Département Génie Electrique et informatique

Travaux pratiques d'automatique

4ème année AE

Année 2011-2012

Semestre 2

Table des matières

ASNL : Méthode du plan de phase
Etude d'un asservissement de position à relais

1 Étude d'un asservissement de position à relais

1.1 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 1), par la méthode du plan de phase.

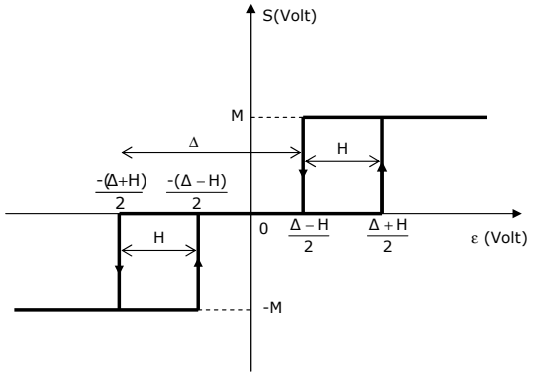


FIG. 1 – Élément non linéaire

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).

1.2 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.
- 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 2 montre un exemple de cycle limite stable vers lequel convergent les trajectoires, divergeant à partir d'un

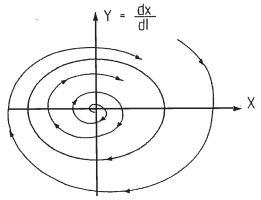


FIG. 2 – Exemple de trajectoire de phase

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.

1.3 Manipulation

Le schéma bloc de l'asservissement à étudier est donné par la figure 3.
- Les paramètres physiques ont été identifiés

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

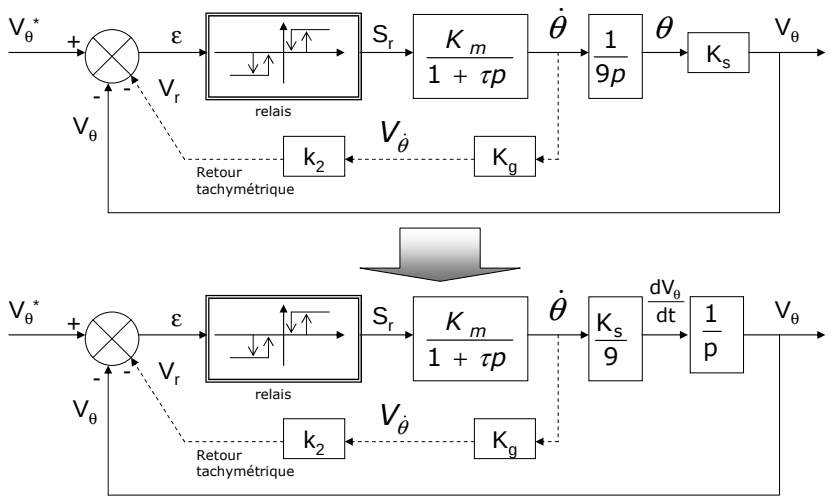


FIG. 3 – Schéma bloc de l'asservissement

- 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 1. 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 ? (amortissement, pulsation propre, erreur statique, dépassement, temps de montée, ...) 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 ? Répéter cette étude en faisant varier la valeur de Δ. Que remarque-t-on sur l'erreur statique ? Le dépassement ?

► Etude de l'hystérésis pur ($\Delta = 0$, $H \neq 0$). Faire une étude similaire pour l'hystéresis pur. Est-ce que le système va se stabiliser vers un point d'équilibre ? Justifier.

► 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.

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 équations des droites de commutation.

► 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).

1.4 Réalisation sur site réel

► Pratiquez les résultats obtenus en 1.3 sur le site réel.

► Observez la commande en régime glissant et en régime optimum.

1.5 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

Commande par retour d'état - Observateur
Régulation de débit d'air LTR 701

2 Régulation de débit d'air LTR 701

L'objectif de la manipulation consiste à réaliser une régulation de début d'air sur un processus de type "sèche-cheveux".

Le modèle étant donné, les études seront menées théoriquement sous Simulink. La commande effective sera ensuite mise en œuvre par xPC Target (voir Annexe xPC Target). xPC Target est un environnement logiciel qui permet de mettre au point, de tester et de mettre en œuvre des applications temps réel sur du matériel de type PC. Cet outil offre l'avantage d'être associé à Matlab/Simulink, ce qui permet de passer de la phase essai/simulation à la phase mise en œuvre/expérimentation très rapidement et très facilement en s'affranchissant de l'étape de codage.

2.1 Description de la manipulation

Il s'agit d'un dispositif (maquette AMIRA de type LTR 701) permettant la régulation du débit d'air (ou flux) propulsé à l'intérieur d'un tube ainsi que la régulation de la température de l'air en un point du tube. Nous ne nous intéresserons dans cette manipulation qu'à la régulation du débit d'air P.

Le dispositif est représenté sur la figure 4.

FIG. 4 – Maquette AMIRA LTR 701

Un schéma de principe est donné sur la figure 5.

La description des principaux éléments de la façade du boîtier est la suivante :

- ACTUATOR 1, entrée ($0 \backsim 10V$) : Commande M de la vitesse du ventilateur produisant le flux d'air. La commande peut être interne au dispositif, réglée par un potentiomètre, ou bien externe, par application d'un signal sur la borne d'entrée.
- ACTUATOR 2, entrée ($0 \backsim 10V$) : Commande du chauffage du flux d'air. La commande peut être interne au dispositif, réglée par un potentiomètre, ou bien externe, par application

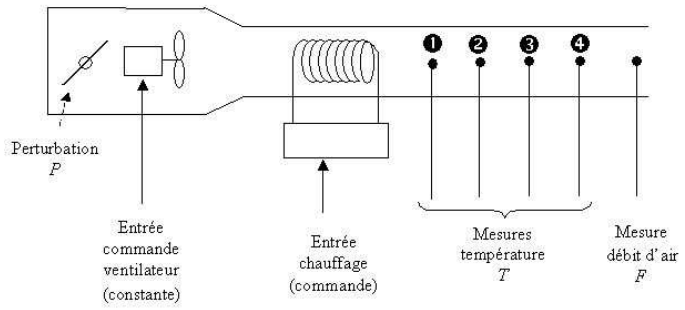


FIG. 5 – Schéma de principe de la maquette

d'un signal sur la borne d'entrée.
- SENSOR 1, sortie $(0 \backsim 10V)$: Mesure de l'angle du papillon (perturbation).
- SENSOR 2 et SENSOR 3, sortie $(0 \backsim 10V)$: Mesures de température.
- SENSOR 4, sortie $(0 \backsim 10V)$: Mesure de pression.

2.2 Modélisation et identification du processus

Modélisation

On souhaite contrôler ce système en débit d'air grâce à la commande M de la vitesse du ventilateur produisant le flux d'air. Le débit d'air est perturbé par le volet, que l'on réglera **80%**. Il est également pertubé, car il est chauffé par une résistance. La commande du chauffage sera réglée en interne, à **10%**.

Le modèle du système perturbé est non linéaire. On a donc identifié le modèle sur un point de fonctionnement (M_0, P_0), et on cherchera dans cette manipulation à mettre en place une commande par retour d'état autour de ce point de fonctionnement.

En approximant le système à un deuxième ordre, le modèle identifié en choisissant $M_0 = 80\%$ et en effectuant un échelon de 10% est le suivant :

$$F(s) = \frac{P(s)}{M(s)} = \frac{0.48}{1 + 0.6323s + 0.1001s^2} \quad (4)$$

Validité du modèle

Afin de vérifier la validité de la fonction de transfert (Equation 4), on se propose de comparer la réponse du système réel et du système identifié à un échelon.

- ▶ Mettre en place un modèle simulink permettant de tracer la réponse du système identifié à un échelon autour du point de fonctionnement utilisé lors de l'identification.
- ▶ Mettre en place un modèle simulink permettant de tracer la réponse du système réel à un échelon autour du point de fonctionnement utilisé lors de l'identification. On se servira de xPCTarget pour mettre en place ce test (voir Annexe xPC Target).
- ▶ Comparer les deux réponses à un échelon de 5%, 10%, 20% (dépassement, temps de réponse, retard, valeur finale, etc). Conclure sur la validité de la fonction de transfert proposée. Eventuellement, proposer une meilleure fonction de transfert pour modéliser le système.

Mise en place du modèle d'état

- ▶ De manière à pouvoir mettre en place une commande par retour d'état, définir les matrices A, B, C, D correspondant à la forme compagne de commandabilité.

2.3 Conception du retour d'état

L'objectif de la commande par retour d'état est de commander le système selon un cahier des charges donné. Le système identifié étant un système du second ordre, on peut choisir facilement la dynamique du système corrigé. Nous avons à notre disposition 3

cahiers des charges différents, deux basés sur des caractéristiques dynamiques souhaitées, un sur du placement de pôles :

Cahier des charges n°1 Le système corrigé doit être un système du second ordre avec les caractéristiques suivantes :
- Dépassement inférieur à 5%,
- Temps de réponse à 5% de 2s[1].

Cahier des charges n°2 Le système corrigé doit être un système du second ordre avec les caractéristiques suivantes :
- Dépassement inférieur à 5%,
- Temps de réponse à 5% de 0,5s.

Cahier des charges n°3 Le système corrigé doit être avoir les valeurs propres suivantes :
- valeur propre de -2 +2i.
- valeur propre de -2 -2i.

Préparation

- ▶ En préparation pour les deux cahiers des charges basés sur la réponse dynamique, déterminer le polynôme caractéristique souhaité pour le système bouclé. En déduire la loi de commande (gain de retour d'état L et gain de précommande l_c).
- ▶ Pour le troisième cahier des charges, justifier le choix des valeurs propres.

Travail expérimental

Simulation des cahier des charges 1 et 2

- ▶ Tester en simulation vos retours d'état pour une consigne de 0 à 0,5V.
- ▶ Comparer les, en regardant notamment la commande.
- ▶ Quelles sont les valeurs propres de ces deux réglages ?

Simulation du cahier des charges 3

- ▶ A l'aide de la fonction *place* de Matlab, déterminer la loi de commande (gain de retour d'état L et gain de précommande l_c).
- ▶ Tester sous simulink cette loi de commande.
- ▶ Comparer la avec les deux précédentes.

Test des commandes sur le système réel Une fois votre simulation validée, tester sur le système réel votre commande. (Attention, le système identifié était autour d'un point de fonctionnement ! ! !)

[1] On rappelle que le temps de réponse à 5% d'un système du second ordre peut être approximé à $tr_{5\%} = \frac{3}{\xi\omega_n}$ pour ξ proche de 0.7.

Régulation d'un niveau d'eau par PID numérique Application sous XPC Target

3 Régulation d'un niveau d'eau par PID numérique

3.1 But de la manipulation

L'objectif de cette manipulation est d'illustrer les différents aspects de la commande d'un processus par calculateur numérique :

- modélisation et identification des paramètres du processus ;
- synthèse théorique d'une commande ;
- mise en oeuvre de la commande en temps réel du processus.

Le processus à commander est ici un réservoir de liquide dont on veut réguler le niveau. L'outil de simulation est Matlab. La mise en oeuvre numérique de la commande en temps réel est faite dans l'environnement logiciel XPC Target.

3.2 Description du procédé et de l'environnement

Le procédé

Le processus illustré Fig. 6 commandé dans ce TP est un simple réservoir cylindrique de section S. L'entrée est le débit d'entrée Q_e et la sortie est le niveau X de fluide dans le réservoir. Un débit de fuite Q_s est assuré à travers une vanne manuelle.

Q_e est proportionnel à une tension de commande $Q_e = k_q U$. Le niveau X est mesuré par un capteur linéaire $X_m = k_c X$.

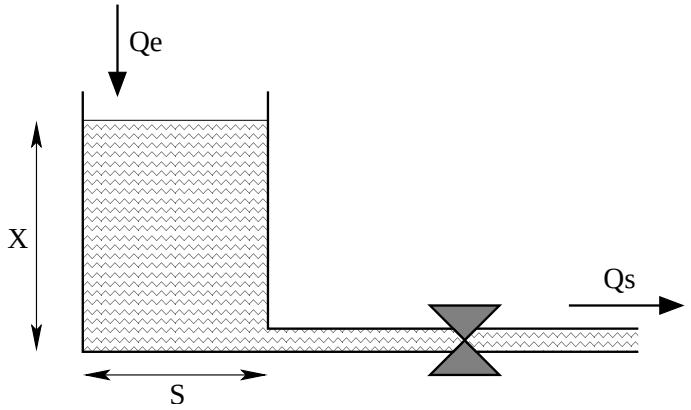


FIG. 6 – *Schéma du principe de fonctionnement du procédé*

L'environnement XPC Target

XPC Target est un environnement logiciel qui permet de mettre au point, de tester et de mettre en oeuvre des applications temps réel sur du matériel de type PC.

Cet outil offre l'avantage d'être associé à Matlab/Simulink, ce qui permet de passer de la phase essai/simulation à la phase mise en oeuvre/expérimentation très rapidement et très facilement en s'affranchissant de l'étape de codage.

La configuration matérielle standard est composée de 2 unités informatiques reliées par une liaison série asynchrone.

