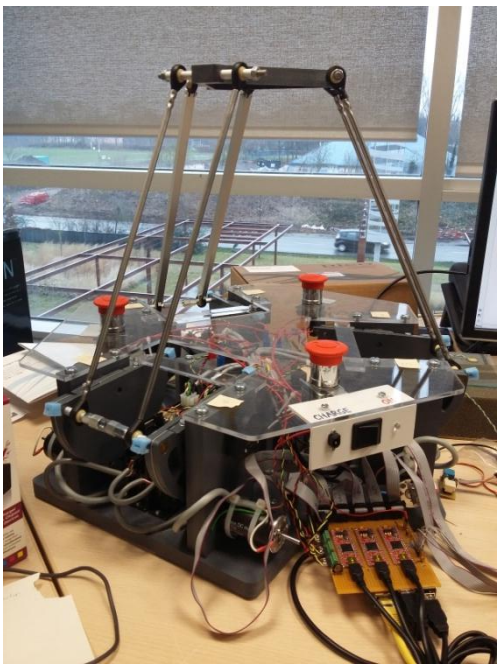
COFFIN Alice
MARRUCHO Diana

Polytech Lille – 4ème année
Informatique, Microélectronique, Automatique
2016-2017

Rapport de projet P34

« Interface Haptique, simulateur de formes et de textures »



Responsables :
M. GIRAUD Frédéric
Mme. SEMAIL Betty
M. GRISONI Laurent

En partenariat avec l'IRCICA

Remerciements

Nous tenions à remercier dans un premier temps l'IRCICA[1], pour nous avoir permis d'accéder aux locaux et le prêt du matériel.

Dans un second temps, nous souhaitons remercier l'équipe *CRIStAL-L2EP-MINT* [2]pour l'accueil et leur disposition.

Ensuite, nous souhaitons remercier nos tuteurs Madame Betty SEMAIL et messieurs Frédéric GIRAUD et Laurent GRISONI d'avoir proposé ce sujet.

Pour finir, nous remercions tout particulièrement Monsieur Frédéric GIRAUD pour ses connaissances techniques, ses conseils et son enseignement apportés lors de ce semestre, et pour le temps qu'il nous a consacré. Merci aussi à Madame Betty SEMAIL pour son accueil et son aide apportée, ainsi qu'à Monsieur Michel AMBERG pour la prise en main du projet.

[1] Institut de Recherche sur les Composants logiciels et matériels pour l'Information et la Communication Avancée

[2] Centre de Recherche en Informatique, Signal et Automatique de Lille - Laboratoire d'Electrotechnique et d'Electronique de Puissance

Sommaire

Table des figures

Introduction

Dans le cadre de notre quatrième année en formation IMA[3] à l'école d'ingénieur Polytech Lille de l'Université de Lille, nous avons rejoint l'équipe de l'IRCICA, *CRIStAL-L2EP-MINT*, afin de réaliser un projet en rapport avec notre formation.

Notre travail consistait à poursuivre le projet et le stage de Valentin CATTIAU, ancien étudiant en Master 2 SMaRT[4] de l'Université de Lille qui a lui même continué le travail de Tanh Hung NGUYEN.

Le but premier de notre projet est l'étude de l'existant et la compréhension de ce qui a déjà été fait.

Ensuite, notre travail sur l'interface haptique, ayant été construite antérieurement par le laboratoire, consistait à étudier la communication entre l'interface via *Matlab Simulink* et le robot via le *Raspberry Pi 3*. Puis, nous avons procédé à l'étude du développement hardware et du code *Matlab Simulink* réalisé qui a nécessité un certain temps suite aux grand nombre de fichiers présents non triés.

Par la suite, nous avons effectué les tests des codes déjà présents et des *SPI SFunctions* de Valentin CATTIAU, et nous avons modifié sa commande de sorte à tester les rotations.

Pour finir, Monsieur Frédéric GIRAUD nous a transmis en semaine 6, après entretien d'avancement de projet, la simulation de la commande du contrôle du robot parallèle à six degrés de liberté su *Matlab Simulink* ayant été réalisée par Valentin CATTIAU. Cependant, suite au manque de temps et à la complexité de réguler à la fois les translations et rotations du système, seules la commande des mouvements de translation ont été étudiées par Valentin. En fin de semaine 8, nous avons reçu le rapport nous permettant de comprendre la simulation réalisée.

Le but final de notre projet était donc la commande du robot dans l'espace cartésien qui permet de découpler les axes de translations et de rotations afin de connaître les forces à appliquer au niveau des six moteurs pour mettre en mouvement la plateforme.

[3] Informatique, Microélectronique, Automatique

[4] Systèmes, Machines autonomes et Réseaux de Terrains

1. Présentation de l'activité de l'équipe projet et état de l'art

1.1. Présentation de l'activité de l'équipe

L'equipe Mint, pour qui nous avons fait ce projet, regroupe au sein de l'INRIA des membres de deux laboratoires, CRIStAL et le L2EP. Ils travaillent principalement sur les interactions Homme-machine qui est un thème de recherche important du fait du développement des machines tout autour de nous.

L'équipe s'attache à développer, dans le cadre du projet STIMTAC, des stimulateurs tactiles : ils travaillent donc dans le domaines de l'haptique. L'haptique est la science du toucher. Le but de ce projet est donc de reproduire l'illusion de toucher des textures ou des formes en utilisant des dispositifs programmables.

Dans le cadre de ce projet, l'équipe a développé des stimulateurs tactiles. En faisant vibrer à très hautes fréquences une dalle tactile, à l'aide d'actionneurs piézoélectriques, et en modulant l'amplitude vibratoire en fonction de la position du doigt sur la plaque, il est possible de donner l'illusion de toucher une surface texturée. En associant ce système avec un écran il est possible de "toucher l'image à l'écran".

Pour améliorer les sensations tactiles Tao ZHENG a réalisé un dispositif permettant de donner l'illusion de toucher une forme géométrique à trois dimensions. Cela est possible grâce à deux robots *Falcon* couplés. Chaque robot est attaché à une extrémité d'une plaque plane (voir *Figure 2* ci-dessous). L'orientation de l'élévation de cette surface en fonction de la position du doigt permet de simuler des courbures. Ainsi, cette surface plane, mais animée, peut paraître courbe, ronde ou formée de vagues. En couplant ce système avec le dispositif de reproduction de texture, il est possible de donner l'illusion d'objet réel comme une orange et la texture de sa peau.

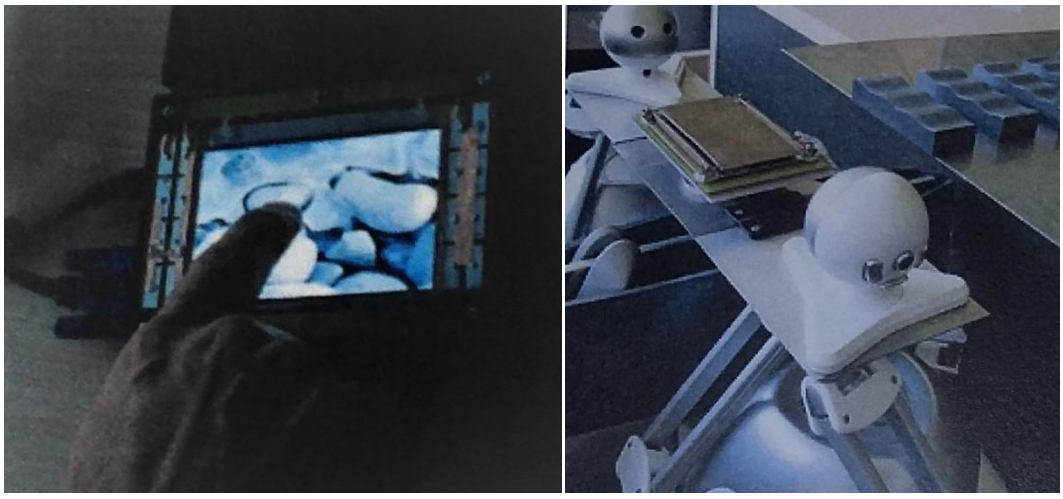


Figure 2 - Ecran stimulateur tactile (à gauche) et couplage des robots Falcon (à droite)

Ce système fonctionnait mais comportait beaucoup de jeu. Pour améliorer les performances et créer des formes plus complexes, il a été décidé de construire une plateforme à six degrés de liberté. Tanh HUNG s'est donc penché sur l'étude et la conception du robot Hexapod à 6DDL. Il a élaboré la structure mécanique du robot ainsi que son architecture de puissance.

1.2. Robot Hexapod

Le robot Hexapod est doté d'une plateforme reliée à 6 bras, chacun piloté par un moteur à courant continu. La présence des 6 bras permet à la plateforme de bouger aussi bien en translation qu'en rotation. Les moteurs sont reliés à des réducteurs de vitesse par cabestans afin de réduire les frottements et les jeux dans la chaîne cinématique.

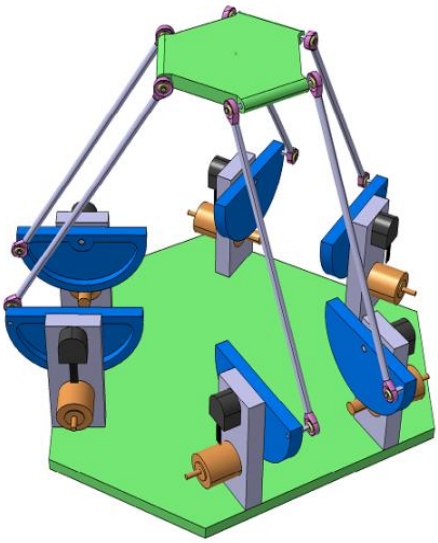


Figure 3 - Conception CAO du robot Hexapod

Un robot de ce type peut se commander de deux façons : soit dans l'espace articulaire (en utilisant la position des moteurs), soit dans l'espace cartésien (en x, y, z dans le repère de la plateforme.

Le travail de Tanh HUNG fut de piloter le robot en boucle ouverte dans l'espace articulaire. Il s'est rendu compte que cette méthode est instable à cause du couplage entre les axes. En effet, lorsqu'un bras est en mouvement, il provoque un mouvement au niveau de la plateforme mais induit aussi des déplacement sur les autres bras. Il existe un couplage entre les différents bras, ce qui produit des vibration. Or, cette application nécessite que la plateforme soit stable car si elle se mettait à vibrer lors d'un déplacement cette vibration serait ressentie sous le doigt de l'utilisateur et perturberait le retour tactile.

Pour éviter ces vibrations il faut découpler les axes, pour cela le robot doit être commandé dans l'espace cartésien (voir partie ***3.1.*** Commande et PID). Valentin CATTIAU a commencé ce travail et a traité uniquement les translations (la partie des rotations étant plus compliquées à implémenter).

1.3. Etat de l'art

1.3.1. Espaces articulaire et cartésien

En robotique il existe deux espace pour contrôler un robot : l'espace articulaire et l'espace cartésien.

L'espace articulaire est rattaché aux angles des moteurs que l'on note θ. De cette façon il est possible de contrôler les robots axe par axe. Cette méthode est utilisée pour contrôler des bras manipulateurs. Avec un robot parallèle, tel que le robot Hexapod étudié dans notre projet, on ne le contrôle pas aisément de cette façon car il existe un couplage.

L'espace cartésien est rattaché à un point donné du robot (le centre de gravité généralement) et les positions de l'objet (pince, plateforme, …) sont données en fonction de ce point, sur x, y et z.

Pour passer d'un espace à l'autre il existe différentes relations :

- Géométrique directe : (x, y, z) = f(θ_i)
- Géométrique inverse : (θ_i) = f(x, y, z)
- Cinématique directe : $(\dot{x}, \dot{y}, \dot{z})$ = f($\dot{\theta}_i$)
- Cinématique inverse : $\dot{\theta}_i$= f$(\dot{x}, \dot{y}, \dot{z})$

Pour $1 \leq i \leq 6$.

