SIMSYS TP1
Simulation dynamique d'un robot manipulateur 6 axes

L'objectif du TP est l'étude de la commande d'un robot 6 axes en dynamique. Pour ce TP, on exploitera la toolbox « Simscape/Multibody » de Matlab. A partir d'une maquette numérique solidworks, on réalisera un export du modèle dynamique que l'on exploitera ensuite dans le cadre du TP.

1. Robot 6 axes

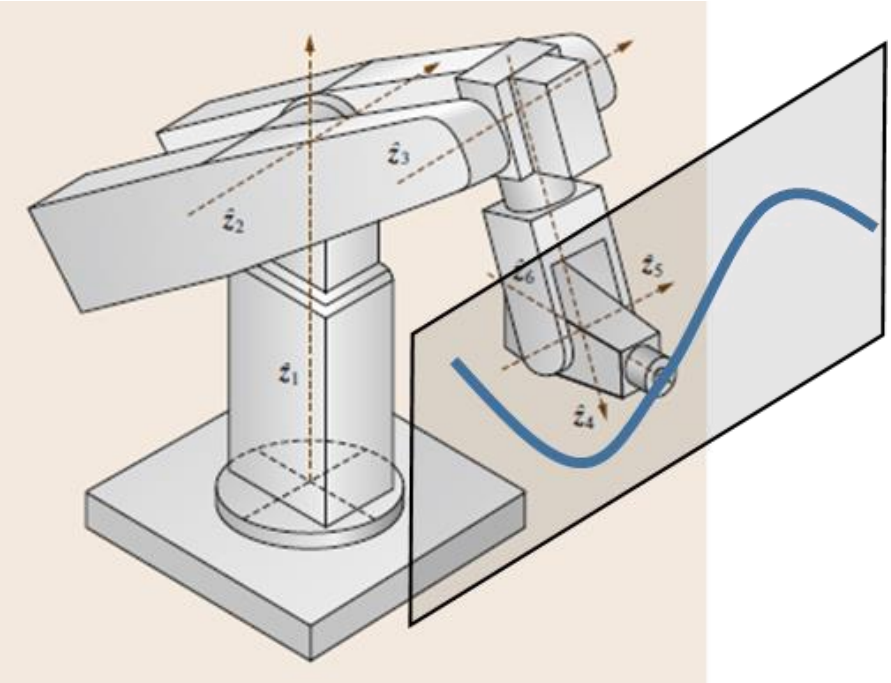

Figure 1 – robot 6 axes

Pour ce TP, nous adopterons l'architecture d'un robot 6 axes « générique » constitué d'une structure porteuse classique RRR et d'un poignet sphérique (3 derniers axes concourants). La figure 1 présente l'architecture générale (issue du Handbook of Robotics édité par Springer) du robot 6 axes et la figure 2 un schéma cinématique associé.

Pour réaliser le suivi de trajectoire, nous allons d'abord définir des trajectoires articulaires à suivre au cours du movement. Ensuite, nous allons générer un modèle dynamique plausible du robot. Enfin, en vérifiant que nous obtenons une trajectoire et une orientation dans l'espace opérationnel correspondant à l'objectif fixé, nous allons

simuler une commande par PD des différents axes et évaluer son impact sur les performances dynamiques du robot.

