COMMANDE ARTICULAIRE D'UN ROBOT MANIPULATEUR DE TYPE RP

Cette manipulation concerne l'étude simulée, sous MATLAB-SIMULINK, de diverses stratégies pour la commande articulaire en position d'un bras manipulateur élémentaire de type RP.

I Présentation du robot

Dans tout le texte, l'espace est muni d'un repère orthonormé direct $\mathcal{R}_0 = (O_0, x_0, y_0, z_0)$, où x_0 est un axe horizontal et y_0 est un axe vertical ascendant. Les unités des diverses grandeurs mentionnées sont celles du Système International. Les conventions adoptées pour les représentations temporelles et fréquentielles des signaux sont classiques, *i.e.* au signal temporel $x(t)$ correspond la transformée de Laplace $X(p)$. Le sigle $\mathcal{U}(t)$ désigne la fonction échelon unité.

On considère le robot manipulateur élémentaire à chaîne cinématique ouverte présenté Figure 1. Celui-ci est constitué d'une liaison rotoïde L1 pivotant autour de l'axe z_0, et dont la configuration est paramétrée par la coordonnée généralisée q_1. Le corps C1 en aval de L1 est le support d'une liaison prismatique L2. L'organe terminal, rigidement fixé au corps C2, est repéré par le point O_3. Pour simplifier les expressions mathématiques, la coordonnée généralisée q_2 permettant de paramétrer la liaison L2 est la distance O_0O_3.

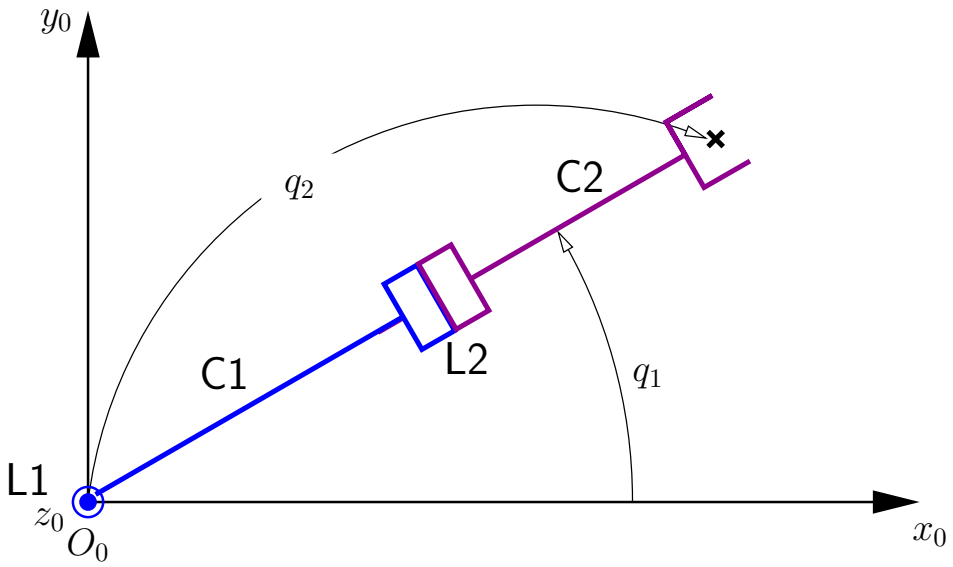

Figure 1 – Robot RP

Le but est d'asservir les variables articulaires $q_1(t)$ et $q_2(t)$ à des lois horaires $q_1^\star(t)$ et $q_2^\star(t)$ préalablement définies. Ces profils, relativement classiques, correspondent à l'enchaînement d'un mouvement uniformément accéléré, d'un mouvement à vitesse constante, et d'un mouvement uniformément décéléré. Ils ont été calculés de telle sorte qu'en l'instant initial t_0 et en l'instant final t_1 les consignes vérifient les conditions : $q_1^\star(t_0) = 0$, $q_1^\star(t_1) = \frac{\pi}{2}$, $q_2^\star(t_0) = 0.5$, $q_2^\star(t_1) = 1$.

