CentraleSupélec

Projet Long

Rapport d'activité de la Séquence 7

Mme. MAKAROV

Karoline CARVALHO BÜRGER
Tiago DE JESUS RODRIGUES
Rafael ELLER CRUZ
Rafael Accácio NOGUEIRA

27 avril 2017

Table des matières

1 Introduction

Vu par l'extérieur, un manipulateur polyvalent pourrait sembler une technologie plutôt inoffensive, pas très différent d'un tapis roulant ou d'une perceuse, mais l'aspect innovative de ce type de machine est précisément sa polyvalence. Étant reprogrammable, le manipulateur n'est pas limité à une utilisation particulière, ce qui veut dire que le fabricant n'a pas besoin de traiter des considérations particulières d'une industrie donnée, mais seulement avec des aspects généraux (vitesse maximale, couple maximale, déplacement maximum, etc). Le but de ce travail est précisément d'explorer cet aspect polyvalent d'un manipulateur en proposant un système qui permet l'utilisateur de commander la position de l'extrémité du manipulateur e ses trajectoires, ensuite on proposera une application simple pour ce système, dont le but est de démontrer son fonctionnement et évaluer sa performance. Le système complet est schématisé de forme simplifiée ci-dessous :

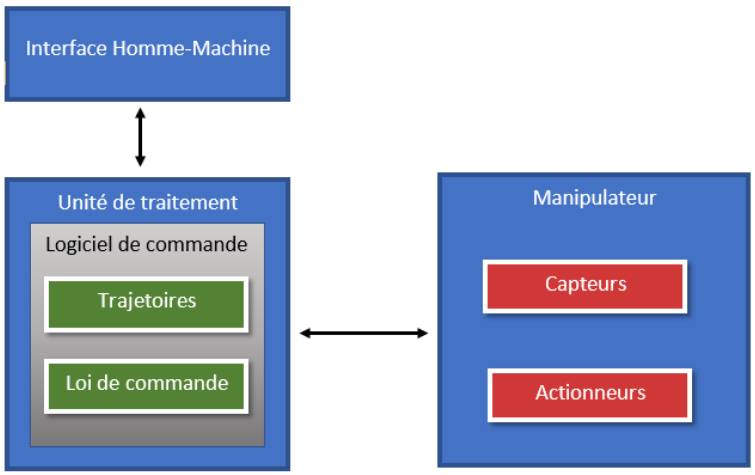

Figure 1 – Schéma simplifié du système.

2 Objectif

L'objectif de notre projet est de concevoir un système de commande pour un robot KUKA qui permet de placer son outil dans n'importe quelle coordonnée accessible pour le robot, en suivant une trajectoire pré-fixée, soit linéaire ou circulaire. Les coordonnées peuvent être fournies en position angulaire des articulations du robot (coordonnées angulaires) ou en position et orientation de l'outil du robot par rapport à sa base (coordonnées opérationnelles).

On développera aussi une application qui fait usage de ce système de commande pour permettre à deux personnes de jouer le Jeu du Morpion.

Ce rapport a pour but de montrer la planification de nos activités pour la séquence 8, ainsi que la façon dont on veut réaliser chaque tâche et les résultats qu'on a déjà obtenus.

3 Division du Travail

On souhaite que le produit final de ce travail soit un jeu de Morpion déployé dans le robot KUKA youBot et commande par une interface graphique, comme on a déjà décrit ci-dessus.

Sur base de ces objectifs, on peut séparer le travail en deux phases différentes que seront développés de manière séquentielle :

— Phase 1 : à propos du système robotique dans un environnement de simulation, lequel on doit traiter et solutionner les problèmes de commande du système ;

— Phase 2 : à propos de la mise en oeuvre du jeu en environnement physique, lequel on doit construire les différentes interfaces du système : robot $\leftrightarrow$ PC et PC $\leftrightarrow$ utilisateur.

Les actions et les sous-produits de la première phase du projet seront séparés en deux structures et développés en parallèle : une concernant l'étude et génération de la trajectoire exécuté par l'effecteur du robot (structure A) et l'autre concernant l'étude et mise en oeuvre de l'asservissement de l'effecteur (structure B).

Après la conclusion de chaque structure, on doit faire l'intégration des parties qui ont déjà été développés. Ainsi, on prévoit observer dans l'environnement de simulation Matlab l'effectuer réaliser une trajectoire stipulée.

De la même manière, la deuxième phase sera séparé en deux structures : une concernant la communication du robot avec l'ordinateur (structure A) et l'autre concernant la création de l'interface graphique de communication de l'ordinateur avec l'utilisateur (structure B).

De la répartition du projet en deux phases bien définies, on a pu séparer le projet en deux différentes domaines : d'automatique et d'informatique. Ainsi, lorsqu'on complète la première phase à propos de la problématique de contrôle, on aura déjà obtenu résultats bien satisfaisantes référents les objectifs d'apprentissage de ce projet. Le diagramme en arbre suivant décrit les différentes phases et structures qu'on a réparti le projet :

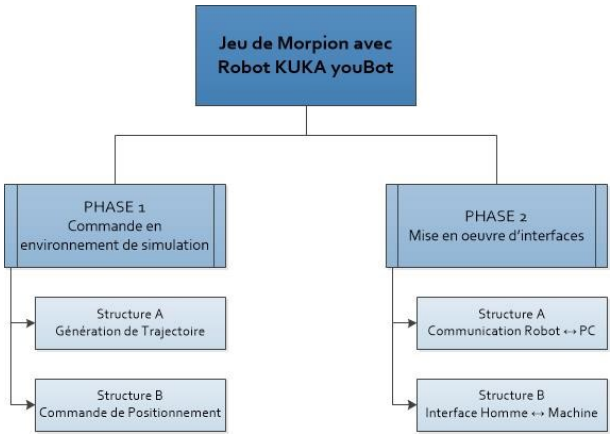

Figure 2 – Diagramme en arbre de la structure de découpage du projet

3.1 Définition des tâches à entreprendre

Ci-dessous, on a défini les tâches à entreprendre en chaque structure de découpage du projet. Dans la section 4 on a décrit les tâches de façon plus détaillée.

3.2 Commande Angulaire et Cartésienne

En réalité, on a déjà commencé à travailler à cette partie du projet dans la séquence 7, car pour développer une loi de commande pour un système, on a besoin de premièrement le modéliser. Comme le modèle du robot nous a été fourni dans un fichier Matlab, on a employé du temps en le comprendre. Alors, la tâche 1.B.1 qui est décrite dans cette section ont été réalisées par tous les membres de l'équipe, et on a déjà fini jusqu'à la tâche 1.B.4. On a réservé 28 créneaux pour les tâches de cette structure, dont 14 sont compris dans les activités de la séquence 7, qui étaient nécessaires pour la suite de cette partie du projet.

Dans la suite, la division de cette partie du travail en tâches est décrit, ainsi que le temps qu'on a l'intention de consacrer à chaque tâche.

— 1.B.1 : Étude à propos de la modélisation des systèmes robotiques : 6 créneaux ;
— 1.B.2 : Compréhension du modèle Matlab du robot fourni : 2 créneaux
— 1.B.3 : Étude et développement des techniques de commande du robot dans l'espace des articulations et choix d'une méthode de commande : 4 créneaux ;
— 1.B.4 : Mis en oeuvre de la loi de commande du robot dans l'espace des articulations dans le Simulink et tests : 2 créneaux ;
— 1.B.5 : Étude et développement des techniques de commande du robot dans l'espace opérationnel et choix d'une méthode de commande : 2 créneaux ;
— 1.B.6 : Mis en oeuvre de la loi de commande du robot dans l'espace opérationnel dans le Simulink et tests : 4 créneaux ;
— 1.B.7 : Optimisation des lois de commande : 2 créneaux ;
— 1.B.8 : Intégration avec le module de génération des trajectoires : 3 créneaux ;
— 1.B.9 : Intégration avec le robot : 3 créneaux ;

3.3 Trajectoires

A fin de organiser et simplifier le travail la création de trajectoires, on a défini deux types de trajectoire, linéaire et arc de cercle. Comme entrée de la fonction sera le type de trajectoire, un point intermédiaire (pour des arcs de cercle) et le point final. La création de chaque un de ces types de trajectoire peut être diviser en quelque parties

Pour les trajectoires linéaires

— 1.A.1 Création d'une trajectoire linéaire entre deux points
— 1.A.2 Calcul du polynôme interpolateur

Pour les trajectoires Circulaires :

— 1.A.3 Calcul du cercle
— 1.A.4 Paramétrisation du cercle
— 1.A.5 Calcul du angle du arc
— 1.A.6 Calcul du polynôme interpolateur

3.4 Communication avec le robot

Cette partie est concerné par l'intégration entre le programme de commande et l'interface du robot Kuka. Donc la première tâche à être accomplie c'est l'étude de cette interface de sorte qu'on puisse définir des tâches et planifier les étapes de cette phase.

