Modélisation, identification et commande dynamique d'un robot d'architecture parallèle

Philippe Lelias

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL DE BRETAGNE

MEMOIRE D'INGENIEUR EN AUTOMATIQUE

par

(LELIAS Philippe)

Soutenu le 13 mars 2013

à l'Institut de Recherche en Communications et Cybernétique de Nantes

(IRCCyN)

École Centrale de Nantes, France

TITRE :

Modélisation, identification et commande dynamique d'un robot d'architecture parallèle

JURY

PRESIDENT : BOURLES Henri

MEMBRES : GUGLIELMI Michel

CHABLAT Damien

LEMOINE Philippe

Remerciements :

Je voudrais en tout premier lieu exprimer toute ma gratitude aux deux acteurs du Conservatoire National des Arts et Métiers, Messieurs Henri Bourlès et Michel Guglielmi, qui m'ont permis de terminer mon cursus d'ingénieur en automatisme industriel

Je tiens également à remercier Monsieur Philippe Lemoine de m'avoir accompagné au quotidien tout au long de ce mémoire sur le plan technique. Enfin, Monsieur Damien CHABLAT, concepteur à l'IRCCyN, pour son expertise scientifique.

Un dernier remerciement aux membres du laboratoire IRCCyN en particulier le service robotique où règne une ambiance innovante et passionnante.

RESUME

Le projet "Orthoglide" a pour but de développer une machine-outil 3 axes d'architecture parallèle, extensible à 5 axes, ne présentant pas les inconvénients inhérents aux mécanismes parallèles. Il a été mis au point une démarche de conception optimale qui repose sur un critère mathématique : l'isotropie. Ce critère traduit l'homogénéité des performances en tout point et dans toutes les directions de l'espace de travail prescrit. Le prototype construit est opérationnel doit être analysé et ses performances validées. Cet objectif est celui du sujet proposé. Des méthodes ont été développées pour faire l'identification géométrique des offsets moteurs par l'observation des déplacements des parallélogrammes du porteur. Cette phase de calibration est essentielle pour garantir la géométrie des pièces usinées.

Avant de calibrer le robot, il est nécessaire de mettre en œuvre sa commande dynamique, Le travail réalisé dans ce mémoire est une contribution à l'ensemble des opérations nécessaires à la modélisation de ce prototype.

Mots clés : Modèle, direct, inverse, géométrique, cinématique, dynamique, identification, calibration et offset.

SUMMARY

The "Orthoglide" is a 3-axis machine tool with parallel architecture, expandable to 5 axes, doesn't have the disadvantages inherent parallel mechanisms. An approach has been developed to optimal design based on a mathematical criterion: Isotropy. Isotropy reflects the uniform performance at all points and in all directions in the workspace required. The prototype is ready to be analyzed and its performance validated. It's the aim of this thesis. Methods have been developed to identify geometric offset motor by observing the movements of the parallelograms. This calibration ensures the right geometry of the pieces.

Before calibrating the robot, it's necessary to implement its dynamic control, the work done in this thesis is a contribution to all the necessary operations for the modeling of this prototype

Keywords: Model, direct, inverse geometric, kinematic, dynamic, identification, calibration and offset

Table des figures

1 INTRODUCTION:

Les machines outils actuelles ont le plus souvent une architecture sérielle qui impose un empilage des axes les uns sur les autres, cette architecture limite la performance dynamique de ces machines du fait de masses élevées à déplacer. Une des idées innovante est l'architecture parallèle qui allège les masses mobiles. La naissance des premiers prototypes de ce type date du début des années 50. Ces machines cinématiques parallèles nommées **PKM** (Parallel Kinematics Machine) en anglais ont une masse en mouvement faible ce qui réduit l'inertie permettant des plus grandes performances dynamiques. Les chercheurs et industriels ont donc trouvé un intérêt à développer ce type de machine. Les difficultés de conception de cette architecture résident, en grande partie, dans la garantie d'une homogénéité des performances au sein d'un espace de travail restreint.

Au sein de l'IRCCyN, l'équipe de robotique a développé un prototype de machine parallèle appelé Orthoglide. Un premier prototype 3 axes a été conçu et étudié. Les résultats obtenus ont conduit à la réalisation d'une machine 5 axes de plus grande dimension destinée à des opérations d'usinage. Cette machine, encore expérimentale, doit être validée. En particulier, il faut implanter une commande dynamique. Le travail réalisé dans ce mémoire est une contribution à l'ensemble des opérations nécessaires à la caractérisation des performances de ce prototype. Le mémoire qui résulte de ce travail est décomposé comme suit :

Le premier chapitre présente un historique des machines parallèles, depuis les premiers mécanismes de la plateforme de Gough à la machine actuelle appelée **Orthoglide.**

Le deuxième chapitre définit les objectifs de conception en mettant en exergue son caractère isotropique.

Le troisième chapitre traite des outils mathématiques qui permettent de passer de l'environnement de travail de l'outil final (espace cartésien) à l'espace des actionneurs (espace articulaire). Plusieurs modèles sont nécessaires : les modèles géométriques, cinématiques et dynamiques. Pour tous ces modèles, il est indispensable de développer les deux transformations : à savoir articulaire vers le cartésien, que l'on appelle directe, et le cartésien vers l'articulaire dénommée alors inverse.

Le quatrième chapitre concerne la commande dynamique du robot.

Le cinquième chapitre est l'étude et l'implémentation de la simulation de la commande dynamique, ainsi que la présentation de quelques résultats commentés.

Tous les modèles nécessitent la connaissance des paramètres. Il n'est pas possible de connaître a priori de façon exacte les paramètres. Il est donc indispensable d'identifier le système. Dans le sixième chapitre on présente une procédure d'identification des paramètres géométriques. Cette opération est connue sous l'appellation de calibration. Une méthode a été développée pour réaliser l'identification géométrique des « offset » moteurs par l'observation des déplacements du porteur. Cette phase de calibration est essentielle pour garantir la géométrie des pièces usinées c'est-à-dire la performance de la machine.

Enfin, en conclusion, nous aborderons les difficultés de réalisation à l'échelle industrielle et apporterons des perspectives de développement.

2 HISTORIQUE:

Les premiers travaux scientifiques sur les mécanismes parallèles datent des années 50, la plateforme de Gough (figure 2.1) [1] était destinée à tester les pneumatiques. Ce système à six degrés de liberté était composé de six vérins actionnés et reliés à une base par des joints de cardan et reliés à l'autre extrémité à une plateforme mobile par des articulations sphériques. D'autres utilisations ont été développées, notamment dans les applications des simulateurs de vol. L'avantage majeur de cette architecture parallèle par rapport à l'architecture sérielle est une meilleure rigidité mécanique pour un ratio de poids-charge bien plus faible. L'inconvénient principal réside dans l'espace de travail et la dextérité globale plus faibles que pour les systèmes sériels.

Figure 2.1 : Plateforme de Gough-Steward

Le développement important qui a suivi date de la fin des années 70 et début des années 80, a vu les scientifiques s'intéresser à une alternative aux technologies sérielles en particulier la famille des robots Delta dont ce mémoire traite.

La famille des robots Delta conçu par Clavel [2] est un robot manipulateur à trois degrés de liberté en translation. Il est composé de trois jambes identiques reliant une plateforme mobile à une base (figure 2.2).

Figure 2.2 : Familles des robots delta

Chaque jambe contient deux articulations pivots et un parallélogramme. La principale caractéristique de cette architecture est l'utilisation de parallélogramme qui contraint le mouvement de la plateforme mobile et permet un déplacement en translation dans le plan cartésien (X, Y et Z).

Les actionneurs sont fixés à la base et les parties mobiles sont légères et trois chaînes cinématiques relient la base à la plate-forme. Les robots Delta offrent donc une faible inertie et par conséquent un bon comportement dynamique. Ainsi, ils peuvent atteindre une vitesse égale à 10m / s et des accélérations jusqu'à 20g [3]. En raison de sa vitesse élevée, il est largement utilisé dans l'industrie d'emballage, médical, pharmaceutique et électronique. Enfin, Un actionneur supplémentaire et une barre centrale télescopique peut être ajoutée pour fournir une rotation d'un degré de liberté supplémentaire autour de l'axe de symétrie du robot, ce qui donne un robot à quatre degré de liberté.

D'autres domaines d'applications tels que l'usinage industriel, simulateur de vol et micro positionnement sont étudiés aujourd'hui.

Le premier prototype d'usinage à grande vitesse fut construit en 1994 par Giddings & Lewis nommée « hexapode » (figure 2.3)[4], il reprenait l'architecture de la plateforme de Gough-Stewart. Le volume de travail 700 X 700 X 750 mm est important mais les performances de la machine ne sont pas homogènes dans cet espace car les équations qui relient les mouvements de l'outil final aux actionneurs ne sont pas linéaires et engendrent une détérioration des performances. De plus, les moteurs se déplacent en fonction du plateau supportant l'outil final.

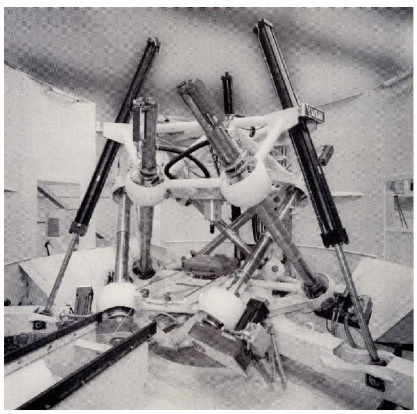


Figure 2.3: Machine outil Gidding & Lewis

Une alternative a été réalisée avec l'ETH de Zurich où a été développée une plateforme nommée « hexaglide » (figure 2.4) : les moteurs sont fixes, les jambes sont de longueur identique et glissent sur des rails, mais la dégradation des performances au sein de l'espace de travail n'est pas résolue, d'où la naissance de « l'orthoglide » décrite dans le chapitre suivant.

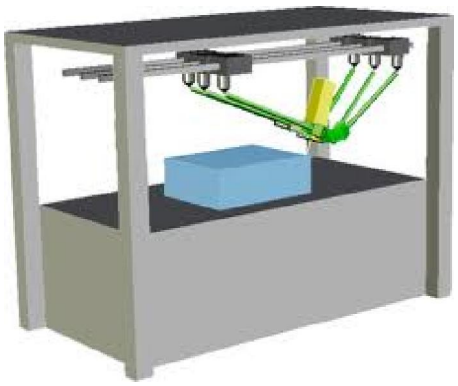


Figure 2.4 : La plateforme hexaglide

3 LA PLATEFORME ORTHOGLIDE:

Avant de présenter la plateforme Orthoglide, certaines définitions de la robotique parallèle sont nécessaires pour la bonne compréhension du lecteur.

3.1 Définitions de base :

Un robot parallèle est constitué de deux sous-ensembles bien distinct, son organe terminal et sa structure mécanique articulée. L'organe ou outil terminal est le dispositif destiné à transformer, manipuler un objet alors que sa structure à pour but de l'amener à une position et orientation données dans des conditions de vitesse et accélération fixées (figure 3.1).

Cette structure dans lequel l'organe terminal est relié à trois chaines parallèles assure une plus grande rigidité et donc une plus grande précision que les structures séries.

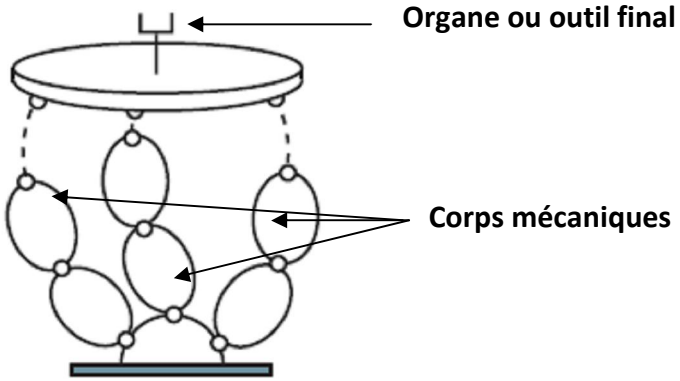


Figure 3.1 : Structure mécanique parallèle

Elle contient des chaines cinématiques dont les corps sont reliés entre eux par des articulations prismatiques ou rotoides. L'ensemble travaille dans deux espaces (opérationnel et articulaire) libre de singularités.

3.1.1 Les articulations :

Une articulation lie deux corps successifs en limitant le nombre de degrés de liberté (n) de l'un par rapport à l'autre. En robotique le cas le plus fréquent est n=1 dans ce cas l'articulation est soit rotoide soit prismatique.

✓ **Articulation rotoide :**

Il s'agit d'une articulation de type pivot réduisant le mouvement entre deux corps à une rotation autour d'un axe qui leur est commun. La situation relative entre deux corps est donnée par l'angle autour du même axe.

✓ **Articulation prismatique :**

Il s'agit d'une articulation de type glissière qui réduit le mouvement entre deux corps à une translation le long d'un axe commun. La position relative entre deux corps est mesurée par la distance le long du même axe.

3.1.2 Les espaces cartésien et articulaire :

✓ **L'espace cartésien ou opérationnel :**

L'espace opérationnel d'un robot est celui dans lequel est représentée la position de l'organe terminal (point **P**). On représente les coordonnées d'un point dans l'espace cartésien par le vecteur :

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

avec la représentation graphique (figure 3.2) du point P :

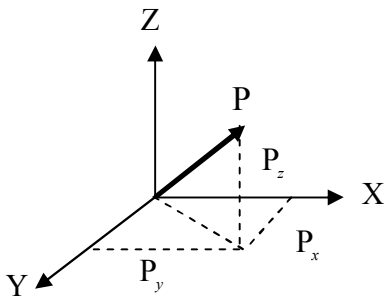


Figure 3.2 Représentation d'un point

✓ **L'espace articulaire :**

L'espace articulaire est celui dans lequel est représenté la position de tous les corps du robot, dans le cas de l'Orthoglide c'est la position des 3 actionneurs (trois prismatiques). On représente les coordonnées articulaires par :

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

3.1.3 Les singularités :

Les singularités sont un problème des architectures parallèles, aux configurations singulières le robot parallèle est exposé à un comportement inhabituel comme la perte ou un gain d'un degré de liberté, un mouvement dans une direction inatteignable ou encore un mouvement de l'outil final lorsque les actionneurs sont verrouillés.

La configuration de la figure 3.3 représente une singularité avec la perte d'un degré de liberté, il existe une direction le long de laquelle aucun mouvement n'est possible. La flèche indique la direction le long de laquelle le mouvement est impossible.