- **Le PC hôte** sert d'unité de développement. A partir de Matlab/Simulink, il est possible d'engendrer directement l'application temps réel (loi de commande dans notre cas). Il fonctionne dans l'environnement Windows habituel.
- **Le PC cible** est orienté *temps réel*. Il est donc démarré (à l'aide d'une disquette ou d'une EPROM) non pas sur DOS ou Windows mais sur un noyau temps réel spécifique. Ce noyau, entre autres choses, permet de recevoir et faire exécuter le code engendré par le PC hôte et d'assurer la communication entre les deux machines (téléchargement, mise en marche, adaptation en ligne de paramètres, rapatriement de données...). Enfin ce PC est équipé de cartes (dans notre cas une carte National Instrument PC6424 avec convertisseurs analogique/numérique et convertisseurs numérique/analogique) qui lui permettent de commander le processus à réguler.

Pour l'application considérée dans ce TP, on n'utilise XPC Target que par l'intermédiaire d'une interface écrite en Matlab. Ainsi la mise en boucle ouverte ou en boucle fermée, la mise en marche ou en arrêt et la modification des paramètres ne demandent aucune manipulation particulière (telle que la régénération du code pour la machine cible par exemple).

La cible est équipée d'un moniteur sur lequel sont visualisés deux types d'information : des informations liées à l'exécution temps réel de l'application en cours (nom de l'application, temps courant, période d'échantillonnage...) ainsi qu'un rapport sur l'exécution des commandes envoyées par le PC hôte (adaptation de paramètres, mise en marche,...). Le second groupe d'informations est graphique et dépend de l'application. Dans notre cas nous avons choisi d'afficher 2 oscilloscopes. Celui du haut donne le tracé de la commande envoyée au processus et celui du bas ceux de la mesure et de la consigne (en boucle ouverte la consigne est égale à la commande).

Description de l'interface

Au cours de ce TP, XPC Target sera utilisé par l'intermédiaire d'une interface écrite en Matlab.

Cette interface permet la mise en boucle ouverte (Fig. 7) ou en boucle fermée (Fig. 8) du processus. Le passage d'une configuration à l'autre se fait au moyen d'un menu déroulant.

Réglages en Boucle Ouverte En boucle ouverte, il est possible de régler :

- la période d'échantillonnage et la durée de l'expérience
- le point de fonctionnement U^*
- l'amplitude de l'échelon de position ΔU et son temps d'application (temps au bout duquel l'échelon sera appliqué)

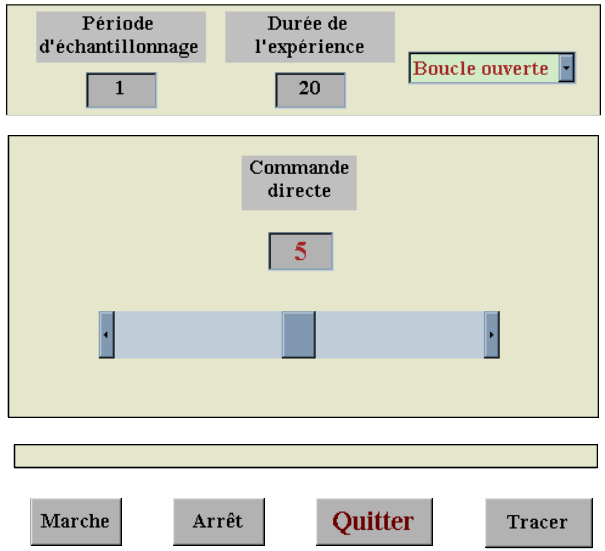


FIG. 7 – *Interface Matlab : réglages en Boucle Ouverte*

Dans l'application, la commande en boucle ouverte est la somme du point de fonctionnement U^* et de l'échelon de position ΔU.

Réglages en Boucle Fermée

En boucle fermée, il est possible de régler :
- la période d'échantillonnage et la durée de l'expérience
- le point de fonctionnement U^*, X^*
- l'amplitude de l'échelon de position X_c et son temps d'application
- les différents paramètres du correcteur PID

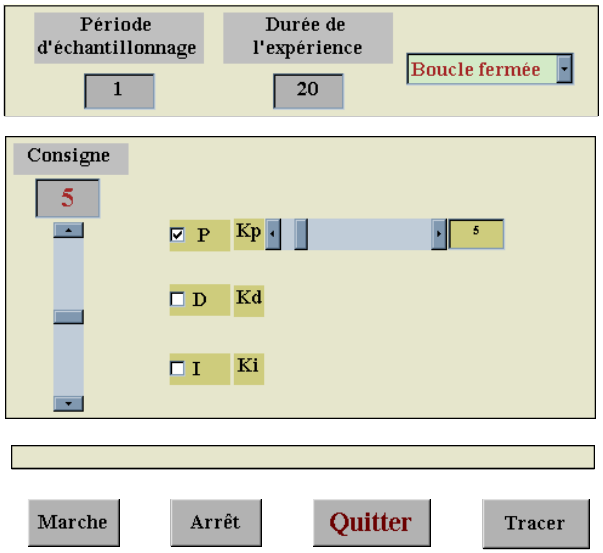


FIG. 8 – *Interface Matlab : réglages en Boucle Fermée*

Exploitation des résultats

Une fois l'expérience terminée, il est possible de :
- sauvegarder les données à l'aide de la fonction *Sove*. La sauvegarde s'effectue dans le fichier *experience.mat* contenant les vecteurs *Temps*, *Consigne*, *Commande* et *Mesure*. Attention : chaque sauvegarde écrase la précédente.
- tracer les réponses directement à l'aide du bouton *Tracer*

3.3 Modélisation et identification du processus

Modélisation

Le modèle de ce système est non linéaire si l'on considère que le débit de fuite Q_s est de type turbulent et dépend du niveau X selon la relation :

$$Q_s = \alpha\sqrt{X}$$

Le bilan volumique conduit à :

$$dX = \frac{1}{S}(Q_e - Q_s)dt \quad \Rightarrow \quad \frac{dX}{dt} = -\frac{\alpha}{S}\sqrt{X} + \frac{1}{S}Q_e$$

Si l'on suppose d'une part que le débit est linéaire en fonction d'une grandeur U de commande : $Q_e = k_q U$ et d'autre part que la mesure de niveau se fait par un capteur linéaire : $X_m = k_c X$, l'équation non linéaire du système s'écrit :

$$\frac{dX_m}{dt} = -\frac{\alpha\sqrt{k_c}}{S}\sqrt{X_m} + \frac{k_q k_c}{S}U$$

soit :

$$\frac{dX_m}{dt} = -\tau\sqrt{X_m} + kU = f(X_m, U) \quad avec : \quad \begin{cases} \tau = \frac{\alpha\sqrt{k_c}}{S} \\ k = \frac{k_q k_c}{S} \end{cases} \qquad (5)$$

On pourrait vérifier (comme dans les TPs de Systèmes Bouclés de 2ème année) la non linéarité en traçant la courbe reliant U^* à $\sqrt{X_m^*}$, et vérifier qu'elle correspond bien à une droite.

Recherche d'un modèle linéarisé

Le modèle linéarisé autour d'un point de fonctionnement (U^*, X_m^*) est obtenu en développant le modèle en série de Taylor autour de (U^*, X_m^*) tronquée au premier ordre. Pour cela on définit :

$$\begin{cases} \Delta X_m = X_m - X_m^* \\ \Delta U = U - U^* \end{cases}$$

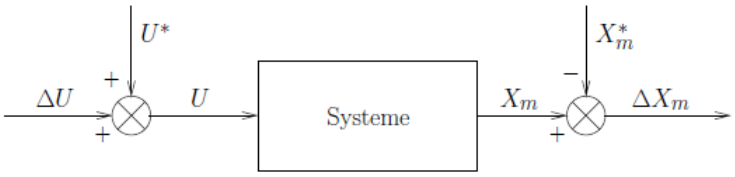


FIG. 9 – *Schéma bloc du système en Boucle Ouverte*

On a alors :

$$\frac{d\Delta X_m}{dt} \approx f(X_m^*, U^*) + \left.\frac{\partial f}{\partial X_m}\right|_{(X_m^*, U^*)} \Delta X_m + \left.\frac{\partial f}{\partial U}\right|_{(X_m^*, U^*)} \Delta U$$

$$\frac{d\Delta X_m}{dt} \approx \frac{-\tau}{2\sqrt{X_m^*}} \Delta X_m + k\Delta U$$

La fonction de transfert valable autour du point de fonctionnement (U^*, X_m^*) s'écrit alors :

$$G(s) = \frac{\Delta X_m(s)}{\Delta U(s)} = \frac{K}{1+Ts} \quad avec : \quad \begin{cases} T = \frac{2\sqrt{X_{m*}}}{\tau} \\ K = kT \end{cases} \qquad (6)$$

Afin de vérifier la validité de cette fonction de transfert et d'identifier les paramètres K et T, étudions la réponse à un échelon de position.

- Tracer la réponse à un échelon d'amplitude 0.5, puis d'amplitude 1, autour du point de fonctionnement $U^* = 5$. Pour ce faire :
  1. Connecter la platine du procédé au boîtier d'interface :
     - relier le point de mesure du niveau X_r (à gauche du réservoir) à l'entrée *In 0* du boîtier.
     - relier le point d'entrée de la commande U à la sortie *Out 0* du boîtier.
     - Attention on n'utilise pas dans le cadre de ce TP la mesure du débit d'entrée (notée également X_r) et située à gauche de la platine.
  2. Lancer Matlab sur le PC hôte.
  3. Taper la commande *>>TP_nivodo*
  4. Choisir l'option boucle ouverte.
  5. Régler la période d'échantillonnage à 1 secondes.
  6. Régler le temps d'expérimentation à 750s pour laisser au régime permanent dû à l'échelon le temps de s'établir, et fixer l'instant d'application de l'échelon à environ $100s$ afin de permettre au système de se positionner sur son point de fonctionnement. De manière à atteindre le point de fonctionnement plus rapidement, fermer la vanne de sortie jusqu'à atteindre un niveau d'environ 20, puis replacer la vanne au 2/3. Ne plus toucher à cette vanne dans la suite de la manipulation, car le repositionnement approximatif affecte très sensiblement les résultats.
  7. Régler les deux paramètres U^* et ΔU pour fixer la valeur de la commande.
  8. Lancer la mesure en cliquant sur "Marche".
- En déduire les paramètres de la fonction de transfert.

Remarque 3 *Pour ces tracés et l'estimation de la fonction de transfert associée penser à retrancher aux vecteurs de résultats*

Mesure et Commande la valeur du point de fonctionnent (U^*, X_m^*) *car les valeurs mesurées sont celles du procédé réel et non de son approximation linéaire.*

3.4 Mise en place d'un correcteur de type PID

Le système est maintenant mis en boucle fermée sur la principe de la figure 10.

Le procédé est commandé numériquement (bloqueur d'ordre 0) autour d'un point de fonctionnement (U^*, X_m^*).

Le correcteur PID continu possède la fonction de transfert continue suivante :

$$PID(s) = K_p + K_d s + K_i \frac{1}{s} \tag{7}$$

Les paramètres saisis sur l'écran de contrôle sont ceux du correcteur continu (Eq. 7). On supposera que la phase de codage de XPC Target utilise une discrétisation de type *avant* ($p = \frac{z-1}{T_e}$) pour réaliser le correcteur numérique.

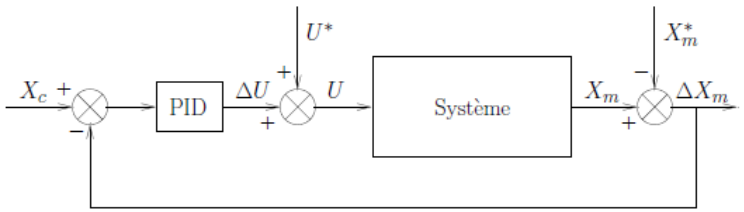


FIG. 10 – *Schéma bloc du système en Boucle Fermée*

Étude de la stabilité dans le cas d'un correcteur proportionnel P

L'objectif est de rechercher théoriquement et expérimentalement la limite de stabilité du système bouclé avec un correcteur proportionnel et d'étudier le comportement du système autour de cette limite.

Étude théorique et simulation On veut étudier la stabilité du système bouclé pour différentes valeurs de la période d'échantillonnage (entre 1 et 20 secondes). Le logiciel de simulation utilisé est Matlab et on fait appel à l'outil *sisotool* qui permet d'étudier un système bouclé selon le schéma indiqué à l'écran.

- Trouver théoriquement $G(z)$ en discrétisant $G(s)$.
- Tracer la courbe critique de l'auto-oscillation $T_e = f(K_p)$ ou $K_p = g(T_e)$ pour T_e entre 1 et 20 secondes.
- Trouver le gain critique pour une période d'échantillonnage $T_e = 4s$.
- Créer la fonction de transfert discrète $G(z)$ en utilisant la commande tf.
- Lancer l'application *sisotool*, importer dans G la fonction de transfert discrétisée du procédé et rechercher manuellement (poignée rouge) la valeur limite K_{lim} pour laquelle le système en boucle fermée devient théoriquement instable.
- Vérifier ces résultats sous Simulink.

Vérification expérimentale On souhaite vérifier expérimentalement la perte de stabilité. Afin de réaliser la manipulation suivante vous devez :

1. Commuter l'interface en mode boucle fermée et sélectionner dans la partie commande uniquement le gain P.
2. Positionner les valeurs du couple (U^*, X_m^*) pour être dans le cas de l'étude considéré ci-dessus.
3. Choisir une période d'échantillonnage $T_e = 4s$ (l'expérience peut être répétée avec d'autres valeurs), et se placer sur le gain limite précédemment trouvé.

Étude de la précision dans le cas d'un correcteur proportionnel P

Il s'agit ici de constater l'influence de la valeur du correcteur proportionnel sur l'erreur statique.

