = \omega_{jr}/2$.
- les performances d'une telle méthode sont d'autant plus acceptable que le rapport de réduction est important et que sont faibles.

Pour le robot Scara, il a été montré qu'une commande Proportionnelle Dérivée suffit et cette commande est asymptotiquement stable **[1]**, une telle commande est représenté par le schéma suivant :

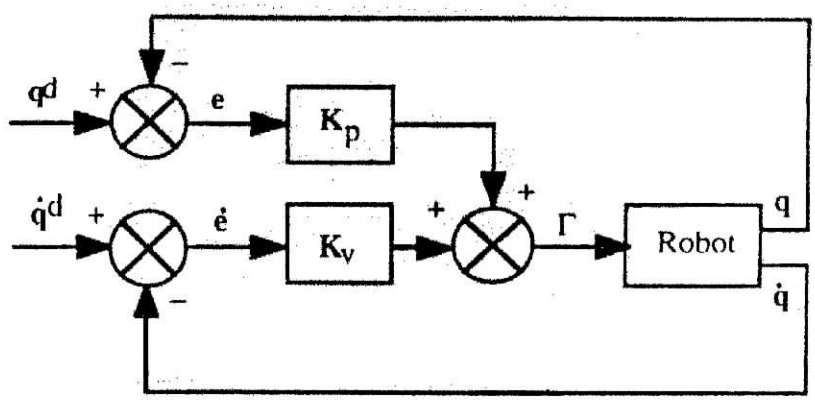


Figure A2.2. *Schéma classique d'une commande PD.*

La loi de commande est donnée par :

$$\Gamma = K_p (q^d - q) + K_v (\dot{q}^d - \dot{q}) \qquad \text{(A2.8)}$$

Dans ce cas la fonction de transfert en boucle fermée devient :

$$\frac{q_j(S)}{q^d{}_j(S)} = \frac{K_{vj}\,S + K_{pj}}{a_j\,S^2 + (K_{vj} + F_{vj})\,S + K_{pj}} \qquad \text{(A2.9)}$$

Par conséquent, l'équation caractéristique devient:

$$\Delta(S) = a_j (S + \omega_j)^2 \qquad \text{(A2.10)}$$

avec $\omega_j > 0$. Les expressions des gains des régulateurs sont donnée par :

$$\begin{cases} K_{pj} = a_j\,\omega_j{}^2 \\ K_{vj} + F_{vj} = 2\,a_j\,\omega_j \end{cases} \qquad \text{(A2.11)}$$

A2.4. Commande par découplage non linéaire

A2.4.1. Introduction

Lorsque l'application exige des évolutions rapides du robot et une grande précision dynamique, il est nécessaire de concevoir un système de commande plus sophistiqué, Ce type de commande est aussi connu sous le nom de *commande dynamique* ou *couple calculé*, parce qu'il est fondé sur l'utilisation du modèle dynamique. Théoriquement, il assure le découplage et la linéarisation des équations du modèle, ayant pour effet une réponse uniforme quelle que soit la configuration du robot **[1]**.

La mise en œuvre de cette méthode exige le calcul du modèle dynamique, elle consiste à transformer par retour d'état le problème de commande d'un système non linéaire en un problème de commande d'un système linéaire ; Dans le cas général, le problème de linéarisation par retour d'état d'un système non linéaire n'est pas facile à résoudre. Cependant, dans le cas des robots-manipulateurs rigides, l'élaboration d'une loi de commande qui linéarise et découple les équations est simplifiée par le fait que le nombre d'actionneurs est égal au nombre de variables articulaires et que le modèle dont on dispose est un modèle inverse qui exprime l'entrée Γ du système en fonction du vecteur d'état $(q, \dot{q})$ et de $\ddot{q}$.

La commande par découplage non linéaire peut être appliqué dans l'espace articulaire comme dans l'espace opérationnel, nous traiterons la commande dans l'espace articulaire seulement.

A2.4.2. Principe

le principe de la commande par découplage non linéaire est de trouver une commande qui linéarise le système, nous posons $x_1 = q$ et $x_2 = \dot{q}$; A partir de la relation (A2.2), le modèle d'état du robot s'écrit **[17]**:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = A^{-1}(x_1)(\Gamma - H(x_1, x_2)) \end{cases} \qquad \text{(A2.12)}$$

ou sous une forme plus compacte $\dot{x} = f(x) + g(x)\,u$

avec :

$$f(x) = \begin{bmatrix} x_2 \\ -A^{-1}(x_1)\,H(x_1, x_2) \end{bmatrix} \;,\; g(x) = \begin{bmatrix} 0 \\ A^{-1}(x_1) \end{bmatrix} \text{ et } u = \Gamma$$

cette représentation d'état nous servira pour la simulation. Si l'on choisit une commande Γ telle que :

$$\Gamma = A(q)\,w(t) + H(q, \dot{q}) \qquad \text{(A2.13)}$$

Dans le cas idéal où le modèle est supposé parfait, le système est régit par l'équation :

$$\ddot{q} = w(t) \qquad \text{(A2.14)}$$

w(t) est un nouveau vecteur de commande, on se ramène donc à un système de n équation linéaire, invariant, découplé et du second ordre. Plusieurs type de commande peuvent être adoptés comme une commande PD ou une commande PID (figure A2.3).

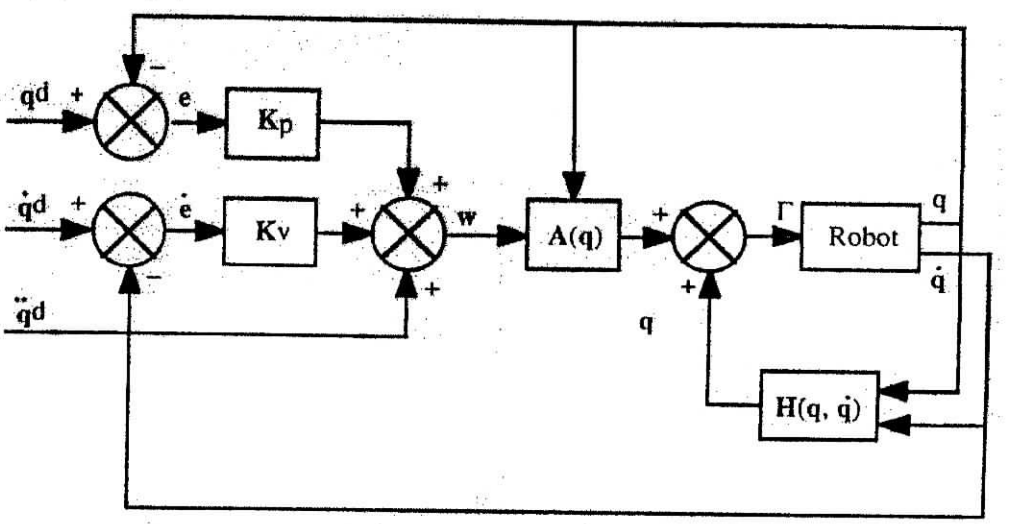


Figure A2.3. *Loi commande PD par découplage non linéaire.*

On désigne respectivement par $\ddot{q}^d(t)$, $\dot{q}^d(t)$ et $q^d(t)$ l'accélération, la vitesse et la position désirées dans l'espace articulaire ; Pour une commande Proportionnel dérivée, on calcul w(t) selon la relation suivant :

$$w(t) = \ddot{q}^d + K_v\,(\dot{q}^d - \dot{q}) + K_p\,(q^d - q) \qquad \text{(A2.15)}$$

où $\mathbf{K_p}$ et $\mathbf{K_v}$ sont des matrices diagonales définies positifs, d'après l'équation (A2.14), la dynamique en boucle fermée est décrite par l'équation linéaire découplée suivante :

$$\ddot{e} + K_v \dot{e} + K_p e = 0 \qquad \text{(A2.16)}$$

où $e = q^d - q$.

la solution de l'équation d'erreur e(t) est globalement exponentiellement stable. Les gains K_{pj} et K_{vj} sont choisis pour imposer à l'erreur de l'axe j une dynamique désirée d'amortissement ξ_j et une pulsation ω_j quelque soit la configuration du robot, nous avons :

$$\begin{cases} K_{pj} = \omega_j^2 \\ K_{vj} = 2\xi_j \omega_j \end{cases} \qquad \text{(A2.17)}$$

généralement on prend un amortissement $\xi_j = 1$ afin d'avoir une réponse sans dépassement.

La mise en œuvre de cette méthode exige le calcul du modèle dynamique en ligne et la connaissance des valeurs numérique des paramètres inertiels et des frottements, pour cette raison les matrices A et H sont estimées, l'algorithme de Newton-Euler cité au §1.5, se prête bien au calcul numérique [17], ainsi nous obtenons $\hat{A}$ et $\hat{H}$ les estimées de A et H, a partir de la commande Γ est calculée, le schéma bloc de cette commande est représentée à la figure A2.4.

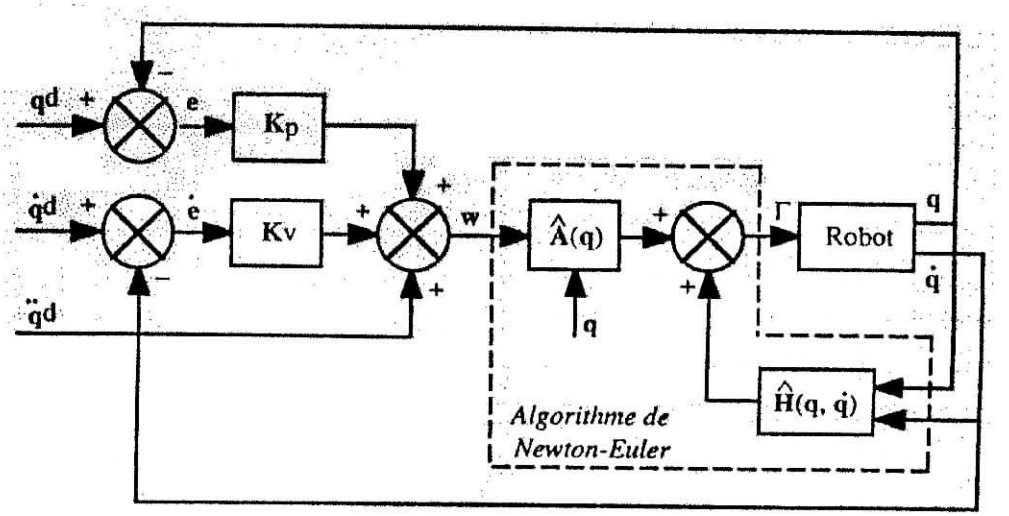


Figure A2.4. *Loi commande PD par découplage non linéaire avec estimée de A et H.*

A2.5 Conclusion

Dans ce chapitre nous nous sommes penchés sur les lois de commande des robots manipulateurs, plusieurs méthodes existent pour la synthèse de ces lois mais nous avons traité deux d'entre elles, la commande classique très utilisée donne certaines performances mais reste limitées et cela suivant le modèle du robot, puis nous avons la commande par découplage non linéaire qui est elle aussi très utilisée et donne de très bons résultats mais demande un coup de calcul plus important, dans la partie simulation nous allons appliquer ces deux commandes et observer les performances qui en résultent.

ANNEXE 3

GENERATION DE TRAJECTOIRE

A3.1. Introduction

Le problème de la génération de mouvement est de calculer les consignes de référence en position, vitesse et accélération qui sont fonction du temps et qui assurent le passage du robot par une trajectoire désirée, définie par une suite de situations de l'organe terminal ou de configuration articulaire **[1]**.

On peut distinguer les classes de mouvement suivantes :

- Le mouvement entre deux points avec trajectoire libre entre les points.
- Les deux points via des points intermédiaires, spécifiés notamment pour éviter les obstacles, avec trajectoires libre entre les points intermédiaires.
- Le mouvement entre deux points avec trajectoire contrainte entre les points (trajectoire rectiligne par exemple)
- Le mouvement entre deux points via des points intermédiaires avec trajectoire contrainte entre les points intermédiaires.

Dans les deux premiers cas, la génération de trajectoire peut se faire directement dans l'espace articulaire. Dans les deux derniers, la trajectoire étant décrite dans l'espace opérationnel, il est préférable de raisonner dans cet espace.

A3.2. Génération de mouvement et systèmes de commande

les deux approches - la génération dans l'espace articulaire et dans l'espace opérationnel - sont schématisé sur la figureA3.1 et la figureA3.2

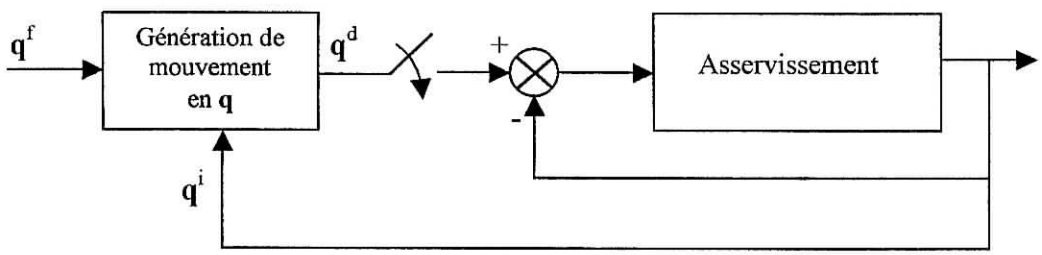


Figure A3.1. *Génération de mouvement dans l'espace articulaire.*

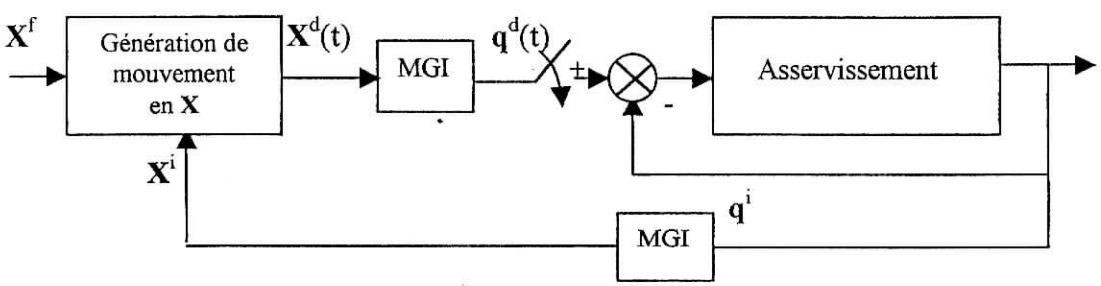


Figure A3.2. *Génération de mouvement dans l'espace opérationnel.*

La génération de mouvement dans l'espace articulaire présente plusieurs avantages :
- Elle nécessite moins de calculs , puisqu'il n'y a pas d'appel au modèle géométrique ou cinématique inverse;
- Le mouvement n'est pas affecté par le passage sur les configurations singulières;
- Les contraintes de vitesses et de couples maximaux sont directement déduites des limites physiques des actionneurs.

En contrepartie, la géométrie de la trajectoire de l'organe terminal dans l'espace opérationnel est imprévisible bien qu'elle soit répétitive : Il y a donc risque de collision lorsque le robot évolue dans un environnement encombré. Ce type de mouvement est par conséquent approprié pour réaliser des déplacements rapides dans un espace dégagé.

La génération de mouvement dans l'espace opérationnel permet de contrôler la géométrie de la trajectoire. En revanche :
- Elle implique la transformation en coordonnées articulaires de chaque point de la trajectoire ;
- Elle peut être mise en échec lorsque la trajectoire calculée passe par une position singulière
- Les limites en couple et en vitesse dans l'espace opérationnel varient selon la configuration du robot.

