Département Génie Electrique et informatique

Travaux pratiques d'automatique

4ème année AE

Semestre 2

v2.0

Table des matières

MANIPULATION 1

Recherche de chemins dans un graphe : LegoRover

Le robot legot EV3

1.1 But de la manipulation

L'objectif de cette manipulation est de commander un robot Lego sur 2 roues motrices et une roue libre pour l'équilibre, dont la commande est semblable à celle d'un rover (Figure 1), de manière :

- à ce qu'il suive de manière efficace une ligne noire au sol pour guider sa trajectoire (commande de bas niveau de type commande continue : Niveau 0)
- à ce qu'il choisisse correctement un chemin à suivre en fonction d'un critère et de contraintes données (commande de haut niveau type Intelligence Artificielle : Niveau 2)
- à ce qu'il suive correctement le chemin voulu (commande type contrôleur d'exécution : Niveau 1)

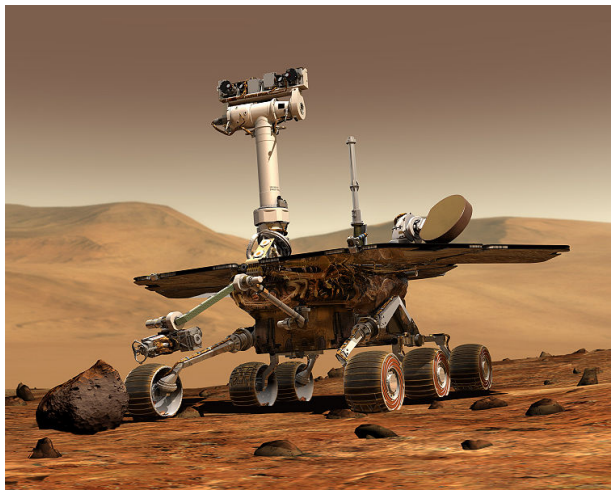


FIGURE 1 – Le rover Curiosity

Ces différents niveau de commande correspondent à différents champs du domaine de l'Automatique en générale, et sont illustrés sur la Figure 2.

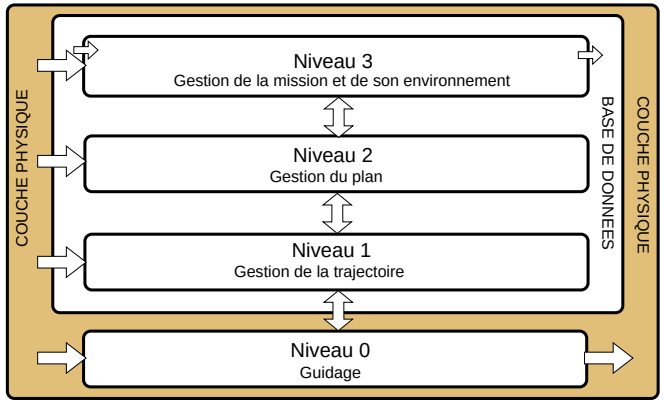


FIGURE 2 – Les différents niveaux de commande dans une architecture d'autonomie

1.2 Présentation de la maquette

On travaille avec un robot Lego monté sur 2 roues et une roue libre pour l'équilibre représenté sur la Figure 16.

Le capteur utilisé pour effectuer le suivi de ligne est un capteur LineLeader (Figure 4). Il est composé de 8 couples photodiodes/phototransistors, qui, après calibration, renvoient un bit à 1 lorsque le couple est devant la couleur noire (Figure 5). Une fonction a été développée pour renvoyer une somme pondérée suivant la position du capteur par rapport à une ligne noire (voir fonction CapteurLigne dans PilotRobert.java).

Le fonctionnement du robot est géré par un microprocesseur ARM9.

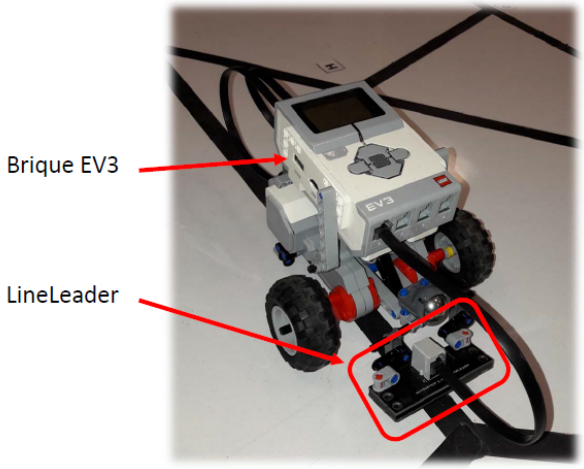


FIGURE 3 – le robot Lego

Le robot se déplace sur une carte constituée de lignes noires et de points identifiés.

1.3 Présentation du dossier TPTomTom

Télécharger dans votre répertoire le dossier TP TomTom contenu sous moodle. Ouvrir le fichier MiseEnPlace-TPTomTom.pdf, suivre la mise en place du projet TPTomTom jusqu'à la fin. Le projet EV3 correspond aux fichiers de programmation du Robot Lego. Le code généré sera embarqué dans la brique du robot. Le projet PC correspond aux fichiers de programmation du PC qui va communiquer avec le robot.

1.4 Présentation des fichiers EV3

Le projet EV3 contient les fichiers suivants :

- `Pilote.java` : Fonction main de la classe Pilote, création d'une mission, démarrage ;
- `PilotRoberto.java` : guidage du robot, régulation

FIGURE 4 – le capteur LineLeader

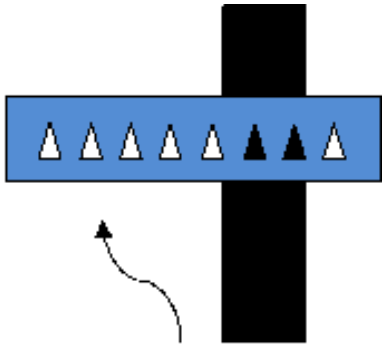


FIGURE 5 – Schéma de principe du suivi de ligne

- `MissionRobot.java` : traite les missions envoyées par le PC et reçues par le robot ;
- `CommBTRobot.java` : gestion de la communication Bluetooth avec le PC ;
- `Message.java` : gestion des messages envoyés par le PC et reçus par le robot ;

1.5 Présentation des fichiers PC

Le projet PC contient les fichiers suivants :

- `ShortestPath.java` : classe principale du PC. Mise en place de la carte, lancement de l'interface graphique, lancement du Dijkstra, construction de la mission, lancement de la

mission sur le robot ;
- `test_BasNiveau.java` : programme de test pour la partie guidage. Test sur 3 points, pour valider le guidage.
- `Dijkstra.java` : code de l'algorithme de Dijkstra ;
- `CommBTPC.java` : gestion de la communication Bluetooth avec le robot Lego ;
- `InterfaceDialogue.java` : interface graphique ;
- `Map.java` : gestion de la carte ;
- `Message.java` : construction des messages entre le PC et le robot ;
- `MissionPC.java` : gestion de la mission ; séquencement des ordres envoyés au robot ;
- `Order.java` : classe ordre à envoyer au robot ;
- `Point.java` : classe point ;

1.6 Mise en place de la partie Guidage

Dans un premier temps, l'objectif est de mettre en place la commande de bas niveau sur le robot, c'est-à-dire le guidage autour d'une ligne noire, suivant le schéma de la Figure 6.

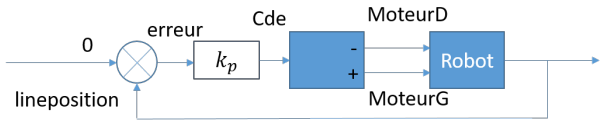


FIGURE 6 – Schéma de régulation

- Expliquer le schéma de régulation. Pourquoi les commandes envoyées sur les deux moteurs sont différentes ?
- Compléter le fichier `PilotRoberto.java` de façon à mettre en place la boucle de régulation proportionnelle autour d'une consigne en vitesse notée `speed` et d'une position centrale sur la ligne noire, sachant que l'erreur et la commande sont de type (float). La valeur de kp est fixée et vaut 10.
- Tester votre régulation en réel avec le fichier `test_BasNiveau.java` avec une valeur de vitesse constante réglée à 300°/s .
- Faire varier la vitesse entre 0 et 500 de manière à déterminer la plage de valeurs acceptable pour la vitesse.
- Faire varier Kp entre 5 et 40 par exemple de manière à mettre en évidence les caractéristiques d'un régulateur proportionnel.

Une fois la commande de bas niveau (Niveau 0 : guidage) validée, on va mettre en place la commande de niveau 1 : gestion de la trajectoire.

1.7 Mise en place de la partie Gestion de la trajectoire

Cette partie consiste à vérifier que la gestion de la trajectoire est bien effectuée sur le robot. Pour cela, on se contente de valider le bon séquencement d'un mission écrite "à la main".

- En reprenant le fichier `test_BasNiveau.java`, créer à la main une mission allant du point A au point Q sur la carte, avec une vitesse de 360°/s en utilisant des commandes du type : `listOfOrders.add(new Order(int angle, int distance, int speed))` ; avec les angles en degrés, la distance en mm et la vitesse en degrés par seconde.
- Valider cette mission sur le robot.

