Projet
Caractérisation d'un robot sériel : commande articulaire et opérationnelle d'un robot simulé réaliste.

1 Présentation du système robotique

L'objectif de ce projet est la caractérisation d'un robot de type "bras" fonctionnant dans le plan, et réalisé à partir de moteurs simulés tous identiques. Ce TP utilisera un robot simulé "réaliste", qui par certains aspects reproduit les phénomènes physiques que vous auriez vus sur les maquettes réelles. Il s'agira en particulier d'établir différents modes de contrôle permettant au bras de réaliser une tâche donnée (écriture, prise/dépôt d'une pièce, etc.) Pour cela, les fonctions utilisées lors du TP1 de modélisation géométrique directe et inverse seront réutilisées autant que possible, moyennant leur adaptation à la modélisation du robot simulé proposé ici. **En pratique, leur utilisation est légèrement modifiée, et les fonctions vous sont donc fournies.** Pour leur utilisation, voir §2.2. Par contre, vous devez réutiliser les fonctions `trans_homogene`, `rotation_Z` et `inv_rotation_Z` que vous avez rédigées lors du TP1.

1.1 Les moteurs DYNAMIXEL simulés

DYNAMIXEL est une gamme de servomoteurs programmables pour robots prenant la forme d'un module tout-en-un (réducteur, contrôleur, fonction réseau et pilote). Petits, légers, puissants, les actionneurs utilisés sur les maquettes réelles projet font partie de la gamme AX, regroupant les servomoteurs d'entrée de gamme. Les moteurs DYNAMIXEL simulés dans ce projet sont tous identiques, et seront utilisés en modes "position". Dans ce mode, les servomoteurs sont déjà asservis individuellement en position, et réagissent à un changement de consigne exprimé en "pas moteur" dont la valeur est absolue et codée entre 0 et 1024, cf. figure 1. La vitesse de déplacement d'une position à une autre est fixe et peut être réglée logiciellement via l'interface Matlab (cf. §1.2).

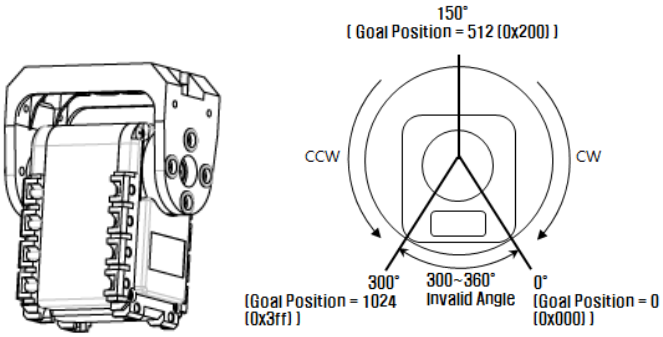

FIGURE 1 – (gauche) Moteur AX12 réel. (droite) Convention pour le repérage des positions angulaires du moteur.

1.2 Logiciel

Le robot simulé peut être commandé à l'aide de fonctions Matlab simples qui permettent de reproduire le mouvement du robot en fonction des paramètres de commande, en particulier la vitesse de déplacement et la position articulaire cible.

Quelques exemples de code permettant de simuler le système :

— Créer un robot 3R plan dont les segments font une longueur de 10cm, 12cm et 8cm :
```
L = [0.1, 0.12, 0.08];
robot = create_robot(L);
```

— Lire la position courante du robot (q est exprimé en pas moteur) :
```
q = robot.getPosition();
```

— Lire le réglage de vitesse courant (v est compris en 1 et 10) :
```
v = robot.getSpeed();
```

— Régler la vitesse à 7 :
```
vitesse = 7;
robot.setSpeed(vitesse);
```

— Régler la position articulaire en pas moteurs, et faire bouger le robot à cette position :

```
q_target = [750, 300, 900];
robot.setPosition(q_target);
```

— Régler la position articulaire en pas moteurs, faire bouger le robot à cette position, et tracer la trajectoire opérationnelle de l'organe terminal :
```
q_target = [512, 350, 600];
robot.setPosition(q_target,1);
```

— Récupérer les positions articulaires au cours du temps (les vecteurs q et t peuvent être utilisés pour tracer ces données) :
```
q_target = [750, 300, 900];
[q,t] = robot.setPosition(q_target);
```

L'aide des fonction est bien entendu disponible afin de préciser leur utilisation (taper `help robot.setPosition` par exemple).