- ▶ Toujours en restant avec un simple correcteur P, réaliser une réponse à un échelon unitaire pour plusieurs valeurs de K et pour une période d'échantillonnage donné (1 seconde par exemple).
- ▶ Pour chacun de ses essais mesurer l'erreur statique. Comparer avec la théorie.

Remarque 4 *Les expériences correspondant à ces essais étant relativement longues, il est judicieux de sauvegarder l'expérimentation dès qu'elle est terminée, puis relancer immédiatement une nouvelle expérience. Cela permet de traiter les données pendant qu'un nouvel essai se déroule.*

Réglage d'un correcteur PI

Afin de corriger l'erreur statique on se propose d'insérer l'intégrateur du PID.

Étude théorique Soit le correcteur continu :

$$PI(s) = K_p + K_i \frac{1}{s} \tag{8}$$

- ▶ Montrer que le correcteur continu (Eq. 8) se discrétise sous la forme (Eq. 9) :

$$PI(z) = \frac{K_{pi}(z + z_{pi})}{z - 1} \tag{9}$$

- ▶ Exprimer K_{pi} et z_{pi} en fonction de K_p et K_i.

Simulations

- ▶ Sous Matlab créer un correcteur discret (Te évidemment à 1 seconde) PI correspondant à $Kp = 1$ et $Ki = 1$.
- ▶ Lancer l'outil rltool de Matlab et importer le système discrétisé dans G et le correcteur PI dans C. Régler ensuite manuellement la position de z_{pi} (qui apparaît comme un cercle rouge) et le gain (poignée rouge) afin de positionner les pôles du système en boucle fermée le plus près possible de l'origine.
- ▶ A l'aide du menu *Compensators/Edit*, relever les valeurs des paramètres du correcteur. En déduire les valeurs correspondantes de K_p et K_i.

Vérification expérimentale

- ▶ Expérimentalement, introduire sur l'écran de contrôle les valeurs de K_p et K_i ainsi trouvées et étudier la réponse à un échelon.
- ▶ Qu'observe-t-on et pourquoi ne retrouve-t-on pas les résultats théoriques ?

Réglage d'un correcteur PID

L'ajout de la partie intégrale annule les erreurs statiques mais introduit des oscillations. La mise en place d'une partie dérivée aura pour effet de corriger ce travers.

- ▶ Expérimentalement, chercher à régler les trois gains du correcteur pour annuler les erreurs statiques tout en minimisant les oscillations.
- ▶ Conclure

ASNL : Méthode du 1er harmonique
Etude d'un asservissement de position à relais

4 Étude d'un asservissement de position à relais

4.1 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.

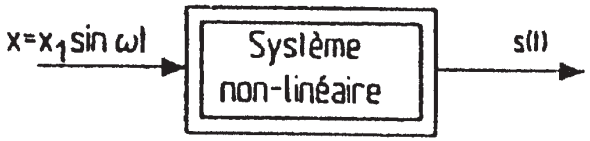


FIG. 11 – Élément non linéaire

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,) s'appelle GAIN COMPLEXE EQUIVALENT.

Étude de la stabilité d'un asservissement non linéaire bouclé

Considérons le système de la figure 12.

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

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

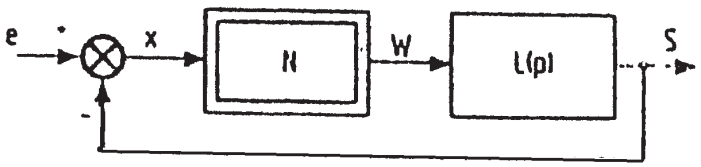


FIG. 12 – Bouclage non linéaire

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 13)

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

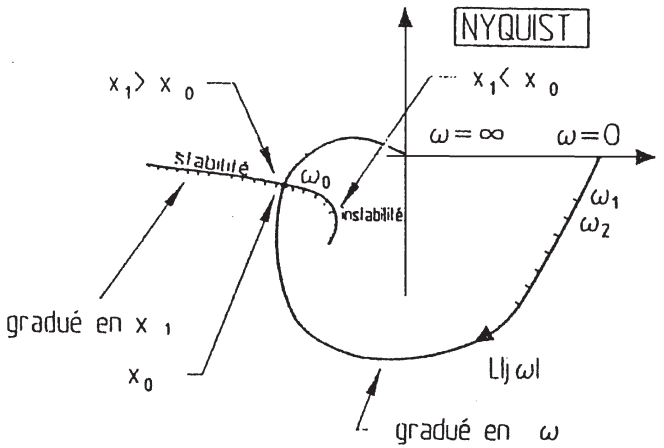


FIG. 13 – Critère géométrique de détermination d'un cycle limite

4.2 Matériel utilisé

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

Remarque 5 *: 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 $\tau = 0.214 s$.*

4.3 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.

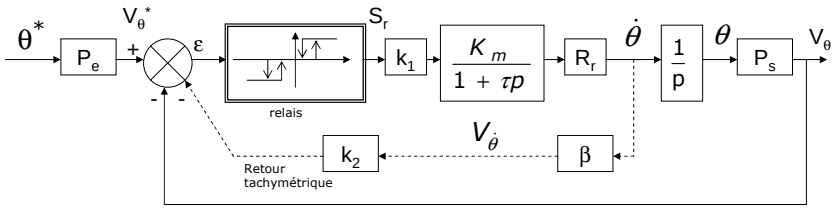


FIG. 14 – Montage des différents éléments

Montage

- Réaliser l'asservissement de position sans le retour tachymétrique représenté par le schéma bloc suivant (Fig. 14) 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 **Main Delayed>XY**, et en se servant du mode persistant de l'oscilloscope **Display**> ∞**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.

4.4 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

Commande numérique
Commande d'un moteur électrique par ordinateur

5 Commande d'un moteur électrique par ordinateur

5.1 But de la manipulation

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.

5.2 Identification du procédé

On dispose d'un moteur électrique, que l'on souhaite commander par xPC Target sous Simulink. Pour une présentation d'xPC Target, se référer à l'annexe A. Seule une voie CAN (*bloc Simulink PCI-6024E AD*) pour acquérir le signal de sortie et une voie CNA (*bloc Simulink PCI-6024E DA*) pour envoyer le signal de commande sont utiles. 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 .mdl qui va gérer la commande. Mettre en place une entrée **échelon** à 4 connectée au bloc CNA.
- ▶ Aller dans *Simulation→ Simulation parameters*. Dans *Solver*, prendre Inf en durée de simulation, type : fixed step, ode4 (Runge Kutta), fixed step size : 0.01. Dans *Real Time Workshop*, renseigner le champs "System target file" par "xpctarget.tlc", puis cliquer sur **Build**. On génère ainsi un programme temps réel qui va être chargé puis lancé sur le moteur. Attention : si vous changez de commande, il faudra toujours recliquer sur *Build* pour relancer la nouvelle 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.

5.3 Régulation numérique avec correcteur proportionnel

On met en place une régulation échantillonnée suivant le schéma de la figure 15.

Etude de la réponse indicielle

- ▶ Calculer la fonction de transfert échantillonnée du moteur lorsque celui est précédé d'un convertisseur numérique ana-

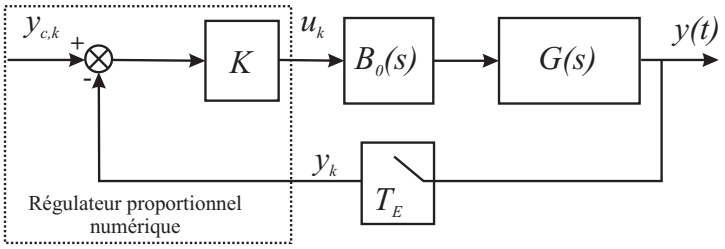


FIG. 15 – Principe de bouclage discret

logique fonctionnant en bloqueur d'ordre zéro :

$$G(Z) = \mathcal{Z}\left[B_0(p)G(p)\right] \tag{10}$$

- ▶ Calculer l'erreur statique de la régulation échantillonnée pour un gain de commande $K = 1$ et une période d'échantillonnage $T_E = 0.35s$.
- ▶ Vérifier le résultat par une simulation en créant l'application `Simulink` correspondante.
- ▶ Vérifier le résultat expérimentalement.

Etude de stabilité

- ▶ En recherchant les conditions de stabilité limite (stabilité simple), calculer et représenter la courbe de stabilité critique :

$$K = f(T_E) \tag{11}$$

où K représente le gain du régulateur proportionnel et T_E la période d'échantillonnage.

- ▶ Pour la période d'échantillonnage $T = 0,35s$, 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.

5.4 Mise en oeuvre d'une commande pile

Etude théorique

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 16.

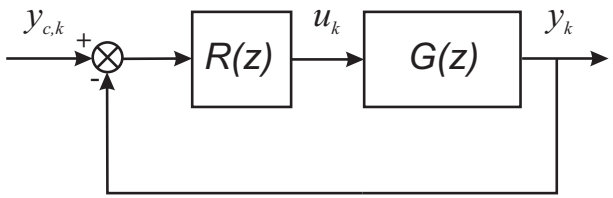


FIG. 16 – Correcteur discret

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 :

$$\frac{Y(z)}{Y_c(z)} = z^{-(d+1)} \tag{12}$$

où le paramètre $d = \frac{\tau_r}{T_E}$ représente le retard pur du procédé.

Mise en oeuvre sur le moteur

Application au cas de la régulation de vitesse du moteur.

- ▶ Donner l'expression de $R(z)$.
- ▶ Vérifier par simulation le résultat attendu puis comparer avec des mesures expérimentales.
- ▶ Conclure

Le robot legot NXT
Commande optimale : le pendule inversé

6 Commande optimale : robot legot NXT

6.1 But de la manipulation

L'objectif de cette manipulation est de réguler le robot lego NXT 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 (Matlab embedded coder notamment).
- 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.

6.2 Présentation de la maquette

On travaille avec un robot NXTway-GS. C'est un robot à deux roues mobiles. Les entrées et les sorties du système sont représentées sur la Figure 17.

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

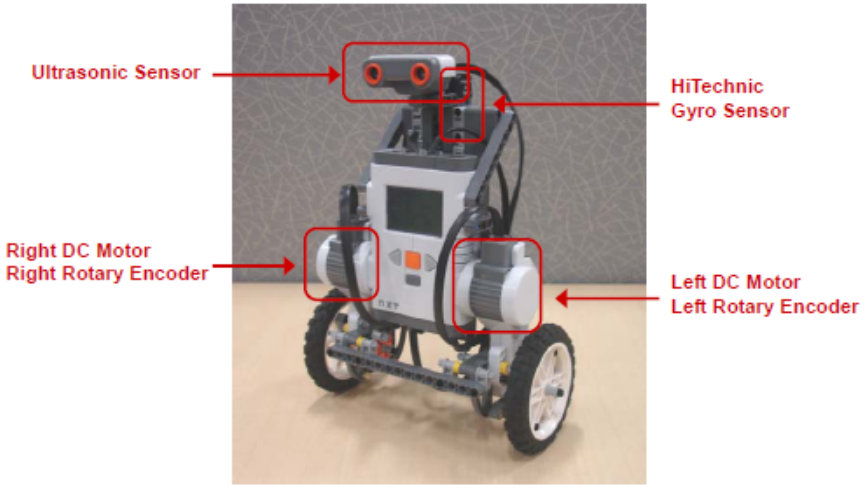


FIG. 17 – le robot NXTway-GS

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 microprocesseur 32 bits ARM7.

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

6.3 Présentation du dossier RobotPendule

Copier sur votre répertoire, le dossier RobotPendule contenu dans Commetud. Ce dossier comporte 3 fichiers Simulink :

- *nxtway_gs.mdl* contenant le simulink principal
- *nxtway_gs_controller.mdl* contenant le simulink du correcteur, c'est sur ce fichier que vous allez travailler.
- *nxtway_gs_plant.mdl* contenant le modèle physique du robot.

Il comporte aussi 5 fichiers Matlab :

- *param_nxtway_gs.m* permettant d'effectuer les 3 fichiers matlab suivants
- *param_controller.m* contenant les paramètres du correcteur, c'est sur ce fichier que vous allez travailler.
- *param_plant.mdl* contenant les paramètres du modèle physique du robot
- *param_sim.m* contenant les paramètres de la simulation
- *iswall.m* contenant la procédure permettant d'arreter la simulation et d'afficher un message d'erreur lorsque le robot rencontre un mur.

Il comporte aussi un fichier *track.bmp* permettant de définir l'environnement dans lequel évolue le robot.

6.4 Présentation du modèle Simulink du robot

Le modèle Simulink fourni va permettre d'une part de simuler le comportement du robot, d'autre part d'effectuer facilement la chaîne de conception et implémenter une commande sur le microprocesseur du robot. Une présentation détaillée du modèle est donnée en annexe.

Le modèle Simulink principal est nxtway_gs.mdl (voir Figure 18).