Dans toute la suite, on désigne par $'$ l'opérateur de transposition, et on note $q(t) \triangleq (q_1(t), q_2(t))'$, $q^\star(t) \triangleq (q_1^\star(t), q_2^\star(t))'$, *etc.*

I.1 Modèle Dynamique Inverse du robot

Les calculs peuvent être significativement simplifiés en désignant par m la « masse équivalente » du robot localisée en O_3 et par $\Gamma \triangleq (\Gamma_1, \Gamma_2)'$ l'effort généralisé – au sens où Γ_1 désigne un couple et Γ_2 une force – appliqué par le robot sur celle-ci.
On montre alors que lorsque le robot n'est pas en contact avec l'environnment, le modèle dynamique inverse s'écrit

$$\Gamma(t) = D(q(t))\ddot{q}(t) + B(q(t), \dot{q}(t)) + G(q(t)), \tag{1}$$

où

- la matrice dynamique $D(q)$ est de la forme

$$D(q) = \begin{pmatrix} mq_2^2 & 0 \\ 0 & m \end{pmatrix}; \tag{2}$$

- les efforts centrifuges et de Coriolis sont regroupés dans

$$B(q, \dot{q}) = \begin{pmatrix} 2mq_2\dot{q}_1\dot{q}_2 \\ -mq_2\dot{q}_1^2 \end{pmatrix}; \tag{3}$$

- $G(q)$ désigne les efforts gravitationnels, avec

$$G(q) = -\begin{pmatrix} mq_2 g_y \cos(q_1) \\ mg_y \sin(q_1) \end{pmatrix}, \tag{4}$$

où g_y désigne le champ de potentiel gravitationnel en O_3 exprimé selon l'axe y_0.

Dans tout le sujet, on admet que m vaut 15 kg.

I.2 Modélisation des actionneurs lorsque le robot est commandé en vitesse

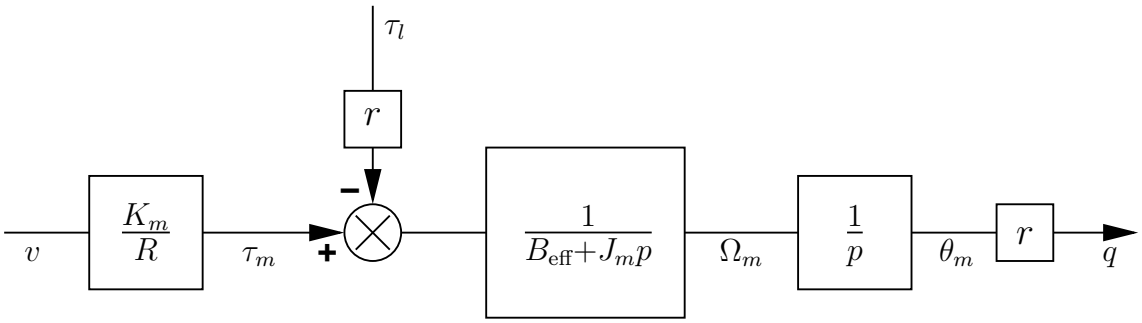

Figure 2 – Schéma des actionneurs

Les moteurs à courant continu utilisés pour mouvoir les deux liaisons sont semblables. Ils admettent un inducteur à aimants permanents, ainsi qu'un induit bobiné d'inductance négligeable. De même, des réducteurs de rapports identiques sont utilisés entre les axes moteurs et les liaisons. Ainsi, un schéma tel que celui présenté Figure 2 peut être dressé pour chaque axe, où

- v désigne la tension d'induit ;
- Ω_m et Θ_m sont les vitesse et position de l'axe moteur ;
- r est le rapport de réduction ;
- τ_m et $r\tau_l$ représentent respectivement l'effort moteur et l'effort exercé par la structure mécanique du robot, exprimés au niveau de l'axe du moteur de la liaison concernée.

Les valeurs numériques des coefficients sont :

- rapport du gain en vitesse du moteur et de la résistance de l'induit : $\frac{K_m}{R} = 0.3$;
- coefficient de frottements efficace de l'ensemble moteur-réducteur : $B_{\text{eff}} = \frac{1}{80}$;
- inertie de l'ensemble moteur-réducteur : $J_m = \frac{1}{100}$;
- rapport de réduction : on considèrera les trois cas $r \in \{\frac{1}{200}, \frac{1}{30}, 1\}$.

