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

COMMANDE PAR INTERFACE UMAC-DELTATAU D'UN BRAS MANIPULATEUR DE TYPE SCARA

Proposé et dirigé par :
M^r O.STIHI

Etudié par :
M^r BENFARES Akli
M^r IRBAH Abdenacer

Promotion : Juin 2006

ENP 10, Avenue Hassen-Badi, EL-HARRACH, ALGER

Remerciements

Ce travail a été réalisé au sein du laboratoire de Commande des Processus (LCP) de l'Ecole Nationale Polytechnique d'Alger sous la direction de

Monsieur Omar STIHI

Nous tenons à le remercier vivement pour nous avoir encadrés et dirigés durant l'élaboration de ce travail ainsi que pour son assistance et tous ses conseils.

Nous tenons aussi à remercier Monsieur CHEKIREB pour ces éclaircissements, et ses conseils fructueux.

Nous remercions les membres du jury, d'avoir accepté l'examination de notre travail.
Nous remerciements vont aussi a Monsieur MECHELIH Hachemi, qui a mis à notre disposition l'interface, objet de notre travail.

Nous remercions également tous les professeurs de la spécialité Automatique de l'ENP, pour leur aide, leurs conseils et leur compréhension durant notre cursus.

Enfin, Nous exprimons notre gratitude à toutes les personnes ayant contribuée de prés ou de loin dans ce travail.

Dédicace

Je dédie ce modeste travail
A ceux qui ont veillé mes nuits
Et qui ont fait de mes rêves les leurs
A mes parents
Que Dieu les garde
Et leur donne longue vie

A mes frères et sœurs
Samir, Elhoues, Adel
Lamia, Nouria, et à ma préférée Miha
Qui m ont soutenus

A mes grands parents ; qui m´ont tant donné
A mon oncle Rachid et sa femme Zouba

A la mémoire de mon cousin Halim
Que Dieu le garde dans son vaste paradis

A mes amis, Hakim, Fateh, Nadir, Omar, Malek, Moh, Idir, Youcef, Cherif, Mounir, Nouar , et spécialement Massine

A mon binôme avec qui j´ai partagé tant de moments

A toutes la promotion Automatique chacun en son nom

Nacer

Dédicace

Je dédie ce modeste travail à mes chers parents qui ont tant souffert pour moi durant toute ces longues années d'études

A mes freres Ali,Moh et sœurs Malika , Drifa, Rachida, Noura, et ma niece Lina sans oublier tous mes amis Chérif, Hakim, youcef ,mon binôme et toute la promotion 2006 automatique

Akli

TABLE DES MATIERES :

CHAPITRE :

III

CHAPITRE :

IV

CHAPITRE :

V

CHAPITRE :

VI

LISTE DES FIGURES :

LISTE DES TABLEAUX :

INDEX :

AMP2	Carte amplificatrice
AUTO TUNING	Auto détermination des paramètres du régulateur
CIRCLE1	Interpolation circulaire dans le sens trigonométrique
CIRCLE2	Interpolation circulaire dans le sens horaire
D-Code	Langage de machines à outil
Delta Tau	fabriquant de l'interface
DSP56303	Digital signal processor de Freescale (Motorola)
DSPGATE	Carte interface entre le moteur et le CPU
G-code	Langage de machines à outil
IOGATE	Input Output GATE
LINEAR	Interpolation linéaire
LMD18200	Pont en format H de NATIONAL SEMICONDUCTOR.
M	Variable de travail
NORMAL	Plan pour l'interpolation circulaire
NOTCH FILTER	Filtre rejecteur
P	Variable de travail
PD	Régulateur Proportionnel Dérivé
PID	Régulateur Proportionnel Intégrale Dérivé
PMAC	Programmable Multi Axes Controller
PMAC EXECUTIF PROGRAMME	Logiciel de l'interface
PMAC TUNING PRO	Logiciel d'auto tuning fourni par Delta Tau
PVT	Position Vitesse Temps
PWM	Pulse Width Modulation (Modulation de largeur d'impulsion)
Q	Variable de travail
SCARA	Selective Compliant Assembly Robotic Arm
SPLINE1	B-Spline cubique uniforme non rationnelle
SPLINE2	B-Spline cubique non rationnelle et non uniforme
TA	Temps d'accélération
T-code	Langage de machines à outil
TM	Temps de mouvement
TS	Temps de courbure
Turbo PMAC2 3U CPU	Unité centrale de traitement de l'interface
UMAC	Universal Motion and Automation Controller
UBUS	UMAC Bus

INTRODUCTION GENERALE :

La robotique est un domaine de technologie moderne au-delà des frontières traditionnelles de l'ingénierie. Comprendre la complexité d'un robot et de ses applications nécessite des connaissances dans diverses disciplines de l'ingénierie.

Toutefois, la commande de robots est devenus une partie importante et intégrale de la technologie robotique liée à d'autres disciplines de la technologie tel que le l'automatique, l'électronique, la mécanique et l'informatique. D'autre part l'industrie continue à introduire des systèmes de commande pour les machines robotiques plus sophistiqués dont l'application devient de plus en plus large.

Le développement technologique dans les domaines du software et du hardware ces dernières années, a donné naissance à de nouvelles générations de contrôleurs de mouvement puissants et flexibles permettant d'un coté l'amélioration de la commande des moteurs grâce à leur vitesse de traitement de l'information et une meilleur interaction homme machine grâce à leur interfaces utilisateurs graphiques et intuitives.

Plusieurs firmes spécialisées dans ce genre de contrôleurs sont alors apparues entre autre **DELTA TAU** [1] proposant une variété de leur produit parmi elles l'interface Turbo UMAC objet de notre étude :

Cette interface est autonome et programmable à partir d'un PC par une liaison série, elle contrôle jusqu'à huit moteurs simultanément et peut être étendue jusqu'à 32 moteurs, elle fournie une grande flexibilité et un large champ d'utilisation technologique.

En vue d'exploiter les capacités de l'interface UMAC pour commander un bras manipulateur de type SCARA, notre travail est porté sur «COMMANDE PAR L'INTERFACE UMAC D'UN MANIPULATEUR DE TYPE SCARA ».

Ce mémoire est organisé en 6 chapitres :

Chapitre1 : Pour mieux cerner les différents aspects de notre travail, des généralités sur la robotique sont énoncées dans ce chapitre, qui contient trois parties essentielles à savoir:

- la génération du mouvement où on présente les méthodes de génération de mouvement dans les espaces articulaire et cartésien ainsi qu'une étude comparative entre les deux.
- la commande en robotique où en présente l'approche décentralisée de type PD et PID ainsi que les méthodes utilisées pour la détermination des paramètres
- présentation du robot SCARA et les dynamiques qui le régissent.

Chapitre 2 : le but de ce chapitre est de présenter l'interface UMAC et ses constituants afin de pouvoir étudier son fonctionnement et sa programmation qui est l'objet du chapitre suivant.

Chapitre3 : dans ce chapitre est étudier les différents types de mouvement que l'UMAC est susceptible de planifier ainsi que les interpolations qui lui y sont associées,et aussi certains aspect qui ont un lien avec leur programmation et leur mise en exécution sont exposés en deux partie a savoir :

- les programmes de mouvement : qui énonce les différentes étapes et définitions pour la mise en œuvre d'un programme de mouvement
- les systèmes en coordination : dont la détermination et la configuration est indispensable pour l'exécution d'un programme de mouvement

Chapitre4 : afin de mieux connaître l'aspect commande de la carte UMAC (qui est l'un des objets de notre travail), une étude succincte de la commande implémentée est faite dans ce chapitre. En effet les différents composants du régulateur implémenté (PID + feed forward), ainsi que le module d'exécution de la commande (Amplificateur), y sont présentés

Chapitre5 : ce chapitre portera sur la manipulation des calcules géométriques directe et inverse par l'interface et leur implémentation, pour pouvoir planifier des trajectoire directement dans l'espace opérationnel (cartésien)

Chapitre6 : dans ce dernier chapitre nous mettrons en application l'interface pour commander un bras manipulateur de type SCARA en planifiant des trajectoires dans les espaces articulaire et cartésien.

CHAPITRE :

I

I- NOTION EN ROBOTIQUE

I-I- Génération de mouvement :

La génération des trajectoires est une étape très importante dans la commande des robots manipulateurs, elle consiste à calculer les consignes de référence en position, en vitesse et en accélération, lesquelles décrivent le mouvement désiré du robot.

Donc la trajectoire est l'évolution de la position et ses dérivées temporelles en fonction du temps pour chacune des articulations.

Le mouvement le plus simple est d'aller d'un point initial à un point final, ainsi le robot est commandé de changer sa position initiale vers une position finale.

Ce type de mouvement convient aux taches de transfert d'objets quand l'espace de travail ne comporte aucun obstacle. Pour certaines raisons telles qu'éviter les obstacles, le chemin à suivre par l'élément terminal peut être contraint par l'addition de points intermédiaires à la position initiale et finale.

Pour assurer le fonctionnement normal du mécanisme on choisira des mouvements continus, pour cela on définit une fonction lisse (dérivée première et secondaire continues) afin d'éviter les risques d'usure et de vibrations pouvant exciter les modes propres du manipulateur.

En résumé, le parcours peut être planifié de différentes manières dont on distingue **[2]:**

- Le mouvement entre deux points avec trajectoire libre entre les points
- Le mouvement entre deux points via des points intermédiaires, spécifiés notamment pour éviter les obstacles, avec trajectoire 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.

La génération de mouvement dans l'espace articulaire et dans l'espace opérationnel est schématisée sur les **Figures I.1** et **I.2**

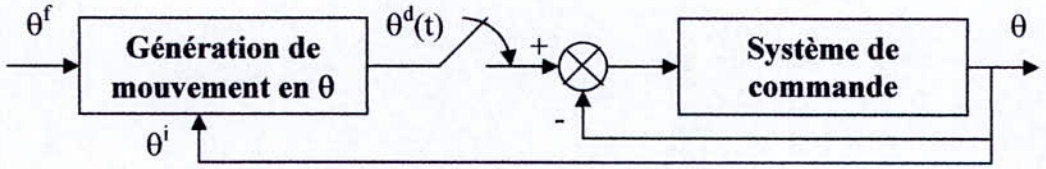


Figure I-1 : *Génération de mouvement dans l'espace articulaire*

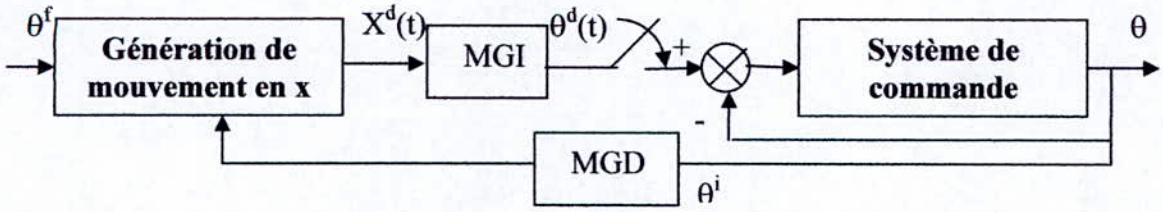


Figure I-2: *Génération de mouvement dans l'espace opérationnel*

La génération de mouvement dans l'espace articulaire présente plusieurs avantages :

- Elle nécessite moins de calcul en ligne, 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.

I.I.1- Génération de mouvement entre deux points dans l'espace articulaire: [2]

On considère un robot à n degrés de liberté. Soit θ_i et θ_f les vecteurs des coordonnées articulaires correspondant aux positions initiale et finale. On désigne respectivement par K_v et K_a les vecteurs des vitesses et accélérations articulaires maximales, le mouvement entre θ_i et θ_f en fonction du temps t est décrit par l'équation suivante :

$$\theta(t) = \theta_i + r(t)D \quad \textbf{Pour} \quad 0 \le t \le t_f \qquad \textbf{(I-1)}$$

$$\dot{\theta}(t) = \dot{r}(t)D \quad \textbf{Avec} \quad D = \theta_f - \theta_i \qquad \textbf{(I-1)}$$

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

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

Donc l'expression **(I.1)** s'écrit aussi :

$$\theta(t) = \theta_f(t) - [1 - r(t)]D \qquad \textbf{(I-2)}$$

Plusieurs fonctions permettent de satisfaire le passage par $\theta_i(t)$ à $\theta_f(t)$ (l'interpolation polynomiale, la loi **Bang -Bang** et la loi **Bang -Bang** avec palier de vitesse **(loi trapèze),** interpolation polynomiale d'ordre trois, ou cinq.

I-I-1-1 La fonction polynomiale cubique :

Il s'agit d'une interpolation simple en effectuant un simple mouvement continu, au moins quatre contraintes sur $\theta(t)$ sont évidentes pour avoir une fonction polynomiale cubique de la forme :

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad \textbf{(I-3)}$$

Ainsi on obtient la vitesse articulaire : $\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$ **(I-4)**

Et aussi l'accélération articulaire : $\ddot{\theta}(t) = 2a_2 + 6a_3 t$ **(I-5)**

Sachant qu'un polynôme de 3ème degré, admet quatre coefficients, il peut être donc construit à partir de quatre contraintes, deux sont obtenues à partir du choix des valeurs initiales et finales de la position :

$\theta(0) = \theta_0$, $\theta_d(t_f) = \theta_f$ quant aux deux autres contraintes, elles proviennent du fait que l'articulation démarre et arrive avec une vitesse nulle : $\dot{\theta}(0) = 0, \dot{\theta}(t_f) = 0$ Et en combinant les deux fonctions :

$\theta(t)$ et $\dot{\theta}(t)$ avec les quatre contraintes, on obtient quatre équations à quatre inconnues :

$$\theta_0 = a_0 \; ; \quad \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \; ; \quad 0 = a_1 \; ; \quad 0 = a_1 + 2a_2 t_f + 6a_3 t_f^2$$

En résolvant ce système d'équations on obtient :

$$a_0 = \theta_0 \; ; \; a_1 = 0 \; ; a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) = \frac{3}{t_f^2} D \; ; \; a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) = -\frac{2}{t_f^3} D$$

Avec ces quatre coefficients, on peut calculer le polynôme cubique qui connecte n'importe quelle position désirée

Donc la fonction de position peut se mettre sous la forme :

$$q(t) = q^f(t) - [1 - r(t)]D \quad \textbf{(I-6)}$$

Avec ;

$$r(t) = 3\left(\frac{t}{t_f}\right)^2 - 2\left(\frac{t}{tf}\right)^3 \quad \textbf{(I-7)}$$

.

I-I-1-2 Interpolation polynomiale (degrés cinq) [2]:

Pour la continuité des accélérations on utilise un polynôme d'interpolation de degré cinq avec les contraintes supplémentaires **:**

$$r(t) = [\theta_i(t) + 10(\frac{t}{t_f})^3 - 15(\frac{t}{t_f})^4 + 6(\frac{t}{t_f})^5]D \quad \textbf{(I-8)}$$

La vitesse et l'accélération maximale sont données par l'équation :

$$\dot{\theta}\max = \frac{15|D|}{8t_f} \quad , \quad \ddot{\theta}\max = \sqrt{\frac{10|D|}{\sqrt{3}t_f{}^2}}$$

A partir de l'expression de la vitesse et de l'accélération maximale, on déduit le temps minimum t_f qu'il faut à une articulation *i* pour réaliser un déplacement donné.

$$t_f = MAX\left[\frac{15|D|}{8Kv}, \sqrt{\frac{10|D|}{\sqrt{3}Ka}}\right] \quad \textbf{(I-9)}$$

L'évolution de la position, vitesse et accélération pour l'articulation j est présentée dans les figures : **Figure I-3, Figure I-4** et **Figure I-5** respectivement : Les simulations sont faites en considérant le facteur temps minimale :

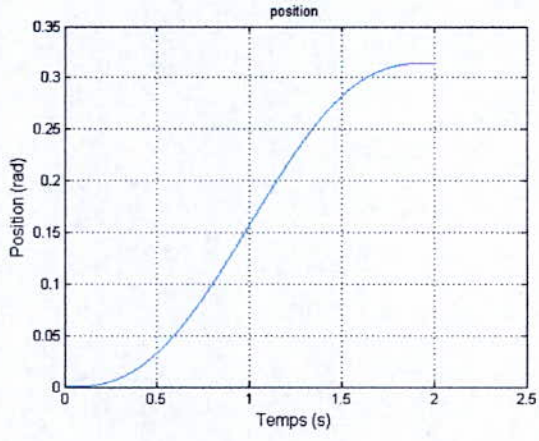


Figure I-3: *Evolution de la position.*

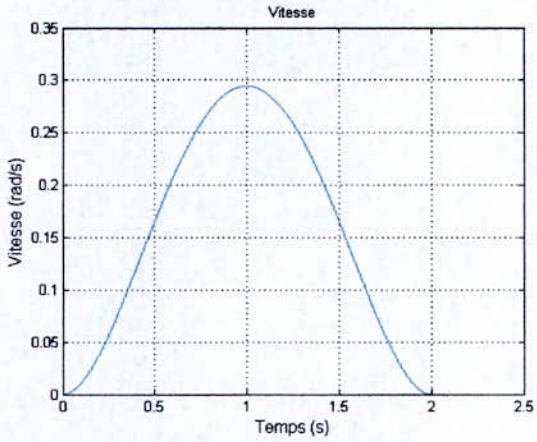


Figure I-4: *Evolution de la vitesse.*

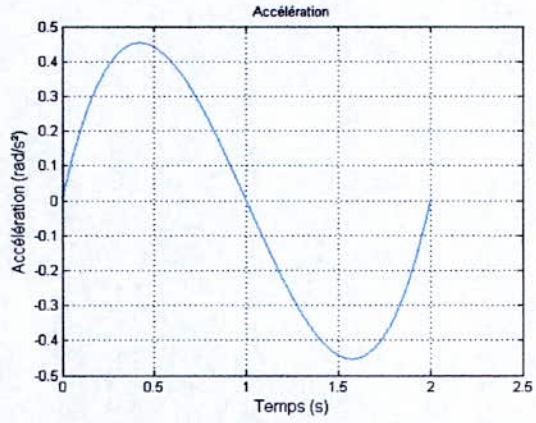


Figure I-5 : *Evolution de l'accélération.*

I-I-1-3 Loi Bang – Bang :

- **Loi Bang – Bang :**

Le mouvement est représenté dans ce cas par une phase d'accélération constante jusqu'à $\frac{t_f}{2}$ puis par une phase de freinage constante. Les vitesses de départ et d'arrivée sont nulles. Le mouvement est donc continu en position et en vitesse, mais discontinu en accélération. La position est donnée par :

$$\begin{cases} \theta_j(t) = [\theta_i + 2(\frac{t}{t_f})^2]D & \text{Pour } 0 \leq t \leq \frac{t_f}{2} \\ \theta_j(t) = \theta_i[-1 + 4(\frac{t}{t_f}) - 2(\frac{t}{t_f})^2]D & \text{Pour } \frac{t_f}{2} \leq t \leq t_f \end{cases} \quad \textbf{(I-10)}$$

Pour une articulation donnée la vitesse et l'accélération maximale sont données par :

$$\dot{\theta}\max = \frac{2|D|}{t_f} \quad , \quad \ddot{\theta}\max = \frac{4|D|}{t_f^{\,2}}$$

- **Loi Bang - Bang avec palier de vitesse (loi trapèze):**

Lorsque la vitesse est saturée, le fait de rajouter un palier de vitesse permet de saturer aussi l'accélération, Le mouvement de l'articulation j est représenté par les relations suivantes :

$$\begin{cases} \theta_j(t)=[\theta_i+(\frac{1}{2}t^2)D_a]D & \text{Pour} \quad 0\le t \le \tau \\ \theta_j(t)=[\theta_i+(t-\frac{\tau}{2})D_v]D & \text{Pour} \quad \tau \le t \le t_f-\tau \\ \theta_j(t)=[\theta_f+\frac{1}{2}(t_f-t)^2D_a]D & \text{Pour} \quad t_f-\tau \le t \le t_f \end{cases} \qquad \textbf{(I-11)}$$

Avec : D_v et D_a les valeurs de saturation de vitesse et d'accélération.

La loi trapèze est la loi optimale en temps celles qui assurent la continuité en position et en vitesse ;

On sait que : $\dot{\theta}\max = \frac{2|D|}{t_f}$, $\ddot{\theta}\max = \frac{4|D|}{t_f^{\,2}}$ **(I-12)**

On peut déduire la condition d'existence du palier de vitesse sur l'articulation j :

$$|D_j| > \frac{K_{vj}^2}{K_{aj}} \qquad \textbf{Avec} \qquad K_v = \sqrt{\frac{|D_j|K_{aj}}{2}} \qquad \textbf{(I-13)}$$

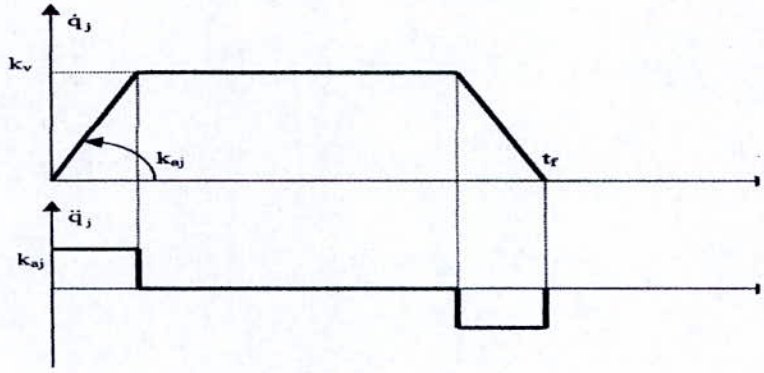


Figure I-6: *Loi trapèze (en vitesse).*

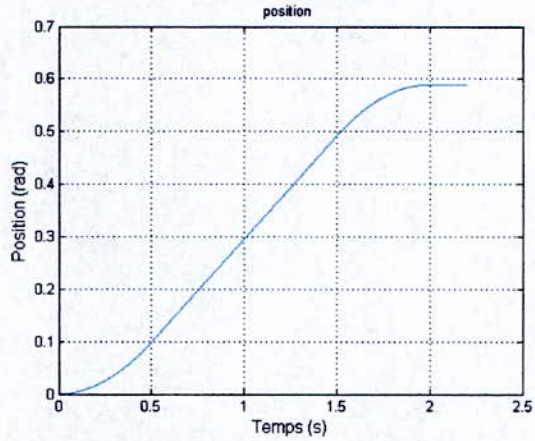


Figure I-7: *Evolution de la position.*

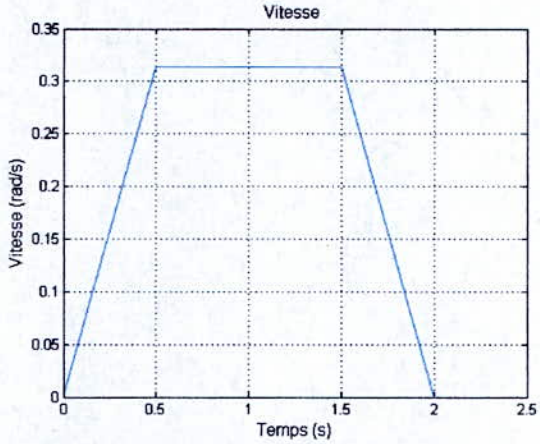


Figure I-8: *Evolution de la vitesse.*

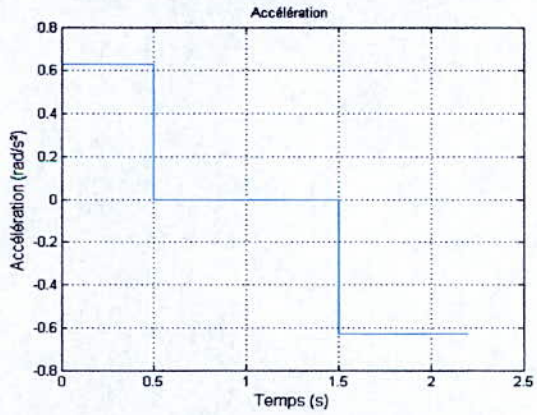


Figure I-9: *Evolution de l'accélération.*

I.I.2- Génération de la trajectoire dans l'espace cartésien :

Les trajectoires dans l'espace articulaire sont plus faciles à calculer et elles sont plus simples, cependant elles deviennent complexes si elles sont décrites dans l'espace cartésien, puisque

dans le premier cas on ne fait aucune correspondance continue entre l'espace articulaire et l'espace cartésien il n'y a aucun problème avec les singularités du mécanisme.

La trajectoire dans l'espace cartésien donne comme résultat la position, la vitesse et l'accélération cartésiennes ($x, \dot{x}, \ddot{x}$), pour certaines structures de commande spécifiques, ces données doivent être converties en grandeurs équivalentes dans l'espace articulaire, une analyse complète devrait utiliser la géométrie inverse, l'inverse du jacobien et la dérivée du jacobien pour pouvoir calculer les angles, les vitesses et les l'accélérations articulaires ($\theta_d, \dot{\theta}_d, \ddot{\theta}_d$) qui vont être utilisées comme un signal de référence pour les régulateurs synthétisés dans l'espace articulaire.

Dans l'espace de travail, la forme du parcours prise par l'élément terminal n'est pas simple, mais plutôt une forme compliquée qui dépend de la géométrie du manipulateur, les formes du parcours sont décrites en terme des fonctions du temps qui déterminent la position et l'orientation cartésiennes, la forme spatiale du parcours entre ses points peut être : une ligne droite, circulaire, sinusoïdale, hélicoïdale ou d'autres formes.

Pour la planification de la trajectoire dans l'espace cartésien, les fonctions qui forment cette trajectoire sont des fonctions du temps qui représentent les variables cartésiennes. Ces parcours peuvent être planifiés directement à partir de la définition, par l'utilisateur, des points de parcours qui sont les spécifications sur ce parcours désiré.

I.I.3- Les problèmes géométriques dans un parcours cartésien [2]:

Plusieurs problèmes sont posés dans la génération des trajectoires cartésiennes à cause de la correspondance continue qui se trouve entre la forme de la trajectoire décrite dans l'espace cartésien et celle décrite dans l'espace articulaire. Nous citons trois problèmes majeurs :

I-I-3-1 Problème 1 :

C'est le problème des points intermédiaires inaccessibles, comme est présenté dans la **Figure I-10**, si on veut que le robot démarre du point A pour arriver au point B en passent par une trajectoire désirée qui est une ligne droite, malgré que le point initial et final du parcours sont compris dans l'espace de travail du robot, il y a certains points qui appartiennent à cette trajectoire et qui n'appartiennent pas à l'espace de travail du manipulateur. Ces points deviennent inaccessibles, ce qui rend impossible d'effectuer cette tache dans l'espace opérationnel. Néanmoins, il n'y aurait pas ce type de problème dans l'espace articulaire.

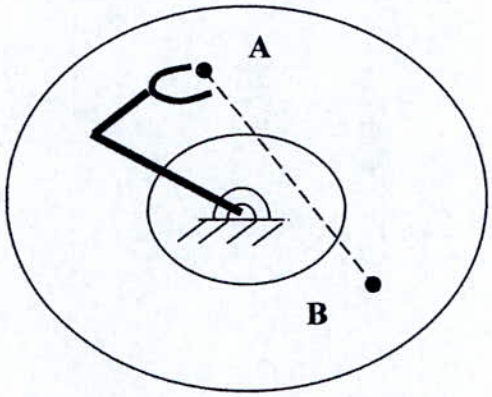


Figure I-10 : *Cas du problème 1*

I-I-3-2 Problème 2 :

C'est le problème de la vitesse articulaire qui est trop élevée en passant près de la singularité. Certains parcours dans l'espace cartésien sont impossibles à exécuter par le robot manipulateur surtout dans le cas où il doit suivre une trajectoire en s'approchant d'une configuration singulière du mécanisme, (par exemple une ligne). Une ou plusieurs vitesses articulaires peuvent augmenter ver l'infini **Figure I-11**. Puisque les vitesses des articulations sont bornées supérieurement, cette situation a pour conséquence la déviation du robot manipulateur de son parcours désiré.

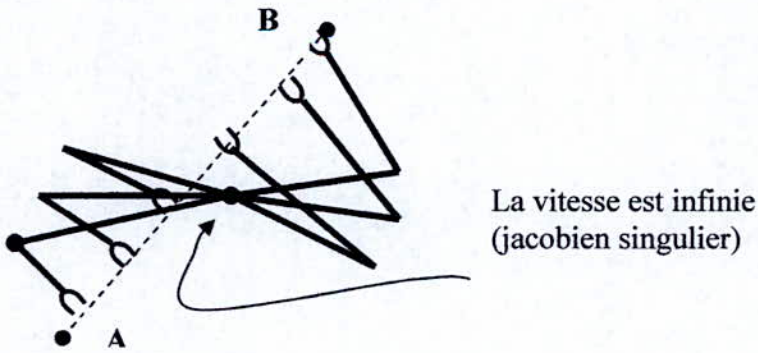


Figure I-11 : *Vitesse très grande en approchant l'axe1*

I-I-3-3 Problème 3 :

Le troisième problème se produit quand il y aura plusieurs solutions pour atteindre un point donné dans la trajectoire, la **Figure I-12** montre un robot manipulateur à deux liaisons de même longueur ayant des butées au niveau des articulations qui restreignent le nombre de solutions avec lesquelles il peut atteindre un point donné dans l'espace. En particulier, un problème surviendrait si le point d'arrivée ne peut pas être atteint dans la même solution physique quand le robot est au point de départ. Comme on voit dans la **Figure I-12** le manipulateur peut atteindre tous les points du parcours pour une certaine solution, mais pas pour n'importe quelle solution. Dans cette situation le système ne peut pas se déplacer à causes des butées mécaniques.

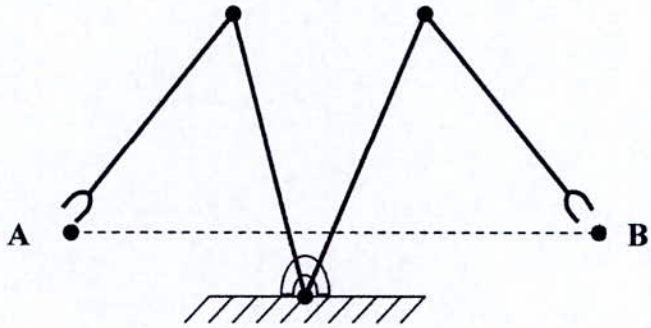


Figure I-12 : *Cas de deux solutions possibles*

I.I.4- Etude comparative et discussion du choix :

Les deux approches étudiées précédemment présentent plusieurs avantages et inconvénients. La génération du mouvement dans l'espace articulaire présente les avantages suivants :

- Elle nécessite moins de calcul en ligne, puisqu'il n'y a pas d'appel au modèle géométrique ou cinématique inverse ;
- Le mouvement n'est pas affecté par le passage par les configurations singulières ;
- Les contraintes de vitesses et de couples maximaux sont directement déduites des limites physiques des actionneurs.

En contre partie, 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 libre.

La génération de mouvement dans l'espace opérationnel permet de contrôler la géométrie de la trajectoire. En revanche :

- Elle exige 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 ;
- Elle est mise en échec chaque fois que les points de la trajectoire engendrée ne sont pas dans l'espace accessible par le robot ou chaque fois que la trajectoire impose une reconfiguration du mécanisme (changement d'aspect en cours de trajectoire);
- les limites en vitesse et en couple dans l'espace opérationnel varient selon la configuration du robot. On exprime alors ces limites par des valeurs traduisant des performances moyennes, satisfaites quelle que soit la configuration du robot. On impose donc au robot de travailler en deçà de ses capacités réelles.

Le choix d'une méthode de génération de mouvement dépend de l'application considérée. Chaque approche à ses propres limites, inhérentes au fait que les contraintes sont exprimées soit dans l'espace articulaire (vitesses, couples, buttées) soit dans l'espace opérationnel (précision, obstacles).

I.I.5- Conclusion :

Nous avons présenté dans cette partie la notion de trajectoire et les différentes interpolations existantes dans l'espace articulaire et opérationnel ainsi que les avantages et les inconvénients de chacune.

Les méthodes de planification de mouvement entres deux points se distinguent par la nature de la trajectoire entres ces deux points. L'intérêt est de considérer un mouvement continu entre les deux points pour éviter les risques d'usures et de vibrations qui peuvent déstabiliser le mécanisme, c'est les interpolations polynomiales d'ordre trois et cinq, bang -bang et trapèze qui peuvent assurer une telle trajectoire.