Le choix d'une méthode de génération de mouvement dépend de la façon avec laquelle on exploite le mouvement ou en d'autres termes de l'application considérée

A3.3. Génération de mouvement entre deux points

On considère un robot de n ddl. Soit $\mathbf{q}^i$, $\mathbf{q}^f$ les vecteurs des coordonnées articulaires correspondant aux configurations initiale et finale. On désigne respectivement par $\mathbf{k}_v$ et $\mathbf{k}_a$ les vecteurs de vitesses et accélérations articulaires maximales. Ces paramètres peuvent être calculer de façon exacte a partir des caractéristiques des actionneurs et des rapport de réduction.

Le mouvement entre $\mathbf{q}^i$ et $\mathbf{q}^f$ en fonction du temps t est décrit par l'équation suivante :

$$q(t) = q^i + r(t).D \quad \text{pour } 0 \le t \le t_f \qquad \text{(A3.1)}$$

$$\dot{q}(t) = \dot{r}(t).D \qquad \text{(A3.2)}$$

avec $D = q^f - q^i$

Les valeurs aux limites de la fonction d'interpolation r(t) sont données par :

$$\begin{cases} r(0) = 0 \\ r(t_f) = 1 \end{cases}$$

L'expression (A3.1) s'écrit aussi :

$$\mathbf{q}(t) = \mathbf{q}^f(t) - [1 - r(t)].\mathbf{D} \qquad \text{(A3.3)}$$

formulation qui convient dans les cas de poursuite de cible, lorsque le terme $\mathbf{q}^f$ varie. Dans ce cas, $\mathbf{D} = \mathbf{q}^f(0) - \mathbf{q}^i$

Plusieurs fonctions permettent de satisfaire le passage par $\mathbf{q}^i$ à t=0 et par $\mathbf{q}^f$ à $t=t_f$. Nous étudierons successivement l'interpolation polynomiale, mais il existe cependant d'autres approches telle que la loi Bang-Bang ou la loi trapèze

A3.4. Génération de mouvement par interpolation polynomiale

les modes d'interpolation polynomiale fréquemment rencontrés sont l'interpolation linéaire et l'interpolation par les polynômes de degrés trois et cinq **[1]**.

A3.4.1. Interpolation linéaire

Il s'agit de l'interpolation la plus simple : le mouvement de chaque articulation est décrit par une équation linéaire en temps. L'équation du mouvement s'écrit :

$$\mathbf{q}(t) = \mathbf{q}^i + (t/t_f).\mathbf{D} \qquad \text{(A3.4)}$$

cette loi de mouvement est continue en position mais discontinue en vitesse. L'utilisation d'une telle loi de mouvement est inacceptable sur les robots réels à cause des à-coups qu'elle provoque.

A3.4.2. Interpolation de degré trois

Si l'on impose une vitesse nulle aux points de départ et d'arrivée, on ajoute deux contraintes aux contraintes de position. Le degré minimal du polynôme qui satisfait ces quatre contraintes est de degré 3 et a pour forme générale :

$$\mathbf{q}(t) = \mathbf{a}_0 + \mathbf{a}_1.t + \mathbf{a}_2.t^2 + \mathbf{a}_3.t^3 \qquad \text{(A3.5)}$$

pour satisfaire les conditions aux limites, les coefficients ont pour expression :

$$\begin{cases} \mathbf{a}_0 = \mathbf{q}^i \\ \mathbf{a}_1 = 0 \\ \mathbf{a}_2 = \left(\dfrac{3}{t_f^2}\right).\mathbf{D} \\ \mathbf{a}_3 = -\dfrac{2}{t_f^3}.\mathbf{D} \end{cases} \qquad \text{(A3.6)}$$

L'expression (A3.5) peut encore s'écrire sous les formes (A3.1) et (A3.3) en prenant :

$$r(t) = 3.\left(\frac{t}{t_f}\right)^2 - 2.\left(\frac{t}{t_f}\right)^3 \qquad \text{(A3.7)}$$

la figure A3.3 donne l'évolution des positions, vitesse et accélération pour l'axe j. cette loi de mouvement assure la continuité des vitesses mais pas celle des accélérations.

Pour une articulation quelconque j, la vitesse est maximum lorsque $t=t_f/2$. Elle a pour valeur :

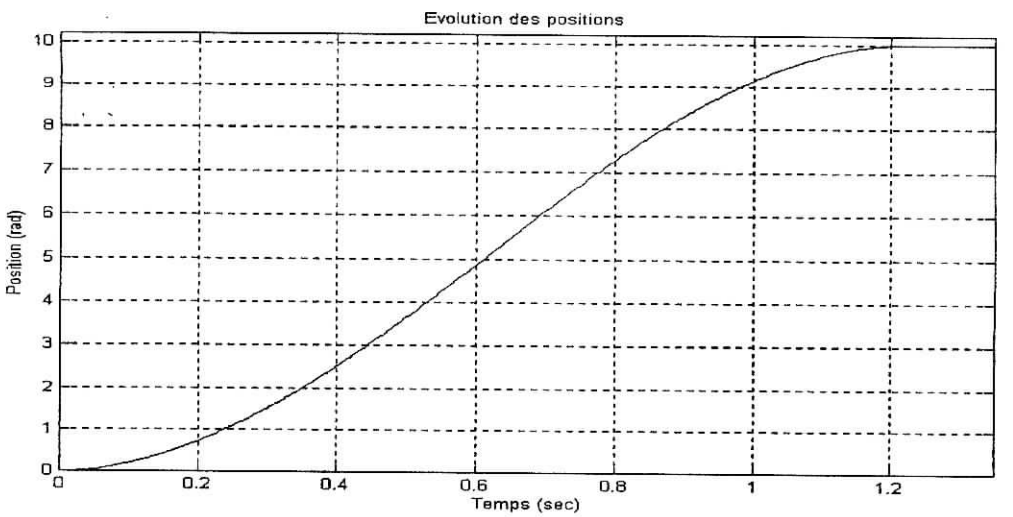
$$\left|\dot{q}_{jmax}\right| = \frac{3|Dj|}{2.t_f} \qquad \text{avec } |D_j| = \left|q_j^f - q_j^i\right| \tag{A3.8}$$

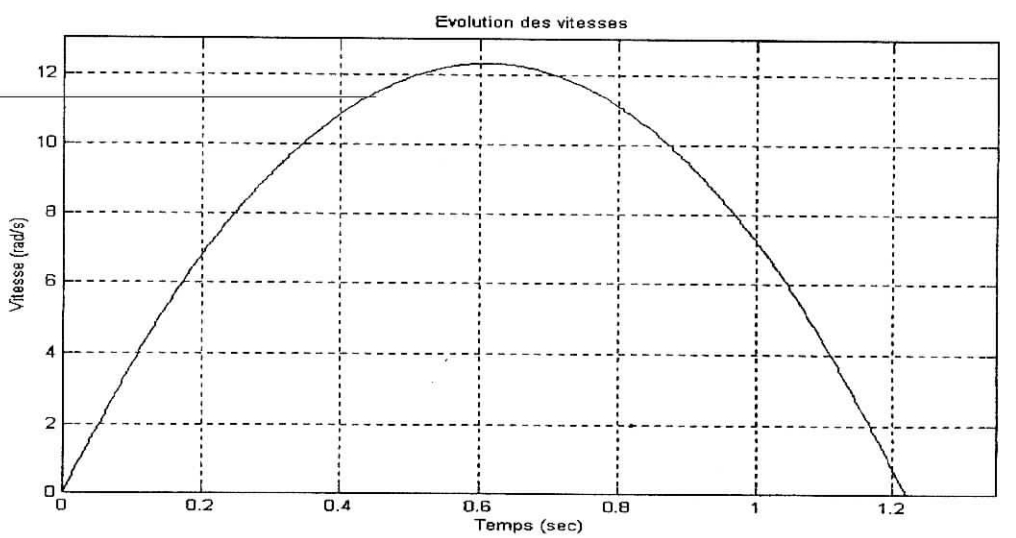
L'accélération est maximum à t=0 et à t= tf. Elle vaut :

$$\left|\ddot{q}_{jimax}\right| = \frac{6.|D_j|}{t_f^2} \tag{A3.9}$$

Simulation pour : (ces simulation sont faite en considérant le facteur temps minimale, voir §2.4.4 qui concerne le calcule de temps minimale).

- I_{max} = 3A; pos_init = 0 ; pos_fin = 10 ;





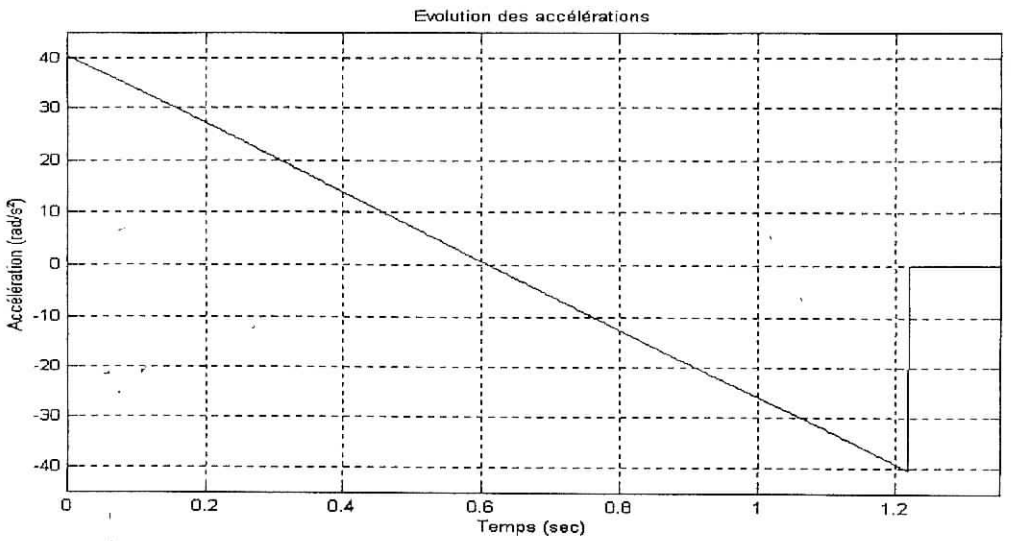


Figure A3.3. *Loi polynomiale de degré trois.*

A3.4.3. Interpolation de degré cinq

Pour les robots à grande vitesse ou transportant des charges importantes, il est nécessaire d'assurer la continuité des accélérations afin d'éviter d'exciter la mécanique. On dit alors que le mouvement est de classe C^2, il faut satisfaire six contraintes et le polynôme d'interpolation doit être de degré cinq **[1]**. Avec les contraintes supplémentaires :

$$\begin{cases} \ddot{q}(0) = 0 \\ \ddot{q}(t_f) = 0 \end{cases} \qquad \text{(A3.10)}$$

On montre que la fonction de position peut se mettre sous la forme (A3.1) ou (A3.3) avec :

$$r(t) = 10.\left(\frac{t}{t_f}\right)^3 - 15.\left(\frac{t}{t_f}\right)^4 + 6.\left(\frac{t}{t_f}\right)^5 \qquad \text{(A3.11)}$$

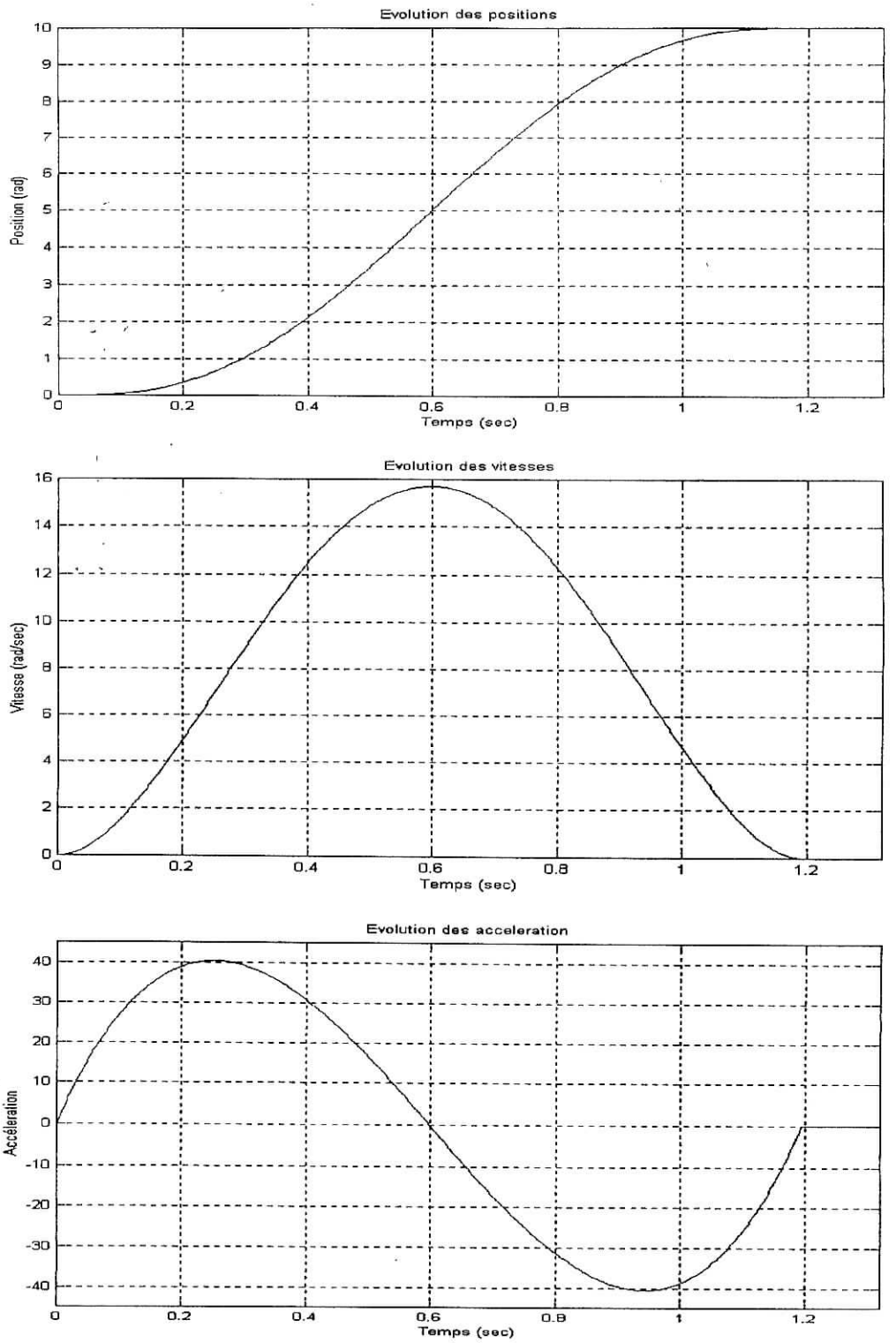
. Les évolutions des positions, vitesse et accélération pour l'articulation j sont présentées à la figureA3.4. les vitesses et accélérations maximales ont pour expression :

$$\left|\dot{q}_{jmax}\right| = \frac{15.\left|D_j\right|}{8.t_f} \qquad \text{(A3.12)}$$

$$\left|\ddot{q}_{jmax}\right| = \frac{10.\left|D_j\right|}{\sqrt{3}.t_f^2} \qquad \text{(A3.13)}$$

Simulation pour : (ces simulation sont faite en considérant le facteur temps minimale, voir §2.4.4).

- $I_{max} = 3A$; pos_init = 0 ; pos_fin = 10 ;



FigureA3.4. *Loi polynomiale de degré cinq.*

A3.4.4. Calcul du temps minimum

Si la durée t_f du mouvement n'est pas spécifiée, ce qui est généralement le cas, et que l'on recherche le temps minimum pour passer de la configuration $\mathbf{q}^i$ à la position $\mathbf{q}^f$ tout en respectant les contraintes de vitesse et d'accélération, on calcule le temps minimum pour

chaque articulation séparément puis on effectue la coordination des articulations sur un temps commun.