— 2.A.0 Étude de l'interface et planification

3.5 Interface Homme ↔ Machine

Comme point de départ du développement de l'IHM, on aura l'obtention et l'utilisation de la fonction concise obtenue en tant que produit final de l'intégration effectuée à la fin de la phase 1 (tâches 1.B.8 et 1.B.9).

Dans un premier temps, on doit définir les caractéristiques, les contraintes et les dimensions de l'environnement de travail de l'effectuer (tableau du jeu). Alors, on doit créer les fonctions qui exécuteront les deux mouvements que le robot réalisera : "X" et "O". Après, on doit définir les paramètres d'entrée et les données nécessaires pour le développement de l'interface et aussi la création et l'administration des objets graphiques de l'interface. Enfin, on va faire des tests, corrections nécessaires, améliorations dans l'interface et l'intégration de ses fonctionnalités avec d'autres produits finaux du projet.

Ensuite, on a ordonné ces tâches et leurs temps estimé pour l'exécution de chaque activité :

— 2.B.0 : Fonction concise qui détermine le paramètre d'entrée du robot (couple) à partir de la position finale et le type de trajectoire désirée (obtenue dans les tâches 1.B18 et 1.B.9) : 6 créneaux ;
— 2.B.1 : Définition de les caractéristiques et dimensions d'environnement de travail (tableau du jeu) : 2 créneaux ;
— 2.B.2 : Définition et création de la structure de données et paramètres nécessaires pour l'IHM : 2 créneaux ;
— 2.B.3 : Création des fonctions de dessin "X" et "O" : 2 créneaux ;
— 2.B.4 : Définition de la disposition graphique et des objets de l'interface : 2 créneaux ;
— 2.B.5 : L'insertion des éléments dans l'interface : 2 créneaux ;
— 2.B.6 : Création et gestion des événements associés à les objets : 2 créneaux ;
— 2.B.7 : Tests, dépannage et intégration : 3 créneaux

Ainsi, on prévoit 6 créneaux pour la réalisation de la tâche 2.B.0 qui on doit obtenir comme résultat de l'intégration des activités dans la première phase du projet et plus 15 créneaux pour l'exécution des autres tâches de cette structure.

3.6 Désignation des tâches

À partir du découpage et définition de las différentes structures du projet, on a défini que les membres Rafael Accacio et Rafael Eller seront responsables de l'exécution de la structure A du projet dans les deux phases, tandis que les membres Karoline et Tiago seront responsables de la structure B. Pour faire cette désignation des tâches, on a considéré les jours disponibles de projet (série) de chaque membre, afin que dans les jours lesquels seulement deux membres seront au laboratoire, chacun travaille en structures différentes. Ainsi, on prévoit obtenir une progression plus rapide et égal pour chaque structure.

Cette organisation n'empêche pas les membres d'aider, discuter et exécuter différentes activités prévues, en prenant en compte la conclusion des tâches respectives. De plus, au fin de chaque phase on prévoit l'intégration des structures, ainsi la relation et le partage d'informations entre l'équipe sera considérée pour l'obtention de bons résultats.

3.7 Calendrier du projet

Afin d'analyser et gérer le progrès de l'équipe et l'exécution des différentes tâches, on a fait l'estimation de la durée d'exécution des activités nécessaires pour la conclusion du projet. La table suivante décrit le calendrier de réalisation du projet :

Identificateur d'activité	Description de l'activité	Unités (1h30)
1.A.1	Trajectoire linéaire entre deux point	2
1.A.2	Calcul du polynôme interpolateur (cas linéaire)	2
1.A.3	Calcul du cercle	2
1.A.4	Paramétrisation le cercle	2
1.A.5	Calcul du angle du arc	2
1.A.6	Calcul du polynôme interpolateur (cas cercle)	2
1.B.1	Étude: modélisation systèmes robotiques	6
1.B.2	Compréhension modèle Matlab du robot	2
1.B.3	Étude: commande espace des articulations	4
1.B.4	Éxecution: commande espace des articulations	2
1.B.5	Étude: commande espace opérationnel	2
1.B.6	Éxecution: commande espace opérationnel	4
1.B.7	Optimisation des lois de commande	2
1.B.8	Intégration avec générateur des trajectoires	3
1.B.9	Intégration avec le robot	3
2.A.0	Étude de l'interface et planification	3
2.B.0	Fonction concise des trajectoires et commande	6
2.B.1	Définition d'environnement de travail (tableau)	2
2.B.2	Création des fonctions "X" et "O"	2
2.B.3	Définition de la disposition graphique d'IHM	2
2.B.4	Définition de données d'entrée nécessaires IHM	1
2.B.5	L'insertion des objects graphiques	3
2.B.6	Création et gestion d'événements	2
2.B.7	Tests, dépannage et intégration	3

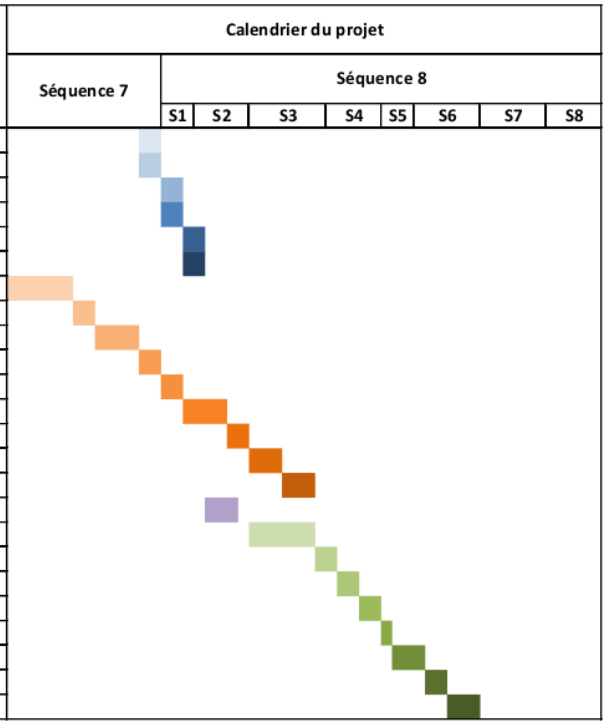

Figure 3 – Calendrier de réalisation du projet.

Les unités du temps sont considérés en créneaux, dans lequel chaque créneau a 1h30 de durée. Les jours dans lesquels seulement moitié de l'équipe sera présente dans le laboratoire, on a considéré la moitié du temps de chaque créneaux. On aura 40 créneaux pour la séquence 8 et 54 créneaux au total pour la conclusion du projet.

4 Méthodologie

4.1 Commande Angulaire et Cartésienne

Avant de commencer à développer le système de commande pour le robot, on doit comprendre son modèle dynamique. L'idée principale est de contrôler la position de l'outil du robot par le couple mécanique développer dans chaque articulation. Alors, la relation entre ces deux grandeurs, avec la position de l'outil

décrite comme des angles de chaque articulation du robot, est le modèle dynamique direct :

$$\boldsymbol{\Gamma} = \boldsymbol{A}(\boldsymbol{q}) * \ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{1}$$

D'où $\boldsymbol{\Gamma}$ le vecteur des couples nécessaires dans chaque articulation du robot, q le vecteur des positions angulaires de chaque articulation du robot, $\boldsymbol{A}(\boldsymbol{q})$ la matrice d'inertie du robot et $\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ les couples dissipatives du robot (Coriollis, friction et gravitationnel).

Pour mettre en œuvre le système de commande du robot, on se basera sur l'équation précédent et on développera les lois de commande premièrement dans l'espace des articulations et ensuite dans l'espace opérationnel.

4.1.1 Commande Angulaire

On commencera pour la commande du robot dans l'espace des articulations pour deux raisons différentes : il est plus facile d'être mis en oeuvre, alors on développe le projet dans un ordre de difficulté croissant ; on n'a pas besoin de relations intermédiaires pour convertir du modèle angulaire au modèle opérationnel (modèle géométrique direct plus jacobien), donc la performance obtenu est due uniquement au système de commande et on peut l'optimiser et l'utiliser comme base pour générer le système de commande dans l'espace opérationnel. On a utilisé deux approches différentes pour générer les lois de commande pour le robot, et on a les comparé pour choisir le plus efficace.