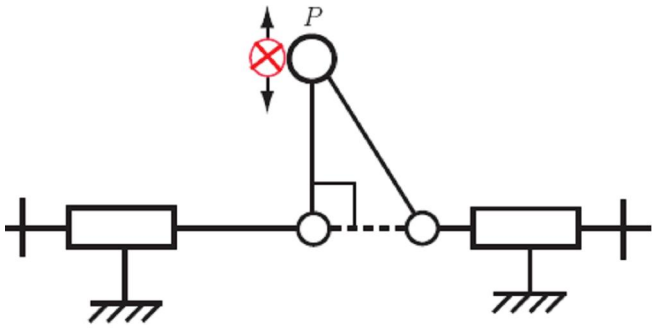


Figure 3.3 : Singularité avec une perte d'un degré de liberté

La configuration de la figure 3.4 représente une singularité avec un mouvement possible de l'effecteur ou du point alors que les actionneurs sont verrouillés. La flèche indique la direction le long de laquelle la rigidité est perdue.

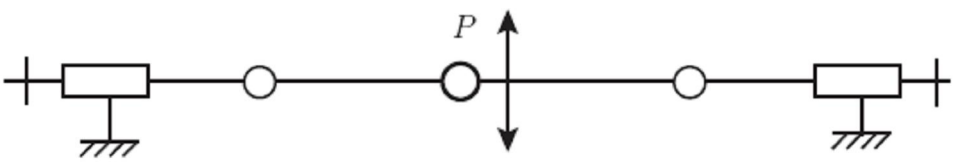


Figure 3.4 : Singularité de mouvement possible

3.2 Les objectifs de conception de l'Orthoglide :

Les difficultés de garantir une homogénéité des performances dans l'espace de travail a conduit à développer à l'IRCCyN une nouvelle plateforme apte à garantir cette homogénéité. Le projet s'est appuyé sur les spécificités et contraintes suivantes afin de concevoir une machine outil rapide, précise et performante dans tout son espace de travail à un coût économique faible (figure 3.5) :

- Réalisation d'une machine outil à taille industrielle à 5 axes suite à un prototype 3 axes.
- Les actionneurs sont fixes et glissants (diminution de l'inertie).
- Le volume de travail de forme régulière (500 X 500 X 500 mm^3).
- L'homogénéité des performances dans tout le volume de travail et dans toutes les directions est garantie avec l'isotropie.
- Les articulations sont simples et les jambes identiques pour un coût peu onéreux.

Figure 3.5 : Le robot Orthoglide

3.3 Choix de l'architecture cinématique :

Il est nécessaire de définir une architecture de base [5] afin de tenir les objectifs cités ci dessus. Il a été fait le choix pour l'Orthoglide d'un mécanisme à trois degrés de liberté en translation composée de chaines cinématiques identiques appelées ***jambes*** et décrites comme PRP_aR avec P pour prismatique, P_a pour parallélogramme et R pour rotoide (figure 3.6). Le mécanisme d'entrée est réalisé avec trois actionneurs d'articulations prismatiques (A_i, B_i) avec $i \in \{x; y; z\}$ et le mécanisme de sortie (outil final ***P***) est relié aux articulations prismatiques (C_i) aux trois jambes (figure 3.7). Chaque jambe est un parallélogramme et s'oriente de manière à déplacer le mécanisme de sortie d'un mouvement de translation

seulement. Les singularités du robot sont des caractéristiques importantes qui influencent les capacités de celui-ci.. Par conséquent, l'architecture a été définie pour éliminer toutes singularités et collisions dans l'espace de travail.

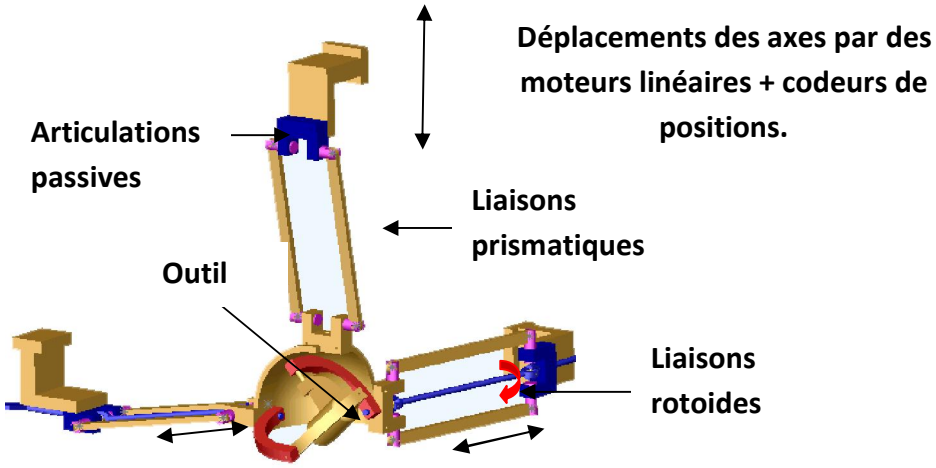


Figure 3.6 : Architecture cinématique de l'Orthoglide.

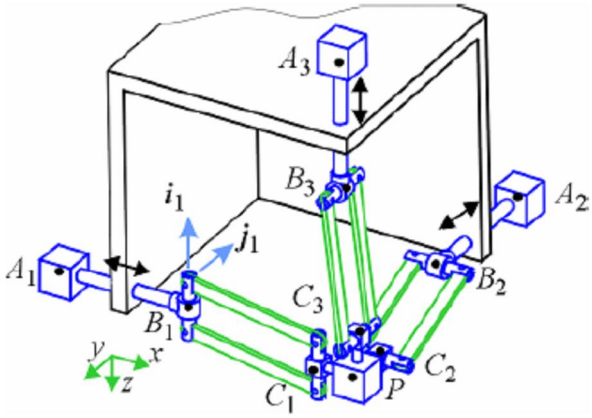


Figure 3.7 : Architecture cinématique

Après avoir étudiées différentes architectures parallèles, l'architecture de chaque jambe (figure 3.8) libre de toute singularité, a été définie :

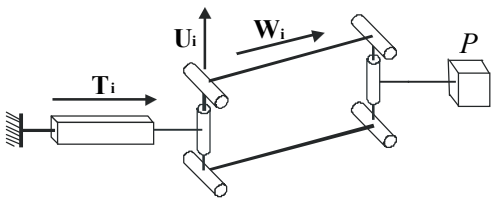


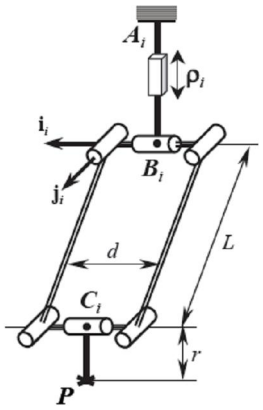
Figure 3.8 : Architecture cinématique d'une jambe

3.4 Matrice Jacobienne et conditions isotropique :

Prenant en compte les propriétés de bases des parallélogrammes, l'architecture cinématique de l'Orthoglide peut être réduite à une forme simplifiée (figure 3.10). Le modèle cinématique équivalent contient trois liaisons rigides reliées par des articulations sphériques au point de centre outil (***TCP*** ou point $P=(p_x, p_y, p_z)$) à un coté et l'autre coté aux articulations prismatiques. Dans [5], les équations de base cinématiques ont été écrites à savoir :

$$\begin{cases} (p_x-\rho_x)^2+p_y^2+p_z^2=L^2; \\ p_x^2+(p_y-\rho_y)^2+p_z^2=L^2; \\ p_x^2+p_y^2+(p_z-\rho_z)^2=L^2; \end{cases} \quad (3.1)$$

Avec (p_x, p_y, p_z) : vecteur de position de sortie ;
(ρ_x, ρ_y, ρ_z) : Vecteur d'entrée des variables d'articulations prismatique ;
L : la longueur des jambes (figure 3.9) ;



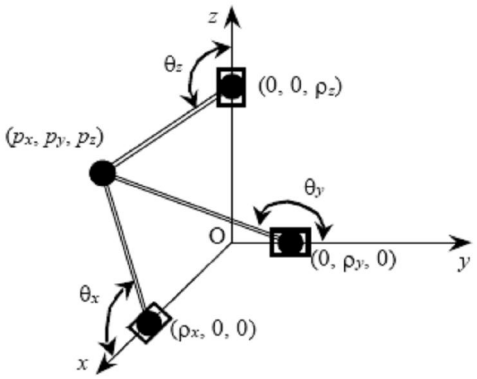


Figure 3.9 Architecture cinématique d'une jambe ***Figure 3.10 Architecture simplifiée***

Pour garantir une homogénéité des performances dans l'espace de travail, l'étude s'appuie sur la matrice Jacobienne (***J***) d'un mécanisme qui relie le vecteur d'entrée ρ au vecteur de sortie p . On utilise plutôt la matrice Jacobienne inverse $(J^{-1}) -> \rho = J^{-1}p$ qui permet de passer des coordonnées cartésiennes aux coordonnées articulaires.

$$J^{-1}(p,\rho)=\frac{\partial \rho}{\partial p}=\begin{pmatrix} 1 & \frac{p_y}{p_x-\rho_x} & \frac{p_z}{p_x-\rho_x} \\ \frac{p_x}{p_x-\rho_y} & 1 & \frac{p_z}{p_y-\rho_y} \\ \frac{p_x}{p_z-\rho_z} & \frac{p_y}{p_z-\rho_z} & 1 \end{pmatrix} \tag{3.2}$$

L'Orthoglide possède une configuration isotrope lorsque les trois jambes (parallélogrammes) sont orthogonales, les vitesses des mouvements des jambes sont exactement les vitesses de l'outil final. Il en est de même dans la transmission des efforts. Dans une configuration isotropique (figure 3.11)

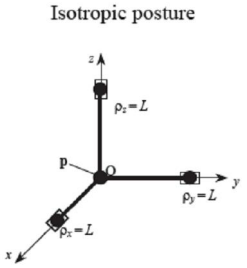


Figure 3.11 Architecture en position isotropique

les relations différentielles sont calculées autour du point ***P*** :

$$p=(p_x,p_y,p_z)=(0,0,0) \text{ et } \rho=(\rho_x,\rho_y,\rho_z)=(L,L,L) \tag{3.3}$$

qui après substitution de (3.2) donne la matrice d'identité Jacobienne suivante :

$$J(p_0,\rho_0)=I_{3x3} \tag{3.4}$$

Ce modèle mathématique de base établi montre donc que cette cinématique parallèle possède un volume de travail quasi cubique dont les performances sont homogènes. Le prochain chapitre traite des différents modèles géométriques et cinématiques du robot.

4 MODÉLES GÉOMÉTRIQUE, CINÉMATIQUE ET DYNAMIQUE

Afin de commander, simuler le comportement dynamique du robot Orthoglide, il est nécessaire de le modéliser, ces modèles permettront de connaitre le comportement géométrique, cinématique et dynamique de la machine. Ces outils mathématiques doivent permettre de passer de l'espace de travail de l'outil final (espace cartésien) à l'espace des coordonnées des actionneurs (espace articulaire) et réciproquement, d'où la notion de modèle inverse et direct. Il est présenté dans ce chapitre, les modèles géométriques, cinématiques, dynamiques direct et inverse.

4.1 Les modèles géométriques :

Le modèle géométrique direct calcule les coordonnées de position de l'organe terminal en fonction des variables articulaires. Il permet donc de passer de l'espace articulaire à l'espace cartésien. Il est donné par la relation suivante :

$$X = f(q) \ ; \tag{4.1}$$

Avec $X = \left[x_1, x_2, x_3\right]^T$ vecteur des coordonnées cartésiennes et $q = \left[q_1, q_2, q_3\right]^T$ le vecteur des variables articulaires.

Le modèle géométrique inverse calcule les variables articulaires en fonction des coordonnées cartésiennes de l'outil final, il est donné par la relation suivante :

$$q = f^{-1}(X) \ ; \tag{4.2}$$

4.2 Les modèles cinématiques :

Le modèle cinématique direct calcule les vitesses des coordonnées de l'outil final en fonction des vitesses articulaires, il est donné par la relation suivante :

$$\dot{X} = J(q)\dot{q} \ ; \tag{4.3}$$

Avec $J(q)$, la matrice Jacobienne de dimension (m x n) du mécanisme, n étant le nombre d'articulations du robots et m le nombre de degré de liberté.

Le modèle cinématique inverse calcule les vitesses articulaires en fonction des vitesses des coordonnées de l'outil final, il est donné par la relation suivante :

$$\dot{q} = J^{-1}\dot{X} \ ; \tag{4.4}$$

4.3 Les modèles dynamiques:

Le modèle dynamique inverse est la relation entre les couples appliquées aux actionneurs et les positions, vitesses et accélérations articulaires. Il est utilisé dans la commande dynamique afin de calculer les couples Γ. Il est présenté par la relation suivante :

$$\Gamma = f(q, \dot{q}, \ddot{q}, \ddot{I}_e) \tag{4.5}$$

Avec $\ddot{I}_e$ est le vecteur représentant l'effort extérieur qu'exerce le robot sur l'environnement et Γ est le vecteur des forces articulaires.

Le modèle dynamique direct exprime les accélérations articulaires en fonction des positions, vitesses et couples des articulations. Il est utilisé pour la simulation afin de calculer l'accélération à partir des couples simulés et il est présenté par la relation suivante :

$$\ddot{q} = g(q, \dot{q}, \Gamma, \ddot{I}_e) \tag{4.6}$$

4.4 Équation dynamique :

Pour traiter la commande dynamique du chapitre suivant, il est nécessaire de rappeler la forme générale de l'équation dynamique du robot Orthoglide développée dans l'annexe I.

$$\boldsymbol{\Gamma} = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) \tag{4.7}$$

Avec Γ, le vecteur des couples articulaires.

A et **H** qui sont respectivement la matrice d'inertie du robot et le vecteur des forces centrifuge et les forces de Coriolis.

Il est nécessaire de compléter la forme générale (4.7) avec l'introduction des frottements. De nombreuses études ont été réalisées afin de mieux analyser les frottements au niveau des articulations, des réducteurs et des transmissions. Les frottements non compensés provoquent des erreurs statiques. Différents modèles de frottements ont été proposés, le modèle de frottements secs $\mathbf{F_s}$ en opposition au mouvement et le modèle de frottements visqueux $\mathbf{F_v}$ qui est fonction de la vitesse. Ce qui amène au modèle dynamique plus complet suivant :

$$\boldsymbol{\Gamma} = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F_v}\dot{\mathbf{q}} + \mathbf{F_s} \tag{4.8}$$

5 LA COMMANDE DYNAMIQUE :

Les robots à structure parallèle permettent d'apporter un gain en vitesse, en cadence de production et en précision. Cependant, la structure mécanique complexe nécessite un développement de la stratégie de commande. Ainsi l'application qui exige rapidité et précision, rend nécessaire de concevoir un système de commande sophistiqué. La commande par découplage non linéaire plus connue sous le nom de commande dynamique est une bonne approche. Dans ce chapitre est présentée la commande dynamique.