Pour calculer le temps minimum on doit saturer la vitesse et/ou l'accélération. on expliquera un peu plus tard qu'on se limitera à saturer que l'accélération, voir tableau ci-dessous

$$t_f = \text{Max}\ (t_{f1}\ ,\ \ldots\ ,\ t_{fn}). \qquad \text{(A3.14)}$$

Fonction d'interpolation	Temps minimum
Interpolation linéaire	$t_{fj} = \dfrac{\lvert D_j \rvert}{k_{vj}}$
Polynôme de degré trois	$t_{fj} = \text{Max}\left[\dfrac{3.\lvert D_j \rvert}{2.k_{vj}}, \sqrt{\dfrac{6\lvert D_j \rvert}{k_{aj}}}\right]$
Polynôme de degré cinq	$t_{fj} = \text{Max}\left[\dfrac{15.\lvert D_j \rvert}{8.k_{vj}}, \sqrt{\dfrac{10\lvert D_j \rvert}{\sqrt{3}.k_{aj}}}\right]$

Tableau A3.1. *Temps minimum pour l'articulation j.*

A3.4.5. Calcul du temps minimum par saturation d'accélération pour un moteur DC

Supposant que notre actionneur et un moteur DC, qui doit déplacer une charge maximale équivalente a un couple résistant $C_{ch\text{-}max}$, cette charge peut être en générale représentée comme un courant max qu'on doit pas dépasser i_{max} , notant que le facteur liant le couple de charge max au niveau du rotor et le courant max au niveau de l'induit est une constante notée k_t (il faut prendre en compte le rapport de réduction si cette charge se présente au niveau de l'articulation après réduction). On peut monter qu'il existe une relation entre le temps mis par l'actionneur pour déplacer et le couple développé par ce dernier, soit :

$$C_{mot} = J.k_a \qquad \text{(A3.15)}$$

k_a : accélération maximale ;
J : Inertie (charge plus articulation) ;
C_{mot} : Couple fourni par le moteur en charge.

$$k_a = \frac{C_{mot}}{J} \qquad / C_{mot} = k_t.i_{max}$$

i_{max} : c'est le courant a ne pas dépasser, ce qui est le cas dans une limitation matériel

$$k_a = \frac{k_t.i_{max}}{J} \qquad \text{(A3.16)}$$

on utilisant les relation du tableau pour une interpolation de degré cinq on obtient directement le temps minimale :

$$t_f = \sqrt{\frac{10.\left|q^f - q^i\right|.J}{\sqrt{3}.k_t.i_{max}}} \qquad \text{(A3.17)}$$

comme il peut y existé des perturbation a la fin du cycle de la trajectoire on maximise t_f comme pour une marge de sécurité de 10% ce qui donne un temps d'exécution de 110% de t_f.

A3.5. Génération de mouvement par la loi trapézoïdale

Avec une loi trapèze, lorsque la vitesse est saturé, le fait de rajouter un palier de vitesse permet de saturer aussi l'accélération et de diminuer le temps de parcours (figureA3.8).
On sait que **[1]** :

$$\left|\dot{q}_{jmax}\right| = \frac{2.\left|D_j\right|}{t_f} \qquad \text{(A3.18)}$$

$$\left|\ddot{q}_{jmax}\right| = \frac{4.\left|D_j\right|}{t_f^2} \qquad \text{(A3.19)}$$

On peut déduire la condition d'existence du palier de vitesse sur l'articulation j:

$$\left|D_j\right| > \frac{k_{vj}^2}{k_{aj}} \qquad \left(\text{On prend } K_v = \sqrt{\frac{\left|D_j\right|.K_{aj}}{2}}\right) \qquad \text{(A3.20)}$$

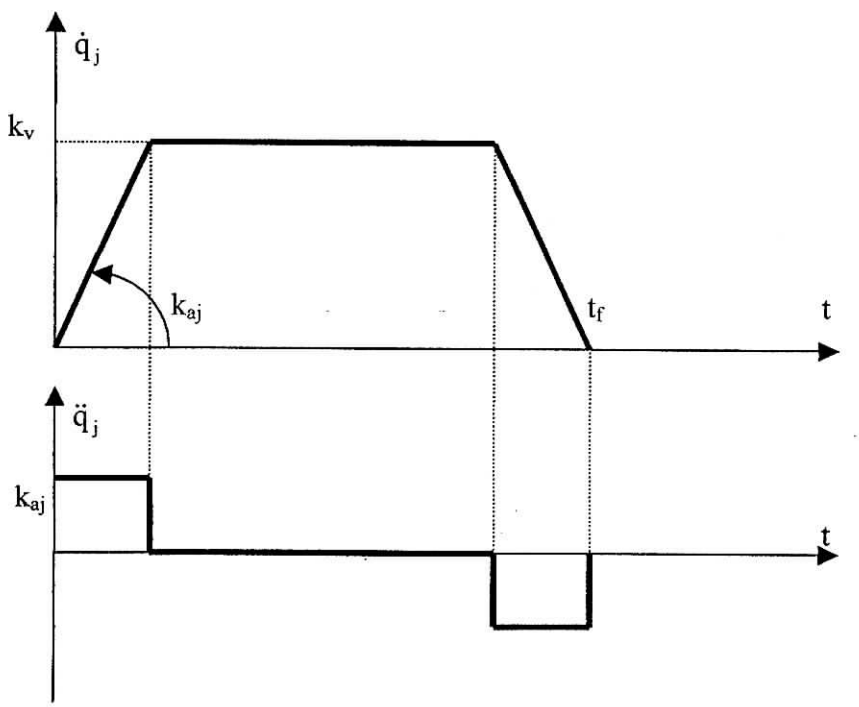


Figure A3.5. *Loi trapèze (en vitesse).*

La loi trapèze est la loi optimale parmi celles qui assurent la continuité en vitesse. Le mouvement de l'articulation j (figure A3.9) est représenté par les relations suivantes :

$$\begin{cases} q_j(t) = q_j^i + \frac{1}{2}.t^2.k_{aj}.\mathrm{Sign}(D_j) & \text{pour } 0 \le t \le \tau_j \\ q_j(t) = q_j^i + (t - \frac{\tau_j}{2}).k_{vj}.\mathrm{Sign}(D_j) & \text{pour } \tau_j \le t \le t_{fj} - \tau_j \\ q_j(t) = q_j^i + \frac{1}{2}.(t_{fj} - t)^2.k_{aj}.\mathrm{Sign}(D_j) & \text{pour } t_{fj} - \tau_j \le t \le t_{fj} \end{cases} \quad \text{(A3.21)}$$

avec :

$$\tau_j = \frac{k_{vj}}{k_{aj}} \quad \text{(A3.22)}$$

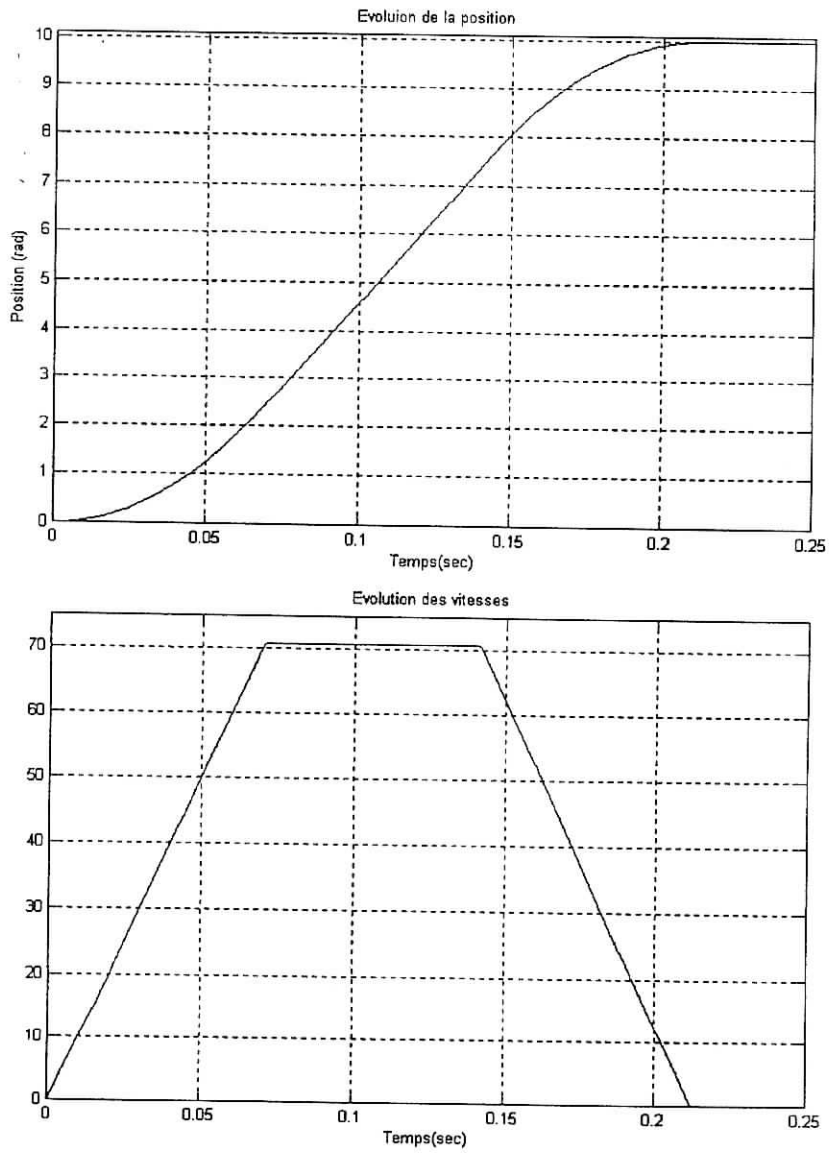


Figure A3.6. *Evolution des position vitesse avec loi trapèze.*

A3.6. Conclusion

Nous avons présenté dans ce chapitre plusieurs méthodes de génération de mouvement couramment utilisées en robotique. Le problème a été traité pour un mouvement entre deux points dans l'espace articulaires, différents modes d'interpolation on été étudiés, notamment la loi Bang-Bang (Trapèze) qui est implanter sur la plupart des contrôleurs industriels. Nous avons établie la relation liant le courant maximum (saturation en accélération) pour un moteur DC et le temps minimum pour une génération de type interpolation.

ANNEXE 4

ETUDE DU

PIC 16F877

A4.1. Introduction

Précédemment, nous avons effectué l'étude du système de réglage échantillonné en se basant sur des considérations pratiques et des limitations de fonctionnements liées directement aux limitations matérielles que se soit l'environnement externe ou le microcontrôlleur lui même, dans ce chapitre on va présenter les caractéristiques ainsi que les périphériques interne spécifique aux PIC16F877, qui est le microcontrôlleur charger d'effectuer la tache du calculateur de processus et d'exécuter l'algorithme du réglage discret, cette étude mettra le point sur les résolutions, fréquences, nombre d'E/S de périphériques, la sortance ainsi que la procédure a suivre pour la programmation, l'environnement de développement et à la fin ICSP (In Circuit serial Programming).

A4.2. Présentation du 16F877

Le 16F877 fait partie de la sous-famille des 16F87x. Cette branche fait partie intégrante de la grande famille des PICs Mid-Range, on peut considérer que le 16F877 constitue le circuit représentant la couche de gamme supérieure de cette famille. De nouveaux circuits ne devraient probablement pas tarder à améliorer encore les performances.

Le 16F877 à les caractéristiques suivantes [7], [8]:

- CPU avec système d'instruction RSIC.
- Seulement 35 instructions.
- Un cycle pour toute les instructions, excepter les instructions de branchements (2 cycles).
- Vitesse de fonctionnement 20 MHz, 200ns par cycle instruction.
- 14 sources d'interruptions.
- Pile mémoire à huit niveaux.
- Mémoire Flash 8K mots x 14 Bits.
  Mémoire RAM 368 mots x 8 Bits.
  Mémoire ROM 256 mots x 8 Bits.
- Mode d'adressage direct, indirect et relative.
- Reset à la mise sous tension (POR)
- Timer Watchdog (WDT)
- Code de protection programmable.
- Fonctionnement en économiseur d'énergie (Sleep).
- Oscillateur sélectable.
- Faible consommation d'énergie
  i < 2 mA typique à 5V, 4 MHz
  i = 20 mA typique à 3V, 32 KHz
  i < 1 mA typique à une source de courant auxiliaire.
- Programmation du composant en mode de transfert série à travers deux pins (ICSP)
- Courant de sortie maximum de 25 mA.
- Accès en mémoire Flash en lecture/écriture.

Les périphériques inclus sont :

- Trois timer :
  Timer0, 8bits timer/compteur avec prédiviseur.
  Timer1, 16bits timer/compteur avec prédiviseur et une entrée horloge auxiliaire.
  Timer2, 8bits timer avec prédiviseur et postdiviseur.

- Deux module CCP (capture 16bits, compare 16bits et PWM 10bits).
- CAN sur 10bits multi-entré(8).
- Port série synchrone en mode SPI (maître/esclave) et I²C (maître/esclave).
- Port USART/SCI/MSSP avec détection d'adresse sur 9bits.
- Port parallèle esclave PSP.

A4.2.1. Organisation mémoire RAM

La mémoire RAM disponible du 16F877 est de 368 octets. Elle est répartie de la manière suivante et donnée page 7 du datasheet :

1) 80 octets en banque 0, adresses 0x20 à 0x6F
2) 80 octets en banque 1, adresses 0xA0 à 0XEF
3) 96 octets en banque 2, adresses 0x110 à 0x16F
4) 96 octets en banque 3, adresses 0x190 à 0x1EF
5) 16 octets communs aux 4 banques, soit 0x70 à 0x7F = 0xF0 à 0xFF = 0x170 à 0x17F = 0x1F0 à 0x1FF

Les octets communs signifies que si on accède au registre (adresse mémoire RAM) 0x70 ou au registre 0XF0, et bien on accède en réalité au même emplacement. Ceci à l'avantage de permettre d'utiliser ces emplacements sans devoir connaître l'état de RP0,RP1, et IRP [**].