La planification de trajectoire dans l'espace articulaire bien qu'elle soit optimal en terme de volume de calcul et évite la notion de singularité. Néanmoins la trajectoire de l'outil est imprévisible.

Par contre la planification du mouvement dans l'espace opérationnel spécifie la trajectoire de l'outil. Mais coûteuse en terme de volume de calcul puisqu'elle fait une correspondance entre les deux espaces et présente la notion de singularité qui doit être prise en compte impérativement lors de calcul du modèle géométrique inverse.

Une fois les références en position sont déterminées on doit les asservir pour assurer une poursuite qui est l'objet de la partie suivante.

CHAPITRE :

I

I- NOTION EN ROBOTIQUE

I-II- La Commande en Robotique :

Dans ce présent chapitre, nous traitons la régulation en robotique, ainsi que le type de commande le plus utilisé. Néanmoins certains volets qui ont trait avec le contrôleur utilisé dans notre application - l' UMAC en l'occurrence - seront traités ; A savoir le pourquoi d'une commande décentraliser (UMAC utilise une commande décentralisée) , le choix du régulateur PID (PID est l'algorithme implémenté dans la carte), ainsi que quelques notions sur l'autotuning (auto ajustement), étant donné que les paramètres du régulateur PID implémentés peuvent être déterminés automatiquement a l'aide du pack logiciel *Pmac Tuning Pro.*

I.II.1- Le pourquoi d'une commande décentralisée: [3]

Commande décentralisée, est la synthèse d'un asservissement pour chaque articulation individuelle du robot, une telle loi de commande. Qui n'est applicable que si les interactions du système robotique totales sont négligeables, du fait que cette loi ne les prenne pas en considération. Ce qui est vérifier dans notre cas, où les moments de Coriolis et centrifuges sont négligés, car l'actionneur est un moteur à courant continu dont les dynamiques sont linéaires et bien connues et possédant un réducteur avec un rapport de réduction important (η =65.5) dont l'effet est largement de découpler le système et réduire les non linéarités entres les articulations. **[3.p168]**. Toutefois la présence du réducteur introduit des frictions qui sont traitées par la boucle. Ce qui permet à ce type de commande de réaliser les performances désirées, et d'un autre côté c'est la plus facile de point de vue implémentation, c'est la raison pour laquelle elle est fréquemment appliquée dans les contrôleurs des robots.

I-II-1-1 Démonstration :

Dans ce qui suit, on suppose la simplification suivante :

$$q_k = \theta_{s_k} = r\theta_{m_k} \qquad \textbf{(I-14)}$$

$$\tau_{l_k} = \tau_k \qquad \textbf{(I-15)}$$

Alors l'équation de mouvement du manipulateur :

$$\sum_{j=1}^{n} d_{jk}(q)\ddot{q} + \sum_{i,j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k \qquad \textbf{(I-16)}$$

L'équation (1.17) représente l'inertie non linéaire, centrifuge, Coriolis et gravitationnelle avec :

$d_{jk}(q)$: Élément de la matrice d'inertie

$c_{ijk}(q)$: Élément de la matrice d'inertie Coriolis et centrifuge

$g_k(q)$: Inertie gravitationnelle

τ_k : Couple de l'articulation k

$\tau_{\ell k}$: Couple de la charge

r : Rapport de réduction du moteur

θ_{m_k} : Position du bras du moteur

θ_{sk} : Position de l'articulation

D'autre part on la dynamique de l'actionneur :

$$j_m\ddot{\theta}_{m_k} + (B_m + \frac{K_b K_m}{R})\dot{\theta}_{m_k} = \frac{K_m}{R}V_k - r_k\tau_k \qquad k = 1,.....,n \qquad \textbf{(I-17)}$$

Avec :

j_m : Moment d'inertie du bras du moteur

B_m : Coefficient de frottement mécanique

K_b : Constante de fem

K_m : constate de couple en N-m/amp

R : Résistance rotorique

V_k : Tension de commande

L'approche plus simple pour le régulateur du système est de considérer le terme non linéaire τ_k défini par (I-17), comme une perturbation sur le moteur (actionneur), et de désigner un régulateur pour chaque articulation, et ceci on se basant sur l'équation (I-18) . Ce qui est important à noter, est que τ_k est proportionnelle au taux de réduction r_k , ce qui permet de réduire les couplages non linéaires. Néanmoins dans le cas ou le facteur de réduction n'est pas important une telle commande peut entraîner des erreurs de poursuites assez importantes.

Dans ce qui suit, nous allons développer seulement les commandes PD, PID. Car c'est le type de commande implémenté dans notre interface de commande à savoir l'UMAC.

I.II.2- Régulateur PD : [3]

La synthèse pour ce cas de figure se fait en considérant la fonction de transfert de l'asservissement (Obtenus en combinant les différentes fonctions de transferts constituant la boucle, et en prenant comme actionneur un moteur a courant continus, dont les fonctions qui le régissent sont préalablement connues), et en considérant aussi les inerties gravitationnelles comme une perturbation de type échelon, qui surgit a l'arrêt de l'articulation.

La boucle de régulation dans le cas d'un régulateur PD est montrée dans la **Figure I-13** :

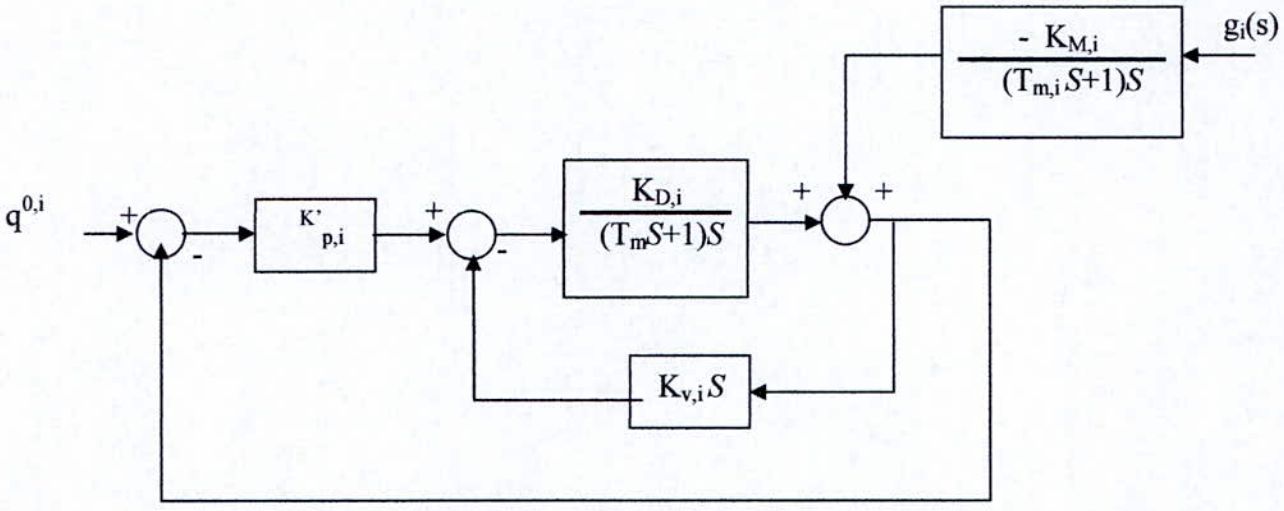


Figure I-13: *Boucle de régulation proportionnelle dérivée*

$K'_{p,i}$: gain proportionnel du régulateur

$K_{D,i}$: gain dérivé

$K_{v,i}$: gain de la boucle de vitesse.

$K_{M,i}$: constante caractérisant le moteur.

Dans notre cas, le PD utilisé est une variante du standard PD, en effet, ce dernier présente une boucle de retour en vitesse dont l'intérêt sera expliqué ultérieurement.

Néanmoins, une telle régulation n'assure pas une bonne précision. Car, une erreur statique est toujours présente lors du positionnement de l'articulation.

I.II.3- Régulateur PID [3]:

On a vu que la boucle d'asservissement PD assure le positionnement de l'articulation i. Néanmoins, l'erreur statique apparaît toujours, qui est causé par la charge extérieure.

Ce moment de la charge extérieur peut être compensé en utilisant deux approches, la première approche consiste à compenser ce moment en le calculant on line. Ce qui cause beaucoup de problèmes, d'un côté dus à la complexité des expressions de ce moment (qui dépendent de la structure du robot) et le volume important de calcul qu'elles entraînent. Et d'un autre côté, il y a l'incertitude dans le détermination des paramètres nécessaires au calcul de ce moment, ce qui rend cette approche peut fiable vis-à-vis de la compensation de l'erreur statique. La deuxième approche consiste à introduire l'intégration de l'erreur en position $(q^0 - q)$, donc on utilisera un contrôleur PID.

Si on permet aux autres articulations de bouger simultanément avec l'articulation i, alors le moment extérieur dû au couplage dynamique influence l'articulation i, ce moment cause aussi une erreur de positionnement sur l'articulation i. L'effet du moment qui est une fonction complète de toutes les coordonnées des articulations, de leurs vitesses ainsi que de leurs accélérations sera compensé par la boucle d'intégration.

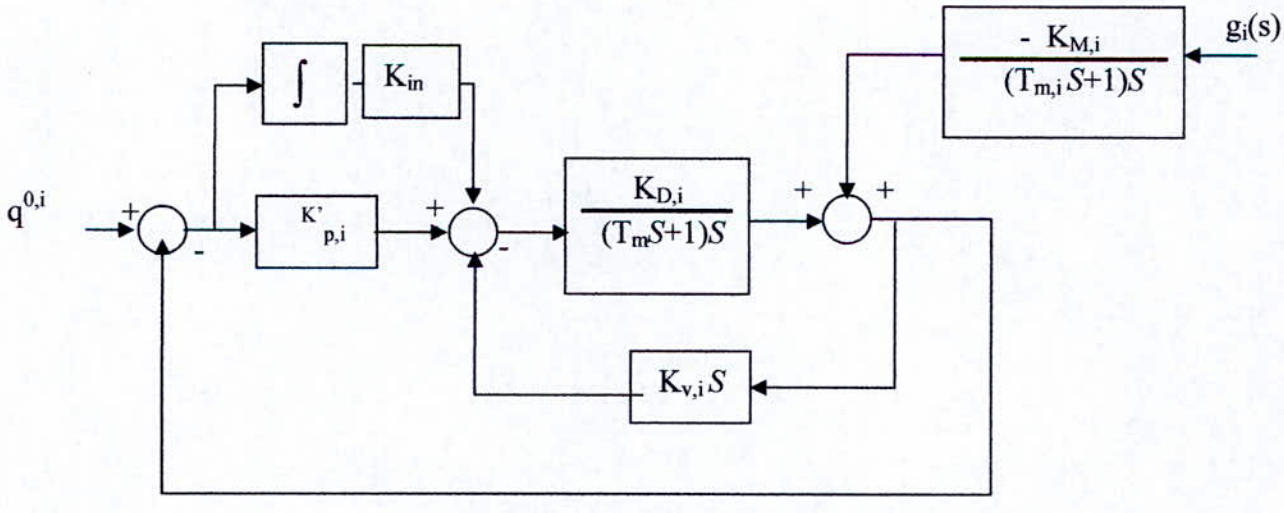


Figure I-14 : *Boucle de régulation PID*

Il est évident que l'introduction de l'intégral dans la boucle augmente l'ordre du model du système, si l'ordre du l'actionneur avec le mécanisme en mouvement est deux, en appliquant la boucle d'intégration, on obtient un système d'ordre trois.

I.II.4- Boucle de vitesse : [4]

Parce que un filtre passe-bas n'élimine pas complètement (mais les diminuent moyennement) les signaux impulsionels dérivatifs causés par les changements du point de fonctionnement ou les perturbations. La modification de l'unité feedback négative de la structure PID est très intéressante **[5].**

Afin de bloquer les effets des changements brusques du point de fonctionnement, on considère une variante de la structure standard du feedback PID, cette variante utilise la sortie du processus à la place du signal d'erreur pour l'action dérivative **[6].** Comme montré dans l'équation générale suivante :

$$u(t) = K_P e(t) + K_I \int_0 e(\tau) d\tau - K_D \frac{dy(t)}{dt} \qquad \textbf{(I-18)}$$

Ou :

- $y(t)$ est la sortie du système.
- $e(t)=r(t)- y(t)$ est le signal d'erreur.
- $r(t)$ est le point de fonctionnement ou la référence.

Le dernier terme de l'équation **(I-19)**, constitue la boucle de retour de la vitesse et, d'où une extra boucle qui n'est pas directement affectée par les changements brusques du point de fonctionnement.

Néanmoins, les changements des perturbations et des bruits qui peuvent survenir a la sortie du système régulé, peut provoquer des signaux de commande infinis.

I.II.5- PID Auto-Tuning (auto ajustement) :

Les régulateurs PID (proportionnel, intégral, et dérivée), sont très utilisés dans la régulation des processus industriels depuis plus de 60 ans. De nos jours avec l'évolution technologique, on assiste à l'émergence des régulateurs PID numériques (particulièrement les Autotuning PID) et qui deviennent de plus en plus répondus dans l'industrie. Ces derniers sont automatiquement configurable (i.e ses paramètres) selon le processus qui est asservis et l'excitation appliquée.

Les paramètres du régulateur ainsi synthétisé sont optimaux à l'asservissement du système et dépendent du comportement de ce dernier.

Les méthodes d'auto-ajustement *(tunning)* du régulateur se divisent en deux principales catégories **[7]** : méthodes *on line* et qui ne nécessitent pas la connaissance du model du système et les méthodes qui construisent le model du système. La première méthode règle les paramètres du PID dans une boucle fermée avec un système donné en utilisant un algorithme d'optimisation comme la méthode de Newton ou la méthode de *steepest descente* pour minimiser certaines fonctions de coût **[8]** , par exemple : l'erreur entre l'entrée et la sortie.
La seconde approche, construit un model du système et en conséquence, elle décide des paramètres du contrôleur, en utilisant une approche déterministe ou une méthode d'optimisation **[9].**

I.II.6- Les approches les plus communes à la synthèse du PID: [10]

Nous présentons maintenant un certain nombre de méthodes basées sur le modèle, et d'autres basées sur l'optimisation des caractéristiques. Ceci n'est pas censé être une liste approfondie, plutôt un choix raisonné des techniques bien établies et utiles. Ces méthodes, et les *AutoTuners* qui les emploient, peuvent exiger du l'utilisateur de choisir quelques paramètres de conception.

I-II-6-1 Les méthodes basées sur le modèle :

Ces méthodes se fondent sur une certaine technique d'identification qui doit fournir une structure de modèle simple et fixe- en raison de la nécessité de dérivation des règles simples de l'ajustement. Quoique approximatif. Ce modèle doit représenter le processus suffisamment pour que le résultat de l'ajustement soit précis. Quand ces méthodes sont employées manuellement c'est très utile de choisir les paramètres de conception possibles. Quand elles sont employées dans un *AutoTuner*, il est important de vérifier si les résultats d'identification sont rendus disponibles à l'utilisateur : si c'est le cas, ils peuvent être une source valable d'information pour le diagnostic de processus, nous citons entre autre :

- La méthode de HAALMAN
- La méthode du symétrique optimum (SO)
- La méthode de DAHLIN ou λ-tuning
- La méthode de kappa tau ou KT
- La méthode de commande par model interne

I-II-6-2 Les méthodes basées sur les caractéristiques :

L'idée principale de ces méthodes est que la description du processus n'est pas un modèle, plutôt quelques valeurs le caractérisant dans le domaine temporel ou fréquentiel. En dérivant quelques méthodes, un modèle peut être impliqué, mais ceci n'est pas censé y être pour représenter le processus, mais plutôt, afin de permettre la prédiction des résultats de l'ajustement. On citera deux méthodes à savoir :

- Méthode de Ziegler-Nichols
- Méthode basée sur le retard (*Relay-based methods*)

Ainsi que d'autres méthodes qui utilisent des algorithmes d'optimisation.

I.II.7- Le feed forward : [11]

Dans la plupart des contrôleurs, la boucle de régulation pour chaque axe est fermée indépendamment, sans souci de l'action d'autres axes (commandes décentralisées). Des effets provoqués par d'autres axes sont juste traités par la boucle comme des perturbations à rejeter. Mais naturellement, avant qu'ils puissent être rejetés, ils doivent créer de vraies erreurs à l'axe — qui peuvent compromettre la qualité du mouvement. Dans des contrôleurs plus sophistiqués ces effets prévisibles d'inter axes peuvent être employés comme des termes ***du feed forward***.

Le but du feed forward est de prévoir le couple exigé pour une trajectoire donnée, laissant les perturbations et les erreurs imprévues à la boucle de retour. Le plus largement répandus sont des termes de feed forward d'un simple axe basés sur la trajectoire propre de cet axe —en particulier sa vitesse et son accélération. Le même principe peut être prolongé aux effets dynamiques sur les axes inter connectés.

L'utilisation du feed forward interconnecté, employant la vitesse commandée instantanée et l'accélération de l'autre axe *en plus des termes standard du feed forward,* réduits nettement les perturbations induites par ces effets. Ceci peut mener une trajectoire réelle plus précise et/ou des possibilités dynamiques accrues tout en maintenant un niveau donné d'exactitude. Le **Tableau I-1** et **TableauI-2** montrent les termes interconnectés des gains exigés pour ce mécanisme.

Articulation	Cross-coupled Action/réaction gain Multiplié par l'accélération des autres articulations	Cross-coupled centrifugal gain Multiplié par le carré de la vitesse des autres articulations	Cross-coupled coriolis gain Multiplié par le produit de vitesse des autres articulations
épaule	$m_2L_2^2+m_2L_1L_2\ COS(\theta_E)$	$-m_2L_1L_2SIN(\theta_E)$	$-2m_2L_1L_2SIN(\theta_E)$
coude	$m_2L_2^2+m_2L_1L_2COS(\theta_E)$	$m_2L_1L_2SIN(\theta_E)$	0

Tableau I-1 : *Les interactions entre les articulations*

En plus des termes standards et constantes d'un moment d'inertie qui peuvent être multipliées par l'accélération commandée pour produire un terme d'accélération feed forward (comme dans un système cartésien), notre robot SCARA a également un composant variable dans son moment d'inertie au niveau du l'articulation d'épaule. Ce dernier change avec l'angle du l'articulation du coude. Ceci signifie que chaque mouvement du coude change le gain feed forward d'accélération optimale. Le tableau ci-dessous montre les composants constants et variables du gain feed forward d'accélération.

articulation	Gain constant feedforward en accélération	Gain variable feedforward en accélération
épaule	$(m_1+m_2)L_1^2+m_2L_2^2$	$2L_1L_2m_1COS(\theta_E)$
coude	$m_2L_2^2\ \ddot{\theta}_E$	0

Tableau I-2 : *Gains proposer pour les paramètres des feed forward d'accélération*

I.II.8- Conclusion :

Dans cette partie nous avons présenté la commande décentralisée de type PID qui est largement appliquée dans la robotique vue la simplicité de son implémentation est les performances quelle peut garantir en positionnement et en poursuite par l'ajout des termes de *feed forward* en vitesse et accélération. Ceci est rendu possible grâce à la réduction des effets de couplage non linéaires par le rapport de réduction des moteurs. Nous avons présenté aussi quelques

méthodes d'auto détermination des paramètres du régulateur qui sont divisées en deux approches celles se basant sur le calcul d'un model et celles sur l'optimisation des caractéristiques.

I-III- Robot SCARA modélisation :

Pour maximiser les performances du système, une exploration et une définition des situations cinématiques et géométriques sont requises. Ceci est fait en exploitant les relations géométriques entres l'organe terminal et les différentes articulations rotoides et prismatiques du mécanisme qui causent son mouvement.

Le modèle géométrique direct calcule les coordonnées de l'organe terminal à partir des positions des articulations. Inversement, le modèle géométrique inverse calcule la position des articulations à partir des coordonnées de l'organe terminal. Les deux transformations sont nécessaires pour l'utilisateur qui veut programmer des mouvements dans l'espace de l'organe terminal (espace cartésien).

Les calculs géométriques directs sont fais au début pour établir les coordonnées de l'organe terminal à partir d'une configuration arbitraire du mécanisme pour le mouvement initial. Les calculs géométriques inverses sont nécessaires au moins au point final de chaque mouvement. Si des trajectoires sont commandées alors ces calculs sont fais à des intervalles étroitement espacés le long de la trajectoire

I.III.1- Analyse cinématique du mécanisme : [11], [2], [3]

Le mécanisme du robot SCARA de notre application possède trois articulations deux rotoides parallèles d'axes vertical –l'épaule S –le coude E et une articulation prismatique V dans la direction verticale donc l'organe terminal possède trois degrés de liberté : trois translations dans l'espace (X,Y,Z).

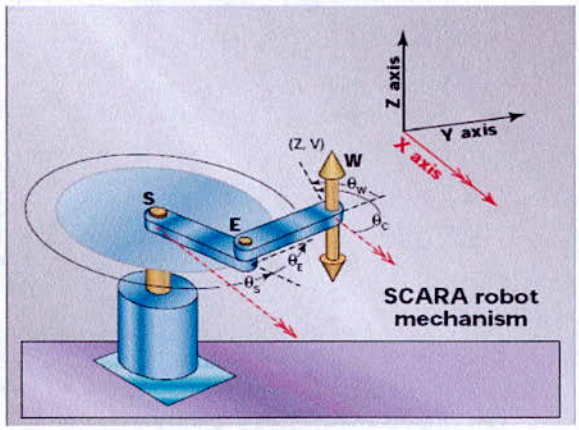
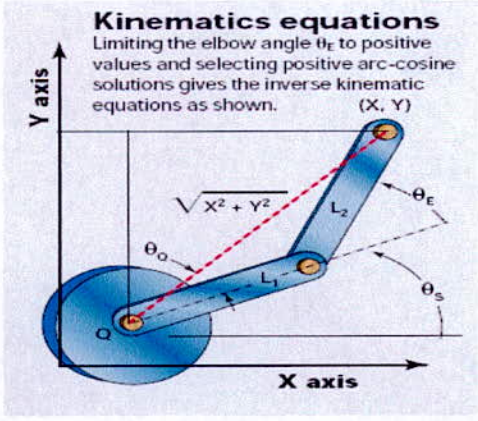


Figure I-15: *robot SCARA*

Les équations du modèle géométrique direct sont :

$$\begin{aligned} X &= L_1\cos(\theta_s)+L_2\cos(\theta_s+\theta_E) \\ Y &= L_1\sin(\theta_S)+L_2\sin(\theta_S+\theta_E) \\ Z &= V+Z_0 \end{aligned} \qquad \textbf{(I-19)}$$

En se limitant à des valeurs positives de l'angle du coude E qui produit une configuration droite *-right-armed* (obtenue en choisissant les solutions positives de l'arc cosinus) on peut écrire les équations du modèle géométrique inverse comme suit :



$$\begin{aligned} \theta_E &= +\cos^{-1}\left(\frac{X^2+Y^2-L^2-L^2}{2L_1L_2}\right) \\ \theta_s+\theta_Q &= \arctan 2(Y,X) \\ \theta_Q &= +COS^{-1}\left(\frac{X^2+Y^2+L_1^2-L_2^2}{2L_1\sqrt{X^2+Y^2}}\right) \\ \theta_S &= \left(\theta_S+\theta_Q\right)-\theta_Q \\ V &= Z-Z_0 \end{aligned} \qquad \textbf{(I-20)}$$

Figure I-16: *robot SCARA dans l'espace articulaire*

Malgré que la simplicité de ce mécanisme simplifie les problèmes généralement rencontrés avec les mécanismes non cartésiens, néanmoins un problème peut être rencontrer dans ce cas de figure qui est le faite qu'il peut y avoir deux solutions dans la transformation géométrique inverse, selon la configuration à droite ou la configuration à gauche du mécanisme.

Il y a aussi des cas où il n'y a pas de solution possible. Ces cas particuliers doivent être traités – en vérifiant que les arguments des fonctions arc cosinus sont hors de la gamme valide, ce qui se produit lors des positions inaccessibles physiquement de l'outil.

I.III.2- Dynamiques et Interactions entres les articulations du robot : [11]

Le robot SCARA que nous avons exploré ici peut également être utilisé pour illustrer les issues dynamiques spéciales trouvées dans les mécanismes non-Cartésiens. Le défi principal de la dynamique est que les équations dynamiques peuvent changer de manière significative avec la position des articulations — en particulier l'articulation du coude dans ce cas-ci — et la vitesse et l'accélération d'une articulation peuvent affecter le couple exigé pour les autres. Les termes de base qui contribuent au couple exigé pour les articulations sont dressés dans le **Tableau I-3** .

couple	Termes d'inertie	Termes centrifuge	Termes de coriolis
Epaule τ_S	$[(m_1+m_2)L_1^2+m_2L_2^2+2m_2L_1L_2\,COS(\theta_E)]\,\ddot{\theta}_S$ $+[m_2L_2^2+ m_2L_1L_2COS(\theta_E)]\ddot{\theta}_E$	$-m_2L_1L_2SIN(\theta_E)\dot{\theta}_E^2$	$-2m_2L_1L_2SIN(\theta_E)\,\dot{\theta}_E\dot{\theta}_S$
Coude τ_E	$[m_2L_2^2+m_2L_1L_2COS(\theta_E)]\ddot{\theta}_E+m_2L_2^2\ddot{\theta}_E$	$m_2L_1L_2SIN(\theta_E)\,\dot{\theta}_S^2$	

Tableau I-3 : *Les inerties de chaque articulation*

Le **Tableau I-3** montre les termes de base qui contribuent au couple exigé pour les articulations d'épaule et du coude. Les bras supérieurs et inférieurs ont les masses *m1* et *m2*, respectivement, les longueurs *L1* et *L2*. Les termes sont séparés dans trois catégories : termes d'inertie, termes de force centrifuge, et termes des forces de Coriolis. En réalité, le moment d'inertie est *mL2/3* si la masse est également distribuée sur la longueur. Pour la simplicité, on suppose que la masse de chaque lien est concentrée à la fin lointaine du lien, donnant une limite mL^2 pour le moment d'inertie.

Certains termes d'inertie seraient aussi présents dans un système cartésien. Par exemple :

$$\tau_E = m_2 L^2 (\ddot{\theta})_E \qquad \textbf{(I-21)}$$

D'autres termes sont uniques aux systèmes non Cartésiens. Dans les termes d'inertie, nous voyons que le couple à une articulation dépend de l'accélération de l'autre articulation. C'est dû à la troisième loi classique newtonienne de l'action et de la réaction.

En outre, les valeurs d'un moment d'inertie dépendent au moins partiellement de l'angle du l'articulation du coude. Plus le bras est prolongé au coude inférieur, plus le moment de l'inertie devient grand au niveau de l'épaule. En plus de ces forces il existe les effets dus aux forces centrifuges et celles de Coriolis. Les termes centrifuges sur une articulation sont proportionnelles *au carré de la vitesse de l'autre articulation*, et sont à un maximum quand les deux liens sont perpendiculaires.

La limite de Coriolis est proportionnelle *au produit des deux vitesses communes*, et elle est également à un maximum quand les deux liens sont perpendiculaires.

I.III.3- Conclusion :

Nous avons exposé dans cette partie le calcul géométrique direct et inverse du robot SCARA qui sont relativement simples à cause de la structure du mécanisme qui n'a que trois articulations nous avons remédié aux problèmes d'existence de plusieurs solutions et de singularité lors du calcul géométrique inverse par un choix des solutions positives de l'arc cosinus et par la vérification de la validité des arguments des fonctions arc cosinus.

CHAPITRE :

II

II- INTERFACE DE COMMANDE UMAC

II-I- Introduction :

PMAC (*Programmable Multi Axis Controler*), est une famille de contrôleurs de mouvement multi axes à hautes performances de la firme DELTA TAU DATA SYSTEMS capable de commander jusqu'à huit axes simultanément. Grâce à la puissance de son processeur DSP.
Le CPU de PMAC et la DSP 56003 de FREESCALE (MOTOROLA) et il assure tous les calculs nécessaires pour les huit axes.
PMAC est disponible sur quatre versions à savoir :
PMAC-PC, PMAC-LITE, PMAC-VME et PMAC-STD ces cartes se différent entres elles par leurs formes, la nature de leurs interfaces bus et par le disponibilité de certains port E/S, mais elles ont toute les même constituant donc in programme écrit pour une version peut être exécuté par toutes les autre versions.

Comme un but général, PMAC peut servir dans une large variété d'application, à partir de celle exigeant des précisions de l'ordre des sous multiples du millimètre à celle exigeant des centaines de kilowatts. Ses diverses applications incluent la robotique, les machines à outils, impression, traitement du papier et du bois, les chaînes d'assemblage, l'agroalimentaire, contrôle camera, soudage automatique, découpage à laser, traitement des gaufrettes de la silicone.

II-II- TURBO UMAC :

Le système turbo PMAC format 3U de Delta Tau combine la puissance de la famille turbo PMAC avec des stratégies de connections intégrées qui donnent à l'utilisateur une flexibilité révolutionnaire et une simplicité d'utilisation. Le coeur de ce système est la carte turbo PMAC CPU 3U décrite ci-dessous. Elle peut être soit connectée directement sur les autre carte accessoires pour former TURBO STACK cette configuration est moins chère et plus compacte que la configuration UMAC mais elle est limitée en terme du nombre d'axe à commander (seulement huit axes), alors que la configuration UMAC permet la commande de 32 axes. Les composants sont connectés à une surface arrière appelée UBUS et installer à l'intérieur d'un casier en format

3U. Un ordinateur PC/104 et plusieurs accessoires de communication optionnels peuvent être installé dans le système UMAC fournissant une flexibilité convenable et une virtuelle expansion illimitée.

II.II.1- Les Constituants de l'UMAC :

II-II-1-1 TURBO PMAC2-3U CPU [12] :

C'est l'unité centrale du traitement du système, elle joue le rôle d'un gestionnaire du système et d'un contrôleur pour les moteurs, Elle contient une DSP Freescale (Motorola) 56303, des mémoires où sont implémentés les microprogrammes et un connecteur série RS 232/422.

Le processeur de traitement du signal numérique **DSP56303** de Freescale (Motorola) est l'élément principal du Turbo Pmac2 3U CPU, car c'est l'unité centrale de cette carte où tous les algorithmes de commande seront calculés pour les huit axes.

Cette carte a sa propre mémoire et microprocesseur, par conséquent elle fonctionne comme un contrôleur autonome qu'on peut commander par l'intermédiaire d'un port série. Le port série est connecté directement à cette carte, le diagramme en block du Turbo Pmac2 3U CPU est présenté par la **Figure IV.2** suivante :

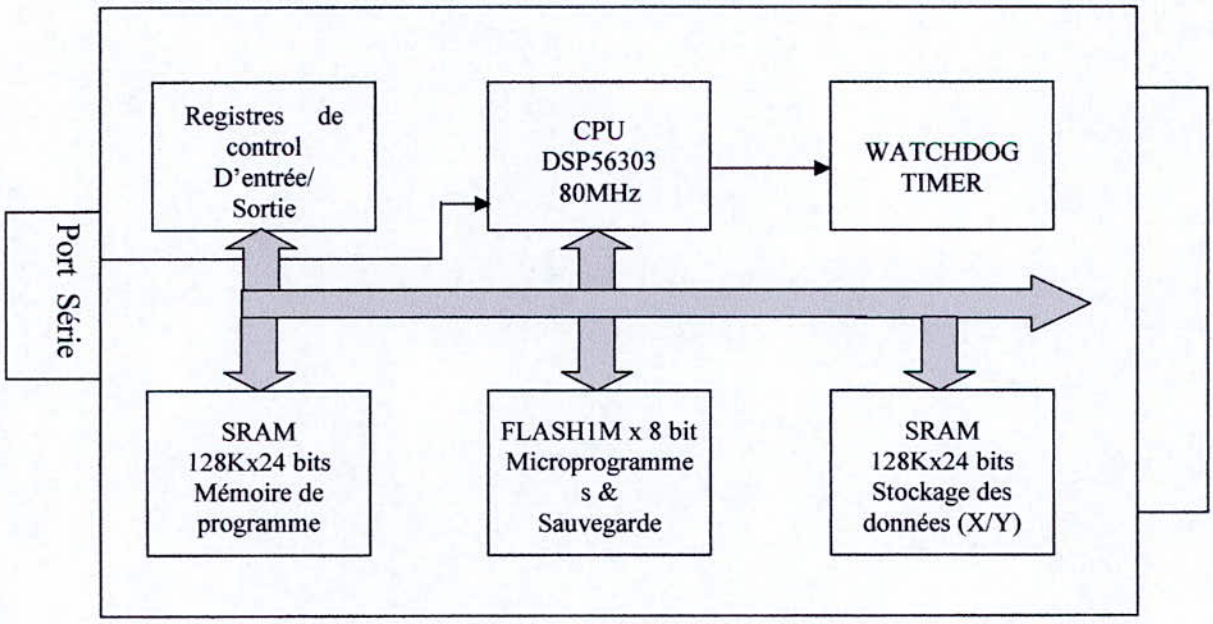


Figure II-1: *Diagramme en block du Turbo Pmac2 3U CPU*

D'après ce schéma on distingue les principaux éléments :

- Une unité centrale de traitement Motorola DSP56303 80 MHz 24 bits.
- Mémoire flash (1M x 8 bits) pour la sauvegarde d'utilisateur et les microprogrammes

- Mémoires SRAM (128k X 24 bits) actives pour les programmes, la compilation et l'assemblage.
- Mémoires SRAM (128k X 24 bits) de données utilisateur.
- Connecteur avec interface série RS-232/422.
- Whatchdog timer.
- Registres de control.