5.1 Commande des mouvements :

5.1.1 Génération d'un mouvement :

La génération de mouvement consiste à calculer, à partir des coordonnées cartésiennes de l'organe terminal, les consignes de référence en position, vitesse et accélération qui détermineront le passage du robot aux points désirés.

A titre d'exemple, le profil de suivant (figure 5.1) est calculé dans l'espace cartésien et génère des points suivant un trois quart de cercle.

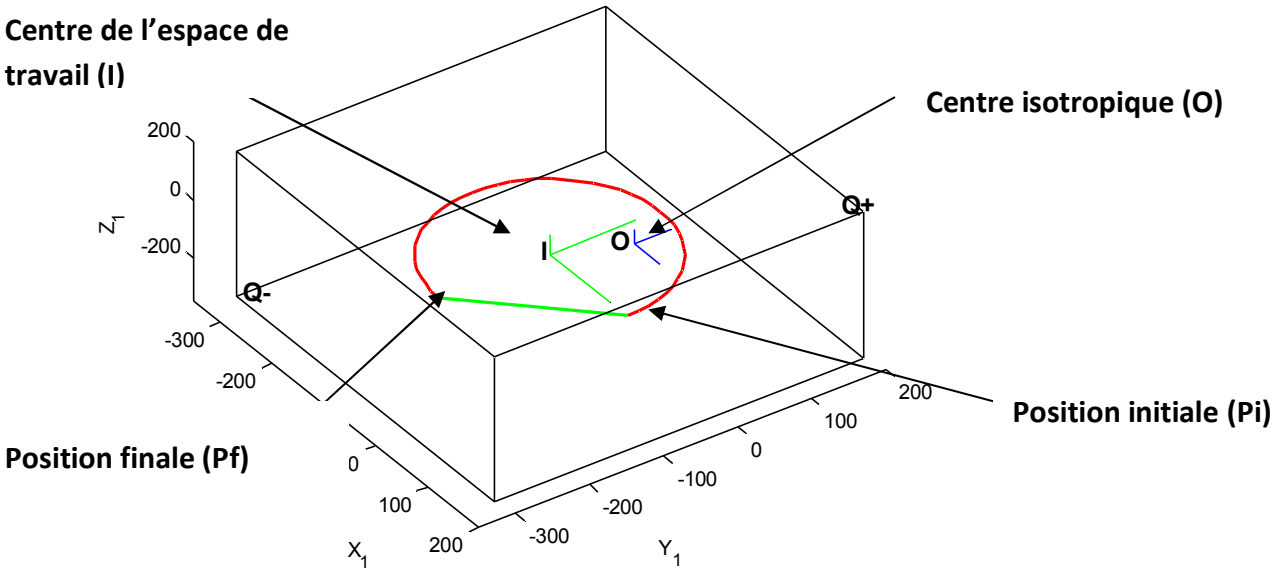


Figure 5.1 : Trajectoire en trois quart de cercle

L'ensemble des points est dans un espace de travail quasi cubique (ici 500 x 500 x 500 mm^3). Dans un plan parallèle au plan XY, les points sont définis par un rayon (R) et un angle Ψ qui est l'angle entre la trajectoire et l'axe Y (figure 5.2).

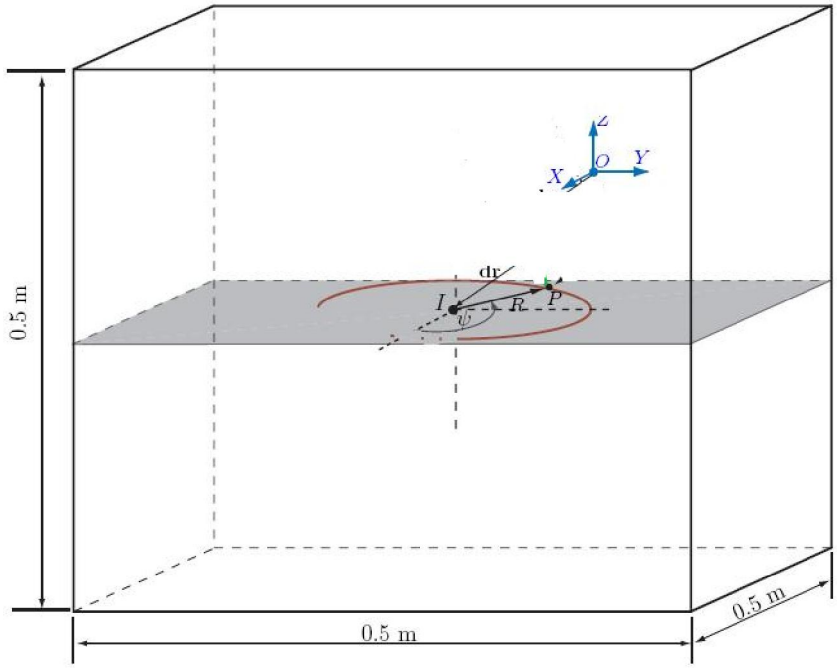


Figure 5.2: Géométrie de la trajectoire

Le vecteur de position d_r du point I qui est le centre géométrique de l'espace de travail cubique est donné par :

$$d_r = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \tag{5.1}$$

La position du point P dans l'espace cartésien est donc définie par :

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} d_x + R\cos\Psi \\ d_y + R\cos\Psi \\ d_z \end{bmatrix} \tag{5.2}$$

5.1.2 Génération du mouvement entre deux points :

La génération de trajectoire nécessite aussi une description du déplacement entre deux points. Il est classiquement utilisé que le temps soit représenté par un polynôme d'interpolation de degré 5 [6]. Avec ce degré, on peut assurer que la continuité des vitesses et accélérations ainsi que des vitesses et accélérations nulles aux positions initiales et finales. Il a été fait le choix du polynôme de degré 5 suivant :

$$r(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5 \tag{5.3}$$

Pour passer de la position initiale à la position finale en respectant les contraintes de vitesses et d'accélérations on calcule le temps minimum pour chaque articulation séparément puis on effectue une coordination des articulations sur un temps commun. Le temps minimum t_f est déduit de la méthode d'interpolation de degré 5 (5.3), calcule le temps qu'il faut pour une articulation pour réaliser un déplacement donné, le temps global minimal t_f est le temps mis par l'articulation la plus contraignante pour laquelle le temps minimum est le plus grand. Il est donné par [7] :

$$t_f = MAX\left[t_{f1},.....,t_{fN}\right] = MAX\left[\frac{15\left|D_j\right|}{8K_{vj}}\right];\sqrt{\left[\frac{10\left|D_j\right|}{\sqrt{3}K_{aj}}\right]} \tag{5.4]}$$

Avec j = 1 à N où N=3 qui représente le nombre de jambes motorisées, $D = P_f - P_i$ et K_a, K_v sont les vecteurs de vitesses et accélérations maximales respectivement.

De la trajectoire semi circulaire (figure 5.1), on présente les positions, vitesses et accélérations dans l'espace cartésien en fonction du temps (figure 5.3, 4 et 5 respectivement).

Vitesse nulle aux positions initiales et finales

Accélération nulle aux positions initiales et finales

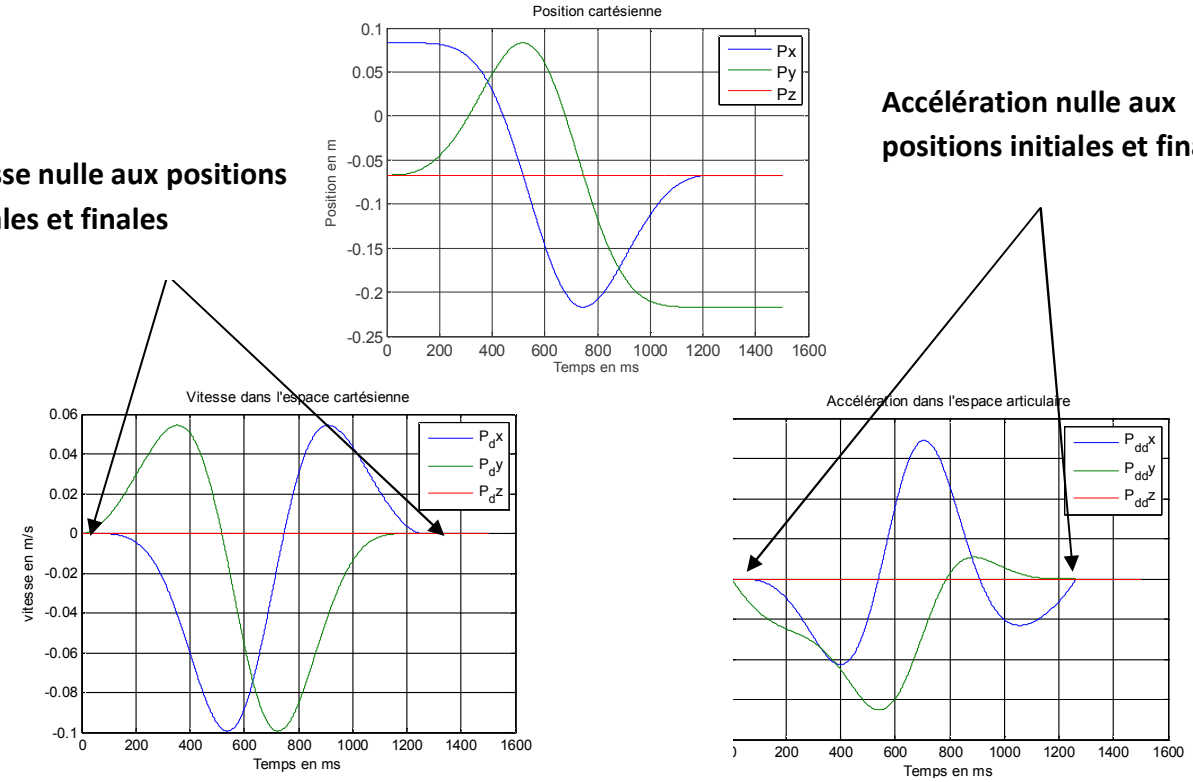


Figure 5.3, 5.4 et 5.5 : Positions, vitesses et accélérations dans l'espace cartésien

5.2 La commande dynamique :

Le moyen le plus simple d'asservir un système est bien entendu d'utiliser un correcteur PID classique qui génère à partir d'une erreur entre la consigne et le retour de mesure une consigne à l'actionneur qui peut être un couple.

Sylvain GUEGAN [8], dans sa thèse montre que les commandes classiques sont implantées dans la plupart des robots industriels actuels. Chacune des articulations est asservie, séparément, par un PID à gains constants. On peut noter que Le PID ne fait aucune hypothèse sur la linéarité du processus. Les avantages sont, bien évidemment, la facilité d'implantation et le faible coût en calcul. En contrepartie, la réponse temporelle du robot varie selon sa configuration et montre des dépassements de consigne et une mauvaise précision au suivi dans les mouvements rapides due à la non linéarité du système. Ceci a conduit à chercher des lois de commande plus performantes.

La commande dynamique est la méthode considérée comme la solution théorique idéale pour la commande des robots manipulateurs parallèles [9].

5.2.1 Principe de la commande dynamique

Le principe adopté est la commande par découplage non linéaire, il est intéressant de découpler les entrées et les sorties pour obtenir finalement un système où une entrée influence une sortie et une seule. Une fois réalisée ce découplage, il est alors aisé, en utilisant des techniques propres aux systèmes mono-entrée mono-sortie de pouvoir régler les performances du système linéarisé.

Il est possible de commander le robot dans l'espace cartésien qui nécessite un capteur externe comme une caméra par exemple. Notre robot possède des actionneurs et des informations des codeurs moteurs dans l'espace articulaire, la commande dans cet espace est développée.

La commande dynamique utilise le modèle dynamique en ligne et la connaissance des valeurs numériques des paramètres inertiels et de frottements. Cela consiste à transformer par retour d'état le problème d'un système non linéaire en un problème de commande d'un système linéaire.

Le nombre d'actionneurs est égal aux nombres de variables articulaires et l'utilisation du modèle dynamique inverse qui exprime l'entrée Γ en fonction de vecteur d'état (5.5)

$$\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} \tag{5.5}$$

avec l'état du système ($\mathbf{q}$ et $\dot{\mathbf{q}}$) permettent une loi de commande dynamique dans l'espace articulaire.

5.2.2 Commande dans l'espace articulaire :

W.Khalil [10] a développé la commande dynamique dans l'espace articulaire présentée dans ce paragraphe.

La commande $\boldsymbol{\Gamma}$ est déduite de l'équation dynamique du chapitre 4 (4.8) rappelée ici :

$$\boldsymbol{\Gamma} = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F_v}\dot{\mathbf{q}} + \mathbf{F_s} \tag{5.6}$$

Comme indiqué dans l'introduction de ce chapitre, la commande dynamique utilisant ce modèle dynamique assure le découplage. La linéarisation du modèle dynamique en ligne a pour effet de donner un comportement uniforme quelle que soit la configuration du robot. Dans notre cas, qui est idéal qui suppose le modèle dynamique parfait, un nouveau vecteur d'état est obtenu, il est strictement égal à l'accélération et il est donné par :

$$\ddot{\mathbf{q}} = \mathbf{w}(t) \tag{5.7}$$

On se ramène donc à un problème de commande d'un système linéaire à n états, invariant du second ordre (double intégrateur ici). La génération du mouvement désiré est complètement spécifiée. On aboutit au schéma de principe (5.6) suivant :

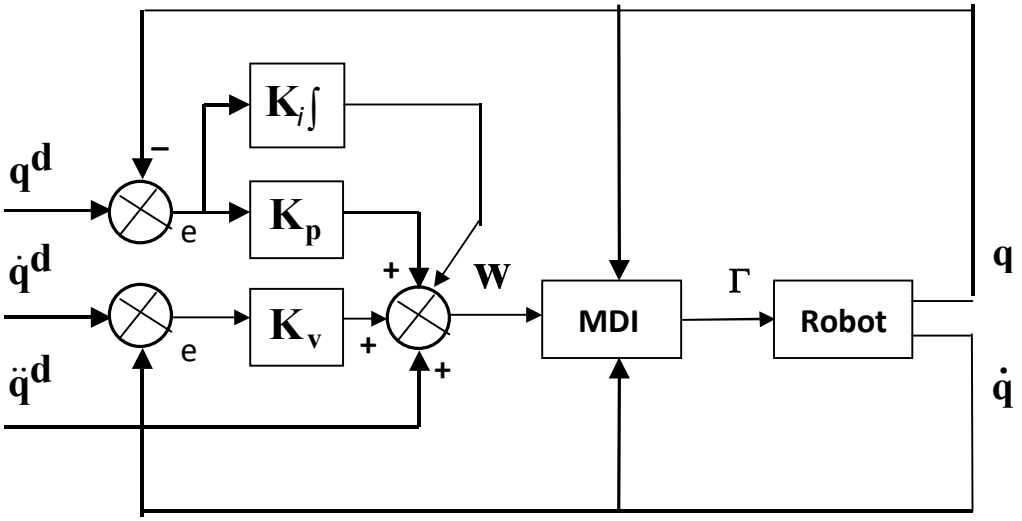


Schéma 5.6 : Loi de commande de l'Orthoglide

Soit $\mathbf{q^d}, \dot{\mathbf{q}}^\mathbf{d}, \ddot{\mathbf{q}}^\mathbf{d}$ la position, la vitesse et l'accélération désirées dans l'espace articulaire et $\mathbf{q}, \dot{\mathbf{q}}$ la position et vitesse mesurées par les codeurs moteurs. Nous calculons $\mathbf{w}(t)$ c'est à dire l'accélération selon le schéma 5.6 qui donne la relation suivante :

$$\mathbf{w}(t) = \ddot{\mathbf{q}}^\mathbf{d} + \mathbf{K_p}(\mathbf{q^d} - \mathbf{q}) + \mathbf{K_v}(\dot{\mathbf{q}}^\mathbf{d} - \dot{\mathbf{q}}) + \mathbf{K_i}\int_{t_0}^{t}(\mathbf{q^d} - \mathbf{q}) \tag{5.8}$$

Suite au découplage avec le nouveau vecteur de commande **w(t)**, nous obtenons un système linéaire.

