UNIVERSITÉ DE SHERBROOKE
Faculté de Génie
Département de génie électrique et génie informatique

Rapport d'APP

Commande de robots redondants

Présenté à
Wael Suleiman

Présenté par
Philippe Boulanger - boup3106
André Jacques - jaca1805
Thierry Judge - judt3001
Anthony Quine - quia2504

Sherbrooke - 27 mars 2020

Table des matières

1 Paramètres Denavit–Hartenberg (DH)

Les paramètres DH présentés au tableau 1 ont été déterminés à partir des repères présentés à la figure 2. Les repères ont été placés stratégiquement afin de simplifier le plus possible les paramètres DH et, du même coup, simplifier les matrices de positions et de rotations.

Tableau 1 – Paramètres DH

Link_i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	0	θ_1
2	0	$-\pi/2$	0	$\theta_2 - \pi/2$
3	dThigh	0	0	θ_3
4	dTibia	π	0	θ_4
5	0	$\pi/2$	0	θ_5
6	footHeight	0	0	θ_6

Les valeurs initiales indiquées dans le guide étudiant sont présentées au tableau 2. La position initiale de la jambe est illustrée à la figure 1.

Tableau 2 – Valeurs initiales et numériques des joints

Joint	**0**	**1**	**2**	**3**	**4**	**5**
Valeur initiale	0	0.3	-aThigh	aThigh+aTibia	aTibia	0.3
Valeur numérique	0	0.3	-0.4062	0.8124	0.4062	0.3