1.3.2. Autres robots existants

Delta

Souvent utilisé comme imprimante 3D ou comme robot "pick-and-place", la plateforme se déplace seulement en translation, il n'y a pas de rotation du plateau. C'est le model de ce robot que Valentin CATTIAU a utilisé lors de son projet et stage.

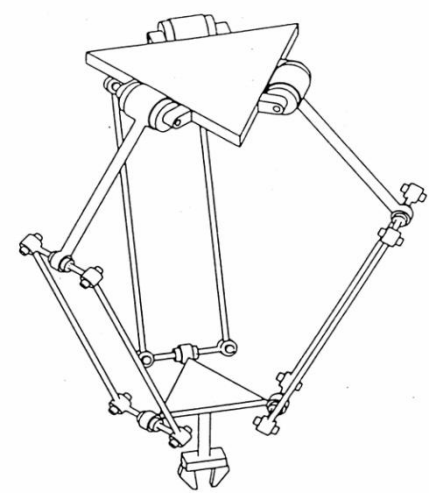


Figure 4 - Robot Delta

Figure 5 - Robot Hexa II

Hexa II

Le robot Hexa a un fonctionnement similaire à celui voulu pour le robot Hexapod. La plateforme se déplace sur x, y et z et peut effectuer des rotations autour de ces axes.

Cependant, nous n'avons pas eu assez de temps pour trouver suffisamment d'information pour nous aider à finir notre partie du projet.

2. Contrôle du robot Hexapod via Raspberry PI 3

2.1. Simulink - Raspberry

A travers un Bus USB, la communication est possible afin de pouvoir commander le robot grâce au pc. Après avoir réaliser des programmes sous *Simulink*, il est possible de téléverser directement sur le Raspberry le fichier d'extension 'slx' en langage C++ de sorte à ce qu'il soit compréhensible pour ce dernier. Le Raspberry communique ensuite avec les six cartes EPOS2 24/5 pour contrôler les différents moteurs et avec les codeurs pour récupérer les positions.

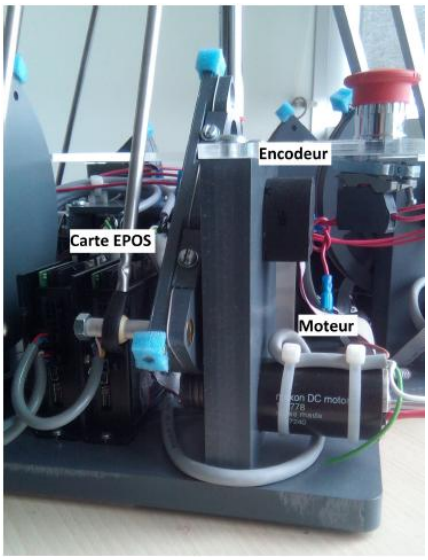


Figure 6 - Hardware

2.2. Tri et test des fichiers existants

Dans une première partie du projet, nous avons eu pour mission de récupérer les programmes réalisés par Valentin durant son stage et de les tester.

Ces tests nous ont permis de voir quel était l'utilité de chaque programme : certains n'étaient pas ou plus fonctionnels.

Pour exemple, lors de ces tests, nous avons effectué des essais du fichier "cercle.slx" de *Matlab Simulink*.

Dans les vidéos disponibles sur notre wiki de suivi de projet**[1]**, nous avons travaillé avec les bras du robot liés deux à deux afin de regarder s'il réalisait bien un cercle, puis nous les avons par la suite détachés.

Comme les bras sont devenus indépendants, il a été nécessaire d'utiliser tous les retours de position et non uniquement un pour deux. Les modifications effectuées sont donc visibles sur la ***Figure 7*** - Modification du programme 'cercle.slx' afin de récupérer toutes les positions page suivante.

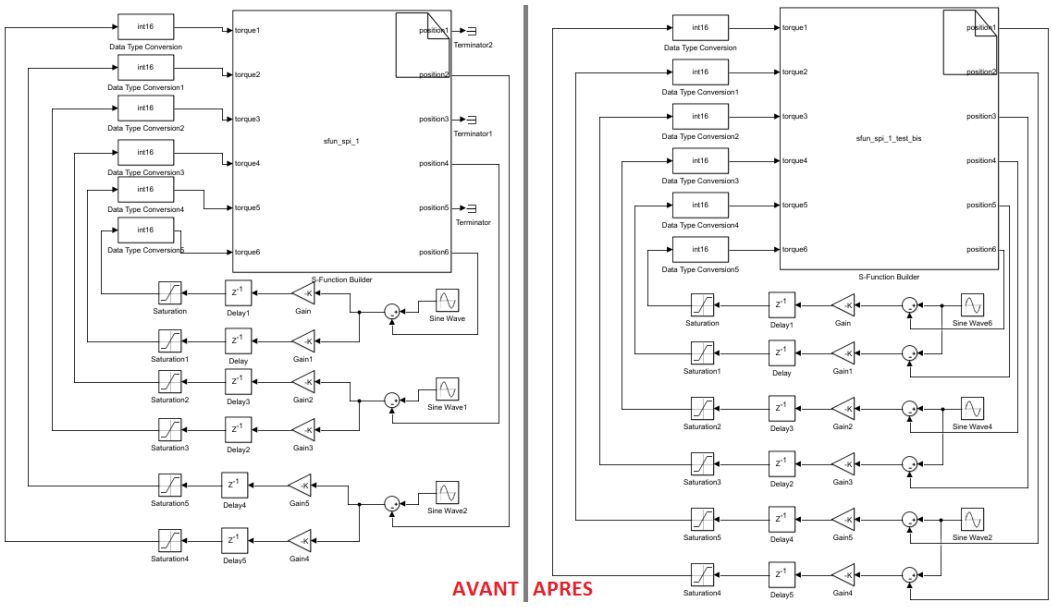


Figure 7 - Modification du programme 'cercle.slx' afin de récupérer toutes les positions

Dans ces vidéos, on peut voir que lorsque les bras sont liés, ils effectuent deux à deux le même mouvement, cependant, lorsqu'ils sont détachés (avec pourtant la même consigne), les mouvements sont désynchronisés.

Il aurait donc été nécessaire de chercher la raison du problème de synchronisation, mais c'est à ce moment là que nous avons été orientées vers une autre partie du projet.

3. Récupération de l'existant

3.1. Commande et PID[5]

Dans un premier temps, nous avons cherché à comprendre le schéma de simulation de la commande *Simulink*.

Cependant, les codes précédemment réalisés n'étaient pas commentés donc nous avons mis du temps à comprendre ce qui avait été fait. Suite à notre demande de renseignements auprès de monsieur GIRAUD, nous avons fini par obtenir le rapport précédemment réalisé par Valentin CATTIAU sur la réalisation de ce schéma.

Ainsi, nous nous sommes aperçues que selon notre cours de Robotique avec monsieur MERZOUKI et avec nos recherches, ce qui était qualifié par notre prédécesseur de cinématique était en réalité la géométrique, et la dynamique était en fait la cinématique, d'où les soucis de compréhension.

3.1.1. Définitions

La **cinématique directe** calcule la vitesse dans le domaine cartésien (domaine opérationnel) en fonction de la vitesse angulaire (domaine articulaire) par l'intermédiaire d'une matrice jacobienne J. Le positionnement d'un objet à plusieurs degrés de liberté en fonction de tous les paramètres d'articulation est le contrôle cinématique direct.

$$\dot{\Omega} = \frac{d}{dt} f(\theta) = J.\dot{\theta}_\iota = \frac{\partial \mathrm{f}}{\partial \theta_i} \dot{\theta}_\iota(t)$$

avec $1 \leq i \leq 3$, $1 \leq j \leq 2$, et où $\dot{\Omega} = \{\dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma}\}$

La **cinématique inverse** calcule la vitesse angulaire (domaine articulaire) en fonction de la vitesse dans le domaine cartésien (opérationnel), soit l'inverse de la cinématique directe, avec également l'intervention de la matrice jacobienne J. Les paramètres d'articulation sont calculés automatiquement à partie de la position souhaitée et de son orientation.

$$\dot{\theta_{\iota_J}}(t) = \frac{\partial \mathrm{f}^{-1}}{\partial \theta_{ij}} (\dot{\Omega})$$

avec $1 \leq i \leq 3$, $1 \leq j \leq 2$, et où $\dot{\Omega} = \{\dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma}\}$

Les **modèles géométriques** sont des modèles mécaniques utilisés dans le domaine de la robotique, principalement pour les robot manipulateurs et parallèles.

[5] Proportionnel, Intégral, Dérivé

La **géométrique directe** caractérise le fonctionnement du robot, soit la configuration finale de celui-ci en fonction des différents (ici six) moteurs.

$$\Omega = f(\theta_{ij})$$

avec $1 \leq i \leq 3, 1 \leq j \leq 2$, et où $\Omega = \{x, y, z, \alpha, \beta, \gamma\}$

La **géométrique inverse** caractérise le positionnement de chaque moteur du robot en fonction de la configuration terminale de celui-ci.

$$\theta_{ij} = f(\Omega)$$

avec $1 \leq i \leq 3, 1 \leq j \leq 2$, et où $\Omega = \{x, y, z, \alpha, \beta, \gamma\}$

3.1.2. Contrôle du robot

Afin de réaliser la commande du robot, une étude en boucle ouverte avait déjà été réalisée par le prédécesseur de Valentin CATTIAU, Tanh Hung NGUYEN, qui a construit le robot Hexapod. Cette étude du comportement nous permet donc de récupérer les gains du correcteur à appliquer au robot.

La modélisation en boucle ouverte sous Simulink est la suivante :

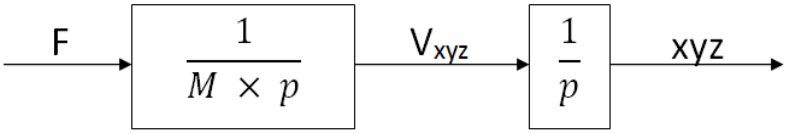


Figure 8 - Modélisation boucle ouverte

Lors de cette simulation, en entrée du système un 'step' a uniquement été appliqué sur l'angle θ_1 d'un des trois bras du robot, soit sur un unique couple de moteur, le résultat obtenu est le suivant :

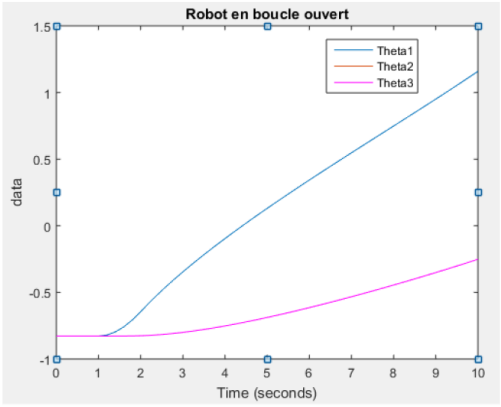


Figure 9 - Graphe de l'étude du robot en boucle ouverte avec une unique sollicitation en θ_1

Sur le graphe précédent, on observe bien que lorsque l'angle θ_1 est sollicité, les angles θ_2 et θ_3, donc les moteurs des deux autres bras, sont sollicités également et de la même façon.

Ainsi, il est donc nécessaire de découpler les axes du robots dans l'espace cartésien et non articulaire pour que les moteurs ne fonctionnement pas indépendamment les uns des autres.

C'est pourquoi, dans un premier temps, nous allons utiliser le modèle cinématique direct puis passer par le modèle géométrique inverse qui nous permet de récupérer les position θ_{ij} des moteurs et enfin, par l'intermédiaire du modèle géométrique direct, on passe de l'espace articulaire à l'espace cartésien.