Le couple Γ est obtenu de (5.6) et (5.8), donné par :

$$\Gamma = \mathbf{A}(\mathbf{q})\mathbf{w}(t) + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F_v}\dot{\mathbf{q}} + \mathbf{F_s} \tag{5.9}$$

Les bases d'un système linéaire peuvent donc être appliquées :

- L'erreur $\mathbf{e} = (\mathbf{q^d} - \mathbf{q})$ sont globalement exponentiellement stables [10].
- Les gains $\mathbf{K_p}, \mathbf{K_v}, \mathbf{K_i}$ sont choisis pour imposer le comportement dynamique d'un axe.
- L'amortissement est $\zeta = 1$ afin d'obtenir la réponse la plus rapide et sans dépassement.
- La pulsation ω est strictement positive

Ce qui amène aux relations suivantes liées à un modèle supposé parfait :

$$\begin{cases} \mathbf{K_v} = (2\zeta + 1)\boldsymbol{\omega} \\ \mathbf{K_p} = (2\zeta + 1)\boldsymbol{\omega}^2 \\ \mathbf{K_i} = \boldsymbol{\omega}^3 \end{cases} \rightarrow \begin{cases} \mathbf{K_v} = 3\boldsymbol{\omega} \\ \mathbf{K_p} = 3\boldsymbol{\omega}^2 \\ \mathbf{K_i} = \boldsymbol{\omega}^3 \end{cases} \quad (5.10)$$

Avec une pulsation calculée ($\boldsymbol{\omega} = 50$) on obtient les coefficients suivant :

$$\begin{cases} \mathbf{K_v} = 150 \\ \mathbf{K_p} = 7500 \\ \mathbf{K_i} = 125000 \end{cases} \quad (5.11)$$

6 LA SIMULATION :

Les modélisations géométriques, cinématiques, dynamiques et la commande dynamique sont testées en simulation afin de montrer les performances théoriquement atteignables. Nous supposons que le robot est correctement représenté par le modèle dynamique direct, il permet de calculer l'accélération de l'effecteur dans l'espace articulaire et en intégrant 2 fois, on obtient sa position en tenant compte du couple appliquée aux actionneurs.

6.1 Schéma de principe :

Le (Schéma 6.1) présente une trajectoire (points, vitesses et accélérations) générée dans l'espace cartésien. Par, le MGI, on calcule les consignes dans l'espace articulaire, le MDI permet de calculer le couple (Γ) et le MDD fournit enfin le retour des mesures de l'accélération. La vitesse et la position sont obtenues par intégrations successives.

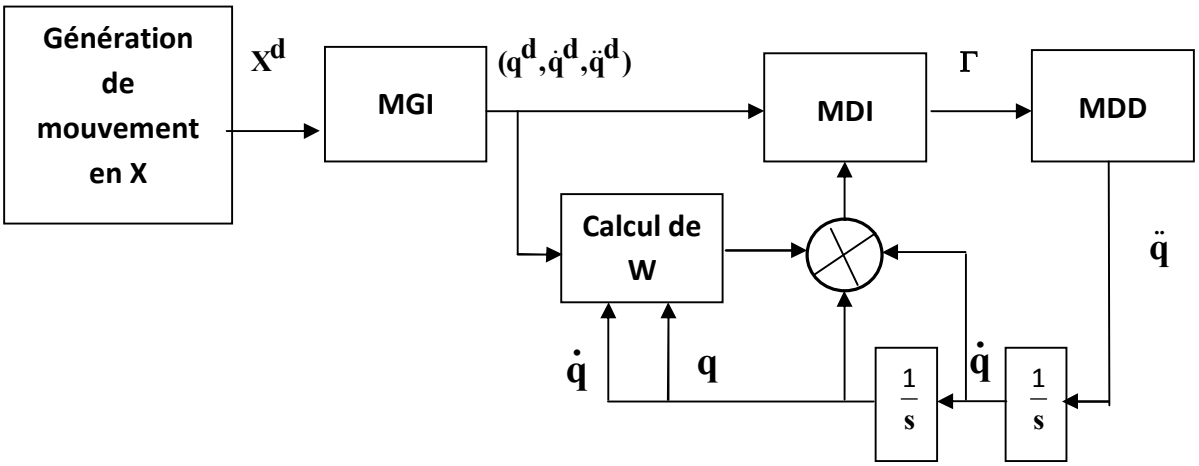


Schéma 6.1 : Principe de la simulation

6.2 Résultats et discussions :

La simulation est effectuée sous le logiciel MATLAB SIMULINK (un schéma est présenté dans l'annexe II), les différentes implémentations des fonctions sont testées à savoir MATLAB FCN, EMBEDED MATLAB et les S-FUNCTION. L'utilisation des ces différentes fonctions n'a pas pour but de tester les résultats de la commande mais a pour objectif d'optimiser à la baisse les temps de calculs. Les comparaisons se font dans l'espace

articulaire des positions, vitesses et accélérations désirées aux positions, vitesses et accélérations estimées.

6.2.1 Temps et optimisations de la simulation :

Le temps de fonctionnement simulé est de 12 s. Le temps moyen de simulation sous une implémentation de fonctions MATLAB FCN est de 6 minutes avec un échantillonnage de 0.01 s qui est un bon compromis entre le temps de simulation et des bons résultats. A titre d'étude et de comparaison, différentes implémentations ont été effectuées dans la génération de mouvement, il était judicieux d'optimiser les temps de calcul du polynôme de degré 5. Les implémentations en MATLAB FCN et S FUNCTION sont proposées en annexe III et IV respectivement.

L'implémentation en C avec les S FUNCTION apporte un gain d'une minute. On utilise une fonction C et le transfert des paramètres géométriques s'effectue à travers cette fonction. Le temps de calcul du polynôme de degré 5 diminue significativement mais il est biasé avec la transmission des paramètres à chaque pas d'échantillonnage. Il serait intéressant d'implémenter en C l'ensemble de l'application, ce qui diminuerait probablement le temps de calcul significativement.

6.2.2 Positions articulaires :

La figure (6.2 a) représente la position calculée, désirée dans l'espace articulaire et la figure (6.2 b) représente le résultat de la simulation c'est-à-dire les positions obtenues par simulation des sorties dans l'espace articulaire. Il n'est montré aucune différence visible, l'ensemble de la simulation est donc valide.

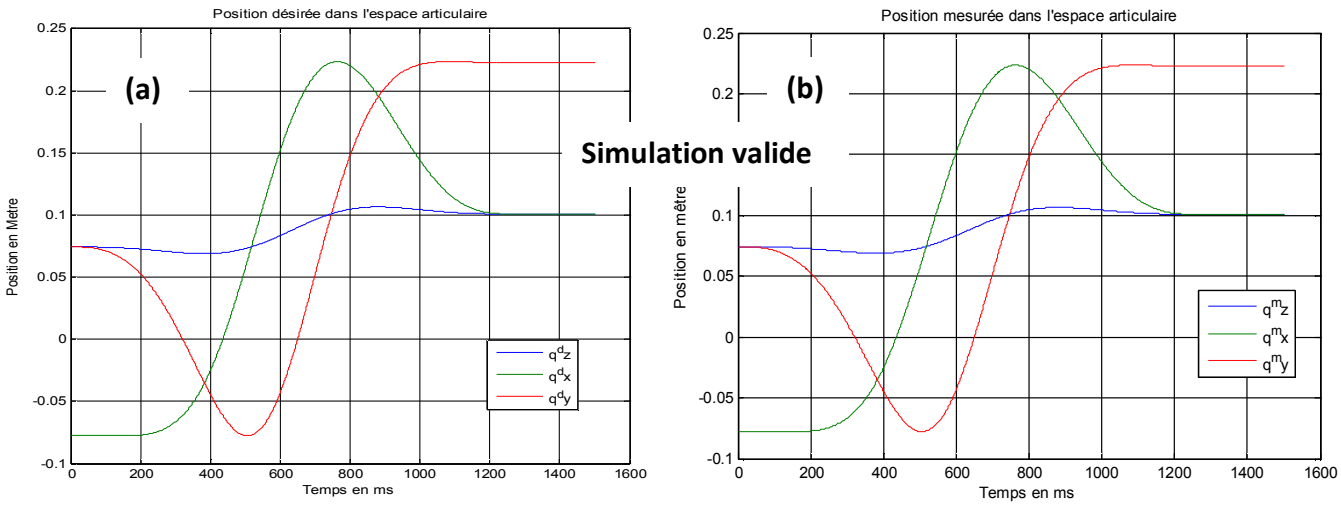


Figure 6.2 : Résultats des positions dans l'espace articulaire

6.2.3 Vitesse articulaire :

La figure (6.3 a) représente la vitesse calculée, désirée dans l'espace articulaire et la figure (6.3 b) représente le résultat de la simulation c'est-à-dire les vitesses obtenues par simulation des sorties dans l'espace articulaire. Il n'est montré aucune différence visible.

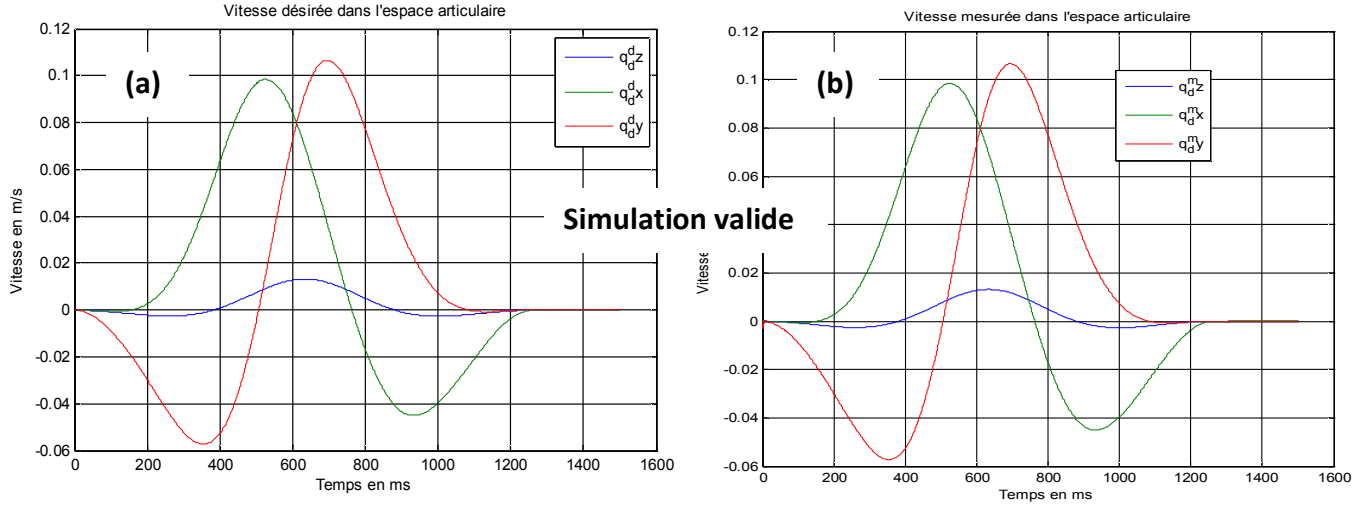
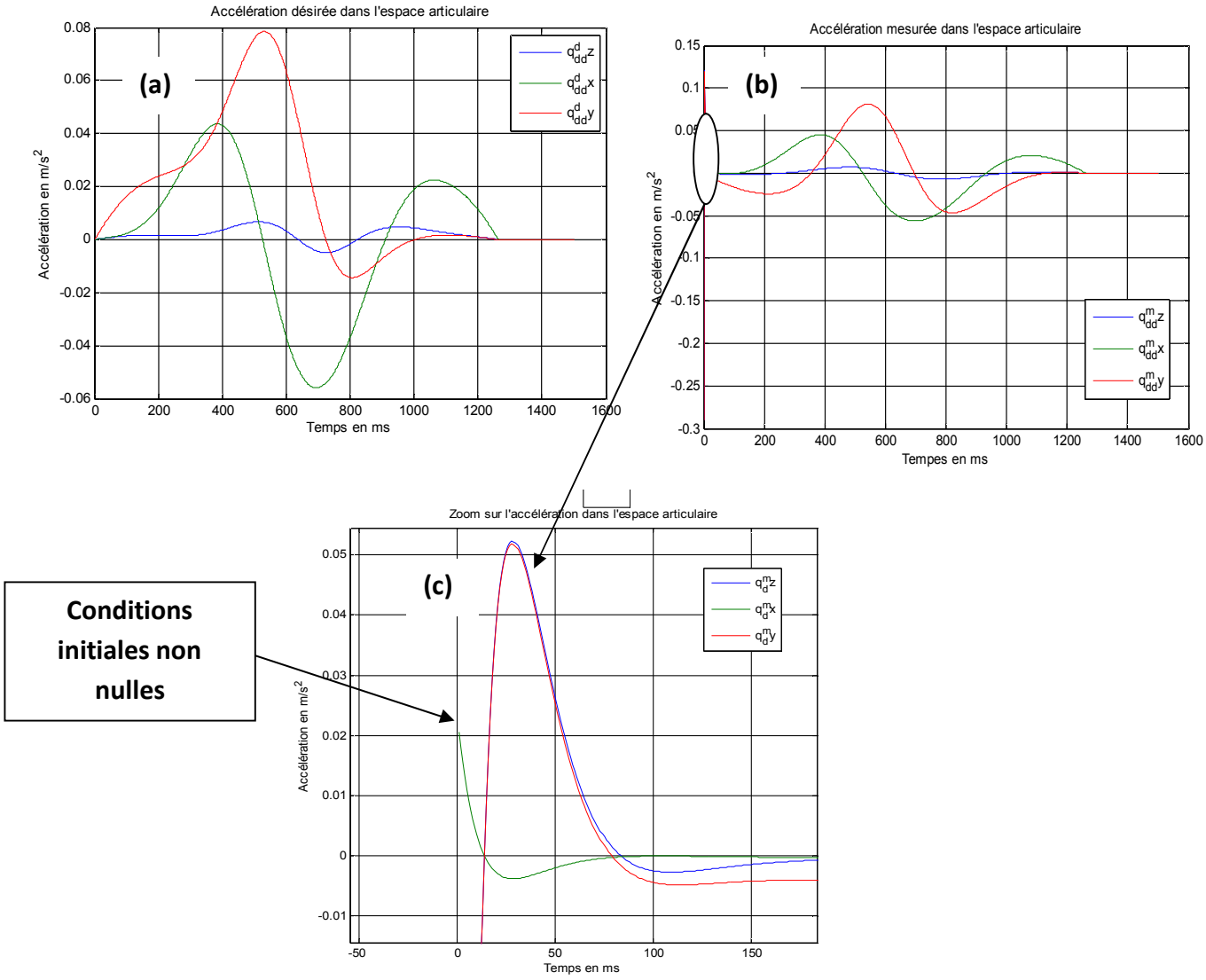