II-II-1-2 DSPGATE [13] :

Nous disposons de deux DSPGATE qui jouent le rôle d'une première interface entre le CPU (Turbo PMAC2-3U CPU) et les signaux d'entrée/sortie pour les moteurs, chacune d'elles peut contrôler quatre axes et chaque axe envoie les signaux de sortie de commande pour les amplificateur et les signaux reçus d'entrée encodeur et indicateurs.

L'unité centrale de traitement (Turbo Pmac2 3U CPU) communique avec les capteurs et les amplificateurs par un assemblage de circuits faits sur commande particulière de **DELTA TAU** [1], désigné sous le nom de DSPGATES (portes de la DSP). Nous disposons de deux DSPGATES classés comme accessoire ACC-24E 4A, qui sont équivalent à deux ACC24E2A avec l'option 1.

Chaque DSPGATE contient quatre canaux (pour commander jusqu'à 4 moteurs), de ce fait, il y à huit canaux qui peuvent être utilisés pour commander jusqu'à huit moteurs. Connectés à la surface arrière UBUS, l'ACC-24E 4A prend un total de deux slots (deux connections).

Une DSPGATE ne contient aucun processeur, il y à quatre canaux contenant les circuits de conversion numérique/analogique DAC, compteurs, temporisateurs et filtres digitaux pour la commande en associant autour d'eux des buffers et des connecteurs pour les accès entrée/sortie.
Cette DSPGATE permet les types de commande suivants :

- Commandes en modulation de largeur des impulsions de ±10V.
- Commandes analogiques de couple de ±10V.
- Commandes d'Impulsion -et- direction pour les moteurs pas à pas.

Chaque canal de ces DSPGATE est constitué par :

- 3 sorties pour les signaux de commande des amplificateurs configurables pour les types de commande suivant :
  - 3 sorties de commande en tension avec modulation de largeur des impulsions PWM.
  - 2 sorties de Convertisseur Digital Analogique DAC, commande en courant.
  - 1 sortie de signaux impulsion -et- direction pour les moteurs pas à pas.
- 3 entrées pour lignes de l'encodeur incrémentale.
- 8 indicateurs entrés, 2 indicateurs sortis.

Remarque :
Vus par le CPU et exactement par la DSP les registres des DSPGATEs et IOGATE sont considérés comme des zones mémoires qu'on peut lire ou écrire.

II-II-1-3 IOGATE [14] :

C'est un composant d'adaptation des signaux d'entrée-sortie digitaux avec isolation optique, elle est faite pour compléter les fonctions d'automatisation de l'UMAC.
Nous disposons d'une carte IOGATE (*Input/Output GATE*) classée comme accessoire ACC12E de DELTA TAU, c'est une carte d'entrée/sortie d'usage universel au système UMAC, en format 3U conçue pour la transmission et la réception des données de l'extérieur du système au CPU, par l'intermédiaire de l'UBUS, elle fournit 24 lignes d'entrées optiquement isolées et 24 lignes de sorties à tension élevée qui sont aussi optiquement isolées, la lecture et l'écriture des données d'entrées/sorties est effectuée en utilisant les variables M, qui correspondent à des pointeurs de zone mémoire.

II-II-1-4 AMPLIFICATEUR DE PUISSANCE [15] :

Il y'a huit amplificateurs répartis sur deux cartes, chaque amplificateur a pour rôle de rendre les signaux de commande délivrée par la DSPGATE amplifié et exploitable.

Nous disposons de deux cartes d'amplificateur AMP-2 (quatre axes analogiques, amplification PWM, 40 VDC 3/5A) en format 3U prenant en globalité 4 slots (quatre connections), destinés pour être connectés à la DSPGATE ACC24E2A et pour contrôler des moteurs à courant continu ou pas à pas, chaque carte possède quatre axes d'amplifications avec une boucle de régulation analogique de courant (pour la régulation du couple du moteur à courant continu), et avec des signaux de sortie du type modulation de larguer des impulsions PWM. La tension

maximale d'alimentation pour ces amplificateurs est de 40VDC, et l'estimation du courant maximum pour chaque amplificateur est de 3A continu pour une puissance maximale de 120W.

II-II-1-5 CIRCUIT D'ALIMENTATION ELECTRIQUE [16] :

L'alimentation électrique aux circuits numériques est de 15V fournie par l'intermédiaire de l'UBUS, et l'alimentation électrique nécessaire pour l'amplification est comprise entre 15V et 40V fournie par l'intermédiaire des lignes d'entrée BUS+ et BUS, le courant ne peut pas excéder 12A (3 par axe) en moyenne et 20A (5 par axe) en pique, une installation des fusibles de 5A à action lente est mise en place pour protéger les shunts (faible résistance), et des fusibles de 15A à action rapide pour une protection des lignes d'alimentation.

La température opérationnelle de ces amplificateurs est de 0°C à 55°C, et la température de stockage et de -12°C à 82°C pour un pourcentage d'humidité de 0% à 95%.

II-II-1-6 BUS DE CONNECTIONS (UBUS) [17] :

Le désir d'avoir une meilleure flexibilité lors de la conception d'une application de contrôle du mouvement complète a causé le développement de l'UBUS (UMAC BUS), c'est un protocole de bus parallèle d'architecture ouvert, et qui est conçu comme interface simple et robuste pour câbler les différents dispositifs (module) en l'utilisant comme une surface arrière commune et familière (standard pour les produits DELTA TAU), cette interface crée un mécanisme uniforme par lequel un processeur peut contrôler toutes les cartes qui sont dans le même système.

Cette norme de bus est décrite à **ANSI/IEEE STD1014-1987** elle à des caractéristiques semblables a la norme de VME, à l'exception de quelques descriptions de broches qui sont différentes, ce qui fait que l'UBUS est non compatible avec la norme de bus VME.

Les connecteurs des cartes sont du modèle **DIN41612** à 96 lignes, le nombre de slots désigne le nombre de connecteurs de ce type (indépendamment de l'alimentation électrique) qui sont présents sur l'UBUS, pour notre cas ont dispose de deux UBUS un pour connecter les amplificateurs avec quatre slots et un pour connecter le CPU, les DSPGATEs et IOGATE avec six slots.

II-III-Conclusion :

Dans ce chapitre nous avons présente l'interface UMAC en mettant l'accent sur sa puissance qui réside dans son processeur DSP 56303 de *freescale* (Motorola) dont le diagramme en bloque et les caractéristiques le confirme.

Nous avons ensuite présenté les différentes cartes ou modules constituant l'UMAC, pour comprendre le fonctionnement de l'interface il est nécessaire d'étudier le rôle de chaque module dans le système global, ceci nous a considérablement aidé lors de la configuration matérielle des différents *jumpers* de chaque carte

CHAPITRE :

III

III- GÉNÉRATION DE MOUVEMENT DANS L'UMAC

III-I- Introduction :

Une tâche fondamentale en robotique consiste à planifier des trajectoires sans collision dans un environnement encombré d'obstacles. Ce problème est très difficile puisque, même dans sa version la plus simple où les contraintes sur les déplacements du robot sont purement géométriques, sa complexité croît rapidement (de façon probablement exponentielle) avec le nombre de degrés de liberté du robot. Dans les vingt dernières années, ce problème a suscité des recherches allant des mathématiques (géométrie algébrique, combinatoire, théorie de la complexité), à la programmation et l'implantation effective sur des robots en passant par la conception et l'analyse d'algorithmes performants.

L'idée centrale de tous les algorithmes est néanmoins toujours la même : construire une représentation de l'ensemble des configurations du robot pour lesquelles il n'y a pas de collision, ce qu'on appelle l'espace libre du robot. On ramène ainsi le problème du déplacement du robot dans l'espace où celui-ci évolue au problème du déplacement d'un point dans l'espace des configurations dont la dimension est en général élevée (six pour un manipulateur à six degrés de liberté). Moyennant une structuration appropriée de cet espace, on peut alors construire un chemin qui correspond à un déplacement sans collision du robot.

III.I.1- Trajectoires des programmes de mouvement : [18], [19]

Parmi les caractéristiques remarquables du PMAC sont, la puissance et la flexibilité des algorithmes de génération des trajectoires. Ces algorithmes permettent d'améliorer les performances d'une grande variété de difficultés de manœuvres, de même ils facilitent à l'utilisateur de faire ses propres compromis.

Néanmoins, entre facilité d'utilisation et le degré de control, il est important de savoir que ces trajectoires sont une série de positions commandées, et c'est à l'asservissement de position de les asservirent.

Dans ce qui suit nous allons présenter les mouvements principaux qu'on peut effectuer à l'aide de l'UMAC, ainsi que la façon avec laquelle ces mouvements sont générés

III.I.2- Mouvements Linéaires :

La classe la plus facile des mouvements qu'on peut faire est celle des mouvements linéaires. Dans ce type de mouvements, un axe se déplace vers la position cible à vitesse indiquée, en accélérant jusqu'à atteindre cette vitesse et en ralentissant de cette vitesse jusqu'à l'arrêt selon le mode commandée.

Si plus d'un mouvement sont indiqués en succession sans pause dans l'intervalle, le premier mouvement se mélangera dans le second avec la même accélération commandée.
Le mode de mouvement linéaire mélangé est le mode par défaut pour les programmes de mouvement. Si dans ce dernier, un autre mode de mouvement est choisi, le programme peut être mis en mode linéaire, avec l'expression **LINÉAIRE**. Le programme peut être pris hors du mode **LINÉAIRE** avec d'autres commandes de mode de mouvements (par exemple **CIRCLE1**, **CIRCLE2**, **RAPIDE**, **PVT**, **SPLINE**).Il est bon en programmant une application donnée, de déclarer le mode **LINÉAIRE** dans chaque programme, et ne pas compter sur le défaut.

Remarque : Dans l'UMAC les profils de vitesse et d'accélération, pour le mouvement linéaire mélangée, sont édictés par les valeurs des variables I a savoir : Ixx87 et Ixx88 qui sont les temps d'accélération TA et celui des S-curve TS respectivement (variables initialisées dans le programme de mouvement).
Les profils de vitesses sont données ci-dessous :

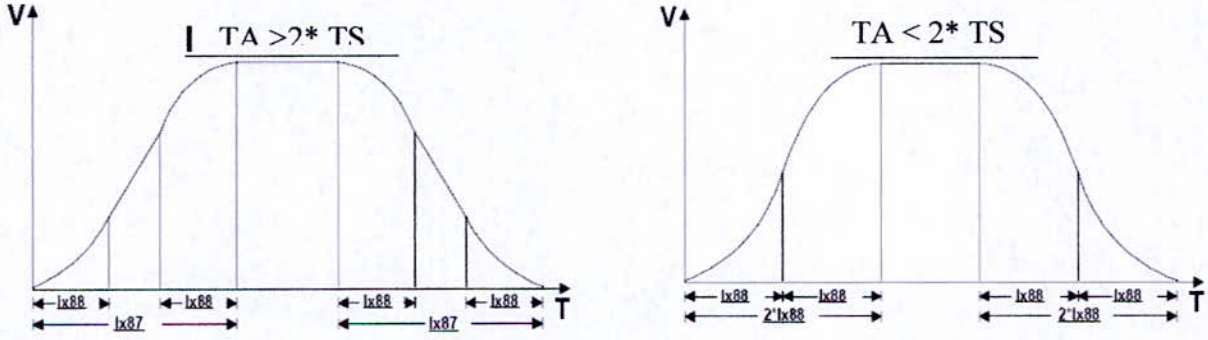


Figure III-1: *Profile de vitesse pour TA > 2*TS et TA < 2* TS*

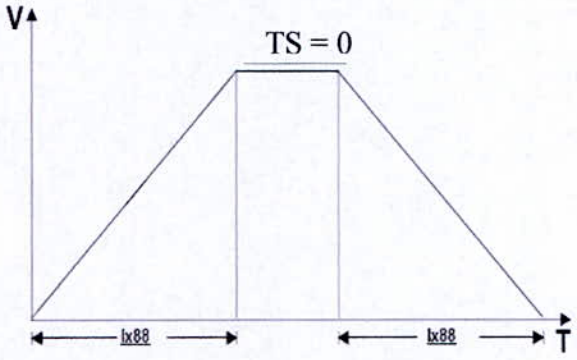


Figure III-2: *Profile de vitesse pour TS = 0*

Le temps de mouvement TM est donné directement ou indirectement par les paramètres de distance et du feedrate F (vitesse) comme illustré dans l'exemple suivant :

```
TM100
X3 Y4
```

or

```
FRAX(X,Y)
X3 Y4 F50
```

$$;TM = \frac{I190 \cdot \sqrt{3^2 + 4^2}}{50} = \frac{5000}{50} = 100 \text{ msec}$$

Pour illustrer comment UMAC fusionne les mouvements linéaires, une série de profils vitesse *f (temps)* sont montrées ci-dessous. Les mouvements sont définis sans s-curve, mais ce concept peut être utilisé pour les mouvements linéaires avec s-curve.

Considérons le programme de mouvement suivant :

```
close
delete gather
undefine all
&1
#1->2000x
OPEN PROG 1 CLEAR
LINEAR      ; mode linéaire
INC         ;  mode Incrémental
TA100       ; le temps d'accélération est 100 msec, TA1
TS0         ; pas de S-curve
TM250       ; temps de mouvement  250 msec, TM1
X10         ; distance 10 units, 20000 comptes
TA250       ; Acce \ decel 250 msec , TA2
X40         ;  distance 40 unités, 80000 comptes
CLOSE
```

Ces deux mouvements sont dessinés sans fusionnement en plaçant une commande DWELL0 entre eux:

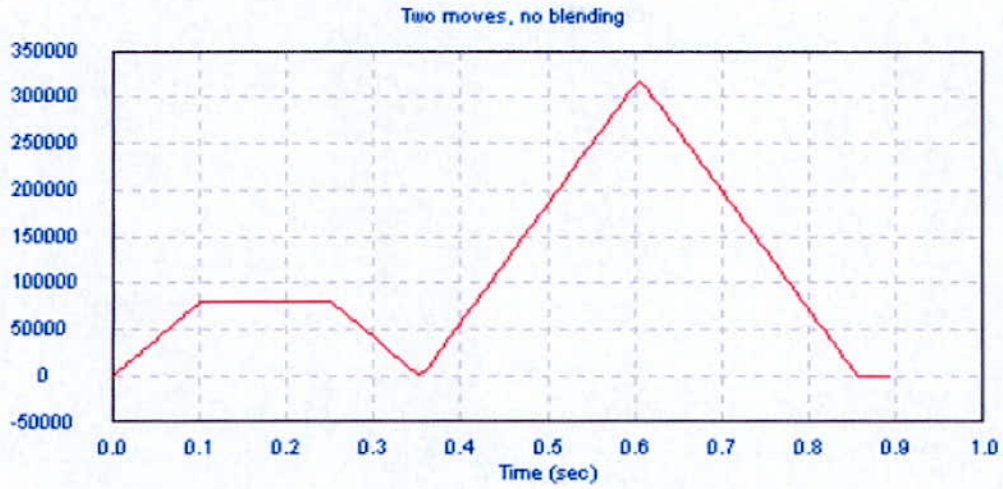


Figure III-3 : *Evolution de la vitesse (sans fusionnement)*

Les deux mouvements sont maintenant dessinés avec fusionnement. Pour trouver le point de fusionnement on trace des lignes droites à travers le milieu des lignes d'accélération de chaque profil de vitesse comme montré ci-dessus :

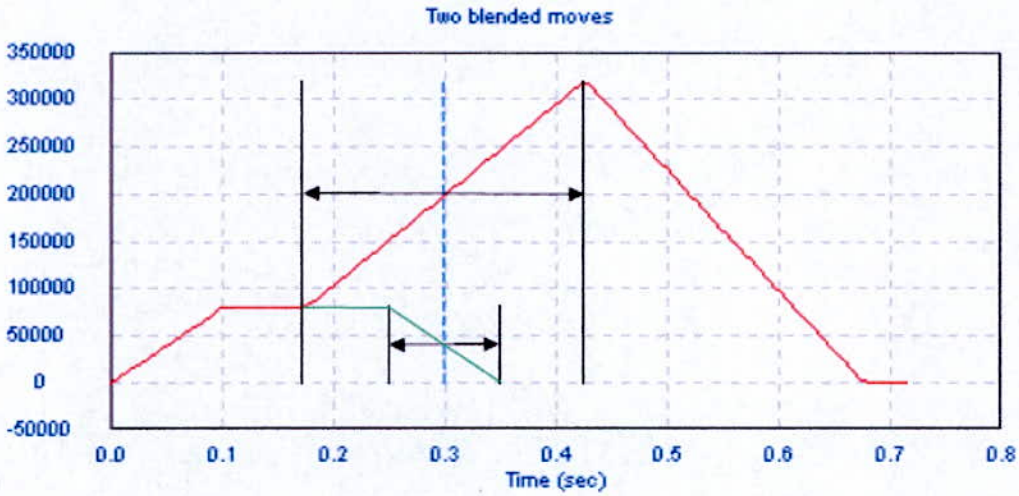


Figure III-4 : *Evolution de la vitesse (avec fusionnement)*

Observations: Le temps total de mouvement est donné par:

$$\frac{TA_1}{2} + TM_1 + TM_2 + \frac{TA_2}{2} = 50 + 250 + 250 + 125 = 675 m\sec$$

L'accélération du second mouvement peut être étendue a une certaine limite, 2*(TM- 1/2TA)

Remarque 1 :

La commande DWELL est l'équivalent d'une demande attente ou pause, une commande DWELL 10, implique que le processeur attend 10 millisecondes avant que le prochain mouvement soit exécuté.

Remarque 2:

La fonction *lookahead* dans le PMAC permet à l'interface de voir deux mouvements en avant de celui qui est entrain d'être exécuté, ce qui permet à la carte de recalculer à chaque fois ces mouvements afin de maintenir l'accélération en dessous de la limite Ixx17. Cependant, il y a des cas ou plus de deux mouvements doivent être recalculés afin de garder l'accélération sous la limite. En effet, un mauvais paramétrage de TM et TA peut provoquer un dépassement de limites d'accélération

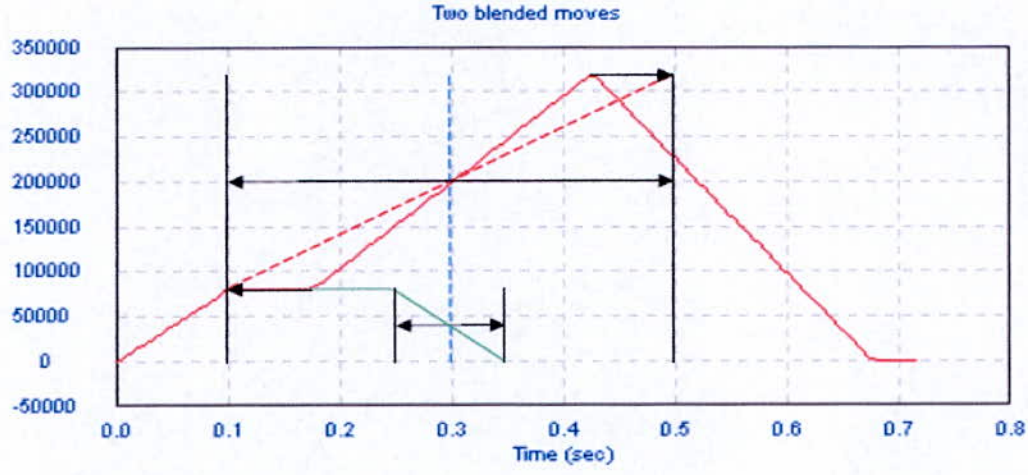


Figure III-5 : *Evolution de la vitesse lors d'un choix inadéquat de TM*

Avantages d'employer les S-Curve :

Dans un moteur électrique, la force ou le couple est proportionnel au courant appliqué à ses enroulements : $F/T = K_T \cdot I$

En outre, selon la loi de newton : $F = m\,a$ ou $T = J\,a$

Les mouvements avec accélération constante (aucune S-curve) causent des changements instantanés ou brusques des accélération et ainsi pour la force/couple et le courant du moteur. Le composant d'accélération S-curve, a comme effet de diminuer la pente ou d'assouplir les profils du courant (couple) en fonction du temps :

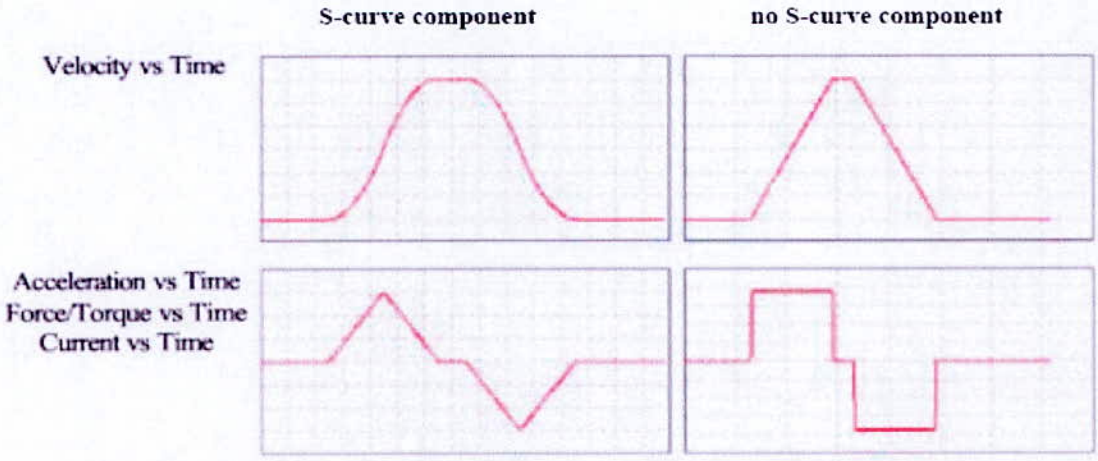


Figure III-6 : *Evolution de l'accélération avec/sans S-curve*

En outre, la S-curve peut être vue comme un filtre passe-bas éliminant les hautes fréquences. Puisque aucune structure n'est parfaitement immobile, toute structure à une certaine haute fréquence, qui sera excitée si la trajectoire contient de l'énergie à cette fréquence. L'accélération avec S-curve, en réduisant les composants à haute fréquence de la trajectoire, réduit l'excitation de ces noeuds résonnants.

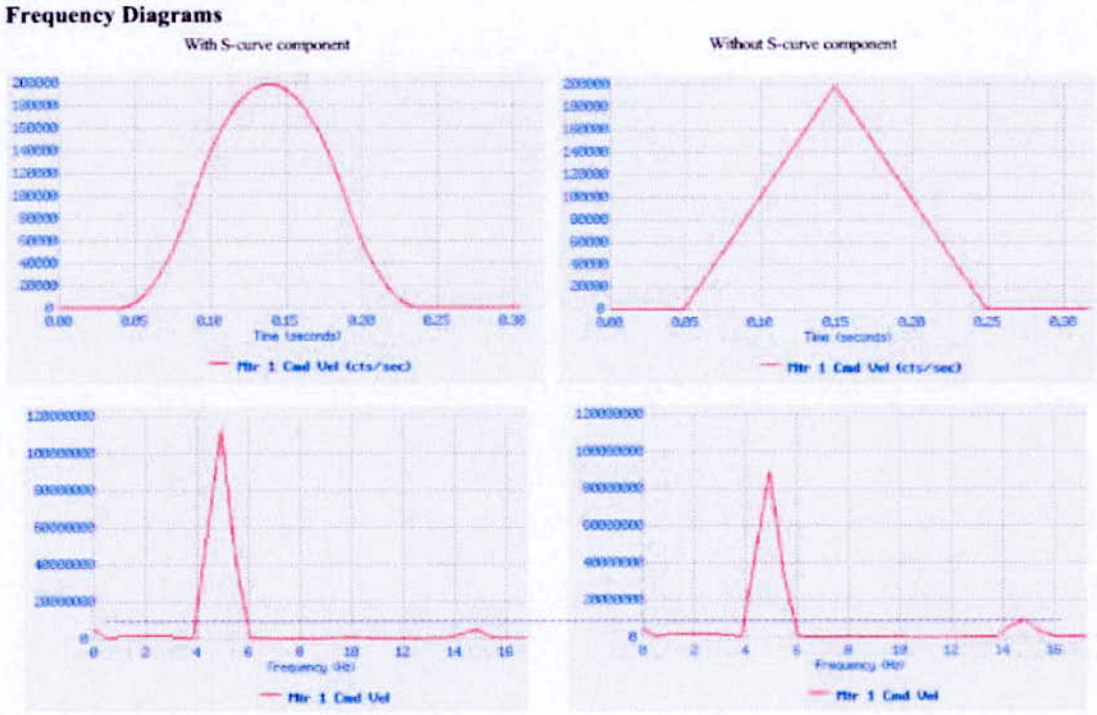


Figure III-7 : *l'effet des S-curve sur la fréquence*

III.I.3- Interpolation circulaire :

PMAC permet l'interpolation circulaire sur les axes X, Y, et Z dans un système en coordination. Comme avec des mouvements linéaires, **le TA** et **TS** commandent l'accélération à partir et jusqu'à un arrêt, et entre les mouvements. Les mouvements circulaires peuvent être (F) vitesse indiqué ou temps indiqué (**TM**), comme les mouvements linéaires. Il est possible de changer dans les deux sens entre les mouvements linéaires et circulaire sans s'arrêter. Ceux ci s'accomplissent en écrivant la commande **LINÉAIRE** quand l'interpolation linéaire est nécessaire et **CIRCLE1** ou **CIRCLE2** pour l'interpolation circulaire.

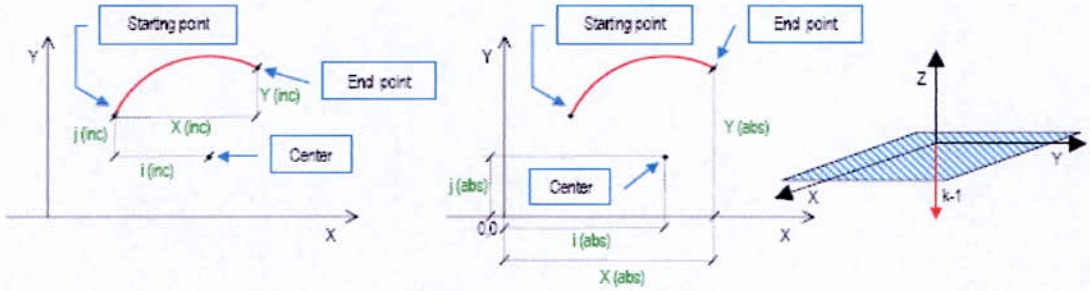


Figure III-8 : *Le mouvement circulaire*

Génération de la trajectoire par le PMAC :

Des paramètres indiqués pour le morceau de mouvement, et la position et la vitesse initial (de l'extrémité du morceau précédent), PMAC calcule le seul chemin du troisième ordre de la trajectoire pour rencontrer les contraintes (fusionner avec le mouvement précédent). Ceci a comme conséquence, l'accélération changeante linéairement, un profil parabolique de vitesse, et un profil de position cubique (polynôme d'ordre trois) pour le segment.

Dans ce qui suit nous allons énumérer les différentes étapes ainsi suivie pour l'élaboration de la trajectoire circulaire.

1) PMAC génère des mouvements d'arc en segmentant l'arc et en exécutant le meilleur ajustement cubique sur chaque segment. La variable I13 détermine le moment pour chaque segment. I13 doit être supérieur à zéro, pour mettre PMAC dans ce mode de segmentation et afin que le déplacement en arc se fait. Si I13 est placé à zéro, les mouvements circulaires seront faits en mode linéaire. La gamme pratique d'I13 pour le mode circulaire d'interpolation est 5-10 millisecondes. Une valeur de 10 millisecondes est recommandée pour la plupart des applications, si la valeur I13 est un peut inférieur à 10 milliseconde, ceci

améliorera l'exactitude de l'interpolation (le calcul des points de la courbe se fait plus souvent) mais consommera également plus de la puissance arithmétique du PMAC.

2) Quand PMAC segmente automatiquement les mouvements (I13 > 0), qui est exigé pour l'interpolation circulaire, les limites des accélérations Ix17 et les limites de la vitesse Ix16. ne sont pas observées

3) Tous axes utilisés dans l'interpolation circulaire sont automatiquement des axes de vitesse pour des mouvements circulaires, même si elles n'étaient pas ainsi indiquées dans une commande **FRAX**. D'autres axes peuvent ou peuvent ne pas être des axes de feedrate. Tous les axes de *non feedrate* commander pour se déplacer avec la même commande de mouvement sera linéairement interpolées afin de finir au même temps. Ceci permet l'interpolation hélicoïdale facile.

4) Le plan pour l'arc circulaire doit être défini par la commande **NORMALE** (le défaut -- **K-1 NORMAL** -- définit le plan XY). Cette commande peut seulement définir des plans dans l'espace XYZ, qui signifie que seul les axes X, Y, et Z peuvent être utilisées pour l'interpolation circulaire. D'autres axes indiqués dans la même commande de mouvement seront interpolées linéairement pour finir au même temps. Les plans généralement utilisés sont :

K-1 NORMAL ; plan XY
J-1 NORMAL ; plan ZX
I-1 NORMAL ; plan YZ

5) Pour mettre le programme dans le mode circulaire, on emploi la commande **CIRCLE1** pour les arcs dans le sens des aiguilles d'une montre ou **CIRCLE2** pour les arcs dans le sens contraire des aiguilles d'une montre. La commande **LINÉAIRE** restaure les mouvements mélangés linéaires. Un mouvement circulaire est indiqué avec une commande de mouvement indiquant le point final du mouvement et le vecteur au centre d'arc ou la distance (rayon) au centre. Le point final peut être indiqué comme position ou comme distance du point de départ, selon si, les axes sont en mode absolu (**ABS**) ou en mode incrémental (**INC**) (individuellement spécifiable).

X{Data} Y{Data} R{Data} ; Le Rayon du cercle est donné

X{Data} Y{Data} I{Data} J{Data} ; les coordonnées du Centre de cercle sont données

6) si la méthode de localisation du centre d'arc est employée, le vecteur est indiqué par ses composantes I, J, et K ('I 'indique la composante parallèle à l'axe X, J à l'axe Y, et K à l'axe 'Z '). Ce vecteur peut être indiqué comme distance du point de départ (c.-à-d. incrémentale), ou de l'origine de XYZ (c.-à-d. absolu).Le choix est fait en indiquant 'R 'dans un rapport d' ABS ou d' INC. (Par exemple **ABS(r)** ou **INC(r)**). Ceci affecte des spécificateurs de I, de J, et de K ensemble (les **ABS** et **INC** sans arguments affectent toutes les axes, mais laissent les vecteurs inchangés).Le défaut est avec des spécifications du vecteur relatif.