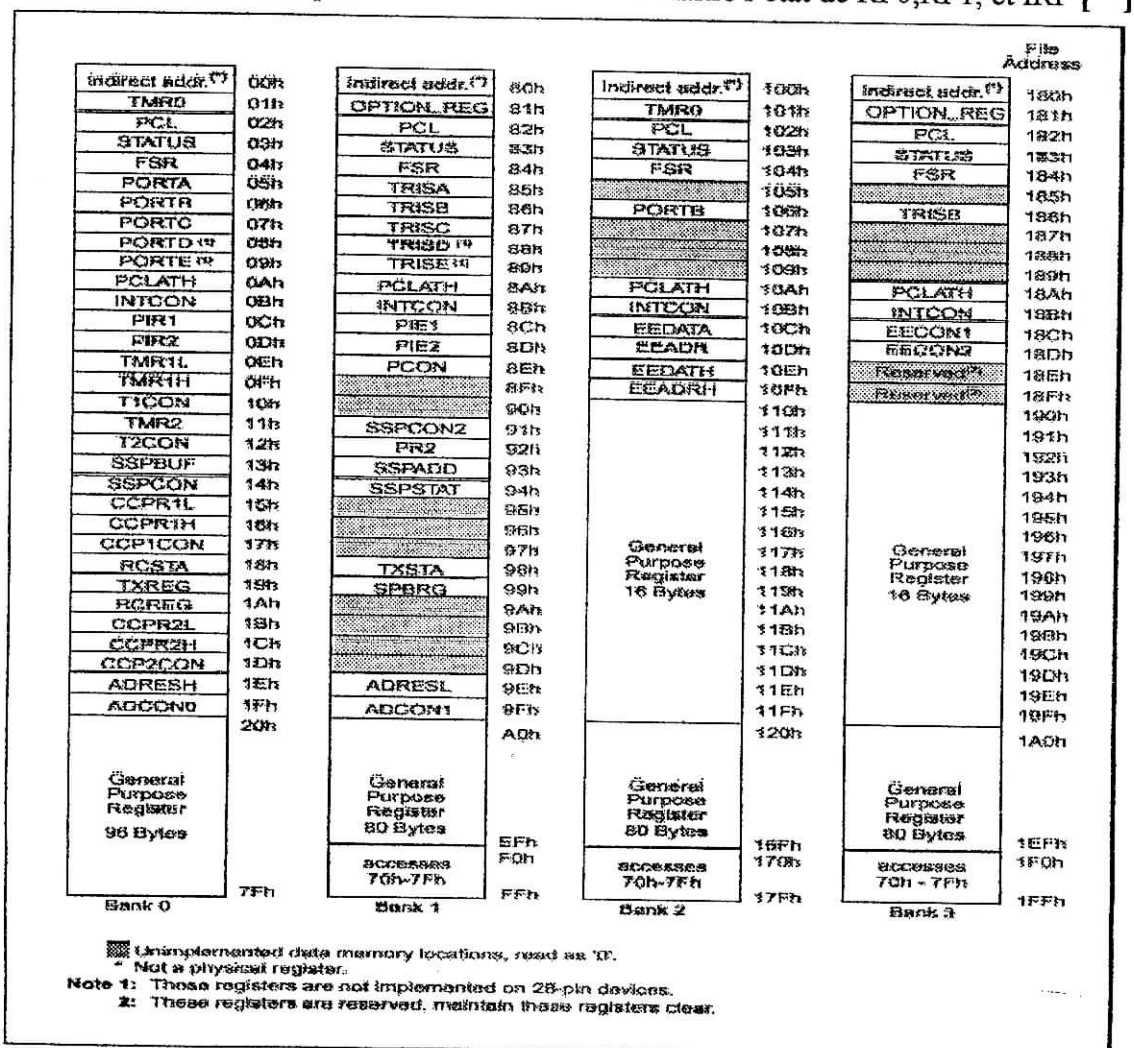


Figure A4.1. *Organisation de la RAM.*

A4.3. Caractéristiques spéciales

A4.3.1. Registre de configuration

Les bits de configurations peuvent être programmés pour sélectionner un mode de fonctionnement particulier. Ces bits sont positionnés dans le même registre sur 14bits dans la mémoire programme (Flash memory) localisé à 2007h. Cette espace est accessible que durant la programmation.

Voici donc ses fonctions, reprises page 122 du datasheet :

- **CP1/CP0** : bits 13/12 et 5/4 : Détermine quelle zone du 16F876 sera protégée contre la lecture. Vous pouvez donc choisir de protéger la totalité du PIC ou seulement une partie. Les différentes zones pouvant être protégées sont les suivantes :

CP1	CP0	
1	1	Aucune protection (_CP_OFF)
1	0	Protection de la zone 0x1F00 à 0x1FFF (_CP_UPPER_256)
0	1	Protection de la zone 0x1000 à 0x1FFF (_CP_HALF)
0	0	Protection de l'intégralité de la mémoire (_CP_ALL)

- **DEBUG** : bit 11 : Debuggage sur circuit. Permet de dédicacer RB7 et RB6 à la communication avec un debugger.
  1 : RB6 et RB7 sont des I/O ordinaires (_DEBUG_OFF)
  0 : RB6 et RB7 sont utilisés pour le Debuggage sur circuit (_DEBUG_ON)

- Bit 10 : non utilisé

- **WRT** : bit 9 : Autorisation d'écriture en flash
  1 : Le programme peut écrire dans les zones non protégées par les bits CP1/CP0 (_WRT_ENABLE_ON)
  0 : Le programme ne peut pas écrire en mémoire flash (_WRT_ENABLE_OFF)

- **CPD** : bit 8 : Protection en lecture de la mémoire eeprom.
  1 : mémoire eeprom non protégée (_CPD_OFF)
  0 : mémoire eeprom protégée (_CPD_ON)

- **LVP** : bit 7 :Utilisation de la pin RB3/PGM pour la programmation
  1 : La pin RB3 permet la programmation du circuit sous tension de 5V (_LVP_ON)
  0 : La pin RB3 est utilisée comme I/O standard (_LVP_OFF)

- **BODEN** : bit 6 : provoque le reset du PIC en cas de chute de tension (surveillance de la tension d'alimentation)
  1 : En service (_BODEN_ON).
  0 : hors service (_BODEN_OFF)

- **PWRTE** :bit 3 : Délai de démarrage à la mise en service. Attention, est automatiquement mis en service si le bit BODEN est positionné.
  1 : délai hors service (sauf si BODEN = 1) (_PWRTE_OFF)
  0 : délai en service (_PWRTE_ON)
- **WDTE** : bit 2 : Watchdog timer

1 : WDT en service (_WDT_ON)
0 : WDT hors service (_WDT_OFF)

- **FOSC1/FOSC0** : bits 1/0 : sélection du type d'oscillateur

11 : Oscillateur de type RC (_RC_OSC)
10 : Oscillateur haute vitesse (_HS_OSC)
01 : Oscillateur basse vitesse (_XT_OSC)
00 : Oscillateur faible consommation (_LP_OSC)

La directive _CONFIG sur environnement MPLAB permet de géré ces bits par simple combinaison logique. Voici un exemple d'utilisation:

__CONFIG _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC.

A4.3.2. Sélection des oscillateurs

Le 16F877 peut opérer en quatre différent modes d'oscillateurs, qui peuvent être sélectionner a partir du registre de configuration avec les bits FOSC1 et FOSC0; Les trois premiers modes LP, XT et HS , un oscillateur type cristal ou céramique doit être connecté comme le montre la figure ci-dessous .

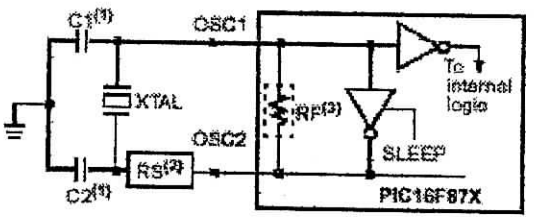


Figure A4.1. *Connexion de l'oscillateur cristal sur les pins OSC1 et OSC2.*

Le tableau 8.1 donne les valeurs recommandées des capacités à utiliser pour un oscillateur de type cristal.

Osc Type	Crystal Freq	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Tableau A4.1. *Valeurs typiques pour oscillateur type cristal.*

A4.3.3. Le Reset

Le 16F877 dispose de 6 types de reset différents les uns les autres qui sont :

- Apparition de la tension d'alimentation après une coupure
- Application d'un niveau bas sur la pin MCLR durant le déroulement du programme
- Application d'un niveau bas sur la pin MCLR durant le mode « Sleep »
- Débordement du Watchdog durant le déroulement du programme
- Débordement du Watchdog durant le mode « Sleep »
- Remontée de la tension d'alimentation après une chute partielle

Parmi ces 6 types, le débordement du Watchdog durant le mode de sommeil n'est pas à proprement parler un reset, puisqu'il provoque le réveil du PIC, sans provoquer un reset à l'adresse 0x00, ni même la modification des registres affectés par les resets classiques.

A4.3.3.1. Le reset et les registres STATUS et PCON

Pour déterminer quel événement a provoqué le reset, 4 bits sont utilisés, 2 se trouvent dans le registre STATUS, et 2 dans un nouveau registre, le registre **PCON**. Voici la description de ce registre:

b0 : **BOR** : Brown Out Reset : Reset sur chute de tension
b1 : **POR** : Power On Reset : Reset par mise sous tension
b2-b7: : Non utilisés, lus comme «0»

Les bits concernés par le reset dans le registre **STATUS** sont les suivants :

b3 : **PD** : Power Down bit : passage en mode Sleep
b4 : **TO** : Time Out bit : reset par débordement du Watchdog

Ces bits sont actifs à l'état bas, c'est lorsqu'ils sont à «0» qu'ils indiquent que l'événement a eu lieu. Il faut noter que le bit BOR est dans un état indéterminé lors de la mise sous tension. Ce bit est forcé à 0 lorsqu'une chute de tension provoque le reset du PIC.

A4.3.4. Les interruptions

Voici un tableau récapitulatif des 14 interruptions disponibles sur le 16F877 :

Déclencheur	Flag	RegistreF	Adr	PEIE	Enable	RegistreE	Adr
Timer0	T0IF	INTCON	0x0B	NON	T0IE	INTCON	0x0B
Pin RB0 / INT	INTF	INTCON	0x0B	NON	INTE	INTCON	0x0B
Ch. RB4/RB7	RBIF	INTCON	0x0B	NON	RBIE	INTCON	0x0B
Convert. A/D	ADIF	PIR1	0x0C	OUI	ADIE	PIE1	0x8C
Rx USART	RCIF	PIR1	0x0C	OUI	RCIE	PIE1	0x8C
Tx USART	TXIF	PIR1	0x0C	OUI	TXIE	PIE1	0x8C
Port série SSP	SSPIF	PIR1	0x0C	OUI	SSPIE	PIE1	0x8C
Module CCP1	CCP1IF	PIR1	0x0C	OUI	CCP1IE	PIE1	0x8C
Module CCP2	CCP2IF	PIR2	0x0D	OUI	CCP2IE	PIE2	0x8D
Timer 1	TMR1IF	PIR1	0x0C	OUI	TMR1IE	PIE1	0x8C
Timer 2	TMR2IF	PIR1	0x0C	OUI	TMR2IE	PIE1	0x8C

EEPROM	EEIF	PIR2	0x0D	OUI	EEIE	PIE2	0x8D
SSP mode I2C	BCLIF	PIR2	0x0D	OUI	BCLIE	PIE2	0x8D
Port parallèle	PSPIF	PIR1	0x0C	OUI	PSPSIE	PIE1	0x8C

- **Déclencheur** : Evénement ou fonction qui est la source de l'interruption.
- **Flag** : Bit qui se trouve positionné lorsque l'événement survient.
- **RegistreF** : Registre auquel le flag appartient.
- **Adr** : Adresse du RegistreF.
- **PEIE** : Indique si le positionnement de PEIE du registre INTCON joue un rôle de mise en service des interruptions périphériques.
- **Enable** : nom du bit qui permet d'autoriser ou non l'interruption.
- **RegistreE** : Registre qui contient ce bit.
- **Adr** : adresse du RegistreE.

A4.3.4.1. Le registre INTCON et les interruptions périphériques

Voici le contenu du registre INTCON :

INTCON pour le 16F877							
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

A). Mise en service des interruptions primaires

Nous disposons de 3 interruptions primaires. T0IF, INTF, RBIF. Pour mettre en service une de ces interruptions, il faut suivre les étapes suivante :

1) Valider le bit concernant cette interruption
2) Valider le bit GIE (General interrupt enable) qui met toutes les interruptions choisies en service

B). Mise en service des interruptions périphériques

Les interruptions périphériques sont tributaires du bit de validation générale des interruptions périphériques PEIE. Voici donc les 3 étapes nécessaires à la mise en service d'une telle interruption :

1) Validation du bit concernant l'interruption dans le registre concerné (**PIE1** ou **PIE2**)
2) Validation du bit **PEIE** (PEripheral Interrupt Enable bit) du registre **INTCON**
3) Validation du bit GIE qui met toutes les interruptions en service.

La mise en service de **PEIE** ne dispense pas l'initialisation du bit GIE, qui reste prioritaire. Cette architecture nous permet donc de couper toutes les interruptions d'un coup (effacement de GIE) ou de couper toutes les interruptions périphériques en une seule opération (effacement de **PEIE**) tout en conservant les interruptions primaires. Ceci nous donne davantage de souplesse dans le traitement des interruptions, mais complique un petit peu le traitement. Voici donc un schéma qui interprète le fonctionnement de la logique d'interruption du 16F877 :

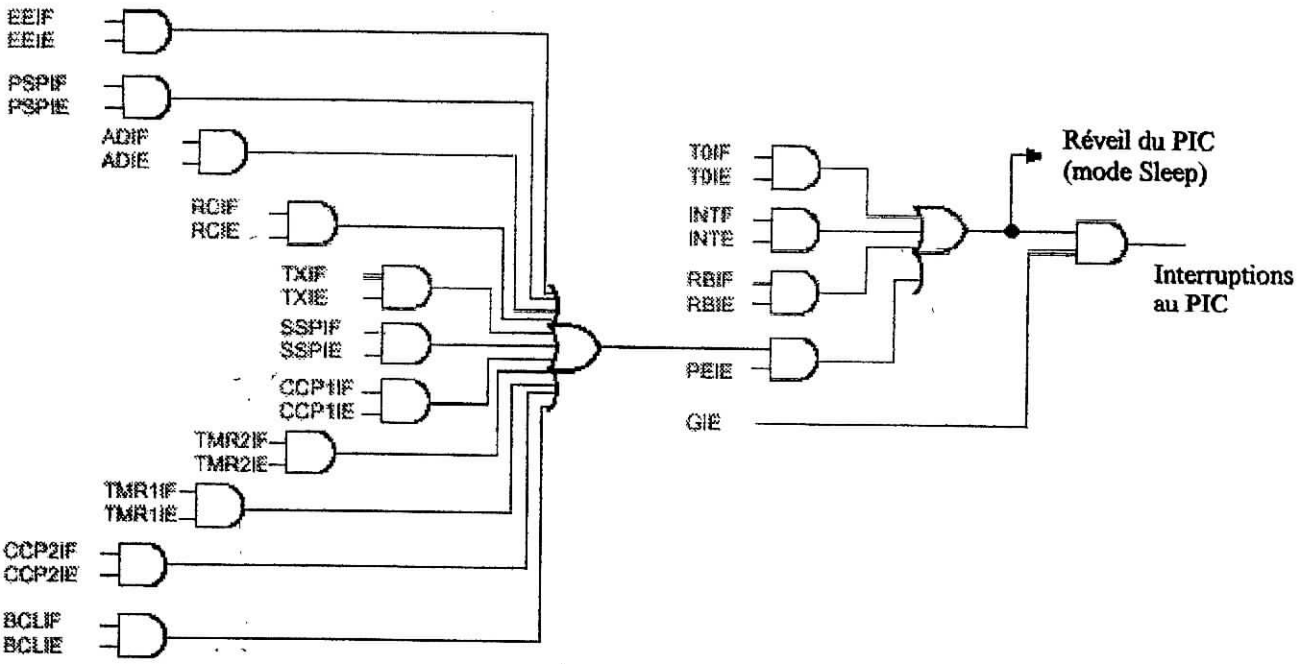


Figure A4.3. *Logique d'interruption.*

A4.3.4.2. Les registres PIE1, PIE2, PIR1 et PIR2

Le 16F877 dispose de plus de sources d'interruptions que ne peut en gérer le registre INTCON. Donc, les autorisations d'interruptions vont se trouver dans les registres PIE1 et PIE2. Les flags correspondants se trouvent dans les registres PIR1 et PIR2.

Les 2 registres d'autorisations (PIE1 et PIE2) se trouvent en banque 1, tandis que les registres de flags (PIR1 et PIR2) se trouvent en banque 0.

Les bits des registres PIEx permettent d'autoriser les interruptions correspondantes, mais ces bits ne sont opérationnels que si le bit PEIE du registre INTCON est mis à 1. Dans le cas contraire, toutes ces interruptions sont invalidées. Pour qu'une des interruptions décrites ici soit opérationnelle, il faut donc la triple condition suivante :

- Le bit GIE du registre INTCON doit être mis à 1
- Le bit PEIE du registre INTCON doit être mis à 1
- Le bit correspondant à l'interruption concernée (registre PIE1 ou PIE2) doit être mis à 1.