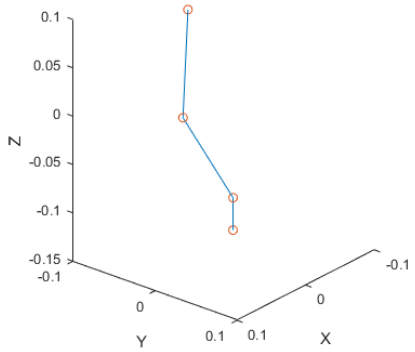

FIGURE 1 – Postition initiale du robot

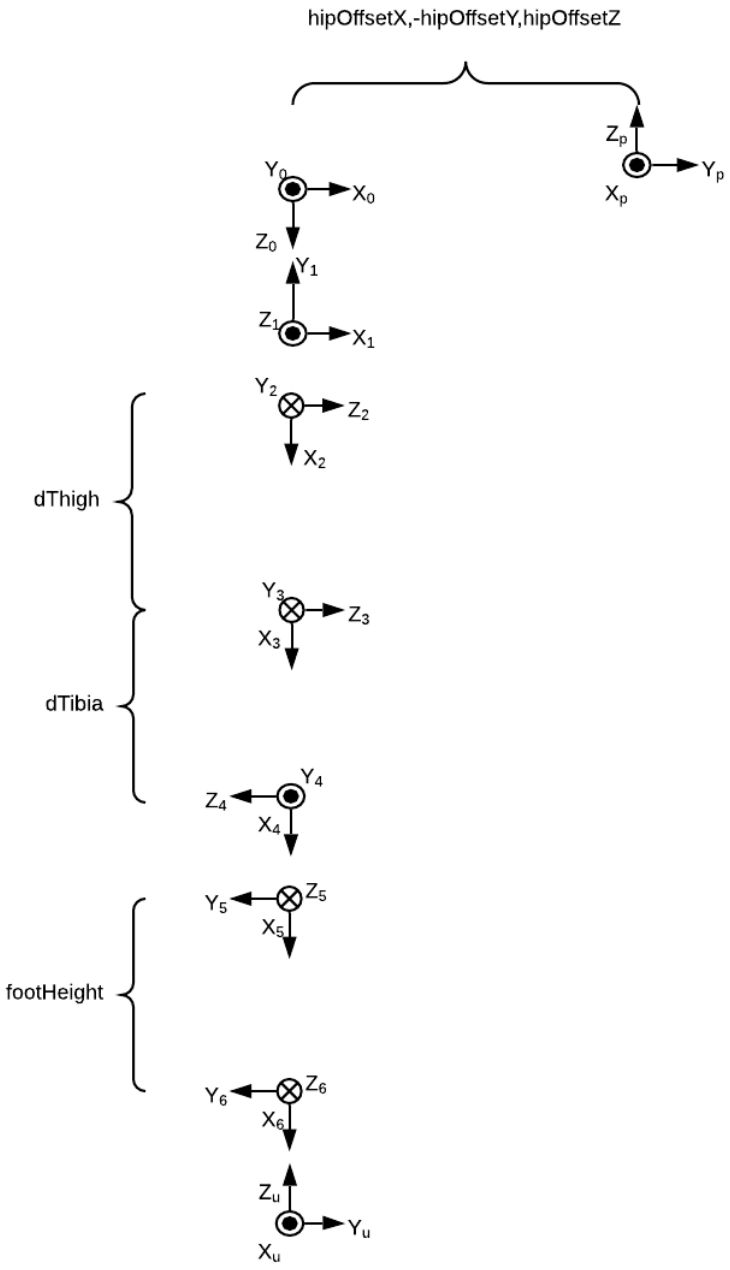

FIGURE 2 – Schéma du DH du robot

2 Trajectoire cubique

La trajectoire cubique est définie entre deux points sur le plan x et y. Les deux points sont la position initiale du pied, x_0, y_0 et la position finale x_1, y_1. La trajectoire en z est définie par deux trajectoires dans le but de soulever le pied pour ensuite le rabaisser. La première trajectoire apporte le pied de $z = z_{initial}$ à $z = 0.025$ et la seconde est de $z = 0.025$ à $z = z_{finale}$ où $z_{initial} = z_{finale}$. La première trajectoire apporte le pieds au point défini par $x_{int} = x_0 + \Delta x$ et $y_{int} = y_0 + \Delta y$. La deuxième trajectoire apporte le pied au point final. La trajectoire cubique est définie selon l'équation suivante [1].

$$q(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{2-1}$$

On en déduit :

$$\dot{q}(t) = 3a_3 t^2 + 2a_2 t + a_1 \tag{2-2}$$

Les coefficients a_3, a_2, a_1, a_0 sont déterminés selon les équations suivantes :

À $t = 0$, $q(t) = a_0$. a_0 est donc la position initiale.

À $t = 0$, $\dot{q}(t) = a_1$. La vitesse initiale et finale du pied doit être de 0 donc $a_1 = 0$.

On peut utiliser les deux équations pour résoudre les deux inconnus restants :

$$a_3 = \frac{-2}{3t_f} a_2 \tag{2-3}$$

$$a_2 = \frac{q_f - q_i}{\frac{1}{3} t_f^2} \tag{2-4}$$

La trajectoire cubique résultante débute à la position de l'effecteur dans la configuration initiale comme il est possible de le constater à la figure 3.

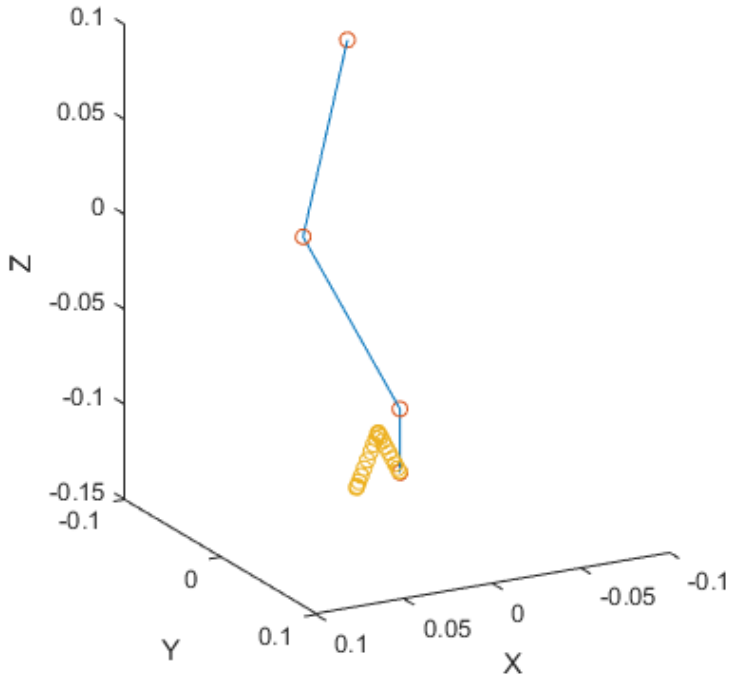

FIGURE 3 – Position initiale du pied avec interpolation souhaitée

3 Pseudo code pour réalisation des tâches

Pour réaliser une tâche, on doit utiliser un algorithme de cinématique inverse. L'algorithme utilisé est celui de la *Jacobian (Pseudo-)inverse* [1].