I.3 Bibliothèque SIMULINK

Sous MATLAB-SIMULINK, l'invocation de `lib_robotRP` permet d'ouvrir une librairie comprenant un bloc représentant le robot RP commandé en vitesse, un bloc générateur des profils de consigne, deux blocs – liés à des "S-functions SIMULINK" devant être complétées – permettant d'expérimenter quelques lois de commande non linéaires sur ce procédé.

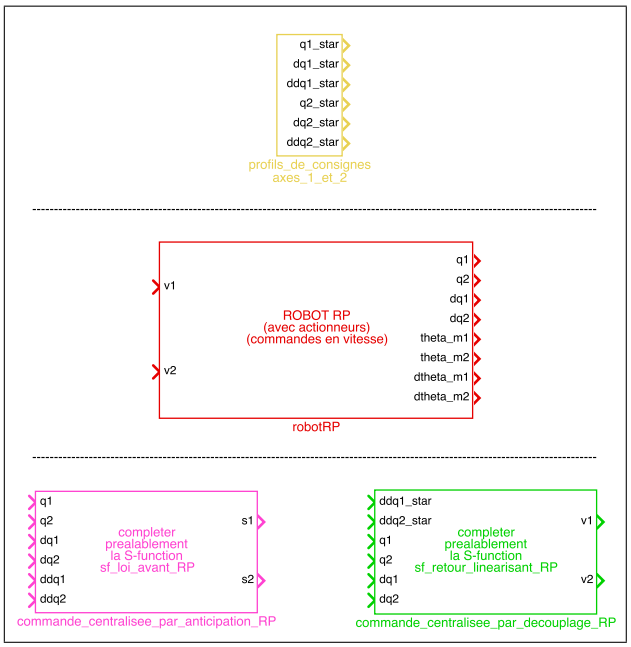

Figure 3 – Librairie `lib_robotRP`

II Travail demandé

II.1 Prise en main de l'outil de simulation et calculs préliminaires

1. Lancer MATLAB et SIMULINK. Ouvrir la librairie `lib_robotRP`. Ouvrir un diagramme SIMULINK vierge, et recopier dans celui-ci les blocs `profils_de_consigne_axes_1_et_2` et `robot_RP`. Visualiser les boîtes de dialogue de ces blocs, ainsi que leur contenu interne (cf. menu *look_under_mask*). Relever l'évolution temporelle des profils de consigne.
2. On considère séparément chacun des rapports de réduction $r \in \{\frac{1}{200}, \frac{1}{30}, 1\}$. Quel que soit l'axe $i \in \{1, 2\}$, la commande sera synthétisée à partir du schéma de principe présenté Figure 4, où les « inerties efficaces » s'écrivent
$$J_{\text{eff}1} = J_{\text{eff}2} = J_m + r^2 m, \tag{5}$$
et où les signaux d_1, d_2 (qui en toute rigueur s'écrivent comme des fonctions distinctes non linéaires en $q_1, q_2, \dot{q}_1, \dot{q}_2$) sont assimilés à des perturbations et prennent des valeurs constantes lorsque le robot est à l'arrêt.