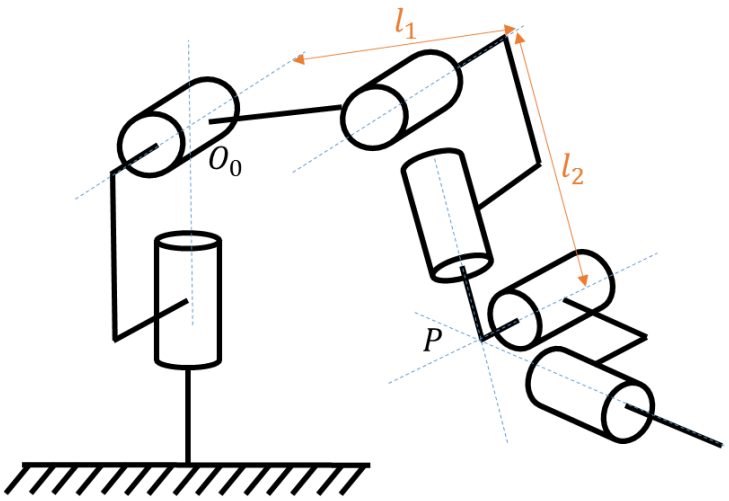

Figure 2 – schéma cinématique

2. Problématique

On cherche à piloter ce robot pour une tâche de soudage. L'objectif est de déposer un cordon de soudure sur une trajectoire représentée figure 1 en grisé.

A tout instant, l'organe terminal du robot doit conserver la pose suivante :

$U_0 = \begin{bmatrix} 0 & 0 & 1 & {}^0p_{6_x} \\ 1 & 0 & 0 & {}^0p_{6_y} \\ 0 & 1 & 0 & {}^0p_{6_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Correspondant à une orientation terminale orthogonale au plan « grisé » de la figure 1, ${}^0p_6$ étant la position du point central du poignet dans l'espace.

Afin de réussir cet asservissement, il est donc nécessaire d'établir le MGI du robot permettant d'exprimer en tout point de la trajectoire les valeurs angulaires correspondantes. Le robot se déplace ensuite à vitesse constante sur les trajectoires afin de souder les parties à souder.

3. Trajectoires à suivre

Le robot devra suivre les trajectoires suivantes :