L'objectif est de déplacer l'effecteur vers une position désirée. On y parvient en minimisant la distance (l'erreur $e = p_d - p_t$) entre la position courante de l'effecteur et la position désirée. Les valeurs de joints sont mis à jour à chaque itération selon l'équation 3-1.

$$q_{t+1} = q_t + kJ^{\#}(q_t)e_t \tag{3-1}$$

où $J^{\#}$ est la pseudo-inverse de la jacobienne J. Pour éviter des problèmes de calcul dans les configurations approchant les singularités, l'équation (3-2) de *damped least-squares (DLS) inverse* est utilisée

$$J^{\#} = J^T(JJ^T + k^2\mathbb{I})^{-1} \tag{3-2}$$

Pour réaliser une seconde tâche simultanément, c'est-à-dire dans le présent cas de s'assurer que l'orientation du pied soit le plus possible parallèle au sol, il faut tenir

compte de deux erreurs, soit l'erreur en position e_p et l'erreur en orientation e_r. L'équation 3-3 indique la valeur des joints pour effectuer le déplacement [2].

$$q_{t+1} = q_t + J_1^{\#} k_p e_{p_t} + \hat{J_2^{\#}}(k_r e_{r_t} - J_2 J_1^{\#} k_p e_{p_t}) \tag{3-3}$$

Cette équation tient compte de l'ordre de priorité des tâches. Dans le cas de la problématique la tâche de position est plus prioritaire que la tâche de rotation.

L'algorithme 1 décrit le pseudocode nécessaire pour réaliser les deux tâches. L'algorithme requiert une fonction pour calculer la cinématique directe à partir de variables de joints $q : K(q)$. Celle-ci retourne la position P_e et l'orientation θ_e de l'effecteur. L'algorithme permet de déplacer l'effecteur vers une position désiré p_d et une rotation désirée θ_d. Dans le cas de la problématique les positions désirées sont les positions déterminées lors de la définition de la trajectoire. L'orientation désirée est de garder le pied parallèle au sol. Puisque le pied commence dans cette orientation, θ_d est fixé à cette orientation.

Algorithme 1 Contrôle pour deux tâches

P_e, θ_e = K(q_0)
$\theta_d = \theta_e$: L'orientation désirée est l'orientation initiale.
$e_p = P_d - P_e$
$e_r = \theta_d - \theta_e$
counterMax = 1500
while ($|e_p| > \epsilon$ or $|e_r| > \epsilon$) and counter < counterMax **do**
    $\Delta q \leftarrow J_1^{\#} k_p e_p + \hat{J_2^{\#}}(k_r e_r - J_2 J_1^{\#} k_p e_p)$
    $q \leftarrow q + \Delta q$
    P_e, $\theta_e \leftarrow$ K(q_0)
    $e_p \leftarrow P_d - P_e$
    $e_r \leftarrow \theta_d - \theta_e$
    counter $\leftarrow$ counter + 1

4 Résultats

Les résultats obtenus dans le script **MATLAB** sont présentés dans les figures de cette section.

D'abord, la figure 4 permet de voir les variations en position de l'effecteur, de même que la non variation de ses angles, assurant ainsi que le pied reste parallèle au sol comme il était requis.

La position initiale en z étant de -0.1146m, et la valeur maximale étant de -0.09m, l'écart est maintenu dans la zone définie dans la problématique.

Aussi, la position finale de l'effecteur en x respecte les requis, en restant en ne dépassant pas un delta de 0.03m.