1.8 Mise en place de la partie Gestion du plan

Cette partie consiste à élaborer une stratégie pour l'élaboration d'un plan de mission. Pour cela, on est amené à définir un graphe dont les sommets sont les points visitables et les arcs les chemins entre ces points..

- Formaliser le problème d'optimisation que vous allez tenter de résoudre pour élaborer un chemin pour le robot en établissant un critère d'optimisation, en vous inspirant de l'annexe intitulée "Théorie des graphes - Algorithme de Dijkstra".

- Compléter la fonction `SetArc` dans le fichier `Map.java` de manière à définir le poids des arcs dans le graphe suivant le critère choisi.
- Proposer sur le papier une méthode de recherche de chemin intuitive, par exemple un algorithme glouton qui permettrait de trouver un chemin entre un point de départ et un noeud fin.
- Ouvrir le fichier `Dijkstra.java`. Etudier le code de manière à retrouver les étapes du pseudo-algorithme donné en annexe.
- Compléter le fichier `Dijkstra.java` :
  - méthode `Trouvmin()` qui retourne le sommet le plus proche du sommet initial : compléter la boucle de recherche du nœud de valeur minimale non encore exploré dans `tab_value`.
  - méthode `MajCcum(int noeud_retenu)` qui met à jour `tab_value` et `tab_noeuds`.
- Tester cet algorithme en utilisant un critère de distance (plus court chemin). Note : penser à décommenter la ligne 53 dans `ShortestPath.java` (ligne BuildPath.ComputeDijkstra).
- On s'intéresse maintenant à faire un chemin le plus rapide en terme de temps de trajet. Modifier la fonction `SetArc` dans le fichier `Map.java` de manière à adapter l'algorithme.
- Mettre en évidence un changement suite à ce changement de critère en changeant éventuellement les vitesses sur la carte.

1.9 Extensions possibles

Cette partie vise à étendre l'algorithme pour prendre en compte d'autres critères ou d'autres façon de modéliser le problème. Par exemple,

- Proposer une solution permettant au robot de partir d'un noeud initial, d'arriver à un noeud fin, en passant par des points particuliers précisés et non ordonnés, de manière optimale (typique d'une tournée de facteur par exemple).
- Proposer une solution si les ressources du robot sont limitées et qu'il peut par exemple effectuer une distance limitée (carburant limité).
- On imagine maintenant un rover sur mars, qui doit effectuer des points de mesures de plus ou moins grande importance, avec une quantité de ressource limitée. Imaginer un algorithme permettant de résoudre sur problème en maximisant les récompenses sur les points de mesures tout en garantissant que les ressources seront suffisantes.

MANIPULATION 2

ASNL : Méthode du plan de phase

Etude d'un asservissement de position à relais

2.1 But de la manipulation

Le but de cette manipulation est d'étudier sous Matlab, le comportement d'un système commandé par un régulateur tout-ou-rien, avec seuil, et/ou hystérésis (Figure 7), par la méthode du plan de phase.

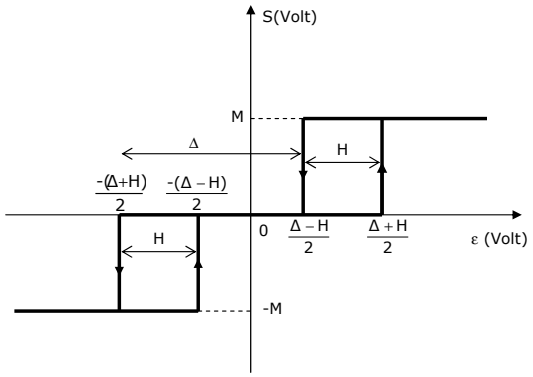


FIGURE 7 – Élément non linéaire

L'intérêt principal du régulateur tout-ou-rien réside dans sa simplicité. Un relais ou un simple interrupteur peuvent servir à matérialiser le dispositif de commande. L'inconvénient est souvent une usure prématurée des composants due aux changements brusques et rapides de la commande. Afin de vérifier certains résultats donnés dans la partie du cours de systèmes asservis non linéaires qui a trait à cette méthode, on simulera un tel système avec Simulink. On tracera les trajectoires dans le plan de phase du système. On évaluera également s'il y a lieu, l'amplitude et la période des auto-oscillations pour différentes valeurs des paramètres (seuil et hystérésis du relais, taux de contre-réaction tachymétrique, conditions initiales).

2.2 Rappels de cours

Généralités sur la méthode du Plan de Phase

La méthode du plan de phase permet d'étudier des systèmes dont le modèle est non linéaire par une méthode dans l'espace d'état (à l'opposé de la méthode du 1er harmonique qui est une méthode fréquentielle). En pratique, cette méthode se limite aux systèmes du deuxième ordre. Au-delà, la représentation graphique est impossible.

Soit un système physique à un degré de liberté régi par une équation différentielle du deuxième ordre qui s'écrit :

$$\ddot{x} = f(x, \dot{x}) \tag{1}$$

si on pose : $\dot{x} = y$ l'équation (1) devient équivalente au système :

$$\begin{cases} \dot{x} = y \\ \\ \dot{y} = f(x, y) \end{cases} \tag{2}$$

L'évolution dépend donc de deux paramètres, la position x et la vitesse $\dot{x}$ du système, que l'on peut définir comme les deux variables d'état du système $X = (x_1 = x, x_2 = \dot{x})$.

- L'état du système est caractérisé dans le plan $(x, v = \dot{x})$, appelé **plan de phase** par le point P de coordonnée $(x_1, x_2) = (x, \dot{x})$
- L'évolution du système en fonction du temps pour des conditions initiales données est décrit par la trajectoire dite **trajectoire de phase** du plan P dans le plan de phase.

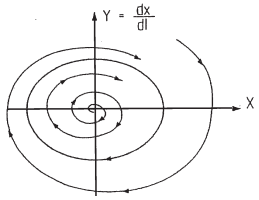


FIGURE 8 – Exemple de trajectoire de phase

— L'ensemble des trajectoires de phase correspondant aux diverses conditions initiales $(x_0, \dot{x}_0)$ en P_0 permises, constitue le **portrait de phase** du système.

Notion de points singuliers

Tout point tel que $\dot{X} = 0$ est appelé un **point fixe**, **point critique** ou encore **point d'équilibre**. Un couple (x_1, x_2) tel que $\dot{X} = 0$ est un point d'équilibre, il peut être stable ou instable. Ces points constituent pour les trajectoires des points singuliers. Pour tout autre point qu'un point singulier, il n'existe qu'une trajectoire de phase qui passe par ce point.

Notion de cycle limite

Les cycles limites sont les trajectoires dans le plan de phase qui correspondent aux «oscillations limites» du système, solutions périodiques vers lesquelles tendent les trajectoires pour toutes les conditions initiales situées dans une certaine région du plan de phase. Rappelons le caractère essentiellement non linéaire de tels phénomènes. La figure 8 montre un exemple de cycle limite stable vers lequel convergent les trajectoires, divergeant à partir d'un foyer instable et convergeant à partir des régions éloignées du plan de phase.

Dans le cas des asservissements du 2ème ordre avec des non linéarités de type discontinu (relais, frottement sec), on peut considérer le système comme linéaire par morceaux. L'intégration de l'équation (2) est relativement facile à faire par des méthodes analytiques. On obtient, pour les différents états de cette non linéarité, une équation non paramétrique indépendante de t de la forme $f(x, y) = 0$ définissant plusieurs types de trajectoires élémentaires. La construction de la trajectoire globale sera faite à partir de ces trajectoires élémentaires raccordées en des «points de commutation» correspondant au changement d'état de la discontinuité. On montre ainsi que dans certains cas (relais avec hystérésis par exemple), la trajectoire obtenue tend vers un cycle limite stable ; on peut alors étudier la stabilité d'un tel cycle, et en calculer l'amplitude et la période par la méthode des transformations ponctuelles de Poincaré.

Propriétés particulières liées à la forme $\dot{x}_1 = x_2$

Sens de parcours des trajectoires de phases : Étant donné que x augmente dans le domaine pour lequel $\dot{x} > 0$ et que x diminue dans le domaine associé à $\dot{x} < 0$, les trajectoires de phases sont parcourues dans le sens des aiguilles d'une montre.

Intersection de l'axe Ox avec les trajectoires de phases : Lorsque la trajectoire de phase coupe l'axe des x, soit elle est perpendiculaire à l'axe Ox, soit elle passe par un point singulier. En effet, la tangente à la trajectoire de phase peut être définie comme la droite faisant un angle α avec l'axe Ox tel que :

$$tan\alpha = \frac{dv}{dx} = \frac{\frac{dv}{dt}}{\frac{dx}{dt}} = \frac{f(x,v)}{v}$$

car par définition $\frac{dv}{dt} = f(x,v)$. On a alors deux cas :
- $f(x,v) \neq 0$ on a alors une tangente verticale
- $f(x,v) = 0$ et on a un point singulier (forme indéterminée).