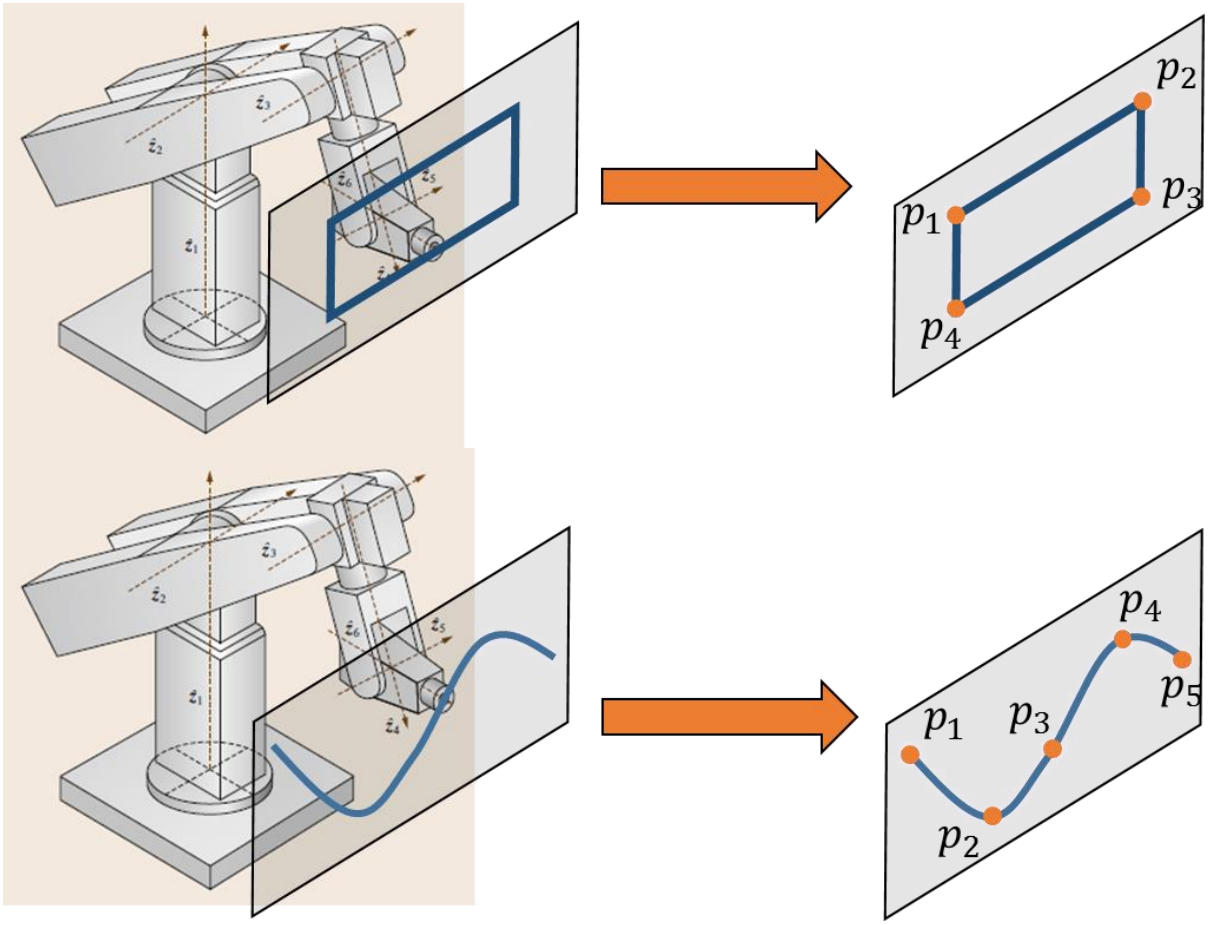

Figure 3 – trajectoires de suivi

Ces deux trajectoires sont définies dans les scripts suivants :

- `TP_6axes_trajectoire_rectangulaire` : La première trajectoire est un suivi de cordon sur une trajectoire rectangulaire. **Exécuter ce script et commenter le résultat.**
- `TP_6axes_trajectoire_polynomiale`: La première trajectoire est un suivi de cordon sur une trajectoire rectangulaire. **Exécuter ce script et commenter le résultat.**

Ces scripts permettent de récupérer des trajectoires articulaires à suivre pour chacun des axes du robot et respecter l'orientation de l'outil dans l'espace opérationnel.
A noter que vous trouverez dans le PDF « SIMSYS_6axes » des informations concernant l'établissement du modèle géométrique direct et inverse du robot. **A quels points doit-on faire particulièrement attention pour exploiter les résultats de ces modèles ?**

4. Assemblage de la maquette numérique et export du modèle numérique

L'objectif de cette partie est d'assembler la maquette numérique du robot et de l'exporter dans Matlab.

Vous avez à votre disposition dans le dossier « maquette numérique » un ensemble de pièces représentant les différentes parties du robot à assembler.

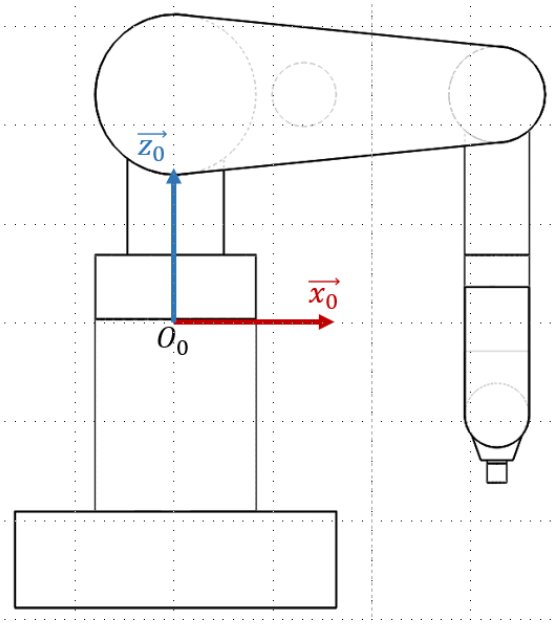

Figure 3 – Définition de l'origine de l'assemblage et de la pose de référence du robot

Afin de pouvoir exploiter une maquette numérique comme modèle dynamique dans la toolbox Multibody, il est nécessaire de respecter les contraintes suivantes :