Pour lancer une simulation :

- Lancer nxtway_gs.mdl

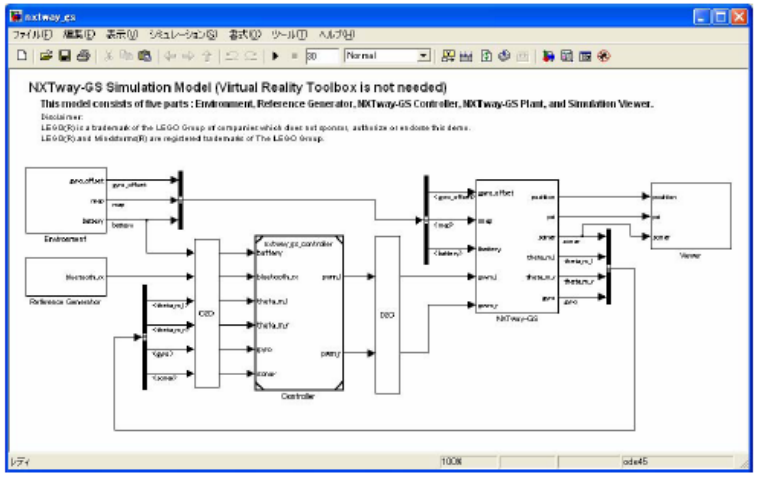
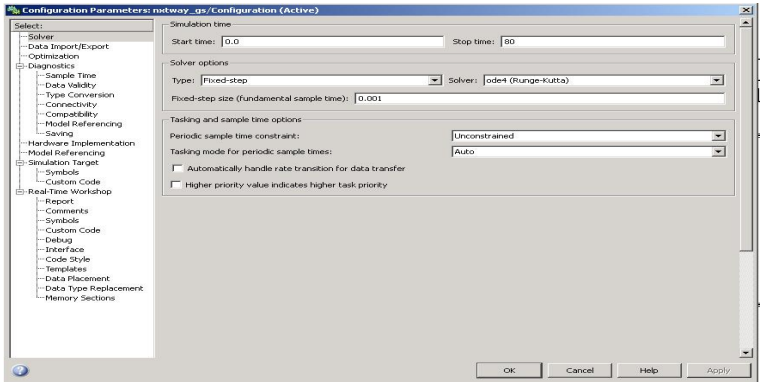


FIG. 18 – Le modèle nxtway_gs.mdl

- lancer param_nxtway_gs.m pour charger les paramètres du modèle
- Dans 'Simulation>Configuration Parameters', choisir les paramètres comme sur la figure



- dans nxtway_gs.mdl, cliquer sur 'Reference Generator', puis sur 'Signal Builder' où l'on peut choisir le profil de test (Figure 19).

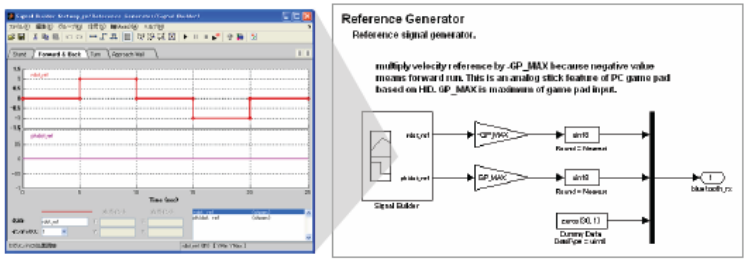


FIG. 19 – Le sous-système de génération d'entrées

- Lancer la simulation
- Pour visualiser les résultats, cliquer sur 'Viewer' ; on a ainsi accès aux courbes de x, y, et ψ (Figure 20).

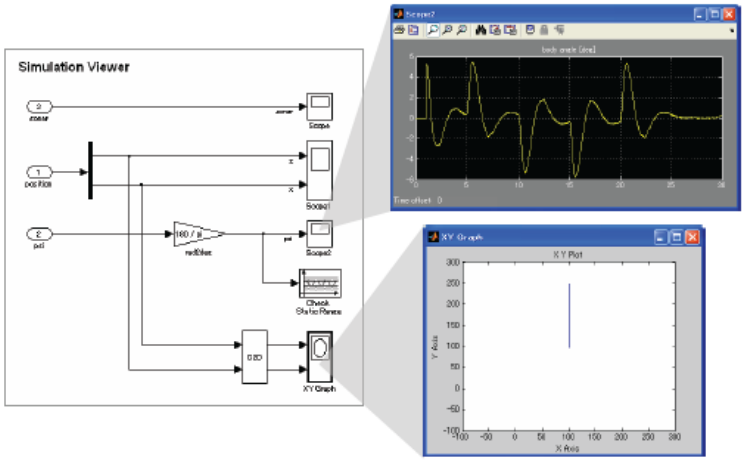


FIG. 20 – Le sous-système de visualisation

Pour compiler et charger le programme dans le robot :

- Après avoir calculé les paramètres du retour d'état, se mettre sur la fenêtre du modèle 'nxtway_gs_controller'.
- Dans la fenêtre de commande de Matlab, taper
  > `nxtbuild('ntxway_app','build')`
- allumer le robot en appuyant sur le bouton orange, vérifier que le robot est branché au port USB de l'ordinateur de développement.
- Dans 'My Files', 'Software files', supprimer le projet précédent
- dans la fenêtre de commande de Matlab, taper
  > `nxtbuild('ntxway_app','rxeflash')`
  Un nombre de bits transmis doit être affiché. Par exemple :
  > `nxtway_app.rxe=28384`
- Débrancher le robot. Sur le robot, cliquer sur 'Software Files', cliquer sur 'nxtway_app', puis sur Run, et sur le bouton droit sur 'Run' en maintenant le robot vertical au sol jusqu'à entendre un bip. Lâcher le robot. Savourer.

6.5 Manipulation

Commande LQR

Simulation Dans une première approche, on souhaite contrôler le robot par retour d'Etat. Afin de déterminer le gain de ce retour d'état, on utilise les principes de commande LQR (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 :

- ▶ Compléter le fichier matlab param_controller. Il vous suffit de définir les matrices de pondération QQ et RR. Ce fichier vous calcule le gain de retour d'état appelé KK ici.
- ▶ Modifier le simulink du controller, pour cela, ouvrir le fichier nxtway_gs_controller.mdl, cliquer sur 'nxtway_app', puis sur 'Balance & Drive Control', puis sur 'Controller'. 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 Q et R 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 Q et R ont de l'importance dans la régulation, sur quelles caractéristiques ?
- ▶ Quelles valeurs de Q et R conservez-vous ? Justifier votre réponse.

Test sur le robot réel Une fois que vous aurez déterminé par simulation un réglage optimal pour Q et R, tester votre correcteur sur le robot. Regarder les effets des perturbations sur votre réglage.

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 25.

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.

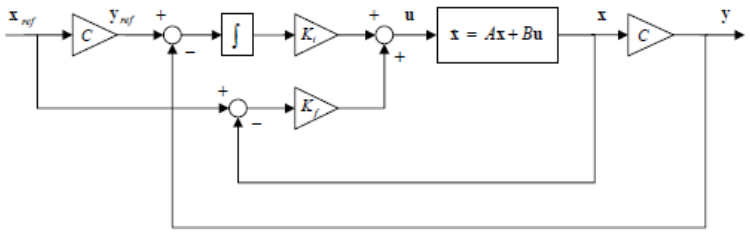


FIG. 21 – Schéma bloc d'une commande par retour d'état avec action intégrale

Commande par placement de pôles

La commande LQR 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 LQR de façon à essayer d'améliorer la dynamique.

- ▶ Trouver les valeurs propres imposées par la commande LQR 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.

6.6 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.

6.7 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
xPC Target

A xPC Target

A.1 Introduction et principes de fonctionnement

xPC Target est un environnement logiciel qui permet de mettre au point, de tester et de mettre en œuvre des applications temps réel sur du matériel de type PC. Cet outil offre l'avantage d'être associé à Matlab/Simulink, ce qui permet de passer de la phase essai/simulation à la phase mise en œuvre/expérimentation très rapidement et très facilement en s'affranchissant de l'étape de codage. La configuration matérielle standard est composée de 2 unités informatiques reliées par une liaison série asynchrone.

Le PC hôte sert d'unité de développement. A partir de Matlab/Simulink, il est possible d'engendrer directement l'application temps réel. Il fonctionne dans l'environnement Windows habituel.

Le PC cible est orienté temps réel. Il est donc démarré (à l'aide d ?une disquette ou d'une EPROM) non pas sur DOS ou Windows mais sur un noyau temps réel spécifique. Ce noyau, entre autres choses, permet de recevoir et faire exécuter le code engendré par le PC hôte et d'assurer la communication entre les deux machines (téléchargement, mise en marche, adaptation en ligne de paramètres, rapatriement de données ...). Enfin ce PC est équipé de cartes (dans notre cas deux cartes National Instrument PCI_6024E avec convertisseurs analogique-numérique et convertisseurs numérique-analogique) qui lui permettent de commander le processus.

La cible est équipée d'un moniteur sur lequel sont visualisés deux types d'information : des informations liées à l'exécution temps réel de l'application en cours (nom de l'application, temps courant, période d ?échantillonnage ...) ainsi qu'un rapport sur l'exécution des commandes envoyées par le PC hôte (adaptation de paramètres, mise en marche, ...). Le second groupe d'informations est graphique et dépend de l'application.
La Figure 22 présente les trois entités manipulées en TP : le PC de développement avec Matlab/Simulink, le PC cible et le système à contrôler.

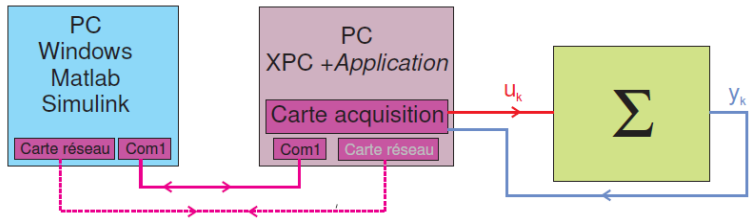


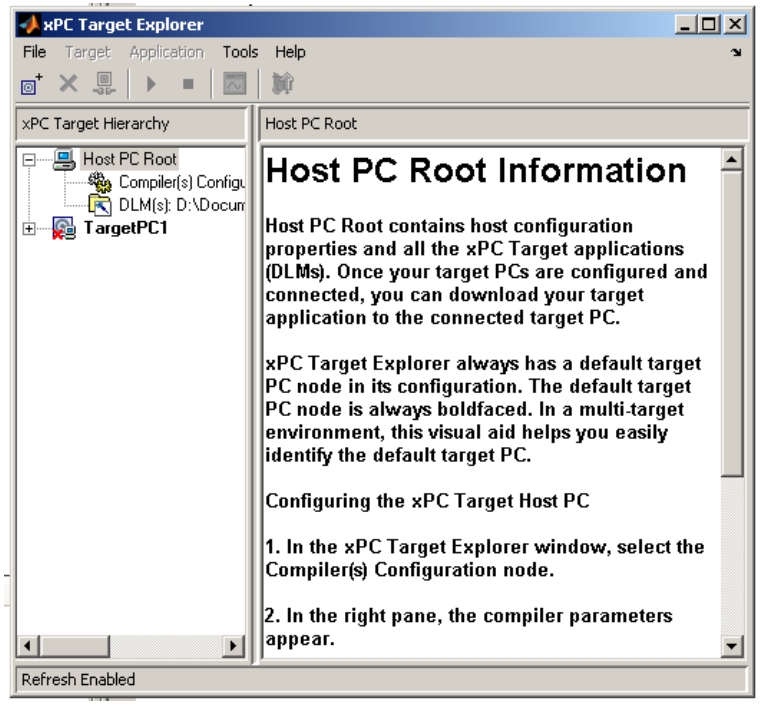
FIG. 22 – Commande déportée sous XPC Target

A.2 Configuration d'xPC Target

Démarrez le PC cible.

1. **Création de la communication entre le PC hôte et le PC cible :** Il s'agit de configurer l'environnement logiciel

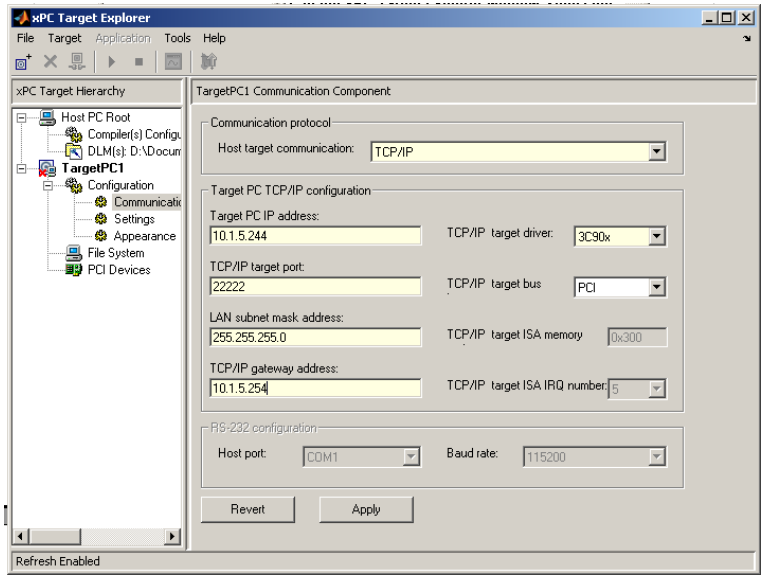
XPCTarget. Dans la fenêtre Command Window de Matlab, tapez la commande
> `xpcexplr`
qui vous ouvre la fenêtre de dialogue de XPCTarget, comme sur la figure suivante :



Ouvrez l'arborescence appelée TargetPC1. Sélectionnez l'onglet appelé Communication. Il faut maintenant spécifiez le type de communication. Il s'agit ici d'une liaison Ethernet. Dans le menu déroulant *Host Target Communication* choisir TCP/IP. Il faut alors fournir l'adresse IP (Target PC IP address), le port (TCP/IP Target Port), le masque de sous-réseau (LAN subnet mask address), l'adresse du Gateway (TCP/IP gateway address) et le driver (TCP/IP Target Driver) de votre PC Cible. Tous ces renseignements se trouvent sur l'écran de votre PC cible :

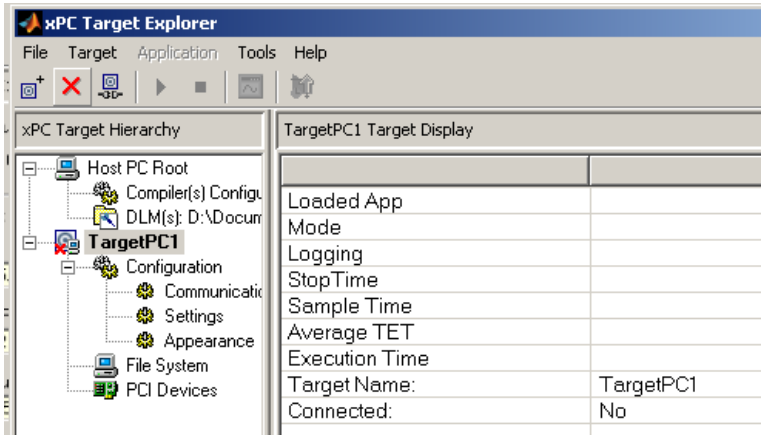
- Target PC IP address : IP Add
- TCP/IP Target Port : Port
- LAN subnet mask address : Subnet
- TCP/IP gateway address : Gateway
- TCP/IP Target Driver : Board

Remplissez les différents champs de la fenêtre de configuration, vous devez obtenir quelque chose de similaire à l'impression d'écran suivante.