Position d'équilibre dans le plan de phase : Les positions d'équilibre sont définies par $v = \dot{x} = 0$ et $\ddot{x} = \dot{v} = 0$, et sont donc sur l'axe des abcisses.

Notion de droites de commutation

La commutation de l'élément non linéaire a lieu lorsque l'entrée ϵ est égale à des valeurs caractéristiques. A ce moment-là, la valeur de sortie du relais bascule. Cela correspond dans le plan d'état à deux droites décalées parallèlement. Par exemple, dans le cas d'un seuil pur, le relais commute pour des valeurs $\epsilon = \pm\frac{\Delta}{2}$. Or on peut exprimer la valeur de ϵ en fonction de l'entrée et de la sortie. Cette équation définit donc deux droites de commutation : lorsque la trajectoire coupe une des droites, le relais commute.

2.3 Manipulation

Le schéma bloc de l'asservissement à étudier est donné par la figure 9.

- Les paramètres physiques ont été identifiés

$$\begin{array}{ll} K_m = 38.57 rad/V.s & K_s = 1.57 V/rad \\ K_g = 0.23 V.s/rad & \tau = 0.27 s \end{array} \quad (3)$$

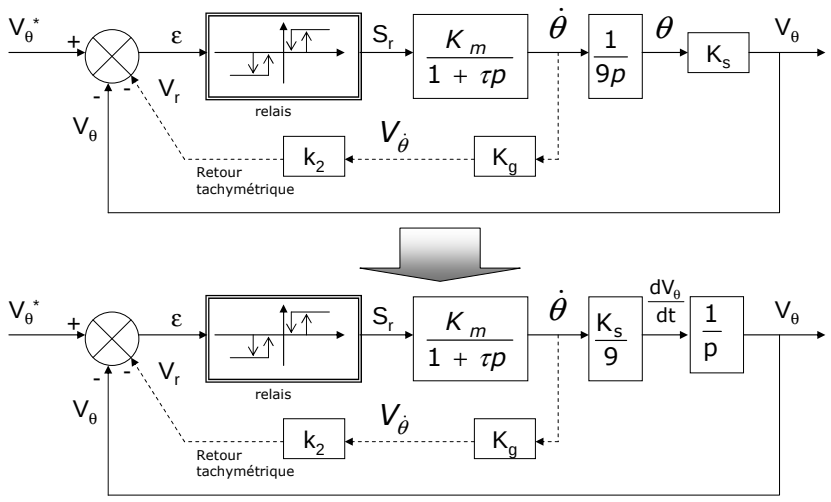


FIGURE 9 – Schéma bloc de l'asservissement

- k_2 est un paramètre ajustable pour régler la correction tachymétrique.
- L'élément non linéaire est constitué par un relais avec seuil et hystérésis dont la caractéristique $S_r = F(\epsilon)$ est représentée par la figure 7. Le niveau de la tension de sortie du relais qui alimente le moteur est de $\pm M = \pm 10V$.

Remarque 1 *: Afin de faciliter l'étude dans le plan de phase de ce système simulé, on exprimera les angles en radians, les tensions en volts et le temps en secondes ; De plus on supposera le système autonome (entrée $V^*_{\dot{\theta}}(t) = 0$), le système étant perturbé de sa position d'équilibre par la condition initiale $V_\theta(0)$, les trajectoires étant représentées dans le plan $(V_\theta(t), dV_\theta/dt)$.*

Étude du système sans correction tachymétrique

Etude du système linéaire :

► Calculer l'expression de la fonction de transfert $V_\theta(p)/V^*_\theta(p)$

sans tenir compte du relais. Quelles sont les caractéristiques du système en boucle fermée ? (amortissement, pulsation propre, erreur statique, dépassement, temps de montée, ...). Sous Simulink, vérifier ces données sur la réponse indicielle du système avec des conditions initiales nulles.

- ▶ Donner sa représentation d'état en posant $X_1 = V_\theta$ et $X_2 = dV_\theta/dt$

Etude du système complet : Rajouter l'élément non linéaire dans le schéma Simulink, paramétrable avec H et Δ.

Les équations différentielles qui régissent le système sont :

$$\begin{cases} \frac{d\dot{V}_\theta}{dt} &= -\frac{1}{\tau}\frac{dV_\theta}{dt} + \frac{K_m K_s S_r}{9\tau} \\ \frac{dV_\theta}{dt} &= \dot{V}_\theta \end{cases}$$

- ▶ Quels sont les conditions d'équilibre pour ce système ?

Etude du seuil pur ($\Delta \neq 0$, $H = 0$)

- ▶ On prend par exemple $\Delta = 5$. Quelle est la condition sur la condition initiale $V_\theta(0)$ pour que le système évolue vers un point d'équilibre ? Montrez-le grâce à Simulink.
- ▶ Quelles sont les caractéristiques de ce système ? (erreur statique, dépassement, fréquence...)
- ▶ Justifier de manière théorique la forme des courbes : asymptotes, pente, etc.
- ▶ Les droites de commutations sont les droites sur lesquelles l'élément non linéaire commute. Quelles sont les équations des droites de commutation dans ce cas-là ? Montrez-les sur vos courbes.
- ▶ Répéter cette étude en faisant varier la valeur de Δ. Que remarque-t-on sur l'erreur statique ? Le dépassement ? Y-a-t-il un lien avec les droites de commutations ? Conclure.

Etude de l'hystérésis pur ($\Delta = 0$, $H \neq 0$) On effectue une étude similaire pour l'hystéresis pur.

- ▶ On prend par exemple $H = 5$. Y-a-t-il une condition sur $V_\theta(0)$ pour que le système évolue vers un point d'équilibre ? Montrez-le grâce à Simulink. Est-ce que le système va se stabiliser vers un point d'équilibre ? Justifier.
- ▶ Justifier de manière théorique la forme des courbes : asymptotes, pente, etc.
- ▶ Les droites de commutations sont les droites sur lesquelles l'élément non linéaire commute. Quelles sont les équations des droites de commutation dans ce cas-là ? Montrez-les sur vos courbes.
- ▶ Répéter cette étude en faisant varier la valeur de H. Y-a-t-il un lien avec les droites de commutations ? Conclure.

Etude avec le relais hystérésis + seuil ($\Delta \neq 0$, $H \neq 0$)

- ▶ Faire une étude similaire dans ce cas. En particulier, pour $H = 5$, trouver expérimentalement le Δ_{lim} entre le mode oscillant et le mode non-oscillant. De manière duale, pour $\Delta = 5$, trouver expérimentalement H_{lim} entre le mode oscillant et le mode non-oscillant.
- — Quelles sont les équations des droites de commutation dans ce cas-là ? Montrez-les sur vos courbes.
- — Conclure

Remarque 2 *Le choix du nombre d'itérations et de la valeur du pas de calcul devra être un bon compromis entre la précision du tracé et le temps d'exécution.*

Influence de la correction tachymétrique sur le régime transitoire

On met maintenant en place sur le système une boucle interne de contre réaction (k_2 non nul).

- ▶ Modifier votre modèle en conséquence et vérifier son bon fonctionnement.
- ▶ Pour $H = 5$, $\Delta = 0$ (hystérésis pur), trouver les nouvelles équations des droites de commutation. Les idetifier sur vos courbes.
- ▶ Identifier deux valeurs du taux de contre réaction qui correspondent l'un à un régime non-glissant, l'autre à un régime

glissant.

- ▶ Chercher expérimentalement le taux de contre réaction limite et celui du régime optimum (origine atteinte après une seule commutation), par exemple par dichotomie.

2.4 Réalisation sur site réel

- ▶ Pratiquez les résultats obtenus en 2.3 sur le site réel.
- ▶ Observez la commande en régime glissant et en régime optimum.

2.5 Références bibliographiques

- ★ http ://moodle.insa-toulouse.fr/course/view.php ?id=66 - Cours d'ASNL sous Moodle
- ★ C. MIRA : *Cours de systèmes asservis non linéaires*. DUNOD UNIVERSITE), 1969.
- ★ K . OGATA : *Modem control engineering* PRENTICE HALL, 1970.
- ★ J.C. GILLES, M. PELEGRIN : *Systèmes asservis non linéaires tomes 1,2 et 3*. DUNOD AUTOMATIQUE, 1975

MANIPULATION 3

ASNL : Méthode du 1er harmonique

Etude d'un asservissement de position à relais

3.1 Présentation de la manipulation

But de la manipulation

Il s'agit d'étudier un asservissement de position dans lequel l'élément d'amplification est un relais (élément non linéaire). Cette étude porte essentiellement sur la détermination expérimentale du lieu critique du relais, à partir de la connaissance du lieu de transfert de la partie linéaire, et des caractéristiques de l'oscillation de pompage (amplitude et pulsation), mesurées expérimentalement.

Rappels de cours : approximation du ler harmonique

Lorsqu'on applique un signal sinusoïdal à un système non linéaire, on obtient en général une sortie $s(t)$ périodique mais non sinusoïdale.