- **Définition d'un solide « base » positionné par rapport à l'origine du repère.** Dans notre cas, la base sera fixe par rapport à l'origine de l'assemblage et orientée comme présenté figure 3.
- **Définition des contraintes entre solides permettant de respecter les liaisons cinématiques.** Il est fondamental de contraindre les solides entre eux afin de traduire les liaisons cinématiques les liant. Une liaison pivot entre deux solides se définit, par exemple, par une contrainte « appui plan » (arrêt axial) et une contrainte « coaxiale » (rotation autour de l'axe de la liaison). Attention, afin de bien respecter les différentes liaisons, exploiter pour les arrêts axiaux les plans de symétrie des pièces.
- **Définition d'une pose de référence** : on ajoute autant de contraintes que nécessaire dans l'assemblage afin de contraindre totalement l'assemblage dans la pose de référence voulue (voir figure 3). On supprime ensuite ces contraintes supplémentaires afin de libérer les degrés de liberté voulus.

- **Définition des propriétés inertielles des différents solides.** Comme nous avons ici une représentation très basique du robot, on a choisi d'attribuer un matériau unique à l'ensemble des pièces, représentatif des masses mises en jeu sur ce genre de robot (ici, un alliage d'aluminium donnant au total pour ce robot une masse de 250 kg).

Une fois ces différentes étapes réalisées, on peut **exporter le modèle** en activant dans les compléments de solidworks « Simscape Multibody Link ». On a alors accès dans le menu « outils » au menu « Simscape Multibody Link ». Avant d'exporter le modèle, il faut choisir dans « settings » le format d'export graphique « .STL ».

Une fois exporté, un fichier XML contenant les différentes caractéristiques du robot est enregistré dans le répertoire de l'assemblage, avec les fichiers graphiques de représentation des segments.

L'étape suivante consiste à ouvrir matlab et se placer dans le répertoire de l'assemblage. On fait ensuite appel à la fonction « smimport » afin **d'importer le modèle.**

Si vous avez correctement suivi les étapes ci-dessus, vous devriez avoir un modèle dans une fenêtre simulink se présentant sous la forme suivante :

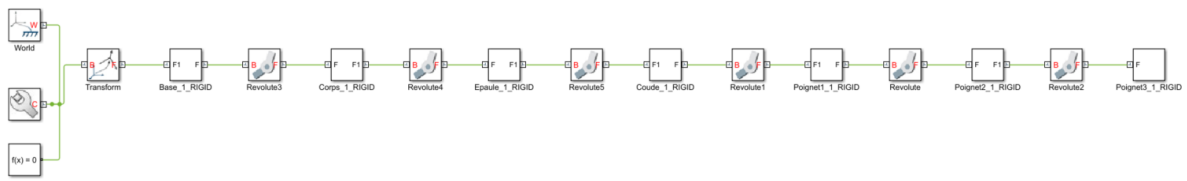

Figure 4 – Modèle Multibody importé

5. Ajustement des modèles dynamiques et géométriques

Avant de pouvoir exploiter la simulation dynamique du robot, il est nécessaire de procéder à un certain nombre de réglages préliminaires.
Tout d'abord, il est nécessaire de paramétrer les liaisons actionnées du robot en précisant ce qui est piloté et ce qui est calculé par la simulation.

1. Dans un premier temps, comme nous avons les données de mouvement calculées à partir du modèle géométrique inverse, nous allons les exploiter pour piloter le modèle, ce qui signifie que nous allons réaliser une simulation par **dynamique inverse. On cherchera donc à obtenir la résolution du problème suivant :**

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + G(\boldsymbol{q}) = \boldsymbol{\tau} \quad (1)$$

Avec $H(\boldsymbol{q})$ la matrice de masse du robot, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ la matrice des effets de Coriolis et centrifuges, $G(\boldsymbol{q})$ le vecteur des actions de gravité et $\boldsymbol{\tau}$ le vecteur des couples d'actionnement.

Pour cela, il va falloir tout d'abord configurer les liaisons pour les piloter en mouvement.