Figure 6.3 : ***Résultats des vitesses dans l'espace articulaire***

6.2.4 Accélération articulaire :

La figure (6.4 a) représente l'accélération calculée, désirée dans l'espace articulaire et la figure (6.b b) représente le résultat de la simulation c'est-à-dire les accélérations obtenues par simulation des sorties dans l'espace articulaire. Au démarrage du robot, une oscillation apparaît. (Figure 6.4 c)



Figures 6.4 : Résultats des accélérations dans l'espace articulaire

La figure 6.4c illustre que les conditions initiales sont non nulles. Les conditions initiales calculées sont à 16 chiffres significatifs (condition initiales de la position articulaire de l'axe x= 0.074361358401491) et lors de l'intégration de l'accélération simulée, le logiciel Simulink réduit à 6 chiffres cette même variable de condition initiale (x= 0.0744). Cela génère une différence entre la consigne et le retour de la simulation, d'où l'oscillation au démarrage du robot. Afin d'éviter ce comportement lié à la précision numérique, la solution a été de changer de nom à cette variable qui venait sans doute interférer avec une variable interne et propre au logiciel Simulink.

L’ensemble simulé donne des performances intéressantes.

Mais une hypothèse indispensable pour sa mise en œuvre est la connaissance parfaite des modèles. Ceux-ci ne peuvent être obtenus à partir de la conception.

Il est donc, nécessaire, comme pour la majorité des cas, de procéder à l’identification de l’Orthoglide à partir du comportement réel entrée-sortie. Ceci fait l’objet de la suite.

7 LA CALIBRATION :

Le chapitre précédent a montré la validité du modèle numérique, la réalité est tout autre chose (figure7.1). L'imprécision des pièces mécaniques et l'imperfection d'assemblage de l'Orthoglide sont des éléments à prendre en compte pour une mise en œuvre d'une commande dynamique robuste. Leur connaissance amène à identifier les paramètres géométrique et dynamique du robot. L'identification des paramètres géométriques est traitée dans ce chapitre.

Figure 7.1 : Orthoglide industrielle (et son concepteur)

La nécessité d'étalonner les robots est apparue avec le développement des langages de programmation permettant de spécifier la position de l'organe terminal, non plus en déplacement dans l'espace articulaire mais en déplacement dans l'espace opérationnel.

Ce type de programmation nécessite l'utilisation des modèles géométriques directes et inverses, fonction des paramètres géométriques du robot qui sont sensibles aux imprécisions de la réalisation.

L'objectif de l'étalonnage géométrique est d'identifier les valeurs exactes des paramètres qui interviennent dans le calcul des modèles géométriques du robot afin d'améliorer la précision statique.

Les imprécisions du robot ont deux origines :

- La connaissance imparfaite de la géométrie : la longueur des bras, le défaut de parallélisme et de perpendicularité des axes et enfin les offsets des codeurs moteurs.
- Certains phénomènes non géométriques sont difficilement modélisables : déformations dues aux variations de température ou à la gravité, jeux mécaniques, frottements, erreurs d'arrondi dans les calculs et même les erreurs de quantification des codeurs moteurs.

Nous nous intéressons dans ce chapitre à l'identification géométrique des offsets codeurs moteurs connue plus communément sous le nom de calibration.

7.1 Les offsets :

Les moteurs des axes de translations du robot doivent fournir une information de position, à savoir ici, la longueur des jambes. On utilise des codeurs incrémentaux pour fournir cette information dont leurs résolutions (erreurs de mesure) sont très bonnes de l'ordre du micron. La position de référence c'est-à-dire le zéro absolu est forcément différent du zéro de référence machine, cela amène à identifier les offsets à la position isotropique du robot.

Dans le chapitre 3, il a été établi un modèle cinématique de base du robot Orthoglide, il vient s'ajouter quatre paramètres géométriques modélisant les imperfections du robot à savoir :

$$(\Delta\rho_x, \Delta\rho_y, \Delta\rho_z, L); \tag{7.4}$$

Avec $\Delta\rho_i$ les offsets ($i \in \{x; y; z\}$) et $\boldsymbol{L}$ la longueur des jambes.

La longueur($\boldsymbol{L}$) des jambes est supposée connue et donc seul les offsets des codeurs sont donc à identifier.

7.1.1 Hypothèse de l'identification :

Les tolérances industrielles de $\pm 0,01mm$ de fabrication et d'assemblage des articulations et liaisons mécaniques sont atteintes facilement et l'identification se fait sous les hypothèses suivantes :

- Les liaisons et les articulations sont supposées parfaites et les codeurs sont supposés parfaits.
- Les manipulateurs sont de corps rigides reliés à des articulations parfaites sans jeu mécanique.
- Les trois jambes sont identiques.
- Les axes des actionneurs linéaires sont orthogonaux entre eux et recoupent vers un seul point (TCP) pour assurer un mouvement de translation à trois degrés de liberté de l'outil final.
- La position des codeurs est définie avec des erreurs (la position zéro codeur est différente de la position mécanique isotrope).

Par conséquent, une étude est développée [11] dans le paragraphe suivant sur des observations de mouvements parallèles spécifiques des jambes.

7.1.2 Analyse de l'influence des offsets :

Sous les hypothèses de modélisation ci-dessus et afin d'évaluer l'influence des offsets des codeurs sur le parallélisme des jambes par rapport au plan cartésien (***x, y, z***) et ainsi calibrer l'Orthoglide, il est proposé de dériver les relations différentielles (3.3 et 3.4) que l'on rappelle ici

$$p = (p_x, p_y, p_z) = (0,0,0) \text{ et } \rho = (\rho_x, \rho_y, \rho_z) = (L, L, L) \tag{3.3}$$

Avec la matrice d'identité déduite :

$$J(p_0, \rho_0) = I_{3x3} \tag{3.4}$$

Pour un déplacement du lieu du ***TCP*** ou le point ***P*** pour trois positions spécifiques de l'Orthoglide. La figure 7.2 indique un déplacement du point ***P*** de sa position isotropique à un minimum et à un maximum.

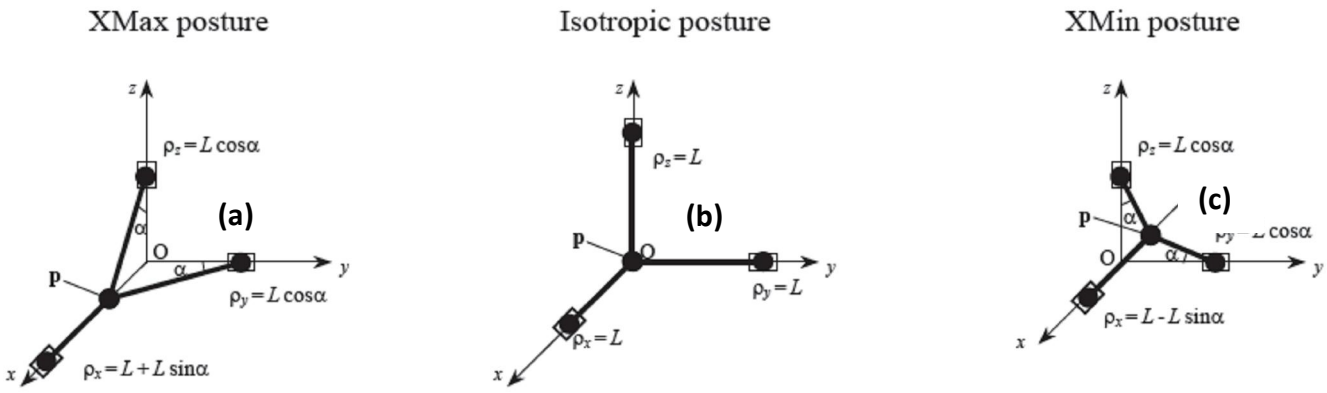


Figure 7.2 : Déplacement du lieu du point P

- En position isotropique (figure 7.1(b)), les relations sont calculées au point P.
- En position maximum (figure 7.1(a)), les relations sont calculées au point P :

$$p = (p_x, p_y, p_z) = (L\sin\boldsymbol{\alpha}, 0, 0)$$
$$\rho = (\rho_x, \rho_y, \rho_z) = (L + L\sin\boldsymbol{\alpha}, L\cos\boldsymbol{\alpha}, L\cos\boldsymbol{\alpha})$$

Où $\boldsymbol{\alpha}$ est l'angle entre les jambes ***y-z*** et les axes correspondants ***(x, y)*** (figure 7.1(a) ou (c)) qui après substitution de (3.2) que l'on rappelle ici :

$$J^{-1}(p,\rho) = \frac{\partial \rho}{\partial p} = \begin{pmatrix} 1 & \frac{p_y}{p_x - \rho_x} & \frac{p_z}{p_x - \rho_x} \\ \frac{p_x}{p_x - \rho_y} & 1 & \frac{p_z}{p_y - \rho_y} \\ \frac{p_x}{p_z - \rho_z} & \frac{p_y}{p_z - \rho_z} & 1 \end{pmatrix} \qquad (3.2)$$

donne la matrice triangulaire Jacobienne basse suivante :

$$J(p(\boldsymbol{\alpha}), \rho(\boldsymbol{\alpha})) = \begin{pmatrix} 1 & 0 & 0 \\ T_{\alpha} & 1 & 0 \\ T_{\alpha} & T_{\alpha} & 1 \end{pmatrix} \qquad (7.5)$$

Avec $T_{\alpha} = \tan\boldsymbol{\alpha}$

- En position minimum (figure 7.1(c)), les calculs sont similaires à la position maximum.

7.1.3 Équation de base :

Des équations cinématique du chapitre 3 (3.1) que l'on rappelle ici :

$$\begin{cases} (p_x - \rho_x)^2 + p_y^2 + p_z^2 = L^2; \\ p_x^2 + (p_y - \rho_y)^2 + p_z^2 = L^2; \\ p_x^2 + p_y^2 + (p_z - \rho_z)^2 = L^2; \end{cases} \quad (3.1)$$

Les offsets des codeurs ($\Delta\rho_x, \Delta\rho_y, \Delta\rho_z$) sont rajoutés et donnent les équations cinématiques de bases suivantes :

$$\begin{aligned} &[p_x - (\rho_x + \Delta\rho_x)]^2 + p_y^2 + p_y^2 = L^2; \\ &p_x^2 + [p_y - (\rho_y + \Delta\rho_y)]^2 + p_z^2 = L^2; \\ &p_x^2 + p_y^2 + [p_z - (\rho_z + \Delta\rho_z)]^2 = L^2; \end{aligned} \quad (7.6)$$

7.2 Méthodologie de calibration :

L'une des techniques [11] consiste à mesurer le parallélisme des jambes par rapport à une surface de base plane. Le ***TCP*** est déplacé le long des axes cartésiens aux positions décrites dans le paragraphe précédent et pour ces déplacements, les jambes sont strictement parallèles au plan cartésien correspondant. Un écart de parallélisme est une donnée à l'identification des offsets. Par exemple, une trajectoire en ligne droite dans l'espace cartésien, les trois axes se déplace pour tracer cette ligne droite, le coté de la jambe ***x*** doit rester strictement parallèle à l'axe ***x,*** et la différence de mesure (Δz) de la jambe par rapport à surface plane (***x, y***) est une donnée de calibration (figure 7.3).

7.2.1 Technique de la mesure :

Un comparateur mécanique simple avec une embase magnétique est utilisé, sa résolution est de ***10 µm.*** Sa position est fixe et mesure deux positions distinctes aux extrémités de la jambe.