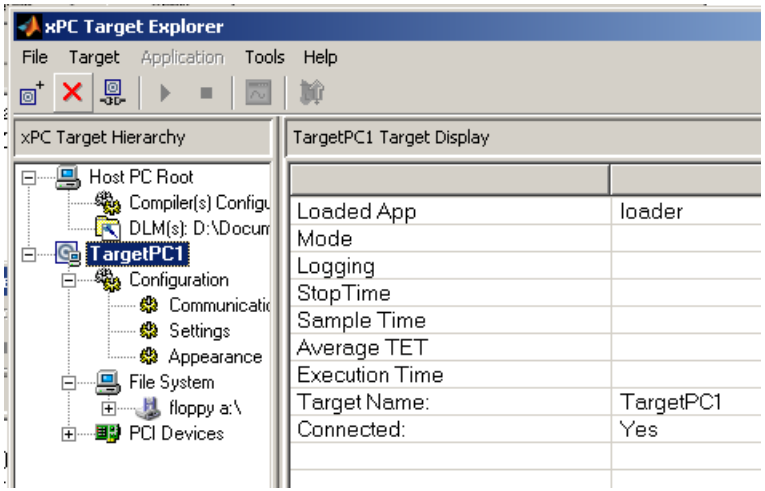


Sauvegardez vos modifications en cliquant sur le bouton *Apply*.

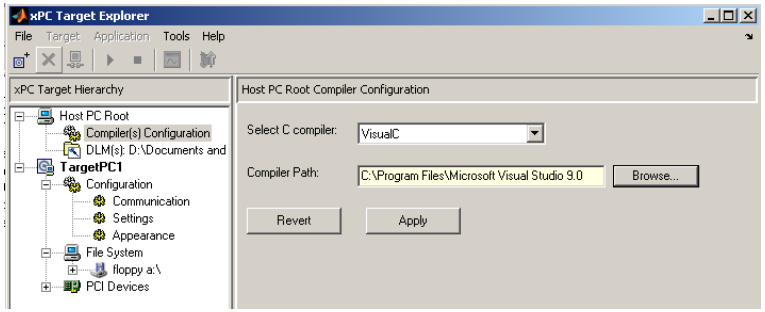
Sélectionnez l'onglet TargetPC1. Cliquez alors sur l'icône permettant de connecter le PC hôte et le PC cible.



A côté de « Connected », le No se transforme en Yes, la connexion entre le PC cible et le PC hôte est réalisée.

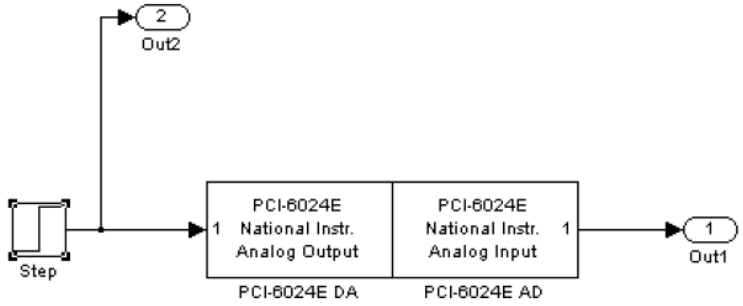


2. **Choix du compilateur d'XPCTarget :** Il faut maintenant spécifier à XPCTarget, le complilateur qu'on va utiliser pour traduire le programme réalisé sous Matlab/Simulink en instruction pour le PC Cible.
   Sélectionnez l'onglet **Compiler(s) Configuration**. Le compilateur sera VisualC. Spécifiez le chemin d'accès au compilateur C :\Program Files\Microsoft Visual Studio 9.0.



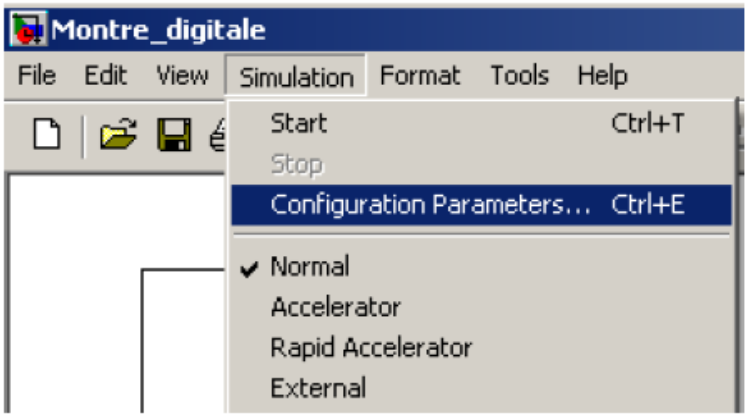
   Sauvegardez vos modifications en cliquant sur le bouton *Apply*.

3. **Compilation d'un modèle développé sous Simulink :** Une fois le modèle réalisé, par exemple celui de la Figure suivante.

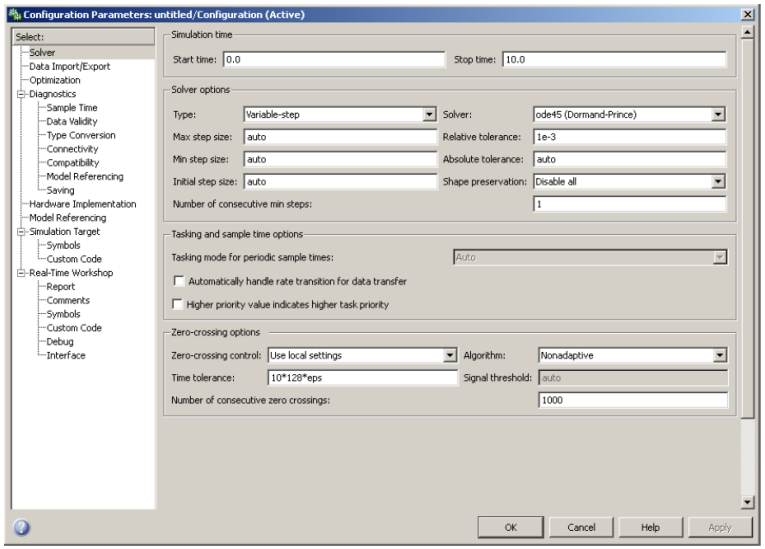


   Remarque : pour récupérer les données sur Matlab, on utilise le bloc « out » de simulink.

   Pour compiler le programme, cliquez sur le menu **Simulation/ Configuration Parameters**

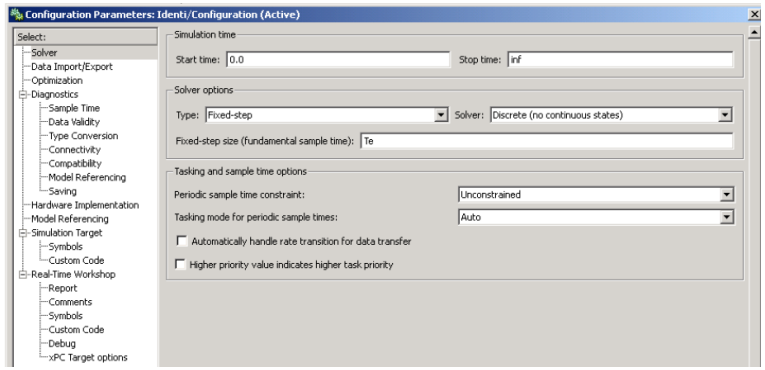


L'écran suivant apparaît :



Sélectionnez l'onglet **Solver**. Dans le parapraphe *Simulation Time* : Mettez « inf » dans Stop Time. Ce réglage permet de fixer le temps au bout duquel le programme doit s'arrêter, ici on met inf pour infini, le programme tournera donc sans interruption sur la cible.

Dans le paragraphe *Solver options* : Choisissez « Fixed-Step » dans Type. Mettez « Discrete » dans Solver. Tapez « Te » dans Fixed-Step size. Sauvegardez vos modifications en cliquant sur le bouton *Apply*. Vous devez avoir l'écran suivant :

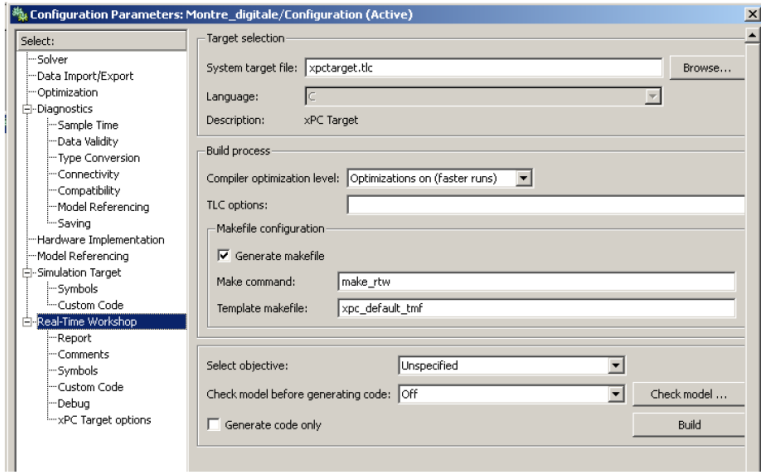


Le paramètre Te sera le pas de calcul de l'horloge interne du PC Cible. Il faudra le spécifier sous Matlab : Dans la fenêtre Command Window de Matlab, fixer le pas de calcul.

Sélectionnez l'onglet **Real-Time Workshop**. Dans le parapraphe *Target selection* : Choisissez à l'aide du bouton Browse « xpctarget.tlc» dans System target file.

Sauvegardez vos modifications en cliquant sur le bouton *Apply*.

Pour compiler le programme, il vous suffit de cliquer sur le bouton *Build*.



La compilation prend un certain temps, elle est terminée lorsque vous avez la possibilité de taper de nouveau une commande dans la fenêtre Command Window de Matlab.

4. **Execution du programme sur le PC cible.** Pour lancer le programme sur le PC cible, il faut taper la commande
   `> +tg`
   dans la fenêtre Command Window de Matlab.

   Lorsque le PC cible exécute un programme, il affiche un certain nombre de données,
   - le temps au bout duquel il arrête l'exécution du programme (StopTime) ici réglé à l'infini
   - le pas temporel d'échantillonnage (StepTime) ici réglé à l'aide de Te
   - Depuis combien de temps il exécute le programme (Execution)

   Pour arrêter le programme sur le PC cible, il faut taper la commande
   `> -tg`
   dans la fenêtre Command Window de Matlab.

   On remarque qu'au lieu d'afficher le temps d'exécution, le PC Cible affiche stopped. Le programme est bien arrêté.

5. **Récupération des données sous Matlab.** Pour récupérer les données issues d'un bloc "Out" utilisé dans un modèle tournant sous xPC, il faut taper la commande :
   `> y= tg.OutputLog ;`
   `> t= tg.TimeLog ;`
   Après cette commande, y contient toutes les données des blocs "Out" du modèle concerné et t le vecteur temps.
   Par exemple, pour tracer les courbes des données du modèle Figure 2, on tape la commande :
   `> plot (t, y( :,1))`
   pour tracer la sortie et
   `> plot (t, y( :,2))`

ANNEXE
Commande par retour d'état

B 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} et la valeur de l'état mesurée x par un gain de retour K et d'utiliser cette valeur en commande. De fait, la commande par retour d'état est identique à une régulation par un Proportionnel Dérivé (PD) dans la théorie de la commande classique. La figure 23 montre le schéma bloc d'une commande par retour d'état.

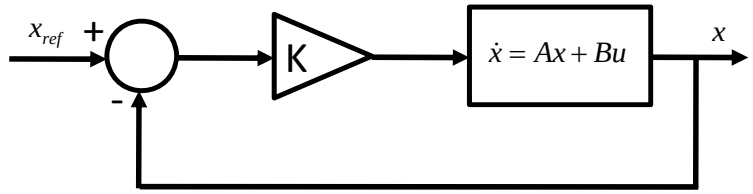


FIG. 23 – Schéma bloc de la commande par retour d'état

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

$$u(t) = -K(x(t) - x_{ref}) \quad (13)$$
$$\dot{x}(t) = (A - BK)x(t) + BKx_{ref} \quad (14)$$

Il est 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.

$$rg(M_c) = n \quad (15)$$
$$M_c = [B, AB, \dots A^{n-1}B] \quad (16)$$

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

Il y a deux méthodes principales 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. Il faut par contre veiller à ce que la valeur de K ne soit pas aberrante.