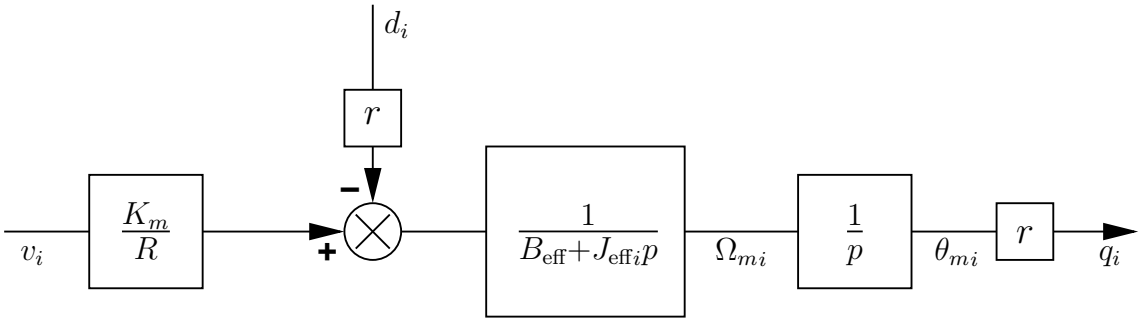

Figure 4 – Schéma utilisé pour la synthèse de la commande articulaire de l'axe i

II.2 Commande en vitesse de type PD

3. Établir l'expression analytique des coefficients K et K_D de la loi de commande Proportionnelle Dérivée $V_i(p) = K(\Theta^\star_{mi}(p) - \Theta_{mi}(p)) - K_D p\Theta_{mi}(p)$ qui, lorsqu'elle est appliquée au modèle précédent pour $d_i(t) \equiv 0$, permet de conférer à la boucle fermée un amortissement unité ainsi qu'une erreur de vitesse ε_{1i} donnée en réponse à une consigne rampe $\theta^\star_{mi}(t) = \theta^1_{mi} t\mathcal{U}(t)$.
4. Pour chaque rapport de réduction $r \in \{\frac{1}{200}, \frac{1}{30}, 1\}$:
   (a) Calculer les valeurs numériques des coefficients K et K_D, de telle sorte que $\frac{\varepsilon_{1i}}{\theta^1_{mi}} = \frac{1}{2}$.
   (b) Tracer les lieux de transfert de la boucle ouverte avant correction, ainsi qu'après introduction du contrôleur précédemment déterminé. Mesurer les marges de stabilité.
   (c) Sous SIMULINK, simuler les réponses temporelles du modèle linéaire représentant la boucle fermée pour des consignes échelons de position et de vitesse, en l'absence de la perturbation rd_i. Vérifier que le modèle admet le comportement souhaité.
   Répéter ces simulations pour une perturbation rd_i constante.
   (d) Créer un nouveau diagramme SIMULINK en remplaçant l'approximation linéaire précédente de l'ensemble {actionneurs,robot} par le modèle non linéaire `robot_RP`. Introduire le correcteur précédent, et placer une consigne fonction des profils de $(q^\star_1(t), q^\star_2(t))'$.
   Simuler la réponse du robot, en présence et en l'absence de gravitation, et exploiter les résultats.
5. Conclure sur la mise en place de la commande PD précédemment synthétisée : validité de l'approximation linéaire de l'ensemble {actionneurs,robot} introduite Figure 4, effet de la gravitation, *etc.*

II.3 Commande en vitesse de type PID

On se place dans l'hypothèse où $r = \frac{1}{200}$, et on met en place une commande PID de la forme $V_i(p) = K\frac{1+T_I p}{T_I p}(\Theta^\star_{mi}(p) - \Theta_{mi}(p)) - K_D p\Theta_{mi}(p)$.

6. À partir d'un lieu de transfert adéquat, proposer une valeur de T_I qui permette d'améliorer la précision de la boucle fermée sans dégrader sa stabilité.
   Pour ce choix de T_I, implémenter la commande sur le diagramme SIMULINK utilisé en 4d. Simuler le comportement obtenu. S'il y a lieu, retoucher « à la main » la valeur de T_I sur l'un des deux axes de façon à obtenir un comportement assez satisfaisant.

II.4 Retour sur la modélisation

Cette section vise à établir le modèle de la Figure 4, utilisé dans les questions précédentes, sur la base des modèles élémentaires détaillés dans les Sections I.2 et I.1.

7. Dans la Figure 4, les « inerties efficaces » (constantes) $J_{\text{eff}1}, J_{\text{eff}2}$ capturent les inerties des actionneurs ainsi que des efforts inertiels imposés par la structure mécanique du robot (ramenés aux axes

des actionneurs). Combiner les modèles des §I.2 et §I.1 de façon à faire apparaître des expressions $J_{\text{eff}i}(q)$ et $rd_i(q)$ fonctions des coordonnées articulaires du robot.

8. Pour chacun des axes i et pour chaque valeur considérée de r, calculer les inerties efficaces extrêmes $(J_{\text{eff}i})_{\text{MIN}}$ et $(J_{\text{eff}i})_{\text{MAX}}$, puis justifier quelle valeur doit être retenue pour $J_{\text{eff}i}$.

II.5 Commande non linéaire centralisée par anticipation

Le but de cette section est de compléter la loi de commande PD proposée au §II.2 par un terme feedforward (loi avant) permettant d'obtenir un suivi satisfaisant du profil de consigne. Pour cela, on introduira dans le schéma de la boucle fermée utilisé en 4d le bloc `commande_centralisee_par_anticipation_RP` permettant d'implémenter la loi avant. Ce bloc est lié à la "S-function" `sf_loi_avant_RP.m`, qui peut être obtenue par recopie du prototype `sf_loi_avant_RP_VERSION_ETUDIANT.m`.