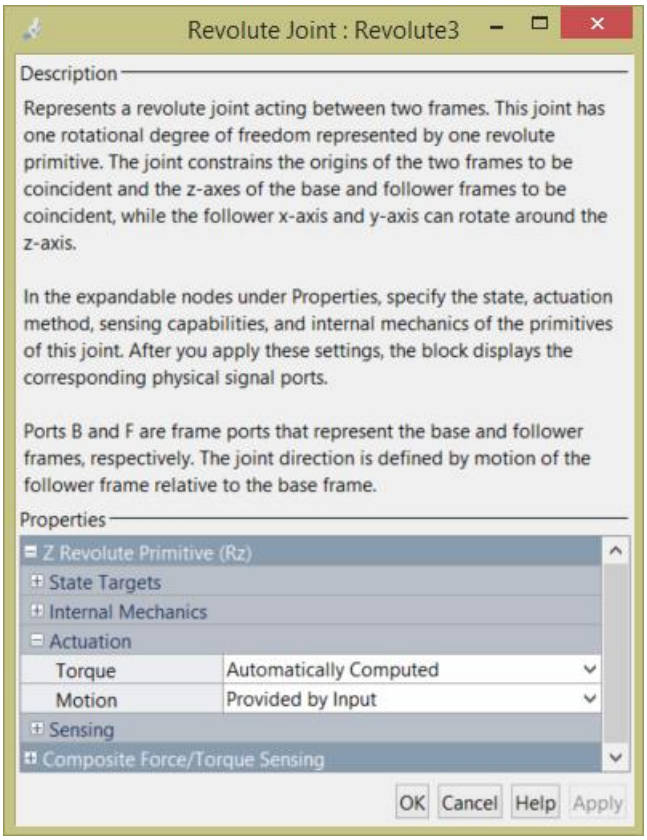

Figure 4 – Définition des données d'actionnement des liaisons

Comme montré figure 4, il faut donc configurer les liaisons pour qu'elles calculent le couple d'actionnement à partir de la simulation et qu'elles prennent en entrée les données de mouvement.

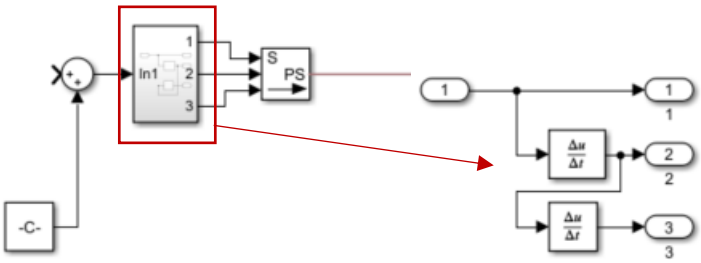

Figure 5 – Architecture de l'actionnement en mouvement

Pour piloter une liaison en mouvement, il sera nécessaire de connecter à l'entrée « q » de la liaison un convertisseur « Simulink to PS » (disponible dans la

librairie SimScape) pour convertir les données simulink vers le système « physique » du robot. Ce bloc devra être paramétré pour recevoir des données de mouvement (en radians ici) ainsi que les deux dérivées de cette valeur (correspondant à $\dot{q}$ et $\ddot{q}$, nécessaires pour pouvoir résoudre l'équation (1)). En supposant que l'entrée est une position, il sera nécessaire de réaliser le sous système présenté figure 5 pour obtenir la vitesse et l'accélération correspondante.

2. Une fois les liaisons correctement paramétrées, il est nécessaire de vérifier le paramétrage du robot. En effet, dans le dossier ressource, vous avez accès au paramétrage du modèle qui a permis la résolution du modèle géométrique inverse. **Ce paramétrage n'est pas forcément le même que celui importé par Multibody !**
   Pour réaliser ce test, il faut dans un premier temps imposer un déplacement nul sur l'ensemble des liaisons et lancer une simulation. La fenêtre de simulation du robot doit apparaitre et vous montrer le robot dans sa configuration de référence. Cette configuration n'est a priori pas celle définie par le paramétrage du modèle géométrique. Afin d'imposer la position de référence au robot, aller explorer le fichier « _Datafile.m » et particulièrement les lignes `smiData.RevoluteJoint(i).Rz.Pos` qui donnent les offsets à imposer au robot pour se trouver en position de référence. Une fois les offsets appliqués sur les différents axes, le robot doit avoir la configuration représentée figure 6.