La fonction `place` de Matlab permet d'effectuer un placement de pôles direct.

`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 à -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 (LQR)

Temps discret Soit le système représenté par le modèle : $x_{k+1} = Ax_k + Bu_k$

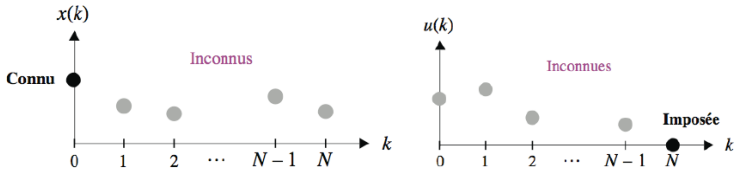


FIG. 24 – Représentation du problème de commande optimale

Le principe de la commande optimale est de s'intéresser à la séquence de commandes futures $u(k)$ afin de satisfaire un critère quadratique. A un instant discret du temps il faut calculer la commande qui fait passer de l'état k à l'état $k+1$ jusqu'à atteindre l'état final x_N. Connaissant le modèle du système, ceci revient à minimiser un critère quadratique sur un horizon N.

$$J = \frac{1}{2}\left(x_N^\top S x_N + \sum_{k=0}^{N}\left(x_k^\top Q x_k + u_k^\top R u_k\right)\right) \tag{17}$$

A l'instant $k = 0$, l'état $x(0)$ est connu, ce que l'on recherche ce sont les N commandes $u(0), u(1) \dots u(N-1)$ (Figure 24) qui minimisent ce critère, sachant que le dernier terme $u(N) = 0$ est imposé et que le terme $x_N^\top S x_N$ représente une contrainte sur l'état final. Ce critère effectue un compromis entre l'erreur de régulation représenté par les termes $x_k^\top Q x_k$ et l'énergie dépensée pour y parvenir $u_k^\top R u_k$. Les matrices de pondération Q, R et S sont les paramètres de conception sur lesquels on peut agir pour modifier la dynamique en boucle fermée.

La résolution de ce problème peut se faire de différentes manières dont la Programmation Non Linéaire que l'on ne détaillera pas ici mais que l'on peut résumer ainsi. On recherche une solution de type retour d'état $u_k = -K_k x_k$.
On montre alors que le gain K_k peut être exprimé par :

$$K_k = R^{-1} B^\top P_k \tag{18}$$

où $P_N = S$ et P est une matrice solution de l'équation dite de Riccati :

$$P_k = Q + A^\top P_{k+1}\left[I + BR^{-1}B^\top P_{k+1}\right]^{-1} A \tag{19}$$

Si l'horizon est suffisant grand (infini) alors la solution devient stationnaire et la matrice K correspondante est solution de :

$$P = Q + A^\top P_{k+1}\left[I + BR^{-1}B^\top P\right]^{-1} A \tag{20}$$

Temps continu Dans le cas continu, la même approche peut être adoptée. La méthode calcule le gain de retour d'état K de manière à minimiser la fonction de coût J suivante :

$$J = \int_0^\infty \left(x(t)^\top Q x(t) + u(t)^\top R u(t)\right) dt \tag{21}$$

De la même manière, cette résolution passe par la détermination d'une matrice P solution de l'équation de Riccati en continu.

$$\dot{P} = PBR^{-1}B^{\top}P - PA - A^{\top}P - Q \tag{22}$$

De la même manière, on peut montrer que pour un horizon infini la matrice P devient stationnaire ($\dot{P} = 0$) et qu'une solution constante pour la matrice P peut être trouvée en résolvant :