Ensuite, il sera possible d'utiliser un correcteur PID qui nous délivrera les forces selon x, y, z.

Enfin, en repassant par la cinématique inverse, nous pouvons réaliser une boucle de contrôle.

Ainsi, nous obtenons le schéma de contrôle du robot Hexapod en boucle fermée suivant :

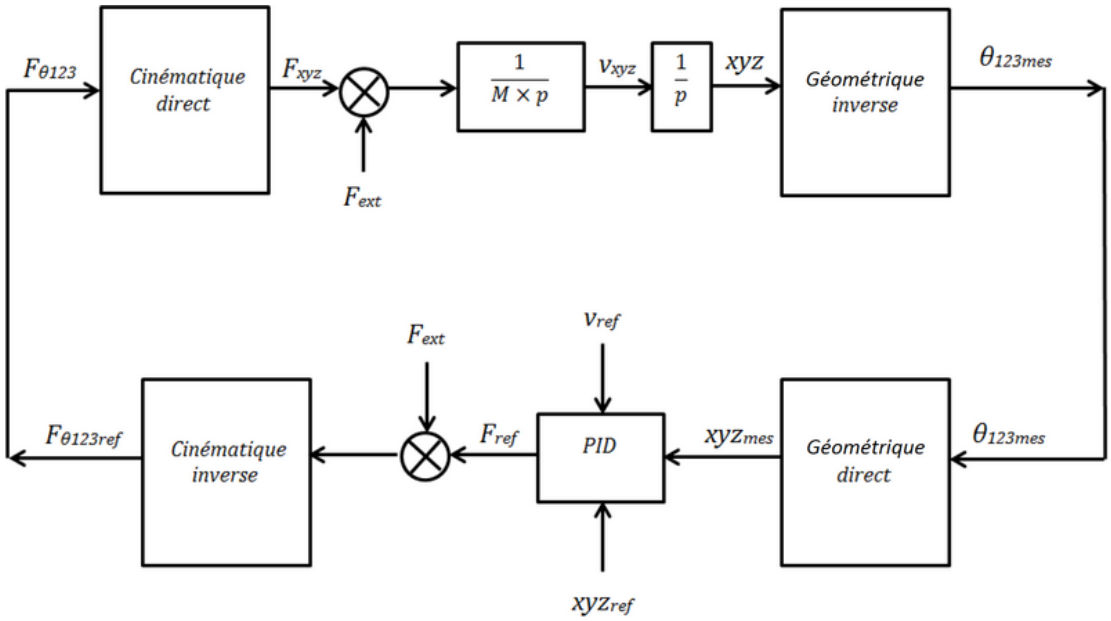


Figure 10 - Schéma Simulink de modélisation de la commande et du contrôle du robot Hexapod

3.1.3. Contrôleur PID

Pour contrôler le robot Hexapod, nous n'utilisons pas le PID offert par *Matlab Simulink* mais un PID préalablement réalisé par Valentin CATTIAU[2].

Ce PID à l'avantage d'insérer en entrée 4 la position de référence désirée qu'on compare à l'entrée 1 qui correspond à la position mesurée. Lors de la comparaison, on en ressort une erreur de position en sortie 2.

Ainsi, on notera ***Kp*** le gain de l'erreur de position, ***Kd*** le gain de l'erreur de vitesse et ***Ki*** le gain intégral de l'erreur de position.

Ce PID, à contrario de celui de *Matlab*, a pour avantage de commander en position et en vitesse le robot. Ainsi, en sortie 1, on en ressort une force à appliquer selon chaque axe, qui sera ensuite transformée par la cinématique inverse en force à appliquer au moteur.

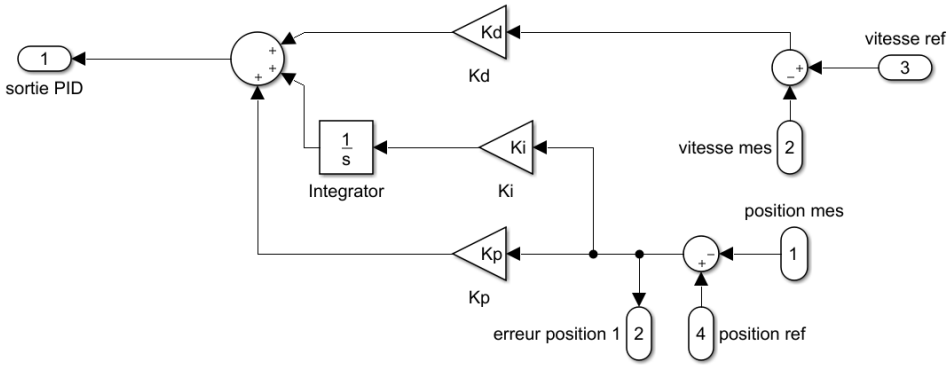


Figure 11 - Schéma Simulink du régulateur PID de la commande

Les valeurs des gains du PID régulateur calculés grâce à la commande Matlab place(A,B,pole) dans ce PID sont les suivants :

- $K_d = 32,98$
- $K_p = 724,9159$
- $K_i = 5,3066.10^3$

3.1.4. Commande globale Simulink

Ci-dessous, en ***Figure 12***, le système a été divisé en deux avec un sous-système décrivant la modélisation du robot, et l'autre la partie contrôle (nécessitant obligatoirement la modélisation robot).

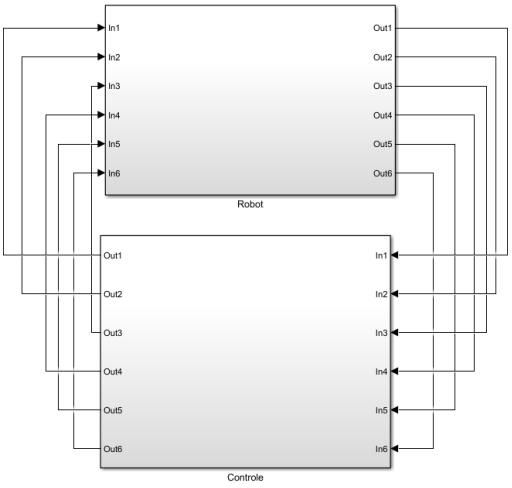


Figure 12 - Modèle global divisé en deux sous-systèmes (Robot et Contrôle)

Les sous-systèmes correspondants sont détaillés en ***Annexe* 1** : Sous-système robot et ***Annexe 2*** : Sous-système contrôle.

3.2. Résultats et tests

Nous avons testé la commande Simulink de Valentin CATTIAU en simulant le déplacement de la plateforme en formant un cercle.

Pour former un cercle il faut mettre une sinusoïde en commande de x et cette même sinusoïde déphasée de $\pi/4$ pour y.

z est commandé par une constante de consigne 0.275 (la hauteur en mètre entre la plateforme et le socle du robot au repos).

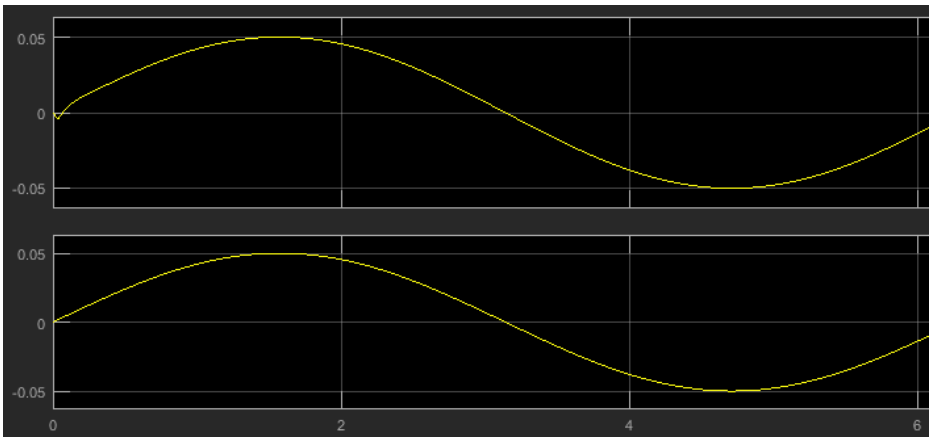


Figure 13 - Valeurs de x mesurées par simulation (courbe supérieure) et de consigne (courbe inférieure)

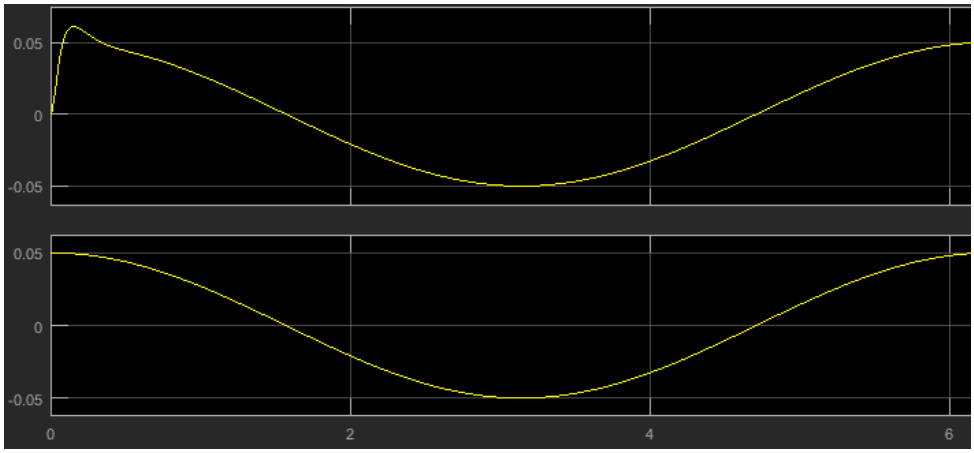


Figure 14 - Valeurs de y mesurées par simulation (courbe supérieure) et de consigne (courbe inférieure)

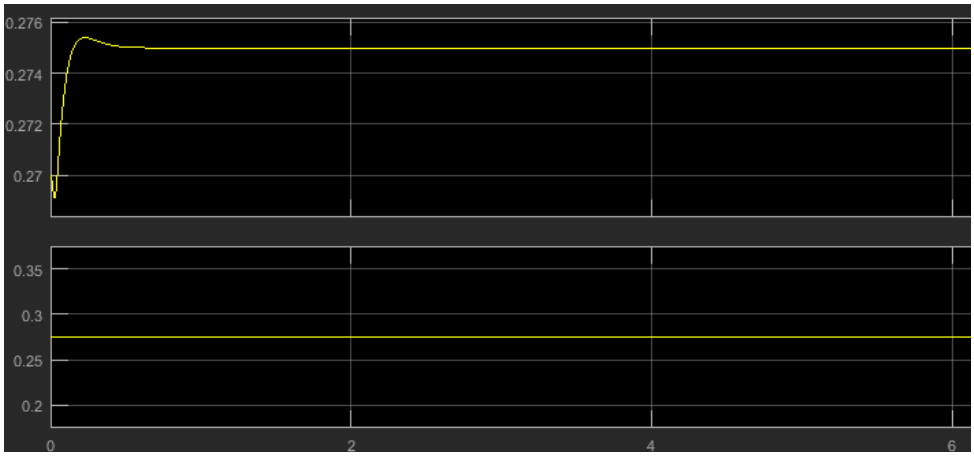


Figure 15 - Valeurs de z mesurées par simulation (courbe supérieure) et de consigne (courbe inférieure)

Lors de la simulation, la valeur initiale des valeurs mesurées étant à 0, il y a un léger dépassement de position au début du déplacement, ce qui est tout à fait normal.