3. Définir les sens de rotation afin de les faire coller au modèle géométrique : les axes de rotation des liaisons du modèle Multibody sont parfois opposés aux axes définis dans le modèle géométrique. Vérifier ces axes et appliquer un facteur correctif sur l'entrée en explorant le modèle Multibody (depuis le Mechanics explorer). Comparer les orientations des axes des « Revolute » avec la figure de référence de la figure 7.

4. Une fois correctement configurés, les axes doivent avoir en entrée un équipement qui doit ressembler à la figure 8. On peut alors y adjoindre une entrée correspondant à la sortie des scripts de trajectoire. Voir pour cela le bloc simin et sa configuration.

5. On peut alors ajouter un outil de mesure permettant de récupérer la position et l'orientation cartésienne du robot pour la comparer avec la position et l'orientation « objectif ». Pour cela, il faut utiliser le bloc « Transform Sensor ». Afin de pouvoir comparer cette position à celle issue de la cinématique inverse, effectuer le changement de base (bloc « transform ») nécessaire pour obtenir cette mesure.

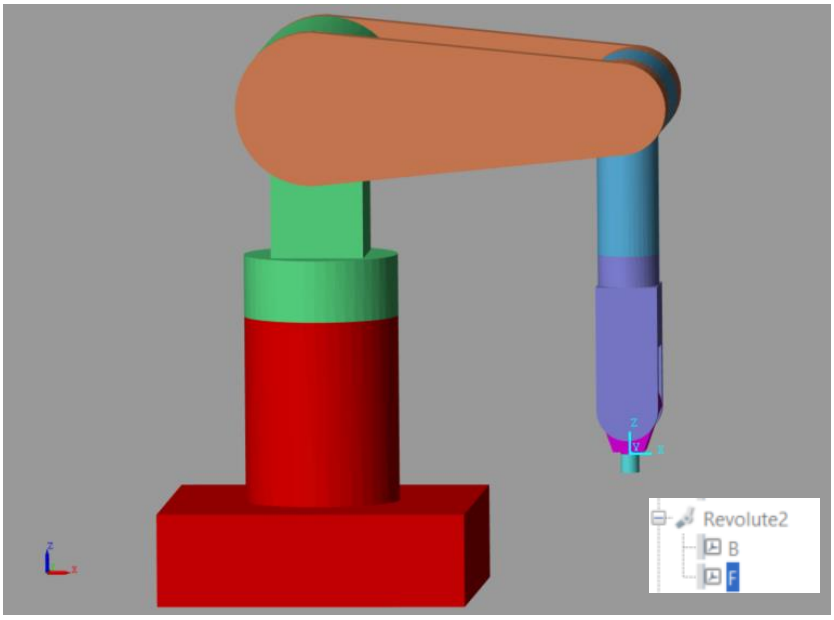

Figure 6 – Configuration de référence du robot

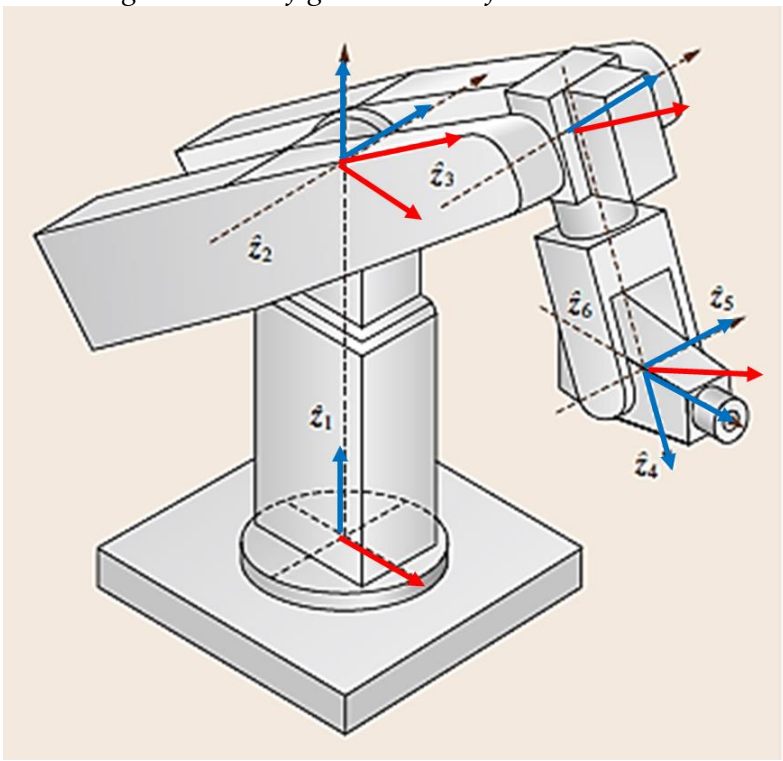

Figure 7 – Axes de rotation tels que définis dans le modèle géométrique.

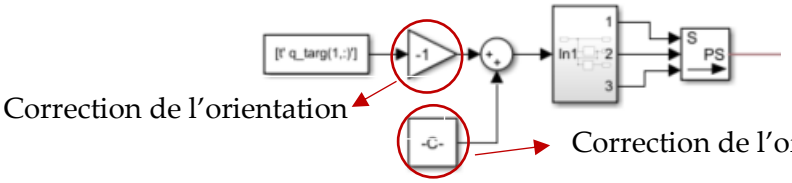

Figure 8 – Définition de l'entrée d'actionnement des liaisons.

6. Dynamique inverse

1. Afin d'affiner le comportement du robot, définir un frottement visqueux sur chacun des axes permettant de respecter les caractéristiques suivantes pour les axes :

Type		Unit	RV-4A