$$PBR^{-1}B^{\top}P - PA - A^{\top}P - Q = 0 \tag{23}$$

Les paramètres de choix sont les poids des matrices pour l'état Q, la commande R, et S. Les matrices sont choisies diagonales. S est prise généralement égale à l'identité. Une solution généralement adoptée pour avoir déjà une indication des ordres de grandeur de ces coefficients est de les déterminer grâce au modèle. 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
```

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. On considère donc une commande de la forme[2] :

$$u(t) = -K_f(x(t) - x_{ref}) + K_i \int_0^t (y(t) - y_{ref})dt \tag{24}$$

appliqué au système dynamique :

$$\dot{x} = Ax + Bu \tag{25}$$
$$y = Cx \tag{26}$$

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

$$u() = -\begin{bmatrix} K_f & K_i \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} K_f & K_i \end{bmatrix} \begin{bmatrix} x_{ref} \\ 0 \end{bmatrix} \tag{27}$$

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

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ C \end{bmatrix} x_{ref} \tag{28}$$

Quand on considère le système stabilisé, on a alors $y_\infty = y_{ref}$. Le schéma bloc d'une telle commande est illustré Figure 25.

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)

[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.

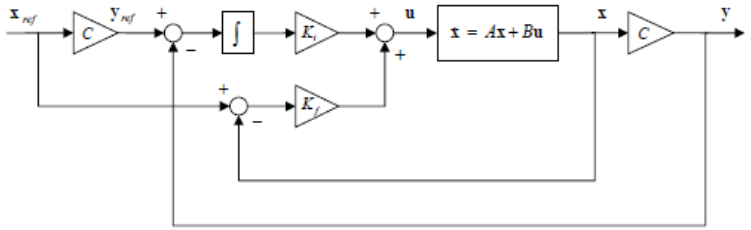


FIG. 25 – Schéma bloc d'une commande par retour d'état avec action intégrale

ANNEXE C – Modèle mathématique du robot lego NXT

tiré de la notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.

3 NXTway-GS Modeling

This chapter describes mathematical model and motion equations of NXTway-GS.

3.1 Two-Wheeled Inverted Pendulum Model

NXTway-GS can be considered as a two wheeled inverted pendulum model shown in Figure 3-1.

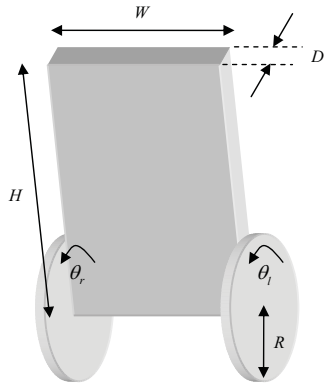
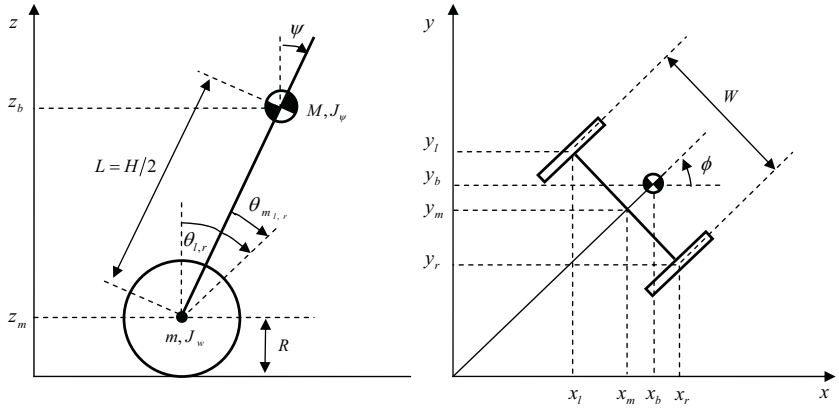


Figure 3-1 Two-wheeled inverted pendulum

Figure 3-2 shows side view and plane view of the two wheeled inverted pendulum. The coordinate system used in 3.2 Motion Equations of Two-Wheeled Inverted Pendulum is described in Figure 3-2.



ψ : body pitch angle $\theta_{l,r}$: wheel angle (l, r indicates left and right) $\theta_{m_{l,r}}$: DC motor angle

Figure 3-2 Side view and plane view of two-wheeled inverted pendulum

Physical parameters of NXTway-GS are the following.

$g = 9.81$	$[m/\sec^2]$	:	Gravity acceleration
$m = 0.03$	$[kg]$	:	Wheel weight
$R = 0.04$	$[m]$	:	Wheel radius
$J_W = mR^2/2$	$[kgm^2]$	:	Wheel inertia moment
$M = 0.6$	$[kg]$	:	Body weight
$W = 0.14$	$[m]$	:	Body width
$D = 0.04$	$[m]$	:	Body depth
$H = 0.144$	$[m]$	:	Body height
$L = H/2$	$[m]$	:	Distance of the center of mass from the wheel axle
$J_\psi = ML^2/3$	$[kgm^2]$	:	Body pitch inertia moment
$J_\phi = M(W^2 + D^2)/12$	$[kgm^2]$	:	Body yaw inertia moment
$J_m = 1\times10^{-5}$	$[kgm^2]$	:	DC motor inertia moment
$R_m = 6.69$	$[\Omega]$	:	DC motor resistance
$K_b = 0.468$	$[V\sec/rad]$	:	DC motor back EMF constant
$K_t = 0.317$	$[Nm/A]$	:	DC motor torque constant
$n = 1$		:	Gear ratio
$f_m = 0.0022$		:	Friction coefficient between body and DC motor
$f_W = 0$		:	Friction coefficient between wheel and floor.

- We use the values described in reference [2] for R_m, K_b, K_t.
- We use the values that seems to be appropriate for J_m, n, f_m, f_W, because it is difficult to measure.

3.2 Motion Equations of Two-Wheeled Inverted Pendulum

We can derive motion equations of two-wheeled inverted pendulum by the Lagrangian method based on the coordinate system in Figure 3-2. If the direction of two-wheeled inverted pendulum is x-axis positive direction at $t = 0$, each coordinates are given as the following.

$$(\theta,\ \ \phi) = \left(\frac{1}{2}(\theta_l + \theta_r)\ \ \ \frac{R}{W}(\theta_r - \theta_l)\right) \tag{3.1}$$

$$(x_m,\ \ y_m,\ \ z_m) = \left(\int \dot{x}_m dt,\ \ \int \dot{y}_m dt,\ \ R\right),\ \ (\dot{x}_m,\ \ \dot{y}_m) = \left(R\dot{\theta}\cos\phi\ \ \ R\dot{\theta}\sin\phi\right) \tag{3.2}$$

$$(x_l,\ \ y_l,\ \ z_l) = \left(x_m - \frac{W}{2}\sin\phi,\ \ y_m + \frac{W}{2}\cos\phi,\ \ z_m\right) \tag{3.3}$$

$$(x_r,\ \ y_r,\ \ z_r) = \left(x_m + \frac{W}{2}\sin\phi,\ \ y_m - \frac{W}{2}\cos\phi,\ \ z_m\right) \tag{3.4}$$

$$(x_b,\ \ y_b,\ \ z_b) = (x_m + L\sin\psi\cos\phi,\ \ y_m + L\sin\psi\sin\phi,\ \ z_m + L\cos\psi) \tag{3.5}$$

The translational kinetic energy T_1, the rotational kinetic energy T_2, the potential energy U are

$$T_1 = \frac{1}{2}m\left(\dot{x}_l^{\,2} + \dot{y}_l^{\,2} + \dot{z}_l^{\,2}\right) + \frac{1}{2}m\left(\dot{x}_r^{\,2} + \dot{y}_r^{\,2} + \dot{z}_r^{\,2}\right) + \frac{1}{2}M\left(\dot{x}_b^{\,2} + \dot{y}_b^{\,2} + \dot{z}_b^{\,2}\right) \tag{3.6}$$

$$T_2 = \frac{1}{2}J_w\dot{\theta}_l^{\,2} + \frac{1}{2}J_w\dot{\theta}_r^{\,2} + \frac{1}{2}J_\psi\dot{\psi}^{\,2} + \frac{1}{2}J_\phi\dot{\phi}^2 + \frac{1}{2}n^2J_m(\dot{\theta}_l - \dot{\psi})^2 + \frac{1}{2}n^2J_m(\dot{\theta}_r - \dot{\psi})^2 \tag{3.7}$$

$$U = mgz_l + mgz_r + Mgz_b \tag{3.8}$$

The fifth and sixth term in T_2 are rotation kinetic energy of an armature in left and right DC motor. The Lagrangian L has the following expression.

$$L = T_1 + T_2 - U \tag{3.9}$$

We use the following variables as the generalized coordinates.

- θ : Average angle of left and right wheel
- ψ : Body pitch angle
- ϕ : Body yaw angle

Lagrange equations are the following

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right) - \frac{\partial L}{\partial\theta} = F_\theta \tag{3.10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\psi}}\right) - \frac{\partial L}{\partial\psi} = F_\psi \tag{3.11}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\phi}}\right) - \frac{\partial L}{\partial\phi} = F_\phi \tag{3.12}$$

We derive the following equations by evaluating Eqs. (3.10) - (3.12).

$$\left[(2m+M)R^2 + 2J_w + 2n^2J_m\right]\ddot{\theta} + \left(MLR\cos\psi - 2n^2J_m\right)\ddot{\psi} - MLR\dot{\psi}^2\sin\psi = F_\theta \tag{3.13}$$

$$\left(MLR\cos\psi - 2n^2J_m\right)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\sin\psi - ML^2\dot{\phi}^2\sin\psi\cos\psi = F_\psi \tag{3.14}$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2J_m\right) + ML^2\sin^2\psi\right]\ddot{\phi} + 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi = F_\phi \tag{3.15}$$

In consideration of DC motor torque and viscous friction, the generalized forces are given as the following

$$\left(F_\theta,\quad F_\psi,\quad F_\phi\right)=\left(F_l+F_r,\quad F_\psi,\quad \frac{W}{2R}\left(F_r-F_l\right)\right) \tag{3.16}$$

$$F_l = nK_t i_l + f_m(\dot{\psi}-\dot{\theta}_l) - f_w\dot{\theta}_l \tag{3.17}$$

$$F_r = nK_t i_r + f_m(\dot{\psi}-\dot{\theta}_r) - f_w\dot{\theta}_r \tag{3.18}$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m(\dot{\psi}-\dot{\theta}_l) - f_m(\dot{\psi}-\dot{\theta}_r) \tag{3.19}$$

where $i_{l,r}$ is the DC motor current.

We cannot use the DC motor current directly in order to control it because it is based on PWM (voltage) control. Therefore, we evaluate the relation between current $i_{l,r}$ and voltage $v_{l,r}$ using DC motor equation. If the friction inside the motor is negligible, the DC motor equation is generally as follows

$$L_m\dot{i}_{l,r} = v_{l,r} + K_b(\dot{\psi}-\dot{\theta}_{l,r}) - R_m i_{l,r} \tag{3.20}$$

Here we consider that the motor inductance is negligible and is approximated as zero. Therefore the current is

$$i_{l,r} = \frac{v_{l,r} + K_b(\dot{\psi}-\dot{\theta}_{l,r})}{R_m} \tag{3.21}$$

From Eq.(3.21), the generalized force can be expressed using the motor voltage.

$$F_\theta = \alpha\left(v_l+v_r\right) - 2\left(\beta+f_w\right)\dot{\theta} + 2\beta\dot{\psi} \tag{3.22}$$

$$F_\psi = -\alpha\left(v_l+v_r\right) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \tag{3.23}$$

$$F_\phi = \frac{W}{2R}\alpha\left(v_r-v_l\right) - \frac{W^2}{2R^2}\left(\beta+f_w\right)\dot{\phi} \tag{3.24}$$

$$\alpha = \frac{nK_t}{R_m},\quad \beta = \frac{nK_tK_b}{R_m} + f_m \tag{3.25}$$

3.3 State Equations of Two-Wheeled Inverted Pendulum

We can derive state equations based on modern control theory by linearizing motion equations at a balance point of NXTway-GS. It means that we consider the limit $\psi \to 0$ ($\sin\psi \to \psi$, $\cos\psi \to 1$) and neglect the second order term like $\dot{\psi}^2$. The motion equations (3.13) – (3.15) are approximated as the following

$$\left[(2m+M)R^2 + 2J_w + 2n^2J_m\right]\ddot{\theta} + \left(MLR - 2n^2J_m\right)\ddot{\psi} = F_\theta \tag{3.26}$$