Par la suite, les valeurs “mesurées” collent parfaitement aux consignes. En plaçant un ‘grahXY’ on voit que le cercle se forme bien. On peut donc en déduire que la commande est correcte et qu’elle peut être implantée sur le robot.

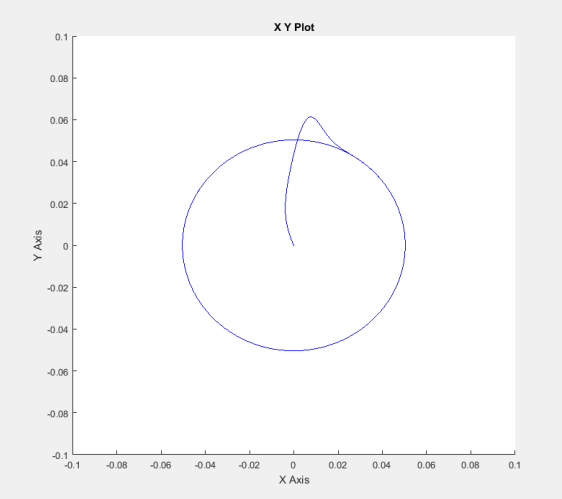


Figure 16 - Graphe XY représentant un cercle effectué par la plateforme

Les sorties des PID sont ensuite dirigées vers le bloc “Cinématique inverse” qui, en sortie, donne la force à appliquer sur chaque moteur pour faire fonctionner le robot.

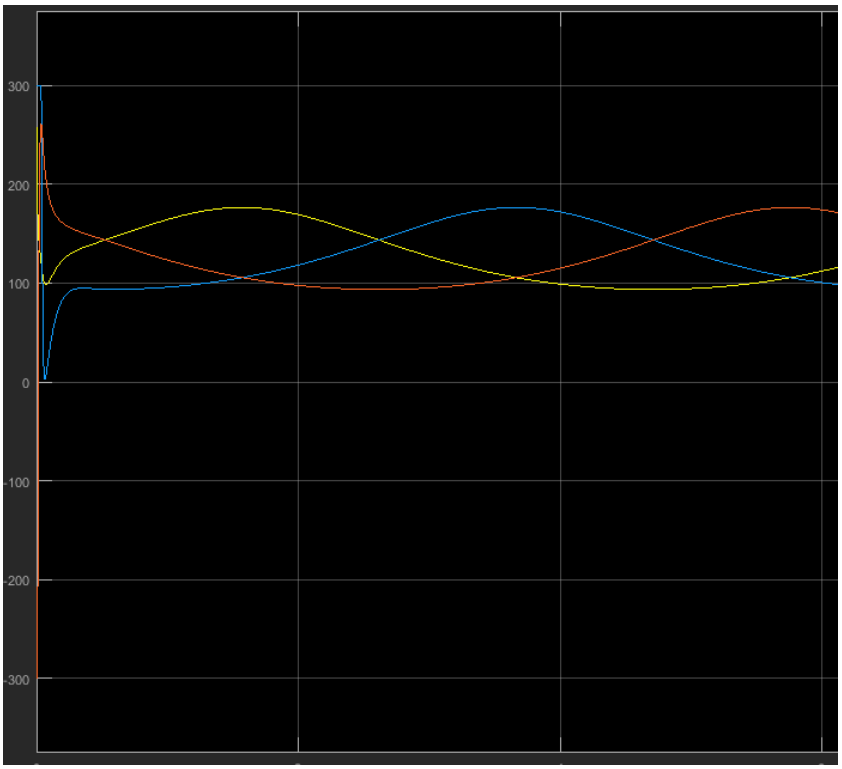


Figure 17 - Couples à appliquer à chaque moteur

4. Modélisation du robot Hexapod à 6 degrés de liberté

Afin de pouvoir modéliser notre système, nous avons dû recalculer les modèles cinématiques et géométriques ainsi que la jacobienne inverse.

Nous expliquons dans la suite comment nous avons déterminé ces parties.

5.1. Jacobienne inverse

La matrice jacobienne J est une matrice associée à la fonction vectorielle en un point donné.

Notre matrice J permet de calculer la vitesse de chaque bras grâce à la formule suivante :

$$\dot{\theta} = \mathbf{J}.\ \mathbf{v}$$

(avec $\dot{\theta}$ le vecteur vitesse de rotation des bras et v le vecteur vitesse généré par le plateau supérieur)

Dans un premier temps avec un code Matlab, nous avons calculé symboliquement la Jacobienne inverse de notre système afin de pouvoir déterminer les paramètres cinématiques que nous avons eu besoin pour les calculs de la cinématique directe et inverse.

A savoir, le modèle cinématique direct permet de calculer les vitesses dans le domaine cartésien ($\dot{\theta}mot$) en fonction de la vitesse angulaire ($\dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}\ et \dot{\gamma}$), et le modèle cinématique inverse permet de passer de la vitesse opérationnelle ($\dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}\ et \dot{\gamma}$) à la vitesse dans le domaine articulaire ($\dot{\theta}mot$)

Le calcul de la jacobienne inverse est disponible en ***Annexe 3*** : Calcul de la Jacobienne.

Nous nous sommes abstenues de mettre en annexe la jacobienne inverse générée par Matlab car elle contient plus de 700,000 caractères.

5.2. Les différents modèles géométriques et cinématiques

5.2.1. Calculs des paramètres

Maintenant que la jacobienne a été calculée (cf partie ***5.1.*** Jacobienne inverse) nous avons pu modifier les modèles cinématiques et géométriques afin de pouvoir obtenir les paramètres du contrôle des translations et des rotations.

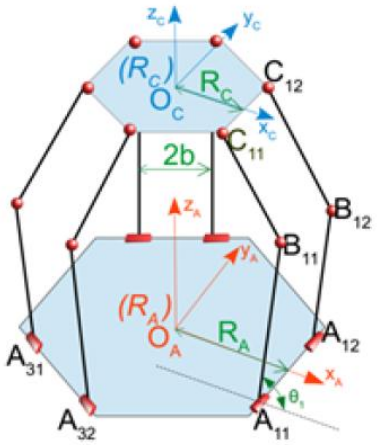


Figure 18 - Schéma représentatif du robot

Calcul de la cinématique inverse

En utilisant le calcul de la jacobienne, on peut retrouver les paramètres articulaires en fonction des paramètres cartésiens.

Ainsi, on effectue le calcul ci-dessous (où J est la jacobienne et v la vitesse de la plateforme) :

$$\dot{\theta_{ij}} = Jv$$

Calcul de la cinématique directe

En utilisant le calcul de la jacobienne, on peut retrouver les paramètres cartésiens en fonction des paramètres articulaires.

Ainsi, on effectue le même calcul qu'en cinématique inverse à part qu'on utilise la matrice jacobienne inverse :

$$\dot{\theta_{ij}} = J^{-1}v$$

Calcul de la géométrique inverse

Comme expliqué en partie ***3.1.1.*** Définitions, le modèle géométrique inverse nous permet de passer de coordonnées cartésiennes à articulaires, soit des coordonnées du plateau mobile (x, y, z, α, β, γ) aux angles des moteurs (θ_{11}, θ_{12}, θ_{21}, θ_{22}, θ_{31}, θ_{32}).

Grâce à l'étude préalablement réalisée par Valentin dans sa partie modélisation du robot, nous savons que :

- Les points A_{ij} dans le repère R_A sont les suivants :
  - $(A_{11})_{R_A} = \begin{pmatrix} R_A \\ -b \\ 0 \end{pmatrix}$
  - $(A_{12})_{R_A} = \begin{pmatrix} R_A \\ b \\ 0 \end{pmatrix}$
  - $(A_{2j})_{R_A} = R_z\left(\frac{2\pi}{3}\right)(A_{1j})_{R_A}$
  - $(A_{3j})_{R_A} = R_z\left(\frac{4\pi}{3}\right)(A_{1j})_{R_A}$
- Les points B_{ij} dans le repère A_{ij} sont les suivants :
  - $(B_{ij})_{R_{Aij}} = \begin{pmatrix} La\cos(\theta_{ij}) \\ 0 \\ La\sin(\theta_{ij}) \end{pmatrix}$ où La correspond à la longueur du bras AB.
- Les points C_{ij} dans le repère A_{ij} sont les suivants :
  - $(C_{ij})_{R_{Aij}} = T_{Oa.Aij}.(C_{ij})_{R_A}$

Ainsi, grâce à cette détermination des points A_{ij}, B_{ij} et C_{ij}, il est donc possible de trouver les angles θ_i des moteurs grâce aux formules suivantes :

- θ_i entrainant les translations : $\theta_i = \frac{\theta_{ij} + \theta_{i(j+1)}}{2}$
- $d\theta_i$ entrainant les rotations : $d\theta_i = \frac{\theta_{ij} - \theta_{i(j+1)}}{2}$

Calcul de la géométrique directe

A partie du modèle géométrique inverse, on en ressort les relations suivantes :

- $\theta_i = \frac{\theta_{ij} + \theta_{i(j+1)}}{2}$
- $d\theta_i = \frac{\theta_{ij} - \theta_{i(j+1)}}{2}$

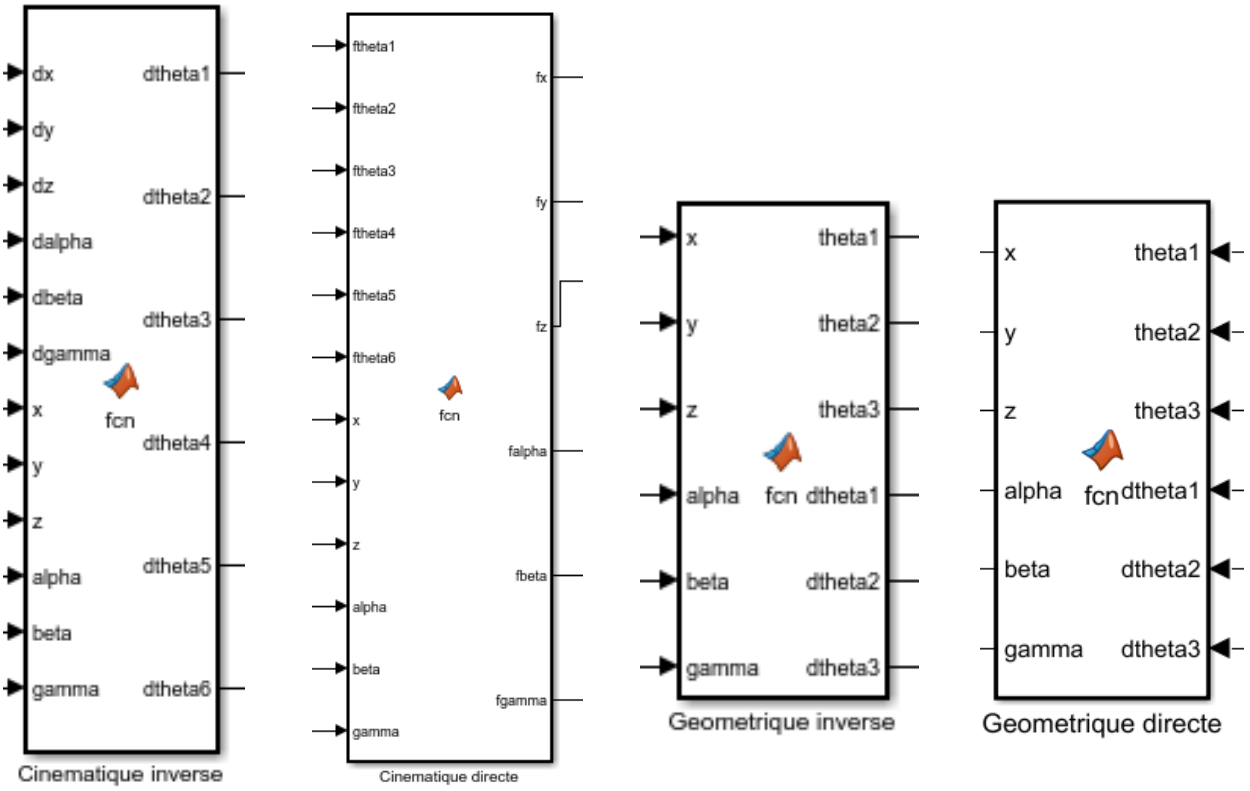
Valentin CATTIAU avait déjà travaillé sur les translations comme expliqué dans la présentation du projet de recherche car cela était similaire au robot Delta.