7) la convention de PMAC doit prendre le chemin le plus court d'arc si la valeur de **R** est positive, et le long chemin d'arc si **R** est négatif :

- si la valeur de R est positive, l'arc au point final de mouvement est l'itinéraire court (< = 180 degrés)
- si la valeur de R est négative, l'arc au point final de mouvement est le long itinéraire (> = 180 degrés)

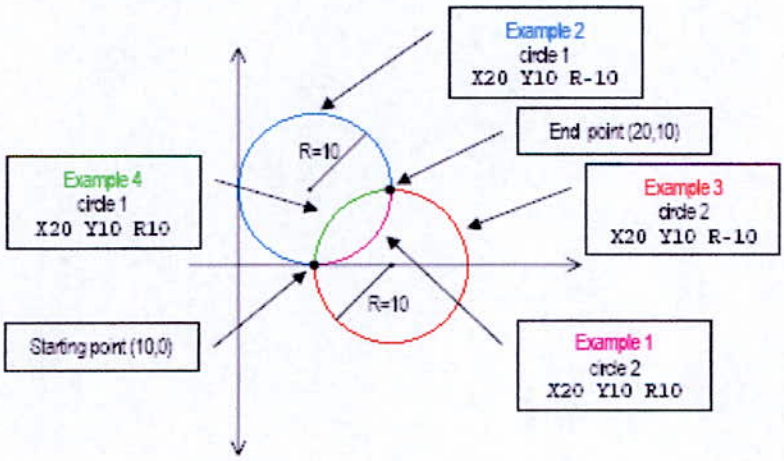


Figure III-9 : *Sens de rotation*

8) En effectuant une interpolation circulaire, les différents axes décrivent une position vs temps près d'une forme d'un sinus ou d'un cosinus. Cela est également vrai pour leurs profils de vitesse et d'accélération. Par conséquent, l'interpolation circulaire est un dispositif idéal aux profils trigonométriques. De plus, la période (et ainsi la fréquence) des ondes du sinus ou du cosinus pourraient être décrites par le temps de mouvement indiqué par TA+TM.

III.I.4- Mouvements Spline :

PMAC peut exécuter les Splines cubiques (cubiques en termes de position-vs-temps,) pour mélanger ensemble une série de points sur un axe. Le mouvement Spline est en particulier convenu pour les géométries « impaires » (non cartésiennes), telles que les tables radiales et les robots d'axe rotatif, où il y a des formes de profils impaires de même axe pour les mouvements réguliers.

On distingue dans le mouvement Spline deux types, a savoir : **SPLINE1** et **SPLINE2**

Fonctionnement :

En mode **SPLINE1**, un long mouvement est coupé en segments égaux en temps, chacun est de temps TA. Chaque axe donne une position de destination dans le programme du mouvement pour chaque segment avec une ligne de commande comme **X1000Y2000**. PMAC crée une courbe cubique de position-vs-temps pour chaque axe de sorte qu'il n'y ait aucun changement soudain de vitesse ou d'accélération aux extrémités du segment. La position commandée à l'extrémité du segment peut "être détendue" légèrement pour rencontrer la vitesse et l'accélération. PMAC peut fonctionner seulement avec des valeurs entieres (milliseconde) pendant les temps de segment TA. Si une valeur non entière est indiquée pour le temps TA, PMAC l'arrondira automatiquement au nombre entier le plus proche. En arrondissant, les vitesses et les temps pour la trajectoire changera. PMAC calcule les "*points d'orientation / way point* " pour chaque axe et pour chaque point le long de la Spline en prenant une moyenne pesée du point indiqué X(n) et les points indiqués de chaque côté, selon l'équation :

$$WP(n) = \frac{X(n-1) + 4X(n) + X(n+1)}{6} \qquad \textbf{(III-1)}$$

PMAC calcule également la vitesse pour chaque axe à chaque point le long de la Spline en prenant la vitesse à mi-chemin entre les vitesses moyennes des segments de chaque côté du point de direction :

$$V(n) = \frac{[X(n+1) - X(n)] + [X(n) - X(n-1)}{2TA} = \frac{X(n+1) - X(n-1)}{2TA} \qquad \textbf{(III-2)}$$

Une fois le calcul des positions et des vitesses exactes aux frontières du segment est effectué, PMAC calcule l'équation cubique unique de position (profil parabolique de vitesse) qui assure la satisfaction des contraintes, et son utilisation pour l'interpolation. Le temps du segment ne peut être changé en marche dans le mode **SPLINE1**.Si le temps du segment est changé au milieu d'un ordre

de mouvement, PMAC apportera automatiquement la partie précédente à un arrêt, et commence alors la section suivante avec le nouveau temps de segmentation. Si les temps du segment sont petits, l'opération sera très approximative.

III.I.5- Spline Non uniforme :

Le mode **SPLINE2** est très semblable au mode **SPLINE1**, sauf que la condition que le temps du segment de Spline TA est variable. Le déplacement de cette contrainte fait du mode **SPLINE2** une B-Spline cubique non rationnelle et non uniforme, tandis que le mode **SPLINE1** est une B-Spline cubique uniforme non rationnelle. Les spécifications "non rationnelle" indiquent qu'il n'y a aucune pondération indépendante (rapports) des différents points dans la spline.

La liberté supplémentaire du temps non uniforme du segment rend le mode **SPLINE2** plus flexible que le mode **SPLINE1**, mais au coût d'un temps supplémentaire de calcul, d'environ 20%. Le mode **SPLINE2** est toujours plus efficace que n'importe quel mode de calcul non-Spline. Si TA est tenu constant, le mode **SPLINE2** produit une trajectoire qui est identique au mode **SPLINE1**. Le segment supplémentaire au début d'une Spline a le même temps que le premier segment programmé ; De même pour le segment supplémentaire à l'extrémité d'une Spline La période combinée de trois segments consécutifs quelconques dans une cannelure **SPLINE2** continue doit être moins de 8.388.608 millisecondes, ou environ 2 heures et 20 minutes.

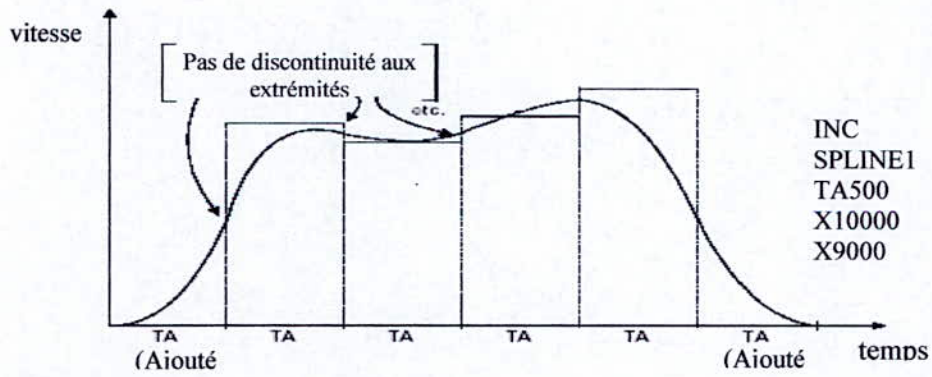


Figure III-10 : *Mouvement Spline-1 programmé pour deux segments*

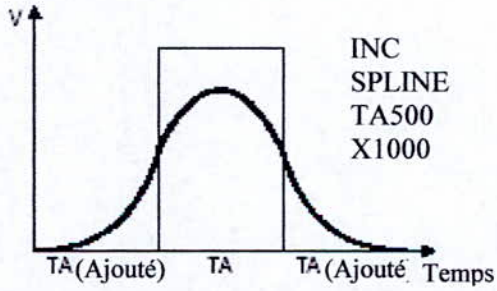


Figure III-11: *Mouvement Spline-2 programmé pour un segment*

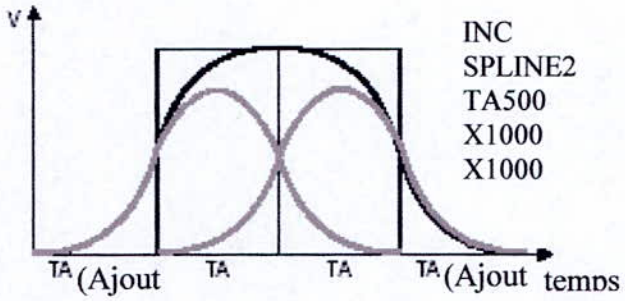


Figure III-12: *Mouvement Spline-2 programmé pour deux segments*

III.I.6- Mode PVT :

Pour l'utilisateur qui désire un contrôle plus direct du profil de la trajectoire, PMAC offre des mouvements en mode Position-Vitesse-Temps (PVT). Dans ces mouvements, l'utilisateur indique les états d'axe directement aux transitions entre les mouvements (à la différence des mouvements composer). Ceci exige plus de calcul par la carte, mais permet une commande plus serrée de la forme du profil. Pour chaque morceau d'un mouvement, l'utilisateur indique la position ou la distance de fin, la vitesse de fin, et l'intervalle en temps du mouvement. PMAC est mis dans ce mode avec le report dans le programme du **PVT {data}**, où **{data}** est une constante, variable, ou expression, représentant le temps ou l'intervalle du mouvement en millisecondes. Cette valeur devrait être un nombre entier ;si elle ne l'est pas, PMAC l'arrondi au nombre entier le plus proche. Le temps du mouvement peut être changé, avec une autre commande **PVT {data}**, ou avec **TA {data}**.Le programme est pris hors de ce mode avec une autre commande d'un mode de mouvement (par exemple **LINÉAIRE, RAPIDE, CERCLE, SPLINE**).

Un mouvement en mode PVT est indiqué pour chaque axe à déplacer avec une expression de la forme **{axis} {data}:{data}**, où **{axe}** est une lettre indiquant l'axe, la première expression **{data}** est une valeur indiquant la position de fin ou la distance du segment, selon si l'axe est dans le mode absolu ou le mode incrémental, respectivement, et la seconde **{data}** est une valeur représentant la vitesse de fin.

Les unités pour la position ou la distance sont les unités de longueur ou d'angle utilisées pour l'axe, comme placées dans l'expression de définition d'axe. Les unités pour la vitesse sont définies comme unités de longueur divisées par des unités de temps, où les unités de longueur sont identiques à celle de position ou de distance, et les unités du temps sont définies Ix90 pour le SC (unités de temps de feedrate).La vitesse indiquée pour un axe est une quantité signée.

A partir des paramètres indiqués pour le segment de mouvement, de la position et de la vitesse commençante (de l'extrémité du morceau précédent), PMAC calcule le seul chemin du troisième ordre de la trajectoire de position pour rencontrer les contraintes. Ceci a comme conséquence l'accélération linéairement changeante, un profil parabolique de la vitesse, et un profil cubique de la position pour le segment de mouvement.

Puisque l'utilisateur peut indiquer (directement ou indirectement) une vitesse de fin différente du zéro pour le mouvement, ce n'est pas une bonne idée de faire un programme de mouvement par transition de point, et le grand soin doit être pris en téléchargeant ces mouvements en temps réel. Avec l'utilisation des expressions **BLOCKSTART** et **BLOCKSTOP** entourant une série de mouvements PVT, dont le bout de la fin du mouvement a une vitesse nulle, il est possible d'employer une commande d'étape et d'exécuter ainsi seulement une partie d'un programme.

Le mode de PVT est le plus utile pour créer des profils de trajectoires arbitraires. Il fournit une approche "*Building Block*" rassemblant des segments paraboliques de vitesse pour créer les profils désirés.

Les diagrammes des segments en mode PVT, ci-dessous, montrent des profils communs de segment de vitesse. Le mode PVT peut créer n'importe quel profil de mouvement.

Le mode PVT fournit d'excellentes possibilités de contournement, parce qu'il prend le chemin commandé interpolé exactement par points programmés. Il crée un chemin connu sous le nom de "SPLINES de Hermite". Les modes **LINÉAIR** et **SPLINE** sont des B-SPLINES de 2èmes et 3émes ordre, respectivement, qui passent à l'intérieur des points programmés.

Comparé au mode SPLINE du PMAC, PVT produit un profil plus précis.

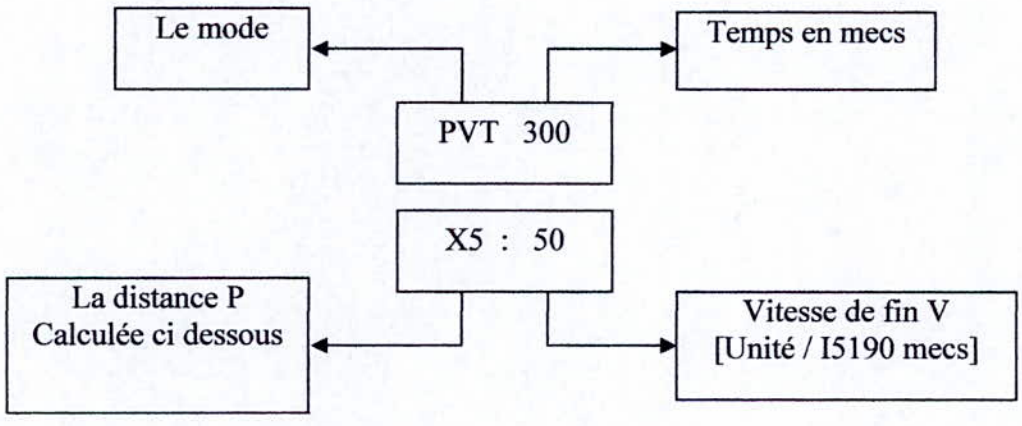


Figure III-13: *Exemple de la syntaxe de PVT*

Les différents profils des trajectoires mode PVT :

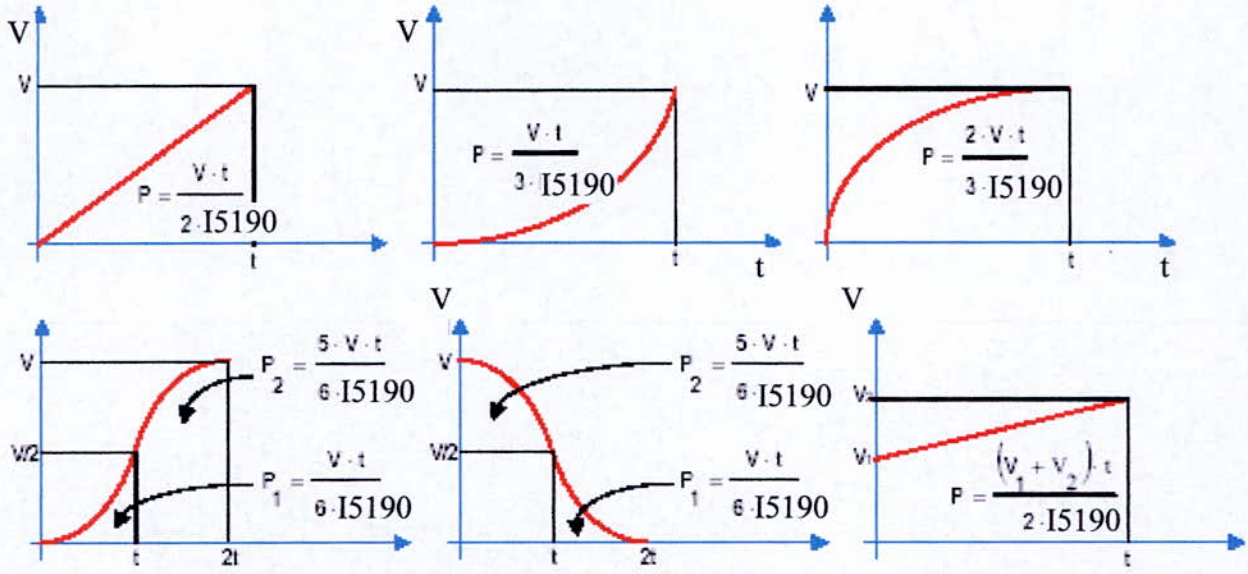


Figure III-14: *Méthode de calcule de la position P pour le mode PVT*

Remarque : Remplacez I5190 pour la variable I5x90 appropriée pour déterminer la base du temps du système en coordination x.

III.I.7- Conclusion :

Ce chapitre a permis de présenter plusieurs méthodes de génération de mouvement utilisées par l'interface.

On a présenté en premier lieu le mouvement linéaire entre deux point dont le profil de vitesse est en trapèze, ensuite les mouvements linéaires mélangés entre plusieurs points, ici on distingue deux cas : soit avec un arrêt à l'extrémité de chaque segment ou sans arrêt. Dans le premier cas on revient au mouvement linéaire entres deux point. Dans le second, les profils de vitesse sont modifiés de sorte a avoir une continuité de mouvement.

L'introduction de la s-curve permet l'assouplissement des profils d'accélération. Ensuite nous avons présenté l'interpolation circulaire car l'interface peut exécuter des mouvements circulaires entre deux points avec le choix du rayon.

Dans l'interpolation spline un long mouvement est découpé en segments de durée TA, les profils de vitesse sont des B-spline cubiques assurant une continuité en accélération a l'extrémité des ces segments.

Et en fin l'interpolations (PVT) position vitesse temps, on a un meilleurs contrôle sur le profil de la trajectoires et de vitesse.

CHAPITRE :

III

III- GÉNÉRATION DE MOUVEMENT DANS L'UMAC

III-II- Programme de Mouvement : [20]

C'est un programme qui contient des informations et des lignes de commande où le processeur de l'UMAC peut les utilisés pour planifier et déclencher le mouvement, l'interface peut supporter jusqu'à 256 programmes de mouvement en même temps (avec extension de 7 autres cartes Turbo PMAC2- 3U). N'importe quel système en coordination peut exécuter l'un de ces programmes à tout moment, même si un autre système en coordination exécute déjà ce même programme.

Un programme de mouvement peut appeler n'importe quel autre programme de mouvement comme sous-programme, avec ou sans arguments. Le langage des programmes de mouvement peuvent être mieux décrit comme un couplage entre un langage de programmation évolué (comme le BASIC ou le Pascal), et un langage machine-outil (G-Code ou D-Code).

III.II.1- Ecriture d'un programme de mouvement :

Dans ce qui suit on va procéder étape par étape dans la création d'un programme de mouvement :

- Ouvrir un buffer pour le programme par la commande OPEN PROG **{constante}**, ou **{constante}** est un entier de 1 jusqu'à 32767 et qui représente le programme de mouvement à ouvrir ;
- Les programmes de mouvement 100, 1001, 1002, et 1003 peuvent contenir des G-code, T-code et D-code pour les machines a outil (G-code ou RS-274 comme méthode de programmation). Néanmoins ces buffers peuvent être utilisés pour des programmes de mouvement avec les commandes usuelles du l'UMAC ;

- PMAC peut contenir jusqu'à 256 programmes de mouvement en même temps. Pour des programme excessivement long et dont la mémoire du PMAC ne peut les contenir, un programme spécial qui est PROG0 peut être utiliser et qui permet le chargement du programme au fur et a mesure que le programme est exécuté ;
- La commande CLEAR vide le buffer qui est ouvert (PLC, rotary. etc) ;
- Plusieurs expressions dans les programmes de mouvement du PMAC sont modal par nature, ceci inclus les modes de mouvement, lesquelles spécifient quel type de trajectoire que la commande en mouvement sera généré ; ces commandes sont LINEAR, RAPID, CIRCLE, PVT et SPLINE ;
- Le mouvement peut être spécifié si il est incrémental (distance) ou absolu (localisation)- sélectionner individuellement par axe- avec les commandes INC et ABS.
  Si les temps de mouvement **TA** (temps d'accélération), **TS** (temps de courbure pour le mode S-Curve), **TM** (temps de mouvement), et/ou la vitesse **F** (Feedrate) ne sont pas déclaré, les valeurs par défaut sont fournies par les paramètres Ixx87, Ixx88, et Ixx89.
  Cependant, il est recommandé de ne pas compter sur ces paramètres et à déclarer les temps de mouvement dans le programme. Ceci permettra une meilleure maintenance du programme
- Les commandes de mouvement elle-même consiste en un axe spécifié suivis d'une ou de deux valeurs (constante ou expression). Tous les axes spécifiés dans la même ligne dans un programme de mouvement vont être exécuté simultanément dans le SC. Cependant dans le cas de lignes consécutives, les commandes seront exécuter séquentiellement. Toutefois dépendantes du mode du mouvement affecter, la valeur spécifier peut être une destination, distance et/ou une vitesse ;
- Certaines commandes de test comme la boucle WHILE et IF. .ELSE peuvent être utilisées. De même que les commandes de branchement comme GOTO (branchement vers une étiquette), GOSUB (branchement vers un sous programme), ainsi que la commande CALL qui permet de faire appelle a un autre programme ou une partie seulement de ce dernier comme sub-programme (ex : CALL 20.150 ; fait appelle à la ligne 150 du programme 20) ;
- La commande CLOSE est impérativement utilisé dans tous les programmes afin de fermer le buffer ouvert pour le programme en cours de même il est préférable de débuter tout programme par un close est de le finir de même.

Remarque :

Dans les programmes de mouvement le PMAC ne fait pas de différence entre majuscule et minuscule dans l'écriture des commandes.

Exemple :

```
CLOSE                  ; ferme tous les buffers ouverts.
DELETE GATHER          ; Effacer toutes les données intermédiaires.
UNDEFINE ALL           ; Effacer les définitions lié a tous les système en coordination.
#1-> 363.889X          ; lier le moteur #1 a l'axe X avec une graduation d'axe en degré
                       ; 65.5*2000 / 360 = 363,889.
OPEN PROG 1 CLEAR      ; Ouvrir le buffer pour écrire le programme.
LINEAR                 ; interpolation Linéaire (le profil de vitesse est linéaire).
INC                    ; mode Incrémentale.
TA100                  ; temps d'accélération est 100 mecs.
TS0                    ; pas de courbure dans le profile de vitesse.
F50                    ; le Feedrate est 50 unités par I5x90 msec.
X180                   ; 180 unités de distance : 180°.
CLOSE                  ; ferme le buffer, programme 1.
```

I5x90 : doit être initialisé à 1000 pour avoir une échelle de temps en msec pour le système en coordination x. Le temps de mouvement TM est donné directement dans le programme, ou indirectement en se basant sur la distance de l'axe, et la vitesse désirée (feedrate).

Exemple :

```
TM500      ou bien      X5F10          ; Vx = 0.5µs
X5
```

Exemple pour deux axes :

```
TM500      ou bien      INC            ; Distance = SQRT (3^2 + 4^2) = 5
X3Y4                    FRAX(X, Y)     ; TM = 5/10 = 0.5 µs
                        X3 Y4 F10      ; Vx = 3/0.5 = 6
```

; Vy = 4/0.5 = 8

III.II.2- Déroulement De L'exécution Des Programmes De Mouvement Dans UMAC :

Fondamentalement, un programme de mouvement existe pour que le processeur fait passer les données de ce dernier vers les routines de générateur de trajectoire qui calculent la série de positions de référence commandée pour les moteurs à chaque cycle Servo, donc un programme de mouvement doit fonctionner en tête du mouvement commandé actuel pour maintenir l'alimentation des générateurs de trajectoires avec les données.

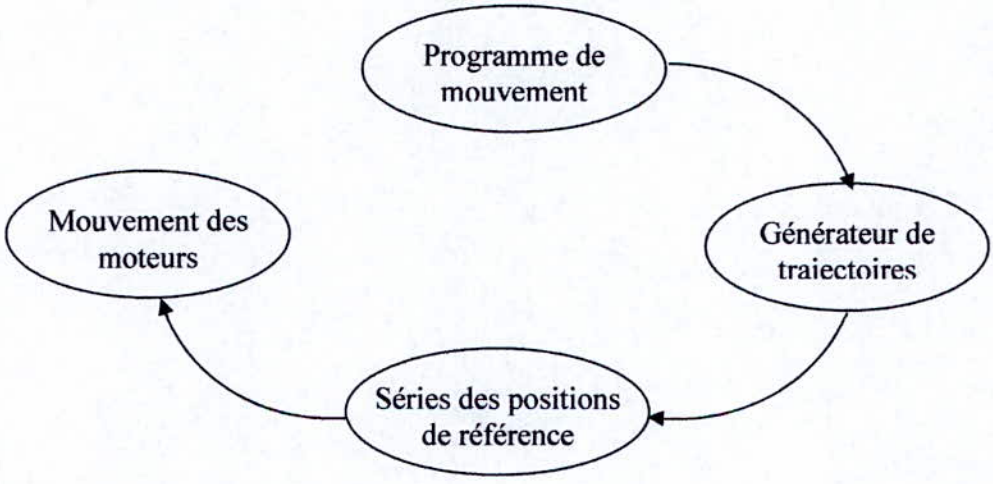


Figure III-15 : *Principe d'interprétation de programme mouvement*

Le processeur traite les lignes du programme de façon à ce qu'un ou deux mouvements soient calculés en avance, ce qui est nécessaire afin de pouvoir composer des mouvements (pour prévoir les limitations appropriées d'accélération et de vitesse qui doivent être faites), et puisque c'est le cas pour les mouvements linéaires composés (I13 doit être égale à 0 : temps de segmentation de mouvement nul) où le processeur calcule deux mouvements à l'avance, et dans tous les autres cas (I13 > 0) il calculera un seul mouvement à l'avance.

Remarque : S'il y'a une singularité au cours du calcul de trajectoire, le processeur ne peut pas finir le calcul, il abandonnera le programme avant que l'exécution de ce mouvement soit censée commencer, en envoyant une erreur d'exécution temporelle à l'ordinateur principale.

III.II.3- Définition d'axe :

Un axe de point de vue logiciel est un élément important des systèmes en coordination, il se réfère à un moteur par une lettre alphabétique sélectionnée parmi X, Y, Z, A, B, C, U, V, et W de telle façon à ce qu'une affectation d'une valeur à cette lettre corresponde à un déplacement du moteur.

Un axe est défini en l'affectant à un moteur avec un facteur de graduation et un offset (notons que X, Y, et Z peuvent être définis en tant que combinaisons linéaires de trois moteurs, de même que pour U, V, et W).

La plus simple définition est : #1->X (attacher le moteur #1 à l'axe X pour le système en coordination courant), maintenant si l'axe X exécute un mouvement, le moteur #1 fera le mouvement. Pour affiner une application (de point de vue réglage et précision) il faut définir les caractéristiques d'axe suivantes :

- Graduation d'axe.
- Offset.
- Type d'axe.

III-II-3-1 Graduation d'axe :

Dans cette partie, on définit une nouvelle unité d'utilisation (en se basant sur un certain nombre d'incrémentations de l'encodeur) qui sera la nouvelle résolution de déplacement du moteur.

Exemple : Nous disposons d'un encodeur de 500 CPR (compte par révolution) avec un moteur dont le rapport de réduction est $\eta = 65.5$. Etant donné que la résolution est augmentée de 4 fois par l'interface (suivant la configuration de I7mn0), ce qui signifie qu'on a 2000 compte par tour

d'encodeur, alors une rotation complète de l'arbre du moteur corresponde à : η * 2000 = 131000 comptes.

Ce qui fait que l'affectation « #1->131000X » permet de travailler avec une unité de déplacement du moteur #1 égale à un tour.

III-II-3-2 Offset :

Par exemple le rapport #1->10000X+20000 place un offset (de 2 unités d'utilisateur) de la position zéro du moteur #1, notons que cet offset est rarement utilisé.

III-II-3-3 Type d'axe :

Pour la définition de ce type d'axe, il peut y avoir deux attributs géométriques où chacun correspond à un mode de mouvement dans l'espace, le troisième type est une propriété :

- Les axes cartésiens.
- Les axes de rotation.
- Les axes fantôme.

Notons que pour la plupart des applications, le choix du type d'axe n'est pas nécessaire. Cependant, pour certaines applications comme l'implémentation du modèle géométrique d'un robot, on ne peut utiliser que les noms particuliers X, Y, Z, U, V et W qui correspondent aux axes cartésiens.

a) Axes cartésiens :

Ce type d'axe peut être groupé suivant une combinaison linéaire de deux ou trois axes pour causer le mouvement de deux ou de trois moteurs : X, Y, Z forment la première base, et U, V, W forment la seconde base.

Exemple : #1->8660.25X-5000Y ; avec : cos (30°) = 0.866 et sin (30°) = 0.5.
#2->5000X+8660.25Y

Dans ce cas le mouvement d'un seul axe X ou Y cause le mouvement des deux moteurs avec un écart final de 30° entre les deux axes des moteurs.

Remarque : On ne peut avoir d'autres combinaisons entre les axes. C'est-à-dire non- linéaire. Cependant, les relations non- linéaires qui peuvent exister entre les axes se feront dans des programmes spéciaux.

b) Axes de rotation (A, B, et C) :

Quand on affecte un moteur à ces axes, la fonction *rollover* du moteur est activée, et quand le programme fonctionne en mode absolu (ABS), cette fonction permet au moteur de prendre la plus courte distance du point de départ au point destiné au tour d'un rang spécifié par Ixx27.

Exemple : I127 = 131000 compte par tour

#1->363.889A ; A est dans les unités des degrés

Dans ce cas, si on applique la commande de mouvement 'A270' dans un programme en mode ABS, le moteur fait le mouvement de 0 à -9000 (-90°), au lieu de faire le mouvement de 0 à 27000 (ce qui l'aurai fait avec I127 = 0).

Remarque : Contrairement aux axes linéaires, ces axes ne peuvent pas être dans une combinaison linéaire.

c) Axes fantôme :

Un axe peut ne pas avoir aucun moteur attaché ''un axe fantôme'', bien que les mouvements programmés pour cet axe ne causent aucun mouvement, le fait qu'un mouvement a été programmé pour cet axe, il peut affecter les mouvements d'autres moteurs. Par exemple, si on désire d'avoir des profils sinusoïdaux sur un axe simple, la manière la plus facile de le faire, est d'avoir en second lieu un axe fantôme et de programmer des mouvements d'interpolation circulaire.

III.II.4- Variables de travail :

Ces variables permettent à l'utilisateur de programmer et de configurer des applications sans avoir à s'encombrer avec les spécifications des adresses. Elles se divisent en quatre classes à savoir :

- **Variables I :** c'est des variables d'initialisation et d'installation qui déterminent la personnalité de la carte mère de l'UMAC et ses accessoires, ces variables sont au nombre de 8191 dont on peut citer les gammes les plus importantes :

- **I0 --- I99 :** variables d'installation globale de l'interface
- **Ixx00 ---- Ixx99 :** configuration du moteur xx pour les applications désirées
- **I7mn0---- I7mn9 :** configuration du canal « n » de la DSPGATE « m »

➢ **Variables P :** sont des variables d'usage universel pour l'utilisateur, elles sont des variables réelles stockées dans des cases mémoire dans l'interface sur 48 bits pour un total de 1024 variables

➢ **Variables Q :** les variables Q, sont comme les variables P. cependant la signification de la variable Q dépend de quel système en coordination elle y est utilisée

➢ **Variables M :** Elles permettent l'accès faciles de l'utilisateur à la mémoire interne de l'interface et les registres d'entrée/sortie. Elles sont des pointeurs vers des locations mémoires que l'utilisateur peut définir la taille, le format et la valeur attribuée à cette location.

III-III- Système en coordination : [18]

Un système en coordination dans l'UMAC est le groupement d'un ou plusieurs moteurs dans le souci de synchronisation des mouvements. Un SC (même avec un seul moteur) peut exécuter un programme de mouvement ; ce que le moteur ne peut faire à lui seul. L'UMAC peut avoir jusqu'à 8 systèmes en coordination, désignés par &1 jusqu'à &8, d'une manière flexible (ex : 8 SC d'un moteur chacun, un seul système en coordination de 8 moteurs, 4 SC de deux moteurs chacun, etc.).
En général, si on veut que certains moteurs bougent d'une manière coordonnée, on les met dans le même SC, et si on veut qu'ils effectuent les mouvements indépendamment les uns des autres on les met dans différents SC. Les différent SC peuvent exécuter différents programmes de mouvement à des moments différents, de même ils peuvent exécuter le même programme à des moments différents.
Un système en coordination doit être établi par l'affectation des moteurs a des axes (voire définition des axes ci-dessus). Un SC doit avoir au moins un seul moteurs affecté à un axe dans ce système. Hormis ce fait, un programme de mouvement ne peut être exécuté.

III.III.1- Base de temps d'un système en coordination :

Chaque système en coordination a sa propre base de temps qui lui permet de contrôler la vitesse de l'interpolation des mouvements dans ce système en coordination. Les routines d'interpolation de mouvement incrémentent un registre « *elapsed-time* » à chaque cycle servo.