Motion Freedom			6 Axis
Drive Method			AC Servo Motor (with Brake)
Encoder Method			Absolute Encoder
Arm Length (L1 + L2)		mm	500 + 500
Max. Speed	J1	º/s	216
	J2		270
	J3		270
	J4		270
	J5		270
	J6		432
Max. Torque	J1	N.m	400
	J2		270
	J3		90
	J4		30
	J5		3.5
	J6		0.5

Pour faire le choix du frottement visqueux, écrire l'équilibre d'un axe en rotation et justifier votre modèle. Appliquer sur chacun des axes du robot.

2. Adapter les entrées sur les axes afin de suivre les trajectoires angulaires générées à partir des scripts.
3. Vérifier que la trajectoire de l'organe terminal est conforme aux trajectoires théoriques. Pour cela, tracer la trajectoire théorique de l'organe terminal dans R0 et sa trajectoire mesurée (utiliser le bloc « To Workspace »).
4. Réaliser des tests en augmentant la vitesse de parcours et en examinant les couples de sortie sur les différents axes. Définir une vitesse de parcours « limite ».

7. Asservissement du robot

A présent, sur la gamme de vitesses de parcours obtenues à l'étape précédente, on va réaliser une commande réaliste du robot en simulant son comportement en dynamique directe. En d'autres termes, on va chercher à résoudre le problème suivant :

$$\ddot{\boldsymbol{q}} = H(\boldsymbol{q})^{-1}(\boldsymbol{\tau} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - G(\boldsymbol{q})) \quad (2)$$

Avec $H(\boldsymbol{q})$ la matrice de masse du robot, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ la matrice des effets de Coriolis et centrifuges, $G(\boldsymbol{q})$ le vecteur des actions de gravité et $\boldsymbol{\tau}$ le vecteur des couples d'actionnement.