2 Travail demandé

Récupérer sur Moodle le fichier `TProbot.zip`. Le décompresser dans le répertoire de votre choix. Matlab doit ensuite être lancé et son chemin de travail spécifié vers ce répertoire.

2.1 Caractérisation des actionneurs

Comme vous pourrez le remarquer, les appels aux fonctions Matlab sont bloquants. Vous n'avez donc accès aux données simulées qu'une fois que le mouvement du robot est terminé. Selon le paramètre de vitesse utilisé, ce mouvement peut être lent ! Soyez donc patients.

1. Exprimer la relation mathématique reliant les valeurs des pas moteurs lus depuis l'actionneur en valeurs angulaires exprimées en radian. On fera attention à introduire une valeur d'offset permettant de régler le 0 de la mesure. Pour cela, vous référer à la figure 1.
2. En déduire une fonction Matlab que vous écrirez vous même `q=step2angle(step,offset)` qui permet de convertir une valeur de pas moteur (variable `step`) en un angle `q`, exprimé en radian. La variable `offset` doit être utilisée de façon à spécifier à quelle valeur de pas moteur correspond l'angle $q = 0$ rad.
3. Inverser la relation obtenue précédemment pour obtenir l'expression reliant cette fois un angle exprimé en radian à la valeur de pas moteur correspondante.
4. En déduire une fonction Matlab que vous écrirez `p=angle2step(q,offset)` qui permet de convertir un angle `q` exprimé en radian en une valeur de pas moteur `p` pour un `offset` donné.
5. Testez vos fonctions de façon à vérifier leur bon fonctionnement (par exemple, en faisant une conversion pas moteur vers angle, puis angle vers pas moteur : vous devriez retrouver, sous certaines conditions à exprimer, la même valeur).
6. Déclarer un robot 1R, de longeur de segment $L = 0.2$cm. Régler la vitesse de déplacement à 5.
   (a) Pour un mouvement allant de la position moteur 300 à 700, tracer l'évolution de la position du moteur en fonction du temps. Cette position sera exprimée tout d'abord en "pas moteur", puis en degrés ou radians ;
   (b) Effectuer ce tracé pour différentes valeurs de consignes de vitesse ;
   (c) Sur le bases de ces tracés en position obtenus pour différents réglages en vitesse, déterminer la vitesse angulaire effectivement obtenue, exprimée en pas moteur par seconde, puis en tours par minute.
   (d) Les tracés de la position articulaire en fonction du temps permettent également d'estimer la vitesse de rotation du moteur par différence finie. Tracer cette vitesse (estimée à partir de la courbe de la position en fonction du temps), et en commenter l'allure. Comment ces tracés évoluent-ils en fonction de la vitesse ? Comparer la valeur de vitesse ainsi obtenue avec celle estimée à la question précédente.

2.2 Robot 3R plan

Il s'agit maintenant de réaliser un robot 3R plan que nous allons commander à l'aide des fonctions spécifiques au robot simulé que vous venez d'utiliser. A l'issue du TP1, vous savez également manipuler les modèles cinématiques direct et inverse d'un robot sériel simulé. Pour cet usage, 2 fonctions MATLAB vous sont fournies. Elle sont quasi-identiques à celles que vous avez utilisées à l'occasion du TP1 :

— `T = mod_geo_dir_3Rplan(L,q)` : pour un robot 3R plan de longueurs de segment `L` et de position articulaire `q` **en radian**, la fonction vous renvoie la matrice de transformation homogène associées à l'organe terminal du bras. En d'autres termes, la fonction vous permet de déterminer la position (x, y) et orientation θ de l'organe terminal du bras pour les consignes motrices `q = [q1, q2, q3]` de chacun des degrés de liberté en rotation du robot ;

— `[qa, qb] = mod_geo_inv_3Rplan(L, target)` : cette fonction vous permet de déterminer les commandes motrices `qa` et `qb` à appliquer à un robot 3R plan de longueurs de segment `L` de façon à atteindre la configuration opérationnelle cible `target = [x,y,theta]`. Pour un robot 3R plan, 2 solutions à ce problème sont donc possibles.