Dans la première approche, il s'agit de considérer que chaque articulation du robot est découplé des autres. Donc on peut écrire pour chaque articulation :

$$\Gamma_i = a_i * \ddot{q}_i + Fv_i * \dot{q}_i + \gamma_i \tag{2}$$

D'où a_i est la magnitute maximale de l'élément (i,i) de la matrice d'inertie, Fv_i est le coefficient de friction par rapport à l'articulation i et γ_i est une perturbation (due, par exemple, au couple gravitationnel).

Si on a le modèle dynamique du robot, on peut trouver les paramètres précédents et contrôler séparément chaque articulation du robot en utilisant un PID par exemple. Dans ce cas, la fonction de transfert qu'on obtiendra entre la position de référence et la vraie position du robot pour chaque articulation est :

$$H_i(s) = \frac{Kd_i s^2 + Kp_i s + Ki_i}{a_i s^3 + (Kd_i + Fv_i)s^2 + Kp_i s + Ki_i} \tag{3}$$

Alors, l'équation caractéristique est $d(s) = a_i s^3 + (Kd_i + Fv_i)s^2 + Kp_i s + Ki_i$, et on peut choisir Kp, Ki e Kd pour déterminer la dynamique du robot en boucle fermée. On veut que le dépassement de la réponse indicielle soit nulle, alors on a besoin que $d(s) = a_i(s + \omega_i)^3$, ω choisi de façon à contrôler le temps de réponse du système. Cependant, il y a un compromis dans le choix de ω, parce que s'il est petit, le système sera trop lent, mais s'il est trop gros, on peut arriver à la fréquence de résonance du système. Alors, on peut calculer Kp, Ki et Kd selon les formules suivantes. On a aussi le schéma de commande du robot obtenu sur simulink.

$$Kp_i = 3a_i(\omega_i)^2 \tag{4a}$$

$$Kd_i = 3a_i(\omega_i) - Fv_i \tag{4b}$$

$$Ki_i = a_i(\omega_i)^3 \tag{4c}$$

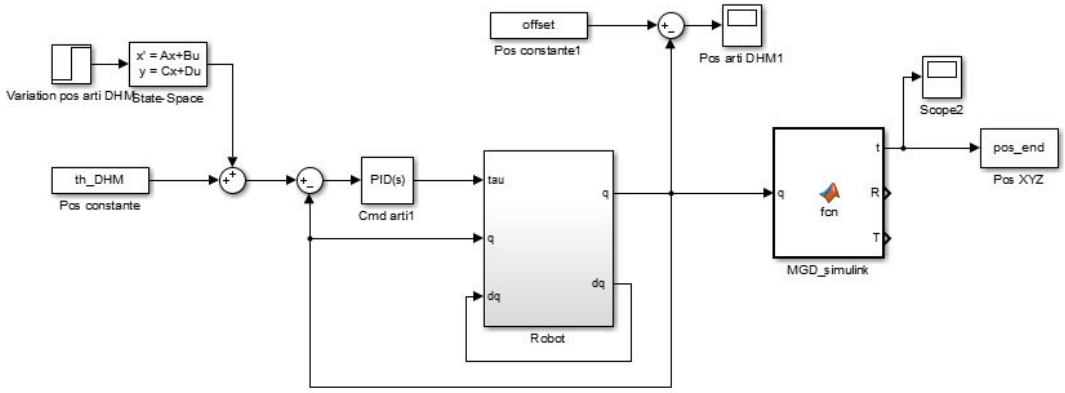

Figure 4 – Schéma de commande angulaire PID

La deuxième approche de commande pour le robot est la technique de compensation des non-linéarités. Il s'agit d'utiliser la matrice d'inertie $\hat{\boldsymbol{A}}(\boldsymbol{q})$ et le vecteur des couples $\hat{\boldsymbol{H}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ estimées pour déterminer le couple nécessaire pour le robot. Alors, on utilisera la relation :

$$\boldsymbol{\Gamma} = \hat{\boldsymbol{A}}(\boldsymbol{q}) * \boldsymbol{w}(t) + \hat{\boldsymbol{H}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{5}$$

Si $\hat{\boldsymbol{A}}$ et $\hat{\boldsymbol{H}}$ modélisent bien le modèle dynamique du robot, on peut choisir $\boldsymbol{w}(t) = \ddot{\boldsymbol{q}}$ et cela correspond exactement au modèle dynamique direct du robot. Avec cette approche, on peut calculer le couple nécessaire avant de l'alimenter dans le robot et de cette façon on peut réduire les fluctuations dans la réponse dues aux non-linéarité du modèle du robot. Cette technique rend plus facile aussi pour mettre en oeuvre la loi de commande. On définit l'erreur de position comme la différence entre la position désirée et la vraie position du robot ($\boldsymbol{e} = \boldsymbol{q_d} - \boldsymbol{q}$), et si on utilise un PID pour le contrôleur :

$$\boldsymbol{w}(t) = \ddot{\boldsymbol{q}}_d + \boldsymbol{Kp}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{Kd}(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{Ki}\int_0^t (\boldsymbol{q}_d - \boldsymbol{q})d\tau = \ddot{\boldsymbol{q}} \tag{6}$$

avec $\boldsymbol{Kp}$, $\boldsymbol{Kd}$ et $\boldsymbol{Ki}$ matrices diagonales.

On peut développer dans le domaine de Laplace :

$$\boldsymbol{w}(t) = \ddot{\boldsymbol{q}}_d + \boldsymbol{Kp}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{Kd}(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{Ki}\int_0^t (\boldsymbol{q}_d - \boldsymbol{q})d\tau$$

$$s^3\boldsymbol{E}(s) + s^2\boldsymbol{Kd} * \boldsymbol{E}(s) + s * \boldsymbol{Kp} * \boldsymbol{E}(s) + \boldsymbol{Ki} * \boldsymbol{E}(s) = s^2 * \boldsymbol{e}(0) + s * \boldsymbol{e}(0) + \boldsymbol{e}(0)$$

Alors, pour chaque articulation $E_i(s) = \dfrac{s^2 * e_i(0) + s * (e_i(0) + \dot{e}_i(0))}{s^3 + Kd_i * s^2 + Kp_i * s + Ki_i}$

Comme il a été dit, on veut que le dépassement de la réponse indicielle soit nulle, et pour une pulsation de brisure ω_i, on a :

$$Kp_i = 3\omega_i^2 \tag{8a}$$

$$Kd_i = 3\omega_i \tag{8b}$$

$$Ki_i = \omega_i^3 \tag{8c}$$

Dans la figure suivante, on a le schéma de cette approche de commande angulaire qu'on a mis en oeuvre sur Simulink. Par rapport à la technique de commande du PID, on peut noter que cette technique permet, en plus, de fournir des consignes de vitesse et accélération et pas seulement de position, ce qui est nécessaire pour que le robot puisse suivre des trajectoires.

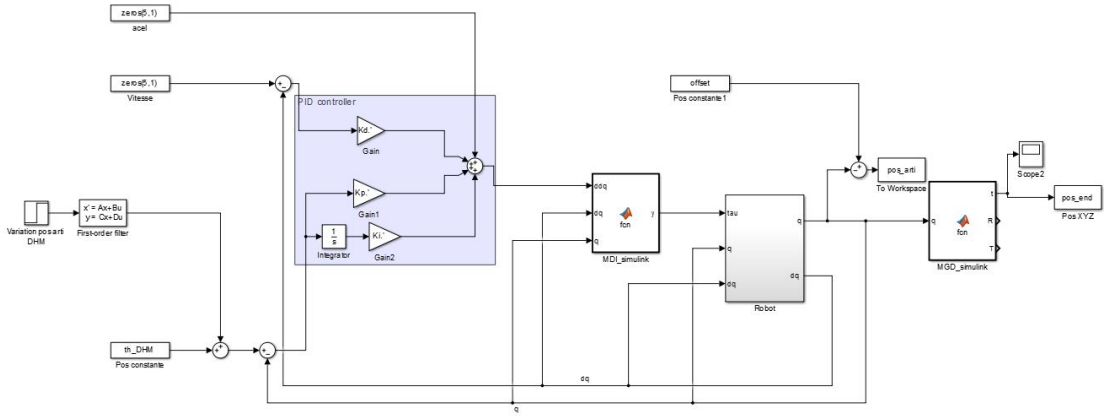

Figure 5 – Schéma de commande angulaire PID avec compensation des non-linéarités

4.1.2 Commande Cartésienne

Pour mettre en oeuvre les lois de commande du robot dans l'espace opérationnel, on utilisera les mêmes deux approches précédentes. La différence ici est qu'on doit utiliser des fonctions pour convertir les informations de l'espace angulaire à l'espace opérationnel. Cela peut être faite en utilisant le Jacobien.