Voici maintenant le rôle de chacun des bits d'autorisation d'interruption, sachant qu'un bit à « 1 » autorise l'interruption correspondante :

PIE1

PSPIE	: Lecture/écriture sur le port PSP //
ADIE	: Conversion analogique/digitale
RCIE	: Réception d'un caractère sur le port série USART
TXIE	: Emission d'un caractère sur le port série USART
SSPIE	: Communication sur le port série synchrone SSP
CCP1IE	: Evénement sur compare/capture registre 1
TMR2IE	: Correspondance de valeurs pour le timer TMR2
TMR1IE	: Débordement du timer TMR1

PIE2

EEIE : Ecriture dans l'EEPROM
BCLIE : Collision de bus pour le port série synchrone I2C
CCP2IE : Evénement sur compare/capture registre 2

Et maintenant, les flags correspondants. Attention, certains flags nécessitent une explication assez poussée, donc pour plus de détail voir **[référence des PICs]**

PIR1

PSPIF : Lecture ou écriture terminée sur le port // du 16F877
ADIF : Fin de la conversion analogique/digitale
RCIF : Le buffer de réception de l'USART est plein (lecture seule)
TXIF : Le buffer d'émission de l'USART est vide (lecture seule)
SSPIF : Fin de l'événement dépendant du mode de fonctionnement comme suit :
Mode SPI : Un caractère a été envoyé ou reçu
Mode I2C esclave : Un caractère a été envoyé ou reçu
Mode I2C maître : Un caractère a été envoyé ou reçu
ou fin de la séquence « start »
ou fin de la séquence « stop »
ou fin de la séquence « restart »
ou fin de la séquence « acknowledge »
ou « start » détecté durant IDLE (multimaître)
ou « stop » détecté durant IDLE (multimaître)
CCP1IF : Evénement compare/capture 1 détecté suivant mode :
Mode capture : capture de la valeur TMR1 réalisée
Mode compare: La valeur de TMR1 atteint la valeur programmée
TMR2IF : La valeur de TMR2 atteint la valeur programmée
TMR1IF : Débordement du timer TMR1

PIR2

EEIF : Fin d'écriture de la valeur en EEPROM
BCLIF : Collision de bus, quand le SSP est configuré en maître I2C
CCP2IF : Evénement compare/capture 2 détecté suivant le mode :
Mode capture : capture du TMR1 réalisée.
Mode compare : La valeur de TMR1 atteint la valeur programmée.

A4.3.4.3. Mécanisme d'interruptions

- Tout d'abord, l'adresse de début de toute interruption est fixe. Il s'agit toujours de l'adresse 0x04.
- Toute interruption provoquera le saut du programme vers cette adresse. Toutes les sources d'interruption arrivant à cette adresse, si il y a lieu d'utiliser plusieurs sources d'interruptions, il faudra déterminer par une routine de test pour déterminer la source d'interruption.
- Le PIC en se connectant à cette adresse, ne sauvent rien automatiquement, hormis le contenu du PC, qui servira à connaître l'adresse du retour de l'interruption. C'est donc à l'utilisateur de se charger des sauvegardes.
- Le contenu du PC est sauvé sur la pile interne (8 niveaux). Donc, si on utilise des interruption, on ne dispose plus que de 7 niveaux d'imbrication pour nos sous-programmes. Moins si on utilise des sous-programmes dans nos interruption.

Le temps de réaction d'une interruption est calculé de la manière suivante :

- Le cycle courant de l'instruction est terminé.
- Le flag d'interruption est lu au début du cycle suivant..
- Celui-ci est achevé, puis le processeur s'arrête un cycle pour charger l'adresse 0x04 dans PC.
- Le processeur se connecte alors à l'adresse 0x04 où il lui faudra un cycle supplémentaire pour charger l'instruction à exécuter.

Le temps mort total sera donc compris entre 3 et 4 cycles :

- Une interruption ne peut pas être interrompue par une autre interruption. Les interruptions sont donc invalidées automatiquement lors du saut à l'adresse 0x04 par l'effacement du bit GIE.
- Les interruptions sont remises en service automatiquement lors du retour de l'interruption. L'instruction RETFIE agit donc exactement comme l'instruction RETURN, mais elle repositionne en même temps le bit GIE.

A4.4. Périphériques utilisés

A4.4.1. Les ports entrée/sortie

Nous allons maintenant voir dans ce chapitre les différents ports du 16F87x dans leur utilisation en tant que PORT I/O, c'est-à-dire sans utiliser les fonctions spéciales des pins concernées. Les PICs 16F87x disposent en effet de 5 ports d'E/S ayant plusieurs possibilités d'utilisation, nous présenterons les caractéristiques les plus générales :

A4.4.1.1. Le PORT A

Il dispose de 6 pins I/O numérotées de RA0 à RA5. Nous avons donc 6 bits utiles dans le registre PORTA et 6 bits dans le registre TRISA (qui sert a déterminer le sens de transfert). Les bits 6 et 7 de ces registres ne sont pas implémentés, lus comme des «0».

Au moment du reset, le PORTA est configuré comme un ensemble d'entrées analogiques. Donc, nous devrons forcer une valeur dans le registre ADCON1 dans notre routine d'initialisation pour pouvoir utiliser ce port comme port d'entrée/sortie de type général. Le registre ADCON1 est le registre de configuration des entrées du convertisseur A/D.

Nous devrons placer la valeur B'x000011x', dans lequel x vaut 0 ou 1, pour utiliser le PORTA en mode I/O « numérique ». Pour plus de précisions voir **[7],[8]** (cette valeur provient du tableau 11-2 du datasheet).

Attention, l'utilisation de cette valeur influe également sur le fonctionnement du PORTE, comme nous le verrons plus loin.

A4.4.1.2. Le PORTB

Ce port IO fonctionnent exactement de la même manière que PORTA, mais concernent bien entendu les 8 pins RB. Tous les bits sont donc utilisés dans ce cas pour le registres TRISB, une particularité du PORTB c'est que les entrées du PORTB peuvent être connectées à une résistance de rappel au +5V de manière interne.

La sélection s'effectuant par le bit 7 du registre OPTION. Le schéma interne visible figureA4.4, les figures du datasheet nous montre que les bits b0 et b4/b7 peuvent être utilisés comme source d'interruption, le bit 0 peut de plus être utilisé de manière autonome pour générer un autre type d'interruption.

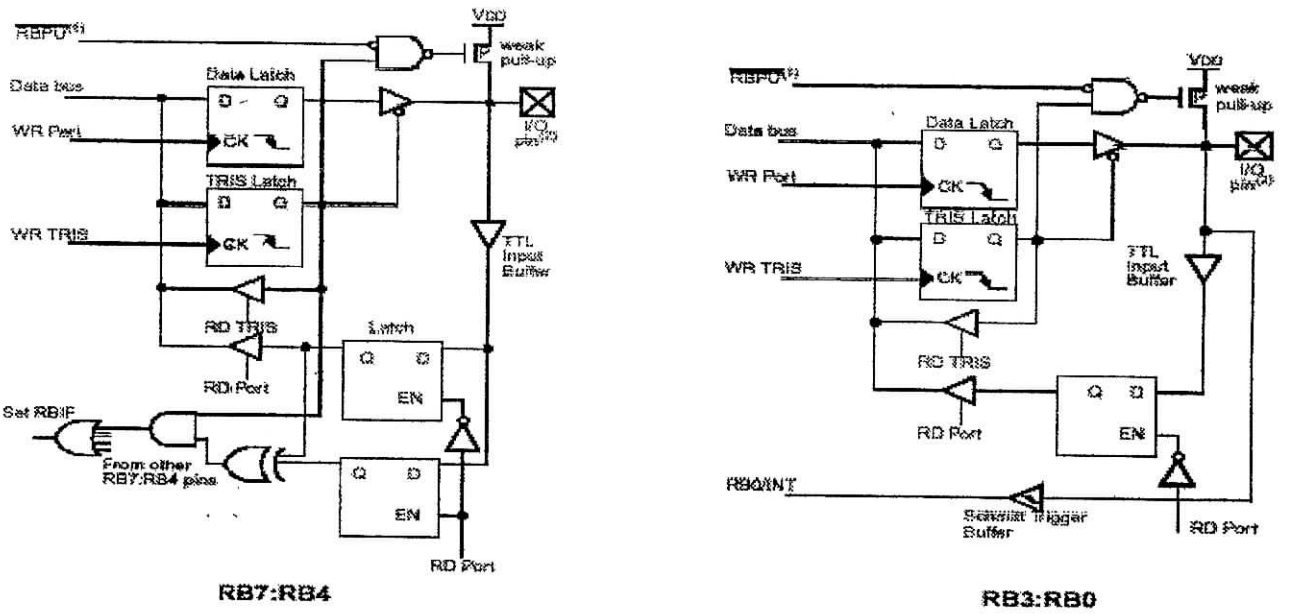


Figure A4.4. *Electronique des pins du PortB.*

Notez la pin RB0 qui, en configuration d'entrée, est de type « trigger de Schmitt » quand elle est utilisée en mode interruption « INT ». La lecture simple de RB0 se fait, elle, de façon tout à fait classique, en entrée de type TTL.

A4.4.1.3. Le PORTC

Ce PORT fonctionne de façon classique. Nous trouvons donc un registre TRISC localisé dans la banque 1, qui permet de décider quelles sont les entrées et quelles sont les sorties. Le fonctionnement est identique à celui des autres TRIS, à savoir que le positionnement d'un bit à « 1 » place la pin en entrée, et que le positionnement de ce bit à «0» place la dite pin en sortie.

Comme pour tous les autres ports, que la mise sous tension du PIC, et tout autre reset, force tous les bits utiles de TRISx à 1, ce qui place toutes les pins en entrée.

Nous trouvons également le registre PORTC, qui, comme les autres PORTx, se trouve dans la banque 0. Son fonctionnement est, de nouveau, identique à celui des autres PORTx Ensuite, au niveau électronique, nous noterons que toutes les pins, lorsqu'elles sont configurées en entrée, sont des entrées de type « trigger de Schmitt ». Ceci permet d'éviter (en simplifiant) les incertitudes de niveau sur les niveaux qui ne sont ni des 0V, ni des +5V, donc, en général, sur les signaux qui varient lentement d'un niveau à l'autre. A la vue de ceci, l'utilisation du PORTC ne devrait poser aucun problème.

A4.4.1.4. Le PORTD

Ce PORT fonctionne de façon identique aux autres, dans son mode de fonctionnement général. Le registre TRISD comportera donc les 8 bits de direction, pendant que le registre

PORTD correspond aux pins I/O concernées. Notons qu'ici également, les 8 pins I/O, en mode entrée, sont du type « trigger de Schmitt ». Prenons garde que le fonctionnement de ce port dépend également de la valeur que nous placerons dans TRISE, qui concerne pourtant, à première vue, le PORTE. Mais, au moment d'une mise sous tension, la valeur qui est placée automatiquement dans TRISE configure le PORTD en port I/O de type général.

A4.4.1.5. Le PORTE

Il ne comporte que 3 pins, RE0 à RE2, mais, contrairement aux autres ports, les bits non concernés de TRISE sont, cette fois, implémentés pour d'autres fonctions. Nous verrons que les pins REx peuvent également être utilisées comme pins d'entrées analogiques. C'est le registre ADCON1 qui détermine si ce port sera utilisé comme port I/O ou comme port analogique.

Par défaut, le port est configuré comme port analogique, et donc, comme pour le PORTA, nous devons placer la valeur adéquate dans ADCON1 pour pouvoir l'utiliser comme port I/O numérique. Au niveau électronique, les pins REx utilisées en entrée seront du type «trigger de Schmitt ».

Le registre TRISE

Le registre TRISE dispose de certains bits de configuration qui ne sont pas présents sur les autres TRISx. Voyons donc le rôle de chacun des bits de TRISE :

b7 :	IBF	: Input Buffer Full status bit
b6 :	OBF	: Output Buffer Full status bit
b5 :	IBOV	: Input Buffer OVerflow detect bit.
b4 :	PSPMODE	: Parallel Slave Port MODE select bit
b3 :	0	: Non implémenté (lu comme « 0 »)
b2 :		: Direction I/O pour RE2 (1 = entrée)
b1 :		: Direction I/O pour RE1
b0 :		: Direction I/O pour RE0

Nous voyons que les bits 0 à 2 fonctionnent de façon identique à ceux des autres TRISx. Un bit à « 1 » positionne la pin en entrée, un bit à « 0 » la positionne en sortie. Les bits 7 à 5 concerne le mode PSP. Le bit 4 (PSPMODE), c'est ce bit qui détermine si le PORTD (et non le PORTE) sera utilisé en port I/O classique. Si nous plaçons 1 ici, le PORTD sera considéré comme un port d'interfaçage avec un microprocesseur. Les bits 0 à 2 sont forcés à « 1 » lors d'une mise sous tension, plaçant RE0 à RE2 en entrée, par contre, les autres bits sont forcés à « 0 » lors de cette mise sous tension. Donc le PORTD fonctionnera en mode classique.

A4.4.2. Les Timers

A4.4.2.1. Timer 0

Le module timer 0 qui peut fonctionner en timer/compteur a les caractéristiques suivantes:

- 8 bits timer/compteur. Registre **TMR0**.
- accès en lectures ou écritures.
- 8 bits de configuration avec prédiviseur programmable. Registre **OPTION_REG**.
- sélection d'horloge interne ou externe.

- interruption au débordement de FFh à 00h. Registre **INTCON**.
- sélection du front du signale d'horloge (montant ou descendant).

Le timer 0 utilise, pour ses incrémentations le registre de 8 bits TMR0. Ce registre contient une valeur indéterminée à la mise sous tension.

clrf TMR0 ; commencer le comptage à partir de 0

Il est impossible d'arrêter le fonctionnement du timer. Si on veux suspendre le comptage du temps (en mode timer), il faut faire passer le timer0 en mode compteur. Ceci, bien entendu, à condition que la pin RA4 ne soit pas affectée à une autre utilisation.

A). L'écriture dans TMR0

Tout d'abord, si nous nous trouvons en mode de fonctionnement « timer » du timer0, nous pouvons être amenés à écrire une valeur (ou à ajouter une valeur) dans le registre TMR0

Au moment de l'écriture d'une valeur dans TMR0, la première chose qu'il va se passer, est que le contenu du prédiviseur, s'il était affecté au timer 0 (Il sert également pour le watchdog) va être remis à 0. (le contenu du prédiviseur n'est pas égale a la valeur du prédiviseur)

Ensuite, les 2 cycles d'instruction suivants seront perdus au niveau du comptage du timer, la mesure de temps reprendra donc à partir du 3ème cycle suivant. Il importe de comprendre que le prédiviseur fonctionne de la façon suivante :

- Le prédiviseur compte jusque sa valeur programmée par PS0/PS2.
- Une fois cette valeur atteinte, TMR0 est incrémenté
- Le prédiviseur recommence son comptage à l'impulsion suivante

Le prédiviseur peut prendre plusieurs valeurs (2, 4, 8, 16, 32, 64, 128, 256) selon la valeur de PS0, PS1, PS2.

Lorsque nous écrivons dans TMR0, il nous appartient, si nécessaire, de prendre en compte la perte d'informations dues a l'effacement du prédiviseur lors de l'écriture.

B). le timing en mode compteur

Il y a quelques précautions à prendre lorsqu'on travaille avec le timer 0 en mode compteur. En effet, les éléments à compter sont, par définition, asynchrones avec l'horloge du PIC. Or la sortie du prédiviseur, et donc l'incrémentation du registre TMR0 est, elle, synchronisée sur le flanc montant de Q4 (Q4 étant le 4ème clock de l'horloge du PIC).