Le temps réel pour le servo cycle est configuré matériellement pour la carte (par les jumpers E98, E29-E33, et E3-E6) et ne change pas, quant à la valeur du temps ajouter au registre « *elapsed-time* » chaque servo cycle n'est qu'un nombre qui se trouve dans un registre mémoire. Il ne correspond pas au temps réel physique du cycle.

L'unité de cette base de temps est 2^{23} (8388608) = 1milliseconde. La valeur par défaut de cette unité de base de temps est égale à la valeur par défaut de la variable I10 qui est de 3713707 représentant la valeur physique du temps cycle servo 422 microsecondes.

Si la valeur du registre base de temps est changée de sa valeur dans I10, l'interpolation des mouvements se fera à une vitesse différente que celle dont le mouvement est programmés. Ceci est appelé « feedrate override ». Il faut noter que le temps physique n'est pas changé, donc les dynamiques de la boucle servo resteront inchangées.

Chaque système en coordination a une variable Ix93 qui contient l'adresse du registre que le système en coordination utilise pour sa base de temps. Avec la valeur par défaut de Ix93, le système en coordination aura les informations de sa base de temps du registre spécifié par la commande % du host. La commande %100 met une valeur égale à I10 dans ce registre.

III.III.2- La configuration des I-Variables pour un système en coordination :

En plus des différentes affectations pour les moteur (affectation de chaque moteurs à un axe), et la définition des systèmes en coordination ainsi que leurs axes, la configuration de quelques variables I est indispensable (variable d'activation des SC I68, variables de délimitation de l'espace de travail Ixx11, Ixx12, Ixx13, Ixx14), pour que le système en coordination soit effectif.

Dans ce qui suit nous citerons quelques configurations des I-Variables, et qui seront nécessaires pour une bonne configuration des systèmes en coordination, comme nous pouvons le

constater, ces variables sont liées, soit à la sécurité du système (*fatal follwing error, warning flag follwing error*) soit à la délimitation de l'espace de travail de notre système (*limites positives et négatives*).

III-III-2-1 I68 control d'activation des systèmes en coordination :

Gamme : 0 – 15

Unités : None

Défaut : 15

I68 contrôle quel SC sera activé dans le PMAC. Un SC doit être activer afin qu'il soit adressé et commandé par ce dernier, ainsi que l'affectation des variables M synchrone pour son utilisation.

I68 peut prendre comme valeurs de 0 jusqu'à 15, le plus grand nombre de SC activé est (I68 + 1), en d'autre terme, toute valeur affecté a I68 active le SC (1) jusqu'à SC (I68 + 1)

Les piles d'allocation des variables M se fait en binaire, cette allocation n'est fractionnée que par des puissances de 2. Les piles d'allocation pour chaque SC sont données par le tableau suivant :

Valeur du I68	Le plus grand nbr de SC activé	Les M-Var sync par SC
0	SC -1	256 mots
1	SC -2	128 mots
2-3	SC 3-4	64 mots
4-7	SC 5-8	32 mots
8-15	SC 9-16	16 mots

Tableau III-1 : *les valeurs de I68*

La valeur par défaut qui est I68= 15 (tous les systèmes sont activés) est toujours valable, même si le nombre de SC dont nous disposons est inférieur.

Néanmoins, une valeur inférieure à celle par défaut dans le cas où on n'utilise pas tous les SC permet d'avoir deux avantages, à savoir :

- L'augmentation et l'amélioration de l'efficience des calculs, car les systèmes en coordination qui ne sont pas activés ne vont pas être vérifié périodiquement
- Une plus grande exécution d'affectation des variables M par mouvement.

Remarque :

I68 est utilisée à chaque (mise sous tension / réinitialisation) seulement, alors pour changer le nombre des systèmes en coordination activés, on a qu'a affecté la nouvelle valeur du I68, la sauvegarder dans la mémoire volatile du PMAC (commande : save), ensuite la charger vers la carte par la commande $$$.

III-III-2-2 Limite fatale d'erreur de poursuite du moteur XX Ixx11 : (*Motor xx Fatal Following Error Limit):*

Gamme 0 – 8, 388,607

Unité : 1/16 count

Défaut : 32,000 (2000 comptes)

Ixx11 place l'amplitude limite de l'erreur de poursuite pour le moteur xx auquel ce dernier s'arrêtera. Quand l'amplitude de l'erreur de poursuite excède Ixx11, le moteur xx est arrêté. Si le système en coordination du moteur exécute un programme alors, le programme est arrêté. Il est facultatif si d'autres moteurs doivent être arrêter quand ce moteur excède son amplitude de l'erreur de poursuite limite ; les bits 21 et 22 de Ixx24 commande ce qui arrive aux autres moteurs (le défaut est que tous Les moteurs sont arrêtés).

Un bit d'état pour le moteur, et un pour le SC (si le moteur est dans un) sont placés, et mis a un si l'erreur de poursuite est atteinte.

Le réglage d'Ixx11 à zéro neutralise la limite fatale de l'erreur de poursuite pour le moteur. Ceci peut être souhaitable pendant le travail du développement initial, mais lui ***est fortement déconseillé*** dans une application réelle. Une limite fatale d'erreur de poursuite est une protection très importante contre divers types de défauts, tels que la perte de signal d'erreur, celle-ci ne peut pas être détectée directement, et cela peut endommager considérablement l'équipement

III-III-2-3 Ixx12 Motor xx Warning Following Error Limit:

Gamme : 0 – 8.388.607

Unités : 1/16 compte

Défaut : 16.000 (1000 comptes)

Ixx12 place l'amplitude de l'erreur de poursuite pour le moteur xx auquel un drapeau d'avertissement est activé. Si cette limite est dépassée, le bit d'état est mis a un pour le moteur ainsi que le SC du moteur (s'il y en a). Le bit d'état du SC est le OU logique des bits d'état de tous les moteurs dans un SC.

Le réglage de ce paramètre à zéro neutralise la fonction drapeaux d'avertissement de la limite d'erreur. Si ceci le paramètre est placé plus grand que l'Ixx11 *limite fatal de l'erreur de poursuite,* le drapeau d'avertissement ne sera jamais activé, parce que la limite fatale neutralisera le moteur d'abord. Si le bit1 d'Ixx97 est placé à 1, le moteur peut « être déclenché » pour les mouvements *homing* et le *jog-untiltrigger move.*

L'unité d'Ixx12 est 1/16 d'un compte. Par conséquent ce paramètre doit tenir une valeur 16 fois plus grand que le nombre de comptes auxquels la limite se produira. Par exemple, si la limite doit être 1000 comptes, Ixx12 devraient être placés à 16.000.

III-III-2-4 Ixx13 Motor xx Positive Software Position Limit :

Gamme :-2^{35} - $+2^{35}$

Unités : compte

Défaut : 0 (désactivé)

Ixx13 place la valeur maximum positive de la position autorisée pour le moteur xx. Il peut fonctionner de deux manières légèrement différentes.

Limite de position réelle : Les fonctions *housekeeping* « ménage » de Turbo PMAC comparent à plusieurs reprises position réelle du moteur xx à Ixx13.Si le moteur est en boucle fermé, et la position réelle est plus grande dans un sens absolu que Ixx13, Turbo PMAC exécute automatiquement la commande *abord*, qui ralenti le moteur jusqu'à l'arrêt au taux de décélération réglé par Ixx15. Si d'autres moteurs sont dans le mouvement coordonné, ils sont également apportés à un arrêt à leur propre taux Ixx15.

Noter qu'en ce mode, la décélération commence après que la limite ait été atteinte, ainsi le mouvement s'arrête en dehors de la limite.

Si le moteur est en mode boucle ouverte (d'une O-commande) quand il dépasse la limite Ixx13, il sera arrêté. Si la limite a été déjà dépassée, aucune commande en boucle ouverte n'est acceptée pour ce moteur, indépendamment de la polarité. Tandis que la limite Ixx13 est dépassée, Turbo PMAC ne permettra plus de commandes de mouvement en sens positif, de même pour un programme de mouvement. Cependant, il permettra des commandes de mouvement dans le sens négatif de n'importe lequel des types, cité auparavant.

2. **Limite de position désirée :** Si le bit15 d'Ixx24 est placé à 1, permettant la vérification de la limite de position désirée, Turbo PMAC va comparé la cible désirée du moteur comme calculée par le programme de mouvement a celle de la position limite, ou l'extrémité d'un segment intermédiaire – à la limite. Si la position cible n'est pas calculée dans le buffer spécial de *lookahead,,* quand cette position est plus grande dans un sens absolu que Ixx13, Turbo PMAC va automatiquement commander un arrêt, qui fait commencer ce moteur à ralentir jusqu'à l'arrêt de mouvement au taux réglé par Ixx15. Si d'autres moteurs sont dans le mouvement coordonné, ils sont également apportés à un arrêt à leur propre taux Ixx15.
Si cette position de cible est calculée dans le buffeur spécial de *lookahead*, quand cette position est plus grande dans un sens absolu que [Ixx13-Ixx41], Turbo PMAC modifie cette position à [Ixx13-Ixx41].Selon l'arrangement du bit 14 d'Ixx24, selon ce bit le PMAC rapporte le moteur à un arrêt commandé en cette position si (bit 14=0) ou continue le programme de mouvement du moteur si I4=1.

Noter qu'il est possible de placer ce paramètre en dehors de la gamme $+2^{35}$ (+64 milliards) si un couple de choses spéciales sont faites. D'abord, le facteur d'échelle Ixx08 pour le moteur doit être réduit pour employer cette position (la gamme du moteur est $+2^{42}$ / Ixx08).En second lieu, la valeur variable doit être calculé à l'intérieur de Turbo PMAC, parce que l'analyseur de commande ne peut pas accepter des constantes en dehors de la gamme $+2^{35}$ (par exemple pour placer I113 à 100 milliards, employer **I113=1000000000*100**).

III-III-2-5 Ixx14 Motor xx Negative Software Position Limit :

Gamme :-2^{35} - $+2^{35}$

Unités : compte

Défaut : 0 (neutralisé)

Ixx14 place la valeur maximale négative de la position autorisée pour le moteur xx. Il peut fonctionner de deux manières légèrement différentes.

1. Limite de position réelle : Les fonctions *housekeeping* « ménage » de Turbo PMAC comparent à plusieurs reprises position réelle du moteur xx à Ixx14.Si le moteur est en boucle fermer, et la position réelle est moins dans un sens absolu que Ixx14, Turbo PMAC émit une commande *abort,* qui fait commencer le moteur à ralentir à un arrêt complet avec un taux décélération réglé par Ixx15.Si les autres moteurs effectuent le même mouvement coordonné, ils sont également apportés à un arrêt à leur propre taux Ixx15. En dehors de la limite.

Si le moteur est en mode boucle ouverte (d'une O-commande) quand il dépasse la limite Ixx14, il sera arrêté. Si la limite a été déjà dépassée, aucune des commandes de boucle ouverte sont acceptée pour ce moteur, indépendamment de la polarité. Tant que la limite Ixx14 est dépassée, Turbo PMAC ne permettra plus de commandes dans la direction négative, même celle émanant de l'exécution d'un programme de mouvement, une commande d'essai, ou de la fonction position suivante. Cependant, la carte permettra des commandes dans la direction positive de n'importe lequel de ces types cités auparavant.

2. **Limite de position désirée :** Si le bit15 d'Ixx24 est placé à 1, permettant vérification de la limite de position désirée, Turbo PMAC va comparé la position cible désirée du moteur comme calculée par le programme de mouvement a celle de la position limite, ou l'extrémité de segment intermédiaire – à la limite. Si cette position cible n'est pas calculée dans un buffer spécial de *lookahead*, quand cette position est moins dans un sens absolu (pas grandeur) que Ixx14, Turbo PMAC émit automatiquement une commande d'arrêt , qui fait, que le moteur commence à ralentir avec un taux réglé par Ixx15. Si d'autres moteurs sont dans le même mouvement coordonné, ils sont également apportés à un arrêt à leur propre taux Ixx15.

Si cette position cible est calculée dans le buffer spécial de *lookahead*, quand cette position est moins dans un sens absolu que [Ixx14+Ixx41], Turbo PMAC modifie cette position selon [Ixx14+Ixx41]. Selon l'arrangement du bit 14 d'Ixx24 , le PMAC rapporte le moteur à un arrêt commandé en cette position si (bit 14=0) ou continue le programme de mouvement du moteur si I4=1.

III-IV-Conclusion :

Dans ce chapitre nous avons présenté les outils nécessaires à la programmation de l'interface. La notion d'axe et leurs types, chaque moteur lui est attachée un axe, maintenant si l'axe exécute le mouvement, le moteur fera ce mouvement. Nous avons trois types d'axes cartésiens, rotations et axes fantôme.

L'interface possède quatre types de variable de travail a savoir les variable d'initialisation I qui déterminent la personnalité de l'UMAC et ses accessoires, les variables P qui sont d'usage universel, les variables Q comme les variables P mais dépendent du système en coordination qui les utilisent et les variables M utilisées comme des pointeur vers des zones mémoires.

Nous avons présenté aussi la notion de système en coordination qui est le groupement d'un ou plusieurs moteurs pour synchroniser leur mouvement, un moteur qui n'est pas affecté à un système en coordination ne peut pas exécuter un mouvement. En plus, certaine configuration des I variables qui active le SC sont rapportées dans cette partie

CHAPITRE :

IV

IV- COMMANDE IMPLEMENTEE

IV-I- Introduction :

Dans le contrôleur UMAC est intégré un régulateur de la boucle de position qui est le standard PID, avec des termes feed forward en accélération et en vitesse ainsi que le *Notch filter*. La plupart des utilisateurs trouveront que ce régulateur est largement suffisant pour contrôler leurs systèmes. Le calibrage de ce régulateur ce fait par les variables I appropriées pour chaque moteur. Ainsi que la détermination des ses paramètres qui se fait par auto ajustement, à l'aide du pack programme *PMAC TuningPro,* de DELTA TAU.

IV-II- Paramétrage du PID

Le programme exécutif de PMAC pour les ordinateurs PC-COMPATIBLES fournit des moyens faciles pour la détermination des paramètres du régulateur PID. Il permet les commandes simples qui effectuent automatiquement des mouvements standard, recueille les données nécessaires à partir de la réponse, tracent ces données, et calculent les statistiques importantes pour la réponse. Ceci permet même aux utilisateurs inexpérimentés de faire quelques jugements selon des règles (fournies) simples pour optimiser l'ajustement. Les instructions et les exemples détaillés sont fournis du manuel logiciel [**20**]

L'Autotuning stimule le moteur, évalue la réponse, et calcule les gains exigés pour le niveau désiré de réponse. Cette fonction permet à l'ordinateur de faire toutes les décisions quant à ce que les gains appropriés devraient être.

IV-III-Fonctionnement du PID : [21]

Le gain proportionnel (« K_p » Ix30) fournie la rigidité (raideur) au système ; le gain en dérivation (« K_D »-Ix31) fournie l'amortissement pour la stabilité ; le gain d'intégration (« K_I »-Ix33) permet l'élimination des erreurs statiques. Ix34 détermine si l'action intégrale est activée tout le temps ou seulement durant les périodes ou la commande en vitesse est nulle.

La **Figure IV-1** montre le schéma de commande implémenté dans l'UMAC.

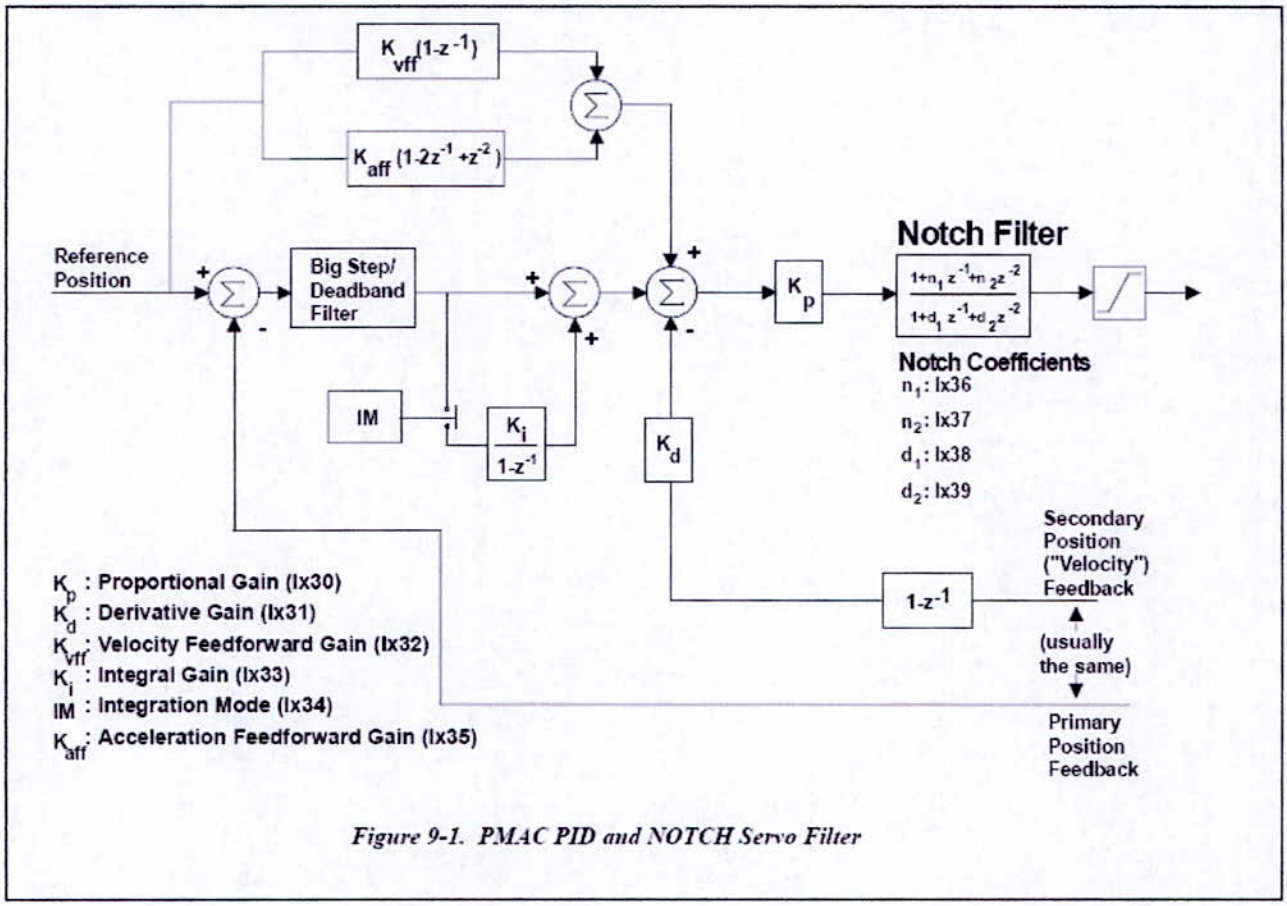


Figure IV-1 : *PID implémenté et le filtre Notch*

En plus, le gain d'anticipation en vitesse (kvff : Ix32) réduit les erreurs de poursuite introduite par l'amortissement (qui sont proportionnel a la vitesse), et le gain d'anticipation en accélération (kaff : Ix35) réduit ou élimine les erreurs de poursuite due au système d'inertie (qui sont proportionnel a l'accélération).

IV-IV- L'algorithme PID implémenté :

L'équation utilisée dans l'algorithme PID pour calculer la commande pour le moteur X est la suivante :

$$DACout(n) = 2^{-19} \cdot Ix30 \left[\left\{ Ix08 \cdot FE(n) + \frac{Ix32 \cdot CV(n) + Ix35 \cdot CA(n)}{128} + \frac{Ix33 \cdot IE(n)}{2^{23}} \right\} - \frac{Ix31 \cdot Ix09 \cdot AV(n)}{128} \right] \quad \textbf{(IV-1)}$$

Où

♦ DACout (n) est la commande en sortie sur 16-bit (-32768 to +32767) durant le servo cycle n. elle est convertie en tension [-10V +10V]. DACout (n) est limité parIx69.

♦ Ix08 est un facteur de mise à l'échelle interne de la position pour le moteur x (sa valeur par défaut est 96).

♦ Ix09 est un facteur de mise à l'échelle interne de la boucle de position du moteur x.

♦ FE(n) est l'erreur de poursuite en comptes durant le servo cycle n, qui est la différence entre la position de référence est la position actuelle durant le cycle [CP(n) – AP(n)].

♦ AV(n) est l'actuelle vitesse durant le servo cycle n,qui est la différence entre les deux dernières positions [AP(n) – AP(n-1)] en comptes par servo cycle.

♦ CV(n) est la vitesse de référence durant le servo cycle n :la différence entre les deux dernières positions de références [CP(n) – CP(n-1)] en compte par servo cycle.

♦ CA(n) est l'accélération de référence durant le servo cycle n, qui est la différence entre les deux dernières vitesse de référence [CV(n)- CV(n-1)] en compte par servo cycle.

IE(n) est l'intégrale de l'erreur de poursuite dans le servo cycle n, qui est :

$$\sum_{j=0}^{n-1} [FE(j)] \quad \textbf{(IV-2)}$$

Pour tous les cycles servo dont l'intégration est activée.

Ix34 =1 éteint l'entrée à l'intégrateur mais pas la sortie quant la vitesse de référence CV est différente de zéro)

IV.IV.1- Filtre de Notch : [21]

L'UMAC peut être utilisé pour mettre en oeuvre un filtre réjecteur (Notch filter). Un filtre rejecteur est un filtre anti-résonances utilisé pour neutraliser les résonances physiques, comme il y a plusieurs méthodes pour synthétiser un filtre rejecteur. Il est recommandé de synthétiser un filtre rejecteur légèrement amortie au alentours de 90% de la fréquence de résonance et un filtre passe bande fortement amortie au delà de la fréquence de résonance pour réduire le gain des hautes fréquences du filtre lui-même.

La forme la forme du filtre rejecteur de l'UMAC est :

$$\frac{N(z)}{D(z)} = \frac{1 + n_1 z^{-1} + n_2 z^{-2}}{1 + d_1 z^{-1} + d_2 z^{-2}} \qquad \textbf{(IV-3)}$$

Le numérateur N (z) est le filtre coupe bande, le dénominateur D (z) est le filtre passe bande. Ce filtre agit à la sortie du régulateur PID.

UMAC utilise quatre variables I pour spécifier complètement le filtre (le filtre coupe bande et le filtre passe bande), deux (Ix36 [N1] et Ix37 [N2]) pour le filtre coupe bande et deux (Ix38 [D1] et Ix39 [D2]) pour le filtre passe bande. Ces variables I représentent les coefficients utilisés dans l'équation différentiel du filtre, elles ont les valeurs dans l'intervalle [-2.0 +2.0] et elles sont sur 24-bit,un bit de signe,deux bit pour la partie entière et 21-bit pour la partie fractionnelle.

IV.IV.2- Auto ajustement du filtre Notch :

PMAC exécutive programme permet une configuration très simple du filtre, sans même avoir besoins de comprendre comment il fonctionne, la façon la plus simple est d'introduire la fréquence de résonance mécanique qu'on souhaite contrôler. *PMAC executive programme* va automatiquement calculer les caractéristique du filtre coupe bande et du filtre passe bande, calcule leurs coefficient, et les introduit a l'interface. Alternativement on peut spécifier individuellement les caractéristiques des filtres coupe bande et passe bande, et le programme calcule les coefficients de ces filtres d'une manière à vérifier les caractéristiques désirées et les introduire à l'interface.

IV.IV.3- Ajustement manuel du filtre Notch :

PMAC, nous permet de calculer manuellement les caractéristiques et les coefficients du filtre. La procédure est simple bien qu'elle implique plusieurs calcules comme expliquer ci-dessous

IV.IV.4- Caractéristiques du filtre Notch :

Un simple filtre coupe bande ou passe bande est caractérisé par deux paramètres. Mais plusieurs approches utilisent différentes paires de paramètres. Les paramètres généralement utilisés sont :

- La fréquence naturelle *(ωn)*

- La fréquence amortie ωd : donnée par l'actuelle amortissement égal à $\omega n * sqrt(1-b^2)$
- Tau d'amortissement *(b)* : allant de 0 (pas d'amortissement) à 1 (amortissement critique)
- Facteur Q : allant de 1 (amortissement critique) à l'infinie (pas d'amortissement), égal à 1/2b

IV-IV-4-1 Calcule des racines du filtre dans le plan des S :

Une fois qu'on a choisit les paramètres du coupe bande ou du passe bande on doit calculer les racines du filtre dans le plan des S. ceci est fait par ces deux équations :

$$\begin{cases} s_{imag} = w_d = w_n.sqrt(1-b^2) \\ s_{real} = -b.w_n = \dfrac{-b.w_d}{sqrt(1-b^2)} \end{cases} \quad \textbf{(IV-4)}$$

Les fréquences utilisées dans ces équations doivent êtres en radians par seconde

IV-IV-4-2 Calcule des racines du filtre dans le plan Z :

Ensuite on doit discrétiser le filtre, on doit trouver les pôles en z en utilisant $z = e^{sT}$ ou T est le pas d'échantillonnage servo. En divisant z en partie réelle et imaginaire.

$$\begin{cases} z_{imag} = \exp(s_{real}.T).\sin(s_{imag}.T) \\ z_{real} = \exp(s_{real}.T).\cos(s_{imag}.T) \end{cases} \quad \textbf{(IV-5)}$$

IV.IV.5- Les coefficients du filtre :

Apres ceci, on calcule les coefficients du filtre digital. Les coefficients sont formés par le produit des racines (pair du complexe conjuguais) :

Donc, les coefficients sont :

$$\begin{cases} -2.z_{real} \\ \exp(s_{real}.T)^2 = {z_{real}}^2 + {z_{imag}}^2 \end{cases} \quad \textbf{(IV-5)}$$

IV.IV.6- Valeurs de I-variable du filtre :

Maintenant, nous assignons simplement les valeurs que nous avons calculées aux I-variables approprié :

N1 (Ix36) ou D1 (Ix38) contient le coefficient z^{-1} ; Le N2 (Ix37) ou le D2 (Ix39) contient le coefficient de z^{-2}

IV-V- Module utiliser pour l'exécution de la commande : [16]

Une fois la commande est calculée par l'algorithme de commande (le PID en l'occurrence) un module externe à la DSPGATE et le processeur (Turbo PMAC). Intervient pour l'adaptation des commandes émises par le régulateur.

Dans ce qui suit nous allons, exposer le type du module utilisé pour la conversion de la commande, sa principale composition ainsi que son principe de fonctionnement.

IV.V.1- Amplificateur mode couple (courant) :

Comme présenté au chapitre II, nous disposons de deux cartes amplificatrices qui peuvent commander jusqu'à 8 moteurs. Ce type d'amplificateur, amplifie la commande issue du contrôleur UMAC qui est une tension représentant une commande en couple pour le moteur. Puisque les équations de base pour les moteurs montrent que le couple développé est proportionnel au courant qui traverse le moteur, ils sont souvent appelés amplificateurs de courant ou amplificateur a boucle de courant donc il faut s'assurer que la boucle de courant est bien fermée pour que le couple (courant) de sortie est proportionnel à la tension d'entrée. C'est pourquoi ils sont connus souvent sous le nom d'amplificateurs à transductance.

La deuxième loi de Newton montre que la force ou le couple est proportionnel à l'accélération linéaire ou angulaire respectivement. Donc la commande dans ces amplificateurs est effectivement une commande en accélération. Il n'y a pas de boucle de vitesse dans ces amplificateurs alors c'est au contrôleur lui-même de fermer la boucle de vitesse pour avoir une bonne stabilité du système. Avec le régulateur PID, ceci est fait par l'action dérivation, c'est pourquoi il est important d'avoir un gain de dérivation suffisant dans ces systèmes pour obtenir des réponses stables.

Ce type d'amplificateurs est populaire pour plusieurs raisons. Ils n'ont pas besoin d'une tachymètre ou d'une circuiterie électronique pour mesurer la vitesse donc ils sont simple et moins cher, parce que le gain de la boucle de courant dépend seulement des propriétés du moteur et non pas de la charge, ils peuvent être ajuster par le fabriquant pour un type de moteur utilisé. Donc l'utilisateur n'a pas à se soucier de l'ajustage de l'amplificateur quand la charge est connectée au moteur. En plus ce type d'amplificateur est bien adapté aux applications exigeant de grandes accélérations/décélérations. Ils ne dépendent pas beaucoup de l'erreur d'intégration qui introduit des temps de retard et des réponses lentes au changement de vitesse.

IV-VI- Boucle de courant :

D'après l'équation de base des moteurs, le couple est proportionnel au courant qui circule dans le moteur, c'est pour cette raison que ces amplificateurs sont destinés pour être commandés en courant (en couple) par le CPU, ils produisent un courant (respectivement couple) proportionnel à la tension d'entée de commande qui est produite par la DSPGATE par l'intermédiaire de la ligne DAC, tout en appliquant sur les bornes du moteur une tension avec modulation de largeur d'impulsions PWM, de ce fait, une boucle de courant avec un régulateur PI est implémenté. L'élément principal de cette boucle est le circuit ***LMD18200 3A, 55V H-bridge*** *de* ***Nationnal Semiconductor***.

IV.VI.1- Description du circuit LMD18200 [22] :

Le LMD18200 est un pont en format H, conçu pour les applications de commande de mouvement, le dispositif est établi en utilisant un processus multi technologique qui combine les circuits bipolaires et les circuits de commande CMOS avec des dispositifs de puissance DMOS sur la même structure monolithique, idéal pour piloter les moteurs à courant continue et les moteurs pas à pas, la **Figure IV-2** présente le schéma interne du LMD 18200.

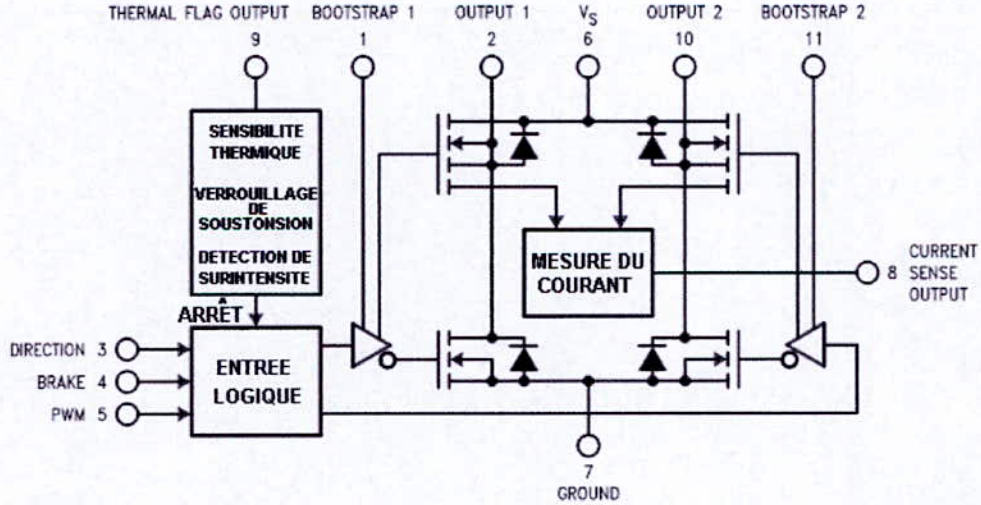


Figure IV-2 : *Le circuit LMD18200*

Comme montre la figure, le LMD18200 possède 11 broches, les fonctions de ces broches sont :

• **Pin 1 et 11, BOOTSTRAP :** Ce sont des entrées où des condensateurs devront être branches, ce qui est nécessaire pour le fonctionnement du circuit.

• **Pin 2 et 10, OUTPUT1 et OUTPUT2 :** ces sorties forment toutes les deux le signal de commande que doit recevoir le moteur en modulation de largeur des impulsions PWM.

• **Pin 3 et 5, DIRECTION et PWM :** Ces entrées forment toutes les deux le signal de commande à amplifier.

• **Pin 4, BRAKE :** Cette entrée digitale est utilisée pour l'arrêt instantané de l'amplification en cas de nécessité, réellement cette entrée est branchée directement avec la sortie AENA de la DSPGATE pour que l'ordre d'arrêt soit donné par le CPU.