Premièrement, pour obtenir les informations cartésiennes de la position de l'outil du robot avec les positions angulaires des articulations on doit utiliser le modèle géométrique directe du robot. Ce modèle est basé sur des changements de repère : on définit un repère dans chaque articulation du robot, un dans la base et un autre dans l'outil. L'idée est de créer une matrice homogène de transformation de repère (position et direction) d'une articulation par rapport à l'articulation précédente, qui dépend des paramètres constructifs du robot et des angles des articulations. Pour multiplier ces matrices, on peut obtenir la position et orientation de l'outil du robot par rapport au repère dans la base du robot comme fonction des angles des articulations. Pour obtenir la vitesse cartésienne de l'outil, on peut utiliser directement le Jacobien, selon la relation :

$$\dot{\boldsymbol{X}} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{9}$$

D'où $\boldsymbol{X}$ est la position de l'outil du robot dans l'espace opérationnel et J est le Jacobien.

Pour générer le couple nécessaire pour le robot avec des erreurs de position cartésienne, on peut utiliser le Jacobien transposée, et l'équation du PID devient maintenant :

$$\boldsymbol{\Gamma} = \boldsymbol{J}^T \left[\boldsymbol{Kp}(\boldsymbol{X}_d - \boldsymbol{X}) + \boldsymbol{Kd}(\dot{\boldsymbol{X}}_d - \dot{\boldsymbol{X}}) + \boldsymbol{Ki} \int_0^t (\boldsymbol{X}_d - \boldsymbol{X}) d\tau \right] \tag{10}$$

avec $\boldsymbol{Kp}$, $\boldsymbol{Kd}$ et $\boldsymbol{Ki}$ matrices diagonales.

Dans cette nouvelle configuration, l'équation qui relie la position désirée et la vraie position du robot n'est plus découplé, alors il est très compliqué de définir analytiquement les paramètres $\boldsymbol{Kp}$, $\boldsymbol{Kd}$ et $\boldsymbol{Ki}$ du contrôleur. Ce qu'on sait est que le terme intégral agit sur l'élimination de l'erreur statique, mais il réduit la stabilité du système, alors on peut définir un temps acceptable pour éliminer les perturbations en échelon et choisir le $\boldsymbol{Ki}$; le terme dérivatif est un terme prédictive, alors il agit en réduisant le dépassement de la réponse ; et le terme proportionnel est liée à l'intensité de l'action de contrôle par rapport à l'erreur, alors il peut régler la vitesse de réponse. Avec ses informations et les spécifications du système, on peut régler manuellement les paramètres $\boldsymbol{Kd}$ et $\boldsymbol{Kp}$. Mais on doit prendre en considération aussi que, si $\boldsymbol{Kd}$ et $\boldsymbol{Kp}$ sont trop gros, le système peut devenir instable.

Le schéma de commande pour le système est montré dans la figure suivante, générer avec le logiciel Simulink.

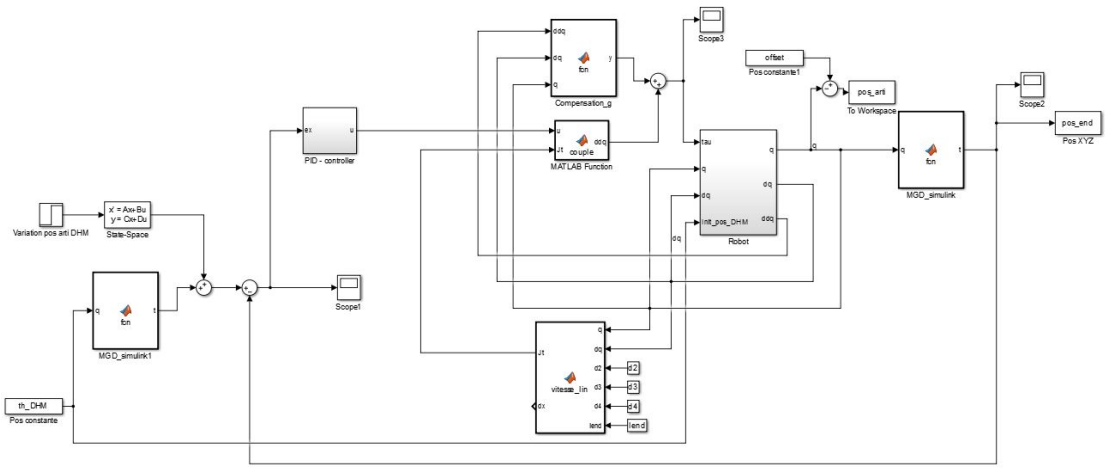

Figure 6 – Schéma de commande cartésienne PID

On peut utiliser les mêmes principes précédents pour générer la loi de commande dans l'espace opérationnel avec la technique de compensation des non-linéarités, sauf que la relation entre le couple et l'erreur de position cartésienne devient :

$$\boldsymbol{\Gamma} = \hat{\boldsymbol{A}}\boldsymbol{J}^{-1}(\boldsymbol{w}(t) - \dot{\boldsymbol{J}} * \dot{\boldsymbol{q}}) + \hat{\boldsymbol{H}} \tag{11}$$

Si le modèle du robot est bon, on aura $\boldsymbol{w}(t) = \dot{\boldsymbol{X}}$, et comme dans le cas équivalent dans l'espace des articulations, on peut écrire, si on utilise un contrôleur PID, et avec $\boldsymbol{e}_x = \boldsymbol{X}_d - \boldsymbol{X}$:

$$E_{x_i}(s) = \frac{s^2 * e_{x_i}(0) + s * (e_{x_i}(0) + \dot{e}_{x_i}(0))}{s^3 + Kd_i * s^2 + Kp_i * s + Ki_i} \tag{12}$$

Comme on veut que le dépassement de la réponse indicielle soit nulle, et pour une pulsation de brisure ω_i choisi maintenant pour la position et orientation de l'outil, on a à nouveau :

$$Kp_i = 3\omega_i^2 \tag{13a}$$

$$Kd_i = 3\omega_i \tag{13b}$$

$$Ki_i = \omega_i^3 \tag{13c}$$

Le schéma de commande pour le système est montré dans la figure suivante, générer avec le logiciel Simulink.

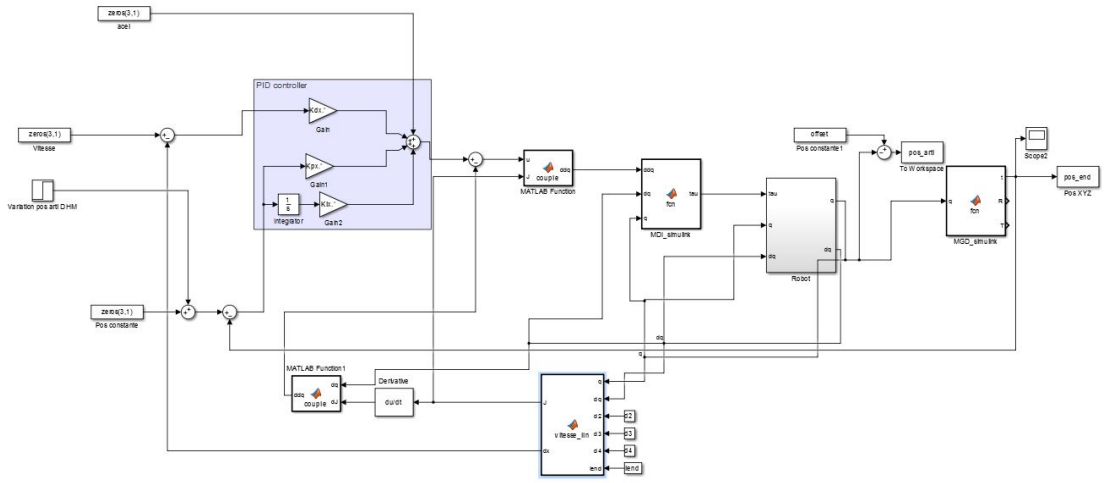

Figure 7 – Schéma de commande cartésienne PID avec compensation des non-linéarités

4.2 Trajectoire

1.A.1 Création d'une trajectoire linéaire entre deux points A partir du point initial et le point final, trouver une paramétrisation pour x, y et z de façon que le trajectoire soit linéaire.

1.A.2 Calcul du polynôme interpolateur Calculer deux polynômes interpolateurs pour que le début e le but de la trajectoires aient la vitesse et la accélération nulle, tournant le mouvement suffisamment lisse.

1.A.3 Calcul du cercle A partir du point initial, le point intermédiaire et le point final, trouver un cercle que passe par ces trois points

1.A.4 Paramétrisation du cercle Gérer une paramétrisation en X,Y et Z pour le cercle. Exemples : $X = 1 - cos(t)$ et $Y = sin(t)$, ou $X = t$ et $Y = \pm\sqrt{1 - t^2}$.