Cependant, malgré les recherches effectuées, nous n'avons pas eu assez de temps pour trouver comment calculer le modèle géométrique direct associé au robot Hexa pour les rotations.

Par la suite, lorsque notre projet sera repris, il serait donc judicieux de commencer par cette partie.

5.2.2. Modélisation en bloc Simulink

Voici ci-dessous les blocs de cinématique et géométrique modifiés afin de résoudre le problème des rotations :



(1) (2) (3) (4)

Figure 19 - Blocs Simulink de cinématique inverse (1), directe (2), géométrique inverse (3) et directe (4)

5.3. Système final sous Simulink

Après calcul des différents paramètres et modifications des codes Matlab, nous avons placé en ***Annexe 8*** : Sous-système robot (avec prise en compte rotations) et ***Annexe 9*** : Sous-système contrôle (avec prise en compte rotations) les différents sous-systèmes de notre robot Hexapsod tout en conservant le PID précédemment réalisé.

Les simulations n'ont pu être réalisée car le bloc de modélisation géométrique directe n'est pas exploitable.

Conclusion

Durant ce semestre, nous avons eu pour mission de poursuivre le travail de recherche effectué par Valentin CATTIAU et Tanh Hung NGUYEN sur l'interface Haptique réalisée à l'IRCICA.

Dans un premier temps, nous avons effectué un tri des programmes à envoyer au raspberry pi 3 de l'interface afin de réaliser les mouvements demandés. Tous les fichiers étaient mélangés et pas tous fonctionnels, du coup nous avons pris pas mal de temps à tester les fonctions sur le robot pour créer un répertoire de sauvegarde des programmes fonctionnels.

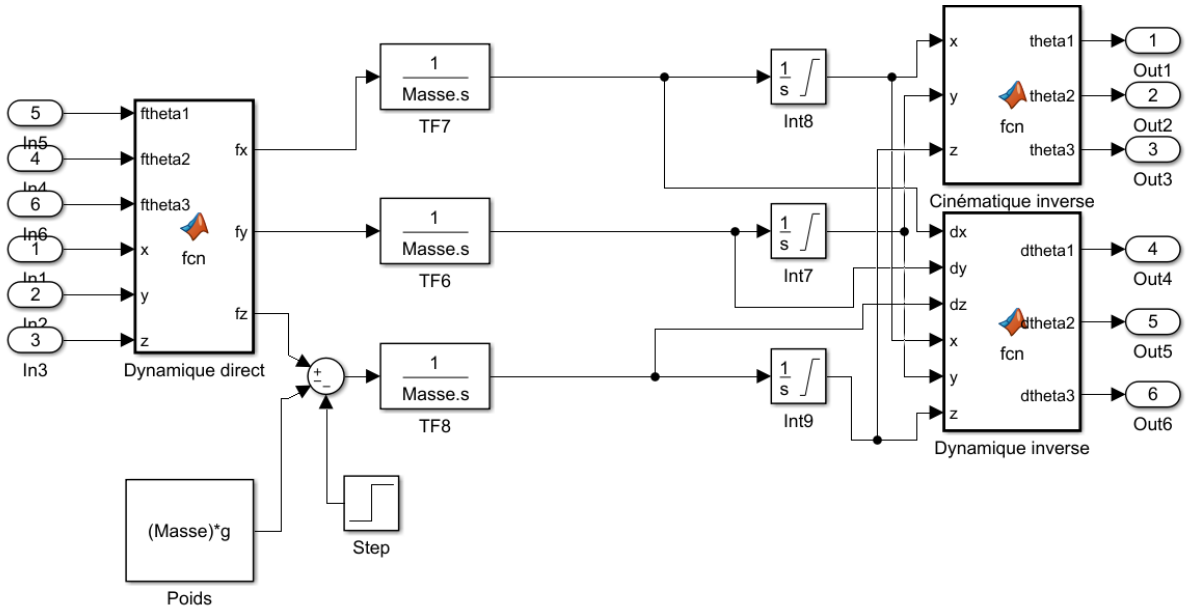
Dans une seconde partie, fin mars, nous avons changé d'objectif : nous devions réaliser la simulation *Matlab Simulink* de la commande du robot Hexapod. Cette commande avait déjà été en partie réalisée par notre prédécesseur dans le cadre de son projet, et nous avons donc modifié les codes des modélisations cinématiques et géométriques afin de les soumettre aux rotations.

Suite à la période de travail réduite, nous n'avons pu terminer notre système de contrôle. Donc, dans la suite, si notre projet venait à être repris, il serait judicieux de débuter par la recherche du modèle géométrique direct.

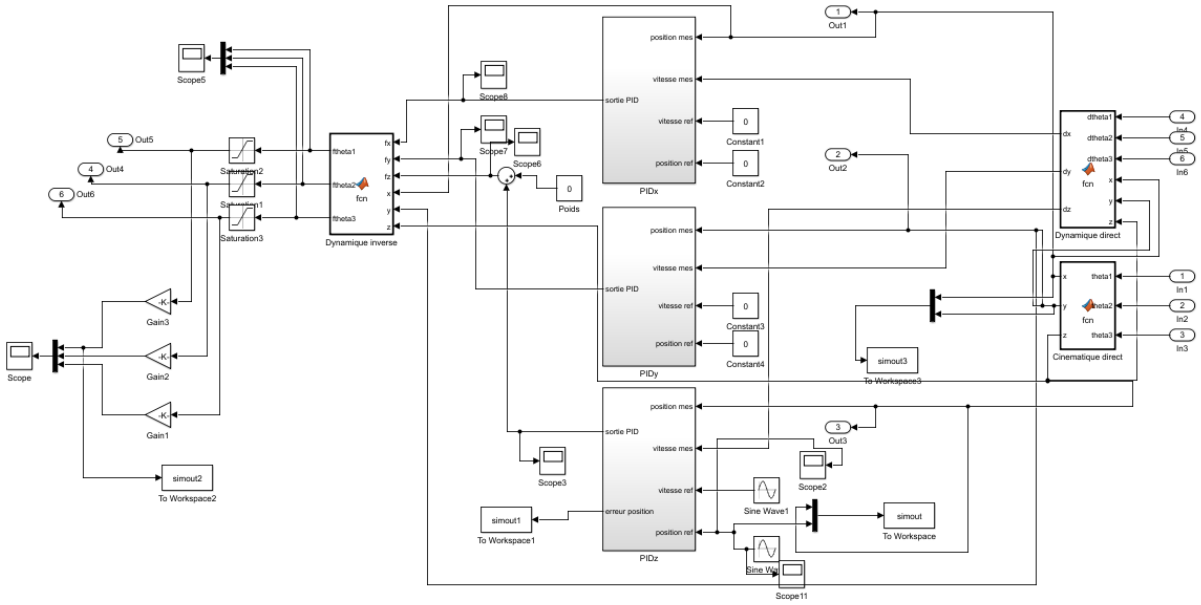
Pendant ce semestre, nous avons donc appris à travailler en binôme sur un même projet de recherche, et nous sommes désolées de n'avoir pu terminer notre mission.

Annexes

Annexe 1 : Sous-système robot (translations)



Annexe 2 : Sous-système contrôle (translations)