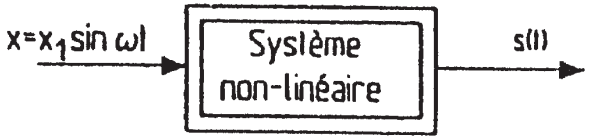


FIGURE 10 – Élément non linéaire

Pour un tel système, on ne peut pas définir une fonction de transfert comme on le fait pour un système décrit par une équation différentielle linéaire à coefficients constants. La sortie $s(t)$ est décomposable en série de Fourier. En ne considérant que le 1er harmonique de $s(t)$: $W_l sin(\omega t + (\phi))$, on définit une fonction de transfert équivalente ou généralisée :

$$module = W_l/X_l \qquad argument = \phi$$

Cette fonction de transfert dépend de la pulsation ω de l'amplitude d'entrée X_l

$$W_l/X_l = B(X_l, \omega) \qquad \phi = \phi(X_l, \omega)$$
$$N(X_l, \omega) = B(X_l, \omega)e^{(j\phi(X_l,\omega))}$$

Un cas très important en pratique est celui où N ne dépend pas de la fréquence, et dépend seulement de l'amplitude du signal d'entrée X_l. C'est le cas d'éléments non linéaires tels que : seuil, saturation, tout ou rien, ... Dans ce cas, la fonction de transfert généralisée $N(X_l)$ s'appelle le gain complexe équivalent.

Étude de la stabilité d'un asservissement non linéaire bouclé

Considérons le système de la figure 11.

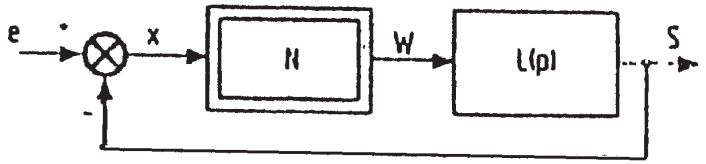


FIGURE 11 – Bouclage non linéaire

La fonction de transfert généralisée en boucle ouverte de ce système est :

$$\frac{s}{x} = N(X_l).L(j\omega)$$

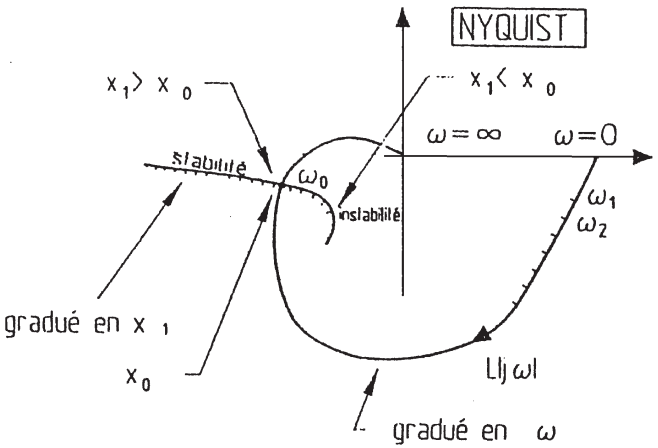


FIGURE 12 – Critère géométrique de détermination d'un cycle limite

Supposons X_l constant ; $N(X_l)$ est alors un nombre fixe (réel ou complexe). En appliquant le critère du revers dans le plan de Nyquist, on peut dire que l'asservissement est stable pour l'amplitude X_l de l'erreur si le lieu de transfert $N(X_l).L(j\omega)$ parcouru dans le sens des ω croissants laisse le point critique - I à gauche.

On peut faire le même raisonnement en considérant la position du lieu $L(j\omega)$ par rapport au point $-\frac{1}{N(X_l)}$

Plus généralement, pour chaque amplitude X_l d'erreur, on peut définir un point critique $-\frac{1}{N(X_l)}$ L'ensemble de ces points constitue le lieu critique de l'élément non linéaire.

Sur ce lieu critique, on peut déterminer des régions pour lesquelles il y a stabilité et des régions pour lesquelles il y a instabilité (voir figure 12)

- Pour $X_l < X_o$ le système est instable. L'amplitude de l'erreur va donc augmenter et on se déplace sur le lieu critique vers X_0.
- Pour $X_l > X_o$ le système est stable, X_l diminue vers X_0.

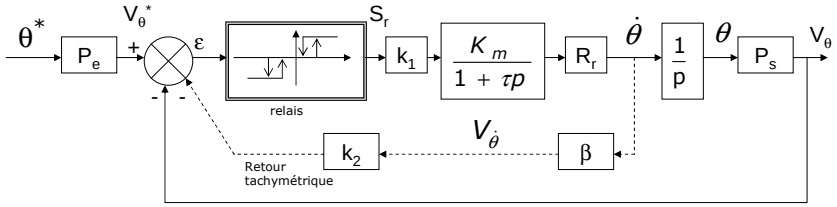


FIGURE 13 – Montage des différents éléments

- A la limite le système oscille avec une amplitude X_0 de l'erreur, à une pulsation ω. Cette oscillation est appelée **pompage**.

3.2 Matériel utilisé

La manipulation comprend un moteur à courant continu étudié en 3ème année, et un relais réglable.

Remarque 3 *: On utilisera une fonction de transfert $G(p)$ pour le moteur assimilable à $\frac{K_m}{p(1+\tau p)}$ avec $K_m = 42.8 rad/s.V$ et $\tau = 0.214 s$.*

3.3 Manipulation

Étude des auto-oscillations. Tracé du lieu critique

Il s'agit de construire expérimentalement le lieu critique correspondant à un relais ayant des caractéristiques déterminées, et de comparer les résultats obtenus avec le lieu critique construit à partir des formules mathématiques étudiées en cours.

Montage

- ▶ Réaliser l'asservissement de position sans le retour tachymétrique représenté par le schéma bloc suivant (Fig. 13) pour lequel :

1. P_e, P_s sont respectivement les gains de conversion (données physiques, tension) de commande et de sortie. P_s vaut $1.58V/rad$;
2. R_r est un réducteur de rapport $1/9$;
3. k_1 est un potentiomètre atténuateur (10 tours) ($0 < k_1 < 1$) qui permet de faire varier le gain en boucle ouverte de la partie linéaire ;
4. La génératrice tachymétrique placée en bout de l'arbre moteur délivre une tension avec un gain $\beta = 0.1V/rad.s^{-1}$.

Mode opératoire

Il s'agit de construire expérimentalement le lieu critique pour le relais avec hystérésis $H = 1V$

- Dans un premier temps, on veut vérifier la caractéristique du relais désirée. On prendra sur X l'entrée du relais, et sur Y la sortie du relais. On réglera l'oscilloscope en mode XY en appuyant sur Aquire»Time mode»XY, et en se servant du mode persistant de l'oscilloscope **Display**$> \infty$**Persist**. Observer la courbe $S_r = f(\epsilon)$ après avoir réglé le seuil (dead band) à zéro, puis l'hystérésis H à la valeur correcte de 1 V sur le module relais simulé. La sortie S_r vaut $\pm 7.25V$.
- **Etude théorique :** On prend $k_1 = 0.5$. Tracer sous Matlab dans le plan de Nyquist le lieu de transfert de la partie linéaire $k_1.L(j\omega)$. L'intersection du lieu critique $-1/N(X_l)$ avec le lieu linéaire $k_1.L(j\omega)$ donne lieu à une autooscillation dont l'amplitude est $X_l = X_0$, et la pulsation de pompage vaut ω. Noter ces valeurs théoriques.
- Pour remettre l'oscilloscope en mode normal, faire **Main Delayed**>**Roll**. Régler k_1 à 0.5. Relever la tension crête à crête ($2X_0$), ainsi que la fréquence pour ces auto-oscillations. Comparer aux valeurs théoriques d'amplitude et de pulsation de pompage.
- En faisant varier l'atténuateur calibré, on fait varier le gain k_1 de $k_1 L(j\omega)$) ; on détermine un nouveau point d'intersection avec le lieu critique et ainsi de suite. Confronter les valeurs théoriques et expérimentales d'amplitude et de pulsation de pompage pour les valeurs de k_1 suivantes : 0.7, 0.6, 0.5, 0.4, 0.3. Commentaires ?
- A partir des données réelles obtenues et du procédé connu, tracer le lieu critique du relais.
- Que peut-on en conclure ?

Amélioration des performances de l'asservissement

Linéarisation par balayage Caractéristique du relais $H = 1V$; atténuateur $k_1 = 0.5$.

- On prend à présent le GBF que l'on règle préalablement pour obtenir une sortie sinusoïdale de fréquence 50 HZ et d'amplitude 0.2V.
- Le système étant en auto-oscillation, ajouter au signal d'erreur, sur une autre entrée du relais, la tension délivrée par le GBF. Augmenter progressivement l'amplitude. Qu'observez-vous ?
- Retrouver ce résultat avec Matlab.
- Conclusion

Correction par boucle secondaire tachymétrique La tension délivrée par la dynamo tachymétrique préalablement atténuée par un potentiomètre remplace ici la tension sinusoïdale précédente : on fait ce qu'on appelle un retour tachymétrique.