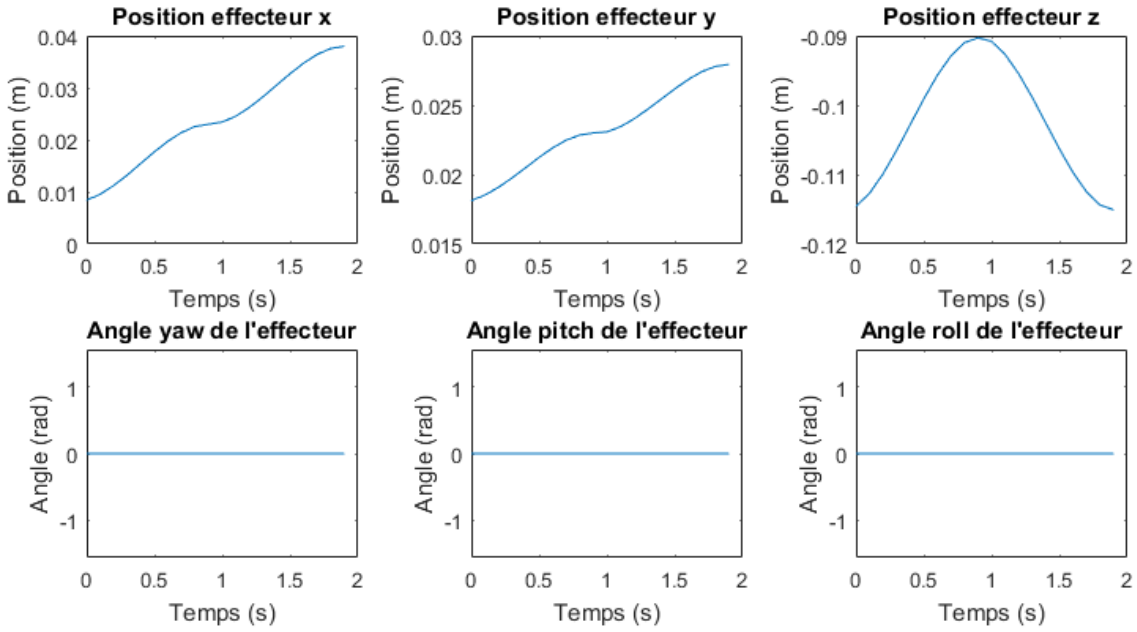

FIGURE 4 – Position et rotation de l'effecteur en fonction du temps

La figure 5 présente la trajectoire de la jambe en utilisant une interpolation avec 20 points. [1] Une vidéo de l'animation est disponible dans le dossier de remise Matlab sous le nom de *APP2_Leg.avi*.

1. Dans le code MATLAB lors de la création de l'animation du mouvement, ce sont 60 points d'interpolation qui sont utilisées pour obtenir un mouvement de 2 secondes avec 30 images par seconde. Pour le fichier texte généré pour le code ROS/Gazebo, ce sont plutôt 50 points d'interpolation qui sont utilisés.

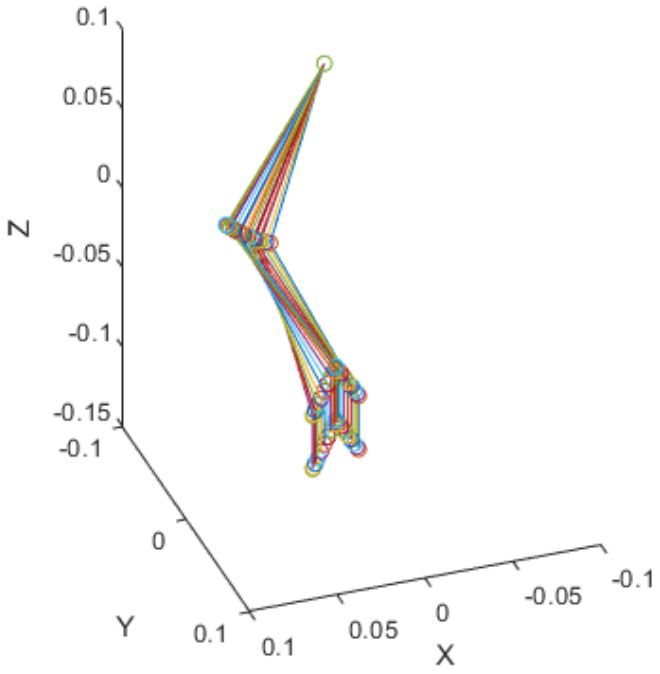

FIGURE 5 – Trajectoire de la jambe avec 20 points

La figure 6 illustre la valeur de chaque joint en fonction tu temps. Ces valeurs d'angle sont obtenues avec l'équation 3-3.