Annexe 3 : Calcul de la Jacobienne

```
function Jmoins1=jacobienne_inverse(x,y,z,alpha,beta,gamma)
La=sym('La');
LA=sym('LA');
Lc=sym('Lc');
Ra=sym('Ra');
Rc=sym('Rc');
b=sym('b');
x=sym('x');
y=sym('y');
z=sym('z');
alpha=sym('alpha');
beta=sym('beta');
gamma=sym('gamma');
La=0.069;
LA=0.069;
Lc=0.33 ;
Ra=0.169;
Rc=0.08;
b=0.049;


%Pour theta11
E11=(2*La*Ra - 2*La*z*sin(beta) - 2*La*Rc*cos(beta)*cos(gamma) - 2*La*b*cos(beta)*sin(gamma) -
2*La*x*cos(beta)*cos(gamma) + 2*La*y*cos(beta)*sin(gamma));

F11=(2*La*b*cos(gamma)*sin(alpha) - 2*La*Rc*sin(alpha)*sin(gamma) -
2*La*z*cos(alpha)*cos(beta) - 2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) +
2*La*Rc*cos(alpha)*cos(gamma)*sin(beta) + 2*La*b*cos(alpha)*sin(beta)*sin(gamma) +
2*La*x*cos(alpha)*cos(gamma)*sin(beta) - 2*La*y*cos(alpha)*sin(beta)*sin(gamma));

G11=-Lc^2+2*Rc*x - 2*b*y + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 - 2*Ra*z*sin(beta) -
2*b^2*cos(alpha)*cos(gamma) - 2*Ra*Rc*cos(beta)*cos(gamma) +
2*b^2*sin(alpha)*sin(beta)*sin(gamma) + 2*Rc*b*cos(alpha)*sin(gamma) -
2*Ra*b*cos(beta)*sin(gamma) - 2*Ra*x*cos(beta)*cos(gamma) + 2*b*y*cos(alpha)*cos(gamma) +
2*Ra*y*cos(beta)*sin(gamma) - 2*b*z*cos(beta)*sin(alpha) + 2*b*x*cos(alpha)*sin(gamma) +
2*Rc*b*cos(gamma)*sin(alpha)*sin(beta) + 2*b*x*cos(gamma)*sin(alpha)*sin(beta) -
2*b*y*sin(alpha)*sin(beta)*sin(gamma);

%Pour theta12
E12=(2*La*Ra - 2*La*z*sin(beta) - 2*La*Rc*cos(beta)*cos(gamma) + 2*La*b*cos(beta)*sin(gamma) -
2*La*x*cos(beta)*cos(gamma) + 2*La*y*cos(beta)*sin(gamma));

F12=(2*La*Rc*cos(alpha)*cos(gamma)*sin(beta) - 2*La*Rc*sin(alpha)*sin(gamma) -
2*La*z*cos(alpha)*cos(beta) - 2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) -
2*La*b*cos(gamma)*sin(alpha) - 2*La*b*cos(alpha)*sin(beta)*sin(gamma) +
2*LA*x*cos(alpha)*cos(gamma)*sin(beta) - 2*LA*y*cos(alpha)*sin(beta)*sin(gamma));

G12=La^2-Lc^2 + 2*Rc*x + 2*b*y + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 - 2*Ra*z*sin(beta) -
2*b^2*cos(alpha)*cos(gamma) - 2*Ra*Rc*cos(beta)*cos(gamma) +
2*b^2*sin(alpha)*sin(beta)*sin(gamma) - 2*Rc*b*cos(alpha)*sin(gamma) +
2*Ra*b*cos(beta)*sin(gamma) - 2*Ra*x*cos(beta)*cos(gamma) - 2*b*y*cos(alpha)*cos(gamma) +
2*Ra*y*cos(beta)*sin(gamma) + 2*b*z*cos(beta)*sin(alpha) - 2*b*x*cos(alpha)*sin(gamma) -
2*Rc*b*cos(gamma)*sin(alpha)*sin(beta) - 2*b*x*cos(gamma)*sin(alpha)*sin(beta) +
2*b*y*sin(alpha)*sin(beta)*sin(gamma);


%Pour theta21
E21=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 - (3*La*b*cos(alpha)*sin(gamma))/2 -
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) +
3^(1/2)*La*y*cos(alpha)*cos(gamma) - 3^(1/2)*La*z*cos(beta)*sin(alpha) +
3^(1/2)*La*x*cos(alpha)*sin(gamma) - (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 + (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 - 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));
```

```
F21=(La*Rc*sin(alpha)*sin(gamma) - La*b*cos(gamma)*sin(alpha) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) +
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) -
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) + 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) -
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) + 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G21=-Lc^2+b*y - Rc*x + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) -
3^(1/2)*Rc*y - 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 +
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 - (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 +
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 - (3*Ra*b*cos(alpha)*sin(gamma))/2 +
(Rc*b*cos(alpha)*sin(gamma))/2 - (Ra*b*cos(beta)*sin(gamma))/2 +
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) - b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) + b*z*cos(beta)*sin(alpha) - b*x*cos(alpha)*sin(gamma) +
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 + (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 + (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 +
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) - 3^(1/2)*Ra*z*cos(beta)*sin(alpha) +
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) + 3^(1/2)*b*x*cos(beta)*cos(gamma) -
3^(1/2)*b*y*cos(beta)*sin(gamma) - (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 -
b*x*cos(gamma)*sin(alpha)*sin(beta) + b*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 - 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
- (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta22
E22=(2*LA*Ra + LA*z*sin(beta) - (3*LA*Rc*cos(alpha)*cos(gamma))/2 -
(LA*Rc*cos(beta)*cos(gamma))/2 + (3*LA*b*cos(alpha)*sin(gamma))/2 +
(LA*b*cos(beta)*sin(gamma))/2 + LA*x*cos(beta)*cos(gamma) - LA*y*cos(beta)*sin(gamma) +
3^(1/2)*LA*y*cos(alpha)*cos(gamma) - 3^(1/2)*LA*z*cos(beta)*sin(alpha) +
3^(1/2)*LA*x*cos(alpha)*sin(gamma) + (3*LA*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*LA*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 - (3^(1/2)*LA*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*LA*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*LA*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*LA*Rc*cos(beta)*sin(gamma))/2 - 3^(1/2)*LA*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*LA*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*LA*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*LA*x*cos(gamma)*sin(alpha)*sin(beta));

F22=(LA*b*cos(gamma)*sin(alpha) + LA*Rc*sin(alpha)*sin(gamma) - 2*LA*z*cos(alpha)*cos(beta) -
2*LA*y*cos(gamma)*sin(alpha) - 2*LA*x*sin(alpha)*sin(gamma) -
3^(1/2)*LA*b*sin(alpha)*sin(gamma) - LA*Rc*cos(alpha)*cos(gamma)*sin(beta) +
LA*b*cos(alpha)*sin(beta)*sin(gamma) + 2*LA*x*cos(alpha)*cos(gamma)*sin(beta) -
2*LA*y*cos(alpha)*sin(beta)*sin(gamma) + 3^(1/2)*LA*Rc*cos(gamma)*sin(alpha) +
3^(1/2)*LA*b*cos(alpha)*cos(gamma)*sin(beta) + 3^(1/2)*LA*Rc*cos(alpha)*sin(beta)*sin(gamma));

G22=LA^2-Lc^2 - b*y - Rc*x + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) -
3^(1/2)*Rc*y + 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 +
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 - (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 -
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 + (3*Ra*b*cos(alpha)*sin(gamma))/2 -
(Rc*b*cos(alpha)*sin(gamma))/2 + (Ra*b*cos(beta)*sin(gamma))/2 -
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) + b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) - b*z*cos(beta)*sin(alpha) + b*x*cos(alpha)*sin(gamma) -
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 - (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 + (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 +
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) - 3^(1/2)*Ra*z*cos(beta)*sin(alpha) +
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) - 3^(1/2)*b*x*cos(beta)*cos(gamma) +
3^(1/2)*b*y*cos(beta)*sin(gamma) + (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 -
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 +
b*x*cos(gamma)*sin(alpha)*sin(beta) - b*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 - 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
- (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta31
```

```
E31=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 - (3*La*b*cos(alpha)*sin(gamma))/2 -
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) -
3^(1/2)*La*y*cos(alpha)*cos(gamma) + 3^(1/2)*La*z*cos(beta)*sin(alpha) -
3^(1/2)*La*x*cos(alpha)*sin(gamma) - (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 - (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 + 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));

F31=(La*Rc*sin(alpha)*sin(gamma) - La*b*cos(gamma)*sin(alpha) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) -
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) -
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) - 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) +
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) - 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G31=-Lc^2 + b*y - Rc*x + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) +
3^(1/2)*Rc*y + 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 -
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 + (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 -
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 - (3*Ra*b*cos(alpha)*sin(gamma))/2 +
(Rc*b*cos(alpha)*sin(gamma))/2 - (Ra*b*cos(beta)*sin(gamma))/2 +
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) - b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) + b*z*cos(beta)*sin(alpha) - b*x*cos(alpha)*sin(gamma) -
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 - (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 - (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 -
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) + 3^(1/2)*Ra*z*cos(beta)*sin(alpha) -
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) - 3^(1/2)*b*x*cos(beta)*cos(gamma) +
3^(1/2)*b*y*cos(beta)*sin(gamma) - (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 -
b*x*cos(gamma)*sin(alpha)*sin(beta) + b*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 + 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
+ (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta32
E32=(2*LA*Ra + LA*z*sin(beta) - (3*LA*Rc*cos(alpha)*cos(gamma))/2 -
(LA*Rc*cos(beta)*cos(gamma))/2 + (3*LA*b*cos(alpha)*sin(gamma))/2 +
(LA*b*cos(beta)*sin(gamma))/2 + LA*x*cos(beta)*cos(gamma) - LA*y*cos(beta)*sin(gamma) -
3^(1/2)*LA*y*cos(alpha)*cos(gamma) + 3^(1/2)*LA*z*cos(beta)*sin(alpha) -
3^(1/2)*LA*x*cos(alpha)*sin(gamma) + (3*LA*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*LA*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 + (3^(1/2)*LA*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*LA*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*LA*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*LA*Rc*cos(beta)*sin(gamma))/2 + 3^(1/2)*LA*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*LA*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*LA*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*LA*x*cos(gamma)*sin(alpha)*sin(beta));

F32=(LA*b*cos(gamma)*sin(alpha) + LA*Rc*sin(alpha)*sin(gamma) - 2*LA*z*cos(alpha)*cos(beta) -
2*LA*y*cos(gamma)*sin(alpha) - 2*LA*x*sin(alpha)*sin(gamma) +
3^(1/2)*LA*b*sin(alpha)*sin(gamma) - LA*Rc*cos(alpha)*cos(gamma)*sin(beta) +
LA*b*cos(alpha)*sin(beta)*sin(gamma) + 2*LA*x*cos(alpha)*cos(gamma)*sin(beta) -
2*LA*y*cos(alpha)*sin(beta)*sin(gamma) - 3^(1/2)*LA*Rc*cos(gamma)*sin(alpha) -
3^(1/2)*LA*b*cos(alpha)*cos(gamma)*sin(beta) - 3^(1/2)*LA*Rc*cos(alpha)*sin(beta)*sin(gamma));

G32=La^2-Lc^2- b*y - Rc*x + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) +
3^(1/2)*Rc*y - 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 -
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 + (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 +
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 + (3*Ra*b*cos(alpha)*sin(gamma))/2 -
(Rc*b*cos(alpha)*sin(gamma))/2 + (Ra*b*cos(beta)*sin(gamma))/2 -
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) + b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) - b*z*cos(beta)*sin(alpha) + b*x*cos(alpha)*sin(gamma) +
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 + (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 - (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 -
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) + 3^(1/2)*Ra*z*cos(beta)*sin(alpha) -
```

```
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) + 3^(1/2)*b*x*cos(beta)*cos(gamma) -
3^(1/2)*b*y*cos(beta)*sin(gamma) + (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 -
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 +
b*x*cos(gamma)*sin(alpha)*sin(beta) - b*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 + 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
+ (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);


theta11=2*atan((-F11-sqrt(E11^2+F11^2-G11^2))/(G11-E11));
theta12=2*atan((-F12-sqrt(E12^2+F12^2-G12^2))/(G12-E12));
theta21=2*atan((-F21-sqrt(E21^2+F21^2-G21^2))/(G21-E21));
theta22=2*atan((-F22-sqrt(E22^2+F22^2-G22^2))/(G22-E22));
theta31=2*atan((-F31-sqrt(E31^2+F31^2-G31^2))/(G31-E31));
theta32=2*atan((-F32-sqrt(E32^2+F32^2-G32^2))/(G32-E32));

theta1=(theta11+theta12)/2;
theta2=(theta21+theta22)/2;
theta3=(theta31+theta32)/2;
dtheta1=(theta11-theta12)/2;
dtheta2=(theta21-theta22)/2;
dtheta3=(theta31-theta32)/2;

Jmoins1=jacobian([theta1,theta2,theta3,dtheta1,dtheta2,dtheta3]);

%Envoyer dans un fichier la jacobienne
save 'Jacobienneinverse.txt'
fid = fopen('Jacobiennefonctioninverse_V3.txt', 'w');

for i=1:6
    for j=1:6
        fprintf(fid, '%s ', Jmoins1(i,j));
    end
    fprintf(fid, ';');
end

fclose(fid);

end
```

Annexe 4 : Code Matlab du bloc cinématique inverse

```
function [dtheta1,dtheta2,dtheta3,dtheta4,dtheta5,dtheta6] =
fcn(dx,dy,dz,dalpha,dbeta,dgamma,x,y,z,alpha,beta,gamma)

Jmoins1=jacobienne_inverse(x,y,z,alpha,beta,gamma);
test=Jmoins1*[dx;dy;dz;dalpha;dbeta;dgamma];

dtheta1=test(1);
dtheta2=test(2);
dtheta3=test(3);
dtheta4=test(4);
dtheta5=test(5);
dtheta6=test(6);
end
```

Annexe 5 : Code Matlab du bloc cinématique directe

```
function [fx,fy,fz,falpha,fbeta,fgamma]  =
fcn(ftheta1,ftheta2,ftheta3,ftheta4,ftheta5,ftheta6,x,y,z,alpha,beta,gamma)

Jmoins1=jacobienne_inverse(x,y,z,alpha,beta,gamma);

J=inv(Jmoins1);

test=J*[ftheta1;ftheta2;ftheta3;ftheta4;ftheta5;ftheta6];

fx=test(1);
fy=test(2);
fz=test(3);
falpha=test(4);
fbeta=test(5);
fgamma=test(6);
end
```