- Pour $H = 1V$; régler $k_1 = 0.5$. Quelle est l'amplitude de l'auto-oscillation ?
- Déterminer expérimentalement la valeur du taux de contre-réaction tachymétrique permettant de diviser par 2 l'amplitude de l'auto-oscillation.
- Retrouver ce résultat sous Matlab en utilisant le lieu critique tracé précédemment.

3.4 Références bibliographiques

- ★ http ://moodle.insa-toulouse.fr/course/view.php ?id=66 - Cours sur la méthode du 1er harmonique sous Moodle
- ★ C. MIRA *Cours de systèmes asservis non linéaires*. DUNOD UNIVERSITE), 1969.
- ★ J.C. GILLES, M. PELEGRIN *Systèmes asservis non linéaires tomes 1,2 et 3*. DUNOD AUTOMATIQUE, 1975

MANIPULATION 4

Commande numérique : réponse pile & Commande PID numérique

Commande d'un moteur électrique par ordinateur

4.1 But de la manipulation et présentation du procédé

On dispose d'un moteur électrique, que l'on souhaite commander. L'entrée du système est une tension $u(t)$ comprise entre +10 et -10V (on pourra rajouter un saturateur pour ne pas dépasser les valeurs limites). La sortie est la vitesse de rotation $\omega(t)$ [1].

Le moteur est modélisé par un premier ordre :

$$G(p) = \frac{\Omega(s)}{U(s)} = \frac{K_m}{1+\tau s}.$$

Le but de cette manipulation est :

— d'étudier l'influence de la période d'échantillonnage ainsi que du gain de boucle sur les performances d'une régulation numérique ;

— de mettre en oeuvre une commande pile ;

— de mettre en place une commande de type PID numérique.

On utilise Matlab pour la commande. La mise en oeuvre numérique de la commande en temps réel est faite dans Matlab grâce à la boîte à outils "Simulink Desktop RealTime" (cf aide sous moodle).

4.2 Etude théorique (préparation)

Régulation numérique avec correcteur proportionnel

On met en place une régulation échantillonnée suivant le schéma de la figure 14.

1. Le procédé en réalité deux sorties $\omega(t)$ et $\omega_2(t)$ qui représentent la vitesse de rotation du moteur filtrée et non filtrée. Nous utiliserons la mesure filtrée par un filtre de Butterworth.

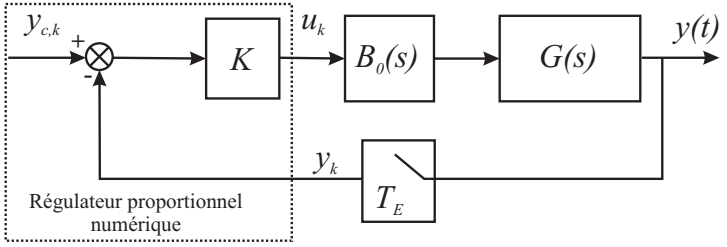


FIGURE 14 – Principe de bouclage discret

Pour réaliser la discrétisation du système, on choisira une discrétisation de type *avant* :

$$p = \frac{z-1}{T_e}.$$

- Calculer la fonction de transfert échantillonnée du moteur
- Calculer la fonction de transfert en boucle fermée, pour un régulateur proportionnel K, en fonction de K, K_m, T_E et τ.
- Calculer l'erreur statique de la régulation échantillonnée en fonction de K, K_m, T_E et τ.
- En recherchant les conditions de stabilité limite, calculer la surface de stabilité critique :

$$K = f(T_E, \tau, K_m), \tag{4}$$

où K représente le gain du régulateur proportionnel et T_E, τ et K_m la période d'échantillonnage, constante de temps et gain du système respectivement.

Régulation numérique par commande à réponse pile

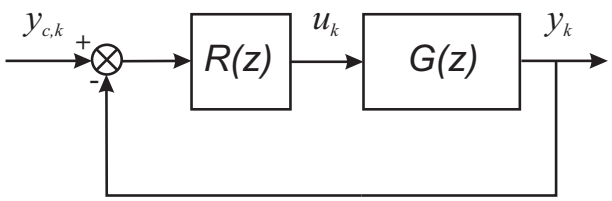


FIGURE 15 – Correcteur discret

On désire réaliser une commande bouclée d'un procédé de fonction de transfert échantillonnée $G(z)$ avec un correcteur discret $R(z)$ selon le schéma de la figure 15.

Dans le cas général, donner l'expression d'un correcteur $R(z)$ permettant d'obtenir pour le système bouclé un modèle par fonction de transfert :

$$\frac{Y(z)}{Y_c(z)} = \frac{1}{z^{(d+1)}} \tag{5}$$

où le paramètre d représente le retard pur du procédé.

Contrôle par Correcteur PID numérique (bonus en cas de temps)

Lors de la commande avec le correcteur proportionnel, vous avez réalisé un système corrigé ayant une erreur statique. On souhaite maintenant utiliser un correcteur de type PID numérique permettant d'éliminer cette erreur statique

- Quel type de correcteur PID numérique (P, PI, PD ou PID) permettrait de vérifier ce cahier des charges ?
- Donnez la représentation par schéma bloc de cette commande,
- Proposez une méthode de réglage des différents paramètres.

4.3 Etude expérimentale

Identification du procédé

On dispose d'un moteur électrique, que l'on souhaite commander via la boîte à outils "Simulink Desktop RealTime" de Matlab. Se référer à l'aide "Procédure de connexion Simulink Desktop Realtime" sous moodle. Le signal issu des capteurs correspond à une tension comprise entre -10.0 et +10.0 Volts. De même les valeurs de sorties qui attaquent les CNA doivent être comprises dans cette même plage de valeurs. On pourra donc rajouter un saturateur pour ne pas dépasser les valeurs permises.

Le moteur est modélisé par un premier ordre

$$G(p) = \frac{K_m}{1+\tau p}.$$

La première étape consiste à identifier les paramètres du moteur en étudiant sa réponse indicielle. Pour cela, sous Matlab,

- Créer un .xls qui va gérer la commande. Mettre en place une entrée **échelon** à 4 connectée au bloc CNA.
- Une fois les paramètres pertinents choisis pour l'identification (on peut par exemple prendre 0.01 comme pas de calcul), lancer la commande. Visualiser la vitesse du moteur à l'oscilloscope ou via Matlab. Appuyer sur le **stop** de l'oscilloscope une fois que le moteur a atteint son régime permanent.
- Grâce à cette réponse, identifier les paramètres du moteur.

Auto-oscillations

- Pour les valeurs de K_m et τ identifiées sur le système réel, tracer la courbe

$$K = f(T_E, \tau, K_m), \tag{6}$$

- Pour la période d'échantillonnage $T = 0,35$s, le gain K_m et temps de réponse τ identifié, donner la valeur du gain K_{lim}

correspondant à la limite de stabilité. Vérifier en simulation avec votre application `Simulink` puis vérifier expérimentalement ce résultat.

- ▶ Pour la période d'échantillonnage $T = 0,35$s, le gain K_m et temps de réponse τ identifiés, calculer l'erreur statique avec $K = 1$, sous `Simulink` et le vérifier en pratique.

Mise en œuvre d'une commande à réponse pile

Appliquez vos résultats théoriques au cas de la régulation de vitesse du moteur, pour $d = 0$.

- ▶ Vérifier par simulation le résultat attendu puis comparer avec des mesures expérimentales.
- ▶ Conclure

Mise en œuvre d'une commande PID numérique (bonus)

Appliquez vos résultats théoriques au cas de la régulation de vitesse du moteur.

- ▶ Donner les spécifications choisies pour régler un correcteur permettant d'éliminer l'erreur statique, tout en gardant des performances acceptables.
- ▶ Vérifier par simulation le résultat attendu puis comparer avec des mesures expérimentales.
- ▶ Conclure

MANIPULATION 5

Commande optimale : le pendule inversé

Le robot legot EV3

5.1 But de la manipulation

L'objectif de cette manipulation est de réguler le robot Lego EV3 de manière à le maintenir vertical sur ses deux roues (manipulation de type "pendule inversé"). Le système est naturellement instable. Pour cela, on va :

- passer d'une conception logicielle à une implémentation matérielle, en passant par une étape de vérification par simulation. La chaîne de conception se fera via Matlab et ses boîtes à outils ;
- mettre en œuvre une commande par retour d'état par un placement de pôles ;
- mettre en œuvre une commande optimale de type lqr, et comparer différentes valeurs de Q et R.

5.2 Présentation de la maquette

On travaille avec un robot Lego EV3. C'est un robot à deux roues mobiles. Les entrées et les sorties du système sont représentées sur la Figure 16.

Au niveau de chaque roue, un capteur de position angulaire donne l'angle de la roue en degré. Un capteur ultrasons permet de mesurer la distance à un obstacle avec une précision importante. Il permet ainsi au robot de prendre une décision d'évitement. Les distances mesurées sont en centimètres. Un capteur gyroscopique permet de mesurer la rotation du robot grâce à un gyroscope à axe unique placé sur un résonateur à quartz. Concrètement, ce capteur gyroscopique mesure le nombre de degrés par seconde ainsi que la direction de la rotation.