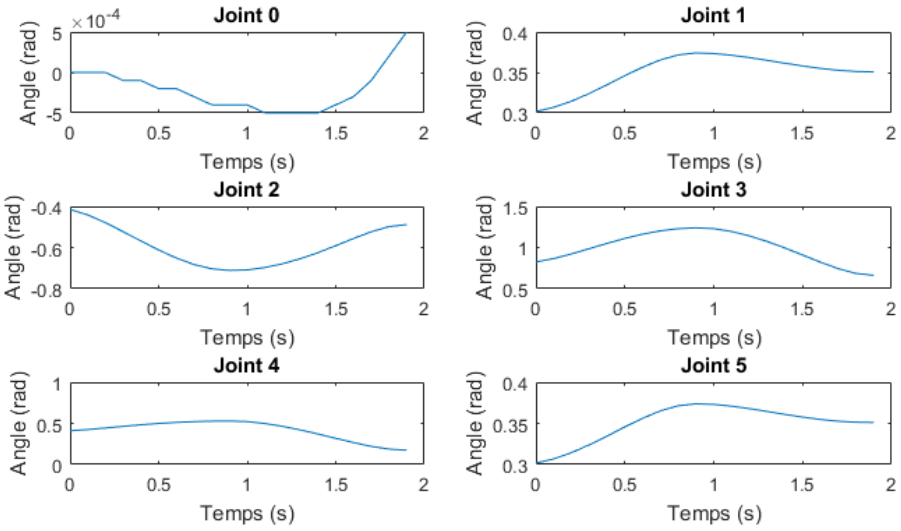

FIGURE 6 – Valeur de chaque joint en fonction du temps

5 ROS / Simulation Gazebo

Pour effectuer la simulation sous ROS/Gazebo, les valeurs des joints ont été extraites sous forme de fichier texte séparé par virgule. Ainsi, la classe de lecture des fichiers fournie pour le laboratoire 4 de la problématique a été modifiée pour utiliser ce caractère de séparation à la place des tabulations.

La vidéo résultant de l'exécution de la commande est fourni à la racine de l'archive de dépôt. Pour en répliquer le résultat, il suffit de placer le fichier `qs.txt` à la racine de votre *workspace* catkin, puis d'exécuter les commandes suivantes :

```
# Terminal #1, lancement de Gazebo avec le modèle de DarwinOP inséré
cd <emplacement_de_votre_catkin_ws>
source devel/setup.bash
catkin_make

roslaunch darwin_gazebo darwin_gazebo.launch

# Terminal #2, lancement des contrôleurs des joints :
cd <emplacement_de_votre_catkin_ws>
source devel/setup.bash

roslaunch darwin_control darwin_control.launch

# Terminal #3, lancement de la boucle de contrôle
cd <emplacement_de_votre_catkin_ws>
source devel/setup.bash

rosrun darwin_control Darwin_Control_node
```

Le code réalisé exécute d'abord une initialisation des valeurs des joints pour placer DarwinOP à la bonne position pour débuter son pas, tout en étant pas trop rapide afin d'éviter un déséquilibre. Les joints *ankle2* et *thigh1*, du côté droit et du côté gauche du robot, sont amenés jusqu'à leurs premières valeurs contenues dans le fichier texte de valeurs d'angles des joints, extraites du code MATLAB.

Ensuite, comme 50 points d'interpolation ont été utilisés pour deux secondes de mouvement, on utilise un *rate* de 25Hz et une boucle de contrôle *while* afin d'envoyer toutes les valeurs de joints du fichier texte aux contrôleurs du robot au bon intervalle de temps.

Le code source est fourni dans le dossier `ROS_APP/src/` de l'archive de dépôt.

6 Noeud de lecture des positions

Ce noeud a été entamé durant la réalisation du laboratoire 4. Il n'a pas été complété étant donné la situation.

Toutefois, le plan pour la réalisation de cette étape était de s'abonner au topic «`/darwin/joint_states`» plutôt qu'à ceux des différents joints comme il était conseillé de le faire dans les consignes du laboratoire 4.