On peut dire que la sortie du prédiviseur passe à 1 au moment où la valeur de comptage est égale à la valeur du prédiviseur choisie. Une fois cette valeur dépassée, la sortie du prédiviseur repasse à 0, jusqu'au multiple suivant.

Mais l'incrémentation de TMR0 se fait sur le test suivant : On a incrémentation si au moment du flanc montant de Q4, la sortie du prédiviseur est à 1.

On voit alors tout de suite que si le signal est trop rapide, on risque de rater le débordement du prédiviseur, celui-ci étant repassé à 0 avant d'avoir eu apparition du flanc montant de Q4.

On peut donc en déduire que pour être certain de ne rien rater, la sortie de notre prédiviseur devra être maintenue au minimum le temps séparant 2 « Q4 » consécutifs. Donc,

l'équivalent de 4 Tosc (temps d'oscillateur), soit la durée d'un cycle d'instruction. Nous aurons, de plus quelques limites dues à l'électronique interne. Ceci nous donne les règles suivantes :

- Si on n'utilise pas de prédiviseur, la sortie du prédiviseur est égale à son entrée, donc :

- La durée totale du clock ne pourra être inférieure à 4 Tosc.
- De plus, on ne pourra avoir un état haut <2 Tosc et un état bas <2 Tosc
- De plus, l'état haut et l'état bas ne pourront durer chacun moins de 20ns.

- Si on utilise un prédiviseur, la sortie est égale à l'entrée divisée par la valeur du prédiviseur, donc :
- La durée totale du clock ne pourra être inférieure à 4 Tosc divisée par le prédiviseur
- De plus, on ne pourra avoir un état haut <2 Tosc divisé par le prédiviseur et idem pour l'état bas
- De plus, l'état haut et l'état bas ne pourront durer chacun moins de 10ns.

Calculons quelles sont les fréquences maximales que nous pouvons appliquer sur la pin TOCKI à un PIC cadencé à une fréquence de 10MHz :

- Sans prédiviseur, la contrainte en terme de Tosc nous donne :

Tosc = 1/f = 1 / 10^7 = 10^{-7} s = 100 ns
Temps minimum en terme de Tosc = 4 Tosc = 400 ns

Ceci respecte la contrainte de temps absolu qui nous impose un temps > 40ns (20+20). Donc, nous aurons comme fréquence maximale :

F = 1/Tosc = 1/(4*10^{-7}) = 2,5 * 10^6 = 2,5 MHz.

Il va de soi qu'avec notre 16F877 cadencé à 20 MHz, la fréquence maximale utilisable sans prédiviseur est de 5MHz. Nous voyons donc que nous pouvons dire que la fréquence maximale (en signal carré) utilisable avec le timer 0 configuré en mode compteur sans prédiviseur, est égale au quart de la fréquence du quartz employé pour son horloge.

- Voyons maintenant le cas du prédiviseur. Utilisons la valeur maximale, soit 256 :

Temps minimum en terme de Tosc = 4 Tosc divisé par le prédiviseur = 400 ns / 256 = 1,56 ns.

Ce temps minimum ne respecte pas le temps minimum absolu imposé, qui est de 10 ns. C'est donc ce dernier temps dont nous tiendrons compte. Notre fréquence maximale est donc de :

F = 1/T = 1/ (20*10^{-9}) = 50 . 10^6 Hz = 50 MHz.

Donc, vous voyez que vos PICs peuvent, à condition d'utiliser le prédiviseur à une valeur suffisante, compter à des fréquences pouvant atteindre 50 MHz.

C). Interruption du Timer0

L'interruption du timer0 est générée quand le registre TMR0 déborde de la valeur FFh à 00h. Ce débordement positionne le bit T0IF du registre **INTCON** <2>, l'interruption peut être

masquer par la mise à zéro du bit T0IE du registre **INTCON** <5>, le bit T0IF doit être effacé dans la routine qui gère l'interruption du timer0 avant de la remettre en service à la fin de cette routine, faute de quoi le microcontrôlleur se bloquerai dans une boucle infinie. Il ne faut pas oublier que la pile contient que 8 niveaux d'imbrication.

A4.4.2.2. Timer 1

A). Caractéristiques du timer 1

Le timer1 fonctionne dans son ensemble comme le timer0. La philosophie est semblable, Il y a ce cependant de notables différences. Tout d'abord, ce timer est capable de compter sur 16 bits, à comparer aux 8 bits du timer0. Il sera donc capable de compter de D'0' à D'65535'.

Le timer1 compte donc sur 16 bits, ce qui va nécessiter 2 registres. Ces registres se nomment **TMR1L** et **TMR1H**. Le contenu de TMR1L et de TMR1H n'est pas remis à 0 lors d'un reset, donc, si on veux compter à partir de 0, il nous incombe de remettre à 0 ces 2 registres. Le timer1 permet également, tout comme le timer 0, de générer une interruption une fois le débordement effectué.

Le timer1 dispose, lui aussi, de la possibilité de mettre en service un prédiviseur. Mais ce prédiviseur ne permet qu'une division maximale de 8. Le résultat final est cependant meilleur que pour le timer 0, qui ne pouvait compter que jusque 256, multiplié par un prédiviseur de 256, soit 65536. Pour notre nouveau timer, nous arrivons à une valeur maximale de 65536 multiplié par 8, soit 524288.

Ceci nous permet d'atteindre des temps plus longs, mais ne nous permet pas d'utiliser des temps aussi courts que pour le timer 0, qui peut générer une interruption tous les 256 clocks d'horloge, notre timer 1 nécessitant 65536 clocks au minimum. En fait, il suffit que lors de chaque interruption nous placions la valeur 0xFF dans **TMR1H**, et nous nous retrouvons avec un compteur sur 8 bits. La souplesse du timer 1 est donc plus importante que celle du timer 0.
Tout comme pour le timer 0, les registres de comptage **TMR1L** et **TMR1H** contiennent une valeur indéterminée après un reset de type P.O.R ou B.O.R. De même, un autre reset ne modifie pas le contenu précédent de ces registres, qui demeurent donc inchangés.

B). Le timer1 et les interruptions

Il est bien entendu qu'un des modes privilégiés de l'utilisation des timers, est la possibilité de les utiliser en tant que générateurs d'interruptions. On rappelle que le timer 1 permet de générer une interruption au moment où timer « déborde », c'est-à-dire au moment où sa valeur passe de 0xFFFF à 0x0000 (16 bits).
pour que le timer 1 génère une interruption, il faut tout simplement que cette interruption soit autorisée. Donc :

- Il faut que le bit d'autorisation d'interruption du timer 1 (TMR1IE) soit mis à 1. Ce bit se trouve dans le registre **PIE1** (banque 1).
- Pour que le registre **PIE1** soit actif, il faut que le bit PEIE d'autorisation des interruptions périphériques soit positionné dans le registre **INTCON**.
- Il faut également, bien entendu, que le bit d'autorisation générale des interruption (GIE) soit positionné dans le registre **INTCON**.

Une interruption sera générée à chaque débordement du timer1. Cet événement sera indiqué par le positionnement du flag TMR1IF dans le registre PIR1. Comme d'habitude, il nous incombe de remettre ce flag à 0 dans la routine d'interruption, sous peine de rester

indéfiniment bloqué dans la dite routine. Voici une séquence qui initialise les interruptions sur le timer 1 :

```
bsf     STATUS,RP0      ; passer en banque 1.
bsf     PIE1,TMR1IE     ; autoriser interruptions timer 1.
bcf     STATUS,RP0      ; repasser banque 0.
bsf     INTCON,PEIE     ; interruptions périphériques en service.
bsf     INTCON,GIE      ; interruptions en service.
```

C). Les différents modes de fonctionnement du timer1.

Le timer1 peut, tout comme le timer 0, fonctionner en mode timer (c'est-à-dire en comptage des cycles d'instruction), ou en mode compteur (c'est-à-dire en comptant des impulsions sur une pin externe). Cependant, le timer 1 dispose de 2 modes différents de mode de comptage : un mode de type synchrone et un mode de type asynchrone.

Nous avons déjà eu une idée de ce qu'est un mode synchrone dans la partie traitant du timer0, le comptage ne s'effectuait, en sortie du prédiviseur, qu'au moment de la montée de Q4, donc en synchronisme avec l'horloge interne. En plus, nous avons la possibilité, au niveau du timer1, d'utiliser un quartz sur les pins T1OSI et T1OSO afin de disposer d'une horloge séparée de l'horloge principale.

Un timer utilisé en mode « timer », n'effectue jamais qu'un simple comptage de cycles d'instruction. Pour résumer, nous pouvons donc dire que le timer peut fonctionner en tant que :

- Timer basé sur l'horloge interne (compteur de cycles d'instruction)
- Timer basé sur une horloge auxiliaire (il ne sera pas traiter, pour plus de détails voir **[7]**)
- Compteur synchrone
- Compteur asynchrone.

Les modes de fonctionnement sont définis en configurant correctement le registre T1CON, (Voir annexe sur 16F877).

D). Le timer1 en mode timer

Dans ce mode, nous trouvons un fonctionnement tout ce qu'il y a de plus simple, puisque nous avons affaire à un comptage des cycles d'instructions internes du PIC, exactement comme nous avions pour le timer 0.
Pour choisir ce mode, nous devons configurer **T1CON** de la façon suivante :

T1CON : B'00ab0001'

« ab » représentent la valeur du prédiviseur choisie, et TMR1CS devra être mis à « 0 ». TMR1ON permet de mettre le timer en service. Ceci configure le fonctionnement interne du timer1

Voici un exemple d'initialisation du timer 1, configuré avec un prédiviseur de 4

```
clrf    TMR1L           ; effacer timer 1, 8 LSB
clrf    TMR1H           ; effacer timer 1, 8 MSB
movlw   B'0010000'      ; valeur pour T1CON
movwf   T1CON           ; prédiviseur = 4, mode timer, timer off
bsf     T1CON,TMR1ON    ; mettre timer 1 en service
```

E). Le timer1 en mode compteur synchrone

Pour travailler dans le mode « compteur synchrone », nous devons configurer T1CON de la façon suivante :

T1CON : B'00ab0011'

Pour pouvoir compter des événements sur la pin T1CKI, il faut que cette pin soit configurée en entrée via le registre **TRISC**. Nous entrons donc sur T1CKI (T1 ClocK In) et nous arrivons sur un trigger de Schmitt. Nous arrivons ensuite sur la sélection du signal via TMR1CS. Comme nous avons mis « 1 » pour ce bit, nous validons donc l'entrée de T1CKI et nous ignorons le comptage des instructions internes (mode timer).

Ensuite, nous arrivons au prédiviseur, dont la valeur est fixée par T1CKPS1 et T1CKPS0, on constate qu'à ce niveau, le comptage se fait de façon asynchrone.

les événements sont comptés dans le prédiviseur indépendamment de l'horloge du PIC. Par contre, la sortie du prédiviseur est envoyée également dans un module de synchronisation. Le bloc de sélection T1SYNC, permet de définir si on utilise la sortie du prédiviseur avant la synchronisation (mode asynchrone) ou après la synchronisation (mode synchrone).

Nous avons mis notre T1SYNC à 0. Comme il est actif à l'état bas, ceci nous sélectionne le signal synchronisé. Le signal est ensuite validé ou non grâce à TMR1ON, signal qui va servir à incrémenter le mot de 16 bits constitué de TMR1L et TMR1L.

Si ce mot de 16 bits passe de 0xFFFF à 0x0000 (débordement), le flag TMR1IF est alors positionné, et peut générer une interruption.

Le timer 1 utilisé en mode « compteur synchrone » ne permet pas de réveiller le PIC sur une interruption du timer1 (mode Sleep). C'est logique, car, puisqu'il n'y a pas de comptage, il n'y a pas de débordement.

La présence du synchroniseur retarde la prise en compte de l'événement, puisqu'avec ce dernier la sortie du prédiviseur ne sera transmise au bloc incrémenteur que sur le flanc montant suivant de l'horloge interne.

Il reste à préciser que le comptage s'effectue uniquement sur un flanc montant de T1CKI, donc sur le passage de T1CKI de « 0 » vers « 1 ». Il ne nous est pas possible ici de choisir le sens de transition, comme le permettait le bit TOSE pour le timer 0.

De plus, le flanc montant de T1CKI n'est pris en compte que s'il a été précédé d'au moins un flanc descendant.

F). Le timer 1 en mode compteur asynchrone

Pour mettre le timer1 en mode compteur asynchrone on reprend les mêmes principes précédents et il suffit bien entendu de supprimer le bloc de synchronisation, donc, on voit qu'un événement est comptabilisé immédiatement, sans attendre une quelconque transition de l'horloge interne.

Les contraintes temporelles, concernant les caractéristiques des signaux appliqués sont toutefois toujours d'application (voir partie précédente). La plus grande différence, c'est que le comptage s'effectue même si le PIC est stoppé, et que son horloge interne est arrêtée. Il est donc possible de réveiller le PIC en utilisant le débordement du timer1.

La valeur à placer dans le registre **T1CON** pour travailler dans ce mode est bien évidemment :

T1CON : B'00ab0111'

Il faut de nouveau ne pas oublier de configurer T1CKI (RC0) en entrée, via le bit 0 de TRISC.

A4.4.2.3. Timer 2

Dans cette partie, nous allons étudier le dernier des trois Timers de notre PIC. Celui-ci dispose d'un certain nombre de caractéristiques différentes des 2 autres.

Cette approche de Microchip permet à l'utilisateur d'avoir une multitude des modes d'utilisations possibles, tout en conservant, pour chaque timer, une facilité d'utilisation et une complexité, donc un coût, abordables. La démarche, lorsque on choisi un timer, sera donc fonction de l'utilisation envisagée **[7]**.

A). Caractéristiques du timer 2

Le timer2 est un compteur sur 8 bits, il possède un prédiviseur. Celui-ci peut être paramètré avec une des 3 valeurs suivantes : 1,4, ou 16. Il est donc pauvre à ce niveau.

Cependant, le timer2 dispose également d'un postdiviseur, qui effectue une seconde division après l'unité de comparaison, que nous allons voir. Ce postdiviseur peut prendre n'importe quelle valeur comprise entre 1 et 16, ce qui donne un grand choix possible à ce niveau.

La valeur du diviseur total, vue par l'utilisateur, est bien entendu obtenue en multipliant la valeur du prédiviseur par celle du postdiviseur. Moyennant ceci, il est possible, avec le timer 2, d'obtenir les valeurs de diviseur suivantes :

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,20,24,28,32,36,40,44,48,52,56,60,64,80,96,112, 128, 144,160,176,192,208,224,240,256

Avec ce timer, on dispose d'un large éventail de diviseurs effectifs. Le timer2 incrémente pour sa part le registre **TMR2**, registre unique puisque comptage sur 8 bits.

Les valeurs de division minimale et maximale sont donc identiques à celles du timer 0, qui disposait également d'un comptage sur 8 bits, avec prédiviseur de 1 à 256, le registre TMR2 est remis automatiquement à 0 lors d'un reset, contrairement au timer 1. Ceci peut être un avantage ou un inconvénient, suivant le type de réaction que nous attendons du timer.

Il faut également tenir compte que ce timer ne dispose d'aucune entrée extérieure via une pin du PIC. Il ne peut donc fonctionner qu'en mode « timer » pur.

B). Le timer 2 et les interruptions

Le timer2 fonctionne, à ce niveau, comme le timer1. Le flag d'interruption se nomme TMR2IF, tandis que le bit d'autorisation s'appelle TMR2IE.