• **Pin 6 et 7, Alimentation électrique :** ces entrées sont utilisées pour l'alimentation électrique de puissance, car si la tension d'alimentation de ce circuit serait V_s, alors le circuit délivre un signal en PWM entre V_s et $-V_s$.

• **Pin 8, CURRENT SENSE :** cette sortie est nécessaire pour la fermeture de la boucle du courant, car elle fournit la mesure de courant dans une tension proportionnelle à ce courant.

• **Pin 9, THERMAL FLAG :** cette sortie qui s'active à la température 145°C, est un indicateur de dépassement de la température acceptable pour l'environnement du circuit.

IV.VI.2- Fonctionnement de la boucle :

Etant présent le courant de référence sur la sortie DAC de la DSPGATE, elle va se comparer avec le courant qui circule dans le moteur **Figure IV-3**, l'erreur passe par un filtre PI (Proportionnelle Intégrale).

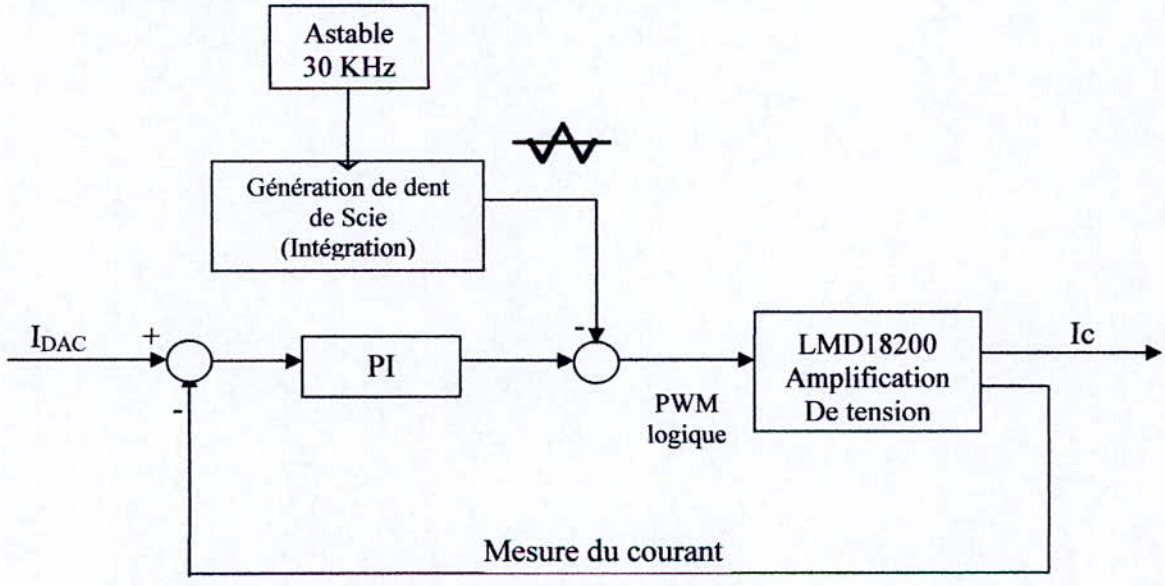


Figure IV-3 : *Schéma fonctionnel de la boucle de courant*

La sortie du filtre va se comparer avec un signal en dent de scie de 30 KHz qui est générée séparément de la boucle et dépendamment de la tension d'alimentation, pour donner un signal de commande des gâchettes du pont H.

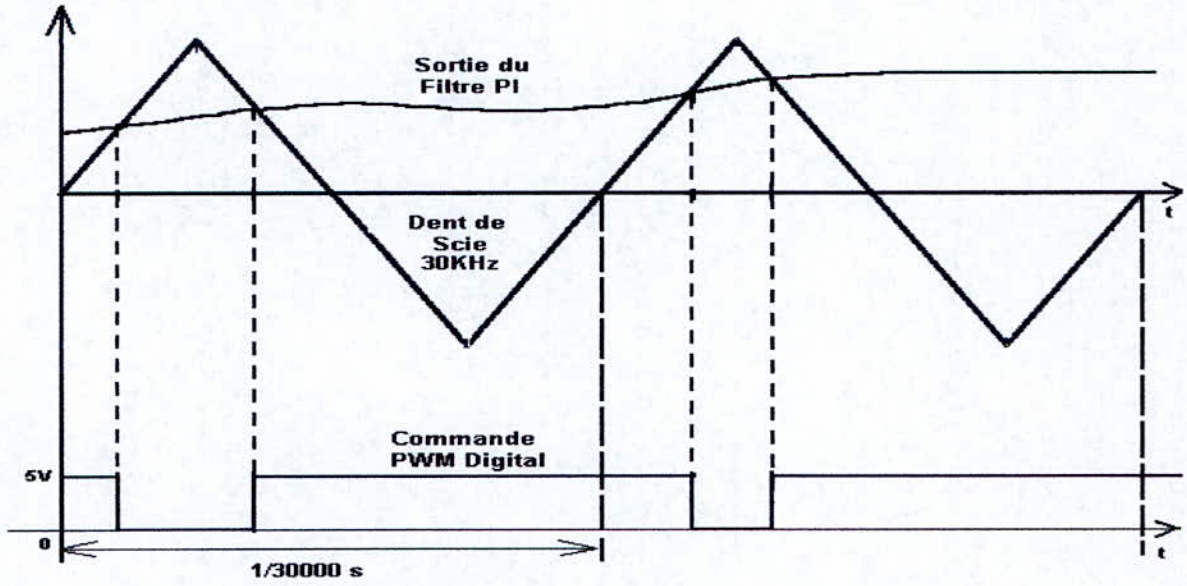


Figure IV-4 : *Présentation du principe de génération de la commande PWM*

Le signal résultant de comparaison vas être utilisé comme une entrée DIRECTION du circuit LMD18200 et l'entrée PWM sera mise à état haut, ce qui va donner sur la sortie OUTPUT1 le même signal amplifié et sur output2 le signal inversement amplifié, il en résulte la tension V_{o1}-V_{o2} sur les bornes du moteur à commander.

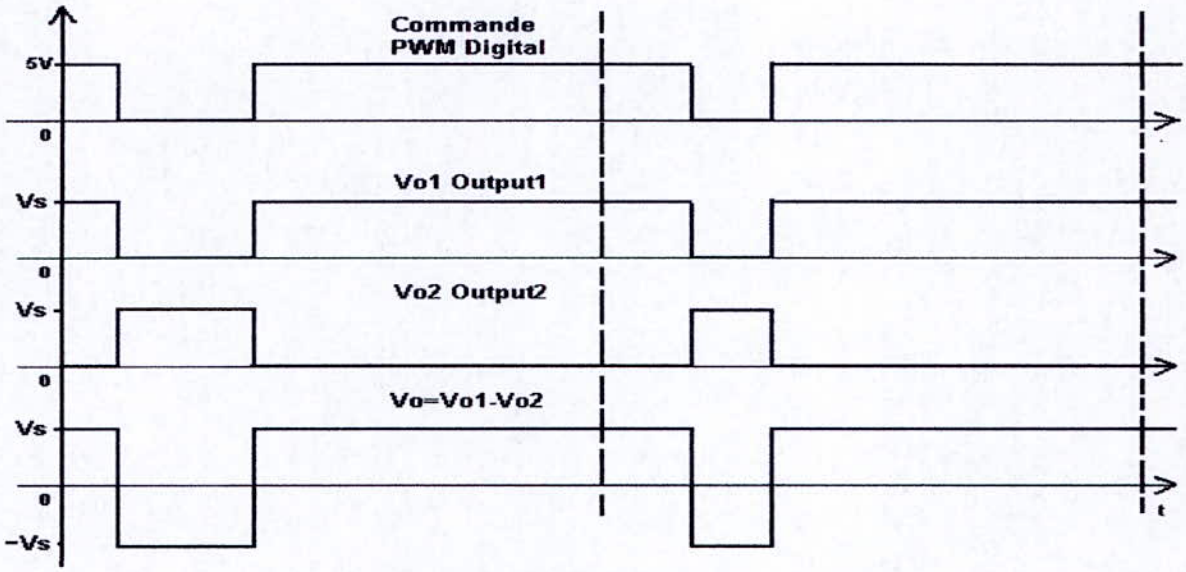


Figure IV-5 : *Amplification de la PWM digitale*

IV-VII- Conclusion :

Le volet traité dans ce chapitre est l'algorithme de commande implémenté au sein de l'UMAC, à savoir le PID, cet algorithme présente plusieurs blocks ajoutés, dont leur utilisation n'est édictée que par l'appréciation de l'utilisateur et des performances désirées, et a titre d'exemple le block Notch filter qui est décortiqué. Un tel filtre sera d'une utilité extrême dans le cas ou les fréquences des commandes requises pour exécuter des mouvements, frôlent les fréquences de résonances exemple fait dans le chapitre précédent pour le cas de mouvement linéaire avec TS= 0.

Les actions feed forward en vitesse et en accélération, peuvent être utilisées pour atteindre les performances désirées et assurer une bonne poursuite.

En plus de l'algorithme de commande, le module d'adaptation de cette dernière, émise par la DSPGATE, à savoir l'amplificateur, qui présente des caractéristiques, qui rendent la commande plus efficace.

CHAPITRE :

V

V- CALCUL GEOMETRIQUE

V-I- Introduction :

L'interface d'axes dont nous disposons (UMAC), fournie des structures facilitant la mise en œuvre et la manipulation des calculs géométriques complexes. Les calculs géométriques sont exigés quand il y a un rapport mathématique non linéaire entre les coordonnées du bout de l'outil et les positions articulaires des mécanismes typiques dans des géométries non cartésiennes. Ils sont généralement plus employés dans les applications de robotique, mais peuvent être employés avec d'autres types tels que les machines outils.

Les routines géométriques sont incluses dans le contrôleur par l'intégrateur, et fonctionnent invisiblement au programmeur et aux opérateurs. Ces routines peuvent être figé ou paramétrées pour les machines, elles peuvent s'adapter aux changements normaux tels que les longueurs d'outil et les différentes extrémités d'effecteurs.

Les calculs « *forward-Kinematic* « (géométrique directe) emploient les positions articulaires comme entrées, et les convertissent en coordonnées de pointe. Ces calculs sont exigés au début des mouvements qui sont programmés dans des coordonnées de pointe pour établir les coordonnées initiales pour le premier mouvement.

Les calculs « géométrique – inverse » emploient les coordonnées de la positions cartésiennes de l'outil comme entrée, et les convertissent en coordonnées articulaires. Ces calculs sont exigés pour le point final, pour chaque mouvement programmé dans les coordonnées cartésiennes.

V-II- Création des programmes géométrique : [20]

UMAC met en application l'exécution des calculs géométriques par des programmes spéciaux *forward-Kinematic* et *inverse-Kinematic* (géométrique directe et géométrique inverse). Chaque système en coordination peut avoir un de ces programmes, et les algorithmes eux même peuvent être exécutés automatiquement aux temps exigés, appelés comme des sous-programmes du programme de mouvement.

V.II.1- Elaboration des programmes géométrique directe (GD) :

&1 OPEN FORWARD
CLEAR
{Les instructions & les fonctions du modèle}
CLOSE

Les instructions sont les mêmes que celle utiliser dans les programmes PLCs (sauf les suivantes ADDRESS, CMDx, and SENDx).
Les fonctions du modèle décrivent les équations qui existent entre les coordonnées cartésiennes et articulaires.
Avant n'importe quelle exécution du programme géométrique direct, UMAC placera les positions actuelles pour chaque moteur XX, dans le système en coordination, dans la variable globale Pxx, le programme peut alors employer ces variables en tant qu'entrée dans les calculs.

Après n'importe quelle exécution du programme géométrique direct, UMAC prendra les valeurs dans Q1– Q9 pour le système en coordination, et copie ces derniers dans les 9 registres d'axes contenus dans ce système en coordination.
Le **Tableau V-1** montre l'affectation de chaque valeur Q à l'axe équivalent

Variable Q	Axe affecté	Variable Q	Axe affecté	Variable Q	Axe affecté
Q1	A	Q4	U	Q7	X
Q2	B	Q5	V	Q8	Y
Q3	C	Q6	W	Q9	Z

Tableau V-1 : *Correspondance des variables Q avec les axes*

Le but de base du programme géométrique directe, est de prendre les valeurs des coordonnées articulaires trouvées dans P1 – P32 pour les moteurs utilisés dans le système en coordination, et calcule les valeurs cartésiennes correspondantes, et les placent dans les variables Q1- Q9.

V.II.2- Elaboration des programmes géométriques inverses (GI) :

```
    &1 OPEN INVERSE
CLEAR
    {Les instructions & les fonctions du modèle inverse}
CLOSE
```

Avant l'exécution du programme géométrique inverse, UMAC placera les positions actuelles pour chaque axe dans le système en coordination dans les variables

Q1– Q9, le programme peut alors employer ces variables en tant que « entrée « dans les calculs.

Après l'exécution du programme géométrique inverse, UMAC lira les valeurs dans les variables Pxx (P1– P32) correspondent aux moteurs xx, dont le commande de définition d'axe est # xx->I.

V.II.3- Programme de géométrie inverse pour le mode PVT :

L'UMAC peut également faire la conversion des vitesses de l'espace cartésien à l'espace articulaire dans le programme de géométrie inverse, pour permettre l'utilisation du mode PVT avec des calculs cinématiques. Avec les mouvements du mode PVT, les calculs de position sont faits juste comme pour n'importe quel autre mode de mouvement, sauf qu'il y a un ensemble de calculs de conversion de vitesse qui doit être fait, où on peut étudier le profil des vitesses angulaires, (en donnant leurs équations mathématiques), pour un calcule de trajectoire optimale.

A l'exécution du mode PVT, UMAC placera automatiquement les valeurs de vitesse commandées des axes à partir des commandes PVT dans les variables Q11 – Q19, pour le système en coordination désiré, avant chaque exécution du programme géométrique inverse. L'unité des vitesses est définie par Isx90 (par exemple : mm/min ou deg/sec).La table suivante montre la variable utilisée pour chaque axe :

Le **Tableau V-2** montre la correspondance des variables Q avec la vitesse des axes

Vitesse d'axe (variables Q associés)	Axes Associés	Vitesse d'axe (variables Q associés)	Axes Associés	Vitesse d'axe (variables Q associés)	Axes associés
Q11	A	Q14	U	Q17	X
Q12	B	Q15	V	Q18	Y
Q13	C	Q16	W	Q19	Z

Tableau V-2 : *Correspondance des variables Q avec les vitesses d'axes*

UMAC placera également Q10 à 1 en ce mode comme indicateur au programme géométrique inverse qu'il devrait employer ces valeurs de vitesses d'axes pour calculer des valeurs articulaires de vitesse pour le moteur.

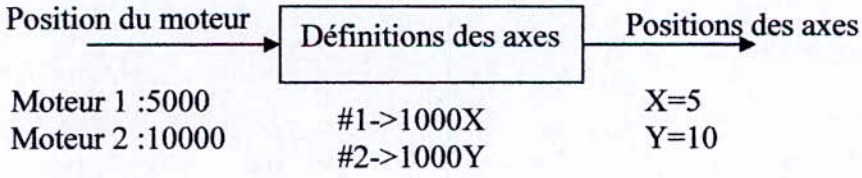
En ce mode, après n'importe quelle exécution du programme géométrique inverse, UMAC lira les valeurs dans les variables P1xx (P101 – P132) pour chaque moteur xx défini en tant qu'axes géométrique inverse (# xx->I). UMAC les emploiera en tant que valeurs de vitesse de moteur avec les valeurs de position dans Pxx pour créer un mouvement de PVT pour le moteur.

V-III- Exécution des programmes géométriques :

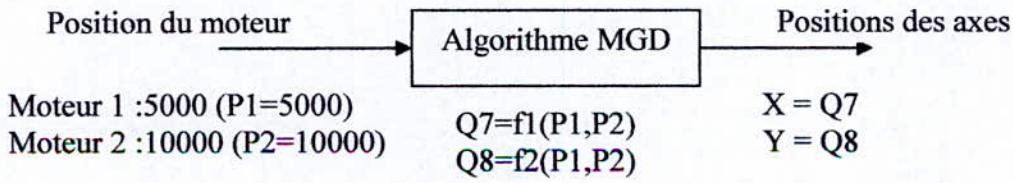
Une fois que les programmes géométriques directs et inverses ont été crées pour un système en coordination, UMAC les exécutera automatiquement aux temps appropriés une fois que les calculs géométriques ont été activés (en plaçant Isx50 à 1). Aucune modification à un programme de mouvement n'est exigée pour l'accès aux programmes géométriques en temps voulu.

Le programme géométrique direct est exécuté automatiquement chaque fois que l'instruction R ou S (exécution étape par étape) est donnée au système en coordination (si Isx50 =1). Ceci est fait pour s'assurer que la position de commencement est correcte pour le calcul du mouvement initial.

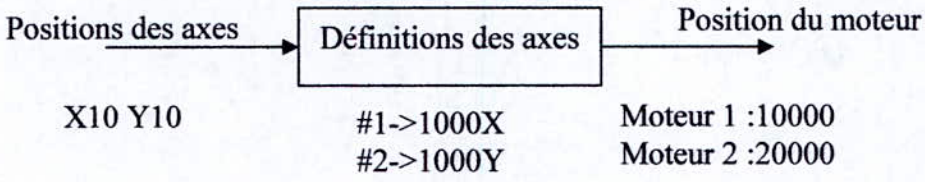
- Conversion du moteur à l'axe sans le programme de GD (Forward-Kinematic) :



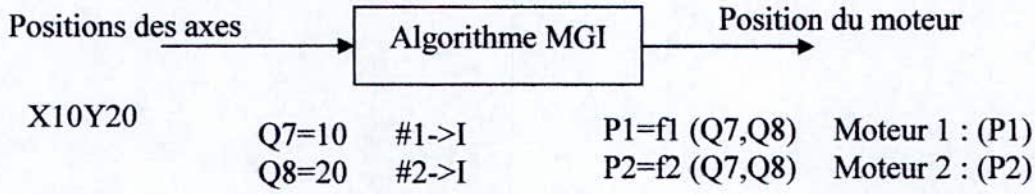
- Conversion du moteur à l'axe avec le programme de GD (Forward-Kinematic) :



- Conversion de l'axe au moteur sans le programme de GI (Inverse Kinematics) :



- Conversion de l'axe au moteur avec le programme de GI (Inverse Kinematics) :



V-IV- Conclusion :

Dans ce chapitre nous avons introduit la manipulation des calculs géométriques direct et inverse par l'interface.

Nous avons expliquer que lors des calculs géométrique directe, UMAC prend les valeurs des coordonnées articulaires trouvées dans P1 – P32 pour les moteurs utilisés dans le système en coordination, et les emplois comme entrées au programme géométrique direct pour calculer les valeurs cartésiennes correspondantes a (X Y Z), et les placent dans les variables Q1- Q9.

Par contre avant l'exécution du programme géométrique inverse, UMAC prend les positions actuelles pour chaque axe dans le système en coordination, qui sont dans les variables Q1– Q9, les emplois comme entrées lors de l'exécution du géométrique inverse pour calculer coordonnées articulaires P1 – P32.

CHAPITRE :

VI

VI- APPLICATION

VI-I- Introduction :

Dans ce présent chapitre nous allons mettre en évidence ce qui a été énoncer dans les chapitres précèdent, a savoir la : commande en boucle fermée, avec l'utilisation des différent structure du régulateur qui sont susceptible d'être implémenté, ainsi qu'une étude comparative entre les apports de chaque structure, que se soit dans le positionnement ou la poursuite de trajectoire, en effet, une panoplie de résultats seront exposés dans ce chapitre qui montre l'intérêt de chaque commande.

De même, le volet génération de trajectoire sera traité dans ce chapitre qui sera scindé en deux parties à savoir :

- Génération de mouvement dans l'espace articulaire : une série de programme de mouvement seront exécuter, comportant quelques types de mouvement qui sont implémenté dans l'UMAC (linéaire, PVT, Spline, circle), ainsi qu'un programme tache TOOK AND PLACE, permettant positionnement et déplacement serait élaborer.
- Génération de mouvement dans l'espace cartésien (opérationnelle) : pour cela, deux programmes du modèle géométrique inverse et directe du robot SCARA seront implémenté.

L'implémentation de ces derniers nous permettra de désigner la trajectoire de l'effecteur lui-même.

VI-II- Teste en boucle ouverte :

Un teste en boucle ouverte est parfois indispensable pour certains systèmes, néanmoins certaines précautions doivent être prisent afin de prémunir le système de tout excès de la commande appliquée.

Le programme *PmacTuningPro*, nous offre la possibilité de faire des testes en boucle ouverte, avec le choix de l'amplitude et la durée de la commande, ceci nous permet de limité les effet d'une telle commande (en boucle ouverte), d'une façons à excité tout le système tout en le protégeant d'une commande qui peut l'endommager.

L'objectif d'une commande en boucle ouverte, et de situer approximativement les exigences que doit satisfaire le régulateur PID utilisé (sans utiliser l'Auto-Tuning), à partir du résultat obtenu, en plus une telle démarche dans notre cas nous renseigne sur le bon câblage des moteurs (implicitement sur l'évolution de la position rapportée par les encodeurs qui sont à leurs tour configurable par la variable I7mn6 = 3 ou 7) — une évolution de la position dans le sens négatif pour une commande en position positive — Comme illustré dans les **FigureVI-1**

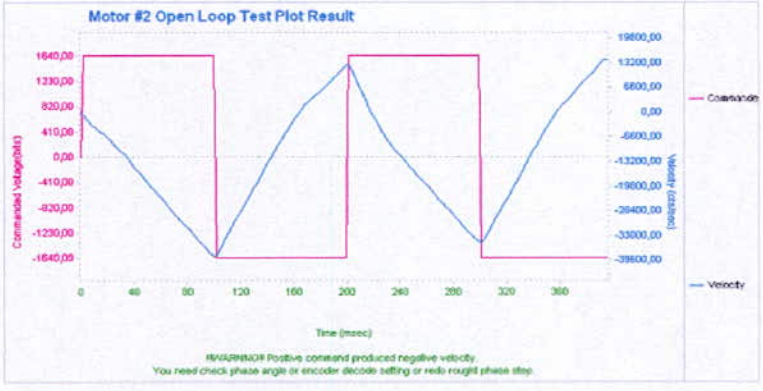


Figure VI-1 : *Réponse en boucle ouverte du moteur 2*

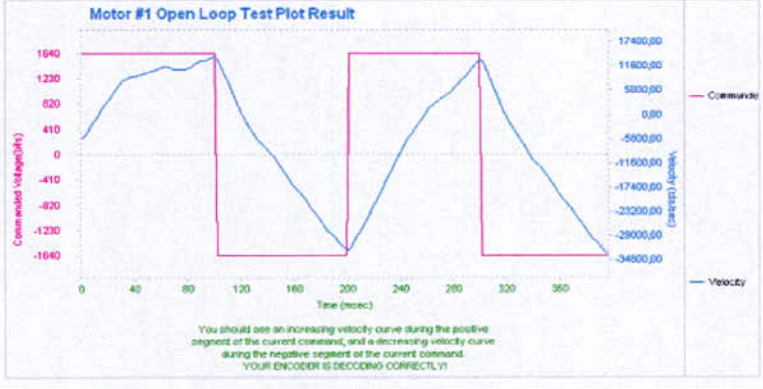


Figure VI-2 : *Réponse en boucle ouverte du moteur 1*

Nous constatons qu'une commande positive donne une réponse négative dans la **Figure VI-1** due à l'inversion des fils d'alimentation du moteur

La lenteur de la réponse du moteur en boucle ouverte est évidente **Figure VI-2**

VI-III-Test en boucle fermée :

Les testes en boucle fermée sont effectués a l'aide de la session *Auto Tuning PID* pour déterminer les paramètres préliminaires de la structure du régulateur choisit (PD, PID, PID+feed

forward), paramètres dont la détermination est faite automatiquement par le logiciel précité, cependant, la méthode suivit pour leur détermination n'est fournis par aucune des références bibliographique, misent à notre disposition par la firme DELTA TAU. Toute fois une constatation a été faite tout au long des testes effectuée et a partir des réponses obtenus, qui nous laisse supposer que la méthode suivit pour l'Auto ajustage des paramètres, est celle basé sur l'optimisation des caractéristique (énoncé au chapitre I partie commande en robotique).

En effet, les réponse obtenus, lors de chaque essai par Auto ajustage, pour un moteur seul ou moteurs/ articulation (moteur avec charge) présentent un temps de réponse t_r = [0.10 0.30] sec, temps de monté t_m = [0.030 0.045] sec, dépassement d= [0 20] %.

Néanmoins, afin d'assurer les performances souhaitées, une sorte de tâtonnement (une correction serai apporté au valeurs déterminer par *Auto Tuning*) sera suivit pour avoir les performances escompter.

Dans notre cas, nous avons optés pour des temps de réponse assez petit tout en tolérant un dépassement $d < 15\%$. Un tel déplacement est utilisé afin de contrecarrer les jeux de mouvement existant au niveau des articulations.

Les étapes suivantes désignent la procédure d'utilisation de l'auto tuning :

En premier lieu :

- On choisi l'option auto select bandwith
- On choisi un facteur d'amortissement entre 0.8 et 1

Ce premier test est effectué, non pas pour déterminer les paramètres du régulateur à implémentés mais pour une auto sélection de la bande passante du système

Une fois la bande passante est estimée :

- On décoche l'option auto select bandwith
- On sélectionne la structure du régulateur (PD, PID, PID+feedforward, avec ou sans filtre passe bas)
- choix de la durée et de l'amplitude des excitations, ces dernières sont choisit d'une façon à assurer l'excitation de toutes les dynamiques du système sans le déstabiliser

On lance l'Auto Tuning et on implémente les valeurs des paramètres suggérer du régulateur

Dans ce qui suit nous allons déterminer les paramètres des structures des régulateurs qui sont susceptible d'être implémentés, structures citées auparavant.

VI.III.1- Régulateur PD :

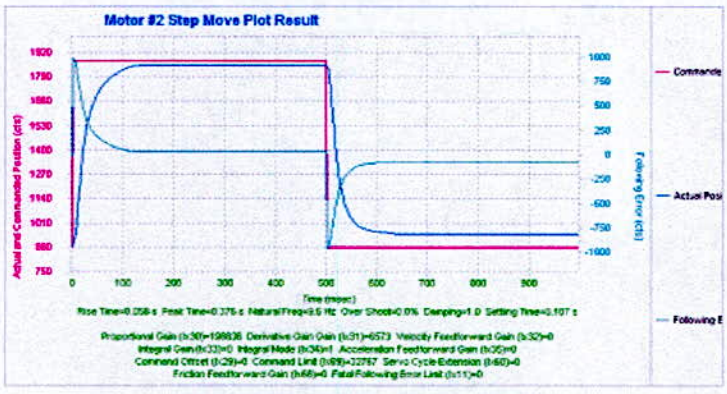


Figure VI-3 : *Réponse à un échelon*

VI.III.2- Régulateur PID :

- **Positionnement :**

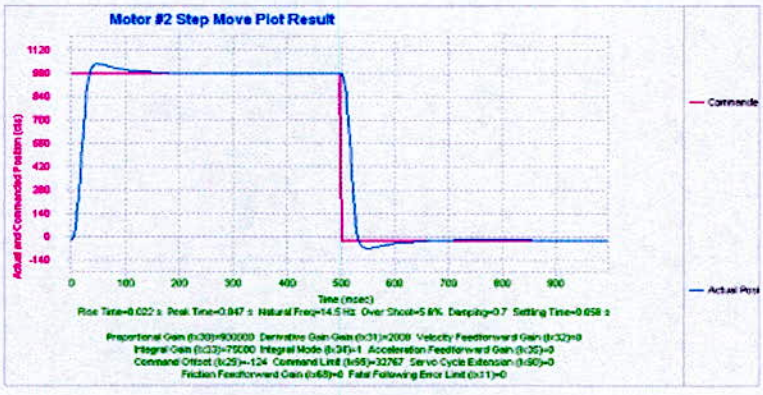


Figure VI-4 : *Réponse à un échelon*

- **Poursuite :**

Figure VI-5 : *Réponse à une rampe*

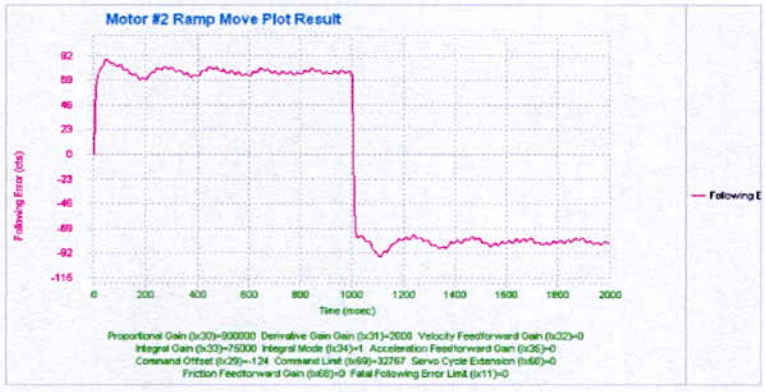


Figure VI-6 : *Erreur de poursuite pour une rampe*

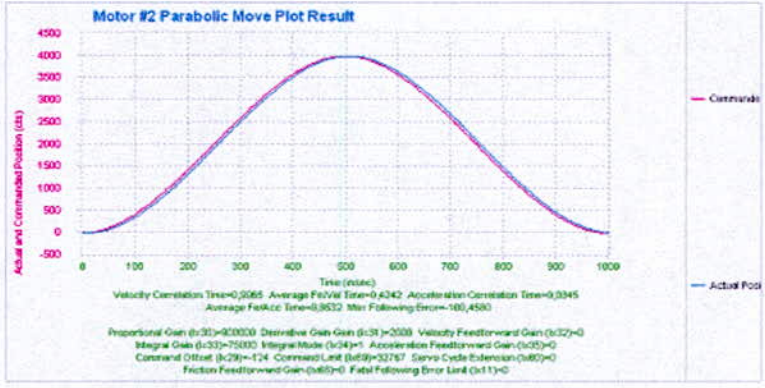


Figure VI-7 : *Réponse à une parabole*

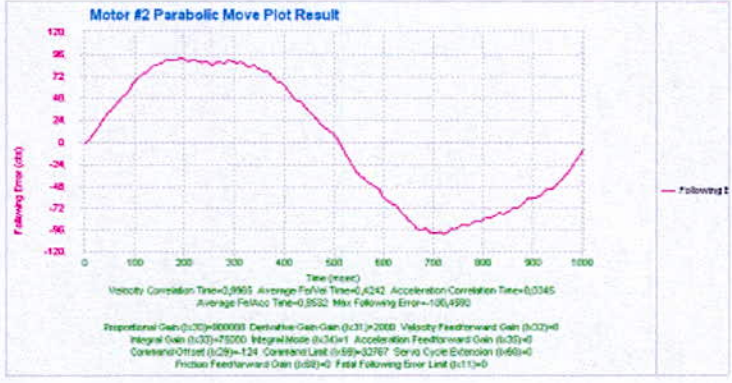


Figure VI-8 : *Erreur de poursuite pour une parabole*

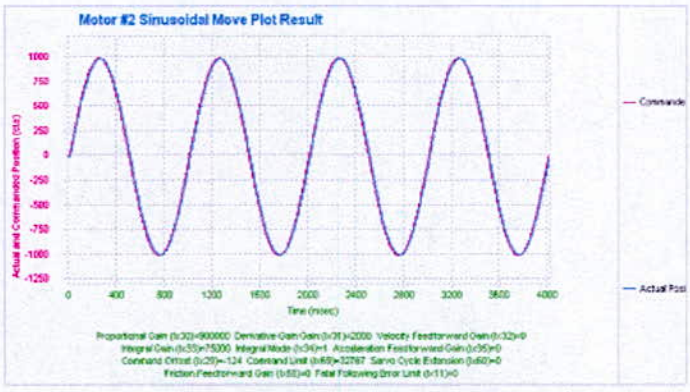


Figure VI-9 : *Réponse sinusoïdale*

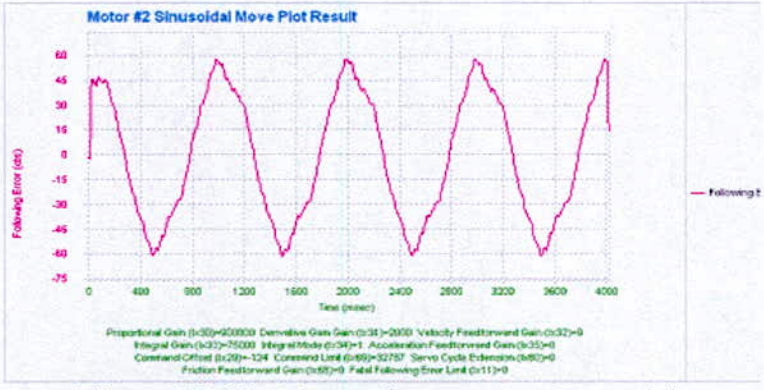


Figure VI-10 : *Erreur de poursuite sinusoïdale*

VI.III.3- Régulateur PID +Feedforward :