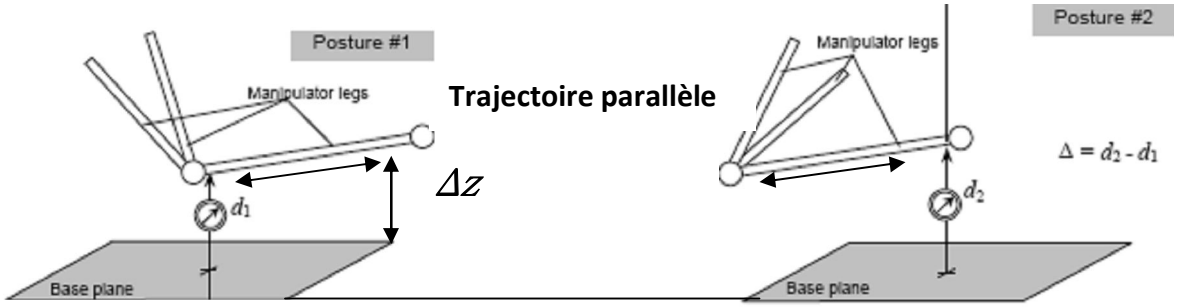


Figure 7.3 : Technique de mesure du parallélisme des jambes

L'expérimentation se déroule comme suit :

- Approcher le manipulateur à la position mécanique isotropique et mesurer le parallélisme de l'axe ***x***.
- Approcher le manipulateur dans une position mécanique ***x-maximum et x-minimum*** et mesurer le parallélisme de l'axe ***x***.
- Effectuer les mêmes opérations pour les deux autres axes (***y, z***)

En utilisant la matrice Jacobienne (3.2) et les équations cinématique de base (7.6), cela donne un système de 12 équations à trois inconnues (3 offsets). Une version réduite du système est obtenue en évaluant la différence entre les positions maximums et minimums du parallélisme de la jambe par rapport à la surface de base plane. Cela amène à un système de six équations linéaires à trois inconnues suivant :

$$\begin{pmatrix} b_1 & c_1 & 0 \\ c_1 & b_1 & 0 \\ b_2 & c_2 & 0 \\ c_2 & b_2 & 0 \\ \hline 0 & b_1 & c_1 \\ 0 & c_1 & b_1 \\ 0 & b_2 & c_2 \\ 0 & c_2 & b_2 \\ \hline b_1 & 0 & c_1 \\ c_1 & 0 & c_1 \\ b_2 & 0 & c_2 \\ c_2 & 0 & b_2 \end{pmatrix} \begin{pmatrix} \Delta\rho_x \\ \Delta\rho_y \\ \Delta\rho_z \end{pmatrix} = \begin{pmatrix} \Delta x_y^+ \\ \Delta y_x^+ \\ \Delta x_y^- \\ \Delta y_x^- \\ \hline \Delta y_z^+ \\ \Delta z_y^+ \\ \Delta y_z^- \\ \Delta z_y^- \\ \hline \Delta x_z^+ \\ \Delta z_x^+ \\ \Delta x_z^- \\ \Delta z_x^- \end{pmatrix} \Rightarrow \begin{pmatrix} b & c & 0 \\ c & b & 0 \\ \hline 0 & b & c \\ 0 & c & b \\ \hline b & 0 & c \\ c & 0 & b \end{pmatrix} \begin{pmatrix} \Delta\rho_x \\ \Delta\rho_y \\ \Delta\rho_z \end{pmatrix} = \begin{pmatrix} \Delta x_y \\ \Delta y_x \\ \hline \Delta y_z \\ \Delta z_y \\ \hline \Delta x_z \\ \Delta z_x \end{pmatrix} \qquad (7.7)$$

Avec

$$\begin{cases} b = b_1 - b_2 \\ c = c_1 - c_2 \\ b_1 = \sin \boldsymbol{\alpha}_1 \\ c_1 = (0,5 + \sin \boldsymbol{\alpha}_1) \tan \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_1 = a \sin(\frac{\boldsymbol{\rho}_{\max}}{L}) \end{cases} \quad (7.8)$$

La solution est alors obtenue par moindre carrés.

7.3 Résultats et commentaires

Pour chaque jambe, la mesure a été répétée deux fois, c'est-à-dire deux aller retour entre les extrémités pour une répététabilité à +/- 0,02mm.

La première expérimentation montre un maximum d'écart de parallélisme sur la jambe y de 3,67mm. De ces valeurs mesurées sur chaque axe, les trois offsets sont calculés avec Matlab (annexe V) et ensuite implémentés dans le modèle dynamique de l'Orthoglide. Il en résulte un écart maximum de la deuxième expérimentation sur l'axe y de 0,14 mm soit une diminution par 26,2. Pour l'axe x le rapport est de 3,91 et l'axe z de 6,34 (voir tableau 7.4).

Expérimentation initiale en millimètres				
Axes	Position minimum	Position maximum	Différence	Offset
X	- 114	+ 113	227	+ 3610
Y	+ 156	- 211	367	+ 3736
Z	+ 135	- 182	317	+ 3879
Expérimentation après calibration				
Axes	Position minimum	Position maximum	Différence	Offset
X	- 13	+ 45	58	- 286
Y	+10	- 4	14	- 1134
Z	+ 8	- 42	50	+ 1799

Tableau 7.4 : Données expérimentales de la calibration

Une troisième expérimentation n'a pas permis d'obtenir de meilleurs résultats. Il est donc nécessaire d'identifier d'autres paramètres géométriques tel que l'orthogonalité de montage mécanique des trois axes. En effet, nous n'avons pas de données expérimentales

de cette orthogonalité, le constructeur du robot n'a fourni aucune information sur ce point. Par conséquent, pour une raison de temps, l'identification géométrique s'est résumée à l'identification des offsets qui génèrent les meilleurs résultats.

8 CONCLUSION :

Ce mémoire porte sur l'étude des machines à structure parallèle utilisées pour tâches rapides. Il montre que cela passe par la définition d'une stratégie de commande adaptée. La commande dynamique est la solution mais elle nécessite la connaissance parfaite du modèle. Ce dernier ne peut être issu de la conception et il faut donc passer obligatoirement par une identification. Nous nous sommes limités à l'identification des paramètres géométriques mais celle-ci doit être prolongée par l'identification des paramètres dynamiques.

L'identification dynamique est une étape importante dans l'amélioration de la précision de la loi de commande dynamique. Elle doit être réalisée dans l'espace articulaire avec une mesure adaptée. Il serait donc intéressant de mettre en œuvre une identification des paramètres du modèle dynamique. La phase d'identification doit alors permettre d'extraire les paramètres dynamiques et géométriques, ces travaux peuvent donc amener à entrevoir une nouvelle façon d'identifier la dynamique d'un robot parallèle.

La commande dynamique en ligne nécessite de nombreux calculs en temps réel du modèle dynamique inverse. Cela nécessite un calculateur très puissant qui n'a pas été testé. Une architecture rigoureuse et une optimisation des temps de calculs du modèle dynamique inverse sont nécessaires à l'atteinte des performances de la robotique parallèle.

****

Références :

[1] Gough V. E., « Contribution to discussion of papers on research in automobile stability, control and tyre performance », Proceedings Auto Div. Inst. Mech. Eng, 1956-1957.

[2] Clavel R., « DELTA, a fast robot with parallel geometry », Proceedings of the 18th International Symposium of Robotic Manipulators, IFR Publication, pp. 91-100, 1988.

[3] Krut, S., 2003, Contribution à l'etude des robots parallèles l''egers, 3T-1R et 3T-2R, à forts d'ébattements angulaires, Ph.D. Thesis, University of Montpellier-II, France.

[4] Majou Félix Analyse cinétostatique des machines parallèle à translations, pp 15-16

[5] Pashkevich A, Kinematic calibration of Orthoglide-type mechanisms from observation of parallel leg motions, pp 4-7

[6] Khalil, W., Dombre, E., "Modélisation, identification et commande des robots des robots", *pp 337, 1999*

[7] Khalil, W., Dombre, E., "Modélisation, identification et commande des robots des robots", *pp 338, 1999*

[8] Guegan S, "Contribution à la modélisation et l'identification dynamique des robots parallèle" pp 120, 2003

[9] Khalil, W., Dombre, E., "Modélisation, identification et commande des robots des robots", *pp 370, 1999*

[10] Khalil, W., Dombre, E., "Modélisation, identification et commande des robots des robots", *pp 371, 1999*

[11] Pashkevich A, Chablat D. et Wenger P., "Kinematic calibration of Orthoglide-type mechanisms from observation of parallel leg motions"

[12] Sylvain GUEGAN, Wisama KHALIL, Damien CHABLAT, Philippe Wenger, « Modélisation Dynamique d'un Robot Parallèle à 3 DDL : l'Orthoglide », IRCCyN, 2002

[13] O. Khatib, "A unified approach for motion and force control of robot manipulators: the operational space formulation", *IEEE J. of Robotics and Automation*, *Vol. RA-3(1), pp. 43-53, février 1987.*

[14] K.W. LILLY et D.E. ORIN, "Efficient O(N) computation of the operational space inertia matrix", Proceedings IEEE International Conference on Robotics and Automation, Cincinnati, US, pp. 1014-1019, 1990.

Annexe I : Modèle dynamique inverse ([12])

Le modèle dynamique inverse du robot donne les couples et ou des forces appliquées au robot en fonction des positions, vitesses et accélérations articulaires. Dans le cadre du modèle lagrangien, il est représenté par une relation de la forme :

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

avec:

- Γ : vecteur des couples/forces des actionneurs, selon que l'articulation est rotoïde ou prismatique
- $\mathbf{q}$: positions articulaires
- $\dot{\mathbf{q}}$: vitesses articulaires
- $\ddot{\mathbf{q}}$: accélérations articulaires
- $\mathbf{f_e}$: effort extérieur (forces et moments) qu'exerce le robot sur l'environnement

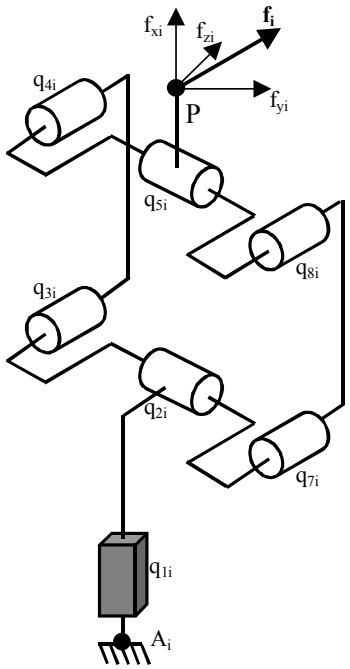


Figure I.1 : Force fi

On représente les réactions de la plate-forme sur la chaîne cinématique i par $\mathbf{f_i}$; les inconnues du modèle dynamique inverse sont alors les 9 composantes des forces

$f_i = \left[f_{xi} \quad f_{yi} \quad f_{zi} \right]^T$ et les 3 forces des articulations indépendantes Γ_{1i}. Le modèle dynamique inverse de chaque chaîne cinématique est composé de 3 équations indépendantes, ce qui donne un total de 9 équations. De plus, les équations de Newton-Euler de la plate forme fournissent 3 équations supplémentaires : on a donc à résoudre un système de 12 équations à 12 inconnues.

Calcul du modèle dynamique inverse de la chaîne cinématique i

Pour chaque chaîne cinématique du robot, on calcule le modèle dynamique de la structure arborescente équivalente obtenue en coupant la boucle planaire de type parallélogramme au niveau de l'articulation passive $\mathbf{q_{8i}}$ (figure I.2).

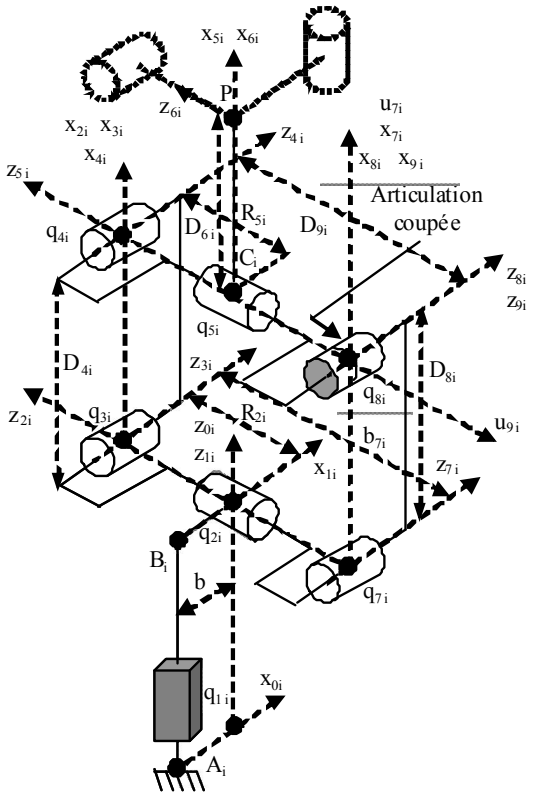


Figure I.2: Placement des repéres dans la chaine cinématique

Ce modèle dynamique s'écrit :

$$\Gamma_{ari} = H_{ari}(q_i, \dot{q}_i, \ddot{q}_i) \qquad \text{(I.2)}$$

avec :

$$\Gamma_{ari} = \begin{bmatrix} \Gamma_{1i} & \Gamma_{2i} & \Gamma_{3i} & \Gamma_{4i} & \Gamma_{5i} & \Gamma_{7i} \end{bmatrix}^T \qquad \text{(I.3)}$$

Le modèle dynamique de la boucle fermée s'obtient à partir de Γ_{ari} et de la matrice G_i.

$$G_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 \end{bmatrix}^T \qquad \text{(I.4)}$$

Dans le modèle complet du robot, on ne considère que les couples des variables articulaires q_{1i}, q_{2i} et q_{3i} (c'est à dire des variables indépendantes de la chaîne cinématique i). Le modèle dynamique inverse de la boucle fermée s'écrit alors :

$$\Gamma_i = \begin{bmatrix} \Gamma_{1i} & \Gamma_{2i} & \Gamma_{3i} \end{bmatrix}^T = G_i^T \Gamma_{ari} \qquad \text{(I.5)}$$

ou encore :

$$\Gamma_i = G_i^T H_{ari}(q_i, \dot{q}_i, \ddot{q}_i) = H_i(q_i, \dot{q}_i, \ddot{q}_i) \qquad \text{(I.6)}$$

où $H_i(q_i, \dot{q}_i, \ddot{q}_i)$ est le modèle dynamique de la chaîne cinématique i.

Lorsque l'on prend en compte la force de réaction f_i sur le point terminal de chaque chaîne cinématique, la forme générale du modèle dynamique de la chaîne cinématique i devient :

$$\Gamma_i = H_i(q_i, \dot{q}_i, \ddot{q}_i) + {}^0J_i^{T\,0} f_i \qquad \text{(I.7)}$$

Calcul de ${}^{0}\mathbf{f}_{i}$ en utilisant le modèle dynamique de la chaîne cinématique i

Pour chaque chaîne cinématique, seule l'articulation $\mathbf{q}_{1i}$ est motorisée.

On a donc en fait:

$$\Gamma_i = \begin{bmatrix} \Gamma_{1i} & 0 & 0 \end{bmatrix}^T \tag{I.8}$$

A partir de l'équation (I.7), on peut écrire :

$${}^{0}f_i = {}^{0}J_i^{-T}\left(\Gamma_i - H_i(q_i, \dot{q}_i, \ddot{q}_i)\right) = -{}^{0}J_i^{-T} H_i(q_i, \dot{q}_i, \ddot{q}_i) + {}^{0}J_i^{-T}\Gamma_i \tag{I.9}$$

Or

$$H_{xi}(q_i, \dot{q}_i, \ddot{q}_i) = {}^{0}J_i^{-T} H_i(q_i, \dot{q}_i, \ddot{q}_i) \tag{I.10}$$

avec $H_{xi}(q_i, \dot{q}_i, \ddot{q}_i)$ l'expression du vecteur $H_i(q_i, \dot{q}_i, \ddot{q}_i)$ dans l'espace cartésien de position au point $\mathbf{P}_i$ [13, 14].