La principale différence provient de l'événement qui cause le positionnement de TMR2IF, donc qui cause l'interruption. Tout comme pour le timer 1, il s'agit d'une interruption périphérique, donc, la procédure pour autoriser les interruptions du timer 2 se fera en 3 étapes:

- Autorisation des interruptions périphériques via le bit PEIE du registre **INTCON**
- Autorisation de l'interruption timer 2 via TMR2IE du registre **PIE1**
- Autorisation générale des interruptions via le bit GIE du registre **INTCON**

C). Le timer 2 et les registres PR2 et T2CON

Le principe du timer 2 est différent des deux autres Timers, dans le sens que l'événement détecté n'est pas le débordement « ordinaire » du timer (c'est-à-dire le passage de 0xFF à 0x00), mais le débordement par rapport à une valeur prédéfinie.

Cette valeur étant mémorisée dans le registre **PR2** (banque 1). Nous pouvons donc avoir, par exemple, débordement de 0x56 à 0x00, en plaçant la valeur 0x56 comme valeur maximale dans le registre PR2. On peut donc dire que le fonctionnement du timer est le suivant :

- On incrémente le contenu du prédiviseur à chaque cycle d'instruction
- Chaque fois que ce contenu correspond à un multiple de la valeur du prédiviseur, on incrémente TMR2 (contenu du timer 2)
- Chaque fois que le contenu de TMR2 dépasse le contenu de PR2, on remet TMR2 à 0, et on incrémente le contenu du postdiviseur.
- Chaque fois que le contenu du postdiviseur correspond à un multiple de la valeur du postdiviseur, on positionne le flag TMR2IF.

Il n'y a pour ce timer qu'une seule source de comptage, à savoir l'horloge principale du PIC divisée par 4, autrement dit le compteur d'instructions. Nous sommes donc bien en présence d'un timer « pur ».

Une prédivision est paramètrée par T2CKPS0 et T2CKPS1. La sortie du prédiviseur incrémente le registre **TMR2**. Cette valeur est comparée avec la valeur contenue dans PR2. Chaque fois que les contenus de TMR2 dépasse celle de PR2, la sortie du comparateur incrémente la valeur contenue dans le postdiviseur. Cette sortie effectue également un reset de TMR2, qui redémarre donc à 0.

Chaque fois que le contenu du postdiviseur est égal à un multiple de la valeur de ce celui-ci, paramètrée par TOUTPS0 à TOUTPS3, le flag TMR2IF est forcé à 1, et une interruption est éventuellement générée.

Cet avantage, combiné à la grande flexibilité de l'ensemble prédiviseur/postdiviseur, permet d'obtenir très facilement des durées d'interruption précises sans complications logicielles.

Il reste à donner la formule de la durée séparant 2 positionnements consécutifs du flag TMR2IF :

Durée totale = temps d'une instruction * prédiviseur * postdiviseur * (PR2 +1)
La valeur maximale est donc bien, comme pour le timer 0 de 16*16*256 = 65536.

A4.4.3. Modules CCPx

Les 16F877 disposent de 2 modules CCP. CCP signifions Capture, Compare, and PWM. Ces modules CCP sont fortement liés et dépendant des timers 1 et 2, ils sont également liés au convertisseur A/D.

Sachons déjà que ces modules augmentent les capacités des Timers. Il faut savoir que les 2 modules CCP1 et CCP2 sont strictement identiques, excepté la possibilité, pour le module CCP2, de démarrer automatiquement la conversion A/D. Cependant les modules capture et compare ne seront pas traités ici, on étudiera que le module PWM.

A4.4.3.1. Module PWM "La théorie du MLI appliquée aux PICs"

Nous savons que nous avons besoin de 2 paramètres pour créer notre signal MLI. D'une part, le temps Tc, qui détermine la fréquence (fixe) de notre signal, et d'autre part le rapport cyclique (variable) de ce dernier [***].

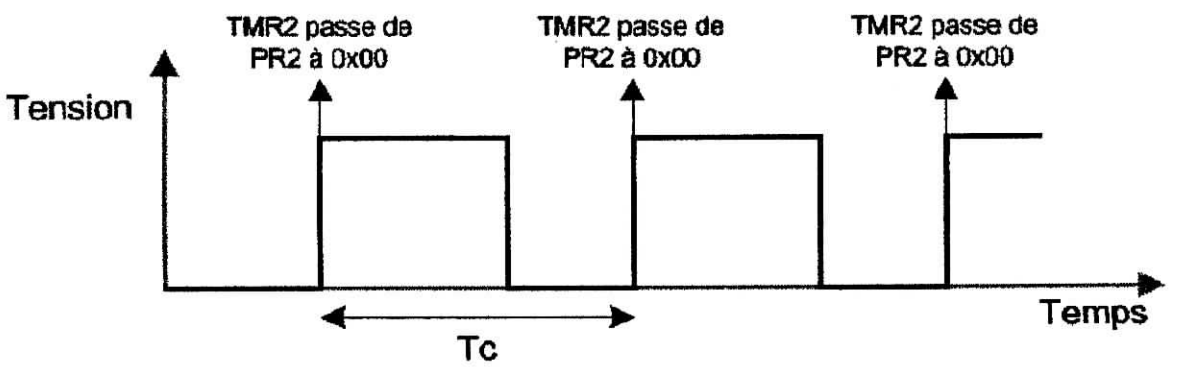
Concernant le rapport cyclique, les PICs influencent plutôt un autre paramètre, c'est-à-dire le temps Th (durée du niveau haut). Les 2 valeurs utilisées dans la programmation seront donc Tc et Th. Nous allons maintenant étudier la façon de gérer ces, à savoir le temps Tc et Th. Le temps Tc est défini tout simplement par le timer 2. On programme éventuellement le prédiviseur, on charge la valeur adéquate dans PR2, et le temps mis par notre TMR2 pour déborder nous donne notre temps TC. Le postdiviseur n'est pas utilisé dans le module PWM. Donc n'intervient pas dans le calcul de Tc. Ceci implique également que nous pouvons utiliser notre timer 2 en tant que générateur pour le module PWM, et, grâce au postdiviseur, travailler avec un autre temps (multiple du premier) dans le reste de notre programme. Nous avons : (Tcy : temps de cycle instruction)

- Temps du cycle utilisé pour le module PWM : Tc = (Tcy * prédiviseur) (PR2 + 1)

Inconvénient : il ne sera pas possible d'obtenir des temps longs avec notre module PWM. Celui-ci est donc prévu pour travailler avec des fréquences importantes. Nous avons déjà parlé du timer 2 dans son emploi classique, ce qui nous intéresse donc ici est la première formule utilisée dans le module PWM. Le module PWM travaille avec le timer 2, et au niveau de Tcy de la façon suivante :

- Vous entrez votre valeur de débordement dans PR2
- A chaque fois que TMR2 déborde de PR2 à 0x00, la pin CCPx passe au niveau 1.

Donc, si nous reprenons un signal PWM quelconque, nous avons déjà.



Nous avons déterminé l'emplacement de montée du signal de « 0 » à « 1 ». Or, comme le temps Tc est le temps séparant 2 montées successives du signal (ou 2 descentes), nous avons défini Tc. Pour rappel :

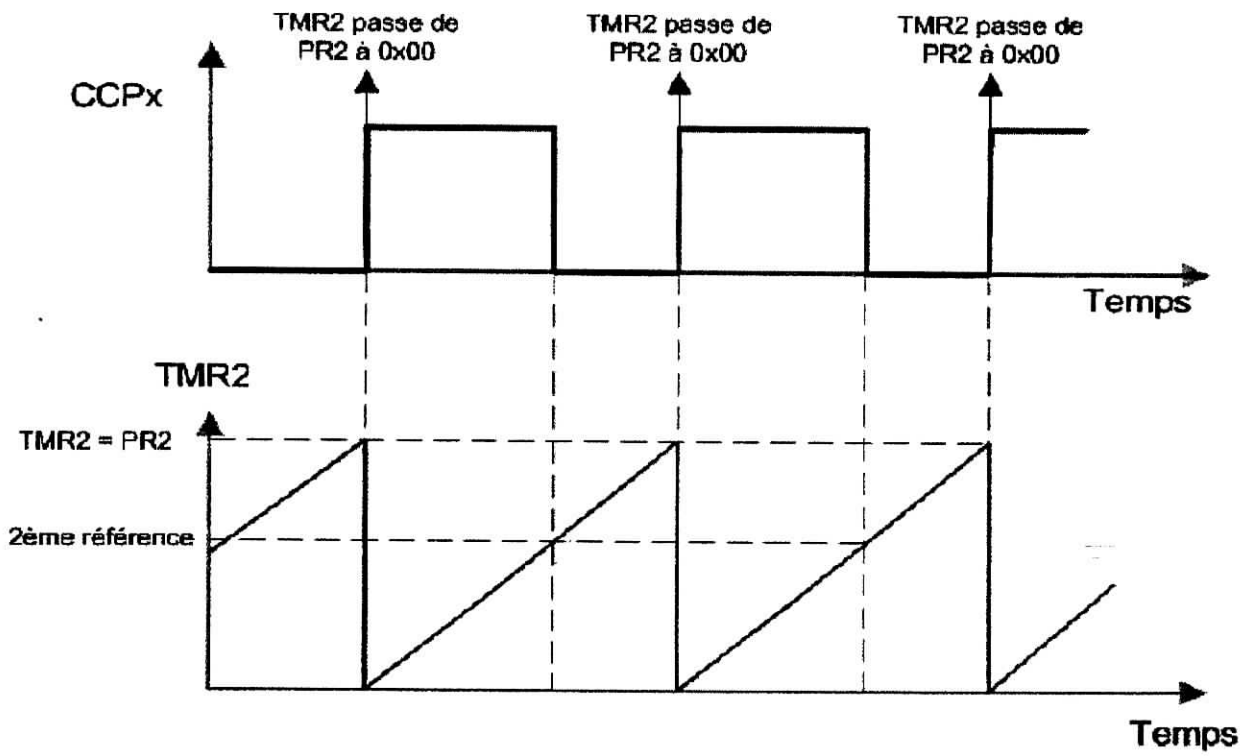
Tc = (PR2 + 1) * Tcy * prédiviseur, ou encore
Tc = (PR2 + 1) * 4 * Tosc * prédiviseur (Tosc, période oscillateur)

En effet, un cycle d'instruction vaut 4 cycles d'oscillateur. Reste donc à définir l'emplacement de redescente de notre signal pour obtenir notre signal complet. Ce qu'il nous faudrait, c'est un deuxième registre « PR2 », qui nous indiquerait, au moment où TMR2 atteindrait cette valeur, que le signal doit retomber à « 0 ». Le principe va donc être le suivant:

Le Timer 2 compte : on suppose que le signal CCP vaut actuellement « 0 » :

- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 repasse à « 0 », CCPx passe à « 1 »
- TMR2 arrive à la seconde valeur de consigne, CCPx passe à « 0 »
- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 = 0, CCPx vaut « 1 »
- TMR2 arrive à la seconde valeur de consigne, CCPx passe à « 0 »
- Retour 1ere étape

Ceci nous donne la représentation suivante :



La valeur inscrite en tant que seconde référence doit être inférieure à celle de (PR2+1), sans quoi TMR2 n'atteindrait jamais cette valeur. Plus cette valeur est faible, moins nous avons le choix des valeurs à inscrire en tant que seconde référence, cette dernière devant

rester inférieure à PR2 et supérieure à 0. Or PR2 dépend du calcul de notre temps Tc. Microchip a adopté une approche révolutionnaire, comme nous ne pouvons pas augmenter la valeur de référence, nous allons simplement y ajouter des « décimales », ceci affinera notre possibilité de réglage, et donc la précision finale obtenue dans les mêmes proportions. Il faut noter que le terme « décimales » est propre en sens "système binaire". Il a été décidé, pour ce PIC, d'ajouter 2 « décimales binaires » à notre compteur TMR2. Ce compteur se présentera donc comme étant de la forme :

$$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0\ ,b^{-1}\ b^{-2}$$

Ceci formant un nombre de 10 bits effectifs, et donc multiplie la précision par 4. Le premier bit après la virgule représente des multiples de ½ (2^{-1}), le second des multiples de ¼ (2^{-2}). Donc :

- Avec un prédiviseur de 1, la précision est de Tcy / 4, soit Tosc
- Avec un prédiviseur de 4, la précision est de Tcy, soit 4 Tosc
- Avec un prédiviseur de 16, la précision est de 4 * Tcy, soit 16 Tosc

Le cycle se déroule donc de la façon suivante :

le Timer 2 compte : supposant que le signal CCPx vaut actuellement « 0 »

- TMR2 (8 bits) arrive à la valeur de PR2 (8 bits)
- Au cycle suivant, TMR2 repasse à « 0 », CCPx passe à « 1 »
- TMR2 + 2 « décimales » atteint la seconde valeur de consigne (8 bits + 2 « décimales »), CCPx passe à « 0 », le timer 2 continue de compter.
- TMR2 (8 bits) arrive à la valeur de PR2 (8 bits)
- Au cycle suivant, TMR2 = 0, CCPx vaut « 1 », et ainsi de suite

Avec un PIC cadencé à 20MHz, et une fréquence du signal PWM de 20 Khz :

Tc = 1/20 KHz = 50 µs
Tc = (PR2 + 1) * Prédiviseur * 4 * Tosc
Tosc = 1/20MHz = 50 ns
PR2 = (TC / (prédiviseur * 4 * Tosc) – 1
PR2 = (50 µs / 200 ns) – 1 = 250 – 1 = 249 11111001

Il sera alors possible d'ajuster notre rapport cyclique sur des valeurs comprises entre B'00000000,00' et B '11111001,11 », soit 999 valeurs différentes, donc une précision sur 10 bits, ou encore une marge d'erreur de 0,1%.

En somme, la précision vaut 1 / ((PR2+1) * 4), et est donc multipliée par 4 par rapport à une comparaison sans utiliser de fractions (sur 8 bits).

Comme il faut intervenir sur des valeurs de consigne de 10 bits, l'écriture ne pourra pas se faire en une seule fois, et donc pourra provoquer une valeur temporaire indésirable. Ce phénomène est appelé « glitch ».

De ce faite Microchip utilise un registre intermédiaire qui servira de valeur de comparaison, et qui sera chargé au moment du débordement de TMR2.
La procédure exacte est donc la suivante :

- Le Timer 2 compte : on imagine que le signal CCP vaut actuellement « 0 »

- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 repasse à « 0 », CCPx passe à « 1 »
- En même temps, la valeur programmée comme consigne par l'utilisateur est copiée dans le registre final de comparaison.
- TMR2 arrive à la valeur de la copie de la seconde valeur de consigne, CCPx passe à « 0 »
- TMR2 arrive à la valeur de PR2
- Au cycle suivant, TMR2 = 0, CCPx vaut « 1 », et ainsi de suite

La durée de Th ne sera effective qu'à partir du cycle suivant celui actuellement en cours.

La formule qui lie les 10 bits de notre second comparateur, sachant que ces 10 bits (COMPAR) expriment un multiple de quarts de cycles d'instructions, donc un multiple de temps d'oscillateur, avec le temps du signal à l'état haut (Th).

Th = COMPAR * prédiviseur * Tosc.
COMPAR = Th / (prédiviseur * Tosc)

On peut également dire, en fonction du rapport cyclique (Rc), que Th = Tc * Rc
Il faut encore rappeler que pour comparer TMR2 avec COMPAR codé sur 10 bits, il faut que TMR2 soit également codé sur 10 bits. Il faut donc compléter TMR2 avec 2 bits qui représentent les quarts de cycle d'instruction, c'est ce qu'il se fait en interne de façon automatique.