1.A.5 Calcul du angle du arc Calculer le angle formé entre le point initial, le centre du cercle et le point final, a fin de former juste un arc de cercle.

1.A.6 Calcul du polynôme interpolateur Calculer deux polynômes interpolateurs pour que le début e le but de la trajectoires aient la vitesse et la accélération nulle, tournant le mouvement suffisamment lisse.

4.3 2.B Interface Homme ↔ Machine

L'objectif de cette structure du travail est obtenir une interface graphique entre l'utilisateur et l'ordinateur qui permet deux joueurs jeter séquentiellement leurs mouvements qui seront réalisés par le robot sur le tableau réel. Ainsi, l'interface doit recevoir comme paramètre d'entrée le mouvement souhaité par l'utilisateur et remettre comme sortie la variable qui va agir sur le robot.

En prenant en compte cet objectif et les limitations de temps prévus pour l'exécution de ces tâches, on souhaite construire une interface fonctionnelle, intuitif et déployé de façon simple. Alors, on a choisi développer l'interface par programmation en langage Matlab et de manière modularisée, à savoir, la séparation des différentes tâches en différentes fonctions et avec haut niveau d'indépendance.

2.B.0 Fonction concise des trajectoires et commande On prévoit obtenir cette fonction comme produit d'intégration des résultats obtenus des structures A et B dans la première phase du projet (tâches 1.B.8 et 1.B.9). Comme produit d'étape de génération de trajectoire on attend obtenir un algorithme qui calcule le signal de consigne à partir de la position initiale de l'effecteur du robot, de la position finale et du type de trajectoire souhaitée. Tandis que dans l'étape d'asservissement, on prévoit obtenir le calcul du couple qui va agir au fil du temps dans le robot à partir de la consigne. Ainsi, en faisant l'intégration de ces deux algorithmes, on pourra obtenir une fonction qui calcule le couple à partir de la position finale souhaité et du type de trajectoire qui le robot va exécuter. Cette fonction sera appelée chaque fois que l'utilisateur, à partir des événements généré dans l'interface, demander un mouvement du robot.

2.B.1 Définition d'environnement de travail (tableau) Au début, on doit déterminer l'origine du repère outil, à savoir le repère de l'environnement de travail du robot. À partir de cette spécification, on doit définir les dimensions et divisions du tableau du jeu du Morpion. Ainsi, on pourra définir les distances parcourues par l'effecteur du robot quand il se déplace sur les carrés du tableau. Comme alternatif de cette détermination fixe de la taille du tableau, on peut aussi paramétrer la dimension du tableau. Ainsi, on laisse l'option pour l'utilisateur choisir la dimension souhaitée à partir de l'entrée de données de l'interface. Alors, on pourra aussi créer une fonction dans laquel le robot va faire le dessin du tableau.

2.B.2 Création des fonctions "X" et "O" À partir de la fonction concise obtenue dans la tâche 2.B.0, on peut créer deux fonctions dans lequel le robot va faire le dessin du mouvement de "X" et "O". La dimension des dessins devra être d'accord avec la dimension du tableau, ainsi cette fonction devra recevoir cette dimension comme un paramètre.

2.B.3 Définition de la disposition graphique d'IHM Dans cette tâche, on va faire l'ébauche de la séquence des actions qu'on attend que l'interface reçoive et quels objets et dispositions graphiques l'interface doit contenir.

2.B.4 Définition de données d'entrée nécessaires IHM À partir des définitions de quels objectifs devront contenir dans l'interface, on va savoir quels les paramètres d'entrée que l'interface recevra. Ainsi, on pourra définir quels sont les autres paramètres et informations nécessaires pour l'exécution totale du jeu. Alors, on

pourra démarrer et nommer les variables qui iront recevoir les paramètres d'entrée du jeu. Afin de faciliter le stockage et gestion de ces paramètres, on doit utiliser variables du type structure.

2.B.5 L'insertion des objects graphiques À partir de la définition précédent, on doit créer et insérer les éléments dans l'interface et déterminer aussi les propriétés respectives des objets.

2.B.6 Création et gestion d'événements À partir de la création des objets, on va faire l'édition de chaque fonction *callback* associée aux événements générés dans l'interface.

2.B.7 Tests, dépannage et intégration Au début, on va tester les fonctionnalités de l'interface créée et, si besoin, on va faire le dépannage et amélioration dans l'algorithme de création. Alors, on prévoit d'intégrer la communication obtenue entre le PC et le robot (résultat de travail de la structure A) avec l'interface créée, afin d'obtenir la communication totale entre le robot $\leftrightarrow$ PC $\leftrightarrow$ utilisateur.

5 Résultats et Discussions

L'objectif principale de la séquence 7 par rapport à ce projet était la familiarisation avec le domaine de modélisation et commande des systèmes robotiques. Alors, on s'est concentré sur l'étude du livre [1] pour comprendre le modèle du robot que nous a été fourni et comment générer les lois de commande pour le système. Ensuite, on a défini l'application qu'on voudrait développer avec le robot et tous les pas nécessaire pour le faire. Ces pas sont montrés dans la section 3.

Pour ces motifs, on n'a pas beaucoup de résultats concrets parce qu'on vien de commencer à mettre en oeuvre la partie 1 de notre travail. On va donc montrer les résultats qu'on a déjà obtenus et les principales difficultés pour les tâches qu'on n'a pas encore fini dans la partie 1 du travail.

5.1 Commande Angulaire et Cartésienne

Pour la commande dans l'espace des articulations, on a mis en oeuvre sur Matlab et Simulink les deux techniques qu'on a montrés dans les subsections 4.1.1 e 4.1.2. Ensuite, on a comparé les deux résultats pour choisir la méthode de commande la plus adaptée à notre solution.

Pour les deux simulations, on a choisi la pulsation de brisure $\omega = 10\ rad/s$ pour toutes les articulations. Si on augmente la fréquence de brisure, on peut réduire le temps de réponse du système, alors on choisira ce temps d'une façon définitive quand l'intégration avec les autres parties du projet serait fait, pour meilleur adapter la commande au robot. Pour des critères de comparaison, on utilisera les mêmes pulsations de brisure pour tous les tests.

Dans la suite, on a les résultats obtenus pour la commande PID, avec une consigne en échelon de $\left[0, -\frac{\pi}{2}, 0, 0, 0\right]$ filtré par un filtre de primier ordre de constant de temps 0.5s, et les efforts de contrôle nécessaires sur chaque articulation.

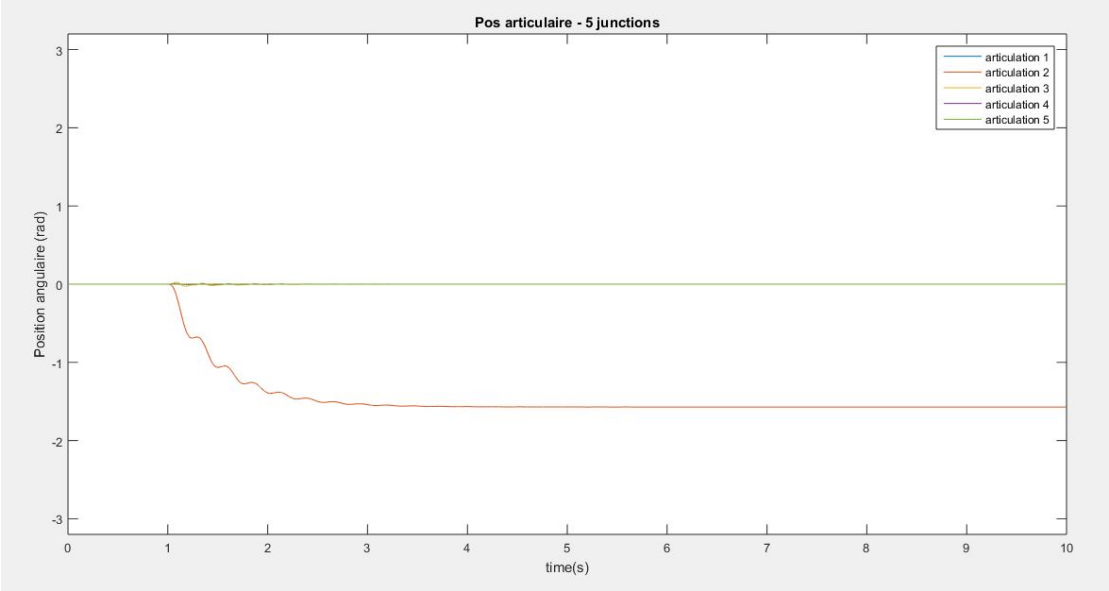

Figure 8 – Réponse temporelle à un échelon de consigne pour le système dans l'espace articulaire commandé pour un PID