Annexe 6 : Code Matlab du bloc géométrique inverse

```
function [theta1,theta2,theta3,dtheta1,dtheta2,dtheta3]=fcn(x,y,z,alpha,beta,gamma)

La=0.069;
Lc=0.33 ;
Ra=0.169;
Rc=0.08;
b=0.049;

%Pour theta11
E11=(2*La*Ra - 2*La*z*sin(beta) - 2*La*Rc*cos(beta)*cos(gamma) - 2*La*b*cos(beta)*sin(gamma) -
2*La*x*cos(beta)*cos(gamma) + 2*La*y*cos(beta)*sin(gamma));

F11=(2*La*b*cos(gamma)*sin(alpha) - 2*La*Rc*sin(alpha)*sin(gamma) -
2*La*z*cos(alpha)*cos(beta) - 2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) +
2*La*Rc*cos(alpha)*cos(gamma)*sin(beta) + 2*La*b*cos(alpha)*sin(beta)*sin(gamma) +
2*La*x*cos(alpha)*cos(gamma)*sin(beta) - 2*La*y*cos(alpha)*sin(beta)*sin(gamma));

G11=-Lc^2+2*Rc*x - 2*b*y + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 - 2*Ra*z*sin(beta) -
2*b^2*cos(alpha)*cos(gamma) - 2*Ra*Rc*cos(beta)*cos(gamma) +
2*b^2*sin(alpha)*sin(beta)*sin(gamma) + 2*Rc*b*cos(alpha)*sin(gamma) -
2*Ra*b*cos(beta)*sin(gamma) - 2*Ra*x*cos(beta)*cos(gamma) + 2*b*y*cos(alpha)*cos(gamma) +
2*Ra*y*cos(beta)*sin(gamma) - 2*b*z*cos(beta)*sin(alpha) + 2*b*x*cos(alpha)*sin(gamma) +
2*Rc*b*cos(gamma)*sin(alpha)*sin(beta) + 2*b*x*cos(gamma)*sin(alpha)*sin(beta) -
2*b*y*sin(alpha)*sin(beta)*sin(gamma);

%Pour theta12
E12=(2*La*Ra - 2*La*z*sin(beta) - 2*La*Rc*cos(beta)*cos(gamma) + 2*La*b*cos(beta)*sin(gamma) -
2*La*x*cos(beta)*cos(gamma) + 2*La*y*cos(beta)*sin(gamma));

F12=(2*La*Rc*cos(alpha)*cos(gamma)*sin(beta) - 2*La*Rc*sin(alpha)*sin(gamma) -
2*La*z*cos(alpha)*cos(beta) - 2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) -
2*La*b*cos(gamma)*sin(alpha) - 2*La*b*cos(alpha)*sin(beta)*sin(gamma) +
2*La*x*cos(alpha)*cos(gamma)*sin(beta) - 2*La*y*cos(alpha)*sin(beta)*sin(gamma));

G12=La^2-Lc^2 + 2*Rc*x + 2*b*y + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 - 2*Ra*z*sin(beta) -
2*b^2*cos(alpha)*cos(gamma) - 2*Ra*Rc*cos(beta)*cos(gamma) +
2*b^2*sin(alpha)*sin(beta)*sin(gamma) - 2*Rc*b*cos(alpha)*sin(gamma) +
2*Ra*b*cos(beta)*sin(gamma) - 2*Ra*x*cos(beta)*cos(gamma) - 2*b*y*cos(alpha)*cos(gamma) +
2*Ra*y*cos(beta)*sin(gamma) + 2*b*z*cos(beta)*sin(alpha) - 2*b*x*cos(alpha)*sin(gamma) -
2*Rc*b*cos(gamma)*sin(alpha)*sin(beta) - 2*b*x*cos(gamma)*sin(alpha)*sin(beta) +
2*b*y*sin(alpha)*sin(beta)*sin(gamma);


%Pour theta21
E21=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 - (3*La*b*cos(alpha)*sin(gamma))/2 -
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) +
3^(1/2)*La*y*cos(alpha)*cos(gamma) - 3^(1/2)*La*z*cos(beta)*sin(alpha) +
3^(1/2)*La*x*cos(alpha)*sin(gamma) - (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 + (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 - 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));

F21=(La*Rc*sin(alpha)*sin(gamma) - La*b*cos(gamma)*sin(alpha) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) +
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) -
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) + 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) -
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) + 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G21=-Lc^2+b*y - Rc*x + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) -
3^(1/2)*Rc*y - 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 +
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 - (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 +
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 - (3*Ra*b*cos(alpha)*sin(gamma))/2 +
```

```
(Rc*b*cos(alpha)*sin(gamma))/2 - (Ra*b*cos(beta)*sin(gamma))/2 +
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) - b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) + b*z*cos(beta)*sin(alpha) - b*x*cos(alpha)*sin(gamma) +
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 + (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 + (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 +
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) - 3^(1/2)*Ra*z*cos(beta)*sin(alpha) +
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) + 3^(1/2)*b*x*cos(beta)*cos(gamma) -
3^(1/2)*b*y*cos(beta)*sin(gamma) - (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 -
b*x*cos(gamma)*sin(alpha)*sin(beta) + b*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 - 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
- (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta22
E22=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 + (3*La*b*cos(alpha)*sin(gamma))/2 +
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) +
3^(1/2)*La*y*cos(alpha)*cos(gamma) - 3^(1/2)*La*z*cos(beta)*sin(alpha) +
3^(1/2)*La*x*cos(alpha)*sin(gamma) + (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 - (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 - 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));

F22=(La*b*cos(gamma)*sin(alpha) + La*Rc*sin(alpha)*sin(gamma) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) -
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) +
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) + 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) +
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) + 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G22=La^2-Lc^2 - b*y - Rc*x + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) -
3^(1/2)*Rc*y + 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 +
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 - (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 -
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 + (3*Ra*b*cos(alpha)*sin(gamma))/2 -
(Rc*b*cos(alpha)*sin(gamma))/2 + (Ra*b*cos(beta)*sin(gamma))/2 -
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) + b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) - b*z*cos(beta)*sin(alpha) + b*x*cos(alpha)*sin(gamma) -
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 - (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 -
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 + (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 +
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) - 3^(1/2)*Ra*z*cos(beta)*sin(alpha) +
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) - 3^(1/2)*b*x*cos(beta)*cos(gamma) +
3^(1/2)*b*y*cos(beta)*sin(gamma) + (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 -
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 +
b*x*cos(gamma)*sin(alpha)*sin(beta) - b*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 - 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
- (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta31
E31=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 - (3*La*b*cos(alpha)*sin(gamma))/2 -
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) -
3^(1/2)*La*y*cos(alpha)*cos(gamma) + 3^(1/2)*La*z*cos(beta)*sin(alpha) -
3^(1/2)*La*x*cos(alpha)*sin(gamma) - (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 - (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 + 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));
```

```
F31=(La*Rc*sin(alpha)*sin(gamma) - La*b*cos(gamma)*sin(alpha) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) -
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) -
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) - 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) +
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) - 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G31=-Lc^2 + b*y - Rc*x + La^2 + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) +
3^(1/2)*Rc*y + 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 -
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 + (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 -
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 - (3*Ra*b*cos(alpha)*sin(gamma))/2 +
(Rc*b*cos(alpha)*sin(gamma))/2 - (Ra*b*cos(beta)*sin(gamma))/2 +
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) - b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) + b*z*cos(beta)*sin(alpha) - b*x*cos(alpha)*sin(gamma) -
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 - (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 + (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 +
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 - (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 -
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) + 3^(1/2)*Ra*z*cos(beta)*sin(alpha) -
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) - 3^(1/2)*b*x*cos(beta)*cos(gamma) +
3^(1/2)*b*y*cos(beta)*sin(gamma) - (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 -
b*x*cos(gamma)*sin(alpha)*sin(beta) + b*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 + 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
+ (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 +
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);

%Pour theta32
E32=(2*La*Ra + La*z*sin(beta) - (3*La*Rc*cos(alpha)*cos(gamma))/2 -
(La*Rc*cos(beta)*cos(gamma))/2 + (3*La*b*cos(alpha)*sin(gamma))/2 +
(La*b*cos(beta)*sin(gamma))/2 + La*x*cos(beta)*cos(gamma) - La*y*cos(beta)*sin(gamma) -
3^(1/2)*La*y*cos(alpha)*cos(gamma) + 3^(1/2)*La*z*cos(beta)*sin(alpha) -
3^(1/2)*La*x*cos(alpha)*sin(gamma) + (3*La*b*cos(gamma)*sin(alpha)*sin(beta))/2 +
(3*La*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 + (3^(1/2)*La*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*La*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*La*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*La*Rc*cos(beta)*sin(gamma))/2 + 3^(1/2)*La*y*sin(alpha)*sin(beta)*sin(gamma) +
(3^(1/2)*La*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*La*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*La*x*cos(gamma)*sin(alpha)*sin(beta));

F32=(La*b*cos(gamma)*sin(alpha) + La*Rc*sin(alpha)*sin(gamma) - 2*La*z*cos(alpha)*cos(beta) -
2*La*y*cos(gamma)*sin(alpha) - 2*La*x*sin(alpha)*sin(gamma) +
3^(1/2)*La*b*sin(alpha)*sin(gamma) - La*Rc*cos(alpha)*cos(gamma)*sin(beta) +
La*b*cos(alpha)*sin(beta)*sin(gamma) + 2*La*x*cos(alpha)*cos(gamma)*sin(beta) -
2*La*y*cos(alpha)*sin(beta)*sin(gamma) - 3^(1/2)*La*Rc*cos(gamma)*sin(alpha) -
3^(1/2)*La*b*cos(alpha)*cos(gamma)*sin(beta) - 3^(1/2)*La*Rc*cos(alpha)*sin(beta)*sin(gamma));

G32=La^2-Lc^2- b*y - Rc*x + Ra^2 + Rc^2 + 2*b^2 + x^2 + y^2 + z^2 + Ra*z*sin(beta) +
3^(1/2)*Rc*y - 3^(1/2)*b*x - (b^2*cos(alpha)*cos(gamma))/2 - (3*b^2*cos(beta)*cos(gamma))/2 -
(3^(1/2)*b^2*cos(alpha)*sin(gamma))/2 + (3^(1/2)*b^2*cos(beta)*sin(gamma))/2 +
3^(1/2)*b*z*sin(beta) - (3*Ra*Rc*cos(alpha)*cos(gamma))/2 - (Ra*Rc*cos(beta)*cos(gamma))/2 +
(b^2*sin(alpha)*sin(beta)*sin(gamma))/2 + (3*Ra*b*cos(alpha)*sin(gamma))/2 -
(Rc*b*cos(alpha)*sin(gamma))/2 + (Ra*b*cos(beta)*sin(gamma))/2 -
(3*Rc*b*cos(beta)*sin(gamma))/2 + Ra*x*cos(beta)*cos(gamma) + b*y*cos(alpha)*cos(gamma) -
Ra*y*cos(beta)*sin(gamma) - b*z*cos(beta)*sin(alpha) + b*x*cos(alpha)*sin(gamma) +
(3^(1/2)*Ra*b*cos(alpha)*cos(gamma))/2 + (3^(1/2)*Rc*b*cos(alpha)*cos(gamma))/2 +
(3^(1/2)*Ra*Rc*cos(alpha)*sin(gamma))/2 - (3^(1/2)*Ra*b*cos(beta)*cos(gamma))/2 -
(3^(1/2)*Rc*b*cos(beta)*cos(gamma))/2 - (3^(1/2)*Ra*Rc*cos(beta)*sin(gamma))/2 -
3^(1/2)*Ra*y*cos(alpha)*cos(gamma) + 3^(1/2)*Ra*z*cos(beta)*sin(alpha) -
3^(1/2)*Ra*x*cos(alpha)*sin(gamma) + 3^(1/2)*b*x*cos(beta)*cos(gamma) -
3^(1/2)*b*y*cos(beta)*sin(gamma) + (3*Ra*b*cos(gamma)*sin(alpha)*sin(beta))/2 -
(Rc*b*cos(gamma)*sin(alpha)*sin(beta))/2 + (3*Ra*Rc*sin(alpha)*sin(beta)*sin(gamma))/2 +
b*x*cos(gamma)*sin(alpha)*sin(beta) - b*y*sin(alpha)*sin(beta)*sin(gamma) -
(3^(1/2)*b^2*cos(gamma)*sin(alpha)*sin(beta))/2 + 3^(1/2)*Ra*y*sin(alpha)*sin(beta)*sin(gamma)
+ (3^(1/2)*Ra*Rc*cos(gamma)*sin(alpha)*sin(beta))/2 -
(3^(1/2)*Ra*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
(3^(1/2)*Rc*b*sin(alpha)*sin(beta)*sin(gamma))/2 -
3^(1/2)*Ra*x*cos(gamma)*sin(alpha)*sin(beta);


theta11=2*atan((-F11-sqrt(E11^2+F11^2-G11^2))/(G11-E11));
```