$$\left(MLR - 2n^2J_m\right)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\psi = F_\psi \tag{3.27}$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2J_m\right)\right]\ddot{\phi} = F_\phi \tag{3.28}$$

Eq. (3.26) and Eq. (3.27) has θ and ψ, Eq. (3.28) has ϕ only. These equations can be expressed in the form

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + G\begin{bmatrix}\theta\\ \psi\end{bmatrix} = H\begin{bmatrix}v_l\\ v_r\end{bmatrix} \tag{3.29}$$

$$E = \begin{bmatrix}(2m+M)R^2 + 2J_w + 2n^2J_m & MLR - 2n^2J_m\\ MLR - 2n^2J_m & ML^2 + J_\psi + 2n^2J_m\end{bmatrix}$$

$$F = 2\begin{bmatrix}\beta+f_w & -\beta\\ -\beta & \beta\end{bmatrix}$$

$$G = \begin{bmatrix}0 & 0\\ 0 & -MgL\end{bmatrix}$$

$$H = \begin{bmatrix}\alpha & \alpha\\ -\alpha & -\alpha\end{bmatrix}$$

$$I\ddot{\phi} + J\dot{\phi} = K(v_r - v_l) \tag{3.30}$$

$$I = \frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2J_m\right)$$

$$J = \frac{W^2}{2R^2}\left(\beta+f_w\right)$$

$$K = \frac{W}{2R}\alpha$$

Here we consider the following variables $\mathbf{x}_1, \mathbf{x}_2$ as state, and $\mathbf{u}$ as input. $\mathbf{x}^T$ indicates transpose of $\mathbf{x}$.

$$\mathbf{x}_1 = \begin{bmatrix} \theta, & \psi, & \dot{\theta}, & \dot{\psi} \end{bmatrix}^T, \quad \mathbf{x}_2 = \begin{bmatrix} \phi, & \dot{\phi} \end{bmatrix}^T, \quad \mathbf{u} = \begin{bmatrix} v_l, & v_r \end{bmatrix}^T \tag{3.31}$$

Consequently, we can derive state equations of two-wheeled inverted pendulum from Eq. (3.29) and Eq. (3.30).

$$\dot{\mathbf{x}}_1 = A_1 \mathbf{x}_1 + B_1 \mathbf{u} \tag{3.32}$$

$$\dot{\mathbf{x}}_2 = A_2 \mathbf{x}_2 + B_2 \mathbf{u} \tag{3.33}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_1(3,2) & A_1(3,3) & A_1(3,4) \\ 0 & A_1(4,2) & A_1(4,3) & A_1(4,4) \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_1(3) & B_1(3) \\ B_1(4) & B_1(4) \end{bmatrix} \tag{3.34}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -J/I \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ -K/I & K/I \end{bmatrix} \tag{3.35}$$

$$A_1(3,2) = -gMLE(1,2)/\det(E)$$
$$A_1(4,2) = gMLE(1,1)/\det(E)$$
$$A_1(3,3) = -2\left[(\beta + f_w)E(2,2) + \beta E(1,2)\right]/\det(E)$$
$$A_1(4,3) = 2\left[(\beta + f_w)E(1,2) + \beta E(1,1)\right]/\det(E)$$
$$A_1(3,4) = 2\beta\left[E(2,2) + E(1,2)\right]/\det(E)$$
$$A_1(4,4) = -2\beta\left[E(1,1) + E(1,2)\right]/\det(E)$$
$$B_1(3) = \alpha\left[E(2,2) + E(1,2)\right]/\det(E)$$
$$B_1(4) = -\alpha\left[E(1,1) + E(1,2)\right]/\det(E)$$
$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

ANNEXE D – Modèle Simulink du robot lego NXT

tiré de la notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.

5 NXTway-GS Model

This chapter describes summary of NXTway-GS model and parameter files.

5.1 Model Summary

The nxtway_gs.mdl and nxtway_gs_vr.mdl are models of NXTway-GS control system. Both models are identical but different in point of including 3D viewer provided by Virtual Reality Toolbox.

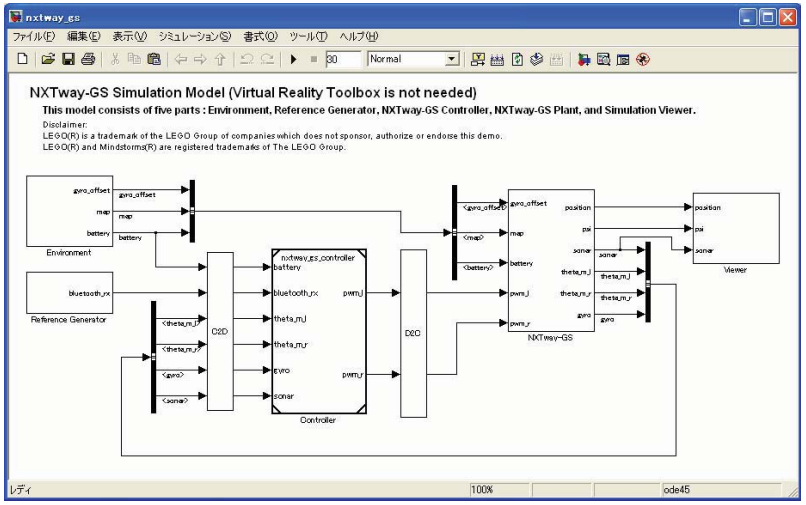


Figure 5-1 nxtway_gs.mdl

Main parts of nxtway_gs.mdl and nxtway_gs_vr.mdl are as follows.

Environment

This subsystem defines environmental parameters. For example, map data, gyro offset value, and so on.

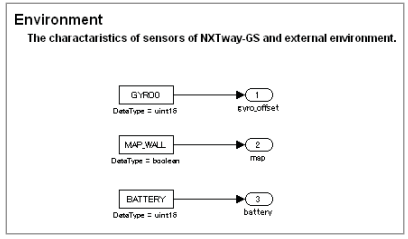


Figure 5-2 Environment subsystem

Reference Generator

This subsystem is a reference signal generator for NXTway-GS. We can change the speed reference and the rotation speed reference by using Signal Builder block. The output signal is a 32 byte data which has dummy data to satisfy the NXT GamePad utility specification.

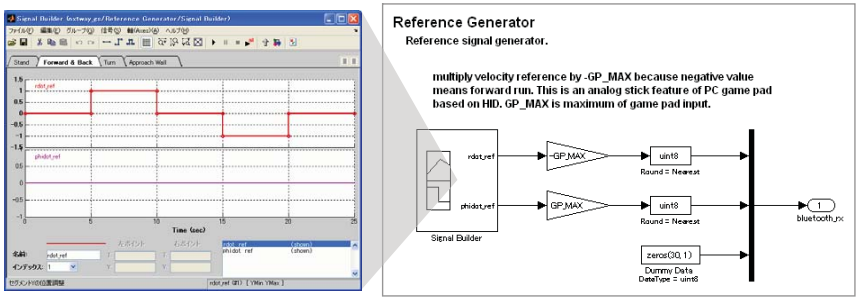


Figure 5-3 Reference Generator subsystem

Figure 5-4 shows a relation between speed and rotation speed reference value and PC game pad analog sticks.

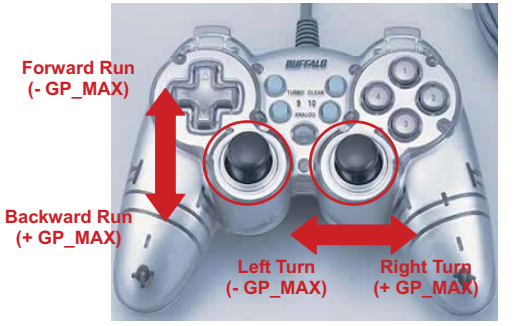


Figure 5-4 Inputs from PC game pad analog sticks

Controller

This block is NXTway-GS digital controller and references nxtway_gs_controller.mdl with Model block. Refer 7 Controller Model (Single Precision Floating-Point Arithmetic) for more details.

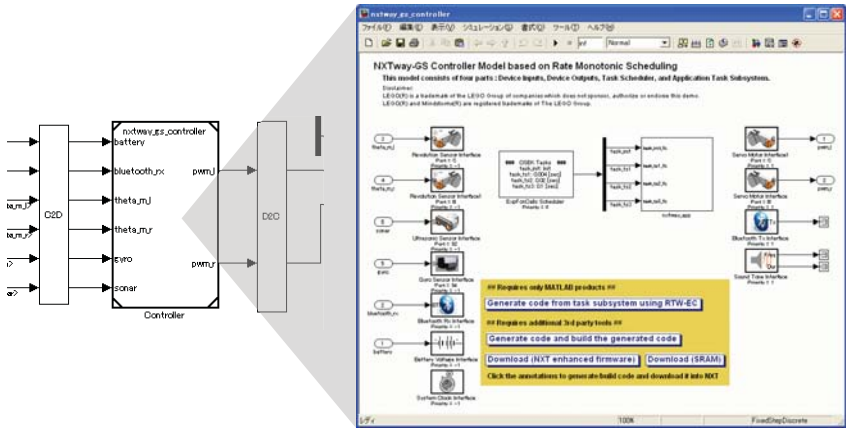


Figure 5-5 Controller block (nxtway_gs_controller.mdl)

The Controller block runs in discrete-time (base sample time = 1 [ms]) and the plant (NXTway-GS subsystem) runs in continuous-time (sample time = 0 [s]). Therefore it is necessary to convert continuous-time to discrete-time and vice versa by inserting Rate Transition blocks between them properly.

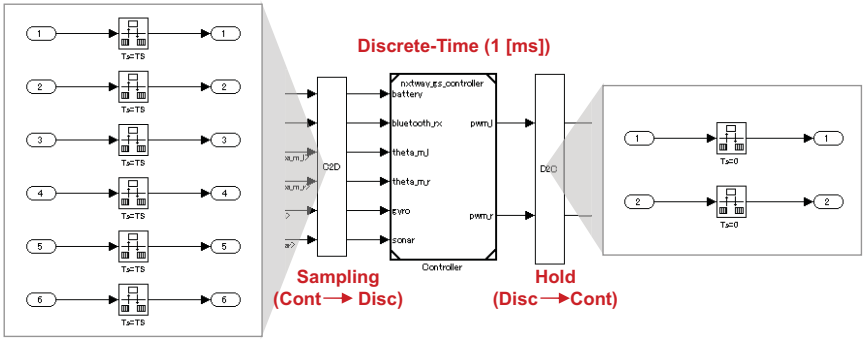


Figure 5-6 Rate conversions between the controller and the plant

NXTway-GS

This subsystem is mathematical model of NXTway-GS. It consists of sensor, actuator, and linear plant model. The plant references nxtway_gs_plant.mdl with Model block. Refer 6 Plant Model for more details.

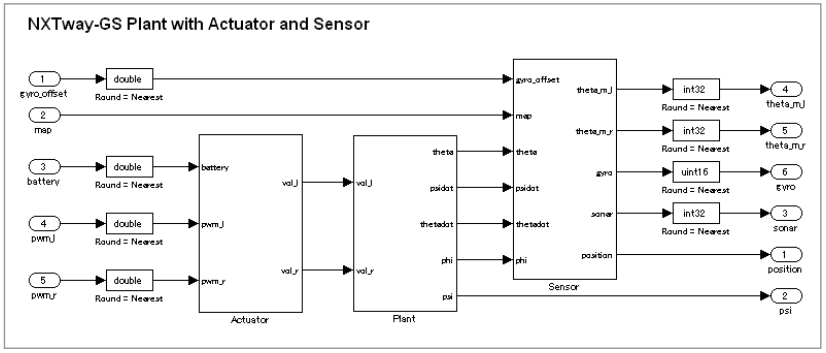


Figure 5-7 NXTway-GS subsystem

Viewer

This subsystem includes simulation viewers. nxtway_gs.mdl includes a position viewer with XY Graph block, and nxtway_gs_vr.mdl does 3D viewer provided by Virtual Reality Toolbox.

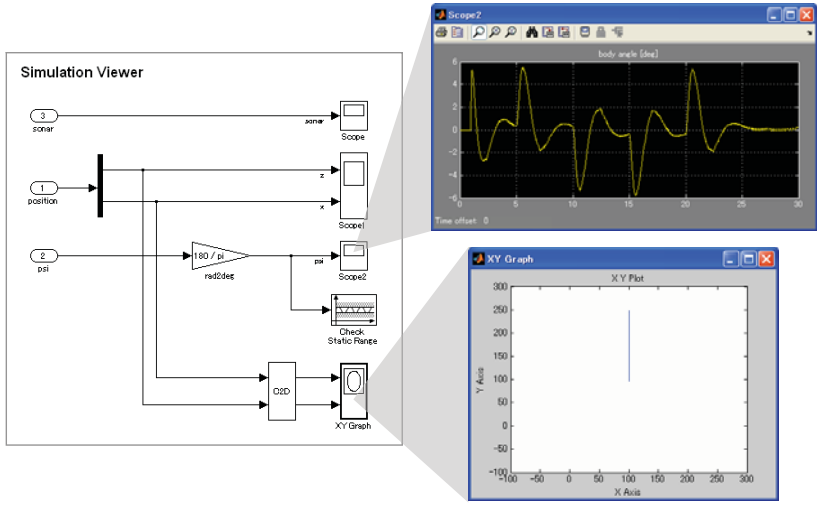


Figure 5-8 Viewer subsystem (nxtway_gs.mdl)

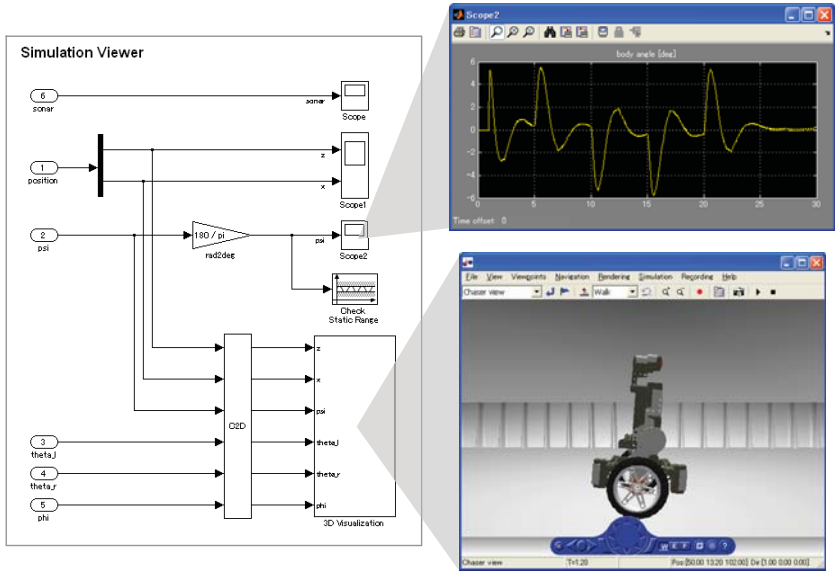


Figure 5-9 Viewer subsystem (nxtway_gs_vr.mdl)

5.2 Parameter Files

Table 5-1 shows parameter files for simulation and code generation.

Table 5-1 Parameter files

File Name	Description
param_controller.m	M-script for controller parameters
param_controller_fixpt.m	M-script for fixed-point settings (Simulink.NumericType)
param_nxtway_gs.m	M-script for NXTway-GS parameters (It calls param_***.m)
param_plant.m	M-script for plant parameters
param_sim.m	M-script for simulation parameters

param_nxtway_gs.m calls param_***.m (*** indicates controller, plant, sim) and creates all parameters in base workspace. Model callback function is used to run param_nxtway_gs.m automatically when the model is loaded.

To display model callback function, choose [**Model Properties**] from the Simulink [**File**] menu.

6 Plant Model

This chapter describes NXTway-GS subsystem in nxtway_gs.mdl / nxtway_gs_vr.mdl.

6.1 Model Summary

The NXTway-GS subsystem consists of sensors, actuators, and linear plant model. It converts the data type of input signals to double, calculates plant dynamics using double precision floating-point arithmetic, and outputs the results after quantization.

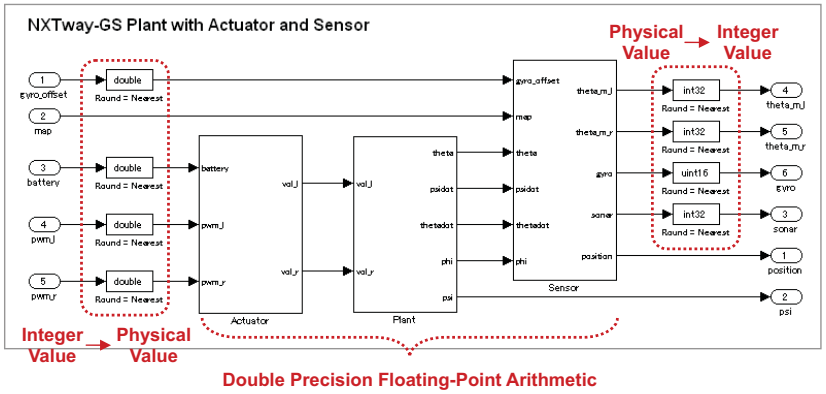


Figure 6-1 NXTway-GS subsystem

6.2 Actuator

Actuator subsystem calculates the DC motor voltage by using PWM duty derived from the controller. Considering the coulomb and viscous friction in the driveline, we model it as a dead zone before calculating the voltage.

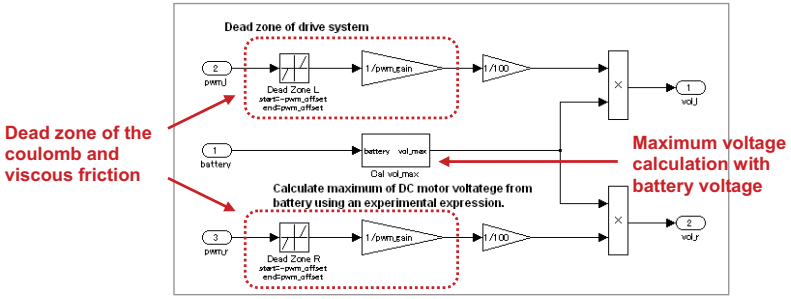


Figure 6-2 Actuator subsystem

DC motor maximum voltage is necessary to calculate PWM duty. In the Cal vol_max subsystem, we use the following experimental equation which is a conversion rule between the battery voltage V_B and the maximum DC motor voltage $V_{\max}$.

$$V_{\max} = 0.001089 \times V_B - 0.625 \tag{6.1}$$

We have derived Eq. (6.1) as the following. Generally speaking, DC motor voltage and rotation speed are proportional to battery voltage. Figure 6-3 shows an experimental result of battery voltage and motor rotation speed at PWM = 100% with no load.

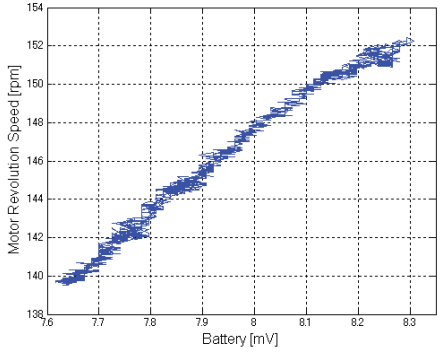


Figure 6-3 Experimental result of battery voltage and motor rotation speed at PWM = 100%

Eq. (6.1) is derived by relating Figure 6-3 and the data described in the reference [1].

6.3 Plant

nxtway_gs_plant.mdl is a model of the state equation derived in 3.3 State Equations of Two-Wheeled Inverted Pendulum. It is referenced in Enabled Subsystem with Model block. Considering the calibration of gyro offset, the plant is modeled so as to start calculation after gyro calibration. Refer 7.1 Control Program Summary for more details about the gyro calibration.

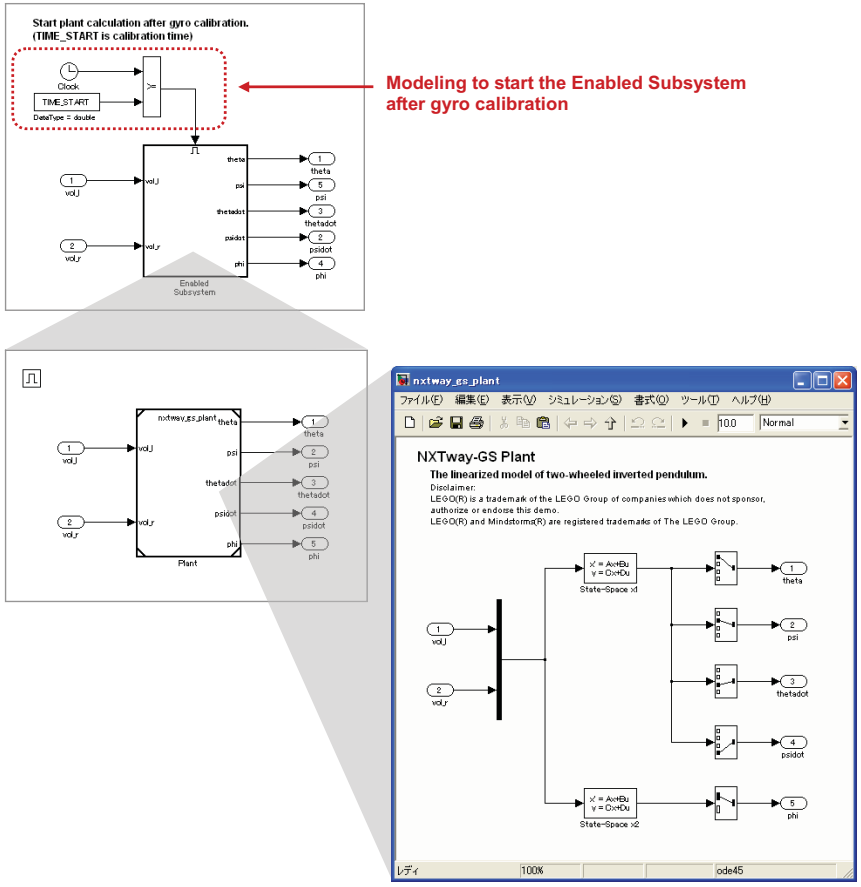


Figure 6-4 Plant subsystem

6.4 Sensor

Sensor subsystem converts the values calculated in plant subsystem to several sensor outputs. Because the computation cost of distance (an output of ultrasonic sensor) calculation and wall hit detection are heavy, we cut the computation steps by inserting Rate Transition blocks. Refer Appendix B.3 for more details about the distance calculation and wall hit detection.

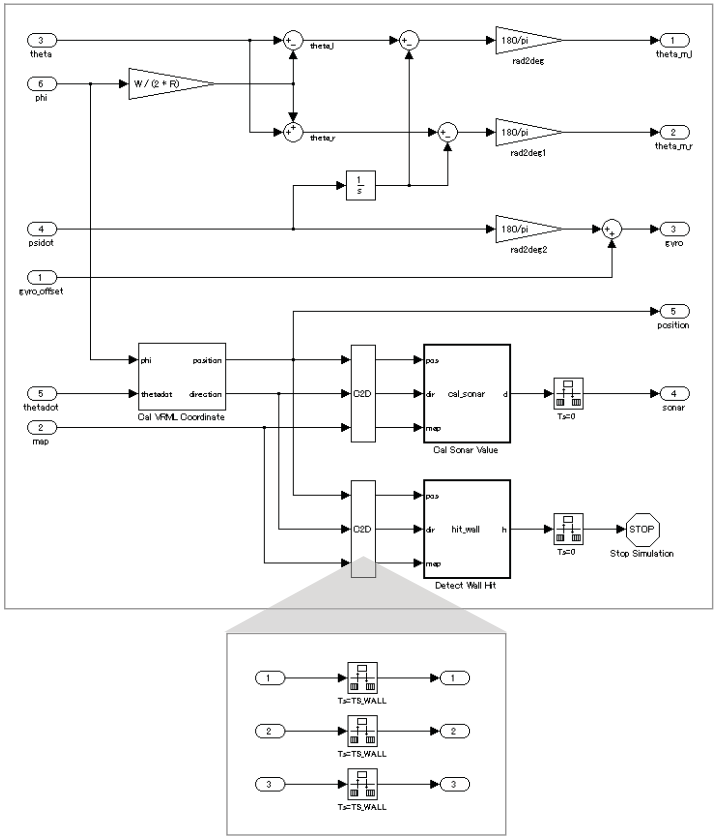


Figure 6-5 Sensor subsystem