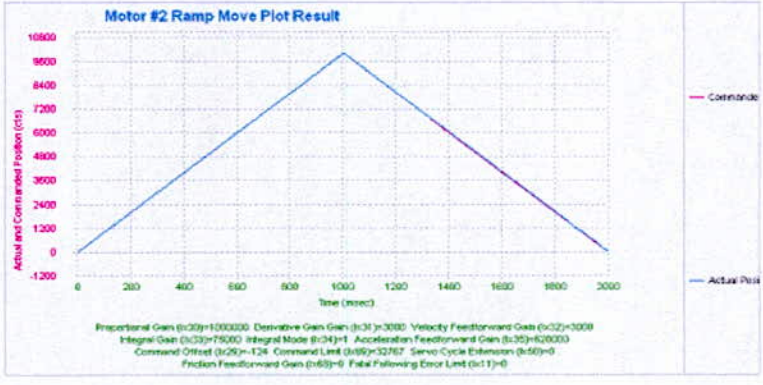


Figure VI-11 : *Réponse à une rampe*

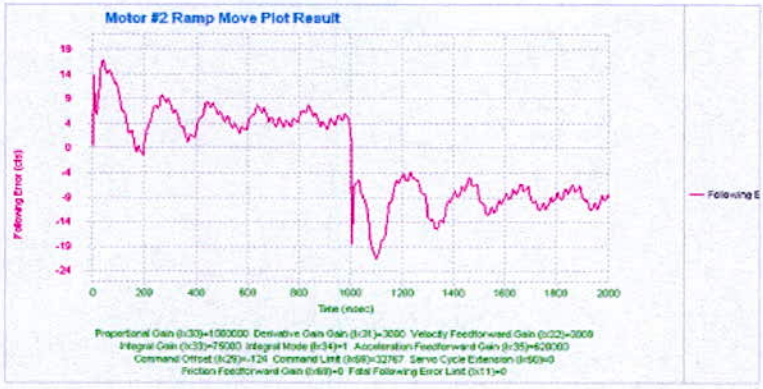


Figure VI-12 : *Erreur de poursuite pour une rampe*

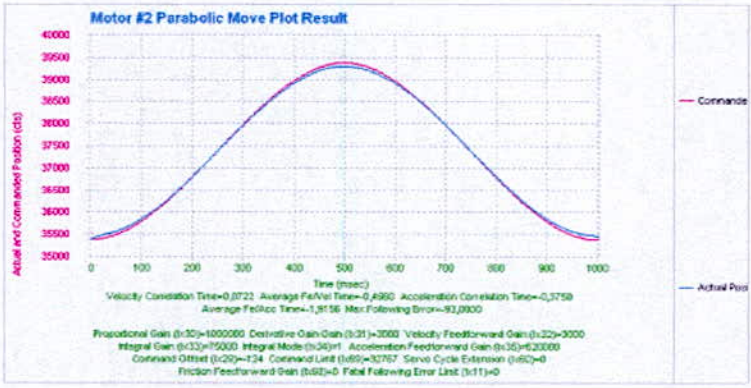


Figure VI-13 : *Réponse à une parabole*

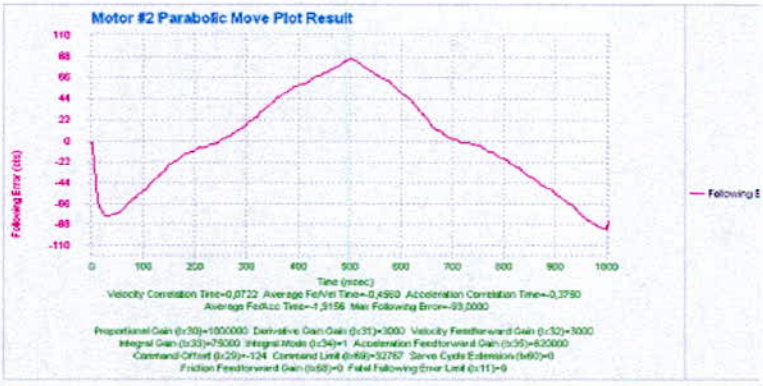


Figure VI-14 : *erreur de poursuite pour une parabole*

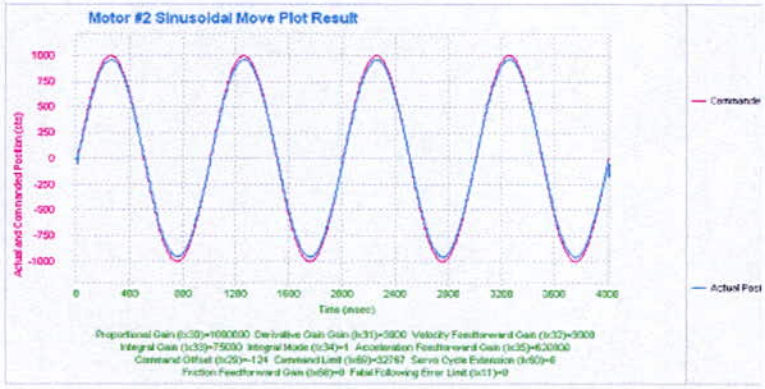


Figure VI-15 : *réponse sinusoïdale*

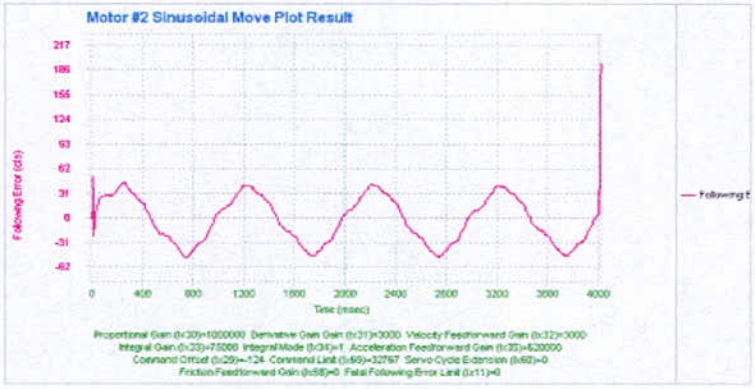


Figure VI-16 : *erreur de poursuite sinusoïdale*

VI.III.4- Interprétation des résultats :

On remarque que le régulateur PD n'assure pas le positionnement, existence d'une erreur statique **FigureVI-3**. Théoriquement ce régulateur peut garantir le positionnement en augmentant les ses gains car les effet de la gravitations qui est la seule perturbation en régime statique, sont compensés par la construction mécanique du bras SCARA. Mais en pratique on ne peut pas augmenter considérablement les gains à cause des saturations qui limitent la vitesse et l'accélération du moteur.

L'introduction de l'intégration élimine l'erreur statique et assure le positionnement **Figure VI-4.**

Toute fois il reste que le PID tout seul n'assure pas la poursuite ceci est remarquable dans la réponse a une rampe **Figure VI-5,** en en sinusoïde **FigureVI-9,** et en parabole **Figure VI-7.**

L'erreur de poursuite est due aux inerties du mécanisme et aux frottements qui sont fonction de la vitesse et de l'accélération.

L'ajout d'actions anticipatrices en vitesse et en accélération permet de réduire l'effet de ces inerties **Figure VI-11.**

A titre de comparaison l'erreur de poursuite dans cas d'un PID pour une rampe **FigureVI-5** est comprise entre +/- 92 incréments qui correspond à +/- 0.25 degré (92/363.889) par contre dans le cas PID+*Feedforward* **Figure VI-11** l'erreur est comprise entre +/- 16 incréments qui correspond à +/-0.044 degré

VI.III.5- L'effet feedforward sur les réponses en vitesse et en accélération :

- **PID**

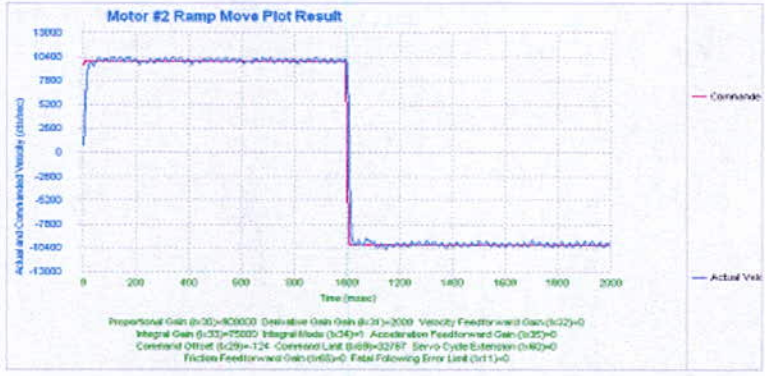


Figure VI-17 : *réponse en vitesse pour une rampe*

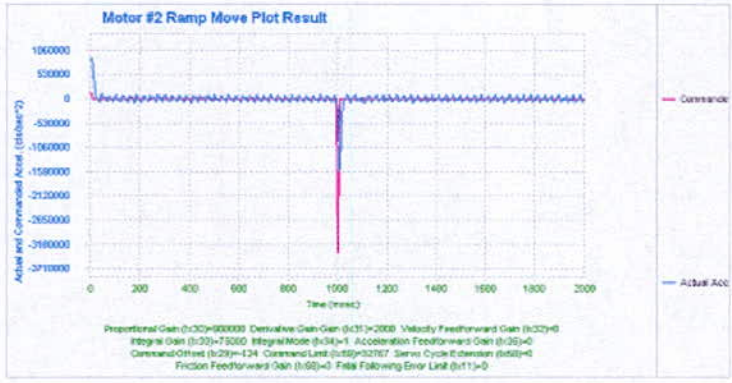


Figure VI-18 : *réponse en accélération pour une rampe*

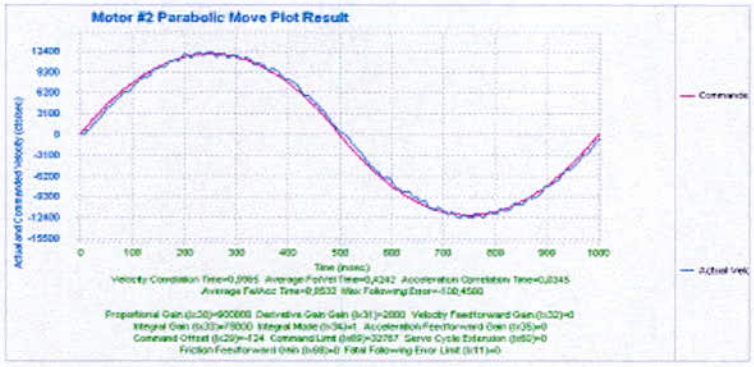


Figure VI-19 : *réponse en vitesse pour une parabole*

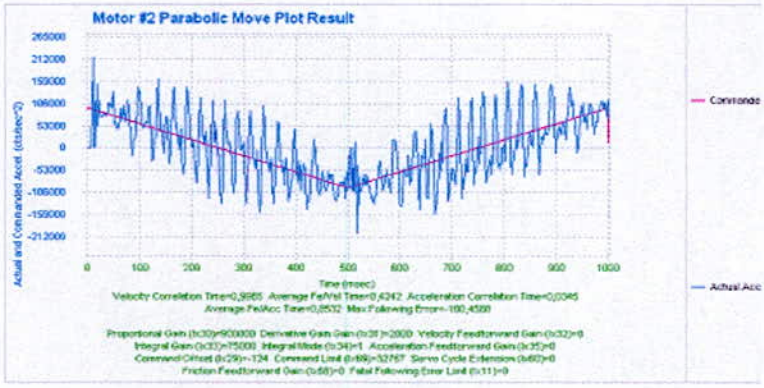


Figure VI-20 : *Réponse en accélération pour une parabole*

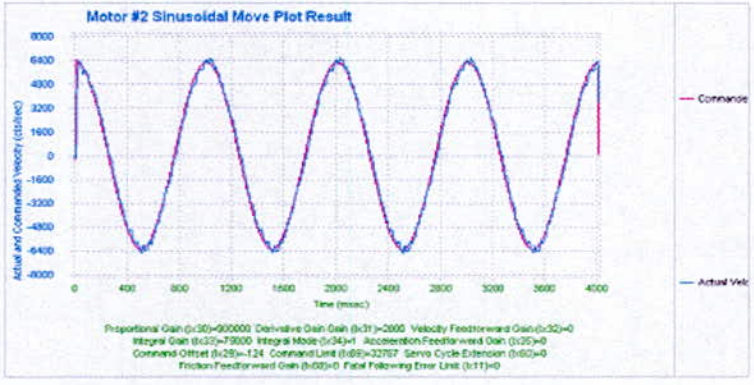


Figure VI-21 : *Réponse en vitesse pour un mouvement sinusoïdale*

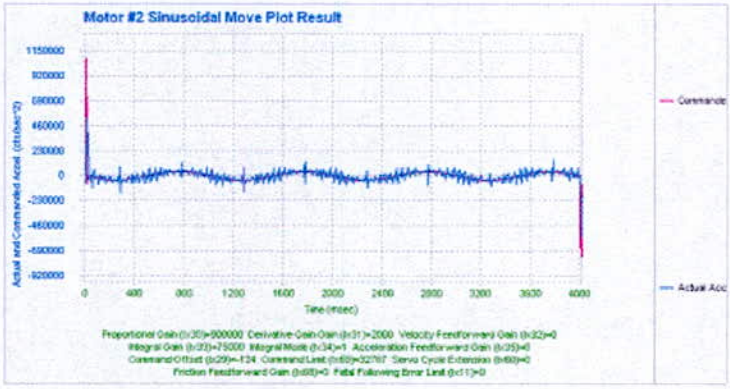


Figure VI-22 : *Réponse en accélération sinusoïdale*

- **PID+feedforward**

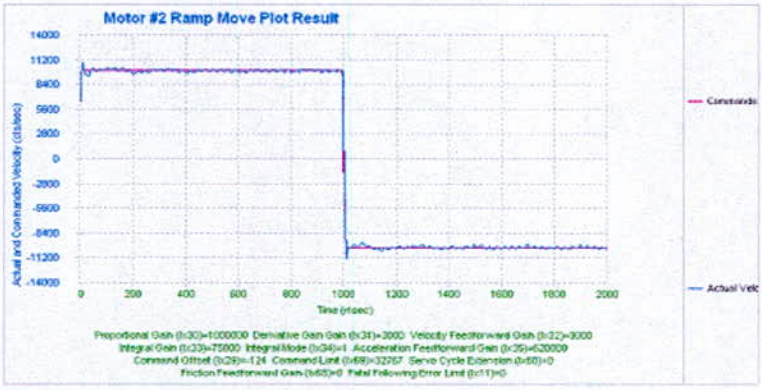


Figure VI-23 : *Réponse en vitesse pour une rampe*

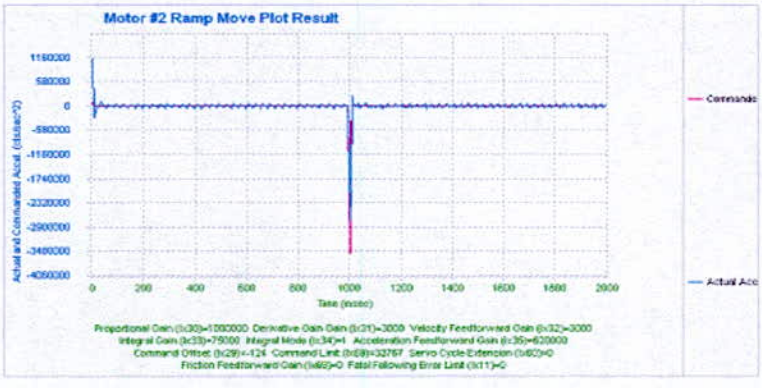


Figure VI-24 : *Réponse en accélération pour une rampe*

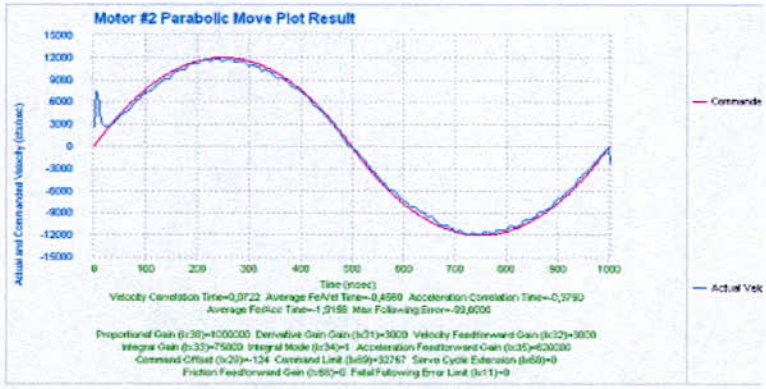


Figure VI-25 : *Réponse en vitesse pour une parabole*

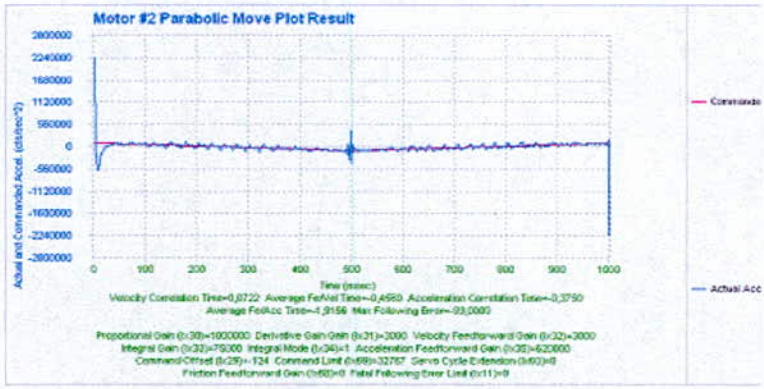


Figure VI-26 : *Réponse en accélération pour une parabole*

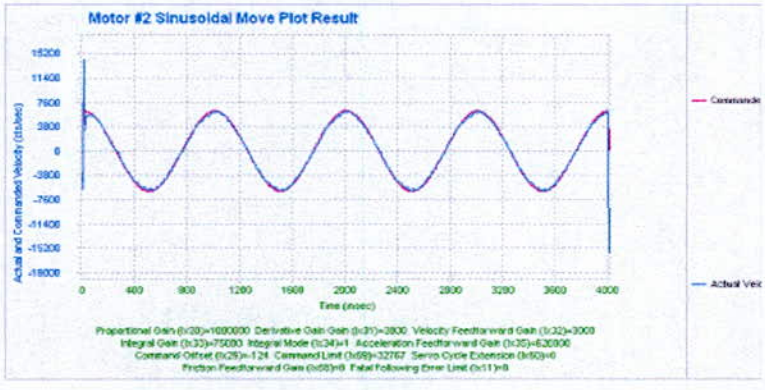


Figure VI-27 : *Réponse en vitesse sinusoïdale*

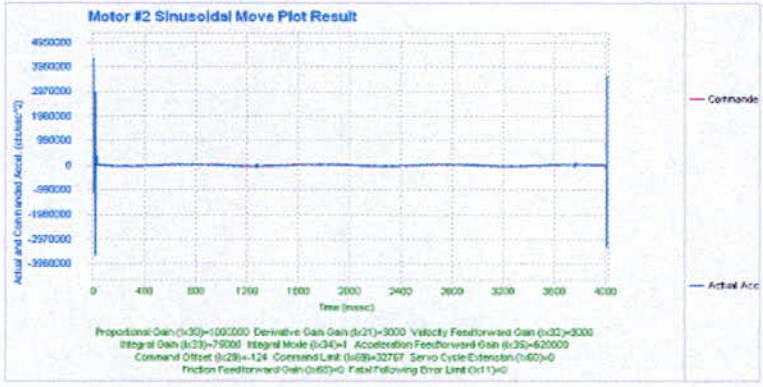


Figure VI-28 : *Réponse en accélération sinusoïdale*

L'effet du feedforward est remarquable aussi sur les réponses en vitesse et en accélération. Dans la réponse en vitesse à une rampe avec le PID les oscillations sont relativement importante **Figure VI-17,** on remarque leur diminution à l'introduction du *feed forward* **Figure VI-23.**

Ce genre d'oscillations peut se manifester dans l'un de ces trois cas :

- la constante électrique du moteur inférieur à la période de découpage du hacheur ce qui n'est pas vérifié

T_e=*1.06 ms*

T_d =*0.033ms*

- La constante mécanique du moteur inférieur a la période d'échantillonnage ce qui n'est pas vérifié aussi.

T_m=*8.5ms*

T_{ech}=*0.422ms*

-La commande présente des oscillations et c'est le cas **Figure VI-30**

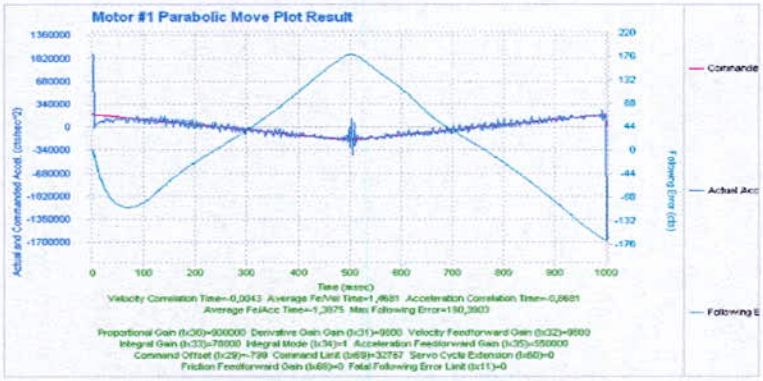


Figure VI-29 : *Réponse en accélération*

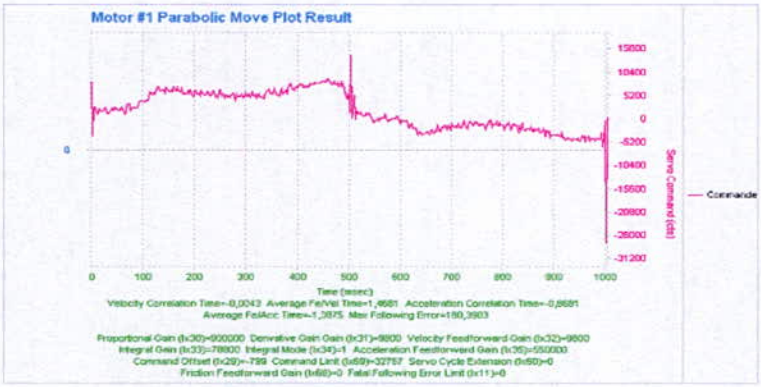


Figure VI-30 : *Servo commande*

VI-IV- Génération de mouvement dans l'espace articulaire :

Après détermination des paramètres du régulateur qui assure les performances désirées, ainsi que leurs implémentation (la structure choisie dans notre cas est celle du PID + Feedforward), on passera à l'exécution de quelques programmes de mouvement.

Le choix des types de mouvement, dans notre cas, n'a pour objectif que l'illustration, afin de mettre en évidence les types de mouvements énoncés dans le chapitre III.

Toutefois, la géométrie de la trajectoire de l'organe terminal dans l'espace opérationnel est imprévisible bien quelle soit répétitive.

Donc un choix d'un mouvement linéaire n'a pas pour conséquence un déplacement linéaire de l'effecteur, mais un profil de position polynomiale, profil de vitesse trapézoïdale.

Dans ce qui suit nous allons présenter le programme de mouvement ainsi que les résultats obtenus pour chaque mouvement :

VI.IV.1- Mouvement linéaire :

Déplacement du moteur1 de 45° et du moteur 2 de 45°

```
close
delete gather
undefine all

&1

#1->363.889A
#2->363.889B
open prog 6 clear

linear
INC             ; mode incrémentale
TA 50           ; temps d'accélération = 50 msec.
TS 30           ; temps de courbure au bout du profil de vitesse = 30 msec.
TM 1000         ; temps de mouvement = 1 sec.
A45 B45

close
```

Résultat obtenu :

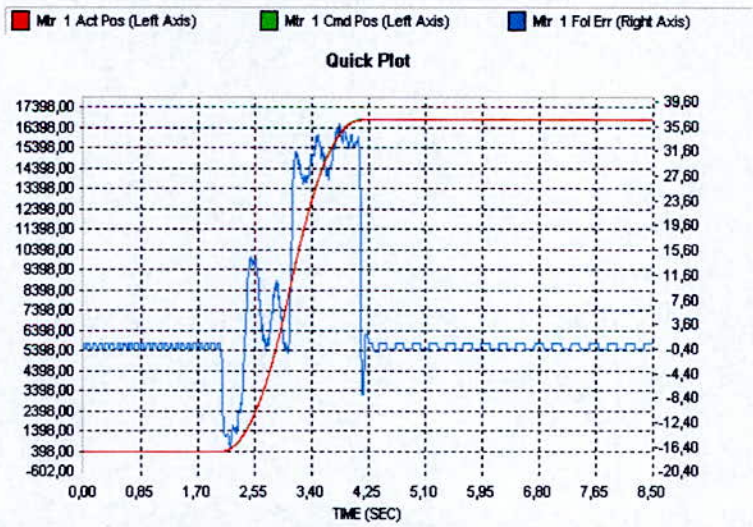


Figure VI-31 : *Profil de la position commandée, désirée et de l'erreur*

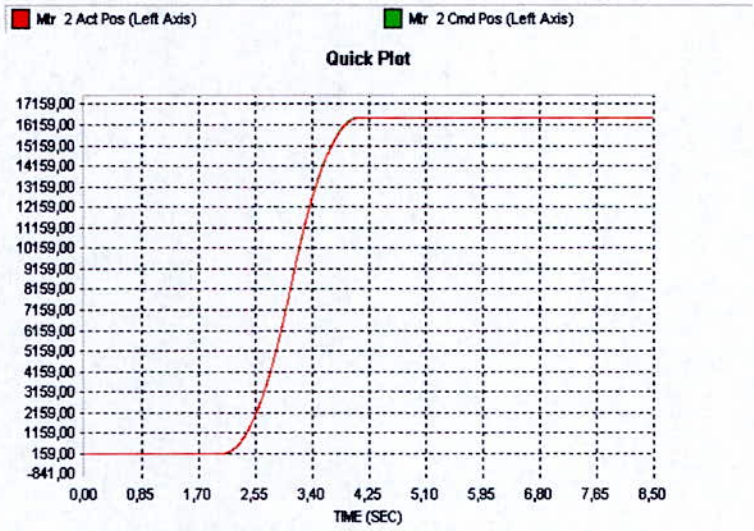


Figure VI-32 : *Profil de la position commandée*

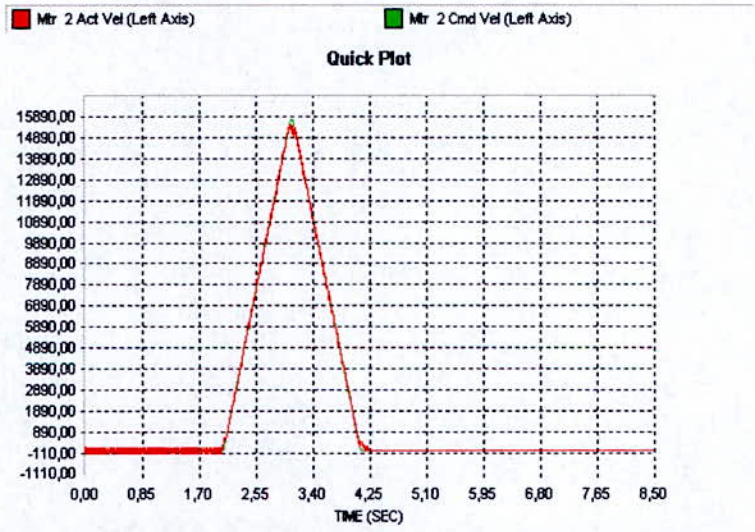


Figure VI-33 : *profil de la vitesse commandée et désirée*

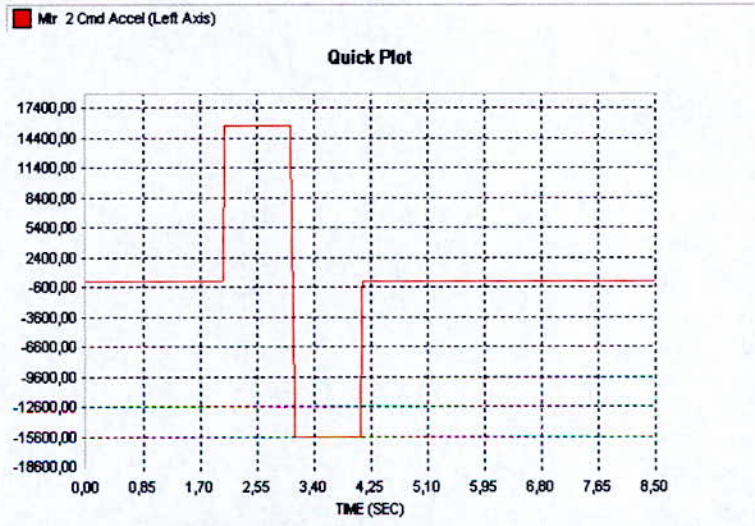


Figure VI-34 : *Profil de la vitesse commandée et désirée*

VI.IV.2- Mouvement spline1 :

```
close
delete gather
undefine all
&1
#1->363.889A
#2->363.889B
open prog 6 clear
spline1
INC                ; mode incrémentale
TA 50              ; temps d'accélération = 100 msec.
TM 1000            ; temps de mouvement = 1 sec.
A45 B45

Close
```

Apres exécution on obtient :