A présent que le robot « fait » correctement ce qu'on lui demande, on va réaliser son asservissement PID sur chacun des axes.

1. Tout d'abord, conservez votre modèle actuel en créant une nouvelle sauvegarde (appelez la « robot asservi » par exemple).

2. Ensuite, reprenez un des axes pour l'actionner non plus en mouvement, mais en couple. Tester la réponse de l'axe en couple (imposer un couple constant en entrée, et regarder la sortie). Justifier le comportement de l'axe. Tester ce comportement sur l'ensemble des axes séparément.

3. Identifiez une fonction de transfert sur chacun des axes, en vous plaçant dans la situation d'inertie la plus défavorable. Déterminez les caractéristiques de ces fonctions et justifiez le résultat.

4. Implémentez un correcteur PID sur chacun des axes et régler, en première approximation, les paramètres du PID en cherchant à obtenir un dépassement inférieur à 10%, une erreur statique nulle et un temps de réponse minimal sur chacun des axes. Veillez à spécifier la condition initiale en position sur chacun des axes (State targets → specify position target).

5. Complétez le schéma en ajoutant une saturation en couple sur chacun des axes. Vérifiez que le comportement des axes reste inchangé.

6. Vérifier pour l'asservissement choisi que le suivi de trajectoire est correct.

7. Tester votre asservissement sur toute la plage de vitesse définie précédemment. Quelle incidence a la vitesse sur le suivi ?

8. Bonus 1

Vous pouvez, en vous inspirant des fichiers de génération de trajectoire, créer de nouvelles trajectoires à suivre et générer de nouveaux mouvements.

9. Bonus 2

Si vous êtes rendu ici, vous avez bien travaillé ☺ Vous pouvez mettre en œuvre un asservissement sur votre système qui soit non plus articulaire mais opérationnel.
En d'autres termes, vous réalisez l'asservissement à partir des coordonnées opérationnelles désirées et réelles du robot et vous reprojetez les efforts obtenus en couples avec la jacobienne du robot.

10. Bonus 3

Jusqu'à présent, le robot ne réalisait de trajectoires que sur un plan virtuel. Nous allons à présent simuler le contact du plan avec la tête de soudage du robot. Pour cela, nous allons définir une liaison sphère/plan entre le centre du poignet du robot et le plan de soudure. Ce dernier sera défini mobile afin de simuler les défauts géométriques dans la profondeur du cordon.

Nous allons utiliser une extension de Multibody qui se nomme « Simscape Multibody Contact Forces Library » et qui est disponible sur le moodle en téléchargement (l'ajouter au path de matlab en exécutant « startup » une fois le fichier dezippe).

1. A partir de la documentation associée à la toolbox, choisir le type de contact le plus pertinent à exploiter dans votre cas.
2. A l'aide de la documentation, placer de manière correcte ce contact dans la simulation que vous avez réalisé jusqu'ici, ou bien à partir du modèle de correction (qui est très mal configuré…). Gardez à l'esprit que le contact doit simuler les défauts du plan de soudure/à la trajectoire de la tête.
3. Piloter la position relative du contact/au référentiel fixe à l'aide d'une liaison de votre choix.
4. Pour piloter cette position, définir un mouvement pour le moment régulier, avec une amplitude de l'ordre du mm et une fréquence peu élevée.
5. Pour piloter la position du plan de manière réaliste, on suppose que la rugosité du cordon est de l'ordre des normes classiques en l'état ($Ra = 1.6\mu m$) et un défaut de planéité de l'ordre du mm (qualité moyenne d'une tôle emboutie d'1mx1m). Définir une fonction permettant de reproduire ces défauts sur le robot.
6. Monitorer les efforts au contact et analyser leur impact sur les couples moteurs à fournir par les différents moteurs au cours de la simulation.
7. A partir des données de réglage du contact, analyser les valeurs et les modifier pour modifier le comportement de la liaison. Analyser les valeurs obtenues et justifier vos choix.

8. Modifier les paramètres des asservissements et conclure sur leur influence.