Les expressions des matrices jacobiennes inverses ${}^{0}\mathbf{J}_i^{-T}$ ont été déterminées précédemment dans le modèle cinématique inverse rappellé ici :

$${}^{0}J_1^{-1} = \begin{bmatrix} -\dfrac{1}{T_{21}} & \dfrac{T_{31}}{S_{21}} & 1 \\ -\dfrac{1}{D_{41}C_{31}S_{21}} & \dfrac{T_{31}}{D_{41}C_{31}T_{21}} & 0 \\ 0 & -\dfrac{1}{D_{41}C_{31}} & 0 \end{bmatrix}$$

$${}^{0}J_2^{-1} = \begin{bmatrix} 1 & -\dfrac{1}{T_{22}} & \dfrac{T_{32}}{S_{22}} \\ 0 & -\dfrac{1}{D_{42}C_{32}S_{22}} & \dfrac{T_{32}}{D_{42}C_{32}T_{22}} \\ 0 & 0 & -\dfrac{1}{D_{42}C_{32}} \end{bmatrix} \tag{I.11}$$

$$ {}^{0}J_{3}^{-1} = \begin{bmatrix} \frac{T_{33}}{S_{23}} & 1 & -\frac{1}{T_{23}} \\ \frac{T_{33}}{D_{43}C_{33}T_{23}} & 0 & -\frac{1}{D_{43}C_{33}S_{23}} \\ -\frac{1}{D_{43}C_{33}} & 0 & 0 \end{bmatrix} $$

On obtient pour la chaîne cinématique 1:

$$ {}^{0}f_{1} = -H_{x1}(q_1, \dot{q}_1, \ddot{q}_1) + \begin{bmatrix} -\frac{1}{T_{21}} \\ \frac{T_{31}}{S_{21}} \\ 1 \end{bmatrix} \Gamma_{11} \qquad \text{(I.11)} $$

On remarque que le coefficient de Γ_{11} correspond à la première colonne de la transposée de la matrice jacobienne inverse du robot (I.12) $\mathbf{{}^{0}J_{p}^{-T}}$.

$$ {}^{0}J_{p}^{-1} = \begin{bmatrix} -\frac{1}{T_{21}} & \frac{T_{31}}{S_{21}} & 1 \\ 1 & -\frac{1}{T_{22}} & \frac{T_{32}}{S_{22}} \\ \frac{T_{33}}{S_{23}} & 1 & -\frac{1}{T_{23}} \end{bmatrix} \qquad \text{(I.12)} $$

De même, pour les chaînes cinématiques 2 et 3, les coefficients Γ_{12} et Γ_{13} ont pour coefficients respectifs la deuxième et la troisième colonne de $\mathbf{{}^{0}J_{p}^{-T}}$:

$$ {}^{0}f_{i} = -H_{xi}(q_i, \dot{q}_i, \ddot{q}_i) + {}^{0}J_{p\,[:,i]}^{-T}\Gamma_{1i} \qquad \text{(I.13)} $$

où $\mathbf{{}^{0}J_{p\,[:,i]}^{-T}}$ est la i^{ème} colonne de la transposée de la matrice jacobienne inverse du robot.

Modélisation dynamique de la plate-forme

On applique les équations d'Euler à l'origine de la plate forme :

$$ {}^{0}F_p = {}^{0}\dot{V}_p \, M_p - M_p \, {}^{0}g \qquad \text{(I.14)} $$

avec $ {}^{0}\mathbf{F_p} $ la force totale extérieure appliquée sur la plate-forme au point P. $ {}^{0}\dot{\mathbf{V}}_{\mathbf{p}} $ étant connue, on peut déterminer $ {}^{0}\mathbf{F_p} $.

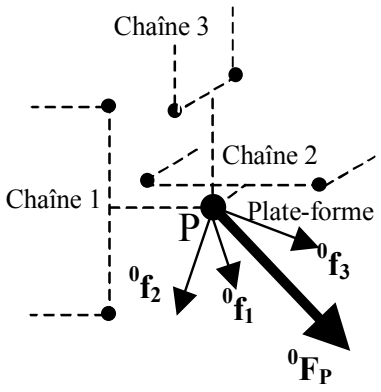


Figure I.2 : Forces extérieures appliquées au point P

Les forces appliquées à la plate-forme dues aux forces de réaction de toutes les chaînes cinématiques sont alors calculées de la manière suivante :

$$ {}^{0}F_p = \sum_{i=1}^{3} {}^{0}f_i \qquad \text{(I.15)} $$

soit d'après l'équation (I.13) :

$$ {}^{0}F_p = \sum_{i=1}^{3} \left(- H_{xi}(q_i, \dot{q}_i, \ddot{q}_i) + {}^{0}J_{p\,[:,i]}^{-T} \Gamma_{1i} \right) \qquad \text{(I.16)} $$

soit encore :

$$ {}^{0}J_p^{-T} \Gamma = {}^{0}F_p + \sum_{i=1}^{3} H_{xi}(q_i, \dot{q}_i, \ddot{q}_i) \qquad \text{(I.17)} $$

On en déduit la forme du modèle dynamique inverse du robot Orthoglide :

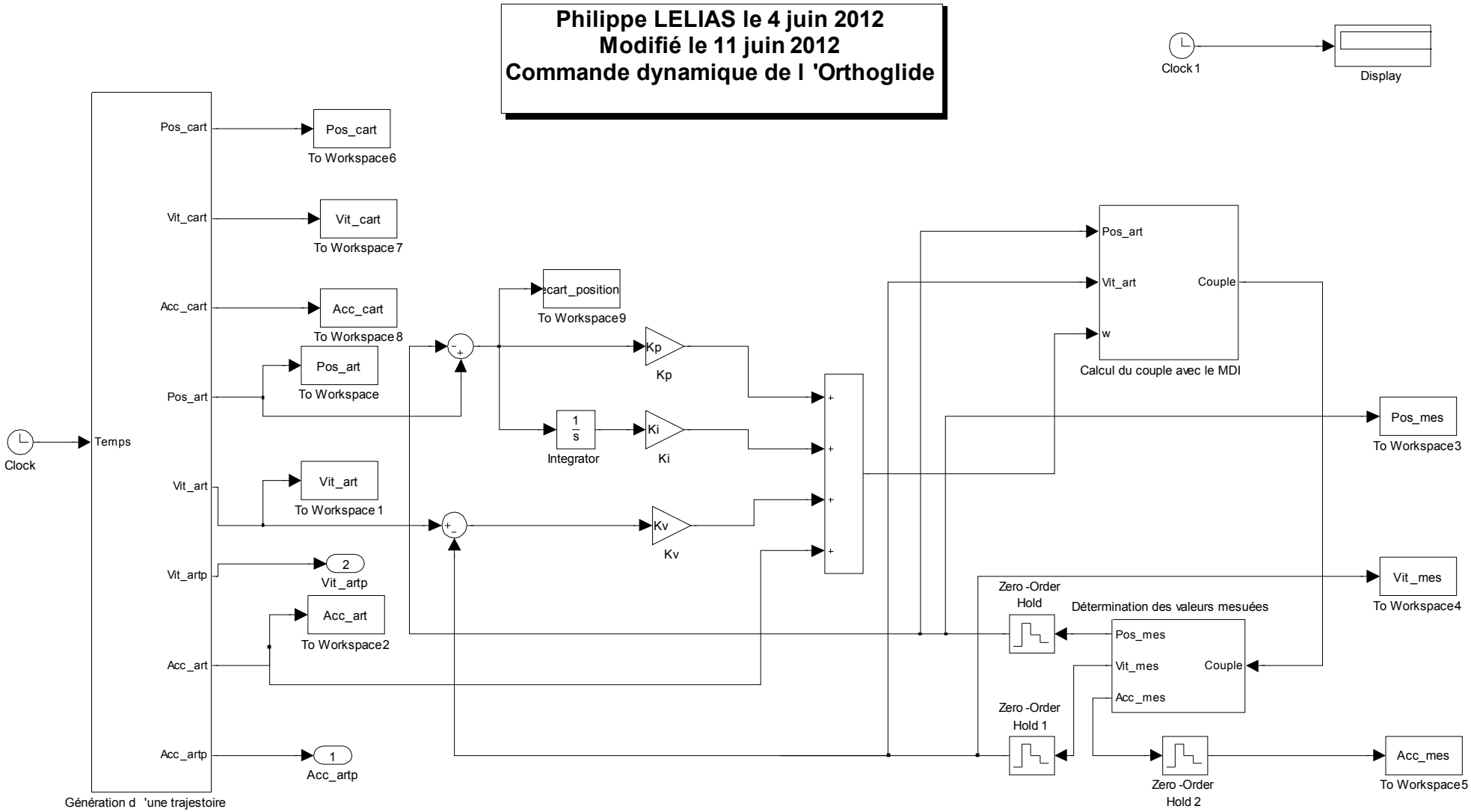
$$ \Gamma = {}^{0}J_p^{T} F_{robot} \qquad \text{(I.18)} $$

avec :

$$\Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \Gamma_{13} \end{bmatrix}^{T} \tag{I.19}$$

et :

$$F_{robot} = {}^{0}F_{p} + \sum_{i=1}^{3} H_{xi}(q_i, \dot{q}_i, \ddot{q}_i) \tag{I.20}$$


Philippe LELIAS le 4 juin 2012
Modifié le 11 juin 2012
Commande dynamique de l 'Orthoglide
Clock 1
Display
Pos_cart
Pos_cart
To Workspace6
Vit_cart
Vit_cart
To Workspace7
Acc_cart
Acc_cart
To Workspace8
Pos_art
Pos_art
To Workspace
Vit_art
Vit_art
To Workspace1
Vit_artp
2
Vit_artp
Acc_art
Acc_art
To Workspace2
Acc_art
Acc_artp
1
Acc_artp
Temps
Clock
Génération d 'une trajestoire
cart_position
To Workspace9
Kp
Kp
1/s
Integrator
Ki
Ki
Kv
Kv
Pos_art
Vit_art
Couple
w
Calcul du couple avec le MDI
Pos_mes
To Workspace3
Vit_mes
To Workspace4
Zero -Order Hold
Détermination des valeurs mesuées
Pos_mes
Vit_mes
Acc_mes
Couple
Zero -Order Hold 1
Zero -Order Hold 2
Acc_mes
To Workspace5

Annexe III : Génération d'une trajectoire en MATLAN FCN

```
function [Cart]=sim_generation_trajectoire(temps)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         Simulation de la commande dynamique du robot Orthoglide                 %
%              Géneration de trajectoire en fonction du temps                     %
%                          Par Philippe LELIAS                                    %
%                             Le 25 Mai 2012                                      %
%                         Modifier le 11 juin 2012
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  MENU PRINCIPAL                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


global  alpha_0 alpha_f OI Qp Qm dx dy dz lbroche Traj tool Kp Kv Ki
temps_final amplitude
t=temps;


%%%%%%%%%%%%%%%%%%%%%%%%%%%  Trajectoire    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch Traj
    case 1
        %  resolution
        res=30;
        %  Rayon de la Trajectoire en métres [m]
        R_Trj=150/1000;
        % Angle entre l'axe X et le plan de la trajectoire
        phi=0*pi/180;

        %  Angle entre v et l'axe Z
        gamma=45*pi/180;

        if t<temps_final
            var=t/temps_final;
            var2=var*var;
            var3=var2*var;
            var4=var3*var;
            var5=var4*var;
            pos_alpha=((10.0*var3)-
(15.0*var4)+(6.0*var5))*amplitude+alpha_0 ;
            vit_alpha=((30.0*var2)-
(60.0*var3)+(30.0*var4))/temps_final*amplitude;
            acc_alpha=((60.0*var)-
(180.0*var2)+(120.0*var3))/(temps_final*temps_final)*amplitude;

            Px=dx+(R_Trj-lbroche*sin(gamma))*cos(pos_alpha);
            Py=dy+(R_Trj-lbroche*sin(gamma))*sin(pos_alpha);
            Pz=dz+(lbroche*cos(gamma));

            %  Vitesse du Point p
            ppx=-(R_Trj-lbroche*sin(gamma))*vit_alpha*sin(pos_alpha) ;
            ppy=(R_Trj-lbroche*sin(gamma))*vit_alpha*cos(pos_alpha);
            ppz=-lbroche*sin(gamma);

            %  Acceleration de Point p
```

```
            pppx=-(R_Trj-
lbroche*sin(gamma))*(vit_alpha^2*cos(pos_alpha)+acc_alpha*sin(pos_alpha));
            pppy=-(R_Trj-
lbroche*sin(gamma))*(vit_alpha^2*sin(pos_alpha)+acc_alpha*cos(pos_alpha));
            pppz=-lbroche*cos(gamma);

        else
            Px=dx+(R_Trj-lbroche*sin(gamma))*cos(alpha_f);
            Py=dy+(R_Trj-lbroche*sin(gamma))*sin(alpha_f);
            Pz=dz+(lbroche*cos(gamma));
            ppx=0;
            ppy=0;
            ppz=-lbroche*sin(gamma);
            pppx=0;
            pppy=0;
            pppz=-lbroche*cos(gamma);

            %%%%%%%%%%%%%%%%%%%%%%%
            pos_alpha=alpha_f;
            vit_alpha=0;
            acc_alpha=0;
            %%%%%%%%%%%%%%%%%%%%%%%
        end
    case 2
    case 3
end
Pos_cart=[Px;Py;Pz];
Vit_cart=[ppx;ppy;ppz];
Acc_cart=[pppx;pppy;pppz];
%Cart=[pos_alpha*ones(3,1); vit_alpha*ones(3,1); acc_alpha*ones(3,1)];
Cart=[Pos_cart; Vit_cart; Acc_cart];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Fin du menu principal                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Annexe IV : Génération d'une trajectoire en S FUNCTION

```
/*************************************************************************
 *        Simulation de la commande dynamique du robot Orthoglide        *
 *             Géneration de trajectoire en fonction du temps            *
 *                        Par Philippe LELIAS                            *
 *                           Le 25 Mai 2012                              *
 *                       Modifier le 19 juillet 2012                     *

*************************************************************************
 *                             MENU PRINCIPAL                            *

*************************************************************************/

 /* Spécification de la S-FUNCTION */


#define S_FUNCTION_NAME  sim_generation_trajectoire_c
#define S_FUNCTION_LEVEL 2


 /* Need to include simstruc.h for the definition of the SimStruct and
    its associated macro definitions.*/

#include <math.h>
#include "simstruc.h"
#include "mex.h"