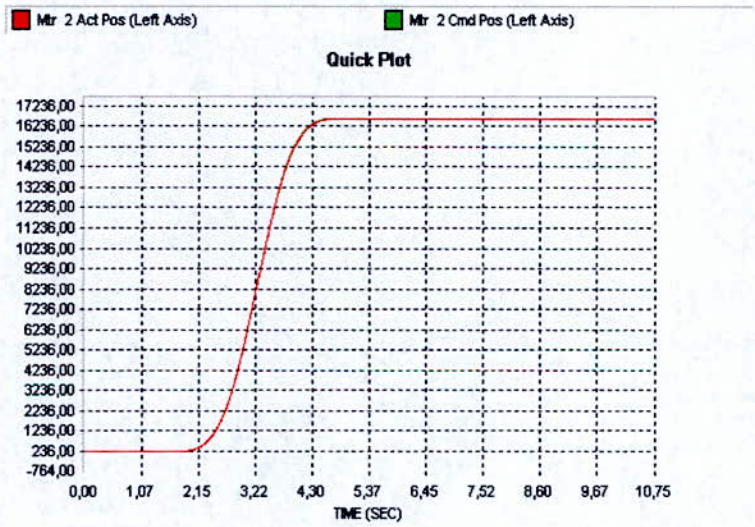


Figure VI-35 : *Poursuite en position (Splin1)*

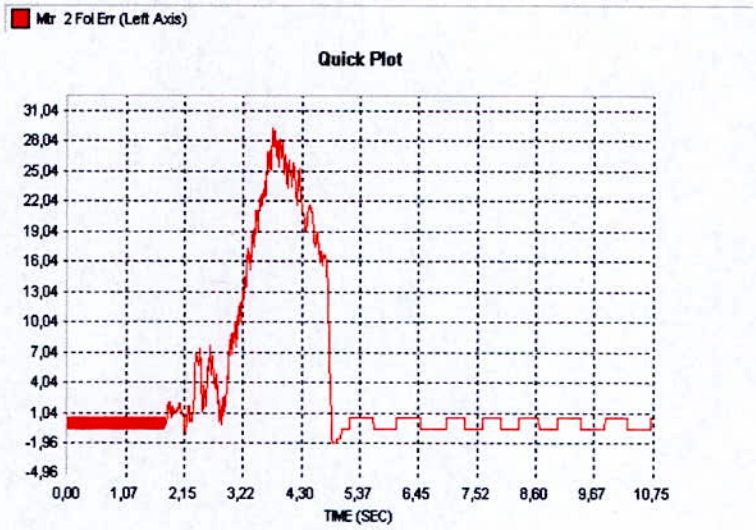


Figure VI-36 : *Erreur de poursuite (Splin1)*

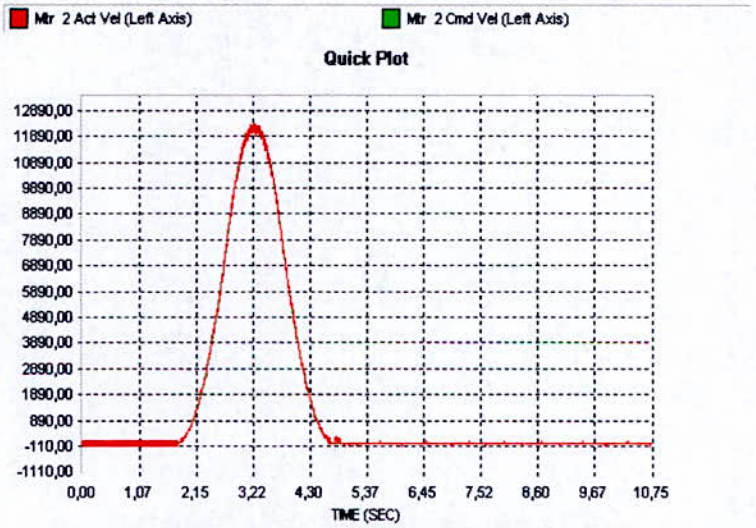


Figure VI-37 : *Profile de vitesse (Splin1)*

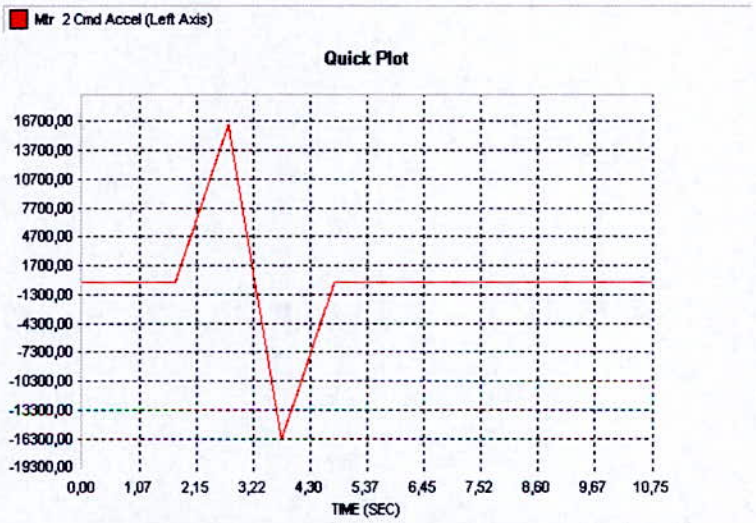


Figure VI-38 : *Profil de l'accélération (Spline1)*

VI.IV.3- Mouvement PVT :

```
close
delete gather
undefine all
&1
#2->363.889x
open prog 17 clear
INC
PVT 300
X40 :50
X40:0
CLOSE
```

Apres exécution on obtient :

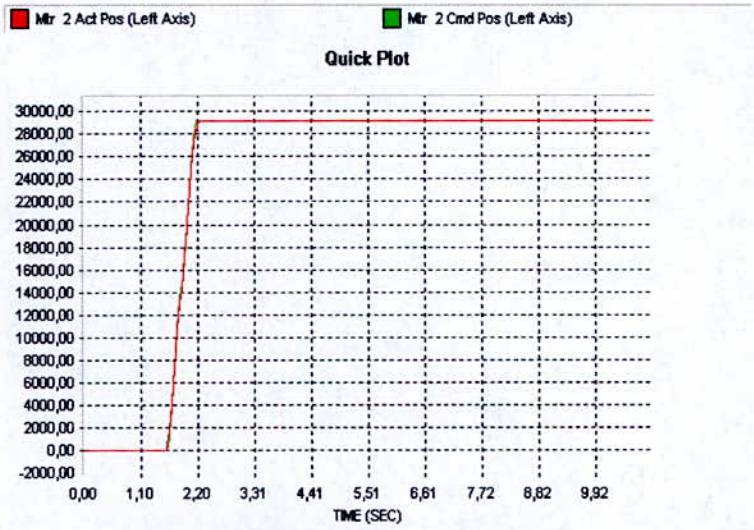


Figure VI-39 : *Poursuite en position (PVT)*

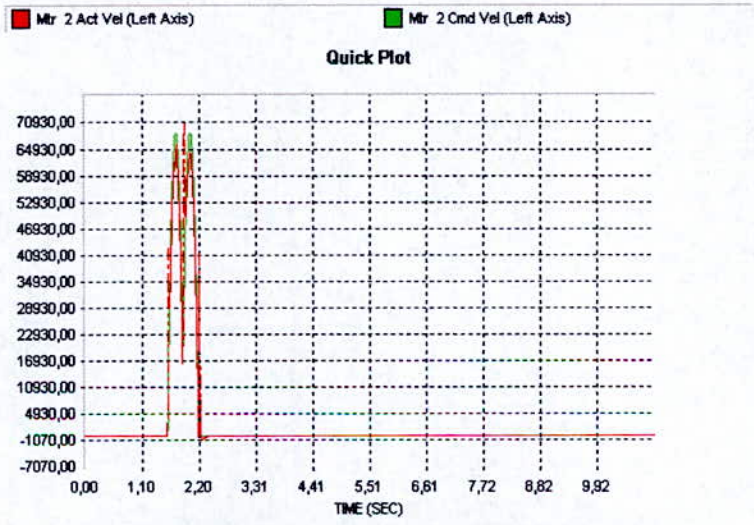


Figure VI-40 : *Poursuite en vitesse (PVT)*

Comme on peut le constater dans les différentes figures rapportées dans cette partie, le choix de type de mouvement pour la génération de trajectoire dans l'espace articulaire est édicté par les limitations en vitesse et en accélération, toutes fois la plus grande précaution doit être prise lors de l'exécution de ces mouvements et le choix du type de mouvement. En effet, un mouvement linéaire sans S-curve (TS = 0) entraîne des changements brusques d'accélération, ce qui peut excite les fréquences raisonnantes du système. Ce qui nous pouvons évité en prenant TS≠0 ou un autre type de mouvement exemple Spline1 **Figure VI-37** et **Figure VI-38**.

VI-V- Génération de mouvement dans l'espace opérationnel :

Pour pouvoir planifier des trajectoires dans l'espace opérationnel on doit implémenter le programme géométrique directe et inverse étudiés au chapitre I.

L'UMAC réserve deux buffers spécial *forward* et *inverse* pour chaque système en coordination pour l'introduction du programme géométrique direct et inverse. On écrit ces programmes dans l'éditeur de pewin32pro puis on les introduit à l'interface pour un système en coordination donné.

Le programme géométrique direct est exécuté à chaque fois une commande R ou S (début de mouvement) est donnée pour le système en coordination si Isx50 =1. Ceci est fait pour s'assurer que la position de départ de l'outil pour le calcul des premiers mouvements.

Le programme géométrique inverse est exécuté automatiquement à chaque fois UMAC calcule une nouvelle position durant l'exécution du programme de mouvement.

Tout programme de mouvement par ce système en coordination subira la transformation géométrique si les calculs géométriques sont activés par Isx50=1.

VI.V.1- Le programme géométrique direct du robot SCARA :

```
I15 = 0                 ; les calcules Trigonométriques en degré
&1                      ; Adressé SC 1
Q91= 200                ; longueur L1 (mm)
Q92=200                 ; longueur L2 (mm)
Q93= 363.889            ; Compte par degré (pour A et B)
&1 OPEN FORWARD
CLEAR                                          ; effacer le contenue précèdent
Q7=Q91*COS(P1/Q93)+Q92*COS((P1+P2)/Q93)        ; position X
Q8=Q91*SIN(P1/Q93)+Q92*SIN((P1+P2)/Q93)        ; position Y
CLOSE
```

VI.V.2- Le programme géométrique inverse du robot SCARA :

```
&1
#1->I          ; Moteur 1 affecté au modèle géométrique cinématique inverse (MGI) dans
SC 1
#2->I          ; Moteur 2 affecté au MGI dans SC 1
M5182->Y:$00203F,22,1          ; SC 1 bit indicateur du temps de calcule
; les constantes du système
Q94=Q91*Q91+Q92*Q92 ; L1^2 + L2^2
Q95=2*Q91*Q92 ; 2*L1*L2
Q96=Q91*Q91-Q92*Q92 ; L1^2 – L2^2
&1 OPEN INVERSE
CLEAR
Q20=Q7*Q7+Q8*Q8                ; X^2+Y^2
Q21=(Q20-Q94)/Q95              ; cos (B)
IF (ABS (Q21) <0.9998)         ; si la variation de B est dans le voisinage [–1°,1°]
; ou [-179°,179°] (le bras est tendu au maximum ?
Q22=ACOS(Q21)                  ; B (deg)
Q0=Q7                          ; X dans cos argument for ATAN2
Q23=ATAN2(Q8)                  ; A+C = ATAN2(Y,X)
Q24=ACOS((Q20+Q96)/(2*Q91*SQRT(Q20)))      ; C (deg)
Q25=Q23-Q24                    ; A (deg)
P1=Q25*Q93                     ; Moteur 1 = 363.889*A
P2=Q22*Q93                     ; Moteur 2 = 363.889*B
ELSE
M5182=1                        ; déclencher erreur de temps de calcule
ENDIF
CLOSE
```

Cet exemple arrête le programme dans le cas où le bras du robot serait tendu au maximum (cos (B) ≈ 0 donc X=L1 et Y=L2). Il fait ceci en activant le bit d'état « *run-time error* » qui correspond à une erreur d'exécution, pour le système en coordination a 1, qui cause l'arrêt automatique de

l'exécution du programme de mouvement (en activant la commande Abort : commence a décéléré immédiatement).

VI.V.3- Le programme cinématique du robot SCARA :

Les expressions des vitesses angulaires sont :

$$\begin{cases} \dot{A} = \dfrac{L_2 \cos(A+B)\dot{X} + L_2 \sin(A+B)\dot{Y}}{L_1 L_2 \sin B} \\ \dot{B} = \dfrac{[-L_1 \cos A - L_2 \cos(A+B)]\dot{X} + [-L_1 \sin A - L_2 \sin(A+B)]\dot{Y}}{L_1 L_2 \sin B} = \dfrac{-X\dot{X} - Y\dot{Y}}{L_1 L_2 \sin B} \end{cases} \quad \text{(IV.3)}$$

Notons que les vitesses deviennent infinies quand l'angle B approche 0° ou 180° (les points singuliers du robot), dans l'exemple suivant, une erreur d'exécution est créée si on est trop près d'une singularité.

```
&1
OPEN INVERSE CLEAR
;le calcule de position est fait dans le programme précèdent MGD
IF (Q10=1)                    ; mode PVT
Q26=SIN(Q25)                  ; sin (B)
IF (Q26>0.0175)               ; n'est pas proche de la singularité ?
  Q27=Q91*Q92*Q26             ; L1*L2*sinB
  Q28=COS(Q25+Q22)            ; cos (A+B)
  Q29=SIN(Q25+Q22)                              ; sin (A+B)
  Q30=(Q92*Q28*Q17+Q92*Q29*Q18)/Q27  ;  dA / dt
  Q31=(-Q7*Q17-Q8*Q18)/Q27                      ; dB / dt
  P101=Q30*Q93                ; vitesse du moteur #1
  P102=Q31*Q93                ; vitesse du moteur #2
ELSE                          ; proche de la singularité
M5182=1                       ; déclencher erreur de temps de calcule
ENDIF
ENDIF
CLOSE
```

VI.V.4- Planification d'une trajectoire spirale :

- **Le programme :**

```
I13=I0                          ; temps de segmentation
OPEN PROG1 CLEAR
ABS                             ; Mode absolue
INC(R)                          ; Rayon du cercle en mode incrémental
HOME 1,2                        ; revenir à la position zero

NORMAL K-1                      ; le mouvement dans le plan X-Y
CIRCLE 1                        ; sens du cercle

P1=0
WHILE (P1<100)                  ; While radius in less than 100
X(-P1+190)Y190I(P1)             ; exécution des cercle avec un rayon progressif
P1=P1+1                         ; on incrémente le rayon
ENDWHILE
CLOSE
```

Apres le chargement de ce programme à l'interface et sont exécution en tapant la commande &1 B1 R dans le terminal, et en activant les calculs géométriques en mettant I5150 = 1, on remarque que le robot fait une spiral.

Le résultat obtenu illustré dans la **FigureVI-41**, est obtenu en récupérant le contenu des adresses $80007 et $ 80008 qui correspondent à l'évolution de la position selon l'axe X et Y respectivement.

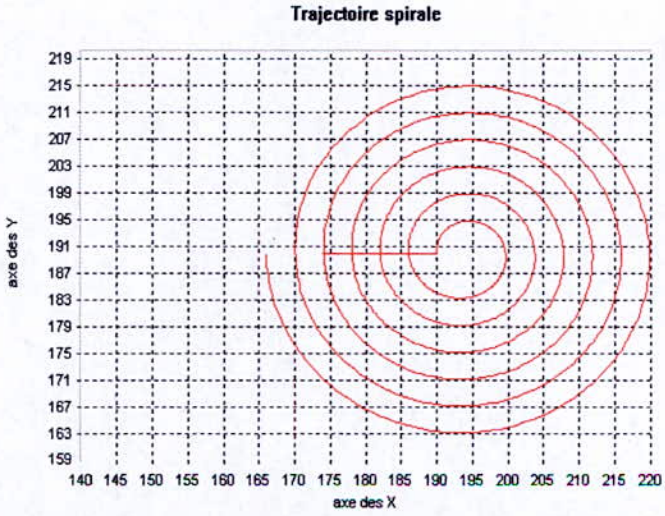


Figure VI-41 : *Trajectoire de l'effecteur dans l'espace cartésien.*

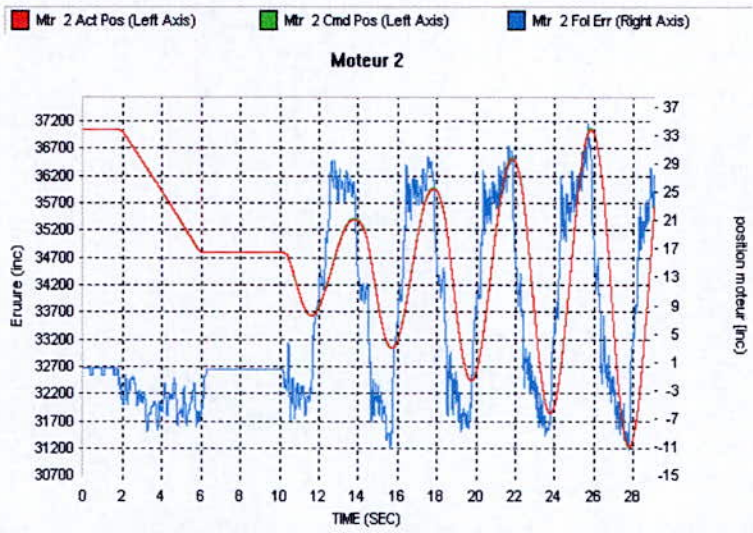


Figure VI-42 : *La position articulaire et erreur de poursuite du moteur 2*

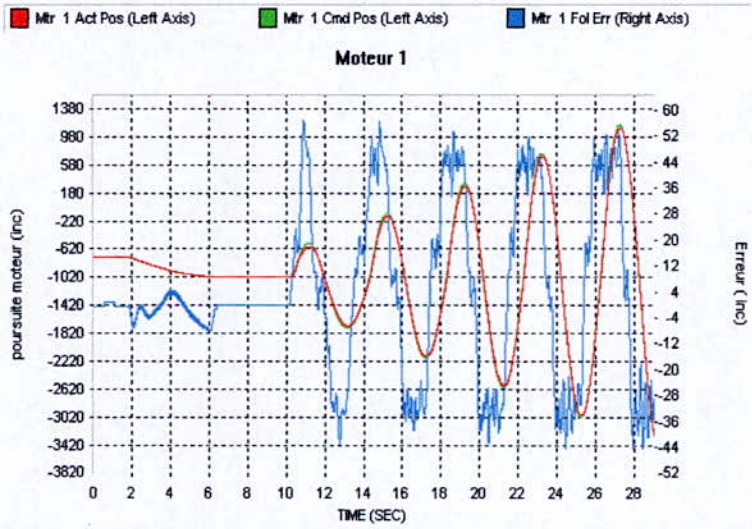


Figure VI-43 : *La position articulaire et erreur de poursuite du moteur 1*

Les **Figures VI-42** et **Figure VI-43** illustrent l'évolution des moteurs 1 et 2 dans le domaine articulaire, à partir des quelle on peut dire que la trajectoire de l'outil terminal (effecteur), présenteras la même allure que celle représentée dans la **Figure VI-41**, en effet la non régidité de l'effecteur du robot exploité, fait que la trajectoire dessinée par ce dernier (en le dotant d'un stylo a son extrémité) présente des profils sinusiens, ceci est dû, en partie, au manque de regidité de l'effecteur cité auparavant, et au jeux de mouvements présent au niveaux des articulations/actionneurs.

Toute fois, les **Figures VI-41** et **Figure VI-42** nous renseignent sur la poursuite de la trajectoire désirée.

VI.V.5- Ecriture :

- Dans ce qui suit nous avons essayés d'affecter une tâche au robot qui est l'écriture d'un caractère :

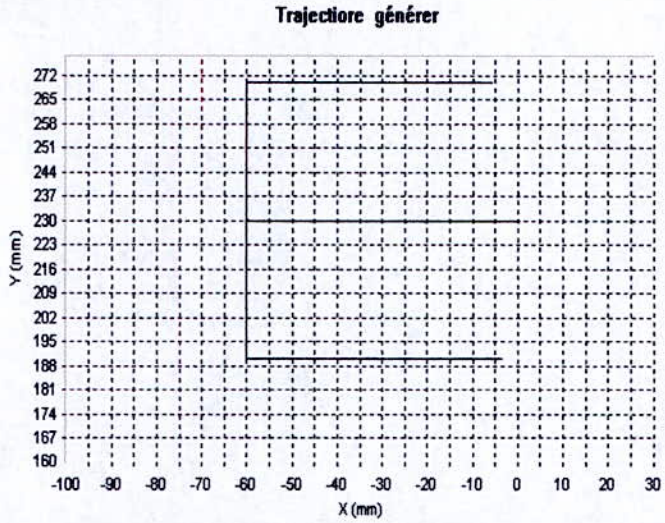


Figure VI-44 : *Trajectoire dans l'espace cartésien.*

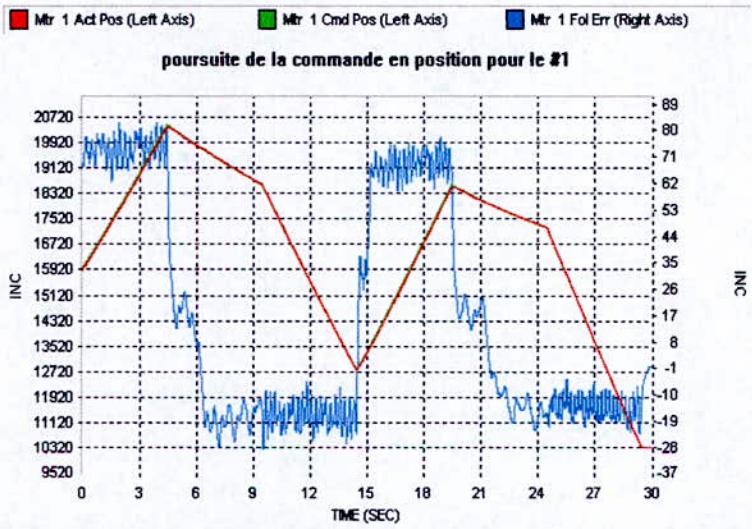


Figure VI-45 : *La position articulaire et erreur de poursuite du moteur 1*

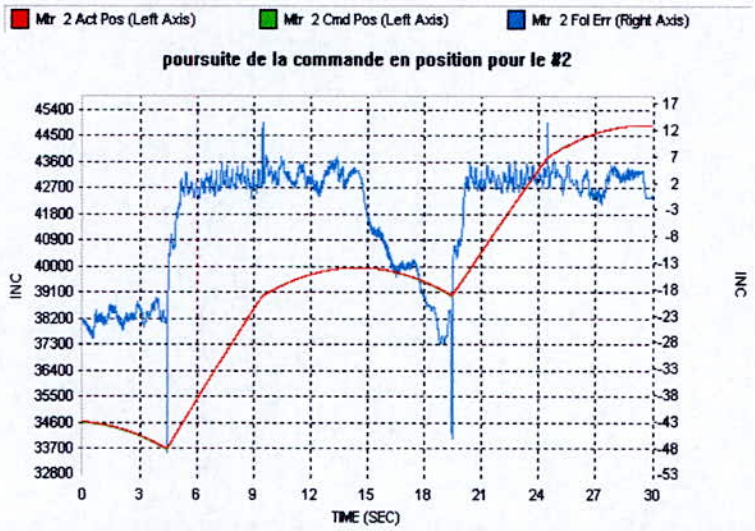


Figure VI-46 : *La position articulaire et erreur de poursuite du moteur 1*

De même, a partir des **Figure VI-45** et **Figure VI-46**, on observe que les trajectoires des moteurs dans l'espace articulaire suivent parfaitement les trajectoires de référence avec une erreur de l'ordre de 0.10° pour le moteur 2 et de 0.20°pour le moteur 1.

- Ecriture d'une chaîne de caractère :

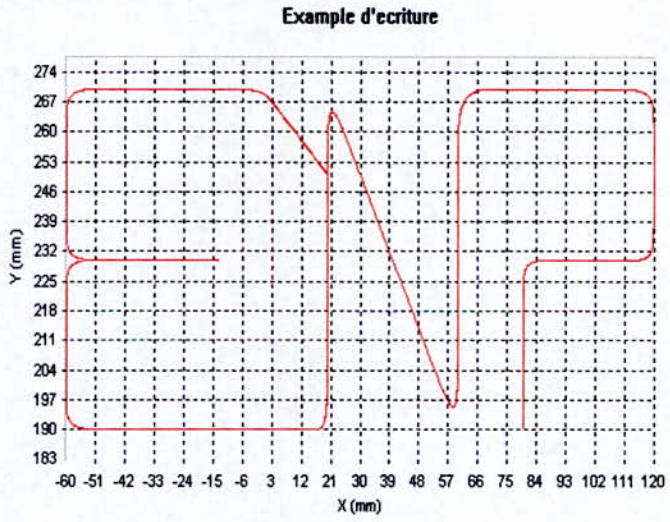


Figure VI-47 : *Trajectoire dans l'espace cartésien.*

VI-VI- Conclusion :

Dans la première partie de ce dernier chapitre nous avons présentés la détermination des paramètres des différentes structures du régulateur (PD, PID, PID avec *feed forward*) et discutés les performances de chaque structure en positionnement et en poursuite. Nous avons constatés que c'est la structure PID avec *feed forward* qui assure le positionnement et la poursuite.

Dans la deuxième partie nous avons exécuté quelques programmes de mouvement dans l'espace articulaire et opérationnel.

Pour l'espace articulaire nous avons vu les différents profils position, vitesse et accélération pour des types de mouvement différents : linéaire, Spline et PVT.

Pour l'espace opérationnel nous avons implémentés les modèles géométriques pour pouvoir planifier des trajectoires dans cet espace ensuite nous avons exécutés trois mouvements : spiral, écriture d'un caractère et l'écriture d'une chaîne de caractères

CONCLUSION GENERALE :

L'objectif de notre travail est d'étudier l'interface UMAC afin d'aboutir à son utilisation pour commander un bras manipulateur de type SCARA.

La documentation sur l'interface se trouve seulement sur le site du fabriquant Delta Tau [1] sous forme de manuels d'utilisation et de documents à vocation commerciale qui nous ont pas facilités la tache pour plus de détails dans l'étude de l'interface.

Dans ce mémoire nous avons étudiés les constituants de l'interface, son fonctionnement et sa programmation pour aboutir à l'objectif désiré.

Nous avons détaillés les différents types de mouvements, leurs interpolations, et exécutions par l'interface, ainsi que l'asservissement des références désirées issus des générateurs de mouvements. Ceci a pour objectif la détermination des différents gains de cet asservissement qui vont garantir les performances souhaitées.

Et afin d'assurer un bon fonctionnement du point de vu sécurité matériel et humaine, nous avons étudiés les différentes limites software et hardware disponibles sur l'interface. En effet l'installation des contacteurs de fin de course sur le robot, et leur mise en exploitation par l'UMAC, ainsi que la détermination des limites software, nous a permis de bien délimiter l'espace de travail, et minimiser les risques de collision (exemple : l'outil avec la base), lors d'une erreur de programmation de la trajectoire.

Nous avons vu aussi les programmes géométriques inverses et directs, leur gestion et exécution par l'interface. Où constatation faite sur leur efficience et leur flexibilité, en effet, l'implémentation de ces deux modèles nous a permis de générer des trajectoires, qui sont parfois fastidieuses à exécuter sur un contrôleur simple.

Tout cela nous a permis d'atteindre l'objectif fixé qui est de commander le robot SCARA pour exécuter différents types de trajectoires dans les espaces articulaire et opérationnel.

Vu la puissance et la flexibilité de l'interface UMAC, nous proposons comme perspectives :

- L'utilisation de l'IOGATE, pour gérer les interactions du bras manipulateur avec son environnement ;
- L'implémentation d'autres algorithmes de commande, pour permettre différentes approches de control que celle du PID ;

- Commande d'un bras de robot ayant des degrés de liberté supérieur a trois degrés de liberté ;

- Exploitation des programmes PLC et PLCC, pour l'implémentation d'automates de gestion de l'environnement du robot.

REFERENCES BIBLIOGRAPHIQUES :

[1] DELTA TAU DATA SYSTEMS, INC. Delta Tau FTP Site, http://www.deltatau.com.

[2] WISAMA KHALIL et ETIENNE DOMBRE, « *Modélisation Identification Et Commande Des Robots* », (2e édition revue et augmentée).

[3] MARK W. SPONG et M. VIDYASAGAR, « *Robot Dynamics And Control* », Edition WILEY, 1987

[4] YUN LI, KIAM HEONG ANG, and GREGORY C.Y. CHONG, "*PID Control System Analysis and Design, Problems, Remedies, and Future Directions*", IEEE Control Systems Magazine. Février, 2006

[5] K.J. ÅSTRÖM and T. HÄGGLUND, « *PID Controllers: Theory, Design, and Tuning.* », Research Triangle Park, NC: Instrum. Soc. Amer., 1995.

[6] BESTune, PID controller tuning [Online]. May 2004. Available: http://bestune.50megs.com

[7] VARUN AGGARWAL, MENG MAO and UNA-MAY O'REILLY, "*A Self-Tuning Analog Proportional-Integral-Derivative (PID) Controller*", Adaptive Hardware Group Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology

[8] H. HJALMARSSON and T. BIRKELAND, "*Iterative feedback tuning of linear time-invariant mimo systems. ",* In 37th IEEE Conference on Decision and Control, pages 3893–3898, 1998.

[9] K. J. ASTROM and T. HAGGLUND. "*Method And An Apparatus In Tuning A PID Regulator*", 1985.

[10] Cox, C.S., W.J.B. Arden and A.F. Doonan. "*Software Facilitates Tuning of Tra-ditional & Predictive Control Strategies*". Anaheim, CA. (1994)

[11] CHATSWORTH and CALIF, "*Stepping off the plane* ", Delta Tau Data Systems, Inc. . DECEMBER 2004

[12] DELTA TAU DATA SYSTEMS, INC. "*UMAC Turbo (3U Turbo PMAC2) Hardware Reference Manua*",
Disponible en format PDF de ftp://ftp.deltatau.com/UMAC/umactbhw.pdf, Août 2001.

[13] DELTA TAU DATA SYSTEMS, INC. "*UMAC ACC-24E2A, the analog +/- 10 Volts axes board*", Disponible en format PDF de ftp://ftp.deltatau.com/UMAC/acc24e2a.pdf, Octobre 2001.

[14] DELTA TAU DATA SYSTEMS, INC. "*UMAC I/O Accessory, HIGH POWER OPTO 24 INPUT/24 OUTPUT BOARD*",
Disponible en format PDF de : ftp://ftp.deltatau.com/UMAC/acc12e.pdf, Décembre 2000.

[15] DELTA TAU DATA SYSTEMS, INC. "*AMP-2, four axes analog PWM amplifier, 48 VDC 3/5 A*", Disponible en format PDF de ftp://ftp.deltatau.com/UMAC/amp-2.pdf, Avril 2001.

[16] DELTA TAU DATA SYSTEMS, INC. "*UMAC Power supply: 20 A / 5 V, 1 A/ +/- 15 V*", Disponible en format Pdf de ftp://ftp.deltatau.com/UMAC/acce2.pdf, Février 2002.

[17] DELTA TAU DATA SYSTEMS, INC."*UMAC UBUS Preliminary Specification*",
Disponible en format pdf de ftp://ftp.deltatau.com/UMAC/ubus.pdf, Avril 2000.

[18] DELTA TAU DATA SYSTEMS, INC. "*PMAC2 USER'S MANUAL*",
Disponible en format PDF de ftp://ftp.deltatau.com/NewIdeasInMotion/pmac2usr.pdf, Janvier2000.

[19] DELTA TAU DATA SYSTEMS, INC. *" PMAC (1) Family Quick Reference Manual version 2.1".*
Disponible en format pdf de ftp://ftp.deltatau.com/NewIdeasInMotion/pmacqref.pdf, Juin 2000

[20] DELTA TAU DATA SYSTEMS, INC. *"Turbo PMAC software manual (version 1.936)"* , Disponible en format pdf de ftp://ftp.deltatau.com/PMACManual/t1t2sw.exe, Septembre 2001.

[21] DELTA TAU DATA SYSTEMS, INC. *"PMAC1 USER'S MANUAL"*, Disponible en format pdf de ftp://ftp.deltatau.com/NewIdeasInMotion/pmac1usr.pdf, Janvier 2000.

[22] NATIONAL SEMICONDUCTOR. *"LMD 18200 H-Bridge Data Sheet"*, Disponible en format pdf de http://www.nsc.com/ LMD18200.pdf, Décembre 1999.

FICHE TECHNIQUE

LA CARTE Turbo PMAC2 3U CPU

Une unité de traitement freescale 56303 80MHz 24bits
- mémoire flash (1Mx8bits) pour la sauvegarde des programmes des utilisateurs et les microprogrammes
- mémoires SRAM (128Kx 24bits) active pour les programmes, la compilation et l'assemblage
- mémoire SRAM (128Kx24bits) des données de l'utilisateur
- connecteur série RS-232/422
- whatcdog timer
- registre de control

Caractéristique de la DSP56303

- 80 millions instructions par seconde (MIPS) à une fréquence horloge de 80MHz
- Architecture Harvard parallèle permettant l'exécution d'une instruction en parallèle avec l'accès mémoire
- Technologie CMOS très faible consommation
- Code instruction compatible avec le noyau DSP56303
- Unité arithmétique et logique de données (ALU) avec un multiplieur accumulateur parallèle de 24bits ou 16bits sous la commande de logiciel
- Unité de commande de programme (PCU), mode adressage optimise pour les applications de DSP, contrôleur instruction cache intégré.

Carte DSPGATE

-commande en modulation de largeur des impulsions de +/-10V
-commande analogique en couple de +/- 10V.
-commande impulsion et direction pour les moteurs pas à pas.

CARTE IOGATE

24 entrées optiquement isolées et 24 sorties à tension élevée optiquement isolées.
Carte d amplificateur AMP-2
Quatre axes analogiques, amplification PWM, 48VDC 3/5A.
UBUS (UMAC BUS) NORME ANSI/IEEE **STD1014-1987**