Le fonctionnement du robot est géré par un processeur ARM9 sous un OS Linux.

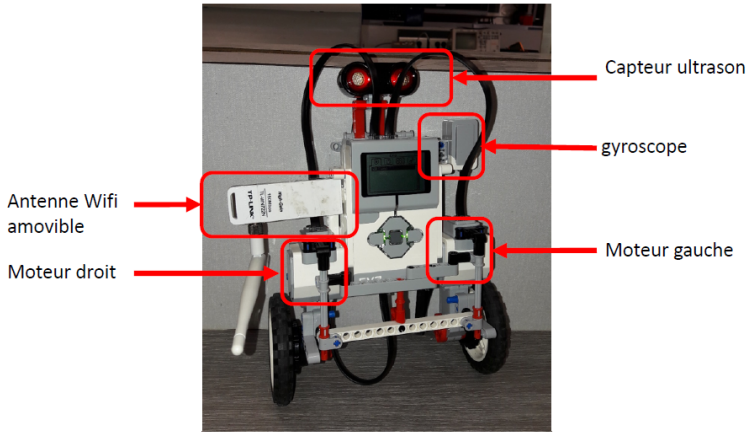


FIGURE 16 – le robot Lego EV3

Pour une présentation plus détaillée du robot et de son modèle, se référer à l'annexe.

5.3 Présentation du dossier RobotPendule

Copier sur votre répertoire, le dossier RobotPendule EV3 à partir de Moodle ainsi que le fichier "Guide pendule Inverse". Ce fichier décrit le contenu de l'archive et vous permet de débuter le TP.

5.4 Manipulation

Commande LQ

Simulation Dans une première approche, on souhaite contrôler le robot par retour d'état. Afin de déterminer le gain de ce retour d'état, on utilise les principes de commande LQ (pour plus d'informations concernant cette commande regarder l'annexe). Pour cela,

vous devez effectuer les tâches suivantes pour plusieurs valeurs des matrices de pondération Q et R :

- Quels sont les états du système, décrire le vecteur d'état. Dans la mesure où le modèle de simulation réparti le même couple sur les deux moteurs des roues, nous ne considèrerons qu'une entrée. Que cela implique-t-il sur la représentation d'état ? Donner la dimension de Q et R.
- Compléter le fichier matlab lego_selbalance_controller.m. Il vous suffit de définir les matrices de pondération QQ et RR. Ce fichier doit calculer le gain de retour d'état appelé KK ici.
- Modifier le simulink "Balance ans Drive Control". Sur ce fichier, vous disposez de plusieurs données :
  — $x1\_ref$ consigne de modèle d'état
  — $theta\_ref$ consigne pour l'angle d'inclinaison des roues
  — $x1$ mesure de modèle d'état
  — $theta$ mesure pour l'angle d'inclinaison des roues
  — vol commande des moteurs des roues.
  Vous devez implémenter le retour d'état en utilisant le gain de retour d'état KK et les différentes données présentes (toutes ne seront pas forcément utilisées à cet endroit là).
- Simuler le comportement du robot. **On effectuera un tableau comparatif des résultats pour différentes valeurs de QQ et RR en fonction des réponses obtenues**. On essaiera le plus possible de classer les réponses en différentes catégories dont on donnera les caractéristiques. Montrer quelles composantes de QQ et RR ont de l'importance dans la régulation, sur quelles caractéristiques ?
- Quelles valeurs de QQ et RR conservez-vous ? Justifier votre réponse.

Test sur le robot réel Une fois que vous aurez déterminé par simulation un réglage optimal pour QQ et RR, tester votre correcteur sur le robot en utilisant le mode "external" de simulink. Regarder les effets des perturbations sur votre réglage, comparer les résultats expérimentaux et la simulation. Conclure.

Action intégrale

Lorsqu'on effectue une commande par retour d'état, l'erreur statique n'est pas forcément nulle. Ceci n'est généralement pas gênant dans la mesure où l'objectif poursuivi est une régulation. Néanmoins, dans le cas où l'objectif consiste à suivre une trajectoire, on peut alors procéder de la même manière que pour les systèmes asservis classiques en insérant un intégrateur dans la chaîne directe comme illustré sur la figure 19.

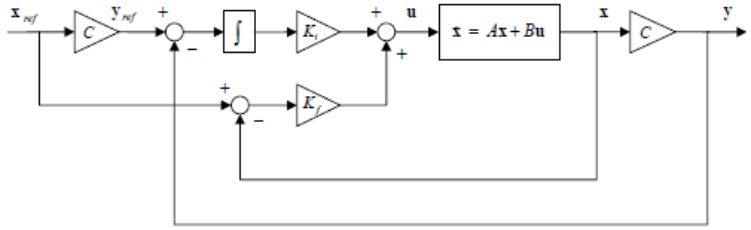


FIGURE 17 – Schéma bloc d'une commande par retour d'état avec action intégrale

On propose dans cette partie d'introduire cet intégrateur dans la fonction de commande du robot.

Une explication succinte de la théorie est donnée en annexe B.

- Compléter le schéma bloc de la commande pour mettre en place l'action intégrale.
- Définir le système augmenté et calculer une valeur de K_i permettant de réduire l'erreur statique par la méthode lqr.
- Tester le comportement du robot en simulation. Comparer avec les résultats précédents. **On effectuera un tableau comparatif des résultats.** Trouver une valeur de K_i qui vous semble optimale.
- Implémenter cette commande en réel sur le robot. Tester. Regarder les effets des perturbations sur votre réglage.

Commande par placement de pôles

La commande LQ a l'avantage et l'inconvénient de ne pas imposer le choix des valeurs propres de la matrice de commande corrigée. Ceci est un avantage car le concepteur n'a pas à réfléchir sur le choix des valeurs propres, mais c'est également un inconvénient car on ne choisit donc pas la dynamique du système.

On propose dans cette partie d'affiner les paramètres trouvés dans la partie LQ de façon à essayer d'améliorer la dynamique.

- ▶ Trouver les valeurs propres imposées par la commande LQ de la première partie qui vous semblent optimales.
- ▶ A l'aide de la fonction *place* de Matlab, trouver la valeur de K_f pour imposer ces valeurs propres au système. Un rappel du placement de pôles est donné en Annexe B.
- ▶ Tester le comportement du robot en simulation. Comparer avec les résultats précédents.
- ▶ Faites varier les valeurs propres imposées de manière à améliorer le comportement du robot. Tester le comportement du robot en simulation. **On effectuera un tableau comparatif des résultats, de manière à conclure sur des coefficients optimaux.**
- ▶ Implémenter la meilleure commande en réel sur le robot. Tester. Regarder les effets des perturbations sur votre réglage.

Placement de pôles avec action intégrale

On propose dans cette partie d'introduire un intégrateur dans la fonction de commande du robot en conservant la commande par placement de pôles.

- ▶ Reprendre le schéma bloc de la commande pour mettre en place l'action intégrale.
- ▶ Définir le système augmenté et calculer une valeur de K_i permettant de réduire l'erreur statique avec la fonction *place* de Matlab.
- ▶ Tester le comportement du robot en simulation. Comparer avec les résultats précédents. **On effectuera un tableau comparatif des résultats.** Trouver une valeur de K_i qui vous semble optimale.
- ▶ Implémenter cette commande en réel sur le robot. Tester. Regarder les effets des perturbations sur votre réglage.

5.5 Conclusion

- ▶ Conclure sur une synthèse les avantages, les inconvénients, et les performances de votre système selon le correcteur utilisé, grâce aux différents tableaux mis en place.
- ▶ Tester éventuellement le robot avec des roues différentes. Conclure.

5.6 Références bibliographiques

- ★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.
- ★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)

Annexe

MANIPULATION A

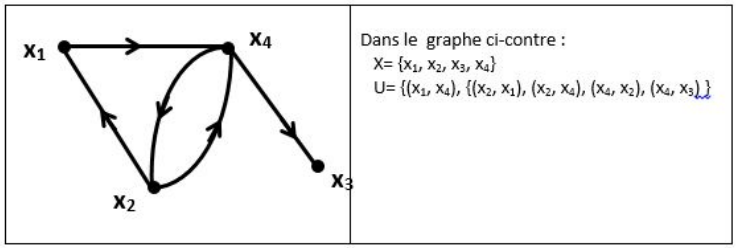
Annexe : Théorie des graphes - Algorithme de Dijkstra

La théorie des graphes est un outil privilégié de modélisation et de résolution de problèmes de décision dans un très grand nombre de domaines allant des sciences fondamentales (physique, informatique) aux applications technologiques les plus concrètes (conception de réseaux de télécommunications, problèmes d'affectation, GPS et recherches de plus courts chemins, etc.). Les concepts de base de la théorie des graphes sont présentés en détail dans le cours Graphes du semestre 2 de la 4AE. Ne sont donc introduits ici que les éléments indispensables pour la mise en oeuvre de l'algorithme de Dijkstra permettant de rechercher le plus court chemin entre un sommet et tous les autres dans un graphe.