En effet, s'abonner à ce topic est plus avantageux selon nous puisqu'il retourne des données dans une structure contenant à la fois un tableau des noms de joints et un tableau des positions de ces joints comme le montre la figure 7.

```
---
header:
  seq: 4148
  stamp:
    secs: 82
    nsecs: 982000000
  frame_id: ''
name: [j_ankle1_l, j_ankle1_r, j_ankle2_l, j_ankle2_r, j_gripper_l, j_gripper_r,
 j_high_arm_l,
  j_high_arm_r, j_low_arm_l, j_low_arm_r, j_pan, j_pelvis_l, j_pelvis_r, j_shoul
der_l,
  j_shoulder_r, j_thigh1_l, j_thigh1_r, j_thigh2_l, j_thigh2_r, j_tibia_l, j_tib
ia_r,
  j_tilt, j_wrist_l, j_wrist_r]
position: [-0.00028123439477845125, 0.00015051651954234302, -0.00028464825569418
65, -0.0002587130242535096, -1.3025353418072427e-07, 1.3202408943868704e-07, 0.0
014193448682391363, 0.0014183383184507292, -6.386684887615957e-07, 3.01092955012
9049e-08, -2.7514071110346094e-07, 7.949105652826916e-06, 1.6644539734933517e-05
, 0.00011424200023224529, -0.00010137439767987644, 0.0004861296905964707, 0.0005
18155018808919, 0.00018434851825865906, -0.00031927947623611175, 0.0002590730112
1641524, -0.00026491699310504657, 7.857296324687013e-07, -0.00015502072945583478
, -0.00015423977684125845]
velocity: [0.008441001243388487, -0.0016612702021020294, -0.006913620025716136,
0.003140585015334645, -3.927933449425203e-05, -3.670170409607123e-05, -0.0004460
117483473321, -0.000628747831633446, -2.867012549399784e-06, -6.2442870413032086
e-06, 9.719871509184392e-06, -7.101850423997278e-05, 1.6703800673164464e-05, -3.
8561045181169345e-06, 1.8566631107881935e-05, -0.00031320665097720694, 3.2503548
83605776e-06, 7.2180064086260 61e-05, -0.00036625513659774165, 0.0005209082779371
567, 0.0007240934346590218, 8.458262959442266e-06, -0.0005088007533177654, -0.00
05062986579994087]
effort: [-0.014673366830830759, 0.0012741996614273887, 0.032087573932049196, -0.
019814111374572008, 2.8921887107458133e-06, -5.174808990204838e-05, -0.142342813
5916005, -0.14210371172884706, 6.620988246552884e-05, -2.132436014434802e-05, 3.
8060991247590 44e-05, -3.629164133656104e-05, -0.0019526019720217391, -0.01141959
2869987838, 0.010132274401186692, -0.044148597251236765, -0.0545822151329034, -0
.017443101661740457, 0.035263035099841744, -0.036883146427868496, 0.019171420595
665012, -6.74870300976238e-05, 0.015199229006590542, 0.01507629312822445]
---
```

FIGURE 7 – Valeurs envoyées sur le topic «/darwin/joint_states»

À l'inverse, la structure de donnée transmise sur les topics conseillés dans le laboratoire ne retourne pas d'information permettant de savoir de quel joint il s'agit, comme le montre la figure 8.

```
---
header:
  seq: 8305
  stamp:
    secs: 83
    nsecs:  72000000
  frame_id: ''
set_point: 0.0
process_value: 1.83055568419e-05
process_value_dot: -2.03145178284e-05
error: -1.83055568419e-05
time_step: 0.001
command: -0.00192162804273
p: 100.0
i: 0.01
d: 10.0
i_clamp: 0.0
antiwindup: False
---
```

FIGURE 8 – Valeurs sur le topic «/darwin/j_pelvis_r_position_controller/state»

Il aurait alors fallu créer une fonction de *callback* pour chaque joint, alors qu'il est possible d'en faire une seule qui va vérifier les valeurs des angles de tous les joints d'intérêt d'un seul coup.

Ainsi, après l'abonnement à ce *topic*, les valeurs des angles des joints d'intérêt auraient été conservées dans un tableau dont l'analyse aurait permis de comparer les valeurs réelles aux valeurs désirées.

La différence qui aurait alors été détectée est fort probablement due à l'aspect théorique de l'interpolation, versus la réalité simulée par Gazebo. Les contrôleurs de joints, avec leurs asservissements munis de PIDs, peuvent se rendre très près des valeurs régies par la boucle de contrôle, mais il est très rare voire impossible qu'ils arrivent à les atteindre parfaitement en pratique.

7 Références

[1] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics : Modelling, Planning and Control*. Springer, London, 1st ed. 2009 édition edition, 2011.

[2] Yoshihiko Nakamura. *Advanced robotics - redundancy and optimization.* Addison-Wesley, 1991.