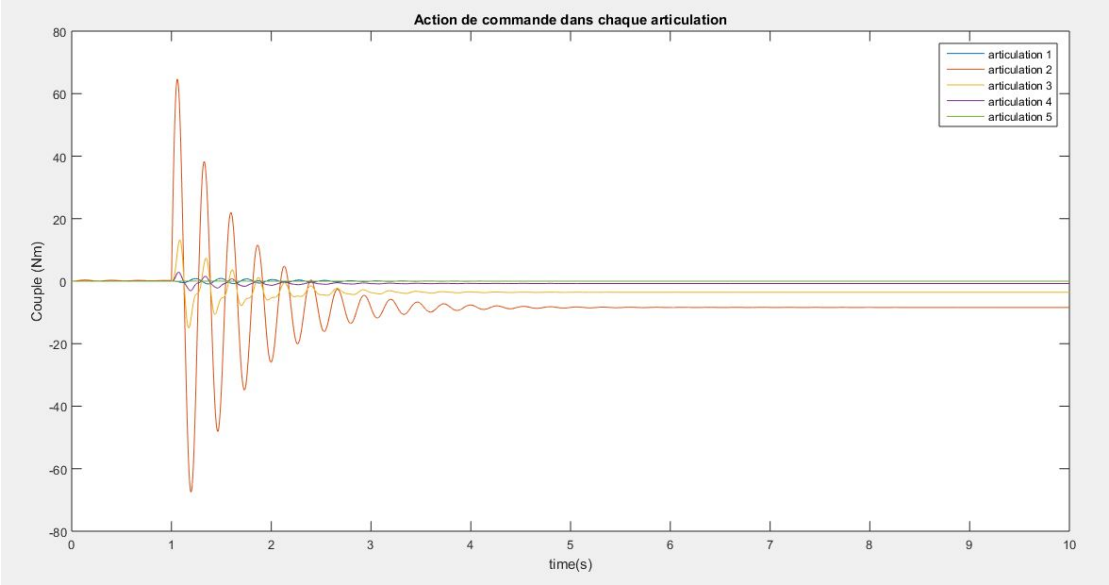

Figure 9 – Efforts de contrôle pour le système commandé pour un PID dans l'espace articulaire

On peut noter dans le graphique que le temps de réponse du système est d'environ 2 secondes et l'erreur statique est nulle, ce qui était attendu une fois qu'on a un contrôleur avec un terme intégrale. Cependant, la réponse contienne beaucoup de fluctuations, on croit en raison des non-linérité du système. Pour le efforts de contrôle, il y a aussi beaucoup de fluctuation et ils sont trop grandes dans le début du mouvement. Alors, cette méthode de commande a quelques problèmes qu'on doit corriger si on l'utilise dans le robot.

Pour la commande PID avec compensation des non-linérités, on a utilisé la même consigne que dans le cas précédent. Dans la suite, on a le graphique qu'on a obtenu dans la simulation pour les position angulaires et les efforts de contrôle.

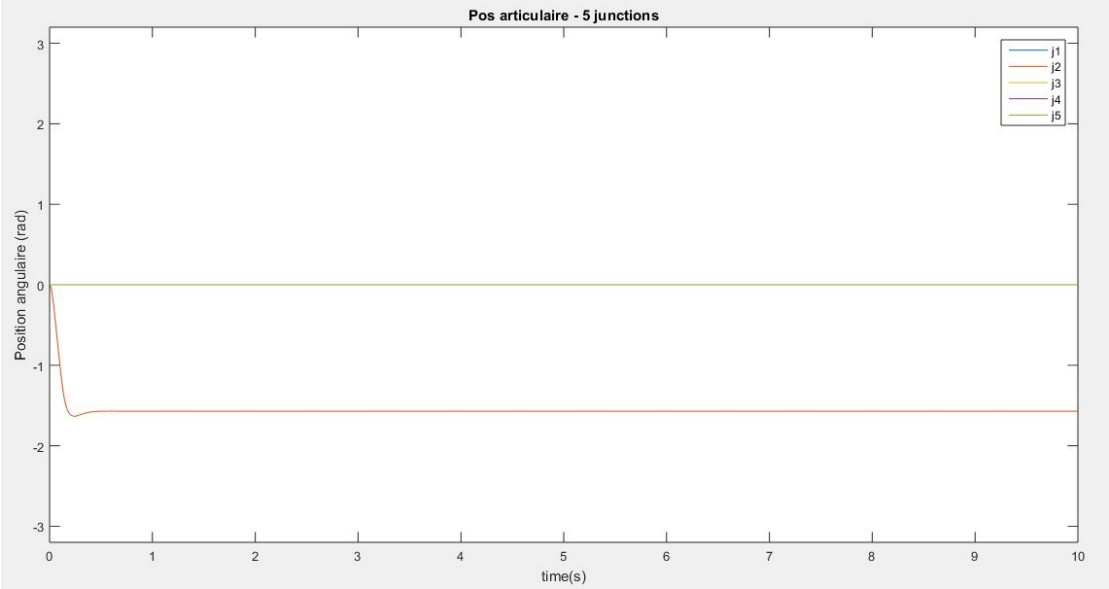

Figure 10 – Réponse temporelle à un échelon de consigne pour le système dans l'espace articulaire commandé pour un PID avec compensation des non-linéarités.

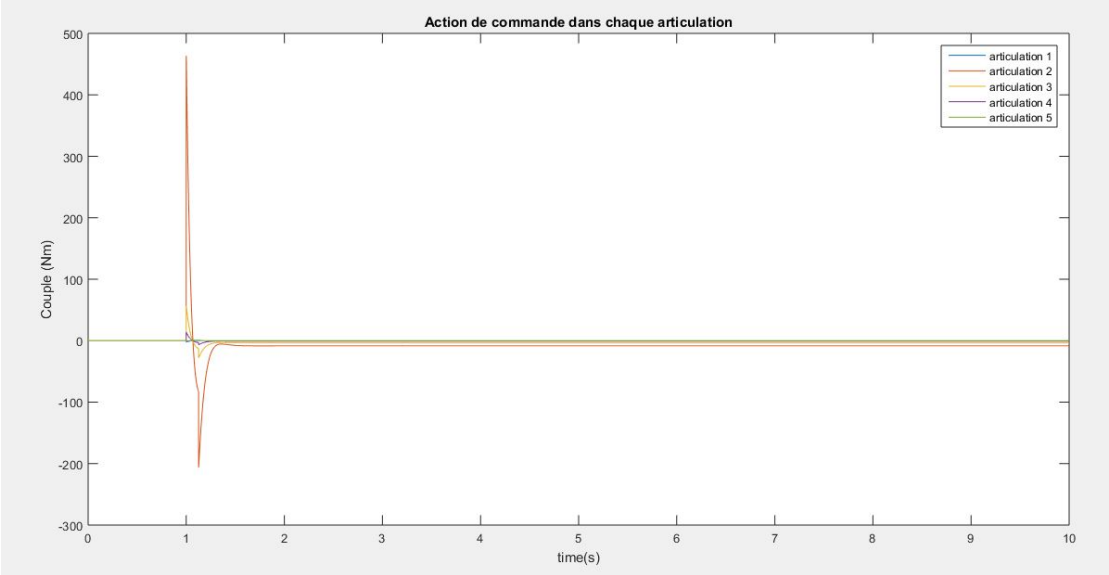

Figure 11 – Efforts de contrôle pour le système commandé pour un PID avec compensation des non-linéarités dans l'espace articulaire.

On peut noter, comme dans le cas du contrôleur PID, que l'erreur statique est nulle en raison du terme intégrale et le temps de réponse est aussi environ 2 secondes. Cependant, on a réduit considérablement les fluctuations, une fois que les non-linéarités du système sont compensées dans le contrôleur. Par contre, on peut voir que les eeforts de contrôle sont beaucoup plus abrupt au début pour assurer ce comportement au système.

On peut voir aussi que ces réponses semple trop bien parce qu'il s'agit des réponses à une modèle simulé du robot. Dans un cas réel, on n'aura pas accès à la vitesse angulaire et on aura aussi des bruits de mesure et des incertitudes du modèle estimé. Alors, on a simulé à nouveau ce modèle en ajoutant un bruit

blanche à la mesure de position et en déterminant de façon numérique la vitesse angulaire du robot. Les résultats sont dans la suite.