A.1 Concepts de base sur les graphes

Graphe, sommets, arcs

Un graphe $G(X, U)$ est constitué
- d'un ensemble de sommets $X = \{x_1, x_2, \ldots, x_n\}$
- d'un ensemble d'arcs reliant ces sommets $U = \{u_1, u_2, \ldots, u_m\}$. Un arc u est défini par un couple de sommets : $u = (x_i, x_j) x_i \in X; x_j \in X$. x_i est l'origine de l'arc u et x_j son extrémité.



Longueur d'un arc et graphes valués

Il est possible d'associer une valeur l_{ij} à chaque arc (x_i, x_j). Cette valeur (qui peut être positive, négative ou nulle) peut par exemple représenter une distance, ou une vitesse, ou un coût ou toute autre grandeur représentative du problème modélisé par le graphe. On parle alors de graphe valué.

Chemins et circuits

Un chemin est une suite d'arcs telle que l'extrémité d'un arc est l'origine de l'arc suivant dans ce chemin. Ainsi par exemple, $(X_2 - X_1 - X_4)$ est un chemin entre les sommets x_1 et x_4.

Un circuit est un chemin dont l'extrémité finale est aussi l'origine du chemin. Ainsi par exemple, $(X_2 - X_1 - X_4 - X_2)$ est un circuit.

La longueur d'un chemin (ou d'un circuit) est égale à la somme des longueurs des arcs qui composent le chemin (ou le circuit).

A.2 Recherche de plus courts chemins

Une problématique classique dans un graphe valué est de rechercher le plus court chemin entre un sommet du graphe que l'on notera s (sommet source) et tous les autres. La théorie des graphes propose différents algorithmes de recherche de plus courts chemins adaptés aux propriétés du graphe. Dans le cas particulier où toutes les longueurs d'un graphe valué sont positives, on peut appliquer l'algorithme de Dijkstra.

Cet algorithme est itératif. On associe 3 informations à chaque sommet xi :
- λ_i : longueur du plus court chemin trouvé à l'itération courante entre s et x_i
- q_i : dernier sommet avant x_i sur ce chemin (permet de reconstituer le plus court chemin à la fin de l'algorithme)
- $m_i \in \{0, 1\}$: le sommet x_i est dit "marqué" lorsque $m_i = 1$. Il est dit "non marqué" lorsque $m_i = 0$. Lorsque le sommet

est marqué, λ_i correspond à la longueur du plus court chemin entre s et x_i (λ_i a sa valeur définitive).

Principe général de l'algorithme

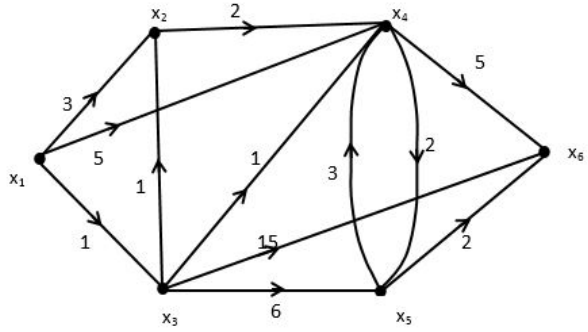
- Initialisations : λ_s est initialisé à 0 (le plus court chemin entre s et s est de longueur nulle) ; les $\lambda_i, \forall i \neq s$ sont initialisé à ∞ (les plus courts chemins entre s et x_i ne sont pas connus en début d'algorithme : leur longueur est donc ∞) ; ces longueurs ne sont pas définitives $\Rightarrow m_i = 0 \forall i$
- A chaque itération :
  - On repère le sommet non encore marqué ($m_i = 0$) qui a le plus petit λ.
  - On marque ce sommet ($m_i = 1$) car, compte tenu de la propriété du graphe, on peut être sûr qu'il n'existe pas de plus court chemin entre s et ce sommet.
  - On essaie d'actualiser les λ des sommets suivants immédiats du sommet qui vient d'être marqué (cf. algorithme détaillé).
  - On recommence une nouvelle itération jusqu'à ce que tous les sommets soient marqués ($m_i = 1$).

Algorithme détaillé

```
-- initialisations
λs ← 0 ; qs ← s ; ms ← 0
pour tout i de 1 à n avec i ≠ s répéter
    λi ← ∞ ; qi ← ϕ ; mi ← 0
fp
-- itérations
tant qu'il existe un sommet non marqué (mi = 0) répéter
    Choisir parmi les sommets non marqués celui dont le λ est minimum ; soit xi ce sommet
    Marquer ce sommet : mi ← 1
    -- actualiser éventuellement les λ des sommets suivants xi
    Pour tout sommet xj ∈ S(i) et tel que mj = 0
        si λi + lij < λj alors
            λj ← λi + lij
            qj ← i
        fsi
    fin pour
fin tant que
Remarque : on peut alors reconstituer le plus court chemin entre s et xi en utilisant qi
```

Mise en oeuvre sur un exemple

Considérons le graphe valué dessiné ci-dessous. On recherche les plus courts chemins entre s=x1 et les autres sommets du graphe.



Le tableau ci-dessous détaille les différentes itérations de l'algorithme. Les cases grisées montrent le sommet qui est marqué au début de chaque itération (sommet non marqué ayant le plus petit λ. Les valeurs actualisées au cours de l'itération des λ et m des sommets suivants celui qui est marqué apparaissent sur la même ligne.

	x_1			x_2			x_3			x_4			x_5			x_6		
	λ	q	m	λ	q	m	λ	q	m	λ	q	m	λ	q	m	λ	q	m
initialisations	0	x_1	0	∞	-	0	∞	-	0	∞	-	0	∞	-	0	∞	-	0
Itération 1	0	x_1	1	**3**	**x_1**	**0**	**1**	**x_1**	**0**	**5**	**x_1**	**0**	∞	-	0	∞	-	0
Itération 2				**2**	**x_3**	**0**	1	x_1	1	**2**	**x_3**	**0**	**7**	**x_3**	**0**	**16**	**x_3**	**0**
Itération 3				2	x_3	1				2	x_3	0	7	x_3	0	16	x_3	0
Itération 4										2	x_3	1	**4**	**x_4**	**0**	**7**	**x_4**	**0**
Itération 5													4	x_5	1	**6**	**x_5**	**0**
Itération 6																6	x_5	1

Reconstitution des plus courts chemins :

Par exemple le plus court chemin entre x_1 et x_6 est de longueur 6 ($\lambda_6 = 6$). Le dernier sommet avant x_6 sur ce chemin est x_5 (car $q_6 = x_5$). Le dernier sommet avant x_5 sur ce chemin est x_4 (car $q_5 = x_4$). Le dernier sommet avant x_4 sur ce chemin est x_3 (car $q_4 = x_3$). Le dernier sommet avant x_3 sur ce chemin est x_1 (car $q_3 = x_1$). Le plus court chemin est donc : $x_1 - x_3 - x_4 - x_5 - x_6$.

A.3 Références bibliographiques

★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.

★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)

MANIPULATION B

Annexe : Commande par retour d'état

B.1 Introduction et principes de fonctionnement

La commande par retour d'état est une technique de contrôle qui consiste à multiplier l'erreur entre la valeur de référence $x_{ref}(t)$ et la valeur de l'état mesurée $x(t)$ par un gain de retour K et d'utiliser cette valeur en commande. De fait, la commande par retour d'état peut être vue comme une généralisation de la commande Proportionnel Dérivé (PD) dans la théorie de la commande classique. La figure 18 montre le schéma bloc d'une commande par retour d'état.

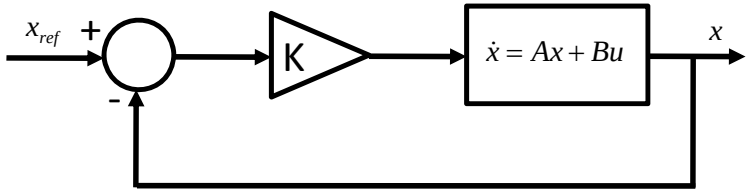


FIGURE 18 – Schéma bloc de la commande par retour d'état

La commande et l'équation d'état de ce système sont les suivants :

$$u(t) = -K(x(t) - x_{ref}(t)), \quad (7)$$
$$\dot{x}(t) = (A - BK)x(t) + BKx_{ref}(t). \quad (8)$$

Il est alors possible de stabiliser le système en choisissant le gain de retour K de manière à placer correctement les valeurs propres de $A - BK$.

Pour utiliser une commande par retour d'état, il est nécessaire que le système soit commandable. Une condition nécessaire et suffisante pour la commandabilité du système est que la matrice de commandabilité M_c soit de rang plein.

$$\mathbf{rang}(M_c) = n, \quad (9)$$
$$M_c = [B, AB, \dots A^{n-1}B], \quad (10)$$

avec n la dimension du système. La "control" toolbox de Matlab permet d'évaluer la matrice de commandabilité.