/* déclaration des paramétres*/

#define  ki                  mxGetPr(ssGetSFcnParam( S, 0))
#define  kp                  mxGetPr(ssGetSFcnParam( S, 1))
#define  kv                  mxGetPr(ssGetSFcnParam( S, 2))
#define  OI                  mxGetPr(ssGetSFcnParam( S, 3))
#define  Qm                  mxGetPr(ssGetSFcnParam( S, 4))
#define  Qp                  mxGetPr(ssGetSFcnParam( S, 5))
#define  Traj                mxGetPr(ssGetSFcnParam( S, 6))
#define  alpha_0             mxGetPr(ssGetSFcnParam( S, 7))
#define  alpha_f             mxGetPr(ssGetSFcnParam( S, 8))
#define  amplitude           mxGetPr(ssGetSFcnParam( S, 9))
#define  dx                  mxGetPr(ssGetSFcnParam( S, 10))
#define  dy                  mxGetPr(ssGetSFcnParam( S, 11))
#define  dz                  mxGetPr(ssGetSFcnParam( S, 12))
#define  lbroche             mxGetPr(ssGetSFcnParam( S, 13))
#define  q0                  mxGetPr(ssGetSFcnParam( S, 14))
#define  temps_final         mxGetPr(ssGetSFcnParam( S, 15))
#define  tool                mxGetPr(ssGetSFcnParam( S, 16))
#define  wn                  mxGetPr(ssGetSFcnParam( S, 17))

#define                      NUM_DISC_STATES           9
#define                      NUM_CONT_STATES           0
```

```
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 18);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    ssSetNumContStates(S, NUM_CONT_STATES);
    ssSetNumDiscStates(S, NUM_DISC_STATES);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 2);
    /*
     * Set direct feedthrough flag (1=yes, 0=no).
     * A port has direct feedthrough if the input is used in either
     * the mdlOutputs or mdlGetTimeOfNextVarHit functions.
     * See matlabroot/simulink/src/sfuntmpl_directfeed.txt.
     */
    ssSetInputPortDirectFeedThrough(S, 0, 0);

    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 9);

    ssSetNumSampleTimes(S, 1);

    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, 0);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);

}

#define MDL_INITIALIZE_CONDITIONS   /* Change to #undef to remove function
*/
#if defined(MDL_INITIALIZE_CONDITIONS)
  /* Function: mdlInitializeConditions
========================================
   * Abstract:
   *    In this function, you should initialize the continuous and discrete
   *    states for your S-function block.  The initial states are placed
   *    in the state vector, ssGetContStates(S) or ssGetRealDiscStates(S).
   *    You can also perform any other initialization activities that your
   *    S-function may require. Note, this routine will be called at the
   *    start of simulation and if it is present in an enabled subsystem
   *    configured to reset states, it will be call when the enabled
subsystem
   *    restarts execution to reset the states.
   */
static void mdlInitializeConditions(SimStruct *S)
```

```
{
}
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  /* Function: mdlStart
=====================================================
   * Abstract:
   *    This function is called once at start of model execution. If you
   *    have states that should be initialized once, this is the place
   *    to do it.
   */
static void mdlStart(SimStruct *S)
{

}
#endif /*  MDL_START */


static void sim_generation_trajectoire_c(SimStruct *S,const real_T t)

{

    real_T      var,var2,var3,var4,var5;
    real_T      res, R_Trj, phi, pi, gamma;
    real_T      pos_alpha, vit_alpha, acc_alpha;
    real_T      Px, Py, Pz, ppx, ppy, ppz, pppx, pppy, pppz;
    real_T *state = ssGetRealDiscStates(S);


  /*************************  Trajectoire
*********************************/

    pi=3.14116;

    // resolution
    res=30;
    //Rayon de la Trajectoire en métres [m]
    R_Trj=150/1000;
    //Angle entre l'axe X et le plan de la trajectoire
    phi=0*pi/180;
     mexPrintf("%f\n=",R_Trj);
    //Angle entre v et l'axe Z
    gamma=45*pi/180;

    if (t<*temps_final)
    {
        var=(t)/(*temps_final);
        var2=var*var;
        var3=var2*var;
        var4=var3*var;
        var5=var4*var;
        pos_alpha=((10.0*var3)-
(15.0*var4)+(6.0*var5))*(*amplitude)+*alpha_0;
        vit_alpha=((30.0*var2)-
(60.0*var3)+(30.0*var4))/(*temps_final)*(*amplitude);
```

```
        acc_alpha=((60.0*var)-
(180.0*var2)+(120.0*var3))/(*temps_final*(*temps_final))*(*amplitude);

        Px=*dx+(R_Trj-*lbroche*sin(gamma))*cos(pos_alpha);
        Py=*dy+(R_Trj-*lbroche*sin(gamma))*sin(pos_alpha);
        Pz=*dz+(*lbroche*cos(gamma));

        //Vitesse du Point p
        ppx=-(R_Trj-*lbroche*sin(gamma))*vit_alpha*sin(pos_alpha) ;
        ppy=(R_Trj-*lbroche*sin(gamma))*vit_alpha*cos(pos_alpha);
        ppz=-*lbroche*sin(gamma);

        //Acceleration de Point p
        pppx=-(R_Trj-
*lbroche*sin(gamma))*(vit_alpha*vit_alpha*cos(pos_alpha)+acc_alpha*sin(pos_
alpha));
        pppy=-(R_Trj-
*lbroche*sin(gamma))*(vit_alpha*vit_alpha*sin(pos_alpha)+acc_alpha*cos(pos_
alpha));
        pppz=-*lbroche*cos(gamma);

        state[0]=Px;
        state[1]=Py;
        state[2]=Pz;
        state[3]=ppx;
        state[4]=ppy;
        state[5]=ppz;
        state[6]=pppx;
        state[7]=pppy;
        state[8]=pppz;
    }
    else
    {
       mexPrintf("%f\n=", "Ok");
        Px=*dx+(R_Trj-*lbroche*sin(gamma))*cos(*alpha_f);
        Py=*dy+(R_Trj-*lbroche*sin(gamma))*sin(*alpha_f);
        Pz=*dz+(*lbroche*cos(gamma));
        ppx=0;
        ppy=0;
        ppz=-*lbroche*sin(gamma);
        pppx=0;
        pppy=0;
        pppz=-*lbroche*cos(gamma);
    }
}
/*************************************************************************
 *                       Fin du menu principal                           *
 *
/************************************************************************/


/* Function: mdlOutputs
=======================================================
 * Abstract:
 *    In this function, you compute the outputs of your S-function
 *    block. Generally outputs are placed in the output vector(s),
 *    ssGetOutputPortSignal.
 *
 *    Qcourant=[q,dq,ddq]; */
```

```
static void mdlOutputs(SimStruct *S, int_T tid)
{

    real_T      *Cart= ssGetOutputPortRealSignal(S,0);
    const real_T t      = ssGetT(S);
    const real_T *state = ssGetRealDiscStates(S);
    sim_generation_trajectoire_c(S, t);
    Cart[0]=state[0];
    Cart[1]=state[1];
    Cart[2]=state[2];
    Cart[3]=state[3];
    Cart[4]=state[4];
    Cart[5]=state[5];
    Cart[6]=state[6];
    Cart[7]=state[7];
    Cart[8]=state[8];

    //mexPrintf("%f\nCart=", Cart[0]);

}


/* Function: mdlTerminate
=====================================================
 * Abstract:
 *    In this function, you should perform any actions that are necessary
 *    at the termination of a simulation.  For example, if memory was
allocated
 *    in mdlStart, this is the place to free it.
 *
 */
static void mdlTerminate(SimStruct *S)
{

}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

ANNEXE V : Identification des paramètres géométriques

```
%*************************Le 16 mars 2012*****************************
%*******Identification des paramètres cinématique de l'orthoglide******
%*************************par Philippe LELIAS*************************

clc; clear; close all;

%---------------Géométrie des jambes--------------------------------
% L=Longueur et d= Largeur  du parallélogramme
% r=Distance entre le point C et le TCP(point P)

%L = 310.25; d= 80; r= 31;
L = 775; d= 125; r= 220; % d et r ne servent à rien dans le modèle

%-------Limites des variables d'articulations du vecteur d'entrés-------
r_min =   270;
r_max  =  150;

%-------------------------Calcul des alphas:---------------------------
%---angle entre les jambes y, z et les axes cartesiens correspondants---
%----------------alpha_1 -->angle en position maximal des jambes--------
%----------------alpha_2 -->angle en  position minimal------------------
alpha_1 = asin(r_max/L)
alpha_2 = asin(r_min/L)

%-----------Calcul des sinus, cosinus et tangente des angles alpha 1 et 2
C1 = cos(alpha_1); S1 = sin(alpha_1); T1=S1/C1;
C2 = cos(alpha_2); S2 = sin(alpha_2); T2=S2/C2;

%***********Methode d'une mesure à double poision**********************
%***********Systéme à 12 équatios linéaire et 3 inconnus****************

%-----------Calcul des coeffients géométrique des parallélogrammes---------
-
b1 = S1; c1 = (0.5+S1)*T1; b2 = -S2;;c2 = -(0.5 - S2)*T2;
disp('**********Identification des paramètres de Orthoglide*************')
disp(' ')
disp('------------------Paramètres du systéme à 12 équations------------')
disp(['le coefficient b1 : ',num2str(b1)])
disp(['le coefficient b2 : ',num2str(b2)])
disp(['le coefficient c1 : ',num2str(c1)])
disp(['le coefficient c2 : ',num2str(c2)])

%-----------Matrice Jacobienne du systéme à 12 équations----------------
    J12 = [ b1      c1      0; ...
            c1      b1      0; ...
            b2      c2      0; ...
            c2      b2      0; ...
            b1      0       c1; ...
            c1      0       b1; ...
            b2      0       c2; ...
            c2      0       b2; ...
            0       b1      c1; ...
            0       c1      b1; ...
            0       b2      c2; ...
            0       c2      b2  ];
```

```
%----------Matrice d'identité 12X12----------------------------------
    G = [ 2  0  1  0  0  0  0  0  0  0  0  0; ...
          0  2  0  1  0  0  0  0  0  0  0  0; ...
          1  0  2  0  0  0  0  0  0  0  0  0; ...
          0  1  0  2  0  0  0  0  0  0  0  0; ...
          0  0  0  0  2  0  1  0  0  0  0  0; ...
          0  0  0  0  0  2  0  1  0  0  0  0; ...
          0  0  0  0  1  0  2  0  0  0  0  0; ...
          0  0  0  0  0  1  0  2  0  0  0  0; ...
          0  0  0  0  0  0  0  0  2  0  1  0; ...
          0  0  0  0  0  0  0  0  0  2  0  1; ...
          0  0  0  0  0  0  0  0  1  0  2  0; ...
          0  0  0  0  0  0  0  0  0  1  0  2 ];

%---Calcul de la matrice de covariance d'identifications des paramétres---
M12 = inv(J12'*J12) * J12'*G*J12 * inv(J12'*J12);

%--------------------Erreurs d'étalonnage-----------------------------

tr12 = M12(1,1) + M12(2,2) + M12(3,3);
f12 = sqrt(tr12/3);
disp(' ')
disp(['Bruit de meusre calculé 12 equations : ',num2str(f12)])

%***********Systéme à 6 équatios linéaire et 3 inconnus****************

%----------Calcul des coeffients mécanique des parallélogrammes----------
b = S1+S2;  c = (T1+T2)/2 + (S1*T1-S2*T2);
disp(' ')
disp('******************Version réduite du système******************')
disp('------------------Parametres du système à 6 équations----------')
disp(['le coefficient b : ',num2str(b)])
disp(['le coefficient c : ',num2str(c)])

%----------Matrice Jacobienne du systéme à 6 équations-----------------
     J6 = [ b       c       0; ...
            c       b       0; ...
            0       b       c; ...
            0       c       b; ...
            b       0       c; ...
            c       0       b  ];

%---Calcul de la matrice de covariance d'identifications des paramétres---
M6 = inv(J6'*J6) * J6'* 2 *J6 * inv(J6'*J6);

%--------------------Erreurs d'étalonnage---------------------------------
tr6 = M6(1,1) + M6(2,2) + M6(3,3);
f6= sqrt(tr6/3);
disp(' ')
disp(['Bruit de mesure calculé 6 equations : ',num2str(f6)])

%***Calcul des offsets avec la matrice  pseudo inverse de Moore-Penrose***

%--------------------Valeur mesurée sur les 3axes-------------------------
%-------------------avec un comparateur mécanique-------------------------
%-------------avant les réglages mécanique et calibration---------------
```

```
%----------------------Expérience n°1 Axe X----------------------------
X_1_y=(2.10+1.48);
X_1_z=1.13+1.14;

%----------------------Expérience n° 1 Axe Y----------------------------
Y_1_x=+(0.8+1.09);
Y_1_z=1.56+2.11;

%----------------------Expérience n° 1 Axe Z----------------------------
Z_1_x=1.35+1.82;
Z_1_y=1.02+1.58;

%---------------Valeur mesurée sur les 3 axes----------------------------
%---------------avec un comparateur mécanique----------------------------
%------------aprés les réglages mécanique et  avant calibration---------

% %----------------------Expérience n°2 Axe X----------------------------
-
X_2_y=-(0.28+0.56);
X_2_z=0.13+0.45;
%
% %----------------------Expérience n° 2 Axe Y----------------------------
-
 Y_2_x=-(0.12+0.64);
 Y_2_z=0.10+0.04;
%
% %----------------------Expérience n° 2 Axe Z----------------------------
-
 Z_2_x=0.8+0.42;
 Z_2_y=0.24;
%
% %----------------------Valeur mesurée sur les 3 axes--------------------
-
% %---------avec un comparateur mécanique aprés les calibrations----------
-
%
% %----------------------Expérience n°3 Axe X----------------------------
-
% X_3_y=-0.23;
% X_3_z=0.27;
%
% %----------------------Expérience n° 3 Axe Y----------------------------
-
% Y_3_x=0.34;
% Y_3_z=-0.10;
%
% %----------------------Expérience n° 3 Axe Z----------------------------
-
% Z_3_x=-0.09;
% Z_3_y=+0.11;
%
%----------------------Matrice 6 équations-------------------------------
%A=[b c 0; c b 0; 0 b c; 0 c b; b 0 c; c 0 b];
E_1=[X_1_y; Y_1_x; Y_1_z; Z_1_y; X_1_z; Z_1_x];
E_2=[X_2_y; Y_2_x; Y_2_z; Z_2_y; X_2_z; Z_2_x];
%E_3=[X_3_y; Y_3_x; Y_3_z; Z_3_y; X_3_z; Z_3_x];

%---------------------Matrice pseudo-inverse de A-----------------------
A=pinv(J6);
```

```
%-------------------Calcul des offsets des axes x,y et z-----------------
disp(' ')
disp('******************Offsets avec 6 équations**********************')
disp('--------------Offset avant réglage et calibration--------------- ')
O_1=A*E_1 / 1.041666666666667e-003



%disp('--------------Offset aprés  réglage et avant calibration---------')
O_2=A*E_2/ 1.041666666666667e-003
%disp('--------------Offset aprés calibration--------------------------')
%O_3=A*E_3
```