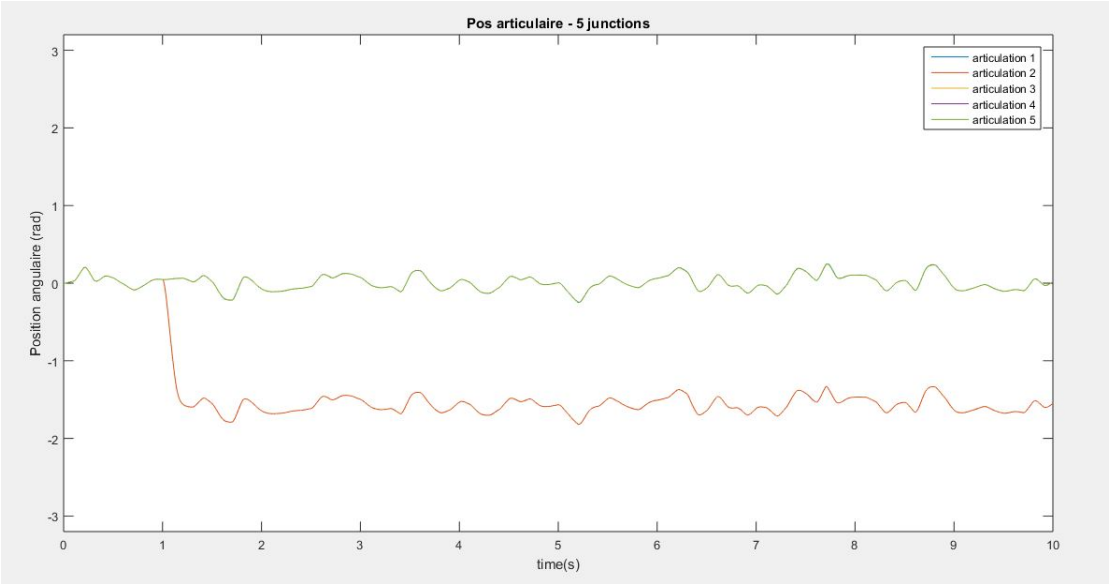

Figure 12 – Réponse temporelle à un échelon de consigne pour le système dans l'espace articulaire commandé pour un PID avec compensation des non-linéarités et en simulant des bruits de mesure.

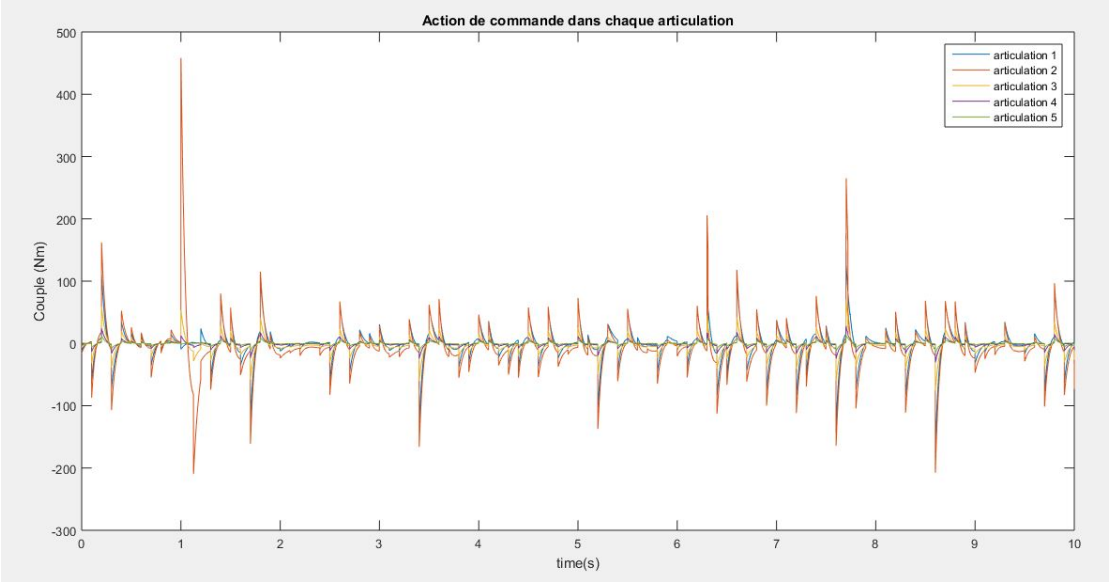

Figure 13 – Efforts de contrôle pour le système commandé pour un PID avec compensation des non-linéarités dans l'espace articulaire et en simulant des bruits de mesure.

On peut voir que, pour la position angulaire, cette méthode semble bien même avec des bruits, mais pour l'efforts de contrôle, il y a plusieurs pics de couple trop grande. Alors, cela peut endommager le robot. On conclut que il n'es pas une bonne idée d'utiliser les consignes de vitesse, parce que les bruits de mesures génèrent des pics abruts quand on fait l'estimation numérique.

Pour la commande dans l'espace opérationnel, on a aussi essayé de mettre

en oeuvre les deux techniques de commande, mais on s'est rendu compte que l'approche du couple calculé ne marcherait pas bien. On a besoinde calculer da dérivé et la pseudo-inverse du Jacobien pour cette approche, alors si le robot a des variations abrupts de position, le dérivé du Jacobien sera trop grande. Le même pour la pseudo-inverse si le robot est proche d'un point de singularité. Alors, à la fin, on a mis en oeuvre juste l'approche du PID.

On a essayé comme consigne de position $[X, Y, Z] = [0.2, 0.2, -0.1]$, aussi un échellon filtré par un filtre de premier ordre. Les réponses angulaire et cartésien et les efforts de contrôle sont montrés dans les figures suivantes.

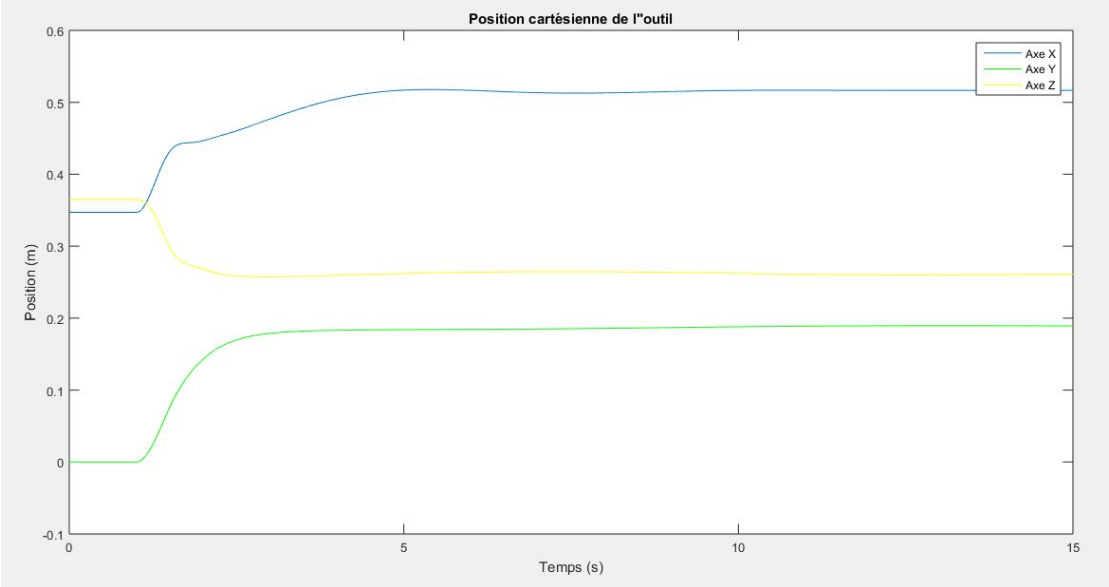

Figure 14 – Réponse temporelle des positions du robot à un échelon filtré de consigne pour le système dans l'espace opérationnel.