`Exemple:` Le système suivant est-il commandable ?
$A = [0, 1; -2, -3], B = [0; 1] \rightarrow$ commandable.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Mc = ctrb(A,B);
> rank(Mc)
ans = 2
```

B.2 Obtention de la matrice K

De nombreuses méthodes existent pour trouver le gain K. Toutefois, le *placement de pôles (et/ou de structure propre)* et la *commande linéaire quadratique (dite optimale)* sont deux méthodes très laregment utilisées par les ingénieurs pour obtenir le gain de retour d'état K.

Placement de pôles direct

Cette méthode consiste à calculer le gain de retour K de manière à placer les pôles (les valeurs propres) de la matrice $A-BK$ à une valeur précise. Les valeurs choisies sont évidemment stables, et sont souvent tirées des performances souhaitées sur le système corrigé (vitesse, amortissement, découplage...). Il faut toutefois veiller à ce que la valeur de K ne soit pas aberrante (numériquement bien conditionnée, limitant les saturations...). Cette méthode requiert

de résoudre un système linéaire.

La fonction `place` de Matlab permet d'effectuer un placement de pôles direct qui minimise la sensibilité.

`Exemple:` Calculer le gain de retour d'état pour le système $A = [0, 1; -2, -3], B = [0; 1]$ de manière à placer les pôles en -5 et -6.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> poles = [-5, -6];
> K = place(A,B, poles)
K =
28.0000 8.0000
```

La commande optimale Linéaire Quadratique (LQ)

La commande par *placement de pôles* a pour objectif de synthétiser K, un correcteur statique par retour d'état (*eg.* tel que $u(t) = Kx(t)$) de telle façon que les pôles de la boucle fermée sont placés aux endroits souhaités dans le demi-plan complexe gauche.

L'idée générale : Le principe de la *commande Linéaire Quadratique (LQ)*, souvent - et abusivement - appelée "commande optimale", est de synthétiser K, un correcteur statique par retour d'état (*e.g.* tel que $u(t) = K(x(t) - x_{ref}(t))$, qui minimise un critère quadratique.

Dans le cas des systèmes échantillonnés, ce critère quadratique n'est autre qu'une somme (finie ou infinie) pondérée de $x(t)$, l'état du système, et $u(t)$, la commande qui lui est appliquée. Dans le cas des systèmes continus, cette somme est une intégrale.

Alors que le placement de pôles peut être vu comme complexe pour un utilisateur (nécessitant de choisir les pôles), la commande linéaire quadratique, quant à elle, est basée sur un critère énergétique, bien plus parlant pour l'ingénieur. A titre introductif, il est à noter que la commande LQ possède, intrinsèquement, de très bonnes propriétés de **robustesse**.

Formulation mathématique (cas continu, a horizon infini) : Soit le système Linéaire à Temps Invariant (LTI) à n_u entrées et n_y sorties défini par sa fonction de transfert $H(s) \in \mathbb{C}^{n_y \times n_u}$, complexe et analytique sur $\mathbb{C}^+$, munie d'une réalisation,

$$\dot{x}(t) = Ax(t) + Bu(t),\ y(t) = Cx(t) + Du(t). \tag{11}$$

L'objectif de la méthode LQ consiste à calculer le gain de retour d'état K de manière à minimiser la fonction de coût J suivante :

$$J = \int_0^\infty \Big((x(t) - x_{ref}(t))^\top Q (x(t) - x_{ref}(t)) + u(t)^\top R u(t) \Big)\, dt \tag{12}$$

sous la contrainte de la dynamique du système (11). Il est à noter que le critère J représente le compromis entre l'énergie du système, $x(t)^T Q x(t)$, et celle de la commande, $u(t)^T R u(t)$.

Il peut être montré (voir cours au second semestre) que la solution de ce problème est donnée par la commande

$$u(t) = -K(x(t) - x_{ref}(t)), \tag{13}$$

avec

$$K = R^{-1} B^T P, \tag{14}$$

où $P > 0$ (*i.e.* définie positive), est la solution de l'Equation Algébrique de Riccati (EAR),

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{15}$$

Il est à noter que le coût partiel est $J(x(t), t) = \frac{1}{2} x(t) T P x(t)$.

Les paramètres de choix sont alors les poids des matrices symétriques pour l'état $Q = Q^T$, la commande $R = R^T$. Les matrices sont le plus souvent choisies comme diagonales. Une solution généralement adoptée pour avoir déjà une indication des ordres de grandeur de ces coefficients est de fixer $R = I_{n_u}$ et $Q = \rho I_n$, puis de

faire varier le scalaire ρ (un $\rho \gg 1$ indique que on va se concentrer sur la minimisation de l'énergie du système, alors $\rho \ll 1$ indique que on va se concentrer sur la minimisation de la commande). On procède ensuite par essai erreur. La fonction `lqr` de Matlab permet d'obtenir le gain de retour avec la méthode LQ.

`Exemple:` Calculer le gain de retour d'état pour le système $A = [0, 1; -2, -3], B = [0; 1]$ en utilisant $Q = [100, 0; 0, 1]$ et $R = 1$.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Q = [100, 0; 0,1]; R = 1;
> K = lqr(A,B,Q, R)
K =
8.1980 2.1377
```

ou

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Q = [100, 0; 0,1]; R = 1;
> [P,L,G] = care(A,B,Q,R); % Résolution de l'EAR
> K=R\1*B'*P;
K =
8.1980 2.1377
```

A propos de la robustesse : Si nous considérons un système mono-entrée mono-sortie, bouclé par une commande $u(t) = Kx(t)$, où K est synthétisé par approche LQ à horizon infinie, avec $R = \rho I$. Dès lors, en notant le transfert de boucle $L(s)$

$$L(s) = K(sI_n - A)^{-1}B, \tag{16}$$

et le transfert $F(s)$ (utilisé pour l'analyse de sensibilité et de robustesse) et la fonction de sensibilité $S(s)$ par

$$F(s) = I + L(s) = S(s)^{-1}, \tag{17}$$

il peut être prouvé que

$$\begin{array}{rrcl} & F(-s)^T F(s) & \geq & 1 \\ \Leftrightarrow & |F(j\omega)| & \geq & 1 \\ \Leftrightarrow & |1 + L(j\omega)| & \geq & 1 \\ \Leftrightarrow & |S(j\omega)| & \leq & 1. \end{array} \tag{18}$$

Par conséquence, si la fonction de sensibilité $|S(j\omega)| \leq 1$, alors
- la marge de module du système ≥ 1,
- la marge de gain est ∞,
- la marge de phase ≥ 60deg.

La commande par retour d'état avec action intégrale

Lorsqu'on effectue une commande par retour d'état, l'erreur statique n'est pas forcément nulle. Ceci n'est généralement pas gênant dans la mesure où l'objectif poursuivi est une régulation. Néanmoins, dans le cas où l'objectif consiste à suivre une trajectoire, on peut alors procéder de la même manière que pour les systèmes asservis classiques en insérant un intégrateur dans la chaîne directe, en l'appliquant sur certaines sorties $y(t)$. On considère donc une commande de la forme [2] (attention, ici nous avons substitué $x(t) - x_{ref}(t)$ par $x_{ref}(t) - x(t)$) :

$$u(t) = -K_f(x_{ref}(t) - x(t)) + K_i \int_0^t (y_{ref}(t) - y(t))dt \tag{19}$$

appliqué au système dynamique :

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{20}$$
$$y(t) = Cx(t) \tag{21}$$

Si l'on introduit une nouvelle variable $z(t)$ telle que $\dot{z} = y_{ref}(t) - y(t)$, c'est-à-dire $\dot{z} = Cx_{ref} - Cx(t)$. La commande a une expres-

2. Les ouvrages sur la théorie de la commande décrivent habituellement cette commande avec $x_{ref} = 0$. On gardera ici l'expression de x_{ref} pour améliorer les performances de suivi.

sion de la forme

$$u(t) = -\begin{bmatrix} K_f & K_i \end{bmatrix}\begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} K_f & K_i \end{bmatrix}\begin{bmatrix} x_{ref} \\ 0 \end{bmatrix} \quad (22)$$

et elle peut être considérée comme étant la commande par retour d'état du système augmenté avec $X = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}\begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ C \end{bmatrix} x_{ref} \quad (23)$$

Quand on considère le système stabilisé, on a alors $y_\infty = y_{ref}$. Le schéma bloc d'une telle commande est illustré Figure 19.

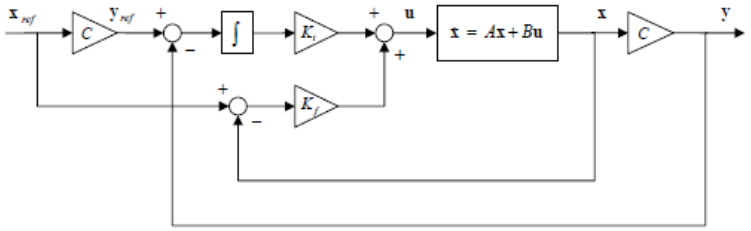


FIGURE 19 – Schéma bloc d'une commande par retour d'état avec action intégrale

B.3 Références bibliographiques

- ★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.
- ★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)