9. Déterminer l'expression théorique du terme feedforward à mettre en place.
10. Compléter le code de la "S-function" `sf_loi_avant_RP.m` (cf. Figure 5) de façon à implémenter cette loi de commande.
11. Pour chacun des rapports de réduction $r \in \{\frac{1}{200}, \frac{1}{30}, 1\}$, la comparer aux stratégies implémentées précédemment, et discuter de leurs avantages et inconvénients respectifs.

II.6 Commande non linéaire centralisée par découplage

Le robot demeurant décrit par `robot_RP`, on souhaite mettre en place une commande en boucle fermée linéarisante découplante. Celle-ci est constituée de deux boucles imbriquées : une boucle interne non linéaire (bloc `commande_centralisee_par_decouplage_RP`) et un retour externe Proportionnel Dérivé.

12. Déterminer l'expression théorique de la commande en boucle fermée recherchée.
13. Compléter le code de la "S-function" `sf_retour_linearisant_RP.m` –obtenue par recopie du prototype `sf_retour_linearisant_RP_VERSION_ETUDIANT.m`– (cf. Figure 6) liée au bloc `commande_centralisee_par_decouplage_RP` de façon que pour chaque axe, le transfert de l'asservissement interne soit un double intégrateur.
14. Proposer un rebouclage externe Proportionnel Dérivé adéquat.
15. Pour chacun des rapports de réduction $r \in \{\frac{1}{200}, \frac{1}{30}, 1\}$, simuler le système asservi. Vérifier ses propriétés de linéarité et de découplage.

```
function [sys,x0,str,ts] = sf_loi_avant_RP(t,x,u,flag)
% NE PAS MODIFIER CI-DESSOUS ; CF. SEULEMENT mdlOutputs PLUS BAS
%
switch flag,
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 3,
    sys = mdlOutputs(t,x,u);
  case {1,2,4,9}
    sys = [];
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
%=============================================================================
function [sys,x0,str,ts] = mdlInitializeSizes
% NE PAS MODIFIER CI-DESSOUS ; CF. SEULEMENT mdlOutputs PLUS BAS
%
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;               % [d1;d2]
sizes.NumInputs      = 6;               % [q1;q2;dq1;dq2;ddq1;ddq2]
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [-1 0];                           % période héritée du bloc père
%=============================================================================
function sys = mdlOutputs(t,x,u)
%
q1 = u(1); q2 = u(2); dq1 = u(3); dq2 = u(4); ddq1 = u(5); ddq2 = u(6);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% À COMPLÉTER À PARTIR D'ICI %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sys DOIT ÊTRE INTIALISÉ AVEC LE CONTENU DU VECTEUR DE SORTIE DU BLOC
% e.g.
sys = [0;0];
%=============================================================================
```

Figure 5 – Code de la "S-function" `sf_loi_avant_RP.m`

```
function [sys,x0,str,ts] = sf_retour_linearisant_RP(t,x,u,flag)
% NE PAS MODIFIER CI-DESSOUS ; CF. SEULEMENT mdlOutputs PLUS BAS
%
switch flag,
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 3,
    sys = mdlOutputs(t,x,u);
  case {1,2,4,9}
    sys = [];
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
%=============================================================================
function [sys,x0,str,ts] = mdlInitializeSizes
% NE PAS MODIFIER CI-DESSOUS ; CF. SEULEMENT mdlOutputs PLUS BAS
%
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;               % [d1;d2]
sizes.NumInputs      = 6;               % [q1;q2;dq1;dq2;ddq1;ddq2]
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [-1 0];                           % période héritée du bloc père
%=============================================================================
function sys = mdlOutputs(t,x,u)
%
ddq1_star = u(1); ddq2_star = u(2);
q1 = u(3); q2 = u(4); dq1 = u(5); dq2 = u(6);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% À COMPLÉTER À PARTIR D'ICI %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sys DOIT ÊTRE INTIALISÉ AVEC LE CONTENU DU VECTEUR DE SORTIE DU BLOC
% e.g.
sys = [0;0];
%=============================================================================
```

Figure 6 – Code de la "S-function" `sf_retour_linearisant_RP.m`