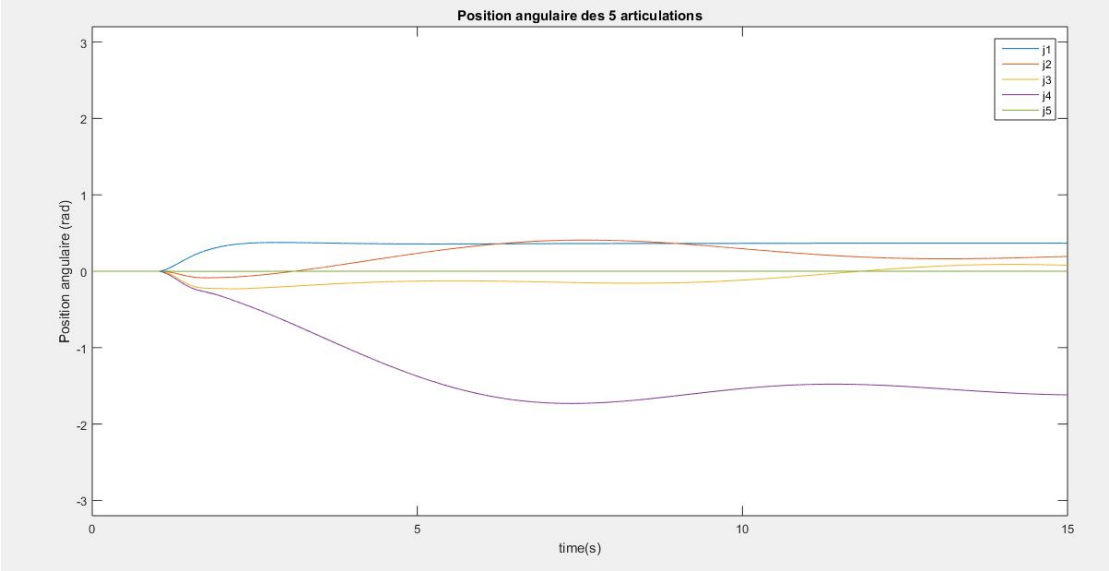

Figure 15 – Réponse temporelle des positions angulaires à un échelon filtré de consigne pour le système dans l'espace opérationnel.

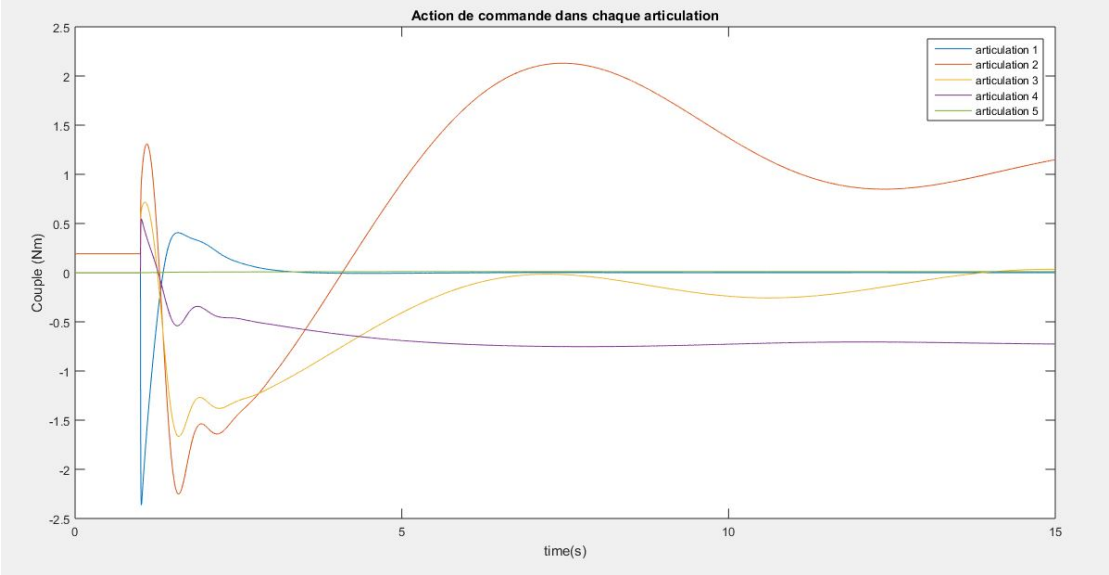

Figure 16 – Efforts de contrôle pour le système dans l'espace opérationnel.

Dans ce cas, on peut voir que le robot est un peu plus lente pour arriver au point de consigne, et il n'arruve pas exactement au point, il reste une erreur en régime permanent, même avec un tèrme intégrale dans le contrôleur. Pour les positions angulaires les efforts de contrôle, on peut voir qu'ils sont bien limités, mais ils ne stabilisent jamais.

Alors, pour la suite on doit fixer ces problèmes et commencer à générer des consignes pour l'orientation d'outil aussi avant de l'intégrer avec le générateur de trajectoire.

5.2 Trajectoires

Linéaire Pour faire la trajectoire linéaire, on a utilisé le polynôme de cinquième dégrée montré en [1], $10(\frac{t}{t_f})^3 - 15(\frac{t}{t_f})^4 + 6(\frac{t}{t_f})^5$, et modifié pour utiliser les entrées, de point initial, point final et le temps total du mouvement.

Par exemple, si le point initial p_i, le point final p_f et le temps t_f sont connus, donc la position suive la fonction suivante :

$$P(t) = (p_f - p_i)(10(\frac{t}{t_f})^3 - 15(\frac{t}{t_f})^4 + 6(\frac{t}{t_f})^5) + p_i \tag{14}$$

Pour un point initial $p_i = [10, 5, 9]$, un point final $p_f = [7, 9, 8]$ et temps final $t_f = 10s$

$$X(t) = (7 - 10)(10(\frac{t}{10})^3 - 15(\frac{t}{10})^4 + 6(\frac{t}{10})^5) + 10 \tag{15a}$$

$$Y(t) = (9 - 5)(10(\frac{t}{10})^3 - 15(\frac{t}{10})^4 + 6(\frac{t}{10})^5) + 5 \tag{15b}$$

$$Z(t) = (8 - 9)(10(\frac{t}{10})^3 - 15(\frac{t}{10})^4 + 6(\frac{t}{10})^5) + 9 \tag{15c}$$

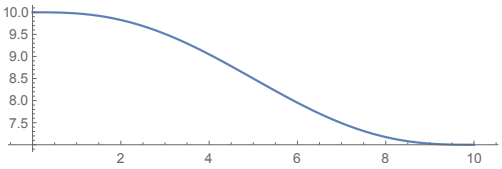

Figure 17 – Graphique de la paramétrisation de X.

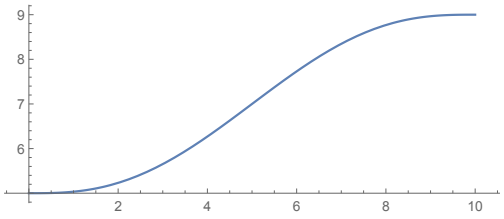

Figure 18 – Graphique de la paramétrisation de Y.

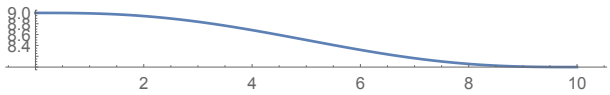

Figure 19 – Graphique de la paramétrisation de Z.

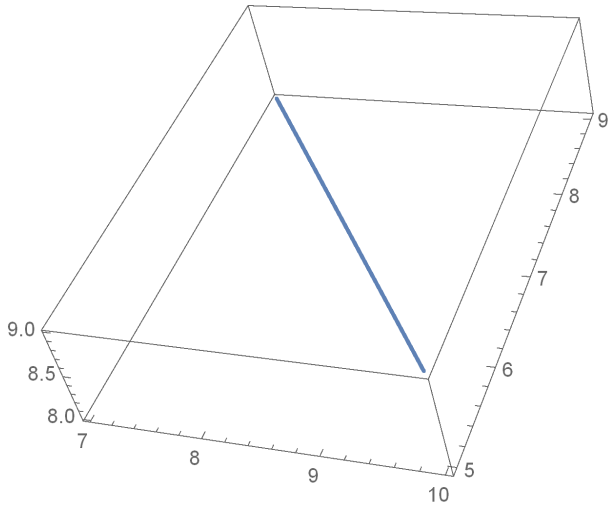

Figure 20 – Représentation tridimensionnel de la trajectoire

Arc de cercle Dans un première moment on a choisi essayer de résoudre le problème en deux dimensions.

Comme paramétrisation du cercle on a pensé en utiliser tel :

$$X = 1 - cos(t) \tag{16a}$$

$$Y = sin(t) \tag{16b}$$

Modifiant la fonction pour la généraliser et en faisant t varier de 0 a la valeur du angle entre les points initial et final et le centre du cercle.

On sait que les dérivatives (1ère et 2ème) de ces équations sont lisse mais ne sont pas prés de 0 au but e a la fin du mouvement, donc il faut aussi créer des polynômes interpolateurs, mais au même temps, attenter pour que la courbe résultante soit le plus prés possible d'un cercle.

6 Conclusion

Ce document donne un résumé de notre travail jusqu'à présent et règlemente tous les travaux qui seront réalisés dans cette séquence, dans laquelle nous espérons terminer ce projet et présenter un système commandé qui soit fonctionnel, robuste et performant. Les informations présentées dans ce document sont sujets à changement en accord avec les besoins du projet et les décisions futures du groupe, une version définitive sera décrite dans le rapport final du projet.

Références

[1] W. Khalil and E. Dombre. *Modeling, Identification and Control of Robots*. Kogan Page Science paper edition. Elsevier Science, 2004.