A4.4.3.2. Les registres utilisés

Nous avons besoin de plusieurs registres pour programmer toutes ces valeurs. En fait, nous les avons déjà tous rencontrés, ne reste donc qu'à expliquer leur rôle dans ce mode particulier. Nous avons besoin d'une valeur de débordement pour notre timer 2, cette valeur se trouve, comme je l'ai déjà dit dans le registre **PR2**. C'est donc une valeur sur 8 bits. La valeur de la seconde comparaison (celle qui fait passer la sortie de 1 à 0) est une valeur de 8 bits complétée de 2 bits fractionnaires. Le nombre entier sera inscrit dans le registre **CCPRxL**. Les 2 bits fractionnaires qui complètent ce nombre sont les bits DCxB1 et DCxB0 du registre **CCPxCON**. Comme nous l'avons vu, ce nombre de 10 bits sera copié en interne par le PIC vers un autre registre, qui sera le registre **CCPRxH** complété de 2 bits internes non accessibles. Notez qu'en mode « PWM », il vous est impossible d'écrire dans ce registre.

Pour lancer le mode « PWM », nous devons donc procéder aux initialisations suivantes :

1) On initialise PR2 : **PR2** = (TC / (prédiviseur * 4 * Tosc) – 1
2) On calcule la valeur de comparaison DCB en valeur fractionnaire suivant la formule :
DCB = Th / (prédiviseur * Tosc). On place les bits 9 à 2 dans **CCPRxL** (valeur entière), les bits 1 et 0 (fraction) étant positionnés dans DCxB1 et DCxB0 du registre **CCPxCON**
3) On place la pin CCPx en sortie en configurant **TRISC**
4) On lance le timer 2 en programmant son prédiviseur
5) On configure **CCPxCON** pour travailler en mode « PWM ».

ANNEXE 5

MODULES UTILISES POUR LA MISE EN ŒUVRE PRATIQUE

A5.1. Présentation du L298

Le L298 est un circuit intégré monolithique à 15 pins qui existe en boîtier Multiwatt et PowerSO20 **[14]**. C'est un composant a haut débit de courant avec des tensions élevées, il renferme deux modules pont H et accepte des entrées de commande à niveau logique TTL, il peut piloter des charges inductives, relais, solénoïdes, moteur DC et pas à pas, il a des entrées Enable qui permettent de mettre en service ou non le pont H associer, il a deux sorties à connecter avec une résistance shunte pour relever la valeur du courant et aussi une entrée d'alimentation basse tension supplémentaire pour les portes logiques de commande.

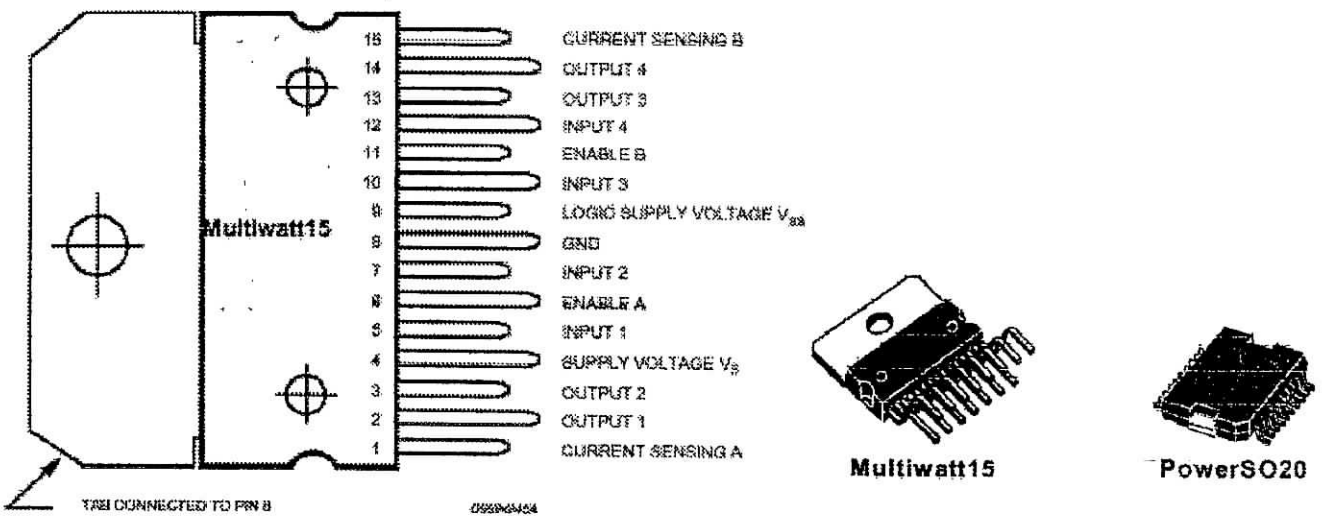


Figure A5.1. *Définition des pins et boîtier externe.*

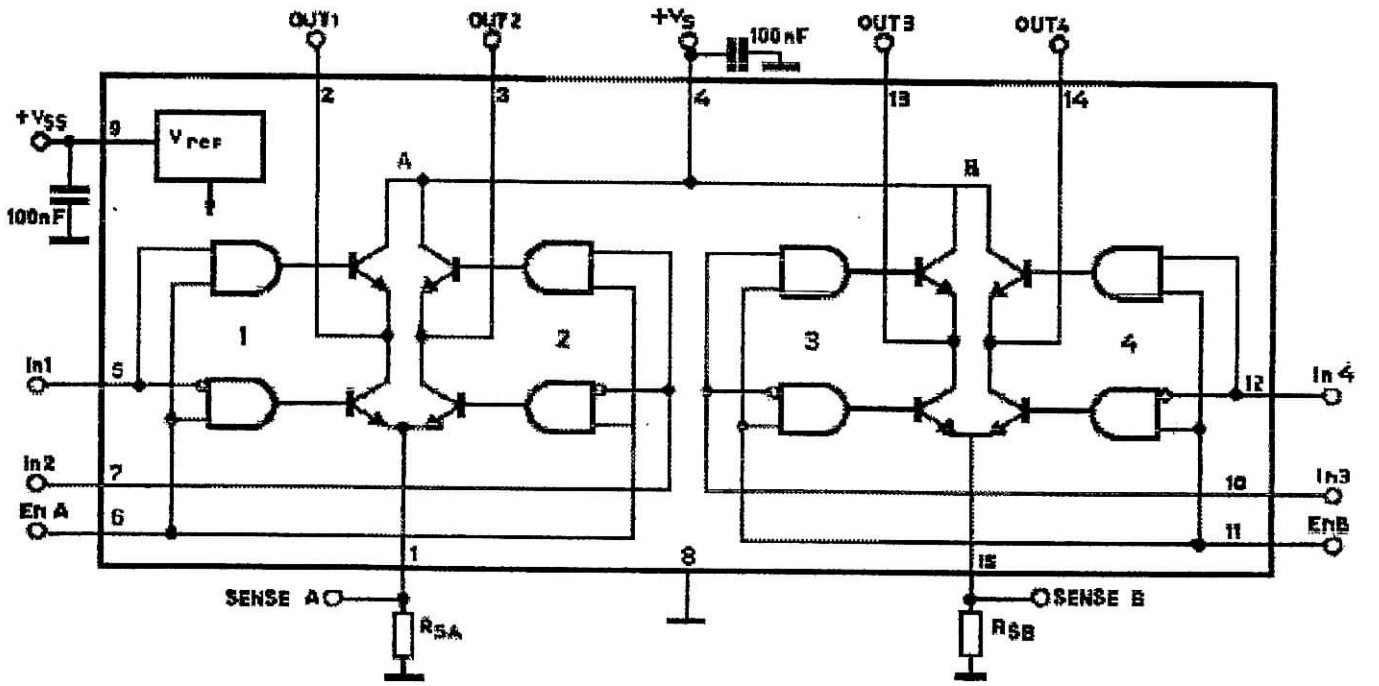


Figure A5.2. *Schéma interne et conditionnement externe des pins.*

Symbole	Paramètre	Value	Unit
V_s	Tension d'application	50	V
V_{ss}	Tension logique	7	V
V_I , V_{en}	Tension d'entrée et de validation	–0.3 à 7	V
I_O	Courant de sortie (pique pour chaque canal) – Non répétitive (t = 100µs) – Répétitive (80% on – 20% off ; t_{on} = 10ms) – DC opérations	 3 2.5 2	 A A A
V_{sens}	Tension relative au courant (courent sense A, B)	–1 à 2.3	V
P_{tot}	Dissipation totale d'énergie (T_{case} = 75°C)	25	W
T_{OP}	Température de fonctionnement des jonctions	–25 à 130	°C
T_{stg} , T_J	Température de jonction et de stockage	–40 à 150	°C

Tableau A5.1. *caractéristiques et valeurs maximales absolues.*

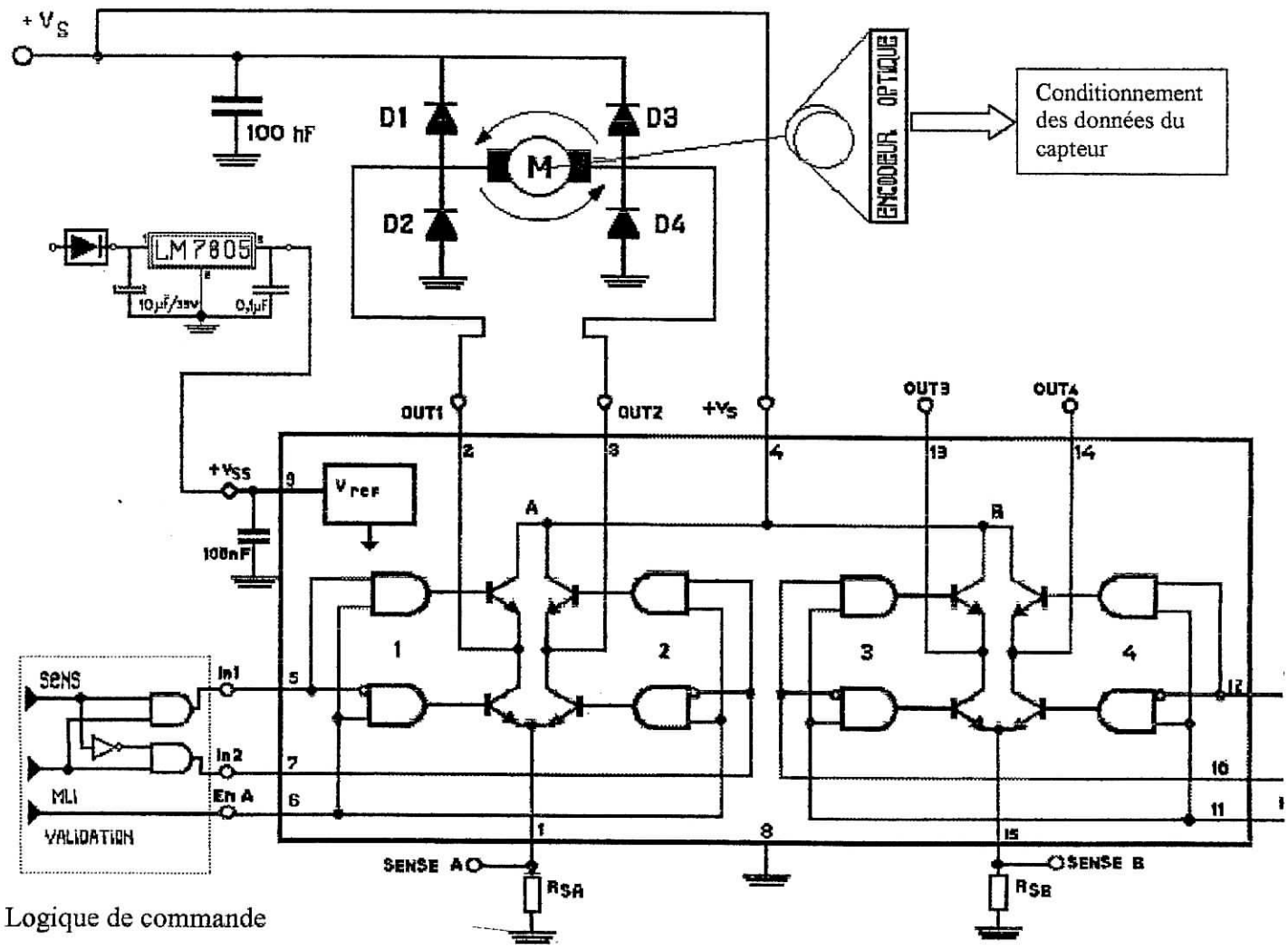


Figure A5.3. *La partie puissance " Gestion du moteur DC ".*

Le circuit de puissance doit contenir tout les éléments nécessaire a la commande des ponts H permettant la gestion du sens et transmettre le signale MLI de façon a faire marcher le moteur en deux phases, une phase active et une phase de roue libre, cette dernière exige 4 diodes de roue libre externe type 1N4006, ainsi que deux condensateurs de filtrage pour LM7805 et deux autres condensateurs pour prévenir des gradients de tension d'alimentation.

A5.6. Circuit imprimé

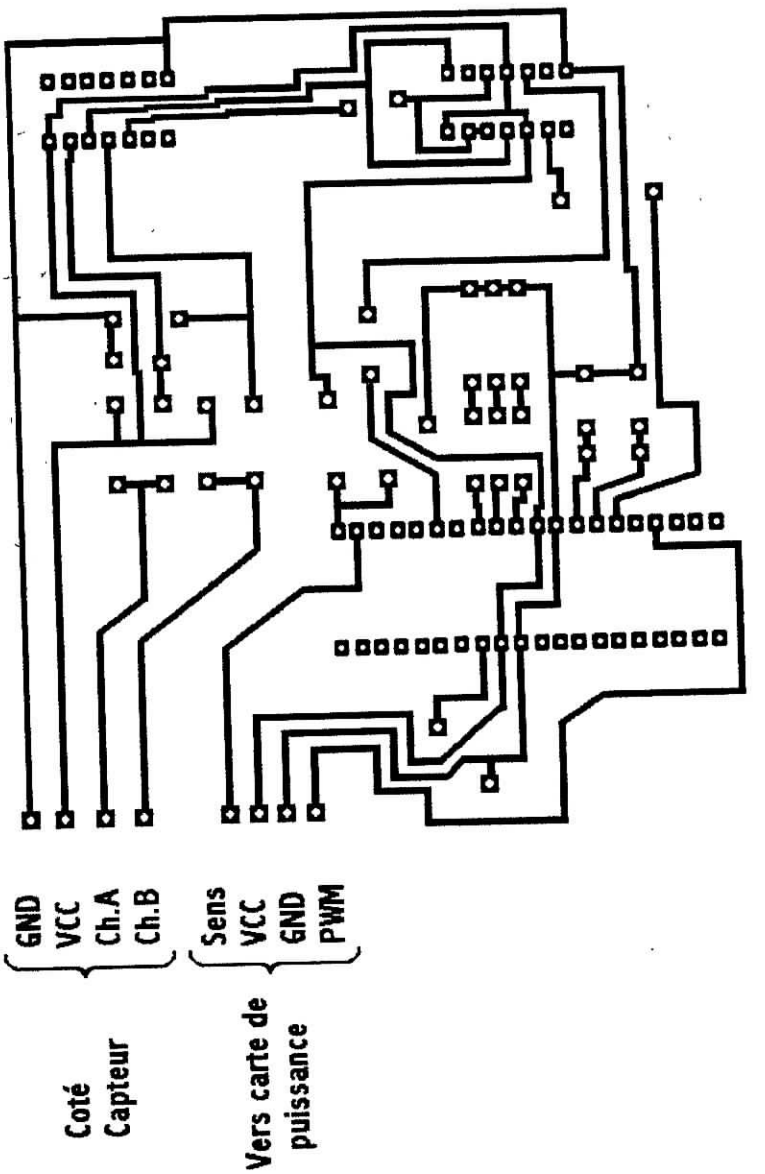


Figure A5.10. *Circuit imprimé de la carte de commande.*