7. Déclarer un robot simulé 3R de longeurs de segment $L_1 = 10$cm, $L_2 = 8$cm et $L_3 = 12$cm.
8. Représenter le schéma cinématique du robot réalisé. Nommer les angles de rotation, et identifier les paramètres géométriques nécessaires.
9. (a) A l'aide de la fonction `mod_geo_dir_3Rplan` déterminer la pose (x, y, θ) de l'organe terminal pour 3 commandes articulaire q_1, q_2 et q_3 de votre choix, exprimées en rad.
   (b) Appliquer ces commandes au robot simulé, pour différentes vitesse, et vérifier que la pose atteinte corresponde bien à celle obtenue sur le robot simulé.
   (c) Commenter l'éventuelle différence relevée.
10. Commenter le mouvement du robot. Partant d'une position initiale quelconque, maîtrise-t'on sa trajectoire dans l'espace opérationel ? Commenter.
11. Décider d'une pose (x, y, θ) cible dans l'espace opérationnel. A l'aide de la fonction `mod_geo_inv_3Rplan`, déterminer les ordres moteurs à appliquer pour atteindre cette pose. Exploiter alors ces ordres moteurs pour commander le robot simulé. Comparer la pose atteinte par son organe terminal avec la consigne. Commenter.
12. Sur un robot réel, on limite traditionnellement le mouvement de chaque moteur à un débattement angulaire suffisant pour la tâche à réaliser. Cela permet également de gérer les auto-collisions, non prises en compte dans la simulation. On vous demande donc d'écrire une function Matlab `new_pos = limit(pos,val_min,val_max)` limitant le débattement angulaire pos entre une val_min et val_max de sorte que chaque moteur ne puisse bouger qu'entre $-\pi/2$ et $\pi/2$. **Il faudra appeler cette fonction systématiquement pour limiter le débattement angulaire articulaire avant tout mouvement du robot simulé.**

3 Génération de trajectoires

On souhaite maintenant faire en sorte que le robot soit capable de suivre une trajectoire. Cette trajectoire peut par exemple permettre d'éviter un obstacle présent autour du robot, etc. Deux types de calcul de trajectoire sont envisageables (cf. section 3 du TP1) :

— trajectoire dans l'espace articulaire : soit une cible opérationnelle (x, y, θ) et une position du robot caractérisée par sa position articulaire q_i^0 initiale. On peut alors déterminer les commandes motrices q_i permettant d'atteindre la cible. Il s'agit ici de calculer par interpolation les commandes motrices intermédiaires permettant d'aller de q_i^0 à q_i.

— trajectoire dans l'espace opérationnel : soit une cible opérationnelle (x, y, θ) et une pose initiale (x^0, y^0, θ^0) de l'organe terminal. On peut alors déterminer par interpolation les pose intermédiaires menant en (x, y, θ), et en déduire les ordres moteurs q_i à appliquer.

On demande à ce que ces 2 types de commande, dites "commande dans l'espace articulaire" et "commande dans l'espace opérationnel" respectivement, soient implémentées sur le robot simulé. Préciser les avantages, inconvénients et limites de ces 2 commandes. On demande d'illustrer leur utilisation dans le cadre d'une tâche d'écriture d'une lettre. La figure obtenue pour cette tâche d'écriture devra être fournie dans votre rapport.

4 Rapport

Vous devez travailler **par binôme**. Vous devez écrire ensemble un rapport (sous la forme d'un fichier PDF) comportant, de manière détaillée, argumentée et commentée les résultats obtenus à chacune des questions posées. Les figures fournies doivent avoir un titre, les axes des figures doivent être précisés, et les figures interprétées. Les codes des fonctions peuvent être fournis dans le rapport, mais uniquement si vous le jugez nécessaire ; à ce titre, le code

écrit dans votre rapport doit être également commenté afin de préciser le rôle de chacune des lignes.

Vous devez remettre votre rapport, ainsi que (i) le fichier Matlab principal lançant le tracé de vos figure, (ii) les 2 fonctions de conversion pas moteur/angle, et (iii) la fonction de `limit` pour le **lundi 6 avril, 16h** au plus tard. N'oubliez pas de préciser vos noms sur le rapport remis.