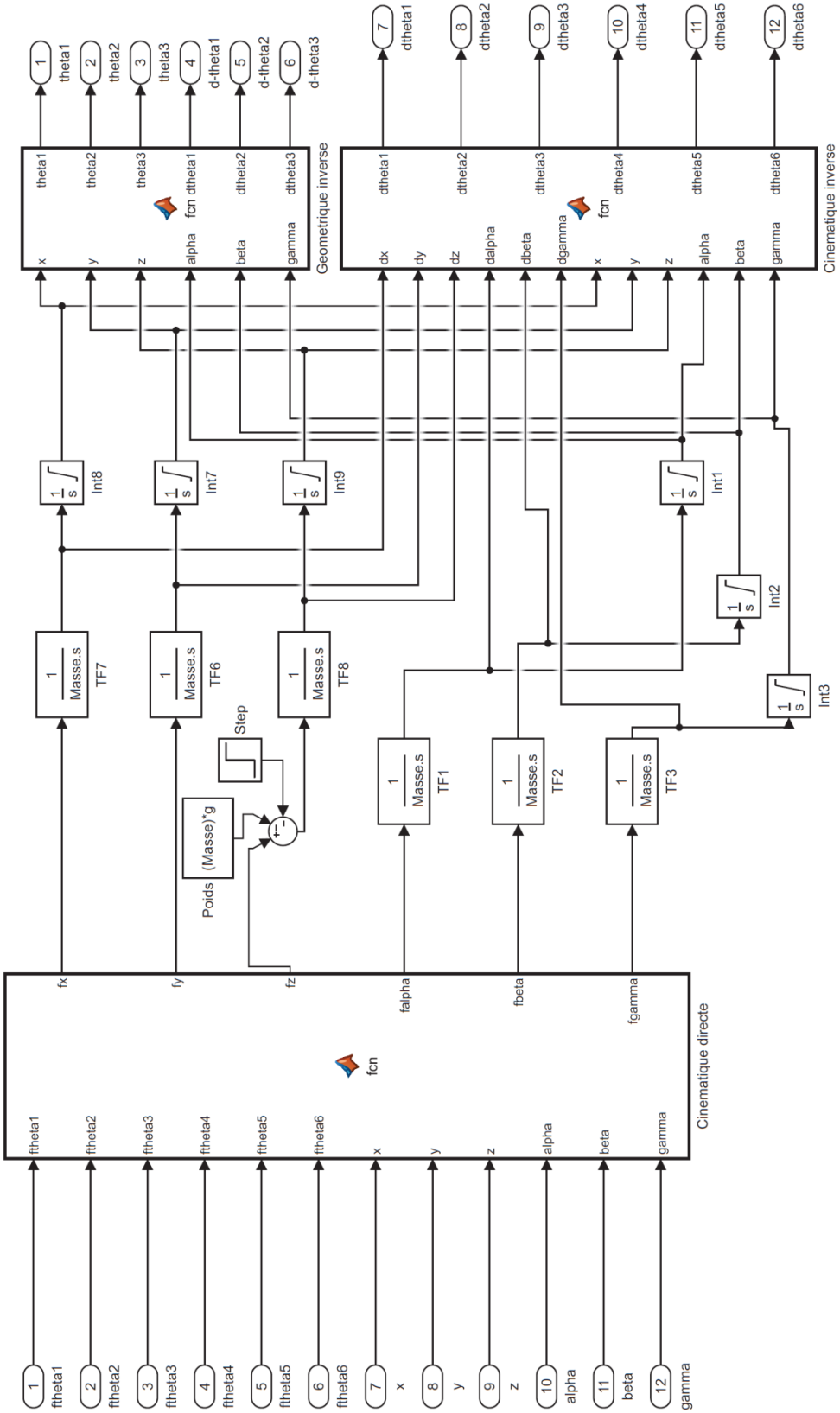
```
theta12=2*atan((-F12-sqrt(E12^2+F12^2-G12^2))/(G12-E12));
theta21=2*atan((-F21-sqrt(E21^2+F21^2-G21^2))/(G21-E21));
theta22=2*atan((-F22-sqrt(E22^2+F22^2-G22^2))/(G22-E22));
theta31=2*atan((-F31-sqrt(E31^2+F31^2-G31^2))/(G31-E31));
theta32=2*atan((-F32-sqrt(E32^2+F32^2-G32^2))/(G32-E32));

theta1=(theta11+theta12)/2;
theta2=(theta21+theta22)/2;
theta3=(theta31+theta32)/2;

dtheta1=(theta11-theta12)/2;
dtheta2=(theta21-theta22)/2;
dtheta3=(theta31-theta32)/2;
end
```

Annexe 7 : Code Matlab du bloc géométrique directe

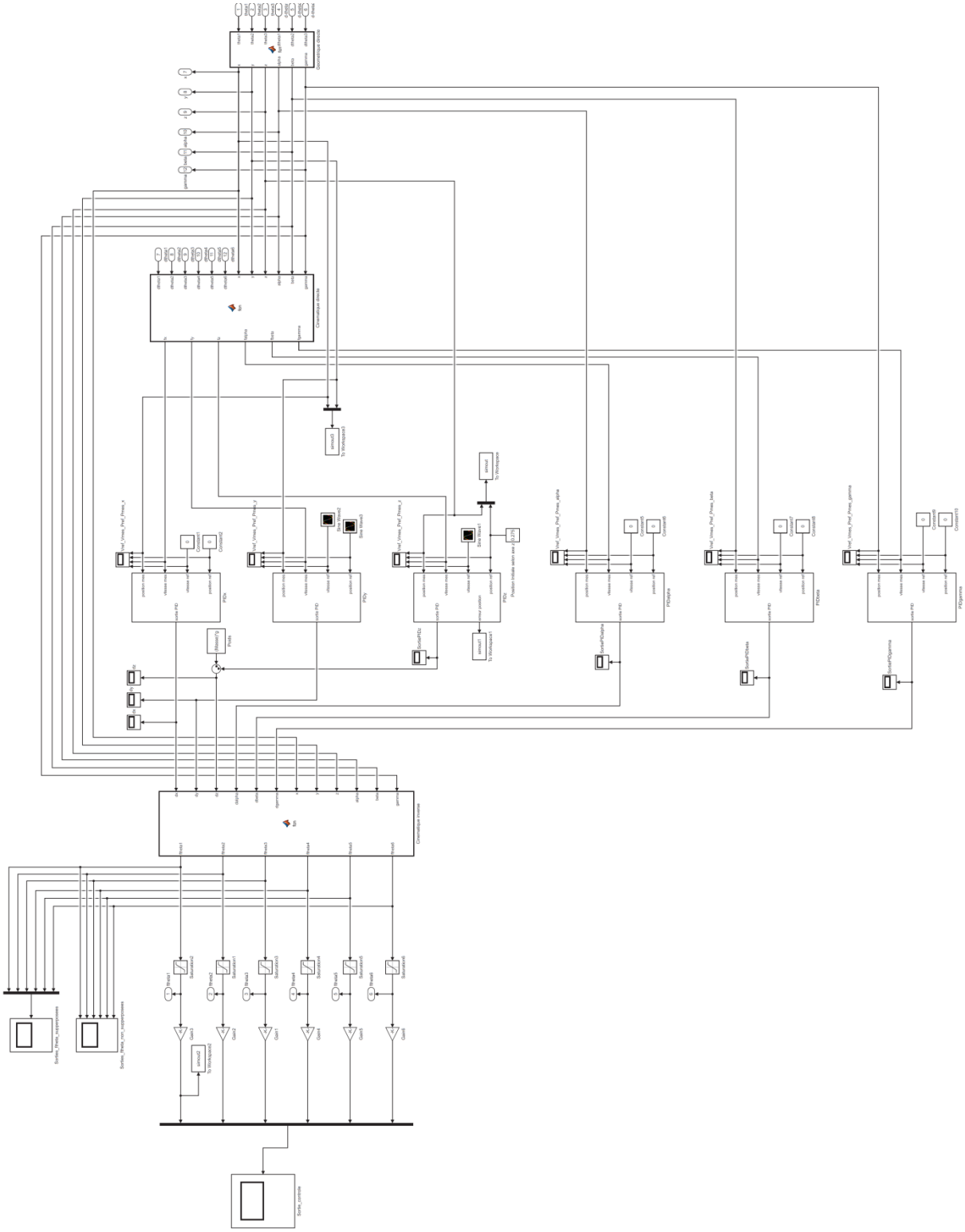
Le code ci-dessous ne prend pas en compte les rotations :

```
function [x,y,z,alpha,beta,gamma] = fcn(theta1,theta2,theta3,dtheta1,dtheta2,dtheta3)

La=0.069;
Lc=0.33 ;
Ra=0.169;
Rc=0.08;
b=0.049;

%Centre des spheres

    sphere1=[Ra - Rc + La*cos(theta1) ; 0 ; La*sin(theta1)];
    sphere2=[Rc/2 - Ra/2 - (La*cos(theta2))/2 ; (3^(1/2)*Rc)/2 - (3^(1/2)*(Ra +
La*cos(theta2)))/2 ; La*sin(theta2)];
    sphere3=[Rc/2 - Ra/2 - (La*cos(theta3))/2 ; (3^(1/2)*(Ra + La*cos(theta3)))/2 -
(3^(1/2)*Rc)/2 ; La*sin(theta3)];

%Determination de l'intersection des 3 spheres de centres sphere1,2,3 et de
%rayon Lc

    dists1s2 = sphere2-sphere1;
    e_x = dists1s2/norm(dists1s2); %u1
    dists3s1 =sphere3-sphere1;
    i = dot(e_x,dists3s1); %scalaire dist et vecteur unitaire
    temp3 = dists3s1 - i*e_x;
    e_y =temp3/norm(temp3); %u2
    e_z = cross(e_x,e_y); %u3
    d = norm(sphere2-sphere1);
    j= dot(e_y,dists3s1);
    x = d / 2;
    y = (-2*i*x + i*i + j*j) / (2*j);
    temp4 = Lc*Lc - x*x - y*y;
    z =sqrt(temp4);
    p_1 = +sphere1+x*e_x + y*e_y -z*e_z;
    x=p_1(1);
    y=p_1(2);
    z=p_1(3);

    alpha=0;
    beta=0;
    gamma=0;
end
```

Annexe 8 : Sous-système robot (avec prise en compte rotations)



Annexe 9 : Sous-système contrôle (avec prise en compte rotations)



Bibliographie

Valentin CATTIAU, *Rapport de projet : Contrôle d'un robot parallèle à 6ddl*, 2016.

Wisama KHALIL et Etienne DOMBRE, *Modélisation, identification et commande des robots*, 2001.

Sites internet utilisés :

https://www.researchgate.net/

http://uksim.info/ems2015/data/0206a232.pdf

http://fab.cba.mit.edu/classes/863.15/section.CBA/people/Spielberg/Rostock_Delta_Kinematics_3.pdf

http://www.cim.mcgill.ca/~paul/clavdelt.pdf

http://etsmtl.ca/Professeurs/ibonev/documents/pdf/ThesisBonev.pdf

http://www.ohio.edu/people/williar4/html/pdf/DeltaKin.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.2249&rep=rep1&type=pdf

[1] Lien du wiki de suivi de projet pour les vidéos en partie "Semaine 5" : http://projets-imasc.plil.net/mediawiki/index.php?title=IMA4_2016/2017_P34

[2] Valentin CATTIAU, *Rapport de projet : Contrôle d'un robot parallèle à 6ddl*, 2016.